55-500213 Algorithms and Data Structures II Reassignment

Task 2: ADS Project: Social Network

Due in	Thursday, 8st July 2021, by 15:00
Submission Method	Online Submission via Blackboard
Marks available	100 (60% of the module assessment)

In-module retrieval is NOT available for this assessment.

1. Introduction

In this task, you will design and develop one of the most important functions of a social network called "friends-you-may-know".

"Friends-you-may-know" provides a group of recommended friends for users based on the current friends they have. The recommended friends are not directly linked to the users, but their potential relationship can be reviewed and measured through their common friends.

In this task, the social network is represented by a weighted undirected graph data structure. The nodes represent users and weighted edges represent the measurement of their friendship, i.e. lower weights mean closer friendships. To find those potential friends, you need to develop a suitable shortest path algorithm to go through the entire network and find the nearest but not directed linked users. The algorithm may consume a large number of resources. It is essential to maximise the performance of the chosen algorithms and data structures.

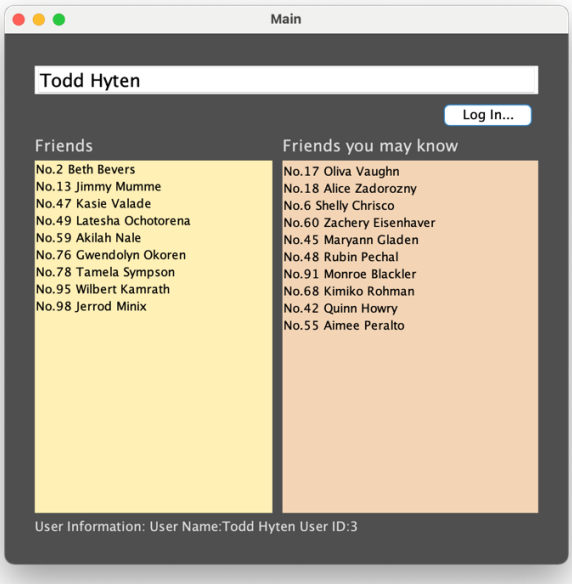

Figure 1. GUI of Social Network Application

As shown in Figure 1, an IntelliJ Project with JFrame GUI is provided for this assignment. In the skeleton structure, Main Class, GUI and their functions for user interactions, such as the buttons, text input and list, have been fully developed as placeholders for your further development.

The work you need to carry out is to implement suitable algorithms and data structures to store, manage and process the network data, and provide recommended friends from the available data. Once the work is done, after typing an existing username and click "Login", the left window will show all the friends and right window will list some recommended friends.

The following learning outcomes are assessed in this task:

- characterise algorithms and describe their attributes using appropriate metrics;
- compare and contrast similar algorithms and describe their relative merits and demerits;
- design novel algorithms;
- optimise implementations of algorithms;
- construct and manipulate the complex data structures required of many algorithms.

2. Getting Started

The Java files in the project provide the specification for each class and methods that you will need to implement. You will need to download the skeleton from Blackboard at "**Reassessment**" → "**Reassessment -002 Project** " → "**SocialNetwork_ADS2.zip**".

This social network has the following place holders and skeleton functions to be designed and developed by you:

- **Loading social network data from files**
  First of all, user name list and social network data should be loaded for the rest of the tasks. Sample dataset contains 100 users. The dataset contains two separate files for user names (`NameList.csv`) and the network distributions (`SocialNetworkData.csv`). In the task, you need to design and develop suitable and effective containers for the datasets. Also, those structures are going to be used for the rest of the task so you need to think twice, and make some wise choices before the implementation.
- **Locating a user from the network**
  Get the user name from `userName.getText()` (Line 25, Main.java) and locate the user from the network. You need to develop searching algorithms for the list and/or graph that returns the location of the nodes.
- **Listing all the friends belongs to the user**
  You need to retrieve all the directly linked nodes and return their full names. You need to return a String array to display them correctly in the GUI (i.e. The JList only accept the String Array format).
- **Listing the top 10 recommended friends for the user**
  You need to calculate the shortest distance between the current user and all other non-directly linked users. Pick up the top ten closest candidates and return their full names. You need to return a String array to display them correctly in the GUI (i.e. The JList only accept the String Array format).

Completing the skeleton code is an essential task. However, you are welcome to add new functions and improve the source code. You can discuss the ideas with your tutor if you have specific additions in mind.

3. Tasks

3.1.Implementation (*50%*)

You need to follow the behaviour defined in the class files and complete all the requested functions of the social network system. It is important that you should choose suitable underlying data structures to represent the top-level social network. You also need to design and develop suitable algorithms to generate, traverse and search the network. The data structures and algorithms must be wisely chosen to maximise the overall performance (memory and time consumption) of the system.

