[BOJ] 2169. 로봇 조종하기

Date	@February 17, 2025
Problem Link	https://www.acmicpc.net/problem/2169
Assignment Link	https://www.acmicpc.net/problem/1520

문제

2169. 로봇 조종하기

NASA에서는 화성 탐사를 위해 화성에 무선 조종 로봇을 보냈다.

지형은 **N×M 배열**로 단순화 한다.

로봇은 움직일 때 배열에서 **(1) 왼쪽(←), 오른쪽(→), 아래쪽(↓)으로 이동**할 수 있지만, 위쪽으로는 이동할 수 없다. 또한 **(2) 한 번 탐사한 지역은 탐사하지 않는다.** 로봇을 **왼쪽 위 (1,1)에서 출발시켜 오른쪽 아래 (N,M)으로** 보내려고 한다.

위의 조건을 만족하면서 ,**(3) 탐사한 지역들의 가치의 합이 최대가 되도록** 하는 프로그램을 작성하시오.

입력

- N, M
- 배열의 각 수는 절댓값이 100을 넘지 않는 정수 (=지역의 가치)

출력

최대 가치의 합

예제

```
# 예제 입력
5 5 # 5x5 배열
10 25 7 8 13
68 24 -78 63 32
12 -69 100 -29 -25
-16 -22 -57 -33 99
7 -76 -11 77 15

# 예제 출력
319
```

아이디어

▼ 문제 유형

💡 **문제 유형**

DP(Dynamic Progamming)

- **DP 테이블 정의**

  `dp[i][j]`: (i,j) 좌표까지 도달할 때 탐사한 지역 가치의 최대 합

- **DP 점화식 구성 (경로 선택)**

  어떤 좌표 (i,j)에 도달하는 방법은 세 가지 입니다.

  - **위(↓)**에서 오는 경우: `dp[i-1][j]`
  - **왼쪽(←)**에서 오는 경우: `dp[i][j-1]`
  - **오른쪽(→)**에서 오는 경우: `dp[i][j+1]`

  하지만, 한 번 탐사한 지역은 다시 올 수 없으므로, 경로를 구하는 방법이 다르다.

▼ 아이디어

💡 **아이디어**

- **특정 좌표로 갈 수 있는 방법 (3가지)**

map[i][j]

10	25	7	8	13
68	24	-78	63	32
12	-69	100	-29	-25
-16	-22	-57	-33	99
7	-76	-11	77	15

http://blog.naver.com/occidere

  - **위(↓)**에서 오는 것
  - **왼쪽(←)**에서 오는 것
  - **오른쪽(→)**에서 오는 것

[예제]

map[i][j]

10	25	7	8	13
68	24	-78	63	32
12	-69	100	-29	-25
-16	-22	-57	-33	99
7	-76	-11	77	15

http://blog.naver.com/occidere

5×5 행렬

- **첫번째 행(`i=0`)은, 왼쪽에서 오른쪽으로 진행하는 경우밖에 없음** (왔던 칸을 다시 못오기 때문에)
  → 하나씩 값을 누적한 값이 그 칸에 도달하는 최대가치가 됨

