고 급 문 제 해 결

<문제 16.3>

2차원 배열 순회 방법 개수

2018117543 한재성

Chapter 9

Dynamic Programming

16.3 - 2차원 배열 순회 방법 개수

There is a robot on an `m x n` grid. The robot is initially located at the **top-left corner** (i.e., `grid[0][0]`). The robot tries to move to the **bottom-right corner** (i.e., `grid[m - 1][n - 1]`). The robot can only move either down or right at any point in time.

Given the two integers `m` and `n`, return *the number of possible unique paths that the robot can take to reach the bottom-right corner.*

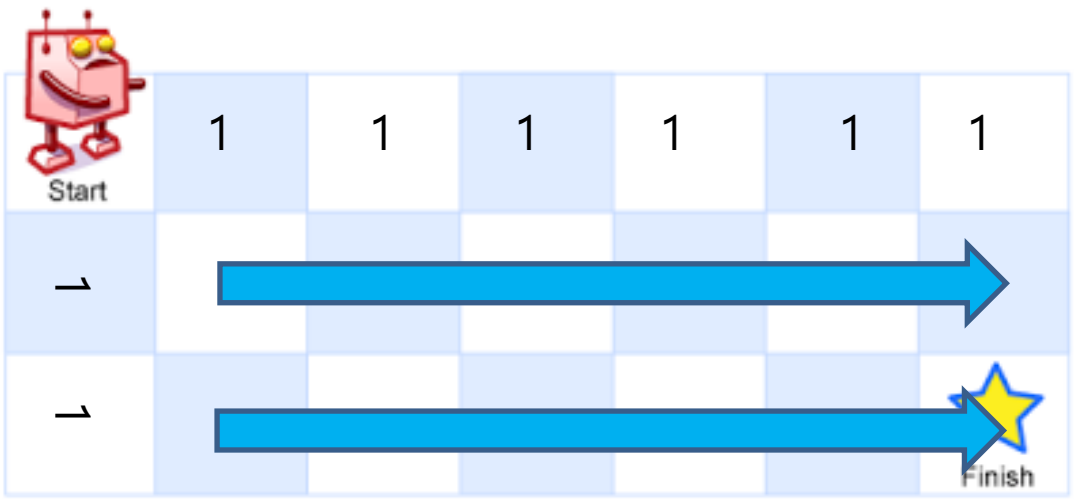
The test cases are generated so that the answer will be less than or equal to $2 * 10^9$.

Example 1:


```
Input: m = 3, n = 7
Output: 28
```

Idea

Example 1:



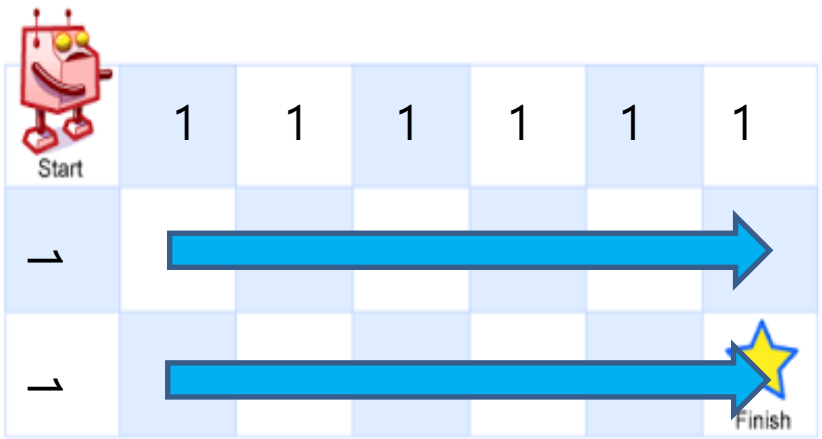

```
Input: m = 3, n = 7
Output: 28
```

Code

```
def uniquePaths(self, m: int, n: int) -> int:
    M = [[0] * n for _ in range(m)]
    for i in range(m):
        M[i][0] = 1
    for i in range(1, n):
        M[0][i] = 1
    for i in range(1, m):
        for j in range(1, n):
            M[i][j] += M[i-1][j] + M[i][j-1]
    return M[m-1][n-1]
```

