SAMSUNG

로봇

한컴에듀케이션
이주현

로봇

문제 개요

- 많은 공장에서 로봇이 이용되고 있다.
- 우리 공장의 로봇은 바라보는 방향으로 궤도를 따라 움직이며
  움직이는 방향은 동, 서, 남, 북 가운데 하나이다.
- 로봇의 이동을 제어하는 명령어는 다음과 같이 두 가지이다.

*** 명령 1. Go k**

- k 는 1 2 또는 3일 수 있다. 현재 향하고 있는 방향으로 k 칸만큼 움직인다.

*** 명령 2. Turn dir**

- dir 은 left 또는 right 이며 각각 왼쪽 또는 오른쪽으로 90° 회전한다.

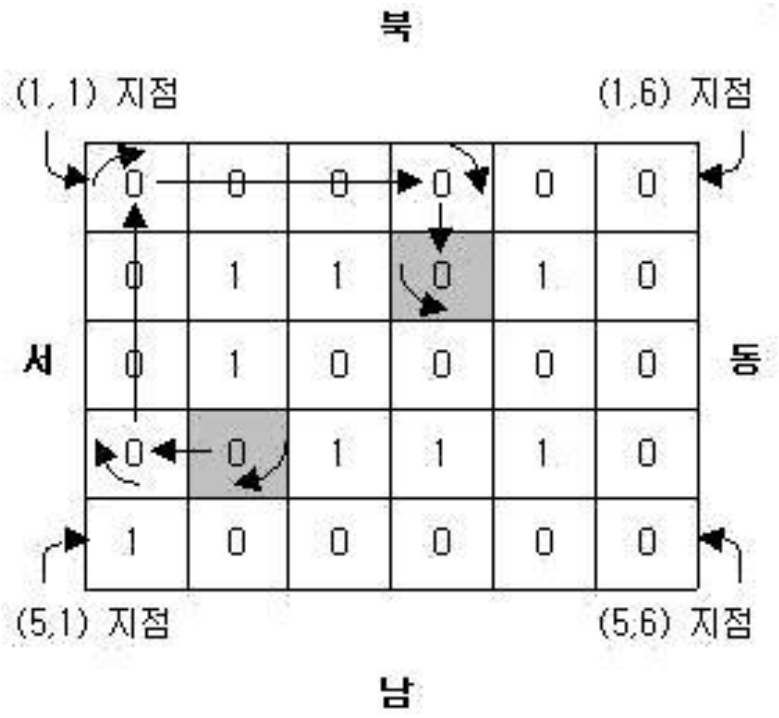
로봇

문제 개요

- 공장 내 궤도가 설치되어 있는 상태가 아래와 같이 0과 1로 이루어진 직사각형 모양으로 로봇에게 입력된다.
- 0은 궤도가 깔려 있어 로봇이 갈 수 있는 지점이고
  1은 궤도가 없어 로봇이 갈 수 없는 지점이다.
- 로봇이 (4, 2) 지점에서 남쪽을 향하고 있을 때
- 이 로봇을 (2, 4) 지점에서 동쪽으로 향하도록 이동시키는 것은
  아래와 같이 9번의 명령으로 가능하다..

로봇

문제 개요



- 로봇의 현재 위치와 바라보는 방향이 주어졌을 때 로봇을 원하는 위치로 이동시키고 원하는 방향으로 바라보도록 하는데
- 최소 몇 번의 명령이 필요한지 구하는 프로그램을 작성하시오.

로봇

문제 개요

[입력 형식]

첫째 줄에 공장 내 궤도 설치 상태를 나타내는
직사각형의 세로 길이 M과 가로 길이 N이 빈칸을 사이에 두고 주어진다.

이 때 M과 N은 둘 다 100이하의 자연수이다.

이어 M줄에 걸쳐 한 줄에 N개씩 각 지점의 궤도 설치 상태를 나타내는
숫자 0 또는 1이 빈칸을 사이에 두고 주어진다.

