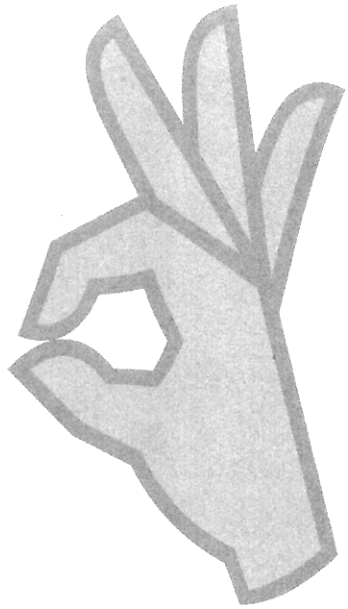
青少年学科奥林匹克竞赛丛书

全国青少年信息学（计算机）奥林匹克分区联赛试题解析（中学）

总策划 王晓敏
主 审 李立新
主 编 章维铣
副主编 曹 文
吴再陵

青少年学科奥林匹克竞赛丛书

全国青少年信息学（计算机）奥林匹克分区联赛试题解析（中学）



总策划　王晓敏
主　审　李立新
主　编　章维铣
副主编　曹　文
　　吴再陵

青少年学科奥林匹克竞赛丛书

编委会

序

国际信息学奥林匹克(International Olympiad in Informatics)简称(IOI),从1989年到2000年,12年赛事的健康发展得益于联合国教科文组织(UNESCO)为这项赛事所做的准确定位:通过竞赛形式对有才华的青少年起到激励作用,促其能力得以发展;让青少年彼此建立联系,推动经验交流,给学校这一类课程增加活力;建立起教育工作者与专家档次上的国际联系,推进学术思想的交流。概括起来说,就是:启迪思路,激励英才,发展学科,促进交流。

学科奥林匹克是智力与能力的竞赛,注重考查全面素质与创造能力。从这个意义上讲,信息学奥林匹克活动是素质教育的一个大课堂。在我国,每年国家集训队都要将"怎样做人,怎样做事,怎样求知和怎样健体"的指导思想纳入培训计划。12年中国队共派出参赛选手47人次,累计获金牌25块、银牌12块、铜牌10块,届届名列前茅,正是因为坚持了全面素质教育的指导思想,把造就高素质、有创造精神的人才作为活动的定位目标。

回顾12年赛事可以看出,参加高手云集的这种世界大赛是有相当难度的:

(1) 没有大纲,赛题范围没有界定,谁也无法去猜测每年的主办国会出什么类型的难题;

(2) 计算机科学与技术发展很快,层出不穷的新思路和新成果会反映到试题中来;

(3) 所要解决的试题往往涉及图论、组合数学、人工智能等大学开设的课程知识;

(4) 比较短的给定解题时间与刁难的测试数据让选手必须拿出高超和精巧的解法,无论在时间上还是空间上都是优化的解法才能取得高分。有许多赛题没有固定的现成的解法,选手要在比赛现场凭借实力,理出思路,构建数学模型,写出算法,编出程序,运行并验证整个构思是否正确,出解的时间是否能达到题目的要求,等等。

可以看出,在这一过程中最重要的是创造能力,我们为激发创新精神,培养创造能力,需要树立新的教育观念和教学方法,还要利用现代化的教学手段。引导学生学用电脑,在使用中帮助开发人脑,这可能是信息学奥林匹克活动的最重要的一个特点。我认为在这项活动中应该培养学生的四种能力:自学能力;实践动手能力;创新能力;上网获取知识,并能区分有用知识和无用知识的能力。这样做的结果使许多选手不但有能力在世界赛场上拿金牌,也有能力在学校的学习中名列前茅。

信息学奥林匹克10年涌现出一大批出类拔萃的计算机后备人才,在他们的带动下,我国的青少年在普及计算机的大潮中阔步前进,取得了可喜的成绩。历史已雄辩地证明:计算机的普及就是要从娃娃做起,这是"科教兴国",中华崛起的需要。为了进一步推动普及,在教委、科协部门的领导下,从1995年起江苏省科协青少年科技中心受中国计算机学会的委托,已连续6年成功组织承办了全国的奥林匹克分区联赛活动,数以万计的青少年从中受益。在这6年中,参与这项活动的老师与专家积累了许多宝贵经验,撰写了《全国青少年信息学(计算机)奥林匹克分区联赛试题解析(中学)》和《青少年信息学(计算机)奥林匹克竞赛小学试题解析》两本书。以算法分析为主线,剖析6届联赛的试题,讲思想、讲方法,侧重基础训练,引导

学生在解题编程的实践中掌握科学思维的方法,提高使用计算机的能力。这两本书可作为广大青少年参与信息学奥林匹克活动的培训教材,我相信这一定会对信息技术的普及起到推波助澜的作用。

青少年是国家的希望,大力开展青少年计算机普及教育活动,必将为我国21世纪在科技、经济、文化各个领域的腾飞,准备好一支用高技术武装起来的建设者队伍。因而,不断提高青少年的科学素养是中华民族永远昂首屹立在世界东方的根基所在,从现在的青少年抓起是大有必要的,也是大有希望的。"精心育桃李,切望青胜蓝"是我和编写这两本书的老师们的共同心愿。

国际信息学奥林匹克中国国家队科学委员会主席
清华大学计算机系教授,博士生导师
吴文虎
2001.4.15

前　言

全国青少年信息学(计算机)奥林匹克及其分区联赛(简称 NOI),是经中国科协、国家教育部批准,由中国计算机学会主办的一项全国性的青少年学科竞赛活动。以在青少年中普及计算机科学为宗旨的 NOI,其成功举办,激发了广大青少年对计算机及其应用的兴趣,开阔了眼界,扩大了知识面,培养了他们的逻辑思维、创造思维以及应用计算机解决实际问题的能力。为了进一步扩大普及面,从 1995 年起 NOI 竞赛予以延伸,组织了首届全国分区联赛,到 2000 年共举办了 6 届。为了使广大青少年了解 NOI 分区联赛的内容和要求,我们根据分区联赛的竞赛大纲编写了这本既可用于课堂教学、又适合各个层次青少年自学的《全国青少年信息学(计算机)奥林匹克分区联赛试题解析(中学)》,奉献给广大青少年计算机爱好者和他们的辅导老师。

本书分基础篇和提高篇两部分,分别对应于全国分区联赛的初赛与复赛试题。紧密围绕分区联赛竞赛大纲所涉及的知识点,以算法分析为主线,剖析了从首届联赛至今 6 届的有关试题,讲思想,讲方法,着重基础训练,引导青少年读者在实践中掌握相关的思想和科学思维方法,以期提高参赛选手的综合能力。

本书由章维铣主编,基础篇第一、二章由吴再陵编写,基础篇第三章、提高篇第一、二章由章维铣编写,提高篇第三、四、五、六章由曹文编写。全书由李立新主审。本书本编写过程中得到了江苏省青少年计算机教育活动中心王晓敏、赵聆老师的大力帮助和支持。本书所引用的试题浸透着多年来积极参与青少年信息学奥林匹克分区联赛命题工作和活动的专家教授的心血和劳动,许多参赛选手的思想方法和解题技巧给予我们极大的启发和借鉴。在此一并表示由衷的谢意。

由于编者水平有限,书中难免会有错误和不妥之处,恳请读者批评指正。

编　者

2001.4.15

目　录

全国青少年信息学(计算机)奥林匹克分区联赛试题解析(中学)

序 …… 吴文虎

基　础　篇

第一章　分区联赛基础部分 …… 2
　第一节　计算机基础知识 …… 2
　第二节　数据结构与算法 …… 19
第二章　阅读程序并写出程序的运行结果 …… 64
第三章　完善程序 …… 87
　第一节　1995 年试题 …… 87
　第二节　1996 年试题 …… 101
　第三节　1997 年试题 …… 110
　第四节　1998 年试题 …… 123
　第五节　1999 年试题 …… 129
　第六节　2000 年试题 …… 136

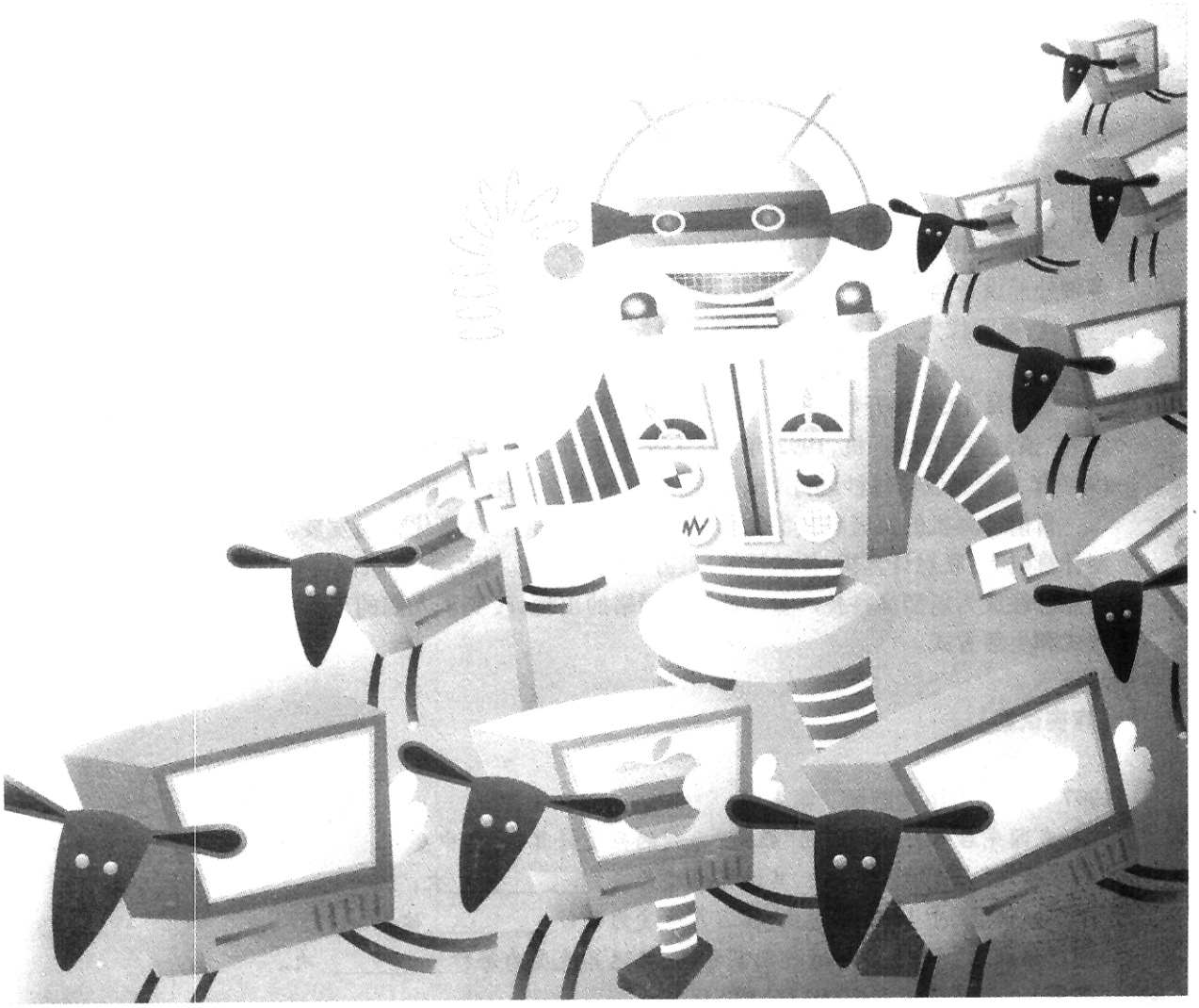
提　高　篇

第一章　1995 年复赛试题解析 …… 144
第二章　1996 年复赛试题解析 …… 187
第三章　1997 年复赛试题解析 …… 210
第四章　1998 年复赛试题解析 …… 236
第五章　1999 年复赛试题解析 …… 255
第六章　2000 年复赛试题解析 …… 274

第五届

基　础　篇

- 分区联赛基础部分
- 阅读程序并写出程序的运行结果
- 完 善 程 序



第一章　分区联赛基础部分

第一节　计算机基础知识

【背景知识】

计算机的产生和发展

计算机的产生是20世纪最重要的科学技术大事件之一。1946年美国宾夕法尼亚大学经过几年的艰苦努力，研制出世界上第一台数字电子计算机——埃尼阿克（“ENIAC”）。自从第一台电子计算机诞生至今的50多年内，电子计算机以异常迅猛的速度发展，到目前为止，计算机发展大致经历了四代：

① 第一代电子管计算机，开始于1946年，结构上以CPU为中心，使用机器语言，速度慢、存储量小，主要用于数值计算；

② 第二代晶体管计算机，开始于1958年，结构上以存储器为中心，使用高级语言，应用范围扩大到数据处理和工业控制；

③ 第三代中小规模集成电路计算机，开始于1964年，结构上仍以存储器为中心，增加了多种外部设备，软件得到一定发展，计算机处理图象、文字和资料功能加强；

④ 第四代大规模和超大规模集成电路计算机，开始于1971年，应用更加广泛，出现了微型计算机。

我国从1956年开始电子计算机的科研和教学工作，1983年12月研制成功每秒运行1亿次的“银河”巨型计算机，1992年11月研制成功每秒运行10亿次的“银河Ⅱ”巨型计算机，1997年又研制成功每秒运行130亿次的“银河Ⅲ”巨型计算机。

目前计算机的发展向微型化和巨型化、多媒体化和网络化方向发展。我国比较著名的微型计算机有：联想计算机、长城计算机、方正计算机等。由于计算机向网络化发展，计算机通信产业已经成为新型高科技产业。计算机网络的出现，改变了人们的工作方式、学习方式、思维方式和生活方式。

【竞赛试题】

1997年　初中组基础题　第1题

我国先后自行研制成功“银河”系列的巨型计算机，其中：

“银河”于1983年问世，其运算速度为每秒_____次；

“银河Ⅱ”于1992年诞生，其运算速度为每秒_____次；

“银河Ⅲ”于1997年通过国家鉴定，其运算速度为每秒_____次。

【答案】 依次填入1亿、10亿、130亿。

【背景知识】

计算机的系统及工作原理

1. 计算机系统组成

计算机系统是由硬件和软件两部分组成的。硬件是指构成计算机的电子元器件即计算机的设备;软件是指程序和有关的文档资料。

(1) 计算机的主要硬件

输入设备:常见有键盘、鼠标、扫描仪等。

输出设备:常见有显示器、打印机、绘图仪等。

中央处理器:又称为CPU,它包括运算器、控制器。运算器可以进行算术运算和逻辑运算;控制器是计算机的指挥系统,它的操作过程是取指令——分析指令——执行指令,循环执行。

存储器:具有记忆功能的物理器件,用于存储信息。存储器分为内存和外存。

① 内存是半导体存储器:它分为只读存储器(ROM)和随机存储器(RAM);

② 外存:磁性存储器——软盘和硬盘;光电存储器——光盘,它们可以作为永久性存储器;

③ 存储器的两个重要技术指标:存取速度和存储容量。内存的存取速度最快(与CPU速度相匹配),软盘存取速度最慢。存储容量是指存储的信息量,它用字节(BIT)作为基本单位,1字节用8位二进制数表示,1KB = 1024B, 1MB = 1024KB, 1GB = 1024MB

(2) 计算机的软件

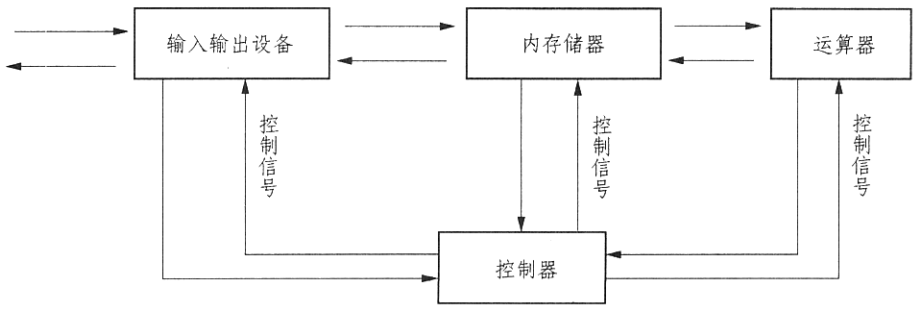
计算机的软件主要分为系统软件和应用软件两类:

① 系统软件:为了使用和管理计算机的软件,主要有操作系统软件如,WINDOWS 95/98/2000/NT4.0、DOS 6.0、UNIX 等;WINDOWS 95/98/2000/NT4.0是多任务可视化图形界面,而DOS是字符命令形式的单任务的操作系统。

② 应用软件:为了某个应用目的而编写的软件,主要有辅助教学软件、辅助设计软件、文字处理软件、工具软件以及其他的应用软件。

2. 计算机的工作原理

到目前为止,电子计算机的工作原理均采用冯·诺依曼的存储程序,并自动完成程序的设计思想。其工作过程如下图所示:



需要注意的是:程序中的数据、指令都采用数字化编码方式,保存在存储器中;程序中的指令必须是属于这台机器的指令系统。

3. 计算机病毒

计算机病毒是一种程序,是人为设计的具有破坏性的程序。它往往使计算机不能正常工作。计算机病毒具有破坏性、传播性、可激发性、潜伏性、隐蔽性等特点。由于计算机病毒危害极大,需要注意隔离计算机病毒的来源,经常用杀病毒软件检查计算机系统和存储器。

【竞赛试题】

1997 年　初中组基础题　第 2 题

下列软件均属于操作系统的是:_____

(A) WPS 与 PC DOS　　(B) WINDOWS 与 MS DOS

(C) WORD 与 WINDOWS　　(D) FOXBASE 与 OS/2

【答案】 正确答案选择(B),因为 WPS、WORD、FOXBASE 都不是操作系统软件。这里 WPS、WORD 都是文字编辑处理软件,是应用软件,而 FOXBASE 是数据库系统软件。

1997 年　高中组基础题　第 1 题

WPS 是属于_____类的软件;FOXBASE 是属于_____类的软件。用 FOXBASE 的命令:"CREATE GZB",在磁盘中生成的是_____文件。

【答案】 在第一个空内填入"应用",第二个空内填入"系统",第三空内填入"GZB. DBF",数据库文件的扩展名为"DBF"。

1998 年　初中组基础题　第 1 题(高中组基础题第 1 题)

操作系统是重要的系统软件,下面几个软件中不属于操作系统的是_____

(A) MS-DOS　　(B) UCDOS　　(C) PASCAL　　(D) WINDOWS 95

【答案】 正确答案选择(C),因为 PASCAL 语言是程序设计的语言系统软件。

1998 年　初中基础题　第 2 题

MS-DOS 系统对磁盘信息进行管理和使用是以_____为单位的。

(A) 文件　　(B) 盘片　　(C) 字节　　(D) 命令

【答案】 正确答案选择(A),因为我们对磁盘信息的存取,必须以访问文件方式进行。这里主要考察学生对 MS-DOS 文件的存储结构的理解。

1998 年　初中组基础题　第 3 题(高中组基础第 2 题)

在计算机内部用来传送、存贮、加工处理的数据或指令(命令)都是以_____形式进行的.

(A) 十进制码　　(B) 智能拼音码　(C) 二进制码　　(D) 五笔字型码

【答案】 正确答案选择(C),这里主要考察学生对计算机工作原理的理解。计算机内部无论是数据还是命令都需要转换成二进制代码才能传送、存贮、加工处理。

1999 年　初中组基础题　第 1 题(高中组基础第 1 题)

微机内的存储器的地址是以(　　)编址的。

(A) 二进制位　　(B) 字长　　(C) 字节　　(D) 微处理器的型号

【答案】 正确答案选择(B),字长表示一个存储单元由多少位二进数组成,八位机一个字长就是一个字节,十六位机一个字长可以表示两个字节。字长位的多少,表明可访问存储器的地址多少。

1999 年　初中组基础题　第 2 题(高中组基础第 2 题)

下列诸因素中,对微机工作影响最小的是(　　)

(A) 尘土　　(B) 噪声　　(C) 温度　　(D) 湿度

【答案】 正确答案选择(B),因为尘土、温度、湿度对微机工作都有较大的影响。

1999 年　初中组基础题　第 3 题(高中基础第 3 题)

在 24 * 24 点阵的字库中,汉字"一"与"编"的字模占用字节数分别是(　　)

(A) 32、32　　(B) 32、72　　(C) 72、72　　(D) 72、32

【答案】 正确答案选择(C),这是因为在汉字编码中,每个汉字无论笔画多少,它们字模所占的字节数总是相同的,这样就排除 B、D;每行 24 点需用 3 个字节存储,24 行则需要 72 个字节,所以选择(C)。本题主要考察学生对汉字字模以及字模存储结构的理解。

1999 年　初中基础题　第 4 题(高中组第 4 题)

将 DOS 系统盘插入 A 驱动器启动机器,随后使用一批应用软件,在此过程中,DOS 系统盘(　　)

(A) 必须始终插入在 A 驱动器中　　(B) 不必再用

(C) 可能有时要插入 A 驱动器中　　(D) 可能有时要插入 B 驱动器中

【答案】 正确答案选择(C),因为操作系统是为用户提供使用和管理计算机的软件,一旦启动成功后,常用的命令驻留在内存中,此时用户可以完成自己的应用软件,一般不需要将系统盘插入在 A 驱中,若需要调用操作系统中的外部命令,则需要将系统盘插入在 A 驱中。

1999 年　初中基础题　第 7 题(高中基础题第 7 题)

计算机能直接执行的指令包括两部分,它们是(　　)

(A) 源操作数与目标操作数　　(B) 操作码与操作数

(C) ASCII 码与汉字代码　　(D) 数字与字符

【答案】 正确答案选择(B),因为计算机的指令系统是由操作码与操作数组成。

1999 年　初中基础题　第 8 题(高中基础题第 8 题)

在微机中,通用寄存器的位数是(　　)

(A) 8 位　　(B) 16 位　　(C) 计算机字长　　(D) 32 位

【答案】 正确答案选择(C),通用寄存器的位数跟机器有关,它取决于计算机字长。

1999 年　初中基础题　第 9 题

在计算机中,ASCII 码是(　　)位二进制代码

(A) 8　　(B) 7　　(C) 12　　(D) 16

【答案】 正确答案选择(B),7 位二进制可以表示 2^7 个状态,因此有 128 个不同的二进制编码,国际上均按照这样的编码来表示控制符号、十进制数、字符、英文字母的大小写以及一些特殊符号等。由于一个字节长度是八位二进制数,所以用一个字节表示 ASC Ⅱ码,则最高位设置为 0;中国的汉字编码是用两个字节表示,为了区别 ASC Ⅱ代码,所以每个字节的最高位设置为 1。本题考察学生对 ASC Ⅱ码的理解。

1999 年　初中基础题　第 10 题(高中基础题第 10 题)

计算机的软件系统通常分为(　　)

(A) 系统软件与应用软件　　(B) 高级软件与一般软件

(C) 军用软件与民用软件　　(D) 管理软件与控制软件

【答案】 正确答案选择(A),本题考察学生对软件系统的知识的理解。

1999 年　初中基础题　第 16 题(高中基础题第 16 题)

启动计算机引导 DOS 是将操作系统(　　)

(A) 从磁盘调入中央处理器　　(B) 从内存储器调入高速缓冲存储器

(C) 从软盘调入硬盘　　(D) 从系统盘调入内存储器

【答案】 正确答案选择(D),因为计算机引导系统后,将系统中常用命令驻留在内存中,方便用户使用和管理计算机。

1999 年　初中基础题　第 19 题(高中基础题第 19 题)

不同的计算机,其指令系统也不相同,这主要取决于(　　)

(A) 所用的操作系统　　(B) 系统的总体结构

(C) 所用的 CPU　　(D) 所用的程序设计语言

【答案】 正确答案选择(C),因为 CPU 包括运算器、控制器,所有的控制和运算操作,均由控制器中的微指令进行操作。

2000 年　初中基础题　第 2 题(高中基础题第 2 题)

在外部设备中,绘图仪属于(　　)

(A) 输入设备　　(B) 输出设备　　(C) 辅(外)存储器　　(D) 主(内)存储器

【答案】 正确答案选择(B),本题考察学生对计算机硬件的理解和认识。绘图仪是受计算机控制,将处理信息的结果,以绘出图形的方式表示出来的一种外部设备。

2000 年　初中组基础题　第 5 题

RAM 中的信息是(　　)

(A) 生产厂家预先写入的　　(B) 计算机工作时随机写入的

（C）防止计算机病毒侵入所使用的　（D）专门用于计算机开机时自检用的

【答案】 正确答案选择(B),因为 RAM 是随机存储器,供计算机工作时随机写入,一旦断电则信息全消失。本题考察学生对计算机内存储器中随机存储器 RAM 知识的理解。

2000 年　初中基础题　第 6 题(高中基础题第 3 题)

计算机主机是由 CPU 与(　　)构成的

（A）控制器　（B）运算器　（C）输入、输出设备　（D）内存储器

【答案】 正确答案选择(D),本题考察学生对主机和中央处理器(CPU)概念的理解,因为 CPU 是由控制器与运算器组成,而主机由 CPU 与内存储器组成,输入、输出设备属于计算机的外设。

2000 年　初中基础题　第 7 题(高中基础题第 4 题)

计算机病毒的特点是(　　)

（A）传播性、潜伏性、易读性与隐蔽性　（B）破坏性、传播性、潜伏性与安全性

（C）传播性、潜伏性、破坏性与隐蔽性　（D）传播性、潜伏性、破坏性与易读性

【答案】 正确答案选择(C),计算机病毒是一种人为编制的计算机程序,通过自我复制来传播,因此传播性是衡量程序是否为病毒程序的首要条件,破坏性是计算机病毒的主要目的,隐蔽性和潜伏性是计算机病毒的显著特点。

2000 年　初中基础题　第 11 题(高中基础题第 5 题)

WINDOWS 9X 是一种(　　)操作系统

（A）单任务字符方式　（B）单任务图形方式

（C）多任务字符方式　（D）多任务图形方式

【答案】 正确答案选择(D),因为 WINDOWS 9X 是一种多任务的可视化的操作系统,它可以同时打开多个窗口,执行多个任务,而这些操作无论是应用程序还是文档编辑窗口,都可以利用图标、菜单或工具进行操作,即所见即所得。所以称之为多任务图形方式的操作系统。

2000 年　初中基础题　第 12 题(高中基础题第 10 题)

某种计算机的内存容量是 640K,这里的 640K 容量是指(　　)个字节

（A）640　（B）640 * 1000　（C）640 * 1024　（D）640 * 1024 * 1024

【答案】 正确答案选择(C),因为 1K 是 1024 个字节,所以 640K = 640 * 1024 个字节

2000 年　初中基础题　第 16 题(高中基础题第 14 题)

不同类型的存储器组成了多层次结构的存储器体系,按存取速度从快到慢的排列是(　　)

（A）快存/辅存/主存　（B）外存/主存/辅存

（C）快存/主存/辅存　（D）主存/辅存/外存

【答案】 正确答案选择(C),因为辅存和外存是同一个概念,快存实质是高速缓存,其运行速度比主存快,所以应该将常用的程序存放在高速缓存区,主存比辅存要快。

2000 年　高中基础题　第 8 题

计算机系统总线上传送的信号有(　　)

(A) 地址信号与控制信号　　(B) 数据信号、控制信号与地址信号

(C) 控制信号与数据信号　　(D) 数据信号与地址信号

【答案】 正确答案选择(B),计算机系统总线分为:地址总线、控制总线和数据总线,因此计算机系统在总线上传送的信号,按其类型,分别通过地址总线、控制总线和数据总线。

2000 年　高中基础题　第 9 题

计算机的运算速度取决于给定的时间内,它的处理器所能处理的数据量。处理器一次能处理的数据量叫字长。已知 64 位的奔腾处理器一次能处理 64 个信息,相当于(　　)字节。

(A) 8 个　　(B) 1 个　　(C) 16 个　　(D) 2 个

【答案】 正确答案选择(A),一个字节由 8 位的二进制数组成,64 位的奔腾处理器一次能处理 64 个信息,则相当于 8 个字节。

【背景知识】

DOS 的常用命令及应用

1. 文件

1) 文件概念:文件是指记录在存储介质(如磁盘、光盘)上的一组相关信息的集合。

2) 文件夹:将文件归类分组存放,每一组给定一个名,则称这个组为文件夹。

3) 文件的基本操作:

建立、存储、复制、删除、重命名、移动、建立子目录(建立文件夹)、删除子目录(删除文件夹)、进入子目录(进入子文件夹)、退出子目录(退出文件夹)

2. 内部命令

1) 内部命令:当启动 DOS 系统时,计算机引导程序将系统以及常用的命令处理模块驻留在计算机的内存中,我们称之为内部命令。

2) 常用的内部命令:

(1) 目录命令:

DIR(显示文件目录)

MD、CD、RD(子目录的建立、进入、删除命令)

(2) 文件操作命令:

COPY(复制命令)、DEL(删除命令)、REN(更改文件名)

TYPE(显示文本文件内容)

(3) 其他内部命令

DATA、TIME、VER、CLS 等

3. 外部命令

1) 外部命令:存储在外存储器上的 DOS 可执行的文件,这些文件程序所占的存储容量比较大,当用户使用外部命令时,计算机从外存调入内存,当执行完外部命令,就自动从内存中退出。

2) 常用的外部命令

(1) 磁盘格式化命令:　FORMAT　盘符　[/S][/V]

其作用，能够清除原盘中所有信息，并将磁盘规范成计算机所能接受的格式，以便有效存储信息。

（2）软盘复制命令：　DISKCOPY　［盘符1：］［盘符2：］

其作用，能够进行软盘之间的全盘复制（以磁道方式），不仅可以复制系统文件而且可以复制隐含文件。

【竞赛试题】

1995年　初中基础题　第1题（高中基础题第1题）

执行① C > DIR 命令后，屏幕上显示如下画面：

```
FORMAT      COM      12145
  SYS       COM       4878
  PUC       BAT        126
XCOPY       EXE      11216
4 FILE(S)   123456   BYTES   FREE
```

接着又顺序执行了如下几条 DOS 命令：

② C > DIR > DF. TXT　//表示将列表显示的目录作为文件写盘//

③ C > TYPE　DF. TXT

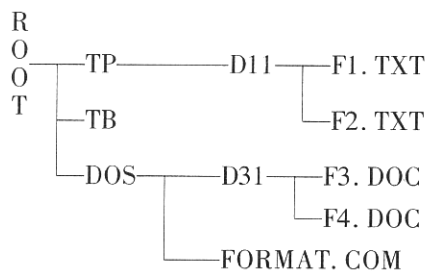
④ C > DIR

试问：执行命令③和④在屏幕上显示的结果是否与①相同？

【答案】 此题主要考察学生对 DOS 内部命令的理解，首先要理解命令②的作用，将 DIR 列表显示的目录作为文件写入 DF. TXT 文件中，屏幕无任何目录显示；当执行命令③时，TYPE 命令将显示文本文件 DF. TXT 的内容，因此执行命令③后显示结果与①相同；执行命令④后，计算机将显示当前磁盘文件目录，无疑将比①增加一个 DF. TXT 文件。命令 C > DIR > DF. TXT 这里第二个“ > ”是传输命令，将 DIR 显示目录传送到 DF. TXT 文件中。

1996年　初中基础题　第1题（高中基础题第1题）

已知 A 盘上的目录和文件组织如下：



其中 TP、TB、DOS、D11、D31、都是子目录名。设当前命令提示符为 A:\TB >，请写出完成如下操作的 DOS 命令：

① 将 F1. TXT 移到 D31 子目录中去；

② 删除子目录 TB；

③ 在 DOS 运行中,没有执行过 PATH 命令,现要用 DOS 子目录中 FORMAT 命令,对插入在 B 驱动器(5.25 英寸高密)中的 360KB 软盘进行格式化工作,请写出相应的操作命令。

本题主要考察学生的 DOS 基本知识和基本操作。主要是文件的复制、删除,文件的访问等知识。

【答案】

① 解答:

A:\TB>COPY A:\TP\D11\F1.TXT A:\DOS\D31

//将 F1.TXT 文件复制到 D31 子目录中//(用绝对路径进行复制)

A:\TB>DEL A:\TP\D11\F1.TXT //删除 D11 子文件夹中的 F1.TXT 文件.//

本题主要考察学生对内部命令知识、文件存放路径知识的理解以及路径的绝对表示方法和相对表示方法。

② 解答:

A:\TB>CD.. A:\>RD TB 或者 DELTREE A:\TB

本题知识点是删除子目录,在删除子目录时要注意:不能在本级目录下删除本级目录,必须在其上级目录下进行删除。

③ 解答:

操作步骤:

A:\TB>CD..

A:\CD DOS

A:\DOS>FORMAT B:/S/4

本题主要考察学生对 DOS 外部命令知识的理解和掌握情况。由于没有运行 PATH 路径搜索命令,要执行 FORMAT.COM 外部命令,必须到具有该外部命令的目录中,计算机才能找到命令并执行该命令,所以用了三个步骤。参数"/S"含义是将 B 盘格式化成带有系统的盘,参数"/4",表示对低密磁盘格式化。

1996 年 高中基础题 第 1 题

本题基本同初中组第 1 题,但第②问题是交换 F2.TXT 与 F3.DOC 两个文件的内容,要求使用的命令不得超过三条。

【答案】 解答:

A:\TB> COPY A:\TP\D11\F2.TXT A:\DOS\D31\HH.DOC

A:\TB> COPY A:\DOS\D31\F3.DOC A:\TP\D11\F2.TXT

A:\TB> REN A:\DOS\D31\HH.DOC F3.DOC

该题主要考察学生对 COPY、REN 命令理解,同时也灵活应用程序设计中的变量交换的思想方法。注意 COPY、REN 命令的不同,复制文件可以在不同路径、不同的盘符之间进行,而更改文件名必须在相同盘符和相同路径下完成,其原理是文件改名只是将该文件的目录名称更换,文件信息的物理存储位置不变。

1996 年 初中基础题 第 2 题

执行命令时，屏幕上显示如下出错信息：

WRITE PROTECT ERROR WRITING DRIVE B

ABORT, RETRY, FAIL?

请说明这是什么错误？应如何校正？

【答案】 此错误是指B驱动器中盘片被写保护，是退出，再试，放弃？用户应该取出盘片，将写保护去掉，再放入B驱，选择“R”再重写完成写操作。本题考察学生上机操作能力，对在上机操作中，经常“显示的出错信息”是否理解以及如何解决。

1997年　初中基础题　第3题（高中基础题第2题）

在MS DOS的根目录中，有如下文件：

TIME. EXE　　TIME. COM　　TIME. BAT

试问：C:\ >TIME <回车>执行的是什么命令？

【答案】 正确答案是：当输入TIME并回车时，计算机执行的是内部命令，屏幕显示当前计算机的时间。本题考察学生对DOS内部命令、外部命令以及其他命令执行顺序的理解。

1998年　初中基础题　第4题（高中基础题第3题）

已知在计算机C:\DOS下有一个正确的FORMAT. COM文件，当执行如下命令：

C:\ >FORMAT　A: <回车>得到的回答是BAD COMMAND OR FILE NAME提示信息，下面解释正确的是_____

（A）根目录中没有AUTOEXEC. BAT文件；

（B）在执行该命令前操作者没执行过PATH命令；

（C）C:\DOS中的FORMAT. COM文件有错；

（D）由于AUTOEXEC. BAT或操作者最后执行过的PATH命令缺少路径C:\DOS，或者根本没有执行PATH命令。

【答案】 正确答案选择（D），因为执行C:\ >FORMAT　A: <回车>命令时，出错信息显示“命令失败或文件名错”，这里文件名是正确的，这说明当前路径下无此命令。如果在执行此命令前曾经执行过PATH C:\DOS（或AUTOEXEC. BAT文件中含有此命令），则计算机也不会报错，它会自动搜索路径，执行C:\DOS中的FORMAT. COM命令。本题考察学生对PATH命令的理解应用以及对执行外部命令的所需要条件的知识的考核。

1998年　初中基础题　第5题（高中基础题第4题）

将A盘上50个文件用C:\ >COPY　A:*. *命令复制到C盘的当前目录中，在复制到某个文件时，由于读数据出错，屏幕显示：

ABORT, RETRY, IGNORE, FAIL?

键入“I”后，继续复制没再出现过错误信息，最后复制的结果是_____。

（A）读数据出错的文件不正确，其他文件正确；

（B）读数据出错的文件不正确，其他文件也不正确；

（C）读数据出错的文件正确，其他文件不正确；

（D）复制的文件完全正确。

【答案】 正确答案选择:(A),在许多文件复制过程中,某一个文件读错误,当键入"I"后,忽略错误,继续复制文件,因此仅此文件无法读取,而其他文件是正确复制且能读取。本题考察学生对系统提供的出错信息处理的理解和应用。

1999 年 初中基础题 第 5 题(高中基础题第 5 题)

以下 DOS 命令中,有可能在磁盘上建立子目录的是(　　)

(A) TYPE　　(B) DIR　　(C) XCOPY　　(D) CD

【答案】 正确答案是:(C),因为 TYPE 是显示文本文件的内容,DIR 显示磁盘的文件目录,CD 是进入或退出子目录,只有 XCOPY 命令能够拷贝文件夹及其子文件夹的内容,因此只有该命令有可能在磁盘上建立子目录。本题主要考察学生对 DOS 命令的理解和应用。

1999 年 初中基础题 第 6 题(高中基础题第 6 题)

在 CONFIG. SYS 文件中,装入特定可安装设备驱动程序的命令是(　　)

(A) BUFFER　　(B) FILES　　(C) DRIVER　　(D) DEVICE

【答案】 正确答案是:(C),这里 BUFFER 是开辟缓冲区,FILES 是数据库系统中,定义所需要建立的文件数,DEVICE 是指装置数,只有 DRIVER 是驱动程序命令。本题考察学生对系统设置命令的理解和应用。

1999 年 初中基础题 第 11 题(高中基础题 11)

执行 DOS 命令:C:\ATTRIB A:*.*的功能是(　　)

(A) 查看 A 盘上所有文件属性　　(B) 查看 A 盘上当前目录中所有文件属性

(C) 查看 A 盘上所有系统文件属性　　(D) 删去 A 盘上所有隐含文件的属性

【答案】 正确答案选择:(B),ATTRIB 的作用是查看当前目录下的文件属性。

1999 年 初中基础题 第 12 题(高中基础题第 12 题)

执行下列 DOS 命令,效果等价的是(　　)组

(A) COPY *.FOR　　与　　COPY *.FOR CON

(B) COPY A:*.* B:　　与　　XCOPY A:*.* B:

(C) COPY FILE1.TXT + FILE2.TXT　　与　　COPY FILE2.TXT + FILE1.TXT

(D) XCOPY A:*.* B:/S　　与　　DISKCOPY A: B:

【答案】 正确答案选择:(D),答案(A)的左边是错误的命令,右边是将文件内容复制到外设,这里是指显示器;COPY 命令只能复制文件,不能复制系统文件和带目录路径的文件夹,所以答案(B)是错的,答案(C)的文件连接复制与文件的先后顺序有关,所以(C)是错的,只有(D)命令的操作效果相同,将 A 盘中的所有文件,包括系统均复制到 B 盘。

1999 年 初中基础题 第 15 题(高中基础题第 15 题)

下列文件名中,属于 DOS 中的保留设备名的为(　　)

(A) AUX　　(B) COM　　(C) CON1　　(D) PRN1

【答案】 正确答案是:(C),CON1 表示接外设端口,其他都和保留设备名无关。

1999 年　初中基础题　第 20 题(高中基础题第 20 题)

对具有隐含属性(H)的当前目录下的文件 AB. TXT,能成功执行的 DOS 命令是(　　)

(A) TYPE　AB. TXT　　(B) COPY　AB. TXT　XY. TXT

(C) DIR　AB. TXT　　(D) REN　AB. TXT　XY. TXT

【答案】 正确答案选择:(A),只有 TYPE 显示文本文件内容的命令,能够成功执行 DOS 命令,其他命令都说没有发现该文件。

【背景知识】

网络的基本知识

1. 网络概念

将地理位置不同的计算机,用通信线连接起来,共同遵守一定的协议,共享计算机的软、硬件资源。因特网是网络的集合,是全球最大的网络。

2. 网络分类

局域网:局限于某个范围内的网络连接

广域网:跨地区的局域网称为广域网。因特网是覆盖全球的广域网。

3. 因特网提供的服务功能主要有:

(1) 信息浏览(WWW)　　(2) 文件传输(FTP)

(3) 发送接收电子邮件(E-maill)　　(4) 电子公告牌(BBS)

(5) 远程登录(telnet)　　(6) 电子商务

4. 网址的结构:http://www. sina. com. cn

http://——超文本浏览协议,www. sina——表示主机域名,

com——网络机构域名,这里是商业网,cn——地区域名,这里是中国域名。

5. 电子邮件的地址:zhangming@ yahoo. com 这里 zhangming 是用户,@ 是分隔符号,yahoo 主机名(雅虎),com 是域名。

【竞赛试题】

2000 年　初中基础题　第 11 题(高中基础题第 6 题)

INTERNET 的规范译名应为(　　)

(A) 英特尔网　　(B) 因特网　　(C) 万维网　　(D) 以太网

【答案】 正确答案选择(B),因特网又称国际互联网。我国于 1994 年正式联入因特网。全国科学技术名词审定委员会于 1997 年 7 月 18 日为 INTERNET 作出了命名,中文名词为“因特网”,注译是“指全球最大的、开放的、由众多网络相互连接而成的计算机网络”。

万维网是 WWW 的中文命名,英语是 WORLD WIDE WEB 广泛联络世界的网,这里是指“基于超文本的、方便用户信息浏览和信息搜索的信息服务系统”。人们通过信息服务系统浏览网上信息。

以太网(ETHERNET)是一种可以随机存取的计算机局域网,它用电缆线连接,在比较小的范围内互通信息共享网络资源。学校计算机房内的教学网是局域网。

2000 年　高中基础题　第 7 题

计算机网络是一个(　　)系统

(A) 管理信息系统　　(B) 管理数据系统

(C) 编译系统　　(D) 在协议控制下的多机互联系统

【答案】 正确答案选择(D),计算机网络是指在一定区域内,多台计算机用通信线连接起来,相互间共同遵守一定协议,共享计算机的软、硬件资源的互联系统。本题考察学生对网络知识的理解。

2000 年　高中基础题　第 11 题

下列哪些计算机网络不是按覆盖地域划分的

(A) 局域网　　(B) 都市网　　(C) 广域网　　(D) 星型网

【答案】 正确答案选择(D),一般的计算机网络按网络的涉及范围的距离划分为广域网和局域网,广域网即 WAN(WIDE AREA NETWORK),城域网(MAN,即 METROPOLITAN AREA NETWORK),局域网(LAN,即 LOCAL AREA NETWORK),都市网属于城域网。如果按网络的层次结构划分有总线型、星型、环型等形式。

【背景知识】

计算机中有关数、编码的基本常识

1. 计算机是智能化的电器设备

计算机就其本身来说是一个电器设备,为了能够快速存储、处理、传递信息,其内部采用了大量的电子元件,在这些电子元件中,电路的通和断、电压高低,这两种状态最容易实现,也最稳定、也最容易实现对电路本身的控制。我们将计算机所能表示这样的状态,用 0,1 来表示,即用二进制数表示计算机内部的所有运算和操作。

2. 二进制数的运算法则

二进制数运算非常简单,计算机很容易实现,其主要法则是:

$0+0=0$　$0+1=1$　$1+0=1$　$1+1=0$; $0\times0=0$　$0\times1=0$　$1\times0=0$　$1\times1=1$

由于运算简单,电器元件容易实现,所以计算机内部都用二进制编码进行数据的传送、计算。

3. 十进制与二进制、八进制、十六进制数之间的相互转换

(1) 数的进制与基数

计数的进制不同,则它们的基数也不相同,如表 1-1 所示。

表　1-1

进　制	基　　数	十进制数	典型示例(转换)
二进制	0 1	10, 7, 23	1010, 111, 10111
三进制	0 1 2	10, 7, 23	101, 21, 212
四进制	0 1 2 3	10, 7, 23	22, 13, 113

续　表

进　制	基　　数	十进制数	典型示例(转换)
八进制	0 1 2 3 4 5 6 7	10, 63, 126	12, 77, 176
十进制	0 1 2 3 4 5 6 7 8 9		
十六进制	0 1 2 3 4 5 6 7 8 9 A B C D E F	10, 63, 254	A, 3F, 15E

(2) 数的权

不同进制的数,基数不同,其每位上所代表的值的大小也不相同,我们称之为"权"

① 十进制数,逢十进一。如,$(219)_{10}=2\times10^2+1\times10^1+9\times10^0$

② 二进制数,逢二进一。如,$(11010)_2=1\times2^4+1\times2^3+0\times2^2+1\times2^1+0\times2^0=26$

③ 八进制数,逢八进一。如,$(273)_8=2\times8^2+7\times8^1+3\times8^0=187$

④ 十六进制数,逢十六进一。如,$(27B)_{16}=2\times16^2+7\times16^1+11\times16^0=635$

从以上的计算中,可以看到,进制不同,基数不同,每位上权值大小也不相同,数值大小也不相同。

(3) 十进制数转换成任意进制数

将十进制数转换成任意进制数的基本方法是:将十进制数除以所定的进制数反向取余。

例如:

① 将 39 用二进制数表示,用如下的短除法,求余数,并反向取余。如果转换成二进制还可以用右边的 2 的 N 次幂表示。

短除法 $39\Rightarrow(100111)_2$

2	3 9		
2	1 9	……	1
2	9	……	1
2	4	……	1
2	2	……	0
2	1	……	0
	0	……	1

2 的 N 次幂表示

$39=2^5+2^2+2^1+2^0\Rightarrow(100111)_2$

② 将 245 用 8 进制数表示,我们可以采取短除法或用 8 的 N 次幂表示

$(245)_{10}=(365)_8$

8	2 4 5		
8	3 0	……	5
8	3	……	6
	0	……	3

8 的 N 次幂表示

$245=5*8^2+6*8^1+3*8^0$

想一想,为什么要反向取余。对于十进制小数要转换成其他进制的数,则是不断将小数部分乘以进制数取整,作为转换后的小数部分,直到为零或精确到小数点后几位。如:

$(0.35)_{10} \approx (0.01011)_2$　　$(0.125)_{10} = (0.001)_2$

(4) 任意进制的数转换成十进制数

将任意进制数转换成十进制数的基本方法是按权展开,见(2)数的权内容。

4. ASCII 编码

由于计算机是电器设备,计算机内部用二进制数,这样对于从外部输入给计算机的所有信息必须用二进制数表示,并且对于各种命令、字符等都需要转换成二进制数,这样就牵涉到信息符号转换成二进制数所采用的编码问题,国际上统一用美国的标准信息编码 ASCII 代码。它用8位二进制数表示,最高位置为0。因此基本的 ASCII 字符集有128个。在这样的编码集中"A"对应编码$(01000001)_2$相当于十进制65。中国汉字编码则用两个字节表示,为了区别一般编码,其最高为设置为1。汉字编码分94个区,每个区94位,一个区号和一个位号就可以惟一确定一个汉字,所以我们称这样的编码为区位码。

5. 汉字输入方法

汉字输入方法有很多种,大体可以分为:流水码、音码、形码、音形码。

(1) 流水码:区位码、电报码、通讯密码等均属于流水码,优点重码率少,缺点难于记忆;

(2) 音码:以汉语拼音作为编码输入汉字,优点是大多数人都易于掌握,但同音字多,重码率高,影响输入的速度;

(3) 音形码:将音码和形码结合起来,输入汉字,减少重码率,提高汉字输入速度;

(4) 形码:根据汉字的字型进行编码,编码的规则比较多,难于记忆,必须经过训练才能较好地掌握。

【竞赛试题】

1996年　初中基础题　第3题(高中基础题第2题)

请用等号或不等号联接表示下列不同进位制数值的大小。

例如:$(3)_{10} < (4)_{10} = (100)_2 < (A)_{16}$

其中圆括号外右下角的下标,表示圆括号内数的进位制。

$(98.375)_{10}$　　$(142.3)_8$　　$(58.5)_{16}$　　$(1011000.0101)_2$

【答案】 本题意在考察学生对不同进制数的转换方法是否掌握。解决方法,以十进制为基准,将其他进制的数均转换为十进制数,方法和步骤如下:

$(142.3)_8 \rightarrow 1\times8^2+4\times8^1+2\times8^0+3\times8^{-1} \rightarrow (98.375)_{10}$

$(58.5)_{16} \rightarrow 5\times16^1+8\times16^0+5\times16^{-1} \rightarrow (88.3125)_{10}$

$(1011000.0101)_2 \rightarrow 1\times2^6+1\times2^4+1\times2^3+1\times2^{-2}+1\times2^{-4} \rightarrow = (56.3125)_{10}$

所以排序结果是:$(98.375)_{10} = (142.3)_8 > (58.5)_{16} > (1011000.0101)_2$

1997年　初中基础题　第7题(高中基础题第3题)

已知 ASCII 码表中的大写字母后有6个其他字符,接着便是小写字母。现已知:A 字母的 ASCII 码为$(41)_{16}${表示16进制数41},试写出如下字母用十进制表示的 ASCII 码:

G→(　　)$_{10}$　　b→(　　)$_{10}$　　t→(　　)$_{10}$

【答案】 该题的解法是先将 A 字母的 ASCII 代码的 16 进制数 41 转换成十进制数，再根据英文字母的排列关系，求得其他字母的十进制的 ASCII 代码。本题主要考察学生数的进制转换以及对常用字符的 ASCII 代码值的了解。

$A \rightarrow$ ASCII 码$\rightarrow(41)_{16} \rightarrow 4\times16^1+1\times16^0=(65)_{10}$

G 的 ASCII 码是$(72)_{10}$　小写字母 a 的 ASCII 码是$(97)_{10}$

b 的 ASCII 码$\rightarrow(98)_{10}$　t 的 ASCII 码$\rightarrow(117)_{10}$

1997 年　初中基础题　第 6 题(高中基础题第 5 题)

一个汉字的机内码目前通常用 2 个字节来表示：第一个字节是区码的区号加$(160)_{10}$；第二个字节是区位码的位码加$(160)_{10}$。

已知：汉字“卻”的区位码是 4020，试写出机内码两个字节的二进制的代码：

【答案】 “卻”的机内区码是 160 + 40 = 200，其二进制代码是$(11001000)_2$“卻”的机内位码是 160 + 20 = 180，其二进制代码是$(10110100)_2$

1	1	0	0	1	0	0	0
1	0	1	1	0	1	0	0

本题考察学生对汉字编码的理解以及数的二进制表示方法。

1998 年　初中基础题　第 6 题

下面四个不同进制的数，最小的一个数是_____。

(A) $(11011001)_2$　(B) $(75)_{10}$　(C) $(37)_8$　(D) $(A7)_{16}$

【答案】 本题正确答案选择(C)。

基本方法是：将任意进制数转换成十进制数，然后进行大小比较。

$(11011001)_2=1\times2^7+1\times2^6+1\times2^4+1\times2^3+1\times2^0=(217)_{10}$

$(37)_8=3\times8^1+7\times8^0=(31)_{10}$

$(A7)_{16}=10\times16^1+7\times16^0=(167)_{10}$

通过转换计算，$(37)_8$ 数最小，所以选择(C)。

1998 年　初中基础题　第 7 题(高中组基础第 6 题)

小张用十六进制、八进制和十进制写了如下的一个等式：52 - 19 = 33

式中三个数是各不相同进位制的数，试问 52、19、33，分别为_____。

(A) 八进制，十进制，十六进制　(B) 十进制，十六进制，八进制

(C) 八进制，十六进制，十进制　(D) 十进制，八进制，十六进制

【答案】 该题正确答案选择(B)

根据题意分析，算式计算结果为 33，这儿 33 不可能是十进制数，否则 52、19 必须是十进

制数,这与题意不合;计算结果也不可能是十六进制数,若是,则52必须是八进制数减去十进制19其结果不可能是十六进制33,所以判断结果选择(B)。其运算:

$(52)_{10}-(19)_{16}=(33)_8 \rightarrow 52-(16+9)=(3\times 8+3)$

本题考察学生对问题分析判断的能力以及对不同进制数之间的转换关系的理解。

1998年　初中基础题　第8题(高中基础题第9题)

如果用一个字节来表示整数,最高位用作符号位,其他位表示数值。例如:

0	0	0	0	0	0	0	1	表示+1

↑　符号位表示正

1	0	0	0	0	0	0	1	表示-1

↑　符号位表示负

① 试问这样表示法的整数A的范围应该是_____。

(A) $-127\leqslant A\leqslant 127$　　(B) $-128\leqslant A\leqslant 128$

(C) $-128\leqslant A<128$　　(D) $-128<A\leqslant 128$

【答案】 根据题意,正确答案选择(A),因为正整数的范围仅能用7位的二进制数表示,由于最高位是零,当后7位全为1时,表示整数127,再加1,需要进位,则符号位变为1,数据发生质的变化,数据由正变为负;而负数道理基本同正数。

② 在这样表示法中,以下_____说法是正确的。

(A) 范围内的每一个数都只有惟一的格式　(B) 范围内每一个数都有两种格式

(C) 范围内的一半数有两种格式　(D) 范围内只有一个数有两种表示格式

本题正确答案选择(D),这是因为正数、负数都只有惟一的表示格式,而零可以有两种格式即:00000000和10000000

1999年　初中基础题　第13题(高中基础题第13题)

已知小写字母"m"的十六进制的ASCII码值是6D,则小写字母"c"的十六进制数的ASCII码值是(　　)

(A) 98　　(B) 62　　(C) 99　　(D) 63

【答案】 本题仍然考核学生对ASCII码的理解以及数的进位制问题,该题正确答案选择(D),因为由m的十六进制ASCII码值是6D,可以知道小写c与m相差十进制数10,相当于十六进制数A,将6D-A=63(十六进制减法),所以选择答案(D)。

1999年　初中基础题　第14题(高中基础题第14题)

计算机中的数有浮点与定点数两种,其中用浮点数表示的数,通常由(　　)这两部分组成。

(A) 指数与基数　(B) 尾数与小数　(C) 阶码与尾数　(D) 整数与小数

【答案】 正确答案选择(C),因为浮点数的表示同数学中的科学计数法有相似之处,由小数及10的N次幂表示,计算机中的浮点数则将小数部分表示为尾数,将10的N次幂的N作为阶码,所以正确答案选择(C)。

1999 年　初中基础题　第 17 题(高中基础题第 17 题)

十进制算术表达式:3 * 512 + 7 * 64 + 4 * 8 + 5 的运算结果,用二进制表示为(　　)

(A) 10111100101　　(B) 11111100101　　(C) 11110100101　　(D) 11111101101

【答案】 正确答案选择(B),考核学生对数的进位制的理解和掌握情况。该题较快的解决方法是将数转换成二进制数比较方便,其方法如下:

$$3*512+7*64+4*8+5=(2^1+2^0)*2^9+(2^2+2^1+2^0)*2^6+2^2*2^3+2^2+2^0$$
$$=2^{10}+2^9+2^8+2^7+2^6+2^5+2^2+2^0\rightarrow(11111100101)_2$$

1999 年　初中基础题　第 18 题(高中基础题第 18 题)

组成"教授"(JIAO SHOU),"副教授"(FU JIAO SHOU)与"讲师"(JIANG SHI)这三个词的汉字,在 GB2312-80 字符集中都是一级汉字,对这三个词排序的结果是(　　)

(A) 教授、副教授、讲师　　(B) 副教授、教授、讲师

(C) 讲师、副教授、教授　　(D) 副教授、讲师、教授

【答案】 正确答案选择(D),计算机对字符的排序是按照字符的 ASCII 码值的大小进行排序的。汉字的排序则是根据汉语拼音的字母的 ASCII 码值进行排序,所以本题的正确答案选择(D)。本题仍然是考察学生对字符的 ASCII 码值知识的理解。

2000 年　初中组基础题　第 1 题

下列无符号数中最小的数是(　　)

(A) $(11011001)_2$　　(B) $(75)_{10}$　　(C) $(37)_8$　　(D) $(2A)_{16}$

【答案】 正确答案应该选择(C),不同进制数的大小比较的基本方法是将它们化成统一进制的数再比较大小,其换算方法同前面讲解的一样。我们均将他们换成十进制数:

$(11011001)_2=(217)_{10}$　　$(37)_8=(31)_{10}$　　$(2A)_{16}=(42)_{10}$

所以正确答案选择(C)。

2000 年　初中基础题　第 3 题

GB2312-80 规定了一级汉字 3755 个,二级汉字 3008 个,其中二级汉字字库中的汉字是以(　　)为序排列的。

(A) 以笔划多少　　(B) 以部首　　(C) 以 ASCII 码　　(D) 以机内码

【答案】 正确答案选择(B),GB2312-80 方案是我国于 1981 年颁布的《信息交换用汉字编码字符集》,共收录了 6763 个汉字,其中一级汉字 3775 个是按照拼音排序,二级汉字 3008 个,按照部首排序,另外还有 682 个图文符号。

第二节　数据结构与算法

【背景知识】

一、数据及数据类型

(1) 数据是指用文字、数字、图像、符号等能被计算机接受和处理的信息。

(2) 数据元素是数据整体中相对独立的单位,例如5是整数中的一个元素,字母"A"是字符集中的一个元素。

又如,全校教师的档案是一个称之为"文件"类型的数据,每个教师的情况称为记录,它是文件中的一个元素。

在一个教师的记录中,可以分为几项,如姓名、年龄、工资等,这样的每一项我们又称之为数据项或称为字段。

(3) 数据类型是指数据种类,如有整数、实数、字符型、布尔型数等。

二、数据结构

1. 数据结构的概念

数据结构是指数据与数据之间的存在一定的联系,这种联系是内在的,具有一定的逻辑关系,我们称之为数据的逻辑结构。数据存储在计算机的存储器上,存在着一定的存储联系,我们称这样的联系为数据的物理结构或存储结构。通常我们所讲的数据结构是指逻辑结构。

2. 数据结构的类型

数据之间的内在联系有多种,常见的数据结构有:线性表、栈、队列、矩阵、树、图。

三、线性表结构

1. 线性表定义

线性表是指具有相同性质的数据元素的一个有限序列,该序列中的数据元素,除第一个和最后一个元素外,都有一个直接的前驱元素和一个直接的后继元素。如下的N个元素即为线性表。

$$(A_1, A_2, \cdots, A_K, \cdots, A_N)$$

其中 A_1 称为表头,A_N 表尾,表中元素的个数称为表长。在线性表中,元素的位置是有序的,第K个元素 A_K 的前一个元素处于K-1的位置是 A_{K-1} 而它的后继元素则处于第K+1的位置是 A_{K+1},数据元素之间的这样有序性就是一种线性的关系。

2. 线性表的存储结构

线性表的逻辑结构是一种有序元素序列,而它的存储结构可以用两种方式表示:

1) 线性表的顺序存储

线性表的顺序存储是线性表的一种最简单的存储结构,其存储方式是:在内存中为线性表开辟一块连续的存储空间——用数组这样的数据类型可以完成这样的存储结构,如下图所示:

```
var a: array[1..n]  of  integer;
```

	A[1]	A[2]	A[3]		A[N]
数组	89	78	80	…	70	90

顺序存储结构的特点是在内存中开辟一块连续的存储空间。这样的存储结构,访问方便,查找方便,但若进行插入、删除操作,将引起数据的移动,它是一种静态存贮,一旦变量被说明以后,其存储空间也被相应开辟,程序运行中不能改变所占空间的大小。

线性表的顺序结构也可以用记录实现:

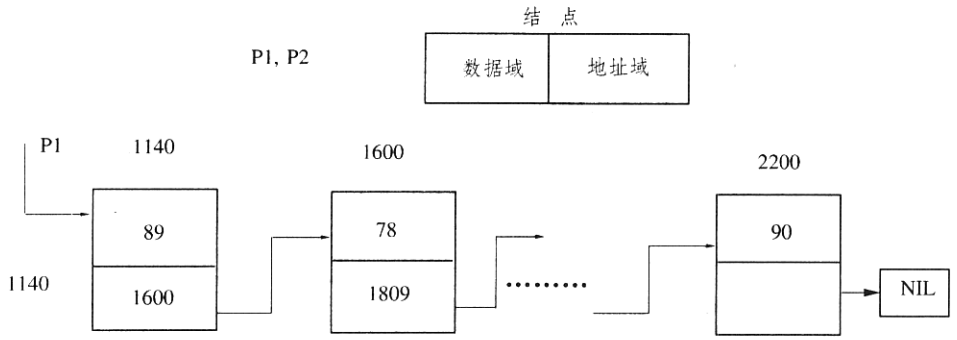
```
type xian = record
              ar: array[1..m] of 基本数据类型;
              point: integer;
              end;
```

ar 的元素的位置指向用 point 实现。

2）线性表的链接存储

数据元素之间的线性表关系，还可以用链接表结构进行存储，这样的线性结构称为线性链表。在 PASCAL 语言中用指针变量可以实现。如前面所定义的(89, 78, 80, … , 70, 90) N 个数据元素线性表，其链接表形式如下：

```
type   point = ^ stu;
       stu = record
             data:integer;
             next:point;
           end;
var   p1, p2: point;
```



链接表结构的特点是存储单元的地址是不需要连续的，每个结点至少有两个域，数据域和地址域，利用地址域的指针变量将结点按序连接。程序运行时，存储空间可以临时申请，使用完后，可以将空间归还。所以又称为动态存贮结构。对链表结构进行访问、查找就没有数组方便，若要查找某个结点，必须从表头开始顺序查找，直到找到所找结点为止，但如果进行插入、删除操作，则不会引起数据的大量移动。

3. 线性表的操作运算

线性表的操作有建立线性表、求线性表的长度、查找一个关键字、插入数据、删除数据、数据的排序、线性表归并、线性表的输出等运算。

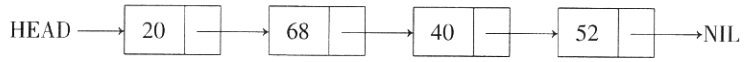
线性表所采用的数据存储结构不同，其操作的算法也不相同。如果采用顺序存储结构，以上的线性表操作运算比较简单，程序实现也非常方便，由读者自己完成。下面我们介绍链接表的一些基本操作。

1）线性链表的建立

建立线性链表的过程，实质是不断生成新结点，将结点链接到表中的过程。链接的方法有多种，一般分为：插入表头、插入表尾、按一定要求插入链表中的某个位置。下面我们举例介绍

插入表尾的程序设计方法。

例 输入一个整数序列,按输入数据的顺序建立如下链表:



我们采用插入表尾的方法来建立一个线性表。

分析:

(1) 从空表开始,申请一个头结点;

(2) 每输入一个数,存贮数据并插入表尾;

(3) 申请一个结点 ;

(4) 重复(2)、(3),直到建立完毕。

程序如下:

```
program exp1 __1;
  type
      ref = ^object;
  object = record
             num: integer;
             next: ref;
             end;
  var
    p, q, r: ref;
          i: integer;
  begin
    writeln('enter natural number, -1 to end:');
    read(x);
    new(q); r:=q; p:=q;{初始化,p 为头指针}
    while x <>-1 do
      begin
        q^.num:=x;
        new(q);
        r^.next:=q;
        r:=q;{r 中间指针}
        read(x);
      end;{while 建立链表过程}
    readln;
    r^.next:=nil;
    q:=p;
    writeln ('the output will be:');
```

```
    while q^.next <  > nil do
      begin
        write(q^.num: 8);
      q:=q^.next;
    end;{while 打印链表过程}
end.
```

运行示例：

enter natural number, -1 to end:

20　68　40　52　-1

the output will be:

20　68　40　52

2）线性链表的查找

当建立好线性链表后，我们该如何访问链表中的一个结点呢？若数据采用静态存贮方式——数组存贮，则访问极其方便，只要直接访问数组中的某一个单元的下标，即可以找到该单元存贮变量的值，而采用动态存贮方式，可以动态管理内存，节省空间，但若访问其中某一个结点，则必须从链表的表头开始顺序访问，直到找到所要查找的单元为止。其基本算法步骤如下：

① 头指针从所指向的表头开始，判断结点中的数据是否为查找的数据（或结点的位置是否为所要查找的位置）；

② 如果查找到数据（或位置），则输出，并且退出查找；否则将当前指针顺序指向下一个结点单元；

③ 重复①、②直到表为空，输出没有找到数据。

3）线性链表的插入、删除、求线性表的长度操作

对于已建立的线性链表，除了可以进行查找运算外，还可以计算线性表的长度，在已建立的线性表中插入一个结点，删除一个结点等基本的线性表的操作运算。

四、栈

1. 栈的定义

栈是一种特殊的线性表，其运算受到限制，仅允许在表的一端进行插入和删除运算。这一端被称为栈顶，相对另一端称为栈底。向一个栈插入新元素又称作进栈、入栈或压栈。它将新元素放到栈顶元素的上面，使之成为新的栈顶元素；从一个栈删除元素又称为出栈或退栈，它是把栈顶元素删除掉，使相邻的元素成为新的栈顶元素。由于栈的操作仅在栈顶一端进行，后进栈的元素必定先被删除，所以把栈的操作称作先进后出表（FILO）。

2. 栈的存储结构

由于栈是线性表，所以线性表的存储结构也适合栈。

1）栈顺序存储结构，用记录类型定义为：

```
TYPE STACK =RECORD
              AR: ARRAY[1..M] OF 基本数据类型;
```

```
    TOP: INTEGER;(栈顶指针)
  END;
```

用 AR 存放栈的元素,TOP 域用来存储栈顶元素所在单元的编号,即栈顶指针,M 表示栈的最大深度。

2) 栈的链接存储结构

栈的链接存储结构与线性表的链接存储结构基本相同。这里不再赘述。

3. 栈的操作运算

(1) 进栈(PUSH)运算,向栈顶插入一个新元素;

(2) 出栈(POP)运算,删除栈顶元素;

(3) 判断栈是否为空或溢出。

就以上的操作运算,我们给出相应的算法描述,读者可以根据算法,用程序设计语言上机完成。

1) 进栈算法

算法步骤

① 首先检查栈是否已满,若满,则作"溢出"出错处理;

② 将栈顶指针加 1;

③ 将新元素赋给栈顶单元。

算法描述:

```
procedure      push (s, x)
  begin
  ① if s.top = m then error ('overflow');
  ② s.top: = s.top + 1;
  ③ s.ar[s.top]: = x;
 end;
```

2) 出栈算法

算法步骤:

① 检查栈是否为空,若为空,则作"下溢出"出错处理;

② 将栈顶元素赋给指定的变参 X 或删除栈顶元素;

③ 将栈顶指针下移即减 1。

算法描述:

```
procedure      pop (s, x)
  begin
  ① if s.top = 0 then error ('underflow');
  ② x: = s.ar[s.top];
  ③ s.top: = s.top  - 1
 end;
```

3) 栈是否为空或溢出

此算法非常简单,只需要用 IF 命令,判断 S.TOP = 0 或判断 S.TOP = M 就可以完成,读者自己练习完成。

有关链接结构的栈的基本操作同顺序存储的栈,其算法基本相同,读者有兴趣可以练习。

在后面竞赛题解中，还要介绍有关栈的应用方面的知识。

五、队列

1. 队列的定义

队列也是一种特殊的线性表，其操作运算是允许在表的一端进行插入运算，而在表的另一端进行删除运算。在运算中，规定插入新元素在表尾进行，删除元素在表头进行。向队列中插入元素称之为入队或进队，从队列中删除元素，称之为出队。由于队列的操作分别在表的两端进行，删除操作是将先进去的元素最先删除，后进入的元素后删除，所以这样的操作称为先进先出(FIFO)。

2. 队列的存储结构

队列的存储结构分为顺序存储和链接存储。

(1) 顺序存储方式，用记录和数组实现；

(2) 链接表存储方式，用记录和指针变量实现。

顺序结构：

```
type   queue = record
           ar:array [1..m] of integer;
           f, r:integer;
       end;
var q:queue; x:integer;
```

在以上定义队列存储结构中，AR 表示存放每个数据元素；F，R 用来指向表头和表尾，即 F 指向队头，R 指向队尾。链接表的表示与顺序存储有所不同，但道理一样。

3. 队列的运算

队列的主要运算有插入、删除、置队空。建队的运算其实质就是插入运算。

1) 插入运算的算法

```
procedure insert(q,x);
  begin
    ① if q.r = m then error ('overflow');
    ② q.r:=q.r+1 ;{队尾指针加 1}
    ③ q.ar[q.r]:=x;{将新元素赋给队尾}
    ④ if q.f = 0 then q.f:=1;
                  {当队为空,则插入后,将队首指针置 1}
  end;
```

2) 删除运算的算法

```
procedure dele (q, x);
  begin
    ① if q.f = 0 then error ('underflow');
    ② x:=q.ar[q.f];
```

```
③ if q.f = q.r then [q.f: =0; q.r: =0]
                else q.f: = q.f + 1;
end;
```

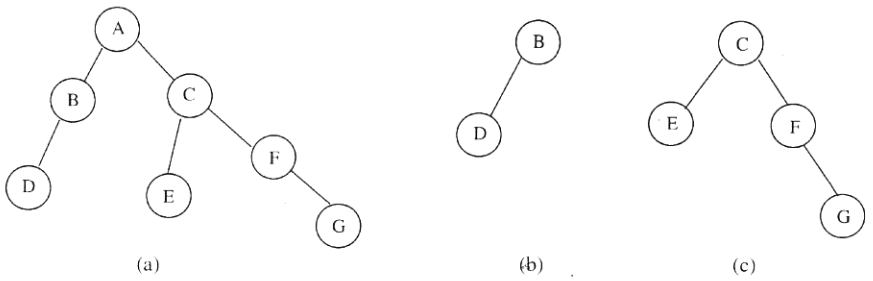
3) 置队空的运算

置队空的运算,非常简单,只要将队的头指针和尾指针均赋为0就可以了。

六、二叉树

1. 二叉树的定义

树的度为2的有序树。它可以是空树,或者是一棵由一个根结点和两棵互不相交的分别称作左子树和右子树所组成的非空树,左或右子树又同是二叉树。所以二叉树的定义是递归定义。如下图所示,常见的二叉树。



2. 二叉树的性质

1) 二叉树的第N层上最多有2^{N-1}个结点

2) 深度为K的二叉树最多有2^K-1个结点

具有N个(N>0)结点的完全二叉树的深度为$\log_2(N+1)$的整数值。

3. 二叉树的存储结构

1) 顺序存储

将一棵二叉树从根结点开始,从左到右顺序编号,其编号为数组的下标,其对应下标变量存放该结点的数据。上图(a)的顺序存储结构如下表:

```
type node = record
              da:char;
              l, r: integer;
            end
var         sta: array[1..n] of node;
            i, j, k: integer; ch:char;
```

	1	2	3	4	5	6	7
DA	A	B	C	D	E	F	G
L	2	4	5	0	0	0	0

R	3	0	6	0	0	7	0

2）链接存储

用链接表存储二叉树，每个结点由三部分组成：数据域、左地址域、右地址域。

数据域	左指针	右指针

左指针、右指针分别指向该结点的左孩子和右孩子的地址域。

4. 二叉树的运算

二叉树的运算主要包括：建立（生成）二叉树、二叉树的遍历、求二叉树的深度、输出二叉树。

1）生成二叉树的算法

① 建立一棵如上图（a）的二叉树。用链接表的存储方式

② 生成二叉树的方法有多种，这里介绍用广义表表示二叉树的方法。上图（a）的广义表的表示为：

A(B(D), C(E,F(, G)))

③ 依次输入字符，第一个为根结点，然后按照完全二叉树的顺序，输入各结点的孩子，设每个结点均有左、右两个孩子（用 LEFT,RIGHT 指向），如果没有孩子，则指向 0；最后输入“@”作为结束符号。

