Master Method

1 Definition

- $T(n)$: upper-bound of function at level n
- **Recurrance:** expression of $T(n)$ in terms of runtime of recursive calls.
- **Base case:** $T(1) \leq k$
- $\forall n > 1 : T(n) \leq \text{(recursive work)} + \text{(current work)}$

For larger, non-base case sizes of n:

$$T(n) \leq aT\left(\frac{n}{b}\right) + O(n^d) \tag{1}$$

There are a recursive calls on inputs of size n/b, where the work done in the *combination step* of the algorithm is of the magnitude n^d.

This leads to the generic **Master Method** for divide-and-conquer algorithms:

$$T(n) = \begin{cases} O(n^d \cdot \log n) & : a = b^d \\ O(n^d) & : a < b^d \\ O(n^{\log_b a}) & : a > b^d \end{cases} \tag{2}$$

Note: In case 1, the base of the logarithm doesn't matter because the difference is a constant factor.

2 Proof

Given the following recurrence:

$$T(n) = aT\left(\frac{n}{b}\right) + cn^d \tag{3}$$

$$T(1) = f \tag{4}$$

expanding it out a few iterations for $T(n)$ reveals the pattern.

$$T(n) = aT\left(\frac{n}{b}\right) + cn^d \tag{5}$$

$$= a\left[aT\left(\frac{n}{b^2}\right) + c\left(\frac{n}{b}\right)^d\right] + cn^d \tag{6}$$

$$= a\left[a\left[aT\left(\frac{n}{b^3}\right) + c\left(\frac{n}{b^2}\right)^d\right] + c\left(\frac{n}{b}\right)^d\right] + cn^d \tag{7}$$

$$= a^3T\left(\frac{n}{b^3}\right) + a^2c\left(\frac{n}{b^2}\right)^d + cn^d \tag{8}$$

Master Method

Let $k = \log_b n$ be the number of iterations in the recurrence before T reaches its base case.

$$\begin{aligned}
T(n) &= a^k f + \sum_{i=0}^{k-1} a^i c \left(\frac{n}{b^i}\right)^d && (9)\\
&= a^k f + cn^d \sum_{i=0}^{k-1} \frac{a^i}{b^{id}} && (10)\\
&= a^k f + cn^d \sum_{i=0}^{k-1} \left(\frac{a}{b^d}\right)^i && (11)\\
&= a^k f + cn^d \left(\frac{\left(\frac{a}{b^d}\right)^k - 1}{\frac{a}{b^d} - 1}\right) && (12)\\
&= n^{\log_b a} f + cn^d \left(\frac{\frac{a^{\log_b n}}{n^d} - 1}{\frac{a}{b^d} - 1}\right) && (13)\\
&= n^{\log_b a} f + \frac{ca^{\log_b n} - cn^d}{\frac{a}{b^d} - 1} && (14)\\
T(n) &= \left(\frac{c}{\frac{a}{b^d} - 1} + f\right) n^{\log_b a} - \frac{cn^d}{\frac{a}{b^d} - 1} && (15)
\end{aligned}$$

Note that the geometric sum formula is not valid when $a/b^d = 1$ so the final formula doesn't work for that case, but it is a special case that makes the original summation trivial to solve. The full definition of the Master Theorem, accounting for all 3 possible cases, is as follows:

$$T(n) = \begin{cases} \left(f + \dfrac{c}{ab^{-d} - 1}\right) n^{\log_b a} - \dfrac{cn^d}{ab^{-d} - 1} = \begin{cases} \Theta(n^{\log_b a}) & a > b^d \\ \Theta(n^d) & a < b^d \end{cases} \\ n^d(f + c\log_b n) = \Theta(n^d \log_b n) \qquad a = b^d \end{cases} \tag{16}$$

Superposition If the work done in each recursive call isn't perfectly modeled by cn^d because there are extra terms, use the theorem on each part of the recurrence letting $f = 0$, add the solutions together, and then finally add $fn^{\log_b a}$.