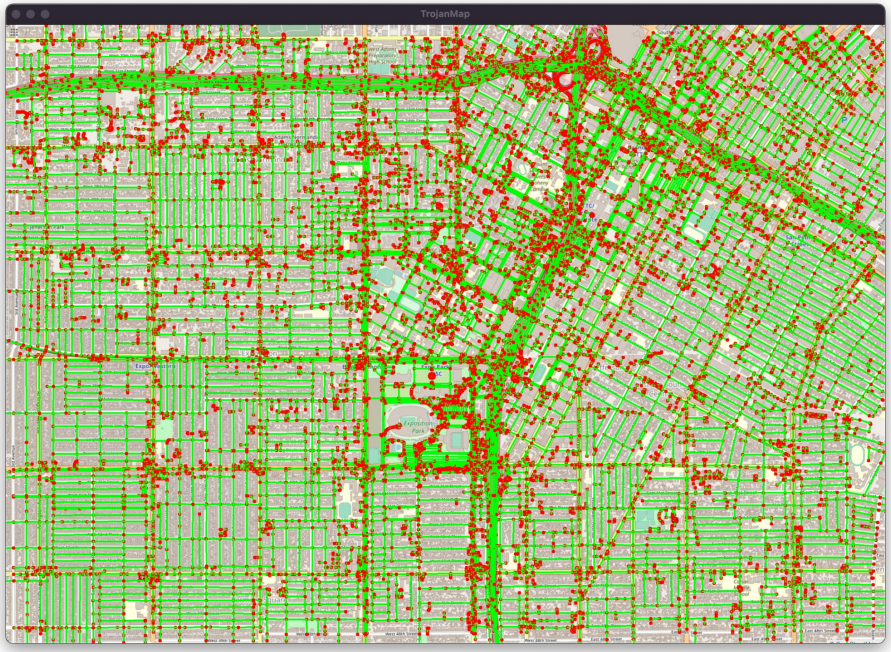
USC Viterbi
School of Engineering

Trojan Map

Presenters:
Amrith Coumaran
Sudharshan Subramaniam Janakiraman

Implement Features of a Mapping Application




```
TrojanMap
*****************************************************************
* Select the function you want to execute.
* 1. Autocomplete
* 2. Find the location
* 3. CalculateShortestPath
* 4. Travelling salesman problem
* 5. Cycle Detection
* 6. Topological Sort
* 7. Find Nearby
* 8. Exit
*****************************************************************
```

Feature : Find Location

Provides the Exact location(Latitude and Longitude)on the map based on the Location name.

Location Name

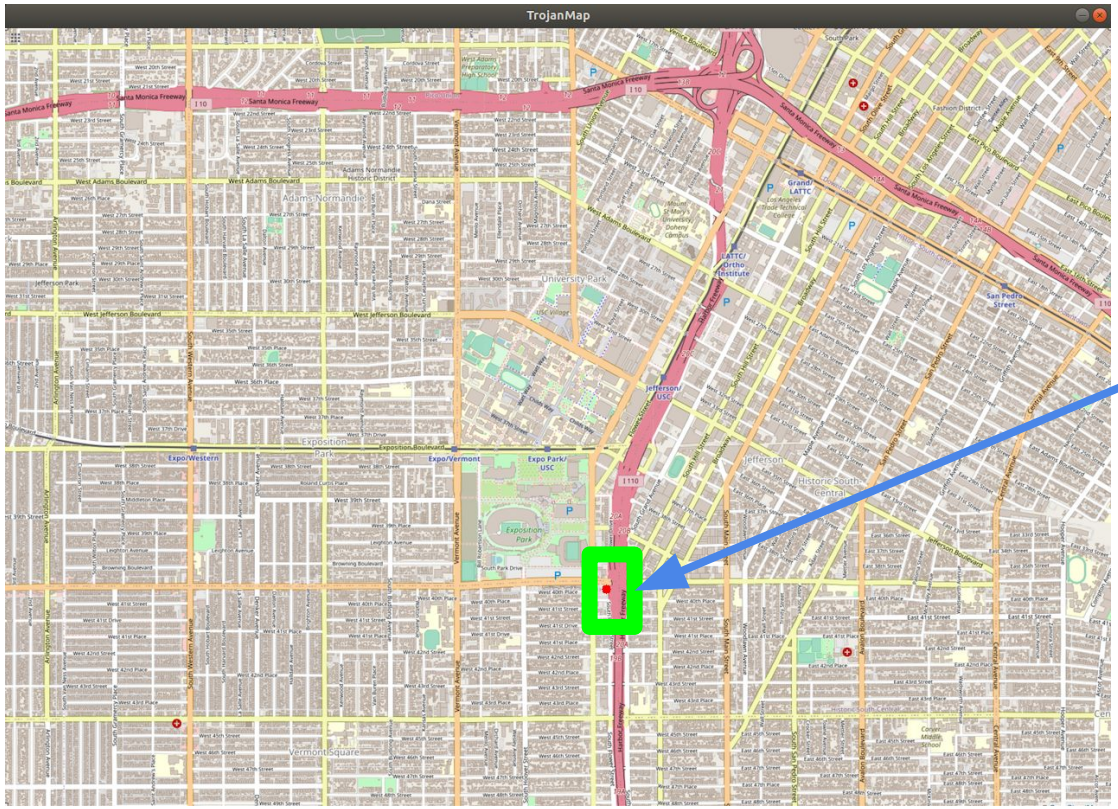
```
* 2. Find the location
*****************************************************************

Please input a location McDonalds
************************Results*****************************
Latitude: 34.0108 Longitude: -118.282
*****************************************************************
```

