Venue ____________________

Student Number |__|__|__|__|__|__|__|__|

Family Name ____________________

First Name ____________________

Curtin University

Department of Computing

EXAMINATION

End of Semester 1, 2017

COMP3001 Design and Analysis of Algorithms

This paper is for all students

This is a CLOSED BOOK examination

Examination paper IS NOT to be released to student

Examination Duration 2 hours

Reading Time 10 minutes

Notes in the margins of exam paper may be written by Students during reading time

Total Marks 100

Supplied by the University

none

Supplied by the Student

Materials

none

Calculator

No calculators are permitted in this exam

Instructions to Students

This paper contains four (4) questions with the following breakdown of marks:

Question One: 22 marks

Question Two: 21 marks

Question Three: 21 marks

Question Four: 36 marks

For Examiner Use Only

Q	Mark
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	

Total ________

QUESTION ONE (Total: 22 marks).

a) **(Total: 7 marks).** Consider the following function.

```
function_X (A, n)

    if n = 0
        function_X ← False

    else if A[n] == 0 then
        function_X ← True

    else
        function_X (A, n − 1)
```

(i) **(2 marks).** What does the function do?

(ii) **(2 marks).** Give the recurrence of the time complexity of the algorithm. Explain your answer.

(iii) **(3 marks).** Guess its upper bound time complexity. Explain your answer. **Note:** *You don't need to formally prove your solution.*

Answer:

(i)

(ii)

(iii)

b) **(Total: 8 marks).** Consider the following function A.

Function A:

```
for i ← 1 to n do
    j ← i
    while j > 0 do
        Set j ← j/2    // an integer division
```

(i) **(4 marks).** For $n = 4$, how many times will the instruction "$j \leftarrow j/2$" be executed?

(ii) **(4 marks).** What is the upper bound time complexity of the function? Explain your answer.

Answer:

(i)

(ii)

c) **(Total: 7 marks).** Suppose you are designing a divide and conquer algorithm for a certain problem. Your algorithm needs dividing a size n input into a number of sub-problems, each of size $n/3$. However, you don't know how many sub-problems, say x, are required in your algorithm; but you know that your algorithm requires a routine that takes $\Theta(n^2)$ time to recombine the sub-solutions. Let $T(n)$ be the time complexity of your algorithm to solve the problem of size n.

(i) **(2 marks).** State the recurrence for the time complexity of your algorithm using the described information. Explain your answer.

Hint. You have to include variable x in your recurrence.

(ii) **(5 marks).** Your goal is to design the algorithm such that it has a time complexity of $\Theta(n^2)$, for an input size of n. What is the largest number of sub-problems, i.e., x, to achieve the goal? Explain your answer.

Answer:

(i)

(ii)

END OF QUESTION ONE

QUESTION TWO (Total: 21 marks).

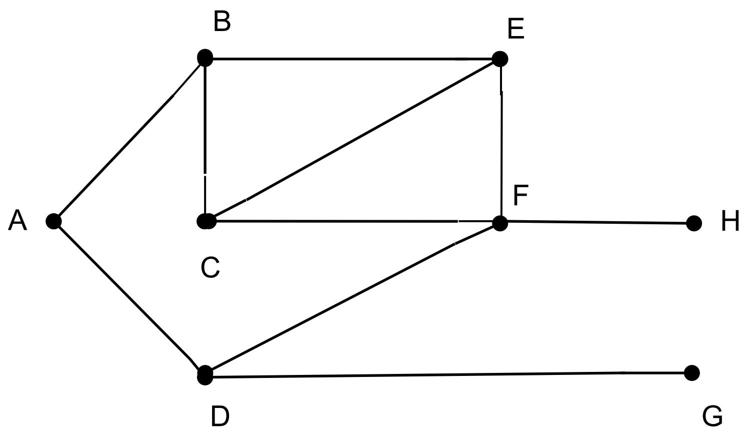
a) **(Total: 6 marks).** Consider the following BFS algorithm in which graph G is represented by an adjacency list L.