二叉树生成算法程序：

```
procedure creat (head);                    {头指针,变参}
  m=7;
  type stlist=^obnode
       obnode =record
                 da:char;
              left, right:stlist;
       end;
  var s:array[1..m] of stlist;
      p: stlist; top,k: integer; ch:char;
begin
  read (ch);      {读第一个字符}
  top:=0;
  while ch<  >'@' do
   begin
  case ch of
  'a'..'z': begin new (p); p^.da:=ch;p^.left:=nil; p^.right:=nil;
                  if top=0 then begin top:=1; head:=p; end
```

```
else case k of
          1: s[top]^.left:=p;
          2: s[top].right:=p;
     end; {case k of}
     end; {case 'a'..'z'}
'(':begin top:=top+1; s[top]:=p; k:=1; end;
')': top:=top-1;
',': k:=2;
end; {case ch of}
read(ch);
end;
```

输出二叉树的过程相当于前序遍历二叉树。

2) 二叉树的遍历

二叉树的遍历运算是树的运算基础。遍历的含义是指按照一定的次序访问树中的所有结点,并且每个结点仅被访问一次的过程。遍历二叉树的方法分为三种:这里我们给出相应的算法实现过程,读者再进一步用高级语言完善算法。

其变量和类型定义为:

```
TYPE LIST =^ NODE;
     NODE = RECORD
              DA: CHAR;
              LEFT, RIGHT: STLIST;
    END;
VAR HEAD: LIST;
```

(1) 前序遍历

前序遍历的基本方法:先访问根结点,再访问左子树,最后访问右子树。前序遍历的算法是一个递归过程,其算法如下:

```
procedure preorder (head) ;                {head 为指针类型}
  begin
    if head < > nil then write(head^.da);  {访问根结点}
    preorder (head^.left) ;                {遍历左子树}
    preorder (head^.right);                {遍历右子树}
  end;
```

图(a)前序遍历的结果是:(A, B, D, C, E, F, G)

(2) 中序遍历

中序遍历的基本方法:先访问左子树,再访问根结点,最后访问右子树。

其算法如下:

```
procedure inorder (head);                  {head 为指针类型}
  begin
```

```
    if head < > nil then
    inorder (head^. left) ;                    {遍历左子树}
    write(head^. da) ;                         {访问根结点}
    inorder (head^. right) ;                   {遍历右子树}
  end;
```

图(a)中序遍历的结果是:(D, B, A, E, C, F, G)

(3) 后序遍历

后序遍历的基本方法:先访问左子树,再访问右子树,最后访问根结点。

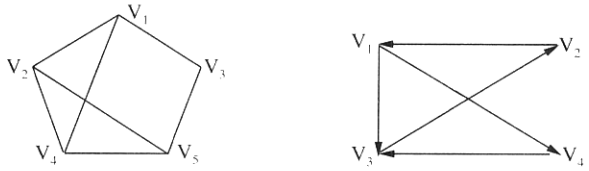
```
procedure postorder (head) ;                   {head 为指针类型}
  begin
    if head < > nil then
    postorder (head^. left) ;                  {遍历左子树}
    postorder (head^. right) ;                 {遍历右子树}
    write(head^. da) ;                         {访问根结点}
  end;
```

图(a)后序遍历的结果是:(D, B, E, G, F, C, A)

七、图

1. 图的概念

图的定义比较复杂,这里可以用简单语言描述其含义:图是由顶点和边组成的,边表示顶点之间的内在联系,如图(a)所示是一种简单的无向图,图(b)是有向图。



图(a)是用5个顶点7条边来表示顶点之间的联系:

(1) 顶点:V_1、V_2、V_3、V_4、V_5, 5个顶点;

(2) 边:(V_1, V_2)、(V_1, V_3)、(V_2, V_4)、(V_2, V_5)、(V_3, V_5)、(V_4, V_5), (V_1, V_4),7条边。

图(b)是一种带有方向性的图,其顶点和边之间的联系:

(1) 顶点: V_1、V_2、V_3、V_4, 4个顶点

(2) 边:(V_1, V_3)、(V_1, V_4)、(V_2, V_1)、(V_3, V_2)、(V_4, V_3), 5条边。

2. 图的存储结构

图的存储结构可以有两种表示方式:邻接矩阵和邻接表

1) 邻接矩阵表示,是一种将顶点之间的边的关系用矩阵方式表示,如果有N个顶点,则表示为:

$$A[I,J]=\begin{cases}1 & (\text{顶点 I 与顶点 J 之间有联系})\\ 0 & (\text{顶点 I 与顶点 J 之间无联系})\end{cases}$$

图(a)、(b)的邻接矩阵如下:

$$A=\begin{array}{c}\\1\\2\\3\\4\\5\end{array}\begin{array}{c}\begin{array}{ccccc}1&2&3&4&5\end{array}\\\begin{bmatrix}0&1&1&1&0\\1&0&0&1&1\\1&0&0&0&1\\1&1&0&0&1\\0&1&1&1&0\end{bmatrix}\end{array}\qquad B=\begin{array}{c}\\1\\2\\3\\4\end{array}\begin{array}{c}\begin{array}{cccc}1&2&3&4\end{array}\\\begin{bmatrix}0&0&1&1\\1&0&0&0\\0&1&0&1\\0&0&1&0\end{bmatrix}\end{array}$$

如果是带权的图,则顶点间有关联的边用具体数字替换。

图的邻接矩阵存储结构,用程序设计语言很容易实现。利用一维数组存放顶点,用二维数组存放顶点之间的关联——边,图(a)算法如下:

```
const n=5;                                          {顶点数}
      e=7;                                          {边数}
type citi=array[1..n,1..n] of 0..1;
     vex=array[1..n] of integer;
var a: citi;   v: vex;
procedure creat1 (ga, gv);
  begin
    ① for x:=1 to n do read (gv[x]);                {读入顶点的数据}
    ② for x:=1 to n do
    ③ for y:=1 to n do
       ga[x, y]:=0;                                 {邻接矩阵初始化}
    ④          for k:=1 to e do
              begin
    ⑤           read(i, j);                         {读入边上的两端点序号}
    ⑥           ga[i, j]:=1; ga[j, i]:=1;           {无向图具有对称边}
            end;
    end;
```

如果是带权的图,则⑤、⑥两步用 read(i, j, w); ga[i, j]:=w;无对称性。

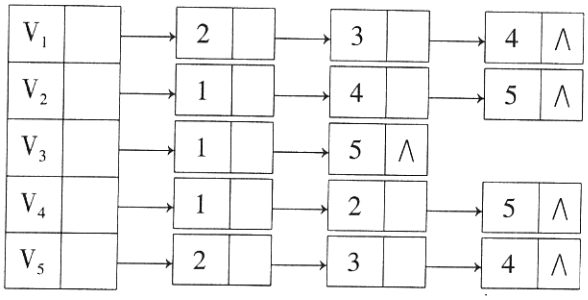
2) 邻接表的表示

用邻接表表示图的存储结构的方法是:

① 建立一个一维数组,数组的下标表示顶点,由该顶点建立一个单向链表;

② 每个单向链表均以顶点为头结点,凡是与该顶点有直接关联的顶点链接到此链表中。

如下表为图(a)的链接表存储结构:



图的邻接表可以有多种形式，有关邻接表存储结构的算法和程序请读者自己完成。

3. 图的操作

对于图的操作主要是遍历图。所谓图的遍历是指从图的某个顶点出发，按照一定的搜索方法，访问图中的每一个顶点，且仅访问一次的操作。根据搜索方法的不同，图的遍历常用两种方法：深度优先遍历法和广度优先遍历法。根据存储方式的不同，图的遍历的算法也不相同，这里我们用邻接矩阵的存储结构介绍深度优先遍历和广度优先遍历的算法。

1）深度优先遍历的算法：

① 从某个顶点出发开始访问，被访问过的顶点作相应的标记，并输出被访问顶点号；

② 从被访问顶点出发，搜索与该点有边的关联的某个邻接点，并且此邻接点未被访问过，再从此邻接点出发搜索与该邻接点有关联的下一个顶点，即重复搜索访问，直到所有点都被访问过为止。

算法过程：

```
procedure shendu(i);                          {i 是开始被访问的顶点}
  begin
    write(i);
    v[i]:=true;
    for j:=1 to n do
      if (ga[i,j]=1) and (not v[j]) then shendu(j); {递归调用}
  end;
```

图(a)深度优先遍历的结果是：(设从 V_1 开始搜索遍历)

(V_1, V_2, V_4, V_5, V_3)

2）广度优先遍历的算法：

① 从某个顶点出发开始访问，被访问过的顶点作相应的标记，并输出被访问顶点号；

② 从被访问顶点出发，依次搜索与该顶点有边的关联的所有未被访问的邻接点，并作相应的标记，说明已被访问过；

③ 再依次根据②中所有被访问的邻接点，访问与这些邻接点相关的所有未被访问过的邻接点，直到所有顶点被访问为止。根据算法描述，可以用队列完成。

算法过程：

```
procedure guangdu(i);
  begin
    {初始化队列为空——q 清空};
```

```
write(i);                                  {访问过的顶点}
v[i]:=true;                                {标记初始点 VI 已被访问过}
insert(q, i);                              {被访问过的顶点 VI 入队}
repeat
  ① k:=delete(q)                           {队首元素出队,第一次是 I 赋给 K}
  ② for j:=1 to n do
      if (ga[k,j]=1) and (not v[j]) then
        begin
          write(j);                        {访问一个邻接点}
          v[j]:=true;                      {标记被访问过的邻接点}
          insert(q, j);                    {顶点 j 入队}
until 队列为空(q);
```

图(a)广度优先遍历的结果是:(设从 V_1 开始搜索遍历)

(V_1, V_2, V_3, V_4, V_5)

八、算法

计算机对问题的求解过程是通过一系列命令来完成的,这种为了完成某个任务而编写的命令的有序集合,我们称之为程序。在设计程序过程中需要考虑对问题求解的方法和步骤,对问题求解的过程和步骤,我们称之为算法。算法的优劣将影响程序运行的效率和执行的结果。没有算法,也就谈不上程序设计。而构成算法的要素是数据的结构和操作方法及步骤。著名计算机专家沃思曾说过:数据结构 + 算法 = 程序设计。下面我们就近几年竞赛中的问题,探讨问题求解的中的数据结构和算法。

【竞赛试题】

2000 年　初中组基础题　第 4 题

算法是指(　　)

(A) 为解决问题而编写的计算机程序　　(B) 为解决问题而采取的方法与步骤

(C) 为解决问题而需要采用的计算机语言　　(D) 为解决问题而采用的计算方法

【问题分析与答案】本题正确答案选择(B),算法是指人们为了解决问题而选取的方法和实施步骤,而程序设计只是用计算机去实现问题求解的一种手段。计算机语言则是程序设计的基础,计算方法是在解决问题过程中所需要的数学模式等。

1995 年　初中组基础题　第 2 题

请将以下程序段表示的计算公式写出来(假设 X 的值已给出)

```
e:=1;
a:=1;
for n:=1 to 10 do   ┐
  a:=a*x/n;         ├ 循环体
  e:=e+a;           ┘
endfor;
```

写出所表示的公式。

【问题分析与答案】

① 此程序结构比较简单，用伪代码语言编写的，其结构分为二段：变量初始化、数据处理；

② 在阅读程序中，抓住程序的关键：循环体所完成功能是什么，再将前后联系起来，就可以得到本题的正确答案；

③ 本程序关键：重复做十次，完成 A：= A * X/N，E：= E + A 显然是累乘和累加即：计算 X^N/N，并将每次结果相加，得到 $1 + X + X^2/2 + X^3/3 + \cdots + X^N/N$ 计算公式，本题中 N 值为 10。

本题考察学生循环结构程序知识及累加和累乘的知识。

1995 年　初中组基础题　第 3 题

列举一个算法，使算法的解能对应相应的问题。

例如，设问题为：学生答题，答对一题可得 10 分，答错一题则要扣去 5 分，输入答对的题数(M)与答错的题数(N)，求最后得分(S)是多少？列举出相应算法为：

```
x:=10;
y:=5;
read(m, n);
s:=x*m-y*n;
```

现有以下问题：用五角钱换成 5 分、2 分与 1 分的硬币，可有多少种换法？请列出问题的算法。

【问题分析与答案】

① 对于本问题，可用穷举方法实现：即将所有可能的解一一列出，找出符合条件的解；

② 可以这样思考：将五角钱全部换成 5 分硬币最多需要 10 个，全部换成 2 分需要 25 个，全部换成 1 分需要 50 个，将五分、二分、一分硬币按不同数目有机组合，凑成五角，就能够得到一组解，多个组合就可以得到多组解。所以可以用循环搜索的穷举方法，其算法如下：

③
```
n:=0 ;
for a:=0 to 10 do
 for b:=0 to 25 do
    begin
     c:=50-a*5-b*2;
     writeln('5--', a:8, '2--', b:8, '1--', c :8);
      n:=n+1;
 end;
writeln('n=', n);
```

1995 年　初中组基础题　第 4 题

已知如下 N*(N+1)/2 个数据，按行的顺序存入数组 A[1], A[2], ……中：

A_{11}

A_{21} A_{22}

A_{31} A_{32} A_{33}

……

A_{N1} A_{N2} A_{N3} …… A_{NN}

其中:第一个下标表示行,第二个下标表示列。若 A_{ij}(I≥J, J, I=1, 2, …, N)存储在A[K]中,试问:K和I, J之间的关系如何表示?给定K值(K≤N*(N+1)/2)后,写出能决定相应的I, J值的算法。

【问题分析与答案】

① K和I, J之间的关系是:

这是一个下三角矩阵,I>=J时,A_{IJ}存储单元前已有I-1行共有(1+2+3+…(I-1))=I*(I-1)/2个元素,所以当前存储单元处于第K个单元与I, J的关系是:

$$K = I*(I-1)/2 + J \quad \{当前列\},$$

如A_{22}应存储在A[3],K=2*(2-1)/2+2=3,结论正确,A_{32}应存储在A[5], K=3*(3-1)/2+2=5,所以结论正确。

② 给定K值(K≤N*(N+1)/2)后,写出能决定相应的I, J值的算法。

当给定K值后,我们可以利用循环搜索方法找到K所对应的行列值I, J,算法如下:

```
read(k);
for i:=1 to k do
  for j:=1 to i do
    if k=(trunc(i*(i-1)/2)+j) then write(k,'——', i, j);
```

当i<=j时,每行元素个数=行数,所以j循环次数<=i。

1995年 初中组基础题 第5题

有红、黄、黑、白四色球各一个,放置在一个内存编号为1、2、3、4四个格子的盒中,每个格子放置一只球,它们的顺序不知。甲、乙、丙三人猜测放置顺序如下:

甲:黑编号1,黄编号2; 乙:黑编号2,白编号3; 丙:红编号2,白编号4。

结果证明甲乙丙三人各猜中了一半,写出四色球在盒子中放置情况及推理过程。

【问题分析与答案】

① 本题是数学推理问题,考察学生推理分析问题的能力。根据题意:如果甲的后半部分对即,则黄编号为2,那么乙必定是白编号3为正确,丙只能选择红编号2,这样红与黄色编号冲突,推理出错。

② 事实上甲猜中的前半部分黑编号1,乙则选择猜中的是白编号3,丙则选择猜中号为红编号2,那么黄则是编号4,推理正确。

1995年 高中组基础题 第2题

列举一个问题,使问题的解能对应相应的算法

例如对算法 x:=10;

```
y:=5;
read(m, n);
s:=x*m-y*n;
```

可列举出如下的问题：

学生答题，答对一题可得 10 分，答错一题则要扣去 5 分，输入答对的题数(M)与答错的题数(N)，求最后得分(S)是多少？

现有以下算法：

```
k:=0 ;
for i:=0 to 10 do
k:=k+(50-i*5)div 2+1
```

请列出一个相应的问题。

【问题分析与答案】

本题灵活性大，主要考察学生对知识的理解和灵活应用，答案不惟一，但其实质是一样的。

在分析问题时注意循环变量在循环体中的作用，本题参加了循环运算，影响计算的结果，所以在列出对应问题时不仅要考虑循环次数及累加运算，还需要考虑循环变量对运算结果的影响。这里我们可以理解为 5 角钱换成 5 分、2 分、1 分的硬币有多少种换法。

如，I=2 表示 5 分硬币 2 枚则有对应 2 分、1 分的硬币取法有 21 种。累加所有的换法即为问题的解。

1995 年　高中组基础题　第 3 题

有标号为 A、B、C、D 和 1、2、3、4 的 8 个球，每两个球装一盒，分装 4 盒。标号为字母的球与标号为数字的球有着某种一一对应的关系(称为匹配)并已知如下条件：

① 匹配的两个球不能在一个盒子内；

② 2 号匹配的球与 1 号球在一个盒子里；

③ A 号和 2 号球在一个盒子里；

④ B 匹配的球和 C 号球在一个盒子里；

⑤ 3 号匹配的球与 A 号匹配的球在一个盒子里；

⑥ 4 号是 A 或 B 号球的匹配球；

⑦ D 号与 1 号或 2 号球匹配。

请写出这四对球匹配的情况。

【问题分析与答案】

本题是一个推理题，考察学生对问题分析判断和逻辑推理能力。

(1) 由③可知 2 号球与 A 球放在一个盒中，所以 A 球只能与 1、3、4 球匹配，如果 A 与 1 号匹配，则由②、⑤可知 1 号球要放在两个盒中，所以 A 球不能与 1 匹配；若 A 球与 3 球匹配，则由⑤得 A 号球与 3 号球放在一个盒中，所以 A 球只能与 4 号球匹配；

(2) B 球仅能与 1、2、3 号球匹配，若与 1 号球匹配，则由④可知 1 号球需和 C 球在一个盒子中与条件②矛盾；若与 2 号球匹配也与条件④矛盾，所以 B 球仅能与 3 号球匹配；

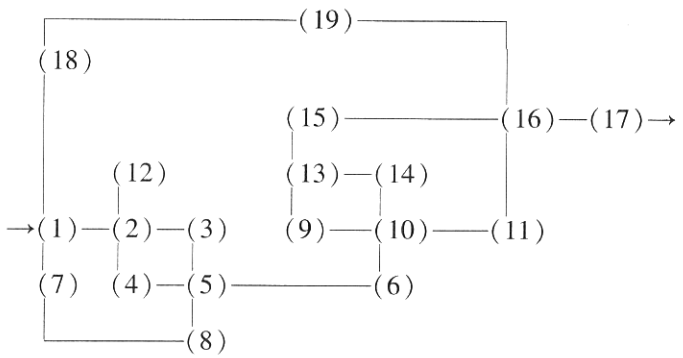
(3) 由⑦可知，D 球仅能与 1、2 号球匹配，若与 1 号球匹配，则由②得到 1 号球与 2 号球的匹配球在一个盒子，则 C 与 1 号球在一个盒子内，这样与④条件矛盾，所以 D 球仅能 2 号球匹配，因而得 C 则和 1 球匹配。

四对球匹配情况的正确答案如下表:

A	B	C	D
4	3	1	2

1995 年　高中组基础题　第 4 题

从入口(1)到出口(17)的可行路线图中,数字标号表示关卡:



现将上面的路线图,按记录结构存储如下:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
NO	1	2	18	7	3	12	4	19	8	5	13	16	6	14	15	9	17	…
PRE	0	1	1	1	2	2	2	3	4	5	6	8	10	11	11	11	12	…

请设计一种能从存储数据中求出从入口到出口经过最少关卡路径的算法。

【问题分析与答案】

① 该题是一个路径搜索问题,根据图示,从入口(1)到出口(17)可以有多种途径,其中最短的路径只有一条,那么如何找最短路径是问题的关键;

② 根据题意,用一维数组存储各关卡号(设 NO),用另一个数组存储访问到某关卡号的前趋关卡号(设 PRE);

③ 由于本题是一个典型的图的遍历问题,此题采用的是图的广度优先遍历算法,并利用队列的方式存储顶点之间的联系。即访问一个点,将其后继结点入队,并存储它的前趋结点,直到最后从(17)点出口;

④ 从最后出口的关卡号(17)开始回访它的前趋关卡号,则回返的搜索路径便是最短路径(跳过许多不必要搜索的关卡);

⑤ 从列表中可以看出出口关卡号(17)的被访问路径最短的是:

(17)← (16)←(19)←(18)←(1)←开始

⑥ 根据题意,要求从存贮数据中写出从入口到出口的最少关卡路径的算法是:

从队列的最后一个关卡号开始,依次回访它的前驱顶点,所得到的路径即为最短路径。

算法如下:

```
i:=1;
```

```
while no[i] < > 17 do
    i: = i + 1;
repeat
    write('(',no[i],')');
    write('←');
    i: = pre[i];
until i = 0;
```

由于篇幅限制，我们希望读者根据问题分析将图的遍历算法补充完整。

1996 年　初中组基础题　第 6 题（高中组基础第 4 题）

已知：ack(m, n)函数的计算公式如下：

$$ack(m,n)=\begin{cases}n+1 & m=0\\ ack(m-1,1) & n=0\\ ack(m-1,ack(m,n-1)) & m\neq 0 \text{ 且 } n\neq 0\end{cases}$$

请计算：初中组：ack(1, 2)与 ack(2, 2)的值。

高中组：计算 ack(1, 3)、ack(2, 4)、ack(3, 3)、ack(3, 4)。

【问题分析与答案】

① 本题考察学生对递归函数算法知识的理解，考察学生是否掌握了计算机递归算法的执行过程。

②不少学生在竞赛中，没有弄清递归调用中栈的变化，因而往往结果出错；

③ 计算方法如下：

ack(1, 2) = 4

↓

→ack(0, ack(1, 1)) = 4

↓

→ack(0, ack(1, 0)) = 3

↓

→ack(0, 1) = 2

ack(2, 2) = 7

用同样方法可以求得 ack(2, 2)的值。

ack(2, 2)

→ack(1, ack(2, 1))→ack(1, 5)→ack(0, ack(1, 4))→ack(0, 6) = 7

↓

→ack(1, ack(2, 0))→ack(1, 3)→ack(0, ack(1, 2)) = 5

↓

→ack(1, 1) = 3

↓

→ack(0, ack(1, 0)) = 3

↓

→ack(0, 1) = 2

高中组竞赛的问题：ack(1, 3)、ack(2, 4)、ack(3, 3)、ack(3, 4)方法同上，请读者自己

完成。

1996 年　初中组基础题　第 7 题

请写出对应计算如下算式的程序段:$Y = A_N X^N + A_{N-1} X^{N-1} + \cdots\cdots A_1 X + A_0$

【问题分析与答案】

① 本问题的算法比较简单,实质是多项式的求值运算,主要是累乘和累加;

② 本题的算法技巧是 X^N 的计算:减少运算次数用:T:=T*X;

③ 程序如下:

```
const n=10;
var a: array[1..n] of integer;
    x, y, t: integer; s:real;
begin
  read(x);  {输入 X 的值}
  read (a[0]);  s:=a[0]; t:=1;
  for y:=1 to n do
    begin
      read(a[y]);  {输入系数}
      t:=t*x;  s:=s+t*a[y];
    end;
    writeln(s);
end.
```

1996 年　初中组基础题　第 8 题(高中组基础题第 5 题)

有 N×N 个数据组成如下方阵:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & \cdots\cdots & A_{1N} \\ A_{21} & A_{22} & A_{23} & \cdots\cdots & A_{2N} \\ A_{31} & A_{32} & A_{33} & \cdots\cdots & A_{3N} \\ \cdots\cdots & \cdots\cdots & & & \\ A_{N1} & A_{N2} & A_{N3} & \cdots\cdots & A_{NN} \end{bmatrix}$$

并已知:$A_{IJ} = A_{JI}$,现将 A_{11}, A_{21}, A_{22}, A_{31}, A_{32}, A_{33},…存储在一维数组 A[1], A[2], …, A[(N*(N+1))/2]中。试问:任给 I,J 怎样求出 K 来,使得 A[K]的值正好是 A_{IJ},请写出由 I,J 计算 K 值的表达式。

【问题分析与答案】

该题是一个对称矩阵,题目要求只存储对称矩阵的下三角矩阵的值,并将其转换成一维数组存储,求出一维数组下标 K 的值与二维数组下标的 I,J 之间的关系。这样问题的解与 1995 年初中组基础题的第 4 题实质是一样的,不再赘述,K 和 I,J 之间的关系如下:

这是一个下三角矩阵存储,I >= J 时,A_{IJ} 存储单元前已有 I-1 行共有(1+2+3+…(I-1))=I*(I-1)/2 个元素,所以当前存储单元处于第 K 个单元与 I,J 的关系表达式:

K = I * (I - 1)/2 + J {当前列},

如 A_{22} 应存储在 A[3], K = 2 * (2 - 1)/2 + 2 = 3,结论正确,A_{32} 应存储在 A[5], K = 3 * (3 - 1)/2 + 2 = 5,结论正确。

1996 年　初中组基础题　第 9 题(高中组基础第 6 题)

已知:A_1, A_2, …, A_{81} 共有 81 个数,其中只有一个数比其他数大,要用最少的比较运算次数,把这个值大的数找出来(假设两个数比较一次能决定出大于、小于或等于这三种情况)请将以下算法补充完整:

第一步:　$s1 = A_1 + A_2 + \cdots + A_{27}$

$s2 = A_{28} + A_{29} + \cdots + A_{54}$

第一次比较(s1, s2):

$s1 > s2$　　取 K = 0

$s1 < s2$　　取 K = 27

$s1 = s2$　　取 K = 54

第二步:　$s1 = A_{K+1} + A_{K+2} + \cdots + A_{K+9}$

$s2 = A_{K+10} + A_{K+11} + \cdots + A_{K+18}$

第二次比较(s1, s2):

$s1 > s2$　　取 K = ①

$s1 < s2$　　取 K = ②

$s1 = s2$　　取 K = ③

第三步:　$s1 = A_{K+1} + A_{K+2} + A_{K+3}$

$s2 = A_{K+4} + A_{K+5} + A_{K+6}$

第三次比较(s1, s2):

$s1 > s2$　　取 K = ④

$s1 < s2$　　取 K = ⑤

$s1 = s2$　　取 K = ⑥

第四步:　$s1 = A_{K+1}$

$s2 = A_{K+2}$

第四次比较(s1, s2):

$s1 > s2$　　⑦ 为最大数

$s1 < s2$　　⑧ 为最大数,$s1 = s2$　⑨ 为最大数。

【问题分析与答案】

本题中指出只有一个数比其他数大,所以,为了减少比较次数,本题采用等分原理,即分段求和的方法:

第一步:首先分为 0 ~ 27、28 ~ 54 两段,分别计算这两段的数据和 s1, s2。如果 s1 > s2,则说明大数在第一段数中,所以 K = 0,否则,如果 s2 > s1 则大数在第二段,K = 27,否则 s1 = s2, K = 54;

通过第一步,可以知道大数的基本位置:或 0 ~ 27,或 28 ~ 54,或 55 ~ 81;

再通过第二步,将大数所在范围内的数分为三段(每段 9 个数),将前两段数的和

$$s1 = A_{K+1} + A_{K+2} + \cdots + A_{K+9}$$

$$s2 = A_{K+10} + A_{K+11} + \cdots + A_{K+18}$$

s1 与 s2 进行大小比较,同理可以取得新的 K 值,将大数所在的范围缩小到 9 个数中。

通过第三步,再将大数所在的那一段数(剩 9 个数)分为为三段(每段 3 个),再计算

$$s1 = A_{K+1} + A_{K+2} + A_{K+3}$$

$$s2 = A_{K+4} + A_{K+5} + A_{K+6}$$

新的 s1 与新的 s2 进行大小比较,同样可以取得新的 K 值,将大数所在的范围缩小在 3 个数中,通过第四步比较便可以得到大数的确切位置。因此,K 的值是通过不断缩小搜索范围的某段数据起点位置。

所以填空为:

① K　② K+9　③ K+18　④ K　⑤ K+3

⑥ K+6　⑦ A_{K+1}　⑧ A_{K+2}　⑨ A_{K+3}

1996 年　高中组基础题　第 7 题

下面是一个利用完全二叉树特性,用顺序表来存储的一棵二叉树,结点数据为字符型(结点层次号从小到大,同一层从左到右顺序存储,#表示空结点,@ 表示存储数据结束)。现要求画出对应该存储结构的二叉树示意图。

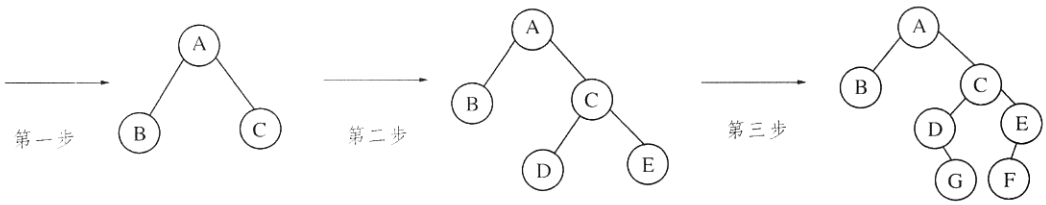
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	B	C	#	#	D	E	#	#	#	#	#	G	F	@

【问题分析与答案】

① 在第二节数据结构中我们介绍了二叉树的概念,完全二叉树只是结点排列是从上到下从左到右的存储结构;

② 二叉树的每个结点最多只有两个孩子,其叶子结点的左右孩子为空,根结点仅有一个;

③ 根据题意,是用顺序表存储一棵二叉树的,其对应的二叉树示意图如下:



④ 根据完全二叉树定义和顺序存储特点,画出第一步图:根结点加左右孩子;接着应该在 B 结点画出它的左右孩子,然而根据存储结构,其左右孩子为空,再在 C 点处画出其左右孩子 D、E,得到如第二步所示的图;同理,我们可以画出第三步的图,这样便完成整个二叉树的建立工作。

1997 年　初中组基础题　第 4 题(高中基础第 4 题)

设数组 A[10..100, 20..100]以行优先的方式顺序存储,每个元素占 4 个字节,且已知 A[10, 20]的地址为 1000,则 A[50, 90]的地址是_____。

【问题分析与答案】

本题考察学生对二维数组顺序存储结构地址的理解。

① 数组大小:A[10..100, 20..100]共占有 91 行,每行有 81 个下标变量共计 7371 个存储空间;

② A[10, 20]的起始地址为 1000, A[10, 21]的起始地址是 1004, A[10, 22]的起始地址是 1008,所以 A[10, 100]的起始地址是:1000 + (100 - 20) * 4 = 1320;

③ 每行的起始地址的计算公式:1000 + (X - 10) * 324 + (100 - 20) * 4;

④ 任意一行一列的起始地址计算公式:1000 + (X - 10) * 324 + (J - 20) * 4;

⑤ 所以 A[50,90]的起始地址是 1000 + 40 * 324 + 70 * 4 = 14240。

1997 年　初中组基础题　第 5 题

下面是一个求:1/1 + 1/2 + 2/3 + 3/5 + 5/8 + 8/13 + 13/21 + 21/32 + …前 20 项的和的程序段,试将程序补充完整:

```
s: =0; a: =1; b: =1;
for k: =1 to 10 do
 begin
  s: = ____①____;
  a: = ____②____;
  s: = ____③____;
  b: = ____④____;
 end;
writeln(s);
```

【问题分析与答案】

本题的作用为计算分数的倒数和,要完成填空,首先需要观察问题中分数变化规律:

(1) 每一项的分子是前一项分母,而每一项的分母数据是前一项分子、分母数值的和,利用数据间的这样的关系可以产生每一项分数;

(2) 将所有的分数求和:用重复累加的方法实现,所以本题用循环结构完成;

(3) 在完成此题填空时注意两个方面:

前 20 项数据和:仅循环 10 次,说明循环体每次加两个分数;

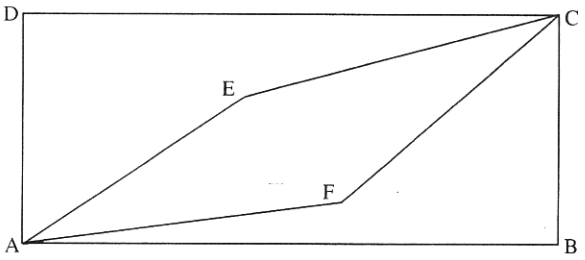
分数的分子、分母产生的方法:第一项分数 A/B; A: = A + B, B/A 产生第二项分数;B: = B + A 生成新一项的分母;

(4) 完成填空:

① S + A/B　　② A + B　　③ S + B/A　　④ B + A

1997 年　初中组基础题　第 8 题(高中基础题第 6 题)

下图中用点表示城市,点与点之间的联系表示城市间的道路:



试问:

① 能否找出一条从A出发,经过图中所有道路一次后又回到出发点的通路来?

② 能否从A出发,找出去每个城市且只去一次的通路来?若能,则写出通路,否则说明理由。

【问题分析与答案】

本题是一个简单的图的搜索问题,它希望通过学生自己所具有的数学知识,进行分析推理,得到问题的解,其目的考察学生的思维能力。

问题①的解答是可以找到这样的通路(答案不惟一),如:A→D→C→E→A→F→C→B→A。从A点到达D、E、F、B的四种出发路线都可以找到满足条件的通路。其原因连接A点的线段是偶数,才可以找到通路。

问题②的解答是只去一次的通路不存在。其原因在于要经过D、E、F、B四点城市,必须从A点或C点出发,而从A或C只能到达其中的3个点(每个点只能去一次),因此要找到这样的通路是不可能的。

1997年　初中组基础题　第9题(高中基础题第7题)

为了便于处理表达式,常常将普通表达式(称为中缀表示)转换为前缀{运算符在前,如X/Y写为/XY}和后缀{运算符在后,如X/Y写为XY/}的表达形式。

在这样的表示中可以不用括号即可确定求值的顺序,如:

$$(P+Q)*(R-S) \rightarrow *+PQ-RS \text{ 或} \rightarrow PQ+RS-*$$

① 试将下面的表达式改写成前缀与后缀的表示形式:

<A> A+B*C/D　　<B> A-C*D+B∧E

② 试将下面的前缀表示还原成中缀的表示形式,同时写出后缀表示:

+△A *B△C {前缀式中△表示一元运算符取负号,如△A表示(-A)}

【问题分析与答案】

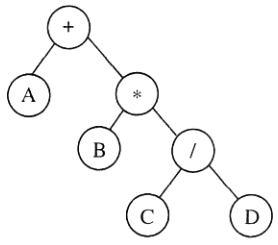
本题是二叉树的遍历的一种应用。二叉树的遍历有三种方法:

前序遍历、中序遍历、后序遍历。在前面的背景知识介绍数据结构与算法的有关内容中,已经介绍了遍历的基本算法。对问题①我们可以将中缀表达式写成一棵表达式树,然后按照前序、后序遍历的方法,遍历二叉树,就可以得到前缀表达式和后缀表达式。

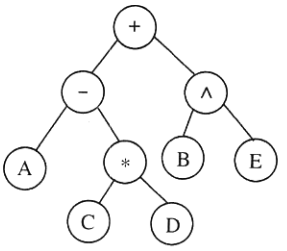
问题①解决方法是将<A>　A+B*C/D和<B>　A-C*D+B∧E用中缀表达式树表示如下图:

前序遍历图(a)和图(b)结果是:<A>为+A*B/CD　<B>为+-A*CD∧BE

后序遍历图(a)和图(b)结果是:<A>为ABCD/*+　<B>为ACD*-BE∧+

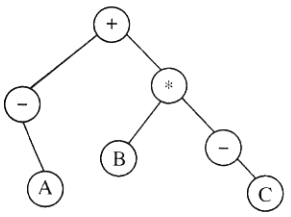


(a)



(b)

问题②解决方法是根据前序遍历的含义:先根结点,再左子树,最后遍历右子树的定义,画出二叉树如右图,然后根据中序遍历、后序遍历的定义,写出中缀表达式和后缀表达式。



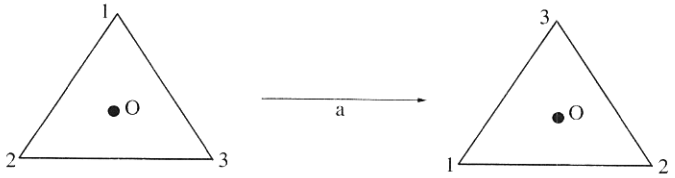
所以前缀表达式:+△A ＊B△C 的中缀及后缀表达式是:

中缀表达式:(－A)＋B＊(－C);

后缀表达式:A△BC△＊＋;

1997 年　初中组基础题　第 10 题

一个将角编了号的正三角形可以绕着外心 O(中心)逆时针旋转 120°,如下图所示:



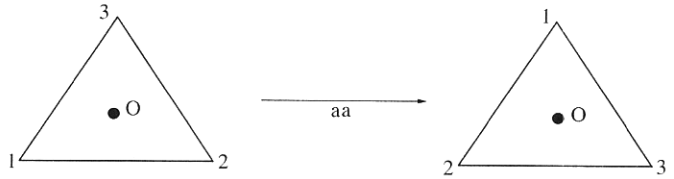
图一　　图二

如果将这一旋转用字母 a 来表示,看作运算对象,同时用 aa 或 a^2 表示旋转 120°后再旋转 120°,也就是说将连续运动看作乘法运算,那么三角形状态(可简称为元素)即可与运动表达式关联起来,请回答:

① 如果将图一的原始三角形连续旋转 120°n 次,简单地表示为 a^n(n 为任意自然数),试求 a^n 的值(指三角形旋转后的结果状态);

② 如果将下面的旋转看作是 a 的逆元素,记为 a^{-1},则有 $a^{-1}=a^2$

试求:a^{-n}



图三

【问题分析与答案】

本题考察学生应用数学知识处理问题的能力:

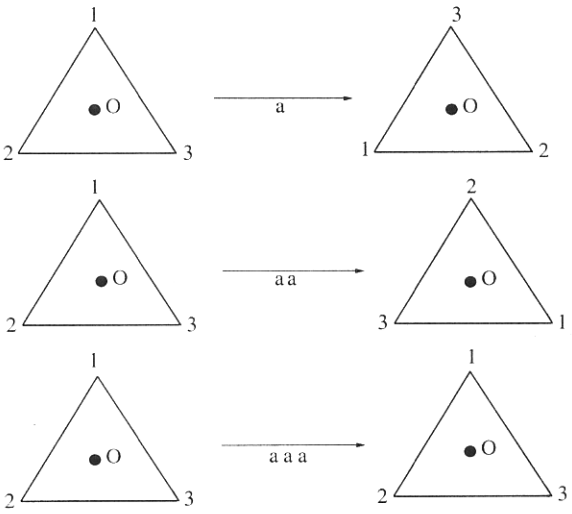
① 命题者用符号代表需要处理的对象,将图一到图二的旋转变化用 a 来表示了正三角形

旋转了120°;

② 根据题意,将处理的对象的连续旋转120°的运动定义为乘法运算;

③ 当原始三角形连续旋转120°n次,则表示为a^n则其变化的规律如下:

本题中三角形的旋转只有三种情况:



无论旋转多少次最终只能是以上图所示的三种情况,所以正确答案是:

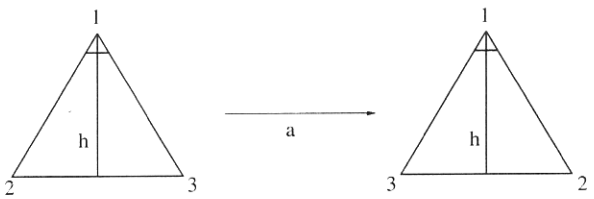
a^n的值仅有三类:
$$\begin{cases} a & \text{当 N mod 3 = 1 时} \\ a^2 & \text{当 N mod 3 = 2 时} \\ a^3 & \text{当 N mod 3 = 0 时} \end{cases}$$

a^{-n}的值仅有三类:
$$\begin{cases} a^2 & \text{当 N mod 3 = 1 时} \\ a & \text{当 N mod 3 = 2 时} \\ a^3 & \text{当 N mod 3 = 0 时} \end{cases}$$

1997年　高中组基础题　第8题

一个将角编了号的正三角形可以进行如下二种运动:

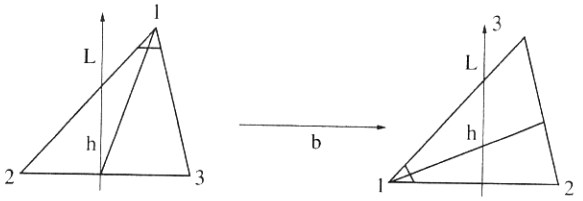
(1) 沿过顶点1的高h翻转180°,我们将这个运动用字母a来表示:



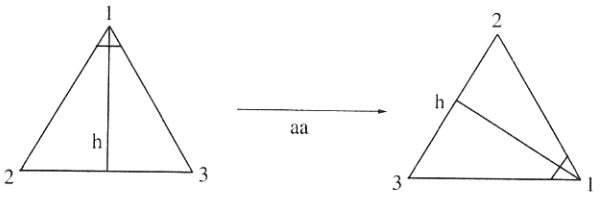
图一　　图二

(2) 沿过三角形的外心,垂直于三角形所在平面的有向轴L(注意:三角形翻转时L轴也随着翻转的),按右手法则旋转120°(右手法则是指用右手大拇指指向L轴的方向,由其余四

指决定旋转方向的法则），我们将这样的运动用字母 b 来表示：



如果将 a，b 作为运算对象，并将两个运动连续进行看作是一种运算（这里不妨也称为乘法）则对图一的三角形而言，aa 的结果便成为：

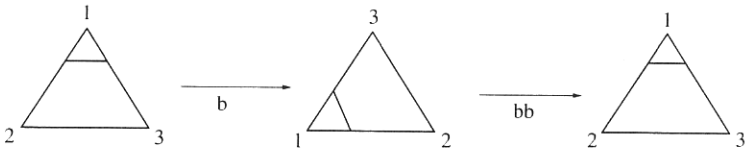


若将运动前后的三角形状态简称为元素，那么三角形状态就可与运动的表达式关联。据此，请回答下列问题：

① 从图一的三角形的原始状态出发，可以运动出多少种不同状态的三角形，试写出最简单的运算表达式（借助于 a，b 与乘法运算）；

② 这样定义的乘法运算是否符合交换律与结合律？

③ 如果将三角形的某种状态运动回到原始状态称之为该元素的逆元素，例如：



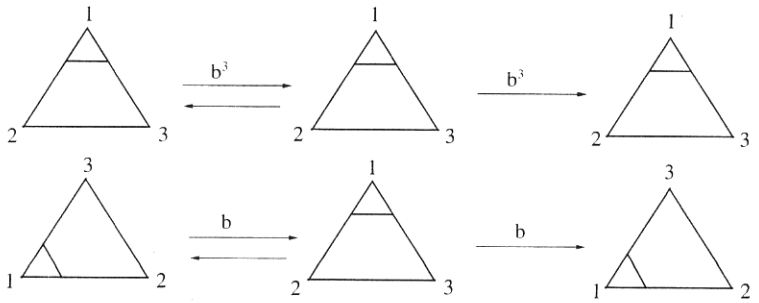
所以 bb 的逆元素为 b，可以表示为 $(bb)^{-1}=b$

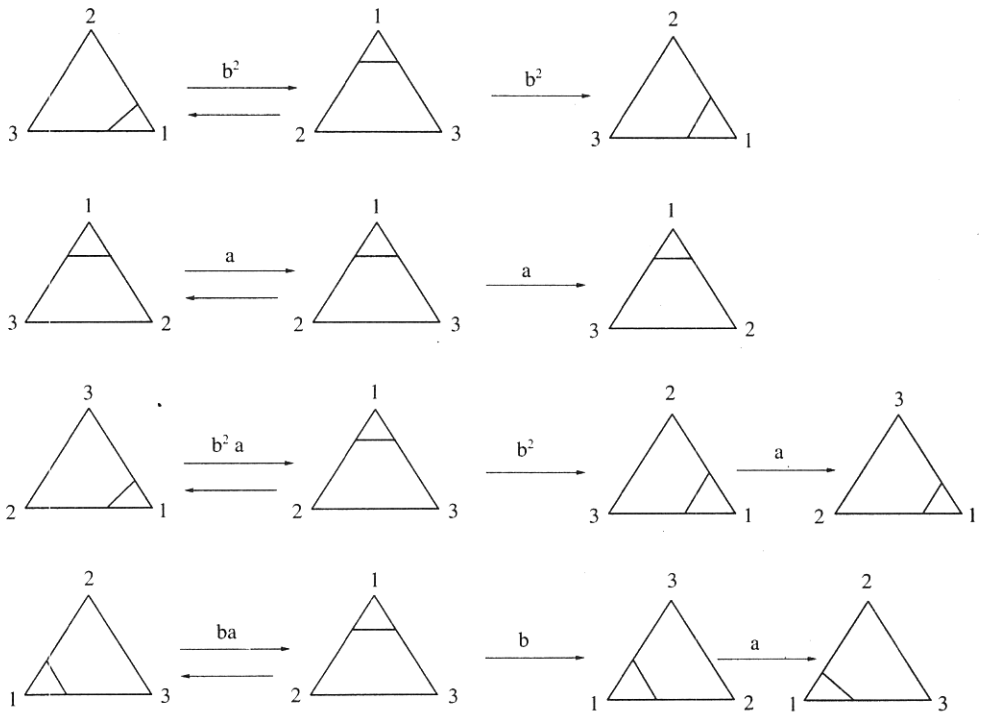
试求：(1) $a^{-1}=$　　(2) $(ab)^{-1}=$　　(3) $((aa)a)^{-1}=$　　(4) $b^{-1}=$

【问题分析与答案】

本题是将初中问题复杂化，目的在于考察学生能否将现实中的问题抽象为数学模型，利用数学知识和计算机知识解决实际生活中的问题。

问题①：该题需要设计 6 个处理对象（用符号 a，b 表示）如下：





三角形的连续的运动表示了乘法运算,其乘法的规则如下:

乘法	b^3	b	b^2	a	b^2a	ba
b^3	b^3	b	b^2	a	b^2a	ba
b	b^2	b^2	b^3	ba	a	b^2a
b^2	b^2	b^3	b	b^2a	ba	a
a	a	b^2a	ba	b^3	b	b^2
b^2a	b^2a	ba	a	b^2	b^3	b
ba	ba	a	b^2a	b	b^2	b^3

正三角形的六种状态可写为:

① b, ② b^2,(bb)或 aba, ③ b^3, (bbb)或(aa) ④ a ⑤ b^2a(bba)或 ab

⑥ ba 或 ab^2, (abb)

问题②:由上述乘法表,可以看出,本题所定义的乘法运算不符合乘法交换律,例如:

$ab \neq ba$,其原因是 $ab = b^2a$,而 $ba = ba$,所以不符合交换律。

乘法的结合律是满足的,例如:(ab)a = a(ba)

因为 $(ab)a = (b^2a)a = b^2$ $a(ba) = b^2$

所以 (ab)a = a(ba)

问题③:从乘法表可以看出每个元素都有逆元素如下:

$(b^3)^{-1} = b^3$ $(b^2)^{-1} = b$ $(b)^{-1} = b^2$

$(a)^{-1} = a$ $(b^2a)^{-1} = b^2a$ $(ba)^{-1} = ba$

另外，从表中我们还可以发现：

$(ab)^{-1}=(b^2a)^{-1}=b^2a=b^{-1}a^{-1}$；

$(ba)^{-1}=b^{-1}a^{-1}$，$(aa)^{-1}=a^{-1}a^{-1}$；　　$(bb)^{-1}=b^{-1}b^{-1}$

所以$(a)^{-1}=a$，　　$(ab)^{-1}=b^2a=ab$　　$((aa)a)^{-1}=(a^3)^{-1}=a$

$b^{-1}=b^2$

从本题的解答中，可以看出问题的求解过程有时比较复杂，必须分析掌握其实质，找出规律性，问题就可以迎刃而解了。

1998年　初中组基础题　第9题（高中组基础题第8题）

下列if语句中，endif表示相应if的结束：

```
y=0
if  x<0
    then  y=5
    else  if  x<10
              then  y=10
              if x<100
                  then  y=100
                  endif
              else  y=200
              endif
endif
```

试指出：

当x=80时，运行的结果是_____，

当x=5时，运行结果为_____。

（A）y=9　　（B）y=5　　（C）y=10　　（D）y=100　　（E）y=200

【问题分析与答案】

本题关键是考察学生对条件语句的嵌套结构知识的理解。注意题意中明确指出：endif表示相应的if结束，因此本题的正确答案是：

当x=80时，运行结果y=200，其执行过程是：x=80，不满足x<0因此执行else后的if x<10条件语句，不满足，跳过if x<100的条件语句执行else y=200，所以当x=80时，y=200。

当x=5时，运行结果是y=100。

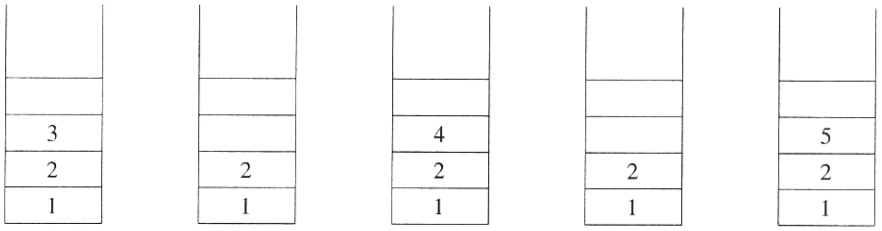
1998年　初中基础题　第10题（高中组基础题第10题）

设栈S的初始状态为空，现有5个元素组成的序列{1，2，3，4，5}，对该序列在S栈上依次进行如下操作（从序列中的1开始，出栈后不再进栈）：进栈、进栈、进栈，出栈、进栈、出栈、进栈。试问出栈的元素序列是_____。

（A）{5，4，3，2，1}　　（B）{2，1}　　（C）{2，3}　　（D）{3，4}

【问题分析与答案】

本题是考察学生对栈知识的理解,现用如下表示栈结构的图,说明操作过程:



1,2,3 进栈　　3 出栈　　4 进栈　　4 出栈　　5 进栈

所以本题正确答案选择:{3,4}即(D)。

1998 年　初中组问题求解　第 1 题(高中组问题求解第 1 题)

已知一个数列 U_1, U_2, U_3, …, U_n, …往往可以找到一个最小的 K 值和 K 个数 a_1, a_2, …, a_n 使得数列从某项开始都满足:$U_{n+k}=a_1U_{n+k-1}+a_2U_{n+k-2}+\cdots+a_kU_n$　　(A)

例如对斐波拉契数列 1,1,2,3,5,…可以发现:当 K=2, $a_1=1$, $a_2=1$ 时,从第 3 项起(即 n>=1)都满足 $U_{n+2}=U_{n+1}+U_n$。试对数列 1^2, 2^2, 3^2, …, n^2, …求 K 和 a_1, a_2, …, a_k 使得(A)式成立。

【问题分析与答案】

根据题意,可以用解方程的方法求得解:

(1) 可以先设定 K=2,列出方程组:$\begin{cases}a_1*1^2+a_2*2^2=3^2\\a_1*2^2+a_2*3^2=4^2\end{cases}$　得不到 a_1, a_2 整数解,

(2) 再设 K=3 得到方程组$\begin{cases}a_1*0^2+a_2*1^2+a_3*2^2=9\\a_1*1^2+a_2*2^2+a_3*3^2=16\\a_1*2^2+a_2*3^2+a_3*4^2=25\end{cases}$　解得 $a_1=1$, $a_2=-3$, $a_3=3$

所以本题的正确答案是:K=3, $a_1=1$, $a_2=-3$, $a_3=3$,满足 $U_{n+3}=U_{n+1}-3*U_n+3*U_{n+2}$

1998 年　初中问题求解　第 2 题

某班有 50 名学生,每位学生发一张调查卡,上面写 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;.

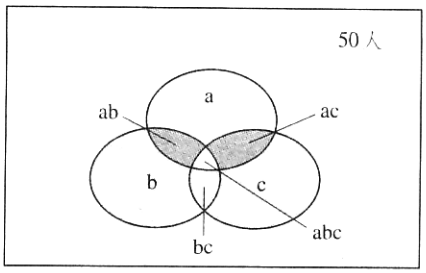
(1) 读过 a 的人数是_____　　(2) 一本书也没有读过的人数是_____。

【问题分析与答案】

本题是推理题,可以利用数学中的集合概念推理得到结果。

在推理过程中主要避免重复计算或丢失数据,我们可以利用数学工具较快的求出解:集合的并集、交集及补集运算(如下图)。

正确答案是(1)读过 a 的人数是:8+2+2=12 人;(2)一本书没有读过是 50-20=30 人。



1998年　初中问题求解　第3题

任给自然数n，k，1≤k≤9，按如下计算步骤求序列 $x_j x_{j-1}\cdots\cdots x_0$ 的步骤：

（1）j=0
（2）如果n>=k则转第3步，否则转第7步
（3）x_j=n　mod k
（4）n　=n div k
（5）j=j+1
（6）回第2步
（7）x_j=n
（8）结束

重复将被除数除以k（进制）求余数和新的被除数直到被除数为零

反向输出余数得到转换后的数

试求当：n=1998，k=3时，　$x_j x_{j-1}\cdots\cdots x_0$ 之值。

【问题分析与答案】

（1）根据题目内容和程序的算法，可以知道这是一个典型的将十进制数转换成任意进制数的程序（1≤k≤9）；

（2）n是被转换的十进制数，k是需要转换的进制数；

（3）转换的方法：重复将被除数除以k（进制），求余数将余数存入 x_j 中，并计算新的被除数，直到被除数为零；

（4）反向输出余数得到转换后的数。

本题正确答案：$x_6x_5x_4x_3x_2x_1x_0=(2202000)_3$，将十进制转化成三进制数。

1998年　高中问题求解　第2题

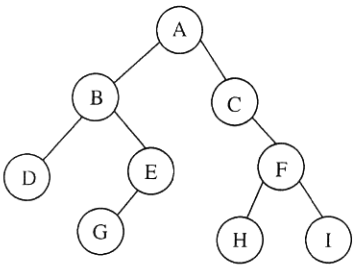
给出一棵二叉树的中序遍历：DBGEACHFI与后序遍历：DGEBHIFCA画出此二叉树。

【问题分析与答案】

（1）本题是二叉树的遍历问题，中序遍历是先访问左子树，再访问根结点，最后访问右子树；

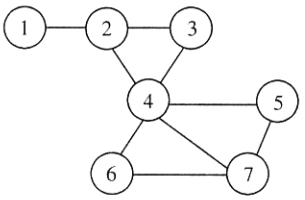
（2）解决问题的方法是：从后序遍历可以知道该树的根结点是A，再从中序遍历可以知道DBGE在根结点A的左子树、CHFI是在根结点A的右子树；

（3）根据中序遍历方法，初步确定DBGE和CHFI位置，再根据后序遍历进行结点位置的调整。如下图所示。



1998 年　高中问题求解　第 3 题

用邻接矩阵表示下面的无向图:



【问题分析与答案】

(1) 本题主要考察学生数据结构中图的概念以及图的存储结构方面的知识;

(2) 图分为有向图和无向图,其存储方式常用两种结构:链接表和邻接矩阵;

(3) 邻接矩阵结构:定义一个顺序存储方式——二维数组,数组的下标表示顶点之间的关联,其结构如下图所示的邻接矩阵表示,是一种将顶点之间的边的关联用矩阵方式表示,如果有 N 个顶点,则表示为:

$$A[I,J]=\begin{cases}1 & (\text{顶点 I 与顶点 J 之间有联系})\quad (1\leqslant I,J\leqslant N)\\ 0 & (\text{顶点 I 与顶点 J 之间无联系})\quad (1\leqslant I,J\leqslant N)\end{cases}$$

$$A=\begin{array}{c} \\ 1\\2\\3\\4\\5\\6\\7\end{array}\begin{array}{c}\begin{array}{ccccccc}1&2&3&4&5&6&7\end{array}\\ \begin{bmatrix}0&1&0&0&0&0&0\\1&0&1&1&0&0&0\\0&1&0&1&0&0&0\\0&1&1&0&1&1&1\\0&0&0&1&0&0&1\\0&0&0&1&0&0&1\\0&0&0&1&1&1&0\end{bmatrix}\end{array}$$

如果是带权的图,则顶点间有关联的边用具体数字替换。

图的邻接矩阵存储结构,用程序设计语言很容易实现。利用一维数组存放顶点,用二维数组存放顶点之间的关联——边,其算法如下:

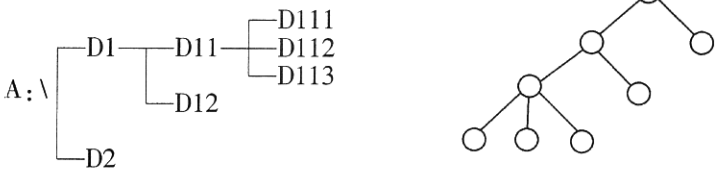
```
const n = 7;   {顶点数}
      e = 9;   {边　数}
type citi = array [1..n, 1..n] of 0..1;
```

```
        vex = array [1..n] of integer;
var a: citi; v: vex;
procedure creatl (ga, gv);
  begin
    ① for x:=1 to n do read (gv[x]);          {读入顶点的数据}
    ② for x:=1 to n do
    ③ for y:=1 to n do
         ga[x, y]:=0;                          {邻接矩阵初始化}
    ④ for k:=1 to e do
         begin
    ⑤      read(i, j);                         {读入边上的两端点序号}
    ⑥      ga[i, j]:=1; ga[j, i]:=1;           {无向图具有对称边}
       end;
end;
```

如果是带权的图，则⑤、⑥两步用 read(i, j, w); ga[i, j]:=w;无对称性。

1999 年　初中组问题求解　第 1 题

在磁盘的目录结构中，我们将与某个子目录有关联的目录数称为度. 例如下图左：



该图表达了 A 盘的目录结构：D1, D11, ..., D2 均表示子目录的名字。在这里，根目录的度为 2，D1 子目录的度为 3，D11 子目录的度为 4，D12，D2，D111，D112，D113 的度均为 1。不考虑子目录的名字，则可简单的图示为如上图右所示的树结构：

若知道一个磁盘的目录结构中，度为 2 的子目录有 2 个，度为 3 的子目录有 1 个，度为 4 的子目录有 3 个。

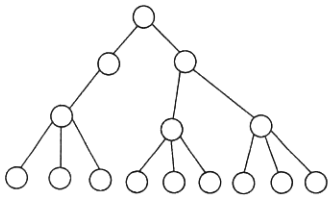
试问：度为 1 的子目录有几个？

【问题分析与答案】

（1）根据题意度是指与某个子目录有关联的目录数；

（2）要求度为 1 的子目录个数，可以根据题意，画出对应的目录树，便可以得到度为 1 的子目录个数；

（3）本题目录树如下图所示：度为 1 的子目录有 9 个。



1999 年　初中组问题求解　第 2 题

根据 Nocomachns 定理,任何一个正整数 n 的立方一定可以表示成 n 个连续的奇数的和。例如:

$1^3 = 1$

$2^3 = 3 + 5$

$3^3 = 7 + 9 + 11$

$4^3 = 13 + 15 + 17 + 19$

在这里,若将每一个式中的最小奇数称为 X,那么当给出 n 之后,请写出 X 与 n 之间的关系表达式:_____

【问题分析与答案】

(1) 当 n = 1 时,最小奇数是 x = 1, 1 个奇数;

(2) 当 n = 2 时,最小奇数是 x = 3, 2 个奇数;

(3) 当 n = 3 时,最小奇数是 x = 7, 3 个奇数,最小奇数 x = 3 * (3 - 1) + 1 = 7;

(4) 当 n = 4 时,最小奇数是 x = 13, 4 个奇数,最小奇数前已有 1 + 2 + 3 个奇数,最小奇数:第 7 个奇数,x = (4 * (4 - 1)/2) * 2 + 1 = 13;

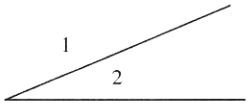
(5) 当 n = k 时,最小奇数前已有 1 + 2 + 3 + 4 + … + k - 1 个奇数,最小奇数第 k 个奇数,x = (k * (k - 1)/2) * 2 + 1。

即最小奇数 X 与 n 之间的关系表达式为:

$$x = n * (n - 1) + 1$$

1999 年　高中组问题求解　第 1 题

将 Ln 定义为求在一个平面中用 n 条直线所能确定的最大区域数目。例如:当 n = 1 时,L1 = 2,进一步考虑,用 n 条折成角的直线(角度任意),放在平面上,能确定的最大区域数目 Zn 是多少?例如:当 n = 1 时,Z1 = 2(如图所示)



当给出 n 后,请写出以下的表达式:

Ln = _____

Zn = _____

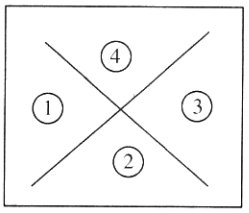
【问题分析与答案】

本题实质是求直线或折线将一个平面分成最大区域数,我们可以从两个方面考虑:

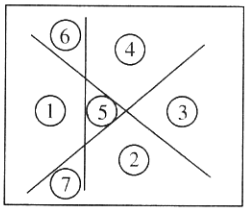
(1) 求在一个平面中用 n 条直线所能确定的最大区域数目;

(2) 求在一个平面中用 n 条折线所能确定的最大区域数目。

问题(1),通过几何知识可以知道 n 条直线两两相交,所构成的区域数目最大:



(a) n=2,L=4



(b) n=3,L=7

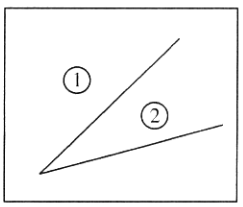
n=1,　　L1=2;　　F(1)=2;
n=2,　　L2=4;　　F(2)=F(1)+2;
n=3,　　L3=7;　　F(3)=F(2)+3;
n=4,　　L4=11;　　F(4)=F(3)+4;
n=5,　　L5=16;　　F(5)=F(4)+5;
……

得到公式:F(n)=F(n-1)+n ,两边相加后得:

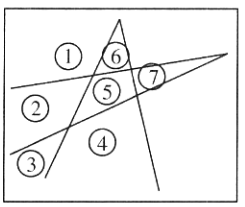
$$n+(n-1)+(n-2)+\cdots+2+1+1=n*(n+1)/2+1$$

所以当给出 n 后,最大区域数目是 Ln=n*(n+1)/2+1

(2) 用 n 条折线所能确定的最大区域数目是:



(c) n=1,z=2



(d) n=2,z=7

当 n=1,　　Zn=2　　F(1)=0+2
当 n=2,　　Zn=7　　F(2)=1*(2*2-1)+4
当 n=3,　　Zn=16　　F(3)=2*(2*3-1)+6
……

所以当 N=K 时,则　　F(K)=(K-1)*(2*K-1)+2*K
所以最大区域数目是:Ln=(n-1)*(2n-1)+2n

2000 年　初中组基础题　第 8 题

设循环队列中数组的下标范围是 1~n,其头尾指针分别为 f 和 r,则其元素个数为(　　)

(A) r-f　　(B) r-f+1　　(C) (r-f)MOD n +1　　(D) (r-f+n) MOD n

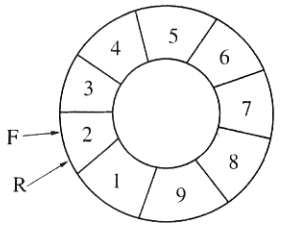
【问题分析与答案】

本题正确答案选择 D。

(1) 本题意在考察学生循环队列的知识,有许多学生容易错误选择 A、B,其主要原因对

于循环队列的结构以及循环队列中头、尾指针操作没有理解;

(2) 循环队列中,头指针和尾指针指向同一个单元,如图所示,因此其元素的个数是:



(r - f + n) mod n

例如:r = 2, f = 2, n = 0 元素个数为 0

r = 9, f = 1,元素个数为 8

2000 年　初中组基础题　第 9 题

在待排序的数据表已经为有序时,下列排序算法中花费时间反而多的是(　　)

(A) 堆排序　　(B) 希尔排序　　(C) 冒泡排序　　(D) 快速排序

【问题分析与答案】

本题正确答案选择(D),其原因在:

(1) 排序的方法有多种,其算法的好坏、时间的长短,大家在学习这部分知识时已经讨论过,这里我们重温一下其排序的特点:

● 堆排序:改进在枚举排序中数据的重复大小比较,它是用建立一棵二叉树(堆)的方法将结点(数据)按照一定顺序上升或下降(下沉),时间复杂性为 $0(n\log_2 n)$;

● 冒泡排序:N 个数排序,要将相邻两个数进行比较和交换,需要的时间复杂性为 $0(n^2)$,所以花费时间长;

● 希尔排序:属于插入排序,但对直接插入排序做了较大改进。基本思想是:将整个线性表进行分割,然后对每个子表分别进行插入排序。当整个线性表达到基本有序时,再最后对它进行一次插入排序。希尔排序的比较次数为:$0(n(\log_2 n)^2)$,因此,当 n 不很大时,其使用价值不大。

● 快速排序:基本思想是:通过一次分割,将线性表分成两部分,其中前一部分的所有元素值均不大于后一部分中的每一个元素值;然后对每一部分再进行分割,直到整个线性表有序为止。由此可知,快速排序的关键是线性表的分割。快速排序的复杂度为 $0(n\log_2 n)$。

(2) 通常情况下,数据的排序,人们常用快速排序法,然而当数据已经有序时,若再用快速排序方法,就不能体现少比较数据、交换数据的特点,需要将数据进行一一比较,这样快速排序蜕化为冒泡排序。

2000 年　高中组基础题　第 12 题

在有 n 个叶子节点的哈夫曼树中,其节点总数为(　　)

(A) 不确定　　(B) 2n - 1　　(C) 2n + 1　　(D) 2n

【问题分析与答案】

本题正确答案选择(B)。

(1) 在二叉树中,如果是一棵满二叉树,则树的总节点数与叶子节点的关系是:

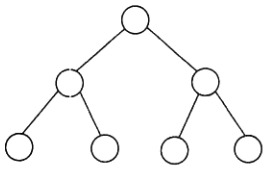
设叶子节点为 n,则总节点数 s:　　s = 2 * n - 1 个,如下图(A)。

(2) 哈夫曼树是一种特殊的满二叉树,因此若有 n 个叶子节点,则其总节点数也是2n - 1。

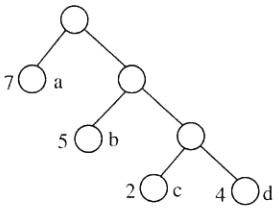
(3) 哈夫曼树又是最优二叉树,它是一类带权路径长度(WPL)最短的树。如有 4 个叶子结点 a、b、c、d,分别带权 7、5、2、4,则如图的二叉树为哈夫曼树,因为它的 WPL = 7 × 1 + 5

×2 +2 ×3 +4 ×3 =35,为最小,如下图(B)。

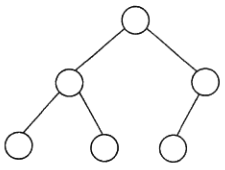
(4) 完全二叉树的叶子节点数与总节点数之间就不一定满足这样的关系,如下图(C)



(A)



(B)



(C)

2000 年　初中组基础题　第 14 题(高中组基础题第 15 题)

某数列有 1000 个各不相同的单元,由低至高按序排列,现要对该数列进行二分法检索,在最坏的情况下,需要视(　　)单元

(A) 1000　　(B) 10　　(C) 100　　(D) 500

【问题分析与答案】

正确答案选择(B),其理论根据在于:

(1) 二分法查找元素其基本思想:将数据元素个数对半分,将待查找的数与中间位置数相比较,若大于该中间位置的数,则在数据段的后半段检索,否则在数据段的前半段查找;

(2) 继续重复第(1)步的过程,只是数据范围不断减少,总是在原数据段的一半内查找;

(3) 正确答案选择(B),在最坏的情况下需要查看 10 个单元;

如查找数据较小则:中间位置为 MID

MID: =(1 +1000) DIV 2 =500;新 MID 是(1 +500) DIV 2 =250, …, 125, 62, 31, 16, 8; 4, 2, 1。

2000 年　初中组基础题　第 15 题(高中组基础题第 13 题)

已知数组 A 中,每个元素 A[I, J]在存储时要占 3 个字节,设 I 从 1 变化到 8,J 从 1 变化到 10,分配内存时是从地址 SA 开始连续按行存储分配的。试问:A[5, 8]的起始地址为(　　)

(A) SA +141　　(B) SA +180　　(C) SA +222　　(D) SA +225

【问题分析与答案】

正确答案选择(A),因为:

(1) 在前几届的竞赛中也有类似的顺序存储结构——数组的地址计算问题,只要掌握数组是顺序存储,占用连续的存储空间,并且注意到问题的要求按行存储分配空间,就能很快计算任意单元的起始地址。

(2) 存储空间分配如下:

A[1, 1]　A[1, 2]　A[1, 3]　…　A[1, 9]　A[1, 10]
A[2, 1]　A[2, 2]　A[2, 3]　…　A[2, 9]　A[2, 10]
…
A[8, 1]　A[8, 2]　A[8, 3]　…　A[8, 9]　A[8, 10]

根据题意,在A[5, 8]前4行计40个单元,而从第5行开始,从A[5, 1]至A[5, 7]共7个单元,这样至A[5, 8]单元的共有47个单元,其地址是SA + (47 * 3) = SA + 141。

2000年　初中组基础题　第17题(高中组基础题第17题)

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

【问题分析与答案】

正确答案选择(D)

(1) 本题主要考察学生对线性表的概念理解和链接存储结构的知识,所以首先需要知道什么是线性表,前面已叙述;

(2) 链接存储的特点,是将零散的存储空间通过指针域链接起来,因此链接存储单元一般至少有两个域:数据域和指针域(指向下一个结点),通过指针将结点链接后生成链接表。

所以存贮单元地址可以连续也可以不连续。

2000年　初中组基础题　第18题(高中组基础题第18题)

下列叙述中,正确的是(　　)

(A) 线性表的线性存储结构优于链表存储结构

(B) 队列的操作方式是先进后出

(C) 栈的操作方式是先进先出

(D) 二维数组是指它的每个数据元素为一个线性表的线性表

【问题分析与答案】

本题正确答案选择(D)。二维数组本身是一个M行N列的矩阵,每行、每列都可以看做一个线性表:如有一个二维数组A,它由M行N列组成那么

$$A = (a_1, a_2, \cdots, a_p) \qquad (P = M \text{ 或 } N)$$

其中某一个数据元素a_j可看成是一列向量的线性表

$$a_j = (a_{1j}, a_{2j}, \cdots, a_{nj}) \qquad 1 \leqslant j \leqslant n$$

$$A_{m\times n} = \left[\begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{bmatrix} \cdots \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix} \right]$$

或者把a_i看成是 是一个行向量的线性表

$$a_i = (a_{i1}, a_{i2}, \cdots, a_{in}) \qquad 1 \leqslant i \leqslant m$$

$$A_{m\times n} = ((a_{11}a_{12}\cdots a_{1n}), (a_{21}a_{22}\cdots a_{2n}), \cdots, (a_{m1}a_{m2}\cdots a_{mn}))。$$

所以二维数组每个数据元素可以看作一个线性表的线性表。

2000年　初中组基础题　第19题(高中组基础题第19题)

电线上停着两种鸟(A, B),可以看出两只相邻的鸟就将电线分为了一个线段。这些线段可分为两类:一类是两端的小鸟相同;另一类则是两端的小鸟不相同。已知:电线两个顶点上正好停着相同的小鸟,试问两端为不同小鸟的线段数目一定是(　　)

(A) 奇数　　(B) 偶数(C) 可奇可偶(D) 数目固定

【问题分析与答案】

本题的正确答案选择(B),该问题需要通过归纳分析,找出问题的规律性,才可以得到正确结论。根据题意,已知线段两端停着相同的鸟,设为(A),则:

