Master Theorem

$$\boxed{T(n) = a\,T\left(\frac{n}{b}\right) + O(n^d)}$$

$a \geq 1 \rightarrow$ no. of subproblems.

$n/b \rightarrow$ size of each subproblem, $(b > 1)$

$O(n^d) \rightarrow$ cost outside recursive calls (e.g. partitioning/merging)

$\boxed{C-1}$ $\log_b^a > d$

Recursive work dominate

$$\boxed{T(n) = O(n^{\log_b^a})}$$

$\boxed{C-2}$ $\log_b^a = d$

Recursive & non-recursive are balanced.

$$\boxed{T(n) = O(n^d \log n)}$$

$\boxed{C-3}$ $\log_b^a < d$

Non-recursive dominat

$$\boxed{T(n) = n^d}$$

Eg-1 $T(n) = 5\,T(n/2) + O(n)$

$a = 5,\ b = 2,\ d = 1$ $\quad \log_2^5 > 1$ $[\log_b^a > d]$ $\boxed{\text{Case-1}}$

$$\boxed{T(n) = O(n^{\log_2^5})} = \boxed{O(n^{2.32})}$$

Algo-2 $T(n) = 2T(n-1) + O(1)$ [Not suitable for master theorem] $\frac{n}{x} = n-1$

$T(n-1) = 2T(n-2) + O(1)$ Solve directly $x = \frac{n}{n-1}$

$b = \frac{n}{n-1}$

$T(n) = 2^2 T(n-2) + O(1)$

$T(n) = 2^3 T(n-3) + O(1)$

$T(n) = 2^n T(0) + O(1) \Rightarrow \boxed{T(n) = O(2^n)}$

Algo-3) $T(n) = 9T(\frac{n}{3}) + O(n^2)$

$a = 9,\ b = 3,\ d = 2$

$\log_3 9 = 2$ [case-2 $\log_a b == d$]

$\boxed{T(n) = O(n^2 \log n)}$

A-1 $\Rightarrow O(n^{2.32})$ (polynomial T.C.)

A-2 $\Rightarrow O(2^n)$ (exponential, grows very fast, worst Algorithm)

A-3 $\Rightarrow O(n^2 \log n)$ (better than 1 & significantly better than 2)

As $(n^2 \log n)$ grows slowly than $n^{2.32}$

$\therefore$ A-3 > A-1 >> A-2

$\boxed{O(n^2 \log n) > O(n^{2.32}) > O(2^n)}$

$T(n) = aT(\frac{n}{b}) + O(n^d)$

log

C-1 $\log_b a > d$	C-2 $\log_b a = d$	C-3 $\log_b a < d$
$O(n^{\log_b a})$	$O(n^d \log n)$	$O(n^d)$