134-加油站

题述

134. 加油站

难度 中等 836

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。
```

思路

暴力

”贪心“

并不严格的算是贪心，因为没有找出局部最优，而是直接从全局最优的角度上思考问题。

直接从全局进行贪心选择，情况如下：

- 情况一：如果gas的总和小于cost总和，那么无论从哪里出发，一定是跑不了一圈的
- 情况二：rest[i] = gas[i]-cost[i]为一天剩下的油，i从0开始计算累加到最后一站，如果累加没有出现负数，说明从0出发，油就没有断过，那么0就是起点。

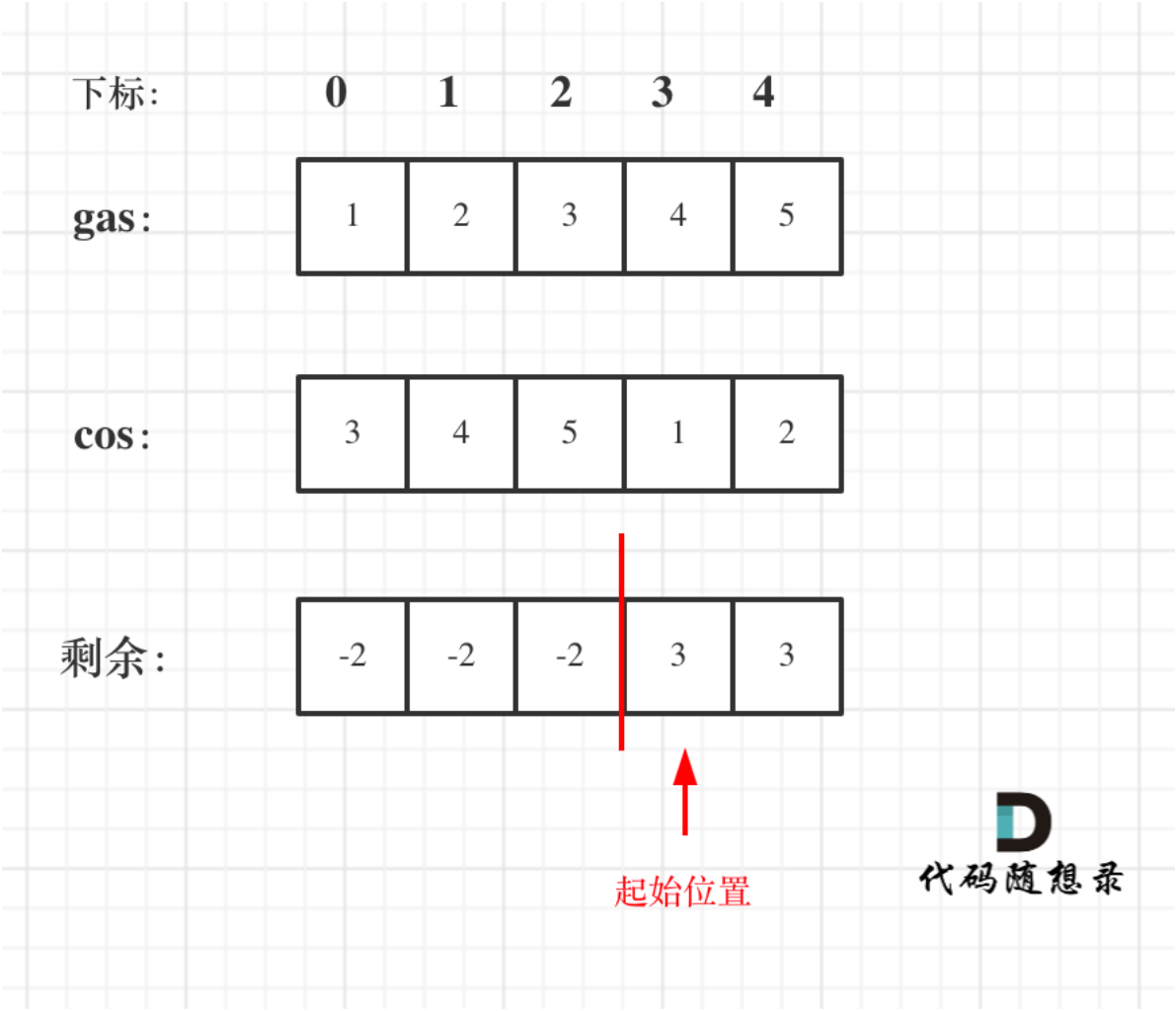
- 情况三：如果累加的最小值是负数，汽车就要从非0节点出发，从后向前，看哪个节点能这个负数填平，能把这个负数填平的节点就是出发节点。

贪心

可以换一个思路，首先如果总油量减去总消耗大于等于零那么一定可以跑完一圈，说明 各个站点的加油站 剩油量rest[i]相加一定是大于等于零的。

每个加油站的剩余量rest[i]为gas[i] - cost[i]。

i从0开始累加rest[i]，和记为curSum，一旦curSum小于零，说明[0, i]区间都不能作为起始位置，起始位置从i+1算起，再从0计算curSum。



那么为什么一旦[i，j] 区间和为负数，起始位置就可以是j+1呢，j+1后面就不会出现更大的负数?

如果出现更大的负数，就是更新j，那么起始位置又变成新的j+1了。

而且j之前出现了多少负数，j后面就会出现多少正数，因为耗油总和是大于零的（前提我们已经确定了一定可以跑完全程）。

那么局部最优：当前累加rest[j]的和curSum一旦小于0，起始位置至少要是j+1，因为从j开始一定不行。全局最优：找到可以跑一圈的起始位置。

局部最优可以推出全局最优，找不出反例，试试贪心！

题解

C++暴力

