VICTORIA UNIVERSITY OF WELLINGTON

Te Whare Wānanga o te Ūpoko o te Ika a Māui

EXAMINATIONS — 2004

MID-YEAR

COMP 303

Design and Analysis of Algorithms

Time Allowed: 3 Hours

Instructions: There are 5 questions, worth 20 marks each.
Answer all questions.

Calculators and other electronic devices are **not** permitted.
Printed foreign language dictionaries **are** permitted.

Question 1.

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

(a) Explain the three definitions given above. State under what circumstances each of O, Ω, and Θ is used. [3 marks]

(b) Explain why asymptotic notation is sometimes inadequate for comparing algorithms. [2 marks]

(c) Using the definitions above, show that:

$$n^2 + 4n \in \Theta(2n^2)$$

[5 marks]

(d) 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 $T(n) \in \Theta(f(n))$.

For each of the following recurrence relations, give the asymptotic complexity (Θ bound) of $T(n)$. Justify your answers using either induction or the Master Theorem (given above), as appropriate.

(i) $T(n) = \begin{cases} 1, & \text{if } n = 0 \\ T(n-1) + n^2, & \text{otherwise} \end{cases}$

(ii) $T(n) = \begin{cases} 1, & \text{if } n = 0 \\ T(\lceil \frac{n}{3} \rceil) + n, & \text{otherwise} \end{cases}$

(iii) $T(n) = \begin{cases} 1, & \text{if } n = 0 \\ 4T(\lceil \frac{n}{2} \rceil) + n \log n, & \text{otherwise} \end{cases}$ [10 marks]

Question 2.

(a) Outline the structure of a typical divide and conquer algorithm. [4 marks]

(b) Outline the structure of a proof of correctness for a typical divide and conquer algorithm. [4 marks]

(c) Suppose that you are given a sequence $s = \langle s_1, s_2, \ldots, s_n \rangle$ that contains n distinct integers (both positive and negative) **in increasing order**. You want to determine whether or not there is a position i such that $s_i = i$.

For example, given the sequence $s = \{-4, -1, 2, 4, 7\}$, $i = 4$ is such a position.

(i) Design an efficient **divide and conquer** algorithm that will solve the problem.

(ii) Outline an argument that your algorithm is correct. In your answer, you should refer to assumptions and requirements for the subproblems.

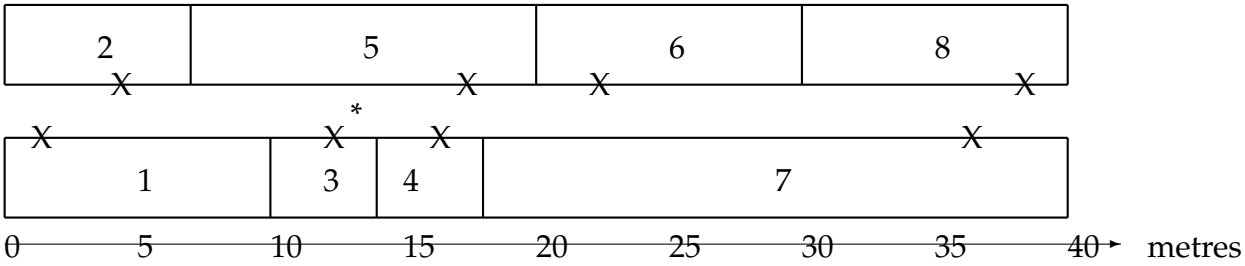
(iii) Give a recurrence relation that describes the worst-case running time of your algorithm. Explain how each part of your recurrence relation relates to the components of your algorithm.

(iv) Give an asymptotic solution for your recurrence relation. [12 marks]

Question 3.

(a) You are a programmer working for a firm that specializes in the installation of fire extinguishers in office buildings. Regulations require that the maximum distance from any office door to the nearest fire extinguisher is ten metres; but because fire extingishers are expensive, the company wishes to minimize the number used in any given corridor. You may assume that offices are laid out along straight corridors, so that for the purposes of this problem, a layout may be described as an ordered sequence of points on a number line.

For example, the diagram below shows eight offices (numbered 1–8); each door is marked with X. The layout may be described by the sequence $\langle 1, 4, 12, 16, 17, 22, 36, 38 \rangle$.



An extinguisher at position 13 (marked * on the diagram) could service offices 2, 3, 4, 5, and 6, but no others (we assume the width of the corridor is negligible).