BFS_Tree_G(V, E)
Input: $G = (V,E)$. $L[x]$ refers to the adjacency list of x.
Output: The BFS tree T;

1. Mark all vertices *new* and set $T = \{\ \}$
2. Mark the start vertex v =*old*
3. insert (Q, v) // ***Q* is a queue**
4. **while** Q is nonempty **do**
5. $\quad x = dequeue(Q)$
6. $\quad$ **for** each vertex w in $L[x]$ marked *new* **do**
7. $\quad\quad T = T \cup \{x, w\}$
8. $\quad\quad$ Mark w = *old*
9. $\quad\quad$ insert (Q, w)

(i) **(2 marks).** Explain why the algorithm has a time complexity of $O(|V| + |E|)$.

(ii) **(2 marks).** Draw the BFS tree for the following graph, starting from **node A**. Follow alphabetical order when necessary.



(iii) **(2 marks).** List the contents of queue Q when the queue is the longest.

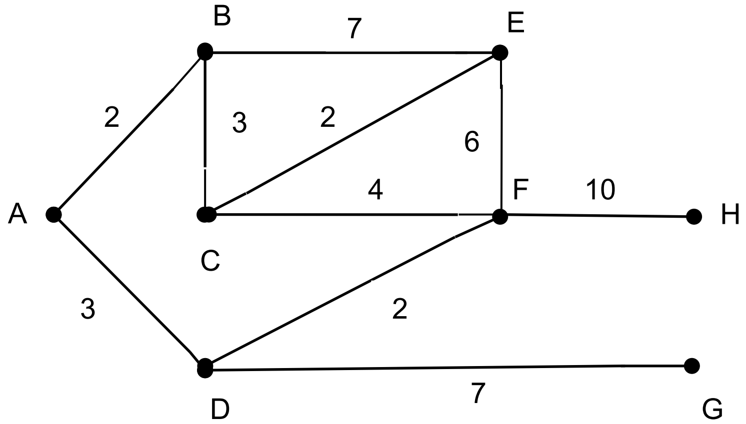
<u>**Answer:**</u>

(i)

(ii) Breadth first search tree

(iii)

b) **(Total: 15 marks).** Consider the following network of cities (A, B, …, H), where each number represents the distance between two connected cities. You are asked to layout rails so that it would be possible to reach any city from any other. The goal is to lay down the *cheapest* combination of rail segments. **Note:** not all cities can be connected by direct segments of rail to all others, and that the distances are given only between those pairs of cities that can be connected. Further, assume that the *cost* of directly connecting any two cities is proportional to the *distance* between them, i.e., *the shorter the rail segment means the cheaper*.



(i) **(3 marks).** Show the adjacency list representation for the network.

(ii) **(2 marks).** What algorithm or algorithms that have been discussed in the lecture can be used to solve the problem? Briefly explain the reason.

(iii) **(8 marks).** Use your selected algorithm in (ii) to solve the problem. Show details of your solution, and compute the cheapest cost.

Note: when selecting some alternatives that have the same value, select one based on alphabetical order.

(iv) **(2 marks).** What algorithm design (e.g., divide and conquer) is used in the algorithm(s) in part (ii)? Briefly explain the reason.

Answer:

(i) Adjacency list representation.

(ii)

(iii) Solution

(iv) Approach.

END OF QUESTION TWO

QUESTION THREE (Total: 21 marks).

a) **(Total: 10 marks).** Consider an array A that contains n integers and a key x of type integer. You want to know if **there are three integers in A that give sum equal to x.** For example, for A = <2, 4, 6, -1, 5, 5> and $x = 11$, you find $2 + 4 + 5 = 11$.

(i) **(3 marks).** Write the pseudocode of an $\boldsymbol{O(n^3)}$ algorithm that returns **True** if there are three integers in A that give sum equal to x; otherwise, it returns **False**. Show that the algorithm has a time complexity of $O(n^3)$.

(ii) **(7 marks).** Write the pseudocode of **another** algorithm to solve the problem in time $\boldsymbol{O(n^2)}$. Show that the algorithm has a time complexity of $O(n^2)$. You are required to give two different solutions in part (i) and part (ii).

