Victoria University of Wellington

DEGREE EXAMINATIONS — 1998

COMP 303

MID-YEAR

COMP 303

DESIGN AND ANALYSIS
OF ALGORITHMS

Time Allowed: 3 Hours

Instructions: There are 7 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.

Question 1. [30 marks]

(a) [4 marks] Divide and conquer algorithms deal with subproblems. What properties must these subproblems have for a divide and conquer algorithm to be efficient?

(b) [10 marks] Give an efficient divide and conquer algorithm for multiplying two n-digit numbers, assuming that the only available operations are multiplication of single digits, and addition of numbers.

(c) [10 marks] Give the general structure of the proof of a divide and conquer algorithm, and use it show your algorithm from part (b) is correct.

(d) [3 marks] Give a recurrence relation for the number of single-digit multiplications performed by your algorithm.

(e) [3 marks] Hence *guess* the asymptotic running time of your algorithm, stating any additional assumptions you make.

Question 2. [30 marks]

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

(a) [6 marks] Explain the above definitions of three kinds of asymptotic notation. Under what circumstances is each one used?

(b) [6 marks] Why do we use asymptotic notation in most algorithm analysis?

(c) [6 marks] Why might we sometimes need to consider more exact ways of expressing complexity?

(d) [12 marks] Given functions f, g and h such that $f \in \Theta(g)$ and $g \in \Theta(h)$, show that $h \in \Theta(f)$.

Question 3. [20 marks]

During a typical day, a student must schedule many activities (lectures, study, assignments, meals, sleeping, recreation). Most days, it isn't possible to complete all the activities planned for the day.

Suppose on a particular day you have n activities, each with a fixed stating 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.

The following algorithm claims 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 s_i ≥ f_j then
      M ← M ∪ {i}
      j ← i
```

(a) [3 marks] Which of the algorithm design techniques discussed in the course does this algorithm use?

(b) [12 marks] Sketch a proof of correctness for this algorithm. You do not need to complete the proof in detail, but should indicate how each correctness criterion for this class of algorithms apply to this problem.

(c) [5 marks] What additional assumption about the input is required for this algorithm to be correct?

Question 4. [32 marks]

An unlicensed busker has three acts: a juggling performance (J) takes two minutes, and earns him $6; a song (S) takes three minutes, and earns $15, while a comedy routine (C) takes two minutes and earns $8. In any case, an incomplete act earns nothing. The busker knows the police will arrive in ten minutes and move him on, and he wants to earn as much as possible in the meantime.

(a) [2 marks] What acts should the busker perform, and how much will he earn?

(b) [4 marks] A *greedy algorithm* won't give an optimal solution to this problem, but it comes close. Give the nearly-optimal solution a greedy algorithm would give for this problem, briefly justifying how the solution is reached.

(c) [10 marks] Give a *dynamic programming* algorithm that, given the act times, earnings and time limit as input, will determine the best acts to perform.

(d) [8 marks] Outline a proof that your algorithm is correct.

(e) [2 marks] The busker's friend has just a single act, which earns him five dollars a minute, no matter how long he performs the act. How much can the friend earn?

(f) [6 marks] Using the two buskers as an illustration, explain how the search space for one problem can often be bounded by solutions to a simpler problem.

Question 5. [18 marks]

Suppose you have designed an algorithm whose solution is described in terms of a set of n elements. Initially, the set is assumed to be partitioned into n subsets of 1 element each. The algorithm needs two operations: **compare**(e_1, e_2) returns true iff the elements e_1 and e_2 are in the same subset, and **merge**(e_1, e_2) merges the subsets containing elements e_1 and e_2 into a single subset.

(a) [7 marks] Describe an efficient data structure for your algorithm to use to represent the set of subsets.

(b) [7 marks] Describe how the **compare** and **merge** operations can be efficiently implemented using your data structure.

(c) [4 marks] Suppose the algorithm requires a total of c **compare/merge** operations. What is the best possible asymptotic cost of the algorithm?

Question 6. [30 marks]

Given a graph $G = (V, E)$, a **Hamiltonian Cycle** is a path through G that visits each node V once and only once, and returns to the starting node.

A path can be described as a subset P of the edges E.

(a) [6 marks] Express precisely the requirement that a path P is a Hamiltonian Cycle in graph $G = (V, E)$.

(b) [6 marks] Not every graph has a Hamiltonian Cycle. Exhibit connected graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ with the same nodes but differing by a single edge, such that G_1 has a Hamiltonian Cycle but G_2 does not.

(c) [6 marks] **HAMD**(G) is the problem of determining whether a graph has a Hamiltonian Cycle. We say this problem is $\mathcal{NP}$-*complete*. What does that mean?

(d) [12 marks] Show that the problem **HAM**(G) of *finding* a Hamiltonian Cycle (assuming one exists) is also $\mathcal{NP}$-complete.

Question 7. [20 marks]

Given an undirected graph $G = (V, E)$, with weight w_e associated with each edge $e \in E$, a *spanning tree* is a tree whose nodes are all the nodes of the graph, and whose edges are a subset of the edges of the graph. A *minimal spanning tree* is a spanning tree of least possible weight.

(a) [12 marks] Outline two greedy algorithms for finding minimal spanning trees. (You do not need to give exact pseudocode, nor a proof of correctness.)

(b) [8 marks] Compare the asymptotic running times of the two algorithms, considering how they behave on *sparse* and *dense* graphs.

****