Gas Station

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, otherwise return -1.

Note:
The solution is guaranteed to be unique.

Solution 1

I have thought for a long time and got two ideas:

- If car starts at A and can not reach B. Any station between A and B can not reach B.(B is the first station that A can not reach.)
- If the total number of gas is bigger than the total number of cost. There must be a solution.
- (Should I prove them?)

Here is my solution based on those ideas:

```
class Solution {
public:
    int canCompleteCircuit(vector<int> &gas, vector<int> &cost) {
        int start(0),total(0),tank(0);
        //if car fails at 'start', record the next station
        for(int i=0;i<gas.size();i++) if((tank=tank+gas[i]-cost[i])<0) {start=i+1
;total+=tank;tank=0;}
        return (total+tank<0)? -1:start;
    }
};
```

written by daxianji007 original link here

Solution 2

I have got one solution to this problem. I am not sure whether somebody has already posted this solution.

```
class Solution {
public:
    int canCompleteCircuit(vector<int> &gas, vector<int> &cost) {

       int start = gas.size()-1;
       int end = 0;
       int sum = gas[start] - cost[start];
       while (start > end) {
          if (sum >= 0) {
             sum += gas[end] - cost[end];
             ++end;
          }
          else {
             --start;
             sum += gas[start] - cost[start];
          }
       }
       return sum >= 0 ? start : -1;
    }
};
```

written by xuewuxiao original link here

Solution 3

```
class Solution {
public:
    int canCompleteCircuit(vector<int> &gas, vector<int> &cost) {
        int i, j, n = gas.size();

        /*
         * If start from i, stop before station x -> no station k from i + 1 to x - 1 can reach x.
         * Bcoz if so, i can reach k and k can reach x, then i reaches x. Contradiction.
         * Thus i can jump directly to x instead of i + 1, bringing complexity from O(n^2) to O(n).
         */
        // start from station i
        for (i = 0; i < n; i += j) {
            int gas_left = 0;
            // forward j stations
            for (j = 1; j <= n; j++) {
                int k = (i + j - 1) % n;
                gas_left += gas[k] - cost[k];
                if (gas_left < 0)
                    break;
            }
            if (j > n)
                return i;
        }

        return -1;
    }
};
```

written by xiaohui7 original link here