

题目描述

工位由序列F1,F2...Fn组成，Fi值为0、1或2。其中0代表空置，1代表有人，2代表障碍物。

- 1、某一空位的友好度为左右连续老员工数之和，
- 2、为方便新员工学习求助，优先安排友好度高的空位，

给出工位序列，求所有空位中友好度的最大值。

输入描述

第一行为工位序列：F1, F2...Fn组成，
1<=n<=10000，Fi值为0、1或2。其中0代表空置，1代表有人，2代表障碍物。

输出描述

所有空位中友好度的最大值。如果没有空位，返回0。

用例

输入	0 1 0
输出	1
说明	第1个位置和第3个位置，友好度均为1。

输入	1 1 0 1 2 1 0
输出	3
说明	第3个位置友好度为3。因障碍物隔断，左边得2分，右边只能得1分。

题目解析

本题最优解题思路如下：

定义一个变量friendShip = 0，

首先对 输入数组 进行正向遍历（从左到右）：

- 遇到“1”，则friendShip++
- 遇到“0”，则说明该空位左边的友好度为friendShip，记录下来，然后将friendShip重置为0
- 遇到“2”，则直接将friendShip重置为0

这样每个空位的左边的友好度就统计出来了。

接着，将friendShip重置为0，

然后对输入数组进行反向遍历（从右到左）：

- 遇到“1”，则friendShip++
- 遇到“0”，则说明该空位右边的友好度为friendShip，累加下来，然后将friendShip重置为0
- 遇到“2”，则直接将friendShip重置为0

这样每个空位的整体友好度就统计出来了，取最大值返回。

JavaScript算法源码

```
1  /* JavaScript Node ACM模式 控制台输入获取 */
2  const readline = require("readline");
3
4  const rl = readline.createInterface({
5    input: process.stdin,
6    output: process.stdout,
7  });
8
9  rl.on("line", (line) => {
10    const arr = line.split(" ");
11    console.log(getResult(arr));
12  });
```

JavaScript算法源码

```
1  /* JavaScript Node ACM模式 控制台输入获取 */
2  const readline = require("readline");
3
4  const rl = readline.createInterface({
5    input: process.stdin,
6    output: process.stdout,
7  });
8
9  rl.on("line", (line) => {
10    const arr = line.split(" ");
11    console.log(getResult(arr));
12  });
13
14 function getResult(arr) {
15   // 记录空位的友好度
16   const ep = {};
17
18   let friendShip = 0;
19   // 从左向右遍历，记录每个空位左边的友好度
20   for (let i = 0; i < arr.length; i++) {
21     switch (arr[i]) {
22       case "0":
23         ep[i] = friendShip;
24         friendShip = 0;
25         break;
26       case "1":
27         friendShip++;
28         break;
29       case "2":
30         friendShip = 0;
31         break;
32     }
33   }
34
35   friendShip = 0;
36   let ans = 0;
37   // 从右向左遍历，累加每个空位右边的友好度，这样就得到了每个空位的友好度，取最大值即可
38   for (let i = arr.length - 1; i >= 0; i--) {
39     switch (arr[i]) {
40       case "0":
41         ans = Math.max(ans, ep[i] + friendShip);
42         friendShip = 0;
43         break;
44       case "1":
45         friendShip++;
46         break;
47       case "2":
48         friendShip = 0;
49         break;
50     }
51   }
52
53   return ans;
54 }
```

Java算法源码

```
1  import java.util.HashMap;
2  import java.util.Scanner;
3
4  public class Main {
5    public static void main(String[] args) {
6      Scanner sc = new Scanner(System.in);
7
8      String[] arr = sc.nextLine().split(" ");
9
10     System.out.println(getResult(arr));
11   }
12
13   public static int getResult(String[] arr) {
14     // 记录空位的友好度
15     HashMap<Integer, Integer> ep = new HashMap<>();
16
17     int friendShip = 0;
18     // 从左向右遍历，记录每个空位左边的友好度
19     for (int i = 0; i < arr.length; i++) {
20       switch (arr[i]) {
21         case "0":
22           ep.put(i, friendShip);
23           friendShip = 0;
24           break;
25         case "1":
26           friendShip++;
27           break;
28         case "2":
29           friendShip = 0;
30           break;
31       }
32     }
33
34     friendShip = 0;
35     let ans = 0;
36     // 从右向左遍历，累加每个空位右边的友好度，这样就得到了每个空位的友好度，取最大值即可
37     for (int i = arr.length - 1; i >= 0; i--) {
38       switch (arr[i]) {
39         case "0":
40         ans = Math.max(ans, ep[i] + friendShip);
41         friendShip = 0;
42         break;
43         case "1":
44         friendShip++;
45         break;
46         case "2":
47         friendShip = 0;
48         break;
49       }
50     }
51
52     return ans;
53   }
54 }
```

