题目描述

小华和小薇一起通过玩积木游戏学习数学。
他们有很多积木，每个积木块上都有一个数字，积木块上的数字可能相同。
小华随机拿一些积木挨着排成一排，请小薇找到这排积木中数字相同且所处位置最远的2块积木块，计算他们的距离，小薇请你帮忙替她解决这个问题。

输入描述

第一行输入为N，表示小华排成一排的积木总数。
接下来N行每行一个数字，表示小华排成一排的积木上数字。

输出描述

相同数字的积木的位置最远距离；如果所有积木数字都不相同，请返回-1。

备注

- 0<=积木上的数字<10^9
- 1<=积木长度<=10^5

用例

输入	5 1 2 3 1 4
输出	3
说明	共有5个积木，第1个积木和第4个积木数字相同，其距离为3。

输入	2 1 2
输出	-1
说明	一共有两个积木，没有积木数字相同，返回-1。

题目解析

这题第一眼看上去好像是要用 双指针 做，但是实操起来却不行。

这题的数组长度会达到10^5，因此时间复杂度要至少控制在O(n)。

我的解题思路如下：

定义一个idx对象，用于存放每个num出现的索引位置，num作为idx对象属性，num出现的索引位置作为idx[num]的数组的元素。

然后遍历nums数组（即从第二行开始输入的数的集合），开始录入num的索引位置到idx对象中。

统计完后，开始遍历idx对象属性，即每个num，然后先判断idx[num]的数组长度是否大于1，若不大于，则不考虑，若大于，则用idx[num]数组的最后一个索引位置 减去 idx[num]数组的第一个索引位置。按照上面逻辑，计算出最大的索引差作为题解。

若没有符合要求的，则返回-1。

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,
});
```

题目解析

这题第一眼看上去好像是要用 双指针 做，但是实操起来却不行。

这题的数组长度会达到10^5，因此时间复杂度要至少控制在O(n)。

我的解题思路如下：

定义一个idx对象，用于存放每个num出现的索引位置，num作为idx对象属性，num出现的索引位置作为idx[num]的数组的元素。

然后遍历nums数组（即从第二行开始输入的数的集合），开始录入num的索引位置到idx对象中。

统计完后，开始遍历idx对象属性，即每个num，然后先判断idx[num]的数组长度是否大于1，若不大于，则不考虑，若大于，则用idx[num]数组的最后一个索引位置 减去 idx[num]数组的第一个索引位置。按照上面逻辑，计算出最大的索引差作为题解。

若没有符合要求的，则返回-1。

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,
});

const lines = [];
let n;
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 1) {
    n = lines[0] - 0;
  }

  if (n && lines.length === n + 1) {
    lines.shift();
    const arr = lines.map(Number);
    console.log(getResult(arr));

    lines.length = 0;
  }
});

function getResult(nums) {
  const idx = {};

  for (let i = 0; i < nums.length; i++) {
    const num = nums[i];
    idx[num] ? idx[num].push(i) : (idx[num] = [i]);
  }

  let ans = -1;
  for (let k in idx) {
    if (idx[k].length > 1) {
      ans = Math.max(ans, idx[k].at(-1) - idx[k][0]);
    }
  }

  return ans;
}
```

Java算法源码

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

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

    int n = sc.nextInt();

    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
      arr[i] = sc.nextInt();
    }

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

  public static int getResult(int[] arr) {
    HashMap<Integer, LinkedList<Integer>> idx = new HashMap<>();

    for (int i = 0; i < arr.length; i++) {
      int num = arr[i];
      idx.putIfAbsent(num, new LinkedList<>());
      idx.get(num).add(i);
    }
```

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,
});

const lines = [];
let n;
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 1) {
    n = lines[0] - 0;
  }

  if (n && lines.length === n + 1) {
    lines.shift();
    const arr = lines.map(Number);
    console.log(getResult(arr));

    lines.length = 0;
  }
});

function getResult(nums) {
  const idx = {};

  for (let i = 0; i < nums.length; i++) {
    const num = nums[i];
    idx[num] ? idx[num].push(i) : (idx[num] = [i]);
  }

  let ans = -1;
  for (let k in idx) {
    if (idx[k].length > 1) {
      ans = Math.max(ans, idx[k].at(-1) - idx[k][0]);
    }
  }

  return ans;
}
```

Java算法源码

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

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

    int n = sc.nextInt();

    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
      arr[i] = sc.nextInt();
    }

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

  public static int getResult(int[] arr) {
    HashMap<Integer, LinkedList<Integer>> idx = new HashMap<>();

    for (int i = 0; i < arr.length; i++) {
      int num = arr[i];
      idx.putIfAbsent(num, new LinkedList<>());
      idx.get(num).add(i);
    }

    int ans = -1;

    for (Integer k : idx.keySet()) {
      LinkedList<Integer> link = idx.get(k);
      if (link.size() > 1) {
        ans = Math.max(ans, link.getLast() - link.getFirst());
      }
    }

    return ans;
  }
}
```

Python算法源码

```python
# 输入获取
```

Python算法源码

```python
# 输入获取
n = int(input())
arr = []
for i in range(n):
    arr.append(int(input()))


# 算法入口
def getResult(arr, n):
    idx = {}

    for i in range(n):
        num = arr[i]
        if idx.get(num) is None:
            idx[num] = [i]
        else:
            idx[num].append(i)

    ans = -1
    for k in idx.keys():
        if len(idx[k]) > 1:
            ans = max(ans, idx[k][-1] - idx[k][0])

    return ans


# 算法调用
print(getResult(arr, n))
```