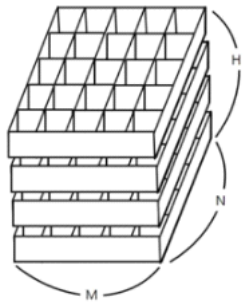
7569 토마토

2021년 4월 8일 목요일 오후 4:17

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

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

0 토마토 1 익토 -1 토마토X

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

XYZ

예제 출력 1 복사

```
-1
```

예제 입력 2 복사

```
5 3 2
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 1 0 0
0 0 0 0 0
```

예제 출력 2 복사

```
4
```

예제 입력 3 복사

```
4 3 2
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
-1 -1 -1 -1
1 1 1 -1
```

예제 출력 3 복사

```
0
```

KeyPoint = 정점을 전부 처음에 Queue에 삽입해둔다.
Queue는 FIFO이므로 해결 가능

정점을 전부 queue에 삽입

```python
from collections import deque
temp = list(map(int,input().split()))
# 각각 M=x N=y H=z
M = temp[0]
N = temp[1]
H = temp[2]
box = [[list(map(int, input().split())) for i in range(N)] for _ in range(H)]
visit = [[[0]*M for i in range(N)] for j in range(H)]
queue = deque()
dx = [1,0,0,-1,0,0]
dy = [0,1,0,0,-1,0]
dz = [0,0,1,0,0,-1]
# queue에 정점을 넣어두는 연산
for i in range(M):
    for j in range(N):
        for k in range(H):
            # box[k][j][i] == 1:
                queue.append([k,j,i])
# bfs로 search
def bfs(box, visit, queue):
    # 큐가 돌아갈동안
    while queue:
        # tmp에서 queue의 내용을 빼준다
        tmp = queue.popleft()
        for i in range(6):
            z = tmp[0] + dz[i]
            y = tmp[1] + dy[i]
            x = tmp[2] + dx[i]
            # 해당 범위 안에 들어갈때
            if 0 <= x < M and 0 <= y < N and 0 <= z < H and box[z][y][x] == 0 and visit[z][y][x] == 0):
                visit[z][y][x] = visit[tmp[0]][tmp[1]][tmp[2]] + 1
                box[z][y][x] = 1
                queue.append([z,y,x])
bfs(box, visit, queue)
def max_value(box, visit, M, N, H):
    max_value = 0
    for x in range(M):
        for y in range(N):
            for z in range(H):
```

→ 범위에 In 했는가!

→ 방문안했으면서 토마토가 있는가

만족하면 tomato 연결

```
                # 토마토가 안익었을때
                if box[z][y][x] == 0:
                    return -1
                max_value = max(max_value, visit[z][y][x])
    return max_value
max_data = max_value(box, visit, M, N, H)
print(max_data)
```