Space & Time Complexity

① What is time Complexity ?

- Time Complexity != Time taken

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

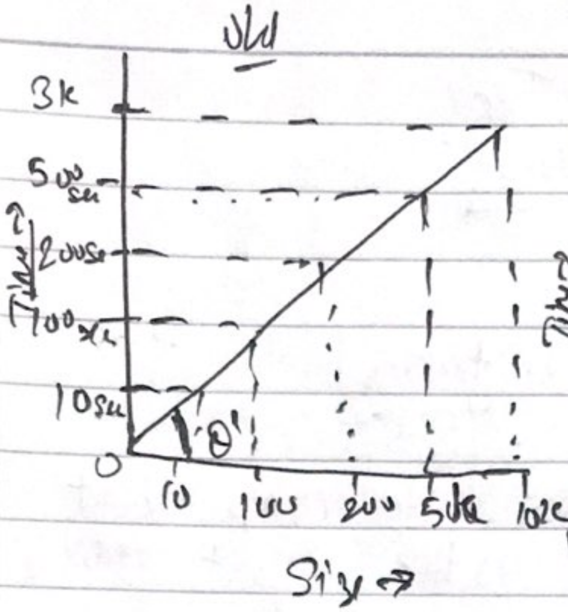
or

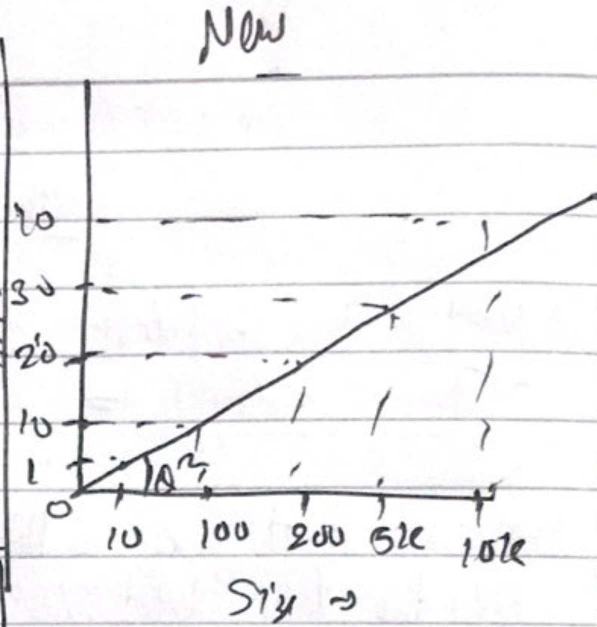
As the input grows time grows is known as time Complexity

example =

Old Computer	New Computer (very fast)
1 million element array	1 million element array
Linear Search for target & that does not exist	—11—
time taken = 10 sec	1 sec

⇒ Both machine have Same time Complexity





Straight line

Steeper straight line

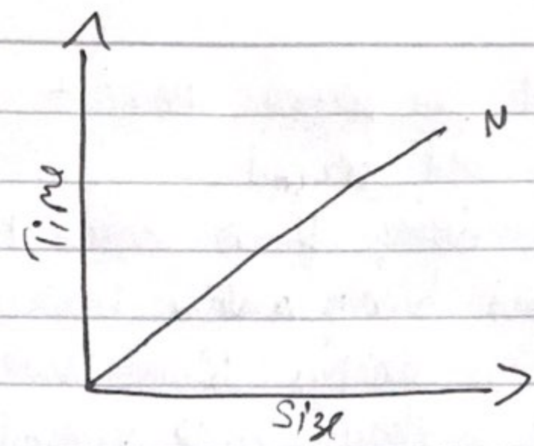
time taken is diff. but relation ship between the size and the time is linear

in above example time is growing linearly as the size is growing

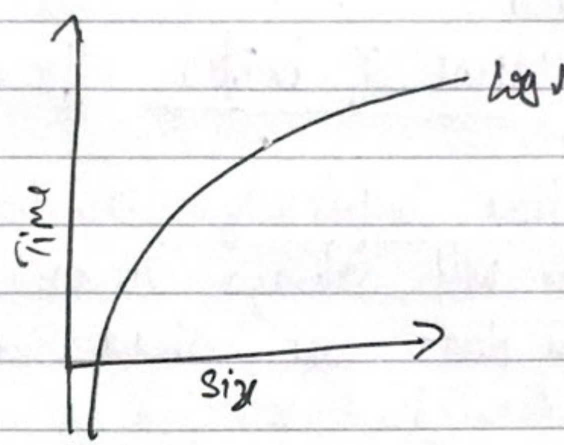
Why?

