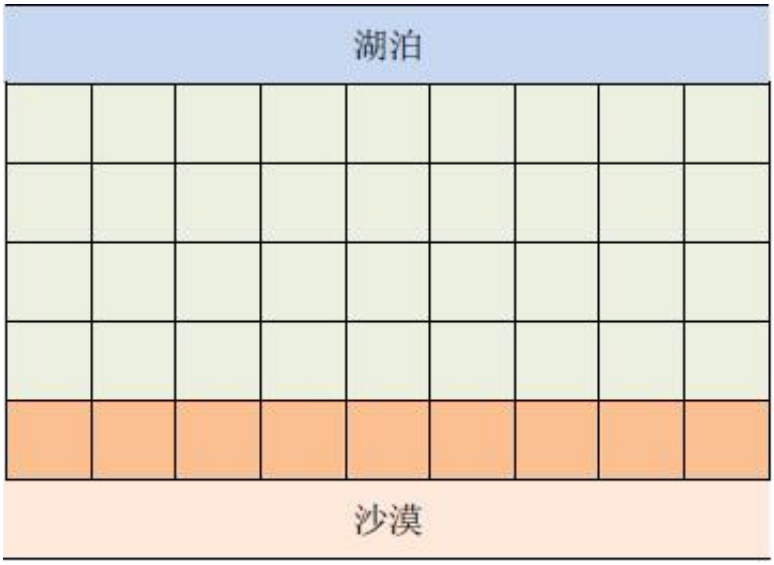
用时: 120ms / 内存: 6246KB

题意

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。

分析

- 如果第一行中，某一个点比它左右的点小，那么它可以由左右流过来，也就是没必要在这个点上建造蓄水场
  所以只选择无法从左右流过来的点作为起始点，dfs求得能覆盖第n行的哪几个点
- 路线如果交叉，那么dfs交叉点开始后面的情况完全相同，可以判断如果[1][x] 能到的点如果不是连续的，那么中间未流到的点也绝对不会被其他点流到，所以可以得出结论，一个蓄水点所能覆盖的点是个连续的区间
- 将右区间+1，问题转化成区间完全覆盖问题，可以用贪心解得：

a. 将每一个区间按照左端点递增顺序排列
b. 设置一个变量表示已经覆盖到的区域。再剩下的线段中找出所有左端点小于等于当前
c. 已经覆盖到的区域的右端点的线段中，右端点最大的线段在加入，直到已经覆盖全部的区域

代码

```
#include<bits/stdc++.h>
#define For(i,a,b) for(int i=(a); i<=(b) ; i++)
#define _For(i,a,b) for(int i=(a); i>=(b) ; i--)
#define Memset(a,b); memset((a),(b),sizeof((a)));
#define Cout(a,b);  printf("%d",(a));printf(b);
#define Coutc(a,b);  printf("%c",(a));printf(b);
#define Couts(a,b);  printf("%s",(a));printf(b);
using namespace std;
const int INF = 0x3f3f3f3f;
typedef  long long LL;typedef  unsigned long long ULL;typedef  long double LDB;
inline LL CinLL(){LL x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')f=-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return x*f;}
inline int Cin(){int x=0,f=1;char ch=getchar();while(!isdigit(ch)){if(ch=='-')f=-1;ch=getchar();}while(isdigit(ch))x=x*10+ch-'0',ch=getchar();return f*x;}
int n,m;
int a[505][505];
int book[505][505];
struct Line{
    int l,r;
}line[505];
int ok[505];
int sx[] = {0,0,0,1,-1};
int sy[] = {0,-1,1,0,0};
int tot = 0;
bool cmp(Line x,Line y)
{
    if(x.l == y.l) return x.r < y.r;
    else return x.l < y.l;
}
void dfs(int nowx,int nowy)
{
    if(nowx == n)
```

```
    {
        if(line[tot].l > nowy) line[tot].l = nowy;
        if(line[tot].r < nowy) line[tot].r = nowy;
        ok[nowy] = 1;
    }
    For(i,1,4)
    {
        int nexx = nowx + sx[i],nexy = nowy + sy[i];
        if(book[nexx][nexy] == 0 && a[nowx][nowy] > a[nexx][nexy] &&nexx <= n && nexx >0 && nexy <=m && nexy > 0 )
        {
            book[nexx][nexy] = 1;
            dfs(nexx,nexy);
        }
    }
}
int main()
{
    ios::sync_with_stdio(false);
    cin>>n>>m;
    For(i,1,n) For(j,1,m) cin>>a[i][j];
    For(i,1,m)
    {
        if(a[1][i] >= a[1][i-1] && a[1][i] >= a[1][i+1])
        {
            ++tot;
            line[tot].l = m+1;
            line[tot].r = 0;
            Memset(book,0);
            book[1][i] = 1;
            dfs(1,i);
            int ll = line[tot].l,rr = line[tot].r;//cout<<ll<<" "<<rr<<endl;
    //        cout<<line[tot].l<<" "<<line[tot].r<<endl;
            while(ll > 1)
            {
                if(a[n][ll] > a[n][ll-1] ) ll--;
                else break;
            }
            while(rr < m-1)
            {
                if(a[n][rr] > a[n][rr+1]) rr++;
                else break;
            }
            line[tot].l  =ll;line[tot].r = rr+1;
      //    cout<<ll<<" "<<rr+1<<endl;
```

```
75
76          }
77      }
78      int o = 1;
79      For(i,1,m) o = o &(ok[i]);
80      if(!o){
81          For(i,1,m) if(!ok[i]) o ++;
82          cout<<0<<endl;
83          cout<<o<<endl;
84          return 0;
85      }
86      sort(line+1,line+tot+1,cmp);
87      int count = 0;
88      int s, e = 1;
89      int index = 1;
90      int ok = 1;
91      while(e <= m)
92      {
93          s = e;//更新覆盖区域
94          for(int i=index; i<=tot; i++)
95          {
96              if(line[i].l <= s)
97              {
98                  if(line[i].r >= s)
99                  {
100                     e = line[i].r;//取符合条件的最远区间。
101                 }
102             }
103             else{
104                 index = i;//不符合条件则需要换区间
105                 break;
106             }
107         }
108         count ++;
109     }
110     cout<<1<<endl;
111     cout<<count<<endl;
112 }
113
```