COMP4500/7500
Advanced Algorithms and Data Structures

Sch[illegible]echnology and Electrical Engineering
[illegible] Queensland, Semester 2, 2023

[illegible]signment 1

Due at 3:00pm, Friday [illegible]er 2023.
This assignment is worth [illegible] 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), 4(b) and 4(c) 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), 4(b) and 4(c) from part B in `a1.pdf`, as well as (ii) your source code file `SecretFinder.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 - incorrect submissions will receive 0 marks.

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. See section 5.3 of the course profile for details. Assessment submissions received after the due time (or any approved extended deadline) will be subject to a 100% late penalty. A one-hour grace period will be applied to the due time after which time the 100% late penalty will be imposed. This grace period is designed to deal with issues that might arise during submission (e.g. delays with Blackboard or Turnitin) and should not be considered a shift of the due time. Please keep a record of your submission time.

If there are medical or exceptional circumstances, then you may apply for an extension capped at a maximum of 14 days from the original deadline. Extensions must be requested via my.UQ (`https://my.uq.edu.au/`). If you have been granted an extension, then the 100% late submission penalty will apply to submissions made after the due date of the approved extension. The one-hour grace period will also apply to the extended deadline.

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 (25 marks total)

Question 1: Constructi[illegible]ed graph [5 marks total]

(a) (1 mark) **Creating y**[illegible]ssignment you are required to use your student number to generate input.

Take your student n[illegible]by "98" and postfix it by "52". This will give you a twelve digit initial input nu[illegible] 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:

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

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 [20 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) (10 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. 20.4 of the text [illegible] That means that you should draw the annotated graph for each stage of the sea[illegible]ke sure that you understand how the algorithm works.) Also show the start and fi[illegible]ertex.

For this question you [illegible] in numerical order in all relevant loops:

for each vertex [illegible]

for each vertex [illegible]

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

(c) (8 marks) Perform steps 3, 4 of the SCC algorithm. In your solution you must list (and draw) the trees in the depth-first forest in the order in which they were constructed. (You do not need to show working.)

Part B (75 marks total): Secret Finder

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

Organisations are interested in discovering *secrets*. Each secret has a *type*. There may be zero or more secrets of the same type. Initially, each organisation only knows one or more *initial secrets*. However, organisations may learn more secrets by invoking *exchange pacts* that have been formed between organisations.

There is a set of k *exchange pacts* that have been formed. Each exchange pact

$$\{\mathsf{firstOrg} \mapsto \mathsf{firstType},\ \mathsf{secondOrg} \mapsto \mathsf{secondType}\}$$

represents an agreement made between distinct organisations firstOrg and secondOrg in which firstOrg agrees to share all of its known secrets of type firstType with secondOrg and secondOrg agrees to share all of its known secrets of type secondType with firstOrg if *both* (i) firstOrg knows at least one secret of type firstType and (ii) secondOrg knows at least one secret of type secondType. It is permissible for firstType and secondType to be the same type of secret. There may be zero or more exchange pacts between the same two organisations.

When an exchange pact $\{\mathsf{firstOrg} \mapsto \mathsf{firstType},\ \mathsf{secondOrg} \mapsto \mathsf{secondType}\}$ is *invoked*, then if

(i) firstOrg knows at least one secret of type firstType when the pact is invoked, and

(ii) secondOrg knows at least one secret of type secondType when the pact is invoked,

then firstOrg shares all of the secrets of type firstType that it currently knows with secondOrg and secondOrg shares all of the secrets of type secondType that it currently knows with firstOrg. Otherwise (if either firstOrg does not know at least one secret of type firstType or secondOrg does not know at least one secret of type secondType), no secrets are shared. After a pact is invoked, each organisation knows the secrets that it knew before the pact was invoked, in addition to any secrets that were shared with it during the invocation.

Only one exchange pact can be invoked at any one time (so that invocations that have taken place can be represented by a, possibly-empty, sequence of exchange pact invocations), however there are no constraints on the number of times that an exchange pact can be invoked (it can be invoked zero or more times), or the order in which exchange pacts can be invoked.

Given a mapping from each organisation to the set of set of secrets that the organisation knows initially, and a set of exchange pacts that have been formed between those organisations, we say that an organisation

g *can discover* a secret s *if and only if* there exists a (possibly empty) sequence of invocations of formed exchange pacts, such that after the exchange pacts have been invoked (in the order described by the sequence of invocations), that organisation knows secret s.

