

信息学奥赛

一本通

初赛真题解析

◆ 一本通信息学名师工作室 编著

真题提炼　专题突破　老题新颜　剖析全面

南京大学出版社

信息学奥赛一本通
初赛真题解析

一本通信息学名师工作室/编著

主　任

董永建

副主任

舒春平

主　编

陈文超　马建辉　严科磊　王　旭

编　委

徐斌立　吕振锋　张义丰　郑高枝

赵玉喜　李艳玲　李　洋　高亮亮

南京大学出版社

图书在版编目(CIP)数据

信息学奥赛一本通.初赛真题解析/一本通信息学名师工作室编著.--南京:南京大学出版社,2022.7(2023.3重印)
ISBN 978-7-305-25726-1

Ⅰ.①信… Ⅱ.①一… Ⅲ.①程序设计—中小学—教学参考资料 Ⅳ.①G634.673

中国版本图书馆CIP数据核字(2022)第081050号

出版发行 南京大学出版社
社　　址 南京市汉口路22号　　邮 编 210093
出 版 人 金鑫荣

书　　名 信息学奥赛一本通·初赛真题解析
编　　著 一本通信息学名师工作室
责任编辑 纪玉媛　　编辑热线 0571－88191227

印　　刷 杭州富阳美术印刷有限公司
开　　本 787×1092　1/16　　印　张 26.25　　字　数 700千
版　　次 2022年7月第1版　2023年3月第2次印刷
ISBN 978-7-305-25726-1
定　　价 79.80元

网　　址 http://www.njupco.com
官方微博 http://weibo.com/njupco
官方微信 njupress
销售咨询热线 0571－88256393　89908120

* 版权所有,侵权必究
* 凡购买南大版图书,如有印装质量问题,请与所购图书销售部门联系调换。

前　言

本书的主编和编者由多年从事信息学奥赛辅导工作的一线教师组成，他们具备较高的专业水平和丰富的教学经验、竞赛辅导经验，因此，能够很好地满足中小学编程学习者和爱好者的实际需求。本书紧密围绕“程序＝算法＋数据结构”这一核心思想，从每个初赛真题的题义分析到算法设计，从朴素算法应用到算法优化、比较分析，从变量的解释到数据结构的应用等多个角度以及维度突破了信息学的各知识重点和难点，让读者全方位地体会到“程序是怎样炼成的”。

按照知识点对历年真题进行分门别类、专题突破、超详解析、题型改编是本书的重要特点，前三章分别对计算机基础知识、程序设计基础知识和数学知识进行了梳理和讲解，每个章节后配套有课堂练习，练习的例题均遴选自 2008 年以来最具代表性的 NOIP 初赛真题。第四、五章围绕阅读程序和完善程序两类题型进行组织编写，覆盖数据结构、算法和数论等知识点，例题分析均包含程序整体功能介绍以及每小题的详细解析，降低了学习难度。自 2019 年以来初赛的题型只有选择题，因此本书对这之前的真题做了题型的改编，以紧贴时代脉搏，提高学习效果。第六、七章汇编了 2019 至 2021 最近三年的入门级、提高级六套真题、参考答案及详细解析，能满足不同层次学生的学习需求。

本书是一本通信息学名师工作室教师集体智慧的结晶，凝聚了多位老师的心血。本书由董永建老师担任主任，舒春平老师担任副主任，由长乐第一中学董永建、温州中学舒春平、深圳外国语学校陈文超、邯郸市第一中学马建辉、宁波四眼碶小学严科磊、苏州黄埭中学王旭负责编写，具体分工如下：第一章由王旭负责编写，第二章由严科磊负责编写，第三章由马建辉负责编写，第四、五章由陈文超负责编写，第六章由董永建负责编写，第七章由舒春平负责编写。全书由董永建、陈文超统稿，马建辉、严科磊、王旭负责本书课件以及相关教学资源的制作。

在本书的撰写过程中，编者还参考了一些网络上的解题思路，在此对各位原创者一并表示感谢。

由于时间仓促、水平有限，书中不足之处在所难免，恳请广大读者指正，请通过指导教师或邮件 fjclyz2006@163.com 向编著者反馈错误和修订建议。

编　者

2022 年 5 月

目　　录

第一章　计算机基础知识 …… 1

第 1 节　计算机概述 …… 1
第 2 节　计算机系统的基本结构 …… 6
第 3 节　计算机软件系统 …… 14
第 4 节　计算机语言 …… 19
第 5 节　数制转换 …… 23
第 6 节　信息编码表示 …… 29
第 7 节　原码、反码和补码 …… 34
第 8 节　计算机网络 …… 38

第二章　程序设计基础知识 …… 50

第 1 节　程序基本常识 …… 50
第 2 节　C＋＋语言基础 …… 54
第 3 节　排序算法 …… 63
第 4 节　基础算法 …… 67
第 5 节　字符数组与字符串 …… 75
第 6 节　链表 …… 78
第 7 节　栈和队列 …… 83
第 8 节　树 …… 90
第 9 节　图 …… 103

第三章　数学问题 …… 114

第 1 节　简单数论 …… 114
第 2 节　排列组合 …… 125
第 3 节　特殊数列 …… 132
第 4 节　容斥原理 …… 136
第 5 节　抽屉原理 …… 140
第 6 节　概率与期望 …… 142
第 7 节　离散数学 …… 145
第 8 节　博弈论入门 …… 150

第四章　阅读程序 …… 151

第 1 节　模拟策略 …… 151
第 2 节　字符处理 …… 163

第 3 节　枚举算法 …… 177
第 4 节　排序算法 …… 186
第 5 节　递推与递归 …… 194
第 6 节　指针和引用 …… 205
第 7 节　搜索算法 …… 208
第 8 节　动态规划 …… 211
第 9 节　树和图 …… 214
第 10 节　数论 …… 225

第五章　完善程序 …… 242

第 1 节　数组 …… 242
第 2 节　字符处理 …… 247
第 3 节　枚举算法 …… 252
第 4 节　链表 …… 260
第 5 节　分治算法 …… 264
第 6 节　搜索算法 …… 275
第 7 节　动态规划 …… 283
第 8 节　栈和队列 …… 290
第 9 节　树和图 …… 296
第 10 节　数论 …… 306

第六章　2019—2022 年 CSP－J 入门级初赛真题及答案解析 …… 312

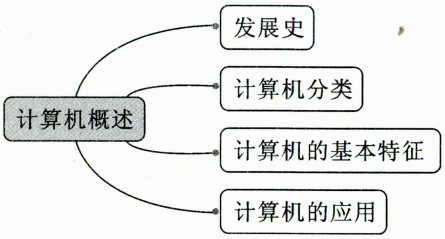
2019 CCF 非专业级别软件能力认证第一轮(CSP－J)入门级 C++语言试题 …… 312
2020 CCF 非专业级别软件能力认证第一轮(CSP－J)入门级 C++语言试题 …… 319
2021 CCF 非专业级别软件能力认证第一轮(CSP－J)入门级 C++语言试题 …… 327
2022 CCF 非专业级别软件能力认证第一轮(CSP－J)入门级 C++语言试题 …… 336
答案解析 …… 345

第七章　2019—2022 年 CSP－S 提高级初赛真题及答案解析 …… 359

2019 CCF 非专业级别软件能力认证第一轮(CSP－S)提高级 C++语言试题 …… 359
2020 CCF 非专业级别软件能力认证第一轮(CSP－S)提高级 C++语言试题 …… 367
2021 CCF 非专业级别软件能力认证第一轮(CSP－S)提高级 C++语言试题 …… 378
2022 CCF 非专业级别软件能力认证第一轮(CSP－S)提高级 C++语言试题 …… 390
答案解析 …… 399

第一章　计算机基础知识

第 1 节　计算机概述



一、发展史

1.计算机发展的五个阶段划分：

代别	年代	逻辑(电子)元件	应用范围
第一代	1946—1958	真空电子管	科学计算、军事研究
第二代	1959—1964	晶体管	数据处理、事物处理
第三代	1965—1970	集成电路	工业控制的各个领域
第四代	1971—至今	大规模、超大规模集成电路	各个领域
第五代	现代	智能计算机系统	人工智能

2.1946 年，世界上第一台电子计算机 ENIAC 在美国宾夕法尼亚大学诞生。

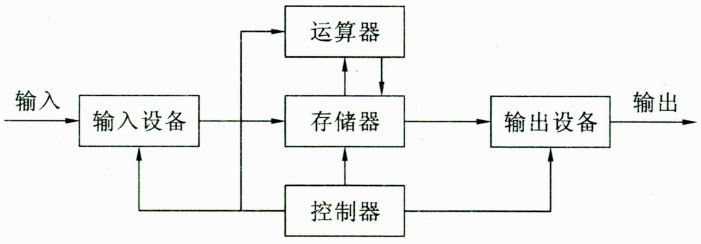
3.计算机界的两位重要人物：

(1)冯·诺伊曼：被称为“计算机之父”，提出了计算机体系结构；

(2)图灵：被称为“人工智能之父”，提出了一种判定机器是否具有智能的试验方法，即图灵试验。此外，图灵提出的著名的图灵机模型为现代计算机的逻辑工作方式奠定了基础。

4.冯·诺伊曼式的计算机体系结构

冯·诺伊曼提出计算机硬件设备由存储器、运算器、控制器、输入设备和输出设备五部分组成(如下图所示)，并且由总线连接。总线包括地址总线、数据总线、控制总线。



冯·诺伊曼的存储程序思想：把计算过程描述为由许多命令按一定顺序组成的程序，然后把程序和数据一起输入计算机，计算机对已存入的程序和数据处理后，输出结果。

二、计算机的分类

1.按规模分：巨型、大型、中型、小型、微型计算机。其中，微型计算机的发展已经历经五代，学校和家庭使用的计算机都是微型计算机，简称微机，又称个人计算机或 PC 机。

2.按用途分：专业计算机、通用计算机。

3.按原理分：模拟计算机、数字计算机。因为计算机在发送端通过调制将数字信号转换为模拟信号，而在接收端通过解调再将模拟信号转换为数字信号。

4.按功能分：单媒体计算机、多媒体计算机。只能处理文字和数字，就是单媒体计算机。不仅能处理文字和数字，而且还能处理图像、文本、音频、视频等多种媒介，就是多媒体计算机。

三、计算机的基本特征

1.运算速度快：可以高速准确地完成各种算术运算；

2.计算精度高：可以有十几位甚至几十位（二进制）有效数字，计算精度可由千分之几到百万分之几；

3.存储能力超强：可以存储大量信息；

4.逻辑运算能力强：能对信息进行比较和判断；

5.自动控制能力强：在程序控制下，可以连续、自动地工作，不需要人的干预。

四、计算机的应用

1.数值计算：是计算机应用的一个基本方面，也是计算机最早应用的领域；

2.数据和信息处理：是目前计算机最广泛的应用领域；

3.过程控制：利用计算机进行生产过程、实时过程的控制；

4.计算机辅助系统：指利用计算机帮助人们完成各种任务，包括 CAD（计算机辅助设计）、CAM（计算机辅助制造）、CAI（计算机辅助教学）、CAT（计算机辅助测试）等。

5.人工智能：用于各种专家系统和机器人构造等。



课堂练习

1.【NOIP2008】在下列关于图灵奖的说法中，不正确的是（　　）。

A. 图灵奖是美国计算机协会于 1966 年设立的，专门奖励那些对计算机事业作出重要贡献的个人

B. 图灵奖有“计算机界诺贝尔奖”之称

C. 迄今为止，还没有华裔计算机科学家获此殊荣

D. 图灵奖的名称取自计算机科学的先驱、英国科学家阿兰·图灵

【答案】C

【分析】图灵奖（A. M. Turing Award），由美国计算机协会（ACM）于 1966 设立，有“计算机界诺贝尔奖”之称，其名称取自计算机科学的先驱、英国科学家阿兰·麦席森·图灵。获奖者的贡献必须是在计算机领域具有持久而重大的技术先进性，故选项 A、B、D 正确。2000 年华裔计算机科学家姚期智由于在计算理论方面的贡献而获奖，故选项 C 不正确。

2.【NOIP2009】关于图灵机，下面的说法正确的是（　　）。

A. 图灵机是世界上最早的电子计算机

B. 由于大量使用磁带操作，图灵机运行速度很慢

C. 图灵机是英国人图灵发明的，在二战中为破译德军的密码发挥了重要作用

D. 图灵机只是一个理论上的计算模型

【答案】D

【分析】最早的电子计算机是ENIAC；图灵机是计算机模型，没有运行速度，更谈不上磁带操作；图灵本人在二战中破译德军密码系统中发挥了重要作用，而不是图灵机发挥作用；图灵的基本思想是用机器来模拟人们用纸笔进行数学运算的过程，而假想出来的一种抽象机器。

3.【NOIP2017】以下和计算机领域密切相关的奖项是(　　)。

A. 奥斯卡奖　　B. 图灵奖　　C. 诺贝尔奖　　D. 普利策奖

【答案】B

【分析】常识题。奥斯卡是电影类的奖项。诺贝尔有六种奖项：物理、化学、生物和医学、文学、经济、和平，没有计算机。普利策是新闻类的奖项。图灵奖是计算机领域被广泛认可的奖项。

4.【NOIP2010】提出“存储程序”的计算机工作原理的是(　　)。

A. 克劳德·香农　　B. 戈登·摩尔

C. 查尔斯·巴比奇　　D. 冯·诺伊曼

【答案】D

【分析】克劳德·艾尔伍德·香农是美国数学家、信息论的创始人。戈登·摩尔提出了摩尔定律。摩尔定律是指IC上可容纳的晶体管数目，约每隔18个月便会增加一倍，性能也将提升一倍。查尔斯·巴比奇是科学管理的先驱者，是第一台可编程的机械计算机的设计者。美籍匈牙利科学家冯·诺伊曼最先提出存储程序的思想，因此冯·诺伊曼又被称为“计算机之父”。

5.【NOIP2011】摩尔定律(Moore'slaw)是由英特尔创始人之一戈登·摩尔(Gordon Moore)提出来的。根据摩尔定律，在过去几十年以及在可预测的未来几年，单块集成电路的集成度大约每(　　)个月翻一番。

A. 1　　B. 6　　C. 18　　D. 36

【答案】C

【分析】摩尔定律是由英特尔(Intel)创始人之一戈登·摩尔(Gordon Moore)提出来的。其内容为：当价格不变时，集成电路上可容纳的元器件的数目，约每隔18—24个月便会增加一倍，性能也将提升一倍。换言之，每一美元所能买到的电脑性能，将每隔18—24个月翻一倍以上。这一定律揭示了信息技术进步的速度。

6.【NOIP2011】1956年(　　)授予肖克利(William Shockley)、巴丁(John Bardeen)和布拉顿(Walter Brattain)，以表彰他们对半导体的研究和晶体管效应的发现。

A. 诺贝尔物理学奖　　B. 约翰·冯·诺伊曼奖

C. 图灵奖　　D. 高德纳奖(Donald E. Knuth Prize)

【答案】A

【分析】威廉·肖克利出生于英国伦敦，后迁往美国加州，物理学家。因对半导体的研究和发现了晶体管效应，与巴丁和布拉顿分享了1956年度的诺贝尔物理学奖。

7.【NOIP2014】计算机界的最高奖是(　　)。

A. 菲尔兹奖　　B. 诺贝尔奖　　C. 图灵奖　　D. 普利策奖

【答案】C

【分析】菲尔兹奖是数学领域的奖项,普利策奖是新闻领域的奖项,诺贝尔奖只设有物理、化学、生物或医学、文学、和平、经济,没有计算机,而图灵奖正是计算机界的最高奖。

8.【NOIP2013】1948 年,(　　)将热力学中的熵引入信息通信领域,标志着信息论研究的开端。

A. 冯·诺伊曼(John von Neumann)　　B. 图灵(AlanTuring)

C. 欧拉(Leonhard Euler)　　D. 克劳德·香农(Claude Shannon)

【答案】D

【分析】克劳德·香农(Claude Shannon)在 1948 年发表论文《A Mathematical Theory of Communication》将热力学中的熵引入信息通信领域,标志着信息论研究的开端。

9.【NOIP2011】从 ENIAC 到当前最先进的计算机,冯·诺伊曼体系结构始终占有重要的地位。冯·诺伊曼体系结构的核心内容是(　　)。

A. 采用开关电路　　B. 采用半导体器件

C. 采用存储程序和程序控制原理　　D. 采用键盘输入

【答案】C

【分析】冯·诺伊曼于 1946 年提出存储程序原理,冯·诺伊曼体系结构的理论要点是:数字计算机的数制采用二进制;计算机应该按照程序顺序执行。人们把冯·诺伊曼的这个理论称为冯·诺伊曼体系结构。

10.【NOIP2012】1946 年诞生于美国宾夕法尼亚大学的 ENIAC 属于(　　)计算机。

A. 电子管　　B. 晶体管

C. 集成电路　　D. 超大规模集成电路

【答案】A

【分析】按照计算机的发展历程,诞生于 1946 年的 ENIAC 处于第一阶段的电子管计算机时代。

11.【NOIP2017】计算机应用的最早领域是(　　)。

A. 数值计算　　B. 人工智能

C. 机器人　　D. 过程控制

【答案】A

【分析】考查计算机发展历史,计算机最早应用的领域正是数值计算,世界上最早的计算机——美国的 ENIAC 的设计初衷是进行数值分析,计算导弹弹道轨迹。

12.【NOIP2013】在 Windows 资源管理器中,用鼠标右键单击一个文件时,会出现一个名为“复制”的操作选项,它的意思是(　　)。

A. 用剪切板中的文件替换该文件

B. 在该文件所在文件夹中,将该文件克隆一份

C. 将该文件复制到剪切板,并保留原文件

D. 将该文件复制到剪切板,并删除原文件

【答案】C

【分析】Windows 资源管理器中,复制的作用是将文件复制到剪切板,同时保留原文件。

不定项选择题

1.【NOIP2008】在下列关于图灵奖的说法中，正确的有(　　)。

A. 图灵奖是美国计算机协会于 1966 年设立的，专门奖励那些对计算机事业作出重要贡献的个人

B. 图灵奖有“计算机界诺贝尔奖”之称

C. 迄今为止，还没有华裔计算机科学家获此殊荣

D. 图灵奖的名称取自计算机科学的先驱、英国科学家阿兰·图灵

【答案】ABD

【分析】图灵奖(A .M. Turing Award)，由美国计算机协会(ACM)于 1966 设立，又叫“A. M.图灵奖”，有“计算机界诺贝尔奖”之称，专门奖励那些对计算机事业作出重要贡献的个人。其名称取自计算机科学的先驱、英国科学家阿兰·麦席森·图灵。获奖者的贡献必须在计算机领域具有持久而重大的技术先进性。大多数获奖者是计算机科学家。2000 年姚期智(Andxew Chi-Chih Yao)获奖原因：由于在计算理论方面的贡献而获奖，包括伪随机数的生成算法、加密算法和通讯复杂性。英国著名学者阿兰·麦席森·图灵(Alan Mathison Turing)不仅以“纸上下棋机”率先探讨了下棋与机器智能的联系，他还是举世公认的“人工智能之父”。图灵不仅是“人工智能之父”，他也是“计算机之父”。

2.【NOIP2017】以下和计算机领域密切相关的奖项有(　　)。

A. 奥斯卡奖　　B. 图灵奖　　C. 诺贝尔奖　　D. 王选奖

【答案】BD

【分析】图灵奖是计算机领域最为重要的奖项之一。中国计算机学会王选奖是中国计算机学会设立的奖项，为了纪念王选院士为中国计算机事业做出的非凡贡献，学习他严谨、务实、奉献、创新、勇于超越的科研精神。该奖授予在计算机科学技术前沿取得重要突破，研究成果通过转化和产业化，创造显著经济或社会效益的科技工作者。中国计算机学会王选奖的前身中国计算机学会创新奖于 2005 年设立。2006 年更名为中国计算机学会王选奖。2007 年 12 月方正集团出资 100 万元人民币作为“中国计算机学会王选奖”基金。根据中国计算机学会官网显示，中国计算机学会王选奖每年评选一次，获奖人数不超过两名，其中一名奖励学术，一名奖励技术或工程。截至 2018 年 10 月，该奖共评选了 9 次，共有 17 人获得该奖项。奥斯卡奖是电影界的奖项。诺贝尔奖未设立计算机相关奖项。

3.【NOIP2018】下列关于图灵奖的说法中，正确的有(　　)。

A. 图灵奖是由电气和电子工程师协会(IEEE)设立的

B. 目前获得该奖项的华人学者只有姚期智教授一人

C. 其名称取自计算机科学的先驱、英国科学家艾伦·麦席森·图灵

D. 它是计算机界最负盛名、最崇高的一个奖项，有“计算机界的诺贝尔奖”之称

【答案】BCD

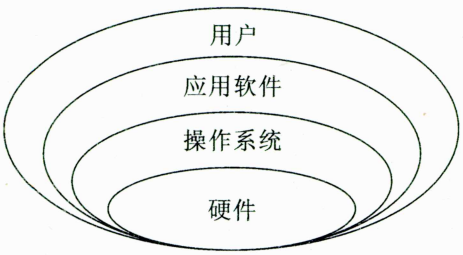
【分析】图灵奖是由美国计算机协会(ACM)创办的。1993 年，姚期智最先提出量子通信复杂性，基本上完成了量子计算机的理论基础。1995 年，提出分布式量子计算模式，后来成为分布式量子算法和量子通讯协议安全性的基础。因为对计算理论包括伪随机数生成、密码学与通信复杂度的突出贡献，美国计算机协会(ACM)把 2000 年度的图灵奖授予他。

第2节 计算机系统的基本结构

一、概述

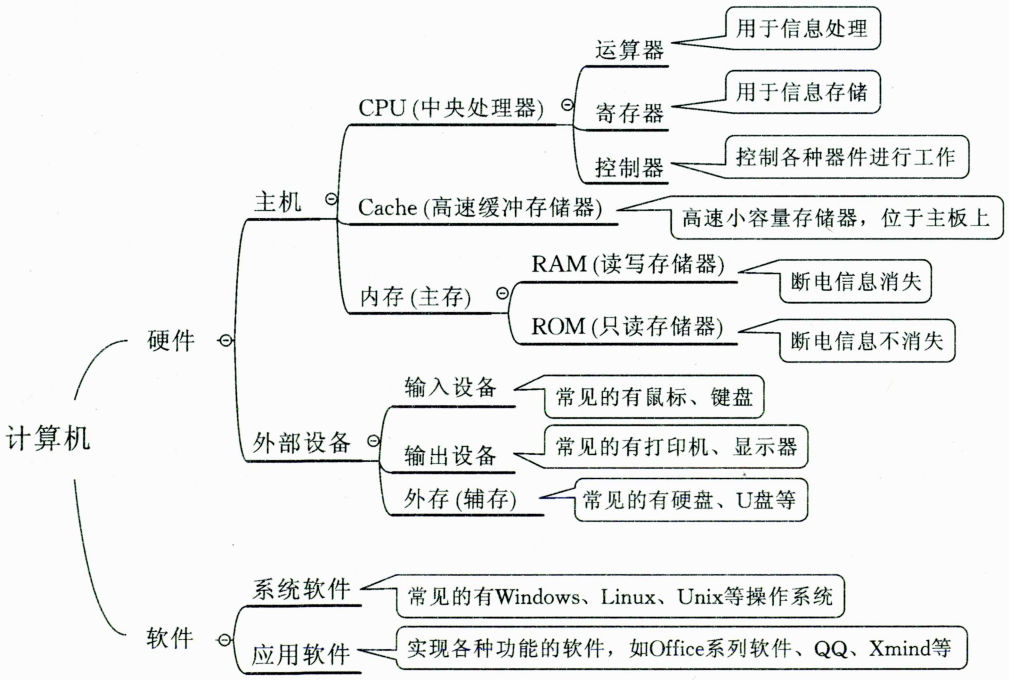
计算机系统由硬件和软件两部分组成。硬件系统是计算机的“躯干”，是物质基础。而软件系统则是建立在这个“躯干”上的“灵魂”，没有安装任何软件的计算机称为裸机。

1.宏观角度



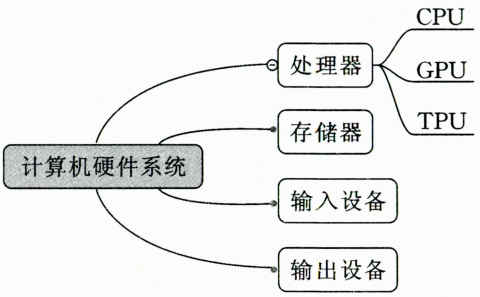
从宏观角度来看，计算机最底层的基础是各种硬件设备，然后在裸机上安装操作系统，之后再在操作系统下安装各种应用软件，最后用户只需使用各种应用软件即可。

2.微观角度



从微观角度来看，计算机组成及各部分的功能如上图所示。值得注意的是外部设备与主机之间、主机中的各个部件之间的存储速度不一样，存储速度由慢到快依次是外存、内存、Cache、CPU。可见CPU的存储速度是最快的。

二、计算机硬件系统



计算机硬件系统由五大部分组成：运算器、控制器、存储器、输入设备、输出设备。

（一）处理器

1.中央处理器（CPU——Central Processing Unit）

由运算器、控制器和一些寄存器组成；

运算器进行各种算术运算和逻辑运算；

控制器是计算机的指挥系统；

CPU 的主要性能指标是主频和字长。

2.图形处理器（GPU——Graphics Processing Unit）

图形处理器（GPU），又称显示核心、视觉处理器、显示芯片，是一种专门在个人电脑、工作站、游戏机和一些移动设备（如平板电脑、智能手机等）上做图像和图形相关运算工作的微处理器。

3.张量处理单元（TPU——Tensor Processing Unit）

张量处理单元（TPU），是一款为机器学习而定制的芯片，经过了专门深度机器学习方面的训练，具有更高效能（每瓦计算能力）。

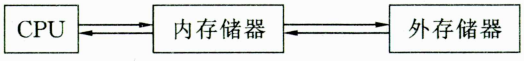
（二）存储器

存储器的主要功能是用来保存各类程序的数据信息。

存储器可分为主存储器和辅助存储器两类。

1.主存储器（也称为内存储器），属于主机的一部分。用于存放系统当前正在执行的数据和程序，属于临时存储器。

2.辅助存储器（也称外存储器），它属于外部设备。用于存放暂不用的数据和程序，属于永久存储器。存储器与 CPU 的关系表示如下：

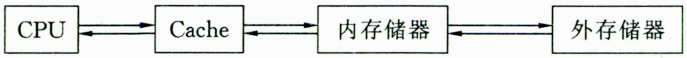


（1）内存储器

内存又称为主存，它和 CPU 一起构成了计算机的主机部分，并且存储的信息可以被 CPU 直接访问。内存由半导体存储器组成，存取速度较快，但一般容量较小。内存中含有很多的存储单元，每个单元可以存放 1 个 8 位的二进制数，即 1 个字节（Byte，简称“B”）。内存中的每个字节各有一个固定的编号，这个编号称为地址。CPU 在存取存储器中的数据时是按地址进行的。所谓存储器容量即指存储器中所包含的字节数，通常用 KB、MB、GB、TB 和 PB 作为存储器的容量单位。它们之间的关系为：

1KB＝1024B　1MB＝1024KB　1GB＝1024MB　1TB＝1024GB　1PB＝1024TB

内存储器通常可以分为随机存储器 RAM、只读存储器 ROM 和高速缓冲存储器 Cache 三种。



(2)外存储器

外存储器又称为辅助存储器,它的容量一般都比较大,而且大部分可以移动,便于在不同计算机之间进行信息交流。在微型计算机中,常用的外存有硬盘、闪存和光盘几种。

①固态硬盘存储器

固态硬盘在接口的规范和定义、功能及使用方法上与普通硬盘完全相同,在产品外形和尺寸上也基本与普通硬盘一致。但其采用闪存作为存储介质,读取速度相对机械硬盘更快。

②机械硬盘存储器

机械硬盘存储器由电机和硬盘组成,一般置于主机箱内。机械硬盘是涂有磁性材料的磁盘组件,用于存放数据。它是一个非常精密的机械装置,磁道间只有百万分之几英寸的间隙,磁头传动装置必须把磁头快速且准确地移到指定的磁道上。

③闪存

闪存又名优盘,是在存储速度与容量上介于软盘与硬盘之间的一种外部存储器。

④光盘

光盘的存储介质不同于磁盘,它属于另一类存储器。根据其制造材料和记录信息方式的不同一般分为三类:只读光盘、一次写入型光盘和可擦写光盘。

(三)输入设备

输入设备是外界向计算机传送信息的装置。在微型计算机系统中,最常用的输入设备是键盘和鼠标,此外还有光电笔、数字化仪、图像扫描仪、触摸屏、麦克风、视频输入设备、条形码扫描器等。

(四)输出设备

输出设备的作用是将计算机中的数据信息传送到外部媒介,并转化成某种为人们所认识的表示形式。在微型计算机中,最常用的输出设备是显示器和打印机。

(五)总线结构

按照总线上传输信息的不同,总线可以分为数据总线(DB),地址总线(AB)和控制总线(CB)三种。

1.数据总线:用来传送数据信息,它主要连接了 CPU 与各个部件,是它们之间交换信息的通路。

2.地址总线:用来传送地址信息。CPU 通过地址总线中传送的地址信息访问存储器,通常是单向的。

3.控制总线:用来传送控制信号,以协调各部件之间的操作。

(六)主要的性能指标

计算机常用指标:

1.字长

字长是指一台计算机所能处理的二进制代码的位数。计算机的字长直接影响到它的精度、功能和速度。字长愈长,表示的数值范围就越大,计算出的结果的有效位数也就越多;字长愈长,能表示的信息就越多,机器的功能就更强。

2.运算速度

运算速度是指计算机每秒钟所能执行的指令条数，一般用 MIPS(Million of Instructions Per Second，即每秒百万条指令)为单位。

3.主频

主频是指计算机 CPU 的时钟频率，它在很大程度上决定了计算机的运算速度。一般时钟频率越高，运算速度就越快。

4.内存容量

内存容量是指内存储器中能够存储信息的总字节数，一般以 GB 为单位。内存容量反映内存储器存储数据的能力。目前计算机的内存容量有 4GB、8GB、16GB 等。

课堂练习

1.【NOIP2008】微型计算机中，控制器的基本功能是(　　)。

A. 控制机器各个部件协调工作　　B. 实现算术运算和逻辑运算

C. 获取外部信息　　D. 存放程序和数据

【答案】A

【分析】微型计算机由运算器、控制器、存储器、输入设备和输出设备五大设备组成。控制器是整个 CPU 的指挥控制中心，对协调整个电脑有序工作极为重要。

2.【NOIP2012】地址总线的位数决定了 CPU 可直接寻址的内存空间大小，例如地址总线为 16 位，其最大的可寻址空间为 64KB。如果地址总线是 32 位，则理论上最大可寻址的内存空间为(　　)。

A. 128KB　　B. 1MB　　C. 1GB　　D. 4GB

【答案】D

【分析】地址总线 Address Bus 是专门用来传送地址的，由于地址只能从 CPU 传向外部存储器或 I/O 端口，所以地址总线总是单向三态的，这与数据总线不同。地址总线的位数决定了 CPU 可直接寻址的内存空间大小，比如 8 位微机的地址总线为 16 位，则其最大可寻址空间为 $2^{16}=64$KB，16 位微型机的地址总线为 20 位，其可寻址空间为 $2^{20}=1$MB。一般来说，若地址总线为 n 位，则可寻址空间为 2^n 字节。32 位 X86 架构是指个人电脑的地址总线是 32 位的，CPU、内存控制器、操作系统都是按 32 位地址总线设计。32 位地址总线可以支持的内存地址代码是 4096MB，也就是有 4GB 的地址代码。当然，如果对这些基础知识不了解的话，我们也可以根据题目条件计算得 $64\text{KB}*2^{32-16}=4194304\text{KB}=4096\text{MB}=4\text{GB}$。

3.【NOIP2014】CPU、存储器、I/O 设备是通过(　　)连接起来的。

A. 接口　　B. 总线　　C. 控制线　　D. 系统文件

【答案】B

【分析】计算机常识问题，总线是一种内部结构，它是 CPU、内存、输入、输出设备传递信息的公用通道，主机的各个部件通过总线相连接，外部设备通过相应的接口电路再与总线相连接，从而形成了计算机硬件系统。

4.【NOIP2015】下列说法正确的是(　　)。

A. CPU 的主要任务是执行数据运算和程序控制

B. 存储器具有记忆能力，其中信息任何时候都不会丢失

C. 两个显示器屏幕尺寸相同，则它们的分辨率必定相同

D. 个人用户只能使用 Wifi 的方式连接到 Internet

【答案】A

【分析】CPU 由运算器、控制器和一些寄存器组成，有的存储器（例如随机存取存储器 RAM）在断电后数据丢失；分辨率不是由显示屏的尺寸决定的，相同尺寸的显示器也可能有不同的分辨率；个人用户还可以拨号上网，宽带接入等。

5.【NOIP2009】关于 CPU，下面说法正确的是（　　）

A. CPU 全称为中央处理器（或中央处理单元）

B. CPU 可以直接运行汇编语言

C. 同样主频下，32 位的 CPU 比 16 位的 CPU 运行速度快一倍

D. CPU 最早是由 Intel 公司发明的

【答案】A

【分析】CPU 只能运行二进制代码也就是机器语言；C 选项中位数只能说明处理的字长，所在的系统硬件指令不同，速度难以比较；Intel 公司推出了世界上第一台微处理器，但是之前 CPU 已由电子管与晶体管实现。

6.【NOIP2011】寄存器是（　　）的重要组成部分。

A. 硬盘　　B. 高速缓存　　C. 内存　　D. 中央处理器（CPU）

【答案】D

【分析】寄存器是中央处理器内的组成部分。每年必考硬件知识题。其中寄存器和一级缓存在 CPU 内，一级缓存又名片上的缓存。二级缓存、只读存储器 ROM 和随机存储器 RAM 都在主板上，二级缓存又名板上的缓存，只读存储器 ROM 和随机存储器 RAM 共同构成内存。外存指硬盘、光盘和可移动磁盘等。CPU 包括运算逻辑部件 ALU、寄存器部件和控制部件等。

7.【NOIP2012】目前计算机芯片（集成电路）制造的主要原料是（　　），它是一种可以在沙子中提炼出的物质。

A. 硅　　B. 铜　　C. 锗　　D. 铝

【答案】A

【分析】考察计算机基础知识，所谓集成电路是将大量的晶体管和电子线路组合在一块硅片上，故又称其为芯片。制造芯片的原材料拥有半导体特性，硅是地壳里含量第二的常见元素，因此采用硅材料的计算机芯片可以廉价地批量生产，美国著名的 IT 产业发祥地被称为硅谷。

8.【NOIP2012】目前个人电脑的（　　）市场占有率最靠前的厂商包括 Intel、AMD 等公司。

A. 显示器　　B. CPU　　C. 内存　　D. 鼠标

【答案】B

【分析】英特尔公司是全球最大的个人计算机零件和 CPU 制造商。

9.【NOIP2015】在 PC 机中，PENTIUM（奔腾）、酷睿、赛扬等是指（　　）。

A. 生产厂家名称　　B. 硬盘的型号

C. CPU 的型号　　D. 显示器的型号

【答案】C

【分析】都是 Intel 公司研发的 CPU 处理器，因此这些都是 CPU 型号。

10.【NOIP2016】以下不是 CPU 生产厂商的是(　　)。

A. Intel　　B. AMD　　C. Microsoft　　D. IBM

【答案】C

【分析】Microsoft 属于软件公司，没有硬件产品，IBM 软硬件都生产。

11.【NOIP2016】以下是 32 位机器和 64 位机器的区别的是(　　)。

A. 显示器不同　　B. 硬盘大小不同　C. 寻址空间不同　　D. 输入法不同

【答案】C

【分析】考察计算机科学基础知识。32 位、64 位指的是地址总线的位数，$2^{32}=4G$，所以 32 位系统最多控制 4GB 内存，同理 64 位机器理论上最多能控制 16EiB 内存。

12.【NOIP2016】某计算机的 CPU 和内存之间的地址总线宽度是 32 位(bit)，这台计算机最多可以使用(　　)的内存。

A. 2GB　　B. 4GB　　C. 8GB　　D. 16GB

【答案】B

【分析】计算机硬件知识，$2^{32}=4GB$ 个内存单元。

13.【NOIP2015】所谓的“中断”是指(　　)。

A. 操作系统随意停止一个程序的运行

B. 当出现需要时，CPU 暂时停止当前程序的执行转而执行处理新情况的过程

C. 因停机而停止一个程序的运行

D. 电脑死机

【答案】B

【分析】考察计算机科学基础知识。中断是指计算机运行过程中，出现某些意外情况需主机干预时，机器能自动停止正在运行的程序并转入处理新情况的程序，处理完毕后再返回原被暂停的程序继续运行。

14.【NOIP2010】主存储器的存取速度比中央(CPU)的工作速度慢的多，从而使得后者的效率受到影响。而根据局部性原理，CPU 所访问的存储单元通常都趋于聚集在一个较小的连续区域中。于是，为了提高系统整体的执行效率，在 CPU 中引入了(　　)。

A. 寄存器　　B. 高速缓存　　C. 闪存　　D. 外存

【答案】B

【分析】高速缓冲存储器是存在于主存与 CPU 之间的一级存储器。闪存卡是利用闪存技术存储电子信息的存储器，一般应用在数码相机，手机等小型数码产品中作为存储介质，样子小巧，犹如一张卡片，所以称之为闪存卡。

15.【NOIP2008】计算机在工作过程中，若突然停电，(　　)中的信息不会丢失。

A. ROM 和 RAM　　B. CPU　　C. ROM　　D. RAM

【答案】C

【分析】CPU 停电不保存信息；ROM 是只读内存(Read-Only Memory)的简称，其特性是一旦存储资料就无法再将之改变或删除，资料并不会因为电源关闭而消失；RAM 是随机存储器(Random Access Memory)的简称，存储单元的内容可按需随意存取，停电不保存信息。

16.【NOIP2009】关于计算机内存，下面的说法正确的是(　　)。

A. 随机存储器(RAM)的意思是当程序运行时，每次具体分配给程序的内存位置是随机而不确定的

B. 1MB 内存通常是指 1024 * 1024 字节大小的内存

C. 计算机内存严格说来包括主存(memory)、高速缓存(cache)和寄存器(register)三个部分

D. 一般内存中的数据即使在断电的情况下也能保留 2 个小时以上

【答案】B

【分析】RAM 不是位置随机，而是随时访问；1MB＝1024KB＝1024 * 1024B；高速缓存和寄存器的物理实现是集成在 CPU 中，这两部分不属于冯·诺伊曼体系中的五大部分的任意一个部分；显然内存中数据断电后会消失。

17.【NOIP2014】断电后会丢失数据的存储器是(　　)。

A. RAM　　B. ROM　　C. 硬盘　　D. 光盘

【答案】A

【分析】RAM 是随机存储器(Random Access Memory)的简称，存储单元的内容可按需随意取出或存入，且存取的速度与存储单元的位置无关的存储器，停电不保存信息。ROM 是只读存储器，数据的保存与是否通电无关。硬盘和光盘都是常见的数据载体，和是否通电无关。

18.【NOIP2012】计算机如果缺少(　　)，将无法正常启动。

A. 内存　　B. 鼠标　　C. U 盘　　D. 摄像头

【答案】A

【分析】内存是计算机启动时必须包含的，另外三个和计算机启动没有关系。

19.【NOIP2018】以下属于输出设备的是(　　)。

A. 扫描仪　　B. 键盘　　C. 鼠标　　D. 打印机

【答案】D

【分析】考查计算机基础知识，扫描仪，键盘，鼠标都是输入设备，而打印机是输出设备。

20.【NOIP2016】以下不是存储设备的是(　　)。

A. 光盘　　B. 磁盘　　C. 固态硬盘　　D. 鼠标

【答案】D

【分析】考察计算机科学基础知识。鼠标属于输入设备，固态硬盘是新型外存设备。

不定项选择题

1.【NOIP2009】关于 CPU，下面说法正确的是(　　)。

A. CPU 全称为中央处理器(或中央处理单元)

B. CPU 能直接运行机器语言

C. CPU 最早是由 Intel 公司发明的

D. 同样主频下，32 位的 CPU 比 16 位的 CPU 运行速度快一倍

【答案】AB

【分析】Intel 最早发明的是微处理器，位数只能说明处理的字长，所在的系统硬件指令不同，速度很难说谁快。

2.【NOIP2008】计算机在工作过程中，若突然停电，(　　)中的信息不会丢失。

A. 硬盘　　B. CPU　　C. ROM　　D. RAM

【答案】AC

【分析】CPU 停电不保存信息；ROM 是只读内存(Read－Only Memory)的简称，是一种只能读出事先所存储数据的固态半导体存储器，其特性是一旦存储资料就无法再将之改变或删除，通常用在不需经常变更资料的电子或电脑系统中，资料不会因为电源关闭而消失；RAM 是随机存储器(Random Access Memory)的简称，存储单元的内容可按需随意取出或存入，且存取的速度与存储单元的位置无关的存储器，停电不保存信息；硬盘是电脑主要的存储媒介之一，不会因停电而使信息丢失。

3.【NOIP2009】关于计算机内存，下列说法正确的是(　　)。

A. 随机存储器(RAM)的意思是当程序运行时，每次具体分配给程序的内存位置是随机而不确定的

B. 一般的个人计算机在同一时刻只能存/取一个特定的内存单元

C. 计算机内存严格说来包括主存(memory)、高速缓存(cache)和寄存器(register)三个部分

D. 1MB 内存通常是指 1024 * 1024 字节大小的内存

【答案】BD

【分析】一般是对字节的一个单元串行操作。1MB＝1024KB＝1024 * 1024B。RAM 不是位置随机，而是随时访问。高速缓存和寄存器的物理实现是集成在 CPU 中，这两部分不属于冯·诺伊曼体系中五大部分的任意一个部分。对于分配程序内存问题有个反例，如果数组的存储方式在位置上不连续的话就跟链表一样，不能实现 O(1)的随机访问。

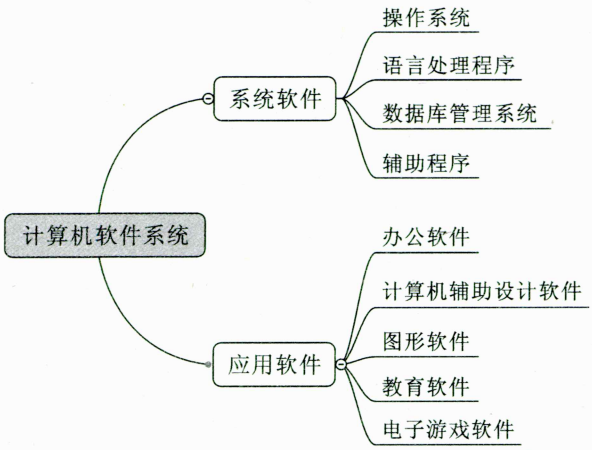
4.【NOIP2012】在计算机显示器所使用的 RGB 颜色模型中，(　　)属于三原色之一。

A. 黄色　　B. 蓝色　　C. 紫色　　D. 绿色

【答案】BD

【分析】RGB：Red，Green，Blue，红绿蓝。

第3节 计算机软件系统



软件系统是计算机的灵魂。没有安装软件的计算机称为“裸机”，无法完成任何工作。硬件为软件提供运行平台。软件和硬件相互关联，两者之间可以相互转化，互为补充。

计算机软件系统按其功能可分为系统软件和应用软件两大类。

一、系统软件

系统软件是指管理、控制和维护计算机及其外部设备，提供用户与计算机之间操作界面等方面的软件，并不专门针对具体的应用问题。

代表性的系统软件有：操作系统、数据库管理系统以及各种程序设计语言的编译系统等，其中最重要的系统软件是操作系统。

（一）操作系统（Operating System）

操作系统是最基本的系统软件，是用于管理和控制计算机所有软件、硬件资源的一组程序。操作系统直接运行在裸机上，其他的软件（包括系统软件和大量的应用软件）都是建立在操作系统基础上的，并得到它的支持和取得它的服务。

操作系统是计算机硬件与其他软件的接口，也是用户和计算机之间的接口。

功能：处理机管理、存储管理、设备管理、信息管理等。

操作系统的性能很大程度上决定了整个计算机系统的性能。

分类：

1.按与用户对话的界面不同，可以分为命令行界面操作系统和图形用户界面操作系统；

2.按能够支持的用户数为标准，可以分为单用户操作系统和多用户操作系统；

3.按是否能够运行多个任务为标准，分为单任务操作系统和多任务操作系统；

4.按系统单功能为标准，分为批处理系统、分时操作系统、实时操作系统、网络操作系统。

（二）语言处理程序

计算机在执行程序时，首先要将存储在存储器中的程序指令逐条取出来，经过译码后向计算机的各部件发出控制信号，使其执行规定的操作。计算机的控制装置能够识别的指令是用机器语言编写的，而用机器语言编写一个程序并不是一件容易的事。绝大多数用户都是用某种程序设计语言（即高级语言），如 BASIC 语言、C 语言等来编写程序。但是用这些

高级语言编写的程序 CPU 不认识，必须要经过翻译变成机器指令后才能被计算机执行。而负责这种翻译的程序称为编译程序。为了在计算机上执行由某种高级语言编写的程序，就必须配置有该种语言的编译系统。

（三）数据库管理系统

计算机的效率主要是指数据处理的效率。

数据库管理系统的功能：有组织的、动态的存储大量的数据信息；使用户能方便、高效地使用这些数据信息。

数据库软件体系包括数据库、数据库管理系统和数据库系统三个部分。

1.数据库（Data Base，DB）是为了满足一定范围内许多用户的需要，在计算机里建立的一组互相关联的数据集合。

2.数据库管理系统（Data Base Management Systems，DBMS）是指对数据库进行组织、管理、查询并提供一定处理能力的系统软件。它是数据库系统的核心组成部分，为用户或应用程序提供访问数据库的方法，数据库的一切操作都是通过 DBMS 进行的。

3.数据库系统（Data Base System，DBS）是由数据库及其管理软件组成的系统。数据库管理员是专门从事数据库建立、使用和维护的工作人员。

DBMS 是位于用户（或应用程序）和操作系统之间的软件。它是在操作系统支持下运行的，借助于操作系统实现对数据的存储和管理，使数据能被各种不同的用户所共享，并保证用户得到的数据是完整的、可靠的。DBMS 与用户之间的接口称为用户接口，且提供给用户可使用的数据库语言。

（四）辅助程序

辅助程序完成一些与管理计算机系统资源及文件有关的任务，如诊断程序、反病毒程序、卸载程序、备份程序、文件解压缩程序等工具类软件。

二、应用软件

应用软件是指专门为解决某个应用领域内的具体问题而编制的软件（或实用程序）。

应用软件一般不能独立的在计算机上运行，必须要有系统软件的支持。应用软件特别是各种专用软件包经常是由软件厂商提供的。

常见的应用软件有以下几类：

1.文字处理软件

用于输入、存储、修改、编辑、打印文字资料（文件、稿件等）。常用的有 WPS，Word 等。

2.办公软件

用于输入、存储、修改、检索各种信息。例如工资管理系统、人事管理系统等。这种软件发展到一定水平后，可以将各个单项软件联接起来，构成一个完整的、高效的管理系统，简称 MIS。

3.计算机辅助设计软件

用于高效的绘制、修改工程图纸，进行常规设计和计算，帮助用户寻求较优的设计方案。常用的有 AutoCAD 等。

4.图形软件

用于图形的生成、表示和操作的软件。根据图形的几何性质和外貌特征，使用程序设计语言对其进行形式描述，是软件处理图形的基础。

5.教育软件

顾名思义，为教育服务的软件系统。根据应用范围的不同可以分为家用教育软件、校用教育软件和远程教育软件。

6.电子游戏软件

利用电子设备为媒介为用户提供游戏活动的软件。

课堂练习

1.【NOIP2008】在以下各项中，(　　)不是操作系统软件。

A. Solaris　　B. Linux　　C. Windows Vista　　D. Sybase

【答案】D

【分析】Solaris是Sun Microsystems研发的计算机操作系统，它被认为是Unix操作系统的衍生版本之一；Sybase，是美国Sybase公司研制的一种关系型数据库系统，是一种典型的Unix或WindowsNT平台上客户机/服务器环境下的大型数据库系统。Linux是一套免费使用和自由传播的类Unix操作系统。Windows Vista是微软(Microsoft)公司的Windows操作系统系列成员之一。

2.【NOIP2009】下列软件中不是计算机操作系统的是(　　)。

A. Windows　　B. Linux　　C. OS/2　　D. WPS

【答案】D

【分析】WPS是文档编辑软件，常用来办公；OS/2是微软和IBM开发的操作系统。

3.【NOIP2010】Linux下可执行文件的默认扩张名为(　　)。

A. exe　　B. com　　C. dll　　D. 以上都不是

【答案】D

【分析】Linux下可执行文件默认没有扩展名，动态链接库英文为DLL，是Dynamic Link Library的缩写形式，DLL是一个包含可由多个程序同时使用的代码和数据的库，且不是可执行文件。

4.【NOIP2012】(　　)不属于操作系统。

A. Windows　　B. DOS　　C. PhotoShop　　D. NOI Linux

【答案】C

【分析】Adobe Photoshop，简称“PS”，是由Adobe Systems开发和发行的图像处理软件。

5.【NOIP2014】下列对操作系统功能的描述最为完整的是(　　)。

A. 负责外设与主机之间的信息交换

B. 负责诊断机器的故障

C. 控制和管理计算机系统的各种硬件和软件资源的使用

D. 将源程序编译成目标程序

【答案】C

【分析】将源程序编译成目标程序是编译器的功能。其余选项均描述的是操作系统的功能，但最为完整的描述是控制和管理计算机系统的各种硬件和软件资源的使用，另外两个选项只涉及了操作系统的部分功能。

6.【NOIP2015】操作系统的作用是(　　)。

A. 把源程序译成目标程序　　B. 便于进行数据管理

C. 控制和管理系统资源　　D. 实现硬件之间的连接

【答案】C

【分析】操作系统是管理和控制计算机硬件与软件资源的计算机系统软件。把源程序编译成目标程序是编译器的作用;便于进行数据管理对应数据库的功能;实现硬件之间的连接是主板的作用。

7.【NOIP2009】关于 BIOS,下列说法正确的是(　　)。

A. BIOS 是计算机基本输入输出系统软件的简称

B. BIOS 里包含了键盘、鼠标、声卡、显卡、打印机等常用输入输出设备的驱动程序

C. BIOS 一般由操作系统厂商来开发完成

D. BIOS 能供提各种文件拷贝、复制、删除以及目录维护等文件管理功能

【答案】A

【分析】BIOS 是英文“Basic Input Output System”的缩略语,直译过来后中文名称就是“基本输入输出系统”。其实,它是一组固化到计算机内主板上一个 ROM 芯片上的程序,保存着计算机最重要的基本输入输出的程序、系统设置信息、开机后自检程序和系统自启动程序。其主要功能是为计算机提供最底层以及最直接的硬件设置和控制。

8.【NOIP2011】有人认为,在个人电脑送修前,将文件放入回收站中就是已经将其删除了。这种想法是(　　)。

A. 正确的,将文件放入回收站意味着彻底删除、无法恢复

B. 不正确的,只有将回收站清空后,才意味着彻底删除、无法恢复

C. 不正确的,即使将回收站清空,文件只是被标记为删除,仍可能通过恢复软件找回

D. 不正确的,只要在硬盘上出现过的文件,永远不可能被彻底删除

【答案】C

【分析】排除法,AD 首先排除,市场有一些软件是可以帮助恢复删除文件的,所以选 C。

9.【NOIP2016】以下不是微软公司出品的软件是(　　)。

A. Powerpoint　　B. Word　　C. Excel　　D. Acrobat Reader

【答案】D

【分析】Acrobat Reader 是属于 Adobe 公司的软件。Adobe 是著名的软件公司,拥有 photoshop,acrobat,audition 等软件。

10.【NOIP2015】下面哪种软件不属于即时通信软件(　　)。

A. QQ　　B. MSN　　C. 微信　　D. P2P

【答案】D

【分析】排除法,QQ、MSN、微信都是常用的即时通信软件,P2P(Peer to Peer)则是计算机网络中的一种常见结构:对等网络。

11.【NOIP2015】下列选项中不属于视频文件格式的是(　　)。

A. TXT　　B. AVI　　C. MOV　　D. RMVB

【答案】A

【分析】TXT 格式是纯文本格式,其余均为常见视频文件格式。

12.【NOIP2014】下列选项中不属于图像格式的是(　　)。

A. JPEG 格式　　B. TXT 格式　　C. GIF 格式　　D. PNG 格式

【答案】B

【分析】TXT 格式是纯文本格式，其余三个选项均为常见的图像格式。

不定项选择题

1.【NOIP2009】关于操作系统，下列说法正确的是(　　)。

A. 多任务操作系统专用于多核心或多个 CPU 架构的计算机系统的管理

B. 在操作系统的管理下，一个完整的程序在运行过程中可以被部分存放在内存中

C. 分时系统让多个用户可以共享一台主机的运算能力，为保证每个用户都得到及时的响应，通常会采用时间片轮转调度的策略

D. 为了方便上层应用程序的开发，操作系统都是免费开源的

【答案】BC

【分析】多任务系统可以是单个 CPU 构架的，普通的 PC 都是多任务的。操作系统不都是免费开源的。

2.【NOIP2014】下列(　　)属于操作系统。

A. Microsoft Word　B. Windows XP　C. Android　　D. Mac OS X

E. Oracle

【答案】BCD

【分析】常识题。常见的操作系统有 Linux，Windows，Mac，Android，iOS 等。Microsoft Word 是微软 office 系列软件里面的文字处理软件。Oracle 是一种数据库。

3.【NOIP2015】以下属于操作系统的有(　　)。

A. Windows XP　　B. Unix　　C. Linux　　D. MacOS

【答案】ABCD

【分析】四个选项都属于操作系统。

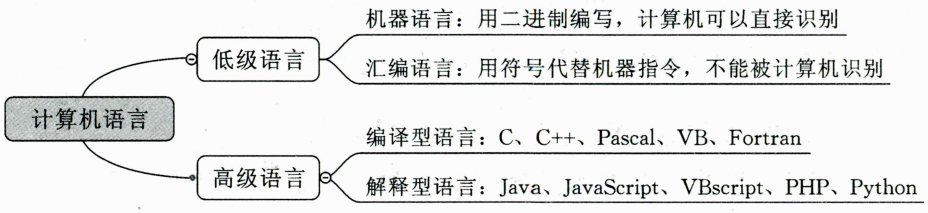
4.【NOIP2015】下列属于视频文件格式的有(　　)。

A. AVI　　B. MPEG　　C. WMV　　D. JPEG

【答案】ABC

【分析】JPEG 格式属于图像格式。

第 4 节　计算机语言



计算机语言分为高级语言和低级语言。

一、低级语言

低级语言依赖于硬件，可读性和可移植性差。低级语言分为机器语言和汇编语言。

1.机器语言

机器语言是用二进制代码表示的计算机能直接识别和执行的指令集和，具有灵活、直接执行和速度快等特点。

2.汇编语言

汇编语言，即第二代计算机语言，用一些容易理解和记忆的字母、单词来代替一个特定的指令，比如，用“ADD”代表数字逻辑上的加减，“MOV”代表数据传递等，汇编语言必须翻译成机器语言才能执行。在今天的实际应用中，其通常被应用在底层，硬件操作和高要求的程序优化的场合。驱动程序、嵌入式操作系统和实时运行程序都需要汇编语言。

二、高级语言

高级语言是一种独立于机器的语言，从解决问题方式的角度分为面向过程语言（C、Pascal、Fortran）和面向对象语言（C++、C＃、Java、Python、JavaScript）。

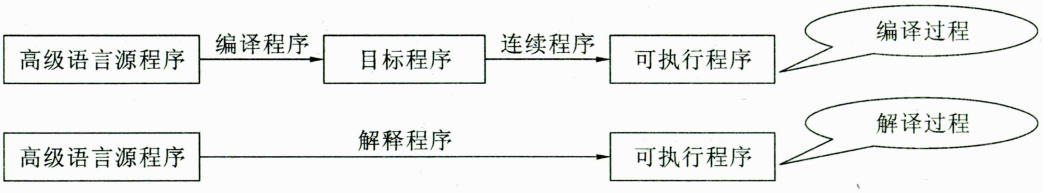
✧ 面向过程：自顶往下逐步求精，最重要的是模块化的思想方法。

✧ 面向对象：把事物抽象成类，再实例化成对象，对象包括属性与行为。

先有面向过程再有面向对象，面向对象是基于面向过程的。当程序规模不是很大时，面向过程的方法会体现出一种优势，而在大型项目中面向对象的方法更有优势。

Fortran 是第一个高级语言，Smalltalk 是第一个支持面向对象的语言。

高级语言设计的程序必须经“翻译”后才能被机器执行。将高级语言编写的程序翻译成机器语言程序，翻译过程有编译和解释两种方式。



1.编译型语言

编译型程序在执行之前需要一个专门的编译过程，把程序编译成为机器语言的文件，运行时不需要重新翻译，直接使用编译的结果就可以了。编译型程序执行效率高，依赖编译器，跨平台性差一些，例如 C/ C++、Pascal/ObjectPascal（Delphi）等。

2.解释型语言

解释是把源程序翻译一句，执行一句的过程，而编译是源程序翻译成机器指令形式的目标

程序的过程，再用链接程序把目标程序链接成可执行程序后才能执行。解释型程序不需要编译，程序在运行时才翻译成机器语言，每执行一次都要翻译一次，因此效率比较低。比如 Basic 语言，专门有一个解释器能够直接执行 Basic 程序，每个语句都是执行的时候才翻译。

解释型语言逐行解释执行，不产生目标文件，例如 ASP、Java、C＃、PHP、JavaScript、VBScript、Perl、Python、Ruby、MATLAB 等。



课堂练习

1.【NOIP2008】下列不属于 NOIP 竞赛推荐使用的语言环境的是(　　)。

A. Dev-C++　　B. Visual C++　　C. free pascal　　D. Lazarus

【答案】B

【分析】考察计算机常识，Visual C++从未在 NOIP 竞赛中被推荐使用，另外三个是允许使用的。但是随着 CCF 对 NOI 系列活动的规则调整，后两个也会被逐步淘汰，在无新增程序设计语言的情况下，NOI 系列赛事自 NOIP2022 开始将仅支持 C++语言。

2.【NOIP2017】下列不属于面向对象程序设计语言的是(　　)。

A. C　　B. C++　　C. Java　　D. C＃

【答案】A

【分析】C 语言是面向过程的编程语言，其余三个都是面向对象的编程语言。

3.【NOIP2008】面向对象程序设计(Object-Oriented Programming)是一种程序设计的方法论，它将对象作为程序的基本单元，将数据和程序封装在对象中，以提高软件的重用性、灵活性和扩展性。下面关于面向对象程序设计的说法中，不正确的是(　　)。

A. 面向对象程序设计通常采用自顶向下设计方法进行设计

B. 面向对象程序设计方法具有继承性(inheritance)、封装性(encapsulation)、多态性(polymorphism)等几大特点

C. 支持面向对象特性的语言称为面向对象的编程语言，目前较为流行的有C++、Java、C＃等

D. 面向对象的程序设计的雏形来自于 Simula 语言，后来在 SmallTalk 语言的完善和标准化的过程中得到更多的扩展和对以前思想的重新注解。至今，SmallTalk 语言仍然被视为面向对象语言的基础

【答案】A

【分析】程序设计的方法包括结构化设计(面向过程)和面向对象设计方法。C 和 Pascal 都是面向过程的程序设计语言。面向过程程序设计才是采用自顶向下设计方法进行设计，而面向对象程序设计则不是。C++是面向对象的程序设计语言。SmallTalk 语言仍然被视为面向对象语言的基础。面向对象的三个基本特征是封装、继承、多态。

4.【NOIP2014】以下属于面向对象的高级语言的是(　　)。

A. 汇编语言　　B. C++　　C. Fortran　　D. Basic

【答案】B

【分析】汇编语言是面向机器的低级语言，Fortran、Basic 是面向过程的高级语言。而 C++是面向对象的高级语言。

5.【NOIP2011】关于汇编语言，下列说法错误的是(　　)。

A. 是一种与具体硬件相关的程序设计语言

B. 在编写复杂程序时，相对于高级语言而言代码量较大，且不易调试
C. 可以直接访问寄存器、内存单元以及 I/O 端口
D. 随着高级语言的诞生，如今已完全被淘汰，不再使用

【答案】D

【分析】把汇编语言翻译成机器语言的过程称为汇编。汇编更接近机器语言，能够直接对硬件进行操作，生成的程序与其他的语言相比具有更高的运行速度，占用更小的内存，因此大量应用在一些对于时效性要求很高的程序、大型程序的核心模块以及工业控制方面。

6.【NOIP2009】关于程序设计语言，下面说法正确的是(　　)。
A. 加了注释的程序一般会比同样的没有加注释的程序运行速度慢
B. 高级语言开发的程序不能使用在低层次的硬件系统(如：自控机床)或低端手机上
C. 高级语言相对于低级语言更容易实现跨平台的移植
D. 以上说法都不对

【答案】C

【分析】注释是被电脑忽略的；高级语言开发的程序可以使用低层硬件，编译后生成目标代码。

7.【NOIP2010】Pascal 语言、C 语言和 C++语言都属于(　　)。
A. 面向对象语言　　B. 脚本语言　　C. 解释性语言　　D. 编译性语言

【答案】D

【分析】考察计算机语言常识，编译是指在应用源程序执行之前，就将程序源代码“翻译”成目标代码(机器语言)，因此其目标程序可以脱离其语言环境独立执行，使用比较方便、效率较高。Pascal 语言和 C 语言都是面向过程语言，而 C++语言是面向对象语言。

8.【NOIP2014】编译器的主要功能是(　　)。
A. 将一种高级语言翻译成另一种高级语言　　B. 将源程序翻译成指令
C. 将低级语言翻译成高级语言　　D. 将源程序重新组合

【答案】B

【分析】编译是从源代码(通常为高级语言)到能直接被计算机或虚拟机执行的目标代码(通常为低级语言或机器语言)的翻译过程。简单讲，编译器就是将“一种语言(通常为高级语言)”翻译为“另一种语言(通常为低级语言甚至机器语言)”的程序。

9.【NOIP2018】下列属于解释执行的程序设计语言是(　　)。
A. C　　B. C++　　C. Pascal　　D. Python

【答案】D

【分析】Python 是解释型语言，其他三个都是编译型语言。

不定项选择题

1.【NOIP2008】NOIP 竞赛推荐使用的语言环境有(　　)。
A. Dev— C++　　B. Visual C++　　C. free pascal　　D. Lazarus

【答案】ACD

【分析】常识，Visual C++从未在 NOIP 竞赛中被推荐使用，另外三个是允许使用的，但是随着 CCF 对 NOI 系列活动的规则调整，后两个也会被逐步淘汰，在无新增程序设计语言的情况下，NOI 系列赛事自 NOIP2022 开始将仅支持 C++语言。

2.【NOIP2008】面向对象程序设计(Object-Oriented Programming)是一种程序设计的方法论,它将对象作为程序的基本单元,将数据和程序封装在对象中,以提高软件的重用性、灵活性和扩展性。下面关于面向对象程序设计的说法中,正确的是(　　)。

A. 面向对象程序设计通常采用自顶向下设计方法进行设计

B. 面向对象程序设计方法具有继承性(inheritance)、封装性(encapsulation)、多态性(polymorphism)等几大特点

C. 支持面向对象特性的语言称为面向对象的编程语言,目前较为流行的有C++、Java、C#等

D. 面向对象的程序设计的雏形来自于 Simula 语言,后来在 SmallTalk 语言的完善和标准化的过程中得到更多的扩展和对以前思想的重新注解。至今,SmallTalk 语言仍然被视为面向对象语言的基础

【答案】BCD

【分析】程序设计的方法包括结构化设计(面向过程)和面向对象设计方法。C 和 Pascal 都是面向过程的程序设计语言。面向过程程序设计方法的四条原则是:1.自顶向下;2.逐步求精 3.模块化;4.限制使用 goto 语句。所以,面向过程程序设计才是采用自顶向下设计方法进行设计,而面向对象程序设计则不是。C++是面向对象的程序设计语言。SmallTalk 语言仍然被视为面向对象语言的基础。面向对象的三个基本特征是封装、继承、多态。

3.【NOIP2017】以下是面向对象的高级语言的有(　　)。

A. 汇编语言　　B. C++　　C. Fortran　　D. Java

【答案】BD

【分析】计算机基础知识的常识问题,C++和 Java 是面向对象的高级语言。汇编语言不是高级语言,Fortran 是面向过程的高级语言。

4.【NOIP2011】汇编语言(　　)。

A. 是一种与具体硬件无关的程序设计语言

B. 在编写复杂程序时,相对于高级语言而言代码量较大,且不易调试

C. 可以直接访问寄存器、内存单元、I/O 端口

D. 随着高级语言的诞生,如今已完全被淘汰,不再使用

【答案】BC

【分析】汇编语言是一种功能很强的程序设计语言,也是利用计算机所有硬件特性并能直接控制硬件的语言。汇编语言作为一门语言,对应于高级语言的编译器,需要一个“汇编器”来把汇编语言源文件汇编成机器可执行的代码。它像机器指令一样,是硬件操作的控制信息,因而仍然是面向机器的语言,使用起来比较繁琐费时,通用性也差。但是,汇编语言用来编制系统软件和过程控制软件时,其目标程序占用内存空间少,运行速度快,有着高级语言不可替代的用途。

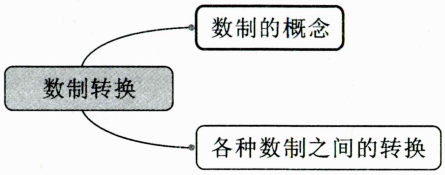
5.【NOIP2010】Pascal 语言,C 语言和 C++语言都属于(　　)。

A. 高级语言　　B. 自然语言　　C. 解释性语言　　D. 编译性语言

【答案】AD

【分析】计算机语言常识,编译是指在应用源程序执行之前,就将程序源代码“翻译”成目标代码(机器语言),因此其目标程序可以脱离其语言环境独立执行,使用比较方便、效率较高。但应用程序一旦需要修改,必须先修改源代码,再重新编译生成新的目标文件才能执行,只有目标文件而没有源代码,修改很不方便。Pascal 语言和 C 语言都是面向过程语言,而 C++语言面向对象语言。

第5节　数制转换



一、计算机的数制

1.十进制：每一位可取0～9这十个数码，计数的基数为10，超过9须用多位数来表示。

2.二进制：每一位可取0～1这2个数码，计数的基数为2，超过1须用多位数来表示。

3.八进制：每一位可取0～7这8个数码，计数的基数为8，超过7须用多位数来表示。

4.十六进制：每一位可取0～9，A，B，C，D，E，F这16个数码，计数的基数为16，超过F须用多位数来表示。

5.权：数制中某一位上的1所表示的数值的大小。例如，十进制123，1的位权是100，2的位权是10，3的位权是1。

各进制转换的关系表

2(B)	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
8(O)	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
10(D)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16(H)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

二、各种数制之间的转换

1.K进制转十进制

按权相加法展开成一个多项式，每项是该位的数码与相应的权之积，把多项式按十进制数的规则进行求和，所得结果即是该数的十进制。

公式：$abcd.efg(K)=d*K^0+c*K^1+b*K^2+a*K^3+e*K^{-1}+f*K^{-2}+g*K^{-3}(10)$

例：$1101.01(2)=1*2^0+0*2^1+1*2^2+1*2^3+0*2^{-1}+1*2^{-2}=1+0+4+8+0+0.25=13.25(10)$

2.十进制转K进制

整数部分：十进制整数转换为K进制整数的方法是倒除法，即除K直至商为0，倒取余数。

例：$172(10)=10101100(2)$

$$
\begin{array}{r|l}
2 & 172 \quad \ldots \text{余}0 \\
2 & 86 \quad \ldots \text{余}0 \\
2 & 43 \quad \ldots \text{余}1 \\
2 & 21 \quad \ldots \text{余}1 \\
2 & 10 \quad \ldots \text{余}0 \\
2 & 5 \quad \ldots \text{余}1 \\
2 & 2 \quad \ldots \text{余}0 \\
2 & 1 \quad \ldots \text{余}1 \\
\multicolumn{2}{r}{0}
\end{array}
$$

小数部分：十进制小数转换为K进制小数的方法是取整法，即小数部分循环乘以K直至为0，顺序取整数部分。

例：0.625(10)＝0.101(2)

0.625*2=1.25--------取出整数部分1
0.25*2=0.5----------取出整数部分0
0.5*2=1.0-----------取出整数部分1

3.二进制转八进制

整数部分：从右向左，二进制3位一组转成八进制。

小数部分：从左向右，二进制3位一组转成八进制。

例：1 111 010 101 101(2)＝17255(8)

例：1 101 110.1011(2)＝156.54(8)

另外，二进制也可以先转成十进制，再由十进制转成八进制。

4.二进制转十六进制

整数部分：从右向左，二进制4位一组转成十六进制。

小数部分：从左向右，二进制4位一组转成十六进制。

例：1 1110 1010 1101(2)＝1EAD(16)

例：11110 1001.10111(2)＝1E9.B8(16)

课堂练习

1.【NOIP2010】一个自然数在十进制下有n位，则它在二进制下的位数与(　　)最接近。

A. 5n　　B. $n * \log_2 10$　　C. $10 * \log_2 n$　　D. $10^n \log_2 n$

【答案】B

【分析】若是八进制，每位在二进制下占3位，可以猜想二进制下的位数是$n * \log_2$(原进制值)。因为自然数在十进制下成n位，所以说这个自然数的数量级为10^n，转化为二进制的长度为$\log_2 10^n = n\log_2 10$。

2.【NOIP2011】在二进制下，1101001＋(　　)＝1110110。

A. 1011　　B. 1101　　C. 1010　　D. 1111

【答案】B

【分析】题目要求求$(1110110)_2 - (1101001)_2$的值，根据二进制的计算规则1＋1＝10，0＋1＝1，可以算出这个值应该是1101。

3.【NOIP2013】二进制数11.01在十进制下是(　　)。

A. 3.25　　B. 4.125　　C. 6.25　　D. 11.125

【答案】A

【分析】二进制整数部分，从低到高每一位的值分别是2的0次方、1次方、2次方……而小数部分从高到低每一位的值分别是2的－1次方、－2次方、－3次方……恰好是连续的。$(11.01)_2 = 2^1 + 2^0 + 2^{-2} = 3.25$。

4.【NOIP2015】二进制数00100100和00010100的和是(　　)。

A. 00101000　　B. 01011101　　C. 01000100　　D. 00111000

【答案】D

【分析】二进制加法满二进一。0+0=0,0+1=1,1+0=1,1+1=10。

故$(00100100)_2+(00010100)_2=(00111000)_2$。

5.【NOIP2014】二进制数 111.101 所对应的十进制数是(　　)。

A. 5.625　　B. 5.5　　C. 6.125　　D. 7.625

【答案】D

【分析】$(111.101)_2=2^2+2^1+2^0+2^{-1}+2^{-3}=7.625$。

6.【NOIP2016】二进制数 00101100 和 01010101 异或的结果是(　　)。

A. 00101000　　B. 01111001　　C. 01000100　　D. 00111000

【答案】B

【分析】考查基本的逻辑运算。异或就是按位运算,相同取 0,不同取 1,直接对每一位进行处理即可。考查基本的二进制加法,1+0=1,0+1=1,1+1=10,0+0=0,故$(00101100)_2+(00010101)_2=(01111001)_2$

7.【NOIP2017】十进制小数 13.375 对应的二进制数是(　　)。

A. 1101.011　　B. 1011.011　　C. 1101.101　　D. 1010.01

【答案】A

【分析】整数部分依次除以 2 求余数,这些余数倒序后就是二进制的整数部分;小数部分依次乘以 2 求进位至整数部分的值,这些进位值顺序就是二进制的小数部分。

所以,13.375=8+4+1+1/4+1/8=1101.011。

8.【NOIP2010】设 X、Y、Z 分别代表三进制下的一位数字,若等式 XY+ZX=XYX 在三进制下成立,那么同样在三进制下,等式 XY * ZX=(　　)也成立。

A. YXZ　　B. ZXY　　C. XYZ　　D. XZY

【答案】B

【分析】XY+ZX=XYX 这个等式在 3 进制下成立,我们不妨把它化成 10 进制,3X+Y+3Z+X=9X+3Y+X,即 3Z=6X+2Y,由于 X、Y、Z 分别代表 0、1、2,枚举后得出 X=1,Y=0,Z=2,这样代入 XY * ZX=$(10)_3$ * $(21)_3=(21)_{10}=(210)_3=(ZXY)_3$。

9.【NOIP2008】与十进制数 28.5625 相等的四进制数是(　　)。

A. 123.21　　B. 131.22　　C. 130.22　　D. 130.21

【答案】D

【分析】整数部分除以 4,倒序取余数;余数部分乘以 4,顺序取进位。

$\frac{28}{4}=7\cdots\cdots0$,

$\frac{7}{4}=1\cdots\cdots3$,

$\frac{1}{4}=0\cdots\cdots1$,

整数部分为 130;

0.5625 * 4=2+0.25,

0.25 * 4=1+0,

小数部分为 21。

10.【NOIP2009】十进制小数 125.125 对应的八进制数是(　　)。

A. 100.1　　B. 175.175　　C. 175.1　　D. 100.175

【答案】C

【分析】整数部分除以 8 取余数，结果反序写；小数部分乘以 8 取整数，正序写。

$\frac{125}{8}=15\cdots\cdots5$，

$\frac{15}{8}=1\cdots\cdots7$，

$\frac{1}{8}=0\cdots\cdots1$，

整数部分为 175；

$0.125*8=1+0$，

小数部分为 1。

11.【NOIP2016】与二进制小数 0.1 相等的八进制数是(　　)。

A. 0.8　　B. 0.4　　C. 0.2　　D. 0.1

【答案】B

【分析】考查基本的进制转换，$(0.1)_2=2^{-1}=4*2^{-3}=4*8^{-1}$，也就是$(0.4)_8$。

12.【NOIP2010】与 16 进制数 A1.2 等值的 10 进制数是(　　)。

A. 101.2　　B. 111.4　　C. 161.125　　D. 177.25

【答案】C

【分析】原式等于 $10(A)*16^1+1*16^0+2*16^{-1}=161.125$。

13.【NOIP2011】一个正整数在二进制下有 100 位，则它在十六进制下有(　　)位。

A. 7　　B. 13　　C. 25　　D. 不能确定

【答案】C

【分析】$2^4=16$，因此 4 位二进制可以写成一位十六进制。

14.【NOIP2012】十六进制数 9A 在(　　)进制下是 232。

A. 四　　B. 八　　C. 十　　D. 十二

【答案】B

【分析】$(9A)_{16}=(2122)_4=(232)_8$。

15.【NOIP2013】在十六进制表示法中，字母 A 相当于十进制中的(　　)。

A. 9　　B. 10　　C. 15　　D. 16

【答案】B

【分析】十六进制中，分别用字母 A～F 来表示 10～15，字母 A 相当于十进制中的 10。

16.【NOIP2015】与二进制小数 0.1 相等的十六进制数是(　　)。

A. 0.8　　B. 0.4　　C. 0.2　　D. 0.1

【答案】A

【分析】$(0.1)_2=1*2^{-1}=0.5$。将十进制小数转成十六进制小数，需要采用“乘 16 取整，顺序排列”法。具体做法是：用 16 乘十进制小数，可以得到积，将积的整数部分取出，再用 16 乘余下的小数部分，又得到一个积，再将积的整数部分取出，如此进行，直到积中的小数部分

为零，或者达到所要求的精度为止。然后把取出的整数部分按顺序排列起来，先取的整数作为十六进制小数的高位有效位，后取的整数作为低位有效位，所以$(0.5)_{10}=(0.8)_{16}$。

17.【NOIP2008】$(2008)_{10}+(5B)_{16}$的结果是(　　)。

A. $(833)_{16}$　　B. $(2089)_{10}$　　C. $(4163)_{8}$　　D. $(100001100011)_{2}$

【答案】A

【分析】进制的计算，可以考虑将不同进制转换成同一进制进行比较。

$(5B)_{16}=5*16^{1}+11(B)*16^{0}=91$，

$(833)_{16}=8*16^{2}+3*16^{1}+3*16^{0}=2099$，

$(4163)_{8}=4*8^{3}+1*8^{2}+1*8^{1}+3*8^{0}=2163$，

$(100001100011)_{2}=2^{11}+2^{6}+2^{5}+2^{1}+2^{0}=2147$。

18.【NOIP2015】下面有四个数据组，每个组各有三个数据，其中第一个数据为八进制数，第二个数据为十进制数，第三个数据为十六进制数。这四个数据组中三个数据相同的是(　　)。

A. 120 82 50　　B. 144 100 68　　C. 300 200 C8　　D. 1762 1010 3F2

【答案】D

【分析】基础进制数转换。A，$(120)_{8}=(80)_{10}$；B，$(68)_{16}=(104)_{10}$；C，$(300)_{8}=(192)_{10}$。

19.【NOIP2010】如果在某个进制下等式 7 * 7＝41 成立，那么在该进制下等式 12 * 12＝(　　)也成立。

A. 100　　B. 144　　C. 164　　D. 196

【答案】B

【分析】通过观察发现这个不是十进制的，且大于 7 进制，在十进制下 7 * 7＝49，49 变为 41，可知是在 12 进制下，因此 12 在十进制下为 14，14 * 14＝196，196 在 12 进制下位 144。所以本题应选 B。

20.【NOIP2018】下列四个不同进制的数中，与其他三项数值上不相等的是(　　)。

A. $(269)_{16}$　　B. $(617)_{10}$　　C. $(1151)_{8}$　　D. $(1001101011)_{2}$

【答案】D

【分析】可以将所有数全部转化为十进制，发现 A，B，C 项均为 617，因此选择 D 项(619)。

但是，此题更加简单的做法是全部转化为二进制：对于 16 进制的数，转化为二进制只需要将每一数位的数转写为四位二进制数。

比如说，$(269)_{16}\rightarrow(0010|0110|1001)\rightarrow(1001101001)_{2}$。

类似的，八进制数转化为二进制的做法是将每一位数字对应转化为三位二进制数，所以$(1151)_{8}\rightarrow(001|001|101|001)_{2}\rightarrow(1001101001)_{2}$。这样可以降低运算量，事实上，算出 A 和 C 均与 D 不相同时，D 就已经可以被选出。

不定项选择题

1.【NOIP2012】十进制下的无限循环小数(不包括循环节内的数字均为0或均为9的平凡情况),在二进制下有可能是(　　)。

A. 无限循环小数(不包括循环节内的数字均为0或均为1的平凡情况)

B. 无限不循环小数

C. 有限小数

D. 整数

【答案】A

【分析】小数转换成二进制用乘法,乘2取整,剩下的再乘2取整,结果是整数部分顺序写出,所以是无限,又因为是十进制循环小数,所以乘的时候也必有循环节。

2.【NOIP2014】下列各无符号十进制整数中,能用八位二进制表示的数有(　　)。

A. 296　　B. 133　　C. 256　　D. 199

【答案】BD

【分析】无符号八位二进制数能够表示的范围是0～255。

3.【NOIP2008】$(2008)_{10}+(5B)_{16}$的结果是(　　)。

A. $(833)_{16}$　　B. $(2099)_{10}$　　C. $(4063)_8$　　D. $(100001100011)_2$

【答案】ABC

【分析】进制的计算,可以考虑将不同进制转换成同一进制进行比较。

$(5B)_{16}=5*16^1+11(B)*16^0=91$,

$(833)_{16}=8*16^2+3*16^1+3*16^0=2099$,

$(4063)_8=4*8^3+6*8^1+3*8^0=2099$,

$(100001100011)_2=2^{11}+2^6+2^5+2^1+2^0=2147$。

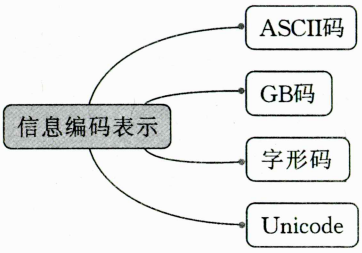
4.【NOIP2011】一个正整数在十六进制下有100位,则它在二进制下可能有(　　)位。

A. 399　　B. 400　　C. 401　　D. 404

【答案】AB

【分析】一位十六进制可以写成4位二进制,考虑到首位前导零的情况,所以在397～400之间。

第6节　信息编码表示



在计算机内部是通过二进制编码的方式表示信息，故在存储信息之前需要先转换成计算机能够识别的编码。

一、西文字符的编码表示—ASCII 码

ASCII 码(American Standard Code for Information Interchange)即美国标准信息交换代码，每个字符占 7 位的二进制位(单位比特或 bit)，能表示 $2^7=128$ 种西文字符，包括大小字母、0－9、控制符(如回车、空格)、通用符(如＋、＝)等字符。

常见的数字、字母的 ASCII 码翻译成十进制如下所示：

字符'0'⟶48

字符'A'⟶65

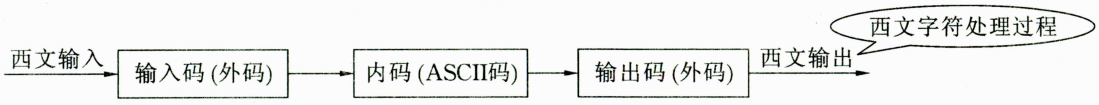
字符'a'⟶97

基本的 ASCII 码：存储时占据一个字节，最高位为 0，其范围是 0－127。

扩展的 ASCII 码：存储时占据一个字节，最高位为 1，其范围是 128－255。

外码：计算机与人进行交换的字形符号，如字符"A"的外码是"A"。

内码：计算机内部存储和使用的数字代码，如字符"A"的内码是 65。



二、中文字符的编码表示—GB 码

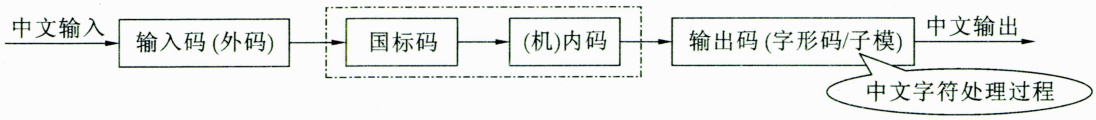
GB 码即国标码，是汉字交换码，是计算机与其他系统或设备间交换汉字信息的标准编码。采用两个字节对每个汉字进行编码，每个字节各取七位，可对 $2^7\times2^7=16384$ 个字符进行编码。

常见的 GB2312－80 按使用频度将汉字分为一级汉字和二级汉字，其中一级汉字按拼音排序，二级汉字按部首排序。国标码先把汉字排列在一个 94 行×94 列的方阵(二维表格)中，每一行称为"区"，每一列称为"位"，组成了一个共有 94 区，每个区有 94 位的字符集。由此引出了表示汉字的两种编码，一种为区位码，另一种为国标码。这两种编码都由两个字节组成，高字节表示"区"的代码，低字节表示"位"的代码。

区位码是用十进制数表示，国标码通常用十六进制表示。两者的换算关系是：区码和位码分别加上十进制数 32。如"国"字在表中的 25 行 90 列，区位码为 2590，国标码为 397AH。



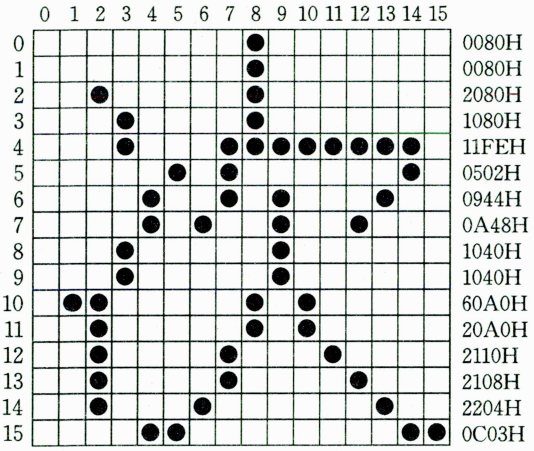
西文内码采用单字节的 ASCII 码，而汉字内码则是将区位码两个字节的最高位分别置为“1”，从而形成两个字节表示的汉字机内码。故通常一个英文字符占 1 个字节，一个中文字符占 2 个字节。



拓展延伸：全角与半角各在什么情况下使用？全角半角主要是针对标点符号来说的，英文全角标点占两个字节，半角占一个字节，而不管是半角还是全角，汉字都要占两个字节。

三、中文字符的编码表示—字形码

字形存储码是指供计算机输出汉字(显示或打印)用的二进制信息，也称字模。通常采用的是数字化点阵字模。例如下图中显示的“次”字，使用的是 16 * 16 点阵。



字模中每一点在存储器中使用一个二进制位(Bit)表示，如果是 1，则说明此处有点，若是 0，则说明没有。一个 16 * 16 点阵的汉字总共需要 16 * 16/8=32 个字节表示。在相同点阵中，不管其笔画繁简，每个汉字所占的字节数相等。字模的表示顺序为先从左到右，再从上到下，也就是先画左上方的 8 个点，再是右上方的 8 个点，然后是第二行左边 8 个点，右边 8 个点，依此类推，画满 16 * 16 个点。

四、Unicode 码

Unicode(统一码、万国码、通用码)为每种语言中的每个字符设定了统一并且唯一的二进制编码，以满足跨语言、跨平台进行文本转换、处理的要求，是国际组织制定的可以容纳世界上所有文字和符号的字符编码方案。UTF-8 和 UTF-16 是两种当前比较流行的 Unicode 编码方式，而 UTF-8 是目前互联网上使用最广泛的一种 Unicode 编码方式。

课堂练习

1.【NOIP2009】关于 ASCII，下面说法正确的是(　　)。

A. ASCII 码就是键盘上所有键的唯一编码

B. 一个 ASCII 码使用一个字节的内存空间就能够存放

C. 最新扩展的 ASCII 编码方案包含了汉字和其他欧洲语言的编码

D. ASCII 码是英国人主持制定并推广使用的

【答案】B

【分析】ASCII 是美国标准信息交换码，由一个字节保存，扩展的 ASCII 用两个字节保存，汉字编码不是 ASCII 的内容。

2.【NOIP2009】已知大写字母 A 的 ASCII 编码为 65(十进制)，则大写字母 J 的十进制 ASCII 编码为(　　)。

A. 71　　B. 72　　C. 73　　D. 以上都不是

【答案】D

【分析】J 的 ASCII 编码为 65+9=74。

3.【NOIP2011】字符“0”的 ASCII 码为 48，则字符“9”的 ASCII 码为(　　)。

A. 39　　B. 57　　C. 120　　D. 视具体的计算机而定

【答案】B

【分析】‘9’的 ASCII 码=‘0’的 ASCII 码+9=48+9=57。

4.【NOIP2011】字符“A”的 ASCII 码为十六进制 41，则字符“Z”的 ASCII 码为十六进制的(　　)。

A. 66　　B. 5A　　C. 50　　D. 视具体的计算机而定

【答案】B

【分析】考虑“Z”−“A”=$(25)_{10}=(19)_{16}$，$(41)_{16}+(19)_{16}=(5A)_{16}$。

5.【NOIP2013】(　　)是一种通用的字符编码，它为世界上绝大部分语言设定了统一并且唯一的二进制编码，以满足跨语言、跨平台的文本交换。目前它已经收录了超过十万个不同字符。

A. ASCII　　B. Unicode　　C. GBK2312　　D. BIG5

【答案】B

【分析】Unicode 是一种通用的字符编码，它为世界上绝大部分语言设定了统一并且唯一的二进制编码，以满足跨语言、跨平台的文本交换。目前 Unicode 已经收录了超过十万个不同的字符。

6.【NOIP2008】在 32 * 32 点阵的“字库”中，汉字“北”与“京”的字模占用字节数之和是(　　)。

A. 512　　B. 256　　C. 384　　D. 128

【答案】B

【分析】一个汉字是$\frac{32*32}{8}=128$ 个字节。

7.【NOIP2010】一个字节(byte)由(　　)个二进制位组成。

A. 8　　B. 16　　C. 32　　D. 以上都有可能

【答案】A

【分析】字节(byte)是计算机信息技术用于计量存储容量和传输容量的一种计量单位,1 个字节等于 8 位二进制。

8.【NOIP2011】一片容量为 8GB 的 SD 卡能存储大约(　　)张大小为 2MB 的数码照片。

A. 1600　　B. 2000　　C. 4000　　D. 16000

【答案】C

【分析】8GB=8 * 1024MB,$\frac{8 * 1024}{2}=4096$。

9.【NOIP2012】矢量图(Vector Image)图形文件所占的存储空间较小,并且不论如何放大、缩小或旋转等都不会失真,是因为它(　　)。

A. 记录了大量像素块的色彩值来表示图像

B. 用点、直线或者多边形等基于数学方程的几何图元来表示图像

C. 每个像素点的颜色信息均用矢量表示

D. 把文件保存在互联网,采用在线浏览的方式查看图像

【答案】B

【分析】矢量图使用直线和曲线来描述图形,这些图形的元素是一些点、线、矩形、多边形、圆和弧线等,它们都是通过数学公式计算获得的。

10.【NOIP2013】一个 32 位整型变量占用(　　)个字节。

A. 4　　B. 8　　C. 32　　D. 128

【答案】A

【分析】一个字节(byte)长度是 8 位,因此一个 32 位整型变量需要占用 4 个字节。

11.【NOIP2014】1TB 代表的字节数量是(　　)。

A. 2 的 10 次方　　B. 2 的 20 次方　　C. 2 的 30 次方　　D. 2 的 40 次方

【答案】D

【分析】本题考查对数据单位大小的理解:1TB=2^{10}GB,1GB=2^{10}MB,1MB=2^{10}KB,1KB=2^{10}B,所以,1TB=2^{10}GB=2^{20}MB=2^{30}KB=2^{40}B。

12.【NOIP2018】1MB 等于(　　)。

A. 1000 字节　　B. 1024 字节　　C. 1000×1000 字节　　D. 1024×1024 字节

【答案】D

【分析】本题考查计算机基础知识,1MB=2^{10}KB=2^{20}B,即 1024 * 1024 字节。

13.【NOIP2015】在计算机内部用来传送、存贮、加工处理的数据或指令都是以(　　)形式进行的。

A. 二进制码　　B. 八进制码　　C. 十进制码　　D. 智能拼音码

【答案】A

【分析】考查计算机基础知识。在计算机内部用来传送、存贮、加工处理的数据或指令均是

以二进制形式进行的。

14.【NOIP2016】如果256种颜色用二进制编码来表示，至少需要(　　)位。

A. 6　　B. 7　　C. 8　　D. 9

【答案】C

【分析】本题考查二进制的基本知识。$256=2^8$，故需要至少8位编码。

15.【NOIP2017】计算机存储数据的基本单位是(　　)。

A. bit　　B. Byte　　C. GB　　D. KB

【答案】B

【分析】注意计算机领域当中基本存储单位与最小存储单位是不同的概念，基本存储单位是字节Byte，最小存储单位是bit，而本题考查的是基本存储单位。

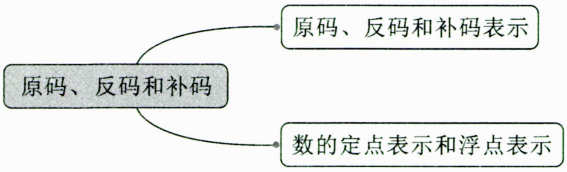
16.【NOIP2017】分辨率为800×600、16位色的位图，存储图像信息所需的空间为(　　)。

A. 937.5KB　　B. 4218.75KB　　C. 4320KB　　D. 2880KB

【答案】A

【分析】考察计算机科学基础知识。16位的位图表示每一个像素单元占用16位的存储空间，因此800 * 600的图片占用空间为$\frac{800*600*16}{8*1024}$KB=937.5KB。

第7节 原码、反码和补码



一、机器数和真值

在学习原码、反码和补码之前，需要先了解机器数和真值的概念。

1.机器数

一个数在计算机中的二进制表示形式，叫做这个数的机器数。机器数是带符号的，在计算机用一个数的最高位存放符号，正数为0，负数为1。

比如，十进制中的数+3，计算机字长为8位，转换成二进制就是00000011。如果是-3，就是10000011。

那么，这里的00000011和10000011就是机器数。

2.真值

机器数的第一位是符号位，后边才是真正的数值，所以机器数的形式值就不等于真正的数值。例如上面的有符号数10000011，其最高位1代表负，其真正数值是-3而不是形式值131(10000011转换成十进制等于131)。所以，为区别起见，将带符号位的机器数对应的真正数值称为机器数的真值。

例：

00000001的真值=+0000001=+1，

10000001的真值=-0000001=-1。

二、数的原码、反码和补码表示

原码、反码、补码是计算机存储一个具体数字的编码方式。数值在计算机中是以补码的方式存储的。

1.原码

原码就是符号位加上真值的绝对值，即用第一位表示符号，其余位表示值。比如如果是8位二进制：

[+1]原=00000001，

[-1]原=10000001。

第一位是符号位。因为第一位是符号位，所以8位二进制数的取值范围就是：

[11111111,01111111]→[-127,127]。

2.反码

反码的表示方法为正数的反码是其本身。

负数的反码是在其原码的基础上，符号位不变，其余各个位取反。

[+1]=[00000001]原=[00000001]反

[-1]=[10000001]原=[11111110]反

3.补码

补码的表示方法为正数的补码就是其本身。

负数的补码是在其原码的基础上，符号位不变，其余各位取反，最后+1(即在反码的基础上+1)。

[+1]=[00000001]原=[00000001]反=[00000001]补，

[-1]=[10000001]原=[11111110]反=[11111111]补。

三、数的定点表示和浮点表示

在计算机中小数点一般有两种表示法：一种是小数点固定在某一位置的定点表示法；另一种是小数点的位置可任意移动的浮点表示法。相应于这两种表示的计算机分别称为定点计算机和浮点计算机。

1.定点表示法

机器中所有数的小数点位置是固定不变的，因而小数点就不必使用记号表示出来。实际上，小数点可固定在任意一个位置上。

2.浮点表示法

在数的定点表示法中，由于数的表示范围较窄常常不能满足各种数值问题的需要。为了扩大数的表示范围，方便用户使用，有些计算机常采用浮点表示法。表示一个浮点数，要用两部分：尾数和阶码。尾数用以表示数的有效数值；阶码用以表示小数点在该数中的位置。

计算机多数情况下采作浮点数表示数值，它与科学计数法相似，把一个二进制数通过移动小数点位置表示成阶码和尾数两部分：

$$N=2^E\times S$$

其中 E 是 N 的阶码(Expoent)，是有符号的整数。S 是 N 的尾数(Mantissa)，是数值的有效数字部分，一般规定取二进制定点纯小数形式。

例：$1011101B=2^{+7}*0.1011101$，$101.1101B=2^{+3}*0.1011101$，$0.01011101B=2^{-1}*0.1011101$

课堂练习

1.【NOIP2009】在字长为16位的系统环境下，一个16位带符号整数的二进制补码为1111111111101101。其对应的十进制整数应该是(　　)

A. 19　　B. -19　　C. 18　　D. -18

【答案】B

【分析】由补码的定义可知二进制数$(1111111111101101)_2$与其十进制真值X的关系为$2^{16}+X=(1111111111101101)_2$，$X=-[2^{16}-(1111111111101101)_2]=-(65536-65517)=-19$。

2.【NOIP2010】一个字长为8位的整数的补码是11111001，则它的原码是(　　)。

A. 00000111　　B. 01111001　　C. 11111001　　D. 10000111

【答案】D

【分析】正数的补码与原码相同。负数的补码负号位为1，其余位为该数绝对值的原码按位取反然后整个数加1。11111001的首位1，作为符号位代表该数是负数，所以保留该位，1111001减1然后取反，再加上符号位1，结果是10000111。

3.【NOIP2013】把64位非零浮点数强制转换成32位浮点数后，不可能(　　)。

A. 大于原数　　B. 小于原数　　C. 等于原数　　D. 与原数符号相反

【答案】D

【分析】根据IEEE的规范，浮点数由符号位、尾码和阶码组成，在将64位浮点数强制转换成32位浮点数时，符号位保留不变，尾码和阶码则可能由于精度问题改变它们的值。因此大于、小于、等于原数的情况都有可能出现。

4.【NOIP2014】下列各无符号十进制整数中，能用八位二进制表示的数中最大的是(　　)。

A. 296　　B. 133　　C. 256　　D. 199

【答案】D

【分析】无符号八位二进制数的表示范围为$0\sim2^8-1$，即0～255，因此选项A与选项C均不能表示。

5.【NOIP2017】在8位二进制补码中，10101011表示的数是十进制下的(　　)。

A. 43　　B. −85　　C. −43　　D. −84

【答案】B

【分析】考查补码的概念以及进制转换。首位是1表示为负数，可以参看第一章第9节内容。负数补码的原码为补码减1后取反，也就是说原码应该为补码取反后加1，也就是$(1010101)_2$，对应的数值为85，所以该数字为−85。

6.【NOIP2010】2E+03表示(　　)。

A. 2.03　　B. 5　　C. 8　　D. 2000

【答案】D

【分析】当我们要标记或运算某个较大或较小且位数较多时，用科学记数法来简化写法，科学记数法的形式为aeb=a * 10^b，a能够用浮点数表示，b不能用浮点数表示。

7.【NOIP2011】计算机中的数值信息分为整数和实数(浮点数)。实数之所以能表示很大或者很小的数，是由于使用了(　　)。

A. 阶码　　B. 补码　　C. 反码　　D. 较长的尾数

【答案】A

【分析】一个浮点数在IEEE754标准下由三个部分组成：符号位(1位)+阶码(指数位，由补码表示，在C++里single有8位，double有11位)+有效数字位(尾数部分，在C++里single有23位，double有52位)。

转化为正常的十进制小数计算过程如下：

$$\text{Value}=(-1)^{\text{符号位}} * (1.\text{有效数字位})_2 * 2^{\text{指数位}}$$

就是说如果转化正浮点数先将(1.有效数字位)转成十进制，然后乘上$2^{\text{指数位}}$，能够看出指数位能够表达的范围非常宽(C++里single $2^{-128}\sim2^{127}$，double $2^{-1024}\sim2^{1023}$)，所以这就是为什么能够表达范围很宽的数，如果说要求很高的精度，那么有效数字位(尾数)的长度就得要求很高。

不定项选择题

1.【NOIP2010】在整数的补码表示法中，以下说法正确的是(　　)。

A. 只有负整数的编码最高位为 1

B. 在编码的位数确定后，所能表示的最小整数和最大整数的绝对值相同

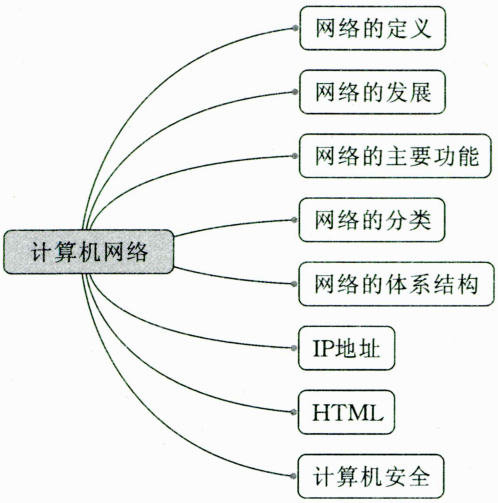
C. 整数 0 只有一个唯一的编码

D. 两个用补码表示的数相加时，若在最高位产生进位，则表示运算溢出

【答案】AC

【分析】8 位所对应的范围为－128～127。－1＋(－1)用补码表示为 11111111＋11111111＝11111110，11111110 是－2 的补码。

第8节 计算机网络



一、网络的定义

计算机网络利用通信线路和设备，把分布在不同地理位置上的多台计算机连接起来，是现代通信技术与计算机技术相结合的产物。

网络中计算机与计算机之间的通信依靠协议进行。协议是计算机收、发数据的规则。TCP/IP 协议族：用于网络的一组通讯协议，包括 IP、TCP、HTTP、SMTP、POP、IMAP、MIME、FTP 等协议。其中电子邮件程序使用的 TCP/IP 协议：使用 SMTP 来发送邮件、使用 IMAP 连接到邮件服务器、使用 POP 从邮件服务器下载邮件。

二、网络的发展

计算机网络的发展过程大致可以分为四个阶段：

第一阶段：计算机网络萌芽阶段

1946 年 2 月 14 日，世界上第一台计算机 ENIAC 在美国诞生。其后的 20 多年，计算机技术一直在寻找关于与通信技术相结合方面的发展，20 世纪 50 年代初，美国在本土和加拿大境内，建立了一个半自动地面防控系统，简称 SAGE(赛其)系统，可以说是网络的雏形。

第二阶段：计算机网络诞生阶段

1969 年，美国国防部高级研究计划署(ARPA)建立 ARPANet。10 月 29 日，在加州大学洛杉矶分校和斯坦福研究所进行互联实现，随后扩展为四校，加入加州大学圣巴巴分校，犹他大学。随后发展成 15 个站点，23 台主机，1973 年，ARPANet 扩展成为国际互联网。

第三阶段：计算机网络协议标准产生与确立

ISO 国际化标准组织经过十年的努力，建立了 OSI 开放式系统互联架构，确立了 TCP/IP 作为通用协议。TCP 负责发现传输的问题，一有问题就发出信号，要求重新传输，直到所有数据安全正确地传输到目的地。而 IP 是给因特网的每一台联网设备规定一个地址。

第四阶段：internet 国际互联

1986 年，NSF 建立了国家科学基金网，实现了全美互联；

1987 年 9 月 14 日，CANET 与德国卡尔斯鲁厄大学互联，向世界发出第一封电子邮件：

Accross Great wall we can reach every corner in the world(跨越长城,走向世界);

1993 年 6 月,美国提出 NII 计划,建立信息高速公路;

1994 年 4 月,中国正式加入信息高速公路;

1997 年 4 月,中国建设了四大网络(中国公用互联网、中国教育科研网、中国科学技术网、中国金桥网)。

三、网络的主要功能

1.资源共享。

2.信息传输。

3.分布式处理。

4.综合信息服务。

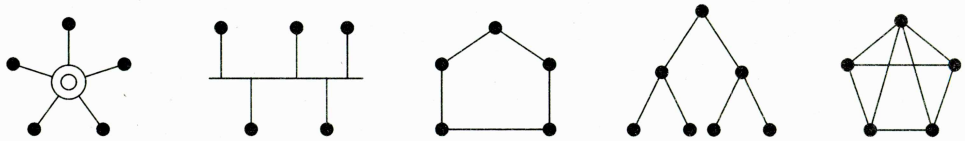
四、网络的分类

1.按网络的地理范围分类

网络按地理范围分为局域网(LAN,1Km 内)、城域网(MAN,几千米至几万米)、广域网(WAN)。Internet(因特网)是当今世界上规模最大、用户最多、影响最广泛的计算机互联网络。以太网泛指所有采用载波监听多路访问协议,即 CSMA/CD 协议的局域网。常见的局域网有以太网、Novell 网、令牌环网、FDDI 网、ATM 网等。

