# 문제 1. 많이 이동하기 ( #1937 )

모든 좌표에서 출발이 가능합니다.

아래와 같은 조건이 주어졌을 때, 가장 많이 이동한 경우 몇 회 이동했는지 계산하는 프로그램을 작성하세요.

**[조건]**

* 이동은 반드시, 현재 좌표의 값보다 더 큰 곳으로만 이동이 가능합니다.

|  |  |  |  |
| --- | --- | --- | --- |
| 14 | 9 | 12 | 10 |
| 1 | 11 | 5 | 4 |
| 7 | 15 | 2 | 13 |
| 6 | 3 | 16 | 8 |

### **입력**

**4**

**14 9 12 10**

**1 11 5 4**

**7 15 2 13**

**6 3 16 8**

### **출력**

**4**

# 문제 2. 끝까지 이동하기 ( #1520 )

좌측 위 좌표에서 출발합니다.

아래와 같은 조건이 주어졌을 때, 우측 하단까지 도달 할 수 있는 경로의 수를 구하세요.

**[조건]**

* 이동은 반드시, 현재 좌표의 값보다 더 작은 곳으로만 이동이 가능합니다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 50 | 45 | 37 | 32 | 30 |
| 35 | 50 | 40 | 20 | 25 |
| 30 | 30 | 25 | 17 | 28 |
| 27 | 24 | 22 | 15 | 10 |

### **입력**

**4 5**

**50 45 37 32 30**

**35 50 40 20 25**

**30 30 25 17 28**

**27 24 22 15 10**

### **출력**

**3**

**정답 코드**

**#1937**

**def** recur(y,x):

**if** dp[y][x] != 0:

**return** dp[y][x]

**for** dy, dx **in** [[1,0],[-1,0],[0,1],[0,-1]]:

ey = y + dy

ex = x + dx

**if** 0 <= ey < n **and** 0 <= ex < n:

**if** graph[y][x] < graph[ey][ex]:

dp[y][x] = max(dp[y][x], recur(ey,ex) + 1)

**return** dp[y][x]

*# return max(recur(y-1,x), recur(y+1,x), recur(y,x+1),recur(y,x-1))*

n = int(input()) *# 그래프의 크기*

graph = [list(map(int,input().split())) **for** \_ **in** range(n)]

dp = [[0 **for** \_ **in** range(n)]**for** \_ **in** range(n)]

*# 모든 점을 방문한다!*

*# 방문한 뒤에 이동할 수 있는 모든 경우의 수를 재귀로 구현한다!*

*# 재귀로 구현한 뒤 DP로 바꾼다!*

**for** y **in** range(n):

**for** x **in** range(n):

recur(y,x)

*# print(dp)*

print(max(map(max, dp)) + 1)

**#1520**

**def** recur(y,x):

**if** y == Y-1 **and** x == X-1:

**return** 1

**if** dp[y][x] != -1: *# 모든 지점에서 계산이 한번이라도 된다면, 재사용!*

**return** dp[y][x]

route = 0

**for** dy, dx **in** [[1,0],[-1,0],[0,1],[0,-1]]:

ey = y + dy

ex = x + dx

**if** 0 <= ey < Y **and** 0 <= ex < X:

**if** graph[y][x] > graph[ey][ex]:

route += recur(ey,ex)

dp[y][x] = route

**return** dp[y][x]

Y , X = map(int,input().split())

graph = [list(map(int,input().split())) **for** \_ **in** range(Y)]

dp = [[-1 **for** \_ **in** range(X)] **for** \_ **in** range(Y)]

answer = recur(0,0)

print(answer)