

CPSC 535 Advanced Algorithms

Project 1: Electric Car Traveler

Instructor: Prof. Doina Bein

Submission date: 14th October 2022

Janhvi Guha
[885186973]
jguha@csu.fullerton.edu

Divyansh Mohan Rao
[885191403]
divyanshrao@csu.fullerton.edu

Summary

This project has been implemented using python 3.8.2. We installed python from <https://www.python.org/downloads/>. We used visual studio code as our IDE. The code is expecting 3 inputs, capacity integer, comma-separated name of cities and distance between each city.

Pseudocode

1. Take cities and distance inputs from the user and put in the array
2. Take c capacity input from user
3. if $c < \text{distance}[0]$:
 return 'Fuel capacity low, cant complete journey'
4. available capacity = $c - \text{distance}[0]$
 Initialize a pointer $j=1$ which points to the current cities
5. While ($j < \text{len}(\text{cities}) - 1$ && $\text{cities} \neq 0$)
 if $\text{distance}[j] * 2 \leq \text{available capacity}$:
 available capacity = available capacity - $\text{distance}[j]$
 $j++$
 else
 append traversed cities to the output array
 make available capacity = c
 if $j \neq \text{len}(\text{distance}) - 1$ and $\text{availCap} < \text{distance}[j+1] * 2$:
 return "Cant travel to next city after city:" + $\text{cities}[j]$
6. Append last stop in list
7. Print output array

How to execute code

To execute code, we need to navigate to the code directory, which in this case is *Users/JanhviGuha/Desktop*, and then execute the code in the terminal using the command *python3 electriccar.py* and then provide all the required inputs in order to get output.

Code

```
#cities[str]: entered by user comma separated
#distance[int]:integer distance from city 0 to city 1....city N
#c: capacity of the electric car to travel in full tank
def electricCar(cities: list[str],distance: list[int],c: int) -> list[str]:
    #Initializing and adding the starting city in the output Traversed array
    traversed = [cities[0]]
    c = int(c)
    #if the distance from city 0 to city 1 is less than capacity
    if len(distance) > 0 and c < distance[0]:
        return 'Fuel capacity low, cant complete journey'
    #Available capacity after subtracting the distance of city 0
    availCap = c - distance[0]
    j = 1
    #while we have not traversed all the cities Do
    while j < len(cities)-1:
        # calculate if we can travel to next city and come back if pump is broken
        if distance[j] * 2 <= availCap:
            #decrease capacity and increase pointer to next city
            availCap -= distance[j]
            j+=1
        else:
            #if we dont have capacity to go to next city and come back then refill on current city
            traversed.append(cities[j])
            #restore capacity
            availCap = c
            #if after refilling also the capacity is not enough to reach next city then return error msg
            if j != len(distance)-1 and availCap < distance[j+1] * 2:
                return "Cant travel to next city after city:" +cities[j]
    #add the last city to traversed list
    traversed.append(cities[-1])
    #return result
    return traversed

cap = input('please enter capacity of car: ')
cityList = input("please Enter cities (Comma Separated): ")
cityList = cityList.split(",")
distanceList = []
for i in range(1,len(cityList)):
```

```
distanceList.append(int(input("please enter distance from city " +cityList[i-1]+ " to city "+
cityList[i] + ": "))))
```

```
print("The miniumum number of stops to reach the destination is : "
,electricCar(cityList,distanceList,cap))
```

