

$T(n) = aT(n/b) \left\{ \begin{array}{l} d > 0 \leftarrow T(n) = O(n^d) \quad a < b^d \\ + n^d \end{array} \right.$
 $= O(n^d \log_b n) \quad a = b^d$
 $= O(n \log_b a) \quad a > b^d$

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

Insertion: LL, LR, RR, RL

Deletion :

L type
(If node deleted is
left node)

L0 - RR

L1 - RL

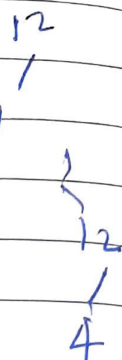
L-1 - RR

R type
(Right node)

R0 - LL

R1 - LL

R(-1) - LR



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

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

fibonacci num:

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

$$n(h) \approx (1.618)^h \text{ or } 2^h$$

min
no of
nodes with height h

$$\text{rotation} : O(1)$$

$$h(1 \text{ node}) = O(1)$$

* Splay trees :

Amortized :

zig - L'

zag - R

zig-zig - LL

zig-zag - LR

zag-zag - RR

zag-zig - RL

multipop(s, k)

total cost = min(s, k)

worst case of multipop : $O(n)$

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

total cost = $O(n)$

ceiling \rightarrow fraction to
greater
 \rightarrow next number.

\Rightarrow Amortized analysis, binary counter
 \downarrow
reference

* M-way trees \rightarrow ensure b/w min

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:

$$h_{\min} = \log_m (n+1)$$

worst case:

$$h_{\max} = O(n)$$

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

$$\rightarrow \begin{cases} n_{\min} = h \\ n_{\max} = m^{h-1} \end{cases}$$

* B-trees: can be empty

root - least 2 child, most m

internal nodes must have atleast $\lceil \frac{m}{2} \rceil$

Keys = 1 less than children.

complexity: $O(h)$

upperbound - $n \leq m^{h-1}$

$$\text{lowerbound} \rightarrow \begin{cases} n \geq 2 \left\lceil \frac{m}{2} \right\rceil^{h-2} \\ n+1 \geq 2 \left\lceil \frac{m}{2} \right\rceil^{h-1} \end{cases}$$

$$\text{no. of elements} - n \geq 2 \left\lceil \frac{m}{2} \right\rceil^{h-1} - 1$$

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

* 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 = $O(n \log_2 n)$

$c_merge(n) = n - 1$

$c(n) = 2c(n/2) + c_merge(n) \quad \forall n > 1$

* 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 :

$c_worst(n) = c_worst(\lfloor \frac{n}{2} \rfloor + 1) \quad \forall n > 1$

using masters $c_worst(n) = O(\log_2 n)$

* Multiplication of large integers :

conventional method requires n^2 multi.

↓
∴ use div + congruence

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

$$c_2 = a_1 * b_1$$

$$c_0 = a_0 * b_0$$

$$c_1 = (a_1 + a_0) + (b_1 + b_0)$$

$$- (c_2 + c_0)$$

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

$$M(1) = 1$$

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

* Strassen's:

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

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

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

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

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

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

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

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

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

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

* Karatsuba: $T(n) = 3T(n/2) + n$

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