134.加油站

> 计算学部十大打卡活动——“龙舞编程新春会”编程打卡（2024-1-31）
>
> 134. 加油站 - 力扣（LeetCode）

一、题目

在一条环路上有 `n` 个加油站，其中第 `i` 个加油站有汽油 `gas[i]` 升。

你有一辆油箱容量无限的的汽车，从第 `i` 个加油站开往第 `i+1` 个加油站需要消耗汽油 `cost[i]` 升。你从其中的一个加油站出发，开始时油箱为空。

给定两个整数数组 `gas` 和 `cost` ，如果你可以按顺序绕环路行驶一周，则返回出发时加油站的编号，否则返回 `-1` 。如果存在解，则 **保证** 它是 **唯一** 的。

示例 1:

```
输入: gas = [1,2,3,4,5], cost = [3,4,5,1,2]
输出: 3
解释:
从 3 号加油站(索引为 3 处)出发，可获得 4 升汽油。此时油箱有 = 0 + 4 = 4 升汽油
开往 4 号加油站，此时油箱有 4 - 1 + 5 = 8 升汽油
开往 0 号加油站，此时油箱有 8 - 2 + 1 = 7 升汽油
开往 1 号加油站，此时油箱有 7 - 3 + 2 = 6 升汽油
开往 2 号加油站，此时油箱有 6 - 4 + 3 = 5 升汽油
开往 3 号加油站，你需要消耗 5 升汽油，正好足够你返回到 3 号加油站。
因此，3 可为起始索引。
```

示例 2:

```
输入: gas = [2,3,4], cost = [3,4,3]
输出: -1
解释:
你不能从 0 号或 1 号加油站出发，因为没有足够的汽油可以让你行驶到下一个加油站。
我们从 2 号加油站出发，可以获得 4 升汽油。 此时油箱有 = 0 + 4 = 4 升汽油
开往 0 号加油站，此时油箱有 4 - 3 + 2 = 3 升汽油
开往 1 号加油站，此时油箱有 3 - 3 + 3 = 3 升汽油
你无法返回 2 号加油站，因为返程需要消耗 4 升汽油，但是你的油箱只有 3 升汽油。
因此，无论怎样，你都不可能绕环路行驶一周。
```

提示:

- `gas.length == n`
- `cost.length == n`
- `1 <= n <= 105`
- `0 <= gas[i], cost[i] <= 104`

二、思路

从头到尾依次扫描每个加油站，查看其是否满足：以该加油站为起点，行驶一周后油量恰好为0。由于题目说：如果存在解，则 **保证** 它是 **唯一** 的。则只需要找到第一个满足条件的加油站即可终止。

假设从x加油站出发经过z加油站最远能到达y加油站，那么从z加油站直接出发，不可能到达y下一个加油站。因为从x出发到z加油站时肯定还有存储的油，这都到不了y的下一站，而直接从z出发刚开始是没有存储的油的，所以更不可能到达y的下一站。

则首先检查第0个加油站，并试图判断能否环绕一周；如果不能，就从第一个无法到达的加油站开始继续检查。

三、代码

```
class Solution {
public:
    int canCompleteCircuit(vector<int>& gas, vector<int>& cost)
    {
        int n = gas.size();
        int i = 0;
        while (i < n)
        {
            int sumOfGas = 0, sumOfCost = 0;
            int cnt = 0;
            while (cnt < n)
            {
                int j = (i + cnt) % n;
                sumOfGas += gas[j];
                sumOfCost += cost[j];
                if (sumOfCost > sumOfGas)//找到能走到的最后一个加油站序号
                {
                    break;
                }
                cnt++;
            }
            if (cnt == n)
            {
                return i;//找到一个满足条件的就返回
            }
            else
            {
                i = i + cnt + 1;//直接将中断点作为起始点
            }
        }
        return -1;
    }
};
```

复杂度分析

时间复杂度：$O(n)$

空间复杂度：$O(1)$