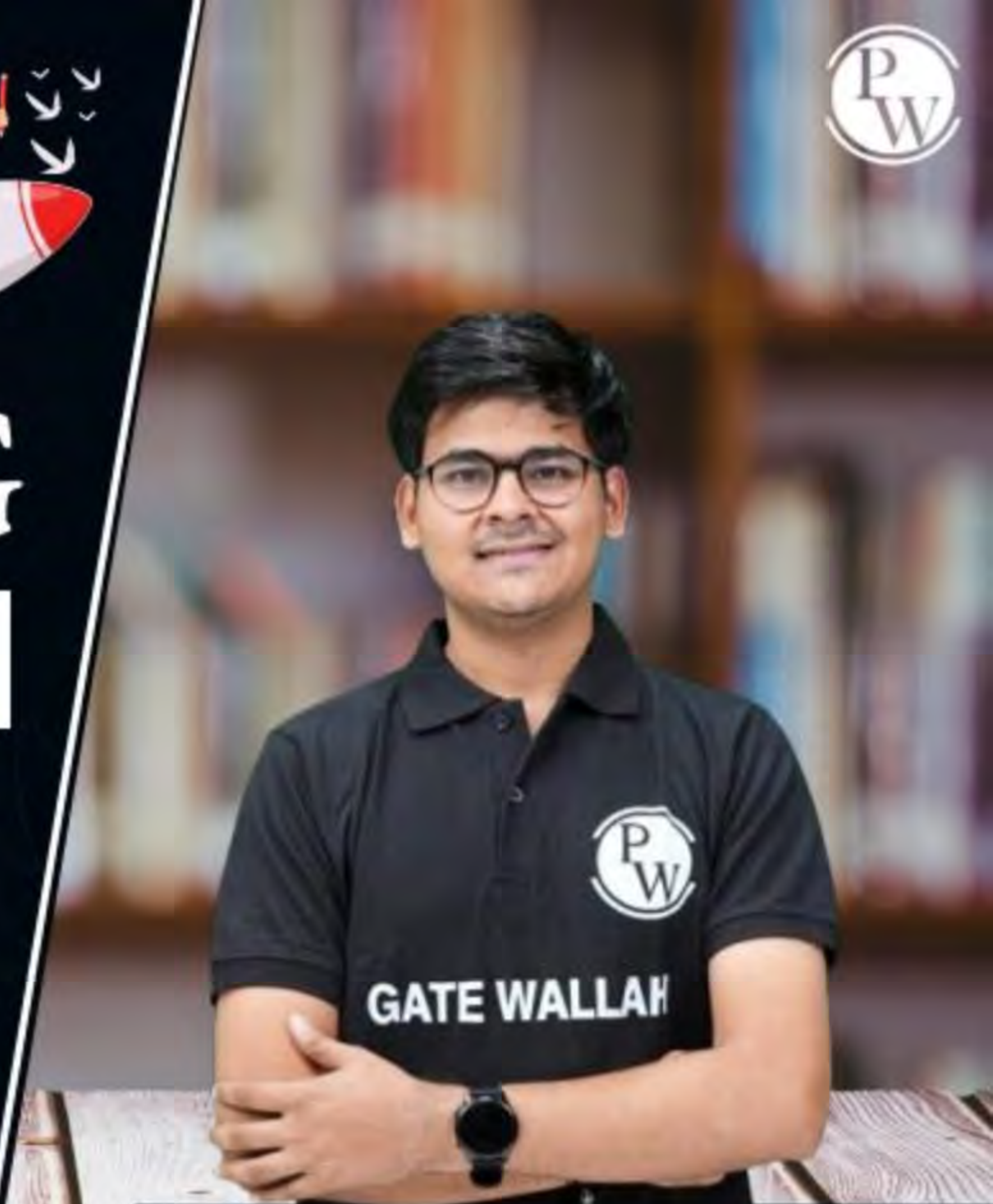
# CS & IT ENGINEERING


## Algorithms

### Analysis of Algorithms

Lecture No.- 12




By- Aditya sir

# Recap of Previous Lecture

Topic

Topic

Topic

Time Complexity of Recursive Algo.

