Public Transport Selection Guidance System

Team:

Ragavan Srinivasan (G01276644)

Gautham Konda Varadarajan (G01342512)

Harivignesh Gomathi Sankara Guru (G01281331)

Achudan Thudukutchi Sadhasivam (G01341497)

Goal:

To build a decision guidance system that helps users decide on the path and mode of transport that suits their interests/needs when travelling from a source to destination city. The user constraints that are taken into consideration are cost, duration, and comfort of transport.

Solution Proposed:

We plan to construct a graph based on the input that we have gathered. Each city from our data denotes a vertex and edge between two vertices are created based on the presence of travel possibility between the two cities.

Once we create a graph, we used Yen's K shortest path algorithm to list the top 5 possible results on each criterion.

We prevented the possibility of showcasing exponential results by considering only the k-top results using the yen's K shortest path algorithm. Here the weightage for results varies based on the user constraints, i.e., if user choose to receive a path that has least cost, then our results are based only on the top k paths with least cost. If users prefer the paths with least travel duration, then the results are displayed based on the path that has the least duration.

Work Split:

Everyone: Researched on the multiple use case scenarios/problem statements along with possible solutions.

Ragavan: Took the top 50 US cities along with their latitude & longitude. Used a python script to fetch the aerial distance between the two cities using distance24 API. Visualized the problem as a graph and created an initial prototype by constructing a graph and traversed it with DFS and then filtered it based on start-end time, source and destination matches. Created helper functions to find the reaching time for a path by consolidating start time and duration.

Achudan & Gautham: Researched and found Yen's algorithm which matched our requirements. Used K shortest path algorithm to efficiently avoid the scenario of exponential results. Implemented logics to

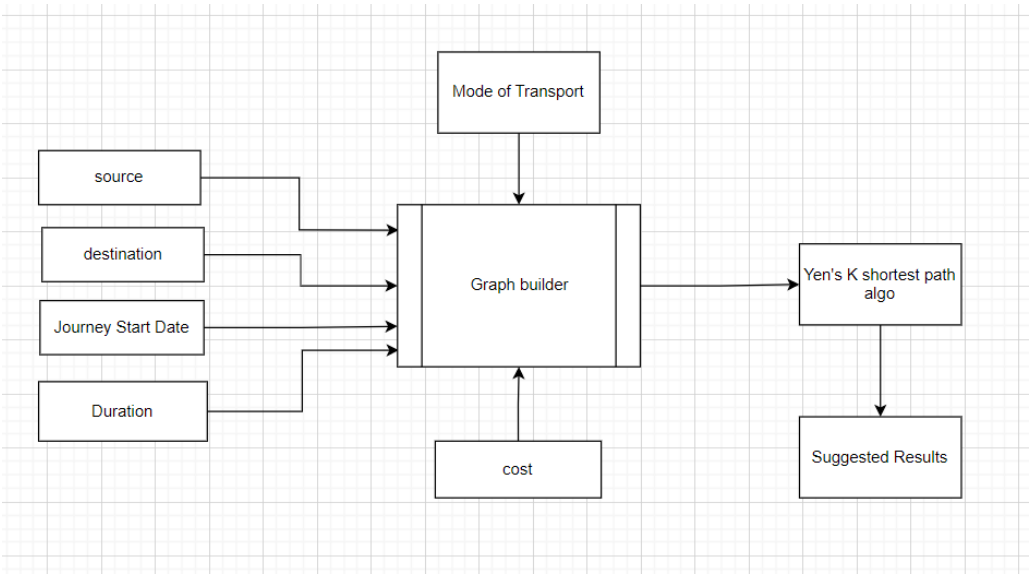
check the time constraints while showcasing the possible graph paths. Implemented logics to make sure different modes of transport are considered while suggesting results to the user.

Gautham: Along with the above mentioned, modifies Yen's algorithm to handle invalid edges based on start time and duration.

Achudan: Developed script using Java to create combination of data. Refactored code and added new classes to adapt the Yen's algorithm to our requirements. Modified Yen's algorithm to handle M stops.

Harivignesh: Developed a logic to fetch data from google maps API for real-time distance, routes, travel duration and available modes of transportation between a pair of cities. Predominantly used this data to update our input for bus and train modes of transport. Updated the finalized yen's algorithm to read and process the input from input files that we gathered.

