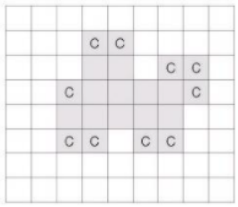
치즈

치즈  ☆

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초	128 MB	8252	3691	2756	45.359%

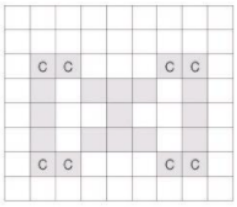
문제

N×M (5≤N, M≤100)의 모눈종이 위에 아주 얇은 치즈가 <그림 1>과 같이 표시되어 있다. 단, N 은 세로 격자의 수이고, M 은 가로 격자의 수이다. 이 치즈는 냉동 보관을 해야만 하는데 실내온도에 내어놓으면 공기와 접촉하여 천천히 녹는다. 그런데 이러한 모눈종이 모양의 치즈에서 각 치즈 격자(작 은 정사각형 모양)의 4변 중에서 적어도 2변 이상이 실내온도의 공기와 접촉한 것은 정확히 한시간만에 녹아 없어져 버린다. 따라서 아래 <그림 1> 모양과 같은 치즈(회색으로 표시된 부분)라면 C로 표시된 모든 치즈 격자는 한 시간 후에 사라진다.

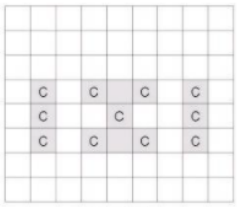


<그림 1>

<그림 2>와 같이 치즈 내부에 있는 공간은 치즈 외부 공기와 접촉하지 않는 것으로 가정한다. 그러므 로 이 공간에 접촉한 치즈 격자는 녹지 않고 C로 표시된 치즈 격자만 사라진다. 그러나 한 시간 후, 이 공간으로 외부공기가 유입되면 <그림 3>에서와 같이 C로 표시된 치즈 격자들이 사라지게 된다.



<그림 2>



<그림 3>

모눈종이의 맨 가장자리에는 치즈가 놓이지 않는 것으로 가정한다. 입력으로 주어진 치즈가 모두 녹아 없어지는데 걸리는 정확한 시간을 구하는 프로그램을 작성하시오.

입력

첫째 줄에는 모눈종이의 크기를 나타내는 두 개의 정수 N, M (5≤N, M≤100)이 주어진다. 그 다음 N개의 줄에는 모눈종이 위의 격자에 치즈가 있는 부분은 1로 표시되고, 치즈가 없는 부분은 0으로 표시된다. 또한, 각 0과 1은 하나의 공백으로 분리되어 있다.

문제에서 보는 키워드

1. "모눈종이의 맨 가장자리에는 치즈가 놓이지 않는다고 가정한다."

(0,0) 부터 BFS를 시작하여, 공기를 만나면 -1로 바꿔주고, 치즈 (1) 을 만나면 +1을 해준다.

Case 1. -1 로 되어있는 칸

=> 외부 공기 -> 다시 0으로 초기화 시켜준다.

Case 2. 0 으로 되어있는 칸

=> 내부 공기 -> 변경 없음

Case 3. 1 or 2로 되어있는 칸

외부 공기가 0칸 혹은 1칸이므로 살아남는 치즈

-> 다시 1로 초기화 시켜준다.

Case 4. 2 이상으로 되어있는 칸

외부 공기 면적이 2칸 이상이 되는 것이므로 0으로 없앰

코드

```
package Algo_Study_BOJ;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

public class Main_2638_치즈 {
  static int[][] map;
  static int N, M;
  static int[] dx = new int[] { -1, 1, 0, 0 };
  static int[] dy = new int[] {  0, 0, 1, -1};

  private static boolean isAllClear() {
    int cnt = 0;
    for(int i = 0; i < N; i++) {
      for(int j = 0; j < M; j++) {
        if(map[i][j] == 1 || map[i][j] == 2) {
          map[i][j] = 1;
          cnt++;
        }
        else {
          map[i][j] = 0;
        }
      }
    }
    if(cnt == 0) {
      return true;
    }
    return false;
  }

  private static void dfs(int r, int c) {

    map[r][c] = -1;
```

```
    for(int i = 0; i < 4; i++) {
      int nr = r + dx[i];
      int nc = c + dy[i];

      if(nr >= 0 && nr < N && nc >= 0 && nc < M) {
        if(map[nr][nc] == 0) {
          dfs(nr, nc);
        }
        else if(map[nr][nc] > 0) {
          map[nr][nc]++;
        }
      }
    }

  }
  public static void main(String[] args) throws Exception {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    StringTokenizer str = new StringTokenizer(br.readLine());

    N = Integer.parseInt(str.nextToken());
    M = Integer.parseInt(str.nextToken());

    map = new int[N][M];
    for(int i = 0; i < N; i++) {
      str = new StringTokenizer(br.readLine());
      for(int j = 0; j < M; j++) {
        map[i][j] = Integer.parseInt(str.nextToken());
      }
    }
    int answer = 0;
    while(!isAllClear()) {
      dfs(0, 0);
      answer++;
    }
    System.out.println(answer);
  }
}
```