题目描述

为了解新学期学生暴涨的问题,小乐村要建立所新学校,
考虑到学生上学安全问题,需要所有学生家到学校的距离最短。
假设学校和所有学生家都走在一条直线之上,请问学校建立在什么位置,
能使得到学校到各个学生家的距离和最短。

输入描述

第一行: 整数 n 取值范围 [1 ,1000],表示有 n户家庭。
第二行: 一组整数 m 取值范围 [0, 10000]， 表示每户家庭的位置，所有家庭的位置都不相同。

输出描述

一个整数，确定的学校的位置。
如果有多个位置，则输出最小的。

用例

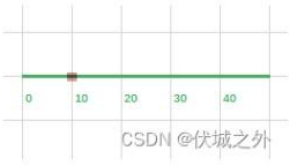
输入	5 0 20 40 10 30
输出	20
说明	20到各个家庭的距离分别为20 0 20 10 10，总和为60，最小

输入	1 20
输出	20
说明	只有一组数据，20到20距离最小，为0

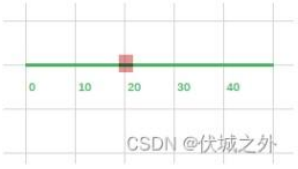
输入	2 0 20
输出	0
说明	有多个地方可选，但是0数值最小

题目解析

0 + 10 + 20 + 30 + 40 = 100



10 + 0 + 10 +20 + 30 = 70



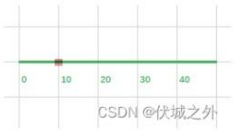
20 + 10 +0 + 10 + 20 = 50

将学校建在30，40点上，其实和建在10，0点上相同，此处不再赘述。

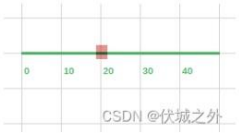
因此，我们发现，将学校建在所有学生家位置的共同中心点位置的距离最短

题目解析

0 + 10 + 20 + 30 + 40 = 100



10 + 0 + 10 +20 + 30 = 70



20 + 10 +0 + 10 + 20 = 50

将学校建在30，40点上，其实和建在10，0点上相同，此处不再赘述。

因此，我们发现，将学校建在所有学生家位置的共同中心点位置的距离最短。

本题其实就是 中位数 定理。

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

  if (lines.length === 2) {
    const n = lines[0] - 0;
    const arr = lines[1].split(" ").map(Number);

    console.log(getResult(arr, n));
    lines.length = 0;
  }
});

function getResult(arr, n) {
  arr.sort((a, b) => a - b);
  const len = arr.length;

  if (len % 2 === 0) {
    const mid = len / 2;
    return arr[mid - 1]; // 取最小的
  } else {
    return arr[Math.floor(len / 2)];
  }
}
```

Java算法源码

```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 = 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) {
    Arrays.sort(arr);

    int len = arr.length;
```

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

  if (lines.length === 2) {
    const n = lines[0] - 0;
    const arr = lines[1].split(" ").map(Number);

    console.log(getResult(arr, n));
    lines.length = 0;
  }
});

function getResult(arr, n) {
  arr.sort((a, b) => a - b);
  const len = arr.length;

  if (len % 2 === 0) {
    const mid = len / 2;
    return arr[mid - 1]; // 取最小的
  } else {
    return arr[Math.floor(len / 2)];
  }
}
```

Java算法源码

```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 = 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) {
    Arrays.sort(arr);

    int len = arr.length;
    if (len % 2 == 0) {
      int mid = len / 2;
      return arr[mid - 1];
    } else {
      return arr[len / 2];
    }
  }
}
```

Python算法源码

```python
import math

# 输入获取
n = int(input())
arr = list(map(int, input().split()))


# 算法入口
def getResult(arr, n):
    arr.sort()

    if n % 2 == 0:
        mid = int(n / 2)
        return arr[mid - 1]
    else:
        return arr[math.floor(n / 2)]


# 算法调用
print(getResult(arr, n))
```

Python算法源码

```python
import math

# 输入获取
n = int(input())
arr = list(map(int, input().split()))


# 算法入口
def getResult(arr, n):
    arr.sort()

    if n % 2 == 0:
        mid = int(n / 2)
        return arr[mid - 1]
    else:
        return arr[math.floor(n / 2)]


# 算法调用
print(getResult(arr, n))
```