	线段为不同小鸟的线段数
A————————B	
A————A————A	0
A————B————A	2
A——A——B——A	2
A——B——B——A	2
A——A——A——B——A	2
A——B——A——B——A	4
A——A——B——A——A	2

观察以上各行,我们通过增加鸟的只数的变化以及它们的排列的变化,可以发现,无论增加多少只鸟,只要两端的小鸟是相同的,则线段为不同的小鸟的线段数一定为偶数。其数学原理:

(1) 两端相同,增加一只不同鸟,产生两条两端不同小鸟的线段;

(2) 两端相同,增加二只不同小鸟,产生两条或四条两端不同小鸟的线段;

(3) 两端相同,在其中增加 N 只不同小鸟,由于线段两端是相同的鸟,通过对称排列,必定是偶数个两端为不同小鸟的线段。

2000 年　初中组基础题　第 20 题(高中组基础题第 16 题)

请仔细阅读下列程序段

```
var
a:array [1..3, 1..4] of integer;
b:array [1..4, 1..3] of integer;
x, y:integer;
begin
程序段 1
  for x:=1 to 3 do
    for y:=1 to 4 do
      a[x, y]:=x-y;
程序段 2
  for x:=4 downto 1 do
    for y:= 1 to 3 do
      b[x, y]:=a[y, x];
  writeln(b[3, 2]);
end.
```

上列程序段的正确输出是(　　)

(A) -1　　(B) -2　　(C) -3　　(D) -4

【问题分析与答案】

本题正确答案选择(A),这是一个阅读程序并写出程序的运行结果的问题,其思考方法可以按如下步骤:

(1) 首先阅读1给数组A赋值的程序段,得到如下的数字阵列(同学们在阅读程序中,应该注意变量的变化);

A[X, Y]	X/Y=1	X/Y=2	X/Y=3	X/Y=4
X=1	0	-1	-2	-3
X=2	1	0	-1	-2
X=3	2	1	0	-1

(2) 程序段2,将数组A的值行列转换后赋值给数组B,所以求数组元素B[3, 2]的值,就是计算数组A元素的A[2, 3]的值即为-1。

2000年　高中组基础题　第20题

一个文本屏幕有25列及80行,屏幕的左上角以(1, 1)表示,而右下角则以(80, 25)表示,屏幕上每一个字符占用两字节(byte),整个屏幕则以线性方式存储在电脑的存储器内,由屏幕左上角开始,位移为0,然后逐列存储。求位于屏幕(X, Y)的第一个字节的位移是(　　):

(A) (Y*80+X)*2-1　　(B) ((Y-1)*80+X-1)*2

(C) (Y*80+X-1)*2　　(D) ((Y-1)*80+X)*2-1

【问题分析与答案】

正确答案选择(B),本题所考核知识,其实质为数据的行列存储结构问题。所谓线性方式,即逐行或逐列,从左到右,从上到下的方式。

(1) 注意:本题左上角(1, 1)处位移为0,逐列存储屏幕上的字符;

(2) 位于屏幕(X, Y)的第一个字节的位移是:

前Y列所占用位移空间(Y-1)*80*2个字节,第Y列前有(X-1)*2个字节,因此位于屏幕(X, Y)的第一个字节的位移是

(Y-1)*80*2+(X-1)*2=((Y-1)*80+X-1)*2。

2000年　初中组问题求解　第1题(高中组问题求解第1题)

已知,按中序遍历二叉树的结果为:abc

问:有多少中不同形态的二叉树可以得到这一遍历结果,并画出这些二叉树。

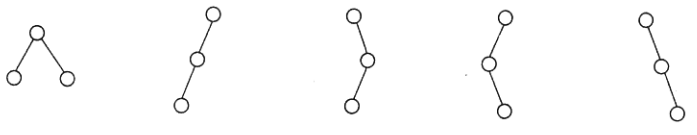
【问题分析与答案】

这道题考查了二叉树的概念及其二叉树的遍历,二叉树在NOI的大纲中是数据结构复赛考核的要求,在数据结构中非常重要,这道题关键在于明确二叉树的结构,下面的形式是符合要求的。

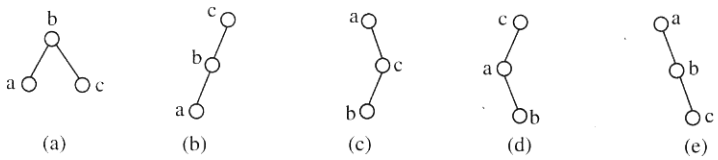
遍历是一种检索的方法,二叉树的遍历实质是按路径将二叉树每一个结点访问一次。通

常要求学生掌握三种遍历：前序遍历、中序遍历和后序遍历，它们的命名都是以根结点或父结点的先后次序来命名的。前序遍历的次序：先父结点，次左结点，后右结点；中序遍历的次序：先左结点，次父结点，后右结点；后序遍历的次序：先左结点，次右结点，后父结点。

从题目中可以知道：只有 a，b，c 三个结点，三个结点存在的可能形态，根据组合的方法有如下 5 种：



a，b，c 具体放置在哪个结点上呢？根据中序遍历的结果，逆向推导，二叉树(a)中按先左，次根，后右的方式遍历，则 a 在左结点，b 在根结点，c 在右结点；二叉树(b)中，对于 c 来说 b 是左结点，对于 a 来说 b 是父结点，遍历的顺序为：从 c 开始，先左结点，看结点 b，b 结点下一层左结点为 a，所以先读 a，再读父结点 b，b 没有右结点，再上行读 b 的父结点 c，所以 a，b，c 的位置如图二叉树(b)；依照这种方法，同样可以推导出其他二叉树中 a，b，c 的位置。这里不再讨论。



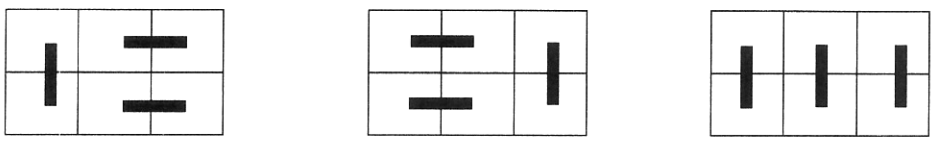
【思考题】

已知：按前序遍历二叉树的结果为：abc

问：有多少种不同形态的二叉树可以得到这一遍历结果，并画出这些二叉树。

2000 年　初中组问题求解　第 2 题

有 2×n 的一个长方形方格，用一个 1×2 的骨牌铺满方格。例如 n=3 时，为 2×3 方格。此时用一个 1×2 的骨牌铺满方格，共有 3 种铺法：



试对给出的任意一个 n(n>0)，求出铺法总数的递推公式。

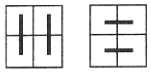
【问题分析与答案】

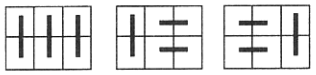
这道题是非常典型的组合数学的题目，也是著名的菲布拉契数列(0，1，1，2，3，5，8，…)的一种延伸的表现形式。旨在考查学生的递推能力和面对问题求解的思维能力。

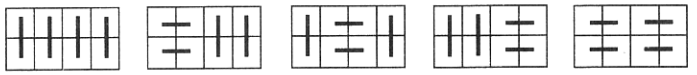
解这道题的方法有2种:

(1) 具体的n=1, 2, 3, 4, …开始,列举出结果,根据列举的部分结果进行猜测,推导出公式。这种推导"猜"和"凑"的成分较多。下面我们来具体实践一下:

当n=1时:  如左图,只有1种铺法,铺法总数表示为F(1)=1;

当n=2时:  如左图,只有2种铺法,铺法总数表示为F(2)=2;

当n=3时:  如左图,只有3种铺法,铺法总数表示为F(3)=3;



当n=4时: 如上图,只有5种铺法,铺法总数表示为F(4)=5;

当n=5时:有8种铺法,(图略)铺法总数表示为F(5)=8;

根据结果:1, 2, 3, 5, 8可以看出:

$$3=2+1;\quad F(3)=F(2)+F(1)$$

$$5=3+2;\quad F(4)=F(3)+F(2)$$

$$8=5+3;\quad F(5)=F(4)+F(3)$$

由此,推出公式:

$$F(n)=F(n-1)+F(n-2)$$

(2) 据组合数学常用的方法进行归纳和推导。

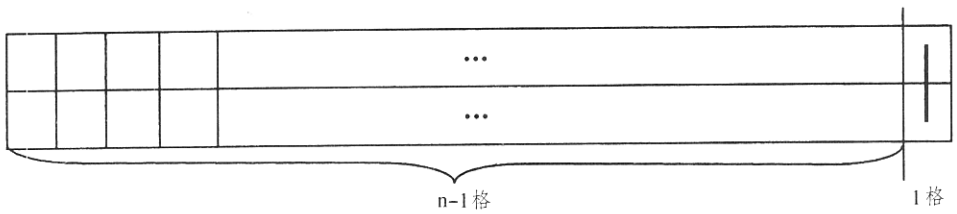
在分析n×2格时,我们递推的思想去考虑,有以下式子①:

$$F(n)=aF(n-1)+bF(n-2)+cF(n-3)+dF(n-4)+\cdots\ (a, b, c, d\cdots是常系数) \qquad ①$$

式子①的含义是:n×2格的路,铺法是全部(a种)的(n-1)×2格路的总铺法加上全部(b种)的(n-2)×2格路的总铺法加上……,注意排除重复的情况。

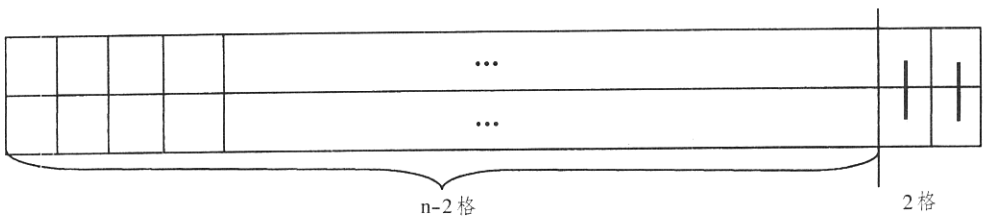
下面我们来具体分析以下这道题:

1) 我们将n格分为1格和n-1格来考虑,分析F(n-1)的情况,如下图:

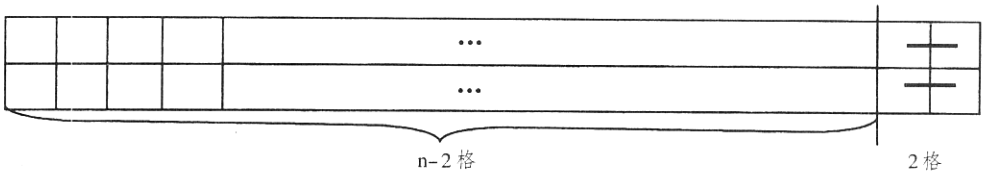


在1格中,只能有一种铺法,则式子①中的F(n-1)常系数a=1;

2) 将n格分为2格和n-2格来考虑,分析F(n-2)的情况,如下图:

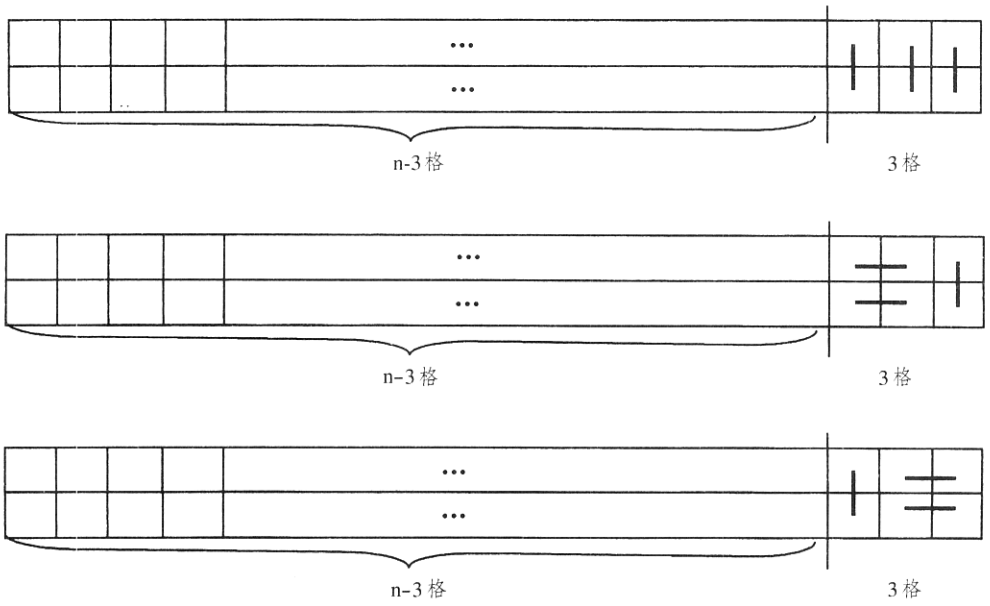


显然，这种铺法包含在第1）步的铺法之中，故此种铺法不再重复考虑；我们来看2格的另一种铺法，如下图：



这种铺法不同于第1）步的铺法，到此，2格的铺法已经全部列出，所以式子①中的F（n－2）常系数b＝1；

3）继续分析F（n－3）的情况：



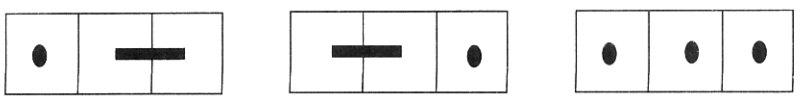
由上图可见，3格铺法都可以归结到1格和2格中去，所以F（n－3）格的所有与F（n－2）格和F（n－1）格重复。故式子①中的F（n－3）格的常系数c＝0。后面的情况我们就不再讨论了。

将a，b，c，…的具体值带入式子①中，得到我们要推导的公式：

$$F(n)=F(n-1)+F(n-2)$$

类似这样题目很多，比如下面一题：

【思考题】 有1×n的一个长方形方格，用一个1×2和1×1的骨牌铺满方格。例如n＝3时，为1×3方格。此时用1×2和1×1的骨牌铺满方格，共有3种铺法：



试对给出的任意一个n（n＞0），求出铺法总数的递推公式。请读者思考。

典型的“铺砖问题”,可以变化出许多种有趣的题目,例如,高中组问题解答(2)就是“铺砖问题”的一个变化类型。

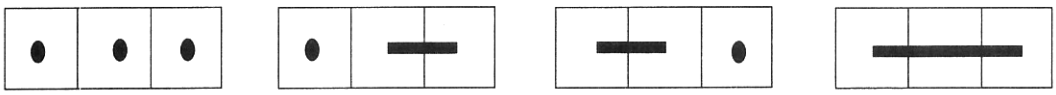
2000 年　高中组问题求解　第 2 题

设有一个共有 n 级的楼梯,某人每步可以走 1 级,也可以走 2 级,也可以走 3 级,用递推公式给出某人从底层开始走完全部楼梯的走法。例如:当 n = 3 时,共有 4 种走法:

即　1 + 1 + 1,　1 + 2,　2 + 1,　3。

阅读完这道题,我们完全可以将题目视为“铺砖问题”。请看转换后的题目内容:

有 1 × n 的一个长方形方格,用一个 1 × 1、1 × 2 和 1 × 3 的骨牌铺满方格。例如 n = 3 时,为 1 × 3 方格。此时用 1 × 1、1 × 2 和 1 × 3 的骨牌铺满方格,共有 4 种铺法:



试对给出的任意一个 n(n > 0),求出铺法总数的递推公式。

这道题的解法也有 2 种:

(1) 从具体的 n = 1, 2, 3, 4,…开始,列举出结果,根据列举的部分结果进行猜测,推导出公式。

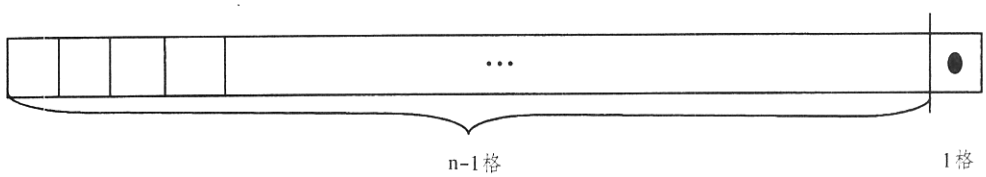
(2) 根据组合数学常用的方法进行归纳和推导。

第 1 种推导“猜”和“凑”的成分较多,容易出错。具体的解法,我们不再讨论,读者可以参看初赛普及组问题解答(2)的第 1 种解法,我们具体讨论第 2 种方法。

推导公式:

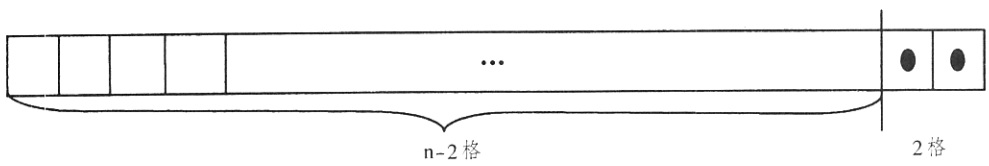
F(n) = aF(n - 1) + bF(n - 2) + cF(n - 3) + dF(n - 4) + …(a, b, c, d…是常系数)①

① 我们将 n 格分为 1 格和 n - 1 格来考虑,分析 F(n - 1)的情况,如下图:

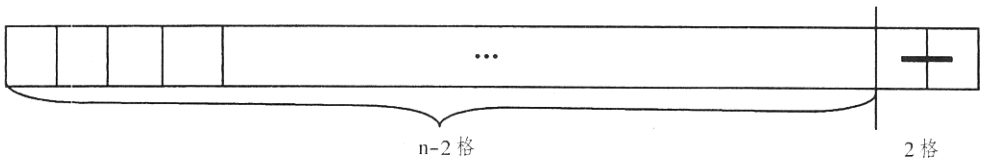


在 1 格中,只能有一种铺法,则式子①中的 F(n - 1)常系数 a = 1;

② 将 n 格分为 2 格和 n - 2 格来考虑,分析 F(n - 2)的情况,如下图:



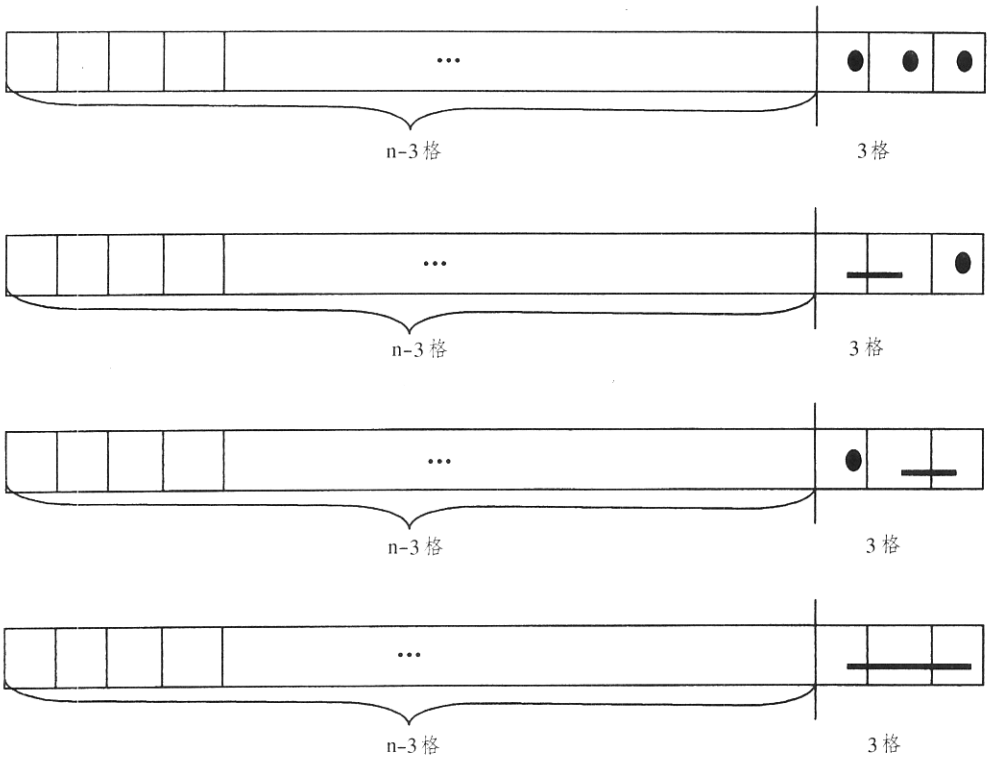
显然,这种铺法包含在第 1)步的铺法之中,故此种铺法不再重复考虑;我们来看 2 格的另一种铺法,如下图:



这种铺法不同于第1)步的铺法，到此，2格的铺法已经全部列出，所以式子①中的F(n-2)常系数b=1；

③ 继续分析F(n-3)的情况：

由下图可见，这些铺法都可以归结到1格或2格中去。



所以F(n-3)格的铺法只有1×3这1种方法。故式子①中的F(n-3)格的常系数c=1。后面的情况都是重复的，我们就不再讨论了。

将a，b，c，…的具体值带入式子①中，得到我们要推导的公式：

$$F(n)=F(n-1)+F(n-2)+F(n-3)$$

第二章　阅读程序写出运行结果

阅读程序、写运行结果这种题型较易失分，因为一旦最终结果不正确，将失掉全题的得分，这种题型主要考察选手：

（1）程序设计语言的掌握情况；

（2）数学运算能力；

（3）细心、耐心的心理品质。

解决这类问题的关键在于能够分析程序的结构以及程序段的功能。

完成这类题的一般方法和步骤是：

（1）从头至尾通读程序，大致把握程序的算法；

（2）通过给程序分段，理清程序的结构和层次，达到读懂程序的目的；

（3）阅读程序中特别注意跟踪主要变量的值的变化，可以用列表的方法，了解变量变化和程序的运行结果，注意发现规律。所谓列表法，就是将各变量名作为表头，在程序的执行过程中，将各变量值的变化记录在相应变量的下方。请看下面的程序：

```
 5 T=0:S=0
10 FOR A=3 TO 21 STEP 5
20 T=T+1:S=S+T+A
30 NEXT A
40 PRINT "T=";T,"S=";S
50 ENd
```

程序中出现了三个变量 A、T、S，因此表头就是 A、T 、S

表

执行的语句	A	T	S
5		0	0
10	3		
20（第一次执行循环体）		1	4
30	8（步长为5）		
20（第二次执行循环体）		2	14
30	13		
20（第三次执行循环体）		3	30
30	18		
20（第四次执行循环体）		4	52
30	23（超过终值，结束循环）		
40	输出结果 T=4　S=52		

（4）按照程序中输出格式的要求，写出运行结果；

（5）带着得到的结果回到程序进行检查。

在阅读程序时，应特别注意过程、函数所完成的子任务以及和主程序之间的参数传递关系。在阅读程序中，比较好的方法是首先阅读主程序，看其需要调用的过程或函数是什么，最后要求输出变量是什么；其次在阅读程序中，将较长的程序分成几个程序段（特别注意循环结构、判断结构），阅读理解各程序段的功能以及各程序段之间的关联；建议大家在读程序时，一定不能浮躁，要有足够的耐心，因为有些程序读起来是比较繁琐的。本章主要讨论这几年分区联赛中，阅读解程序并写出程序运行结果的竞赛题。

1996 年　初中基础题　第 4 题

阅读下列程序段，写出程序段运行后变量 x 的值。

```
x1:=3 ;
x2:=8 ;
for i:=1 to 5 do
  begin
    x:=(x1+x2)*2;
    x1:=x2 ; x2:=x;
  end;
writeln('x=',x) ;
```

循环结构，应用数据轮换方式，求两个数和的 2 倍。

【问题分析与答案】

本题程序结构简单，循环 5 次，用数据轮换求两个数的和，运行后变量 x 的值变化：

i=1　时　x= 22　　,x1=8　　, x2=22 ；

i=2　时　x= 60　　,x1=22　　, x2=30 ；

i=3　时　x=104　　,x1=30　　, x2=104；

所以当循环 5 次后，x 的值是 744

1996 年　初中组基础题　第 5 题（高中组基础题第 3 题）

阅读程序，写出程序段运行后数组元素 a1，a2，…，a11 中的值。

```
a[1]:=1;
a[2]:=1;  k:=1;
repeat
  a[k+2]:=1 ;
  for  i:=k  downto  2  do
    a[i]:=a[i] +a[i-1 ];
  k:=k+1;
  until  k>=10;
```

【问题分析与答案】

该程序是一个循环嵌套程序，在阅读该程序时：

（1）可以先理解内循环程序的功能：重复求从当前数组元素开始，求其本身原有的值与前

一个元素值的和;

(2) 将第(1)步工作,重复执行 10 次,输出数组元素 a1, a2, …, a11 中的值。

(3) 写程序结果:由于是输出变量 a1, a2, …, a11 中的值,而数组元素值不断变化,可以用列表方式求解。

	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11
k=1,	1	1	1								
k=2,	1	2	1	1							
k=3,	1	3	3	1	1						
k=4,	1	4	6	4	1	1					
			……								
k=10,	1	10	45	120	210	256	210	120	45	10	1

通过列出变量表,可以很快发现数据变化规律,得到计算公式,推导运行结果。本题数据运算规律是:当前行中的某一列数据是由前一行的前一列数与前一行的当前列数的和组成(除每一行的第一列和最后两列外),所以结果如上表。

1998 年　初中组写程序运行结果题　第 1 题(高中组写运行结果题第 1 题)

```
program  exp1  (input,output);
var  i, s, max:integer;
     a:array [1..10] of integer;
begin
     for i:=1 to 10 do read (a[i]);
     max:=a[1] ;s:=a[1];
     for i:=2 to 10 do                          }
       begin                                    }
         if  s<0  then  s:=0;                   }  累加,并比较大小
           s:= s+a[i];                          }  保存最大数据序列和
           if s>max then max:=s;                }
       end;                                     }
     writeln('max=', max)
  end.
```

初中:输入:-2　13　-1　4　7　8　-1　-18　24　6

　　输出:max=

高中:输入:8　9　-1　24　6　5　11　15　-28　9

　　输出:max=

【问题分析与答案】

本题正确答案是:

初中组答案:max=42,高中组答案:max=77

本题是一个简单的数列求和问题,通过循环输入了 10 个数据,再利用一个循环结构计算这一列数据的和并输出最大的数据序列和。问题的关键是:

（1）if　s<0　then　s:=0；这条语句作用是使 s<0 的数列和值屏蔽，令其为 0；

（2）if　s>max　then　max:=s；此语句作用保存最大数据序列的和。

1998 年　初中组写程序运行结果题　第 2 题

```
program　exp2 (input, output);
const　n=5;
var　i,j,k: integer;
    a:array[1..2*n, 1..2*n] of　integer;
begin
  k:=1;
  for i:=1 to 2*n-1 do
    if i<=n　then
        if　odd(i)　then
              for j:=i　downto　1　do
                begin
                  a[i-j+1,j]:=k;　k:=k+1
                end
        else　for j:=1　to　i　do
            begin
              a[i-j+1, j]:=k;　k:=k+1;
            end
  else　if odd(i)　then
              for　j:=n　downto　i-n+1 do
                begin
                  a[i-j+1, j]:=k;　k:=k+1;
                end
        else　for j:=i-n+1 to n　do
              begin
                a[i-j+1,j]:=k;　k:=k+1;
              end;
    for　i:=1　to　n　do
      begin
        for　j:=1 to n do
          write(a[i,j]:3);
          writeln
      end;
end.
```

i<=n，奇数行、偶数行分别赋值

i>n，奇数行、偶数行分别赋值

输出蛇形矩阵

【问题分析与答案】

本题正确答案是：

1	3	4	10	11
2	5	9	12	19
6	8	13	18	20
7	14	17	21	24
15	16	22	23	25

本题程序从表面看很复杂,其实仔细分析可以发现程序的结构很简单。它是一个循环结构程序,循环体内执行一个条件语句:

```
if  i < = n  then …  else  …
```

问题的关键在于条件语句内部又嵌套了条件语句,看起来比较复杂,考生往往感到问题比较难,因而出错的人比较多,下面我们仔细分析问题的解:

(1) 首先看程序的整体结构是由两个循环语句组成:

```
for i: = 1 to  2 * n - 1  do
    完成一个条件语句;
for  i: = 1  to  n  do
    完成一段打印程序 ;
```

(2) 阅读程序可以从前到后阅读,也可以从程序最后需要输出的结果,向前阅读程序。本题可以首先阅读程序的最后一段:

```
for  i: = 1  to  n  do
    begin
        for  j: = 1 to n do
            write(a[i, j]:3);
        writeln
end;
```

该程序段是输出一个 N 行、N 列的数据阵列,数据来自于数组 A[I, J]的值。

(3) 其次分析数据处理程序段:

① 循环语句: for i: = 1 to 2 * n - 1 do 内只执行一条条件语句:

if i < = n then … else … 该语句的作用是将数据处理分为两段:

i < = n 程序段和 i > n 程序段;

② 当 i < = n 时,执行的程序段功能:

```
if  odd(i)  then
        for j: = i  downto  1  do
          begin
            a[i - j + 1, j]: = k;      k: = k + 1
          end
  else  for j: = 1  to  i  do
      begin
        a[i - j + 1, j]: = k;   k: = k + 1;
      end
```

从程序段中可以分析知道：当奇数行时，数组单元赋值的对象是行下标逐渐增加，列下标逐渐减小的下标变量，其值是递增的自然数；否则若为偶数行则数组单元赋值的对象是行下标逐渐减小，列下标逐渐增大。

当 i=1 时，执行结果：a[1, 1]=1，　执行 k:=k+1　得 k=2；

当 i=2 时，执行结果：a[2, 1]=2，　执行 k:=k+1　得 k=3，

a[1, 2]=3，　执行 k:=k+1　得 k=4；

当 i=3 时，执行结果：a[1, 3]=4，　执行 k:=k+1　得 k=5，

a[2, 2]=5，　执行 k:=k+1　得 k=6，

a[3, 1]=6，　执行 k:=k+1　得 k=7；

当 i=4 时，执行结果：a[4, 1]=7，　执行 k:=k+1　得 k=8，

a[3, 2]=8，　执行 k:=k+1　得 k=9，

a[2, 3]=9，　执行 k:=k+1　得 k=10

a[1, 4]=10，　执行 k:=k+1　得 k=11；

……

至此，可以得到 i<=n (5) 数组下标单元的数据，并将这一部分进行排列得：

1　3　4　10　11

2　5　9　12

6　8　13

7　14

15

通过部分数据的输出排列，可以发现此数据的赋值和输出是按照一定的规律进行赋值和输出的：

随着行数增加，数据赋值个数在增加，其规律是决定行数 i；

数据的排列是按对角斜线逐渐递增排列自然数的；

③ 讨论当 i>n 时的情况：

```
if odd(i)  then  for  j:=n  downto  i-n+1 do
                 begin
                     a[i-j+1, j]:=k;      k:=k+1;
                 end
           else  for j:=i-n+1 to n  do
                 begin
                   a[i-j+1, j]:=k;    k:=k+1;
                 end;
```

同样按照奇数、偶数分别处理：

当 i=6 时，偶数，执行结果：a[5, 2]=16，执行 k:=k+1，得 k=17，a[4, 3]=17，执行 k:=k+1，得 k=18，a[3, 4]=18，a[2, 5]=19；

同理可以知道

a[3, 5]=20，a[4, 4]=21，a[5, 3]=22，a[5, 4]=23，a[4, 5]=24，a[5, 5]=25

所以，我们发现数字阵列的下三角矩阵也符合数据的排列是按对角斜线逐渐递增排列自

然数的,但随着行数增加,每行赋值数据在减少;至此完成整个问题的求解。

1998 年　初中组写程序运行结果题　第 3 题(高中组写运行结果题第 2 题)

```
program  exp3 (input, output);
const  n = 10;
var
  s, i : integer;
function  co(i1:integer) : integer;
    var j1,s1 : integer;
  begin
      s1 := n;
      for j1 := (n-1)  downto  (n-i1+1)  do
          s1 := s1 * j1  div  (n-j1+1);
      co := s1
  end;
```

被调用的函数

函数所计算的表达式是 C_m^n

```
begin
  s := n+1;
  for i := 2  to  n  do  s := s+co(i);
    writeln('s = ',s);
end.
```

主程序,循环 n-1 次,累加 co(I)函数的值。

【问题分析与答案】

本题正确答案是:

11 + 10 * 9/2 + 10 * 9 * 8/2/3 + 10 * 9 * 8 * 7/2/3/4 + … + 10 + 1 = 772。

(1) 阅读程序中,如果有过程或函数,一般先阅读主程序,了解主程序中是如何调用过程和函数的。本程序中的主程序是通过一个循环结构(n-1 次)累加函数的值;

(2) 再阅读过程或函数,了解过程或函数的功能。本题是调用一个函数,其函数的功能是通过一个循环结构完成累乘和累除的计算;

(3) 通过列表了解变量变化的趋势,以便很快掌握函数所要计算的表达式,快速得到程序的运行结果:

s 初始化为 s = 11;

当 i = 2,计算 s := s + co(2),co(2)的值:10 * 9/2,所以 s = 11 + 10 * 9/2

当 i = 3,计算 s := s + co(3),co(3)的值:10 * 9 * 8/2/3,所以

s = 11 + 10 * 9/2 + 10 * 9 * 8/2/3 = 11 + 10 * 9/(2!) + 10 * 9 * 8/(3!)

……

(4) 由以上计算,可以发现它的一般规律,该程序计算的表达式是考生非常熟悉的组合计算公式:

$$C_m^n = m!/(n! * (m-n)!) = m * (m-1) * \cdots * (m-n+1)/n!$$

当 m = 10, n = 0　得　$C_{10}^0 = 1$,当 m = 10, n = 1 得 $C_{10}^1 = 10$,

当 m = 10, n = 2　得　$C_{10}^2 = 10 * 9/2! = 45$,同理,可以得到其他项的值。利用组合公式

计算的对称性，可以得到 C_{10}^{6}、C_{10}^{7}、C_{10}^{8}、C_{10}^{9}、C_{10}^{10}的值。所以本题计算表达式是：

$$S = C_{10}^{0} + C_{10}^{1} + C_{10}^{2} + C_{10}^{3} + C_{10}^{4} + C_{10}^{5} + C_{10}^{6} + C_{10}^{7} + C_{10}^{8} + C_{10}^{9} + C_{10}^{10} = 772$$

1998 年　初中组写程序运行结果题　第 4 题

```
program  exp4(input, output);
  const n=3;
var   i,j,s,x:integer;
      p:      array[0..n+1] of integer;
      g:      array[0..100] of integer;
begin
      for  i:=0 to 100 do  g[i]:=0;
      p[0]:=0; p[n+1]:=100;
      for i:=1 to n do read(p[i]);  readln;

      for i:=0 to n do
        for j:=i+1 to n+1 do
          g[abs(p[j]-p[i])]:=g[abs(p[j]-p[i])]+1;

    s:=0;
    for i:=0  to  100  do
      if g[i]>0  then  begin
                         write(i,:4);  s:=s+1;
                       end;

    writeln;
writeln('s=',s);
writeln('input data:');  readln(x);
writeln(g[x])

end.
```

初始化程序段
p[i]赋初值

通过循环结构
给数组 g 赋值。

输出 g 数组中不为零的
下标变量的值。

输出 g 数组中第 x
单元下标变量的值。

输入：10　20　65
　input　data:　10
输出：

【问题分析与答案】

本题正确答案是 10　20　35　45　55　65　80　90　100
　　s=9
　　input　data:　10
　　10　2

（1）本题主要考核循环结构与数组下标的有机结合，给数组赋值的操作。

（2）程序分为四段：初始化程序段、数组赋值程序段、输出 G 变量数组的值、输出某个下标单元的值；

(3) 本程序的难点在第二程序段:

```
for i:= 0 to n do
    for j:= i+1 to n+1 do
        g[abs(p[j]-p[i])]:=g[abs(p[j]-p[i])]+1;
```

我们可以通过列表,了解变量变化的情况,掌握程序的功能。

p[0]=0,	p[1]=10,	p[2]=20,	p[3]=65, p[4]=100
当 i=0:	j=1,	p[j]-p[i]=10,	g[10]=1;
	j=2,	p[j]-p[i]=20,	g[20]=1;
	j=3,	p[j]-p[i]=65,	g[65]=1;
	j=4,	p[j]-p[i]=100,	g[100]=1 ;
当 i=1:	j=2,	p[j]-p[i]=10,	g[10]=2;
	j=3,	p[j]-p[i]=55,	g[55]=1;
	j=4,	p[j]-p[i]=90,	g[90]=1;
当 i=2:	j=3,	p[j]-p[i]=45,	g[45]=1;
	j=4,	p[j]-p[i]=80,	g[80]=1;
当 i=3:	j=4,	p[j]-p[i]=35,	g[35]=1;

这样 g 数组的值,一目了然。

(4) 输出 g 数组中,变量单元不为零的所有下标变量的值;

(5) 输出 g 数组中第 10 下标变量的值,完成全部程序的值。

1998 年　高中组写程序运行结果题　第 3 题

```
program  exp3(input,output);
  var i,j,s:integer;
      b:array[0..5]  of  integer;
  begin
      s:=1;
      for i:=1  to  5  do  b[i]:=i   { 给 b 数组赋值:1,2,3,4,5};
      j:=1;
      while  j>0  do
        begin
          j:=5;
          while  (j>0) and (b[j]=10+j-5) do j:=j-1;
          if j>0 then
            begin
              s:=s+1;  b[j]:=b[j]+1;
              for i:=j+1 to 5 do  b[i]:=b[j]+i-j
            end;
```

```
        end;
        writeln('s = ',s);
end.
```

【问题分析与答案】

本题正确答案是 s = 252 。

（1）本题主要目的输出变量 s 的值；

（2）程序的主要结构如下：

```
j: =5;
while  (j>0) and (b[j] =10 +j -5) do j: =j -1;
if j>0 then
  begin
      s: =s +1;  b[j]: =b[j] +1;
      for i: =j +1 to 5 do  b[i]: =b[j] +i -j
  end;
```

（3）通过列表，找出 s 变量、b 数组值的变化规律：

① j = 5：b[5] < >10，　s = 2，　b[5] = 6　b[5] < >10，　s = 3，　b[5] = 7

　　…　　b[5] < >10，　s = 6，　b[5] = 10

② j = 4：b[4] < >9，　s = 7，　b[4] = 5　b[5] = 5 + 5 - 4 = 6

　　b[5] < >10，　s = 9，　b[5] = 7　…

通过列表可以发现，j 的值变化一次，b[5]的值又被重新赋值，其变化规律是：

第一次 b[5]变化 5 次最后 b[5] = 10 ；

第二次 b[4]变化引起 b[5]变化，其变化次数为 4 次；

第三次 当 b[3]变化后导致 b[4]变化，同样也引起 b[5]值的变化，而且是循环嵌套的变化，其变化规律如下表：

变量		b[1]	b[2]	b[3]	b[4]	b[5]
	s = 1	1	2	3	4	5
j = 5：	s = 2	1	2	3	4	6
	s = 3	1	2	3	4	7
	s = 4	1	2	3	4	8
	s = 5	1	2	3	4	9
	s = 6	1	2	3	4	10
j = 4：	s = 7	1	2	3	5	6
	s = 8	1	2	3	5	7
	s = 9	1	2	3	5	8
	s = 10	1	2	3	5	9
	s = 11	1	2	3	5	10
	s = 12	1	2	3	6	10
	s = 13	1	2	3	6	8

	s =	14	1	2	3	6	9
	s =	15	1	2	3	6	10
	s =	16	1	2	3	7	10
	s =	17	1	2	3	7	9
	s =	18	1	2	3	7	10
	s =	19	1	2	3	8	10
	s =	20	1	2	3	8	10
	s =	21	1	2	3	9	10
j = 3:	s =	22	1	2	4	9	6
	s =	23	1	2	4	5	7
	s =	24	1	2	4	5	8
	s =	25	1	2	4	5	9
	s =	26	1	2	4	5	10
	s =	27	1	2	4	6	10
	s =	28	1	2	4	6	8
	s =	29	1	2	4	6	9
	s =	30	1	2	4	6	10
	s =	31	1	2	4	7	10
	s =	32	1	2	4	7	9
	s =	33	1	2	4	7	10
		……					
j = 1:	s =	246	4	7	8	9	10
	s =	247	5	7	8	9	10
	s =	248	5	6	7	8	10
	s =	249	5	6	7	9	10
	s =	250	5	6	8	9	10
	s =	251	5	7	8	9	10
	s =	252	6	7	8	9	10
	s =	252					

同理,b[4]、b[3]、b[2]、b[1]也有类似的变化,而 s 变量是计数器,记录变化次数。

变化计数　s = 1 + 5 + (5 + 4 + 3 + 2 + 1) + (15 + 10 + 5 + 4 + 1) + (35 + 20 + 10 + 4 + 1) + (70 + 35 + 15 + 6) = 252

1998 年高中组写程序运行结果题第 4 题

```
program  exp4 (input,output);
  const n = 4;
  type  se = array[1..n * 2] of char;
  var   i, j, i1, j1, k, s, t, s1, l, swap:integer;
        temp:char;
```

```
        a:se;
    begin
    for  i:=1  to  n*2  do  read(a[i]);  readln;
    s:=0;  t:=0;
    for i:=1 to n*2 do
      if a[i]='1' then s:=s+1 else if a[i]='0' then t:=t+1;
    if  (s<>n)  or  (t<>n)  then  writeln('error')
      else  begin
              s1:=0;
统计相邻
数不同的  for i:=1 to 2*n-1  do  if a[i]<>a[I+1] then s1:=s1+1;
数的个数    writeln('jamp=', s1);  swap:=0;
          for i:=1 to 2*n-1 do
            for j:=i+1 to 2*n do
              if a[i]<>a[j] then
              begin
               temp:=a[i];a[i]:=a[j] ;a[j]:=temp;
               s:=0;
                for l:=1 to 2*n-1 do
                  if a[l]<>a[l+1]  then  s:=s+1;
               if s>swap  then
                  begin  swap:=s;  i1:=i;  j1:=j  end;
                  temp:=a[i]; a[i]:=a[j]; a[j]:=temp;
              end;
      if  swap>0  then  writeln('maxswap=',swap-s1,'i=', i1,'j=',j1)
      end
  end.
```

赋值及初始化阶段（第一段）

输入：10101100

【问题分析与答案】

本题正确答案是：jamp=5　maxswap=2　i=6　j=7

（1）本题比较复杂，求解的最终目的，输出几个变量的值，为了明确本程序的功能，我们将程序分为几段阅读理解；

（2）第一段程序初始化阶段：给数组 a[i]赋值 0 和 1 字符，并分别统计 0、1 字符的个数；

（3）统计 0、1 字符数的个数是否分别为 N，如果不是则输出“出错”，否则做任务(4)；

```
if  (s<>n)  or  (t<>n)  then  writeln('error')
                else  begin
```

（4）统计所输入的字符中相邻数不相同的数的个数：

```
for i:=1 to 2*n-1  do  if a[i]<>a[i+1] then s1:=s1+1;
writeln('jamp=', s1);  swap:=0;
```

根据题意,本题输入数据是:10101100,所以相邻数不相同数的个数 s1 =5;其程序段是:

```
for i: =1 to 2 * n -1   do   if a[i] < >a[i +1] then s1: =s1 +1;
        writeln('jamp =', s1);   swap: =0;
```

(5) 双重循环内完成的条件语句:

这段程序比较复杂,主要任务是将 2n 个数进行比较和交换,每当比较与交换后,数据再次进行相邻数之间是否相同进行统计,保存最多的相邻数不同的数据个数,以及相应位置,如:

i =1,j =2 ~2 * n, a[1] =0,a[2] =1,所以 a[1] < >a[2],交换 a[1]、a[2],结果是 01101100,再次进行相邻两个数的比较得到 s =4, swap =4, i1 =1, j1 =2,再将交换的数还原,又得:10101100;再将 a[i]与 a[3]进行比较,同样用相邻两个数进行比较与交换,保存最多的相邻数不同的数据个数;

(6) 输出相应变量的值:jamp =5   maxswap =2   i =6   j =7

1999 年　初中组写程序运行结果题第 1 题

```
program   excp1;
  var
      x, y, y1, jk, j1, g, e:integer;
      a:array[1..20]of 0..9;
  begin
    x: =3465; y: =264; jk: =20;
  for j1: = 1 to   20   do   a[j1]: = 0;
    while y <  >0 do
    begin
        y1: =y mod 10;
        y: = y div 10;
      while   y1 < >0 do
      begin
          g: =x;
          for   e: = jk   downto   1   do
          begin
              g: =g +a[e];
              a[e ]: =g mod 10;
              g: = g div 10
          end;
          y1: = y1 -1
      end;
      jk: =jk -1
  end;
  j1 =1;
  while   a[j1] =0 do          j1: =j1 +1;
```

```
    for jk:=j1 to 20 do          write(a[jk]:4)
    writeln
  end.
```

程序输出结果为：_____

【问题分析与答案】

本题正确答案是：9　1　4　7　6　0

（1）本题主要考察学生对多重循环结构程序的执行过程的理解以及除法运算中的商、余数问题的基本方法；

（2）程序主要分为：初始化程序段、多重循环嵌套处理数据程序段、输出程序段；

（3）该程序用一个三位数的位数控制外循环次数，用其余数控制内循环次数，其任务完成数组的赋值和计算；

（4）通过列表了解变量变化情况：

x=3465　y=264　jk=20

① y1=4，y=26，g=3465，做 for 循环后：a[20]=5，　a[19]=6，　a[18]=4，a[17]=3，其他为0；

② y1=3，y=26，g=3465，做 for 循环后：a[20]=0，　a[19]=3，　a[18]=9，a[17]=6，其他为0；

③ 直到 y1=0 退出 while y1<>0　再完成 jk:=jk-1；继续执行 while　y<>0 的外循环。

（5）读者在完成这类任务时需要细心、耐心考察变量值的变化。

本题程序结构比较好，主要考察循环结构的知识，让学生掌握循环运行的过程，其程序本身没有多少值得研究的算法。

1999年　初中组写程序运行结果题第2题

```
program  excp2
  var
    i,j:integer;
    a:array[1..14]  of  integer;

  procdure sw(i1,j1:integer);
    var k1: integer;
    begin
    for  k1:= 1  to(j1 - i1 +1) div 2  do
      begin
        a[i1+k1-1]:=a[i1+ k1-1]+a[j1-k1+1];
        a[j1-k1+1]:=a[i1+k1-1]-a[j1-k1+1];
        a[i1+k1-1]:=a[i1-k1+1]-a[j1-k1+1];
      end;
    end;
```

```
begin
  j:=211;
  for i:=1 to 14 do
    begin
      a[i]:=i;    j:=j-i
    end;
  sw(1,4);sw(5,10);
  sw(11,14);sw(1,14);

  for i:=1   to 14   do
    begin
      if   j   mod   i=1   then write (a[i]:3);
      j:=j-a[i];
    end;
    writeln
end.
```

程序运行结果是:_____

【问题分析与答案】

本题正确答案是 12　5　10

(1) 本题的程序结构非常清晰,易于阅读,读者只要把握过程的功能、主程序如何与过程衔接,就可以很快知道问题的解;

(2) 过程作用的主要程序段是利用数学的和、差计算,交换两个下标变量的内容,交换单元是首尾两个存储单元的值进行交换。

```
for   k1:= 1   to(j1 - i1 +1) div 2   do
  begin
    a[i1+k1-1]:=a[i1+ k1-1]+a[j1-k1+1];
    a[j1-k1+1]:=a[i1+k1-1]-a[j1-k1+1];
    a[i1+k1-1]:=a[i1-k1+1]-a[j1-k1+1];
  end;
```

(3) 主程序的作用是给 a 数组赋值,分段调用过程进行数据交换,其实质进行了四次逆排序:

① 逆排序结果:　4　3　2　1

② 逆排序结果:10　9　8　7　6　5

③ 逆排序结果:14　13　12　11

④ 总逆排序结果:11　12　13　14　5　6　7　8　9　10　1　2　3　4

(4) 在程序中 j 初值为 211,经过 j:=j-i　j=106。

(5) 将 j 除以 i 余数为 1 的下标变量输出。

再次从 j 中减去 a 数组的值。所以输出结果为:a[2]、a[5]、a[10]。

即:12　5　10。

1999 年　高中组写程序运行结果题　第 2 题

设数组 a[1], a[2], …, a[n],已存入了数据,调用不同的排序程序,则数据比较的次数将会不同,试计算分别调用下列不同的排序过程的比较运算的次数。其中 swap(i, j)表示 a[i]与 a[j]进行交换。

```
(1) procedure  sort1(n:integer);
      var i, j:integer;
      begin
       for  i:=1  to  n-1  do
         for  j:=1  to  n  do
           if a[j] < a[i] then  swap (i, j)
      end;
```

调用该过程的语句为 sort1(n),比较运算的次数为:_____

```
(2) procedure sort2(i, n:integer);
      var j :integer;
      begin
       if  i= n  then  write(a[n])
                 else
                   for  j:=i+1  to  n  do
       if  a[j] < a[i]  then  swap (i, j);
                   write(a[i]);
                   sort2(i+1, n);
       end
```

调用该过程的语句为 sort2(0, n),比较运算的次数为:_____

```
(3) procedure  sort3(i, j:integer);
      var  m: integer;
      begin
        if  i〈〉j  then
          begin
            m:=(i+j) div  2;
            sort3(i, m);sort3(m+1, j);
            merge;{假设合并的元素分别为 p、q 个,需要比较 p+q 次}
          end;
      end;
```

调用该过程的语句为 sort3(0, n),比较运算的次数为:_____

【问题分析与答案】

本题是一个考察学生对算法时间复杂性的理解问题。算法的时间复杂性决定于循环次数和循环嵌套的层数。循环次数越多、循环嵌套层数越多则所花费的时间越长,一般用函数式 $O(n^a)$ 表示。

(1) 程序 1 是通过双重循环结构,完成数据的大小比较和数据的交换,所以调用该过程的语句为 sort1(n),比较运算的次数为:n*(n-1);

(2) 程序 2 是一个通过递归调用完成数据的大小比较的,当用 sort2(0, n)进行递归调用其重复大小比较运算的次数为:

$$n+(n-1)+(n-2)+\cdots+1=n*(n+1)/2$$

(3) 程序 3 是采用二分归并的方法进行数据大小比较和排列的,当调用该过程的语句为 sort3(0, n),比较运算的次数为:

$$2*n/2+2*n/4+2*n/8+2*n/16\cdots=2n*(1/2+1/4+1/8+\cdots)\approx 2n$$

所以三个程序中,第 3 个程序速度最快。

2000 年　初中组写程序运行结果　第 1 题(高中组写程序运行结果第 2 题)

```
program  noi__002;
  var  i, j, l, n, k, s, t:integer;
       b:  array[1..10] of 0..9;
  begin
    readln(l, n); s:=l;k:=1;t:=l;
    while  s<n  do
      begin  k:=k+1; t:=t*l;s:=s+t  end;
    s:=s-t;  n:=n-s-1;
    for  i:=1  to  10  do  b[i]:=0;
    j:=11;
    while  n>0  do
      begin
        j:=j-1;  b[j]:=n  mod  l;  n:=n div l;
      end;
    for i:=10-k+1  to  10  do
        write(chr(ord('a')+b[i]));
  end.
```

输入:4　167

输出:

【问题分析与答案】

本题正确答案是 bbac。

(1) 首先是初始化程序

```
readln(l, n);  s:=l;  k:=1;  t:=l;
```

根据题意:l=4,　n=167,　s=4,　k=1,　t=4。

(2) 程序　while　s<n　do

```
begin  k:=k+1;  t:=t*l;  s:=s+t  end;
```

计数,累乘和累加。这里所要计算的表达式是:

$s=L+L^2+L^3+\cdots\cdots+L^k=4+4^2+4^3+\cdots\cdots+4^k$直到 s>=N 即 s=340>N 为止；

(3) 计算　s:=s-t; n:=n-s-1,得 s=340-256=84, n=167-84-1=82;

所以得此时的变量的值是：　l=4, n=82, s=84, k=4, t=256;

(4) 数组 b 初始化,全部清零;

(5) 将 n 数据转换为 l 进制,即$(82)_{10}=(1102)_4$并将按照从下标大的元素开始存放余数值;

```
j:=11;
    while  n>0  do
        begin  j:=j-1;  b[j]:=n mod l;  n:=n div l end;
```

数组 b 中,b[10]=2, b[9]=0, b[8]=1, b[7]=1,所以对应二进制是 1102。

(6) 将数组 b 中, b[7], b[8], b[9], b[10] 的值分别和字符'a'的序数值相加,再转换成字符输出。

```
for i:=10-k+1  to  10  do
    write(chr(ord('a')+b[i]));
```

所以输出结果:bbac 。

2000 年　初中组写程序运行结果　第 2 题

```
program  noi__004;
  var  i, j, j1, j2, p, q:integer;  p1:boolean;
                b, c: array[1..100] of  integer;
  begin
    readln(q, p);  j:=1;  p1:=true;  b[j]:=q;  j1:=0;
    while  (q>0)and  p1  do
      begin
        j1:=j1+1;  c[j1]:=q*10  div  p;  q:=q*10-c[j1]*p;
        if  q>0  then
        begin
          j2:=1;
          while (b[j2]<>q)and (j2<=j)  do  j2:=j2+1;
          if  b[j2]=q  then
                        begin
                          p1:=false; write('0.');
                          for  i:=1  to  j2-1  do  write (c[i]:1);
                          write('(');
                          for  i:=j2  to  j1  do  write (c[i]:1);
                          writeln(')');
                        end
                        else    begin  j:=j+1; b[j]:=q      end;
        end;
    end;
```

余数不为零时,重复做将新的被除数除以除数,将商保存。

```
if  q=0  then
          begin
            write('0.');
             for  i:=1 to  j1   do      write(c[i]:1);
             writeln
          end;
   readln
end.
```

将Q/P能够除尽的小数输出处理

输入 ① 1 8 输出

输入 ② 2 7 输出

【问题分析与答案】

本题的正确答案是:1/8=0.125, 2/7 =0.{285714}

(1) 在读程序时,读者应该把握住程序的重点语句,也就是能表达出程序的主要目的的主要的语句,在该程序中,下面的语句是本程序的重点:

```
j1:=j1+1;  c[j1]:=q*10  div  p;  q:=q*10-c[j1]*p;
```

该段程序是一个典型的逐位求商的语句,是q被p除,商放在c[j1]中。根据问题中给的数据可以知道,本题得到的结果是一个纯小数的数据。

对于纯小数的处理有两种情况:

① q被p除能除尽;

② q被p除不能除尽。

(2) while q>0 and p1 do 重复做,将每次产生的余数q值存在数组b中,判断下标变量b[j2]是否与q相等,若相等则是无限循环小数,按循环小数处理;否则将余数扩大10倍,再进行除法运算。

```
while  q>0  and  p1  do
begin
if  q>0  then  begin
                 j2:=1;
                 while (b[j2]<>q)and (j2<=j)  do  j2:=j2+1;
          if  b[j2]=q  then
              begin
                p1:=false; write('0.');
                for  i:=1  to  j2-1  do  write (c[i]:1);
                write('{');
                 for  i:=j2  to  j1  do  write (c[i]:1);
                 writeln('}');
              end
否则继续做除法        else    begin  j:=j+1; b[j]:=q  end
              end;
```

循环小数处理过程

（3）当 q=0，表示 q/p 能够除尽，即余数为零则输出小数点后所有数

```
if  q=0  then
          begin
            write('0.');
            for  i:=1 to  j1   do      write(c[i]:1);
            writeln
          end;
```

根据问题输入 1 、8 和 2 、7 即将 1/8 和 2/7 分别化为小数，所以答案是：

① 0.125　　② 0.{285714}

2000 年　高中组写程序运行结果　第 1 题

```
program  noi ____003;
  const  n=7;  m=6;
  var  i, j, x0, y0, x1, y1, x2, y2:integer;
       d:real;  p:boolean;  g:array[0..n, 0..m]  of  0..1;
       function  disp(x1, y1, x2, y2:integer):real;
       begin
          disp:=sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
       end;
{两点间距离计算公式}
       begin
         for i:=0  to  n  do
           for  j:=0  to  m  do
             g[i,j]:=0;
{N 行 M 列矩阵初始值为零}
       readln(x1, y1, x2, y2);
       g[x1, y1]:=1;
       g[x2, y2]:=1;
{读入起点位置和终点位置}
        p:=true;
        while     p  do
          begin
            p:=false;
```

```
d:=disp(x1, y1, x2, y2);
  x0:=x1;
  y0:=y1;
  for i:=4  to  n  do
    for  j:=0  to  m  do
    if  (d>disp(i, j, x2, y2))and(g[i, j]=0 )then
        begin
            d:=disp(i, j, x2, y2);
              x0:=i;  y0:=j
        end;
if (x0<>x1) or (y0<>y1) then
     begin
       x1:=x0; y1:=y0; p:=true; g[x1, y1]:=1;
     end;
```

{本段程序是在矩阵中的某个范围找寻一个点,使得它与点(x2, y2)距离尽可能的大且该点没有被访问过。}

```
  d:=disp(x1, y1, x2, y2);      x0:=x2;  y0:=y2;
for  i:=0  to  3  do
    for  j:=0  to  m  do
     if  (d<disp(x1, y1, i, j))and(g[i, j]=0)  then
         begin
           d:=disp(x1, y1, i, j);
             x0:=i;  y0:=j
         end;
    if (x0<>x2)  or (y0<>y2)  then
      begin  x2:=x0; y2:=y0; p:=true; g[x2, y2]:=1; end;
end;
```

{本段程序是在矩阵中的某个范围找寻一个点,使得它与点(x1, y1)距离尽可能的小且该点没有被访问过。}

```
writeln(x1, y1, x2, y2)
end.
```

输入: 7 6 0 0　　输出:

【问题分析与答案】

本题正确答案是:当输入7 6 0 0 输出是4 3 0 2。

(1) 本程序阅读需要分块进行,否则就无法理解程序的功能。读者可以发现程序分为五部分:距离计算、初始化、搜索点并找最小距离、搜索点找最大距离、输出;

(2) 初始化矩阵:数组 g 的所有元素清零,即循环加 g[i, j]=0;

(3) 读入起始位置:(x1, y1)、(x2, y2)并令 g[x1, y1]=1 和 g[x2, y2]=1,由此可以知道矩阵中的两个点已被访问过;

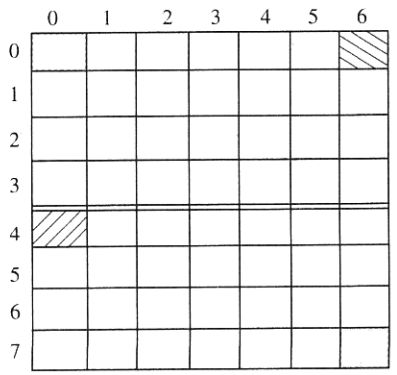
（4）搜索未被访问的点并找距离(x2, y2)最小点位置,点的搜索范围从第四行开始的矩阵下半段:

```
for i: = 4  to  n  do
  for  j: = 0  to  m  do
  if  (d > disp(i, j, x2, y2))and(g[i, j] =0 ) then
        begin
            d: = disp(i, j, x2, y2);
            x0: = i;  y0: = j
         end;
 if (x0 < > x1) or (y0 < > y1) then
        begin
          x1: = x0;  y1: = y0;  p: = true;  g[x1, y1]: = 1;
        end;
```

（5）搜索未被访问的点并找距离(x1, y1)最大点位置,点的搜索范围从第零行开始的矩阵上半段:

```
d: = disp(x1, y1, x2, y2);      x0: = x2;  y0: = y2;
for  i: =0  to  3  do
  for  j: =0  to  m  do
  if  (d < disp(x1, y1, i, j))and(g[i, j] =0)  then
      begin
        d: = disp(x1, y1, i, j);
          x0: = i;  y0: = j
      end;
  if  (x0 < > x2)  or (y0 < > y2)  then
      begin  x2: = x0;  y2: = y0;  p: = true;  g[x2, y2]: = 1; end;
  end;
```

由搜索可以得到如下图所示的矩阵图(各自移动了一个点):



由上面的分析,我们不难发现有两个点在移动,一个点在矩阵的下半段,一个在矩阵的上半段,一个希望距离最小,一个则希望距离最大,由此使我们可以联想到这是一个互相追逐的游戏或是一个模拟“猫捉老鼠”问题的求解。

该程序猫从点(7, 6)开始,老鼠从点(0, 0)开始,移动过程中:
猫的移动: (7, 6),(4, 0),(4, 6),(4, 1),(4, 5), (4, 2),…,(4, 3)
老鼠的移动:(0, 0),(0, 6),(1, 0),(1, 6),(0, 1), (0, 5),…,(0, 2)

所以该程序的最后运行结果为:4 3 0 2。

第三章　完善程序

第一节　1995年试题

一、初中组第1题

【问题描述】

求出所有满足下列条件的二位数：将此二位数的个位数字与十位数字进行交换，可得到一个新的数，要求新数与原数之和小于100。

程序要求：每行输出6个满足要求的数。

【算法说明】

分解每一个二位数，然后重新组成一个新数，当满足条件时，用计数器来统计个数。

【程序清单】

```
begin
  k:=0;
  for i:= ____①____ to 99 do
  begin
  x:= ____②____; y:= ____③____;
  j:=x*10+y;
  if ____④____ then
    begin
          k:=k+1;
          write(I:4);
          ____⑤____ then writeln;
    end
  end
end.
```

【问题分析】

为求出所有满足条件的二位数，程序采用枚举所有的二位数i。对每一个二位数i，分拆为个位数字x和十位数字y，交换后得到的新数j=x*10+y因此①处填10，②处为i mod 10，求得个位数字x，③处填i div 10，求得十位数字y，得到的这一对新数j和原数i，应满足i+j<100，这是填空④的内容。填空⑤处是为每行输出6个二位数设置的。程序中统计满足条件二位数的计数器显然是k，边统计，边输出，每行输出6个即为k逢6要换行。所以5处应为if k mod 6=0。

二、初中组第2题

【问题描述】

找出小于33的6个正整数,用这些整数进行加法运算,使得包括原来的整数在内能组成尽可能多的不同整数。

例如:用2,3,5这3个数能组成下面的数

2,3,5

2+3=5,　　5已经存在

2+5=7,　　3+5=8,　　2+3+5=10

所以用2,3,5能组成6个不同的数。

程序要求:输出所选的这6个数,以及能组成不同整数的个数。

【算法说明】

选择的这6个数,用来组成数时应该尽可能不重复,引入数组a保存找出的这6个数。

【程序清单】

```
begin
  a[1]:=1;t:=0;
  for i:=2 to 6 do
    begin
    ____①____;
    for j:=1 to i-1 do
      s:=____②____;
    a[i]:=____③____;
   end;
  for i:=1 to 6 do
  begin
    t:=____④____;
    write(a[I],' ')
  end;
  writeln('能组成不同整数的个数:',t)
end.
```

【问题分析】

掌握程序中变量s的意义是求解本题的关键。根据算法说明中提到的"选择的这6个数,用来组成数时应该尽可能不重复"这一策略,同时,这6个数要小于33的限制条件,我们不妨假设前i-1个数a_1, a_2, …, a_{i-1}已找到,对第i个数a_i,确定其值的原则为:(1)a_i要尽可能小且满足$a_1<a_2<\cdots<a_{i-1}<a_i$;(2)由$a_i$参与加法的结果应大于$a_1+a_2+\cdots+a_{i-1}$,这样保证与前i-1个数所组成的整数不重复。显然$a_i=a_1+a_2+\cdots+a_{i-1}+1$。因此,s的意义为前i-1个数之和,填空②为s+a[j],填空①置s为初值0即s:=0;填空③为s+1。

我们还可以换一种方法来思考:所选6个数$1=a_1<a_2<a_3<a_4<a_5<a_6<33$,这6个数在组成数时,每一个数都有参与加法或不参与加法两种可能,很自然想到,这6个数在组成数时,

可用一个6位二进制数表示，$b_6b_5b_4b_3b_2b_1$，其中$b_i=0$表示a_i不参与加法运算，$b_i=1$表示a_i参与加法运算。6位二进制数的每一种情况就表示组成的一个数，且各不相同。000001表示只有a[1]=1时的值，当有a[1]=1和a[2]时，能组成的数为000001，000010，000011这3个不同的数，所以a[2]取2，即二进制000010，……依次可得a[3]=4，即二进制000100，a[4]=8，a[5]=16，a[6]=32。

能组成不同整数的个数t=a[1]+a[2]+a[3]+a[4]+a[5]+a[6]，所以④处应填t+a[i]，累加得到。

三、初中组第3题

【问题描述】

求出2～1000之间长度最长的、成等差数列的素数(质数)。

例如：在2～50之间的全部素数有

2，3，5，7，11，13，17，19，23，29，31，37，41，43，47

其中公差为1的素数数列为2，3，其长度为2

公差为2的素数数列为3，5，7，其长度为3

……

程序要求：输出满足条件的素数数列。

【算法说明】

首先用筛选法求出此范围内的全部素数，存放在数组b中，然后用2个变量i，j，逐步求出满足条件的素数数列。

【程序清单】

```
begin
  max:=0; num:=1000;
  for i:=2 to num do b[I]:=i;
  for i:=2 to ____①____ do
  if ____②____ then
  begin
    k:=i+i;
    while k <=num do
    begin
        b[k]:=0;
        k:=k+i
    end
  end;
  for i:=2 to num-1 do
  if ____③____ then
```

```
begin
  j:=1;
  d[j]:=b[i];
  for i1:= ____④____ do
    if b[i1]<>0 then
    begin
        delta:= ____⑤____;
        k:=delta;
        while(i+k<=num) and ____⑥____ do
    begin
        j:=j+1;
        d[j]:=i+k;
        k:=k+delta
    end;
    if  j>max then begin
                    max:=j;
                    c:=d          {数组 d 的值赋给数组 c}
                  end;
    j:=1
  end
end;
writeln('The max length is:', max);
write('The string is:');
for i:= 1 to max do write(c[i],'');
writeln
end.
```

【问题分析】

本题考察选手对筛选法求素数的方法和采用逐步扫描法在一个序列中寻找满足某一条件的子序列的算法的掌握程度。

筛选法求素数的算法思想是将 2～m(求素数的范围)存放于数组 b 中:b[i]:=i;然后从 i=2 开始(2 为第一个素数),将 b 中 b[i]后面值为 2 的倍数的元素划去,这些元素的值为 k,易知 k 等于 2, 4, 6, 8…,采用 B[k]:=0 实现 k 被划去。筛去 2 的倍数后,继续往后找第 1 个没有被划去的元素,留下该元素,划去该元素值的倍数……直到所有的合数都划去为止。例如求 2～10 之间的素数,数组 b 中元素值的变化情况:

初始状态	(2, 3, 4, 5, 6, 7, 8, 9, 10)
划去 2 的倍数	(2, 3, 0, 5, 0, 7, 0, 9, 0)
划去 3 的倍数	(2, 3, 0, 5, 0, 7, 0, 0, 0)
划去 5 的倍数	(2, 3, 0, 5, 0, 7, 0, 0, 0)

划去7的倍数　　(2, 3, 0, 5, 0, 7, 0, 0, 0)

最后b中所有的非零元素就是所求的素数。

由上述分析,程序的第一部分(筛选法求素数)for i:=2 to ___①___ do是枚举第i个没有被划去的数(素数),___①___处应填num,___②___处应填b[i]<>0。

程序的第二部分是利用两个变量i,j,逐步求出长度最长的成等差的素数序列。数组b中存放了所求范围的所有整数,循环变量i确定当前所求等差数列的第一个元素b[i],并把b[i]用d[1]保存。所求等差数列的公差delta可由b[i]与它后面每个元素的差求得。程序中变量max记录最长的等差数列的长度,数组c保存当前最长的等差数列。程序采用循环搜索法枚举所有的等差素数数列。它的基本算法如下:

1. 循环变量i求出数列的第一个元素,该元素b[i]应为素数,所以___③___处应填b[i]<>0。

2. 循环变量i1求出数列的第二个元素。i1的取值应为i+1到num,数列的公差delta=b[i1]-b[i]。

3. 在求得当前数列的第一个元素d[1]后,再枚举下面的元素,利用公差delta作为增量,逐项求得:

```
k:=delta;
while (i+k<=num)and(b[i+k]<>0)do
begin j:=j+1;d[j]:=i+k;k:=k+delta end;
```

4. 将所求的等差数列长度j与max比较,记录当前最长的等差数列。

填空④应为:i+1 to num,⑤应为b[i1]-b[i]填空⑥应为(b[i+k]<>0)。

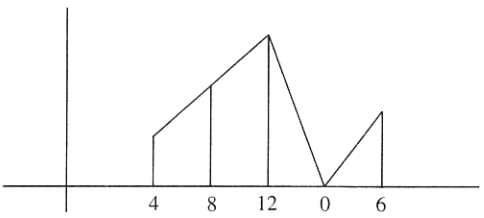
四、初中组第4题/高中组第1题

【问题描述】

求出二个整形数组错位相加的最大面积

1. 数组面积的定义:(限定数组头尾不为0)

设有一个数组C=(4, 8, 12, 0, 6)



则C的面积为Sc=(4+8)/2+(8+12)/2+12/2+6/2

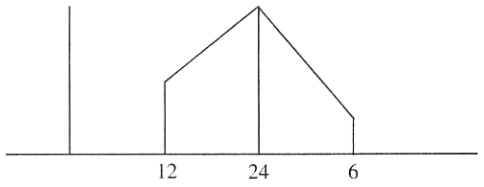
也就是说,Sc=各梯形面积之和(其中梯形的高约定为1,三角形作为梯形的特殊情况处理)。

又如D=(12,24,6)是,其面积的定义为

Sd=(12+24)/2+(24+6)/2

2. 数组错位相加的定义

设有2个正整数的数组a,b,长度为n,当n=5时:



a = (34, 26, 15, 44, 12)　　b = (23, 46, 4, 0, 18)

对 a、b 进行错位相加,可能有下列情况

	34	26	15	44	12					
+)						23	46	4	0	18
	34	26	15	44	12	23	46	4	0	18

或:

		34	26	15	44	12				
+)						23	46	4	0	18
		34	26	15	44	35	46	4	0	18

或:

			34	26	15	44	12			
+)						23	46	4	0	18
			34	26	15	67	58	4	0	18

或:……

最后有:

						34	26	15	44	12
+)	23	46	4	0	18					
	23	46	4	0	18	34	26	15	44	12

可以看到:由于错位不同,相加的结果也不同。

程序要求:找出一个错位相加的方案,使得输出的数组面积为最大。

【算法说明】

设 a, b 的长度为 10,用 a, b: array[1..10] of integer 表示,其结果用数组 c, d: array[1..30] of integer 表示。

错位相加的过程可以从开始不重叠,然后逐步重叠,再到最后的不重叠。

梯形面积的计算公式为:(上底 + 下底) * 高/2,其中约定高为 1,故可写为(上底 + 下底)/2。

【程序清单】

```
const n = 10;
……
function sea:real;{计算数组 C 面积}
begin
  j1: = 1;
```

```
    while ____①____ do j1:=j1+1;
    if  j1=3*n  then sea:=0
                else begin
                       j2:=3*n;
                       while ____②____ do j2:=j2-1;
                       if  j1=j2 then sea:=0
                       else begin
                              j3:=c[j1]+c[j2];
                              for  j4:=j1+1 to j2-1 do
                                   J3:=j3+c[j4]*2;
                              sea:=j3/2
                            end
                     end
end;
begin {主程序}
  for i:=1 to n do read(a[i]);
  for j:=1 to n do read(b[j]);
  ____③____;
  for i:=1 to 2*n+1 do
begin
  for j:=1 to 3*n do ____④____;
  for j:=1 to n do c[j+n]:=a[j];
  for j:=1 to n do ____⑤____;
  p:=sea;
  if p>s then begin
                d:=c;
                s:=p
                end;
  end;
  for i:=1 to 3*n do write(d[i],' ');
  writeln;
  writeln('s=',s)
end.
```