.



Initial Work:

Constructed a graph by using cities as vertices and creating an edge between two cities if there exists a path. Cost, start time, duration is added to the adjacency list between the source and destination. Traversed the graph with DFS and found the overall possible solution between the source and destination cities. Found overall costs for each of the filtered results,

Initial Output:

```
Src ->  New York  ; Destination ->  Milwaukee  ; Cost ->  234.0 ; Strt Date & Time  2022-12-14 07:09:53
234.0

Src ->  New York  ; Destination ->  Washington  ; Cost ->  345.0 ; Strt Date & Time  2022-12-15 07:09:53
Src ->  Washington  ; Destination ->  Milwaukee  ; Cost ->  456.0 ; Strt Date & Time  2022-12-17 15:32:38
801.0
```

Public Transport Selection Guidance System

Finalized solution:

The initial results did not include modes of transport and specify the difference between each mode of transport. Also, the initial results were not working properly for certain start time constraints. To fix all this issues, we sed Yen's algorithm which was an extension to the Dijkstra's algorithm to filter the top k results. Modified the algorithm for our requirement by making sure that travel duration, cost and travel comfort are considered. Created a logic to give weightage based on the user constraints and used that weightage to sort the results. Filtered the results to make sure that start time of the subsequent destination is greater than the time it took to reach that destination.

Data Collection Techniques:

We have gathered data for three different modes of transport – air, bus, and train. Currently, the data was constructed for the 50 major cities in the USA for the travel dates between 17th December to 21st December. Though the data is not real time, we have made sure that the data depicts values that are close to real time for both flight, bus, and train. For instance, the distance between the two cities is calculated google distance matrix API and distance24 API. We have used a reference table to find the co-ordinates of the all the 50 cities and used that data to find the distance between two cities.

The cost and travel duration were calculated based on the travel distance. The proportionality constant is decided based on the average speed of bus, train, and commercial aircraft. The cost is also determined with similar ratio that is near to real time values.

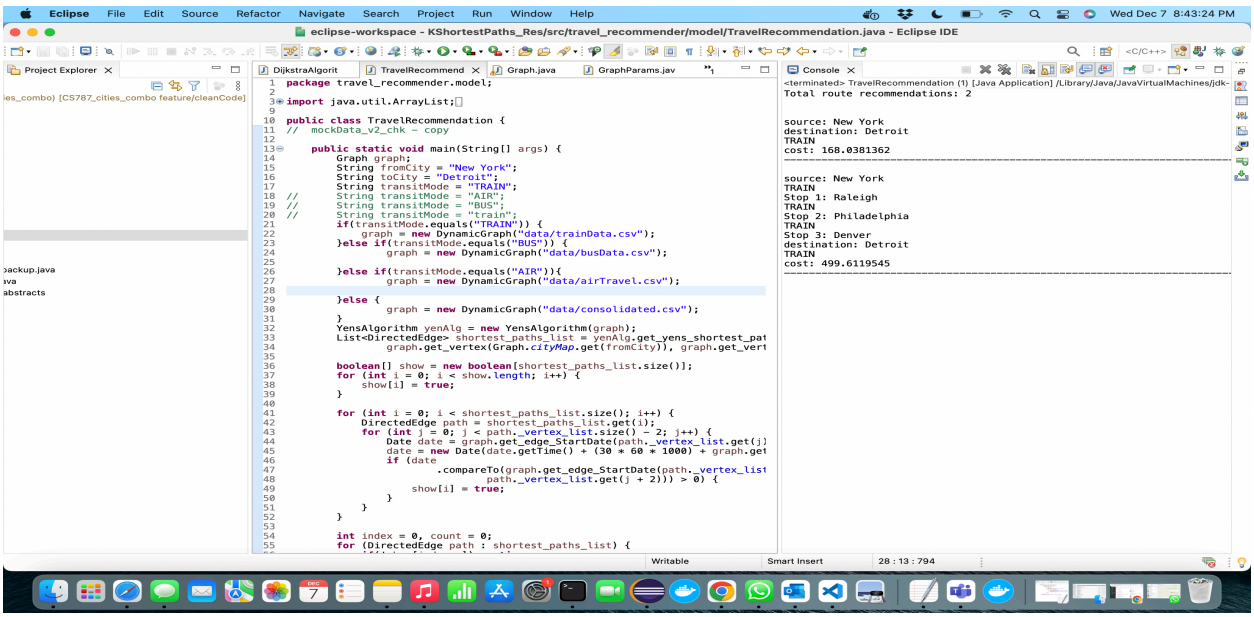
Final Results:

Scenario 1

From City: New York

To City: Detroit

Constraints: Single mode of transportation



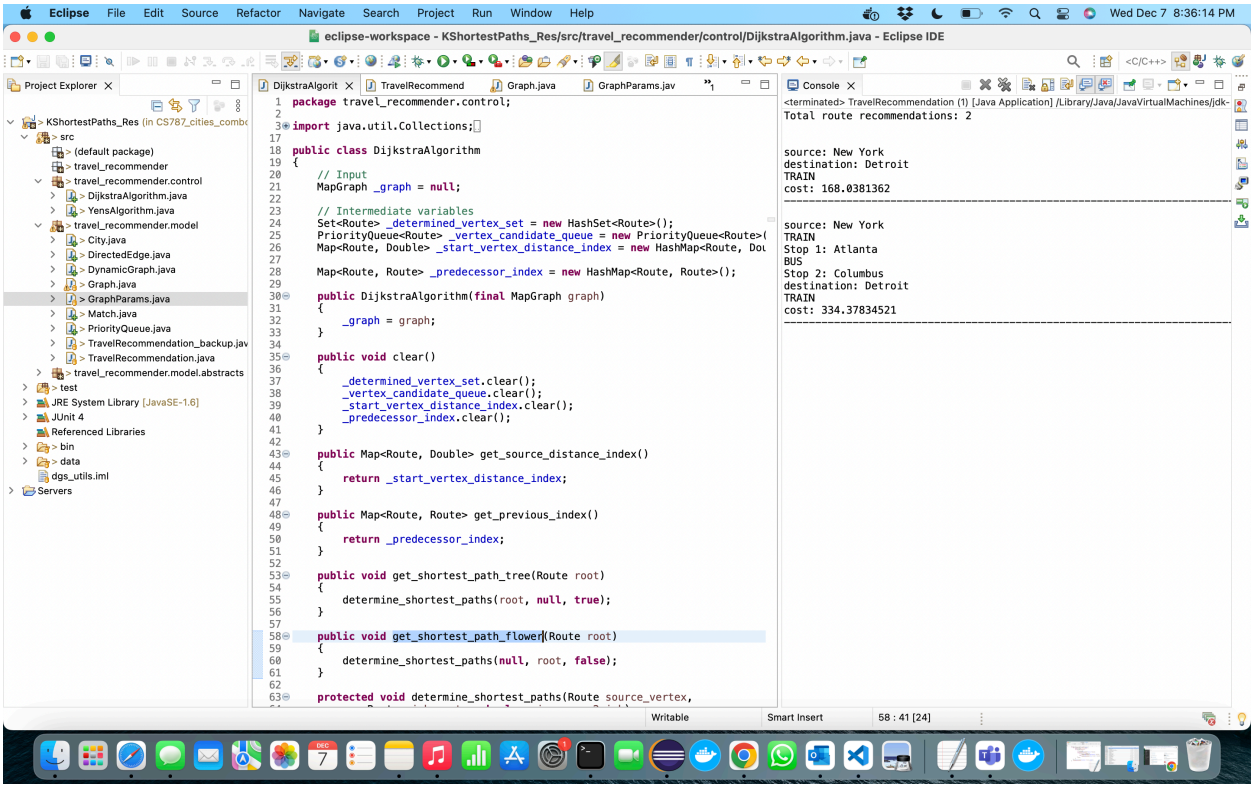
Public Transport Selection Guidance System

Scenario 2

From City: New York

To City: Detroit

Constraints: Any/mixed mode of transport



Solution Proposed:

- In future we plan to include more geographic locations other than cities into our system.
- Build an user interface for the application for better user experience.
- More filter options other than cost and duration to provide more flexibility to the users in filtering the output.

Reference:

https://en.wikipedia.org/wiki/Yen%27s_algorithm

https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

https://developers.google.com/maps/documentation/directions/get-directions?hl=en_GB - mode