Curtin College in association with Curtin University

DATA STRUCTURES AND ALGORITHMS ASSIGNMENT TRIMESTER-Two 2020

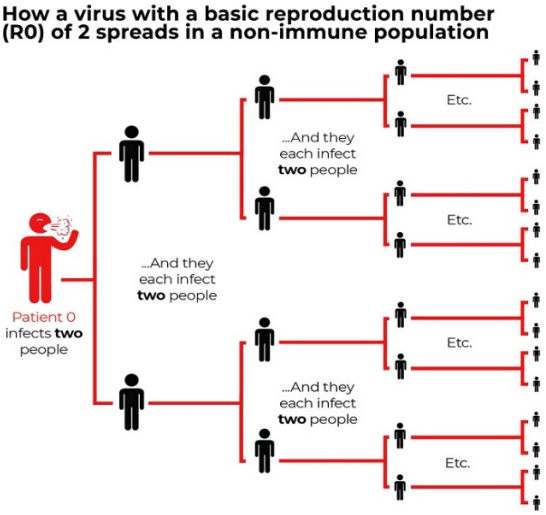
DUE: Sunday September 6, 23:59 UTC+8

BACKGROUND

In practicals you have implemented and learned about a number of algorithms and ADTs and will be implementing more of these in the remaining practicals. In this assignment, you will be making use of this knowledge to implement a system to explore and compare a variety of ADT implementations. Feel free to re-use the generic ADTs from your practicals. However, remember to self-cite; if you submit work that you have already submitted for a previous assessment (in this unit or any other) you have to specifically state this.

THE PROBLEM 60%

This assignment requires the extension of your graph code to apply it to social network analysis. You will be simulating the spread of disease through a population. In the network, at each timestep, there will be a probability of each connection infecting those they come in contact with. On the next timestep, the newly infected will have the potential of infecting their neighbours, and so on. We will have different categories of connections, which will affect the transmission rate. There will also be different interventions possible, reducing or eliminating some categories of transmission.



R0 can be viewed as an intrinsic property of the virus, whereas the Reff takes into account the effect of implemented control measures (The Conversation)

Image Source: https://www.abc.net.au/news/2020-04-30/coronavirus-spread-early-reaction-result-countries/12200256

Your program should be called **CoronaVirusSim** and the simulation should be designed similar to the example image shown above. You do not have to make the exactly same graph but the image shown above can be a good reference to visualise the problem. Your program should have three starting options:

- No command line arguments : provides usage information
- "-i" : interactive testing environment
- "-s" : simulation mode

(usage: `CoronaVirusSim –s netfile trans_rate recov_rate death_rate int_code`)

When the program starts in interactive mode, it should show the following main menu:

(1) Load network
(2) Set rates/interventions
(3) Node operations (find, insert, delete)
(4) Edge operations (like/follow - add, remove)
(5) New infection
(6) Display network
(7) Display statistics
(8) Update (run a timestep)
(9) Save network

You can structure the menu/UI differently, just make sure at least those options are included.

When running in simulation mode, you will give the input files and parameters on the command line, then output (append) the simulation state and statistics for each “timestep” to a file (unique filename per run).

In interactive mode you should be able to display the following statistics/information:

- Overall statistics for the population
- List of names in each health category (e.g. Susceptible, Infected, Recovered)
- A person’s record (e.g. age, health status, date/timestep infected/recovered)

Coding should be done in your preferred language chosen for the DSA unit- Python or Java.

Once you have a working program, you will investigate the impact of the various parameters and interventions. This investigation will be written up as The Report.

Remember: think before you code!

USER GUIDE AND REPORT (35%)

You need to submit your User Documentation and Report in doc/docx, pdf or Jupyter Notebook format.
Your **User Documentation** will be minimum 4-6 pages, with a section for each simulation and should include the following:

- An **overview** of each of your program's purposes and features.
- A **guide** on how to use your programs/simulations.
- A **discussion** of your code, explaining how it works, any additional features and how you implemented them.