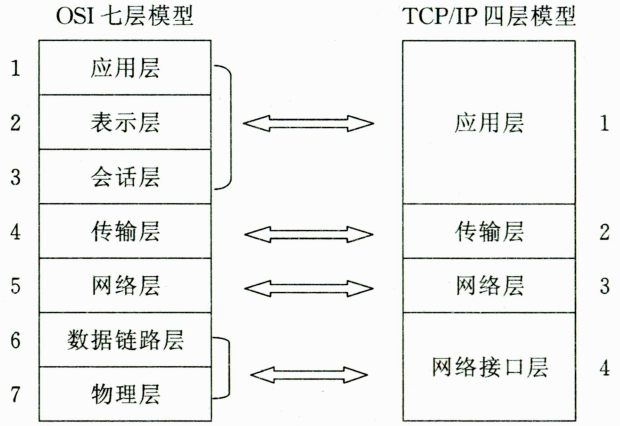
2.按网络的拓扑结构分类

网络按拓扑结构分为星形、总线形、环形、树形、网状形。网络的拓扑结构是指计算机作为网络节点和通信线路所组成的几何形状。



五、网络的体系结构

ISO(国际标准化组织)制定了一个国际标准 OSI(开放式通信系统互联参考模型),并作为一种理论下的模型。TCP/IP 被广泛使用,成为网络互联事实上的标准。



OSI七层网络模型	TCP/IP四层概念模型	对应网络协议
应用层(Application)	应用层	HTTP、TFTP、FTP、NFS、WAIS、SMTP
表示层(Presentation)		Telnet、Rlogin、SNMP、Gopher
会话层(Session)		SMTP、DNS
传输层(Transport)	传输层	TCP、UDP
网络层(Network)	网络层	IP、ICMP、ARP、RARP、AKP、UUCP
数据链路层(DateLink)	数据链路层	FDDI、Ethernet、Arpanet、PDN、SLIP、PPP
物理层(Physical)		IEEE801.1A、IEEE802.2到IEEE802.11

六、IP地址

Internet上的计算机地址有两种表示形式:IP地址与域名。

1.域名

IP地址是数字标识,不便记忆和书写,网络上的数字型IP地址相对应的字符型地址,就被称为域名。域名含有英文字母和数字,不区分大小写且完整域名不超过255个字符。

域名格式:主机名.机构名.网络名(2级域名).国家或行业域名(顶级域名)。

例如,www.cs.tsinghua.edu.cn//如果主机名被省略,系统默认是www。

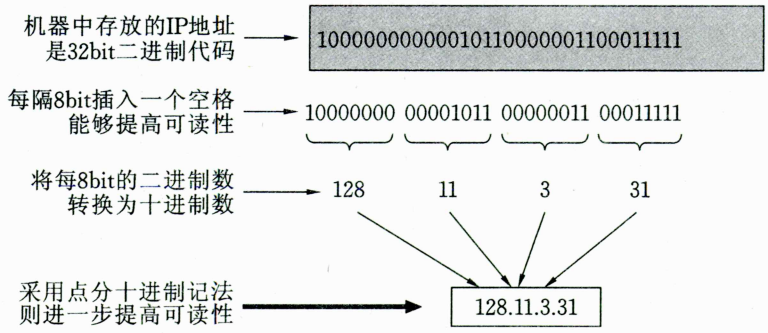
常见域名:com代表商业性的机构或公司;tech代表科技、技术;org代表非营利的组织、团体;gov代表政府部门;net代表从事Internet相关的机构或公司;edu代表教育机构。cn是中国,us是美国,jp是日本。

2.IP地址

IP地址的表达形式有IPV4和IPV6两类,IPV4网络使用32位地址,IPV6网络使用128位地址,目前常见的是IPV4。

IPV4地址表示:

IPV4地址长度为32位,以X1.X2.X3.X4格式表示,每个X可以表示为8位的二进制,而每个X的十进制的值范围为0~255。IPV4地址可以分为A、B、C、D、E共5类,其中A、B、C类是基本类,D和E类是多播和保留地址,都由网络地址与主机地址两部分组成。





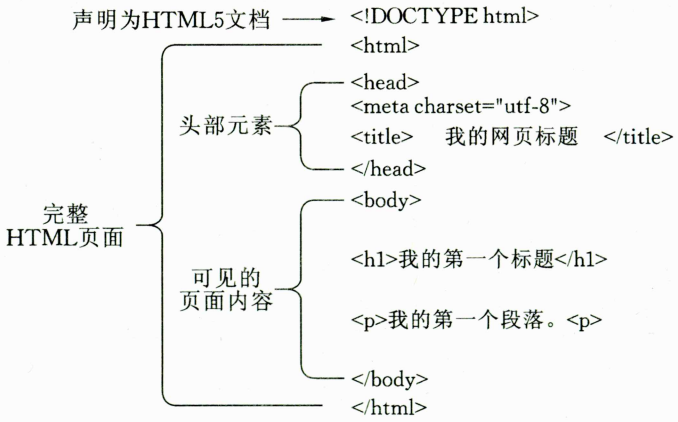
A: 0.0.0.0-127.255.255.255，0<=X1<128
B: 128.0.0.0-191.255.255.255，128<=X1<192
C: 192.0.0.0-223.255.255.255，192<=X1<224
D: 224.0.0.0-239.255.255.255，224<=X1<240
E: 240.0.0.0-255.255.255.255，240<=X1<255

下列地址中，属于B类IP地址的是 (C)
A.27.33.119.2
B.192.97.32.121
C.133.201.189.32
D.126.33.82.107
E.10.11.12.13

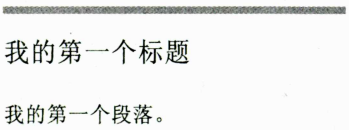
七、HTML

HTML 的全称为超文本标记语言，是一种标记语言。包括一系列标签，通过这些标签可以将网络上的文档格式统一，使分散的 Internet 资源连接为一个逻辑整体。HTML 文本是由 HTML 命令组成的描述性文本，HTML 命令可以说明文字、图形、动画、声音、表格、链接等。

下面图 a 是 HTML 网页的代码结构，图 b 是 HTML 网页代码在浏览器中的运行效果。



（a）示例代码



（b）网页效果

八、计算机安全

（一）计算机硬件安全

1.计算机在使用过程中，对外部环境有一定的要求。
2.计算机的芯片和硬件设备指令泄密。
3.硬件泄密。

(二)存储安全

计算机安全中最重要的是存储数据的安全,其主要面临的威胁包括计算机病毒、非法访问、计算机电磁辐射、硬件损坏等。

(三)计算机病毒

计算机病毒(Computer Virus)是在计算机程序中插入的破坏计算机功能,能自我复制的一组计算机指令或者程序代码。计算机病毒主要通过网络与磁盘传播,主要造成软件和数据的损坏,较为彻底的清除方法是格式化磁盘。

计算机病毒的特征:

1.繁殖性:计算机病毒可以像生物病毒一样进行繁殖,并且是否具有繁殖、感染的特征是判断某段程序为计算机病毒的首要条件。

2.破坏性:计算机中毒后,可能会导致正常的程序无法运行,文件被删除或受到不同程度的损坏。

3.传染性:通过修改别的程序将自身的复制品或其变体传染到其他无毒的对象上,这些对象可以是一个程序也可以是系统中的某一部件。

4.潜伏性:计算机病毒依附于其他媒体寄生的能力,侵入后的病毒潜伏到条件成熟才发作,可使电脑变慢。

5.隐蔽性:计算机病毒具有很强的隐蔽性,时隐时现、变化无常,处理起来非常困难。

6.可触发性:病毒一般都设定了触发条件,例如系统时钟的某个时间、系统运行了某些程序等。

课堂练习

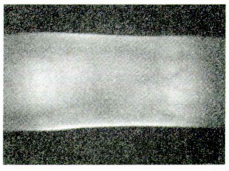
1.【NOIP2008】Web2.0 是近年来互联网的热门概念之一,其核心思想是互动与分享。下列网站中,(　　)是典型的 Web2.0 应用。

A. Sina　　B. Flickr　　C. Yahoo　　D. Google

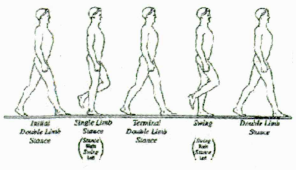
【答案】B

【分析】Web2.0 是相对 web1.0 的新一类互联网应用的统称。Web1.0 的主要特点在于用户通过浏览器获取信息。Web2.0 则更注重用户的交互作用,用户既是网站内容的浏览者,也是网站内容的制造者。Flickr,雅虎旗下图片分享网站,是一家提供免费及付费数位照片储存、分享方案线上服务,也提供网络社群服务的平台。一般认为 Flickr 是 Web2.0 应用方式的绝佳例子。除了许多使用者在 Flickr 上分享他们的私人照片,该服务也可作为网络图片的存放空间,受到许多网络作者的喜爱。

2.【NOIP2011】生物特征识别,是利用人体本身的生物特征进行身份认证的一种技术。目前,指纹识别、虹膜识别、人脸识别等技术已广泛应用于政府、银行、安全防卫等领域。以下不属于生物特征识别技术及其应用的是(　　)。



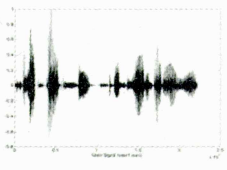
A. 指静脉验证



B. 步态验证



C. ATM 机密码验证



D. 声音验证

【答案】C

【分析】C是属于物理验证。

3.【NOIP2012】蓝牙和Wi-Fi都是(　　)设备。

A. 无线广域网　　B. 无线城域网　　C. 无线局域网　　D. 无线路由器

【答案】C

【分析】无线局域网利用射频的技术,使用电磁波在空中进行通信连接。

4.【NOIP2012】以下不属于目前3G(第三代移动通信技术)标准的是(　　)。

A. GSM　　B. TD-SCDMA　　C. CDMA2000　　D. WCDMA

【答案】A

【分析】3G系统的三大主流标准分别是WCDMA(宽带CDMA),CDMA2000和TD-SCDMA(时分双工同步CDMA)。

5.【NOIP2012】(　　)是主要用于显示网页服务器或者文件系统的HTML文件内容,并让用户与这些文件交互的一种软件。

A. 资源管理器　　B. 浏览器　　C. 电子邮件　　D. 编译器

【答案】B

【分析】HTML(Hyper Text Markup Language)超级文本标记语言,阅读工具为浏览器。

6.【NOIP2012】(　　)是目前互联网上常用的E-mail服务协议。

A. HTTP　　B. FTP　　C. POP3　　D. Telnet

【答案】C

【分析】HTTP(Hyper Text Transfer Protocol)超文本传输协议,FTP(File Transfer Protocol)文件传输协议,POP3(Post Office Protocol-Version 3)邮局协议版本3,Telnet协议是TCP/IP协议族中的一员,是Internet远程登录服务的标准协议和主要方式。

7.【NOIP2012】仿生学的问世开辟了独特的科学技术发展道路。人们研究生物体的结构、功能和工作原理,并将这些原理移植于新兴的工程技术之中。以下关于仿生学的叙述,错误的是(　　)。

A. 由研究蝙蝠,发明雷达　　B. 由研究蜘蛛网,发明因特网

C. 由研究海豚,发明声呐　　D. 由研究电鱼,发明伏特电池

【答案】B

【分析】蜘蛛网和因特网除了模型相似,其他没有任何必然联系。因特网始于1969年的美国,又称互联网。是美军在ARPA(阿帕网,美国国防部研究计划署)制定的协定下将美国西南部的大学UCLA(加利福尼亚大学洛杉矶分校)、Stanford Research Institute(斯坦福大学研究学院)、UCSB(加利福尼亚大学)和University of Utah(犹他州大学)的四台主要的计算机连接起来。这个协定由剑桥大学的BBN和MA执行,在1969年12月开始联机。

8.【NOIP2013】通常在搜索引擎中,对某个关键词加上双引号表示(　　)。

A. 排除关键词,不显示任何包含该关键词的结果

B. 将关键词分解,在搜索结果中必须包含其中的一部分

C. 精确搜索,只显示包含整个关键词的结果

D. 站内搜索,只显示关键词所指向网站的内容

【答案】C

【分析】通常在搜索引擎中,对某个关键词加上双引号表示关键搜索,只显示包含整个关键

词的结果。

9.【NOIP2013】中国的国家顶级域名是(　　)。

A. .cn　　B. .ch　　C. .chn　　D. .china

【答案】A

【分析】中国的国家顶级域名是.cn。

10.【NOIP2014】以下哪一种属于电子邮件收发的协议(　　)。

A. SMTP　　B. UDP　　C. P2P　　D. FTP

【答案】A

【分析】SMTP 全称 Simple Mail Transfer Protocol,即简单邮件传输协议。UDP 是 User Datagram Protocol,是与 TCP 协议相对应的一种高效但不可靠的协议。P2P 是 Peerto Peer,为无专用集中服务器的对等传输协议。FTP 是 File Transfer Protocol,为文件传输协议。

11.【NOIP2015】FTP 可以用于(　　)。

A. 远程传输文件　　B. 发送电子邮件　　C. 浏览网页　　D. 网上聊天

【答案】A

【分析】FTP 是 File Transfer Protocol(文件传输协议)的英文简称,而中文简称为"文传协议",用于 Internet 上的控制文件的双向传输。

12.【NOIP2016】以下不属于无线通信技术的是(　　)。

A. 蓝牙　　B. Wi-Fi　　C. GPRS　　D. 以太网

【答案】D

【分析】本题考察计算机科学基础知识。以太网使用有线连接,比如双绞线。

13.【NOIP2017】下列协议中与电子邮件无关的是(　　)。

A. POP3　　B. SMTP　　C. WTO　　D. IMAP

【答案】C

【分析】WTO 是世界贸易组织,与本题无关。其他三个选项为 POP3:Post Office Protocol-Version 3,邮局协议 3;SMTP:Simple Mail Transfer Protocol,简单邮件传输协议;IMAP:Internet Mail Access Protocol,因特网邮件访问协议。

14.【NOIP2018】广域网的英文缩写是(　　)。

A. LAN　　B. WAN　　C. MAN　　D. LNA

【答案】B

【分析】考查计算机基础知识。WAN(Wide Area Network)是广域网的缩写。

15.【NOIP2008】TCP/IP 是一组构成互联网基础的网络协议,字面上包括两组协议:传输控制协议(TCP)和网际协议(IP)。TCP/IP 协议把 Internet 网络系统描述成具有四个层次功能的网络模型,其中提供源节点和目的节点之间的信息传输服务,包括寻址和路由器选择等功能的是(　　)。

A. 链路层　　B 网络层　　C. 传输层　　D. 应用层　　E.会话层

【答案】B

【分析】TCP/IP 协议把 Internet 网络系统描述成具有四个层次功能的网络模型,从下往上

分别是网络接口层、网络层、传输层、应用层。其中提供源节点和目的节点之间的信息传输服务，包括寻址和路由器选择等功能的是网络层。

16.【NOIP2012】无论是 TCP/IP 模型还是 OSI 模型，都可以视为网络的分层模型，每个网络协议都会被归入某一层中。如果用现实生活中的例子来比喻这些“层”，以下最恰当的是(　　)。

A. 中国公司的经理与缅甸公司的经理交互商业文件

第 4 层	中国公司经理		缅甸公司经理
	↑ ↓		↑ ↓
第 3 层	中国公司经理秘书		缅甸公司经理秘书
	↑ ↓		↑ ↓
第 2 层	中国公司翻译		缅甸公司翻译
	↑ ↓		↑ ↓
第 1 层	中国邮递员	←→	缅甸邮递员

B. 军队发布命令

第 4 层	司令						
	↓						
第 3 层	军长 1				军长 2		
	↓				↓		
第 2 层	师长 1		师长 2		师长 3		师长 4
	↓		↓		↓		↓
第 1 层	团长 1	团长 2	团长 3	团长 4	团长 5	团长 6	团长 7	团长 8

C. 国际会议中，每个人都与他国地位对等的人直接进行会谈

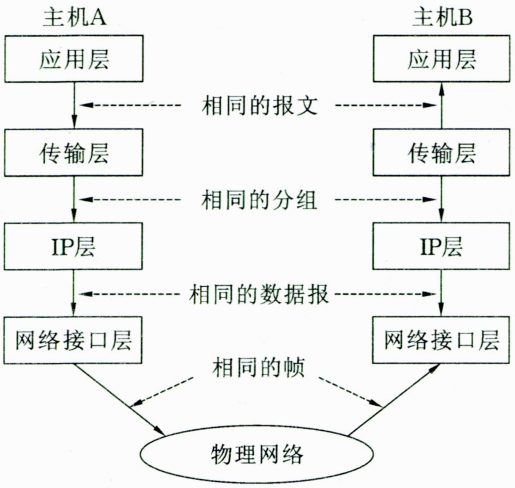
第 4 层	英国女王	←→	瑞典国王
第 3 层	英国首相	←→	瑞典首相
第 2 层	英国外交大臣	←→	瑞典外交大臣
第 1 层	英国驻瑞典大使	←→	瑞典驻英国大使

D. 体育比赛中，每一级比赛的优胜者晋级上一级比赛

第 4 层	奥运会
	↑
第 3 层	全运会
	↑
第 2 层	省运会
	↑
第 1 层	市运会
	↑

【答案】A

【分析】TCP/IP 模型。



根据四个选项，A 项最符合该图的形式。

17.【NOIP2014】TCP 协议属于哪一层协议(　　)。

A. 应用层　　B. 传输层　　C. 网络层　　D. 数据链路层

【答案】B

【分析】在因特网协议族(Internet protocol suite)中，TCP 层是位于 IP 层之上，应用层之下的传输层。

18.【NOIP2009】关于互联网，下列说法正确的是(　　)。

A. 新一代互联网使用的 IPv6 标准是 IPv5 标准的升级与补充

B. 互联网的入网主机如果有了域名就不再需要 IP 地址

C. 互联网的基础协议为 TCP/IP 协议

D. 互联网上所有可下载的软件及数据资源都是可以合法免费使用的

【答案】C

【分析】IPv6 与 IPv5 一点关系也没有，IPv6 是 IPv4 的替代版本；IP 是必须的，域名是为了方便记忆；D 中注意盗版是非法的。

19.【NOIP2013】IPv4 协议使用 32 位地址，随着其不断被分配，地址资源日趋枯竭。因此，它正逐渐被使用(　　)位地址的 IPv6 协议所取代。

A. 40　　B. 48　　C. 64　　D. 128

【答案】D

【分析】IPv4 协议使用 32 位地址，IPv6 协议则使用 128 位地址，可以有效解决地址资源枯竭的问题。

20.【NOIP2014】下列几个 32 位 IP 地址中，书写错误的是(　　)。

A. 162.105.128.27　　B. 192.168.0.1　　C. 256.256.129.1　　D. 10.0.0.1

【答案】C

【分析】32 位 IP 地址中的 4 段数据均为八位二进制数，因此每段数字的范围均在 0～255。

21.【NOIP2009】关于 HTML，下列说法正确的是(　　)。

A. HTML 实现了文本、图形、声音乃至视频信息的统一编码

B. HTML 全称为超文本标记语言

C. 网上广泛使用的 Flash 动画都是由 HTML 编写的

D. HTML 也是一种高级程序设计语言

【答案】B

【分析】文本、图形甚至声音和视频的编码没有统一；Flash 是由 Adobe 公司的 Flash 软件制作；HTML 只不过是组合成一个文本文件的一系列标签，不是高级语言。

22.【NOIP2010】下列 HTML 语句中，正确产生一个指向 NOI 官方网站的超链接是(　　)。

A.<a url="http://www.noi.cn"欢迎访问 NOI 网站

B. 欢迎访问 NOI 网站

C.<a>http://www.noi.cn

D.欢迎访问 NOI 网站

【答案】B

【分析】HTML 中超链接代码部分的基本常识。

23.【NOIP2011】为解决 Web 应用中的不兼容问题，保障信息的顺利流通，(　　)制定了一系列标准，涉及 HTML、XML、CSS 等，并建议开发者遵循。

A. 微软　　B. 美国计算机协会(ACM)

C. 联合国教科文组织　　D. 万维网联盟(W3C)

【答案】D

【分析】W3C 最重要的工作是发展 Web 规范(称为推荐，Recommendations)，这些规范描述了 Web 的通信协议(比如 HTML 和 XHTML)和其他的构建模块。

24.【NOIP2015】计算机病毒是(　　)。

A. 通过计算机传播的危害人体健康的一种病毒

B. 人为制造的能够侵入计算机系统并给计算机带来故障的程序或指令集合

C. 一种由于计算机元器件老化而产生的对生态环境有害的物质

D. 利用计算机的海量高速运算能力而研制出来的用于疾病预防的新型病毒

【答案】B

【分析】考察计算机科学基础知识。计算机病毒(Computer Virus)是编制者在计算机程序中插入的破坏计算机功能或者数据的代码，能影响计算机使用，能自我复制的一组计算机指令或者程序代码。

不定项选择题

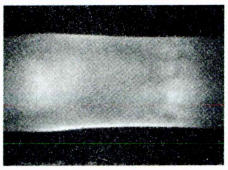
1.【NOIP2016】可以将单个计算机接入到计算机网络中的网络接入通讯设备有(　　)。

A. 网卡　　B. 光驱　　C. 鼠标　　D. 显卡

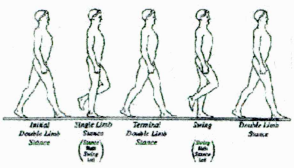
【答案】A

【分析】网络接口控制器(network interface controller,NIC),又称网络接口控制器,网络适配器(network adapter),网卡(network interface card),或局域网接收器(LAN adapter),是一块被设计用来允许计算机在计算机网络上进行通讯的计算机硬件。

2.【NOIP2011】生物特征识别,是利用人体本身的生物特征进行身份认证的一种技术。目前,指纹识别、虹膜识别、人脸识别等技术已广泛应用于政府、银行、安全防卫等领域。以下属于生物特征识别技术及其应用的是(　　)。



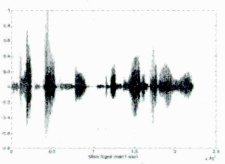
A. 指静脉验证



B. 步态验证



C. ATM 机密码验证



D. 声音验证

【答案】ABD

【分析】C 属于物理验证。

3.【NOIP2011】为计算机网络中进行数据交换而建立的规则、标准或约定的集合成为网络协议。下列英文缩写中,(　　)是网络协议。

A. HTTP　　B. TCP/IP　　C. FTP　　D. WWW

【答案】ABC

【分析】超文本传输协议(HTTP,Hyper Text Transfer Protocol)是互联网上应用最为广泛的一种网络协议;TCP/IP,Transmission Control Protocol/Internet Protocol 的简写,中译名为传输控制协议/因特网互联协议,又名网络通讯协议,是 Internet 最基本的协议、Internet 国际互联网络的基础,由网络层的 IP 协议和传输层的 TCP 协议组成;FTP 是 File Transfer Protocol(文件传输协议)的英文简称,而中文简称为"文传协议",用于 Internet 上的控制文件的双向传输;WWW 是环球信息网的缩写(亦作"Web""WWW""W3",英文全称为"World Wide Web"),中文名字为"万维网""环球网"等,常简称为 Web,它不是一种协议。

4.【NOIP2012】以下(　　)属于互联网上的 E-mail 服务协议。

A. HTTP　　B. FTP　　C. POP3　　D. SMTP

【答案】CD

【分析】HTTP(Hyper Text Transfer Protocol)超文本传输协议、FTP(File Transfer Protocol)文件传输协议、POP3(Post Office Protocol-Version 3)邮局协议版本 3、SMTP(Simple Mail Transfer Protocol)即简单邮件传输协议。

5.【NOIP2016】以下属于无线通信技术的有(　　)。

A. 蓝牙　　B. Wi-Fi　　C. GPRS　　D. 以太网

【答案】ABC

【分析】计算机基础知识，以太网是有线局域网。GPRS(General Packet Radio Service)是通用分组无线服务技术的简称，它是GSM移动电话用户可用的一种移动数据业务，属于第二代移动通信中的数据传输技术。

6.【NOIP2009】关于计算机网络，下列说法正确的是(　　)。

A. 网络协议之所以有很多层，主要是由于新技术需要兼容过去老的实现方案

B. 新一代互联网使用的IPv6标准是IPv5标准的升级与补充

C. TCP/IP是互联网的基础协议簇，包含有TCP和IP等网络与传输层的通讯协议

D. 互联网上每一台入网主机通常都需要使用一个唯一的IP地址，否则就必须注册一个固定的域名来标明其地址

【答案】C

【分析】网络协议分层不是为了兼容，而是根据网络分层模型来的。新的IPv6是IPv4的升级。必须要有IP地址才能进入互联网。域名本质上也是IP，只是用一种更容易方便记忆的方式，相当于C++里typedef的作用。

7.【NOIP2015】下列选项不是正确的IP地址的有(　　)。

A. 202.300.12.4　　B. 192.168.0.3　　C. 100:128:35:91　　D. 111-119-35-21

【答案】ACD

【分析】32位IP地址中的4段数据均为八位二进制数，因此每段数字的范围均在0～255，段与段之间用'.'隔开。

8.【NOIP2009】关于HTML，下列说法正确的是(　　)。

A. HTML全称超文本标记语言，实现了文本、图形、声音乃至视频信息的统一编码

B. HTML不单包含有网页内容信息的描述，同时也包含对网页格式信息的定义

C. 网页上的超链接只能指向外部的网络资源，本网站网页间的联系通过设置标签来实现

D. 点击网页上的超链接从本质上就是按照该链接所隐含的统一资源定位符(URL)请求网络资源或网络服务

【答案】BD

【分析】HTML没有实现统一编码。本网站页面间也可以用超链接。所以本题应选BD。

9.【NOIP2008】在下列防火墙(firewall)的说法中，正确的有(　　)。

A. 防火墙是一项协助确保信息安全的设备，其会依照特定的规则，允许或是限制数据通过

B. 防火墙可能是一台专属的硬件或是安装在一般硬件上的一套软件

C. 网络层防火墙可以视为一种IP数据包过滤器，只允许符合特定规则的数据包通过，其余的一概禁止穿越防火墙

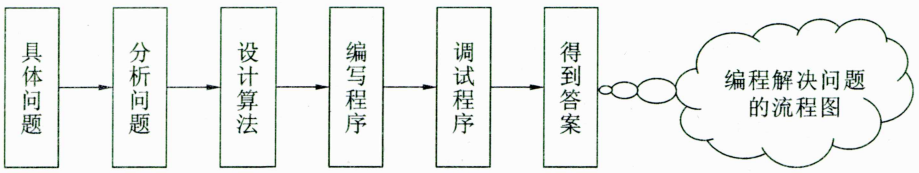
D. 应用层防火墙是在TCP/IP的"应用层"上工作，可以拦截进出某应用程序的所有数据包

【答案】ABCD

【分析】根据防火墙的设计功能可知，四个选项都是正确的。

第二章　程序设计基础知识

“程序＝算法＋数据结构”，其中算法可以理解为解决实际问题的方法，数据结构则是计算机存储、组织数据的方式。



算法是程序设计的核心和灵魂，有了算法就可以结合数据结构转变成程序。

第 1 节　程序基本常识

一、算法的定义及特征

算法就是解决问题的操作步骤。一个算法必须满足以下五个重要的特征：

1.有穷性：执行有穷步，在有穷的时间内完成。

2.确切性：每一条指令必须有确切的含义，不会产生歧义。在任何条件下算法只有唯一的一条执行路径。

3.可行性：算法中的操作可以通过执行有限次来实现。

4.输入：一个算法有零个或者多个输入。

5.输出：一个算法有一个或者多个输出。

二、算法的复杂度

同一个问题可用不同的算法来解决，而一个算法质量的优劣将影响到算法乃至程序的效率。算法分析的目的在于选择合适的算法和改进算法。一个算法的评价主要从时间复杂度和空间复杂度来考虑。

（一）空间复杂度

空间复杂度指执行算法所需占用的内存空间。算法执行时所需的存储空间包括程序本身占用的空间、输入数据占用的空间以及算法执行时所需空间。

算法在时间的高效性和空间的高效性之间通常是矛盾的，所以一般会取一个平衡点。通常假设程序运行在足够大的内存空间中，所以研究更多的是算法的时间复杂度。

（二）时间复杂度

1.定义及表示

指算法执行时所需消耗时间，通常用算法执行次数来衡量，记作 T(n)＝O(f(n))，其中 f(n)是算法执行次数的函数。用大写 O()来体现算法时间复杂度的记法，称之为大 O 记法。

2.计算步骤

(1)找出基本操作确定规模 n。通常取最深层循环内的语句所描述的操作作为基本操作。

(2)计算执行次数的函数 f(n)并求出 T(n)=O(f(n))。O(f(n))是取 f(n)中 n 增长最快的那项系数为 1 的形式。

例 1:求出以下算法的时间复杂度。

```
void fundemo (int n){
    int i=1,j=100;
    while (i<n){
        i+=2;
        ++j;
    }
}
```

分析:该算法基本操作为 i+=2 和++j,规模为 n。i 最后的值为 1+2m(m 代表执行次数),则 1+2m+k=n(k 是足够小的数),m=(n−1−k)/2,故 f(n)=(n−1−k)/2,此时 f(n)增长最快的那项是 n,故 T(n)=O(f(n))=O(n)。

3.计算规则

(1)加法规则

$T(n,m)=T1(n)+T2(m)=O(max\{f(n),g(m)\})$　　//并列算法

(2)乘法规则

$T(n,m)=T1(n)*T2(m)=O(f(n)*g(m))$　　//嵌套算法

(3)比较规则

$O(1)<O(\log n)<O(n)<O(n\log n)<O(n^2)<O(n^3)<\cdots<O(n^K)<O(2^n)<O(n!)<O(n^n)$

课堂练习

1.【NOIP2011】在使用高级语言编写程序时,一般提到的“空间复杂度”中的“空间”是指(　　)。

A. 程序运行时理论上所占的内存空间　　B. 程序运行时理论上所占的数组空间

C. 程序运行时理论上所占的硬盘空间　　D. 程序源文件理论上所占的硬盘空间

【答案】A

【分析】空间复杂度(Space Complexity)是对一个算法在运行过程中临时占用存储空间大小的量度,这个存储空间是指计算机内存。

2.【NOIP2013】斐波那契数列的定义如下:$F_1=1$,$F_2=1$,$F_n=F_{n-1}+F_{n-2}$(n≥3)。如果用下面的函数计算斐波那契数列的第 n 项,则其时间复杂度为(　　)。

```
int F(int n) {
    if (n<=2)
    return 1;
    else
    return F(n-1)+F(n-2);
}
```

A. O(1)　　B. O(n)　　C. $O(n^2)$　　D. $O(F_n)$

【答案】D

【分析】该函数可以理解成一棵二叉树,根节点的值等于两个子节点的和,叶子节点的值为

1。函数的最终返回值为 F_n，因此一共有 F_n 个叶子节点和 F_n-1 个非叶子节点，总的时间复杂度是 $O(F_n)$。

3.【NOIP2013】T(n)表示某个算法输入规模为 n 时的运算次数。如果 T(1)为常数，且有递归式 T(n)=2 * T(n/2)+2n，那么 T(n)=(　　)。

A. O(n)　　B. O(nlogn)　　C. $O(n^2)$　　D. $O(n^2\log n)$

【答案】B

【分析】题中的递归式，相当于归并排序的复杂度，每次分治成两个子问题，合并子问题的代价为 n，我们可以想象成共有 logn 层，每次代价总共需要 O(n)。因此总的时间复杂度为O(nlogn)。

$$T(n)=2*T\left(\frac{n}{2}\right)+2n=2*\left(2*T\left(\frac{n}{4}\right)+2*\frac{n}{2}\right)+2n=4*T\left(\frac{n}{4}\right)+2*2n=\cdots$$

$$=2^k*T\left(\frac{n}{2^k}\right)+k*2n=\cdots=2^{\log_2 n}*T(1)+2\ n\log_2 n=n+2\ n\log_2 n=O(n\log n)$$

4.【NOIP2015】设某算法的计算时间表示为递推关系式 T(n)=T(n−1)+n(n 为正整数)及 T(0)=1，则该算法的时间复杂度为(　　)。

A. O(logn)　　B. O(nlogn)　　C. O(n)　　D. $O(n^2)$

【答案】D

【分析】本题考查复杂度的计算方法。$T(n)=T(n-1)+n=T(n-2)+(n-1)+n=\cdots=T(0)+1+2+\cdots+n=1+\frac{(n+1)*n}{2}$，所以是 $O(n^2)$。

5.【NOIP2016】假设某算法的计算时间表示为递推关系式

$$T(n)=2T\left(\frac{n}{4}\right)+\sqrt{n}$$

$$T(1)=1$$

则算法的时间复杂度为(　　)。

A. O(n)　　B. $O(\sqrt{n})$　　C. $O(\sqrt{n}\log n)$　　D. $O(n^2)$

【答案】C

【分析】可以根据递推关系式一路推导得出结果：

$$T(n)=2T\left(\frac{n}{4}\right)+\sqrt{n}=2\left(2T\left(\frac{n}{16}\right)+\sqrt{\frac{n}{4}}\right)+\sqrt{n}=\cdots=2^kT\left(\frac{n}{4^k}\right)+k\sqrt{n}=\cdots$$

$$=2^{\log_4 n}T(1)+\sqrt{n}\log_4 n=\sqrt{n}*(1+\log_4 n)=O(\sqrt{n}\log n)$$

6.【NOIP2017】若某算法的计算时间表示为递推关系式：

$$T(N)=2T(N/2)+N\log N$$

$$T(1)=1$$

则该算法的时间复杂度为(　　)。

A. O(N)　　B. O(NlogN)　　C. $O(N\log^2 N)$　　D. $O(N^2)$

【答案】C

【分析】考查主定理。可以根据递推关系式一路推导得出结果：

$$T(N)=2T(\frac{N}{2})+NlogN=2(2T(\frac{N}{4})+\frac{N}{2}log\frac{N}{2})+NlogN$$

$$=4T(\frac{N}{4})+N(log\frac{N}{2}+logN)=\cdots=2^kT(\frac{N}{2^k})+N(log\frac{N}{2^{k-1}}+\cdots+log\frac{N}{2}+logN)$$

$$=2^kT(\frac{N}{2^k})+Nlog\frac{N^k}{2^{0+1+2+\cdots+k-1}}=2^kT(\frac{N}{2^k})+N(klogN-k*(k-1)/2)$$

$$=\cdots=2^{logN}T(1)+N(logN*logN-logN*(logN-1)/2)$$

$$=N+N(logN*logN-logN*(logN-1)/2)$$

$$=O(Nlog^2N)$$

7.【NOIP2012】如果对于所有规模为 n 的输入，一个算法均恰好进行(　　)次运算，我们可以说该算法的时间复杂度为 $O(2^n)$。

A. 2^{n+1}　　B. 3^n　　C. $n*2^n$　　D. 2^{2n}

【答案】A

【分析】计算机科学中，算法的时间复杂度是一个函数，它定量描述了该算法的运行时间。时间复杂度常用大 O 符号表述，忽略这个函数的低阶项和首项系数，$2^{n+1}=2*2^n$。

第2节 C++语言基础

由于篇幅所限，本书在此只介绍C++语言的大致语法和使用，以帮助读者快速读懂程序。更详细的使用请参考《信息学奥赛一本通》这本书的前面几章或者自行在网上查找资源。

一、程序的基本构成

```
#include<iostream>
using namespace std;
int main(){                    //main()是程序开始执行的地方
    cout<<"Hello world";       //输出 Hello world
    return 0;
}
```

(1)头文件<iostream>包含了程序中如输入输出即cin、cout的定义。

(2)using namespace std;告诉编译器使用std命名空间。命名空间是为了作为附加信息来区分不同库中相同名称的函数、类、变量等。本质上，命名空间就是定义了一个范围。

(3)下一行int main()是主函数，程序从这里开始执行。

(4)//是一个单行注释，用于解释说明。单行注释以//开头，在行末结束。

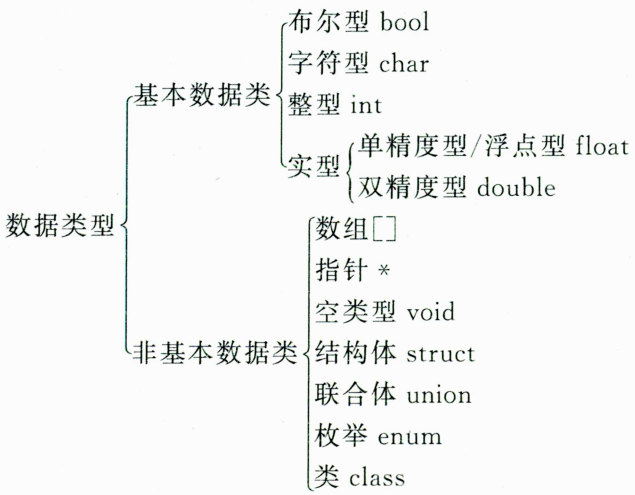
(5)cout<<"Hello World";用于输出，在屏幕上显示消息"Hello World"。

(6)return 0;终止main()函数，并向调用进程返回值0。

(7)分号是语句结束符。也就是说每个语句必须以分号结束。

(8){}称为块，其中的所有语句是一个语句块。

二、变量及数据类型




```
#include<iostream>
using namespace std;
int main(){
    float x;
    int i;
```

```
    x=3.7;
    i=(int)x;//强制类型转换
    cout<<"x="<<x<<",i="<<i<<endl;//输出 x=3.7,i=3
    return 0;
}
#include<iostream>
using namespace std;
struct stuInfo {
    int ID;
    char name[20];
    char sex;
    int age;
};
int main(){
    stuInfo x={13,"xiaoma",'m',35};
    stuInfo * p=&x;          //指针 p 指向 x
    cout<<x.ID<<endl;
    cout<<(* p).ID<<endl; //指针访问方式一
    cout<<p->ID<<endl;    //指针访问方式二
    return 0;
}
```

三、类

C++在C语言的基础上增加了面向对象编程。类是C++的核心特性，通常被称为用户定义的类型，其包含了数据表示和用于处理数据的方法。类中的数据和方法都称为类的成员。

```
class Box{//方法一
    public://public 是类型修饰符，具体意义见“类访问修饰符”
        double length; //长度
        double width;  //宽度      } 类的数据成员，也就是类中的变量
        double height; //高度
        double getVolume(){
            return length *width*height;  } 类的成员函数，也就是类中的函数
        }
};
```

四、程序的基本结构

一般一个程序可以用顺序、选择和循环三种基本结构组合而成。

1.顺序结构

顺序结构是最简单、最常用的结构，语句与语句之间按从上到下的顺序执行即可。

2.选择结构

选择结构是先根据条件做出判断，再决定执行哪一种操作的算法结构，必须包含判断框。当条件 P 成立(或称为真)时执行 A，否则执行 B，不可能两者同时执行，但 A 或 B 两个框中可以有一个是空的，即不执行任何操作。常用的是 if、if…else、switch。

```
#include<iostream>
using namespace std;
int main(){
    int a=20;
    if (a==10){//如果 if 条件为真,则输出下面的语句
        cout<<"a 的值是 10"<<endl;
    }else if (a==20){//如果 else if 条件为真,则输出下面的语句
        cout<<"a 的值是 20"<<endl;
    }else if (a==30){
        cout<<"a 的值是 30"<<endl;
    }else {
        cout<<"没有匹配的值"<<endl;
    }
    cout<<"a 的准确值是"<<a<<endl;
    return 0;
}
```

3.循环结构

在一些算法中，经常会出现从某处开始，按照一定条件，反复执行某一处理步骤的情况，这就是循环结构。常用的有 while、for、do… while

```
#include<iostream>
using namespace std;
int main(){
    for (int a=1;a<10;a=a+1){
        cout<< "a 的值:"<<a<<endl;
    }
    int b=1;
    while (b<10){
        cout<<"b 的值:"<<b<<endl;
        b++;
    }
    return 0;
}
```

五、函数

函数是用来实现某些特定功能而封装成的一个模块。在创建函数时，必须编写其定义。所有函数定义包括以下组成部分：

```
返回类型    函数名(形参列表){
              函数主体
            }
```

1.函数名：每个函数都必须有一个名称。通常，适用于变量名称的规则同样也适用于函数名称。

2.形参列表：保存传递给函数的值的变量列表。如果没有值传递给函数，则其形参列表为空。

3.函数主体：处理函数正在执行的任务的一组语句，这些语句包含在一组大括号中。

4.返回类型：函数可以将值发送回调用它的程序模块。返回类型是要发送回的值的数据类型。

例如：编写求半径为 r 的圆的面积的函数 area。

```
//这是一个求值的函数，函数中设置一个形参 r，用于接收圆的半径。
//将参数类型和返回值类型定义为 double。
double area(double r) //返回值类型为 double
{
    double s;
    s=3.14*r*r;
    return s;//返回计算结果
}
```

在执行函数时，函数参数，局部变量（包括 const 局部变量），函数调用后返回的地址都在内存的栈区上创建，函数执行结束时这些存储单元自动被释放。

六、递归函数

递归，就是在运行的过程中调用自己。

构成递归需具备的条件：

1.子问题须与原始问题为同样的事，且更为简单；

2.不能无限制地调用本身，必须有个出口，化简为非递归状况处理。

例如编写代码求 1＋2＋…＋n 的值，其中 n≤20。

```
#include<iostream>
using namespace std;
//求解 1+2…+n
int sum(int x){
    if (x==1)return 1;//出口，这样就不能无限制地调用本身了
    return sum(x-1)+x;//子问题 sum(x-1)与原始问题 sum(x)同样的事，且更为简单
}
int main(){
    int n;
    cin>>n;
    cout<<sum(n);
    return 0;
}
```

课堂练习

1.【NOIP2014】若有如下程序段，其中 s、a、b、c 均已定义为整型变量，且 a、c 均已赋值，c>0。

```
s=a;
for (b=1;b<=c;b++)
    s+=1;
```

则与上述程序段功能等价的赋值语句是(　　)。

A. s=a+b　　B. s=a+c　　C. s=s+c　　D. s=b+c

【答案】B

【分析】s 初值是 a，循环体执行 c 次，每次加 1，因此 s=a+c。

2.【NOIP2014】要求按以下程序计算 s=1+1/2+1/3+…+1/10。

```
#include<iostream>
using namespace std;
int main(){
    int n;
    float s;
    s=1.0;
    for (n=10;n>1;n--)
        s=s+1/n;
    cout<<s<<endl;
    return 0;
}
```

程序运行后输出结果错误，导致错误结果的程序行是(　　)。

A. s=1.0;　　B. for (n=10;n>1;n--)

C. s=s+1/n;　　D. cout<<s<<endl;

【答案】C

【分析】将不同类型进行算术运算时，会进行类型转换。

(1)级别从高到底：double>float>long long>int>char>bool；

(2)级别从高到底：无符号数>有符号数；

由于 n 是整数，因此表达式中的 1/n 会处理为整数除法，而这里我们需要的结果应该为实数。要使结果正确，需要使用实数除法，可以改为 1.0/n 或 1/(float)n 等。

3.【NOIP2014】有以下程序：

```
#include<iostream>
using namespace std;
int main(){
    int s,a,n;
    s=0;
    a=1;
```

```
    cin>>n;
    do {
        s+=1;
        a-=2;
    } while (a!=n);
    cout<<s<<endl;
    return 0;
}
```

若要使程序的输出值为 2，则应该从键盘给 n 输入的值是(　　)。

A. −1　　B. −3　　C. −5　　D. 0

【答案】B

【分析】要使得最终 s=2，需要执行两次 do…while 循环体，n 的值应该为−3。

4.【NOIP2016】有以下程序：

```
#include<iostream>
using namespace std;
int main(){
    int k=4,n=0;
    while (n<k){
        n++;
        if (n%3!=0)
            continue;
        k--;
    }
    cout<<k<<","<<n<<endl;
    return 0;
}
```

程序运行后的输出结果是(　　)。

A. 2,2　　B. 2,3　　C. 3,2　　D. 3,3

【答案】D

【分析】continue 语句的作用是跳过当前循环体未执行的语句(即本题的 k--;)，而继续执行下一次循环。当 n<3 时，if 条件始终成立，k 不会发生任何变化，因此只有在 n=3 时才会执行 k--使得 k=3，并终止循环。

5.【NOIP2018】为了统计一个非负整数的二进制形式中 1 的个数，代码如下：

```
int CountBit (int x){
  int ret=0;
  while (x){
      ret++;
      _________;
  }
```

```
    return ret;
}
```

则空格内要填入的语句是(　　)。

A. x>>=1　　B. x&=x-1　　C. x|=x>>1　　D. x<<=1

【答案】B

【分析】考察位运算知识。观察程序可知 ret 存的是 x 二进制 1 的个数，带入一些不同的 x 值到程序中，会发现 x&=x−1 的作用是使 x 最低位的 1 置零，所以符合要求。

6.【NOIP2013】下列程序中，正确计算 1,2,…,100 这 100 个自然数之和 sum（初始值为 0）的是(　　)。

A.	`i=1;`
	`do{`
	`  sum+=i;`
	`  i++;`
	`} while (i<=100);`
		B.	`i=1;`
			`do{`
			`  sum+=i;`
			`  i++;`
			`} while (i>100);`

C.	`i=1;`
	`while (i<100){`
	`  sum+=i;`
	`  i++;`
	`}`
		D.	`i=1;`
			`while (i>=100){`
			`  sum+=i;`
			`  i++;`
			`}`

【答案】A

【分析】可以正确的计算 1,2,3,…,100 之和，而 B 和 D 循环执行条件是 i>100，显然在 i 初始等于 1 的情况下，循环不会被执行。C 只能求 1 至 99 的和。

7.【NOIP2014】若有变量 int a，float x,y，且 a=7,x=2.5,y=4.7，则表达式 x+a%3*(int)(x+y)%2/4 的值大约是(　　)。

A. 2.500000　　B. 2.750000　　C. 3.500000　　D. 0.000000

【答案】A

【分析】考察运算符号的优先级和运算顺序。运算级优先级为()大于%，*，/大于+，−，同一级别的从左往右计算，所以(int)(x+y)=7，a%3*(int)(x+y)%2/4=1*7%2/4=0，所以最后答案为 x，即 2.500000。

8.【NOIP2014】设变量 x 为 float 型且已赋值，则以下语句中能将 x 中的数值保留到小数点后两位，并将第三位四舍五入的是(　　)。

A. x=(x*100)+0.5/100.0;　　B. x=(x*100+0.5)/100.0;

C. x=(int)(x*100+0.5)/100.0;　　D. x=(x/100+0.5)*100.0;

【答案】C

【分析】通过将 float 型强制转换为 int 型可实现截取整数的操作，而在截取之前加 0.5，就实现了四舍五入的功能。本题也可以使用排除法，通过代入具体的数值进行验证。假设 x=1.116，A,B,C,D 的计算结果分别为 111.605,1.121,1.12,51.116。

9.【NOIP2014】以下程序段实现了找第二小元素的算法。输入是 n 个不等的数构成数组 S，输出 S 中第二小的数 SecondMin。在最坏情况下，该算法需要做(　　)次比较。

```
if (S[1]<S[2]){
    FirstMin=S[1];
    SecondMin=S[2];
}else {
    FirstMin=S[2];
    SecondMin=S[1];
}
for (i=3;i<=n;i++)
  if (S [i]<SecondMin)
    if (S [i]<FirstMin){
        SecondMin=FirstMin;
        FirstMin=S [i];
    }else {
        SecondMin=S [i];
}
```

A. 2n　　B. n－1　　C. 2n－3　　D. 2n－2

【答案】C

【分析 1】前两个元素比较了一次，之后每个元素，在最坏情况下都要和两个元素比较，因此共比较了 1＋2(n－2)＝2n－3 次。

【分析 2】代入法，n＝2 时，比较 1 次，n＝3 时，比较 3 次。

10.【NOIP2008】递归过程或函数调用时，处理参数和返回地址，通常使用一种称为(　　)的数据结构。

A. 队列　　B. 多维数组　　C. 线性表　　D. 栈

【答案】D

【分析】递归过程或者函数调用时需要把处理参数和返回地址存放在当前栈顶，当执行函数结束后，需要将当前栈顶存放的信息重新取出才能回到调用位置恢复场景。

11.【NOIP2012】在程序运行过程中，如果递归调用的层数过多，会因为(　　)引发错误。

A. 系统分配的栈空间溢出　　B. 系统分配的堆空间溢出

C. 系统分配的队列空间溢出　　D. 系统分配的链表空间溢出

【答案】A

【分析】递归调用的是内存栈区。C++可编程内存在基本上分为：静态区、堆区、栈区。静态(全局)存储区——static：主要存放静态数据、全局数据和常量，也是程序结束后，由操作系统释放。堆区——heap：亦称动态内存分配。程序在运行的时候用 malloc 或 new 申请任意大小的内存，程序员自己负责在适当的时候用 free 或 delete 释放内存。栈区——stack：在执行函数时，函数参数，局部变量(包括 const 局部常量)，函数调用后返回的地址都在栈上创建，每次递归都会生成一个栈内存，层数过多，会导致栈空间溢出。

不定项选择题

1.【NOIP2013】下列程序中，正确计算 1,2,…,100 这 100 个自然数之和 sum（初始值为 0）的是(　　)。

A.	`for (i=1;i<=100;i++)`
	`　　sum+=i;`
		B.	`i=1;`
			`while (i>100){`
			`　　sum+=i;`
			`　　i++;`
			`}`

C.	`i=1;`
	`do{`
	`　　sum+=i;`
	`　　i++;`
	`} while (i<=100);`
		D.	`i=1;`
			`do{`
			`　　sum+=i;`
			`　　i++;`
			`} while (i>100);`

【答案】AC

【分析】A 和 C 可以正确的计算 1 到 100 的和，而 B 和 D 循环执行的条件是 i>100，显然在 i 初始等于 1 的情况下，循环不会被执行。

第3节　排序算法

一、内排序和外排序的定义

由于待排序的记录数量不同，使得排序过程中涉及的存储器不同，可将排序方法分为两大类：内排序与外排序。

1.内排序：待排序记录存放在计算机内存中进行的排序过程。插入排序、快速排序、选择排序、归并排序、基数排序等目前竞赛研究的算法基本上都是内排序方法。

2.外排序：待排序记录的数量很大，以致于内存不能一次容纳全部记录，所以在排序过程中需要对外存进行访问的排序过程。

二、衡量效率的方法

1.内排序：比较次数，也就是时间复杂度。

2.外排序：IO次数，也就是读写外存的次数。

三、排序稳定性

排序前后相同元素的相对位置不变，则称排序算法是稳定的，否则排序算法是不稳定的。如原序列 ri=rj 且 ri 位于 rj 之前，排序后 ri 仍在 rj 之前，则称该排序是稳定的。

如：数据 57,491,26,3,492,5 进行升序排序。

稳定算法排序后为：3,5,26,491,492,57。

不稳定算法排序后可能为：3,5,26,492,491,57。

四、常用排序算法复杂度

排序法	最坏时间	平均时间复杂度	稳定性	空间复杂度
插入排序	$O(n^2)$	$O(n^2)$	稳定	O(1)
选择排序	$O(n^2)$	$O(n^2)$	不稳定	O(1)
冒泡排序	$O(n^2)$	$O(n^2)$	稳定	O(1)
希尔排序	$O(n^s)$，$1<s<2$，取决于增量序列	$O(n\log_2 n)$	不稳定	O(1)
快速排序	$O(n^2)$	$O(n\log_2 n)$	不稳定	$O(\log_2 n)$
堆排序	O(n logn)	$O(n\log_2 n)$	不稳定	O(1)
归并排序	O(n logn)	$O(n\log_2 n)$	稳定	O(n)



课堂练习

1.【NOIP2011】体育课的铃声响了，同学们都陆续地奔向操场，按老师的要求从高到矮站成一排。每个同学按顺序来到操场时，都从排尾走向排头，找到第一个比自己高的同学，并站在他的后面。这种站队的方法类似于(　　)算法。

A. 快速排序　　B. 插入排序　　C. 冒泡排序　　D. 归并排序

【答案】B

【分析】插入排序基本思想是：把待排序的纪录按其关键码值的大小逐个插入到一个已经排好序的有序序列中，直到所有的记录插入完为止，得到一个新的有序序列。

2.【NOIP2018】以下排序算法中，不需要进行关键字比较操作的算法是(　　)。

A. 基数排序　　B. 冒泡排序　　C. 堆排序　　D. 直接插入排序

【答案】A

【分析】冒泡排序、堆排序和插入排序都是基于两两元素关键字比较的排序。基数排序是直接将元素通过某些规则放到数组的相应位置，不是基于两两元素关键字比较的排序。

3.【NOIP2009】排序算法是稳定的意思是关键码相同的记录排序前后相对位置不发生改变，下列排序算法不稳定的是(　　)。

A. 冒泡排序　　B. 插入排序　　C. 归并排序　　D. 快速排序

【答案】D

【分析】快速排序会使数据左右调换，让值相同的两个元素的相对位置发生变化，而这个变化是不稳定的原因。

4.【NOIP2009】快速排序最坏情况下的算法复杂度为(　　)。

A. O(logn)　　B. O(n)　　C. O(nlogn)　　D. $O(n^2)$

【答案】D

【分析】假设当快速排序时选择的轴值总数是最靠边的数，且当输入数据有序时，快速排序就退化成了冒泡排序。

5.【NOIP2009】快速排序平均情况和最坏情况下的算法时间复杂度分别为(　　)。

A. 平均情况 O(nlogn)，最坏情况 $O(n^2)$　　B. 平均情况 O(n)，最坏情况 $O(n^2)$

C. 平均情况 O(n)，最坏情况 O(nlogn)　　D. 平均情况 O(logn)，最坏情况 $O(n^2)$

【答案】A

【分析】快速排序平均情况时间复杂度是 $O(n\log_2 n)$，最坏情况是 $O(n^2)$。

6.【NOIP2012】如果不在快速排序中引入随机化，有可能导致的后果是(　　)。

A. 数组访问越界　　B. 陷入死循环

C. 排序结果错误　　D. 排序时间退化为平方级

【答案】D

【分析】快速排序的期望时间复杂度是 O(nlogn)，最坏情况(例如排一个已经排好序的序列)是 $O(n^2)$。

7.【NOIP2010】基于比较的排序时间复杂度的下限是(　　)，其中 n 是待排的元素个数。

A. O(n)　　B. O(nlogn)　　C. O(logn)　　D. $O(n^2)$

【答案】B

【分析】快速排序、归并排序都是基于比较的排序，时间复杂度不会低于 nlogn。而桶排序、基数排序都不是基于比较的排序。

8.【NOIP2013】(　　)的平均时间复杂度为 O(nlogn)，其中 n 是待排序的元素个数。

A. 快速排序　　B. 插入排序　　C. 冒泡排序　　D. 基数排序

【答案】A

【分析】快速排序平均时间复杂度均为 O(nlogn)，而插入排序和冒泡排序的平均时间复杂度为 $O(n^2)$。基数排序的时间复杂度，还取决于待排序的数字的位数。

9.【NOIP2014】以下时间复杂度不是 $O(n^2)$ 的排序方法是(　　)。

A. 插入排序　　B. 归并排序　　C. 冒泡排序　　D. 选择排序

【答案】B

【分析】归并排序的时间复杂度是 O(nlogn)。

10.【NOIP2017】对于给定的序列 $\{a_k\}$，我们把(i,j)称为逆序对当且仅当 $i<j$ 且 $a_i>a_j$。那么序列 1,7,2,3,5,4 的逆序对数为(　　)个。

A. 4　　B. 5　　C. 6　　D. 7

【答案】B

【分析】按照逆序对的标准，依次清点一下即可，每个数字查看右侧有几个比自己小的。逆序对为(7,2)，(7,3)，(7,5)，(7,4)，(5,4)。

11.【NOIP2012】使用冒泡排序对序列进行升序排序，每执行一次交换操作将会减少 1 个逆序对，因此序列 5,4,3,2,1 需要执行(　　)次交换操作，才能完成冒泡排序。

A. 0　　B. 5　　C. 10　　D. 15

【答案】C

【分析】在数组中，如果 $i<j$ 但 $a[i]>a[j]$，我们称其为一对逆序对。根据题目意思每交换一次会减少一个逆序对，所以交换次数等于逆序对数等于 10，逆序对分别为(5,4)、(5,3)、(5,2)、(5,1)、(4,3)、(4,2)、(4,1)、(3,2)、(3,1)、(2,1)。

12.【NOIP2017】设 A 和 B 是两个长为 n 的有序数组，现在需要将 A 和 B 合并成一个排好序的数组，任何以元素比较作为基本运算的归并算法在最坏情况下至少要做(　　)次比较。

A. n^2　　B. nlogn　　C. 2n　　D. 2n−1

【答案】D

【分析 1】考察归并排序算法中归并的过程。对于两个长度为 n 的有序数组，首先我们考虑合并有序数组的方式肯定是比较序列的头，选一个出来放到新的序列中，当某个序列为空的时候，就另外一个序列的数组依次放到最后。最坏情况下显然是两个数组依次弹出数字，直到最后一个数组仅剩下一个数字。这种情况下，除去最后一个数字以外每一个数字都进行了一次比较，所以总共需要比较 2n−1 次。

【分析 2】归并排序当有一方为空的时候，就可以不用再进行比较，直接将另一方剩余元素接在数组后面即可。否则每进行一次比较，均有一个元素可以接到数组后面，故总比较次数为总元素个数减去一方为空时另一方的剩余元素个数。所以，最少的剩余元素是 1，最多的比较次数为 2n−1。

不定项选择题

1.【NOIP2009】排序算法是稳定的意思是关键码相同的记录排序前后相对位置不发生改变，下列排序算法稳定的是(　　)。

A. 插入排序　　B. 基数排序　　C. 归并排序　　D. 冒泡排序

【答案】ABCD

【分析】希尔排序、快速排序、选择排序、堆排序都是不稳定的。

2.【NOIP2017】下列算法中，(　　)是稳定的排序算法。

A. 快速排序　　B. 堆排序　　C. 希尔排序　　D. 插入排序

【答案】D

【分析】稳定的排序指的是对于原序列两个相等的元素，在排完序之后前后位置关系仍然保持不变。只有插入排序是相邻元素的比较，碰到相等不需要交换，所以可以保持相等元素的顺序不改变，其他三个排序都不是相邻元素的比较，因此并不是稳定的排序。

3.【NOIP2010】原地排序是指在排序过程中(除了存储待排序元素以外的)辅助空间的大小与数据规模无关的排序算法。以下属于原地排序的有(　　)。

A. 冒泡排序　　B. 插入排序　　C. 基数排序　　D. 选择排序

【答案】ABD

【分析】只有基数排序需要额外的“桶”记录临时排序结果，其余选项均为原地排序。

4.【NOIP2016】下列算法中运用分治思想的有(　　)。

A. 快速排序　　B. 归并排序　　C. 冒泡排序　　D. 计数排序

【答案】AB

【分析】快速排序和归并排序都是分治算法的经典案例。

5.【NOIP2013】(　　)的平均时间复杂度为 O(nlogn)，其中 n 是待排序的元素个数。

A. 快速排序　　B. 插入排序　　C. 冒泡排序　　D. 归并排序

【答案】AD

【分析】快速排序和归并排序的平均时间复杂度均为 O(nlogn)，而插入排序和冒泡排序的平均时间复杂度为 $O(n^2)$。

6.【NOIP2017】以下排序算法在最坏情况下时间复杂度最优的有(　　)。

A. 冒泡排序　　B. 快速排序　　C. 归并排序　　D. 堆排序

【答案】CD

【分析】冒泡排序和快速排序最坏情况下的时间复杂度为 $O(n^2)$，而归并排序和堆排序是稳定的 O(nlogn)。

7.【NOIP2011】对于序列“7、5、1、9、3、6、8、4”，在不改变顺序的情况下，去掉(　　)会使逆序对的个数减少 3。

A. 7　　B. 5　　C. 3　　D. 6

【答案】CD

【分析】对于一个包含 N 个非负整数的数组 A[1…n]，如果有 i<j，且 A[i]>A[j]，则称(A[i]，A[j])为数组 A 中的一个逆序对。此序列的逆序列有 14 对：(7,5)，(7,1)，(7,3)，(7,6)，(7,4)，(5,1)，(5,3)，(5,4)，(9,3)，(9,6)，(9,8)，(9,4)，(6,4)，(8,4)。如果要使其逆序列的个数减少 3 对，则将其中出现了 3 次的数字删除即可。7 出现了 5 次，5 出现了 4 次，1 出现了 2 次，9 出现了 4 次，3 出现了 3 次，6 出现了 3 次，8 出现了 2 次，4 出现了 5 次。故数字 3 和 6 是满足题意的。

第4节　基础算法

由于篇幅所限，这里主要对基础算法做一些简单的回顾总结，更详细的内容参考《信息学奥赛一本通》相应章节。

一、高精度计算

高精度数值处理是采用模拟算法对位数达到上百位甚至更多位数的数字进行各种运算，包括加法、减法、乘法、除法等基础运算。核心思想是对参与运算的两个数分别用两个数组进行存储。常见问题有“汉诺塔的步数”“国王的米粒”等。

二、穷举算法

穷举法即枚举法，是从可能的解的集合中一一枚举各元素，用题目给定的检验条件判定哪些是无用的，哪些是有用的。能使命题成立即为其解。常见问题有“百钱买百鸡”“素数判断”等。

三、递推算法

递推法是一种重要的数学方法，在数学的各个领域中都有广泛运用，也是计算机用于数值计算的一个重要算法。递推算法的首要问题是得到相邻数据项间的关系（即递推关系）。递推算法避开了求通项公式的麻烦，把一个复杂问题的求解，分解成了连续的若干步简单运算。一般说来，可以将递推算法看成是一种特殊的迭代算法。常见问题有“斐波那契数列”“过河卒”等。

四、递归算法

递归程序设计是C++语言程序设计中一种重要的方法，它使许多复杂的问题变得简单以及容易解决。递归特点是函数或过程调用自己本身，其中直接调用自己称为直接递归，而将A调用B，B调用A的递归叫做间接递归。常见问题有“汉诺塔问题”“Ackermann函数”等。

五、搜索与回溯算法

搜索与回溯是计算机解题中常用的算法，很多问题无法根据某种确定的计算法则来求解，可以利用搜索与回溯的技术求解。回溯是搜索算法中的一种控制策略。它的基本思想是：为了求得问题的解，先选择某一种可能情况向前探索，在探索过程中，一旦发现原来的选择是错误的，就退回一步重新选择，继续向前探索，如此反复进行，直至得到解或证明无解。常见问题有“八皇后问题”“骑士游历问题”等。

六、贪心算法

贪心算法是指对问题求解时，总是做出在当前看来是最好的选择。也就是说，不从整体最优上加以考虑，它所做出的仅是在某种意义上的局部最优解。

贪心算法没有固定的算法框架，算法设计的关键是贪心策略的选择。必须注意的是，贪心算法不是所有问题都能得到整体最优解，选择的贪心策略必须具备无后效性，即某个状态以后的过程不会影响以前的状态，只与当前状态有关。所以对所采用的贪心策略一定要仔细分析其是否满足无后效性。常见问题有“删数问题”“排队打水问题”等。

七、分治算法

分治就是指的分而治之，即将较大规模的问题分解成几个较小规模的子问题，通过对较小规模子问题的求解达到对整个问题的求解。将问题分解成两个较小子问题求解时的分治方法称之为二分法。常见问题有“归并排序”“比赛日程安排”等。

八、广度优先搜索

广度优先算法的核心思想是：从初始节点开始，应用算符生成第一层节点，检查目标节点是否在这些后继节点中，若没有，再用产生式规则将所有第一层的节点逐一扩展，得到第二层节点，并逐一检查第二层节点中是否包含目标节点。若没有，再用算符逐一扩展第二层的所有节点……，如此依次扩展，检查下去，直到发现目标节点为止。这种搜索的次序体现沿层次向横向扩展的趋势，称之为广度优先搜索。常见问题有“分油问题”“八数码问题”等。

九、动态规划

动态规划程序设计是解决多阶段决策过程最优化问题的一种途径。动态规划的设计方法对不同的问题，有各具特色的解题方法，不存在一种万能的动态规划算法，可以解决各类最优化问题。需要满足“最优子结构”和“无后效性”的两项基本条件。动态规划解决问题的步骤：

1.确定状态：状态是一个数学形式；

2.确定状态转移方程和边界条件：递归或递推；

3.程序实现。

动态规划大致可以分为以下几类：线性、区间、背包型、树型、数位、状态压缩等。常见问题有“最长不下降序列”“最长公共子序列”等。



课堂练习

1.【NOIP2016】周末小明和爸爸妈妈三个人一起想动手做三道菜。小明负责洗菜、爸爸负责切菜、妈妈负责炒菜。假设做每道菜的顺序都是：先洗菜 10 分钟，然后切菜 10 分钟，最后炒菜 10 分钟。那么做一道菜需要 30 分钟。注意：两道不同菜的相同步骤不可以同时进行。例如第一道菜和第二道的菜不能同时洗，也不能同时切，那么做完三道菜的最短时间需要(　　)分钟。

A. 90　　B. 60　　C. 50　　D. 40

【答案】C

【分析】本题考查基本常识，三个人分别负责洗菜切菜炒菜，每个人完成自己的工序之后交给下一个人，那么第一道菜做完需要 30 分钟，但后面每 10 分钟都能完成一道菜，总共需要 50 分钟。本题的目的在于体现计算机 CPU 中流水线优化的意义。

2.【NOIP2017】2017 年 10 月 1 日是星期日，1999 年 10 月 1 日是(　　)。

A. 星期三　　B. 星期日　　C. 星期五　　D. 星期二

【答案】C

【分析】考查闰年与同余的相关知识。直接计算两个日期之间相差多少天即可，平年有 365 天，365%7=1，如果 2017(平年)年 10 月 1 日是星期日，那么 2016 年 10 月 1 日是星期六，所

以得出第一个结论：两个相差一个平年的日期的星期相差1；闰年有366天，366%7=2，如果2016年(闰年)10月1日是星期六，那么2015年10月1日是星期四，所以得出第二个结论：两个相差一个闰年的日期的星期相差2；1999－2017年相差18年，其中有5个闰年(2000，2004，2008，2012，2016)，所以相差的天数为(0－18－5)%7＝－2，即把星期天往前推2天，答案是星期五。

3.【NOIP2018】下面的故事与(　　)算法有着异曲同工之妙。

从前有座山，山里有座庙，庙里有个老和尚在给小和尚讲故事；从前有座山，山里有座庙，庙里有个老和尚在给小和尚讲故事；从前有座山，山里有座庙，庙里有个老和尚给小和尚讲故事……

A. 枚举　　B. 递归　　C. 贪心　　D. 分治

【答案】B

【分析】考察算法基础知识。显然“从前有座山……”这个故事描述的是递归算法。和尚讲的故事里继续套用和尚讲故事，正如函数不断调用本身，形成递归。

4.【NOIP2011】(　　)是一种选优搜索法，按选优条件向前搜索，以达到目标。当探索到某一步时，发现原先选择并不优或达不到目标，就退回一步重新选择。

A. 回溯法　　B. 枚举法　　C. 动态规划　　D. 贪心法

【答案】A

【分析】搜索与回溯：因为是选优，所以排除枚举法，又因为会退回一步重新选择，而动态规划和贪心都是不会退的，所以是回溯法。

5.【NOIP2013】将(2，6，10，17)分别存储到某个地址区间为0～10的哈希表中，如果哈希函数h(x)＝(　　)，将不会产生冲突，其中a mod b表示a除以b的余数。

A. x mod 11　　B. x^2 mod 11

C. 2x mod 11　　D. $\lfloor\sqrt{x}\rfloor$ mod 11，其中$\lfloor\sqrt{x}\rfloor$表示$\sqrt{x}$下取整

【答案】D

【分析】哈希查找算法，将(2，6，10，17)代入四个选项，当我们选取D做哈希函数时，(2，6，10，17)散列后的值分别为(1，2，3，4)，不会产生冲突。A、B、C选项，6和17都产生冲突。

6.【NOIP2012】(　　)就是把一个复杂的问题分成两个或者更多的相同或相似的子问题，再把子问题分成更小的子问题……直到最后的子问题可以简单的直接求解。而原问题的解就是子问题解的并。

A. 动态规划　　B. 贪心　　C. 分治　　D. 搜索

【答案】C

【分析】题目大意就是分治法的定义。

7.【NOIP2016】给定含有n个不同数的数组L＝<x1，x2，…，xn>。如果L中存在xi(1<i<n)使得x1<x2<…<xi－1<xi>xi＋1>…>xn，则称L是单峰的，并称xi是L的“峰顶”。现在已知L是单峰的，请把a－c三行代码补全到算法中使得算法正确找到L的峰顶。

```
a. Search(k+1,n)    b. Search(1,k-1)    c. return L[k]
Search(1,n)
1.k←⌊n/2⌋
2.if L[k]> L[k-1] and L[k]> L[k+1]
3.then _____
```

4.else if L[k]> L[k-1] and L[k] < L[k+1]

5.then _____

6.else _____

正确的填空顺序是(　　)。

A. c,a,b　　B. c,b,a　　C. a,b,c　　D. b,a,c

【答案】A

【分析】考查对二分查找的理解。

如果第 k 个数大于第 k－1 个数和第 k＋1 个数，那么 k 就是峰顶；

如果第 k 个数大于第 k－1 个数，但小于第 k＋1 个数，那么峰顶在 k 的右侧，需要在 k＋1 到 n 的范围内去寻找峰顶；否则，应该去 1 到 k－1 范围内去寻找峰顶。

8.【NOIP2017】在 n(n≥3)枚硬币中有一枚质量不合格的硬币(质量过轻或质量过重)，如果只有一架天平可以用来称重且称重的硬币数没有限制，下面是找出这枚不合格的硬币的算法。请把 a－c 三行代码补全到算法中。

a.　A←X∪Y

b.　A←Z

c.　n←|A|

算法 Coin(A,n)

(1)k←⌊n/3⌋

(2)将 A 中硬币分成 X,Y,Z 三个集合，使得|X|= |Y|= k,|Z|= n- 2k

(3)　if W(X)≠W(Y)//W(X),W(Y)分别为 X 或 Y 的重量

(4)　then _____

(5)　else _____

(6)　_____

(7)if n> 2 then go to 1

(8)if n= 2 then 任取 A 中 1 枚硬币与拿走硬币比较，若不等，则它不合格；若相等，则 A 中剩下的硬币不合格.

(9)if n= 1 then A 中硬币不合格

正确的填空顺序是(　　)。

A. b,c,a　　B. c,b,a　　C. c,a,b　　D. a,b,c

【答案】D

【分析】分治：首先要理解这几个符号，a 操作是将 X 和 Y 的并集赋值给 A，b 操作是将 Z 赋值给 A，c 操作是将 A 集合的大小赋值给 n。

那么这里其实就是一个三分的操作，第 3 行判断 X，Y 质量是否相等，如果不相等，显然不合格的硬币出现在 X 或 Y 中，那么这里要做的操作就是继续搜索 X 和 Y 的并集，所以这里填 a 操作。

接下来第 5 行，若 X，Y 质量相等，则不合格的硬币在 Z 中，所以这里填 b 操作。

第 6 行将新集合的大小更新到 n，选择 c 操作。

9.【NOIP2011】应用快速排序的分治思想，可以实现一个求第 K 大数的程序。假定不考虑极端的最坏情况，理论上可以实现的最低的算法时间复杂度为(　　)。

A. $O(n^2)$　　B. O(nlogn)　　C. O(n)　　D. O(1)

【答案】C

【分析】分治：应用快排的思想，随机选一个轴数 k，把数列分为左右两端，一边都小于 k，一

边都大于等于 k，然后要么在左边要么在右边。极端最好的情况，随机选的轴数 k，就是需要的答案，比较的次数是 n－1 次；极端最坏的情况是这样的，随机选的轴数 k 是序列中的极值，不能把数列分开，每次问题的规模只能减少 1；平均的情况是，每次查找范围期望缩小一半。所以复杂度就是 $n+\frac{n}{2}+\frac{n}{4}+\frac{n}{8}+\cdots<2n=O(n)$。综上所述，这题答案应该选 C。

10.【NOIP2014】设有 100 个数据元素，采用折半搜索时，最大比较次数为(　　)。

A. 6　　B. 7　　C. 8　　D. 10

【答案】B

【分析】分治算法，在 2^n-1 个元素中采用折半搜索，最大比较次数为 n 次。而 $2^6-1=63<100<127=2^7-1$。

11.【NOIP2009】有一个由 4000 个整数构成的顺序表，假定表中的元素已经按升序排列，采用二分查找定位一个元素。则最多需要(　　)比较就能确定是否存在所查找的元素。

A. 11 次　　B. 12 次　　C. 13 次　　D. 14 次

【答案】B

【分析】分治：在二分查找的时候，可以每次去掉总数的一半，最后剩下 1 个的时候，就说明找到了，每次剩下的总数变化过程为：

4000－>2000－>1000－>500－>250－>125－>63－>32－>16－>8－>4－>2－>1；则只需要 12 次就可以定位到所需元素。我们还可以发现，$2^{11}-1=2047$，$2^{12}-1=4095$，排序二叉树高度也为 12。

12.【NOIP2008】对有序数组{5，13，19，21，37，56，64，75，88，92，100}进行二分查找，成功查找元素 19 的查找长度(比较次数)是(　　)。

A. 1　　B. 2　　C. 3　　D. 4

【答案】B

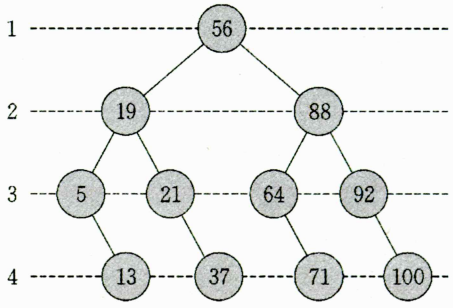
【分析】分治算法，需要了解二分查找，也叫折半查找。第一次 l=1，r=11，mid=6，a[mid]=56，第二次 l=1，r=5，mid=3，a[mid]=19。

13.【NOIP2008】对有序数组{5，13，19，21，37，56，64，71，88，92，100}进行二分查找，等概率的情况下查找成功的平均查找长度(平均比较次数)是(　　)。

A. 35/11　　B. 34/11　　C. 33/11　　D. 32/11　　E.34/10

【答案】C

【分析】分治算法，根据二分查找的特点，可以构造二叉树(见下图所示)：



56 需要查找 1 次；19 和 88 需要查找 2 次；5、21、64、92 需要查找 3 次，13、37、71、100 需要查找 4 次，平均查找长度(ASL)＝查找次数和/总个数$=\frac{1*1+2*2+3*4+4*4}{11}=\frac{33}{11}$。

14.【NOIP2015】在数据压缩编码的应用中，哈夫曼(Huffman)算法是一种采用了(　　)思想的算法。

A. 贪心　　B. 分治　　C. 递推　　D. 回溯

【答案】A

【分析】哈夫曼算法用了贪心思想。

15.【NOIP2008】将数组{8,23,4,16,77,−5,53,100}中的元素按从大到小的顺序排列，每次可以交换任意两个元素，最少需要交换(　　)次。

A. 4　　B. 5　　C. 6　　D. 7

【答案】B

【分析】贪心：先将数据从大到小排好序，即{100,77,53,23,16,8,4,−5}，与原序列{8,23,4,16,77,−5,53,100}进行比较。通过1次交换，最好的情况是2个数归位，可以发现4、53形成一个轮换序列，用1次交换，可以使这两个数归位；然后，考虑通过2次交换，最好的情况是有3个数归位，我们发现其中100、8、−5形成一个轮换序列，需要2次交换，其中77、23、16形成一个轮换序列，需要2次交换，至此序列已经有序了，总计5次交换。

16.【NOIP2018】给定一个含N个不相同数字的数组，在最坏情况下，找出其中最大或最小的数，至少需要N−1次比较操作。则最坏情况下，在该数组中同时找最大与最小的数至少需要(　　)次比较操作。(⌈ ⌉表示向上取整，⌊ ⌋表示向下取整)

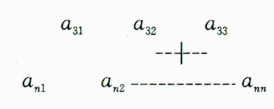
A. ⌈3N/2⌉−2　　B. ⌊3N/2⌋−2　　C. 2N−2　　D. 2N−4

【答案】A

【分析1】贪心：数组大小为N，若N为偶数，则需要$\frac{3*(N-2)}{2}+1=\frac{3N}{2}-2$次比较；若N为奇数，则需要$\frac{3(N-1)}{2}$次比较。偶数先比较前两个数，作为最大值和最小值的初始值，需要一次比较次数。奇数将第一个元素作为最大值和最小值的初始值。再以两个数为一组，首先比较这两个数的大小，然后分别与当前最小值和最大值比较。两个公式可以合并成$\left\lceil\frac{3N}{2}\right\rceil-2$。问题规模缩减的策略类似过河问题，策略一：一次性减少2个规模，需要3次比较。策略二：一次性减少1个规模，需要2次比较，当用策略二减少2个规模时，需要4次比较，因此贪心选择策略一。

【分析2】代入法，假设N=1时，需要0次比较，N=2时，需要1次比较。

17.【NOIP2017】有正实数构成的数字三角形排列形式。



第一行的数为a_{11}，第二行的数从左到右依次为a_{21}，a_{22}，…，第n行的数为a_{n1}，a_{n2}，…，a_{nn}。从a_{11}开始，每一行的数a_{ij}只有两条边可以分别通向下一行的两个数$a_{(i+1)j}$和$a_{(i+1)(j+1)}$。用动态规划算法找出一条从a_{11}向下通到a_{n1}，a_{n2}，…，a_{nn}中某个数的路径，使得该路径上的数之和达到最大。

令C[i,j]是从a_{11}到a_{ij}的路径上的数的最大和，并且C[i,0]=C[0,j]=0，则C[i,j]=(　　)。

A. max{C[i-1,j-1],C[i-1,j]}+a_{ij}　　B. C[i-1,j-1]+C[i-1,j]

C. max{C[i-1,j-1],C[i-1,j]}+1　　D. max{C[i,j-1],C[i-1,j]}+a_{ij}

【答案】A

【分析】经典 dp，每个位置 C[i][j]可以由 C[i-1][j-1]，C[i-1][j]两个状态更新，那么要求路径和最大，则取这两个位置的较大值，加上当前这个结点上的数字 a_{ij}。

不定项选择题

1.【NOIP2009】散列表的地址区间为 0－10，散列函数为 H(K)＝K%11。采用开地址法的线性探查法处理冲突，并将关键字序列 26，25，72，38，8，18，59 存储到散列表中，这些元素存入散列表的顺序并不确定。假定之前散列表为空，则元素 59 存放在散列表中的可能地址有（　　）。

A. 5　　B. 7　　C. 9　　D. 10

【答案】ABC

【分析】搜索计算各个的散列值，分别为 4、3、6、5、8、7、4。5 有可能为 26、59，7 有可能为 26、38、72、59，9 有可能为 26、38、72、18、8、59。

2.【NOIP2012】从顶点 A_0 出发，对有向图（　　）进行广度优先搜索(BFS)时，一种可能的遍历顺序是 A_0，A_1，A_2，A_3，A_4。

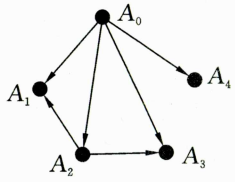


图 A

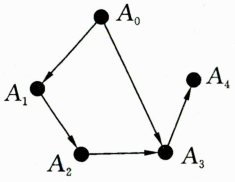


图 B

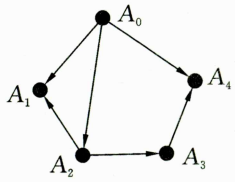


图 C

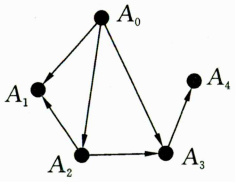


图 D

【答案】AD

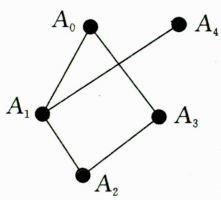
【分析】搜索，广度优先搜索相对于是树上的层次遍历，要按照队列的进队出队完成操作，B 中的 A_2 节点不能在第 3 位访问到；C 中 A_3 是最后一个访问到；所以 B 和 C 都不符合要求。

3.【NOIP2013】以 A_0 作为起点，对下面的无向图进行深度优先遍历时（遍历的顺序与顶点字母的下标无关），最后一个遍历到的顶点可能是（　　）。

A. A_1　　B. A_2　　C. A_3　　D. A_4

【答案】CD

【分析】搜索，从题中右图 A_0 出发深度优先遍历，有 3 种可能的路径：(A_0，A_1，A_2，A_3，A_4)，(A_0，A_1，A_4，A_2，A_3)，(A_0，A_3，A_2，A_1，A_4)，因此 A_1、A_2 不可能是最后遍历到的顶点。

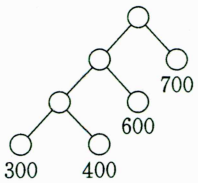


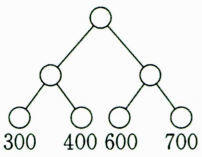
4.【NOIP2011】现有一段文言文，要通过二进制哈夫曼编码进行压缩。简单起见，假设这段文言文只由4个汉字“之”“乎”“者”“也”组成，它们出现的次数分别为700、600、300、400。那么，“也”字的编码长度可能是(　　)。

A. 1　　B. 2　　C. 3　　D. 4

【答案】BC

【分析】贪心，哈夫曼编码采取贪心思想，它实际上是构建了一棵二叉树，每次取出权重最小的两个点合并成新的点，新点权重为原来权重的和。最后这个点编码长度就是所在深度－1。这题最后的二叉树形态可能有两种：





第5节　字符数组与字符串

字符串实际上是用null字符即'\0'终止的一维字符数组。内容固定的字符串叫字符串常量，它用双引号括起来，而字符常量用单引号括起来。即使一个字符的字符串也不是字符类型，因为字符串末尾有'\0'作为结束符。

C++提供了两种类型的字符串表示形式：

✧ C语言风格的字符串，在头文件cstring中；

✧ C++引入的string类，在头文件string中。

一、C风格字符串

1.字符数组

```
#include<cstdio>
using namespace std;
int main(){
    char str[]="I love china";
    printf("%s\n",str);
    return 0;
}
```

2.字符指针

```
#include<cstdio>
using namespace std;
int main(){
    char *str="I love china";
    printf("%s\n",str);
    return 0;
}
```

二、C++引入的string类

C++标准库提供了string类类型，支持上述所有C风格字符串操作。

```
#include<iostream>
#include<string>
using namespace std;
int main(){
    string s1="Hello";
    string s2="world";
    string s3;
    int len;
    s3=s1;              //赋值s1到s3
    cout<<"s3:"<<s3<<endl;
    s3=s1+s2;           //连接s1和s2
    cout<<"s1+s2:"<<s3<<endl;
    len=s3.size();  //连接后s3的总长度
```

```
    cout<<"s3.size():"<<len<<endl;
    return 0;
}
```

三、字符数组和字符串相关函数总结

字符数组相关函数总结

函数	全拼	头文件	功能	实例
memset	memoryset	cstring	内存填充	memset (a,'b',sizeof(a))
gets	get string	cstdio	得到字符数组,允许字符数组里面有空格	gets(str)
puts	put string	cstdio	输出字符数组	puts (str)
atoi	ascii to integer	cstdlib	字符数组转数字	atoi(str)
strcat	string catenate	cstring	连接字符串	strcat(d,s);
strcpy	string copy	cstring	复制字符串	strcpy(char * des,const char * source)
memcpy	memory copy	cstring	复制字节	memcpy(d,s,(strlen(s)+1));
strcmp	stringcompare	cstring	比较字符串	strcmp (s1,s2);
strlen	string length	cstring	计算字符串长度	strlen(str);
strstr	string string	cstring	子串在母串的位置	strstr(s,sub);
strncpy	stringn copy	cstring	求子串	strncpy(char d[],const char s[], int numchars);
strncat	stringn catenate	cstring	结尾追加 n 个字符	strncat(char d[],const chars [],int numchars);

字符串(stl 容器)相关函数总结

函数	全拼	头文件	功能	实例
empty	empty	string	判断字符串是否为空	s.empty()
getline	getline	string	带空格的字符串输入	getline(cin,s)
size	size	string	求字符串的长度	s.size()
insert	insert	string	在母串某个位置插入子串	s.insert(pos,s2);
substr	substring	string	求子串	s.substr(pos,len)
erase	erase	string	删除特定长度字符	s.erase(pos,len)
find	find	string	查找子串的位置	s.find(sub)
replace	replace	string	替换子串	s.replace(pos,len,news)
c_str		string	获取 C 语言风格的字符数组首地址指针	s.c_str()

课堂练习

1.【NOIP2008】设字符串 S="Olympic",S 的非空子串的数目是(　　)。

A. 28　　B. 29　　C. 16　　D. 17

【答案】A

【分析】长度为 1 的有 7 个,长度不为 1 的有 $C_7^2=21$ 个(首尾各取一个位置),总数为 28。也可以用另外一种枚举的方法:7+6+5+4+3+2+1=28(取 1 个字符,取 2 个字符,……)。

2.【NOIP2012】原字符串中任意一段连续的字符组成的新字符串称为子串。则字符串"AAABBBCCC"共有(　　)个不同的非空子串。

A. 3　　B. 12　　C. 36　　D. 45

【答案】C

【分析】枚举:这题和上面那题的区别在于子串会出现重复的情况。总共长度 9,可以手动寻找全部的不同非空子串。长度为 1 的有 3 个:A、B、C;长度为 2 的有 5 个:AA、AB、BB、BC、CC;长度为 3 的有 7 个:AAA、AAB、ABB、BBB、BBC、BCC、CCC;长度为 4 的有 6 个:AAAB、AABB、ABBB、BBBC、BBCC、BCCC;长度为 5 的有 5 个:AAABB、AABBB、ABBBC、BBBCC、BBCCC;长度为 6 的有 4 个:AAABBB、AABBBC、ABBBCC、BBBCCC;长度为 7 的有 3 个:AAABBBC、AABBBCC、ABBBCCC;长度为 8 的有 2 个:AAABBBCC、AABBBCCC;长度为 9 的有 1 个:AAABBBCCC,共计 36 个。

3.【NOIP2017】若串 S="copyright",其子串的个数是(　　)。

A. 72　　B. 45　　C. 46　　D. 36

【答案】C

【分析】空串算 1 个,非空串的情况可以考虑在字符串中任取两个位置作为左端和右端得到的子串,方法数是 $C_{10}^2=45$,合起来就是 46。

4.【NOIP2016】以下关于字符串的判定语句中,正确的是(　　)。

A. 字符串是一种特殊的线性表　　B. 串的长度必须大于零

C. 字符串不可以用数组来表示　　D. 空格字符组成的串就是空串

【答案】A

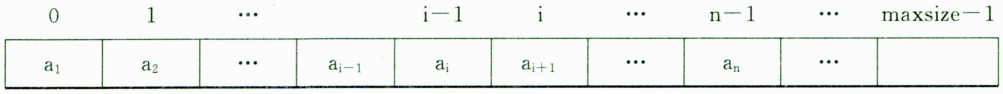
【分析】本题采用排除法。字符串允许为长度为 0 的空串,排除 B 选项;字符数组也可以实现字符串功能,排除 C 选项;空串指的是不包含任何字符,长度为 0 的字符串,因此排除 D 选项。

第6节 链表

线性表在内存中有顺序表和链表两种实现方式。

一、顺序表

用一组地址连续的存储单元依次存储线性表中的数据元素，此时线性表是顺序表，数据元素间的逻辑关系通过元素下标反映出来。



1.地址计算：$Loc(a_i)=Loc(a_1)+(i-1)*k$，$Loc(a_{i+1})=Loc(a_i)+k$。

2.特点：逻辑上相邻的元素在物理位置上也相邻。

3.优点：只需存放数据元素自身的信息，存储密度大，空间利用率高，存取速度快。

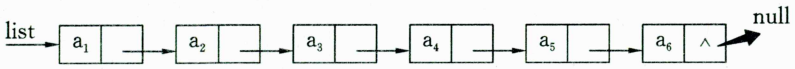
4.缺点：需事先分配存储空间，容易造成空间浪费，插入删除操作时效率低。

二、链表

用一组地址任意的存储单元（可以连续，也可不连续）依次存储线性表中的各个元素。链表可以用指针来实现，也可以用数组来实现。

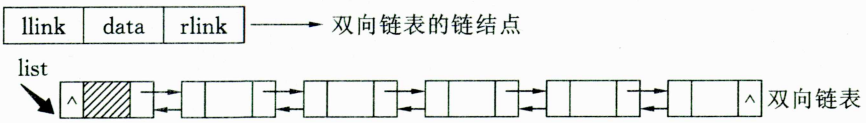
1.单向链表/线性链表

此链表中每个结点由两部分构成：元素自身信息即数据域（用 data 表示），指向直接后继元素位置的信息称为“指针域”（用 link 表示）。整个链表由一个称为外指针/头结点指针 list 指出，以表明链表的首地址，当链表为空时，list 为 null。用线性链表存储线性表时，数据元素间的逻辑关系通过指针反映出来。



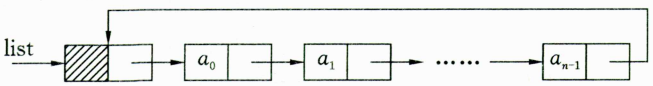
2.双向链表

双向链表的每个链结点除了数据域 data 外设置两个指针域，一个 llink 指向直接前驱结点，一个 rlink 指向直接后继结点。双向链表有循环线性和非循环线性，也可根据需要在链表前设置头结点 list。



3.循环链表

链表最后一个链结点的指针指向链表的第 1 个链结点，整个链表形成一个环，从表中任意结点出发均可找到表中其他结点。



课堂练习

1.【NOIP2014】链表不具有的特点是(　　)。

A. 不必事先估计存储空间　　B. 可随机访问任一元素

C. 插入、删除不需要移动元素　　D. 所需空间与线性表长度成正比

【答案】B

【分析】链表无法随机访问任意元素，查找任一结点都需要进行复杂度为O(n)的顺序查找过程。链表是一种物理存储单元上非连续、非顺序的存储结构，数据元素的逻辑顺序是通过链表中的指针链接次序实现的。链表由一系列结点(链表中每一个元素称为结点)组成，结点可以在运行时动态生成。每个结点包括两个部分：一个是存储数据元素的数据域，另一个是存储下一个结点地址的指针域。相比于线性表顺序结构，操作复杂。由于不必须按顺序存储，链表在插入的时候可以达到O(1)的复杂度，比另一种线性表顺序表快得多，但是查找一个结点或者访问特定编号的结点则需要O(n)的时间。

2.【NOIP2015】线性表若采用链表存储结构，要求内存中可用存储单元地址(　　)。

A. 必须连续　　B. 部分地址必须连续

C. 一定不连续　　D. 连续不连续均可

【答案】D

【分析】考查数据结构基础知识。数组存储单元地址要连续，而链表可以在产生新结点的时候动态申请内存(不一定连续)，也可以事先申请一部分连续的内存来给未来新增的结点预留空间。链表存储单元地址连续或者不连续都可以。

3.【NOIP2011】在含有n个元素的双向链表中查询是否存在关键字为k的元素，最坏情况下运行的时间复杂度是(　　)。

A. O(1)　　B. O(logn)　　C. O(n)　　D. O(nlogn)

【答案】C

【分析】在链表里查询一个数需要一个一个扫过去，最坏情况是最后一个才找到或未找到，所以是O(n)的。

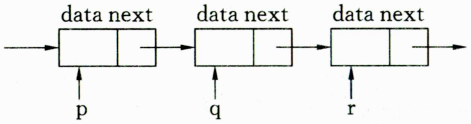
4.【NOIP2014】对长度为n的有序单链表，若检索每个元素的概率相等，则顺序检索到表中任一元素的平均检索长度为(　　)。

A. n/2　　B. (n+1)/2　　C. (n−1)/2　　D. n/4

【答案】B

【分析】第1个数需要检索1次，第2个数需要检索2次，第3个数需要检索3次，……平均检索长度＝总检索次数/个数$=\frac{1+2+3+\cdots+n}{n}=\frac{n+1}{2}$。

5.【NOIP2014】有以下结构体说明和变量定义，如图所示，指针p、q、r分别指向一个链表中的三个连续结点。



```
struct node {
    int data;
    node*next;
}*p,*q,*r;
```

现要将 q 和 r 所指结点的先后位置交换，同时要保持链表的连续，以下程序段中错误的是(　　)。

A. q->next=r->next;p->next=r;r->next=q;

B. p->next=r;q->next=r->next;r->next=q;

C. q->next=r->next;r->next=q;p->next=r;

D. r->next=q;q->next=r->next;p->next=r;

【答案】D

【分析】本题考查指针的理解和应用。r->next=q;q->next=r->next;，第一步操作将 r 的后继指向了 q，所以第二步操作的时候实质是将 q 的后继指向了 q 本身而不是原来 r 的后继。

6.【NOIP2010】双向链表中有两个指针域 llink 和 rlink，分别指向该结点的前驱及后继。设 p 指向链表中的一个结点，它的左右结点均非空。现要求删除结点 p，则下面语句序列中错误的是(　　)。

A. p->rlink->llink=p->rlink;p->llink->rlink=p->llink;delete p;

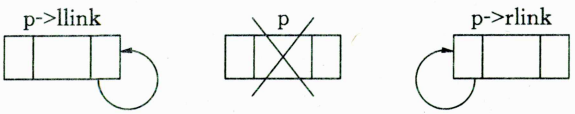
B. p->llink->rlink=p->rlink;p->rlink->llink=p->llink;delete p;

C. p->rlink->llink=p->llink;p->rlink->llink->rlink=p->rlink;delete p;

D. p->llink->rlink=p->rlink;p->llink->rlink->llink=p->llink;delete p;

【答案】A

【分析】A 中将 p 的后继的前驱改为指向 p 的后继，p 的前驱的后继改为指向 p 的前驱，这样链表就中断了。



7.【NOIP2015】双向链表中有两个指针域 llink 和 rlink，分别指向前驱及后继，设 p 指向链表中的一个结点，q 指向一待插入结点，现要求在 p 前插入 q，则正确的插入为(　　)。

A. p->llink=q;q->rlink=p;
p->llink->rlink=q;q->llink=p->llink;

B. q->llink=p->llink;p->llink->rlink=q;
q->rlink=p;p->llink=q->rlink;

C. q->rlink=p;p->rlink=q;
p->llink->rlink=q;q->rlink=p;

D. p->llink->rlink=q;q->rlink=p;
q->llink=p->llink;p->llink=q;

【答案】D

【分析】

A 选项的第一句已经把 p->llink 更改成了 q；第三句 p->llink->rlink=q；就等价于 q->rlink=q，指向了自身，造成链接对象丢失；

B 选项的第二句 p->llink->rlink=q；等价于 p=q，结果导致 p 被替换；

C 选项，由于要求在 p 前插入 q，因此第二句 p->rlink=q；不需要更改；

只有 D 可以达到正确的插入要求。要在 p 之前插入 q，首先应该让 q 的左右指针分别指向 p 的前驱和 p，然后让另外两个指针指向 p 即可，即 p 的前驱的右指针和 p 的左指针。

不定项选择题

1.【NOIP2009】在带尾指针（链表指针 clist 指向尾结点）的非空循环单链表中每个结点都以 next 字段的指针指向下一个节点。假定其中已经有 2 个以上的结点。下面说法正确的是（　　）。

A. 如果 p 指向一个待插入的新结点，在头部插入一个元素的语句序列为

p->next=clist->next;clist->next=p;

B. 如果 p 指向一个待插入的新结点，在尾部插入一个元素的语句序列为

p->next=clist;clist->next=p;

C. 在头部删除一个结点的语句序列为

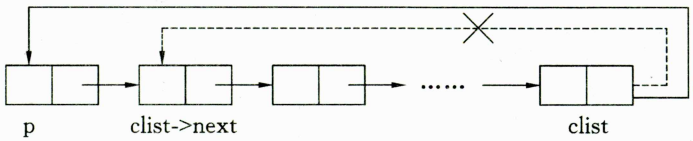
p=clist->next;clist->next=clist->next->next;delete p;

D. 在尾部删除一个结点的语句序列为

p=clist;clist=clist->next;delete p;

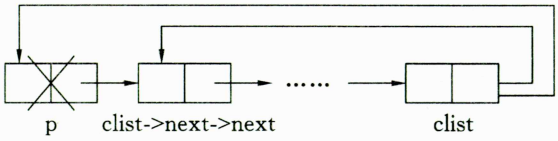
【答案】AC

【分析】A 选项是从前面插入 p 结点。



B 选项要求在尾部插入结点，插完之后 clist 要指向这个新的尾结点，所以 B 应该改为“p->next=clist->next;clist->next=p;clist=p;”。

C 选项的过程是删除第一个结点。



D 中要循环找到尾指针的上一个元素才能进行删除。

2.【NOIP2010】双向链表中有两个指针域 llink 和 rlink，分别指向该结点的前驱及后继。设 p 指向链表中的一个结点，它的左右结点均为非空。现要求删除结点 p，则下列语句序列中正确的是（　　）。

A. p->rlink->llink=p->rlink;p->llink->rlink=p->llink;delete p;

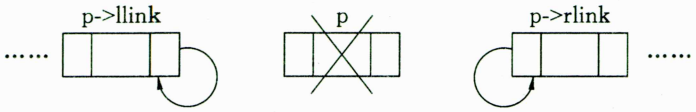
B. p->llink->rlink=p->rlink;p->rlink->llink=p->llink;delete p;

C. p->rlink->llink=p->llink;p->rlink->llink->rlink=p->rlink;delete p;

D. p->llink->rlink=p->rlink;p->llink->rlink->link=p->llink;delete p;

【答案】BCD

【分析】选项A中的语句执行后，结点P右边的结点的左指针指向自身，左边的结点的右指针也指向自身。当p被释放后，其左、右结点之间便产生了断裂，不再是一个链表。其余选项经检验都是正确的。



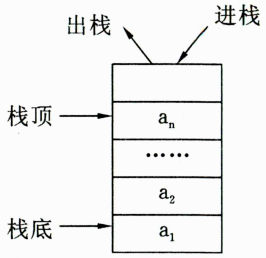
第7节　栈和队列

一、栈

栈是只能在某一端插入和删除的特殊线性表。

用桶堆积物品，先堆进来的压在底下，随后一件一件往上堆。取走时，只能从上面一件一件取。堆和取都在顶部进行，底部一般是不动的。栈就是一种类似桶堆积物品的数据结构，进行删除和插入的一端称栈顶，另一端称栈底。插入一般称为进栈(PUSH)，删除则称为退栈(POP)。栈也称为后进先出表(LIFO 表)。

一个栈可以用定长为 n 的数组 s 来表示，用一个栈指针 top 指向栈顶。若 top=0，表示栈空，top=n 则栈满。进栈时 top 加 1，退栈时 top 减 1，当 top<0 时为下溢，栈指针在运算中永远指向栈顶。



1.进栈(PUSH)算法

①若 top>=n 时，则给出溢出信息，作出错处理(进栈前首先检查栈是否已满，满则溢出；不满则作②)；

②top++(栈指针加 1，指向进栈地址)；

③s[top]=x，结束(x 为新进栈的元素)；

2.退栈(POP)算法

①若 top<=0，则给出下溢信息，作出错处理(退栈前先检查是否已为空栈，空则下溢；不空则作②)；

②x=s[top]，(退栈后的元素赋给 x)；

③top--，结束(栈指针减 1，指向栈顶)。

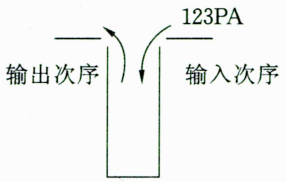
进栈、出栈的 C++实现过程程序：

```
#define n 100
void push(int s[],int *top,int *x)              //入栈
{
    if (*top==n)printf("overflow");
    else {*top++;s[*top]=*x;}
}
void pop(int s[],int *y,int *top)               //出栈
{
    if (*top==0)printf("underflow");
    else {*y=s[*top];*top--;}
}
```

对于出栈运算中的“下溢”，程序中仅给出了一个标志信息，而在实际应用中，下溢作为控制程序转移的判断标志，是十分有用的。对于入栈运算中的“上溢”，则是一种致命错误，将使程序无法继续运行，所以要设法避免。

从历届初赛可以看出，栈也是属于比较容易出题的知识点，选择题、解答题等题型都有可能出现。

例题 1 设输入元素为 1、2、3、P 和 A，输入次序为 123PA，如图所示。元素经过栈后到达输出序列，当所有元素均到达输出序列后，有哪些序列可以作为高级语言的变量名？



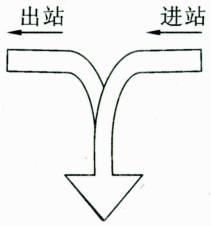
【解答】高级语言变量名的定义规则：以字母开头，字母与数字的组合。

由于必须以字母开头，所以，第一个可能出现的字母是 P，那么必然要求 123 已经先后入栈，这样便可得到以 P 开头的、并且 123 按逆序排列的（即 321）、同时 A 位于 P 后任一位置的变量名序列。此外，还需要考虑以 A 开头的可能情况，这只有一种情形为 AP321。所以 AP321，PA321，P3A21，P32A1，P321A 可以作为高级语言的变量名。

例题 2 火车站列车调度，假设调度站两侧的轨道为单向行驶轨道。

①如果进站的车厢序列为 123，则可能的出站车厢序列是什么？

②如果进站的车厢序列为 123456，问能否得到 135426 和 435612 的出站序列。



二、队列

队列是限定在一端进行插入，另一端进行删除的特殊线性表。就像排队买东西，排在前面的人买完东西后离开队伍（删除），而后来的人总是排在队伍末尾（插入）。通常把队列的删除和插入分别称为出队和入队。允许出队的一端称为队头，允许入队的一端称为队尾。所有需要进队的数据项，只能从队尾进入，队列中的数据项只能从队头离去。由于总是先入队的元素先出队（先排队的人先买完东西），这种表也称为先进先出（FIFO）表。

队列可以用数组 Q[m+1]来存储，数组的上界 m 即是队列所容许的最大容量。在队列的运算中需设两个指针：

head：队头指针，指向实际队头元素的前一个位置；

tail：队尾指针，指向实际队尾元素所在的位置。

一般情况下，两个指针的初值设为 0，这时队列为空，没有元素。图 1(a)画出了一个由 6 个元素构成的队列，数组定义 Q[11]。

Q(i) i=3,4,5,6,7,8，头指针 head=2，尾指针 tail=8。

队列中拥有的元素个数为 L＝tail－head，现要让排头的元素出队，则需将头指针加 1，即 head++。这时头指针向上移动一个位置，指向 Q(3)，表示 Q(3)已出队，见图 1(b)。

如果想让一个新元素入队，则需尾指针向上移动一个位置，即 tail++，这时 Q(9)入队，见图 1(c)。

当队尾已经处在最上面时，即 tail＝10，见图 1(d)，如果还要执行入队操作，则要发生“上溢”，但实际上队列中还有三个空位置，所以这种溢出称为“假溢出”。

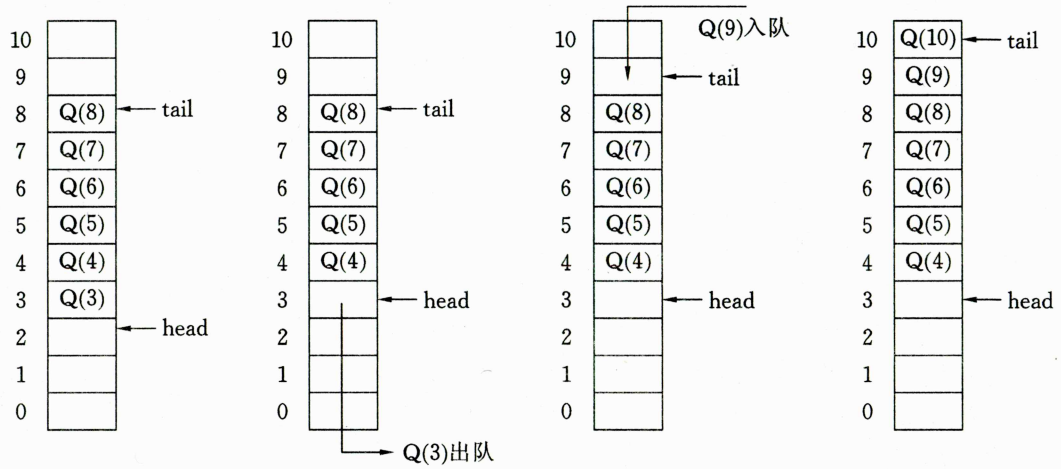


图 1　队列

克服假溢出的方法有两种。一种是将队列中的所有元素均向低地址区移动，显然这种方法很浪费时间；另一种方法是将数组存储区看成是一个首尾相接的环形区域。当存放到 n 地址后，下一个地址就“翻转”为 1，在结构上采用这种技巧来存储的队列称为循环队列，见图 2。