Latitude

Longitude

Feature : Find Location

Provides the Exact location(Latitude and Longitude)on the map based on the Location name.



What if user makes an spelling mistake ?

We find the closest available name in the map using **FindClosestName().**

- Time Complexity **O(nlp)**
- **n = #Unique ID's, l = Input name Length, p = Comparison Name Length.**
- Utilizes **Dynamic Programming : Tabulation Approach.**
- Find closest name based on shortest **EditDistance O(n^2).**

Incorrect Name

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

Please input a location McDonaldDuck
*****************************Results*****************************
No matched locations.
Did you mean McDonalds instead of McDonaldDuck? [y/n]y
Latitude: 34.0108 Longitude: -118.282
*****************************************************************
Time taken by FindClosestName function: 37 ms
```

Latitude of Corrected Name

Longitude of Corrected Name

Corrected Name

Time to Find the Closest Name on map

What if user makes an spelling mistake ?

Incorrect Name

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

Please input a location McDonaldDuck
**********************Results**************************************
No matched locations.
Did you mean McDonalds instead of McDonaldDuck? (Y/n)y
Latitude: 34.0108 Longitude: -118.282
**********************Results**************************************
Time taken by FindClosestName function: 37 ms
```

Corrected Name

Time to Find the Closest Name on map

Latitude of Corrected Name

Longitude of Corrected Name

		H	E	L	L	O
	0	1	2	3	4	5
H	1	0	1	2	3	4
O	2	1	1	2	3	3
L	3	2	2	1	2	3
A	4	3	3	2	2	3

Edit Distance : #Operations needed to convert a string to another string

3 Operations needed to convert **Hello** to **Hola.**

Feature : Calculate Shortest Path

Provides the shortest path on map from source to destination

Implementation:

- **Dijkstra** | Time Complexity : **O ((m+n)log n)**
- **Bellman Ford** | Time Complexity : **O(mn)**

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

Please input the start location:Vermont & 39th (Metro 204 Northbound) (#05658)
Please input the destination:McDonalds
**************************Dijkstra*****************************
**************************Results******************************
"6512303434","4015372453","6792034223","6815190429","122670230","4020099362","6813379581","1869430500","5481562307","16
30940734","5618016862","1630940732","1630940683","6814481787","1832254580","213431660","1630944607","1768800493","18551
44017","1855144898","1855143713","4060034843","1855143710","1855145664","7404342034","1855145665","1855166098","8383519
583","1855166099","1855173102","1732243610","6653019471","6653019480","6653023687","6653019476","6653019472","665301947
3","358789632","1759017528",
The distance of the path is:0.971707 miles
*************************************************************
Time taken by function: 54 ms

