DSA - Tutorial 3 (Recurrence Relation: Solving through Substitution Method, Tree Method, & Master Theorem)

25-01-2025

Solving the Recurrence Relation

Consider the recurrence relation:

$$T(n) = 2T\left(\frac{n}{2}\right) + n, \quad \text{where } T(1) = \mathcal{O}(1).$$

We solve this using three methods: the **Tree Method**, the **Substitution Method**, and the **Master Theorem**.

1. Master Theorem

The Master Theorem is used to solve divide-and-conquer recurrence relations of the form:

$$T(n) = aT\left(\frac{n}{b}\right) + n^c,$$

where:

- a: The number of subproblems,
- b: The factor by which the input size is divided,
- c: The exponent of the polynomial representing the cost of work done outside the recursive calls.

To apply the theorem: 1. Compute the value of $p = \log_b a$, where p is the critical exponent that determines the balance between the recursive calls (a) and the cost of splitting/merging (n^c). 2. Compare p and c:

- If $p < c$: The cost of n^c dominates, and $T(n) = \Theta(n^c)$.
- If $p = c$: Both terms contribute equally, and $T(n) = \Theta(n^c \log n)$.
- If $p > c$: The cost of recursion dominates, and $T(n) = \Theta(n^p)$.

Applying Master Theorem

For the recurrence relation:

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

we identify:

- $a = 2$,
- $b = 2$,
- $c = 1$ (since $n^1 = n$).

Calculate $p = \log_b a = \log_2 2 = 1$.

- Compare p and c: Here, $p = c = 1$.
- By the second case of the Master Theorem ($p = c$), the solution is:

$$T(n) = \Theta(n^c \log n) = \Theta(n \log n).$$

Thus, the solution is:

$$T(n) = \mathcal{O}(n \log n).$$

2. Tree Method

The tree method visualizes the recurrence as a recursion tree:

- At the **first level**, the work is:

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

  Work done at this level $= n$.
- At the **second level**, each $T\left(\frac{n}{2}\right)$ expands into $T\left(\frac{n}{4}\right)$, so the total work at this level is:

  $$2 \cdot \frac{n}{2} = n.$$
- At the **third level**, each $T\left(\frac{n}{4}\right)$ expands into $T\left(\frac{n}{8}\right)$, so the total work at this level is:

  $$4 \cdot \frac{n}{4} = n.$$
- At the ***i*-th level**, the work is:

  $$2^i \cdot \frac{n}{2^i} = n.$$

The recursion stops when $n/2^i = 1$, which implies:

$$i = \log_2(n).$$

Total work across all levels is:

$$\text{Total Work} = n + n + \cdots + n \quad (\text{for } \log_2(n) \text{ levels}).$$

Thus, the total work is:

$$T(n) = n \cdot \log_2(n) + \mathcal{O}(1).$$

Therefore, the solution is:

$$T(n) = \mathcal{O}(n \log n).$$

3. Substitution Method

We assume a solution of the form $T(n) = cn \log n$ and verify it.

Substitute into the recurrence:

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

1. Substitute $T\left(\frac{n}{2}\right) = c \cdot \frac{n}{2} \cdot \log\left(\frac{n}{2}\right)$:

$$T(n) = 2 \cdot \left(c \cdot \frac{n}{2} \cdot \log\left(\frac{n}{2}\right)\right) + n.$$

2. Simplify:

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

$$T(n) = cn(\log n - \log 2) + n.$$

$$T(n) = cn \log n - cn \log 2 + n.$$

3. Combine terms:

$$T(n) = cn \log n + n(1 - c \log 2).$$

For large n, the $cn \log n$ term dominates. Therefore, the solution is:

$$T(n) = \mathcal{O}(n \log n).$$