2002 年全国青少年信息学（计算机）
奥林匹克分区联赛复赛提高组试题解题报告

By 湖北 伍先军

E-mail:wuxianjun2001@sohu.com

题一 均分纸牌（存盘名 NOIPG1）

[问题描述]

有 N 堆纸牌，编号分别为 1，2，…, N。每堆上有若干张，但纸牌总数必为 N 的倍数。可以在任一堆上取若干张纸牌，然后移动。

移牌规则为：在编号为 1 堆上取的纸牌，只能移到编号为 2 的堆上；在编号为 N 的堆上取的纸牌，只能移到编号为 N-1 的堆上；其他堆上取的纸牌，可以移到相邻左边或右边的堆上。

现在要求找出一种移动方法，用最少的移动次数使每堆上纸牌数都一样多。

例如 N=4，4 堆纸牌数分别为：

① 9 ② 8 ③ 17 ④ 6

移动 3 次可达到目的：

从 ③ 取 4 张牌放到 ④ （9 8 13 10） -> 从 ③ 取 3 张牌放到 ②（9 11 10 10）-> 从 ② 取 1 张牌放到①（10 10 10 10）。

[输 入]：

键盘输入文件名。文件格式：

N（N 堆纸牌，1 <= N <= 100）

A1 A2 … An （N 堆纸牌，每堆纸牌初始数，l<= Ai <=10000）

[输 出]：

输出至屏幕。格式为：

所有堆均达到相等时的最少移动次数。‘

[输入输出样例]

a.in：

```
4
9 8 17 6
```

屏慕显示：

```
3
```

分析：如果你想到把每堆牌的张数减去平均张数，题目就变成移动正数，加到负数中，使大家都变成 0，那就意味着成功了一半！拿例题来说，平均张数为 10,原张数 9，8，17，6，变为-1，-2，7，-4，其中没有为 0 的数，我们从左边出发：要使第 1 堆的牌数-1 变为 0，只须将-1 张牌移到它的右边（第 2 堆）-2 中；结果是-1 变为 0，-2 变为-3，各堆牌张数变为 0，-3，7，-4；同理：要使第 2 堆变为 0，只需将-3 移到它的右边（第 3 堆）中去，各堆牌张数变为 0，0，4，-4；要使第 3 堆变为 0，只需将第 3 堆中的 4 移到它的右边（第 4 堆）中去，结果为 0，0，0，0，完成任务。每移动 1 次牌，步数加 1。也许你要问，负数张牌怎么移，不违反题意吗？其实从第 i 堆移动-m 张牌到第 i+1 堆，等价于从第 i+1 堆移动 m 张牌到第 i 堆，步数是一样的。

如果张数中本来就有为 0 的，怎么办呢？如 0，-1，-5，6，还是从左算起（从右算起也完全一样），第 1 堆是 0，无需移牌，余下与上相同；再比如-1，-2，3，10，-4，-6，从左算起，第 1 次移动的结果为 0，-3，3，10，-4，-6；第 2 次移动的结果为 0，0，0，10，-4，-6，现在第 3 堆已经变为 0 了，可节省 1 步，余下继续。

程序清单

```
program NOIPG1;
 const maxn=100;
 var i,j,n,step:integer;ave:longint;
    a:array[1..maxn]of integer;
    f:text;filename:string;
 begin
   write('Input filename:');readln(filename);
   assign(f,filename);reset(f);
   readln(f,n);ave:=0;step:=0;
   for i:=1 to n do begin
     read(f,a[i]); inc(ave,a[i]);
   end;
   ave:=ave div i;
   for i:=1 to n do a[i]:=a[i]-ave;
   i:=1;j:=n;
   while a[i]=0 do inc(i);{过滤左边的 0}
   while a[j]=0 do dec(j);{过滤右边的 0}
   while (i<j) do begin
     inc(a[i+1],a[i]);{将第 i 堆牌移到第 i+1 堆中去}
     a[i]:=0;{第 i 堆牌移走后变为 0}
     inc(step);{移牌步数计数}
     inc(i);{对下一堆牌进行循环操作}
     while a[i]=0 do inc(i);{过滤移牌过程中产生的 0}
   end;
   writeln(step);
 end.
```