The details of these classes are given in their respective Java files. You need to develop specific members from the skeleton code in the `SocialNetwork` class. The key methods that need to be developed are:

`Load`	Loading social network data from files	*15%*
`FindUserID`	Locating a user from the network	*10%*

`GetMyFriends`	Listing all the friends belongs to the user	*10%*
`GetRecommended`	Listing the top 10 recommended friends for the user	*15%*

You may need to create additional classes, methods and helper functions (such as underlying list and graph structures) for the implementation. When you develop helper functions, you are advised to break the tasks down into manageable and testable units and use IntelliJ debug tool and/or console output to locate and correct code errors.

During the implementation, you must observe the following requirements:

1. While you can change and add extra methods and classes to help your task, the provided classes menthods (i.e. `Load()`, `FindUserID()`, `GetMyFriends()`, `GetRecommended()`) must not be removed from the project packages. Failure to preserve this minimum specification will result in a loss of marks.
2. You may use or manipulate any code already written during tutorial sessions. The code must be compatible with the skeleton structure.
3. The use of the Java Collections Classes is prohibited. You must implement all data structures and algorithms from base types. However, the use of File I/O and Strings is allowed. Please ask if you are not sure about whether you can use certain built-in Java classes.
4. You may use anything from the first assignment or module tutorials, but you must understand the code you submit in fulfilment of this assignment as you may be asked to explain it during a walkthrough.
5. This assignment is an individual piece of work. Your submission must be in the form of a working IntelliJ Project based on the skeleton project provided.

3.2.Code Profiling (*20%*)

Once your system is up and running, the next step is to maximise the overall performance. For this task, speed, i.e. execution efficiency, is the primary goal.

Alongside the code, a report must be submitted that supports the chosen data structures and algorithms with suitable profiling information. You should describe why you have chosen your underlying algorithm and data structures to achieve the requirement implementation based on critical analysis. Additionally, you need to identify hotspots in your code and consider how those areas of code can be further improved for performance gains. Thus, the following content MUST be included in the report:

- reasons and explanation of your chosen algorithms and data structurers to implement the task. You need to compare and contrast similar algorithms and data structures. Your choices should be considered and supported by profiling evidence.
- using the profiler, discuss the hotspots of your developed algorithms and how you might improve performance. You must include evidence of profiling, such as screenshots, to support your identification of hotspots. It is not necessary to code up alternative solutions for this part of the work, but you are expected to write about what changes could be made if you were to do so in the pursuit of execution speed.

A good report is based on evidence (such as using snapshots of profiling tools) and is expected to have an in-depth discussion on alternative data structures and algorithms. The report should be no more than three sides of A4 in standard Arial 12 with at least 2cm margins. Your report should be submitted as either an MS Word or PDF file. In this task, you only need to analyse the code inside `SocialNetwork` classes.

If your code does not work correctly, then you should still submit the report discussing what you attempted to do and why and compare it to alternative approaches from a more theoretical approach for partial credit.

3.3.Optimisation (*15%*)

Fifteen marks can be obtained by taking one hotspot you discussed while profiling your code and implementing the optimisation. You should submit this in a separate PDF or Word document that shows the code before and after along with evidence of the optimisation. **Do not fold this report into the main code profiling report.**

3.4.Additional Features (*15%*)

Another fifteen marks are for you to do something creative above the given specification. For example, you can change the data structure to allow more functions, such as "block friends" or "send friend request". You are encouraged to use untaught and more advanced algorithms and data structures for the graph and name list. The level of extra work required to obtain full marks for this section should be in proportion to the rest of the assignment. If you need to explain what extra you have created, you should submit this in a separate PDF or Word document (**do not fold it into other reports- keep it separate to make it very clear what you have done**).

4. Marking Scheme

Your assignment will be marked by using grade based approach (see appendix):

Implementation (50%)						
Z-G	F	E	D	C	B	A
Use incorrect data structures and algorithms	The requested algorithms and data structures are partly developed	The underlying data structures are correctly implemented.	Plus, the underlying data structures are well-developed to suit the functionality of the entire system	Plus, some attempts at effective algorithms and data structures	Plus, the algorithms and data structures are effectively implemented	The system is robust. Plus, the system is designed and developed by using suitable software development approaches and standards.
After applying the rubric, marks can be further reduced from statements and expressions, causing runtime errors.						

Code Profiling Report (20%)						
	Z-F	E	D	C	B	A
Suitability of the chosen algorithms and data structures (10%)	Reasons of chosen algorithms and data structure are not clear	Basic description of using certain algorithms and data structures	Use the correct algorithms and data structures	A detailed description of using specific algorithms and data structures	The student shows an in-depth understanding of the choices by evaualting their performances.	Plus, using the scientific evidence and research approaches to evaluating the performances.
Hotspots analysis and performance optimisation (10%)	Hotspots are not identified	Use the Java profiling tool correctly to count the samples	Plus, have found hotspots using profiling tools	Plus, discuss the hotspots of your developed algorithm	Plus, an in-depth discussion on how to improve system performance.	Using critical evaluation and analysis approaches to discuss the potential system optimisation.

