Problem Set 3

Due date: Electronic submission of the pdf file of this homework is due on **2/7/2025 before 11:59pm** on ecampus.

Name: **Chayce J Leonard**

Resources. (All people, books, articles, web pages, etc. that have been consulted when producing your answers to this homework)
Justin Cantu Notes (Math 300 - I did not take CSCE222 I apologize)
Wikipedia
Persuall
Class Textbook

On my honor, as an Aggie, I have neither given nor received any unauthorized aid on any portion of the academic work included in this assignment. Furthermore, I have disclosed all resources (people, books, web sites, etc.) that have been used to prepare this homework.

Signature: **[Chayce Leonard]**

Make sure that you describe all solutions in your own words.

Read chapters 2 and 4 in our textbook before attempting to solve these problems.

Problem 1 (20 points)**.** Use mathematical induction to show that when n is an exact power of 2, the solution of the recurrence

$$T(n) = \begin{cases} 2 & \text{if } n = 2, \\ 2T(n/2) + n & \text{if } n = 2^k \text{ for } k > 1, \end{cases}$$

is $T(n) = n \log_2 n$.

Solution. Base Case (n = 2):
The recurrence directly gives $T(2) = 2$. Using the proposed formula:

$$T(2) = 2 \log_2 2 = 2 \times 1 = 2$$

This matches the base case exactly.

Inductive Hypothesis:
Assume the formula holds for $n = 2^k$, that is:

$$T(2^k) = 2^k \log_2 2^k = k \cdot 2^k$$

Inductive Step:
Consider $n = 2^{k+1}$. By the recurrence relation:

$$T(2^{k+1}) = 2T\left(2^{k+1}/2\right) + 2^{k+1}$$

Simplify using algebraic manipulation:

$$= 2T(2^k) + 2^{k+1}$$

Substitute using the inductive hypothesis:

$$= 2(k \cdot 2^k) + 2^{k+1}$$

$$= k \cdot 2^{k+1} + 2^{k+1}$$

Factor common terms:

$$= 2^{k+1}(k+1)$$

Which matches the formula $T(n) = n \log_2 n$ since:

$$2^{k+1} \log_2 2^{k+1} = 2^{k+1}(k+1)$$

Thus by mathematical induction, $T(n) = n \log_2 n$ holds for all $n = 2^k$ where $k \geq 1$.

Problem 2 (20 points)**.** We can express insertion sort as a recursive procedure as follows. In order to sort $A[1..n]$, we recursively sort $A[1..n-1]$ and then insert $A[n]$ into the sorted array $A[1..n-1]$. Write a recurrence for the running time of this recursive version of insertion sort.

Solution. Base Case (n = 1):
For an array with one element, the sorting is trivially complete:

$$T(1) = O(1)$$

Recursive Case (n ¿= 1):
The recurrence consists of two components:

1. Time to sort the first $n-1$ elements recursively: $T(n-1)$
2. Time to insert the n-th element into the sorted subarray $A[1..n-1]$: $O(n)$

This gives the recurrence relation:

$$T(n) = T(n-1) + O(n)$$

Problem 3 (20 points)**.** V. Pan has discovered a way of multiplying 68×68 matrices using $132,464$ multiplications, a way of multiplying 70×70 matrices using $143,640$ multiplications, and a way of multiplying 72×72 matrices using $155,424$ multiplications. Which method yields the best asymptotic running time when used in a divide-and-conquer matrix-multiplication algorithm? How does it compare to Strassen's algorithm?

Solution. To determine which method yields the best asymptotic running time, we analyze the divide-and-conquer matrix multiplication algorithm for each of the given methods. In a divide-and-conquer approach, the running time is determined by the recurrence:

$$T(n) = aT\left(\frac{n}{b}\right) + O(n^d)$$

where a is the number of subproblems, b is the factor by which the problem size is divided, and d is the exponent for the cost of combining the results.

For matrix multiplication, $b = 2$ because the matrix is divided into four quadrants, and a is the number of multiplications required for the subproblems. The asymptotic running time is determined by the solution to the recurrence, which is:

$$T(n) = O\left(n^{\log_b a}\right)$$

Thus, the key is to compute $\log_2 a$ for each method.

1. Method for 68×68 matrices:
Here, $a = 132,464$. The asymptotic running time is:

$$\log_2 132,464 \approx 17.03$$

Thus, the running time is $O(n^{17.03})$.

2. Method for 70×70 matrices:
Here, $a = 143,640$. The asymptotic running time is:

$$\log_2 143,640 \approx 17.13$$

Thus, the running time is $O(n^{17.13})$.

3. Method for 72×72 matrices:
Here, $a = 155,424$. The asymptotic running time is:

$$\log_2 155,424 \approx 17.23$$

Thus, the running time is $O(n^{17.23})$.

