백준 3109번 - 빵집

백준 3109번 - 빵집

시간제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1초	256MB	15889	5491	3631	31.053%

문제

유명한 제빵사 김원웅은 빵집을 운영하고 있다. 원웅이의 빵집은 글로벌 재정 위기를 피해가지 못했고, 결국 심각한 재정 위기에 빠졌다.

원웅이는 지출을 줄이고자 여기저기 지출을 살펴보던 중에, 가스비가 제일 크다는 것을 알게되었다. 따라서 원웅이는 근처 빵집의 가스관에 몰래 파이프를 설치해 훔쳐서 사용하기로 했다.

빵집이 있는 곳은 R*C 격자로 표현할 수 있다. 첫째 열은 근처 빵집의 가스관이고, 마지막 열은 원웅이의 빵집이다.

원웅이는 가스관과 빵집을 연결하는 파이프를 설치하려고 한다. 빵집과 가스관 사이에는 건물이 있을 수도 있다. 건물이 있는 경우에는 파이프를 놓을 수 없다.

가스관과 빵집을 연결하는 모든 파이프라인은 첫째 열에서 시작해야 하고, 마지막 열에서 끝나야 한다. 각 칸은 오른쪽, 오른쪽 위 대각선, 오른쪽 아래 대각선으로 연결할 수 있고, 각 칸의 중심끼리 연결하는 것이다.

원웅이는 가스를 되도록 많이 훔치려고 한다. 따라서, 가스관과 빵집을 연결하는 파이프라인을 여러 개 설치할 것이다. 이 경로는 겹칠 수 없고, 서로 접할 수도 없다. 즉, 각 칸을 지나는 파이프는 하나이어야 한다.

원웅이 빵집의 모습이 주어졌을 때, 원웅이가 설치할 수 있는 가스관과 빵집을 연결하는 파이프라인의 최대 개수를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 R과 C가 주어진다. (1 ≤ R ≤ 10,000, 5 ≤ C ≤ 500)

다음 R개 줄에는 빵집 근처의 모습이 주어진다. '.'는 빈 칸이고, 'x'는 건물이다. 처음과 마지막 열은 항상 비어있다.

출력

첫째 줄에 원웅이가 놓을 수 있는 파이프라인의 최대 개수를 출력한다.

예제 입력1

```
5 5
.xx..
..x..
.....
...x.
...x.
```

예제 출력1

```
2
```

출처

출처

알고리즘 분류

- 그래프 이론
- 그리디 알고리즘
- 그래프 탐색
- 깊이 우선 탐색

접근 방법

`파이프라인의 경로는 서로 겹칠 수 없고 접할 수도 없다` 라는 문제의 조건에 따라, 아래와 같이 `i행 0열` 이 `k행 c-1열` 에 연결 될 경우 `i+1행 0열` 에서는 `k+1행 c-1열` 부터 연결될 수 있다.

`i행 0열` 에서 `k행 c-1열`에 연결될 수 있는 k값들이 여러개 있는 경우, 연결할 수 있는 k_{min} 을 선택하는 것이 추후의 선택에 있어 유리하다는 것을 보여준다.

여기서 또 다른 의문이 생길 수 있다.

`i행 0열`에서 `k행 c-1열` 을 연결할 수 있음에도 불구하고, 연결하지 않으므로서 최적해를 구성하는 방법이 없을까?

즉, 아래와 같이 `1행 0열`에서 연결될 수 있는 `4행 c-1열` 을 연결하지 않고 진행하였을 때, `(3,4)행 0열`에서 연결할 수 있는 파이프의 수가 더 많을 수 있을까?

결론부터 말하자면, 위와 같은 경우는 존재할 수 없다.

`1행 0열`에서 `(1,2,3)행 c-1열` 에 연결할 수 없었다는 것은 `1행 0열` ~ `k행 c-1`열 까지 **모든 경로**를 탐색하였을 때에 k에 도달하지 못했음을 시사하고, 이는 2,3,4 행에서 탐색을 진행하더라도 `[i][j]` (i, j는 k에 도달하기 위한 중간 경로)를 거칠 수 없다는 뜻이므로 2,3,4행에서도 도달이 불가능하다는 것을 의미하기 때문이다.

그렇다면, `1행 0열` ~ `R행 0열` 까지 순차적으로 DFS로 도달 가능한 경로를 찾되, 그 중 도달 가능한 `k행 c-1열`중 최소의 k에 매핑시키는 것이 최적해라는 것을 알 수 있다.

`(r-1, c+1)`, `(r, c+1)`, `(r+1, c+1)` 방향으로 탐색할 경우, 가장 먼저 도달하는 지점이 k_{min} 임을 보장할 수 있으므로 순서를 지키며 dfs를 수행한다.

소스코드

```cpp
#define FASTIO cin.tie(0)->sync_with_stdio(false), cout.tie(0)
////////////////////////////////////////////////////////////////
#include <bits/stdc++.h>
using namespace std;
int R, C;
char board[10001][501];
const int dir[3][2] = {{-1, 1}, {0, 1}, {1, 1}};
bool visited[10001][501] = {false,};

bool dfs(int r, int c) {
    if (c == C-1)
        return true;
    for (int d = 0; d < 3; d++) {
        int nr = r + dir[d][0], nc = c + dir[d][1];
        if (nr < 0 || nc < 0 || nr >= R || nc >= C || board[nr][nc] == 'x' ||
visited[nr][nc]) {
            continue;
        }
        visited[nr][nc] = true;
        if (dfs(nr, nc))
            return true;
    }
    return false;
}
int main(void){
    FASTIO;
////////////////////////////////////////////////////////////////
    cin >> R >> C;
    for (int i = 0; i < R; i++) {
        for (int j = 0; j < C; j++) {
            cin >> board[i][j];
        }
    }
    int ans = 0;
    for (int i = 0; i < R; i++) {
        ans += dfs(i, 0);
    }
    cout << ans ;
    return 0;
}
```