Master Theorem

1. Base Case : T(n) <= a constant for all sufficiently small n
2. For all larger n :

$$T(n) \leq aT(n/b) + O(n^d)$$

where

a = number of recursive calls (>= 1)
b = input size shrinkage factor (> 1)
d = exponent in running time of "combine step" (>=0)
[a,b,d independent of n]

Base doesn't matter (only changes leading constants)

$$T(n) = \begin{cases} O(n^d \log n) & \text{if } a = b^d \quad \text{(Case 1)} \\ O(n^d) & \text{if } a < b^d \quad \text{(Case 2)} \\ O(n^{\log_b a}) & \text{if } a > b^d \quad \text{(Case 3)} \end{cases}$$

Base matters

Divide and conquer examples:

$T(n) = 2T(\frac{n}{2}) + O(n) = O(n \log n)$ — MergeSort()

$T(n) = 4T(\frac{n}{2}) + O(n) = O(n^2)$ — Naive D&C Integer Multiplication

$T(n) = 3T(\frac{n}{2}) + O(n) = O(n^{\log_2 3})$ — Improved D&C Integer Multiplication

$T(n) = T(\frac{3}{4}n) + O(n) = O(n)$ — Median finding