벽 부수고 이동하기

문제

N×M의 행렬로 표현되는 맵이 있다. 맵에서 0은 이동할 수 있는 곳을 나타내고, 1은 이동할 수 없는 벽이 있는 곳을 나타낸다. 당신은 (1, 1)에서 (N, M)의 위치까지 이동하려 하는데, 이때 최단 경로로 이동하려 한다. 최단경로는 맵에서 가장 적은 개수의 칸을 지나는 경로를 말하는데, 이때 시작하는 칸과 끝나는 칸도 포함해서 센다.

만약에 이동하는 도중에 한 개의 벽을 부수고 이동하는 것이 좀 더 경로가 짧아진다면, 벽을 한 개 까지 부수고 이동하여도 된다.

한 칸에서 이동할 수 있는 칸은 상하좌우로 인접한 칸이다.

맵이 주어졌을 때, 최단 경로를 구해 내는 프로그램을 작성하시오.

입력

첫째 줄에 N(1 ≤ N ≤ 1,000), M(1 ≤ M ≤ 1,000)이 주어진다. 다음 N개의 줄에 M개의 숫자로 맵이 주어진다. (1, 1)과 (N, M)은 항상 0이라고 가정하자.

출력

첫째 줄에 최단 거리를 출력한다. 불가능할 때는 -1을 출력한다.

