BOJ 7569 : 토마토

beberiche · 어제

BFS Java algolithms boj

algolithms

목록 보기 5/6

문제

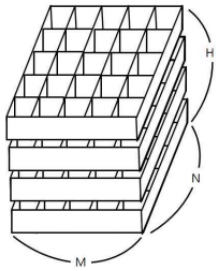
7569번 제출 맞힌 사람 숏코딩 재채점 결과 채점 현황 내 제출 난이도 기여 질문 검색

토마토 성공

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	256 MB	60230	24722	18129	41.249%

문제

철수의 토마토 농장에서는 토마토를 보관하는 큰 창고를 가지고 있다. 토마토는 아래의 그림과 같이 격자모양 상자의 칸에 하나씩 넣은 다음, 상자들을 수직으로 쌓아 올려서 창고에 보관한다.



창고에 보관되는 토마토들 중에는 잘 익은 것도 있지만, 아직 익지 않은 토마토들도 있을 수 있다. 보관 후 하루가 지나면, 익은 토마토들의 인접한 곳에 있는 익지 않은 토마토들은 익은 토마토의 영향을 받아 익게 된다. 하나의 토마토에 인접한 곳은 위, 아래, 왼쪽, 오른쪽, 앞, 뒤 여섯 방향에 있는 토마토를 의미한다. 대각선 방향에 있는 토마토들에게는 영향을 주지 못하며, 토마토가 혼자 저절로 익는 경우는 없다고 가정한다. 철수는 창고에 보관된 토마토들이 며칠이 지나면 다 익게 되는지 그 최소 일수를 알고 싶어 한다.

토마토를 창고에 보관하는 격자모양의 상자들의 크기와 익은 토마토들과 익지 않은 토마토들의 정보가 주어졌을 때, 며칠이 지나면 토마토들이 모두 익는지, 그 최소 일수를 구하는 프로그램을 작성하라. 단, 상자의 일부 칸에는 토마토가 들어있지 않을 수도 있다.

입력

입력

첫 줄에는 상자의 크기를 나타내는 두 정수 M,N과 쌓아올려지는 상자의 수를 나타내는 H가 주어진다. M은 상자의 가로 칸의 수, N은 상자의 세로 칸의 수를 나타낸다. 단, 2 ≤ M ≤ 100, 2 ≤ N ≤ 100, 1 ≤ H ≤ 100 이다. 둘째 줄부터는 가장 밑의 상자부터 가장 위의 상자까지에 저장된 토마토들의 정보가 주어진다. 즉, 둘째 줄부터 N개의 줄에는 하나의 상자에 담긴 토마토의 정보가 주어진다. 각 줄에는 상자 가로줄에 들어있는 토마토들의 상태가 M개의 정수로 주어진다. 정수 1은 익은 토마토, 정수 0 은 익지 않은 토마토, 정수 -1은 토마토가 들어있지 않은 칸을 나타낸다. 이러한 N개의 줄이 H번 반복하여 주어진다.

토마토가 하나 이상 있는 경우만 입력으로 주어진다.

출력

여러분은 토마토가 모두 익을 때까지 최소 며칠이 걸리는지를 계산해서 출력해야 한다. 만약, 저장될 때부터 모든 토마토가 익어있는 상태이면 0을 출력해야 하고, 토마토가 모두 익지는 못하는 상황이면 -1을 출력해야 한다.

예제 입력 1 복사

```
5 3 1
0 -1 0 0 0
-1 -1 0 1 1
0 0 0 1 1
```

예제 출력 1 복사

```
-1
```

풀이과정

전형적인 BFS 문제, 단 시작해야하는 지점이 여러곳이며, 3차원 배열이라는 점을 명심하고 푼다면 쉽게 풀 수 있을 것이다.

일단 시작해야하는 부분들을 먼저 Queue 에 넣고 나서 돌리기 시작하는 방식으로 진행했고, 토마토가 익은 경우에는 기존의 값에서1씩 확대해서 진행하는 식으로 했다. 한번의 탐색이 끝난 경우에도 0 이 존재한다면, 익지 않은 토마토가 있다고 판단하였고, 아닌 경우에는 해당 삼차원 배열안에 가장 큰 값이 바로 토마토 모두 익은 날짜의 값이 된다.

정답

```
import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

public class Main {
    static int[][][] map;
    static boolean[][][] visited;
    static Queue<int[]> q;
    static final int[] DI = new int[]{0, 0, 0, 0, -1, 1};
    static final int[] DJ = new int[]{0, 0, -1, 1, 0, 0};
    static final int[] DK = new int[]{-1, 1, 0, 0, 0, 0};
    static int H, M, N;

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        N = sc.nextInt();
        M = sc.nextInt();
        H = sc.nextInt();

        map = new int[H][M][N];

        for (int i = 0; i < H; i++) {
            for (int j = 0; j < M; j++) {
                for (int k = 0; k < N; k++) {
                    map[i][j][k] = sc.nextInt();
                }
            }
        }

        visited = new boolean[H][M][N];
        q = new LinkedList<>();
        for (int i = 0; i < map.length; i++) {
            for (int j = 0; j < map[0].length; j++) {
                for (int k = 0; k < map[0][0].length; k++) {
                    if (map[i][j][k] == 1) {
                        visited[i][j][k] = true;
                        q.add(new int[]{i, j, k});
                    }
                }
            }
        }
```

```
            BFS();
            System.out.println(check());
        }

        public static void BFS() {
            while (!q.isEmpty()) {
                int[] curr = q.poll();
                int currI = curr[0];
                int currJ = curr[1];
                int currK = curr[2];
                for (int d = 0; d < 6; d++) {
                    int ni = currI + DI[d];
                    int nj = currJ + DJ[d];
                    int nk = currK + DK[d];

                    if (ni < 0 || nj < 0 || nk < 0 || ni >= H || nj >= M || nk >= N) continue;

                    if (visited[ni][nj][nk]) continue;

                    if (map[ni][nj][nk] != 0) continue;

                    visited[ni][nj][nk] = true;
                    map[ni][nj][nk] = map[currI][currJ][currK] + 1;
                    q.add(new int[]{ni, nj, nk});
                }
            }
        }

        public static int check() {
            int day = 1;
            for (int i = 0; i < H; i++) {
                for (int j = 0; j < M; j++) {
                    for (int k = 0; k < N; k++) {
                        if (map[i][j][k] == 0) {
                            return -1;
                        } else {
                            day = Math.max(day, map[i][j][k]);
                        }
                    }
                }
            }
            return day-1;
        }
    }
```

벨

새로운 것에 관심이 많고, 프로젝트 설계 및 최적화를 좋아합니다.

다음 포스트
BOJ 14500 : 테트로미오

이전 포스트
Programmers : 귤 고르기

0개의 댓글

댓글을 작성하세요

댓글 작성