# 05.28 정기 OMS :
# 내 BFS에선
# 시간이 흘러

# 01

# 백준킹 우승자 발표

# 축하드립니다

완전백짱이셔 님입니다!

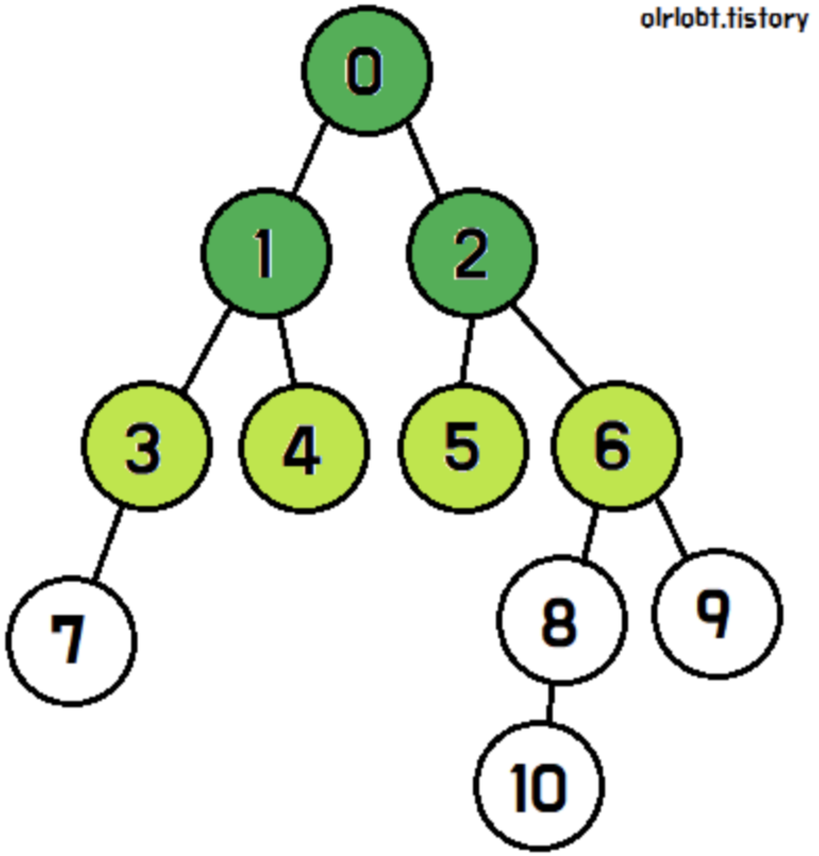
# 02

# OMS : 백범준님
# 내 BFS는 시간이 흘러

# BFS가 뭐죠?

너비 우선 탐색 (Breadth-First Search) !

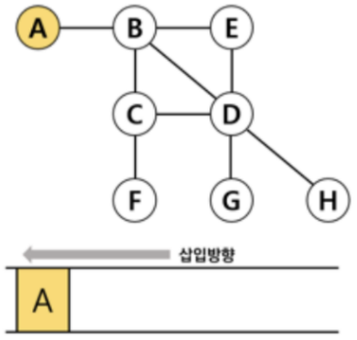
트리나 그래프를 탐색하는 기법 중 하나로,
시작 노드로부터 자식 노드들을 순서대로 탐색하면서
[너비를 우선으로 탐색하는 알고리즘]이다.

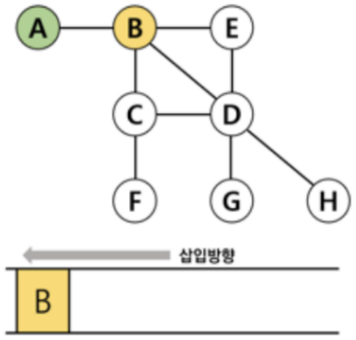


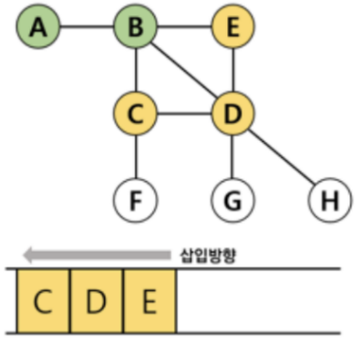
## BFS가 뭐죠?

