全国信息学奥林匹克联赛（NOIP2010）复赛

提高组
（请选手务必仔细阅读本页内容）

一. 题目概况
中文题目名称 机器翻译 乌龟棋 关押罪犯 引水入城
英文题目与子目录名 translate tortoise prison flow
可执行文件名 translate tortoise prison flow
输入文件名 translate.in tortoise.in prison.in flow.in
输出文件名 translate.out tortoise.out prison.out flow.out
每个测试点时限 1 秒 1 秒 1 秒 1 秒
测试点数目 10 10 10 10
每个测试点分值 10 10 10 10
附加样例文件 有 有 有 有
结果比较方式 全文比较（过滤行末空格及文末回车）
题目类型 传统 传统 传统 传统

二. 提交源程序文件名
对于 pascal 语言 translate.pas tortoise.pas prison.pas flow.pas
对于 C 语言 translate.c tortoise.c prison.c flow.c
对于 C++语言 translate.cpp tortoise.cpp prison.cpp flow.cpp

三. 编译命令（不包含任何优化开关）
对于 pascal 语言　fpc translate.pasfpc tortoise.pasfpc prison.pas fpc flow.pas
对于 C 语言
gcc -o translate
translate.c -lm
gcc -o tortoise
tortoise.c -lm
gcc -o prison
prison.c -lm
gcc -o flow
flow.c -lm
对于 C++语言 g++ -o translate
translate.cpp -lm
g++ -o tortoise

tortoise.cpp -lm
g++ -o prison
prison.cpp -lm
g++ -o flow
flow.cpp -lm

四. 运行内存限制
内存上限 128M 128M 128M 128M

注意事项：
1、文件名（程序名和输入输出文件名）必须使用英文小写。
2、C/C++中函数 main()的返回值类型必须是 int，程序正常结束时的返回值必须是 0。
3、全国统一评测时采用的机器配置为：CPU P4 3.0GHz，内存 1G，上述时限以此配置为准。各省在自测时可根据具体配置调整时限。

换页

1. 机器翻译
(translate.pas/c/cpp)
【问题描述】
小晨的电脑上安装了一个机器翻译软件，他经常用这个软件来翻译英语文章。
这个翻译软件的原理很简单，它只是从头到尾，依次将每个英文单词用对应的中文含义来替换。对于每个英文单词，软件会先在内存中查找这个单词的中文含义，如果内存中有，软件就会用它进行翻译；如果内存中没有，软件就会在外存中的词典内查找，查出单词的中文含义然后翻译，并将这个单词和译义放入内存，以备后续的查找和翻译。
假设内存中有 M 个单元，每单元能存放一个单词和译义。每当软件将一个新单词存入内存前，如果当前内存中已存入的单词数不超过 M?1，软件会将新单词存入一个未使用的内存单元；若内存中已存入 M 个单词，软件会清空最早进入内存的那个单词，腾出单元来，存放新单词。
假设一篇英语文章的长度为少次词典？假设在翻译开始前，内存中没有任何单词。

【输入】
输入文件名为 translate.in，输入文件共 2 行。每行中两个数之间用一个空格隔开。
第一行为两个正整数 M 和 N，代表内存容量和文章的长度。
第二行为 N 个非负整数，按照文章的顺序，每个数（大小不超过 1000）代表一个英文单词。文章中两个单词是同一个单词，当且仅当它们对应的非负整数相同。

【输出】
输出文件 translate.out 共 1 行，包含一个整数，为软件需要查词典的次数。

【输入输出样例 1】
translate.in translate.out

3 7
1 2 1 5 4 4 1

5

【输入输出样例 1 说明】
整个查字典过程如下：每行表示一个单词的翻译，冒号前为本次翻译后的内存状况：
空：内存初始状态为空。
1. 1：查找单词 1 并调入内存。
2. 1 2：查找单词 2 并调入内存。
3. 1 2：在内存中找到单词 1。
4. 1 2 5：查找单词 5 并调入内存。
5. 2 5 4：查找单词 4 并调入内存替代单词 1。
6. 2 5 4：在内存中找到单词 4。
7. 5 4 1：查找单词 1 并调入内存替代单词 2。
共计查了 5 次词典。

【输入输出样例 2】
translate.in translate.out
2 10
8 824 11 78 11 78 11 78 8 264

6

【数据范围】
对于 10%的数据有 M=1，N≤5。
对于 100%的数据有 0≤100，0≤1000。

2. 乌龟棋
(tortoise.pas/c/cpp)
【问题描述】
小明过生日的时候，爸爸送给他一副乌龟棋当作礼物。
乌龟棋的棋盘是一行 N 个格子，每个格子上一个分数（非负整数）。棋盘第 1 格是唯一的起点，第 N 格是终点，游戏要求玩家控制一个乌龟棋子从起点出发走到终点。

