{% note info %} **摘要** Title: 901. 滑雪 Tag: 记忆化搜索、dp Memory Limit: 64 MB Time Limit: 1000 ms {% endnote %}

Powered by:**NEFU AB-IN**

Link

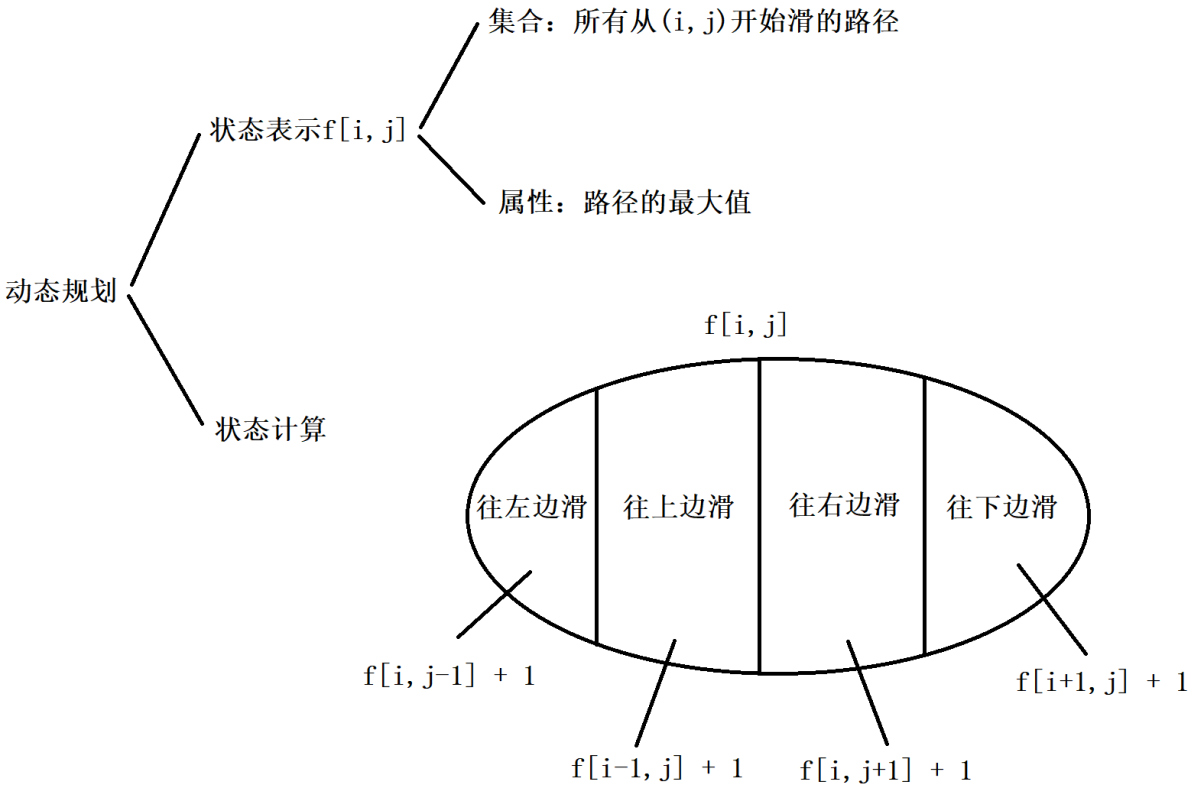
@TOC

901. 滑雪

- 题意

> 给定一个 R 行 C 列的矩阵，表示一个矩形网格滑雪场。
>
> 矩阵中第 i 行第 j 列的点表示滑雪场的第 i 行第 j 列区域的高度。
>
> 一个人从滑雪场中的某个区域内出发，每次可以向上下左右任意一个方向滑动一个单位距离。
>
> 当然，一个人能够滑动到某相邻区域的前提是该区域的高度低于自己目前所在区域的高度。
>
> 下面给出一个矩阵作为例子：
>
> ```
> 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
> ```
>
> 在给定矩阵中，一条可行的滑行轨迹为 $24 - 17 - 2 - 1$。
>
> 在给定矩阵中，最长的滑行轨迹为 $25 - 24 - 23 - \ldots - 3 - 2 - 1$，沿途共经过 25 个区域。
>
> 现在给定你一个二维矩阵表示滑雪场各区域的高度，请你找出在该滑雪场中能够完成的最长滑雪轨迹，并输出其长度(可经过最大区域数)。

- 思路



其实就是dfs+记忆化，每次爆搜的时候，能利用之前的状态

- 代码

```
'''
Author: NEFU AB-IN
Date: 2022-03-14 09:29:17
FilePath: \ACM\Acwing\901.py
LastEditTime: 2022-03-14 17:06:34
'''
N = 320
f = [[-1] * N for _ in range(N)] #先把每个状态初始化为-1，表示状态未被算过
g = [[0] * N for _ in range(N)]


def dp(x, y):
    if f[x][y] != -1:
        return f[x][y]

    f[x][y] = 1 # 最少就只走当前的格子

    for i in range(4):
        a = x + dx[i]
        b = y + dy[i]
        if a >= 1 and a <= r and b >= 1 and b <= c and g[x][y] > g[a][b]:
            f[x][y] = max(f[x][y], dp(a, b) + 1)
    return f[x][y]
```

```
r, c = map(int, input().split())

for i in range(1, r + 1):
    g[i][1:] = list(map(int, input().split()))

dx = [-1, 0, 1, 0]
dy = [0, 1, 0, -1]
res = 1

for i in range(1, r + 1):
    for j in range(1, c + 1):
        res = max(res, dp(i, j))

print(res)
```