/CA

EE2-10C

IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2012

EEE/ISE PART II: MEng, BEng and ACGI

Corrected Copy

ALGORITHMS AND COMPLEXITY

Monday, 11 June 4:00 pm

Time allowed: 1:30 hours

There are TWO questions on this paper.

Answer BOTH questions. Question One carries 20 marks. Question Two carries 30 marks.

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

Examiners responsible First Marker(s) : M.M. Draief

Second Marker(s) : D.B. Thomas

EE2-10C

ALGORITHMS AND COMPLEXITY 2012

1. The *Master Theorem* states that:

Let $T(n)$ be the number of operations performed by an algorithm that takes an input of size n. If $T(n)$ satisfies, $T(n) = 0$ for $n = 1$, and for $n \geq 2$

$$T(n) = aT(n/b) + O(n^d)\,, \tag{1.1}$$

where $a > 0, b > 1$ and $d \geq 0$. Then

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b(a) \\ O(n^d \log(n)) & \text{if } d = \log_b(a) \\ O(n^{\log_b(a)}) & \text{if } d < \log_b(a)\,. \end{cases} \tag{1.2}$$

a) Most students responded correctly to this question

a) Explain how equation (1.1) arises in the context of a divide-and-conquer algorithms. [4]

b) Students often provided a completely proof whereas a grpahical answer would have sufficed (as in lecture)

b) Describe the balance between solving subproblems and combining them in each of the cases in equation (1.2). [4]

c) For each of the algorithms below derive the corresponding complexity.

c) Questions i and ii were correctly answered by a majority of students.

i) The problem is solved by dividing the initial problem into two subproblems of half the size, recursively solving each subproblem, and then combining the solutions in cubic time, i.e. $O(n^3)$. [3]

ii) The problem is solved by dividing the initial problem into two subproblems each of quater of the size, recursively solving each subproblem, and then combining the solutions in time $O(\sqrt{n})$. [3]

iii) The problem is solved by recursively solving one subproblem of size $n-1$, where n is the size of the initial problem, and then performing additional operations requiring linear time, i.e. $O(n)$. [3]

c) Question iii, many students try to apply 1.2 which does not apply here.

d) Given n sets $S_1, \ldots, S_n$, where S_i is a subset of $\{1, \ldots, m\}$, describe what the following pseudocode does and derive its complexity in terms of the parameters m and n.

d) A majority of students did this reasonably.

```
for i = 1 to n {
  for j = 1 to n {
    Fail=true
    for each element x ∈ S_i {
      if x ∈ S_j then Fail=false}
    if Fail then
    report that S_i and S_j are disjoint
  }
}
```

[3]

a) Many students did not provide all the solutions or missed some as they did not follow a logic way of listing them.

a)iii) Many students tried to prove optimality in general while the question was to do it for this special case using i

a) in ii, many students did not order the coins and so gave a bogus algorithm and failed to provide the correct complexity.

2. We need to give change of some amount of money using the fewest number of coins. Suppose that the currency of interest has coins with n distinct values in $\{c_1, c_2, \ldots, c_n\}$.

a) i) Provide all possible solutions for giving change of 34 when the coins are in $\{1, 5, 10, 25, 100\}$. [2]

ii) A clever cashier decides to use the following algorithm:

At each iteration, add a coin of the largest value that does not take us past the amount to be paid.

Write an efficient pseudocode for this algorithm when the coins are in $\{c_1, c_2, \ldots, c_n\}$ and analyse its running time. [5]

iii) For the special case of giving change of 34 using coins in $\{1, 5, 10, 25, 100\}$, show that the solution found by the above greedy procedure is optimal. [5]

b) Suppose that we now have coins in $\{1, 10, 25\}$.

i) Apply the greedy algorithm to give change of 30. [2]

ii) Show that there is a better solution than the one provided by the greedy algorithm. [2]

b) Most students did this question convieniently

c) As illustrated above the greedy algorithm is not always optimal. We now describe a dynamic programming approach to the general problem.

Let $C[m]$ to be the minimum number of coins we need to give change of an amount m using coins from $\{c_1, c_2, \ldots, c_n\}$. We recursively define the value of an optimal solution as follows

$$C[m] = \begin{cases} \infty & \text{if } m < 0 \\ 0 & \text{if } m = 0 \\ 1 + \min_{1 \le i \le n} (C[m - c_i]) & \text{if } m \ge 1\,. \end{cases} \tag{2.1}$$

c) A number of students struggled with the question as it seems that many did not get the point of dynamic programming and only repeated material from the course that was not necessarily relevant.

i) Apply the above dynamic programming with coins from $\{1, 10, 25, 50\}$ to give change of 29. [4]

ii) Derive the running time of the algorithm when we use n coins $\{c_1, c_2, \ldots, c_n\}$ to give change of an amount m. [5]

d) In this final question we describe another dynamic programming solution.

Let $C[m, n]$ be the smallest number of coins used to give change of an amount m, using only coins in $\{c_1, c_2, \ldots, c_n\}$. We will assume that $c_1 = 1$.