Example 1 As a running example, consider the ($n = 6$) organisations $\{g0,\ g1,\ g2,\ g3,\ g4,\ g5\}$ and their corresponding initial secrets:

$$\begin{aligned} g0 &: \{s3\} \\ g1 &: \{s3\} \\ g2 &: \{s2, s4, s5\} \\ g3 &: \{s3\} \\ g4 &: \{s0\} \\ g5 &: \{s1\} \end{aligned}$$

where secrets $s0$, $s1$ and $s2$ are of secret type $t0$, secrets $s3$ and $s4$ are of secret type $t1$, and secret $s5$ is of secret type $t2$ (so that the number of secrets initially known by at least one organisation is $m = 6$); and the following ($k = 12$) exchange pacts that have been formed between them:

$$\begin{gathered} \{g0 \mapsto t0,\ g1 \mapsto t2\} \\ \{g0 \mapsto t1,\ g4 \mapsto t0\} \\ \{g0 \mapsto t1,\ g4 \mapsto t1\} \\ \{g0 \mapsto t2,\ g3 \mapsto t1\} \\ \{g1 \mapsto t0,\ g2 \mapsto t0\} \\ \{g1 \mapsto t1,\ g2 \mapsto t2\} \\ \{g2 \mapsto t1,\ g4 \mapsto t2\} \\ \{g3 \mapsto t0,\ g4 \mapsto t0\} \\ \{g3 \mapsto t2,\ g5 \mapsto t0\} \\ \{g4 \mapsto t0,\ g5 \mapsto t1\} \\ \{g4 \mapsto t2,\ g5 \mapsto t1\} \\ \{g4 \mapsto t2,\ g5 \mapsto t2\} \end{gathered}$$

We have, for example that organisation $g0$ can discover secret $s3$ because after the empty sequence of exchange pacts, $\langle\rangle$, have been invoked, organisation $g0$ knows $s3$ (because $s3$ is one of the initial secrets of $g0$).

Organisation $g0$ can also discover secrets $s0$, $s1$ and $s5$ because it will know these secrets after invoking the following sequence of exchange pacts (in the given sequential order):

$$\begin{gathered} \{g0 \mapsto t1,\ g4 \mapsto t0\} \\ \{g1 \mapsto t1,\ g2 \mapsto t2\} \\ \{g0 \mapsto t0,\ g1 \mapsto t2\} \\ \{g0 \mapsto t2,\ g3 \mapsto t1\} \\ \{g3 \mapsto t2,\ g5 \mapsto t0\} \\ \{g3 \mapsto t0,\ g4 \mapsto t0\} \\ \{g0 \mapsto t1,\ g4 \mapsto t0\} \end{gathered}$$

- When $\{g0 \mapsto t1,\ g4 \mapsto t0\}$ is first invoked, $g0$ knows at least one secret ($\{s3\}$) of type $t1$; and $g4$ knows at least one secret ($\{s0\}$) of type $t0$, and so $g0$ shares secrets $\{s3\}$ with $g4$ and $g4$ shares secrets $\{s0\}$ with $g1$.
- When $\{g1 \mapsto t1,\ g2 \mapsto t2\}$ is then invoked, $g1$ knows at least one secret ($\{s3\}$) of type $t1$; and $g2$ knows at least one secret ($\{s5\}$) of type $t2$, and so $g1$ shares secrets $\{s3\}$ with $g2$ and $g2$ shares secrets $\{s5\}$ with $g1$.

- When $\{g0 \mapsto t0,\ g1 \mapsto t2\}$ is then invoked, $g0$ knows at least one secret ($\{s0\}$) of type $t0$; and $g1$ knows at least one secret ($\{s5\}$) of type $t2$, and so $g0$ shares secrets $\{s0\}$ with $g1$ and $g1$ shares secrets $\{s5\}$ with $g0$.

- When $\{g0 \mapsto t2,\ g3$ [illegible]ked, $g0$ knows at least one secret ($\{s5\}$) of type $t2$; and $g3$ knows at least one se[illegible], and so $g0$ shares secrets $\{s5\}$ with $g3$ and $g3$ shares secrets $\{s3\}$ with $g0$.

