CURTIN UNIVERSITY

School of Electrical Engineering, Computing and Mathematical Sciences

Computing Discipline

Test 2 – S1/ 2019

SUBJECT: Design and Analysis of Algorithms Unit Code COMP3001

TIME ALLOWED:

55 minutes test. The supervisor will indicate when answering may commence.

AIDS ALLOWED:

To be supplied by the Candidate: Nil

To be supplied by the University: Nil

Calculators are NOT allowed.

GENERAL INSTRUCTIONS:

This paper consists of Two (2) questions with a total of 50 marks.

ATTEMPT ALL QUESTIONS

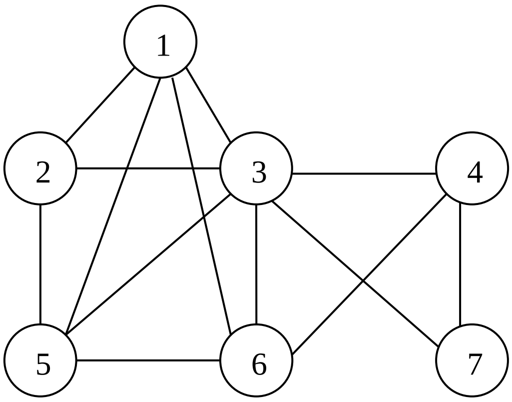
Name: ____

Student No: ____

Tutorial Time/Tutor: ___________________________________

QUESTION ONE (total: 30 marks): Graph and Heap

a) **(Total: 12 marks).** Consider the following undirected graph G(*V*, *E*), and a breadth-first-search algorithm to answer the following questions.



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

```
Mark all vertices new and set T = { }
Mark the start vertex v =old
insert (Q, v) // Q is a queue
 while Q is nonempty do
        x = dequeue (Q)
        for each vertex w in L[x] marked new do
            T = T ∪ {x,w}
            Mark w = old
            insert (Q,w)
```

(i) **(2 marks).** Give the adjacency list for the graph.

(ii) **(1 mark).** Give the maximum clique in the given graph.

(iii) **(4 marks).** Draw the breath-first-search tree of the graph. Assume the root of the tree is node 1, and vertices are put into the queue in increasing order when there is a tie.

(iv) **(5 marks).** Analyse the time complexity of **BFS_Tree_G(*V*, *E*)** to show that its time complexity is $O(|E| + |V|)$ if its input is in an adjacency list.

Answer:

(i) Adjacency list

(ii) Maximum clique:

(iii) BFS

(iv) Time complexity

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

```
HEAPIFY (A, i)

l ← LEFT_CHILD (i)
r ← RIGHT_CHILD (i)
if l ≤ heap_size[A] and A[ l ] < A[i]
        then a ← l
        else a ← i
if r ≤ heap_size[A] and A[r] < A[a]
        then a ← r
if a ≠ i
        then exchange A[i] with A[a]
        HEAPIFY (A, a)
```

(i) **(2 marks).** Is function HEAPIFY() a MIN_HEAPIFY() or a MAX_HEAPIFY() function? Justify your answer by explaining the lines in the code that correspond to a min-heap or a max-heap.

(ii) **(2 marks).** Modify the function to its opposite function, that is, if your answer to part (i) is a MIN_HEAPIFY modify it into a MAX_HEAPIFY. You need to give only the modified lines.

(iii) **(4 marks).** Explain why 1) the worst case running time of function HEAPIFY is $O(\log n)$ and 2) its best case is $O(1)$. **You are not required to give a formal proof for the time complexity.**

Answer:

(i)

(ii)

(iii) Worst case:

Best case:

c) **(Total: 10 marks).** Consider the following HEAPSORT algorithm.

HEAPSORT(*A*)
Input: Array $A[1\ldots n]$, $n = A.length$
Output: Sorted array $A[1\ldots n]$

```
BUILD-MAX-HEAP(A)
for  i = A.length downto 2
    do exchange A[1] with A[i]
       A.heap_size = A.heap_size - 1
       MAX-HEAPIFY(A, 1)
```

(i) **(3 marks).** Consider the following array $A = (7, 2, 9, 4, 3, 5, 1, 6)$. Show the content of array A after using function BUILD_MAX_HEAP (A). **Show your details to get partial credits.**

Hint. It is recommended that you first construct the max-heap in its binary tree. Then, convert the resulting tree into its array form.

