Previous Next

134. Gas Station

March 12, 2019 | 32.1K views

Average Rating: 4.61 (38 votes)

There are N gas stations along a circular route, where the amount of gas at station i is `gas[i]`.

You have a car with an unlimited gas tank and it costs `cost[i]` of gas to travel from station i to its next station (i+1). You begin the journey with an empty tank at one of the gas stations.

Return the starting gas station's index if you can travel around the circuit once in the clockwise direction, otherwise return -1.

Note:

- If there exists a solution, it is guaranteed to be unique.
- Both input arrays are non-empty and have the same length.
- Each element in the input arrays is a non-negative integer.

Example 1:

```
Input: 
gas  = [1,2,3,4,5]
cost = [3,4,5,1,2]

Output: 3

Explanation:
Start at station 3 (index 3) and fill up with 4 unit of gas. Your tank = 0 + 4 = 4
Travel to station 4. Your tank = 4 - 1 + 5 = 8
Travel to station 0. Your tank = 8 - 2 + 1 = 7
Travel to station 1. Your tank = 7 - 3 + 2 = 6
Travel to station 2. Your tank = 6 - 4 + 3 = 5
Travel to station 3. The cost is 5. Your gas is just enough to travel back to station
Therefore, return 3 as the starting index.
```

Example 2:

```
Input: 
gas  = [2,3,4]
cost = [3,4,3]

Output: -1

Explanation:
You can't start at station 0 or 1, as there is not enough gas to travel to the next st
Let's start at station 2 and fill up with 4 unit of gas. Your tank = 0 + 4 = 4
Travel to station 0. Your tank = 4 - 3 + 2 = 3
Travel to station 1. Your tank = 3 - 3 + 3 = 3
You cannot travel back to station 2, as it requires 4 unit of gas but you only have 3.
Therefore, you can't travel around the circuit once no matter where you start.
```

Solution

Approach 1: One pass.

Intuition

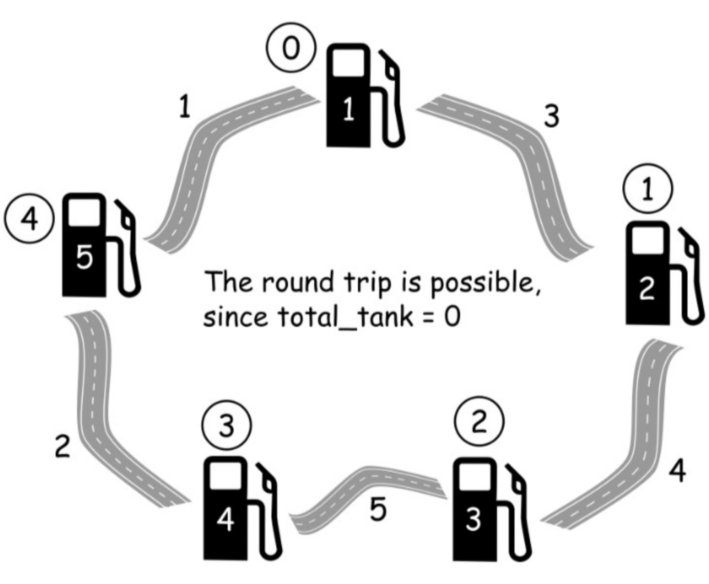
The first idea is to check every single station :

- Choose the station as starting point.
- Perform the road trip and check how much gas we have in tank at each station.

That means $\mathcal{O}(N^2)$ time complexity, and for sure one could do better.

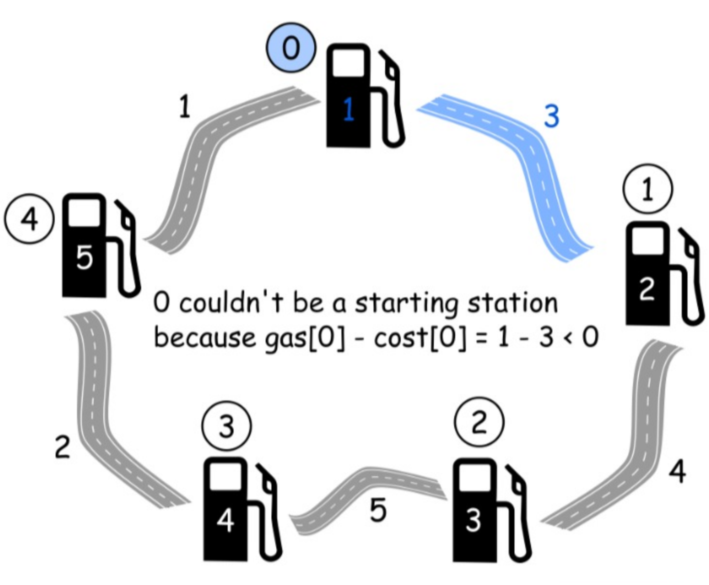
Let's notice two things.

