

CSU22012: Data Structures and Algorithms Project:

DESIGN DOCUMENT:



Name: Karthik Venkatesh Nagaraj
TCD ID: 20330418

- **Part 1:**

For the first part of this assignment, the way I went through with completing it, is that I created classes **stopsEdges**, **stopsInfo** and **tripInfo** which hold the information from the transfers.txt, stops.txt and stops_times.txt files respectively.

And once all the input files had been parsed through, the program takes in the initial stop and the destination stop from the main function, and uses Dijkstra's Shortest Path Algorithm to find the shortest

```
-----
Graph Setup
-----
StopsFile Completed
TransfersFile Reading Completed
StopTimes File Reading Completed
Graph Setup Complete.
Cost to travel between these stops: 10.0
Stops in Between:
1111
->
1112
->
1065
->
8046
->
8039
->
8066
->
10429
->
12366
->
2222
```

path between the two stops, calculates the cost to travel between them, and also prints a list of stops between the two chosen spots. For eg: The results for the

stops “1111” and “2222”. We used **Dijkstra** because we thought it would be far more efficient to find the shortest path between any two stops rather than using Floyd Warshall where we would find the shortest path between every stop.

- **Part 2:**

Part 2 involved implementing a search function where the user has the option to enter a string which contains the first few words from a bus stop’s name, then displays a list of meaningful stops. Inorder to do this, we used a Ternary Search Tree as instructed by the assignment requirements. I created a TST class which implements a Node class which contains the properties of a node in the tree.

```
-----
Search Results:
-----
HASTINGS FS CLIFF AVE WB
HASTINGS ST FS ALPHA AVE WB
HASTINGS ST FS BETA AVE EB
HASTINGS ST FS BOUNDARY RD EB
HASTINGS ST FS DUTHIE AVE EB
HASTINGS ST FS DUTHIE AVE WB
HASTINGS ST FS FELL AVE EB
HASTINGS ST FS FELL AVE WB
HASTINGS ST FS GAMMA AVE WB
HASTINGS ST FS GILMORE AVE EB
HASTINGS ST FS GILMORE AVE WB
HASTINGS ST FS HOLDOM AVE EB
HASTINGS ST FS HOLDOM AVE-- WB
HASTINGS ST FS HOWARD AVE EB
HASTINGS ST FS HOWARD AVE WB
HASTINGS ST FS HYTHE AVE EB
HASTINGS ST FS HYTHE AVE WB
HASTINGS ST FS INGLETON AVE EB
HASTINGS ST FS INGLETON AVE WB
HASTINGS ST FS INLET DR EB
HASTINGS ST FS KENSINGTON AVE EB
HASTINGS ST FS KENSINGTON AVE WB
HASTINGS ST FS MACDONALD AVE EB
HASTINGS ST FS MADISON AVE EB
HASTINGS ST FS MADISON AVE WB
HASTINGS ST FS SPERLING AVE EB
HASTINGS ST FS SPERLING AVE WB
HASTINGS ST FS SPRINGER AVE EB
HASTINGS ST FS SPRINGER AVE WB
HASTINGS ST FS WILLINGDON AVE WB
HASTINGS ST NS ALPHA AVE EB
HASTINGS ST NS WILLINGDON AVE EB
Length 33
```

Another requirement from the project specification required us to make the stop names more meaningful, which meant moving keywords like FLAGSTOP, WB, SB, EB and NB towards to the end of the string. For eg: “WB HASTINGS ST FS HOLDOM AVE” becomes “HASTINGS ST FS HOLDOM AVE WB”.

- **Part 3:**

This part required us to implement a feature where the user can input a string of the format “hh:mm:ss” and then the program

```
Your Input Time: 17:18:19
Search results are as follows:

-----
1
StopID: 297
Arrival Time: 17:18:19
-----
2
StopID: 373
Arrival Time: 17:18:19
-----
3
StopID: 430
Arrival Time: 17:18:19
-----
4
StopID: 511
Arrival Time: 17:18:19
-----
```

displays a list of bus trips that are arriving at the same time, where all the buses are sorted by means of the Stop ID. Inorder to do this, I parse the stop_times.txt file and use only the entries that have valid arrival_time and once again use the **tripInfo** class from part1, and store

it in an ArrayList<tripInfo>. The program then searches for bus trips which have arrival_time equal to the input time and displays them sorted by tripID. I used **bubble sort** to fulfill my sorting purposes.

(Image Doesn't Contain Full Result)

- **Part 4:**

To create a simple UI interface, I used the terminal screen where I implemented this ASCII Art Generator Library to create some ASCII art to make it a little more visually pleasing. I also created an infinitely-continuous menu where the user has the option to either choose 1/2/3/4 for Part 1 func./ Part 2 func. / Part 3 func. / Quit the program.

[illegible]