Q1-Q4 Solutions for HW6-CS 6033 Fall 2024

Q1 → Master Theorem - Recurrence Formulas

1. Solve these recurrence formulas using Θ notation:
   - $T(n) = 2T(n/3) + 1$
   - $T(n) = 5T(n/4) + n$
   - $T(n) = 7T(n/7) + n$
   - $T(n) = 9T(n/3) + n^2$
   - $T(n) = 8T(n/2) + n^3$
   - $T(n) = 7T(n/2) + \Theta(n^2)$
   - $T(n) = T(n/2) + \Theta(1)$
   - $T(n) = 5T(n/4) + \Theta(n^2)$

Recurrence formula	a	b	$n^{\log_b a}$	f(n)	Case	T(n)
$2T(n/3) + 1$	2	3	$n^{\log_3 2}$	1	1 → cost dominated by leaves	$\Theta(n^{\log_3 2})$
$5T(n/4) + n$	5	4	$n^{\log_4 5}$	n	1 → cost dominated by leaves	$\Theta(n^{\log_4 5})$
$7T(n/7) + n$	7	7	$n^{\log_7 7}$	n	2 → cost same at every level	$\Theta(nlogn)$
$9T(n/3) + n^2$	9	3	$n^{\log_3 9}$	n^2	2 → cost same at every level	$\Theta(n^2 logn)$
$8T(n/2) + n^3$	8	2	$n^{\log_2 8}$	n^3	2 → cost same at every level	$\Theta(n^3 logn)$
$7T(n/2) + \Theta(n^2)$	7	2	$n^{\log_2 7}$	n^2	1 → cost dominated by leaves	$\Theta(n^{\log_2 7})$
$T(n/2) + \Theta(1)$	1	2	$n^{\log_2 1}$	1	2 → cost same at every level	$\Theta(logn)$
$5T(n/4) + \Theta(n^2)$	5	4	$n^{\log_4 5}$	n^2	3 → cost dominated by root	$\Theta(n^2)$

Q2 → Master Theorem - Divide and Conquer problems

2. Suppose you came up with three solutions to a homework problem:

   - The first solution, Algorithm A, divides the original problem into 5 subproblem of size $n/2$, recursively solves the subproblems, and then solves the original problem by combining the subproblems in linear time.
   - The second solution, Algorithm B, divides the original problem into two subproblems of size $\frac{9}{10}n$, recursively solves the subproblems, and then solves the original problem by combining the subproblems in linear time.
   - The third solution, Algorithm C, divides the original problem into problems of size $n/3$, recursively solves the subproblems and then solves the original problem by combining the subproblems in $\Theta(n^2)$ time

   Provide the recurrence formula for each of the algorithms. What are the running times of each of these algorithms (in Θ notation), and which of your algorithms is fastest?

Recurrence formula	**a**	**b**	$n^{log_b a}$	**f(n)**	**Case**	**T(n)**
$5T(n/2) + \Theta(n)$	5	2	$n^{log_2 5}$	n	1 → cost dominated by leaves	$\Theta(n^{log_2 5})$
$2T(9n/10) + \Theta(n)$	2	10/9	$n^{log_{10/9} 2}$	n	1 → cost dominated by leaves	$\Theta(n^{log_{10/9} 2})$
$3T(n/3) + \Theta(n^2)$	3	3	$n^{log_3 3}$	n^2	3 → cost dominated by root	$\Theta(n^2)$

Comparing the powers of n for the running times, we find that $2 < \log_2 5 < \log_{10/9} 2$. . This shows us that Algorithm C is the fastest.
(P.S: Since the value of a wasn't clarified for Algorithm C initially, answers with different values of a will be considered).

Q3 → Matrix Multiplication

3. Matrix multiplication:

- Divide the 4×4 matrix A matrix into 4 smaller matrices of size 2×2:

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \text{ to create: } \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

Show the 2×2 matrices A_{11}, A_{12}, A_{21}, and A_{22}.

- Perform some of the calculations needed to compute $A \times B$ using Strassen's algorithm:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \cdot \begin{bmatrix} 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 \\ 25 & 26 & 27 & 28 \\ 29 & 30 & 31 & 32 \end{bmatrix} = C$$

For the matrices given above:

  - Compute P_1, P_2, and C_{12}
  - Compute $A_{11} \cdot B_{12}$ and $A_{12} \cdot B_{22}$
  - Check to see that $C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$.

- Verify that $C_{22} = P_5 + P_1 - P_3 - P_7$ by replacing each P_i with its value and reducing the expression.

$$A_{11} = \begin{bmatrix} 1 & 2 \\ 5 & 6 \end{bmatrix} A_{12} = \begin{bmatrix} 3 & 4 \\ 7 & 8 \end{bmatrix} A_{21} = \begin{bmatrix} 9 & 10 \\ 13 & 14 \end{bmatrix} A_{22} = \begin{bmatrix} 11 & 12 \\ 15 & 16 \end{bmatrix}$$

$$B_{11} = \begin{bmatrix} 17 & 18 \\ 21 & 22 \end{bmatrix} B_{12} = \begin{bmatrix} 19 & 20 \\ 23 & 24 \end{bmatrix} B_{21} = \begin{bmatrix} 25 & 26 \\ 29 & 30 \end{bmatrix} B_{22} = \begin{bmatrix} 27 & 28 \\ 31 & 32 \end{bmatrix}$$

$$P_1 = A_{11} * (B_{12} - B_{22}) = \begin{bmatrix} -24 & -24 \\ -88 & -88 \end{bmatrix}$$

$$P_2 = (A_{11} + A_{12}) * B_{22} = \begin{bmatrix} 294 & 304 \\ 758 & 784 \end{bmatrix}$$

$$C_{12} = P_1 + P_2 = \begin{bmatrix} 270 & 280 \\ 670 & 696 \end{bmatrix}$$

$$A_{11} * B_{12} = \begin{bmatrix} 65 & 68 \\ 233 & 244 \end{bmatrix}$$

$$A_{12} * B_{22} = \begin{bmatrix} 205 & 212 \\ 437 & 452 \end{bmatrix}$$

$$(A_{11} * B_{12}) + (A_{12} * B_{22}) = \begin{bmatrix} 270 & 280 \\ 670 & 696 \end{bmatrix} = C_{12}$$

