国际国内奥林匹克信息学(计算机)
1996 年竞赛试题解析

吴文虎　王建德　编著

北京大学出版社
北　京

内 容 简 介

本书收集了 1996 年国际、国内信息学奥林匹克竞赛试题及题解。全书分四章:第八届国际奥林匹克信息学竞赛中国组队赛试题分析;第十三届全国奥林匹克信息学竞赛试题分析;第八届国际奥林匹克信息学竞赛试题分析;题库。这些试题及题解涉及计算机程序设计的典型算法和基本的数据结构知识,介绍了程序设计的思路和技巧,且具有代表性和趣味性;不仅适合于广大青少年计算机爱好者,而且对大学生和计算机专业人员也有参考价值。

图书在版编目(CIP)数据

国际国内奥林匹克信息学(计算机)竞赛试题解析:1996 年/ 吴文虎,王建德编著 .—北京:北京大学出版社,1997 .2
ISBN 7-301-03376-1
Ⅰ .国… Ⅱ .①吴… ②王… Ⅲ .电子计算机-试题-解题 Ⅳ .TP3-44

书 名:国际国内奥林匹克信息学(计算机)1996 年竞赛试题解析
著作责任者:吴文虎 王建德
责 任 编 辑:沈承凤
标 准 书 号:ISBN 7-301-03376-1/ TP337
出 版 者:北京大学出版社
地 址:北京市海淀区中关村北京大学校内 100871
电 话:出版部 62752015 发行部 62559712 编辑部 62752032
排 版 者:盛达电脑照排中心
印 制 者:
发 行 者:北京大学出版社
经 销 者:新华书店
1787×1092 毫米 16 开本 9 .625 印张 240 千字
1997 年 6 月第一版 1997 年 6 月第一次印刷
定 价:16 元

序

人类正进入信息时代。计算机及程序设计是现代信息技术的核心内容,对社会进步正在产生难以估量的深远影响。学习计算机知识和掌握计算机应用技能是时代需要,也是人材必备的科学素养。“计算机的普及要从娃娃做起”是具有战略意义的远见卓识,是落实“科教兴国”战略的一项重要工作。信息学奥林匹克竞赛是在青少年中普及计算机知识的重要形式。我国从首届开始就积极组队参加“国际信息学奥林匹克”(International Olympiad in Informatics,简称 IOI)竞赛。本书作者,清华大学吴文虎教授正是中国队的领队和总教练,他率中国队在历届参赛中均取得优异成绩(荣获 17 块金牌,6 块银牌,8 块铜牌),在计算机学会的领导和支持下他还积极组织国内的相应的竞赛活动。

本书收集了 1996 年国际、国内信息学奥林匹克竞赛试题及题解。这些试题及题解涉及计算机程序设计的典型算法和基本的数据结构知识,介绍了程序设计的思路和技巧,且具有代表性和趣味性。

当前,我国不少中小学已将计算机课列为必修课,计算机开始进入家庭。本书出版将有助于信息学在提高指导下的普及和在普及基础上的提高,不仅适合于广大青少年计算机爱好者,而且对大学生和计算机专业人员也有参考价值。

中国科学院院士
中国计算机学会副理事长
北京大学计算机科学技术系主任

前　言

一个国家、一个民族要想不落伍,要想跻身于世界先进民族之林,关键在于要拥有高素质的人;综合国力的竞争,说到底是人才的竞争。电子计算机是现代科学与 技术的基础与核心,它的飞速发展,把社会生产力水平提到前所未有的高度。电脑对人类社会的发展所起的巨大作用,特别是对于人类智能的发展所起的促进作用,已为人们普遍认识到。计算跟语言一样,是人类社会每时每刻都不可缺少的。电脑作为“ 人类通用智力工具 ”,不仅可以帮助我们进行复杂计算,还可以在人的指挥下创造更多的思维成果,起到人脑延伸的作用。在有了计算机的信息时代,计算已经成为与理论研究、实验研究并重的第三种科学研究方法,电脑在认识世界和改造世界,以及在开发人类智能方面所起的无与伦比的作用不容忽视。正是因为如此,计算机与基础教育相结合已经成为世界的大趋势。“ 计算机的普及要从娃娃做起 ”已经成为“ 科教兴国 ”的一项重要内容。

