CS 101 Fall 2021 - Quiz 4-A

10/25/2021 - 30 Minutes **Name:** **ID number:**

The Master Theorem for $T(n) = aT(\frac{n}{b}) + \Theta(n^d)$: If $\log_b a = d$ then $T(n) = O(n^d \log n)$ else $T(n) = O(n^{max(\log_b a, d)})$.

Problem 1 Notes of Discussion (5 pts)

I promise that I will complete this QUIZ independently, and will not use any electronic products or paper-based materials during the QUIZ, nor will I communicate with other students during this QUIZ.

True or False: I have read the notes and understood them.

Problem1
T

Problem 2 True or False (3×2 pts)

The following questions are True or False questions, you should judge whether each statement is true or false.

Note: You should write down your answers in the box below.

Problem 2.1	Problem 2.2	Problem 2.3
T	F	T

(1) Queue is the common data structure for implementation of Breadth First Traversal.

(2) The degree and the depth of the root node are both zero in all tress.

(3) If a is an ancestor of b, then there is exactly one unique path from a to b in the tree.

Problem 3 Recurrence and the Master Theorem (8pts)

Given the recurrence $T(n) = aT(n/b) + f(n)$ with $T(1) = 1$.

(1) If the recurrence indicates a divide and conquer algorithm,

a. the original problem of size n is divided into __A__ subproblems and each subproblem has size __D__(2pts);

(A) a (B) b (C) n/a (D) n/b (E) $f(n)$

b. $f(n)$ is the time complexity of __B__. (2pts)

(A) Divide and Conquer (B) Divide and Combine (C) Conquer and Combine

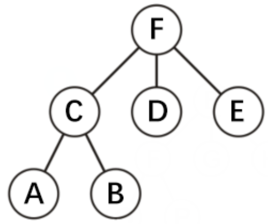
(2) a. If $(a, b, f(n)) = (2, 3, 3\sqrt{n})$, then the solution to this recurrence is $T(n) =$ __$O(n^{\log_3 2})$__. (2pts)

b. If $(a, b, f(n)) =$ __(2, 2, n)__, then the recurrence indicates the **Merge Sort** algorithm covered in our lecture. The solution to this recurrence is $T(n) =$ __$O(n \log n)$__. (2pts)

Note: Write your answer for time complexity in asymptotic order form i.e. $T(n) = O(g(n))$.

Problem 4 Tree Traversal (6pts)

Run **Depth First Traversal** on the tree shown below.



Note:

1. Decide on an appropriate data structure to implement the traversal.

2. When you are pushing the children of a node into your data structure, please push them **in a reverse order** i.e. from right to left.

3. **Show all current elements in your data structure at each step** clearly . **Popping a node** or **pushing a sequence of children** can be considered as one single step.

4. **Write down your traversal sequence** i.e. the order that you pop elements out of the data structure. *Don't worry if you can't write the right answer at one chance. You can scratch in this paper but please* ***mark your final answer****.*

Stack:

F	□
C	
D	D
E	E
A	
B	B
D	D
E	E
D	
E	E

Sequence:

F C A B D E

Problem 5 Matrix Multiplication(10pts)

Recall that Strassen found an more efficient approach to calculate matrix multiplication $\mathbf{A} \times \mathbf{B}$, where $\mathbf{A}$ and $\mathbf{B}$ are both square matrices of size $n \times n$ ($n = 2^k$).

(1) Let $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix}$, calculate $\mathbf{A} \times \mathbf{B}$. How many scalar multiplications do you perform?(2pts)

$\mathbf{A} \times \mathbf{B} = \begin{bmatrix} 1 \times 4 + 2 \times 2 & 1 \times 3 + 2 \times 1 \\ 3 \times 4 + 4 \times 2 & 3 \times 3 + 4 \times 1 \end{bmatrix} = \begin{bmatrix} 8 & 5 \\ 20 & 13 \end{bmatrix}$. 8 scalar multiplications.