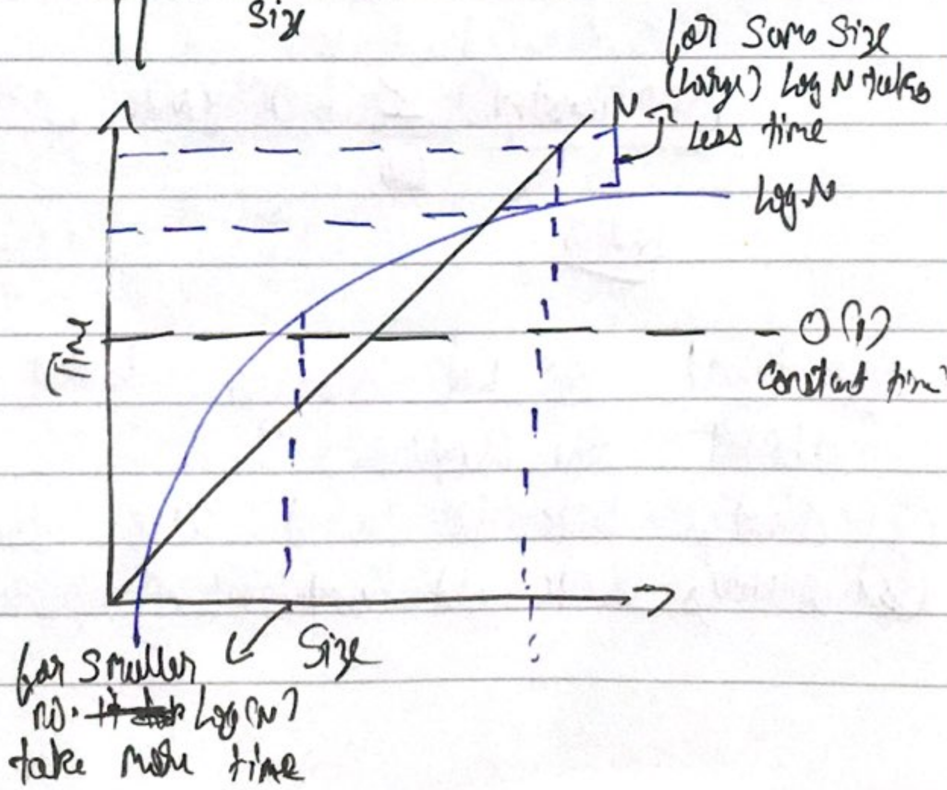
linear search grows linearly (N)
Binary search grows with (Log N)

graph for Linear Search



graph for Binary search





- If Size is fixed for Larger no. it may go like above and beyond
- For Larger size always linear search takes more time than binary search, whereas
- For Smaller size always binary search (log N) takes more time than linear search (N)

Which one better?
- Log (N) because of better efficiency

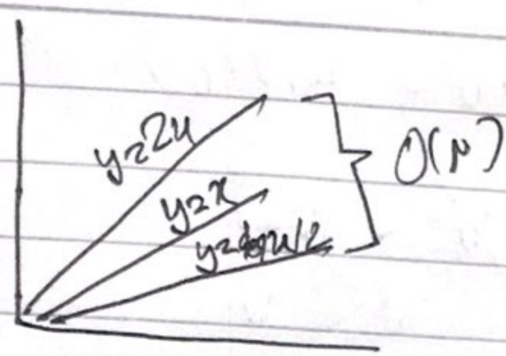
For Constant time Complexity, Size does not matter, time will always remain ~~same~~ Constant, "We don't care about Smaller no."

$$\underbrace{O(1)\ (\text{constant})}_{\text{better}} \leq O(\log N) < O(N)$$

Que What do we Consider when thinking about the Complexity?

