COMP4500/7500
Advanced Algorithms and Data Structures

School of Information Technology and Electrical Engineering
The University of Queensland, Semester 2, 2020

Assignment 1

Due at 4:00pm, Monday the 5th October 2020.
This assignment is worth 20%, for both (COMP4500) and (COMP7500), of your final grade.

This assignment is to be attempted **individually**. It aims to test your understanding of graphs and graph algorithms. Please read this entire handout before attempting any of the questions.

Submission. Answers to each of the questions in part A and Question 4(a) and 4(b) from part B should be clearly labelled and included in a pdf file called `a1.pdf`.

You need to submit (i) your written answers to parts A and Question 4(a) and 4(b) from part B in `a1.pdf`, as well as (ii) your source code file `TransmissionFinder.java` electronically using Blackboard according to the exact instructions on the Blackboard website: `https://learn.uq.edu.au/`

You can submit your assignment multiple times before the assignment deadline but only the last submission will be saved by the system and m suring that you have submitted the file m. You will be marked on the files that you su that are submitted according to

Submitted work should be neat, legible and simple to understand – you may untidy or difficult to read and comprehend.

For the programming part, you will be penalised for submitting files that are not compatible with the assignment requirements. In particular, code that is submitted with compilation errors, or is not compatible with the supplied testing framework will receive 0 marks.

COVID-19 temporary changes to assessment extension procedures. If you are requesting an extension to assessment based on medical grounds, we don't require a medical certificate, we'll accept a statement of circumstances signed by you. We'll accept statements of circumstance until 1 November 2020.

Statement of Circumstance: Please ensure your statement includes the following details:

- Name
- Student Number
- Details of the circumstances including dates of incapacitation
- Is dated and signed

Late submission. Unless you have been approved to submit an assignment after the due date: late assignments will lose 10% of the marks allocated to the assignment immediately, and a further 10% of the marks allocated to the assignment for each additional calendar day late. Assignments more than 5 calendar days late will not be accepted.

If there are medical or exceptional circumstances that will affect your ability to complete an assignment by the due date, then you can apply for an extension as per Section 5.3 of the electronic course profile (ECP). Requests must be made at least 48 hours prior to the submission deadline. Assignment extensions longer than 7 calendar days will not be granted.

School Policy on Student Misconduct. You are required to read and understand the School Statement on Misconduct, available at the School's website at:

`http://www.itee.uq.edu.au/itee-student-misconduct-including-plagiarism`

This is an individual assignment. If you are found guilty of misconduct (plagiarism or collusion) then penalties will be applied.

Part A (30 marks total)

Question 1: Constructing SNI and directed graph [5 marks total]

(a) (1 mark) **Creating your SNI.** In this assignment you are required to use your student number to generate input.

Take your student num [illegible]
digit initial input numbe [illegible] 12] (so that $d[1] = 9, d[2] = 8, \ldots, d[12] = 2$).

Apply the following algorithm to these twelve digits:

```
1  for $i$ = 2 to 12
2      if $d[i] == d[i-1]$
3          $d[i] = (d[i] + 3) \mod 10$
```

After applying this algorithm, the resulting value of d forms your 12-digit SNI. Write down your initial number and your resulting SNI.

(b) (4 marks) Construct a graph S with nodes **all** the digits $0, 1, \ldots, 9$. If 2 digits are adjacent in your SNI then connect the left digit to the right digit by a directed edge. For example, if "15" appears in your SNI, then draw a directed edge from 1 to 5. Ignore any duplicate edges. Draw a diagram of the resulting graph. (You may wish to place the nodes so that the diagram is nice, e.g., no or few crossing edges.)

Question 2: Strongly connected components [25 marks total]

Given a directed graph $G = (V, E)$, a subset of vertices U (i.e., $U \subseteq V$) is a *strongly connected component* of G if, for all $u, v \in U$ such that $v \neq u$,

a) u and v are mutually reachable, and

b) there does not exist a set $W \subseteq V$ such that $U \subset W$ and all distinct elements of W are mutually reachable.

