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

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                       # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][i]
                       # 경로 수 초기화
  path cnt = 0
  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
     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][i]
                       # 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]
di = [1, 0, 0, -1]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

50—	45	37	32	<b>30</b>
35	50	40	20 🕶	- 25
30	30	25	17	28
27	24	22	15	10

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][i]
                       # 경로 수 초기화
  path cnt = 0
  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
     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][i]
                      # 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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

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

```
def dfs(i, i):
                     # 현재 위치
  if i == M-1 and j == N-1: # 목적지에 도착하면. 1 반환
  if dp[i][i] != -1:
                       # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][i]
                      # 경로 수 초기화
  path cnt = 0
  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][ni]: # 범위 내에서 더 작은 경로만 통과
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
  dp[i][j] = path_cnt
                      # 현재 위치 (i, i) 에서 갈 수 있는 경로 수 경로 수 삽입
  return dp[i][i]
                      # 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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

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

```
def dfs(i, j):
                    # 현재 위치
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][j]
                      # 경로 수 초기화
  path cnt = 0
  for k in range(4):
    ni = i + di[k]
    ni = i + di[k]
    if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][ni]: # 범위 내에서 더 작은 경로만 통과
    path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
  dp[i][i] = path_cnt
                     # 현재 위치 (i. i) 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)]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

50—	45	37	32	<b>30</b>
35	50	40	20 ←	- 25
30	30	25	17	28
27	24	22	15	10

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][j] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][i]
                       # 경로 수 초기화
  path cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
  dp[i][j] = path_cnt
                      # 현재 위치 (i, i) 에서 갈 수 있는 경로 수 경로 수 삽입
  return dp[i][i]
                      # 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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

50—	45	37	32	<b>30</b>
35	50	40	20 🕶	- 25
30	30	25	17	28
27	24	22	15	10

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                       # 방문한 경로라면, 기록된 경로 수 반환
     return dp[i][j]
                       # 경로 수 초기화
  path_cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
                      # 현재 위치 (i, i) 에서 갈 수 있는 경로 수 경로 수 삽입
  dp[i][j] = path_cnt
  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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

## 현재 위치

# 현재 코드

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

```
def dfs(i, j):
                    # 현재 위치
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][j] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][j]
                     # 경로 수 초기화
  path cnt = 0
  for k in range(4):
    ni = i + di[k]
    ni = i + di[k]
    if ni<0 or nj<0 or ni>=M or nj>=N or arr[i][j] <= arr[ni][ni]: # 범위 내에서 더 작은 경로만 통과
    path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
  dp[i][i] = path_cnt
                     # 현재 위치 (i, i) 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)]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                         # (0,0)에서 시작
```

## 현재 위치

# 현재 코드

50—	45	37	32	<b>30</b>
35	50	40	20 🕶	- 25
30	30	25	17	28
27	24	22	15	10

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][j] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][j]
                      # 경로 수 초기화
  path cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
  dp[i][i] = path_cnt
                      # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입
  return dp[i][i]
                      # 현재 위치 (i, i) 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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

## 현재 위치

# 현재 코드

```
현재 위치 (i, j)

dp = [

[-1, -1, -1, ※, -1]

[-1, -1, -1, 1]

[-1, -1, -1, 1, -1]

]

path_cnt 구하는 위치
```

50—	45	37	32	<b>30</b>
35	50	40	20 🕶	- 25
30	30	25	17	28
27	24	22	15	10

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][j] != −1:
                       # 방문한 경로라면, 기록된 경로 수 반환
     return dp[i][j]
                       # 경로 수 초기화
  path_cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
                      # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입
  dp[i][i] = path_cnt
  return dp[i][i]
                      # 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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

50—	45	37	32	<b>30</b>
35	50	40	20 🕶	- 25
30	30	25	17	28
27	24	22	15	10

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][i]
                      # 경로 수 초기화
  path_cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
                      # 현재 위치 (i, i) 에서 갈 수 있는 경로 수 경로 수 삽입
  dp[i][j] = path_cnt
  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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

