## 1[子串分值](https://www.luogu.com.cn/problem/P8715)

定义字符串函数f（S）为字符串S中恰好出现一次的字符的个数，对给定的字符串S求S的所有连续字串Si的f（Si）之和。

因为每个子串只有恰出现一次的字符有贡献，可以考虑每个S中每个字符的贡献，统计只包含一个该字符的子串个数即为它的贡献，记录每个字符上一次出现的位置和下一次出现的位置，两个距离相乘即为子串个数。

## 2[E - Candy Piles (atcoder.jp)](https://atcoder.jp/contests/agc002/tasks/agc002_e)

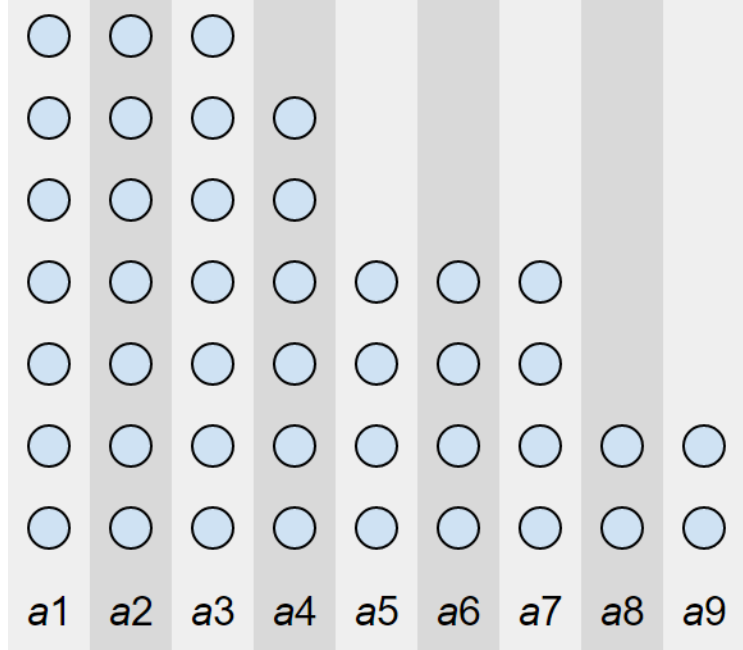
有n堆糖果，两人轮流操作，每次进行下列操作中的一个：

1将当前最大的那堆糖果吃完

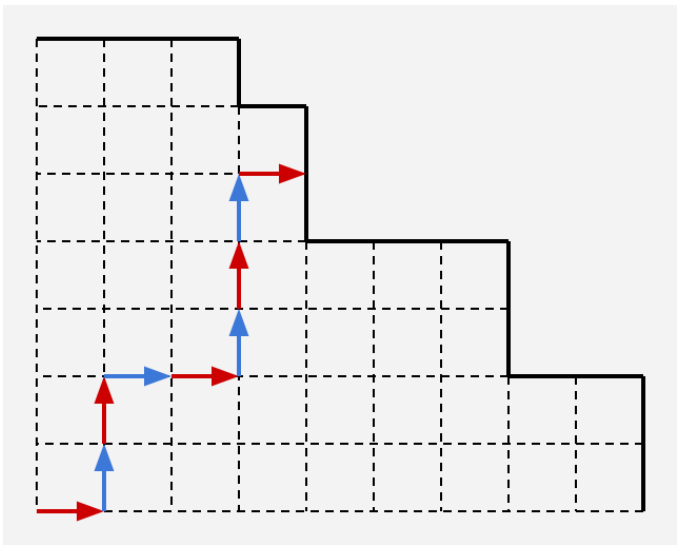
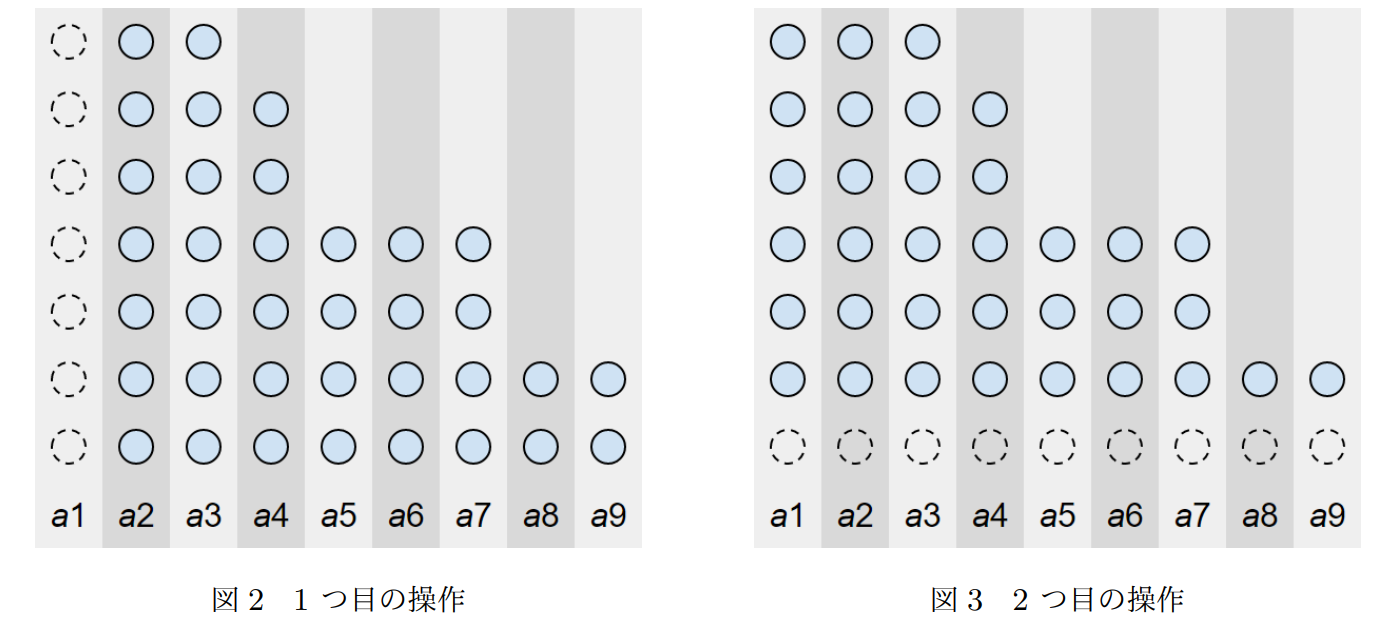
2将每堆糖果吃一个