For any vertices $v, u \in V$, v and u are mutually reachable if there is both a path from u to v in G and a path from v to u in G.

The problem of finding the strongly connected components of a directed graph can be solved by utilising the depth-first-search algorithm. The following algorithm SCC(G) makes use of the basic depth-first-search algorithm given in lecture notes and the textbook, and the transpose of a graph; recall that the transpose of a graph $G = (V, E)$ is the graph $G^T = (V, E^T)$, where $E^T = \{(u, v) \mid (v, u) \in E\}$ (see Revision Exercises 3: Question 6). (For those who are interested, the text provides a rigorous explanation of why this algorithm works.)

```
SCC(G)
1   call DFS($G$) to compute finishing times $u.f$ for each vertex $u$
2   compute $G^T$, the transpose of $G$
3   call DFS($G^T$), but in the main loop of DFS, consider the vertices in order of decreasing $u.f$
4   output the vertices of each tree in the depth-first forest of step 3 as a separate
    strongly connected component
```

(a) (12 marks) Perform step 1 of the SCC algorithm using S as input. Do a depth first search of S (from Question 1A), showing colour and immediate parent of each node at each stage of the search as in Fig. 22.4 of the textbook and the week 3 lecture notes. Also show the start and finish times for each vertex.

For this question you sho

for each vertex

for each vertex $v \in G.Adj[u]$.

(b) (3 marks) Perform step 2 of the SCC algorithm and dra

(c) (10 marks) Perform steps 3, 4 of the SCC algorithm. List the trees in the depth-first forest in the order in which they were constructed. (You do not need to show working.)

Part B (70 marks total)

[Be sure to read through to the end before starting]

There is a virus called COMP-4500 in a population. You are in charge of determining the most likely path that this virus could have been transmitted through the population between two people.

Every person within the population is assigned with a unique identifier. An 'interaction' takes place between two people, at a given time, and has a probability score describing how likely the virus is transmitted from the transmitter to the receiver. A single interaction is represented as a tuple;

```
(personFrom, personTo, timeOfInteraction, transmissionProbability)
```

Where:

- `0 < transmissionProbability` $\leq$ `1`.
- `timeOfInteraction` is a non-negative integer.

For example, the tuple $(P_1, P_2, 8, 0.8)$ represents an interaction that took place between persons P_1 and P_2, at time 8, which has a probability of transmitting COMP-4500 from P_1 to P_2 of 0.8 (80%).

One person within the population is known to have COMP-4500 at time 0, this person will be referred to as the `startPerson`. There is another person within the population who is confirmed case of COMP-4500, who is labelled as the `endPerson`. Additionally, there is a set of interactions that took place between members of the population (provided as a list).
A 'valid path' is a sequence of interactions such that:

- all times are non-decreasing.
- the `personTo` identifier of one interaction matches the `personFrom` identifier of the next interaction

The probability of COMP-4500 being transmitted along a path is the product of the `transmissionProbability` fields of all interactions along that path. We aim to find a valid path from the `startPerson` to the `endPerson` with the highest probability of transmission.

Example 1: For example, say you are provided with the following set of interactions;

$(P_1, P_2, 4, 0.8)$	$(P_2, P_1, 4, 0.7)$	$(P_4, P_2, 9, 0.4)$	$(P_5, P_6, 14, 0.9)$
$(P_1, P_3, 7, 0.8)$	$(P_3, P_1, 15, 0.9)$	$(P_4, P_6, 18, 0.7)$	$(P_6, P_4, 18, 0.8)$
$(P_1, P_3, 15, 0.9)$	$(P_3, P_5, 7, 0.5)$	$(P_5, P_3, 7, 0.5)$	$(P_6, P_5, 14, 0.9)$
$(P_1, P_4, 6, 0.2)$	$(P_3, P_6, 6, 0.9)$		

And `startPerson` $= P_1$, `endPerson` $= P_6$.
Then there are two valid paths from P_1 to P_6,

$\rangle\rangle$

The transmission probability f $_1$
The transmission probability for L_2 is $0.2 \cdot 0.7 = 0.14$.
L_1 is the valid path with the highest probability of transmission so the exp s
the list,

$$\langle (P_1, P_3, 7, 0.8), (P_3, P_5, 7, 0.5), (P_5, P_6, 14, 0.9)\rangle$$

Note that there are other paths from P_1 to P_6 which have higher transmission probabilities, e.g. $\langle (P_1, P_3, 15, 0.9), (P_3, P_6, 6, 0.9)\rangle$ which are not valid paths since there is a decrease in time, $6 < 15$.