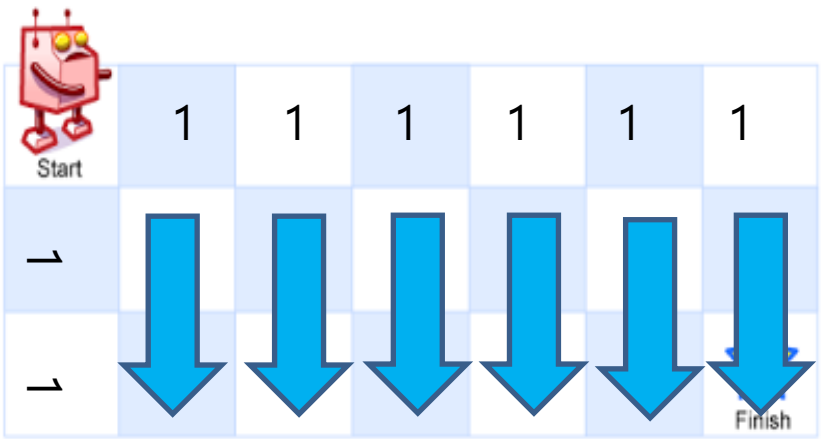
응용 1. 공간 복잡도 줄이기

Example 1:




```
Input: m = 3, n = 7
Output: 28
```

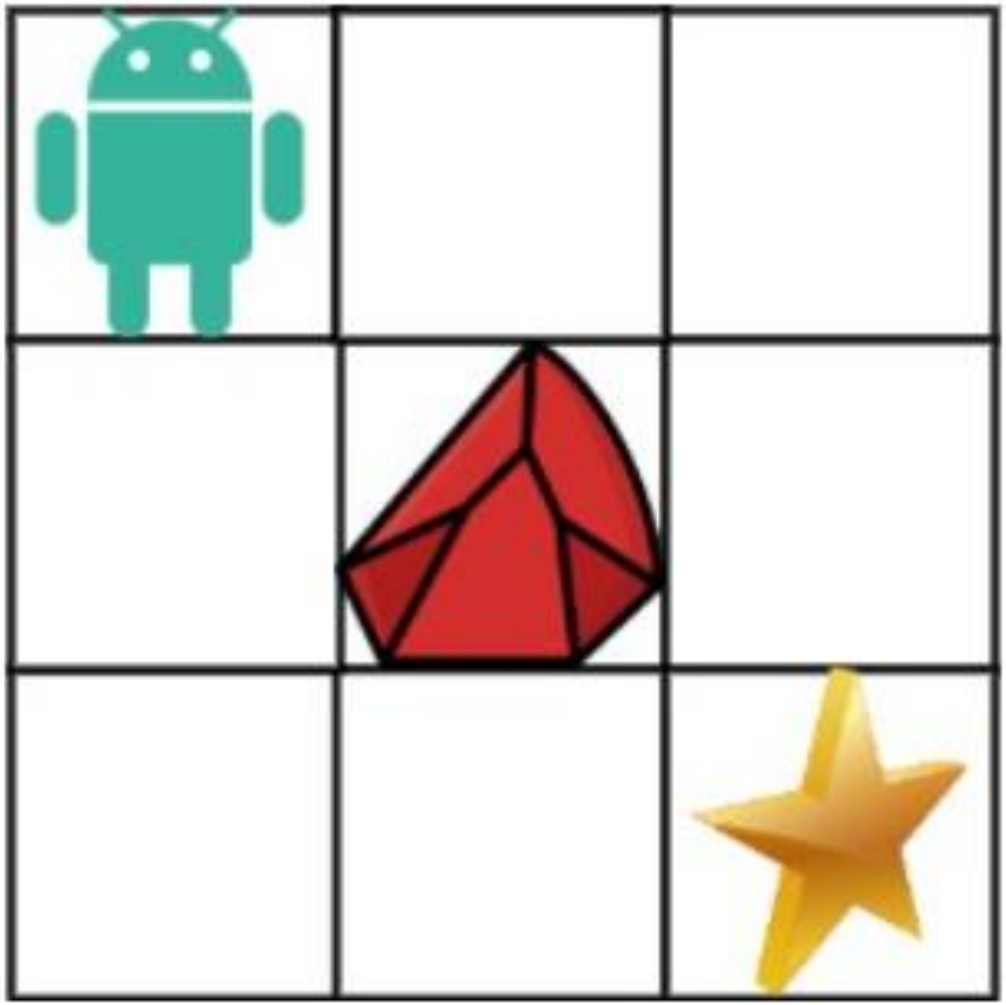
Example 1:



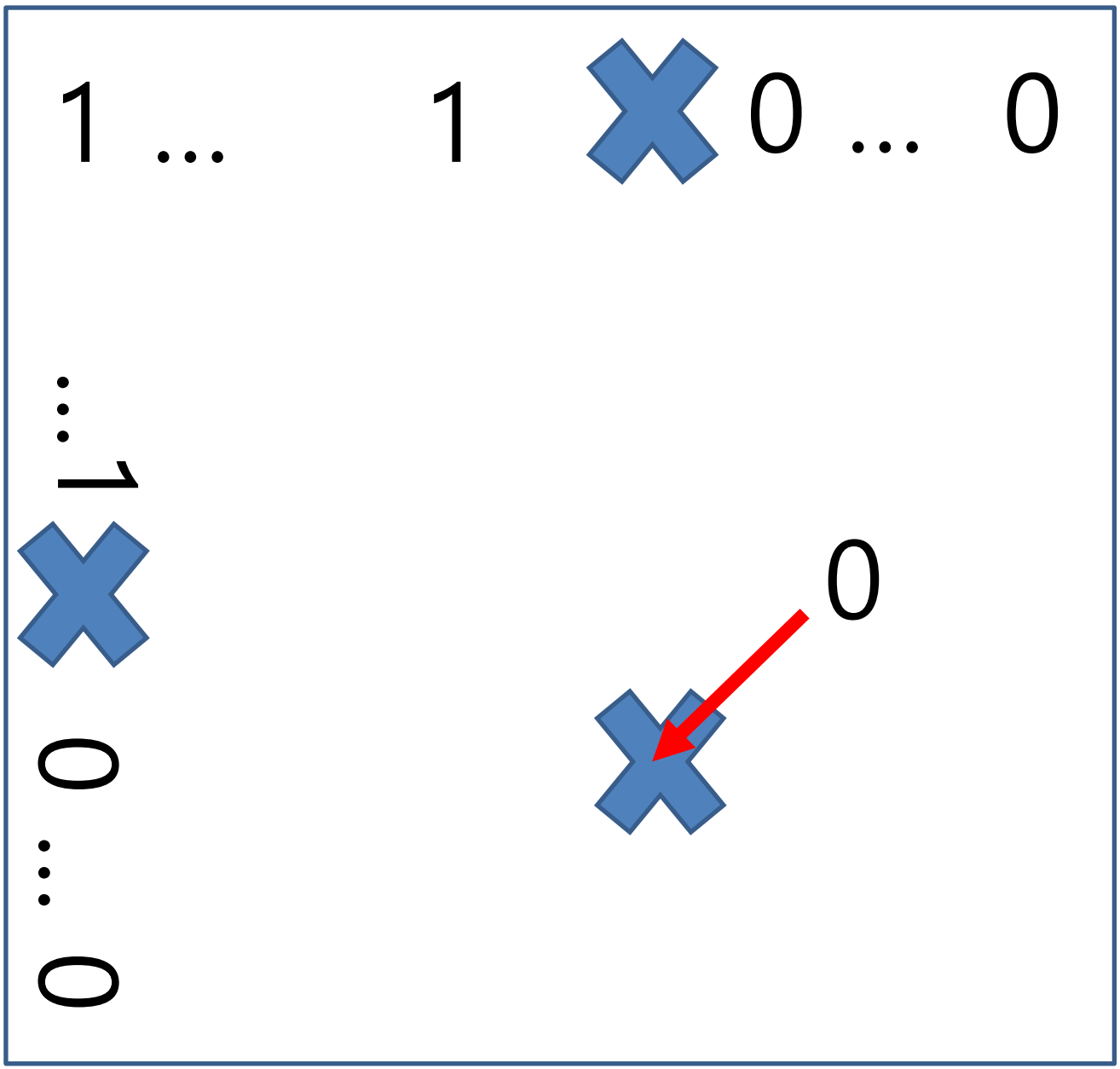

```
Input: m = 3, n = 7
Output: 28
```

$$Space\ complexity = O(\min(m,n))$$

응용 2. 장애물



Idea



code

```python
def uniquePathsWithObstacles(self, obstacleGrid: List[List[int]]) -> int:
    m, n = len(obstacleGrid), len(obstacleGrid[0])
    M = [[0] * n for _ in range(m)]

    obs_appeared = False
    for i in range(m):
        if obstacleGrid[i][0] == 1:
            obs_appeared = True
        if obs_appeared:
            M[i][0] = 0
        else:
            M[i][0] = 1

    obs_appeared = False

    for i in range(n):
        if obstacleGrid[0][i] == 1:
            obs_appeared = True
        if obs_appeared:
            M[0][i] = 0
        else:
            M[0][i] = 1

    for i in range(1, m):
        for j in range(1, n):
            M[i][j] += M[i-1][j] + M[i][j-1]
            if obstacleGrid[i][j] == 1:
                M[i][j] = 0

    return M[m-1][n-1]
```