50—	45	37	32	<b>30</b>
35	50	40	20	25
30	30	25	17	28
27	24	22	15	10

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][i]
                       # 경로 수 초기화
  path cnt = 0
  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
    path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
  dp[i][j] = path_cnt
                      # 현재 위치 (i, i) 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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

50—	45	37	32	<b>30</b>
35	50	40	20	25
30	30	25	17	28
27	24	22	15	10

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][i]
                       # 경로 수 초기화
  path cnt = 0
  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
    path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
  dp[i][j] = path_cnt
                      # 현재 위치 (i, i) 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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

50—	45	37	32	<b>30</b>
35	50	40	20	25
30	30	25	17	28
27	24	22	15	10

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                       # 방문한 경로라면, 기록된 경로 수 반환
     return dp[i][j]
                       # 경로 수 초기화
  path_cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
                      # 현재 위치 (i, i) 에서 갈 수 있는 경로 수 경로 수 삽입
  dp[i][j] = path_cnt
  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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

50—	45	37	32	<b>30</b>
35	50	40	20	25
30	30	25	17	28
27	24	22	15	10

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][i]
                      # 경로 수 초기화
  path_cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
  dp[i][j] = path_cnt
                      # 현재 위치 (i, i) 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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

50—	45	37	32	<b>30</b>
35	50	40	20	25
30	30	25	17	28
27	24	22	15	10

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][i]
                       # 경로 수 초기화
  path cnt = 0
  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
    path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
  dp[i][j] = path_cnt
                      # 현재 위치 (i, i) 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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

50—	45	37	32	<b>30</b>
35	50	40	20	25
30	30	25	17	28
27	24	22	15	10

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                       # 방문한 경로라면, 기록된 경로 수 반환
     return dp[i][j]
                       # 경로 수 초기화
  path_cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
                      # 현재 위치 (i, i) 에서 갈 수 있는 경로 수 경로 수 삽입
  dp[i][j] = path_cnt
  return dp[i][j]
                      # 현재 위치 (i, i) 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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

50—	45	37	32	<b>30</b>
35	50	40	20	25
30	30	25	17	28
27	24	22	15	10

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][i]
                      # 경로 수 초기화
  path_cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
  dp[i][j] = path_cnt
                      # 현재 위치 (i, i) 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]
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, -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

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][j] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][i]
                       # 경로 수 초기화
  path cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
  dp[i][j] = path_cnt
                      # 현재 위치 (i, i) 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]
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, -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

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                       # 방문한 경로라면, 기록된 경로 수 반환
     return dp[i][j]
                       # 경로 수 초기화
  path_cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
                      # 현재 위치 (i, i) 에서 갈 수 있는 경로 수 경로 수 삽입
  dp[i][j] = path_cnt
  return dp[i][j]
                      # 현재 위치 (i, i) 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]
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, -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

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                       # 방문한 경로라면, 기록된 경로 수 반환
     return dp[i][j]
                       # 경로 수 초기화
  path_cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
                      # 현재 위치 (i, i) 에서 갈 수 있는 경로 수 경로 수 삽입
  dp[i][j] = path_cnt
  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]
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, -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

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][i]
                      # 경로 수 초기화
  path_cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
  dp[i][j] = path_cnt
                      # 현재 위치 (i, i) 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]
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, ※, -1, ② 1]

        [-1, -1, -1, 1, 1]

        [-1, -1, -1, 1, -1]

        [-1, -1, -1, 1, -1]

        ]

path_cnt 구하는 위치
```

50—	45	<b>3</b> 7	32	30
35	50	40	20	25
30	30	25	17	28
27	24	22	15	10

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and j == N-1: # 목적지에 도착하면. 1 반환
  if dp[i][j] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][i]
                      # 경로 수 초기화
  path cnt = 0
  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
     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]
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, ¾, -1, ② 1]

[-1, -1, ¾, 1, 1]

[-1, -1, -1, 1, -1]

[-1, -1, -1, 1, -1]

]
path_cnt 구하는 위치
```

50—	45	<b>37</b>	32	30
35	50	40	20	25
30	30	25	17	28
27	24	22	15	10

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                       # 방문한 경로라면, 기록된 경로 수 반환
     return dp[i][j]
                       # 경로 수 초기화
  path_cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
                      # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입
  dp[i][j] = path_cnt
  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]
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, ¾, 2, ② 1]

