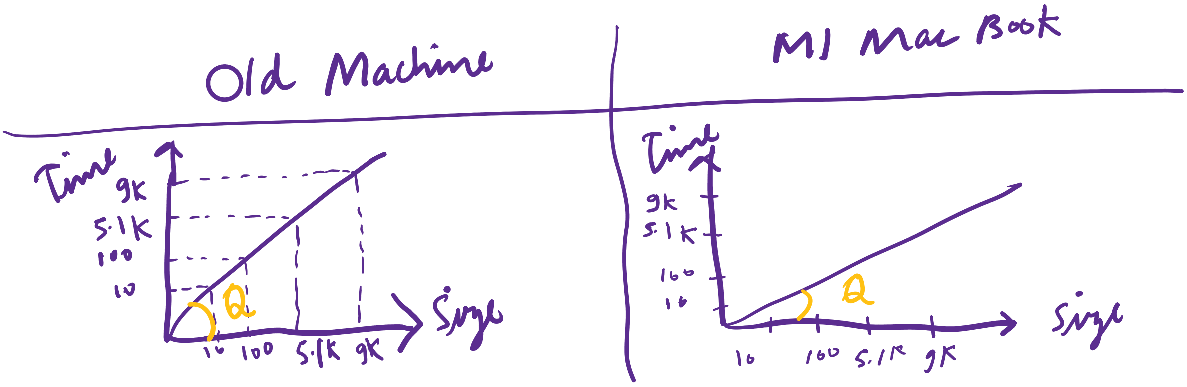
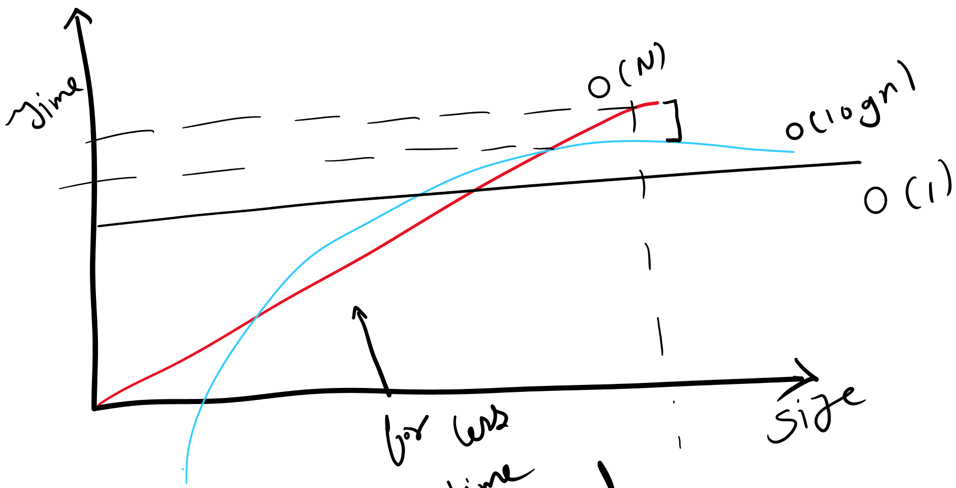
Time complexity != Time Taken

* Function that gives us the relationship about how the time will grow as the input grows.





for less time linear is good

$O(1) < O(\log N) < O(N) \leftarrow$ for large

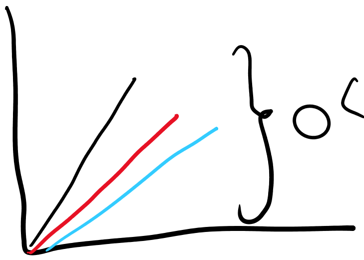
$O(1) > O(\log N) > O(N) \leftarrow$ for small

what do we consider when thinking about complexity:-

1. Always look for worst case complexity.
2. Always look at complexity for large/∞ data.

data.

3



$O(N)$

$y = x$
$y = 2x$
$y = 4x$

* Even tho values of actual time is diffrent they are all growing linerly.

* we dont care about actual time. This is why, we ignore all constrants.

4 $O(N^3 + \log N)$

* from Point no. 2
$1\ mil = ((1\ mil))^3 + \log(mil))$
$= (1\ mil^3 + 6\ sec)$

very small hence ignore

Always ignore less dominating term.

