First Circular Tour

Given information about **N** petrol pumps (say **arr[]**) that are present in a circular path. The information consists of the distance of the next petrol pump from the current one (in **arr[i][1]**) and the amount of petrol stored in that petrol pump (in **arr[i][0]**). Consider a truck with infinite capacity that consumes 1 unit of petrol to travel 1 unit distance. The task is to find the index of the first starting point such that the truck can visit all the petrol pumps and come back to that starting point.

Note: Return -1 if no such tour exists.

Examples:

> **Input:** arr[] = {{4, 6}, {6, 5}, {7, 3}, {4, 5}}.
> **Output:** 1
> **Explanation:** If started from 1st index then a circular tour can be covered.
>
> **Input:** arr[] {{6, 4}, {3, 6}, {7, 3}}
> **Output:** 2

Naive Approach:

> As the capacity of the truck is infinite it is feasible to fill the truck with all the amount of petrol available at each petrol pump.

A **Simple Solution** is to consider every petrol pumps as a starting point and see if there is a possible tour. If we find a starting point with a feasible solution, we return that starting point.

Time Complexity: O(N^2)
Auxiliary Space: O(1)

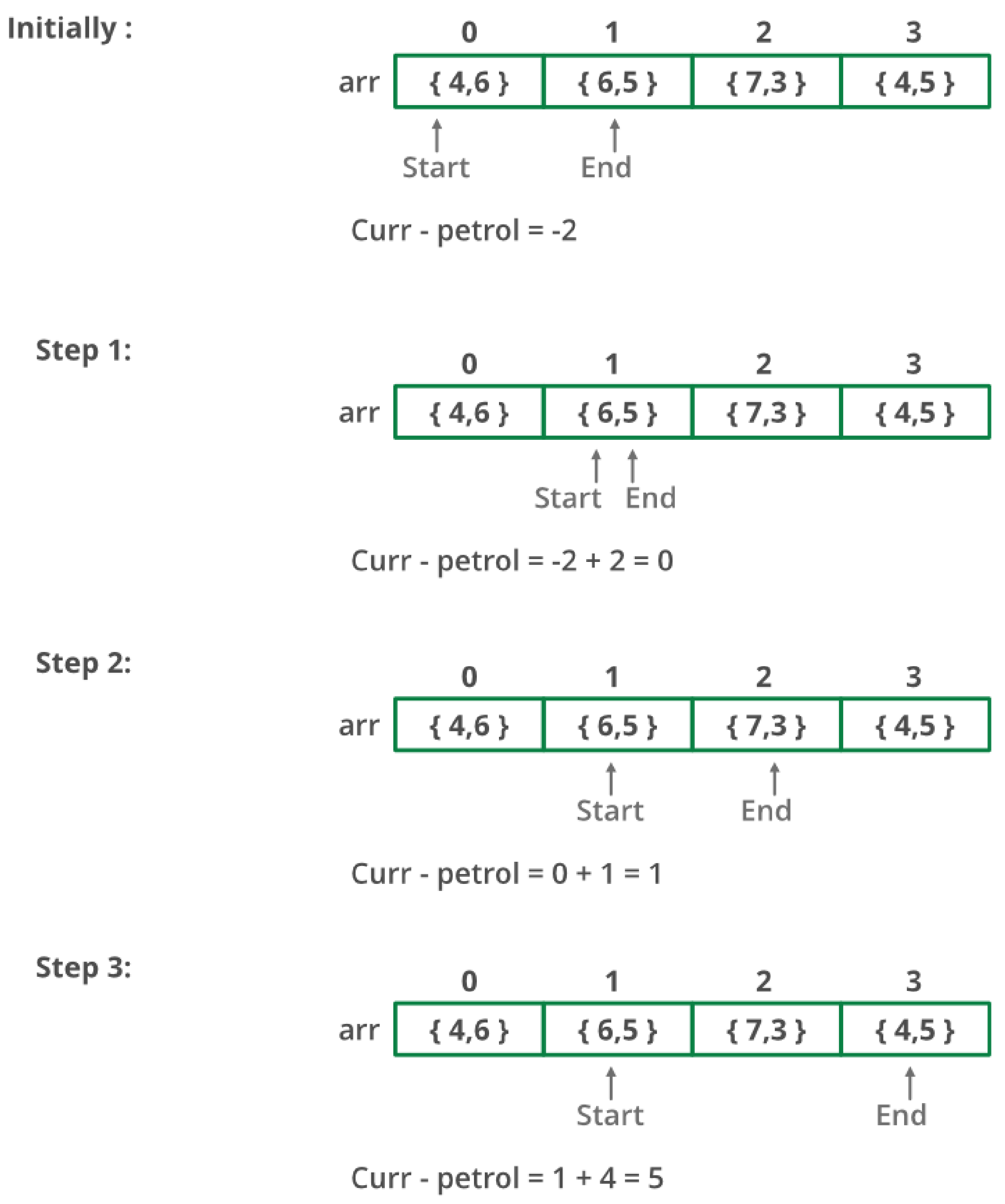
First Circular Tour Using Queue:

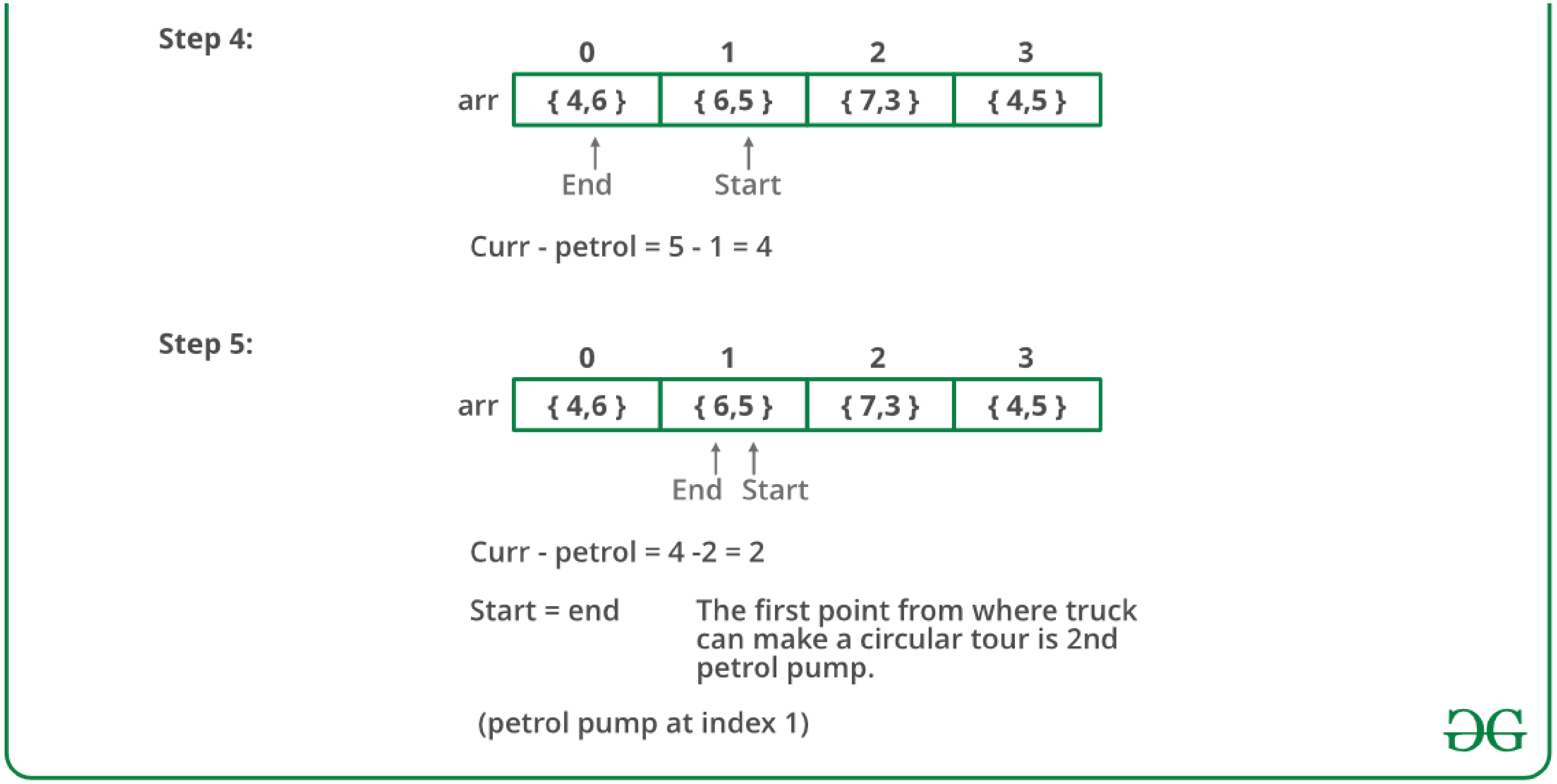
> Instead of checking the whole array for each starting point, we can store the current tour inside a **queue**.
>
> At the moment, the current amount of petrol becomes inefficient (i.e., negative) to reach the next petrol pump, remove the current starting point from the queue and consider the next point as the new starting point.
>
> Instead of building a separate queue, we can use the array itself as a queue with the help of start and end pointers.
>
> **Note:** Each petrol pump will be added in the queue at most twice and will be removed at most once.

