Victoria University of Wellington

DEGREE EXAMINATIONS — 2000

MID-YEAR

COMP 303

DESIGN AND ANALYSIS
OF ALGORITHMS

Time Allowed: 3 Hours

Instructions: There are 4 questions, worth 25 marks each.
Answer all questions.
Make sure your answers are clear, complete and to the point.

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

Useful results

Asymptotic Notation

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

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. [25 marks]

Consider the following (rather silly) sorting algorithm:

```
1 sillysort(A, i, k)
2   if i < k
3     j ← ⌊(i+k)/2⌋
4     sillysort(A, i, j)
5     sillysort(A, j + 1, k)
6     if A[i] ≥ A[j + 1]
7       swap A[i] and A[j + 1]
8     Sort(A, i + 1, k)
```

There are two recursive calls on **sillysort** (lines 4 and 5). The call on **Sort** (line 8) is not a recursive call; it is a call on some other sorting algorithm (insertion sort, perhaps).

(a) [6 marks] Show that **sillysort**$(A, 1, n)$ will correctly sort an array A with n elements (array indexes start at 1). You may assume that **Sort**(A, x, y) correctly sorts the segment of A between index x and y inclusive.

(b) [6 marks] Suppose that **Sort** uses some sorting algorithm with $\Theta(n^2)$ complexity, such as insertion sort or selection sort.

(i) Write a recurrence relation for the asymptotic complexity of **sillysort**.

(ii) Solve your recurrence relation. You may assume that n is a power of 2, so that the floor operation may be ignored. You may appeal to the Master theorem, either rigorously proving all conditions, or as a first guess which is then checked by induction.

(c) [5 marks] Now suppose that **Sort** uses some sorting algorithm with $\Theta(n \log n)$ complexity, such as merge sort.

(i) Write a recurrence relation for the asymptotic complexity of **sillysort**.

(ii) Explain why it is not possible to use the Master method to solve your recurrence relation.

(d) [8 marks] Finally, suppose that **Sort** uses a recursive call on **sillysort** itself; that is, line 8 is replaced by

```
8       sillysort(A, i + 1, k)
```

State the complexity of **sillysort**. Justify your answer.

Question 2. [25 marks]

(a) [2 marks] Suppose we have analysed a sorting algorithm, and discovered that it does exactly $\frac{n(n+1)}{2}$ comparison operations for any input of size n. Give *different* O, Ω, and Θ bounds for the complexity of the algorithm.

(b) [3 marks] Explain how we usually justify ignoring constant factors in algorithm analysis. Describe circumstances that might require us to consider constant factors.

(c) [3 marks] Draw a Venn diagram to depict the following sets:

$$O(1), \Omega(1), \Theta(1), O(n)$$

(d) [5 marks] Using the definitions for O and Ω given on the front page of this paper, show that

$$f(n) \in O(g(n)) \quad \text{if and only if} \quad g(n) \in \Omega(f(n))$$

(e) [3 marks] Explain why *polynomial reductions* are of interest in analysing the complexity of problems.

(f) [3 marks] I claim there is some Problem A that is polynomially equivalent to the problem of deciding whether a graph has a Hamiltonian cycle. I also claim that I have implemented a $O(n^3)$ algorithm for solving Problem A. Say whether you believe me, and explain why or why not.

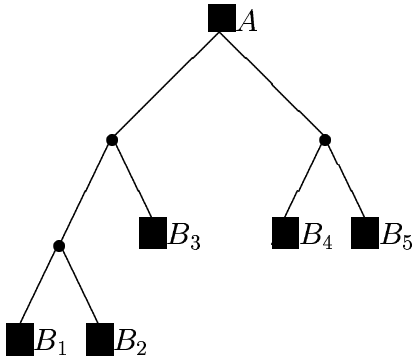
(g) [6 marks] Describe the important characteristics of **two** varieties of probabilistic algorithms.

Question 3. [25 marks]

The computer centre at a large organization is installing a network. They have a single large server computer (A) that is required to serve n client computers ($B_1, \ldots, B_n$). The network will be built using router nodes, each of which has a single connection to a parent node (which may be the server A or another router) and two connections to child nodes (which may be client computers, other routers, or one of each). This means that the network is a binary tree rooted on A, with router nodes as its internal nodes, and client computers at the leaves.

Each of the client computers B_i has an associated workload w_i. To get the maximum efficiency, clients with large workloads should be closer to the root than clients with smaller workloads. In fact, the aim is to produce the tree that minimizes $\Sigma_{i=1}^{n} w_i \times d_i$, where d_i is the distance (measured in network hops) from A to B_i.

For example, the diagram below shows a possible organization for five clients ($n = 5$), whose workloads are as shown in the table. (Computers are shown as squares, routers as small circles.) The table also shows the distance from the root for each client, in this arrangement. The total "cost" for this network (44) is found by summing the products $w_i \times d_i$ for each row of the table.



Client	w_i	d_i	$w_i \times d_i$
B_1	1	3	3
B_2	3	3	9
B_3	5	2	10
B_4	5	2	10
B_5	6	2	12

(a) [5 marks] Describe another problem with which you are familiar, that most closely resembles this problem. State which of the classes of algorithm we studied will be most suitable for this problem.

(b) [12 marks] Present an algorithm for solving this problem: given an array $w[1..n]$ as input, your algorithm should produce a description of the optimal network arrangement. You will need to design some suitable way of representing the tree; you should do this to make your algorithm as simple as possible.

(c) [8 marks] Outline a proof that your algorithm meets all the requirements for the problem.

Question 4. [25 marks]

A student has been overseas for the holidays, and is about to return to New Zealand. To pay his exorbitant fees, he has decided to pack his bag with drugs to take home and sell on the streets of Wellington. He has 3 bags of opium weighing 25 grams each, which he can sell for $500 profit each; 6 bags of cannabis leaf weighing 10 grams each, which he can sell for $100 profit each; and 2 bags of the recently banned drug tobacco, weighing 50 grams each, which he can sell for $900 profit each. He knows that the vigilant customs officers will catch him if he tries to carry more than 120 grams in total, so he will not take any more than that; he has to decide what to take with him, and what to leave behind for the lucky hotel cleaners.

Drug	Weight per bag	Profit per bag	Number of bags available
Opium	25g	$500	3
Cannabis	10g	$100	6
Tobacco	60g	$900	4

(a) [7 marks] Suppose our enterprising student is willing to open the bags, so he can take full or partly-full bags of any of the drugs.

(i) Briefly describe a greedy algorithm that will enable him to maximize his total profit. Describe the algorithm in general, assuming there are n drugs with weights $w_{1..n}$, profit margins $p_{1..n}$, and with $b_{1..n}$ bags available.

(ii) Show the result of your algorithm for the above problem instance.

(b) [10 marks] Before he can start the job of repackaging, his friend returns to the hotel. She tells him that, if the bags have been opened, the sniffer dogs will smell the drugs and he will be caught. The problem has thus changed, and he must now decide what bags to take given that he can only take whole bags.

(i) Show that the greedy algorithm will no longer work for this problem instance.

(ii) Describe an appropriate dynamic programming algorithm for solving the problem in general, and give the solution for this instance.

(iii) Briefly outline a proof that your algorithm is correct.

(iv) Give the asymptotic complexity of the algorithm. Justify your answer.

(c) [8 marks] An alternative to a dynamic programming technique for this problem is to explore the search space of all possible combinations of the different drug bags.

(i) Draw part of the search tree for this problem instance.

(ii) Discuss how the greedy algorithm from part (a) can be used to prune the search tree for this problem.

****