EE 538 PROJECT - TROJAN MAP

PRESENTED BY:

1. Vaishnavi Channakeshava (USC ID : 9673718359)
2. Sanjana Vasudeva (USC ID : 7071819723)

1. Autocomplete

Input : Input a partial name

Output : Return all possible locations with partial name as the prefix.

Time Complexity : O(n)

```
*****************************************
* 1. Autocomplete
*****************************************

Please input a partial location:Chi
*************************Results**********
Chinese Street Food
Chick-fil-A
Chipotle
*****************************************
Time taken by function: 29 ms
```

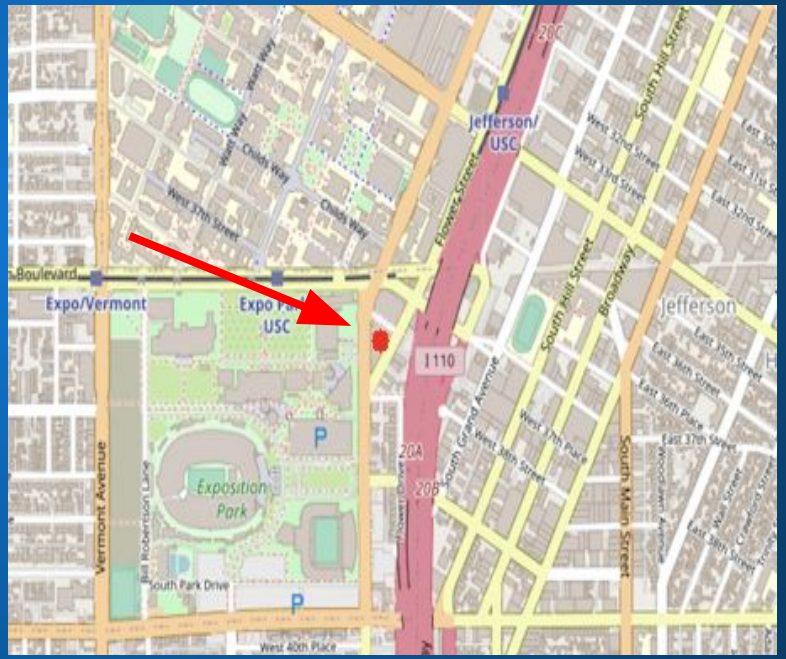
2. Get Position

Input : Location name

Output : Return Latitude and Longitude

Time Complexity : O(n)

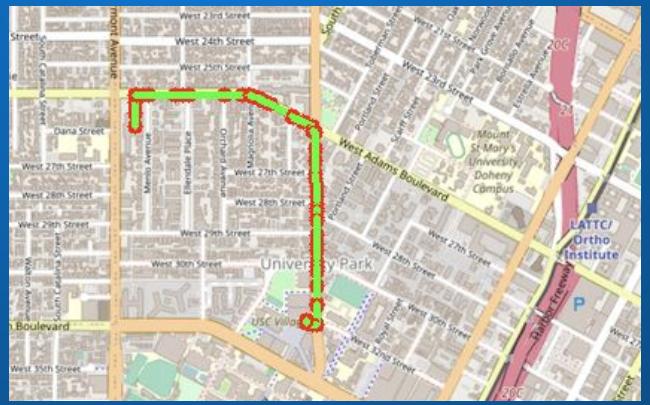
```
**********************************************
* 2. Find the location
**********************************************

Please input a location:Target
**************************Results***********
Latitude: 34.0257 Longitude: -118.284
**********************************************
Time taken by function: 39 ms
```