【问题分析】

1. 根据对数组面积定义，限定数组头尾不为0，且梯形的高约定为1，若数组 $C=(C_k, C_{k+1}, \cdots, C_{k+l-1})$，且 $C_k<>0$，$C_{k+l-1}<>0$，则面积：

$$S_c=(C_k+C_{k+1})/2+(C_{k+1}+C_{k+2})/2+\cdots+(C_{k+l-2}+C_{k+l-1})/2$$

$$=(C_k + 2C_{k+1} + 2C_{k+2} + \cdots + 2C_{k+l-2} + C_{k+l-1})/2$$

因此,经数组 a,b 错位相加后,寻找 c 中一段首尾元素均不为 0 的元素序列,然后根据上述面积公式求之。函数 sea 的定义中,先求出首尾两个元素的位置 j1 和 j2,当 j2 - j1 >0 时,用下面的这段程序求出数组 c 的面积:

```
j3:=c[j1]+c[j2];
for  j4:=j1+1 to j2-1 do inc(j3, c[j4]*2);
sea:=j3/2;
```

所以空格①处应为:(c[j1]=0)and(j1<3*n),空格②处应为:(c[j2]=0)and(j2>j1)。

3. 主程序中采用循环搜索数组 a,b 的所有错位情形,由题意可知,a,b 所有错位情况共有 2n+1 种,for i:=1 to 2*n+1 do 语句中,填空⑤的作用是将 b 数组按第 i 种错位方式加到数组 c 中,而语句 for j:=1 to n do c[j+n]:=a[j]说明数组 a 每次都固定在数组 c 的中间 n 个位置上,实现错位由 b 数组的头的位置来决定,用 i 作为 b[1]的位置,所以⑤处应为 c[i+j-1]:=c[i+j-1]+b[j]。程序③处和④处较容易填出:③应为 s:=0,④应为[j]:=0;

本题最终解答为:

①	(c[j1]=0)and(j1<3*n)
②	(c[j2]=0)and(j2>j1)
③	s:=0
④	c[j]:=0;
⑤	c[i+j-1]:=c[i+j-1]+b[j]

五、高中组第 2 题

【问题描述】

表的操作:设有一个表,记为 $L=(a_1, a_2, \cdots, a_n)$,其中:

L:表名。 $a_1, a_2, \cdots, a_n$ 为表中的元素。

当 a_i 为 0~9 数字时,表示元素,为大写字母时表示是另一个表,但不能循环定义。

例如下列表的定义是合法的(约定 L 是第一个表的表名)。

L=(1, 3, K, 8, 0, 4)

K=(3, P, 4, H, 7)

P=(2, 3)

H=(4, 0, 5, 3)

程序要求:当全部表给出之后,求出表中所有元素的最大元素,以及表中全部元素的和。

【算法说明】

表用记录类型定义: 设 lmax 为表中元素最大个数

```
tabtype = record
        length:0..lmax;                    {长度}
        element: array[1..lmax] of char;   {表体}
end;
```

再定义队列：　qtype = record
　　base:array[0..lmax] of char;
　　front,rear:0..lmax;
　　end;

为此，设计一个字符入队的过程 inqueue，出队函数 outqueue，表中最大元素及元素求和均采用递归计算。

【程序清单】

```
const lmax=38;
……
var t:array['A'..'Z']of tabtype;
    s:string[lmax];

procedure inqueue(var q:qtype; c : char);
begin
  q.rear:=    ①    ;
  q.base[q.rear]:=c
end;

function outqueue(var q:qtype):char;
begin
  q.front:=    ②    ;
  outqueue:=q.base[q.front]
end.

function maxnumber(c:char):char;
var max:char;
begin
  max:=chr(0);
  for i:=1 to t[c].length do
  begin
  ch:=t[c].element[i];
  if    ③    then m:=maxnumber(ch)
                else m:=ch
  if  max<m then max:=m
  end;
    ④
end;

function total(c:char):integer
```

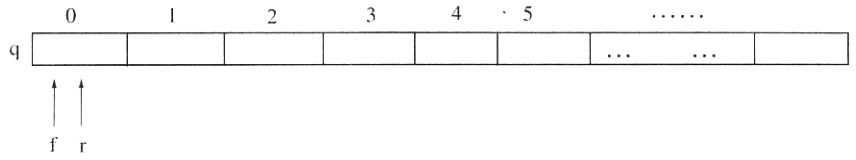
```
var k,i:integer;
begin
  k:=0;
  for i:=1 to t[c].length do
  begin
  ch:=t[c].lelment[i];
  if ____⑤____then m := total(ch);
                 else m:=ord(ch)-ord('0');
  k:=k+m
  end;
  total:=k
end;

begin
  max:=36;
 for tabno:='A' to 'Z' do t[tabno].length:=0;
 q.front:=0;q.rear:=0;
 inqueue(q,'L');
 while q.front<>.rear do
 begin
  tabno:=outqueue(q);
  write(tabno,'=');
  readln(s);
  i:=1;
  while s[i]<>'('do i:=i+1;
  while s[i]<>')' do
  begin
    if(s[i]>='a')and(s[i]<='z')
      then s[i]:=chr(ord(s[i])+ord('A')-ord('a'));
    if(s[i]>='A')and(s[i]<='Z')
    then begin
           inc(t[tabno].length);
        t[tabno].element[t[tabno].length] := s[i];
        inqueue(q, s[i])
          end
        else if(s[i]>='0')and(s[i]<='9')
           then begin
          inc(t[tabno].length);
```

```
    [tabno].element[t[tabno].length] := s[i]
            end;
  inc(i)
  end;
  end;
    writeln('the max number in table L is:', maxnumber('L'));
    writeln('Total is:', total('L'))
  end.
```

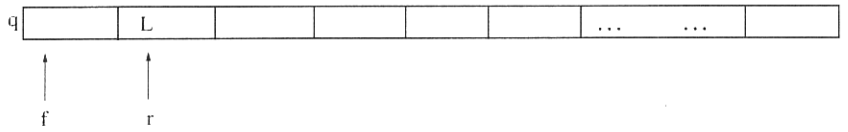
【问题分析】

广义线性表(简称广义表)是线性表的一种推广,如果允许构成线性表的元素本身又可以是一个线性表,那么这样的线性表是广义表。应用广义表可以描述树结构。本题考核选手对广义表遍历操作的掌握以及队列、递归等知识。题目采用递归定义广义表,约定表名为大写字母且L为第一个表的表名,用记录类型 tabtype 表示表,数组 element 表示表体。因此,仔细阅读主程序可发现,每一个表用一字符串 s 读入,且把 s 中所有的字母、数字看成为表的元素,其中小写字母化为大写字母。先读入表L,将其元素存入数组元素 t[L]中,对L中出现的字母,依次进队列 q;若 q 不空,队列头元素出队,读入以该元素为名的线性表,若再出现字母,再入队,……直到队列 q 空为止。下面图例根据题目所给的例子,说明广义表的输入过程:

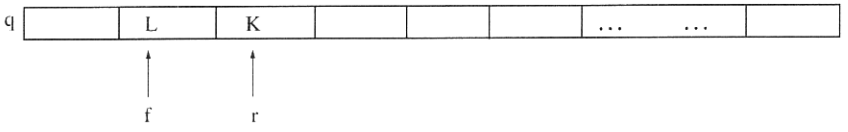
(1) 置队列 q 为空队列



(2) 'L'进队

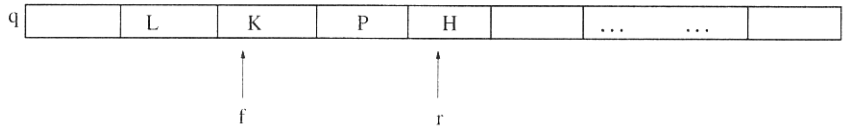


(3) 'L',读入 s='(1 3 K 8 0 4)', 'K'入队



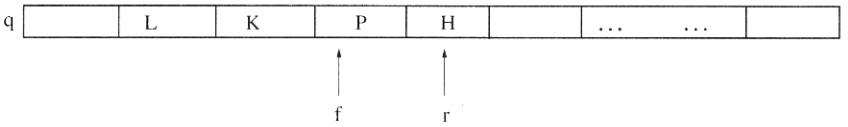
得表 t[L]='13K804'

(4) 'K'出队,读入 s='(3 P 4 H 7)', 'P', 'H'入队



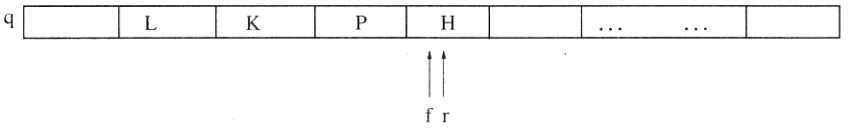
得表 t[K] = '3P4H7'

(5) 'P'出队,读入 s = '(2 3)'



得表 t[P] = '23'

(6) 置队列 q 为空队列'H'出队,读入 s = '(4053)'



得表 t[H] = '4053'

此时 f = r,队空结束输入。

由上述图示,入队过程 inqueue 定义中的①处为移动头尾指针,应为 q. rear + 1 或(q. rear + 1) mod (max + 1);出队过程 outqueue 定义中的②处为移动头指针,应为 q. front + 1 或(q. front + 1) mod (max + 1)。

函数 maxnumber 是求表中的最大元素,根据 t 数组存放的广义表,从 t[1]开始搜索。语句

if ____③____ then m: = maxnumber(ch)

else m: = ch;

为递归搜索,当前元素为大写字母时,即转入搜索该字母为表名的表。因此③处应填(ch < = 'A') and (ch > = 'Z')。

变量 max 记录搜索当前表所得元素的最大元素,并作为函数的值。所以④处应为 maxnumber: = max;这里特别强调,在函数定义中,函数名至少要被赋值一次,否则函数无法返回所求结果。

函数 total 求广义表个元素(均为数字)之和,亦采用递归计算,与函数 maxnumber 类同。

本题给出的广义表遍历算法,如果把广义表看作树的话,本题也是树的遍历算法。该树采用递归存储结构,因此在遍历中很方便地采用递归算法。而这种递归遍历算法,实际上实现了树的深度优先遍历。

本题最终答案为:

① q. rear + 1 或(q. rear + 1) mod (max + 1)

② q. front + 1 或(q. front + 1) mod (max + 1) 采用 mod 取值时把队列当作循环队列

③⑤ (ch > = 'A') and (ch < = 'Z')

④ maxnumber: = max

六、高中组第3题

【问题描述】

设有一个实数，以字符串形式存放于数组x中，用x::array[1..n]of char表示。其中x[1]若为'-'，表示负数；若为'+','.',' '，则表示正数。若为数字，也认为是正数。

例如　　x=(' ','2','0',' ','3','.','5','%')　　则表示203.5

　　x=('-','1','.',' ','2','0','%')　　则表示-1.2

约定：在字符串x中，除x[1]外，其后可以包含有若干个'.'与' '，但仅以第一次出现的为准，空格不起任何作用，并以字符'%'作为结束标志。

程序要求：将输入的字符串还原成实数(小数点后无用的0应除去)，还原的结果以下列形式存放(不需要输出)

f：数符。正数放0，负数放1。

a：array[1..n] of integer；存放数字，不放小数点。

k：表示A中有效数字的个数。

j：表示小数点后的位数。

例如：数203.24，还原后结果的存放是：

f=0

a=(2, 0, 3, 2, 4)

k=5

j=2

又如：数-33.0740，还原后结果的存放是：

f=1

a=(3, 3, 0, 7, 4)

k=5

j=3

【算法说明】

x：array[1..10] of char；可放长度定为10；首先读入字符串，然后处理数的符号，在还原的过程中，需要判定正数部分与小数部分，同时去除多余的空格和小数点，并约定输入是正确的，不用作出错检查。

【程序清单】

```
begin
for i:=1 to 10 do a[i]:=0;
for i:=1 to 10 do read(x[i]);
j:=0;f:=0;k:=0;b:=0;
if x[1] = '-' then begin
                    ①
                    ②
                  end
```

```
                    else if x[1]:=' 'then I:=2
                                        else I:=1;
  while ____③____ do i:=i+1;
  while ____④____ do
    begin
      if (x[i]>='0')and(x[i]<='9')
        then begin
               k:=k+1;
               ____⑤____;
               if b=1 then ____⑥____
             end
        else if(x[I]='.')and(b=0) then b:=1;
      i:=i+1
    end;
  if j>0 then while a[k]=0 do begin
                                  ____⑦____
                                  ____⑧____
                              end
end.
```

【问题分析】

本题程序中描述的算法十分清晰,其主要算法分析如下:

(1) 初始化:数组 A 清零;输入数组 x;n 个主要变量赋初值 0。

(2) if x[1]='-' then begin

……

end

else……;

进行实数的符号处理,显然 then 子句中空格①②中有一处应记录负数标志 f:=1。另一处从下面的程序中可知,i 作为循环扫描的移动指针,确定下一位置的值为 2,因此,应填 i:=2。

(3) while ____③____ do i:=i+1;移动指针作后移处理,惟一的原因是有空格,因此③处应为(x[i]=' ')and(i<10)。

(4) 算法的主要部分是接下来的 while 循环。循环扫描 x 数组的每一个元素 x[i];若x[i]是数字字符,有效数字个数计数器 k 加 1,并将 x[i]转化为数字 a[k],所以空格⑤应为 a[k]:=ord(x[i])-ord('0'),若 x[i]='.'则表示整数部分已结束,从 x[i+1]起应是小数部分,表示这种情形的标志是 b=1,而程序的小数位数的计数应在 b=1 时进行,因此⑥处应实现这种处理 j:=j+1。

(5) 程序的最后部分将小数点后无用的 0 去除。无用的 0 是处在最后一个非 0 有效数字后面,所以应从后往前检查,若 a[k]=0(k 的起始值为有效数字的个数),k 应减 1,同时表示小数位数 J 也减 1。所以⑦,⑧应填 k:=k-1 和 j:=j-1。

本题的最终答案为：

① f:=1

② i:=2

③ (x[i]=' ')and(i<10)

④ x[i]<>'%'

⑤ a[k]:=ord(x[i])-ord('0')

⑥ k:=k-1

⑦ j:=j-1

第二节　1996年试题

一、初中组第1题

【问题描述】

设有n个不同整数的数列：例如n=4时，有4个不同整数的数列为17，4，16，5。数列中的第1个数17，比它后面的三个数都大，则称数17的逆数为3。数列中的第2个数4比它后面的数都小，则称数4的逆数为0。同时记数列中全部逆数的和称为数列的逆数。上例中，数列17，4，16，5的逆数为3+0+1+0=4

程序要求：当给出n个不同整数的数列后，求出此数列的逆数。

【算法说明】

为求得上面问题的解，设置数组a:array[1..n] of integer和逆数计数器s，然后用一个二重循环求出数列的逆数。

【程序清单】

```
const n=10;
var i,j,s:integer;
  a:array[1..n] of integer;
begin
  s:=0;
  for i:=1 to n do read(a[i]);
  for i:=1 to ____①____ do
    for j:=____②____ to n do
      if a[i]>a[j] then ____③____;
  writeln('s=',s)
end.
```

【问题分析】

因为数列中的数只与它后面的数进行比较，因此外循环的循环变量i的终值应为n-1，(因为数列的最后一个数逆数为0)，而内循环的循环变量j是第i个数的后面的数的位置，其初值应为i+1。二重循环的循环体只有一条语句，累加逆数和，所以③处为s:=s+1。

本题的最终答案为:

① n-1

② i+1

③ s:=s+1

二、初中组第2题

【问题描述】

装球:设有n个盒子(n足够大,可装入任何数量的球),分别编号1,2,…。同时有k个小球(k>0),今将k个小球装入到盒子中去。装入规则如下:

(1) 第一个盒子不能为空。

(2) 装入必须严格按递增顺序进行。

例如,当k=8,n=6时,装入方法有1,2,5或1,3,4。

(3) 在满足上面的两个条件下,要求有球的盒子尽可能多。

(4) 装完后,相邻盒子中球的个数差的绝对值之和最小(未装的盒子不计)。

如上例中

装入法1,2,5,则差的绝对值之和为2-1+5-2=4

装入法1,3,4,则差的绝对值之和为3-1+4-3=3

程序要求:给出k(k表示小球的个数)之后,求出满足上述四个条件的装入方法。

【算法说明】

设置数组a:array[1..n] of integer;用数组元素代表盒子,然后依次装入小球。

【程序清单】

```
const n=20;
var i, j, k, l:integer;
  a:array[1..n] of integer;
begin
  readln(k);
  ____①____;
  j:=1;
  while ____②____ do begin
    a[j]:=j;____③____;j:=j+1
  end;
  l:=j-1;
  while k>0 do begin
    ____④____;
    k:=k-1;
    l:=l-1;
  end;
  for I:=1 to ____⑤____ do
```

```
    write(a[I]:4)
end.
```

【问题分析】

由题目给出的条件(1),第一个盒子至少放一个小球,即 a[i] > =1;根据条件(2)盒子中放的小球数必须严格按递增的顺序,因此 a[1] <a[2] <…<a[n-1] <a[n]。为达到放球的盒子尽可能多,也就是 n 尽可能大的目的,那么在小球数 k 确定的情况下,对每个盒子放球时,要尽可能节约,显然 a[1] =1,a[2] =2,…,a[n] =n 是最节约的放法,具体实现时,先根据盒子的编号,从 1 号盒子放 1 个球,2 号盒子放 2 个球,…,n 号盒子放 n 个球,n 的确定可根据余下的球数小于 n+1 为止。也就是第 n+1 个盒子就不够放了。因此,填空②、③处的作用应是余下的球数已不够放下一号盒子,所以③处是计算余下的球数,应填 k: =k-j,而②处为循环控制条件 k > =j。上述这种放球的思路可以仔细阅读程序得到启发:语句 a[j]: =j 告诉我们程序先利用第 j 号盒子放 j 个球的方法,满足题目给出的前三个条件。

用上述方法放球后,如果余下的球数为 0,放球过程结束,如果还有球多出来,把这些余下的球再分配到各个装球的盒子中,考虑到条件(4):装完之后,相邻的盒子中球的个数差的绝对值之和为最小,如何分配,这是程序要处理的第二个问题。由于 $a_1 < a_2 < \cdots < a_n$,相邻盒子中球个数差的绝对值之和:

$$\begin{aligned} s &= (a_2 - a_1) + (a_3 - a_2) + \cdots + (a_{n-2} - a_{n-1}) \\ &= a_2 - a_1 + a_3 - a_2 + \cdots \\ &\quad + a_{n-2} - a_{n-1} + a_n - a_{n-1} = a_n - a_1 \end{aligned}$$

显然 s 与第 1 个盒子、第 n 个盒子的球数有关,增加第 1 个盒子的球数,可以使 s 更小,但第 1 个盒子加 1 个,后面的盒子至少都要加 1 个,而多出来的球数 < =n,所以只能从编号大的加起,逐个往编号小的盒子中加 1 个球,直到加完为止(最多加 n 个盒子)。程序中空格④应填 a[1]: =a[1] +1。变量 j 的最后值为装球盒子数加 1,所以输出部分循环 i 的终值应为装球的盒子数:j -1。

本题的最终答案为:

1　for i: =1 to n do a[i]: =0 {去除这一句无妨程序的功能}
2　k > =j
3　k: =k-j
4　a[1]: =a[1] +1
5　j-1

三、初中组第 3 题/高中组第 1 题

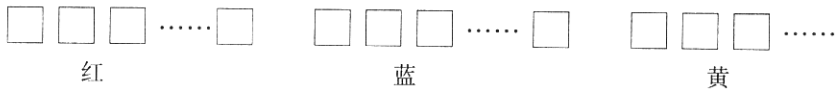
【问题描述】

积木游戏:设有 n 个小木块排成一排,如下图:

□ □ □ ……

游戏开始时,每个小木块向下的一面涂有红、黄、蓝三种颜色之中的一种(约定:0 表示红色,1 表示黄色,2 表示蓝色)。要求通过翻看与交换方式对小木块重新排列(翻看的规则为每

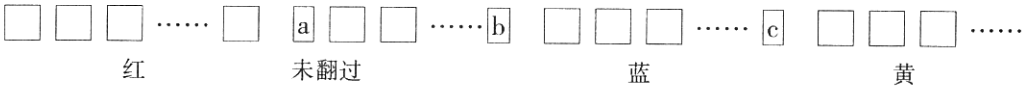
个小木块只能看一次),最终成为下面的形状:



即相同颜色的木块排列在一起,设计一个翻看与交换的方案,使得用最少的交换次数实现上面的要求。

【算法说明】

翻看小木块时,可以从两端进行。例如,设中间状态如下:



此时,可以从两个方向看,即从 a 或 b 处开始:

1. 若看 a 则有三种可能性:

   为红色,则不用交换

   为蓝色,交换一次,即 a 与 b 交换

   为黄色,交换两次,即 c 与 b 交换一次,然后 a 与 c 再交换一次

   此时,平均交换次数为 1。

2. 若看 b,也有三种可能性:

   为蓝色,则不用交换

   为红色,交换一次,即 b 与 a 交换。

   为黄色,交换一次,即 b 与 c 交换。

   此时,平均交换次数为 2/3。

由此可见,从 b 处翻看直到游戏结束,次数最少符合题目要求。

【程序清单】

```
const n=20;
var i,tem,r,b,y:integer;
    a:array[1..n] of 0..2;
begin
  for i:=1 to n do read(a[i]);
  r:=1; ____①____;y:=n;
  while ____②____ do
    if ____③____ then begin
      tem:=a[r];a[r]:=a[b];a[b]:=tem;
      r:=r+1
    end
    else if ____④____ then begin
      tem:=a[b];a[b]:=a[y];a[y]:=tem;
      ____⑤____; ____⑥____;
    end
    else b:=b=1
```

```
for i:=1 to n do write(a[i]:3)
  end.
```

【问题分析】

根据算法说明，从 b 处翻看小木块底面的颜色，分三种情形。程序中用嵌套的 if - then - else 语句将三种情形区分开来，其中 else b:=b-1;的处理中不发生交换，显然翻看到的颜色为蓝色。③处的条件根据 then 子句中交换 a[b]与 a[r]且只有 r 往后移动一位。显然翻看到红色，所以③处应填 a[b]=0。④处的条件应为翻看到的黄颜色，所以应填 a[b]=1。关键是⑤和⑥处，程序作何处理。指针 y 相当于示例中的 c 处，该处应为蓝色，其后面就是黄色木块区域。当 b 处翻看到黄色时，将 a[b]与 a[y]交换位置，此时 y 处变为黄色也就是黄色木块区域向左伸长了一块。原来 y 处的兰色木块被换到了 b 处，因此，未翻过区域的右端应向左缩进一位，即⑤、⑥处分别位 y:=y-1 及 b:=b-1。

b 的初始位置为 n 块木块的最右端处，1 应为 b:=n，整个未翻看过的区域在 r 与 b 之间，其中 r 为左端，b 为右端，所以循环控制条件应为 b>=r。

本题的最终答案为：

① b:=n

② b>=r

③ a[b]=0

④ a[b]=1

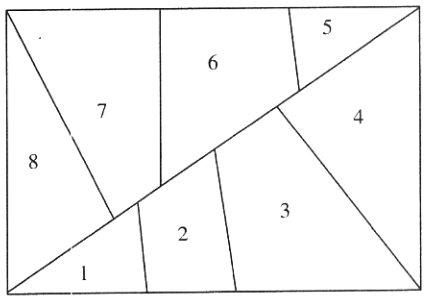
⑤ b:=b-1

⑥ y:=y-1

四、高中组第 2 题

【问题描述】

四色问题。设有下列形状的图形：有 8 个区域，其编号为 1，2，…，n。（n=8）



	1	2	3	4	5	6	7	8
1	0	1	0	0	0	0	1	1
2	1	0	1	0	0	1	1	0
3	0	1	0	1	0	1	0	0
4	0	0	1	0	1	1	0	0
5	0	0	0	1	0	1	0	0
6	0	1	1	1	1	0	1	0
7	1	1	0	0	0	1	0	1
8	1	0	0	0	0	0	1	0

图形中各区域的相邻关系用上边的邻接矩阵表示：1——相邻，0——不相邻。

问题要求：将上面图形的每一个部分涂上红(1)，黄(2)，蓝(3)，绿(4)四种颜色之一，要求相邻的部分不同色。

【算法说明】

用数组r:array[1..n,1..n] of 0..1; 表示邻接矩阵

s:array[1..n] of Integer; 表示涂的颜色。

采用回溯的方法,首先给第一个图形涂上红色(1),然后在下面的图形中依次涂上其他颜色,当有矛盾时回溯解决。

【程序清单】

```
const n=8;
var i, j, k:integer;
  r:array[1..n, 1..n] of 0..1;
  s:array[1..n] of integer;
begin
  for i:=1 to n do
    begin
      for j:=1 to n do read(r[i, j]); readln
    end;
  ____①____; i:=2; j:=1;
  while i<=n do
    begin
      while(j<=4)and(i<=n)do
        begin
          k:=1;
          while ____②____ do k:=k+1;
          if k<i then ____③____else begin
                                      ____④____
                                      i:=i+1;   j:=1
                                    end
        end;
      if j>4 then begin i:=i-1;   ____⑤____end;
    end;
  for i:=1 to n do writeln(i,'→',s[i])
end.
```

【问题分析】

本题只要求出一种符合条件的涂色方案,采用回溯算法。其中,数组s存放涂色的方案,即s[i]为第i号区域涂的颜色(红色为1,黄色为2,蓝色为3,绿色为4),在进行回溯时,需要将搜索位置返回前一个点,因此数组s的操作依照栈操作的特性。观察程序可知栈顶指针i指向栈顶元素的后一个位置,即入栈时,先s[i]←x,再i:=i+1;出栈时,先i:=i-1,再x←s[i],因此在阅读本题程序时,一定要注意栈指针设置方式。程序中变量j代表颜色号,变量k循环控制先于当前区域i涂色的区域号。

本题的主要算法分析如下：

（1）二重循环 i,j 读入邻接矩阵。

（2）第一块区域涂红色，所以①处应为 s[1]:=1，栈指针指向下一块区域 i:=2，颜色号 j 从 1 开始试探。

（3）对当前 i 区域及准备涂的颜色 j 是否会“相邻区域颜色不同”，也就是说在 i 区域之前已经涂色的区域 1～i-1 中是否存在既与区域 i 相邻，又涂了颜色 j 的区域。利用下列语句组进行判断：

k:=1;　while ___②___ do k:=k+1;

k 的值若小于 i，表示与区域 i 相邻且已涂了第 j 号颜色的区域存在，因此②处的循环控制条件应为(k<i)and((s[k]<>j)or(r[i, k]<>1))，由于 r[i, k]只有 0 和 1 两个值，所以(s[k]<>j)or(r[i, k]<>1)可写为 r[i, k]*s[k]<>j。

（4）while(j<=4)and(i<=n)do 语句寻找第 i 号区域的颜色。因此对于上述 k 的值，当 k<i 时表示当前颜色不符合要求，③处应为 j:=j+1，④处将颜色 j 涂在区域 i 上：s[i]:=j。

（5）算法(4)结束后有两种可能：第 1 种情形是第 i 号区域找到了所涂颜色 j，这时 j<=4，第 2 种情形是第 i 号区域找不到合适的颜色可涂，这时 j>4。对于第 2 种情形，必须回过去调整已涂颜色区域的涂色方法，这就是回溯。所以 if j>4 then 子句中实现回到上一区域，寻找上一区域新的颜色。经 i:=i-1 后，s[i]表示上一区域所涂的颜色号，所以该区域的颜色要从 s[i]+1 号颜色开始涂。故⑤处为 j:=s[i]+1。

（6）外循环 while i<=n do 控制区域到 n 号区域涂好颜色为止。

本题的最终答案为：

1　s[1]:=1

2　(k<i)and(s[i]*r[i, k]<>j)或(k<i)and((s[k]<>j)or(r[i, k]<>1))

3　j:=j+1;

4　s[i]:=j

5　j:=s[i]+1

五、高中组第 3 题

【问题描述】

多项式加法运算：一个仅含有 x 的多项式可以用下列的方式表示：

（系数,指数),(系数,指数),…,(0, 0)。其中(0, 0)作为结束标志。

例如：$P(x)=4x^6-3x^3+2x^2-1$ 可表示为：(4, 6),(-3, 3),(2, 2),(-1, 0),(0, 0)

$Q(x)=x^4-x+1$　可表示为：(1,4),(-1,1),(1,0),(0,0)

当用上面的方式给出 2 个多项式之后，程序对这两个多项式进行加法运算，结果也用上面的方式给出。

例如：上面的 P(x)和Q(x)相加的结果为：$4x^6+x^4-3x^3+2x^2-x$

表示结果为：(4, 6),(1, 4),(-3, 1),(2, 2),(-1, 1),(0, 0)

【算法说明】

多项式可用数组 p:array[1..n, 1..2] of integer 表示。

分别以 p1 表示 p，p2 表示 q，p3 表示结果。处理的过程为将 p 赋值到 p3，然后逐项检查

q,当发现有相同的方次时,进行系数相加;当发现没有相同方次时,插入到 p3 中去。

【程序清单】

```
var x, y, i, i1, j, j1, j2:integer;
    p1,p2,p3:array[1..20, 1..2] of integer
begin
  j1:=0;
  write('Input P(x) =');
  read(x, y);
  while x<>0 do begin
    j1:=j1+1; p1[j1,1]:=x;p1[j1, 2]:=y;
    read(x, y)
  end;
  j1:=j1+1;  p1[j1, 1]:=0;  p1[j1, 2]:=0;
  write('Input Q(x) =');
  read(x, y); j2:=0;
  while x<>0 do begin
    j2:=j2+1; p2[j2,1]:=x; p2[j2, 2]:=y;
    read(x, y)
  end;
  j2:=j2+1; p2[j2, 1]:=0; p2[j2, 2]:=0;
  for i:=1 to j1 do begin
    p3[i, 1]:=p1[i, 1];
    p3[i, 2]:=p1[i, 2]
  end;
  i:=1;
  while ____①____ do begin
    If ____②____ then begin
      for j:=j1 down to 1 do begin
        p3[j+1, 1]:=p3[j, 1];
        p3[j+1, 2]:=p3[j, 2]
      end;
      p3[i, 1]:=p2[i, 1]; p3[i, 2]:=p2[i, 2]; j1:=j1+1
    end
    else begin
      i1:=1;
      while p2[i, 2]<p3[i1, 2] do ____③____;
      if p2[i, 2]=p3[i1, 2] then
        p3[i1, 2]:=____④____
```

```
        else begin
                For j:=j1 downto i1 do begin
                p3[j+1, 1]:=p3[j, 1];
                p3[j+1, 2]:=p3[j, 2]
              end;
          p3[i1, 1]:=p[I, 1]; p3[i1, 2]:=p3[I, 2];____⑤____;
        end;
        end;
        i:=i+1;
        end;
        for j:=1 to j1-2 do write('(', p3[j, 1],',', p3[j, 2],'),');
      writeln('(',p3[j+1, 1],',',p3[j+1, 2],')');
end.
```

【问题分析】

程序实现的是线性表的归并算法。多项式相加的规则是指数相同，系数相加。多项式 p 和 q 利用了将系数不为 0 的项用一对数字表示某一项的方式，分别存放在一个二维数组中。p 和 q 相加的结果用另一个二维数组存放。根据算法说明，归并运算实际上是在表示多项式 q 的数组 p2 和表示相加结果的数组 p3 之间进行，p3 的初始值为表示多项式 p 的数组 p1，其初始长度为 p1 的长度。以后在归并过程中，p3 的长度将会变大，因此，仔细阅读程序可发现，表示 p3 长度的变量 j1 是一个关键的量。程序的主要部分是如何实现将 p2 的每一分量加入到数组 p3 中去。

程序利用循环扫描数组 p2 的方法，其扫描指针为变量 i，因此①处为循环控制条件，由题意可知，应为 p2[i, 1] < >0；下面的循环体用嵌套的 if - then - else 结构分别处理 p2 数组的当前分量 p2(i)是插入到 p3 还是与 p3 中某一分量合并(指数相同时)。从 if 条件②成立时处理的程序段 for j:=j1 downto 1 do …可知，是将 p3 各项依次向后移一位置，p2(i)插入 p3 的第 1 个位置，可知，②应为 p2[i, 2] > p3[1,2]，即 p2(i)的指数大于 p3 的最大指数(注：数组表示多项式均利用降幂形式)。当②处条件不成立时，p2(i)将与 p3 中各分量逐项比较，程序用语句 i1:=1;while p2[i, 2] < p3[i1, 2] do ____3____; 来实现，③处实现的时表示 p3 分量位置的指针 i1 往后移，直到 p2[i, 2] < p3[i1, 2]不成立，所以空格③处应填 i1:=i1+1;然后比较 p2 和 p3 当前分量的指数，若相同，则系数相加 p3[i1, 1]:=p3[i1, 1]+p2[i, 1]。指数不同，此时 p2[i, 2] > p3[i1, 2]，就把 p2 的第 i 个分量插入到 p3 的 i1 位置。空格⑤处应处理 p3 的长度增加，即 j1:=j1+1。

题目给出的程序中，忽略了一种情形，当 p2 与 p3 中两个同类项相加，系数之和为 0 时，是否要保存在 p3 中，程序没有进行处理，因此，p3 中会出现系数为 0 的项，读者可自行将该程序作一改动，使之更符合题目要求。

本题的最终答案为：

① p2[i, 1] < >0

② p2[i, 2] > p3[1, 2]

③ i1:=i1+1

④ p3[i1,1]+p2[i,1]

⑤ j1:=j1+1

第三节 1997年试题

一、初中组第1题

【问题描述】

读入n个不相同且不为0的数(1<=n<=100),不用排序,求出其中第r个大的数(1≤r≤n),即有r-1个数比它大,其余的数都比它小。例如:输入3,14,22,15,17,6,其中第3个大的数为15。

【算法说明】

以数组a[1..100]记录读入的n个数,并以0结束(0本身不是n个数中的数)。然后从第一个数开始,将它与其余的数进行比较并记录出比它大的数的个数(存于变量y中),若y=r-1时,得到所求结果:否则对下一个数进行同样的处理。

【程序清单】

```
var r,i,j,k,x,y:integer;
    a:array[1..100] of integer;
  p:boolean;
begin
  j:=0;
  readln(x);
  while ____①____ do
    begin
    ____②____
    a[j]:=x;
    ____③____
    end;
  readln(r); p:=true; i:=1;
  while p do
  begin
    ____④____; y:=0;
    for k:=1 to j do
    if x<a[k] then ____⑤____;
    if ____⑥____ then begin
                        writeln(x);
                        p:=false
```

```
                    end
        else i:=i+1
    end
end.
```

【问题分析】

该程序由两部分组成:输入 n 个数据,以 0 结束输入;输入 r 并求 n 个数据中第 r 个大的数并输出。

由程序中语句 a[j]:=x;可以判定,输入数据到变量 x,然后由下标变量 j 决定出 x 存入的数组元素 a[j];由于 j 的初值为 0(循环外的语句 j:=0;)这样②很显然地应为 j:=j+1;整个循环由当循环控制,所以①的条件必须为 x<>0;由于第一次读入数据到 x 中的语句在循环之外,由此可以推出③应为 Readln(x);这样输入 n 个数据的部分便填空完善好了。出循环时 j 中正好放着 n 的值。它被用于循环语句 for k:=1 to j do 语句中,控制 x 和所有 n 个已知数比较,以决定它是第几个大的数。在从 n 个数中找第 r 个大的数的循环中,由于 k 是循环控制变量,由 if x<a[k]可知要确定是否为第 r 个大的数在 x 中,它一定是由数组元素预先赋给 x,可以推定该数组元素的下标不可能是 k,只可能为 i,所以④应为 x:=a[i];⑤当然为y:=y+1;⑥也就自然成为条件 y=r-1 了。

一般说来初步确定填空结果后还应更进一步的阅读整个程序,推敲所填结果是否正确。由于输入数据时,用户以 0 作为输入结束,为了防止 n 过大,①的改进为(x<>0)and(j<100)可能更好一点。

阅读程序另有一个附带的目的是学习他人编程的优点和特点,以不断充实和提高自己的程序设计能力。该程序在找第 r 个大数的过程中,通过逻辑变量 p 控制找到第 r 个大数后中止继续找的过程,便是可以借鉴的控制循环条件。

本题最终的答案为:

① x<>0 或(x<>0)and(j<100)

② j:=j+1

③ readln(x)

④ x:=A[i]

⑤ y:=y+1

⑥ y:=r-1

二、初中组第 2 题

【问题描述】

在进行正整数的除法运算时,可以通过减法来实现。例如 x÷y=q…r(q:商,r:余数)可通过下列的方式实现:

q:=0; r:=x;

while r>=y do begin r:=r-y; q:=q+1 end;

结果,商在 q 中,余数在 r 中。

【算法说明】

上面的算法有一个缺点,就是当 x 比较大、y 比较小时,则运算的次数非常多,速度太慢。为提高速度,下面给出改进的算法:先找一个非常接近 x 的数 w,且满足:$w = y * 2^k$,$y * 2^{k-1} <= x < w$,然后通过减法与移位的运算,以较少的运算次数完成除法。

【程序清单】

```
var x, y, w, r, q:integer;
begin
    readln(x);
    r:=x;
      ____①____
      while w<=r do ____②____
      q:=0;
      while ____③____ do
      begin
        w:=w div 2;
        ____④____
        if r>=w then begin
                       ____⑤____;
                     r:=____⑥____;
                     end;
      end;
      writeln(q, '…', r);
end.
```

【问题分析】

对于该题算法说明,程序实现的关键在定初值 w 和除法过程。对照程序循环语句 while w <=r do ____②____;是实现选定 w 初值的循环($w = y * 2^k$;$y * 2^{k-1} <= x < w$)。可见②应为 w:=w+w(乘以 2 的幂次)或 w:=w*2;相应地①即为 w 的开始初值,应为 w:=y;程序以下的部分应为除法过程,由于赋值得 $w = y * 2^k$,且满足 $y * 2^{k-1} <= x < w$;在 w>y 时除法是可以进行的(k>0 的情况,k=0 时,商 q=0,r=x,除法过程不必进行)。因此条件③应为 w>y;商怎样决定呢?关键的语句应为 w:=w div 2,这一语句相当于将除法的除数除以 2,那么保持除法式子成立必须商扩大 2 倍,可见④应为 q:=q+q;程序最后⑤⑥要填的部分正是问题描述中给出的除法过程,所以⑤为 q:=q+1;⑥r:=r-w;程序中描述的过程在逐步深入理解的过程中有时还可以通过举例来模拟。例如 17÷3=5…2 相当于数学式子 17=3*5+2。程序中首先求得 w=3*2^3(k=3),相除过程中 w,q,r 分别对应于除数,商和余数,它们在除法过程中的变化情况:可用数学的等式来表达:

17	=	w	*	q	+	r	
		$3*2^3$		0		17	开始状态
		$3*2^2$		（除数除以2） 0（商乘以2）			第一次除法过程
17		$3*2^2$		1		5	
		$3*2$		（w div 2） $2*(q+q)$			第二次除法过程
17		$3*2$		2		5	
		3		（w div 2） $4*(q+q)$			第三次除法过程
17		3		5		2	

这时 w > y 除法过程结束。

该问题的填空答案应为：

① w := y

② w := w + w

③ w > y

④ q := q + q

⑤ q := q + 1

⑥ r - w

三、初中组第3题高中组第1题

【问题描述】

一个正整数（非素数）可以表示成它的因子（1与其本身除外）的乘积。例如：12有因子2,3,4,6,所以可表示为：12 = 2 * 2 * 3 = 4 * 3 = 2 * 6。给出任一个正整数n，求出它所有的因子乘积的表达式（交换律得出的不同式子算同一种）。

【算法说明】

读入一个整数n，首先求出它的所有的因子以及每个因子可能的次数。例如：整数48：

因子：2　3　4　6　8　12　16　24

次数：4　1　2　1　1　1　1　1

将上面的结果存入数组a：array[0..20, 1..2]中。其中：a[i, 1]表示因子；a[i,2]表示次数。然后用简单回溯的方法求出所有可能的表示：

数组b[0..20]记录取数情况；c：array[1..20]工作单元。

【程序清单】

```
var a:array[0..20, 1..2] of integer;
c,b:array[0..20] of integer;
n, m, i, j, s, k, l:integer;
begin
    writeln:readln(n);
```

```
for i:=1 to 20 do a[i,1]:=0;
    ①    ; a[0..2]:=1; j:=0;
for i:=2 to n-1 do
begin
    s:=0; m:=n;
    while(m<>0)and(m mod i=0)do
    begin
      m:=m div i;
      ②    ;
    end;
    if   ③    then begin
                        j:=j+1;   ④    ;
                        a[j,2]:=   ⑤    ;
                      end
    end;
  for i:=0 to j do b[i]:=0;
    whil b[0]=0 do
    begin
      k:=j;
      while   ⑥    do k:=k-1;
      b[k]:=b[k]+1;
      for l:=   ⑦    do b[l]:=0;
      s:=1;
      for i:=1 to j do
      if b[i]<>0 then for l:=1 to b[i] do
                    ⑧    ;
      if s=n then begin
                    for i:=1 to j do c[i]:=b[i];
                    write('(');  m:=1;
                    for i:=1 to j do
                    while(c[i]>0)and(m<>n)do
                    begin
                      m:=m*a[i,1];
                      if m=n then write(a[i,j])
                      else begin
                             write(a[i,1],'*');     c[i]:=c[i]-1;
                             end;
                    end;
```

```
            writeln( ')');
            end
        end
    end.
```

【问题分析】

程序的开始部分显然是要根据输入的 n 将因子及因子的次数存入数组 a 中，由 for i:=1 to 20 do a[i, 1]:=0;可知程序该循环上一行三个语句是对 a[0, 1],a[0, 2]初始化。如果继续读程序可以发现②实现记录因子的可能次数，即②应为 s:=s+1;③④⑤部分是决定了因子 i 重复 s 次后填入数组 a 的过程，所以③应为 s>0;④为 a[j, 1]:=i;⑤为 a[j, 2]:=s;回过头来看 a[1, 1]中填写的是从 2 开始试探的第一个因子(≠1)，那么 a[0,1]初始化的值也就应该是特殊因子 1 了，所以①应为 a[0, 1]:=1;因子填写完毕后，控制变量 j 中正好是不等于 1 的不同因子个数。程序的下面一部分应是利用 b 和 c 用回溯法求分解式输出的过程了。回溯过程是一个利用栈的过程，这一点在竞赛大纲中基本算法知识点是有明确要求的。数组 b 用于记录在求因子表达式中取用某因子的个数情况。因此对数组 b 究竟怎样把它作为栈进行回溯的过程，在算法说明中就略而不谈了，要由参赛者根据程序去理解。以 n=12 为例，程序第一部分获得数组 a 如下：

a:	1	2	3	4	6	…
	1	2	1	1	1	…
	0	1	2	3	4	j=4

从已给的程序可知栈的开始位置为 k=j，即栈的初始状态为：

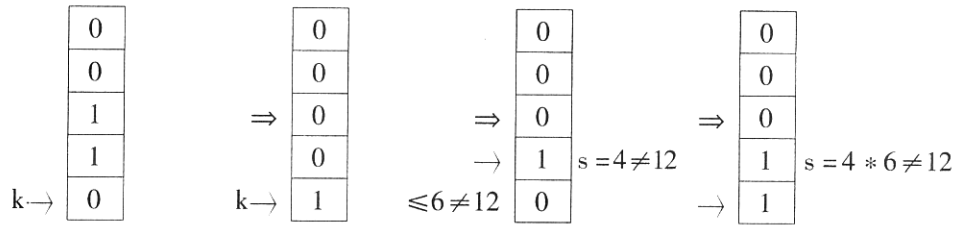
0	0
1	0
2	0
3	0
k=j→4	0

栈中的数值(取因子个数)决定了是否输出分解式和进退栈过程(if s=n then 输出分解式)，例如 b 为：

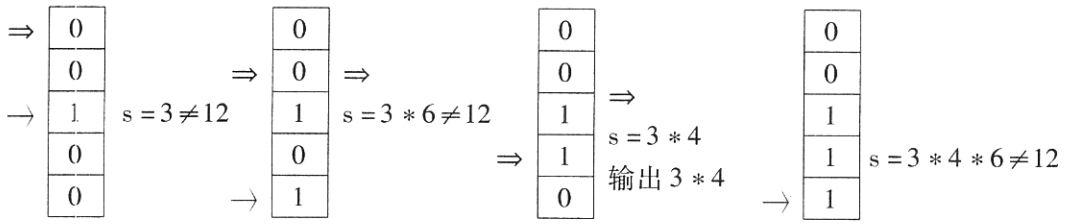
	0
	0
	1
→	1
	0

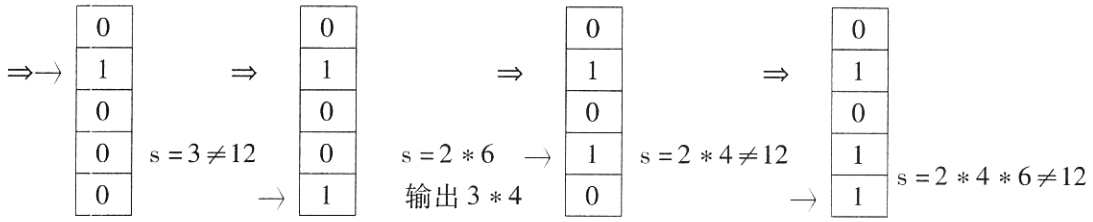
时，s=a[2, 1]*a[3, 1]=3*4=12

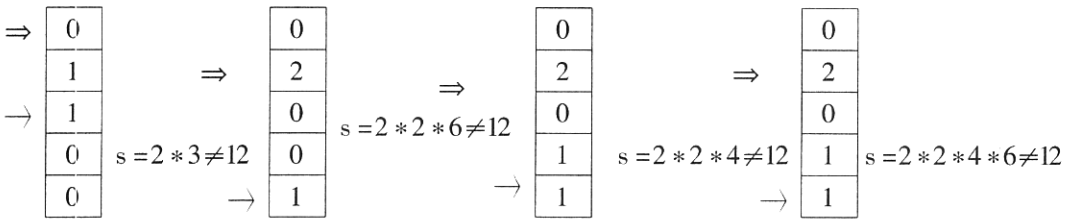
所以输出分解式(3*4)：可见⑧应为 s:=s*a[i, 1];根据回溯的思路，b 栈的变化情况应为：

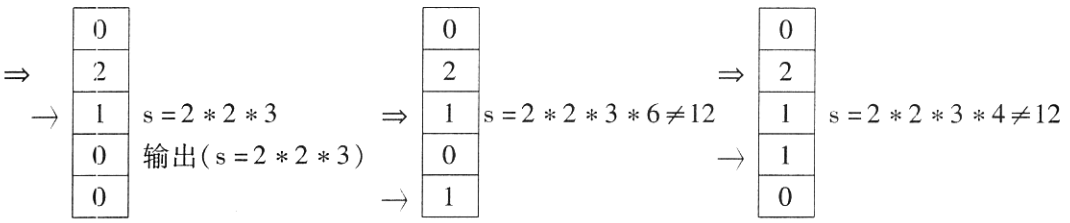


b 初始状态

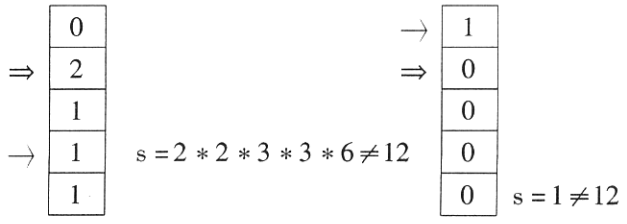








可见⑥应为 b(k) = a(k,2);⑦应为 for l: = k + 1 to j do b[l]: =0;



该题的填空答案应为：

① a[0，1]：=1（只要≠n）

＊如果分解式中要求 n=n 这种特殊情况，那么初值就应该取 a[0，1]：=n

② s：=s+1

③ s>0

④ a[j，1]：=i

⑤ s

⑥ b[k]=a[k，2]

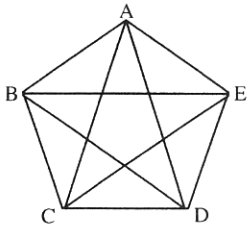
⑦ k+1 to j

⑧ s：=s＊a[i，1]

四、高中组第2题

【问题描述】

给出一个凸多边形，可以取得若干个内接三角形，同时约定内接三角形必须有一条边（仅能有一条边）与凸多边形的边相重合，例如：下面的5边形中，可能有的内接三角形有5种：



△ACD，△BDE，△CEA，△DAB，△EBC

问题要求：当依次给出凸多边形的每个顶点的2个坐标之后，找出一个面积最大的内接三角形，输出该三角形的面积与三个顶点的坐标。

【算法说明】

凸多边形的每个顶点用一对坐标（x，y）表示：用数组 p：array[1..2＊n]of point；存贮输入的顶点坐标；同时编制一个由三角形的三个顶点计算其面积的函数 sea。

【程序清单】

```
const n=6;
type point=record x，y:real end;
var p:array[1..2＊n] of point;
    i，j:integer; q1，q2，q3:point;
function sea(p1，p2，p3:point):real;
  var s1，s2，s3，p4:real;
   begin
   s1:=sqrt(p1.x-p2.x)＊(p1.x-p2.x)+(p1.y-p2.y)＊(p1.y-p2.y);
   s2:=sqrt(p1.x-p3.x)＊(p1.x-p3.x)+(p1.y-p3.y)＊(p1.y-p3.y);
   s3:=sqrt(p2.x-p3.x)＊(p2.x-p3.x)+(p2.y-p3.y)＊(p2.y-p3.y);
```

```
    p4:= ____①____;sea:=sqrt(p4*(p4-s1)*(p4-s2)*(p4-s3));
  end;
begin
  for i:=1 to n do readln(p[i].x, p[i].y);smax:=0;
  for i:=1 to n-1 do ____②____
  for i:=1 to n do
  for j:= ____③____ do
  if ____④____ then
  begin
    smax:=sea(p[i], p[i+1], p[j]);
    q1:=q[i]; q2:= ____⑤____; q3:=p[j]
  end;
  writeln(smax, q1.x, q1.y, q2.x, q2.y, q3.x, q3.y)
end.
```

【问题分析】

该问题的程序在阅读问题描述和算法说明后,首先要理解计算由 p1,p2,p3 三点决定的三角形面积的过程 sea,已知两点 p1,p2,边 p1p2 的计算公式应为:

$$\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$$

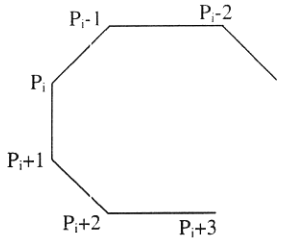
其中(x_1, y_1)是点 p_1的 x, y 坐标;(x_2, y_2)是点 P_2的 x, y 坐标。

已知三边长 s1, s2, s3 求三角形面积可用公式:

$$\sqrt{p4(p4-s1)(p4-s2)(p4-s3)}$$

其中 p4=(s1+s2+s3)/2;

可见过程中的①应为,p4=(s1+s2+s3)/2;



主程序的第一部分应为读入凸多边形各顶点的坐标值,然后才是求最大面积三角形的处理过程,从程序可以看出最大面积的三角形的面积值在变量 smax 中。怎样取三点组成三角形,从赋值语句 smax:=sea(p[i], p[i-1], p[j])可以看出取三角形的规律来:(左图)

当取 $p_i p_{i+1}$为一边时,三角形的另一个顶点只可能取 p_{i+3}, p_{i+4},…,p_{i-2};也就是说 j 循环的控制变量变化范围,即③应为 for j:=i+3 to i+n-2 do;④只是取最大面积的常规比较,即 smax<sea(p[i], p[i-1],p[j]); q1,q2,q3 是最大面积三角形三顶点的坐标,所以⑤应为 q2:=p[i+1];程序中为了适应 i 从 1 变化到 n 时 j 从 4 变到 2n-2,p 数组的上界取为 2(n,因此在 p_1~p_n点数据输入完毕后,要对 p_{n+1}, p_{n+2}, …, p_{2n-2}赋值,即②应填为 p[n+i]:=p[i]。

这样该题的最后答案为:

① (s1+s2+s3)/2

② p[n+i]:=p[i]

③ i+3 to i+n-2

④ smax<sea(p[i], p[i+1], p[j])

⑤ p[i+1]

五、高中组第3题

【问题描述】

拼图形:边长为1的正方形面积为1,从边长为1的正方形出发可以用2个边长为1的正方形拼成面积为2的长方形:

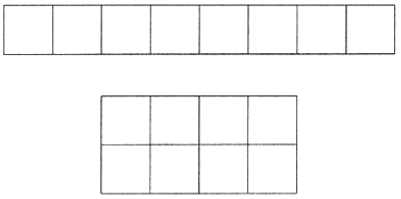
同时约定:

1. 边长对应相等的长方形被认为是相同的;

2. 长度相等的边才能拼接,且两个边必须重合。

从面积为2的长方形出发,用2个面积为2的长方形可拼出面积为4的长方形(包括正方形),拼法如下:

同样再从面积为4的长方形(包括正方形)出发,可以拼成面积为8的长方形,拼法如下:

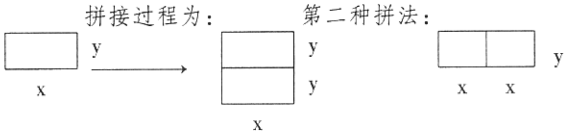


可以按上面的方法继续拼下去。

问题:输入一个数n,输出面积不超过n的所有可能拼法。例如:当n=20时,输出(1,1),(2,1),(4,2),(8,2),(16,3)即面积为1的拼法1种,面积为2的拼法1种,面积为4的拼法2种,面积为8的拼法2种,面积为16的拼法3种。

【算法说明】

矩形可以用三个数x,y,s来表示,其中x,y表示边长,s表示面积,并用数组g[1..100,1..3]表示图形。



当给出n之后,可能拼接的次数r满足:$2^r <= n < 2^{r+1}$(不包括面积为1的拼法);用数组b[1..100]记录各种面积可能出现的拼法。

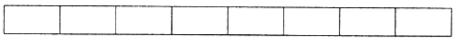
【程序清单】

```
type g=record      x, y, z:integer      end;
var g1:array[1..100]of g ;
    i, j, n, s1, jj, j1, j2, i1:integer;
    b:array[1..100]of integer;
    gw:g;
function eq(qk:g):boolean;
  var jeq:integer; p:boolean;
  begin
    p:=true; jep:=1;
    while(p and(jeq<=j))do
    if((gk.x=g1[jeq].x)and(gk.y=g1[jeq].y)
      or((gk.x=g1[jeq].y)and(gk.y=g1[jeq].x))
    then p:=false else jeq:=jeq+1;
    eq:=p
  end;
begin
  readln(n); s1:=1; jj:=1;
  while ____①____ do
  begin ____②____; jj:=jj+1 end;
        ____③____; j1:=1; j:=1; g1[j].x:=1; g1[j].y:=1; g1[j].z:=1;
    for i:=2 to jj do
    begin
      j2:=j;
      for i1:=j1 to j2 do
      begin
      gw.x:=g1[i1].x*2; gw.y:=g1[i1].y;gw.z:=g1[i1].z*2;
      if ____④____then begin
                          j:=j+1; g1[j]:=gw
                        end;
      gw.x:=g1[i1].x; ____⑤____
      if eq(gw)then begin
                      j:=j+1; ____⑥____
                    end;
    end;
    j1:=j2+1
  end;
  for i:=1 to n do b[i]:=1;
```

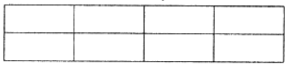
```
    for i:=1 to j do ____⑦____
    for i:=1 to n do if ____⑧____ then write('(',i,',',b[i],')');
end.
```

【问题分析】

该题需要正确的理解问题描述和算法说明中的补充说明。实际上每拼一个矩形都可能产生2种拼图结果，只是因为在拼装过程中会有相同边长而看作是同一种图形而已。例如面积为8时有两种图形

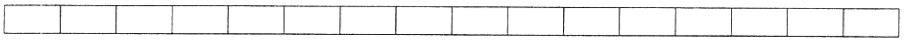


x×y=8×1

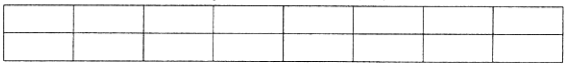


x×y=4×2

这样，面积为16时，可能的结果是如下四种：

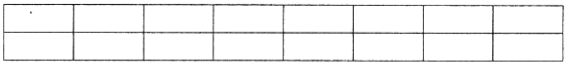


x×y=16×1　(1)

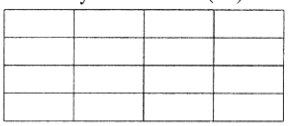


x×y=8×2　(2)

以上两种图形由8*1产生。



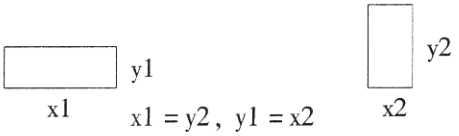
x×y=8×2　(3)



x×y=4×4　(4)

以上两种图形由4×2产生。由于(3)，(4)是同一种图形，所以面积为16时的图形只有(1)，(2)，(4)三种，这就是问题描述中n=20时，输出结果中(16,3)的产生原因。读程序时，首先要明确布尔型函数过程eq的意义，如下矩形被认为是相同的：

x1=x2，y1=y2



x1=y2，y1=x2

过程中出现的全局变量j，应为已经存贮在g1中的，拼接产生的不同矩形个数。要完善程

序填写的空格,全部在主程序中。主程序开始部分,首先找出拼接的最大次数(jj)和可拼接矩形的最大面积,这样①,②,③就比较容易推理出来了:

① 应为条件 s1 < =n;

② s1: =2 * s1;

③ jj: =jj -1;

例如:n =20,

拼接 1 次, 面积 s1 =2, jj =2;

拼接 2 次, 面积 s1 =4, jj =3;

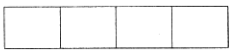
拼接 3 次, 面积 s1 =8, jj =4;

拼接 4 次, 面积 s1 =16, jj =5;

拼接 5 次, 面积 s1 =32, jj =6;

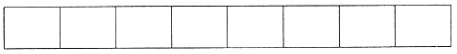
结果 jj =5,程序接着作了拼接的准备工作。

接着的循环控制 for i: =2 to jj do,可以判定是控制求各次拼接产生的矩形的循环。首先进行横向拼接,如:



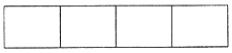
x × y =4 × 1

拼接的结果为:



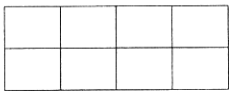
2x × y =8 × 1

可见④应为:eq(gw),表示产生的拼接结果是否已经出现过。接着应实现纵向拼接,如:



x × y =4 × 1

拼接的结果为:



x × 2y =4 × 2

可见⑤应为 gw. y: =g1[i1]. y * 2;

⑥ 应为 g1[j]: =gw;

程序的最后部分是结果的输出。根据算法说明 b[1] ~ b[n]应存贮面积 1 ~ n 的拼接矩形个数。所以⑦处应填为:b[g1[i]. z]: =b[g1[i]. z] +1;⑧也就比较显然地应为条件 b[i] < >0 了。

该题的最后结果应为:

① s1 < =n

② s1: =s1 +s1 或 s1: =2 * s1

③ jj: =jj -1 也可写为 dec(jj)

④ eq(gw)

⑤ gw.y:=g1[i1].y*2
⑥ g1[j]:=gw
⑦ b[g1[i].z]:=b[g1[i].z]+1 或 inc(b[g1[i].z])
⑧ b[i]<>0

第四节　1998年试题

一、初中组题

【问题描述】

输入一长度不超过80个字符的字符串(称为源串),该字符串由小写英文字母、空格组成,并以'.'结束。单词是由连续字母组成,两个单词之间至少有一个空格。本程序的功能为:首先找出字符串中所有单词并保留一个空格作为单词分隔,存入数组ch中。然后用键盘输入一个待查找的单词,以字符'$'结束。采用顺序查找的方法在ch中进行查找,若找到,则输出该单词在ch中出现的序号(若有多个位置出现该单词,则只输出第一个序号位置)。若不存在,则输出'NOT FOUND'。

【程序清单】

```
var
  a, b, ch: array[1..80] of char;
  i, j, k, n, m : integer;
begin
  n := 0;
  repeat
    ____①____; read(a[n]);
  until a[n]='.';
  readln;
  k:=0;
  for  i:=1 to n do
  if(a[i]>='a')and(a[i]<='z')then
  begin
  k:=k+1;
  ____②____;
end
  else if k <> 0 then if ch[k]<>'□' then {□表示一个空格,以下同}
              begin
                k:=k+1;
                ch[k]:='□'
              end;
```

```
M:=0;    ③    ;
repeat
  m:=m+1; read(b[m]);
until    ④    ;
i:=1;j:=1;k:=1;b[m]:='□';
while(i<=n)and(j<=m)do
begin
  if    ⑤    then   begin  i:=i+1;j:=j+1 end
                          else begin
                    while ch[i] <> '□' do    ⑥    ;
                    i:=i+1;j:=1;k:=k+1
                                end
end;
if    ⑦    then writeln(k:4)
             else writeln('NOT FOUND')
end.
```

【问题分析】

根据提意,该程序完成的功能可分为如下几个:

(1) 输入一个长度不超过80个字符的字符串(成为源串),该字符串由小写英文字母、空格组成,并以"."结束。

(2) 将输入的字符串切分成单词存入数组ch中,单词的概念及单词存入ch的组织方法(以一个空格隔开),题目中都给出了专门的说明,这是在理解题意时必须十分注意的地方。

(3) 在ch中查找输入的单词,特别要注意的是要查找单词的输入方法,及查找结果的输出,题目中都给了明确的说明。

这样对程序清单作一次浏览,可较为容易地看出相应功能对应的程序段;虽然相应功能的程序段之间有些语句还一下子不能确定其定义,但仔细阅读剖析时便可自然而然地明白了。

功能(1)清楚地对应着程序段:

```
n:=0;
repeat…readln;
```

可以看出该段程序的作用是逐个读入用户输入的字符,直到'.'结束,所以①处应填写的语句应该是n:=n+1;

该段程序执行完毕后,a[1],…,a[n]中存贮了输入的字符串,a[n]中为'.'字符,字符串的长度为n。

功能(2)对应的程序段为:

k:=0; for j:=1 to n do 两个语句。

根据题意分析,可以知道输入的字符串从a数组送入数组ch的差别,仅在于a中单词间可能有多个空格,而ch中单词间只能有一个空格,而且ch中最后一个单词后也是一个空格,a中字符串的结束标记'.'字符在ch中已不存在了。

例如:a 数组中输入的内容为:

a□□abc□ab□.　　(长度为 11)

整理到数组 ch 中为:

a□abc□ab□　　(长度为 9)

这样仔细阅读程序段时可得到②处应填为:

ch[k]:=a[i];

请注意边界情况,从 a 中读到 a[n]时,'.'怎样处理的,在 ch 中最后一个单词后有无空格,都需要结合程序清单仔细推敲。

功能(3)对应的程序段是程序的最后部分,比较长,要完善补充的内容(填格子)也较多,相应的解题的困难也就集中在该部分了。

这一部分对应的程序段中

```
m:=0;
repeat
  m:=m+1; read(b[m])
until ____④____;
```

很显然是用于读入要查找的单词,根据题意,要查找的单词输入时由'$'结尾,所以④处应填写的语句应为:

b[m]='$';

该部分程序段中,while 语句用于在 ch 中查找输入的单词时显然的。

由于程序段中出现了 ch[i],while 条件中出现了(j<=m),由此可以判断 i,j 应该时比较过程中用于 ch 中元素和 b 中元素的位置指针(下标);那么根据 while 后的条件(i<=n)即可判断 n 为 ch 的长度上限,可见③处应填的内容为:

n:=k;　　(k 在第(2)功能中获得)

⑤处应填入条件:　　ch[i]:=b[j]

⑥处应填入语句:　　i:=i+1;

这样程序段中变量 k 用于记录查找单词在 ch 中的位置也就可以肯定了。

⑦处的条件,在以上格子的意义明确后,应填的内容也就可以确定为:j>m-1 或 j=m

这样该题的答案为:

① n:=n+1

② ch[k]:=a[i]

③ n:=k

④ b[m]='$'

⑤ ch[i]=b[j]

⑥ i:=i+1

⑦ j>m-1 或 j=m

二、高中组题

【问题描述】

FBZ 串问题。已知一个由0,1 字符串组成的长度为2^n的字符串。请按以下规则将已给出的字符串分解为 FBZ 串:

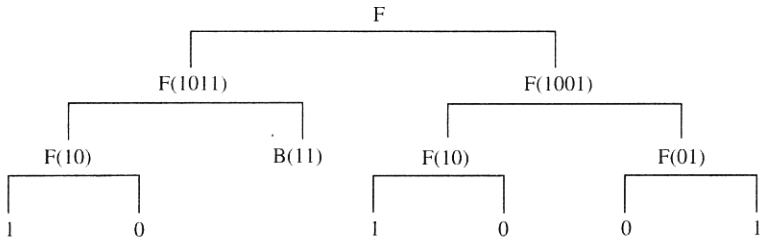
若其中字符全为'1',则称其为'B'串;

若其中字符全为'0',则称其为'Z'串;

若不全为'0',同时也不全为'1',则称'F'串。若此串为 F 串,则应将此串分解为 2 个长为2^{n-1}的子串。

对分解后的子串,仍按以上规则继续分解,直到全部为 B 串或为 Z 串为止。

例如 n=3 时,给出 01 串为:'10111001'



最后输出:FFFBZBFFBZFZB

问题:给出 01 串,分解成 FBZ 串。

【程序清单】

```
const n=8;
var
  i, j, st11, st12, s, t:integer;
  str1 : array[1..n^2, 1..n] of char;
  str2 : array[1..40] of char;
begin
  for i:=1 to n^2 do
    for j:=1 to n do str1[i, j]:='□';
  st11:=1;   st12:=1;   st2:=0;
  for i:=1 to n do read(str1[1, i]); readln;
  while ____①____ do
  begin
    s:=0;   t:=0;
    for i:=1 to n do
      begin
        if str1[st12, i] ='1' then s:=s+1;
        if str1[st12, i] ='0' then t:=t+1
```

```
        end;
    If ____②____ then   begin
                          st2:=st2+1; str2[st2]:='B'
                        end
                else if ____③____ then begin
                                          st2:=st2+1; str2[st2]:='Z'
                                        end
                Else begin
                     st2:=st2+1; str2[st2]:='F'; j:=(s+t) div 2;
                     for s:=n*2-2 downto ____④____ do
                       for t:=1 to n do
                         str1[s+2, t]:=str1[s,t];
                     st11:=st11+2;
                     for i:=1 to j do
                       begin
                         str1[st12+1, i]:=str1[st12, i];
                         str1[st12+2, i]:= ____⑤____
                       end;
                   for i:= ____⑥____ do
                     begin
                       str1[st12+1, i]:='□';str1[st12+2, i]:='□'
                     end
                   end
    end;
    for i:=1 to st2 do write(str2[i]);writeln
end.
```

【问题分析】

（1）整个程序按功能分为三个部分是十分明显的；

输入0,1字符组成的字符串（长度为2的幂次，程序中 $n=8=2^3$）；

（2）由while语句生成FBZ串存贮于数组str2中；

（3）输出形成的FBZ串。

问题处理的难点在第二部分，分解过程中0，1字符串的存贮是由二维数组str1承担的。因此弄清楚下标变量st12，及变量st11在功能（2）中的意义和变化是十分重要的。

根据题意和说明②，③是很容易确定的，它们应为：

②T=0　　表示搜索的0，1串全为1

③S=0　　表示搜索的0,1串全为0

下标变量st12，指示着要判定FBZ的0,1组成的字符串位置（行号）。其初值为1，即指向输入的0,1字符串（二维数组str1的第一行）。

当要判定的0,1字符串不是B或Z串而为F串时,程序中如何进行一分为二的呢?这是需要细细模拟程序段运行的。

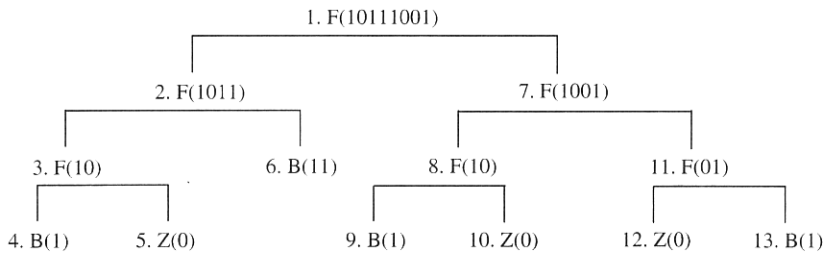
从j:=(s+t)div 2;语句可知一分为二是借助于j变量来控制的。

阅读循环语句for i:=1 to j do即可发现当前0,1字符串一分为二时,存贮于本行的下两行(st12+1行和st12+2行,本行为st12所指的行);所以⑤应该填写的内容为:

```
str1[st12, i+j];
```

相应地对于⑥也就可以填出结果了,其目的是使数组每行用不着的元素位置皆填充以' '(空格字符),所以⑥的循环控制范围应为:j+1 to n。

对于填空位置④需要弄清楚的是,为了得到输出的FBZ串,由F串进行一分为二分解,构成的实际上是一棵二叉树结构,最后输出可理解为这棵树的中序遍历,对于题目中给出的样例:



扫描0,1字符串的实际上是上图中的编号的顺序。也就是说输入0,1字符分解后存贮于str1数组的情况为:str数组元素为str1[st12,i]

st12 \ i	1	2	3	4	5	6	7	8
1	1	0	1	1	1	0	0	1
2	1	0	1	1				
3	1	0						
4	1							
5	0							
6	1	1						
7	1	0	0	1				
8	1	0						
9	1							
10	0							
11	0	1						
12	0							
13	1							
14								
15								
16								

为了获得上述存贮顺序,程序段中采用了每处理一行,将以下行下推二行的方法来存贮当前一行的一分为二的结果。可见④的填充内容为:st12+1;

相应地①也可以知道了,应为:st12<=st11。这就是说变量st11是指向待处理行的最后位置。

综上所述该题的填充结果应为：

① st12 < = st11

② t = 0

③ s = 0

④ st12 + 1

⑤ str1[st12, i + j];

⑥ j + 1 to n

二叉树的遍历算法都是以递归的方法来实现的。本例采用了一个开销较大的二维数组str1[1..2 * n, n]来实现FBZ串输出结果，程序运行工作量相对来说是相当可观的。

第五节　1999试题

一、初中组题

【问题描述】

下面程序的功能是从键盘读取a,b数组的元素，a,b数组均已从小到大排好序(无相同元素)，现将a,b合并为数组c，同样要求数组c也是从小到大排好序(有相同元素时只保留一个)。