点评：基本题（较易） 本题有 3 点比较关键：一是善于将每堆牌数减去平均数，简化了问题；二是要过滤掉 0（不是所有的 0，如-2，3，0，-1 中的 0 是不能过滤的）；三是负数张牌也可以移动，这是辩证法（关键中的关键）。

题二 字串变换 （存盘名: NOIPG2）

[问题描述]:

已知有两个字串 A$, B$ 及一组字串变换的规则（至多 6 个规则）：

A1$ -> B1$

A2$ -> B2$

规则的含义为：在 A$中的子串 A1$ 可以变换为 B1$、A2$ 可以变换为 B2$ …。

例如：A$ = 'abcd'　B$ = 'xyz'

变换规则为：

‘abc’->‘xu’　‘ud’->‘y’　‘y’->‘yz’

则此时，A$ 可以经过一系列的变换变为 B$，其变换的过程为：

‘abcd’->‘xud’->‘xy’->‘xyz’

共进行了三次变换，使得 A$ 变换为 B$。

[输入]:

键盘输人文件名。文件格式如下：

A$ B$

A1$ B1$ \

A2$ B2$ |-> 变换规则

... ... /

所有字符串长度的上限为 20。

[输出]:

输出至屏幕。格式如下：

若在 10 步（包含 10 步）以内能将 A$ 变换为 B$ ，则输出最少的变换步数；否则输出"NO ANSWER!"

[输入输出样例]

b.in:

abcd xyz

abc xu

ud y

y yz

屏幕显示：

3

分析：本题是典型的广度优先搜索的例子，但如果只采用正向搜索，某些情况下计算量过大，速度过慢，故采取双向搜索且判重并适当剪枝，效果较好。

程序清单

```
{$A-,B-,D-,E-,F-,G-,I-,L-,N-,O-,P-,Q-,R-,S-,T-,V-,X-,Y-}
{$M 8192,0,655360}
program NOIPG2;
  const maxn=2300;
  type
    node=record{定义节点数据类型}
      str:string[115];dep:byte;
    end; {str 表示字串，其长度不会超过 115（长度超过 115 的字串不可能通过变换成为目标字串，因为题目限定变换 10 次之内，且串长不超过 20，即起始串最多可经过 5 次变换时增长，中间串的最大长度为 20+5*19=115，否则经过余下的步数不可能变为长度不超过 20 的目标串），dep 表示深度}
  ctype=array[1..maxn]of ^node;
  bin=0..1;
var
  maxk:byte;c:array [0..1]of ctype;
  x0:array[0..6,0..1]of string[20];
  filename:string;
  open,closed:array [0..1] of integer;
procedure Init;{读取数据，初始化}
  var f:text;temp:string;i,j:integer;
  begin
    for i:=0 to 1 do
      for j:=1 to maxn do new(c[i,j]);
    write('Input filename:');readln(filename);
    assign(f,filename);reset(f);i:=0;
    while not eof(f) and (i<=6) do begin
      readln(f,temp);
      x0[i,0]:=copy(temp,1,pos(' ',temp)-1);
      x0[i,1]:=copy(temp,pos(' ',temp)+1,length(temp));
      inc(i);
    end;
    maxk:=i-1;close(f);
  end;
procedure calc;
  var i,j,k:integer;st:bin;
      d:string;f:text;
  procedure bool(st:bin);{判断是否到达目标状态或双向搜索相遇}
    var i:integer;
    begin
      if x0[0,1-st]=c[st,closed[st]]^.str then begin
        {如果到达目标状态，则输出结果，退出}
        writeln(c[st,closed[st]]^.dep);
        halt;
      end;
      for i:=1 to closed[1-st] do
        if c[st,closed[st]]^.str=c[1-st,i]^.str then begin
          {如果双向搜索相遇（即得到同一节点），
```

```
        则输出结果（2 个方向搜索的步数之和），退出}
        writeln(c[st,closed[st]]^.dep+c[1-st,i]^.dep);
        halt;
      end;
  end;
procedure checkup(st:bin);{判断节点是否与前面重复}
  var i:integer;
  begin
    for i:=1 to closed[st]-1 do
      if c[st,i]^.str=c[st,closed[st]]^.str then begin
        dec(closed[st]);exit;{如果节点重复，则删除本节点}
      end;
    bool(st);{如果节点不重复，再判断是否到达目标状态}
  end;
procedure expand(st:bin);{扩展产生新节点}
  var i,j,k,lx,ld:integer;
  begin
    inc(open[st]);d:=c[st,open[st]]^.str;{队首节点出队}
    k:=c[st,open[st]]^.dep;ld:=length(d);
    for i:=1 to maxk do begin
      {从队首节点（父节点）出发产生新节点（子节点）}
      lx:=length(x0[i,st]);
      for j:=1 to ld do begin
        if (copy(d,j,lx)=x0[i,st]) and (length(copy(d,1,j-1)+x0[i,1-st]
           +copy(d,j+lx,ld))<=115) then begin
        {如果新节点的串长超过 115，则不扩展！即剪掉此枝}
          if closed[st]>=maxn then exit;{如果队列已满，只好退出}
          inc(closed[st]);{新节点入队}
                c[st,closed[st]]^.str:=copy(d,1,j-1)+x0[i,1-st]+copy(d,j+lx,ld);
          c[st,closed[st]]^.dep:=k+1;{子节点深度=父节点深度+1}
          checkup(st);{检查新节点是否重复}
        end;
      end;
    end;
  end;
Begin
  for st:=0 to 1 do begin{正向(st=0)逆向(st=1)搜索节点队列初始化}
    open[st]:=0;closed[st]:=1;
    c[st,closed[st]]^.str:=x0[0,st];c[st,closed[st]]^.dep:=0;
    bool(st);
  end;
  repeat
    {选择节点数较少且队列未空、未满、深度未达到 10 的方向先扩展}
    if (open[0]<=open[1]) and not ((open[0]>=closed[0]) or
      (closed[0]>=maxn) or (c[0,closed[0]]^.dep>10)) then expand(0);
    if (open[1]<=open[0]) and not ((open[1]>=closed[1]) or
      (closed[1]>=maxn) or (c[1,closed[1]]^.dep>10)) then expand(1);
    {如果一方搜索终止，继续另一方的搜索，直到两个方向都终止}
    if not ((open[0]>=closed[0]) or (closed[0]>=maxn) or
      (c[0,closed[0]]^.dep>10)) then expand(0);
    if not ((open[1]>=closed[1]) or (closed[1]>=maxn) or
      (c[1,closed[1]]^.dep>10)) then expand(1);
  until (open[0]>=closed[0]) or (c[0,closed[0]]^.dep>10) or (closed[0]>=maxn)
     and (closed[1]>=maxn) or (open[1]>=closed[1]) or (c[1,closed[1]]^.dep>10);
  {终止条件：任一方队空（无解）或搜索深度超过 10（10 步内无解）
   或双方均溢出（可能有解也可能无解，应尽量避免，要尽量把节
   点数组开大一点，采用双向搜索，采取剪枝措施等）}
  End;
BEGIN
  init; calc; writeln('NO ANSWER!')
END.
```

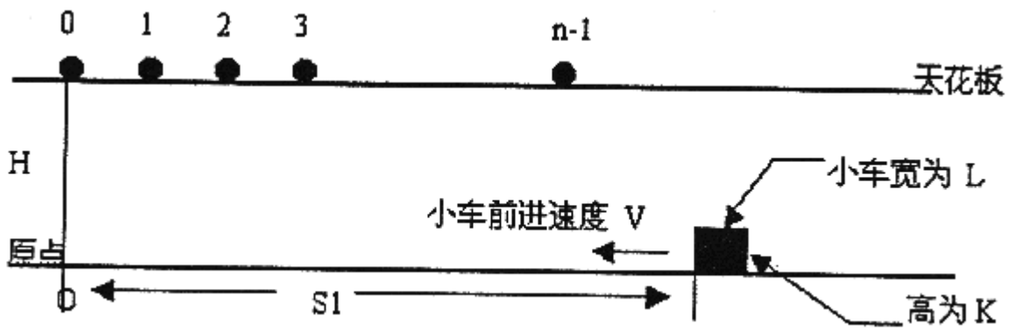
点评：基本题（较难） 考察队列、（双向）广度优先搜索算法及字符串的运算，基本上可以考察出参赛者的数据结构和算法水平。