응용 3. 물고기 잡기

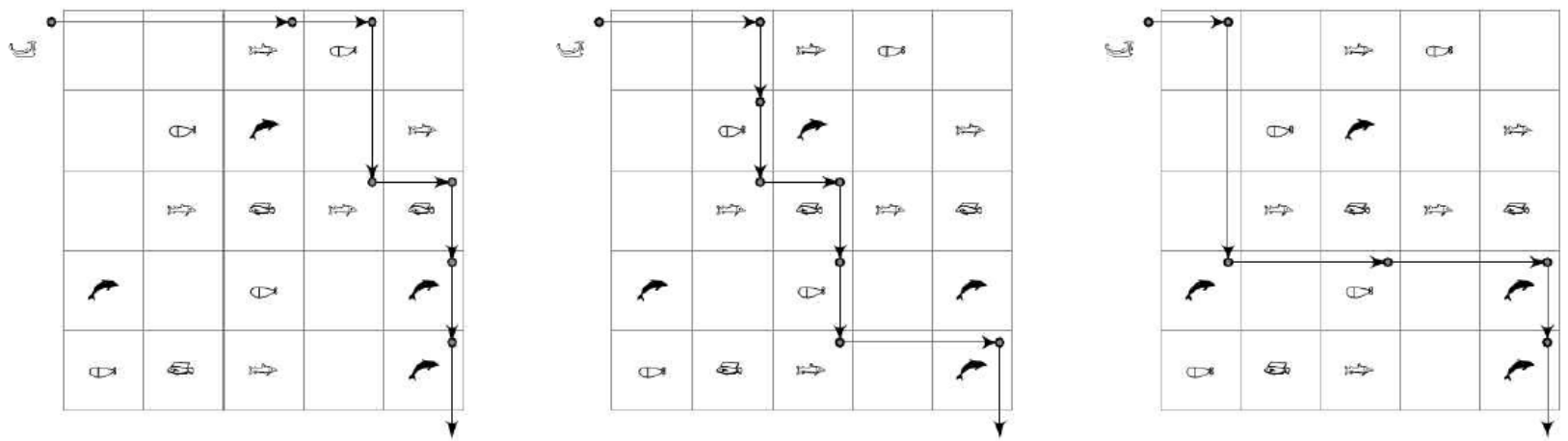
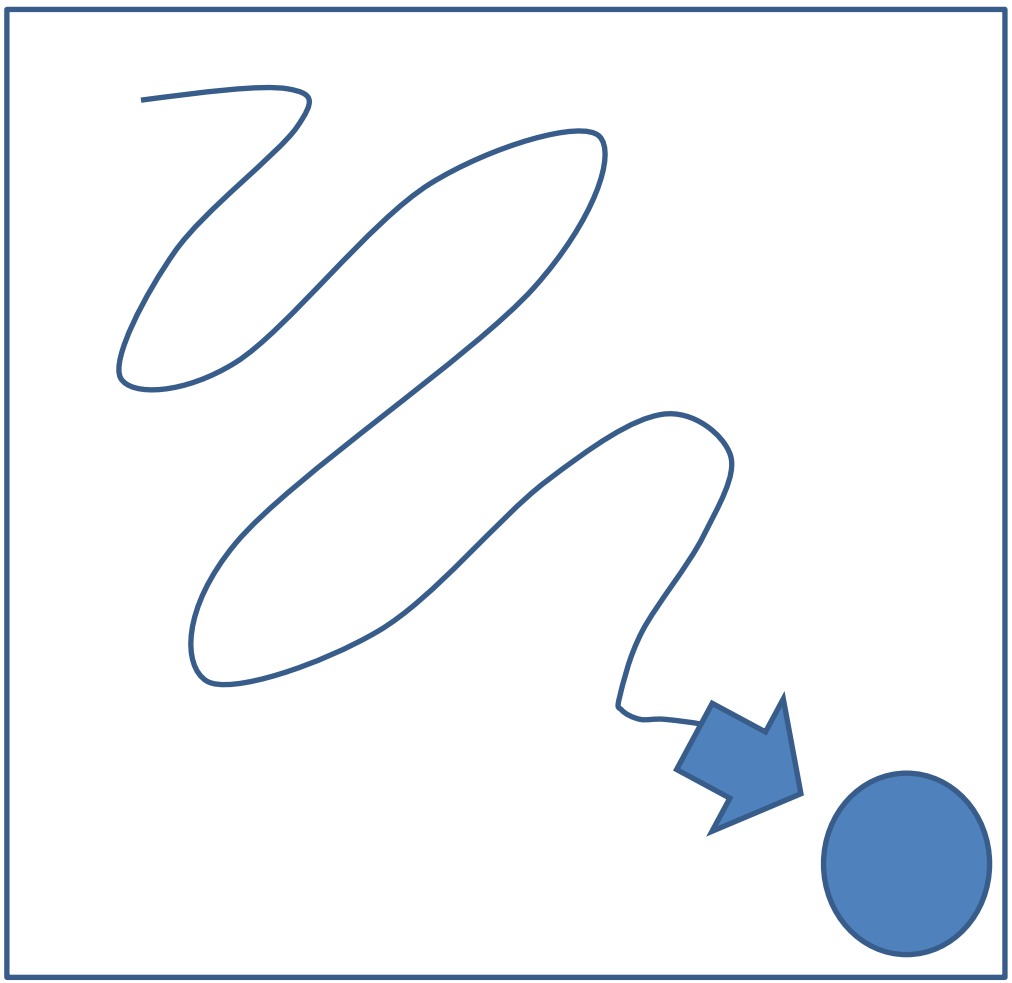


