NAME: ....

STUDENT NUMBER: ....

THE UNIVERSITY OF QUEENSLAND
School of Information Technology and Electrical Engineering

Midsemester Examination, Semester Two 2020

COMP4500/7500 ADVANCED ALGORITHMS and DATA STRUCTURES

Exam Duration: 60 minutes (plus 30 additional minutes for uploading answers)

Reading Time: 0 minutes (no reading time)

Exam Conditions:
This examination is open book.
You may not communicate with anyone regarding the exam during this time period.

Instructions To Students:

- The exam questions will be uploaded as a PDF to Blackboard Tuesday 15/9/2020 at 8:00am. The exam sheet will only be available for the first 30 minutes, please download the sheet during this time.
- You have until the end of the 90 minute period to submit your answers to Blackboard in a single PDF file.
- Please ensure you have access to a scanner or a good quality camera and can create a PDF file from these for submission. Your work must be easy to read and scanned or pictured answers must be very clear. It is your responsibility to ensure you upload your answers within this time period, late submissions will not be marked.
- This examination is designed to be completed in one hour. The additional 30 minutes is to be used for uploading solutions. To ensure your answers are submitted please finish working after 1 hour to ensure you have time to upload your solutions.
- You may print the exam and write on the exam paper, write your answers on blank paper or write electronically using a suitable device.
- You are required to sign the declaration below indicating that the work you have submitted is your own, and that you have followed the rules set for this examination. You should submit a signed copy of this coversheet alongside your assignment.
- Answer ALL five (5) questions in your Blackboard PDF submission. Marks for each question are shown in brackets. The total number of marks for this paper is 100.

Certification (must be signed before submission):
I certify that my submitted answers are entirely my own work and that I have neither given nor received any unauthorised assistance on this assessment item.

Signed: [illegible] Date: 15/09/2020

Question 1 [15 marks]

Let f and g be non-decreasing functions such that $f(n) \in O(n^a)$ and $g(n) \in O(\lg n)$ where $a \geq 1$ and let $h(n) = g(f(n))$. Using the definition of O-notation, prove that the function $h(n) \in O(\lg n)$ for any real constant $a \geq 1$. Show your working.
You may find the following log laws useful:

$$\lg(x \cdot y) = \lg(x) + \lg(y)$$
$$\lg(x^y) = y \cdot \lg(x)$$

Hint: If $n \geq 2$ then $k \leq k \cdot \lg n$ for any constant k.

Question 2 [20 marks]

Solve the following recurrence equations to get a tight asymptotic bound on the complexity of the corresponding functions. Justify your answers by using the master theorem. You may assume that $T(n) = \Theta(1)$ for suitable constant n. (You may leave logarithmic expressions in your answers if they evaluate to non-integral values.)

a) [10 marks] $T(n) = 8T(\frac{n}{3}) + n^3$

b) [10 marks] $T(n) = 4T(\frac{n}{2}) + n^2 + 3n$

Question 3 [15 marks]

Given the following recurrence

$$\begin{aligned} T(n) &= 1 && \text{for } 1 \leq n \leq 2 \\ T(n) &= T(n-2) + 4n && \text{for } n \geq 3 \end{aligned}$$

prove that $T(n) \in O(n^2)$ using the substitution method. Clearly describe your inductive assumptions and boundary conditions, and show your working. Justify that what you have proven is sufficient to show that $T(n) \in O(n^2)$ by referring to the definition of O-notation.

Question 4 [20 marks]