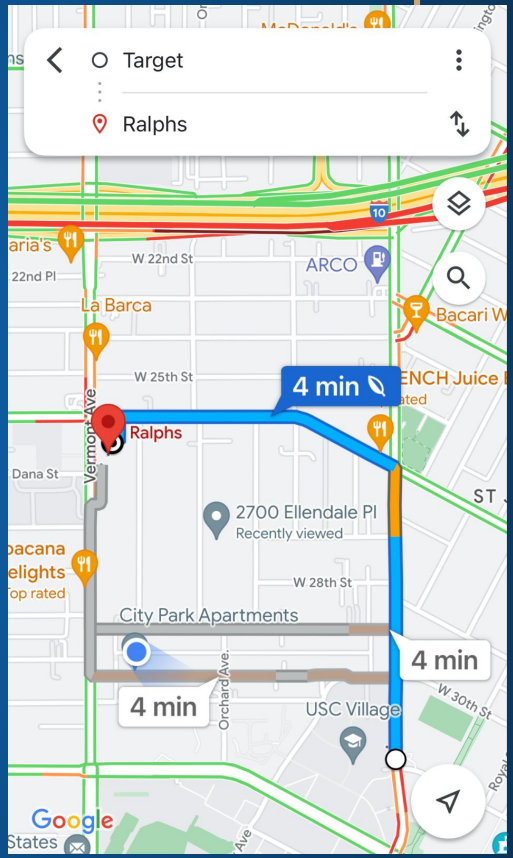


3. Calculate shortest distance
3.1 Dijkstra Algorithm

- ➔ The Algorithm returns the shortest path between the two locations
- ➔ The algorithm is implemented uses priority queue with min heap.
- ➔ Time Complexity : O((V+E) log(V)) - Vertices and E - Edges on the map.
- ➔ Example - Source : Ralphs
  Destination : Target

```
************************************************************
* 3. CalculateShortestPath
************************************************************

Please input the start location:Ralphs
Please input the destination:Target
************************Dijkstra****************************
************************Results*****************************
"2578244375","4380040154","4380040158","4380040167","6805802087","8410938469","6813416131","7645318201","6813416130","6813416129","123318563","452688940","6816193777","123408705","6816193774","452688933","452688931","123230412","6816193770","6787470576","4015442011","6816193692","6816193693","6816193694","4015377691","544693739","6816193696","6804883323","6807937309","6807937306","6816193698","4015377690","4015377689","122814447","6813416159","6813405266","4015372488","4015372487","6813405229","122719216","6813405232","4015372486","7071032399","4015372485","6813379479","6813379584","6814769289","5237417650",
The distance of the path is:0.927969 miles
************************************************************
Time taken by function: 107 ms
```

