华为OD机试 - 分割数组的最大差值（Java & JS & Python）

题目描述

给定一个由若干整数组成的数组nums ，可以在数组内的任意位置进行分割，将该数组分割成两个非空子数组（即左数组和右数组），分别对子数组求和得到两个值，计算这两个值的差值，请输出所有分割方案中，差值最大的值。

输入描述

第一行 输入数组 中元素个数n，1 < n ≤ 100000
第二行输入数字序列，以空格进行分隔，数字取值为4字节整数

输出描述

输出差值的最大取值

用例

输入	6 1 -2 3 4 -9 7
输出	10
说明	将数组 nums 划分为两个非空数组的可行方案有: 左数组 = [1] 且 右数组 = [-2,3,4,-9,7]，和的差值 = | 1 - 3 | = 2 左数组 = [1,-2] 且 右数组 = [3,4,-9,7]，和的差值 = | -1 - 5 | =6 左数组 = [1,-2,3] 且 右数组 = [4,-9,7]，和的差值 = | 2 - 2 | = 0 左数组 = [1,-2,3,4] 且右数组=[-9,7]，和的差值 = | 6 - (-2) | = 8， 左数组 = [1,-2,3,4,-9] 且 右数组 = [7]，和的差值 = | -3 - 7| = 10最大的差值为10

题目解析

我的解题思路如下：

定义一个leftSum，用于统计左数组的和，初始为0

定义一个rightSum，用于统计右数组的和，初始为sum(nums)

然后，开始从 i = 0，开始遍历输入的数组nums的每一个元素nums[i]，

leftSum += nums[i]

rightSum -= nums[i]

然后计算两个和的差值绝对值diff，比较最大的maxDiff，若大于maxDiff，则maxDiff = diff

之后，在 i++，循环上面逻辑，直到 i = nums.length-2，因为左右数组不能为空，因此右数组至少有一个nums[nums.length-1]元素

上面算法是一个O(n)时间复杂度，对于1 < n ≤ 100000数量级而言，不会超时。

```java
import java.util.Arrays;
import java.util.Scanner;

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

    int n = Integer.parseInt(sc.nextLine());
    long[] nums = Arrays.stream(sc.nextLine().split(" ")).mapToLong(Long::parseLong).toArray();

    System.out.println(getResult(nums, n));
  }

  public static long getResult(long[] nums, int n) {
    long leftSum = 0;
    long rightSum = Arrays.stream(nums).sum();

    long maxDiff = 0;

    for (int i = 0; i < n - 1; i++) {
      leftSum += nums[i];
      rightSum -= nums[i];

      long diff = Math.abs(leftSum - rightSum);
      if (diff > maxDiff) maxDiff = diff;
    }

    return maxDiff;
  }
}
```