TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS — 2009

END-YEAR

COMP303
Design and Analysis of Algorithms

Time Allowed: 3 Hours

Instructions:

- *Read each question carefully before attempting it.*
- This examination will be marked out of **180** marks.
- Answer all questions.
- You may answer the questions in any order. Make sure you clearly identify the question you are answering.
- Non-electronic foreign language-English dictionaries are permitted.
- Reference material, *calculators*, use of mobile phones, laptop computers, PDAs or other electronic devices is NOT PERMITTED.

Questions	Marks
1. Basic Algorithm Analysis and Verification	[30]
2. Divide and Conquer	[30]
3. Greedy Algorithms	[30]
4. Dynamic Programming	[30]
5. Graphs	[20]
6. Computability and Complexity	[15]
7. Various Topics	[25]

The following definitions are provided for your convenience. You may find it useful to tear off this front page of the paper.

Asymptotic notation:

$$
\begin{aligned}
O(g(n)) &= \{f(n) \mid (\exists d)(\mathbf{aa}\, n)[0 \leq f(n) \leq d.g(n)]\} \\
\Omega(g(n)) &= \{f(n) \mid (\exists c > 0)(\mathbf{aa}\, n)[f(n) \geq c.g(n) \geq 0]\} \\
\Theta(g(n)) &= \{f(n) \mid (\exists c > 0, d)(\mathbf{aa}\, n)[0 \leq c.g(n) \leq f(n) \leq d.g(n)]\}
\end{aligned}
$$

Master Theorem: Let $T(n)$ be defined by the recurrence $T(n) = aT(n/b) + f(n)$.
Let $\alpha = \log_b a$.

1. If $(\exists \epsilon > 0)[f(n) \in O(n^{\alpha-\epsilon})]$ then $T(n) \in \Theta(n^\alpha)$.
2. If $f(n) \in \Theta(n^\alpha)$ then $T(n) \in \Theta(n^\alpha \log n)$.
3. If $(\exists \epsilon > 0)[f(n) \in \Omega(n^{\alpha+\epsilon})]$ and $(\exists c < 1)(\mathbf{aa}\, n)[a.f(n/b) \leq c.f(n)]$ then $T(n) \in \Theta(f(n))$.

Logarithms:

$$\log_a x = y \text{ if and only if } a^y = x$$
$$\log_a x = \log_b x \div \log_b a$$

Question 1. Basic Algorithm Analysis and Verification [30 marks]

(a) Using the definitions for O, Ω, and Θ given on the front page of this paper, show that:

(i) [4 marks] $n^3 + n^2 \in O(n^3)$,

(ii) [4 marks] $n^2 + n \in \Theta(2n^2)$,

(iii) [4 marks] $\Omega(n^2) \subseteq \Omega(kn)$, for any positive integer k.

(b)

(i) [5 marks] Write a specification of the problem of finding the smallest and largest elements of a sequence. The input to the problem is a sequence s, and the output is two values, *small* and *large*, where *small* is the smallest, and *large* is the largest.

(ii) [4 marks] Write pseudo-code describing an algorithm that uses a loop to solve this problem.

(iii) [4 marks] Give a precise asymptotic running time (Θ bound) for your algorithm. Justify your answer. (You do not need to provide a complete proof.)

(iv) [5 marks] Your algorithm should contain a loop. Give a loop invariant that could be used to prove this algorithm is correct. (You do not need to provide a complete proof.)

Question 2. Divide and Conquer [30 marks]

(a) [5 marks] Write pseudocode to define the basic structure of a typical divide-and-conquer algorithm. Explain the components of your scheme.

(b) This question is about the problem of finding the *median* of an unordered sequence of integers. If a sequence s has length $2n$ or $2n + 1$, then the median is nth largest number.

(i) [3 marks] State precisely the relationship between the input and output for this problem.

(ii) [14 marks] Write an $O(n)$ divide-and-conquer algorithm to solve this problem.

(iii) [8 marks] Use the *assumption-requirement technique* to show that your algorithm is correct.

Question 3. Greedy Algorithms [30 marks]

(a) Suppose on a particular day you have n activities, each with a fixed starting time s_i and finishing time f_i . In general, the times for some activities will overlap, so it won't be possible to schedule all activities, but you want to do as many of them as possible.