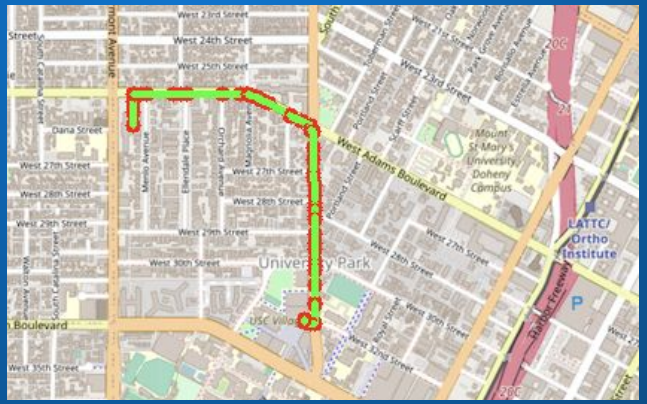


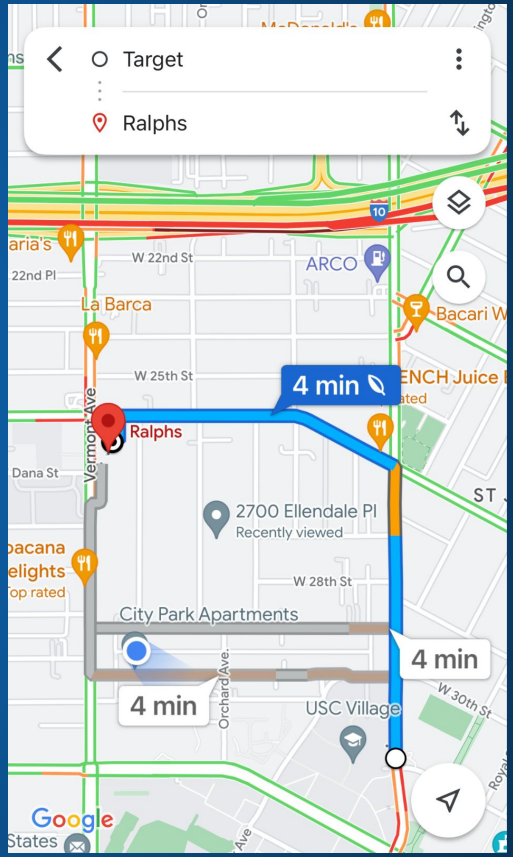


3.2 Bellman-Ford Algorithm

- ➔ The Algorithm returns the shortest path between the two locations .
- ➔ The algorithm uses unordered maps to save the shortest distance .
- ➔ Time Complexity : O(V*E) - Vertices and E - Edges on the map.
- ➔ Example - Source : Target
  Destination : Ralphs

```
**************************Bellman_Ford**************************
**************************Results*******************************
"2578244375","4380040154","4380040158","4380040167","6805802087","8410938469","6813416131","7645318201","6813416130","6813416129","123318563","452688940","6816193777","123408705","6816193774","452688933","452688931","123230412","6816193770","6787470576","4015442011","6816193692","6816193693","6816193694","4015377691","544693739","6816193696","6804883323","607937309","6807937306","6816193698","4015377690","4015377689","122814447","6813416159","6813405266","4015372488","4015372487","6813405229","122719216","6813405232","4015372486","7071032399","4015372485","6813379479","6813379584","6814769289","5237417650",
The distance of the path is:0.927969 miles
*****************************************************************
Time taken by function: 9262 ms
```





Comparison of the two algorithms

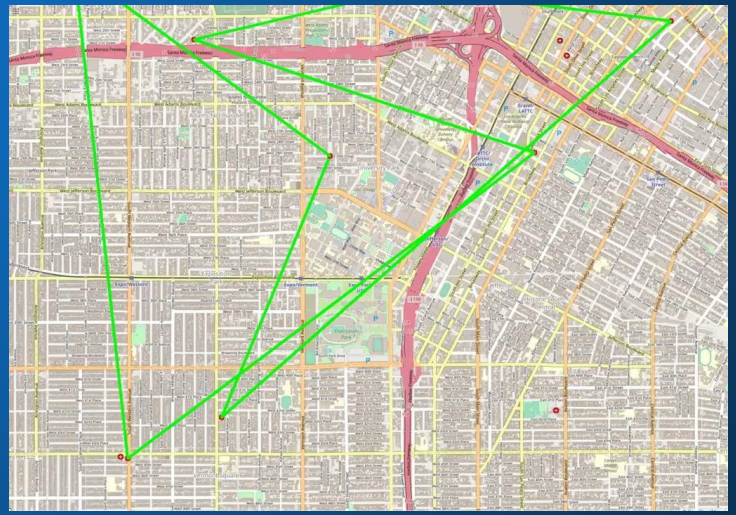
Examples	Time Taken by Dijkstra Algorithm	Time Taken by Bellman Ford Algorithm
Source : Ralphs Destination : Target	104 ms	9194 ms
Source : Chipotle Destination : University Park	98 ms	8613 ms
Source : Dulce Destination : Bank of America	81 ms	8587 ms
Average time taken	94.33 ms	8798 ms

Runtime comparison of the two algorithms

2. Travelling Salesman Problem
2.1 TSP using Brute-Force

- ➔ Given a list of locations, the algorithm returns the shortest path which visit all the places and back to the start point.
- ➔ The algorithm takes the permutation of all possible paths possible and returns the shortest path.
- ➔ **Time Complexity :** O(n!)

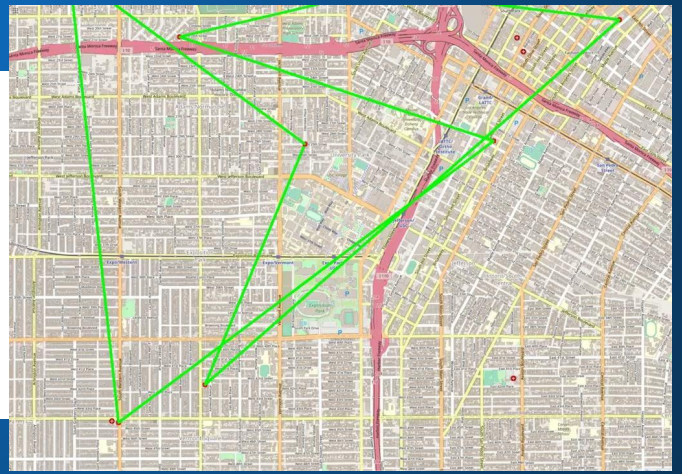
```
Please input the number of the places:8
"6813405268","7162324675","5309161269","1770628326","123454874","7864621790","6792460853","6808042778",
Calculating ...
************************Results*****************************
TravellingTrojan_Brute_force
"6813405268","7864621790","5309161269","1770628326","123454874","6808042778","6792460853","7162324675","6813405268",
The distance of the path is:10.8911 miles
************************************************************
You could find your animation at src/lib/output0.avi.
Time taken by function: 16 ms
```



2.2 TSP using Backtracking

- ➔ Given a list of locations, the algorithm returns the shortest path which visit all the places and back to the start point.
- ➔ The algorithm takes the permutation of all possible paths possible and returns the shortest path.
- ➔ **Time Complexity :** O(n!)

