CSE 450 Assignment 3

14th October, 2022

Submission Instructions: Deadline is **11:59pm on 10/21/2022**. Late submissions will be penalized, therefore please ensure that you submit (file upload is completed) before the deadline. Additionally, you can download the submitted file to verify if the file was uploaded correctly. **Please TYPE UP YOUR SOLUTIONS and submit a PDF** electronically, via *Canvas*. Furthermore, please note that the graders will grade 2 out of the 4 questions randomly. Therefore, if the grader decides to check questions 1 and 4,and you haven't answered question 4, you'll lose points for question 4. Hence, please answer all the questions.

1. Let u and v be two n bit numbers where for simplicity n is a power of 2. The traditional multiplication algorithm requires $O(n^2)$ operations. A divide and conquer based algorithm splits the numbers into two equal parts, computing the product as

$$\begin{aligned} uv &= (a2^{\frac{n}{2}} + b)(c2^{\frac{n}{2}} + d) \\ &= ac2^n + (ad + bc)2^{\frac{n}{2}} + bd \end{aligned}$$

The multiplications ac, ad, bc, and bd are done using this algorihtm recuresiverly. **[25 points]**

(a) Determine the computing time of the above algorithm

Solution: $T(n) = 4T(n/2) + \theta(n) = \theta(n^2)$. We divide the problem into 4 subproblems (ac, ad, bc, bd) of size n/2 each. The dividing step takes constant time, and the recombining step involves adding and shifting n-bit numbers and therefore takes time $\theta(n)$. This gives the recurrence relation $T(n) = 4T(n/2) + \theta(n)$. For this case, we have $a = 4, b = 2, f(n) = n$, so $n^{log_b a} = n^2$. $f(n) = O(n^{log_b a - \epsilon})$. Master Theorem applies. Therefore $T(n) = \theta(n^2)$

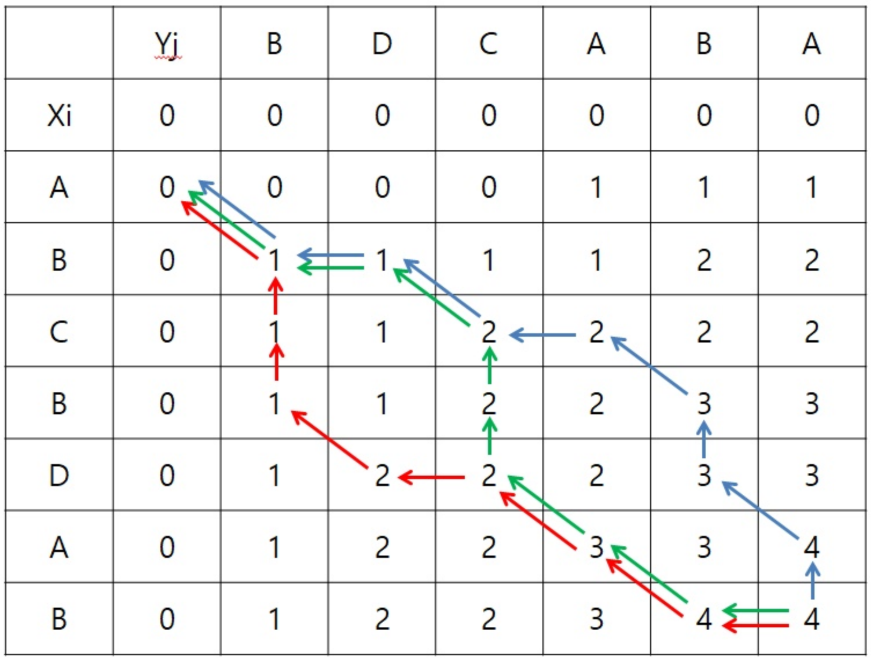
(b) What is the computing time if $ad + bc$ is computed as $(a + b)(c + d) - ac - bd$?