吃完最后一个糖果的人输，问谁有必胜策略

每次进行操作2不会改变每堆糖果间的大小关系，可以将每堆糖果排序

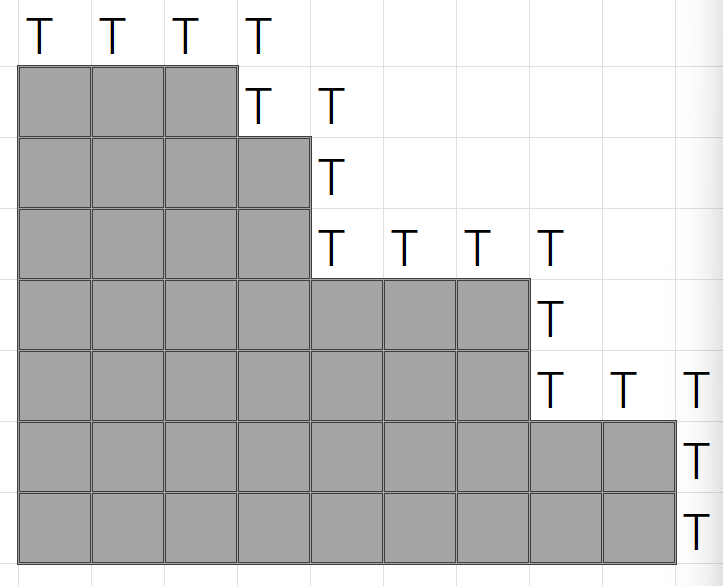


操作一相当于消去最左边一行，操作二相当于消去最下面一行

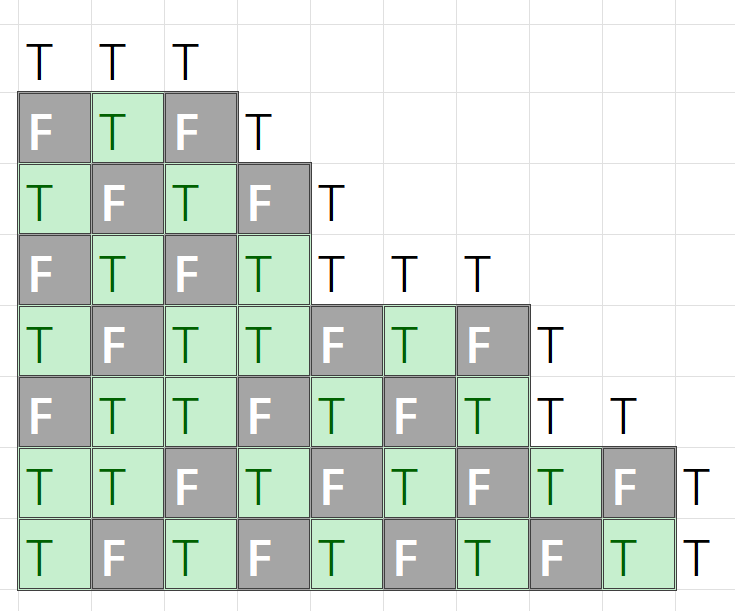


转换成一个网格图，从左下角出发，每次可以向上或向右走一格，走出边界的人输。

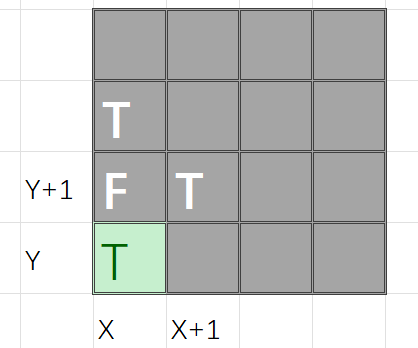
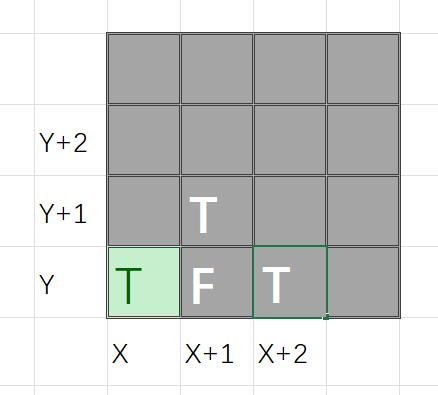
用T表示先手必胜状态，F表示先手必败状态，则网格外的点都可以看做T（上一个人走到这输）



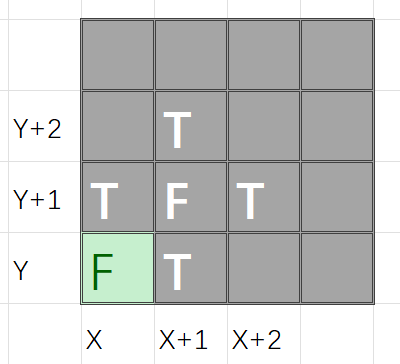
若一个点的右方和上方均为必胜态，该点为必败态，若一个点的右方和上方有一个必败态，该点为必胜态，由此可以得到所有点的状态。



发现网格内同一对角线的状态都一样，可以证明确实如此

假设(x,y)为必胜态，则它右方或上方为必败态，而必败态的右方和上方都为必胜态，则（x+1,y+1）为必胜态；



假设(x,y)为必败态，则它右方和上方为必胜态，由上可知（x+1,y+2）和（x+2,y+1）为必胜态，则（x+1,y+1）为必败态。

由此可得（1,1）的状态和（i,i）状态是一致的，求出最大的(a,a)位置，根据它向右和向上最多能走步数的奇偶性判断(a,a)的状态，即为(1,1)的状态.

## 3 [露营 (nowcoder.com)](https://ac.nowcoder.com/acm/problem/210359)

给定n\*m的矩阵，有k个位置的高度是确定的，要求给出一种填充方案使得任意相邻的两个点高度之差为1，无解则输出No。

首先有两个必有条件，任意两点的距离必须小于等于高度差，并且距离与高度差的奇偶性必须相等。当两点之间的高度差和距离差合法的时候，我们一定可以按照从低到高的方式填充。注意到每次填充都是取最低点，比如设矩阵第一行上的两个点(0,0),(0,5)高度已知为1和4，那么填充前它们之间的格子(包括自己)高度为1,x,x,x,x,4，填充以后为1,2,3,4,5,4，显然符合条件。以此可以类推多个点也可以这样填充。

#include<bits/stdc++.h>

using namespace std;

const int inf=-100000;

int dx[] = {1, 0, -1, 0};

int dy[] = {0, 1, 0, -1};

struct node{

    int h;

    int x,y;

    bool operator<(const node b)const{

        return h>b.h;

    }

};

int main()

{