> It's impossible to perform the road trip if `sum(gas) < sum(cost)`. In this situation the answer is `-1`.



One could compute total amount of gas in the tank `total_tank = sum(gas) - sum(cost)` during the round trip, and then return `-1` if `total_tank < 0`.

> It's impossible to start at a station `i` if `gas[i] - cost[i] < 0`, because then there is not enough gas in the tank to travel to `i + 1` station.



The second fact could be generalized. Let's introduce `curr_tank` variable to track the current amount of gas in the tank. If at some station `curr_tank` is less than `0`, that means that one couldn't reach this station.

Next step is to mark this station as a new starting point, and reset `curr_tank` to zero since one starts with no gas in the tank.

Algorithm

Now the algorithm is straightforward :

1. Initiate `total_tank` and `curr_tank` as zero, and choose station `0` as a starting station.
2. Iterate over all stations :
   - Update `total_tank` and `curr_tank` at each step, by adding `gas[i]` and subtracting `cost[i]`.
   - If `curr_tank < 0` at `i + 1` station, make `i + 1` station a new starting point and reset `curr_tank = 0` to start with an empty tank.
3. Return `-1` if `total_tank < 0` and `starting station` otherwise.

Why this works

Let's imagine the situation when `total_tank >= 0` and the above algorithm returns N_s as a starting station.

Algorithm directly ensures that it's possible to go from N_s to the station 0. But what about the last part of the round trip from the station 0 to the station N_s ?

> How one could ensure that it's possible to loop around to N_s ?

Let's use here the proof by contradiction and assume that there is a station $0 < k < N_s$ such that one couldn't reach this station starting from N_s.

The condition `total_tank >= 0` could be written as

$$\sum_{i=0}^{i=N} \alpha_i \ge 0 \qquad (1)$$

where $\alpha_i = \mathrm{gas[i]} - \mathrm{cost[i]}$.

Let's split the sum on the right side by the starting station N_s and unreachable station `k` :

$$\sum_{i=0}^{i=k} \alpha_i + \sum_{i=k+1}^{i=N_s-1} \alpha_i + \sum_{i=N_s}^{i=N} \alpha_i \ge 0 \qquad (2)$$

The second term is negative by the algorithm definition - otherwise the starting station would be before N_s. It could be equal to zero only in the case of $k = N_s - 1$.

