2146번: 다리 만들기

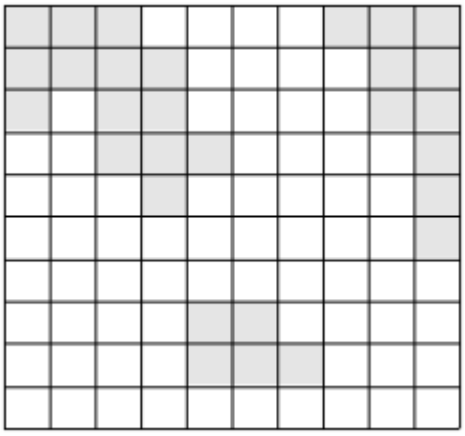
다리 만들기 성공

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
2 초	128 MB	5130	1562	999	30.099%

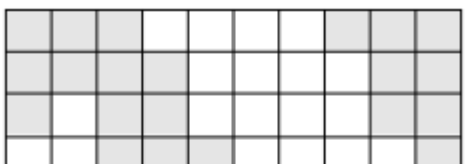
문제

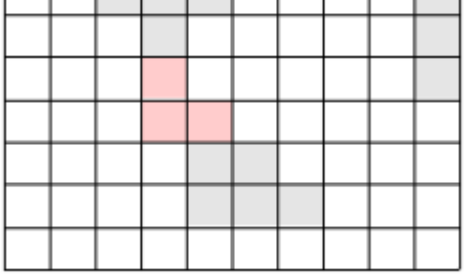
여러 섬으로 이루어진 나라가 있다. 이 나라의 대통령은, 섬들을 잇는 다리를 만들겠다는 공약으로 인기몰이를 해 당선될 수 있었다. 하지만 막상 대통령에 취임하자, 다리를 놓는다는 것이 아깝다는 생각을 하게 되었다. 그래서 그는, 생색내는 식으로 한 섬과 다른 섬을 잇는 다리 하나만을 만들기로 하였고, 그 또한 다리를 가장 짧게 하여 돈을 아끼려 하였다.

이 나라는 N×N크기의 이차원 평면상에 존재한다. 이 나라는 여러 섬으로 이루어져 있으며, 섬이란 동서남북으로 육지가 붙어있는 덩어리를 말한다. 다음은 세 개의 섬으로 이루어진 나라의 지도이다.



위의 그림에서 색이 있는 부분이 육지이고, 색이 없는 부분이 바다이다. 이 바다에 가장 짧은 다리를 놓아 두 대륙을 연결하고자 한다. 가장 짧은 다리란, 다리가 격자에서 차지하는 칸의 수가 가장 작은 다리를 말한다. 다음 그림에서 두 대륙을 연결하는 다리를 볼 수 있다.





면

물론 위의 방법 외에도 다리를 놓는 방법이 여러 가지 있으나, 위의 경우가 놓는 다리의 길이가 3으로 가장 짧다(물론 길이가 3인 다른 다리를 놓을 수 있는 방법도 몇 가지 있다).

지도가 주어질 때, 가장 짧은 다리 하나를 놓아 두 대륙을 연결하는 방법을 찾으시오.

문제 유형 : 그래프

연결 요소 (component 찾기) + flooding 문제 였다. 이 문제는 각 섬에서 부터 다른 섬까지 갈 때 만나는 최단 시간을 구하는 문제 이다. 하지만 모든 섬이 동시에 시작 하기 때문에 그 방법을 찾는 요령을 생각해 냈어야 했다.

문제 접근 방법

1. 섬이 서로 이어진다 라는 것을 표현 하려면 각 섬이 구분 되어야 한다. 섬을 확인하면서 섬을 구분한다.

2. 섬이 이어지는 과정을 단계별로 확인 해야 한다면, flooding을 해야 한다.

3. 섬이 동시에 시작 하려면, 시작점을 모두 q에 넣고 시작해야 한다.

어려웠던 점.

각 섬이 만나는 순간을 확인해서 최단 경로의 수를 확인해야 한다. 하지만 처음 만나는 유형의 알고리즘 이었다. 처음에는 cnt라는 전역 변수를 두어서 bfs 단계 마다, 증가 시키고 만나는 순간의 카운터 변수를 출력하려고 하였다. 하지만 그래프 탐색은 다 방향에서 이루어 나기 때문에 구현하기 귀찮은 예외처리가 생겼다.

그래서 게시판을 참고했는데, BFS를 통해 모든 공간을 탐색하면서 만나는 순간을 sorting이 가능한 자료구조에 집어 넣는다. 모든 탐색이 끝난후 최솟 값을 출력한다.