【程序说明】

程序中N表示数组a,b的长度，i,j,k分别表示数组a,b,c的取数或存数的指针。

【程序清单】

```
const n = 8; m = 2 * n;
type
    arr1 = array[1..n]of integer;
    arr2 = array[1..m]of integer;
var
    a,b:arr1;
      c:arr2;
    i, j, k:integer;

procedure copy(x:arr1; var y:arr2; var i, j:integer);
begin
    i: = i + 1; y[i]: = x[j]; j: = j + 1;
end;

begin
(1) [for i: = 1 to n do read(a[i]);   readln;
    [for i: = 1 to n do read(b[i]);   readln;
```

```
(2) i:=1;j:=1;____①____
    while ____②____ do
      if a[i]<b[j] then copy(a, c, k, i)
                   else if b[j]<a[i] then copy(b, c, k, j)
(3)                                  else begin
                                            copy(a, c, k, i);
                                            ____③____
                                            end;
(4) while ____④____ do copy(a, c, k, i);
    while ____⑤____ do copy(b, c, k, j);
(5) for i:=1 to k do write(c[i]:4);
      writeln;
    end.
```

【问题分析】

这道题的题意十分明确,该程序中使用了过程 copy;从题意可知 a,b 数组都是 n 个元素的数组,初值皆由键盘输入,且都是排好顺序输入的,a,b 本身无相同元素,但 a,b 之间是有可能出现相同元素的,例如 n=3 时:

a[1]=3	a[2]=6	a[3]=9
b[1]=4	b[2]=6	b[3]=8

合并的数组 c 只有 5 个元素

c[1]=3　　c[2]=4　　c[3]=6　　c[4]=8　　c[5]=9

由于过程 copy 中没有要填写的空格子,因而首先弄清 copy 过程的功能就十分重要了;该过程除参数比较多外,过程体十分简单而且明白。

不妨设相应的实参为:copy(a, c, 3, 2);过程体完成的工作是:

```
a[1]   a[2]   a[3]   c[1]   c[2]   c[3]   c[4]
         ↑                      ↑
```

将 a[2]的值送 c[4],结果指针都后移了一个位置;

```
a[1]   a[2]   a[3]   c[1]   c[2]   c[3]   c[4]
         ↑                             ↑
```

这样主程序的填空就比较容易了。

(1) 输入 a,b 数组的数据;

(2) 显然是初始化 a,b,c 数组的指针初值,根据 copy 的功能,①为 k:=0 是明确的。

(3) 应该是比较 a,b 数组元素决定向 c 数组送元素的部分;因此只有 a 和 b 元素都没有送完全的情况下,才继续循环。

可见②为(i<=n)and(j<=n);

从循环体的 if 语句看到要填的空格的位置在 a[i] = b[i]的情况，通过调用过程语句 copy(a,c,k,i)已将 a[i]送入 c[k]，i 指针已向后移了一个位置，根据题意，这个情况下 b[j]是不必要送 c 数组的，所以，③应为 j: = j + 1。

(4) 这一部分表示比较进程中当 a 或 b 元素已全部送入数组 c，另一数组还有剩余元素时，只要直接逐个送 c 数组即可，可见④表示 b 数组已送完的情况，所以④应为 i < = n。

⑤的情况表示 a 数组的元素已全部送 c 数组，b 的元素还没有送完，所以⑤应为 j < = n。

最后结果：

① k: = 0

② (i < = n) and (j < = n)

③ j: = j + 1

④ i < = n

⑤ j < = n

二、高中组第 1 题

【问题描述】

求一棵树的深度与宽度。

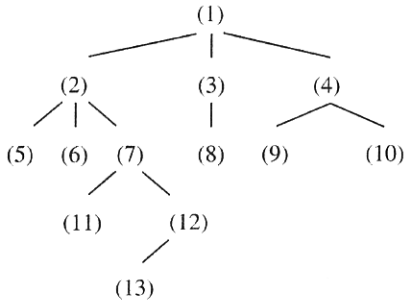
【算法说明】

树可用数组 tree:array[1..n, 1..5]of integer;

其中:tree[i, 1]表示结点号;tree[i, 2]——tree[i, 5]所属结点

如右图可表示为：

1	2	3	4	0
2	5	6	7	0
3	8	0	0	0
4	9	10	0	0
5	0	0	0	0
6	0	0	0	0
7	11	12	0	0
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0
11	0	0	0	0



在求解的过程中，用到数组 g:array[1..n, 1..7]of integer;

其中:g[i, 1]表示父结点，g[i, 2]表示层次，

g[i, 3]表示本结点号，g[i, 4]——g[i, 7]表示子女结点;

同时，设 2 个指针 sp1(取数指针)，sp2(存数指针)

【程序清单】

```
const n=13;
var i, j, k, sp1, sp2, n1, n2, jmax, p:integer;
    tree:array[1..n, 1..5]of integer;
    g:array[1..n, 1..7]of integer;
begin
(1) for i:=1 to n do
    begin
      tree[i, 1]:=i;
        for j:=2 to 5 do read(tree[i, j]);readln;
    end;

(2) sp1:=1; sp2:=1;  g[1, 1]:=0;  g[1, 2]:=1;  g[1, 3]:=1;
    for i:=4 to 7 do g[1, i]:=tree[1, i-2];

(3) while ____①____ do
    begin
      p:=g[sp1, 2];  n2:=g[sp1, 3];____②____;  j:=4;
      while ____③____ do
        begin
          n1:=g[sp1, j];  j:=j+1;____④____;
          g[sp2, 1]:=n2;  g[sp2, 2]:=p;  g[sp2, 3]:=n1;
          for i:=1 to 4 do g[sp2, i+3]:=tree[n1, i+1];
        end;
      ____⑤____;
    end;
(4) writeln('maxd=',g[sp2, 2]);  j:=1;  k:=g[1, 2];jmax:=0;
    for i:=2 to sp2 do
    if ____⑥____then j:=j+1
                  else begin
                         if j>jmax then jmax:=j;
                         ____⑦____;k:=g[1, 2];
                       end;
    if j>jmax then jmax:=j;
    writeln('max1=', jmax);
    end.
```

【问题分析】

(1) 经审题可以发现该程序主要特点是树结构的存储方式。

（2）初始化，由输入数据形成数组 tree，即由用户给定树结构。

以下的处理全属根据 tree 中的数据构筑 g 数组中数据的过程。

（3）处理根结点，根结点无父结点，属于第一层；

按层处理各结点：

从程序看 p：=g[sp1，2]指向当前处理层号，该层非零结点元素应属 p+1 层，所以②应为 p：=p+1；n2 是当前层处理的结点号；所以③应为 g[sp1，j]<>0 表示零元素都要处理；

n1 中为要处理的非零元素（结点号）；

由 g[sp2，1]：=n2，可知 sp2 指向处理结点所在的二维数组 g 的行号，根结点在第 1 行，所以④应为 sp2：=sp2+1；

由此可见控制循环（处理每个结点）的条件①应为 sp1<=sp2；⑤应为 sp2：=sp1+1；

通过（2）数组 G，对样例的数据而言成为：

	G[i，1]	G[i，2]	G[i，3]	G[i，4]	G[i，5]	G[i，6]	G[i，7]
i=1	0	1	1	2	3	4	0
i=2	1	2	2	5	6	7	0
i=3	1	2	3	8	0	0	0
i=4	1	2	4	9	10	0	0
i=5	2	3	5	0	0	0	0
i=6	2	3	6	0	0	0	0
i=7	2	3	7	11	12	0	0
i=8	3	3	8	0	0	0	0
i=9	4	3	9	0	0	0	0
i=10	4	3	10	0	0	0	0
i=11	7	4	11	0	0	0	0
i=12	7	4	12	13	0	0	0
i=13	12	5	13	0	0	0	0

（4）为结果输出部分

这时输出的 g[sp2，2]之值即为最大层号，对该例而言层号为 5，即树的深度为 5；

下面的部分将是处理树的最大宽度：

程序段由 j 记录同一层的结点个数，因此⑥应为条件 k=g[i，2]；当层号发生变更时，便表示该层的最大宽度，因此⑦应为对下一层的准备工作 j：=1。

该题的答案应为：

① sp1<=sp2

② p：=p+1

③ g[sp1，j]<>0

④ sp2：=sp2+1

⑤ sp2：=sp1+1

⑥ k=g[i，2]

⑦ j：=1

请注意,该题应用了树的深度与宽度的概念,但没有交待清楚,这是该题的欠缺。

该题中将树的结点层号最大值定义为树的深度(根结点层号为1);分支结点子女数目最大值定义为树的宽度。

三、高中组第2题

【问题描述】

用生成法求出1, 2, …, r的全排列(r<=8)。

【算法说明】

用数组a:array[1..r]of integer;表示排列;

初始化时,a[i]:=1 (i=1, 2, …, r)

设中间的某一个排列为a[1], a[2], …, a[r]

则求出下一个排列的算法为:

① 从后面向前找,直到找到一个顺序为止(设下标为j-1,则a[j-1]<a[j])

② 从a[j]-a[r]中,找出一个a[k]比a[j-1]小的最小元素

③ 将a[j-1]与a[k]交换

④ 将a[j], a[j+1], …, a[r]由小到大顺序。

【程序清单】

```
    const r=7;
    var
      n, i, s, k, j, i1, t:intger;
      a:array[1..r]of integer;
    procedure print1;
      var ik:integer;
(1) begin
      for ik:=1 to r do write(a[ik]:8);writeln;
    end
    begin
(2) for i:=1 to r do ____①____;
    print1;
(3) s:=1;
    for i:=2 to r do s:=s*i;
    s:=s-1;
```

```
(4) for i:= ____②____ do
    begin
      j:=r;
      while ____③____ do j:=j-1;        ┐算法步骤①
      k:=j;                             ┘
      for i1:=j+1 to r do
        if ____④____ then k:=i1;        ┐算法步骤②
      t:=a[j-1]; a[j-1]:=a[k]; a[k]:=t;          算法步骤③
      for i1:=j to r-1 do
        for k:=i1+1 to r do
          if ____⑤____ then
                begin                              算法步骤④
                    t:=a[i1]; a[i1]:=a[k]; a[k]:=t;
                end;
        print1;
    end;
end.
```

【问题分析】

该题的算法给出了四步，但没有给出例子，应以实例对照理解后，再去读程序就十分明确了。

取 r=5 为例

a[1]	a[2]	a[3]	a[4]	a[5]	
1	3	2	5	4	已知当前排列
			↑		

∵ a[3]<a[4]　∴ =4　　根据算法步骤①

∵ a[3]<a[4],a[3]<a[5]　∴ k=5

a[5]<a[4]

a[1]	a[2]	a[3]	a[4]	a[5]	
1	3	4	5	2	根据算法步骤③
1	3	4	2	5	根据算法步骤④生成的新排列

有了以上理解即可开始着手对程序分析了，程序中的各个对应功能段是十分明确的。

程序中：

(1) 功能十分明确，是过程 print1 的过程体。

(2) 输出生成的排列元素

主程序的初始状态，输出的第一个排列

1　2　3　4 … r

所以①应为　a[i]=i

(3) 明显地 s=r!，s-1 说明输出一组排列因此(4)循环要生成其余r!-1组排列，所以：

② 可为 i:=1 to s 或 j:=s downto 1

③ 应填为 a[j-1]>a[j]　　[算法步骤①]

结果:将从右找到的满足条件的下标

a[j-1]<a[j]的下标记录在变量 k 中;

④ 应为(a[i1]>a[j-1])and(a[i1]<a[k])　　[算法步骤②]

⑤ 应为 a[i1]>a[k]　　[算法步骤④]

整个程序完全是按照算法给定的步骤实现的,所以理解题中给出的信息是十分重要的,如果连题意给出的信息都不深入了解,要得出正确结果是十分困难的。

该题的答案为:

① a[i]:=i

② s downto 1 或 1 to s

③ a[j-1]>a[j]

④ (a[i1]>a[j-1])and(a[i1]<a[k])

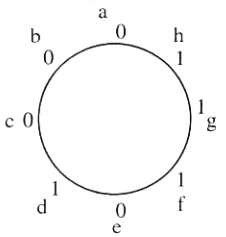
⑤ a[i1]>a[k]

第六节　2000 年试题

一、初中组、高中组第 1 题

【问题描述】

将 2^n 个 0 和 2^n 个 1,排成一圈。从任一个位置开始,每次按逆时针的方向以长度为 n+1 的单位进行数二进制数。要求给出一种排法,用上面的方法产生出来的 2^{n+1} 个二进制数都不相同。当 n=2 时,即有 2^2 个 0 和 2^2 个 1 排列如右:



比如,从 a 位置开始,逆时针方向取三个数 000,然后再从 b 位置上开始取三个数 001,接着取 010……可得到 000, 001, 010, 101, 011, 111, 110 共 8 个二进制数,且都不相同。

【程序说明】

以 n=4 为例,即有 16 个 0、16 个 1,数组 a 用以记录 32 个 0、1 的排法,数组 b 统计二进制数是否已出现过。

【程序清单】

```
begin
  for i:=1 to 36 do a[i]:=0;
  for i:=28 to 32 do a[i]:=1;
  p:=1; a[6]:=1;
  while(p=1)do
    begin
      j:=27;
```

```
while a[j] = 1 do j: = j - 1;
        ①
      for i: = j + 1 to 27 do    ②
      for i: = 0 to 31 do b[i]: = 0;
      for i: = 1 to 32 do
        begin
          ③
          for k: = i to i + 4 do s: = s * 2 + a[k];
          ④
        end;
      s: = 0;
      for i: = 0 to 31 do s: = s + b[i];
        if    ⑤    then p: = 0;
  end;
  for i: = 1 to 32 do
  for j: = i to i + 4 do
  write(a[j]);
  end.
```

【问题分析】

本题采用穷举法输出所有符合条件的二进制数。具体算法如下：

（1）数据结构的处理：用数组 a 的每一位存放 0 和 1 的数（n = 4），有 16 个 0 和 16 个 1，由于数据是环行表示，所以最后四位数要分别和数据列的前四位组成新的二进制数，因此数组 a 需要定义为：a[1..36]of 0..1，b 数组用于判断所产生的二进制数是否重复（判重）；

（2）为了减少循环次数，算法中将后五位全部置 1，前五位全部置 0；

（3）用二进制加法运算规则，从第 27 位开始向前按位搜索，如果为 1 则继续向前，否则将当前位置为 1，从当前位置的后一位到第 27 位置全置为 0（二进制加法运算）即：

```
while a[j] = 1 do j: = j - 1;      (a[j]: = 1);
      ①
for i: = j + 1 to 27 do  ②         (a[i]: = 0);
```

（4）判断新产生的二进制数与已有的二进制数是否重复：

用双重循环完成：扫描 32 位二进制数，顺序将每五位二进制数转换成十进制数，并且判断该数是否存在（即判重）。

```
for i: = 1 to 32 do
begin                                  在③处填写 s: = 0;
      ③                                将五位二进制数转换成
  for k: = i to i + 4 do s: = s * 2 + a[k];   十进制数判断重复，所以
  ④                                    ④处填写 b[s]: = 1
```

```
end;
```

这题算法的巧妙之处在于:判重的方法,s 是新产生的数,将该数作为数组下标,并且将此下标变量置为 1,相同数只取一个。在③处填写 s:=0 是初始化,在④处填写 b[s]:=1。

(5) 判断是否生成 2^{n+1} 个不同的二进制数,即 32 个二进制数,如果没有达到 32 个则重复做(3)、(4)、(5)过程,否则置 p:=0,退出循环。

```
s:=0;
  for i:=0 to 31 do s:=s+b[i];
  if    ⑤    then p:=0;
```

统计有多少个不同二进制数所在

⑤填写 s=32

(6) 输出 2^{n+1} 个不同的二进制数(即 32 个二进制数)。

```
for i:=1 to 32 do
  for j:=i to i+4 do
    write(a[j]);
```

在完成这类练习中,读者首先需要理解题意,然后分块读取程序,了解每个程序块及变量的作用,最后再填写空白处。在填写中,可以自上而下的填写,也可以根据题意从程序的输出结果向前填写。注意程序中变量的作用以及相互间的关系。

二、初中组第 2 题

【问题描述】

多项式乘法运算:$p(x)=2x^2-x+1$, $q(x)=x+1$

$$p(x)*q(x)=(2x^2-x+1)(x+1)=2x^3+x^2+1$$

【程序说明】

多项式的表示:系数、指数

如上例中:

p(x) 系数	指数	q(x) 系数	指数
2	2	1	1
-1	1	1	0
1	0	0	0
0	0		

p*q 的结果存入 c 中。其输出格式是:依次用一对括号内的(系数、指数)分别来表示。如上例的输出结果表示为:(2,3)(1,2)(1,0)

【程序清单】

```
begin
jp:=0;
readln(x,y);
while x<>0 do
begin
  jp:=jp+1;  p[jp,1]:=x;
  p[jp,2]:=y;readln(x,y);
end;
```

输入 p 多项式各项的系数和指数;

```
jq:=0;
  readln(x, y);
    while x<>0 do
      begin
        jq:=jq+1;  q[jq, 1]:=x;
        q[jq, 2]:=y;  readln(x, y);
        end;
```

输入 q 多项式中的各项

```
jc:=1;c[jc, 1]:=0;  c[jc, 2]:=-1000;  {c 数组存放结果,初始化}
  for i:=1 to jp do
   begin
    ____①____
```

将第一个多项式中的一项去乘以另一个多项式的各项。基本规则:读入项的系数→x 中,指数→y 所以①填入 x:=p[i, 1];

```
  y:=p[i, 2];
    for j:=1 to jq do
      begin
      ____②____
```

将第二个多项式的各项,依次与第一个多项式当前项相乘,因此需要将系数乘,指数相加。x1、y1 分别表示乘积后多项式的系数和指数,所以②处填写 x1:=x*q[j, 1];

```
    y1:=y+q[j, 2];
    k:=1;
    while y1<c[k, 2] do k:=k+1;
    if y1=c[k, 2] then ____③____
    else
  begin
```

将当前乘积后多项式按指数的降幂插入到相应位置。当多项式指数与 c 数组中某项相同时,指数不变,系数相加。所以③处填写:c[k, 1]:=c[k, 1]+x1;否则,没有相同指数项

```
  for l:=jc downto k do
  begin
  c[l+1, 1]:=c[l, 1];
  c[l+1, 2]:=c[l, 2];
  end;
  c[k, 1]:=x1;  c[k, 2]:=y1;
  ____④____
  end;
  end;
end;
```

则插入恰当位置,这就需要把插入位置之后的各项向后移动,因此 c 多项式的项数又增加一项,所以④填写:jc:=jc+1;

```
for i:=1 to jc do
  if ____⑤____ then
    write('(',c[i, 1],',',c[i, 2],')');
```

输出多项式各项,将系数为零的项去掉,所以⑤处填写:c[i, i]<>0

【问题分析】

多项式乘法运算的数学规则并不复杂,主要步骤如下:

(1)将一个多项式中的每一项去乘以另一个多项式的各项;

(2)将各项求代数和。

程序设计中需要解决问题的关键:

(1)如何表示多项式中的每一项:每项的系数、x 的指数。算法提示中已经介绍用二维数组存储结构:行坐标表示多项式的项数,列坐标的第一列作为每项的系数,第二列作为 x 的指数,这样的存储结构即节省存储空间,又利于多项式的运算;

p 数组存储第一个多项式各项,q 数组存储第二个多项式的各项,c 数组存储乘积后的多项式各项;

(2)多项式乘法运算:

将第一个多项式的一项去乘以第二个多项式每一项:系数相乘,指数相加;

可以用双重循环结构完成;

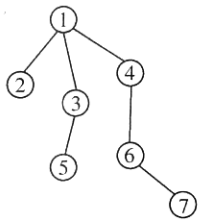
(3)多项式同类项的合并:

数学中多项式排列一般按降幂排列,合并同类项的实质是将 x 指数相同的项,系数相加,指数不同的项依次降幂排列。在此题完善程序中,它是利用插入排序的方法,将多项式的各项存储在 c 数组中;

(4)输出运算结果:用循环语句完成输出乘法运算后,多项式的结果。

由此,我们可以填空原程序,实现运算功能。此问题的数学运算是多项式乘法,而计算机的程序算法是属于基本的运算:线性表的建立、排序与归并。类似这样的应用问题很多。

三、高中组第 2 题



【问题描述】

求一棵树的深度和宽度。例如有如右的一棵树:树的深度为从根结点开始到叶结点结束的最大深度,树的宽度为同一层上结点数的最大值。在右图中树的深度为 4,宽度为 3。

用邻接表来表示树,见下表:

1	2	3	4	0	0
2	0	0	0	0	0
3	5	0	0	0	0
4	6	0	0	0	0
5	0	0	0	0	0
6	7	0	0	0	0
7	0	0	0	0	0

【程序清单】

```
bgein
for i:=1 to 14 do for j:=1 to 6 do tree[i, j]:=0;
for j:=1 to 14 do tree[j, 1]:=j;
  tree[1, 2]:=2;  tree[1, 3]:=3;  tree[1, 4]:=4;  tree[2, 2]:=5;
  tree[2, 3]:=6;  tree[3, 2]:=7;  tree[3, 3]:=8;  tree[4, 2]:=9;
```

```
tree[4, 3]: =10;  tree[4, 4]: =11;  tree[7, 2]: =12;  tree[7, 3]: =13;
tree[13, 2]: =14;
 sp1: =1; sp2: =1;
 for i: =1 to 6 do q[1, i]: =tree[1, 1];
 q[1,0]: =1;
 while ____①____ do
 begin
  l: = ____②____; j: =2;
  while ____③____ do
    begin
     sp2: =sp2 +1;  q[sp2, 0]: =l;  q[sp2, 1]: =q[sp1, j];
     for i: =2 to 6 do
      q[sp2,i]: =tree[q[sp1, j], i];
     j: =j +1;
    end;
     sp1: =sp1 +1;
    end;
    writeln ____④____
  for i: =0 to 20 do d[i]: =0;
  for i: =1 to sp2 do
    d[q[i, 0]]: = ____⑤____
    max: =d[1];
    for i: =2 to 20 do
     if d[i] >max then max: =d[i];
    readln;
  end.
```

【问题分析】

该题主要考核高中学生对“树”这样的数据结构，它的存储结构的邻接表表示法的理解和掌握情况以及用队列的方式表示树的深度搜索算法，用循环计算方法求解树的宽度。它的基本算法如下：

（1）设置数据的存储结构：用数组 tree 表示树的存储结构→邻接表（假设树的度为 4，树的度的定义：树中某个结点的最大子树数或后继结点数）；程序中的树设定 14 个结点，每个结点最多 4 个孩子，第 5 个孩子为 0，用于判断某层的结束；

（2）用 q 数组表示队列，其中 sp1 表示出队的指针，sp2 表示入队的指针，q[i, 0]表示层数；

（3）d 数组统计同一层上的结点数，以便求得宽度；

（4）初始化：tree[i,j]: =0；将树中的 14 个结点赋初值：

```
for j: =1 to 14 do tree[j, 1]: =j;
```

(5) 将树的结构用二维数组邻接表形式存储,其树的结构如上图:

```
tree[1, 2]: =2;  tree[1, 3]: =3;
tree[1, 4]: =4;  tree[2, 2]: =5;    tree[2, 3]: =6;
tree[3, 2]: =7;
tree[3, 3]: =8;  tree[4, 2]: =9;    tree[4, 3]: =10;
tree[4, 4]: =11; tree[7, 2]: =12;   tree[7, 3]: =13;
tree[13, 2]: =14;
```

其邻接表,读者可以仿照前面的问题提示部分画出存储邻接表的结构;

(6) 将根结点及其他的后继结点——孩子结点入队列,队列指针指向队头:sp1: =1; sp2: =1;并且记录指针指向树的第一层:q[1, 0]: =1; {用 q[i, 0]表示层数};

(7) 用双重循环实现"树的深度搜索":

(a) 取出一个结点,层次加 1 即填空②中 1: =q[sp1, 0] +1,如果后继结点不为零,则将后继结点进队即填空③为 q[sp1, j] < >0;

(b) 进队操作:

```
sp2: =sp2 +1;  q[sp2, 0]: =1;  q[sp2, 1]: =q[sp1, j];
for i: =2 to 6 do
  q[sp2, i]: =tree[q[sp1, j], i];
  j: =k +1;
```

队尾的指针加 1,保存层数,将当前同一层次兄弟结点入队

(c) 再取当前队列的首位(出队),sp1: =sp1 +1,重复 a、b、c 过程

(d) 输出深度:④为 writeln(1)或 writeln(q[sp2, 0]);

(e) 重复直到 sp1 > sp2 为止,所以填空①应填写:sp1 < =sp2;

(8) 用循环语句完成树的宽度的搜索:该程序用了一个比较巧妙的算法:

```
for i: =1 to sp2 do      d[q[i, 0]]: = ⑤
```

所以第⑤处填空应为:d[q[i, 0]]: =d[q[i, 0]] +1; {统计同一层次上结点的个数}

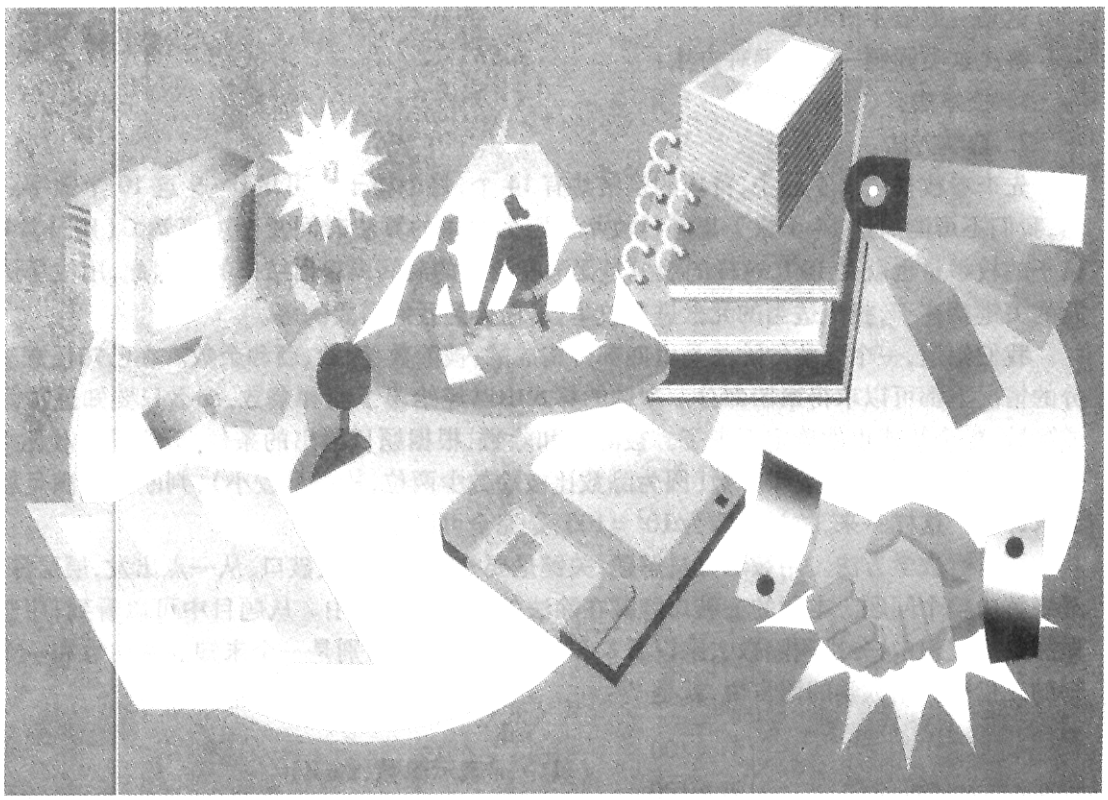
(9) 用循环语句寻找层次中,最多的孩子结点数。

```
for i: =2 to 20 do
if d[i] >max then max: =d[i];
```

至此,完成了整个任务。从分析情况可以看出,问题并不复杂,关键在于掌握基本的算法。掌握基本知识,再加上灵活性,信息学(计算机)初赛的问题还是比较容易解决的。

提　高　篇

- 1995 年复赛试题分析
- 1996 年复赛试题分析
- 1997 年复赛试题分析
- 1998 年复赛试题分析
- 1999 年复赛试题分析
- 2000 年复赛试题分析

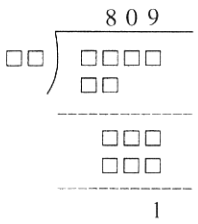


第一章　1995年复赛试题解析

一、初中组第1题

【问题描述】

设有下列的算式：



求出□中的数字，并打印出完整的算式来。

【问题分析】

这是一道数字游戏题。

解决这类问题一般有两种方法：

1. 枚举法；
2. 数学方法。

先来看枚举法。本题中，待填的空格共有14个，每个格子中可填0…9这10个数字。显然，我们不可能对这14个格子都进行枚举。因为枚举的算法时间复杂度高达$O(10^n)$，14个格子的枚举量也就是10^{14}，这样的规模是无法在要求的时间内得出结果的。因此，用枚举法解题的关键就在于，找出适当的元素进行枚举，从而得到我们需要的解。

我们知道，一个完整的除法是由四部分构成的：被除数、除数、商和余数。在已知任意三部分的情况下都可以求得第四部分。本题的算式中已经给出了商和余数，那么只要知道被除数或除数，整个算式也就确定下来了。被除数和除数，根据题目给出的条件，分别是4位数和2位数。所以，我们只需枚举除数（因为除数比被除数少两位，枚举量较小），判断是否满足题目要求即可。这样一来，枚举量降为$10^2=100$，是完全可以承受的。

再看看数学方法。用数学方法解题，关键是要找到问题的突破口：从一点出发，层层深入，最终解决整个问题。本题的突破口隐藏在除法的计算过程之中。从题目中可以看到，用竖式做除法的过程中，分别用除数去乘以了8和9，得到的结果分别是一个未知的两位数和一个未知的三位数。用数学的语言说，就是

$$\begin{cases}8x<100\\9x\geqslant 100\end{cases}$$　（其中，x表示除数，$x\in Z$）

解这个方程，得 x = 12

而$\frac{被除数}{12}$ = 809…1，所以，被除数 = 9709。

这样，很容易得出结果：

```
      809
   ┌──────
12 │ 9709
     96
   ──────
      109
      108
   ──────
        1
```

【数据结构】

1. 枚举法：

```
var x,                          {除数}
y:integer;                      {被除数}
```

2. 数学方法：

无须定义数据结构，直接输出算式即可。

【算法分析】

由于数学方法仅只是将算式输出，因此这里就不予讨论了。这里将主要考虑枚举法的实现过程：

<table>
<tr><td colspan="3">x←10~99（枚举除数 X）</td></tr>
<tr><td rowspan="3">判断当前的除数
X 是否满足要求</td><td rowspan="2">是</td><td>输出结果</td></tr>
<tr><td>程序终止</td></tr>
<tr><td>否</td><td>继续枚举</td></tr>
</table>

判断除数 X 是否满足要求的过程：

<table>
<tr><td colspan="5">y←809*x+1　（由除数 X 计算的被除数 Y）</td></tr>
<tr><td rowspan="2">判断 x 是否为
四位数</td><td rowspan="2">是</td><td rowspan="2">判断 8*x 是否为两位数
以及 9*x 是否为三位数</td><td>是</td><td>返回 true</td></tr>
<tr><td>否</td><td>返回 false</td></tr>
<tr><td>否</td><td colspan="3">返回 false</td></tr>
</table>

【程序清单】

1. 枚举法：

```
program c1995_1a;                          {枚举法}

  var x,                                   {除数}
      y:integer;                           {被除数}
  function right:boolean;                  {判断除数 x 是否满足要求}
   begin
     if (y>999) and (y<10000) and (8*x<100) and (9*x>99)
       then right:=true
       else right:=false
   end;
  procedure print;                         {输出}
   begin
   writeln('       809 ');
     writeln('      -------');
     writeln(x,' ) ',y);
     writeln('       ',8*x);
     writeln('      -------');
     writeln('         ',y-800*x);
     writeln('         ',9*x);
     writeln('      -------');
     writeln('           1 ')
   end;
  begin
   for x:=10 to 99 do                      {枚举除数 x}
       begin
          y:=809*x+1;                      {根据除数 x 计算得被除数 y}
          if y>9999 then break;            {如果 y 超过 4 位数,则退出}
          if right then print              {如果除数 x 满足要求,则输出}
       end
  end.
```

2. 数学方法：

```
program C1995_1B;                                    {数学方法}
begin
  writeln('       809 ');
  writeln('     - - - - - - -');
  writeln('12 ) 9709 ');
  writeln('      96    ');
  writeln('     - - - - - - -');
  writeln('       109 ');
  writeln('       108 ');
  writeln('     - - - - - - -');
  writeln('         1 ')
end.
```

BASIC 语言参考程序：

```
REM C1995_1B
FOR X = 10 TO 99
 Y = 809 * X + 1
 IF Y > 9999 THEN END
 IF (Y > 999) AND (Y < 10000) AND(8 * X < 100) AND (9 × X > 99)  THEN - GOSUB 100
 NEXT X
END
100 : REN PRINT
PRINT"          809"
PRINT"    - - - - - - - - -"
PRINTX;")";Y
PRINT"      ";8 * X
PRINT  "     - - - - - - - - -"
PRINT  "       ";Y - 800 * X
PRINT  "       ";9 * X
PRINT  "     - - - - - - - -"
PRINT  "           1"
RETURN
```

【运行结果】

```
   809
12|9709
  96
  ---
  109
  108
  ---
    1
```

【小结】

本题我们采用了两种不同的方法,都得到了圆满的解决。两种方法在效率上虽然有差别,但有一个共同点,就是抓住了问题的关键。枚举法的关键,在于枚举什么?枚举不同的对象,在效率上是千差万别的。而数学方法的关键,是找出问题的突破口。只有从一点突破,才能不断深入,彻底地解决问题。

二、初中组第2题

【问题描述】

方阵填数:在一个N×N的方阵中,填入1,2,…,N×N个数,并要求构成如下的格式:

例:

N=5

13	14	15	16	1
12	23	24	17	2
11	22	25	18	3
10	21	20	19	4
9	8	7	6	5

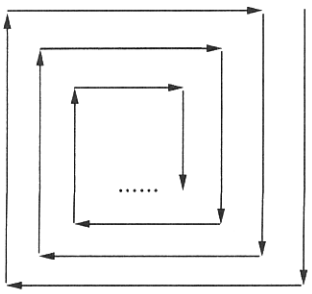
N=6

16	17	18	19	20	1
15	30	31	32	21	2
14	29	36	33	22	3
13	28	35	34	23	4
12	27	26	25	24	5
11	10	9	8	7	6

【问题分析】

这是一个经典的螺旋形方阵填数问题。解决这样的问题,关键在于从填数的规则中找到便于我们解题的规律。

如下图,表示的是将1到N×N依次填入方阵所经过一条螺旋形的路径,箭头方向表示填数的方向。

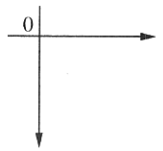


从中不难发现，填数的方向依次为：↓、←、↑、→；↓、←、↑、→……如此循环往复，直至所有的数都填完。这是一条好规律，因为这条规律所体现的是循环的结构，循环结构在计算机中是很容易实现的。我们只需循环按四个方向将 1 到 N * N 这 N * N 个数依次填入方阵即可。

【数据结构】

```
const maxn = 10;                                          {最大规模为 n = 10}
    di:array [0..3, 0..1] of integer
      =((1, 0),(0, -1),(-1, 0),(0, 1));                  {定义↓、←、↑、→四个方向}
type square = array [0..maxn+1,0..maxn+1] of word;        {定义方阵类型}
var sq:square;                                            {保存方阵中的数}
    n:word;                                               {方阵边长}
```

【算法分析】



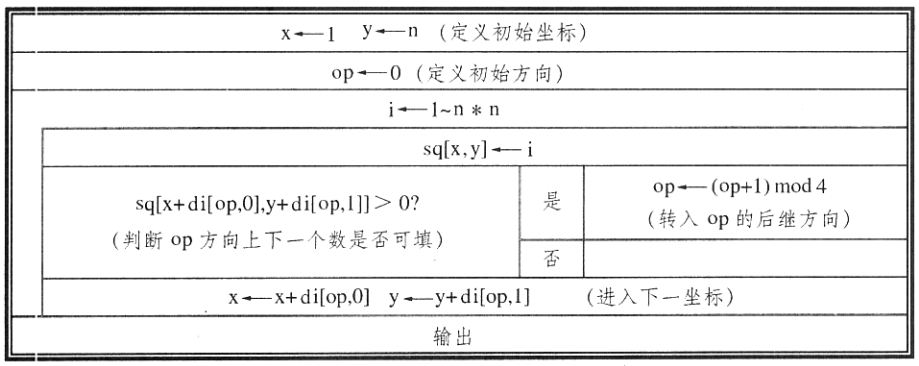
我们定义四个方向 direction[0] ~ direction[3]，依次表示“↓、←、↑、→”四个方向，如果将其数字化的话。就是

direction:array [0..3, 0..1] of integer = ((1, 0), (0, -1), (-1, 0), (0, 1))；其中，direction[t, 0]、direction[t, 1]分别表示 direction[t]方向上纵坐标与横坐标的增减情况。而我们的坐标按右图的定义。

初始坐标定为(1, n)，初始方向为 0 方向(即“↓”方向)。不断在此方向上填数，直至该方向上无法再填数(也就是遇上方阵的边界或已填过数的坐标)，则转入下一方向(当前方向的后继方向)，进行同样的操作。直至所有数都填完，那么整个螺旋形方阵也就填完了，只需输出即可。

注：一个方向的后继方向按如下定义：若方向 i1 和 i2 满足 i1 = i2 - 1，我们就说方向 i2 是方向 i1 的后继方向(令 0 - 1 = 3，即方向 0 是方向 3 的后继方向)。

算法流程如下：



【程序清单】

```
{$A+,B-,D-,E-,F-,G+,I-,L-,N+,O-,P-,Q-,R-,S-,T-,
V-,X-,Y-}
{$M 65520,0,655360}
program c1995_2;
const maxn=10;                                          {最大规模为 n=10}
      di:array[0..3,0..1] of integer
         =((1,0),(0,-1),(-1,0),(0,1));                  {定义↓、←、↑、→四个方向}
type square=array[0..maxn+1,0..maxn+1] of word;         {定义方阵类型}
var sq:square;                                          {保存方阵中的数}
    n:word;                                             {方阵边长}
procedure create;                                       {初始化过程}
var i:word;
 begin
   fillchar(sq,sizeof(sq),0);                           {清零}
   for i:=1 to n do
     begin
       sq[0,i]:=maxint; sq[n+1,i]:=maxint;
       sq[i,0]:=maxint; sq[i,n+1]:=maxint
     end                                                {给方阵边界赋值}
 end;
procedure solve;                                        {填数过程}
 var i,x,y,op:word;
 begin
   x:=1; y:=n;                                          {设置初始坐标}
   op:=0;                                               {设置初始方向}
   for i:=1 to n*n do
     begin
       sq[x,y]:=i;                                      {在方阵中填数}
       if sq[x+di[op,0],y+di[op,1]]>0                   {判断当前方向是否还能填数}
         then op:=(op+1) mod 4;                         {转入下一方向}
       inc(x,di[op,0]); inc(y,di[op,1])                 {转入下一坐标}
     end;
 end;

procedure print;                                        {输出过程}
 var i,j:word;
```

```
  begin
    for i:=1 to n do
      begin
        for j:=1 to n do write(sq[i,j]:4);
        writeln
      end
  end;

begin
  readln(n);                                    {读入 n}
  create;                                       {初始化}
  solve;                                        {填数}
  print                                         {输出}
end.
```

BASIC 语言参考程序：

```
REM C1995_2B
INPUT"N=";N
DIM DI(4,2),SQ(0:N+1,0:N+1)
DI(1,1)=1:DI(1,2)=0
DI(2,1)=0:DI(2,2)=-1
DI(3,1)=-1:DI(3,2)=0
DI(4,1)=0:DI(4,2)=1
FOR I=0 TO N+1
FOR J=0 TO N+1
SQ(I,J)=0
NEXT J,I
FOR I=1 TO N
SQ(0,I)=1
SQ(N+1,I)=1
SQ(I,0)=1
SQ(I,N+1)=1
NEXT I
GOSUB SOL
GOSUB PRT
END
SOL:REM SOLVE
```

```
X = 1 : Y = N
OP = 1
FOR I = 1 TO N * N
 SQ(X, Y) = I
 IF SQ(X + DI(OP, 1), Y + DI(OP, 2)) > 0 THEN OP = OP MOD 4 + 1
 X = X + DI(OP, 1) : Y = Y + DI(OP, 2)
NEXT I
RETURN
PRT : REM PRINT
 FOR I = 1 TO N
   FOR J = 1 TO N
   PRINT TAB(4 * J - 3); SQ(I, J);
   NEXT J
   PRINT
 NEXT I
RETURN
```

【运行结果】

输入:n = 3

输出:

```
7  8  1
6  9  2
5  4  3
```

输入:n = 5

输出:

```
13  14  15  16   1
12  23  24  17   2
11  22  25  18   3
10  21  20  19   4
 9   8   7   6   5
```

输入:n = 10

输出:

```
28  29  30  31  32  33  34  35  36   1
27  58  59  60  61  62  63  64  37   2
26  57  80  81  82  83  84  65  38   3
25  56  79  94  95  96  85  66  39   4
24  55  78  93 100  97  86  67  40   5
23  54  77  92  99  98  87  68  41   6
```

22	53	76	91	90	89	88	69	42	7
21	52	75	74	73	72	71	70	43	8
20	51	50	49	48	47	46	45	44	9
19	18	17	16	15	14	13	12	11	10

【小结】

本题的难度不是很高，解决的方法也很多。不过，诸如螺旋形方阵的变形也很多，如：蛇形方阵、三角螺旋形方阵等。解决这类问题，往往要从填数的规则中，找出适合于我们编程实现的规律——例如本题的循环规律着手。

三、初中组第 3 题

【问题描述】

若将一个正整数化为二进制数，在此二进制数中，我们将数字 1 的个数多于数字 0 的个数的这类二进制数称为 A 类数，否则就称其为 B 类数。

例如：$(13)_{10}=(1101)_2$

其中 1 的个数为 3，0 的个数为 1，则称此数为 A 类数；

$(10)_{10}=(1010)_2$

其中 1 的个数为 2，0 的个数也为 2，称此数为 B 类数；

$(24)_{10}=(11000)_2$

其中 1 的个数为 2，0 的个数为 3，则称此数为 B 类数；

程序要求：求出 1～1000 之中（包括 1 与 1000），全部 A、B 两类数的个数。

【问题分析】

这是一道统计题。统计题往往有两种解法：第一种方法，就是逐一枚举情况，并且统计结果；第二种方法，是利用问题本身所包含的数学规律，利用数学方法解决问题。

我们分析一下这两种方法的可行性。先看枚举法。枚举法的正确性和可实现性是不必怀疑的，关键是它的时间效率是否能满足我们的要求。由于本题的数据规模仅为 1000，而逐一枚举，逐一统计的复杂度为 $O(n\log_2 n)$，因此枚举法在时间方面是可以承受的。

再来分析一下问题的数学规律。我们先来看一个简单而特殊的例子：在数集$[(10000)_2,(111111)_2]$中，有多少个 A 类数？根据定义，若其中的一个数$(1X_1X_2X_3X_4X_5)_2$是 A 类数，则$X_1X_2X_3X_4X_5$中至少应有三个数字为“1”，也就是说可能有 3 个、4 个或 5 个“1”，而且这些“1”是自由地分布在$X_1X_2X_3X_4X_5$中的。我们知道，一个长度为 m、包含 n 个“1”的 01 串共有 C_m^n 种。因此，形如$(1X_1X_2X_3X_4X_5)_2$的 A 类数就有$(C_5^3+C_5^4+C_5^5)$个。

进一步，稍微一般化的问题：数集$[(X_1X_2X_3\cdots X_n000\cdots0)_2,(X_1X_2X_3\cdots X_n111\cdots1)_2]$中有多少个 A 类数？假设该数集中的数均为二进制 m 位数，$X_1X_2X_3X_4X_5$中存在 k 个“1”，那么，后 m－n 位中应有至少$[m/2]+1-k$个“1”。所以数集$[(X_1X_2X_3\cdots X_n000\cdots0)_2,(X_1X_2X_3\cdots X_n111\cdots1)_2]$中的 A 类数个数为 $\sum_{i=[\frac{m}{2}]+1-k}^{m-n} C_{m-n}^i$。

这样,我们就解决了数集$[(X_1X_2X_3\cdots X_n000\cdots0)_2,(X_1X_2X_3\cdots X_n111\cdots1)_2]$中A类数个数问题。现在的关键就是要将1到n这些数划分成若干个上述的数集,分别求其中A类数的个数。

我们来看这样一个二进制数:$(1010)_2$。统计$(1)_2$到$(1010)_2$中的A类数个数:

首先看小于$(1000)_2$的部分,即从$(1)_2$到$(111)_2$的部分。我们可以将其划分为三部分

$(1)_2$

$(10)_2\sim(11)_2$

$(100)_2\sim(111)_2$

这三个部分两个符合"$[(X_1X_2X_3\cdots X_n000\cdots0)_2,(X_1X_2X_3\cdots X_n111\cdots1)_2]$"的格式,另一个只有一个数$(1)_2$,是A类数。完全可以用上面的公式算出其中A类数的个数。

再看大于等于$(1000)_2$的部分。我们将其分为两个部分

$(1000)_2\sim(1001)_2$

$(1010)_2$

同样的,一个部分符合"$[(X_1X_2X_3\cdots X_n000\cdots0)_2,(X_1X_2X_3\cdots X_n111\cdots1)_2]$"的格式,另一个只有一个数$(1010)_2$,也可以用上面的公式算出其中A类数的个数。

那么,将问题推广一些:求$(1)_2$到$(1000\cdots01000\cdots1000\cdots0100\cdots\cdots)_2$中A类数的个数。

设$N=(1000\cdots01000\cdots1000\cdots0100\cdots\cdots)_2$,N共p位,即$p=[\log_2 n]$。

首先求1到2^p-1中的A类数的个数。将这些数划分成p部分

$(1)_2$

$(10)_2\sim(11)_2$

$(100)_2\sim(111)_2$

⋮

$2^{p-1}\sim2^p-1$

这p部分的情况于前面的相似,因此也可以用公式解决。

再求2^p到N中的A类数的个数。这部分的划分就比较麻烦了,我们对N的每一个二进制位进行扫描,最高位除外。若扫描到"1",如$(X_1X_2X_3\cdots X_n1Y_1Y_2Y_3\cdots Y_n)_2$,则将$(X_1X_2X_3\cdots X_n0000\cdots0)_2\sim(X_1X_2X_3\cdots X_n1111\cdots1)_2$作为一部分划分出来,进行计算。最后再判断N本身是否为A类数,若是,则在总数上再加1。

这个算法的复杂度是基于N的二进制位长度的,复杂度约为$O(\log^3 n)$。

【数据结构】

1. 枚举法:

```
type settype = set of 0..15;

var i,                          {枚举变量}
    a,                          {记录A类数个数}
    b:word;                     {记录B类数个数}
    s:settype absolute i;       {集合对应变量i,
                                 便于统计i中1的个数}
```

2. 数学方法:

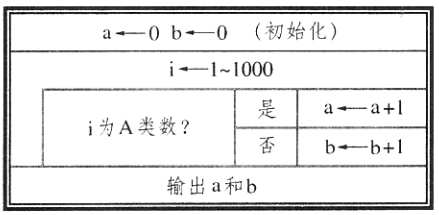
```
const maxlen = 100;                                    {允许最多 100 位二进制}
  prime:array [1..25] of byte
    =(2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71,
73, 79, 83, 89, 97);                                   {100 以内的素数表}

type highnumber = array [0..maxlen] of shortint;       {定义高精度数类型}
     linkhigh = ^highnumber;                           {指针型高精度数(便于操作)}

var hn,                                                {高精度数 n}
    bn,                                                {n 的二进制表示}
    a,                                                 {记录 A 类数个数}
    b:highnumber;                                      {记录 B 类数个数}
```

【算法分析】

1. 枚举法：

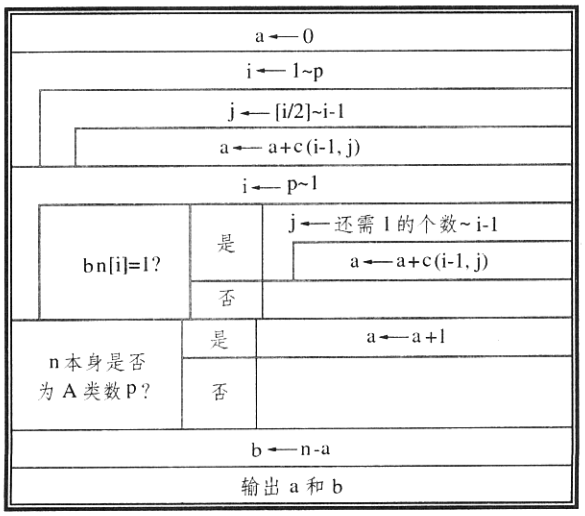
枚举法的算法比较浅显，就是枚举 1 到 1000 中的每一个数，判断其是否为 A 类数。是，则将 A 类数个数加 1；否则，将 B 类数个数加 1。

算法流程如下：



2. 数学方法：

数学方法解题过程刚才已经讲得很详细了，这里给出算法流程：



【程序清单】

1. 枚举法:

```
{$ A+, B-, D-, E-, F-, G+, I-, L-, N+, O-, P-, Q-, R-, S-, T-,
V-, X-, Y-}
{$ M 65520, 0, 655360}
program c1995_3a;                                    {枚举法程序}

const n=1000;                                        {数据规模:n=1000}

type settype=set of      0..15;

var i,                                               {枚举变量}
a,                                                   {记录 A 类数个数}
b:word;                                              {记录 B 类数个数}
s:settype absolute i;                                {集合对应变量 i,
                                                     便于统计 i 中 1 的个数}
function judgea:boolean;                             {判断 i 是否为 A 类数}
 var i, j,                                           {循环变量}
     t:word;                                         {记录 1 的个数}
 begin
   j:=16; repeat dec(j) until j in s;                {判断 i 的二进制长度}
   t:=0;
   for i:=j downto 0 do
     if i in s then inc(t);                          {记录 i 中 1 的个数}
   if t>(j+1) div 2                                  {判断 1 的个数是否较多}
     then judgea:=true
     else judgea:=false
 end;

begin
  a:=0; b:=0;                                        {初始化}
  for i:=1 to n do                                   {枚举 i}
if judgea                                            {判断 i 是否为 A 类数}
  then inc(a)
  else inc(b);
  writeln(a, ' ', b)                                 {输出}
end.
```

2. 数学方法：

```
{$ +, B -, D -, E -, F -, G +, I -, L -, N +, O -, P -, Q -, R -, S -, T -,
V -, X -, Y -}
{$ M 65520, 0, 655360}
program c1995_3b;                                    {数学方法程序}

const maxlen = 100;                                  {允许最多 100 位二进制}
  prime: array [1..25] of byte
    = (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);                                 {100 以内的素数表}

type highnumber = array [0..maxlen] of shortint;
                                                     {定义高精度数类型}
       linkhigh = ^highnumber;                       {指针型高精度数(便于操作)}

var hn,                                              {高精度数 n}
    bn,                                              {n 的二进制表示}
    a,                                               {记录 A 类数个数}
    b: highnumber;                                   {记录 B 类数个数}
function max(a, b: shortint): shortint;              {取 a, b 较大者}
 begin
   if a > b then max: = a
          else max: = b
 end;

procedure highinc(var a: highnumber);                {高精度加 1 过程: a: = a + 1}
 var i: word;
 begin
   i: = 1; inc(a[1]);
   while a[i] = 10 do
     begin
       a[i]: = 0;
       inc(a[i + 1]);
     end;
   if i > a[0] then a[0]: = i
 end;
procedure highadd(var a: highnumber; b: highnumber);
                                                     {高精度加法过程: a: = a + b}
```

```
var i:word;
begin
  for i:=1 to max(a[0], b[0])+1 do
    begin
      inc(a[i], b[i]);
      inc(a[i+1], a[i] div 10);
      a[i]:=a[i] mod 10
    end;
  a[0]:=i; while a[a[0]]=0 do dec(a[0]);
end;

procedure highdec(var a:highnumber;b:highnumber);
                                                    {高精度减法过程:a:=a-b}
var i:word;
begin
  for i:=1 to a[0] do
    begin
      dec(a[i], b[i]);
      if a[i]<0
        then begin
          inc(a[i], 10);
          dec(a[i+1])
        end
    end;
  while (a[a[0]]=0) and (a[0]>0) do dec(a[0])
end;

procedure highmul(var a:highnumber;b:shortint);    {高精度乘法过程:a:=a*b}

var i, g:word;
begin
  i:=1; g:=0;
  while not ((i>a[0]) and (g=0)) do
    begin
      inc(g, word(a[i])*b);
      a[i]:=g mod 10;
      g:=g div 10;
      inc(i)
```

```
      end;
    a[0]:=i-1
  end;

procedure highshr(var a:highnumber;var m:shortint);
                                        {高精度右位移过程:a:=a shr 1    m:=a and 1}
  var i:word;
  begin
    m:=0;
    for i:=a[0] downto 1 do
      begin
        m:=m*10+a[i];
        a[i]:=m shr 1;
        m:=m and 1;
      end;
    if a[a[0]]=0 then dec(a[0])
  end;

function highC(m, n:shortint):linkhigh;  {高精度求组合数:c(m, n)}
  var a:linkhigh;
      t:array [1..25] of word;
      i, j:word;

  function factors(n, f:shortint):word;     {求因数个数}
    var t:word;
    begin
t:=0;
while n>0 do
    begin
      n:=n div f;
      inc(t, n);
    end;
factors:=t
    end;

  begin
    for i:=1 to 25 do
      t[i]:=factors(m, prime[i])-factors(n, prime[i])-factors(m-n, prime[i]);
```

```
                                                    {求 c(m, n)中含个素因数个数}
    new(a);
    fillchar(a^, sizeof(a^), 0); a^[0]:=1; a^[1]:=1;
                                                    {初始化}
    for i:=1 to 25 do
      for j:=1 to t[i] do
        highmul(a^, prime[i]);                      {因数相乘求 c(m, n)}
    highc:=a;
  end;

procedure initialize;                               {读数据过程}
  var s:string;
      i, code:word;
  begin
    readln(s); hn[0]:=length(s);
    for i:=1 to hn[0] do
      val(s[hn[0]+1-i], hn[i], code);               {读入高精度数 n}
  end;

procedure create;                                   {将高精度数 n 转化为其二进制
                                                    表示 bn}
  var temp:highnumber;
  begin
    temp:=hn; bn[0]:=0;
    while not (temp[0]=0) do
      begin
        inc(bn[0]);
        highshr(temp, bn[bn[0]])
      end;
  end;

procedure solve;                                    {计算 A 类数个数}
  var i, j:word;
      t:shortint;
  begin
    fillchar(a, sizeof(a), 0);                      {初始化}
    for i:=1 to bn[0]-1 do
      for j:=i div 2 to i-1 do
```

```
        highadd(a, highc(i-1, j)^);                {计算 1 到 2^p - 1 间的 A 类数个数}
    t:=bn[0] div 2;
    for i:=bn[0]-1 downto 1 do
      if bn[i]=1
        then begin
          for j:=max(t, 0) to i-1 do
                highadd(a, highc(i-1, j)^);
          dec(t)
        end;                                       {计算 2^p 到 n-1 之间的 A 类数个数}
    if t<=0                                        {判断 n 本身是否为 A 类数}
      then highinc(a);
    b:=hn; highdec(b, a)                           {计算 B 类数个数}
  end;

procedure print(a:highnumber);                     {输出高精度数}
  var i:word;
  begin
    if a[0]=0
      then write(0)
      else for i:=a[0] downto 1 do write(a[i]);
    write(' ')
  end;

begin
  initialize;                                      {读数据}
  create;                                          {初始化}
  solve;                                           {计算 A、B 类数个数}
  print(a); print(b);   writeln                    {输出}
end.
```

BASIC 语言参考程序:

```
REM C1995_3B
A = 0
FOR I = 1 TO 1000
N = I
L = 0
T = 0
DO WHILE N > 0
 IF N MOD 2 = 1 THEN T = T + 1
 L = L + 1
 N = N\2
LOOP
IF T > L\2 THEN A = A + 1
NEXT I
B = 1000 - A
PRINT  "A = ";A
PRINT  "B = ";B
END
```

【运行结果】

A 类数:538

B 类数:462

【小结】

本题的两种方法在时间复杂度上的差异对解题影响并不是很大,但如果我们将问题规模扩大,则第二种方法将体现出明显的优势。后来,CTSC'99 就出了一道类似的题目,规模达 10^{30},必须用数学方法才能解决。

四、初中组第 4 题/高中组第 1 题

【问题描述】

编码问题:设有一个数组 A:ARRAY[0..N-1] OF INTEGER;数组中存放的元素为 0~N-1之间的整数,且 A[i]≠A[j](当 i≠j 时)。

例如:N=6 时,A=(4, 3, 0, 5, 1, 2)

此时,数组 A 的编码定义如下:

A[0]的编码为 0;

A[i]的编码为:在 A[0],A[1],…,A[i-1]中比 A[i]的值小的个数(i=

1,2,…,N－1)

所以上面数组 A 的编码为:B＝(0, 0, 0, 3, 1, 2)

程序要求解决以下问题:

① 给出数组 A 后,求出其编码;

② 给出数组 A 的编码后,求出 A 中的原数据。

【问题分析】

本题要求我们进行编码的转换,并且给出了转换的规则,因此算法就很浅显了,就是完全按照题目给出的规则来做。关键是设计怎样的数据结构,采取什么实现方案,以便于我们处理。而算法复杂度则是次要的,因为本题的规模比较小,各种算法在时空效率上的差别不大。

任务一的要求是统计在在 A[i]前比 A[i]小的数的个数,这不难实现,仅需一个二重循环。

任务二是任务一的一个逆操作,也就是任务一怎样做,它就倒着做。那么,我们就从后往前做,每次从未被取走的数中,找出第 B[i]＋1 大的数,作为 A[i]的值。这也可以用一个二重循环来实现。

【数据结构】

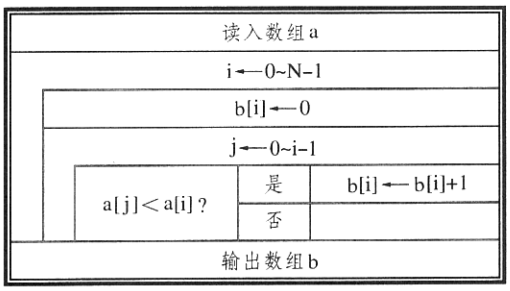
```
const maxn = 10;                                  {最大规模为 n = 10}
type list = array [0..MaxN - 1] of word;          {定义数组 a, b 的类型}
var a,                                            {数组 a}
    b:list;                                       {数组 b}
    n:word;                                       {数组长度 n}
var unget:array [0..MaxN - 1] of boolean;         {存放尚未被取走的数
                                                   true:未被取走
                                                   false:已被取走}
```

【算法分析】

1. 任务一:

任务一比较简单,只需对数组 a 任一元素 a[i] (i＝1, 2,…, n－1),统计在其前面、值比它小的元素个数,记为 w。那么,b[i]＝w。实现这个算法只需一个二重循环即可,时间复杂度是 $O(n^2)$。而空间方面需要开两个一维数组,复杂度是 O(2n)。

算法流程如下:



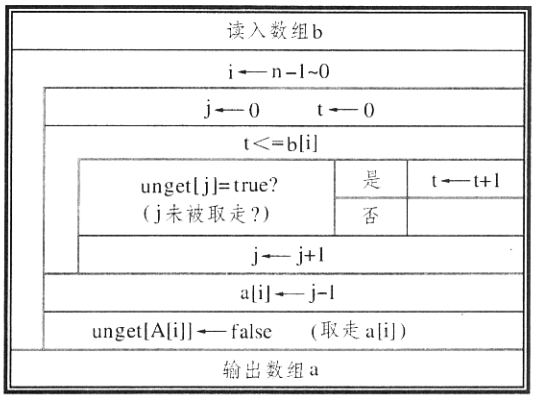
另外,如果再增加一棵排序二叉树保存数组 b 的方案,则对 a 的每个元素统计时,就可以通过二分查找,用一个对数级复杂度的算法继承前面计算的结果,将时间复杂度降为

O(nlogn)。由于这个算法对本题的改进不是很明显,而且会大大增加编程的复杂度,就不予考虑了。不过,这个算法在数据规模较大时,会体现出明显的优势(例如 n 等于几百时)。

2. 任务二:

任务二相对比较麻烦,因为它不可能通过简单的统计得出结果。我们需要另外设一个列表 unget,存放尚未被取走的元素。从 b 数组的最后一个元素向前操作。对于任一元素 b[i](i=1,2…,n-1),若其值为 w,则在 unget 表中找出第 w+1 大的元素 p,使 a[i]=p,并将 p 从 unget 表中去除。这个算法的实现也是一个二重循环,时间复杂度 $O(n^2)$。空间方面,由于要多开一个 unget 表,因此复杂度为 O(3n)

算法流程如下:



当然,任务二也可以用二分查找的方法将时间复杂度降为 O(nlogn)。

【程序清单】

```
program c1995_4;

const maxn = 10;                                    {最大规模为 n = 10}

type list = array [0..maxn - 1] of word;            {定义数组 a, b 的类型}

var a,                                              {数组 a}
  b:list;                                           {数组 b}
  n:word;                                           {数组长度 n}
  s:string;

function unfinished(var c:word):boolean;            {从字符串中得到一个数}
 var i:word;
     code:integer;
 begin
   i:=1;  while (i<=length(s)) and (not (s[i] in ['0'..'9'])) do inc(i);
   if i>length(s)
```

```
        then begin
               unfinished:=false;
               exit
               end;
     delete(s, 1, i-1);
     i:=1; while (i<length(s)) and (s[i] in ['0'..'9']) do inc(i);
     val(copy(s, 1, i-1), c, code); delete(s, 1, i);
     unfinished:=true
  end;

procedure getlist(var l:list);                    {由字符串得到数组}
  var c:word;                                     {存放数组元素}
  begin
     n:=0;                                        {初始化}
     while unfinished(c) do                       {判断是否读完}
        begin
          l[n]:=c;                                {加入一个元素}
          inc(n)
        end
  end;
procedure putlist(l:list);                        {输出数组}
  var i:word;
  begin
     write('(');
     for i:=0 to n-2 do write(l[i], ', ');
     writeln(l[n-1], ')');
  end;

procedure taska;                                  {任务一}

  procedure exchange;                             {转换过程}
    var i, j:integer;
    begin
      for i:=0 to n-1 do
      begin
      b[i]:=0;
      for j:=0 to i-1 do
```

```
        if a[j] < a[i] then inc(b[i])          {统计在 a[i]前比 a[i]小的元素个数}
      end;
    end;

  begin
    getlist(a);                                {读入数组 a}
    exchange;                                  {转换}
    write('b = '); putlist(b)                  {输出数组 b}
  end;

procedure taskb;                               {任务二}

  procedure exchange;
    var unget:array [0..maxn - 1] of boolean;  {存放尚未被取走的数
                                                true:未被取走
                                                false:已被取走}
        i, j,                                  {循环变量}
        t:integer;                             {计数器}
  begin
    fillchar(unget, sizeof(unget), true);      {初始化}
    for i: = n - 1 downto 0 do
      begin
        j: = 0; t: = 0;
        while t < = b[i] do
          begin
        if unget[j] then inc(t);
        inc(j)
      end;                                     {找出未被取走的第 b[i] + 1 大的数}
    a[i]: = j - 1;
    unget[a[i]]: = false                       {将 a[i]取走}
      end
  end;

begin
  getlist(b);                                  {读入数组 b}
  exchange;                                    {转换}
  write('a = '); putlist(a)                    {输出数组 a}
```

```
  end;

begin
   readln(s);                                    {读入数组}
   case s[1] of                                  {判断要执行的任务}
    'a':taska;                                   {执行任务一}
    'b':taskb                                    {执行任务二}
   end
end.
```

BASIC 语言参考程序：

```
REM C1995_4B
MAXN = 10
DIM A(MAXN), B(MAXN), C(MAXN)
print"INPUT A OR B:"
INPUT S $:L = LEN(S $)
CH $ = MID $(S $, 1, 1)
SELECT CASE CH $
CASE"A"
GOSUB TASKA
CASE"B"
GOSUB TASKB
END SELECT
END
TASKA:REM TASK1
 N = -1
 FOR I = 1 TO L
  IF (MID $(S $, I, 1) > ="0") AND (MID $(S $, I, 1) < ="9") THEN
   N = N + 1
   A(N) = VAL(MID $(S $, I, 1))
  END IF
 NEXT I
 B(0) = 0
 FOR I = 1 TO N
  B(I) = 0
  FOR J = 0 TO I - 1
```

```
    IF A(J) < A(I) THEN B(I) = B(I) + 1
   NEXT J
NEXT I
PRINT "B = (";
FOR I = 0 TO N - 1
 PRINT B(I);
NEXT I
PRINT B(N);")"
RETURN
TASKB:REM TASK2
   FOR I = 0 TO MAXN
    C(I) = 1
   NEXT I
N = -1
FOR I = 1 TO L
 IF (MID $(S $, I, 1) > = "0") AND (MID $(S $, I, 1) < = "9") THEN
    N = N + 1
    B(N) = VAL(MID $(S $, I, 1))
  END IF
NEXT I
FOR I = N TO 0 STEP -1
  J = 0:T = 0
  DO WHILE T < = B(I)
  T = T + C(J)
  J = J + 1
  LOOP
  A(I) = J - 1
  C(A(I)) = 0
NEXT I
PRINT "A = (";
FOR I = 0 TO N - 1
PRINT A(I);
NEXT I
PRINT A(N);")"
RETURN
```

【运行结果】

输入:A = (0, 1, 2, 3, 4, 5)

输出:B = (0, 1, 2, 3, 4, 5)

输入:A = (5, 4, 3, 2, 1, 0)
输出:B = (0, 0, 0, 0, 0, 0)

输入:A = (2, 5, 3, 1, 7, 6, 8, 0, 4)
输出:B = (0, 1, 1, 0, 4, 4, 6, 0, 4)

输入:B = (0, 0, 0, 0, 0)
输出:A = (4, 3, 2, 1, 0)

输入:B = (0, 1, 2, 3, 4, 5, 6, 7)
输出:A = (0, 1, 2, 3, 4, 5, 6, 7)

输入:B = (0, 1, 0, 2, 4, 3, 1, 2, 5, 0)
输出:A = (4, 8, 1, 5, 9, 7, 2, 3, 6, 0)

【小结】

本题涉及了排列与整数的转换,不过没有给出具体的转换方法,仅要求我们完成转换的前期工作。如果将排列 a 所对应的编码 b,按一个“递增进制”转换为 10 进制数的话,就实现了排列与整数的一一对应。

五、初中组第 5 题/高中组第 2 题

【问题描述】

灯的排列问题:设在一排上有 N 个格子(N≤20),在格子中放置有不同颜色的灯,每种灯的个数记为 $N_1, N_2, \cdots, N_p$(p 表示不同颜色灯的个数)。

放灯时要遵守下列规则:

① 同一种颜色的灯不能分开;

② 不同颜色的灯之间至少要有一个空位置。

例如:N = 8(格子数)
R = 2(红灯数)
B = 3(蓝灯数)

放置的方法有:

R - B 顺序

R	R		B	B	B		
R	R			B	B	B	
R	R				B	B	B
	R	R		B	B	B	
	R	R			B	B	B
		R	R		B	B	B

B－R 顺序

B	B	B		R	R		
B	B	B			R	R	
B	B	B				R	R
	B	B	B		R	R	
	B	B	B			R	R
		B	B	B		R	R

放置的方式为 12 种。

数据输入的方式为：

N

P1(颜色,为一个字母) N1(灯的数量)

P2 N2

……

Q(结束标记,Q 本身不是灯的颜色)

程序要求：求出一种顺序的排列方案及排列总数。

【问题分析】

本题对彩灯的排列作了两个限制：

同一种颜色的灯不能分开：

这一条件的约束,使得灯的数量失去了实质的意义。因为我们可以将连续的相同颜色的灯就看作一盏灯。因此,我们只需考虑每种颜色一盏灯的情况。

不同颜色的灯之间至少要有一个空位置：

这一限制可以减少灯的放置总数,降低时间复杂度。

既然题目给了我们两个优惠的条件,那么我们就该好好利用它们。分析一下复杂度,一种顺序的放置方案数应该是 C_{k+1}^{p},其中 k 表示不放灯的空格个数,p 表示颜色数。因此,搜索的时间复杂度就是 $O(C_{k+1}^{p})$。这是一个指数级的复杂度,不过由于题目的种种限制,使得 p 和 k 都已很小了。这样,搜索的时间复杂度已经能够承受了。

至于总的排列方案数,则为 $C_{k+1}^{p} \times p!$。因为,颜色数为 p,且各种颜色位置可互换,因此颜

色的排列顺序共 p! 种。而每种排列顺序的放置方案数是 C_{k+1}^{p}，总方案数就是 $C_{k+1}^{p} \times p!$。

【数据结构】

```
const maxn = 20;                                    {最大数据规模 n = 20}
type colorlist = array [1..maxn] of record
                                    color:char;
                                    mount:word
                                  end;              {记录每种颜色的灯的信息}
 line = array [1..maxn + 1] of word;                {记录放置方案}

var n,                                              {格子数}
    p,                                              {颜色数}
    m:word;                                         {空格数}
    tot:longint;                                    {记录总放置方案}
    cl:colorlist;                                   {记录每种颜色的灯的信息}
    l:line;                                         {记录放置方案}
```

【算法分析】

本题的算法围绕搜索进行。首先将相同颜色的灯合并为 1 个灯，按一个颜色顺序搜索。

搜索按格子进行，对于任一格子 l[i]，我们可以枚举它的两种状态——放灯或不放。若放灯，则 l[i+1] 不能放灯，然后枚举 l[i+2]；若不放，则枚举 l[i+1]。

算法流程如下：

<table>
<tr><td rowspan="6">i>m?</td><td rowspan="2">是</td><td rowspan="2">所有颜色都填完?</td><td>是</td><td>输出</td></tr>
<tr><td></td><td>方案数加1</td></tr>
<tr><td></td><td>否</td><td></td></tr>
<tr><td rowspan="3">否</td><td rowspan="2">颜色未填完?</td><td>是</td><td>将当前颜色填入格子 i</td></tr>
<tr><td></td><td>搜索格子 i+2</td></tr>
<tr><td></td><td>否</td><td></td></tr>
<tr><td colspan="3">i 格子不填颜色</td></tr>
<tr><td colspan="3">搜索格子 i+1</td></tr>
<tr><td colspan="5">方案数*p!</td></tr>
</table>

【程序清单】

```
program c1995_5;
const maxn = 20;                                    {最大数据规模 n = 20}
type colorlist = array [1..maxn] of record
                                    color:char;
                                    mount:word
                                  end;              {记录每种颜色的灯的信息}
 line = array [1..maxn + 1] of word;                {记录放置方案}
```

```
var n,                                                    {格子数}
    p,                                                    {颜色数}
    m:word;                                               {空格数}
    tot:longint;                                          {记录总放置方案}
    cl:colorlist;                                         {记录每种颜色的灯的信息}
    l:line;                                               {记录放置方案}

procedure initialize;                                     {读数据过程}
 var x:char;
 begin
   readln(n);                                             {读入格子数 n}
   p:=0; read(x);
   while x<>'q' do
     begin
       inc(p);
       cl[p].color:=x;   readln(cl[p].mount);
       read(x)
     end;                                                 {读入各种颜色的信息}
 end;

procedure create;                                         {初始化过程}
 var i:word;
 begin
   m:=n+p;
   for i:=1 to p do
     dec(m, cl[i].mount)                                  {计算空格数}
 end;

procedure print;                                          {输出过程}
 var i, j:word;
 begin
   for i:=1 to m do
     if l[i]=0
       then write('_')
       else for j:=1 to cl[l[i]].mount do
                write(cl[l[i]].color);
   writeln
 end;
```

```
procedure search(le, p1:word);                          {搜索过程}
 begin
   if le > m
     then begin
            if p1 > p then begin
                             print;
                             inc(tot)
                             end;
            exit
      end;
   if p1 < = p then begin
                      l[le]: = p1;  l[le + 1]: = 0;
                      search(le + 2, p1 + 1)            {搜索在该点放灯的情况}
                    end;
   l[le]: = 0;  search(le + 1,  p1)                     {搜索不在该点放灯的情况}
 end;

function f(p:word):longint;                             {计算阶乘}
 var g:longint;       i:word;
 begin
   g: = 1;
   for i: = 2 to p do g: = g * i;
   f: = g
 end;

begin
 initialize;                                            {读数据}
  create; tot: = 0;                                     {初始化}
  search(1, 1);                                         {搜索方案}
  writeln(tot * f(p))                                   {输出结果}
end.
```

