【华为OD机考 统一考试机试C卷】贪吃的猴子（C++ Java JavaScript Python C语言）

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

目前在考C卷，经过两个月的收集整理，C卷真题已基本整理完毕
抽到原题的概率为2/3到3/3，也就是最少抽到两道原题。请注意：大家刷完C卷真题，最好要把B卷的真题刷一下，因为C卷的部分真题来自B卷。
另外订阅专栏还可以联系笔者开通在线 OJ 进行刷题，提高刷题效率。
真题目录：华为OD机考机试 真题目录（C卷 + D卷 + B卷 + A卷） + 考点说明
专栏：2023华为OD机试(B卷+C卷+D卷)（C++JavaJSPy）
华为OD面试真题精选：华为OD面试真题精选
在线OJ：点击立即刷题，模拟真实机考环境

题目描述

只贪吃的猴子，来到一个果园，发现许多串香蕉排成一行，每串香蕉上有若干根香蕉。每串香蕉的根数由数组numbers给出。

猴子获取香蕉，每次都只能从行的开头或者末尾获取，并且只能获取N次，求猴子最多能获取多少根香蕉。

输入描述

第一行为数组numbers的长度

第二行为数组numbers的值每个数字通过空格分开

第三行输入为N，表示获取的次数

备注:

- 1 ≤ numbers.length ≤ 100000
- 1 ≤ numbers ≤ 100
- 1 ≤ N ≤ numbers.length

输出描述

按照题目要求能获取的最大数值

用例

输入

```
7
1 2 2 7 3 6 1
3
```

输出

```
10
```

说明

> 第一次获取香蕉，无论是从行的开头或者末尾获取，得到的香蕉根数目为1, 但是，从行末尾获取能获取到最优的策略，后面可以直接得到香蕉根数目6和3。因此最终根数为1+6+3=10

用例2

输入

```
3
1 2 3
3
```

输出

```
6
```

说明

> 全部获取所有的香蕉，因此最终根数为1+2+3=6

用例3

输入

```
4
4 2 2 3
2
```

输出

```
7
```

说明

> 第一次获取香蕉为行的开头，第二次获取为行的末尾，因此最终根数为4+3=7

题目解析

要求从：每次都只能从行的开头或者末尾获取

我们以用例1解释题目：

```
1 2 2 7 3 6 1
```

每次都是开头或末尾：

第一次：开头和末尾都是1，

第二次：

如果我们第一次是开头，此时数字就是【2 2 7 3 6 1】，开头就是2 结尾就是1。

如果我们第一次是结尾，此时数字就是【1 2 2 7 3 6 】，开头就是1 结尾就是6。

这样我们就会发现第一次选末尾 第二次选末尾，第三次仍然选末尾，这样就是最多根。

我们以用例3解释题目：

```
4 2 2 3
```

我们一眼可以看出，第一次选开头 第二次选末尾。

从这两个例子，我们好像找不到啥规律啊，但是我们把视角转到**选不到的桃子**，你会发现，无论每次是选开头还是结尾，选不到的桃子永远是连续的，对不对！！！ 再转念一想，我们把数组看成一个环，选中的开头和结尾是不是也就是连续的啊。这样我们自然而然就想到了**【滑动窗口】**

试验了两种解法，**一种的选中的是连续的**，**一种是未选中的连续（选中的就是数组-未选中的）**。我觉得从未选中的角度去解题比较简单。

最终就转换为：求某个连续的区间是的总和最小。

解题思路

1. 读取输入，包括数组长度、数组元素（每串香蕉的数量），以及猴子可以获取的次数。
2. 计算数组中所有香蕉的总数。
3. 如果猴子可以获取的次数等于数组长度，即猴子可以拿走所有的香蕉，直接返回总数。
4. **计算猴子不能拿走的连续香蕉串的最小总数。**这是通过滑动窗口实现的，窗口大小为 `数组长度 - N`。
5. 初始化滑动窗口的和为窗口内的第一段连续香蕉串的和。
6. 滑动窗口，每次向右移动一位，更新窗口和，并记录最小的窗口和。
7. 猴子能获取的最大香蕉数是总和减去最小窗口和。

模拟计算过程

给定输入：

```
数组长度：7
香蕉数量：1 2 2 7 3 6 1
猴子次数：3
```

1. 计算香蕉总数：`1 + 2 + 2 + 7 + 3 + 6 + 1 = 22`
2. 窗口大小：`7 - 3 = 4`
3. 初始化窗口和：`1 + 2 + 2 + 7 = 12`