Example 2: A different problem could involve the following list of interactions:

$(P_1, P_2, 14, 0.3)$	$(P_2, P_3, 10, 0.2)$	$(P_3, P_2, 10, 0.3)$	$(P_4, P_2, 3, 0.9)$
$(P_1, P_3, 8, 0.9)$	$(P_2, P_4, 3, 0.9)$	$(P_3, P_4, 7, 0.8)$	$(P_4, P_3, 7, 0.7)$
$(P_2, P_1, 14, 0.4)$	$(P_3, P_1, 8, 0.9)$		

With `startPerson` $= P_1$, `endPerson` $= P_4$.
With this set of interactions there are no valid paths from P_1 to P_4.

<u>Your task</u> is to design, implement and analyse an algorithm that takes as input:

- A list of interactions that does not contain any null elements (non-null list).
- A `startPerson` identifier.
- An `endPerson` identifier.

and, *without modifying any of the supplied interactions in any way*, it returns as output either:

(a) A non-null list of interactions such that

  (i) Each interaction is non-null and is an element of the input interaction list.
  (ii) The first interaction in the list has a `personFrom` identifier matching the `startPerson`.
  (iii) The last interaction in the list has a `personTo` identifier matching the `endPerson`.
  (iv) For all interactions in the list, the `personTo` identifier of one entry matches the `personFrom` identifier of the next entry.
  (v) The time of all interactions in the order of the list are non-decreasing.
  (vi) There are no other paths from `startPerson` to `endPerson` such that the product of their `transmissionProbability` fields is greater than the product of `transmissionProbabilities` fields in this list.

  OR

(b) The value null if no such list of transfers exists.

Question 3: Design and implement a solution (50 marks)

Design and implement an algorithm that answers the question above. Your algorithm should be as efficient as possible. Marks will be deducted for inefficient algorithms (e.g. a brute-force approach is not appropriate). Clearly structure and comment your code and use meaningful variable names.

- Your algorithm shoul nFinder.findTransmissionPath from the Transmiss ailable in the zip file that accompanies this hand

  The zip file for the assignment also includes some other code that you w TransmissionFinder as well as some junit4 test classes to hel ur code.

- Do not modify any of the files in package `assignment1` other than `TransmissionFinder`, since we will test your code using our original versions of these other files.

- You may not change the class name of the `TransmissionFinder` class or the package to which it belongs. You may not change the signature of the `TransmissionFinder.findTransmissionPath` method in any way or alter its specification. (That means that you cannot change the method name, parameter types, return types or exceptions thrown by the method.)

- You are encouraged to use Java 8 SE API classes, but no third party libraries should be used. (It is not necessary, and makes marking hard.)

- Don't write any code that is operating-system specific (e.g. by hard-coding in newline characters etc.), since we will batch test your code on a Unix machine. Your source file should be written using ASCII characters only.

- You may not write and submit additional classes. Your solution to Q3 should be self contained in the `TransmissionFinder` package. You may use nested classes.

- The JUnit4 test classes as provided in the package `assignment1.test` are not intended to be an exhaustive test for your code. Part of your task will be to expand on these tests to ensure that your code behaves as required.

Your implementation will be evaluated for correctness and efficiency by executing our own set of junit test cases. Code that is submitted with compilation errors, or is not compatible with the supplied testing framework will receive 0 marks. A Java 8 compiler will be used to compile and test the code.