题三 自由落体（存盘名:NOIPG3）

[问题描述]:

在高为 H 的天花板上有 n 个小球，体积不计，位置分别为 0，1，2，…．n-1。在地面上有一个小车（长为 L，高为 K，距原点距离为 S1）。已知小球下落距离计算公式为 d = 1/2*g*(t^2)，其中 g=10，t 为下落时间。地面上的小车以速度 V 前进。

如下图：



小车与所有小球同时开始运动，当小球距小车的距离 <= 0.00001 时，即认为小球被小车接受（小球落到地面后不能被接受）。

请你计算出小车能接受到多少个小球。

[输入]：

键盘输人：

H，S1，V，L，K，n （l<=H，S1，V，L，K，n <=100000）

[输出]：

屏幕输出：

小车能接受到的小球个数。

[输入输出样例]

[输入]:

5.0 9.0 5.0 2.5 1.8 5

[输出]:

1

分析：显然，小车太慢（即 V<=Vmin）或太快（V>Vmax）时，一个球也接不到。即在 V<=Vmin 或 V>Vmax 时输出为 0。下面分别求 Vmin 和 Vmax。当第 n-1 个小球落地的瞬间，小车在小球的右端离小球尚有 e=0.00001 的距离，小车的这个极小速度就是 Vmin。小车从天花板落到地面的时间 t1= $\sqrt{\frac{2H}{g}}$,这段时间内小车走了 S1-（n-1）-e,所以 Vmin= $\frac{S_1-(n-1)-e}{t_1}$ 。当第 1 个小球落到距小车的上表面为 e 的瞬间，小车在小球的左端离小球距离为 e，小车的这个极大速度就是 Vmax。小球从天花板落到离小车上表面为 e 的距离的时间为 t2= $\sqrt{\frac{2(H-K-e)}{g}}$,小车移动的距离为 S1+L+e,所以 Vmax= $\frac{S_1+L+e}{t_2}$ 。

那么，当 Vmin<V<=Vmax 时，就可接到球了。显然，时间段[t2,t1]是小车接球的时间，在 t2 时刻，小车的位置为：左表面离原点距离为 S1-V*t2,右表面离原点距离为 S1-V*t2+L；在 t1 时刻，小车的位置为：左表面离原点距离为 S1-V*t1,右表面离原点距离为 S1-V*t1+L；故小车的接球范围(在小车运动范围外扩展 e)为[S1-V*t1-e，S1-V*t2+L+e],球的个数就等于接球范围内所包含的 0~n-1 之间的整数的个数.

程序清单

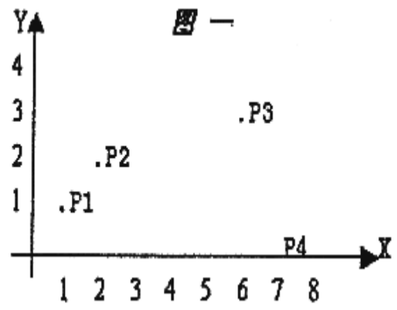
```
program NOIPG3;
  const g=10{重力加速度};e=1E-5;{小车接受小球的极限距离}
  var H,s1,v,l,k,t1,t2,Vmin,Vmax:real;
      n2,n1,num,n:integer;
  begin
    readln(h,s1,v,l,k,n);num:=-1;
    t1:=sqrt(2*h/g);{小球落地时间}
    if h<=k+e then t2:=0 else t2:=sqrt(2*(h-k-e)/g);{小球落到小车上的最短时间}
    if s1-v*t2+L+e<0
      then num:=0
      else n2:=trunc(s1-v*t2+L+e);{小车接受的球的最大编号为 n2}
    if n2>n-1 then n2:=n-1;{n2 取 trunc(s1-v*t2+L+e)与 n-1 的较小值}
    if s1-v*t1-e<=0
      then n1:=0
      else if s1-v*t1-e>n-1
           then num:=0
           else if (s1-v*t1-e)=trunc(s1-v*t1-e)
                then n1:=trunc(s1-v*t1-e){小车接受的球的最小编号为 n1}
                else n1:=trunc(s1-v*t1-e)+1;
    if num=-1 then num:=n2-n1+1;{小车接受的球的个数为 num}
    writeln(num);
  end.
```