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.Queue;
import java.util.StringTokenizer;

public class Main {
    public static final int[] dx = { 0, 0, 1, -1 };
    public static final int[] dy = { 1, -1, 0, 0 };
    public static final ArrayList<Integer> list = new ArrayList<>();

    private void solve() {
        int n = sc.nextInt();
```

```
int n = sc.nextInt();

// 1. BFS 통해 섬의 번호를 지정해준다.
// 2. 섬인 위치는 모두 먼저 큐에 넣는다 (동시에 간척사업을 시작하기 위함)
// 3. 간척사업 중 먼저 서로 다른 섬이 닿는다면 최단경로
//   * 최단 경로가 아닐 수도 있기에 닿는 경우를 모두 저장 후 최소값 출력.

int[][] a = new int[n + 1][n + 1];
int[][] dist = new int[n + 1][n + 1];
boolean[][] c = new boolean[n + 1][n + 1];
Queue<Pair> q = new LinkedList<>();

for (int i = 1; i <= n; i++) {
    String[] s = sc.readLine().split(" ");

    for (int j = 1; j <= n; j++) {
        a[i][j] = Integer.parseInt(s[j - 1]);
    }

}

int cnt = 0;
for (int i = 1; i <= n; i++) {

    for (int j = 1; j <= n; j++) {

        if (a[i][j] == 1 && !c[i][j]) {

            numberNominate(a, c, j, i, ++cnt);


        }
    }

}


for (int i = 1; i <= n; i++) {

    for (int j = 1; j <= n; j++) {

        if (a[i][j] != 0) {
            q.add(new Pair(j, i));

        }

    }

}

bfs(q, a, c, dist);

int ans = Integer.MAX_VALUE;
for(int v : list) {
    if (ans > v) {
        ans = v;
    }
}

System.out.println(ans);

}
```

```
public static void numberNominate(int[][] a, boolean[][] c, int x, int y, int idx) {
    int n = a.length - 1;
    Queue<Pair> q = new LinkedList<>();

    q.add(new Pair(x, y));
    a[y][x] = idx;
    c[y][x] = true;

    while (!q.isEmpty()) {

        x = q.peek().x;
        y = q.peek().y;
        q.poll();

        for (int k = 0; k < 4; k++) {
            int nx = x + dx[k];
            int ny = y + dy[k];

            if (0 < nx && nx <= n && 0 < ny && ny <= n) {

                if (a[ny][nx] == 1 && !c[ny][nx]) {
                    q.add(new Pair(nx, ny));
                    a[ny][nx] = idx;

                    c[ny][nx] = true;
                }
            }
        }

    }

}

public static void bfs(Queue<Pair> q, int[][] a, boolean[][] c, int[][] dist) {
    int n = a.length - 1;

    while (!q.isEmpty()) {
        int x = q.peek().x;
        int y = q.peek().y;
        q.poll();

        for (int k = 0; k < 4; k++) {
            int nx = x + dx[k];
            int ny = y + dy[k];
            if (0 < nx && nx <= n && 0 < ny && ny <= n) {

                if (a[ny][nx] != 0 && a[ny][nx] != a[y][x]) {
                    list.add(dist[ny][nx] + dist[y][x]);
                }

                if (a[ny][nx] == 0) {
                    q.add(new Pair(nx, ny));

                    a[ny][nx] = a[y][x];
                    dist[ny][nx] = dist[y][x] + 1;
                }
            }
        }
```

```
    }

}

public static void main(String[] args) {
    sc.init();

    new Main().solve();
}

static class Pair {
    int x;
    int y;

    Pair(int x, int y) {
        this.x = x;
        this.y = y;
    }
}

static class sc {
    private static BufferedReader br;
    private static StringTokenizer st;

    static void init() {
        br = new BufferedReader(new InputStreamReader(System.in));
        st = new StringTokenizer("");
    }

    static String readLine() {
        try {
            return br.readLine();
        } catch (IOException e) {
        }
        return null;
    }

    static String readLineReplace() {
        try {
            return br.readLine().replaceAll("\\s+", "");
        } catch (IOException e) {
        }
        return null;
    }

    static String next() {
        while (!st.hasMoreTokens()) {
            try {
                st = new StringTokenizer(br.readLine());
            } catch (IOException e) {
            }
        }
        return st.nextToken();
    }

    static long nextLong() {
        return Long.parseLong(next());
    }
```

```
        static int nextInt() {
            return Integer.parseInt(next());
        }

        static double nextDouble() {
            return Double.parseDouble(next());
        }
    }
}
```

입력

첫 줄에는 지도의 크기 N(100이하의 자연수)가 주어진다. 그 다음 N줄에는 N개의 숫자가 빈칸을 사이에 두고 주어지며, 0은 바다, 1은 육지를 나타낸다.

출력

첫째 줄에 가장 짧은 다리의 길이를 출력한다.

예제 입력

```
10
1 1 1 0 0 0 0 1 1 1
1 1 1 1 0 0 0 0 1 1
1 0 1 1 0 0 0 0 1 1
0 0 1 1 1 0 0 0 0 1
0 0 0 1 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0
0 0 0 0 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0
```

예제 출력

3

힌트

알고리즘 분류

- BFS
- DFS