# Designing Documents

Name: Zhongyuan Liu

ID: 19336865

## Overview :

For this assignment, I only solved two of the three functionalities. Although the last one was not completed, I still worked very hard to design and conceive the first two, and finally achieved a relatively good effect. I made a selectable UI in the console for the convenience of the user to use this system.

## part 1:

For the first part, my solution was to use the data in three different input files to build a "graph" to concatenate everything, and then use the graph's shortest path algorithm to find the two stops the shortest distance between.

I created separate classes to create graphs, busStop, busStopConnect, etc.. For the busStop class, this class is used to store data from the stop.txt file, and the variable names in it are named according to the names given in the stop.txt file. For the class busStopConnect, this class can be thought of as the main class for making graphs. First use readBusStops() to read stops.txt, and then use HashMap<Integer, ArrayList<ConnectionNode>> to adjoin each station and station, thus successfully decomposing stop information.

Then, use readTransfer() to read the transfer.txt file and calculate the associated costs. The stop\_id and associated cost are stored using another class called nodeConnect. In this way, the direct edge of the graph has been basically added. Then use the data in stop\_times.txt to add continuous edges between stations. To break down the information in stop\_times.txt, another class called busJourney is used, which is based on another class called busJourneyDetails, and the variable names in this class are also determined by the names in the file. Read the times in the stop\_times.txt file and filter out invalid times, then add them to an array list of type busJourneyDetails in busJourney.

In order to complete the setup of the chart, a for loop needs to be used, from 1 to the number of valid strokes. Compare all consecutive trips in the loop, use makeConnection() to add consecutive edges between 2 stations with the same trip. After all directed and continuous edges are created, the graph setup is over.

I chose to use Dijkstra's shortest path algorithm because the assumption is to find the shortest path between two given stations and be the most time efficient, rather than computing the shortest path between all stations which is the most time efficient.

## part 2:

In Part 2, a class called searchTree was created to store the name of the station, then the search function was used to search for all relevant information, and a map was used to return the details of all matching stations.

First, we store all the busStopNames into searchTree, use a method called getBusStopNames() to read the stop.txt file and return an arraylist of busStopNames, then we use makeSence() to make sure the data are all meaningful, then put them added to searchTree. After the user enters the information they want to search, we use searchTree to get all matching site names. Then use the map generated from createMap() to get all the information about the matching stations and output them.

## UI:

For the UI part, I used different For loops and if conditional judgment statements to write an alternative UI interface. First, at the beginning, enter the numbers represented by different functionalities to enter the selected functionality. Then press the prompt to enter the relevant data, after running, you can get the relevant results, and finally you can enter 0 to exit.