Problem Set 3

> **Due date:** Electronic submission of the pdf file of this homework is due on **2/9/2023 before 11:59pm** on ecampus.

Name: Jawahar Sai Nathani

Resources. (All people, books, articles, web pages, etc. that have been consulted when producing your answers to this homework)

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: Jawahar Sai Nathani

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. First let's check for base cases when n = 2 and 4
<u>n = 2</u>
$T(n) = n \log_2 n$
$\Rightarrow T(2) = 2 \log_2 2$
$\Rightarrow T(2) = 2$
<u>n = 4</u>
$T(4) = 4 \log_2 4 = 4 * 2 = 8$
According to the recurrence $T(4) = 2T(4/2) + 4$
$T(4) = 2 * T(2) + 4 = 2 * 2 + 4 = 8$
Relationship holds for both n = 2 and 4.
<u>**Inductive Case**</u>
let's assume $T(n) = n \log_2 n$, when $n = 2^k$. For $n = 2^{k+1}$

$$\begin{aligned} T(2^{k+1}) &= 2T(2^{k+1}/2) + 2^{k+1} \\ &= 2T(2^k) + 2^{k+1} \\ &\text{substitute } \ T(2^k) = 2^k \log_2 2^k \ \text{ according to our assumption} \\ &= 2\left(2^k \log_2\left(2^k\right)\right) + 2^{k+1} \\ &= 2^{k+1} \log_2(2^k) + 2^{k+1} \\ &= 2^{k+1}\left(\log_2(2^k) + 1\right) \\ &= 2^{k+1}\left(\log_2(2^k) + \log_2 2\right) \\ &= 2^{k+1} \log_2(2^{k+1}) \\ &= n \log_2(n) \end{aligned}$$

Therefore, $T(n) = n \log_2 n$ for the above recurrence holds.

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. In insertion sort, we first sort the starting $n-1$ elements and then insert A[n] into the sorted array.
To sort the array of $n-1$ elements it takes $T(n-1)$ running time and to insert A[n] into the sorted array, it takes $\Theta(n)$ runtime, since we need to compare A[n] with all the elements in the array to find the correct index.
Hence, insertion sort runtime for an array with n elements is

$$T(n) = T(n-1) + \Theta(n)$$

Whereas $n = 1$ is a special case. For $n = 1$ since there are no other elements to sort, we just need to insert the element into the array. Runtime for $n = 1$ is $\Theta(1)$. Therefore

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \\ T(n-1) + \Theta(n) & \text{if } n > 1 \end{cases}$$

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. The recurrence rule for time complexity of an algorithm with 'b' sub-problems and 'a' multiplications is

$$T(n) = aT(n/b)$$

Applying master theorem, T(n) = $\Theta(n^{\log_b a})$. Time complexity for the first method is
$\Theta(n^{\log_{68} 132464})$ and $\log_{68} 132464 \approx 2.7951284874$
Time complexity for 2nd method is
$\Theta(n^{\log_{70} 143640})$ and $\log_{70} 143640 \approx 2.7951226897$
Time complexity for 3nd method is
$\Theta(n^{\log_{72} 155424})$ and $\log_{72} 155424 \approx 2.7951473911$
and Time complexity for Strassen's algorithm with 7 multiplications and 2 sub-processes is
$\Theta(n^{\log_2 7})$ and $\log_2 7 \approx 2.81$
Therefore, method 2 yields the best asymptotic running time and is better than Strassen's algorithm.

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. Let's consider
$P_1 = (a+b)(c-d) = ac - ad + bc - bd$
$P_2 = ad$
$P_3 = bc$

$$\begin{aligned} P_1 + P_2 - P_3 &= ac - ad + bc - bd + ad - bc \\ &= ac - bd \\ &= \text{real component} \\ P_2 + P_3 &= ad + bc \\ &= \text{imaginary component} \end{aligned}$$

$\Rightarrow (a+bi)(c+di) = (P_1 + P_2 - P_3) + i(P_2 + P_3)$
P_1, P_2 and P_3 all have one multiplication in each. Therefore, 2 complex numbers can be multiplied only using 3 multiplications following the above method.

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. $T(n) = aT(n/b) + f(n)$ when $a = 1, b = 2, f(n) = \Theta(1)$.
$\Theta(n^{\log_b a}) = \Theta(n^{\log_2 1}) = \Theta(n^0) = \Theta(1)$
$f(n) = \Theta(n^{\log_b a})$.
According to the second asymptotic bound of Master's theorem if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{log_b a} \lg n)$
$\Rightarrow T(n) = \Theta(n^{\log_2 1} \lg n) = \Theta(1 * \lg n) = \Theta(\lg n)$.
Therefore, $T(n) = \Theta(\lg n)$.

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!