1280：滑雪

题目描述

小明喜欢滑雪，因为滑雪的确很刺激，可是为了获得速度，滑的区域必须向下倾斜，当小明滑到坡底，不得不再次走上坡或等着直升机来载他，小明想知道在一个区域中最长的滑坡。滑坡的长度由滑过点的个数来计算，区域由一个二维数组给出，数组的每个数字代表点的高度。下面是一个例子：

1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

一个人可以从某个点滑向上下左右相邻四个点之一，当且仅当高度减小，在上面的例子中，一条可行的滑坡为25-24-17-16-1（从25开始到1结束），当然25-24……2-1更长，事实上这是最长的一条。

输入

输入的第一行为表示区域的二维数组的行数R和列数C（1≤R、C≤100），下面是R行，每行有C个数代表高度。

输出

输出区域中最长的滑坡长度。

输入样例

```
5 5
1 2 3 4 5
16 17 18 19 6
15 24 25 20 7
14 23 22 21 8
13 12 11 10 9
```

输出样例

```
25
```

解析

基本的DFS，注意缓存数据即可。

编码

```
#include <bits/stdc++.h>

using namespace std;
#define M 101   //行列最大值为100;
int f[M][M];  //在i,j点的最大滑行距离
int r, c, mapInfo[M][M], ans;

//四方向遍历：下上右左
int dx[] = {0, 0, 1, -1};
int dy[] = {1, -1, 0, 0};

//深搜
int dfs(int x, int y) {
    //如果已经计算过了，就直接返回。缓存机制：
    if (f[x][y]) {
        return f[x][y];
    }
    //起始长度为1
    int maxx = 1;
    int step = 0;
    for (int i = 0; i < 4; i++) {
        int nx = x + dx[i];
        int ny = y + dy[i];
        step = 0;
        //边界判断
       if (nx <= 0 || nx > r || ny <= 0 || ny > c) {
            continue;
        }
        //新的节点比当前的要小
       if (mapInfo[nx][ny] < mapInfo[x][y]) {
            //继续向下搜索
          step = dfs(nx, ny) + 1;
        }
        //记录以当前节点作为开始的最长路线
       maxx = max(maxx, step);
    }
    f[x][y] = maxx;
    return f[x][y];
}

int main() {
    scanf("%d%d", &r, &c); //读入行数和列数
   //读入地图数据
   for (int i = 1; i <= r; i++) {
        for (int j = 1; j <= c; j++) {
            scanf("%d", &mapInfo[i][j]);
```

```
            }
        }
        //遍历每一个节点进行神搜
        for (int i = 1; i <= r; i++) {
            for (int j = 1; j <= c; j++) {
                //计算以当前节点作为起始的滑雪最大长度
                f[i][j] = dfs(i, j);
                ans = max(ans, f[i][j]);

            }
        }
        printf("%d\n", ans);
        return 0;
}
```