COMP4500/7500
Advanced Algorithms and Data Structures

School of Information Technology and Electrical Engineering
The University of Queensland, Semester 2, 2018

Assignment 1

Due at 4:00pm, Monday the 17th September 2018.
This assignment is worth 20% (COMP4500) or 15% (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 `MinimumCostFinder.java` as well as any other source code files that you have written in the `assignment1` package 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 marked. Only submit the files listed above. You are responsible for ensuring that you have submitted the files that you intended to submit in the way that we have requested them. You will be marked on the files that you submitted and not on those that you intended to submit. Only files that are submitted according to the instructions on Blackboard will be marked.

Submitted work should be neat, legible and simple to understand – you may be penalised for work that is 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.

Late submission. Late assignments will lose 10% of the maximum mark for the assignment immediately, and a further 10% of the maximum mark for the assignment for each additional day late. Assignments more than 5 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 (35 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 number and prefix it by "98" and postfix it by "52". This will give you a twelve digit initial input number. Call the digits of that number $d[1], d[2], \ldots, d[12]$ (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 [30 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) (15 marks) Perform step 1 of the SCC algorithm using S as input. Do a depth first search of S (from Question 1b), 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 should visit vertices in numerical order in all relevant loops:

for each vertex $u \in G.V$ and

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

(b) (3 marks) Perform step 2 of the SCC algorithm and draw S^T.

(c) (12 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 (65 marks total): Finding the minimum cost

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

You are in a remote archipelago containing n island *locations* $\mathcal{P} = \{P_0, P_1, \ldots, P_{n-1}\}$. Deliveries between islands can be expensive, and so you want work out how to send a package from one island to another for the minimum cost.

The only way to get a package from one island to another is using the deliveries, $\mathcal{D}$, that are scheduled. Each *delivery* is represented by a quintuple (P_u, P_v, t_u, t_v, c) where c is the cost of the delivery, t_u is the time that the delivery departs the source location $P_u \in \mathcal{P}$, and t_v is the time that the delivery arrives at the destination location $P_v \in \mathcal{P}$. For each such delivery we have that the cost is a non-negative integer, both the departure and arrival times are non-negative integers, and the departure time of the delivery is strictly less than its arrival time at the destination (i.e. $0 \le t_u < t_v$). Island locations may simultaneously send and receive multiple deliveries.

A *route* r from one island location, P_s, to another, P_d, is a non-empty sequence of deliveries from $\mathcal{D}$:

$$\langle (P_{u0}, P_{v0}, t_{u0}, t_{v0}, c_0), (P_{u1}, P_{v1}, t_{u1}, t_{v1}, c_1), \ldots, (P_{u(x-1)}, P_{v(x-1)}, t_{u(x-1)}, t_{v(x-1)}, c_{x-1}) \rangle$$

where the number of deliveries in the sequence is x, such that the first delivery $r(0)$ departs P_s (i.e. $P_s = P_{u0}$), the last delivery $r(x-1)$ arrives at P_d (i.e. $P_d = P_{v(x-1)}$), and for each $i \in 0, \ldots, x-2$, we have that delivery $r(i)$ arrives at the same location that the next delivery, $r(i+1)$, departs from (i.e. $P_{vi} = P_{u(i+1)}$) at a time $t_{vi} \le t_{u(i+1)}$. The *cost of a route* is the sum of the costs of the deliveries in the route, i.e. $\sum_{i=0}^{x-1} c_i$.

You task is to design, implement and analyze an algorithm that takes as input:

- a set of n island locations $\mathcal{P} = \{P_0, P_1, \ldots, P_{n-1}\}$,
- a source location $P_s \in \mathcal{P}$ and destination location $P_d \in \mathcal{P}$ such that $P_s \ne P_d$,
- non-negative integer times t_s and t_d such that $0 \le t_s \le t_d$,
- a list of m deliveries $\mathcal{D}$ such that for each delivery $(P_u, P_v, t_u, t_v, c) \in \mathcal{D}$, $P_u \in \mathcal{P}$ and $P_v \in \mathcal{P}$ and $t_s \le t_u$ and $t_v \le t_d$ [the deliveries may not be ordered in any particular way – they may appear in any possible order]

and returns the minimum cost of any route (consisting only of deliveries from $\mathcal{D}$) from location P_s to P_d that departs P_s no earlier than time t_s, and arrives at P_d no later than time t_d, if at least one such route exists. If no such route exists the algorithm should return the distinguished value -1. You algorithm must be designed and implemented as efficiently as possible.

Example 1 Given inputs $\mathcal{P} = \{P_0, P_1, P_2, P_3, P_4, P_5\}$, $P_s = P_0$ and $t_s = 2$, $P_d = P_5$ and $t_d = 20$, and deliveries

$$\begin{aligned}\mathcal{D} \; = \; & (P_0, P_1, 3, 4, 10),\\ & (P_0, P_1, 5, 7, 2),\\ & (P_0, P_1, 10, 11, 1),\\ & (P_0, P_3, 8, 9, 0),\\ & (P_1, P_2, 5, 6, 2),\\ & (P_1, P_2, 7, 9, 3),\\ & (P_1, P_4, 8, 10, 1),\\ & (P_1, P_5, 12, 13, 9),\\ & (P_2, P_5, 4, 6, 1),\\ & (P_2, P_5, 11, 15, 3),\\ & (P_4, P_0, 11, 12, 2),\\ & (P_4, P_5, 9, 10, 1)\end{aligned}$$