**************************Bellman Ford*************************
**************************Results******************************
"6512303434","4015372453","6792034223","6815190429","122670230","4020099362","6813379581","1869430500","5481562307","16
30940734","5618016862","1630940732","1630940683","6814481787","1832254580","213431660","1630944607","1768800493","18551
44017","1855144898","1855143713","4060034843","1855143710","1855145664","7404342034","1855145665","1855166098","8383519
583","1855166099","1855173102","1732243610","6653019471","6653019480","6653023687","6653019476","6653019472","665301947
3","358789632","1759017528",
The distance of the path is:0.971707 miles
*************************************************************
Time taken by function: 10303 ms
```

Time Taken By Dijkstra

Shortest Path

Time Taken By Bellman Ford

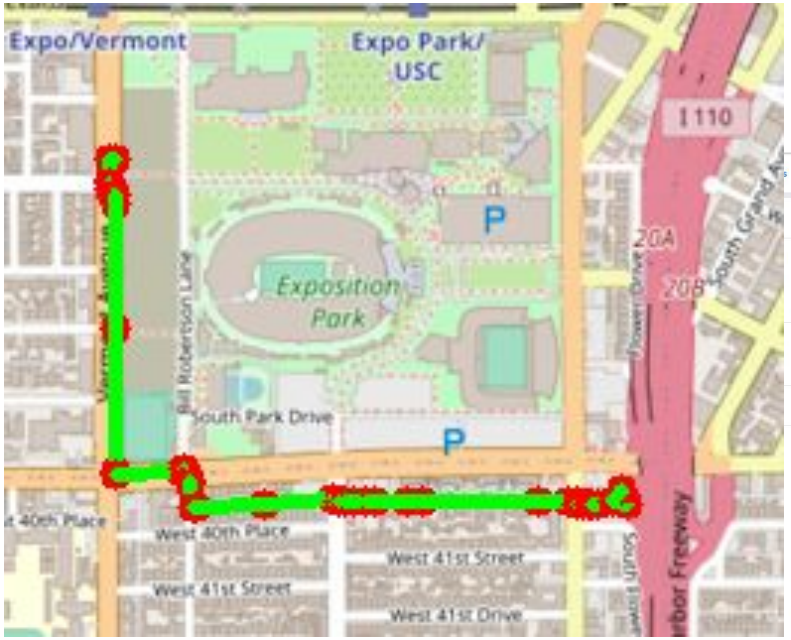
Bellman Ford is Slower Compared to Dijkstra

Feature : Calculate Shortest Path

Provides the shortest path on map from source to destination

Dijkstra's Shortest Path Algorithm

- Greedy Algorithm
- Max Heap Data Structure
- Time Complexity : **O((m+n)(logn))**



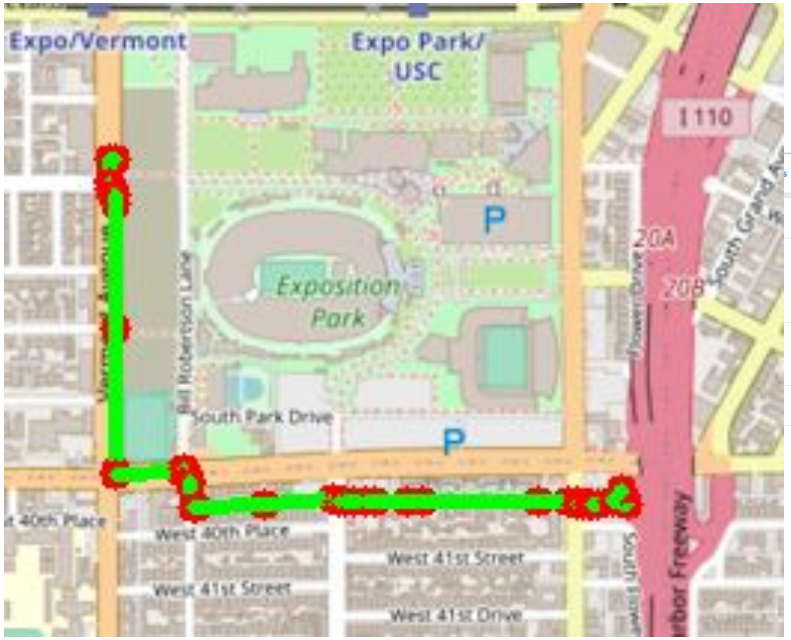
Google maps shows almost similar route (Takes multiple factors into account

Feature : Calculate Shortest Path

Provides the shortest path on map from source to destination

Bellman Ford Shortest Path Algorithm :

- Versatile Graph Algorithm
- Early Stopping When No Change in Shortest Path
- Time Complexity = **O(mn)**



Google maps shows almost similar route (Takes multiple factors into account

Dijkstra VS Bellman Ford

Shortest Distance Calculated by Dijkstra and Bellman Ford

Source	***Destination***	***Dijkstra***	***Bellman Ford***	***Google Maps***
Ralphs	Target	0.927 miles	0.927 miles	0.9 miles
FaceHaus	Western & Adams 3	2.002 miles	2.002 miles	2 miles
Vermont & 39th	McDonalds	0.972 miles	0.972 miles	0.9 miles

Bellman Ford is Slower Compared to Dijkstra

Dijkstra VS Bellman Ford

Running Time Comparison between Dijkstra and Bellman Ford

Source	***Destination***	***Dijkstra***	***Bellman Ford***	***Bellman Ford without early stopping***
Ralphs	Target	39 ms	7859 ms	43 Minutes Approx
FaceHaus	Western & Adams 3	167 ms	8814 ms	43 Minutes Approx
Vermont & 39th	McDonalds	54 ms	10303 ms	43 Minutes Approx

Bellman Ford is Slower Compared to Dijkstra

Feature : Travelling Salesman Problem

Provides the shortest path on map to visit all given location only once and return to where the user started

Algorithm:

- Brute Force : Generate All Possible combinations
- BackTracking : Generate Some Permutations and skip others which does not have the best cost
- 2 Opt : Heuristic Based Approach

```
*************************************************************
* 4. Travelling salesman problem
*************************************************************

In this task, we will select N random points on the map and you need to find the path to travel these points and back to the start point.