滑动窗口并计算最小窗口和：

- 窗口 `[2, 2, 7, 3]` 和为 `14`，最小和仍为 `12`
- 窗口 `[2, 7, 3, 6]` 和为 `18`，最小和仍为 `12`
- 窗口 `[7, 3, 6, 1]` 和为 `17`，最小和仍为 `12`

5. 猴子能获取的最大香蕉数： `总和 - 最小窗口和 = 22 - 12 = 10`

因此，猴子能获取的最大香蕉数为 `10`。

C++

```cpp
#include <iostream>
#include <vector>
#include <climits>

using namespace std;
// 计算猴子能获取的最大香蕉数
int maxBananas(const vector<int>& numbers, int N) {
    int total = 0; // 初始化数组总和为0
    // 计算数组中所有香蕉的总数
    for (int number : numbers) {
        total += number;
    }

    // 如果N等于数组长度，猴子可以拿走所有的香蕉
    if (N == numbers.size()) {
        return total;
    }

    int minWindowSum = INT_MAX; // 初始化最小窗口和为最大整数值
    int currentWindowSum = 0; // 初始化当前窗口和为0
    int windowSize = numbers.size() - N; // 计算窗口大小

    // 初始化窗口的和
    for (int i = 0; i < windowSize; i++) {
        currentWindowSum += numbers[i];
    }
    minWindowSum = currentWindowSum; // 将当前窗口和赋值给最小窗口和

```

```
    // 通过滑动窗口计算最小窗口和
    for (int i = windowSize; i < numbers.size(); i++) {
        // 窗口滑动，加上新进入窗口的元素，减去离开窗口的元素
        currentWindowSum += numbers[i] - numbers[i - windowSize];
        // 更新最小窗口和
        minWindowSum = min(minWindowSum, currentWindowSum);
    }

    // 猴子能获取的最大香蕉数是总和减去最小窗口和
    return total - minWindowSum;
}

int main() {
    int len; // 读取数组长度
    cin >> len;
    vector<int> numbers(len); // 创建数组存储每串香蕉的数量
    for (int i = 0; i < len; i++) {
        cin >> numbers[i]; // 循环读取每串香蕉的数量
    }
    int N; // 读取猴子可以获取的次数
    cin >> N;
    cout << maxBananas(numbers, N) << endl; // 输出猴子能获取的最大香蕉数
    return 0;
}
```

Java

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int len = scanner.nextInt(); // 读取数组长度
        int[] numbers = new int[len]; // 创建数组存储每串香蕉的数量
        for (int i = 0; i < len; i++) {
            numbers[i] = scanner.nextInt(); // 循环读取每串香蕉的数量
        }
        int N = scanner.nextInt(); // 读取猴子可以获取的次数
        System.out.println(maxBananas(numbers, N)); // 输出猴子能获取的最大香蕉数
     }

```

```java

    // 定义方法计算猴子能获取的最大香蕉数
    private static int maxBananas(int[] numbers, int N) {
        int total = 0; // 初始化数组总和为0
        // 计算数组中所有香蕉的总数
        for (int number : numbers) {
            total += number;
        }

        // 如果N等于数组长度，猴子可以拿走所有的香蕉
        if (N == numbers.length) {
            return total;
        }

        int minWindowSum = Integer.MAX_VALUE; // 初始化最小窗口和为最大整数值
        int currentWindowSum = 0; // 初始化当前窗口和为0
        int windowSize = numbers.length - N; // 计算窗口大小

        // 初始化窗口的和
        for (int i = 0; i < windowSize; i++) {
            currentWindowSum += numbers[i];
        }
        minWindowSum = currentWindowSum; // 将当前窗口和赋值给最小窗口和

        // 通过滑动窗口计算最小窗口和
        for (int i = windowSize; i < numbers.length; i++) {
            // 窗口滑动，加上新进入窗口的元素，减去离开窗口的元素
            currentWindowSum += numbers[i] - numbers[i - windowSize];
            // 更新最小窗口和
            minWindowSum = Math.min(minWindowSum, currentWindowSum);
        }

        // 猴子能获取的最大香蕉数是总和减去最小窗口和
        return total - minWindowSum;
    }
}
```

javaScript

```javascript
// 使用Node.js的readline模块来处理输入
const readline = require('readline');
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

// 读取输入
rl.on('line', (len) => {
    len = parseInt(len);
    rl.on('line', (numbers) => {
        numbers = numbers.split(' ').map(Number);
        rl.on('line', (N) => {
            N = parseInt(N);
            console.log(maxBananas(numbers, N)); // 输出猴子能获取的最大香蕉数
            rl.close();
        });
    });
});

