Venue ____________________

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

Family Name ____________________

First Name ____________________

School of Electrical Engineering, Computing and Mathematical Sciences

EXAMINATION

End of Semester 1, 2018

COMP3001 Design and Analysis of Algorithms

This paper is for Bentley Campus 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: 24 marks
Question Three: 18 marks
Question Four: 36 marks

ATTEMPT ALL QUESTIONS

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: 9 marks).** Consider the following algorithm.

Function A (x, y) // $y \leq x$
  $a \leftarrow 0$
  $b \leftarrow x$
  while $y \leq b$ **do**
    $b \leftarrow b - y$
    $a \leftarrow a + 1$

  return a, b

(i) **(3 marks).** What will be returned by **Function A** (i.e., the values of a and b) given input $x = 525$, and $y = 50$?

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

(iii) **(2 marks).** What is the worst-case time complexity of the algorithm?
Hint. The input size is the value of x.

(iv) **(2 marks).** What is the best-case time complexity of the algorithm?

Answer:

(i)

(ii)

(iii)

(iv)

b) **(5 marks).** Consider an array A that contains n integers. Design an algorithm to find three integers in A that give the largest sum. Example A = <2, 4, 6, -1, 5, 5> will result in 6 + 5 + 5= 16. Your algorithm should be as efficient as possible in terms of its upper bound time complexity. What is the time complexity of your algorithm?
Note: you are not required to present your algorithm in pseudocode. A clear explanation is sufficient.

Answer:

c) **(4 marks).** Consider a function whose body is

sum ← 0
for i ← 1 to f(n) **do**
    sum ← sum + i

What is the upper bound running time complexity of the function if f(n) is $O(n^2)$ and the value of f(n) is n? Explain your answer.

Answer:

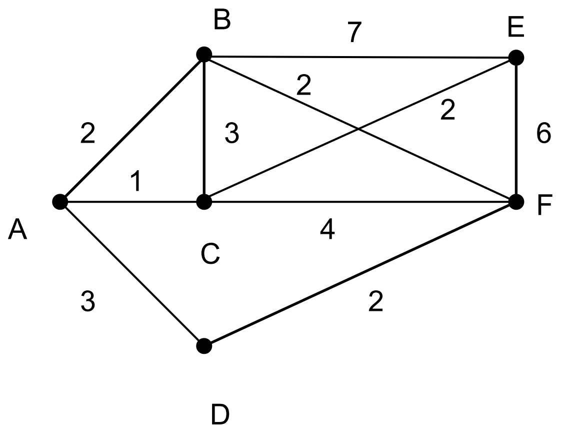
d) **(4 marks).** Show the **recurrence tree** for $T(n) = T(n/5) + T(4n/5) + n$ to guess its asymptotic upper bound complexity. **Hint.** Similar to one question in your assignment.

Answer:

END OF QUESTION ONE

QUESTION TWO (Total: 24 marks).

a) **(Total: 8 marks).** Consider the following weighted graph.



(i) **(3 marks).** Show the adjacency/weight matrix representation for the network.

(ii) **(1 mark).** Give one maximal clique for the graph.

(iii) **(4 marks).** Give the minimum vertex cover for the graph if using a greedy approach, i.e., the correct solution to your assignment. Show the details of generating the result. Use alphabetical order when necessary.

Answer:

(i) Adjacency/weight matrix

(ii) Maximal clique:

(iii) Minimum Vertex cover:

b) **(Total: 16 marks).** Consider the following Kruskal's algorithm (copied from the lecture slide).

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    // Line A
            W1 ← W1 ∪ W2   // Line B
            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.

(i) **(7 marks).** Use the Kruskal's algorithm to find the minimum cost spanning tree of the network. Use alphabetical order when necessary.

(ii) **(Sub-total: 9 marks).** Answer the each of the following questions.

- **(2 marks).** What is the purpose of using **Line A**?
- **(5 marks).** Explain one possible data structure to implement the disjoint set used in Kruskal's algorithm. You must describe the data structure and the operators used to implement the disjoint set.
- **(2 marks).** What is the time complexity of **Line A** and **Line B** for each iteration? Justify your answer.

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

(i) Kruskal's solution.

(ii)

- Purpose of **Line A:**

- Data structure to implement disjoint set:

- Time complexity.

END OF QUESTION TWO

QUESTION THREE (Total: 18 marks).

a) **(8 marks).** Consider a sequence of n integers $A = \langle a_1, a_2, \ldots, a_n \rangle$. For $j \geq i$, let A_{ij} be a subsequence of A that starts from a_i and ends at a_j, i.e., $A_{ij} = \langle a_i, a_{i+1}, \ldots, a_j \rangle$, and let S_{ij} be the sum of all integers in A_{ij}, i.e., $S_{ij} = a_i + a_{i+1} + \ldots + a_j$. Note that $S_{ii} = a_i$. The problem is to find the maximum S_{ij}.

For example, for $A = \langle 2, -5, 2, -1, 4, -9, 4 \rangle$, $S_{11} = 2$, $S_{12} = 2 + (-5) = -3$, $S_{13} = 2 + (-5) + 2 = -1$, $S_{33} = 2$, $S_{34} = 2 + (-1) = 1$, and the maximum consecutive subsequence has the sum of $S_{35} = 2 + (-1) + 4 = 5$.

Write the pseudocode of an $O(n^2)$ algorithm to find the maximum S_{ij}, and the starting and ending indices of the maximum subsequence, i.e., i and j. Explain why your algorithm is $O(n^2)$. For the example, your pseudocode gives as output the maximum subsequence = 5, $i = 3$ and $j = 5$ because the maximum S_{ij} is for $S_{35} = 5$.