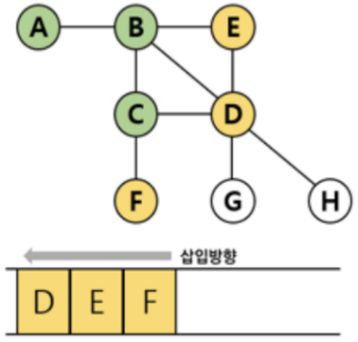
수도 코드로 BFS를 나타내 보아요.

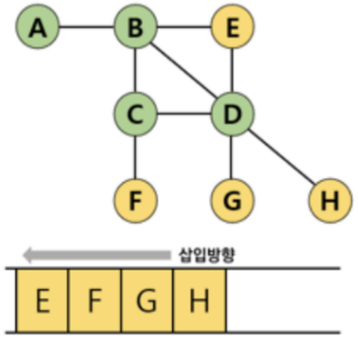
시작 노드를 방문 표시 후, 큐에 넣는다.
큐에 아무 노드가 없을 때까지:
    큐 가장 앞 노드를 꺼낸다.
    꺼낸 노드에 인접한 노드들을 모두 보면서:
    처음 방문한 노드면:
        방문한 노드 표시를 해준다.
        predecessor 변수를 큐에서 꺼낸 노드로 설정
        큐에 넣어준다.

