

EXAMINATIONS — 2010  
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, PDAs or other electronic devices is **NOT PERMITTED**.

<b>Questions</b>	<b>Marks</b>
1. Asymptotic Analysis	[30]
2. Divide and Conquer	[30]
3. Greedy Algorithms	[30]
4. Dynamic Programming	[30]
5. Graphs	[30]
6. Computability and Complexity	[10]
7. Various Topics	[20]

The following definitions are provided for your convenience. You may find it useful to tear off this front page of the paper.

**Asymptotic notation:**

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

**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  $(\exists c < 1)(\mathbf{a a n})[a.f(n/b) \leq c.f(n)]$  then  $T(n) \in \Theta(f(n))$ .

**Logarithms:**

$$\begin{aligned} \log_a x = y &\text{ if and only if } a^y = x \\ \log_a x &= \log_b x \div \log_b a \end{aligned}$$

## Question 1. Asymptotic Analysis

[30 marks]

(a) [3 marks] Find a theta notation for the number of times the statement  $x=x+1$  is executed and justify your answer.

```
i = 2
while (i < n) {
    i = i * i
    x = x + 1
}
```

(b) [12 marks] Using the definitions of asymptotic notations given in the front of this exam, prove that  $n! \in o(n^n)$  where  $o$  is an *asymptotically tight* bound:  $o(g(n)) = \{f(n) : \text{for any positive constant } c > 0, \text{ there exists a constant } n_0 > 0 \text{ such that } 0 \leq f(n) < cg(n) \text{ for all } n \geq n_0\}$ .

(c) [5 marks] What is wrong with the following “proof” that any algorithm has a run time that is  $O(n)$ ?

We must show that the time required for an input of size  $n$  is at most a constant times  $n$ .

**Basis Step.** Suppose that  $n = 1$ . If the algorithm takes  $C$  units of time for an input of size 1, the algorithm takes at most  $C * 1$  units of time. Thus, the assertion is true for  $n = 1$ .

**Inductive Step.** Assume that the time required for an input of size  $n$  is at most  $C'n$  and that the time for processing an additional item is  $C''$ . Let  $C$  be the maximum of  $C'$  and  $C''$ . Then the total time required for an input of size  $n + 1$  is at most:

$$C'n + C'' \leq Cn + C = C(n + 1)$$

The inductive step has been verified.

By induction, for input of size  $n$ , the time required is at most a constant times  $n$ . Therefore, the run time is  $O(n)$ .

(d) Using the definitions for  $O$ ,  $\Omega$ , and  $\Theta$  given on the front page of this paper, show that:

(i) [4 marks]  $6n + 1 \in \Theta(n)$ ,

(ii) [6 marks]  $\frac{(n^2 + \log_2(n))(n+1)}{n+n^2} \in \Theta(n)$

## Question 2. Divide and Conquer

[30 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) [10 marks] You are interested in analysing some hard-to-obtain data from two separate databases. Each database contains  $n$  numerical values - so there are  $2n$  values total - and you may assume that no two values are the same. You'd like to determine the median of this set of  $2n$  values, which we will define here to be the  $n$ th smallest value.

However, the only way you can access these values is through *queries* to the databases. In a single query, you can specify a value  $k$  to one of the two databases, and the chosen database will return the  $k$ th smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.

Give an algorithm that finds the median value using at most  $O(\log n)$  queries. State both the problem and algorithm using formal description of inputs and outputs and use pseudocode.

(c) [10 marks] Prove that your algorithm above is correct using the technique described in the lectures.

(d) [5 marks] Show the complexity of your algorithm, if necessary using the Master Method described on the front page of this exam.

### Question 3. Greedy Algorithms

[30 marks]

(a) Suppose that we are given  $n$  tasks each of which takes the same amount of time to complete. Suppose further that each task has a deadline by which it is supposed to finish and a penalty that is applied if the task does not finish by the deadline. The problem is to find a schedule of all of the tasks that minimises the total penalty.

(i) [10 marks] Develop a greedy algorithm to solve this problem, and *prove* that your algorithm is correct and optimal.

