4179 - 불!

난이도	골드 4

문제

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	256 MB	17103	4104	2798	22.087%

지훈이는 미로에서 일을 한다. 지훈이를 미로에서 탈출하도록 도와주자!

미로에서의 지훈이의 위치와 불이 붙은 위치를 감안해서 지훈이가 불에 타기전에 탈출할 수 있는지의 여부, 그리고 얼마나 빨리 탈출할 수 있는지를 결정해야한다.

지훈이와 불은 매 분마다 한칸씩 수평또는 수직으로(비스듬하게 이동하지 않는다) 이동한다.

불은 각 지점에서 네 방향으로 확산된다.

지훈이는 미로의 가장자리에 접한 공간에서 탈출할 수 있다.

지훈이와 불은 벽이 있는 공간은 통과하지 못한다.

입력

입력의 첫째 줄에는 공백으로 구분된 두 정수 R과 C가 주어진다. 단, $1 \leq R, C \leq 1000$ 이다. R은 미로 행의 개수, C는 열의 개수이다.

다음 입력으로 R줄동안 각각의 미로 행이 주어진다.

각각의 문자들은 다음을 뜻한다.

- #: 벽
- .: 지나갈 수 있는 공간
- J: 지훈이의 미로에서의 초기위치 (지나갈 수 있는 공간)
- F: 불이 난 공간

J는 입력에서 하나만 주어진다.

출력

지훈이가 불이 도달하기 전에 미로를 탈출 할 수 없는 경우 IMPOSSIBLE 을 출력한다.

지훈이가 미로를 탈출할 수 있는 경우에는 가장 빠른 탈출시간을 출력한다.

풀이과정

> Index of /~acm00/090613/data
>
> http://acm.student.cs.uwaterloo.ca/~acm00/090613/data/

위 링크에서 테케를 제공해줘서 많은 도움을 받았다.

이 문제는 정말 하루종일 붙잡고 있었던 것 같다. 메모리 초과도 여러 번 나고, 배열의 크기가 1000*1000일 때만 결과가 다르게 나와서 고생 꽤나 했던 문제다. 아직 골드를 풀만큼은 아닌가보다. 그래도 어떻게든 풀려서 뿌듯하긴 했다...

제출 번호	아이디	문제	결과	메모리	시간	언어	코드 길이	제출한 시간
36111560	lyuashley	4179	메모리 초과			Java 11 / 수정	3827 B	1일 전
36111543	lyuashley	4179	런타임 에러 (main class Main)			Java 11 / 수정	3845 B	1일 전
36111412	lyuashley	4179	메모리 초과			Java 11 / 수정	3684 B	1일 전
36111019	lyuashley	4179	메모리 초과			Java 11 / 수정	3998 B	1일 전
36110635	lyuashley	4179	메모리 초과			Java 11 / 수정	3484 B	1일 전
36109993	lyuashley	4179	시간 초과			Java 11 / 수정	3143 B	1일 전
36109910	lyuashley	4179	시간 초과			Java 11 / 수정	3053 B	1일 전
36109848	lyuashley	4179	시간 초과			Java 11 / 수정	3099 B	1일 전
36109761	lyuashley	4179	시간 초과			Java 11 / 수정	3222 B	1일 전
36109546	lyuashley	4179	시간 초과			Java 11 / 수정	2994 B	1일 전

삽질의 흔적 1

36113494	lyuashley	4179	틀렸습니다			Java 11 / 수정	5855 B	23시간 전
36113462	lyuashley	4179	틀렸습니다			Java 11 / 수정	5790 B	23시간 전
36113404	lyuashley	4179	런타임 에러 (NullPointer)			Java 11 / 수정	5777 B	23시간 전
36112952	lyuashley	4179	메모리 초과			Java 11 / 수정	5880 B	23시간 전
36112555	lyuashley	4179	틀렸습니다			Java 11 / 수정	5193 B	1일 전
36112472	lyuashley	4179	틀렸습니다			Java 11 / 수정	5231 B	1일 전
36112433	lyuashley	4179	틀렸습니다			Java 11 / 수정	5271 B	1일 전
36112371	lyuashley	4179	틀렸습니다			Java 11 / 수정	5313 B	1일 전
36112340	lyuashley	4179	런타임 에러 (main class Main)			Java 11 / 수정	5331 B	1일 전
36112053	lyuashley	4179	메모리 초과			Java 11 / 수정	4980 B	1일 전
36111955	lyuashley	4179	메모리 초과			Java 11 / 수정	5068 B	1일 전
36111753	lyuashley	4179	메모리 초과			Java 11 / 수정	3635 B	1일 전