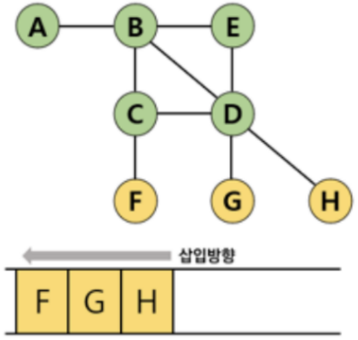


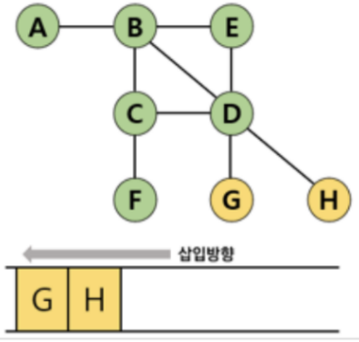



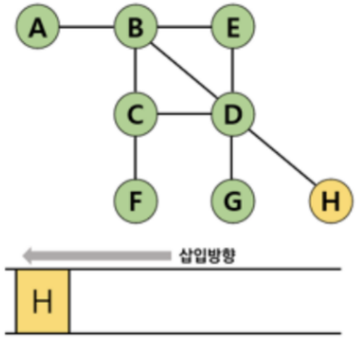



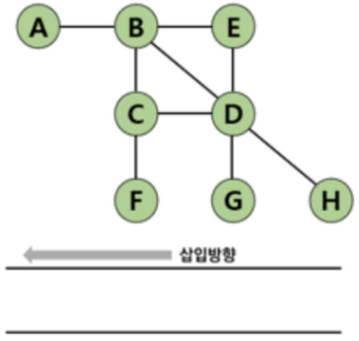




https://cdragon.tistory.com



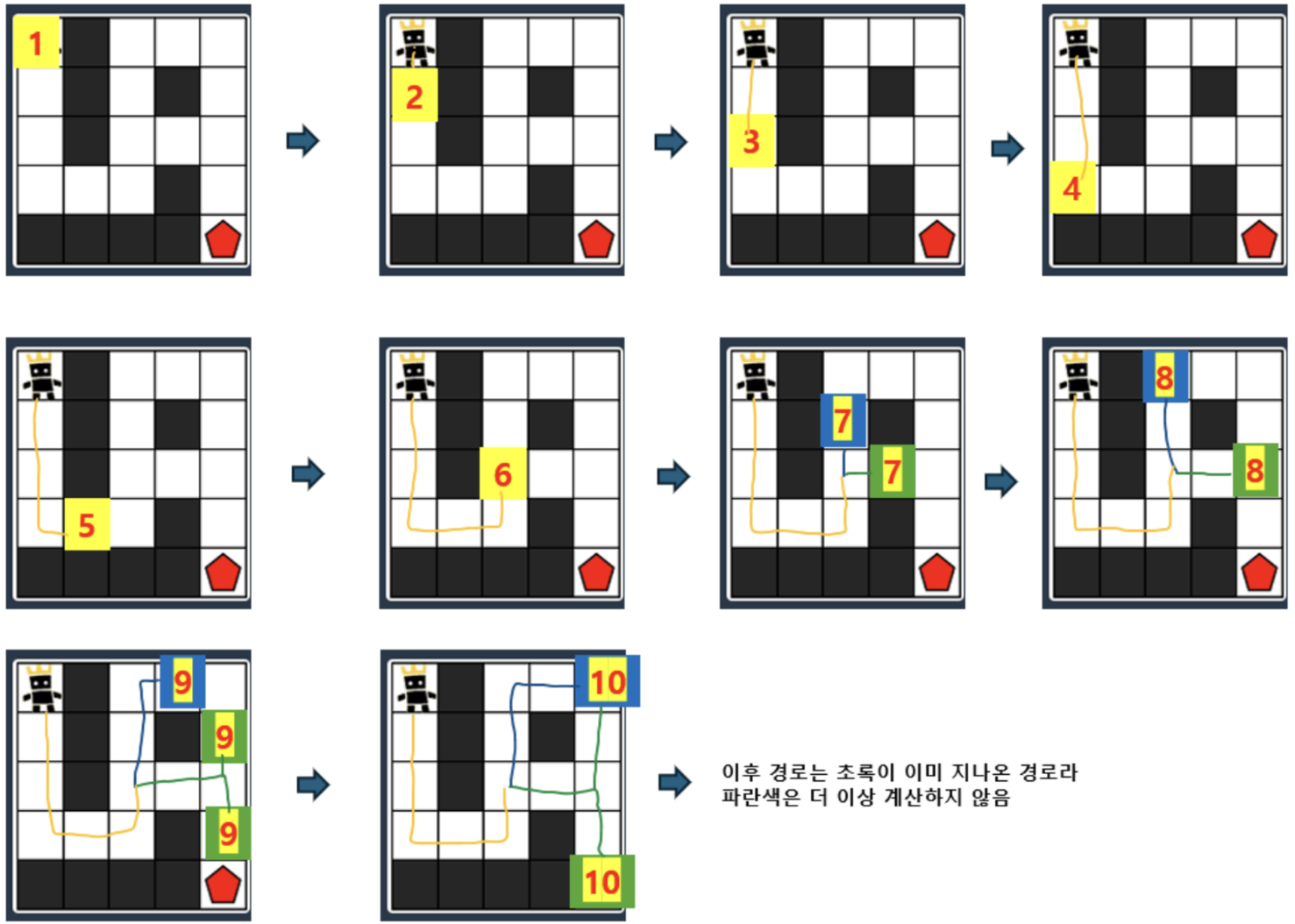

https://cdragon.tistory.com

# BFS가 뭐죠?

2차원 배열로 나타낸 맵이 있네요
상하좌우가 이어진 그래프와 같다!
현재 경로 기준 상하좌우를 모두 분석

BFS를 이용하여 목표지점까지의 탐색
→ 가장 먼저 탐색되어 반환 받은 경로
== 최단 경로

*최단 경로가 보장되나요??
→ 네! 너비 우선 탐색은 보장됩니다.