点评：送分题 本题“物理味”有余而“信息味”不足，连循环语句都用不上！难见的“送分题”，可物理较差的人也得不到多少分哦！

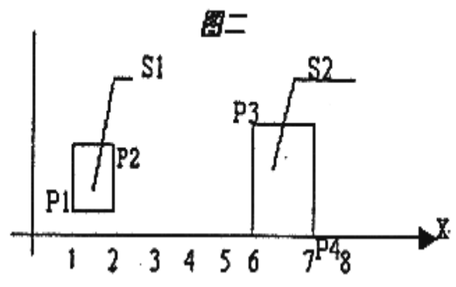
题四 矩形覆盖（存盘名 NOIPG4）

[问题描述]:

在平面上有 n 个点（n <= 50），每个点用一对整数坐标表示。例如：当 n＝4 时，4 个点的坐标分另为：
p1（1，1），p2（2，2），p3（3，6），P4（0，7），见图一。



图一



图二

这些点可以用 k 个矩形（1<=k<=4）全部覆盖，矩形的边平行于坐标轴。当 k=2 时，可用如图二的两个矩形 sl，s2 覆盖，s1，s2 面积和为 4。问题是当 n 个点坐标和 k 给出后，怎样才能使得覆盖所有点的 k 个矩形的面积之和为最小呢。约定：覆盖一个点的矩形面积为 0；覆盖平行于坐标轴直线上点的矩形面积也为 0。各个矩形必须完全分开（边线与顶点也都不能重合）。

[输入]:

键盘输人文件名。文件格式为

n k
xl y1
x2 y2
... ...
xn yn （0<=xi,yi<=500)

[输出]:

输出至屏幕。格式为：

一个整数，即满足条件的最小的矩形面积之和。

[输入输出样例]

d.in :

4 2
1 1
2 2
3 6
0 7

屏幕显示：

4

分析

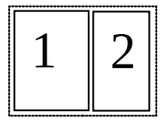
1、本题的难度较大。如果你这样认为：即在假定已用 i 个矩形（面积和满足最小）覆盖所有点的基础上，穷举所有 2 个矩形合并成 1 个矩形（条件是：在所有合并方案中使合并后面积最小），从而使矩形个数减少为 i-1——那就错了，可是却可以通过前 4 组测试数据！

正确的做法是对不同的 K 值分别进行计算，好在 K 值较小，否则...

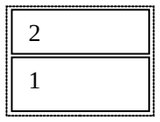
讨论：

k=1，只要求出 n 个点坐标的最大、最小值，就可求得矩形的位置与面积；

k=2，有 2 个矩形，它们只有 2 种分布形式：左右式（flag=0），上下式（flag=1）



flag=0

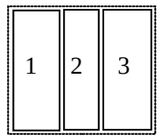


flag=1

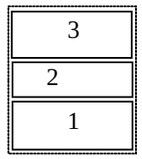
对于左右式，显然要先将所有点按横坐标升序排列，可将点 1~点 i-1 放入矩形 1 中，将点 i~点 n 放入矩形 2 中，求两矩形的面积之和；如果面积和比上一个值小，记下；让 i 从 2 循环到 n,就可完成左右式的全部搜索；

