Text:

1.1. The Master Method

We shall now look at a method called *master method/theorem* which is a *cook book* for many well-known recurrence relations. It presents a framework and formulae using which solutions to many recurrence relations can be obtained very easily. Almost all recurrences of type $T(n) = aT(n - b) + f(n)$ and $T(n) = aT(n/b) + f(n)$ can be solved easily by doing a simple check and identifying one of the many cases mentioned in the following theorem. We shall look at the theorems separately in the following sub sections.

1.1.1. *Decreasing functions:*

The master theorem is a formula for solving recurrences of the form $T(n) = aT(n - b) + f(n)$, where $a \geq 1$ and $b > 0$ and $f(n)$ is asymptotically positive. (Asymptotically positive means that the function is positive for all sufficiently large n.)
This recurrence describes an algorithm that divides a problem of size n into sub problems, each of size n-b, and solves them recursively.

The theorem is as follows:

If $T(n) = a\, T(n\text{-}b) + f(n)$, where $a \geq 1$, $b > 0$, & $f(n) = O(n^k)$, and $k \geq 0$

Case 1: ***if a = 1,***
$T(n) = O(n * f(n))$ or $O(n^{k+1})$

E.g. 1) $T(n) = T(n - 1) + 1$ — $O(n)$
2) $T(n) = T(n - 1) + n$ — $O(n^2)$
3) $T(n) = T(n - 1) + \log n$ — $O(n \log n)$

Case 2: ***if a > 1,***
$T(n) = O(a^{n/b} * f(n))$ or $O(a^{n/b} * n^k)$

E.g. 1) $T(n) = 2T(n - 1) + 1$ — $O(2^n)$
2) $T(n) = 3T(n - 1) + 1$ — $O(3^n)$
3) $T(n) = 2T(n - 1) + n$ — $O(n\, 2^n)$

Case 3: ***if a < 1,***
$T(n) = O(f(n))$ or $O(n^k)$

1.1.2. *Dividing functions:*

The master theorem is a formula for solving recurrences of the form $T(n) = aT(n/b)+f(n)$, where $a \geq 1$ and $b > 1$ and $f(n)$ is asymptotically positive.
This recurrence describes an algorithm that divides a problem of size n into sub problems, each of size n/b, and solves them recursively. (Note that n/b might not be an integer, but replacing $T(n/b)$ with $T(\lfloor n/b \rfloor)$ or $T(\lceil n/b \rceil)$ does not affect the asymptotic behavior of the recurrence. So we will just ignore floors and ceilings here.)

The theorem is as follows:

Given *T(n) = aT(n/b)+f(n)*, where *a ≥ 1 and b > 1* and *f(n)* = *Θ(n^k log^p n)*

Consider $log_b\,(a)$ & *k*

Case 1: *If $\boldsymbol{log_b\,(a)}$ > k => T(n) = Θ($n^{log_b(a)}$)*

Case 2: *If $\boldsymbol{log_b\,(a)}$ = k &*

- ***2.1: If p > -1 => T(n) = Θ(n^k log^{p+1} n)***
- ***2.2: If p = -1*** **=> T(n) = Θ(n^k log log n)**
- ***2.3: If p < -1 => T(n) = Θ(n^k)***

Case 3: *If $\boldsymbol{log_b\,a}$ < k &*

- ***3.1: If p > 0 => T(n) = Θ(n^k log^p n)***
- ***3.2: If p ≤ 0 => T(n) = Θ(n^k)***

Examples (Case 1):

Example 10.1:

T(n) = 2T(n/2) + 1

Here, a = 2, b = 2,

f(n) = Θ (1) = Θ (n^0 log^0 n), ∴ k = 0 & p = 0

Now, $\boldsymbol{log_b\,(a) = log_2\,(2)}$ ***= 1 > k***

∴, Case 1 is satisfied

T(n) = Θ(n^1)

Examples (Case 2):

Example 10.2:

T(n) = 2T(n/2) + n / log n

Here, a = 2, b = 2,

f(n) = Θ (n ***log^{-1} n****), ∴ k = 1 & p = -1*

Now, $\boldsymbol{log_b\,(a) = log_2\,(2)}$ ***= 1 = k & p = -1***

∴, Case 2.3 is satisfied

T(n) = Θ(n^k log log n) *= Θ (n log log n)*

Example 10.3:

T(n) = 2T(n/2) + n / log^2 n

Here, a = 2, b = 2,

f(n) = Θ (n ***log^{-2} n****), ∴ k = 1 & p = -2*

Now, $\boldsymbol{log_b\,(a) = log_2\,(2)}$ ***= 1 = k & p < -1***

∴, Case 2.2 is satisfied

T(n) = Θ(n^k) *= Θ (n)*

Examples (Case 3):

Example 10.4:

T(n) = 2T(n/2) + n^2 log n

Here, a = 2, b = 2,

f(n) = Θ (n^2 log^1 n), ∴ k = 2 & p = 1

Now, $\boldsymbol{log_b\,(a) = log_2\,(2)}$ ***= 1 < k & p > 0***

∴, Case 3.1 is satisfied

T(n) = Θ(n^k log^p n) = *Θ(n^2 log n)*

Example 10.5:

$T(n) = 4T(n/2) + n^3$

Here, $a = 4$, $b = 2$,

$f(n) = \Theta(n^3) = \Theta(n^3 \log^0 n)$, $\therefore$ $k = 3$ & $p = 0$

Now, $\log_b(a) = \log_2(4) = 2 < k$ **& $p = 0$**

$\therefore$, Case 3.2 is satisfied

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