Figure 17.7: Alternate paths for a fisherman. Different types of fish have different values, which are known to the fisherman.

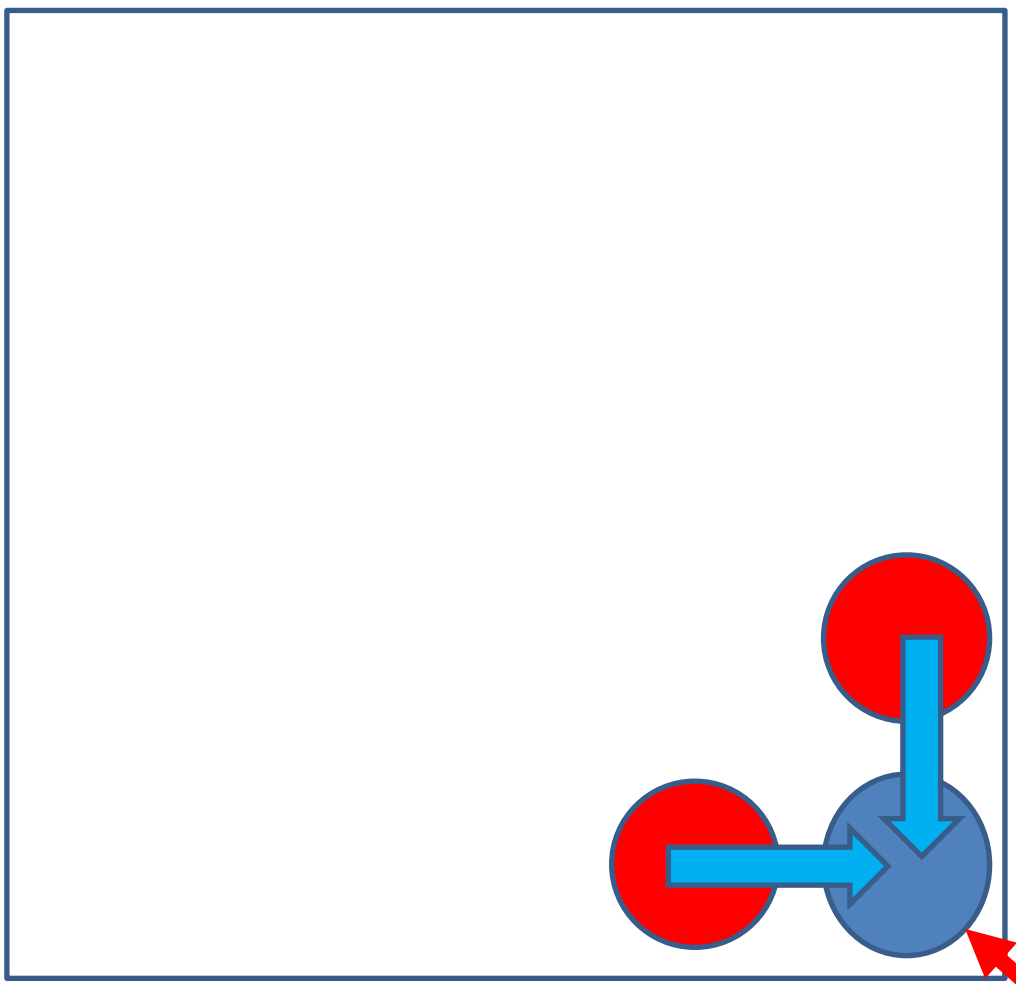
차이점: 모든 경로 vs 한 최적 경로의 값

Idea



$M[i][j]$
= i, j까지 왔을 때의 최대로 많이 잡은 물고기 마리 수

Idea



물고기 있으면 +1

code

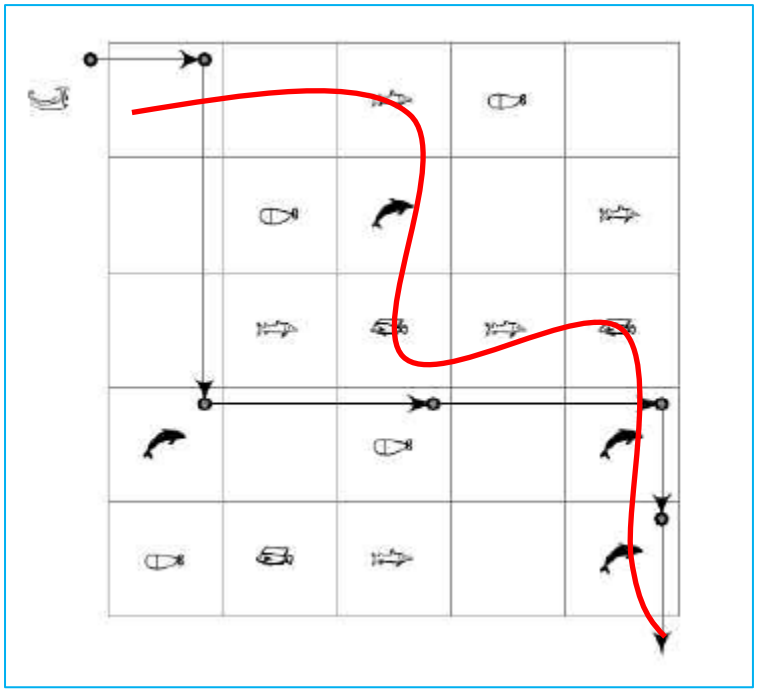
$$M[i][j] = M[i-1][j] + M[i][j-1] + fish[i][j]$$

```
M[0][0] = 0 if fishmap[0][0] == -1 else 1
for i in range(1, m):
    if fishmap[i][0] > -1:
        M[i][0] = M[i-1][0] + 1
    else:
        M[i][0] = M[i-1][0]
for i in range(1, n):
    if fishmap[0][i] > -1:
        M[0][i] = M[0][i-1] + 1
    else:
        M[0][i] = M[0][i-1]

for i in range(1, m):
    for j in range(1, n):
        if fishmap[i][j] > -1:
            M[i][j] = max(M[i-1][j], M[i][j-1]) + 1
        else:
            M[i][j] = max(M[i-1][j], M[i][j-1])
```

결과

[0, 0, 1, 2, 2]
[0, 1, 2, 2, 3]
[0, 2, 3, 4, 5]
[1, 2, 4, 4, 6]
[2, 3, 5, 5, 7]



Summary

- 문제 16.3 : 2차원 배열 순회 방법 개수
  - ✓기본 문제
    - https://leetcode.com/problems/unique-paths/
  - ✓응용 1.공간 복잡도 줄이기
  - ✓응용 2.장애물
    - https://leetcode.com/problems/unique-paths-ii/
  - ✓응용 3.물고기

들어 주셔서 감사합니다