Victoria University of Wellington

DEGREE EXAMINATIONS — 1999

MID-YEAR

COMP 303

DESIGN AND ANALYSIS
OF ALGORITHMS

Time Allowed: 3 Hours

Instructions: There are 5 questions of varying weights, totalling 180 marks.
Answer all questions, allowing about one minute per mark.
Make sure your answers are clear, complete and to the point.

No reference material is allowed.
No calculators are allowed.
Foreign language dictionaries *are* permitted.

Useful results

Asymptotic notation

$$\begin{aligned} O(f(n)) &= \{g \mid (\exists c)(\textbf{aa}\ n)[g(n) \leq c.f(n)]\} \\ \Omega(f(n)) &= \{g \mid (\exists c)(\textbf{aa}\ n)[g(n) \geq c.f(n)]\} \\ \Theta(f(n)) &= O(f(n)) \cap \Omega(f(n)) \end{aligned}$$

de l'Hôpital's Rule

Suppose $\lim_{n\to\infty} f(n) = a$ and $\lim_{n\to\infty} g(n) = b$, where a and b may be zero, non-zero and finite, or infinite (∞).

If a and b are both zero or both infinite, then

$$\lim_{n\to\infty} f(n)/g(n) = \lim_{n\to\infty} f'(n)/g'(n).$$

Otherwise,

$$\lim_{n\to\infty} f(n)/g(n) = a/b.$$

Master theorem

Let $T(n)$ be defined by the recurrence $T(n) = aT(n/b) + f(n)$. Let $\alpha = \log_b a$. Let ϵ be a small positive constant.

1. If $f(n) \in O(n^{\alpha-\epsilon})$, 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 $f(n) \in \Omega(n^{\alpha+\epsilon})$, and if $a.f(n/b) \leq c.f(n)$ for some $c < 1$ and almost all n, then $T(n) \in \Theta(f(n))$.

Question 1. [30 marks]

(a) [6 marks] Explain the use of the asymptotic notations O, Ω, and Θ in analysis of algorithms and problems.

(b) [6 marks] Explain how the *principle of invariance* justifies our use of asymptotic notation to analyse algorithms.

(c) [6 marks] Why might asymptotic notation sometimes be inadequate for comparing algorithms?

(d) [12 marks] Using the definitions for O, Ω, and Θ given on the front page of this paper, show that

$$2n^2 \in \Theta(n^2)$$

Question 2. [34 marks]

(a) [8 marks] Describe, in informal terms, the complexity classes $\mathcal{P}$, $\mathcal{NP}$, and $\mathcal{NPC}$.

(b) [6 marks] For each of the following properties, state whether the property is known to be true; known to be false; unknown but probably true; unknown but probably false. Justify each answer.

(i) $\mathcal{P} \subseteq \mathcal{NP}$

(ii) $\mathcal{NP} \subseteq \mathcal{P}$

(iii) $\mathcal{NPC} \subset \mathcal{NP}$

(c) [20 marks] The **Travelling Salesperson** problem is defined as follows.

Given a connected, directed graph G with weighted edges, find a way of visiting every node of the graph by traversing its edges in a way that minimizes the total weight of the edges traversed.

(i) Compare and contrast this problem with the **Hamiltonian Cycle** problem discussed in lectures.

(ii) Do you think the **Travelling Salesperson** problem has a polynomial solution? Explain why or why not.

(iii) Do you think **Travelling Salesperson** is $\mathcal{NP}$-**Hard**? Explain why or why not.

Question 3. [36 marks]

Given a finite set X, a **cover** for X is a set C of sets, such that every member of X belongs to at least one of the sets in C; that is, so that $X = \bigcup_{S \in C} S$. The size of a cover is the number of sets in the set C, that is, $|C|$.

For example, if $X = \{1, 2, 3, 4\}$, then a possible cover for X is the set of sets $C = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{1, 4\}, \{1, 3, 4\}\}$, with size 5.

