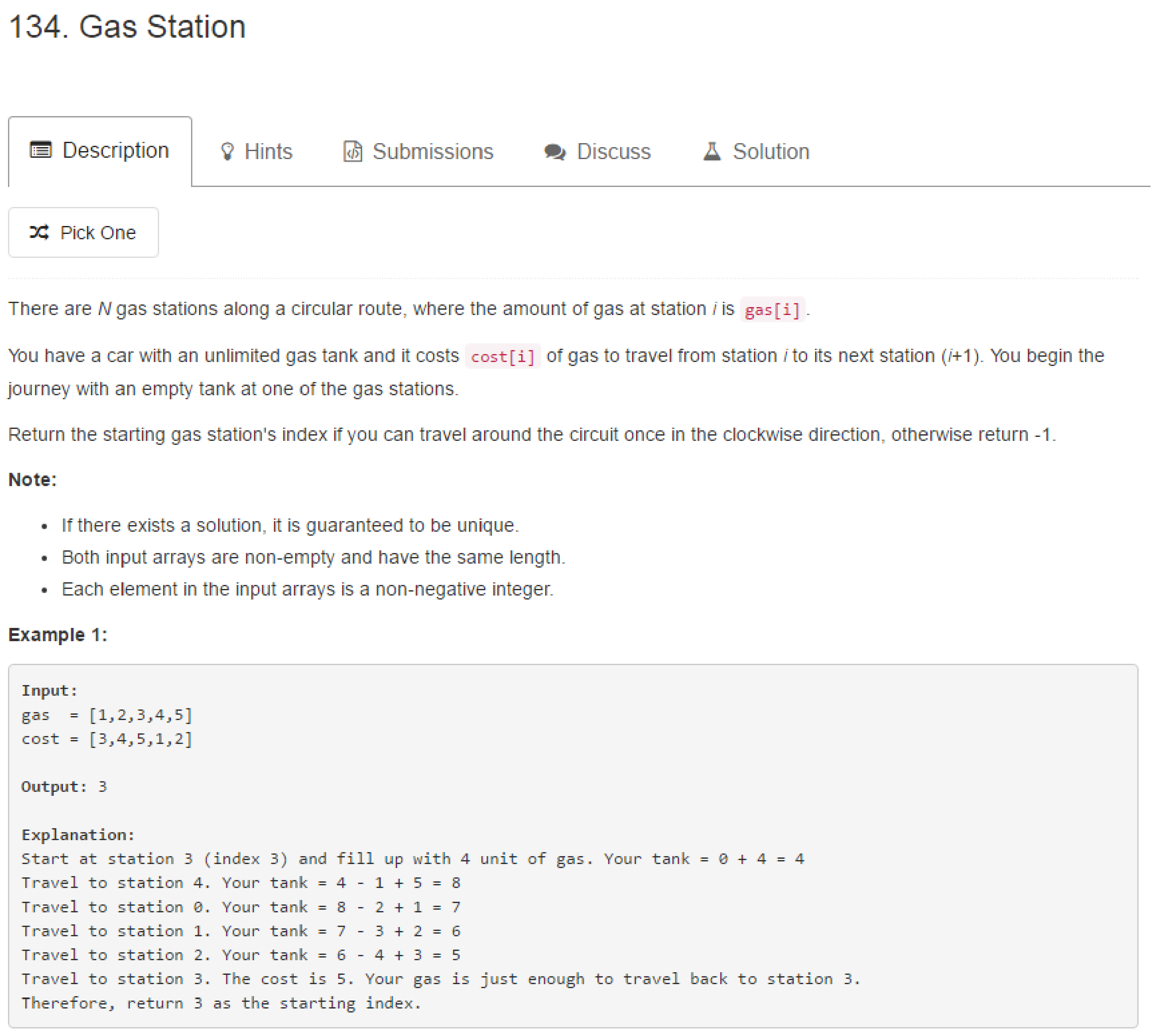
题意：

在一个圆形路径上有N个加油站，在位置 i 上的汽油的数目为gas[i].
你有一个汽车，这个汽车的油箱是无限容量的，它从加油站 i 到 加油站 （i+1）需要耗费的汽油数为cost[i]. 开始这段旅程的时候，你的起始状态是在加油站中的一个，油箱为空的.
若一次性完成整个的圆形路途，返回你的其实加油站的序号，若不能完成整个路途，返回-1.
注意：
解决方案保证是唯一的.

解决方案：
1.暴力解法比较好想，但是超时了。就是从每一个站开始，一直走一圈，累加过程中的净余的油量，看它是不是有出现负的，如果有则失败，从下一站开始重新再走一圈，如果没有出现负的出现，则这个站可以作为起始点，成功。可以看出每次需要扫描一圈，对每一个站都要做一次赛秒，所以时间复杂度为O(n^2).
给出提高的算法：方法主要思想是把这个圈划分为一个个的负序列，以及一个正序列（如果存在的话）。从任意一个站出发，我们可以累加油的净余量，如果出现负的，序列结束，开启一个新的，并且证明旧的这个序列的起点不能作为起点，因为会出现负油量，不能继续前进。而且不仅这个负序列的起点不能作为起点，负序列中的任意一点都不能作为起点。



## 134. 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 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 3.
Therefore, return 3 as the starting index.
```

```java
public class L143 {

      public int canCompleteCircuit(int[] gas, int[] cost) {
          if(gas == null || cost == null || gas.length == 0 || cost.length == 0) {
              return -1;
          }

          int sum = 0; //到达当前加油站的总净容量
          int total = 0; //整个完成一圈的总容量
          int pointor = 0; //定义起点。

          for(int i = 0; i < gas.length; i++) {
              int diff = gas[i] - cost[i];
              sum += diff;
              total += diff;
              if(sum < 0) { //到达该节点油不够，那么这之间的节点都不能作为起点，因为这之间的节点都是正净容量。
                  sum = 0;
                  pointor = i+1;
              }
          }
          return total >= 0 ? pointor : -1;
      }

}
```