Illustration:

Below image is a dry run of the above approach:

Dash
All
Articles
Videos
Problems
Quiz
Contest




Step 4:
0 1 2 3
arr { 4,6 } { 6,5 } { 7,3 } { 4,5 }
End
Start
Curr - petrol = 5 - 1 = 4
Step 5:
0 1 2 3
arr { 4,6 } { 6,5 } { 7,3 } { 4,5 }
End Start
Curr - petrol = 4 -2 = 2
Start = end
The first point from where truck can make a circular tour is 2nd petrol pump.
(petrol pump at index 1)

Follow the below steps to implement the idea:

- Set two pointers, **start = 0** and **end = 1** to use the array as a queue.
  - If the amount of petrol is efficient to reach the next petrol pump then increment the end pointer (circularly).
  - Otherwise, remove the starting point of the current tour, i.e., increment the start pointer.
- If the start pointer reaches **N** then such a tour is not possible. Otherwise, return the starting point of the tour.

Below is the implementation of the above approach:

C++ Java

Dash
All
Articles
Videos
Problems
Quiz
Contest

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

public class Petrol
{
    // A petrol pump has petrol and distance to next petrol pump
    static class petrolPump
    {
        int petrol;
        int distance;

        // 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
```

Dash
All
Articles
Videos
Problems
Quiz
Contest

```
        // 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);

    }

}
```

Output

```
Start = 2
```

Time Complexity: O(N)
Auxiliary Space: O(1)

First Circular Tour by checking only possible valid Starting Positions:

Another efficient solution can be to find out the first petrol pump where the amount of petrol is greater than or equal to the distance to be covered to reach the next petrol pump.

> Mark that petrol pump as **start** and check whether we can finish the journey towards the end point.

> - If in the middle, at any petrol pump, the amount of petrol is less than the distance to be covered to reach the next petrol pump, then we can say we cannot complete the circular tour from **start**.
> - Find the next start petrol pump where the amount of petrol is greater than or equal to the distance to be covered and we mark it as **start**. Continue this process till all points are visited or a starting point is found.

Let us discuss why we need not look at any petrol pump in between the initial petrol pump marked as **start** and the new **start**.

> Let us consider there was a petrol pump at **k**th position between the old start and new start. This petrol pump will break the range into two parts. The case is that
>
> - both the parts can have negative sum,
> - the starting partition can have a negative sum or
> - the later half has a negative sum.
>
> In the first and the last case we will not be able to reach the new start point.
>
> And for the second case though we will be able to reach the new start but will not be able to complete the tour because we will not be able to cover some part in between **0** to the **k**th position. [As we already found that we could not reach to **start** from **0** and also we are not able to reach **k** from **start**. So the tour cannot be completed]

Follow the steps mentioned below to implement the idea:

- Find the first possible petrol pump where the amount of petrol is greater than the distance to the next petrol pump.
- Traverse from **i = start to N**:

    - If the amount of petrol becomes inefficient (i.e., negative) we need to update the new start point.
        - Traverse from **i+1 to N** and find the point where petrol is greater than the distance to the next petrol pump.
    - Start from the new start point and continue the above procedure.
- Start from **0** to the found **start** point. If the sum of the petrol is non-negative then the start point is feasible otherwise not.

Below is the implementation of the above approach:

C++ Java

