题目描述

输入一行字符串，字符串可转换为N*N的数组，数组可认为是一个水域，判断多少天后，水域被全部污染。
数组中只有0和1，0表示纯净，1表示污染，每天只可污染上下左右的水域，如果开始全部被污染，或永远无法污染，则返回-1。

输入描述

无

输出描述

无

用例

输入	1,0,1,0,0,0,1,0,1
输出	2
说明	输入转化为数组为： 1 0 1 0 0 0 1 0 1 第一天后水域变为 1 1 1 1 0 1 1 1 1 第二天全部被污染

输入	0,0,0,0
输出	-1
说明	无

题目解析

本题就是[illegible]

的变种题。

可以采用图的多源BFS来解决。图的多源BFS实现关键点在于理解queue的作用。更多解析请看上面链接博客的解释。

JavaScript算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

rl.on("line", (line) => {
  const arr = line.split(",").map(Number);
  const n = Math.sqrt(arr.length);
  let total = arr.length; // 未污染区域数量

  const matrix = new Array(n).fill(0).map(() => new Array(n));
  let queue = [];

  for (let i = 0; i < n; i++) {
    for (let j = 0; j < n; j++) {
      matrix[i][j] = arr[n * i + j];
      if (matrix[i][j] === 1) {
        queue.push([i, j]);
        total--;
      }
    }
  }

  if (queue.length === 0 || queue.length === arr.length) {
    return console.log(-1);
  }

  const offset = [
    [-1, 0],
    [1, 0],
    [0, -1],
    [0, 1],
  ];

  while (queue.length && total) {
    const newQueue = [];

```

```
    for (const [i, j] of queue) {
      const num = matrix[i][j];

      for (let m = 0; m < offset.length; m++) {
        const [offsetX, offsetY] = offset[m];

        const newI = i + offsetX;
        const newJ = j + offsetY;

        if (
          newI >= 0 &&
          newI < n &&
          newJ >= 0 &&
          newJ < n &&
          matrix[newI][newJ] === 0
        ) {
          matrix[newI][newJ] = num + 1;
          newQueue.push([newI, newJ]);
          total--;

          if (total === 0) {
            console.log(num);
            return;
          }
        }
      }
    }

    queue = newQueue;
  }
});
```

Java算法源码

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

public class Main {
  // 输入获取
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    Integer[] arr =
        Arrays.stream(sc.nextLine().split(",")).map(Integer::parseInt).toArray(Integer[]::new);

    getResult(arr);
  }

  // 算法入口
  public static void getResult(Integer[] arr) {
    int n = (int) Math.sqrt(arr.length);
    int total = arr.length;

    LinkedList<Integer[]> queue = new LinkedList<>();

    int[][] matrix = new int[n][n];
    for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
        matrix[i][j] = arr[n * i + j];
        if (matrix[i][j] == 1) {
          queue.add(new Integer[] {i, j});
          total--;
        }
      }
    }

    if (queue.size() == 0 || queue.size() == arr.length) {
      System.out.println(-1);
      return;
    }
```

```
38
39      int[][] offsets = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};
40
41      while (queue.size() > 0 && total > 0) {
42        Integer[] tmp = queue.removeFirst();
43        int i = tmp[0];
44        int j = tmp[1];
45
46        int num = matrix[i][j];
47
48        for (int[] offset : offsets) {
49          int newI = i + offset[0];
50          int newJ = j + offset[1];
51
52          if (newI >= 0 && newI < n && newJ >= 0 && newJ < n && matrix[newI][newJ] == 0) {
53            matrix[newI][newJ] = num + 1;
54            queue.add(new Integer[] {newI, newJ});
55            total--;
56            if (total == 0) {
57              System.out.println(num);
58              return;
59            }
60          }
61        }
62      }
63    }
64  }
```

Python算法源码

```
1  import math
2
3  # 输入获取
4  arr = list(map(int, input().split(",")))
5
6
7  # 算法入口
8  def getResult():
9      n = int(math.sqrt(len(arr)))
10     total = len(arr)
11
12     matrix = [[0 for _ in range(n)] for _ in range(n)]
13     queue = []
14
15     for i in range(n):
16         for j in range(n):
17             matrix[i][j] = arr[n * i + j]
18             if matrix[i][j] == 1:
19                 queue.append([i, j])
20                 total -= 1
21
```

```
    if len(queue) == 0 or len(queue) == len(arr):
        print(-1)
        return

    offsets = ((-1, 0), (1, 0), (0, -1), (0, 1))

    while len(queue) > 0 and total > 0:
        newQueue = []
        for i, j in queue:
            num = matrix[i][j]

            for offsetX, offsetY in offsets:
                newI = i + offsetX
                newJ = j + offsetY

                if 0 <= newI < n and 0 <= newJ < n and matrix[newI][newJ] == 0:
                    matrix[newI][newJ] = num + 1
                    newQueue.append([newI, newJ])
                    total -= 1
                    if total == 0:
                        print(num)
                        return

        queue = newQueue


# 算法调用
getResult()
```