Time Complexity

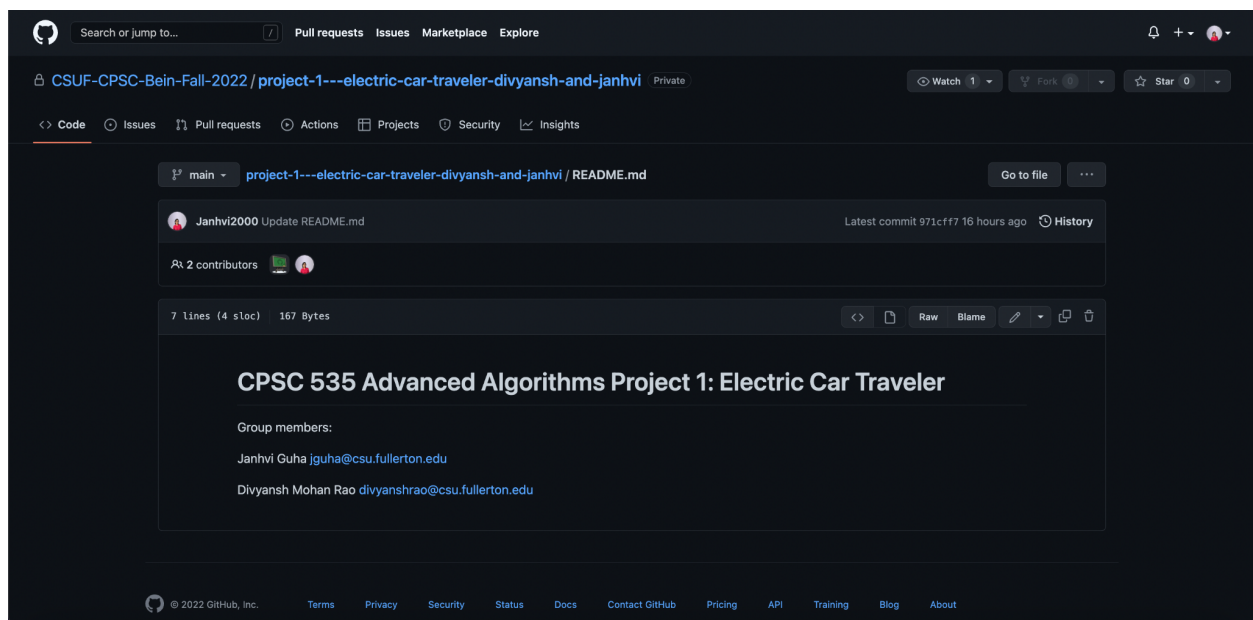
The time complexity of this algorithm is asymptotically $O(n)$ because the while loop will run for $n-2$ unit of time in worst case.

Space complexity

The space complexity of this algorithm will be equal to size N in worst case when all cities needs to traverse.

Corner Cases

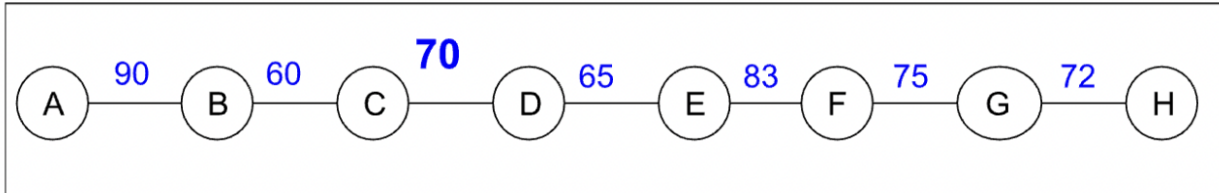
- 1) If the capacity of the car is less than distance between the first city and second city, then return appropriate error msg
- 2) If the capacity of the car even after refill is not able to reach the next city and back, then show appropriate error msg



Test Cases

Case 1:

Input:

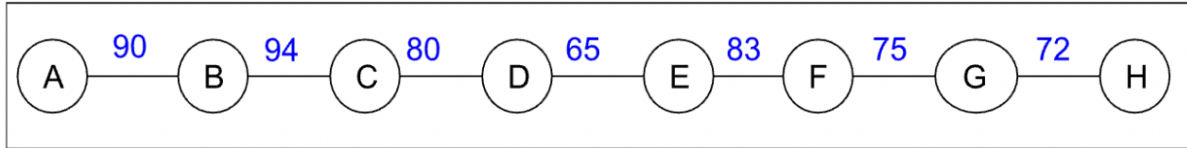


Output:

```
electriccar.py x
Users > JanhviGuha > Desktop > electriccar.py > ...
1 #cities[str]: entered by user comma separated
2 #distance[int]: integer distance from city 0 to city 1...city N
3 #c: capacity of the electric car to travel in full tank
4 def electricCar(cities: list[str], distance: list[int], c: int) -> list[str]:
5     #Initializing and adding the starting city in the output Traversed array
6     traversed = [cities[0]]
7     c = int(c)
8     #If the distance from city 0 to city 1 is less than capacity
9     if len(distance) > 0 and c < distance[0]:
10         return 'Fuel capacity low, cant complete journey'
11     #Available capacity after subtracting the distance of city 0
12     availCap = c - distance[0]
13     j = 1
14     #While we have not traversed all the cities Do
15     while j < len(cities)-1:
16         # calculate if we can travel to next city and come back if pump is broken
17         if distance[j] * 2 <= availCap:
18             #decrease capacity and increase pointer to next city
19             availCap -= distance[j]
20             j+=1
21         else:
22             #If we dont have capacity to go to next city and come back then refill on
23             traversed.append(cities[j])
24             #restore capacity
25             availCap = c
26             #If after refilling also the capacity is not enough to reach next city th
27             if j != len(distance)-1 and availCap < distance[j+1] * 2:
28                 return "Cant travel to next city after city:" +cities[j]
29             #add the last city to traversed list
30             traversed.append(cities[-1])
31             #return result
32             return traversed
33
34 cap = input('please enter capacity of car: ')
35 cityList = input("please Enter cities (Comma Separated): ")
36 cityList = cityList.split(",")
37 distanceList = []
38 for i in range(1,len(cityList)):
39     distanceList.append(int(input("please enter distance from city " +cityList[i-1]+
40
zsh x
(base) JanhviGuha@Janhvis-Air ~ % cd desktop
(base) JanhviGuha@Janhvis-Air desktop % python3 electriccar.py
please enter capacity of car: 300
please Enter cities (Comma Separated): A,B,C,D,E,F,G,H
please enter distance from city A to city B: 90
please enter distance from city B to city C: 60
please enter distance from city C to city D: 70
please enter distance from city D to city E: 65
please enter distance from city E to city F: 83
please enter distance from city F to city G: 75
please enter distance from city G to city H: 72
The miniumum number of stops to reach the destination is : ['A', 'D', 'G', 'H']
(base) JanhviGuha@Janhvis-Air desktop %
```

Case 2:

Input:



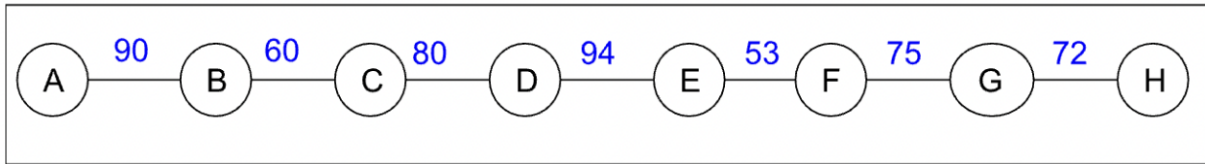
Output:

```
electriccar.py
Users > JanhviGuha > Desktop > electriccar.py > ...
1 #cities[str]: entered by user comma separated
2 #distance[int]: integer distance from city 0 to city 1...city N
3 #c: capacity of the electric car to travel in full tank
4 def electricCar(cities: list[str], distance: list[int], c: int) -> list[str]:
5     #Initializing and adding the starting city in the output Traversed array
6     traversed = [cities[0]]
7     c = int(c)
8     #if the distance from city 0 to city 1 is less than capacity
9     if len(distance) > 0 and c < distance[0]:
10         return 'Fuel capacity low, cant complete journey'
11     #Available capacity after subtracting the distance of city 0
12     availCap = c - distance[0]
13     j = 1
14     #while we have not traversed all the cities Do
15     while j < len(cities)-1:
16         # calculate if we can travel to next city and come back if pump is broken
17         if distance[j] * 2 <= availCap:
18             #decrease capacity and increase pointer to next city
19             availCap -= distance[j]
20             j+=1
21         else:
22             #if we dont have capacity to go to next city and come back then refill on
23             traversed.append(cities[j])
24             #restore capacity
25             availCap = c
26             #if after refilling also the capacity is not enough to reach next city th
27             if j != len(distance)-1 and availCap < distance[j+1] * 2:
28                 return "Cant travel to next city after city:" +cities[j]
29             #add the last city to traversed list
30             traversed.append(cities[j])
31             #return result
32             return traversed
33
34 cap = input('please enter capacity of car: ')
35 cityList = input("please Enter cities (Comma Separated): ")
36 cityList = cityList.split(",")
37 distanceList = []
38 for i in range(1,len(cityList)):
39     distanceList.append(int(input("please enter distance from city " +cityList[i-1]+
40
41 print("The miniumum number of stops to reach the destination is : ",electricCar(cityList, distanceList, cap))

zsh
(base) JanhviGuha@Janhvis-Air desktop % python3 electriccar.py
please enter capacity of car: 300
please Enter cities (Comma Separated): A,B,C,D,E,F,G,H
please enter distance from city A to city B: 90
please enter distance from city B to city C: 94
please enter distance from city C to city D: 80
please enter distance from city D to city E: 65
please enter distance from city E to city F: 83
please enter distance from city F to city G: 75
please enter distance from city G to city H: 72
The miniumum number of stops to reach the destination is : ['A', 'C', 'E', 'G', 'H']
(base) JanhviGuha@Janhvis-Air desktop %
```