国际信息学奥林匹克是联合国教科文组织倡导的五项国际青少年学科竞赛的内容之一。宗旨是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,推动交流,促进理解;宣传信息学这一新兴学科,给学校这一类课程增加动力,启发新的思路;建立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。这种竞赛之所以冠之以奥林匹克这个词,也是希望“ 更快、更高、更强 ”。这是一种智力与应用计算机能力的大赛。从益智的角度看,是用电脑帮助开发人脑,重在提高思维能力与创新能力。在中国队的训练中强调德智体美全面发展,强调打好数理化、文史地、音乐、美术、体育等课的基础,这是学习和掌握电脑的基本条件,绝不提倡单科独进;强调在心态上要自立、自尊、自信、自强,要怀着中华民族的自豪感和自信心去参赛,这种心态是学习、训练和取胜的重要条件。从 1989 年到 1996 年我国已八次参赛,夺得金牌 17 块,银牌 6 块,铜牌 8 块,每届都取得名列前茅的好成绩。在 IOI'95(荷兰)突破了前 6 届比赛女孩与金牌无缘的记录,中国队的两名女选手荣登了金牌领奖台;IOI'96(匈牙利)实现了“ 全金 ”的突破,四名中国队队员每人拿到一块金牌。

从大局看,竞赛不是目的。如果把拿金牌当作目的,眼光未免狭小了些,我们把竞赛当作推动普及的手段,目的只有一个“ 科教兴国 ”,希望由一批拔尖的学生带动起一大批或者一代青少年学科学爱科学,在他们中间脱颖一批最优秀的人才,担起中国腾飞的大任。竞赛是青少年喜闻乐见的形式,属于课外活动,带有因材施教、因材施测的特点。普及是有层次的,学科竞赛对青少年而言属于比较高的层次,不能要求人人参加,因为它有相当大的难度。信息学奥林匹克的试题,涉及程序设计语言、常用算法、组合数学、图论、人工智能搜索等知识,学起来是很吃力的,况且我们提倡自学,许多中国队的选手也是通过自学掌握这些内容的。信息学试题的另一特点是没有一成不变的解法,鼓励选手创新。这对于创造意识和创造能力的培养,无疑是大有好处的。我们曾经问过一些曾是中国队的正式队员或预备队员,而现在就读于清华大学的选手:参加过这种活动和没有参加过,你自己感觉有什么不同?他们回答说:譬如一把刀,磨过和没磨过就是不一样。奥林匹克学科活动使我们打开了眼界,增长了知识,提高了分析问题和

解决问题的能力,特别是有了比较强的创造欲望。选手们的这种感受也使我们当老师的受到鼓舞。因此,我们就将赛题和解法写出来供大家学习。这些题目比较新颖,很难去套用固定算法或模式。这中间有些招数是选手们想出来的。从中也可以看出信息学奥林匹克要求创新,鼓励创新的意图。对青少年读者而言,书中给出的解法,我们希望仅仅起到一点抛砖引玉的作用,并且热切盼望能够引出更多的美玉来。作为老师,我和王建德都这样想:“精心育桃李,热望青胜蓝”。这是我们写这本书的初衷。

吴文虎

1997 年 1 月 12 日

目　　录

第一章　第八届国际奥林匹克信息学竞赛中国组队赛试题分析

1.1　建立绿化保护区

一、试题

请认真阅读下面的试题,并解答后面的问题。

某一块区域上种有 N 棵树,每一棵树本身的直径都很小,相对于整个区域来说可忽略不计。我们在这块区域上建立直角坐标系,则每一棵树的位置由其所对应的坐标表示。

为了保护这些树,需要建立一个简单的保护区,用一根绳子以其中某些树为支点,围成一个封闭的多边形区域,使每一棵树均在这个多边形区域中。

【问题】　读入每一棵树的坐标,寻找一种建立保护区的方案,使绳子的长度最小。

【输入要求】

从文本文件中读入每棵树的坐标(文件名由键盘输入)。

文本文件的格式为:

第 1 行为树的总棵数 N(N≤10);

第 2 行至第 N + 1 行为 N 棵树的坐标(横纵坐标均为整数,范围为: - 1000—1000)。

【输出要求】

按多边形的顶点次序输出每个顶点的坐标(输出时,任一顶点均可作为起始点,次序为顺时针或逆时针均可)和绳子的长度(保留小数点后两位数字)。

例如,某一文本文件的内容为:

```
 8
 0          0
-1          0
-1         -2
 1         -2
 0          2
 1          1
 1         -1
-2          1
```