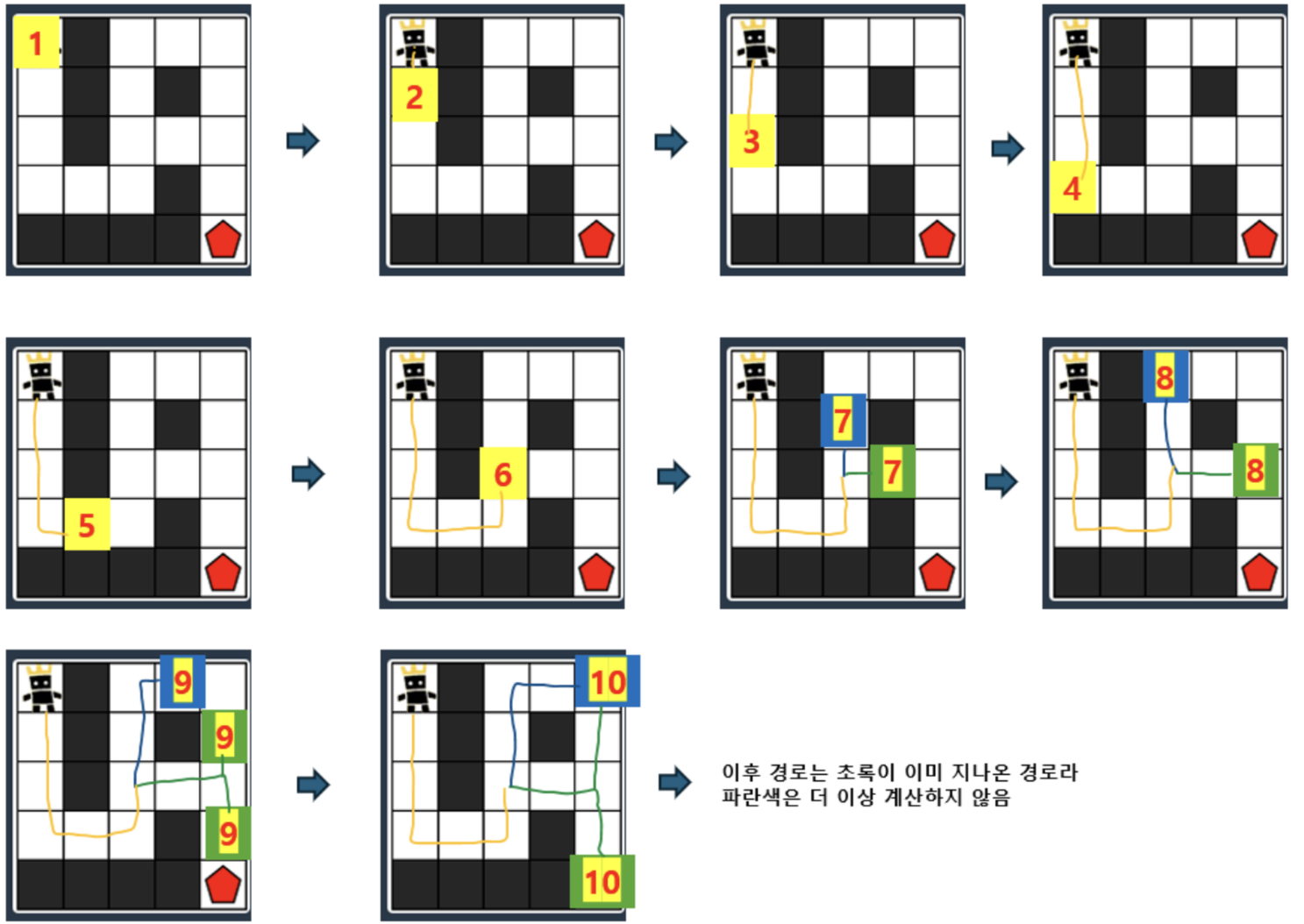

출처 : 발광견 vlog

# BFS가 뭐죠?

수도 코드로 BFS를 나타내 보아요.

시작 노드를 방문 표시 후, 큐에 넣는다.
큐에 아무 노드가 없을 때까지:
    큐 가장 앞 노드를 꺼낸다.
    꺼낸 위치에서 상하좌우를 모두 보면서:
    유효한 길이면서 처음 방문:
        직전 위치를 저장
        방문한 노드 표시를 해준다.
        갱신된 위치를 큐에 넣어준다.

가장 먼저 도착지점에 도착하면
가장 가까운 경로를 찾은 아이겠지요.
판서로 추가 설명!




출처 : 발광견 vlog

# BFS가 뭐죠?


이걸 대충 어떻게 구현하나요?

```
while q:
    x, y = q.popleft()
    if (x, y) == goal:            # 조기 종료
        break
    for dx, dy in fordxdy:
        nx, ny = x+dx, y+dy
        # 경계 및 장애물 체크
        if 0 <= nx < R and 0 <= ny < C and grid[nx][ny] == 0:
            if dist[nx][ny] == INF:    # 아직 미방문
                dist[nx][ny] = dist[x][y] + 1
                parent[nx][ny] = (x, y)
                q.append((nx, ny))
```

출처 : 발광견 vlog

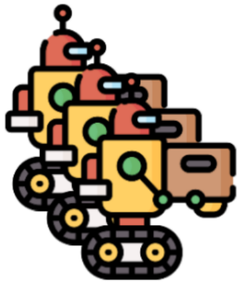
# BFS로만 해결할 수 없었던 것...