Time Limit 2 sec *Memory Limit* 192 MB

Test Cases

Input	*Output*
6 4 0100 1110 1000 0000 0111 0000	15
4 4 0111 1111 1111 1110	-1

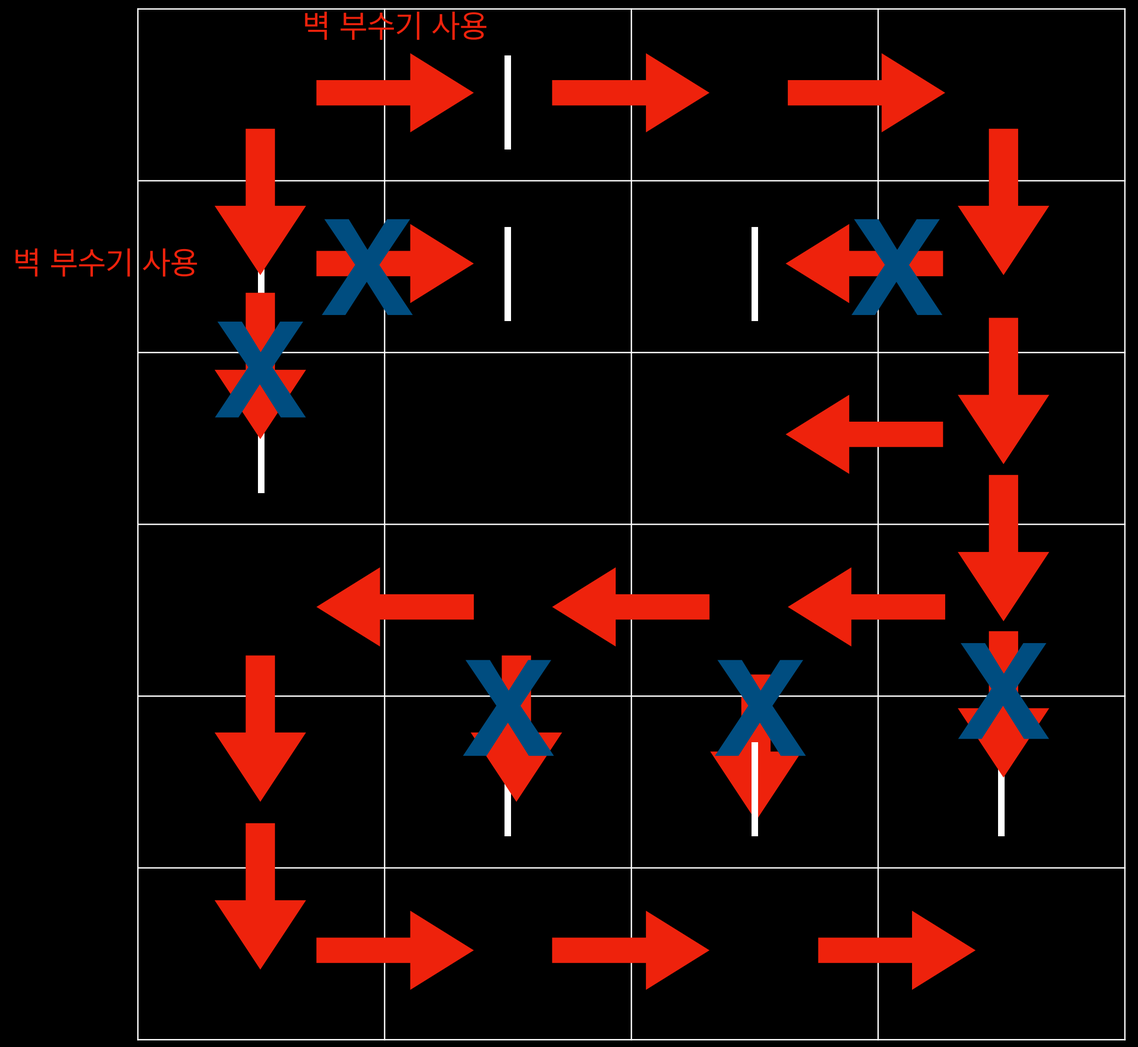
Approach

생각보다 쉽지 않은 문제라고 생각합니다. 처음에 문제를 읽고 단순하게 flag를 둬서 문제를 접근하면 잘못된 결과를 낼 수도 있다고 생각했습니다.

1. 그래프를 문자열로 구현하자, 문자가 더 적은 용량을 차지하기 때문
2. 상하좌우 이동 구현
3. 최단 경로? -> BFS로 해결하자
4. 그래프 문제에서 가장 중요한 OOB를 잘 체크하자
5. 벽을 부수기 스킬을 시전하기 전, 벽을 부수기 스킬을 시전한 후 -> 2가지 case로 나누어 도착