- When $\{g3 \mapsto t2,\ g5$ [illegible]ked, $g3$ knows at least one secret ($\{s5\}$) of type $t2$; and $g5$ knows at least one se[illegible]), and so $g3$ shares secrets $\{s5\}$ with $g5$ and $g5$ shares secrets $\{s1\}$ with $g3$.

- When $\{g3 \mapsto t0,\ g4 \mapsto t0\}$ is then invoked, $g3$ knows at least one secret ($\{s1\}$) of type $t0$; and $g4$ knows at least one secret ($\{s0\}$) of type $t0$, and so $g3$ shares secrets $\{s1\}$ with $g4$ and $g4$ shares secrets $\{s0\}$ with $g3$.

- When $\{g0 \mapsto t1,\ g4 \mapsto t0\}$ is then invoked, $g0$ knows at least one secret ($\{s3\}$) of type $t1$; and $g4$ knows at least one secret ($\{s0, s1\}$) of type $t0$, and so $g0$ shares secrets $\{s3\}$ with $g4$ and $g4$ shares secrets $\{s0, s1\}$ with $g0$.

Organisation $g0$ cannot, however, discover secrets $s2$ or $s4$.

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

- A mapping from the organisations to their initial secrets, initialSecrets,
- a set of exchange pacts, exchangePacts
- an organisation $g \in$ initialSecrets.keySet(), and
- a secret s,

and returns true if and only if organisation g can discover secret s; otherwise the algorithm should return false.

For example, given the mapping from organisations to their initial secrets, and exchange pacts from Example 1,

1. given organisation $g0$, and secret $s0$, your algorithm should return true.
2. given organisation $g0$, and secret $s1$, your algorithm should return true.
3. given organisation $g0$, and secret $s2$, your algorithm should return false
4. given organisation $g0$, and secret $s3$, your algorithm should return true.
5. given organisation $g0$, and secret $s4$, your algorithm should return false.
6. given organisation $g0$, and secret $s5$, your algorithm should return true.

You algorithm must be designed and implemented as efficiently as possible.

Question 3: Design and implement an efficient solution (50 marks)

Design and implement an algorithm that answers the question above. Your algorithm should be as efficient as possible. Marks will be [illegible]t algorithms (e.g. a brute-force approach is not appropriate). Clearly structure and com[illegible] meaningful variable names.

- Your algorithm shou[illegible]n the static method `SecretFinder.canDiscover` from the `SecretFinder` class [illegible]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 `SecretFinder` 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 `SecretFinder`, since we will test your code using our original versions of these other files.

- You may not change the class name of the `SecretFinder` class or the package to which it belongs. You may not change the signature of the `SecretFinder.canDiscover` 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.)

- Your implementation should be in Java 1.8. 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 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.

Question 4: Worst-case analysis (25 marks)

This question involves performing an analysis of the *worst-case time complexity* and *worst-case space complexity* of your algorithm f[illegible]

(a) (5 marks) For the pur[illegible]e *time complexity* analysis in Q4(b) and the *worst-case space complexity* analysis [illegible]ar and concise pseudocode that summarizes the algorithm you used in your im[illegible]uestion 3. You may use the programming constructs used in Revision solutions, a[illegible]nce of common abstract data types like sets, maps, queues, lists, graphs, as well [illegible] sorting etc.

Clearly structure and comment your pseudocode. Use meaningful variable names.

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

(b) (12 marks) Let n be the total number of organisations, m be the total number of initial secrets[1], and k be the number of formed exchange pacts.

Provide an asymptotic upper bound on the worst case time complexity of your algorithm in terms of parameters n, m and k. 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.).

To simplify your analysis, you should make the (incorrect but simplifying) assumption that *HashSet* (or *HashMap*) operations that have expected-case time complexity $O(1)$ actually have worst-case time complexity $O(1)$. E.g. checking for set-membership in a *HashSet* has expected-case time complexity that is $O(1)$.

[Make your analysis as clear and concise as possible – it should be no more than a page using minimum 11pt font. Longer descriptions will not be marked. Also note that to receive any marks for this question, you must justify your solution – it is not enough to only give an asymptotic upper bound without explanation.]

(c) (8 marks) As for Q4(b), let n be the total number of organisations, m be the total number of initial secrets, and k be the number of formed exchange pacts.

Provide an asymptotic upper bound on the worst case space complexity of your algorithm in terms of parameters n, m and k. 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 space usage etc.).

