Date 11/10/2023 Page

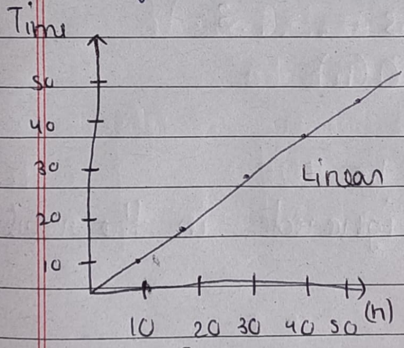
Day - 24

Time and Space Complexity

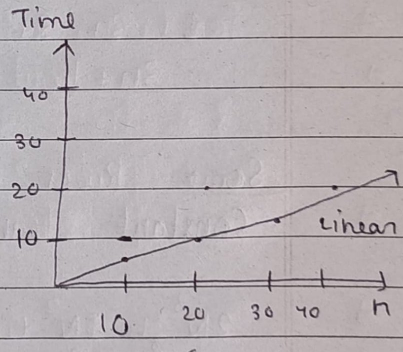
=> Time complexity is not time taken by any computer.

=> Because a fast or new computer can ~~also~~ solve the same problem ~~with~~ in less time.

Time Complexity: It is the total time taken by an algo. to run - as a function of length of the i/p.



(old)



(new)

=> Time = n

Time = n/2

Time = n

=> But both the curve are linear that means both are same.

=> Time = n

=> There is no existence of constants.

Date ______ Page ______

⇒ We have to handle worst case.

⇒ Worst case: O (Big-Oh)

⇒ Best case: Ω (Omega)

⇒ Avg. Case: Θ (Theta)

⇒
```
for ( i=1 ; i<=n ; i++){
        cout << "chamka";
3
```

$1 \quad 2 \quad 3 \quad \ldots \quad n$

⇒ $1 + 3 + 3 + 3 + 3 + \ldots 3$

⇒ $3n + 1 = O(n)$

Some Rules will be

⇒ Constant term ignored in the result.

⇒ $2N^3 + 3N^2 + N$

⇒ $N^3 + \cancel{N^2} + \cancel{N}$

⇒ N^3

Linear Search

```
for( i=0 ; i<n ; i++ ){
    if( arr[i] == x ){
        cout << "Got";
        break;
}   }
```

Date ______ Page ______

6	7	5	1	8

=> Let $x = 6$ →

=> It will find at 0^{th} index that means takes 1sec (let).

=> So, it is best case →
$\Omega(1)$

=> Let $x = 8$,

=> This is worst case, $O(n)$

=> Let $x = 5$,

=> $\Theta(n)$

=> If the algo doesn't depend on the i/p. Then $O(1)$, $\Omega(1)$, $\Theta(1)$ are their T.C.

Another ex:

```
for (i=1; i<=n i++) → n
    for( j=1 ; j<=n; j++){ → n
        cout << "Chamka";
    }
```

So, $O(n^2)$

So, $1+2+\dots+n$

$n*(n+1)/2$

$\frac{n^2+n}{2} = O(n^2)$

=>
```
for( i=1; i<=n; i++)
    for( j= 1; j <=i; j++){
        cout<< "Chamka";
    }
```

=>

$i=1$	$i=2$	- - -	$i=n$
$j = 1$ to 1	$j = 1$ to 2		$j = 1$ to n
1 time	2 time		n time

Date ____ Page ____

$\Rightarrow$
```
for(i=1; i<=n; i++)
    for(j=1; j<=i^2; j++){
        cout << "Chamka";
}
```

$i=1$	$i=2$		$i=n$
$j=1$ to 1	$j=1$ to 4	...	$j=1$ to n^2
1 time	4 time		n^2 time

$\Rightarrow$ $n(n+1)(2n+1)/2$

$\Rightarrow$ $n^3+n^2+\ldots$

$\Rightarrow$ $O(n^3)$

$\Rightarrow$
```
for(i=1; i<=n; i=i*2){
    cout << "chamka"
}
```

$i=1(2^0)$	$i=2(2^1)$	$i=4(2^2)$		$i=n(2)^k$
1	1	1	...	1

$1+1+1+\ldots+1$

It show that, Chamka prints (power +1) time

So, when, $n=2^k \longrightarrow$ ~~$O(k+1)$~~ $\longrightarrow O(k+1)$

Now,

$\log n = k \log 2$

$k = \dfrac{\log n}{\log_2}$

$k = \log_2 n$

So, $O(\log_2 n+1)$

$\Rightarrow O(\log_2 n)$

Date ________ Page ________

Half Half Half

$\Rightarrow$ 1 - 2 - ~~3~~4 - 8 - 16 - 32 - - - n

$\Rightarrow$ $(\log_2 n)$

1 - 3 - 9 - 27 - . . . . n

$\Rightarrow$ $(\log_3 n)$

$\Rightarrow$

```
for( i = n/2; i <= n; i++)      -> n/2
for( j = 1; j <= n; j = 2 x j)  -> log2 n
for( k = 1; k <= n; k = 2 x k)  -> log2 n
{
    cout << " Chamka";
}
```

$n * \log_2 n * \log_2 n$

$O(n(\log_2 n)^2)$

```
for( i = 1; i <= n; i++)
  for( j = 1; j <= n; j = j + i)
  {  cout << " chamka"; }
```

$\Rightarrow$

$i = 1$	$i = 2$	$i = 3$	- -	$i = n$
$j = 1$ to n	$j = \cancel{1}\ n/2$	$j = n/3$		$j = n/n = 1$
n time	$n/2$	$n/3$		1 time

$\Rightarrow$ $n + n/2 + n/3 + \ldots + 1$

$\Rightarrow$ $n + n/2 + n/3 + \ldots + n/n$

$\Rightarrow$ $n(\underbrace{1 + \frac{1}{2} + 1/3 + \ldots + 1/n}_{\text{Harmonic Series}})$

$\Rightarrow$ $O(n(\log_e n))$

* **Space Complexity:**

⇒ It is the amount of space taken by an algo. – as a fun^n of length of i/p.

⇒ **Auxillary space:**
Here, given things don't come.

⇒ **Total space Complexity:**
Here, everything includes.

Ex:

n

4	6	5	3	2

Given

n

16	36	25	9	4

Answer

⇒ Auxillary Space: $n \to O(n)$

Total space: $n + n = 2n \to O(n)$

⇒ Worst

$O(N!)$
$O(2^N)$
$O(N^3)$
$O(N^2)$
$O(N\log N)$
$O(N)$
$O(\sqrt{n})$
$O(\log n)$
$O(1)$

Best