University of South Australia

INFS 2042 Data Structures Advanced

Assignment 2 – Contact Tracing

UniSA STEM
The University of South Australia
2024

Originally written by Brandon Matthews
Modified by Daniel Ablett, and Gun Lee

1. Introduction

To track and reduce the spread of a disease during an epidemic or pandemic situation it is critical that authorities and health experts can trace who has been in contact with whom, when the contact occurred and where. This is known as **contact tracing**. Efficiently searching potentially millions of people and where they have been will require an efficient way to store and navigate through the data.

In this assignment, you are tasked with building a basic contact tracing system. You must use your knowledge of data structures and search algorithms to efficiently store and process large quantities of contact tracing data. You are not restricted to the data structures and algorithms explored in this course. You may also make use of structures and algorithms from the Data Structures Essentials course.

2. Requirements

Your client has provided you with a strict set of system requirements that the program must meet. How you achieve those requirements and which algorithms or data structures you use are up to you. You must implement the program in Java using OpenJDK 11 or newer. You should also aim to make the program as efficient as possible. For example, exhaustively searching lists in nested loops would not be the most efficient implementation in many cases.

Generally, it is easier to design with optimisation in mind. When using the following data structures: Binary Search Tree, Self-Balancing Search Tree, Graph, Skip List, Blockchain, Hash Map, Hash Set etc. **you must implement the data structure yourself**. It is expected that a selection of these structures will be required to meet the client requirements as efficiently as possible.

You may use provided data structures in Java libraries (such as Linked List, Queue, Stack etc.) only if they are **not** a part of the content covered in this course to support the implementation of other structures and store data where necessary. Be wary of functions that are built into provided data structures, if you do use them ensure you consider their performance impact.

You are also required to provide supporting documentation, in this, you must explain each data structure you used, what they were used for and why. This **includes** cases where you have used Java's built-in data structures. Consider your implementation in the context of a real contact tracing application. The data provided for this assignment, as described below, is for 40 people, with 80 visits to 6 locations. In a real application we likely have millions of people, with tens or hundreds of millions of visits to hundreds of thousands of locations. Your implementation should be efficient for storage and processing of large amounts of data.

Remember, it is not enough that your system implements the requirements, it must implement them efficiently.

2.1 System Requirements

Below are a set of requirements for the operation of the program as provided by your client.

- The system administrator would like the ability to load existing data from the provided .csv files. The code to read the files is already provided by the client however they have not implemented a method to store the data.
- In addition, public health officials need the ability to add a new Person, Location or Visit to the data. The client has provided the input command parsing code to support this however they have not implemented the functionality.
- Public health officials need the ability to search for a Person by name. This should show them all details about the person. This includes listing all visits the Person has made.
    - Hint: This would require an efficient means of searching for the Person and all Visits in which the Person has visited any Location.
    - If a startDate and endDate are provided, this should also filter the list of Visits to only include those between these times.
- Public health officials need the ability to search for a Location by name. This should show them all details about the location. This includes listing all people that have visited the location.
    - If a startDate and endDate are provided, this should also filter the list of Visits to only include those between these times.
- The public health officials would like the ability to produce a list of potential contacts up to (n) levels away from a given person (including known contacts).
    - If n = 1, the list will contain only direct contacts of the given person.
    - If n = 2, the list will contain all direct contacts (n=1) of the given person and all contacts of those contacts (n=2).
    - If n=r, the list will contain all n=1 to r=r-1 contacts of the given person and all contacts of those contacts (n=r).
    - Hint: This would require an efficient method of identifying contacts of a given person based on their visits.
- Public health officials also need the ability to specify if the person is a new Active Case (i.e., they have become infected with the virus).
    - When an Active Case is added, they also need to see an estimation of where, when and from whom the person likely contracted the virus. Your program should output the most likely contact source including the location and time of contact. **Note**: The most likely contact source is the pair of people with the highest Chance of Spread (C) as defined later in this document.
    - If a new Active Case has no immediate contacts that are also an Active Case, the program should instead find the nearest or most likely Active Case. That is, the existing Active Case for which each contact between them and the new Active Case have the highest total Chance of Spread (C).
    - Hint: This would require a method for identifying the person from which the visit during which the person most likely contracted the virus.
- The public health officials would like to output a trace of the transmission of the virus from its original source to a target person. In this process this trace should ensure the date each person along the path was infected is correct by verifying the start date of their infection is the **day after** the contact with the highest Chance of Spread (C). In a 'real world' data set this would be useful for identifying different branches of the virus as it spreads and tracing the virus back to its original source.
    - Hint: this would require a method for tracing the path through each person backwards from the given person until no previous source case can be found (in the provided data).
- The public health officials would like to be able to produce a list of all active cases.
- The program must be robust and user friendly, so it does not crash but print proper messages.

