IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2017

EEE/EIE PART II: MEng, BEng and ACGI

ALGORITHMS AND COMPLEXITY

Corrected copy

Monday, 19 June 10:00 am

Time allowed: 1:30 hours

There are TWO questions on this paper.

Answer ALL questions. Question One carries 40% of the marks. Question Two carries 60%.

Any special instructions for invigilators and information for candidates are on page 1.

Examiners responsible First Marker(s) : D.F.M. Goodman
Second Marker(s) : C. Thomas

ALGORITHMS AND COMPLEXITY

1. a) For each of the following statements, state whether it is true or false and provide a supporting proof.

   i) $2^n = O(3^n)$ [3]

   ii) $n! = \Omega(2^n)$ [3]

   iii) If $f(n) > 0$, $g(n) > 0$ and $f(n) = O(g(n))$ then $g(n) = \Omega(f(n))$. [4]

   b) Give a tight bound for each of the following recurrence relations, or explain why it's not possible to do so.

   Carefully justify your answers.

   i) $T(n) = 16T(n/4) + O(n)$ [3]

   ii) $T(n) = 9T(n/3) + n(n+1)$ [3]

   iii) $T(n) = 2T(n-1) + 2^{-n}$ [4]

Master Theorem. If $T(n)$ satisfies

$$T(n) = a\,T(\,n/b\,) + O(n^d)$$

for some $a > 0$, $b > 1$ and $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

2. The programming language Earthworm has only one function `split(s, i)` for splitting a string s of length n at position i. It returns two substrings `left` and `right`, so that `left` consists of the first i characters of the string, and `right` consists of the last $n - i$ characters. For example, `split('abcd', 1)` returns `'a', 'bcd'`. The `split` function takes n units of time to run on a string of length n.

You are given the task of using the `split` function to create an efficient implementation of the function `multisplit(s, b)`. This takes a string s of length n and a sorted array of integers b of length k, and returns a sequence of strings corresponding to splitting s at b_1, b_2, etc. For example, `multisplit('abcd', [1, 3])` would return `'a', 'bc', 'd'`.

a) The "right-first" algorithm for implementing `multisplit` is to first split at b_k, then to split the left part at position b_{k-1}, then the left part of that at position b_{k-2}, and so on to the last split at b_1. How many units of time will this algorithm take? Express your answer in terms of n and the values in the array b. [5]

b) In the worst case where the string has to be split into n parts of length 1, i.e. when $b_i = i$ for $i = 1, \ldots, n-1$, how many units of time will the right-first algorithm take? [5]

c) The "binary-split" algorithm repeatedly divides the left and right parts into two equal subparts. So for `'abcd'` it would first split into `'ab'` and `'cd'`, then split `'ab'` into `'a'`, `'b'`, and `'cd'` into `'c'`, `'d'`. How many units of time will this algorithm take for the worst-case considered in part (b) when n is a power of 2? [5]

d) For the general case, use dynamic programming to find how long the optimal sequence of splits would take. Write your answer using pseudocode. You do not need to compute the complexity of the dynamic program.

Hint: you may find it useful to consider the subproblem of finding the optimal number of units of time $C(i : j)$ taken to split the substring of characters in positions i to $j - 1$. [10]

e) For the worst case considered in parts (b) and (c), use the algorithm in (d) to compute the optimal number of units of time to split strings of length 2, 4 and 8. How does the binary-split algorithm compare to these optimal times? [5]