로봇

문제 개요

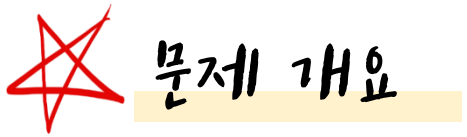
[입력 형식]

다음 줄에는 로봇의 출발 지점의 위치 (행과 열의 번호)와
바라보는 방향이 빈칸을 사이에 두고 주어진다.

마지막 줄에는 로봇의 도착 지점의 위치 (행과 열의 번호)와
바라보는 방향이 빈칸을 사이에 두고 주어진다.

방향은 동쪽이 1, 서쪽이 2, 남쪽이 3, 북쪽이 4로 주어진다.
출발지점에서 도착지점까지는 항상 이동이 가능하다.

로봇



[출력 형식]

첫째 줄에 로봇을 도착 지점에 원하는 방향으로 이동시키는데 필요한 최소 명령 횟수를 출력한다.

로봇

문제 개요

[입력 예]

5 6

0 0 0 0 0 0

0 1 1 0 1 0

0 1 0 0 0 0

0 0 1 1 1 0

1 0 0 0 0 0

4 2 3

2 4 1

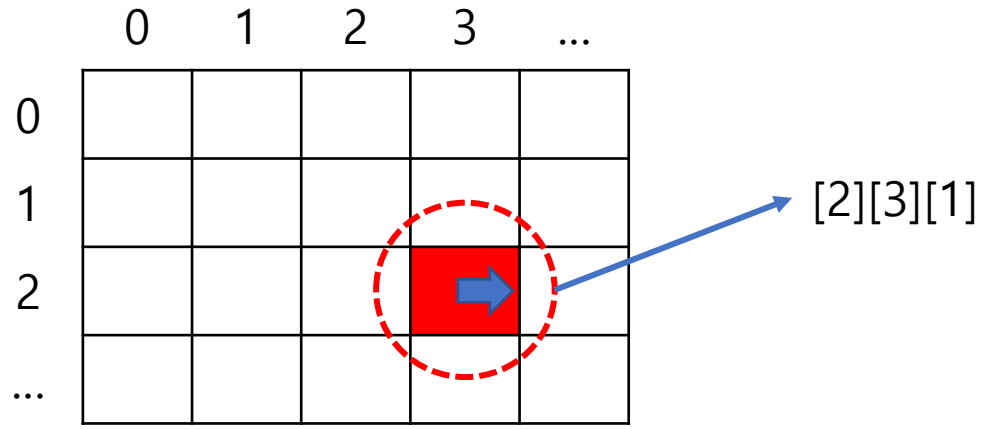
[출력 예]

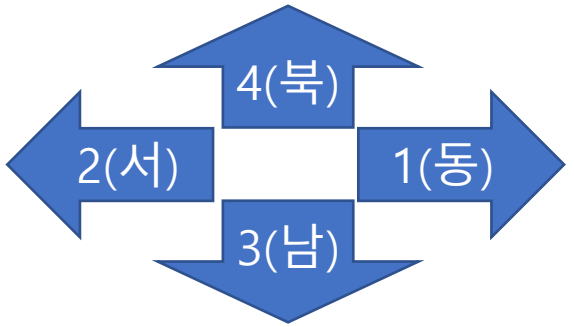
9

로봇

문제 분석

- 로봇의 상태를 나타내는 방법을 생각해보자.
- 행번호, 열번호와 함께 방향 또한 필요하다.
- 따라서 [row][col][dir]로 로봇의 상태를 나타낼 수 있다.