2.2 Supporting Documentation

You must provide a document to support your program design and demonstrate your program meets the requirements. This must include:

- One-page summary of your program design and the reasoning behind your design decision.
  Explain all data structures and algorithms you used, what they were used for, and your reasoning for selecting them. (e.g., estimate of overall performance, space and time-efficiency)

- Sample outputs from your program. (no page limit)
  This is to demonstrate that your program meets the requirements. Provide headings to clarify what requirement does the provided sample output demonstrates.

3. Data and Code

For simplicity, a limited data set is provided. A person is only considered infectious if they are currently an active case and only the dates between which they are infections is recorded. All of the data is artificial data that has been procedurally generated. A person is only considered an active case if they have an activeStartDate, and they either don't have an activeEndDate or the activeEndDate is after the "current date".

3.1 Provided Data Format

The data in the provided CSV files are structured as follows:

- Person.csv – A list of people where each person has:
  - name
    - The person's full name, or the purposes of this assignment you can assume the person's full name is unique within the data set
  - activeStartDate
    - The date after the person is estimated to have contracted the virus the virus (empty if they have not contracted the virus)
  - activeEndDate
    - The date the person stopped being contagious (or is estimated to stop if after the "current date")
- Location – A list of locations where each location has:
  - name
    - The location's name, or the purposes of this assignment you can assume the person's full name is unique within the data set
- Visit – A list of visits by a person to a location where each visit has:
  - personName
    - Name of the person that visited the location
  - locationName
    - Name of the location the person visited
  - date
    - Date of the visit
  - entryTime
    - Time the person entered the location
  - exitTime
    - Time the person exited the location

3.2 Provided Code

The client has provided the basic interface commands they wish to use to handle the data. You are free to add commands for your testing purposes if you wish, however you must keep the commands listed here the same. The provided base code handles the parsing of these commands and provides some supporting types and functions. It is recommended that you retain the command functionality and build upon it, however **you are free to modify the base code however you want/need to meet the requirements.** See testfiles/test.txt in the provided code for a set of example commands.

The program is configured with an artificial "CURRENT_DATE" variable that relates to the provided data files. You should use whenever referring to the current date. This is configured by an initialization command in the test files.

For simplicity, a limited data set is provided. A person is only considered infectious if they are currently an active case and only the dates between which they are infections is recorded. All of the data is artificial data that has been procedurally generated. A person is only considered an active case if they have an activeStartDate, and they either don't have an activeEndDate or the activeEndDate is after the "current date".

Command	Purpose	Parameters
INIT	Initializes the program and sets the artificial CURRENT_DATE	currentDate – the artificial current date for the program
LOAD_DATA	Loads data from the provided People, Locations and Visits CSV files	peoplePath – path to people csv locationPath – path to location csv visitPath – path to visit csv
ADD_PERSON	Adds a new person	personName – name of the person to add
ADD_LOCATION	Adds a new location	locationName – name of the location to add
ADD_VISIT	Adds a new visit	personName – name of the person locationName – name of the location date – date of the visit entryTime – time visit started exitTime – time visit ended
GET_PERSON	Finds the Person by name and lists all visits (or a filtered list of visits between startDate and endDate)	personName – name of the person to get startDate (optional) – filter visit list by this start time endDate (optional) – filter visit list by this end time
GET_LOCATION	Finds the Location by name and lists all visits (or a filtered list of visits between startDate and endDate)	locationName – name of the location to get startDate (optional) – filter visit list by this start time endDate (optional) – filter visit list by this end time
LIST_CONTACTS	Finds the Person by name and lists all contacts within (n) contacts of the given person. i.e. n=1 is direct contact, n=2 is contact with an n=1 contact … n=N is contact with an n=N-1 contact.	personName – person to get contacts of n – number of levels of contact
CURRENTLY_ACTIVE	Lists all currently active people.	