Optimisation (15%)					
Z-F	E	D	C	B	A
No or minimal optimisation work	Have some evidence of optimisations. However, the code is poorly constructed without plans.	The optimisation is well-implemented based on the performance analysis of the hotspots	Plus, the problems caused by hotspots are solved. The code is running more effectively.	The entire system is running more effectively. Extra evaluation and optimisation are carried out.	plus, the extra work is based on an in-depth understanding of the algorithms and data structures.

Additional Features (15%)					
Z-F	E	D	C	B	A
Very limited system functional extension.	The student applies for some basic extensions under the scale of the developed classes and interface.	Some improvements for the underlying algorithm and data structures that allow the system to handle more complicated situations.	The student shows their skills of implementing some more advanced algorithms and data structures for additional features.	Plus, the implemented new features are reliable. The system is also effective and robust.	Plus, the new features contains cutting-edge techniques from computer science studies.

Your assignment will be inspected by checking their functional correctness and method of implementation. If there are aspects of the implementations that you are unable to code, then you should leave the stub function in place and comment out the code that is causing issues to ensure the IntelliJ project compiles. The commented code will be reviewed for a portion of the marks.

You may be asked to give a walkthrough of your code before a mark is awarded. During the walkthrough, you will be asked to demonstrate your understanding of the code and what it does. You will be notified by email if you need to have a walkthrough.

5. Submission Process

Your assignment should be submitted electronically through the module's Blackboard site as a single ZIP file that contains all your source code and reports.

Check your upload to ensure you have submitted the correct files successfully as issues will not be considered after the deadline.

Appendix: GRADE-BASED ASSESSMENT

What is it?

Marking is often subjective. In evaluating code style or the functionality of a piece of work there is no single right answer and no single wrong answer. Rather, a piece of work sites on a continuum from perfect to awful. Most academics grading work know this and know that the difference between 7/10 and 6/10 is to some degree a matter of subjective judgement.

Grade-based assessment breaks the link between the quality of each aspect of an assignment submission and a simple number. Instead work is allocated a letter grade that is later converted programmatically into a number using a spreadsheet.

How it works

Each assessment task is marked against a number of criteria that may cover aspects of the work such as coding style, presentation or software architecture. Each criterion is allocated a number of marks, so, for example, 15 marks may be available for the software architecture. Rather than assume that there is an objective difference between an architecture that is worth 11/15 and one that is worth 12/15, grade-based assessment allocates a broad letter grade to the architecture. This grade is converted to a mark through a simple arithmetic substitution. The overall mark for the assignment is the sum of the marks given to each of the criteria.

The system has grades A down to F which broadly mirror the classifications given to SHU degrees. An F grade represents a borderline fail. The highest marks (aka “First”) are split in two with the B grade covering the range 71 to 80 and the A grade covering 81 to 100.

Each grade can be modified by a +/- (eg A-, C+). The grade A+ implies that the work is as near perfect as could be expected from an undergraduate and is given only very rarely.

Grade Descriptors

The grade descriptions in the following table are generic and indicative; the university has provided grade descriptors for each level of study. These are available on Blackboard and should be used alongside this document.

Grade	Descriptor
A FIRST (Excellent)	● Exceptional work of the highest quality, demonstrating excellent knowledge and understanding, analysis, organisation, accuracy, relevance, presentation and appropriate skills. ● Where relevant the work may achieve or be close to publishable standard. ● It is difficult to suggest ways that the work could be improved/extended without introducing concepts from higher levels of study. ● There is extensive evidence of the independent investigation, learning and thought.
B FIRST (Excellent)	● Very high quality work demonstrating excellent knowledge and understanding, analysis, organisation, accuracy, relevance, presentation and appropriate skills. ● Work which may extend existing debates or interpretations. ● The work as presented is difficult to fault, but there are some obvious areas where it could be extended/improved. ● There is ample evidence of the independent investigation, learning and thought.
C UPPER SECOND (Very good)	● High quality work demonstrating good knowledge and understanding, analysis, organisation, accuracy, relevance, presentation and appropriate skills. ● The work as presented has some very minor errors, and there are some obvious areas where it could be extended/improved. ● There is some limited evidence of independent investigation and learning.
D LOWER SECOND (Good)	● Competent work, demonstrating reasonable knowledge and understanding. ● Some analysis, organisation, accuracy, relevance, presentation and appropriate skills are shown. ● The work as presented has some minor errors, but is limited in scope. ● Does not significantly extend work previously presented.
E THIRD (Sufficient)	● Work of limited quality. ● Demonstrates some relevant knowledge and understanding.
F FAIL (Insufficient)	● Work does not meet the standards required for this stage of an Honours degree. ● There may be evidence of some basic understanding of relevant concepts and techniques but this does not meet the level that was taught in class.
G FAIL (Insufficient)	● Some attempt has been made to tackle the assessment. ● The work has little or no merit.
Z	● Work of no merit or work was not submitted.