BOJ

#20057. 마법사 상어와 토네이도

https://www.acmicpc.net/problem/20057

25.02.10

Problem

https://www.acmicpc.net/problem/20057 **시간: 1시간 10분**

복잡한 구현 문제. 문제의 로직을 어떻게 **시간 안에** 코드로 옮길 수 있을지 고민해보자!

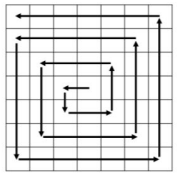
마법사 상어와 토네이도

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	512 MB	12126	8520	5754	71.098%

문제

마법사 상어가 토네이도를 배웠고, 오늘은 토네이도를 크기가 N×N인 격자로 나누어진 모래밭에서 연습하려고 한다. 위치 (r, c)는 격자의 r행 c열을 의미하고, A[r][c]는 (r, c)에 있는 모래의 양을 의미한다.

토네이도를 시전하면 격자의 가운데 칸부터 토네이도의 이동이 시작된다. 토네이도는 한 번에 한 칸 이동한다. 다음은 N = 7인 경우 토네이도의 이동이다.



토네이도가 한 칸 이동할 때마다 모래는 다음과 같이 일정한 비율로 흩날리게 된다.

		2%		
	10%	7%	1%	
5%	α	y ← x		
	10%	7%	1%	
		2%		

토네이도가 x에서 y로 이동하면, y의 모든 모래가 비율과 α가 적혀있는 칸으로 이동한다. 비율이 적혀있는 칸으로 이동하는 모래의 양은 y에 있는 모래의 해당 비율만큼이고, 계산에서 소수점 아래는 버린다. α로 이동하는 모래의 양은 비율이 적혀있는 칸으로 이동하지 않은 남은 모래의 양과 같다. 모래가 이미 있는 칸으로 모래가 이동하면, 모래의 양은 더해진다. 위의 그림은 토네이도가 왼쪽으로 이동할 때이고, 다른 방향으로 이동하는 경우는 위의 그림을 해당 방향으로 회전하면 된다.

토네이도는 (1, 1)까지 이동한 뒤 소멸한다. 모래가 격자의 밖으로 이동할 수도 있다. 토네이도가 소멸되었을 때, 격자의 밖으로 나간 모래의 양을 구해보자.

📝 Problem implementation

문제 요구 사항 정리

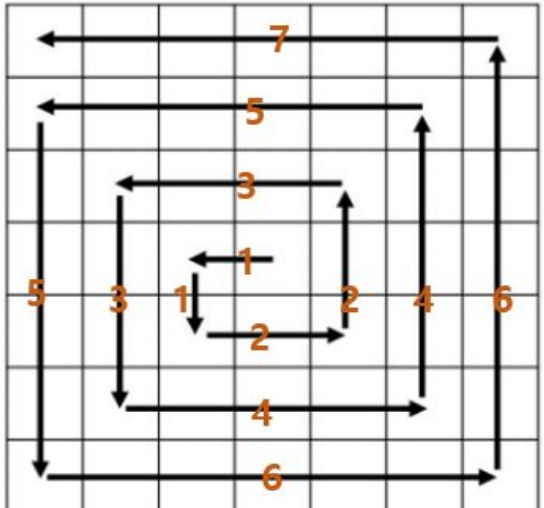
1. 게임 시작 시 토네이도는 지도의 가운데 칸에서부터 시작하여 (1, 1)에서 종료.
2. 매 초 토네이도 이동, 반 시계 방향으로 회전 고려
3. 토네이도 이동 시 흩날리는 모래 계산
    a. 방향 별로 흩날리는 모래 계산이 달라지는것 유의
4. 지도 밖으로 나간 모래의 양 계산

Hint

Step 1.

> *2.* 매 초 토네이도 이동, 반 시계 방향으로 회전 고려

- **N X N 지도 위에 토네이도 이동 시뮬레이션을 진행할 때 방향 벡터 필요.**
- **언제 방향 전환을 할까? (정답 코드는 더 쉬운 1번으로 구현됨. 1번 추천)**
  - **1. 이동 규칙성 찾기 (좌하 1번 -> 우상 2번 -> 좌하 3번 -> ...)**
  - **2. 방향 전환시 조건 찾기 (현재 방향 기준 다음 방향의 칸이 방문한 적 있으면 방향 유지, 없으면 방향 전환)**



Hint

Step 2.

> *3.* 토네이도 이동 시 흩날리는 모래 계산

- **현재 진행하고 있었던 방향 별로 흩날리는 모래의 계산도 달라져야 한다.**

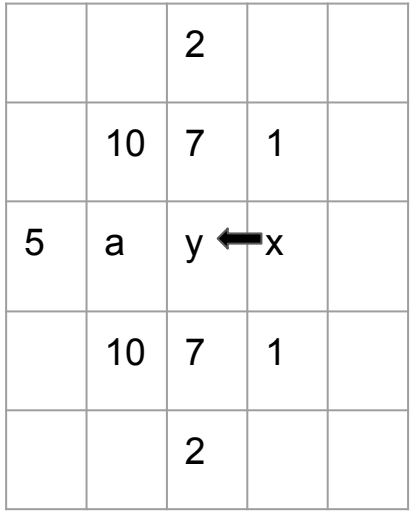
```
# 2. 방향별 모래 비율 위치

left = [(1, 1, 0.01), (-1, 1, 0.01), (1, 0, 0.07), (-1, 0, 0.07), (1, -1, 0.1),

          (-1, -1, 0.1), (2, 0, 0.02), (-2, 0, 0.02), (0, -2, 0.05), (0, -1, 0)]

right = [(x, -y, z) for x, y, z in left]

down = [(-y, x, z) for x, y, z in left]

up = [(y, x, z) for x, y, z in left]
```