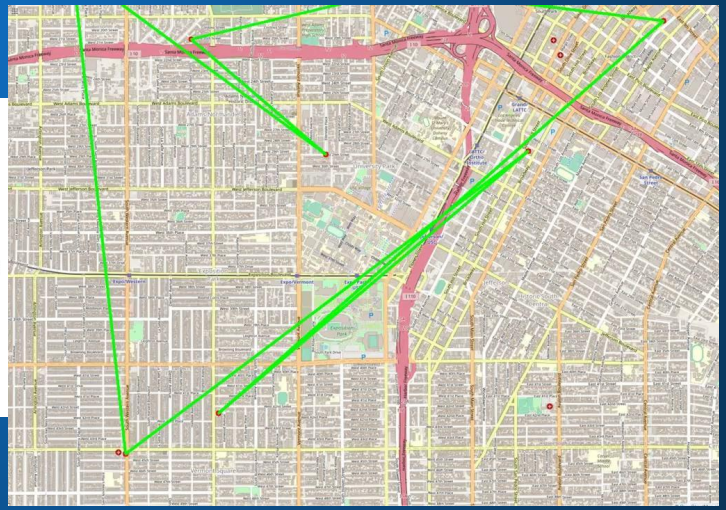
```
Calculating ...
**********Results**********
TravellingTrojan_Backtracking
"6813405268","7864621790","5309161269","1770628326","123454874","6808042778","6792460853","7162324675","6813405268",
The distance of the path is:10.8911 miles
************************************************************
You could find your animation at src/lib/output0_backtracking.avi.
Time taken by function: 18 ms
```



2.3 TSP using 2-Opt Heuristic

- ➔ Given a list of locations, the algorithm returns the shortest path which visit all the places and back to the start point.
- ➔ The algorithm takes the permutation of all possible paths possible and returns the shortest path.
- ➔ **Time Complexity :** O(n*k) where k is the number of improvements possible

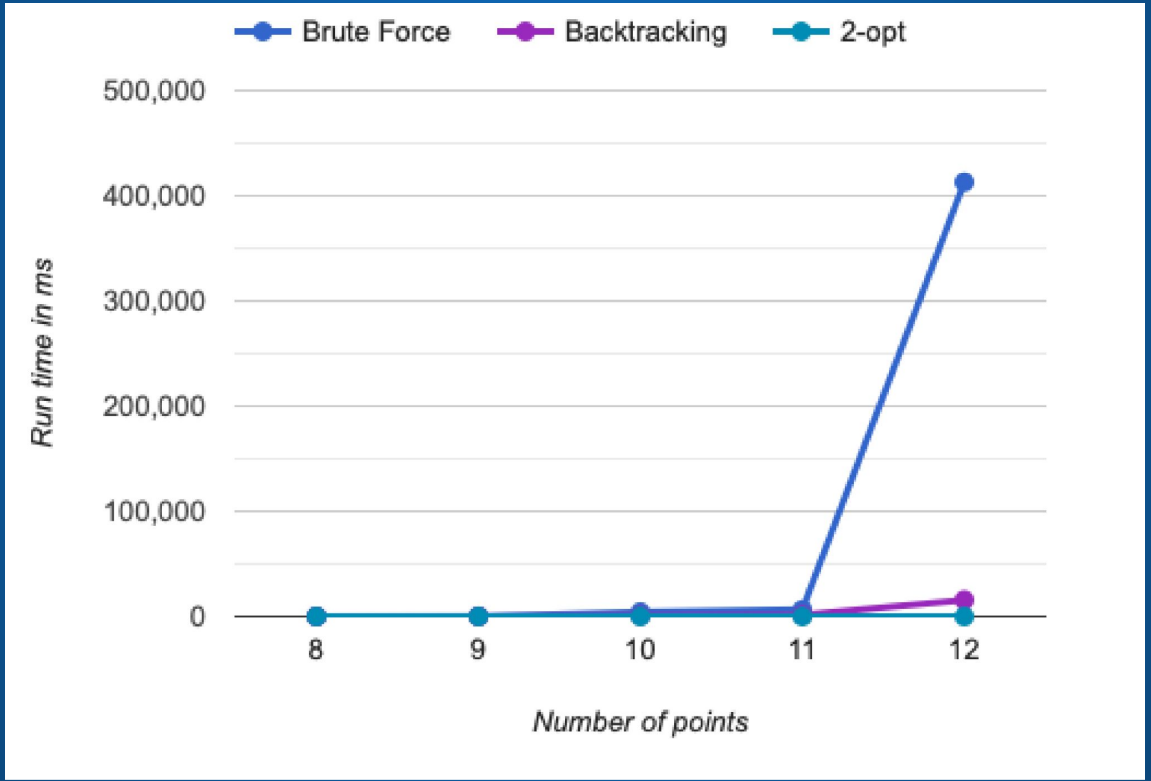
```
Calculating ...
**********Results**********
TravellingTrojan_2opt
"6813405268","7864621790","5309161269","1770628326","123454874","6808042778","6792460853","7162324675","6813405268",
The distance of the path is:10.8911 miles
**********
You could find your animation at src/lib/output0_2opt.avi.
Time taken by function: 1 ms
```