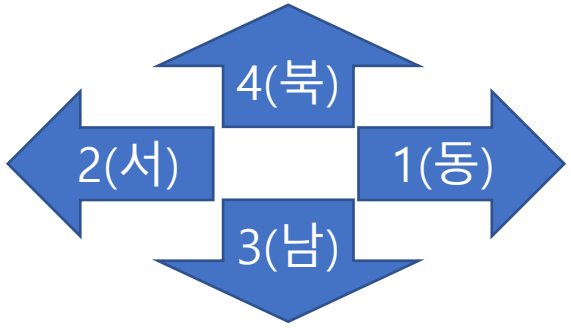




로봇

문제 분석

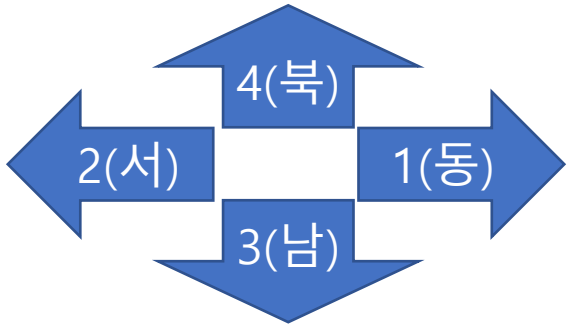
- 로봇은 방향을 전환하는 것에도 명령이 필요하다.
- 방향전환 명령은 왼쪽 또는 오른쪽으로 바꾸는 데는 한번의 명령을 필요로 한다.
- 하지만 역방향으로 방향을 전환하기 위해서는 두 번의 명령이 필요하다.



로봇

문제 분석

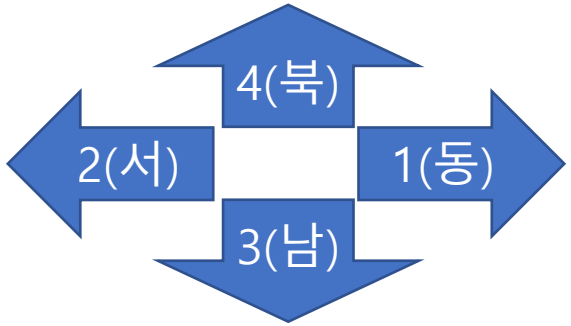
- 아래 그림에서 현재 1번 방향인 경우 3번 또는 4번으로 방향을 전환하는 데는 1의 명령이 필요하지만
  2로 방향을 전환하는 데는 2의 명령이 필요하다.



로봇

문제 분석

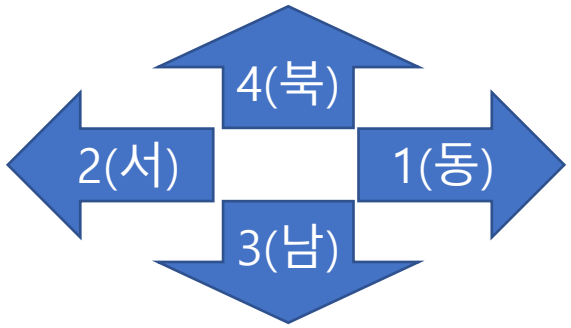
- 따라서 방향을 전환하는 경우 역방향으로 전환하는 것을 고려하지 않는다.
- 예를 들어 현재 1의 방향이라면 3또는 4 방향으로의 전환만 고려한다.



로봇

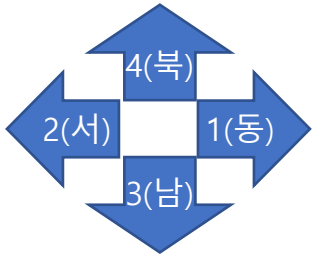
문제 분석

- 이유는 BFS를 사용할 것이기 때문이다.
- BFS는 한 단계씩 만 진행한다.
- 그렇지 않은 경우 목표로 한 상태에 도달했을 때 최소의 명령 수로 도달했음을 보증할 수 없다.



로봇

문제 분석



1	1	1	1	1	1
1	0				1
1		1	1		1
1		1	1		1
1					1
1	1	1	1	1	1

예를 들어 보자.

처음에 로봇은 서쪽(2번 방향)을 보고 있다.

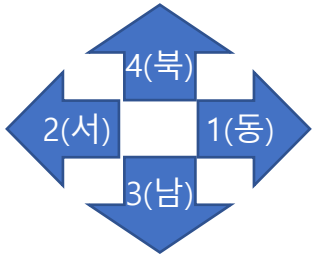
이때 명령 수는 0이다.

보라색 위치에 도착하여

북쪽(4) 방향을 바라보는 것이 목표이다.