// 计算猴子能获取的最大香蕉数
function maxBananas(numbers, N) {
    let total = numbers.reduce((acc, val) => acc + val, 0); // 计算数组中所有香蕉的总数

    if (N === numbers.length) {
        return total; // 如果N等于数组长度，猴子可以拿走所有的香蕉
    }

    let minWindowSum = Infinity; // 初始化最小窗口和为无穷大
    let currentWindowSum = 0; // 初始化当前窗口和为0
    let windowSize = numbers.length - N; // 计算窗口大小

    for (let i = 0; i < windowSize; i++) {
        currentWindowSum += numbers[i];
    }
    minWindowSum = currentWindowSum;

    for (let i = windowSize; i < numbers.length; i++) {
        currentWindowSum += numbers[i] - numbers[i - windowSize];
        minWindowSum = Math.min(minWindowSum, currentWindowSum);
```

```
    }

    return total - minWindowSum; // 猴子能获取的最大香蕉数是总和减去最小窗口和
}
```

Python

```python
import sys

# 计算猴子能获取的最大香蕉数的函数
def max_bananas(numbers, N):
    # 计算数组中所有香蕉的总数
    total = sum(numbers)

    # 如果N等于数组长度，猴子可以拿走所有的香蕉
    if N == len(numbers):
        return total

    # 初始化最小窗口和为无穷大
    min_window_sum = float('inf')
    # 初始化当前窗口和为0
    current_window_sum = 0
    # 计算窗口大小
    window_size = len(numbers) - N

    # 初始化当前窗口的和
    for i in range(window_size):
        current_window_sum += numbers[i]
    min_window_sum = current_window_sum

    # 通过滑动窗口计算最小窗口和
    for i in range(window_size, len(numbers)):
        # 窗口滑动，加上新进入窗口的元素，减去离开窗口的元素
        current_window_sum += numbers[i] - numbers[i - window_size]
        # 更新最小窗口和
        min_window_sum = min(min_window_sum, current_window_sum)

    # 猴子能获取的最大香蕉数是总和减去最小窗口和
    return total - min_window_sum


```

```
34
35
36
37
38
39
40
41
42
43
44
```

```python

# 读取数组长度
array_length = int(input())
# 读取数组，将输入的字符串分割并转换为整数列表
numbers = list(map(int, input().strip().split()))
# 读取猴子可以获取的次数
N = int(input())
# 输出猴子能获取的最大香蕉数
print(max_bananas(numbers, N))


```

C语言

```c
#include <stdio.h>
#include <limits.h>

// 计算猴子能获取的最大香蕉数
int maxBananas(int numbers[], int len, int N) {
    int total = 0; // 初始化数组总和为0
    // 计算数组中所有香蕉的总数
    for (int i = 0; i < len; i++) {
        total += numbers[i];
    }

    // 如果N等于数组长度，猴子可以拿走所有的香蕉
    if (N == len) {
        return total;
    }

    int minWindowSum = INT_MAX; // 初始化最小窗口和为最大整数值
    int currentWindowSum = 0; // 初始化当前窗口和为0
    int windowSize = len - N; // 计算窗口大小

    // 初始化窗口的和
    for (int i = 0; i < windowSize; i++) {
        currentWindowSum += numbers[i];
    }
    minWindowSum = currentWindowSum; // 将当前窗口和赋值给最小窗口和

```

```
    // 通过滑动窗口计算最小窗口和
    for (int i = windowSize; i < len; i++) {
        // 窗口滑动，加上新进入窗口的元素，减去离开窗口的元素
        currentWindowSum += numbers[i] - numbers[i - windowSize];
        // 更新最小窗口和
        if (currentWindowSum < minWindowSum) {
            minWindowSum = currentWindowSum;
        }
    }

    // 猴子能获取的最大香蕉数是总和减去最小窗口和
    return total - minWindowSum;
}

int main() {
    int len; // 读取数组长度
    scanf("%d", &len);
    int numbers[len]; // 创建数组存储每串香蕉的数量
    for (int i = 0; i < len; i++) {
        scanf("%d", &numbers[i]); // 循环读取每串香蕉的数量
    }
    int N; // 读取猴子可以获取的次数
    scanf("%d", &N);
    printf("%d\n", maxBananas(numbers, len, N)); // 输出猴子能获取的最大香蕉数
    return 0;
}
```

文章目录

我们以用例3解释题目：

解题思路

模拟计算过程

C++

Java

javaScript

Python

C语言