(ii) **(4 marks).** Does the HEAPSORT() sort array A in *increasing* or in *decreasing* order? Justify your answer by **explaining how HEAPSORT() works**; thus an explanation based on only one line is not sufficient.

(iii) **(3 marks).** What is the running time of HEAPSORT() if initially array A is sorted in increasing order? Explain your answer.

<u>Answer:</u>

(i)

(ii)

(iii)

d) **(2 marks).** Explain one main advantage and one main disadvantage of using the leftist tree than the binary heap tree.

Answer:

Advantage:

Disadvantage:

END OF QUESTION ONE

QUESTION TWO (total: 20 marks): Greedy Algorithms

a) **(Total: 6 marks).** Consider a 0/1 Knapsack problem with a knapsack that can hold 50 units of weight, and the following item set.

	Items		
	A	**B**	**C**
Weight	10	20	30
Value	$80	$100	$150

(i) **(2 marks).** If the selection is greedy by weight, what items are selected and what is the total value of the selected items?

(ii) **(2 marks).** If the selection is greedy by value/weight, what items are selected and what is the total value of the selected items?

(iii) **(2 marks).** Does the item selection approach in (i) or (ii) produce optimal value? Justify your answer.

Answer:

(i)

(ii)

(iii)

b) **(Total: 14 marks).** Consider the following pseudocode for Dijkstra's algorithm.

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*.

```
1.   mark vertex u
2.   d[u] ← 0
3.   for each unmarked vertex  v ∈ V do
4.       if edge (u,v) exists then d[v] ← weight (u, v)
5.       else d[v] ← ∞
6.   while there exists an unmarked vertex do
7.       let v be an unmarked vertex such that d[v] is minimal
8.       mark vertex v
9.       for all edges (v, x) such that x is unmarked do
10.          if d[x] > d[v] + weight[v, x] then
11.              d[x] ←  d[v] + weight[v, x]
```

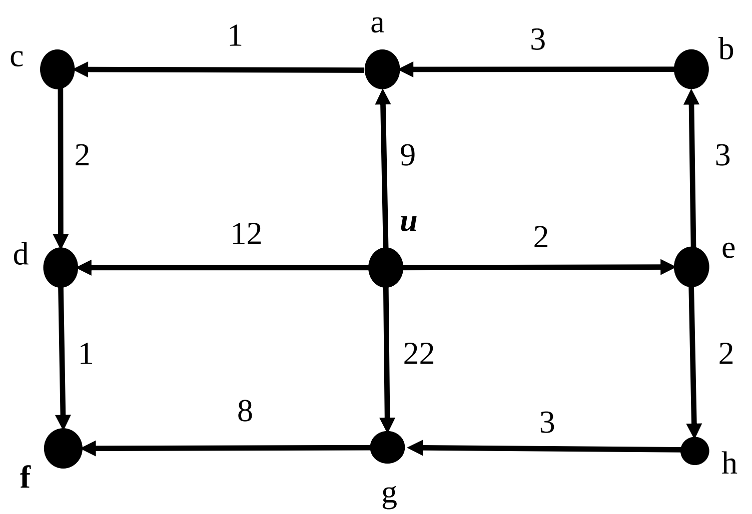
(i) **(1 mark).** Explain why Dijkstra's algorithm is a greedy approach.

(ii) **(1 mark).** Consider that the Dijkstra's algorithm is implemented using a min-heap in which every node contains each vertex of the graph and its *value*. What *value* is stored in the min-heap node?

(iii) **(2 mark).** What is the time complexity of Lines 3 to 5? Explain your answer.

(iv) **(8 marks).** Use Dijkstra's algorithm to find the shortest paths from vertex ***u*** of the following graph. The template for the solution is provided below.

(v) **(2 marks).** Show the shortest path to node **f** and its cost.



Answer:

(i)

(ii)

(iii)

(iv) Using Dijkstra's algorithm.

Step#	**Vertex to be marked**	**Distance to vertex**								
		u	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**
0	***u***									
1										
2										
3										
4										
5										
6										
7										
8										

(v) Shortest path to node f:

Cost:

END OF TEST PAPER