Time Complexity

Time Complexity

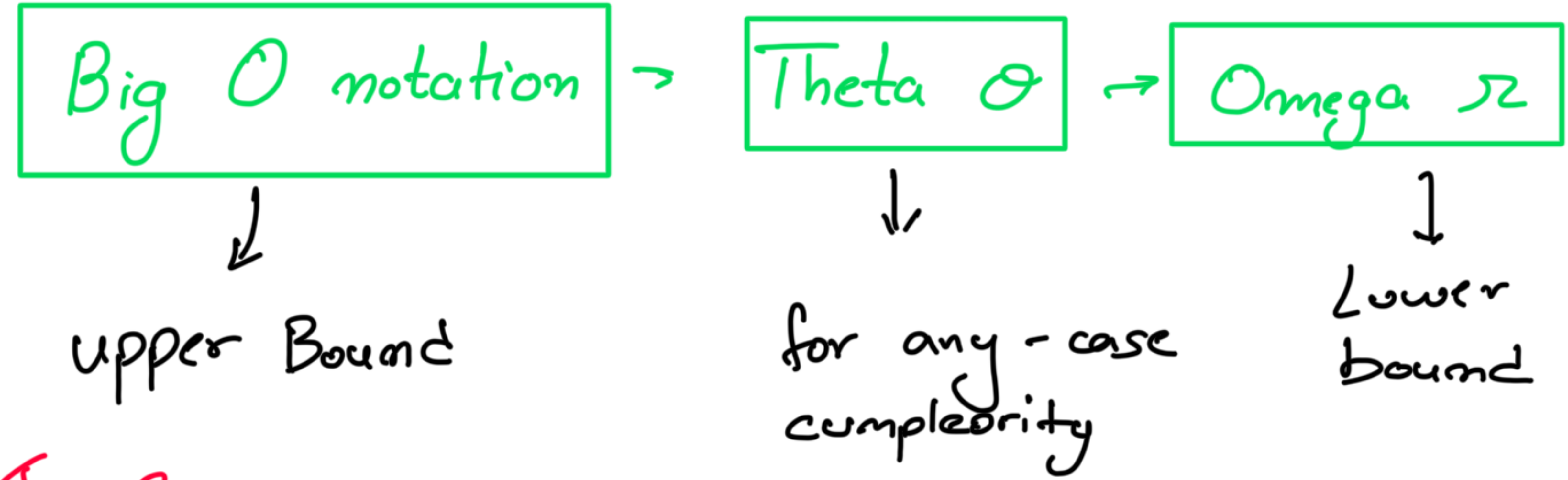
→ The amount of time taken by an algorithem to run.

It is use to find and compare two program

If i run heavy code on fast machine and light code on slow machine then definatly fast machine take no time.

Time complexity is not based on machine

Notation for time complexity



Big O notation → Theta θ → Omega Ω

Big O notation ↓ upper Bound

Theta θ ↓ for any-case cumplexity

Omega Ω ↓ Lower bound

Imp

Upper Bound: Maximum time a algorithem can take.

Like max to max time taken by any algorithem.

Lower Bound : Min to min time any algorithem can take.

Big O Notation

for for-loops time complexity is based on <u>no. of iterations</u> .

Constant time → O(1)

Linear time → $O(n)$

Logarthmic time → $O(n\log n)$

Quadratic time → $O(n^2)$

Cubic time → $O(n^3)$

How to find all of this time

linear loop

Take an example of loop that iterating 0 to n.

```
  for( i = 0 → n)
{
     cout << " Hellow" ;
}
```

Here time complexity is $O(n)$

$O(n)$ → loop (itrations)

Here o denoted the or represent the loop.

n is denoting How many time loop will itrate.

