(03/06/2023)

* Time complexity

Time Required to Run the code/program

OR

Time complexity of an algorithm quantifies the amount of time taken by a program to run as function of length of input.

Eg 1:

```
int n;
cin>>n;
int a=0;
for(int i=1; i<=n; i++)
{
    a=a+1;
}
```

Explan:- n=5

i=1,2,3,4,5.

linear ∝ to n.

eg 2:

```
int n;
cin>>n;
int a=0;
for(int i=1;i<=n;i++)
{
for(int j=1;j<=n;j++)
{
a=a+1;
}
}
```

i=1 j=1 to n.

i=2 → n.

i=3 = n.

i=n n

n×2

$\propto n^2$

* Space Complexity

How much space Required for run the code/prog

- Running time should be minimum
- Space should be minimum to exect/Run the program

Then program should be efficient & fast

- Space complexity

E.g.

```
int n;          ← 4 bytes
cin >> n;
int a = 0;      ← 4 bytes
for (int i = 1; i <= n; i++)   ← 4 bytes
{
    a = a + 1;
}
```

- Total Space Required is 12 Bytes
- Program space will be constant it does not depend on Input.

int n, int a = 0 ; int i these all are Variable & Variable is of 4 bytes so 4 × 3 = 12 Bytes

E.g. Array of size N.

int arr [n]; (int n)

Space complexity is ∝ to n.

* Cases

i. Worst case.
ii. Average case.
iii. Best case.

E.g.

Search . 20 10 5 100 300 17 238

x = 238

x = 20

- The element which got in first comparision is called Best case
- The element which need to compare multiple time, or at last, or element not got is called worst case Program Required more time

Worst case Time Complexity :- at last element will get n.

Best case Time Complexity :- got element at first position Constant O(1).

- Space complexity

e.g.
```
int n;          ← 4 bytes.
cin >> n;
int a=0;        ← 4 bytes.
                ← 4 bytes
for(int i = 1; i<=n; i++)
{
    a = a+1;
}
```

- Total space Required is 12 Bytes.
- Program space will be constant it does not depend on Input.

∴ int n, int a=0; int i these all are Variable & Variable is of 4 bytes so $4 \times 3 = 12$. Bytes.

e.g. Array of size N.

int arr [n];   (int n).

Space complexity is $\propto$ to n.

* Cases

i. Worst case
ii. Average case
iii. Best case

e.g

Search 20 10 5 100 300 17 238.

$x = 238$.

$x = 20$.

- The element which got in first comparision is called Best case.
- The element which need to compare multiple time, or at last, or element not got is called worst case program Required more time.

Worst case Time Complexity :- at last element will get n.

Best case Time complexity :- got element at first position Constant $O(1)$.

- Average case

$$\frac{\text{Total time}}{\text{Total no of cases}} = \frac{1+2+3+\dots n}{n} = \frac{\frac{n(n+1)}{2}}{n}$$

$$\boxed{\frac{n+1}{2}}$$

Best case complexity : constant

Worst case complexity : $\propto$ n (Directly proportional to n)

Average case complexity : $\propto$ n

Denoted by

Best case : Ω (Big omega) $\rightarrow \Omega(1)$

Worst case : O ($\rightarrow O(n)$

Average : Θ $\rightarrow \Theta(n)$

→ Time complexity

e.g

```
int n,m;
cin>>n>>m;
for(int i=1; i<=n;i++)
{
    a=a+i;      → n.
}
for (int j=1; j<=m;j++)
{
    a=a+1;      → m.
}
```

O/p

$O(n+m)$

e.g

```
int n,m;
cin>>n>>m.
int a=0;
(int i=1; i<=n; i++)
{
    for(j=1; j<=m;j++)
    {
        a=a+ rand();
    }
}
```

O/p: $O(nm)$

Date
Page

e.g

```
Int n,m;
  cin>>n>>m;
  for(int i=1;i<=n;i++){   ← n
    for(int j=1;j<n;j++)   ← m.
    {
      a=a+ rand();
    }
  }
  For (int i=1;i<=n;i++){   ← n.
      a=a+rand();
```

O/P. Both loops are there then (n×m).

$O(n \times m + n)$

e.g

```
int n;
cin>>n;
int a=0; i=n;
while(i>=1)
{
  a=a+1;
  i/=2;
}
```

O/P:

$\frac{n}{2^k} \geq 1.$

= Comparision of function

	n	n^2	n^3
n=1	1 unit	1 unit	1 unit
n=2	2 unit	4 unit	8 unit
n=3	3 unit	9 unit	27 unit

$O(n) < O(n^2) < O(n^3)$

$O(n)$ is faster than (n^2) (n^3).

$O(n^3)$ is slower

*	n	logn.
n=1	1 unit	0 unit.
n=2	2 unit	$\log_2 2 = 1$
n=1024	1024 unit	$\log_2 (2)^{10}$ $10\log_2 2$

$O(\log n)$ Require less ~~time~~ time than $O(n)$.

$O(\log n)$ is better than $O(n)$.

* sqrt(n)   log n

log n algorithm ~~theor~~ Run time C is better than Root.