Time complexity

▷ Relation blw input size & running Time (operations)

1. What is time complexity?

⇓

It is a func^n that tells us, or gives relationship that how the time is going to grow as the size of input grows.

2. What do we consider when thinking about complexity:

1→ Always look for worst case complexity
2→ Always look at complexity for large n/∞ data.
3→ ~~Ignore~~ constant

Even though value of actual time is diff they are all growing linearly.

we don't care abt actual time, we care abt how the time is growing with growing input.

That's why we don't take care of constant in the calculation of Time complexity

4→ $O(N^3 + \log N)$

~~*~~from pt 2

⇒ 1 mil ∋ $((1\text{mil})^3 + \log(1\text{mil}))$

⇒ $((1\text{mil})^3 + 6\text{sec})$ — very small hence ignore

⇒ $(1\text{mil})^3$

Always ignore less dominating terms.

3. Big O notation

☆ wordy definition

⇓

$O(N^3)$ ⇒ upper bound.

⇓

Mtlb time will not exceed N^3. It will be either N^3 or less than N^3

☆ Mathematical pow.

$f(n) = O(g(n))$

4. Big omega notation → lower bound.

⇓

opposite of big O.

⇓

$\Omega(N^3)$ ⇒ at least of N^3 time.

Q what if an algorithm has $U_b < U_p$ as N^2

5. Little O notation

⇓

This is also giving the upper bound

⇓

loose upper bound.

Big O	Vs	little O.
$f \leq g$		$f < g$ strictly slower.

5. Little omega

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

Space complexity (Auxiliary space)

→ Auxiliary space: It is the extra space or temporary space used/taken by an algorithm.

→ Space complexity: In simple terms it is the equal to input space plus the auxiliary space.

↓

It is total space taken by the algorithm with respect to the input size. It includes both Auxiliary space & space used by input.

Q. Find time complexity

```
for (i=1; i<=n; ) {
    for (j=1; j<=k; j++) {
        // some operation that
        // takes time t
    }
    i = i+k;
}
```

⇒ ~~Outer loop = N~~

Inner loop = O(k t)

~~Outer loop = O(kt * time~~

for outer loop ⇒ i = 1, i = 1 + k, i = 1 + 2k, ... i = 1 + xk

$1 + xk = n$

$xk = n - 1$

$x = \frac{n-1}{k}$

Ans ⇒ O(kt x outer loop ka time)

$= O\left(kt\left(\frac{n-1}{k}\right)\right)$

$O(t(n-1)) \Rightarrow O(nt) = O(n)$