Time & space Complexity.

Most Important

* What is time complexity?

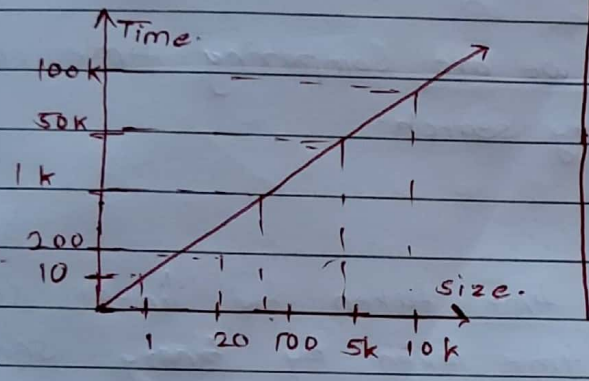
① Time complexity is not about the time taken to run the program.

* Example:-

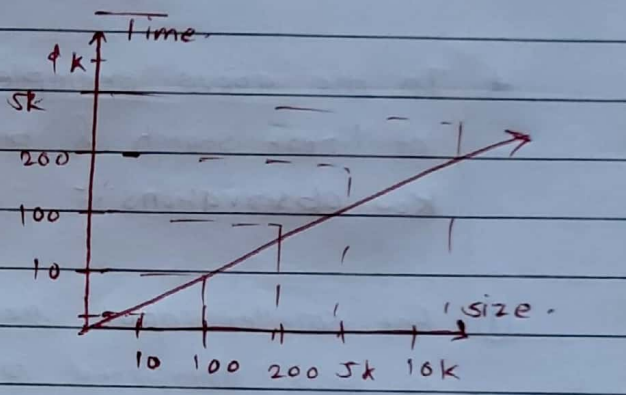
old machine.	New Macbook pro.

① Array of 1,00,000 elements in both machines, we have to search for an element in array with linear search algo.

* it takes 10 sec.



* it takes 1 sec.

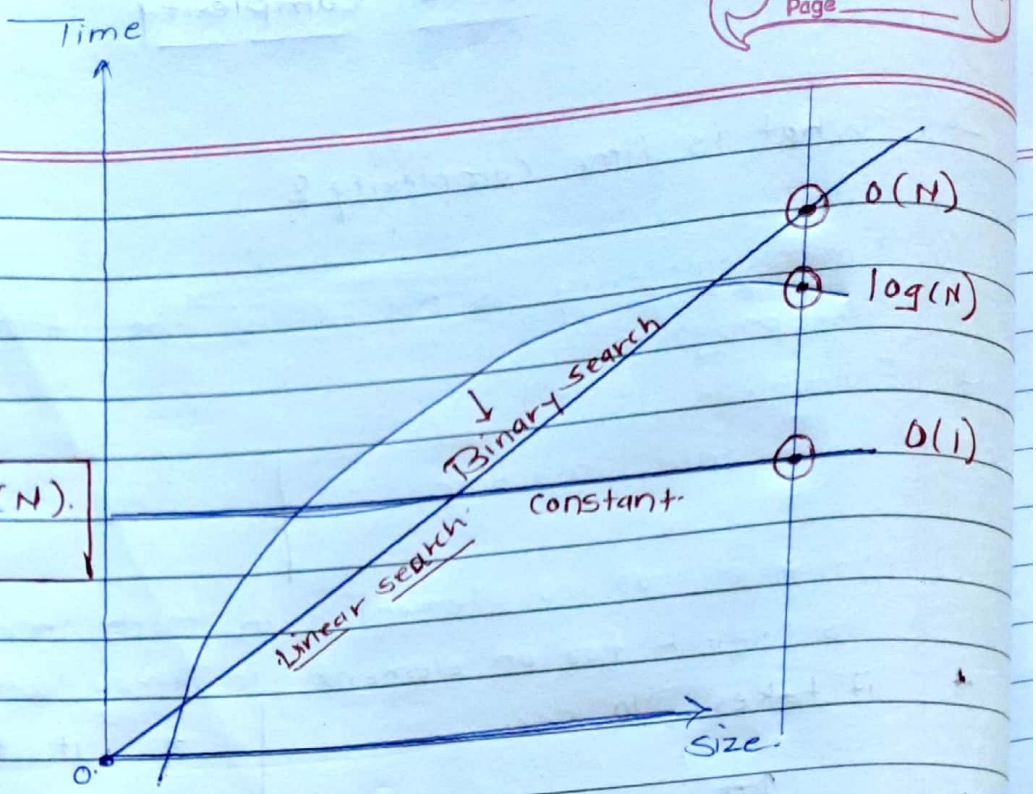


* Both graphs are same.

* Imp:- Function that gives us the relationship about how the time will grow on input grows. This is time complexity.

> Time Complexity != Time taken to run algo.

* Why?



$O(1) < O(\log N) < O(N)$.

* In the above example we are Comparing time complexity of linear search & binary search.

Key observations:-

① For smaller input sizes time, taken by linear search is less Compared to time taken by binary search.