이미 지나온 길은 다시 못가므로 오른쪽으로 갈 수 밖에 없다

출발점 (1,1) →

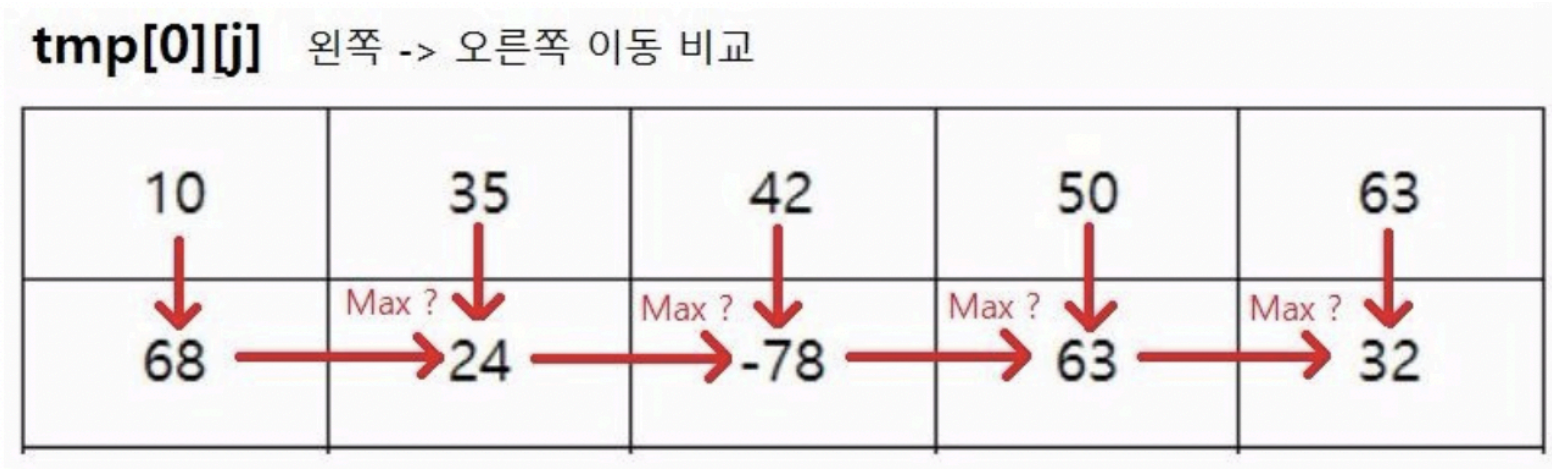
10	25	7	8	13

d[1][j]

10	35	42	50	63

http://blog.naver.com/occidere

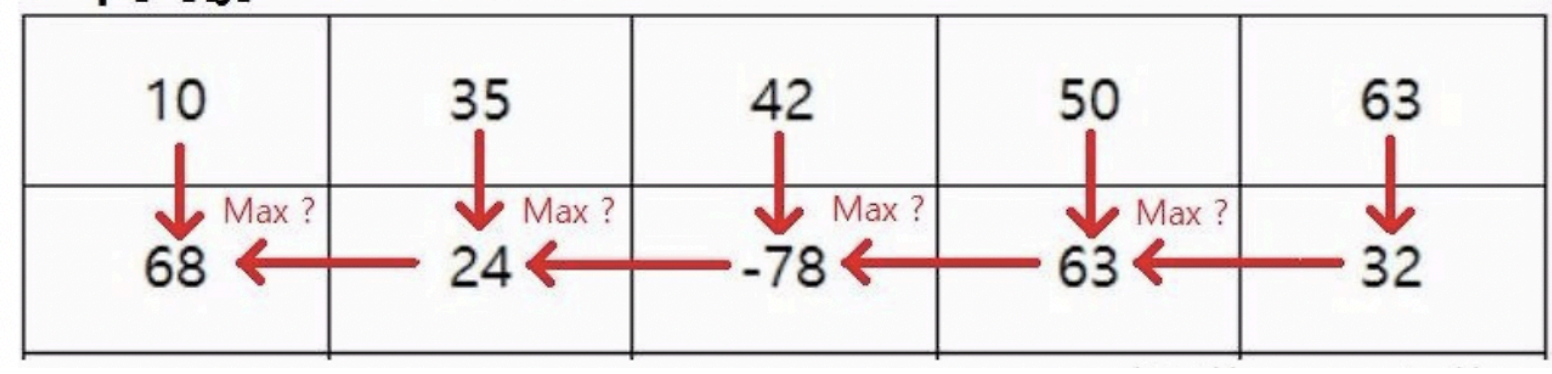
- 나머지 행들은 순서대로 탐색하여 좌표값들을 업데이트
  → 두번째 행(`i=1`)부터는 왼쪽/오른쪽을 나누어 각 좌표의 최대가치를 업데이트할 수 있음
- 이때, 각 행들은 **두가지 임시배열(왼쪽→오른쪽으로 진행시 구해진 최대값, 오른쪽→왼쪽으로 진행시 구해진 최대값)로 표현**
  - 왼쪽→오른쪽(`left_to_right`): **위**에서 왔을 때와 **왼쪽**에서 왔을 때의 값이 비교되어, 최대값으로 업데이트