Comparison of the three algorithms

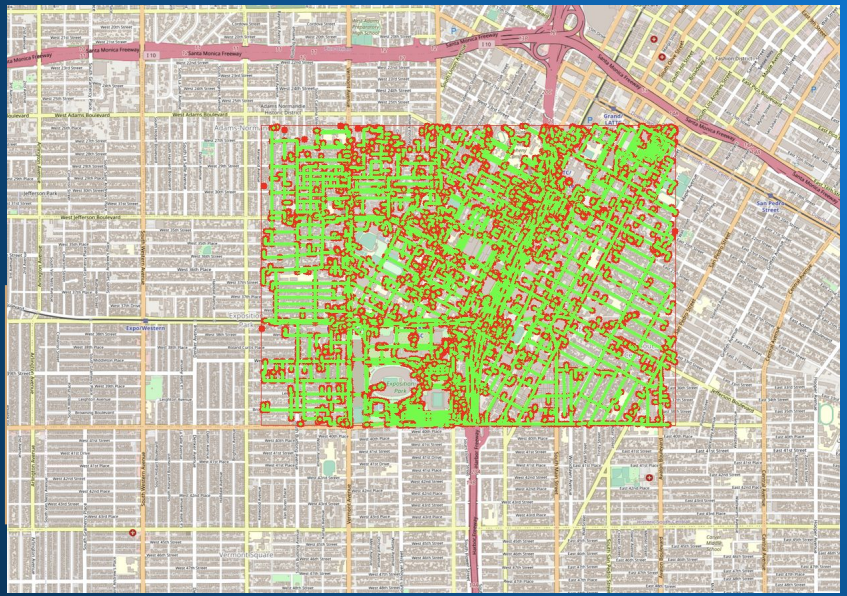
Examples	Time Taken by Brute-Force Algorithm (ms)	Time Taken by Backtracking Algorithm (ms)	Time Taken by 2-opt Heuristic Algorithm (ms)
8	18	22	0
9	168	77	0
10	4297	917	1
11	6592	1568	1
12	412980	15693	3