Question 4: Worst-case time complexity analysis (20 marks)

This question involves performing an analysis of the worst-case time complexity of your algorithm from Question 3.

(a) (5 marks) For the purpose of the *worst-case time complexity* analysis in Q4(b), provide clear and concise pseudocode that summarizes the algorithm you used in your implementation from Question 3. You may use the programming constructs used in Revision solutions, and assume the existence of common abstract data types like sets, maps, queues, lists, graphs, as well as basic routines like sorting etc.

Clearly structure and comment your pseudocode.

[It should be no more than one page using minimum 11pt font. Longer descriptions will not be marked.]

(b) (15 marks)

Let i be the number of interactions in the supplied list (e.g. 14 in example 1); p be the number of people in the population (e.g. 6 in example 1); t be the total time elapsed (e.g. 18 in example 1).

Provide an asymptotic upper bound on the worst-case time complexity of your algorithm in terms of parameters i and p and t. Make your bound as tight as possible and justify your solution using your pseudocode from Q4(a)

You must clearly sta ns of any data structures that you

To simplify your analysis, you may make the (incorrect but simpl *HashSet* (or *HashMap*) operations that have expected-case time complexi ly have worst-case time complexity $O(1)$. E.g. checking for set-membership in a ed-case time complexity that is $O(1)$.

[Make your analysis as clear and concise as possible – it should be no more than $\frac{3}{4}$ of a page using minimum 11pt font. Longer descriptions will not be marked.]

Evaluation Criteria

Question 1

- **Question 1 (a) (1 mark)**

  1 : correct answer to question given

  0 : answer not given or contains one or more mistakes

- **Question 1 (b) (4 marks)**

  4 : The answer to question 1(a) is 100% correct, and the correct graph is given.

  0 : If the answer to question 1(a) is not 100% correct, or the answer contains at least one mistake.

Question 2

If the graph produced for Question 1 is given and 100% correct, then the following marking scheme applies. If the graph produced for Question 1 is mostly correct (but contains a minor error), then the student will receive 2/3 of the marks obtained using the following marking criteria. Else, if the graph produced for Question 1 contains more than a minor error, zero marks will be given for all aspects of this question.

- **Question 2 (a) (12 mar**

  12 : Correct answer give

  10 : All stages of the traversal shown, including relevant features f iate parent and start and finish times), but there are one or two min

  8 : All stages of the traversal are shown, however at most one of the releva vertex (e.g. start-time) may not be included and there may be up to three mistakes.

  6 : Most stages of the traversal are shown, however at most two of the relevant features for each vertex (e.g. start-time) may not be included and there may be up to four mistakes.

  3 : Most stages of the traversal are shown, however at most two of the relevant features for each vertex (e.g. start-time) may not be included and there may be up to five mistakes.

  0 : Otherwise.

- **Question 2 (b) (3 marks)**

  3 : correct answer to question given

  0 : answer not given or contains one or more mistakes

- **Question 2 (c) (10 marks)**

  This part of the question will be marked correct with respect to the finishing times for each vertex calculated in Q2(a). If those finishing times are not given in Q2(a) then no marks will be awarded for this section. Otherwise, the following marking criteria applies.

  10 : All the trees in the depth-first forest are listed in the order in which they were constructed. All aspects of the depth-first forest are correct.

  8 : All the trees in the depth-first forest are listed in the order in which they were constructed, but there was an error in the traversal that produced the forest.

5 : All of the trees in the depth-first forest are listed, however the order in which the trees were created may not be clear, or there may be one or two errors in the traversal that produced the forest.

3 : Either: (i) all of the strongly connected components of the graph are correctly listed, but the trees (from the depth-first forest) from which these connected components were derived are not clearly listed, or (ii) all of the trees in the depth-first forest are listed, however the order in which the trees were created may not be clear and there may be up to three errors in the traversal that produced the forest.

0 : Otherwise.

Question 3 (50 marks)

Your implementation will be evaluated for correctness and efficiency by executing our own set of junit test cases.