left_to_right[i][j] = max(dp[i-1][j], left_to_right[i][j-1]) + grid[i][j]

tmp[0][j] 좌->우	78	102	24	113	145

- 오른쪽→왼쪽(right_to_left): **위**에서 왔을 때의 값과 **오른쪽**에서 왔을 때의 값이 비교되어, 최대값으로 업데이트



right_to_left[i][j] = max(dp[i-1][j], right_to_left[i][j+1]) + grid[i][j]

tmp[1][j] 우->좌	172	104	80	158	95

- 이후, 두 임시배열을 비교함으로써, DP 테이블의 각 좌표를 최대값으로 업데이트 할 수 있음

tmp[0][j] 좌->우	78	102	24	113	145
tmp[1][j] 우->좌	172	104	80	158	95
대소비교후 최종 d[i][j]	172	104	80	158	145

dp[i][j] = max(left_to_right[j], right_to_left[j])

정답

- DP 테이블을 만들고, (i,j)까지 올 수 있는 최대값을 저장한다.
- 첫번째 행은 왼쪽에서 오른쪽으로만 업데이트한다.
- 두번째 행부터, 3가지 방향(위, 왼쪽, 오른쪽)에서 오는 값을 고려한다.
- 각 행을 왼쪽→오른쪽, 오른쪽→왼쪽 두 번 계산하여 최대값으로 `dp[i][j]`를 갱신한다.
- 마지막 `dp[n-1][m-1]` 값이 정답이 된다.

풀이

```
import sys
input = sys.stdin.readline

# 입력
N, M = map(int, input().split()) # 지도의 크기
grid = [list(map(int, input().split())) for _ in range(N)] # 각 지역의 가치

# DP 테이블
dp = [[0] * M for _ in range(N)]

# 첫 번째 행 초기화 (왼쪽에서 오른쪽으로 누적합)
dp[0][0] = grid[0][0]
for j in range(1, M):
    dp[0][j] = dp[0][j-1] + grid[0][j] # (0,1)→(0,2)→(0,3)..

# 두 번째 행부터 (왼쪽에서 오른쪽, 오른쪽에서 왼쪽으로 진행)
for i in range(1, N):
    left_to_right = [0] * M
    right_to_left = [0] * M

    # 왼쪽 → 오른쪽
    left_to_right[0] = dp[i-1][0] + grid[i][0] # 첫번째 열은 위쪽에서만 올 수 있음
    for j in range(1, M): # 두번째 열부터는 위쪽, 왼쪽에서 오는 경우 중 선택
        left_to_right[j] = max(dp[i-1][j], left_to_right[j-1]) + grid[i][j]

    # 오른쪽 → 왼쪽
    right_to_left[M-1] = dp[i-1][M-1] + grid[i][M-1] # 마지막 열은 위쪽에서만 올 수 있음
    for j in range(M-2, -1, -1): # 그 다음 열부터는 위쪽, 오른쪽에서 오는 경우 중 선택
        right_to_left[j] = max(dp[i-1][j], right_to_left[j+1]) + grid[i][j]

    # 두 개의 배열을 비교해 dp[i][j] 갱신
    for j in range(M):
        dp[i][j] = max(left_to_right[j], right_to_left[j])

# 정답 출력
print(dp[N-1][M-1]) # 마지막 위치에 저장된 값이 탐사한 지역 가치 합의 최대값
```

제출 번호	아이디	문제	결과	메모리	시간	언어	코드 길이	제출한 시간
90196588	learntosurf	2169	맞았습니다!!	94980 KB	1120 ms	Python 3 / 수정	983 B	8초 전

과제

1520. 내리막길