(2) Recall that Strassen's algorithm partitions matrices and computes $\mathbf{A} \times \mathbf{B} = \left[\begin{array}{c|c} \mathbf{A_{11}} & \mathbf{A_{12}} \\ \hline \mathbf{A_{21}} & \mathbf{A_{22}} \end{array}\right] \times \left[\begin{array}{c|c} \mathbf{B_{11}} & \mathbf{B_{12}} \\ \hline \mathbf{B_{21}} & \mathbf{B_{22}} \end{array}\right]$ in the following way($\mathbf{A_{ij}}$ and $\mathbf{B_{ij}}$ are $2^{k-1} \times 2^{k-1}$ submatrices of $\mathbf{A}$ and $\mathbf{B}$). Let

$$\mathbf{P_1} = \mathbf{A_{11}} \times (\mathbf{B_{12}} - \mathbf{B_{22}}) \qquad \mathbf{P_2} = (\mathbf{A_{11}} + \mathbf{A_{12}}) \times \mathbf{B_{22}} \qquad \mathbf{P_3} = (\mathbf{A_{21}} + \mathbf{A_{22}}) \times \mathbf{B_{11}}$$

$$\mathbf{P_4} = \mathbf{A_{22}} \times (\mathbf{B_{21}} - \mathbf{B_{11}}) \qquad \mathbf{P_5} = (\mathbf{A_{11}} + \mathbf{A_{22}}) \times (\mathbf{B_{11}} + \mathbf{B_{22}}) \qquad \mathbf{P_6} = (\mathbf{A_{12}} - \mathbf{A_{22}}) \times (\mathbf{B_{21}} + \mathbf{B_{22}})$$

$$\mathbf{P_7} = (\mathbf{A_{11}} - \mathbf{A_{21}}) \times (\mathbf{B_{11}} + \mathbf{B_{12}})$$

then we can obtain:

$$(\mathbf{A} \times \mathbf{B})_{\mathbf{11}} = \mathbf{P_5} + \mathbf{P_4} - \mathbf{P_2} + \mathbf{P_6} \qquad (\mathbf{A} \times \mathbf{B})_{\mathbf{12}} = \mathbf{P_x} + \mathbf{P_2}$$

$$(\mathbf{A} \times \mathbf{B})_{\mathbf{21}} = \mathbf{P_3} + \mathbf{P_y} \qquad (\mathbf{A} \times \mathbf{B})_{\mathbf{22}} = \mathbf{P_1} + \mathbf{P_5} - \mathbf{P_3} - \mathbf{P_7}$$

What value should x and y take? How many scalar multiplications do you perform if you apply this to (1)? (2pts)
Hint: Matrix multiplication still applies to partitioned matrices.

$(x, y) = (1, 4)$. 7 scalar multiplications.

(3) Use Strassen's algorithm from (2) to come up with a divide-and-conquer algorithm to calculate the matrix multiplication $\mathbf{A} \times \mathbf{B}$ in more efficient than $\Theta(n^3)$ time. Write down your main idea briefly. (4pts)

1. If the problem is reduced into $n = 1$ i.e. $k = 0$, return the scalar product ab.
2. Else we partition $\mathbf{A}$ and $\mathbf{B}$, and calculate all multipliers and multiplicands in each $\mathbf{P_i}$. **(Divide)**
3. Recur for each $\mathbf{P_i}$, all seven of which are subproblems of size $n/2$. **(Conquer)**
4. Compute $(\mathbf{A} \times \mathbf{B})_{\mathbf{ij}}$ accordingly using linear combinations of $\mathbf{P_i}$, put them together to form the solution. **(Merge)**

(4) What is the time complexity of your algorithm? Write down the corresponding recurrence and solve it. You **are not required** to show your analysis and calculation. (2pts)
Note: You can assume that all the numbers involved are small enough so that basic arithmetic operations like scalar addition and scalar multiplication take $O(1)$ time.

Time complexity for dividing and merging subproblems is $\Theta(n^2)$ and the original problem is divided into 7 subproblems of half size, hence $T(n) = 7T(n/2) + \Theta(n^2)$. Then by the Master Theorem $T(n) = O(n^{\log_b a}) = O(n^{\log_2 7})$.