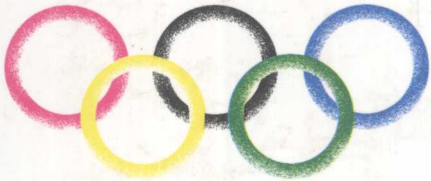
信息学奥林匹克竞赛题解精编

江 涛 王颖寒 骆静辰 编



南京大学出版社

信息学奥林匹克竞赛题解精编

江 涛 王颖寒 骆静辰 编

南京大学出版社

内容简介

本书汇集了近几年的全国信息学奥林匹克竞赛(NOI)、中国队组队选拔赛(CTSC)及国际信息学奥林匹克竞赛(IOI)的竞赛试题,并由多年从事计算机教育、辅导参赛学生的老师以及参加过此类比赛的优秀选手,根据多年的经验和亲身体会给出了较为详细的分析和参考程序。程序均用 TURBO PASCAL 7.0 实现。

本书可作为程序设计竞赛的爱好者,特别是广大中小学信息学竞赛选手和辅导老师的参考用书。

图书在版编目(CIP)数据

信息学奥林匹克竞赛题解精编 / 江涛, 王颖寒, 骆静辰主编. —南京: 南京大学出版社, 2000.6
(学科奥林匹克竞赛题解精编)
ISBN 7-305-03064-3

Ⅰ.信… Ⅱ.①江…②王…③骆… Ⅲ.信息学-解题 Ⅳ.G201

中国版本图书馆 CIP 数据核字(2000)第 30253 号

书　　名　信息学奥林匹克竞赛题解精编
编　　者　江涛　王颖寒　骆静辰
责任编辑　高锦明
装帧设计　龚　彬
责任校对　刘子普
出版发行　南京大学出版社
(南京汉口路 22 号南京大学校内　邮编 210093)
印　　刷　南京印刷制版厂
经　　销　全国各地新华书店
开　　本　850×1168　1/32　印张 22.25　字数 948 千
2000 年 6 月第 1 版　2000 年 6 月第 1 次印刷
印　　数　1-5000
定　　价　29.00 元
ISBN　7-305-03064-3/TP·194

序

中学学科(数、理、化)竞赛题典,1992 年问世后,颇受欢迎,很快销售一空.为了适应各方面的需要,我们编成这套最新的竞赛题典,既搜集了近五年来的赛题,又从原来的题典中精选出一部分内容.这样,主要的竞赛(如国际竞赛)均保持完整,篇幅又不过于庞大,可以称为最新最精的题典.

信息科学(计算机)竞赛,五年前刚刚起步,现在材料已经很多.这次也编为一册.随着计算机的广泛使用,信息科学竞赛将会越来越引起人们的关注.

对于学科竞赛,有各种各样的看法,总的说来,与体育竞赛类似,正面效应远远超过负面影响.所以近年来,各学科的竞赛,不仅没有停滞的趋势,反而蓬勃发展,日益扩大.

"天下大事,必作于细。"我们希望,这套新题典的出版,对于中学学科竞赛的发展,对于科学知识的普及,对于国民素质的提高,能够起着一点积极有益的作用.

单　墫

《学科奥林匹克竞赛题解精编》

编　委　会

前　言

学科竞赛是中学生最喜欢和参加最为广泛的课外活动形式之一。随着计算机科学技术的飞速发展和个人电脑的日益普及，信息学奥林匹克竞赛已逐渐成为中学时代的学科竞赛之一，并以极快的速度在发展、壮大。

迄今为止，我国已连续举办了十五届全国信息学奥林匹克竞赛（简称NOI），国际信息学奥林匹克竞赛（简称IOI）也成功地举办了十次。在所有学科的国际竞赛中，只有信息学竞赛中国是首届派团参赛的。而且我国计算机竞赛选手在著名的国家队总教练吴文虎教授带领下，屡次在国际赛上取得优异的成绩，这一切充分说明了我国对科教兴国、开发计算机人才的重视。中国的中学生在新兴的科学技术领域同样是优秀的，“计算机的普及要从娃娃做起”的教育战略思想的成果也是巨大的。

在信息学竞赛浪潮的推动下，各个省市的竞赛也如火如荼地开展起来。这几年，在竞赛中一些很优秀、新颖、有意义的试题更是层出不穷。我们编者将这些赛题进行了归纳性总结，合编成此书，为的是给大家提供一个较为全面的题库资料，更希望大家各抒己见，借此引发大家的创造精神。这也正是我们编此书的初衷。

所谓信息学（计算机）竞赛和其它科目的竞赛不同。它的宗旨是推动计算机在青少年中的普及，全面提高青少年的综合素质。因此信息学属于课外因材施教活动之列，重点在于创造能力的培养。而其竞赛的特点正是在掌握计算机程序设计的基本技能基础上，着重考察学生的分析问题能力和创造能力。活学活用的教学方法始终贯穿于各届竞赛中，而经典算法的生搬硬套不是它的考试目的。