<u>**Hint.**</u> For each integer $a \in A$, you need to find two integers $b, c \in A$ whose sum is equal to $x - a$. Isn't it similar to the problem solved in your tutorial?

<u>**Answer:**</u>

(i) $O(n^3)$ solution.

(ii) $O(n^2)$ solution.

b) **(Total: 11 marks).** Suppose you have a rod of n units long to sell. Let p_i be the price of a rod of integer-length i, for $i \in \{1, 2, 3, \dots n\}$. Your goal is to cut the rod of length n into some integer-length pieces such that you can maximize the total sale price. This problem is called the *Rod-cutting problem*.

Suppose you want to use a greedy technique to solve the *Rod-cutting problem* for a rod of length n, and some given prices p_i, for $i \in \{1, 2, 3, \dots n\}$.

(i) **(3 marks).** What greedy criterion should you use (i.e., what is the algorithm "greedy" on)? Justify your chosen greedy criterion. **You are not asked to write a pseudocode for this question.**

(ii) **(4 marks).** Consider you have a rod of length $n = 10$, and assume the price of each rod of length 1, 2, 3, 5, and 6 are p_1 = \$2, p_2 = \$3, p_3 = \$6, p_5 = \$12, and p_6 = \$15. Use your greedy approach in part (i) to generate the total sale prices for this example.

(iii) **(4 marks).** What is the time complexity of the greedy approach in part (i) if you consider only a constant number of different prices, i.e., lengths? Justify your answer.

Answer

(i)

(ii) Solution for the example.

(iii) Time complexity.

END OF QUESTION THREE

QUESTION FOUR (Total: 36 marks).

a) **(4 marks).** When using the Rabin-Karp string-matching algorithm, explain why choosing a small q is not a good idea. How large q should be?

Answer:

b) **(Total: 10 marks).** Consider the word "Mississippi".

(i) **(5 marks).** Construct the Huffman code for each letter in "Mississippi".

(ii) **(5 marks).** Generate the list of number pairs computed by Zip-Lempel algorithm (as used in gzip) for the word "Mississippi", given the following ASCII codes:

M	77
i	105
p	112
s	115

Hint: there are 9 pairs

Answer

(i) Huffman code for each letter in "Mississippi".

(ii) Zip-Lempel

c) **(Total: 16 marks)**. Consider the following **incomplete** tables generated by the dynamic programming algorithm for solving the matrix chain problem, where the five consecutive matrices are A, B, C, D, and E.

m (*j* \ *i*)	1	2	3	4	5
1	0				
2	20	0			
3	35	12	0		
4	44	21	9	0	
5	?	37	21	36	0

s (*j* \ *i*)	1	2	3	4	5
1					
2	1				
3	2	2			
4	2	2	3		
5	?	2	4	4	

(i) **(3 marks).** If matrices A and D have dimensions A = 5x4, and D = 3x3, what are the dimensions of matrices B and C, and E?

(ii) **(6 marks).** Calculate $m[1, 5]$ and $s[1, 5]$.

(iii) **(2 marks).** Show the optimal bracketing of the matrices in part (i) that produces the number of multiplications in $m[1, 5]$.

(iv) **(5 marks).** Let $P(n)$ be the number of alternative parenthesizations of a sequence of n matrices. Calculate $P(5)$, i.e., the total number of possible parenthesizations that you need to consider to obtain the solution in part (iii) if you use **a brute-force approach**. Explain your answer.

<u>Answer:</u>

(i) Dimensions of the matrices.

(ii) $m[1, 5]$ and $s[1, 5]$.

(iii) Bracketing

(iv) Compute P(5).

d) **(Total: 6 marks).** Consider the following parallel algorithm **XXX**.

Algorithm XXX
for $i = 1$ to $\log n$ **do**
  forall P_j where $1 \le j \le n/2$ **do in parallel**
    if $2j \bmod 2^i = 0$ **then**
      $A[2j] \leftarrow A[2j] + A[2j - 2^{i-1}]$
    endif
  endfor
endfor

(i) **(1 mark).** What does algorithm **XXX** do?