Runtime comparison of the three TSP algorithms



5. Cycle Detection

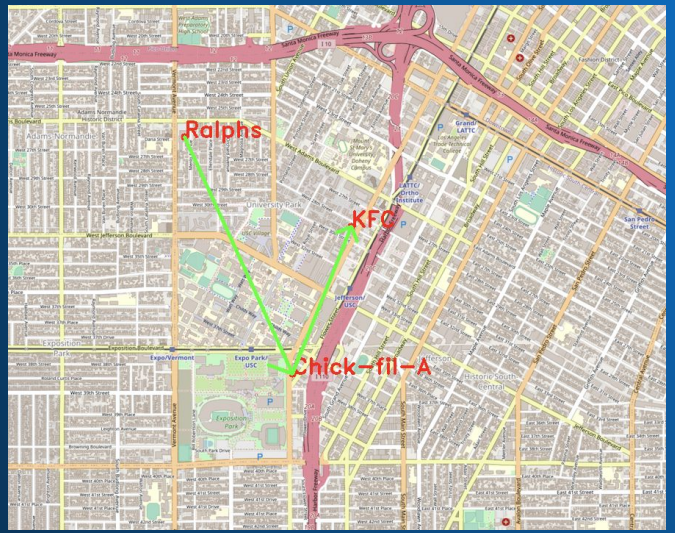
- ➔ Given four points of the square-shape subgraph the algorithm returns true if cycle is present else False.
- ➔ Time complexity : O(V + E) where V is the number of vertices and E is the number of edges
- ➔ Example : Input : square = {-118.299, -118.264, 34.032, 34.011}
  Output : true



```
************************************************************
* 5. Cycle Detection
************************************************************

Please input the left bound longitude(between -118.320 and -118.250):-118.299
Please input the right bound longitude(between -118.320 and -118.250):-118.264
Please input the upper bound latitude(between 34.000 and 34.040):34.032
Please input the lower bound latitude(between 34.000 and 34.040):34.011
**************************Results*****************************
there exists a cycle in the subgraph
************************************************************
Time taken by function: 2 ms
```

6. Delivering Trojan

- ➔ Given a vector of location Names and dependencies the algorithm returns a feasible route satisfying all dependencies
- ➔ We use an unordered map to create a Directed Acyclic Graph from which we get the list of adjacent locations.
- ➔ Time Complexity : O(V+E) where V is the number of vertices and E is the number of edges



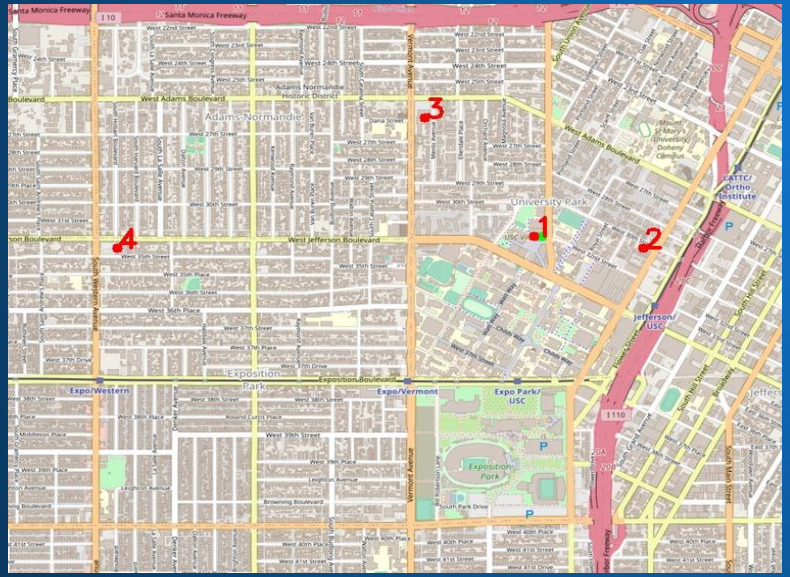

```
**************************************************************
* 6. Topological Sort
**************************************************************

Please input the locations filename:
Please input the dependencies filename:
**************************Results*****************************
Topological Sorting Results:
Ralphs
Chick-fil-A
KFC
**************************************************************
Time taken by function: 1 ms
```

7. Find Nearby

Input : Attribute type, Location name, radius , k number of locations

Output : A list of k locations closest to the name of interest within radius r.

Time Complexity: O(n)




```
**************************************************
* 7. Find Nearby
**************************************************

Please input the attribute:supermarket
Please input the locations:Target
Please input radius r:10
Please input number k:10
**************************Results*****************
Find Nearby Results:
1 Trader Joes
2 Cal Mart Beer & Wine Food Store
3 Ralphs
4 Food 4 Less
**************************************************
Time taken by function: 28 ms
```

Future Work

- ➔ We can make more improvements such as adding a stop to the trip.
- ➔ Provide different routes based on the mode of transportation.
- ➔ We can explore on other algorithms to find the shortest path between two locations.
- ➔ We will work on improving the GUI

Objectives Learned

- ➔ The course structure created increased our knowledge on data structures by implementing a variety of algorithms.
- ➔ Learnt the importance of choosing appropriate data structures when solving a problem through this project and assignments.
- ➔ The various algorithms learnt in this project were Dijkstra, Bellman-Ford, 2-opt for TSP, Cycle Detection and Topological Sort.

THANK YOU