[22.07.29] 얼음행성탐사

TS얼음행성탐사

김 태 현

문 제

- R * C 크기의 빙하에 N개의 구조물이 있다.
- 로봇이 투하지역에 투하되어 탐사지역으로 이동한다.
- 이동은 상하좌우 4방향으로만 가능하고 한번 움직이면 구조물과 부딪힐 때까지 멈추거나 방향도 바꿀 수 없다. 범위를 벗어나면 바다에 빠진다.
- 탐사 지역을 지나치면 탐사가 완료된다.
- 탐사 완료시까지 구조물에 부딪히는 횟수를 최소로 해야 한다.

void init(int R, int C, int N, int sRow[], int sCol[])

빙하 크기 : R * C	5 <= R <= 700 , 5 <= C <= 10,000
구조물 개수 : N	0 <= N<= 30,000
구조물 좌표 : (sRow[i], sCol[i])	0 <= i < N
좌표 범위 : (0,0) ~ (R-1, C-1)	

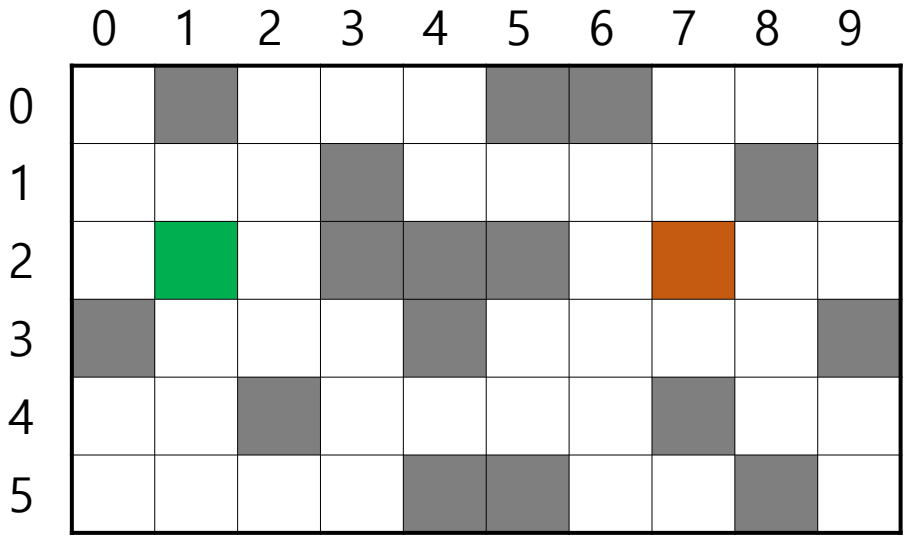
int minDamage(int sr, int sc, int er, int ec) <= 30

투하 지역 : (sr, sc)
탐사 지역 : (er, ec)
투하 지역에서 탐사 지역을 지나치기까지 구조물에 부딪히는 횟수를
최소로 하는 횟수 반환

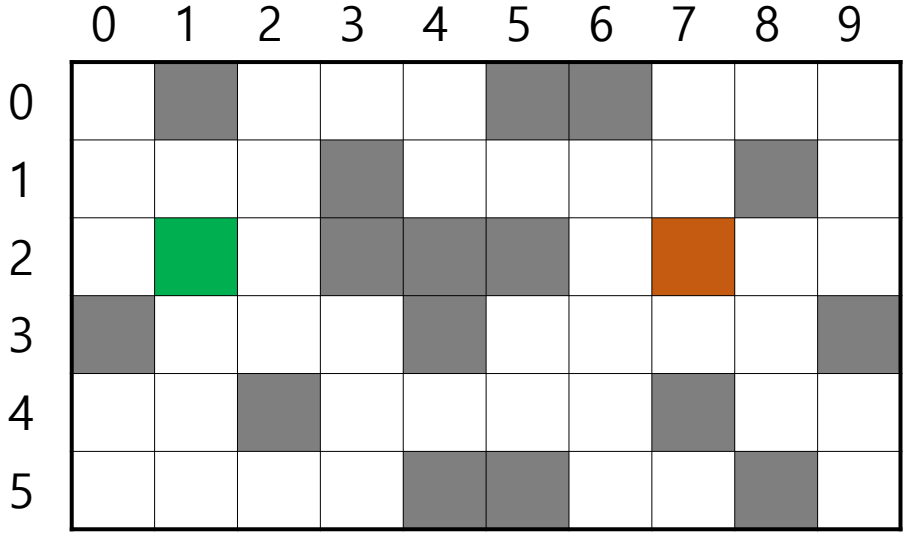
Naive

BFS

- queue에 (row, col, damage) 등록
- 4방향 각각 한 칸씩 이동하며 바다 or 구조물 만날때까지 이동
- 구조물을 만나면 로봇 위치 (r, c)을 (r, c, damage+1)로 등록
- 로봇이 위치할 수 있는 위치 개수 : 구조물 개수 * 4 = 120,000
- O(30*700*10,000*4) 보다는 훨씬 적겠지만 최적화 필요해 보임



행, 열별 구조물 오름차순 관리



- (row, col, damage) 의 R[row] 에서 col 이상인 가장 작은 값 선택 = c2
  C[col] 에서 row 이상인 가장 작은 값 선택 = r2
  => linear search
- 위에서 선택한 바로 직전 값 선택 = r1 , c1
- 구조물 있는 경우만 queue에 등록
  (r1+1, col) , (r2-1, col) , (row, c1+1) , (row, c2-1)
- 좌표 1씩 이동하면서 구조물을 찾는게 아닌 구조물 단위로 검색하니 더 효율적인 검색 가능

