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。

用例

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

题目解析

首先，这道题题目描述中有一个关键词"为了防剐蹭"，隐式含义应该是尽可能让停车位置左右两边是空位，这样才能起到防剐蹭的效果。

然后，满足上面停车条件后，找到离当前停的车A最近的车B，求解A~B之间的距离，比如用例中

```
1,0,0,0,0,1,0,0,1,0,1
B   A
```

当车停在第3个车位，满足防剐蹭要求，此时离停车位置最近的车停在第1个车位上，距离为2。

最后，找出所有的停车情况下的A~B距离，求出其中最大那个距离。（.....太绕口了）

另外如果没有满足防刮蹭要求，比如101，或者满足一半防刮蹭要求，比如1001，此时A~B最近距离都是1。

还有一种情况，001，此时将车停在边界0上，A~B最近距离为2。

解题思路很简单，把输入字符串中连续0子串截取出来，先判断是否为边界子串，若是，则最近停车距离就是子串自身长度，若不是，则最近停车距离是：

- 当是101这种情况，即只有1个0时，最近停车距离为1
- 当是1001，10001这种情况，最近停车距离为Math.ceil(zeroLen / 2)

Java算法源码

```java
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println(getResult(sc.nextLine().split(",")));
  }

  public static int getResult(String[] arr) {
    StringBuilder sb = new StringBuilder();
    for (String s : arr) sb.append(s);

    String[] sArr = sb.toString().split("1");
    int maxLen = 0;

    for (int i = 0; i < sArr.length; i++) {
      String ele = sArr[i];

      if (i == 0 || i == sArr.length - 1) {
        maxLen = Math.max(maxLen, ele.length());
      } else {
        if (ele.length() == 1) {
          maxLen = Math.max(maxLen, 1);
        } else {
          maxLen = Math.max(maxLen, ele.length() / 2);
        }
      }
    }

    return maxLen;
  }
}
```

JS算法源码

```
/* 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 str = line.split(",").join("");
  console.log(maxDistance(str));
});

/* 最大车间距 */
function maxDistance(str) {
  const arr = str.split("1");

  let maxLen = 0;
  arr.forEach((ele, index) => {
    if (index === 0 || index === arr.length - 1) {
      maxLen = Math.max(maxLen, ele.length);
    } else {
      if (ele.length === 1) {
        maxLen = Math.max(maxLen, 1);
      } else {
        maxLen = Math.max(maxLen, Math.ceil(ele.length / 2));
      }
    }
  });
  return maxLen;
}
```

Python算法源码

```python
# 输入获取
s = "".join(input().split(","))


# 算法入口
def getResult():
    arr = s.split("1")

    maxLen = 0
    for i in range(len(arr)):
        ele = arr[i]
        if i == 0 or i == len(arr) - 1:
            maxLen = max(maxLen, len(ele))
        else:
            if len(ele) == 1:
                maxLen = max(maxLen, 1)
            else:
                maxLen = max(maxLen, len(ele) // 2)

    return maxLen


# 算法调用
print(getResult())
```