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

示例 1
输入：
22220
00000
00000
01111
输出：
8
说明：
示例 2
输入：
22220
00020
00010
01111
输出：
15
说明：
示例 3
输入：
20000
00020
00000
00111
输出：
3
说明：

```
import java.util.Scanner;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        // 注意 hasNext 和 hasNextLine 的区别

        int[][] arr = new int[300][300];
        int r = 0;
        int colLen = 0;
        while (in.hasNextLine()) { // 注意 while 处理多个 case
            String line = in.nextLine();
            String[] split = line.split("");
            int c = 0;
            for (String s : split) {
                arr[r][c++] = Integer.parseInt(s);
            }
            colLen = c;
            r++;
        }
        int rowLen = r;

        int ans = 0;
        for (r = 0; r < rowLen; r++) {
            for (int c = 0; c < colLen; c++) {
                int ret = func(arr, r, c, rowLen, colLen);
                ans = Math.max(ans, ret);
            }
        }

        System.out.println(ans);
    }

    private static int func(int[][] arr, int row, int col, int rowLen, int colLen) {
        int currVal = arr[row][col];
        if (currVal == 0) {
            return 0;
        }
        int ret = currVal;
        arr[row][col] = 0;

        if (row - 1 >= 0 && arr[row - 1][col] > 0) {
```

```
            ret += func(arr, row - 1, col, rowLen, colLen);
        }
        if (row + 1 < rowLen && arr[row + 1][col] > 0) {
            ret += func(arr, row + 1, col, rowLen, colLen);
        }
        if (col - 1 >= 0 && arr[row][col - 1] > 0) {
            ret += func(arr, row, col - 1, rowLen, colLen);
        }
        if (col + 1 < colLen && arr[row][col + 1] > 0) {
            ret += func(arr, row, col + 1, rowLen, colLen);
        }

        return ret;
    }
}
```