14502번: 연구소

연구소 풀이

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
2 초	512 MB	3104	1684	1110	54.815%

문제

인체에 치명적인 바이러스를 연구하던 연구소에서 바이러스가 유출되었다. 다행히 바이러스는 아직 퍼지지 않았고, 바이러스의 확산을 막기 위해서 연구소에 벽을 세우려고 한다.

연구소는 크기가 N×M인 직사각형으로 나타낼 수 있으며, 직사각형은 1×1 크기의 정사각형으로 나누어져 있다. 연구소는 빈 칸, 벽으로 이루어져 있으며, 벽은 칸 하나를 가득 차지한다.

일부 칸은 바이러스가 존재하며, 이 바이러스는 인접한 빈 칸으로 모두 퍼져나갈 수 있다. 새로 세울 수 있는 벽의 개수는 3개이며, 꼭 3개를 세워야 한다.

예를 들어, 아래와 같이 연구소가 생긴 경우를 살펴보자.

```
2 0 0 0 1 1 0
0 0 1 0 1 2 0
0 1 1 0 1 0 0
0 1 0 0 0 0 0
0 0 0 0 0 1 1
0 1 0 0 0 0 0
0 1 0 0 0 0 0
```

이 때, 0은 빈 칸, 1은 벽, 2는 바이러스가 있는 곳이다. 아무런 벽을 세우지 않는다면, 바이러스는 모든 빈 칸으로 퍼져나갈 수 있다.

2행 1열, 1행 2열, 4행 6열에 벽을 세운다면 지도의 모양은 아래와 같아지게 된다.

```
2 1 0 0 1 1 0
1 0 1 0 1 2 0
0 1 1 0 1 0 0
0 1 0 0 0 1 0
0 0 0 0 0 1 1
```

```
0 1 0 0 0 0 0
0 1 0 0 0 0 0
```

바이러스가 퍼진 뒤의 모습은 아래와 같아진다.

```
2 1 0 0 1 1 2
1 0 1 0 1 2 2
0 1 1 0 1 2 2
0 1 0 0 0 1 2
0 0 0 0 0 1 1
0 1 0 0 0 0 0
0 1 0 0 0 0 0
```

벽을 3개 세운 뒤, 바이러스가 퍼질 수 없는 곳을 안전 영역이라고 한다. 위의 지도에서 안전 영역의 크기는 27이다.

연구소의 지도가 주어졌을 때 얻을 수 있는 안전 영역 크기의 최대값을 구하는 프로그램을 작성하시오.

처음에 생각한 풀이는

while 모든 경우의 수

벽세우기 -> flooding

사실 모든 경우의 수를 세우는 것은 비효율 적으로 보일 수 있으나, 인풋의 갯수가 64개가 한계이다.

이 위에 3개의 벽을 세우는 경우의수는 64C3 밖에 없다.

시간 복잡도는 64C3 *(그 때마다 일어나는 Flooding + 0의 갯수 세기)

```
import java.io.BufferedWriter;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.util.Arrays;

public class Main {
    public static BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    public static BufferedWriter out = new BufferedWriter(new OutputStreamWriter(System.out));
    public static int n , m;
    public static int map[][];
    public static int dx[] = {0,1,-1,0};
    public static int dy[] = {1,0,0,-1};
```

```
public static void main(String args[]) throws Exception {
    getInput();
    int score = buildWall(0);

    out.write(score+ "");
    in.close();
    out.close();
}

public static int getInput() throws Exception{
    int cnt = 0;
    String[]line = in.readLine().split(" ");
    n = Integer.parseInt(line[0]);
    m = Integer.parseInt(line[1]);
    map = new int [n][m];
    for(int i =0; i<n; i++) {
        line = in.readLine().split(" ");
        for(int j =0; j<m; j++) {
            map[i][j] = Integer.parseInt(line[j]);
            if(map[i][j]==0)cnt ++;
        }
    }
    return cnt;
}

public static void test()throws Exception {
    for(int i =0; i<n; i++) {
        for(int j =0; j<m; j++) {
            out.write(map[i][j]+" ");
        }out.write("\n");
    }
}


public static void flood(int m[][],int x, int y) {
    int tx = x;
    int ty = y;
    for(int i =0; i< 4; i++) {
        tx = x+dx[i];
        ty =y+dy[i];
        if (isInRange(tx,ty)&&m[tx][ty]==0) {
            m[tx][ty]=3;
            flood(m,tx,ty);
        }
    }
    return ;
}
public static int  calc(int map[][]) {
    int cnt = 0;
    int tmp[][] = new int[n][m];
    for(int i =0; i<n; i++) {
        tmp[i] = Arrays.copyOf(map[i], map[i].length);
    }
    for(int i =0; i<n; i++)
```

```
            for(int j =0; j<m; j++) {
                if(map[i][j]==2) {
                    flood(tmp,i,j);
                }
            }
        for(int i=0;i<n; i++) {
            for(int j =0; j<m; j++) {
                if (tmp[i][j]== 0) cnt ++;
            }
        }
        return cnt;
    }
    public static int buildWall(int step){
        if(step==3) return calc(map);
        int i, j, score = -1;

        for(i=0;i<n;i++)
            for(j=0;j<m;j++)
                if(map[i][j]==0){
                    map[i][j] = 1;
                    score = Math.max(buildWall(step+1), score);
                    map[i][j] = 0;
                }
        return score;
    }
    public static boolean isInRange(int x , int y) {
        return x>=0 && y >= 0 && x<n && y<m;
    }
}
```

이 소스에서 기억할 만한 method는 3개으 벽을 세우는 과정이다. 2차원 배열의 3개의 요소에 연속해서 어떠한 작업을 하려면 함수 없이는 6중 for loop이 필요하다. 하지만 step이라는 파라메터를 가지는 재귀함수를 이용하면 비교적 쉽게 구현 할 수 있다.

입력

첫째 줄에 지도의 세로 크기 N과 가로 크기 M이 주어진다. (3 ≤ N, M ≤ 8)

둘째 줄부터 N개의 줄에 지도의 모양이 주어진다. 0은 빈 칸, 1은 벽, 2는 바이러스가 있는 위치이다. 2의 개수는 2보다 크거나 같고, 10보다 작거나 같은 자연수이다.

빈 칸의 개수는 3개 이상이다.

출력

첫째 줄에 얻을 수 있는 안전 영역의 최대 크기를 출력한다.

예제 입력

```
7 7
2 0 0 0 1 1 0
0 0 1 0 1 2 0
0 1 1 0 1 0 0
0 1 0 0 0 0 0
0 0 0 0 0 1 1
0 1 0 0 0 0 0
0 1 0 0 0 0 0
```

예제 출력

```
27
```

예제 입력 2

```
4 6
0 0 0 0 0 0
1 0 0 0 0 2
1 1 1 0 0 2
0 0 0 0 0 2
```

예제 출력 2

```
9
```

예제 입력 3

```
8 8
2 0 0 0 0 0 0 2
2 0 0 0 0 0 0 2
2 0 0 0 0 0 0 2
2 0 0 0 0 0 0 2
2 0 0 0 0 0 0 2
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

예제 출력 3

```
3
```

힌트

출처

- 문제를 만든 사람: baekjoon
- 빠진 조건을 찾은 사람: dotorya