② For larger input time taken by linear search is more as Compared to time taken by binary search, And it keeps on increasing as Compared to time taken by linear search.

③ We only Consider the case with having large input sizes i.e. worst case because it allows us to be know whether the algorithm is efficient in the long term or not. Worst case ~~anyti~~ analysis helps us to determine worst wor scenario.

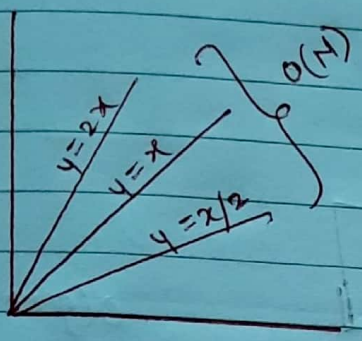
④ Thus we can say that binary search is mor efficient as Compare to linear search.

classmate
Date
Page

* Key points to consider when dealing with time complexity:-

① Always look for worst case scenario.

② Always take large input sizes / infinity into consideration.

③ 

Here, the actual time is different but in all cases the time is growing linearly as the input grows.

* Dont care about actual time.
* Ignore constants.

④ Ignore less dominating terms.

Example:- $O(N^3 + \log(N))$

here, $N = 1 \times 10^6 = 1$ Million.

$O((10^6)^3 + \log(10^6))$ $\quad \log(10^6) = 6$.

$= O((10^6)^3 + 6)$ ⇒ here, 6 is very small as

$= O((10^6)^3)$ Compared to 1 million cube

$\approx O(N^3)$ thats why we ignored it.

Big O Notation. (Big-oh)

① Big O means simple Upper bound (limit).

"A mathematical notation which describes the upper limit of the function depicting the relationship between the time & input size."

* For example :- $O(N^3)$
The time complexity cannot exceed $O(N^3)$. It ca
be $O(N^2)$, $O(N)$, $O(\log N)$, but it can't be
greater than $O(N^3)$

* Mathematics behind it :-

If $f(n) = O(g(n))$
then.

$$\lim_{x \to \infty} \frac{f(n)}{g(n)} < \infty$$

$O(g(n))$

$f(n)$
$\longrightarrow 6(n^3) + 3n + 5 = O(n)^3$

$$= \lim_{n \to \infty} \frac{6(n^3) + 3n + 5}{n^3}$$

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

putting $n = \infty$

$$= 6 + \frac{3}{\infty} + \frac{5}{\infty} \Rightarrow 6 + 0 + 0 = \boxed{6}$$

$$= 6$$

$$< \infty$$

And, thats why we ignore less dominating term & Constants !

* **Big Omega (Ω):** (Opposite of Big oh.)

In words: $\Omega(N^3)$ => lower bound.

A mathematical Expression which describes "lower bound." of the size & input & time.

* **Mathematics:-**

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

$O(n^3)$

* It means that time complexity ~~exceeds~~ can not be below of $O(n^3)$. at least $O(n^3)$ be time complexity.

* **Theta Notation:- (Θ)** Combining Both.

Θ means time complexity lies between upper bound & lower bound of the expression.

* **Mathematics:-**

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

* **Little o Notation:-** This is also giving upper bound.

* In words:- lose up. bound.

Big oh	theta Θ
$y = O(g)$	$y = \Theta(g)$
$y \leq g$	$y < g$ strictly slower

Maths:- $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$

Ex.

