백준 6087번 - 레이저 통신

시간제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1초	128MB	8352	2869	1976	32.135%

문제

크기가 1×1인 정사각형으로 나누어진 W×H 크기의 지도가 있다. 지도의 각 칸은 빈 칸이거나 벽이며, 두 칸은 '`C`'로 표시되어 있는 칸이다.

'`C`'로 표시되어 있는 두 칸을 레이저로 통신하기 위해서 설치해야 하는 거울 개수의 최솟값을 구하는 프로그램을 작성하시오. 레이저로 통신한다는 것은 두 칸을 레이저로 연결할 수 있음을 의미한다.

레이저는 C에서만 발사할 수 있고, 빈 칸에 거울('`/`', '`\`')을 설치해서 방향을 90도 회전시킬 수 있다.

아래 그림은 H = 8, W = 7인 경우이고, 빈 칸은 '`.`', 벽은 '`*`'로 나타냈다. 왼쪽은 초기 상태, 오른쪽은 최소 개수의 거울을 사용해서 두 '`C`'를 연결한 것이다.

```
7 . . . . . .          7 . . . . . .
6 . . . . . C          6 . . . . /-C
5 . . . . . *          5 . . . . | *
4 * * * * . *          4 * * * * | *
3 . . . . * .          3 . . . * | .
2 . . . . * .          2 . . . * | .
1 . C . . * .          1 . C . . * | .
0 . . . . . .          0 . \-------/ .
  0 1 2 3 4 5 6            0 1 2 3 4 5 6
```

입력

첫째 줄에 W와 H가 주어진다. (1 ≤ W, H ≤ 100)

둘째 줄부터 H개의 줄에 지도가 주어진다. 지도의 각 문자가 의미하는 것은 다음과 같다.

- `.`: 빈 칸
- `*`: 벽
- `C`: 레이저로 연결해야 하는 칸

'`C`'는 항상 두 개이고, 레이저로 연결할 수 있는 입력만 주어진다.

출력

첫째 줄에 C를 연결하기 위해 설치해야 하는 거울 개수의 최솟값을 출력한다.

예제 입력1

```
7 8
.......
......C
......*
*****.*
....*..
....*..
.C..*..
.......
```

예제 출력1

```
3
```

출처

출처

알고리즘 분류

- 그래프 이론
- 그래프 탐색
- 너비 우선 탐색
- 다익스트라
- 0-1 너비 우선 탐색

접근 방법

각각의 cell 들을 그래프의 정점의 관점으로 본다면 `.`과 `.`사이에 edge가 존재한다고 볼 수 있다. 이 때 정점에서 뻗어나가는 간선은 최대 4개(상,하,좌,우)가 될 것이고 현재 좌표에 `/`나 `\`가 존재할 수도 있으므로 현재 방향에서 수직으로 꺾이는(?) 간선에 1이라는 가중치를 적용해준다.

이렇게 할 경우, 최단 경로를 찾는다는 관점에서 특정 좌표로 부터 시작하여 나머지 정점들 까지 도착하기 위한 최소 거울 수와 1대1로 매핑된다는 것을 알 수 있다.

해당 좌표까지 최단 경로로 탐색을 한다는 가정하에, 특정 좌표를 방문할 때의 최단 경로가 현재 탐색을 진행하고 있는 경로보다 더 거리가 길 경우 해당 경로로 탐색하는 것은 더 이상 무의미하므로 skip해주는 방식으로 진행한다.

간선의 가중치는 0 또는 1이고 오로지 탐색을 진행하고 있는 방향이 변할 경우에만 1의 가중치를 가지므로 특정 좌표 `[i][j]`에 도착할 때 `dist[i][j]` 값이 현재 탐색 중인 경로와 같다는 것은 이전에 다른 방향으로 탐색을 진행하여 동일한 가중치로 `dist[i][j]`에 도달했다는 것이므로 이 경우는 생략해서는 안된다.

→ 이렇게 탐색을 진행해도 사이클이 절대 발생하지 않는 이유는 탐색 중인 간선이 방향을 수직으로 꺾을 때에만 가중치가 1이 되기 때문이다.

소스코드

```cpp
#define FASTIO cin.tie(0)->sync_with_stdio(false), cout.tie(0)
#include <bits/stdc++.h>
using namespace std;
int W, H;
char board[105][105];
vector<pair<int,int>> pos;
const int dir[4][2] = {{-1, 0}, {0, 1}, {1, 0}, {0, -1}};
int dist[105][105];
int main(void){
    FASTIO;
    cin >> W >> H;
    for(int i = 0; i<H; i++){
        for(int j=0; j<W; j++){
            cin >> board[i][j];
            if(board[i][j] == 'C')
                pos.emplace_back(i, j);
        }
    }

    for(int i=0; i<105; i++){
        for(int j=0; j<105; j++)
            dist[i][j] = 1e9;
    }
    typedef tuple<int, int, int, int> tp; // {cost, y, x, direction}
    priority_queue<tp, vector<tp>, greater<tp>> PQ;
    for(int i=0; i<4; i++)
        PQ.push({0, pos[0].first, pos[0].second, i});

    dist[pos[0].first][pos[0].second] = 0;
    while(!PQ.empty()){
        auto [mirror, y, x, d] = PQ.top(); PQ.pop();
```

```
        int nd[3] = {d, (d+1)%4, (d+3)%4};
        int nm[3] = {mirror, mirror+1, mirror+1};
        for(int i=0; i<3; i++){
            int ny = y + dir[nd[i]][0], nx = x + dir[nd[i]][1];
            if(ny < 0 || nx < 0 || ny >= H || nx >= W || board[ny][nx] == '*')
continue;
            if(dist[ny][nx] < nm[i]) continue;
            dist[ny][nx] = nm[i];
            PQ.push({nm[i], ny, nx, nd[i]});
        }
    }
    cout << dist[pos[1].first][pos[1].second];
    return 0;
}
```