1. always look for worst case Complexity.
2. Always look at Complexity / infinite data

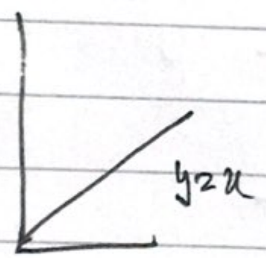
(3)

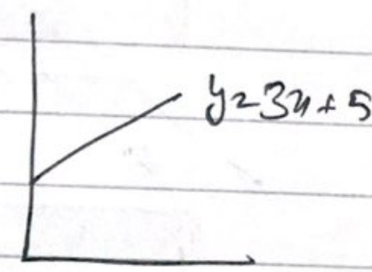


Here the actual time is different but in all cases the time is growing linearly as the input grows

- We don't care about actual time
- Ignore Constants

eg





relationship is same
constant does not matter

(4) Ignore Less dominating terms

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

for $N = 1 \times 10^6$

$O(10^{18} + \log 10^6)$

$O(10^{18} + 6)$

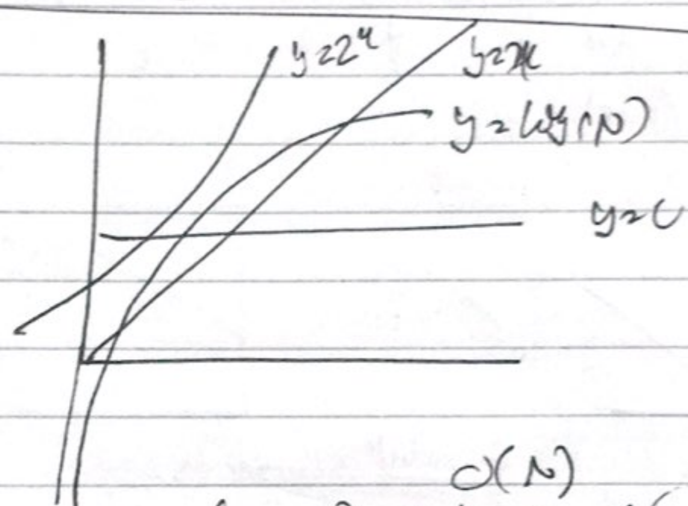
(less dominating) → ignore

$\underline{O(10^{18})}$

eg: $O(3N^3 + 4N^2 + 5N + 6)$

$\Rightarrow O(N^3 + \cancel{N^2} + \cancel{N})$

$= O(N^3)$



exponentially

$O(1) < O(\log(N)) < O(N) < O(N\log N) < O(2^n)$

Big Oh Notation

Word definition:-

$O(N^3) \rightarrow$ upper bound

$\Rightarrow$ The Complexity cannot exceed N^3

Maths

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

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

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

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

$$\lim_{n \to \infty} 6 + \frac{3}{N^2} + \frac{5}{N^3} = \underline{6 + 0 + 0}$$

$(6 < \infty)$

finite value

Big omega notation : (opposite of big Oh)

$\Omega(N^3)$ = Lower bound
$\Rightarrow$ it will take atleast N^3 Complexity

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

Ques What if an algo. as L.B & U.B as N^2

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

Theta Notation

$$\boxed{\Theta(N^2) \Rightarrow 0 < \lim_{n \to \infty} \frac{f(N)}{g(N)} < \infty}$$

Both upper bound & lower bound ($\Rightarrow N^2$) as same

Little o-Notation
- This also give upper bound
- Loose uper bound

Stronger Statement

Big oh	Little oh
$f = O(g)$	$f = o(g)$
$f \le g$	$f < g$
	Strictly slower than g

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

eg $f = N^2$ $\quad g = N^3$

$$\lim_{n \to \infty} \cancel{f(N)} \frac{N^2}{N^3} = 0$$

Little Omega ω

- gives lower bound, but gives loose lower bound

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

$f > g$ $\quad$ for Big om $f \geq g$