

【华为OD机考 统一考试机试C卷】求幸存数之和 (C++ Java JavaScript Python)

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

给一个正整数数列 nums，一个跳数 jump，及幸存数量 left。

运算过程为：从索引0的位置开始向后跳，中间跳过 J 个数字，命中索引为 J+1 的数字，该数被敲出，并从该点起跳，以此类推，直到幸存 left 个数为止，然后返回幸存数之和。

约束：

- 0是第一个起跳点
- 起跳点和命中点之间间隔 jump 个数字，已被敲出的数字不计入在内。
- 跳到末尾时无缝从头开始（循环查找），并可以多次循环。
- 若起始时 left > len(nums) 则无需跳数处理过程。

方法设计：

```
1 | * @param nums 正整数数列，长度范围 [1, 10000]
2 | * @param jump 跳数，范围 [1, 10000]
3 |
```

```
~
4  * @param left 幸存数量，范围 [0, 10000]
5  * @return 幸存数之和
6  int sumOfLeft(int[] nums, int jump, int left){
7
8  }
```

输入描述

第一行输入正整数数列

第二行输入跳数

第三行输入幸存数量

输出描述

输出幸存数之和

用例

输入	1,2,3,4,5,6,7,8,9 4 3
输出	13
说明	从1（索引为0）开始起跳，中间跳过 4 个数字，因此依次删除 6,2,8,5,4,7。剩余1,3,9，返回和为13

解题思路

本题考试时为Lettoce模式,无需自己获取输入数据。

本题主要是模拟操作，按照每次跳数的位置，从数列中删掉跳到的数组，直到剩余幸存数量的数字。

C++

```
1  #include <iostream>
2  #include <list>
3  #include <numeric>
4
```

```

4  #include <sstream>
5  #include <string>
6  #include <vector>
7  #include <numeric>
8  using namespace std;
9  // 计算幸存数之和
10 int sumOfLeft(int nums[], int jump, int left, int length) {
11     // 如果幸存数量大于等于数组长度, 则直接返回数组元素之和
12     if (left >= length) {
13         return accumulate(nums, nums + length, 0);
14     }
15
16     // 使用vector存储数组元素, 方便删除操作
17     vector<int> lst(nums, nums + length);
18
19     // 初始化起跳点索引为0
20     int index = 0;
21     // 当列表大小大于幸存数量时, 执行删除操作
22     while (lst.size() > left) {
23         // 计算下一个要删除元素的索引
24         index = (index + jump + 1) % lst.size();
25         // 删除计算出的索引处的元素
26         lst.erase(lst.begin() + index);
27         // 由于删除元素后, 列表会缩短, 下一个起跳点应当向向前移动一位
28         index = index - 1 ;
29     }
30
31     // 计算并返回剩余元素之和
32     return accumulate(lst.begin(), lst.end(), 0);
33 }
34
35 int main() {
36     string line;
37     vector<int> nums;
38     int jump, left;
39
40     // 读取一行输入, 按逗号分割, 转换为整数数组
41     getline(cin, line);
42     stringstream ss(line);
43     string num;
44     --

```

```

45 while (getline(ss, num, ',')) {
46     nums.push_back(stoi(num));
47 }
48
49 // 读取跳数
50 cin >> jump;
51 // 读取幸存数量
52 cin >> left;
53
54 // 输出幸存数之和
55 cout << sumOfLeft(&nums[0], jump, left, nums.size()) << endl;
56
57 return 0;
}

```

Java

```

1 import java.util.*;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         Scanner sc = new Scanner(System.in);
7
8         // 读取一行输入, 按逗号分割, 转换为整数数组
9         int[] nums = Arrays.stream(sc.nextLine().split(",")).mapToInt(Integer::parseInt).toArray();
10        // 读取跳数
11        int jump = Integer.parseInt(sc.nextLine());
12        // 读取幸存数量
13        int left = Integer.parseInt(sc.nextLine());
14
15        // 输出幸存数之和
16        System.out.println(sumOfLeft(nums, jump, left));
17    }
18
19    public static int sumOfLeft(int[] nums, int jump, int left) {
20        // 如果幸存数量大于等于数组长度, 则直接返回数组元素之和
21        if (left >= nums.length) {
22            return Arrays.stream(nums).sum();
23        }
24    }
25 }

