信息学奥赛笔记13

——前缀和专题测试的T1, T2。

[U406291]中心下标(H)https://www.luogu.com.cn/problem/U406291?contestId=158718

题目描述

给你一个长度为n整数数组`a`，请计算数组的 **中心下标** 。

数组 **中心下标** 是数组的一个下标，其左侧所有元素相加的和等于右侧所有元素相加的和。

如果中心下标位于数组最左端，那么左侧数之和视为 `0` ，因为在下标的左侧不存在元素。这一点对于中心下标位于数组最右端同样适用。

如果数组有多个中心下标，输出 **最靠近左边** 的那一个。如果数组不存在中心下标，输出 `-1` 。

输入格式

第一行一个正整数n，表示数组的长度

第二行n个整数，表示数组`a`

输出格式

一个整数，代表中心下标，如果没有则输出`-1`

样例 #1

样例输入 #1

```
6
1 7 3 6 5 6
```

样例输出 #1

```
3
```

提示

对于30%的数据，有$1 \leq n \leq 10^4$， $-10^3 \leq a[i] \leq 10^3$

对于100%的数据，有$1 \leq n \leq 10^6$， $-10^{12} \leq a[i] \leq 10^{12}$

样例解释： 在下标为3时，左边和为$1+7+3=11$， 右边和为$5+6=11$。

30分做法(暴力搜索)

我们只需要按照题目的意思，模拟一遍，暴力搜索一个下标为中心下标，然后检查它的左边和是否等于右边和即可，开long long就可以拿90分。

```cpp
#include <iostream>
#include <vector>
#include <cmath>
#include <climits>
using namespace std;
int main() {
    int n;
    cin >> n;
    vector<long long> nums(n);
    for (int i = 0; i < n; i++) {
        cin >> nums[i];
    }
    for (int i = 0; i < n ; i++) {
        long long sum1 = 0, sum2 = 0; //sum1求左边和，sum2求右边和
        for (int j = 0; j < i; j++) {
            sum1 += nums[j];
        }
        for (int j = i + 1; j < n; j++) {
            sum2 += nums[j];
        }
        if (sum1 == sum2) { //左边和 == 右边和，i就是中心下标。
            cout << i << endl;
            return 0;
        }
    }
    cout << -1 << endl;
    return 0;
}

```

100分做法(前缀和 + 滑动窗口)

那么，在暴搜的代码，咱们可以发现，对于同一段的和我们加了很多次重复的。

比如说当我们假设中心下标为`3`时，我们需要统计左边的和$a[0]+a[1]+a[2]$

但是我们在假设中心下标为`5`的时候，在统计左边的和时，依然需要计算上述式。所以想到可以利用前缀和优化。

所以我们可以将整个数组分为三段，第一段`a`是中心下标左边的和，第二段`b`是中心下标本身，第三段`c`是中心下标右边的和。

所以我们可以发现，$a+b+c=$整个数组的和`total`永远成立。

所以我们在顺序遍历中心下标的时候，可以发现，设当前中心下标为`a[i]`，当前的左边的和为`sum`，整个数组的和为`total`，可以解得，右边的和为$total-sum-a[i]$

那如果左边的和等于右边的和就是判断$total-sum-a[i]==sum$则说明当前的`i`就是中心下标

那么从当前变动到下一个中心下标的时候，左边的和会增加刚刚我们处理的`a[i]`。

```cpp
#include <iostream>
#include <vector>
#include <cmath>
#include <climits>
using namespace std;
int main() {
    int n, ans = INT_MAX;
    cin >> n;
    vector<long long> nums(n);
    long long total = 0, sum = 0;
    for (int i = 0; i < n; i++) {
        cin >> nums[i];
        total += nums[i]; //计算数组总和
    }
    int c = 2;
    for (int i = 0; i < n; i++) {
        if (total - nums[i] == 2 * sum) { //判断i是否是中心下标
            ans = min(ans, i);
        }
        sum += nums[i];  //左边的和加上当前的i
    }
    if (ans == INT_MAX) ans = -1; //如果答案没有被更新过，不存在中心下标。
    cout << ans << endl;
    return 0;
}
```