로봇

문제 분석



1	1	1	1	1	1
1	0	2			1
1		1	1		1
1		1	1		1
1					1
1	1	1	1	1	1

시작 위치에서

역방향(동쪽 1)으로 회전하는 것을

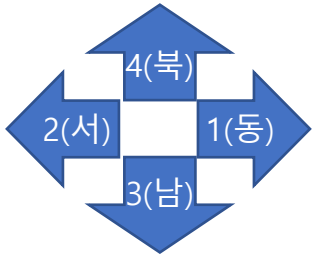
우선하여 실행해 본다.

이 로봇을 주황색 로봇이라고 하자.

명령 수는 2가 된다.

로봇

문제 분석



1	1	1	1	1	1
1	2 0 1				1
1		1	1		1
1		1	1		1
1					1
1	1	1	1	1	1

시작 위치에서

이제 남쪽(3) 방향으로 회전하는

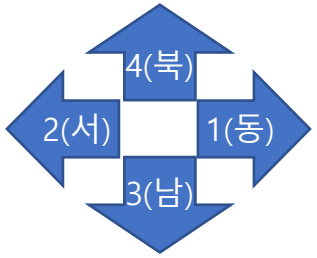
로봇도 출발시켜본다.

이 로봇을 녹색 로봇이라 하자.

명령수는 1이 된다.

로봇

문제 분석

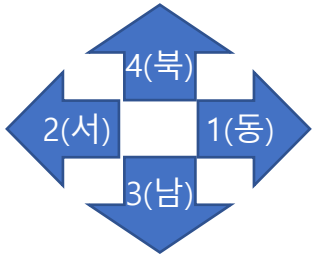


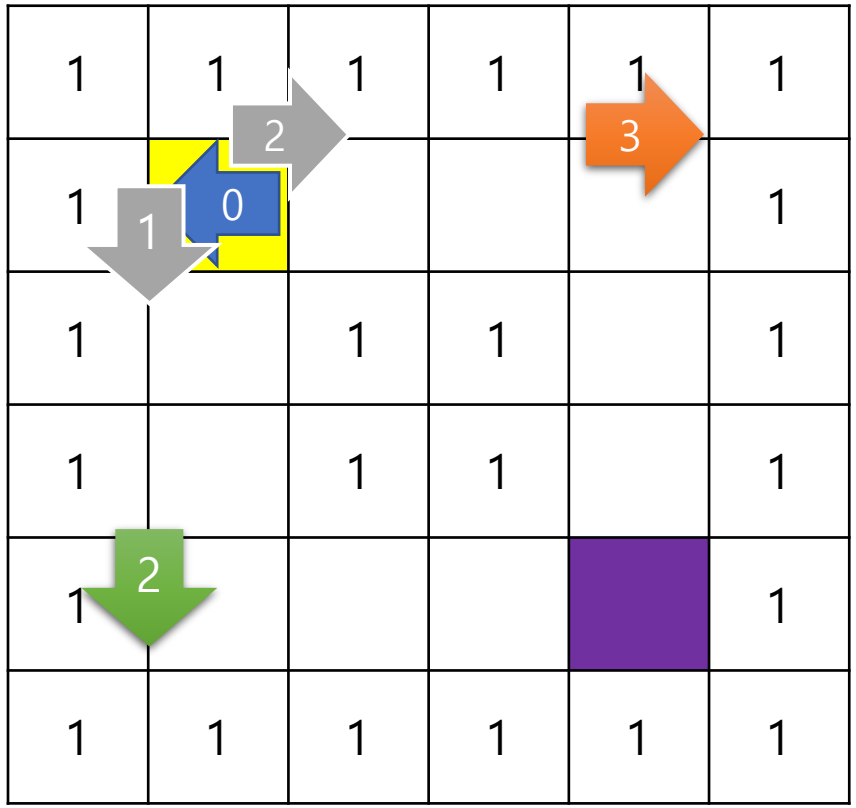
1	1	1	1	1	1
1					1
1		1	1		1
1		1	1		1
1					1
1	1	1	1	1	1

주황색을 먼저 큐에 넣었으므로
먼저 진행시키면 명령수 3에 왼쪽
그림과 같은 위치에 온다.