OS 수업 중 과제에 이런 조건이 있었습니다.

중앙 관제 노드

명령 전달

상태 전달



자율 주행 물류 로봇

| | | | | | | |
|---|---|---|---|---|---|---|
| X | X | X | X | X | X | X |
| A | | | 7 | | | X |
| X | | 1 | X | 4 | | X |
| B | | 2 | X | 5 | | X |
| X | | 3 | X | 6 | | X |
| C | | | | | S | X |
| X | X | X | X | X | W | X |

중앙 관제 노드

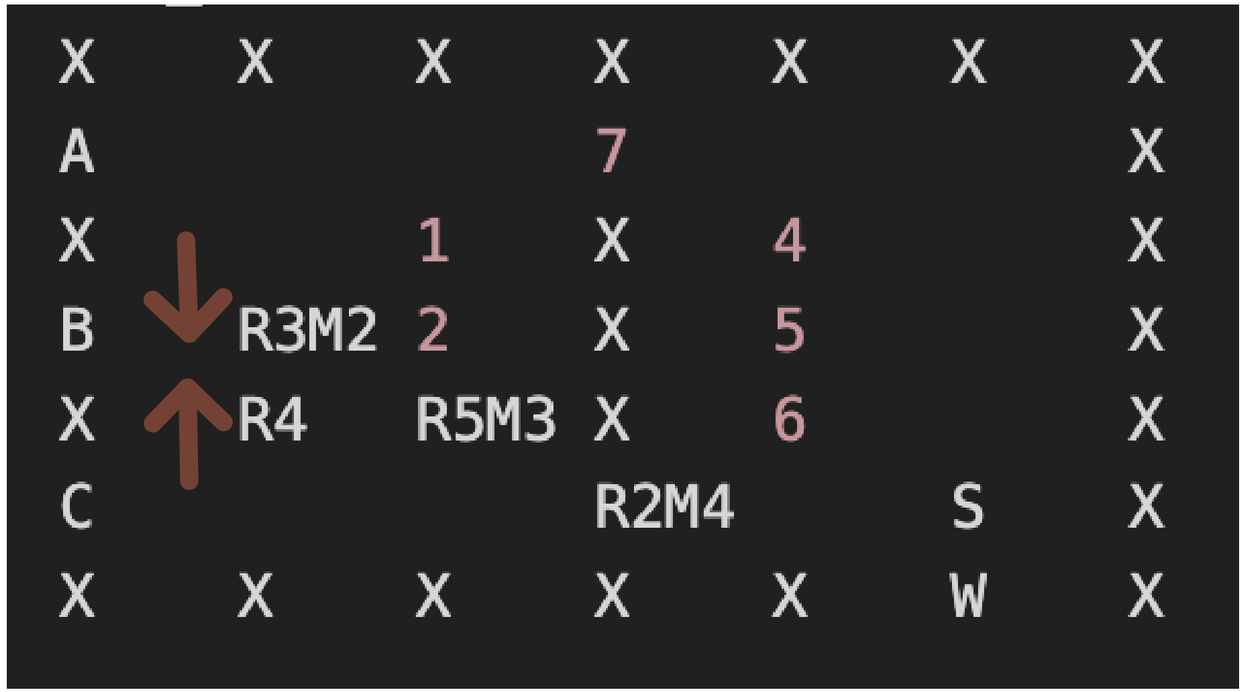
- 물류 로봇이 이동할 수 있는 조건을 고려하여 모든 물류 로봇들이 각자 하역 장소까지 도달하게 해야함
- 모든 물류 로봇에게 상태 정보를 전달 받을 때까지 대기 후 물류 로봇들의 상태 출력 (요구사항 4)
- 이후 상태 정보들을 기반으로 각 물류 로봇에게 대기 혹은 상하좌우 이동 행동 지시 후 단위 스텝 증가(요구사항4)
- 행동 지시 후 모든 물류 로봇들을 Unblock 필수 (요구사항 3)
- 모든 물류 로봇이 물건을 운송하였으면 시뮬레이션 종료