The **Corona Virus spread report** will comprise of a mini-paper that is 2-3 pages long and follow the structure of a standard academic report. Required sections are:

- **Abstract:** Explain the purpose of the simulation and state the parameters you have considered in designing the simulation, and the outcomes/recommendations.
- **Background:** Discuss the purpose of the program/simulation and your choice of parameters.
- **Methodology:** Describe how you have chosen to set up the simulation, and why. Include commands, input files, and outputs – anything needed to reproduce your results.
- **Results:** Present the results of your simulations – include tables, plots (optional) and discussion.
- **Conclusion and Future Work:** Give conclusions and what further investigations could follow.
- **References**

(You can find examples on google but do not copy and paste. Just use the examples as reference)

CODING STANDARDS (5%)

Your code submission must conform to coding standards emphasised in the lecture and practicals. Remember: consistency is key!

SUBMISSION

Submit electronically via Moodle. Make sure to submit early. You can submit multiple times – we will only mark the last attempt. Take care not to submit your last version late though. Read the submission instructions very carefully.

You should submit a single file, which should be zipped (.zip). The file must be named DSA_Assignment_<id> where the <id> is replaced by your student id ignoring the angle brackets. There should be no spaces in the file name; use underscores as shown.

The file must contain the following:

- Your **code**. This means all files needed to run your program. That includes input files used as part of the assignment if that is required to run your program.
- **README** file including short descriptions of all files and dependencies, and information on how to run the programs.
- **User Documentation**, as described above.
- Your **unit test harnesses**. One of the easiest ways for us to be sure that your code works is to make sure that you've tested it properly. Make it easy for us to test your work - the test harness for class X should be called UnitTestX, or can be included in the class file for Python/Java.
- A signed and dated **cover sheet**. These are available on Moodle You can sign a hard copy and scan it in or you can fill in a soft copy and digitally sign it.

Make sure that your zip file contains what is required. Anything not included in your submission may not be marked, even if you attempt to provide it later. It is your responsibility to make sure that your submission is complete and correct.

REQUIREMENTS FOR PASSING THE UNIT

Please note: As specified in the unit outline, it is necessary to have attempted the assignment in order to pass the unit. Section 2.5 of Unit outline explains the weightage associated with the assignment which is 30% of overall marks. As a guide you should be able to achieve 15% out of 30% weightage of the assignment. Note that the marks indicated in this section represent maximums, achieved only if you completely satisfy the requirements of the relevant section.

Plagiarism is a serious offence. This assignment has many correct solutions so plagiarism will be easy for us to detect (and we will). For information about plagiarism, please refer to http://academicintegrity.curtin.edu.au.

In the case of doubt, you may be asked to explain your code and the reason for choices that you have made as part of coding to the unit coordinator. A failure to adequately display knowledge required to have produced the code will most likely result in being formally accused of cheating.

Finally, be sure to secure your code. If someone else gets access to your code for any reason (including because you left it on a lab machine, lost a USB drive containing the code or put it on a public repository) you will be held partially responsible for any plagiarism that results.

LATE SUBMISSION

As specified in the unit outline, you must submit the assignment on the due date. Acceptance of late submissions is not automatic and will require supporting documentation proving that the late submission was due to unexpected factors outside your control. See

the unit outline for details as to the procedure for requesting that an assessment be accepted after the due date.

Also note that IT related issues are almost never a valid excuse.

In the event that you submit your assignment late and are deemed to have a valid excuse, you will be penalised 10% (that is, 10% out of 100%, not out of what you would have received) per calendar day that you are late, up to a maximum of seven (7) calendar days. Any work submitted after this time will not be marked and you will automatically fail the unit. Note that if you are granted an extension you will be able to submit your work up to the extended time without penalty – this is different from submitting late.

CLARIFICATIONS AND AMENDMENTS

This assignment specification may be clarified and/or amended at any time. Such clarifications and amendments will be announced on the unit's Blackboard page. These clarifications and amendments form part of the assignment specification and may include things that affect mark allocations or specific tasks. It is your responsibility to be aware of these by monitoring the Assignment Assessment Blackboard page.