$$\sum_{i=k+1}^{i=N_s-1} \alpha_i \le 0 \qquad (3)$$

Equations `(2)` and `(3)` together results in

$$\sum_{i=0}^{i=k} \alpha_i + \sum_{i=N_s}^{i=N} \alpha_i \ge 0 \qquad (4)$$

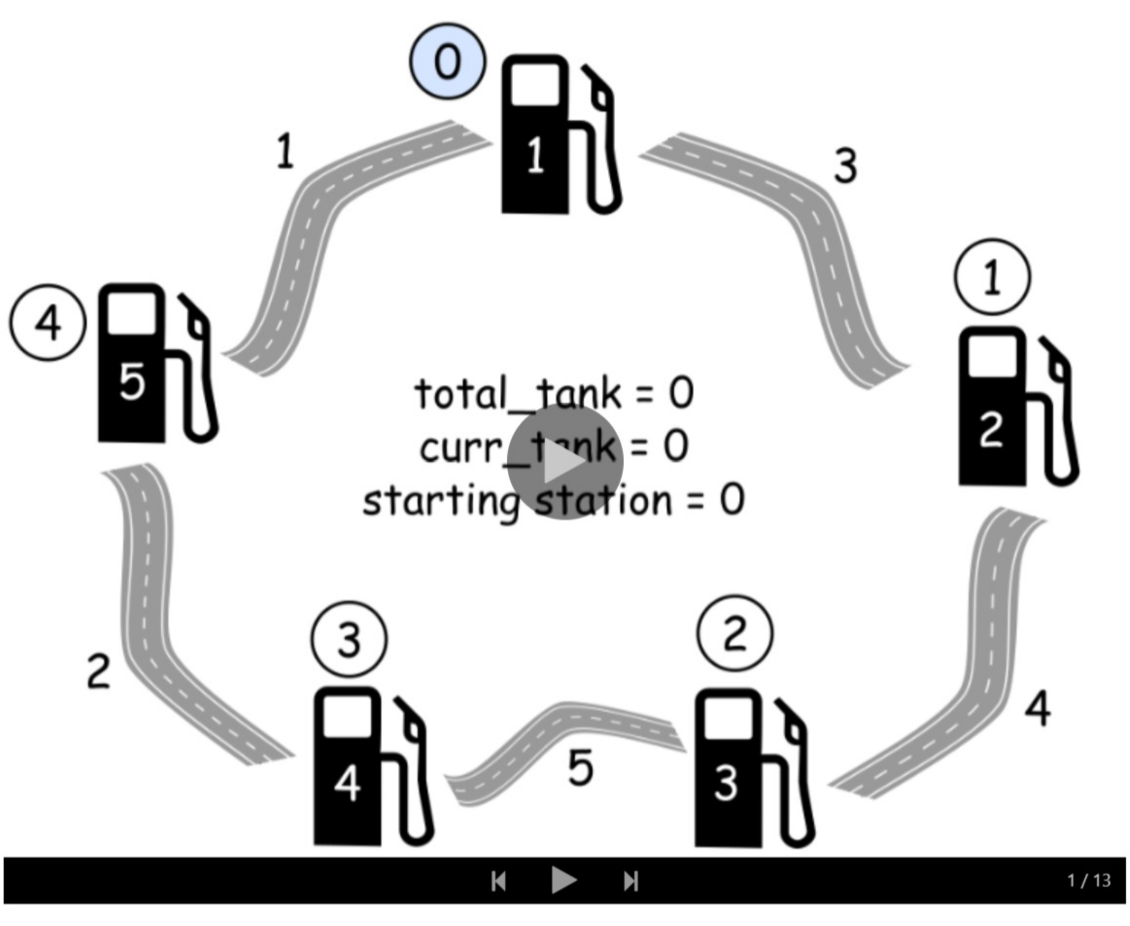
At the same time the station k is supposed to be unreachable from N_s that means

$$\sum_{i=N_s}^{i=N} \alpha_i + \sum_{i=0}^{i=k} \alpha_i < 0 \qquad (5)$$

Eqs. `(4)` and `(5)` together result in a contradiction. Therefore, the initial assumption — that there is a station $0 < k < N_s$ such that one couldn't reach this station starting from N_s — must be false.

Hence, one could do a round trip starting from N_s, that makes N_s to be an answer. The answer is unique according to the problem definition.

Implementation



C++ Java Python Copy

```python
1  class Solution:
2      def canCompleteCircuit(self, gas, cost):
3          """
4          :type gas: List[int]
5          :type cost: List[int]
6          :rtype: int
7          """
8          n = len(gas)
9          
10         total_tank, curr_tank = 0, 0
11         starting_station = 0
12         for i in range(n):
13             total_tank += gas[i] - cost[i]
14             curr_tank += gas[i] - cost[i]
15             # If one couldn't get here,
16             if curr_tank < 0:
17                 # Pick up the next station as the starting one.
18                 starting_station = i + 1
19                 # Start with an empty tank.
20                 curr_tank = 0
21         
22         return starting_station if total_tank >= 0 else -1
```

Complexity Analysis

- Time complexity : $\mathcal{O}(N)$ since there is only one loop over all stations here.
- Space complexity : $\mathcal{O}(1)$ since it's a constant space solution.

Further reading

There are numerous variations of gas problem, here are some examples :

Find the cheapest path between two stations if at most Δ stops are allowed.

Find the cheapest path between two stations if the vehicle has a given tank capacity.

Rate this article:

Previous Next