    ios::sync\_with\_stdio(false);

    cin.tie(0);

    int n,m;

    cin>>n>>m;

    const int N=n;const int M=m;

    priority\_queue<node> pq;

    int high[N+5][M+5];

    int vis[N+5][M+4];

    memset(vis,0,sizeof(vis));

    int k;cin>>k;int flag=1;

    for(int i=0;i<k;i++)

    {

        int x,y,h;

        cin>>x>>y>>h;

        pq.push({h,x,y});

        high[x][y]=h;

        vis[x][y]=1;

    }

    if(k==0)pq.push({1,1,0});

    while(!pq.empty())

    {

        node s=pq.top();pq.pop();

        int h=s.h;

        for(int i=0;i<=3;i++)

         {

            int tx=s.x+dx[i];

            int ty=s.y+dy[i];

            if(tx>n||tx<1||ty>m||ty<1||vis[tx][ty]) continue;

                high[tx][ty]=h+1;

                pq.push({h+1,tx,ty});

                vis[tx][ty]=1;

         }

    }

    for(int i=1;i<=n;i++)

     for(int j=1;j<=m;j++)

     {

        if(i+1<=n)

        if(abs(high[i][j]-high[i+1][j])!=1)

        flag=0;

        if(j+1<=m)

        if(abs(high[i][j]-high[i][j+1])!=1)

        flag=0;

     }

    if(flag)

    {

    cout<<"Yes"<<endl;

    for(int i=1;i<=n;i++){

     for(int j=1;j<=m;j++)cout<<high[i][j]<<" ";

     cout<<endl;

     }

    }

    else

    cout<<"No"<<endl;

    return 0;

}