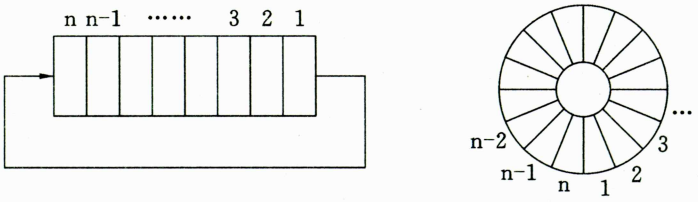


图 2　循环队列

循环队列的入队算法如下：

1.tail++；

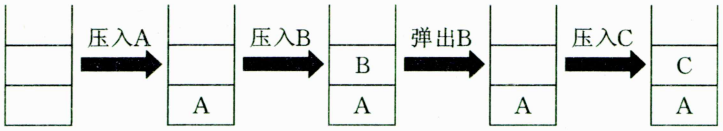
2.若 tail＝n＋1，则 tail＝1；

3.若 head＝tail，尾指针与头指针重合，表示元素已装满队列，则作上溢出错处理；

4.否则，Q(tail)＝x，结束(x 为新入队元素)。

课堂练习

1.【NOIP2013】【NOIP2018】下图中所使用的数据结构是(　　)。



A. 哈希表　　B. 栈　　C. 队列　　D. 二叉树

【答案】B

【分析】图中描述的是一个后进先出的容器表达,因此是栈。

2.【NOIP2008】设栈S的初始状态为空,元素a,b,c,d,e,f依次入栈S,出栈的序列为b,d,f,e,c,a,则栈S的容量至少应该是(　　)。

A. 6　　B. 5　　C. 4　　D. 3

【答案】C

【分析】满足"后进先出"原则,栈最多元素时是4个,从上到下依次为f,e,c,a。下图每一列描述了栈内元素的变化情况(最下面为栈底)。

		f			
	d	e	e		
b	c	c	c	c	
a	a	a	a	a	a

3.【NOIP2009】有六个元素FEDCBA从左至右依次顺序进栈,在进栈过程中会有元素被弹出栈。下列不可能是合法出栈序列的是(　　)。

A. EDCFAB　　B. DECABF　　C. CDFEBA　　D. BCDAEF

【答案】C

【分析】C选项中当C出栈,此时F是被压在E下,不可能先于E出栈。如下图

C
D
E
F

4.【NOIP2010】元素R1,R2,R3,R4,R5入栈的顺序为R1,R2,R3,R4,R5。如果第一个出栈的是R3,那么第5个出栈的不可能是(　　)。

A. R1　　B. R2　　C. R4　　D. R5

【答案】B

【分析】R3出栈时,R2为栈顶,R1在R2之下,肯定晚于R2之后出栈,R2不可能最后一个出栈,自然不会是第5个出栈。选项A的可以出栈顺序为R3,R4,R5,R2,R1;选项C的可以出栈顺序为R3,R2,R1,R5,R4;选项D的可以出栈顺序为R3,R2,R1,R4,R5。

5.【NOIP2012】如果一个栈初始时为空，且当前栈中的元素从栈底到栈顶依次为 a,b,c(如右图所示)，另有元素 d 已经出栈，则可能的入栈顺序是(　　)。

c	栈顶
b	
a	栈底

A. a,d,c,b　　B. b,a,c,d　　C. a,c,b,d　　D. d,a,b,c

【答案】D

【分析】栈的性质决定了栈里面三个元素的相对入栈顺序是 a,b,c，结合四个选项，只有 D 选项符合 a,b,c 三个元素的相对位置。

6.【NOIP2015】今有一空栈 S，对下列待进栈的数据元素序列 a,b,c,d,e,f 依次进行进栈，进栈，出栈，进栈，进栈，出栈的操作，则此操作完成后，栈 S 的栈顶元素为(　　)。

A. f　　B. c　　C. a　　D. b

【答案】B

【分析】按照题目意思模拟一下即可。

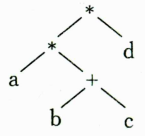
操作：进栈当前栈(带方括号表示栈顶元素)为[a]；
操作：进栈当前栈(带方括号表示栈顶元素)为 a,[b]；
操作：出栈当前栈(带方括号表示栈顶元素)为[a]；
操作：进栈当前栈(带方括号表示栈顶元素)为 a,[c]；
操作：进栈当前栈(带方括号表示栈顶元素)为 a,c,[d]；
操作：出栈当前栈(带方括号表示栈顶元素)为 a,[c]。

7.【NOIP2017】表达式 a*(b+c)*d 的后缀形式是(　　)。

A. abcd*+*　　B. abc+*d*
C. a* bc+ * d　　D. b+ c* a* d

【答案】B

【分析 1】本题考查数据结构中的栈与后缀表达式的计算方法。后缀表达式的计算过程是从左往右遇到数值就进栈，遇到一个运算符就将栈顶的两个数值弹出进行运算，将结果进栈。如果该式子的运算顺序和题干中的运算顺序一致即是答案。如图



【分析 2】通过加括号和去括号的方法来求解，先针对上面的表达式添加上括号：

((a*(b+c))* d)

然后一层层去掉括号，并将运算符放在括号的右边。

(a*(b+c))　d*
a　(b+c)*　d*
ab　c　+*　d*

8.【NOIP2017】对于入栈顺序为 a,b,c,d,e,f,g 的序列，下列不可能是合法的出栈序列的是(　　)。

A. a,b,c,d,e,f,g　　B. a,d,c,b,e,g,f
C. a,d,b,c,g,f,e　　D. g,f,e,d,c,b,a

【答案】C

【分析】考查栈的相关知识。可以逐一尝试模拟操作。

a,b,c,d,e,f,g:每个元素入栈后立即出栈。

a,d,c,b,e,g,f:入栈 a,弹出 a;入栈 bcd,弹出 dcb;入栈 e,弹出 e;入栈 fg,弹出 gf。

g,f,e,d,c,b,a:先把所有字母进栈,再依次出栈。

正确选项中,无论如何操作,都不可能使得 b 出现在 c 的前面。

9.【NOIP2011】广度优先搜索时,需要用到的数据结构是(　　)。

A. 链表　　B. 队列　　C. 栈　　D. 散列表

【答案】B

【分析】广度优先搜索,即 bfs,使用的数据结构是队列,而深度优先搜索(dfs)使用的数据结构是栈。

10.【NOIP2017】向一个栈顶指针为 hs 的链式栈中插入一个指针 s 指向的结点时,应执行(　　)。

A. hs->next=s;

B. s->next=hs;hs=s;

C. s->next=hs->next;hs->next=s;

D. s->next=hs;hs=hs->next;

【答案】B

【分析】本题考察数据结构基础知识。栈顶为 hs,此时 s 要进栈,由于是链式栈,所以首先将 s 的下一个节点位置指向 hs,然后把 hs 指向 s 所指向的地址。

11.【NOIP2012】(　　)是一种先进先出的线性表。

A. 栈　　B. 队列　　C. 哈希表(散列表)　D. 二叉树

【答案】B

【分析】队列是一个抽象的数据结构,它有先进先出的特性,不过值得一提的是它既能够用线性表表示,同样也能用链表表示。

不定项选择题

1.【NOIP2010】元素 R1,R2,R3,R4,R5 入栈的顺序为 R1,R2,R3,R4,R5。如果第 1 个出栈的是 R3,那么第 5 个出栈的可能是(　　)。

A. R1　　B. R2　　C. R4　　D. R5

【答案】ACD

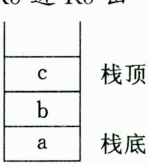
【分析】栈的特点是先进后出,因为 R1 比 R2 先进,因此肯定在 R2 后出栈。

A 选项的入栈出栈次序为 R1 进 R2 进 R3 进 R3 出 R2 出 R4 进 R5 进 R5 出 R4 出 R1 出

C 选项的入栈出栈次序为 R1 进 R2 进 R3 进 R3 出 R2 出 R1 出 R4 进 R5 进 R5 出 R4 出

D 选项的入栈出栈次序为 R1 进 R2 进 R3 进 R3 出 R2 出 R1 出 R4 进 R4 出 R5 进 R5 出

2.【NOIP2012】如果一个栈初始时为空,且当前栈中的元素从栈底到栈顶依次为 a,b,c(如右图所示),另有元素 d 已经出栈,则可能的入栈顺序有(　　)。



A. a,b,c,d　　B. b,a,c,d　　C. a,c,b,d　　D. d,a,b,c

【答案】AD

【分析】栈的性质决定了栈里面三个元素的相对入栈顺序是a,b,c,结合四个选项,A、D选项符合a,b,c三个元素的相对位置。A选项的出入栈顺序为a进,b进,c进,d出;D选项的出入栈顺序为d进,d出a进,b进,c进。

3.【NOIP2017】对于入栈顺序为a,b,c,d,e,f,g的序列,下列(　　)不可能是合法的出栈序列。

A. a,b,c,d,e,f,g　　C. a,d,b,c,g,f,e

B. a,d,c,b,e,g,f　　D. g,f,e,d,c,b,a

【答案】C

【分析】C选项因为b比c先入栈,不可能比c先出栈。A选项的出入栈顺序为a进,a出,b进,b出,c进,c出,d进,d出,e进,e出,f进,f出,g进,g出;B选项的出入栈顺序为a进,a出,b进,c进,d进,d出,c出,b出,e进,e出,f进,g进,g出,f出;D选项的出入栈顺序为a进,b进,c进,d进,e进,f进,g进,g出,f出,e出,d出,c出,b出,a出;

第8节 树

在实际生活中，可以用线性结构描述数据元素之间逻辑关系的问题是很广泛的，但也有很多问题不能依靠线性结构来解决，例如家谱、行政组织机构等都是非线性的数据结构。其中树就是一种非线性的数据结构。

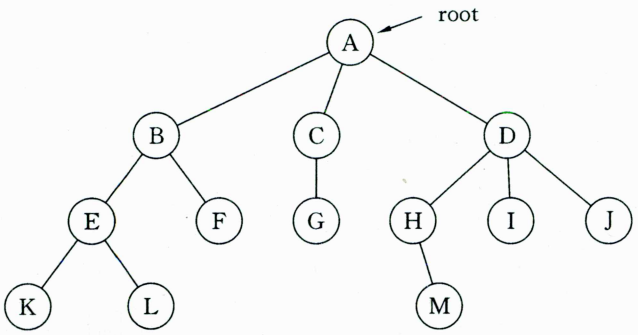
一、树的定义

树作为一种非线性的数据结构，是由 n(n≥0)个结点组成的有限集合。

1.如果 n 为 0 时，树为空树；

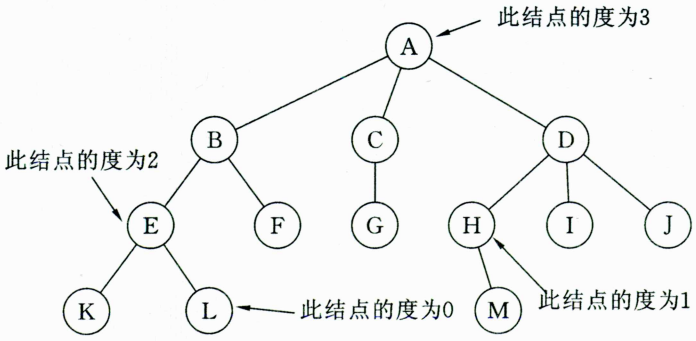
2.如果 n>0 时，树有一个特定的结点——根结点。

根结点只有直接后继，没有直接前驱。除根结点以外的其他结点划分为 m(m≥0)个互不相交的有限集合 T0，T1，T2，…，Tm－1，每个集合是一棵树，称为根结点的子树。树的示例如下：

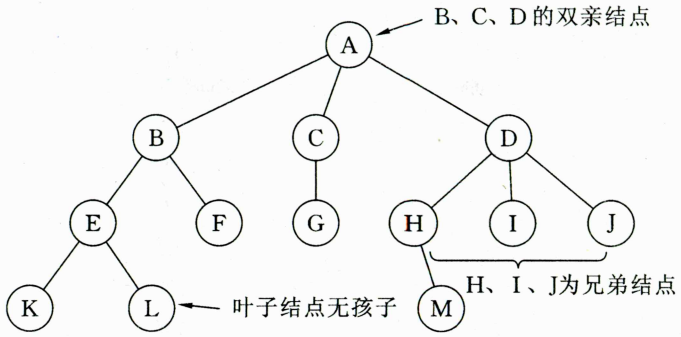


二、树的相关概念

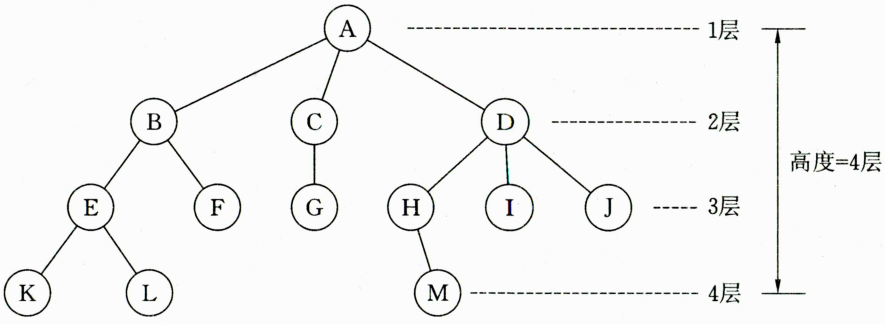
1.树的度：结点拥有的子树的数量为结点的度，度为 0 的结点是叶结点，度不为 0 的结点为分支结点，树的度定义为树的所有结点中度的最大值。



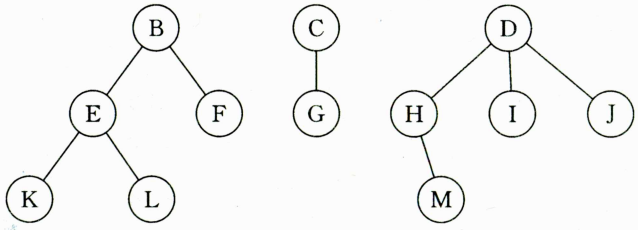
2.树的前驱和后继：除根结点没有前驱外，其余每个结点都有唯一的一个前驱结点。每个结点可以有 0 或多个后继结点。结点的直接后继称为结点的孩子，结点的直接前驱称为结点的父亲。结点的孩子的孩子称为结点的孙子，结点称为子孙的祖先。同一个双亲的孩子之间互称兄弟。



3.树中结点的层次：树中根结点为第1层，根结点的孩子为第2层，依次类推。树中结点的最大层次称为树的深度或高度。



4.森林：是由n(n>=0)棵互不相交的树组成的集合。



三、树的性质

1.除根结点没有父结点外，其余结点有且仅有一个父结点。

2.n个结点的树，有且仅有n−1条边。

3.树中任意两个结点之间有且仅有一条简单路径(指路径上的顶点都不相同的路径，不存在自环和重边)。

四、二叉树

二叉树(binary tree，简写BT)是一种度数为2的树，即二叉树的每个结点最多有两个子结点。每个结点的子结点分别称为左孩子、右孩子，它的两棵子树分别称为左子树、右子树。

1.二叉树的遍历

所谓的遍历是指按一定的规律和次序访问树中的各个结点，而且每个结点仅被访问一次。遍历一般按照从左到右的顺序，共有4种遍历方法：前序遍历、中序遍历、后序遍历、层次遍历。

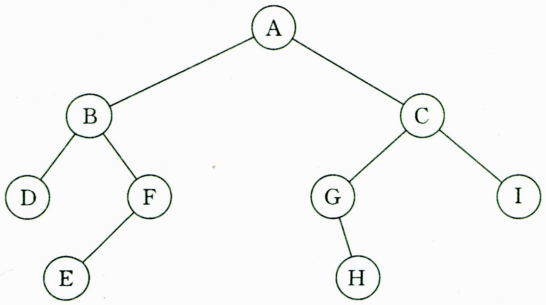
(1)先序遍历：先访问根结点，再访问左子树，最后访问右子树。

(2)后序遍历:先左子树,再右子树,最后根结点。

(3)中序遍历:先左子树,再根结点,最后右子树。

(4)层次遍历:每一层从左到右访问每一个结点。

例 1 求下图所示树的各种遍历结果。



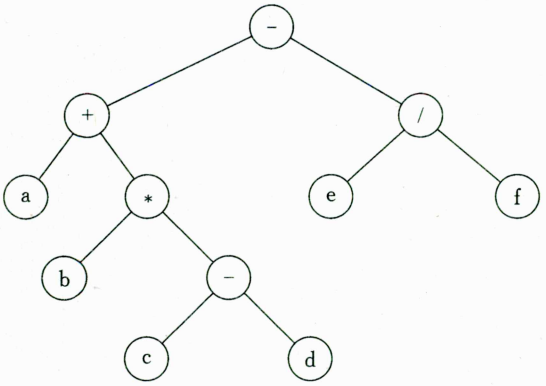
先序遍历结果:ABDFECGHI。

中序遍历结果:DBEFAGHCI。

后序遍历结果:DEFBHGICA。

层次遍历结果:ABCDFGIEH。

例 2 关于前面讲的表达式树,我们可以分别用先序、中序、后序的遍历方法得出完全不同的遍历结果,如对于下图遍历结果如下,它们正好对应表达式的 3 种表示方法。



表达式(a+b*(c-d)-e/f)的二叉树

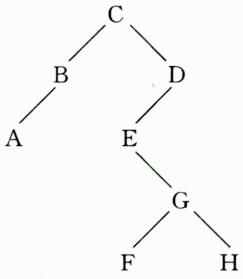
-+a*b-cd/ef (前缀表示、波兰式)

a+b*c-d-e/f (中缀表示)

abcd-*+ef/- (后缀表示、逆波兰式)

结论:已知前序序列和中序序列可以确定出二叉树;已知中序序列和后序序列也可以确定出二叉树;但已知前序序列和后序序列却不可以确定出二叉树。

例 3 有二叉树中序序列为 ABCEFGHD,后序序列为 ABFHGEDC。请画出此二叉树,并求前序序列。



解答:根据后序序列知根结点为 C,

因此左子树:中序序列为 AB,

后序序列为 AB;

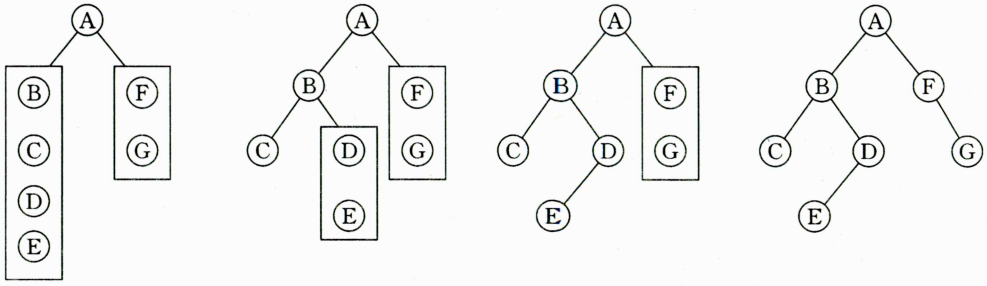
右子树:中序序列为 EFGHD,

后序序列为 FHGED;

依次推得该二叉树的结构图(如右图)。

前序序列为 CBADEGFH。

例 4　已知结点的前序序列为 ABCDEFG，中序序列为 CBEDAFG。构造出二叉树。过程见下图：



2.二叉树的性质

【性质 1】在二叉树的第 i 层上最多有 2^{i-1} 个结点($i \geqslant 1$)。

证明：很简单，用归纳法。当 $i=1$ 时，$2^{i-1}=1$ 显然成立；现在假设第 $i-1$ 层时命题成立，即第 $i-1$ 层上最多有 2^{i-2} 个结点。由于二叉树的每个结点的度最多为 2，故在第 i 层上的最大结点数为第 $i-1$ 层的 2 倍，即 $2 * 2^{i-2}=2^{i-1}$。

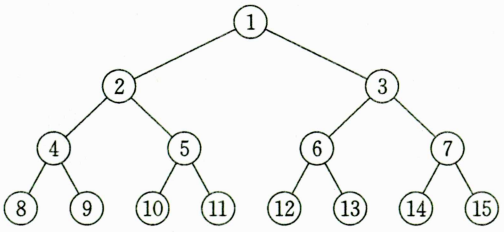
【性质 2】深度为 k 的二叉树至多有 2^k-1 个结点($k \geqslant 1$)。

证明：在具有相同深度的二叉树中，仅当每一层都含有最大结点数时，其树中结点数最多。因此利用性质 1 可得，深度为 k 的二叉树的结点数至多为 $2^0+2^1+\cdots+2^{k-1}=2^k-1$，故命题正确。

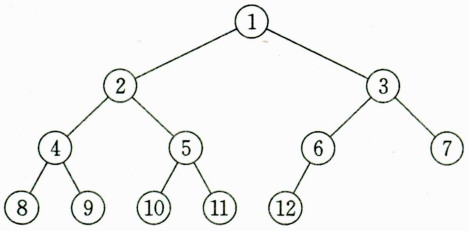
特别：一棵深度为 k 且有 2^k-1 个结点的二叉树称为满二叉树。如图 A 为深度为 4 的满二叉树，这种树的特点是每层上的结点数都是最大结点数。

可以对满二叉树的结点进行连续编号，约定编号从根结点起，自上而下，从左到右，由此引出完全二叉树的定义，深度为 k，有 n 个结点的二叉树当且仅当其每一个结点都与深度为 k 的满二叉树中编号从 1 到 n 的结点一一对应时，称为完全二叉树。

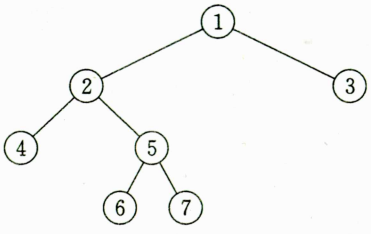
下图 B 就是一个深度为 4，结点数为 12 的完全二叉树。它有如下特征：叶结点只可能在层次最大的两层上出现；对任一结点，若其右分支下的子孙的最大层次为 m，则在其左分支下的子孙的最大层次必为 m 或 m+1。下图 C、D 不是完全二叉树，请大家思考为什么？



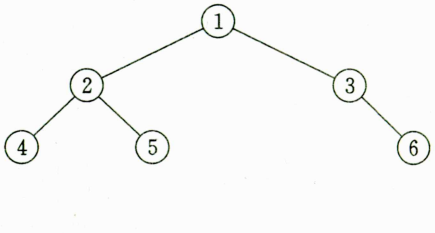
图A



图B



图C



图D

【性质 3】对任意一棵二叉树，如果其叶结点数为 n_0，度为 2 的结点数为 n_2，则一定满足 $n_0=n_2+1$。

证明：因为二叉树中所有结点的度数均不大于 2，所以结点总数（记为 n）应等于 0 度结点数 n_0、1 度结点 n_1 和 2 度结点数 n_2 之和：

$$n=n_0+n_1+n_2\cdots\cdots(式子\ 1)$$

另一方面，1 度结点有一个孩子，2 度结点有两个孩子，故二叉树中孩子结点总数是

$$n_1+2n_2$$

树中只有根结点不是任何结点的孩子，故二叉树中的结点总数又可表示为

$$n=n_1+2n_2+1\cdots\cdots(式子\ 2)$$

由式子 1 和式子 2 得到：

$$n_0=n_2+1$$

例题 1 有 n 个结点的二叉树，已知叶结点个数为 n_0，写出度为 1 的结点的个数 n_1 的计算公式；若此树是深度为 k 的完全二叉树，写出 n 为最小的公式；若二叉树中仅有度为 0 和度为 2 的结点，写出求该二叉树结点个数 n 的公式。

【解答】 (1)记度为 2 的结点个数为 n_2，

则 $n=n_0+n_1+n_2$ ①

又因为除了根结点以外，其余结点均由父结点发出，所以

$n=1+n_1+2*n_2$ ②

由①②两式得 $n_1=n+1-2n_0$。

(2)当树是深度为 k 的完全二叉树时，n 的最小值 $n_{min}=2^{k-1}$。

(3)当二叉树中仅有度为 0 和度为 2 的结点时，

$n=n_0+n_2$，$n=2n_2+1$，

所以 $n_0=n_2+1$，

$n=2n_0-1$。

【性质 4】具有 n 个结点的完全二叉树的深度为 $floor(\log_2 n)+1$。

证明：假设深度为 k，根据完全二叉树的定义，前面 k−1 层一定是满的，所以 $n>2^{k-1}-1$。但 n 又要满足 $n\leqslant 2^k-1$。所以，$2^{k-1}-1<n\leqslant 2^k-1$。变换一下为 $2^{k-1}\leqslant n<2^k$。

以 2 为底取对数得到：$k-1\leqslant \log_2 n<k$。而 k 是整数，所以 $k=floor(\log_2 n)+1$。

【性质 5】具有 n 个结点的二叉树的高度至少为 $\log_2(n+1)$

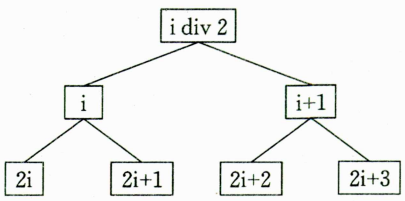
证明：根据“性质 2”可知，高度为 k 的二叉树最多有 2^k-1 个结点。反之，对于包含 n 个节点的二叉树的高度至少为 $\log_2(n+1)$。

【性质 6】对于一棵 n 个结点的完全二叉树的任一个结点（编号为 i），有以下关系：

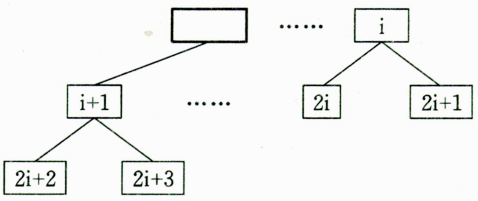
①如果 i=1，则结点 i 为根，无父结点；如果 i>1，则其父结点编号为 i/2。

②如果 2*i>n，则结点 i 无左孩子，否则左孩子编号为 2*i。如果 2*i+1>n，则结点 i 无右孩子，否则右孩子编号为 2*i+1。

证明：略，我们只要验证一下即可，总结如下图。



结点i和i+1在同一层上



结点i和i+1不在同一层上

例题 2　已知一棵完全二叉树共有 892 个结点，试求：

(1)树的高度；(2)叶子结点数；(3)单支结点数；(4)最后一个非终端结点的序号。

【解答】(1)已知深度为 k 的二叉树至多有 2^k-1 个结点($k\geqslant 1$)，由于 $2^9-1<892<2^{10}-1$，所以树的高度为 10。

(2)对完全二叉树来说，度为 1 的结点只能是 0 或 1。由 $n=n_0+n_1+n_2$ 和 $n_0=n_2+1$ 得：

①设 $n_1=0$，则有 $892=n_0+0+n_2=n_2+1+n_2=2n_2+1$，因得到的 n_2 不为整数而出错；

②设 $n_1=1$，则有 $892=n_0+1+n_2=n_2+1+1+n_2=2n_2+2$，得到 $n_2=445$，代入 $n_0=n_2+1$，得到 $n_0=446$，故叶子结点数为 446。

(3)由(2)可知单支结点数为 1。

(4)对有 n 个结点的完全二叉树，最后一个树叶结点，即序号为 n 的叶结点其双亲结点 n/2，即为最后一个非终端结点，也即序号为向下取整(892/2)=446。此外，由(2)可知 $n_2=445$，$n_1=1$；则最后一个非终端结点的序号为 445+1=446。

课堂练习

1.【NOIP2013】已知一棵二叉树有 10 个节点，则其中至多有(　　)个节点有 2 个子节点。

A. 4　　B. 5　　C. 6　　D. 7

【答案】A

【分析】对于任意一棵二叉树，度为 2 的结点数肯定比度为 0 的结点少 1，在一棵含 10 个结点的二叉树中，度为 2 的结点数至多为 4，如果超过 4，则总结点数将超过 10。完全二叉树满足度为 2 的结点数最多的情况。

2.【NOIP2013】已知一棵二叉树有 2013 个节点，则其中至多有(　　)个节点有 2 个子节点。

A. 1006　　B. 1007　　C. 1023　　D. 1024

【答案】A

【分析 1】对于任意一棵二叉树，度为 2 的结点数肯定比度为 0 的结点少 1，在一棵含 2013 个结点的二叉树中，度为 2 的结点数至多为 1006，如果超过 1006，则总结点数将超过 2013。

【分析 2】设度为 0 的结点数为 n_0，度为 1 的结点数为 n_1，度为 2 的结点数为 n_2，由题意：$n_0+n_1+n_2=2013$，在二叉树中有 $n_0=n_2+1$，所以有 $2*n_2+n_1=2012$；所以 n_1 的值为偶数，最小的值为 0，可以得到 $n_2=1006$，选 A。

3.【NOIP2011】如果根结点的深度记为 1，则一棵恰有 2011 个叶结点的二叉树的深度最少是(　　)。

A. 10　　B. 11　　C. 12　　D. 13

【答案】C

【分析 1】因为恰好是 2011 个叶节点，所以总节点个数最少要有 4021 个，深度最少，那就是每一层都要排满。既然排满那么前 i 的节点总数就是 2^i-1，又因为 $2^{11}-1<4021$ 且 $2^{12}-1>4021$ 所以是 12 层。

【分析 2】如果这棵二叉树是完全二叉树，那么深度应该是最少的。必须是 12 层的完全二叉树的叶子结点才能达到 2011 个。

4.【NOIP2010】如果树根算是第一层，那么一颗 n 层的二叉树最多有(　　)个结点。

A. 2^n-1　　B. 2^n　　C. 2^n+1　　D. 2^{n+1}

【答案】A

【分析】当该树为满二叉树时，这棵树拥有最多的结点，一颗 n 层二叉树有 2^n-1 结点。也可以用代入法。

5.【NOIP2009】一个包含 n 个分支结点（非叶结点）的非空二叉树，它的叶结点数目最多为（　　）。

A. 2n+1　　B. 2n−1　　C. n−1　　D. n+1

【答案】D

【分析 1】假设一颗树的边数为 v，叶结点个数为 x，则所有边分为两类：两端是分支结点（n−1 条）或有一端是叶子结点（x 条），x+n−1=v，v<=2 * n，所以叶结点数量最多为 n+1。

【分析 2】题目要求叶结点数目最多，那么可以知道分支结点的度为 2，假设叶结点数为 x，结点数＝边数＋1；x+n=2 * n+1，x=n+1

【分析 3】题中 $n=n_1+n_2$，因为 $n_0=n_2+1$，所以 $n_0=n-n_1+1<=n+1$

6.【NOIP2009】最优前缀编码，也称 Huffman 编码。这种编码组合的特点是对于较频繁使用的元素给与较短的唯一编码，以提高通讯的效率。下面编码组合哪一组不是合法的前缀编码（　　）。

A. (00,01,10,11)　　B. (0,1,00,11)

C. (0,10,110,111)　　D. (1,01,000,001)

【答案】B

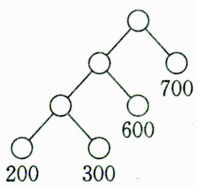
【分析】Huffman 编码不允许一个编码是另一个的前缀。B 选项，当给 00 进行编码时，会出现编码不唯一的情况，可以由第三个 00 进行编码，也可以由 2 个第一个 0 进行编码。

7.【NOIP2011】现有一段文言文，要通过二进制哈夫曼编码进行压缩。简单起见，假设这段文言文只由 4 个汉字“之”“乎”“者”“也”组成，它们出现的次数分别为 700、600、300、200。那么，“也”字的编码长度是（　　）。

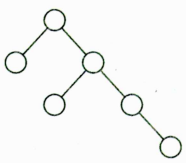
A. 1　　B. 2　　C. 3　　D. 4

【答案】C

【分析】哈夫曼编码采取贪心思想，它实际上是构建了一棵二叉树，每次取出权重最小的两个点合并成新的点，新点权重为原来权重的和。最后这个点编码长度就是所在深度−1。这题最后的二叉树形态如下图所示，此时 200 对应的点深度为 4，其编码长度为 3。



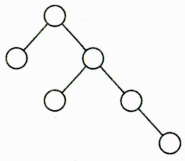
8.【NOIP2016】一棵二叉树如右图所示，若采用顺序存储结构，即用一维数组元素存储该二叉树中的结点（根结点的下标为 1，若某结点的下标为 i，则其左孩子位于下标 2i 处、右孩子位于下标（2i+1）处），则图中所有结点的最大下标为（　　）。



A. 6　　B. 10　　C. 12　　D. 15

【答案】D

【分析】下标最大的结点为最下最右的结点。$2^4-1=15$，顺序存储会浪费空间，适合存储完全二叉树结构。各层结点的编号为（自左到右），第一层：1；第二层：2，3；第三层：6，7；第四层：15。



9.【NOIP2016】一棵二叉树如右图所示，若采用二叉树链表存储该二叉树（各个结点包括结点的数据、左孩子指针、右孩子指针）。如果没有左孩子或者右孩子，则对应的为空指针。那么该链表中空指针的数目为（　　）。

A. 6　　B. 7　　C. 12　　D. 14

【答案】B

【分析】每个叶子结点有 2 个空指针，只有一个孩子的结点有 1 个空指针，统计计算即可。

10.【NOIP2010】完全二叉树的顺序存储方案，是指将完全二叉树的结点从上到下、从左到右依次存放到一个顺序结构的数组中。假定根结点存放在数组的 1 号位置上，则第 k 号结点的父结点如果存在的话，应当存放在数组中的（　　）号位置。

A. 2k　　B. 2k+1　　C. k/2 下取整　　D. (k+1)/2

【答案】C

【分析】其实就是父结点的编号是多少，稍微寻找一下规律便知道，当 k 是偶数，父结点是 $\frac{k}{2}$，当 k 是奇数，父结点是 $\frac{k-1}{2}$，综合一下就是 $\left\lfloor\frac{k}{2}\right\rfloor$。

1	2	3	4	5	6	7	
第一层根结点	第二层 1 号结点的左儿子	第二层 1 号结点的右儿子	第三层 2 号结点的左儿子	第三层 2 号结点的右儿子	第三层 3 号结点的左儿子	第三层 3 号结点的右儿子	…

11.【NOIP2008】完全二叉树共有 2 * N−1 个结点，则它的叶结点数是（　　）。

A. N−1　　B. N　　C. 2 * N　　D. 2^N-1

【答案】B

【分析 1】二叉树中度为 2 的结点数比度为 0 的结点数少 1，两者数量之和为奇数；完全二叉树中度为 1 的结点数或 0 或 1；此完全二叉树的结点总数为 2 * N−1，该数为奇数，说明度为 1 的结点数为 0；因此度为 2 的结点数是 N−1，度为 0（叶结点）的结点数为 N。

【分析 2】代入法，N=1 时，叶子结点个数为 1；N=2 时，叶子结点个数为 2。



12.【NOIP2009】一个包含 n 个分支结点（非叶结点）的非空满 k 叉树，k>=1，它的叶结点数目为（　　）。

A. nk+1　　B. nk−1　　C. (k+1)n−1　　D. (k−1)n+1

【答案】D

【分析 1】m 层满 k 叉树有 $\frac{1-k^m}{1-k}$ 个结点，非叶结点有 $\frac{1-k^{m-1}}{1-k}$ 个，叶结点有 k^{m-1} 个。$\frac{1-k^m}{1-k}-\frac{1-k^{m-1}}{1-k}=k^{m-1}$

【分析 2】结点数＝边数＋1，假设叶结点数为 x，可以得到 x＋n＝k＊n＋1，x＝(k－1)＊n＋1。

13.【NOIP2015】如果根的高度为 1，具有 61 个结点的完全二叉树的高度为(　　)。

A. 5　　B. 6　　C. 7　　D. 8

【答案】B

【分析】如果根的高度为 1，那么高度为 i 的满二叉树的结点数是 2^i-1，因为题中是完全二叉树，且 $2^5-1<61<2^6-1$，所以层数为 6。

14.【NOIP2014】一棵具有 5 层的满二叉树中结点数为(　　)。

A. 31　　B. 32　　C. 33　　D. 16

【答案】A

【分析】n 层的满二叉树中结点数为 2^n-1。

15.【NOIP2018】根结点深度为 0，一棵深度为 h 的满 k(k＞1)叉树，即除最后一层无任何子结点外，每一层上的所有结点都有 k 个子结点的树，共有(　　)个结点。

A. $(k^{h+1}-1)/(k-1)$　　B. k^{h-1}

C. k^h　　D. $(k^{h-1})/(k-1)$

【答案】A

【分析 1】考查数据结构基本知识，对于 k 叉树，每一层的结点个数依次为 $1,k,k^2,k^3,\cdots,k^h$。由等比数列求和公式可知，答案为 $\frac{k^{h+1}-1}{k-1}$。

【分析 2】代入法，本题可以通过画二叉树 k＝2，三叉树 k＝3 来排除选项。

16.【NOIP2013】二叉树的(　　)第一个访问的结点是根结点。

A. 先序遍历　　B. 中序遍历　　C. 后序遍历　　D. 以上都是

【答案】A

【分析】二叉树先序遍历第一个访问的结点是根结点。

17.【NOIP2013】二叉查找树具有如下性质：每个结点的值都大于其左子树上所有结点的值、小于其右子树上所有结点的值。那么，二叉查找树的(　　)是一个有序序列。

A. 先序遍历　　B. 中序遍历　　C. 后序遍历　　D. 宽度优先遍历

【答案】B

【分析】二叉树每个结点的值，大于其所有左子结点、小于其所有右子结点的值，因此按照中序遍历(左子结点、根、右子结点)，得到的一定是一个有序遍历。

18. 前序遍历序列与中序遍历序列相同的二叉树为(　　)。

A. 根结点无左子树的二叉树

B. 根结点无右子树的二叉树

C. 只有根结点的二叉树或非叶子结点只有左子树的二叉树

D. 只有根结点的二叉树或非叶子结点只有右子树的二叉树

【答案】D

【分析】前序遍历的顺序依次为“根结点-左子树-右子树”，中序遍历的顺序依次为“左子树-根结点-右子树”。所以当二叉树只有根结点或非叶子结点只有右子树时，前序遍历和中序遍历的顺序都退化成了：根、右子树，故二者得到的序列相同。要使中序遍历序列与前序遍历序列相同，每个非叶子结点都不能有左子树。

19.【NOIP2015】前序遍历序列与后序遍历序列相同的二叉树为(　　)。

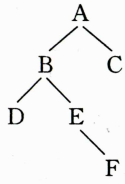
A. 非叶子结点只有左子树的二叉树　　B. 只有根结点的二叉树

C. 根结点无右子树的二叉树　　D. 非叶子结点只有右子树的二叉树

【答案】B

【分析】前序遍历的顺序依次为“根结点—左子树—右子树”，后序遍历的顺序依次为“左子树—右子树—根结点”。由于前序遍历序列中的根结点一定在第一个，要使后序遍历序列与之相同，左、右子树必须为空。

20.【NOIP2011】右图是一棵二叉树，它的先序遍历是(　　)。



A. ABDEFC　　B. DBEFAC

C. DFEBCA　　D. ABCDEF

【答案】A

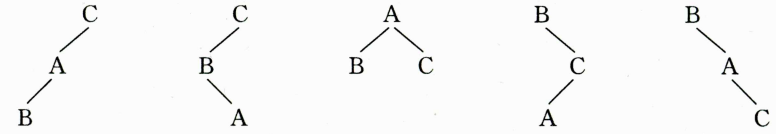
【分析】先序遍历的顺序是根左右。

21.【NOIP2012】如果一棵二叉树的中序遍历是 BAC，那么它的先序遍历不可能是(　　)。

A. ABC　　B. CBA　　C. ACB　　D. BAC

【答案】C

【分析】所有的情况如下图所示：

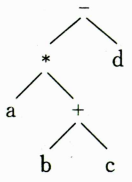


22.【NOIP2009】【NOIP2016】表达式 a*(b+c)-d 的后缀表达式是(　　)。

A. abcd*+-　　B. abc+*d-　　C. abc*+d-　　D. -+*abcd

【答案】B

【分析】后缀表达式的主要特点是左右根，根结点表示符号，左右两边是式子或数字。该树如下：

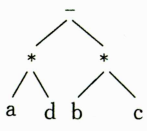


23.【NOIP2018】表达式 a*d-b*c 的前缀形式是(　　)。

A. ad*bc*-　　B. -*ad*bc　　C. a*d-b*c　　D. -**adbc

【答案】B

【分析】本题考查前缀表达式的理解。



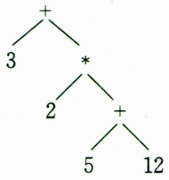
前缀表达式为 -*ad*bc。

24.【NOIP2010】前缀表达式+3 * 2+5 12 的值是(　　)。

A. 23　　B. 25　　C. 37　　D. 65

【答案】C

【分析】运算符号只能作为父结点,而数字是叶结点。



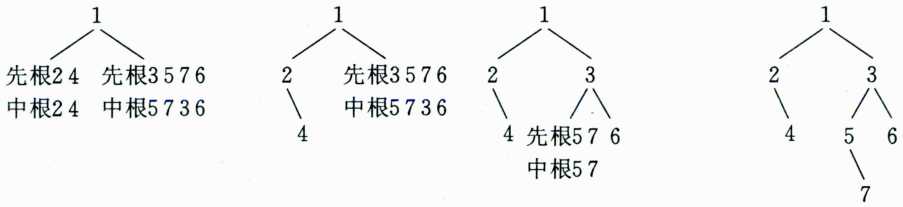
中缀表达式是 3+2 * (5+12)=37。

25.【NOIP2008】二叉树 T,已知其先根遍历是 1 2 4 3 5 7 6(数字为结点的编号,以下同),中根遍历是 2 4 1 5 7 3 6,则该二叉树的后根遍历是(　　)。

A. 4 2 5 7 6 3 1　　B. 4 2 7 5 6 3 1　　C. 7 4 2 5 6 3 1　　D. 4 2 7 6 5 3 1

【答案】B

【分析】二叉树的遍历,先构造二叉树,根据先根遍历序列可知根为 1,因此在中根遍历序列中 1 的左边都是左子树,1 的右边都是右子树,以此类推。



26.【NOIP2010】一棵二叉树的前序遍历序列是 A B C D E F G,后序遍历序列是 C B F E G D A,则根结点的左子树的结点个数可能是(　　)。

A. 2　　B. 3　　C. 4　　D. 5

【答案】A

【分析】根据前序与后序遍历,可知根结点为 A;

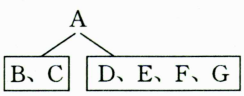
假设结点的左子树的结点个数只有 1 个,后序遍历肯定是 B 开始,与题目矛盾,所以不可能;

假设结点的左子树的结点个数只有 2 个,包括 B、C,先遍历完 B、C,再遍历 D、E、F、G 是可能的;

假设结点的左子树的结点个数只有 3 个,包括 B、C、D,先遍历完 B、C、D,再遍历 E、F、G,在后序变量序列中 D 不可能在 E、F、G 后面遍历;

同理结点的左子树的结点个数 4 和 5 也是不可能的,结点 D 必定在右子树。

我们发现 B、C 整体在 D、E、F、G 的左边。



不定项选择题

1.【NOIP2008】设 T 是一棵有 n 个顶点的树，下列说法正确的是(　　)。

A. T 是连通的、无环的　　B. T 是连通的，有 n−1 条边

C. T 是无环的，有 n−1 条边　　D. 以上都不对

【答案】ABC

【分析】n 个顶点的树有且仅有 n−1 条边，而且连通无环，这些都是树的性质。

2.【NOIP2018】下列说法中，是树的性质的有(　　)。

A. 无环　　B. 任意两个结点之间有且只有一条简单路径

C. 有且只有一个简单环　　D. 边的数目恰是顶点数目减 1

【答案】ABD

【分析】树是一个 n 个结点 n−1 条边的连通图，故不可能存在环，其余选项均为树的一些常见基本性质。

3.【NOIP2015】下列有关树的叙述中，叙述正确的有(　　)。

A. 在含有 n 个结点的树中，边数只能是(n−1)条

B. 在哈夫曼树中，叶结点的个数比非叶结点个数多 1

C. 完全二叉树一定是满二叉树

D. 在二叉树的前序序列中，若结点 u 在结点 v 之前，则 u 一定是 v 的祖先

【答案】AB

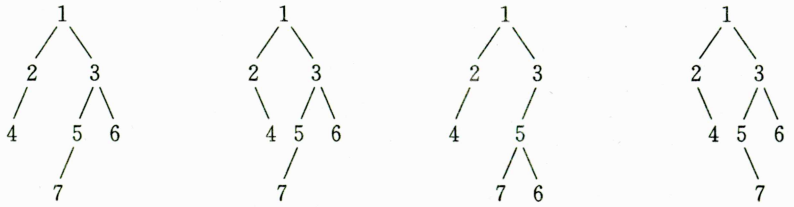
【分析】排除法，C 选项，完全二叉树不一定是满二叉树，满二叉树一定是完全二叉树；D 选项，在二叉树的前序序列中，若结点 u 在结点 v 之前，则 u 不一定是 v 的祖先，也有可能 u 和 v 分属两棵不同的子树，而 v 的祖先一定在 v 之前。

4.【NOIP2008】二叉树 T，已知其先根遍历是 1 2 4 3 5 7 6(数字为结点的编号，以下同)，后根遍历是 4 2 7 5 6 3 1，则该二叉树的可能的中根遍历是(　　)。

A. 4 2 1 7 5 3 6　　B. 2 4 1 7 5 3 6　　C. 4 2 1 7 5 6 3　　D. 2 4 1 5 7 3 6

【答案】ABD

【分析】可以由先根遍历序列和 A、B、C、D 中的任一中根遍历序列组成一棵确定的树，如果能确定，则写出其后根遍历的序列，和题中叙述的后根序列比对即可。



5.【NOIP2011】如果根结点的深度记为 1，则一棵恰有 2011 个叶子结点的二叉树的深度可能是(　　)。

A. 10　　B. 11　　C. 12　　D. 2011

【答案】CD

【分析】结点数为 n 的二叉树，最小深度为完全二叉树深度为 $\lfloor \log_2 n \rfloor + 1$ // 下取正整后 +1

最大深度为 n，退化为单链长度。

所以可能的树深度范围：$\lfloor \log_2 n \rfloor + 1 \sim n$，题目中 n 为 2011。高度范围是 12～2011。故选择 CD。

6.【NOIP2012】一棵二叉树一共有 19 个结点，其叶子结点可能有(　　)个。

A. 1　　B. 9　　C. 10　　D. 11

【答案】ABC

【分析】一棵任意形态的 n 个结点的二叉树的叶子数范围是 $1 \sim \left\lfloor \frac{n+1}{2} \right\rfloor$。1 指的是一条链的情况，$\left\lfloor \frac{n+1}{2} \right\rfloor$ 指的是二叉哈夫曼树的情况。

7.【NOIP2018】2－3 树是一种特殊的树，它满足两个条件：

(1)每个内部结点有两个或三个子结点；

(2)所有的叶结点到根的路径长度相同。

如果一棵 2－3 树有 10 个叶结点，那么它可能有(　　)个非叶结点。

A. 5　　B. 6　　C. 7　　D. 8

【答案】CD

【分析】最底层有 10 个结点；倒数第二层只有两种可能性：第一种情况，5 个结点，每个结点的度都为 2。第二种情况，4 个结点，两个结点的度为 3，两个结点的度为 2；

根据倒数第二层的结点数，倒数第三层和第四层的结点数是唯一确定的，上往下每层结点分别为(1,2,5,10)和(1,2,4,10)。

第9节　图

图(Graph)是一种复杂的非线性结构。在人工智能、工程、数学、物理、化学、生物和计算机科学等领域中,图结构有着广泛的应用。

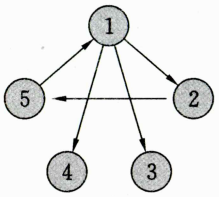
一、图的定义

点用边连起来就叫作图,严格意义上讲,图是一种数据结构,定义为 graph=(V,E)。V 是一个非空有限集合,代表顶点(结点),E 代表边的集合。

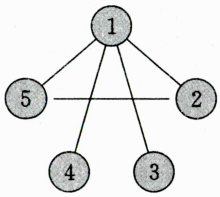
二、图的相关概念

1.有向图:图的边有方向,只能按箭头方向从一点到另一点,(a)就是一个有向图。

2.无向图:图的边没有方向,可以双向,(b)就是一个无向图。



(a)



(b)

3.结点的度:无向图中与结点相连的边的数目,称为结点的度。

4.结点的入度:在有向图中,以这个结点为终点的有向边的数目。

5.结点的出度:在有向图中,以这个结点为起点的有向边的数目。

6.权值:边的“费用”,可以形象地理解为边的长度。

7.连通:如果图中结点 U,V 之间存在一条从 U 通过若干条边、点到达 V 的通路,则称 U、V 是连通的。

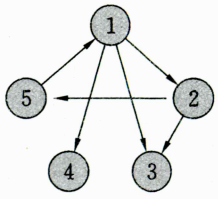
8.回路:起点和终点相同的路径,称为回路,或“环”。

9.完全图:一个 n 阶的完全无向图含有 n*(n−1)/2 条边,一个 n 阶的完全有向图含有 n*(n−1)条边。

10.稠密图:一个边数接近完全图的图。

11.稀疏图:一个边数远远少于完全图的图。

12.强连通分量:有向图中任意两点都连通的最大子图。右图中,1−2−5 构成一个强连通分量。特殊地,单个点也算一个强连通分量,所以右图有三个强连通分量:1−2−5,4,3。



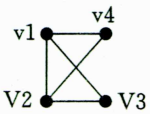
三、图的存储结构

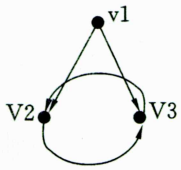
1.二维数组邻接矩阵存储

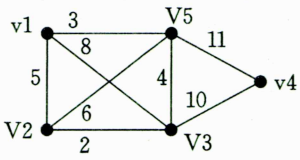
图的邻接矩阵存储方式是用一个二维数组(称邻接矩阵)存储图中的边或弧的信息。它适用于稠密图。

定义 int G[101][101],其中 G [i][j]表示从点 i 到点 j 的边的权值,定义如下:

$$G[i][j]=\begin{cases}1\text{ 或权值} & \text{当 } v_i \text{ 与 } v_j \text{ 之间有边或弧时,取值为 1 或权值}\\ 0\text{ 或}\infty & \text{当 } v_i \text{ 与 } v_j \text{ 之间无边或弧时,取值为 0 或}\infty\text{(无穷大)}\end{cases}$$





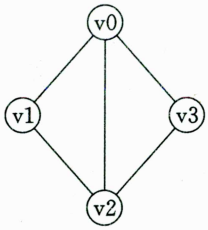


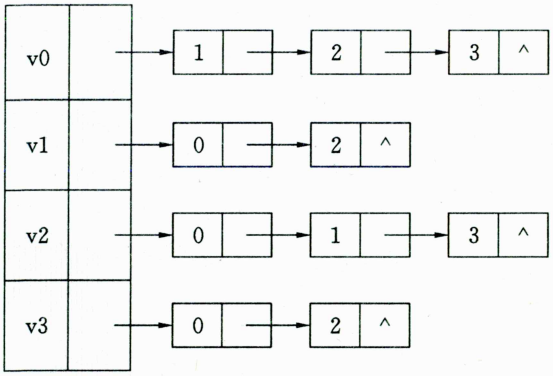
上图中的3个图对应的邻接矩阵分别如下：

$$G(A)=\begin{bmatrix}0&1&1&1\\1&0&1&1\\1&1&0&0\\1&1&0&0\end{bmatrix}\quad G(B)=\begin{bmatrix}0&1&1\\0&0&1\\0&1&0\end{bmatrix}\quad G(C)=\begin{bmatrix}\infty&5&8&\infty&3\\5&\infty&2&\infty&6\\8&2&\infty&10&4\\\infty&\infty&10&\infty&11\\3&6&4&11&\infty\end{bmatrix}$$

2.邻接表存储结构

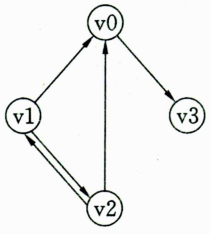
邻接表是一种将数组与链表相结合的存储方法，其具体实现为：将图中顶点用一个一维数组存储，每个顶点 Vi 的所有邻接点用一个单链表来存储，链表中存放与当前结点相邻的结点在数组中的下标，适用于稀疏图。

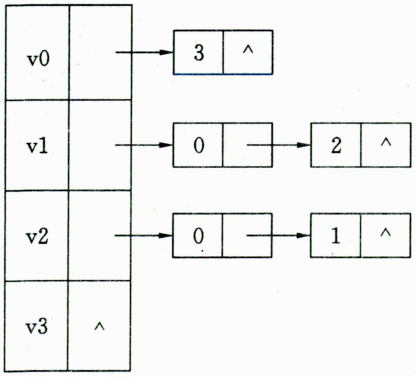




无向图第i个链表的边结点数目正好是第i个顶点的度。

对于具有n个结点、e条边的无向图，邻接表中数组有n个顶点结点、链表有2e个边结点。





有向图第i个链表的边结点数目正好是第i个顶点的出度。

对于具有n个结点、e条边的有向图，邻接表中数组有n个顶点结点、链表有e个边结点。

四、深度优先与广度优先遍历

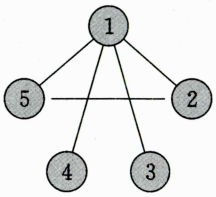
从图中某一顶点出发系统地访问图中所有顶点，使每个顶点恰好被访问一次，这种运算操作被称为图的遍历。为了避免重复访问某个顶点，可以设一个标志数组 visited [i]，未访

问时值为 false，访问一次后改为 true。图的遍历分为深度优先遍历和广度优先遍历两种方法，两者的时间效率都是 O(n * n)。

1.深度优先遍历

深度优先遍历与深度搜索算法相似，从一个点 A 出发，将这个点标为已访问 visited [i] =true，然后再访问所有与之相连且未被访问过的点。当 A 的所有邻接点都被访问过后，退回到 A 的上一个点(假设是 B)，再从 B 的另一个未被访问的邻接点出发，继续遍历。

例如对右边的这个无向图深度优先遍历，假定先从 1 出发，程序按如下顺序遍历：



(a)1→2→5，然后退回到 2，退回到 1。

(b)从 1 开始再访问未被访问过的点 3，3 没有未访问的邻接点，退回 1。

(c)再从 1 开始访问未被访问过的点 4，再退回 1。

(d)起点 1 的所有邻接点都已访问，遍历结束。

2.广度优先遍历

广度优先遍历并不常用，从编程复杂度的角度考虑，通常采用的是深度优先遍历。

广度优先遍历和广度优先搜索相似，因此使用广度优先遍历一张图并不需要掌握新的知识，在原有的广度优先搜索的基础上，做一些小的修改，就变成广度优先遍历算法。

五、一笔画问题

如果一个图存在一笔画，则一笔画的路径叫作欧拉路，如果最后又回到起点，那这个路径叫作欧拉回路。

定义奇点是指跟这个点相连的边数目有奇数个的点。对于能够一笔画的图，有以下两个定理。

定理 1：存在欧拉路的条件，图是连通的，有且只有 2 个奇点。

定理 2：存在欧拉回路的条件，图是连通的，有 0 个奇点。

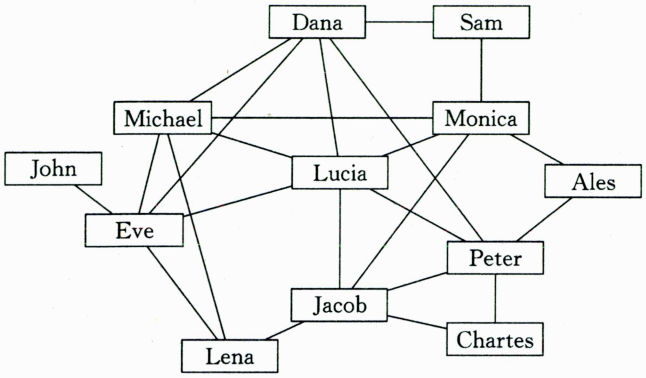
两个定理的正确性是显而易见的，既然每条边都要经过一次，那么对于欧拉路，除了起点和终点外，每个点如果进入了一次，一定要出去一次，显然是偶点。对于欧拉回路，每个点进入和出去次数一定都是相等的，显然没有奇点。

求欧拉路的算法很简单，使用深度优先遍历即可。

根据一笔画的两个定理，如果寻找欧拉回路，对任意一个点执行深度优先遍历；找欧拉路，则对一个奇点执行深度优先搜索，时间复杂度为 O(m+n)，m 为边数，n 是点数。

课堂练习

1.【NOIP2016】Lucia 和她的朋友以及朋友的朋友都在某社交网站上注册了账号。下图是他们之间的关系图，两个人之间有边相连代表这两个人是朋友，没有边相连代表不是朋友。这个社交网站的规则为如果某人 A 向他(她)的朋友 B 分享了某张照片，那么 B 就可以对该照片进行评论；如果 B 评论了该照片，那么他(她)的所有朋友都可以看见这个评论以及被评论的照片，但是不能对该照片进行评论(除非 A 也向他(她)分享了该照片)。现在 Lucia 已经上传了一张照片，但是她不想让 Jacob 看见这张照片，那么她可以向以下朋友(　　)分享该照片。



A. Dana，Michael，Eve　　B. Dana，Eve，Monica

C. Michael，Eve，Jacob　　D. Micheal，Peter，Monica

【答案】A

【分析】为使 Jacob 看不到照片，Lucia 不可以向 Jacob 及他们共同的朋友 Monica 和 Peter 分享照片，就是找图中没有与 Jacob 连线的人。

2.【NOIP2014】有向图中每个顶点的度等于该顶点的(　　)。

A. 入度　　B. 出度　　C. 入度与出度之和　D. 入度与出度之差

【答案】C

【分析】有向图中每个顶点的度分成两种，分别是入度和出度，所有的入度之和与所有的出度之和相等，对于一个顶点来说，该顶点的度等于该顶点的入度与出度之和。

3.【NOIP2014】在无向图中，所有顶点的度数之和是边数的(　　)倍。

A. 0.5　　B. 1　　C. 2　　D. 4

【答案】C

【分析】图论基础题，每条边都会使两个端点的度增加 1，所以度数之和为点数两倍。

4.【NOIP2016】设简单无向图 G 有 16 条边且每个顶点的度数都是 2，则图 G 有(　　)个顶点。

A. 10　　B. 12　　C. 8　　D. 16

【答案】D

【分析】考察图论基本知识。简单无向图中两个顶点之间最多有一条边，一个有 16 条边并且每个点都有两条边相连接(每个顶点的度数都是 2)，那么它应该是一个 16 个顶点首尾相

接组成的环。

5.【NOIP2010】关于拓扑排序，下面说法正确的是(　　)。

A. 所有连通的有向图都可以实现拓扑排序

B. 对一个图而言，拓扑排序的结果是唯一的

C. 拓扑排序中入度为 0 的结点总会排在入度大于 0 的结点的前面

D. 拓扑排序结果序列中的第一个结点一定是入度为 0 的点

【答案】D

【分析】A 选项，当图中有环，必然无法拓扑排序；B 选项，同一时刻，入度为 0 的点不唯一，拓扑排序也就不唯一；C 选项，如下 1－3－2 也是正确的拓扑排序，但 3 却在 2 之前。

$$1 \rightarrow 3$$
$$2$$

6.【NOIP2008】设 T 是一棵有 n 个顶点的树，下列说法不正确的是(　　)。

A. T 有 n 条边　　B. T 是连通的

C. T 是无环的　　D. T 有 n－1 条边

【答案】A

【分析】n 个顶点的树有且仅有 n－1 条边，而且连通无环，这些都是树的性质。

7.【NOIP2017】设 G 是有 n 个结点、m 条边(n≤m)的连通图，必须删去 G 的(　　)条边，才能使得 G 变成一棵树。

A. m－n＋1　　B. m－n　　C. m＋n＋1　　D. n－m＋1

【答案】A

【分析】一棵树有 n－1 条边，故答案为 m－(n－1)＝m－n＋1。

8.【NOIP2015】6 个顶点的连通图的最小生成树，其边数为(　　)。

A. 6　　B. 5　　C. 7　　D. 4

【答案】B

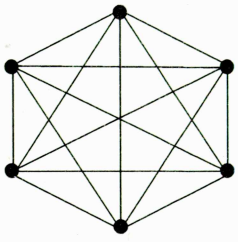
【分析】n 个顶点的连通图的生成树的边数为 n－1。

9.【NOIP2014】设 G 是有 6 个结点的完全图，要得到一棵生成树，需要从 G 中删去(　　)条边。

A. 6　　B. 9　　C. 10　　D. 15

【答案】C

【分析】完全图如下图所示，6 个点的完全图有 $\frac{6*5}{2}=15$ 条边，而 6 个点的树有 6－1＝5 条边，因此需要删去 15－5＝10 条边。



10.【NOIP2009】已知 n 个顶点的有向图，若该图是强连通的(从所有顶点都存在路径到达其他顶点)，则该图中最少有(　　)条有向边。

A. n　　B. n+1　　C. n−1　　D. n*(n−1)

【答案】A

【分析】n 个点首尾相接，构成环。

11.【NOIP2011】无向完全图是图中每对顶点之间都恰有一条边的简单图。已知无向完全图 G 有 7 个顶点，则它共有(　　)条边。

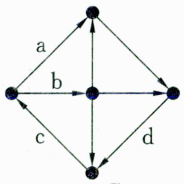
A. 7　　B. 21　　C. 42　　D. 49

【答案】B

【分析 1】n 个点的完全图有 $1+2+\cdots+(n-1)=\frac{(n-1)*n}{2}$ 条边。当 n=7 时，带入公式得 21。

【分析 2】根据无向完全图的概念，可以知道每对顶点之间都恰有一条边，那么边的个数就是 C(N,2)。当 n=7 时，C(7,2)=21。

12.【NOIP2011】对一个有向图而言，如果每个结点都存在到达其他任何结点的路径，那么就称它是强连通的。例如，下图就是一个强连通图。事实上，在删掉边(　　)后，它依然是强连通的。

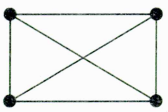


A. a　　B. b　　C. c　　D. d

【答案】A

【分析】强联通是指在有向图中任意两点都可以互相到达，枚举四个选项根据定义判断一下即可。去掉边 b 后，左边的点到不了中间点；去掉边 c 后，下面的点到不了左边的点；去掉边 d 后，右边的点到不了下面的点。

13.【NOIP2013】在一个无向图中，如果任意两点之间都存在路径相连，则称其为连通图。下图是一个有 4 个顶点、6 条边的连通图。若要使它不再是连通图，至少要删去其中的(　　)条边。

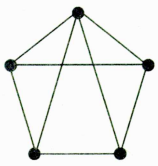


A. 1　　B. 2　　C. 3　　D. 4

【答案】C

【分析】通过观察容易发现，只要切断 3 条边，就能孤立一个结点。同时不存在切断两条边就能使图不连通的情况，因此答案是 3。

14.【NOIP2013】在一个无向图中，如果任意两点之间都存在路径相连，则称其为连通图。右图是一个有 5 个顶点、8 条边的连通图。若要使它不再是连通图，至少要删去其中的(　　)条边。



A. 2　　B. 3　　C. 4　　D. 5

【答案】B

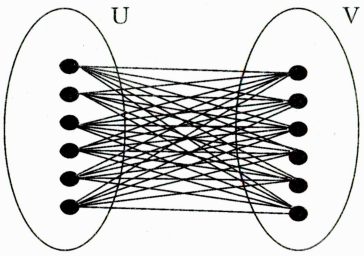
【分析】通过观察容易发现，只要切断 3 条边，就能孤立一个结点。同时不存在切断两条边就能使图不连通的情况，因此答案是 3。

15.【NOIP2013】二分图是指能将顶点划分成两个部分，每一部分内的顶点间没有边相连的简单无向图。那么，12 个顶点的二分图至多有(　　)条边。

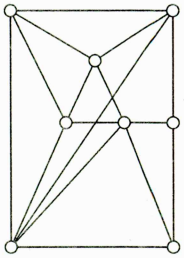
A. 18　　B. 24　　C. 36　　D. 66

【答案】C

【分析】二分图如下图所示，12 个结点的二分图，假设一边有 x 个结点，另一边则有(12－x)个结点，因此最多有 x(12－x)条边，当 x 取 6 时得到最大值为 36。



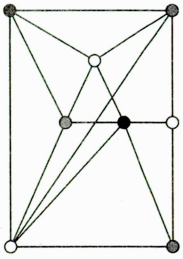
16.【NOIP2015】对图 G 中各个结点分别指定一种颜色，使相邻结点颜色不同，则称为图 G 的一个正常着色。正常着色图 G 所必需的最少颜色数，称为 G 的色数。那么下图的色数是(　　)。



A. 3　　B. 4　　C. 5　　D. 6

【答案】A

【分析】手动染色。选取度数最多的点，开始染色，直接模拟，最终只要 3 种颜色即可。



17.【NOIP2016】G 是一个非连通简单无向图，共有 28 条边，则该图至少有(　　)个顶点。

A. 10　　B. 9　　C. 8　　D. 7

【答案】B

【分析】8个顶点的简单无向图只有在为完全图的情况下才会有$\frac{8*7}{2}=28$条边，但完全图不符合“非连通”的要求。存在9个顶点28条边的非连通简单无向图，例如1个孤立点和一个8个顶点的完全图。

18.【NOIP2017】由四个不同的点构成的简单无向连通图的个数是(　　)。

A. 32　　B. 35　　C. 38　　D. 41

【答案】C

【分析】4个不同点构成简单无向连通图，最多有C(4,2)=4*(4−1)/2=6条边，最少有4−1=3条边(树)，但不是所有的任选3条边都满足条件，有一种情况是三个点形成一个三角形而孤立一个点，这种情况共有4种，比如A、B、C、D这4个点三条边分别连接A−B、B−C、C−A，而D孤立着，这样就不行，所以总个数为$C_4^2=6$，$C_6^3-4+C_6^4+C_6^5+C_6^6=38$。

19.【NOIP2018】由四个没有区别的点构成的简单无向连通图的个数是(　　)。

A. 6　　B. 7　　C. 8　　D. 9

【答案】A

【分析】考察图论基础知识。连通图指的是两个点之间都有路径可以到达，简单图是既不含平行边也不含自环的图，无向图指的是图中的边是无向边。

枚举可得有6种由4个没有区别的点构成不同结构的简单无向连通图。

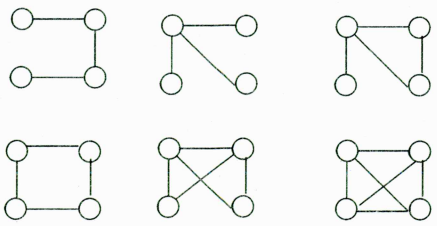
三条边的图有两个(长为3的路、星图)；

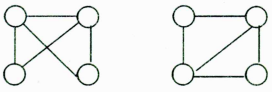
四条边的图有两个(圈、三角形加一条边)；

五条边的图有一个(完全图挖掉一条边)。

六条边的图有一个(即4个点的完全图)。

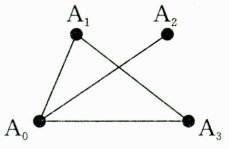
综上，一共有6种。





这两个是等价的

20.【NOIP2013】以A_0作为起点，对下面的无向图进行深度优先遍历时，遍历顺序不可能是(　　)。



A. A_0,A_1,A_2,A_3　　B. A_0,A_1,A_3,A_2

C. A_0,A_2,A_1,A_3　　D. A_0,A_3,A_1,A_2

【答案】A

【分析】从题中 A_0 出发深度优先遍历，我们分析选项 A，从 A_0 到 A_1，从 A_1 可以到 A_3，不能回溯 A_0 而到达 A_2，因此是不可能的。有 4 种所有可能的路径为（A_0，A_1，A_3，A_2），（A_0，A_3，A_1，A_2），（A_0，A_2，A_1，A_3），（A_0，A_2，A_3，A_1）。

21.【NOIP2015】具有 n 个顶点，e 条边的图采用邻接表存储结构，进行深度优先遍历和广度优先遍历运算的时间复杂度均为（　　）。

A. $O(n^2)$　　B. $O(e^2)$　　C. $O(ne)$　　D. $O(n+e)$

【答案】D

【分析】用邻接表存储时，遍历整张图的复杂度是 $O(n+e)$。

22.【NOIP2013】对一个 n 个顶点、m 条边的带权有向简单图用 Dijkstra 算法计算单源最短路时，如果不使用堆或其他优先队列进行优化，则其时间复杂度为（　　）。

A. $O(mn+n^3)$　　B. $O(n^2)$　　C. $O((m+n)\log n)$　　D. $O((m+n^2)\log n)$

【答案】B

【分析】Dijkstra 算法，在不使用堆或者其他优先队列的情况下，每次寻找当前距离最小的点需要 O(n)，而从确定的点开始更新距离次数可以小于等于 n（使用邻接矩阵或者其他数据结构），因此总的时间复杂度为 $O(n^2)$。应用堆优化的迪杰斯特拉算法复杂度为 $O((m+n)\log n)$。

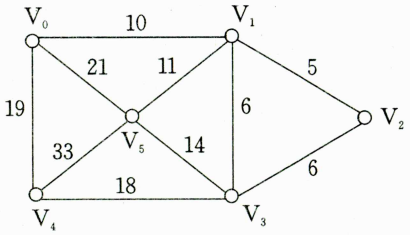
23.【NOIP2009】右图给出了一个加权无向图，从顶点 V_0 开始用 Prim 算法求最小生成树，则依次加入最小生成树的顶点集合的顶点序列为（　　）。

A. V0，V1，V2，V3，V5，V4

B. V0，V1，V5，V4，V3，V3

C. V1，V2，V3，V0，V5，V4

D. V1，V2，V3，V0，V4，V5



【答案】A

【分析】Prim 算法指的是每次找的是从未选定的集合中与当前已选定的点的集合里连边距离最短的结点进行扩展，因为从 V0 开始扩展。

已选定的点的集合	未选定的集合	连边距离最短
（V0）	（V1，V4，V5）	V0V1 连边距离为 10
（V0，V1）	（V2，V3，V4，V5）	V1V2 连边距离为 5
（V0，V1，V2）	（V3，V4，V5）	V2V3 连边距离为 6
（V0，V1，V2，V3）	（V4，V5）	V1V5 连边距离为 11
（V0，V1，V2，V3，V5）	（V4）	V3V4 连边距离为 18
（V0，V1，V2，V3，V5，V4）		

不定项选择题

1.【NOIP2014】以下哪些结构可以用来存储图(　　)。

A. 邻接矩阵　　B. 栈　　C. 邻接表　　D. 二叉树

【答案】AC

【分析】考查基本数据结构知识。邻接矩阵和邻接表是存储图的两种基本方式。

2.【NOIP2009】若3个顶点的无权图G的邻接矩阵用数组存储为{{0,1,1},{1,0,1},{0,1,0}},假定在具体存储中顶点依次为v_1,v_2,v_3,关于该图,下面的说法正确的是(　　)。

A. 该图是有向图

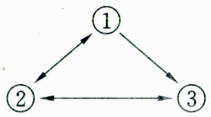
B. 该图是强连通的

C. 该图所有顶点的入度之和减所有顶点的出度之和等于1

D. 从v_1开始的深度优先遍历所经过的顶点序列与广度优先的顶点序列是相同的

【答案】ABD

【分析】画出图即可知,入度之和等于出度之和。



3.【NOIP2012】已知带权有向图G上的所有权值均为正整数,记顶点u到顶点v的最短路径的权值为d(u,v)。若v1,v2,v3,v4,v5是图G上的顶点,且它们之间两两都存在路径可达,则以下说法正确的有(　　)。

A. v1到v2的最短路径可能包含一个环

B. d(v1,v2)=d(v2,v1)

C. d(v1,v3)≤d(v1,v2)+d(v2,v3)

D. 如果v1→v2→v3→v4→v5是v1到v5的一条最短路径,那么v2→v3→v4是v2到v4的一条最短路径

【答案】CD

【分析】最短路径一定是简单路径,最短路径中不能出现环路,所以A不正确;因为是有向图,所以正向和反向的最短路径长度不一定相同,所以B不正确;C和D都符合最短路径的要求,其中,D可以考虑尝试反证法。

4.【NOIP2015】以下图中一定可以进行黑白染色的有(　　)。(黑白染色:为各个结点分别指定黑白两种颜色之一,且相邻结点颜色不同)

A. 二分图　　B. 完全图　　C. 树　　D. 连通图

【答案】AC

【分析】可以进行黑白染色的图要求不存在奇环,二分图、树一定不存在奇环。对二分图的两个部分分别用不同颜色染色;对树的奇数层和偶数层分别用不同颜色染色(树也是一个二分图,因为无奇环)。

5.【NOIP2018】下列关于最短路算法的说法,正确的是(　　)。

A. 当图中不存在负权回路但是存在负权边时,Dijkstra算法不一定能求出源点到所有

点的最短路

B. 当图中不存在负权边时，调用多次 Dijkstra 算法能求出每对顶点间最短路径

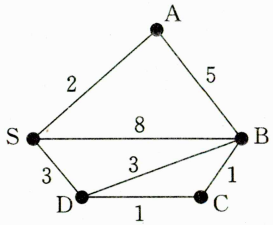
C. 图中存在负权回路时，调用一次 Dijkstra 算法也一定能求出源点到所有点的最短路

D. 当图中不存在负权边时，调用一次 Dijkstra 算法不能用于每对顶点间最短路计算

【答案】ABD

【分析】考察了一个最短路算法的知识点，Dijkstra 计算的是单源最短路，不支持负权边和负环。

6.【NOIP2011】对下图使用 Dijkstra 算法计算 S 点到其余各点的最短路径长度时，到 B 点的距离 d[B]初始时赋为 8，在算法的执行过程中还会出现的值有(　　)。



A. 3　　B. 7　　C. 6　　D. 5

【答案】BCD

【分析】此题考查图论中求单源最短路径的算法 Dijkstra 算法的掌握程度，理解 relax 松弛操作。Dijkstra 的思想是建个一维数组记录起点到其他各点的距离(不直接相连则为无限大)，然后找一个距离起点最近的点作为中间结点，更新起点到各个点的距离，然后把中心结点加个用过的标记，继续找下一个距离起点最近的点为中心结点，直到所有的点都当过中心结点就结束。这题中，第一个找到的中间结点是 A，这时把 d[B]更新为 7，然后找到的中心结点为 D，这时把 d[B]更新为 6，把 d[C]更新为 4；下一个找到的中心结点为 C，这时把 d[B]更新为 5，所以选 BCD。

第三章　数学问题

第 1 节　简单数论

在 20 世纪初之前，数论被称为算术，之后人们才开始使用数论的名称。数论主要研究整数的性质，正整数分为素数、合数和 1。素数产生了很多人们能理解，但是又难以证明的问题，比如孪生素数、哥德巴赫猜想等，在解决这些问题的过程中产生了很多新的数学思想和方法，推动了数学学科的发展。

算数基本定理（也叫唯一分解定理），即任意一个大于 1 的自然数都可分解成若干素数的乘积的形式就是数论的内容，那什么是素数，素数又有哪些性质，我们怎样得到素数呢？带着这些问题，让我们开始学习本节的内容。

一、整数性质

（一）（带余除法，即欧几里德除法）设 a，b 为整数，$b\neq 0$，则存在整数 q 和 r，使得 $a=bq+r$，其中 $0\leqslant r<|b|$，并且 q 和 r 由上述条件唯一确定；整数 q 被称为 a 被 b 除得的（不完全）商，数 r 称为 a 被 b 除得的余数。

注意：r 共有 $|b|$ 种可能的取值：$0,1,\cdots,|b|-1$。若 $r=0$，即为 a 被 b 整除的情形，因此，证明 $b|a$ 的基本手法是将 a 分解为 b 与一个整数之积。

带余除法的核心是关于余数 r 的不等式：$0\leqslant r<|b|$。

（二）任何一个正整数 n，都可以写成 $n=2^m l$ 的形式，其中 m 为非负整数，l 为奇数。

（三）若 $a\in Z$，$a>1$，则 a 的除 1 以外的最小正因数 q 是一个质（素）数. 如果 $q\neq a$，则 $q\leqslant\sqrt{a}$。

推论：如果不超过 $\sqrt{a}$ 的所有质数均不是 a 的约数，则 a 必为质数。

（四）算术基本定理（素数唯一分解定理）任何一个大于 1 的正整数 a，能唯一地表示成质（素）因数的乘积（不计较因数的排列顺序），即

任何大于 1 的整数 a 能唯一地写成下面

$$a=p_1^{\alpha_1}p_2^{\alpha_2}\cdots p_k^{\alpha_k}\qquad(1)$$

的形式，其中 p_i 为素数（$p_i<p_j(i<j)$），$a_i\in N_+$，$i=1,2,\cdots,k$。

上式叫作整数 a 的标准分解式。

推论 1：若 a 的标准分解式是（1）式，则 d 是 a 的正因数的充要条件是：

$$d=p_1^{\beta_1}p_2^{\beta_2}\cdots p_k^{\beta_k},0\leqslant\beta_i\leqslant a_i,i=1,2,\cdots,k。\qquad(2)$$

应注意（2）不能称为是 d 的标准分解式，其原因是其中的某些 β_i 可能取零值（当 d 不含某个素因数 p_i 时，$\beta_i=0$）。

推论 2：设 $a=bc$，且 $(b,c)=1$，若 a 是整数的 $k(k\geqslant 2)$ 次方，则 b,c 也是整数的 k 次方。特别地，若 a 是整数的平方，则 b,c 也是整数的平方。

一般地，设正整数 $a,b,\cdots,c$ 之积是一个正整数的 k 次方幂（$k\geqslant 2$），若 $a,b,\cdots,c$ 两两互素，则 $a,b,\cdots,c$ 都是正整数的 k 次方幂。

二、整除

（一）常见数字的整除判定方法

1.一个数的末位能被 2 或 5 整除，这个数就能被 2 或 5 整除；一个数的末两位能被 4 或 25 整除，这个数就能被 4 或 25 整除；一个数的末三位能被 8 或 125 整除，这个数就能被 8 或 125 整除。

2.一个数各位数字和能被 3 整除，这个数就能被 3 整除；一个数各位数字和能被 9 整除，这个数就能被 9 整除。

3.如果一个整数的奇数位上的数字之和与偶数位上的数字之和的差能被 11 整除，那么这个数能被 11 整除。

4.如果一个整数的末三位与末三位以前的数字组成的数之差能被 7、11 或 13 整除，那么这个数能被 7、11 或 13 整除。

5.如果一个数能被 99 整除，这个数从后两位开始两位一截所得的所有数（如果有偶数位，则拆出的数都是两位数；如果是奇数位，则拆出的数中有若干个两位数，还有一个是一位数）的和是 99 的倍数，这个数一定是 99 的倍数。

（二）整除的性质

1.性质 1　如果数 a 和数 b 都能被数 c 整除，那么它们的和或差也能被 c 整除。即如果 $c|a, c|b$，那么 $c|(a\pm b)$。

2.性质 2　如果数 a 能被数 b 整除，b 又能被数 c 整除，那么 a 也能被 c 整除。即如果 $b|a, c|b$，那么 $c|a$。

3.性质 3　如果数 a 能被数 b 与数 c 的积整除，那么 a 也能被 b 或 c 整除。即如果 $bc|a$，那么 $b|a$ 或 $c|a$。

4.性质 4　如果数 a 能被数 b 整除，也能被数 c 整除，且数 b 和数 c 互质，那么 a 一定能被 b 与 c 的乘积整除。即如果 $b|a, c|a$，且 $(b,c)=1$，那么 $bc|a$。

例如：如果 $3|12, 4|12$，且 $(3,4)=1$，那么 $(3\times4)|12$。

5.性质 5　如果数 a 能被数 b 整除，那么 am 也能被 bm 整除。如果 $b|a$，那么 $bm|am$（m 为非 0 整数）。

6.性质 6　如果数 a 能被数 b 整除，且数 c 能被数 d 整除，那么 ac 也能被 bd 整除。如果 $b|a$，且 $d|c$，那么 $bd|ac$。

三、余数

（一）余数三大余数定理

1.余数的加法定理。

a 与 b 的和除以 c 的余数，等于 a、b 分别除以 c 的余数之和。

例如：23、16 除以 5 的余数分别是 3 和 1，所以 $23+16=39$ 除以 5 的余数等于 4，即两个余数的和为 $(3+1)$。

当余数的和比除数大时，所求的余数等于余数之和再除以 c 的余数。

例如：23、19 除以 5 的余数分别是 3 和 4，所以 $23+19=42$ 除以 5 的余数等于 $3+4=7$ 除以 5 的余数，即为 2。

2.余数的减法定理。

a 与 b 的差除以 c 的余数，等于 a、b 分别除以 c 的余数之差。

例如:23、16 除以 5 的余数分别是 3 和 1,所以 23－16＝7 除以 5 的余数等于 2,即两个余数差 3－1＝2。

当余数的差不够减时,补上除数再减。

例如:23、14 除以 5 的余数分别是 3 和 4,23－14＝9 除以 5 的余数等于 4,即两个余数差为(3＋5)－4＝4。

3.余数的乘法定理。

a 与 b 的乘积除以 c 的余数,等于 a、b 分别除以 c 的余数的积。

例如:23、16 除以 5 的余数分别是 3 和 1,所以 23×16 除以 5 的余数等于 3×1＝3。

当余数的积比除数大时,所求的余数等于余数之积再除以 c 的余数。

例如:23、19 除以 5 的余数分别是 3 和 4,所以 23×19 除以 5 的余数等于 3×4 除以 5 的余数,即 2。

乘方:如果 a 与 b 除以 m 的余数相同,那么 a^n 与 b^n 除以 m 的余数也相同。

(二)同余定理

1.同余的定义

若两个整数 a、b 被自然数 m 除有相同的余数,那么称 a、b 对于模 m 同余,用式子表示为:a≡b(mod m)左边的式子叫作同余式(读作 a 同余于 b,模 m)。

2.同余的重要性质及推论

若两个数 a,b 除以同一个数 m 得到的余数相同,则 a,b 的差定能被 m 整除。

例如:17 与 11 除以 3 的余数都是 2,所以 17－11 能被 3 整除。即如果有 a≡b(mod m),那么一定有 a－b＝mk,k 是整数,即 m|(a－b)。

3.余数判别法

当一个数不能被另一个数整除时,虽然可以用长除法求得余数,但当被除位数较多时,计算是很麻烦的。建立余数判别法的基本思想是为了求出“N 被 m 除的余数”,我们希望找到一个较简单的数 R,使得 N 与 R 对于除数 m 同余.由于 R 是一个较简单的数,所以可以通过计算 R 被 m 除的余数来求得 N 被 m 除的余数。

(1)整数 N 被 2 或 5 除的余数等于 N 的个位数被 2 或 5 除的余数;

(2)整数 N 被 4 或 25 除的余数等于 N 的末两位数被 4 或 25 除的余数;

(3)整数 N 被 8 或 125 除的余数等于 N 的末三位数被 8 或 125 除的余数;

(4)整数 N 被 3 或 9 除的余数等于其各位数字之和被 3 或 9 除的余数;

(5)整数 N 被 11 除的余数等于 N 的奇数位数之和与偶数位数之和的差被 11 除的余数(不够减的话先适当加 11 的倍数再减);

(6)整数 N 被 7、11 或 13 除的余数等于先将整数 N 从个位起按右往左每三位分节,奇数节的数之和与偶数节的数之和的差被 7、11 或 13 除的余数就是原数被 7、11 或 13 除的余数。

四、素数与约数

素数是指在大于 1 的自然数中,除了 1 和它本身以外不再有其他因数的自然数。

(一)素数的判定

1.方法一:对于一个数 N,可以从 2 枚举到 N－1,从而判断这个数是不是素数,时间复杂度为 O(N)。

```
bool ck_prime (int n) {
```

```
    if (n==1)   return false;
    if (n==2)   return true;
    for (int i=2;i<n;i++)
        if (n%i==0)
          return false;
    return true;
}
```

2.方法二：不难发现 N 的因数是成对出现的，所以对于任何一个整数 N，只需要从 1 枚举到$\sqrt{N}$，时间复杂度为 O($\sqrt{N}$)。

```
bool ck_prime (int n) {
if (n==1)     return false;
if (n==2)     return true;
for (int i=2;i*i<=n;i++)//或者 i<=sqrt(n)
    if (n% i==0)
        return false;
return true;
}
```

3.方法三：Miller-Rabin 素数判定

有时候我们想快速地知道一个数是不是素数，而这个数又特别大，导致 O($\sqrt{N}$)的算法不能通过，这时候我们可以对其进行 Miller-Rabin 素数测试，可以大概率测出其是否为素数。

需要说明的两个基本定理：

费马小定理：当 p 为素数，且 a，p 互质，有 $a^{p-1}\equiv 1 \bmod p$。反过来不一定成立，也就是说，如果 a，p 互质，且 $a^{p-1}\equiv \bmod p$，不能推出 p 是素数。

二次探测：如果 p 是一个素数，$0<x<p$，则方程 $x^2\equiv 1 \bmod p$ 的解为 x=1 或 x=p−1。

具体实现代码如下：

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
inline ll quickPow(ll a,ll b,ll p) {
    ll ans=1;
    while(b) {
        if(b & 1) ans=ans*a %p;
        a=a*a %p;
        b>>=1;
    }
    return ans;
}
const int test_time=10;
bool millerRabin(ll n) {
  if (n<3 ||n%2==0) return n==2;
```

```
    ll a=n-1,b=0;
    while (a%2==0) a/=2,++b;
    // test_time 为测试次数,建议设为不小于 10 的整数以保证正确率,但也不宜过大,否则会影响效率
    for (int i=1,j;i<=test_time;++i) {
      ll x=rand() %(n-2)+2,v=quickPow(x,a,n);
      if (v==1) continue;
      for (j=0;j<b;++j) {
        if (v==n-1) break;
        v=v*v%n;
      }
      if (j>=b) return 0;
    }
    return 1;
}
int main() {
    ll n;
    scanf("%lld",&n);
    if(millerRabin(n)) cout<<"Yes" <<endl;
    else cout<<"No"<<endl;
    return 0;
}
```

(二)素数的筛选

求 1～n 之间的所有素数,称为素数的筛选问题。

筛选素数的方法有很多种,一般都是基于“素数的倍数一定不是素数”的思想。公元前 250 年古希腊数学家埃拉托斯特尼发明了一种寻找素数的方法,我们称之为“埃拉托斯特尼筛法”。

1.方法一:埃拉托斯特尼筛法/Eratosthenes 筛选,也叫埃氏筛法。

基本思想:素数的倍数一定不是素数。

实现方法:用一个长度为 N+1 的数组保存信息(0 表示素数,1 表示非素数),先假设所有的数都是素数(初始化为 0),从小到大枚举每一个素数 x,把 x 的倍数都标记为非素数(置为 1)。实际上小于 x^2 的 x 的倍数在扫描更小的数时就已经被标记过,因此可对埃筛法优化,对于每一个 x,把大于等于 x^2 的 x 的倍数标记为合数即可,时间复杂度为 O(nloglogn)。

```
void primes(int n){
    memset(vis,0,sizeof(vis));
    vis[1]=1;              //1 不是素数标记为 1
    for (int i=2;i<=n;i++){
        if (vis[i])        //vis[i]=1 代表 i 不是素数
            continue;
        cout<<i<<endl;//否则输出素数 i
```

```
            for (int j=i;j<=n/i;j++)
                vis[i*j]=1; //大于等于 i²的 i 的倍数标记为合数
        }
    }
```

2.方法二:线性筛法(欧拉筛)

通过上述代码,发现此方法还可以继续优化,因为上述的方法中,每一个数有多组因数可能会被筛多次,例如,30 会被 2,3,5 各筛一次导致计算冗余,可以对此方法进行改进,得到更加高效的筛法。

基本思想:每个合数只被它的最小的素因子筛一次。

实现方法:数组 v 记录每个数的最小素因子,每个合数只会被它的最小的素因子筛一次,时间复杂度为 O(n)。

```
int v[MAXN],prime[MAXN];
void primes(int n){
memset(v,0,sizeof(v));   //数组 v 保存最小素因子;
int m=0;   //素数数量
for (int i=2;i<=n;i++){
    if (v[i]==0){    //i 是素数
        v[i]=i;      //素数 i 的最小素因子是其本身
        prime[++m]=i;
    }
    for (int j=1;j<=m;j++){      //给当前的数 i 乘一个素因子
        if (prime[j]> v[i]||prime[j]> n/i)
            break;          //i 有比 prime[j]更小的素因子或者超出 n 的范围
        v[i*prime[j]]=prime[j];//prime[j]是合数 i*prime[j]的最小素因子
    }
}
for (int i=1;i<=m;i++)
    cout<<prime[i]<<endl;      //输出
}
```

(三)素数的性质

1.素数 p 的约数只有两个:1 和 p。

2.初等数学基本定理:任一大于 1 的自然数,要么本身是素数,要么可以分解为几个素数之积,且这种分解是唯一的。

3.素数的个数是无限的。

4.素数的个数公式 $\pi(x)$是不减函数。

5.若 n 为正整数,在 n^2 到$(n+1)^3$ 之间至少有一个素数。

6.若 n 为大于或等于 2 的正整数,在 n 到 n! 之间至少有一个素数。

7.若素数 p 为不超过 n($n\geqslant 4$)的最大素数,则 $p>\frac{n}{2}$。

(四)质因数分解

把一个合数分解成若干个质因数乘积的形式(即求质因数的过程)叫做分解质因数(也叫分解素因数)。

对 n 分解质因数为质因数的乘积,要从最小的素数开始,我们从 2 开始试除,能整除,就输出 2,再对商继续试除,直到把包含 2 的因子"除干净";再用下一个素数试除,然后再下一个素数试除……直到商为 1 结束。

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int n,i=2;
    scanf("%d",&n);
    cout<<n<<'=';
    do {
        while (n% i==0){
            cout<<i;
            n=n/i;
            if (n!=1)cout<<'*';
        }
        i++;
    }
    while (n!=1);//n 没有除尽,继续操作
  return 0;
}
```

(五)约数的判定

若 d≥$\sqrt{N}$是 N 的约数,则 N/d≤$\sqrt{N}$也是 N 的约数,也就是说约数总数成对出现。因此,只需扫描 d=1~$\sqrt{N}$,看 d 能否整除 N,若能整除,则 N/d 也是 N 的约数,算法时间复杂度为 O($\sqrt{N}$)。

```
#include<bits/stdc++.h>
using namespace std;
int p[100100];
void deal(int n){
        int m=0;
        for (int i=1;i*i<=n;i++){
                if (n% i==0){
                        p[++m]=i;
                        if (i!=n/i)p[++m]=n/i;
                }
        }
        sort (p+1,p+1+m);
```

```
        for (int i=1;i<=m;i++)cout<<p[i]<<'  ';
}
int main(){
        int x;
        cin>> x;
        deal(x);
        return 0;
}
```

五、最大公约数与最小公倍数

(一)最大公约数:设整数 a,b,…,c 不全为零,同时整除 a,b,…,c 的整数(如±1)称为它们的公约数。因为 a,b,…,c 不全为零,故同时整除 a,b,…,c 的整数只有有限多个,其中最大的一个称为 a,b,…,c 的最大公约数,用符号(a,b,…,c)表示。

当(a,b,…,c)=1(即 a,b,…,c 的公约数只有±1)时,称 a,b,…,c 互素(互质)。注意,(如涉及三个及三个以上的整数)此时不能推出 a,b,…,c 两两互素;但反过来,若 a,b,…,c 两两互素,则显然有(a,b,…,c)=1。

(二)最小公倍数:设 a,b,…,c 均是非零整数,一个同时为 a,b,…,c 倍数的数称为它们的公倍数,a,b,…,c 的公倍数有无穷多个,其中最小的一个正整数称为 a,b,…,c 的最小公倍数,记作[a,b,…,c]。

(三)由最大公约数、最小公倍数的定义,不难得出它们的一些简单性质:

1.a,b 的任何一个公约数都是它们最大公约数的约数,即若 m|a,m|b,则 m|(a,b);

2.a,b 的任何一个公倍数都是它们最小公倍数的倍数,即若 a|m,b|m,则[a,b]|m;

3.若 $b\in N_+$,则(0,b)=b,[1,b]=b;

4.对于任意的整数 x,有(a,b)=(a,b+ax);

5.(辗转相除法)设 $a,b\in Z,b\neq 0$,由带余除法知 $a=bq+r,0\leqslant r<|b|,q,r\in Z$。

故(a,b)=(b,r)。

设 $a>b>0$,作带余除法如下:

$a=bq_1+r_1,0\leqslant r_1<b,q_1,r_1\in Z$;

$b=r_1q_2+r_2,0\leqslant r_2<r_1,q_2,r_2\in Z$;

……

由于 $b>r_1>r_2>\cdots$,故必存在 $n\in N_+$,使得 $r_n=0$,而 $r_{n-1}\neq 0$(约定 $r_0=b$).