The **minimum cover** problem is to find, given a set X and a cover C, the smallest subset M of C that is still a cover for X. In the above example, the subset $M = \{\{1, 2\}, \{1, 3, 4\}\}$ is a minimum cover for X and C, since $M \subseteq C$, M covers X, and no subset of C with fewer than 2 elements covers X. Other minimum covers for the same problem instance are $\{\{1, 2\}, \{3, 4\}\}$ and $\{\{2, 3\}, \{1, 4\}\}$.

The following algorithm has been proposed to solve minimum cover:

MinCov(X, C)
  $U \leftarrow X$
  $M \leftarrow \emptyset$
  while $U \neq \emptyset$ do
    select $S \in C - M$ that maximises $|S \cap U|$
    $U \leftarrow U - S$
    $M \leftarrow M \cup \{S\}$
  return M

(a) [4 marks] Which of the categories of algorithms discussed in this course does this algorithm fall into? Explain why.

(b) [12 marks] Assume you are given a naive implementation of this algorithm. That is, no special data structures are used to make any of the set operations more efficient than linear scans. What would the asymptotic running time of this implementation be? Give your answer in terms of n and m, where $|X| = n$ and $|C| = m$. Explain your answer. Note that m may be smaller than, equal to, or larger than n.

(c) [8 marks] Briefly describe an implementation that is more efficient, and justify why it is more efficient.

(d) [12 marks]

Is the given **MinCov** algorithm a correct solution for the minimum cover problem? Explain your answer.

Question 4. [44 marks]

The problem of **making optimal change** is about finding the minimal number of coins whose total value is exactly the amount of change required.

Let the set of available coins be described by the sequence C, such that C_i is the value of the ith largest coin. If there are n distinct coins, this sequence has length n. For example, the coins that are legal tender in New Zealand would be described by the sequence $C = \langle 200, 100, 50, 20, 10, 5 \rangle$, for which $n = 6$.

Let V be the total amount of change required. You may assume that there is an infinite supply of each value of coin.

The solution S is a sequence of length n, giving the number of each denomination of coins required. A solution is *acceptable* if $V = \Sigma_{i=1}^{n} S_i C_i$, and *optimal* if $\Sigma_{i=1}^{n} S_i$ is minimized.

(a) [10 marks] Describe a *greedy algorithm* for making optimal change, given V and C as inputs. Your algorithm should produce a correct, optimal solution for the New Zealand coins, as long as V is a multiple of 5.

(b) [6 marks] Give values for C and V for which your algorithm produces an acceptable but suboptimal solution, and state what the optimal solution for that problem is.

(c) [12 marks] Design a *dynamic programming* algorithm for making optimal change, and show how it correctly handles the situation you gave in (b).

(d) [8 marks] Explain why your dynamic programming algorithm is correct (a complete proof is not necessary).

(e) [8 marks] What is the asymptotic running time of your algorithm? Justify your answer. A rigorous proof is *not* required, but you should state any additional assumptions you make.

Question 5. [36 marks]

(a) [6 marks] Divide and conquer algorithms deal with subproblems. What properties must these subproblems have for a divide and conquer algorithm to be efficient? Under what conditions is it *not* appropriate to use divide and conquer?

(b) [12 marks] Give the general structure of a divide and conquer algorithm. In terms of this general structure, state what needs to be proved to show that the algorithm is correct.

(c) [18 marks] Assume you are given two sorted lists of length n:

$$u = \langle u_1, u_2, \ldots, u_n \rangle$$
$$v = \langle v_1, v_2, \ldots, v_n \rangle$$

Devise an efficient *divide and conquer* algorithm for finding the *median* of the combined lists; that is, to find the nth largest of the $2n$ elements. Do *not* merge the two lists!

Hint Recall that the median of a single sorted list can be determined in constant time. Let u_m be the median of u, and v_m be the median of v. What can you say about the median of the combined lists if $u_m = v_m$? if $u_m < v_m$? if $u_m > v_m$?

****