NEW_CASE	Sets the given Person to now be an active case and the date and time they tested positive. Also outputs the most likely infection source for the target and updates the activeStartDate for the person (1 day after the contact took place), if no viable contact is detected, sets the activeStartDate to the CURRENT_DATE variable in DateHelpers.	personName – name of the person to make a new case
TRACE_PATH	Traces the path that the virus travelled from person to person until it reaches the target.	personName – name of the person to trace the virus transmission for

3.3 Calculating the Chance of Spread

For this assignment, we have an imaginary virus that has a high chance of spreading and becomes detectable and contagious the **following day**. That is. if John is detected as an active case on 5/1/2021, they must have caught the virus some day **before** 5/1/2021

For this virus the chance of contact between an active case and another person resulting in a spread to that person is based on the overlap in time spent by two people at a given location, the time since the active case contracted the virus and the incubation time. The chance is the percentage of one hour spent in contact (in the same location).

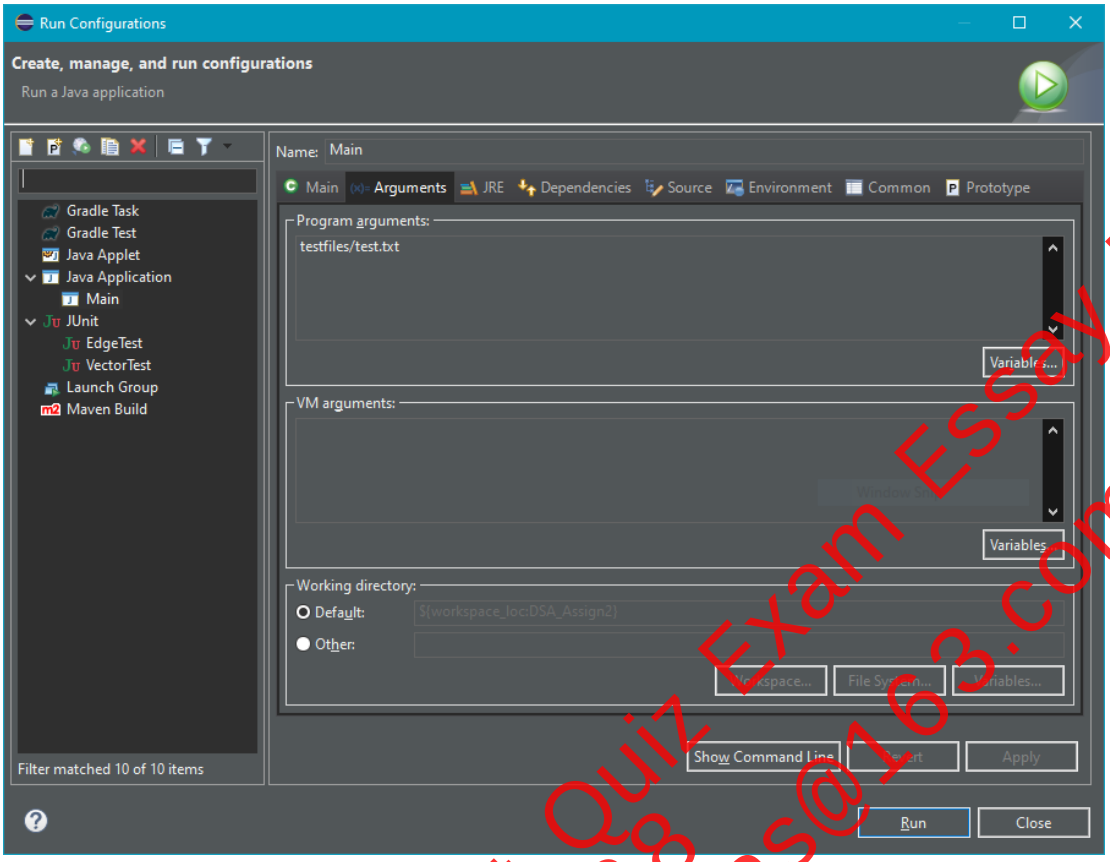
Let D be the time spent by two people in the same location (in minutes)
The Chance of Spread (C) is:

$$C = \left(\frac{D}{60} \times 100\right)$$

Note that C cannot be less than 0% or greater than 100%.

3.4 Running the Provided Code

To run the provided code you will need to pass it the path to the test file as a program argument through the "Run Configuration" in eclipse. The default included test file is at the relative location "testfiles/test.txt". Throughout development it may help to create your own test files and data sets that you can use to help with implementation of specific functions. If you are using your own test file, make sure you update the "Arguments" under the "Run Configuration" in eclipse. Note that a different test file may be used for marking.



4. Submission Details

All assignments must be submitted via the learnonline submission system through the course webpage. You must submit **two files**, a report and a zip file containing your programming solution.

Report
Submit a single pdf file with the name **emailID.pdf** (replace **emailID** with your own!) as outlined in section 2.2

Code
Make sure you add your name and email ID into the comments on all of the Java file you have modified. Create a single zip file with the name **emailID.zip**. When unzipped it should contain a functional eclipse project. You may work on the project in another IDE however you must ensure it works as an eclipse project as it will be marked based on eclipse.

Late Submissions and Extensions

Late submissions will be penalised by scaling the marks by 70% unless with pre-approved extension. Application for extension should be lodged through the course website and it requires a document proving acceptable reasons for extension (e.g., medical certificate, or a letter from your work supervisor). Please check the course outline available on the course website for further details on late submission and extensions.

Academic Misconduct

Students must be aware of the academic misconduct guidelines available from the University of South Australia website. Deliberate academic misconduct such as plagiarism is subject to penalties. Information about Academic integrity can be found in Section 9 of the Assessment policies and procedures manual at:
https://i.unisa.edu.au/policies-and-procedures/codes/assessment-policies/

All of the assignments are compared using special tools designed to look for similarities between Java programs. The plagiarism checking programs do not just compare the actual Java code, but instead perform comparisons on the code after it has been compiled. Performing cosmetic changes such as reformatting code, renaming variables, or reordering code may fool a human under casual inspection but will still be detected by the plagiarism checker as being similar. Beware that if you use generative AI tools, this may result in another person using the same tool submitting solutions with high similarities.

Any assignments found to be in violation of the university's rules on academic misconduct will become subject of Academic Integrity investigation which will decide the penalty. Furthermore, you may also fail the course and/or receive an official note in your academic transcript.

The best way to avoid being penalised for plagiarism is to not cheat on your assignment. Do not share your code with anyone, do not let anyone else do your assignment for you, and do not leave your computer unattended or share your password with anyone. If you are working with friends it is ok to discuss the assignment and possible ways of solving it, but you should not share your code. Sharing code with others is still considered academic misconduct. The golden rule for working on your assignments is never show another student the material you intend on handing up.

5. Assessment Criteria

- Program Code (70%)
  - Runs without any errors or crashing (5%)
  - Meets Client Requirements (40%)
  - Quality of Implementation and Code (25%)
    - Is the code well-structured and logical?
    - Is the code readable?
    - Have efficient control-flow elements been used (while loops, for loops)?
    - Sufficient comments. Also, name and email ID must be added in comments.
    - Code reusability
- Report (30%)
  - Logical reasoning for Data Structure and Algorithm selection is documented in supporting documentation. (25%)
    - Choices of data structure and algorithm
    - Efficient usage of space
    - Efficient computation time
  - Provided sample outputs are relevant to the requirements. (5%)