현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1, -1]
        [-1, -1, -1, -1, -1]
        [-1, -1, -1, -1, -1]
        [-1, -1, -1, -1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 0

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1, -1]
        [-1, -1, -1, -1, -1]
        [-1, -1, -1, -1, -1]
        [-1, -1, -1, -1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 0

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1, -1]
        [-1, -1, -1, -1, -1]
        [-1, -1, -1, -1, -1]
        [-1, -1, -1, -1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 1

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1, -1]
        [-1, -1, -1, -1, -1]
        [-1, -1, -1, -1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 1

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1, -1]
        [-1, -1, -1, -1, -1]
        [-1, -1, -1, -1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 0

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1, -1]
        [-1, -1, -1, -1, -1]
        [-1, -1, -1, -1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 1

현재 위치

현재 코드

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1, -1]
        [-1, -1, -1, -1, -1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 1

현재 위치

현재 코드

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1, -1]
        [-1, -1, -1, -1, -1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 0

현재 위치

현재 코드

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1, -1]
        [-1, -1, -1, -1, -1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 1

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1, -1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 1

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1, -1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 0

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1, -1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 0

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1, -1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 1

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1, -1]
        [-1, -1, -1,  1,  1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 1

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1, -1]
        [-1, -1, -1,  1,  1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 0

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1, -1]
        [-1, -1, -1,  1,  1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 1

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1,  1]
        [-1, -1, -1,  1,  1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 1

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1,  1]
        [-1, -1, -1,  1,  1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 0

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1,  1]
        [-1, -1, -1,  1,  1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 1

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1, -1,  1]
        [-1, -1, -1,  1,  1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 1+1

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1,  2,  1]
        [-1, -1, -1,  1,  1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 2

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1,  2,  1]
        [-1, -1, -1,  1,  1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 0

현재 위치 (i, j)

```
dp = [
        [-1, -1, -1,  2,  1]
        [-1, -1, -1,  1,  1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 2

현재 위치 (i, j)

```
dp = [
        [-1, -1,  2,  2,  1]
        [-1, -1, -1,  1,  1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 2

현재 위치 (i, j)

```
dp = [
        [-1, -1,  2,  2,  1]
        [-1, -1, -1,  1,  1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 0

현재 위치 (i, j)

```
dp = [
        [-1, -1,  2,  2,  1]
        [-1, -1, -1,  1,  1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 2

현재 위치 (i, j)

```
dp = [
        [-1,  2,  2,  2,  1]
        [-1, -1, -1,  1,  1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 2

현재 위치 (i, j)

```
dp = [
        [-1,  2,  2,  2,  1]
        [-1, -1, -1,  1,  1]
        [-1, -1, -1,  1, -1]
        [-1, -1, -1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 2

현재 위치 (i, j)

```
dp = [
        [-1,  2,  2,  2,  1]
        [ 1, -1, -1,  1,  1]
        [ 1, -1, -1,  1, -1]
        [ 1,  1,  1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

현재 위치 (i, j)

```
dp = [
        [-1,  2,  2,  2,  1]
        [ 1, -1, -1,  1,  1]
        [ 1, -1, -1,  1, -1]
        [ 1,  1,  1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 2

현재 위치 (i, j)

```
dp = [
        [-1,  2,  2,  2,  1]
        [ 1, -1, -1,  1,  1]
        [ 1, -1, -1,  1, -1]
        [ 1,  1,  1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 2+1

현재 위치 (i, j)

```
dp = [
        [ 3,  2,  2,  2,  1]
        [ 1, -1, -1,  1,  1]
        [ 1, -1, -1,  1, -1]
        [ 1,  1,  1,  1, -1]
    ]
```

path_cnt 구하는 위치

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

```python
def dfs(i, j):                  # 현재 위치
    if i == M-1 and j == N-1 :  # 목적지에 도착하면, 1 반환
        return 1

    if dp[i][j] != -1:          # 방문한 경로라면, 기록된 경로 수 반환
        return dp[i][j]

    path_cnt = 0                # 경로 수 초기화
    for k in range(4):
        ni = i + di[k]
        nj = j + dj[k]
        if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][nj]:  # 범위 내에서 더 작은 경로만 통과
            continue
        path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가

    dp[i][j] = path_cnt         # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입

    return dp[i][j]             # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 반환

M, N = map(int, input().split())      # M: row 수 / N: column 수
arr = [list(map(int, input().split())) for _ in range(M)]
di = [0, 1, -1, 0]                    # right / down / up / left
dj = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)]     # 초기 경로 값 -1

print(dfs(0, 0))                      # (0,0)에서 시작
```

path_cnt = 3