$y = N^2$ $\quad g = N^3$

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

* little omega:-

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

Maths: $\lim_{N \to \infty} \frac{g(N)}{g(N)} = \infty$

Ex. $\lim_{n \to \infty} \frac{N^3}{N^2} = \lim_{N \to \infty} N = \infty$

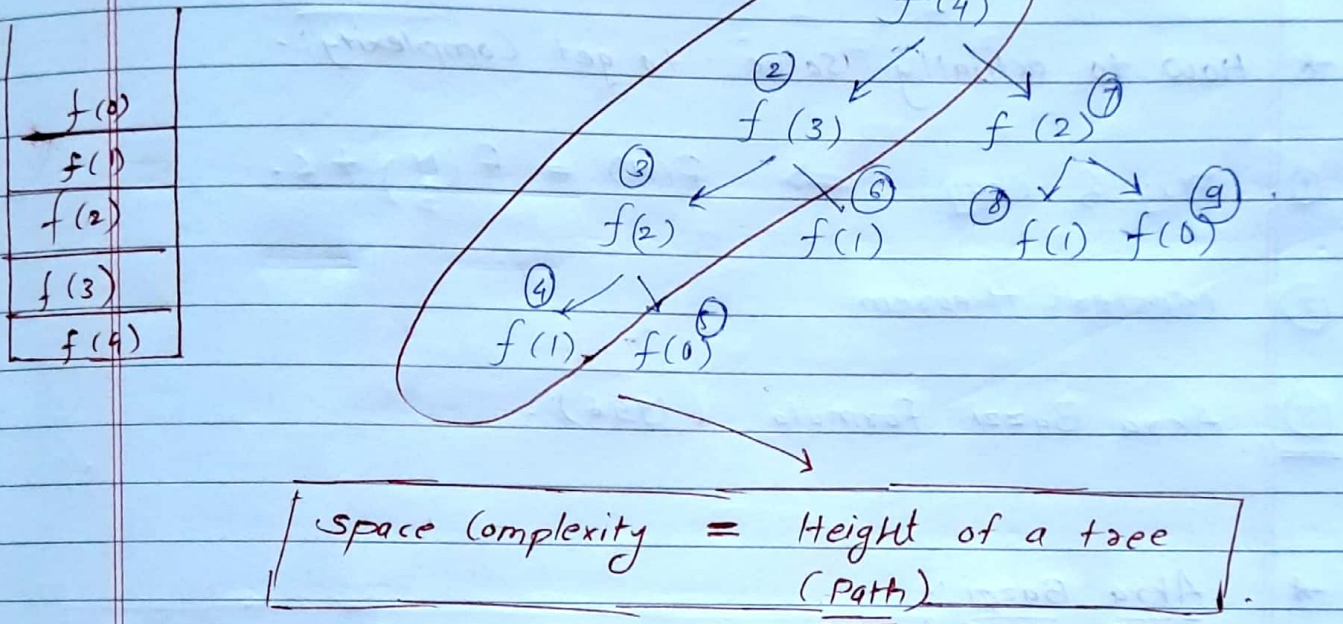
Space Complexity or Auxiliary space ?

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

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

③ For example, if we want to compare standard sorting algorithm on the basis of space, the Auxiliary space would be better criteria than space Complexity. Merge sort uses O(n) auxiliary space, insertion sort & heap sort use O(1) auxiliary space. space complexity of all algorithm is O(n) though.

* Recursive Algorithm:-



Space Complexity = Height of a tree (Path).

* 2 Types of Recursion:-

① Linear

→ $F(N) = F(N-1) + F(N-2)$

② Divide & Conquer.

$F(N) = F(\frac{N}{2}) + O(1)$.

① Divide & Conquer Recurrence:-

* Form:- $T(x) = a_1 T(b_1 x + \varepsilon_1(n)) + a_2 T(b_2 x + \varepsilon_2(n) + \cdots + a_k T(b_k x + \varepsilon_k(n)) + g(n)$

for $x \geq x_0$

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

$9 \to a_1$, $\frac{N}{3} \to b_1$, $\frac{4}{3} \to a_2$, $\frac{5N}{6} \to b_2$, $4N^3 \to g(n)$

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

$2 \to a_1$, $\frac{N}{2} \to b_1$, $N-1 \to g(n)$

* How to actually solve to get Complexity:-

① Plug & chug $\Rightarrow$ $F(N) = F\left(\frac{N}{2}\right) + C.$

② Master's theorem

③ Akra Bazzi formula (1996).

imp.

* **Akra Bazzi:-**

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

* What is P?

$\rightarrow$ $a_1 b_1^P + a_2 b_2^P + \cdots + a_n b_n^P = 1.$

i.e. $\sum_{i=1}^{k} a_i b_i^P = 1.$

$u^{-2} = \frac{u^{-2+1}}{-2+1}$

$= \frac{u^{-1}}{-1}$

classmate Date Page

* Example:-

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

$$g(n) = n-1$$

From this, $a_1 = 2$, $b_1 = \frac{1}{2}$.

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

i.e. $\boxed{P=1}$.

* Put P in formula:-

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

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

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

$$= \Theta\left(x + x\left[\left[\log u + \frac{1}{u}\right]_1^x\right]\right)$$

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

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

$$= \Theta\left(x \log x + 1\right)$$ ignore constant.

$\boxed{T(N) = \Theta(x \log x.}$ // Time Complexity.

For array of size N Tim Complexity = $\underline{O(N \log N)}$.

Q. $T(N) = 2T\left(\frac{N}{2}\right) + \frac{8}{9}T\left(\frac{3N}{4}\right) + N^2$.

a_1 b_1 a_2 b_2

$\Rightarrow$

$$2 \times \left(\frac{1}{2}\right)^P + \frac{8}{9} \times \left(\frac{3}{4}\right)^P = 1.$$

$$2 \times \frac{1}{4_2} + \frac{8}{9} \times \frac{9}{16_2} = 1.$$

$$\frac{1}{2} + \frac{1}{2} = 1 \qquad \therefore \boxed{P = 2}.$$

* Put P in Formula:-

$$T(N) = \Theta\left(x^P + x^P \int \frac{g(n)}{u^{P+1}}\, du\right)$$

$$= \Theta\left(x^2 + x^2 \int \frac{u^2}{u^3}\, du\right)$$

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

ignore the constant.

$$= \Theta\left(x^2 + x^2 \log u\right)$$

$$\underline{T(N) = \Theta(x^2 \log u)}$$

* Note:- When you don't get value of P.

then if P < Power of (g(n))

then ans = g(n).