【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++

```cpp
#include <iostream>
#include <algorithm>
#include <unordered_set>
using namespace std;

int getSumOfMaxAndMinN(int size, int nums[], int n) {
    unordered_set<int> numSet;  // 使用无序集合去重

    // 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
    for (int i = 0; i < size; i++) {
        if (nums[i] < 0 || nums[i] > 1000) return -1;  // 不符合要求，返回-1
        numSet.insert(nums[i]);
    }

    if (numSet.size() < n * 2) return -1;  // 数组中不足2n个不同的数字，返回-1
```

```cpp

    int distinctNums[numSet.size()];  // 存储去重后的数字
    int index = 0;
    for (auto val : numSet) {
        distinctNums[index++] = val;
    }

    sort(distinctNums, distinctNums + numSet.size());  // 排序

    int left = 0;
    int right = numSet.size() - 1;
    int sum = 0;

    while (n > 0) {
        sum += distinctNums[left] + distinctNums[right];  // 计算最大和最小n个数字之和
        left++;
        right--;
        n--;
    }

    return sum;
}

int main() {
    int size;
    cin >> size;

    int nums[size];
    for (int i = 0; i < size; i++) {
        cin >> nums[i];
    }

    int n;
    cin >> n;

    cout << getSumOfMaxAndMinN(size, nums, n) << endl;

    return 0;
}
```

Java

```java
import java.util.*;

public class Main {
    public static int getSumOfMaxAndMinN(int size, int[] nums, int n) {
        Set<Integer> numSet = new HashSet<>(); // 使用HashSet去重

        // 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
        for (int i = 0; i < size; i++) {
            if (nums[i] < 0 || nums[i] > 1000) return -1; // 不符合要求，返回-1
            numSet.add(nums[i]);
        }

        if (numSet.size() < n * 2) return -1; // 数组中不足2n个不同的数字，返回-1

        int[] distinctNums = new int[numSet.size()]; // 存储去重后的数字
        int index = 0;
        for (int val : numSet) {
            distinctNums[index++] = val;
        }

        Arrays.sort(distinctNums); // 排序

        int left = 0;
        int right = numSet.size() - 1;
        int sum = 0;

        while (n > 0) {
            sum += distinctNums[left] + distinctNums[right]; // 计算最大和最小n个数字之和
            left++;
            right--;
            n--;
        }

        return sum;
    }

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

```

```
        int size = scanner.nextInt();

        int[] nums = new int[size];
        for (int i = 0; i < size; i++) {
            nums[i] = scanner.nextInt();
        }

        int n = scanner.nextInt();

        System.out.println(getSumOfMaxAndMinN(size, nums, n));
    }
}
```

JavaScript

复制

```
const readline = require('readline');

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

function getSumOfMaxAndMinN(size, nums, n) {
  const numSet = new Set(nums);

  for (let num of numSet) {
    if (num < 0 || num > 1000) {
      return -1;
    }
  }

  if (numSet.size < n * 2) {
    return -1;
  }

  const distinctNums = [...numSet].sort((a, b) => a - b);

  let left = 0;
  let right = distinctNums.length - 1;
  let sum = 0;
```

```
  while (n > 0) {
    sum += distinctNums[left] + distinctNums[right];
    left++;
    right--;
    n--;
  }

  return sum;
}

rl.on('line', (input) => {
  const inputArr = input.split(' ').map(Number);
  if (inputArr.length === 1) {
    const size = inputArr[0];
    rl.once('line', (line) => {
      const nums = line.split(' ').map(Number);
      rl.once('line', (nLine) => {
        const n = Number(nLine);
        console.log(getSumOfMaxAndMinN(size, nums, n));
        rl.close();
      });
    });
  }
});
```

Python

```
from typing import List
def getSumOfMaxAndMinN(size: int, nums: List[int], n: int) -> int:
    numSet = set(nums)  # 使用集合去重

    # 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
    for num in numSet:
        if num < 0 or num > 1000:
            return -1  # 不符合要求，返回-1

    if len(numSet) < n * 2:
        return -1  # 数组中不足2n个不同的数字，返回-1
```

```
    distinctNums = sorted(list(numSet))  # 排序

    left = 0
    right = len(numSet) - 1
    sum = 0

    while n > 0:
        sum += distinctNums[left] + distinctNums[right]  # 计算最大和最小n个数字之和
        left += 1
        right -= 1
        n -= 1

    return sum

if __name__ == '__main__':
    size = int(input())
    nums = list(map(int, input().split()))
    n = int(input())

    print(getSumOfMaxAndMinN(size, nums, n))
```

C语言

```
#include <stdio.h>
#include <stdlib.h>

// 比较函数，用于qsort
int compare(const void *a, const void *b) {
    return (*(int *)a - *(int *)b);
}

// 函数用于计算最大N个数与最小W个数的和
int getSumOfMaxAndMinN(int size, int nums[], int n) {
    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

    qsort(nums, size, sizeof(int), compare); // 对数组进行排序

    int distinct[size]; // 用于存储去重后的数字
    int distinctSize = 0; // 去重后的数组大小
    int prev = -1; // 用于记录前一个数字
```

```
    // 遍历数组，进行去重
    for (int i = 0; i < size; i++) {
        if (nums[i] < 0 || nums[i] > 1000) return -1; // 数字范围检查
        if (nums[i] != prev) {
            distinct[distinctSize++] = nums[i];
            prev = nums[i];
        }
    }

    if (distinctSize < n * 2) return -1; // 去重后数字不足以获取最大最小W个数

    int sum = 0;
    // 计算最大和最小n个数字之和
    for (int i = 0; i < n; i++) {
        sum += distinct[i] + distinct[distinctSize - 1 - i];
    }

    return sum;
}

int main() {
    int size, n;
    scanf("%d", &size);

    int nums[size];
    for (int i = 0; i < size; i++) {
        scanf("%d", &nums[i]);
    }

    scanf("%d", &n);

    printf("%d\n", getSumOfMaxAndMinN(size, nums, n));

    return 0;
}
```

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

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