we have that there are six possible routes from P_s to P_d that depart P_s no earlier than time t_s and arrive at P_d no later than t_d:

$$\begin{aligned}&\langle (P_0, P_1, 3, 4, 10), (P_1, P_2, 5, 6, 2), (P_2, P_5, 11, 15, 3)\rangle\\ &\langle (P_0, P_1, 3, 4, 10), (P_1, P_2, 7, 9, 3), (P_2, P_5, 11, 15, 3)\rangle\\ &\langle (P_0, P_1, 3, 4, 10), (P_1, P_5, 12, 13, 9)\rangle\\ &\langle (P_0, P_1, 5, 7, 2), (P_1, P_2, 7, 9, 3), (P_2, P_5, 11, 15, 3)\rangle\\ &\langle (P_0, P_1, 5, 7, 2), (P_1, P_5, 12, 13, 9)\rangle\\ &\langle (P_0, P_1, 10, 11, 1), (P_1, P_5, 12, 13, 9)\rangle\end{aligned}$$

with costs $10+2+3 = 15$, $10+3+3 = 16$, $10+9 = 19$, $2+3+3 = 8$, $2+9 = 11$ and $1+9 = 10$, respectively. The route with the lowest cost is therefore:

$$\langle (P_0, P_1, 5, 7, 2), (P_1, P_2, 7, 9, 3), (P_2, P_5, 11, 15, 3)\rangle$$

which has a cost of 8. Your algorithm should therefore return the value 8 for this example.

Example 2 Given inputs $\mathcal{P} = \{P_0, P_1, P_2, P_3, P_4, P_5\}$, $P_s = P_0$ and $t_s = 2$, $P_d = P_5$ and $t_d = 20$, and deliveries

$$\begin{aligned}\mathcal{D} \; = \; & (P_0, P_1, 15, 16, 1),\\ & (P_0, P_3, 8, 9, 0),\\ & (P_1, P_2, 5, 6, 2),\\ & (P_1, P_2, 7, 9, 3),\\ & (P_1, P_4, 8, 10, 1),\\ & (P_1, P_5, 12, 13, 9),\\ & (P_2, P_5, 4, 6, 1),\\ & (P_2, P_5, 11, 15, 3),\\ & (P_4, P_0, 11, 12, 2),\\ & (P_4, P_5, 9, 10, 1)\end{aligned}$$

we have that there are no possible routes from P_s to P_d that depart P_s no earlier than time t_s and arrive at P_d no later than t_d. Your algorithm should therefore return the value -1 for this example.

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. Clearly structure and comment your code.

- Your algorithm should be implemented in the static method `MinimumCostFinder.findMinimumCost` from the `MinimumCostFinder` class in the `assignment1` package that is available in the zip file that accompanies this handout.

  The zip file for the assignment also includes some other code that you will need to compile the class `MinimumCostFinder` as well as some junit4 test classes to help you get started with testing your code.
- Do not modify any of the files in package `assignment1` other than `MinimumCostFinder`, since we will test your code using our original versions of these other files.
- You may not change the class name of the `MinimumCostFinder` class or the package to which it belongs. You may not change the signature of the `MinimumCostFinder.findMinimumCost` 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.)
- Dont 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 write additional classes, but these must belong to the package `assignment1` and you must submit them as part of your solution – see the submission instructions for details.
- 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.

Hints: It may be very inefficient to solve the problem by enumerating all of the possible routes from P_s to P_d. The problem *can* be solved in polynomial time! Algorithms similar to the ones you are already familiar with from class could be adapted to solve the problem, or inspire a solution.

Question 4: Worst-case time complexity analysis (15 marks)

This question involves performing an analysis of the worst-case time complexity of your algorithm from Question 3, in terms of parameters n and m.

(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 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) (10 marks)

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

You must clearly state any assumptions that you make (e.g. on the choice of implementations of any data structures that you use, and their running time etc.).

[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

Zero marks will be given for all aspects of this question if the graph produced in Question 1 is not given or is not 100% correct. Otherwise, the following marking scheme applies.

- **Question 2 (a) (15 marks)**

  15 : Correct answer given.

  12 : All stages of the traversal shown, including relevant features for each vertex (colour, immediate parent and start and finish times), but there are one or two minor mistakes.

  9 : All stages of the traversal are shown, however at most one of the relevant features for each 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) (12 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.

  12 : Correct answer given.

  9 : All trees listed in the depth-first forest in the order in which they were constructed, but there is one error in the traversal that produced the forest.

  6 : All trees listed in the depth-first forest in the order in which they were constructed, but there are two errors in the traversal that produced the forest.

  3 : All trees listed in the depth-first forest in the order in which they were constructed, but there are three errors in the traversal that produced the forest.

0 : Not all trees listed in the depth-first forest in the order in which they were constructed, or there are more than three errors in the traversal that produced the forest.

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) (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 n and m. 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 clearly stated. Asymptotic notation should be used correctly and the asymptotic time complexity given has been simplified to remove lower order terms and unnecessary constant factors.

  7 : A correct asymptotic upper bound on the worst-case time complexity of the algorithm from Q3 is given in terms of parameters n and m. The asymptotic upper bound should be reasonably tight for the algorithm at hand. The worst-case time complexity analysis is mostly clearly justified with respect to the pseudocode from Q4(a). Any assumptions made in the analysis are mostly 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 n and m, 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 n and m, 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.