Please input the number of the places:4
"4696690002","1836119965","7199106822","123120136",
Calculating ...
*************************Results*****************************
TravellingTrojan Brute force
"4696690002","7199106822","123120136","1836119965","4696690002",
The distance of the path is:5.28877 miles
*************************************************************
You could find your animation at src/lib/output0.avi.
Time taken by function: 0 ms

Calculating ...
*************************Results*****************************
TravellingTrojan Backtracking
"4696690002","7199106822","123120136","1836119965","4696690002",
The distance of the path is:5.28877 miles
*************************************************************
You could find your animation at src/lib/output0 backtracking.avi.
Time taken by function: 0 ms

Calculating ...
*************************Results*****************************
TravellingTrojan 2opt
"4696690002","7199106822","123120136","1836119965","4696690002",
The distance of the path is:5.28877 miles

You could find your animation at src/lib/output0 2opt.avi.
Time taken by function: 0 ms
```

BRUTEFORCE Shortest Path

BACKTRACKING Shortest Path

2opt Shortest Path

Travelling Salesman Problem (Brute Force)

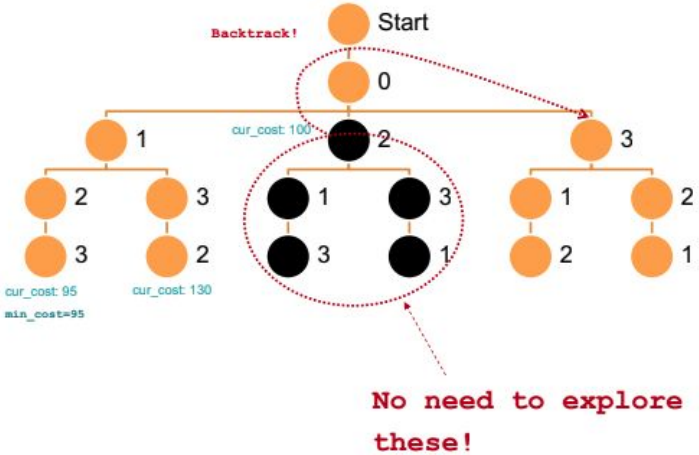
Generate All possible Combination of path and selects the path which has shortest distance.

Time Complexity O(n!)

Travelling Salesman Problem (BackTracking)

Skips combination of path does not give optimal cost at each iteration

