$T(n) = aT(n/b) + n^d$ $d>0$ ← $T(n) = O(n^d)$ $a < b^d$

$= O(n^d \log_b n)$ $a = b^d$

$= O(n \log_b a)$ $a > b^d$.

* **AVL Trees :** $BF = (-1, 0, 1)$

Insertion : LL, LR, RR, RL

Deletion :

L type (If node deleted is left node)	R type (Right node)
L0 - RR	R0 - LL
L1 - RL	R1 - LL
L-1 - RR.	R(-1) - LR.

Complexity : $h = O(\log_2 n)$; $h \leq 2\log_2 n$.

$N(n) > 2^i \times N(h-2i)$

fibonacci rm :

$n(h) = n(h-1) + n(h-2) + 1$

$n(h) \geq (1.62)^h$ or 2^h

min no of nodes with height h

rotation : $O(1)$

h(1 node) = $O(1)$

* **Splay trees :**

zig	- L'
zig	- R
zig-zig	- LL
zig-zag	- LR
zag-zag	- RR
zag-zig	- RL

Amortized :

multipop(S, k) :
total cost = min(S, k)

worst case of multipop : $O(n)$

" " sequence : $O(n^2)$

total cost = $O(n)$

ceiling → fraction to greater next number.

⇒ Amortized analysis, binary counter
↓ refer note

* M-Way trees → ensure balanced
complexity of any operation is O(h)
min no: of elements : h
max :
at height h : $m^{h-1}(m-1)$ keys

Total no: of elements = $m^h - 1$

Best case: worst case:

$n_{min} = \log_m^{n+1}$ $h = O(n)$ → max.

h varies from n to max of m^{h-1}

↳ $n_{min} = h$
$n_{max} = m^{h-1}$

* B-trees : can be empty
root - atleast 2 child, most m
internal nodes must have atleast $\lceil \frac{m}{2} \rceil$

Keys = 1 less than children.

complexity : O(h)
upperbound — $n \leq m^{h-1}$
lowerbound → $n : 2\lceil \frac{m}{2} \rceil^{h-2}$
$h+1 : 2\lceil \frac{m}{2} \rceil^{h-1}$

no: of elements — $n \geq 2\lceil \frac{m}{2} \rceil^{h-1} - 1$

$\log_m(n+1) \leq h \leq \log\left(\frac{n+1}{\lceil \frac{m}{2} \rceil}\right) + 1$

* B+ trees – $O(\log n)$ for every operation.

* Binary search : ~~$T(n) = T(n/2) + c$~~
~~$T(n) = O(\log n)$~~

* Merge- sort :

Avg, best, worst case $= O(n \log_2 n)$

c-merge(n) $= n - 1$

$c(n) = 2c(n/2) + c\text{-merge}(n)$ $\forall n > 1$

* Quick sort :

BC : c-best(n) $= \boxed{O(n \log_2 n)}$

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

WC : $T(n) = T(n-1) + n$

c-worst(n) $= \boxed{O(n^2)}$

* Binary search :

c-worst(n) $=$ c-worst$\left(\left\lfloor \frac{n}{2} \right\rfloor + 1\right)$ $\forall n > 1$

using masters, c-worst(n) $= O(\log_2 n)$

* Multiplication of large integers :

conventional method requires n^2 mult.

$\downarrow$

$\therefore$ use div & conquer.

$c = a * b = c_2 10^2 + c_1 10^1 + c_0 \ldots$

$c_2 = a_1 * b_1$ $\quad c_1 = (a_1 + a_2) * (b_1 + b_0) - (c_2 + c_0)$

$c_0 = a_0 * b_0$

$M(n) = 3M(n/2) \quad \forall n > 1$

$M(1) = 1$

using master's; $O(n^{\log_2 3}) = O(n^{1.585})$

:

* **Strassen's :**

$m_1 = (a_{00} + a_{11}) * (b_{00} + b_{11})$

$m_2 = (a_{10} + a_{11}) * b_{00}$

$m_3 = a_{00} * (b_{01} - b_{11})$

$m_4 = a_{11} * (b_{10} - b_{00})$

$m_5 = (a_{00} + a_{01}) * b_{11}$

$m_6 = (a_{10} - a_{00}) * (b_{00} + b_{01})$

$m_7 = (a_{01} - a_{11}) * (b_{10} + b_{11})$

$$c = \begin{bmatrix} m_1 + m_4 - m_5 + m_7 & m_3 + m_5 \\ m_2 + m_4 & m_1 + m_3 - m_2 + m_6 \end{bmatrix}$$

$T(n) = 7\,T(n/2) + n^2$

$T(n) = O(N)^{2.8}$

* **Karat. suba :** $T(n) = 3T(n/2) + n$

$T(n) = O(n)^{1.59}$