则一种正确的输出为:

```
 0          2
 1          1
 1         -2
-1         -2
-2          1
LENGTH = 11.81
```

图 1.1-1 示了这个例子。(注意:

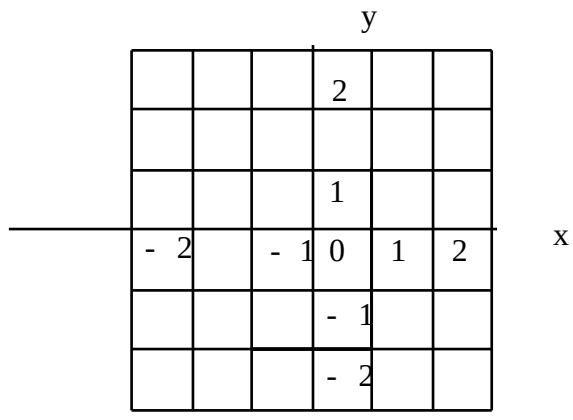


图 1.1-1 表

该例所输出的多边形仅含五个顶点,输出少了或多了均算错。)

【问题 1】 请列举你认为可能的一些输入错误。

【问题 2】 请你为上题编制 4 组测试数据(不包括输入错误的情况),给出输入数据及输出结果,要求你的测试数据能尽量起到全面测试的目的。

【问题 3】 下面给出某选手的程序,请对该程序做少量修改使之成为正确的程序(不用考虑输入判错功能)。

```
PROGRAM wrap_problem(input,output);

CONST
   maxn = 11;

TYPE
   point = RECORD
              x,y:longint;
           END;

VAR
   p:ARRAY[1..maxn] OF point;
   i,n,r:integer;
   datafile:text;
   filename:string;
   length_sum:real;

FUNCTION len(p1,p2:point):integer;
      BEGIN
         len := sqr(p1.x-p2.x) + sqr(p1.y-p2.y);
   END;

FUNCTION thete(p1,p2:point):real;
CONST
   maxslope = 1000;
VAR
   dx,dy:real;
   r:real;
   BEGIN
```

```
dx：= p2 .x-p1 .x;
dy：= p2 .y-p1 .y;
IF  x< >0
  THEN r：= 2 + dy/ (dx * maxslope)
  EL E
    IF  y>0
      THEN r：= 1
      ELSE r：= 3;
   IF  x>0 THEN
     IF  y> =0
       THEN r：= r-2
       ELSE r：= r+2;
       thete：= r;
END;

PROCEDURE wrap(var k:integer);
VA
   i,j:integer;
   t:point;
   th,thl,thmin:real;
BE  IN
  j：= 1;
t：= p[1];
FO  i：=2 TO n DO
   IF p[i] .y<t .y THEN
     BE  IN
       t：= p[i];
       j：= i;
     END; {then}
k：= 0;
p[n+1] ：= p[j];
thmin：= 0;
RE  EAT
   k：= k+1;
   t：= p[k];
   p[k] ：= p[j];
   p[j] ：= t;
   th：= thmin;
   thmin：= 0;
   FO  i：= k+1 TO n+1 DO
     BE  IN
       thl：= thete(p[k],p[i]);
       IF  thl< thmin) OR ((thl= thmin) AND (len(p[k],p[i])>len(p[k],p[j])))
```

```
                THEN  E  IN
                        j：= i;
                        thmin：= thl;
                      END
            END;
        UNTIL j = n + 1;
END;

BE  IN
    writeln;
    write( 'filename :  ');
    readln(filename);
    assign(datafile,filename);
    reset(datafile);
    readln(datafile,n);
    FO  i：= 1 TO n DO
        readln(datafile,p[i] .x, p[i] .y);
    close(datafile);
    wrap(r);
    writeln( 'the answer is:  ');
    length_sum：= 0;
    FO  i：= 1 TO r DO
        BE  IN
            writeln(p[i] .x:6,p[i] .y:6);
            IF  i< kl)
                TH  N
                    length_sum：= length_sum + sqrt(len(p[i],p[i+1]))
                EL  E
                    length_sum：= length_sum + sqrt(len(p[i],p[1]));
        END;
        writeln( 'length =  ',length_sum:8:2);
END .
```