R

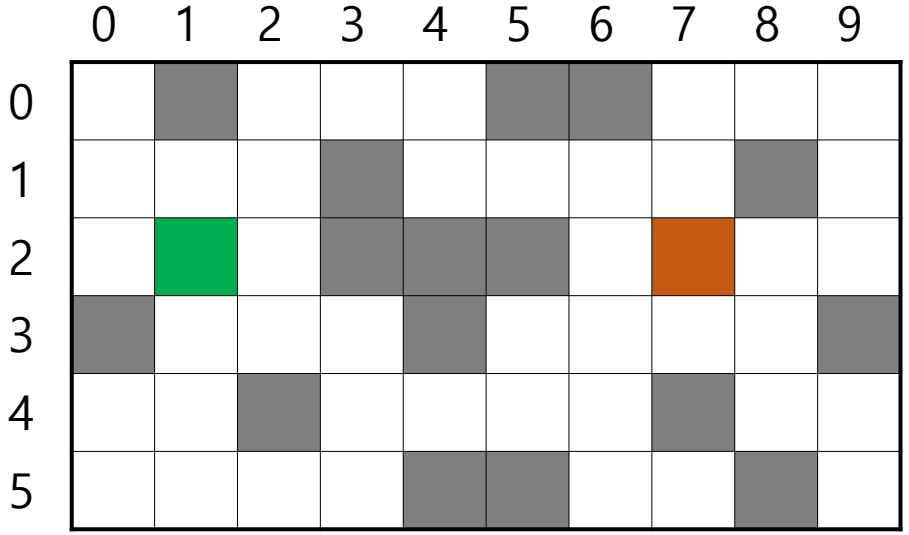
row	[cols,..]
0	1, 5, 6
1	3, 8
2	3, 4, 5
3	0, 4, 9
4	2, 7
5	4, 5, 8

C

col	[rows,..]
0	3
1	1
2	4
3	1, 2
4	2, 3, 5
5	0, 2, 5
6	0
7	4
8	1, 5
9	3

lower_bound()

크거나 같은 가장 작은 위치 검색



- (row, col, damage) 의 R[row] 에서 col 이상인 가장 작은 값 선택 = c2
  C[col] 에서 row 이상인 가장 작은 값 선택 = r2
  => lower_bound
- 위에서 선택한 바로 직전 값 선택 = r1 , c2
- 구조물 있는 경우만 queue에 등록
  (r1+1, col) , (r2-1, col) , (row, c1+1) , (row, c2-1)
- 만나게 되는 구조물을 log 복잡도로 구하기 때문에 안정적

R

row	[cols,..]
0	1, 5, 6
1	3, 8
2	3, 4, 5
3	0, 4, 9
4	2, 7
5	4, 5, 8

C

col	[rows,..]
0	3
1	1
2	4
3	1, 2
4	2, 3, 5
5	0, 2, 5
6	0
7	4
8	1, 5
9	3

구현 스타일

행, 열 별 관리

1. vector, sort
2. set

바다 처리(예외 처리)

1. 양 끝 (row, -1), (row, C), (-1, col), (R, col) 등록
2. 등록하지 않고 end(), begin() 으로 예외 처리

목표 지점 도달 확인

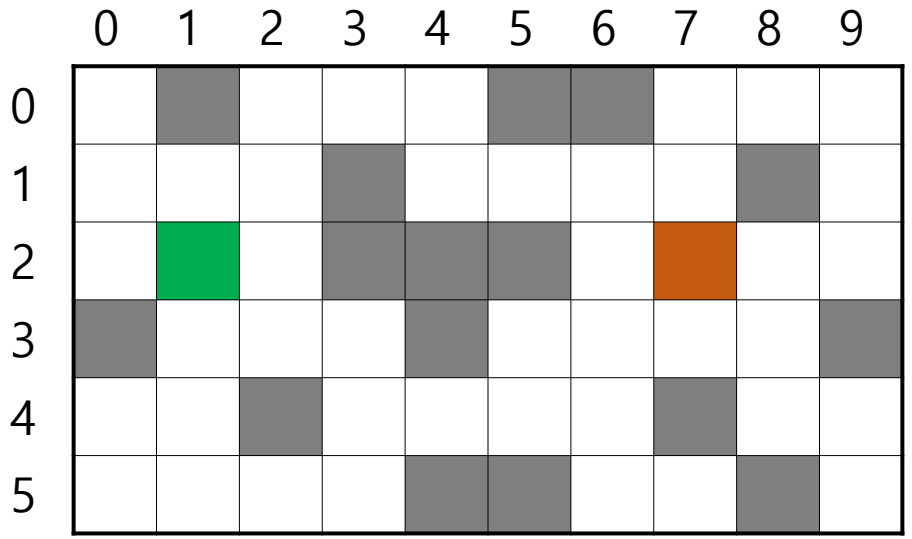
1. (er, ec) 등록하고 만나는지 확인 (set인 경우)
2. r1<->r2 , c1<->c2 구간에 포함되는지 확인

R

row	[cols,..]
0	1, 5, 6
1	3, 8
2	3, 4, 5
3	0, 4, 9
4	2, 7
5	4, 5, 8

C

col	[rows,..]
0	3
1	1
2	4
3	1, 2
4	2, 3, 5
5	0, 2, 5
6	0
7	4
8	1, 5
9	3




감사합니다