对于上下式，先将所有点按纵坐标升序排列，依此类推。

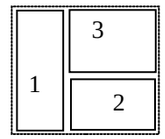
k=3，有 3 个矩形，它们有 6 种分布形式：



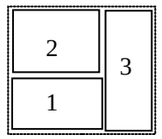
flag=0



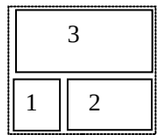
flag=1



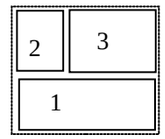
flag=2



flag=3



flag=4



flag=5

要用两重循环进行搜索：设 i,j 为循环变量，将点 1~i-1 放入矩形 1 中，点 i~j-1 放入矩形 2 中，点 j~n 放入矩形 3 中；点必须在放入前排好序（均为升序）：对于 flag=0,所有点按横坐标排序；对于 flag=1,所有点按纵坐标排序；对于 flag=2,所有点先按横坐标排序，然后点 i~n 按纵坐标排序；对于 flag=3,所有点先按横坐标排序，然后点 1~j-1 按纵坐标排序；对于 flag=4,所有点先按纵坐标排序，然后点 1~j-1 按横坐标排序；对于 flag=5,所有点先按纵坐标排序，然后点 i~n 按横坐标排序；至于 k=4,4 个矩形有 22 种分布形式，实在太复杂！幸好测试数据中没有 K=4 的情形(似乎有意放了一马？)。据说本题全国没有一人全对！（只要求

K=1，2，3）

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 NOIPG4;
 const maxn=50;maxk=3;
 type  rect=record{定义"矩形"数据类型}
        l,r,t,b:word;{矩形的左边，右边，下边，上边距坐标轴的距离}
       end;
       vxy=record{定义"点"数据类型}
         x,y:word;{点的横、纵坐标}
       end;
 var  ju:array[1..maxk]of rect;
      v:array[1..maxn,0..2] of vxy;v0:vxy;
      n,k,i,j,ii,jj:byte;f:text;filename:string;
      Smin,temp:longint;
 function intersect(jui,juj:rect):boolean;{判断两矩形是否有公共点}
  var b1,b2,t1,t2,l1,l2,r1,r2:word;
  begin
    b1:=jui.b;b2:=juj.b;t1:=jui.t;t2:=juj.t;
    l1:=jui.l;l2:=juj.l;r1:=jui.r;r2:=juj.r;
    intersect:=((l2<=r1) and (l2>=l1) or (r2<=r1) and (r2>=l1) or (l2<=l1)
        and (r2>=r1)) and ((t2<=b1) and (t2>=t1) or (b2<=b1) and (b2>=t1)
        or (b2>=b1) and (t2<=t1));
  end;
function area(ju:rect):longint;{求矩形的面积}
 var temp:longint;
 begin
        temp:=ju.b-ju.t;area:=temp*(ju.r-ju.l);
        {不能直接写成 area:=(ju.b-ju.t)*(ju.r-ju.l);因为这样可能会溢出！}
 end;
procedure insert(v:vxy;var ju:rect);{将点放入矩形}
 begin
   if v.x<ju.l then ju.l:=v.x;
   if v.x>ju.r then ju.r:=v.x;
   if v.y<ju.t then ju.t:=v.y;
   if v.y>ju.b then ju.b:=v.y;
 end;
procedure init;{初始化}
 begin
   write('Input filename:');readln(filename);
   assign(f,filename);reset(f);readln(f,n,k);
   for i:=1 to n do begin
     read(f,v[i,0].x,v[i,0].y);
     v[i,1].x:=v[i,0].x;v[i,1].y:=v[i,0].y;
   end;
   for i:=1 to n-1 do{按横坐标升序排列各点，存入 v[i,0]}
     for j:=i+1 to n do
       if v[i,0].x>v[j,0].x then begin
         v0:=v[i,0];v[i,0]:=v[j,0];v[j,0]:=v0;
       end;
   for i:=1 to n-1 do{按纵坐标升序排列各点，存入 v[i,1]}
     for j:=i+1 to n do
       if v[i,1].y>v[j,1].y then begin
         v0:=v[i,1];v[i,1]:=v[j,1];v[j,1]:=v0;
       end;
 end;
procedure solve;{核心计算}
 begin
   smin:=maxlongint;
   case k of
     1:begin{K=1 的情形}
        ju[1].b:=v[n,1].y;ju[1].t:=v[1,1].y;
        ju[1].r:=v[n,0].x;ju[1].l:=v[1,0].x;
        smin:=area(ju[1]);
       end;
   2:for jj:=0 to 1 do begin{K=2 的情形}
      {flag=0,1 的情形}
      ju[1].b:=v[1,jj].y;ju[1].t:=v[1,jj].y;
      ju[1].r:=v[1,jj].x;ju[1].l:=v[1,jj].x;
      for i:=2 to n do begin
        insert(v[i-1,jj],ju[1]);{将第 i-1 点放入矩形 1}
        ju[2].b:=v[i,jj].y;ju[2].t:=v[i,jj].y;{将第 i 至 n 点放入矩形 2}
        ju[2].r:=v[i,jj].x;ju[2].l:=v[i,jj].x;
        for ii:=i+1 to n do insert(v[ii,jj],ju[2]);
        if not intersect(ju[1],ju[2]) then begin{如果两矩形不交叉}
          temp:=0;for ii:=1 to k do temp:=temp+area(ju[ii]);
          if temp<smin then smin:=temp;
        end;
      end;
     end;
   3:begin
      for jj:=0 to 1 do begin {flag=0,1 的情形}
        ju[1].b:=v[1,jj].y;ju[1].t:=v[1,jj].y;
```

