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

EXAMINATIONS — 2011

END-YEAR

COMP 303
Design and Analysis of Algorithms

Time Allowed: Three 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 or other electronic devices is NOT PERMITTED.

Questions	Marks
1. Asymptotic Analysis	[25]
2. Divide and Conquer	[40]
3. Greedy Algorithms	[35]
4. Graphs	[15]
5. Dynamic Programming	[35]
6. Various Topics	[30]

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:

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

Question 1. Asymptotic Analysis [25 marks]

(a) [10 marks] Compare the following sets (perhaps by drawing Venn diagrams):

$O(1), O(2), O(n), \Omega(1), \Theta(1), \Theta(n), \Theta(n^2), \Theta(n(n+1)), \Theta(logn), \Theta(\sqrt{n})$

(b) [15 marks] Assume you have functions f and g such that $f(n)$ is $O(g(n))$. For each of the following statements, decide whether you think it is true or false, and give a proof or counterexample.

1. $log_2(f(n))$ is $O(log_2(g(n)))$.
2. $2^{f(n)}$ is $O(2^{g(n)})$.
3. $f(n)^2$ is $O(g(n)^2)$.

Question 2. Divide and Conquer [40 marks]

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

(b) [20 marks] Consider an n-node complete binary tree T, where $n = 2^d - 1$ for some d. Each node v of T is labelled with a real number x_v. You may assume that the real numbers labelling the nodes are all distinct. A node v of T is a *local minimum* if the label x_v is less than the label x_w for all nodes w that are joined to v by an edge.

You are given such a complete binary tree T, but the labelling is only specified in the following way: for each node v, you can determine the value x_v by *probing* the node v. Assume that this *probing* operation can be potentially very expensive. Show how to find a local minimum of T using only $O(logn)$ *probes* to the nodes of T.

State your solution (presumably a Divide and Conquer algorithm) following the template from the lectures. *Prove your algorithm is correct following the lectures.*

(c) [15 marks] Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for sufficiently small n. Make your bounds as tight as possible, and justify your answers.

1. $T(n) = 4T(n/3) + n\ logn$.
2. $T(n) = 2T(n/2) + n/logn$.
3. $T(n) = \sqrt{n}T(\sqrt{n}) + n$.

Question 3. Greedy Algorithms [35 marks]

Consider the problem of making change of n cents using the fewest number of coints. Assume that each coin's value is an integer.

(a) [10 marks] Describe a greedy algorithm to make change consisting of 25c, 10c, 5c, and 1c coins. Prove that your algorithm yields an optimal solution. Follow the lecture slides style of presenting and proving greedy algorithms.

(b) [10 marks] Suppose that the available coins are in the denominations that are powers of c, i.e., the denominations are $c^0, c^1, \ldots, c^k$ for some integers $c > 1$ and $k > 1$. Show that the greedy algorithm always yields an optimal solution.

(c) [5 marks] Give a set of coin denominations for which the greedy algorithm does not yield an optimal solution. Your set should include a 1c coin so that there is a solution for every value of n.

(d) [10 marks] Give an $O(nk)$-time algorithm that makes changes for any set of k different coin denominations, assuming that one of the coins is 1c. Prove that the time your algorithm takes is as required.

Question 4. Graphs [15 marks]

(a) [15 marks] There are two types of professional wrestlers: "babyfaces" ("good guys") and "heels" ("bad guys"). Between any pair of professional wrestlers, there may or may not be a rivalry. Suppose we have n professional wrestlers and we have a list of r pairs of wrestlers for which there are rivalries. Given an $O(n + r)$- time algorithm that determines whether it is possible to designate some of the wrestlers as babyfaces and the remainder as heels such that each rivalry is between a babyface and a heel. If it is possible to perform such a designation, your algorithm should produce it.

Question 5. Dynamic Programming [35 marks]

A certain string-processing language allows a programmer to break a string into two pieces. Because this operation copies the string, it costs n time units to break a string of n characters into two pieces. Suppose a programmer wants to break a string into many pieces. The order in which the breaks occur can affect the total amount of time used. For example, suppose that the programmer wants to break a 20-character string after characters 2, 8, and 10 (numbering the characters in ascending order from the left-hand end, starting from 1). If she programs the breaks to occur in left-to-right order, then the first break costs 20 time units, the second break costs 18 time units (breaking the string from characters 3 to 20 at character 8), and the third break costs 12 time units, totalling 50 time units. If she programs the breaks to occur in right-to-left order, however, then the first break costs 20 time units, the second break costs 10 time units, and the third break costs 8 time units, totalling 38 time units. In yet another order, she could break first at 8 (costing 20), then break the left piece at 2 (costing 8), and finally the right piece at 10 (costing 12), for a total cost of 40.

(a) [10 marks] Design an algorithm that, given the numbers of characters after which to break, determines a least-cost way to sequence those breaks. More formally, given a string S with n characters and an array $L[1 \dots m]$ containing the break points, compute the lowest cost for a sequence of breaks, along with a sequence of breaks that achieves this cost.

(b) [10 marks] Prove that this problem has an *optimal substructure* property.

(c) [10 marks] Prove that your algorithm is correct.

(d) [5 marks] Show what complexity your algorithm has for both time and space.

Question 6. Various Topics [30 marks]

(a) [10 marks] Prove that the problem of sorting n elements *based on comparisons* is $\Theta(n\ logn)$.

(b) [10 marks] What is a *Tutte Polynomial* and how can *pruning* and *bounding* help with their computation?

(c) [10 marks] State your favourite *nonblocking* data structure and explain how it works and what makes it *not block*.

****