Solution: $T(n) = 3T(n/2) + \theta(n) = \theta(n^{log_2 3})$. If we write ad + bc as (a + b)(c + d) - ac - bd, then we only need to compute three integer multiplications of size n/2, namely ac, bd, and (a + c)(c + d). This is advantageous since we replace one multiplication with additions and subtractions, which are less expensive operations. The divide step will now take time $\theta(n)$, since we need to calculate a + c and c + d, and the recombining step will still take $\theta(n)$. This leads to the recurrence relation $T(n) = 3T(n/2) + \theta(n) = \theta(n^2)$. We have a = 3, b = 2, and $f(n) = n$, so $n^{log_b a} \approx n^{1.585}$. $f(n) = O(n^{1.585-\epsilon})$. Master Theorem applies. Therefore, $T(n) = \theta(n^{log_2 3})$.

2. Find the longest common subsequence (as described in class and the slides) of the following two sequences by Dynamic Programming Algorithm. Show all your work. [**25 points**]

   X = ABCBDAB,   Y=BDCABA

   Solution: Refer to the lecture slide



	Yj	B	D	C	A	B	A
Xi	0	0	0	0	0	0	0
A	0	0	0	0	1	1	1
B	0	1	1	1	1	2	2
C	0	1	1	2	2	2	2
B	0	1	1	2	2	3	3
D	0	1	2	2	2	3	3
A	0	1	2	2	3	3	4
B	0	1	2	2	3	4	4

Longest Common Subsequence : BDAB, BCAB, BCBA

3. Find the optimal order of multiplying four matrices whose sequence of dimensions is (13, 5, 89, 3, 34) (i.e. the first matrix is 13 x 5, the second matrix is 5 x 89, and so on) and also the fewest number of multiplications needed to find the resulting matrix. Show all the steps of computation. [**25 points**]

Solution: The optimal order of multiplying $A_1 \times A_2 \times A_3 \times A_4$ is $((A_1 \times (A_2 \times A_3)) \times A_4)$. The fewest number of multiplications needed to find the resulting matrix is 2856.
The steps of computations are :
m[1,2] = 13 x 5 x 89 = 5785
m[2,3] = 5 x 89 x 3 = 1335
m[3,4] = 89 x 3 x 34 = 9078
m[1,3] = min{m[2,3] + 13 x 5 x 3, m[1,2] + 13 x 89 x 3} = m[2,3] + 13 x 5 x 3 = 1530
m[2,4] = min{m[3,4] + 5 x 89 x 34, m[2,3] + 5 x 3 x 34} = m[2,3] + 5 x 3 x 34 = 1845
m[1,4] = min{m[2,4] + 13 x 5 x 34, m[1,2] + m[3,4] + 13 x 89 x 34, m[1,3] + 13 x 3 x 34} = m[1,3] + 13 x 3 x 34 = 2856

4. Let A_1, ..., A_n be the matrices where the dimensions of A_i are d_{i-1} x d_i, for i = 1, ..., n. Here is a strategy for determining the best order in which to perform the matrix multiplications to compute A_1 x A_2 x .... x A_n : At each step, choose the largest remaining dimension (from among d_1, ... d_{n-1}), and multiply two adjacent matrices which share that dimension.. **[25 points]**

   (a) What is the order of the running time of this algorithm (only to determine the order in which to multiply the matrices, not including the actual multiplications)?

   Solution: To determine the order in which to multiply the matrices in equivalent to sort the sequence of given dimensions $d_1, ..., d_{n-1}$ (expect the first dimension d_0 and the last dimension d_n). The time complexity will be $O(nlogn)$

   (b) Either give a convincing argument that this strategy will always minimize the number of multiplications, or give an example where it fails to do so.

   Solution: The strategy does not work for all the cases. Following is a counterexample. $A_1 \times A_2 \times A_3$ where the dimensions of A_1, A_2 and A_3 are $1 \times 2, 2 \times 3$ and 3×4.
   The strategy will give the order of $A_1 \times (A_2 \times A_3)$, which requires 32 multiplications. $2 \times 3 \times 4 + 1 \times 2 \times 4 = 32$
   The optimal order should be $(A_1 \times A_2) \times A_3$, which only requires 18 multiplication. $1 \times 2 \times 3 + 1 \times 3 \times 4 = 18$