로봇

문제 분석

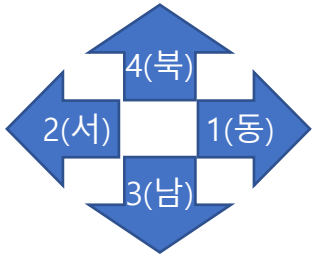




이제 녹색을 진행시키면 명령수 2에 왼쪽 그림과 같은 위치에 온다.

로봇

문제 분석



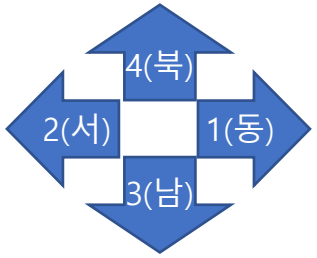
1	1	1	1	1	1
1					1
1		1	1		1
1		1	1		1
1					1
1	1	1	1	1	1

2 3 0 1 4 2

다시 주황색의 방향을 남쪽으로 회전시키면
왼쪽 그림과 같이 된다.
명령 수는 4이다.

로봇

문제 분석

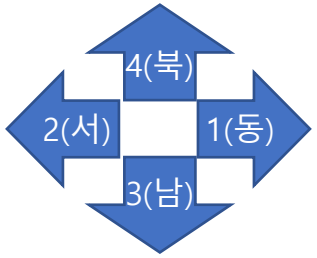


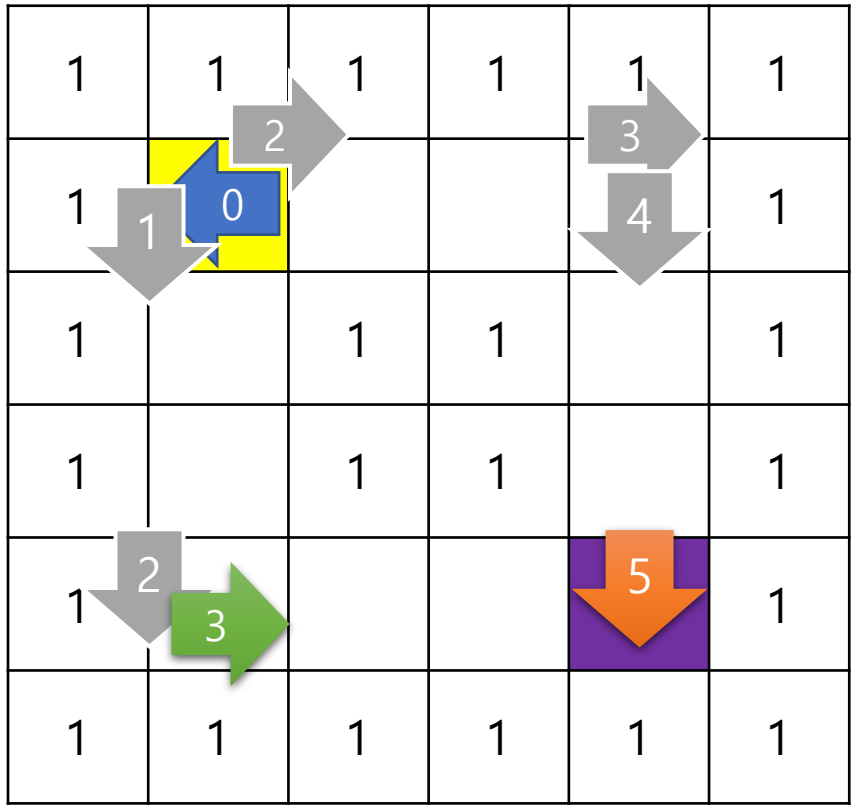
1	1	1	1	1	1
1					1
1		1	1		1
1		1	1		1
1					1
1	1	1	1	1	1

녹색의 방향을 동쪽으로 회전시키면
왼쪽 그림과 같이 된다.
명령 수는 3이다.