(i) Show that the problem has the **optimal substructure** property. Be sure to define carefully what it means for one problem instance to be a subproblem of another.

(ii) Design a **greedy** algorithm that will return positions for the minimum number of extinguishers that can service all the offices on a corridor.

(iii) Outline an argument that your algorithm is correct. [12 marks]

(b) Below are two algorithms for finding a minimal spanning tree $G' = (V', E')$ in a graph $G = (V, E)$.

Prim(V, E) returns (V', E')
  $V' \leftarrow \{\}$
  $E' \leftarrow \{\}$
  while $V' \neq V$
    $e \leftarrow$ shortest edge $x \rightarrow y$
      such that $x \in V'$ but $y \notin V'$
    $E' \leftarrow E' \cup \{e\}$
    $V' \leftarrow V' \cup \{y\}$

Kruskal(V, E) returns (V', E')
  $V' \leftarrow V$
  $E' \leftarrow \{\}$
  while $\#E' < \#V - 1$
    $e \leftarrow$ shortest edge $x \rightarrow y$ in E
    $E \leftarrow E - \{e\}$
    if no path from x to y in E'
      $E' \leftarrow E' \cup \{e\}$

(i) Briefly explain how Kruskal's algorithm may be efficiently implemented using a **Union-Find** data structure to manage disjoint, non-empty sets of nodes.

(ii) Let $n = \#V$ be the number of nodes, and $a = \#E$ be the number of edges, in the graph G. Give asymptotic running times for the two algorithms: justify your answer. [8 marks]

Question 4.

A third-year computer science student has reached the deadline for numerous assignments, but has only one twenty-four hour day left in which to do them. He knows there is not enough time to complete them all. The student knows what assignments have been set, how long each will take, and how many marks he will get for each assignment that he completes; he wants to determine which assignments to complete to gain the maximum possible marks.

The details are summarized in the following table, in which the n courses are numbered $i = 1, 2, 3, 4, 5$. For each course, the table specifies:

- $t[i]$: the time needed to complete each assignment;
- $m[i]$: the number of marks for each completed assignment;
- $k[i]$: the number of assignments to be completed.

The table also shows, for each course, the ratio of marks to hours ($m[i]/t[i]$). $T = 24$ is the total time available.

Course	i	$t[i]$ (hours)	$m[i]$ (marks)	$k[i]$	$m[i]/t[i]$ (marks per hour)
COMP 301	1	2	5	4	2.5
COMP 302	2	8	16	1	2.0
COMP 303	3	8	20	2	2.5
COMP 304	4	5	13	3	2.6
COMP 305	5	10	10	2	1.0

$T = 24$

(a) Define **acceptable solution** and **correct solution** for the general problem: that is, assuming that the values n and T and the arrays k, t, and m are inputs. [2 marks]

(b) Our student thinks he can select the assignments using a greedy algorithm: look at each assignment in decreasing order of marks per hour, doing it only if it can be completed in the available time.

Show that the greedy algorithm does **not** give the optimal solution for the problem instance described above. [3 marks]

(c) Show that the general problem has the **optimal substructure** property. [5 marks]

(d) Describe an appropriate **dynamic programming** algorithm for solving the general problem. [4 marks]

(e) Another student has realized that it is not necessary to complete a whole assignment to gain some marks. She has determined that the mark-to-time ratio remains the same even if an assignment is not completed: for example, by spending three hours on a six-hour, ten-mark assignment, she can gain five marks.

(i) List the assignments and partial assignments this student should work on to maximize her mark.

(ii) Describe a **greedy algorithm** that will solve this version of the general problem. [6 marks]

Question 5.

(a) Given an unordered sequence of integers, there is a $O(n)$ algorithm (using the divide-and-conquer strategy) for selecting the kth largest element (for arbitrary k). Explain why there can be no algorithm for solving this problem that is asymptotically better. [4 marks]

(b) Explain why *polynomial reductions* are of interest in analysing the complexity of problems. [4 marks]

(c) Describe, in informal terms, the complexity classes $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{NPC}$.

[3 marks]

(d) Suppose that $\mathcal{P} \neq \mathcal{NP}$. Draw a Venn diagram showing the following complexity classes:

(i) $\Theta(n)$
(ii) $O(n^2)$
(iii) $\Theta(n^2)$
(iv) $\mathcal{P}$
(v) $\mathcal{NP}$ [5 marks]

(e) State the important characteristics of **two** varieties of probabilistic algorithms.

[4 marks]

****