……
1 2 3 4 5 ……N

乌龟棋中 M 张爬行卡片，分成 4 种不同的类型（M 张卡片中不一定包含所有 4 种类型的卡片，见样例），每种类型的卡片上分别标有 1、2、3、4 四个数字之一，表示使用这种卡片后，乌龟棋子将向前爬行相应的格子数。游戏中，玩家每次需要从所有的爬行卡片中选择一张之前没有使用过的爬行卡片,控制乌龟棋子前进相应的格子数,每张卡片只能使用一次。
游戏中，乌龟棋子自动获得起点格子的分数，并且在后续的爬行中每到达一个格子，就得到该格子相应的分数。玩家最终游戏得分就是乌龟棋子从起点到终点过程中到过的所有格子的分数总和。
很明显，用不同的爬行卡片使用顺序会使得最终游戏的得分不同，小明想要找到一种卡片使用顺序使得最终游戏得分最多。
现在，告诉你棋盘上每个格子的分数和所有的爬行卡片，你能告诉小明，他最多能得到多
【输入】
输入文件名 tortoise.in。输入文件的每行中两个数之间用一个空格隔开。
第 1 行 2 个正整数 N 和 M，分别表示棋盘格子数和爬行卡片数。
第 2 行 N 个非负整数，a1a2

……aN

，其中 ai 表示棋盘第 i 个格子上的分数。
第 3 行 M 个整数，b1b2

……bM

，表示 M 张爬行卡片上的数字。
输入数据保证到达终点时刚好用光 M 张爬行卡片，即 N?1=Σ
M
ib
1
。
【输出】
输出文件名 tortoise.out。

输出只有 1 行，1 个整数，表示小明最多能得到的分数。

【输入输出样例 1】
tortoise.in tortoise.out
9 5
6 10 14 2 8 8 18 5 17
1 3 1 2 1

73

【输入输出样例 1 说明】
小明使用爬行卡片顺序为 1，1，3，1，2，得到的分数为 6+10+14+8+18+17=73。注意，由于起点是 1，所以自动获得第 1 格的分数 6。

【输入输出样例 2】
tortoise.in tortoise.out
13 8
4 96 10 64 55 13 94 53 5 24 89 8 30
1 1 1 1 1 2 4 1

455

【数据范围】
对于 30%的数据有 1
≤
N
≤
30，1
≤
M
≤
12。
对于 50%的数据有 1≤N≤120，1
≤
M
≤
50，且 4 种爬行卡片，每种卡片的张数不会超过 20。
对于 100%的数据有 1≤N≤350，1≤M≤120，且 4 种爬行卡片，每种卡片的张数不会超过 40；0≤ai≤100，1≤i≤N；1≤bi≤4，1≤i≤M。输入数据保证 N?1=
∑
M
ib
1
。

3．关押罪犯
(prison.pas/c/cpp)
【问题描述】
S 城现有两座监狱，一共关押着 N 名罪犯，编号分别为 1~N。他们之间的关系自然也极不和谐。很多罪犯之间甚至积怨已久，如果客观条件具备则随时可能爆发冲突。我们用“怨气值”（一个正整数值）来表示某两名罪犯之间的仇恨程度，怨气值越大，则这两名罪犯之间的积怨越多。如果两名怨气值为 c 的罪犯被关押在同一监狱，他们俩之间会发生摩擦，并

造成影响力为 c 的冲突事件。
每年年末，警察局会将本年内监狱中的所有冲突事件按影响力从大到小排成一个列表，然后上报到 S 城 Z 市长那里。公务繁忙的 Z 市长只会去看列表中的第一个事件的影响力，如果影响很坏，他就会考虑撤换警察局长。
在详细考察了 N 名罪犯间的矛盾关系后，警察局长觉得压力巨大。他准备将罪犯们在两座监狱内重新分配，以求产生的冲突事件影响力都较小，从而保住自己的乌纱帽。假设只要处于同一监狱内的某两个罪犯间有仇恨，那么他们一定会在每年的某个时候发生摩擦。那么，应如何分配罪犯，才能使 Z 市长看到的那个冲突事件的影响力最小？这个最小值是多

少？

【输入】
输入文件名为 prison.in 第一行为两个正整数 N 和 M，分别表示罪犯的数目以及存在仇恨的罪犯对数。
接下来的 M 行每行为三个正整数 aj，bj，cj，表示 aj 号和 bj 号罪犯之间存在仇恨，其怨气值为 cj。数据保证 Nba
jj
≤1，0000000001