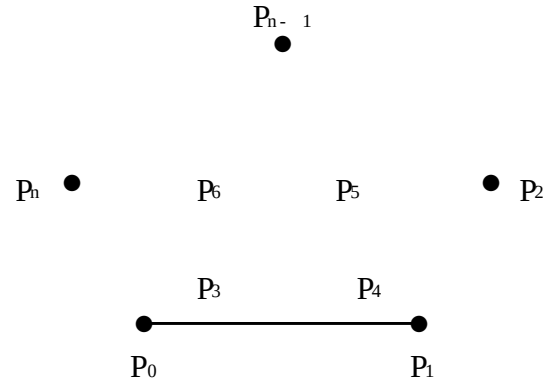
二、算法分析

试题所寻求的一种建立保护区的方案实际上是求一个点集 q = {$P_0 \cdots P_{n-1}$}的凸包 ch(q),它是一个包含点集 q 的最小凸多边形,即 q 中的每一个点或者在 P 的边界上或者在 P 的内部。在直观上,我们可以把 q 中的每一个点看做露在板外的铁钉,那末凸包 ch(q)就是包含所有铁钉的一个拉紧的橡皮绳所构成的形状。如图 1 .1-2 所示。

问题 1,2 要求读者为程序的黑箱测试选定有代表性的测试用例。其中问题 1 要求为输入设计无效等价类;问题 2 要求为凸包的计算设计 4 组输入数据,使其尽可能多地覆盖有效等价类。

我们先来列举一些可能的输入错误:

图 1.1-2



(1) N 值过大(N > 10)或过小(N ≤ 0)或者为实数;

(2) 树的坐标(x,y)超出范围: x(或 y) < - 1000 或者 x(或 y) > 1000;

(3) 出现非法字符,例如输入的 N, X, Y 为非数字符;

(4) 输入格式不正确,例如第二行起的坐标个数与 N 值不等;又如同一行出现多个坐标;

(5) 两棵树的坐标重叠。

下面,我们设计四组测试数据,每组测试数据代表一类问题:

(1) 含一个点的凸包

输入数据	输出结果
1	1000 1000
1000 1000	Length = 0.00

(2) 含二个点的凸包

输入数据	输出结果
2	0 - 1000
0 - 1000	0 1000
0 1000	Length = 4000.00

(3) 凸包内部含点

输入数据	输出结果
5	
0 2	0 - 2
2 0	2 0
- 2 0	0 2
0 - 2	- 2 0
0 0	Length = 11.31

(4) 凸包的边经过若干点且有平行边

输入数据	输出结果
6	
0 3	0 - 2
2 3	1 - 2
3 3	3 3
1 - 2	0 3
0 - 2	- 3 0

- 3　　0　　Length = 17 .23

最后我们对问题 3 中给出的选手程序进行白箱测试。测试前先分析整个程序,根据程序的流程明确每个过程函数的功能。然后对每一个子模块设计测试用例,这些测试用例尽可能多地覆盖模块内的程序路径。同时对每一个测试用例确定期望的输出,通过对照测试后的输出与期望的输出,找出错误的位置和性质,改正错误。

1 . 分析 len 函数中的错误

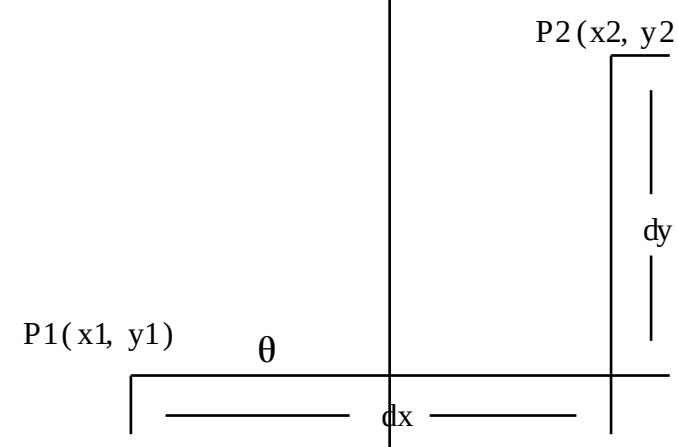
Len 函数的功能是计算 p1 和 p2 两点间距离的平方。模块内的计算公式完全正确,但函数返回值的数据类型应改为实型。