로봇

문제 분석

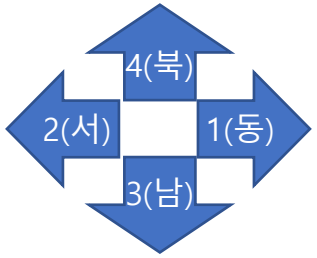


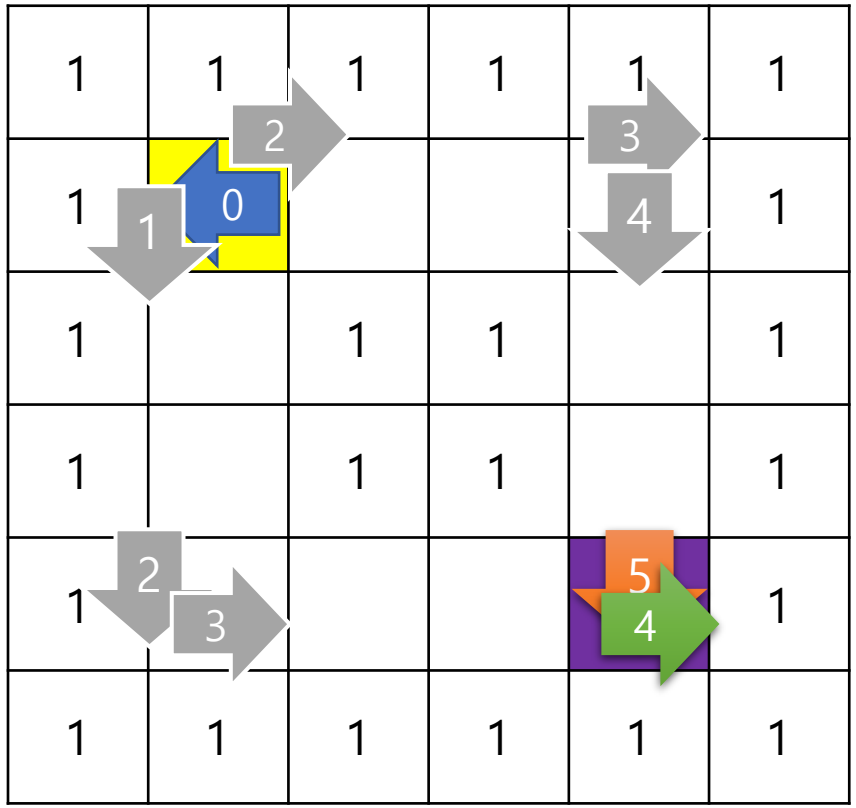


주황색로봇을 이동시키면
왼쪽 그림과 같이 된다.
명령 수는 5이다.