(i) [3 marks] Describe precisely the *feasible* solutions to this problem.

The following algorithm is intended to solve this scheduling problem, returning the largest set M of activities that can be completed.

```
schedule(s, f)
    M ← {1}
    j ← 1
    for i ← 2 to n
        if f_j ≤ s_i then
            M ← M ∪ {i}
            j ← i
```

(ii) [3 marks] Describe the running-time of this algorithm, using O, Θ, or Ω. Justify your answer.

(iii) [4 marks] Describe an example input for which this algorithm fails to find the largest set of activities that can be completed. (You may use a diagram.)

(iv) [12 marks] Give an algorithm for this problem that finds the optimal solution for all inputs.

(b)

(i) [4 marks] Describe a variation of the scheduling problem just given. Your variation should be such that it *cannot* be solved by a greedy algorithm.

(ii) [4 marks] Explain why the problem you just described cannot be solved by a greedy algorithm.

Question 4. Dynamic Programming [30 marks]

Some time back, you helped a group of friends who were doing simulations for a computation-intensive investment company, and they've come back to you with a new problem. They're looking at n consecutive days of a given stock, at some point in the past. The days are numbered $i = 1, 2, ..., n$; for each day i, they have a price *p(i)* per share for the stock on that day.

For certain (possibly large) values of k, they want to study what they call *k-shot strategies*. A k-shot strategy is a collection of m pairs of days $(b_1, s_1), ..., (b_m, s_m)$, where $0 \leq m \leq k$ and

$$1 \leq b_1 < s_1 < b_2 < s_2 ... < b_m < s_m \leq n.$$

We view these as a set of up to k nonoverlapping intervals, during each of which the investors buy 1,000 shares of the stock (on day b_i) and then sell it (on day s_i). The *return* of a given k-shot strategy is simply the profit obtained from the m buy-sell transactions, namely,

$$1,000 \sum_{i=1}^{m} p(s_i) - p(b_i).$$

The investors want to assess the value of k-shot strategies by running simulations on their n-day trace of the stock price.

(a) [15 marks] Design and write down the pseudocode for an efficient algorithm that determines, given the sequence of prices, the k-shot strategy with the maximum possible return. Since k may be relatively large in these simulations, your running time should be polynomial in both n and k; it should not contain k in the exponent.

(b) [5 marks] Show the asymptotic complexity of your algorithm.

(c) [10 marks] Show that your algorithm correctly gives the optimal answer.

Question 5. Graphs [20 marks]

We have a connected graph with no loops $G = (V, E)$, and a specific vertex $u \in V$. Suppose we compute a depth-first search tree rooted at u, and obtain a tree T that includes all nodes of G. Suppose we then compute a breadth-first search tree rooted at u, and obtain the same tree T.

(a) [10 marks] Prove that G is acyclic.

(b) [10 marks] Prove that $G = T$. In other words, if T is both a depth-first search tree and a breadth-first search tree rooted at u, then G cannot contain any edges that do not belong to T.

Question 6. Computability and Complexity [15 marks]

(a) [3 marks] What is the difference between *a problem* and *an algorithm*?

(b) Define and explain in plain English the classes

(i) [2 marks] *P*,

(ii) [2 marks] *NP*,

(iii) [2 marks] *NP-Complete*, and

(iv) [2 marks] *NP-Hard*.

(c) You are given two problems A and B, and told that A is NP-complete. How would you:

(i) [2 marks] show that B is *NP-Hard*?

(ii) [2 marks] show that B is *NP-Complete*?

Question 7. Various Topics [25 marks]

(a) [5 marks] Describe a problem that would be suitable for a Monte Carlo algorithm and explain how randomness will help you make the algorithm more efficient.

(b) [5 marks] Describe a problem that would be suitable for a Las Vegas algorithm and explain how randomness will help you make the algorithm more efficient.

(c) [5 marks] Give an example of an approximation algorithm and explain how does the approximation help you solve the problem more efficiently.

(d) [2 marks] Define what is meant by a *convolution*.

(e) [8 marks] Describe an efficient *Fast Fourier Transform* algorithm.

****