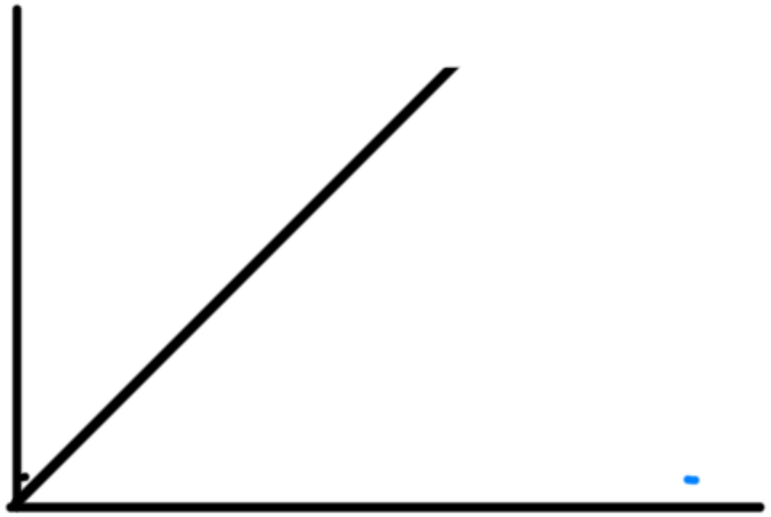
Loop is going 0 to 5 or 2 to m

$O(5)$ $O(m)$

$O(5-2)$ $O(m-2)$

$$\boxed{O(\text{loop Ending} - \text{Starting})}$$

Graph



$O(n)$

For nested loops.

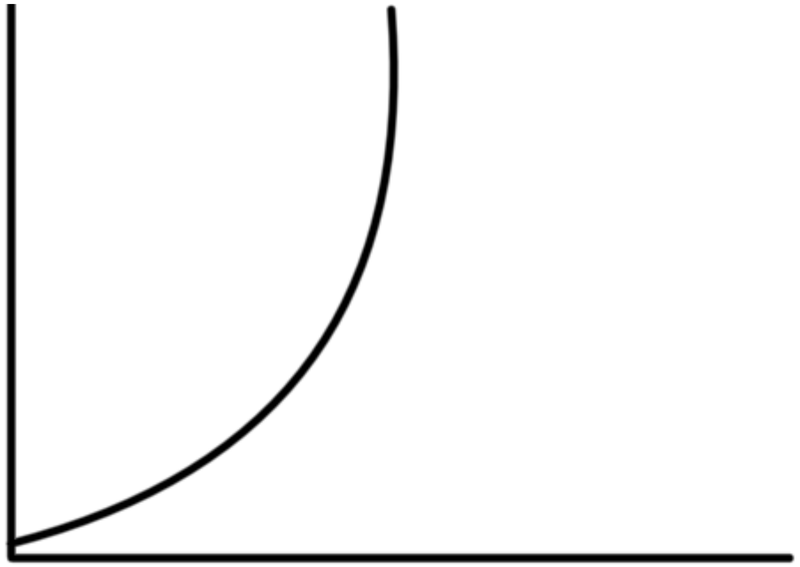
```
for( 0 → n )
{
    for ( 0 → n )
    {

    }
}
```

In nested loops time complexity of inner loop is O(n) then outer loop itrate O(n)

$$O(n) \times O(n)$$

$$= O(n^2)$$

Graph



$O(n^2)$

what if we have more than 2 nested loops

```
for ( )
{
        for ( )
          {
                  for ( )
                  {      . . . . —
```

```
      }
    }
  }
```

Then the time complexity will be

$$O(n) \times O(n) \times O(m) \times O(a) \dots$$

$$\boxed{O(n^2 \times \underbrace{m}_{\text{If some loops go } O \to m} \times \underbrace{a}_{\text{If some go } O \to a} \times \dots\dots)}$$

Multiple diffrent loops / not nested :

what if loops are parallel like :-

```
for ( o → m ) {    }
for ( o → m ) {    }
for ( o → a ) {    }
```

Here we add all of the O( ) because all are parallel not nested . they work idividulaly

Here $\boxed{O(n) + O(m) + O(a)}$

Question

$$f(n) = 2n^2 + 3n$$

ignore lower degree because we are finding upper Bound

$$2n^2 + 3n \leftarrow \text{ignore}$$

$$n^2$$

$$\boxed{\text{ans} = O(n^2)}$$

$f(n) = 4n^4 + 3n^2 = \boxed{O(n^4)}$

$f(n) = N^2 + \log n = \boxed{O(N^2)}$

$f(n) = 3n^3 + 2n^2 + 5 = O(n^3)$

$f(n) = 2001 = O(c)$ constant time...

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

$f(n) = n/4 = O(n)$

$f(n) = n^5/4 = O(n^5)$