Name: Amey Hengle
UID: 122283961
Email: ameyhen@umd.edu

Homework Assignment 2 – MSML606

Problem 1

(a) $T(n) = 2T(n/2) + n^2; \quad T(1) = 1$

I will use the Master Theorem to solve this.

- First, we need to identify the core parameters from the recurrence relation: $a = 2$ (number of subproblems), $b = 2$ (factor by which the problem size is reduced), and $f(n) = n^2$. This is the cost of the work done outside the recursive calls
- Next to calculate the critical exponent to determine the baseline polynomial growth: $n^{\log_b a} = n^{\log_2 2} = n^1 = n$.
- Comparing $f(n) = n^2$ with n^1, we can clearly see that $f(n)$ grows polynomially faster than the number of leaves. Specifically, $n^2 = \Omega(n^{1+\epsilon})$ for $\epsilon = 1$.
- This places the recurrence squarely in **Case 3** of the Master Theorem (the "root-heavy" case). Because the regularity condition $af(n/b) \leq cf(n)$ also holds ($2(n/2)^2 = n^2/2 \leq cn^2$ for $c < 1$), the work done at the root dominates the total runtime
- Therefore, the asymptotic complexity is $T(n) = \Theta(n^2)$.

(b) $T(n) = T(n/2) + T(n/3) + n; \quad T(0) = 1$

Since we cannot apply the Master Theorem to this relation because the subproblems are divided into unequal sizes ($n/2$ and $n/3$). Instead, I will analyze this by conceptalizing a recursion tree

- At the root level (Level 0), the algorithm performs n amount of work.
- At level 1, the work is the sum of the two recursive branches: $n/2 + n/3 = \frac{5}{6}n$.
- At level 2, the work evaluates to $(n/4 + n/6) + (n/6 + n/9) = \frac{25}{36}n$, which is $(\frac{5}{6})^2 n$.
- I noticed that the total work done per level forms a geometric series: so total work $= n \sum_{i=0}^{\infty} (\frac{5}{6})^i$
- Because the common ratio of this geometric series ($r = 5/6$) is strictly less than 1, the series converges to a constant value ($1/(1 - \frac{5}{6}) = 6$). This implies that the total work is bounded by a constant multiple of the work done at the root
- Therefore, the overall time complexity is $T(n) = \Theta(n)$.

Problem 2

Based on the problem description, we are given following components to construct recurrence relation:

- "Create 5 separate arrays" –¿ so $a = 5$.
- "Each one third of the size" –¿ so $b = 3$.
- "Takes linear time in n to accomplish" –¿ so non-recursive overhead is $f(n) = cn$ or $\Theta(n)$.

Therefore we get recurrence relation: $T(n) = 5T(n/3) + \Theta(n)$.

To solve this, I am applying the Master Theorem. I compute the critical exponent: $n^{\log_b a} = n^{\log_3 5}$. Since $\log_3 5 \approx 1.46$, the leaves grow at a rate of $n^{1.46}$.

When I compare $f(n) = n^1$ to the leaf growth $n^{1.46}$, I find that $n^{1.46}$ is polynomially larger. This triggers **Case 1** of the Master Theorem. This means the bulk of the computational work occurs at the base of the recursion tree.

Ths, the final complexity is $T(n) = \Theta(n^{\log_3 5})$.

Problem 3

- Following the hint to apply the Master Theorem, the constants are: $a = 7$, $b = 2$, and $f(n) = 3n^2 + 2$.
- For asymptotic analysis, we can drop the constants and lower-order terms, treating $f(n)$ simply as n^2
- To calculate the critical exponent: $n^{\log_b a} = n^{\log_2 7}$. Since $2^2 = 4$ and $2^3 = 8$, Since $\log_2 7$ is roughly 2.81.
- Comparing $f(n) = n^2$ to the critical exponent $n^{2.81}$ it is evident that $n^{2.81}$ is polynomially larger than n^2 by a factor of $n^{0.81}$. This perfectly satisfies **Case 1** of the Master Theorem.
- Consequntly, the algorithm's runtime is determined by the number of leaves in the recursion tree, giving an asymptotic bound of $T(n) = \Theta(n^{\log_2 7})$.

Problem 4

The Master Theorem requires a recurrence of the form $T(n) = aT(n/b) + f(n)$ where $a \geq 1$, $b > 1$, and $f(n)$ is an asymptotically positive polynomial function.

- $T(n) = 2T(n/2) + 2^n$: Here, the driving function $f(n) = 2^n$ is an exponential function, not a polynomial. While 2^n certainly grows faster than $n^{\log_2 2}$, standard textbook definitions (like CLRS) dictate that $f(n)$ must be polynomially bounded for the basic Master Theorem to apply. This fails.
- $T(n) = 2T(n/3) + \sin(n)$: The Master Theorem requires $f(n)$ to be asymptotically positive. Because the sine function continuously oscillates between -1 and 1, it violates this positivity and regularity condition. This fails.
- $T(n) = T(n-2) + 2n^2 + 1$: This is a subtractive recurrence relation $(n-2)$. The Master Theorem is explicitly designed for "Divide and Conquer" algorithms where the problem size is divided by a constant factor (n/b), not subtracted. This fails.
- since none of the options satisfy the strict mathematical constraints of the Master Theorem, the answer is **(d) None of these**.

Problem 5

(a) $T(n) = 2T(n/2) + n$

- The root does n work. At Level 1, there are 2 nodes each doing $n/2$ work, totaling n. At Level 2, there are 4 nodes each doing $n/4$ work, totaling n.
- Every level contributes exactly n work. Since the tree depth is $\log_2 n$, the total complexity is the sum of these levels: $T(n) = \Theta(n \log n)$.

(b) $T(n) = 3T(n/2) + n$

- The root does n work. Level 1 does $3 \times (n/2) = 1.5n$ work. Level 2 does $9 \times (n/4) = 2.25n$ work.
- The work per level increases geometrically by a factor of $3/2$. Because the series is growing, the vast majority of the work is concentrated at the leaf nodes. The number of leaves is $3^{\log_2 n} = n^{\log_2 3}$, resulting in $T(n) = \Theta(n^{\log_2 3})$.

c) $T(n) = T(n/2) + T(n/4) + n$

- The root does n work. Level 1 does $n/2 + n/4 = 0.75n$ work.
- The work is decreasing geometrically at each level by a factor of $3/4$. Because this ratio is less than 1, the total work converges to a constant multiple of the root. Thus, the root heavily dominates, yielding $T(n) = \Theta(n)$.

(d) $T(n) = 3T(n/3) + n\log(n)$

- The root does $n \log n$ work. At Level 1, the work is $3\times((n/3)\log(n/3))$, which simplifies to $n\log(n) - n\log(3)$. For large n, each level essentially contributes roughly $n \log n$ work.
- There are $\log_3 n$ levels in this tree, and each of the levels contributes $O(n \log n)$ work. Multiplying the work per level by the depth of the tree gives $T(n) = \Theta(n \log^2 n)$.

Resources, Tools, and AI Usage Statement

- I have used Perplexity to assist with formatting the equations in latex
- I prepared and edited this document using Overleaf, and I used its built-in grammar and spell-check tools to improve clarity and correctness.