Fundamental Algorithms 4 - Solution Examples

Exercise 1 (Master Theorem)
Use the master method to give tight asymptotic bounds for the following recurrences.

1. \( T(n) = 9T\left(\frac{n}{3}\right) + n \)
2. \( T(n) = 27T\left(\frac{n}{3}\right) + n^3 \)
3. \( T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n} \)
4. \( T(n) = 2T\left(\frac{n}{2}\right) + n^2 \)
5. \( T(n) = 8T\left(\frac{n}{2}\right) + \frac{n^3}{\log n} \)

Solution:
Please refer to the Chapter 4 Divide-and-Conquer in the textbook (Introduction to Algorithms: CLRS) for the methods of solving recurrences, statement and proof of Master theorem.

1. In this case, \( n^{\log_b a} = n^2 \), \( f(n) = n \). Since \( f(n) \) is polynomially smaller than \( n^{\log_b a} \), case 1 of the master theorem implies that \( T(n) = \Theta(n^2) \).

2. In this case, \( n^{\log_b a} = n^3 \) and \( f(n) = n^3 \). Since \( f(n) \) is asymptotically the same as \( n^{\log_b a} \), case 2 of the master theorem implies that \( T(n) = \Theta(n^3 \log n) \).

3. This case is similar to 2, \( n^{\log_b a} = \sqrt{n} \) and \( f(n) = \sqrt{n} \). Since \( f(n) \) is asymptotically the same as \( n^{\log_b a} \), case 2 of the master theorem implies that \( T(n) = \Theta(\sqrt{n} \log n) \).

4. In this case, \( n^{\log_b a} = n \). Since \( f(n) \) is asymptotically larger than \( n^{\log_b a} \), case 3 of the master theorem asks us to check whether \( af\left(\frac{n}{b}\right) \leq cf(n) \) for some \( c < 1 \) and all \( n \) sufficiently large. This is indeed the case, so \( T(n) = \Theta(f(n)) = \Theta(n^2) \).

5. In this case, \( n^{\log_b a} = n^3 \), \( f(n) = \frac{n^3}{\log n} \). \( f(n) \) is smaller than \( n^{\log_b a} \) but by less than a polynomial factor. Therefore, the master theorem makes no claim about the solution to the recurrence.

Exercise 2 (Recursion Tree Method)
1. Use a recursion tree to determine a good asymptotic upper bound on the following recurrence

\[ T(n) = 4T\left(\frac{n}{2} + 2\right) + n \]

2. Use the substitution method to verify your answer.
Solution:

The subproblem size for a node at depth $i$ is $n/2^i$. Thus, the tree has $\lg n + 1$ levels and $4^{\lg n} = n^2$ leaves.

The total cost over all nodes at depth $i$, for $i = 0, 1, 2, \ldots, \lg n - 1$, is $4^i(n/2^i + 2) = 2'n + 2 \cdot 4^i$.

$$T(n) = \sum_{i=0}^{\lg n-1} (2'n + 2 \cdot 4^i) + \Theta(n^2)$$

$$= \sum_{i=0}^{\lg n-1} 2^i n + \sum_{i=0}^{\lg n-1} 2 \cdot 4^i + \Theta(n^2)$$

$$= \frac{2^{\lg n} - 1}{2 - 1} n + 2 \cdot \frac{4^{\lg n} - 1}{4 - 1} + \Theta(n^2)$$

$$= (2^{\lg n} - 1)n + \frac{2}{3}(4^{\lg n} - 1) + \Theta(n^2)$$

$$= (n - 1)n + \frac{2}{3}(n^2 - 1) + \Theta(n^2)$$

$$= \Theta(n^2).$$

We guess $T(n) \leq c(n^2 - dn)$.

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

$$\leq 4c[(n/2 + 2)^2 - d(n/2 + 2)] + n$$

$$= 4c(n^2/4 + 2n + 4 - dn/2 - 2d) + n$$

$$= cn^2 + 8cn + 16c - 2cdn - 8cd + n$$

$$= cn^2 - cdn + 8cn + 16c - cdn - 8cd + n$$

$$= c(n^2 - dn) - (cd - 8c - 1)n - (d - 2) \cdot 8c$$

$$\leq c(n^2 - dn),$$

where the last step holds for $cd - 8c - 1 \geq 0$. 