则有 $(a,b)=(b,r_1)=(r_1,r_2)=\cdots=(r_{n-1},r_n)=r_{n-1}$。

由此不仅得到 a,b 的最大公约数,而且将上述过程反过来,即得裴蜀定理。

6.裴蜀定理:设 a,b 是不全为 0 的整数,则存在整数 x,y,使得 ax+by=(a,b);

(其中 x,y 不唯一。如加上等式 ab−ab=0,有 a(b+x)+b(y−a)=1,即裴蜀等式除通过辗转相除得到外,也可通过“凑”、恒等变形等其他方法得到)。

特别地,两个整数 a,b 互素的充要条件是存在整数 x,y,使得 ax+by=1。

事实上,条件的必要性是裴蜀定理的一个特例。反过来,若有 x,y 使等式成立,不妨设(a,b)=d,则 d|a,d|b,故 d|ax 及 d|by,于是 d|(ax+by),即 d|1,从而 d=1。

7.设两个正整数 a,b 的分解式分别为

$a=p_1^{a1}p_2^{a2}\cdots p_k^{ak}$，$\alpha_i\geqslant 0$，$i=1,2,,\cdots,k$；

$b=p_1^{\beta 1}p_2^{\beta 2}\cdots p_k^{\beta k}$，$\beta_i\geqslant 0$，$i=1,2,,\cdots,k$；

取 $\gamma_i=\min\{\alpha_i,\beta_i\}$，$\delta_i=\max\{\alpha_i,\beta_i\}$，

则 $(a,b)=p_1^{\gamma 1}p_2^{\gamma 2}\cdots p_k^{\gamma k}$，$[a,b]=p_1^{\delta 1}p_2^{\delta 2}\cdots p_k^{\delta k}$。

注：此结论对多个正整数 $a_1,a_2,\cdots,a_n$ 仍成立。

8.两个整数的最大公约数与最小公倍数满足：$(a,b)\cdot[a,b]=|ab|$。

（但注意，这只限于两个整数的情形，对于多于两个整数的情形，类似结论不成立。）

（四）最大公约数和最小公倍数的求法

假设 x 和 y 的最大公约数是 m，最小公倍数是 n，由性质 8 可知，$xy=mn$。故只需求出最大公约数即可求得最小公倍数。

1.方法一：枚举法

对两个整数 a 和 b，共同的约数在 1 和 min(a,b)之间，从大到小枚举，若判断它能同时整除 a 和 b，即为两个数的最大公约数。该算法的最坏时间复杂度为 O(min(a,b))。

```
#include<bits/stdc++.h>
using namespace std;
int a,b;
int main(){
    cin>>a>>b;
    for (int i=min(a,b);i>0;i--){
        if (a% i==0&&b% i==0){
            cout<<i<<''<<a*b/i;
            break;
        }
    }
    return 0;
}
```

2.方法二：更相减损算法

若 c 是 a 和 b 的公约数(a>b)，c 亦是 b 和 a−b 的公约数，可以表示成 gcd(a,b)=gcd(b,a−b)。采用函数进行迭代，当迭代到 a=b 时，a 为 a 和 b 的最大公约数，该算法时间复杂度为 O(max(a,b))。

(1)具体代码递归实现如下：

```
#include<bits/stdc++.h>
using namespace std;
int gcd (int a,int b){
    if (a==b) return a;
    if (a<b) {
        swap (a,b);
    }
    return gcd(b,a-b);
}
```

```
int a,b;
int main(){
    cin>>a>>b;
    cout<<gcd(a,b)<<''<<a*b/gcd(a,b)<<endl;
    return 0;
}
```

(2)具体代码非递归实现如下：

```
#include<bits/stdc++.h>
using namespace std;
int gcd (int a,int b){//非递归
    while (a!=b){
        if (a>b)a=a-b;
        else b=b-a;
    }
    return a;
}
int a,b;
int main(){
    cin>>a>>b;
    cout<<gcd (a,b)<<''<<a*b/gcd (a,b)<<endl;
    return 0;
}
```

3.方法三：辗转相除法(欧几里得算法)

可以表示成 gcd(a,b)==gcd(b,a%b)(a>b)。因此可以采用函数进行迭代(即递归)，当迭代到 b==0 时，a 为 a 和 b 的最大公约数。若 b<=a/2，gcd(a,b)变为 gcd(b,a%b)，规模至少缩小一半，若 b>a/2，a%b 也将规模最少缩小为一半。即经过一次迭代，gcd(a,b)数据规模至少变为原来的一半，所以该算法的时间复杂度为 $O(\log_2(\max(a,b)))$。

输入 a,b，输出它们的最大公约数和最小公倍数代码如下：

```
#include<bits/stdc++.h>
using namespace std;
int a,b;
int gcd (int a,int b){
    if (b==0)
    return a;
    else return gcd(b,a% b);
}
int main(){
    cin>>a>>b;
    cout<<gcd(a,b)<<''<<a*b/gcd(a,b);
//为了防止 a*b 爆 long long,经常写成 a/gcd(a,b)*b
    return 0;
}
```

课堂练习

1.【NOIP2013】下面是根据欧几里得算法编写的函数，它所计算的是 a 和 b 的(　　)。

```
int euclid(int a,int b){
    if (b==0)
        return a;
    else
        return euclid(b,a% b);
}
```

A. 最大公共素因子　　B. 最小公共素因子

C. 最大公约数　　D. 最小公倍数

【答案】C

【分析】欧几里得算法用来计算 a 和 b 的最大公约数。

2.【NOIP2018】10000 以内，与 10000 互质的正整数有(　　)个。

A. 2000　　B. 4000　　C. 6000　　D. 8000

【答案】B

【分析】考察数论基础知识。10000 的素因子只有 2 和 5，能被 2 整除的数有 5000 个，能被 5 整除的数有 2000 个，但是即能被 2 整除，也能被 5 整除的数被算了两遍，即能被 10 整除的 1000 个数。所以答案是 10000－5000－2000＋1000＝4000(详见第 4 节容斥原理)。

3.【NOIP2018】从 1 到 2018 这 2018 个数中，共有(　　)个包含数字 8 的数。

【答案】544

【分析】考察数学知识。

个位数是 8 的个数是$\left\lfloor\frac{2018}{10}\right\rfloor$＝201 加上 2018 这个数就是 202 个。

十位数是 8 个位数不是 8 的个数是$\left\lfloor\frac{2018}{100}\right\rfloor$*(10－1)＝180 个。

百位数是 8，十位个位都不是 8 的是$\left\lfloor\frac{2018}{1000}\right\rfloor$*$(10-1)^2$＝162 个。

所以总数有 202＋180＋162＝544 个。

第2节 排列组合

排列组合是组合学的基本概念。排列是从给定个数的元素中取出指定个数的元素进行排序。组合是从给定个数的元素中仅仅取出指定个数的元素，不考虑排序。因此排列与元素的顺序有关，组合与顺序无关。如 231 与 213 是两个排列，{2,3,1}和{2,1,3}是一个组合。

一、两个基本原理

（一）加法原理

做一件事，完成它可以有 n 类办法，在第一类办法中有 m_1 种不同的方法，在第二类办法中有 m_2 种不同的方法，……，在第 n 类办法中有 m_n 种不同的方法，那么完成这件事共有 $N=m_1+m_2+m_3+\cdots+m_n$ 种不同方法。

加法原理分类计数法的要求：每一类中的每一种方法都可以独立地完成此任务；两类不同办法中的具体方法，互不相同（即分类不重）；完成此任务的任何一种方法，都属于某一类（即分类不漏）。

（二）乘法原理

做一件事，完成它需要分成 n 个步骤，做第一步有 m_1 种不同的方法，做第二步有 m_2 种不同的方法，……，做第 n 步有 m_n 种不同的方法，那么完成这件事共有 $N=m_1\times m_2\times m_3\times\cdots\times m_n$ 种不同的方法。

这里要注意区分两个原理，要做一件事，完成它若是有 n 类办法，是分类问题，第一类中的方法都是独立的，因此用加法原理；做一件事，需要分 n 个步骤，步与步之间是连续的，只有将分成的若干个互相联系的步骤，依次相继完成，这件事才算完成，因此用乘法原理。

这样完成一件事的分“类”和“步”是有本质区别的，因此也将两个原理区分开来。

乘法原理分步计数法的要求：任何一步的一种方法都不能完成此任务，必须且只须连续完成这 n 步才能完成此任务；各步计数相互独立；只要有一步中所采取的方法不同，则对应完成此事的方法也不同。

二、排列与组合

（一）排列的定义

从 n 个不同元素中，任取 m($m\leqslant n$)个元素，按照一定的顺序排成一列，叫作从 n 个不同元素中取出 m 个元素的一个排列。

从排列的意义可知，如果两个排列相同，不仅这两个排列的元素必须完全相同，而且排列的顺序必须完全相同，这就告诉了我们判断两个排列是否相同的方法。

（二）排列数公式

从 n 个不同元素中取出 m($m\leqslant n$)个元素的所有排列，可以用记号和公式表示为 $P_n^m=A_n^m=\dfrac{n!}{(n-m)!}$。

当 $m=n$ 时，称为全排列，此时表示为 $A_n^n=P_n^n=n!=n*(n-1)*\cdots*2*1$。

（三）组合的定义

从 n 个不同元素中，任取 m($m\leqslant n$)个元素并成一组，叫作从 n 个不同元素中取出 m 个

元素的一个组合。

从组合的定义知，如果两个组合中的元素完全相同，不管元素的顺序如何，都是相同的组合；只有当两个组合中的元素不完全相同时，才是不同的组合。

（四）组合数的公式

组合的记号和公式可以表示为：

$$C_n^m=\binom{n}{m}=\frac{n!}{m!(n-m)!}=C_n^{n-m}$$

由此可知：

$A_n^m=C_n^m A_m^m=C_n^{n-m}A_m^m$

$C_n^m=C_{n-1}^m+C_{n-1}^{m-1}$

$C_n^0=C_n^n=1$

$0!=1$

三、排列组合常见的计数方法

（一）特殊优先

特殊元素，优先处理；特殊位置，优先考虑

例 1：六人站成一排，求

①甲不在排头，乙不在排尾的排列数；

②甲不在排头，乙不在排尾，且甲乙不相邻的排列数。

分析一：

先考虑排头，排尾，但这两个要求相互有影响，因而考虑分类。

第一类：乙在排头，有 P(5,5)种站法；

第二类：乙不在排头，当然也不能在排尾，有 4 * 4 * P(4,4)种站法；

共 P(5,5)+4 * 4 * P(4,4)种站法。

分析二：

第一类：甲在排尾，乙在排头，有 P(4,4)种方法；

第二类：甲在排尾，乙不在排头，有 3 * P(4,4)种方法；

第三类：乙在排头，甲不在排头，有 4 * P(4,4)种方法；

第四类：甲不在排尾，乙不在排头，有 6 * P(4,4)种方法；

共 P(4,4)+3 * P(4,4)+4 * P(4,4)+6 * P(4,4)=312 种。

（二）捆绑法与插空法

1.捆绑法：在解决排列问题的时候，有些问题需要相邻元素在一起，这种把相邻元素看作一个整体，再与其他元素一起排列，同时注意捆绑元素的内部排列的方法叫捆绑法。

例 2：8 人排成一队

①甲乙必须相邻；

②甲乙不相邻；

③甲乙必须相邻且与丙不相邻；

④甲乙必须相邻，丙丁必须相邻；

⑤甲乙不相邻，丙丁不相邻。

分析：①甲乙必须相邻，就是把甲乙捆绑（甲乙可交换）和 7 人排列 P(7,7) * 2；

②甲乙不相邻，P(8,8)－P(7,7) * 2；

③甲乙必须相邻且与丙不相邻，先求甲乙必须相邻且与丙相邻，P(6,6) * 2 * 2；甲乙必须相邻且与丙不相邻，P(7,7) * 2－P(6,6) * 2 * 2；

④甲乙必须相邻，丙丁必须相邻 P(6,6) * 2 * 2；

⑤甲乙不相邻，丙丁不相邻，P(8,8)－P(7,7) * 2 * 2＋P(6,6) * 2 * 2。

例 3：7 个不同的球放到 6 个不同的盒子中，要求每个盒子至少放一个球，一共有多少种方法？

解答：根据题目要求，其中一个盒子必须放 2 个，其他每个盒子放 1 个球，所以从 7 个球中挑出 2 个球看成一个整体，则有 C(7,2)，这个整体和剩下 5 个球放入 6 个盒子里，则有 P(6,6)，方法是 C(7,2) * P(6,6)。

2.插空法：在解决对于某几个元素要求不相邻的问题时，先将其他元素排好，再将指定的不相邻的元素插入已排好元素的间隙或两端位置，从而将问题解决的方法叫插空法。

例 4：某人射击 8 枪，命中 4 枪，恰好有三枪连续命中，有多少种不同的情况？

分析：因为连续命中的三枪与单独命中的一枪不能相邻，因而这是一个插空问题。另外没有命中的之间没有区别，不必计数。即在四发空枪之间形成的 5 个空中选出 2 个的排列，即 P(5,2)。

例 5：马路上有编号为 1,2,3,…,10 十个路灯，为节约用电又看清路面，可以把其中的三只灯关掉，但不能同时关掉相邻的两只或三只，在两端的灯也不能关掉的情况下，求满足条件的关灯方法共有多少种？

分析：即关掉的灯不能相邻，也不能在两端。又因为灯与灯之间没有区别，因而问题转化为在 7 盏亮着的灯形成的不包含两端的 6 个空中选出 3 个空放置熄灭的灯。

所以共 C(6,3)＝20 种方法。

（三）排除法

排除法：先计算中所有情况的答案，再把不符合条件的答案排除掉的方法叫排除法。

例 6：三行三列共九个点，以这些点为顶点可组成多少个三角形？

分析：有些问题正面求解有一定困难，可以采用间接法。所求问题的方法数＝任意三个点的组合数－共线三点的方法数。

所以，共 C(9,3)－8＝76 种。

例 7：正方体 8 个顶点中取出 4 个，可组成多少个四面体？

分析：所求问题的方法数＝任意选四点的组合数－共面四点的方法数。

所以，共 C(8,4)－12＝70－12＝58 个。

（四）隔板法

隔板法：隔板法使用的前提是所有元素必须相同，然后需要把这些元素分成若干组，每组至少有一个元素的计数方法。

1.隔板法的基本题型

n 个相同元素，不同个 m 组，每组至少有一个元素；则只需在 n 个元素的 n－1 个间隙中放置 m－1 块隔板把它隔成 m 份，求共有多少种不同方法？其解题思路为：将 n 个相同的元素排成一行，n 个元素之间出现了(n－1)个空档，现在我们用(m－1)个“档板”插入(n－1)个空档中，就把 n 个元素隔成有序的 m 份，每个组依次按组序号分到对应位置的几个元素(可能是 1 个、2 个、3 个、4 个、….)，这样不同的插入办法就对应着 n 个相同的元素分到 m 组的一种分法，这种借助于虚拟“档板”分配元素的方法称为插板法。

例 8：10 个名额分配到八个班，每班至少一个名额，问有多少种不同的分配方法？

分析：把 10 个名额看成十个元素，在这十个元素之间形成的九个空中，选出七个位置放置档板，则每一种放置方式就相当于一种分配方式。因而共有 $C_9^7=C_9^2=36$ 种。

2.使用隔板法求不定方程的正整数解的个数

不定方程 $x_1+x_2+\cdots+x_m=n(m,n\in N)$ 正整数解 $(x_1,x_2,x_3\cdots,x_n)$ 的个数为 $C(m-1,n-1)$。

求解过程分析如下：n 可看作是 n 个 1(k 个 1 代表数值 k)，1 与 1 之间有 n−1 个空位，要保证是正整数解，即每个 $x_i(i=1,2,\cdots\cdots,m)$ 位置上必须保证有一个 1，则只能在 1 与 1 之间的 n−1 个空位的相应位置上插入 m−1 个隔板(隔成 m 个 x)，第 i 个隔板前的两个隔板间的 1 的个数即表示 x_i(紧靠第 i 个隔板前的 k 个 1 即表示 x=k)。所以从 n−1 个 1 与 1 之间的空位中选 m−1 个空位插入隔板可解得不定方程解的个数，即 $C(m-1,n-1)$。

不定方程 $x_1+x_2+\cdots+x_m=n(m,n\in N,)$ 非负整数解 $(x_1,x_2,\cdots\cdots x_m)$ 的个数为 $C(n,n+m-1)=C(m-1,n+m-1)$。

求解过程分析如下：令 $y_i=x_i+1(i=1,2,\cdots\cdots,m)$，则 $y_1+y_2+\cdots+y_m=n+m$，所以不定方程 $x_1+x_2+\cdots+x_m=n$ 的非负整数解的个数等于不定方程 $y_1+y_2+\cdots+y_m=n+m$ 的正整数解的个数，即 $C(m-1,n+m-1)=C(n,n+m-1)$。

四、可重复排列

从 n 个不同的元素中，每次取出 m 个元素，但同一元素可以重复取出，排成一列，称为一个可重复排列。在作一个可重复排列时，如果元素 a 被取上几次，排列中它就出现几次，但同一元素的位置交换不能认为是不同排列。两个可重复排列相同当且仅当所取的元素相同，并且同一元素取的次数相同，在排列中占的位置也相同。从 n 个元素中可重复地选取 m 个元素的可重复排列个数称为可重复排列种数。

例如，由 1,2,3,4,5,6,7,8,9 可以组成多少个五位数？最高位可以是 1,2,3,4,5,6,7,8,9 中的任何一个，因而有 9 种确定最高位的方式；由于题目中没有限制数字不重复，即允许数字重复，因而次高位，一直到个位都各有 9 种确定的方式；

因此可以组成 $9\times9\times9\times9\times9=9^5$(个)五位数。

课堂练习

1.【NOIP2016】有 7 个一模一样的苹果，放到 3 个一样的盘子中，一共有(　　)种放法。

A. 7　　B. 8　　C. 21　　D. 3^7

【答案】B

【分析】穷举一下，因为盘子一样，保证为不减序列，8 种放法，分别为(0,0,7),(0,1,6),(0,2,5),(0,3,4),(1,1,5),(1,2,4),(1,3,3),(2,2,3)。

2.【NOIP2017】甲、乙、丙三位同学选修课程，从 4 门课程中，甲选修 2 门，乙、丙各选修 3 门，则不同的选修方案共有(　　)种。

A. 36　　B. 48　　C. 96　　D. 192

【答案】C

【分析】根据组合与乘法原理，四门选修课选两门的方法为 $\frac{4*3}{2}=6$ 种，选 3 门可视为不要一

门的方案，即 4 种。由乘法原理可得答案为 6 * 4 * 4＝96。

3.【NOIP2018】设含有 10 个元素的集合的全部子集数为 S，其中由 7 个元素组成的子集数为 T，则 T/S 的值为(　　)。

A. 5/32　　B. 15/128　　C. 1/8　　D. 21/128

【答案】B

【分析】考察排列与组合基础知识。10 个元素的集合的子集数有 $2^{10}=1024$ 个，由 7 个元素组成的子集数有 $C_{10}^{7}=C_{10}^{3}=120$ 种，故答案是 $\frac{120}{1024}=\frac{15}{128}$。

4.【NOIP2008】书架上有 4 本不同的书 A、B、C、D。其中 A 和 B 是红皮的，C 和 D 是黑皮的。把这 4 本书摆在书架上，满足所有黑皮的书都排在一起的摆法有_____种。满足 A 必须比 C 靠左，所有红皮的书要摆放在一起，所有黑皮的书要摆放在一起，共有_____种摆法。

【答案】12　4

【分析】①首先我们将两本黑书捆绑在一起，他们是分先后的，再将它与两本红书放在一起进行全排列，得 $P_3^3 * P_2^2=12$。②问题规模很小，完全可以枚举，有 ABCD，BACD，ABDC，BADC 4 种。

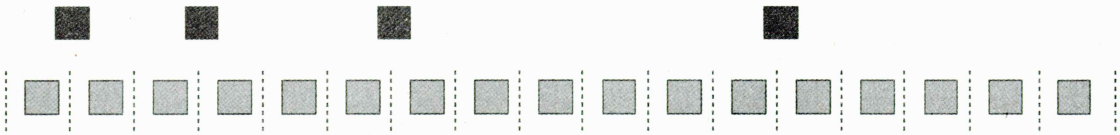
5.【NOIP2008】书架上有 21 本书，编号从 1 到 21，从其中选 4 本，其中每两本的编号都不相邻的选法一共有_____种。

【答案】3060

【分析】对于不相邻的选法，可以先不考虑不相邻元素的排法，而是先将余下的元素排好，然后将需要不相邻的元素在已排好元素间插空。

为了便于理解，可以将问题转化为：将 21 个盒子(17 个黑色、4 个红色)摆成一行，要求 4 个红色的盒子不能相邻(注：对于每一种摆法，将书按编号依次放入，其红色盒子对应书的编号肯定就是不相邻的，即为一种解，见下图)。

计算：将 17 个黑色盒子摆成一行，这 17 个盒子间共有 18 个空位，在这些空位中任选 4 个，即为 C_{18}^{4} 种。



4个不相邻元素的一种放法（18条红线表示17个盒子间的18空位）

【推广】n 个元素中有 m 个元素不相邻的放法有 C_{n-m+1}^{m}。对于本题 $C_{21-4+1}^{4}=C_{18}^{4}=3060$。

6.【NOIP2017】将 7 个名额分给 4 个不同的班级，允许有的班级没有名额，有(　　)种不同的分配方案。

A. 60　　B. 84　　C. 96　　D. 120

【答案】D

【分析】经典的组合数题目，采用隔板法，先给总数＋4，使得题目变成 11 个人分给 4 个班，每个班至少有一个人，接下来就是直接计算答案了，总共 10 个间隔，需要选择 3 个位置放下隔

板，方案数共有 $C_{7+4-1}^{3}=C_{10}^{3}=120$ 种。

7.【NOIP2011】每份考卷都有一个 8 位二进制序列号。当且仅当一个序列号含有偶数个 1 时，它才是有效的。例如，00000000、01010011 都是有效的序列号，而 11111110 不是。那么，有效的序列号共有_____个。

【答案】128

【分析】$\frac{2^8}{2}=128$，也可以理解成前面七位随便取，通过最后一位来确定奇偶性。

8.【NOIP2012】在全国赛期间，主办单位为了欢迎来自全国各地的选手，举行了盛大的晚宴。在第十八桌，有 5 名大陆选手和 5 名港澳选手共同进膳。为了增进交流，他们决定相隔就坐，即每个大陆选手左右相邻的都是港澳选手、每个港澳选手左右相邻的都是大陆选手。那么，这一桌共有_____种不同的就坐方案。注意：如果在两个方案中，每个选手左边相邻的选手均相同，视为同一个方案。

【答案】2880

【分析】先让大陆选手坐成一周的排列方法有 $N_1=\frac{P_5^5}{5}$，港澳选手插入到 5 个空中的排列种数有 $N_2=P_5^5$，就坐方案的总数是 $N=N_1*N_2=2880$。

9.【NOIP2013】7 个同学围坐一圈，要选 2 个不相邻的作为代表，有_____种不同的选法。

【答案】14

【分析】选择第一个同学共有 7 种选项，第二个同学由于不能相邻，只剩下 4 个可选，同时每对同学会被重复计算两次，因此一共有 $\frac{7*4}{2}=14$ 种不同的选法。

10.【NOIP2014】由数字 1,1,2,4,8,8 所组成的不同的四位数的个数是_____。

【答案】102

【分析】可以按照包含的数字分类讨论：包含 1248 的有 24 种，1188 的有 6 种，11xx 一共有 $C_4^2*P_3^2=36$ 种，同理 88xx 也有 36 种，合计 102 种。

11.【NOIP2016】从一个 4×4 的棋盘(不可旋转)中选取不在同一行也不在同一列上的两个方格，共有_____种方法。

【答案】72

【分析】先定第一个棋子后，第二个棋子只能在与之不同的 3 行 3 列上选取，即有 9 种选法。由于棋子的选择没有先后顺序，每种方案都被重复计算了一次，最终结果需要除以 2。因此总方案数计算得 $\frac{16*9}{2}=72$ 种。

12.【NOIP2018】方程 a * b=(a or b) * (a and b)，在 a,b 都取[0,31]中的整数时，共有_____组解(* 表示乘法；or 表示按位或运算；and 表示按位与运算)。

【答案】454

【分析】首先需要知道一个结论：

a and b<=min(a,b)<=max(a,b)<=a or b<2*max(a,b)，

那么对答案的贡献形式一定是 a and b=min(a,b) 并且 a or b=max(a,b)，

那么最小值的二进制一定是最大值二进制的子集。

枚举所有二进制可能的位数，由于要删去a=b的重复情况，对于i位二进制数，答案应该为 $2*\sum_{i=0}^{5} C_5^i * 2^i = 486$，求和减去32之后得到答案454。

13.【NOIP2009】小陈现有2个任务A、B要完成，每个任务分别有若干步骤如下：A=a1->a2->a3，B=b1->b2->b3->b4->b5。在任何时候，小陈只能专心做某个任务的一个步骤。但是如果愿意，他可以在做完手中任务的当前步骤后，切换至另一个任务，从上次此任务第一个未做的步骤继续。每个任务的步骤顺序不能打乱，例如……a2->b2->a3->b3……是合法的，而……a2->b3->a3->b2……是不合法的。小陈从B任务的b1步骤开始做，当恰做完某个任务的某个步骤后，就停工回家吃饭了。当他回来时，只记得自己已经完成了整个任务A，其他的都忘了。试计算小陈饭前已做的可能的任务步骤序列共有(　　)种。

【答案】70

【分析】完成任务可能情况有{a1,a2,a3,b1},{a1,a2,a3,b1,b2},{a1,a2,a3,b1,b2,b3},{a1,a2,a3,b1,b2,b3,b4},{a1,a2,a3,b1,b2,b3,b4,b5}，每组里面任意取出3个位置留给任务A的三个子任务，并且b1要放在第一个，应该是组合数，所以得到 $C_3^3+C_4^3+C_5^3+C_6^3+C_7^3=70$。

14.【NOIP2009】某个国家的钱币面值有1,7,7^2,7^3共计四种，如果要用现金付清10015元的货物，假设买卖双方各种钱币的数量无限且允许找零，那么交易过程中至少需要流通(　　)张钱币。

【答案】35

【分析】10015化成7进制数是41125，正常是4×7+1=29张7^3面额的，1张7^2面额，2张7面额的，5张1面额的。因为可以无限且允许找零，并要求最少流通数量。这样就把7进制上大于等于4的数a，用找零7−a的方法代替，这样就能达到最少。这里29、1、2、5中只有5是大于4的，所以用一张大额的，并7−5找零的方法计算。这样，总数为29+1+2+(1+7−5)=35张。

15.【NOIP2002】将N个红球和M个黄球排成一行。例如，N=2，M=3可得到以下10种排法：

红红黄黄黄红黄红黄黄红黄黄红黄黄红红黄黄黄红黄红黄黄黄黄红红。

问题：当N=4，M=3时有(　　)种不同排法。(不用列出每种排法)

【答案】35

【分析】这是一个可重复排列的问题，求解方法为(4+3)!/4!/3!=35。

第3节 特殊数列

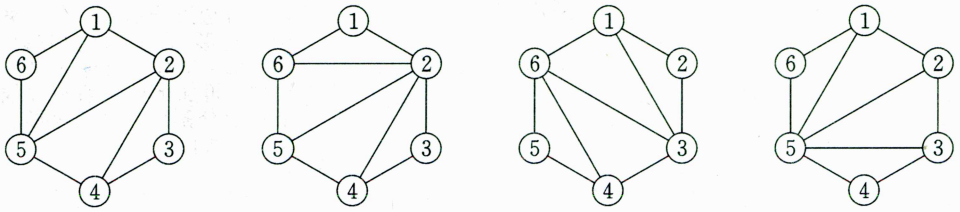
一、Catalan 数

在一个有 n+2 条边的凸多边形中，可以画出 n−1 条不相交的对角线将多边形分为 n 个三角形，设所有满足条件的方案数为 h_n，定义 $h_0=1$，则

$$h_n=\frac{1}{n+1}C_{2n}^{n}=C_{2n}^{n}-C_{2n}^{n-1}=\sum_{i=0}^{n-1}h_i h_{n-1-i}(h_0=1,h_1=1)。$$

其前几项为(从第 0 项开始)：1,1,2,5,14,42,…。

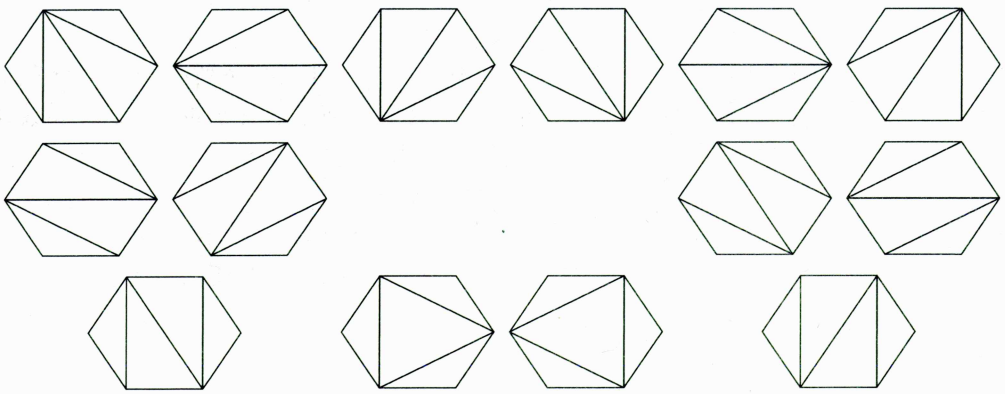
这个式子是如何得到的呢？我们简单推演一下，比如，当 n=4 时，凸多边形为凸六边形，有 3 条不相交的对角线，其中四种图形如下图所示：



由上图可知，每次在连接完一条不相交的对角线后，可以看出这条对角线把当前图分割成了 2 部分。设其中一块有 k+2 条边(这样就可以得到 k 个三角形)。同理，另一块图形可得到 n−1−k 个三角形。由乘法原理，这样剖分的方案数就为 h(k) * h(n−k−1)(其中 n 为三角形的总数量)。

显然，k 可以从 0 一直变化到 n−1(注意要考虑与边界情况，并注意到这里的不会重复计数)。

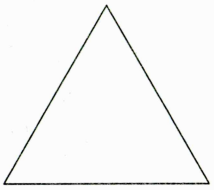
具体的方案数为 h(4)=h(0) * h(3)+h(1) * h(2)+h(2) * h(1)+h(3) * h(0)=14。下图是具体的 14 种方案。



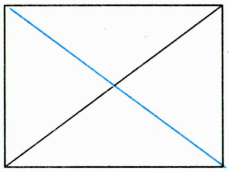
以三角形为例，我们初始化 h(0)=1(注：h 为 Catalan 数函数)，那三角形(n=1)对应的 h(1)=h(0) * h(0)=1。三角形划分成三角形的方案个数也为 1(它本身)，然后我们通过三角形推出四边形(n=2)，四边形可以通过连接一条对角线划分成两个三角形，因为有两条对角线，划分方案有两种，此时 h(2)=h(0) * h(1)+h(1) * h(0)=2。

我们可以通过三角形和四边形推出五边形(n=3)的划分方案个数。定住五边形的五个顶点，以其中任意两个顶点为三角形的两个顶点，再任意找出其他顶点中的任意一个顶点为

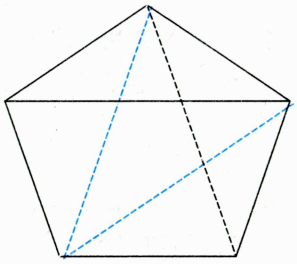
三角形的第三个顶点，我们发现可以划分成两种（一个三角形和一个四边形），一种（三个三角形），故而递归下去计算 h(3)=h(0)*h(2)+h(1)*h(1)+h(2)*h(0)=5，以此类推 h(n)=h(0)*h(n−1)+…+h(n−1)*h(0)。



三角形：一种方案



四边形：两种方案（黑色一种分成两个，蓝色一种分成两个，总计两种方案）。



五边形：五种方案[黑色实线和黑色虚线都将五边形分割成一个三角形和一个四边形，（两种这种方案），其中四边形又可分成两种方案，现在是四种，再加上两条蓝色虚线将整个五边形分割成三个三角形的方案，共计五种]

二、Catalan 数列的应用

（一）括号序列：给出一个有 n 个运算符 n+1 个运算数的算式，要求在算式中任意添加括号，那么不同的运算顺序有多少种？

分析：考虑最后一个运算的符号，它的左边有 k 个运算符，右边则有 n−1−k 个运算符，由乘法原理可知这个问题满足递推关系（1），于是答案是 Catalan 数列。

（二）有 2n 个人排队进入剧场，入场费 50 元，其中 n 个人带着一张 50 元，另外 n 个人带着 100 元的钞票，剧院售票处没有任何零钱，有多少种情况满足无论什么时候售票处都能找的开零钱。

（三）一个 n*n 的网格图，从左下走到右上，每次只能向上或向右走，不能走到左下到右上的对角线的上方，一共有多少种走法？

分析：（二）、（三）两个问题本质是求 n 个+1 和 n 个−1 组成的序列中任意部分和都非负的序列总数（模型的转换希望读者认真思考），于是这两个问题的答案也是 Catalan 数列。

（四）有 n 个结点的形态不同的二叉树一共有多少棵？

分析请读者自行完成，这个问题的答案也满足递推关系（1）。

（五）出栈序列统计，有 n 个数和一个栈，本质不同的合法序列一共有多少种？

提示：将一次入栈操作视为+1，一次出栈操作视为−1，剩余的分析请读者自行完成。

三、第二类斯特林数

（一）第二类斯特林数定义

第二类斯特林数 S(n,k) 是将 n 个元素的集合划分成 k 个不可辨认的非空盒子的划分个数。注意，这里的不可辨认是指不能把一个盒子与另一个盒子分辨开，它们看起来都一样。

下面给出第二类斯特林数的递推关系和证明：

S(n,k)=k*S(n−1,k)+S(n−1,k−1)；S(n,1)=1(n≥1)，S(n,n)=1。

上面的递推式可以用组合证明：一方面，如果将元素 n 单独拿出来划分成 1 个集合，那

么方法数是 S(n−1,k−1)；另一方面，如果元素 n 所在的集合不止一个元素，那么可以先将剩下的 n−1 个元素划分好以后再选一个集合把 n 放进去，方法数是 k * S(n−1,k)。

由第二类斯特林数的定义可知，第二类斯特林数表示把 n 个不同的数划分为 m 个集合的方案数，要求不能为空集，写作 S(n,m)。划分集合不必考虑排列次序。考虑 S(n,m)可以由以下两项转移得到：

1.S(n−1,m−1)，将 n−1 个不同元素划分为 m−1 个集合，则第 n 个元素必须单独放入第 m 个集合，方案数为 S(n−1,m−1)。

2.S(n−1,m)，将 n−1 个不同元素已经划分为 m 个集合，则第 n 个元素可以放在 m 个集合中任意一个里面，方案数为 S(n−1,m) * m。

由上可以得出递推式：S(n,m)=S(n−1,m−1)+S(n−1,m) * m。

3.在各种不同的盒子中放球的模型的方案数：

(1)n 个不同的球放入 m 个相同的盒子中，不允许盒子为空：方案数=S(n,m)，这是第二类斯特林数的定义。

(2)n 个不同的球放入 m 个不同的盒子中，不允许盒子为空：方案数=S(n,m) * m!，盒子有区别，乘上盒子所有排列即可。

(3)n 个不同的球放入 m 个相同的盒子中，允许盒子为空：

方案数=$\sum_{i=0}^{m}$S(n,i)，枚举非空盒子个数即可。

(4)n 个不同的球放入 m 个不同的盒子中，允许盒子为空：

方案数=$\sum_{i=0}^{m}$A(i,m) * S(n,i)，因为盒子不同，所以要乘上排列数。

例题：盒子与球。

有 3 个一模一样的盒子和 6 个不同颜色的球，将这些球放到 3 个盒子里并且保证每个盒子里至少有 1 个球，求放置的方案总数。

答案：第二类斯特林数 S(6,3)。

课堂练习

1.【NOIP2018】关于 Catalan 数 Cn=(2n)!/(n+1)!/n!，下列说法中错误的是(　　)。

A. Cn 表示有 n+1 个结点的不同形态的二叉树的个数

B. Cn 表示含 n 对括号的合法括号序列的个数

C. Cn 表示长度为 n 的入栈序列对应的合法出栈序列个数

D. Cn 表示通过连接顶点而将 n+2 边的凸多边形分成三角形的方法个数

【答案】A

【分析】考察 Catalan 数的知识，这句话正确的应该是 C_n 表示有 n 个结点的不同形态二叉树的个数。

这道题如果不了解也不要紧，自己举个例子，选一个数字代入验证即可。

2.【NOIP2015】结点数为 5 的不同形态的二叉树一共有_____种(结点数为 2 的二叉树一共有 2 种：一种是根结点和左儿子，另一种是根结点和右儿子)。

【答案】42

【分析】利用 DP 求解，设 f(i) 表示结点数为 i 的二叉树的形态数，f(i)=$\sum_{k=0}^{i-1}$f(k) * f(i−k+1)，f(0)=1。

这其实就是 Catalan 数，f(0)＝1，f(1)＝1，f(2)＝2，f(3)＝5，f(4)＝14，f(5)＝42。

3.【NOIP2014】把 M 个同样的球放到 N 个同样的袋子里，允许有的袋子空着不放，问共有多少种不同的放置方法(用 K 表示)？

例如：M＝7，N＝3 时，K＝8；在这里认为(5，1，1)和(1，5，1)是同一种放置方法。

问：M＝8，N＝5 时，K＝_____。

【答案】18

【分析】使用列举法解决，注意做到不重、不漏。建议按照有球袋子数分类列举，每种方案严格按照非降排列(省略空袋子)：

1 个有球的袋子：(8)；

2 个有球的袋子：(1，7)(2，6)(3，5)(4，4)；

3 个有球的袋子：(1，1，6)(1，2，5)(1，3，4)(2，2，4)(2，3，3)；

4 个有球的袋子：(1，1，1，5)(1，1，2，4)(1，1，3，3)(1，2，2，3)(2，2，2，2)；

5 个有球的袋子：(1，1，1，1，4)(1，1，1，2，3)(1，1，2，2，2)；

故总方案数为 1＋4＋5＋5＋3＝18。

4.【NOIP2007】(子集划分)将 n 个数{1，2，…，n}划分成 r 个子集。每个数都恰好属于一个子集，任何两个不同的子集没有共同的数，也没有空集。将不同划分方法的总数记为 S(n，r)。例如，S(4，2)＝7，这 7 种不同的划分方法依次为{(1)，(234)}，{(2)，(134)}，{(3)，(124)}，{(4)，(123)}，{(12)，(34)}，{(13)，(24)}，{(14)，(23)}。当 n＝6，r＝3 时，S(6，3)＝_____。(提示：先固定一个数，对于其余的 5 个数考虑 S(5，3)与 S(5，2)，再分这两种情况对原固定的数进行分析。)

【答案】90

【分析】

方法一：s(n，k)＝s(n－1，k－1)＋k＊s(n－1，k)，分别是 a_n 单独一堆，a_n 与其他一起。将 6 单独放到 1 个子集的方法数是 S(5，2)，即在其余的 2 个子集中放其余 5 个数。将 6 不单独放到 1 个子集的方法数是 3＊S(5，3)，即在 3 个子集里先放好其余 5 个数。然后将 6 依次放入 3 个子集中的某一个。即 S(6，3)＝S(5，2)＋3＊S(5，3)，类似可得

$$S(5,2)=S(4,1)+2*S(4,2),$$

$$S(5,3)=S(4,2)+3*S(4,3),$$

已知 S(4，2)＝7，容易看出，S(4，1)＝1，S(4，3)＝6(这时，只有一个子集中含有两个数，共 6 种可能，为(12)，(13)，(14)，(23)，(24)，(34))。

于是，S(5，2)＝1＋2＊7＝15，S(5，3)＝7＋3＊6＝25，S(6，3)＝15＋3＊25＝90。

S(n，k)通常称为第 2 类 Stirling 数，见《组合数学引论》，孙淑玲等著。

方法二：用递归思想做，很容易知道 S(n，1)＝1，为了利用这点，先分析 S(n－1，r)与 S(n－1，r－1)的情况，而 S(n，r)＝S(n－1，r)＊r＋S(n－1，r－1)，其具体情况分析如下：

假设已知 n－1 个球放置的情况数，如今要多加入一个球，有两种新情况：一是放入原有球的箱子，即情况数为 S(n－1，r)＊r(每种情况中放入任意箱子都是新情况，所以乘以箱子数 r)，二是取出所有球，让新加入的球独占一个箱子，则余下 r－1 个箱子可放球，有 n－1 个球可用来放，即情况数为 S(n－1，r－1)。

综上，得出 S(n，r)＝S(n－1，r)＊r＋S(n－1，r－1)。

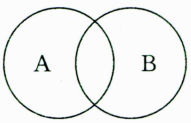
方法三：7 个球放入 4 个箱子无非是 2＋2＋2＋1 或者 3＋2＋1＋1 或者 4＋1＋1＋1 三种情况。所以分别求解再加起来为 C(7，1)＊C(6，2)＊C(4，2)＊C(2，2)/P(3，3)＋C(7，3)＊C(4，2)＋C(7，4)。

第4节 容斥原理

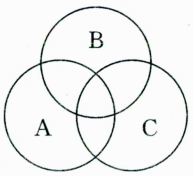
在计数时，我们必须注意无一重复，无一遗漏。为了使重叠部分不被重复计算，人们研究出一种新的计数方法，这种方法的基本思想是：先不考虑重叠的情况，把包含于某内容中所有对象的数目先计算出来，然后再把计数时重复计算的数目排斥出去，使得计算的结果既无遗漏又无重复，这种计数的方法称为容斥原理。

一、集合间的容斥关系

(一)两集合的容斥关系公式：$|A\cup B|=|A|+|B|-|A|\cap|B|$



(二)三个集合的容斥关系公式：$|A\cup B\cup C|=|A|+|B|+|C|-|A\cap B|-|A\cap C|-|B\cap C|+|A\cap B\cap C|$



(三)容斥原理

$$\left|\bigcup_{i=1}^{n}A_i\right|=\sum_{i=1}^{n}|A_i|-\sum_{i,j,i\neq j}|A_i\cap A_j|+\sum_{i,j,k,i\neq j\neq k}|A_i\cap A_j\cap A_k|-\cdots\pm|A_1\cap\cdots\cap A_n|$$

减2个的交　　加3个的交　　减4个的交……

$$\left|\bigcup_{i=1}^{n}s_i\right|=\sum_{i=1}^{n}s_i+(-1)^1*\sum_{1\leqslant i<j\leqslant n}|s_i\cap s_j|+(-1)^2\sum_{1\leqslant i<j<k\leqslant n}|s_i\cap s_j\cap s_k|+\cdots+(-1)^{n+1}*|s_1\cap\cdots\cap s_j|$$

例1：有多少能被3或5或7整除且小于1000的正整数？

分析：设A_1、A_2、A_3分别表示被3整除、被5整除、被7整除的数构成的集合，根据公式，

$$\left|\bigcup_{i=1}^{3}A_i\right|=\sum_{i=1}^{3}|A_i|-\sum_{i\neq j}|A_i\cap A_j|+\sum|A_1\cap A_2\cap A_3|$$

于是答案为

$$\left\lfloor\frac{1000}{3}\right\rfloor+\left\lfloor\frac{1000}{5}\right\rfloor+\left\lfloor\frac{1000}{7}\right\rfloor-\left\lfloor\frac{1000}{15}\right\rfloor-\left\lfloor\frac{1000}{21}\right\rfloor-\left\lfloor\frac{1000}{35}\right\rfloor+\left\lfloor\frac{1000}{105}\right\rfloor=543。$$

例2：在1到300的整数中，计算：

(1)能被3或被5或被7整除的数有多少个？

(2)不能被3、5、7中的任何一个整除的数有多少个？

(3)能被3整除，但不能被5、7中的任何一个整除的数有多少个？

(4)与300互质的数有多少个？

解：设$S=\{1,2,\cdots,300\}$，A_3、A_5、A_7分别表示S中3、5、7的倍数集，则$|A_3|=100$，$|A_5|=60$，$|A_7|=42$，$|A_3\cap A_5|=20$，$|A_3\cap A_7|=14$，$|A_5\cap A_7|=8$，$|A_3\cap A_5\cap A_7|=2$，于是(1)$|A_3\cup A_5\cup A_7|=162$；

(2)300−162=138;

(3)$|A_3|-|A_3\cap A_5|-|A_3\cap A_7|+|A_3\cap A_5\cap A_7|=100-20-14+2=68$;

(4)因为 $300=2^2\times3\times5^2$,所以 300−S 中 2,3,5 的倍数的个数即为所求。

即 $300-(\frac{300}{2}+\frac{300}{3}+\frac{300}{5})+(\frac{300}{2\times3}+\frac{300}{2\times5}+\frac{300}{3\times5})-\frac{300}{2\times3\times5}$

$=300(1-\frac{1}{2}-\frac{1}{3}-\frac{1}{5}+\frac{1}{2\times3}+\frac{1}{2\times5}+\frac{1}{3\times5}-\frac{1}{2\times3\times5})$

$=300(1-\frac{1}{2}-\frac{1}{3}+\frac{1}{2\times3}-\frac{1}{5}+\frac{1}{2\times5}+\frac{1}{3\times5}-\frac{1}{2\times3\times5})$

$=300[(1-\frac{1}{2}-\frac{1}{3}+\frac{1}{2\times3})-\frac{1}{5}(1-\frac{1}{2}-\frac{1}{3}+\frac{1}{2\times3})]$

$=300[(1-\frac{1}{2})(1-\frac{1}{3})-\frac{1}{5}(1-\frac{1}{2})(1-\frac{1}{3})]=300(1-\frac{1}{2})(1-\frac{1}{3})(1-\frac{1}{5})$。

二、错位排列问题

(一)错位排列的定义

对于一个 1……n 的任意排列,有一种情况是第 i 个位置上的数不是 i 对任意 i 都成立,例如 n=3 时,有且仅有排列 231 和 312 是满足要求的,这样的排列被称为错位排列。

现在的问题是,能否对于任意 n,给出错位排列的个数?

分析:利用容斥原理。设 A_i 表示 i 这个数在原来位置上的排列的集合,则错位排列的个数

$$D_n=n!-|\bigcup_{i=1}^{n}A_i|$$

现在我们来算 $|\bigcup_{i=1}^{n}A_i|$,先算 $\sum|A_i|$,因为有 1 个数在固定的位置上,有 C(n,1)种情况,对于每一种情况,剩下的 n−1 个数任意排列,于是有 C(n,1) * (n−1)!=n! 种情况。再算 $\sum|A_i\cap A_j|$,因为有 2 个数在固定的位置上,有 C(n,2)种情况,对于每一种情况,剩下 n−2 个数任意排列,于是有 C(n,2) * (n−2)!=n! /2 种情况。

一般地,因为有 k 个数在固定的位置上,另外(n−k)个数任意排列,所以

$$\sum|A_{i1}\cap A_{i2}\cap A_{is}\cap\cdots\cdots\cap A_{ik}|=\binom{n}{k}*(n-k)!=\frac{n!}{k!*(n-k)!}*(n-k)!=\frac{n!}{k!}$$

综上,由容斥原理,我们可以得到错位排列的公式

$$D_n=n!(1-\frac{1}{1!}+\frac{2}{1!}-\frac{1}{3!}+\cdots+(-1)^n*\frac{1}{n!})。$$

(二)错位排列的递推关系

n 个元素的错排记为 D(n),分为两个步骤:

1.把第 n 个元素放在一个位置,一共有 n−1 种方法,比如放在位置 k 处;

2.放编号为 k 的元素,这时有两种情况:(1)把它放到位置 n,那么剩下 n−2 个元素的错排就有 D(n−2)种方法;(2)第 k 个元素不把它放到位置 n,这时对于这 n−1 个元素的错排,有 D(n−1)种方法。

综上得 D(n)=(n−1) * [D(n−2)+D(n−1)]。特殊地,D(1)=0,D(2)=1。

课堂练习

1.【CSP-S/2019】一次期末考试，某班有15人数学得满分，有12人语文得满分，并且有4人语、数都是满分，那么这个班至少有一门得满分的同学有(　　)人。

A. 23　　B. 21　　C. 20　　D. 22

【答案】A

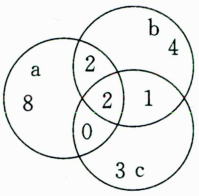
【分析】由容斥原理可以得知，答案为15+12−4=23。

2.【NOIP2015】在1和2015之间(包括1和2015在内)不能被4、5、6三个数任意一个数整除的数有_____个。

【答案】1075

【分析】本题采用容斥原理求解，首先不能被4、5、6三个数任意一个数整除的数=所有数−能被4、5、6三个数任意一个数整除的数，设f(i)表示n以内能被i整除的数的个数，f(i)=n/i。所以ans=2015−f(4)−f(5)−f(6)+f(20)+f(12)+f(30)−f(60)=1075。

3.【NOIP1998】某班有50名学生，每位学生发一张调查卡，上写a,b,c三本书的书名，将读过的书打√，结果统计数字如下：只读a者8人；只读b者4人；只读c者3人；全部读过的有2人；读过a,b两本书的有4人；读过a,c两本书的有2人；读过b,c两本书的有3人；(1)读过a的人数是_____；(2)一本书也没有读过的人数是_____。



【答案】(1)读过a的人数是12人。(2)一本书也没读过的人数是30人。

【分析】如图，可清晰的看到结果，比抽象推理更容易避免重复计算或丢失数据。读过a的人数是8+2+2=12，一本书也没有读过的人数是50−8−2−2−4−1−3=30。

4.【NOIP2002】在书架上放有编号为1,2,…,n的n本书。现将n本书全部取下然后再放回去，当放回去时要求每本书都不能放在原来的位置上。

例如，n=3时，原来位置为1　2　3；

放回去时只能为3　1　2　或　2　3　1　这两种。

问题：当n=5时满足以上条件的放法共有(　　)种。(不用列出每种放法)

【答案】44

【分析】此题要求的是错排数，详见上文介绍。

方法一：

```
F(N)=(N-1)*(F(N-1)+F(N-2));
F(1)=0;
F(2)=1;
F(3)=2*(0+1)=2;
F(4)=3*(2+1)=9;
F(5)=4*(F(4)+F(3))=44。
```

方法二：

总数S=N!(1!-1/1!+1/2!-1/3!……(-1)^n*1/N!)。

方法三：

f(0)=1;

f(x)=x*f(x-1)+1(x mod 2=0);

f(x)=x*f(x-1)-1(x mod 2=1);

所以 f(5)=5*f(4)-1=5*(4*f(3)+1)-1=5*(4*2+1)-1=44。

方法四：

5!-5*4!+10*3!-10*2!+5*1!-1=44。

5.【NOIP2015】重新排列 1234 使得每一个数字都不在原来的位置上，一共有_____种排法。

【答案】9

【分析】方法一：只有四个数，可以采用手工枚举的方式。

2341 2143 2413 3142 3412 3421 4123 4312 4321

因此答案为 9。

方法二：这是错排类问题。n 个不同元素的一个错排可由下述两个步骤完成：

第一步，“错排”1 号元素(将 1 号元素排在第 2 至第 n 个位置之一)，有 n－1 种方法。

第二步，“错排”其余 n－1 个元素，按如下顺序进行。视第一步的结果，若 1 号元素落在第 k 个位置，第二步就先把 k 号元素“错排”好，k 号元素的不同排法将导致两类不同的情况发生：1、k 号元素排在第 1 个位置，留下的 n－2 个元素在与它们的编号集相等的位置集上“错排”，有 f(n－2)种方法；2、k 号元素不排第 1 个位置，这时可将第 1 个位置“看成”第 k 个位置(也就是说本来准备放到 k 位置为元素，可以放到 1 位置中)，于是形成(包括 k 号元素在内的)n－1 个元素的“错排”，有 f(n－1)种方法。据加法原理，完成第二步共有 f(n－2)＋f(n－1)种方法。根据乘法原理，n 个不同元素的错排种数 f(n)＝(n－1)[f(n－2)＋f(n－1)](n＞2)。

假设有 n 封信，第一封信可放在(2～n)任一个信封里，共 n－1 种放法，设第一封信放在了第 k 个信封里，若此时第 k 封信放在了第 1 个信封里，则只要将剩下的 n－2 错排，即 f(n－2)，若第 k 封信没有放在了第 1 个信封里，可将第 1 封信的位置看成是“第 k 个位置”，即将 n－1 封信错排，即为 f(n－1)。由递推可得，f(n)＝(n－1)＊[f(n－1)＋f(n－2)]，f(1)＝0，f(2)＝1，f(3)＝2，f(4)＝9。

第5节 抽屉原理

抽屉原理源于如下一些问题所反映的基本事实：即“把10个苹果任意放入9个抽屉里，一定有一个抽屉里至少放有两个苹果”或者“7只鸽子飞进4个鸽巢，至少有两只鸽子飞进同一鸽巢”，因此抽屉原理又称鸽巢原理。抽屉原理更一般的描述为如果每一个抽屉代表一个集合，每一个苹果就可以代表一个元素。有n+1个元素放到n个集合中去，其中必定有一个集合里至少有两个元素。

一、第一抽屉原理

原理1：把n+1个物体放到n个抽屉里，则至少有一个抽屉里的东西不少于2件。

原理2：把mn+1(n不为0)个物体放到n个抽屉里面，则至少有一个抽屉里面有不少于(m+1)的物体。

原理3：把无穷多件物体放入n个抽屉，则至少有一个抽屉里有无穷个物体。

二、第二抽屉原理

把(mn−1)个物体放入n个抽屉中，其中必有一个抽屉不多于(m−1)个物体。

例如将3*5−1=14个物体放入5个抽屉中，则必定有一个抽屉中的物体数目不多于3−1=2。

总结：推广来说，把n个物体放进m个盒子，那么至少有一个盒子包含$\left\lceil \frac{n}{m} \right\rceil$个或更多的物体。

三、抽屉原理的简单应用

一位国际象棋大师有11周的时间备战一场锦标赛，他决定每天至少下一盘棋，但为了不使自己过于疲劳他还决定每周不能下棋超过12盘，证明：存在连续的若干天，期间这位大师恰好下了21盘棋。

证明：令a_1表示第1天所下的盘数，a_2表示前2天所下的盘数，a_3表示前3天所下的盘数……由于每天都要下至少一盘棋，故数列$a_1,a_2,a_3,\cdots,a_{77}$是一个严格递增的序列。此外，$a_1\geqslant 1$，而且由于每周下棋最多是12盘，即$a_{77}\leqslant 12*11=132$。因此，我们有

$$1\leqslant a_1<a_2<a_3<\cdots a_{77}\leqslant 132$$

此外，序列$a_1+21,a_2+21,\cdots,a_{77}+21$也是一个严格递增的序列：

$$22\leqslant a_1+21<a_2+21<a_3+21<\cdots<a_{77}+21\leqslant 153$$

于是，这154个数

$$a_1,a_3,a_3,\cdots,a_{77},a_1+21,a_2+21,a_3+21,\cdots\cdots,a_{77}+21$$

中的每一个都是1到153之间的一个整数。根据鸽巢原理可知，它们中间有两个是相等的。既然$a_1,a_2,a_3,\cdots,a_{77}$中没有相等的数，并且$a_1+21,a_2+21,a_3+21,\cdots,a_{77}+21$中也没有相等的数，因此必然存在一个i和一个j使得$a_i=a_j+21$。从而，这位国际象棋大师在第j+1，j+2，j+3，…，i天这段时间内一共下了21盘棋。

例题【NOIP2010】记T为一队列，初始时为空，现有n个总和不超过32的正整数依次入列。如果无论这些数具体为何值，都能找到一种出队的方式，使得存在某个时刻队列T中的数之和恰好为9，那么n的最小值是_____。

分析：组合数学题目的一种常见思路是先猜测答案再证明。试图通过鸽巢原理猜出答案。

设 S_i 表示前 i 个数的和，并规定 $S_0=0$，事实上原题要求出最小的 n，使得存在 $0\leqslant i<j\leqslant n$ 满足 $S_j=S_i+9$。于是我们可以把 S 数组看作鸽子，用不能同时取的一组（即差为 9）的集合构造笼子，构造方法如下：一共有 18 个集合按如下方式选取，{0,9}、{1,10}、{2,11}、{3,12}、{4,13}、{5,14}、{6,15}、{7,16}、{8,17}、{18,27}、{19,28}、{20,29}、{21,30}、{22,31}、{23,32}、{24}、{25}、{26}，根据题意，我们一旦在某个集合中取了两个元素，那么一定存在某个时刻队列 T 中数之和恰好为 9，于是由鸽巢原理我们可以知道 n=18（S 数组有 19 个元素）一定满足条件。

下面我们来证明 n=17 不可行，事实上只要构造出一组不合法的序列即可。可以选择如下不合法的序列：1　1　1　1　1　1　1　1　10　1　1　1　1　1　1　1　1。

综上，原问题所求 n 的最小值为 18。

课堂练习

1.【CSP-J/2019】一副纸牌除掉大小王有 52 张牌，四种花色，每种花色 13 张。假设从这 52 张牌中随机抽取 13 张纸牌，则至少（　　）张牌的花色一致。

A. 4　　B. 2　　C. 3　　D. 5

【答案】A

【分析】根据鸽巢原理，最平均的分法就是每种花色先有 3 张，此时最后一张无论是什么花色都会导致有 4 张牌花色一致。

2.【NOIP2012】如果平面上任取 n 个整点（横纵坐标都是整数），其中一定存在两个点，它们连线的中点也是整点，那么 n 至少是_____。

【答案】5

【分析】分析一下两个点的中点是整点有什么性质，(x_1,y_1)，(x_2,y_2) 的中点是 $\left(\frac{x_1+x_2}{2},\frac{y_1+y_2}{2}\right)$，要求中点是整点，所以 (x_1+x_2)，(y_1+y_2) 均是偶数。所以从整点坐标的奇偶性来讨论，只有（奇，偶），（奇，奇），（偶，奇），（偶，偶）四种情况，由抽屉原理，至少要 5 个点。

3.【NOIP2010】记 T 为一队列，初始时为空，现有 n 个总和不超过 32 的正整数依次入列。如果无论这些数具体为何值，都能找到一种出队的方式，使得存在某个时刻队列 T 中的数之和恰好为 9，那么 n 的最小值是_____。

【答案】18

【分析】本题可用抽屉原理求解。设 a_i 为各正整数值，则 T 的队列顺序为 $a_1,a_2,a_3,\cdots,a_n$，设 b_i 为前 i 项数之和（前缀和），则 $b_0=0,b_1=a_1,b_2=a_1+a_2,b_3=a_1+a_2+a_3,\cdots$ 如队列 T 中的数之和恰好为 9，实际上即是找到某个 b_j 和 b_i，使得 $b_j-b_i=9$。由题意可知 b_i 取值范围为 1－32，现将这 32 个数构造为集合{1,10}，{2,11}，…，{8,17}，{18,27}，{19,28}，…，{23,32}，{24}，{25}，{26}，这 17 个集合中的任一个集合不能包含两个或两个以上的，否则它们的差为 9。例如当 n=17 时，队列 T 为 11111111011111111，即 $b_1=1,b_2=2,\cdots b_8=8,b_9=18,b_{10}=19,b_{11}=20,\cdots,b_{17}=26$，它们中没有任意两个数是在同一集合内的，所以不存在数之和恰好等于 9。故根据抽屉原理可得，当 n=18 时，至少存在两个在同一个集合，即它们的差为 9。因此，答案为 n=18。

第6节 概率与期望

概率是反映随机事件出现的可能性大小。随机事件是指在相同条件下，可能出现也可能不出现的事件。例如从一批有正品和次品的商品中随意抽取一件，抽得的是正品就是一个随机事件。

设对某一随机现象进行了n次试验与观察，其中A事件出现了m次，即其出现的频率为m/n。经过大量反复试验，常有m/n越来越接近于某个确定的常数(此论断证明详见伯努利大数定律)，该常数即为事件A出现的概率，常用P(A)表示，其中$0 \leqslant P(A) \leqslant 1$。

一、几种特殊事件的概率

(一)等可能事件的概率，如果一次试验中共有n种等可能出现的结果，其中事件A包含的结果有m种，那么事件A的概率为$p(A)=\frac{m}{n}$。

(二)互斥事件：不可能同时发生的两个事件，叫作互斥事件，也叫不相容事件。如果事件$A_1,A_2,\cdots,A_n$彼此互斥，那么$A_1,A_2,\cdots,A_n$中至少有一个发生的概率为$p(A_1+A_2+\cdots+A_n)=p(A_1)+p(A_2)+\cdots+p(A_n)$。

(三)对立事件：事件A,B为互斥事件，且必有一个发生，则A,B叫对立事件，记A的对立事件为$\bar{A}$。由定义知$p(A)+p(\bar{A})=1$。

(四)相互独立事件：事件A(或B)是否发生对事件B(或A)发生的概率没有影响，这样的两个事件叫作相互独立事件。

相互独立事件同时发生的概率：两个相互独立事件同时发生的概率，等于每个事件发生的概率的积。即$p(A \cdot B)=p(A) \cdot p(B)$.若事件$A_1,A_2,\cdots,A_n$相互独立，那么这n个事件同时发生的概率为$p(A_1 \cdot A_2 \cdot \cdots \cdot A_n)=p(A_1) \cdot p(A_2) \cdot \cdots \cdot p(A_n)$。

(五)独立重复试验：若n次重复试验中，每次试验结果的概率都不依赖于其他各次试验的结果，则称这n次试验是独立的。

独立重复试验的概率：如果在一次试验中，某事件发生的概率为p，那么在n次独立重复试验中，这个事件恰好发生k次的概率为$p_n(k)=C_n^k \cdot p^k(1-p)^{n-k}$。

二、概率计算中的几个常用公式

(一)$P(A-B)=P(A)-P(AB)$，当$B \subseteq A$，即B包含在A中时，有$P(A-B)=P(A)-P(B)$，$P(A) \geqslant P(B)$。

(二)对任意两个事件，有$P(A \cup B)=P(A)+P(B)-P(AB)$。

(三)A_1、A_2为不相容事件，则$P(A_1 \cup A_2)=P(A_1)+P(A_2)$。

三、数学期望

数学期望又称均值，简称期望，是试验中所有可能结果的概率乘以其结果的总和，是最基本的数学特征之一，反映随机变量平均取值的大小。

需要注意的是，期望值并不一定等同于常识中的“期望”，“期望值”也许与每一个结果都不相等，是该变量输出值的平均数，并不一定包含于变量的输出值集合里。

设P(x)是一个离散概率分布，代表x发生的概率，自变量x的取值范围为$\{x_1,x_2,\cdots,x_n\}$。其期望被定义为

$$E(x)=\sum_{k=1}^{n} x_k P(x_k)。$$

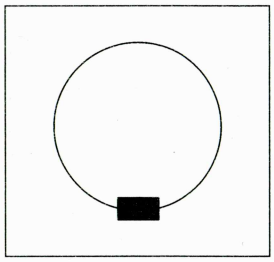
课堂练习

1.【NOIP2017】小明要去南美洲旅游，一共乘坐三趟航班才能到达目的地，其中第 1 个航班准点的概率是 0.9，第 2 个航班准点的概率为 0.8，第 3 个航班准点的概率为 0.9。如果存在第 i 个(i=1,2)航班晚点，第 i+1 个航班准点，则小明将赶不上第 i+1 个航班，旅行失败；除了这种情况，其他情况下旅行都能成功。请问小明此次旅行成功的概率是(　　)。

A. 0.5　　B. 0.648　　C. 0.72　　D. 0.74

【答案】D

【分析】统计各种情况的概率，然后加起来。也可以采用扣除法，即 1－(0.1＊0.8)－(0.2＊0.9)＝0.74。



2.【NOIP2017】欢乐喷球：儿童游乐场有个游戏叫“欢乐喷球”，正方形场地中心能不断喷出彩色乒乓球，以场地中心为圆心还有一个圆形轨道，轨道上有一列小火车在匀速运动，火车有六节车厢。假设乒乓球等概率落到正方形场地的每个地点，包括火车车厢。小朋友玩这个游戏时，只能坐在同一个火车车厢里，可以在自己的车厢里捡落在该车厢内的所有乒乓球，每个人每次游戏有三分钟时间，则一个小朋友独自玩一次游戏期望可以得到(　　)个乒乓球。假设乒乓球喷出的速度为 2 个/秒，每节车厢的面积是整个场地面积的 1/20。

A. 60　　B. 108　　C. 18　　D. 20

【答案】C

【分析】答案为 180 秒＊2 个/秒＊$\frac{1}{20}$＝18 个。

3.【NOIP2017】一家四口人，至少两个人生日属于同一月份的概率是(　　)(假定每个人生日属于每个月份的概率相同且不同人之间相互独立)。

A. 1/12　　B. 1/144　　C. 41/96　　D. 3/4

【答案】C

【分析】先计算四个人都在不同月份出生的概率为$\frac{P_{12}^4}{12^4}=\frac{55}{96}$，因此，至少两个人在同一月份出生的概率是$1-\frac{55}{96}=\frac{41}{96}$。

4.【NOIP2018】假设一台抽奖机中有红、蓝两色的球，任意时刻按下抽奖按钮，都会等概率获得红球或蓝球之一。有足够多的人每人都用这台抽奖机抽奖，假如他们的策略均为：抽中蓝球则继续抽球，抽中红球则停止。最后每个人都把自己获得的所有球放到一个大箱子里，最终大箱子里的红球与蓝球的比例接近于(　　)。

A. 1:2　　B. 2:1　　C. 1:3　　D. 1:1

【答案】D

【分析】设 x 为抓到红球前抓的蓝球的个数，E(x)表示抓到红球前抓的蓝球个数的期望。

x＝0,1,2,3,4,…,

$P(0)=\frac{1}{2}P(1)=\left(\frac{1}{2}\right)^2,P(2)=\left(\frac{1}{2}\right)^3,\cdots,P(n)=\left(\frac{1}{2}\right)^{n+1}$。

所以期望 $=0*P(0)+1*P(1)+2*P(2)+\cdots+n*P(n)=\sum_{i=0}^{n} i*\left(\frac{1}{2}\right)^{i+1}$。

当 n 趋向于正无穷时，$\sum_{i=0}^{\infty}\frac{1}{2^i}=1$，所以 E(x) 趋向于 1。

即红球前抓的蓝球的期望接近 1 个，所以红：蓝=1:1。

5.【NOIP2013】现有一只青蛙，初始时在 n 号荷叶上。当它某一时刻在 k 号荷叶上时，下一时刻将等概率地随机跳到 1,2,…,k 号荷叶之一上，直至跳到 1 号荷叶为止。当 n=2 时，平均一共跳 2 次；当 n=3 时，平均一共跳 2.5 次。则当 n=5 时，平均一共跳_____次。



【答案】37/12

【分析】假设在 $k(k>1)$ 号荷叶上，期望需要 $f(k)$ 次跳到 1 号荷叶。则

$$f(k)=1+\frac{1}{k}\sum_{i=1}^{k}f(i),$$

$$f(k)=\frac{k}{k-1}\left(1+\sum_{i=1}^{k-1}f(i)\right),$$

依次计算出 $f(1)=0,f(2)=2,f(3)=\frac{5}{2},f(4)=\frac{9}{2},f(5)=\frac{37}{12}$。

第7节　离散数学

一、逻辑运算

逻辑运算又称布尔运算。它是数字符号化的逻辑推演法，包括联合、相交、相减。布尔是19世纪最重要的数学家之一，出版了《逻辑的数学分析》，通过使用数学方法研究逻辑问题，成功地建立了逻辑运算。

进行逻辑运算，需要先掌握各种运算，注意运算符的优先级比较，做题时要细心。

（一）常用的逻辑运算符

1.! 逻辑非，是逻辑运算中的一种，就是指本来值的反值。比如我们定义一个布尔型变量a，它的初值为true，!a的值就变成了false。

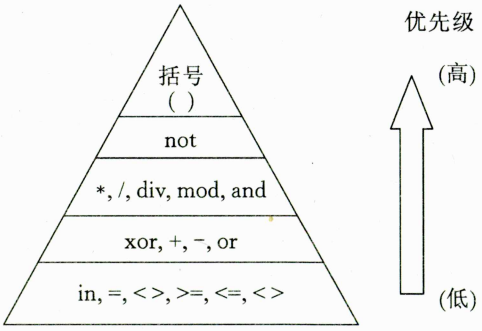
2.&& 逻辑与，是逻辑运算中的一种，双目运算，只有 && 左右两边的逻辑值全为true时，表达式的值才为true；

3.|| 逻辑或，是逻辑运算中的一种，双目运算，只要 || 左右两边的逻辑值有一个为true时，表达式的值就为true。

在初赛的时候，经常以选择题的形式出现，∧表示与，∨表示或，¬表示非。

（二）运算优先级比较

括号＞非＞与＞或、异或（或和异或是同级别的），如果加入加减乘除，就是以下这样：



注意：同级的运算符不分高低，计算时按照从左到右运算。

下面我们通过试题分析来讲解逻辑运算的应用。

例题1. 若A＝True，B＝False，C＝True，D＝False，以下逻辑运算表达式真的有（　　）。

A. (A∧B)∨(C∧D∨¬A)　　B. ((¬A∧B)∨C)∧¬B

C. (B∨C∨D)∨D∧A　　D. A∧(D∨¬C)∧B

题解： 一个个算结果，比如A选项(A∧B)∨(C∧D∨¬A)，根据运算级的比较，我们可以定下运算的顺序，然后按运算顺序计算结果。注意，这类题是有小技巧的。比如A选项可以先看中间的∨，为什么呢？因为∨的左右有一边是真就行，可以不去看另一边。

A选项的结果：(A∧B)∨(C∧D∨¬A)，(A∧B)＝假，(C∧D∨¬A)中C∧D＝假，¬A＝假，所以(C∧D∨¬A)＝假。于是A选项可以简写为假∨(假∨假)＝假。

B选项的结果：((¬A∧B)∨C)∧¬B，如果¬B是假那么就可以不去看前面的((¬A∧B)∨C)，可惜的是¬B是真，那么就要看((¬A∧B)∨C)，发现C是真，所以不看(¬A∧B)，于是B选项可以简写为(？∨真)∧真＝真。

C选项的结果：(B∨C∨D)∨D∧A，D∧A＝假，所以不得不看前面部分(B∨C∨D)，只

要 BCD 有一个是真，那么(B∨C∨D)＝真，而容易发现 C＝true。所以 C 选项可以简写为真∨假＝真。

D 选项的结果：A∧(D∨¬C)∧B，我们很容易发现 D 选项的特殊结构为？∧？∧？，三个？有一个是假，那么 D 为假，A 和 B 不用计算便可看出，所以先发现 B＝假，所以D＝假。

例题 2. 计算 23＋2|2&5＊3－6^5＝(　　)。

题解： 数字也有逻辑运算，当然也可以混合加减乘除。

这里举例说明运算的操作：

| &：22 & 5 | | |

3.【NOIP2010】以下逻辑表达式的值恒为真的是(　　)。

A. P∨(¬P∧Q)∨(¬P∧¬Q)　　B. Q∨(¬P∧Q)∨(P∧¬Q)

C. P∨Q∨(P∧¬Q)∨(¬P∧Q)　　D. P∨¬Q∨(P∧¬Q)∨(¬P∧¬Q)

【答案】A

【分析】B、C 中 P=Q=false,D 中 P=false,Q=true 时表达式为假,注意运算的优先级顺序为¬>∧>∨。

4.【NOIP2013】逻辑表达式(　　)的值与变量 A 的真假无关。

A. (A∨B)∧¬A　　B. (A∨B)∧¬B

C. (A∧B)∨(¬A∧B)　　D. (A∨B)∧¬A∧B

【答案】C

【分析】A 选项,当 A 为真的时候,此式恒为假,当 A 为假的时候,值取决于 B;B 选项,当 B 为真的时候,此式恒为假,当 B 为假的时候,值取决于 A;C 选项,当 B 为真的时候,此式恒为真,当 B 为假的时候,此式恒为假,结果与 A 的真假无关;D 选项,当 B 为真的时候,此式结果为¬A,与 A 有关,当 B 为假的时候,此式恒为假。

5.【NOIP2012】本题中,我们约定布尔表达式只能包含 p、q、r 三个布尔变量,以及“与”(∧)、“或”(∨)、“非”(¬)三种布尔运算。如果无论 p、q、r 如何取值,两个布尔表达式的值总是相同,则称它们等价。例如,(p∨q)∨r 和 p∨(q∨r)等价,p∨¬p 和 q∨¬q 也等价;而 p∨q 和 p∧q 不等价。那么,两两不等价的布尔表达式最多有_____个。

【答案】256

【分析】对于 p、q、r 三个变量,每个变量可取 0,1 两种取值,共有 8 种组合。对于每种组合,代入表达式只有 0 和 1 两种答案。因此两两不等价的表达式只有 $2^8=256$ 种。

我们可以通过构造来方便理解,对于每个元素的取值,我们可以得到 8 个子单元 p∧q∧r,¬p∧q∧r,p∧¬q∧r,p∧q∧¬r,¬p∧¬q∧r,¬p∧q∧¬r,p∧¬q∧¬r,¬p∧¬q∧¬r,对于 p、q、r 的每一个取值,我们有且仅有一个子单元为真,这方便了我们进行选择。对于每一个布尔表达式可以这么构造,从这 8 个子单元任意选一些元素,将它们用或连接,这样就可以构造出一个布尔表达式,而且任意两个选取元素不全相同的布尔表达式两两不等价,从这里我们能够很容易得出一共有$2^8=256$ 种两两不等价的表达式。

6.【NOIP2018】甲乙丙丁四人在考虑周末要不要外出郊游。

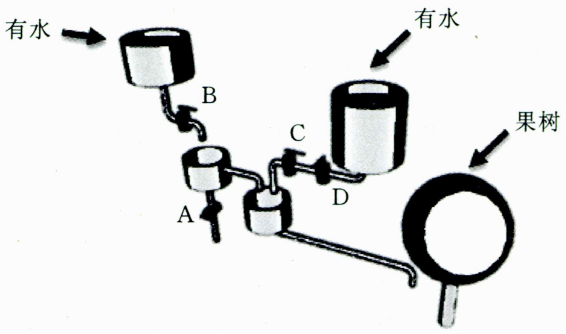
已知①如果周末下雨,并且乙不去,则甲一定不去;②如果乙去,则丁一定去;③如果丙去,则丁一定不去;④如果丁不去,而且甲不去,则丙一定不去。如果周末丙去了,则甲_____(去了/没去),乙_____(去了/没去),丁_____(去了/没去),周末_____(下雨/没下雨)。

【答案】去了,没去,没去,没下雨

【分析】考察逻辑推理基础知识。

首先丙去了,根据④得知甲去了或者丁去了,又根据③得知丁不去,所以甲一定去了。由于丁不去,根据②得知乙不去。由于甲去了,并且乙不去,根据①得周末没下雨。

7.【NOIP2016】下图表示一个果园灌溉系统,有 A、B、C、D 四个阀门,每个阀门可以打开或关上,所有管道粗细相同,以下设置阀门的方法中,可以让果树浇上水的是(　　)。



A. B 打开，其他都关上　　B. AB 都打开，CD 都关上

C. A 打开，其他都关上　　D. D 打开，其他都关上

【答案】A

【分析】简单分析可得，左侧的水流向果树的条件为 B 打开且 A 关闭，右侧的水流向果树的条件为 CD 同时打开。要使果树浇上水，至少一侧的水要流向果树，即 B 开 A 关，或者 CD 同时开。

8.【NOIP2001】在 a，b，c，d，e，f 六件物品中，按下面的条件能选出的物品是(　　)。

(1)a，b 两样至少有一样

(2)a，d 不能同时取

(3)a，e，f 中必须有 2 样

(4)b，c 要么都选，要么都不选

(5)c，d 两样中选一样

(6)若 d 不选，则 e 也不选

【答案】a，b，c，f

【分析】这是一个逻辑推理问题，硬性手动推导很容易出错，建议采用穷举法，这里一共有六件物品，每件物品都有“取”和“不取”两种情况，所以所有的可能性是 $2^6=64$ 种，每种试验一下并不复杂，答案是唯一的。

9.【NOIP2004】75 名儿童到游乐场去玩。他们可以骑旋转木马，坐滑行铁道，乘宇宙飞船。已知其中 20 人这三种东西都玩过，55 人至少玩过其中的两种。若每样乘坐一次的费用是 5 元，游乐场总共收入 700，可知有(　　)名儿童没有玩过其中任何一种。

【答案】10

【分析】集合的交并补。

3 种东西都玩过的共用去 $3\times5\times20=300$(元)，

只玩过两种东西的共用去 $2\times5\times(55-20)=350$(元)，

那么，只玩过一种东西的人数为 $(700-300-350)\div5=10$(人)，

所以，什么也没有玩的人数为 $75-55-10=10$(人)。

不定项选择题

1.【NOIP2008】设 A＝true，B＝false，C＝true，D＝false，以下逻辑运算表达式值为真的有（　　）。

A. (A∧B)∨(C∧D∨¬A)　　B. ((¬A∧B)∨C)∧¬D

C. (B∨C∨D)∨D∧A　　D. A∧(D∨¬C)∧B

【答案】BC

【分析】∧与，∨或，¬非。这里需要注意运算的优先级顺序：¬(非)＞∧(与)＞∨(或)。

2.【NOIP2011】在布尔逻辑中，逻辑“或”的性质有（　　）。

A. 交换律：P∨Q＝Q∨P

B. 结合律：P∨(Q∨R)＝(P∨Q)∨R

C. 幂等律：P∨P＝P

D. 有界律：P∨1＝1(1 表示逻辑真)

【答案】ABCD

【分析】代入 0 和 1 检验一下即可，∨是或运算。

3.【NOIP2014】若逻辑变量 A、C 为真，B、D 为假，以下逻辑运算表达式为真的有（　　）。

A. (B∨C∨D)∨D∧A　　B. ((¬A∧B)∨C)∧¬B

C. (A∧B)∨(C∧D∨¬A)　　D. A∧(D∨¬C)∧B

【答案】AB

【分析】考查基本的布尔运算。运算的优先级为¬＞∧＞∨。

4.【NOIP2012】逻辑异或(⊕)是一种二元运算，其真值表如下所示。

a	b	a⊕b
False	False	False
False	True	True
True	False	True
True	True	False

以下关于逻辑异或的性质，正确的有（　　）。

A. 交换律：a⊕b＝b⊕a

B. 结合律：(a⊕b)⊕c＝a⊕(b⊕c)

C. 关于逻辑与的分配律：a⊕(b∧c)＝(a⊕b)∧(a⊕c)

D. 关于逻辑或的分配律：a⊕(b∨c)＝(a⊕b)∨(a⊕c)

【答案】AB

【分析】代入计算即可发现 CD 错误。C 在 a＝T，b＝T，c＝F 时等式不成立，D 也是类似。

第8节 博弈论入门

有种很有意思的游戏，即有物体若干堆，可以是火柴棍或是围棋子等。两个人轮流从堆中取物体若干，规定最后取光物体者取胜。这是我国民间很古老的一个游戏，别看这游戏极其简单，却蕴含着深刻的数学原理。下面我们来分析一下要如何才能够取胜。

（一）最简单取石子游戏

只有一堆 n 个物品，两个人轮流从这堆物品中取物，规定每次至少取一个，最多取 m 个，最后取完者得胜。

显然，如果 n=m+1，那么由于一次最多只能取 m 个，所以，无论先取者拿走多少个，后取者都能够一次拿走剩余的物品，后者取胜。因此我们发现了如何取胜的法则：如果 n=(m+1)r+s，(r 为任意自然数，s≤m)，那么先取者要拿走 s 个物品；如果后取者拿走 k(k≤m)个，那么先取者再拿走 m+1−k 个，结果剩下(m+1)*(r−1)个，保持这样的取法，那么先取者肯定获胜。总之，要保持给对手留下(m+1)的倍数，最后就能获胜。即若 n=k(m+1)，则后取者胜，反之，先取者获胜。若 n%(m+1)==0，先取者必败。

这个游戏还可以有其他变相的玩法：两个人轮流报数，每次至少报一个，最多报十个，谁能到 100 者获胜；从一堆 100 个石子中取石子，最后取完的获胜。

（二）Nim 取石子游戏

有 n 堆石子，两个人轮流从某一堆取任意多的物品，规定每次至少取一个，多者不限，取走最后石子的人获胜。

这个问题就是最经典的 Nim 取石子问题。

令 C=A(1) xor A(2) xor A(3)… xor A(n)，设 A[i]代表第 i 堆的石子个数，若 C>0，则记为利己态，用 S 表示；若 C=0，则记为利他态，用 T 表示。

[证明]既然是 S 态，则此时 C>0，我们要使得 C 变为 0。

设 C 转化为二进制后，最高位的 1 是第 p 位。那么一定存在一个 A(t)的二进制最高位的 1 是第 p 位。(显然，C 的第 p 位不可能是 1)

然后，把第 t 堆石子的个数变为 x=A(t) xor C。因为 A(t)和 C 的二进制最高位的 1 是同一位。那么异或之后这一位就变成了 0，所以 x 一定小于 A(t)。

此时的 C'=A(1) xor A(2) xor … xor A(t) xor C xor A(t+1) xor … xor A(n)。把 C 代入，得到 C'=A(1) xor A(2) xor … xor A(n) xor A(1) xor A(2) xor … xor A(n)，由异或的性质可得 C'=0。因此，只要在第 t 堆石子中取出 A(t)−x 颗石子，就把 S 态变为了 T 态。

课堂练习

1.【NOIP2005】取火柴游戏的规则如下：一堆火柴有 N 根，A、B 两人轮流取出。每人每次可以取 1 根或 2 根，最先没有火柴可取的人为败方，另一方为胜方。如果先取者有必胜策略则记为 1，先取者没有必胜策略记为 0。当 N 分别为 100，200，300，400，500 时，先取者有无必胜策略的标记顺序为_____(回答应为一个由 0 或 1 组成的字符串)。

【答案】11011

【分析】经典的取石子问题，有结论当 N%3=0 时先手必败，否则必胜。当 N=100、200、400、500 时，A 都有必胜的把握，但是，当 N=300 时，A 无必胜的把握。

第四章　阅读程序

第 1 节　模拟策略

1.【NOIP2015】

```
01 #include<iostream>
02 using namespace std;
03 int main(){
04     int a,b,c;
05     a=1;b=2;c=3;
06     if (a>b){
07         if (a>c)
08             cout<<a<<' ';
09         else
10             cout<<b<<' ';
11     }
12     cout<<c<<endl;
13     return 0;
14 }
```

●判断题

(1)把第 1 行 iostream 改为 cstdio,程序仍然可以正常运行。　(　　)

(2)如果把 a 的初值改为−1,结果不会改变。　(　　)

(3)程序结果为 3。　(　　)

(4)如果把 c 的初值改为 5,结果不会改变。　(　　)

●选择题

(5)把 b 的的初值改为 4,结果为(　　)。

A. 1　　B. 2　　C. 3　　D. 4

(6)该程序的最坏时间复杂度为(　　)。

A. O(1)　　B. O(n)　　C. $O(n*m)$　　D. $O(2^n)$

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	√	√	×	C	A

【分析】主要考查 if 语句的应用,此程序中第一个 if 的条件不成立,因此不会进入内层的 if—else 语句。故不会输出 a 或者 b 的值,即输出只有一个 c 的值 3。

(1)只有包含 iostream 才可以使用 cin、cout。

(2)模拟即可得出答案。

(3)模拟即可得出答案。

(4)模拟即可得出答案。

(5)模拟即可得出答案。

(6)显然为常数时间复杂度。

2.【NOIP2013】

```
01 #include<iostream>
02 using namespace std;
03 int main(){
04     int a,b;
05     cin>>a>>b;
06     cout<<a<<"+"<<b<<"="<<a+b<<endl;
07 }
```

●判断题

(1)当 a,b 为负数且(a+b)在 int 范围内,输出的等式在数学上也成立。 ( )

(2)当输入为 0 0 时,输出为 0 + 0 = 0。 ( )

(3)将 03 行的 int 改为 signed 程序不会出错。 ( )

(4)当输出为-1+1=0 时候,输入可以为 2 -2。 ( )

●选择题

(5)输入为 114514 1919810 时,输出为( )。

A. 114514+1919810=2034324　　B. 114514+1919810=2034314

C. 114514+1919810=2024324　　D. 114514+1919810=1034324

(6)时间复杂度为( )。

A. O((a+b)ln(a-b))　　B. O(1)

C. $O\left(\frac{a^b}{w}\right)$　　D. $O(\sqrt{\log^b a})$

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	√	×	A	B

【分析】本题功能:输入 a、b,输出它们的和。

(1)输出的等式不合理,"-2+-2=-4"在数学上是不合法的。

(2)输出不会有空格。

(3)signed 等效替代于 signed int,main 的类型也可以为 signed,程序不会出错。

(4)输入为 2-2 时,输出为 2+-2=0,显然和要求的输出不同。

(5)因为两个数都没有超过 int 范围,相加后的值 2034324 也没有超过,故答案为 A。

(6)没有执行什么其他操作,时间复杂度为 O(1)。

3.【NOIP2014】

```
01 #include<iostream>
02 using namespace std;
03 int main(){
04     int a,b,c,d,ans;
05     cin>>a>>b>>c;
06     d=a-b;
07     a=d+c;
```

```
08      ans=a*b;
09      cout<<"Ans="<<ans<<endl;
10      return 0;
11 }
```

●判断题

(1)ans 一定为 c 的倍数。　　(　　)

(2)输入为 8 8 8,输出为 64。　　(　　)

(3)将 04 行移动到 02 03 行之间,程序不会发生错误。　　(　　)

(4)将 09 行的 ans 改为 a * b,输出结果不会发生改变。　　(　　)

●选择题

(5)输入为 2 3 4 时,答案为(　　)。

A. Ans = 9　　B. Ans=9　　C. A n s = 9　　D. 9

(6)将 06 的“－”改为“＋”,(5)的输出为(　　)。

A. Ans=27　　B. Ans=36　　C. Ans=54　　D. Ans=81

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	√	√	B	A

【分析】

(1)ans 和 c 没有倍数关系。

(2)输出应该为 Ans=64。

(3)定义变量放在 main 的外面作为全局变量也可行。

(4)确实不会发生改变。

(5)答案显然为 Ans=9,选 B。

(6)d=2+3=5,a=5+4=9,a * b=27,故输出 Ans=27。

4.【NOIP2012】

```
01 #include<iostream>
02 using namespace std;
03 int a,b,c,d,e,ans;
04 int main(){
05      cin>>a>>b>>c;
06      d=a+b;
07      e=b+c;
08      ans=d+e;
09      cout<<ans<<endl;
10 }
```

●判断题

(1)当输入为 11 45 14 时,输出为 70。　　(　　)

(2)输入的数都为负数时,输出可能为正数。　　(　　)

(3)输入的数都为正数时,输出可能为负数。　　(　　)

(4)将 ans 的类型改为 char,输出结果不会改变。　　(　　)

●选择题

(5)输入为 19 19 810 时,输出为(　　)。

A. 867　　B. 857　　C. 967　　D. 767

(6)输出为 28 时,输入可以为(　　)。

A. 5 8 7　　B. 1 5 7　　C. 1 1 25　　D. 1 1 4

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	√	√	×	A	AC

【分析】

(1)11+45+45+14≠70

(2)有可能为正数,因为可能会溢出。

(3)有可能为负数,理由同上。

(4)ans 是 char 类型时,输出结果是计算结果数字对应的字符,将不会输出正确的答案。

(5)19+19+19+810=867。

(6)一个一个枚举即可,1+1+1+25=28。

5.【NOIP2011】

```
01 #include<iostream>
02 using namespace std;
03 int main(){
04     int i,n,m,ans;
05     cin>>n>>m;
06     i=n;
07     ans=0;
08     while (i<=m){
09         ans+=i;
10         i++;
11     }
12     cout<<ans<<endl;
13     return 0;
14 }
```

●判断题

(1)删去第 7 行,运行结果不变。　　(　　)

(2)将第 8 行的<=改为<,输出减小 n。　　(　　)

(3)可以实现一个复杂度为 O(1)的代码,效果与上述代码等价。　　(　　)

(4)当 m<n 时,程序不会运行错误。　　(　　)

●选择题

(5)输入 10 20,输出(　　)。

A. 5　　B. 165　　C. 20　　D. 10

(6)时间复杂度为(　　)。

A. O(max{m−n,0})　　B. O(n)

C. $O(n^2)$　　D. O(mlogn)

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	√	√	B	A

【分析】这个程序就是计算 n 一直加到 m 的和。

(1)ans 没赋初值 0，系统会自动赋一个不一定为 0 的初值，运行结果会发生变化。

(2)计算时不包含 m，故输出减小 m。

(3)由分析知，上述代码实现了等差数列求和，可以利用公式在 O(1)内解决。

(4)m<n 不进行循环，能正常运行。

(5)10 一直加到 20 的和为 165。

(6)若 n>m，不执行循环，否则循环了 m−n+1 次，复杂度是 O(m−n)级别。

6.【NOIP2015】

```
01 #include<iostream>
02 using namespace std;
03 struct point {
04     int x;
05     int y;
06 };
07 int main(){
08     struct EX {
09         int a;
10         int b;
11         point c;
12     }e;
13     e.a=1;
14     e.b=2;
15     e.c.x=e.a+e.b;
16     e.c.y=e.a*e.b;
17     cout<<e.c.x<<','<<e.c.y<<endl;
18     return 0;
19 }
```

●判断题

(1)输出结果为 3 2。　(　　)

(2)将 08～12 行移到 06～07 行之间，程序运行结果不会发生改变。　(　　)

(3)将 01 行的 iostream 改为 cstdio 程序运行结果不会发生改变。　(　　)

(4)程序的时间复杂度为 O(1)。　(　　)

●选择题

(5)该程序会输出(　　)个数。

A. 1　　B. 2　　C. 3　　D. 0

(6)将 13 行改为 e.a=4;，输出将为(　　)。

A. 6,8　　B. 6,7　　C. 7,8　　D. 8,6

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	√	×	√	B	A

【分析】主要考查结构体的应用。直接按照题目意思模拟过程即可。e.c.x=e.a+e.b=1+2=3;e.c.y=e.a*e.b=1*2=2。因此输出为3,2(不含引号,注意不要漏掉逗号或写成空格)。

(1)输出应为3,2。

(2)结构体可以定义在main函数外面。

(3)cout需要iostream库头。

(4)时间复杂度确实是O(1)。

(5)输出两个数。

(6)手动模拟即可,答案为6,8。

7.【NOIP2009】

```
01 #include<iostream>
02 using namespace std;
03 int main()
04 {
05     int a[3],b[3];
06     int i,j,tmp;
07     for (i=0;i<3;i++)
08         cin>>b[i];
09     for (i=0;i<3;i++)
10     {
11         a[i]=0;
12         for (j=0;j<=i;j++)
13         {
14             a[i]+=b[j];
15             b[a[i]%3]+=a[j];
16         }
17     }
18     tmp=1;
19     for (i=0;i<3;i++)
20     {
21         a[i]%=10;
22         b[i]%=10;
23         tmp*=a[i]+b[i];
24     }
25     cout<<tmp<<endl;
26     return 0;
27 }
```

●判断题

(1)输入的3个数越大,则输出的结果也越大。　　(　　)

(2)执行完第17行后,b[i]比a[i]相对应的值要大。　　(　　)

(3)将第18行的tmp=1改为tmp=0,不论输入什么数据,输出结果都是0。　　(　　)

(4)若输入数据中包含字母,如3 3 a则程序会出错。　　(　　)

●选择题

(5)输入2 3 5,输出的结果是(　　)。

A. 414　　B. 415　　C. 416　　D. 417

(6)输入1 1 1,输出的结果是(　　)。

A. 36　　B. 48　　C. 72　　D. 108

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	√	×	C	C

【分析】本题全面考查模拟策略的应用,程序功能:

(1)输入0 0 1时和输入0 0 2时答案均为0。

(2)输入0 0 1,执行完第17行后,a[2]=b[2]=1。

(3)0乘任何数都为0。

(4)字母以整型读入会转化为其对应的ASCII码。

(5)直接模拟即可。

(6)直接模拟即可。

8.【NOIP2009】

```
01 #include<iostream>
02 using namespace std;
03 const int c=2009;
04 int main()
05 {
06     int n,p,s,i,j,t;
07     cin>>n>>p;
08     s=0;t=1;
09     for (i=1;i<=n;i++)
10     {
11         t=t*p%c;
12         for (j=1;j<=i;j++)
13             s=(s+t)%c;
14     }
15     cout<<s<<endl;
16     return 0;
17 }
```

●判断题

(1)将12—13行改为s=(s+1ll*t*i)% c;,程序输出不会改变。　　(　　)

(2)将 15 行改为 printf("%d\n",s);程序输出不会改变。 (　　)

(3)将 08 行的 s=0;去掉,程序输出不会改变。 (　　)

(4)将 03 行 const 去掉,程序输出不会发生变化。 (　　)

●选择题

(5)输入为 11 2,输出为(　　)。

A. 782　　B. 762　　C. 802　　D. 114

(6)该算法的时间复杂度为(　　)。

A. O(1)　　B. O(n)　　C. $O(n^2)$　　D. O(nlogn)

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	√	×	√	A	C

【分析】

(1)12—13 行要执行的就是 s=(s+111*t*i)%c;这么一个操作。

(2)在 C++14 的新标准库下,iostream 包含 printf。

(3)s 值初始不一定为 0。

(4)去掉 const 后 c 的值也为常量。

(5)直接模拟,可以得到三个变量值的变化过程。

i	1	2	3	4	5	6	7	8	9	10	11
t	2	4	8	16	32	64	128	256	512	1024	39
s	2	10	34	98	258	642	1538	1577	158	353	782

(6)显然时间复杂度为 $O(n^2)$。

9.【NOIP2008】

```
01 #include<iostream>
02 using namespace std;
03 int main()
04 {
05     int i,a,b,c,d,f[4];
06     for (i=0;i<4;i++)cin>>f[i];
07     a=f[0]+f[1]+f[2]+f[3];
08     a=a/f[0];
09     b=f[0]+f[2]+f[3];
10     b=b/a;
11     c=(b*f[1]+a)/f[2];
12     d=f[(b/c)%4];
13     if (f[(a+b+c+d)%4]>f[2])
14         cout<<a+b<<endl;
15     else
16         cout<<c+d<<endl;
17     return 0;
```

```
18 }
```

●判断题

(1)将 05 行移到 02 03 之间,程序不会出错。　　(　　)

(2)将 13 行的“>”改为“>=”,输出不会发生改变。　　(　　)

(3)将 10 行改为 b/=a;输出不会发生改变。　　(　　)

(4)输出只会有一行。　　(　　)

●选择题

(5)当输入为 9 19 29 39,输出为(　　)。

A. 21　　B. 22　　C. 23　　D. 24

(6)时间复杂度为(　　)。

A. O(1)　　B. O(a)　　C. O(aloga)　　D. $O(a^{loga})$

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	√	√	√	C	A

【分析】

(1)变量定义可以定义在 main 函数的外面。

(2)程序运行输出没有发生改变。

(3)b/=a 和 b=b/a 的操作都是一样的。

(4)显然只会输出一个整数。

(5)本程序直接进行模拟。过程为

```
a=f[0]+f[1]+f[2]+f[3]=96
a=a/f[0]=10
b=f[0]+f[2]+f[3]=77
b=b/a=7
c=(b*f[1]+a)/f[2]=143/29=4
d=f[(b/c)%4]=f[1]=19
f[(a+b+c+d)%4]=f[0]<f[2]
```

c+d=4+19=23。

(6)显然时间复杂度为 O(1)。

10.【NOIP2014】

```
01 #include<iostream>
02 using namespace std;
03 const int SIZE=100;
04 int alive[SIZE];
05 int n;
06 int next (int num){
07     do {
08         num++;
```

```
09          if (num>n)num=1;
10      } while (alive[num]==0);
11      return num;
12  }
13  int main(){
14      int m,i,j,num;
15      cin>>n>>m;
16      for (i=1;i<=n;i++)
17          alive[i]=1;
18      num=1;
19      for (i=1;i<=n;i++){
20          for (j=1;j<m;j++)
21              num=next(num);
22          cout<<num<<" ";
23          alive[num]=0;
24          if (i<n)num=next(num);
25      }
26      cout<<endl;
27      return 0;
28  }
```

●判断题

(1)若输入 100 0,该程序会运行错误。 (　　)

(2)若把 17 行去掉,程序结果不会发生改变。 (　　)

(3)如果把 24 行的"<"改为"<=",则程序结果发生改变。 (　　)

(4)若输入 11 3,则输出 3 6 9 1 5 10 4 11 8 2 7。 (　　)

●选择题

(5)该程序的最坏时间复杂度为(　　)。

A. O(1)　　B. O(n)　　C. O(n * m)　　D. (2^n)

(6)若输入为 10 4,则第 4 个数为(　　)。

A. 5　　B. 6　　C. 7　　D. 8

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	×	√	√	C	C

【分析】此题为经典的约瑟夫问题。由于数据范围较小,直接模拟即可。

(1)alive 数组大小为 100,n 必须小于 100。

(2)alive 显然需要初始化为 1。

(3)更改后 num 显然答案会改变。

(4)根据题意模拟即可。

(5)程序中有双重循环,时间复杂度显然。

(6)根据题意模拟即可。

11.【NOIP2017】

```
01 #include<iostream>
02 using namespace std;
03 int main(){
04     int n,i,j,x,y,nx,ny;
05     int a[40][40];
06     for (i=0;i<40;i++)
07         for (j=0;j<40;j++)
08             a[i][j]=0;
09     cin>>n;
10     y=0;x=n-1;
11     n=2*n-1;
12     for (i=1;i<=n*n;i++){
13         a[y][x]=i;
14         ny=(y-1+n)%n;
15         nx=(x+1)%n;
16         if ((y==0&&x==n-1)||a[ny][nx]!=0)
17             y=y+1;
18         else {y=ny;x=nx;}
19     }
20     for (j=0;j<n;j++)
21         cout<<a[0][j]<<" ";
22     cout<<endl;
23     return 0;
24 }
```

●判断题

(1)11 行改为 n=(n<<1)-1;程序运行结果不变。　　(　　)

(2)输出的数字有 n(n 是最初输入的数值)个。　　(　　)

(3)将 12 行的 i++改为++i 程序运行结果不变。　　(　　)

(4)将 06～08 行去掉,程序运行结果不变。　　(　　)

●选择题

(5)输入为 3 时,输出为(　　)。

A. 17 24 1 8 15　　B. 17 21 3 5 2　　C. 1 5 36 7 5　　D. 17 2 8 3 1

(6)输入为 4 时,输出为(　　)。

A. 30 39 48 1 10 19 28　　B. 24 39 48 1 10 19 28

C. 30 39 48 1 5 19 14　　D. 30 39 24 1 10 19 28

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	×	√	×	A	A

【分析】由于 n 比较小,直接模拟即可。其实这是一个幻方,看出来的直接就可以手动写出方案,看不出来的按照它给的规则写:第一行中间是 1,下一个数写在上一个数的右上格(第

一行的上一行是最后一行，最后一列的右面是第一列），如果右上格已经填过就填它右上的下面那个格（能填右上填右上，填不了右上就填右侧那个）。

（1）n * 2 等价于 n<<1。

（2）输出的数有（n*2-1）个。

（3）for 循环中 i++和++i 等价。

（4）去掉 06～08 行后数组 a 定义的初始值不一定为 0。

（5）见解析，模拟即可。

（6）见解析，模拟即可。

第 2 节　字符处理

1.【NOIP2018】

```
01 #include<cstdio>
02 char st[100];
03 int main(){
04     scanf("%s",st);
05     for (int i=0;st[i];++i){
06         if ('A'<=st[i]&&st[i]<='Z')
07             st[i]+=1;
08     }
09     printf("%s\n",st);
10     return 0;
11 }
```

●判断题

(1)第 5 行的循环将执行 n 次,n 为字符串长度。　(　　)

(2)输入的字符串必须由大写字母组成。　(　　)

(3)输出最多含有 25 种大写字母。　(　　)

(4)若输入的字符串由 n 种不同字符组成,则输出由 n 种不同字符组成。　(　　)

●选择题

(5)输入 QuanGuo,则输出的结果是(　　)。

A. RuanHuo　　B. quanguo　　C. QuanGuo　　D. PuanFuo

(6)如果将第 07 行的 st[i]+=1;改成 st[i]-=1;,则第(5)题的输出结果为(　　)。

A. QUANGUO　　B. PuanFuo　　C. quanguo　　D. QvboGvp

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	×	√	×	A	B

【分析】阅读程序可知,程序先读入字符串,再把字符串中每个大写字母的 ASCII 码值加一。

(1)易得循环会经过 st[i]不为空的点,即其长度。

(2)输入的字符串包含小写字母等且不影响运行。

(3)每个大写字母的 ASCII 码值将加一,“A”全变为“B”,但无字符能变为“A”。

(4)反例如 s1="Z[",s2="[[",“[”为“Z”的 ASCII 码+1。

(5)把每个大写字母的 ASCII 码值加一,模拟即可。

(6)把每个大写字母的 ASCII 码值减一,模拟即可。

2.【NOIP2016】

```
01 #include<iostream>
02 using namespace std;
03 int main(){
04     int i,length1,length2;
05     string s1,s2;
```

```
06    cin>>s1>>s2;
07    length1=s1.size();
08    length2=s2.size();
09    for (i=0;i<length1;i++)
10        if (s1[i]>='a'&&s1[i]<='z')
11            s1[i]-='a'-'A';
12    for (i=0;i<length2;i++)
13        if (s2[i]>='a'&&s2[i]<='z')
14            s2[i]-='a'-'A';
15    if (s1==s2)
16        cout<<"="<<endl;
17    else if (s1>s2)
18        cout<<">"<<endl;
19    else
20        cout<<"<"<<endl;
21    return 0;
22 }
```

●判断题

(1)第 6 行输入的字符串如果包括数字、各类符号,可能运行错误。 (　　)

(2)若 length1<length2,输出为“<”。 (　　)

(3)若去掉第 10、13 行,输出结果不变。 (　　)

(4)若 s1 和 s2 中的各字符互不相同,则输出一定不为“=”。 (　　)

●选择题

(5)输入 Welcome 与 Chinese,count 的结果是(　　)。

A.<　　B. =　　C.>　　D. 0

(6)若字符串的长度为 n,则算法的时间复杂度是(　　)。

A. O(n)　　B. $O(n\log^2 n)$

C. $O(n^2)$　　D. O(nlogn)

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	×	×	C	A

【分析】分析程序可知,此程序的功能是忽略大小写的情况下比较两个字符串的大小。显然两个字符串在忽略大小写的情况下是相等的,因此输出等于号“=”。

(1)string 可存储任意字符串。

(2)比较字符串大小应先比较第一位字母,而非长度。

(3)去掉将多修改非小写字母,答案可能变化。

(4)s1='ab',s2='AB',输出为“=”。

(5)模拟得到结果为“>”。

(6)题中仅有一层循环。

3.【NOIP2014】

```
01 #include<iostream>
02 #include<string>
03 using namespace std;
04 int main(){
05     string st;
06     int i,len;
07     getline(cin,st);
08     len=st.size();
09     for (i=0;i<len;i++){
10         if (st[i]>='a'&&st[i]<='z')
11             st[i]=st[i]-'a'+'A';
12     }
13     cout<<st<<endl;
14     return 0;
15 }
```

●判断题

(1)输入的字符串可以是任意字符,包括字母、数字、各类符号甚至中文汉字。　(　　)

(2)如果去掉第 10 行,输出结果不变。　(　　)

(3)输出结果可以包含小写字母。　(　　)

(4)算法时间复杂度为 O(1)。　(　　)

●选择题

(5)输入 Hello,输出的结果是(　　)。

A. HELLO　　B. hello　　C. Hello　　D. ello

(6)输出的结果不可能是(　　)。

A. WELCOME　　B. WELCOME−1　　C. Welcome　　D. ELCOME

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	×	×	×	A	C

【分析】本题可以采用分析程序功能的方法,此程序的功能:读入一个字符串,将字符串里的所有小写字母转化为相应的大写字母后输出。注意不要写错字母,漏掉标点符号以及空格。

(1)仅需寻找小写字母,求对应的大写字母,其他不变,字母可以是符号等。

(2)去掉后,非小写字母也会变动,导致答案改变。

(3)易知小写字母均化为大写字母,故输出无小写字母。

(4)易知有一执行 len 次的循环,复杂度为 O(len)。

(5)依题目模拟即可。

(6)输出结果不可以包含小写字母。

4.【NOIP2008】

```
01 #include<iostream>
02 #include<cstring>
03 using namespace std;
```

```
04 int i,j,len;
05 char s[50];
06
07 int main()
08 {
09     cin>>s;
10     len=strlen(s);
11     for (i=0;i<len;++i)
12     {
13         if (s[i]>='A'&&s[i]<='Z')s[i]-='A'-'a';
14     }
15     for (i=0;i<len;++i)
16     {
17         if (s[i]<'x')s[i]+=3;
18         else s[i]+=-23;
19     }
20     cout<<s<<'/';
21     for (j=1;j<4;j++)
22     {
23         for (i=0;i<len-j;i=i+j)
24         {
25             s[i]=s[i+j];
26         }
27     }
28     cout<<s<<endl;
29     return 0;
30 }
```

●判断题

(1)输入的字符串长度为 n,则输出的字符串长度为 2n。 (　　)

(2)如果去掉第 18 行,输出结果不变。 (　　)

(3)若输入仅有字母,输出结果可以包含大写字母。 (　　)

(4)算法时间复杂度为 O(1)。 (　　)

●选择题

(5)输入 FGuvw,输出的结果是(　　)。

A. ijxxz/zxxzz　　B. ijxyz/xxzzz　　C. ijxyz/ixzzz　　D. ijxyz/zxzzz

(6)输出的结果不可能是(　　)。

A. fmw/www　　B. qpp/ppp　　C. vgo/ooo　　D. nouuuu

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	×	×	D	D

【分析】本题是关于字符串的操作，也比较简单。首先是将字符串全部变为小写字母，然后进行替换，替换的规则是往后移3位，即a→d，b→e，c→f，…，x→a，y→b，z→c，第二部分也是简单的模拟。

(1)答案为两个长len的字符串及中间相隔的'/'，长度应为2len+1。

(2)去掉后，满足s[i]≥x小写字母无法变动，导致答案改变。

(3)易知大写字母均化为小写字母，故输出无大写字母。

(4)易知复杂度为O(len)。

(5)依题目模拟即可。

(6)输出结果应该包含'/'。

5.【NOIP2015】

```
01 #include<iostream>
02 #include<string>
03 using namespace std;
04 int main(){
05     string str;
06     int i;
07     int count;
08     count=0;
09     getline (cin,str);
10     for (i=0;i<str.length();i++){
11         if (str[i]>='a'&&str[i]<='z')
12             count++;
13     }
14     cout<<"Ithas"<<count<<"lowercases"<<endl;
15     return 0;
16 }
```

●判断题

(1)第7行输入的字符串可以是任意字符，包括字母、数字、各类符号甚至中文汉字及符号。（　　）

(2)执行完第10行循环后，count的值可能为0。（　　）

(3)若去掉11行，输出结果不变。（　　）

(4)若输入的字符串中各字符互不相同，则count不为0。（　　）

●选择题

(5)输入CSP2001，count的结果是(　　)。

A. 3　　B. 7　　C. 8　　D. 0

(6)若字符串的长度为n，算法的时间复杂度是(　　)。

A. O(n)　　B. O(nlogn)　　C. $O(n^2)$　　D. $O(n\log^2 n)$

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	√	×	×	D	A

【分析】这个程序统计了字符串中小写字母的个数。注意不要遗漏提示信息或写错提示信息。

(1)string 可存储任意字符串。

(2)字符串为 AB,无小写字母。

(3)将多统计非小写字母导致答案变大。

(4)字符串为 AB,count=0。

(5)易得字符串中无小写字母,答案为 0。

(6)题中仅有一层循环,被执行 n 次 O(1)。

6.【NOIP2009】

```
01 #include<iostream>
02 using namespace std;
03 const int maxn=50;
04 void getnext(char str[])
05 {
06     int l=strlen(str),i,j,k,temp;
07     k=l-2;
08     while (k>=0&&str[k]>str[k+1])k--;
09     i=k+1;
10     while (i<l&&str[i]>str[k])i++;
11     temp=str[k];
12     str[k]=str[i-1];
13     str[i-1]=temp;
14     for (i=l-1;i>k;i--)
15         for (j=k+1;j<i;j++)
16             if (str[j]>str[j+1])
17             {
18                 temp=str[j];
19                 str[j]=str[j+1];
20                 str[j+1]=temp;
21             }
22     return;
23 }
24 int main()
25 {
26     char a[maxn];
27     int n;
28     cin>>a>>n;
29     while (n>0)
30     {
31         getnext(a);
32         n--;
```

```
33      }
34      cout<<a<<endl;
35      return 0;
36 }
```

●判断题

(1)每次循环找到的 k 一定不同。　　(　　)

(2)如果把第 18～20 行换成 swap(str[j],str[j+1]),输出结果不变。　　(　　)

(3)去掉第 32 行会导致程序超时。　　(　　)

(4)输入的字符串可包含任意字母。　　(　　)

●选择题

(5)输入 NOIP3,输出的结果是(　　)。

A. NPOI　　B. NPIO　　C. NIPO　　D. PONI

(6)输入 CSP2021 2,输出的结果是(　　)。

A. CPS2120　　B. CSP2120　　C. CPS2210　　D. CSP2021

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	√	√	√	A	B

【分析】 getnext()中 k 开始用 while 找位置,有交换、字符移动等操作。

(1)易证当 n>strlen(a)时,一定有不同的 k 存在。

(2)第 18～20 行即交换 str[j],str[j+1],等价于 swap(str[j],str[j+1])。

(3)去掉第 32 行 n 将无变化,第 29 行的循环会一直死循环。

(4)找位置时,有交换、字符移动时仅需比较大小,什么字母无影响。

(5)依题目模拟即可。

(6)依题目模拟即可。

7.【NOIP2010】

```
01 #include<iostream>
02 #include<string>
03 using namespace std;
04 int main()
05 {
06     string s;
07     char m1,m2;
08     int i;
09     getline(cin,s);
10     m1='';
11     m2='';
12     for (i=0;i<s.length();i++)
13         if (s[i]>m1)
14         {
15             m2=m1;
```

```
16              m1=s[i];
17          }
18          else if (s[i]>m2)
19              m2=s[i];
20          cout<<m1<<''<<m2<<endl;
21          return 0;
22 }
```

【提示】

字符	空格	'0'	'A'	'a'
ASCII 码	32	48	65	97

●判断题

(1)去掉第 10、11 行,程序会运行错误。 ( )

(2)去掉第 18 行的 else,程序会运行错误。 ( )

(3)将 12 行替换为"for (i=s.length()-1;i>=0;i--)",运行结果不变。 ( )

(4)将 18 行的"s[i]>m2"替换为"s[i]>=m2",运行结果不变。 ( )

●选择题

(5)当输入为 AABBCCDD 时,输出为( )。

A. B A　　B. C B　　C. D D　　D. D A

(6)当输入的字符串为大写字符串时,输出不可能为( )。

A. Z B　　B. A B　　C. T B　　D. G A

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	√	√	C	B

【分析】程序的功能是:寻找出现在一个字符串中,字典序严格最大与严格次大的字符。

(1)虽然局部变量没有赋值可能会导致答案错误,但是不导致运行错误。

(2)去掉 else 后,该 if 语句依旧成立,可能导致答案错误,不导致运行错误。

(3)将循环顺序颠倒后,每个元素依然都被考虑,所以运行结果不变。

(4)替换后,严格次大的字符将会变成非严格次大的字符。

(5)由该程序的功能可知最大字符为 D,次大字符为 C。

(6)显然次大字符不能比最大字符大,则 B 是错的。

8.【NOIP2011】

```
01 #include<iostream>
02 #include<string>
03 using namespace std;
04 int main(){
05     string map="22233344455566677778889999";
06     string tel;
07     int i;
08     cin>>tel;
```

```
09      for (i=0;i<tel.length();i++)
10          if ((tel[i]>='0')&&(tel[i]<='9'))
11              cout<<tel[i];
12          else
13          if ((tel[i]>='A')&&(tel[i]<='Z'))
14              cout<<map[tel[i]-'A'];
15          cout<<endl;
16          return 0;
17 }
```

●判断题

(1)第 7 行输入的字符串可以是任意字符,包括字母、数字、各类符号甚至中文汉字及符号。（　　）

(2)如果去掉第 13 行,程序无运行错误。（　　）

(3)输出结果可以包含大写字母。（　　）

(4)输出的字符串至多包含 8 种不同数字（　　）

●选择题

(5)输入 CCF－NOIP－2011,输出的结果是(　　)。

A. 22366472011　　B. 223－6647－2011　　C. 22366462011　　D. 22366482011

(6)输入 WELCOME－CSP,输出的结果是(　　)。

A. 9352663277　　B. 9342663277　　C. 9342663278　　D. 9342663377

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	√	×	×	A	A

【分析】观察题目中给出的 map 子串,我们可以发现题目要求输出在九键输入法下的每个数字或字母对应的按键。

(1)仅需寻找数字与字母,求对应的按键,字母可以是符号等。

(2)去掉后,tel[i]－'A'可能数组越界。

(3)易知输出为一串数字串。

(4)字母对应 8 种数字,数字对应 10 种数字,所以至多包含 10 种。

(5)依题目模拟即可。

(6)依题目模拟即可。

9.【NOIP2012】

```
01 #include<iostream>
02 #include<string>
03 using namespace std;
04 int n,ans,i,j;
05 string s;
06 char get(int i){
07     if (i<n)
08         return s[i];
```

```
09      else
10          return s[i-n];
11  }
12  int main(){
13      cin>>s;
14      n=s.size();
15      ans=0;
16      for (i=1;i<=n-1;i++){
17          for (j=0;j<=n-1;j++)
18              if (get(i+j)<get(ans+j)){
19                  ans=i;break;
20              }else if (get(i+j)>get(ans+j))
21                  break;
22      }
23      for (j=0;j<=n-1;j++)
24          cout<<get(ans+j);
25      cout<<endl;
26  }
```

