Advanced Algorithms Analysis and Design

Lecture 9

Master Theorem

Dr. Shamila Nasreen

Previous Lectures

- Iterative Algorithms: Analyzing Loops
- Few Analysis Examples
- Analysis of Control Structure
- Recursive calls
- While and Repeat Loop
- Recurrence Relation

Today's Lectures

- Master Theorem
- Master Theorem three cases
- Examples
- Limitations

Master Theorem

- A powerful tool for analyzing the time complexity of **divide-and-conquer** algorithms in the field of **Design and Analysis of Algorithms**.

- The Master Theorem simplifies the process of solving recurrence relations, which describe the runtime of recursive algorithms.

Why the Master Theorem

In algorithm design, many algorithms (e.g., merge sort, binary search) use a **divide-and-conquer** strategy:

1. **Divide**: Split the problem into smaller subproblems.
2. **Conquer**: Solve the subproblems recursively.
3. **Combine**: Merge the solutions to solve the original problem.

- The runtime of such algorithms is often expressed as a **recurrence relation**, which can be complex to solve.
- The Master Theorem provides a straightforward way to determine the time complexity without unrolling the recurrence or building a recursion tree, **saving time** and **effort**.

- Master's theorem can only be applied on decreasing and dividing recurring functions.
- If the relation is not decreasing or dividing, master's theorem must not be applied.

Master Theorem: Divide and Conquer

Masters Theorem for Dividing Functions

Consider a relation of type –

```
T(n) = aT(n/b) + f(n)
```

where, **a >= 1** and **b > 1**,

n – size of the problem

a – number of sub-problems in the recursion

n/b – size of the sub problems based on the assumption that all sub-problems are of the same size.

f(n) – represents the cost of work done outside the recursion -> (nk logn p) ,where k >= 0 and p is a real number;

If the recurrence relation is in the above given form, then there are three cases in the master theorem to determine the asymptotic notations –

Master Theorem: Three Cases

◆ **Case 1: $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$**

Case 1

(i.e., $f(n)$ grows slower than $n^{\log_b a}$)

- Condition: $a > b^k$
- Result:

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

◆ **Case 2: $f(n) = \Theta(n^{\log_b a} \log^p n)$ for some $p \geq -1$**

Case 2

(i.e., $f(n)$ matches $n^{\log_b a}$ up to a polylog factor)

- Condition: $a = b^k$
- Result:
  - If $p > -1$: $T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$
  - If $p = -1$: $T(n) = \Theta(n^{\log_b a} \log \log n)$
  - If $p < -1$: $T(n) = \Theta(n^{\log_b a})$

◆ **Case 3: $f(n) = \Omega(n^{\log_b a + \varepsilon})$ and regularity condition holds**

Case 3

(i.e., $f(n)$ grows faster than $n^{\log_b a}$)

- Condition: $a < b^k$
- Result:
  - If $p \geq 0$: $T(n) = \Theta(n^k \log^p n)$
  - If $p < 0$: $T(n) = \Theta(n^k)$

Master Theorem: Three Cases

- If $a > b^k$, then $T(n) = (n^{\log b\ a})$ [$\log_b a = \log a / \log b$.] Case 1
- If $a = b^k$ Case 2
  - If $p > -1$, then $T(n) = (n^{\log b\ a} \log^{p+1} n)$
  - If $p = -1$, then $T(n) = (n^{\log b\ a} \log \log n)$
  - If $p < -1$, then $T(n) = (n^{\log b\ a})$
- If $a < b^k$, Case 3
  - If $p >= 0$, then $T(n) = (n^k \log^p n)$.
  - If $p < 0$, then $T(n) = (n^k)$

Master Theorem: Example 1

$$T(n) = 8T(n/2) + n^2$$

We will solve it using the **Master Theorem**.

🧠 Step 1: Match to Master Theorem Format

General form:

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

Compare:

- $a = 8$
- $b = 2$
- $f(n) = n^2$

🧠 Step 2: Compute $\log_b a$

$$\log_b a = \log_2 8 = 3$$

🧠 Step 3: Compare $f(n)$ to $n^{\log_b a} = n^3$

We have:

- $f(n) = n^2$
- $n^2 = O(n^{3-\varepsilon})$ for $\varepsilon = 1$

This matches **Case 1** of the Master Theorem: If $f(n) = O(n^{\log_b a - \varepsilon})$, then $T(n) = \Theta(n^{\log_b a})$

Final Answer:

$$T(n) = \Theta(n^3)$$

Master Theorem: Example 2

Consider a recurrence relation given as T(n) = 4T(n/2) + n^2

Step 1: Match with Master Theorem Form

General form of the Master Theorem:

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

From the given recurrence:

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

Step 2: Compute $\log_b a$

$$\log_b a = \log_2 4 = 2$$

Now compare:

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

Since:

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

Step 3: Apply **Master Theorem Case 2**

If:

$$f(n) = \Theta(n^{\log_b a} \log^p n) \quad \text{with } p = 0$$

Then:

$$T(n) = \Theta(n^{\log_b a} \log^{p+1} n) = \Theta(n^2 \log n)$$

$$f(n) = \Theta(n^2 \cdot \log^0 n)$$

Since $\log^0 n = 1$, this means:

$$p = 0$$

Final Answer:

$$T(n) = \Theta(n^2 \log n)$$

Master Theorem: Example 3

Consider a recurrence relation given as $T(n) = 2T(n/2) + n^2$

From the recurrence:

- a=?
- b=?
- f(n)=?

•Compute $\log_b a$

•Which case is it?

Master Theorem: Example 4

Consider a recurrence relation given as $T(n) = 16T(n/4) + n$

From the recurrence:

- a=?
- b=?
- $f(n)$=?

•Compute $\log_b a$

•Which case is it?

Master Theorem: Example 4

Consider a recurrence relation given as T(n) = 16T(n/4) + n

For the recurrence

$$T(n) = 16\,T(\tfrac{n}{4}) + n,$$

we identify:

- $a = 16$
- $b = 4$
- $f(n) = n$

---

1. Compute $\log_b a$

$$\log_b a = \log_4 16 = 2 \quad (\text{because } 4^2 = 16).$$

So $n^{\log_b a} = n^2$.

---

2. Compare $f(n)$ to $n^{\log_b a}$

We have $f(n) = n$. Since

$$n = O(n^{2-\varepsilon})$$

with $\varepsilon = 1$ (because $n = n^{2-1}$), $f(n)$ grows strictly slower than n^2.

3. Master Theorem Case

This is **Case 1** of the Master Theorem: a>b

> If $f(n) = O\left(n^{\log_b a-\varepsilon}\right)$ for some $\varepsilon > 0$, then
>
> $$T(n) = \Theta\left(n^{\log_b a}\right).$$

Here $\log_b a = 2$, so:

$$\boxed{T(n) = \Theta(n^2).}$$

Master Theorem: Example 5

Consider a recurrence relation given as T(n) = 2T(n/2) + nlog n

We solve

$$T(n) = 2\,T\left(\tfrac{n}{2}\right) + n\log n$$

by the Master Theorem.

> ◆ **Case 2: $f(n) = \Theta(n^{\log_b a} \log^p n)$ for some $p \geq -1$**
>
> (i.e., $f(n)$ matches $n^{\log_b a}$ up to a polylog factor)
>
> - Condition: $a = b^k$
> - Result:
>   - If $p > -1$: $T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$
>   - If $p = -1$: $T(n) = \Theta(n^{\log_b a} \log\log n)$
>   - If $p < -1$: $T(n) = \Theta(n^{\log_b a})$

1. **Identify parameters**
   - $a = 2$
   - $b = 2$
   - $f(n) = n\log n$
2. **Compute $\log_b a$:**

$$\log_2 2 = 1,$$

   so $n^{\log_b a} = n^1 = n.$

3. **Match $f(n)$ against $n^{\log_b a}$**

   We have

$$f(n) = n\log n = \Theta\left(n^{\log_b a} \log^1 n\right),$$

Master Theorem: Example 5

Consider a recurrence relation given as T(n) = 2T(n/2) + nlog n

◆ **Case 2:** $f(n) = \Theta(n^{\log_b a} \log^p n)$ **for some** $p \geq -1$

(i.e., $f(n)$ matches $n^{\log_b a}$ up to a polylog factor)

- Condition: $a = b^k$
- Result:
  - If $p > -1$: $T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$
  - If $p = -1$: $T(n) = \Theta(n^{\log_b a} \log \log n)$
  - If $p < -1$: $T(n) = \Theta(n^{\log_b a})$

4. Apply Case 2

   If $f(n) = \Theta\left(n^{\log_b a} \log^p n\right)$, then

$$T(n) = \Theta\left(n^{\log_b a} \log^{p+1} n\right) = \Theta\left(n \log^2 n\right).$$

✅ **Final Answer**

$$\boxed{T(n) = \Theta\left(n \log^2 n\right).}$$

Master Theorem: Example 6

Consider a recurrence relation given as T(n) = 3T(n/4) + nlog n

◆ **Case 3: $f(n) = \Omega(n^{\log_b a+\varepsilon})$ and regularity condition holds**

(i.e., $f(n)$ grows faster than $n^{\log_b a}$)

- Condition: $a < b^k$
- Result:
  - If $p \geq 0$: $T(n) = \Theta(n^k \log^p n)$
  - If $p < 0$: $T(n) = \Theta(n^k)$