[Make your analysis as clear and concise as possible – it should be no more than a page using minimum 11pt font. Longer descriptions will not be marked. Also note that to receive any marks for this question, you must justify your solution – it is not enough to only give an asymptotic upper bound without explanation.]

[1]The *total number of initial secrets* is defined to be the size of the set of all secrets s such that at least one organisation initially knows secret s.

Evaluation Criteria

Question 1

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

  1 : correct answe

  0 : answer not gi r 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) (10 marks)**

  10 : Correct answer given.

  8 : All stages of the traversal shown on an annotated graph (like CLRS Fig. 20.4 [4th]), including relevant features for each vertex (colour, immediate parent and start and finish times), but there are one or two minor mistakes.

  6 : All stages of the traversal are shown on an annotated graph (like CLRS Fig. 20.4 [4th]), 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.

  4 : Most stages of the traversal are shown on an annotated graph (like CLRS Fig. 20.4 [4th]), 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.

  2 : Most stages of the traversal are shown on an annotated graph (like CLRS Fig. 20.4 [4th]), 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) (2 marks)**

  2 : correct answer to question given

  0 : answer not given or contains one or more mistakes

- **Question 2 (c) (8 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.

  8 : 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.

6 : 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.

4 : All of the tre[illegible] forest are listed, however the order in which the trees were created may no[illegible]nay be one or two errors in the traversal that produced the forest.

2 : Either: (i) all [illegible]ected components of the graph are correctly listed, but the trees (from the [illegible]om which these connected components were derived are not clearly listed, or [illegible]n the depth-first forest are listed, however the order in which the trees were c[illegible]ear 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 two pages 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 implementa[illegible]e pseudocode.

2 : Pseudocode gi[illegible] the algorithm you used in your implementation from Question 3. Potenti[illegible]an necessary. May not be commented, or have a very clear correspondence [illegible]entation from Q3 and the pseudocode.

1 : Pseudocode gi[illegible] summarize the algorithm you used in your implementation from Question [illegible]ear, or overly verbose, or the correspondence to the actual implementation [illegible]

0 : Work with little or no academic merit

- **Question 4(b) (12 marks)**

For this part of the question, the analysis should be no more than one 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.

12 : A correct asymptotic upper bound on the worst-case time complexity of the algorithm from Q3 is given and justified in terms of parameters n, m and k. 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.

9 : A mostly correct asymptotic upper bound on the worst-case time complexity of the algorithm from Q3 is given and justified in terms of parameters n, m and k. 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.

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

3 : An attempt has been made to give and justify an asymptotic upper bound on the worst-case time complexity of the algorithm from Q3 in terms of parameters n, m and k, however it contains either a major mistake, or many mistakes, or gives an unreasonably loose upper bound, or the justification for the analysis with respect to the pseudocode from Q4(a) may be poor (even if the bound is correct).

0 : Work with little or no academic merit.

- **Question 4(c) (8 marks)**

For this part of the question, the analysis should be no more than 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.

8 : A correct asymptotic upper bound on the worst-case space complexity of the algorithm from Q3 is given and justified in terms of parameters n, m and k. The asymptotic upper bound should be as tight as reasonably possible for the algorithm at hand. The worst-case space 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 asympt[illegible]y given has been simplified to remove lower order terms and unnecessary co[illegible]

6 : A mostly corr[illegible] bound on the worst-case space complexity of the algorithm from Q3 is give[illegible]ms of parameters n, m and k. The asymptotic upper bound should be reaso[illegible]gorithm at hand. The worst-case space complexity analysis is mostly clearly[illegible]ct to the pseudocode from Q4(a). Any assumptions made in the analysis are[illegible]nd clearly stated.

4 : A reasonable attempt has been made to give and justify a reasonably-tight asymptotic upper bound on the worst-case space complexity of the algorithm from Q3 in terms of parameters n, m and k, however either it contains some minor mistakes but is otherwise reasonably justified, or aspects of the justification of the analysis with respect to the pseudocode from Q4(a) are unclear. Assumptions made in the analysis may be unclear.

2 : An attempt has been made to give and justify an asymptotic upper bound on the worst-case space complexity of the algorithm from Q3 in terms of parameters n, m and k, however it contains either a major mistake, or many mistakes, or gives an unreasonably loose upper bound, or the justification for the analysis with respect to the pseudocode from Q4(a) may be poor (even if the bound is correct).

0 : Work with little or no academic merit.