PAPER CODE NO.	EXAMINER : Prudence Wong
COMP108	DEPARTMENT : Computer Science Tel. No. 795-4257

UNIVERSITY OF LIVERPOOL

MAY 2007 EXAMINATIONS

Bachelor of Arts : Year 1
Bachelor of Science : Year 1
Bachelor of Science : Year 2
Master of Engineering : Year 1
No qualification aimed for: Year 1

ALGORITHMIC FOUNDATIONS

TIME ALLOWED : TWO hours

INSTRUCTIONS TO CANDIDATES

Answer **FOUR** questions.

If you attempt to answer more questions than the required number of questions (in any section), the marks awarded for the excess questions answered will be discarded (starting with your lowest mark).
All logarithms are to the base 2.

UNIVERSITY OF LIVERPOOL

Question 1

1A. State (without proof) the *order of magnitude*, in the form of $O(f(n))$, of the following functions. Use the simplest $f(n)$ possible in your answers.

I. $n^2 + n\log^3 n + 5n + 10$

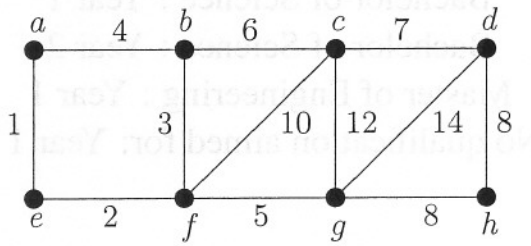
II. $6\sqrt{n^3} + \sqrt{n^5} + 7n^2 + n$

III. $5\log n + \sqrt{n} + \log^3 n + 1$

IV. $3^n + n^4 + 2^n + n^2$

[4 marks]

1B. Consider the following graph G. The label of an edge is the cost of the edge.



Using *Kruskal's* algorithm, draw a *minimum spanning tree* (MST) of the graph G.

Write down the order in which the edges are selected.

Is the MST drawn unique? (i.e., is it the one and only MST for the graph?)

[8 marks]

1C. Referring to the same graph above, find the shortest paths from the vertex a to *all* other vertices in the graph G using *Dijkstra's* algorithm.

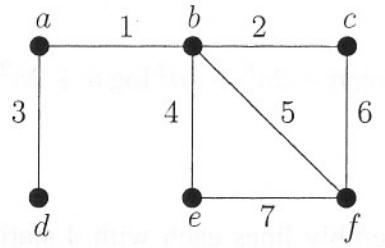
Show the changes of the labels of the vertices step by step and give the order in which edges are selected.

[13 marks]

UNIVERSITY OF LIVERPOOL

Question 2

2A. Consider the following graph G. The label of an edge is the edge number. Give the incidence matrix of the graph G.



[4 marks]

2B. The time complexity of the merge sort algorithm to sort n numbers, denoted by $T(n)$, can be described by the following recurrence.

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 2T(n/2) + n & \text{if } n > 1. \end{cases}$$

Show that $T(n) = O(n \log n)$ by the *substitution method*. (Hint: show that $T(n) \leq 2n \log n$ for $n \geq 2$ by Mathematical Induction.)

[8 marks]

2C. A length-n sequence S of characters $S[0], S[1], \cdots, S[n-1]$ is called a palindrome if S is the same as its reverse.
For example, CIVIC, DAD, NOON, RADAR are all palindromes.

Design and write a pseudo code algorithm to determine if a sequence S of characters is a palindrome or not.

What is the worst case time complexity of your algorithm (in big-O notation)? Explain briefly.

[13 marks]

UNIVERSITY OF LIVERPOOL

Question 3

3A. **I.** Briefly describe the idea of the *greedy* method.

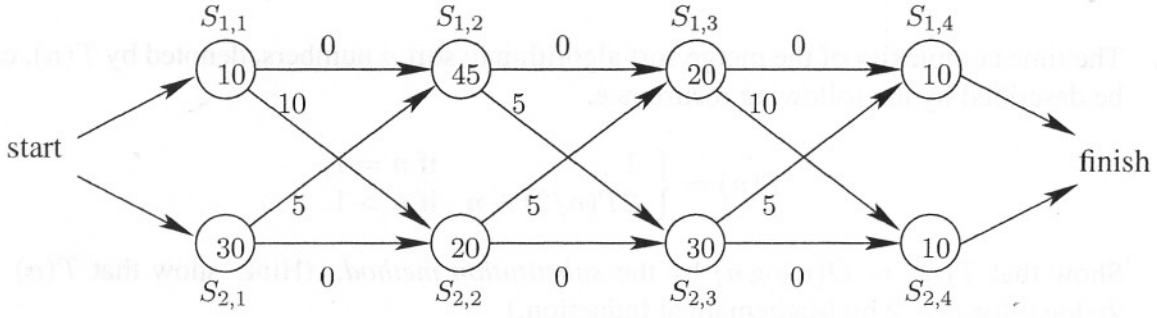
II. Briefly describe the idea of the *divide-and-conquer* method.

