* Time & Space complexity.

* Time complexity:

→ It is the amount of time taken by an algorithm to run.

→ as a function of length of the input.

Why? → for making better program
↳ comparison of Algo

→ Big O notation → Theta Θ → Omega Ω

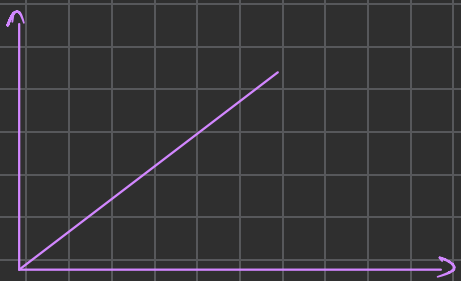
- Big O notation → Upper Bound → at max how much time (worst case)
- Theta Θ → for avg case complexity → on Avg how much time
- Omega Ω → lower Bound → At least how much time.

→ Constant time → $O(1)$
→ linear time → $O(n)$
→ logarithmic time → $O(\log n)$
→ Quadratic time → $O(n^2)$
→ Cubic time → $O(n^3)$

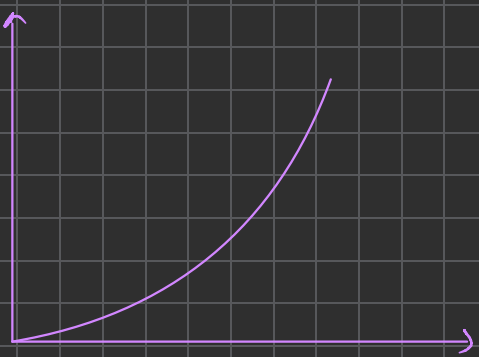
Graphs:

→ O(1)

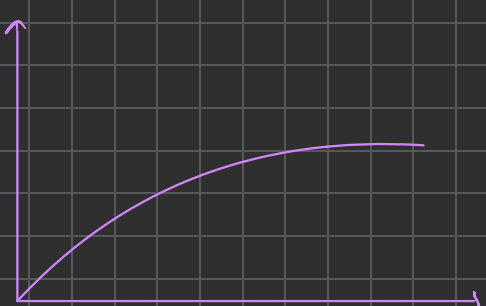
→ O(n)



→ O(n²)



→ O(log n)



O(n!)
O(2ⁿ)
O(n³)
O(n²)
O(n log n)
O(n)
O(log n)
O(1)

Complexity ↑

Questions. Ans.

$f(n) \rightarrow 2n^2+3n \rightarrow O(n^2)$

$f(n) \rightarrow 4n^4+3n^3 \rightarrow O(n^4)$

$f(n) \rightarrow N^2+\log N \rightarrow O(N^2)$

$f(n) \rightarrow 12001 \rightarrow O(1)$

$f(n) \rightarrow 3n^3+2n^2+5 \rightarrow O(n^3)$

$f(n) \rightarrow \frac{n^3}{300} \rightarrow O(n^3)$

$f(n) \rightarrow 5n^2+\log n \rightarrow O(n^2)$

$f(n) \rightarrow n/4 \rightarrow O(n)$

Q. for(0 → n){ ... } $O(n)$ → print Array

Q// for(0 → n/2){ ... } $O(n/2) = O(n)$ → Swap

Q// for(0 → n){ ... } $O(n)$ → linear Search.

Q. time complexity?

```
int a=0, b=0;
for(int i=0; i<N; i++){
    a = a + rand();
}
for(int j=0; j<M; j++){
    b = b + rand();
}
```

O(n) + O(m) = O(n+m)

Q.
```
for(0 → N){        ⇒ O(n)
    for(0 → N){
        :
    }
}
```
O(n) × O(n) = O(n²)

Q
```
for(0 → N){        → O(n)
    for(N → i){    → worst case N → 0
        :
    }
}
```
O(n) × O(n) ⇒ O(n²)

→ we will see recursive time complexities in future video

→ Stack is TLE

10^8 operation rule → most of the modern machine can perform 10^8 operation/seconds.

→ Time Complexity.

$\leq [10 \ldots 11]$	$O(n!)$, $O(n^6)$
$< [15 \ldots 18]$	$O(2^n \times n^2)$
< 100	$O(n^4)$
< 400	$O(n^3)$
< 2000	$O(n^2 + \log n)$
$< 10^4$	$O(n^2)$
$< 10^6$	$O(n \log n)$
$< 10^8$	$O(n)$, $O(\log n)$

* Space Complexity:

→ if there is fixed size then the Space complexity will be constant.

```
→ for(0-n){
      vector<int> v(n)
      for(0-n){

      }
  }
```

here in every iteration the vector will be created of size n. so space complexity = O(n)