삽질의 흔적 2

제출 번호	아이디	문제	결과	메모리	시간	언어	코드 길이	제출한 시간
36116589	lyuashley	4179	맞았습니다!!	73560 KB	580 ms	Java 11 / 수정	6885 B	21시간 전
36116578	lyuashley	4179	런타임 에러 (main class Main)			Java 11 / 수정	6903 B	21시간 전
36115885	lyuashley	4179	틀렸습니다			Java 11 / 수정	7767 B	22시간 전
36115678	lyuashley	4179	맞았습니다!!	71744 KB	572 ms	Java 11 / 수정	3357 B	22시간 전
36115614	lyuashley	4179	메모리 초과			Java 11 / 수정	4994 B	22시간 전
36115356	lyuashley	4179	메모리 초과			Java 11 / 수정	5003 B	22시간 전
36115346	lyuashley	4179	컴파일 에러			Java 11 / 수정	5004 B	22시간 전
36113556	lyuashley	4179	틀렸습니다			Java 11 / 수정	5919 B	23시간 전

수많은 삽질 끝에 성공

사실 아직까지도 뭐가 그렇게 메모리가 초과가 났는지 정확하게는 모르지만, 7576번의 영향으로 visited 배열을 안 쓰고 다른 데서 체크해보려다가 방문 여부가 제대로 체크가 안 되서 Queue에서

메모리 초과가 난듯했다. 방문 여부에 좀 더 초점을 두고 코드 이곳저곳을 고쳤더니 겨우 해결되었다.

소스코드

```
package BOJ4179;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.LinkedList;
import java.util.Queue;
import java.util.StringTokenizer;

class Loc {
    int x;
    int y;
    int time;

    Loc(int x, int y, int t){
        this.x = x;
        this.y = y;
        this.time = t;
    }
}

public class Main {

    static int R, C;
    static Queue<Loc> q;
    static Queue<Loc> fire;
    static int[][] maze;
    static int[] dx = {-1, 1, 0, 0};
    static int[] dy = {0 ,0 ,-1, 1};

    public static void main(String[] args) throws IOException {

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st = new StringTokenizer(br.readLine());
        R = Integer.parseInt(st.nextToken());
        C = Integer.parseInt(st.nextToken());
        maze = new int[R][C];
        q = new LinkedList<>();
        fire = new LinkedList<>();

        for(int i=0 ; i<R ; i++){
            char[] tmpArr = br.readLine().toCharArray();
            for(int j=0 ; j<C ; j++) {
                char tmp = tmpArr[j];
                if(tmp == '#') {
                    maze[i][j] = -1;
                }
                else if(tmp == 'F'){
                    maze[i][j] = -2;
                    fire.add(new Loc(i, j, 1));
                }
                else if(tmp == 'J') {
                    q.add(new Loc(i,j, 1));
```

```
                    maze[i][j] = 1;
                }
                else maze[i][j] = 0;
            }
        }

        int answer = bfs();

        // 0인 경우 impossible
        if(answer == 0)
            System.out.println("IMPOSSIBLE");
        else
            System.out.println(answer);

        br.close();

    }

    static int bfs() {
        Loc curJ, curF;
        int fireLen, jhLen;
        while(!q.isEmpty()){

            int nx, ny;
            fireLen = fire.size();
            for(int f=0 ; f<fireLen ; f++){
                curF = fire.poll();
                for(int i=0 ; i<4 ; i++){
                    nx = curF.x + dx[i];
                    ny = curF.y + dy[i];

                    if(nx < 0 || ny < 0 || nx >= R || ny >= C) continue;
                    if(maze[nx][ny] == -1 || maze[nx][ny] == -2) continue;

                    fire.add(new Loc(nx, ny, curF.time+1));
                    maze[nx][ny] = -2;
                }
            }

            jhLen = q.size();
            for(int j=0 ; j<jhLen ; j++){
                curJ = q.poll();

                for(int i=0 ; i<4 ; i++){
                    nx = curJ.x + dx[i];
                    ny = curJ.y + dy[i];

                    if(nx < 0 || ny < 0 || nx >= R || ny >= C) return curJ.time;
                    if(maze[nx][ny] == -1 || maze[nx][ny] == -2 || maze[nx][ny] == 1) continue;


                    q.add(new Loc(nx, ny, curJ.time+1));
                    maze[nx][ny] = 1;
                }
            }

        }
        return 0;
    }

}
```

💡 주요 포인트

1. 지훈이는 배열 밖으로 벗어나야 성공이다!

이전의 BFS 문제에서는 2차원 배열이 주어지고 그 안에서만 움직여야 했고, 배열 밖으로 넘어가지 않게 조건문을 걸어 스킵시켜야 했다. 불의 위치를 체크해줄 때도 적용된다.

