Custom Search

Write an Article Login

Find the first circular tour that visits all petrol pumps

Suppose there is a circle. There are n petrol pumps on that circle. You are given two sets of data.

1. The amount of petrol that every petrol pump has.
2. Distance from that petrol pump to the next petrol pump.

Calculate the first point from where a truck will be able to complete the circle (The truck will stop at each petrol pump and it has infinite capacity). Expected time complexity is O(n). Assume for 1 litre petrol, the truck can go 1 unit of distance.

For example, let there be 4 petrol pumps with amount of petrol and distance to next petrol pump value pairs as {4, 6}, {6, 5}, {7, 3} and {4, 5}. The first point from where truck can make a circular tour is 2nd petrol pump. Output should be "start = 1" (index of 2nd petrol pump).

Recommended: Please solve it on "*PRACTICE*" first, before moving on to the solution.

A **Simple Solution** is to consider every petrol pumps as starting point and see if there is a possible tour. If we find a starting point with feasible solution, we return that starting point. The worst case time complexity of this solution is O(n^2).

We can **use a Queue** to store the current tour. We first enqueue first petrol pump to the queue, we keep enqueueing petrol pumps till we either complete the tour, or current amount of petrol becomes negative. If the amount becomes negative, then we keep dequeueing petrol pumps till the current amount becomes positive or queue becomes empty.

Instead of creating a separate queue, we use the given array itself as queue. We maintain two index variables start and end that represent rear and front of queue.

C/C++

```
// C program to find circular tour for a truck
#include <stdio.h>
```

```
   int distance;
};

// The function returns starting point if there is a possible solution,
// otherwise returns -1
int printTour(struct petrolPump arr[], int n)
{
    // Consider first petrol pump as a starting point
    int start = 0;
    int end =  1;

    int curr_petrol = arr[start].petrol - arr[start].distance;

    /* Run a loop while all petrol pumps are not visited.
      And we have reached first petrol pump again with 0 or more petrol */
    while (end != start || curr_petrol < 0)
    {
        // If curremt amount of petrol in truck becomes less than 0, then
        // remove the starting petrol pump from tour
        while (curr_petrol < 0 && start != end)
        {
            // Remove starting petrol pump. Change start
            curr_petrol -= arr[start].petrol - arr[start].distance;
            start = (start + 1)%n;

            // If 0 is being considered as start again, then there is no
            // possible solution
            if (start == 0)
               return -1;
        }

        // Add a petrol pump to current tour
        curr_petrol += arr[end].petrol - arr[end].distance;

        end = (end + 1)%n;
    }

    // Return starting point
    return start;
}

// Driver program to test above functions
int main()
{
    struct petrolPump arr[] = {{6, 4}, {3, 6}, {7, 3}};

    int n = sizeof(arr)/sizeof(arr[0]);
    int start = printTour(arr, n);

    (start == -1)? printf("No solution"): printf("Start = %d", start);

    return 0;
}
```

Run on IDE

Java

```
//Java program to find circular tour for a truck

public class Petrol
{
    // A petrol pump has petrol and distance to next petrol pump
```

```java
        // constructor
        public petrolPump(int petrol, int distance)
        {
            this.petrol = petrol;
            this.distance = distance;
        }
    }

    // The function returns starting point if there is a possible solution,
    // otherwise returns -1
    static int printTour(petrolPump arr[], int n)
    {
        int start = 0;
        int end = 1;
        int curr_petrol = arr[start].petrol - arr[start].distance;

        // If current amount of petrol in truck becomes less than 0, then
        // remove the starting petrol pump from tour
        while(end != start || curr_petrol < 0)
        {

            // If current amount of petrol in truck becomes less than 0, then
            // remove the starting petrol pump from tour
            while(curr_petrol < 0 && start != end)
            {
                // Remove starting petrol pump. Change start
                curr_petrol -= arr[start].petrol - arr[start].distance;
                start = (start + 1) % n;

                // If 0 is being considered as start again, then there is no
                // possible solution
                if(start == 0)
                    return -1;
            }
            // Add a petrol pump to current tour
            curr_petrol += arr[end].petrol - arr[end].distance;

            end = (end + 1)%n;
        }

        // Return starting point
        return start;
    }

    // Driver program to test above functions
    public static void main(String[] args)
    {

        petrolPump[] arr = {new petrolPump(6, 4),
                            new petrolPump(3, 6),
                            new petrolPump(7, 3)};

        int start = printTour(arr, arr.length);

        System.out.println(start == -1 ? "No Solution" : "Start = " + start);

    }

}
//This code is contributed by Sumit Ghosh
```

```
# Python program to find circular tour for a track

# A petrol pump has petrol and distance to next petrol pimp
class PetrolPump:

    # Constructor to create a new node
    def __init__(self,petrol, distance):
        self.petrol = petrol
        self.distance = distance

# The funtion return starting point if there is a possible
# solution otherwise returns -1
def printTour(arr):

    n = len(arr)
    # Consider first petrol pump as starting point
    start = 0
    end = 1

    curr_petrol = arr[start].petrol - arr[start].distance

    # Run a loop whie all petrol pumps are not visited
    # And we have reached first petrol pump again with 0
    # or more petrol
    while(end != start or curr_petrol < 0 ):

        # If current amount of petrol pumps are not visited
        # And we have reached first petrol pump again with
        # 0 or more petrol
        while(curr_petrol < 0 and start != end):

            # Remove starting petrol pump. Change start
            curr_petrol -= arr[start].petrol - arr[start].distance
            start = (start +1)%n

            # If 0 is being considered as start again, then
            # there is no possible solution
            if start == 0:
                return -1

        # Add a petrol pump to current tour
        curr_petrol += arr[end].petrol - arr[end].distance

        end = (end +1) % n

    return start

# Driver program to test above function
arr = [PetrolPump(6,4), PetrolPump(3,6), PetrolPump(7,3)]
start = printTour(arr)

print "No solution" if start == -1 else "start =", start

# This code is contributed by Nikhil Kumar Singh(nickzuck_007)
```

Run on IDE

Output:

```
start = 2
```

queue. The total number of operations is proportional to total number of enqueue operations. Therefore the time complexity is O(n).

Auxiliary Space: O(1)

Recommended: Please solve it on "*PRACTICE*" first, before moving on to the solution.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Queue

Login to Improve this Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Recommended Posts:

Sliding Window Maximum (Maximum of all subarrays of size k)
An Interesting Method to Generate Binary Numbers from 1 to n
Find the largest multiple of 3 | Set 1 (Using Queue)
Minimum time required to rot all oranges
Implement Queue using Stacks
Reversing a queue using recursion
Smallest multiple of a given number made of digits 0 and 9 only
Sorting a Queue without extra space
Implement PriorityQueue through Comparator in Java
Sharing a queue among three threads

(Login to Rate)

3.6 Average Difficulty : **3.6/5.0**
Based on **229** vote(s)

Basic Easy Medium Hard Expert

Add to TODO List

Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments Share this post!

▲