There are two ways to prove $C_{22} = P_5 + P_1 - P_3 - P_7$. You can prove it using the values given for A and B or using a generic method. Both will be shown here.

$$P_1 = A_{11} * (B_{12} - B_{22}) = A_{11} * B_{12} - A_{11} * B_{22} = \begin{bmatrix} -24 & -24 \\ -88 & -88 \end{bmatrix}$$

$$P_3 = (A_{21} + A_{22}) * B_{11} = A_{21} * B_{11} + A_{22} * B_{11} = \begin{bmatrix} 802 & 844 \\ 1106 & 1164 \end{bmatrix}$$

$$P_5 = (A_{11}+A_{22})*(B_{11}+B_{22}) = A_{11}*B_{11}+A_{22}*B_{11}+A_{11}*B_{22}+A_{22}*B_{22} = \begin{bmatrix} 1256 & 1308 \\ 2024 & 2108 \end{bmatrix}$$

$$P_7 = (A_{11}-A_{21})*(B_{11}+B_{12}) = A_{11}*B_{11}+A_{11}*B_{12}-A_{21}*B_{11}-A_{21}*B_{12} = \begin{bmatrix} -640 & -672 \\ -640 & -672 \end{bmatrix}$$

$$C_{22} = A_{21} * B_{12} + A_{22} * B_{22} = \begin{bmatrix} 1070 & 1112 \\ 1470 & 1528 \end{bmatrix}$$

Using the values, we get

$$P_5 + P_1 - P_3 - P_7 = \begin{bmatrix} 1070 & 1112 \\ 1470 & 1528 \end{bmatrix} = C_{22}$$

Using the generic method, we get

$$\begin{aligned} & P_5 + P_1 - P_3 - P_7 \\ &= A_{11} * B_{11} + A_{11} * B_{12} + A_{11} * B_{22} - A_{11} * B_{22} + A_{22} * B_{11} + A_{22} * B_{22} \\ &- (A_{11} * B_{11} + A_{11} * B_{12} - A_{21} * B_{11} + A_{21} * B_{11} + A_{22} * B_{11} - A_{21} * B_{12}) \\ &= A_{22} * B_{22} - (-A_{21} * B_{12}) = A_{22} * B_{22} + A_{21} * B_{12} = C_{22} \end{aligned}$$

Q4 → Matrix Multiplication

We begin by partitioning matrix A and matrix B into square blocks of size $n \times n$.

Let:

$$A = [A_1 \ A_2 \ A_3]$$

where each A_i is an $n \times n$ matrix.

Similarly, partition matrix B into three blocks:

$$B = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

where each B_i is an $n \times n$ matrix.

The matrix product $C = A \times B$ can be expressed as:

$$C = [A_1 \ A_2 \ A_3] \times \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

This expands into:

$$C = A_1B_1 + A_2B_2 + A_3B_3$$

Each multiplication A_iB_i is a multiplication between two $n \times n$ matrices, where we can apply Strassen's algorithm.

For each pair A_i and B_i, we use Strassen's algorithm to compute the matrix multiplication:

$$C_i = A_iB_i$$

Strassen's algorithm for multiplying two $n \times n$ matrices has a time complexity of $O(n^{\log_2 7}) \approx O(n^{2.81})$.

Once the products $C_1 = A_1B_1$, $C_2 = A_2B_2$, and $C_3 = A_3B_3$ are computed using Strassen's algorithm, we sum these matrices to get the final result:

$$C = C_1 + C_2 + C_3$$

The time complexity of the solution is dominated by the three Strassen matrix multiplications, each taking $O(n^{2.81})$, and the matrix additions, which take $O(n^2)$ time.

Thus, the total time complexity is: $T(n) = O(n^{2.81})$

```
EXTRACT_SUBMATRIX(M, row_start, row_end, col_start, col_end)
    // Initialize the submatrix with the correct dimensions
    num_rows = row_end - row_start + 1
    num_cols = col_end - col_start + 1
    submatrix = NEW_MATRIX(num_rows, num_cols)

    // Copy the values from the original matrix to the submatrix
    for i = 1 to num_rows
        for j = 1 to num_cols
            // Copy the element from the original matrix to the submatrix
            submatrix[i][j] = M[row_start + i - 1][col_start + j - 1]

    return submatrix


MULTIPLY_MATRICES(A, B)
    // Initialize an n x n matrix filled with zeroes
    answer = NEW_MATRIX(n, n)

    for i = 1 to 3
        // Extract submatrices

        // From A: all rows, columns (i-1)*n + 1 to i*n
        A_submatrix = EXTRACT_SUBMATRIX(A, 1, n, (i-1)*n + 1, i*n)

        // From B: rows (i-1)*n + 1 to i*n, all columns
        B_submatrix = EXTRACT_SUBMATRIX(B, (i-1)*n + 1, i*n, 1, n)

        // Perform Strassen multiplication and accumulate the result
        answer = MATRIX_ADD(answer, STRASSEN(A_submatrix, B_submatrix))

    return answer
```