50 : All of our tests pass

45 : at least 90% of our tests pass

40 : at least 80% of our tests pass

35 : at least 70% of our tests pass

30 : at least 60% of our tests pass

25 : at least 50% of our tests pass

20 : at least 40% of our tests pass

15 : at least 30% of our tests pass

10 : at least 20% of our tests pass

5 : at least 10% of our tests pass

0 : less than 10% of our test pass or work with little or no academic merit

Note: Code that is submitted with compilation errors, or is not compatible with the supplied testing framework will receive 0 marks. A Java 8 compiler will be used to compile and test the code.

Question 4

For any marks to be received for this section, a plausible solution to Question 3 must have been presented in Q3. If a plausible solution to Q3 has been attempted, the following marking criteria applies.

- **Question 4(a) (5 marks)**

  For this question, the pseudocode given must be no longer than one page using minimum 11pt font. Longer solutions will receive 0 marks, otherwise the following marking criteria applies.

  5 : Clear and concise pseudocode that summarizes the algorithm you used in your implementation from Question 3. Clearly commented, and clear correspondence between the implementation from Q3 and the pseudocode.

3 : Mostly clear and concise pseudocode that summarizes the algorithm you used in your implementation from Question 3. Mostly clearly commented, and mostly clear correspondence between the implementation from Q3 and the pseudocode.

2 : Pseudocode given that summarizes the algorithm you used in your implementation from Question 3. Potentially more verbose than necessary. May not be commented, or have a very clear correspondence between the implementation from Q3 and the pseudocode.

1 : Pseudocode given that attempts to summarize the algorithm you used in your implementation from Question 3. Aspects are unclear, or overly verbose, or the correspondence to the actual implementation may be confusing.

0 : Work with little or no academic merit

- **Question 4(b)(i) - analysis (10 marks)**

  For this part of the question, the analysis should be no more than 3/4 of a page using minimum 11pt font. Longer solutions will receive 0 marks. Also, Q4(a) must have been answered and received a mark of at least 2. Otherwise the following marking criteria applies.

10 : A correct asymptotic upper bound on the worst-case time complexity of the algorithm from Q3 is given in terms of parameters i, p and t. The asymptotic upper bound should be as tight as reasonably possible for the algorithm at hand. The worst-case time complexity analysis is clearly justified with respect to the pseudocode from Q4(a). Any assumptions made in the analysis are reasonable and cl e asymptotic time complexity g sary constant factors.

7 : A correct asymptoti rom Q3 is given in terms of parameters i, p and t. The asy e reasonably tight for the algorithm at hand. The worst-case time complexity ana with respect to the pseudocode from Q4(a). Any assumption reasonable and clearly stated.

5 : A reasonable attempt has been made to give a reasonably-tight asymptotic upper bound on the worst-case time complexity of the algorithm from Q3 in terms of parameters i, p and t, however either it contains some minor mistakes but is otherwise reasonably justified, or the justification of the analysis with respect to the pseudocode from Q4(a) is unclear or lacking. Assumptions made in the analysis may be unclear.

3 : An attempt has been made to give an asymptotic upper bound on the worst-case time complexity of the algorithm from Q3 in terms of parameters i, p and t, however it contains either a major mistake, or many mistakes, or gives an unreasonably loose upper bound. The justification for the analysis with respect to the pseudocode from Q4(a) may be unclear or lacking.

0 : Work with little or no academic merit.

- **Question 4(b)(ii) - time complexity (5 marks)**

  Some additional marks are allocated for solutions which give favourable time complexity upper bounds. These will be marked according to the time complexity of the pseudocode supplied in Q4(a). Q4(a) must have been answered and received a mark of at least 2, and a mark of at least 3 must have been awarded for Q4(b)(i). The following marking criteria applies.

5 : The pseudocode algorithm has an optimal worst case time complexity upper bound.

3 : The pseudocode algorithm has a near optimal worst case time complexity upper bound.

2 : The pseudocode algorithm has a polynomial worst case time complexity upper bound.

0 : The pseudocode algorithm has an exponential (or worse) worst case time complexity upper bound.