(ii) [5 marks] What is the worst-case time of your algorithm?

(b) [15 marks] Note that the following problem is such that no polynomial time algorithm is known. Develop a greedy algorithm for this problem, which may or may not always give an optimal solution. Analyse the worst-case time of your algorithm. Does your algorithm always yield an optimal solution? If so, prove it; if not, give example input for which your algorithm is not optimal. If your algorithm is optimal and runs in polynomial time, you are immediately famous and will get a guaranteed A+ in COMP303.

**Bin-Packing Problem.** Given  $n$  objects of sizes  $s_1, \dots, s_n$ , where  $0 < s_i \leq 1$ , find the smallest number of bins, each of capacity one, into which the objects can be packed.

## Question 4. Dynamic Programming

[30 marks]

(a) [10 marks] Write a dynamic programming algorithm to compute  $a^n$  based on the formulas. Show that your algorithm is correct and state and justify its asymptotic complexity.

$$\begin{aligned} a^n &= a^{n/2}a^{n/2}, & n \text{ even} \\ a^n &= a^{(n-1)/2}a^{(n-1)/2}a, & n \text{ odd} \end{aligned}$$

(b) Suppose it's nearing the end of the trimester and you're taking  $n$  courses, each with a final project that still has to be done. Each project will be graded on the following scale: It will be assigned an integer number on a scale of 1 to  $g > 1$ , higher numbers being better grades. Your goal, of course, is to maximise your average grade on the  $n$  projects.

You have a total of  $H > n$  hours in which to work on the  $n$  projects cumulatively, and you want to decide how to divide up this time. For simplicity, assume  $H$  is a positive integer, and you'll spend an integer number of hours on each project. To figure out how best to divide up your time, you've come up with a set of functions  $\{f_i : i = 1, 2, \dots, n\}$  (rough estimates, of course) for each of your  $n$  courses; if you spend  $h \leq H$  hours on the project for course  $i$ , you'll get a grade of  $f_i(h)$ . (You may assume that the functions  $f_i$  are *nondecreasing*: if  $h < h'$ , then  $f_i(h) \leq f_i(h')$ .)

So the problem is: Given these functions  $\{f_i\}$ , decide how many hours to spend on each project (in integer values only) so that your average grade, as computed according to the  $f_i$ , is as large as possible. In order to be efficient, the running time of your algorithm should be polynomial in  $n$ ,  $g$ , and  $H$ ; none of these quantities should appear as an exponent in your running time.

(i) [4 marks] Show that the problem has the **optimal substructure** property.

(ii) [6 marks] Write a requested algorithm that solves this problem.

(iii) [6 marks] Show that your algorithm is correct.

(iv) [4 marks] State and justify the asymptotic complexity of your algorithm.

## Question 5. Graphs

[30 marks]

- (a) [5 marks] Discuss at least two different graph representations and state the advantages and disadvantages of each.
- (b) [10 marks] State an algorithm that finds *connected components* for a graph  $A$  and show that it is correct.
- (c) [5 marks] Describe what are *implicit graphs* and how they can help you in solving problems.
- (d) [10 marks] Describe at least two techniques for speeding up graph traversal and show an example of a problem where you can use an implicit graph to represent it as well as how these two techniques can be applied.

## Question 6. Computability and Complexity

[10 marks]

- (a) [2 marks] What is the difference between *a problem* and *an algorithm*?
- (b) Define and explain in plain English the classes
  - (i) [2 marks] *P*,
  - (ii) [2 marks] *NP*,
  - (iii) [2 marks] *NP-Complete*, and
  - (iv) [2 marks] *NP-Hard*.

## Question 7. Various Topics

[20 marks]

- (a) [5 marks] Describe what is *arithmetic coding*.
- (b) [5 marks] Describe what is a *Tutte Polynomial*.
- (c) [5 marks] Give an example of an *approximation* algorithm.
- (d) [5 marks] State at least three classes of *probabilistic algorithms*.

\*\*\*\*\*