- 자율 주행 물류 로봇(이하 물류 로봇)
  - 해당 물류 로봇이 요구하는 물건 적재 지역을 거쳐 정해진 하역 장소까지 도달해야 함
  - 초기 물류 로봇은 W에 위치해야 함 (요구사항 1)
  - 모든 물류 로봇은 한 번의 단위 스탭 중에 대기 혹은 이동 중 하나의 행동만 가능 (요구사항 4)
    - 상하좌우 중 한 곳으로만 이동 가능
    - 이동할 위치에 다른 물류 로봇이 있을 경우 이동 불가 (단, 하역 장소, 대기 장소에서는 중복 가능)
    - 적재할 물건이 있는 장소 외 다른 물건이 있는 장소에 이동 불가
  - 물류 로봇은 중앙 관제 노드가 정해준 행동을 취한 후 자신의 상태 정보를 관제 노드에게 전달 후 Block(요구사항 3&4)

# BFS로만 해결할 수 없었던 것...

여기서 문제는?? 로봇들이 각자의 경로를 가지게 되면....

무한 대기 상태..!



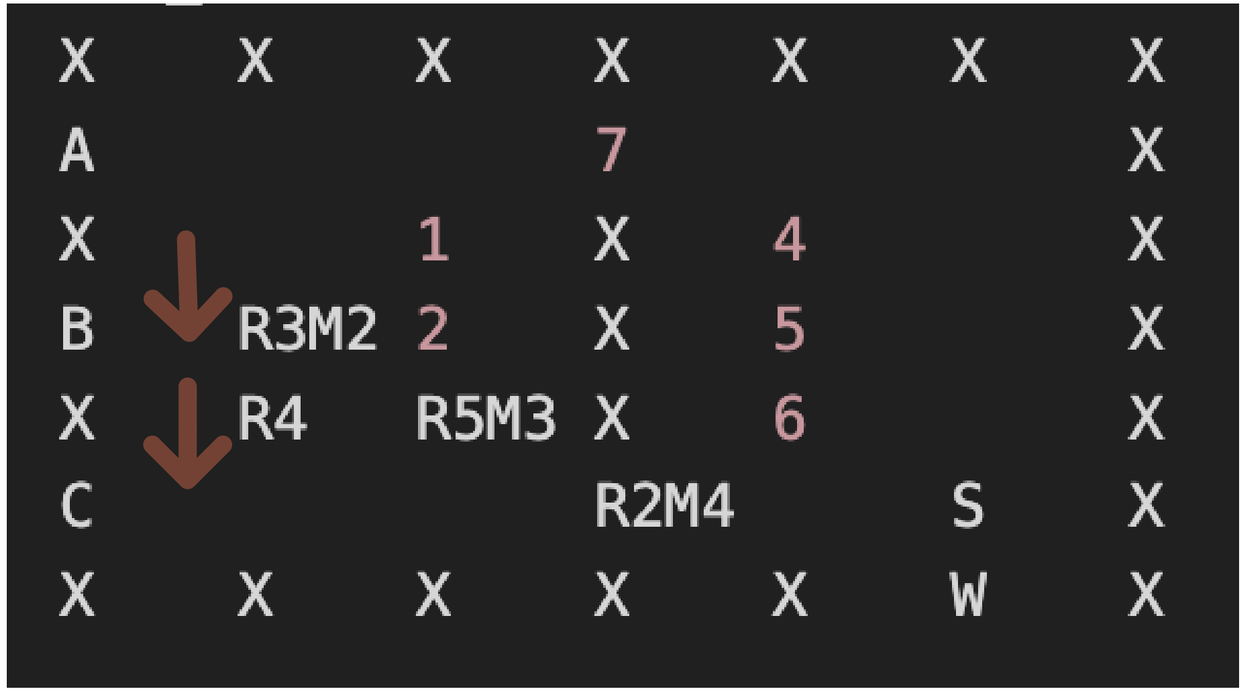
# BFS로만 해결할 수 없었던 것...

로봇을 한 개 씩 보내면 안되나요??

됩니다!
하지만 여러 로봇들을 동시에 컨트롤 하는 것이 가산점의 요소였습니다.

# BFS로만 해결할 수 없었던 것...

그러면 어떻게 해결하나요...