●判断题

(1)删除第 15 行程序,运行结果不会发生变化。 (　　)

(2)将第 16 行中的 i=1 改为 i=0,运行结果不变。 (　　)

(3)在 17、18 行中间插入“if (get(i+j)==get(ans+j)) continue;”,运行结果不变。 (　　)

(4)将 21 行的 break 换成 continue,因为这是 for 语句的最后一句,所以运行结果不变。 (　　)

●选择题

(5)当输入为 ABCDEFG 时,输出结果为(　　)。

A. ABCDEFG　　B. GFEDCBA　　C. ACEGFDB　　D. AGBFCED

(6)当输入为 CBBADADA 时,输出结果为(　　)。

A. ABABCDAD　　B. ABBCDADA　　C. ACBBADAD　　D. ADADACBB

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	√	√	×	A	C

【分析】这个程序实际是在暴力求解字符串的最小表示法,把这个字符串看成一个环,然后找一个从某个位置开始的长度为 n 的字典序最小的字符串。

(1)全局变量 int 不赋值,系统默认赋值为 0。

(2)int i=0 会一直运行,直到不满足 j<=n-1。

(3)因为下面的 if (get(i+j)<get(ans+j))和 else if (get(i+j)>get(ans+j))中的get(i+j)和 get(ans+j)都不同,所以当两者相等时就可以跳出本次循环了。

(4)continue 是跳出本次循环,而 break 是跳出当前 for 语句,两者不同。

(5)因为是递增,实际上都不符合 if (get(i+j)<get(ans+j)),该条件原样输出。

(6)应该输出 ACBBADAD。

a[i]	C	B	B	A	D	A	D	A
i	0	1	2	3	4	5	6	7
ans	0	1	2	3	3	5	5	7

10.【NOIP2016】

```
01 #include<iostream>
02 using namespace std;
03 int main(){
04     char a[100][100],b[100][100];
05     string c[100];
06     string tmp;
07     int n,i=0,j=0,k=0,total_len[100],length[100][3];
08     cin>>n;
09     getline(cin,tmp);
10     for (i=0;i<n;i++){
11         getline(cin,c[i]);
12         total_len[i]=c[i].size();
13     }
14     for (i=0;i<n;i++){
15         j=0;
16         while (c[i][j]!=':'){
17             a[i][k]=c[i][j];
18             k=k+1;
19             j++;
20         }
21         length[i][1]=k-1;
22         a[i][k]=0;
23         k=0;
24         for (j=j+1;j<total_len[i];j++){
25             b[i][k]=c[i][j];
26             k=k+1;
27         }
28         length[i][2]=k-1;
29         b[i][k]=0;
30         k=0;
31     }
32     for (i=0;i<n;i++){
33         if (length[i][1]>=length[i][2])
34             cout<<"NO,";
35         else {
36             k=0;
```

```
37              for (j=0;j<length[i][2];j++){
38                  if (a[i][k]==b[i][j])
39                      k=k+1;
40                  if (k>length[i][1])
41                      break;
42              }
43              if (j==length[i][2])
44                  cout<<"NO,";
45              else
46                  cout<<"YES,";
47          }
48      }
49      cout<<endl;
50      return 0;
51  }
```

●判断题

(1)若输入的字符串不包括':',可能运行错误。 (　　)

(2)若 n=1 且 length[i][1]<length[i][2],输出为"YES,"。 (　　)

(3)若去掉第 40、41 行,输出结果不变。 (　　)

(4)输出结果为一行字符串。 (　　)

●选择题

(5)输入 1 AB:ACDEbFBkBD,输出的结果是(　　)。

A. YES,　　B. NO,　　C. 运行错误　　D. 运行超时

(6)输入 1 AR:ACDBrT,输出的结果是(　　)。

A. YES,　　B. NO,　　C. 运行错误　　D. 运行超时

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	×	√	√	A	B

【分析】分析程序功能可知,此程序的功能是判断左边的字符序列是否为右边字符序列的子序列。

(1)由 16 行知,无':'循环会一直进行,直至数组越界。

(2)当 length[i][1]<length[i][2]时,左边的字符序列不一定为右边字符序列的子序列。

(3)因为当 k>length[i][1]时,k=length[i][1]+1,a[i][k]为空,不等于 b[i][j],k 不会变化。

(4)只换行 1 次,应为 1 行字符串。

(5)左边的字符序列是右边字符序列的子序列。

(6)左边的字符序列不是右边字符序列的子序列。

11.【NOIP2018】

```
01 #include<iostream>
02 using namespace std;
03 string s;
04 long long magic(int l,int r){
05     long long ans=0;
06         for (int i=l;i<=r;++i){
07           ans=ans*4+s[i]-'a'+1;
08     }
09     return ans;
10 }
11 int main(){
12     cin>>s;
13     int len=s.length();
14     int ans=0;
15     for (int l1=0;l1<len;++l1){
16         for (int r1=l1;r1<len;++r1){
17             bool bo=true;
18             for (int l2=0;l2<len;++l2){
19                 for (int r2=l2;r2<len;++r2){
20                     if (magic(l1,r1)==magic(l2,r2)
21                         &&(l1!=l2||r1!=r2))
22                             bo=false;
23                 }
24             }
25
26             if (bo){
27                 ans+=1;
28             }
29         }
30     }
31     cout<<ans<<endl;
32     return 0;
33 }
```

●判断题

(1)输出一定不为0。（　　）

(2)将15、16行与18、19行对调，答案不变。（　　）

(3)输入字符串最多包含5种字符串，若更多则容易互相冲突。（　　）

(4)调用第12行需要iostream库头。（　　）

●选择题

(5)输入abacaba，则输出的结果是(　　)。

A. 16　　B. 17　　C. 18　　D. 19

(6)输入 abcbbcc,则输出的结果是(　　)。

A. 16　　B. 20　　C. 18　　D. 19

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	√	×	√	A	B

【分析】magic 本质是通过 hash 算法生成 hash 值来判断字符串是否相同,所以分析可知这段代码本质是求字符串中仅出现过一次的子串个数,答案为 16 个。

(1)字符串本身就仅出现一次。

(2)由于两个两重循环都枚举一个区间,对调无妨。

(3)由 magic 函数得最多 4 种。

(4)C++常识,正确。

(5)统计仅出现过一次的子串个数即可。

(6)统计仅出现过一次的子串个数即可。

第 3 节　枚举算法

1.【NOIP2013】

```
01 #include<iostream>
02 #include<string>
03 using namespace std;
04 int main(){
05     string str;
06     cin>>str;
07     int n=str.size();
08     bool isPlalindrome=true;
09     for (int i=0;i<n/2;i++){
10         if (str[i]!=str[n-i-1]){
11             isPlalindrome=false;
12         }
13     }
14     if (isPlalindrome)
15         cout<<"Yes"<<endl;
16     else
17         cout<<"No"<<endl;
18     return 0;
19 }
```

●判断题

(1)如果去掉第 18 行,程序不能正常运行。　(　　)

(2)如果去掉第 8 行的初始化,程序可能得不到正确答案。　(　　)

(3)在 11 行下添加一行 break;,程序运行结果不变。　(　　)

(4)如果输入 abceecba,输出 Yes。　(　　)

●选择题

(5)程序的时间复杂度为(　　)。

A. O(n)　　B. O(logn)　　C. O(nlogn)　　D. O(n^2)

(6)输入 abcdefghijklmnmlkjihgfedcba,输出为(　　)。

A. No　　B. Yes　　C. N0　　D. YES

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	√	√	√	A	B

【分析】本题功能:输入一行字符串,判断该字符串是否能成为回文串。

(1)C++标准规定,main 函数若不加返回值,则编译器默认加上 return 0;。

(2)若没有初始化,则 isPlalindrome 初始值不确定,可能得不到正确答案。

(3)显然程序运行结果不变。

(4)根据题意分析即可。

(5)根据程序分析即可。

(6)根据题意分析即可。

2.【NOIP2017】

```
01 #include<iostream>
02 using namespace std;
03 int main(){
04     int t[256];
05     string s;
06     int i;
07     cin>>s;
08     for (i=0;i<256;i++)
09         t[i]=0;
10     for (i=0;i<s.length();i++)
11         t[s[i]]++;
12     for (i=0;i<s.length();i++)
13         if (t[s[i]]==1){
14             cout<<s[i]<<endl;
15             return 0;
16         }
17     cout<<"no"<<endl;
18     return 0;
19 }
```

●判断题

(1)第7行输入的字符串s可以是任意字符,包括字母、数字、各类符号甚至中文汉字及符号。(　　)

(2)若去掉8~9行,输出结果不变。(　　)

(3)若答案为no,则说明字符串s中的每个字符出现次数都大于1。(　　)

(4)若输入的字符串中各字符互不相同,则输出结果为no。(　　)

●选择题

(5)若输入的字符为yzywYZYW,则输出为(　　)。

A. y　　B. z　　C. w　　D. W

(6)若输入的字符为YZYWyzyw,则输出为(　　)。

A. Y　　B. Z　　C. W　　D. w

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	√	×	B	B

【分析】采用分析程序功能的方法来分析此题可得,此程序先统计输入字符串中每个字母出现的次数,然后求这个字符串中第一个仅出现一次的字母。

(1)中文字符储存时对应的ASCII码为负数,会导致访问负数下标。

(2)t数组定义在主函数内,初值不一定为0。

(3)输出no说明找不到出现次数为1的字符,说明在s中出现的字符出现次数一定大

于 1。

(4)输出结果明显为第一个字符。

(5)直接寻找该字符串第一个仅出现一次的字符即可。

(6)直接寻找该字符串第一个仅出现一次的字符即可。

3.【NOIP2017】

```
#include<iostream>
using namespace std;
int main(){
    string ch;
    int a[200];
    int b[200];
    int n,i,t,res;
    cin>>ch;
    n=ch.length();
    for (i=0;i<200;i++)
        b[i]=0;
    for (i=1;i<=n;i++){
        a[i]=ch[i-1]-'0';
        b[i]=b[i-1]+a[i];
    }
    res=b[n];
    t=0;
    for (i=n;i>0;i--){
        if (a[i]==0)
            t++;
        if (b[i-1]+t<res)
            res=b[i-1]+t;
    }
    cout<<res<<endl;
    return 0;
}
```

●判断题

(1)输入的字符串长度必须小于等于 200,否则可能会出现运行时错误。　(　　)

(2)若输入字符串为 10011010110011011010111110001,则会输出 12。　(　　)

(3)去掉第 10 行和第 11 行,程序一定可以正常运行。　(　　)

(4)输入的字符串必须由 01 字符组成。　(　　)

●选择题

(5)程序的时间复杂度为(　　)。

A. O(1)　　B. O(nlogn)　　C. O(n)　　D. $O(n^2)$

(6)若输入 10104050104010901090801 00,输出为(　　)。

A. 0　　B. 5　　C. 26　　D. 13

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	×	×	×	C	D

【分析】此题直接模拟较为复杂，可以采用分析程序功能的方法来解决。该程序将读进来的字符串每一个位置的数值转化为01存在a数组里，而b数组的意义是a数组的前缀和，b[i]也可以理解为“位置i前面共有多少个1”，随后倒序枚举，每次遇到0就将t加一，也就是说，t的意义是“位置i之后有多少个0”。所以此程序的功能是在一个0、1串中找一个位置，使得该位置前面1的个数加上该位置后面0的个数最大，最后输出最大的值。

(1)题目中的数组大小为200，显然正确。

(2)根据上面的分析模拟即可。

(3)去掉后，b数组没有初始化，答案不能保证。

(4)输入字符可以由数字组成。

(5)显然时间复杂度为O(n)。

(6)此时b[i]表示前i个数字的和，于是程序的功能变成在一个数字串中找一个位置，使得该位置前面数字的和加上该位置后面0的个数最小，模拟即可。

4.【NOIP2010】

```
01 #include<iostream>
02 using namespace std;
03 int main(){
04     const int SIZE=10;
05     int data[SIZE],i,j,cnt,n,m;
06     cin>>n>>m;
07     for (i=1;i<=n;i++)
08         cin>>data[i];
09     for (i=1;i<=n;i++){
10         cnt=0;
11         for (j=1;j<=n;j++)
12             if ((data[i]<data[j])||(data[j]==data[i]&&j<i))
13                 cnt++;
14         if (cnt==m)
15             cout<<data[i]<<endl;
16     }
17     return 0;
18 }
```

●判断题

(1)若输入的n大于等于10，则程序可能发生运行时错误。 (　　)

(2)若输入的m大于等于n，则程序没有输出。 (　　)

(3)若把第14行的==换成>=，则程序运行结果不变。 (　　)

(4)输入5 2 96 −8 0 16 87，则会输出17。 (　　)

●选择题

(5)输入 6 3 1 5 3 7 8 4,则输出为(　　)。

A. 1　　B. 3　　C. 4　　D. 5

(6)程序的时间复杂度为(　　)。

A. O(n)　　B. O(logn)　　C. O(nlogn)　　D. $O(n^2)$

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	√	×	×	C	D

【分析】本题求在序列 data 中,哪些数之前有 m 个数不比它小。因此答案是 16。本题也可以进行模拟。

(1)data 数组的大小为 10,所以 n 要小于 10,否则运行时会错误。

(2)易知第 i 个数最多只有 i−1 个数不比它大且在它前面,显然 m 大于等于 n 时没有输出。

(3)如果更改,则满足条件的数有更多。

(4)根据题意模拟即可。

(5)根据题意模拟即可。

(6)阅读程序可以看到从 1 到 n 的双重循环,显然时间复杂度为 $O(n^2)$。

5.【NOIP2015】

```
01 #include<iostream>
02 #include<string>
03 using namespace std;
04 int main(){
05     int len,maxlen;
06     string s,ss;
07     maxlen=0;
08     do {
09         cin>>ss;
10         len=ss.length();
11         if (ss[0]=='#')break;
12         if (len>maxlen){
13             s=ss;
14             maxlen=len;
15         }
16     } while(true);
17     cout<<s<<endl;
18     return 0;
19 }
```

●判断题

(1)输出可以包含# 。　　(　　)

(2)如果去掉第7行的初始化,程序可能得不到正确答案。 ( )

(3)输出一定有字符。 ( )

(4)如果把第12行的>改为>=,程序结果不会改变。 ( )

●选择题

(5)程序的时间复杂度级别为( )。

A. 线性 B. 对数 C. 常数 D. 平方

(6)输入I am a citizen of China # ,输出( )。

A. am B. a C. citizen D. China

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	√	×	×	A	C

【分析】本题可以采用分析程序功能的方法,简单分析可知,该程序实现的是输出长度最长的串,如果有多个,输出最前面的那一个。

(1)如果#不是作为每个字符串的开头出现,则不会退出程序。

(2)若无初始化,则maxlen的值不确定,可能无法得到正确答案。

(3)如果输入只有#,则答案为空串。

(4)若改为>=,则后面长度相等的串也会被替换。

(5)根据程序显然为线性。

(6)根据程序模拟可以得出。

6.【NOIP2018】

```
01 #include<cstdio>
02 int main(){
03     int x;
04     scanf("%d",&x);
05     int res=0;
06     for (int i=0;i<x;++i){
07         if (i*i%x==1){
08             ++res;
09         }
10     }
11     printf("%d",res);
12     return 0;
13 }
```

●判断题

(1)如果把第11行的"%d"改为"%lld",程序可以正常输出答案。 ( )

(2)删去12行的语句,运行结果将发生改变。 ( )

(3)如果输入2147483648,程序输出4。 ( )

(4)程序输入的数字小于等于40360时,正确性可以保证。 ( )

●选择题

(5)如果输入100,程序输出( )。

A. 20　　B. 4　　C. 0　　D. 9

(6)程序的时间复杂度为(　　)。

A. O(x)　　B. O(1)　　C. O(xlogx)　　D. $O(x^2)$

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	×	√	B	A

【分析】

(1)更改后,printf 会多读取 4B 的数据,输出错误。

(2)return 0;语句并不影响输出结果。

(3)输入 2147483648 后,x 会溢出,易知答案不是 4。

(4)当 x 小于等于 40360 时,i 的平方不会超出 int 的范围,正确性可以保证。

(5)根据题意模拟即可。

(6)只有一重 x 的循环。

7.【NOIP2016】

```
01 #include<iostream>
02 using namespace std;
03 int main(){
04     int Max,Min,sum,count=0;
05     int tmp;
06     cin>>tmp;
07     if (tmp==0)
08         return 0;
09     Max=Min=sum=tmp;
10     count++;
11     while (tmp!=0){
12         cin>>tmp;
13         if (tmp!=0){
14             sum+=tmp;
15             count++;
16             if (tmp>Max)
17                 Max=tmp;
18             if (tmp<Min)
19                 Min=tmp;
20         }
21     }
22     cout<<Max<<","<<Min<<","<<sum/count<<endl;
23     return 0;
24 }
```

●判断题

(1)程序一定会输出3个整数并用逗号隔开。 ( )

(2)若把Max,Min,sum的数据类型改为double,则输出结果会改变。 ( )

(3)输入1 2 0 3 4 5 0 7 0时,count值最终为8。 ( )

(4)程序的时间复杂度瓶颈在于第22行计算答案过程。 ( )

●选择题

(5)程序的时间复杂度为( )。

A. O(1) B. O(count) C. O(∞) D. O(tmp)

(6)若输入为1 9 2 8 3 7 4 6 0 5,则会输出( )。

A. 9,1,5 B. 9,1,6 C. 9,1,7 D. 9,1,8

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	√	×	×	C	A

【分析】可以采用分析功能的方法。此程序会依次读入若干整数,并以0作为结束标志。程序求出了这个整数序列中的最大值、最小值以及平均数(向下取整),注意程序读到0时停止,忽略0以及后面的输入,输出请注意不要漏掉逗号,或者把逗号打成空格。

(1)当第一个输入为0时,没有输出。

(2)更改后,sum除以count不会取整,答案改变。

(3)程序读入第一个0时退出,此时count为2。

(4)程序的时间复杂度主要是和第11行至19行的循环有关。

(5)若tmp≠0恒成立,则程序永不停止,复杂度→∞。

(6)根据题意和分析模拟即可。

8.【NOIP2012】

```
01 #include<iostream>
02 using namespace std;
03 int n,i,temp,sum,a[100];
04 int main(){
05     cin>>n;
06     for (i=1;i<=n;i++)
07         cin>>a[i];
08     for (i=1;i<=n-1;i++)
09         if (a[i]>a[i+1]){
10             temp=a[i];
11             a[i]=a[i+1];
12             a[i+1]=temp;
13         }
14     for (i=n;i>=2;i--)
15         if (a[i]<a[i-1]){
16             temp=a[i];
```

```
17              a[i]=a[i-1];
18              a[i-1]=temp;
19          }
20      sum=0;
21      for (i=2;i<=n-1;i++)
22          sum+=a[i];
23      cout<<sum/(n-2)<<endl;
24      return 0;
25  }
```

●判断题

(1)输入的 n 小于 100,否则可能会出现运行时错误。　　(　　)

(2)第 10 行到第 12 行实现了交换的功能。　　(　　)

(3)把 sum 的数据类型改为 double,程序运行结果不会改变。　　(　　)

(4)当 n 为 2 时,程序可以正常运行。　　(　　)

●选择题

(5)程序的时间复杂度为(　　)。

A. O(1)　　B. O(nlogn)　　C. O(n)　　D. $O(n^2)$

(6)14～19 行的代码段功能为(　　)。

A. 排序整个序列　　B. 把序列中的最小值放到序列开头

C. 把序列中的最小值放到序列末尾　　D. 把序列中的最大值放到序列末尾

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	√	×	×	C	B

【分析】去除一个最大值和最小值后的平均值——向下取整,程序中通过两次比较,将最大值和最小值放到了尾和首。

(1)题目中的数组大小为 100,输入 100 会导致数组越界。

(2)根据代码显然可以得出。

(3)更改后,sum 除以 n－2 不会取整,答案改变。

(4)n 为 2 时,sum 除以 0,发生运行时错误。

(5)显然时间复杂度为 O(n)。

(6)这段代码倒序枚举,如果当前的数比前一个数来得小,那么就交换两个数,不难发现,当前枚举到的位置的数一定是最小的,当枚举到 2 时,要么第一个数原本就是最小的,要么第二个数最小,且会发生一次交换,故选 B。

第4节 排序算法

1.【NOIP2010】

```
01 #include<iostream>
02 using namespace std;
03 void swap(int &a,int &b)
04 {
05     int t;
06     t=a;
07     a=b;
08     b=t;
09 }
10 int main()
11 {
12     int a1,a2,a3,x;
13     cin>>a1>>a2>>a3;
14     if (a1>a2)
15         swap(a1,a2);
16     if (a2>a3)
17         swap(a2,a3);
18     if (a1>a2)
19         swap(a1,a2);
20     cin>>x;
21     if (x<a2)
22         if (x<a1)
23             cout<<x<<''<<a1<<''<<a2<<''<<a3<<endl;
24         else
25             cout<<a1<<''<<x<<''<<a2<<''<<a3<<endl;
26     else
27         if (x<a3)
28             cout<<a1<<''<<a2<<''<<x<<''<<a3<<endl;
29         else
30             cout<<a1<<''<<a2<<''<<a3<<''<<x<<endl;
31     return 0;
32 }
```

●判断题

(1)去掉第14行到第19行,不会影响程序的运行结果。 ( )

(2)将第一行的iostream改成cstdio会编译错误。 ( )

(3)本题输出结果和将四个数从小到大排序一样。 ( )

(4)如果输入91 2 20\n77,输出为2 20 77 91。 ( )

●选择题

(5)如果输入114514 191 810\n258,程序输出( )。

A. 114 514 258 1919810　　B. 114514 191 810\n258

C. 114514 191 810 258　　D. 191 258 810 114514

(6)如果输入 1 1 1\n2,程序输出(　　)。

A. 1 1 1 2　　B. 1 1 2 1　　C. 1 2 1 1　　D. 2 1 1 1

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	√	√	√	D	A

【分析】在读入 x 之前,通过 3 次比较将 a1,a2,a3 从小到大排好序,然后将 x 插入适当位置,也就是将输入的四个数按照从小到大排序后输出。

(1)根据分析可得,删掉这几行会影响程序结果。

(2)cin 和 cout 的使用需要调用 iostream 库。

(3)见上述分析。

(4)见上述分析,将这四个数排序好就是这个样子。

(5)同上。

(6)同上。

2.【NOIP2010】

```
01 #include<iostream>
02 using namespace std;
03 int main()
04 {
05     const int SIZE=100;
06     int na,nb,a[SIZE],b[SIZE],i,j,k;
07     cin>>na;
08     for (i=1;i<=na;i++)
09         cin>>a[i];
10     cin>>nb;
11     for (i=1;i<=nb;i++)
12         cin>>b[i];
13     i=1;
14     j=1;
15     while ((i<=na)&&(j<=nb)){
16         if (a[i]<=b[j]){
17             cout<<a[i]<<'';
18             i++;
19         }
20         else {
21             cout<<b[j]<<'';
22             j++;
23         }
24     }
```

```
25      if (i<=na)
26          for (k=i;k<=na;k++)
27              cout<<a[k]<<' ';
28      if (j<=nb)
29          for (k=j;k<=nb;k++)
30              cout<<b[k]<<' ';
31      return 0;
32  }
```

●判断题

(1)保证 a 数组和 b 数组有序,输出的序列一定是一个不降序列。 (　　)

(2)如果输入 0 0,不会输出数。 (　　)

(3)如果删掉第 13 行和第 14 行不影响程序结果。 (　　)

(4)使用 C++98 不会 CE。 (　　)

●选择题

(5)该程序时间复杂度是(　　)。

A. O(na+nb)　　B. O(max{na,nb}logmax{na,nb})

C. O(na · nb)　　D. O(na^{nb})

(6)如果输入 5\n1 3 5 7 9\n4\n2 6 10 14,输出(　　)。

A. 1 2 3 5 6 7 9 10 14　　B. 14 10 9 7 6 5 3 2 1

C. 1 3 5 7 9 2 6 10 14　　D. 5 1 3 5 7 9 4 2 6 10 14

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	√	×	√	A	A

【分析】本程序是对给出的两列数进行输入,运用的是归并排序中合并的方法(但是只是在两个数组本身有序时,最后的序列才有序)。本题数据不大,模拟也可以解得答案。

(1)见分析。

(2)按程序模拟即可。

(3)影响了合并。

(4)没有出现 C++98 之后版本的特性。

(5)显然循环次数是 O(na+nb)的,所以复杂度如此。

(6)见分析。

3.【NOIP2011】

```
01 #include<iostream>
02 #include<cstring>
03 using namespace std;
04 const int SIZE=100;
05 int main(){
06     int n,i,sum,x,a[SIZE];
07     cin>>n;
08     memset(a,0,sizeof(a));
```

```
09      for (i=1;i<=n;i++){
10          cin>>x;
11          a[x]++;
12      }
13      i=0;sum=0;
14      while (sum<(n/2+1)){
15          i++;
16          sum+=a[i];
17      }
18      cout<<i<<endl;
19      return 0;
20  }
```

●判断题

(1)当第 8 行的 sizeof(a)改为 SIZE 时,运行结果不会发生改变。　　(　　)

(2)当第 11 行的 a[x]++改成++a[x]时,运行结果不会发生改变。　　(　　)

(3)当第 14 行的 sum<(n/2+1)改为 sum<(n+1)/2 时,运行结果不会发生改变。　　(　　)

(4)数组 a 的所有数中的最大值为 1。　　(　　)

●选择题

(5)输入 11 4 5 6 6 4 3 3 2 3 2 1,输出的结果为(　　)。

A. 3　　B. 4　　C. 5　　D. error

(6)输入 5 1 2 3 4 5,输出的结果为(　　)。

A. 2　　B. 3　　C. 4　　D. 5

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	√	×	×	A	B

【分析】

(1)sizeof(a)指的是 a 的内存大小,而 SIZE 指的是数组大小。

(2)a[x]++与++a[x]执行后都是把 a[x]+1,并没有执行什么其他语句,所以不影响。

(3)n 为偶数时两者值不同,若 n=2,(n/2+1)=2,(n+1)/2=1。

(4)a[x]指 x 的出现个数,显然最大值可以大于 1。

(5)依题意模拟即可。

(6)依题意模拟即可。

4.【NOIP2014】

```
01 #include<iostream>
02 #include<string>
03 using namespace std;
04 const int SIZE=100;
05 int main(){
06     string dict[SIZE];
```

```
07      int rank[SIZE];
08      int ind[SIZE];
09      int i,j,n,tmp;
10      cin>>n;
11      for (i=1;i<=n;i++){
12          rank[i]=i;
13          ind[i]=i;
14          cin>>dict[i];
15      }
16      for (i=1;i<n;i++)
17          for (j=1;j<=n-i;j++)
18              if (dict[ind[j]]>dict[ind[j+1]]){
19                  tmp=ind[j];
20                  ind[j]=ind[j+1];
21                  ind[j+1]=tmp;
22          }
23      for (i=1;i<=n;i++)
24          rank[ind[i]]=i;
25      for (i=1;i<=n;i++)
26          cout<<rank[i]<<" ";
27      cout<<endl;
28      return 0;
29  }
```

●判断题

(1)该程序的本质是按字典序对字符串排序。 (　　)

(2)如果输入 0,没有任何输出。 (　　)

(3)如果使用 C++98 编译,不会出现编译错误。 (　　)

(4)如果去掉第 12 行,不影响。 (　　)

●选择题

(5)输入 7\naaa\naba\nbbb\naaa\naaa\nccc\naa,输出(　　)。

A. 2 5 6 3 4 7 1　　B. 1 2 3 4 5 6 7

C. 1 7 4 3 6 5 2　　D. 7 6 5 4 3 2 1

(6)这个程序使用的是什么排序(　　)。

A. 插入排序　　B. 希尔排序　　C. 选择排序　　D. 冒泡排序

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	√	√	A	D

【分析】此程序实现了对输入字符串的冒泡排序,并按输入顺序输出每个字符串按照字典序从小到大排好序中的位置。由于冒泡排序是稳定的,因此相同的字符串应该保持输入的顺序。

(1)见分析,准确来说是稳定排序,并非不稳定排序,所以不能直接说排序。

(2)有换行。
(3)该程序未使用任何 C++98 不能兼容的语法。
(4)因为之后会对这个数组的 n 个元素进行修改。
(5)见分析。
(6)见分析。

5.【NOIP2016】

```
01#include<iostream>
02 using namespace std;
03 int main(){
04     int a[6];
05     int pi=0;
06     int pj=5;
07     int t,i;
08     while (pi<pj){
09         t=a[pi];
10         a[pi]=a[pj];
11         a[pj]=t;
12         pi++;
13         pj--;
14     }
15     for (i=0;i<6;i++)
16         cout<<a[i]<<",";
17     cout<<endl;
18     return 0;
19 }
```

●判断题

(1)将第 8 行改成 pi<=pj 不影响程序结果。　(　　)
(2)程序输出六个数,逗号只出现在相邻两个数之间。　(　　)
(3)如果 a={1,2,3,4,5,6},那么输出 6,5,4,3,2,1。　(　　)

●选择题

(4)这个程序在(　　)。
A. 将已知序列翻转　　B. 求序列的逆
C. 求序列的卷积　　D. 将原始数列随机打乱

(5)这个程序的时间复杂度是(　　)。
A. $O(n^2)$　　B. $O(n)$　　C. $O(n\log n)$　　D. $O(n\sqrt{n})$

(6)如果 a={6,5,4,3,2,1},那么输出(　　)。
A. 6,5,4,3,2,1　　B. 6 5 4 3 2 1　　C. 1 2 3 4 5 6　　D. 1,2,3,4,5,6,

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	×	×	A	B	D

【分析】可以采用分析功能的方法。此程序将一个数组首尾翻转后输出,注意千万不要遗漏

最后一个逗号。

(1)将相同的数交换不会对结果造成影响。

(2)最后还有一个逗号。

(3)最后还有一个逗号。

(4)同分析。

(5)因为每一个数只会访问一次,所以时间复杂度为 O(n)。

(6)根据分析,选 D。

6.【NOIP2017】

```
01 #include<iostream>
02 using namespace std;
03 int n,s,a[100005],t[100005],i;
04 void mergesort(int l,int r){
05     if (l==r)
06     return;
07     int mid=(l+r)/2;
08     int p=l;
09     int i=l;
10     int j=mid+1;
11     mergesort(l,mid);
12     mergesort(mid+1,r);
13     while (i<=mid&&j<=r){
14         if (a[j]<a[i]){
15             s+=mid-i+1;
16             t[p]=a[j];
17             p++;
18             j++;
19         }else {
20             t[p]=a[i];
21             p++;
22             i++;
23         }
24     }
25     while (i<=mid){
26         t[p]=a[i];
27         p++;
28         i++;
29     }
30     while (j<=r){
31         t[p]=a[j];
32         p++;
33         j++;
34     }
```

```
35      for (i=l;i<=r;i++)
36          a[i]=t[i];
37  }
38
39  int main(){
40      cin>>n;
41      for (i=1;i<=n;i++)
42          cin>>a[i];
43      mergesort(1,n);
44      cout<<s<<endl;
45      return 0;
46  }
47
```

●判断题

(1)如果将第 10 行的 mid+1 改成 mid,不会影响程序结果和时间复杂度。　(　　)

(2)如果将第 7 行的(l+r)/2 改成(l+r+1)/2 不会影响程序结果和时间复杂度。

(　　)

(3)程序输出结果不可能是 0。　(　　)

●选择题

(4)该程序的时间复杂度是(　　)。

A. O(n)　　B. O(nlogn)　　C. $O(n\sqrt{n})$　　D. $O(\frac{n^2}{w})$

(5)如果输入 6\n 2 6 3 4 5 1,输出(　　)。

A. 1　　B. 2　　C. 4　　D. 8

(6)该程序是求输入序列的(　　)。

A. 元素总和　　B. 逆序列　　C. 逆序对　　D. 卷积

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	√	×	B	D	C

【分析】此程序是利用归并排序求逆序对个数 s,直接求解逆序对即可。

(1)答案是错误的,如反例 321,错误程序会输出 0。

(2)显然答案不会变,仍然可以保证程序会递归 O(logn)层,每个元素只会被访问 O(logn)次,所以复杂度不变,仍然是 O(nlogn)。

(3)根据分析,在序列是单调不降序列时会输出 0,如 3\n 1 2 3。

(4)原因同第二题。

(5)手动计算逆序对。

(6)同分析。

第5节 递推与递归

1.【NOIP2008】

```
01 #include<iostream>
02 using namespace std;
03 void foo(int a,int b,int c)
04 {
05     if (a>b)
06         foo(c,a,b);
07     else
08         cout<<a<<','<<b<<','<<c<<endl;
09 }
10 int main()
11 {
12     int a,b,c;
13     cin>>a>>b>>c;
14     foo(a,b,c);
15     return 0;
16 }
```

判断题

(1)该程序只输出一行。 (　　)

(2)如果输入的三个数都相同,程序会运行错误。 (　　)

(3)如果输入3 1 2,输出2,3,1。 (　　)

(4)如果输入3 2 1,程序会超时。 (　　)

选择题

(5)输入9 10 8,输出为(　　)。

A. 9,10,8　　B. 9,8,10　　C. 10,8,9　　D. 8,9,10

(6)令n代表输入变量a、b、c的次数,则n=3该程序的时间复杂度为(　　)。

A. O(n)　　B. O(logn)　　C. $O(\sqrt{\log n})$　　D. $O(\log(n\sqrt{n}))$

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	×	√	×	A	A

【分析】本程序是一个简单的递归,直接模拟即可。

(1)因为程序只有输出一行才会结束,而程序不会出现死循环。

(2)程序最多只会调用3次,所以不会运行错误。

(3)手动模拟即可。

(4)理由同2。

(5)手动模拟即可。

(6)因为函数只会调用3n次,所以时间复杂度为O(n)。

2.【NOIP2008】

```
01 #include<iostream>
02 using namespace std;
03 void f(int a,int b,int c)
04 {
05     cout<<a<<b<<c<<'/';
06     if (b<c)
07         f(a,c,b);
08     else if (a<b)
09     {
10         if (a<c)
11             f(c,a,b);
12         else
13             f(b,c,a);
14     }
15 }
16 int main()
17 {
18     int a,b,c;
19     cin>>a>>b>>c;
20     f(a,b,c);
21     cout<<endl;
22     return 0;
23 }
```

●判断题

(1)程序可能没有输出。（　　）

(2)程序可能会死循环。（　　）

(3)输入 10 10 10 程序会运行错误。（　　）

(4)输入 258 114514 1919810 程序会运行错误。（　　）

●选择题

(5)如果 n=3 代表输入变量个数 n=3,那么该程序时间复杂度是(　　)。

A. $O(2^n)$　　B. $O(n!)$　　C. $O(n^2)$　　D. $O(n^n)$

(6)如果输入 1 3 2,则输出是(　　)。

A. 132/213/231/321/312/　　B. 132/213/231/312/321/

C. 213/132/231/321/312/　　D. 213/132/231/312/321/

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	×	×	B	B

【分析】本程序的含义是,给出三个数,输出这三个数按照字典序所对应的所有的全排列。

(1)由第五行可得程序必定会有输出。

(2)通过分析可得不会出现死循环。

(3)通过分析可得这不会运行错误。

(4)通过分析可得这不会运行错误。

(5)因为全排列为 O(n!)个,所以复杂度为 O(n!)。

(6)通过手动模拟得到答案。

3.【NOIP2014】

```
01 #include<iostream>
02 using namespace std;
03 int fun(int n){
04     if (n==1)return 1;
05     if (n==2)return 2;
06     return fun(n-2)-fun(n-1);
07 }
08 int main(){
09     int n;
10     cin>>n;
11     cout<<fun(n)<<endl;
12     return 0;
13 }
```

●判断题

(1)输入 114514 时,在普通计算机上程序运行时间不会超过 1s。 (　　)

(2)输入 0 程序不会出现运行错误。 (　　)

(3)该程序开启 O2 不会出现错误。 (　　)

(4)输入 6,输出 7。 (　　)

●选择题

(5)时间复杂度为(　　)。

A. $O(2^n)$　　B. $O(n^n)$　　C. $O(n^{\log n})$　　D. O(n)

(6)输入 7 时输出(　　)。

A. −11　　B. 11　　C. −10　　D. 10

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	√	√	A	A

【分析】fun 为递归函数,直接模拟难度较大。如果把 fun 视为数列,参数 n 视为该序列的项数,则容易发现,该序列满足 fun(1)=1,fun(2)=2,n≥3 时,fun(n)=fun(n−2)−fun(n−1)。而本题的目标转化为求数列的第 7 项,因此可以使用递推依次计算数列各项:

n	1	2	3	4	5	6	7
fun(n)	1	2	−1	3	−4	7	−11

(1)原因见第五题。

(2)将 0 代入会发现一直进行 fun(n−2)−fun(n−1)的递归,没有结束边界点而发生栈溢出。

(3)程序开 O2 优化时编译上没有错误。

(4)具体计算过程看上述表格。
(5)因为每一函数会调用前两个函数并且没有记忆化,所以时间复杂度为 $O(2^n)$。
(6)具体计算过程看上述表格。

4.【NOIP2010】

```
01 #include<iostream>
02 using namespace std;
03 const int NUM=5;
04 int r(int n)
05 {
06     int i;
07     if (n<=NUM)
08         return n;
09     for (i=1;i<=NUM;i++)
10         if (r(n-i)<0)
11             return i;
12     return -1;
13 }
14 int main()
15 {
16     int n;
17     cin>>n;
18     cout<<r(n)<<endl;
19     return 0;
20 }
```

●判断题

(1)将第 4 行的 int 改为 unsigned,答案不会错误。　(　　)
(2)程序开启 O2 优化不会返回错误。　(　　)
(3)如果输入−1,程序会输出−1。　(　　)
(4)该问题 r(n)的值没有规律。　(　　)

●选择题

(5)如果输入 7,程序会输出(　　)。
A. −1　　B. 5　　C. 1　　D. 3
(6)如果输入 16,程序会输出(　　)。
A. 16　　B. −1　　C. 4　　D. 1

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	√	√	×	C	C

【分析】本程序是进行模拟,中间有递归,可以列一张表帮助处理问题。

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
r(n)	1	2	3	4	5	−1	1	2	3	4	5	−1	1	2	3	4

可以看到这个值是每 6 个一个循环周期，也可以通过这个特性解题。

(1)由表可知，当访问的值中出现 r 值为－1 时会出现错误。

(2)没有错误所以不会编译错误。

(3)根据第 7 行和第 8 行可以知道。

(4)由表可知程序有循环节 6。

(5)由表可知输出为 1。

(6)由表可知输出为 4。

5.【NOIP2011】

```
01 #include<iostream>
02 using namespace std;
03 int solve(int n,intm){
04     int i,sum;
05     if (m==1)return 1;
06     sum=0;
07     for (i=1;i<n;i++)
08         sum+=solve(i,m-1);
09     return sum;
10 }
11 int main(){
12     int n,m;
13     scanf(''%d%d'',&n,&m);
14     cout<<solve(n,m)<<endl;
15     return 0;
16 }
```

●判断题

(1)将第 7 行＜改成＜＝程序会出现运行错误。 (　　)

(2)本题使用 C++98 编译不会出现编译错误。 (　　)

(3)本题输入 0 0 不会出现运行错误。 (　　)

(4)本题不可能输出 0。 (　　)

●选择题

(5)如果输入 7 4，输出(　　)。

A. 11　　B. 20　　C. 21　　D. 12

(6)如果输入 10 0，输出(　　)。

A. 0　　B. 1　　C. 2　　D. 3

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	×	√	×	B	A

【分析】这个程序用到了递归，可以借助表格来计算。

	n=1	n=2	n=3	n=4	n=5	n=6	n=7
m=1	1	1	1	1	1	1	1
m=2	0	1	2	3	4	5	6
m=3	0	0	1	3	6	10	15
m=4	0	0	0	1	4	10	20

其实可以发现 solve (n,m)=solve (n,m-1)+solve (n-1,m-1)。

(1)当输入 10 0 的时候会出现运行错误。

(2)iostream 库支持 scanf 函数只有在 C++11 后再有。

(3)手动模拟即可。

(4)当输入 10 0 的时候会出现 0。

(5)根据上表可得。

(6)手动模拟即可。

6.【NOIP2011】

```
01 #include<iostream>
02 using namespace std;
03 int n;
04 void f2(int x,int y);
05 void f1(int x,int y)
06 {
07     if (x<n)
08         f2(y,x+y);
09 }
10 void f2(int x,int y)
11 {
12     cout<<x<<' ';
13     f1(y,x+y);
14 }
15 int main()
16 {
17     cin>>n;
18     f1(0,1);
19     return 0;
20 }
```

●判断题

(1)把 04 行删除,程序不会出现编译错误。　(　　)

(2)输入为 10,输出为 125。　(　　)

(3)把 07 行的“x<n”改为“x<=n”,程序运行结果会出现改变。　(　　)

(4)该程序的时间复杂度为 O(n)。　(　　)

●选择题

(5)当输入为 35 时,输出为(　　)。

A. 1 2 5 13 34　　B. 1 2 5 13　　C. 1 2 13　　D. 2 5 13

(6)输出为 1 2 5 时,n 的值可以为(　　)。

A. 3　　B. 5　　C. 9　　D. 11

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	√	×	A	B

【分析】斐波那契数隔一个输出一个,直到后一个数大于等于 n。

(1)函数必须先声明才能调用,f1 函数中有调用 f2,所以得先声明 f2。

(2)0,1,1,2,3,5,8,13,输出应该为 1 2 5 13。

(3)n 为斐波那契数列中的数时,修改后输出会不同。

(4)时间复杂度显然达不到 O(n)。

(5)0,1,1,2,3,5,8,13,21,34,55,输出为 1 2 5 13 34。

(6)显然,n∈[4,8]。

7.【NOIP2015】

```
01 #include<iostream>
02 using namespace std;
03 int fun(int n,int fromPos,int toPos){
04     int t,tot;
05     if (n==0)return 0;
06     for (t=1;t<=3;t++)
07         if (t!=fromPos&&t!=toPos)break;
08     tot=0;
09     tot+=fun(n-1,fromPos,t);
10     tot++;
11     tot+=fun(n-1,t,toPos);
12     return tot;
13 }
14 int main(){
15     int n;
16     cin>>n;
17     cout<<fun(n,1,3)<<endl;
18     return 0;
19 }
```

●判断题

(1)当 n 为小于 1000 的正整数时,将第 9 行和第 11 行一起去掉,程序输出结果为 1。(　　)

(2)当 n 为小于 1000 的正整数时,将第 9 行或第 11 行中其中一行去掉,程序输出 n。(　　)

(3)函数中的 fromPos 与 toPos 与答案无关。(　　)

(4)该程序的时间复杂度为 $O(2^n)$。(　　)

●选择题

(5)fun(5,1,3)的值为(　　)。

A. 31　　B. 23　　C. 66　　D. 9

(6)fun(n,1,3)的通项公式为(　　)。

A. fun(n−1,1,3) * 2+1　　B. 2 * n

C. 2^n-1　　D. $\left(\left(\frac{1+\sqrt{5}}{2}\right)^n-\left(\frac{1-\sqrt{5}}{2}\right)^n\right)/\sqrt{5}$

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	√	√	√	A	C

【分析】本题考查了一个递归函数,但是仔细观察可以发现,后两个参数对函数的返回值完全没有影响。递推式可以简写为 fun(i)=fun(i−1) * 2+1,通项为 2^n-1。

(1)两行全去掉时,tot 总会等于 1,则程序输出 1。

(2)当去掉其中一行,可归纳证明 fun(i)=i。

(3)fromPos 与 toPos 和函数求解没有必要联系。

(4)易得 T(n)=2 * T(n−1)−1,求得时间复杂度为 $O(2^n)$。

(5)通过通项公式可以计算出答案为 31。

(6)通项公式即为 2^n-1,可以用等比数列的相关知识推导。

8.【NOIP2017】

```
01 #include<iostream>
02 using namespace std;
03 int g(int m,int n,int x){
04     int ans=0;
05     int i;
06     if (n==1)
07         return 1;
08     for (i=x;i<=m/n;i++)
09         ans+=g(m-i,n-1,i);
10     return ans;
11 }
12 int main(){
13     int t,m,n;
14     cin>>m>>n;
15     cout<<g(m,n,0)<<endl;
16     return 0;
17 }
```

●判断题

(1)把第 6 行去掉,程序总会输出 1。　(　　)

(2)将第 4 行的内容接在第 2 行的后面,程序输出与原样不同。　(　　)

(3)把第 5 行去掉,程序会编译错误。　(　　)

(4)此程序的功能是求将 m 个无序物品无序地分成 n 份的方案数。 (　　)

●选择题

(5)输入 7 3,则输出(　　)。

A. 2　　B. 4　　C. 6　　D. 8

(6)输入 5 5,则输出(　　)。

A. 1　　B. 3　　C. 5　　D. 7

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	√	√	√	D	D

【分析】此程序是求将 m 个无序物品无序地分成 n 份的方案数。

(1)将第 6 行去掉,函数总会返回 1,故程序总会输出 1。

(2)将 ans 改为全局变量,每次递归计算的 ans 都是非独立的,故输出与原样不同。

(3)将第 5 行的声明去掉,程序会编译错误。

(4)此程序是求将 m 个无序物品无序地分成 n 份的方案数。

(5)可以通过该程序的组合意义求出 g(7,3,0)=8。

(6)可以通过该程序的组合意义求出 g(5,5,0)=7。

9.【NOIP2018】

```
01 #include<iostream>
02 using namespace std;
03 int n,m;
04 int findans(int n,int m){
05     if (n==0) return m;
06     if (m==0) return n%3;
07     return findans(n-1,m)-findans(n,m-1)+findans(n-1,m-1);
08 }
09 int main() {
10     cin>>n>>m;
11     cout<<findans(n,m)<<endl;
12     return 0;
13 }
```

●判断题

(1)将第 5 行和第 6 行一起去掉,程序会出现死循环。 (　　)

(2)当输入的 n,m 的绝对值在 1000 以内时,程序一定会正常运行。 (　　)

(3)若将该递归程序执行记忆化,则程序的时间复杂度为 O(nm)。 (　　)

(4)将第 3 行接在第 9 行后,则程序会编译错误。 (　　)

●选择题

(5)输入 5 6,则输出为(　　)。

A. 5　　B. 6　　C. 7　　D. 8

(6)输入 2 4,则输出为(　　)。

A. 5　　B. 6　　C. 7　　D. 8

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	×	√	×	D	B

【分析】本题可以通过列表法，列一个二维表格，以 n 从小到大，m 从小到大的顺序依次算出每个格子的值。格子(n,m)的值可以通过格子(n－1,m),(n,m－1)和(n－1,m－1)的值得到。

n/m	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	0	3	2	5	4	7
2	2	－1	4	1	6	3	8
3	0	1	2	3	4	5	6
4	1	0	3	2	5	4	7
5	2	－1	4	1	6	3	8

(1)第 5 行和第 6 行一起去掉后程序没有递归边界，故出现死循环。

(2)当输入的 n,m 均取负值时，递归不会碰到递归边界，故出现死循环。

(3)将递归记忆化后，每个格点只会被访问一次，故程序时间复杂度为 O(nm)。

(4)将第 3 行接在第 9 行后，n,m 从全局变量变为局部变量，不会导致程序编译错误。

(5)可以通过列表法求出，findans(5,6)＝8。

(6)可以通过列表法求出，findans(2,4)＝6。

10.【NOIP2014】

```
01 #include<iostream>
02 using namespace std;
03 int fun(int n,int minNum,int maxNum){
04     int tot,i;
05     if (n==0)return 1;
06     tot=0;
07     for (i=minNum;i<=maxNum;i++)
08         tot+=fun(n-1,i+1,maxNum);
09     return tot;
10 }
11 int main(){
12     int n;
13     cin>>n;
14     cout<<fun(n,1,6)<<endl;
15     return 0;
16 }
```

●判断题

(1)将第 5 行删掉，程序编译错误。　　(　　)

(2)当输入的 n 的绝对值在 1000 以内时,程序不一定能正常运行。 (　　)

(3)将第 4 行的内容接在第 2 行后,程序输出与原样不同。 (　　)

(4)将第 4 行的内容去掉,程序将会运行错误。 (　　)

●选择题

(5)fun(2,1,6)的值为(　　)。

A. 15　　B. 10　　C. 6　　D. 3

(6)fun(3,1,6)的值为(　　)。

A. 20　　B. 10　　C. 4　　D. 1

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	√	×	×	A	A

【分析】fun 为递归函数,直接模拟难度较大。但由于数据范围较小,可以用递推依次计算。此题中 maxNum=6 为常数,故可以用矩阵列出 fun 与 n 和 minNum 的关系。另外,根据程序可知:

$$fun(n,minNum,maxNum)=\sum_{i=minNum}^{maxNum} fun(n-1,i+1,maxNum)$$

$$fun(n,minNum+1,maxNum)=\sum_{i=minNum+1}^{maxNum} fun(n-1,i+1,maxNum)$$

因而有

$$fun(n,minNum,maxNum)$$
$$=fun(n-1,minNum+1,maxNum)+fun(n-1,minNum+1,maxNum)$$

从而可以简化计算。

	n=0	n=1	n=2	n=3
minNum=1	1	6	15	20
minNum=2	1	5	10	10
minNum=3	1	4	6	4
minNum=4	1	3	3	1
minNum=5	1	2	1	0
minNum=6	1	1	0	0
minNum=7	1	0	0	0

(1)省去一个边界条件,不导致程序编译错误。

(2)当输入的 n 取负值,minNum>maxNum 时还是会结束递归。

(3)将 tot,i 定义为全局变量后,每一次调用都会影响到 tot,i 的值,故程序的输出与原样不同。

(4)将声明去掉,程序将会编译错误,但不会运行错误。

(5)可以通过列表法,查表得到 fun(2,1,6)=15。

(6)可以通过列表法,查表得到 fun(3,1,6)=20。

第 6 节　指针和引用

1.【NOIP2008】

```
01 #include<iostream>
02 using namespace std;
03 void func(int ary[],int n)
04 {
05     int i=0,j,x;
06     j=n-1;
07     while(i<j)
08     {
09         while(i<j&&ary[i]>0)i++;
10         while(i<j&&ary[j]<0)j--;
11         if(i<j){
12             x=ary[i];
13             ary[i++]=ary[j];
14             ary[j--]=x;
15         }
16     }
17 }
18
19 int main()
20 {
21     int a[20],i,m;
22     m=10;
23     for(i=0;i<m;i++)
24     {
25         cin>>a[i];
26     }
27     func(a,m);
28     for(i=0;i<m;i++)
29         cout<<a[i]<<"";
30     cout<<endl;
31     return 0;
32 }
```

●判断题

(1)将第 13 行的 ary[i++]=ary[j]改为 ary[++i]=ary[j],程序输出结果不变。(　　)

(2)输入 5 4 −6 −11 6 −59 22 −6 1 10,输出结果为 5 4 10 16 22 −59 −11 −6 −6。(　　)

(3)将 func 函数中的所有变量 n 全部改成 a,程序能通过编译且输出结果不变。(　　)

●选择题

(4)该代码时间复杂度为(　　)。

A. O(nlogn)　　B. O(n^2)　　C. O(n)　　D. O(logn)

题号	(1)	(2)	(3)	(4)
答案	×	×	√	C

【分析】一个类似于快排的函数 func,作用于数组 a 使得数列正负分开。

(1)i++表示返回 i 本身,然后 i 加 1;++i 表示 i 加 1,然后返回 i 本身。

(2)简单模拟可得输出结果为 5 4 10 1 6 22 −59 −6 −11 −6。

(3)main 函数中的 a 数组为局部变量,因此 func 函数中的变量 a 是不会与 main 函数中的 a 数组冲突的。

(4)该代码的原理是通过从左往右的指针与从右往左的指针同时扫描寻找不符合条件的数并交换位置,因此时间复杂度为 O(n)。

2.【NOIP2015】

```
01 #include<iostream>
02 using namespace std;
03 void fun(char *a,char *b){
04     a=b;
05     (*a)++;
06 }
07 int main(){
08     char c1,c2,*p1,*p2;
09     c1='A';
10     c2='a';
11     p1=&c1;
12     p2=&c2;
13     fun(p1,p2);
14     cout<<c1<<c2<<endl;
15     return 0;
16 }
```

●判断题

(1)将第 5 行的(* a)++;改为 a++;,程序输出结果不变。　　(　　)

(2)* 为取地址符。　　(　　)

(3)执行完第 4 行后 c1=c2。　　(　　)

●选择题

(4)输出结果为(　　)。

A. Ab　　B. Aa　　C. Ba　　D. Bb

【答案】

题号	(1)	(2)	(3)	(4)
答案	×	×	×	A

【分析】

(1)(*a)++表示指针a所指向的地址里的值加1,a++表示指针a所指向的地址+1。而且该行为在指针指向一个变时量,为未定义行为,在实际实现代码中使用危险。

(2)&为取地址符。

(3)a=b是指将b指向的地址赋值给a,即让a指向b所指向的地址,虽然改变了c2的值,但是修改后c1='A',c2= 'B'。

(4)第4行执行完后,a指向c2的地址,所以(*a)++会使得c2='a'+1='b'。

第7节 搜索算法

1.【NOIP2012】

```
01 #include<iostream>
02 using namespace std;
03 int n,i,j,a[100][100];
04 int solve(int x,int y){
05     int u,v;
06     if (x==n)return a[x][y];
07     u=solve(x+1,y);
08     v=solve(x+1,y+1);
09     if (u>v)
10         return a[x][y]+u;
11     else
12         return a[x][y]+v;
13 }
14 int main(){
15     cin>>n;
16     for(i=1;i<=n;i++)
17         for(j=1;j<=i;j++)cin>>a[i][j];
18     cout<<solve(1,1)<<endl;
19     return 0;
20 }
```

●判断题

(1)该程序读入 n+n*n 个整数。 (　　)

(2)该程序能正常运行。 (　　)

(3)输出结果均为整数。 (　　)

(4)该程序有多组测试。 (　　)

●选择题

(5)若输入为：

```
5
2
-1  4
2  -1  -2
-1  6  4  0
3  2  -1  5  8
```

则结果是(　　)。

A. 11　　B. 12　　C. 13　　D. 14

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	×	√	√	×	D

【分析】该程序是在找一条从(1,1)出发到(n,x)的最大的路径。

(1)该程序读入 1+n*(n+1)/2 个整数。

(2)它确实可以正常运行。

(3)因为输出的类型是 int,所以结果均为整数。

(4)该程序只调用一次 solve 函数,没有体现多组测试。

(5)这个程序是在找一条从(1,1)到(n,x)(1≤x≤n)的最大和路径,每次只能从(i,j)走到(i+1,j)或(i+1,j+1),通过观察可以得到最大和路径为 14。

2.【NOIP2013】

```
#include<cstring>
#include<iostream>
using namespace std;
const int SIZE=100;
int n,m,p,a[SIZE][SIZE],cnt;

void colour(int x,int y){
    cnt++;
    a[x][y]=1;
    if ((x>1) && (a[x-1][y]==0)) colour(x-1,y);
    if ((y>1) && (a[x][y-1]==0)) colour(x,y-1);
    if ((x<n) && (a[x+1][y]==0)) colour(x+1,y);
    if ((y<m) && (a[x][y+1]==0)) colour(x,y+1);
}

int main(){
    int i,j,x,y,ans;
    memset(a,0,sizeof(a));
    cin>>n>>m>>p;
    for(i=1;i<=p;i++){
        cin>>x>>y;
        a[x][y]=1;
    }
    ans=0;
    for(i=1;i<=n;i++)
        for(j=1;j<=m;j++)
            if(a[i][j]==0){
                cnt=0;
                colour(i,j);
                if (ans<cnt)ans=cnt;
            }
    cout<<ans<<endl;
    return 0;
```

```
34 }
```

判断题

(1)由于没有赋初值,该程序会运行错误。 (　　)

(2)如果将第 18 行去掉,程序结果会发生改变。 (　　)

(3)如果将第 30 行的 ans<cnt 改成 ans<=cnt,输出结果会发生改变。 (　　)

(4)该程序有可能输出 114514。 (　　)

选择题

(5)若输入为

```
6 5 9
1 4
2 3
2 4
3 2
4 1
4 3
4 5
5 4
6 4
```

则输出结果是(　　)。

A. 114514　　B. 1919810　　C. 7　　D. 8

(6)该程序的时间复杂度为(　　)。

A. O(1)　　B. O(n)　　C. O(n * m)　　D. $O(2^n)$

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	×	×	C	C

【分析】本题功能:输入一个 n * m 的矩阵,求最大的连通块大小。

(1)在之后会赋值,程序不会运行错误。

(2)全局变量初值为 0,因此可以不用再初始化为 0。

(3)该程序只要求出最大值即可,因此将小于改成小于等于不影响结果。

(4)该程序最多输出 10000。

(5)最大的连通块大小为 7。

(6)该程序会把每个点都走一遍,即 O(n * m)。

第 8 节　动态规划

1.【NOIP2013】

```
01 #include<iostream>
02 using namespace std;
03 int main(){
04     const int SIZE=100;
05     int height[SIZE],num[SIZE],n,ans;
06     cin>>n;
07     for(int i=0;i<n;i++){
08         cin>>height[i];
09         num[i]=1;
10         for (int j=0;j<i;j++){
11             if ((height[j]<height[i])&&(num[j]>=num[i]))
12                 num[i]=num[j]+1;
13         }
14     }
15     ans=0;
16     for (int i=0;i<n;i++){
17         if (num[i]>ans)ans=num[i];
18     }
19     cout<<ans<<endl;
20 }
```

●判断题

(1)将第 4 行的程序移动到第 2、3 行中间，程序能够正常运行。　(　　)

(2)第 6 行输入 n=5，则输出 ans 的值一定小于 5。　(　　)

(3)把 01 行的 iostream 改为 cstdio 时不会编译错误。　(　　)

(4)如果输出是 1，则 height 数组中的数一定是递减的。　(　　)

●选择题

(5)当 n=6 时，输入 height 数组为 2 5 3 11 12 4，输出为(　　)。

A. 4　　B. 2　　C. 14　　D. 6

(6)如果将第 13 行的 height[j]<height[i]改成 height[j]>height[i]，则第(5)题的输出结果为(　　)。

A. 4　　B. 2　　C. 14　　D. 6

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	×	×	×	A	B

【分析】分析程序功能可知，该程序求

(1)常数变量可以放在 main 函数之外。

(2)当 height 递增时，ans=5。

(3)cin 和 cout 函数需要 iostream 库。

(4)height 全部相等时,输出也是 1。

(5)程序的作用是输出严格最长上升子序列的长度,所以输出 4({2,3,11,12})。

(6)修改后程序的作用是输出严格最长下降子序列的长度,所以输出 2({5,3})。

2.【NOIP2016】

```
01 #include<iostream>
02 using namespace std;
03 int lps(string seq,int i,int j){
04     int len1,len2;
05     if(i==j)
06         return 1;
07     if(i>j)
08         return 0;
09     if(seq[i]==seq[j])
10         return lps(seq,i+1,j-1)+2;
11     len1=lps(seq,i,j-1);
12     len2=lps(seq,i+1,j);
13     if (len1>len2)
14         return len1;
15     return len2;
16 }
17 int main(){
18     string seq;cin>>seq;
19     int n=seq.size();
20     cout<<lps(seq,0,n-1)<<endl;
21     return 0;
22 }
```

●判断题

(1)n 代表 seq 的长度。 (　　)

(2)输入 acmerandacm,输出 5。 (　　)

(3)第 7～8 行代码删去后程序仍能正常运行。 (　　)

(4)程序最好情况下的时间复杂度为 $O(n^2)$。 (　　)

●选择题

(5)程序的最坏时间复杂度为(　　)。

A. O(nlogn)　　B. $O(n^2)$　　C. O(n)　　D. $O(2^n)$

(6)函数 lps(seq,i,j)的用途是(　　)。

A. 求字符串 seq 的最长回文子序列长度。

B. 求字符串 seq 中区间[i,j]上的最长回文子串长度。

C. 求字符串 seq 中区间[i,j]上的最长回文子序列长度。

D. 求字符串 seq 中区间[i,j]上的最长相同前缀后缀长度。

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	√	×	×	D	C

【分析】分析程序功能可以知道，此程序的功能是求最长的回文子序列长度。

(1)seq.size()表示 seq 的长度。

(2)cadac 为输入串的一个最长回文子序列。

(3)输入 aa 时，删去这两行会运行错误。

(4)最好情况下，整个序列都是同一个字母，时间复杂度为 O(n)。

(5)最坏情况下，整个序列都是不同字符，答案区间每扩大 1，复杂度就乘以 2，所以总复杂度为 $O(2^n)$。

(6)分析程序功能可以知道，此函数的功能是求区间内最长的回文子序列长度，即原问题的子问题。

第9节 树和图

1.【NOIP2016】

```
#include<iostream>
#include<cstring>
using namespace std;
int map[100][100];
int sum[100],weight[100];
int visit[100];
int n;
void dfs(int node){
    visit[node]=1;
    sum[node]=1;
    int v,maxw=0;
    for (v=1;v<=n;v++){
        if (! map[node][v]||visit[v])
            continue;
        dfs(v);
        sum[node]+=sum[v];
        if (sum[v]>maxw)
            maxw=sum[v];
    }
    if (n-sum[node]>maxw)
        maxw=n-sum[node];
    weight[node]=maxw;
}
int main(){
    memset(map,0,sizeof(map));
    memset(sum,0,sizeof(sum));
    memset(weight,0,sizeof(weight));
    memset(visit,0,sizeof(visit));
    cin>>n;
    int i,x,y;
    for (i=1;i<n;i++){
        cin>>x>>y;
        map[x][y]=1;
        map[y][x]=1;
    }
    dfs(1);
    int ans=n,ansN=0;
```

```
38      for (i=1;i<=n;i++)
39          if (weight[i]<ans){
40              ans=weight[i];
41              ansN=i;
42          }
43      cout<<ansN<<""<<ans<<endl;
44      return 0;
45  }
```

●判断题

(1)ansN 和 ans 的值不可能一样。　　(　　)

(2)25～28 行删掉后程序照样输出正常结果。　　(　　)

(3)本程序的时间复杂度为 O(n)。　　(　　)

(4)38 行的 i++改为++i 程序照样输出正常结果。　　(　　)

●选择题

(5)输入为

```
11
1 2
1 3
2 4
2 5
2 6
3 7
7 8
7 11
6 9
9 10
```

输出为(　　)。

A. 2 5　　B. 3 4　　C. 4 5　　D. 3 5

(6)n 为 11 时,ans 的值最小为(　　)。

A. 4　　B. 5　　C. 6　　D. 7

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	√	×	√	A	B

【分析】阅读本题代码发现:题目中使用了邻接矩阵与 DFS,同时只读取了 n－1 条边,因此可知代码是在处理一个树上问题,下一步看到统计答案部分,发现统计了最小的 weight[i],以及对应的 i。

再查看 DFS 函数,统计了 sum 和 weight 两个量,sum 量明显指的是子树大小(注意:以 1 为根情况下的子树大小,子树大小与整个树的根的位置显然是有关系的)。而 weight 是在每个结点的最大的子树大小和 n－sum[node]中取最大值,其意义正是“以当前结点为根的

情况下，最大的一个子树大小”。

所以，此程序求解的问题是：给定一棵树，求树的重心，使得剩下的图中，最大的连通块所包含的结点数最少。如果有多个重心，取编号最小的，输出的是删去的结点编号以及最大的连通块结点数。

按照题意，手动模拟一下即可得到答案。

(1)ansN 和 ans 之间没有什么关系，有可能相等。

(2)定义在 main 外面的变量初始都为 1，所以不用 memset 为 0。

(3)用邻接矩阵存边，从 u 找出边的复杂度为 O(n)，总复杂度为 $O(n^2)$。

(4)++i 和 i++在 for 循环中作用一致。

(5)见分析，画出图之后就发现 2 为树的重心。

(6)树为一条链时，最中间的点刚好可以把树分成 5 5，所以最小答案为 5。

2.【NOIP2008】

```
01 #include<iostream>
02 #include<cstring>
03 using namespace std;
04 #define MAX 100
05 void solve(char first[],int spos_f,int epos_f,
06             char mid[],int spos_m,int epos_m)
07 {
08     int i,root_m;
09     if(spos_f>epos_f)
10         return;
11     for(i=spos_m;i<=epos_m;i++)
12         if(first[spos_f]==mid[i])
13         {
14             root_m=i;
15             break;
16         }
17     solve(first,spos_f+1,spos_f+(root_m-spos_m),
18             mid,spos_m,root_m-1);
19     solve(first,spos_f+(root_m-spos_m)+1,epos_f,
20             mid,root_m+1,epos_m);
21     cout<<first[spos_f];
22 }
23 int main()
24 {
25     char first[MAX],mid[MAX];
26     int len;
27     cin>>len;
```

```
28      cin>>first>>mid;
29      solve(first,0,len-1,mid,0,len-1);
30      cout<<endl;
31      return 0;
32 }
```

●判断题

(1)将 25 行移到 22 行和 23 行之间,程序不会出错。　　(　　)

(2)将 09 行和 10 行去掉,程序可以得出相同的结果。　　(　　)

(3)该程序的时间复杂度为 O(n)。　　(　　)

(4)将 25 行的 char 改为 int 类型,程序可以得到相同的结果。　　(　　)

●选择题

(5)输入为 7\nABDCEGF\nBDAGECF 时,输出为(　　)。

A. DBGEFCA　　B. DBGFECA　　C. GBDEFCA　　D. ABCDEFG

(6)该程序要解决的问题是(　　)。

A. 给出先序遍历和后序遍历求中序遍历　B. 给出先序遍历和中序遍历求后序遍历

C. 给出中序遍历和后序遍历求前序遍历　D. 给出前序遍历和中序遍历求层序遍历

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	×	×	×	A	B

【分析】根据先序遍历和中序遍历求后序遍历。len 表示我们所需要处理的字符串的长度,也就是一棵树的结点数。solve 过程中 first,mid 根本就没有改变,将他们作为全局变量来调用完全可以。solve 首先根据 first 中的第一个字母,它必然是该棵树的根,在 mid 中找到该字母,将其所在位置记录为 root_m(root 在 mid 字符串中的位置),这样根据 mid 便可以得出左右子树的结点数,下面便先左再右,各自进行递归。

(1)变量可以定义在 main 外面。

(2)程序将会无限递归下去。

(3)时间复杂度显然不为 O(n)。

(4)改为 int 类型后,输出就不为字符串了。

(5)见分析。

(6)见分析。

3.【NOIP2012】

```
01 #include<iostream>
02 #include<string>
03 using namespace std;
04 int lefts[20],rights[20],father[20];
05 string s1,s2,s3;
06 int n,ans;
07 void calc(int x,int dep)
```

```
08 {
09     ans=ans+dep*(s1[x]-'A'+1);
10     if(lefts[x]>=0)calc(lefts[x],dep+1);
11     if(rights[x]>=0)calc(rights[x],dep+1);
12 }
13 void check(int x)
14 {
15     if(lefts[x]>=0)check(lefts[x]);
16     s3=s3+s1[x];
17     if(rights[x]>=0)check(rights[x]);
18 }
19 void dfs(int x,intt h)
20 {
21     if(th==n)
22     {
23         s3="";
24         check (0);
25         if (s3==s2)
26         {
27             ans=0;
28             calc (0,1);
29             cout<<ans<<endl;
30         }
31         return;
32     }
33     if (lefts[x]==-1&&rights[x]==-1)
34     {
35         lefts[x]=th;
36         father[th]=x;
37         dfs (th,th+1);
38         father[th]=-1;
39         lefts[x]=-1;
40     }
41     if (rights[x]==-1)
42     {
43         rights[x]=th;
44         father[th]=x;
45         dfs(th,th+1);
46         father[th]=-1;
47         rights[x]=-1;
```

```
48      }
49      if (father[x]>=0)dfs(father[x],th);
50  }
51  int main()
52  {
53      cin>>s1;
54      cin>>s2;
55      n=s1.size();
56      memset(lefts,-1,sizeof(lefts));
57      memset(rights,-1,sizeof(rights));
58      memset(father,-1,sizeof(father));
59      dfs (0,1);
60  }
```

●判断题

(1)将 23 行去掉,程序运行结果与原来一致。　　(　　)

(2)该程序时间复杂度为 $O(n^3)$。　　(　　)

(3)将 02 行去掉,程序运行结果与原来一致。　　(　　)

(4)55 行函数时间复杂度为 O(1)。　　(　　)

●选择题

(5)当输入为 ABCDEF\n BCAEDF 时,输出为(　　)。

A. 54　　B. 55　　C. 56　　D. 57

(6)此程序主要执行的算法思路是(　　)。

A. 深搜　　B. 广搜　　C. 动规　　D. 分治

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	×	√	B	A

【分析】由先序遍历、中序遍历的序列构造出二叉树,再求出结点对应值的和——ABCDEF 基本值分别对应 123456,求出的是各个结点基本值乘以该结点深度的和。

值得注意的是,在构造二叉树的时候是严格按照先序遍历的方式进行深度优先搜索,x 相当于作为判断的结点,th 相当于要插入的结点,如果 x 没有左右子树就可以尝试往 x 的两个子树都分别添加 th 结点进行判断,如果 x 有左子树只能往右子树插,如果 x 有右子树那么就插不进去了,如果在这个结点插入是不行的,就往根结点跳。

(1)23 行的作用是清零。

(2)此程序复杂度为指数级。

(3)memset 在 string 库头,所以去掉会出现编译错误。

(4)string 类型的函数 size()复杂度为 O(1)。

(5)见分析,模拟即可。

(6)从代码中的回溯操作可以看出是深度优先搜索。

4.【NOIP2018】

```
01 #include<cstdio>
02 int n,d[100];
03 bool v[100];
04 int main(){
05     scanf("%d",&n);
06     for(int i=0;i<n;++i){
07         scanf("%d",d+i);
08         v[i]=false;
09     }
10     int cnt=0;
11     for(int i=0;i<n;++i){
12         if(!v[i]){
13             for(int j=i;!v[j];j=d[j]){
14                 v[j]=true;
15             }
16             ++cnt;
17         }
18     }
19     printf("%d\n",cnt);
20     return 0;
21 }
```

●判断题

(1)将第 7 行的 d+i 换成 &d[i],程序运行不受影响。 (　　)

(2)第 12 行的!v[i]与 v[i]==false 语句意思一致。 (　　)

(3)程序的输出结果 cnt 至少等于 1。 (　　)

(4)若输入的数组 d 中有重复的数字,则程序会进入死循环。 (　　)

●选择题

(5)若输入数字为 5 1 2 3 4 5,则输出为(　　)。

A. 0　　B. 1　　C. 2　　D. 5

(6)若输入数字为 10 7 1 4 3 2 5 9 8 0 6,则输出为(　　)。

A. 3　　B. 6　　C. 7　　D. 8

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	√	×	×	B	B

【分析】

(1)d+i 是 d[i]的取地址写法。

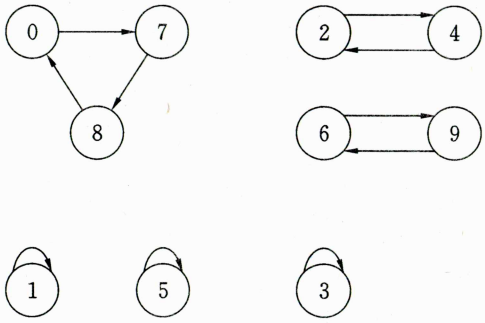
(2)!v[i]与 v[i]==false 语句意思一致。

(3)当 n=0 时,cnt=0。

(4)循环条件为!v[j],即当前位置没有到过,所以一个点最多经过一次,总时间复杂度O(n)。

(5)直接模拟,1次就遍历了所有点。

(6)发现输入先读入点的个数,再读入d数组,d数组是0到9这10个点的排列。d数组的第i个值就是i处边的另一端点,程序求的是这个图中环的个数,共6个,分别是(0,7,8),(1),(2,4),(5),(6,9)。



5.【NOIP2011】

```
01 #include<iostream>
02 using namespace std;
03 const int V=100;
04 int n,m,ans,e[V][V];
05 bool visited[V];
06 void dfs(int x,int len){
07     int i;
08     visited[x]=true;
09     if (len>ans)ans=len;
10     for (i=1;i<=n;i++)
11         if ((!visited[i])&&(e[x][i]!=-1))
12             dfs(i,len+e[x][i]);
13     visited[x]=false;
14 }
15 int main(){
16     int i,j,a,b,c;
17     cin>>n>>m;
18     for(i=1;i<=n;i++)
19         for(j=1;j<=n;j++)
20             e[i][j]=-1;
21     for(i=1;i<=m;i++){
22         cin>>a>>b>>c;
23         e[a][b]=c;
24         e[b][a]=c;
25     }
```

```
26      for(i=1;i<=n;i++)
27          visited[i]=false;
28      ans=0;
29      for(i=1;i<=n;i++)
30          dfs(i,0);
31      cout<<ans<<endl;
32      return 0;
33  }
```

●判断题

(1)输入的数据应满足 a,b＜100。 (　　)

(2)该程序实现的是多源最短路。 (　　)

(3)在没有重边与自环的情况下,m 最大为n^2。 (　　)

(4)对于一个 n 个点的边权均为 w 的完全图,输出为(n－1) * w。 (　　)

●选择题

(5)求该程序消耗空间约为(　　)。

A. 100MB　　B. 1GB　　C. 40KB　　D. 100KB

(6)

输入

4 6

1 2 10

2 3 20

3 4 30

4 1 40

1 3 50

2 4 60

输出(　　)

A. 150　　B. 60　　C. 210　　D. 10

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	×	×	√	C	A

【分析】该题求的是每个点最多经过一次的最长路。

(1)数组 e 的下标显然不能超过 99。

(2)该题求的是每个点最多经过一次的最长路。

(3)无向图 m 最大为$\frac{n(n-1)}{2}$。

(4)任意一条经过所有点的简单路径均为答案。

(5)100 * 100 * 32bit＝40000B＝40KB。

(6)该题求的是每个点最多经过一次的最长路,观察可知最长长度为 150。

6.【NOIP2010】

```
01 #include<iostream>
02 using namespace std;
03 const int SIZE=100;
04 int n,m,r[SIZE];
05 bool map[SIZE][SIZE],found;
06 bool successful()
07 {
08     int i;
09     for (i=1;i<=n;i++)
10         if (!map[r[i]][r[i% n+1]])
11             return false;
12         return true;
13 }
14 void swap(int *a,int *b)
15 {
16     int t;
17     t=*a;
18     *a=*b;
19     *b=t;
20 }
21 void perm(int left,int right)
22 {
23     int i;
24     if(found)
25         return;
26     if(left>right){
27         if(successful()){
28             for(i=1;i<=n;i++)
29                 cout<<r[i]<<'';
30             found=true;
31         }
32         return;
33     }
34     for(i=left;i<=right;i++){
35         swap(r+left,r+i);
36         perm(left+1,right);
37         swap(r+left,r+i);
38     }
39 }
40 int main()
41 {
```

```
42      int x,y,i;
43      cin>>n>>m;
44      memset(map,false,sizeof(map));
45      for(i=1;i<=m;i++){
46          cin>>x>>y;
47          map[x][y]=true;
48          map[y][x]=true;
49      }
50      for (i=1;i<=n;i++)
51          r[i]=i;
52      found=false;
53      perm(1,n);
54      if (!found)
55          cout<<"No solution!"<<endl;
56      return 0;
57  }
```

●判断题

(1)在 44 行前加一个左斜杠,程序能正常运行,结果不变。 (　　)

(2)去掉 14 行的 *,运行结果不变。 (　　)

(3)将 48 行删去不影响输出结果。 (　　)

(4)将 map 类型改为 int 不会影响运行结果。 (　　)

●选择题

(5)该程序时间复杂度为(　　)。

A. $O(n^{mlogn} \times m^{nlogm})$　B. O(n * m)　C. O(m! * n)　D. O(n! * n)

(6)输入:9 12 1 2 2 3 3 4 4 5 5 6 6 1 1 7 2 7 3 8 4 8 5 9 6 9

输出(　　)。

A. 1 2 3 4 5 6 7 8 9　　B. 9 8 7 2 4 3 5 1 6

C. 3 2 7 1 6 9 5 4 8　　D. 1 6 9 5 4 8 3 2 7

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	×	√	D	D

【分析】

(1)注释是双斜线,单斜线编译错误。

(2)* 代表引用,去掉后不能修改函数外的值,swap 失效。

(3)原图为双向边,改为单向边显然有影响。

(4)bool 型的 True/False 等价于 int 的 1/0。

(5)递归过程最坏情况下为全排列 n!,每次检验复杂度为 O(n),故总复杂度为 O(n! * n)。

(6)本题输入的是一个图,根据输入可以把边画出来,然后题目要求输出一个遍历序列,且字典序最小,所以答案是 1 6 9 5 4 8 3 2 7。如果一开始看不出来可以先进行模拟,然后再寻找其中的规律。

第 10 节　数论

1.【NOIP2012】

```
01 #include<iostream>
02 using namespace std;
03 int n,i,ans;
04 int main(){
05     cin>>n;
06     ans=0;
07     for(i=1;i<=n;i++)
08         if(n%i==0)ans++;
09     cout<<ans<<endl;
10 }
```

判断题

(1)该程序可能输出负数。　　(　　)

(2)可以将第 7 行的 i 初始值赋为 0。　　(　　)

(3)输入 18,输出结果为 6。　　(　　)

(4)该程序可以正常运行。　　(　　)

选择题

(5)如果将第 8 行的 ans++改成 ans--,那么输入 18,输出结果为(　　)。

A. 6　　B. −6　　C. man　　D. ans

(6)该算法的时间复杂度为(　　)。

A. O(n)　　B. O(1)　　C. $O(n^2)$　　D. O(nlogn)

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	√	√	B	A

【分析】通过简单分析,我们可知,本题求 n 的约数有多少个。

(1)这个程序是在求 n 的约数有多少个,不可能有负数。

(2)模数不能为 0。

(3)18 的约束个数为 6。

(4)没有 return 0;也能运行。

(5)改成自减,结果就是−6。

(6)枚举小于 n 的数,判断是否为 n 的约数,时间复杂度为 O(n)。

2.【NOIP2013】

```
01 #include<iostream>
02 using namespace std;
03 int main(){
04     int a,b,u,i,num;
05     cin>>a>>b>>u;
06     num=0;
```

```
07      for(i=a;i<=b;i++)
08          if((i%u)==0)num++;
09      cout<<num<<endl;
10      return 0;
11  }
```

●判断题

(1)u 可以为 0。 (　　)

(2)b－a 一定不小于 num。 (　　)

(3)输入 1 9 2,输出 4。 (　　)

(4)该程序时间复杂度与 u 有关。 (　　)

●选择题

(5)输出结果为 10,输入数据可能为(　　)。

A. 10 27 2　　B. 20 30 10　　C. 5 99 3　　D. 41 83 4

(6)输入 151 981 7,输出结果为(　　)。

A. 117　　B. 118　　C. 120　　D. 119

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	√	×	D	D

【分析】本题功能:输入 3 个整数 a,b,u,问在[a,b]区间内有几个数是 u 的倍数。

(1)除数不能为 0。

(2)若 a>b,那么 num 为 0。

(3)1 到 9 之间 2 的倍数有 2,4,6,8。

(4)该题时间复杂度为 O(b－a),与 u 无关。

(5)A 输出结果为 9,B 输出结果为 2,C 输出结果为 32,D 输出结果为 10。

(6)1 到 n 范围内 u 的倍数明显有$\left\lfloor\frac{n}{u}\right\rfloor$个,因此答案为$\left\lfloor\frac{981}{7}\right\rfloor-\left\lfloor\frac{151-1}{7}\right\rfloor=119$。

3.【NOIP2016】

```
01 #include<iostream>
02 using namespace std;
03 int main(){
04     int i=100,x=0,y=0;
05     while(i>0){
06         i--;
07         x=i%8;
08         if (x==1)
09             y++;
10     }
11     cout<<y<<endl;
12     return 0;
13 }
```

●判断题

(1)程序输出 12。　　(　　)

(2)若将第 6 行替换为“i++”,则程序无法在 1s 之内结束。　　(　　)

(3)若将第 4 行的“i=100”替换为“i=10”,则输出 1.2。　　(　　)

(4)若将第 6 行的“i－－”后加上一个“i++”,则程序会死循环。　　(　　)

●选择题

(5)若将第 8 行去掉,则程序输出(　　)。

A. 98　　B. 99　　C. 100　　D. 101

(6)若将第 9 行去掉,则程序输出(　　)。

A. 0　　B. 1　　C. 2　　D. 3

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	√	×	√	C	A

【分析】易知此程序为计算 0～99 以内除以 8 余 1 的数有多少个,计算可知共 13 个。

(1)计算可知程序输出 13。

(2)每次循环都执行 i++的话,i 越来越大,不可能满足 i>0,所以程序无法在 1s 之内结束。

(3)根据此程序功能可知应输出 1。

(4)每次循环,i 的值都不变,故该程序会死循环。

(5)若将第 8 行去掉,则该程序求的是区间[1,i]有多少整数,故输出 100。

(6)若将第 9 行去掉,y 没有被操作,程序输出 0。

4.【NOIP2009】

```
01 #include<iostream>
02 using namespace std;
03 int a,b;
04 int work(int a,int b) {
05     if(a%b) return work(b,a%b);
06     return b;
07 }
08
09 int main(){
10     cin>>a>>b;
11     cout<<work(a,b)<<endl;
12     return 0;
13 }
```

●判断题

(1)在第 2 行下面添加 #define int long long 程序可以正常运行。　　(　　)

(2)将第 3 行的 int 改成 double 结果不会改变。　　(　　)

(3)不能输入 0 0。　　(　　)

(4)输入 20 12 结果输出 4。　　(　　)

●选择题

(5)输入 2012 13,输出(　　)。

A. 2012　　B. 2013　　C. 13　　D. 1

(6)该算法的时间复杂度级别为(　　)。

A. 线性时间　　B. 对数时间　　C. 平方时间　　D. 常数时间

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	√	×	√	D	B

【分析】本程序是求 a 与 b 的最大公约数。

(1)int main()的返回值只能是 int,所以加#define int long long 会报错。

(2)double 在处理的时候转为 int。

(3)可以输入 0 0。

(4)20 和 12 的最大公约数为 4。

(5)因为 13 是质数,且 2012 不是 13 的倍数,所以 2012 和 13 的最大公约数为 1。

(6)辗转相除的时间复杂度级别是对数时间。

5.【NOIP2012】

```
1 #include<iostream>
2 using namespace std;
3 int n,i,ans;
4 int gcd(int a,int b){
5     if (a%b==0)
6         return b;
7     else
8         return gcd(b,a%b);
9 }
10
11 int main(){
12     cin>>n;
13     ans=0;
14     for(i=1;i<=n;i++)
15         if(gcd(n,i)==i)ans++;
16     cout<<ans<<endl;
17 }
```

●判断题

(1)将 n 定义为 double 类型,程序的输出结果不变。(　　)

(2)输入一个正整数,输出结果不可能不大于 1。(　　)

(3)输入 100000000000,输出结果为 144。(　　)

(4)第 12 行可改为 scanf("%d",&n);。(　　)

●选择题

(5)输入 120,输出为(　　)。

A. 15　　B. 16　　C. 18　　D. 14

(6)该算法时间复杂度为(　　)。

A. O(n)　　B. O(nlogn)　　C. $O(\sqrt{n})$　　D. $O(n^2)$

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	×	×	×	B	B

【分析】本题是求 n 的约数的个数。

(1)gcd 函数参数都为 int 类型,double 类型的 n 会被强制转化为 int 类型。

(2)输入 1,输出结果为 1。

(3)n 为 int 类型,输入 100000000000 会溢出。

(4)scanf 函数的调用必须要有 cstdio 库。

(5)本题就是求 n 的约数个数,$120=2^3\times3\times5$,所以约数有 $4\times2\times2=16$ 个。

(6)一次辗转相除法的时间复杂度为 O(logn),执行 n 次时间复杂度即为 O(nlogn)。

6.【NOIP2014】

```
01 #include<iostream>
02 using namespace std;
03 int main(){
04     int a,b,i,tot,c1,c2;
05     cin>>a>>b;
06     tot=0;
07     for(i=a;i<=b;i++){
08         c1=(i/10)%10;
09         c2=i%10;
10         if ((c1+c2)%3==0)tot++;
11     }
12     cout<<tot<<endl;
13     return 0;
14 }
```

●判断题

(1)将第 6 行去掉会导致运行错误。　　(　　)

(2)将第 4 行去掉会导致编译错误。　　(　　)

(3)若 a,b 都是 99 以内的整数,将第 8 行替换为“c1=i/10”没有影响。　　(　　)

(4)若 a,b 都是 99 以内的整数,则答案为区间[a,b]中 3 的倍数的数量。　　(　　)

●选择题

(5)输入 1100,则输出(　　)。

A. 31　　B. 32　　C. 33　　D. 34

(6)输入 110000,则输出(　　)。

A. 3333　　B. 3400　　C. 258　　D. 1950

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	√	√	√	D	B

【分析】易得此程序的功能就是统计 a 到 b 之间十位数字与个位数字之和能被 3 整除的数有多少个。

(1)局部变量没有赋初值不会导致运行错误,会导致答案错误。

(2)若将声明去掉,则导致编译错误。

(3)当 i 是 99 以内的正整数时,"i/10"与"(i/10)%10"等价。

(4)当 a,b 都是 100 以内的正整数时,十个位加起来是 3 的倍数则说明该数是 3 的倍数。

(5)100 以内的正整数,有 33 个 3 的倍数,tot=33,而当 i=100 时,c1=c2=0,此时(c1+c2)%3==0,tot++后 tot=34。

(6)由于只关心十位与个位,每一个 100 都是独立的,于是答案为 34 * 100=3400。

7.【NOIP2017】

```
01 #include<iostream>
02 using namespace std;
03 int main(){
04     int n,m;
05     cin>>n>>m;
06     int x=1;
07     int y=1;
08     int dx=1;
09     int dy=1;
10     int cnt=0;
11     while (cnt!=2){
12         cnt=0;
13         x=x+dx;
14         y=y+dy;
15         if (x==1||x==n){
16             ++cnt;
17             dx=-dx;
18         }
19         if (y==1||y==m){
20             ++cnt;
21             dy=-dy;
22         }
23     }
24     cout<<x<<""<<y<<endl;
25     return 0;
26 }
```

●判断题

(1)若将 16、20 行去掉,则程序会出现死循环。　　(　　)

(2)若将第一行的 iostream 改为 cstdio,则程序会出现编译错误。　　(　　)

(3)若将 17、21 行去掉,则程序会出现死循环。　　(　　)

(4)记 L=min(n,m),则时间复杂度为 $O(L^2)$。　　(　　)

●选择题

(5)输入 2 3,则输出(　　)。

A. 1 1　　B. 2 2　　C. 1 3　　D. 3 1

(6)输入 2 2,则输出(　　)。

A. 1 1　　B. 1 2　　C. 2 1　　D. 2 2

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	√	√	×	C	D

【分析】分析功能可知,该程序是模拟一个初始位于一个长为 n,宽为 m 的盒子的左下角有一个小球初始向正右上方移动,碰到盒子边界之后就会完全反弹,求第二次碰到盒子时小球的坐标,通过画图模拟也可以得出结果。

(1)将 16、20 行去掉,cnt 的值永远为 0,则程序会出现死循环。

(2)cin、cout 需要 iostream 库。

(3)将 17、21 行去掉,方向没有改变,则小球走出边界后一直往该方向走,程序死循环。

(4)第 1、2 次碰撞的时间复杂度均为 O(L),则总时间复杂度为 O(L)。

(5)通过模拟小球的移动,第一次撞到(2,2),第二次撞到(1,3)。

(6)当该点到达(2,2)时,x 与 y 同时到达边界,则终止结点为(2,2)。

8.【NOIP2014】

```
01 #include<iostream>
02 using namespace std;
03 const int SIZE=100;
04 int main(){
05     int p[SIZE];
06     int n,tot,i,cn;
07     tot=0;
08     cin>>n;
09     for(i=1;i<=n;i++)p[i]=1;
10     for(i=2;i<=n;i++){
11         if (p[i]==1)tot++;
12         cn=i*2;
13         while(cn<=n){
14             p[cn]=0;
15             cn+=i;
16         }
17     }
```

```
18      cout<<tot<<endl;
19      return 0;
20 }
```

●判断题

(1)n 的值为 100 时程序不会运行错误。 (　　)

(2)该程序的时间复杂度为 O(n)。 (　　)

(3)将第 12 行的 cn=i * 2 改为 cn=i,程序输出结果不变。 (　　)

(4)输入 30,输出结果为 10。 (　　)

●选择题

(5)本程序的功能为(　　)。

A. 求 n 以内的质数　　B. 求 n 以内与 n 互质的数

C. 求 n 的约数　　D. 求 n 以内与 n 不互质的数

(6)输入 100,输出结果为(　　)。

A. 24　　B. 22　　C. 25　　D. 23

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	√	√	A	C

【分析】通过阅读程序后,简单分析可知,本题是在筛选 n 以内的质数。

(1)C++的数组下标是从 0 开始的,因此最大下标为数组长度减 1。

(2)埃氏筛法的时间复杂度为 O(nloglogn)。

(3)第 11 行以后的代码都不再访问 p[i],因此此时 p[i]的值对输出不会造成任何影响。

(4)30 以内的质数有 2,3,4,5,11,13,17,19,23,29,共 10 个。

(5)此程序的功能就是用筛法求 n 以内的质数,输出质数个数。

(6)1 到 100 以内的质数为:2、3、5、7、11、13、17、19、23、29、31、37、41、43、47、53、59、61、67、71、73、79、83、89、97。

9.【NOIP2012】

```
01 #include<iostream>
02 using namespace std;
03 const int SIZE=20;
04 int data[SIZE];
05 int n,i,h,ans;
06 void merge(){
07      data[h-1]=data[h-1]+data[h];
08      h--;
09      ans++;
10 }
11 int main(){
12      cin>>n;
13      h=1;
14      data[h]=1;
```

```
15      ans=0;
16      for(i=2;i<=n;i++){
17          h++;
18          data[h]=1;
19          while(h>1&&data[h]==data[h-1])
20              merge();
21      }
22      cout<<ans<<endl;
23  }
```

●判断题

(1)去掉第 15 行,程序输出结果不变。　　(　　)

(2)输入 5,输出结果为 3。　　(　　)

(3)将第 19 行的"=="改为">=",程序输出结果会改变。　　(　　)

(4)输入 2147483647,程序一定正常运行。　　(　　)

●选择题

(5)输入 2012,输出结果为(　　)。

A. 2015　　B. 2018　　C. 2004　　D. 2006

(6)ans 的上界为(　　)。

A. n　　B. $\sqrt{n}$　　C. logn　　D. $n\sqrt{n}$

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	√	×	×	C	A

【分析】本题是统计合并的次数,合并的过程执行一次便统计一次。合并的条件是 data[h]=data[h−1],即最终的结果是将输入的一个数通过归并的方式拆分为若干个 2 的整数次方的和,而 2^m 需要 2^m-1 次合并,所以最终的结果为 n 减去 n 在二进制下 1 的个数。

(1)ans 为全局变量,初值为 0。

(2)$(5)_{10}=(101)_2$,共 2 位 1,所以结果为 5−2=3。

(3)data[h]一定小于等于 data[h−1]。

(4)数组越界,程序可能会运行错误。

(5)$(2012)_{10}=(11111011100)_2$,共 8 位 1,所以结果为 2012−8=2004。

(6)最终的结果为 n 减去 n 在二进制下 1 的个数,因此 ans 上界为 n。

10.【NOIP2011】

```
01 #include<cstring>
02 #include<iostream>
03 #include<string>
04 using namespace std;
05 const int SIZE=10000;
06 const int LENGTH=10;
07 int n,m,a[SIZE][LENGTH];
08 int h(int u,int v){
```

```
09      int ans,i;
10      ans=0;
11      for (i=1;i<=n;i++)
12          if (a[u][i]!=a[v][i])ans++;
13      return ans;
14  }
15
16  int main(){
17      int sum,i,j;
18      cin>>n;
19      memset(a,0,sizeof(a));
20      m=1;
21      while (1){
22          i=1;
23          while ((i<=n)&&(a[m][i]==1))i++;
24          if (i>n)break;
25          m++;
26          a[m][i]=1;
27          for (j=i+1;j<=n;j++)a[m][j]=a[m-1][j];
28      }
29      sum=0;
30      for (i=1;i<=m;i++)
31          for (j=1;j<=m;j++)sum+=h(i,j);
32      cout<<sum<<endl;
33      return 0;
34  }
```

●判断题

(1)第 21 行至多执行 n 次。 (　　)

(2)将第 12 行的!=改成==,结果不影响。 (　　)

(3)输入的 n 为 11,不会数组越界。 (　　)

(4)该程序可以正常运行。 (　　)

●选择题

(5)输入 7,输出结果为(　　)。

A. 7　　B. 6　　C. 57344　　D. 114514

(6)该算法的时间复杂度为(　　)。

A. O(1)　　B. O(n)　　C. $O(n^2)$　　D. $O(2^{2n})$

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	×	√	C	D

【分析】

(1)a 数组是 $0\sim2^n-1$ 的二进制表示，所以 $m=2^n$，因此第 21 行执行次数为 2^n。

(2)h(i,j)表示第 i 个数和第 j 个数有多少位二进制不同，改变逻辑符号会影响结果。

(3)因为 LENGTH=10，所以会发生数组越界。

(4)该程序确实可以正常运行。

(5)首先是对一个数进行分析，我们能够得出对于每一个数有 $S=\sum_{i=0}^{n} i * C_n^i$，所以 $2S=\sum_{i=0}^{n}(i+n-i) * C_n^i=\sum_{i=0}^{n} n * C_n^i=n * \sum_{i=0}^{n} C_n^i=n * 2^n$，所以 $S=n * 2^{n-1}$。因为有 2^n 个数，所以 $sum=2^n * n * 2^{n-1}=n * 2^{2n-1}$，即 57344。

(6)该程序的时间复杂度为 $O(2^{2n})$。

11.【NOIP2013】

```
01 #include<iostream>
02 using namespace std;
03 int main(){
04     int a,b,u,v,i,num;
05     cin>>a>>b>>u>>v;
06     num=0;
07     for (i=a;i<=b;i++)
08         if (((i%u)==0)||((i%v)==0))
09             num++;
10     cout<<num<<endl;
11     return 0;
12 }
```

●判断题

(1)u * v 可以为 0。　　(　　)

(2)去掉第 06 行，程序输出结果会发生改变。　　(　　)

(3)将第 08 行的“||”改为“&&”，程序输出结果不会发生改变。　　(　　)

(4)输入 1 9 3 4，输出结果为 4。　　(　　)

●选择题

(5)输入 51 82 3 6，输出结果为(　　)。

A. 15　　B. 11　　C. 5　　D. 16

(6)输入 97 207 4 6，输出结果为(　　)。

A. 125　　B. 50　　C. 36　　D. 75

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	√	×	√	B	C

【分析】本题功能：输入 4 个整数，a,b,u,v，问在[a,b]区间内有几个数是 u 或 v 的倍数。

(1)u * v=0 即 u、v 中至少一个为 0，但是 u、v 作为除数不能为 0。

(2)num 为局部变量,必须赋初值。

(3)修改后该代码功能会变为求在[a,b]区间内同时是 u 和 v 的倍数的数的个数。

(4)1 到 9 中 3 的倍数为 3,9,4 的倍数为 4,8。

(5)3 和 6 为倍数关系,因此只要统计 3 的倍数即可。

(6)答案即为 4 的倍数个数+6 的倍数个数−4 和 6 的最小公倍数的倍数个数。

12.【NOIP2018】

```
01 #include<iostream>
02 using namespace std;
03 const int N=110;
04 bool isUse[N];
05 int n,t;
06 int a[N],b[N];
07 bool isSmall( ){
08     for (int i=1;i<=n;++i)
09         if (a[i]!=b[i])return a[i]<b[i];
10     return false;
11 }
12 bool getPermutation(intpos){
13     if (pos>n){
14         return isSmall();
15     }
16     for (int i=1;i<=n;++i){
17         if (!isUse[i]){
18             b[pos]=i;isUse[i]=true;
19             if (getPermutation(pos+1)){
20                 return true;
21             }
22             isUse[i]=false;
23         }
24     }
25     return false;
26 }
27 void getNext(){
28     for (int i=1;i<=n;++i){
29         isUse[i]=false;
30     }
31     getPermutation(1);
32     for (int i=1;i<=n;++i){
```

```
33          a[i]=b[i];
34      }
35 }
36 int main(){
37      scanf("%d%d",&n,&t);
38      for (int i=1;i<=n;++i){
39          scanf("%d",&a[i]);
40      }
41      for (int i=1;i<=t;++i){
42          getNext();
43      }
44      for (int i=1;i<=n;++i){
45          printf("%d",a[i]);
46          if (i==n)putchar('\n');else putchar('');
47      }
48      return 0;
49 }
```

●判断题

(1)若 t=0,则输入的排列与输出的排列相同。　(　　)

(2)此程序的功能是求出给定的排列的上 t 个排列。　(　　)

(3)若将 46 行去掉,则输出效果不变。　(　　)

(4)将 32～34 行去掉,则输出效果不变。　(　　)

●选择题

(5)该程序的时间复杂度为(　　)。

A. O(nlogt)　　B. O(nt)　　C. $O(nt^2)$　　D. $O(n^{2t})$

(6)若输入 3 1 1 2 3,则输出(　　)。

A. 1 2 3　　B. 1 3 2　　C. 2 1 3　　D. 3 2 1

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	×	×	×	B	B

【分析】不难看出其实就是在求给定排列的下 t 个排列结果。

(1)若 t=0,则排列没有变化,则输入的排列与输出的排列相同。

(2)此程序的功能是求出给定排列的下 t 个排列。

(3)将换行符和空格去掉,数字会黏在一起。

(4)若将 32～34 行去掉,则 a 数组没有被操作,输出与输入排列相同。

(5)求下一个排列的时间复杂度是 O(n)的,执行 t 次,时间复杂度为 O(nt)。

(6)易知 1 2 3 的下 1 个排列为 1 3 2。

13.【NOIP2009】

```
01 #include<iostream>
02 using namespace std;
03 #define maxn 50
04 const int y=2009;
05 int n,c[maxn][maxn],i,j,s=0;
06 int main()
07 {
08     cin>>n;
09     c[0][0]=1;
10     for (i=1;i<=n;i++)
11     {
12         c[i][0]=1;
13         for (j=1;j<i;j++)
14             c[i][j]=c[i-1][j-1]+c[i-1][j];
15         c[i][i]=1;
16     }
17     for (i=0;i<=n;i++)
18         s=(s+c[n][i])% y;
19     cout<<s<<endl;
20     return 0;
21 }
```

●判断题

(1)该程序的第 03 行需要加分号才能正常运行。 (　　)

(2)该程序可能输出 2009。 (　　)

(3)第 05 行的 s 可以不用赋初值。 (　　)

(4)输入 17,输出 486。 (　　)

●选择题

(5)该程序的实质是(　　)。

A. 快速排序　　B. 区间最大和　　C. 区间前缀和　　D. 杨辉三角

(6)该算法的时间复杂度级别为(　　)。

A. 线性时间　　B. 对数时间　　C. 平方时间　　D. 常数时间

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	√	×	D	C

【分析】通过阅读程序可知,本题是在求杨辉三角第 n 行数的和,对 y 取模的结果。

(1)define 的定义本来就不用加分号。

(2)因为取模了,所以不可能超过 2008。

(3)全局变量 s 自动初始化为 0,故 s 可以不赋初值。

(4)本程序求的是杨辉三角第 n 行数的和并对输入的 y 取模,递推方程为

c[i][j]=c[i-1][j-1]+c[i-1][j]。根据二项式定理$(a+b)^n$的各项系数之和为2的n次方，而且展开式中奇数项二项式系数之和等于偶数项二项式系数之和等于2的(n-1)次方，当n=17时，有系数和为$2^{17}=131072$，而131072%2009=487，本题也可以通过模拟来得到答案。

(5)这个算法的实质就是杨辉三角。

(6)这道题的时间复杂度级别是平方时间。

14.【NOIP2013】

```
01 #include<iostream>
02 using namespace std;
03 int main(){
04     const int SIZE=100;
05     int n,f,i,j,t,left,right,middle,a[SIZE];
06     cin>>n>>f;
07     for (i=1;i<=n;i++)
08         cin>>a[i];
09     for (i=1;i<=n;i++)
10         for (j=1;j<=i;j++)
11         if (a[i]>=a[j])
12         {
13             t=a[j];
14             a[j]=a[i];
15             a[i]=t;
16         }
17     left=1;
18     right=n;
19     do {
20           middle=(left+right)/2;
21           if (f<=a[middle])
22             right=middle;
23           else
24             left=middle+1;
25     } while (left<right);
26     cout<<left<<endl;
27     return 0;
28 }
```

●判断题

(1)将第04行的程序移动到02、03行的中间，程序能够正常运行。　(　　)

(2)将第11行的“=”删除，运行结果会改变。　(　　)

(3)将第07行的“i=1;i<=n;”改为“i=0;i<n;”运行结果不会改变。　(　　)

(4)若第08行输入n个相同的数字，程序最后输出的left值为1。　(　　)

●选择题

(5)当 n=5,f=7,a={8,4,7,5,6}时,则结果为(　　)。

A. 3　　B. 4　　C. 5　　D. 7

(6)输入仍是第(5)题的条件,将第 11 行的“>=”改为“<=”,则结果为(　　)。

A. 3　　B. 4　　C. 5　　D. 7

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	√	×	×	√	B	C

【分析】

(1)常量定义位置放在 main 函数内外不影响。

(2)排序时,交换相等的数是无效的。

(3)原二分范围为[1,n],若修改那么二分范围就改变了。

(4)由模拟可知,右指针不断左移直至左右指针都为 1。

(5)程序会将数据升序排序,在其中二分检索出寻找第一个大于等于 f 的元素的位置。

(6)程序会将数据降序排序,由于大于等于 f 的数在二分的位置左侧,所以找不到大于等于 f 的数,左指针不断右移直至左右指针都为 5。

15.【NOIP2009】

```
01 #include<iostream>
02 using namespace std;
03 int n,m,i,j,p,k;
04 int a[100],b[100];
05 int main()
06 {
07     cin>>n>>m;
08     a[0]=n;i=0;p=0;k=0;
09     do{
10         for (j=0;j<i;j++)
11             if (a[i]==a[j])
12             {
13                 p=1;k=j;break;
14             }
15         if (p)break;
16         b[i]=a[i]/m;
17         a[i+1]=a[i]% m*10;
18         i++;
19     } while (a[i]!=0);
20     cout<<b[0]<<".";
21     for (j=1;j<k;j++)cout<<b[j];
22     if (p)cout<<"(";
23     for (j=k;j<i;j++)cout<<b[j];
```

```
24      if (p)cout<<")";
25      cout<<endl;
26      return 0;
27 }
```

●判断题

(1)输入的 m 可以为任意自然数。　　(　　)

(2)输入－1 2,会输出－0.5。　　(　　)

(3)输入 1 2 时,程序会输出 0.(5)。　　(　　)

(4)输入 0 1 时,程序会输出 0。　　(　　)

●选择题

(5)输入 5 13,输出为(　　)。

A. 0.(384615)　　B. 0　　C. 18　　D. 8

(6)删去下列第(　　)行后,用(5)中的输入运行代码后输出结果发生改变。

A. 25　　B. 26　　C. 21　　D. 24

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	×	×	×	×	A	D

【分析】本程序是将分数变成小数,循环部分用括号括起来。在求小数的过程中,如果出现了之前出现过的数字(指余数)则开始循环,p 就是判断是否循环的的标志,p＝1 表示循环,p＝0 表示不循环。

