

【华为OD机考 统一考试机试C卷】最大N个数与最小N个数的和（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

给定一个数组，编写一个函数来计算它的最大N个数与最小N个数的和。你需要对数组进行去重。

说明：

- 数组中数字范围[0, 1000]
- 最大N个数与最小N个数不能有重叠，如有**重叠，输入非法**返回-1
- 输入非法返回-1

输入描述

- 第一行输入M， M标识数组大小
- 第二行输入M个数，标识数组内容
- 第三行输入N， N表达需要计算的最大、最小N个数

输出描述

输出最大N个数与最小N个数的和

用例

输入	5 95 88 83 64 100 2
输出	342
说明	最大2个数[100,95],最小2个数[83,64], 输出为342。

输入	5 3 2 3 4 2 2
输出	-1
说明	最大2个数[4,3],最小2个数[3,2], 有重叠输出为-1。

C++

```
1  #include <iostream>
2  #include <algorithm>
3  #include <unordered_set>
4  using namespace std;
5
6  int getSumOfMaxAndMinN(int size, int nums[], int n) {
7      unordered_set<int> numSet; // 使用无序集合去重
8
9      // 将数组中的数字插入到集合中, 同时判断数字是否符合要求
10     for (int i = 0; i < size; i++) {
11         if (nums[i] < 0 || nums[i] > 1000) return -1; // 不符合要求, 返回-1
12         numSet.insert(nums[i]);
13     }
14
15     if (numSet.size() < n * 2) return -1; // 数组中不足2n个不同的数字, 返回-1
```

```
16
17     int distinctNums[numSet.size()]; // 存储去重后的数字
18     int index = 0;
19     for (auto val : numSet) {
20         distinctNums[index++] = val;
21     }
22
23     sort(distinctNums, distinctNums + numSet.size()); // 排序
24
25     int left = 0;
26     int right = numSet.size() - 1;
27     int sum = 0;
28
29     while (n > 0) {
30         sum += distinctNums[left] + distinctNums[right]; // 计算最大和最小n个数字之和
31         left++;
32         right--;
33         n--;
34     }
35
36     return sum;
37 }
38
39 int main() {
40     int size;
41     cin >> size;
42
43     int nums[size];
44     for (int i = 0; i < size; i++) {
45         cin >> nums[i];
46     }
47
48     int n;
49     cin >> n;
50
51     cout << getSumOfMaxAndMinN(size, nums, n) << endl;
52
53     return 0;
54 }
```

Java

```
1 import java.util.*;
2
3 public class Main {
4     public static int getSumOfMaxAndMinN(int size, int[] nums, int n) {
5         Set<Integer> numSet = new HashSet<>(); // 使用HashSet去重
6
7         // 将数组中的数字插入到集合中, 同时判断数字是否符合要求
8         for (int i = 0; i < size; i++) {
9             if (nums[i] < 0 || nums[i] > 1000) return -1; // 不符合要求, 返回-1
10            numSet.add(nums[i]);
11        }
12
13        if (numSet.size() < n * 2) return -1; // 数组中不足2n个不同的数字, 返回-1
14
15        int[] distinctNums = new int[numSet.size()]; // 存储去重后的数字
16        int index = 0;
17        for (int val : numSet) {
18            distinctNums[index++] = val;
19        }
20
21        Arrays.sort(distinctNums); // 排序
22
23        int left = 0;
24        int right = numSet.size() - 1;
25        int sum = 0;
26
27        while (n > 0) {
28            sum += distinctNums[left] + distinctNums[right]; // 计算最大和最小n个数字之和
29            left++;
30            right--;
31            n--;
32        }
33
34        return sum;
35    }
36
37    public static void main(String[] args) {
38        Scanner scanner = new Scanner(System.in);
39    }
```

```

40     int size = scanner.nextInt();
41
42     int[] nums = new int[size];
43     for (int i = 0; i < size; i++) {
44         nums[i] = scanner.nextInt();
45     }
46
47     int n = scanner.nextInt();
48
49     System.out.println(getSumOfMaxAndMinN(size, nums, n));
50 }
}

```

JavaScript

```

1  const readline = require('readline');
2
3  const rl = readline.createInterface({
4      input: process.stdin,
5      output: process.stdout
6  });
7
8  function getSumOfMaxAndMinN(size, nums, n) {
9      const numSet = new Set(nums);
10
11     for (let num of numSet) {
12         if (num < 0 || num > 1000) {
13             return -1;
14         }
15     }
16
17     if (numSet.size < n * 2) {
18         return -1;
19     }
20
21     const distinctNums = [...numSet].sort((a, b) => a - b);
22
23     let left = 0;
24     let right = distinctNums.length - 1;
25     let sum = 0;
26

