CSU22012: Data Structures and Algorithms Group Project

Due date: Friday April 30th 23:59.

Late submissions will be subject to penalty of -33% per day.

Summary/highlights:

- Implementation of a bus management system based on Vancouver bus system data
- Topics covered: graphs, searching and sorting, tries
- Group project, 4 people per group (pick your own groups)
- Register groups by April 16th and send me a link to project repository
- Submission to blackboard only – no webcat/junit tests needed
- 40% of your overall CSU22012 mark
- Deliverables: code, pre-recorded 5-minute demo, design document

Groups – deadline April 16th:

This a group assignment with 4 members per group. Please pick your own groups and email me the names and ids of the people in your group by April 16th, together with a link to the repository where you will be storing your project code, with the subject line CSU22012 Group Members (please make sure to use this subject as otherwise it will be near-impossible for me to locate the emails).

The repository should not contain just the final submission, but should start with an empty template and show the history of ALL of the contributions. You should also maintain a todo file, progress, etc in the repository, as well versions of design document. This will be used to verify that all team members have more or less equally contributed to the project. If you are using a public repository, please include only the link to it in the email. If the repository is private, you will need to add me as a member to be able to see your contributions. My github account is duspari, bitbucket one ivanad, and if you are using college-hosted gitlab.scss.tcd.ie, my college username is duspari.

If you do not have a group, I have provided a google doc which all of you should be able to edit, to help you search for partners or fill in empty spots in already formed groups. Please note smaller or larger groups are not allowed.

https://docs.google.com/document/d/1QHWl0DMUnr_oKH7HutKvNURbJ0Gvi5i9POrGdUC2jY0/edit?usp=sharing

Assignment Code Specification – 4 parts:

This assignment is less prescribed than the previous individual ones. You are given high-level specs, and any design decisions are up to you, as long as the end product meets the specifications and is efficient based on both space and time complexity. You are allowed to import any additional java classes you wish, and you do not need to write junit tests. There will be no automatic marking via webcat and submission is blackboard only.

You are given the following input files (obtained by using TransLink open API https://developer.translink.ca/ enabling access to data about Vancouver public transport system)

- stops.txt – list of all bus stops in the system, cca 8,000 entries
- transfers.txt – list of possible transfers and transfer times between stops, cca 5,000 entries

- stop_times.txt – daily schedule containing the trip times of all routes on all stops, cca 1,7 million entries

Your system needs to provide the following functionality:

1. Finding shortest paths between 2 bus stops (as input by the user), returning the list of stops en route as well as the associated "cost".

Stops are listed in stops.txt and connections (edges) between them come from stop_times.txt and transfers.txt files. All lines in transfers.txt are edges (directed), while in stop_times.txt an edge should be added only between 2 consecutive stops with the same trip_id.

eg first 3 entries in stop_times.txt are

9017927, 5:25:00, 5:25:00,646,1,,0,0,
9017927, 5:25:50, 5:25:50,378,2,,0,0,0.3300
9017927, 5:26:28, 5:26:28,379,3,,0,0,0.5780

This should add a directed edge from 646 to 378, and a directed edge from 378 to 379 (as they're on the same trip id 9017927).

Cost associated with edges should be as follows: 1 if it comes from stop_times.txt, 2 if it comes from transfers.txt with transfer type 0 (which is immediate transfer possible), and for transfer type 2 the cost is the minimum transfer time divided by 100.

2. Searching for a bus stop by full name or by the first few characters in the name, using a ternary search tree (TST), returning the full stop information for each stop matching the search criteria (which can be zero, one or more stops)

In order for this to provide meaningful search functionality please move keywords flagstop, wb, nb, sb, eb from start of the names to the end of the names of the stops when reading the file into a TST (eg "WB HASTINGS ST FS HOLDOM AVE" becomes "HASTINGS ST FS HOLDOM AVE WB")

3. Searching for all trips with a given arrival time, returning full details of all trips matching the criteria (zero, one or more), sorted by trip id

Arrival time should be provided by the user as hh:mm:ss. When reading in stop_times.txt file you will need to remove all invalid times, e.g., there are times in the file that start 27/28 hours, so are clearly invalid. Maximum time allowed is 23:59:59.

4. Provide front interface enabling selection between the above features or an option to exit the programme, and enabling required user input. It does not matter if this is command-line or graphical, as long as functionality/error checking is provided.

You are required to provide error checking and show appropriate messages in the case of erroneous inputs – eg bus stop doesn't exist, wrong format for time for bus stop (eg letters instead of numbers), no route possible etc.

Deliverables and submission:

Please submit all in a single .zip file to blackboard only. Each group member should submit the exact same file through their blackboard profile:

Please note maximum file size for the whole submission is 100mb, so you'll need to lower the quality of your demo video if it defaults to more than this. Please do not submit input files as they are near enough to 100mb just by themselves.

1. All .java files.

2. 2-page design document

A document explaining the design decisions (a choice of data structures and algorithms used to implement each of the 3 main features), justifying them based on specific space/time trade-offs between alternatives you have considered.

3. 5-minute demo recording

A recording demonstrating the functionality of your system, illustrating the requirements from all 3 specifications parts, accounting for multiple type of cases (eg no path, no stop id, multiple stop ids, etc). Maximum duration should be 5 minutes.

4. Readme.txt

Readme.txt file should contain the following information:

- Link to repository with all the code and iterations of the document (same as in the group registration email you send to me)
- Group members list and summary of contributions of each member in the team

Marking schema:

All students in the group will receive the same mark.

- 20 points for implementation of each of the 3 main parts (10 for basic implementation, 10 for accounting for all edge cases) – 60 points total
- 10 points for the interface enabling selection of functionality
- 10 points for the demo
- 20 points for the analysis in the design document