(1)m＝0 时运行错误。

(2)模拟可知,会输出 0.0－5。

(3)输入 1 2 时,程序会输出 0.05。

(4)输入 0 1 时,程序会输出 0.0。

(5)由上文分析知,求出 5/13 的小数形式,在循环节左右加上括号即可。

(6)删去 24 行后运行(5)中的输入,输出结果会少一个右括号。

第五章　完善程序

第 1 节　数组

1.【NOIP2013】序列重排

全局数组变量 a 定义如下：const int SIZE=100;int a[SIZE],n;它记录着一个长度为 n 的序列 a[1],a[2],…,a[n]。

现在需要一个函数，以整数 p(1≤p≤n)为参数，实现如下功能：将序列 a 的前 p 个数与后 n−p 个数对调，且不改变这 p 个数(或 n−p 个数)之间的相对位置。例如，长度为 5 的序列 1,2,3,4,5，当 p=2 时重排结果为 3,4,5,1,2。

有一种朴素的算法可以实现这一需求，其时间复杂度为 O(n)、空间复杂度为 O(n)：

```
void swap1(intp){
    int i,j,b[SIZE];
    for (i=1;i<=p;i++)
        b[ ① ]=a[i];
    for (i=p+1;i<=n;i++)
        b[i-p]= ② ;
    for (i=1;i<= ③ ;i++)
        a[i]=b[i];
}
```

我们也可以用时间换空间，使用时间复杂度为 $O(n^2)$、空间复杂度为 O(1)的算法：

```
void swap2(intp){
    int i,j,temp;
    for (i=p+1;i<=n;i++){
        temp=a[i];
        for (j=i;j>= ④ ;j--)
            a[j]=a[j-1];
        ⑤ =temp;
    }
}
```

●选择题

(1)①处应填(　　)。

A. p+i　　B. n-p+i

C. n-p+i-1　　D. i

(2)②处应填(　　)。

A. a[i]　　B. a[i-p]

C. a[n-i]　　D. b[i]

(3)③处应填(　　)。

A. n　　B. p

C. p+1　　D. n-1

(4)④处应填(　　)。

A. 1　　B. i-p

C. n-i　　D. i-p+1

(5)⑤处应填(　　)。

A. a[i-p+1]　　B. a[i]

C. a[i-p]　　D. a[i-1]

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	B	A	A	D	C

【分析】本题一共给出了2种算法：

第一种是开一个临时数组，将后n－p个数和前p个数依次拷贝到临时数组中，再把临时数组复制给当前数组。这一朴素的算法时间和空间复杂度均为O(n)。

第二种是每次把整个数组往右移一位(最右边的挪到最左边)，这样右移p次即是最终的数组。省去了O(n)的临时空间，但是时间复杂度上升至$O(n^2)$。

(1)由于不能改变前p个数(或后n－p个数)之间的相对位置，所以要将第i位上的数挪到第n－p＋i位上。

(2)将第i位上的数挪到第i－p位上。

(3)将对调好的临时数组拷贝到原数组(即答案数组)。

(4)将原数组的前p个数均往右移一位。

(5)将第i位(后n－p位)上的数挪到第i－p位(前n－p位)上。

2.【NOIP2014】最大子矩阵和

给出m行n列的整数矩阵，求最大的子矩阵和(子矩阵不能为空)。

输入第一行包含两个整数m和n，即矩阵的行数和列数。之后m行，每行n个整数，描述整个矩阵。程序最终输出最大的子矩阵和。

```
#include<iostream>
using namespace std;
const int SIZE=100;
int matrix[SIZE+1][SIZE+1];
int rowsum[SIZE+1][SIZE+1];
//rowsum[i][j]记录第i行前j个数的和
int m,n,i,j,first,last,area,ans;
int main(){
    cin>>m>>n;
    for (i=1;i<=m;i++)
        for (j=1;j<=n;j++)
            cin>>matrix[i][j];
    ans=matrix[  ①  ];
    for (i=1;i<=m;i++)
        [  ②  ];
    for (i=1;i<=m;i++)
```

```
        for (j=1;j<=n;j++)
            rowsum[i][j]= ③ ;
    for (first=1;first<=n;first++)
        for (last=first;last<=n;last++){
            ④ ;
            for (i=1;i<=m;i++){
                area+= ⑤ ;
                if (area>ans)ans=area;
                if (area<0)area=0;
            }
        }
    cout<<ans<<endl;
    return 0;
}
```

选择题

(1)①处应填(　　)。

A. [0][0]　　B. [0][n]

C. [m][0]　　D. [1][1]

(2)②处应填(　　)。

A. rowsum[i][0]=matrix[i][1]　　B. rowsum[i][0]=1

C. rowsum[i][0]=0　　D. rowsum[i][0]=ans;

(3)③处应填(　　)。

A. rowsum[i][j-1]　　B. rowsum[i-1][j]+matrix[i][j]

C. matrix[i][j]　　D. rowsum[i][j-1]+matrix[i][j]

(4)④处应填(　　)。

A. area=0　　B. ans=area

C. area=ans　　D. ans=0

(5)⑤处应填(　　)。

A. matrix[i][last]　　B. rowsum[i][last]-rowsum[i][first-1]

C. matrix[i][fist]　　D. rowsum[i][last]-rowsum[i][first]

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	D	C	D	A	B

【分析】本题解决最大子矩阵和所用的算法:计算数组 rowsum;枚举子矩阵的左边界 first 和右边界 last,将原问题转化为求解一维的最大子段和问题,用贪心法即可解决。

(1)因为所求最大子矩阵和所涉及的子矩阵不能为空,必须有一个初值,所以我们需要将 ans 设置为矩阵的左上角元素,也就是[1][1],取其他单元格的值也可以。

(2)因为后面要求每行前缀和,所以需要将 0 列清零,用于之后的统计。

(3)求当前行每一列的前缀和,使用前缀和的方法统计每行的 sum 值。

(4)从 first 列到 last 列之间求最大子段和,需要将当前的值初始化为 0。

(5)这里求第 i 行的 first 列到 last 列之间的数值和,这里采用前缀和方法来加速计算过程。

3.【NOIP2008】矩阵中的数字

有一个 n * n(1≤n≤5000)的矩阵 a,对于 1≤i<n,1≤j≤n,a[i][j]<a[i+1][j],a[j][i]<a[j][i+1]。即矩阵中左右相邻的两个元素,右边的元素一定比左边的大。上下相邻的两个元素,下面的元素一定比上面的大。给定矩阵 a 中的一个数字 k,找出 k 所在的行列(注意:输入数据保证矩阵中的数各不相同)。

```
#include<iostream>
using namespace std;
int n,k,answerx,answery;
int a[5001][5001];
void FindKPosition()
{
    int i=n,j=n;
    while (j>0)
    {
        if (a[n][j]<k) break;
        j--;
    }
    ____①____;
    while (a[i][j]!=k)
    {
        while (____②____&&i>1)i--;
        while (____③____&&j<=n)j++;
    }
    ____④____;
    ____⑤____;
}
int main()
{
    int i,j;
    cin>>n;
    for (i=1;i<=n;i++)
        for (j=1;j<=n;j++)
            cin>>a[i][j];
    cin>>k;
    FindKPosition();
    cout<<answerx<<""<<answery<<endl;
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. j--　　B. j++

C. i++　　D. i--

(2)②处应填(　　)。

A. a[i][j]>k　　B. a[i][j]<k

C. a[i][j]<=k　　D. a[i][j]!=k

(3)③处应填(　　)。

A. a[i][j]>k　　B. a[i][j]<k

C. a[i][j]>=k　　D. a[i][j]!=k

(4)④处应填(　　)。

A. answerx=i+1　　B. answerx=i-1

C. answerx=j　　D. answerx=i

(5)⑤处应填(　　)。

A. answery=j+1　　B. answery=j-1

C. answery=i　　D. answery=j

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	B	A	B	D	D

【分析】FindKPosition 这个函数就是来找 k 的位置的过程。

(1)根据题目的条件不难发现一个性质,就是 a[n][j]是 n*j 个格子中最大的一个数,因此 k 先与 a[n][j]进行比较,此时找到的 j 表示的意义是 n*j 个格子里的数都小于 k,也就是 k 所在的位置(answeri,answerj)一定满足 answerj>j,所以①就该填 j++。

(2)后面的 i--,可以知道 k<a[i][j],因为上面的小,所以②填 a[i][j]>k。

(3)同理根据 j++知道 k>a[i][j],因为右边的大,所以③填 k>a[i][j]。

(4)因为数据不会重复,所以经过循环就发现 i,j 就是答案,只要赋值即可,所以④为 answerx=i。

(5)同理⑤为 answery=j。

第 2 节　字符处理

1.【NOIP2008】字符串替换

给定一个字符串 S(S 仅包含大小写字母),下面的程序将 S 中的每个字母用规定的字母替换,并输出 S 经过替换后的结果。程序的输入是两个字符串,第一个字符串是给定的字符串 S,第二个字符串 S'由 26 个字母组成,它是 a—z 的任一排列,大小写不定,S'规定了每个字母对应的替换字母:S'中的第一个字母是字母 A 和 a 的替换字母,即 S 中的 A 用该字母的大写替换,S 中的 a 用该字母的小写替换;S'中的第二个字母是字母 B 和 b 的替换字母,即 S 中的 B 用该字母的大写替换,S 中的 b 用该字母的小写替换;……以此类推。

```
#include<iostream>
#include<string.h>
char change[26],str[5000];
using namespace std;
void CheckChangeRule(){
    int i;
    for (i=0;i<26;i++)
    {
        if (__①__)
            change[i]-='A'-'a';
    }
}
voidChangeString(){
    int i;
    for (i=0;i<strlen(str);i++)
    {
        if (__②__)
            str[i]=change[str[i]-'A']-'a'+'A';
        else
            __③__
    }
}
int main()
{
    int i;
    cin>>str;
    cin>>change;
    CheckChangeRule();
    __④__
    cout<<str<<endl;
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. change[i]>='a'&&change[i]<='z'　　B. change[i]<'A'||change[i]>'Z'

C. change[i]>='A'&&change[i]<='Z'　　D. change[i]<'a'&&change[i]>'z'

(2)②处应填(　　)。

A. str[i]>='a'&&str[i]<='z'　　B. str[i]<'A'&&str[i]>'Z'

C. str[i]<'a'&&str[i]>'z'　　D. str[i]>='A'&&str[i]<='Z'

(3)③处应填(　　)。

A. change[str[i]-'a']=str[i];　　B. str[i]=change[str[i]-'a'];

C. str[i]=change[str[i]-'A'];　　D. change[str[i]-'z']=str[i];

(4)④处应填(　　)。

A. int len=strlen(str);　　B. Changestring();

C. ChangeString();　　D. changeString();

【答案】

题号	(1)	(2)	(3)	(4)
答案	C	D	B	C

【分析】程序功能概述：

(1)从①处下一行 change[i]的赋值语句可知，change[i]最终存储的是字母表中第 i 个字母对应的另一个小写字母(0≤i<26)，而 CheckChangeRule 是对于 change 的预处理函数，所以①如果 change[i]是大写字母，则转换成小写字母。

(2)change[str[i]-'A'-'a'+'A'的功能是将 str[i]这个大写字母变成应该替换成的另一个大写字母，所以②的 str[i]应是大写字母。

(3)因为 str[i]是小写字母，可以直接利用 change[str[i]-'a']将 str[i]变成应该替换成的另一个小写字母。

(4)主程序调用 ChangeString()函数也是必须的。

2.【NOIP2016】读入整数

请完善下面的程序，使得程序能够读入两个 int 范围内的整数，并将这两个整数分别输出，每行一个。

输入的整数之间和前后只会出现空格或者回车，输入数据保证合法。

例如：

输入

```
  123
 -789
```

输出

```
  123
 -789
```

```
#include<iostream>
using namespace std;
int readint(){
```

```
    int num=0;          //存储读取到的整数
    int negative=0;     //负数标识
    char c;             //存储当前读取到的字符
        c=cin.get();
    while ((c<'0'||c>'9')&&c!='-')
        c=  ①  ;
    if (c=='-')
        negative=1;
    else
         ②  ;
    c=cin.get();
    while (  ③  ){
         ④  ;
        c=cin.get();
    }
    if (negative==1)
         ⑤  ;
    return num;
}
int main(){
    int a,b;
    a=readint();
    b=readint();
    cout<<a<<endl<<b<<endl;
    return 0;
}
```

选择题

(1)①处应填(　　)。

A. c-'0'　　B. '0'

C. c+'0'　　D. cin.get()

(2)②处应填(　　)。

A. num=0　　B. num=c-'0'

C. num=c-'a'　　D. num=c

(3)③处应填(　　)。

A. c>='0'&&c<='9'　　B. c>='a'&&c<='z'

C. c<'0'||c>'9'　　D. c!='-'

(4)④处应填(　　)。

A. num=num+c-'0'　　B. num=num*10+c-'0'

C. num=num+c　　D. num=num*10+c

(5)⑤处应填(　　)。

A. num=-num　　B. num=num+num

C. num--　　D. num++

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	D	B	A	B	A

【分析】此题主要实现的是 readint 函数，该函数通过逐个读取字符的方法读取一个整数。

该函数的正确流程应该是：过滤掉非数值字符，读取连续数值字符直到遇到非数值字符，同时将读取的字符转化为数值，最后输出数值。

(1)第一个 while 循环实现的功能是过滤非数字，这一步只需要不断读取就好，所以仿照上文写 cin.get()。

(2)当退出过滤非数字的 while 循环时有两种情况：读取到了数字的第一个字符或者读取到了负数标记。else 分支对应的是读取到了数值，所以应该用字符 c 更新 num 的大小。注意字符型的 c 必须减去'0'或 48 才能得到对应整型的数值 num。

(3)第三空是读取连续的数字，因此循环条件为读取到的字符是数字字符。

(4)第四空是用读取的数字字符对 num 值进行更新，就是 num*10+c-'0'。

(5)第五空对应的是负数的处理，如果负数标记为 1，那么应该取相反数。

3.【NOIP2009】最大连续子段和

给出一个数列(元素个数不多于 100)，数列元素均为负整数、正整数、0。请找出数列中的一个连续子数列，使得这个子数列中包含的所有元素之和最大，在和最大的前提下还要求该子数列包含的元素个数最多，并输出这个最大和以及该连续子数列中元素的个数。例如数列为 4 −5 3 2 4 时，输出 9 和 3；数列为 1 2 3 −5 0 7 8 时，输出 16 和 7。

```
#include<iostream>
using namespace std;
int a[101];
int n,i,ans,len,tmp,beg;
int main(){
    cin>>n;
    for (i=1;i<=n;i++)
        cin>>a[i];
    tmp=0;
    ans=0;
    len=0;
    beg=  ①  ;
    for (i=1;i<=n;i++){
        if (tmp+a[i]>ans){
            ans=tmp+a[i];
            len=i-beg;
        }
        else if (  ②  &&i-beg>len)
            len=i-beg;
        if (tmp+a[i]  ③  ){
```

```
                beg=  ④  ;
                tmp=0;
            }
            else
                ⑤ ;
        }
        cout<<ans<<""<<len<<endl;
        return 0;
    }
```

●选择题

(1)①处应填(　　)。

A. 1　　B. 0

C. n　　D. 101

(2)②处应填(　　)。

A. tmp+a[i]>ans　　B. tmp+a[i]=ans

C. tmp+a[i]>n　　D. tmp+a[i]==ans

(3)③处应填(　　)。

A.<0　　B. <ans

C. <mp　　D. <min(0,ans,tmp)

(4)④处应填(　　)。

A. 1　　B. 0

C. n　　D. i

(5)⑤处应填(　　)。

A. tmp+ = a[i]　　B. tmp= i+ 1

C. beg= i　　D. beg= i+ 1

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	B	D	A	D	A

【分析】首先来解释一下变量，ans 就是所求的和，len 就是长度，beg 就是起始位置的前一个位置，而 i－beg 正好就是当前所选序列的长度，tmp 是当前序列的和。

(1)①就应该是起始位置 1 的前一个，所以①填 0。

(2)逐一扫描，因为是连续和，所以一重循环即可，首先判断当前这个数是否使解更优，更优就填进去，否则再看是否和最优解相等，相等的话根据题目，要长度更长的，再进行长度的判断，因此②填 tmp+a[i]==ans。

(3)如果当前解已经为负数，且下一个数是正数，那么肯定把当前解全部舍掉，取下一个数，如果下一个数是负数，那加上当前解肯定比它本身更小，因此当前解也会被舍掉，所以无论下一个数是正是负，当前解都会被舍掉，所以重新清 0。因此③填<0。

(4)tmp 归零，beg 是起始位置的前一个，而起始位置是下一个数，所以④填 i。

(5)如果当前解是正的，那么则继续留到下一位，因此⑤填 tmp+=a[i]。

第3节 枚举算法

1.【NOIP2014】数字删除

下面程序的功能是将字符串中的数字字符删除后输出，请填空。

```
#include<iostream>
using namespace std;
int delnum(char *s){
    int i,j;
    j=0;
    for (i=0;s[i]!='\0';i++)
        if (s[i]<'0'  ①  s[i]>'9'){
            s[j]=s[i];
            ②  ;
        }
    return ③ ;
}
const int SIZE=30;
int main(){
    char s[SIZE];
    int len,i;
    cin.getline(s,sizeof(s));
    len=delnum(s);
    for (i=0;i<len;i++)cout<< ④ ;
    cout<<endl;
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. &&　　B. &&s[i]>='a'&&s[i]<='z'&&

C. ||　　D. &&s[i]>='A'&&s[i]<='Z'&&

(2)②处应填(　　)。

A. i++　　B. i= j

C. ++j　　D. j=i

(3)③处应填(　　)。

A. j　　B. i

C. s[j]　　D. s[i]

(4)④处应填(　　)。

A. i　　B. s[i]

C. (!s[i])　　D. (s[i]==1)

【答案】

题号	(1)	(2)	(3)	(4)
答案	C	C	A	B

【分析】此题考查简单的字符串处理。delnum就是删数字的函数，len是删完后的字符串长度，逐个访问字符s[i]，如果不是数字字符，那就是有效字符，保存至s[j]，j是用来记非数字的最后一位。

①中需要判断当前字符是不是数字，字符为不为数字的条件此处应该为或关系。

②中需要将i位置上的字符赋值给j位置。这之后需要更新非数字的最后一位的位置，因此需要将j加1。

③中需要范围处理后的字符串的长度，因此我们需要返回j。

④需要输出最后的结果，因此需要输出s[i]。

(1)应用逻辑或判断当前字符是不是数字。

(2)j为当前有几个非数字的计数器，为了下一次更新需要将j加1。

(3)需要返回处理后的字符串的长度即j。

(4)需要输出最后的结果即s[i]。

2.【NOIP2012】坐标统计

输入n个整点在平面上的坐标。对于每个点，可以控制所有位于它左下方的点(即x、y坐标都比它小)，它可以控制的点的数目称为“战斗力”。依次输出每个点的战斗力，最后输出战斗力最高的点的编号(如果若干个点的战斗力并列最高，输出其中最大的编号)。

```
#include<iostream>
using namespace std;
const int SIZE=100;
int x[SIZE],y[SIZE],f[SIZE];
int n,i,j,max_f,ans;
int main(){
    cin>>n;
    for (i=1;i<=n;i++)
        cin>>x[i]>>y[i];
    max_f=0;
    for (i=1;i<=n;i++){
        f[i]=  ①  ;
        for (j=1;j<=n;j++){
            if (x[j]<x[i]&&  ②  )
                ③ ;
        }
        if (  ④  ){
            max_f=f[i];
            ⑤ ;
        }
    }
    for (i=1;i<=n;i++)
        cout<<f[i]<<endl;
    cout<<ans<<endl;
}
```

●选择题

(1)①处应填(　　)。

A. i　　B. 0

C. x[i]　　D. y[i]

(2)②处应填(　　)。

A. y[i]<y[j]　　B. y[j]<y[i]

C. y[i]<=y[j]　　D. y[j]<=y[i]

(3)③处应填(　　)。

A. f[i]=x[i]　　B. f[i]=0

C. f[i]++　　D. f[i]=y[i]

(4)④处应填(　　)。

A. f[i]>max_f　　B. f[i]<max_f

C. f[i]>=max_f　　D. f[i]<=max_f

(5)⑤处应填(　　)。

A. ans=i　　B. ans<max_f

C. ans=f[i]　　D. ans=0

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	B	B	C	C	A

【分析】先大致浏览一遍，可以得出f[i]应该是第i个点的战斗力大小，ans应该是战斗力最高的点的编号，max_f是战斗力最高的点的战斗力。所以第一空明显应该是赋初值，第二、三、五三空都比较简单，关键在于第四空，要注意题目中说到，战斗力相同要取编号大的，所以应该要加等于号。

(1)第一个空赋初值，即f[i]=0。

(2)只有x坐标和y坐标都大于另一个点，才可以控制该点，即x[j]<x[i]和y[j]<y[i]。

(3)如果一个点被另一个点控制，则该点的战斗力应加1，即f[i]++。

(4)取最高战斗力且保证最大编号，即f[i]>=max_f。

(5)获取最大战斗力的编号，即ans=i。

3.【NOIP2011】子矩阵

输入一个n1 * m1的矩阵a和一个n2 * m2的矩阵b，问a中是否存在子矩阵和b相等。若存在，输出所有子矩阵左上角的坐标，若不存在，输出“There is no answer”。

```
#include<iostream>
using namespace std;
const int SIZE=50;
int n1,m1,n2,m2,a[SIZE][SIZE],b[SIZE][SIZE];
int main(){
    int i,j,k1,k2;
    bool good,haveAns;
    cin>>n1>>m1;
```

```
    for (i=1;i<=n1;i++)
        for (j=1;j<=m1;j++)
            cin>>a[i][j];
    cin>>n2>>m2;
    for (i=1;i<=n2;i++)
        for (j=1;j<=m2;j++)
            ____①____;
    haveAns=false;
    for (i=1;i<=n1-n2+1;i++)
        for (j=1;j<=____②____;j++){
            ____③____;
                for (k1=1;k1<=n2;k1++)
                    for (k2=1;k2<=____④____;k2++){
                        if (a[i+k1-1][j+k2-1]!=b[k1][k2])
                            good=false;
                    }
                if (good){
                    cout<<i<<''<<j<<endl;
                    ____⑤____;
                }
            }
    if (!haveAns)
        cout<<"Thereisnoanswer"<<endl;
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. cin>>b[i][j]　　B. cin>>a[i][j]

C. cin>>b[n1][m1]　　D. cin>>b[n2][m2]

(2)②处应填(　　)。

A. m1　　B. m1-m2+1

C. m1-1　　D. m1+1

(3)③处应填(　　)。

A. good=0　　B. good='1'

C. good=false　　D. good=1

(4)④处应填(　　)。

A. k1+1　　B. m2

C. m2-1　　D. k1

(5)⑤处应填(　　)。

A. break　　B. return

C. haveAns=true　　D. haveAns=false

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	A	B	D	B	C

【分析】可以发现一开始并没有读入 b,所以第一空就应该是读入。然后下面是一个暴力的枚举,其中 i 和 j 变量的枚举表示子矩阵在 A 中的起点(左上角)。k1,k2 在枚举子矩阵的长度。根据上一个 for 的条件可以推出第二空和第四空。第三空联系上下文应该是给 good 赋初值,最后一空要注意不是跳出循环应该是修改 haveAns 的值。

(1)读入 b,即 cin>>b[i][j]。

(2)暴力枚举起点,即 m1−m2+1。

(3)赋给 good 初值,即 good=true。

(4)暴力枚举,由(2)可以推出来填 m2。

(5)有解则修改 haveAns 的值,即 haveAns=true。

4.【NOIP2015】打印月历

输入月份 m(1≤m≤12),按一定格式打印 2015 年第 m 月的月历。

例如,2020 年 1 月的月历打印效果如下(第一列为周日):

```
S    M    T    W    T    F    S
                1    2    3    4
5    6    7    8    9   10   11
12   13   14   15   16   17   18
19   20   21   22   23   24   25
26   27   28   29   30   31
```

```
#include<iostream>
using namespace std;
const int dayNum[]={-1,31,__①__,31,30,31,30,31,31,30,31,30,31};
int m,offset,i;
int main(){
    cin>>m;
    cout<<"S\tM\tT\tW\tT\tF\tS"<<endl;//'\t'为 TAB 制表符
    offset=3;
    for (i=1;i<m;i++)
        offset=(offset+__②__)%7;
    for (i=0;i<offset;i++)
        cout<<'\t';
    for (i=1;i<=__③__;i++){
        cout<<i;
        if (i==dayNum[m]__④____⑤__%7==0)
            cout<<endl;
        else
            cout<<'\t';
    }
```

```
    return 0;
}
```

选择题

(1)①处应填(　　)。

A. 28　　B. 29

C. 30　　D. 31

(2)②处应填(　　)。

A. dayNum[0]　　B. dayNum[i-1]

C. dayNum[i]　　D. dayNum[3]

(3)③处应填(　　)。

A. dayNum[m]　　B. dayNum[m*m]

C. m　　D. m*m

(4)④处应填(　　)。

A. &&　　B. ||

C. !　　D. ==

(5)⑤处应填(　　)。

A. dayNum[offset]　　B. !

C. (offset+dayNum[0])　　D. (offset+i)

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	B	C	A	B	D

【分析】该程序实现了月历的输出，关键是要弄清楚 offset 的作用。offset 是偏移量，主要表示月份的开头，即该月份是从星期几开始的，这里的 1 月份从周三开始，前面空 3 天，即 offset=3。

(1)该年是闰年，所以 2 月有 29 天。

(2)offset 代表该月份是从星期几开始的，即每个月前面要空 offset 天，推断出答案为 C。

(3)dayNum[m]代表此月的天数，所以 i 从 1 到 dayNum[m]。

(4)输出完毕或者到周六都要换行，所以用逻辑或。

(5)(offset+i)%7 判断是否到了周六。

5.【NOIP2012】排列数

输入两个正整数 n,m(1≤n≤20,1≤m≤n)，在 1～n 中任取 m 个数，按字典序从小到大输出所有这样的排列。

例如：

输入

3 2

输出

1 2

1 3

2 1

2 3

3 1

3 2

```
#include<iostream>
#include<cstring>
using namespace std;
const int SIZE=25;
bool used[SIZE];
int data[SIZE];
int n,m,i,j,k;
bool flag;
int main(){
    cin>>n>>m;
    memset(used,false,sizeof(used));
    for (i=1;i<=m;i++){
        data[i]=i;
        used[i]=true;
    }
    flag=true;
    while (flag){
        for (i=1;i<=m-1;i++)cout<<data[i]<<"";
        cout<<data[m]<<endl;
        flag= ①  ;
        for (i=m;i>=1;i--){
            ②  ;
            for (j=data[i]+1;j<=n;j++)
                if (!used[j]){
                    used[j]=true;
                    data[i]= ③  ;
                    flag=true;
                    break;
                }
            if (flag){
                for (k=i+1;k<=m;k++)
                    for (j=1;j<= ④  ;j++)
                        if (!used[j]){
                            data[k]=j;
                            used[j]=true;
                            break;
                        }
                ⑤  ;
            }
        }
    }
}
```

●选择题

(1)①处应填(　　)。

A. 0　　B. 1

C. !flag　　D. used[m]

(2)②处应填(　　)。

A. used[data[i]]=false　　B. used[i]=false

C. used[data[i]]=true　　D. used[i]=true

(3)③处应填(　　)。

A. !flag　　B. data[j]

C. j　　D. flag

(4)④处应填(　　)。

A. n+m　　B. k

C. n　　D. m

(5)⑤处应填(　　)。

A. return 0　　B. continue

C. !flag　　D. break

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	A	A	C	C	D

【分析】used[i]==0 表示数字 i 还没有被使用过，每次倒着找到第一个能变大的数字然后变大，接着把后面的数字直接从小到大安排，就生成了一个新的组合。flag 是标记能不能找到一个新的排列，第一层的循环意义是把排列中的第 i 位清零，就是第二个空需要实现的功能，如果找到一个排列，就把后面的排列补全(在剩余的元素中找到最小的排列)，然后跳出循环，就是第四和第五个空的意思。

(1)flag 是标记能不能找到一个新的排列，因此在 while 循环的开始应标记为 0。

(2)used[i]==0 表示数字 i 还没有被使用过，而此处是要弃用数字 i，所以标记为 0。

(3)把第 i 个位置存放的数字更改为 j。

(4)查找数字的范围是 1～n。

(5)如果找到一个排列就把后面的排列补全(在剩余的元素中找到最小的排列)，然后跳出循环。

第4节 链表

1.【NOIP2018】简单链表

对于一个1到n的排列P(即1到n中每一个数在P中恰好出现了一次)，令q_i为第i个位置之后第一个比P_i值更大的位置，如果不存在这样的位置，则$q_i=n+1$。

举例来说，如果n=5且P为15423，则q为26656。

下列程序读入了排列P，使用双向链表求解了答案。试补全程序，数据范围为$1\leqslant n\leqslant 10^5$。

```
#include<iostream>
using namespace std;
const int N=100010;
int n;
int L[N],R[N],a[N];
int main()
{
    cin>>n;
    for (int i=1;i<=n;++i){
        int x;
        cin>>x;
        ____①____;
    }
    for (int i=1;i<=n;++i){
        R[i]=____②____;
        L[i]=i-1;
    }
    for (int i=1;i<=n;++i){
        L[____③____]=L[a[i]];
        R[L[a[i]]]=R[____④____];
    }
    for (int i=1;i<=n;++i){
        cout<<____⑤____<<" ";
    }
    cout<<endl;
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. a[i]=x　　B. a[x]=i

C. x=a[i]　　D. a[x]=*i

(2)②处应填(　　)。

A. i+1　　B. i*2

C. n-i　　D. i+1

(3)③处应填(　　)。

A. R[a[i]]　　B. L[a[i]]

C. L[R[i]]　　D. R[L[i]]

(4)④处应填(　　)。

A. a[L[i]]　　B. a[i]

C. R[L[i]]　　D. L[i]

(5)⑤处应填(　　)。

A. L[i]　　B. a[i]

C. R[i]　　D. R[L[i]]

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	B	D	A	B	C

【分析】本题主要考查双向链表的使用。

(1)标记每一个值的位置,a[x]=i表示x在原来序列中的位置i,用下标作为数组的值,这种用法在桶排里用过。

(2)跟下一行对称即可,这里就是在构建链表的初始状态,那么链表中第i个元素的下一个元素应该指向i+1。

(3)跟下一行对称即可,这里是在链表中删除第i个元素,那么将当前元素的下一个元素的左指针指向当前元素的上一个元素。

(4)同上,将当前元素的上一个元素的右指针指向当前元素的下一个元素。

(5)题目中需要输出的是每个元素的右边第一个比它大的元素,那么就是链表中这个元素的下一个元素,即R[i]。

2.【NOIP2016】交朋友

根据社会学研究表明,人们都喜欢找和自己身高相近的人做朋友。现在有n名身高两两不相同的同学依次走入教室,调查人员想预测每个人在走入教室的瞬间最想和已经进入教室的哪个人做朋友。当有两名同学和这名同学的身高差一样时,这名同学会更想和高的那个人做朋友。比如一名身高为1.80米的同学进入教室时,有一名身高为1.79米的同学和一名身高为1.81米的同学在教室里,那么这名身高为1.80米的同学会更想和身高为1.81米的同学做朋友。对于第一个走入教室的同学我们不作预测。

由于我们知道所有人的身高和走进教室的次序,所以我们可以采用离线的做法来解决这样的问题,用排序加链表的方式帮助每一个人找到在他之前进入教室的并且和他身高最相近的人。

```
#include<iostream>
using namespace std;
#define MAXN 200000
#define infinity 2147483647
int answer[MAXN],height[MAXN],previous[MAXN],next[MAXN];
int rank[MAXN];
int n;
void sort (int l,int r){
```

```
    int x=height[rank[(l+r)/2]],i=l,j=r,temp;
        while (i<=j){
        while (height[rank[i]]<x)i++;
        while (height[rank[j]]>x)j--;
        if (  ①  ){
            temp=rank[i];rank[i]=rank[j];rank[j]=temp;
            i++;j--;
        }
    }
    if (i<r)sort(i,r);
    if (l<j)sort(l,j);
}
int main(){
    cin>>n;
    int i,higher,shorter;
    for (i=1;i<=n;i++){
        cin>>height[i];
        rank[i]=i;
    }
    sort(1,n);
    for (i=1;i<=n;i++){
        previous[rank[i]]=rank[i-1];
        ②  ;
    }
    for (i=n;i>=2;i--){
        higher=shorter=infinity;
        if (previous[i]!=0)
            shorter=height[i]-height[previous[i]];
        if (next[i]!=0)
            ③  ;
        if (  ④  )
            answer[i]=previous[i];
        else
            answer[i]=next[i];
        next[previous[i]]=next[i];
        ⑤  ;
    }
    for (i=2;i<=n;i++)
        cout<<i<<":"<<answer[i];
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. i<j　　B. i>j

C. i<=j　　D. i>=j

(2)②处应填(　　)。

A. next[rank[i-1]]=rank[i]　　B. next[rank[i+1]]=rank[i]

C. next[i]=rank[i+1]　　D. next[rank[i]]=rank[i+1]

(3)③处应填(　　)。

A. higher=height[next[i]]-height[i]

B. higher=height[i]-height[previous[i]]

C. higher=max(higher,height[next[i]]-height[i])

D. higher=min(higher,height[next[i]]-height[i])

(4)④处应填(　　)。

A. shorter<=higher　　B. shorter<higher

C. answer[i]>previous[i];　　D. answer[i]<previous[i];

(5)⑤处应填(　　)。

A. next[previous[i]]=next[i]　　B. previous[next[i]]=previous[i]

C. previous[next[i]]=next[i]　　D. next[next[i]]=next[i]

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	C	D	A	B	B

【分析】本题主要考查链表的操作,用数组实现。按时间顺序从后往前计算与每位同学身高最相近的人,然后在链表中将该同学删去。

(1)i<=j,考查快排的知识点,sort()完成身高排序,可以知道身高由低到高。

(2)next[rank[i]]=rank[i+1],题干中提示了,链表方式存储比当前个子高和矮的人,模仿上面的 previous 的写法就行。

(3)higher=height[next[i]]-height[i],链表构建完后,对于每个人来说,跟他身高最相近的就是他前后两个人,那么根据上一个判断可以发现,这里是在求出当前这个人和他前后两个人的身高差,由于上面已经求了当前这个人和他前一个人的身高差,那么这里就要求和他后一个人的身高差,但要注意身高是从小到大的,所以后一个人的身高必然大于等于前一个人,那么这里填 higher=height[next[i]]-height[i]。

(4)shorter<higher,可以看出 shorter 就是与矮的人的身高差,higher 就是与高的人的身高差,然后根据题干要求,如果和矮的人的身高差更小则选矮的,否则选高的。

(5)previous[next[i]]=previous[i],还是模仿上面的 next 写。

第5节 分治算法

1.【NOIP2016】郊游活动

有n名同学参加学校组织的郊游活动，已知学校给这n名同学的郊游总经费为A元，与此同时第i位同学自己携带了M_i元。为了方便郊游，活动地点提供B(B≥n)辆自行车供人租用，租用第j辆自行车的价格为C_j元，每位同学可以使用自己携带的钱或者学校的郊游经费，为了方便账务管理，每位同学只能为自己租用自行车，且不会借钱给他人，他们想知道最多有多少位同学能够租用到自行车。

本题采用二分法。对于区间[l,r]，我们取中间点mid并判断租用到自行车的人数能否达到mid，判断过程利用贪心算法实现。

```
#include<iostream>
using namespace std;
#define MAXN 1000000
int n,B,A,M[MAXN],C[MAXN],l,r,ans,mid;
bool check (int nn){
    int count=0,i,j;
    i=  ①  ;
    j=1;
    while (i<=n){
        if (  ②  )
            count+=C[j]-M[i];
        i++;
        j++;
    }
    return  ③  ;
}
void sort (int a[],int l,int r){
    int i=l,j=r,x=a[(l+r)/2],y;
    while (i<=j){
        while (a[i]<x)i++;
        while (a[j]>x)j--;
        if (i<=j){
            y=a[i];a[i]=a[j];a[j]=y;
            i++;j--;
        }
    }
    if (i<r) sort(a,i,r);
    if (l<j) sort(a,l,j);
}
int main(){
    int i;
    cin>>n>>B>>A;
```

```
    for (i=1;i<=n;i++)
        cin>>M[i];
    for (i=1;i<=B;i++)
        cin>>C[i];
    sort(M,1,n);
    sort(C,1,B);
    l=0;
    r=n;
    while (l<=r){
        mid=(l+r)/2;
        if (  ④  ){
            ans=mid;
            l=mid+1;
        }else
            r=  ⑤  ;
    }
    cout<<ans<<endl;
    return 0;
}
```

选择题

(1)①处应填(　　)。

A. n-nn+1　　B. 1

C. n-nn　　D. nn

(2)②处应填(　　)。

A. M[j]>C[i]　　B. M[i]>C[j]

C. M[i]<C[j]　　D. M[j]<C[i]

(3)③处应填(　　)。

A. count<=A　　B. count>=A

C. count　　D. A-count

(4)④处应填(　　)。

A. check(mid-1)　　B. !check(mid)

C. check(mid)　　D. check(mid+1)

(5)⑤处应填(　　)。

A. break　　B. r=mid+1

C. r=mid　　D. r=mid-1

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	A	C	A	C	D

【分析】本题主要考查分治算法和贪心算法。check 函数使用贪心算法判断是否能够让 nn 个人租到自行车。贪心策略如下：让钱最多的 nn 个人租最便宜的 nn 辆车，其中，钱最少的租最便宜的车，钱第二少的租第二便宜的车……钱最多的租最贵的车。

(1)题目里提示本题是贪心,让最有钱的同学去租最便宜车就可以减少学校 A 的消耗,j=1 说明车是从便宜的开始,函数入口形式参数 nn 要让最有钱的 nn 个同学去租。

(2)此处下一行是统计钱不够的同学让学校出钱数量 count,可以判断 count 是学校的支出;i++说明当前 i 这位同学已经租下了第 j 辆车,即将尝试下一个同学,如果自己带的钱够租,那就不要让学校出钱,如果自己带的钱不够,那就让学校负担 C[j]-M[j]的钱;j++说明便宜的第 j 辆车被租走了,即将尝试下一辆车。

(3)函数返回值是 bool,这里是关系运算,返回的应该是 A 够不够,④处成立的处理语句中 mid+1 说明④是成立的情况,所有 count 必须不大于 A。

(4)测试当租车的人为 mid 的时候钱是否够。

(5)二分套路,租车的人太多,钱不够,要试试人更少的情况。

2.【NOIP2008】找第 k 大的数

给定一个长度为 1000000 的无序正整数序列,以及另一个数 n(1≤n≤1000000),接下来以类似快速排序的方法找到序列中第 n 大的数(关于第 n 大的数:例如序列{1,2,3,4,5,6}中第 3 大的数是 4)。

```
#include<iostream>
using namespace std;
int a[1000001],n,ans=-1;
void swap (int & a,int & b)
{
    int c;
    c=a;a=b;b=c;
}
int FindKth (int left,int right,int n)
{
    int tmp,value,i,j;
    if (left==right) return left;
    tmp=rand(    )%(right-left)+left;
    swap(a[tmp],a[left]);
    value=  ①
    i=left;
    j=right;
    while (i<j)
    {
        while (i<j&&  ②  )j--;
        if (i<j){a[i]=a[j];i++;} else break;
        while (i<j&&  ③  )i++;
        if (i<j){a[j]=a[i];j--;} else break;
    }
     ④
    if (i<n)return  FindKth(  ⑤  );
    if (i>n)return  ⑥
```

```
    return i;
}
int main()
{
    int i;
    int m=1000000;
    for (i=1;i<=m;i++)
        cin>>a[i];
    cin>>n;
    ans=FindKth(1,m,n);
    cout<<a[ans];
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. a[left];　　B. a[tmp];

C. a[right];　　D. a[n];

(2)②处应填(　　)。

A. a[j]<value　　B. a[j]>value

C. a[j]<a[i]　　D. a[j]>a[i]

(3)③处应填(　　)。

A. a[i]<value　　B. a[i]>value

C. a[i]<a[j]　　D. a[i]>a[j]

(4)④处应填(　　)。

A. a[i]=a[left];　　B. a[i]=a[right];

C. a[i]=a[tmp];　　D. a[i]=value;

(5)⑤处应填(　　)。

A. i,right,n　　B. i+1,m,n

C. i+1,right,n　　D. i,m,n

(6)⑥处应填(　　)。

A. FindKth(1,i-1,n);　　B. FindKth(left,i-1,n);

C. FindKth(1,i,n);　　D. FindKth(left,i,n);

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	A	A	B	D	C	B

【分析】

(1)tmp 为随机位置,根据上一句 a[tmp]与 a[left]交换,将随机位置的值换到了 left 位置上,下面都要参照该值进行比较,value 应赋值为 a[left]。

(2)根据 j--可知 j 为右侧游标,如果 a[j]<value,则 j 慢慢往左移动。

(3)根据 i++可知 i 为左侧游标,如果 a[i]>value,则 i 慢慢往右移动。

(4)循环 while (i<j)结束时 i、j 相等,需要把 value 放到 a[i]保存起来。

(5)如果 i 比 n 小,到右侧[i+1,right]继续寻找。

(6)如果 i 比 n 大,到左侧[left,i-1]继续寻找。

3.【NOIP2017】切割绳子

有 n 条绳子,每条绳子的长度已知且均为正整数。绳子可以以任意正整数长度切割,但不可以连接。现在要从这些绳子中切割出 m 条长度相同的绳段,求绳段的最大长度是多少。

输入:第一行是一个不超过 100 的正整数 n,第二行是 n 个不超过 10^6 的正整数,表示每条绳子的长度,第三行是一个不超过 10^8 的正整数 m。

输出:绳段的最大长度,若无法切割,输出 Failed。

```
#include<iostream>
using namespace std;
int n,m,i,lbound,ubound,mid,count;
int len[100];//绳子长度
int main(){
    cin>>n;
    count=0;
    for (i=0;i<n;i++){
        cin>>len[i];
        ____①____;
    }
    cin>>m;
    if (____②____){
        cout<<"Failed"<<endl;
        return 0;
    }
    lbound=1;
    ubound=1000000;
    while (____③____){
        mid=____④____;
        count=0;
        for (i=0;i<n;i++)
            ____⑤____;
        if (count<m)
            ubound=mid-1;
        else
            lbound=mid;
    }
    cout<<lbound<<endl;
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. 空　　B. count+=len[i]

C. count+　　D. count=count*10+len[i]

(2)②处应填(　　)。

A. count==m　　B. count<m

C. count>m　　D. count!=m

(3)③处应填(　　)。

A. count=m　　B. lbound=ubound

C. lbound<ubound　　D. lbound>ubound

(4)④处应填(　　)。

A. (lbound+ubound+1)/2　　B. (lbound+ubound-1)/2

C. (ubound-lbound+1)/2　　D. (lbound+ubound+1)

(5)⑤处应填(　　)。

A. count+=len[i]/mid　　B. count=(count+len[i])/mid

C. count+=len[i]/lbound　　D. count=(count+len[i])/ubound

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	B	B	C	A	A

【分析】本题主要考查二分算法和贪心算法。二分算法中间的 for 循环使用贪心算法计算能够切割出的绳段个数

结合第①②空可知，当绳段长度为 1 时可以切出的绳段个数最多，即所有绳子的长度相加，所以输出 Failed 的条件应该是所有绳子的总长度小于 m。故用 count 计算总长度，第①空填 count=count+len[i]或 count+=len[i]，第二空填 count<m 或 m>count。

第③④⑤空均在二分里，第③空是二分的结束条件，故应该填 lbound<ubound 或者 ubound>lbound，即当前二分的区间长度还大于 1 的时候，说明没有确定最终值是多少。第④空需要特别注意，由于底下 ubound 是通过 mid-1，lbound 是通过 mid 来更新的，所以这空需要填(lbound+ubound+1)/2 或(lbound+ubound+1)>>1 或(lbound+ubound)/2+1，三者意思相同。

第⑤空是用来求当确定绳子的切割长度为 mid 后，最多能切多少段绳段，故应该填 count=count+len[i]/mid 或者 count+=len[i]/mid。

(1)当绳段长度为 1 时可以切出的绳段个数最多，即所有绳子的长度相加，故用 count 计算总长度。

(2)输出 Failed 的条件应该是所有绳子的总长度小于 m。

(3)第③空是二分的结束条件，故应该填 lbound<ubound。

(4)由于底下 ubound 是通过 mid-1，lbound 是通过 mid 来更新的，所以这空需要填(lbound+ubound+1)/2。

(5)当确定绳子的切割长度为 mid 后，最多能切多少段绳段。

4.【NOIP2011】大整数开方

输入一个正整数 n($1\leqslant n<10^{100}$)，试用二分法计算它的平方根的整数部分。

```
#include<iostream>
#include<string>
using namespace std;
const int SIZE=200;
struct hugeint{
    int len,num[SIZE];
};
//其中 len 表示大整数的位数;num[1]表示个位、num[2]表示十位,以此类推
hugeint times(hugeint a,hugeint b){
//计算大整数 a 和 b 的乘积
    int i,j;hugeint ans;
    memset(ans.num,0,sizeof(ans.num));
    for (i=1;i<=a.len;i++)
    for (j=1;j<=b.len;j++)
        ①  +=a.num[i]*b.num[j];
    for (i=1;i<=a.len+b.len;i++){
        ans.num[i+1]+=ans.num[i]/10;
        ②  ;
}
if (ans.num[a.len+b.len]>0)
    ans.len=a.len+b.len;
else
    ans.len=a.len+b.len-1;
    return ans;
}
hugeint add(hugeint a,hugeint b){
    int i;hugeint ans;
    memset(ans.num,0,sizeof(ans.num));
    if (a.len>b.len)
        ans.len=a.len;
    else
        ans.len=b.len;
    for (i=1;i<=ans.len;i++){
        ③  ;
        ans.num[i+1]+=ans.num[i]/10;
        ans.num[i]%=10;
    }
    if (ans.num[ans.len+1]>0)ans.len++;
    return ans;
}
hugeint average(hugeint a,hugeint b){
    int i;hugeint ans;
```

```
    ans=add(a,b);
    for (i=ans.len;i>=2;i--){
        ans.num[i-1]+=(  ④  )*10;
        ans.num[i]/=2;
    }
    ans.num[1]/=2;
    if (ans.num[ans.len]==0)ans.len--;
    return ans;
}
hugeint plustwo(hugeint a){
    int i;hugeint ans;
    ans=a;
    ans.num[1]+=2;
    i=1;
    While((i<=ans.len)&&(ans.num[i]>=10)){
        ans.num[i+1]+=ans.num[i]/10;
        ans.num[i]%=10;
        i++;
    }
    if (ans.num[ans.len+1]>0)  ⑤  ;
    return ans;
}
bool over (hugeint a,hugeint b){
    int i;
    if (  ⑥  )return false;
    if (a.len>b.len)return true;
    for (i=a.len;i>=1;i--){
        if (a.num[i]<b.num[i])return false;
        if (a.num[i]>b.num[i])return true;
    }
    return false;
}
int main(){
    string s;
    int i;
    hugeint target,left,middle,right;
    cin>>s;
    memset(target.num,0,sizeof(target.num));
    target.len=s.length();
    for (i=1;i<=target.len;i++)
        target.num[i]=s[target.len-i]-  ⑦  ;
```

```
    memset(left.num,0,sizeof(left.num));
    left.len=1;
    left.num[1]=1;
    right=target;
    do{
        middle=average(left,right);
        if (over(  ⑧  ))right=middle;
        else left=middle;
    } while (!over(plustwo(left),right));
    for (i=left.len;i>=1;i--)
        cout<<left.num[i];
    cout<<endl;
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. ans.num[j-i+1]　　B. ans.num[i+j-1]

C. ans.num[i+j+1]　　D. ans.num[i-j+1]

(2)②处应填(　　)。

A. ans.num[i]=ans.num[i-1]%10　　B. ans.num[i]=ans.num[i]/10

C. ans.num[i]=ans.num[i]%10　　D. ans.num[i]=ans.num[i-1]*10

(3)③处应填(　　)。

A. ans.num[i]+=a.num[i]+b.num[i]

B. ans.num[i]=abs(a.num[i]-b.num[i])

C. ans.num[i]=min(a.num[i],b.num[i])

D. ans.num[i]=a.num[i]+b.num[i]

(4)④处应填(　　)。

A. ans.num[i]|1　　B. ans.num[i]&2

C. ans.num[i- 1]&1　　D. ans.num[i]&1

(5)⑤处应填(　　)。

A. ans.num[ans.len+1]%10　　B. ans.len++

C. ans.len--　　D. ans.num[ans.len++]=1

(6)⑥处应填(　　)。

A. times(a,a)<times(a,b)　　B. a.len<=b.len

C. a.len<b.len　　D. add(a,a)<add(b,b)

(7)⑦处应填(　　)。

A. 65　　B. 97

C. 34　　D. 48

(8)⑧处应填(　　)。

A. times(left,right),target　　B. times(middle+1,middle+1),target

C. times(middle,middle),target　　D. add(middle,middle),target

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
答案	B	C	A	D	B	C	D	C

【分析】快速浏览一下这题，二分法＋高精度，然后看各个函数过程的功能。通过对程序的观察，可以总结出各个函数的作用：add 函数实现的是高精度加法；average 函数计算大整数 a 和 b 的平均数的整数部分；plustwo 函数计算大整数 a 加 2 后的结果；over 函数所实现的功能是：若大整数 a＞b，则返回 true，否则返回 false。

(1)高精度乘法，不考虑可能的进位，直接计算并存储每一位。

(2)高精度乘法，处理进位后的余数，显然是对于 10 取模。

(3)函数 add 是高精度加法，第③空就是处理到第 i 位时的加法运算，注意之前有进位，它还要加上它自己。

(4)看主函数中的 middle=average(left,right)；可以知道这是求 a 和 b 的平均值，里面有个④，可以轻松看出④所在的循环是 ans 已经等于 a＋b 了，现在正在除以 2。模拟笔算用的竖式除法，可以知道④这个地方是退位，就用 ans.num[i]%2 就行了。注意%2 与 &1 等价。

(5)观察⑤所在的 if 语句，ans.len 代表位数由于进位增长 1 位，故应该填 ans.len++ 。

(6)看到 over 中的 ifa.len＞b.len 这段，得知如果 a 比 b 长，也就是 a 比 b 大，那 over 就为真。看来这个 over 就是个比大小的函数，a＞b 为真，a＜b 为假。⑥所在的地方与下面一个 if 语句类似，写上 a.len<b.len。

(7)将 char 类型的数字转化为 int 类型的数字，显然应该减去'0'，'0'的 ASCII 码为 48。

(8)此空的意思是判断当前结果与答案之间的关系，显然应该将 middle 的平方与答案进行比较。

5.【NOIP2015】中位数

给定 n(n 为奇数且小于 1000)个整数，整数的范围在 0～m(0＜m＜2^{31})之间，请使用二分法求这 n 个整数的中位数。所谓中位数，是指将这 n 个数排序之后，排在正中间的数。

```
#include<iostream>
using namespace std;
const int MAXN=1000;
int n,i,lbound,rbound,mid,m;
int x[MAXN];
int main(){
    cin>>n>>m;
    for (i=0;i<n;i++)
        cin>>x[i];
    lbound=0;rbound=m;
    while (  ①  ){
        mid=(lbound+rbound)/2;
        ②  ;
        for (i=0;i<n;i++)
```

```
            if (  ③  )
                ④ ;
            if (count>n/2)
                lbound=mid+1;
            else
                ⑤ ;
    }
    cout<<rbound<<endl;
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. lbound<rbound　　B. lbound+1<rbound

C. lboud<=rbound　　D. rbound-lbound>1

(2)②处应填(　　)。

A. int count=0　　B. int p=0

C. int count=1　　D. int p=1

(3)③处应填(　　)。

A. x[i]>x[p]　　B. x[i]>=x[p]

C. mid<x[i]　　D. x[mid]>i

(4)④处应填(　　)。

A. p=i　　B. p=max(p,i)

C. count*=1　　D. count++

(5)⑤处应填(　　)。

A. mid=rbound　　B. rbound=mid+1

C. rbound=mid　　D. rbound=mid-1

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	A	A	C	D	C

【分析】这是一道经典的左闭右闭的二分题，整个题目的过程为：先二分答案，再遍历整个数组验证有多少个数比 mid 小，将这个个数存入 count 中，据此判断左右边界的缩小。

(1)此空是二分的条件，所以应该为 lbound<rbound。具体的细节可以通过规模较小的例子手动模拟得出。

(2)第二空应该是将计数器 count 初始化操作，显然初始为 0。

(3)count 的作用是统计数组 x 中有多少个数比 mid 大。

(4)与③同理。

(5)注意二分的前提是左闭右闭，并且最后输出的是 rbound，对于二分掌握程度高的可以很容易看出答案。

第6节　搜索算法

1.【NOIP2009】国王放置

在n*m的棋盘上放置k个国王，要求k个国王互相不攻击，有多少种不同的放置方法。假设国王放置在第(x,y)格，国王攻击的区域是(x-1,y-1),(x-1,y),(x-1,y+1),(x,y-1),(x,y+1),(x+1,y-1),(x+1,y),(x+1,y+1)。读入三个数n,m,k，输出答案。题目利用回溯法求解，棋盘行标号为0～n-1，列标号为0～m-1。

```
#include<iostream>
using namespace std;
int n,m,k,ans;
int hash[5][5];
void work(int x,int y,int tot){
    int i,j;
    if (tot==k){
        ans++;
        return;
    }
    do{
        while (hash[x][y]){
            y++;
            if (y==m){
                x++;
                y=  ①  ;
            }
            if (x==n)
                return;
        }
        for (i=x-1;i<=x+1;i++)
            if (i>=0&&i<n)
                for (j=y-1;j<=y+1;j++)
                    if (j>=0&&j<m)
                        ②  ;
    ③  ;
        for (i=x-1;i<=x+1;i++)
            if (i>=0&&i<n)
                for (j=y-1;j<=y+1;j++)
                    if (j>=0&&j<m)
                        ④  ;
        y++;
        if (y==m){
            x++;
            y=0;
```

```
            }
            if (x==n)
                return;
        }
        while (1);
    }
    int main(){
        cin>>n>>m>>k;
        ans=0;
        memset(hash,0,sizeof(hash));
        ⑤ ;
        cout<<ans<<endl;
        return 0;
    }
```

●选择题

(1)①处应填(　　)。

A. 1　　B. 0

C. x　　D. m

(2)②处应填(　　)。

A. hash[i][j]++　　B. hash[i][j]--

C. hash[i][j]=0　　D. hash[i][j]=1

(3)③处应填(　　)。

A. work(x,y,tot++)　　B. work(x,y,++tot)

C. work(x,y,tot+1)　　D. work(x,y,tot)

(4)④处应填(　　)。

A. hash[i][j]++　　B. hash[i][j]--

C. hash[i][j]=0　　D. hash[i][j]=1

(5)⑤处应填(　　)。

A. work(0,0,0)　　B. work(0,0,1)

C. work(1,1,1)　　D. work(n,m,k)

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	B	A	C	B	A

【分析】本题考查搜索与回溯,考查学生对回溯的理解。

x,y表示当前所在的位置,tot指的是当前已经放置的国王数,hash数组存储的是某个格子被控制的次数,而不仅仅是判断是否被控制。所以①所在的while循环是在寻找下一个可以放置国王的位置,当一行走到头的时候需要换行。接下来的回溯伴随着国王控制的周边格子上数值的改变与还原,由此②④便很容易得出,在x,y放下一个国王后对周围的格子进行控制,所以这两个只要是相对的进行加减对应的标记即可,③这个位置则是回溯调用。

(1)所在的while循环是寻找下一个可以放置国王的位置,当一行走到头的时候需要换行。

(2)对国王控制的周边格子上数值的改变。

(3)从当前(x,y)位置进行下一个国王的放置。

(4)对国王控制的周边格子上数值的还原。

(5)调用 work 函数进行放置国王,初始位置为(0,0),初始未放置任何国王时参数 tot 为 0。

2.【NOIP2009】寻找等差数列

有一些长度相等的等差数列(数列中每个数都为 0～59 的整数),设长度均为 L,将等差数列中的所有数打乱顺序放在一起。现在给你这些打乱后的数,问原先 L 最大可能为多大?先读入一个数 n(1≤n≤60),再读入 n 个数,代表打乱后的数。输出等差数列最大可能长度 L。

```
#include<iostream>
#include<cstring>
using namespace std;
int hash[60];
int n,x,ans,maxnum;
int work(int now){
    int first,second,delta,i;
    int ok;
    while ( ① &&!hash[now])
        ++now;
    if (now>maxnum)
        return 1;
    first=now;
    for (second=first;second<=maxnum;second++)
        if (hash[second]){
            delta= ② ;
            if (first+delta* ③ >maxnum)
                break;
            if (delta==0)
                ok=( ④ );
            else {
                ok=1;
                for (i=0;i<ans;i++)
                    ok= ⑤ &&(hash[first+delta*i]);
            }
            if (ok){
                for (i=0;i<ans;i++)
                    hash[first+delta*i]--;
                if (work(first))
                    return 1;
            for (i=0;i<ans;i++)
```

```
                hash[first+delta*i]++;
            }
        }
        return 0;
}
int main(){
    int i;
    memset(hash,0,sizeof(hash));
    cin>>n;
    maxnum=0;
    for (i=0;i<n;i++){
        cin>>x;
        hash[x]++;
        if (x>maxnum)
            maxnum=x;
    }
    for (ans=n;ans>=1;ans--)
    if (n% ans==0&&  ⑥  ){
        cout<<ans<<endl;
        break;
    }
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. now<maxnum　　B. now>=maxnum

C. now<=maxnum　　D. now>maxnum

(2)②处应填(　　)。

A. second-first　　B. second

C. first-second　　D. second-ans

(3)③处应填(　　)。

A. ans-1　　B. ans-delta

C. ans　　D. ans-first

(4)④处应填(　　)。

A. hash[first]>ans　　B. hash[first]>=ans

C. hash[second]<=ans　　D. hash[second]<ans

(5)⑤处应填(　　)。

A. 0　　B. (!ok)

C. 1　　D. ok

(6)⑥处应填(　　)。

A. work(0)　　B. work(0)<=0

C. work(ans)　　D. (!work(ans))

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	C	A	B	B	D	A

【分析】

题目要求最大的 L，因此采取穷举的算法，从 n 开始，引入一个 work 函数判断是否可以使全部的数分为长度均为 ans 的等差数列，所以 ans 必须是 n 的约数，且使这些数可以满足条件。另外，hash[x]的值代表 x 出现的次数。

(1)work 一开始就跳过没有的数，条件是 hash[now]=0，还有一个就是 now<=maxnum，因为 maxnum 是最大的，但不一定是 59，所以填 C。

(2)找到了一个等差数列的起始项，不能确定第二项，因此穷举第二项，delta 就是公差，因此填 A。

(3)如果末项比 maxnum 大肯定不合法，所以填 B，根据末项的计算公式 first+delta*ans-delta>maxnum，即 first+delta*(ans-1)>maxnum。

(4)ok 是用来检验等差数列能否存在的，如果 delta 为 0，那意味着至少要有 ans 个 first 才满足条件，因此填 B。

(5)如果 delta 不为 0，那就逐一检查，hash[first+delta*i]==0，则 ok 就应变为 false，根据位运算的法则，所以应该填 D。

(6)由于数是 0～59，所以可能的最小的值为 0，因此填 A。

3.【NOIP2010】过河问题

在一个月黑风高的夜晚，有一群人在河的右岸，想通过唯一的一根独木桥走到河的左岸。在伸手不见五指的黑夜里，过桥时必须借助灯光来照明，不幸的是，他们只有一盏灯。另外，独木桥上最多能承受两个人同时经过，否则将会坍塌。每个人单独过独木桥都需要一定的时间，不同的人需要的时间可能不同。两个人一起过独木桥时，由于只有一盏灯，所以需要的时间是较慢的那个人单独过桥所花费的时间。现在输入 N(2≤N<1000)和这 N 个人单独过桥需要的时间，请计算总共最少需要多少时间，他们才能全部到达河左岸。

例如，有 3 个人甲、乙、丙，他们单独过桥的时间分别为 1、2、4，则总共最少需要的时间为 7。具体方法：甲、乙一起过桥到河的左岸，甲单独回到河的右岸将灯带回，然后甲，丙在一起过桥到河的左岸，总时间为 2+1+4=7。

```
#include<iostream>
#include<cstring>
using namespace std;
const int SIZE=100;
const int INFINITY=10000;
const bool LEFT=true;
const bool RIGHT=false;
const bool LEFT_TO_RIGHT=true;
const bool RIGHT_TO_LEFT=false;
int n,hour[SIZE];
bool pos[SIZE];
int max(int a,int b){
```

```
    if (a>b)
        return a;
    else
        return b;
}
int go (bool stage)
{
    int i,j,num,tmp,ans;
    if (stage==RIGHT_TO_LEFT)
    {
        num=0;
        ans=0;
        for (i=1;i<=n;i++)
          if (pos[i]==RIGHT)
          {
              num++;
              if (hour[i]>ans)
                  ans=hour[i];
          }
        if ( ① )
            return ans;
        ans=INFINITY;
        for (i=1;i<=n-1;i++)
          if (pos[i]==RIGHT)
              for (j=i+1;j<=n;j++)
                  if (pos[j]==RIGHT){
                      pos[i]=LEFT;
                      pos[j]=LEFT;
                      tmp=max(hour[i],hour[j])+ ② ;
                      if (tmp<ans)
                          ans=tmp;
                      pos[i]=RIGHT;
                      pos[j]=RIGHT;
                  }
        return ans;
    }
    if (stage==LEFT_TO_RIGHT)
    {
        ans=INFINITY;
        for (i=1;i<=n;i++)
            if ( ③ )
            {
```

```
                pos[i]=RIGHT;
                tmp=  ④  ;
                if (tmp<ans)
                    ans=tmp;
                 ⑤ ;
            }
        return ans;
    }
    return 0;
}
int main()
{
    int i;
    cin>>n;
    for (i=1;i<=n;i++){
        cin>>hour[i];
        pos[i]=RIGHT;
    }
    cout<<go(RIGHT_TO_LEFT)<<endl;
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. num<=0　　B. num<=1

C. num<=2　　D. num<=3

(2)②处应填(　　)。

A. go(LEFT)　　B. go(RIGHT)

C. go(LEFT_TO_RIGHT)　　D. go(RIGHT_TO_LEFT)

(3)③处应填(　　)。

A. pos[i]=LEFT　　B. pos[i]=RIGHT

C. pos[i]=LEFT_TO_RIGHT　　D. pos[i]=RIGHT_TO_LEFT

(4)④处应填(　　)。

A. hour[i]+go(RIGHT_TO_LEFT)　　B. hour[i]-go(RIGHT_TO_LEFT)

C. hour[i]+go(LEFT_TO_RIGHT)　　D. hour[i]-go(LEFT_TO_RIGHT)

(5)⑤处应填(　　)。

A. pos[i]=LEFT　　B. pos[i]=RIGHT

C. pos[i]=LEFT_TO_RIGHT　　D. pos[i]=RIGHT_TO_LEFT

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	C	C	A	A	A

【分析】

本小题中，LEFT 可用 true 代替，LEFT_TO_RIGHT 可用 true 代替，RIGHT_TO_LEFT 可用 false 代替。

本题就是模拟过河的过程。pos 记录的是这个人是在左岸还是右岸，stage 表示的是当前的状态，即现在是左岸到右岸还是右岸到左岸。如果是右岸到左岸，num 表示的是在右岸的人数，ans 表示这些人中最慢的一个，如果只有 1 个人或者 2 个在右岸，那么直接过去就行了，这时的时间 go 等于 ans，所以①填 num<=2，然后穷举过河的两个人，为何要穷举读者可以想一下，tmp 是这两个人之外的人过河的时间，因此②填 go(LEFT_TO_RIGHT)，然后进行回溯。如果是左岸到右岸，那么找 pos 中在左岸的人，然后穷举，所以③填 pos[i]==LEFT，④填 hour[i]+go(RIGHT_TO_LEFT)，然后再回溯，所以⑤填 pos[i]= LEFT。

(1)如果只有 1 个人或者 2 个在右岸，那么直接过去就行了。

(2)tmp 等于两个人从左岸到右岸的时间 max(hour[i],hour[j])加上其中一个人从左岸走回右岸的时间 go(LEFT_TO_RIGHT)。

(3)如果左岸到右岸只需要一个人，那么找 pos 中在左岸的人，然后穷举。

(4)同第(3)问，如果是左岸到右岸，那么找 pos 中在左岸的人，然后穷举。

(5)最后回溯。

第 7 节　动态规划

1.【NOIP2015】双子序列最大和

给定一个长度为 n(3≤n≤1000)的整数序列，要求从中选出两个连续子序列，使得这两个连续子序列的序列和之和最大，最终只需输出这个最大和。一个连续子序列的序列和为该连续子序列中所有数之和。要求：每个连续子序列长度至少为 1，且两个连续子序列之间至少间隔 1 个数。

```
#include<iostream>
using namespace std;
const int MAXN=1000;
int n,i,ans,sum;
int x[MAXN];
int lmax[MAXN];
//lmax[i]为仅含 x[i]及 x[i]左侧整数的连续子序列的序列和中最大的序列和
int rmax[MAXN];
//rmax[i]为仅含 x[i]及 x[i]右侧整数的连续子序列的序列和中最大的序列和
int main(){
    cin>>n;
    for (i=0;i<n;i++)cin>>x[i];
    lmax[0]=x[0];
    for (i=1;i<n;i++)
        if (lmax[i-1]<=0)
            lmax[i]=x[i];
        else
            lmax[i]=lmax[i-1]+x[i];
    for (i=1;i<n;i++)
        if (lmax[i]<lmax[i-1])lmax[i]=lmax[i-1];
    ____①____;
    for (i=n-2;i>=0;i--)
        if (rmax[i+1]<=0)
            ____②____;
        else
            ____③____;
    for (i=n-2;i>=0;i--)
        if (rmax[i]<rmax[i+1])
            ____④____;
    ans=x[0]+x[2];
    for (i=1;i<n-1;i++){
        sum=____⑤____;
        if (sum>ans)ans=sum;
    }
    cout<<ans<<endl;
```

```
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. lmax[n-1]=x[n-1]　　B. rmax[n]=x[n]

C. rmax[n-1]=x[n-1]　　D. lmax[n]=x[n]

(2)②处应填(　　)。

A. rmax[i]=x[i+1]　　B. rmax[i]=x[i]

C. x[i]=rmax[i]　　D. rmax[i]=x[i-1]

(3)③处应填(　　)。

A. rmax[i]=rmax[i-1]+x[i]　　B. rmax[i]=0x3f

C. rmax[i-1]=rmax[i]+x[i]　　D. rmax[i]=rmax[i+1]+x[i]

(4)④处应填(　　)。

A. rmax[i]=rmax[i+1]　　B. rmax[i]=rmax[i-1]

C. rmax[i]=x[i+1]　　D. rmax[i]=x[i-1]

(5)⑤处应填(　　)。

A. lmax[i]+rmax[i]　　B. lmax[i-1]+rmax[i+1]

C. lmax[i-2]+rmax[i+1]　　D. lmax[i-2]+rmax[i+2]

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	C	B	D	A	B

【分析】

(1)rmax[n-1]=x[n-1],仿照左侧初始化。

(2)rmax[i]=x[i],从后面累加得出来的和是负数,舍弃掉那段区间的值。

(3)rmax[i]=rmax[i+1]+x[i],从后面累加得出来的和不是负数,就把这个数与这段区间连接,累加加上这个数。

(4)rmax[i]=rmax[i+1],从右向左,这段程序和上面 2 个 for 循环对称。

(5)lmax[i-1]+rmax[i+1],穷举间隔位置。

2.【NOIP2018】小猪购物

一只小猪要买 N 件物品(N 不超过 1000)。

它要买的所有物品在两家商店里都有卖。第 i 件物品在第一家商店的价格是 a[i],在第二家商店的价格是 b[i],两个价格都不小于 0 且不超过 10000。如果在第一家商店买的物品的总额不少于 50000,那么在第一家店买的物品都可以 95 折(价格变为原来的 0.95 倍)。

求小猪买齐所有物品所需最少的总额。

输入:第一行一个数 N,接下来 N 行,每行两个数。第 i 行的两个数分别代表 a[i],b[i]。

输出:输出一行一个数,表示最少需要的总额,保留两位小数。

试补全程序。

```
#include<cstdio>
#include<cstdlib>
#include<iostream>
```

```
using namespace std;
const int Inf=1000000000;
const int threshold=50000;
const int maxn=1000;
int n,a[maxn],b[maxn];
bool put_a[maxn];
int total_a,total_b;
double ans;
int f[threshold];
int main(){
    scanf("%d",&n);
    total_a=total_b=0;
    for (int i=0;i<n;++i){
        scanf ("%d%d",a+i,b+i);
        if (a[i]<=b[i])total_a+=a[i];
        else total_b+=b[i];
    }
    ans=total_a+total_b;
    total_a=total_b=0;
    for (int i=0;i<n;++i){
        if (  ①  ){
            put_a[i]=true;
            total_a+=a[i];
        } else {
            put_a[i]=false;
            total_b+=b[i];
        }
    }
    if (  ②  ){
        printf ("%.2f",total_a*0.95+total_b);
        return 0;
    }
    f[0]=0;
    for (int i=1;i<threshold;++i)
        f[i]=Inf;
    int total_b_prefix=0;
    for (int i=0;i<n;++i)
        if (!put_a[i]){
            total_b_prefix+=b[i];
            for (int j=threshold-1;j>=0;--j){
                if (  ③  >=threshold&&f[j]!=Inf)
                    ans=min(ans,(total_a+j+a[i])*0.95+  ④  );
                f[j]=min(f[j]+b[i],j>=a[i]?  ⑤  :Inf);
```

```
            }
        }
    printf ("%.2f",ans);
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. a[i]<=b[i]　　B. a[i]<b[i]

C. a[i]*0.95<=b[i]　　D. a[i]*0.95<b[i]

(2)②处应填(　　)。

A. total_a>=threshold　　B. total_a>=total_b

C. total_b>=threshold　　D. total_a<total_b

(3)③处应填(　　)。

A. j+a[i]　　B. total_b-a[i]

C. total_a+j+a[i]　　D. total_b-total_b_prefix

(4)④处应填(　　)。

A. total_b-total_b_prefix　　B. f[j]+total_a-total_b

C. total_a-total_b_prefix　　D. f[j]+total_b-total_b_prefix

(5)⑤处应填(　　)。

A. f[j-a[i]]+b[i]　　B. f[j-b[i]]+a[i]

C. f[j-a[i]]　　D. f[j-b[i]]

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	C	A	C	D	C

【分析】本题功能介绍：

(1)由于上面已经考虑过不打折，那么这里就考虑打折。所以当第一家店打折后的价格便宜时选择a[i]，这里要注意，因为我们希望第一家店打折，那么就应该尽可能买第1家店的物品，所以当价格一样时我们也要选择a[i]而不是b[i]，所以这里应该是a[i]*0.95<=b[i]。

(2)因为if中有一个return 0，所以考虑一下什么情况下答案是total_a*0.95+total_b即可。

(3)观察两个循环可以发现，f[j]代表的就是前i件物品，在第1家店里购买j元的情况下在第2家店里购买物品的最小花费，可以看出f[j]与该选项无关。若买下当前第i件物品可以使得花费足够打折，那么就可以更新答案，所以这个空应该填total_a+j+a[i]。

(4)观察上一个选项，可以看出答案分成打折与没打折，此时只要将没打折的商品价格求和，即f[j]+total_b-total_b_prefix。

(5)这里在更新f[i]的状态，对于当前第i件物品，显然可以考虑两种情况，在第1家店里买和在第2家店里买，f[j]+b[i]就是在第2家店里买，那么另外一个情况就是在第1家店里买，并且这里还判断了j>=a[i]，显然这里跟背包一样，应该填写f[j-a[i]]。

3.【NOIP2010】烽火传递

烽火台又称烽燧，是重要的军事防御设施，一般建在险要处或交通要道上。一旦有敌情发生，白天燃烧柴草，通过浓烟表达信息；夜晚燃烧干柴，以火光传递军情。在某两座城市之

间有 n 个烽火台，每个烽火台发出信号都有一定的代价，为了使情报准确传递，在连续 m 个烽火台中至少要有一个发出信号。现输入 n、m 和每个烽火台发出信号的代价，请计算总共最少花费多少代价，才能使敌军来袭之时，情报能在这两座城市之间准确传递？

例如，有 5 座烽火台，它们发出信号的代价依次为 1，2，5，6，2，且 m 为 3，则总共最少花费的代价为 4，即由第 2 个和第 5 个烽火台发出信号。

```
#include<iostream>
using namespace std;
const int SIZE=100;
int n,m,r,value[SIZE],heap[SIZE],
    pos[SIZE],home[SIZE],opt[SIZE];
    //heap[i]表示用顺序数组存储的堆 heap 中第 i 个元素的值
    //pos[i]表示 opt[i]在堆 heap 中的位置，即 heap[pos[i]]=opt[i]
    //home[i]表示 heap[i]在序列 opt 中的位置，即 opt[home[i]]=heap[i]
void swap(int i,int j)
{//交换堆中的第 i 个和第 j 个元素
    int tmp;
    pos[home[i]]=j;
    pos[home[j]]=i;
    tmp=heap[i];
    heap[i]=heap[j];
    heap[j]=tmp;
    tmp=home[i];
    home[i]=home[j];
    home[j]=tmp;
}
void add(int k)
{//在堆中插入 opt[k]
    int i;
    r++;
    heap[r]=  ①  ;
    pos[k]=r;
     ②  ;
    i=r;
    while ((i>1)&&(heap[i]<heap[i/2])){
        swap(i,i/2);
        i/=2;
    }
}
void remove(int k){//在堆中删除 opt[k]
    int i,j;
    i=pos[k];
    swap (i,r);
```

```
        r--;
        if (i==r+1)
            return;
        while ((i>1)&&(heap[i]<heap[i/2])){
            swap (i,i/2);
            i/=2;
        }
        while (i+i<=r){
            if ((i+i+1<=r)&&(heap[i+i+1]<heap[i+i]))
                j=i+i+1;
            else
                ___③___;
            if (heap[i]>heap[j]){
                ___④___;
                i=j;
            }
            else
                break;
        }
    }
    int main(){
        int i;
        cin>>n>>m;
        for (i=1;i<=n;i++)
            cin>>value[i];
        r=0;
        for (i=1;i<=m;i++){
            opt[i]=value[i];
            add(i);
        }
        for (i=m+1;i<=n;i++){
            opt[i]=___⑤___;
            remove(___⑥___);
            add(i);
        }
        cout<<heap[1]<<endl;
        return 0;
    }
```

●选择题

(1)①处应填(　　)。

A. k　　B. opt[k]

C. opt[r]　　D. pos[k]

(2)②处应填(　　)。

A. home[k]=r　　B. home[r]=opt[k]

C. home[r]=k　　D. home[pos[r]]=opt[k]

(3)③处应填(　　)。

A. r/=2　　B. j=i+i

C. j+=j;　　D. j=i+i-1;

(4)④处应填(　　)。

A. swap(heap[i],heap[i])　　B. heap[i]=heap[j]

C. swap(i,j)　　D. swap(j,r)

(5)⑤处应填(　　)。

A. heap[1]　　B. value[i]+heap[r]

C. heap[r]　　D. value[i]+heap[1]

(6)⑥处应填(　　)。

A. i-m　　B. i-m+1

C. pos[i-m]　　D. pos[i-m+1]

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	B	C	B	C	D	A

【分析】本题算法是动态规划+小根堆排序。设F[i]表示将信息从1传到i所花费的最小代价,点燃i之前,必须先点燃[i-M+1,Fi-1]区间中的某一个烽火台,因此状态转移方程为F[i]=min(F[k]+value[i])k,[i-M+1,Fi-1],由于小根堆对F[k]中的值进行排序,并将最小的值赋值给heap[1],因此状态转移方程即化简为F[i]=heap[1]+value[i]。

(1)将花费值opt[k]存入。

(2)根据提示保存home值。

(3)左儿子比右儿子小,选择左儿子。

(4)父亲比儿子大,就交换二者,维护小根堆。

(5)根据题意value[i]+heap[1]。

(6)删去无法贡献的点i-m。

第8节 栈和队列

1.【NOIP2014】双栈模拟数组

只使用两个栈结构 stack1 和 stack2，模拟对数组的随机读取。作为栈结构，stack1 和 stack2 只能访问栈顶（最后一个有效元素）。栈顶指针 top1 和 top2 均指向栈顶元素的下一个位置。

输入：第一行包含两个整数，分别是数组长度 n 和访问次数 m，中间用单个空格隔开。第二行包含 n 个整数，依次给出数组各项（数组下标从 0 到 n－1）。第三行包含 m 个整数，需要访问的数组下标。对于每次访问，输出对应的数组元素。

```
#include<iostream>
using namespace std;
const int SIZE=100;
int stack1[SIZE],stack2[SIZE];
int top1,top2;
int n,m,i,j;
void clearStack(){
    int i;
    for (i=top1;i<SIZE;i++)
        stack1[i]=0;
    for (i=top2;i<SIZE;i++)
        stack2[i]=0;
}
int main(){
    cin>>n>>m;
    for (i=0;i<n;i++)
        cin>>stack1[i];
    top1=  ①  ;
    top2=  ②  ;
    for (j=0;j<m;j++){
        cin>>i;
        while (i<top1-1){
            top1--;
             ③  ;
            top2++;
        }
        while (i>top1-1){
            top2--;
             ④  ;
            top1++;
        }
        clearStack();
        cout<<stack1[  ⑤  ]<<endl;
```

```
    }
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. 0　　B. n

C. 1　　D. -1

(2)②处应填(　　)。

A. 1　　B. 2

C. n　　D. 0

(3)③处应填(　　)。

A. stack2[top2]=stack1[top1]　　B. top1=top2

C. stack2[top1]=stack1[top1]　　D. stack2[top1]=stack1[top2]

(4)④处应填(　　)。

A. stack1[top1]=stack2[top1]　　B. stack1[top2]=stack2[top1]

C. stack1[top1]=stack2[top2]　　D. stack1[top2]=stack2[top2]

(5)⑤处应填(　　)。

A. top1　　B. top2

C. top1-1　　D. top2-1

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	B	D	A	C	C

【分析】访问第 i 个元素时，stack1 从栈底到栈顶依次存放第 1 到第 i 个元素，而 stack2 从栈底到栈顶一次存放第 n 到第(i+1)个元素。

(1)stack1 栈顶元素下标为 n−1，top1 指向栈顶的下一个位置 n。

(2)stack2 此时为空，故 top2 指向栈顶元素的下一个位置即 0。

(3)当下一次要访问的元素在 i 之前时，通过将 stack1 的栈顶元素取出并放到 stack2 中实现“指针的向前移动”。

(4)当下一次要访问的元素在 i 之后时，通过将 stack2 的栈顶元素取出并放到 stack1 中可以实现“指针的向后移动”。

(5)输出 stack1 栈顶为要求的答案，top1−1 是因为 top1 指向栈顶的下一个位置。

2.【NOIP2012】新壳栈

小 Z 设计了一种新的数据结构“新壳栈”。首先，它和传统的栈一样支持压入、弹出操作。此外，其栈顶的前 c 个元素是它的壳，支持翻转操作。其中，c>2 是一个固定的正整数，表示壳的厚度。小 Z 还希望，每次操作，无论是压入、弹出还是翻转，都仅用与 c 无关的常数时间完成。聪明的你能帮助她编程实现“新壳栈”吗？

程序期望的实现效果如以下两表所示。其中，输入的第一行是正整数 c，之后每行输入都是一条指令。另外，如遇弹出操作时栈为空，或翻转操作时栈中元素不足 c 个，应当输出相应的错误信息。

表 1　指令的涵义

指令	涵义
1[空格]e	在栈顶压入元素 e
2	弹出(并输出)栈顶元素
3	翻转栈顶的前 c 个元素
0	退出

表 2　输入输出样例

输入	输出	栈中的元素 (左为栈底,右为栈顶)	说明
3			输入正整数 c
1 1	1	压入元素 1	
1 2	1 2	压入元素 2	
1 3	1 2 3	压入元素 3	
1 4	1 2 3 4	压入元素 4	
3	1 4 3 2	翻转栈顶的前 c 个元素	
1 5	1 4 3 2 5	压入元素 5	
3	1 4 5 2 3	翻转栈顶的前 c 个元素	
2	3	1 4 5 2	弹出栈顶元素 3
2	2	1 4 5	弹出栈顶元素 3
2	5	1 4	弹出栈顶元素 5
3	错误信息	1 4	由于栈中元素不足 c 个,无法翻转,故操作失败,输出错误信息
2	4	1	弹出栈顶元素 4
2	1	空	弹出栈顶元素 1
2	错误信息	空	由于栈为空,无法弹出栈顶元素,故操作失败,输出错误信息
0		空	退出

```
#include<iostream>
using namespace std;
const int NSIZE=100000,CSIZE=1000;
int n,c,r,tail,head,s[NSIZE],q[CSIZE];
//数组 s 模拟一个栈,n 为栈的元素个数
//数组 q 模拟一个循环队列,tail 为队尾的下标,head 为队头的下标
bool direction,empty;
int previous(int k){
    if (direction)
        return ((k+c-2)%c)+1;
    else
```

```
    return (k%c)+1;
}
int next (intk){
    if (direction)
    ____①____;
else
    return((k+c-2)%c)+1;
}
void push(){
    int element;
    cin>>element;
    if (next(head)==tail){
        n++;
        ____②____;
        tail=next(tail);
    }
    if (empty)
        empty=false;
    else
        head=next(head);
    ____③____=element;
}
void pop(){
    if (empty){
        cout<<"Error:the stack is empty!"<<endl;
        return;
    }
    cout<<____④____<<endl;
    if (tail==head)
        empty=true;
    else {
        head=previous(head);
        if (n>0){
            tail=previous(tail);
            ____⑤____=s[n];
            n--;
        }
    }
}
void reverse(){
    int temp;
```

```
    if ( ⑥ ==tail){
        direction=!direction;
        temp=head;
        head=tail;
        tail=temp;
    } else
        cout<<"Error:lessthan"<<c<<"elements in the stack!"<<endl;
}
int main(){
    cin>>c;
    n=0;
    tail=1;
    head=1;
    empty=true;
    direction=true;
    do{
        cin>>r;
        switch(r){
            case1:push();break;
            case2:pop();break;
            case3:reverse();break;
        }
} while (r!=0);
return 0;
}
```

●选择题

(1)①处应填(　　)。

A. k%c+1　　B. k%c

C. k　　D. (k+1)%c

(2)②处应填(　　)。

A. s[n]=q[head]　　B. s[n]=tail

C. s[n]=head　　D. s[n]=q[tail]

(3)③处应填(　　)。

A. q[tail]　　B. tail

C. head　　D. q[head]

(4)④处应填(　　)。

A. q[tail+1]　　B. tail

C. q[head]　　D. q[head+1]

(5)⑤处应填(　　)。

A. tail　　B. q[tail+1]

C. q[tail]　　D. q[head]

(6)⑥处应填(　　)。

A. previous(head)　　B. next(head)

C. previous(tail)　　D. next(tail)

【答案】

题号	(1)	(2)	(3)	(4)	(5)	(6)
答案	A	D	D	C	C	B

【分析】设计了一个新的数据结构，利用一个栈和一个队列，其中队列里的元素可以翻转。实现的时候本题运用到了循环队列的方法，如果有多于c的数，那么就放入到栈中；如果队列中有空位，那么就把栈顶存入到队列中。

(1)当队列翻转过来时，队列的队尾会反过来，所以应按照从队尾求满员时队头的方式计算从队头求满员时队尾。

(2)如果有多出来的数，那么将多余的数放入栈中。

(3)当从栈顶压入一个数时从队列的队头中插入这个数。

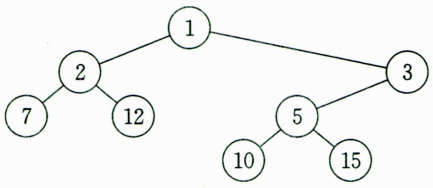
(4)当求栈顶的时候就输出队列的队头。

(5)如果队列有空，那么就把栈顶的数加入到队列中。

(6)这里要通过队头求满员时队尾来确认队列是否能够翻转。

第9节 树和图

1.【NOIP2011】笛卡尔树



对于一个给定的两两不等的正整数序列，笛卡尔树是这样的一棵二叉树：首先，它是一个最小堆，即除了根结点外，每个结点的权值都大于父结点的权值；其次，它的中序遍历恰好就是给定的序列。例如，对于序列7、2、12、1、10、5、15、3，右图就是一棵对应的笛卡尔树。现输入序列的规模n(1≤n≤100)和序列的n个元素，试求其对应的笛卡尔树的深度d(根结点深度为1)，以及有多少个叶结点的深度为d。

```
#include<iostream>
using namespace std;
const int SIZE=100+5;
const int INFINITY=1000000;
int n,a[SIZE],maxDeep,num;
void solve (int left,int right,int deep){
    int i,j,min;
    if (deep>maxDeep){
        maxDeep=deep;
        num=1;
    }else if (deep==maxDeep)
        ①  ;
    min=INFINITY;
    for (i=left;i<=right;i++)
        if (min>a[i]){
            min=a[i];
            ②  ;
        }
    if (left<j)  ③  ;
    if (j<right)  ④  ;
}
int main(){
    int i;
    cin>>n;
    for (i=1;i<=n;i++)
        cin>>a[i];
    maxDeep=0;
    solve (1,n,1);
    cout<<maxDeep<<''<<num<<endl;
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. num++　　B. return

C. ++left　　D. --right

(2)②处应填(　　)。

A. i=j　　B. j++

C. j=i　　D. j--

(3)③处应填(　　)。

A. solve(j,right,deep+1)　　B. solve(j+1,right,deep+1)

C. solve(left,j,deep+1)　　D. solve(left,j-1,deep+1)

(4)④处应填(　　)。

A. solve(left,j,deep+1)　　B. solve(j+1,right,deep+1)

C. solve(j,right,deep+1)　　D. solve(left,j-1,deep+1)

【答案】

题号	(1)	(2)	(3)	(4)
答案	A	C	D	B

【分析】这题很简单,只要知道什么是中序遍历就能解决。中序遍历就是按照左根右的顺序遍历一颗二叉树。看 solve 这个过程,就是在 left 到 right 这个树中找到这个数的根,然后根左根右分别继续 solve。deep 就是深度,maxDeep 就是最大深度,看输出,还有个 num,这就是最大深度的叶结点的个数了。所以很容易得出第一空①。

然后②那个地方就是找最小值,也就是这个子树的根,看程序,下面用到了 j,而上面没有一处用 j,所以这里应该有个 j,用来记录根的位置。然后下面分治,根据中序遍历的性质,根的左边是左子树,右边是右子树,分别再 solve 进去,深度加一。

(1)num 记录最大深度的叶结点的个数,因此要加 1。

(2)j 用来记录根的位置,要标为 min 在 a 数组的下标。

(3)根据中序遍历的性质,在根的左边即为左子树,递归处理左子树。

(4)根据中序遍历的性质,在根的右边即为右子树,递归处理右子树。

2.【NOIP2013】二叉查找树

二叉查找树具有如下性质:每个结点的值都大于其左子树上所有结点的值、小于其右子树上所有结点的值。试判断一棵树是否为二叉查找树。

输入的第一行包含一个整数 n,表示这棵树有 n 个顶点,编号分别为 1,2,…,n,其中编号为 1 的为根结点。之后的第 i 行有三个数 value,left_child,right_child,分别表示该结点关键字的值、左子结点的编号、右子结点的编号;如果不存在左子结点或右子结点,则用 0 代替。输出 1 表示这棵树是二叉查找树,输出 0 则表示不是。

```
#include<iostream>
using namespace std;
const int SIZE=100;
const int INFINITE=1000000;
struct node{
    int left_child,right_child,value;
```

```
};
nodea[SIZE];
int is_bst(int root,int lower_bound,int upper_bound){
    int cur;
    if (root==0)return 1;
    cur=a[root].value;
    if ((cur>lower_bound)&&(  ①  )&&
        (is_bst(a[root].left_child,lower_bound,cur)==1)&&
        (is_bst(  ②  ,  ③  ,  ④  )==1))
        return 1;
    return 0;
}
int main(){
    int i,n;
    cin>>n;
    for (i=1;i<=n;i++)
        cin>>a[i].value>>a[i].left_child>>a[i].right_child;
    cout<<is_bst(  ⑤  ,-INFINITE,INFINITE)<<endl;
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. cur<upper_bound　　B. cur<=upper_bound

C. cur>upper_bound　　D. cur>=upper_bound

(2)②处应填(　　)。

A. left_child　　B. right_child

C. a[root].left_child　　D. a[root].right_child

(3)③处应填(　　)。

A. left_child　　B. right_child

C. cur+1　　D. cur

(4)④处应填(　　)。

A. left_child　　B. upper_bound-1

C. right_child　　D. upper_bound

(5)⑤处应填(　　)。

A. -1　　B. 0

C. 1　　D. 2

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	A	D	D	D	C

【分析】本题考查二叉查找树(Binary Search Tree),又称二叉搜索树,是指一棵空树或者具有以下性质的二叉树:

1.若任意结点的左子树不空,则左子树上所有结点的值均小于它的根结点的值;

2.若任意结点的右子树不空，则右子树上所有结点的值均大于它的根结点的值；

3.任意结点的左、右子树也分别为二叉查找树；

4.没有键值相等的结点。

(1)根据前一个条件 cur>lower_bound 可知这里应该是 cur<upper_bound，即 cur 必须要在给定的范围(lower_bound，upper_bound)内。

(2)is_bst()函数的三个参数分别代表当前根结点的编号、区间左右范围的值。在这里是通过 cur 将区间[lower_bound，upper_bound]一分为二，分别遍历两个子树，右子树根结点为 a[root].right_child。

(3)右子树的区间范围为[cur，upper_bound]，区间左端点是 cur。

(4)右子树的区间范围为[cur，upper_bound]，区间右端点是 upper_bound。

(5)这是 is_bst()函数的初始值，1 代表根结点。

3.【NOIP2015】最短路径问题

无向连通图 G 有 n 个结点，依次编号为 1，2，3，…，n。用邻接矩阵的形式给出每条边的边长，要求输出以结点 1 为起点出发，到各结点的最短路径长度。

使用 Dijkstra 算法解决该问题：利用 dist 数组记录当前各结点与起点的已找到的最短路径长度；每次从未扩展的结点中选取 dist 值最小的结点 v 进行扩展，更新与 v 相邻结点的 dist 值；不断进行上述操作直至所有结点均被扩展，此时 dist 数据中记录的值即为各结点与起点的最短路径长度。

```
#include<iostream>
using namespace std;
const int MAXV=100;
int n,i,j,v;
int w[MAXV][MAXV];
//邻接矩阵,记录边长
//其中 w[i][j]为连接结点 i 和结点 j 的无向边长度,若无边,则为-1
int dist[MAXV];
int used[MAXV];
//记录结点是否已扩展(0:未扩展;1:已扩展)
int main(){
    cin>>n;
    for (i=0;i<n;i++)
       for (j=0;j<n;j++)
           cin>>w[i][j];
    dist[0]=0;
    for (i=1;i<n;i++)dist[i]=-1;
    for (i=0;i<n;i++)used[i]=0;
    while (true){
        ①  ;
        for (i=0;i<n;i++)
            if (used[i]!=1&&dist[i]!=-1&&(v==-1||  ②  ))
                ③  ;
        if (v==-1)break;
```

```
        ____④____;
        for (i=0;i<n;i++)
            if (w[v][i]!=-1&&(dist[i]==-1||____⑤____))
                dist[i]=dist[v]+w[v][i];
    }
    for (i=0;i<n;i++)cout<<dist[i]<<endl;
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. v=0　　B. v=-1

C. dis[v]=1　　D. dis[v]=0

(2)②处应填(　　)。

A. dist[v]!=-1　　B. v==n-1

C. used[i]!=1　　D. dist[i]<=dist[v]

(3)③处应填(　　)。

A. used[i]=1　　B. used[v]=1

C. v++　　D. v=i

(4)④处应填(　　)。

A. w[v][v]=0　　B. w[v][v]=1

C. used[v]=1　　D. dist[v]++

(5)⑤处应填(　　)。

A. !used[v]　　B. !used[i]

C. dist[v]-dist[i]+(used[i]==1)<=w[v][i]

D. dist[v]+w[v][i]<=dist[i]

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	B	D	D	C	D

【分析】

(1)填 v=-1 是因为在下面的 if 语句中有一个条件是 v==-1,而在整个程序里还没有给 v 赋过初值,所以这一空应该是赋初值。

(2)根据 v 的含义,应该是要找一个没有用过的 dist 最小的值,因此填 dist[i]<dist[v]。

(3)是更新 v,所以我们需要把当前 v 的值更新为 i。

(4)纵观整个程序可以发现 used 数组并没有被修改过,所以这个空就应该是 used[v]=1。

(5)根据 Dijkstra 算法可以推出更新条件是 dist[v]+w[v][i]<dist[i]。

4.【NOIP2017】最长路径

给定一个有向无环图,每条边长度为 1,求图中的最长路径长度。

输入:第一行是结点数 n(不超过 100)和边数 m,接下来 m 行,每行两个整数 a、b,表示从结点 a 到结点 b 有一条有向边。结点标号从 0 到(n-1)。

输出:最长路径长度。

提示:先进行拓扑排序,然后按照拓扑排序计算最长路径。

```
#include<iostream>
using namespace std;
int n,m,i,j,a,b,head,tail,ans;
int graph[100][100];  //用邻接矩阵存储图
int degree[100];      //记录每个结点的入度
int len[100];         //记录以各结点为终点的最长路径长度
int queue[100];       //存放拓扑排序结果
int main(){
    cin>>n>>m;
    for (i=0;i<n;i++)
        for (j=0;j<n;j++)
            graph[i][j]=0;
    for (i=0;i<n;i++)
        degree[i]=0;
    for (i=0;i<m;i++){
        cin>>a>>b;
        graph[a][b]=1;
        ____①____;
    }
    tail=0;
    for (i=0;i<n;i++)
    if (____②____){
        queue[tail]=i;
        tail++;
    }
    head=0;
    while (tail<n-1){
        for (i=0;i<n;i++)
            if (graph[queue[head]][i]==1){
                ____③____;
                if (degree[i]==0){
                    queue[tail]=i;
                    tail++;
                }
            }
        ____④____;
        }
    ans=0;
    for (i=0;i<n;i++){
        a=queue[i];
        len[a]=1;
        for (j=0;j<n;j++)
```

```
            if (graph[j][a]==1&&len[j]+1>len[a])
                len[a]=len[j]+1;
        if (  ⑤  )
        ans=len[a];
    }
    cout<<ans<<endl;
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. graph[b][a]=1;　　B. degree[b]=degree[b]+1

C. degree[b]=degree[a]+1　　D. graph[b][a]=graph[a][b]+1

(2)②处应填(　　)。

A. graph[1][i]!=0　　B. graph[1][i]==0

C. degree[i]==0　　D. degree[i]!=0

(3)③处应填(　　)。

A. degree[i]--　　B. head++

C. tail--　　D. tail++

(4)④处应填(　　)。

A. graph[queue[head]][i]++　　B. tail-=head

C. graph[queue[tail]][i]==1　　D. ++head

(5)⑤处应填(　　)。

A. ans+1<len[a]　　B. ans<len[a]

C. len[a]-1>=ans　　D. !ans

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	B	C	A	D	B

【分析】本题首先用广度优先算法(BFS)进行拓扑排序,需要用队列实现,然后按照拓扑顺序递推计算到每个点的最长路径,并将所有点的最长路径取最大值后输出。

(1)拓扑排序开始前的准备,显然需要计算的就是每个结点的入度,所以填写 B。

(2)拓扑排序初始化,就是将入度为 0 的点入队,所以这里应该填 C。

(3)每次将当前队首结点从图中删除,并遍历所有与它相邻的点,如果存在边,就删除这条边,使得这条边指向的点的入度－1,所以填 A。

(4)将当前结点 queue[head]从图中删除的操作已经做完了,那这里需要做的就是将当前结点从队列中弹出,这里是用数组模拟队列,所以只要 head++即可。

(5)最长路径的更新,判断满足的话会更新 ans,显然就是判断 len[a]>ans。

5.【NOIP2016】交通中断

有一个小国家,国家内有 n 座城市和 m 条双向的道路,每条道路连接着两座不同的城市。其中 1 号城市为国家的首都。由于地震频繁可能导致某一个城市与外界交通全部中

断。这个国家的首脑想知道，如果只有第 i(i>1)个城市因地震而导致交通中断时，首都到多少个城市的最短路径长度会发生改变。如果因为无法通过第 i 个城市而导致从首都出发无法到达某个城市，也认为到达该城市的最短路径长度改变。

对于每一个城市 i，假定只有第 i 个城市与外界交通中断，输出有多少个城市会因此导致到首都的最短路径长度改变。

我们采用邻接表的方式存储图的信息，其中 head[x]表示顶点 x 的第一条边的编号，next[i]表示第 i 条边的下一条边的编号，point[i]表示第 i 条边的终点，weight[i]表示第 i 条边的长度。

```
#include<iostream>
#include<cstring>
using namespace std;
#define MAXN 6000
#define MAXM 100000
#define infinity 2147483647
int head[MAXN],next[MAXM],point[MAXM],weight[MAXM];
int queue[MAXN],dist[MAXN],visit[MAXN];
int n,m,x,y,z,total=0,answer;
void link(int x,int y,int z){
    total++;
    next[total]=head[x];
    head[x]=total;
    point[total]=y;
    weight[total]=z;
    total++;
    next[total]=head[y];
    head[y]=total;
    point[total]=x;
    weight[total]=z;
}
int main(){
    int i,j,s,t;
    cin>>n>>m;
    for (i=1;i<=m;i++){
        cin>>x>>y>>z;
        link(x,y,z);
    }
    for (i=1;i<=n;i++)dist[i]=infinity;
    ①;
    queue[1]=1;
    visit[1]=1;
    s=1;
    t=1;
```

```
//使用 SPFA 求出第一个点到其余各点的最短路长度
while (s<=t){
    x=queue[s% MAXN];
    j=head[x];
    while (j!=0){
        if (  ②  ){
            dist[point[j]]=dist[x]+weight[j];
            if (visit[point[j]]==0){
                t++;
                queue[t% MAXN]=point[j];
                visit[point[j]]=1;
            }
        }
        j=next[j];
    }
     ③ ;
    s++;
}
for (i=2;i<=n;i++){
    queue[1]=1;
    memset(visit,0,sizeof(visit));
    visit[1]=1;
    s=1;
    t=1;
    while (s<=t){//判断最短路长度是否不变
        x=queue[s];
        j=head[x];
        while (j!=0){
            if (point[j]!=i&&  ④
                &&visit[point[j]]==0){
                 ⑤ ;
                t++;
                queue[t]=point[j];
            }
            j=next[j];
        }
        s++;
    }
    answer=0;
    for (j=1;j<=n;j++)
        answer+=1-visit[j];
    cout<<i<<":"<<answer-1<<endl;
```

```
    }
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. dist[1]=1　　B. dist[n]=1

C. dist[1]=-1　　D. dist[1]=0

(2)②处应填(　　)。

A. dist[point[x]]+weight[j]<dist[x]

B. dist[j]+weight[x]<dist[point[x]]

C. dist[x]+weight[j]<dist[point[j]]

D. dist[x]+weight[point[j]]<dist[j]

(3)③处应填(　　)。

A. t--　　B. visit[x]=0

C. point[j]=j　　D. nxt[j]=point[j]

(4)④处应填(　　)。

A. dist[x]+weight[j]==dist[point[j]]

B. dist[x]+weight[point[j]]==dist[point[j]]

C. dist[x]+weight[point[j]]==dist[point[x]]

D. dist[point[x]]+weight[point[j]]==dist[point[j]]

(5)⑤处应填(　　)。

A. visit[point[j]]=0　　B. visit[nxt[j]]=0

C. visit[queue[t]]=1　　D. visit[point[j]]=1

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	D	C	B	A	D

【分析】本题功能介绍：

(1)要先做一次最短路，将1入队，也已经将1标记为已访问，那么这里需要将dist[1]初始化，即dist[1]=0。

(2)dist[x]+weight[j]<dist[point[j]]，松弛操作，下面一行就是提示，送分题。

(3)visit[x]=0，这个算法的难点，x结点允许再次进入队列，这是最短路径快速算法(SPFA)和BFS算法的区别。

(4)dist[x]+weight[j]==dist[point[j]]，题目上已经有提示了，判断最短路径长度是否不变，如果不变就要改变visit[]状态。

(5)visit[point[j]]=1，不变的标记为1，和第(4)空一起，结合最后一段代码answer+=1-visit[j]统计中断交通后导致到首都的dist[]发生改变的数量。如果visit的值为1，则1-visit[j]为0，即最短路径长度不变，不需要统计进答案中。

第 10 节 数论

1.【NOIP2010】哥德巴赫猜想

哥德巴赫猜想是指任一大于 2 的偶数都可以写成两个质数之和。迄今为止，这仍然是一个著名的世界难题，被誉为数学王冠上的明珠。试编写程序，验证任一大于 2 且不超过 n 的偶数都能写成两个质数之和。

```
#include<iostream>
using namespace std;
int main(){
    const int SIZE=1000;
    intn,r,p[SIZE],i,j,k,ans;
    bool tmp;
    cin>>n;
    r=1;
    p[1]=2;
    for (i=3;i<=n;i++){
        ____①____;
        for (j=1;j<=r;j++)
            if (i% ____②____ ==0){
                tmp=false;
                break;
            }
        if (tmp){
            r++;
            ____③____;
        }
    }
    ans=0;
    for (i=2;i<=n/2;i++){
        tmp=false;
        for (j=1;j<=r;j++)
            for (k=j;k<=r;k++)
                if (i+i== ____④____ ){
                    tmp=true;
                    break;
                }
        if (tmp)
            ans++;
    }
    cout<<ans<<endl;
    return 0;
}
```

若输入 n 为 2010，则输出 ⑤ 时表示验证成功，即大于 2 且不超过 2010 的偶数都满足哥德巴赫猜想。

●选择题

(1)①处应填(　　)。

A. k=0　　B. p[i]=i

C. tmp=true　　D. p[i]=r

(2)②处应填(　　)。

A. p[j]　　B. p[n]

C. n　　D. j

(3)③处应填(　　)。

A. p[r]=i　　B. p[j]=i

C. tmp=true　　D. ans++

(4)④处应填(　　)。

A. j+k　　B. p[j]+p[k]

C. p[j]　　D. p[k]

(5)⑤处应填(　　)。

A. 2010　　B. 1005

C. 2009　　D. 1004

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	C	A	A	B	D

【分析】主程序的第一段是寻找所有不大于 n 的素数，对于 i，若所有小于 i 的素数均不能被它整除，则 i 是素数。数组 p 用于保存素数。

(1)tmp 的值用于判断 i 是否是素数。

(2)利用循环将 i 和数组 p 的每一个素数相除，以此判断 i 是否为素数。

(3)是将这些素数存储到 p 中。

(4)i+i 是当前需要验证的偶数，j、k 是枚举 p 中的项，若有 p[j]+p[k]=i+i，则表示 i+i 满足哥德巴赫猜想，并将 ans 加 1。

(5)是输出一共有多少个数需要验证，注意循环是从 2 * 2=4 开始的，所以需要将 2 排除，则一共有 1004 个偶数。

2.【NOIP2017】快速幂

请完善下面的程序，该程序使用分治法求 $x^p \bmod m$ 的值。

输入：三个不超过 10000 的正整数 x，p，m。

输出：$x^p \bmod m$ 的值。

提示：若 p 为偶数，$x^p=(x^2)^{p/2}$；若 p 为奇数，$x^p=x*(x^2)^{(p-1)/2}$。

```
#include<iostream>
using namespace std;
int x,p,m,i,result;
int main(){
```

```
    cin>>x>>p>>m;
    result= ①  ;
    while ( ②  ){
        if (p%2==1)
            result= ③  ;
        p/=2;
        x= ④  ;
    }
    cout<< ⑤  <<endl;
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. 0　　B. 1

C. p/2　　D. p-1/2

(2)②处应填(　　)。

A. p　　B. x

C. p/2!=0　　D. A&C

(3)③处应填(　　)。

A. result%m　　B. x%m

C. result*x%m　　D. result*result

(4)④处应填(　　)。

A. x*x　　B. x*x%m

C. x%m　　D. x*x%result

(5)⑤处应填(　　)。

A. x　　B. m

C. p　　D. result

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	B	A	C	B	D

【分析】本题考查快速幂算法的实现。

(1)一开始需要初始为1。

(2)应该保证p>0。

(3)应该result=result*x%m。

(4)应该x=x*x%m。

(5)result就是最终的答案。

3.【NOIP2017】大整数除法

给定两个正整数p和q,其中p不超过10^{100},q不超过100000,求p除以q的商和余数。

输入:第一行是p的位数n,第二行是正整数p,第三行是正整数q。

输出:两行,分别是p除以q的商和余数。

```
#include<iostream>
using namespace std;
int p[100];
int n,i,q,rest;
char c;
int main(){
    cin>>n;
    for (i=0;i<n;i++){
        cin>>c;
        p[i]=c-'0';
    }
    cin>>q;
    rest=  ①  ;
    i=1;
    while (  ②  &&i<n){
        rest=rest*10+p[i];
        i++;
    }
    if (rest<q)
        cout<<0<<endl;
    else {
        cout<<  ③  ;
        while (i<n){
            rest=  ④  ;
            i++;
            cout<<rest/q;
        }
        cout<<endl;
    }
    cout<<  ⑤  <<endl;
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. q[0]　　B. n

C. p[0]　　D. 0

(2)②处应填(　　)。

A. rest<q　　B. rest!=q

C. rest<=q　　D. rest

(3)③处应填(　　)。

A. rest/q　　B. rest%q

C. rest　　D. rest&q

(4)④处应填(　　)。

A. pow(i,10)%q+p[i]　　B. rest*10+p[i]%q

C. rest%q*10+p[i]　　D. rest+p[i]

(5)⑤处应填(　　)。

A. rest-q　　B. rest

C. rest|q　　D. rest%q

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	C	A	A	C	D

【分析】本题考查高精度除以单精度算法的实现。

(1)利用竖式计算,先取第一位 p[0]。

(2)第 1 个空和第 2 个空目的是为了做上述的事情,那么这里的判断显然就是 rest<q,当前数字还不足以除出大于 0 的数,则继续添加数字。

(3)这里可以看到,已经判断了商为 0 的情况,那么 else 中应该就是计算商,第一位输出的显然就是 rest/q,即商的第一位答案。

(4)添加下一个数字。

(5)最后输出的是小于 q 的数值,即余数,故答案是 rest%q。

4.【NOIP2018】最大公约数之和

下列程序想要求解整数 n 的所有约数两两之间最大公约数的和对 10007 求余后的值,试补全程序。

举例来说,4 的所有约数是 1,2,4。1 和 2 的最大公约数为 1,2 和 4 的最大公约数为 2,1 和 4 的最大公约数为 1,所以答案为 1+2+1=4。

要求:getDivisor 函数的复杂度为 $O(\sqrt{n})$,gcd 函数的复杂度为 O(logmax(n,m))。

```
#include<iostream>
using namespace std;
const int N=110000,P=10007;
int n;
int a[N],len;
int ans;
void getDivisor(){
    len=0;
    for (int i=1; ① <=n;++i)
        if (n%i==0){
          a[++len]=i;
          if ( ② !=i)a[++len]=n/i;
        }
}
int gcd(int a,int b){
    if (b==0){
      ③ ;
```

```
    }
    return gcd(b, ____④____ );
}
int main(){
    cin>>n;
    getDivisor();
    ans=0;
    for (int i=1;i<=len;++i){
        for (int j=i+1;j<=len;++j){
            ans=( ____⑤____ )%P;
        }
    }
    cout<<ans<<endl;
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. i*2　　B. i*i

C. i+i　　D. i*i*i

(2)②处应填(　　)。

A. n　　B. n/i

C. n%2　　D. n-i

(3)③处应填(　　)。

A. return a　　B. return b

C. return a+b　　D. return a-b

(4)④处应填(　　)。

A. a/b　　B. a

C. a%b　　D. b/a

(5)⑤处应填(　　)。

A. gcd(a[i+1],a[j])　　B. ans+gcd(a[i+1],a[j-1])

C. gcd(a[i],a[j])　　D. ans+gcd(a[i],a[j])

【答案】

题号	(1)	(2)	(3)	(4)	(5)
答案	B	B	A	C	D

【分析】getDivisor()求n的所有约数并保存在数组a中。

(1)枚举约数的时候，需要枚举到n，所以第①空需要填i*i，i*i<=n就是枚举到sqrt(n)。

(2)如果n%i==0，i和n/i都是n的约数，第②空填n/i用于防止得到重复的约数。

(3)第③④空利用辗转相除法求最大公约数函数，分别填入a和a%b。

(4)见上填a%b。

(5)第⑤空两层循环用于枚举约数，依次求两个约数a[i]、a[j]的最大公约数并更新答案ans，所以需要填上ans+gcd(a[i],a[j])。

第六章　2019—2022 年 CSP－J 入门级初赛真题及答案解析

2019 CCF 非专业级别软件能力认证第一轮(CSP－J)入门级 C＋＋语言试题

一、单项选择题(共 15 题,每题 2 分,共计 30 分;每题有且仅有一个正确选项)

1.中国的国家顶级域名是(　　)。

A. .cn　　B. .ch　　C. .chn　　D. .china

2.二进制数 11 1011 1001 0111 和 01 0110 1110 1011 进行逻辑与运算的结果是(　　)。

A. 01 0010 1000 1011　　B. 01 0010 1001 0011

C. 01 0010 1000 0001　　D. 01 0010 1000 0011

3.一个 32 位整型变量占用(　　)个字节。

A. 32　　B. 128　　C. 4　　D. 8

4.若有如下程序段,其中 s、a、b、c 均已定义为整型变量,且 a、c 均已赋值(c 大于 0)

```
s=a;
for(b=1; b<=c; b++) s=s-1;
```

则与上述程序段功能等价的赋值语句是(　　)。

A. s=a－c;　　B. s=a－b;　　C. s=s－c;　　D. s=b－c;

5.设有 100 个已排好序的数据元素,采用折半查找时,最大比较次数为(　　)。

A. 7　　B. 10　　C. 6　　D. 8

6.链表不具有的特点是(　　)。

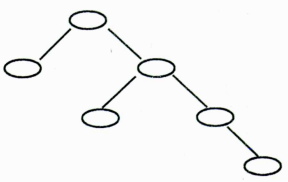
A. 插入删除不需要移动元素　　B. 不必事先估计存储空间

C. 所需空间与线性表长度成正比　　D. 可随机访问任一元素

7.把 8 个同样的球放在 5 个同样的袋子里,允许有的袋子空着不放,共有(　　)种不同的分法。(提示:如果 8 个球都放在一个袋子里,无论是哪个袋子,都只算同一种分法)

A. 22　　B. 24　　C. 18　　D. 20

8.一棵二叉树如右图所示,若采用顺序存储结构,即用一维数组元素存储该二叉树中的结点(根结点的下标为 1,若某结点的下标为 i,则其左孩子位于下标 2i 处、右孩子位于下标 2i＋1 处),则该数组的最大下标至少为(　　)。



A. 6　　B. 10

C. 15　　D. 12

9.100 以内最大的素数是(　　)。

A. 89　　B. 97　　C. 91　　D. 93

10.319 和 377 的最大公约数是(　　)。

A. 27　　B. 33　　C. 29　　D. 31

11.新学期开学了,小胖想减肥,健身教练给小胖制定了两个训练方案。方案一:每次连续跑

3 公里可以消耗 300 千卡(耗时半小时);方案二:每次连续跑 5 公里可以消耗 600 千卡(耗时 1 小时)。小胖每周周一到周四能抽出半小时跑步,周五到周日能抽出一小时跑步。另外,教练建议小胖每周最多跑 21 公里,否则会损伤膝盖。请问如果小胖想严格执行教练的训练方案,并且不想损伤膝盖,每周最多通过跑步消耗(　　)千卡。

A. 3000　　B. 2500　　C. 2400　　D. 2520

12. 一副纸牌除掉大小王有 52 张牌,四种花色,每种花色 13 张。假设从这 52 张牌中随机抽取 13 张纸牌,则至少(　　)张牌的花色一致。

A. 4　　B. 2　　C. 3　　D. 5

13. 一些数字可以颠倒过来看,例如 0、1、8 颠倒过来还是本身,6 颠倒过来是 9,9 颠倒过来看是 6,其他数字颠倒过来都不构成数字。类似的,一些多位数也可以颠倒过来看,比如 106 颠倒过来是 901。假设某个城市的车牌只由 5 位数字组成,每一位都可以取 0 到 9。请问这个城市最多有(　　)个车牌倒过来恰好还是原来的车牌。

A. 60　　B. 125　　C. 75　　D. 100

14. 假设一棵二叉树的后序遍历序列为 DGJHEBIFCA,中序遍历序列为 DBGEHJACIF,则其前序遍历序列为(　　)。

A. ABCDEFGHIJ　　B. ABDEGHJCFI　　C. ABDEGJHCFI　　D. ABDEGHJFIC

15. (　　)是计算机科学领域的最高奖。

A. 图灵奖　　B. 鲁班奖　　C. 诺贝尔奖　　D. 普利策奖

二、阅读程序(程序输入不超过数组或字符串定义的范围;判断题正确填√,错误填×;除特殊说明外,判断题每题 1.5 分,选择题每题 3 分,共计 40 分)

1.