하지만 지훈이는 지금 미로를 탈출해야 성공인 상황이다. 즉, 다음 가능한 좌표가 배열의 범위 밖일 때의 최소 시간을 구해야한다. 이 부분에서 코드가 조금 달라졌었다.

2. 매 초마다 불 또는 지훈이의 가능한 모든 위치를 체크해주고 갱신해야 한다.

여기서 제일 많이 삽질했던 것 같다. 처음에는 아무 생각 없이 큐를 하나만 만들고 해당 큐가 비어있을 때까지 while문을 돌렸다. 결과는 당연히 메모리 초과.

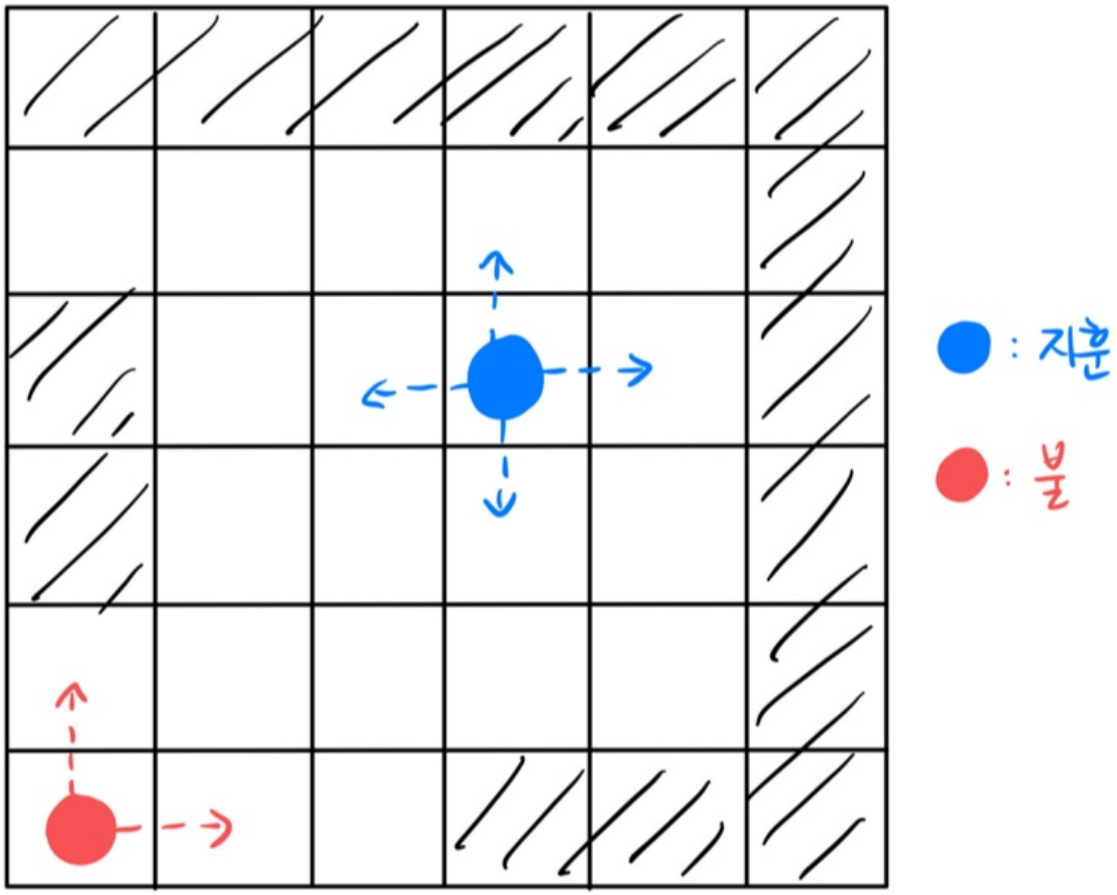
그래서 불의 큐와 지훈이의 큐를 따로 만들어서 구현도 해봤는데, 방문여부를 이상하게 체크해주고 있어서 여기서도 삽질을 많이 했다.

그러다가 다음과 같이 코드를 짜봤다.