```
class Solution {
public:
    int canCompleteCircuit(vector<int>& gas, vector<int>& cost)
    {
        //暴力
        for(int i = 0; i < cost.size(); i++)
        {
            int rest=gas[i]-cost[i];    //记录剩余油量
            int index = (i+1) % cost.size();
            while(rest > 0 && index != i)
            {
                //模拟以i为起点判断能否绕行一圈
                rest += gas[index] - cost[index];
                index = (index + 1) % cost.size();
            }
            // 如果以i为起点跑一圈，剩余油量>=0，返回该起始位置
            if (rest >= 0 && index == i) return i;
        }
        return -1;
    }
};
```

贪心

```
class Solution {
public:
    int canCompleteCircuit(vector<int>& gas, vector<int>& cost)
    {
        //贪心 三种情况
        //情况一：如果gas的总和小于cost总和，那么无论从哪里出发，一定是跑不了一圈的
        //情况二：rest[i] = gas[i]-cost[i]为一天剩下的油，i从0开始计算累加到最后一站，如果累加没有出现负数，说明从0出发，油就没有断过，那么0就是起点。
        //情况三：如果累加的最小值是负数，汽车就要从非0节点出发，从后向前，看哪个节点能这个负数填平，能把这个负数填平的节点就是出发节点。
        int curSum = 0;
        int min = INT_MAX; //从起点出发，油量的最小值
        for(int i=0;i<gas.size();i++)
        {
            int rest = gas[i] - cost[i];
            curSum += rest;
            if(curSum < min)
            {
                min= curSum;
            }
        }

        if(curSum < 0) return -1;   //情况一
        if(min >= 0) return 0;      //情况二

        for(int i = gas.size() - 1;i >=0; i--)
        {
            int rest = gas[i] - cost[i];
            min += rest;
            if (min >= 0)
```

```
            {
                return i;
            }
        }
        return -1;
    }
};
```

贪心

```
class Solution {
public:
    int canCompleteCircuit(vector<int>& gas, vector<int>& cost) {
        int curSum = 0;
        int totalSum = 0;
        int start = 0;
        for (int i = 0; i < gas.size(); i++) {
            curSum += gas[i] - cost[i];
            totalSum += gas[i] - cost[i];
            if (curSum < 0) {   // 当前累加rest[i]和 curSum一旦小于0
                start = i + 1;  // 起始位置更新为i+1
                curSum = 0;     // curSum从0开始
            }
        }
        if (totalSum < 0) return -1; // 说明怎么走都不可能跑一圈了
        return start;
    }
};
```

Python

```
class Solution:
    def canCompleteCircuit(self, gas: List[int], cost: List[int]) -> int:
        start = 0
        curSum = 0
        totalSum = 0
        for i in range(len(gas)):
            curSum += gas[i] - cost[i]
            totalSum += gas[i] - cost[i]
            if curSum < 0:
                curSum = 0
                start = i + 1
        if totalSum < 0: return -1
        return start
```

思考

两种贪心算法，对于第一种贪心方法，其实我认为就是一种直接从全局选取最优的模拟操作，思路还是好巧妙的，值得学习一下

对于第二种贪心方法，才真正体现出贪心的精髓，用局部最优可以推出全局最优，进而求得起始位置。