Case 3:

Input:



Output:

```
electriccar.py x
Users > JanhviGuha > Desktop > electriccar.py > ...
1 #cities[str]: entered by user comma separated
2 #distance[int]: integer distance from city 0 to city 1...city N
3 #c: capacity of the electric car to travel in full tank
4 def electricCar(cities: list[str], distance: list[int], c: int) -> list[str]:
5     #Initializing and adding the starting city in the output Traversed array
6     traversed = [cities[0]]
7     c = int(c)
8     #If the distance from city 0 to city 1 is less than capacity
9     if len(distance) > 0 and c < distance[0]:
10         return 'Fuel capacity low, cant complete journey'
11     #Available capacity after subtracting the distance of city 0
12     availCap = c - distance[0]
13     j = 1
14     #While we have not traversed all the cities Do
15     while j < len(cities)-1:
16         # calculate if we can travel to next city and come back if pump is broken
17         if distance[j] * 2 <= availCap:
18             #decrease capacity and increase pointer to next city
19             availCap -= distance[j]
20             j+=1
21         else:
22             #if we dont have capacity to go to next city and come back then refill on
23             traversed.append(cities[j])
24             #restore capacity
25             availCap = c
26             #if after refilling also the capacity is not enough to reach next city th
27             if j != len(distance)-1 and availCap < distance[j+1] * 2:
28                 return "Cant travel to next city after city:" +cities[j]
29             #add the last city to traversed list
30             traversed.append(cities[j-1])
31             #return result
32             return traversed
33
34 cap = input('please enter capacity of car: ')
35 cityList = input("please Enter cities (Comma Separated): ")
36 cityList = cityList.split(",")
37 distanceList = []
38 for i in range(1,len(cityList)):
39     distanceList.append(int(input("please enter distance from city " +cityList[i-1]+
40
41 print("The minimum number of stops to reach the destination is : ",electricCar(cityList, distanceList, cap))

zsh x
(base) JanhviGuha@Janhvis-Air desktop % python3 electriccar.py
please enter capacity of car: 300
please Enter cities (Comma Separated): A,B,C,D,E,F,G,H
please enter distance from city A to city B: 90
please enter distance from city B to city C: 60
please enter distance from city C to city D: 80
please enter distance from city D to city E: 94
please enter distance from city E to city F: 53
please enter distance from city F to city G: 75
please enter distance from city G to city H: 72
The minimum number of stops to reach the destination is : ['A', 'C', 'F', 'H']
(base) JanhviGuha@Janhvis-Air desktop %
```