CS 101 Fall 2021 - Quiz 4-B

10/26/2021 - 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
F	F	T

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

(2) There exists at least one non-leaf node in a tree whose depth is the height of the tree.

(3) If b is a descendant of a, 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 $T(n) = aT(n/b) + cn^d$ 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 __E__(2pts);

(A) a (B) b (C) c (D) n/a (E) n/b (F) n^d

b. cn^d is the time complexity of __AC__. *Note: This question has one or more correct answer(s).* (2pts)

(A) Dividing the original problem into several subproblems

(B) Recurring for all subproblems

(C) Merging solutions to subproblems into the overall one

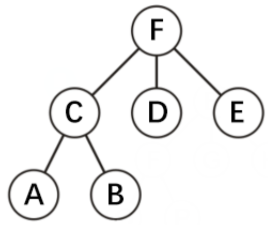
(2) a. If $(a, b, c, d) = (2, 3, \frac{1}{2}, \frac{3}{2})$, then the solution to this recurrence is $T(n) =$ __$O(n^{\frac{3}{2}})$__. (2pts)

b. If the recurrence indicates the **Strassen's algorithm** covered in our lecture which multiplies two 2-by-2 partitioned matrices via only 7 matrix multiplications, then $(a, b, d) =$ __(7,2,2)__ and the solution to this recurrence is $T(n) =$ __$O(n^{\log_2 7})$__. (2pts)

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

Problem 4 Tree Traversal (6pts)

Run **Breadth 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 **alphabetically** i.e. from left to right.
3. **Show every current element 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***.

Queue:

F

□

C D E

D E

D E A B

E A B

A B

B

Sequence:

F C D E A B

CS 101 Fall 2021 - Quiz 4-B

10/26/2021 - 30 Minutes **Name:** **ID number:**

Problem 5 Magical Matrix (10pts)

Let's consider such a special square matrix of size $n \times n$ ($n = 2^k$) named **Magical Matrix $\mathbf{H_k}$**, which satisfies the following properties:

(a) $\mathbf{H_0} = \begin{bmatrix} c \end{bmatrix}$, where c is a 1×1 constant.

(b) For $k \geq 1$, define $\mathbf{H_k} = \left[\begin{array}{c|c} \mathbf{H_{k-1}} & \mathbf{H_{k-1}} \\ \hline \mathbf{H_{k-1}} & -\mathbf{H_{k-1}} \end{array}\right]$, where $\mathbf{H_k}$ is a $2^k \times 2^k$ matrix and $\mathbf{H_{k-1}}$ is a $2^{k-1} \times 2^{k-1}$ matrix.

Let $\mathbf{v} = \begin{bmatrix} \mathbf{v_1} \\ \mathbf{v_2} \end{bmatrix}$ be a column vector of length $n = 2^k$, where $\mathbf{v_1}$ is the upper half of $\mathbf{v}$ of length $\frac{n}{2} = 2^{k-1}$ and $\mathbf{v_2}$ is the bottom half of $\mathbf{v}$ also of length $\frac{n}{2} = 2^{k-1}$. Now, we are going to develop a faster approach to calculate the matrix-vector product $\mathbf{H_k v}$.

(1) When $c = 1$ and $\mathbf{v_1} = \mathbf{v_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, write down $\mathbf{H_2}$ and calculate $\mathbf{H_2 v}$ according to the definition above. (2pts)

$$\mathbf{H_2} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}, \quad \mathbf{H_2 v} = \begin{bmatrix} 2 \\ -2 \\ 0 \\ 0 \end{bmatrix}$$

(2) Write the matrix-vector product $\mathbf{H_k v}$ in terms of $\mathbf{H_{k-1}}$, $\mathbf{v_1}$ and $\mathbf{v_2}$. (2pts)

Hint: Matrix multiplication still applies to partitioned matrices.

$$\mathbf{H_k v} = \begin{bmatrix} \mathbf{H_{k-1}}(\mathbf{v_1} + \mathbf{v_2}) \\ \mathbf{H_{k-1}}(\mathbf{v_1} - \mathbf{v_2}) \end{bmatrix}$$

(3) Use your result from (2) to come up with a divide-and-conquer algorithm to calculate the matrix-vector product $\mathbf{H_k v}$ in more efficient than $\Theta(n^2)$ time. Write your main idea briefly. (3pts)

1. If the problem is reduced into $n = 1$ i.e. $k = 0$, return the scalar product $c\mathbf{v}$.
2. Else we divide $\mathbf{v}$ into upper half $\mathbf{v_1}$ and bottom half $\mathbf{v_2}$, and extract $\mathbf{H_{k-1}}$ from $\mathbf{H_k}$. **(Divide)**
3. Recur for $\mathbf{H_{k-1} v_1}$ and $\mathbf{H_{k-1} v_2}$, both of which are subproblems of size $n/2$. **(Conquer)**
4. Compute $\mathbf{H_{k-1} v_1} + \mathbf{H_{k-1} v_2}$ and $\mathbf{H_{k-1} v_1} - \mathbf{H_{k-1} v_2}$, put them together as upper and bottom half of the result respectively 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 or 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 $\Theta(n)$ and the original problem is divided into 2 subproblems of half size, hence $T(n) = 2T(n/2) + n$. Then by the Master Theorem $log_b a = d = 1$, so $T(n) = O(n \log n)$.