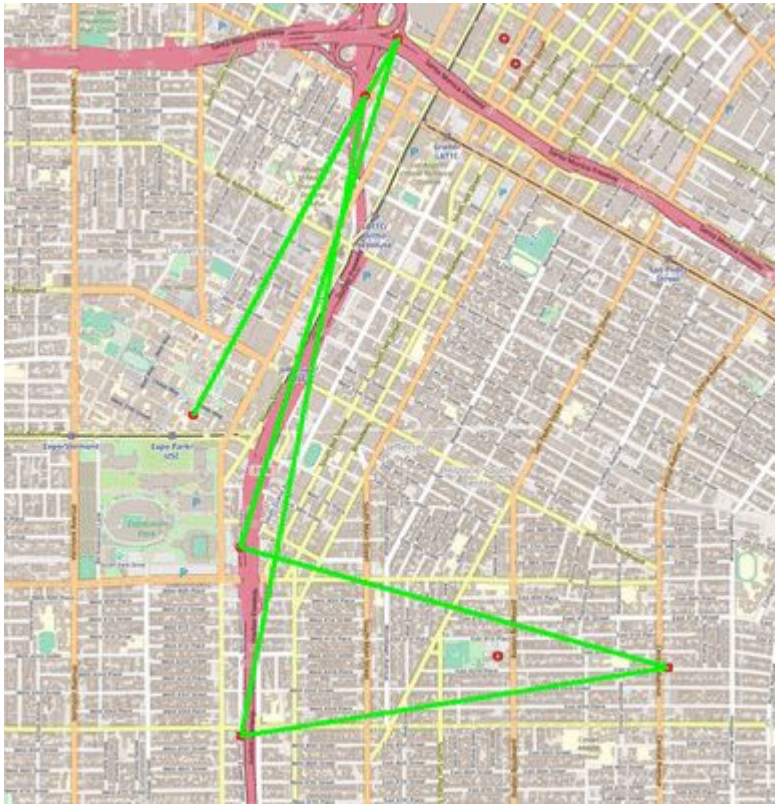
Time Complexity : O((n-1)!)



Travelling Salesman Problem (2-opt)

Heuristic Based Approach which fastens the computational time

- Not Always guaranteed to provide optimal solution
- Iterates through all combinations of 2-opt swap/move and calculates the best distance
- reverses the path between two places if it is optimal



Time Complexity : O((n^ 2) * optimal iteration)
Optimal iteration = #iterations after which no
improvement is found

Interviewer: It say that you are extremly fast at Math. What is 34x23?

Me: 45

Interviewer: That's not even close

Me: Yeah but it was fast

2-opt Approach does not always give the optimal solution. It is much faster compared to Backtracking when the number of places increase.

Feature : Travelling Salesman Problem

Time and Cost Comparison of TSP Algorithms

	BruteForce (ms)	Backtracking (ms)	2opt (ms)	Cost Bruteforce / Backtracking (miles)	Cost 2opt (miles)
4	0	0	0	4.4766	4.4766
6	16	34	4	7.0898	7.0898
8	1182	2378	767	10.4065	10.4065
10	137199	258405	13932	12.7386	12.7386
12	-	-	21341	-	11.276

Algorithm Exceeds 10 minute for Brute Force & Backtracking

2 opt still finds answer
(Might or might not be optimal)

Feature : Cycle Detection

Checks if there a cycle exists in the subgraph

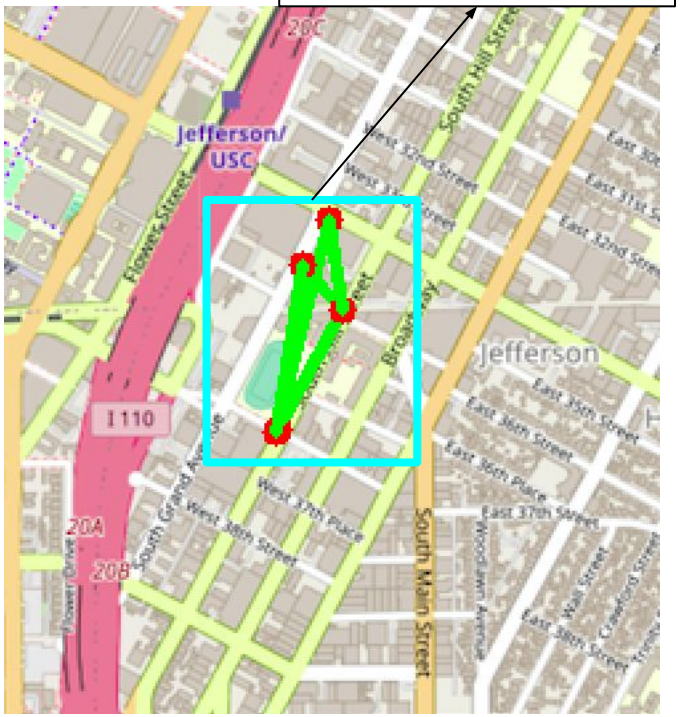
- Get the nodes in the given range and generate a subgraph
- Perform DFS recursively on the given ids
- Keep updating the predecessor map for plotting purposes
- **Time Complexity = O(m+n)**

Input Sub-graph boundaries

