EE2-10C

IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2018

EEE/EIE PART II: MEng, BEng and ACGI

ALGORITHMS AND COMPLEXITY

Tuesday, 29 May 10:00 am

Time allowed: 1:30 hours

Corrected copy

(No correction)

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) :	D.B. 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) $2n^2 + 3n^3 + 4n^4 = O(n^4)$. [3]

   ii) $n^2 + 2^{-n} = O(n^2)$. [3]

   b) Describe an algorithm (using pseudocode or a precise description in words) for each of the following tasks, and give a tight upper bound for its running time in O notation. You may assume and state running times for standard algorithms without proving them.

   i) Calculate the mean of an array of n values. [3]

   ii) Count the number of unique items in an array of n values. So, for example, the array $\{1,5,3,1,3,1,3\}$ has $n = 7$ but only 3 unique items. [4]

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

   i) $T(n) = 8T(n/4) + n\sqrt{n}$. [3]

   ii) $T(n) = aT(n/3) + O(n^3)$ assuming $a < 27$. [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. Moria Mining Corporation (MMC) holds the rights to n mine shafts. They estimate that mine shaft i ($0 \leq i < n$) has a total amount of gold g_i. Their competitor, Balrog Incorporated (BI), is going to mount a hostile takeover of these mine shafts in T days. MMC wants to maximise the amount of gold they extract before this happens. Each day, only a single shaft can be mined and the amount of gold they can extract from mine i if they spend t days mining it is

$$m_i(t) = g_i \cdot \left(1 - \frac{1}{t+1}\right).$$

MMC have asked you to devise a dynamic programming algorithm to maximise the total amount of gold they can extract in the time remaining.

We will define $G(n,T)$ as the maximum amount of gold that can be mined from mine shafts 0 to $n-1$ in T days.

a) Explain what each of the following special cases for $G(n,T)$ means in a single non-mathematical sentence, and find the solution in each case:

i) $G(n,0)$. [2]

ii) $G(1,T)$. [2]

iii) $G(n,1)$. [2]

b) Write an equation for $G(n,T)$, by considering the options for the number of days t that MMC should spend mining shaft $n-1$, and expressing the optimal solution in terms of $m_{n-1}(0)$, $m_{n-1}(1)$, $m_{n-1}(2)$, ..., and $G(n-1,T)$, $G(n-1,T-1)$, $G(n-1,T-2)$, ... [6]

c) Write pseudocode for an efficient algorithm to compute $G(n,T)$, the maximum amount of gold they can mine. Your algorithm does not need to return the amount of time they should spend mining each shaft, and your pseudocode should not be substantially longer than 20 lines. [10]

d) Compute the time complexity of your algorithm in terms of n and T.

Note that complexity notation for two variables behaves as you would expect for one variable. In particular $O(f(x)) \times O(g(y)) = O(f(x)g(y))$. [4]

e) Using your answer to parts (a) and (b), find the maximum amount of gold that could be mined for $n = 3$ mines each of which have a total amount of gold $g_i = 600$ in $T = 3$ days. [4]