BASIC 语言参考程序:

```
REM C1995_5B
  MAXN = 20
  DIM CL $(MAXN), MNT(MAXN), L(MAXN + 1)
 INPUT N
   P = 0
 INPUT X $
  DO WHILE X $ < > "Q"
    P = P + 1
    CL $(P) = X $ :INPUT MNT(P)
    INPUT X $
  LOOP
M = 0
      FOR I = 1 TO P
        M = M + MNT(I)
      NEXT I
  M = N - M + P
   DIM G(P), D(P)
   FOR I = 1 TO P
     D(I) = M - 2 * (P - I)
   NEXT I
   T = 1 :G(T) = 0 :TOL = 0
DO WHILE T > 0
      G(T) = G(T) + 1
  IF G(T) > D(T) THEN
        G(T) = 0
        T = T - 1
  ELSE
        IF T = P THEN
    GOSUB PRT
                ELSE
                 T = T + 1
                 G(T) = G(T - 1) + 1
        END IF
  END IF
   LOOP
     N = 1
     FOR I = 2 TO P
  N = N * I
```

```
        NEXT I
PRINT N * TOL
    END
    PRT:REM
TOL = TOL + 1:G(0) = 1
    FOR I = 1 TO P
      FOR J = G(I - 1) TO G(I) - 1
        PRINT "_";
  NEXT J
      FOR J = 1 TO MNT(I)
        PRINT CL $(I);
      NEXT J
      NEXT I
  FOR J = G(P) + 1 TO M
    PRINT "_";
  NEXT J
        PRINT
  RETURN
```

【运行结果】

输入：

6

R 1

Q

输出：

R _____

__R _____

____R _____

_____R ____

_____R __

_____R

6

输入：

8

R 2

B 3

Q

输出:

RR __BBB ____
RR ____BBB __
RR ______BBB
__RR __BBB __
__RR ____BBB
____RR __BBB
12

输入:

20
B 5
R 3
Y 2

输出:

BBBBB __RRR __YY ____________
BBBBB __RRR ____YY ____________
BBBBB __RRR ______YY ____________
BBBBB __RRR ________YY ____________
BBBBB __RRR __________YY ____________
BBBBB __RRR ____________YY ____________
BBBBB __RRR ____________YY ____
BBBBB __RRR ____________YY __
BBBBB __RRR ____________YY
BBBBB ____RRR __YY __
BBBBB ____RRR ____YY ____________
BBBBB ____RRR ______YY ____________
BBBBB ____RRR ________YY ____________
BBBBB ____RRR __________YY ____________
BBBBB ____RRR ____________YY ____
BBBBB ____RRR ____________YY __
BBBBB ____RRR ____________YY
BBBBB ______RRR __YY ____________
BBBBB ______RRR ____YY ____________
BBBBB ______RRR ______YY ____________
BBBBB ______RRR ________YY ____________
BBBBB ______RRR __________YY ____
BBBBB ______RRR ____________YY __

BBBBB _____RRR _____YY
BBBBB _____RRR __YY _____
BBBBB _____RRR ____YY _____
BBBBB _____RRR _____YY _____
BBBBB _____RRR _____YY ____
BBBBB _____RRR _____YY __
BBBBB _____RRR _____YY
BBBBB _____RRR __YY _____
BBBBB _____RRR ____YY _____
BBBBB _____RRR _____YY ____
BBBBB _____RRR _____YY __
BBBBB _____RRR _____YY
BBBBB _____RRR __YY _____
BBBBB _____RRR ____YY ____
BBBBB _____RRR _____YY __
BBBBB _____RRR _____YY
BBBBB _____RRR __YY ____
BBBBB _____RRR ____YY __
BBBBB _____RRR _____YY
BBBBB _____RRR __YY __
BBBBB _____RRR ____YY
BBBBB _____RRR __YY
__BBBEB __RRR __YY _____
__BBBEB __RRR ____YY _____
__BBBEB __RRR _____YY _____
__BBBEB __RRR _____YY _____
__BBBEB __RRR _____YY _____
__BBBEB __RRR _____YY ____
__BBBEB __RRR _____YY __
__BBBEB __RRR _____YY
__BBBEB ____RRR __YY _____
__BBBEB ____RRR ____YY _____
__BBBEB ____RRR _____YY _____
__BBBEB ____RRR _____YY _____
__BBBEB ____RRR _____YY ____
__BBBEB ____RRR _____YY __
__BBBEB ____RRR _____YY
__BBBBB _____RRR __YY _____
__BBBBB _____RRR ____YY _____

__BBBBB _____RRR _____YY _____
__BBBBB _____RRR _____YY ____
__BBBBB _____RRR _____YY __
__BBBBB _____RRR _____YY
__BBBBB _____RRR __YY _____
__BBBBB _____RRR ___YY _____
__BBBBB _____RRR _____YY ____
__BBBBB _____RRR _____YY __
__BBBBB _____RRR _____YY
__BBBBB _____RRR __YY _____
__BBBBB _____RRR ___YY ____
__BBBBB _____RRR _____YY __
__BBBBB _____RRR _____YY
__BBBBB _____RRR __YY ____
__BBBBB _____RRR ___YY __
__BBBBB _____RRR _____YY
__BBBBB _____RRR __YY __
__BBBBB _____RRR ___YY
__BBBBB _____RRR __YY
__BBBBB __RRR __YY _____
__BBBBB __RRR ___YY _____
____BBBBB __RRR _____YY _____
____BBBBB __RRR _____YY _____
____BBBBB __RRR _____YY ____
____BBBBB __RRR _____YY __
____BBBBB __RRR _____YY
____BBBBB ____RRR __YY _____
____BBBBB ____RRR ___YY _____
____BBBBB ____RRR _____YY _____
____BBBBB ____RRR _____YY ____
____BBBBB ____RRR _____YY __
____BBBBB ____RRR _____YY
____BBBBB _____RRR __YY _____
____BBBBB _____RRR ___YY _____
____BBBBB _____RRR _____YY ____
____BBBBB _____RRR _____YY __
____BBBBB _____RRR _____YY
____BBBBB _____RRR __YY _____
____BBBBB _____RRR ___YY ____

____BBBBB _____RRR _____YY __
____BBBBB _____RRR _____YY
____BBBBB _____RRR __YY ____
____BBBBB _____RRR ____YY __
____BBBBB _____RRR _____YY
____BBBBB _____RRR __YY __
____BBBBB _____RRR ____YY
____BBBBB _____RRR __YY
_____BBBBB __RRR __YY _____
_____BBBBB __RRR ____YY _____
_____BBBBB __RRR _____YY _____
_____BBBBB __RRR _____YY ____
_____BBBBB __RRR _____YY __
_____BBBBB __RRR _____YY
_____BBBBB ____RRR __YY _____
_____BBBBB ____RRR ____YY _____
_____BBBBB ____RRR _____YY ____
_____BBBBB ____RRR _____YY __
_____BBBBB ____RRR _____YY
_____BBBBB _____RRR __YY _____
_____BBBBB _____RRR ____YY ____
_____BBBBB _____RRR _____YY __
_____BBBBB _____RRR _____YY
_____BBBBB _____RRR __YY ____
_____BBBBB _____RRR ____YY
_____BBBBB _____RRR __YY
_____BBBBB __RRR __YY _____
_____BBBBB __RRR ____YY _____
_____BBBBB __RRR _____YY ____
_____BBBBB __RRR _____YY __
_____BBBBB __RRR _____YY
_____BBBBB ____RRR __YY _____
_____BBBBB ____RRR ____YY ____
_____BBBBB ____RRR _____YY __
_____BBBBB _____RRR _____YY
_____BBBBB _____RRR __YY ____

_____BBBBB _____RRR ___YY __
_____BBBBB _____RRR _____YY
_____BBBBB _____RRR __YY __
_____BBBBB _____RRR ___YY
_____BBBBB _____RRR __YY
_____BBBBB __RRR __YY _____
_____BBBBB __RRR ___YY ___
_____BBBBB __RRR _____YY __
_____BBBBB __RRR _____YY
_____BBBBB ___RRR __YY ___
_____BBBBB ___RRR ___YY __
_____BBBBB ___RRR _____YY
_____BBBBB _____RRR __YY __
_____BBBBB _____RRR ___YY
_____BBBBB _____RRR __YY
_____BBBBB __RRR __YY ___
_____BBBBB __RRR ___YY __
_____BBBBB __RRR _____YY
_____BBBBB ___RRR __YY __
_____BBBBB ___RRR ___YY
_____BBBBB _____RRR __YY
_____BBBBB __RRR __YY __
_____BBBBB __RRR ___YY
_____BBBBB ___RRR __YY
_____BBBBB __RRR __YY

990

【小结】

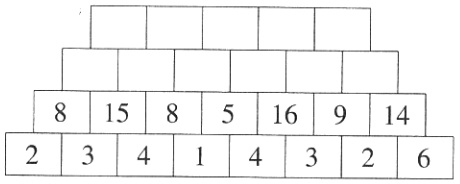
本题要求输出排列的方案,从而限制了算法的优化。如果将排列的方案去掉,那么就可以用 $C_{k+1}^{p} \times p!$ 来计算结果。这样效率会大大提高,可以承受较大规模的数据。

六、高中组第3题

【问题描述】

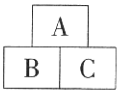
设有一个四层的积木块,1~4层积木块的数量依次为:5, 6, 7, 8

如下图所示放置:



其中，给出第三层与第四层所标的数字，并已知第三层的数据是由第四层的数据计算出来的。

计算的方法是：第三层的某个数据 A 是由第四层相邻的两个数据 B，C 经过某种计算后产生的：



计算所用到的计算符为：+，-，×，且无优先级之分（自左向右计算），运算符最多为 2 个。

如：3 +4 ×5 =35　　5 ×4 +3 =23

可以看出，上图中的第三层的数据是由第四层的数据用以下计算公式计算出来的：

$$A = B \times C + B$$

也就是：8 =2 ×3 +2，15 =3 ×4 +3，…，14 =2 ×6 +2

程序要求：

给出第三层与第三层的数据后，将第一、二层的每块积木标上相应的数据，并输出整个完整的积木图及计算公式。

① 输入数据不存在出错的情况，同时也不会超过整数的范围；

② 计算时可允许出现以下情况：

A = B　　（即可理解为运算符的个数为零）

A = B × B + B　　（即全部由 B 产生）

【问题分析】

这是一道搜索题。

题目规定运算符最多为 2 个，2 个运算符最多可有 3 个运算数参加运算。因此，搜索的对象最多为 5 个。而运算数有两种（B 或 C）；运算符有三种（+，-，×），四种取法：+，-，×和不取。所以，最坏情况下的时间复杂度为 $O(4^2 \times 2^3)$，这是完全可以承受的。

此外，由于题目规定运算符无优先级之分，因此我们就无须进行复杂的表达式计算。这就使编程复杂度大大降低了。

【数据结构】

```
const symbol :array [0..3] of Char
             =(' ','+','-','*');          {定义运算符}
                                           maxn =8;{数塔的宽度 n =8}

type tower = array [1..4, 1..maxn] of integer;    {定义数塔类型}
     operation = array [0..2] of integer;         {定义存放运算数(符)的数组}
```

```
var n:integer;                    {数塔宽度 n}
    t:tower;                      {数塔}

var sym,                          {存放运算符的数组}
    op:operation;                 {存放运算数的数组}
```

【算法分析】

定义四个运算符:+,-,×和“空”,分别用 1,2,3 和 0 表示,其中“空”表示不取。例如,下面的表达式:

$$a = b$$

则两个运算符均为“空”。

设两个运算符分别为 sym[1]和 sym[2],三个运算数分别为 op[1]、op[2]和 op[3]。sym 的取值范围为 0~3,op 的取值范围为 0~1(0 表示 B,1 表示 C)。

分别枚举 sym 和 op,从而得到表达式,然后判断表达式是否满足要求:若满足,即得到我们需要的解;否则,继续枚举。这样整个题目就圆满解决了。

算法流程如下:

<table>
<tr><td colspan="3">op[1] ← 0~1</td></tr>
<tr><td colspan="3">sym[1] ← 0~3</td></tr>
<tr><td colspan="3">op[2] ← 0~1</td></tr>
<tr><td colspan="3">sym[2] ← 0~3</td></tr>
<tr><td colspan="3">op[3] ← 0~1</td></tr>
<tr><td rowspan="2">判断当前的op和
sym(即表达式)
是否满足要求</td><td rowspan="2">是</td><td>输出</td></tr>
<tr><td>退出</td></tr>
<tr><td>否</td><td></td></tr>
</table>

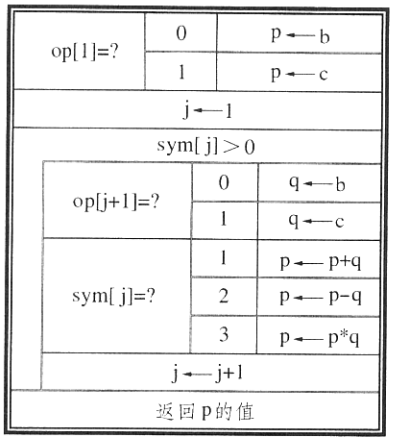
判断表达式是否满足要求的流程:

<table>
<tr><td colspan="3">k ← 1</td></tr>
<tr><td colspan="3">(k<= 7) and (表达式对第 3 行第 k 个元素适用)</td></tr>
<tr><td colspan="3">k ← k+1</td></tr>
<tr><td rowspan="2">k<= 7?</td><td>是</td><td>表达式满足要求</td></tr>
<tr><td>否</td><td>表达式不满足要求</td></tr>
</table>

判断表达式对第 3 行第 k 个元素是否适用的流程:

<table>
<tr><td colspan="3">a ← t[3,k] b ← t[4,k] c ← t[4,k+1]</td></tr>
<tr><td>p ← calc(b,c)</td><td colspan="2">(calc(b,c)表示有关 b 和 c 表达式的值)</td></tr>
<tr><td rowspan="2">a=p?</td><td>是</td><td>返回 true</td></tr>
<tr><td>否</td><td>返回 false</td></tr>
</table>

calcb(b, c)的计算流程：



【程序清单】

```
program g1995_3;

const symbol:array [0..3] of char
          =(' ','+','-','*');                    {定义运算符}
      maxn=8;                                    {数塔的宽度 n=8}
type tower=array [1..4, 1..maxn] of integer;     {定义数塔类型}
      operation=array [0..2] of integer;         {定义存放运算数(符)的数组}

var n:integer;                                   {数塔宽度 n}
    t:tower;                                     {数塔}

procedure initialize;                            {读数据过程}
 var i:integer;
 begin
   for i:=3 to 4 do
     begin
       n:=0;
       while not eoln do
         begin
           inc(n);
           read(t[i, n])
         end;                                    {读入第 i 层的数据}
       readln
     end;
```

```
    dec(n)
  end;
procedure solve;                          {搜索过程}
  var sym,                                {存放运算符的数组}
      op:operation;                       {存放运算数的数组}

  function calculate(y, x:integer):integer;   {表达式计算函数}
   var p,                                 {保存运算结果}
       j:integer;                         {循环变量}

   function choose(op:integer):integer;   {调出运算数}
    begin
      case op of
      0:  choose:=t[y+1, x];
      1:  choose:=t[y+1, x+1]
          end
    end;

  begin
    p:=choose(op[0]);                     {初始化}
    for j:=1 to 2 do
    case sym[j] of
       0:begin
           calculate:=p;
           exit
         end;
       1:inc(p, choose(op[j]));
       2:dec(p, choose(op[j]));
       3:p:=p*choose(op[j])
  end;                                    {计算}
    calculate:=p
  End;

procedure print;                          {输出过程}
  var i, j:integer;

  function choose(op:integer):char;       {判断运算数是 B 还是 C}
```

```
    begin
      case op of
      0:choose:='b';
      1:choose:='c'
      end;
    end;

begin
     for i:=2 downto 1   do
       for j:=1 to n+i-3 do t[i,j]:=calculate(i,j);   {计算数塔的前2层}
     for i:=1 to 4 do
begin
     for j:=1 to n+i-3 do write(t[i,j],' ');
     writeln;
     end;                                              {输出数塔}

     write('a=',choose(op[0]));
     for i:=1 to 2 do
       begin
         if sym[i]=0 then exit
                     else write(symbol[sym[i]]);
         write(choose(op[i]))
       end;                                            {输出表达式}
end;

 procedure judge;                                      {判断过程}
  var k:integer;
  begin
    k:=1; while (k<=n) and (calculate(3,k)=t[3,k]) do inc(k);
                                                       {判断表达式是否符合要求}
    if k>n
      then begin
                  print;                               {输出}
                  writeln;
                  halt;                                {退出}
           end
  end;
```

```
begin
  for op[0]:=0 to 1 do                {枚举第一个运算数}
    for sym[1]:=0 to 3 do             {枚举第一个运算符}
      for op[1]:=0 to 1 do            {枚举第二个运算数}
        for sym[2]:=0 to 3 do         {枚举第二个运算符}
          for op[2]:=0 to 1 do        {枚举第三个运算数}
            judge;                    {判断}
 end;
begin
 initialize;                          {读数据}
 solve;                               {搜索并输出}
end.
```

【运行结果】

输入:

8 15 8 5 16 9 14

2 3 4 1 4 3 2 6

输出:

17408 6615 4128 13685 21760

128 135 48 85 160 135

8 15 8 5 16 9 14

2 3 4 1 4 3 2 6

a = b * c + b

【小结】

本题的难度不大,考的主要是编程功底。不过只要理清算法的思路,逐层深入,就不难将问题分为几个模块。对模块分别实现后,再将几个模块联系起来,整个问题就实现了。

第二章　1996年复赛试题解析

一、初中组第1题

【问题描述】

从键盘读入2个100以内的正整数，进行乘法运算并以竖式输出。

例如，输入：89　13　　又如：输入16　8

输出格式：

```
   89          16
 * 13        *  8
-----       -----
  267         128
  89
-----
 1157
```

【问题分析】

这是一道基础题，主要考察选手编程的基本功。

设两个正整数分别为A、B。由于题目要求以竖式输出，因此必须用B的每一位与A相乘，并输出相应的值。这里可以使用字符串类型。即将B转换成字符串，然后从后往前每次取一位数与A相乘。

还需注意的是输出的格式。在输出B的某一位数与A的积时，要注意前面的空格数，使中间和最后的结果以及A、B都互相按要求对齐。这可以用输出语句的场宽设置来实现。

最后还必须考虑一些特例。例如，当B是一位数时，只需输出一行结果；当B的某一位数为零，根据要求需输出相当于A的长度个“0”。这些，都是测试时可能遇到的，如果不加以注意，就会导致失分，是很划不来的。

【数据结构】

```
var a , b, i, len : byte ;          {两个整数a,b;i为循环变量;len为b的长度}
    s:string[3] ;                   {b相应的字符串}
```

【算法分析】

读入两个正整数a,b;输出a*b的竖式形式;

将b转化为字符串s;len设为字符串s的长度;

```
for i: =len to 1 do
  [if s[i] = '0' then 输出length(a)个'0'
                 else writeln(a乘以s[i]的积:6 +i -len)]
if len < >1 then 输出a*b的积
```

【程序清单】

```
program C1996_1;
var a, b, i, len:byte;                    {两个整数 a,b;i 为循环变量;len 为 b 的长度}
    s:string[3];                          {b 相应的字符串}
begin
 readln(a, b);
  str(b, s);                              {将 b 转化为字符串 s}
  writeln(a:6); writeln(' *', b:4);       {输出 a * b 的竖式形式}
  writeln('- - - - - - - - - - -');
  len:=length(s);                         {求得 s 的长度}
  for i:=len downto 1 do                  {从后向前取 b 的每一位相乘}
      if s[i]<>'0' then writeln((ord(s[i])-48)*a:6+i-len)      {输出每一步的积}
                   else if a>=100         {当 b 的当前位为 0 时的特殊处理}
                        then writeln('000':6+i-len)
                        else  if a>=10 then writeln('00':6+i-len)
                                       else writeln('0':6+i-len);
  if len<>1 then begin writeln('- - - - - - - - - - -');
                                          {若 b 的长度不为 1,则输出最后结果}
                       writeln(a*b:6)
                 end
end.
```

BASIC 语言参考程序:

```
 REM C1996_1B
 INPUT "A, B";A, B
 X=INT(B/10)
 Y=B MOD 10
PRINT
 IF A>=10 THEN PRINT TAB(10);A ELSE PRINT TAB(11);A
PRINT "   *";
 IF B>=10 THEN PRINT TAB(10);B ELSE PRINT TAB(11);B
PRINT "- - - - - - - - - - - - - - - - - -"
 IF X=0 THEN
   C$=STR$(A*Y)
   PRINT TAB(13-LEN(C$));C$
   ELSE
C$=STR$(A*Y)
```

```
PRINT TAB(13 - LEN(C $));C $
C $ = STR $(A * X)
PRINT TAB(12 - LEN(C $));C $
PRINT " - - - - - - - - - - - - - - - - - - - - -"
C $ = STR $(A * B)
PRINT TAB(13 - LEN(C $));C $
      END IF
 END
```

【运行示例】

输入:89　13

输出:

```
    8 9
  * 1 3
  -----
  2 6 7
  8 9
  -----
1 1 5 7
```

输入:25　47

输出:

```
    2 5
  * 4 0
  -----
    0 0
1 0 0
-------
1 0 0 0
```

输入:98　2

输出:

```
    9 8
*     2
  -----
  1 9 6
```

【小结】

应该说,这是一道较简单的题目。但是在做这道题时,还是需要全面的考虑各种情况,才能保证拿满分。很多看似简单的题目,实际上都需要我们能更全面的考虑,才能得到更好的解决。而在竞赛中,在简单题上失分是非常可惜的,因此,同学们必须培养全面考虑问题的习惯,避免在竞赛后留下遗憾。

二、初中组第2题

【问题描述】

输入三个自然数N,i,j (1<=i<=N,1<=j<=N),输出在一个N*N格的棋盘中,与格子(i, j)同行、同列、同一对角线的所有格子的位置。

如:n=4, i=2, j=3表示了棋盘中的第二行第三列的格子,如下图:

第一列	第二列	第三列	第四列	
				第一行
		2, 3		第二行
				第三行
				第四行

要求编制一个程序,根据输入的N, i, j的值,输出与格子(i, j)在同一行、同一列、同一对角线上的所有格子位置,例如:当n=4, i=2, j=3时,输出的结果是:

```
(2, 1) (2, 2) (2, 3) (2, 4)     {同一行上格子的位置}
(1, 3) (2, 3) (3, 3) (4, 3)     {同一列上格子的位置}
(1, 2) (2, 3) (3, 4)            {左上到右下对角线上的格子的位置}
(4, 1) (3, 2) (2, 3) (1, 4)     {左下到右上对角线上的格子的位置}
```

【问题分析】

本题的关键是找到与格子(i, j)同行、同列、同一对角线的格子的位置与N,i,j的关系。

通过观察可以发现,与(i, j)同行的格子的横坐标都为i,纵坐标为1~N;与(i, j)同列的格子恰好相反:纵坐标都为j,横坐标为1~N。

而要找与(i, j)同一对角线的格子则相对复杂。首先,存在两条经过(i, j)的对角线:一条从棋盘的左上到右下,另一条从左下到右下。对于在从左上到右下的对角线上的格子(x, y),可以发现其始终满足等式x - y = i - j。而对于从左上到右下的对角线上的格子(x', y'),满足x' + y' = i + j。还需注意的是与(i, j)处在同一对角线上的格子并不一定是每一行每一列都存在的,这在编程时应特别注意。

【数据结构】

```
var n, i, j, k:integer; {n为棋盘规模;i,j为给定格子的坐标;k为循环变量}
```

【算法分析】

读入n, i, j;

枚举列号,输出与(i, j)同一行的所有格子;

枚举行号,输出与(i, j)同一列的所有格子;

枚举行号,判断在该行上是否有与(i, j)处在同一从左上到右下对角线上的格子,有则输出其坐标;

枚举行号,判断在该行上是否有与(i, j)处在同一从左下到右上对角线上的格子,有则输出其坐标;

【程序清单】

```
program c1996_2;
var n, i, j, k:integer;                    {n 为棋盘规模;i, j 为给定格子的坐标;k 为循环变量}
begin
  readln(n, i, j);                                          {读入数据}
  for k:=1 to n do write('(', i, ', ', k, ')');             {输出同行的格子}
  writeln;
  for k:=1 to n do write('(', k, ', ', j, ')');             {输出同列的格子}
  writeln;
  for k:=1 to n do                                          {输出同一对角线上的所有格子}
    if (k+j-i>0) and (k+j-i<=n) then write('(', k, ', ', k+j-i, ')');
  writeln;
  for k:=n downto 1 do                                      {输出同一对角线上的所有格子}
     if (i+j-k>0) and (i+j-k<=n) then write('(', k, ', ', i+j-k, ')');
  writeln
end.
```

BASIC 语言参考程序：

```
REM C1996_2b
INPUT"N, I, J=";N, I, J
FOR K=1 TO N
PRINT  "(";I;", ";J;")";
NEXT K
PRINT
FOR K=1 TO N
PRINT"(";K;", ";J;")";
NEXT K
PRINT
FOR K=1 TO N
 IF (K+J-I>0) AND (k+J-I<=N) THEN PRINT"(";K;", ";K+J-I;")";
NEXT K
PRINT
FOR K=N TO 1 STEP -1
 IF (I+J-K>0) AND (I+J-K<=N) THEN PRINT"(";K;", ";I+J-K;")";
NEXT K
PRINT
END
```

【运行示例】

输　入	4 4 1	1 1 1
输　出	(4, 1)(4, 2)(4, 3)(4, 4) (1, 1)(2, 1)(3, 1)(4, 1) (4, 1) (4, 1)(3, 2)(2, 3)(1, 4)	(1, 1) (1, 1) (1, 1) (1, 1)

【小结】

本题也是一道很基本的题目,关键在于考察选手从特殊情况推出一般规律的能力,正如哲学上所说的"从现象到本质","通过事物的外部现象找到其内部的规律"。这也是参加信息学竞赛时不可欠缺的一项能力。

三、初中组第3题

【问题描述】

从键盘输入一长度不超过40的字符串,按要求进行删除、插入或替换操作。

例如:'This is a book.' 现对该字符串进行编辑,编辑功能有:

D:删除一个字符,命令的方式为:

D　a　其中a为被删除的字符

例如:D　s　表示删除字符's',若字符串中有多个's',则删除第一次出现的。如上例中删除的结果为:'Thi is a book.'

I:插入一个字符,命令的格式为:

I　a1　a2　其中a1表示插入到指定字符前面,a2表示将要插入的字符。

例如:I　s　d　表示在指定字符's'的前面插入字符'd',若原串中有多个's',则插入在最后一个字符的前面,如上例中:

原　串:'This is a book.'

插入后:'This ids a book.'

R:替换一个字符,命令格式为:

R　a1　a2　其中a1为被替换的字符,a2为替换的字符,若在原串中有多个a1则应全部替换。例如:

原　串:'This is a book.'

输入命令:R　o　e

替换后的字符串为:'This is a beek.'

在编辑过程中,若出现被改的字符不存在时,则给出提示信息。

【问题分析】

目前我们常用的Turbo Pascal系统提供了一种字符串数据类型:String,并且预定义了许多关于字符串类型的函数和过程,大大方便了我们对字符串的处理。在本题中,使用字符串类型,可以简化编程并提高程序效率。

关于字符串类型的详细内容，同学们可以参考相关的书籍资料，这里只就本题需要用到的字符串操作做一简单介绍：

- Delete 过程：

  语法：Delete(Var st:String;pos, num:Byte)

  语义：删除字符串 st 中从 pos 开始的 num 个字符。

- Insert 过程

  语法：Insert(obj:String;Var target:String;pos:Byte)

  语义：将串 obj 插入到串 target 的 pos 位置上。

- Length 函数

  语法：Length(st:String) :Byte

  语义：函数值为字符串 st 的长度

- Pos 函数

  语法：Pos(obj, target:String) :Byte

  语义：寻找串 target 中第一个出现的 obj 字符串，返回其所在位置；若没有找到，函数值为 0。

另外，字符串也可以当作基类型为 Char 的数组来处理，即可以通过使用下标来访问字符串中的每一个字符。还需注意 String 类型的字符串其长度不能超过 255。

【数据结构】

```
var s:string[40];          {读入的字符串}
    op ,                   {op 为操作符,可能为 D(删除),I(插入),R(替换)}
    c1:char;               {c1 为插入或替换的字符}
    p, i, len:byte;        {p 为进行操作的字符在 s 中的位置}
                           {i 为循环变量;len 为 s 的长度}
```

【算法分析】

```
读入字符串 s; 读入操作符 op 及需进行操作的字符 sp;
求出 sp 在字符串 s 中的位置 p,若不存在,则输出提示信息,程序结束;
求 s 的长度 len;
case op of
  'D': 删除 s 中位置 p 处的字符;
  'I' :[ 读入插入的字符 c1;
         找到 s 中最后一个 sp;
         在 sp 的位置上插入 c1] ;
  'R' :[ 读入替换的字符 c1;
         for i := 1 to Len do 若 s 的 i 位置上为 sp,则替换为 c1 ] ;
end;
输出操作后的串 s。
```

【程序清单】

```
program c1996_3;
var s:string[40];
    op, c1:char;
    p, i, len:byte;
begin
  readln(s);                                    {读入串 s}
  repeat read(op) until op in ['D', 'I', 'R'];  {读入操作符 op}
  repeat read(c1) until c1 <>' ';               {读入需操作的字符}
  p:=pos(c1, s);                                {求其在 s 中的位置}
  if p=0 then begin                             {没找到,则输出提示信息,程序结束}
                writeln('there is no ', c1, ' in the string. ');
                halt
              end;
  len:=length(s);                               {求 s 的长度 len}
  case op of
    'D':delete(s, p, 1);                        {进行删除操作}
    'I':begin                                   {进行插入操作}
          repeat read(c1) until c1 <>' ';       {读入插入的字符 c1}
          i:=p+1;
          while i<=len do                       {找最后一个 s[p]的位置 i}
            begin
              if s[i]=s[p] then p:=i;
              inc(i)
            end;
          insert(c1, s, p)                      {将 c1 插入到 i 处}
        end;
    'R':begin                                   {进行替换操作}
          repeat read(c1) until c1 <>' ';       {读入替换的字符 c1}
          for i:=p+1 to len do                  {将 s 中所有的 s[p]替换为 c1}
            if s[i]=s[p] then s[i]:=c1;
          s[p]:=c1
        end
  end;
  writeln(s)                                    {输出经过操作的串 s}
end.
```

BASIC 语言参考程序：

```
REM C1996_3B
DIM S $(40)
INPUT SS $
L = LEN(SS $)
FOR K = 1 TO L
 S $(K) = MID $(SS $, K, 1)
NEXT K
INPUT "OP $ = 'D', 'I', 'R'";OP $
INPUT CL $
P = 0:K = 1
DO WHILE (P = 0)AND(K < = L)
   IF CL $ = S $(K)THEN P = K ELSE K = K + 1
      LOOP
IF P = 0 THEN PRINT "there is no";CL $;"in the string!":END
SELECT CASE OP $
CASE"D", "d"
FOR K = P TO L - 1
S $(K) = S $(K + 1)
NEXT K
L = L - 1
CASE"I", "i"
INPUT CL $
K = L
DO WHILE (S $(K) < > S $(P))AND (K > 0)
   K = K - 1
LOOP
P = K
FOR K = L TO P STEP - 1
 S $(K + 1) = S $(K)
NEXT K
S $(P) = CL $
L = L + 1
CASE"R", "r"
INPUT CL $
FOR K = L TO p step  - 1
if s $(k) = s $(p) then S $(K) = CL $
```

```
NEXT K
END SELECT
FOR K = 1 TO L
PRINT S $(K);
NEXT K
PRINT
END
```

【运行示例】

输　入	a123b12aa. D 2	abc dc e. I c f	abababababab. R b a	abcdefg R s I
输　出	a13b12aa	abc dfc e.	aaaaaaaaaaa.	there is no s in the string.

【小结】

要求你计算一个10位数的平方根,你会使用计算机,还是掰着手指头算呢?答案是显然的。

采用先进的工具常常可以取得事半功倍的效果。在编程时,采用高级高效的数据类型和数据结构也往往可以简化编程的复杂度,提高程序的效率,并可提高程序的可读性——何乐而不为呢?

四、初中第4题/高中第1题

【问题描述】

有2^n(n<=6)个球队进行单循环比赛,计划在2^n-1天内完成,每个队每天进行一场比赛。设计一个比赛的安排,使在2^n-1天内每个队都与不同的对手比赛。

例如n=2时的比赛安排:

队	1	2	3	4	
比赛	1 = =2		3 = =4		一天
	1 = =3		2 = =4		二天
	1 = =4		2 = =3		三天

【问题分析】

本题的数据规模是比较小的(题目限定n<=6,即$2^n<=64$),于是,可以使用搜索来解。其具体思路为:枚举每一天;在某天,对于在这天还未参加比赛的球队i,从在这天还未参加过比赛的其他球队中找一个还未曾与i比赛过的球队与i比赛。如此往复即可得到全部的安排。其算法复杂度为$O(N^3)$,在本题中,复杂度最大达到$O((2^6)^3)=O(262144)$,完全可以忍受。

【数据结构】

```
type tlist = array[1..maxn] of boolean;
     tbox = array[1..maxn] of tlist;
var
 a:tbox; {s[i, j]表示球队 i 与球队 j 是否比赛过,是则为 true,否则为 false}
 b:tlist; {b[i]表示球队 i 在当前这一天是否参加过比赛,是则为 true,否则为 false}
```

【算法分析】

```
读入 n;n:=2^n;
for d:=1 to n-1 do
  for i:=1 to n do
    if 球队 i 在第 d 天还未安排比赛
      then for j:=1 to n do
             if 球队 j 在第 d 天还未安排比赛 and i 与 j 还未比赛过
               then [ 这天 i 与 j 比赛;break ]
```

【程序清单】

```
program g1996_1;
const maxn=64;
type tlist=array[1..maxn] of boolean;
     tbox=array[1..maxn] of tlist;
var a:tbox;
    b:tlist;
    n, d, i, j:integer;
begin
  readln(n);                          {读入 n}
  n:=round(exp(ln(2)*n));             {求 n:=2^n}
  fillchar(a, sizeof(a), 0);          {数组 a 初始化}
  for d:=1 to n-1 do                  {枚举每一天}
    begin
      fillchar(b, sizeof(b), 0);      {数组 b 初始化}
      write('<', d, '>');             {输出天数}
      for i:=1 to n do                {枚举每一球队}
        if not b[i] then begin        {若在第 d 天还未安排比赛}
                         b[i]:=true;      {则为其安排比赛}
                         write(i);
                         for j:=1 to n do
                                         {从在这天还未参加过比赛的其他球队中}
```

```
                                        {找一个还未曾与 i 比赛过的球队 j}
                                    if not b[j] and not a[i, j] then break;
                                write('= =', j, '  ');              {i 与 j 比赛}
                                b[j]:=true; a[i, j]:=true; a[j, i]:=true
                            end;
        writeln
    end
end.
```

BASIC 语言参考程序:

```
REM C1996_4B
MAXN=64
DIM A(MAXN, MAXN), B(MAXN)
INPUT"N=";N
NN=1
FOR I=1 TO N
NN=NN*2
NEXT I
FOR I=1 TO MAXN
 FOR J=1 TO MAXN
  A(I, J)=0
 NEXT J
NEXT I
FOR D=1 TO NN-1
 FOR I=1 TO MAXN
  B(I)=0
 NEXT I
PRINT"<";D;">";
FOR I=1 TO NN
 IF B(I)=0 THEN
  B(I)=1
  PRINT I;
  FOR J=1 TO NN
  IF (B(J)=0)AND(A(I, J)=0) THEN
  PRINT"==";J;" ";
  B(J)=1:A(I, J)=1:A(J, I)=1
  J=NN
```

```
  END IF
  NEXT J
END IF
NEXT I
PRINT
NEXT D
END
```

【运行示例】

输入:1
输出:
〈1〉1 = =2

输入:3
输出:

〈1〉	1 = =2	3 = =4	5 = =6	7 = =8
〈2〉	1 = =3	2 = =4	5 = =7	6 = =8
〈3〉	1 = =4	2 = =3	5 = =8	6 = =7
〈4〉	1 = =5	2 = =6	3 = =7	4 = =8
〈5〉	1 = =6	2 = =5	3 = =8	4 = =7
〈6〉	1 = =7	2 = =8	3 = =5	4 = =6
〈7〉	1 = =8	2 = =7	3 = =6	4 = =5

【小结】

虽然搜索的复杂度往往比较高,但其算法常常比较简单,程序编起来也比较容易,因此,对于数据规模不是很大的问题,搜索不失为一个不错的选择。

五、高中组第 2 题

【问题描述】

数制转换

设有一个字符串 A $ 的结构为: A $ = 'm < n > p'

其中 m 为数字串(长度 < =20),而 n,p 均为 1 或 2 位的数字串(其中所表达的内容在 2 - 10 之间)。

程序要求:从键盘上读入 A $ 后(不用正确性检查),将 A $ 中的数字串 m(n 进制),以 p 进制的形式输出。

例如:A $ = '48 < 10 > 8'

其意义为:将 10 进制数 48,转换成 8 进制数输出。

输出结果为:48 <10 > =60 <8 >

【问题分析】

数制转换是信息学的基础知识,是每个选手必须熟练掌握的内容,因此,本题的基本算法是非常简单明了的。但由于需转换的数的长度最大可能到20,简单的整数类型已经无法承受,因此,在编本题时需要用到高精度计算。这是本题又一个考核的内容。

高精度计算的原理,简单说来,就是用数组来存储数,一般是数组的每单元存储原数字的1或2位数,然后,模仿竖式计算的方法进行运算。具体的算法,可以参考程序清单中的Add(加法),Mul(乘法),Divi(除法)等过程。

【数据结构】

```
const maxsize = 100;        {高精度数组的最大长度}
type tnum = array[0..maxsize] of byte;
                            {高精度数组的类型定义,其中 tnum[0]为数字的长度}
var a:string[20];           {需转换的数,存为字符串形式}
    s:string[26];           {读入的字符串}
    x ,                     {高精度数 x 存储 n^i}
    b1 ,                    {高精度数 b1 存储 a[i+1] * n^i}
    b : tnum;               {高精度数 b 存储 a 转换成的 10 进制数}
```

【算法分析】

读入字符串 s;从中分离出 a,m,n;

若 a 为 0 或 1,则直接输出结果,并结束程序;

若将 a 转换成 10 进制数;

用求余数的方法,递归输出与 a 相应的 n 进制数的每一位。

【程序清单】

```
{$A+,B-,D-,E+,F-,G+,I-,L-,N-,O-,P-,Q-,R-,S-,T-,
 V+,X+,Y-}
{$M 16384,0,655360}
program g1996_2;
const maxsize = 100;
type tnum = array[0..maxsize] of byte;
var a:string[20];
    s:string[26];
    x, b, b1:tnum;
    m, n, p1, p2, i:byte;
    e:integer;

procedure mul(var a:tnum;  x:integer);
                                {高精度数 a 乘以整型数 x,并将结果存放在 a 中}
var i, p:integer;
```

```
begin
  p:=0;  for i:=1 to a[0] do   {用 x 去乘 a 的每一位,并为下一位数传递进位}
    begin
      inc(p, a[i]*x);
      a[i]:=p mod 10;
      p:=p div 10
    end;
  if p<>0 then repeat
                 inc(a[0]);
                 a[a[0]]:=p mod 10;   p:=p div 10
               until p=0
end;

procedure add(var a:tnum;   b:tnum);{两个高精度数 a、b 相加,结果储存在 a 中}
var i, p:integer;
begin
  if a[0]<b[0] then a[0]:=b[0];
  p:=0;
  for i:=1 to a[0] do   {将 a 与 b 的每一位依次相加}
    begin
      inc(p, a[i]+b[i]);
      a[i]:=p mod 10;
      p:=p div 10
    end;
  if p<>0 then begin
                 inc(a[0]);
                 a[a[0]]:=p
               end
end;

function divi(var a:tnum;   x:byte):byte;
                {高精度数 a 除以整数 x,并将商储存在 a 中;函数值返回余数}
var i, p, g:integer;
begin
  p:=0;
  for i:=a[0] downto 1 do
    begin
      g:=(10*p+a[i]) div x;
```

```
        p:=10*p+a[i]-g*x;
        a[i]:=g
      end;
    divi:=p;
    while (a[0]>0) and (a[a[0]]=0) do dec(a[0])
end;

procedure out;                    {将十进制数 b 转换成 n 进制数,递归输出结果的每一位}
var r:byte;
begin
  if b[0]<>0 then begin
                    r:=divi(b, n);
                    out;
                    write(r)
                  end
end;

begin
  readln(s);
  p1:=pos('<', s);
  p2:=pos('>', s);
  a:=copy(s, 1, p1-1);                          {获取 a}
  val(copy(s, p1+1, p2-1-p1), m, e);            {获取 m}
  val(copy(s, p2+1, length(s)-p2), n, e);       {获取 n}
  if (a='1') or (a='0') then begin
                                                {若 a 为 0 或 1,则直接输出结果,退出程序}
                               writeln(a, '<', m, '>=', a, '<', n, '>');
                               halt
                             end;
                                                {以下将 m 进制数 a,转换成十进制数 b}
  fillchar(x, sizeof(x), 0);
  fillchar(b, sizeof(b), 0);
  x[1]:=1; x[0]:=1;
  for i:=1 to p1-1 do
    begin
      b1:=x;
      mul(b1, ord(a[p1-i])-48);
```

```
        add(b, b1);
        if i < p1 then mul(x, m)
      end;
    write(copy(s, 1, p2), '=');
    out;                                          {输出相应的 n 进制数}
    writeln('<', n, '>')
end.
```

【运行示例】

输　入	101 <2 >10	62 <8 >5	221 <9 >6
输　出	101 <2 > =5 <10 >	62 <8 > =200 <5 >	221 <9 > =501 <6 >

【小结】

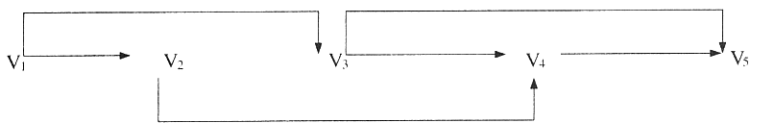
竞赛题除了要考察选手对基本功的熟练情况外，更重要的，是考察选手灵活、综合地运用所学知识的能力。本题就考了两个知识点：数制转换和高精度计算，可以说是一道综合题，尽管其难度并不大。

解综合题的原则，就是将原本比较复杂的题目分解成若干个较易解决的子问题，逐个解决后再将其以一定的方法联合起来，也就解决了原来的题目。这与我们常用的分治算法是同一个原理。

六、高中组第 3 题

【问题描述】

在一个地图上有 N 个地窖（N < =20），每个地窖中埋有一定数量的地雷。同时，给出地窖之间的连接路径，并规定路径都是单向的。某人可以从任一处开始挖地雷，然后沿着指出的连接往下挖（仅能选择一条路径），当无连接时挖地雷工作结束。设计一个挖地雷的方案，使他能挖到最多的地雷。



$V_1, V_2, V_3, \cdots, V_6$ 表示地雷

输入格式：

N：　　（地窖的个数）

$W_1, W_2, W_3, \cdots\cdots W_N$　　（地窖中地雷数）

$A_{12} \cdots\cdots\cdots A_{1N}$　　（$A_{IJ}=1$ 表示地窖 I 到地窖 J 有通路，$A_{IJ}=0$

$A_{23}\cdots A_{2N}$　　表示地窖I到地窖J无通路。)

……

$A_{N-1\,N}$

输出格式:

$K_1--K_2--\cdots\cdots K_V$　　(挖地雷的顺序)

MAX　　(挖地雷的数量)

【问题分析】

本题是运用动态规划的一道经典问题。

若有一个过程可以分为若干个阶段,每一个阶段都有若干决策可供选择,如果第i阶段的决策已定,那么i阶段以后的活动仅与i阶段的过程状态有关,与i阶段以前怎样到达i阶段状态无关。举例来说,从上海到北京有多种路线可走,而当到达徐州以后,从上海到北京就只取决于徐州到北京如何走法了,与上海如何到徐州无关。这就称作"无后效性",它是多阶段决策最优化问题的特征。

如果每一阶段的过程状态(决策),组成了整个问题的最优决策序列:

$$D1,\ D2,\ \cdots,\ Di,\ Di+1,\ \cdots,\ Dn$$

那么Di, Di+1, …, Dn一定是相对于第i阶段的状态Di的最优决策序列,否则,D1, D2, …, Di, Di+1, …, Dn便不可能是最优决策序列。简言之,即最优策略的子策略必然是最优的。这一原理通常称为最优化原理。

一个问题,如果同时符合无后效性和最优化原理,那就可以考虑使用动态规划的方法:利用最优化原理找出递推关系,从而找到最优决策序列,这样往往就能使枚举数量大大下降。这就是用动态规划设计算法的主要思路。

具体来看本题。由于题目条件规定所有路径都是单向的,于是本题就符合了无后效性和最优化原理,可以用动态规划解决。

设W(i)为第i个地窖所有的地雷数,A(i, j)表示第i个地窖与第j个地窖之间是否有通路,F(i)为在从第i个地窖开始最多可以挖到的地雷总数,则有如下关系式:

$$F(i)=\max_{i<j<=n,\ A(i,j)=1}\{F(j)+W(i)\}$$

于是就可以通过递推的方法,从后往前逐个求出所有的F(i),再从中找一个最大的即为问题的解。

对于具体所走的路径,可以通过一个向后的链接来实现,详情可参见算法分析和程序清单。

【数据结构】

```
var a:array[1..maxn, 1..maxn] of shortint;          {地窖之间的连通关系}
    w:array[1..maxn] of integer;                    {每个地窖所有的地雷数}
    f:array[1..maxn] of
        record
          sum:longint;     {f[i].sum 为从第 i 个地窖开始最多可以挖到的地雷总数}
          next:shortint      {f[i].next 为向后连接,即应该再向后挖的地窖的编号}
        end;
```

n:shortint;

【算法分析】

读入 n,数组 w,数组 a;
初始化数组 f;
f[n].sum := w[n];
for i := n - 1 downto 1 do
[求第 i+1 到第 N 个地窖中,与第 i 个地窖连接的、所挖地雷数最大的地窖 k;
f[i].sum := f[k].sum + w[i];f[i].next := k]
找所挖地雷数最大的地窖 k;
从 k 开始,通过向后的连接,输出路径;
输出最大的地雷数目。

【程序清单】

```
program g1996_3;
const maxn=20;
var a:array[1..maxn, 1..maxn] of shortint;
    w:array[1..maxn] of integer;
    f:array[1..maxn] of record
                          sum:longint;
                          next:shortint
                          end;
    i, j, n, k:shortint;
    max:longint;
begin
  readln(n);                                    {读入数据}
  for i:=1 to n do read(w[i]); readln;
  for i:=1 to n-1 do
    begin
      for j:=i+1 to n do read(a[i, j]);
      readln
    end;
  fillchar(f, sizeof(f), 0);                    {f 数组初始化}
  f[n].sum:=w[n];                               {从 f[n]开始,向前求每一个 f[i]}
  for i:=n-1 downto 1 do
    begin
      max:=0; k:=0;                             {求 max{f[j].sum},作为 f[i]的后继}
      for j:=i+1 to n do

        if (a[i, j]=1) and (max<f[j].sum) then begin
```

```
                                        max:=f[j].sum; k:=j
                                    end;
            f[i].sum:=w[i]+max; f[i].next:=k        {得到f[i]}
    end;
  k:=1;                                  {找一个max{f[i].sum}作为问题的解}
  for i:=2 to n do
    if f[i].sum>f[k].sum then k:=i;
  max:=f[k].sum;
  write(k); k:=f[k].next;                          {输出路径方案}
  while k<>0 do
    begin
      write('-', k);
      k:=f[k].next
    end;
  writeln;
  writeln(max)                                     {输出所挖地雷的最大值}
end.
```

【运行示例】

输　入	6 5 10 20 5 4 5 1 0 1 0 0 0 1 0 0 1 0 0 1 1 1	6 10 15 20 15 5 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
输　出	3 -4 -5 -6 34	3 20

【小结】

动态规划问世以来,在工农业生产、经济、军事、工程技术等许多方面都得到了广泛的应用,取得了显著的效果。

动态规划运用于信息学竞赛是在90年代初期,它以独特的优点获得了出题者的青睐。此后,它就成为了信息学竞赛中必不可少的一个重要方法,几乎在所有的国内和国际信息学竞赛中,都至少有一道动态规划的题目。所以,掌握好动态规划,是非常重要的。

动态规划是一种方法,是考虑问题的一种途径,而不是一种算法。因此,它不像深度优先和广度优先那样可以提供一套模式。它必须对具体问题进行具体分析。需要丰富的想像力和创造力去建立模型求解。

七、高中组第 4 题

【问题描述】

现有 1 克、2 克、3 克、5 克、10 克、20 克的砝码各若干枚，问用这些砝码可以称出多少种不同的重量。（设砝码的总重不超过 1000 克，且砝码只能放在天平的一端）

【问题分析】

一看到这道题，首先想到的自然是用穷举法，即将所有可能的砝码组合一一列举，看这些组合总共可以称出多少种不同的质量，其时间复杂度为

O （S1 * S2 * S3 * S4 * S5 * S6 ）（其中 Si 为第 I 种砝码的数量）

但仔细分析一下题目的规模，就会发现这种算法是行不通的。由于题目给出的限定条件是所有砝码的总质量不超过 1000 克，当砝码数比较多时，用上述算法编的程序不可能在规定时间内出解。例如，当六种砝码的个数分别为 200，100，30，20，20，10 时，其总质量为 990 克，符合题义，如用上述算法做，则时间复杂度为 O（200 * 100 * 30 * 20 * 20 * 10 ） = O（2.4 * 10^9 ），显然是无法忍受的。

那么该如何设计该题的算法呢？我们不妨再来研究一下上面的穷举算法。题目限定所有砝码的总质量不超过 1000 克，也就是说，可以称出的质量种数应该也不超过 1000，而对于上面提到的那个数据，程序却枚举了 2.4 * 10^9 次，与 1000 相去甚远，其原因何在呢？关键就在于重复计算！

我们知道，用两个 1 克的砝码和用一个 2 克的砝码产生的效果是一样的，进而言之，当两种不同的砝码组合所称出的质量相同时，再加上相同的砝码，其称出的质量也必然还是一样。这就是穷举算法复杂度过高的原因所在。对于两种总质量相同的组合，穷举法不是采取剪枝，而是都继续向下搜索，这样，实际上就等于多用了一倍的时间。如此反复的重复计算，就会导致程序浪费大量的时间。

因此，设计新算法的关键就变成如何对重复的组合尽早剪枝。在这里，我们不妨使用宽度优先搜索的算法：由一个砝码也不取开始扩展结点，当扩展出的某一个结点所对应的质量数在前面已经出现过时，则不再从该结点扩展下去，并删掉该结点；如此往复，直到没有结点可扩展为止。统计扩展的结点总数，就可得到可以称出的质量总数。具体算法请参考算法分析。

【数据结构】

```
const w:array[1..6] of byte = (1, 2, 3, 5, 10, 20);
                                                {常量数组，六种砝码的单位质量}
      maxweight = 1000;                         {最大质量，也是队列的最大长度}

type twl = array[1..6] of integer;              {六种砝码的数量}
     tlist = array[0..maxweight] of record
          {队列类型，其中 we 为该结点所对应砝码组合的总质量，sn 为每种砝码的个数}
                    we:integer;
                    sn:twl
```

```
                                    end;
  var a:tlist;                                          {队列变量}
      s:twl;                                            {存放数据给出的每种砝码的数量}
      b:array[1..maxweight] of boolean;
  {布尔数组,标记某个质量 k 是否可被称出。若可被称出,则 b[k]=true,否则为 false}
  head, tail:integer;                                   {队首,队尾指针}
```

【算法分析】

```
  读入 s;队列 a 清空;b 清成 false;
  head := 0; tail := 0;
  while  还有结点可扩展  do
      [    取队列头元素 a[Head];
    for i := 1 to 6   do                                {试探每种砝码}
     if  a[Head].sn[I] <= s[i] then                     {新组合}
           [ cw := a[Head].we + w[i] ;                  {cw 为新组合的总质量}
                 if  not B[cw] then 获得质量 cw 的组合进队列;
       ];
         inc(head)
                                                        {队首元素出队}
      ]
  writeln (tail) .
```

【程序清单】

```
{$ A+, B-, D-, E+, F-, G+, I-, L-, N-, O-, P-, Q-, R-, S-, T-,
 V+, X+, Y-}
{$ M 16384, 0, 655360}
program g1996_4;
const w:array[1..6] of byte=(1, 2, 3, 5, 10, 20);
      maxweight=1000;
type  twl=array[1..6] of integer;
      tlist=array[0..maxweight] of record
                                     we:integer;
                                     sn:twl
                                   end;
var a:tlist;
    s:twl;    i:byte;
    b:array[1..1000] of boolean;
    head, tail, cw:integer;
begin
  for i:=1 to 6 do read(s[i]); readln;                  {读入数据}
```

```
  fillchar(a, sizeof(a), 0); fillchar(b, sizeof(b), 0);          {a,b 初始化}
  head:=0; tail:=0;                                              {队列指针初始化}
  while head<=tail do                                            {开始宽度搜索}
    begin
      for i:=1 to 6 do                                           {试探每种砝码}
        if a[head].sn[i]<s[i]                                    {新组合可以得到}
          then begin
                 cw:=a[head].we+w[i];                            {cw 为新组合的总质量}
                 if not b[cw] then begin                         {进队操作}
                                     inc(tail);
                                     a[tail].we:=cw; b[cw]:=true;
                                     a[tail].sn:=a[head].sn;
                                     inc(a[tail].sn[i])
                                   end
               end;
      inc(head)                                                  {队首元素出队}
    end;
  writeln('total=', tail)                                        {输出结果}
end.
```

【运行示例】

输　入	1 1 0 0 0 0	6 5 4 3 2 1	200 100 30 20 20 10
输　出	total=3	total=83	total=990

【小结】

可以想像，如果在竞赛时采用的是没有剪枝的穷举算法，当测试数据的规模很大时，由于“超时”会丢掉很多分。而用了宽度优先搜索，当数据规模很大时，用时将会大幅度减少。

穷举算法和宽度优先搜索算法，其本质是一样的，都属于搜索的范畴。但从穷举到之后的宽度优先搜索，程序的效率有了质的飞跃。而我们从前者想到后者，都是建立在对问题和算法的细致分析的基础上的。由此可见，在开始编程之前，对问题进行仔细分析，不仅是可取的，而且是必要的。凡事需“三思而后行”。

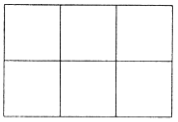
第三章　1997年复赛试题解析

一、初中组第1题

【问题描述】

设有一个n*m方格的棋盘(1≤m,n≤100),求出该棋盘中包含多少个正方形、多少个长方形(不包括正方形)。

例如:当n=3, m=2时



正方形的个数有8个;即边长为1的正方形有6个;边长为2的正方形有2个。

长方形的个数有10个;即2*1的长方形有4个;1*2的长方形有3个;3*1的长方形有2个;3*2的长方形有1个。

程序要求:输入n和m,输出正方形的个数与长方形的个数。

如上例输入:3 2

输出:8, 10

【问题分析】

首先对于n*m方格的棋盘中的任意的正方形而言,只要确定了左上角的位置和边长后,该正方形就完全确定了,如果两个正方形的左上角位置不同或左上角位置相同但边长不同,则这两个正方形显然不相同,因此只要穷举除最下面和最右边之外的所有格点(两条边的交叉点),算出以它们作为正方形的左上角位置时边长不同的正方形个数,然后累加起来即得到全部的正方形个数。同样地,以某一个格点为长方形的左上角位置的长方形(包括正方形)个数也可以类似地算出,然后累加起来即得到全部的长方形个数,再去掉前面算出的正方形个数即为程序中要求的长方形个数。

【数据结构】 (略)

【算法分析】

为了求出以某一个格点为正方形的左上角位置时的正方形个数,我们对所有的边进行编号,将从上到下的m+1条横向边定为第0行到第m行,将从左到右的n+1条纵向边定为第0列到第n列,这样以第i行与第j列的交叉点为正方形的左上角位置的正方形的边长最大只能为m-i与 n-j中的最小值,因此以第i行与第j列的交叉点为正方形的左上角位置的正方形个数为min(m-i, n-j),类似地以第i行与第j列的交叉点为长方形的左上角位置的长方形个数为(包括正方形)(m-i)*(n-j)个,因为长方形的高可以取1到m-i中的任一值,长方形的宽可以取1到n-j中的任一值。

【程序清单】

```
program c1997_1;
var i, j, m, n:longint;
    s:longint;
begin
    write('input n,m:');
    readln(n, m);
    s:=0;
    for i:=0 to n-1 do
        for j:=0 to m-1 do
            if n-i<m-j
                then s:=s+n-i
                else s:=s+m-j;
    write(s, ', ');
        s:=-s;          {预先将所有的正方形扣除}
    for i:=0 to n-1 do
        for j:=0 to m-1 do s:=s+(n-i)*(m-j);
    writeln(s)
end.
```

BASIC 语言参考程序：

```
INPUT "INPUT N, M:"; M, N
S = 0
FOR I = 0 TO N - 1
   FOR J = 0 TO M - 1
        IF N - I < M - J THEN S = S + N - I ELSE S = S + M - J
   NEXT J
NEXT I
PRINT S; ",";
S = -S
FOR I = 0 TO N - 1
    FOR J = 0 TO M - 1
        S = S + (N - I) * (M - J)
    NEXT J
NEXT I
PRINT S
```

【运行示例】(下划线表示输入)

```
input n,m:1 1
1, 0
input n,m:4 5
40, 110
input n,m:10 10
385, 2640
input n,m:30 20
4970, 92680
input n,m:50 50
42925, 1582700
```

【小结】

在算法设计后,编程时要注意数据类型的设定。由于已知:1≤m,n≤100,所以固定某个格点为左上角后,包含该格点的长方形数目不超过 m * n 个,全部的长方形总数不超过 m * n * m * n≤100000000 个,可见用长整型变量来编程是不会溢出范围的,所以程序中使用了长整型(longint)。程序编好后必须进行测试,自己设计测试数据时一般要注意如下几点:

(1) 测试数据的结果应是自己能把握正确性的,一般说来题例就可作为这样的测试数据。

(2) 测试数据应覆盖所求问题的范围;该问题若使用整型,那么测试数据取 n、m 在 15 以内还可能是正确的,取 20 以上就可能出不了结果。

(3) 对于特殊的边界数据要尽可能验证,例如该问题 n = m = 1 的情况;n = m = 100 的情况。

二、初中组第2题

【问题描述】

将 1,2,…,9 共 9 个数排成下列形态的三角形:其中:a ~ i 分别表示 1,2,…,9 中的一个数字,并要求同时满足下列条件:

```
        a
      b   c
    d       e
  f   g   h   i
```

(1) a < f < i;

(2) b < d, g < h, c < e;

(3) a + b + d + f = f + g + h + i = i + e + c + a = P。

程序要求:根据输入的边长之和 P,输出所有满足上述条件的三角形的个数以及其中的一

种方案。

【问题分析】

本题明显是穷举问题，只要根据题目条件逐个穷举三角形每条边上的某些变量即可到问题的解。

【数据结构】

为了判断某个数字是否已被赋予某个变量，程序中用布尔型数组 used 记录下数字的使用情况，若 used[d] = true，则表示数字 d 已被赋予某个变量。

【算法分析】

由于每条边上的和为定值 P，因此每条边上均可少穷举一个变量，余下的变量的值可以通过计算得出。另外，三个角上的变量被两条边公用，因此三个角上的变量应首先被穷举，并且由三条边上的总和为 3P 可推得：(a+b+d+f)+(f+g+h+i)+(i+e+c+a)=3P，另外已知所有变量之和为定值 45，所以三个角上的变量 a、f、i 之和为 3P-45，其中变量 a 的值不超过 p-16，变量 f 的值不超过(3*p-45-a-1) div 2。总共需要穷举的变量个数为五个，它们是 a, f, b, g, c。

【程序清单】

```
program c1997_2;
var a, b, c, d, e, f, g, h, i, p, bd, gh, ce, total:longint;
    used:array[1..100] of boolean;
    first:boolean;
procedure print;
begin
      writeln(a:40);
      writeln(b:35, c:10);
      writeln(d:30, e:20);
      writeln(f:25, g:8, h:14,i:8);
      first:=false;
end;
begin
  write('input p:');
  readln(p);
  if (p<17) or (p>23) then
     begin writeln('total=',0); halt end;
  total:=0;
  for i:=1 to 9 do used[i]:=false;
  first:=true;
  for a:=1 to p-16 do
   for f:=a+1 to (3*p-45-a-1) div 2  do
```

```
begin
used[a]:=true;
 used[f]:=true;
 i:=3*p-45-a-f;
 if (i>f) and (i<10) then
 begin
 used[i]:=true;
   bd:=p-a-f;
   for b:=1 to (bd-1) div 2 do
     if (b<10) and not(used[b]) then
     begin
     used[b]:=true;
       d:=bd-b;
       if (d<10) and not(used[d]) then
       begin
       used[d]:=true;
         gh:=p-f-i;
         for g:=1 to (gh-1) div 2 do
           if (g<10) and not(used[g]) then
           begin
           used[g]:=true;
             h:=gh-g;
             if (h<10) and not(used[h]) then
             begin
             used[h]:=true;
             ce:=p-a-i;
               for c:=1 to (ce-1) div 2 do
                 if (c<10) and not(used[c]) then
                 begin
                 used[c]:=true;
                   e:=ce-c;
                   if (e<10) and not(used[e]) then
                     begin
                     total:=total+1;
                     if first then print
                     end;
                   used[c]:=false
```

```
                    end;
                  used[h]:=false
                  end;
                used[g]:=false
                end;
              used[d]:=false
              end;
            used[b]:=false
            end;
          used[i]:=false
          end;
        used[a]:=false;
        used[f]:=false;
    end;
    writeln('total=',total)
end.
```

BASIC 语言参考程序：

```
DIM USED(100)
INPUT "INPUT P:"; P
IF P < 17 OR P > 23 THEN
    PRINT "TOTAL = "; 0
    END
END IF
TOTAL = 0
FIRST = 1
FOR I = 1 TO 9
    USED(I) = 0
NEXT I
FOR A = 1 TO P - 16
 FOR F = A + 1 TO (3 * P - 45 - A - 1) \ 2
  USED(A) = 1
  USED(F) = 1
  I = 3 * P - 45 - A - F
  IF I < 10 AND I > F AND USED(I) = 0 THEN
     USED(I) = 1
```

```
BD = P - A - F
FOR B = 1 TO (BD - 1) \ 2
  IF B < 10 AND USED(B) = 0 THEN
     USED(B) = 1
     D = BD - B
     IF D < 10 AND USED(D) = 0 THEN
       USED(D) = 1
       GH = P - F - I
       FOR G = 1 TO (GH - 1) \ 2
         IF (G < 10) AND USED(G) = 0 THEN
           USED(G) = 1
           H = GH - G
           IF (H < 10) AND USED(H) = 0 THEN
             USED(H) = 1
             CE = P - A - I
             FOR C = 1 TO (CE - 1) \ 2
               IF (C < 10) AND USED(C) = 0 THEN
                 USED(C) = 1
                 E = CE - C
                 IF (E < 10) AND USED(E) = 0 THEN
                    TOTAL = TOTAL + 1
                    IF FIRST THEN
                       PRINT "          "; A
                       PRINT "      "; B; "      "; C
                       PRINT "   "; D; "          "; E
                       PRINT F; "   "; G; "      "; H; "      "; I
                       FIRST = 0
                    END IF
                 END IF
                 USED(C) = 0
               END IF
             NEXT C
             USED(H) = 0
           END IF
           USED(G) = 0
         END IF
       NEXT G
```

```
            USED(D) = 0
          END IF
          USED(B) = 0
        END IF
      NEXT B
      USED(I) = 0
      END IF
      USED(F) = 0
      USED(A) = 0
    NEXT F
    NEXT A
    PRINT "TOTAL = "; TOTAL
```

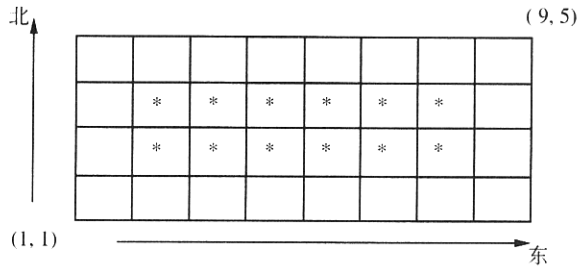
【运行示例】(下划线表示输入)

```
input p:23
                    7
                2       3
            6               4
        8       1       5       9
total = 2

input p:18
total = 0

input p:19
                    1
                5       3
            9               8
        4       2       6       7
total = 4
input p:20
                    1
                6       3
            8               7
        5       2       4       9
total = 6
```

三、初中组第3题

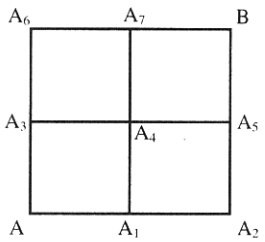
【问题描述】

设有一个N*M(1≤ N≤50, 1≤ M≤ 50)的街道(如下所示):



规定行人从A(1, 1)出发,在街道上只能向东或北方向行走。

如下为N=3,M=3的街道图,从A出发到达B共有6条可供行走的路径:



1. A-A1-A2-A5-B
2. A-A1-A4-A5-B
3. A-A1-A4-A7-B
4. A-A3-A4-A5-B
5. A-A3-A4-A7-B
6. A-A3-A6-A7-B

若在N*M的街道中,设置一个矩形障碍区域(包括围住该区域的街道)不让行人通行,如前面图中用"*"表示的部分。此矩形障碍区域用2对顶点坐标给出,前图中的2对顶点坐标为:(2, 2),(8, 4),此时从A出发到达B的路径仅有两条。

程序要求

任务一:给出N,M后,求出所有从A出发到达B的路径的条数。

任务二:给出N,M,同时再给出此街道中的矩形障碍区域的2对顶点坐标(X1, Y1),(X2, Y2),然后求出此种情况下所有从A出发到达B的路径的条数。

【问题分析】

由于在街道上只能向东或北方向行走,因此要想到达坐标为(i, j)的顶点的话,必定要经过坐标为(i-1, j)的顶点或坐标为(i, j-1)的顶点,假设从起始顶点到达坐标为(i, j)的顶点的路径总数为a[i, j],则a[i, j] = a[i-1, j] + a[i, j-1],因此我们可以使用逐行递推的方法来求出从起始顶点到达任意一个顶点的路径总数,即使有障碍区域,这一方法也完全适

用，只要将从起始顶点到达障碍区域中的顶点的路径总数置为 0 即可。

【数据结构】

由于只要计算从起始顶点到达终点的路径总数，而使用逐行递推的方法只涉及两行顶点，因此只需要记录两行数据，而计算当前顶点的路径总数时只用到了同一行前一个顶点的路径总数和前一行同一列的顶点的路径总数，因此可以只使用一个一维数组记录当前行中各个顶点的路径总数，再用迭代的方法来计算下一行中各个顶点的路径总数，具体的迭代式为 a[j] = a[j - 1] + a[j]，其中 a[j - 1]表示下一行中第 j - 1 列的顶点的路径总数，右式中的 a[j] 表示当前行中第 j 列的顶点的路径总数，左式中的 a[j] 表示下一行中第 j 列的顶点的路径总数。另外当 N 与 M 较大时，计算出的路径总数将变得很大，这就要用一个数组来存储一个数，即高精度数，因此程序中使用一个二维数组来记录到达当前行中各个顶点的路径总数，数组的第一下标记录当前行顶点的列号，第二下标表示高精度数中的位，下标 1 对应个位。具体说明如下：

```
const maxn = 50;   len = 30;
type arraytype = array [1..maxn,1..len + 1] of integer;
```

【算法分析】

程序采用逐行递推和迭代法依次计算从起点到各顶点的路径总数，从而最终算出到达终点的路径总数，具体算法如下：

```
置数组 a 第一行第一列元素为 1;
for i:=2 to m do  {从第二行推到第 m 行}
    for j:=2 to n do {迭代计算第 i 行的第 2 列到第 n 行顶点的路径总数}
        若有障碍区域且顶点(i, j)在障碍区域内，则置 a[j]为 0,
        否则 a[j]:=a[j] + a[j - 1];
```

再将其中的 a[j]化为一个二维数组，加进高精度加法运算即得到完整的算法，细节详见程序清单。

【程序清单】

```
program c1997_3;
const maxn = 50; len = 30;
type arraytype = array [1..maxn,1..len + 1] of integer;
var i, j, k, m, n, x1, y1, x2, y2:longint;
    ch:char;
    a:arraytype;
begin
    write('input n,m:');
    readln(n, m);
    write('是否有障碍区域(y/n):');
    readln(ch);
    if (ch = 'Y') or (ch = 'y') then
        begin
```

```
            write('input x1, y1, x2, y2:');
            readln(x1, y1, x2, y2)
        end;
    fillchar(a,sizeof(a), 0);
    for i:=1 to m do a[i, 1]:=1;
    for i:=2 to m do
        for j:=2 to n do
        begin
            if (ch='Y') or (ch='y') and (i in [x1..x2]) and (j in [y1..y2])
              then for k:=1 to len do a[j, k]:=0
              else for k:=1 to len do
                    begin
                        a[j, k]:=a[j, k]+a[j-1, k];
                        if a[j, k]>=10 then {有进位,高一位加 1}
                        begin
                            a[j, k+1]:=a[j, k+1]+1;
                            a[j, k]:=a[j, k]-10
                        end
                    end
        end;
    k:=len;
    while (k>1) and (a[n, k]=0) do k:=k-1;
    for i:=k downto 1 do write(a[n, i]);
    writeln
end.
```

BASIC 语言参考程序:

```
    MAXN = 50
    DIM A(MAXN, 31)
    INPUT "INPUT N,M:"; N, M
    INPUT "     (Y/N)"; CH $
    IF CH $ = "Y" OR CH $ = "Y" THEN
        INPUT "INPUT X1, Y1, X2, Y2:"; X1, Y1, X2, Y2
    END IF
    FOR I = 1 TO MAXN
        FOR J = 1 TO 30
            A(I, J) = 0
```

```
        NEXT J
    NEXT I
    FOR I = 1 TO N
        A(I, 1) = 1
    NEXT I
    FOR I = 2 TO M
        FOR J = 2 TO N
            IF CH $ < > "Y" AND CH $ < > "Y" OR I < X1 OR I > X2 OR J <
Y1 OR J > Y2 THEN
                FOR K = 1 TO 30
                    A(J, K) = A(J, K) + A(J - 1, K)
                NEXT K
                FOR K = 1 TO 30
                    IF A(J, K) > = 10 THEN
                        A(J, K + 1) = A(J, K + 1) + 1
                        A(J, K) = A(J, K) - 10
                    END IF
                NEXT K
            ELSE
                FOR K = 1 TO 30
                    A(J, K) = 0
                NEXT K
            END IF
        NEXT J
    NEXT I
    K = 30
    WHILE k > 1 AND A(N, K) = 0
        K = K - 1
    WEND
    FOR I = k TO 1 STEP -1
        PRINT CHR $(A(N, I) + 48);
    NEXT I
    PRINT
```

【运行示例】(下划线表示输入)

```
input n,m:2 2
是否有障碍区域(y/n):n
2
```

```
input n,m:10 10
是否有障碍区域(y/n):n
48620
input n,m:50 50
是否有障碍区域(y/n):n
254776122589808569027304286OO
input n,m:30 40
是否有障碍区域(y/n):y
input x1, y1, x2, y2:5 5 15 15
118200946737728400
input n, m:50 50
是否有障碍区域(y/n):y
input x1, y1, x2, y2:2 2 49 49
2
input n,m:50 50
是否有障碍区域(y/n):y
input x1,y1,x2,y2:2 2 7 5
873586936014973809750037800
```

【小结】

任务一要从 a(1, 1)到达 b(n, m),不论哪条路径总是有 n-1 格横向,m-1 格纵向,这正好是从 m+n-2 个元素中取 n-1 个组合。求组合数可使用现成的公式来计算,即 $C_{m+n-2}^{n-1}=\frac{(m+n-2)!}{(m-1)!\ (n-1)!}$,这样就需要专门编写高精度运算的过程,而高精度乘除运算算法比较麻烦,并且任务二在处理时很难得出简明的组合公式,因此在解本题时采用递推的方法是比较合适的,同时该算法在实现时也比较容易。

四、高中组第1题:

【问题描述】

在一个 N*N 的棋盘上(1≤N≤10),填入 1, 2, …, N*N 共 N*N 个数,使得任意两个相邻的数之和为素数。

例如:当 N=2 时,有:

1	2
3	4

当 N=4 时,一种可以填写的方案如下:在这里我们约定:左上角的格子里必须填数字 1。

1	2	11	12
16	15	8	5
13	4	9	14
6	7	10	3

程序要求：

输入 N，

输出：如有多种解，则输出第一行、第一列之和为最小的排列方案；若无解，则输出“NO!”。

【问题分析】

由于质数的变化是没有规律的，本问题除了穷举之外别无它法，因为 N 的值是从 1 到 10 之间的任意一个自然数，所以穷举时必需用递归或回溯的方法来实现，具体来说，就是将所有的方格编一个次序，为了尽快得到输出要求的结果，填数的顺序应是先填第一行再填第一列，选择填入格子中的数应从小到大依次选择，首先选一个 2 到 N * N 之间的自然数填入第一行的第二列方格，要求填入的自然数与已填数的相邻方格中的数之和为质数，然后用相同的方法填第一行的第三列至第 N 列的方格，填完第一行之后，再依次填第一列中从第二行至第 N 行的方格，往后再填第二行中第二列至第 N 列的方格及第二列中第三行至第 N 行的方格，……直到右下角最后一个方格填好数为止，此时将填好的数按 N 行 N 列对齐的格式输出即可。在填数的过程中，要确保填入的数是没有用过的，若当前方格选不到可填之数时，要回溯到上一格，改变上一格所填之数后继续向前填数。

【数据结构】

由于填入棋盘方格中的最大数是 n^2，所以相邻（横或竖）两个格子中的数之和一定小于 $2n^2$。预先将 $2n^2$ 之内的素数一次算出，保存起来可以减少判定相邻两个格子中的数之和是否为素数的工作量。该问题中 $n \leqslant 10$，所以可将 200 以内的素数一次算出保存在数组 P 中（共 46 个）：2　3　5　7　11　13　17　19　23　29　31　37　41　43　47　53　59　61　67　71　73　79　83　89　97　101　103　107　109　113　127　131　137　139　149　151　157　163　167　173　179　181　191　193　197　199

这样判定给定的和是否为素数只要一个查表的过程即可实现。为了选择 $1 \sim n^2$ 中的一个数填入棋盘格子，可采用布尔数组 used 来标记已用过的数（已填入棋盘格子的数），used[i] = true，表示 i 已填入棋盘中的某个格子，used[i] = false 则表示 i 是可填入棋盘格子的数（$1 \leqslant i \leqslant n^2$）。A 数组用于记录棋盘格子填数的状态，A[1, 1]固定为 1。

【算法分析】

程序中使用递归算法，算法用一个递归过程实现，其中形式参数 x 与 y 表示当前格子所在的行与列，dep 表示递归的深度，即当前已填过数的格子数。具体请看程序清单和程序注释。

【程序清单】

```
program g1997_1;
const maxn = 10;
type arr1 = array [1..2 * maxn * maxn] of boolean;
var i, j, k, m, n, x, y:integer;
```

```
    p:arr1;
    a:array [1..maxn,1..maxn] of integer;
    used:array[1..maxn * maxn] of boolean;

function ok(x, y, k:integer):boolean;
begin
     ok:=true;
     if x>1 then if not(p[a[x-1, y]+k]) then ok:=false;
     if y>1 then if not(p[a[x, y-1]+k]) then ok:=false;
end;
procedure print;
  var i, j:integer;
begin
     for i:=1 to n do
     begin
          for j:=1 to n do write(a[i, j]:4);
          writeln
     end;
     halt
end;

procedure try(x, y, dep:integer);
var i:integer;
begin
     if dep=n * n                                    {已填好所有格子}
        then print
        else for i:=1 to n * n do
                 if not(used[i]) and ok(x, y, i)     {i 未用过且可填入格子(x, y)中}
                    then
                    begin
                         a[x, y]:=i;
                         used[i]:=true;
                         if y=n                      {当前行填完,转下一行 x 列}
                            then try(x+1, x, dep+1)
                            else if x=n              {当前列填完,转下一列 y+1 行}
                                    then try(y+1, y+1, dep+1)
                                    else if x<=y
```

```
                                        then try(x, y+1, dep+1)
                                                              {填本行下一列}
                                        else try(x+1, y, dep+1);
                                                              {填本列下一行}
                            used[i]:=false
                        end
end;
begin
    write('Input n:');
    readln(n);
    if n=1 then begin writeln('NO!'); halt end;
    for i:=1 to 2*n*n do p[i]:=true;
    for i:=2 to n*3 div 2 do                                  {筛选法求质数表}
    begin
        j:=i*2;
        while j<=2*n*n do
        begin p[j]:=false;j:=j+i end
    end;
    a[1,1]:=1;
    for i:=1 to n*n do used[i]:=false;
    used[1]:=true;
    try(1,2,1);
    writeln('NO!');
end.
```

【运行示例】（下划线表示输入）

```
Input n:1
NO!

Input n:2
                                        1    2
                                        4    3

Input n:3
NO!

Input n:4
```

```
 1   2  11  12
 4   9   8   5
 7  10   3  14
 6  13  16  15
```

```
Input n:5
 1   2   3   4   7
 6   5  14  15  16
13  24  23   8  21
10  19  18  11  20
 9  22  25  12  17
```

【小结】

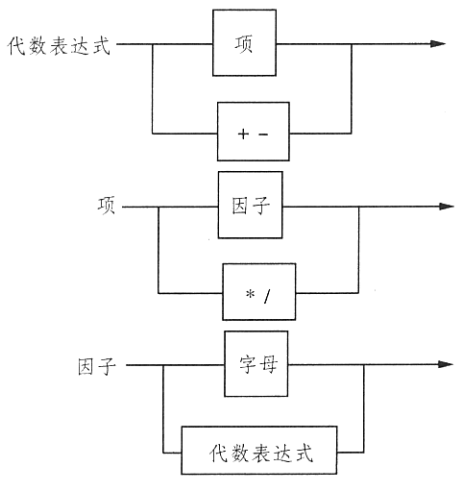
在选择数据填入格子时,并没有必要都依据素数表,因为素数不可能是二个奇数之和也不可能是二个偶数之和,只可能是一个奇数加一个偶数之和。因此棋盘格子中所填之数无论是行还是列都必然是奇偶或偶奇相邻的。

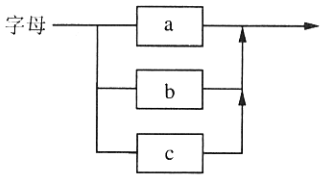
如果问题中去掉 a[1, 1] =1 的约定,那么还可以发现 n 为奇数时,a[1, 1]中必须填奇数才有可能有解,否则一定无解。什么原因,能说明清楚吗?(请从 n 确定后,n^2个数中奇数和偶数的个数进行思考!)