所以，本书对近几年来的NOI、IOI以及中国队组队选拔赛（CTSC）试题进行了细致全面的分析并给出了程序解答。其中体现的思路和算

法，是编者在吸取了很多优秀选手的经验、心得之后，结合自己的实践而写成的，在一定的理论基础上更突出实际经验。但也正是由于这一点，往往会产生一些不成熟却很有新意的想法，虽然有时不能严格地证明其正确性，但是如果这些新的思想、新的方法能给读者的思路有新的启发、带来新的灵感，那我们编者所期望的“抛砖引玉”的目的也就达到了。

本书在编写过程中，得到了张御冬、吴朝晖、巴健全等老师的大力支持和帮助，特此致谢！

由于作者水平有限，时间仓促，书中难免有缺漏错误之处，敬请原谅。

编　者

2000 年 1 月

目　录

92' NOI－1 文章排版 把一段文章按要求排版。

文章的输入方式为：由键盘输入一个以回车符结束的文章（最大长度2000个字符）。

排版时以单词为基本单位。单词由不含空格的任意字符组成，是长度小于20个字符的串。空格符是分隔单词的唯一字符，在输入时连续的空格符在处理时应首先化简为单个空格符。

在排版前应先输入排版后每行的字符数N，排版后将整理好的文章按行输出。输出时应保证不将一个完整的单词截断，并且要求输出的总行数最小。

将每个不足N个字符的行用空格符补足，填充空格符的方式有以下3种：

1.将填充的空格符置于每行的末尾，并要求每行的起始为单词。

2.将填充的空格置于每行的起始，并要求每行的末尾为单词。

3.将填充的空格符尽可能平均分配在每行中，并保证每行的起始和末尾均为单词。

试编程对输入的一段文章分别完成上述三个要求。

【算法分析】 本题是一个对字符串进行简单处理的题目，对选手的细致解题是一个考验，需要注意的是题中的一些具体要求。注意点如下：

A. 总长不大于2000，单词长度小于20。因此，在读入文章时要加以检查。

B. 单词的分隔符只有空格，但可以有连续多个。

读入时要加以处理，使单词之间只要一个空格即可。当然我们也可以把单词收集起来，放入字符串数组中，以后就对单词整个地处理。

C. 每行长度为N，但要保持单词在一行的完整性。

因此，我们要检查是否有单词的长度超过N，如有则肯定不能排版。若有一个单词的长度为n－1则任务3也一定不能排版。

D. 要求输出总行数最小。

因此一般地想法是:我们在前面要尽可能地每行多放单词。

E.三种补充空格方式构成了三种任务。

但是对于任务 3,它有一个特别的要求是:要保证每行的起始和末尾均为单词。这样有些情况就没有解,如:N = 10,文章为:"abcded"。而更让人忽视的是,在有些情况按上面 D.的处理方法不能适用任务 3 了,如:N = 10,文章为:"ab abc abc abcde"。这时第一行不能尽量放为"ab abc abc",而应该为"ab abc"。

由于任务 3 的特殊性,我们下面着重分析一下它的算法。

单词的内容在排版时并不重要,关键是长度。为了简化问题的操作,我们可专门用一数组 Len[0..2000] of integer 记录每个单词的长度。

一行以某个单词开头时,单词个数不能固定,最少是两个,最多到不能放为止。如果我们若用回溯搜索的方法,速度会太慢。但本题的"输出行数最少"是符合"最佳原理"的,即:设从第 1 到第 m 个单词的最优解为 Min_Line(1,m),第 k 个单词是其中某行的结尾,则它前面的单词安排也一定是最优的(= Min_Line(1,k))。

因此,我们可用动态规划来解决这个问题。

动态规划方程为:

$$Min\_Line(1,k) = Min\{Min\_line(1,i-1)+1\} \quad 0<i<=k$$

其中从 i 到 k 的单词可以组成满足任务 3 要求的一行。

若找不到这样的 i,则不能以 k 为结尾。

边界条件为:Min_line(1,0) = 0。

判断"满足从 i 到 k 的单词是否可以组成满足任务 3 要求的一行"的方法:

(1)对于一个单词:

if (i = k) and (Len[k] = n) then ok: = true

else ok: = false

如果仅有一个单词,但长度不等于 n,不能满足要求。

(2)对于多个单词:

temp: = k - i (temp 为单词数 - 1,也是单词间最少空格数。)

for j: = i to k do temp: = temp + Len[j] 求最小累计长

if temp > n then ok: = false else ok: = true

若累计长度不大于 n 可通过中间加空格方式满足任务 3。

【数据结构】

```
type
  rec = record                    {存放以某单词结尾时的最佳情况}
        mark:boolean;             {是否存在以本单词结尾的方案标志}
        last,                     {记录上一行的最后一个单词}
        minrow,                   {记录最优方案的行数 }
        length:integer;           {记录最优方案时本行的单词总长度}
```

```
    end;
var
  data:array[0..2000] of char;   { 读入的原始文章字符 }
  word:array[0..2000] of record  {存放每个单词在 data 中的位置和长度}
                                 start,long:integer;
                               end;
  row:array[0..2000] of rec;  { 每个单词的最优方案 }
  len,n,num:integer;
```