```
*************************************************************
* 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):34.020
Please input the lower bound latitude(between 34.000 and 34.040):34.010
*************************Results*****************************
there exists a cycle in the subgraph
*************************************************************
Time taken by function: 115 ms
```

Output

Detected cycle



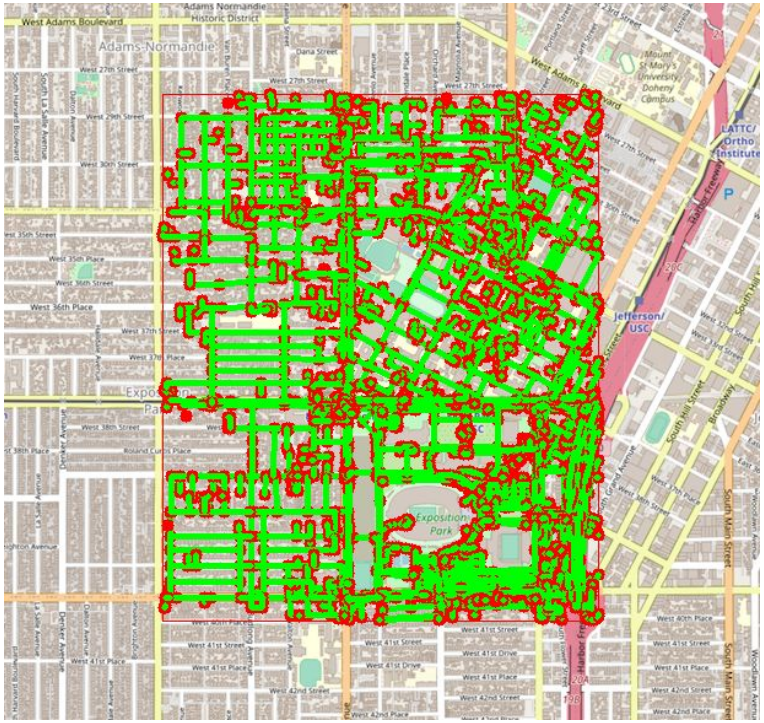
Feature : Cycle Detection

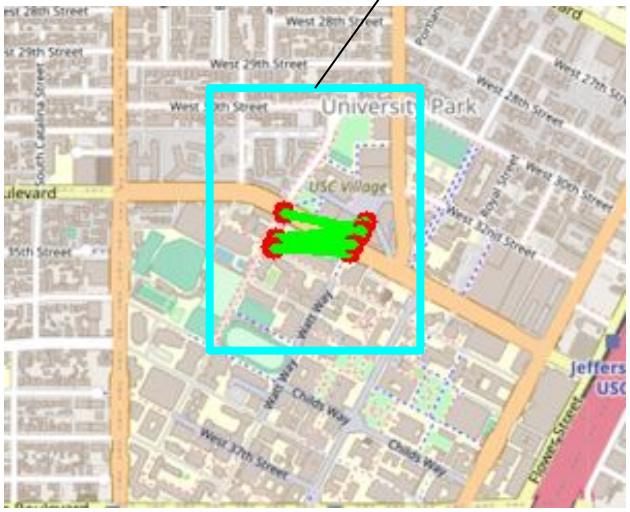
```
*************************************************************
* 5. Cycle Detection
*************************************************************