五、高中组第2题

【问题描述】

代数表达式的定义如下:





此处字母表示 a～c 之间的小写字母，例如，下面的式子是合法的代数表达式：

a；

a＋b＊(a＋c)；

a＊a/(b＋c)

下面的式子是不合法的代数表达式：

ab；

a＋a＊/(b＋c)；

程序要求：

输入：输入一个字符串，以“；”结束，(“；”本身不是代数表达式中字符，仅作为结束)；

输出：若表达式正确，则输出“OK”；

若表达式不正确，则输出“ERROR”，及错误类型。

错误类型 1 为运算符号缺少或过多，错误类型 2 为括号不配对，其它错误均属于错误类型 3。例如：

输入 a＋(b)；

输出　OK

输入 a＋(b＋c＊a；

输出 ERROR 2

【问题分析】

本题主要测试选手的审题能力和对表达式定义(题目给定)的理解能力。在理解定义的基础上，要求能归纳出从左到右逐个扫描表达式中的字符后，判定表达式是否符合定义给出的规则，即给定的表达式对还是错？进一步还要指出错的情况下是哪一类错？

从定义可知，输入的表达式字符串，除结束标记“；”字符外，只可能出现如下九种字符：“a”，“b”，“c”(小写字母)，“＋”，“－”，“＊”，“/”，和“(”，“)”，如果在读字符的过程中，遇到这 9 个字符以外的字符，立即就可以报告第一类错误信息。

对第二类错误而言，我们只要在对表达式的扫描过程中，对左右括号作一特殊统计处理即可，初始时置计数变量为 1，从左向右扫描表达式，遇到左括号时，计数变量加 1，遇到右括号时，则计数变量减 1，如果在扫描过程中，计数变量的值出现负数或扫描结束时计数变量的不为零，则报告第二类错误信息。

本题中最困难的是检查第三类错误，一般说来我们通常可用表格来判断表达式有无第三类错误，表中打“√”的方格表示同一列中的字符可以跟在同一行中的字符之后，打“×”的方格则表示同一列中的字符不可以跟在同一行中的字符之后。如字符“＊”可以跟在字符“a”之后，但字符“(”则不可以跟在字符“a”之后。

	a	b	c	+	−	*	/	(	)
a	×	×	×	√	√	√	√	×	√
b	×	×	×	√	√	√	√	×	√
c	×	×	×	√	√	√	√	×	√
+	√	√	√	×	×	×	×	√	×
−	√	√	√	×	×	×	×	√	×
*	√	√	√	×	×	×	×	√	×
/	√	√	√	×	×	×	×	√	×
(	√	√	√	×	×	×	×	√	×
)	×	×	×	√	√	√	√	×	√

有了这张表格之后,要检查出第三类错误就十分简单了。需要注意的是表达式的第一个字符不能为“+”、“-”、“*”、“/”和“)”,最后一个字符不能为“+”、“-”、“*”、“/”和“(”。

【数据结构】

为了方便判断输入的表达式(用一个字符串型变量 ex 存储)字符串是否包含非法字符,程序用一个常量字符集,集合中的元素为表达式中允许出现的字符:“a”,“b”,“c”(小写字母),“+”,“-”,“*”,“/”,和“(”,“)”。

【算法分析】

程序在读入一个表达式后,首先检查其中是否包含第一类错误,然后检查其中是否包含第二类错误,最后检查其中是否包含第三类错误,只要在检查过程中发现了某一类错误,程序就报告相应的错误信息,并结束程序运行。若未查出任何错误,则输出“OK!”。程序中用一个 case 语句实现第三类错误的查表功能,具体细节请看程序清单。

【程序清单】

```
program g1997_2;
const charset:set of char = ['a', 'b', 'c', '+', '-', '*', '/', '(', ')'];
var i, left:integer;
    ex:string;
procedure error(code:integer);
begin
    writeln('error ', code);
    halt
end;
```

```
begin
    write('input a expression:');
    readln(ex);
    for i:=1 to length(ex) do
        if not(ex[i] in charset) then error(1);
    left:=0;
    i:=1;
    while (left>=0) and (i<=length(ex)) do
    begin
        if ex[i]='('
            then left:=left+1
            else if ex[i]=')' then left:=left-1;
        i:=i+1
    end;
    if left<>0 then error(2);
    if ex[1] in ['+','-','*','/'] then error(3);
    if ex[length(ex)] in ['+','-','*','/'] then error(3);
    for i:=2 to length(ex)-1 do
        case ex[i-1] of
          '(':if ex[i] in [')','+','-','*','/'] then error(3);
          ')':if ex[i] in ['(','a','b','c'] then error(3);
          '+','-','*','/':if ex[i] in [')','+','-','*','/'] then error(3);
          'a'..'c':if ex[i] in ['(','a','b','c'] then error(3);
        end;
    writeln('OK')
end.
```

【运行示例】（下划线表示输入）

```
input a expression:a+x
error 1
input a expression:(((b+c)))
OK
input a expression:a+b(c+a)
error 3
input a expression:(a+(b+c)
error 2
input a expression:a+)b+c(
error 2
```

【小结】

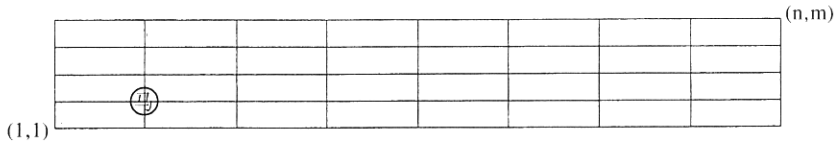
本题所用的算法仅适用于表达式的判错,对表达式的一般处理(如计算及前后缀转换等)则必须使用堆栈处理。

六、高中组第3题

【问题描述】

骑士游历:

设有一个 n * m 的棋盘(2≤n≤50, 2≤m≤50),如下图,在棋盘上左下角有一个中国象棋马。



马走的规则为:①马走日字,从(1, 1)可走到(2, 3)和(3, 2); ② 马只能向右走。

任务1:当 n, m 输入之后,找出一条从左下角到右上角的路径。

例如:n =4,m =4 时,输出路径时格式如下:(1, 1)→(2, 3)→(4, 4)。

若不存在路径,则输出'NO'。

任务2:当 n,m 给出之后,同时给出马起点的位置和终点的位置,试找出从起点到终点的所有路径的数目。

例如:(n =10, m =10),(1, 5)(起点),(3, 5)(终点)

输出:2

输入格式:n, m, x1, y1, x2, y2 (x1, y1 表示起点坐标,x2, y2 表示终点坐标)

输出格式:路径数目(若不存在从起点到终点的路径,输出 0)

【问题分析】

对于任务一,我们可以用宽度优先搜索算法将马从起点出发所能到达的每一点,按到达的最少步数从小到大的次序生成到一个队列中,直到终点进入队列或没有任何新的点可到达为止。若终点已经进入队列,则往回倒推输出问题的解。如 n =10, m =10,起点为(1, 5),终点为(3, 5),则初始时起点(1, 5)进队,考虑从起点(1, 5)出发一步就能到达的点,根据规则可走到(2, 3)、(2, 7)、(3, 2)和(3, 6)四点,依次让这四个点进队,接下去再考虑从起点(1,5)出发两步就能到达的点,显然这些点必定要经过(2, 3)、(2, 7)、(3, 2)及(3, 6)四点之一方能到达,所以只要将从(2, 3)、(2, 7)、(3, 2)、(3, 6)四点出发一步能达的点放进队列即可得到从起点出发两步所能到达的全部点,实际上从(2, 3) 出发一步就到达了终点(3,5),此时即可往回倒推得到它是从上一点(2, 3)跳过来的,而(2, 3)又来自起点(1, 5),从而得到问题的解为(1, 5)→(2, 3)→(3, 5)。对于任务二,完全可以参照初中组试题三的算法设计,只要将递推公式略微修正一下即可。因为马要到达当前点(x, y),根据规则它只能从(x - 1, y -

2)、(x-2, y-1)、(x-1, y+2)、(x-2, y+1)四点到达,设到达当前点(x, y)的路径总数为path(x, y),则根据加法原理有 path[x, y] = path[x-1, y-2]+path[x-2, y-1]+path[x-1, y+2]+path[x-2, y+1],初始时可将除起点之外的所有点的路径总数置为0,将起点的路径总数置为1,则根据上述递推公式就不难推出从起点到终点的路径总数了。

【数据结构】

对于任务一,队列中的每一个元素表示的是棋盘上的某一点,同时还要记录该点的上一点,因此程序用一维数组 q 表示这个队列,数组 q 的下标表示队列元素的序号,数组 q 的每个元素则是一个包含三个元素的一维数组,分别记录棋盘上的一点的横坐标、纵坐标及其上一点在队列中的序号,对问题分析中的例子而言,数组 q 的第一元素为(1, 5, 0),其中(1, 5)为起点坐标,0 表示该点没有上一点,数组 q 的随后四个依次元素为(2, 3, 1)、(2, 7, 1)、(3, 2, 1)和(3, 6, 1),其中的 1 表示它们均来自于 q[1]点,最后一个元素为(3, 5, 2)。由于棋盘上的元素最多进队一次,数组 q 可说明为 q:array[1..maxn * maxn, 1..3] of integer,其中 maxn 为棋盘的最大尺寸。

对于任务二,当棋盘较大时,从起点到达终点的路径总数显然将会超出长整型的范围,因此记录路径总数时要使用高精度数即一维数组来表示,而棋盘上最多可以有 50 * 50 = 2500 个点,要记录下每一点的路径总数就要用 2500 个一维数组,这在存储空间上将是十分困难的,因此我们要考虑能不能少记录一些数据,使得存储空间不至于那么紧张,考虑到递推公式 path[x, y] = path[x-1, y-2]+path[x-2, y-1]+path[x-1, y+2]+path[x-2, y+1]中求当前点的路径总数时只与其前面两列的数据有关,这就启发我们使用逐列递推的方法,这样就只需要记录下三列数据,因此可以只使用一个二维数组记录当前列及其前两列中所有点的路径总数,再用迭代的方法向右逐列求出每一列所有点的路径总数,直到终点所在列为止。

综上所述,任务二的数据结构选择如下:

```
const maxn = 50; len = 30;
type arraytype = array[ -1..maxn+2, 1..len+1] of integer;
var  path:array[1..3] of arraytype;
```

其中数组 path 的第三维下标表示高精度数中的位,下标 1 对应个位;第一维下标表示列,下标值 3 表示当前列,1、2 表示前两列;第二维下标表示点的纵坐标,即点所在的行。

【算法分析】

任务一是一个典型的队列生成算法,程序中用过程 task1 实现这一算法,其中 head 指向队头元素,tail 则指向队尾,每次将队头元素表示的点取出,然后将其一步能达且尚未进队的点加在队尾,处理完当前队头元素后将 head 加 1 重复这一步骤,直至终点进入队列或没有任何新的点可到达为止。

在处理任务二时,在棋盘的上下边各加了两行,并将到达这些棋盘之外的点的路径总数置为0,这样在递推过程中就不再需要判断点是否出界了,初始时将起点所在列及其前一列作为 path[2]与 path[1]置初值0(除起始点之外),迭代时每次用上述递推公式先算 path[3],然后将 path[2]赋给 path[1],将 path[3] 赋给 path[2],再将 path[3]清零,然后再用递推公式计算 path[3],直到终点所在列算出为止。细节详见程序清单。

【程序清单】

```
program G1997_3;
const maxn=50;
      dx:array[1..4] of integer=(1, 1, 2, 2);
      dy:array[1..4] of integer=(2, -2, 1, -1);
var select:integer;
procedure task1;
var i, j, m, n, head, tail, x, y:integer;
    found:boolean;
    q:array[1..maxn*maxn, 1..3] of integer;
procedure print;
var i, step:integer;
    path: array[1..2*maxn, 1..2] of integer;
begin
     i:=tail; step:=0;
     while i>0 do
     begin
          step:=step+1;
          path[step, 1]:=q[i, 1];
          path[step, 2]:=q[i, 2];
          i:=q[i, 3];
     end;
     for i:=step downto 2 do write('(', path[i, 1], ', ', path[i, 2], ')->');
     writeln('(', path[1, 1], ', ', path[1, 2], ')');
     halt
end;
begin
     write('input n, m:');
     readln(n, m);
     q[1, 1]:=1; q[1, 2]:=1; q[1, 3]:=0;
     head:=1; tail:=2;
     while head<tail do
     begin
          for i:=1 to 4 do
          begin
               x:=q[head, 1]+dx[i];
               y:=q[head, 2]+dy[i];
               if (x>0) and (x<=n) and (y>0) and (y<=m) then
```

```
                    begin
                        found:=false;
                        for j:=1 to tail-1 do
                            if (q[j,1]=x) and (q[j,2]=y)
                                then found:=true;
                        if not(found) then
                            begin
                                q[tail,1]:=x;
                                q[tail,2]:=y;
                                q[tail,3]:=head;
                                if (x=n) and (y=m) then print;
                                tail:=tail+1;
                            end
                    end
            end;
            head:=head+1
        end;
        writeln('NO')
end;
procedure task2;
const len=20;
type arraytype=array[-1..maxn+2,1..len+1] of integer;
var i,j,m,n,x,y,x1,y1,x2,y2:integer;
    path:array[1..3] of arraytype;
    temp:arraytype;
begin
    write('Input n,m,x1,y1,x2,y2:');
    readln(n,m,x1,y1,x2,y2);
    fillchar(path,sizeof(path),0);
    path[2,y1,1]:=1;
    for x:=x1+1 to x2 do
    begin
        for y:=1 to m do
        begin
            for i:=1 to 4 do
                for j:=1 to len do
                    path[3,y,j]:=path[3,y,j]+path[3-dx[i],y-dy[i],j];
```

```
                for j:=1 to len do
                begin
                    path[3,y,j+1]:=path[3,y,j+1]+path[3,y,j] div 10;
                    path[3,y,j]:=path[3,y,j] mod 10
                end
            end;
            path[1]:=path[2];
            path[2]:=path[3];
            fillchar(path[3],sizeof(path[3]),0);
        end;
        j:=len;
        while (j>1) and (path[2,y1,j]=0) do j:=j-1;
        for i:=j downto 1 do write(path[2,y1,i]);
        writeln
end;

begin
        writeln('   1 - - - task 1');
        writeln('   2 - - - task 2');
        write('   please input your select(1 or 2):');
        readln(select);
        if select=1 then task1 else task2
end.
```

【运行示例】（下划线表示输入）

```
    1 - - - task 1
    2 - - - task 2
    please input your select(1 or 2):1
input n,m:9 5
(1,1)->(3,2)->(5,3)->(7,4)->(9,5)
    1 - - - task 1
    2 - - - task 2
    please input your select(1 or 2):1
input n,m:3 3
NO
    1 - - - task 1
    2 - - - task 2
    please input your select(1 or 2):2
```

```
input n, m, x1, y1, x2, y2:30  30  1  15  3  15
2
    1 - - - task 1
    2 - - - task 2
    please input your select(1 or 2):2
input n, m, x1, y1, x2, y2:30  30  1  15  5  15
8
    1 - - - task 1
    2 - - - task 2
    please input your select(1 or 2):2
input n, m, x1, y1, x2, y2:30  30  1  15  10  15
460
    1 - - - task 1
    2 - - - task 2
    please input your select(1 or 2):2
input n, m, x1, y1, x2, y2:50  50  1  25  40  25
3323759302857476
```

第四章　1998年复赛试题解析

一、初中组第1题

【问题描述】

将1，2，…，9共9个数分成三组，分别组成三个三位数，且使这三个三位数构成1:2:3的比例，试求出所有满足条件的三个三位数。

例如：三个三位数192，384，576满足以上条件。

【问题分析】

本题只要穷举123到329之间的所有三位数i，如果i、2*i、3*i这三个数中所包含的9个数字刚好是1到9则输出i、2*i、3*i。

【数据结构】

为了记录1到9这9个数字在i、2*i、3*i中是否出现过，程序中用一个一维数组a。a[d]=1表示数字d在i、2*i、3*i这三个数中出现过，a[d]=0则表示未出现。

【算法分析】

对穷举的三位数i，若i、2*i、3*i中包含1到9全部九个数字，则输出i、2*i、3*i。

【程序清单】

```
program c1998_1;
var i, j, s:integer;
    a:array[1..9] of integer;
begin
    for i:=123 to 329 do
    begin
        for j:=1 to 9 do a[j]:=0;
        a[i div 100]:=1;
        a[i div 10 mod 10]:=1;
        a[i mod 10]:=1;
        a[2*i div 100]:=1;
        a[2*i div 10 mod 10]:=1;
        a[2*i mod 10]:=1;
        a[3*i div 100]:=1;
        a[3*i div 10 mod 10]:=1;
        a[3*i mod 10]:=1;
```

```
        s:=0;
        for j:=1 to 9 do s:=s+a[j];
        if s=9 then writeln(i:10, 2*i:10, 3*i:10)
    end
end.
```

BASIC 语言参考程序：

```
DIM A(10)
FOR I = 123 TO 329
    FOR J = 1 TO 9
        A(J) = 0
    NEXT J
    A(I \ 100) = 1
    A(I \ 10 MOD 10) = 1
    A(I MOD 10) = 1
    A(2 * I \ 100) = 1
    A(2 * I \ 10 MOD 10) = 1
    A(2 * I MOD 10) = 1
    A(3 * I \ 100) = 1
    A(3 * I \ 10 MOD 10) = 1
    A(3 * I MOD 10) = 1
    S = 0
    FOR J = 1 TO 9
        S = S + A(J)
    NEXT J
    IF S = 9 THEN PRINT I, 2 * I, 3 * I
NEXT I
```

【运行示例】

```
192     384     576
219     438     657
273     546     819
327     654     981
```

二、初中组第2题

【问题描述】

用高精度计算出S=1! +2! +3! +…+n! (n≤50),其中"!"表示阶乘,例如:5! = 5*4*3*2*1。输入正整数N,输出计算结果S。

【问题分析】

这是一个典型的高精度加法和乘法问题,计算n!要用高精度乘法,计算S则用高精度加法,另外计算1!, 2!, 3!, …, n!时可以使用递推的方法来提高算法的效率,如当前已算出了i!的值,则将它再乘上i+1即得到(i+1)!的值。

【数据结构】

程序用一维数组sum存储s的值,用一维数组fac存储i!的值。两个数组均表示高精度数,下标1对应个位。

【算法分析】

程序中从i!的值计算(i+1)!的值时使用了迭代的方法,用语句for j:=1 to maxlen do fac[j]:=fac[j]*i来实现。具体的高精度加法和乘法算法见程序清单及注释。

【程序清单】

```
program c1998_2;
const maxlen=100;
var i, j, n:integer;
    sum, fac:array[1..maxlen+1] of integer;
begin
    write('input n:');
    readln(n);
    for i:=1 to maxlen do sum[i]:=0;
    for i:=1 to maxlen do fac[i]:=0;
    fac[1]:=1;
    for i:=1 to n do
    begin
        for j:=1 to maxlen do fac[j]:=fac[j]*i;
        for j:=1 to maxlen do
        begin
            fac[j+1]:=fac[j+1]+fac[j] div 10;
            fac[j]:=fac[j] mod 10
        end;
        for j:=1 to maxlen do sum[j]:=sum[j]+fac[j];
        for j:=1 to maxlen do
```

```
            begin
                sum[j+1]:=sum[j+1]+sum[j] div 10;
                sum[j]:=sum[j] mod 10
            end;
        end;
        i:=maxlen;
        while sum[i]=0 do i:=i-1;
        write('s=');
        for j:=i downto 1 do write(sum[j]);
        writeln
end.
```

BASIC 语言参考程序：

```
    MAXLEN = 100
    DIM SUM(MAXLEN + 1), FAC(MAXLEN + 1)
    INPUT "input n:"; N
    FOR I = 1 TO MAXLEN
        SUM(I) = 0
    NEXT I
    FOR I = 1 TO MAXLEN
        FAC(I) = 0
    NEXT I
    FAC(1) = 1
    FOR I = 1 TO N
        FOR J = 1 TO MAXLEN
            FAC(J) = FAC(J) * I
        NEXT J
        FOR J = 1 TO MAXLEN
            FAC(J + 1) = FAC(J + 1) + FAC(J) \ 10
            FAC(J) = FAC(J) MOD 10
        NEXT J
        FOR J = 1 TO MAXLEN
            SUM(J) = SUM(J) + FAC(J)
        NEXT J
        FOR J = 1 TO MAXLEN
            SUM(J + 1) = SUM(J + 1) + SUM(J) \ 10
```

```
            SUM(J) = SUM(J) MOD 10
        NEXT J
    NEXT I
    I = MAXLEN
    WHILE SUM(I) = 0
          I = I - 1
    WEND
    PRINT 's = ';
    FOR J = I TO 1 STEP -1
        PRINT CHR $(SUM(J) + 48);
    NEXT J
    PRINT
```

【运行示例】 (下划线表示输入)

input n:6

s=873

input n:10

s=4037913

input n:22

s=1177652997443428940313

input n:48

s=126781637985540517671726433732557319251676942269506804209403 13

三、初中组第3题

【问题描述】

任何一个正整数都可以用2的幂次方表示。 例如:$137=2^7+2^3+2^0$

同时约定方次用括号来表示,即 a^b 可表示为a(b)。

由此可知,137可表示为:

$2(7)+2(3)+2(0)$

进一步:$7=2^2+2+2^0$(2^1用2表示)

$3=2+2^0$

所以最后137可表示为:

$2(2(2)+2+2(0))+2(2+2(0))+2(0)$

又如:$1315=2^{10}+2^8+2^5+2+1$

所以1315最后可表示为:

$2(2(2+2(0))+2)+2(2(2+2(0)))+2(2(2)+2(0))+2+2(0)$

输入:正整数(n≤20000)

输出:符合约定的 n 的 0,2 表示(在表示中不能有空格)

【问题分析】

由于问题中 n 的范围不超过 20000,所以 n 展开为 2 的幂次方后第一项的幂次不超过 14,所以只要预先将 0 到 14 的 2 的幂次方表示写出来后放在一个常量数组中,则本问题就成为了如何将一个十进制数转换为二进制数的问题了。

【数据结构】

程序中使用字符串常量数组 p 存储 0 到 14 的 2 的幂次方表示式,再用一个一维数组 b 存储输入的十进制数 n 对应的二进制数的各位数码,b[0]表示最低位。

【算法分析】

将 n 转化为二进制数的方法为“除二取余,除尽取倒”,结果存储在数组 b 中。输出时首项之前没有加号,其余各项之前均有加号,因此程序中用布尔变量 first 表示当前输出的项是否为首项,first 为 true 表示当前输出的项是首项,first 为 true 表示当前输出的项不是首项。

【程序清单】

```
program c1998_3;
const p:array[0..14] of string
        =('(0)', '', '(2)', '(2+2(0))', '(2(2))', '(2(2)+2(0))',
          '(2(2)+2)', '(2(2)+2+2(0))', '(2(2+2(0)))', '(2(2+2(0))+2(0))',
          '(2(2+2(0))+2)', '(2(2+2(0)+2+2(0))',
          '(2(2+2(0))+2(2))', '(2(2+2(0))+2(2)+2(0))', '(2(2+2(0))+
          2(2)+2)');
var i, n:integer;
    first:boolean;
    b:array[0..15] of integer;
begin
    write('Input n:');
    readln(n);
    for i:=0 to 15 do b[i]:=0;
    i:=0;
    while n>0 do
    begin
        b[i]:=n mod 2;
        n:=n div 2;
        i:=i+1
    end;
    first:=true;
    for i:=15 downto 0 do
        if b[i]=1 then
```

```
            if first
                then begin write(2, p[i]); first:=false end
                else write('+2', p[i]);
    writeln
end.
```

BASIC 语言参考程序:

```
DIM P $(14), B(15)
P $(0) = "(0)"
P $(1) = ""
P $(2) = "(2)"
P $(3) = "(2+2(0))"
P $(4) = "(2(2))"
P $(5) = "(2(2)+2(0))"
P $(6) = "(2(2)+2)"
P $(7) = "(2(2)+2+2(0))"
P $(8) = "(2(2+2(0)))"
P $(9) = "(2(2+2(0))+2(0))"
P $(10) = "(2(2+2(0))+2)"
P $(11) = "(2(2+2(0)+2+2(0))"
P $(12) = "(2(2+2(0))+2(2))"
P $(13) = "(2(2+2(0))+2(2)+2(0))"
P $(14) = "(2(2+2(0))+2(2)+2)"
INPUT "Input n:"; N
FOR I = 0 TO 15
    B(I) = 0
NEXT I
I = 0
WHILE N > 0
   B(I) = N MOD 2
   N = N \ 2
   I = I + 1
WEND
FIRST = 1
FOR I = 15 TO 0 STEP -1
  IF B(I) = 1 THEN
```

```
            IF FIRST THEN
                PRINT "2"; P $(I);
                FIRST = 0
            ELSE
                PRINT " +2"; P $(I);
            END IF
        END IF
    NEXT I
PRINT
```

【运行示例】（下划线表示输入）

Input n:73

2(2(2) +2) +2(2 +2(0)) +2(0)

Input n:136

2(2(2) +2 +2(0)) +2(2 +2(0))

Input n:255

2(2(2) +2 +2(0)) +2(2(2) +2) +2(2(2) +2(0)) +2(2(2)) +2(2 +2(0)) +2(2) +2 +2(0)

Input n:1384

2(2(2 +2(0)) +2) +2(2(2 +2(0))) +2(2(2) +2) +2(2(2) +2(0)) +2(2 +2(0))

Input n:16385

2(2(2 +2(0)) +2(2) +2) +2(0)

【小结】

本题所用的方法只适用于 20000 以内整数，如果将 n 的范围扩大到长整型，则程序将要作大的改动，因此解决本问题的最好的方法是用递归过程，因为问题的定义本身就是递归的，另外由于第一项之前没有加号，其余项之前有加号，处理时要分别对待。具体细节见下面的程序清单：

```
program c1998_3a;
var
  c:array[0..31] of longint;
  n, i:longint;

procedure print(x:longint);
var
  i, m:longint;
  b:array[0..31] of byte;
```

```
    begin
  case x of
    0:write(0);
    2:write(2);
    else
    begin
      m:=0;
      fillchar(b, sizeof(b), 0);
      while x<>0 do                    {将x转换为二进制数存储在数组b中}
      begin
        b[m]:=x mod 2;
          x:=x div 2;
          m:=m+1
      end;
      i:=m-1;
      while b[i]=0 do i:=i-1;          {找最高幂次项的幂次i}
      if i=1
        then write('2')
        else begin write('2('); print(i); write(')'); end;
      i:=i-1;
      while (i>=0) do                  {处理其余非最高幂次项的幂次i}
      begin
        if b[i]<>0 then
        begin
          if i=1
            then write('+2')
            else
            begin write('+2(');
              print(i);
              write(')');
            end;
        end;
        i:=i-1
      end
    end
  end
end;
```

```
begin
  write('Input n:');
  readln(n);
  print(n);
  writeln;
end.
```

四、高中组第 1 题

【问题描述】

车从始发站(称为第 1 站)开出,在始发站上车的人数为 a,然后到达第 2 站,在第 2 站有人上、下车,但上、下车的人数相同,因此在第 2 站开出时(即在到达第 3 站之前)车上的人数保持为 a 人。从第 3 站起(包括第 3 站)上、下车的人数有一定规律:上车的人数都是前两站上车人数之和,而下车人数等于上一站上车人数,一直到终点站的前一站(第 n - 1 站),都满足此规律。现给出的条件是:共有 n 个车站,始发站上车的人数为 a,最后一站下车的人数是 m(全部下车)。试问 x 站开出时车上的人数是多少?

输入:a, n, m 和 x

输出:从 x 站开出时车上的人数

【问题分析】

从问题的叙述可以知道:从第 3 站起(包括第 3 站)上、下车的人数只与前面两站的上下车的人数有关,即上车的人数都是前两站上车人数之和,而下车人数等于上一站上车人数,一直到终点站的前一站(第 n - 1 站),都满足此规律。所以只要得到了第一站的上车人数(第一站没有下车的人)和第二站的上车人数及下车人数,就可以推算出每一站的上车人数及下车人数,现在第一站的上车人数是已知值 a,第 2 站上、下车的人数相同,但不知道其具体的值,因此问题的关键在于求出第 2 站上、下车的人数,不妨设第 2 站上、下车的人数为 b,则从第三站起每一站的上、下车的人数均可以表示为 u * a + v * b 的形式,其中系数 u、v 均可推算得到,然后通过对系数的加减运算得出最后一站的人数 sa * a + sb * b,由于最后一站的人数是已知值 m,b 的值由此可以算出,算出 b 的值后,再用同样的方法推算出从 x 站开出时车上的人数即得到问题的解。

【数据结构】

由于要记录各站上下车人数,程序中用两个一维数组 ca 与 cb 来完成这一工作,其中第 i 站的上车人数为 ca[i] * a + cb[i] * b,算出各站的上车人数之后累加就得出全程上车人数总和。由于从第三站起下车人数等于前一站的上车人数,因此各站下车人数就不需要再开辟新的数组来存储。

【算法分析】

程序中首先根据第一站和第二站的上车人数推算各站上车人数,即计算数组 ca 和 cb 的所有元素的值,然后累计各站上车人数总和,再减去各站下车人数,计算出 b 的值之后,最终算

得从x站开出时车上的人数。

【程序清单】

```
program g1998_1;
const maxn = 100;
var i, a, n, m, x, b, sa, sb:longint;
    ca, cb:array[1..maxn] of longint;
begin
    write('Input a, n, m, x:');
    readln(a, n, m, x);
    ca[1]:=1;  ca[2]:=0;                    {第一站上车人数为1*a+0*b}
    cb[1]:=0;  cb[2]:=1;                    {第二站上车人数为0*a+1*b}
    for i:=3 to n-1 do   {计算第三站到终点站前一站(即第n-1站)的上车人数}
    begin
        ca[i]:=ca[i-1]+ca[i-2];
        cb[i]:=cb[i-1]+cb[i-2]
    end;
    sa:=0; sb:=0;                           {累计各站上车人数}
    for i:=1 to n-1 do sa:=sa+ca[i];
    for i:=1 to n-1 do sb:=sb+cb[i];
    sb:=sb-1;                               {减去第二站下车人数}
    for i:=2 to n-2 do sa:=sa-ca[i]; {减去第三站到第n-1下车人数}
    for i:=2 to n-2 do sb:=sb-cb[i]; {也就是第二站到第n-2上车人数}
    b:=(m-sa*a) div sb;   {此时的sa*a+sb*b就是到达终点站时的人数m}
    sa:=0;  sb:=0;
    for i:=1 to x do sa:=sa+ca[i];      {累计从第一站到第x站上车人数}
    for i:=1 to x do sb:=sb+cb[i];
    if x>2 then sb:=sb-1;                   {减去第二站下车人数}
    for i:=3 to x do sa:=sa-ca[i-1]; {减去第三站到第x站下车人数}
    for i:=3 to x do sb:=sb-cb[i-1];
    writeln(sa*a+sb*b)
end.
```

【运行示例】 (下划线表示输入)

```
Input a, n, m, x:5 7 32 4
13
Input a, n, m, x:0 10 40 6
8
Input a, n, m, x:10 15 2378 8
```

138

【小结】

考虑到计算每一站的人数时，a 和 b 的系数也可以用迭代的方法递推，因此程序也可以不用数组，有兴趣的同学可自己动手编一下。

五、高中组第 2 题

【问题描述】

设有 n 个正整数(n≤20)，将它们联接成一排，组成一个最大的多位整数。

例如：n = 3 时，3 个整数 13，312，343 联接成的最大整数为：34331213

又如：n = 4 时，4 个整数 7，13，4，246 联接成的最大整数为：7424613

程序输入：n

n 个数

程序输出：联接成的多位数。

【问题分析】

本例因为涉及将两个自然数连接起来的问题，采用字符串来处理比较方便。首先我们自然会想到大的字符串应该排在前面，因为如果 A 与 B 是两个由数字字符构成的字符串且 A > B，一般情况下有 A + B > B + A，但是当 A = B + C 时，按字符串的大小定义有 A > B，此时有可能出现 A + B < B + A 的情况，如 A = '121'，B = '12'，A = B + '1'，则 A + B = '12112'，B + A = '12121'，A + B < B + A。为了解决这个问题，我们根据题意引进另一种字符串比较办法，将 A + B 与 B + A 相比较，如果前者大于后者，则认为 A > B，按这一定义将所有的数字字符串从大到小排序后连接起来所得到的数字字符串即是问题的解。

【数据结构】

程序中使用一个一维数组 num 存储从键盘输入的 n 个自然数，再用一个一维数组 digitstr 存储 num 数组中相应元素转换成的字符串，digitstr 数组的每个元素为一个字符串型变量，形式如下：

```
const maxn = 20;
type stringtype = string[80];
digitstr:array[1..maxn] of stringtype;
```

【算法分析】

程序首先将从键盘输入的 n 个自然数一一转化成字符串存储到 digitstr 数组中，然后将 digitstr 数组按上述大小定义进行排序，排序时先将所有字符串中的最大值选出来存在数组的第一个元素中，再从第二至最后一个元素中最大的字符串选出来存在数组的第二个元素中，直到从最后二个元素中选出最大的字符串存在数组的倒数第二个元素中为止。

【程序清单】

```
program g1998_2;
const maxn = 20;
type stringtype = string[80];
var i, j, n:integer;
    temp:stringtype;
    num:array [1..maxn] of longint;
    digitstr:array[1..maxn] of stringtype;
begin
    write('Input n:'); readln(n);
    write('Input ', n, ' integer data(>0):');
    for i:=1 to n do read(num[i]); readln;
    for i:=1 to n do digitstr[i]:='  ';
    for i:=1 to n do
        while num[i]<>0 do
        begin
            digitstr[i]:=chr(ord('0')+num[i] mod 10)+digitstr[i];
            num[i]:=num[i] div 10    {将数转换成字符串}
        end;
    for i:=1 to n-1 do
        for j:=i+1 to n do
            if digitstr[i]+digitstr[j]<digitstr[j]+digitstr[i]
                then begin temp:=digitstr[i];
                           digitstr[i]:=digitstr[j];
                           digitstr[j]:=temp
                           end;
    for i:=1 to n do write(digitstr[i]);
    writeln
end.
```

【运行示例】 (下划线表示输入)

```
Input n:3
Input 3 integer data(>0):121 21 3
321121

Input n:4
Input 4 integer data(>0):13 24 75 42
75422413
```

Input n:4
Input 4 integer data(>0):1341 133 1321 37
3713411331321

Input n:6
Input 6 integer data(>0):321 32 407 135 13 217
4073232121713513

【小结】

需要说明的是,按本题定义的字符串的大小定义是有序的,即如果 a + b≥b + a,b + c≥c + b,则一定有 a + c≥c + a。证明方法如下:

引理:记 na 为 n 个字符串 a 按字符串加法运算规则相加之和,则由 a + b≥b + a 可推导出 na + mb≥mB + nA,其中 m, n 为任意的自然数。用反证法可证明反过来也成立。

设 la 为字符串 a 的长度,lb 为字符串 b 的长度,lc 为字符串 c 的长度,再设 n = lb × lc, m = la × lc, k = la × lb,则 na, mb, kc 三个字符串等长,根据引理及已知条件 a + b≥b + a, b + c≥c + b, 有 na + mb≥mb + na, mb + kc≥kc + mb,从而得到 na≥mb≥kc,所以 na + kc≥kc + na,所以 a + c≥c + a。

综上所述,要使 n 个字符串拼接起来后得到一个最大的字符串和式,则一定要将按上述定义最大的字符串放在第一个,否则必可通过将最大的字符串与它左侧的字符串交换得到更大的字符串和式。

高中组第3题

【问题描述】

著名科学家卢斯为了检查学生对进位制的理解,他给出了如下的一张加法表,表中的字母代表数字。例如:

+	L	K	V	E
L	L	K	V	E
K	K	V	E	KL
V	V	E	KL	KK
E	E	KL	KK	KV

其含义为:

L + L = L, L + K = K, L + V = V, L + E = E

K + L = K, K + K = V, K + V = E, K + E = KL

……

E + E = KV

根据这些规则可推导出:L = 0, K = 1, V = 2, E = 3

同时可以确定该表表示的是4进制加法

程序输入:n(n≤9)表示行数,以下 n 行,每行包括 n 个字符串,每个字串间用空格隔开。(字串仅有一个为‘+’号,其他都由大写字母组成)

程序输出:

① 各个字母表示什么数,格式如:L=0, K=1,…

② 加法运算是几进制的。

③ 若不可能组成加法表,则应输出“ERROR!”

【问题分析】

首先根据输入的 n 可知加法运算是 n-1 进制的,在此基础上我们再对表中出现的字母逐一作出猜测,猜测的范围是 0 到 n-1 之间尚未被猜过的数字,每猜一个数之后就对表格作一次检查,如果检查未发现任何错误,则继续猜测下一个字母,否则对当前字母换猜下一数字,假如当前字母无论如何猜测均行不通,则回溯到上一个字母,对上一个字母换猜下一数字,直到所有字母均猜出或全部猜完(无解)为止。

【数据结构】

对于输入的加法表,为了便于验证,比较好的方法是将它们转化为 n^2 个加式来存储,存储结构采用一维数组 ex 表示,每个数组元素为一个长度为 4 的字符串变量,用于存储一个加式,该字符串的第一个字母表示加数,第二个字母表示被加数,第三和第四个字母表示加式之和,若加式之和只有一个字母则第三个字母用空格字符填充。如加式“E+V=KK”存储为“EVKK”,加式“L+L=L”则存储为“LLL”。另外为了记录最终结果,程序用一个一维数组 code 记录字母对应的数字,数组下标的类型为字符类型,基类型为整型;为了判断当前所猜数字是否已用过,程序中用一个集合 used 记录已用过的数字,同样对已猜过的字母,程序也用一个集合 known 记录之。

【算法分析】

程序用递归的方法实现对每个字符的猜测,递归过程如下:

```
procedure try(dep:integer;  used:digitset;  known:letterset);
begin
   if dep>n then  输出解,结束程序
     else
      for i:=0 to n-2 do
          if  数字 i 尚未用过 then
          begin
               记录当前字符为 i
               if 加法表没有错误 then try(dep+1, used+[i], known+[ch])
          end;
end;
```

其中 dep 表示递归的层数,当前被猜测的字符为加法表中的第 dep 行中的第一个字符。判断加法表是否有错的算法如下:对 n^2 个加式逐一扫描,若当前加式中所有字母均已猜过且至少有一个是本层递归调用所猜字母,则验证之,验证有错即中止验证,重猜当前字母,否则继续验证下一加式。

【程序清单】

```
program g1998_3;
const maxn = 9;
type   digitset = set of 0..9;
       letterset = set of char;
var i, j, k, n, total:integer;
    ch:char;
    line:string;
    list:array[1..maxn, 1..maxn] of string[2];
    ex:array[1..maxn * maxn] of string[4];
    code:array[char] of integer;
    s:set of char;
procedure print;
var ch:char;
begin
    for ch:='A' to 'Z' do
        if code[ch] >=0 then writeln(ch, '=', code[ch]);
    writeln(n-1, ' 进制');
    halt
end;
procedure try(dep:integer;  used:digitset;  known:letterset);
var i, j, k:integer;
    ch:char;
    error:boolean;
begin
  if dep > n then print
    else
      for i:=0 to n-2 do
        if not(i in used) then
        begin
          ch:=list[dep, 1, 2];
          code[ch]:=i;
          k:=1;
          error:=false;
          while not(error) and (k <=total) do
          begin
            s:=[];
```

```
                for j:=1 to 4 do s:=s+[ex[k,j]];
                if (s*[ch]=[ch]) and (s<=known+[ch])
                    then if code[ex[k,1]]+code[ex[k,2]]<>
                              code[ex[k,3]]*(n-1)+code[ex[k,4]]
                              then error:=true;
                k:=k+1
              end;
              if not(error) then try(dep+1,used+[i],known+[ch])
            end;
end;
begin
    readln(n);
    for i:=1 to n do
    begin
      readln(line);                    {读入加法表中的一行}
      j:=1;  k:=0;
      while j<=length(line) do         {扫描当前行,将其中的字母分离出来}
      begin
        while (j<=length(line)) and not(line[j] in ['A'..'Z',' +']) do
              j:=j+1;                  {跳过空白类字符}
        if j<=length(line) then        {记录加式的一部分}
        begin
          k:=k+1;                  list[i,k]:='';
          while (j<=length(line)) and (line[j] in ['A'..'Z',' +']) do
          begin
            list[i,k]:=list[i,k]+line[j];                    j:=j+1
          end;
          if length(list[i,k])=1
            then list[i,k]:=' '+list[i,k]
        end
      end
    end;
    {加式已用表格形式存储在二维数组 list 中,每个元素存储二个字符,内容为加式中的被加数、加数或和,不足二个字符时在前面加一个空格字符,下面将加法表转换到一维数组 ex 中去}
    total:=0;
    for i:=2 to k do
```

```
            for j:=2 to k do
            begin
                total:=total+1;
                ex[total]:=ex[total]+list[i,1,2]+list[1,j,2]+list[i,j]
            end;
        for ch:='A' to 'Z' do code[ch]:=-1;    {-1 表示相应字母未猜过}
        code[' ']:=0;                          {空格字符解释为 0}
        try(2,[],[' ']);                       {已猜字符集合初始时只包含空格字符}
        writeln('Error!')
end.
```

【运行示例】

```
3
 +   M   L
M   ML   M
L   M   L
L=0
M=1
2 进制

4
 +   M   N   P
M   N   MP M
N   MP MM N
P   M   N   P
M=1
N=2
P=0
3 进制

6
 +   M   L   K   N   H
M   L   H   M   MK N
L   H   N   L   MM MK
K   M   L   K   N   H
N   MK MM N   MH ML
H   N   MK H   ML MM
H=3
```

K = 0

L = 2

M = 1

N = 4

5 进制

8

+	M	N	L	P	Q	R	S
M	S	LL	P	R	M	LQ	N
N	LL	LR	LQ	LM	N	LS	LP
L	P	LQ	M	S	L	N	R
P	R	LM	S	N	P	LL	LQ
Q	M	N	L	P	Q	R	S
R	LQ	LS	N	LL	R	LP	LM
S	N	LP	R	LQ	S	LM	LL

L = 1

M = 2

N = 6

P = 3

Q = 0

R = 5

S = 4

7 进制

【小结】

假如给出的加法表是不完备的,即少了若干行若干列的话,本程序所用方法依然有效,只要加进对进制的穷举即可,正式比赛时因受时间限制,不宜考虑得太多,但赛后应多总结。

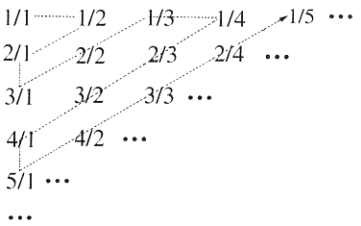
第五章　1999 年复赛试题解析

一、初中组第 1 题 Cantor 表

【问题描述】

现代数学的著名证明之一是 Georg Cantor 证明了有理数是可枚举的。他是用下面这一张表来证明这一命题的：

1/1　1/2　1/3　1/4　1/5 …
2/1　2/2　2/3　2/4 …
3/1　3/2　3/3 …
4/1　4/2 …
5/1 …
…



我们以 Z 字形给上表的每一项编号。第一项是 1/1，然后是 1/2，2/1，3/1，2/2，…

输入：整数 N(1≤N≤10000000)

样例：

INPUT

N = 7

输出：表中的第 N 项

OUTPUT

1/4

【问题分析】

首先根据 N 的值不难算出第 N 个有理数所在的斜线的序号 k 及其在第 k 条斜线上的序号。不难发现，在这张表中每一条斜线上的每一个分数的分子和分母的和都相等，且等于斜线的序号 +1，假设 1/1 所在的斜线序号为 1。同时也发现，当 k 是偶数时，第 k 条斜线上的有理数按分子加 1 分母减 1 的规律从 1/k 渐变成 k/1；当 k 是奇数时，第 k 条斜线上的有理数按分子减 1 分母加 1 的规律从 k/1 渐变成 1/k；如 N = 8 时，其所处的位置是第 4 条斜线上的第 2 个，此时 k 是偶数，第 4 条斜线上的有理数应按分子加 1 分母减 1 的规律从 1/4 渐变成 4/1，由此得出第 8 个分数是 2/3。

【数据结构】（略）

【算法分析】

首先将 N 依次减去 1，2，3，…，k，当余下的值（依然存放在变量 N 中）小于等于 k 不再循环。此时的 k 即为所求的有理数所在的斜线的序号，变量 N 的值即为所求的有理数在第 k 条斜线上的序号，判断 k 是否为偶数，如果是则输出 n/(k+1−n)，否则输出(k+1−n)/n。

【程序清单】

```
program c1999_1;
var i, j, k, n:longint;
begin
      write('Input n:');
      readln(n);
      k:=1;
      while n>k do
      begin
            n:=n-k;
            k:=k+1;
      end;
      if k mod 2=0
         then writeln(n, '/', k+1-n)
         else writeln(k-n+1, '/', n)
end.
```

BASIC 语言参考程序:

```
      INPUT "Input n:"; N
      K = 1
      WHILE N > K
            N = N - K
            K = K + 1
      WEND
      IF K MOD 2 = 0 THEN PRINT N;  "/"; K + 1 - N ELSE PRINT K - N + 1;
"/";N
```

【运行示例】 (下划线表示输入)

```
Input n:15
1/5

Input n:85
7/7

Input n:1999
18/46

Input n:10278
19/125
```

二、初中组第 2 题/高中组第 2 题　回文数

【问题描述】

若一个数(首位不为零)从左向右读与从右向左读都一样,我们就将其称之为回文数。

例如:给定一个 10 进制数 56,将 56 加 65(即把 56 从右向左读),得到 121 是一个回文数。

又如:对于 10 进制数 87:

STEP1:87 + 78 = 165　　STEP2:165 + 561 = 726

STEP3:726 + 627 = 1353　　STEP4:1353 + 3531 = 4884

在这里的一步是指进行了一次 N 进制的加法,上例最少用了 4 步得到回文数 4884。

写一个程序,给定一个 N(2 < = N < = 10,N = 16)进制数 M,求最少经过几步可以得到回文数。

如果在 30 步以内(包含 30 步)不可能得到回文数,则输出"Impossible!"

样例:

INPUT	OUTPUT
N = 9　M = 87	STEP = 6

【问题分析】

本题的难度较低,主要是考察选手对数据结构的掌握与运用。问题可以分为三个部分。第一,是对读入数据的处理。本题中,数据的进制是可变的,所以用整型数(即十进制数)不是很方便,考虑一般情况,我们可以采用字符串读入,再对数据作后期加工,把它的每一位都分离开来,存入一个数组里。在转化过程中要注意到 16 进制的特殊性,我们对于字母 A B C D E F 要单独处理。第二,进行判断与运算。有的选手在比赛时是用字符串作运算的,也有的选手在比赛时是用整型数组运算的,其实存储时用什么形式表示都无所谓,关键是做加法运算的时候要符合 n 进制的规律。所以,运算时一律可以用 10 进制形式存储,A 用 10,B 用 11,…,F 用 15 表示。判断是否构成回文数时也一样用 10 进制,逐对比较对称的两个数位上的数字,看它们是否相等。做加法的次数以 30 次为限。

第三,输出结果。如果当前的数是回文数,则输出步数;否则按"Impossible"输出。

【数据结构】

由于输入数据可以是不同进制的数,所以要用一个字符串 str 存储输入的 n 进制数,并将它转化到一个一维数组 digit 中作为高精度数处理,digit 为整型数组,每个元素存储输入的 n 进制数的一位,并用一个单独的变量 len 记录 digit 的有效长度,digit[1]记录个位数字,digit[len]记录最高位数字。

【算法分析】

本题的关键在于回文数的判断和相加。判断 digit 数组是否为回文数的方法如下:使用变量 i 和 j 存放当前要比较的两个数组元素的下标,开始时,i = 1,j = len,若 digit[i] = digit[j]则 i 增加 1,同时 j 减少 1,当 i≥j 或某次比较失败则比较结束,此时若 i < j 且 digit[i] < > digit[j]则 digit 数组不是回文数,反之则是回文数。若 digit 数组不是回文数,则用迭代的方法将其作回文相加处理,具体方法是将 digit 数组的前半段的每个元素与其后半段的对应元素相加,即

for i:=1 to len div 2 do digit[i]:=digit[i]+digit[len+1-i];然后根据对称性将前半段复制到后半段,即 for i:=1 to len div 2 do digit[len+1-i]:=digit[i];若位数为奇数则正中间一位乘以2,即 if odd(len) then digit[len div 2+1]:=digit[len div 2+1]*2;最后将新的 digit 数组作进位处理。

【程序清单】

```
program c1999_2;
const max=10000;
type arraytype=array [1..max] of longint;
var i,n, len, step:longint;
    str:string;
    digit:arraytype;
function palindrome(var digit:arraytype;  len:longint):boolean;
var i, j:longint;
begin
    i:=1;  j:=len;
    while (i<j) and (digit[i]=digit[j]) do
    begin i:=i+1;  j:=j-1 end;
    if i<j then palindrome:=false else palindrome:=true
end;
begin
    write('Input n:'); readln(n);
    write('Input number:');  readln(str);
    for i:=1 to max do digit[i]:=0;
    len:=length(str);
    for i:=1 to len do
        if str[i]>='A'
          then digit[len+1-i]:=ord(str[i])-ord('A')+10
          else digit[len+1-i]:=ord(str[i])-ord('0');
    step:=0;
    while (step<30) and not(palindrome(digit, len)) do
    begin
        for i:=1 to len div 2 do digit[i]:=digit[i]+digit[len+1-i];
        for i:=1 to len div 2 do digit[len+1-i]:=digit[i];
        if odd(len) then digit[len div 2+1]:=digit[len div 2+1]*2;
        for i:=1 to len do
            if digit[i]>=n then
                begin digit[i+1]:=digit[i+1]+1;
```

```
                    digit[i]:=digit[i]-n end;
            if digit[len+1]>0 then len:=len+1;
            step:=step+1
        end;
        if step=30
            then writeln('Impossible!')
            else writeln('STEP=', step)
end.
```

BASIC 语言参考程序：

```
    MAX = 1000
    DIM S $(20), DIGIT(MAX)
    INPUT "Input n:"; N
    INPUT "Input number:"; S $
    FOR I = 1 TO MAX
        DIGIT(I) = 0
    NEXT I
    FOR I = 1 TO LEN(S $)
        IF MID $(S $, I, 1) >= "A" THEN
         DIGIT(LEN(S $) + 1 - I) = ASC(MID $(S $, I, 1)) - ASC("A") + 10
        ELSE
            DIGIT(LEN(S $) + 1 - I) = ASC(MID $(S $, I, 1)) - ASC("0")
        END IF
    NEXT I
    STEPS = 0
    H = LEN(S $)
10  I = 1
    J = H
    WHILE (I < J) AND (DIGIT(I) = DIGIT(J))
        I = I + 1
        J = J - 1
    WEND
    IF I < J THEN PALINDROME = 0 ELSE PALINDROME = 1
    IF STEPS < 30 AND PALINDROME = 0 THEN
        FOR I = 1 TO H \ 2
            DIGIT(I) = DIGIT(I) + DIGIT(H + 1 - I)
```

```
            NEXT I
            FOR I = 1 TO H \ 2
                DIGIT(H + 1 - I) = DIGIT(I)
            NEXT I
            IF H MOD 2 = 1 THEN DIGIT(H \ 2 + 1) = DIGIT(H \ 2 + 1) * 2
            FOR I = 1 TO H
                IF DIGIT(I) > = N THEN
                    DIGIT(I + 1) = DIGIT(I + 1) + 1
                    DIGIT(I) = DIGIT(I) - N
                END IF
            NEXT I
            IF DIGIT(H + 1) > 0 THEN H = H + 1
            STEPS = STEPS + 1
            GOTO 10
        ELSE
            GOTO 20
        END IF
20    IF PALINDROME = 0 THEN PRINT " Impossible!" ELSE PRINT " STEP
=" ; STEPS
```

【运行示例】（下划线表示输入）

```
Input n:2
Input number:10011
STEP =4

Input n:16
Input number:AC27
STEP =6

Input n:10
Input number:89
STEP =24

Input n:2
Input number:101111
Impossible!
```

三、初中组第3题/高中组第3题　旅行家的预算

【问题描述】

一个旅行家想驾驶汽车以最少的费用从一个城市到另一个城市(假设出发时油箱是空的)。给定两个城市之间的距离D1、汽车油箱的容量C(以升为单位)、每升汽油能行驶的距离D2、出发点每升汽油价格P和沿途油站数N(N可以为零),油站i离出发点的距离Di、每升汽油价格Pi(i=1, 2, …, N)。

计算结果四舍五入至小数点后两位。

如果无法到达目的地,则输出"No Solution"。

样例:

INPUT

D1=275.6　C=11.9　D2=27.4　P=2.8　N=2

油站号I	离出发点的距离Di	每升汽油价格Pi
1	102.0	2.9
2	220.0	2.2

OUTPUT

26.95(该数据表示最小费用)

【问题分析】

看到这道题,许多人都马上判断出穷举是不可行的,因为数据都是以实数的形式给出的。但是,不用穷举,有什么方法是更好的呢?递推是另一条常见的思路,但是具体方法不甚明朗。

既然没有现成的思路可循,那么先分析一下问题不失为一个好办法。由于汽车是由始向终单向开的,我们最大的麻烦就是无法预知汽车以后对汽油的需求及油价变动;换句话说,前面所买的多余的油只有开到后面才会被发觉。

提出问题是解决的开始。为了着手解决遇到的困难,取得最优方案,那就必须做到两点,即只为用过的汽油付钱;并且只买最便宜的油。如果在以后的行程中发现先前的某些油是不必要的,或是买贵了,我们就会说:"还不如当初不买。"由这一个想法,我们可以得到某种启示:假设我们在每个站都买了足够多的油,然后在行程中逐步发现哪些油是不必要的,以此修改我们先前的购买计划,节省资金;进一步说,如果把在各个站加上的油标记为不同的类别,我们只要在使用时用那些最便宜的油并为它们付钱,其余的油要么是太贵,要么是多余的,在最终的计划中会被排除。要注意的是,这里的便宜是对于某一段路程而言的,而不是全程。

【数据结构】

数据结构采用一个有序线性表oil,存放由便宜到贵的各种油,线性表中每个元素为一个记录类型的数据,记录一种油在油箱中的容量和价格。在一路使用汽油一路补充汽油的过程中同步修改数据,求得最优方案。

【算法分析】

由此,我们得到如下算法:从起点起,每到一个站都把油箱加满(终点除外);每经过两站

之间的距离,都按照从便宜到贵的顺序使用油箱中的油,并计算花费,因为这是在最优方案下不得不用的油;当汽车到达某一站时发现当前加油站的油价低于油箱中仍保存的某些汽油的油价,则说明以前的购买是不够明智的,其效果一定不如购买当前加油站的油,所以,明智的选择是用本站的油代替以前购买的高价油,留待以后使用,由于我们不是真的开车,也没有为备用的油付过钱,因而这样的反悔是可行的;当我们开到终点时,意味着路上的费用已经得到,此时剩余的油就没有用了,可以忽略。

【程序清单】

```
program c1999_3;
const max = 1000;
type recordtype = record price, content:real end;
var i, j, n, point, tail:longint;
    content, change, distance2, money, use:real;
    price, distance, consume:array[0..max] of real;
    oil:array [0..max] of recordtype;
begin
    write('Input DI, C, D2, P:');
    readln(distance[0], content, distance2, price[0]);
    write('Input N:');   readln(n);   distance[n+1]:=distance[0];
    for i:=1 to n do
    begin
        write('Input D[', i, '], ', 'P[', i, ']:');
        readln(distance[i], price[i])
    end;
    distance[0]:=0;
for i:=n downto 0 do consume[i]:=(distance[i+1]-distance[i])/distance2;
{计算相邻两站间的油耗}
    for i:=0 to n do
        if consume[i]>content then
            begin writeln('No Solution');   halt end;
    money:=0;   tail:=1;   change:=0;
    oil[tail].price:=price[0]*2;   oil[tail].content:=content;
    for i:=0 to n do
    begin
        point:=tail;
        while (point>=1) and (oil[point].price>=price[i]) do
        begin
            change:=change+oil[point].content;
```

```
                point: = point - 1
            end;
            tail: = point + 1;
            oil[tail]. price: = price[i];
            oil[tail]. content: = change;
            use: = consume[i]; point: = 1;
            while (use > 1e - 6) and (point < = tail) do
                if use > = oil[point]. content
                    then begin use: = use - oil[point]. content;
                        money: = money + oil[point]. content * oil[point]. price;
                        point: = point + 1 end
                    else begin oil[point]. content: = oil[point]. content - use;
                        money: = money + use * oil[point]. price;
                        use: = 0 end;
            for j: = point to tail do oil[j - point + 1]: = oil[j];
            tail: = tail - point + 1;
            change: = consume[i]
        end;
        writeln(money:0:2)
end.
```

BASIC 语言参考程序：

```
MAX = 100
DIM PRICE(MAX), DISTANCE(MAX), CONSUME(MAX), OIL(MAX, 2)
INPUT "Input DI, C, D2, P:"; DISTANCE(0), CONTENT, DISTANCE2, PRICE(0)
INPUT "Input N:";  N
DISTANCE(N + 1) = DISTANCE(0)
FOR I = 1 TO N
    PRINT "Input D["; I; "], "; "P["; I; "]:";
    INPUT DISTANCE(I), PRICE(I)
NEXT I
DISTANCE(0) = 0
FOR I = N TO 0 STEP -1
    CONSUME(I) = (DISTANCE(I + 1) - DISTANCE(I)) / DISTANCE2
NEXT I
FOR I = 0 TO N
```

```
    IF CONSUME(I) > CONTENT THEN
        PRINT "No Solution"
        END
    END IF
NEXT I
MONEY = 0
TAIL = 1
CHANGE = 0
OIL(TAIL, 1) = PRICE(0) * 2
OIL(TAIL, 2) = CONTENT
FOR I = 0 TO N
    P = TAIL
    WHILE P >= 1 AND OIL(P, 1) >= PRICE(I)
          CHANGE = CHANGE + OIL(P, 2)
          P = P - 1
    WEND
    TAIL = P + 1
    OIL(TAIL, 1) = PRICE(I)
    OIL(TAIL, 2) = CHANGE
    USE = CONSUME(I)
    P = 1
    WHILE USE > .000001 AND P <= TAIL
          IF USE >= OIL(P, 2) THEN
             USE = USE - OIL(P, 2)
             MONEY = MONEY + OIL(P, 2) * OIL(P, 1)
             P = P + 1
          ELSE
             OIL(P, 2) = OIL(P, 2) - USE
             MONEY = MONEY + USE * OIL(P, 1)
             USE = 0
          END IF
    WEND
    FOR J = P TO TAIL
        FOR K = 1 TO 2
            OIL(J - P + 1, K) = OIL(J, K)
        NEXT k
    NEXT J
```

```
        TAIL = TAIL - P + 1
        CHANGE = CONSUME(I)
    NEXT I
    PRINT INT(MONEY * 100 + .5) / 100
```

【运行示例】(下划线表示输入)

```
Input DI, C, D2, P:9.99  1.59  29.8  99.9
Input N:0
334.90

Input DI, C, D2, P:199.9  9.0  10.0  99.9
Input N:1
Input D[1], P[1]:100.0  99.9
No Solution

Input DI, C, D2, P:87.75  13.03  5.75  7.29
Input N:3
Input D[1], P[1]:22.10  7.38
Input D[2], P[2]:24.21  6.81
Input D[3], P[3]:82.08  6.96
105.95

Input DI, C, D2, P:475.6  11.9  27.4  14.98
Input N:6
Input D[1], P[1]:102.0  9.99
Input D[2], P[2]:220.0  13.29
Input D[3], P[3]:256.3  14.79
Input D[4], P[4]:275.0  10.29
Input D[5], P[5]:277.6  11.29
Input D[6], P[6]:381.8  10.09
192.15
```

【小结】

注意:每到一站都要将油加满,以确保在有解的情况下能走完全程。

本题的一个难点在于认识到油箱中油的可更换性,在这里,突破现实生活中的思维模式显得十分重要,在比赛中想到这一点很困难,比较容易想到的是分治法,以下给予简单介绍:

首先找到(从后向前)油价最低的加油站,显然车至该站油箱应为空,这样就可将起点至该站与该站至终点作为两段独立考虑,分别求其最小费用,二者之和即为总费用,这样一直分下去,若某段只有起点与终点两个加油站时无需再分,如某一段油价最低的加油站即为起点,则如能一次加油即到达该段终点则最好,若不能,则加满油再考虑油箱有油情况下的二分法,考虑起点之外所有的加油站中从后往前油价最低的加油站,若该加油站位于起点加满油后不能到达之处,则到达该站时油箱应该为空,以该站为界将全程分为两个独立段考虑,前半段为有油情况,后半段为无油情况。第二种情况,若该加油站处于起点加满油后能到达之处,则将该段总路程缩短为该加油站至终点的情况,该加油站在该段路程中最便宜,若从该站加满油仍不能到达终点,则继续分治即可,程序被设计成一个递归函数 money,形式参数 start 表示起点站,形式参数 stop 表示终点站,形式参数 rest 表示到达加油站 start 时汽车油箱余下的油的容量,money 函数最终计算出从加油站 start 到 stop 区间内的最小费用,细节详见程序清单和注释。

```
program c1999_3a;const max=1000;
var i, n:longint;
    c, d2:real;
    p, d, consume:array[0..max] of real;
function minp(b, e:longint):longint;{在 b 站到 e 站之间从后往前找油价最低的站}
var i, k:longint;
    tempminp:real;
begin
     tempminp:=p[e];  k:=e;
     for i:=e-1 downto b do
         if p[i]<tempminp then
         begin tempminp:=p[i];  k:=i end;
     minp:=k
end;
function money(start, stop:longint;rest:real):real;
var k:longint;
begin
  if stop-start=1
     then money:=((d[stop]-d[start])/d2-rest)*p[start]
     else
     begin
       k:=minp(start, stop-1);
       if k<>start{油价最低的加油站不是起点站}
         then money:=money(start, k, rest)+money(k, stop, 0)
```

```
            else if d[stop] - d[start] <= d2 * c{在起点加满油能直接到达该段终点}
                then money: = ((d[stop] - d[start])/d2 - rest) * p[start]
                else
                begin
                  k: = minp(start + 1, stop - 1);
                if d[k] - d[start] <= d2 * c then{在起点加满油能到达加油站 k}
  money: = (c - rest) * p[start] + money(k, stop, c - (d[k] - d[start])/d2)
  {在起点处加满油箱共加油 c - rest 升,到达加油站 k 用去(d[k] - d[start])/d2}
  else money: = money(start, k, rest) + money(k, stop, 0)
                end
    end
end;
begin
    write('Input D1, C, D2, P:');
    readln(d[0], c, d2, p[0]);
    write('Input N:');  readln(n);  d[n+1]: = d[0];
    for i: = 1 to n do
    begin
        write('Input D[', i, '], ', 'P[', i, ']:');
        readln(d[i], p[i])
    end;
    d[0]: = 0;
    for i: = n downto 0 do consume[i]: = (d[i+1] - d[i])/d2;
    for i: = 0 to n do
      if consume[i] > c then
        begin writeln('No Solution'); halt end;
    writeln(money(0, n+1, 0):0:2){起点站编号为 0,终点站编号为 n+1}
end.
```

四、高中组第 1 题　拦截导弹

【问题描述】

某国为了防御敌国的导弹袭击,发展出一种导弹拦截系统。但是这种导弹拦截系统有一个缺陷:虽然它的第一发炮弹能够到达任意的高度,但是以后每一发炮弹都不能高于前一发的高度。某天,雷达捕捉到敌国的导弹来袭。由于该系统还在试用阶段,所以只有一套系统,因此有可能不能拦截所有的导弹。

输入导弹依次飞来的高度(雷达给出的高度数据是不大于 30000 的正整数),计算这套系

统最多能拦截多少导弹,如果要拦截所有导弹最少要配备多少套这种导弹拦截系统。

样例:

INPUT	OUTPUT
389 207 155 300 299 170 158 65	6(最多能拦截的导弹数)
	2(要拦截所有导弹最少要配备的系统数)

【问题分析】

第一部分是要求输入数据串中的一个最长不上升序列的长度,可使用递推的方法,具体做法是从序列的第一个元素开始,依次求出以第 i 个元素为最后一个元素时的最长不上升序列的长度 longest(i),递推式为 longest(1) = 1,longest(i) = max(longest(j) + 1),其中 i > 1,j = 1,2,…,i - 1,且 j 同时要满足条件:序列中第 j 个元素大于等于第 i 个元素。

第二部分比较有意思。由于它紧接着第一问,所以比赛时很容易受前面的影响,采取多次求最长不上升序列的办法,然后得出总次数,其实这是不对的。要举反例并不难,比如长为 7 的高度序列"7 5 4 1 6 3 2",用多次求最长不上升序列("7 5 4 3 2"、"1"、"6")的结果为 3 套系统;但其实只要 2 套,分别击落"7 5 4 1"与"6 3 2"。那么,正确的做法又是什么呢?