```
01 #include<cstdio>
02 #include<cstring>
03 using namespace std;
04 char st[100];
05 int main(){
06   scanf("%s",st);
07   int n=strlen(st);
08   for(int i=1;i<=n;++i){
09     if(n%i==0){
10          char c=st[i-1];
11          if (c>='a')
12            st[i-1]=c-'a'+'A';
13     }
14   }
15   printf("%s",st);
16   return 0;
17 }
```

●判断题

(1)输入的字符串只能由小写字母或大写字母组成。(　　)

(2)若将第 08 行的 i=1 改为 i=0,程序运行时会发生错误。(　　)

(3)若将第 08 行的 i<=n 改为 i*i<=n,程序运行结果不会改变。(　　)

(4)若输入的字符串全部由大写字母组成,那么输出的字符串就跟输入的字符串一样。(　　)

●选择题

(5)若输入的字符串长度为18,那么输入的字符串跟输出的字符串相比,至多有(　　)个字符不同。

A. 18　　B. 6　　C. 10　　D. 1

(6)若输入的字符串长度为(　　),那么输入的字符串跟输出的字符串相比,至多有36个字符不同。

A. 36　　B. 100000　　C. 1　　D. 128

2.

```
01 include<cstdio>
02 using namespace std;
03 int n,m;
04 int a[100],b[100];
05
06 int main(){
07     scanf("%d%d",&n,&m );
08     for (int i=1;i<=n;++i)
09         a[i]=b[i]=0;
10     for (int i=1;i<=m;++i){
11         int x,y;
12         scanf("%d%d",&x,&y);
13         if (a[x]<y && b[y]<x){
14                 if (a[x]>0)
15                   b[a[x]]=0;
16                 if (b[y]>0)
17                   a[b[y]]=0;
18                 a[x]=y;
19                 b[y]=x;
20         }
21      }
22      int ans=0;
23      for (int i=1;i<=n;++i){
24          if (a[i]==0)
25              ++ans;
26          if (b[i]==0)
27              ++ans;
28      }
29      printf("%d\n",ans);
30      return 0;
31  }
```

假设输入的n和m都是正整数,x和y都是在[1,n]范围内的整数,完成下面的判断题和

单选题：

●判断题

(1)当 m＞0 时，输出的值一定小于 2n。(　　)

(2)执行完第 27 行的++ans 时，ans 一定是偶数。(　　)

(3)a[i]和 b[i]不可能同时大于 0。(　　)

(4)若程序执行到第 13 行时，x 总是小于 y，那么第 15 行不会被执行。(　　)

●选择题

(5)若 m 个 x 两两不同，且 m 个 y 两两不同，则输出的值为(　　)。

A. 2n－2m　　B. 2n＋2　　C. 2n－2　　D. 2n

(6)若 m 个 x 两两不同，且 m 个 y 都相等，则输出的值为(　　)。

A. 2n－2　　B. 2n　　C. 2m　　D. 2n－2m

3.

```
01 #include<iostream>
02 using namespace std;
03 const int maxn=10000;
04 int n;
05 int a[maxn];
06 int b[maxn];
07 int f(int l,int r,int depth){
08      if (l>r)
09        return 0;
10      int min=maxn,mink;
11      for (int i=1;i<=r;++i){
12        if (min>a[i]){
13            min=a[i];
14            mink=i;
15        }
16      }
17      int lres=f(l,mink-1,depth+1);
18      int rres=f(mink+1,r,depth+1);
19      return lres+rres+depth*b[mink];
20  }
21  int main(){
22      cin>>n;
23      for (int i=0;i<n;++i)
24          cin>>a[i];
25      for (int i=0;i<n;++i)
26          cin>>b[i];
27      cout<<f(0,n-1,1)<<endl;
28      return 0;
29  }
```

●判断题

(1)如果 a 数组有重复的数字，则程序运行时会发生错误。(　　)

(2)如果 b 数组全为 0,则输出为 0。(　　)

●选择题

(3)当 n=100 时,最坏情况下,与第 12 行的比较运算执行的次数最接近的是(　　)。

A. 5000　　B. 600　　C. 6　　D. 100

(4)当 n=100 时,最好情况下,与第 12 行的比较运算执行的次数最接近的是(　　)。

A. 100　　B. 6　　C. 5000　　D. 600

(5)当 n=10 时,若 b 数组满足,对任意 0≤i<n,都有 b[i]=i+1,那么输出最大为(　　)。

A. 386　　B. 383　　C. 384　　D. 385

(6)(4 分)当 n=100 时,若 b 数组满足,对任意 0≤i<n,都有 b[i]=1,那么输出最小为(　　)。

A. 582　　B. 580　　C. 579　　D. 581

三、完善程序(单选题,每小题 3 分,共计 30 分)

1.(矩阵变幻)有一个奇幻的矩阵,在不停地变幻,其变幻方式为:数字 0 变成矩阵 $\begin{bmatrix}0&0\\0&1\end{bmatrix}$,数字 1 变成矩阵 $\begin{bmatrix}1&1\\1&0\end{bmatrix}$。最初该矩阵只有一个元素 0,变幻 n 次后,矩阵会变成什么样?

例如,矩阵最初为 [0];矩阵变幻 1 次后为 $\begin{bmatrix}0&0\\0&1\end{bmatrix}$,矩阵变幻 2 次后为 $\begin{bmatrix}0&0&0&0\\0&1&0&1\\0&0&1&1\\0&1&1&0\end{bmatrix}$。

输入一行一个不超过 10 的正整数 n,输出变幻 n 次后的矩阵。

试补全程序。

提示:

"<<"表示二进制左移运算符,例如 $(11)_2<<2=(1100)_2$;

而"∧"表示二进制异或运算符,它将两个参与运算的数中的每个对应的二进制位一一进行比较,若两个二进制位相同,则运算结果的对应二进制位为 0,反之为 1。

```
01 #include<cstdio>
02 using namespace std;
03 int n;
04 const int max_size=1<<10;
05
06 int res[max_size][max_size];
07
08 void recursive(int x,int y,int n,int t){
09       if (n==0){
10          res[x][y]= ①  ;
11          return;
12        }
13       int step=1<<(n-1);
14       recursive( ②  ,n-1,t);
15       recursive(x,y+step,n-1,t);
16       recursive(x+step,y,n-1,t);
```

```
17        recursive( ③ ,n-1,!t);
18    }
19
20    int main(){
21        scanf("%d",&n);
22        recursive(0,0, ④ );
23        int size= ⑤ ;
24        for (int i=0;i<size;++i){
25            for (int j=0;j<size;++j)
26                printf("%d",res[i][j]);
27            puts("");
28        }
29        return 0;
30    }
```

(1)①处应填(　　)。

A. n%2　　B. 0　　C. t　　D. 1

(2)②处应填(　　)。

A. x-step,y-step　　B. x,y-step

C. x-step, y　　D. x,y

(3)③处应填(　　)。

A. x-step,y-step　　B. x+step,y+step

C. x-step,y　　D. x,y-step

(4)④处应填(　　)。

A. n-1,n%2　　B. n,0　　C. n,n% 2　　D. n-1,0

(5)⑤处应填(　　)。

A. 1<<(n+1)　　B. 1<<n　　C. n+1　　D. 1<<(n-1)

2.(计数排序)计数排序是一个广泛使用的排序方法。下面的程序使用双关键字计数排序，将 n 对 10000 以内的整数，从小到大排序。

例如，有三对整数(3,4)、(2,4)、(3,3)，那么排序之后应该是(2,4)、(3,3)、(3,4)。

输入第一行为 n，接下来 n 行，第 i 行有两个数 a[i] 和 b[i]，分别表示第 i 对整数的第一关键字和第二关键字。

从小到大排序后输出。

数据范围 $1\leqslant n\leqslant 10^7$，$1\leqslant a[i],b[i]\leqslant 10^4$。

提示：应先对第二关键字排序，再对第一关键字排序。数组 ord[]存储第二关键字排序的结果，数组 res[]存储双关键字排序的结果。

试补全程序。

```
01 #include<cstdio>
02 #include<cstring>
03 using namespace std;
04 const int maxn=10000000;
05 const int maxs=10000;
06
07 int n;
```

```
08 unsigned a[maxn],b[maxn],res[maxn],ord[maxn];
09 unsigned cnt[maxs+1];
10
11  int main(){
12    scanf("%d",&n);
13    for (int i=0;i<n;++i)
14      scanf("%d %d",&a[i],&b[i]);
15    memset(cnt,0,sizeof(cnt));
16    for (int i=0;i<n;++i)
17      ____①____;  // 利用 cnt 数组统计数量
18     for (int i=0;i<maxs;++i)
19      cnt[i+1]+=cnt[i];
20    for (int i=0;i<n;++i)
21      ____②____;  // 记录初步排序结果
22    memset(cnt,0,sizeof(cnt));
23    for (int i=0;i<n;++i)
24      ____③____;  // 利用 cnt 数组统计数量
25    for (int i=0;i<maxs;++i)
26      cnt[i+1]+=cnt[i];
27    for (int i=n-1;i>=0;--i)
28      ____④____;  // 记录最终排序结果
29    for (int i=0;i<n;++i)
30      printf("%d %d\n",____⑤____);
31    return 0;
32  }
```

(1)① 处应填(　　)。

A. ++cnt[i]　　B. ++cnt[b[i]]

C. ++cnt[a[i]*maxs+ b[i]]　　D. ++cnt[a[i]]

(2)② 处应填(　　)。

A. ord[--cnt[a[i]]]=i　　B. ord[--cnt[b[i]]]=a[i]

C. ord[--cnt[a[i]]]=b[i]　　D. ord[--cnt[b[i]]]=i

(3)③ 处应填(　　)。

A. ++cnt[b[i]]　　B. ++cnt[a[i]*maxs+b[i]]

C. ++cnt[a[i]]　　D. ++cnt[i]

(4)④ 处应填(　　)。

A. res[--cnt[a[ord[i]]]]=ord[i]　　B. res[--cnt[b[ord[i]]]]=ord[i]

C. res[--cnt[b[i]]]=ord[i]　　D. res[--cnt[a[i]]]=ord[i]

(5)⑤处应填(　　)。

A. a[i], b[i]　　B. a[res[i]], b[res[i]]

C. a[ord[res[i]]], b[ord[res[i]]]　　D. a[res[ord[i]]], b[res[ord[i]]]

2020 CCF 非专业级别软件能力认证第一轮(CSP－J)入门级 C＋＋语言试题

一、单项选择题(共 15 题,每题 2 分,共计 30 分;每题有且仅有一个正确选项)

1.在内存储器中每个存储单元都被赋予一个唯一的序号,称为(　　)。

A. 下标　　B. 地址　　C. 序号　　D. 编号

2.编译器的主要功能是(　　)。

A. 将源程序翻译成机器指令代码　　B. 将一种高级语言翻译成另一种高级语言

C. 将源程序重新组合　　D. 将低级语言翻译成高级语言

3.设 x＝true, y＝true, z＝false,以下逻辑运算表达式值为真的是(　　)。

A. (x∧y)∧z　　B. x∧(z∨y)∧z

C. (x∧y)∨(z∨x)　　D. (y∨z)∧x∧z

4.现有一张分辨率为 2048×1024 像素的 32 位真彩色图像。要存储这张图像,需要(　　)的存储空间。

A. 4MB　　B. 8MB　　C. 32MB　　D. 16MB

5.冒泡排序算法的伪代码如下:

```
输入:数组 L,n ≥ 1。输出:按非递减顺序排序的 L。
算法 BubbleSort:
1. FLAG ← n      //标记被交换的最后元素位置
2. while FLAG>1 do
3.        K ← FLAG−1
4.        FLAG ← 1
5.          for j=1 to k do
6.          if L(j)>L(j+1) then do
7.          L(j)←→L(j+1)
8.          FLAG ← j
```

对 n 个数用以上冒泡排序算法进行排序,最少需要比较(　　)次。

A. n　　B. n－2　　C. n^2　　D. n－1

6.设 A 是 n 个实数的数组,考虑下面的递归算法:

```
XYZ (A[1..n])
1.if n=1 then return A[1]
2.else temp ← XYZ (A[1..n−1])
3.      if temp<A[n]
4.      then return temp
5.      else return A[n]
```

则算法 XYZ 的输出是(　　)。

A. A 数组的平均　　B. A 数组的最小值　　C. A 数组的最大值　　D. A 数组的中值

7.链表不具有的特点是(　　)。

A. 插入删除不需要移动元素　　B. 可随机访问任一元素

C. 不必事先估计存储空间　　D. 所需空间与线性表长度成正比

8.有 10 个顶点的无向图至少应该有(　　)条边才能确保是一个连通图。

A. 10　　B. 12　　C. 9　　D. 11

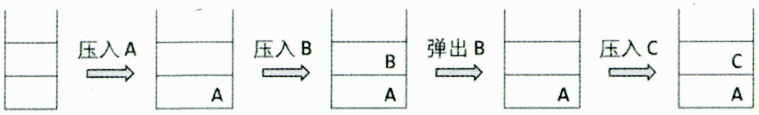
9. 二进制数 1011 转换成十进制数是(　　)。

A. 10　　B. 13　　C. 11　　D. 12

10. 五个小朋友并排站成一列，其中有两个小朋友是双胞胎，如果要求这两个双胞胎必须相邻，则有(　　)种不同排列方法。

A. 24　　B. 36　　C. 72　　D. 48

11. 下图中所使用的数据结构是(　　)。



A. 哈希表　　B. 二叉树　　C. 栈　　D. 队列

12. 独根树的高度为 1。具有 61 个结点的完全二叉树的高度为(　　)。

A. 7　　B. 5　　C. 8　　D. 6

13. 干支纪年法是中国传统的纪年方法，由 10 个天干和 12 个地支组合成 60 个天干地支。由公历年份可以根据以下公式和表格换算出对应的天干地支。

天干＝(公历年份)除以 10 所得余数，地支＝(公历年份)除以 12 所得余数。

天干	甲	乙	丙	丁	戊	己	庚	辛	壬	癸		
	4	5	6	7	8	9	0	1	2	3		
地支	子	丑	寅	卯	辰	巳	午	未	申	酉	戌	亥
	4	5	6	7	8	9	10	11	0	1	2	3

例如，今年是 2020 年，2020 除以 10 余数为 0，查表为“庚”；2020 除以 12，余数为 4，查表为“子”，所以今年是庚子年。

请问 1949 年的天干地支是(　　)。

A. 己亥　　B. 己丑　　C. 己卯　　D. 己酉

14. 10 个三好学生名额分配到 7 个班级，每个班级至少有一个名额，一共有(　　)种不同的分配方案。

A. 56　　B. 84　　C. 72　　D. 504

15. 有五副不同颜色的手套(共 10 只手套，每副手套左右手各 1 只)，一次性从中取 6 只手套，请问恰好能配成两副手套的不同取法有(　　)种。

A. 30　　B. 150　　C. 180　　D. 120

二、阅读程序(程序输入不超过数组或字符串定义的范围；判断题正确填√，错误填×；除特殊说明外，判断题每题 1.5 分，选择题每题 3 分，共计 40 分)

1.

```
01 #include<cstdlib>
02 #include<iostream>
03 using namespace std;
04
05 char encoder[26]={'C','S','P',0};
06 char decoder[26];
```

```
07
08 string st;
09
10 int main(){
11     int k=0;
12     for (int i=0;i<26;++i)
13       if (encoder[i]!=0)++k;
14     for (char x='A';x<='Z';++x) {
15       bool flag=true;
16       for (int i=0;i<26;++i)
17         if (encoder[i]==x){
18           flag=false;
19           break;
20         }
21         if (flag){
22         encoder[k]=x;
23         ++k;
24         }
25     }
26     for (int i=0;i<26;++i)
27       decoder[encoder[i]-'A']=i+'A';
28     cin>>st;
29     for (int i=0;i<st.length();++i)
30       st[i]=decoder[st[i]-'A'];
31     cout<<st;
32     return 0;
33 }
```

●判断题

(1)输入的字符串应当只由大写字母组成,否则在访问数组时可能越界。(　　)

(2)若输入的字符串不是空串,则输入的字符串与输出的字符串一定不一样。(　　)

(3)将第 12 行的“i<26”改为“i<16”,程序运行结果不会改变。(　　)

(4)将第 26 行的“i<26”改为“i<16”,程序运行结果不会改变。(　　)

●单选题

(5)若输出的字符串为 ABCABCABCA,则下列说法正确的是(　　)。

A. 输入的字符串中既有 A 又有 P　　B. 输入的字符串中既有 S 又有 B

C. 输入的字符串中既有 S 又有 P　　D. 输入的字符串中既有 A 又有 B

(6)若输出的字符串为 CSPCSPCSPCSP,则下列说法正确的是(　　)。

A. 输入的字符串中既有 J 又有 R　　B. 输入的字符串中既有 P 又有 K

C. 输入的字符串中既有 J 又有 K　　D. 输入的字符串中既有 P 又有 R

2.

```
01 #include<iostream>
02 using namespace std;
03
04 long long n,ans;
05 int k,len;
06 long long d[1000000];
07
08 int main(){
09     cin>>n>>k;
10     d[0]=0;
11     len=1;
12     ans=0;
13     for (long long i=0;i<n;++i){
14         ++d[0];
15             for (int j=0;j+1<len;++j){
16                 if (d[j]==k){
17                     d[j]=0;
18                     d[j+1]+=1;
19                     ++ans;
20                 }
21             }
22             if (d[len-1]==k){
23                 d[len-1]=0;
24                 d[len]=1;
25                 ++len;
26                 ++ans;
27             }
28     }
29     cout<<ans<<endl;
30     return 0;
31 }
```

假设输入的 n 是不超过 2^{62} 的正整数，k 都是不超过 10000 的正整数，完成下面的判断题和单选题：

●判断题

(1)若 k=1，则输出 ans 时，len=n。(　　)

(2)若 k>1，则输出 ans 时，len 一定小于 n。(　　)

(3)若 k>1，则输出 ans 时，k^{len} 一定大于 n。(　　)

●单选题

(4)若输入的 n 等于 10^{15}，输入的 k 为 1，则输出等于(　　)。

A. $(10^{30}-10^{15})/2$　　B. $(10^{30}+10^{15})/2$

C. 1　　D. 10^{15}

(5)若输入的 n 等于 205,891,132,094,649(即 3^{30})，输入的 k 为 3，则 输出等于(　　)。

A. $(3^{30}-1)/2$　　B. 3^{30}　　C. $3^{30}-1$　　D. $(3^{30}+1)/2$

(6)若输入的 n 等于 100,010,002,000,090,输入的 k 为 10,则输出等于(　　)。

A. 11,112,222,444,543　　B. 11,122,222,444,453

C. 11,122,222,444,543　　D. 11,112,222,444,453

3.

```
#include<algorithm>
#include<iostream>
using namespace std;

int n;
int d[50][2];
int ans;

void dfs(int n,int sum){
  if (n==1){
    ans=max(sum,ans);
    return;
  }
  for (int i=1;i<n;++i){
    int a=d[i-1][0],b=d[i-1][1];
    int x=d[i][0],y=d[i][1];
    d[i-1][0]=a+x;
    d[i-1][1]=b+y;
    for (int j=i;j<n-1;++j)
      d[j][0]=d[j+1][0],d[j][1]=d[j+1][1];
    int s=a+x+abs(b- y);
    dfs(n-1,sum+s);
    for (int j=n-1;j>i;--j)
      d[j][0]=d[j-1][0],d[j][1]=d[j-1][1];
    d[i- 1][0]=a,d[i-1][1]=b;
    d[i][0]=x,d[i][1]=y;
  }
}

int main(){
  cin>>n;
  for (int i=0;i<n;++i)
    cin>>d[i][0];
  for (int i=0;i<n;++i)
    cin>>d[i][1];
  ans=0;
  dfs(n,0);
  cout<<ans<<endl;
  return 0;
}
```

假设输入的 n 是不超过 50 的正整数,d[i][0]、d[i][1]都是不超过 10000 的正整数,完成

下面的判断题和单选题：

●判断题

(1)若输入 n 为 0，此程序可能会死循环或发生运行错误。(　　)

(2)若输入 n 为 20，接下来的输入全为 0，则输出为 0。(　　)

(3)输出的数一定不小于输入的 d[i][0]和 d[i][1]中的任意一个。(　　)

●单选题

(4)若输入的 n 为 20，接下来的输入是 20 个 9 和 20 个 0，则输出为(　　)。

A. 1917　　B. 1908　　C. 1881　　D. 1890

(5)若输入的 n 为 30，接下来的输入是 30 个 0 和 30 个 5，则输出为(　　)。

A. 2020　　B. 2030　　C. 2010　　D. 2000

(6)(4 分)若输入的 n 为 15，接下来的输入是 15 到 1，则输出为(　　)。

A. 2420　　B. 2220　　C. 2440　　D. 2240

三、完善程序(单选题，每小题 3 分，共计 30 分)

1.(质因数分解)给出正整数 n，请输出将 n 质因数分解的结果，结果从小到大输出。

例如：输入 n=120，程序应该输出 2 2 2 3 5，表示 120=2×2×2×3×5。输入保证2≤n≤10^9。提示：先从小到大枚举变量 i，然后用 i 不停试除 n 来寻找所有的质因子。

试补全程序。

```
01 #include<cstdio>
02 using namespace std;
03
04 int n,i;
05
06 int main(){
07   scanf("%d",&n);
08   for(i= ①  ;  ②  <=n;i++){
09     ③  {
10         printf("%d ",i);
11         n=n/i;
12       }
13   }
14   if( ④ )
15   printf("%d ",  ⑤  );
16   return 0;
17  }
```

(1)①处应填(　　)。

A. n-1　　B. 0　　C. 1　　D. 2

(2)②处应填(　　)。

A. n/i　　B. n/(i*i)　　C. i*i*i　　D. i*i

(3)③处应填(　　)。

A. if (i*i<=n)　　B. if (n%i==0)

C. while (i*i<=n)　　D. while(n%i==0)

(4)④处应填(　　)。

A. n>1　　B. n<=1　　C. i+i<=n　　D. i<n/i

(5)⑤处应填(　　)。

A. 2　　B. i　　C. n/i　　D. n

2.(最小区间覆盖)给出 n 个区间，第 i 个区间的左右端点是[a_i, b_i]。现在要在这些区间中选出若干个，使得区间[0, m]被所选区间的并覆盖(即每一个 0≤i≤m 都在某个所选的区

间中）。保证答案存在，求所选区间个数的最小值。

输入第一行包含两个整数 n 和 m（$1 \leqslant n \leqslant 5000$，$1 \leqslant m \leqslant 10^9$）。

接下来 n 行，每行两个整数 a_i，b_i（$0 \leqslant a_i, b_i \leqslant m$）。

提示：使用贪心法解决这个问题。先用 $O(n^2)$ 的时间复杂度排序，然后贪心选择这些区间。

试补全程序。

```
#include<iostream>

using namespace std;

const int MAXN=5000;
int n,m;
struct segment{ int a,b;} A[MAXN];

void sort() //排序
{
  for (int i=0;i<n;i++)
  for (int j=1;j<n;j++)
    if ( ① )
    {
        segment t=A[j];
        ②
    }
}

int main()
{
  cin>>n>>m;
  for (int i=0;i<n;i++)
    cin>>A[i].a>>A[i].b;
  sort();
  int p=1;
  for (int i=1;i<n;i++)
    if ( ③ )
      A[p++]=A[i];
  n=p;
  int ans=0,r=0;
  int q=0;
  while (r<m)
  {
    while ( ④ )
      q++;
    ⑤ ;
    ans++;
```

```
39   }
40   cout<<ans<<endl;
41   return 0;
42 }
```

(1)①处应填(　　)。

A. A[j].b<A[j-1].b　　B. A[j].b>A[j-1].b

C. A[j].a<A[j-1].a　　D. A[j].a>A[j-1].a

(2)②处应填(　　)。

A. A[j-1]=A[j]; A[j]=t;　　B. A[j+1]=A[j]; A[j]=t;

C. A[j]=A[j-1]; A[j-1]=t;　　D. A[j]=A[j+1]; A[j+1]=t;

(3)③处应填(　　)。

A. A[i].b<A[p-1].b　　B. A[i].b>A[i-1].b

C. A[i].b>A[p-1].b　　D. A[i].b<A[i-1].b

(4)④处应填(　　)。

A. q+1<n && A[q+1].b<=r　　B. q+1<n && A[q+1].a<=r

C. q<n && A[q].a<=r　　D. q<n && A[q].b<=r

(5)⑤处应填(　　)。

A.r=max(r, A[q+1].a)　　B. r=max(r, A[q].b)

C. r=max(r, A[q+1].b)　　D. q++

2021 CCF 非专业级别软件能力认证第一轮(CSP－J)入门级 C＋＋语言试题

一、单项选择题(共 15 题，每题 2 分，共计 30 分；每题有且仅有一个正确选项)

1. 以下不属于面向对象程序设计语言的是(　　)。

A. C++　　B. Python　　C. Java　　D. C

2. 以下奖项与计算机领域最相关的是(　　)。

A. 奥斯卡奖　　B. 图灵奖　　C. 诺贝尔奖　　D. 普利策奖

3. 目前主流的计算机储存数据最终都是转换成(　　)数据进行储存。

A. 二进制　　B. 十进制　　C. 八进制　　D. 十六进制

4. 以比较作为基本运算，在 N 个数中找出最大数，最坏情况下所需要的最少的比较次数为(　　)。

A. N^2　　B. N　　C. N－1　　D. N＋1

5. 对于入栈顺序为 a，b，c，d，e 的序列，下列(　　)不是合法的出栈序列。

A. a，b，c，d，e　　B. e，d，c，b，a　　C. b，a，c，d，e　　D. c，d，a，e，b

6. 对于有 n 个顶点、m 条边的无向连通图(m＞n)，需要删掉(　　)条边才能使其成为一棵树。

A. n－1　　B. m－n　　C. m－n－1　　D. m－n＋1

7. 二进制数 101.11 对应的十进制数是(　　)。

A. 6.5　　B. 5.5　　C. 5.75　　D. 5.25

8. 如果一棵二叉树只有根结点，则这棵二叉树高度为 1。那么高度为 5 的完全二叉树有(　　)种不同的形态。

A. 16　　B. 15　　C. 17　　D. 32

9. 表达式 a＊(b＋c)＊d 的后缀表达式为(　　)，其中“＊”和“＋”是运算符。

A. ＊＊a＋bcd　　B. abc＋＊d＊　　C. abc＋d＊＊　　D. ＊a＊＋bcd

10. 6 个人，两个人组一队，总共组成三队，不区分队伍的编号。不同的组队情况有(　　)种。

A. 10　　B. 15　　C. 30　　D. 20

11. 在数据压缩编码中的哈夫曼编码方法，在本质上是一种(　　)的策略。

A. 枚举　　B. 贪心　　C. 递归　　D. 动态规划

12. 由 1，1，2，2，3 这五个数字组成不同的三位数有(　　)种。

A. 18　　B. 15　　C. 12　　D. 24

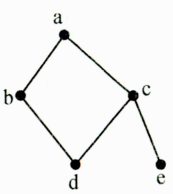
13. 考虑如下递归算法

```
solve(n)
    if n<=1 return 1
    else if n>=5 return n*solve(n-2)
    else return n*solve(n-1)
```

则调用 solve(7)得到的返回结果为(　　)。

A. 105　　B. 840　　C. 210　　D. 420

14. 以 a 为起点，对右边的无向图进行深度优先遍历，则 b、c、d、e 四个点中有可能作为最后一个遍历到的点的个数为(　　)。



A. 1　　B. 2　　C. 3　　D. 4

15.有四个人要从A点坐一条船过河到B点，船一开始在A点。该船一次最多可坐两个人。已知这四个人中每个人独自坐船的过河时间分别为1,2,4,8,且两个人坐船的过河时间为两人独自过河时间的较大者。则最短用时(　　)时间可以让四个人都过河到B点(包括从B点把船开回A点的时间)。

A. 14　　B. 15　　C. 16　　D. 17

二、阅读程序(程序输入不超过数组或字符串定义的范围;判断题正确填√,错误填×;除特殊说明外,判断题每题1.5分,选择题每题3分,共计40分)

1.

```
01 #include<iostream>
02 using namespace std;
03
04 int n;
05 int a[1000];
06
07 int f(int x)
08 {
09     int ret=0;
10     for (;x;x &=x-1) ret++;
11     return ret;
12 }
13
14 int g(int x)
15 {
16     return x &-x;
17 }
18
19 int main()
20 {
21     cin>>n;
22     for (int i=0;i<n;i++) cin>>a[i];
23     for (int i=0;i<n;i++)
24         cout<<f(a[i])+g(a[i])<<' ';
25     cout<<endl;
26     return 0;
27 }
```

●判断题

(1)输入的n等于1001时,程序不会发生下标越界。(　　)

(2)输入的a[i]必须全为正整数,否则程序将陷入死循环。(　　)

(3)当输入为5 2 11 9 16 10时,输出为3 4 3 17 5。(　　)

(4)当输入为1 511998时,输出为18。(　　)

(5)将源代码中g函数的定义(14—17行)移到main函数的后面,程序可以正常编译运行。(　　)

●单选题

(6)当输入为 2 －65536 2147483647 时,输出为(　　)。

A. 65532 33　　B. 65552 32

C. 65535 34　　D. 65554 33

2.

```
01 #include<iostream>
02 #include<string>
03 using namespace std;
04
05 char base[64];
06 char table[256];
07
08 void init()
09 {
10     for (int i=0;i<26;i++) base[i]='A'+i;
11     for (int i=0;i<26;i++) base[26+i]='a'+i;
12     for (int i=0;i<10;i++) base[52+i]='0'+i;
13     base[62]='+',base[63]='/';
14
15     for (int i=0;i<256;i++) table[i]=0xff;
16     for (int i=0;i<64;i++) table[base[i]]=i;
17     table['=']=0;
18 }
19
20 string decode(string str)
21 {
22     string ret;
23     int i;
24     for (i=0;i<str.size();i+=4) {
25       ret+=table[str[i]]<<2 | table[str[i+1]]>>4;
26       if (str[i+2]!='=')
27          ret+=(table[str[i+1]] & 0x0f)<<4 | table[str[i+2]]>>2;
28       if (str[i+3] !='=')
29          ret+=table[str[i+2]]<<6 | table[str[i+3]];
30     }
31     return ret;
32 }
33
34 int main()
35 {
36     init();
37     cout<<int(table[0])<<endl;
```

```
38
39      string str;
40      cin>>str;
41      cout<<decode(str)<<endl;
42      return 0;
43 }
```

●判断题

(1)输出的第二行一定是由小写字母、大写字母、数字和“+”“/”“=”构成的字符串。(　　)

(2)可能存在输入不同,但输出的第二行相同的情形。(　　)

(3)输出的第一行为-1。(　　)

●单选题

(4)设输入字符串长度为 n,decode 函数的时间复杂度为(　　)。

A. $O(\sqrt{n})$　　B. $O(n)$　　C. $O(n \log n)$　　D. $O(n^2)$

(5)当输入为 Y3Nx 时,输出的第二行为(　　)。

A. csp　　B. csq　　C. CSP　　D. Csp

(6)当输入为 Y2NmIDIwMjE=时,输出的第二行为(　　)。

A. ccf2021　　B. ccf2022

C. ccf 2021　　D. ccf 2022

3.

```
01 #include<iostream>
02 using namespace std;
03
04 const int n=100000;
05 const int N=n+1;
06
07 int m;
08 int a[N],b[N],c[N],d[N];
09 int f[N],g[N];
10
11 void init()
12 {
13     f[1]=g[1]=1;
14     for (int i=2;i<=n;i++) {
15         if (!a[i]) {
16             b[m++]=i;
17             c[i]=1,f[i]=2;
18             d[i]=1,g[i]=i+1;
19         }
20         for (int j=0;j<m && b[j]*i<=n;j++) {
21             int k=b[j];
22             a[i*k]=1;
23             if (i%k==0) {
```

```
24                c[i*k]=c[i]+1;
25                f[i*k]=f[i]/c[i*k]*(c[i*k]+1);
26                d[i*k]=d[i];
27                g[i*k]=g[i]*k+d[i];
28                break;
29            }
30            else {
31                c[i*k]=1;
32                f[i*k]=2*f[i];
33                d[i*k]=g[i];
34                g[i*k]=g[i]*(k+1);
35            }
36        }
37    }
38 }
39
40 int main()
41 {
42    init();
43
44    int x;
45    cin>>x;
46    cout<<f[x]<<'  '<<g[x]<<endl;
47    return 0;
48 }
```

假设输入的 x 是不超过 1000 的自然数，完成下面的判断题和单选题：

●判断题

(1)若输入不为 1，把第 13 行删去不会影响输出的结果。(　　)

(2)第 25 行的 f[i]/c[i * k]可能存在无法整除而向下取整的情况。(　　)

(3)在执行完 init()后，f 数组不是单调递增的，但 g 数组是单调递增的。(　　)

●单选题

(4)init 函数的时间复杂度为(　　)。

A. O(n)　　B. O(n log n)　　C. O($\sqrt{n}$)　　D. O(n^2)

(5)在执行完 init()后，f[1]，f[2]，f[3]，…，f[100]中有(　　)个等于 2。

A. 23　　B. 24　　C. 25　　D. 26

(6)当输入为 1000 时，输出为(　　)。

A. 15 1340　　B. 15 2340　　C. 16 2340　　D. 16 1340

三、完善程序(单选题，每小题 3 分，共计 30 分)

1.(Josephus 问题)有 n 个人围成一个圈，依次标号 0 至 n－1。从 0 号开始，依次 0，1，0，1，…交替报数，报到 1 的人会离开，直至圈中只剩下一个人，求最后剩下人的编号。

试补全模拟程序。

```
01 #include<iostream>
02
03 using namespace std;
04
05 const int MAXN=1000000;
06 int F[MAXN];
07
08 int main() {
09     int n;
10     cin>>n;
11     int i=0,p=0,c=0;
12     while ( ① ) {
13         if (F[i]==0) {
14             if ( ② ) {
15                 F[i]=1;
16                 ③ ;
17             }
18             ④ ;
19         }
20         ⑤ ;
21     }
22     int ans=-1;
23     for (i=0;i<n;i++)
24         if (F[i]==0)
25           ans=i;
26     cout<<ans<<endl;
27     return 0;
28 }
```

(1)①处应填(　　)。

A. i<n　　B. c<n　　C. i<n-1　　D. c<n-1

(2)②处应填(　　)。

A. i%2==0　　B. i%2==1　　C. p　　D. !p

(3)③处应填(　　)。

A. i++　　B. i=(i+1)%n　　C. c++　　D. p ^=1

(4)④处应填(　　)。

A. i++　　B. i=(i+1)%n　　C. c++　　D. p ^=1

(5)⑤处应填(　　)。

A. i++　　B. i=(i+1)%n　　C. c++　　D. p ^=1

2.(矩形计数)平面上有 n 个关键点,求有多少个四条边都和 x 轴或者 y 轴平行的矩形,满足四个顶点都是关键点。给出的关键点可能有重复,但完全重合的矩形只计一次。

试补全枚举算法。

```
01 #include<iostream>
02
03 using namespace std;
04
05 struct point{
06     int x,y,id;
07 };
08
09 bool equals(point a,point b){
10     return a.x==b.x && a.y==b.y;
11 }
12
13 bool cmp(point a,point b){
14     return ____①____;
15 }
16
17 void sort(point A[],int n){
18     for (int i=0;i<n;i++)
19         for (int j=1;j<n;j++)
20             if (cmp(A[j],A[j-1])){
21                 point t=A[j];
22                 A[j]=A[j-1];
23                 A[j-1]=t;
24             }
25 }
26
27 int unique(point A[],int n){
28     int t=0;
29     for (int i=0;i<n;i++)
30         if (____②____)
31             A[t++]=A[i];
32     return t;
33 }
34
35 bool binary_search(point A[],int n,int x,int y){
36     point p;
37     p.x=x;
38     p.y=y;
39     p.id=n;
40     int a=0,b=n-1;
41     while (a<b){
42         int mid=____③____;
```

```
43          if ( ④ )
44              a=mid+1;
45          else
46              b=mid;
47      }
48      return equals(A[a],p);
49  }
50
51  const int MAXN=1000;
52  point A[MAXN];
53
54  int main(){
55      int n;
56      cin>>n;
57      for (int i=0;i<n;i++){
58          cin>>A[i].x>>A[i].y;
59          A[i].id=i;
60      }
61      sort(A,n);
62      n=unique(A,n);
63      int ans=0;
64      for (int i=0;i<n;i++)
65          for (int j=0;j<n;j++)
66              if ( ⑤ && binary_search(A,n,A[i].x,A[j].y) &&binary_
search(A,n,A[j].x,A[i].y)){
67                  ans++;
68              }
69      cout<<ans<<endl;
70      return 0;
71  }
```

(1)①处应填(　　)。

A. a.x !=b.x ? a.x<b.x : a.id<b.id

B. a.x !=b.x ? a.x<b.x : a.y<b.y

C. equals(a, b) ? a.id<b.id : a.x<b.x

D. equals(a, b) ? a.id<b.id : (a.x !=b.x ? a.x<b.x : a.y<b.y)

(2)②处应填(　　)。

A. i==0 || cmp(A[i], A[i-1])

B. t==0 || equals(A[i], A[t-1])

C. i==0 || !cmp(A[i], A[i-1])

D. t==0 || !equals(A[i], A[t-1])

(3)③处应填(　　)。

A. b-(b-a)/2+1　　B. (a+b+1)>>1

C. (a+b)> > 1　　D. a+(b-a+1)/2

(4)④处应填(　　)。

A. !cmp(A[mid],p)　　B. cmp(A[mid],p)

C. cmp(p, A[mid])　　D. !cmp(p,A[mid])

(5)⑤处应填(　　)。

A. A[i].x==A[j].x

B. A[i].id<A[j].id

C. A[i].x==A[j].x && A[i].id<A[j].id

D. A[i].x<A[j].x && A[i].y<A[j].y

2022 CCF 非专业级别软件能力认证第一轮(CSP－J)
入门级 C＋＋语言试题

一、单项选择题(共 15 题，每题 2 分，共计 30 分；每题有且仅有一个正确选项)

1.以下哪种功能没有涉及 C++语言的面向对象特性支持？(　　)。

A. C++中调用 printf 函数

B. C++中调用用户定义的类成员函数

C. C++中构造一个 class 或 struct

D. C++中构造来源于同一基类的多个派生类

2.有 6 个元素，按照 6、5、4、3、2、1 的顺序进入栈 S，请问下列哪个出栈序列是非法的(　　)。

A. 5 4 3 6 1 2　　B. 4 5 3 1 2 6

C. 3 4 6 5 2 1　　D. 2 3 4 1 5 6

3. 运行以下代码片段的行为是(　　)。

```
int x=101;
int y=201;
int *p=&x;
int *q=&y;
p=q;
```

A. 将 x 的值赋为 201　　B. 将 y 的值赋为 101

C. 将 q 指向 x 的地址　　D. 将 p 指向 y 的地址

4.链表和数组的区别包括(　　)。

A. 数组不能排序，链表可以

B. 链表比数组能存储更多的信息

C. 数组大小固定，链表大小可动态调整

D. 以上均正确

5. 对假设栈 S 和队列 Q 的初始状态为空。存在 e1～e6 六个互不相同的数据，每个数据按照进栈 S、出栈 S、进队列 Q、出队列 Q 的顺序操作，不同数据间的操作可能会交错。已知栈 S 中依次有数据 e1、e2、e3、e4、e5 和 e6 进栈，队列 Q 依次有数据 e2、e4、e3、e6、e5 和 e1 出队列。则栈 S 的容量至少是(　　)个数据。

A. 2　　B. 3　　C. 4　　D. 6

6. 对表达式 a＋(b－c)＊d 的前缀表达式为(　　)，其中＋、－、＊是运算符。

A. ＊＋a－bcd　　B. ＋a＊－bcd

C. abc－d＊＋　　D. abc－＋d

7. 假设字母表 {a，b，c，d，e} 在字符串出现的频率分别为 10%，15%，30%，16%，29%。若使用哈夫曼编码方式对字母进行不定长的二进制编码，字母 d 的编码长度为(　　)位。

A. 1　　B. 2　　C. 2 或 3　　D. 3

8. 一棵有 n 个结点的完全二叉树用数组进行存储与表示，已知根结点存储在数组的第 1 个位置。若存储在数组第 9 个位置的结点存在兄弟结点和两个子结点，则它的兄弟结点和右子结点的位置分别是(　　)。

A. 8、18　　B. 10、18　　C. 8、19　　D. 10、19

9. 考虑由 N 个顶点构成的有向连通图，采用邻接矩阵的数据结构表示时，该矩阵中至少存在（　　）个非零元素。

A. N－1　　B. N　　C. N＋1　　D. N^2

10. 以下对数据结构的表述不恰当的一项为（　　）。

A. 图的深度优先遍历算法常使用的数据结构为栈

B. 栈的访问原则为后进先出，队列的访问原则是先进先出

C. 队列常常被用于广度优先搜索算法

D. 栈与队列存在本质不同，无法用栈实现队列

11. 以下哪组操作能完成在双向循环链表结点 p 之后插入结点 s 的效果（其中，next 域为结点的直接后继，prev 域为结点的直接前驱）？（　　）。

A. `p->next->prev=s; s->prev=p; p->next=s; s->next=p->next;`

B. `p->next->prev=s; p->next=s; s->prev=p; s->next=p->next;`

C. `s->prev=p; s->next=p->next; p->next=s; p->next->prev=s;`

D. `s->next=p->next; p->next->prev=s; s->prev=p; p->next=s;`

12. 以下排序算法的常见实现中，哪个选项的说法是错误的？（　　）。

A. 冒泡排序算法是稳定的　　B. 简单选择排序是稳定的

C. 简单插入排序是稳定的　　D. 归并排序算法是稳定的

13. 八进制数 32.1 对应的十进制数是（　　）。

A. 24.125　　B. 24.250　　C. 26.125　　D. 26.250

14. 一个字符串中任意个连续的字符组成的子序列称为该字符串的子串，则字符串 abcab 有（　　）个内容互不相同的子串。

A. 12　　B. 13　　C. 14　　D. 15

15. 以下对递归方法的描述中，正确的是（　　）

A. 递归是允许使用多组参数调用函数的编程技术

B. 递归是通过调用自身来求解问题的编程技术

C. 递归是面向对象和数据而不是功能和逻辑的编程语言模型

D. 递归是将用某种高级语言转换为机器代码的编程技术

二、阅读程序（程序输入不超过数组或字符串定义的范围；判断题正确填√，错误填×；除特殊说明外，判断题每题 1.5 分，选择题每题 3 分，共计 40 分）

(一)

```
01 #include <iostream>
02
03 using namespace std;
04
05 int main()
06 {
07     unsigned short x, y;
08     cin>>x>>y;
09     x=(x | x<<2) & 0x33;
10     x=(x | x<<1) & 0x55;
11     y=(y | y<<2) & 0x33;
```

```
12    y=(y | y<<1) & 0x55;
13    unsigned short z=x | y<<1;
14    cout<<z<<endl;
15    return 0;
16 }
```

假设输入的 x、y 均是不超过 15 的自然数，完成下面的判断题和单选题：

●判断题

16. 删去第 7 行与第 13 行的 unsigned，程序行为不变。（　　）

17. 将第 7 行与第 13 行的 short 均改为 char，程序行为不变。（　　）

18. 程序总是输出一个整数“0”。（　　）

19. 当输入为“2 2”时，输出为“10”。（　　）

20. 当输入为“2 2”时，输出为“59”。（　　）

●单选题

21. 当输入为“13 8”时，输出为（　　）。

A. “0”　　B.“209”　　C. “197”　　D. “226”

(二)

```
01 #include <algorithm>
02 #include <iostream>
03 #include <limits>
04
05 using namespace std;
06
07 const int MAXN=105;
08 const int MAXK=105;
09
10 int h[MAXN][MAXK];
11
12 int f(int n, int m)
13 {
14    if (m==1) return n;
15    if (n==0) return 0;
16
17    int ret=numeric_limits<int> ::max();
18    for (int i=1; i<=n; i++)
19       ret=min(ret, max(f(n-i, m), f(i-1, m-1))+1);
20    return ret;
21 }
22
23 int g(int n, int m)
24 {
25    for (int i=1; i<=n; i++)
26       h[i][1]=i;
```

```
27    for (int j=1; j<=m; j++)
28        h[0][j]=0;
29
30    for (int i=1; i<=n; i++) {
31        for (int j=2; j<=m; j++) {
32            h[i][j]=numeric_limits<int> ::max();
33            for (int k=1; k<=i; k++)
34            h[i][j]=min(
35                h[i][j],
36         max(h[i-k][j], h[k-1][j-1])+1);
37        }
38    }
39
40    return h[n][m];
41 }
42
43 int main()
44 {
45    int n, m;
46    cin>>n>>m;
47    cout<<f(n, m)<<endl<<g(n, m)<< endl;
48    return 0;
49 }
```

假设输入的 n、m 均是不超过 100 的正整数，完成下面的判断题和单选题：

●判断题

22. 当输入为“7 3”时，第 19 行用来取最小值的 min 函数执行了 449 次。（　　）

23. 输出的两行整数总是相同的。（　　）

24. 当 m 为 1 时，输出的第一行总为 n。（　　）

●单选题

25. 算法 g(n,m)最为准确的时间复杂度分析结果为（　　）。

A. $O(n^{\frac{3}{2}}m)$　　B. $O(nm)$　　C. $O(n^2m)$　　D. $O(nm^2)$

26. 当输入为“20 2”时，输出的第一行为（　　）。

A. “4”　　B. “5”　　C. “6”　　D. “20”

27.（4 分）当输入为“100 100”时，输出的第一行为（　　）。

A. “6”　　B. “7”　　C. “8”　　D. “9”

（三）

```
01 #include<iostream>
02
03 using namespace std;
04
05 int n, k;
06
```

```
07 int solve1()
08 {
09     int l=0, r=n;
10     while (l<=r) {
11     int mid=(l+r)/2;
12         if (mid *mid<=n) l=mid+1;
13         else r=mid-1;
14     }
15     return l-1;
16 }
17
18 double solve2(double x)
19 {
20     if (x==0) return x;
21     for (int i=0; i<k; i++)
22       x=(x+n/x)/2;
23     return x;
24 }
25
26 int main()
27 {
28     cin>>n>>k;
29     double ans=solve2(solve1());
30     cout<<ans<<' '<<(ans*ans==n)<<endl;
31     return 0;
32 }
```

假设 int 为 32 位有符号整数类型，输入的 n 是不超过 47000 的自然数、k 是不超过 int 表示范围的自然数，完成下面的判断题和单选题：

●判断题

28. 该算法最准确的时间复杂度分析结果为 O(log n+k)。（　　）

29. 当输入为“9801 1”时，输出的第一个数为“99”。（　　）

30. 对于任意输入的 n，随着输入 k 的增大，输出的第二个数会变成“1”。（　　）

31. 该程序存在缺陷。当输入的 n 过大时，第 12 行的乘法有可能溢出，因此应当将 mid 强制转换为 64 位整数再计算。（　　）

●单选题

32. 当输入为“2 1”时，输出的第一个数最接近（　　）。

A. 1　　B. 1.414　　C. 1.5　　D. 2

33. 当输入为“3 10”时，输出的第一个数最接近（　　）。

A. 1.7　　B. 1.732　　C. 1.75　　D. 2

34. 当输入为“256 11”时，输出的第一个数（　　）。

A. 等于 16　　B. 接近但小于 16

C. 接近但大于 16　　D. 前三种情况都有可能

三、完善程序（单选题，每小题 3 分，共计 30 分）

（一）（枚举因数）从小到大打印正整数 n 的所有正因数。

试补全枚举程序。

```
01 #include<bits/stdc++.h>
02 using namespace std;
03
04 int main() {
05   int n;
06   cin>>n;
07
08  vector<int>fac;
09  fac.reserve((int)ceil(sqrt(n)));
10
11  int i;
12  for (i=1; i*i<n;++i) {
13    if ( ① ) {
14    fac.push_back(i);
15    }
16   }
17
18 for (int k=0; k<fac.size();++k) {
19  cout<< ② <<" ";
20  }
21   if ( ③ ) {
22    cout<< ④ <<" ";
23   }
24  for (int k=fac.size()-1; k>=0;--k) {
25    cout<< ⑤ <<" ";
26   }
27 }
```

35. ①处应填（　　）。

A. `n%i==0`　　B. `n%i==1`

C. `n%(i-1)==0`　　D. `n%(i-1)==1`

36. ②处应填（　　）。

A. `n/fac[k]`　　B. `fac[k]`

C. `fac[k]-1`　　D. `n/(fac[k]-1)`

37. ③处应填（　　）。

A. `(i-1)*(i-1)==n`　　B. `(i-1)*i==n`

C. `i*i==n`　　D. `i*(i-1)==n`

38. ④处应填（　　）。

A. `n-i`　　B. `n-i+1`

C. `i-1`　　D. `i`

39.⑤处应填(　　)。

A. n/fac[k]　　B. fac[k]

C. fac[k]-1　　D. n/(fac[k]-1)

(二)(洪水填充)现有用字符标记像素颜色的8×8图像。颜色填充的操作描述如下:给定起始像素的位置和待填充的颜色,将起始像素和所有可达的像素(可达的定义:经过一次或多次的向上、下、左、右四个方向移动所能到达且终点和路径上所有像素的颜色都与起始像素颜色相同),替换为给定的颜色。

试补全程序。

```
01 #include<bits/stdc++.h>
02 using namespace std;
03
04 const int ROWS=8;
05 const int COLS=8;
06
07 struct Point {
08 int r, c;
09  Point(int r, int c) : r(r), c(c) {}
10 };
11
12 bool is_valid(char image[ROWS][COLS], Point pt,
13 int prev_color, int new_color) {
14    int r=pt.r;
15    int c=pt.c;
16    return (0<=r && r<ROWS && 0<=c && c<COLS &&
17       ①  && image[r][c]!=new_color);
18 }
19
20 void flood_fill(char image[ROWS][COLS], Point cur, int new_color) {
21  queue<Point>queue;
22  queue.push(cur); 23
24  int prev_color=image[cur.r][cur.c];
25   ②  ;
26
27  while (!queue.empty()) {
28    Point pt=queue.front();
29    queue.pop();
30
31   Point points[4]={  ③  , Point(pt.r-1, pt.c),
32                  Point(pt.r, pt.c+1), Point(pt.r, pt.c-1)};
33    for (auto p : points) {
34      if (is_valid(image, p, prev_color, new_color)) {
35        ④  ;
```

```
36          ⑤ ;
37        }
38      }
39    }
40 }
41
42 int main() {
43 char image[ROWS][COLS]={{'g', 'g', 'g', 'g', 'g', 'g', 'g', 'g'},
44                         {'g', 'g', 'g', 'g', 'g', 'g', 'r', 'r'},
45                         {'g', 'r', 'r', 'g', 'g', 'r', 'g', 'g'},
46                         {'g', 'b', 'b', 'b', 'b', 'r', 'g', 'r'},
47                         {'g', 'g', 'g', 'b', 'b', 'r', 'g', 'r'},
48                         {'g', 'g', 'g', 'b', 'b', 'b', 'b', 'r'},
49                         {'g', 'g', 'g', 'g', 'g', 'b', 'g', 'g'},
50                         {'g', 'g', 'g', 'g', 'g', 'b', 'b', 'g'}};
51
52  Point cur(4, 4);
53  char new_color='y';
54
55  flood_fill(image, cur, new_color);
56
57  for (int r=0; r<ROWS; r++) {
58    for (int c=0; c<COLS; c++) {
59      cout<<image[r][c]<<" ";
60    }
61    cout<<endl;
62   }
63  // 输出：
64  //g g g g g g g g
65  //g g g g g g r r
66  //g r r g g r g g
67  //g y y y y r g r
68  //g g g y y r g r
69  //g g g y y y y r
70  //g g g g g y g g
71  //g g g g g y y g
72
73  return 0;
74 }
```

40. ①处应填(　　)。

A. `image[r][c]==prev_color`

B. `image[r][c]!=prev_color`

C. `image[r][c]==new_color`

D. `image[r][c]!=new_color`

41. ②处应填(　　)。

A. `image[cur.r+1][cur.c]=new_color`

B. `image[cur.r][cur.c]=new_color`

C. `image[cur.r][cur.c+1]=new_color`

D. `image[cur.r][cur.c]=prev_color`

42. ③处应填(　　)。

A. `Point(pt.r, pt.c)`　　B. `Point(pt.r, pt.c+1)`

C. `Point(pt.r+1, pt.c)`　　D. `Point(pt.r+1, pt.c+1)`

43. ④处应填(　　)。

A. `prev_color=image[p.r][p.c]`

B. `new_color=image[p.r][p.c]`

C. `image[p.r][p.c]=prev_color`

D. `image[p.r][p.c]=new_color`

44. ⑤处应填(　　)。

A. `queue.push(p)`　　B. `queue.push(pt)`

C. `queue.push(cur)`　　D. `queue.push(Point(ROWS,COLS))`

答案解析

2019 CCF 非专业级别软件能力认证第一轮(CSP－J)入门级 C＋＋语言试题参考答案及解析

【答案】

一、单项选择题

1	2	3	4	5	6	7	8	9	10
A	D	C	A	A	D	C	C	B	C
11	12	13	14	15					
C	A	C	B	A					

二、阅读程序

1.(1)×　(2)√　(3)×　(4)√　(5)B　(6)B

2.(1)√　(2)×　(3)×　(4)×　(5)A　(6)A

3.(1)×　(2)√　(3)A　(4)D　(5)D　(6)B

三、完善程序

第 1 题					第 2 题				
(1)	(2)	(3)	(4)	(5)	(1)	(2)	(3)	(4)	(5)
C	D	B	B	B	B	D	C	A	B

【解析】

一、单项选择题

1.常识题，中国国家顶级域名为“.cn”。

2.本题考查二进制的基本运算。按位与的操作是对二进制每一位对应做与，两个数这一位都是 1，结果才是 1。

3.一个字节包含 8 位二进制数，故应该占 4 字节。

4.分析过程可以知道，s 先被赋值成了 a，又减掉了 c 个 1，所以就是 a－c。

5.比 100 大的最小的以 2 为底数的幂是 128，为 2^7，所以需要 7 次。

6.链表只能顺序查找元素，不能实现随机访问。

7.为了避免重复，我们可以设后边的袋子装的不比前边多，把所有情况列出来即可，共 18 种。

8.下标最大的一个结点是最右下方的结点，下标为 2＊(2＊(2＊1＋1)＋1)＋1＝15。

9.可以通过从大到小枚举每个数，验证得知 100 以内最大素数为 97。

10.可以通过试除的方法算出这两个数的最大公约数是 29。

11.连续跑 5 公里肯定是更好的，所以就在周五到周日 3 天跑 5 公里，周一到周四中选 2 天跑 3 公里即可。

12. 根据鸽巢原理，最平均的分法就是每种花色先有 3 张，此时最后一张无论是什么花色都会导致有 4 张牌花色一致。

13. 只需要考虑前 3 位，后 2 位是根据前边对应过来的，前两位取这 5 个数都可以，中间那一位只能取 0,1,8，所以总共 5*5*3=75 个。

14. 根据后序遍历找当前子树的根，根据中序遍历再把当前子树拆成左右子树，递归建树，然后进行前序遍历。

15. 计算机科学领域的最高奖为图灵奖。

二、阅读程序

1. 分析程序过程可以发现，该程序将所有 i 为 n 的约数的位置的小写字母都转化为了大写字母。

判断题：

(1)因为字符串包含的字符不仅仅可以是字母。

(2)如果 i 的初值为 0 的话，在 n%i==0 和 st[i-1]处都会出错。

(3)n 的约数是有可能大于 sqrt(n)的。

(4)大写字母的 ASCII 值都比小写字母小，因此不会发生任何改变。

选择题：

(5)18 拥有 6 个因数：1,2,3,6,9,18。

(6)若想有 36 个不同，那么这个数字必须至少有 36 个因数。$100000=2^5*5^5$，(5+1)*(5+1)=36。

2. 判断题：

(1)此时一定至少有一个位置被赋值成了非 0 数。

(2)如果 m=1 且输入 1 5，当 i=1 时，++ans 后值为 1，是奇数。

(3)如果 m=1 且输入了 1 1 后 a[1]=b[1]=1>0。

(4)第 15 行是否执行与 x 和 y 的大小关系无关。

反例：

3 2

1 2

1 3

选择题：

(5)每次操作一定会把两个 0 变成非负数，所以答案为 2n-2m。

(6)除了第一次进入 if，之后的进入就会发现因为 x 两两不同，所以 a[x]肯定为 0，但是 b[y]记录的上一次进入时候的 x 的值，就相当于把上次的 a[x]置零，然后再修改，能够发现本质上非零的个数还是没变。

3. 判断题：

(1)有重复的数字是不会对程序有影响的。

(2)每次函数返回值的时候都有 depth*b[mink]，但是因为 b 数组全为 0，所以返回的每一个值都是 0。

选择题：

(3)最坏情况下，每次找到的都恰好是最前边的或者每次找到的都恰好是最后边的，那么总体执行次数为 100+99+…+1=5050，约为 5000 次。

(4)最好情况下,每次都能够二分整个区间,复杂度就会达到 O(n log n)级别,约为 600。

(5)想要值最大,就应该让大数的深度尽可能大,最好的策略就是 a 按照递增顺序,这样每次分割都会只去掉最前面一个数字。故答案为 $1^2+2^2+\cdots+10^2=386$。

(6)每一次都是二分,可以算出从浅到深每层的数量是 1,2,4,8,16,32,37,乘上深度加起来即可。

三、完善程序

1.(1)递归到这一个位置的情况,那就直接赋值成应该的数。

(2)左上部分的左上角坐标就是(x,y)。

(3)右下部分的左上角坐标是(x+step,y+step)。

(4)分析功能可知,后两个参数应该为递归深度以及左上角的数值,依次是 n,0。

(5)分析可知最终矩阵的大小应该为 2^n,也就是(1<<n)。

2.(1)此处实现的功能应该为统计 b[i] 的数量。

(2)此时实现的功能为按 b 排序,记录位置。

(3)此处实现的功能应该为统计 a[i] 的数量。

(4)此处实现的功能应该为按 a 排序放好,记录好每个位置放的是原来的第几个数。

(5)此处实现的功能应该为找到排序以后对应的是原来的第几个数。

2020 CCF 非专业级别软件能力认证第一轮(CSP－J)
入门级 C＋＋语言试题参考答案及解析

【答案】

一、单项选择题(共 15 题,每题 2 分,共计 30 分)

1	2	3	4	5	6	7	8	9	10
B	A	C	B	D	B	B	C	C	D
11	12	13	14	15					
C	D	B	B	D					

二、阅读程序

1.(1)√ (2)× (3)√ (4)× (5)C (6)D

2.(1)× (2)× (3)√ (4)D (5)A (6)D

3.(1)× (2)√ (3)× (4)C (5)B (6)D

三、完善程序

第 1 题					第 2 题				
(1)	(2)	(3)	(4)	(5)	(1)	(2)	(3)	(4)	(5)
D	D	D	A	D	C	C	C	B	B

【解析】

一、单项选择题

1.每个存储单元都会被分配一个地址,我们可以访问物理地址进行读取/写入数据。

2.编译器的作用是把高级语言翻译成计算机能理解的机器语言(0 和 1)。

3.∧ 为“与”——逻辑乘法、∨ 为“或”——逻辑加法。可以把原式中符号替换为 * 和＋,结果不为 0 的只有(x∧y)∨(z∨x)。故选 C。

4.每个像素占 32 位,4 字节。2048×1024 个像素占 $2048\times1024\times32$ bit $=2^{11+10+5}$ bit $=2^{26}$ bit $=2^{23}$ B $=2^{3}$ MB 。

5.这个是进行优化的冒泡排序。不优化的冒泡排序是这样的,我们需要 n－1 趟排序,第 i 趟我们就可以排好第 i 大的位置($1\leqslant i\leqslant n-1$),所以我们每次仅需要排第一个到 n－i＋1 的即可。

然后再来看题目的伪代码。在最好情况下,初始序列已经从小到大排好序了,那么也不会交换,也不会更新 FLAG,直接就结束循环了。

6.翻译成 C++代码如下:

```
double XYZ(int a[],int n)
{
```

```
    if(n==1) return a[1];
    else
    {//该部分代码返回的是 XYZ(a,n-1)和 a[n]之间的最小值,即 a 序列中的最小值
      temp=XYZ(a,n-1);
      if(temp<a[n]) return temp;
      else return a[n];
    }
}
```

7. 链表就是为了解决数组插入删除需要移动元素以及扩展麻烦的这个问题，当然也带来了不可随机访问的问题，它是用指针进行存储的，存储的是下一个的位置。

8. 对于任意一个 n 个顶点的连通图，当且仅当该连通图为树时边数最小，此时边数为 n－1。

9. 用“按权展开求和”$2^3+2^1+2^0=11$。

10. 将两个双胞胎捆绑到一起，进行排列A_4^4，双胞胎两个人也有 2 种站法，$A_4^4\times 2=48$。

11. 先进先出为队列，先进后出为栈。

12. 对于任意一棵深度为 k 的完全二叉树，其结点数 n 一定满足$2^{k-1}-1<n\leqslant 2^k-1$，题目意思可以理解为找出一个最大的深度 k，满足$2^{k-1}-1<61$，通过计算，算出最大的深度为 6。

13. $1949\equiv 9(\mathrm{mod}\ 10)$，$1949\equiv 5(\mathrm{mod}\ 12)$，也就是己丑。

14. 10 个三好学生，中间有 9 个空，每个班至少有 1 个名额，我们只要把板子放进空里就能保证。填上 6 个空就能分为 7 个组合，也就是$C_9^6=84$。

15. 首先我们先从 5 副手套中选 2 副手套，也就是C_5^2，接着我们从剩下的 6 只中选择 4 只C_6^4，减去 3 种剩下两只手套也成对的情况，也就是$C_5^2\times(C_6^4-3)$。

二、阅读程序

1.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
encoder	C	S	P	A	B	D	E	F	G	H	I	J	K	L	M	N	O	Q	R	T	U	V	W	X	Y	Z
与‘A’相差	2	18	15	0	1	3	4	5	6	7	8	9	10	11	12	13	14	16	17	19	20	21	22	23	24	25

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
decoder	D	E	A	F	G	H	I	J	K	L	M	N	O	P	Q	C	R	S	B	T	U	V	W	X	Y	Z
英语字母表	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

判断题：

(1)第 30 行 st[i]＝decoder[st[i]－'A']，如果字符串 st 输入的数中不是大写字母，会导致 st[i]－'A'的值大于或小于 0。

(2)可以发现 st[i]＝decoder[st[i]－'A']，如果 st[]字符串取“ABCD…”的话，是一模一样的。

(3)因为 encoder 数组只有前三个进行了初始化，26 或者 16 并不会影响 k 的值。

(4)第 26 行因要遍历所有的 encoder 数组，产生 decoder 数组，改成 16 的话没办法把 decoder 数组中存入' A '～' Z '。

单选题：

(5)根据英语字母表和 decoder 的对应可以查到若输出 ABCABCABCA，则输入为 CSPCSPCSPC，故既有 S 又有 P。

(6)根据英语字母表和 decoder 的对应可以查到若输出 CSPCSPCSPCSP，则输入为 PRNPRNPRNPRN，故既有 P 又有 R。

2.这题应该是将 n 转化为一个 k 进制，然后每一位用数组 d 存起来，输出按程序过程进行的进位次数。

判断题

(1)当 n=0，k=1 时，模拟得到 len=1。

(2)当 n=1，k=2 时，模拟得到 len=1。

(3)当 n 取得满足条件时的最大值，其每一位都为 k−1。所以 $n=k^{len}-1<k^{len}$。所以对于任意的 n 都满足 $n<k^{len}$。

单选题

(4)将10^{15}转化为一进制，每循环一次 ans 就会增加一次，故答案为10^{15}(忽略在实际情况下由于 d 数组的大小限制会产生 RE 的情况)。

(5)从低位到最高位：第一位 ans 增加3^{29}次、第二位 ans 增加3^{28}次……最高位 ans 增加3^{0}次，根据等比数列求和公式可得答案为 $\sum_{i=0}^{29} 3^i=\frac{3^{30}-1}{2}$。

(6)考虑更为一般的进位，也就是把 $\sum_{i=1}^{15}\left\lfloor\frac{n}{10^i}\right\rfloor$算出来，算出来为 D。

3.题目相当于给出 AB 两个序列，每次搜索 s=A 序列前两数的和+B 序列前两数差的绝对值，然后再把每个数列的两个数合并，再进行下一次搜索。虽然本程序是会求全排列，但是根据题目要求，只要贪心求数列从头到尾依次合并就可以了。

判断题：

(1)n 为 0 程序不会出错，dfs 直接不执行 if、不执行 for 循环退出。

(2)因为数组中所有的数都为 0，根据 s=a+x+abs(b−y)，那么所有的求和操作得到的结果都是 0。

(3)n=1，d[i][0]和 d[i][1]均比较大，ans 反而变小。

单选题：

(4)递归之前，合并两行第 i−1 和 i 个位置的元素，递归之后，若 d[i][0]和 d[i][1]两行的元素相同，就回溯回上一层的数组。题目中要求获得所有结果中的最大值，先模拟 dfs 变量 n 从 20 到 1 的情况，每次 i 都取 1，发现 ans 的值如下(每个小括号代表递归一层、一共 19 层)：$ans=\sum_{i=2}^{20} 9i=\frac{(2+20)\times 19}{2}\times 9=1881$。

(5)这题和第 4 题差不多，根据 s=a+x+|b−y|，a 与 x 的值为 0，与 a 和 x 无关，同样 dfs 一次到 n=1(共 29 层)，发现 ans 的值为 28×5，然后模拟回溯一次(仅回到上一层)，发现结果变小了。推理得到这个 dfs 能取得的最大值为

$$ans=\sum_{i=0}^{28} i\times 5=\frac{(1+28)\times 28}{2}\times 5=2030。$$

然后模拟回溯一次(仅回到上一层)，发现结果变小了。推理得到这个 dfs 能取得的最大值为 2030。

(6)

初始值	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
d[i][0]	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
d[i][1]	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
第 1 层递归 i=1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
d[i][0]	29	13	12	11	10	9	8	7	6	5	4	3	2	1	1
d[i][1]	29	13	12	11	10	9	8	7	6	5	4	3	2	1	1
第 2 层递归 i=1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
d[i][0]	42	12	11	10	9	8	7	6	5	4	3	2	1	1	1
d[i][1]	42	12	11	10	9	8	7	6	5	4	3	2	1	1	1

需要注意的是 d [i] [0] 和 d [i] [1] 中的数字不同，公差为 1。同样先 dfs 一次到 n=1，然后再去找规律。第一次深搜到达叶子结点时 ans 的值如下式，每个()代表递归一层，对答案的贡献：

$$ans=\sum_{i=1}^{14}\left(\sum_{j=1}^{15}j+\left|\sum_{j=i+1}^{15}(j-i)\right|\right)=2240。$$

三、完善程序

1. 质因数分解是一个常见的题目

(1)因为 2 是最小的素数，所以从 2 开始。如果从 1 开始直接死循环。

(2)枚举到 $\sqrt{n}$ 即可，题目中满足条件的只有 D 选项。

(3)要把 i 这个质因子全部除光。

(4)如果 n>1，那就说明还存在 n 这个大素数，没有除干净。

(5)因为 n 是质数，所以输出 n 即可。

2. 先将 n 个区间按照起点顺序从小到大排序，从排好序的第一个区间开始，令 r 表示已经覆盖到的区域 [0,r]，在剩下的区间中找出所有左端点小于等于当前已经覆盖区域的右端点，并且右端点大于等于之前覆盖区域的右端点 r，取右端点最大的区间加入已覆盖区域并赋值给 r。直到 r=m 结束。

(1)排序肯定先对左端点排序，再去调换位置。

(2)交换变量的基本操作。

(3)线段如果不包含了就应该加上，也就是当前的右端点比最后一个右端点大，第一个肯定在，如果包含那就没必要选择了。

(4)根据题解方法，找到左端点小于等于当前已覆盖区域的右端点，并且右端点大于等于当前已覆盖区域的右端点的区间，然后更新已覆盖区间 [0,r] 的右端点。

(5)r 会被更新为当前的右端点。

2021 CCF 非专业级别软件能力认证第一轮(CSP－J)入门级 C＋＋语言试题参考答案及解析

【答案】

一、单项选择题

1	2	3	4	5	6	7	8	9	10
D	B	A	C	D	D	C	A	B	B
11	12	13	14	15					
B	A	C	B	B					

二、阅读程序

1.(1)×　(2)×　(3)×　(4)√　(5)×　(6)B

2.(1)×　(2)√　(3)√　(4)B　(5)B　(6)C

3.(1)√　(2)×　(3)×　(4)A　(5)C　(6)C

三、完善程序

第 1 题					第 2 题				
(1)	(2)	(3)	(4)	(5)	(1)	(2)	(3)	(4)	(5)
D	C	C	D	B	B	D	C	B	D

【解析】

一、单项选择题

1.C 语言是面向过程的程序设计语言。

2.图灵奖是计算机领域的国际最高奖项。

3.计算机中以二进制方式进行储存数据。

4.以第一个数为初始值，从第二个数开始比较，最坏情况下要比较到最后一个数才能得到最大值，即比较 n－1 次。

5.由于 c 先出栈，a，b 均需先入栈。a 在 b 入栈后不可能比 b 先出栈。

6.n 个结点的树有 n－1 条边，因此需要删除 m－(n－1)条边，即 m－n＋1 条。

7.$(101.11)_2=1*2^2+0*2^1+1*2^0+1*2^{-1}+1*2^{-2}=(5.75)_{10}$

8.高度为 5 的完全二叉树第 5 层最多有 $2^4=16$ 个结点，因此从左到右依次选择若干个连续结点的方案数为 16。

9.先算 b＋c，因此 abc＋d，

再算 a＊(b＋c)，因此 abc＋＊d，

最后算 a＊(b＋c)＊d，因此 abc＋＊d＊。

10.区分队伍编号情况下方案数为 $C_6^2\ C_4^2\ C_2^2=90$，在不区分队伍编号的情况下每种情况都会重复 $A_3^3=6$ 次，因此最终答案为 $\frac{C_6^2\ C_4^2\ C_2^2}{A_3^3}=15$。

11.哈夫曼编码每次把频率最低的两个结点(字符)合并产生新的结点，将新的结点放到集合

中，删除用来合并的结点，重复上述过程直到剩下一个结点为止。每次选择频率最小的两个结点就是贪心的思想。

12. 三个数都不重复时，方案数为$A_3^3=6$；

有两个数重复时，方案数为 3 * 2 * 2＝12；

总方案数为 18。

13. solve(7)＝7 * 5 * 3 * 2 * 1＝210。

14. 从 a 向 b 的方向开始搜索，路径只能为 a→b→d→c→e，终点为 e。从 a 向 c 的方向开始搜索，路径可为 a→c→d→b→e 或 a→c→e→d→b，终点为 b 和 e。因此搜索终点只可能是 b，e。

15. 贪心算法，为节约时间，令 4，8 过河，1，2 驶船从 B 回到 A。故

1，2 过河，1 返回，花费时间 3；

4，8 过河，2 返回，花费时间 10；

1，2 过河，花费时间 2；

故总用时为 15。

二、阅读程序

1. 判断题：

(1)a 数组定义长度为 1000，因此下标范围为 0～999。

(2)f 函数的作用是统计传入参数在二进制补码表示下 1 的个数，对负数依然适用。

(3)g 函数的作用是提取出传入参数在二进制补码表示下最低位的 1，因此 f(10)＋g(10)＝2＋2＝4。

(4) 511998 的二进制补码表示为 1111100111111111110，因此 f(511998)＝16，g(511998)＝2。

(5)主函数前 g 函数缺少函数声明，无法在主函数中使用，会报错。

单选题：

(6)非负数的补码表示与其二进制表示相同，负数的补码表示为其绝对值的二进制表示取反并加 1。因为 $65536=2^{16}$，所以－65536 的补码表示为 11111111111111110000000000000000，由此可知 f(－65536)＝16，g(－65536)＝65536；因为 $2147483647=2^{31}-1$，所以 214748347 的补码表示为 01111111111111111111111111111111，因此 f(2147483647)＝31，g(2147483647)＝1。

2. 判断题：

(1)Base64 是网络上最常见的用于传输 8Bit 字节码的编码方式之一，它的解码规则是把 4 个字节变成 3 个字节。解密得到的是任意的，没有限定。

(2)字符串后的等号不会被计算进去，因此 a＝与 a==输出的第二行明显相同。

(3)table[0]初始值为 0xff(十进制下的 255)，在 char 中表示－1(char 只能表示－128 到 127，因此 0xff 对应－1)，强制转换为 int 后值为－1。

单选题：

(4)decode 函数只有一个循环，所以时间复杂度显然为 O(n)。

(5)手动模拟即可。

①通过查 table 表 Y3Nx 得到序号(十进制)24，55，13，49；

②将这些十进制数转成 6 位二进制 011000，110111，001101，110001；

③将二进制 8 位一组从新组合：01100011，01110011，01110001。

④根据 ASCII 码表，得到字符 csq。

(6)排除法,首先确定位数为 8,其次确定最后两个字符不同。

3.判断题:

(1)如果 x 不等于 1,f[1],g[1]也不会被调用。

(2)对于一个大于 1 的正整数 n 可以分解质因数:

$n=\prod_{i=1}^{k} p_i^{a_i}=p_1^{a1}\cdot p_2^{a2}\cdot\cdots\cdot p_k^{ak}$,则 n 的正约数的个数就是 $f(n)=\prod_{i=1}^{k}(a_i+1)=(a_1+1)(a_2+1)\cdots(a_k+1)$。其中 $a_1,a_2,a_3,\cdots,a_k$ 是 $p_1,p_2,p_3,\cdots,p_k$ 的指数。

f[i]为 i 的约数个数,c[x]为 x 的最小质因子的次方数,对应上面公式中的 a_i,用来辅助求 f[i]。当一个合数 i * k 被筛的时候,该合数的最小质因子 k 是第一次出现,程序执行第 31 行和第 32 行;如果合数 i * k 中已经有最小质因子 k 了,根据上面的 $f(n)=\prod_{i=1}^{k}(a_i+1)=(a_1+1)(a_2+1)\cdots(a_k+1)$,F[i * k]=f[i]/(c[i]+1) * (c[i * k]+1),程序执行第 24 行和第 25 行。

(3)对于一个大于 1 的正整数 n,可以分解质因数:$n=\prod_{i=1}^{k} p_i^{ai}=p_1^{a1}\cdot p_2^{a2}\cdot\cdots\cdot p_k^{ak}$,则 n 的正约数的和为 $f(n)=\prod_{i=1}^{k}(\sum_{j=0}^{ai} p_i^j)=(1+p_1+p_1^2+\cdots+p_1^{a1})(1+p_2+p_2^2+\cdots+p_2^{a2})\cdots(1+p_k+p_k^2+\cdots+p_k^{ak})$。

g[i]为 i 的约数和,对应上面公式中的 f(n);

d[i * k]里面存的是上面公式没有质因子 k 的其余各项乘积,$(1+p_2+p_2^2+\cdots+p_2^{a2})\cdots(1+p_k+p_k^2+\cdots+p_k^{ak})$用来辅助求 g[i],g[i]不是单调递增的。举个例子:$g[8]=(1+2^1+2^2+2^3)=1+2+4+8=15$,$g[9]=(1+3^1+3^2)=1+3+9=13$。

单选题:

(4) a[i]里面存着 0 表示 i 是素数,存着 1 表示 i 是合数;b[i]里面存着从小到大被筛出来的素数,去除第 24 至 27、30 至 35 行代码,就是素数线性筛算法,因为每个数只会被其最小的质因子筛到一次,所以复杂度为 O(n)。

(5) f[i]为 i 的约数个数,约数个数为 2 的显然是质数,而 100 以内的素数有 25 个。

(6) 根据 $f[i]=\prod_{p}(c+1)$,$g[i]=\prod_{p}\sum_{i=0}^{c} p^c$,可以直接手算出答案。

三、完善程序

1.(1)c 记录离开的人数,c=n−1 表示圈中已离开 n−1 人,仅剩一人,即为结束条件。

(2)p 记录当前报的数,报 1 离开。

(3)当前的人报 1 离开,离开的人数+1。

(4)更新报数状态,0 和 1 交替报。

(5)i 表示当前报数的人,需要从 0～(n−1)依次循环报数。

2.(1)这个选项应该在下面题目都完成后,才能确定,请同学们先思考下面的题目。程序接下来要根据坐标进行二分查找,因此排序函数肯定要与坐标相关。比较有迷惑性的是 D,不能选 D 的原因在于,根据第 43 行代码,二分查找只要坐标比较就可以了,不用存坐标的序号再进行比较。举个极端的例子,假设选项 D 正确,并且最终需要查找的是 A[0],当查找范围是 a=0,b=1;只有两个需要查找的数时,根据第 39 行代码得知 p.id=n,肯定大于 A[mid].id,那么函数 cmp(A[mid], p)会返回 true,使得 mid=a+1;结果导致找不到答案了,与假设矛盾,综上所述,D 是不对的。

(2)该部分为去重操作。

(3)该部分为二分查找。首先 A 选项是有问题的,不能取到中间的值;其他三个选项对于从 a 到 b 有奇数个数的情况,取值是一样的。但是如果是从 a 到 b 有偶数个数的情况,C 选项的 mid 偏向左,而 B、D 选项偏向右,根据 48 行代码可以得到结论,应该选择 C。举个极端的例子,假设最终需要查找的是 A[0],查找范围是 a＝0,b＝1;如果用 B、D 选项偏向右,取偏大值,那么永远找不到答案。

(4)点是按升序排的,p 比 A[mid]大时才会更新左边界。比较有迷惑性的是 D 选项,它和 B 选项的区别是 p 和 A[mid]是否相等。将 44 行代码和 48 行代码结合起来看,根据第 48 行代码可知,a 是最终有可能成功答案的序号。假设最终需要查找的是 A[0],并且假设 D 选项正确,那么当 p 和 A[mid]相等就能去执行第 44 行代码 a＝mid＋1,导致 a 就肯定取不到最终答案 0 了,和第一个假设相互矛盾。综上所述,D 是不对的。

(5)代码枚举一个矩先通过枚举矩形的左下角与右上角的坐标,根据枚举成功的左下角与右上角,构造出左上角和右下角的坐标,最后通过二分查找确定新构造出的左上角和右下角是否存在。

2022 CCF 非专业级别软件能力认证第一轮(CSP－J)
入门级 C＋＋语言试题参考答案及解析

【答案】

一、单项选择题(共 15 题,每题 2 分,共计 30 分)

1	2	3	4	5	6	7	8	9	10
A	C	D	C	B	B	B	C	B	D
11	12	13	14	15					
D	B	C	B	B					

二、阅读程序(共 3 题,除特殊说明外,判断题每题 1.5 分,选择题每题 3 分,共计 40 分)

(一) **16.**√ **17.**× **18.**× **19.**× **20.**× **21.**B

(二) **22.**× **23.**√ **24.**√ **25.**C **26.**C **27.**B

(三) **28.**√ **29.**√ **30.**× **31.**× **32.**C **33.**B **34.**A

三、完善程序(共 10 题,每题 3 分,共计 30 分)

(一)					(二)				
35	36	37	38	39	40	41	42	43	44
A	B	C	D	A	A	B	C	D	A

【解析】

一、单项选择题

1.printf 是从 C 语言开始就有的用于输出的函数,没有涉及 C++的面向对象特性。

2.模拟入栈出栈过程,C 选项中 6 比 5 先出栈非法。

3.第三行将 p 指向 x 的地址,第四行将 q 指向 y 的地址,第五行将 p 指向的地址修改为 q 指向的地址(即 y 的地址)。

4.选项 A:错误,数组可以进行排序;选项 B:错误,二者能存储的信息均取决于其长度;选项 C:正确,数组在声明时需要指定大小,链表可以动态插入或删除节点;选项 D:错误。

5.栈的特点是后进先出,队列的特点是先进先出,所以出队列顺序＝入队列顺序＝出栈顺序。依题意模拟过程,得栈的容量至少为 3。

6.建出表达式树,先序遍历即为前缀表达式。

7.建出哈夫曼树,d 的编码为 00,长度为 2 位。

8.节点 i 的父亲节点为$\left\lfloor \frac{i}{2} \right\rfloor$,左儿子为 2·i,右儿子为 2·i+1。所以节点 9 的父亲节点为节点 4,兄弟节点为节点 8,右儿子为 19。

9.非 0 元素最少即边数最少,边数最少的有向连通图为环,边数为 N。

10.栈和队列本质上都是功能受限的线性表,本质相同,可以用两个栈实现入队和出队,选项 D 错误。

11.选项 A:错误,`s->next=p->next` 时 `p->next` 已经被修改为 s;选项 B:错误,`s->`

next=p->next 时 p->next 已经被修改为 s；选项 C：错误，p->next->prev=s 时 p->next已经被修改为 s。

12. 堆排序、快速排序、希尔排序、选择排序不是稳定的排序算法；基数排序、冒泡排序、插入排序、归并排序是稳定的排序算法。

13. 按位权展开即可。

14. 长度为 0 的子串：空串；长度为 1 的子串：a，b，c；长度为 2 的子串：ab，bc，ca；长度为 3 的子串：abc，bca，cab；长度为 4 的子串：abca，bcab；长度为 5 的子串：abcab。共 13 个内容不同的子串。

15. 递归是通过调用自身来求解问题的编程技术，B 选项正确。

二、阅读程序

(一)

判断题：

16. unsigned short 16 位都是数值位；short 1 位是符号位，15 位是数值位，由程序可得位数最多为 8 位，short 类型足够存放运算结果。

17. 改为 char 以后，当输入为两位数时，x，y 分别读入的是第一个数的十位和个位，改变了程序的行为和结果。

18. 当输入为“2 2”时，输出为“12”。

19. 当输入为“2 2”时，输出为“12”。

20. 当输入为“2 2”时，输出为“12”。

单选题：

21. 当输入为“13 8”时，输出为“209”。

(二)

判断题：

22. 递推可得，当输入为“7 3”时，第 19 行用来取最小值的 min 函数执行了 448 次。

23. 函数 f，g 分别用递归和递推解决了同样的问题。

24. 正确。递归函数 f 中，若 m 为 1，返回值为 n；递推函数 g 中，若 j 为 1，h[i][j]为 i，则若 m 为 1，h[n][m]就为 n。

单选题：

25. g 函数最内层循环一共会执行$\frac{n(n+1)(m-1)}{2}$次，时间复杂度为 $O(n^2m)$。

26. 递推可得，当输入为“20 2”时，输出的第一行为“6”。

27. 找规律可得，当输入为“100 100”时，输出的第一行为“7”。

(三)

判断题：

28. solve1 函数是二分函数，二分的范围为 n，复杂度为 O(log n)，solve2 函数里面只有 k 次循环，复杂度为 O(k)，所以整体复杂度为 O(log n+k)。

29. solve1 函数的返回值为$\lfloor\sqrt{n}\rfloor$，当输入为“9801 1”时，solve1 返回值为 99，把 99 代入 solve2 函数，返回值为 99，即输出的第一个数为“99”。

30. 若 n 的算术平方根为无理数，输出的第二个数为“0”。

31. 当 $n\geqslant 4$ 时，$\sqrt{n}\leqslant\left\lfloor\frac{n}{2}\right\rfloor$，所以在二分过程中，mid 不会大于$\left\lfloor\frac{n}{2}\right\rfloor$，所以 mid · mid 不会超过

23500×23500=552250000，不会溢出。

单选题：

32. solve1 函数的返回值为$\lfloor\sqrt{n}\rfloor$，当输入为“2 1”时，solve1 返回值为 1，把 1 代入 solve2 函数，返回值为 1.5，即输出的第一个数为“1.5”。

33. solve1 函数的返回值为$\lfloor\sqrt{n}\rfloor$，当输入为“3 10”时，solve1 返回值为 1，把 1 代入 solve2 函数，模拟前几次循环，即可得出答案为“1.732”。

34. solve1 函数的返回值为$\lfloor\sqrt{n}\rfloor$，当输入为“256 11”时，solve1 返回值为 16，因为$\sqrt{256}=16$，所以无论 solve2 中进行多少次循环，答案都不会改变，输出的第一个数等于 16。

三、完善程序

（一）

35. fac 里面从小到大存储 n 小于$\sqrt{n}$的正因数，所以选 A。

36. 从小到大输出 n 小于等于$\sqrt{n}$的正因数，所以选 B。

37. 判断是否有 $i^2=n$，有则输出 i，所以选 C。

38. 判断是否有 $i^2=n$，有则输出 i，所以选 D。

39. 从小到大输出 n 大于$\sqrt{n}$的正因数，所以选 A。

（二）

40. 判断此位置为旧颜色 prev_color 时，需要更新颜色，所以选 A。

41. 将起点位置更新颜色，所以选 B。

42. 上下左右四个相邻点，可以发现少了下方点 Point(pt.r+1, pt.c)，所以选 C。

43. 将当前位置更新颜色，所以选 D。

44. 将当前位置加入队列，所以选 A。

第七章　2019—2022 年 CSP－S 提高级初赛真题及答案解析

2019 CCF 非专业级别软件能力认证第一轮(CSP－S)提高级 C＋＋语言试题

一、单项选择题(共 15 题,每题 2 分,共计 30 分;每题有且仅有一个正确选项)

1.若有定义:int a＝7;float x＝2.5, y＝4.7;,则表达式 x＋a%3 * (int)(x＋y)%2 的值是(　　)。

A. 0.000000　　B. 2.750000　　C. 2.500000　　D. 3.500000

2.下列属于图像文件格式的有(　　)。

A. WMV　　B. MPEG　　C. JPEG　　D. AVI

3.二进制数 11 1011 1001 0111 和 01 0110 1110 1011 进行逻辑或运算的结果是(　　)。

A. 11 1111 1101 1111　　B. 11 1111 1111 1101

C. 10 1111 1111 1111　　D. 11 1111 1111 1111

4.编译器的功能是(　　)。

A. 将源程序重新组合

B. 将一种语言(通常是高级语言)翻译成另一种语言(通常是低级语言)

C. 将低级语言翻译成高级语言

D. 将一种编程语言翻译成自然语言

5.设变量 x 为 float 型且已赋值,则以下语句中能将 x 中的数值保留到小数点后两位,并将第三位四舍五入的是(　　)。

A. x＝(x * 100＋0.5)/100.0;　　B. x＝(int)(x * 100＋0.5)/100.0;

C. x＝(x/100＋0.5) * 100.0;　　D. x＝x * 100＋0.5/100.0;

6.由数字 1, 1, 2, 4, 8, 8 所组成的不同的 4 位数的个数是(　　)。

A. 104　　B. 102　　C. 98　　D. 100

7.排序的算法很多,若按排序的稳定性和不稳定性分类,则(　　)是不稳定排序。

A. 冒泡排序　　B. 直接插入排序　　C. 快速排序　　D. 归并排序

8.G 是一个非连通无向图(没有重边和自环),共有 28 条边,则该图至少有(　　)个顶点。

A. 10　　B. 9　　C. 11　　D. 8

9.一些数字可以颠倒过来看,例如 0、1、8 颠倒过来还是本身,6 颠倒过来是 9,9 颠倒过来看是 6,其他数字颠倒过来都不构成数字。类似的,一些多位数也可以颠倒过来看,比如 106 颠倒过来是 901。假设某个城市的车牌只有 5 位数字,每一位都可以取 0 到 9。请问这个城市有(　　)个车牌倒过来恰好还是原来的车牌,并且车牌上的 5 位数能被 3 整除。

A. 40　　B. 25　　C. 30　　D. 20

10.一次期末考试,某班有 15 人数学得满分,有 12 人语文得满分,并且有 4 人语、数都是满分,那么这个班至少有一门得满分的同学有(　　)人。

A. 23　　B. 21　　C. 20　　D. 22

11. 设 A 和 B 是两个长为 n 的有序数组，现在需要将 A 和 B 合并成一个排好序的数组，请问任何以元素比较作为基本运算的归并算法，在最坏情况下至少要做(　　)次比较。

A. n^2　　B. n log n　　C. 2n　　D. 2n−1

12. 以下哪个结构可以用来存储图(　　)。

A. 栈　　B. 二叉树　　C. 队列　　D. 邻接矩阵

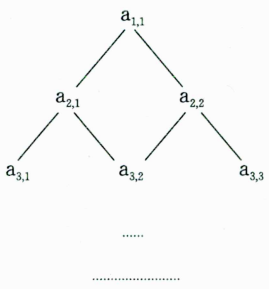
13. 以下(　　)算法不属于贪心算法。

A. Dijkstra 算法　　B. Floyd 算法　　C. Prim 算法　　D. Kruskal 算法

14. 有一个等比数列，共有奇数项，其中第一项和最后一项分别是 2 和 118098，中间一项是 486，公比可能为(　　)。

A. 5　　B. 3　　C. 4　　D. 2

15. 有正实数构成的数字三角形排列形式如图所示。第一行的数为 $a_{1,1}$，第二行的数从左到右依次为 $a_{2,1}$，$a_{2,2}$，第 n 行的数为 $a_{n,1}$，$a_{n,2}$，…，$a_{n,n}$。从 $a_{1,1}$ 开始，每一行的数 $a_{i,j}$ 只有两条边可以分别通向下一行的两个数 $a_{i+1,j}$ 和 $a_{i+1,j+1}$。用动态规划算法找出一条从 $a_{1,1}$ 向下通到 $a_{n,1}$，$a_{n,2}$，…，$a_{n,n}$ 中某个数的路径，使得该路径上的数之和最大。



令 C[i][j]是从 $a_{1,1}$ 到 $a_{i,j}$ 的路径上的数的最大和，并且 C[i][0]=C[0][j]=0，则 C[i][j]=(　　)。

A. max{C[i−1][j−1],C[i−1][j]}+$a_{i,j}$　　B. C[i−1][j−1]+C[i−1][j]

C. max{C[i−1][j−1],C[i−1][j]}+1　　D. max{C[i][j−1],C[i−1][j]}+$a_{i,j}$

二、阅读程序（程序输入不超过数组或字符串定义的范围；判断题正确填√，错误填×；除特殊说明外，判断题每题 1.5 分，选择题每题 4 分，共计 40 分）

1.

```
01 #include<cstdio>
02 using namespace std;
03 int n;
04 int a[100];
05
06 int main(){
07   scanf("%d",&n);
08   for (int i=1;i<=n;++i)
09     scanf("%d",&a[i]);
10   int ans=1;
11   for (int i=1;i<=n;++i){
12     if (i>1 && a[i]<a[i-1])
```

```
13      ans=i;
14    while (ans<n && a[i]>=a[ans+1])
15      ++ans;
16    printf("%d\n",ans);
17  }
18  return 0;
19 }
```

判断题

(1)(1 分)第 16 行输出 ans 时，ans 的值一定大于 i。(　　)

(2)(1 分)程序输出的 ans 小于等于 n。(　　)

(3)若将第 12 行的“<”改为“!=”，程序输出的结果不会改变。(　　)

(4)当程序执行到第 16 行时，若 ans－i>2，则 a[i＋1]<=a[i]。(　　)

选择题

(5)(3 分)若输入的 a 数组是一个严格单调递增的数列，此程序的时间复杂度是(　　)。

A. O(log n)　　B. $O(n^2)$　　C. O(n log n)　　D. O(n)

(6)最坏情况，此程序的时间复杂度是(　　)。

A. $O(n^2)$　　B. O(log n)　　C. O(n)　　D. O(n log n)

2.