0≤
jc
，且每对罪犯组合只出现一
次。

【输出】
输出文件 prison.out 共 1 行，为 Z 市长看到的那个冲突事件的影响力。如果本年内监狱中未发生任何冲突事件，请输出 0。

【输入输出样例】
prison.in prison.out
4 6
1 4 2534
2 3 3512
1 2 28351
1 3 6618
2 4 1805
3 4 12884

3512

【输入输出样例说明】
罪犯之间的怨气值如下面左图所示，右图所示为罪犯的分配方法，市长看到的冲突事件影响力是 3512（由 2 号和 3 号罪犯引发）。其他任何分法都不会比这个分法更优。

【数据范围】
对于 30%的数据有 N≤15。
对于 70%的数据有 N≤2000，M≤50000。
对于 100%的数据有 N≤20000，M≤100000。
2 1
3 4
1805 6618
2534 3512
12884
28351 2 1
3 4
2534 3512

4.引水入城
(flow.pas/c/cpp)
【问题描述】
湖泊

沙漠
在一个遥远的国度，一侧是风景秀美的湖泊，另一侧则是漫无边际的沙漠。该国的行政区划十分特殊，刚好构成一个 N 行 M 列的矩形，如上图所示，其中每个格子都代表一座城市，每座城市都有一个海拔高度。
为了使居民们都尽可能饮用到清澈的湖水，现在要在某些城市建造水利设施。水利设施有两种，分别为蓄水厂和输水站。蓄水厂的功能是利用水泵将湖泊中的水抽取到所在城市的蓄水池中。因此，只有与湖泊毗邻的第 1 行的城市可以建造蓄水厂。而输水站的功能则是通过输水管线利用高度落差，将湖水从高处向低处输送。故一座城市能建造输水站的前提，是存在比它海拔更高且拥有公共边的相邻城市，已经建有水利设施。
由于第 N 行的城市靠近沙漠，是该国的干旱区，所以要求其中的每座城市都建有水利设施。那么，这个要求能否满足呢？如果能，请计算最少建造几个蓄水厂；如果不能，求干旱区中不可能建有水利设施的城市数目。

【输入】
输入文件名为 flow.in。输入文件的每行中两个数之间用一个空格隔开。

输入的第一行是两个正整数 N 和 M，表示矩形的规模。
接下来 N 行，每行 M 个正整数，依次代表每座城市的海拔高度。

【输出】
输出文件名为 flow.out。
输出有两行。如果能满足要求，输出的第一行是整数 1，第二行是一个整数，代表最少建造几个蓄水厂；如果不能满足要求，输出的第一行是整数 0，第二行是一个整数几座干旱区中的城市不可能建有水利设施。

【输入输出样例 1】
flow.in flow.out
2 5
9 1 5 4 3
8 7 6 1 2

1
1

换页

【样例 1 说明】
只需要在海拔为 9 的那座城市中建造蓄水厂，即可满足要求。

【输入输出样例 2】
flow.in flow.out
3 6
8 4 5 6 4 4
7 3 4 3 3 3
3 2 2 1 1 2

1
3

【样例 2 说明】
湖泊
8 4 5 6 4 4
7 3 4 3 3 3
3 2 2 1 1 2
沙漠
上图中，在 3 个粗线框出的城市中建造蓄水厂，可以满足要求。以这 3 个蓄水厂为源头在干旱区中建造的输水站分别用 3 种颜色标出。当然，建造方法可能不唯一。

【数据范围】
本题共有 10 个测试数据，每个数据的范围如下表所示：
测试数据编号 能否满足要求 N M
1 不能 ≤ 10 ≤ 10
2 不能 ≤ 100 ≤ 100
3 不能 ≤ 500 ≤ 500
4 能 = 1 ≤ 10
5 能 ≤ 10 ≤ 10
6 能 ≤ 100 ≤ 20
7 能 ≤ 100 ≤ 50
8 能 ≤ 100 ≤ 100
9 能 ≤ 200 ≤ 200
10 能 ≤ 500 ≤ 500
对于所有的 10 个数据，每座城市的海拔高度都不超过 10
6
。
换页
第一题：translate