[4 marks]

3B. Show that the function $n^3 \log n + 2n^3 + 3n^2 \log n + 2n^2 + 5n + 2$ is $O(n^3 \log n)$.

[8 marks]

3C. Suppose there are two assembly lines each with 4 stations, $S_{i,j}$. The assembly time is given in the circle representing the station and the transfer time is given next to the arrow from one station to another.



I. Using dynamic programming, fill in the table of the minimum time $f_i[j]$ needed to get through station $S_{i,j}$. Show all the intermediate steps in computing these values.

j	$f_1[j]$	$f_2[j]$
1	...	...
2	...	...
3	...	...
4	...	...

II. What is the minimum time f^* needed to get through the assembly line?

III. Which stations should be chosen?

[13 marks]

Question 4

4A. Which of the following problems is/are NP-complete problem(s)?

I. Finding minimum spanning tree (MST) in a weighted undirected graph.

II. Vertex Cover Problem.

III. 0/1 Knapsack problem.

IV. Finding the n-th Fibonacci number.

[4 marks]

4B. **I.** Describe what a *decision problem* is and what an *optimisation problem* is.

II. An *optimisation* problem can be turned into a *decision* problem if we add a parameter k; and then ask whether the optimal value in the optimisation problem is at most or at least k.

State the decision version of the following optimisation problems:

a. Given an undirected graph G, find the minimum number of colours that is needed to colour the edges in G such that no two edges with the same colour share a common endpoint.

b. Given an undirected graph G, find the maximum number of vertices such that for any two such vertices, there is no edge connecting them.

[8 marks]

4C. Consider the following recursive algorithm to compute Fibonacci numbers.

Algorithm $F(n)$
  if $0 \leq n \leq 1$ **then**
    result = 1
  else
    if $n > 1$ **then**
      result = $F(n-1) + F(n-2)$
  return result

Suppose $f(n)$ denote the time complexity of the above algorithm. $f(n)$ satisfies the following recurrence

$$f(n) = \begin{cases} 1 & \text{if } n \leq 1, \\ f(n-1) + f(n-2) + 1 & \text{if } n > 1. \end{cases}$$

I. Show that $f(n) > 2f(n-2)$.

II. Using the *iterative method*, show that $f(n)$ is exponential in n.

III. The above algorithm is not efficient. Design and write the pseudo code of a faster (non-recursive) algorithm using the concept of dynamic programming.

What is the time complexity of the faster algorithm (in big-O notation)?

[13 marks]

UNIVERSITY OF LIVERPOOL

Question 5

5A. What are the *time complexities* (in big-O notation) of the following pseudo codes?

I.

$s = 0, i = 1$
while $i <= n$ **do**
begin
  $j = 1$
  while $j <= m$ **do**
  begin
    $s = s + i * j$
    $j = j + 1$
  end
  $i = i + 1$
end
Output s

II.

$i = 1, count = 0$
while $i < n$ **do**
begin
  $i = 2 * i$
  $count = count + 1$
end
Output $count$

[4 marks]

5B. Suppose we are given n numbers $A[0], A[1], \cdots, A[n-1]$. Write a pseudo code of an algorithm to sort the numbers in *descending* order (from the largest to the smallest).

What is the name of your sorting algorithm?

What is the time complexity (in big-O notation) of your algorithm?

[8 marks]

5C. **I.** In the Traveling Salesman problem, we are given a collection of cities and the cost of travel between each pair of them. The problem is to find the cheapest way of visiting all of the cities and returning to the starting point.

Describe an exhaustive search algorithm to find the cheapest such path.

How many paths you have to check to find the cheapest path? Explain briefly.

Does it take polynomial time or exponential time to find the solution?

II. Describe the conditions to determine whether an undirected graph has an Euler path (i.e., a path visiting every edge exactly once), and Euler circuit (starts and ends at the same vertex).

Does it take polynomial time or exponential time to check the conditions?

[13 marks]