寻找最大价值的矿堆

题目描述：

给你一个由 '0'（空地）、'1'（银矿）、'2'（金矿）组成的的地图，矿堆只能由上下左右相邻的金矿或银矿连接形成。超出地图范围可以认为是空地。

假设银矿价值 1 ，金矿价值 2，请你找出地图中最大价值的矿堆并输出该矿堆的价值

输入描述：

地图元素信息如：

22220

00000

00000

11111

地图范围最大 300*300

0<= 地图元素 <=2

输出描述：

矿堆的最大价值

补充说明：

题目描述：

给你一个由 '0'（空地）、'1'（银矿）、'2'（金矿）组成的的地图，矿堆只能由上下左右相邻的金矿或银矿连接形成。超出地图范围可以认为是空地。

假设银矿价值 1 ，金矿价值 2，请你找出地图中最大价值的矿堆并输出该矿堆的价值

输入描述：

地图元素信息如：

22220

00000

00000

11111

地图范围最大 300*300

0<= 地图元素 <=2

输出描述：

矿堆的最大价值

补充说明：

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Main {
    private static int count = 0;
    private static int result = 0;

    public static void main(String[] args) {

        Scanner in = new Scanner(System.in);
        List<String> list = new ArrayList<>();
        while (in.hasNextLine()) {
            list.add(in.nextLine());
        }
        //输入内容
        int[][] area = new int[list.size()][list.get(0).length()];
        for (int i = 0; i < list.size(); i++) {
            char[] charArr = list.get(i).toCharArray();
            for (int j = 0; j < charArr.length; j++) {

                area[i][j] = Integer.parseInt(String.valueOf(charArr[j]));
            }
        }
        //挖矿，挖过的矿则将矿地变为 0，直到连续矿地无矿可挖
        deg(area);
    }

    static void deg(int[][] area) {
        for (int c = 0; c < area.length; c++) {
            for (int k = 0; k < area[0].length; k++) {
                dfs(area, c, k);
                result = Math.max(count, result);
                count = 0;
            }
        }
        System.out.println(result);
    }

    //DFS 深度优先算法
```

```
    static void dfs(int[][] area, int c, int k) {
        if (!inArea(area, c, k)) {
            return;
        }
        if (area[c][k] == 0) {
            return;
        }
        count += area[c][k];
        area[c][k] = 0;
        dfs(area, c - 1, k);
        dfs(area, c + 1, k);
        dfs(area, c, k - 1);
        dfs(area, c, k + 1);
    }

    static boolean inArea(int[][] area, int c, int k) {
        return 0 <= c && c < area.length && 0 <= k && k < area[0].length;
    }
}
```