Java算法源码

```
1 import java.util.HashMap;
2 import java.util.Scanner;
3
4 public class Main {
5     public static void main(String[] args) {
6         Scanner sc = new Scanner(System.in);
7
8         String[] arr = sc.nextLine().split(" ");
9
10        System.out.println(getResult(arr));
11    }
12
13    public static int getResult(String[] arr) {
14        // 记录空位的友好度
15        HashMap<Integer, Integer> ep = new HashMap<>();
16
17        int friendShip = 0;
18        // 从左向右遍历, 记录每个空位左边的友好度
19        for (int i = 0; i < arr.length; i++) {
20            switch (arr[i]) {
21                case "0":
22                    ep.put(i, friendShip);
23                    friendShip = 0;
24                    break;
25                case "1":
26                    friendShip++;
27                    break;
28                case "2":
29                    friendShip = 0;
30                    break;
31            }
32        }
33
34        friendShip = 0;
35        int ans = 0;
36        // 从右向左遍历, 累加每个空位右边的友好度, 这样就得到了每个空位的友好度, 取最大值即可
37        for (int i = arr.length - 1; i >= 0; i--) {
38            switch (arr[i]) {
39                case "0":
40                    ans = Math.max(ans, ep.get(i) + friendShip);
41                    friendShip = 0;
42                    break;
43                case "1":
44                    friendShip++;
45                    break;
46                case "2":
47                    friendShip = 0;
48                    break;
49            }
50        }
51
52        return ans;
53    }
54 }
```

Python算法源码

```
1 # 输入获取
2 arr = input().split()
3
4
5 # 算法入口
6 def getResult(arr):
7     # 记录空位的友好度
8     ep = {}
9
10    friendShip = 0
11    # 从左向右遍历, 记录每个空位左边的友好度
12    for i in range(len(arr)):
13        if arr[i] == "0":
14            ep[i] = friendShip
15            friendShip = 0
16        elif arr[i] == "1":
17            friendShip += 1
18        else:
19            friendShip = 0
20
21    friendShip = 0
22    ans = 0
23    # 从右向左遍历, 累加每个空位右边的友好度, 这样就得到了每个空位的友好度, 取最大值即可
24    for i in range(len(arr) - 1, 0, -1):
25        if arr[i] == "0":
26            ans = max(ans, ep[i] + friendShip)
27            friendShip = 0
28        elif arr[i] == "1":
29            friendShip += 1
30        else:
31            friendShip = 0
32
33    return ans
```



伏城之外 已关注

👍 0 🗨 4



📄 0 📁

专栏目录

已订阅

Python算法源码

```
1  # 输入获取
2  arr = input().split()
3
4
5  # 算法入口
6  def getResult(arr):
7      # 记录空位的友好度
8      ep = {}
9
10     friendShip = 0
11     # 从左向右遍历，记录每个空位左边的友好度
12     for i in range(len(arr)):
13         if arr[i] == "0":
14             ep[i] = friendShip
15             friendShip = 0
16         elif arr[i] == "1":
17             friendShip += 1
18         else:
19             friendShip = 0
20
21     friendShip = 0
22     ans = 0
23     # 从右向左遍历，累加每个空位右边的友好度，这样就得到了每个空位的友好度，取最大值即可
24     for i in range(len(arr) - 1, 0, -1):
25         if arr[i] == "0":
26             ans = max(ans, ep[i] + friendShip)
27             friendShip = 0
28         elif arr[i] == "1":
29             friendShip += 1
30         else:
31             friendShip = 0
32
33     return ans
34
35
36 # 算法调用
37 print(getResult(arr))
```

[复制](#)