```
  ju[1].r:=v[1,jj].x;ju[1].l:=v[1,jj].x;
  for i:=2 to n-1 do begin
    insert(v[i-1,jj],ju[1]);
    ju[2].b:=v[i,jj].y;ju[2].t:=v[i,jj].y;
    ju[2].r:=v[i,jj].x;ju[2].l:=v[i,jj].x;
    if intersect(ju[1],ju[2]) then continue;
    for j:=i+1 to n do begin
      insert(v[j-1,jj],ju[2]);
      ju[3].b:=v[j,jj].y;ju[3].t:=v[j,jj].y;
      ju[3].r:=v[j,jj].x;ju[3].l:=v[j,jj].x;
      for ii:=j+1 to n do insert(v[ii,jj],ju[3]);
      if intersect(ju[2],ju[3]) then continue;
      temp:=0;for ii:=1 to k do temp:=temp+area(ju[ii]);
      if temp<smin then smin:=temp;
    end;
  end;
end;

{flag=2 的情形:先竖直划分大矩形；再在右矩形中水平划分}
ju[1].b:=v[1,0].y;ju[1].t:=v[1,0].y;
ju[1].r:=v[1,0].x;ju[1].l:=v[1,0].x;
      for i:=2 to n-1 do begin
          for ii:=1 to n do v[ii,2]:=v[ii,0];{所有点按横坐标升序排列，存入
v[i,2]}
        for ii:=i to n-1 do{将点 i 至 n 按纵坐标升序排列，存入 v[i,2]}
          for jj:=ii+1 to n do
            if v[ii,2].y>v[jj,2].y then begin
              v0:=v[ii,2];v[ii,2]:=v[jj,2];v[jj,2]:=v0;
            end;{结果：所有点先按横坐标升序排列，然后点 i 至 n 按纵坐标升
序排列}
        insert(v[i-1,2],ju[1]);{将第 i-1 点放入矩形 1}
        ju[2].b:=v[i,2].y;ju[2].t:=v[i,2].y;{将第 i 点放入矩形 2}
        ju[2].r:=v[i,2].x;ju[2].l:=v[i,2].x;
        if intersect(ju[1],ju[2]) then continue;
        for j:=i+1 to n do begin
          insert(v[j-1,2],ju[2]);{将第 j-1 点放入矩形 2}
          ju[3].b:=v[j,2].y;ju[3].t:=v[j,2].y;{将第 j 至 n 点放入矩形 3}
          ju[3].r:=v[j,2].x;ju[3].l:=v[j,2].x;
          for ii:=j+1 to n do insert(v[ii,2],ju[3]);
          if intersect(ju[2],ju[3]) then continue;
          temp:=0;for ii:=1 to k do temp:=temp+area(ju[ii]);
          if temp<smin then smin:=temp;
    end;
  end;
end;