2 . 分析 thete 函数中的错误

thete 函数的功能是计算的输出 p1 与 p2 连线的优先级。凸包的计算应从最低位置(y 坐标值最小)的一个点 p0 (若这样的顶点有多个,则选取 x 坐标值最小的点为 p0)出发,按逆时针进行。每寻找一个新凸包点,当前凸包点 pk 与所有非凸包点的连线都要被扫描一次,从中选取优先级最小的连线〈pk , pj〉(即相对于 pk 来说, pj 的极角最小)作为凸包边。thete(p1 , p2)函数就是确定当前凸包点 p1 与非凸包点 p2 的连线的优先级 r。那么这个优先级是如何确定的呢 ? 由 thete(p1 , p2)函数的程序流程可以看出:

dx 为 p2 点与 p1 点间 x 坐标的增量; dy 为 p2 点与 p1 点间 y 坐标的增量,(如图 1 .1-3 所示)。

图 1 .1-3



dy/dx 为 tgθ。为避免 θ = 2nπ + π/2 或 θ = 2nπ + 3π/ 4 时 tgθ 趋于∞,规定(见图 1 .1-4)。

dx > 0,　dy≥0 时,　　0≤r = dy/ (dx * maxslope) < 1

dx > 0　dy < 0 时,　　3 < r = 4 + dy/ (dx * maxslope)≤4

dx < 0　dy≥0 时,　　1 < r = 2 + dy/ (dx * maxslope)≤2

dx < 0　dy < 0 时,　　2 < r = 2 + dy/ (dx * maxslope) < 3

问题是 maxslope 究竟设多大方能使优先级 r 的值限定在 1 - 4 的范围内呢 ? 如果我们设

p1 = (1, - 1000)　　p2 = (2, 1000)

dx = 2 - 1 = 1　　dy = 1000 - (- 1000) = 2000

若 maxslope = 1000, 则

r = dy/ (dx * maxslope) = 2000/ 1000 = 2 > 1

超出范围。只有当 maxsloped 在 - 1000≤x, y≤1000 的条件下,真正达到任两点连线的最大值时,r 才能限定在期望的范围内。显然当两点分别为(- 1000,1000)和(1000, - 1000)时,它们之间的连线最长($2000^2 + 2000^2 = 8 * 1000 \approx 3000$)。我们将常量 maxslope 修正为 3000。

r= 1

P2

P2

图 1 1-4

3 . 分析 wrap 过程中的错误

过程 wrap(VAR k:integer)的功能是计算凸包顶点并返回凸包上的点数 k。过程一开始是计算最低点 p[j],并放入 p[n + 1]。这个最低点便成为凸包上的第一个顶点。但是,当我们测试第四组数据

```
6
0     3
2     3
3     3
1   - 2
0   - 2
- 3    0
```

时,程序陷入了死循环。问题是最低点有(1, - 2)和(0, - 2)。程序按输入顺序选取(1, - 2)作为第一个凸包顶点,而凸包的算法要求在多个最低点中选取最左边的点(0, - 2)作为出发点。显然 wrap 过程中对出发点的判断条件是不全面的,应将第一个 for 循环中的

'IF (P[i] .y < t .y) THEN …'更正为

'IF (P[i] .y < t .y) OR ((P[i] .y = t .y) AND (P[i] .x < t .x)) THEN …'。

接下来我们分析 REPEAT…UNTIL 循环。该循环一开始累计凸包的顶点数 k; pj 设为凸包的第 k 个顶点,将凸包当前边的优先级 thmin 转赋给 th;然后初始化 thmin,并顺序搜索非凸包点 pk + 1,…, pn + 1 ,在 pk 与它们的连线中求一条具有最小优先权值的连线作为新的凸包边。这个最小优先权的值将存储在 thmin 中。但是测试的结果是:无论输入什么数据,都不能达到输出的期望值。原因有如下两点:

① thmin 用于记载目前为止得到的最小优先权值。如果 thmin 一开始设为 0,则无论如何也不会得到一条与 pk 相连的新凸包边的,因为该边的权值再小也不会小于 0。因此搜索前必须将 thmin 初始化为一个最大值。我们将“ thmin：= 0”改为“ thmin：= 4”。

