2021-1

12주차 문제풀이

Sangmyung ICPC Team

마법사 상어와 비바리기

LEE JIWOO

Step1 – 필요한 기능

1. 모든 구름이 d_i 방향으로 s_i칸 이동한다.
2. 각 구름에서 비가 내려 구름이 있는 칸의 바구니에 저장된 물의 양이 1 증가한다.
3. 구름이 모두 사라진다.
4. 2에서 물이 증가한 칸 (r, c)에 물복사버그 마법을 시전한다. 물복사버그 마법을 사용하면, 대각선 방향으로 거리가 1인 칸에 물이 있는 바구니의 수만큼 (r, c)에 있는 바구니의 물이 양이 증가한다.
    - 이때는 이동과 다르게 경계를 넘어가는 칸은 대각선 방향으로 거리가 1인 칸이 아니다.
    - 예를 들어, (N, 2)에서 인접한 대각선 칸은 (N-1, 1), (N-1, 3)이고, (N, N)에서 인접한 대각선 칸은 (N-1, N-1)뿐이다.
5. 바구니에 저장된 물의 양이 2 이상인 모든 칸에 구름이 생기고, 물의 양이 2 줄어든다. 이때 구름이 생기는 칸은 3에서 구름이 사라진 칸이 아니어야 한다.

Step1 – 필요한 기능

move_clouds(dir, dis) ---------- 구름 이동 함수

rain() ---------- 비 내리기 함수

remove_clouds() ---------- 구름 없애기 함수

copy_water() ---------- 물복사 함수

generate_clouds() ---------- 새로운 구름 생성 함수

Step2 – 구현: move_clouds()

```
def move_clouds(dir, dis):
    global clouds
    if dir == 1: unit_vector = (0, -1)
    elif dir == 2: unit_vector = (-1, -1)
    elif dir == 3: unit_vector = (-1, 0)
    elif dir == 4: unit_vector = (-1, 1)
    elif dir == 5: unit_vector = (0, 1)
    elif dir == 6: unit_vector = (1, 1)
    elif dir == 7: unit_vector = (1, 0)
    else: unit_vector = (1, -1)
```

방향(dir)과 거리(dis)를 매개변수로 받음
방향에 따른 단위 벡터(unit_vector)를 설정

```
temp = clouds
clouds = []
for t in temp:
    x, y = t[0], t[1]
    for _ in range(dis):
        x += unit_vector[0]
        if x == -1: x = n-1
        elif x == n: x = 0

        y += unit_vector[1]
        if y == -1: y = n-1
        elif y == n: y = 0
    clouds.append((x, y))
```

clouds에는 최종적으로 이동이 완료된
구름의 좌표만 넣을 것임
← temp 선언, 격자의 경계를 고려하며
거리만큼 반복하여 단위 벡터 값을 더하고 좌표 갱신

Step2 – 구현: rain(), remove_clouds()

```
def rain():
    for cloud in clouds:
        A[cloud[0]][cloud[1]] += 1
```

이동된 구름 위치에 비 내리기~

```
removed = clouds
```

원래 함수로 구현했는데 걍 이렇게 함~

Step2 – 구현: copy_water()

```
def copy_water():
    water_memo = []
    pos = [(-1, -1), (-1, 1), (1, -1), (1, 1)]

    for cloud in clouds:
        water_cnt = 0
        for p in pos:
            if (0 <= cloud[0]+p[0] < n) and (0 <= cloud[1]+p[1] < n):
                if A[cloud[0]+p[0]][cloud[1]+p[1]]:
                    water_cnt += 1
        water_memo.append((cloud[0], cloud[1], water_cnt))

    for memo in water_memo:
        A[memo[0]][memo[1]] += memo[2]
```

대각선 위치에 물이 차 있는 바구니 개수(water_cnt)와,
해당 구름 위치를 water_memo에 기록해둠.
구해진 물복사 값을 더해서 물 양 갱신

Step2 – 구현: generate_clouds()

```
def generate_clouds():
    global clouds
    clouds = []
    for i in range(n):
        for j in range(n):
            if ((i, j) not in removed) and A[i][j] >= 2:
                A[i][j] -= 2
                clouds.append((i, j))
```

격자 전체를 확인하여 삭제된 구름이 있던 자리는 아닌지(not in removed),
바구니에 찬 물의 양은 2 이상인지 확인.
➔ 조건에 맞는 좌표는 clouds에 삽입 → 새로운 구름 생성

Step2 – 구현: 입출력

```
total_water = 0
n, m = map(int, input().split())
A = [list(map(int, stdin.readline().split())) for _ in range(n)]

removed = []
clouds = [(n-1, 0), (n-1, 1), (n-2, 0), (n-2, 1)]
```

초기 clouds는 (N, 1), (N, 2), (N-1, 1), (N-1, 2)에 해당하는 위치로 초기화

```
for _ in range(m):
    d, s = map(int, input().split())
    move_clouds(d, s)
    rain()
    removed = clouds
    copy_water()
    gernerate_clouds()

for i in range(n):
    for j in range(n):
        total_water += A[i][j]

print(total_water)
```

이동 명령 수(m) 만큼 실행 과정 반복!
모든 바구니에 들어있는 물의 양에 대한 누적 합계 구하고
출력~!~

Thanks