Design Document

1. Introduction

This CSU22012 final project implements a bus management system using data from Vancouver bus system. It has three main features, which allows users to find shortest paths between two bus stops, search for bus stops by name and search for trips by arrival time. For this project, I imported algorithms algs4.jar from <https://algs4.cs.princeton.edu> . I implemented my program in one BusSystem.java file. I think it is not necessary to implement several classes because the algorithms I used is imported from algs4.jar and java.util.Collections.

2.Implemenration & Algorithms Analysis

2.1 Part 1

--Implementation:

The first part of the project is to find the shortest path between two bus stops. The user enters two bus stops separated by space character, and the program will return the stops and routines along with the cost.

If the user entered a invalid bus stop id, or if the user entered a string instead of bus stop ids, the program will tell the user the input is invalid and let the user enter again.

--Algorithms analysis:

The algorithm I use for finding the shortest path is Dijkstra. Dijkstra is a single source shortest path. It can find the shortest path between two vertices, which is suitable for this case. Dijkstra works with non-negative weights, so it is appropriate for this project because all costs are positive values. Time complexity of Dijkstra's algorithm is O(ElogV). E refers to the number of edges and V refers to the number of vertices in the graph. Space complexity of Dijkstra's algorithm is O(V2). V refers to the number of vertices in the graph. Some algorithms such as Bellman-Ford is not suitable for this program because it works with non-negative cycles, but in this program the bus routines may contain cycles.

2.2 Part 2

-- Implementation:

The second part of the program is to search for bus stops by searching the full name or by first few characters. The algorithm I use for this part is a ternary search tree. If the user input is the first characters or the full name of the bus stops, the program will print out the full bus stop information.

If no bus stops were found, the program will tell the user there’s no matching bus stop.

--Algorithms analysis:

The reason for using a ternary search tree over other algorithms is that it is more space efficient. It only contains three pointers for each node. Also, ternary search tree is faster than binary search tree in terms of performance. Time complexity of ternary search tree average running time is O(log n), and the time complexity of worst case running time is O(n). In terms of space, binary search tree would require more space than ternary search tree. In this program, ternary search tree would be better than binary search tree because we’re dealing with large inputs.

2.3 Part 3

-- Implementation:

The third part of the program is to search for all trips with a given arrival time and return full details of all trips matching the criteria (zero, one or more), sorted by trip id. In this program I used collection sort imported from java.util.Collections to sort all trips by trip id, and compare given arrival time with each arrival time in stop\_times.txt.

For error checking, if the time user entered is not in the correct format or if the time starts at 27/28 hours, the program will print that the user has not entered a valid input, and will let the user type again. If there’s no trip at the given arrival time, the program will tell the user there is no matching trip.

--Algorithms analysis:

The method I used for this is Collections.sort(). The time complexity of Collections. sort() is O(n log n). The space complexity of Collections.sort() is O(log n). In this program we are dealing with large inputs. Collections.sort() has good performance. The time complexity is approximately the same as quick sort, but it has better space complexity than quick sort, which has a worst case complexity of O(n).

3.Interface

The interface enables the user to select one of the three above functionalities to perform. The interface will tell the user to type 'ShortestPath' to find shortest paths between 2 bus stops, type 'SearchBusStop' to search for bus stops, or type 'SearchTrip' to search for trips. Users also have the option to exit the program.

If the user didn’t enter any of the three features and didn’t select “Exit”, the program will print the user the input hasn’t selected a feature.