Hint. What if you generate all possible S_{ij}, and find the maximum with its starting and ending indices?

<u>Answer:</u>

b) **(Total: 10 marks).** Consider the following recursive function to produce the n^{th} Fibonacci number.

```
Fib (n)
   if n ≤ 1
       return n
   else
       x = Fib (n – 1)
       y = Fib (n – 2)
       return (x + y)
```

(i) **(2 marks).** Give the recurrence function of the time complexity of **Fib (*n*)**. Justify your answer.

(ii) **(2 marks).** Explain why the time complexity of the recurrence is $O(2^n)$. **Note:** you **are not required** to give a formal proof, e.g., by induction. A short but clear argument is sufficient for your explanation.

(iii) **(4 marks).** Write the pseudocode of the **top-down** dynamic programming function for Fib (n). **Hint.** Similar to how to convert the recursive function of the knapsack into its top down dynamic programming. You can use an array F that can store n integers. You may assume that each element in F has been initialized with value of -1.

(iv) **(2 marks).** Explain why the time complexity of the **top-down** dynamic programming function for Fib (n) in part (iii) is $O(n)$.

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

(i)

(ii)

(iii) Top-down dynamic programming

(iv) Time complexity

END OF QUESTION THREE

QUESTION FOUR (Total: 36 marks).

a) **(Total: 6 marks).** A file contains only digits in the following frequency: 0 (10), 1 (42), 2 (11), 3 (36), 4 (12), 5 (34), 6 (13), 7 (17), 8 (14), 9 (16).

(i) **(3 marks).** Draw a code tree for Huffman code.

(ii) **(3 marks).** Construct the Huffman code.

The pseudocode of Huffman's algorithm is given below.

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
```

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

(i) Code tree:

(ii)

b) **(Total: 6 marks).** Consider the following Rabin-Karp string matcher algorithm.

KARP-MATCHER (*T*, *P*, *d*, *q*)
Input: Text T, pattern P, radix d (which is typically $= |\Sigma|$), and the prime q.
Output: valid shifts s where P matches

1. $n \leftarrow length[T]$
2. $m \leftarrow length[P]$
3. $h \leftarrow d^{m-1} \bmod q$
4. $p \leftarrow 0$
5. $t_0 \leftarrow 0$
6. **for** $i \leftarrow 1$ to m
7.   **do** $p \leftarrow (d*p + P[i]) \bmod q$
8. $t_0 \leftarrow (d*t_0 + T[i]) \bmod q$
9. **for** $s \leftarrow 0$ to $n\text{-}m$
10.   **do if** $p = t_s$
11.   **then if** $P[1..m] = T[s+1..s+m]$
12.   **then** "pattern occurs with shift s "
13.   **if** $s < n\text{-}m$
14.   **then** $t_{s+1} \leftarrow (d*(t_s - T[s+1]*h) + T[s+m+1]) \bmod q$

(i) **(4 marks).** For $T = 47691447$, P=47, d=10, and q=11, use Line 6-8 of the algorithm to compute t_0 and Line 14 to compute t_1.

(ii) **(2 marks).** For the example in part (i), how many spurious hits are there?

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

(i)

(ii)

c) **(Total: 14 marks).** Consider the following instance of the 0/1 knapsack problem for capacity C = 9, weights w = [1, 6, 3, 2, 5], and profits p = [6, 18, 9, 7, 11], and the following dynamic programming program.

Knapsack(S, C)
Input: Set S of n items with p_i profit and w_i weight, and maximum total weight C
Output: maximum profit $P[w]$ of a subset S with total weight at most w, for w= 0, 1, … C

```
for k ← 0 to C do
    P[k] ← 0
for i ← n downto 1 do        // Line A
    for k ← C downto w_i do      // Line B
        if P[k- w_i] + p_i > P[k] then
            P[k] ← P[k- w_i] + p_i
```

(i) **(10 marks).** Use the program to fill in the entries in the following table, find the optimal profit, and determine what items should be selected to achieve the optimal profit.

(ii) **(2 marks).** Will the program give the correct result if we replace "for $i \leftarrow n$ downto 1 do" in **Line A** to "for $i \leftarrow 1$ to n do"? Explain your answer.

(iii) **(2 marks).** Will the program give the correct result if we replace "for $k \leftarrow C$ downto w_i do" in **Line B** to "for $k \leftarrow w_i$ to C do"? Explain your answer.

Answer:

$i \backslash k$	0	1	2	3	4	5	6	7	8	9
5										
4										
3										
2										
1										

Selected items:

Profits:

d) **(Total: 10 marks).** Consider the following parallel search algorithm.

Algorithm Parallel_Search (x, A[1 .. n])
index ← -1 // initialized with an invalid index value
forall P_i **do in parallel** // $1 \leq i \leq n$
  if $A[i] = x$ **then**
    index ← i
  endif
endfor

(i) **(2 marks).** Explain why the algorithm can not be used in the EREW model.

(ii) **(2 marks).** Which type of CRCW model is used? (i.e., **common:** all PEs write same thing, **arbitrary:** only store one value, e.g., choose PE with the smallest index, or **reduction:** apply min, max, sum). Justify your answer.

(iii) **(4 marks).** Is the parallel algorithm cost optimal and cost efficient? Justify your answer.

(iv) **(2 marks).** Explain how to modify the CRCW algorithm so that it can run in an CREW model. **Note:** You are not asked to write the pseudocode of the CREW algorithm.

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

(i)

(ii)

(iii)

(iv)

END OF EXAMINATION PAPER