简单模拟队列的操作，不需要开布尔数组，也不需要循环队列，朴素就很快了。

```
program translate;

var q:array[0..10000]of longint;
    m,n,x,i,j:longint;
    head,tail,ans:longint;
    f:boolean;

begin
    assign(input,'translate.in');reset(input);
    assign(output,'translate.out');rewrite(output);
    readln(m,n);
    head:=1;
    tail:=0;
    ans:=0;
    for i:=1 to n do
        begin
            read(x);
            f:=false;
            for j:=head to tail do
                if x=q[j] then begin f:=true; break end;
            if f then continue;
            inc(tail);
```

```
            q[tail]:=x;
            inc(ans);
            if ans>m then inc(head);
        end;
    readln;
    writeln(ans);
    close(input);
    close(output);
end.
```

第二题：tortoise

简单动态规划，不能按一般的有并列项的时候降并列项的维度这个策略来做（也许可能做成功，但是我在考试的时候做了一个半小时还是挂了，于是果断写了个 50 分的）。正确的方法具体的思路是不用枚举步数,通过 4 个指针的关系将步数算出来。程序可以通过递归来实现，但是非递归的话写出来比较清晰，而且也只有四行，没有写递归的必要。

```
program tortoise;

var f:array[0..41,0..41,0..41,0..41]of longint;
    a:array[0..355]of longint;
    sum:array[1..4]of longint;
    n,m,i,x:longint;
    x1,x2,x3,x4:longint;

procedure max(var a,b:longint);
begin
    if b>a then a:=b;
end;

begin
    assign(input,'tortoise.in');reset(input);
    assign(output,'tortoise.out');rewrite(output);
    fillchar(f,sizeof(f),0);
    fillchar(sum,sizeof(sum),0);
    readln(n,m);
    for i:=1 to n do read(    readln;
    for i:=1 to m do
        begin
            read(x);
            inc(sum[x]);
        end;
    readln;
    f[0,0,0,0]:=a[1];
```

```
    for x1:=0 to sum[1] do
        for x2:=0 to sum[2] do
            for x3:=0 to sum[3] do
                for x4:=0 to sum[4] do
                    begin
                        if x1+x2+x3+x4=0 then continue;
                        f[x1,x2,x3,x4]:=0;
                        if x1>0 then max(f[x1,x2,x3,x4],f[x1-1,x2,x3,x4]);
                        if x2>0 then max(f[x1,x2,x3,x4],f[x1,x2-1,x3,x4]);
                        if x3>0 then max(f[x1,x2,x3,x4],f[x1,x2,x3-1,x4]);
                        if x4>0 then max(f[x1,x2,x3,x4],f[x1,x2,x3,x4-1]);
                        inc(f[x1,x2,x3,x4],a[x1+2*x2+3*x3+4*x4+1]);
                    end;
    writeln(f[sum[1],sum[2],sum[3],sum[4]]);
    close(input);
    close(output);
end.
```

第三题：prison

这个题目实在叫人无语，重拍一遍 AC，但省里测出来的是 60 分，具体情况还不知道究竟怎么了。话说我当时记得清清楚楚是 84 行，这次重拍 82 行（上次多了个如果 0 的时候可行直接输出的废话），完全一样的程序，只能情何以堪了。

话说还是前面那个话写的好，最大里面求最小，最小里面求最大，先想二分答案。再一看 ci<=1,000,000,000，还有什么好说的呢？二分一个 max，然后进行二分图的染色，染色的时候只看大于 max 的边，不超过的直接忽略。时间复杂度是 O((m+n)*log max(c[i]))；网上说的什么先快排，可以说是完全没有必要。

```
program prison;

type link=^node;
     node=record
        v,c:longint;
        next:link;
     end;

var g:array[1..20000]of link;
    n,m:longint;
    color:array[1..20000]of longint;
    low,mid,high,i:longint;
    f:boolean;

procedure insert(x,y,z:longint);
```

```
var p:link;
begin
    new(p);
    p^.v:=y;
    p^.c:=z;
    p^.next:=g[x];
    g[x]:=p;
end;

procedure init;
var x,y,z:longint;
begin
    readln(n,m);
    fillchar(g,sizeof(g),0);
    high:=0;
    for i:=1 to m do
        begin
            readln(x,y,z);
            insert(x,y,z);
            insert(y,x,z);
            if z>high then high:=z;
        end;
end;

procedure dfs(u:longint);
var p:link;
    v:longint;
begin
    p:=g[u];
    while p<>nil do
        begin
            if p^.c>mid then begin
                v:=p^.v;
                if color[v]=color[u] then begin f:=false; exit; end;
                if color[v]=0 then begin
                    color[v]:=3-color[u];
                    dfs(v);
                    if not f then exit;
                end;
            end;
            p:=p^.next;
        end;
end;
```

```
begin
    assign(input,'prison.in');reset(input);
    assign(output,'prison.out');rewrite(output);
    init;
    low:=0;
    while low<high do
        begin
            mid:=(low+high)>>1;
            fillchar            f:=true;
            for i:=1 to n do
                if color[i]=0 then begin
                    color[i]:=1;
                    dfs(i);
                    if not f then break;
                end;
            if f then high:=mid
                 else low:=mid+1;
        end;
    writeln(high);
    close(input);
    close(output);
end.
```