# 그래서 준비했다! 시간 축을 더한 BFS

그러면 어떻게 해결하나요...

| | | | | |
|---|---|---|---|---|
| X | 과자1 | X | 과자2 | X |
| | 범준2 | | 범준1 | |

경로가 겹치는 범준봇 2마리를 만들었습니다.
(판서로 설명)

# 그래서 준비했다! 시간 축을 더한 BFS

이걸 어떻게 표현하나요? 범준1이 가장 먼저 경로를 탐색했고 그에 따라 움직인다고 가정할게요

| | | | | |
|---|---|---|---|---|
| X | 과자1 | X | 과자2 | X |
| | 범준2 | | 범준1 | |

0초

| | | | | |
|---|---|---|---|---|
| X | 과자1 | X | 과자2 | X |
| | | 범준1예약석 | | |

경로계산 탐색할 때에는 현재와 1초뒤의 맵을 봅니다!

범준 2 입장에선 오른쪽으로 가는 길이 막혔네요.

| | | | | |
|---|---|---|---|---|
| X | 과자1 | X | 과자2 | X |
| | 범준2 | 범준1 | | |

1초

범준 2 입장에선 대기하거나 , 왼쪽으로 밖에 가지 못합니다.

# 그래서 준비했다! 시간 축을 더한 BFS

이걸 어떻게 표현하나요? 범준1이 가장 먼저 경로를 탐색한다고 가정할게요

| | | | | |
|---|---|---|---|---|
| X | 과자1 | X | 과자2 | X |
| | 범준2 | 범준1 | | |

1초

| | | | | |
|---|---|---|---|---|
| X | 과자1 | X | 과자2 | X |
| | 범준1 예약석 | | | |

경로계산 탐색할 때에는 현재와 1초뒤의 맵을 봅니다!

범준 2 입장에선 대기도, 오른쪽으로 가지도 못하겠네요

| | | | | |
|---|---|---|---|---|
| X | 과자1 | X | 과자2 | X |
| 범준2 | 범준1 | | | |

2초

범준2 는 왼으로 가야만 합니다.

# 그래서 준비했다! 시간 축을 더한 BFS

이걸 어떻게 표현하나요? 범준1이 가장 먼저 경로를 탐색한다고 가정할게요

| X | 과자1 | X | 과자2 | X |
|---|---|---|---|---|
| 범준2 | 범준1 | | | |

2초

| X | 범준1 예약석 | X | 과자2 | X |
|---|---|---|---|---|
| | 범준2 예약석 | | | |

경로계산 탐색할 때에는 현재와 1초뒤의 맵을 봅니다!

이젠 범준2에겐 오른쪽으로 가는 경로가 생겼습니다!

| X | 범준1도착 | X | 과자2 | X |
|---|---|---|---|---|
| | 범준2 | | | |

3초

이젠 우리가 모두 예상한대로 움직이겠네요

# 그래서 준비했다! 시간 축을 더한 BFS

핵심은?

| X | 과자1 | X | 과자2 | X |
|---|---|---|---|---|
|  | 범준2 |  | 범준1 |  |

| X | 과자1 | X | 과자2 | X |
|---|---|---|---|---|
|  | 범준2 | 범준1 |  |  |

| X | 과자1 | X | 과자2 | X |
|---|---|---|---|---|
| 범준2 | 범준1 |  |  |  |

| X | 범준1도착 | X | 과자2 | X |
|---|---|---|---|---|
|  | 범준2 |  |  |  |

범준2 입장에서는
범준1의 각 step에 따른 위치를(예약석으로 표현되어 있음)
표기한 부분까지 고려하였기 때문에

자신이 가고 싶은 경로가 범준1이 존재할 때에는 벽으로 보이고

시간이 지나면 범준1이 도착하여(범준1의 경로를 막지 않음)
그때에서야, 과자2로 가는 이상적인 경로가 탐색이 된다는 점!

(판서로 추가 설명!)

