Algorithm

미로탈출 명령어

(Greedy)

[프로그래머스] 미로 탈출 명령어

미로 탈출 명령어

문제 설명

`n` x `m` 격자 미로가 주어집니다. 당신은 미로의 (x, y)에서 출발해 (r, c)로 이동해서 탈출해야 합니다.

단, 미로를 탈출하는 조건이 세 가지 있습니다.

1. 격자의 바깥으로는 나갈 수 없습니다.
2. (x, y)에서 (r, c)까지 이동하는 거리가 총 `k` 여야 합니다. **이때, (x, y)와 (r, c)격자를 포함해, 같은 격자를 두 번 이상 방문해도 됩니다.**
3. 미로에서 탈출한 경로를 문자열로 나타냈을 때, 문자열이 사전 순으로 가장 빠른 경로로 탈출해야 합니다.

이동 경로는 다음과 같이 문자열로 바꿀 수 있습니다.

- l: 왼쪽으로 한 칸 이동
- r: 오른쪽으로 한 칸 이동
- u: 위쪽으로 한 칸 이동
- d: 아래쪽으로 한 칸 이동

예를 들어, 왼쪽으로 한 칸, 위로 한 칸, 왼쪽으로 한 칸 움직였다면, 문자열 `"lul"` 로 나타낼 수 있습니다.

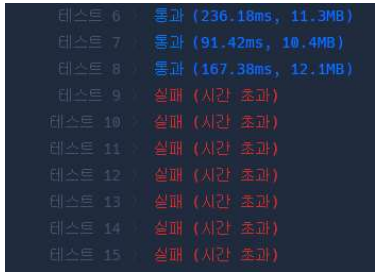
미로에서는 인접한 상, 하, 좌, 우 격자로 한 칸씩 이동할 수 있습니다.

예를 들어 다음과 같이 3 x 4 격자가 있다고 가정해 보겠습니다.

- n x m 미로에서 탈출하기 위해 커맨드를 입력해야 함
- K 번을 움직여 탈출 목적지에 도달해야 함.
- 이동 가능한 방향은 상하좌우 네 방향
- 같은 지점을 중복해서 지나갈 수 있음
- 가능한 이동 경로 중에 사전 순으로 나열했을 때 가장 앞선 경우의 경로를 반환하도록 함.

1차 시도 - BFS

- BFS로 풀어서 몇 가지 테스트 케이스는 통과 했으나
- 대부분의 경우에는 시간 초과가 발생
- BFS는 너비 우선 탐색이라 탐색할 필요 없는 경로도 탐색을 해서 시간이 오래 걸리는 듯




```python
from collections import deque

def solution(n, m, x, y, r, c, k):
    candidates = []
    start = (x, y, 0, "")
    q = deque()
    q.append(start)

    while(q):
        row, col, step, string = q.pop()
        if row == r and col == c and step == k:
            candidates.append(string)
        elif step < k:
            for rp, cp, s in [(1, 0, "d"), (0, 1, "r"), (-1, 0, "u"), (0, -1, "l")]:
                nrow = row + rp
                ncol = col + cp
                ns = string + s
                if 1 <= nrow <= n and 1 <= ncol <= m:
                    q.append((nrow, ncol, step + 1, ns))

    candidates.sort()

    if len(candidates) == 0:
        return "impossible"

    return candidates[0]
```

구글링 후 2차 시도 – Greedy

- 어짜피 사전 순으로 나열할 것이기 때문에 'd' -> 'l' -> 'r' -> 'u' 방향 순으로 우선적으로 움직이는 것이 유리함.
- 가야할 움직임 수를 저장
- 가야할 움직임 수가 목표지점까리의 거리보다 훨씬 많으면 rl을 우선 부여한다.
- 하지만 rl을 부여하기 전에 내려가야할 거리가 있으면 down부터 먼저하고 그 다음에 왼쪽으로 이동할 거리가 있으면 l 부터 하고 나서 rl을 부여한다.
- 가야할 거리와 remain이 일치하게 되면 d -> l -> r -> u 순으로 입력한다.

```python
def cal_dist(x, y, r, c):
    return abs(r - x) + abs(c - y)

def solution(n, m, x, y, r, c, k):
    answer = ''
    distance = cal_dist(x, y, r, c)
    if distance > k or (k - distance) % 2 == 1:
        return "impossible"
    remain = k
    while x < n and remain > cal_dist(x, y, r, c):
        remain -= 1
        x += 1
        answer += "d"
    while 1 < y and remain > cal_dist(x, y, r, c):
        remain -= 1
        y -= 1
        answer += "l"
    while remain > cal_dist(x, y, r, c):
        remain -= 2
        answer += "rl"
    if x < r:
        answer += "d" * (r - x)
        x = r
    if y > c:
        answer += "l" * (y - c)
        y = c
    if y < c:
        answer += "r" * (c - y)
        y = c
    if x > r:
        answer += "u" * (x - r)
        x = r
    return answer
```