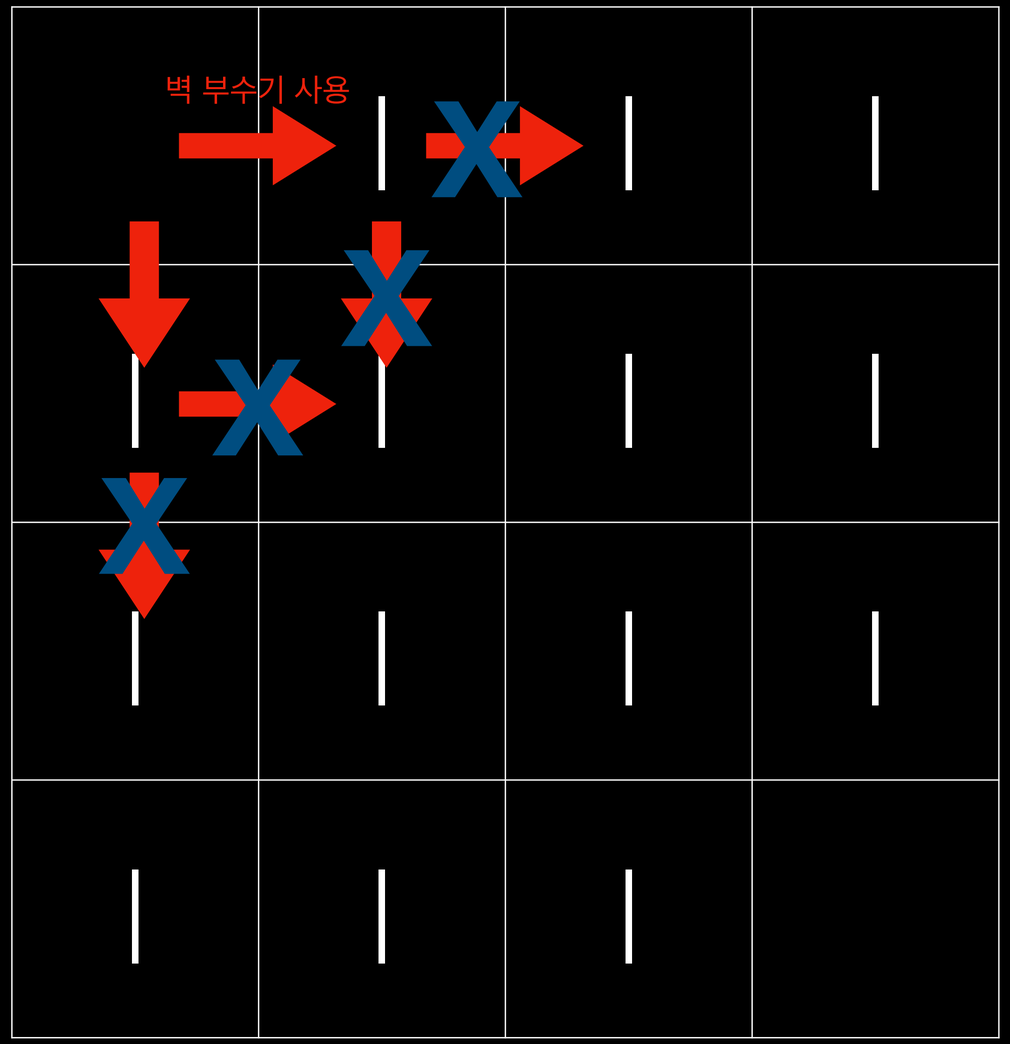
Case1

Output	*Input*
6 4	
0100	
1110	
1000	15
0000	
0111	
0000	



Case1

Output	*Input*
4 4 0111 1111 1111 1110	-1



Code

```python
import sys
from collections import deque
input = sys.stdin.readline

N, M = map(int, input().split())
graph = [input() for _ in range(N)]
visited = [[[-1 for _ in range(2)] for _ in range(M)] for _ in range(N)]
dx = [0, 0, -1, 1]  # 좌, 우, 하, 상
dy = [-1, 1, 0, 0]

# 범위 체크 함수
def checkRange(rx, ry):
    if 0 <= rx < N and 0 <= ry < M:
        return True
    return False
```

Code

```python
def bfs(x, y):
    visited[x][y][0] = 1
    q = deque([])
    q.append([x, y, 0])  # 벽을 부순뒤에 visited[][][1] 로 체크한다.
    while q:
        x, y, flag = q.popleft()
        for i in range(4):
            rx, ry = x + dx[i], y + dy[i]
            if checkRange(rx, ry): # 범위 체크
                # 벽 부수기 쓰기전 + 벽 부수기 시전 후
                if graph[rx][ry] == '0' and visited[rx][ry][flag] == -1:
                    visited[rx][ry][flag] = visited[x][y][flag] + 1
                    q.append([rx, ry, flag])
                # 벽 부수기 전에 벽을 부숴야 하는 상황
                if graph[rx][ry] == '1' and flag == 0 and visited[rx][ry][flag+1] == -1:
                    visited[rx][ry][flag+1] = visited[x][y][flag] + 1
                    q.append([rx, ry, 1])
```

Code

```python
# 벽 부수기 스킬 사용, 미사용 둘다 방문 못한다면 목표 지점까지 도달 불가
if visited[N-1][M-1] == [-1, -1]:
    print(-1)
else:
    # 벽 부수기 스킬 사용
    if visited[N-1][M-1][0] == -1:
        print(visited[N-1][M-1][1])
    # 벽 부수기 스킬 사용 x
    elif visited[N-1][M-1][1] == -1:
        print(visited[N-1][M-1][0])
    # 둘다 도달한 경우, 둘 중 작은 값 출력
    else:
        print(min(visited[N-1][M-1]))
```