$C[m, n]$ can be defined using two cases (1) we do not use coin c_n or (2) we use coin c_n and reduce the amount.

Combine the two cases to derive a new dynamic program.

Hint: Note that since $c_1 = 1$, we have $C[m, 1] = m$. [5]

d) This is a hard question meant to giva an edge to tht students who properly explored dynamic programming to do well. Very few students attempted this question.

Algorithms & Complexity EE2-10C

Solutions - 2012 1/4

1/

a/ Initial pb of size n subdivided into $\underline{a}$ smaller subproblems of size n/b each.

Once each of these pbs is solved, it takes $O(n^d)$ operations to combine their solutions in order to obtain a solution to the initial pb.

b) + $d > \log_b a$: Here combining subproblems is the most expensive part of algo

+ $d < \log_b a$: It is cheaper to combine the problems than it is to solve all the subproblems at hand.

+ $d = \log_b a$: combining & solving are as expensive to perform.

c)

i) $T(n) = 2T(n/2) + O(n^3) \Rightarrow T(n) = O(n^3)$

$a = b = 2$; $d = 3$ $\log_a b = 1 < 3 = d$.

ii) $T(n) = 2T(n/4) + O(\sqrt{n}) \Rightarrow T(n) = O(n$

$a = 2, b = 4$ $d = 1/2$ $\log_a b = 2 > 1/2 = d$.

iii) $T(n) = T(n-1) + n = T(n-2) + n-1 + n-2$

By induction; $T(n) = \frac{n(n-1)}{2} = O(n^2)$

2/4

d) Code checks whether $S_1 \dots S_n \subset \{1, \dots, m\}$ are disjoint or not. It takes $O(n^2 m)$ operations

~~35~~

2/ ~~$34 = 25 + 5 + 4 \times 1$~~
~~$34 = 25 + 5 + 4$~~

a/

i/ $\boxed{34 = 1 \times 25 + 1 \times 5 + 4 \times 1 \qquad 6 \text{ coins}}$

$34 = 1 \times 25 + 9 \times 1$	10 coins
$34 = 3 \times 10 + 4 \times 1$	7 "
$34 = 2 \times 10 + 2 \times 5 + 4 \times 1$.	8
$34 = 2 \times 10 + 1 \times 5 + 9 \times 1$	12
$34 = 2 \times 10 + 14 \times 1$	16
$34 = 1 \times 10 + 4 \times 5 + 4 \times 1$	9
$34 = 1 \times 10 + 3 \times 5 + 9 \times 1$	13
$34 = 1 \times 10 + 2 \times 5 + 14 \times 1$	17
$34 = 1 \times 10 + 1 \times 5 + 19 \times 1$	21
$34 = 6 \times 5 + 4 \times 1$	10.
$34 = 5 \times 5 + 9 \times 1$	14
$34 = 4 \times 5 + 14 \times 1$	18
$34 = 3 \times 5 + 19 \times 1$	22
$34 = 2 \times 5 + 24 \times 1$	26
$34 = 1 \times 5 + 29 \times 1$	30
$34 = 34 \times 1$	34

3/4

ii) Sort coins by value $c_1 < \dots < c_n$

$S \leftarrow \emptyset$

While $(x \neq 0)$

Let k largest integer such that $c_k \leq x$.

if $k = 0$ return no solution found.

else $x \leftarrow x - c_k$

$S \leftarrow S \cup \{k\}$

return S.

$O(n \log n)$ for sorting + cost going through the sequence $c_1 \dots c_n$.

iii) By i) $34 = 1 \times 25 + 5 \times 1 + 4 \times 1$: 6 coins is optimal & it is the solution provided by the greedy algorithm.

b) i) $30 = 25 + 5 \times 1$ ~~which~~ is greedy solution.

~~30 25~~

ii) Optimal $30 = 3 \times 10$ 3 coins instead of 6 coins with greedy

c) $c[29] = 1 + \min \begin{cases} c[29-50] = \infty \\ c[29-25] = c[4] = 4 \\ c[29-10] = c[19] \\ c[29-1] = c[28] \end{cases}$

$c[19] = 1 + \min \begin{cases} c[19-50] = \infty \\ c[19-25] = \infty \\ c[19-10] = c[9] = 9 \\ c[19-1] = c[18] \end{cases}$

$c[18] = 1 + \min \begin{cases} c[18-10] = c[8] = 8 \\ c[18-1] = 17 = 6 \end{cases}$

$c[29] = c[4] + 1 = 5$

Solution $\boxed{29 = 25 + 4 \times 1}$

ii) Equation (2.1) takes at most n operations to go through all the coins. & this has to be done at most m times.

$O(mn)$ complexity

d) $c[m, n] = \begin{cases} \infty & m < 0 \\ 0 & m = 0 \\ m & n = 1 \\ \min\{c[m, n-1], 1 + c[m - c_i, n]\} & m \geq 1 \\ & n \geq 2 \end{cases}$