로봇

문제 분석

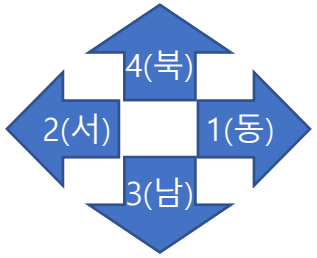


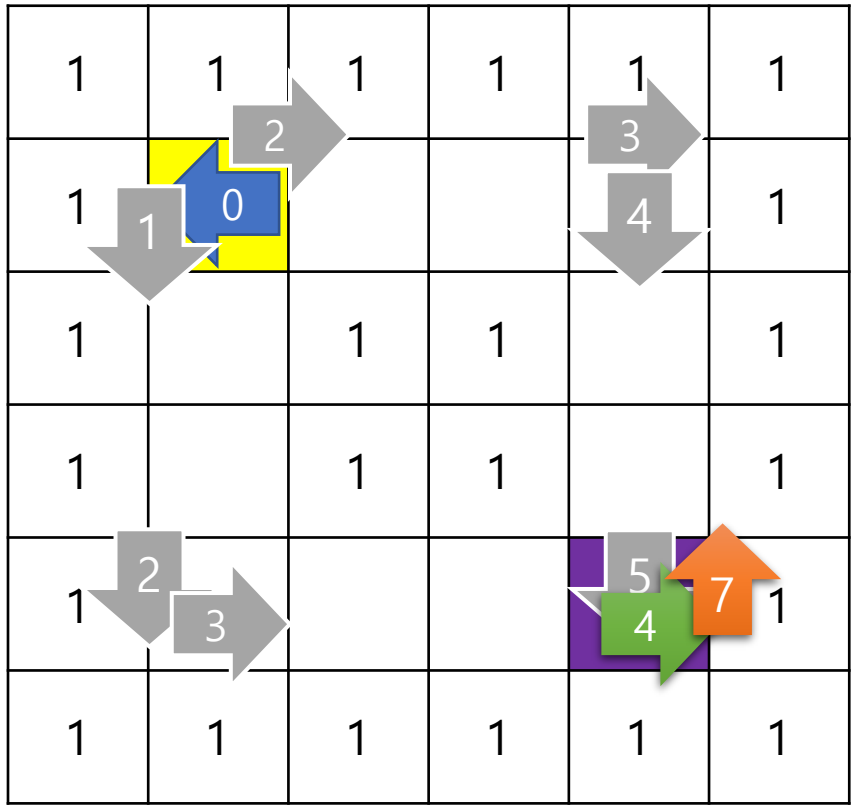


녹색로봇을 이동시키면
왼쪽 그림과 같이 된다.
명령 수는 4이다.

로봇

문제 분석



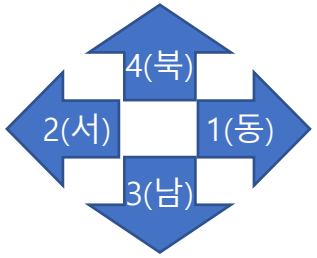


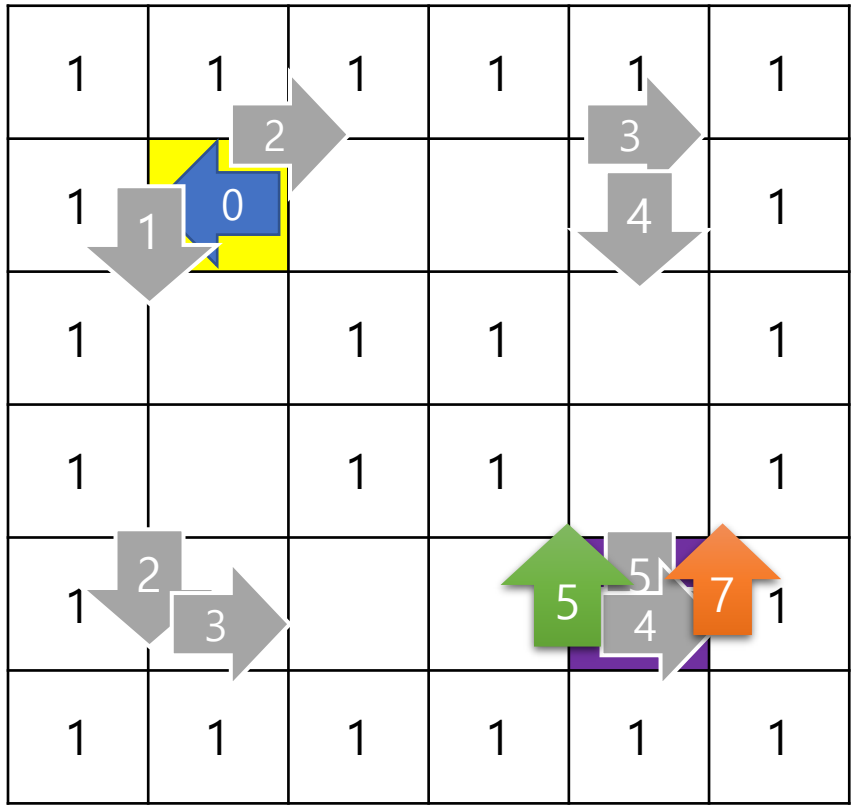
주황색로봇을 역방향(북쪽)으로 회전시키면
왼쪽 그림과 같이 된다.
명령 수는 7이다.

목표와 같은 상태이다.
그런데 이것이 최소의 명령 수 일까?

로봇

문제 분석





녹색로봇을 역방향(북쪽)으로 회전시키면
왼쪽 그림과 같이 된다.
명령 수는 5이다.

