

Project report

Algorithm 2

Group 1

1. Name

- 1) Chanran Kim (shining04@csu.fullerton.edu)
- 2) Vinh Nguyen (vinhgod123@csu.fullerton.edu)
- 3) Brian Alvarez (briandalvarez@csu.fullerton.edu)
- 4) Christopher Contreras (CContreras71@csu.fullerton.edu)

2. Pseudocode

Problem: Find a starting city on a circular route such that your car can return to the starting city with zero or more gallons of gas.

Inputs:

- An array of integers *city_distances*
- An array of integers *fuel*
- An integer *mpg*

Outputs: An int value that is the index of the optimal starting city for the car to complete its route

Constraints and Assumptions:

- The route is circular, meaning that the last city connects to the first city
- There will always be exactly one valid starting city
- The *city_distances* array and *fuel* array must be the same length
- The *city_distances* array or *fuel* array cannot be empty
- *mpg* cannot be less than or equal to 0

Pseudocode:

```
function find_starting_city(city_distances, fuel, mpg)
    starting_city_index = 0
    current_fuel = 0
    total_fuel = 0
    total_distance = 0
```

```
    if city_distances or fuel is empty
```

```

        raise an error

    if length of city_distances != length of fuel
        raise an error

    if mpg <= 0
        raise an error

    for i from 0 to length(city_distances)
        total_fuel = total_fuel + (fuel[i] * mpg)
        total_distance = total_distance + city_distances[i]

    if total_fuel < total_distance:
        raise an error

    for i from 0 to length(city_distances)
        current_fuel = current_fuel + (fuel[i] * mpg)
        current_fuel = current_fuel - city_distances[i]

    if current_fuel < 0
        set starting_city_index to the next city
        reset current_fuel to 0

return starting_city_index

```

3. Proving efficiency of the pseudocode

Time complexity

1. $O(1)$ - Initialization: variables `starting_city_index` and `current_fuel` are initialized to 0, which is in constant time, resulting in $O(1)$
2. $O(1)$ - Error Handling: Consists of simple conditional checks such as verifying array lengths and checking the value of a variable, all of which take constant time, resulting in $O(1)$
3. $O(n)$ - Error Handling Loop: `city_distances` array has a length of n . Since the loop runs n times and each operation within the loops takes constant time, it results in $O(n)$
4. $O(n)$ - Main loop: `city_distances` array has a length of n . Since the loop runs n times and each operation within the loops takes constant time, it results in $O(n)$
5. $O(1) + O(1) + O(n) + O(n) = O(n)$

Space complexity

1. $O(1)$ - has a fixed amount of space for a few variables, leading to no additional data structures used in that scale, leading it to be a constant making it $O(1)$

Time Complexity: $O(n)$

Space Complexity: $O(1)$