Master Theorem

The runtime of sequential algorithms can be derived by constructing a runtime table; however, calculating the runtime of recursive algorithms requires different methods. One popular method is called the *master theorem*, which is particularly useful for analyzing divide-and-conquer algorithms that split the input into smaller subproblems.

Specifically, if the algorithm can be written as a recurrence function of the form

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

where

- a is the number of recursive calls per caller instance.
- $\frac{n}{b}$ (or $\left\lfloor \frac{n}{b} \right\rfloor$ or $\left\lceil \frac{n}{b} \right\rceil$ if b does not divide n evenly) represents the size of each subproblem,
- $f(n)$ is a function that describes the cost outside the recursive calls

the master theorem provides a straightforward way to determine the asymptotic behavior of the recurrence. It states:

> Let $a \geq 1$ and $b > 1$ be constants, $f(n)$ be an asymptotically positive function, and $T(n)$ be defined on the nonnegative integers by the recurrence
>
> $$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$
>
> then $T(n)$ is bounded asymptotically as follows

case 1. If $f(n) = O(n^{(\log_b a)-\epsilon})$, then $T(n) = \Theta(n^{\log_b a})$

case 2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$

case 3. If $f(n) = \Omega(n^{(\log_b a)+\epsilon})$ and $af\left(\frac{n}{b}\right) \leq cf(n)$ for some constant $0 < c < 1$ when n is sufficiently large, then $T(n) = \Theta(f(n))$

> where $\lg n = \log_2 n$, $0 < \epsilon \leq 1$, and additional criteria in case 3 is the *regularity condition*.

Specifically, the master theorem compares the functions $f(n)$ and $n^{\log_b a}$ to determine the runtime behavior of the algorithm. Ideally,

- if $n^{\log_b a}$ when rounded down is still an upper boundary function of $f(n)$, then the recurrence has a $\Theta(n^{\log_b a})$ runtime.
- if $f(n)$ and $n^{\log_b a}$ have the same asymptotic boundary function, then the recurrence has a $\Theta(n^{\log_b a} \lg(n))$ runtime.
- if $n^{\log_b a}$ when rounded up is a lower boundary function of $f(n)$ and $f(n)$ does less work as the subdivisions get smaller for sufficiently large values of n, then the recurrence has a $\Theta(f(n)))$ runtime.

The master theorem is used to show that the functions $f(n)$ and $n^{\log_b(a)}$ are polynomially different from each other when they are not equal; hence, some recurrence that satisfies the initial criteria can fall between cases, thus making the master theorem unable to determine their runtimes.

If $f(n) = O(n^k)$ with constant $k \geq 0$, the simplified version of the Master Theorem can be applied:

> Let $a \geq 1$, $b > 1$, and $k \geq 0$ be constants and let $T(n)$ be defined on the nonnegative integers by the recurrence
>
> $$T(n) = aT\left(\frac{n}{b}\right) + O(n^k)$$
>
> then
>
> $$T(n) = \begin{cases} \Theta(n^{\log_b a}) & \text{if } a > b^k \\ \Theta(n^k \lg(n)) & \text{if } a = b^k \\ \Theta(n^k) & \text{if } a < b^k \end{cases}$$