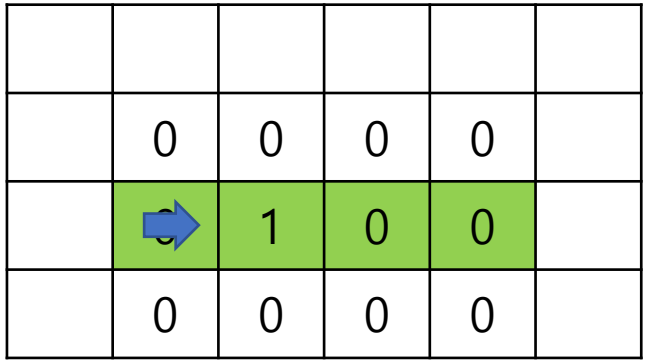
목표와 같은 상태이다.
그런데 주황색 로봇보다 명령수가 2개 적다.

그리고 이것이 최소 명령수이다.

로봇

문제 분석

- 로봇은 현재 방향으로 전진할 수 있다.
- 한 번의 명령으로 1칸 또는 2칸 또는 3칸 전진할 수 있다.
- 이때 1칸 앞이 막혀 있는 경우 2칸 또는 3칸 앞이 비어 있어도 전진하는 것이 불가능하다.



로봇

문제 분석

- 하지만 1칸 앞을 이전에 방문한 적이 있는 경우
  2칸 또는 3칸 앞을 방문할 수 있는지 확인해 봐야 한다.
- 따라서 전진이 불가능한 경우와
  이전에 방문한 적이 있는 경우는 다르게 처리되어야 한다.

로봇

해법 연구

- 문제 해결 프로세스를 정리해보자.

- BFS를 이용하여 로봇의 상태를
  명령 횟수에 따라 순차적으로 관리한다.
- 이미 만들어본 상태를 체크하기 위하여
  visit[row][col][dir]와 같은 형식을 이용할 수 있다.
- 방향을 회전하는 명령은 이웃한 방향으로만 회전한다.
  현재 방향이 1, 2라면 3, 4로 회전하는 경우만 고려한다.
  현재 방향이 3, 4라면 1, 2로 회전하는 경우만 고려한다.

로봇

해법 연구

- 전진하는 명령은 1, 2, 3칸을 하나의 명령으로 진행할 수 있는데
  1번째 칸이 막인 경우 2, 3번 칸을 고려하지 않아야 한다.

- 하지만 1번째 칸을 현재방향으로 만들어 보았다면
  2, 3번째 칸을 방문할 수 있는 경우 방문해 보아야 한다.

로봇 code example :

```
int R, C, A[105][105], visit[105][105][5];
int fr, re, sr, sc, sd, er, ec, ed;
int dr[5]={ 0, 0, 0, 1, -1 };
int dc[5]={ 0, 1, -1, 0, 0 };

struct Queue{
    int r, c, d, lev;
} que[40010];

void push( int r, int c, int d, int lev){
    if (visit[r][c][d]) return;
    visit[r][c][d] = 1;
    que[re++] = {r, c, d, lev};
}
```

로봇 code example :

```
void input(){
    scanf("%d %d", &R, &C);
    for ( int i=1; i <= R; i++) {
        for ( int j=1; j <= C; j++) {
            scanf("%d", A[i] + j);
            A[i][j] = !A[i][j];
        }
    }
    scanf("%d %d %d %d %d %d", &sr, &sc, &sd, &er, &ec, &ed);
}
```

로봇 code example :

```
int bfs(){
    push(sr, sc, sd, 0);
    for (; fr < re;++fr) {
        Queue t = que[fr];
        if (t.r==er && t.c==ec && t.d==ed) return t.lev;
        /// 1. change direction
        int nd= t.d>2? 1:3;
        push(t.r, t.c, nd, t.lev + 1);
        push(t.r, t.c, nd+1, t.lev + 1);
        /// 2. go staight forward 1,2,3
        for ( int i=1; i <= 3; i++) {
            int r=t.r + dr[t.d] * i, c= t.c + dc[t.d] * i;
            if (A[r][c] == 0) break;
            push(r, c, t.d, t.lev + 1);
        }
    }
    return 0;
}
```

로봇 code example :

```
int main(){
//    freopen("input.txt", "r", stdin);
    input();
    printf("%d \n", bfs());
    return 0;
}
```

SAMSUNG

감사합니다