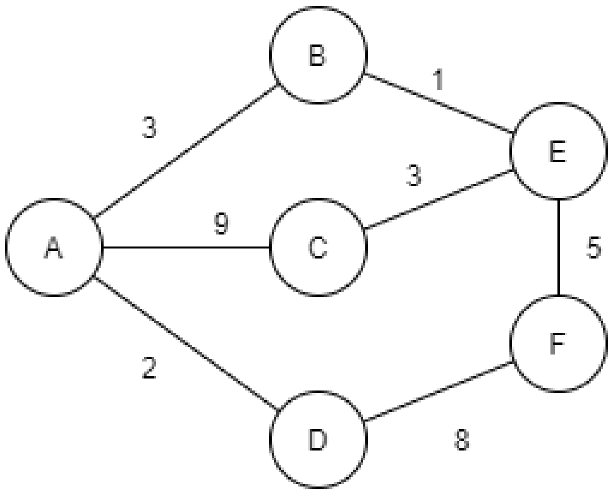
Dijkstra's algorithm (with pseudocode given below) takes as input a connected, weighted graph without negative weight edges G with vertices $G.V$, edges $G.E$, weights w and a source vertex $s \in G.V$ and solves the single-source shortest paths problem.

```
DIJKSTRA($G, w, s$)
1  // $G$ is the graph, $w$ the weight function, $s$ the source vertex
2  INIT-SINGLE-SOURCE($G, s$)
3  $S = \emptyset$          // $S$ is the set of visited vertices
4  $Q = G.V$       // $Q$ is the priority queue maintaining $G.V - S$
5  while $Q \neq \emptyset$
6      $u =$ EXTRACT-MIN($Q$)
7      $S = S \cup \{u\}$
8      for each vertex $v \in G.Adj[v]$
9          RELAX($u, v, w$)
```

```
INIT-SINGLE-SOURCE($G, s$)
1  for each vertex $v \in G.V$
2      $v.d = \infty$
3      $v.\pi =$ NIL
4  $s.d = 0$
```

```
RELAX($u, v, w$)
1  if $v.d > u.d + w(u, v)$
2      $v.d = u.d + w(u, v)$
3      $v.\pi = u$
```

a) [10 marks]
List the distance to all vertices from s ($v.d$), the parents of all vertices ($v.\pi$) and the vertices in the order in which they are added to the set of visited vertices, S if Dijkstra's algorithm is executed on the following undirected weighted graph, given that the source vertex s is taken to be vertex A.



b) [10 marks] Your friend tells you that they have come up with an algorithm which can solve the shortest path problem with negative weight edges in $O((V + E) \lg V)$ time. You are sceptical so you ask them to show you their idea and they give you the following psuedocode.

```
MODIFIED-DIJKSTRA($G, w, s$)
// find the minimum weight edge in the graph
min = 0
for each $u \in G.V$
    for each $v \in G.Adj[v]$
        if $w(u, v) <$ min
            min $= w(u, v)$

// then increase all edge weights by |min|
for each $u \in G.V$
    for each $v \in G.Adj[v]$
        $w(u, v) = w(u, v) + |\text{min}|$
// now run Dijksta's algorithm with the new graph
return DIJKSTRA(G,w,s)
```

Does this algorithm solve the shortest path problem for graphs with negative weight edges? If not, provide an example graph for which this algorithm would *not* work.

Question 5 [30 marks]

Prim's algorithm (with pseudocode given below) takes as input a connected, undirected, weighted graph G with vertices $G.V$, edges $G.E$, weights w and a vertex $r \in G.V$ and returns the set of edges in a minimum spanning tree of G.

```
MST-PRIM($G, w, r$)
$T = \emptyset$
for each $u \in G.V$
    $u.key = \infty$
    $u.\pi$ = NIL
$r.key = 0$
$Q = G.V$
while $Q \neq \emptyset$
    $u$ = EXTRACT-MIN($Q$)
    if $u \neq r$
        $T = T \cup \{(u, u.\pi)\}$
    for each $v \in G.Adj[u]$
        if $v \in Q$ and $w(u, v) < v.key$
            $v.\pi = u$
            $v.key = w(u, v)$    // Decrease key
return $T$
```

This implementation assumes that G is implemented using an adjacency list structure. Suppose instead, that you are trying to find the minimum spanning tree of a graph G that is implemented with an adjacency matrix such that $w(u, v) = \infty$ if u and v do not have an edge between them.

a) [10 marks]
What modifications would need to be made to Prim's algorithm as described in the provided pseudocode to find the minimum spanning tree of G.? You may use line numbers to reference your changes. You may not convert the adjacency matrix to an adjacency list for this question.
b) [10 marks]
Another approach would be to first convert the adjacency matrix representation of G to an adjacency list representation and then use Prim's algorithm as described in the provided pseudocode. Write a pseudocode algorithm for solving the minimum spanning tree problem using this approach. You may assume the following abstract functions exist:

- MST-PRIM(F, w, r) as described in the pseudocode above.
- INITIALISE-EMPTY-GRAPH() which initialises an empty graph with no vertices in $\Theta(1)$ time.
- G.ADD-VERTEX(v) which adds a vertex to G with an empty adjacency list in $\Theta(1)$ time.
- l.APPEND(v) which appends v to the list l in $\Theta(1)$ time.

c) [10 marks]
Provide a big-O upper bound for each of the two approaches ((a) and (b)). Is one of these approaches superior in terms of asymptotic time complexity?