② 由于凸包的构造是从最低点 p0 出发逆时针顺序进行的,因此当 0≤r≤2 时,是构造凸包的右链(即 p0 至最高点 ph 逆时针方向排列的顶点序列)。在构造过程中,若与凸包点 pk 相连的诸条连线中, < pk ,pj > 的优先级值最小,则说明相对 pk 来说 pj 的极角最小,因此 pj 应成为凸包的候选点。REPEAT…UNTIL 中嵌套的 for 循环就是做这件事。FOR 循环搜索每一个非凸包点 pk + 1…pn 。每搜索一个非凸包点 pj ,计算 pj 与 pk 连线的优先级 thl。若 thl < thmin 或者 thl = thmin,但连线距离 pk 近,则确定 < pk ,pj > 作为凸包的下一条候选边,但是这里忽视了一个重要因素:如果 < pk ,pj > 是构造左链时的候选边的话(th > 3),则该连线的优先级值一定要大于当前凸包边(thl > th)。否则最终将不能形成封闭的凸包。因此 for 循环体内对凸包候选点的判断条件应修改成:

```
'IF  (thl < thmin) OR (thl = thmin)AND (len(p[k],p[i])) > len(p[k],p[j]))
   A  N ((th < 2) OR (thl > th)) THEN BEGIN
    j：= i; thmin：= thl;
   END;
```

…'

4.分析主程序中的错误

由于修正了过程和函数中的错误,因此主程序调用这一些函数和过程可确切计算出凸包的顶点。如果程序有错的话,则可能是输出方面的问题。果然不出所料,编译'IF (i < k1) THEN…'一行时出现语法错误——kl 未在变量说明中定义。而 if 语句的位置正处在完成输出打印的 for 循环中,该循环按顺时针方向依次输出凸包顶点 P1,…,Pr,并累计凸包上各条边的总绳长,即:

< p1 ,p2 > 的绳长 + < p2 ,p3 > 的绳长 + … + < pr ,p1 > 的绳长

如果输出的凸包顶点 pi 非最后一个顶点 pi (i < r)时,则应将 < pi ,pi + 1 > 的绳长累计进总绳长;如果输出的是凸包的最后一个顶点 pr ,则应将 < pr ,p1 > 的绳长累计进总绳长,因为凸包是封闭的。显然应将'IF (i < kl) THEN …'改为'IF (i < r) THEN…'。

至此,选手程序中的错误全部修正完毕,修改后的程序可以计算出任意点集的凸包。下面给出正确程序的样本及其注解。

```
PROGRAM wrap_problem(input,output);

CONST
   maxn = 11;   {树的最多棵数}

TYPE
   point = RECORD
             x,y: longint;     {坐标类型}
           END;
VAR
   p:ARRAY[1..maxn] OF point;     {点集}
   i,n,r:integer;    {辅助变量,实际点数,凸包顶点数}
   datafile:text;    {文件变量}
   filename:string;    {文件名串}
   length_sum:real;    {总绳长}

FUNCTION len(p1,p2:point):real;     {求点 p1 和点 p2 间距离的平方}
BEGIN
   len:= sqr(p1.x - p2.x) + sqr(p1.y - p2.y);
END;    {len}

FUNCTION thete(p1,p2:point):real;    {计算和返回 < p1 ,p2 > 的优先级}
CONST
   maxslope = 3000;    {任两棵树间绳长的最大值}
VAR
   dx,dy:real;    {p1 与 p2 间 x 坐标和 y 坐标的增量,连线优先级}
   r:real;
BEGIN
   dx:= p2.x - p1.x;
   dy:= p2.y - p1.y;
```

```
  IF  x< >0
    TH  N
      r：=2+dy/(dx*maxslope)
    EL  E
      IF  y>0
        THEN r：=1
        ELSE r：=3;
  IF  x>0
    THEN
    IF  y>=0
      THEN r：=r-2
      ELSE r：=r+2;
  thete：=r;
END;{thete}

PROCEDURE wrap(VAR k:integer);  {计算凸包顶点并返回凸包点数 k}
VA
   i,j:integer;  {辅助变量}
   t:point;  {辅助点}
   th  thl,thmin:real;
     {上一条凸包边的优先级,当前连线的优先级,当前最小的优先级值}
BE  IN
{找出最低点 p[j](若这样的顶点有多个,则选取 x 坐标最小的一个顶点),并放入 p[n+1]}
j：=1;
t：=p[1];
FO  i：=2 TO n DO
   IF  p[i].y<t.y) OR ((p[i].y=t.y) AND (p[i].x<t.x)) THEN
     BE  IN
       t：=p[i];
       j：=i;
     END;  {then}
k：=0;  {从最低点出发构造凸包}
p[n+1]：=p[j];
thmin：=0;  {当前最小的优先级值初始化}
RE  EAT
   k：=k+1;
   t：=p[k];  {pj 与 pk 交换,作为凸包的下一个顶点}
   p[k]：=p[j];
   p[j]：=t;
   th：=thmin;
   thmin：=4;
   FO  i：=k+1 TO n+1 DO  {依次搜索每一个非凸包点 pk+1 ...pn+1}
     BE  IN
```

```
            thl：= thete(p[k],p[i]); {计算 < pk,pi > 连线的优先级值}
            IF  ( hl< thmin) OR ((thl= thmin) AND (len(p[k],p[i]) >
                 len(p[k],p[j]))) AND ((th <2) OR (thl> th)) THEN
              BEG N
                 {若 < pk,pi > 的优先级目前最小(若存在相同优先级值的连线但 < pk,pi > 的长度较
                  长)且构造左链时优先级值递增,则 < pk,pi > 作为凸包的候选边}
                 j：= i;
                 thmin：= thl;
              END;  {then}
         END;  {for}
   UNTIL j = n + 1;  {直至返回最低点}
END;{wrap}

BE  IN
   writeln;
   write( 'filename: ');  {输入文件名串}
   readln(filename);
   assign(datafile,filename);  {文件名串与文件变量连接}
   reset(datafile);  {文件读准备}
   readln(datafile,n);  {读入树的棵数}
   FO  i：= 1 TO n DO  {读入每棵树的坐标}
      readln(datafile,p[i] .x,p[i] .y);
   close(datafile);
   wrap(r);  {计算凸包顶点返回凸包上的顶点数 r}
   writeln( 'the answer is : ');
   length_sum：=0;  {总绳长初始化}
   FO  i：= 1 TO r DO  {逆时针方向输出凸包上的顶点并累计总绳长}
      BE  IN
         writeln(p[i] .x:6,p[i] .y:6);
         IF  i< r)
            TH  N
               length_sum：= length_sum + sqrt(len(p[i],p[i + 1]))
            EL  E
               length_sum：= length_sum + sqrt(len(p[i],p[1]));
      END; {for}
   writeln( 'length =  ',length_sum:8:2);  {输出总绳长}
END .{main}
```