# 그래서 준비했다! 시간 축을 더한 BFS

수도 코드로 새로 나타내보자

시작 위치를 방문 표시 후, 큐에 넣는다.
큐에 아무 위치가 없을 때까지:
    큐 가장 앞 요소를 꺼낸다.
    꺼낸 위치에서 상하좌우를 모두 보면서:
    (유효한 길이면서) (처음 방문) (해당 step에 reserver 즉 예약된 곳이 아니면):
        직전 위치를 저장
        방문한 위치 표시를 해준다. (해당 스텝의 해당 위치에)
        갱신된 위치를 큐에 넣어준다.
    step을 1 늘린다

성공적으로 경로를 찾았으면:
    위치를 백트래킹한다.
    백트래킹된 경로와 시간 정보를 reserve[][][] (시간,x축위치,y축위치)에 표기

# 그래서 준비했다! 시간 축을 더한 BFS

#bfs_3d 함수 중 일부 발췌
로봇 하나가 시간축을 고려하여 경로를 찾는 구현 코드
오른쪽 코드를 경로를 찾을 때까지 반복합니다!
(도착하거나, 큐에 더이상 요소가 없을 때까지)

```
for(int dir=0;dir<=DIR_WAIT;dir++){
// 상하좌우 (0~3) move, 4 WAIT 모든 명령어 기준 가능한 경로 전부 탐색
    int nr = cur.r + dr[dir]; //명령어에 따른 위치
    int nc = cur.c + dc[dir];
    int nt = cur.t + 1; //1초 추가

    if(dir==WAIT){ nr=cur.r; nc=cur.c; } //움직임 X => WATI

    if(!is_inside(nr,nc)) continue; //맵 벗어나는 경우 X
    if(is_wall(nr,nc))    continue; //벽으로 가는 경우 X
    if(is_rack_cell(nr,nc)
        && rack_idx_at(nr,nc)!=forbid_rack_idx) continue;
        //본인 적재 장소가 아닌경우
    if ( (nr==ROW_A&&nc==COL_A && my_unload!='A') ||
         (nr==ROW_B&&nc==COL_B && my_unload!='B') ||
         (nr==ROW_C&&nc==COL_C && my_unload!='C') )
        continue; //본인의 하역장소가 아닌 경우 X
    if(reserve[nt][nr][nc])             continue;
    // 다른 로봇이 해당 시간에 "예약"한 경우 X
    if(vis[nt][nr][nc])                 continue;
    // 이미 해당 시간 방문(중복 검사) X

    vis[nt][nr][nc]=1; //위 모든 조건 통과시 방문기록을 남기고 dir 저장
    prv[nt][nr][nc]=dir;
    Q[tail++] = (struct node){nr,nc,nt}; //큐에 현재 노드 추가
}
```

# 그래서 준비했다! 시간 축을 더한 BFS

만약 경로를 찾았으면 다른 로봇들이
탐색한 로봇의
해당하는 시간에,
그 위치가,
벽으로 인식되도록 reserve에 표기해주면 끝!

*해당 과제에서는 매 step(초)마다 탐색을 계속하기 때문에, 1초에서의 예약값만 표기

```
int len = bfs3d(robots[i].row, robots[i].col,
                tgt_r,tgt_c,
                robots[i].required_payload,
                robots[i].unload,
                reserve,&first_dir);
//bfs 3d를 통해 dir을 받고 len을 반환 받는다

if(len<0){ first_dir = WAIT; } //경로가 없을 때 -1을 반환하므로

// 1초인 경우의 즉 다음 행동을 하고 나서 로봇의 위치를 예약한다
int nr = robots[i].row + dr[first_dir];
int nc = robots[i].col + dc[first_dir];
bool share = // 만약 공유 공간이라면 예약할 필요가 없다
    (nr == ROW_W && nc == COL_W) ||                          /* W 셀 */
    (robots[i].unload == 'A' && nr == ROW_A && nc == COL_A) ||
    (robots[i].unload == 'B' && nr == ROW_B && nc == COL_B) ||
    (robots[i].unload == 'C' && nr == ROW_C && nc == COL_C);
if(!share){ //공유 공간이 아니면 1초에서의 로봇 좌표를 예약한다 -> 다른 로봇과 병목 방지
    reserve[1][nr][nc] = 1;
    reserve[1][robots[i].row][robots[i].col] = 1;
}
```

# 그래서 준비했다! 시간 축을 더한 BFS

그래서 운체 과제 어떻게 됐을까요?

실제 시연으로 보여드리겠습니다.

매개 변수는  2A:4C:2B:2C:3A

# 03

## 다음 연사자는요?

# 04

# 뒤풀이

가실 분? 어디로 갈까요?

# OMS 끗