IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2016

EEE/EIE PART II: MEng, BEng and ACGI

ALGORITHMS AND COMPLEXITY

Corrected copy

Monday, 6 June 2:00 pm

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

ii) $T(n) = 9T(n/3) + 3n^2$ [3]

iii) $T(n) = T((n/2)^2)/2 + 1/n$ [4]

b) Give a bound on the complexity of the following operations, and justify your answers in terms of a specific algorithm. *The algorithm code does not need to be given, unless it is needed to support your argument. You do not have to give the best possible algorithm.*

i) Multiplication of two $n \times n$ matrices. [4]

ii) Sorting an array of length n. [6]

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. A univariate polynomial p in variable x consists of a sum of n_p terms:

$$p = \sum_{i=1}^{n_p} c_p[i] x^{d_p[i]}$$

where $c_p[1], \ldots, c_p[n]$ are coefficients and $d_p[1], \ldots, d_p[n]$ are integer degrees with $d_p[i] \geq 0$.

You may assume that any arithmetical operation is $O(1)$ and that appending to a vector is $O(1)$.

a) Multiplication of two polynomials p and q is defined as:

$$p \times q = \sum_{i=1}^{n_p} \sum_{j=1}^{n_q} c_p[i] \cdot c_q[j] \cdot x^{d_p[i]+d_q[i]}$$

i) What is the time complexity of polynomial multiplication based on the formula above? [3]

ii) Using this algorithm, what is the best achievable time complexity for calculating p^k for an arbitrary polynomial p and k a power of 2 ($k = 2, 4, 8, 16, \ldots$)? [7]

b) A non-zero polynomial is *canonical* if $d_p[i] = i - 1$ and $c_p[n_p] \neq 0$. This ensures that all powers of x must be present, they are ordered from smallest to largest power, there are no repeated terms, and the degree of the polynomial is $n_p - 1$.

If a polynomial is not known to be canonical, we call it *irregular*.

i) Give pseudocode for converting an irregular polynomial p to a canonical polynomial, and state the time complexity. You may rely on common library operations on vectors. [6]

ii) If we have two canonical polynomials p and q and want to produce a canonical result $p \times q$, what is the time complexity? You do not need to give pseudo-code, but justify your result. [4]

iii) What is the time complexity of calculating the canonical result p^k where p is canonical and k is a power of 2? [6]

iv) Is it better to calculate p^k using irregular or canonical polynomials? [4]

