ALGORITHMS AND COMPLEXITY

Students did well here as this problem is very similar to a problem we looked at in class.

1. In this exercise, we analyse a number of ways of multiplying two n-digit binary numbers $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$ where the a_is and b_is are in $\{0, 1\}$.

a) If one uses the standard multiplication procedure, how many multiplications of one-digit numbers are required ? [2]

b) Suppose that n is a power of 2. Let $a' = (a_1, \ldots, a_{n/2})$, $a'' = (a_{(n/2)}, \ldots, a_n)$, $b' = (b_1, \ldots, b_{n/2})$, $b'' = (b_{(n/2)}, \ldots, b_n)$, so that

Some students had issues with part c though.

$$a = a'2^{n/2} + a'', \quad b = b'2^{n/2} + b''.$$

i) Express $a \times b$ in terms of a', a'', b', b''. [2]

ii) Write a recursive procedure for performing the multiplication $a \times b$, using one-digit multiplications, one-digit additions and shifts. [4]

iii) Derive its complexity, in terms of the number of one-digit multiplications, one-digit additions and shifts involved, in O notation. Is it quicker than the standard multiplication procedure? [4]

c) We now describe an alternative algorithm for performing multiplications.

i) Using the identity

$$(a' + a'')(b' + b'') - a'b' - a''b'' = a'b'' + a''b',$$

write another recursive procedure for performing the multiplication $a \times b$. [4]

ii) Derive its complexity in O notation. How does it compare to the other two multiplication procedures? [4]

Master Theorem

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 $n \geq 2$

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

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

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

2. Suppose you are given a list of n numbers $a_1, a_2, \ldots, a_n$. The goal is to output another list of numbers b_{ij} for $i = 1, \ldots, n$ and $j = i+1, i+2, \ldots n$ such that

$$b_{ij} = a_i + a_{i+2} + \cdots + a_{j-1} + a_j \,.$$

[3]

a) We start with a simple example. Compute the list of b_{ij}'s for $a_1 = 1, a_2 = 2, \ldots, a_n = n$.

b) To solve the problem in general, we propose the following simple algorithm.

for $i = 1, \ldots, n-1$ **do**
  for $j = i+1, i+2, \ldots, n$ **do**
    Add up elements a_i through a_j
    Store the results in b_{ij}
  end for
end for

i) Show that the number of operations performed by the above algorithm is $O(n^3)$. [6]

Hint: Use the fact that $\sum_{i=1}^{n} i^2 = \frac{(2n-1)n(n+1)}{6} = O(n^3)$, *for n large.*

ii) Describe an alternative more efficient procedure for computing the list of b_{ij} and derive its complexity.

[6]

Many students had trouble understanding the notation bij corresponding to the table of the sums of the number ai.

3. Suppose that we want to multiply four matrices A, B, C, D. This involves multiplying two matrices at a time. Matrix multiplication is not commutative in general, $A \times B$ is not necessarily equal to $B \times A$ but it is associative, i.e. $A \times (B \times C) = (A \times B) \times C$.

We will refer to the choice of the positions of the brackets, i.e. the order in which the multiplication is performed, as *parenthisation*. In particular, to compute $A \times B \times C$, there are two possible parenthisations $A \times (B \times C)$ and $(A \times B) \times C$.

Most students answered question a correctly

a) Multiplying an $m \times n$ matrix by $n \times p$ matrix takes mnp multiplications of the entries of the two matrices.

Consider the matrices A_1, A_2, A_3 and A_4 that are 5×4, 4×6, 6×2 and 2×7 respectively. Consider all possible ways of performing the multiplication $A_1 \times A_2 \times A_3 \times A_4$ and compute their costs in terms of the number multiplications of the entries of the matrices considered. [4]

b) Assume that we want to multiply n matrices $A_1, \ldots, A_n$ where the matrix A_i has dimension $p_{i-1} \times p_i$, for $i = 1, \ldots, n$. Our goal is to perform the fewest number of multiplications of the entries of the matrices by carefully choosing the order in which the multiplications will be performed.

Many students found this question challenging and had problems justifying b) i).

i) Let $P(n)$ be the number of alternative parenthisations of the product $A_1 \times \cdots \times A_n$. Show that

$$P(n) = \begin{cases} 1 & \text{if } n = 1, 2 \\ \sum_{k=1}^{n-1} P(k)P(n-k) & \text{if } n \geq 3\,. \end{cases}$$

[4]

ii) A first solution would be to use exhaustive search/brute force over the $P(n)$ parenthisations. Is this approach efficient? [4]

Hint: Show that $P(n+1) \geq P(n) + P(n-1)$ and use the fact that the Fibonnacci numbers gow exponentially in n.

c) We are now going to describe a different approach based on Dynamic Programming. For $1 \leq i \leq j \leq n$, let $A_{i,\ldots,j} = A_i \times A_{i+1} \times \cdots \times A_j$ which has dimension $p_{i-1} \times p_j$. Let $m[i, j]$ be the minimum (optimal) number of multiplications needed to compute $A_{i,\ldots,j}$.

The students who answered b were able to solve c.

i) Assume that optimal solution of $A_{i,\ldots,j}$ involves splitting into $A_{i,\ldots,k}$ and $A_{k+1,\ldots,j}$, $i \leq k < j$. Derive a relationship between $m[i, j]$, $m[i, k]$, $m[k+1, j]$, p_{i-1}, p_k and p_j. [6]

ii) Apply the above idea to the following example A_1, A_2, A_3 and A_4 with $p_0 = 5, p_1 = 4, p_2 = 6, p_3 = 2$, and p_4=7. Find $m[1, 4]$ and the solution to this instance of the parenthisation problem.

Using the analysis of 3.a), describe how the above dynamic programme compares to the outcome of an exhaustive search? [7]