【程序描述】

```
uses crt; {读文章时用 readkey,若用 read 不能超过键盘缓冲区 128 个字符}
procedure error; { 错误处理 }
  begin
    writeln('Data error!'); halt;
  end;

procedure readtext; {读入原始数据}
  var
    ch:char;
    i:integer;
  begin
    writeln('Input Text: ');
    ch:=readkey;
    len:=0;
    while ch<>#13 do
      begin
        write(ch); inc(len);
        if len>2000 then error; {文章长度检查}
        data[len]:=ch;
        ch:=readkey;
      end;
    writeln;
    write('n=');readln(n);
  end;

procedure getword; { 从 data 中收集单词,并得到单词长度 }
  var
```

```
    i:integer;
  begin
    i:=1;
    while i<=len do
      begin
        while (i<=len) and (data[i]=' ') do inc(i); {删除空格}
        if i>=len then break;
        inc(num);word[num].start:=i; { 新单词的开始处 }
        while (i<=len) and (data[i]<>' ') do inc(i);
        word[num].long:=i-word[num].start; { 新单词的长度 }
      end;
  end;
procedure work1; {任务 1 }
...{略}
procedure work2; {任务 2 }
...{略}

procedure work3; {任务 3 }
  var long,i,j,k,temp,min,p:integer;
  begin
    for i:=1 to num do { 对明显不可能情况处理 }
      if (word[i].long=n-1) or (word[i].long>n) then error;
    row[0].minrow:=0;row[0].mark:=true; { 初始边界}
    for i:=1 to num do
      begin
        min:=num;
        row[i].mark:=false; {是否有以第 i 单词结尾方案的标志}
        for j:=i downto 1 do if row[j].mark then
        begin { 按状态转移方程求最佳方案 }
          temp:=i-j;
          for k:=j to i do inc(temp,word[k].long);
          if temp>n then break;
          if (j<i-1)
            or and (j=i)) then {找到可行解}
              if min>row[j+1].minrow+1 then {找到更优的解}
                begin min:=row[j].minrow+1;p:=j;
```

```
                    long:=temp-i+j;
                    row[i].mark:=true; {存在方案标记=true}
                end;
            end;
        if row[i].mark then {记录最优方案}
            begin
                row[i].minrow:=min; {以第i个单词结尾的最少行数}
                row[i].last:=p; {上一行结尾单词是哪个}
                row[i].length:=long; {本行的单词总长}
            end;
        end;
    if not row[num].mark then error; {如果不能以最后一个单词结尾则无解}
end;

procedure print_word(i:integer);
    var j:integer;
    begin
        with word[i] do
            for j:=start to start+long-1 do write(data[j]);
    end;

procedure out(i:integer); {输出排版结果}
    var
        j,k,temp:integer;
    begin
        if row[i].last<>0 {从最后一个,递归地反方向打印方案}
        then out(row[i].last);
        k:=i-row[i].last-1; {这一行的单词数-1}
        temp:=n-row[i].length; {这一行所需的空格数}
        print_word(row[i].last+1); {打印第一个单词}
        for j:=row[i].last+2 to i do {打印其它单词时要加空格}
            begin
                write(' ':temp div k); {所要加的空格数}
                print_word(j);
                temp:=temp-temp div k;dec(k); {为计算下一次空格数准备}
            end;
```

```
    writeln;
  end;
```

【精益求精】 仔细分析任务3,它有一个特殊的要求:保证每行的起始和末尾均为单词。也就是说,每行至少有两个单词。我们可采用贪心算法来完成:

预处理:尽可能地在每一行多放单词,并记录每行的单词数。

逐步调整:从最后一行开始,如当前行只有一个单词,则判断,上一行最后一个单词是否可移至这一行(即总长是否不超过n),是则继续处理上一行;否则退出,输出不能按任务3排版。

以上算法实现,请读者自行完成。

92'NOI-2 逻辑表达式 由英文字符和符号、*、+、()组成逻辑表达式,英文字母表示变量,变量有两种可能的取值,False或True;、*、+。括号()可改变表达式的运算次序,且可以嵌套。

逻辑"非"运算的公式如下表:

A	~A
True	False
False	True

逻辑"与"和逻辑"或"的运算公式如下:

A	B	A*B	A+B
False	False	False	False
False	True	False	True
True	False	False	True
True	True	True	True

两个逻辑表达式等价,当且仅当两个公式中相同名字的变量取任何一组值时两个公式的值都相同。如:

A*(B+C) 与 A*B+A*C 等价
A*(~A+B) 与 A*B 等价
(~A+A)*B+C 与 B+C 等价
A*B+A*~B 与 A 等价
而:
A+B 与 A*B 不等价