

华为OD机试 - 计算最接近的数 (Java & JS & Python)

原创

伏城之外

已于 2023-07-06 14:27:04 修改

651

收藏 2

版权

分类专栏:

华为OD机试AB (Java & JS & Python)

文章标签:


华为机试

算法

Java

JavaScript

Python

 华为OD机试AB (Java & JS & Python) 同时被 2 个专栏收录

¥59.90
¥99.00

3382 订阅 371 篇文章

已订阅

该专栏为热销专栏榜 第2名

题目描述

给定一个数组X和正整数K，请找出使表达式：

$$X[i] - X[i + 1] - \dots - X[i + K - 1]$$

结果最接近于数组 **中位数** 的下标 i，如果有多个 i 满足条件，请返回最大的 i。

其中，数组中位数：长度为N的数组，按照元素的值大小升序排列后，下标为 N/2 元素的值

输入描述

无

输出描述

无

备注

- 数组X的元素均为正整数
- X的长度n取值范围：2 ≤ n ≤ 1000
- K大于0且小于数组的大小
- i 的取值范围: 0 ≤ i < 1000
- 题目的排序数组X[N]的中位数是X[N/2]

用例

输入	[50,50,2,3],2
输出	1
说明	1. 中位数为50: [50,50,2,3]升序排序后变成 [2,3,50,50], 中位数为下标4/2=2的元素50 2. 计算结果为1: X [50,50,2,3] 根据题目计算X[i] - ... - X[i + k - 1] 得出三个数0 (X[0] - X[1] = 50 - 50) 、48 (X[1] - X[2] = 50 - 2) 和 -1 (X[2]-X[3] = 2 - 3) , 其中48最接近50, 因此返回下标1。

题目解析

本题应该是采用核心代码模式，非ACM模式，因此不需要我们处理输入输出。

下面代码仍然以ACM模式实现，但是会将输入输出处理和核心代码分离。考试时，只需要写出核心代码即可。

本题可以使用滑动窗口解题。

本题其实就是要遍历所有长度为k的滑窗，滑窗内部元素需要按照表达式

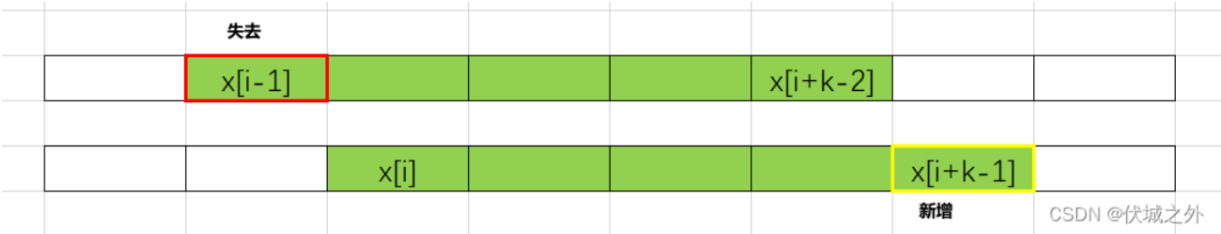
$$X[i] - X[i + 1] - \dots - X[i + K - 1]$$

来求得该滑窗对应得表达式计算结果window。

然后，求解window和数组x中位数mid的差距，差距绝对值越小，则说明二者越接近

结果最接近于数组中位数的下标 i，如果有多个 i 满足条件，请返回最大的 i。

本题的难点在于滑窗右移时，新、旧滑窗的状态转移。



假设旧滑窗的表达式计算结果为window，那么新滑窗的表达式计算结果应该是：

$$window - x[i-1] + x[i] * 2 - x[i + k - 1]$$

其中 $x[i] * 2$ 的原因是，新滑窗中 $x[i]$ 是正的，而旧滑窗中 $x[i]$ 是负的，为了将 $-x[i]$ 变为 $+x[i]$ ，则需要为 $-x[i] + x[i] * 2$

```
1 import java.util.Arrays;
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 line = sc.nextLine();
9
10    int i = line.lastIndexOf(",");
11
12    int[] x =
13        Arrays.stream(line.substring(1, i -
14    1).split(",")).mapToInt(Integer::parseInt).toArray();
15
16    int k = Integer.parseInt(line.substring(i + 1));
17
18    System.out.println(getResult(x, k));
19 }
20
21 public static int getResult(int[] x, int k) {
22     int n = x.length;
23
24     // x数组的中位数
25     int mid = Arrays.stream(x).sorted().toArray()[n / 2];
26
27     // 初始化滑窗0~k-1, window为滑窗内部元素的表达式计算结果
28     int window = x[0];
29     for (int i = 1; i < k; i++) {
30         window -= x[i];
31     }
32
33     // window和中位数的差距
34     int minDiff = Math.abs(mid - window);
35     // window滑窗起始索引
36     int idx = 0;
37
38     // 滑窗右移
39     for (int i = 1; i <= n - k; i++) {
40         // 右移一格后, 新滑窗的表达式计算结果
41         window += -x[i - 1] + 2 * x[i] - x[i + k - 1];
42
43         // 新滑窗window值和中位数的差距
44         int diff = Math.abs(mid - window);
45
46         // 结果最接近于数组中位数的下标 i , 如果有多个 i 满足条件, 请返回最大的 i
47         if (diff <= minDiff) {
```

```
46         minDiff = diff;
47         idx = i;
48     }
49 }
50
51 return idx;
52 }
53 }
```