我们的目标是用最少的系统击落所有导弹,至于系统之间怎么分配导弹数目则无关紧要;上面错误的想法正是承袭了"一套系统尽量多拦截导弹"的思维定势,忽视了最优解中各个系统拦截数较为平均的情况,本质上是一种贪心算法。如果从每套系统拦截的导弹方面进行思考行不通的话,我们就应该换一个思路,从拦截某个导弹时应该选择的系统入手。题目告诉我们,已有系统目前的瞄准高度必须不低于来犯导弹高度,所以,当已有的系统均无法拦截该导弹时,就不得不启用新系统。如果已有系统中有一个能拦截该导弹,我们是应该继续使用它,还是另起炉灶呢?事实是:无论用哪套系统,只要拦截了这枚导弹,那么系统的瞄准高度就等于导弹高度,这一点对旧的或新的系统都适用。而新系统能拦截的导弹高度最高,即新系统的性能优于任意一套已使用的系统。既然如此,我们当然应该选择已有的系统。如果已有系统中有多个可以拦截该导弹,究竟选哪一个呢?显而易见当前瞄准高度较高的系统的"潜力"较大,而瞄准高度较低的系统则不同,它能打下的导弹别的系统也能打下,它够不到的导弹却未必是别的系统所够不到的。所以,当有多个系统供选择时,要选瞄准高度最低的使用,当然瞄准高度同时也要大于等于来犯导弹高度。

【数据结构】

由于输入数据只知道是一行,而不知道到底有多少个数,因此要用一个字符串变量 line 接受输入的一行数据,然后再通过扫描 line 将它转化为一系列的整数存储到一个一维数组 height 中。另外用一维数组 longest 记录以任一元素为最后一个元素时最长不上升序列的长度,一维数组 sys 记录下已启用的每一套系统的当前瞄准高度,数组元素个数就是系统数目。

【算法分析】 (略)

【程序清单】

```
program g1999_1;
const max = 1000;
var i, j, current, maxlong, minheight, select, tail, total:longint;
    height, longest, sys:array [1..max] of longint;
```

```
    line:string;
begin
    readln(line);
    i:=1;
    while i<=length(line) do                          {扫描字符串 line}
    begin
        while (i<=length(line)) and (line[i]=' ') do i:=i+1;{跳过空格字符}
        current:=0;
       while (i<=length(line)) and (line[i]<>' ') do{将连续的数字字符转为整数}
        begin
            current:=current*10+ord(line[i])-ord('0');
            i:=i+1
        end;
        total:=total+1;
        height[total]:=current
    end;
    longest[1]:=1;
    for i:=2 to total do
    begin
        maxlong:=1;
        for j:=1 to i-1 do
        begin
            if height[i]<=height[j]
              then if longest[j]+1>maxlong
                      then maxlong:=longest[j]+1;
            longest[i]:=maxlong
       end;
    end;
    maxlong:=longest[1];
    for i:=2 to total do
      if longest[i]>maxlong then maxlong:=longest[i];
    writeln(maxlong);
    sys[1]:=height[1];   tail:=1;
    for i:=2 to total do
    begin
        minheight:=maxint;
        for j:=1 to tail do
```

```
                if sys[j] > height[i] then
                    if sys[j] < minheight then
                        begin minheight := sys[j];   select := j end;
            if minheight = maxint
                then begin tail := tail + 1;   sys[tail] := height[i] end
                else sys[select] := height[i]
        end;
        writeln(tail)
end.
```

【运行示例】 (下划线表示输入)

```
300  250  275  252  200  138  245
5
2

181  205  471  782  1033  1058  1111
1
7

465  978  486  476  324  575  384  278  214  657  218  445  123
7
4

236  865  858  565  545  445  455  656  844  735  638  652  569  714  845
6
7
```

五、高中组第4题　邮票面值设计

【问题描述】

给定一个信封,最多只允许粘贴N张邮票,计算在给定K(N+K≤40)种邮票的情况下(假定所有的邮票数量都足够),如何设计邮票的面值,能得到最大值MAX,使在1~MAX之间的每一个邮资值都能得到。

例如,N=3,K=2,如果面值分别为1分、4分,则在1分~6分之间的每一个邮资值都能得到(当然还有8分、9分和12分);如果面值分别为1分、3分,则在1分~7分之间的每一个邮资值都能得到。可以验证当N=3,K=2时,7分就是可以得到的连续的邮资最大值,所以MAX=7,面值分别为1分、3分。

样例：

INPUT

N = 3　K = 2

OUTPUT

1　3

MAX = 7

【问题分析】

考试的时候给出的数据规模是 n + k ≤ 40，因此不少选手都想方设法采用动态规划；实际上，本题具有明显的后效性，前面方案的不同使得构成某面值的邮票数不同，动态规划是不成立的。一般认为解决这类问题的方法只有递归搜索，所能解决的问题规模也远远达不到题目的要求（n 和 k 基本不能同时超过 6），这可以说是条件中的一个陷井。即便对于较小的 n 和 k，要想出解也要具有正确的思路才行。

首先面值为 1 的邮票是必需的，可以固定；此后每加进一种面值都要把当前所能取得的金额记录下来，通过 k - 1 次的添加得到一种方案，穷举所有的方案得到最优解。这就是算法的基本框架。

【数据结构】

在搜索的过程中，对数据的记录方式很关键。如果仅仅记录一个金额能否被达到，则这样的记录基本上没有用，因为下一次添加邮票时不知道能否从这个金额出发再加上一张新邮票，必须全部重算，浪费了大量时间；所以记录最少要用几张邮票达到一个面值是比较方便的，它有利于添加新邮票时迅速判断可行性。另外，搜索一种邮票面值的起讫点也要注意，应当是从上一个面值加 1 直到当前连续取得的最大金额加 1。在这个范围之外的解都可以不予考虑：过大导致不连续；小于前一面值时可将前后面值交换，仍得到增序的邮票面值序列。程序中用一维数组 s 记录搜索过程中每种邮票的面值，数组 result 记录当前最优解，数组 d 记录当前情况下每个面值最少要用几张邮票才能达到，maxc 表示当前情况下不能达到的最小面值。

【算法分析】　（略）

【程序清单】

```
program g1999_4;
const max = 250;
type arraytype = array [0..max] of longint;
var i, kind, num, maxcontinue:longint;
    s, result:array[1..10] of longint;
    d:arraytype;
function continue(var d:arraytype):longint;
var i, c:longint;
begin
    c:=1;
```

```
        while d[c] <=num do c:=c+1;
        continue:=c
end;
procedure try(dep, maxc:longint;d:arraytype);
var i, j, k:longint;
    temp:arraytype;
begin
      if dep>kind
         then if maxc>maxcontinue
                   then begin maxcontinue:=maxc;   result:=s end
                   else
    else for i:=s[dep-1]+1 to maxc do
          begin
          s[dep]:=i;
                   temp:=d;
{以下计算添加一种面值为i的邮票后,各种面值最少需贴的邮票张数}
                 for j:=1 to num-1 do  {贴j张面值为i的邮票}
                     for k:=0 to max-j*i do
                         if temp[k+j*i]>d[k]+j then temp[k+j*i]:=d[k]+j;
                         if num*i<=max then temp[num*i]:=num;
                         try(dep+1, continue(temp), temp)
          end
end;
begin
      write('N='); readln(num);
      write('K='); readln(kind);
      s[1]:=1;
      for i:=1 to max do d[i]:=maxint;
      for i:=1 to num do d[i]:=i;
      d[0]:=0; maxcontinue:=0;
      try(2, continue(d), d);
      for i:=1 to kind do write(result[i], ' ');
      writeln;
      writeln('MAX=', maxcontinue-1);
end.
```

【运行示例】 (下划线表示输入)

N=7

K = 3
1 8 13
MAX = 69

N = 7
K = 4
1 5 24 37
MAX = 165

N = 10
K = 3
1 10 26
MAX = 146

N = 5
K = 5
1 4 9 31 51
MAX = 126

第六章　2000 年复赛试题解析

一、初中组第 1 题　计算器的改良

【问题描述】

NCL 是一家专门从事计算器改良与升级的实验室，最近该实验室收到了某公司所委托的一个任务：需要在该公司某型号的计算机上加上解一元一次方程的功能。实验室将这个任务交给了一个刚进入的新手 ZL 先生。为了很好地完成这个任务，ZL 先生首先研究了一些一元一次方程的实例：

4 + 3X = 8

6a - 5 + l = 2 - 2a

5 + 12Y　= 0

ZL 先生被主管告之，在计算器上键入的一个一元一次方程中，只包含整数、小写字母及 + 、- 、= 这三个数学符号（当然，符号“ - ”既可作减号，也可作负号）。方程中并没有括号，也没有除号，方程中的字母表示未知数。

问题求解

编写程序，解输入的一元一次方程，将解方程的结果（精确至小数点后三位）输出至屏幕。

你可假设对键入的方程的正确性的判断是由另一个程序员在做，或者说可认为键入的一元一次方程均为合法的，且有惟一实数解。

样　例

输入：

6a - 5 + l = 2 - 2a

输出：

a = 0. 750

【问题分析】

本题旨在考察选手处理字符串的能力，由于在计算器上键入的一个一元一次方程只包含整数、小写字母及 + 、- 、= 这三个数学符号，因此我们可以将 + 、- 号视作常数项或一次项系数的正负号，将连续的数字字符转换为一个整数，然后合并同类项，解出未知数即可。

【数据结构】

由于等号两侧的系数要分别计算，所以程序用 lcx 和 lc0 记录等号左边的一次项系数值和常数项值，用 rcx，rc0 记录等号右边的一次项系数值和常数项值。输入的一元一次方程存储在字符串变量 eqution 中，再将字符串 eqution 以等号为界一分为二，分别用字符串变量 left 和 right 存储。对 left 和 right 的扫描用过程 turn 实现，形式变量 ex 用于接受 left 或 right 的值，变量形式参数 cx，c0 将扫描求得的一次项系数值和常数项值传回主程序。

【算法分析】

本题的关键在于扫描字符串将其中的一次项系数值和常数项值分离出来，首先一次项和常数项一般以正负号或数字开始，一次项以字母结束，常数项则以数字结束，且两者结束之后的下一字符必为正负号，除非该项为最后一项，此时字符串结束。因此扫描字符串时以一项为一个单位，从一项的开始字符可确定该项系数的符号，若当前项以正负号开始，则这个正负号就是该项系数的符号，若当前项以数字开始，则表示正号，此时实际上是将正号省略掉了；类似地从一项的结束字符可确定该项系数是否为一次项，若当前项以字母结束，则当前项为一次项，否则当前项为常数项。另外当前项为一次项时，系数为 1 可以省略，此时按正常扫描将得到系数为 0，程序中对这一情况作了特殊处理，具体条件见程序清单。

【程序清单】

```
program c2000_1;
var i, lcx, lc0, rcx, rc0:longint;
    eqution, left, right:string;
procedure turn(ex:string; var cx, c0:longint);
var i, c:longint;
    flag:boolean;
    item:string;
begin
    i:=1;
    cx:=0;
    c0:=0;
    while i<=length(ex) do                    {循环一次为扫描其中的一项}
begin
  flag:=true;                                 {flag 表示当前的符号,true 表示正号}
  c:=0;
    repeat
        if ex[i]='-' then flag:=false;        {当前项开始字符为负号}
        if (ex[i]>='0') and (ex[i]<='9')      {处理数字字符}
          then c:=c*10+ord(ex[i])-48;         {当前数字加到系数之尾}
        i:=i+1;
    until (i>length(ex)) or (ex[i]='+') or (ex[i]='-');
      {当前项结束,此时扫描结束或遇到正负号,第 i-1 个字符为当前项结束字符}
if (c=0) and (not(ex[i-1] in ['0'..'9'])) and (ex[i-2]<>'0') then c:=1;
    {结束字符不是数字字符(而是字母)且其前一字符不为 0,但扫描到的系数为 0,
    此时即为一次项系数为 1 被省略的情况}
    if (ex[i-1]>='0') and (ex[i-1]<='9')      {合并同类项系数,条件表示常数项}
      then if flag then c0:=c0+c else c0:=c0-c
```

```
            else if flag then cx: = cx + c else cx: = cx - c;
        end;
end;
begin
        write('Input a simple eqution:');
        readln(eqution);
        i: =1;
        left: ='';
        while eqution[i] < >'=' do
        begin left: = left + eqution[i];   i: = i + 1 end;
        i: = i + 1;
        right: ='';
        while i < = length(eqution) do
        begin right: = right + eqution[i];   i: = i + 1 end;
        turn(left, lcx, lc0);
        turn(right, rcx, rc0);
        writeln('x =', (rc0 - lc0)/(lcx - rcx):0:3)
end.
```

BASIC 语言参考程序:

```
PRINT "Input a simple eqution:";
INPUT EQUTION $
I = 1
EX $(1) = ""
WHILE MID $(EQUTION $, I, 1) < > "="
      EX $(1) = EX $(1) + MID $(EQUTION $, I, 1)
      I = I + 1
WEND
I = I + 1
EX $(2) = ""
WHILE I < = LEN(EQUTION $)
      EX $(2) = EX $(2) + MID $(EQUTION $, I, 1)
      I = I + 1
WEND
FOR K = 1 TO 2
    I = 1
```

```
        CX(K) = 0
        C0(K) = 0
        WHILE I < = LEN(EX $(K))
         FLAG = 1
         C = 0
         DO
            IF MID $(EX $(K), I, 1) = "-" THEN FLAG = 0
            IF MID $(EX $(K), I, 1) > = "0" AND MID $(EX $ (K), I, 1)
< = "9" THEN
               C = C * 10 + ASC(MID $(EX $(K), I, 1)) - 48
            END IF
            I = I + 1
         LOOP UNTIL I > LEN(EX $(K)) OR MID $(EX $(K), I, 1) = "+"
OR MID $(EX $(K), I, 1) = "-"
       IF C = 0 AND (MID $(EX $(K), I - 1, 1) < "0" OR MID $(EX $(K),
I - 1, 1) > = "9") THEN
            IF I < = 2 THEN C = 1
            IF I > 2 THEN IF MID $(EX $(K), I - 2, 1) < > "0" THEN C = 1
         END IF
         IF MID $(EX $(K), I - 1, 1) > = "0" AND MID $(EX $(K), I -
1, 1) < = "9" THEN
            IF FLAG = 1 THEN C0(K) = C0(K) + C ELSE C0(K) = C0(K) - C
         ELSE
            IF FLAG = 1 THEN CX(K) = CX(K) + C ELSE CX(K) = CX(K) - C
         END IF
        WEND
    NEXT K
    PRINT "x =";  INT((C0(2) - C0(1)) / (CX(1) - CX(2)) * 1000 + .5) /
1000
```

【运行示例】（下划线表示输入）

Input a simple eqution:20 + 3x = -18

x = -12.667

Input a simple eqution: -6 + 12x = 0

x = 0.500

Input a simple eqution:47 - 2 = 6y + 3

x = 7.000

Input a simple eqution: -25a + 18 - 2 = -7a - 2
x = 1.000

Input a simple eqution: -a + 1a - 3 = a - 3
x = 0.000

二、初中组第2题　税收与补贴问题

【问题描述】

每样商品的价格越低,其销量就会相应增大。现已知某种商品的成本及其在若干价位上的销量(产品不会低于成本销售),并假设相邻价位间销量的变化是线性的且在价格高于给定的最高价位后,销量以某固定数值递减(我们假设价格及销售量都是整数)。

对于某些特殊商品,不可能完全由市场去调节其价格。这时候就需要政府以税收或补贴的方式来控制(所谓税收或补贴就是对于每个产品收取或给予生产厂家固定金额的货币)。

问题求解

你是某家咨询公司的项目经理,现在你已经知道政府对某种商品的预期价格,以及在各种价位上的销售情况。要求你确定政府对此商品是应收税还是补贴的最少金额(也为整数),才能使商家在这样一种政府预期的价格上,获取相对其他价位上的最大总利润。

总利润 = 单位商品利润 * 销量
单位商品利润 = 单位商品价格 - 单位商品成本(- 税金 or + 补贴)

输　入

输入的第一行为政府对某种商品的预期价,第二行有两个整数,第一个整数为商品成本,第二个整数为以成本价销售时的销售量,以下若干行每行都有两个整数,第一个为某价位时的单价,第二个为此时的销量,以一行 -1, -1 表示所有已知价位及对应的销量输入完毕,输入的最后一行为一个单独的整数表示在已知的最高单价外每升高一块钱将减少的销量。

输　出

输出有两种情况;若在政府预期价上能得到最大总利润,则输出一个单独的整数,数的正负表示是补贴还是收税,数的大小表示补贴或收税的金额最小值。若有多解,取绝对值最小的输出。

如在政府预期价上不能得到最大总利润,则输出"NO　SOLUTION"。

样　例

输入

31

28　130
30　120
31　110
-1　-1
15
输出
4

【问题分析】

本题在审题过程中必须先弄清楚以下几个概念：

1. 商品在某一销售价位（单位商品价格）上的总利润 = 单位商品利润 × 该价位上的销售量；

2. 单位商品利润 = 单位商品价格 - 单位商品成本（ - 税金 or + 补贴）；

其中，税金或补贴应理解为政府对每个商品收取或补贴的固定金额，对不同价位的销售，这种税金或补贴都是一样的。因此，如果政府对每个商品收取税金 X 元，相当于该商品的单位商品成本提高了 X 元，或者该单位商品利润减少了 X 元，如果政府对每个商品采用补贴 X 元，相当于该商品的单位商品成本降低了 X 元，或者该单位商品利润增加了 X 元；税金或补贴对商品销售总利润的调节作用表现在减少或增加了各个价位上的总利润。假设政府对每个商品收取 X 元税金，那么某一价位上商品总利润要减少 X 乘以该价位上的销售量，就是说，销量大的价位被减少的利润多，销量小的价位被减少的利润少。若利用补贴调节，则销量大的价位增加的利润多，销量小的价位增加的利润少。因此，政府可采用税金或补贴使得某个商品在预期价格上相对于其他价位有最大的利润。下表用题目给出的样例来说明这种调节作用。

单价	销量	单位成本	在税金 - X 元或补贴 X 元下的总利润								
			0	-1	+1	-2	+2	-3	+3	-4	+4
28	130	28	0	-130	130	-260	+260	-390	390	-520	520
29	125	28	125	0	250	-125	375	-250	500	-375	625
30	120	28	240	120	360	0	180	-120	600	-240	720
31	110	28	330	220	440	110	550	0	660	-110	770
32	95	28	380	285	475	190	570	95	665	0	760
33	80	28	400	320	480	240	560	160	640	80	720
34	65	28	390	325	455	260	520	195	585	130	650
35	50	28	350	300	400	250	450	200	500	150	550
36	35	28	280	245	315	210	350	175	385	140	440
37	20	28	180	160	200	140	220	120	240	100	260
38	5	28	50	45	55	40	60	35	65	30	70

表中第 1 列为各整数价位，第 2 列为对应价位的销量（题中输入数据中没有列出的价位及销量，可根据题目假设："相邻价位同销量的变化是线性的"计算得到），第 3 列为单位商品成本。

从表的第4列开始,给出了在政府采用收税或补贴下各价位的总利润,税收或补贴为零时,政府预期价位上的总利润为330,而总利润最大的价位是33,当税收从1开始逐一递增时(税收用负数表示),处在预期价位上面的商品各价位总利润减少幅度较大,而处在预期价位下面的各价位总利润减少的幅度较小,而补贴从1开始逐一递增时,上面价位的总利润增加的幅度比下面价位大。因此,政府采取税收或补贴,可调节各价位的总利润之间的大小关系。从表中可知,当补贴为4时,政府预期价位上的总利润最大。

综上所述,根据输入数据,列出各价位在政府税收或补贴为零时的总利润(不妨称为税前利润);以政府预期价位为基准,价位高于预期价位的称高价位段,价位低于预期价位的称低价位段。若利润高于预期价位上利润的价位仅出现在高价位段,就采用补贴的办法进行调节;反之,假如利润高于预期价位上利润的价位仅出现在低价位段,就采用税收的办法进行调节。每次调节从税金或补贴最小的开始,逐步增加,直到预期价位取得相对于其他价位的总利润最大为止。

预期价位上不能得到最大利润的判断:从以上分析可知,不同价位上的总利润在相同的税收或补贴下增(减)幅是不同的,如果高价位段和低价位段都出现总利润高于预期价总利润的价位,这时政府的调控就无法实现。

例如表1中将价位30的销售量改为130,根据税前利润情况,要使预期价位31为最大总利润,采用补贴的方法。下面列出30、31、32三种价位在补贴从1变化到4时,预期价位的总利润小于上面价位的总利润,出现此类情形,政府的补贴调控无法达到目的。若采用税收调节,出现低价位总利润大于预期价位总利润,此时,这种调节无法进行下去。

除以上之外,下面两种情形也应该认为调控无法实现(1)虽然预期价位总利润高于其他价位,但此时的总利润小于等于零;(2)预期价销售量为零。

			0	1	2	3	4
30	130	28	260	390	520	650	780
31 *	110	28	330	440	550	660	770
32	95	28	380	475	570	665	760

【数据结构】

程序用一维数组p和v存放从成本价开始的各价位及相应的销售量。

【算法分析】

本题的关键在于如何正确理解问题本身,实际的算法并不困难,只需采用穷举的方法求解即可,具体细节详见程序清单及注释。

【程序清单】

```
program c2000_2;
var a, b, delta, i, j, k, last, maxv, highest, try:longint;
    p, v:array[0..8000] of longint;
begin
```

```
write('Input highest price:');
readln(highest);
i:=0;
readln(a, b);
p[0]:=a-1;
while a<>-1 do                          {数据输入及按线性建立完整的价位表}
begin
    if a=p[i]+1
       then begin i:=i+1;  p[i]:=a;  v[i]:=b end
       else begin delta:=(v[i]-b) div (a-p[i]);
                  for j:=p[i]+1 to a do
                  begin i:=i+1;
                        p[i]:=p[i-1]+1;
                        v[i]:=v[i-1]-delta
                  end;
            end;
    readln(a, b)
end;
readln(delta);                          {delta 为线性增量}
repeat
     i:=i+1;
     p[i]:=p[i-1]+1;
     v[i]:=v[i-1]-delta;
until v[i]<0;
last:=i-1;
try:=0;                                 {try 为枚举量}
writeln('Result is:');
while true do
begin
    maxv:=-maxint;
    for i:=1 to last do
        if ((p[i]-p[1])*v[i]+try*v[i]>maxv) or
           ((p[i]-p[1])*v[i]+try*v[i]=maxv) and
           (p[i]=highest) then
        begin maxv:=(p[i]-p[1])*v[i]+try*v[i]; k:=p[i] end;
    if k=highest then begin writeln(try);  halt end;
    maxv:=-maxint;
```

```
            for i:=1 to last do
                if ((p[i]-p[1])*v[i]-try*v[i]>maxv) or
                    ((p[i]-p[1])*v[i]-try*v[i]=maxv) and
                    (p[i]=highest) then
                    begin maxv:=(p[i]-p[1])*v[i]-try*v[i]; k:=p[i] end;
            if k=highest then begin writeln(-try); halt end;
            try:=try+1
        end;
end.
```

BASIC 语言参考程序:

```
    DIM P(8010), V(8010)
    INPUT "Input highest price:";  HIGHEST
    I = 0
    INPUT A, B
    P(0) = A - 1
    WHILE A <> -1
        IF A = P(I) + 1 THEN
            I = I + 1
            P(I) = A
            V(I) = B
        ELSE
            DELTA = (V(I) - B) \ (A - P(I))
            FOR J = P(I) + 1 TO A
                I = I + 1
                P(I) = P(I - 1) + 1
                V(I) = V(I - 1) - DELTA
            NEXT J
        END IF
        INPUT A, B
    WEND
    INPUT DELTA
    DO
      I = I + 1
      P(I) = P(I - 1) + 1
      V(I) = V(I - 1) - DELTA
```

```
    LOOP UNTIL V(I) < 0
    LAST = I - 1
    TRY = 0
    PRINT "Result is:";
    WHILE 1 = 1
        MAXV = -10000
        FOR I = 1 TO LAST
    IF (P(I) - P(1)) * V(I) + TRY * V(I) > MAXV OR (P(I) - P(1)) * V
(I) + TRY * V(I) = MAXV AND P(I) = HIGHEST THEN
            MAXV = (P(I) - P(1)) * V(I) + TRY * V(I)
            K = P(I)
          END IF
        NEXT I
        IF K = HIGHEST THEN PRINT TRY: END
        MAXV = -MAXINT
        FOR I = 1 TO LAST
          IF (P(I) - P(1)) * V(I) - TRY * V(I) > MAXV OR (P(I) - P
(1)) * V(I) - TRY * V(I) = MAXV AND P(I) = HIGHEST THEN
            MAXV = (P(I) - P(1)) * V(I) - TRY * V(I)
            K = P(I)
          END IF
        NEXT I
        IF K = HIGHEST THEN PRINT -TRY: END
        TRY = TRY + 1
    WEND
```

【运行示例】（下划线表示输入）

```
Input highest price:315
280 1300
300 1200
310 1100
-1 -1
150
Result is:
-32

Input highest price:77
74 280
```

75 266
76 264
77 260
78 239
-1 -1
50
Result is:
9

Input highest price:4011
1 79990
7999 10
-1 -1
10
Result is:
-20

【小结】

需要注意的是:当在政府预期的价位上取得最大利润时,即使在其他价位上也能取得相同的最大利润值,此时应视为已找到解。

三、初中组第3题/高中组第2题　乘积最大

【问题描述】

今年是国际数学联盟确定的"2000——世界数学年",又恰逢我国著名数学家华罗庚先生诞辰90周年。在华罗庚先生的家乡江苏金坛,组织了一场别开生面的数学智力竞赛的活动,你的一个好朋友XZ也有幸得以参加。活动中,主持人给所有参加活动的选手出了这样一道题目:

设有一个长度为N的数字串,要求选手使用K个乘号将它分成K+1个部分,找出一种分法,使得这K+1个部分的乘积能够为最大。

同时,为了帮助选手能够正确理解题意,主持人还举了如下的一个例子:

有一个数字串:312,当N=3,K=1时会有以下两种分法:

1) 3*12=36
2) 31*2=62

这时,符合题目要求的结果是:31*2=62

现在,请你帮助你的好朋友XZ设计一个程序,求得正确的答案。

输　入

程序的输入共有两行：

第一行共有 2 个自然数 N，K(6≤N≤40，1≤K≤6)

第二行是一个长度为 N 的数字串。

输　出

结果显示在屏幕上，相对于输入，应输出所求得的最大乘积(一个自然数)。

样　例

输入

4　2

1231

输出

62

【问题分析】

首先由于问题的规模为 6≤N≤40，1≤K≤6，要在长为 N 的数字串中插入 K 个乘号使得乘积最大，显然结果将会超出长整型数的范围，所以运算时要用到高精度乘法运算算法。其次，让我们考虑一下穷举的算法行不行，根据组合数学的知识可知，在长为 N 的数字串中插入 K 个乘号的方案共有 C_{n-1}^{k} 种，当 N = 40，K = 6 时，将会有超过 6500000 种的插入方案，而高精度乘法运算本身也较费时间，由此可以得出结论：穷举算法将会超时，必须寻找更为高效的算法，看来只有动态规划算法才能在规定时间内算出问题的解。那么如何才能找到动态规划算法的正确思路呢？问题中只有乘号插入的位置是变化的，取数字串中的任意一段来考虑，其中均可能包含任意个(少于或等于 K 个)乘号，若能求出在任意一段插入 K - 1 个乘号的最大乘积，则只需穷举第 K 个乘号的插入位置 P，该乘号将数字串分割成前后两段，前半段包含 K - 1 个乘号，其最大值根据假设已经算出，将它与后半段的值相乘得到第 K 个乘号在位置 P 时的最大乘积，取所有的第 K 个乘号不同插入方案中的最大乘积即为问题的解。类似地求在任意一段中插入 K - 1 个乘号的最大乘积时，需预先算出在任意一段中插入 K - 2 个乘号时的最大乘积，依次类推，求在任意一段中插入 1 个乘号的最大乘积时，需预先算出在任意一段中插入 0 个乘号时的最大乘积，此时的值为已知值，即为该段数字构成的数的值。最后我们可以得出递推式，设 c[p1，p2，k]表示在长为 N 的数字串中，从第 p1 个数字到第 p2 个数字之间插入 k 个乘号的最大乘积，则 c[p1，p2，0]为第 p1 个数字到第 p2 个数字之间全部数字构成的数的值，当 k≥1 时，有 $c[p1, p2, k] = \max_{p=p1+k-1}^{p2-1}(c[p1, p, k-1] * c[p+1, p2, 0])$。

【数据结构】

程序中用一维整型数组 d 存储键盘输入的数字串，用二维数组 max 存储在任意一段中插入若干个乘号时的最大乘积，数组元素的第一下标表示该段的开始位置，第二下标表示该段的结束位置，由于最大乘积可能超过长整型数，所以二维数组 max 的每个元素是一个用一维数组存储的高精度数，实际上数组 max 是一个三维数组，具体说明如下：

```
const maxn = 40;
type arr1 = array [0..maxn - 1] of byte;    {该类型用于存储高精度数}
     arr2 = array [1..maxn, 1..maxn] of arr1;
var max: arr2;
```

【算法分析】

由于计算在任意一段中插入k个乘号的最大乘积只与在任意一段中插入k-1个乘号的最大乘积相关,因此程序中借助于一个类型为arr2的临时数组temp来推导每增加一个乘号后任意一段的最大乘积,具体细节见程序清单与注释。

【程序清单】

```
program c2000_3;
const maxn=40;
type arr1=array[0..maxn-1] of byte;
     arr2=array[1..maxn,1..maxn] of arr1;
var i,j,k,m,n,w:integer;
    ch:char;
    c:arr1;
    d:array[1..maxn] of integer;
    max:arr2;
procedure calculate(i:integer;  var max:arr2);
                                        {计算在任意一段中插入i个乘号的最大乘积}
var j,m,p,p1,p2,w:integer;
    temp:arr2;
begin
  for p1:=1 to n-i do
    for p2:=p1+i to n do
    begin
      for j:=0 to maxn-1 do temp[p1,p2,j]:=0;
      for p:=p1+i-1 to p2-1 do
      begin
        for j:=0 to maxn-1 do c[j]:=0;
        m:=p2;  w:=0;
        while m>p do
        begin
          for j:=0 to n-1 do
            if max[p1,p,j]<>0 then c[j+w]:
              =c[j+w]+max[p1,p,j]*d\[m\];
          for j:=0 to n-2 do
          begin
            c[j+1]:=c[j+1]+c[j] div 10;
            c[j]:=c[j] mod 10
          end;
```

```
                    m:=m-1;
                    w:=w+1
                end;
                j:=n-1;
while (j>0) and (c[j]=temp[p1, p2, j]) do j:=j-1;
                                        {比较两个高精度数的大小}
                if c[j]>temp[p1, p2, j] then temp[p1, p2]:=c;
          end
      end;
   max:=temp
end;
begin
      write('Input n, k=');
      readln(n, k);
      write('Input digit string:');
      for i:=1 to n do
      begin
            read(ch);
            d[i]:=ord(ch)-48
      end;
      readln;
      for i:=1 to n-1 do
          for j:=i to n do
          begin
                for m:=0 to maxn-1 do max[i, j, m]:=0;
                w:=0;                   {变量 w 用于记录高精度数的有效位数}
                for m:=j downto i do
                begin max[i, j, w]:=d[m];w:=w+1 end;
          end;
      for i:=1 to k do calculate(i, max);
      j:=n-1;
      while (j>0) and (max[1, n, j]=0) do j:=j-1;
      for i:=j downto 0 do write(max[1, n, i]);
      writeln
end.
```

BASIC 语言参考程序:

```
INPUT "Input n, k = "; N, K
DIM NUM(N, N, N), MAX(N, N, N), TEMP(N, N, N), C(N)
INPUT "Input digit string:"; D $
FOR I = 1 TO N
   FOR J = I TO N
      T = -1
      FOR P = J TO I STEP -1
         T = T + 1
         NUM(I, J, T) = ASC(MID $(D $, P, 1)) - 48
      NEXT P
      FOR P = 0 TO N
         MAX(I, J, P) = NUM(I, J, P)
      NEXT P
   NEXT J
NEXT I

FOR I = 1 TO K
    FOR P1 = 1 TO N - 1
       FOR P2 = P1 + I TO N
          FOR J = 0 TO N
             TEMP(P1, P2, J) = 0
          NEXT J
          FOR M = P1 + I - 1 TO P2 - 1
             FOR J = 0 TO N
                C(J) = 0
             NEXT J
             FOR J1 = 0 TO N
                FOR J2 = 0 TO N
                  IF MAX(P1, M, J1) < > 0 AND NUM(M + 1, P2, J2) < > 0
THEN C(J1 + J2) = C(J1 + J2) + MAX(P1, M, J1) * NUM(M + 1, P2, J2)
                  END IF
                NEXT J2
             NEXT J1
             FOR J = 0 TO N - 1
                C(J + 1) = C(J + 1) + C(J) \ 10
                C(J) = C(J) MOD 10
```

```
                NEXT J
                J = N
                WHILE J > 0 AND C(J) = TEMP(P1, P2, J)
                      J = J - 1
                WEND
                IF C(J) > TEMP(P1, P2, J) THEN
                   FOR J = 0 TO N
                       TEMP(P1, P2, J) = C(J)
                   NEXT J
                END IF
            NEXT M
        NEXT P2
    NEXT P1
    FOR P1 = 1 TO N - 1
        FOR P2 = P1 TO N
            FOR J = 0 TO N
                MAX(P1, P2, J) = TEMP(P1, P2, J)
            NEXT J
        NEXT P2
    NEXT P1
NEXT I
H = N
WHILE MAX(1, N, H) = 0
      H = H - 1
WEND
FOR I = H TO 0 STEP -1
PRINT CHR $(MAX(1, N, I) + 48);
NEXT I
PRINT
```

【运行示例】（下划线表示输入）

```
Input n, k = 6 1
Input digit string: 101010
10100

Input n, k = 9 4
Input digit string: 321044105
5166000
```

Input n, k = 8 3
Input digit string:22222222
234256

Input n, k = 10 5
Input digit string:7777777777
1722499009

四、初中组第4题/高中组第3题　单词接龙

【问题描述】

单词接龙是一个与我们经常玩的成语接龙相类似的游戏,现在我们已知一组单词,且给定一个开头的字母,要求出以这个字母开头的最长的“龙”(每个单词都最多在“龙”中出现两次),在两个单词相连时,其重合部分合为一部分,例如 beast 和 astonish,如果接成一条龙则变为 beastonish,另外相邻的两部分不能存在包含关系,例如 at 和 atide 间不能相连。

输　入

输入的第一行为一个单独的整数 n(n≤20)表示单词数,以下 n 行每行有一个单词,输入的最后一行为一个单个字符,表示“龙”开头的字母。你可以假定以此字母开头的“龙”一定存在。

输　出

只需输出以此字母开头的最长的“龙”的长度

样　例

输入
5
at
touch
cheat
choose
tact
a
输出
23(连成的“龙”为 atoucheatactactouchoose)

【问题分析】

本题明显是要考搜索,至于是使用宽度优先搜索还是使用深度优先搜索,这要视选手本人的情况而定,一般是对哪种算法熟悉就采用哪一种。实际上使用两种算法各有所长,也各有所短,宽度优先搜索速度较快,对存储空间要求较高;而深度优先搜索则不存在内存不足的问题,但运行速度相对较慢,对数据量不大的情况,采用任何一种方法均可。

【数据结构】

一般来说，大凡搜索问题在开始搜索之前要尽量做好准备工作，如搜索中要反复使用的数据要预先算好并存储，而不应在搜索时再临时去计算。另外，选择合理的数据结构也十分重要，这样做一方面能够节省存储空间，另一方面可以降低数据的运算复杂度，如本题中搜索中一直要进行两个单词是否可以连接的判断，因此预先就应该用一个邻接矩阵将之记录起来。程序最后要输出最长的龙的长度，则我们不仅要知道某一个单词是否可以与龙尾的单词相连，而且还要知道如果该单词可以与龙尾的单词相连的话，将之连接在龙尾后，龙的长度增加了多少，因此邻接矩阵的定义如下：

$$link[i,j]=\begin{cases}0 & \text{单词 j 不能接在单词 i 之后}\\ len & \text{单词 j 能接在单词 i 之后，接上去之后龙的长度增加 len}\end{cases}$$

由于只要求计算出最长的龙的长度，因此队列元素只需要记录龙的长度，而不需要记录龙对应的字符串，这样就可以节省大量的存储空间。另外题目中规定每个单词最多只能使用两次，因此搜索时要记录下当前结点中的龙是由哪些单词构成的，每个单词分别被使用了几次。综上所述，数据结构定义如下：

```
type node = record state:array [1..maxn] of byte; {记录结点中的龙的单词构成情况}
                last:byte; {记录结点中的龙的最后一个单词的序号}
                len:longint {记录结点中的龙的长度}
     end;
```

【算法分析】

计算 link 数组的算法如下：

```
for i:=1 to n do
  for j:=1 to n do
  begin
     link[i,j]:=0;
     find:=false;
     len:=1; {变量 len 表示单词 i 与单词 j 重叠部分的长度}
     while (len<length(word[i])) and (len<length(word[j])) and (not(find)) do
     begin  {判断单词 i 从 p1 位置开始到最后一个字母为止的子串是否与单词 j 的开头部分长为 len 的子串相同}
           p1:=length(word[i])-len+1;
           p2:=1;
           while (p1<=length(word[i])) and (word[i,p1]=word[j,p2]) do
           begin p1:=p1+1;  p2:=p2+1 end;
           if p1>length(word[i]) then find:=true else len:=len+1
     end;
     if find then link[i,j]:=length(word[j])-len
end;
```

下面以样例为例说明宽度优先搜索的过程，请看下表：

结点序号	当前结点所对应的龙	父结点序号
1	at = at	
2	atouch = at + touch	1
3	atact = at + tact	1
4	atoucheat = atouch + cheat	2
5	atouchoose = atouch + choose	2
6	atactouch = atact + touch	3
7	atactact = atact + tact	3
8	atoucheatouch = atoucheat + touch	4
9	atoucheatact = atoucheat + tact	4
10	atactoucheat = atactouch + cheat	6
11	atactouchoose = atactouch + choose	6
12	atactactouch = atactact + touch	7
13	atoucheatoucheat = atoucheatouch + cheat	8
14	atoucheatouchoose = atoucheatouch + choose	8
15	atoucheatactouch = atoucheatact + touch	9
16	atoucheatactact = atoucheatact + tact	9
17	atactoucheatouch = atactoucheat + touch	10
18	atactoucheatact = atactoucheat + tact	10
19	atactactoucheat = atactactouch + cheat	12
20	atactactouchoose = atactactouch + choose	12
21	atoucheatoucheatact = atoucheatoucheat + tact	13
22	atoucheatactoucheat = atoucheatactouch + cheat	15
23	Atoucheatactouchoose = atoucheatactouch + choose	15
24	atoucheatactactouch = atoucheatactact + touch	16
25	atactoucheatoucheat = atactoucheatouch + cheat	17
26	Atactoucheatouchoose = atactoucheatouch + choose	17
27	atactoucheatactouch = atactoucheatact + touch	18
28	atactactoucheatouch = atactactoucheat + touch	19
29	Atoucheatoucheatactact = atoucheatoucheatact + tact	21
30	Atoucheatactoucheatact = atoucheatactoucheat + tact	22
31	Atoucheatactactoucheat = atoucheatactactouch + cheat	24
32	Atoucheatactactouchoose = atoucheatactactouch + choose	24
33	Atactoucheatoucheatact = atactoucheatoucheat + tact	25
34	Atactoucheatactoucheat = atactoucheatactouch + cheat	27
35	Atactoucheatactouchoose = atactoucheatactouch + choose	27
36	Atactactoucheatoucheat = atactactoucheatouch + cheat	28
37	Atactactoucheatouchoose = atactactoucheatouch + choose	28

首先将所有满足开头字母为指定字母的单词放进队列,作为初始队列,然后每次从队头取出一个元素进行扩展,将扩展出的新结点(接上一个新单词之后的龙)增加在队尾,直到不再

有新结点产生为止。最后扫描整个队列，将队列中最长的龙的长度求出即可。

以下给出宽度优先搜索的详细算法：

```
head:=1;  {head 为队头指针,tail 指向队尾}
while head<tail do
begin
        for i:=1 to n do{尝试在队头元素之后接上第 i 个单词}
        if (q[head].state[i]<2) and (link[q[head].last, i]>0) then
          with q[tail] do  {将新元素添加到队尾}
          begin
              for j:=1 to n do state[j]:=q[head].state[j];
              state[i]:=state[i]+1;
              last:=i;
              len:=q[head].len+link[q[head].last, i];
              tail:=tail+1
          end;
    head:=head+1
 end;
```

【程序清单】

```
program c2000_4;
const maxn=20;
type node=record state:array [1..maxn] of byte;
                 last:byte;
                 len:longint
           end;
var i, j, n, p1, p2, head, tail, len, maxlen:longint;
    head_ch:char;
    find:boolean;
    word:array[1..maxn] of string;
    link:array[1..maxn, 1..maxn] of longint;
    q:array [1..2300] of node;
begin
  readln(n);
  for i:=1 to n do readln(word[i]);
  read(head_ch);
  for i:=1 to n do
    for j:=1 to n do
    begin
```

```
link[i, j]: =0;
find: =false;
len: =1;                          {变量 len 表示单词 i  与单词 j 重叠部分的长度 }
while (len < length(word[i])) and (len < length(word[j])) and (not(find)) do
  begin                           {判断单词 i 从 p1 位置开始到最后一个字母为止的子
                                  串是否与单词 j 的开头部分长为 len 的子串相同}
      p1: =length(word[i]) -len +1;
      p2: =1;
      while (p1 < =length(word[i])) and (word[i, p1] =word[j, p2]) do
      begin p1: =p1 +1;   p2: =p2 +1 end;
      if p1 >length(word[i]) then find: =true else len: =len +1
  end;
  if find then link[i, j]: =length(word[j]) -len
end;
tail: =1;
 for i: =1 to n do
     if word[i, 1] =head_ch then
        with q[tail] do
        begin
              for j: =1 to n do state[j]: =0;
              state[i]: =1;
              last: =i;
              len: =length(word[i]);
              tail: =tail +1
        end;
 head: =1;
 while head <tail do
 begin
      for i: =1 to n do
          if (q[head].state[i] <2) and (link[q[head].last, i] >0) then
             with q[tail] do
             begin
                   for j: =1 to n do state[j]: =q[head].state[j];
                   state[i]: =state[i] +1;
                   last: =i;
                   len: =q[head].len +link[q[head].last, i];
                   tail: =tail +1
```

```
                    end;
                head:=head+1
            end;
            maxlen:=q[1].len;
            for i:=2 to tail-1 do
                if maxlen<q[i].len then maxlen:=q[i].len;
            writeln(maxlen)
end.
```

BASIC 语言参考程序：

```
MAXN = 20
DIM WORD(MAXN), LINK(MAXN, MAXN), STATE(500, MAXN), LENGTH(500),
LAST(500)
INPUT "INPUT N:"; N
FOR I = 1 TO N
    PRINT "WORD("; I; ")";
    INPUT WORD $(I)
NEXT I
INPUT "INPUT HEAD LETTER:"; HEADCH $
FOR I = 1 TO N
   FOR J = 1 TO N
      LINK(I, J) = 0
      FIND = 0
      D = 1
      WHILE D < LEN(WORD $(I)) AND D < LEN(WORD $(J)) AND FIND = 0
       P1 = LEN(WORD $(I)) - D + 1
       P2 = 1
       WHILE P1 <= LEN(WORD $(I)) AND MID $(WORD $(I), P1, 1) =
MID $(WORD $(J), P2, 1)
            P1 = P1 + 1
            P2 = P2 + 1
       WEND
       IF P1 > LEN(WORD $(I)) THEN FIND = 1 ELSE D = D + 1
      WEND
      IF FIND = 1 THEN LINK(I, J) = LEN(WORD $(J)) - D
   NEXT J
```

```
NEXT I
TAIL = 1
FOR I = 1 TO N
    IF MID $(WORD $(I), 1, 1) = HEADCH $ THEN
        FOR J = 1 TO N
            STATE(TAIL, J) = 0
        NEXT J
        STATE(TAIL, I) = 1
        LAST(TAIL) = I
        LENGTH(TAIL) = LEN(WORD $(I))
        TAIL = TAIL + 1
    END IF
NEXT I
HEAD = 1
WHILE HEAD < TAIL
      FOR I = 1 TO N
          IF STATE(HEAD, I) < 2 AND LINK(LAST(HEAD), I) > 0 THEN
             FOR J = 1 TO N
                 STATE(TAIL, J) = STATE(HEAD, J)
             NEXT J
             STATE(TAIL, I) = STATE(TAIL, I) + 1
             LAST(TAIL) = I
             LENGTH(TAIL) = LENGTH(HEAD) + LINK(LAST(HEAD), I)
             TAIL = TAIL + 1
          END IF
      NEXT I
      HEAD = HEAD + 1
WEND
MAXLEN = LENGTH(1)
FOR I = 2 TO TAIL - 1
    IF MAXLEN < LENGTH(I) THEN MAXLEN = LENGTH(I)
NEXT I
PRINT MAXLEN
```

【运行示例】 (下划线表示输入)

```
1
ENVELOPE
E
```

Maxlen = 15

2
ABABABAB
ABABABC
A
Maxlen = 19

4
ABABABC
ABABABD
ABABABA
CDABABA
A
Maxlen = 43

8
NO
NEW
NAME
NEVER
NATIONAL
NECESSARY
EVER
ME
N
Maxlen = 9

6
ACT
TOUCH
CHEAT
CHOOSE
TACT
SENCITIVE
A
Maxlen = 31

6

MANY
YOUTH
THIS
SYSTEM
MAIN
NAVY
M
Maxlen = 38

【小结】

使用深度优先搜索算法在数据结构与 link 数组的计算方面与宽度优先搜索算法一致,在算法的实现上可采用递归或回溯法实现,这里给出递归算法如下:

```
procedure search(current:node);                    {current 表示当前龙的状态}
var i:longint;
    temp:node;
begin
    for i:=1 to n do                               {尝试在队头元素之后接上第 i 个单词}
        if (current.state[i]<2) and (link[current.last, i]>0) then
            with temp do                           {temp 记录当前龙加上单词 i 之后的状态}
            begin
                for j:=1 to n do state[j]:=current.state[j];
                state[i]:=state[i]+1;
                last:=i;
                len:=current.len+link[current.last, i];
                if maxlen<len then maxlen:=len;
                search(temp)
            end
end;
```

主程序中只要将所有开头字母与指定的开头字母相同的单词作为龙的初始状态去调用递归过程 search 即可求出问题的解,详细算法如下:

```
maxlen:=0;
for i:=1 to n do
    if word[i, 1]=head_ch then                     { head_ch 表示指定的开头字母}
        with current do
        begin
            for j:=1 to n do state[j]:=0;
            state[i]:=1;
            last:=i;
```

```
            len:=length(word[i]);
        if maxlen<len then maxlen:=len;
        search(current)
    end;
```

五、高中组第1题　进制转换

【问题描述】

我们可以用这样的方式来表示一个十进制数:将每个阿拉伯数字乘以一个以该数字所处位置的(值减1)为指数,以10为底数的幂之和的形式。例如:123可表示为$1*10^2+2*10^1+3*10^0$这样的形式。

与之相似的,对二进制数来说,也可表示成每个二进制数码乘以一个以该数字所处位置的(值-1)为指数,以2为底数的幂之和的形式。一般说来,任何一个正整数R或一个负整数-R都可以被选来作为一个数制系统的基数。如果是以R或-R为基数,则需要用到的数码为0,1,…,R-1。例如,当R=7时,所需用到的数码是0,1,2,3,4,5和6,这与其是R或-R无关。如果作为基数的数绝对值超过10,则为了表示这些数码,通常使用英文字母来表示那些大于9的数码。例如对16进制数来说,用A表示10,用B表示11,用C表示12,用D表示13,用E表示14,用F表示15。在负进制数中是使用-R作为基数,例如-15(十进制)相当于110001(-2进制),并且它可以被表示为2的暴级数的和数:

$$110001=1*(-2)^5+1*(-2)^4+0*(-2)^3+0*(-2)^2+0*(-2)^1+1*(-2)^0$$

问题求解

设计一个程序,读入一个十进制数和一个负进制数的基数,并将此十进制数转换为此负进制下的数:$-R\in\{-2,-3,-4,\cdots,-20\}$

输　入

输入的每行有两个输入数据。

第一个是十进制数N($-32768\leqslant N\leqslant 32767$);第二个是负进制数的基数-R。

输　出

结果显示在屏幕上,相对于输入,应输出此负进制数及其基数,若此基数超过10,则参照16进制的方式处理。

样　例

输入

```
30000    -2
-20000   -2
28800    -16
-25000   -16
```

输出

30000 = 11011010101110000 (base 2)

-20000 = 1111011000100000 (base 2)

28000 = =19180 (bese 16)

25000 = 7FB8 (base 16)

【问题分析】

拿到本题,首先会想到进制转换中的转换规则"除 R 取余,除尽取倒"。很多选手就从这点出发去寻找规律,结果是花了很长时间也没有能够找出规律来,最后不得不采用穷举的方法。实际上,由于题目中给出的测试数据范围较小,仅为 -32767 到 32767,用穷举法完全可以彻底解决本问题,穷举时只需从 0 开始逐一穷举 -R 进制数,对被穷举的 -R 进制数算出其对应的十进制值,若该值等于键盘输入的十进制数 N,则输出该 -R 进制数,结束程序,否则重复考察下一个 -R 进制数,直到找到要求的 -R 进制数为止。

【数据结构】

其中 -R 进制数用数组 d 保存,d[0]表示最低位。

【算法分析】

穷举算法具体细节如下:

```
m:=1;
置数组 d 全零;
d[1]:=-1;
repeat
        d[1]:=d[1]+1;
        i:=1;
        while d[i]=r do
        begin
                d[i]:=0;
                i:=i+1;
                d[i]:=d[i]+1
        end;
        if i>m then m:=i;
        s:=0;
        for i:=m downto 1 do s:=s*(-r)+d[i]
until n=s;
for i:=m downto 1 do if d[i]<=9 then write(d[i]) else write(chr(d[i]+55));
```

【程序清单】

```
program g2000_1;
var i, m, n, r, s:longint;
    a:array[1..100] of longint;
```

```
begin
    write('Input n, -r:');
    readln(n, r);
    r:=-r;
    write(n, '=');
    m:=1;
    for i:=1 to 100 do a[i]:=0;
    a[1]:=-1;
    repeat
        a[1]:=a[1]+1;
        i:=1;
        while a[i]=r do
        begin
            a[i]:=0;          i:=i+1;          a[i]:=a[i]+1
        end;
        if i>m then m:=i;
        s:=0;
        for i:=m downto 1 do s:=s*(-r)+a[i]
    until n=s;
    for i:=m downto 1 do
        if a[i]<=9 then write(a[i]) else write(chr(a[i]+55));
    writeln('    (base ', r, ')')
end.
```

【运行示例】（下划线表示输入）

```
Input n, -r:27993 -8
27993=72651     (base 8)

Input n, -r:-37336 -16
-37336=AFE8     (base 16)

Input n, -r:-569 -2
-569=1011011011     (base 2)

Input n, -r:-304 -20
-304=GG     (base 20)

Input n, -r:683 -2
683=11111111111     (base 2)
```

【小结】

虽然程序中将一个 -R 进制数转换为十进制数 s 使用秦九韶算法,即使如此,穷举法的效率还是十分低的,若输入数据范围扩大到长整型范围,则穷举法显然将超时,那么十进制整数与 -R 进制数之间的转换规则是否存在呢?答案是肯定的,只不过靠观察是很难观察出来的,最好的办法是从 -R 进制数的定义出发来推导其与十进制数的转换规则。

设 $N=(d_m\ d_{m-1}\cdots d_1\ d_0)_{-R}$

$=d_m*(-R)^m+d_{m-1}*(-R)^{m-1}+\cdots+d_1*(-R)^1+d_0*(-R)^0$

其中 $0\leqslant d_i<R$,$i=0,1,2,\ldots,m$,除最后一项外,其余各项均为 -R 的倍数,因此上式可写成:

$$N=Q*(-R)+d_0 \qquad (1)$$

$$d_0=N+QR$$

$$0\leqslant N+QR<R$$

$$\frac{-N}{R}\leqslant Q<\frac{-N}{R}+1$$

所以 Q 为区间$\left[\frac{-N}{R},\ \frac{-N}{R}+1\right]$中的惟一整数

而 $Q=d_m*(-R)^{m-1}+d_{m-1}*(-R)^{m-2}+\cdots+d_1*(-R)^0$

重复上述过程,即可依次求出 $d_1,d_2,\cdots,d_m$,在 PASCAL 语言中求(1)式中的 Q 可使用 N 与 -R 经过 div 运算和 mod 运算求得,由 div 运算和 mod 运算的定义可知:

$N=(N\ \mathrm{div}(-R))*(-R)+N\ \mathrm{mod}(-R)$

其中,$-R<N\ \mathrm{mod}(-R)<R$,所以当 $N\ \mathrm{mod}(-R)\geqslant 0$ 时,$Q=N\ \mathrm{div}(-R)$;

当 $N\ \mathrm{div}(-R)<0$ 时,有 $N=(N\ \mathrm{div}(-R)+1)*(-R)+(R+N\ \mathrm{mod}(-R))$,$Q=N\ \mathrm{div}(-R)+1$。

具体的算法如下:

```
m:=0;
while n<>0 do
begin
    q:=n div (-r);
    if n mod (-r)<0 then q:=q+1;
    m:=m+1;          d[m]:=n-q*(-r);          n:=q
end;
for i:=m downto 1 do
    if d[i]<=9 then write(d[i]) else write(chr(d[i]+55));
```

实际上,进制转换的算法是递归的,(1)式中只要求出了 Q 对应的—R 进制数,再加上末位的 d_0 即得到了 N 对应的—R 进制数,以下给出相应的递归过程:

```
procedure print(n:longint);
```

```
    var q:longint;
begin
    if n<>0 then      begin
                        q:=n div (-r);      if n mod (-r)<0 then q:=q+1;
                        print(q);
                      if n+q*r<=9 then write(n+q*r) else write(chr(n+q*r
                                                +55));
                      end;
end;
```

六、高中组第4题　方格取数

【问题描述】

设有N*N的方格图(N≤8),我们将其中的某些方格中填入正整数,而其他的方格中则放入数字0。如下图所示(见样例):

向右

A	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	13	0	0	6	0	0
3	0	0	0	0	7	0	0	0
4	0	0	0	14	0	0	0	0
5	0	21	0	0	0	4	0	0
6	0	0	15	0	0	0	0	0
7	0	14	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

向下

B

某人从图的左上角的A点出发,可以向下行走,也可以向右走,直到到达右下角的B点。在走过的路上,他可以取走方格中的数(取走后的方格中将变为数字0)。

此人从A点到B点共走两次,试找出2条这样的路径,使得取得的数之和为最大。

输　入

输入的第一行为一个整数N(表示N*N的方格图),接下来的每行有三个整数,前两个表示位置,第三个数为该位置上所放的数。一行单独的0表示输入结束。

输　出

只需输出一个整数,表示2条路径上取得的最大的和。

样　例

输入

```
8
```

2 3 13
2 6 6
3 5 7
4 4 14
5 2 21
5 6 4
6 3 15
7 2 14
0 0 0
输出
67

【问题分析】

本题是从1997年国际信息学奥林匹克的障碍物探测器一题简化而来,如果人只能从A点到B点走一次,则可以用动态规划算法求出从A点到B点的最优路径。具体的算法描述如下:从A点开始,向右和向下递推,依次求出从A点出发到达当前位置(i,j)所能取得的最大的数之和,存放在sum数组中,原始数据经过转换后用二维数组data存储,为方便处理,对数组sum和data加进零行与零列,并置它们的零行与零列元素为0。易知

$$sum[i,j]=\begin{cases}data[i,j] & \text{当 } i=0 \text{ 或 } j=0\\ \max(sum[i-1,j],sum[i,j-1])+data[i,j] & \text{当 } i>0,\text{且 } j>0\end{cases}$$

求出sum数组以后,通过倒推即可求得最优路径,具体算法如下:

```
置sum数组零行与零列元素为0
for i:=1 to n do
    for j:=1 to n do
        if sum[i-1,j]>sum[i,j-1]
          then sum[i,j]:=sum[i-1,j]+data[i,j]
          else sum[i,j]:=sum[i,j-1]+data[i,j];
i:=n;  j:=n;
while (i>1) or (j>1) do
      if (i>1) and (sum[i,j]=sum[i-1,j]+data[i,j])
          then begin data[i,j]:=-1; i:=i-1 end
          else begin data[i,j]:=-1; j:=j-1 end;
data[1,1]:=-1;
```

凡是被标上-1的格子即构成了从A点到B点的一条最优路径。

那么是否可以按最优路径连续走两次而取得最大数和呢?这是一种很自然的想法,并且对样例而言也是正确的,具体算法如下:

(1) 求出数组sum,
(2) s1:=sum[n,n],
(3) 求出最优路径,

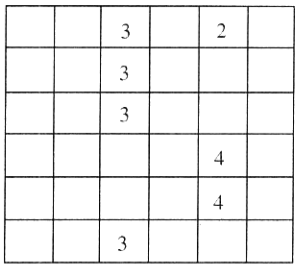
（4）将最优路径上的格子中的值置为 0，

（5）将数组 sum 的所有元素置为 0，

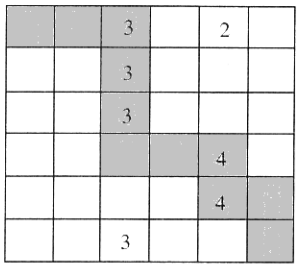
（6）第二次求出数组 sum，

（7）输出 s1 + sum[n, n]。

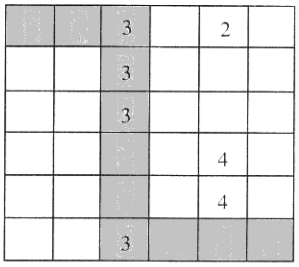
虽然这一算法保证了连续的两次走法都是最优的，但却不能保证总体最优，相应的反例也不难给出，请看下图：



图一



图二



图三

图二按最优路径走一次后，余下的两个数 2 和 3 就不可能同时取倒了，而按图三中的非最优路径走一次后却能取得全部的数，这是因为两次走法之间的协作是十分重要的，而图二中的走法并不能考虑全局，因此这一算法只能算是“贪心算法”。虽然在一些情况下，贪心算法也能够产生最优解，但总的来说“贪心算法”是一种有明显缺陷的算法。

既然简单的动态规划行不通，那么看看穷举行不行呢？因为问题的规模比较小，启发我们从穷举的角度出发去思考，首先让我们来看看 N = 8 时，从左上角 A 到达右下角 B 的走法共有多少种呢？显然从 A 点到 B 点共需走 14 步，其中向右走 7 步，向下走 7 步，共有 $C_{14}^{7} = 3432$ 种不同的路径，走两次的路径组合总数为 $C_{3432}^{2} = 3432 * 3431/2 = 5887596$，从时间上看是完全可以承受的，但是如果简单穷举而不加优化的话，对极限数据还是会超时的，优化的最基本的方法是以空间换时间，具体到本题就是预先将每一条路径以及路径上的数字之和（称之为路径的权 weight）求出并记录下来，然后用双重循环穷举任意两条不同路径之组合即可。考虑到记录所有的路径需要大量的存储空间，我们可以将所有的格子逐行进行编号，这样原来用二维坐标表示的格子就变成用一个 1 到 n^2 之间的自然数来表示，格子（i, j）对应的编号为（i - 1）* n + j。一条路径及其权使用以下的数据结构表示：

```
const maxn = 8;
type arraytype = array[1..2 * maxn - 2] of byte;
     recordtype = record path:arraytype;
                         weight:longint
                  end;
```

数组 path 依次记录一条路径上除左上和右下的全部格子的编号。

将所有的路径以及路径的权求出并记录下来的算法可用一个递归的过程实现，其中 i, j 表示当前位置的行与列，step 表示步数，sum 记录当前路径上到当前位置为止的数之和，当前路径记录在数组 position 中（不记录起始格），主程序通过调用 try(1, 1, 0, data[1, 1])求得所有路径。

```
procedure try(i, j, step, sum:longint);
begin
    if (i=n) and (j=n)
       then begin
                 total:=total+1;
                 a[total].path:=position;
                 a[total].weight:=sum;
            end
       else begin
                 if i+1<=n then
                 begin
                      position[step]:=i*n+j;
                      try(i+1, j, step+1, sum+data[i+1, j])
                 end;
                 if j+1<=n then
                 begin
                      position[step]:=(i-1)*n+j+1;
                      try(i, j+1, step+1, sum+data[i, j+1])
                 end
            end
end;
```

在历举了二条不同的路径后,只要将二条路径的权相加再减去二条路径中重叠格子中的数即为从这二条路径连走两次后取得的数之和,具体算法如下:

```
for i:=1 to n do   {将二维数组转化为一维数组}
    for j:=1 to n do d1[(i-1)*n+j]:=data[i, j];
max:=0;
for i:=1 to total-1 do
    for j:=i+1 to total do
    begin
        current:=a[i].weight+a[j].weight;
        for k:=1 to 2*n-3 do   {判断重叠格子,但不包括起点的终点}
        begin
            if a[i].path[k]=a[j].path[k] {第k步到达同一方格}
               then current:=current-d1[a[i].path[k]]
        end;
        if current>max then max:=current
    end;
writeln(max-data[1, 1]-data[n, n]);
```

应该看到穷举的效率是十分低下的,如果问题的规模再大一些,穷举法就会超时,考虑到

走一次可以使用动态规划，则只需穷举走第一次的路径，而走第二次可以用动态规划，这样可大大提高程序的效率，其算法复杂度为 $O(n^2C_{2n-2}^{n-1})$，实现时只需将前面二种算法结合起来即可，但这样做仍然不能使人满意，因为只要穷举了从A点到B点所有路径，算法的效率就不可能很高，要想对付尽可能大的n，还是要依靠动态规划算法。实际上本问题完全可以用动态规划解决，只是递推起来更为复杂些而已，前面在考虑只走一次的情况，只需考虑一个人到达某个格子(i, j)的情况，得出 sum[i, j] = max(sum[i-1, j], sum[i, j-1]) + data[i, j]，现在考虑两个人同时从A出发，则需考虑两个人到达任意两个格子(i1, j1)与(i2, j2)的情况，显然要到达这两个格子，其前一状态必为(i1-1, j1), (i2-1, j2)；(i1-1, j1), (i2, j2-1)；(i1, j1-1), (i2-1, j2)；(i1, j1-1), (i2, j2-1) 四种情况之一，类似地，可以推导出：设 p = max(sum[i1-1, j1, i2-1, j2], sum[i1-1, j1, i2, j2-1], sum[i1, j1-1, i2-1, j2], sum[i1, j1-1, i2, j2-1])，则

$$sum[i1, j1, i2, j2] = \begin{cases} 0 & \text{当 } i1=0 \text{ 或 } j1=0 \text{ 或 } i2=0 \text{ 或 } j2=0 \\ p + data[i1, j1] & \text{当 } i1, j1, i2, j2 \text{ 均不为零，且 } i1=i2, j1=j2 \\ p + data[i1, j1] + data[i2, j2] & \text{当 } i1, j1, i2, j2 \text{ 均不为零，且 } i1 \neq i2 \text{ 或 } j1 \neq j2 \end{cases}$$

具体算法如下：

```
置 sum 数组所有元素全为 0;
for i1:=1 to n do
    for j1:=1 to n do
        for i2:=1 to n do
            for j2:=1 to n do
            begin
                if sum[i1-1, j1, i2-1, j2]>sum[i1, j1, i2, j2]
                    then sum[i1, j1, i2, j2]:=sum[i1-1, j1, i2-1, j2];
                if sum[i1-1, j1, i2, j2-1]>sum[i1, j1, i2, j2]
                    then sum[i1, j1, i2, j2]:=sum[i1-1, j1, i2, j2-1];
                if sum[i1, j1-1, i2-1, j2]>sum[i1, j1, i2, j2]
                    then sum[i1, j1, i2, j2]:=sum[i1, j1-1, i2-1, j2];
                if sum[i1, j1-1, i2, j2-1]>sum[i1, j1, i2, j2]
                    then sum[i1, j1, i2, j2]:=sum[i1, j1-1, i2, j2-1];
                sum[i1, j1, i2, j2]:=sum[i1, j1, i2, j2]+data[i1, j1];
                if (i1<>i2) or (j1<>j2)
                    then sum[i1, j1, i2, j2]:=sum[i1, j1, i2, j2]+data[i2, j2]
            end;
writeln(sum[n, n, n, n])
```

【数据结构】 （略）

【算法分析】 （略）

【程序清单】

```
program g2000_4;
const maxn = 10;
type arraytype = array [0..maxn, 0..maxn] of longint;
var i, j, k, n, i1, i2, j1, j2:longint;
    data:arraytype;
    sum:array [0..maxn, 0..maxn, 0..maxn, 0..maxn] of longint;
function max(x, y:longint):longint;
begin
    if x > y then max:=x else max:=y;
end;
begin
    for i:=1 to maxn do
        for j:=1 to maxn do data[i, j]:=0;
    readln(n);
    repeat
        readln(i, j, k);
        data[i, j]:=k
    until (i=0) and (j=0) and (k=0);
    fillchar(sum, sizeof(sum), 0);
    for i1:=1 to n do
        for j1:=1 to n do
            for i2:=1 to n do
                for j2:=1 to n do
                begin
                    if sum[i1-1, j1, i2-1, j2] > sum[i1, j1, i2, j2]
                        then sum[i1, j1, i2, j2]:=sum[i1-1, j1, i2-1, j2];
                    if sum[i1-1, j1, i2, j2-1] > sum[i1, j1, i2, j2]
                        then sum[i1, j1, i2, j2]:=sum[i1-1, j1, i2, j2-1];
                    if sum[i1, j1-1, i2-1, j2] > sum[i1, j1, i2, j2]
                        then sum[i1, j1, i2, j2]:=sum[i1, j1-1, i2-1, j2];
                    if sum[i1, j1-1, i2, j2-1] > sum[i1, j1, i2, j2]
                        then sum[i1, j1, i2, j2]:=sum[i1, j1-1, i2, j2-1];
                    sum[i1, j1, i2, j2]:=sum[i1, j1, i2, j2]+data[i1, j1];
                    if (i1<>i2) or (j1<>j2)
                      then sum[i1, j1, i2, j2]:=sum[i1, j1, i2, j2]+data[i2, j2]
                end;
```

```
        writeln('Maxscore = ', sum[n, n, n, n])
end.
```

【运行示例】（下划线表示输入）

```
3
1 1 10
1 3 5
2 2 6
2 3 4
3 1 8
3 2 2
0 0 0
Maxscore = 30

7
1 3 2
1 4 3
2 3 3
3 3 3
5 5 4
6 5 4
7 3 2
7 5 4
0 0 0
Maxscore = 25

8
1 1 13
1 3 7
1 8 14
2 2 1
2 4 2
4 3 5
5 5 4
6 2 6
7 8 16
0 0 0
Maxscore = 60
```

8
1 1 1
1 8 1
2 2 2
2 7 2
3 4 3
3 5 3
4 4 3
4 5 3
7 2 2
7 7 2
8 1 1
8 8 1
0 0 0
Maxscore = 18

图书在版编目(CIP)数据

全国青少年信息学(计算机)奥林匹克分区联赛试题解析(中学)/章维铣主编. —南京:南京大学出版社,2001.5

(青少年学科奥林匹克竞赛丛书)

ISBN 7-305-03710-9

Ⅰ.全... Ⅱ.章... Ⅲ.计算机课-中学-竞赛题-解题 Ⅳ.G634.675

中国版本图书馆 CIP 数据核字(2001)第 027499 号

丛 书 名 青少年学科奥林匹克竞赛丛书
书　　名 全国青少年信息学(计算机)奥林匹克分区联赛试题解析(中学)
主　　编 章维铣
出版发行 南京大学出版社
社　　址 南京市汉口路 22 号　邮编 210093
电　　话 025-3596923　025-3592317　传真 025-3303347
网　　址 http://www.njupress.com
电子函件 nupress1@publicl.ptt.js.cn
经　　销 全国新华书店
印　　刷 扬中印刷厂
开　　本 787×1092　1/16　印　张 20　　字　数 471 千
版　　次 2001 年 6 月第 1 版　2001 年 6 月第 1 次印刷
印　　数 1—18000
定　　价 28.00 元
ISBN 7-305-03710-9/TP·219

青少年学科奥林匹克竞赛丛书

内容提要

本书紧密围绕全国青少年信息学(计算机)奥林匹克分区联赛竞赛大纲所涉及的知识点，以算法分析为主线，剖析了从首届联赛至今的六届联赛的有关试题，讲思想，讲方法，注重学生思维训练和能力提高。本书由基础篇和提高篇两部分组成，分别对应全国分区联赛的初赛与复赛试题。

本书聘请国际信息学奥林匹克中国国家队总教练吴文虎担任顾问，由多年来积极参与竞赛命题工作与活动的专家教授、计算机高级教师编写。读者对象为广大初、高中学生、家长、辅导教师及计算机爱好者。

责任编辑　孙　辉
装帧设计　杨小民
责任校对　刘子普

ISBN 7-305-03710-9
9 787305 037108 >

ISBN 7-305-03710-9/TP · 219
定价:28.00元