1 2　字串的模式匹配

一、试题

子串的定位操作通常称为串的模式匹配,是各种串处理系统中最重要的操作之一。我们

定义 index(s,t)为求模式串 t 在主串 s 中位置的函数。让模式串 t 自左至右在主串 s 上滑动进行匹配检查。设 q 为 s 中第一个与 t 相等的子串,若 q 存在,则函数 index(s,t)的值为 q 的首字符在 s 中的位置,否则函数值为零(t 不能是空串)。

例如: a = 'bei'; b = 'beiging'; c = 'i';

则 index(b,a) = 1;

index(b,c) = 3;

index(c,a) = 0;

定位函数的算法有很多,下面给出一种较好的算法——kmp 算法。

算法简述:

```
CONST maxlen = 串被确认的最大长度;
TYPE
      strtp = RE  ORD
                ch:ARRAY[1 . .maxlen] OF char;
                curlen:0 . .maxlen ;
              END;
PROC get_next (t:strtp );
     VAR next:ARRAY[1 . .t .curlen] OF integer;
     {求模式串 t 的 next 函数值并存入数组 next 中}
     j:= 1;
     k:= 0;
     next[1]:= 0;
     WHILE j < t .curlen DO
     IF  k = 0) OR (t .ch[j] = t .ch[k])
        THEN [j:= j + 1;k:= k + 1;next[j]:= k]
        ELSE k:= next[k]
ENDP; {get_next}

FUNC index(s,t :strtp):integer;
{利用模式串 t 的 next 函数值求模式串 t 在主串 s 中位置的 kmp 算法}
i:= 1;
j:= 1;
WH LE (i< =s .curlen) AND (j< =t .curlen) DO
    IF  j = 0) OR (s .ch[i] = t .ch[j])
       THEN [i:= i+ 1;j:= j + 1;]
       ELSE j:= next[j];
    IF j > t .curlen
       THEN return (i - t .curlen)
       ELSE return(0)
    ENDIF; {index}
```

请仔细阅读以上算法,并完成下面两个问题:

【问题 1】 请说明 next 函数值表示的意义及其作用。