Comparison:
The method for 68×68 matrices yields the best asymptotic running time, as it has the smallest $\log_2 a$ value (17.03).

Comparison to Strassen's Algorithm:
Strassen's algorithm reduces the number of multiplications to $a = 7$ for 2×2 matrices. The asymptotic running time is:

$$\log_2 7 \approx 2.81$$

Thus, Strassen's algorithm has a significantly better asymptotic running time of $O(n^{2.81})$ compared to the methods discovered by V. Pan, which all have $\log_2 a > 17$. This demonstrates that Strassen's algorithm is far more efficient asymptotically.

Problem 4 (20 points)**.** Show how to multiply the complex numbers $a+bi$ and $c+di$ using only three multiplications of real numbers. The algorithm should take a, b, c, and d as input and produce the real component $ac-bd$ and the imaginary component $ad+bc$ separately. [Hint: First study Karatsuba's integer multiplication algorithm.]

Solution. To multiply two complex numbers $a+bi$ and $c+di$, the standard formula is:

$$(a+bi)(c+di) = (ac-bd) + (ad+bc)i$$

This requires four real multiplications: ac, bd, ad, and bc. However, using a method inspired by Karatsuba's algorithm, we can reduce the number of real multiplications to three.

Algorithm: 1. Compute $p_1 = a \cdot c$ (one multiplication). 2. Compute $p_2 = b \cdot d$ (one multiplication). 3. Compute $p_3 = (a+b) \cdot (c+d)$ (one multiplication).

Using these three products, we can derive the real and imaginary components:

$$\text{Real part: } \mathrm{Re} = p_1 - p_2$$

$$\text{Imaginary part: } \mathrm{Im} = p_3 - p_1 - p_2$$

Explanation: - The real part of the product is $ac-bd$. From the computed values, $p_1 = ac$ and $p_2 = bd$, so the real part is simply $p_1 - p_2$. - The imaginary part of the product is $ad+bc$. Expanding $p_3 = (a+b)(c+d)$, we get:

$$p_3 = ac + ad + bc + bd$$

From this, subtract $p_1 = ac$ and $p_2 = bd$ to isolate $ad + bc$:

$$p_3 - p_1 - p_2 = ad + bc$$

Final Result: Using only three real multiplications (p_1, p_2, and p_3), we can compute the product of two complex numbers:

$$\text{Real part: } \mathrm{Re} = p_1 - p_2$$

$$\text{Imaginary part: } \mathrm{Im} = p_3 - p_1 - p_2$$

This method reduces the number of real multiplications from four to three, while still producing the correct result.

Problem 5 (20 points)**.** Use the master method to show that the solution to the binary-search recurrence

$$T(n) = T(n/2) + \Theta(1)$$

is $T(n) = \Theta(\lg n)$. Clearly indicate which case of the Master theorem is used.

Solution. We are tasked with solving the recurrence:

$$T(n) = T\left(\frac{n}{2}\right) + \Theta(1)$$

using the Master Theorem and showing that the solution is $T(n) = \Theta(\lg n)$.

Step 1: Identify parameters for the Master Theorem.
The general form of the Master Theorem is:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where: - a is the number of subproblems, - b is the factor by which the problem size is divided, - $f(n)$ is the cost of the work done outside the recursive calls.

For the given recurrence: - $a = 1$ (one subproblem), - $b = 2$ (the problem size is halved), - $f(n) = \Theta(1)$ (constant work outside the recursion).

Step 2: Compute the critical value $p = \log_b a$.
The critical value p is given by:

$$p = \log_b a = \log_2 1 = 0$$

Step 3: Compare $f(n)$ to $n^p = n^0 = \Theta(1)$.
The function $f(n) = \Theta(1)$ is asymptotically equal to $n^p = \Theta(1)$. This corresponds to **Case 2** of the Master Theorem, which states:

$$\text{If } f(n) = \Theta(n^p \log^k n) \text{ with } k = 0, \text{ then } T(n) = \Theta(n^p \log^{k+1} n).$$

Step 4: Apply Case 2 of the Master Theorem.
Since $f(n) = \Theta(1)$, $n^p = \Theta(1)$, and $k = 0$, the solution to the recurrence is:

$$T(n) = \Theta(\log n)$$

Conclusion: Using the Master Theorem, we have shown that the solution to the recurrence $T(n) = T(n/2) + \Theta(1)$ is:

$$T(n) = \Theta(\lg n)$$

Work out your own solutions, unless you want to risk an honors violation!

Checklist:
☐ Did you add your name?
☐ Did you disclose all resources that you have used?
(This includes all people, books, websites, etc. that you have consulted)
☐ Did you sign that you followed the Aggie honor code?
☐ Did you solve all problems?
☐ Did you submit the pdf file of your homework? Check!