```

复制

```

26
27
28 while (n > 0) {
29     sum += distinctNums[left] + distinctNums[right];
30     left++;
31     right--;
32     n--;
33 }
34
35 return sum;
36 }
37
38 rl.on('line', (input) => {
39     const inputArr = input.split(' ').map(Number);
40     if (inputArr.length === 1) {
41         const size = inputArr[0];
42         rl.once('line', (line) => {
43             const nums = line.split(' ').map(Number);
44             rl.once('line', (nLine) => {
45                 const n = Number(nLine);
46                 console.log(getSumOfMaxAndMinN(size, nums, n));
47                 rl.close();
48             });
49         });
50     }
51 });

```

Python

```

1 from typing import List
2 def getSumOfMaxAndMinN(size: int, nums: List[int], n: int) -> int:
3     numSet = set(nums) # 使用集合去重
4
5     # 将数组中的数字插入到集合中, 同时判断数字是否符合要求
6     for num in numSet:
7         if num < 0 or num > 1000:
8             return -1 # 不符合要求, 返回-1
9
10    if len(numSet) < n * 2:
11        return -1 # 数组中不足2n个不同的数字, 返回-1
12
13

```

```

13 distinctNums = sorted(list(numSet)) # 排序
14
15 left = 0
16 right = len(numSet) - 1
17 sum = 0
18
19 while n > 0:
20     sum += distinctNums[left] + distinctNums[right] # 计算最大和最小n个数字之和
21     left += 1
22     right -= 1
23     n -= 1
24
25 return sum
26
27 if __name__ == '__main__':
28     size = int(input())
29     nums = list(map(int, input().split()))
30     n = int(input())
31
32     print(getSumOfMaxAndMinN(size, nums, n))

```

C语言

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  // 比较函数, 用于qsort
5  int compare(const void *a, const void *b) {
6      return (*(int *)a - *(int *)b);
7  }
8
9  // 函数用于计算最大N个数与最小W个数的和
10 int getSumOfMaxAndMinN(int size, int nums[], int n) {
11     if (n * 2 > size) return -1; // 如果最大最小W个数总和大于数组大小, 则输入非法
12
13     qsort(nums, size, sizeof(int), compare); // 对数组进行排序
14
15     int distinct[size]; // 用于存储去重后的数字
16     int distinctSize = 0; // 去重后的数组大小
17     int prev = -1; // 用于记录前一个数字
18
19     for (int i = 0; i < size; i++) {
20         if (nums[i] != prev) {
21             distinct[distinctSize] = nums[i];
22             distinctSize++;
23             prev = nums[i];
24         }
25     }
26
27     int left = 0, right = distinctSize - 1, sum = 0;
28     while (n > 0) {
29         sum += distinct[left] + distinct[right];
30         left++;
31         right--;
32         n--;
33     }
34
35     return sum;
36 }
37
38 int main() {
39     int size, n;
40     scanf("%d %d", &size, &n);
41     int nums[size];
42     for (int i = 0; i < size; i++) {
43         scanf("%d", &nums[i]);
44     }
45
46     int result = getSumOfMaxAndMinN(size, nums, n);
47     if (result == -1) {
48         printf("Input is illegal\n");
49     } else {
50         printf("%d\n", result);
51     }
52
53     return 0;
54 }

```

```

18
19 // 遍历数组, 进行去重
20 for (int i = 0; i < size; i++) {
21     if (nums[i] < 0 || nums[i] > 1000) return -1; // 数字范围检查
22     if (nums[i] != prev) {
23         distinct[distinctSize++] = nums[i];
24         prev = nums[i];
25     }
26 }
27
28 if (distinctSize < n * 2) return -1; // 去重后数字不足以获取最大最小W个数
29
30 int sum = 0;
31 // 计算最大和最小n个数字之和
32 for (int i = 0; i < n; i++) {
33     sum += distinct[i] + distinct[distinctSize - 1 - i];
34 }
35
36 return sum;
37 }
38
39 int main() {
40     int size, n;
41     scanf("%d", &size);
42
43     int nums[size];
44     for (int i = 0; i < size; i++) {
45         scanf("%d", &nums[i]);
46     }
47
48     scanf("%d", &n);
49
50     printf("%d\n", getSumOfMaxAndMinN(size, nums, n));
51
52     return 0;
53 }

```

完整用例

用例1

5
95 88 83 64 100
2

用例2

5
3 2 3 4 2
2

用例3

5
-1 0 1 2 3
2

用例4

5
1001 200 300 400 500
2

用例5

5
10 20 30 40 50
6

用例6

5
5 5 5 5 5
1

用例7

5
1 2 3 4 5
2

用例8

5
10 20 30 40 50
0

用例9

4
10 20 30 40
2

用例10

5
0 10 20 0 10
1

文章目录

- 华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷
- 题目描述
- 输入描述
- 输出描述
- 用例
- C++
- Java
- JavaScript
- Python
- C语言
- 完整用例
 - 用例1

用例2
用例3
用例4
用例5
用例6
用例7
用例8
用例9
用例10

机考真题 华为OD



CSDN @算法大师