[U406783]密码破译https://www.luogu.com.cn/problem/U406783?contestId=158718

题目背景

苏联军方缴获了纳粹德国的一个加密后的密码本。

题目描述

密码本中含有密码序列和密钥，密码序列是一个 循环 的数组a，密钥是一个 非0的 整数k。

为了获得正确的密码，你需要替换掉每一个数字。所有数字会 **同时** 被替换。

- 如果 `k > 0` ，将第 `i` 个数字用 **接下来** `k` 个数字之和替换。
- 如果 `k < 0` ，将第 `i` 个数字用 **先前的** `k` 个数字之和替换。

由于 `a` 是循环的， `a[n - 1]` 下一个元素是 `a[0]` ，且 `a[0]` 前一个元素是 `a[n - 1]` 。

时间紧迫！任务重要！请你输出解密后的纳粹密码。

输入格式

第一行两个整数n, k，表示密码序列长度和密钥

第二行n个整数，代表加密后的密码。

输出格式

n个整数，代表解密后的密码序列。

样例 #1

样例输入 #1

```
4 3
5 7 1 4
```

样例输出 #1

```
12 10 16 13
```

提示

对于30%的数据，有$1 \leq |k| < n \leq 1000, -1000 \leq a[i] \leq 1000$

对于50%的数据，有$1 \leq |k| < n \leq 10^4, -10^9 \leq a[i] \leq 10^9$

对于100%的数据，有$1 \leq |k| < n \leq 10^5, -10^9 \leq a[i] \leq 10^9$

样例解释：由于$k = 3 > 0$，原数组将被替换为[7 + 1 + 4, 1 + 4 + 5, 4 + 5 + 7, 5 + 7 + 1] = [12, 10, 16, 13]

首先对于环形数组一共有两种处理方式。

第一种对循环数组的数理方式（数学）

数组当中第`n - 1`下标的下一位为`0`那么，我们可以发现，这个下标的变动无论如何都无法跳出`0`到`n - 1`。这和模运算是一样的。

所以我们可以利用下方的两个公式来计算。

$$(i+1)\%n$$
$$(i-1+n)\%n$$

那么第一个公式的作用就是往右走，走到数组的最右边$n-1$的时候$+1$就变成了0。

往左走的时候，从0需要变动到$n-1$那么我们会发现$0-1=-1$ 负数取模会导致结果为负数，那么我们可以把结果先加上n再对n取模，就变动到了最后一位$n-1$。

第二种对循环数组的数理方式（拷贝）

我们把数组进行一次完全拷贝。

变成$a[0], a[1] \dots a[n-1], a[0], a[1] \dots a[n-1]$。

那么，我们在往右加的时候从`a[n - 1]`直接往右加即可，因为`a[n - 1]`的下一个值就是$a[0]$只不过下标是n

同理，我们要往左找的时候，我们就应该取的是$i+n$也就是第二个周期的数组，这样我们从`a[n]`(其实是`a[0]`)往左的时候，直接就是数组的最后一个`a[n - 1]`。

方法一更适合需要使用到循环数组很多次，也就是循环很多次的场景。

方法二更适用于只需要循环数组2次-3次的场景，因为需要循环几次就需要将数组拷贝几遍，这会导致时空复杂度的同时增加。

那么将数组**由环化链**后这道题就变成了，给你一个整数数组，让你求每一位长度为k的和。这就是最基本的前缀和，不做赘述。

```cpp
#include<bits/stdc++.h>
using namespace std;
long long n, k;
long long a[1000005], sum[1000005];
int main() {
    cin >> n >> k;
    for (int i = 0; i < n; i++) {
        cin >> a[i];
        a[i + n] = a[i];  //拷贝数组
        sum[i] = a[i] + sum[i - 1];

    }
    for (int i = n; i < 2 * n; i++) {
        sum[i] = a[i] + sum[i- 1];  //求第二个周期的前缀和
    }
    for (int i = 0; i < n; i++) {
        if (k > 0) {  //如果K为正，往右加
           cout << sum[i + k] - sum[i] << " ";
        }
        if (k < 0) {
           cout << sum[i + n - 1] - sum[i + n + k - 1] << " ";  //注意，k为负的时候，本身就是<0的，加k即可
        }
    }
    return 0;
}
```