Please input the left bound longitude(between -118.320 and -118.250):-118.300
Please input the right bound longitude(between -118.320 and -118.250):-118.280
Please input the upper bound latitude(between 34.000 and 34.040):34.030
Please input the lower bound latitude(between 34.000 and 34.040):34.010
*************************Results*************************
there exists a cycle in the subgraph
*************************************************************
Time taken by function: 13 ms
```

Input Sub-graph boundaries

Output

First Detected cycle





Feature : Cycle Detection

Input Sub-graph boundaries

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

Please input the left bound longitude(between -118.320 and -118.250):-118.290
Please input the right bound longitude(between -118.320 and -118.250):-118.289
Please input the upper bound latitude(between 34.000 and 34.040):34.025
Please input the lower bound latitude(between 34.000 and 34.040):34.020
*************************Results*****************************
there exist no cycle in the subgraph
*****************************************************************
Time taken by function: 0 ms
```

Output

Linear Path

Feature : Cycle Detection

Two points are not considered as a cycle

Input Sub-graph boundaries

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

Please input the left bound longitude(between -118.320 and -118.250):-118.300
Please input the right bound longitude(between -118.320 and -118.250):-118.299
Please input the upper bound latitude(between 34.000 and 34.040):34.002
Please input the lower bound latitude(between 34.000 and 34.040):34.001
                        Results
*************************************************************
there exist no cycle in the subgraph
*************************************************************
Time taken by function: 0 ms
```

Output

Linear Path

Feature : Topological Sort

Finds a path using the given IDs based on some conditions

- Generate a DAG map from the given inputs
- Check if the input has a cycle using DFS for directed acyclic graphs
- Perform DFS recursively on the given ids using the DAG map
- Time Complexity = O(m+n)

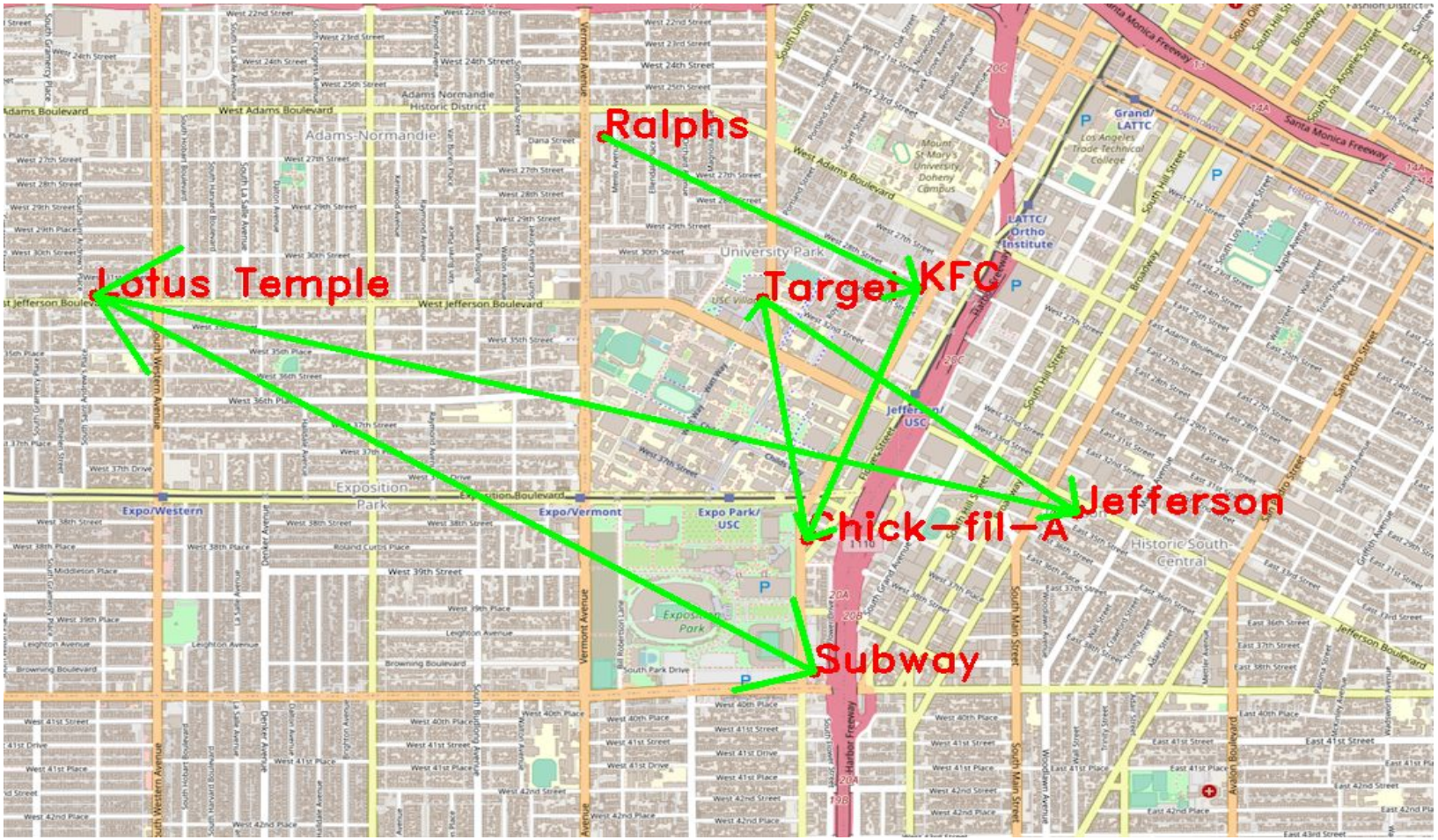
input > topologicalsort_locations.csv

```
Name
Ralphs
Target
KFC
Chick-fil-A
Lotus Temple
Jefferson
Subway
```

topologicalsort_dependencies.csv

```
Source, Destination
Ralphs,Target
Ralphs,KFC
Chick-fil-A,Target
KFC,Target
Target,Lotus Temple
Target,Jefferson
KFC,Subway
```

Feature : Topological Sort



Feature : Topological Sort

- If the dependencies graph has a cycle

input > topologicalsort_locations.csv

```
Name
Ralphs
Target
KFC
Chick-fil-A
Lotus Temple
Jefferson
Subway
```

topologicalsort_dependencies.csv

```
Source, Destination
Ralphs,Target
Target,Ralphs
Ralphs,KFC
Chick-fil-A,Target
KFC,Target
Target,Lotus Temple
Target,Jefferson
KFC,Subway
```

Cycle

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

Please input the locations filename:/home/amrith/Documents/ee538/final-project-SudharshanSubramaniamJanakir
aman/input/topologicalsort_locations.csv
Please input the dependencies filename:/home/amrith/Documents/ee538/final-project-SudharshanSubramaniamJana
kiraman/input/topologicalsort_dependencies.csv
************************Results*************************************
There is no topological sort for the given graph.
*************************************************************
Time taken by function: 0 ms
```

Output

Feature : Find Nearby

Returns k locations in the radius r from a given location which belonged to a specific class