(ii) **(2 marks).** Compute the work complexity of **XXX**. Explain your answer.

(iii) **(1 mark).** Is **XXX** work efficient? Why?

(iv) **(2 marks).** Is EREW PRAM model sufficient to run **XXX**? Explain your answer.

<u>**Answer:**</u>

(i)

(ii)

(iii)

(iv)

END OF QUESTION FOUR

Attachment

Logarithms:

$\log_b a = \lg a/\lg b$, $\log_{10} 2 = 0.301$, $\log_{10} 3 = 0.477$, $\log_{10} 4 = 0.602$, $\log_{10} 7 = 0.845$, $\log_3 9 = 2$, $\log_3 8 = 1.89$, $\log_2 3 = 1.585$.

Master Theorem:

if $T(n) = aT(n/b) + f(n)$ then

$$T(n) = \begin{cases} \Theta\left(n^{\log_b a}\right) & f(n) = O\left(n^{\log_b a - \varepsilon}\right) \to f(n) < n^{\log_b a} \\ \Theta\left(n^{\log_b a} \lg n\right) & f(n) = \Theta\left(n^{\log_b a}\right) \to f(n) = n^{\log_b a} \\ \Theta(f(n)) & f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right) \to f(n) > n^{\log_b a} \\ & \text{if } af(n/b) \le cf(n) \text{ for c} < 1 \text{ and large } n \end{cases}$$

Huffman (*C*)

```
n ← |C|
Q ← C
for i ← 1 to n-1
        do allocate a new node z
                left[z] ← x ← EXTRACT-MIN(Q)
                right[z] ← y ← EXTRACT-MIN(Q)
                f[z] ← f[x] + f[y]
                INSERT(Q, z)
return EXTRACT-MIN(Q) //return the root of the tree
```

$$H = \sum_{i=1}^{n} - p_i \log_2 p_i$$

Prim's Algorithm

MST-PRIM(G, w, r)

```
for each u ∈ V[G]
    do key[u] ← ∞
        π[u] ← NIL   // π[u]  means parent of vertex u
key[r] ← 0
Q ← V[G]                    // BUILD-MIN-HEAP
while Q ≠ φ
    do u ← EXTRACT-MIN (Q)
        for each v ∈ Adj[u]
            do if v ∈ Q and w(u,v) < key[v]
                then π[v] ← u
                    key[v] ← w(u,v)        // DECREASE-KEY
```

Kruskal's Algorithm
Input: An undirected graph G(V,E) with a cost function c on the edges
Output: T the minimum cost spanning tree for G

```
T ← {}
VS ← {}
for each vertex v ∈ V[G] do
    VS ← VS ∪ {v}
    Sort the edges of E in nondecreasing order of weight c

    for each edge (v,w) ∈ E, taken in nondecreasing order by weight c do
        if v and w are in disjoint sets W1 and W2 in VS then
            W1 ← W1 ∪ W2
            VS ← VS – W2
            T ← T∪ (v,w)
    return T
```

- VS is a set of disjoint-sets of vertices; Initially each vertex is in in a set by itself in VS.
- Each set W in VS represents a connected set of vertices forming a spanning tree.

Dijkstra_Single_source_shortest_path_G(V, E, u)
Input: G =(V,E), the weighted directed graph and u the source vertex
Output: for each vertex, v, $d[v]$ is the length of the shortest path from u to v.

```
mark vertex u
d[u] ← 0
for each unmarked vertex  v ∈ V do
    if edge (u,v) exists then d [v] ← weight (u,v)
    else d[v] ← ∞
while there exists an unmarked vertex do
    let v be an unmarked vertex such that d[v] is minimal
    mark vertex v
    for all edges (v,x) such that x is unmarked do
        if d[x] > d[v] + weight[v,x] then
            d[x] ←  d[v] + weight[v,x]
```

END OF EXAMINATION PAPER