题目描述：

停车场有一横排车位，0 代表没有停车，1 代表有车。至少停了一辆车在车位上，也至少有一个空位没有停车。

为了防剐蹭，需为停车人找到一个车位，使得距停车人的车最近的车辆的距离是最大的，返回此时的最大距离。

输入描述：

1、一个用半角逗号分割的停车标识字符串，停车标识为 0 或 1，0 为空位，1 为已停车。

2、停车位最多 100 个。

输出描述：

输出一个整数记录最大距离。

补充说明：

示例 1

输入：

```
1,0,0,0,0,1,0,0,1,0,1
```

输出：

```
2
```

说明：

当车停在第 3 个位置上时，离其最近的的车距离为 2（1 到 3）。

当车停在第 4 个位置上时，离其最近的的车距离为 2（4 到 6）。

其他位置距离为 1。

因此最大距离为 2。

```
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
        while (in.hasNext()) { // 注意 while 处理多个 case
            String[] ss = in.nextLine().split(",");
            int len = ss.length;
            int distance = 0;
```

```
                for(int i = 0; i < len; i++)    {
                    if(ss[i].equals("1")) continue;
                    int left = i,right = i;
                    while(left >= 0 && !ss[left].equals("1")) {
                        left--;
                    }
                    while(right < len && !ss[right].equals("1")) {
                        right++;
                    }
                    if(left == -1 || right == len) {
                        distance = Math.max(Math.max(i - left, right - i), distance);
                    } else {
                        distance = Math.max(Math.min(i - left, right - i), distance);
                    }
                    // System.out.println(distance + " " + i + left + right);
                }

                System.out.println(distance);
            }
        }
}
```