```
01 #include<iostream>
02 using namespace std;
03
04 const int maxn=1000;
05 int n;
06 int fa[maxn],cnt[maxn];
07
08 int getRoot(int v){
09    if (fa[v]==v) return v;
10    return getRoot(fa[v]);
11  }
12
13 int main(){
14  cin>>n;
15  for (int i=0;i<n;++i){
16      fa[i]=i;
17      cnt[i]=1;
18  }
19  int ans=0;
20  for (int i=0;i<n-1;++i){
21      int a,b,x,y;
22      cin>>a>>b;
23      x=getRoot(a);
24      y=getRoot(b);
25      ans+=cnt[x]*cnt[y];
26      fa[x]=y;
27      cnt[y]+=cnt[x];
28  }
```

```
29   cout<<ans<<endl;
30   return 0;
31 }
```

●判断题

(1)(1分)输入的a和b值应在[0，n－1]的范围内。(　　)

(2)(1分)第16行改成fa[i]=0;,不影响程序运行结果。(　　)

(3)若输入的a和b的值均在[0，n－1]范围内,则对于任意0≤i<n,都有0≤fa[i]<n。(　　)

(4)若输入的a和b的值均在[0，n－1]范围内,则对于任意0≤i<n,都有1≤cnt[i]≤n。(　　)

●选择题

(5)当n等于50时,若a、b的值都在[0,49]的范围内,且在第25行时x总是不等于y,那么输出为(　　)。

A. 1276　　B. 1176　　C. 1225　　D. 1250

(6)此程序的时间复杂度是(　　)。

A. O(n)　　B. O(log n)　　C. $O(n^2)$　　D. O(n log n)

3.本题t是s的子序列的意思是:从s中删去若干个字符,可以得到t;特别的,如果s=t,那么t也是s的子序列;空串是任何串的子序列。例如“acd”是“abcde”的子序列,“acd”是“acd”的子序列,但“adc”不是“abcde”的子序列。

s[x..y]表示s[x]...s[y]共y－x＋1个字符构成的字符串,若x>y,则s[x..y]是空串。t[x..y]同理。

```
01 #include<iostream>
02 #include<string>
03 using namespace std;
04 const int max1=202;
05 string s,t;
06 int pre[max1],suf[max1];
07
08 int main(){
09   cin>>s>>t;
10   int slen=s.length(),tlen=t.length();
11   for (int i=0,j=0;i<slen;++i){
12     if (j<tlen && s[i]==t[j])++j;
13     pre[i]=j;// t[0..j-1]是s[0..i]的子序列
14   }
15   for (int i=slen-1,j=tlen- 1;i>=0;--i){
16     if (j>=0 && s[i]==t[j])--j;
17     suf[i]=j;// t[j+1..tlen-1]是s[i..slen-1]的子序列
18   }
19   suf[slen]=tlen-1;
20   int ans=0;
21   for(int i=0,j=0,tmp=0;i<=slen;++i){
22     while ( j<=slen && tmp>=suf[j]+1)++j;
```

```
23      ans=max(ans,j-i-1);
24      tmp=pre[i];
25    }
26    cout<<ans<<endl;
27    return 0;
28 }
```

提示：

t[0..pre[i]－1]是 s[0..i]的子序列；

t[suf[i]＋1..tlen－1]是 s[i..slen－1]的子序列。

●判断题

(1)(1 分)程序输出时，suf 数组满足：对任意 0≤i＜slen，suf[i]≤suf[i＋1]。(　　)

(2)(2 分)当 t 是 s 的子序列时，输出一定不为 0。(　　)

(3)(2 分)程序运行到第 23 行时，j－i－1 一定不小于 0。(　　)

(4)(2 分)当 t 是 s 的子序列时，pre 数组和 suf 数组满足：对任意 0≤i＜slen，pre[i]＞suf[i＋1]＋1。(　　)

●选择题

(5)若 tlen＝10，输出为 0，则 slen 最小为(　　)。

A. 10　　B. 12　　C. 0　　D. 1

(6)若 tlen＝10，输出为 2，则 slen 最小为(　　)。

A. 0　　B. 10　　C. 12　　D. 1

三、完善程序(单选题，每小题 3 分，共计 30 分)

1.(匠人的自我修养)一个匠人决定要学习 n 个新技术。要想成功学习一个新技术，他不仅要拥有一定的经验值，而且还必须要先学会若干个相关的技术。学会一个新技术之后，他的经验值会增加一个对应的值。给定每个技术的学习条件和习得后获得的经验值，给定他已有的经验值，请问他最多能学会多少个新技术。

输入第一行有两个数，分别为新技术个数 n($1\leqslant n\leqslant 10^3$)，以及已有经验值($\leqslant 10^7$)。

接下来 n 行。第 i 行的两个正整数，分别表示学习第 i 个技术所需的最低经验值($\leqslant 10^7$)，以及学会第 i 个技术后可获得的经验值($\leqslant 10^4$)。

接下来 n 行。第 i 行的第一个数 m_i($0\leqslant m_i<n$)，表示第 i 个技术的相关技术数量。紧跟着 m 个两两不同的数，表示第 i 个技术的相关技术编号。输出最多能学会的新技术个数。

下面的程序以 $O(n^2)$的时间复杂度完成这个问题，试补全程序。

```
01 #include<cstdio>
02 using namespace std;
03 const int maxn=1001;
04
05 int n;
06 int cnt[maxn];
```

```
int child[maxn][maxn];
int unlock[maxn];
int points;
int threshold[maxn],bonus[maxn];

bool find(){
  int target=-1;
  for (int i=1;i<=n;++i)
    if( ① && ② ){
      target=i;
      break;
    }
  if(target==-1)
    return false;
  unlock[target]=-1;
  ③ ;
  for (int i=0;i<cnt[target];++i)
    ④ ;
  return true;
}

int main(){
  scanf("%d%d",&n,&points);
  for (int i=1;i<=n;++i){
    cnt[i]=0;
    scanf("%d%d",&threshold[i],&bonus[i]);
  }
  for (int i=1;i<=n;++i){
    int m;
    scanf("%d",&m);
    ⑤ ;
    for (int j=0;j<m;++j){
      int fa;
      scanf("%d",&fa);
      child[fa][cnt[fa]]=i;
      ++cnt[fa];
    }
  }
  int ans=0;
  while(find())
    ++ans;
  printf("%d\n",ans);
```

```
49   return 0;
50 }
```

(1)①处应填(　　)。

A. unlock[i]<=0　　B. unlock[i]>=0

C. unlock[i]==0　　D. unlock[i]==-1

(2)②处应填(　　)。

A. threshold[i]>points　　B. threshold[i]>=points

C. points>threshold[i]　　D. points>=threshold[i]

(3)③处应填(　　)。

A. target=-1　　B.--cnt[target]

C. bonus[target]=0　　D. points+=bonus[target]

(4)④处应填(　　)。

A. cnt[child[target][i]]-=1　　B. cnt[child[target][i]]=0

C. unlock[child[target][i]]-=1　　D. unlock[child[target][i]]=0

(5)⑤处应填(　　)。

A. unlock[i]=cnt[i]　　B. unlock[i]=m

C. unlock[i]=0　　D. unlock[i]=-1

2.(取石子)Alice 和 Bob 两个人在玩取石子游戏。他们制定了 n 条取石子的规则,第 i 条规则为:如果剩余的石子个数大于等于 a[i]且大于等于 b[i],那么他们可以取走 b[i]个石子。他们轮流取石子。如果轮到某个人取石子,而他无法按照任何规则取走石子,那么他就输了。一开始石子有 m 个。请问先取石子的人是否有必胜的方法?

输入第一行有两个正整数,分别为规则个数 n($1\leqslant n\leqslant 64$),以及石子个数 m($\leqslant 10^7$)。

接下来 n 行。第 i 行有两个正整数 a[i]和 b[i]($1\leqslant a[i]\leqslant 10^7$,$1\leqslant b[i]\leqslant 64$)。

如果先取石子的人必胜,那么输出 Win,否则输出 Loss。

提示:

可以使用动态规划解决这个问题。由于 b[i]不超过 64,所以可以使用 64 位无符号整数去压缩必要的状态。

status 是胜负状态的二进制压缩,trans 是状态转移的二进制压缩。

试补全程序。

代码说明:

“～”表示二进制补码运算符,它将每个二进制位的 0 变成 1,1 变为 0;

而“∧”表示二进制异或运算符,它将两个参与运算的数中每个对应的二进制位一一进行比较,若两个二进制位相同,则运算结果的对应二进制位为 0,反之为 1。

ull 标识符表示它前面的数字是 unsigned long long 类型。

```
01 #include<cstdio>
02 #include<algorithm>
03 using namespace std;
04
05 const int maxn=64;
06
07 int n,m;
08 int a[maxn],b[maxn];
```

```
09 unsigned long long status,trans;
10 bool win;
11
12 int main(){
13   scanf("%d%d",&n,&m);
14   for (int i=0;i<n;++i)
15     scanf("%d%d",&a[i],&b[i]);
16   for(int i=0;i<n;++i)
17     for(int j=i+1;j<n;++j)
18       if (a[i]>a[j]){
19         swap(a[i],a[j]);
20         swap(b[i],b[j]);
21       }
22   status= ① ;
23   trans=0;
24   for(int i=1,j=0;i<=m;++i){
25     while (j<n && ② ){
26       ③ ;
27       ++j;
28     }
29     win= ④ ;
30     ⑤ ;
31   }
32   puts(win ? "Win" : "Loss");
33   return 0;
34 }
```

(1)①处应填(　　)。

A. 0　　B. ~0ull　　C. ~0ull^1　　D. 1

(2)②处应填(　　)。

A. a[j]<i　　B. a[j]==i　　C. a[j] !=i　　D. a[j]>i

(3)③处应填(　　)。

A. trans |=1ull<<(b[j]-1)　　B. status |=1ull<<(b[j]-1)

C. status+=1ull<<(b[j]-1)　　D. trans+=1ull<<(b[j]-1)

(4)④处应填(　　)。

A. ~status| trans　　B. status & trans

C. status | trans　　D. ~status & trans

(5)⑤处应填(　　)。

A. trans=status | trans^win　　B. status=trans>>1^win

C. trans=status^trans | win　　D. status=status<<1^win

2020 CCF 非专业级别软件能力认证第一轮(CSP－S)提高级 C＋＋语言试题

一、单项选择题(共 15 题，每题 2 分，共计 30 分；每题有且仅有一个正确选项)

1.请选出以下最大的数(　　)。

A. $(550)_{10}$　　B. $(777)_{8}$　　C. 2^{10}　　D. $(22F)_{16}$

2.操作系统的功能是(　　)。

A. 负责外设与主机之间的信息交换

B. 控制和管理计算机系统的各种硬件和软件资源的使用

C. 负责诊断机器的故障

D. 将源程序编译成目标程序

3.现有一段 8 分钟的视频文件，它的播放速度是每秒 24 帧图像，每帧图像是一幅分辨率为 2048×1024 像素的 32 位真彩色图像。要存储这段原始无压缩视频，需要(　　)的存储空间。

A. 30G　　B. 90G　　C. 150G　　D. 450G

4.今有一空栈 S，对下列待进栈的数据元素序列 a，b，c，d，e，f 依次进行：进栈，进栈，出栈，进栈，进栈，出栈的操作，则此操作完成后，栈底元素为(　　)。

A. b　　B. a　　C. d　　D. c

5.将(2，7，10，18)分别存储到某个地址区间为 0～10 的哈希表中，如果哈希函数 h(x)＝(　　)，将不会产生冲突，其中 a mod b 表示 a 除以 b 的余数。

A. x^2 mod 11

B. 2x mod 11

C. x mod 11

D. [x/2] mod 11，其中 [x/2] 表示 x/2 下取整

6.(　　)不能用贪心法精确求解。

A. 霍夫曼编码问题　　B. 0－1 背包问题

C. 最小生成树问题　　D. 单源最短路径问题

7.具有 n 个顶点，e 条边的图采用邻接表存储结构，进行深度优先遍历运算的时间复杂度为(　　)。

A. O(n＋e)　　B. $O(n^2)$　　C. $O(e^2)$　　D. O(n)

8.二分图是指能将顶点划分成两个部分，每一部分内的顶点间没有边相连的简单无向图。那么，24 个顶点的二分图至多有(　　)条边。

A. 144　　B. 100　　C. 48　　D. 122

9.广度优先搜索时，一定需要用到的数据结构是(　　)。

A. 栈　　B. 二叉树　　C. 队列　　D. 哈希表

10.一个班学生分组做游戏，如果每组三人就多两人，每组五人就多三人，每组七人就多四人，这个班的学生人数 n 在(　　)区间，已知 n＜60。

A. 30＜n＜40　　B. 40＜n＜50　　C. 50＜n＜60　　D. 20＜n＜30

11.小明想通过走楼梯来锻炼身体，假设从第 1 层走到第 2 层消耗 10 卡热量，接着从第 2 层走到第 3 层消耗 20 卡热量，再从第 3 层走到第 4 层消耗 30 卡热量，依此类推，从第 k 层走到第 k＋1 层消耗 10k 卡热量(k＞1)。如果小明想从 1 层开始，通过连续向上爬楼梯

消耗 1000 卡热量，至少要爬到第（　　）层楼。

A. 14　　B. 16　　C. 15　　D. 13

12. 表达式 a * (b+c)－d 的后缀表达形式为（　　）。

A. abc * +d－　　B. －+ * abcd　　C. abcd * +－　　D. abc+ * d－

13. 从一个 4×4 的棋盘中选取不在同一行也不在同一列上的两个方格，共有（　　）种方法。

A. 60　　B. 72　　C. 86　　D. 64

14. 对一个 n 个顶点、m 条边的带权有向简单图用 Dijkstra 算法计算单源最短路时，如果不使用堆或其他优先队列进行优化，则其时间复杂度为（　　）。

A. $O((m+n^2)\log n)$　　B. $O(mn+n^3)$

C. $O((m+n)\log n)$　　D. $O(n^2)$

15. 1948 年，（　　）将热力学中的熵引入信息通信领域，标志着信息论研究的开端。

A. 欧拉(Leonhard Euler)　　B. 冯·诺伊曼(John von Neumann)

C. 克劳德·香农(Claude Shannon)　　D. 图灵(Alan Turing)

二、阅读程序（程序输入不超过数组或字符串定义的范围；判断题正确填√，错误填×；除特殊说明外，判断题每题 1.5 分，选择题每题 3 分，共计 40 分）

1.

```
01 #include<iostream>
02 using namespace std;
03
04 int n;
05 int d[1000];
06
07 int main(){
08     cin>>n;
09     for (int i=0;i<n;++i)
10       cin>>d[i];
11    int ans=-1;
12    for (int i=0;i<n;++i)
13      for (int j=0;j<n;++j)
14        if (d[i]<d[j])
15          ans=max(ans,d[i]+d[j]-(d[i] & d[j]));
16    cout<<ans;
17    return 0;
18  }
```

假设输入的 n 和 d[i]都是不超过 10000 的正整数，完成下面的判断题和单选题：

判断题

(1) n 必须小于 1000，否则程序可能会发生运行错误。（　　）

(2) 输出一定大于等于 0。（　　）

(3) 若将第 13 行的 j=0 改为 j=i+1，程序输出可能会改变。（　　）

(4) 将第 14 行的 d[i]<d[j]改为 d[i] !=d[j]，程序输出不会改变。（　　）

●单选题

(5)若输入 n 为 100,且输出为 127,则输入的 d[i]中不可能有(　　)。

A. 127　　B. 126　　C. 128　　D. 125

(6)若输出的数大于 0,则下面说法正确的是(　　)。

A. 若输出为偶数,则输入的 d[i]中最多有两个偶数

B. 若输出为奇数,则输入的 d[i]中至少有两个奇数

C. 若输出为偶数,则输入的 d[i]中至少有两个偶数

D. 若输出为奇数,则输入的 d[i]中最多有两个奇数

2.

```
#include<iostream>
#include<cstdlib>
using namespace std;

int n;
int d[10000];

int find(int L,int R,int k){
   int x=rand()%(R-L+1)+L;
   swap(d[L],d[x]);
   int a=L+1,b=R;
   while (a<b){
     while (a<b && d[a]<d[L])
       ++a;
     while (a<b && d[b]>=d[L])
       --b;
     swap(d[a],d[b]);
   }
   if (d[a]<d[L])
     ++a;
   if (a-L==k)
     return d[L];
   if (a-L<k)
     return find(a,R,k-(a-L));
   return find(L+1,a-1,k);
}

int main(){
   int k;
   cin>>n;
   cin>>k;
   for (int i=0;i<n;++i)
```

```
33      cin>>d[i];
34    cout<<find(0,n-1,k);
35    return 0;
36  }
```

假设输入的 n，k 和 d[i]都是不超过 10000 的正整数，且 k 不超过 n，并假设 rand()函数产生的是均匀的随机数，完成下面的判断题和单选题：

●判断题

(1)第 9 行的 x 的数值范围是 L+1 到 R，即[L+1,R]。(　　)

(2)将第 19 行的 d[a]改为 d[b]，程序不会发生运行错误。(　　)

●单选题

(3)(2.5 分)当输入的 d[i]是严格单调递增序列时，第 17 行的 swap 平均执行次数是(　　)。

A. O(n log n)　　B. O(n)　　C. $O(\log^2 n)$　　D. $O(n^2)$

(4)(2.5 分)当输入的 d[i]是严格单调递减序列时，第 17 行的 swap 平均执行次数是(　　)。

A. $O(n^2)$　　B. O(n)　　C. O(n log n)　　D. O(log n)

(5)(2.5 分)若输入的 d[i]为 i，此程序①平均的时间复杂度和②最坏情况下的时间复杂度分别是(　　)。

A. O(n)，$O(n^2)$　　B. O(n)，O(n log n)

C. O(n log n)，$O(n^2)$　　D. O(n log n)，O(n log n)

(6)(2.5 分)若输入的 d[i]都为同一个数，此程序平均的时间复杂度是(　　)。

A. O(n)　　B. O(log n)　　C. O(n log n)　　D. $O(n^2)$

3.

```
01 #include<iostream>
02 #include<queue>
03 using namespace std;
04
05 const int maxl=2000000000;
06
07 class Map{
08     struct item{
09         string key;int value;
10     } d[maxl];
11     int cnt;
12  public:
13     int find(string x){
14       for (int i=0;i<cnt;++i)
15         if (d[i].key==x)
16           return d[i].value;
17       return -1;
18     }
```

```
19    static int end(){ return-1;}
20    void insert(string k,int v){
21      d[cnt].key=k;d[cnt++].value=v;
22    }
23  } S[2];
24
25  class Queue{
26    string q[maxl];
27    int head,tail;
28  public:
29    void pop(){++head;}
30    string front(){ return q[head+1];}
31    bool empty(){ return head==tail;}
32    void push(string x){ q[++tail]=x;}
33  } q[2];
34
35  string st0,st1;
36  int m;
37
38  string LtoR(string s,int L,int R){
39    string t=s;
40    char tmp=t[L];
41    for (int i=L;i<R;++i)
42      t[i]=t[i+1];
43    t[R]=tmp;
44    return t;
45  }
46
47  string RtoL(string s,int L,int R){
48    string t=s;
49    char tmp=t[R];
50    for (int i=R;i>L;--i)
51      t[i]=t[i-1];
52    t[L]=tmp;
53    return t;
54  }
55
56  bool check(string st,int p,int step){
57    if (s[p].find(st) !=s[p].end())
58      return false;
59    ++step;
60    if (s[p^1].find(st)==s[p].end()){
```

```
61      s[p].insert(st,step);
62      q[p].push(st);
63      return false;
64    }
65    cout<<s[p^1].find(st)+step<<endl;
66    return true;
67  }
68
69  int main(){
70    cin>>st0>>st1;
71    int len=st0.length();
72    if (len !=st1.length()){
73      cout<<-1<<endl;
74      return 0;
75    }
76    if (st0==st1){
77      cout<<0<<end1;
78      return 0;
79    }
80    cin>>m;
81    s[0].insert(st0,0);s[1].insert(st1,0);
82    q[0].push(st0);q[1].push(st1);
83    for (int p=0;
84        !(q[0].empty() && q[1].empty());
85        p^=1){
86      string st- q[p].front();q[p].pop();
87      int step=s[p].find(st);
88      if ((p==0 &&
89          (check(LtoR(st,m,len-1),p,step) ||
90          check(RtoL(st,0,m),p,step)))
91            ||
92          (p==1  &&
93            (check(LtoR(st,0,m),p,step) ||
94            check(RtoL(st,m,len-1),p,step))))
95          return 0;
96    }
97    cout<<-1<<endl;
98    return 0;
99  }
```

判断题

(1)输出可能为0。(　　)

(2)若输入的两个字符串长度均为101时,则m=0时的输出与m=100时的输出是一样的。(　　)

(3)若两个字符串的长度均为 n,则最坏情况下,此程序的时间复杂度为 O(n!)。(　　)

●单选题

(4)(2.5 分)若输入的第一个字符串长度由 100 个不同的字符构成,第二个字符串是第一个字符串的倒序,输入的 m 为 0,则输出为(　　)。

A. 49　　B. 50　　C. 100　　D. －1

(5)(4 分)已知当输入为 0123\n3210\n1 时输出为 4,当输入为 012345\n543210\n1 时,输出为 14,当输入为 01234567\n76543210\n1 时输出为 28,则当输入为 0123456789ab\nba9876543210\n1 输出为(　　)(其中\n 为换行符)。

A. 56　　B. 84　　C. 102　　D. 68

(6)(4 分)若两个字符串的长度均为 n,且 0＜m＜n－1,且两个字符串的构成相同(即任何一个字符在两个字符串中出现的次数均相同),则下列说法正确的是(　　)。

提示:考虑输入与输出有多少对字符前后顺序不一样。

A. 若 n、m 均为奇数,则输出可能小于 0

B. 若 n、m 均为偶数,则输出可能小于 0

C. 若 n 为奇数、m 为偶数,则输出可能小于 0

D. 若 n 为偶数、m 为奇数,则输出可能小于 0

三、完善程序(单选题,每小题 3 分,共计 30 分)

1.(分数背包)小 S 有 n 块蛋糕,编号从 1 到 n。第 i 块蛋糕的价值是 w_i,体积是 v_i。他有一个大小为 B 的盒子来装这些蛋糕,也就是说装入盒子的蛋糕的体积总和不能超过 B。

他打算选择一些蛋糕装入盒子,并希望盒子里装的蛋糕的价值之和尽量大。

为了使盒子里的蛋糕价值之和更大,他可以任意切割蛋糕。具体来说,可以选择一个 α(0＜α＜1),并将一块价值是 w,体积为 v 的蛋糕切割成两块,其中一块的价值是 α·w,体积是 α·v,另一块的价值是(1－α)·w,体积是(1－α)·v。他可以重复无限次切割操作。现要求编程输出最大可能的价值,以分数的形式输出。

比如 n＝3,B＝8,三块蛋糕的价值分别是 4、4、2,体积分别是 5、3、2。那么最优的方案就是将体积为 5 的蛋糕切成两份,一份体积是 3,价值是 2.4,另一份体积是 2,价值是 1.6,然后把体积是 3 的那部分和后两块蛋糕打包进盒子。最优的价值之和是 8.4,故程序输出 42/5。

输入的数据范围为 $1\leqslant n\leqslant 1000$,$1\leqslant B\leqslant 10^5$,$1\leqslant w_i, v_i\leqslant 100$。

提示:将所有的蛋糕按照性价比 w_i/v_i 从大到小排序后进行贪心选择。

试补全程序。

```
01 #include<cstdio>
02 using namespace std;
03
04 const int maxn=1005;
05
06 int n,B,w[maxn],v[maxn];
07
08 int gcd(int u,int v){
09   if(v==0)
10       return u;
```

```
11   return gcd(v,u%v);
12 }
13
14 void print(int w,int v){
15   int d=gcd(w,v);
16   w=w/d;
17   v=v/d;
18   if(v==1)
19     printf("%d\n",w);
20   else
21     printf("%d/%d\n",w,v);
22 }
23
24 void swap(int &x,int &y){
25   int t=x;x=y;y=t;
26 }
27
28 int main(){
29   scanf("%d%d",&n,&B);
30   for(int i=1;i<=n;i++){
31     scanf("%d%d",&w[i],&v[i]);
32   }
33   for(int i=1;i<n;i++)
34     for(int j=1;j<n;j++)
35       if( ① ){
36         swap(w[j],w[j+1]);
37         swap(v[j],v[j+1]);
38       }
39   int curV,curW;
40   if( ② ){
41     ③
42   }else{
43     print(B*w[1],v[1]);
44     return 0;
45   }
46
47   for(int i=2;i<=n;i++)
48     if(curV+v[i]<=B){
49     curV+=v[i];
50     curW+=w[i];
51     }else{
52     print( ④ );
```

```
53        return 0;
54        }
55      print(  ⑤  );
56      return 0;
57  }
58
59
```

(1)①处应填(　　)。

A. w[j]/v[j]<w[j+1]/v[j+1]　　B. w[j]/v[j]>w[j+1]/v[j+1]

C. v[j]*w[j+1]<v[j+1]*w[j]　　D. w[j]*v[j+1]<w[j+1]*v[j]

(2)②处应填(　　)。

A. w[1]<=B　　B. v[1]<=B　　C. w[1]>=B　　D. v[1]>=B

(3)③处应填(　　)。

A. print(v[1], w[1]); return 0;　　B. curV=0; curW=0;

C. print(w[1], v[1]); return 0;　　D. curV=v[1]; curW=w[1];

(4)④处应填(　　)。

A. curW*v[i]+curV*w[i],v[i]

B. (curW-w[i])*v[i]+(B-curV)*w[i],v[i]

C. curW+v[i], w[i]

D. curW*v[i]+(B-curV)*w[i],v[i]

(5)⑤处应填(　　)。

A. curW,curV　　B. curW,1　　C. curV,curW　　D. curV,1

2.(最优子序列)取 m=16,给出长度为 n 的整数序列$a_1,a_2,\cdots,a_n(0\leqslant a_i<2^m)$。对于一个二进制数 x,定义其分值 w(x)为 x+popcnt(x),其中 popcnt(x)表示 x 二进制表示中 1 的个数。对于一个子序列$b_1,b_2,\cdots,b_k$,定义其子序列分值 S 为$w(b_1\oplus b_2)+w(b_2\oplus b_3)+w(b_3\oplus b_4)+\cdots+w(b_{k-1}\oplus b_k)$。其中⊕表示按位异或。对于空子序列,规定其子序列分值为 0。求一个子序列使得其子序列分值最大,输出这个最大值。

输入第一行包含一个整数 n(1≤n≤40000),接下来一行包含 n 个整数$a_1,a_2,\cdots,a_n$。

提示:考虑优化朴素的动态规划算法,将前$\frac{m}{2}$位和后$\frac{m}{2}$位分开计算。

Max[x][y]表示当前的子序列下一个位置的高 8 位是 x、最后一个位置的低 8 位是 y 时的最大价值。

试补全程序。

```
01 #include<iostream>
02
03 using namespace std;
04
05 typedef long long LL;
06
07 const int MAXN=40000,M=16,B=M>>1,MS=(1<<B)-1;
08 const LL INF=1000000000000000LL;
09 LL Max[MS+4][MS+4];
```

```
10
11  int w(int x)
12  {
13    int s=x;
14    while (x)
15    {
16      ①  ;
17      s++;
18    }
19    return s;
20  }
21
22  void to_max(LL &x,LL y)
23  {
24    if(x<y)
25      x=y;
26  }
27
28  int main()
29  {
30    int n;
31    LL ans=0;
32    cin>>n;
33    for (int x=0;x<=MS;x++)
34      for (int y=0;y<=MS;y++)
35        Max[x][y]=-INF;
36    for (int i=1;i<=n;i++)
37    {
38      LL a;
39      cin>>a;
40      int x= ②  ,y=a & MS;
41      LL v= ③  ;
42      for (int z=0;z<=MS;z++)
43        to_max(v, ④  );
44      for (int z=0;z<=MS;z++)
45        ⑤  ;
46      to_max(ans,v);
47    }
48    cout<<ans<<endl;
49    return 0;
50  }
```

(1)①处应填(　　)。

A. x>>=1　　B. x^=x & (x^(x+1))

C. x-=x|-x　　D. x^=x & (x^(x-1))

(2)②处应填(　　)。

A. (a & MS)<<B　　B. a>>B

C. a & (1<<B)　　D. a & (MS<<B)

(3)③处应填(　　)。

A. -INF　　B. Max[y][x]　　C. 0　　D. Max[x][y]

(4)④处应填(　　)。

A. Max[x][z]+w(y^z)　　B. Max[x][z]+w(a^z)

C. Max[x][z]+w(x^(z<<B))　　D. Max[x][z]+w(x^z)

(5)⑤处应填(　　)。

A. to_max(Max[y][z],v+w(a^(z<<B)))

B. to_max(Max[z][y],v+w((x^z)<<B))

C. to_max(Max[z][y],v+w(a^(z<<B)))

D. to_max(Max[x][z],v+w(y^z)

2021 CCF 非专业级别软件能力认证第一轮(CSP－S)提高级 C＋＋语言试题

一、单项选择题(共 15 题,每题 2 分,共计 30 分;每题有且仅有一个正确选项)

1.在 Linux 系统终端中,用于列出当前目录下所含的文件和子目录的命令为(　　)。

A. ls　　B. cd　　C. cp　　D. all

2.二进制数 00101010_2 和 00010110_2 的和为(　　)。

A. 00111100_2　　B. 01000000_2　　C. 00111100_2　　D. 01000010_2

3.在程序运行过程中,如果递归调用的层数过多,可能会由于(　　)引发错误。

A. 系统分配的栈空间溢出　　B. 系统分配的队列空间溢出

C. 系统分配的链表空间溢出　　D. 系统分配的堆空间溢出

4.以下排序方法中,(　　)是不稳定的。

A. 插入排序　　B. 冒泡排序　　C. 堆排序　　D. 归并排序

5.以比较为基本运算,对于 2n 个数,同时找到最大值和最小值,最坏情况下需要的最小的比较次数为(　　)。

A. 4n－2　　B. 3n＋1　　C. 3n－2　　D. 2n＋1

6.现有一个地址区间为 0～10 的哈希表,对于出现冲突情况,会往后找第一个空的地址存储(到 10 冲突了就从 0 开始往后),现在要依次存储(0,1,2,3,4,5,6,7),哈希函数为 $h(x)=x^2 \bmod 11$。请问 7 存储在哈希表哪个地址中(　　)。

A. 5　　B. 6　　C. 7　　D. 8

7.G 是一个非连通简单无向图(没有自环和重边),共有 36 条边,则该图至少有(　　)个点。

A. 8　　B. 9　　C. 10　　D. 11

8.令根结点的高度为 1,则一棵含有 2021 个结点的二叉树的高度至少为(　　)。

A. 10　　B. 11　　C. 12　　D. 2021

9.前序遍历和中序遍历相同的二叉树为当且仅当(　　)。

A. 只有 1 个点的二叉树　　B. 根结点没有左子树的二叉树

C. 非叶子结点只有左子树的二叉树　　D. 非叶子结点只有右子树的二叉树

10.定义一种字符串操作为交换相邻两个字符。将 DACFEB 变为 ABCDEF 最少需要(　　)次上述操作。

A. 7　　B. 8　　C. 9　　D. 6

11.有如下递归代码

```
solve(t,n):
    if t=1 return 1;
    else return 5*solve(t-1,n) mod n;
```

则 solve(23,23)的结果为(　　)。

A. 1　　B. 7　　C. 12　　D. 22

12.斐波那契数列的定义为 $F_1=1$,$F_2=1$,$F_n=F_{n-1}+F_{n-2}$($n\geqslant 3$)。现在用如下程序来计算斐波那契数列的第 n 项,其时间复杂度为(　　)。

```
F(n):
    if n<=2 return 1;
    else return F(n-1)+F(n-2);
```

A. O(n)　　B. O(n^2)　　C. O(2^n)　　D. O(n log n)

13. 有 8 个苹果从左到右排成一排，你要从中挑选至少一个苹果，并且不能同时挑选相邻的两个苹果，一共有(　　)种方案。

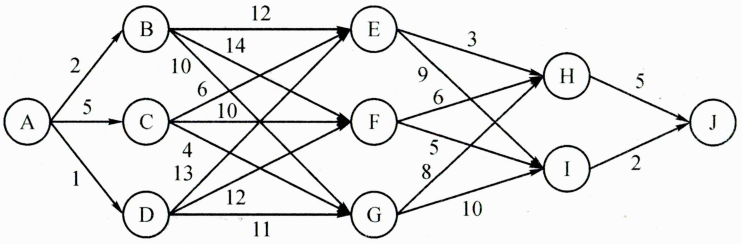
A. 36　　B. 48　　C. 54　　D. 64

14. 设一个三位数 $n=\overline{abc}$，a，b，c 均为 1～9 之间的整数，若以 a、b、c 作为三角形的三条边可以构成等腰三角形(包括等边)，则这样的 n 有(　　)个。

A. 81　　B. 120　　C. 165　　D. 216

15. 有如下的有向图，结点为 A，B，…，J，其中每条边的长度都标在图中。则结点 A 到结点 J 的最短路径长度为(　　)。

A. 16　　B. 19　　C. 20　　D. 22



二、阅读程序(程序输入不超过数组或字符串定义的范围；判断题正确填√，错误填×；除特殊说明外，判断题每题 1.5 分，选择题每题 3 分，共计 40 分)

1.

```
01 #include<iostream>
02 #include<cmath>
03 using namespace std;
04
05 const double r=acos(0.5);
06
07 int a1,b1,c1,d1;
08 int a2,b2,c2,d2;
09
10 inline int sq(const int x) { return x*x;}
11 inline int cu(const int x) { return x*x*x;}
12
13 int main()
14 {
15     cout.flags(ios::fixed);
16     cout.precision(4);
17
18     cin>>a1>>b1>>c1>>d1;
19     cin>>a2>>b2>>c2>>d2;
20
21     int t=sq(a1-a2)+sq(b1-b2)+sq(c1-c2);
22
23     if (t<=sq(d2-d1)) cout<<cu(min(d1,d2))*r*4;
24     else if (t>=sq(d2+d1)) cout<<0;
25     else {
26         double x=d1-(sq(d1)-sq(d2)+t)/sqrt(t)/2;
```

```
27          double y=d2-(sq(d2)-sq(d1)+t)/sqrt(t)/2;
28          cout<<(x*x*(3*d1-x)+y*y*(3*d2-y))*r;
29      }
30      cout<<endl;
31      return 0;
32  }
```

假设输入的所有数的绝对值都不超过1000，完成下面的判断题和单选题：

●判断题

(1)将第21行中t的类型声明从int改为double，不会影响程序运行的结果。(　　)

(2)将第26、27行中的/ sqrt(t)/2替换为/ 2/sqrt(t)，不会影响程序运行的结果。(　　)

(3)将第28行中的x * x改成sq(x)，y * y改成sq(y)，不会影响程序运行的结果。(　　)

(4)当输入为0 0 0 1 1 0 0 1时，输出为1.3090。(　　)

●单选题

(5)当输入为1 1 1 1 1 1 1 2时，输出为(　　)。

A. 3.1416　　B. 6.2832　　C. 4.7124　　D. 4.1888

(6)这段代码的含义为(　　)。

A. 求圆的面积并　　B. 求球的体积并

C. 求球的体积交　　D. 求椭球的体积并

2.

```
01 #include<algorithm>
02 #include<iostream>
03 using namespace std;
04
05 int n,a[1005];
06
07 struct Node
08 {
09     int h,j,m,w;
10
11     Node(const int _h,const int _j,const int _m,const int _w):
12         h(_h),j(_j),m(_m),w(_w)
13     { }
14
15     Node operator+(const Node &o) const
16     {
17         return Node(
18             max(h,w+o.h),
19             max(max(j,o.j),m+o.h),
20             max(m+o.w,o.m),
21             w+o.w);
22     }
```

```
};

Node solve1(int h,int m)
{
    if (h>m)
        return Node(-1,-1,-1,-1);
    if (h==m)
        return Node(max(a[h],0),max(a[h],0),max(a[h],0),a[h]);
    int j=(h+m)>>1;
    return solve1(h,j)+solve1(j+1,m);
}

int solve2(int h,int m)
{
    if (h>m)
        return-1;
    if (h==m)
        return max(a[h],0);
    int j=(h+m)>>1;
    int wh=0,wm=0;
    int wht=0,wmt=0;
    for (int i=j;i>=h;i--) {
        wht+=a[i];
        wh=max(wh,wht);
    }
    for (int i=j+1;i<=m;i++) {
        wmt+=a[i];
        wm=max(wm,wmt);
    }
    return max(max(solve2(h,j),solve2(j+1,m)),wh+wm);
}

int main()
{
    cin>>n;
    for (int i=1;i<=n;i++) cin>>a[i];
    cout<<solve1(1,n).j<<endl;
    cout<<solve2(1,n)<<endl;
    return 0;
}
```

假设输入的所有数的绝对值都不超过1000，完成下面的判断题和单选题：

●判断题

(1)程序总是会正常执行并输出两行两个相等的数。(　　)

(2)第28行与第38行分别有可能执行两次及以上。(　　)

(3)当输入为5－10 11－9 5－7时，输出的第二行为7。(　　)

●单选题

(4)solve1(1，n)的时间复杂度为(　　)。

A. O(log n)　　B. O(n)　　C. O(n log n)　　D. $O(n^2)$

(5)solve2(1，n)的时间复杂度为(　　)。

A. O(log n)　　B. O(n)　　C. O(n log n)　　D. $O(n^2)$

(6)当输入为10－3 2 10 0－8 9－4－5 9 4时，输出的第一行为(　　)。

A. 13　　B. 17　　C. 24　　D. 12

3.

```
01 #include<iostream>
02 #include<string>
03 using namespace std;
04
05 char base[64];
06 char table[256];
07
08 void init()
09 {
10     for (int i=0;i<26;i++) base[i]='A'+i;
11     for (int i=0;i<26;i++) base[26+i]='a'+i;
12     for (int i=0;i<10;i++) base[52+i]='0'+i;
13     base[62]='+',base[63]='/';
14
15     for (int i=0;i<256;i++) table[i]=0xff;
16     for (int i=0;i<64;i++) table[base[i]]=i;
17     table['=']=0;
18 }
19
20 string encode(string str)
21 {
22     string ret;
23     int i;
24     for (i=0;i+3<=str.size();i+=3) {
25         ret+=base[str[i]>>2];
26         ret+=base[(str[i] & 0x03)<<4 | str[i+1]>>4];
27         ret+=base[(str[i+1] & 0x0f)<<2 | str[i+2]>>6];
28         ret+=base[str[i+2] & 0x3f];
29     }
```

```
30      if (i<str.size()) {
31          ret+=base[str[i]>>2];
32          if (i+1==str.size()) {
33              ret+=base[(str[i] & 0x03)<<4];
34              ret+="==";
35          }
36          else {
37              ret+=base[(str[i] & 0x03)<<4 | str[i+1]>>4];
38              ret+=base[(str[i+1] & 0x0f)<<2];
39              ret+="=";
40          }
41      }
42      return ret;
43  }
44
45  string decode(string str)
46  {
47      string ret;
48      int i;
49      for (i=0;i<str.size();i+=4) {
50          ret+=table[str[i]]<<2 | table[str[i+1]]>>4;
51          if (str[i+2] !='=')
52              ret+=(table[str[i+1]] & 0x0f)<<4 | table[str[i+2]]>>2;
53          if (str[i+3] !='=')
54              ret+=table[str[i+2]]<<6 | table[str[i+3]];
55      }
56      return ret;
57  }
58
59  int main()
60  {
61      init();
62      cout<<int(table[0])<<endl;
63
64      int opt;
65      string str;
66      cin>>opt>>str;
67      cout<<(opt ? decode(str) : encode(str))<<endl;
68      return 0;
69  }
```

假设输入总是合法的(一个整数和一个不含空白字符的字符串,用空格隔开),完成下面的判断题和单选题:

●判断题

(1)程序总是先输出一行一个整数,再输出一行一个字符串。(　　)

(2)对于任意不含空白字符的字符串 str1,先执行程序输入 0 str1,得到输出的第二行记为 str2;再执行程序输入 1 str2,输出的第二行必为 str1。(　　)

(3)当输入为 1 SGVsbG93b3JsZA==时,输出的第二行为 HelloWorld。(　　)

●单选题

(4)设输入字符串长度为 n,encode 函数的时间复杂度为(　　)。

A. $O(\sqrt{n})$　　B. $O(n)$　　C. $O(n \log n)$　　D. $O(n^2)$

(5)输出的第一行为(　　)。

A. 0xff　　B. 255　　C. 0xFF　　D. −1

(6)当输入为 0 CSP2021csp 时,输出的第二行为(　　)。

A. Q1NQMjAyMWNzcAv=　　B. Q1NQMjAyMGNzcA==

C. Q1NQMjAyMGNzcAv=　　D. Q1NQMjAyMWNzcA==

三、完善程序(单选题,每小题 3 分,共计 30 分)

1.(魔法数字)小 H 的魔法数字是 4。给定 n,他希望用若干个 4 进行若干次加法、减法和整除运算得到 n。但由于小 H 计算能力有限,计算过程中只能出现不超过 M=10000 的正整数。求至少可能用到多少个 4。

例如,当 n=2 时,有 2=(4+4)/4,用到了 3 个 4,是最优方案。

试补全程序。

```
01 #include<iostream>
02 #include<cstdlib>
03 #include<climits>
04
05 using namespace std;
06
07 const int M=10000;
08 bool Vis[M+1];
09 int F[M+1];
10
11 void update(int &x,int y) {
12     if (y<x)
13         x=y;
14 }
15
16 int main() {
17     int n;
18     cin>>n;
19     for (int i=0;i<=M;i++)
20         F[i]=INT_MAX;
21     ____①____;
22     int r=0;
23     while (____②____) {
```

```
24          r++;
25          int x=0;
26          for (int i=1;i<=M;i++)
27              if (  ③  )
28                  x=i;
29          Vis[x]=1;
30          for (int i=1;i<=M;i++)
31              if (  ④  ) {
32                  int t=F[i]+F[x];
33                  if (i+x<=M)
34                      update(F[i+x],t);
35                  if (i !=x)
36                      update(F[abs(i-x)],t);
37                  if (i%x==0)
38                      update(F[i/x],t);
39                  if (x%i==0)
40                      update(F[x/i],t);
41              }
42      }
43      cout<<F[n]<<endl;
44      return 0;
45  }
```

(1)①处应填(　　)。

A. F[4]=0　　B. F[1]=4　　C. F[1]=2　　D. F[4]=1

(2)②处应填(　　)。

A. !Vis[n]　　B. r<n

C. F[M]==INT_MAX　　D. F[n]==INT_MAX

(3)③处应填(　　)。

A. F[i]==r　　B. !Vis[i] && F[i]==r

C. F[i]<F[x]　　D. !Vis[i] && F[i]<F[x]

(4)④处应填(　　)。

A. F[i]<F[x]　　B. F[i]<=r　　C. Vis[i]　　D. i<=x

2.(RMQ 区间最值问题)给定序列 $a_0,\cdots,a_{n-1}$ 和 m 次询问,每次询问给定 l,r,求 $\max\{a_l,\cdots,a_r\}$。

为了解决该问题,有一个算法叫 the Method of Four Russians,其时间复杂度为 O(n+m),步骤如下:

●建立 Cartesian(笛卡尔)树,将问题转化为树上的 LCA(最近公共祖先)问题。

●对于 LCA 问题,可以考虑其 Euler 序(即按照 DFS 过程,经过所有点,环游回根的序列),即求 Euler 序列上两点间一个新的 RMQ 问题。

●注意新的问题为±1 RMQ,即相邻两点的深度差一定为 1。

下面解决这个±1 RMQ 问题,"序列"指 Euler 序列:

●设 t 为 Euler 序列长度。取 $b=\left\lfloor\frac{\log_2 t}{2}\right\rfloor$。将序列每 b 个分为一大块，使用 ST 表(倍增表)处理大块间的 RMQ 问题，复杂度 $O\left(\frac{t}{b}\log t\right)=O(n)$。

●(重点)对于一个块内的 RMQ 问题，也需要 O(1)的算法。由于差分数组 2^{b-1}种，可以预处理出所有情况下的最值位置，预处理复杂度 $O(b2^b)$，不超过 O(n)。

●最终，对于一个查询，可以转化为中间整的大块的 RMQ 问题，以及两端块内的 RMQ 问题。

试补全程序。

```
001 #include<iostream>
002 #include<cmath>
003
004 using namespace std;
005
006 const int MAXN=100000,MAXT=MAXN<<1;
007 const int MAXL=18,MAXB=9,MAXC=MAXT/MAXB;
008
009 struct node {
010     int val;
011     int dep,dfn,end;
012     node*son[2];// son[0],son[1] 分别表示左右儿子
013 } T[MAXN];
014
015 int n,t,b,c,Log2[MAXC+1];
016 int Pos[(1<<(MAXB-1))+5],Dif[MAXC+1];
017 node*root,*A[MAXT],*Min[MAXL][MAXC];
018
019 void build() { // 建立 Cartesian 树
020     static node*S[MAXN+1];
021     int top=0;
022     for (int i=0;i<n;i++) {
023         node*p=&T[i];
024         while (top && S[top]->val<p->val)
025             ①  ;
026         if (top)
027             ②  ;
028         S[++top]=p;
029     }
030     root=S[1];
031 }
032
033 void DFS(node*p) { // 构建 Euler 序列
```

```
034     A[p->dfn=t++]=p;
035     for (int i=0;i<2;i++)
036         if (p->son[i]) {
037             p->son[i]->dep=p->dep+1;
038             DFS(p->son[i]);
039             A[t++]=p;
040         }
041     p->end=t-1;
042 }
043
044 node*min(node*x,node*y) {
045     return ____③____ ? x : y;
046 }
047
048 void ST_init() {
049     b=(int)(ceil(log2(t)/2));
050     c=t/b;
051     Log2[1]=0;
052     for (int i=2;i<=c;i++)
053         Log2[i]=Log2[i>>1]+1;
054     for (int i=0;i<c;i++) {
055         Min[0][i]=A[i*b];
056         for (int j=1;j<b;j++)
057             Min[0][i]=min(Min[0][i],A[i*b+j]);
058     }
059     for (int i=1,l=2;l<=c;i++,l<<=1)
060         for (int j=0;j+l<=c;j++)
061             Min[i][j]=min(Min[i-1][j],Min[i-1][j+(l>>1)]);
062 }
063
064 void small_init() { //块内预处理
065     for (int i=0;i<=c;i++)
066         for (int j=1;j<b && i*b+j<t;j++)
067             if (____④____)
068                 Dif[i] |=1<<(j-1);
069     for (int S=0;S<(1<<(b-1));S++) {
070         int mx=0,v=0;
071         for (int i=1;i<b;i++) {
072             ____⑤____;
073             if (v<mx) {
074                 mx=v;
075                 Pos[S]=i;
```

```
076             }
077         }
078     }
079 }
080
081 node*ST_query(int l,int r) {
082     int g=Log2[r-l+1];
083     return min(Min[g][l],Min[g][r-(1<<g)+1]);
084 }
085
086 node*small_query(int l,int r) { // 块内查询
087     int p=l/b;
088     int S=  ⑥  ;
089     return A[l+Pos[S]];
090 }
091
092 node*query(int l,int r) {
093     if (l>r)
094         return query(r,l);
095     int pl=l/b,pr=r/b;
096     if (pl==pr) {
097         return small_query(l,r);
098     } else {
099         node*s=min(small_query(l,pl*b+b-1),small_query(pr*b,r));
100     if (pl+1<=pr-1)
101         s=min(s,ST_query(pl+1,pr-1));
102     return s;
103     }
104 }
105
106 int main() {
107     int m;
108     cin>>n>>m;
109     for (int i=0;i<n;i++)
110         cin>>T[i].val;
111     build();
112     DFS(root);
113     ST_init();
114     small_init();
115     while (m--) {
116         int l,r;
117         cin>>l>>r;
```

```
118         cout<<query(T[l].dfn,T[r].dfn)->val<<endl;
119     }
120     return 0;
121 }
```

(1)①处应填(　　)。

A. p->son[0]=S[top--]　　B. p->son[1]=S[top--]

C. S[top--]->son[0]=p　　D. S[top--]->son[1]=p

(2)②处应填(　　)。

A. p->son[0]=S[top]　　B. p->son[1]=S[top]

C. S[top]->son[0]=p　　D. S[top]->son[1]=p

(3)③处应填(　　)。

A. x->dep<y->dep　　B. x<y

C. x->dep>y->dep　　D. x->val<y->val

(4)④处应填(　　)。

A. A[i*b+j-1]==A[i*b+j]->son[0]

B. A[i*b+j]->val< A[i*b+j-1]->val

C. A[i*b+j]==A[i*b+j-1]->son[1]

D. A[i*b+j]->dep<A[i*b+j-1]->dep

(5)⑤处应填(　　)。

A. v+=(S>>i &1)?-1: 1　　B. v+=(S>>i & 1)? 1:-1

C. v+=(S>>(i-1)& 1)? 1:-1　　D. v+=(S>>(i-1)& 1)?-1: 1

(6)⑥处应填(　　)。

A. (Dif[p]>>(r-p*b)) & ((1<<(r-l))-1)

B. Dif[p]

C. (Dif[p]>>(l-p*b)) & ((1<<(r-l))-1)

D. (Dif[p]>>((p+1)*b-r)) & ((1<<(r-l+1))-1)

2022 CCF 非专业级别软件能力认证第一轮(CSP－S)
提高级 C＋＋语言试题

一、单项选择题(共 15 题，每题 2 分，共计 30 分；每题有且仅有一个正确选项)

1. 在 Linux 系统终端中，用于切换工作目录的命令为(　　)。

A. ls　　B. cd　　C. cp　　D. all

2. 你同时用 time 命令和秒表为某个程序在单核 CPU 的运行计时。假如 time 命令的输出如下：

```
real    0m30.721s
user    0m24.579s
sys     0m6.123s
```

以下最接近秒表计时的时长为(　　)。

A. 30s　　B. 24s　　C. 18s　　D. 6s

3. 若元素 a、b、c、d、e、f 依次进栈，允许进栈、退栈操作交替进行，但不允许连续三次退栈操作，则不可能得到的出栈序列是(　　)。

A. dcebfa　　B. cbdaef　　C. bcaefd　　D. afedcb

4. 考虑对 n 个数进行排序，以下最坏时间复杂度低于 $O(n^2)$ 的排序方法是(　　)。

A. 插入排序　　B. 冒泡排序　　C. 归并排序　　D. 快速排序

5. 假设在基数排序过程中，受宇宙射线的影响，某项数据异变为一个完全不同的值。请问排序算法结束后，可能出现的最坏情况是(　　)。

A. 移除受影响的数据后，最终序列是有序序列

B. 移除受影响的数据后，最终序列是前后两个有序的子序列

C. 移除受影响的数据后，最终序列是一个有序的子序列和一个基本无序的子序列

D. 移除受影响的数据后，最终序列基本无序

6. 计算机系统用小端(Little Endian)和大端(Big Endian)来描述多字节数据的存储地址顺序模式，其中小端表示将低位字节数据存储在低地址的模式、大端表示将高位字节数 据存储在低地址的模式。在小端模式的系统和大端模式的系统分别编译和运行以下 C++ 代码段表示的程序，将分别输出什么结果？(　　)

```
unsigned x=0xDEADBEEF;
unsigned char*p=(unsigned char*)&x;
printf("%X",*p);
```

A. EF、EF　　B. EF、DE　　C. DE、EF　　D. DE、DE

7. 一个深度为 5(根结点深度为 1)的完全 3 叉树，按前序遍历的顺序给结点从 1 开始编号，则第 100 号结点的父结点是第(　　)号。

A. 95　　B. 96　　C. 97　　D. 98

8. 强连通图的性质不包括(　　)。

A. 每个顶点的度数至少为 1

B. 任意两个顶点之间都有边相连

C. 任意两个顶点之间都有路径相连

D. 每个顶点至少都连有一条边

9. 每个顶点度数均为 2 的无向图称为“2 正规图”。由编号为从 1 到 n 的顶点构成的所有 2 正规图，其中包含欧拉回路的不同 2 正规图的数量为(　　)。

A. n!　　B. (n－1)!　　C. n!/2　　D. (n－1)!/2

10. 共有 8 人选修了程序设计课程，期末大作业要求由 2 人组成的团队完成。假设不区分每个团队内 2 人的角色和作用，请问共有多少种可能的组队方案？(　　)

A. 28　　B. 32　　C. 56　　D. 64

11. 小明希望选到形如“省 A ·￡￡DDD”的车牌号。车牌号在“·”之前的内容固定不变；后面的 5 位号码中，前 2 位必须是大写英文字母，后 3 位必须是阿拉伯数字(￡代表 A 至 Z，D 表示 0 至 9，两个￡和三个 D 之间可能相同也可能不同)。请问总共有多少个可供选择的车牌号？(　　)

A. 20280　　B. 52000　　C. 676000　　D. 1757600

12. 给定地址区间为 0～9 的哈希表，哈希函数为 h(x) = x% 10，采用线性探查的冲突解决策略(对于出现冲突情况，会往后探查第一个空的地址存储；若地址 9 冲突了则从地址 0 重新开始探查)。哈希表初始为空表，依次存储(71，23，73，99，44，79，89)后，请问 89 存储在哈希表哪个地址中？(　　)

A. 9　　B. 0　　C. 1　　D. 2

13. 对于给定的 n，分析以下代码段对应的时间复杂度，其中最为准确的时间复杂度为(　　)。

```
int i, j,k=0;
for (i=0; i<n; i++) {
  for (j=0; j<n; j*=2) {
    k=k+n/2;
  }
}
```

A. O(n)　　B. O(n log n)　　C. $O(n\sqrt{n})$　　D. $O(n^2)$

14. 以比较为基本运算，在 n 个数的数组中找最大的数，在最坏情况下至少要做(　　)次运算。

A. n/2　　B. n－1　　C. n　　D. n＋1

15. ack 函数在输入参数“(2,2)”时的返回值为(　　)。

```
unsigned ack(unsigned m, unsigned n) {
  if (m==0) return n+1;
  if (n==0) return ack(m-1, 1);
  return ack(m-1, ack(m, n-1));
}
```

A. 5　　B. 7　　C. 9　　D. 13

二、阅读程序(程序输入不超过数组或字符串定义的范围；判断题正确填√，错误填×；除特殊说明外，判断题每题 1.5 分，选择题每题 3 分，共计 40 分)

(一)

```
01 #include<iostream>
02 #include<string>
```

```
03 #include<vector>
04
05 using namespace std;
06
07 int f(const string &s, const string &t)
08 {
09     int n=s.length(), m=t.length();
10
11     vector<int>shift(128, m+1);
12
13     int i, j;
14
15     for (j=0; j<m; j++)
16         shift[t[j]]=m-j;
17
18     for (i=0; i<=n-m; i+=shift[s[i+m]]) {
19         j=0;
20         while (j<m && s[i+j]==t[j])j++;
21         if (j==m) return i;
22     }
23
24     return-1;
25 }
26
27 int main()
28 {
29     string a, b;
30     cin>>a>>b;
31     cout<<f(a, b)<<endl;
32     return 0;
33 }
```

假设输入字符串由 ASCII 可见字符组成，完成下面的判断题和单选题：

●判断题

16. (1 分) 当输入为“abcde fg”时，输出为－1。(　　)

17. 当输入为“abbababbbab abab”时，输出为 4。(　　)

18. 当输入为“GoodLuckCsp2022 22”时，第 20 行的“j++”语句执行次数为 2。(　　)

●单选题

19. 该算法最坏情况下的时间复杂度为(　　)。

A. O(n＋m)　　B. O(n log m)　　C. O(m log n)　　D. O(nm)

20. f(a, b)与下列(　　)语句的功能最类似。

A. a.find(b)　　B. a.rfind(b)

C. a.substr(b)　　D. a.compare(b)

21. 当输入为“baaabaaabaaabaaaa aaaa”，第 20 行的“j++”语句执行次数为（　　）。

A. 9　　B. 10　　C. 11　　D. 12

（二）

```
01 #include<iostream>
02
03 using namespace std;
04
05 const int MAXN=105;
06
07 int n, m, k, val[MAXN];
08 int temp[MAXN], cnt[MAXN];
09
10 void init()
11 {
12     cin>>n>>k;
13     for (int i=0; i<n; i++) cin>>val[i];
14     int maximum=val[0];
15     for (int i=1; i<n; i++)
16         if (val[i]>maximum) maximum=val[i];
17     m=1;
18     while (maximum>=k) {
19         maximum/=k;
20         m++;
21     }
22 }
23
24 void solve()
25 {
26     int base=1;
27     for (int i=0; i<m; i++) {
28         for (int j=0; j<k; j++) cnt[j]=0;
29         for (int j=0; j<n; j++) cnt[val[j]/base%k]++;
30         for (int j=1; j<k; j++) cnt[j]+=cnt[j-1];
31         for (int j=n-1; j>=0; j--) {
32             temp[cnt[val[j]/base%k]-1]=val[j];
33             cnt[val[j]/base%k]--;
34         }
35         for (int j=0; j<n; j++) val[j]=temp[j];
36         base*=k;
37     }
38 }
39
40 int main()
```

```
41 {
42     init();
43     solve();
44     for (int i=0; i<n; i++) cout<<val[i]<<' ';
45     cout<<endl;
46     return 0;
47 }
```

假设输入的 n 为不大于 100 的正整数，k 为不小于 2 且不大于 100 的正整数，val[i]在 int 表示范围内，完成下面的判断题和单选题：

●判断题

22. 这是一个不稳定的排序算法。（　　）

23. 该算法的空间复杂度仅与 n 有关。（　　）

24. 该算法的时间复杂度为 O(m (n+k))。（　　）

●单选题

25. 当输入为“5 3 98 26 91 37 46”时，程序第一次执行到第 36 行，val[]数组的 内容依次为（　　）。

A. 91 26 46 37 98　　B. 91 46 37 26 98

C. 98 26 46 91 37　　D. 91 37 46 98 26

26. 若 val[i]的最大值为 100，k 取（　　）时算法运算次数最少。

A. 2　　B. 3　　C. 10　　D. 不确定

27. 当输入的 k 比 val[i]的最大值还大时，该算法退化为（　　）算法。

A. 选择排序　　B. 冒泡排序　　C. 计数排序　　D. 桶排序

(三)

```
01 #include<iostream>
02 #include<algorithm>
03
04 using namespace std;
05
06 const int MAXL=1000;
07
08 int n, k, ans[MAXL];
09
10 int main(void)
11 {
12     cin>>n>>k;
13     if (!n) cout<<0<<endl;
14     else
15     {
16         int m=0;
17         while (n)
18         {
19             ans[m++]=(n%(-k)+k)%k;
20             n=(ans[m-1]-n)/k;
```

```
21        }
22        for (int i=m-1; i>=0; i--)
23            cout<<char(ans[i]>=10 ?
24                ans[i]+'A'-10 :
25                ans[i]+'0');
26        cout<<endl;
27    }
28    return 0;
29 }
```

假设输入的 n 在 int 范围内，k 为不小于 2 且不大于 36 的正整数，完成下面的判断题和单选题：

●判断题

28. 该算法的时间复杂度为 $O(\log_k n)$。（　　）

29. 删除第 23 行的强制类型转换，程序的行为不变。（　　）

30. 除非输入的 n 为 0，否则程序输出的字符数为 $O(\lfloor \log_k |n| \rfloor+1)$。（　　）

●单选题

31. 当输入为“100 7”时，输出为（　　）。

A. 202　　B. 1515　　C. 244　　D. 1754

32. 当输入为“ －255 8”时，输出为“（　　）”。

A. 1400　　B. 1401　　C. 417　　D. 400

33. 当输入为“1000000 19”时，输出为“（　　）”。

A. BG939　　B. 87GIB

C. 1CD428　　D. 7CF1B

三、完善程序（单选题，每小题 3 分，共计 30 分）

（一）（归并第 k 小）已知两个长度均为 n 的有序数组 a1 和 a2（均为递增序，但不保证严格单调递增），并且给定正整数 k（1≤k≤2n），求数组 a1 和 a2 归并排序后的数组里第 k 小的数值。

试补全程序。

```
01 #include<bits/stdc++.h>
02 using namespace std;
03
04 int solve(int*a1, int*a2, int n, int k) {
05   int left1=0, right1=n-1;
06   int left2=0, right2=n-1;
07   while (left1<=right1 && left2<=right2) {
08     int m1=(left1+right1)>>1;
09     int m2=(left2+right2)>>1;
10     int cnt=  ①  ;
11     if (  ②  ) {
12       if (cnt<k) left1=m1+1;
13       else right2=m2-1;
14     } else {
15         if (cnt<k) left2=m2+1;
```

```
16        else right1=m1-1;
17    }
18  }
19  if ( ③ ) {
20    if (left1==0) {
21      return a2[k-1];
22    } else {
23      int x=a1[left1-1], ④ ;
24      return std::max(x, y);
25    }
26  } else {
27        if (left2==0) {
28        return a1[k-1];
29    } else {
30        int x=a2[left2-1], ⑤ ;
31        return std::max(x, y);
32    }
33  }
34 }
```

34. ①处应填(　　)。

A. (m1+m2)*2　　B. (m1-1)+(m2-1)

C. m1+m2　　D. (m1+1)+(m2+1)

35. ②处应填(　　)。

A. a1[m1]==a2[m2]　　B. a1[m1]<=a2[m2]

C. a1[m1]>=a2[m2]　　D. a1[m1]!=a2[m2]

36. ③处应填(　　)。

A. left1==right1　　B. left1<right1

C. left1>right1　　D. left1!=right1

37. ④处应填(　　)。

A. y=a1[k-left2-1]　　B. y=a1[k-left2]

C. y=a2[k-left1-1]　　D. y=a2[k-left1]

38. ⑤处应填(　　)。

A. y=a1[k-left2-1]　　B. y=a1[k-left2]

C. y=a2[k-left1-1]　　D. y=a2[k-left1]

(二)(容器分水)有两个容器,容器1的容量为a升,容器2的容量为b升;同时允许下列的三种操作,分别为:

(1) FILL(i):用水龙头将容器i(i∈{1,2})灌满水;

(2) DROP(i):将容器i的水倒进下水道;

(3) POUR(i,j):将容器i的水倒进容器j(完成此操作后,要么容器j被灌满,要么容器i被清空)。

求只使用上述的两个容器和三种操作,获得恰好c升水的最少操作数和操作序列。上述a、b、c均为不超过100的正整数,且c≤max{a,b}。

试补全程序。

```
#include<bits/stdc++.h>
using namespace std;
const int N=110;

int f[N][N];
int ans;
int a, b, c;
int init;

int dfs(int x, int y) {
  if (f[x][y]!=init)
    return f[x][y];
  if (x==c || y==c)
    return f[x][y]=0;
  f[x][y]=init-1;
  f[x][y]=min(f[x][y], dfs(a, y)+1);
  f[x][y]=min(f[x][y], dfs(x, b)+1);
  f[x][y]=min(f[x][y], dfs(0, y)+1);
  f[x][y]=min(f[x][y], dfs(x, 0)+1);
  int t=min(a-x, y);
  f[x][y]=min(f[x][y], ①  );
  t=min(x, b-y);
  f[x][y]=min(f[x][y], ②  );
  return f[x][y];
}

void go(int x, int y) {
  if ( ③  )
    return;
  if (f[x][y]==dfs(a, y)+1) {
    cout<<"FILL(1)"<<endl;
    go(a, y);
  } else if (f[x][y]==dfs(x, b)+1) {
    cout<<"FILL(2)"<<endl;
    go(x, b);
  } else if (f[x][y]==dfs(0,y)+1) {
    cout<<"DROP(1)"<<endl;
    go(0, y);
  } else if (f[x][y]==dfs(x,0)+1) {
    cout<<"DROP(2)"<<endl;
    go(x, 0);
  }else {
    int t=min(a-x, y);
```

```
44    if (f[x][y]==  ④  ) {
45      cout<<"POUR(2,1)"<<endl;
46      go(x+t, y-t);
47    } else {
48      t=min(x, b-y);
49      if (f[x][y]==  ⑤  ) {
50        cout<<"POUR(1,2)"<<endl;
51        go(x-t, y+t);
52      } else
53     assert(0);
54    }
55  }
56 }
57
58 int main() {
59  cin>>a>>b>>c;
60  ans=1<<30;
61  memset(f, 127, sizeof f);
62   init=**f;
63   if ((ans=dfs(0, 0))==init-1)
64     cout<<"impossible";
65   else {
66     cout<<ans<<endl;
67     go(0, 0);
68  }
69 }
```

39. ①处应填(　　)。

A. dfs(x+t, y-t)+1　　B. dfs(x+t, y-t)-1

C. dfs(x-t, y+t)+1　　D. dfs(x-t, y+t)-1

40. ②处应填(　　)。

A. dfs(x+t, y-t)+1　　B. dfs(x+t, y-t)-1

C. dfs(x-t, y+t)+1　　D. dfs(x-t, y+t)-1

41. ③处应填(　　)。

A. x==c || y==c　　B. x==c && y==c

C. x>=c || y>=c　　D. x>=c && y>=c

42. ④处应填(　　)。

A. dfs(x+t, y-t)+1　　B. dfs(x+t, y-t)-1

C. dfs(x-t, y+t)+1　　D. dfs(x-t, y+t)-1

43. ⑤处应填(　　)。

A. dfs(x+t, y-t)+1　　B. dfs(x+t, y-t)-1

C. dfs(x-t, y+t)+1　　D. dfs(x-t, y+t)-1

答案解析

2019 CCF 非专业级别软件能力认证第一轮(CSP－S)提高级 C＋＋语言试题参考答案及解析

【答案】

一、单项选择题

1	2	3	4	5	6	7	8	9	10
D	C	D	B	B	B	C	B	B	A
11	12	13	14	15					
D	D	B	B	A					

二、阅读程序

1.(1)×　(2)√　(3)√　(4)√　(5)D　(6)A

2.(1)√　(2)×　(3)√　(4)×　(5)C　(6)C

3.(1)√　(2)×　(3)×　(4)×　(5)D　(6)C

三、完善程序

第 1 题					第 2 题				
(1)	(2)	(3)	(4)	(5)	(1)	(2)	(3)	(4)	(5)
C	D	D	C	B	C	B	A	D	D

【解析】

一、单项选择题

1.(int)(x＋y)＝(int)7.2＝7，式子等于 2.5＋7%3＊7%2，右式从左到右算，所以答案是 3.5。

2.本题考查计算机基础知识，WMV，MPEG，AVI 均为视频文件格式，而 JPEG 是图像文件。

3.本题考查位运算的基本知识，按位依次做或运算即可。

4.编译器的作用是将一种语言(通常是高级语言)翻译成另一种语言(通常是低级语言)，这是编译器的功能。

5.其他选项都没有强制转换类型操作，也就意味着数值不会发生变化。如果 x 小数点后三位大于 5，加上 0.5 再截断取整，就会进 1(五入)，否则会舍掉小数部分(四舍)。

6.可以按照包含的数字分类讨论：包含 1，2，4，8 的有 $P_4^4=24$ 种，包含 1，1，8，8 的有 $\frac{P_4^4}{P_2^2*P_2^2}=6$ 种，11××型和 88××型(××为 2，4，8 中任意两种)的有 $C_3^2*\frac{P_4^4}{P_2^2}=36$ 种。

7.本题考查排序算法的基本知识，冒泡排序，直接插入排序和归并排序都属于稳定排序，稳定排

序的定义是排序后值相同的元素的相对位置不会发生变化，而快速排序属于不稳定排序。

8. 本题考查图论的基本知识，8 个点的完全图的边数为 $\frac{8*7}{2}=28$ 条，而如果想让图不连通，至少还需要额外加一个点，故至少有 9 个点。

9. 首先本题只需考虑由 0,1,8,6,9 构成的五位数，如果一个数是 3 的倍数，那么它的数位之和一定也是 3 的倍数。同时由于本题的特殊性质，只需确定前三位就能够确定这么一个数字。那么我们按照第三位数的情况分类讨论：

如果第 3 位数是 0，那么第一位数＋第二位数对 3 取模的余数必须为 0，合法的组合共有 11 种，即(00,99,66,09,90,69,96,06,60,18,81)。

如果第 3 位数是 1，那么第一位数＋第二位数对 3 取模的余数必须为 1，合法的组合共有 7 种，即(10,01,16,61,19,91,88)。

如果第 3 位数是 8，那么第一位数＋第二位数对 3 取模的余数必须为 2，合法的组合共有 7 种，即(80,08,86,68,89,98,11)。

综上所述，总共有 25 种。

10. 由容斥原理可以得知，答案为 15＋12－4＝23。

11. 归并算法的原理：比较当前的 Ai 和 Bi，如果 Ai 更小，则放入 Ai，否则放入 Bi。如果某个数组为空，就把这个数组里的所有元素直接顺次放入新数组，而不需要进行任何比较。因此，如果想使得比较次数尽可能地大，就需要构造两个数组，使得算法运行到最后一个数组为空时另一个数组剩余的元素尽可能少，而最少可以为 1。那么在这种情况下，除了最后一个元素没有被比较以外，其余每一个元素被放入新数组之前都进行了一次比较，故答案为 2n－1。

12. 邻接矩阵和邻接表都是常用的表示图的数据结构。

13. Floyd 算法属于动态规划。其他三个算法多多少少都涉及贪心的思想。

14. $\frac{486}{2}=243$，$\frac{118098}{486}=243$，$243=3^5$，所以选项只有 3 符合要求。

15. 本题考查动态规划的基本知识，为了计算走到当前位置花费，需要枚举从哪里走到当前位置，也就是从两种走法中取一个最大的。随后需要加上当前位置的权值以表示走到当前位置用的代价。

二、阅读程序

1. 分析程序过程可以发现，该程序对于每一个下标 i 求出了一个 ans，表示 a[i]到 a[ans]之间的数字都不大于 a[i]，并且 a[ans＋1]大于 a[i]。

判断题：

(1)考虑一个递增的数组，每一个 ans 都等于 i。

(2)因为有 ans＜n 这一条件，所以一定会小于等于 n。

(3)事实上即使没有第 12 行的条件判断，而每次都将 ans 的初值设置为 i 的话，答案不会发生变化。第 12 行的判断实际上是一种优化：如果当前位置的值不小于上一个值的话，那么求出的 ans 一定不小于上一个求出的 ans。所以只需要在当前值比上一个值小的时候再从 i 开始枚举。

(4)ans 比 i 更大，说明从 i 到 ans 中间的每一个数都不大于 a[i]。

选择题：

(5)如果严格单调递增，那么第 14 行的 while 循环永远不会进入，故复杂度就是 O(n)。

(6)考虑一个严格递减的序列，它永远满足第 12 行的条件，因此每次的 ans 都被设置为 i，但是每次 while 循环都会循环到序列结束，故这种情况下复杂度为 $O(n^2)$。

2.分析程序过程，fa 数组和 getRoot 函数构成了并查集的基本操作，cnt 表示的应该是并查集集合内点的个数。但是需要注意的是，主函数内对两个集合的合并操作并没有判断两个点本来就属于同一集合的情况。

判断题：

(1)并查集初始化的范围是[0，n－1]，所以点的范围也应该如此。

(2)这一步对应的是并查集的初始化操作，必须将 fa[i]初始化为 i。

(3)fa 数组的意义是该集合的祖先对应的点，对应的正好是点的范围。

(4)虽然 cnt 数组的意义是集合的大小，但合并时没有判定两个点本来就属于同一集合的情况，如果两个点来自相同的集合，也会执行一次合并操作，并导致 cnt 数组计算出现重复！

选择题：

(5)每次都是选择两个不同集合的根结点进行合并，我们对于每一个点进行计数，每个点都会产生 n－1＝49 的贡献(因为第 25 行的计数相当于一个集合的每个点都累计上另一个集合的点的个数的贡献)，一共有 n＝50 个点，而且每一对结点的贡献被重复计算了一次，所以答案为 $\frac{50*49}{2}=1225$。

(6)这是一个没有路径压缩的并查集，所以整体复杂度可以达到 $O(n^2)$。

3.判断题：

(1)可知 suf[i]的含义是求出 s[i…slen－1]字符串最多能从右向左匹配到 t 字符串(子序列匹配)的长度。所以 suf[i]从右向左一定是非严格递减的。此外，从代码中也可以看出，第 17 行 suf[i]＝j，j 是没有增加操作的，也可以得出 suf 数组从右向左是递减的。

(2)代入相等的两个串，如 abc 和 abc，可以发现输出为 0。

(3)代入 aaaa 和 bbbb，发现不满足条件。

(4)在 i＝slen－1 的时候不满足条件。

选择题：

(5)s 串的长度此时至少为 1，长度为 0 时会输入不进去。

(6)输出为 2，证明最多可以删掉连续的两个字符还会使 t 是 s 的子序列。s 如果长度小于等于 11 肯定不可以，因为删掉两个数字长度就小于 10 了，t 的长度为 10，不可能是 s 的子串，12 是可以的，例如 s＝aaaaaaaabbaa，t＝aaaaaaaaaa。

三、完善程序

1.本题的关键在于 find 函数，它的基本流程类似于拓扑排序：找到一个入度为 0 并且能够选取的技术，然后从 DAG 上删除这个技术，unlock 维护的就是每一个技术的度数。

(1)此处说明当前技术可学，也就是没有未学完的前置技术，所以应该为 unlock[i]＝＝0。unlock[i]＜＝0 不可以的原因是会把 unlock[i]＝＝－1 的点再处理一次，是不对的，因为这份代码里 unlock[i]＝＝－1 的含义是处理过了。

(2)另一个当前技术可学的前提就是当前得分达到了相关技术的门槛。

(3)如果能学这个技术了，肯定要学掉，获得对应的经验值。

(4)该语句的意思是，当前技术如果被学掉了，相当于它作为先修技能的技术都少了一个先修技能，也就是把 unlock 值都减去 1。

(5)因为 unlock[i] 需要初始化为 i 技能所需的先修技能的个数,即这里输入的 m。

2.本题实际上是一个 DP,下面一步步还原 status,trans 两个变量的意义。实际上可以从 trans 入手:trans 表示的意思应该是对于 i 颗石子,哪些转移是合法的。首先我们记 f[i]表示还剩下 i 颗石子时的必胜/必败情况,那么 f[i]=1 的充要条件就是:所有 i 颗石子取完之后的状态中至少有一个为 0。即所有的 f[i−j]=0,j 为 i 状态下可以取到的所有石子。由于 b[i]只有 64,所以 j 的范围应当小于 64,换言之我们只需要保留 f[i−1]到 f[i−64]的数据就足够推出 f[i]。而且由于 f 的取值只能为 0/1,所以这 64 个 f 的值就可以用一个 64 位二进制数表示,也就是 status 的意义! 对于 status 的第 j 位(从低往高)实际上保存的是当前 f[i−j]的值(最低位从 1 开始)。

(1)此处应该对 DP 数组初始化,即 f[0]=0,其余均为 1,即除了最后 1 位为 0 外全为 1。

(2)此处是维护 trans 的值,i 是单调递增的,a[j]同样也是单调递增,所以当前的石子数大于等于 a[j]后,所有的 i 都可以取 a[j]对应的规则,因此应该为 a[j]==i 开始。

(3)如果满足可取的条件,那么就把取 b[j]石子对应的策略的位设置为 1。

(4)此处是整个 DP 的关键,前面提到 status 保存了 f[i−1]到 f[i−64]的信息,而能够对结果有贡献的位置只有 trans 对应的位置。这些位必须至少有一个 0,当前的 f 才为 1,所以应该是对 status 取反后再与 trans 按位与,得到当前的 f[i]也就是 win。

(5)此处实现的功能为更新 status,也就是把 f[i]添加到 status 的最后一位,于是对应的处理方式就是将 status 整体左移一位,溢出的部分就相当于废弃掉了,然后将 win 添加到 status 的末位。

2020 CCF 非专业级别软件能力认证第一轮(CSP－S)提高级 C＋＋语言试题参考答案及解析

【答案】

一、单项选择题

1	2	3	4	5	6	7	8	9	10
C	B	B	B	D	B	A	A	C	C
11	12	13	14	15					
C	D	B	D	C					

二、阅读程序

1.(1)×　(2)×　(3)√　(4)√　(5)C　(6)C

2.(1)×　(2)√　(3)C　(4)B　(5)A　(6)D

3.(1)√　(2)×　(3)×　(4)D　(5)D　(6)C

三、完善程序

第 1 题					第 2 题				
(1)	(2)	(3)	(4)	(5)	(1)	(2)	(3)	(4)	(5)
D	B	D	D	B	D	B	C	A	B

【解析】

一、单项选择题

1. $(777)_8=551$，$2^{10}=1024$，$(22F)_{16}=559$。

2. 操作系统的作用，A 描述的是接口，C 无联系，D 是编译器。

3. 一帧为 2048 * 1024 * 32/8，总帧数为 2048 * 1024 * 32/8 * 8 * 60 * 24/1024/1024/1024＝90(GB)。

4. 注意问的是栈底，模拟题意可得栈内元素依次为(从栈底到栈顶)a、ab、a、ac、acd、ac。

5. A 中各个数为 4,5,1,5；B 中各个数为 4,3,9,3；C 中各个数为 2,7,10,7；D 中各个数为 1,3,5,9，仅有 D 没有重复数字，不产生冲突。

6. 0—1 背包问题需要用动态规划解决，贪心算法是不正确的，其他均可以直接用贪心解决。

7. 邻接表存储，那么每个点在遍历过程中恰好被访问一次，每条边在有向图中会被访问一次、无向图中会被访问两次，所以时间复杂度为 O(n＋e)。

8. 假设二分图两部分的点数分别为 x,24－x，那么最优情况就是一部分的每个点都和另一部分的所有点连边，边数为 x(24－x)。根据均值不等式有 $x(24-x)<=\left[\frac{x+(24-x)}{2}\right]^2=144$，当且仅当 x＝12 取等号。

9. 广度优先搜索一定会用到队列，属于基础知识。

10. 3,5,7 两两互质，根据中国剩余定理，方程组在模 3 * 5 * 7＝105 意义下有唯一解，不难解

出 n=53。

11. 爬到 x 层总共的热量为 10(1+2+3+…+x−1)=5x(x−1)，令 5x(x−1)>=1000，求得 x 能取的最小正整数为 15。

12. 写一个符号栈，然后模拟一下就行了。

13. 先考虑第一个位置，有 16 种选法，选了一个之后只剩 9 个位置，两个方格没有区别，所以再除以 2，答案就是 16 * 9/2=72。

14. 没优化的 Dijkstra，每次枚举找 n 个点，枚举 n 次。

15. 香农提出了信息熵的概念，为信息论和数字通信奠定了基础。

二、阅读程序

1. 这个题比较正常，双重循环求两个数的和减去他们按位与的值。

判断题：

(1)n=1000 不会发生数组越界。

(2)当所有 d 都相等时，ans 不会改变，依旧为−1。

(3)若输入为严格单调递减，就不能到第 15 行，输出变为−1。

(4)改完后考虑 d[j]>d[i]的情况，跟 d[j]<d[i]在 15 行时求得的答案相同，无影响。

单选题：

(5)如果有比 127 大的，设为 x，因为(d[i]&d[j])不大于 d[i]，d[j]，一定存在一个数 y 满足 y≠x，所以结果也一定比 127 大。

(6)两个偶数与完还是偶数；一奇一偶与完是偶数；两个奇数与完还是奇数。

偶+偶−(偶 & 偶)=偶，奇+奇−(奇 & 奇)=奇，偶+奇−(偶 & 奇)=奇。

2. 这个题目是在找第 k 小的数。

判断题：

(1)若正好随机到 R−L+1 的倍数，取余结果为 0，还是可以到 L 的。

(2)当 L<R 时，12～18 行的循环结束时，a=b 在范围内，没有问题；当 L=R 时，b≤R，不会数组越界。

单选题：

(3)注意第 10 行的交换会破坏原序列递增的性质，所以并不是每一层的交换次数都是 1。可以证明平均交换次数是 $O(\log^2 n)$，具体可以通过分类讨论和归纳法得到，过程较为复杂。

(4)单调递减时，第一层的交换次数和序列长度成正比，第一层的交换次数一定是 O(n)的。在第 10 行选择了 x 之后，有$\frac{x-1}{n}$的概率往长度为 x−1 的一边递归，有 $\frac{n-x}{n}$ 的概率往长度为 n−x 一边递归。

而递归下去之后，序列递减性质可能被破坏，但是每一层交换次数不会超过序列长度，设 N(n)表示长度为 n 的严格递减序列的平均交换次数，M(n)表示任意一个长度为 n 序列的交换次数。

$$N(n)=O(n)+\frac{1}{n}\sum_{i=1}^{n}\left(\frac{i-1}{n}M(i-1)+\frac{n-i}{n}M(n-i)\right)$$

$$M(n)=O(n)+\frac{1}{n}\sum_{i=1}^{n}\left(\frac{i-1}{n}M(i-1)+\frac{n-i}{n}M(n-i)\right)$$

而 M(n)=O(n) 是可以归纳证明的，因此平均交换次数是 O(n)。

(5)假设平均时间复杂度为 T(n)，不难发现时间复杂度和每一层的长度之和成正比。

$$T(n)=O(n)+\frac{1}{n}\sum_{i=1}^{n}\left(\frac{i-1}{n}T(i-1)+\frac{n-i}{n}T(n-i)\right)$$

可以归纳证明 $T(n)=O(n)$。

最差情况之一是每次都选择 x=R，且 k=1，明显时间复杂度是 $O(n)$ 的。

(6)随机并没有用，每一层 a 不变，b 要从 R 到 L，时间复杂度都是 $O(n^2)$ 的。

3. 程序的意思为给定 st0，st1，每次操作可以将 st0 的[0，m]循环右移一位，或是将[m，n－1]循环左移一位。问从 st0 变成 st1 的最少步数。

程序用的是双向广搜实现。注意到 Map，Queue 就是 STL 的 map，queue，但特别的是 Map 查询复杂度是 O(cnt * n)的(string 的比较是 O(n)的)。

判断题：

(1)两个相等直接判断为 0。

(2)st0 有可能[0，100]这一段循环左移一次就变成了 st1，但是循环右移一次不可以。这时 m=0 输出就是 1，m=100 输出就大于 1。

(3)最坏情况下，所有可能的状态都被访问，再算上 Map 中查询时的复杂度，总复杂度就是 $O((n!)^2\times n)$。

单选题：

(4)相当于只能进行[0，99]循环左移的操作，无论如何都不可能将 st0 变成 st1，输出－1。

(5)容易构造出一个次数 $O(n^2)$ 的解，猜想最优答案就是关于 n 的一个二次多项式，待定系数法求解，不难得到 68 的答案。

(6)小于 0 就是无解。C 中，考虑输入与输出有多少对字符前后顺序不一样，我们将这些字符对称为逆序对(不妨假设所有字符互不相同)。不难发现将长度为奇数的区间循环左移/右移，逆序对奇偶性不变；将长度为偶数的区间循环左移/右移，逆序对奇偶性改变。在 C 中，两个能循环移动的区间长度都是偶数，所以如果逆序对个数初始是奇数，无论怎么操作都不能让逆序对变成 0。

三、完善程序

1. (1)性价比更高的在前面，注意类型，要转换为乘法。

(2)看 else 进行推理，也就是 v[1]不超过 B，超过就是 else 对应的特殊情况。

(3)下面的 for 循环是从 2 开始的，已经判断过，第 1 件物品全部都要。

(4)else 的情况就是这块蛋糕没办法整块取完，(B－curV)就是剩余能取的体积，$\frac{B-curV}{v_i}\times w_i+curW$ 就是最后的答案，通分后即为 D 选项。

(5)全部装完，分母为 1。

2. (1)要进行的运算就是去掉 x 二进制的某一位，以数出 x 的二进制有多少个 1。不难发现 D 选项就是令 x 异或上自己的 lowbit(最低的二进制位)。

(2)y 为 a 二进制的低 8 位，x 为 a 二进制的高八位。

(3)此时 Max 并未赋值，给其赋值为 0 即可。注意：v 的初值要取 0 而不是－INF，因为 a[i]作为选取的第一个元素时，42～43 行有可能不更新 v，就要求初值取 v=0 来更新后面的值。

(4)注意低 8 位的权值和高 8 位的权值是可以拆开算的。计算和上一个位置异或值低 8 位的贡献，枚举上一个位置的低 8 位 z，然后用 z∧y 的权值更新。

(5)接下来加上和下一个位置异或值高 8 位的贡献，枚举下一个位置的高 8 位 z，计算高 8 位的权值更新到 Max[z][y]。

2021 CCF 非专业级别软件能力认证第一轮(CSP－S)提高级 C＋＋语言试题参考答案及解析

【答案】

一、单项选择题

1	2	3	4	5	6	7	8	9	10
A	B	A	C	C	C	C	B	D	A
11	12	13	14	15					
A	C	C	C	B					

二、阅读程序

1.(1)√ (2)× (3)× (4)√ (5)D (6)C

2.(1)√ (2)× (3)× (4)B (5)C (6)B

3.(1)× (2)√ (3)× (4)B (5)D (6)D

三、完善程序

第 1 题					第 2 题				
(1)	(2)	(3)	(4)	(5)	(1)	(2)	(3)	(4)	(5)
D	A	D	C	A	D	A	D	D	C

【解析】

一、单项选择题

1.在 Linux 系统终端中，cd、pwd、ls 等命令的作用是更改、显示目录路径和查看目录中的文件。

ls 全拼 list，功能是列出目录的内容及其属性内容信息。

cd 全拼 change directory，功能是从当前工作目录切换到指定的工作目录。

cp 全拼 copy，功能是复制文件或目录。

all 没有这样的命令。

2.二进制加法，1＋1＝10，0＋1＝1，1＋0＝0，0＋0＝0，注意进位。

```
  00101010
+ 00010110
----------
  01000000
```

3.递归函数的函数信息、参数和局部变量都存储在系统栈空间上，因此递归调用的层数过多很可能会导致栈空间溢出。

4.稳定性指的是对需要排序的数据元素序列，将其按关键字进行排序，若相同关键字元素之间的位置关系，排序前与排序后的相对位置不发生变化，称此排序方法满足稳定性，否则称这种排序方法满足不稳定性，堆排序中任何与原始顺序有关的信息都在堆的创建阶段

丢失了，只区分左右孩子，因此不能保证稳定性。

5. 如果用擂台法，每一次用当前数更新最大、小值，需要比较 4n－2 次，将 2n 个数分为 n 对，两个比较大小 1 次，共 n 次。一对数中较大数不需要参与最小值的比较，较小数不需要参与最大值的比较，因此接下来在 n 个数中找最大值，在另外 n 个数中找最小值，共 (n－1) * 2 次，总次数为 3n－2。

6. 根据题意直接模拟即可。每个数被放位置为 0,1,4,9,5,3,3,5，其中最后的 3,5 与前面的数冲突，因此应分别放在 6,7 位置。

7. 假设一个非连通简单无向图有 n 个点，显然将图分成两个连通块是较优的方案，其中一个连通块只有一个点的方案显然最佳，那么最多有 (n－1) * (n－2)/2(孤立一个点)条边。因此 (n－1)(n－2)/2≥36，得到 n≥10。

8. n 层的二叉树最多 2^n-1 个结点，$2^{10}-1=1023<2021<2^{11}-1=2047$，所以该二叉树的高度至少为 11。

9. 前序遍历为根左右，中序遍历为左根右，根肯定要访问，如果存在左子树，中序遍历一定先访问，前序遍历后访问，发生冲突，因此非叶子结点不能有左子树的存在。

10. 可以将字符 A～F 看作 1～6，根据冒泡排序，每交换一次相邻的逆序字符，都会使逆序对个数减 1，因此答案即序列逆序对个数。

11. 认真观察递归函数，可以发现 $\text{solve}(t,n)=5^{t-1} \bmod n$，因此 $\text{solve}(23,23)=5^{22} \bmod 23$。由于 23 是质数，因此可根据费马小定理(p 为质数，满足 $a^{p-1}\equiv 1 \pmod p$)得到 $5^{22}\equiv 1 \pmod{23}$。

12. 考虑一棵递归树，第一层只有 F(n)一个结点，第二层结点为 F(n－1)和 F(n－2)两个，第三层结点为 F(n－2)，F(n－3)，F(n－3)和 F(n－4)四个，以此类推，第 x 层有 2^{x-1} 个结点，而递归树最多有 n－1 层，因此总结点数为 $1+2+4+\cdots+2^{n-1}=2^{n-1}$，时间复杂度为 $O(2^n)$。

13. 设 F(n)为有 n 个苹果时的总方案数，考虑第 n 个苹果的状态：若选择第 n 个苹果，第 n－1 个苹果无法被选，因此此时方案数即为有 n－2 个苹果时的方案数；若不选第 n 个苹果，此时方案数即为有 n－1 个苹果时的方案数，因此，F(n)＝F(n－1)＋F(n－2)。又根据 F(1)＝2，F(2)＝3，因此 F(n)明显为斐波那契数列。最终答案要在斐波那契数列的基础上减去一个苹果都不选的一种方案数。

14. 先计算等腰三角形的三种情况个数，再减去等边三角形 a＝b＝c 的情况个数。如果腰的长度是 x，底的长度必须小于 2x，有 1＋3＋5＋7＋9＋9＋9＋9＋9＝61 种，每个均有 3 种排列。等边三角形 9 个，会被多算两次，答案即为 3 * 61－2 * 9＝165。

15. 此图为 DAG，因此直接设 f(u)为从起点走到 u 点的最短路径长度，按照 BCD->EFG->HI 的顺序手动模拟即可。

二、阅读程序

1. 运用球的体积交公式计算，需要注意小数与整数的混合运算。

判断题：

(1)无论 t 是 int 还是 double，之后计算时都转换为 double，值并没有改变。

(2)除法式左边是一个整数，整数/2 是进行整除，不同于整数/小数的小数除法，会导致答案不同。

(3)sq 的参数会被转为 int，但是 x 和 y 是 double 类型，会导致结果改变。

(4)注意到 $r=\frac{\pi}{3}$，因此可以手动模拟计算。

单选题：

(5)手动模拟计算即可。

(6)容易发现程序求解的是一个三维问题，a，b，c 表示球心坐标，d 表示球半径，AD 排除；观察到第 24 行有可能输出 0，所以不可能是体积并，B 排除；第 24 行的实际意思是两球相离时输出 0，符合体积交的定义，选择 C。

2.本题的任务是采用分治法求最大子段和。一个长度为 n 的完整数组的最大子段和有三种可能性，这个子段全部都在[1，n/2]里面、全部都在[n/2+1，n]里面、子段经过了区间的中点。那么前两种情况可以递归求解，对于第三种情况，其实就是[1，n/2]中最大的后缀和再加上[n/2+1，n]中最大的前缀和。

solve1 函数返回的结构体四个参数描述了函数对应区间的数值特征，分别代表 h 此区间最大的前缀和、j 此区间最大的子段和、m 此区间最大的后缀和、w 此区间所有元素的总和。

solve2 函数则直接通过循环计算当前区间“经过中点”的最大子段和，再与递归计算的前两种方案求最大值，可以发现，这样并不会遗漏情况，同样能够求出当前区间的最大子段和。

判断题：

(1)solve1 与 solve2 明显为求序列最大子段和的两个不同的正确解法，都可以输出正确答案。

(2)当 n>0 时，这两行都不会被执行，或者这两行各执行一次。

(3)手动模拟计算即可，答案为 11。

单选题：

(4)因为 T(n)=2T(n/2)，所以 T(n)=O(n)。

(5)因为 T(n)=O(n)+2T(n/2)，所以 T(n)=O(n logn)。

(6)手动模拟求最大子段和即可。

3.程序新定义了一个字符串加密规则，实现了这种规则下的编码和解码功能。

init 函数的功能显然，规定了每个加密数字对应的加密字符。需要注意的是，其中 base[i]表示加密数字 i 对应的加密字符，table[i]表示加密字符 i 对应的加密数字，两者互为逆运算。

encode 函数对输入的字符串进行加密，每三个字母会加密成 4 个字母，不足的在后面补“=”，按照一定的规则和二进制计算把三个字母的 ASCII 码组合成 0～63 以内的 4 个加密数字，然后再用 base 得到每个数字对应的加密字符。

decode 函数相反，每次读取 4 个加密字符，用 table 得到他们原来的加密数字，然后做反向的二进制计算得到原来三个字母的 ASCII 码。

要求对 ASCII 码、不同进制、类型之间的转换规则足够熟悉。

判断题：

(1)decode 是解密函数，过程中有可能任何字符，如果出现'\n'，就不是一行了。

(2)decode 过程与 encode 过程互为逆过程。

(3)1 表示调用 decode 函数，手动模拟即可。

单选题：

(4)encode 函数只有一个循环,时间复杂度显然为 O(n)。

(5)table[0]初始值为 0xff(十进制下的 255),在 char 中表示－1(char 只能表示－128～127,因此 0xff 对应－1),强转为 int 后为－1。

(6)长度%3==1,因此末尾有两个=,排除 AC,然后模拟 BD 选项不一样的部分就可以了。

三、完善程序

1.(1)该部分为初始化部分。

(2)若 n 还没算出最优值,就继续算。

(3)寻找没有算出最优值的数中的当前答案最优的松弛。

(4)没有算出最优值才能继续下一步操作。

2.程序的目的是采用笛卡尔树解决 RMQ 问题,细分如下几步:

- 建立笛卡尔树。
- 使用 DFS 求欧拉序。
- 求分块后块与块之间的最小值,解决块内最小值。
- 完成预处理工作后依次回答询问。

(1)A,构建笛卡尔树,连接当前结点的左孩子。

(2)D,构建笛卡尔树,当前结点链接到断开位置的右孩子。

(3)A,这是在树上求 LCA,因此返回更浅的结点。

(4)D,笛卡尔树的性质保证深度越浅的点 val 越大。

(5)D,在对应位置的程序中可以发现,如果 mask 的第 i－1 为 1,代表 a[i]=a[i－1]－1。

(6)C,从第 89 行可以看出,在 mask 中,较低位在二进制的左边,因此右移的位数与 l 有关。

2022 CCF 非专业级别软件能力认证第一轮(CSP－S)
提高级 C＋＋语言试题参考答案及解析

【答案】

一、单项选择题

1	2	3	4	5	6	7	8	9	10
B	A	D	C	A	B	C	B	D	A
11	12	13	14	15					
C	D	B	B	B					

二、阅读程序(共 3 题,除特殊说明外,判断题 每题 1.5 分,选择题每题 3 分,共计 40 分)

(一) **16.**√ **17.**× **18.**√ **19.**D **20.**A **21.**B

(二) **22.**× **23.**× **24.**√ **25.**D **26.**D **27.**C

(三) **28.**√ **29.**× **30.**× **31.**A **32.**B **33.**B

三、完善程序(共 10 题,每题 3 分,共计 30 分)

(一)					(二)				
34	35	36	37	38	39	40	41	42	43
D	B	C	C	A	A	C	A	A	C

【解析】

一、单项选择题

1. cd 即 change directory 的首字母缩写,意为切换工作目录。ls 查看文件和目录,cp 复制目录或文件,all 不是完整指令。

2. real 即实际使用时间;user 即用户时间;sys 即系统时间。这里秒表统计的是总时间。

3. 由于子串“fedcb”,D 选项连续退栈操作需要 5 次。

4. 归并排序,每次划分区域 O(logn),每个数在两组之间比较排序,n 个数遍历,最坏时间复杂度均为 O(nlog n),显然低于 $O(n^2)$。而其余排序最坏时间复杂度均为$O(n^2)$,如在某些情况下快速排序会退化。

5. 与其他基于比较操作的排序算法不同,基数排序是以桶排序思想为基础的,即使某一个值发生了改变,只要排除这个数据,其余数据在排序后仍是有序序列。

6. DE 是高位字节,EF 是低位字节,结合题设:小端模式下的首字节是 EF,大端模式的首字节是 DE。

7. 由于按照前序遍历(根－左－右)的顺序编号,因此每棵子树的结点编号是连续的,根节点是 1 号结点,深度为 1。根结点的第一个儿子是 2 号结点,并且 2 号结点为根的子树一定是一棵深度为 4 的满三叉树,这个满三叉树有 1＋3＋9＋27＝40 个结点。所以根结点的

第二个儿子是 42 号，同理，根结点的三号儿子是 82 号。又因为 82 号的第一个儿子是 83 号，并且以 83 号为根的子树有 1＋3＋9＝13 个结点，因此 82 号的第二个儿子是 82＋13＋1＝96 号，96 号的第一个儿子是 97 号，而 97 号的其中一个儿子是 100 号。

8. 在有向图 G 中，对于每一对结点，互相之间都存在路径，则称 G 是强连通图。而路径可以由很多条边组成，显然不一定要满足 B 选项，B 选项是完全图的条件。

9. 设想一下，每个点都要且只能与两个点相连，这很显然只能是一个环，由于环上可以从任意点出发，我们设起点是 1 号点，然后枚举其他点的顺序，总共得到(n－1)！种方案。但是对于无向图的环来说，两个不同的序列会对应同一个环的顺时针与逆时针，因此总的数量是$\frac{(n-1)!}{2}$。

10. 不区分两人的角色和作用，因此应该是组合数$C_8^2=28$。

11. 大写字母 26 个，阿拉伯数字 10 个，根据乘法原理有 26 * 26 * 10 * 10 * 10＝676000。

12. 在 89 存储之前的哈希表如下：71－1 23－3 73－4 99－9 44－5 79－0

h(89)＝9，从地址 9 开始查找第一个空的地址，可以找到 2 号地址。

13. 第 3 行 j＝0 需要更正为 j＝1。更正后，j 每次翻倍，直到超过 n，共运 log n 次，因此总时间复杂度为 O(nlog n)。

14. 使用擂台法，最坏情况下需要进行 n－1 次比较。

15. 代入计算，计算的同时可以将计算过的函数值记录，避免重复计算。

二、阅读程序

(一)

判断题：

16. 不难看出这个程序的目的是在 s 串中寻找 t 串，进行字符串匹配，shift 数组将 i 快速跳转到下一个可能的位置。abcde 中没有 fg，匹配不成功，输出－1。

17. 第 18 行，字符串下标从 0 开始，因此查找的结果是 3。

18. 模拟可得：只在 i==13 时执行过 j++，共 2 次；i==11 是并不会执行到 j++语句。由于字符串 t 中只有字符'2'，在 s[i＋m]不等于'2'时，i 会直接加上 m＋1，而 m＝2，因此 i 依次取值 0、3、6、9、12，此时 s[14]等于'2'，所以下一个 i 是 13，然后 s[13＋0]==t[0]，s[13＋1]==t[1]，最终得到答案 13。

单选题：

19. 最简单的 hack 方式：s＝"aaa…aaa"，t＝"aaab"。每次需要判断到第 m－1 个字符才知道不匹配，并且每次位移 i 只能＋1 或者＋2，此时时间复杂度可以达到 O(nm)。

20. 在 s 字符串中找到 t 字符串第一次出现的位置。

21. 根据程序，可能的 i 的取值分别是 0、5、6、7、8、13，其中 i＝5 时，j++运行 3 次；i＝6 时，j++运行 2 次；i＝7 时，j++运行 1 次；i＝13 时，j++运行 4 次；其余的情况均不可能。因此共 3＋2＋1＋4＝10 次。

(二)

首先需要理解 init 函数，在读入 val[]数组之后，求出数组的最大值 maximum，然后将 maximum 不停地除以 k，同时 m++，说明 m 是 k 进制下的位数，此时是在求 val[]在 k 进制下的最长位数。

接下来，在 solve( ) 函数中，第 28 到 35 行是非常经典的基数排序的思路，而 val[]/base%k的语句很容易看出是在 k 进制下提取出某一位，再结合 m 的意义，可以发现每次排序都是按照 val 在 k 进制下的第 i 位从小到大排序的，最终会实现对整个val[]数组的排序。

判断题：

22. 该程序是 k 进制下的基数排序，基数排序是一个稳定的排序算法。排序算法的稳定性，通俗地讲就是能保证排序前 2 个相等的数在序列的前后位置顺序和排序后它们两个的前后位置顺序相同。

23. cnt 数组的下标范围是 0～k－1，空间复杂度与 k 也有关系，是 O(n＋k)。

24. 第 28、30 行循环运行 k 次，第 29、31、35 行循环运行 n 次，最外层循环运行 m 次，时间复杂度为 O(m×(n＋k))。

单选题：

25. 第一次执行时，数字是按照三进制下的最低位从小到大排序，其中 98、26 三进制下最后一位是 2，91、37、46 三进制下最后一位是 1。由于基数排序是稳定的，所以当前位相同时顺序不变，得到 D 选项。

26. 时间复杂度为 O(m×(n＋k))，当 k 取 10，m＝2，当 k 取 100 时，m＝1。但是时间复杂度还和 n 有关：当 n＝10 时，k 取 10 更优；当 n＝100 时，k 取 100 更优，因此 k 的最优取值与 n 有关。

27. 当 k 大于所有 val[i]时，得到的是标准的计数排序。需要注意的是，计数排序是统计每个数字的出现次数，而桶排序的核心思想就是将要排序的数据按规则分类存储，分到几个有序的桶里，每个桶里的数据再单独进行排序。桶排序完之后，再把每个桶里的数据按照顺序依次取出，组成的序列就是有序的了。

(三)

通过第 19、20 行，可以发现 n 在不断除以 k，并且使用 ans[]存下每次模 k 的余数，这是经典的 k 进制转换模型；输出部分，0 到 9 直接输出，10 以上转换成大写字母再输出，很明显这道题是在做进制转换。第 19 行对负数取模可以直接理解成对 k 取模，功能相同。

但是，这道题的第 20 行并非正常的进制转换操作，它调用值是 ans[m－1]－n，当原本的 n 是正数时，新的 n 会得到负数；当 n 是负数时，新的 n 是正数并且绝对值会变大以后再除以 k，因此结果不是真正的进制转换，在一些情况下，得到的位数会比进制转换更多。

判断题：

28. ans[m－1]一定小于 k，因此每次循环后 n 的绝对值都会除以 k，直到绝对值变成 0。

29. 如果 ans[i]＞＝10，程序将输出‘A’到‘Z’的大写字母，如果去掉强制类型转换，那么程序会输出数字而不是字母。

30. 这道题并非正常的进制转换操作，比如当 n 是负数时，ans[m－1]－n 比原有的 n 绝对值更大，因此运行次数会相对增加。

单选题：

31. 根据程序模拟，手动计算即可。

32. 根据程序模拟，手动计算即可。

33. 根据程序模拟，手动计算即可，需要注意字母的转换细节。

三、完善程序

（一）

34. 在这个函数中，二分是为了确定 a1[]和 a2[]中哪些部分是属于合并后前 k 个数字的。所以 cnt 代表的是 a1[0..m1]和 a2[0..m2]中数字的个数，也就是合并数组当前位置的下标，同时，需要考虑到数组下标是从 0 开始的。

35. 这是为了比较两个数组对应的中间位置的大小关系。观察第 12、13 行，cnt＜k 时，舍弃 a1[left1..m1]子段；cnt＞＝k 时，舍弃 a2[m2..right2]子段。说明此时 a1[left1..m1]在归并之后一定在 a2[m2..right2]前面，又因为从小到大排序，那么 a1[m1]必须小于等于a2[m2]。

36. 观察程序，此处 if 语句会退出该函数，在程序的第 7 行，循环结束的条件是 left1＞right1 或者 left2＞right2 此时 if 语句是这两种情况中的某一类，显然，是 C.left1＞right1。

37. 在程序的第 23 行，a1 数组的前 left1 个数（注意下标从 0 开始）组成 k 个数，a2 数组组成 k 个数的个数应该是 k－left1，需要注意的依旧是下标从 0 开始。

38. 与④同理，在程序的第 30 行，a2 数组的前 left2 个数组成 k 个数，a2 数组组成 k 个数的个数应该是 k－left2，需要注意下标从 0 开始。

（二）

39. 这里是把容器 2 里面的水倒往容器 1，所以 x＋t 同时 y－t。

40. 这里是把容器 1 里面的水倒往容器 2，所以 y＋t 同时 x－t。

41. go()函数的功能是输出方案，判断是否已经找到 c 升的水，如果当前已经得到了 c 升水就不需要再查找方案了。

42. 判断下一步是否为将容器 2 中的水倒入容器 1 中。

43. 与④同理，判断下一步是否为将容器 1 中的水倒入容器 2 中。

参考文献

[1] 曹文，董永建，吴涛.信息学奥赛一本通·初赛篇[M].南京：南京大学出版社，2018.

[2] 张子臻，王晓鹏，刘溯.信息学竞赛金牌导航·初赛篇[M].广州：华南理工大学出版社，2020.

[3] 董永建.信息学奥赛一本通 C++版[M].南京：南京大学出版社，2020.

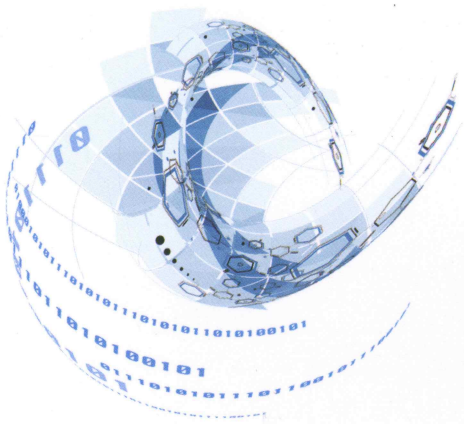
[4] 曹文，吴涛.全国青少年信息学奥林匹克分区联赛初赛培训教材[M].杭州：浙江大学出版社，2011.

[5] 胡旭红.全国青少年信息学竞赛培训教材·初赛[M].杭州：浙江大学出版社，2011.

[6] 丁向民.CCF CSP 第一轮认证一本通[M].北京：清华大学出版社，2021.

[7] 丁向民，王雄壮，缪佳丽.信息学奥赛真题分类解析·初赛篇[M].北京：清华大学出版社，2020.

[8] 陈文博，常强，陈跃坚.信息学奥林匹克竞赛初赛精讲精练[M].北京：机械工业出版社，2021.



一本通

ISBN 978-7-305-25726-1



9 787305 257261 >

定价：79.80元

责任编辑：纪玉媛
设计人员：宇　轩