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

```
/* 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 arr = line.split(" ");
  console.log(getResult(arr));
```

JavaScript算法源码

```javascript
/* 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 arr = line.split(" ");
  console.log(getResult(arr));
});

function getResult(arr) {
  // 记录空位的友好度
  const ep = {};

  let friendShip = 0;
  // 从左向右遍历，记录每个空位左边的友好度
  for (let i = 0; i < arr.length; i++) {
    switch (arr[i]) {
      case "0":
        ep[i] = friendShip;
        friendShip = 0;
        break;
      case "1":
        friendShip++;
        break;
      case "2":
        friendShip = 0;
        break;
    }
  }

  friendShip = 0;
  let ans = 0;
  // 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
  for (let i = arr.length - 1; i >= 0; i--) {
    switch (arr[i]) {
      case "0":
        ans = Math.max(ans, ep[i] + friendShip);
        friendShip = 0;
        break;
      case "1":
        friendShip++;
        break;
      case "2":
        friendShip = 0;
        break;
    }
  }

  return ans;
}
```

Java算法源码

```java
import java.util.HashMap;
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    String[] arr = sc.nextLine().split(" ");

    System.out.println(getResult(arr));
  }

  public static int getResult(String[] arr) {
    // 记录空位的友好度
    HashMap<Integer, Integer> ep = new HashMap<>();

    int friendShip = 0;
    // 从左向右遍历，记录每个空位左边的友好度
    for (int i = 0; i < arr.length; i++) {
      switch (arr[i]) {
        case "0":
          ep.put(i, friendShip);
          friendShip = 0;
          break;
        case "1":
          friendShip++;
          break;
        case "2":
          friendShip = 0;
          break;
      }
    }

    // friendShip = 0;
```

Java算法源码

```java
import java.util.HashMap;
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    String[] arr = sc.nextLine().split(" ");

    System.out.println(getResult(arr));
  }

  public static int getResult(String[] arr) {
    // 记录空位的友好度
    HashMap<Integer, Integer> ep = new HashMap<>();

    int friendShip = 0;
    // 从左向右遍历，记录每个空位左边的友好度
    for (int i = 0; i < arr.length; i++) {
      switch (arr[i]) {
        case "0":
          ep.put(i, friendShip);
          friendShip = 0;
          break;
        case "1":
          friendShip++;
          break;
        case "2":
          friendShip = 0;
          break;
      }
    }

    friendShip = 0;
    int ans = 0;
    // 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
    for (int i = arr.length - 1; i >= 0; i--) {
      switch (arr[i]) {
        case "0":
          ans = Math.max(ans, ep.get(i) + friendShip);
          friendShip = 0;
          break;
        case "1":
          friendShip++;
          break;
        case "2":
          friendShip = 0;
          break;
      }
    }

    return ans;
  }
}
```

Python算法源码

```python
# 输入获取
arr = input().split()


# 算法入口
def getResult(arr):
    # 记录空位的友好度
    ep = {}

    friendShip = 0
    # 从左向右遍历，记录每个空位左边的友好度
    for i in range(len(arr)):
        if arr[i] == "0":
            ep[i] = friendShip
            friendShip = 0
        elif arr[i] == "1":
            friendShip += 1
        else:
            friendShip = 0

    friendShip = 0
    ans = 0
    # 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
    for i in range(len(arr) - 1, 0, -1):
        if arr[i] == "0":
            ans = max(ans, ep[i] + friendShip)
            friendShip = 0
        elif arr[i] == "1":
            friendShip += 1
        else:
            friendShip = 0

    return ans
```

Python算法源码

复制

```python
# 输入获取
arr = input().split()


# 算法入口
def getResult(arr):
    # 记录空位的友好度
    ep = {}

    friendShip = 0
    # 从左向右遍历，记录每个空位左边的友好度
    for i in range(len(arr)):
        if arr[i] == "0":
            ep[i] = friendShip
            friendShip = 0
        elif arr[i] == "1":
            friendShip += 1
        else:
            friendShip = 0

    friendShip = 0
    ans = 0
    # 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
    for i in range(len(arr) - 1, 0, -1):
        if arr[i] == "0":
            ans = max(ans, ep[i] + friendShip)
            friendShip = 0
        elif arr[i] == "1":
            friendShip += 1
        else:
            friendShip = 0

    return ans


# 算法调用
print(getResult(arr))
```