Design Document  
Final Project CSU22012

short line

Wen Geng Lin  
9th April, 2022

# Main Features

1. Shortest Path
2. Search for bus stop by bus stop ID
3. Search for trips by time

# 

# 

# Shortest path

**Adjacency Matrix:**

Time complexity: O(1)

Space complexity: O(n^2)

An adjacency matrix was chosen to represent the bus stops (vertices) and connections (paths) between them, it was chosen due to the fact that dijkstra’s algorithm requires it or something similar and its ease of implementation over an adjacency list. The access time is great so there is no need for a better alternative.

**Dijkstra’s algorithm**

Time complexity: O(n^2)

Space complexity: O(n^2)

Dijkstra’s algorithm was chosen due to it’s speed over some other path finding algorithms such as floyd marshall.

**HashMap**

Time complexity: O(1)

Space complexity: O(n)

Two hashmaps were used to link the stop ID to the adjacency matrix and vice versa, it allowed the optimization of populating the adjacency matrix as it made the stop IDs easily organized as we can get the adjacency matrix index with an O(1) time complexity. It was chosen for it’s ability to map a key to a value.

# Search for bus stop by bus stop ID

**Ternary search tree**

Average case: O(l+ln(N))

Miss: O(ln(N))

Insert: O(L+ln(N))

It was used as it was required by the assignment guidelines

# Search for trips by time

**HashMap**

Time complexity: O(1)

Space complexity: O(n)

A hashmap was used to link each time to an arraylist of trips that have that arrival time, it was chosen as it allowed us to bypass sorting the list by time and allowed us to access the list in O(1) time complexity.

**ArrayList**

Time complexity -get: O(1)

Space complexity: O(n)

An arraylist was chosen as it allowed the storage of all the details linked to each arrival time. It has quick O(1) access time and we were unsure how large each list was going to be hence its selection over an array.