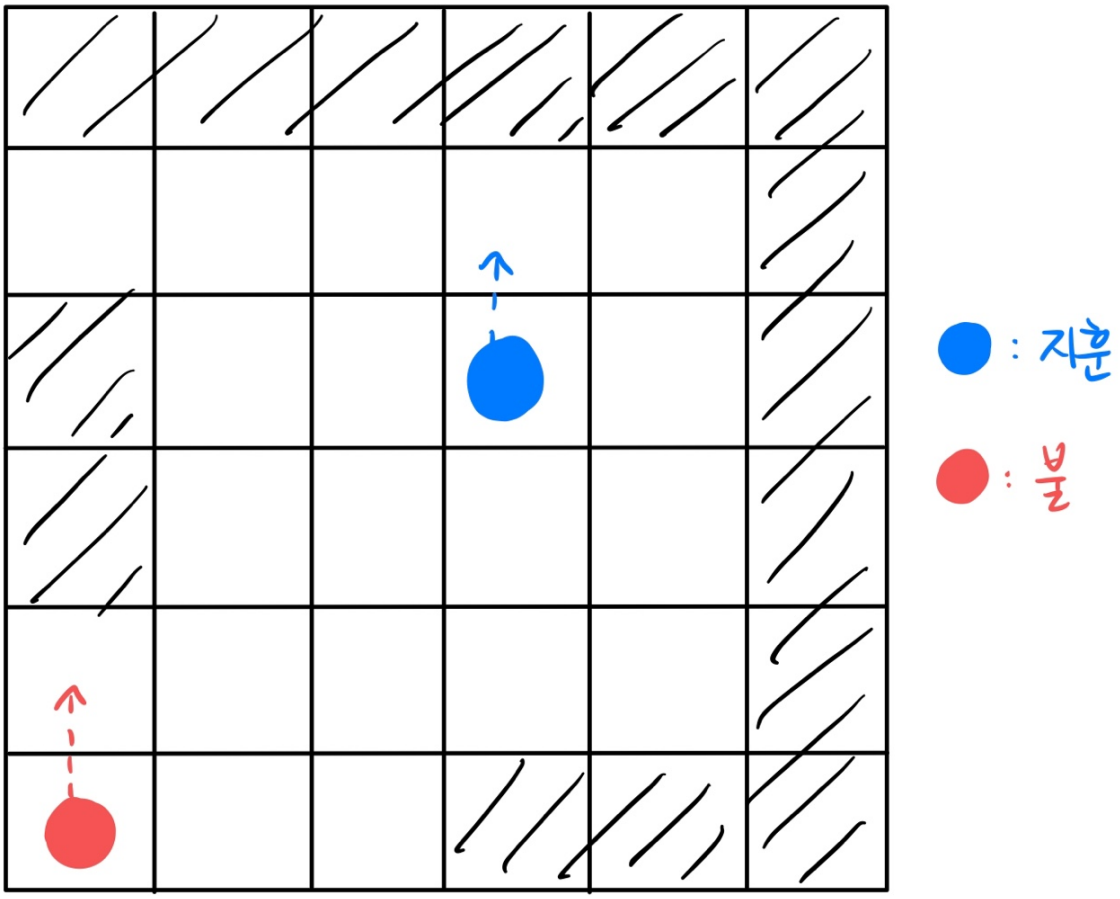
```
while(!q.isEmpty()) {
    int nx, ny;
    Loc curJ;   // 현재 지훈이의 위치
    for(int i=0 ; i<4 ; i++) {
        nx = curJ.x + dx[i];
        ny = curJ.y + dy[i];

        if(nx < 0 || ny < 0 || nx >= R || ny >= C) return curJ.time;
        if(maze[nx][ny] == -1 || maze[nx][ny] == -2 || maze[nx][ny] == 1) continue;

        q.add(new Loc(nx, ny, curJ.time+1));
        maze[nx][ny] = 1;
    }

    Loc curF;   // 현재 불의 위치
    for(int i=0 ; i<4 ; i++) {
        nx = curF.x + dx[i];
        ny = curF.y + dy[i];

        if(nx < 0 || ny < 0 || nx >= R || ny >= C) continue;
        if(maze[nx][ny] == -1 || maze[nx][ny] == -2) continue;

        fire.add(new Loc(nx, ny, curF.time+1));
        maze[nx][ny] = 1;
    }

}
```

위와 같이 코드를 짰더니, **특정 시간(초)에서의 가능한 불 또는 지훈이의 위치가 모두 갱신되어야 하는데, 하나만 갱신된 채 그 다음 단계로** 넘어갔다.

- 의도했던 바



- 현실



그러면 해당 시간에 큐에 들어있는 만큼 갱신을 시켜줘야 한다는 건데,

그렇다면 해당 시간에 큐의 사이즈를 따로 저장해두고 그 사이즈만큼 반복문을 돌려 pop을 해주면 되지 않을까?

해서 나온 코드가 현재 코드였고, 결과는 성공이었다.

사담

설명에서는 삽질을 한 부분에 대해 많이 언급은 안 해서 금방 풀린 것처럼 보일 수 있겠지만...위의 제출 사진에서 보다시피 삽질을 엄청 많이 했던 문제였다. 오기가 생겨서 계속 붙잡고 있었는데 결국 **맞았습니다!!** 라는 문구를 보고야 말았다. (뿌듯) 아직 골드는 좀 힘들지만, 그래도 아예 못 풀 정도는 아닌 것 같아 더 도전해보련다.