Recurrence Relations using Master Method

---

1. Introduction to Basic Concepts

- **Recursion:** A technique where a function calls itself.
- **Recursive Algorithm:** An algorithm that uses recursion.
- **Recurrence Relation:** An equation that represents the time complexity of a recursive algorithm.
- **Master Method (Master Theorem):**
  One of the three standard techniques used to solve recurrence relations and find their time complexity.

2. Standard Form of Master Method

To apply the Master Method, the recurrence relation **must** be written in the following form:

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

Parameters Explanation

- **n:** Size of the input problem
- **a:** Number of subproblems
  - Condition: $a \geq 1$ (number of subproblems cannot be zero or negative)
- **b:** Factor by which the problem size is reduced
  - Condition: $b > 1$ (problem size must decrease)
- **f(n):** Cost of work done outside the recursive calls
  - Examples: dividing the problem, combining results, loop overheads

3. Three-Step Process to Apply Master Method

Step 1: Identify Parameters

- Compare the given recurrence relation with

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

- Identify values of **a**, **b**, and **f(n)**.

Step 2: Calculate

- Compute:

$$n^{log_b a}$$

- This represents the **cost of the recursion tree structure**.

Step 3: Compare and Apply Case

- Compare $f(n)$ with $n^{log_b a}$.
- Based on the comparison, apply **Case 1, Case 2, or Case 3**.

4. The Three Cases of Master Theorem

Instead of memorizing complex definitions, simply compare these two values:

- **Tree Cost:** $n^{log_b a}$
- **Function Cost:** $f(n)$

Case 1: $n^{log_b a} > f(n)$

- **Logic:** The tree cost is greater than the function cost.
- **Action:** The answer is the tree cost.
- **Solution:**

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

Case 2: $n^{log_b a} = f(n)$

- **Logic:** The tree cost is equal to the function cost.
- **Action:** Multiply the tree cost by $logn$.
- **Solution:**

$$T(n) = \Theta\left(n^{log_b a} \cdot logn\right)$$

Case 3: $n^{log_b a} < f(n)$

- **Logic:** The tree cost is less than the function cost.
- **Action:** The answer is the function cost (but first, check the **Regularity Condition**).
- **Regularity Condition:**

$$a \cdot f(n/b) \leq c \cdot f(n), \quad where\ c < 1$$

- **Solution:**

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

Note for students: In Case 3, ensure that $c < 1$ during the check.

Summary Table for the Board

Condition	Comparison	Final Answer (Complexity)
Case 1	$n^{\log_b a} > f(n)$	$\Theta\left(n^{\log_b a}\right)$
Case 2	$n^{\log_b a} = f(n)$	$\Theta\left(n^{\log_b a} \cdot \log n\right)$
Case 3	$n^{\log_b a} < f(n)$	$\Theta(f(n))$

5. Solved Examples

Example 1: Case 1

Problem:

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

Step 1: Identify Parameters

- $a = 2, b = 2, f(n) = n$

Step 2: Calculate

$$n^{\log_2 2} = n^1 = n$$

Step 3: Compare

- $f(n) = n$
- $n^{\log_b a} = n$
- $f(n) < n^{\log_b a}$

Conclusion

- **Case 1 applies**
- **Answer:**

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

Example 2: Case 2

Problem:

$$T(n) = T\left(\frac{n}{1.5}\right) + 1$$

Step 1: Identify Parameters

- $a = 1, b = 1.5, f(n) = 1$

Step 2: Calculate

$$n^{\log_{1.5}1} = n^0 = 1$$

Step 3: Compare

- $f(n) = 1$
- $n^{\log_b a} = 1$

Conclusion

- **Case 2 applies**
- **Answer:**

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

Example 3: Case 3

Problem:

$$T(n) = 4T\left(\frac{n}{2}\right) + n^3$$

Step 1: Identify Parameters

- $a = 4, b = 2, f(n) = n^2$

Step 2: Calculate

$$n^{\log_2 4} = n^2$$

Step 3: Compare

- $f(n) = n^3$
- $n^{\log_b a} = n^2$

Regularity Check

$$af\left(\frac{n}{b}\right) = 4\left(\frac{n}{2}\right)^2 = n^2$$

Choose $c = \frac{1}{2}$, condition holds.

Conclusion

- **Case 3 applies**
- **Answer:**

$$T(n) = \Theta\left(n^2\right)$$