SCS1308 - Foundations of Algorithm

Tutorial - 03

Solving recurrence equations- Part 2

Solving recurrence equations

Techniques for solving recurrence equations:

- Recursion tree method - Discussed Last Week
- Substitution method
- Iteration method
- Master Theorem

Substitution Method

- ➔ Guess the solution.
- ➔ Use induction to find the constants and show that the solution works.

How to find a Guess

- ➔ We can use the recursion tree method to find a guess.

Recursion Tree Method to Find a Guess

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 2T\left(\frac{n}{2}\right) + n & \text{if } n > 1. \end{cases}$$

```
                        T(n)
                        Cost = n
                          |
         --------------------------------
         |                                |
      T(n/2)                           T(n/2)
     Cost = n/2                       Cost = n/2
         |                                |
    ------------                     ------------
    |          |                     |          |
 T(n/4)     T(n/4)                T(n/4)     T(n/4)
 Cost=n/4   Cost=n/4              Cost=n/4   Cost=n/4
    |          |                     |          |
     (continues...)                  (continues...)
```

Level	Number of nodes	Cost per node	Total cost
0	1	n	n
1	2	$n/2$	n
2	4	$n/4$	n
3	8	$n/8$	n
...	...	...	n
$\log n$	n nodes	1	n

So every level contributes n, and there are log n + 1 levels:

$$T(n)=n(\log n+1)$$

Induction Proof

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 2T\left(\frac{n}{2}\right) + n & \text{if } n > 1. \end{cases}$$

T(n)=n(logn+1)

Basis: n = 1 => T(n)=1(log 1 +1) = 1

Inductive Step: Assume for some m>=0 T(m)= m(log m +1)

Then for n=2m

$$\begin{aligned} T(2m) &= 2T(m) + 2m \\ &= 2(m(\log m + 1)) + 2m \\ &= 2m(\log m) + 2m + 2m \\ &= 2m(\log m) + 4m \end{aligned}$$

Rewrite in terms of n=2m:

$$\begin{aligned} T(n) &= n(\log(n/2) + 1) + 2n \\ &= n(\log n - \log 2 + 2) \\ &= n(\log n - 1 + 2) \\ &= n(\log n + 1) \end{aligned}$$

By induction on powers of two, the formula holds for all $n=2^k$, k≥0

Conclusion. T(n)=n(log n+1). Therefore T(n)=O(nlogn).

Iteration Method

We keep on substituting the smaller terms again and again until we reach the base condition and find a pattern from it. Thus the base term can be replaced by its value, and we get the value of the expression.

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 2T\left(\frac{n}{2}\right) + n & \text{if } n > 1. \end{cases}$$

Master's Theorem

The Master Theorem applies to recurrences of the following form:

$$T(n) = aT(n/b) + f(n).$$

where $a \geq 1$ and $b > 1$ are constants and $f(n)$ is a monotonically increasing function and

$f(n)$ is $O(n^d)$ where $d>=0$ Then,

Case 1 : if $a < b^d$ => $T(n)$ is $O(n^d)$

Case 2 : if $a = b^d$ => $T(n)$ is $O(n^d \log n)$

Case 3 : if $a > b^d$ => $T(n)$ is $O(n^{\log_b a})$

Activity

For each of the following recurrences, give an expression for the runtime T (n) if the recurrence can be solved with the Master Theorem. Otherwise, indicate that the Master Theorem does not apply.

1. $T(n) = 3T(n/2) + n^2$

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

3. $T(n) = T(n/2) + 2^n$

4. $T(n) = 2^nT(n/2) + n^n$

5. $T(n) = 16T(n/4) + n$

Solutions

1. $T(n) = 3T(n/2) + n^2 \Longrightarrow T(n) = \Theta(n^2)$ (Case 3)

2. $T(n) = 4T(n/2) + n^2 \Longrightarrow T(n) = \Theta(n^2 \log n)$ (Case 2)

3. $T(n) = T(n/2) + 2^n \Longrightarrow \Theta(2^n)$ (Case 3)

4. $T(n) = 2^nT(n/2) + n^n \Longrightarrow$ Does not apply (a is not constant)

5. $T(n) = 16T(n/4) + n \Longrightarrow T(n) = \Theta(n^2)$ (Case 1)

Thank you