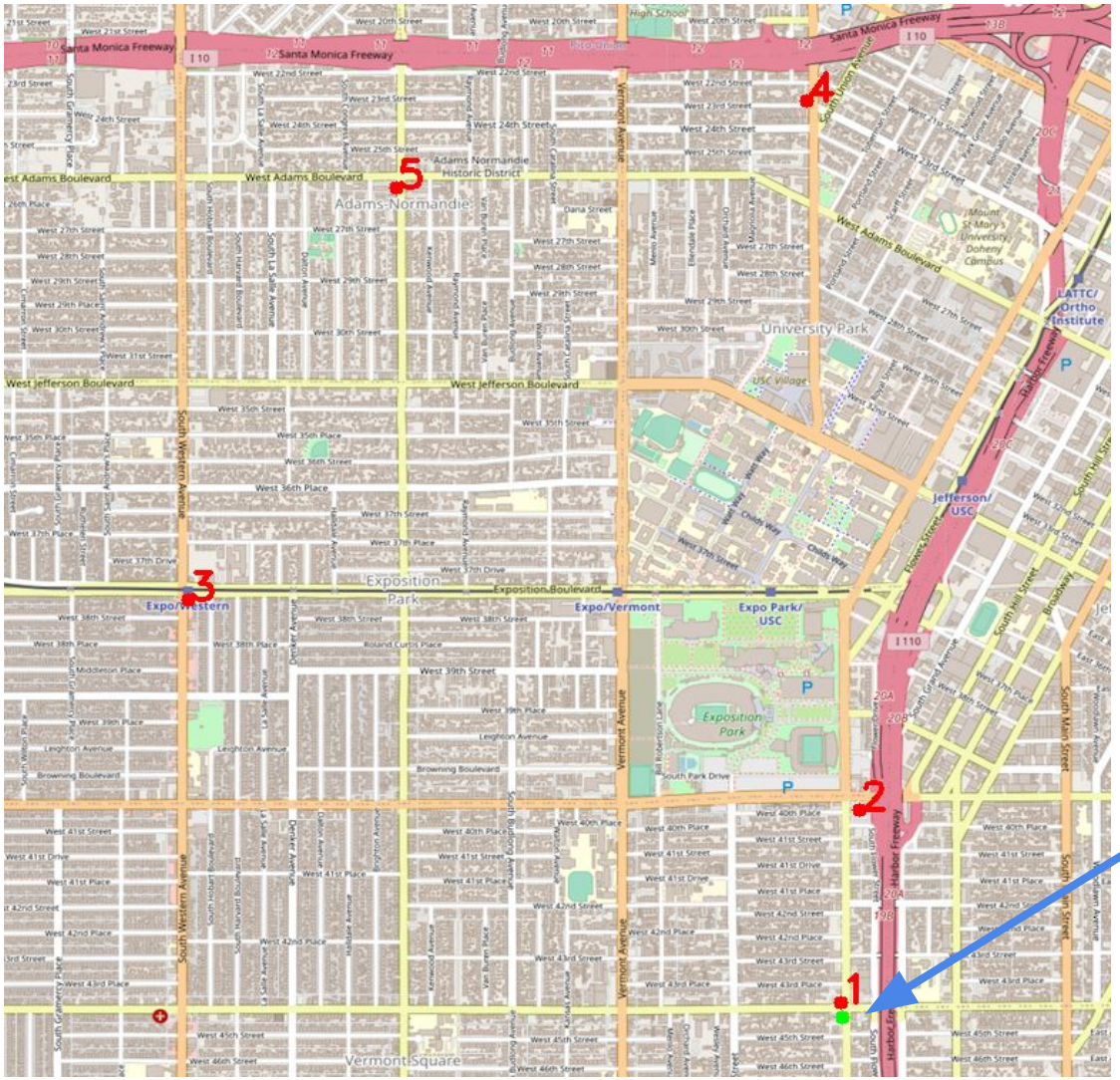
- Iterate through all data points which satisfies the given conditions
- Use STL heap function to sort it based on distances
- Time Complexity = O(n)

Inputs

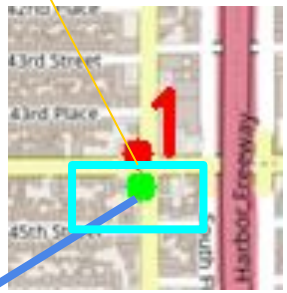
```
**********************************
* 7. Find Nearby
**********************************

Please input the attribute:fuel
Please input the locations:Shell
Please input radius r:10
Please input number k:5
**********************Results**
Find Nearby Results:
1 76
2 Chevron
3 76 1
4 Arco
5 Adams Fuel
**********************************
Time taken by function: 130 ms
```

Output

Feature : Find Nearby



Center Location



Future Improvements

- Heuristic approaches for Travelling Salesman problem
- Interactive tool for User-interface Input and Data Visualization

Lesson Learnt

- Choosing different data structures and algorithms, will impact the performance of problem being solved. (Eg: Dijkstra will perform better using heap)

Course Outcome

- Successfully learnt about utilization of various data structures as well as optimized algorithms for problem solving.
- Learnt how to optimally select algorithms and data structures for a given problem.
- Understood version control using git

Thank you!