**「CODEVS1913」数字梯形问题**

给定一个由n 行数字组成的数字梯形如下图所示。梯形的第一行有m 个数字。从梯形  
的顶部的m 个数字开始，在每个数字处可以沿左下或右下方向移动，形成一条从梯形的顶  
至底的路径。  
规则1：从梯形的顶至底的m条路径互不相交。  
规则2：从梯形的顶至底的m条路径仅在数字结点处相交。  
规则3：从梯形的顶至底的m条路径允许在数字结点相交或边相交。

对于给定的数字梯形，分别按照规则1，规则2，和规则3 计算出从梯形的顶至底的m  
条路径，使这m条路径经过的数字总和最大。

第1 行中有2个正整数m和n（m,n<=20），分别  
表示数字梯形的第一行有m个数字，共有n 行。接下来的n 行是数字梯形中各行的数字。  
第1 行有m个数字，第2 行有m+1 个数字，…。

将按照规则1，规则2，和规则3 计算出的最大数字总和输出

每行一个最大总和。

2 5  
2 3  
3 4 5  
9 10 9 1  
1 1 10 1 1  
1 1 10 12 1 1

66  
75  
77

**题解**

将蚯蚓的游戏略加改动

1、与蚯蚓的游戏规则一完全相同

2、规则二，由于可以共点，所以就不用拆点了，源点与上层继续保持容量为1，并一将底层与汇点的容量限制改为无穷大即可，其它类似，

3、规则三，由于边和节点都可以共用，此时没有必要把原网络销毁，直接在上一个规则的前提下修改每一条边的容量即可（注意源点与顶层节点之间的容量还是1，否则就无法满足是m条路径取得的最大值，其他边的容量改成无穷大即可）

#include<cstdio>

#include<cstring>

#define inf 0x7fffffff

#define X 1501

#define N 3001

#define M 50001

using namespace std;

int n,m,k,ans,cnt=1,tot;

struct data{int from,to,v,c,next;}e[M];

int head[N],inq[N],from[N],dis[N],q[N],x[1001];

void ins(int u,int v,int w,int c)

{

cnt++;

e[cnt].to=v;e[cnt].from=u;

e[cnt].v=w;e[cnt].c=c;

e[cnt].next=head[u];head[u]=cnt;

}

void insert(int u,int v,int w,int c)

{ins(u,v,w,c);ins(v,u,0,-c);}

bool spfa()

{

int t=0,w=1,i,now;

memset(dis,-1,sizeof(dis));

q[0]=dis[0]=0;inq[0]=1;

while(t!=w)

    {

     now=q[t];t++;i=head[now];

     if(t==N)t=0;

     while(i)

     {

     if(e[i].v>0&&dis[now]+e[i].c>dis[e[i].to])

     {

     dis[e[i].to]=dis[now]+e[i].c;

     from[e[i].to]=i;

     if(!inq[e[i].to])

{q[w++]=e[i].to;if(w==N)w=0;inq[e[i].to]=1;}

     }

     i=e[i].next;

     }

     inq[now]=0;

    }

    if(dis[N-1]==-1)return 0;

    return 1;

}

void mcf()

{

int i,x=inf;

i=from[N-1];

while(i)

{

x=min(x,e[i].v);

i=from[e[i].from];

}

i=from[N-1];

while(i)

{

e[i].v-=x;e[i^1].v+=x;

ans+=x\*e[i].c;

i=from[e[i].from];

}

}

void rebuild()

{

    cnt=1;tot=0;

    memset(head,0,sizeof(head));

    for(int i=1;i<=n;i++)

       for(int j=1;j<=m+i-1;j++)

       {

          tot++;

          if(i<n)

          {

              insert(tot,tot+i+m,1,x[tot+i+m]);

              insert(tot,tot+i+m-1,1,x[tot+i+m-1]);

          }

      }

    for(int i=1;i<=m;i++)insert(0,i,1,x[i]);

    for(int i=1;i<=m+n-1;i++)insert(tot-i+1,N-1,inf,0);

}

void work1(){while(spfa())mcf();}

void work2()

{

    rebuild();

    while(spfa())mcf();

}

void work3()

{

    int t=cnt;

    for(int i=2;i<=t;i+=2)

        if(e[i].from!=0){e[i].v=inf;e[i^1].v=0;}

        else{e[i].v=1;e[i^1].v=0;}

    while(spfa())mcf();

}

int main()

{

scanf("%d%d",&m,&n);

for(int i=1;i<=n;i++)

       for(int j=1;j<=m+i-1;j++)

       {

          tot++;

          scanf("%d",&x[tot]);

          insert(tot,tot+X,1,x[tot]);

          if(i<n)

          {

              insert(tot+X,tot+i+m,1,0);

              insert(tot+X,tot+i+m-1,1,0);

          }

      }

    for(int i=1;i<=m;i++)insert(0,i,1,0);

    for(int i=1;i<=m+n-1;i++)insert(tot-i+X+1,N-1,1,0);

    ans=0;work1();printf("%d\n",ans);

    ans=0;work2();printf("%d\n",ans);

    ans=0;work3();printf("%d\n",ans);

return 0;

}