```

```

24
25 // 使用LinkedList存储数组元素, 方便删除操作
26 LinkedList<Integer> list = new LinkedList<>();
27 for (int num : nums) {
28     list.add(num);
29 }
30
31 // 初始化起跳点索引为0
32 int index = 0;
33 // 当列表大小大于幸存数量时, 执行删除操作
34 while (list.size() > left) {
35     // 计算下一个要删除元素的索引
36     index = index + jump + 1;
37     // 为了实现循环跳跃, 索引可能会超出列表大小, 因此取模
38     index = index % list.size();
39     // 删除计算出的索引处的元素
40     list.remove(index);
41     // 由于删除元素后, 列表会缩短, 下一个起跳点应当向前移动一位
42     index = index - 1;
43 }
44
45 // 计算并返回剩余元素之和
46 return list.stream().mapToInt(Integer::intValue).sum();
47 }
48 }

```

javaScript

```

1 // 导入必要的库
2 const readline = require('readline');
3
4 // 创建readline接口实例
5 const rl = readline.createInterface({
6     input: process.stdin,
7     output: process.stdout
8 });
9
10 // 读取输入
11 rl.on('line', (line) => {
12     // 根据输入行数, 分别处理
13

```

```

13  if (!this.nums) {
14      // 读取一行输入, 按逗号分割, 转换为整数数组
15      this.nums = line.split(',').map(Number);
16  } else if (!this.jump) {
17      // 读取跳数
18      this.jump = Number(line);
19  } else if (!this.left) {
20      // 读取幸存数量
21      this.left = Number(line);
22      // 输出幸存数之和
23      console.log(sumOfLeft(this.nums, this.jump, this.left));
24      rl.close();
25  }
26  });
27
28  // 计算幸存数之和
29  function sumOfLeft(nums, jump, left) {
30      // 如果幸存数量大于等于数组长度, 则直接返回数组元素之和
31      if (left >= nums.length) {
32          return nums.reduce((acc, val) => acc + val, 0);
33      }
34
35      // 使用数组存储元素, 方便删除操作
36      let list = nums.slice();
37
38      // 初始化起跳点索引为0
39      let index = 0;
40      // 当列表大小大于幸存数量时, 执行删除操作
41      while (list.length > left) {
42          // 计算下一个要删除元素的索引
43          index = (index + jump + 1) % list.length;
44          // 删除计算出的索引处的元素
45          list.splice(index, 1);
46          // 由于删除元素后, 列表会缩短, 下一个起跳点应当向前移动一位
47          index = index - 1 ;
48      }
49
50      // 计算并返回剩余元素之和
51      return list.reduce((acc, val) => acc + val, 0);
52  }

```

Python

```
1 def sum_of_left(nums, jump, left):
2     # 如果幸存数量大于等于数组长度, 则直接返回数组元素之和
3     if left >= len(nums):
4         return sum(nums)
5
6     # 使用列表存储数组元素, 方便删除操作
7     lst = nums[:]
8
9     # 初始化起跳点索引为0
10    index = 0
11    # 当列表大小大于幸存数量时, 执行删除操作
12    while len(lst) > left:
13        # 计算下一个要删除元素的索引
14        index = (index + jump + 1) % len(lst)
15        # 删除计算出的索引处的元素
16        del lst[index]
17        # 由于删除元素后, 列表会缩短, 下一个起跳点应当向前移动一位
18        index = index - 1
19
20    # 计算并返回剩余元素之和
21    return sum(lst)
22
23 # 读取一行输入, 按逗号分割, 转换为整数数组
24 nums = list(map(int, input().split(',')))
25 # 读取跳数
26 jump = int(input())
27 # 读取幸存数量
28 left = int(input())
29
30 # 输出幸存数之和
31 print(sum_of_left(nums, jump, left))
```

完整用例

用例1

1,2,3,4,5,6,7,8,9,10

2

4

用例2

1,2,3,4,5

6

2

用例3

1,2,3,4,5

1

5

用例4

1,2,3,4,5,6,7,8,9,10

3

0

用例5

1

1

1

用例6

1,2,3,4,5,6,7,8,9,10

1

3

用例7

1,2,3,4,5,6,7,8,9,10

10

用例8

2,4,6,8,10,12
3
2

用例9

1,2
1
1

用例10

1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
14
3

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
 - 解题思路
 - C++
 - Java
 - javaScript
 - Python
 - 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