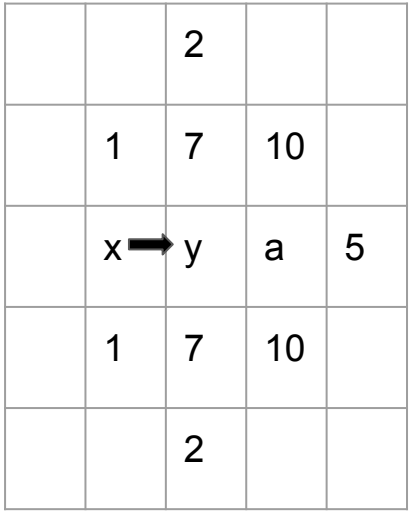
Hint

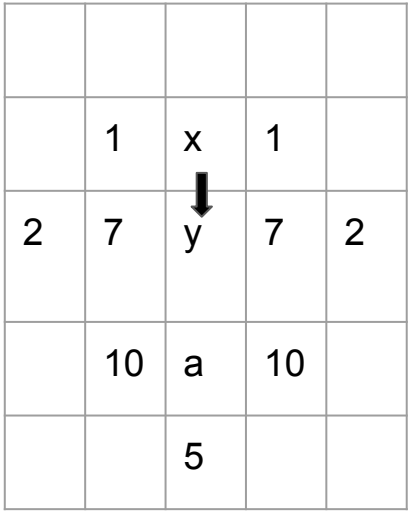
Step 2.

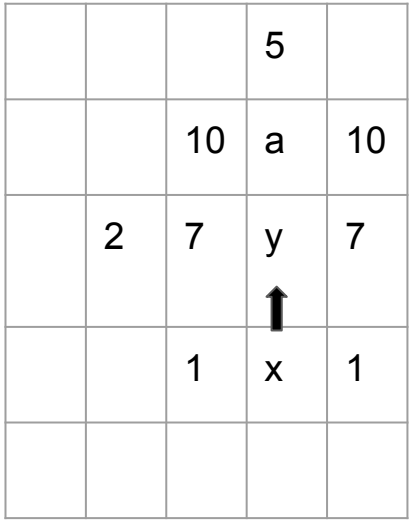
3. 토네이도 이동 시 흩날리는 모래 계산

ex) 좌측 방향에서는 y기준 10%는 벡터 (-1, -1) (1,-1)인데
우측 방향에서는 (-1,1) (1,1)이고
하단 방향에서는 (1,-1) (1,1)이며
상단 방향에서는 (-1,-1) (-1,1)
즉 모래가 날리는 칸은 좌측 방향인 (x,y) 기준 우측(x,-y) 하단(-y,x) 상단(y,x)









✨Solution

```
# 모래 계산하는 함수
def recount(time, dx, dy, direction):
    global ans, s_x, s_y

    # y좌표 계산 & x좌표 갱신
    for _ in range(time):
        s_x += dx
        s_y += dy
        if s_y < 0:  # 범위 밖이면 stop
            break

        # 3. a, out_sand
        total = 0  # a 구하기 위한 변수
        for dx, dy, z in direction:
            nx = s_x + dx
            ny = s_y + dy
            if z == 0:  # a (나머지)
                new_sand = sand[s_x][s_y] - total
            else:  # 비율
                new_sand = int(sand[s_x][s_y] * z)
                total += new_sand

            if 0 <= nx < N and 0 <= ny < N:    #인덱스 범위이면 값 갱신
                sand[nx][ny] += new_sand
            else:  # 범위 밖이면 ans 카운트
                ans += new_sand
```

```
N = int(input())
sand = [list(map(int, input().split())) for _
in range(N)]

# 2. 방향별 모래 비율 위치
left = [(1, 1, 0.01), (-1, 1, 0.01), (1, 0,
0.07), (-1, 0, 0.07), (1, -1, 0.1),
        (-1, -1, 0.1), (2, 0, 0.02), (-2, 0,
0.02), (0, -2, 0.05), (0, -1, 0)]
right = [(x, -y, z) for x, y, z in left]
down = [(-y, x, z) for x, y, z in left]
up = [(y, x, z) for x, y, z in left]

s_x, s_y = N//2, N//2  # 시작좌표(x좌표)
ans = 0  # out_sand

# 1.토네이도 회전 방향(y위치)
for i in range(1, N + 1):
    if i % 2:
        recount(i, 0, -1, left)
        recount(i, 1, 0, down)
    else:
        recount(i, 0, 1, right)
        recount(i, -1, 0, up)

print(ans)
```

💻 Assignment

백준 #20056. 마법사 상어와 파이어볼(골드 4)

시뮬레이션 & 복잡한 구현 조건 문제

시뮬레이션과 복잡한 조건이 들어간 가운데 발제문제보다 난이도가 더 높아 정답률이 더 떨어지는 문제.