第四题：flow

考试的时候第一反应就是二分图，构造费用流，弄了半天没有弄出来，最后没有时间了，连60分的搜索都写不下去了，写了个40分的骗分酱油收场。

然后开始想啊想，后来在车上想到了下面的这么个方法。觉得还是蛮简单的，只是考试的时候想不到，真是个悲剧~~~~

首先先搞定3个不可以的非常简单，不再阐述。对于可以的，可以发现每个绿洲能FILL的沙漠必然是一个连续的区间，如果不是连续的区间，则必然没有不可以（根据拓扑学理论容易证明），这里不再细证。

这样一来，这个问题就具备最优子结构的性质了。自信观察不难发现，问题转化为了选取最少的区间覆盖数轴。状态转移方程是f[i]=min(f[a[j].st-1])+1，其中a[j]存储的是预处理的区间信息。

最后是时间复杂度，预处理的时间复杂度是O(n*m^2)，动态规划的时间复杂度是O(m^2)，对于题目中的数据，完全可以承受。

```
program flow;
```

```
const dx:array[1..4]of longint=(1,0,-1,0);
      dy:array[1..4]of longint=(0,1,0,-1);

var a:array[0..501,0..501]of longint;
    st,ed,f:array[0..500]of longint;
    g:array[1..500,1..500]of boolean;
    n,m:longint;
    q:array[0..250000]of record x,y:longint;end;

procedure bfs(i,j:longint);
var head,tail:longint;
    x,y,k:longint;
begin
    head:=0;
    tail:=1;
    q[1].x:=i;
    q[1].y:=j;
    g[i,j]:=true;
    while head<>tail do
        begin
            inc(head);
            i:=q[head].x;
            j:=q[head].y;
            for k:=1 to 4 do
                begin
                    x:=i+dx[k];
                    y:=j+dy[k];
                    if (a[x,y]<a[i,j])and(not g[x,y]) then begin
                        g[x,y]:=true;
                        inc(tail);
                        q[tail].x:=x;
                        q[tail].y:=y;
                    end;
                end;
        end;
end;

procedure Ending(x,y:longint);
begin
    writeln(x);
    writeln(y);
    close(input);
    close(output);
    halt;
```

```
end;

procedure init;
var i,j,p,q:longint;
    ok:array[1..500]of boolean;
    filled:longint=0;
begin
    fillchar(a,sizeof(a),127);
    readln(n,m);
    for i:=1 to n do
        begin
            for j:=1 to m do read(a[i,j]);
            readln;
        end;
    fillchar(ok,sizeof(ok),0);
    for j:=1 to m do
        begin
            fillchar(g,sizeof(g),0);
            g[1,j]:=tr            bfs(1,j);
            for i:=1 to m do ok[i]:=ok[i]or g[n,i];
            for p:=1 to m do
                if g[n,p] then break;
            if (p=m)and(not g[n,p]) then begin
                st[j]:=m+1;
                ed[j]:=0;
                continue;
            end;
            st[j]:=p;
            for q:=p+1 to m do
                if not g[n,q] then break;
            if (q=m)and(g[n,q]) then ed[j]:=q
                                else ed[j]:=q-1;
        end;
    for j:=1 to m do
        if ok[j] then inc(filled);
    if filled<m then Ending(0,m-filled);
end;

function min(a,b:longint):longint;
begin
    if a<b then exit(a)
           else exit(b);
end;
```

```
procedure dp;
var i,j:longint;
begin
    f[0]:=0;
    for i:=1 to m do
        begin
            f[i]:=maxlongint;
            for j:=1 to m do
                if (st[j]<=i)and(ed[j]>=i) then f[i]:=min(f[i],f[st[j]-1]);
            inc(f[i]);
        end;
    Ending(1,f[m]);
end;

begin
    assign(input,'flow.in');reset(input);
    assign(output,'flow.out');rewrite(output);
    init;
    dp;
end.
```

总结：NOIP2010 就这么结束了，觉得要总结的很多，收获的也很多。果然觉得自己还是很沙茶，很多东西都还不会，会的很多东西也都还学的很虚。最近这段时间回去补文化课，压力也不小。无论如何，接下来都要更加努力了。