[-1, -1, -1, 1, 1]

[-1, -1, -1, 1, -1]

[-1, -1, -1, 1, -1]

]
path_cnt 구하는 위치
```

50—	45	<b>3</b> 7	32	30
35	50	40	20	25
30	30	25	17	28
27	24	22	15	10

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][i]
                      # 경로 수 초기화
  path_cnt = 0
  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
     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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

```
현재 위치 (i, j)

dp = [

[¾, -1, ② 2, 1]

[-1, ¾, -1, 1, 1]

[-1, -1, -1, 1, -1]

[-1, -1, -1, 1, -1]

]
path_cnt 구하는 위치
```

50—	<b>→</b> 45	37	32	30
35	50	40	20	25
30	30	25	17	28
27	24	22	15	10

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and j == N-1: # 목적지에 도착하면. 1 반환
  if dp[i][j] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][i]
                      # 경로 수 초기화
  path cnt = 0
  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
     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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

```
현재 위치 (i, j)

dp = [

[¾, -1, ② 2, 1]

[-1, ¾, -1, 1, 1]

[-1, -1, -1, 1, -1]

[-1, -1, -1, 1, -1]

]
path_cnt 구하는 위치
```

50—	<b>→</b> 45	37	32	30
35	50	40	20	25
30	30	25	17	28
27	24	22	15	10

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                       # 방문한 경로라면, 기록된 경로 수 반환
     return dp[i][j]
                       # 경로 수 초기화
  path_cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
                      # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 경로 수 삽입
  dp[i][j] = path_cnt
  return dp[i][j]
                      # 현재 위치 (i, i) 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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

```
현재 위치 (i, j)

dp = [

[¾, 2, ② 2, 1]

[-1, ¾, -1, 1, 1]

[-1, -1, -1, 1, -1]

[-1, -1, -1, 1, -1]

]
path_cnt 구하는 위치
```

50—	<b>+</b> 45	37	32	30
35	50	40	20	25
30	30	25	17	28
27	24	22	15	10

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][i]
                      # 경로 수 초기화
  path_cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
  dp[i][j] = path_cnt
                      # 현재 위치 (i, i) 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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

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

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][i]
                      # 경로 수 초기화
  path_cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
  dp[i][j] = path_cnt
                      # 현재 위치 (i, i) 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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

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

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][j] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][j]
                      # 경로 수 초기화
  path_cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
  dp[i][j] = path_cnt
                      # 현재 위치 (i, i) 에서 갈 수 있는 경로 수 경로 수 삽입
  return dp[i][j]
                      # 현재 위치 (i, i) 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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

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

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                       # 방문한 경로라면, 기록된 경로 수 반환
     return dp[i][j]
                       # 경로 수 초기화
  path_cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
                      # 현재 위치 (i, i) 에서 갈 수 있는 경로 수 경로 수 삽입
  dp[i][j] = path_cnt
  return dp[i][j]
                      # 현재 위치 (i, i) 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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

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

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                       # 방문한 경로라면, 기록된 경로 수 반환
     return dp[i][j]
                       # 경로 수 초기화
  path_cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
                      # 현재 위치 (i, i) 에서 갈 수 있는 경로 수 경로 수 삽입
  dp[i][j] = path_cnt
  return dp[i][j]
                      # 현재 위치 (i, i) 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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```

```
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

```
# 현재 위치
def dfs(i, j):
  if i == M-1 and i == N-1: # 목적지에 도착하면, 1 반환
  if dp[i][i] != -1:
                      # 방문한 경로라면. 기록된 경로 수 반환
     return dp[i][i]
                      # 경로 수 초기화
  path_cnt = 0
  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
     path_cnt += dfs(ni, nj) # 현재 위치 (i, j) 에서 갈 수 있는 경로 수 추가
  dp[i][j] = path_cnt
                      # 현재 위치 (i, i) 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]
dp = [[-1] * N for _ in range(M)] # 초기 경로 값 -1
print(dfs(0, 0))
                          # (0,0)에서 시작
```