Big-OH Notation :

* Word definition :-
$O(N^3) \rightarrow$ Upper bound

* Maths :-

$$f(N) = O(g(N))$$

$$\boxed{\lim_{N \to \infty} \frac{f(N)}{g(N)} < \infty}$$

$$\underset{g(N)}{O(N^3)} = O(\underset{f(N)}{6N^3 + 3N + 5})$$

$$\lim_{N \to \infty} \frac{6N^3 + 3N + 5}{N^3}$$

$$= \lim_{N \to \infty} 6 + \frac{3}{N^2} + \frac{5}{N^3} = 6 + \frac{3}{\infty} + \frac{5}{\infty}$$

$$= 6 + 0 + 0$$

$$= 6 < \infty$$

finite value

Big Omega :- Opposite of Big OH

<u>words</u> $\Omega(N^3)$ (lower bound)

<u>Maths</u> :-

$$\boxed{\lim_{n \to \infty} \frac{f(n)}{g(n)} > 0}$$

Q1. What if an algo has lower bound & upper bound as (N^2).

$= O(N^2)$ & $\Omega(N^2)$

Theta Notation (Combining Both)

$\Theta(N^2) \ni$ words :→ Both upper bond & lower bond

$$\boxed{0 < \lim_{n\to\infty} \frac{f(N)}{g(N)} < \infty}$$

Little O notation :→

* This is also giving upper bond.

words : loose upper bond

Big Oh	little O
$f = O(g)$	$f = o(g)$
$f \le g$	$f < g$ strictly lower

Maths :→

$$\boxed{\lim_{n\to\infty} \frac{f(N)}{g(N)} = 0}$$

Ex. $f = N^2$ $\qquad g = N^3$

$$\lim \frac{N^2}{N^3} = \lim_{N\to\infty} \frac{1}{N} = 0$$

$$\lim_{N \to \infty} \frac{N^2}{N^3} = \lim_{N \to \infty} \frac{1}{N} = 0$$

* little Omega :-

Big Ω	little ω
$f = \Omega(g)$	$f = \omega(g)$
$f \geq g$	$f > g$

Maths :-

$$\boxed{\lim_{N \to \infty} \frac{f(N)}{g(N)} = \infty}$$

Space Complexity or Auxiliary Space?

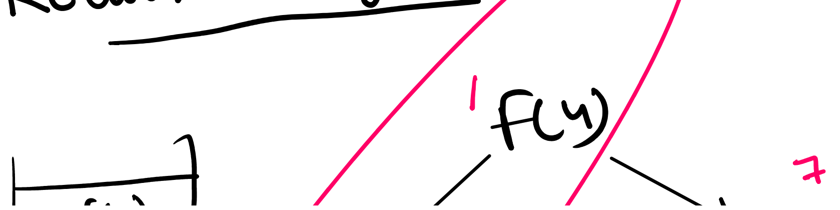
Auxiliary Space is the extra space or temporary space used by an algorithm.

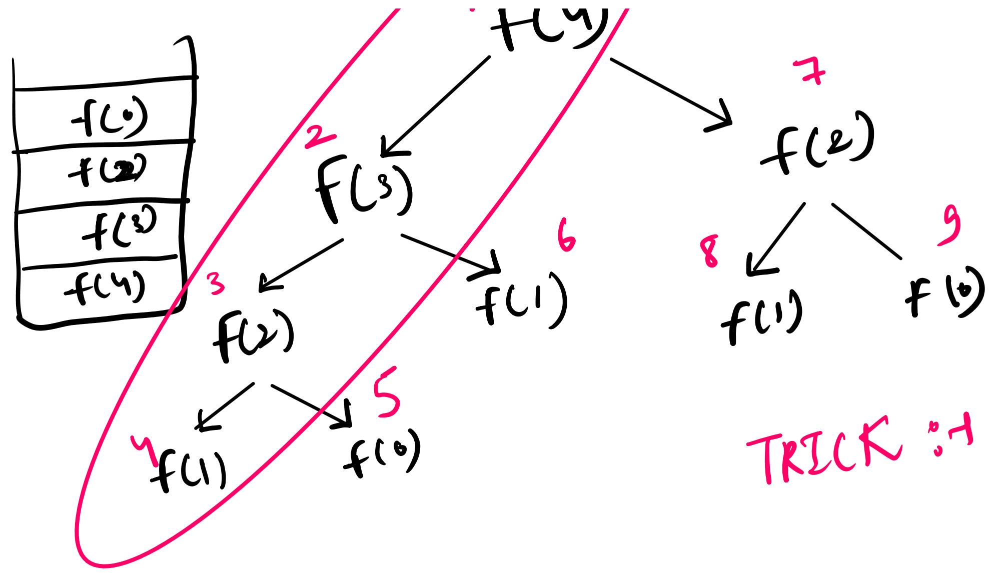
Space Complexity of an algorithm is total space taken by the algorithm with respect to the input size. Space complexity includes both Auxiliary space and space used by input.