```
 // Java program to find circular tour for a truck
public class Circular
{

    // A petrol pump has petrol and distance to next petrol
    // pump
    static class petrolPump {
        int petrol;
        int distance;

        public petrolPump(int petrol, int distance)
        {
            this.petrol = petrol;
            this.distance = distance;
        }
    }

    // The function returns starting point if there is a
```

Dash
All
Articles
Videos
Problems
Quiz
Contest

```
// possible solution, otherwise returns -1
static int printTour(petrolPump arr[], int n)
{
    int start = 0;

    for (int i = 0; i < n; i++) {
        // Identify the first petrol pump from where we
        // might get a full circular tour
        if (arr[i].petrol >= arr[i].distance) {
            start = i;
            break;
        }
    }

    // To store the excess petrol
    int curr_petrol = 0;
    int i;
    for (i = start; i < n;) {

        curr_petrol
            += (arr[i].petrol - arr[i].distance);

        // If at any point remaining petrol is less than
        // 0, it means that we cannot start our journey
        // from current start
        if (curr_petrol < 0) {
            // We move to the next petrol pump
            i++;
```

Dash
All
Articles
Videos
Problems
Quiz
Contest

```
                // We try to identify the next petrol pump
                // from where we might get a full circular
                // tour
                for (; i < n; i++) {
                    if (arr[i].petrol >= arr[i].distance) {

                        start = i;

                        // Reset rem_petrol
                        curr_petrol = 0;

                        break;
                    }
                }
            }

            else {
                // Move to the next petrolpump if
                // curr_petrol is
                // >= 0
                i++;
            }
        }

        // If remaining petrol is less than 0 while we reach
        // the first petrol pump, it means no circular tour
        // is possible
        if (curr_petrol < 0) {
            return -1;
        }

        for (int j = 0; j < start; j++) {

            curr_petrol
                += (arr[j].petrol - arr[j].distance);

            // If remaining petrol is less than 0 at any
            // point before we reach initial start, it means
            // no circular tour is possible
            if (curr_petrol < 0) {
                return -1;
            }
        }

        // If we have successfully reached intial_start, it
        // means can get a circular tour from final_start,
        // hence return it
        return start;
    }

    // Driver code
    public static void main(String[] args)
    {

        petrolPump arr[]
            = { new petrolPump(6, 4), new petrolPump(3, 6),
                new petrolPump(7, 3) };
        int n = arr.length;
        int start = printTour(arr, n);

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

Output

```
Start = 2
```

Time Complexity: O(N)
Auxiliary Space: O(1)

First Circular Tour by using Single Loop:

The idea is similar to the above approach.

> Here we will use another variable to substitute the extra loop from start till the latest found start point. The variable will store the sum of utilized petrol from **0** till the latest start petrol pump.

Below is the implementation of the above approach:

C++ Java

Dash
All
Articles
Videos
Problems
Quiz
Contest

```
// Java program to find circular tour for a truck
import java.util.*;
class GFG {

  // A petrol pump has petrol and distance to next petrol pump
  static class petrolPump {

    int petrol;
    int distance;

    petrolPump(int a, int b) {
      this.petrol = a;
      this.distance = b;
    }
  };

  // The function returns starting point if there is a
  // possible solution, otherwise returns -1
  static int printTour(petrolPump p[], int n)
  {

    // deficit is used to store the value of the capacity as
    // soon as the value of capacity becomes negative so as
    // not to traverse the array twice in order to get the
    // solution
```

Get 90% Refund!
Courses Tutorials Jobs Practice Contests

P

```
      for (int i = 0; i < n; i++) {
        capacity += p[i].petrol - p[i].distance;
        if (capacity < 0) {
          // If this particular step is not done then the
          // between steps would be redundant
          start = i + 1;
          deficit += capacity;
          capacity = 0;
        }
      }
      return (capacity + deficit >= 0) ? start : -1;
    }

    // Driver code
    public static void main(String[] args) {
      petrolPump arr[] = { new petrolPump(6, 4),
                           new petrolPump(3, 6),
                           new petrolPump(7, 3) };

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

      if (start == -1)
        System.out.print("No solution");
      else
        System.out.print("Start = " + start);

    }
  }
```

Dash
All
Articles
Videos
Problems
Quiz
Contest

Output

```
Start = 2
```

Time Complexity: O(N)
Auxiliary Space: O(1)

Mark as Read

Report An Issue

If you are facing any issue on this page. Please let us know.