{flag=3 的情形}
for j:=3 to n do begin
  for ii:=1 to n do v[ii,2]:=v[ii,0];
  for ii:=1 to j-2 do
    for jj:=ii+1 to j-1 do
      if v[ii,2].y>v[jj,2].y then begin
        v0:=v[ii,2];v[ii,2]:=v[jj,2];v[jj,2]:=v0;
      end;
  ju[3].b:=v[j,2].y;ju[3].t:=v[j,2].y;
  ju[3].r:=v[j,2].x;ju[3].l:=v[j,2].x;
  for ii:=j+1 to n do insert(v[ii,2],ju[3]);
  for i:=2 to j-1 do begin
    ju[2].b:=v[i,2].y;ju[2].t:=v[i,2].y;
    ju[2].r:=v[i,2].x;ju[2].l:=v[i,2].x;
    for ii:=i+1 to j-1 do insert(v[ii,2],ju[2]);
    ju[1].b:=v[1,2].y;ju[1].t:=v[1,2].y;
    ju[1].r:=v[1,2].x;ju[1].l:=v[1,2].x;
    for ii:=2 to i-1 do insert(v[ii,2],ju[1]);
    if intersect(ju[1],ju[2]) or intersect(ju[2],ju[3]) or
      intersect(ju[1],ju[3]) then continue;
    temp:=0;for ii:=1 to k do temp:=temp+area(ju[ii]);
    if temp<smin then smin:=temp;
  end;
end;

{flag=4 的情形}
for j:=3 to n do begin
  for ii:=1 to n do v[ii,2]:=v[ii,1];
  for ii:=1 to j-2 do
    for jj:=ii+1 to j-1 do
      if v[ii,2].x>v[jj,2].x then begin
        v0:=v[ii,2];v[ii,2]:=v[jj,2];v[jj,2]:=v0;
      end;
  ju[3].b:=v[j,2].y;ju[3].t:=v[j,2].y;
  ju[3].r:=v[j,2].x;ju[3].l:=v[j,2].x;
  for ii:=j+1 to n do insert(v[ii,2],ju[3]);
  for i:=2 to j-1 do begin
    ju[2].b:=v[i,2].y;ju[2].t:=v[i,2].y;
    ju[2].r:=v[i,2].x;ju[2].l:=v[i,2].x;
```

```
    for ii:=i+1 to j-1 do insert(v[ii,2],ju[2]);
    ju[1].b:=v[1,2].y;ju[1].t:=v[1,2].y;
    ju[1].r:=v[1,2].x;ju[1].l:=v[1,2].x;
    for ii:=2 to i-1 do insert(v[ii,2],ju[1]);
    if intersect(ju[1],ju[2]) or intersect(ju[2],ju[3]) or
      intersect(ju[1],ju[3]) then continue;
    temp:=0;for ii:=1 to k do temp:=temp+area(ju[ii]);
    if temp<smin then smin:=temp;
  end;
end;

{flag=5 的情形}
ju[1].b:=v[1,1].y;ju[1].t:=v[1,1].y;
ju[1].r:=v[1,1].x;ju[1].l:=v[1,1].x;
for i:=2 to n-1 do begin
  for ii:=1 to n do v[ii,2]:=v[ii,1];
  for ii:=i to n-1 do
    for jj:=ii+1 to n do
      if v[ii,2].x>v[jj,2].x then begin
        v0:=v[ii,2];v[ii,2]:=v[jj,2];v[jj,2]:=v0;
      end;
      insert(v[i-1,2],ju[1]);
      ju[2].b:=v[i,2].y;ju[2].t:=v[i,2].y;
      ju[2].r:=v[i,2].x;ju[2].l:=v[i,2].x;
      if intersect(ju[1],ju[2]) then continue;
      for j:=i+1 to n do begin
        insert(v[j-1,2],ju[2]);
        ju[3].b:=v[j,2].y;ju[3].t:=v[j,2].y;
        ju[3].r:=v[j,2].x;ju[3].l:=v[j,2].x;
        for ii:=j+1 to n do insert(v[ii,2],ju[3]);
        if intersect(ju[2],ju[3]) then continue;
        temp:=0;for ii:=1 to k do temp:=temp+area(ju[ii]);
        if temp<smin then smin:=temp;
      end;
    end;
  end;
 end;
end;
begin{主程序}
  init;
  solve;
  writeln(smin);
```

end.

点评：压轴题 据说，本次复赛主要是前三题的竞争，可见本题能得分的人相当少，但是 K=1 应该说是送分的，K=2 也是比较容易的。通过测试，发现在 K=3 的第 4、5 组测试数据中仅用到了 flag=1 的情形，也就是说，只要写出 flag=1 的程序段就 OK 了（没写 flag=0,2,3,4,5 的同学偷着乐?）。