For example, if we want to compare standard sorting algorithms on the basis of space, then Auxiliary Space would be a better criteria than Space Complexity. Merge Sort uses O(n) auxiliary space, Insertion sort and Heap Sort use O(1) auxiliary space. Space complexity of all these sorting algorithms is O(n) though.

Recursive Algorithm :→

TC → O(N)





TRICK :-

Only call that are interlinked with each other will be in the stack at some time

2 types of recursions :-

① Linear

$$F(N) = f(N-1) + f(N-2)$$

② Divide & Conquer

$$f(N) = f\left(\frac{N}{2}\right) + O(1)$$

* Divide & Conquer Recurrence :-

Form :-

$$T(x) = a_1 T(b_1 x + \varepsilon_1(n)) + a_2 T(b_2 x + \varepsilon_2(n)) + \dots + a_k T(b_k x + \varepsilon_k(n))$$

$+ \cdots + a_k T(b_k x + \varepsilon_k(x))$

$+ g(x)$

for $x \geq x_0$

same unit

$$\underbrace{T(N) = T\left(\frac{N}{2}\right)}_{a_1 = 1} + C$$

$g(x) = C$

$a_1 = 1$

$b_1 = 1/2$

$\varepsilon_1(x) = 0$

$$T(N) = 9\,T(N/3) + \frac{4}{3}\,T\left(\frac{5}{6}N\right) + 4N^3$$

$a_1 \quad b_1 \qquad a_2 \quad b_2 \qquad g(N)$

$$T(N) = 2T\left(\frac{N}{2}\right) + \frac{(N-1)}{g(N)}$$

$a_1 \quad b_2$

when you get ans from this + what you are doing with that answer take how much time.

* How to actually solve to get complexity :)

① Plug & chug

$$F(N) = F(N/2) + C$$

$$F(N) = F(N/2) + C$$

② master's Theorem

③ Akra Bazzi (1996)

Akra Bazzi :-

$$\boxed{T(x) = \Theta\left(x^p + x^p \int_1^x \frac{g(u)}{u^{p+1}}\, du\right)}$$

What is P?

$$a_1 b_1^p + a_2 b_2^p + \dots\dots = 1$$

$$\boxed{\sum_{i=1}^{K} a_i b_i^p = 1}$$

Ex: $T(N) = 2T\left(\frac{N}{2}\right) + (N-1)$

$a_1 = 2$

$g(x) = n-1$

$b_1 = 1/2$

$a_1 = 2$

$b_1 = 1/2$

$$2 \times \left(\frac{1}{2}\right)^p = 1$$

$$\boxed{p = 1}$$

Put p in formula :-

$$T(n) = \Theta\left(x^1 + x^1 \int_1^x \frac{u-1}{u^2}\, du\right)$$

$$= \Theta\left(x + x \int_1^x \frac{1}{u} - \frac{1}{u^2}\, du\right)$$

$$= \Theta\left(x + x\left[\int_1^x \frac{du}{u} - \int_1^x \frac{du}{u^2}\right]\right)$$

$$= \Theta\left(x + x\left[\left[\log u + \frac{1}{u}\right]\right]\right)$$

$$= \Theta\left(x + x\left(\log x + \frac{1}{x} - 1\right)\right)$$

$$= \Theta(x + x\log x + 1 - x)$$

$$= \Theta(x\log x + 1)$$

$$= \Theta(\ldots x) \quad // \text{Time} \ldots$$

$$= \theta(x \log x)$$ // Time complexity