Loop Complexities

# Topics to be covered

**Topic** Loop Complexities Advanced.

**Topic** Space Complexity

* Nested loops Complexity:

↳ (loop within loop)

2 types:-

1) Nested Dependent loops

2) Nested independent loops

eg1 :

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

$\rightarrow TC = O(n^2)$

Independent loops

$i=1 : j : 1 \longrightarrow n : O(n)$

$i=2 : j : 1 \longrightarrow n : O(n)$

$\vdots$

$i=n : j \longrightarrow 1 \text{ to } n : O(n)$

$\Big\} \; O(n^2)$

* Mutually Exclusive loops:-

(not nested)

```
Algo AJ(n, m)
{
    for(i=1; i<=n; i++)
    {
        print(i)
    }
    For(j=1; j<=m; j++)
    {
        print(j)
    }
}
```

$\text{for loop over } i \rightarrow O(n)$

$\text{for loop over } j \rightarrow O(m)$

TC of AJ(n, m)

$\rightarrow O(n+m)$

$\Rightarrow O(\max(n, m))$

```
Algo AJ(n)
{
   for(i=1;i<=n;i++)
   {  print(i)
   }
   for(i=1;i<=n;i++)
   { For(j=1;j<=n/2;j++)
       { print(i+j)
       }
   }
   for(i=1;i<=n;i++)
     For(j=1;j<=n/5;j++)
       For(k=1;k<=n;k++)
       {  break
       }
}
```

$\to O(n)$

$O(n * n/2) = O(n^2)$

$O(1)$

$O(n^2)$

$O(n+n^2+n^2) = O(n^2)$

TC of AJ(n)?

A) $O(n)$

B) $O(n^2) \to 58.7\%$

C) $O(n^3) \to 31.5\%$

D) $O(n^4)$

① for(i=1; i<=n; i=i+1)

{ print(i)

}

$\longrightarrow O(n)$

② For(i=1; i<=n; i=i+2)

{ print(i)

}

$\longrightarrow 1, 3, 5 \ldots \Big\} \rightarrow O(n/2) \rightarrow \underline{O(n)}$

③ for(i=1; i<=n; i=i+7)

{

print(i)

}

$\longrightarrow \underbrace{1, 8}_{+7}, \underbrace{8, 15}_{+7}, 15, 22 \ldots \approx O(n/7) = \underline{\underline{O(n)}}$

Generalised Form:-

```
for(i=1; i<=n; i=i+b)
  {
      print(i)
  }
```

$\rightarrow O(n/b) = O(n)$

① for(i=1; i<=n; i=i*2)

{ print(i)

}

way1

$i = 1,\ 1*2=2^1,\ 2^2,\ 2^3 \ldots 2^k$

Lets assume loop runs for k times

$2^k = n$ (for last iteratn)

$\boxed{k = \log_2 n}$

way2

Let n=16.

$1 \Rightarrow, \underbrace{2, 4, 8, 16}_{4 \text{ times}}$

$\log_2 16 = 4$

② 
```
for(i=1; i<=n; i=i*5)
{ print(i)
}
```

$i = 1, 5^1, 5^2 \ldots 5^k$

$k^{th}$ itratn (last iter), $5^k = n$

$k = \log_5 n$

$\longrightarrow O(\log_5 n)$

## Generalised Form

```
for (i=1; i<=n; i=i*b)
{
    print(i)
}
```

$\Longrightarrow \boxed{O(\log_b n)}$

(Q) Algo AJ(n)

```
{
    for (j=1; j<=n/2; j++)
    {  print(j)
    }

    i=0
    while(i<=n)
    {  print(i)
       i=i+3
    }

    AJ2(n)
}
```

$\rightarrow O(n/2) \rightarrow O(n)$

$\rightarrow O(n/3) \rightarrow O(n)$

$\rightarrow O(\log n)$

$\rightarrow$ overall $O\left(\frac{n}{2}+\frac{n}{3}+\log n\right)$

$\rightarrow \boxed{O(n)}$

81% ✓

80%

Assume AJ2(n) is $O(\log n)$

$\rightarrow$ TC of AJ(n) = ?

A) $O(n)$ ✓

B) $O(n^2)$

C) $O(\log_3 n)$

D) $O(n\log_3 n)$

$O(n \times n \times 2) = \underline{O(n^2)}$

min $\underline{\underline{70\%}}$

$\boxed{49.5\%}$

★
```
for (i=1; i<=n; i++)            → n times
{ for (j=1; j<=n; j++)          → n times
  { for (k=n/2; k<=n; k=k+n/2)
    {
      print("AJ")
    }
  }
}
```

only runs 2 times

$k = n/2$

$n/2 + n/2 = n$

$n + n/2 = \frac{3n}{2}$ stop

A) $O(n)$

B) $O(n^2) \rightarrow 49.5\%$

C) $O(n^3) \rightarrow 28\%$

D) $O(n^2 \log n)$

For (i=1; i<=n; i=i*5)

$O(\log_5 n)$

6) Incr

```
i=1
while(i<=n)
{
    i=i*5
    print(i)
}
```

$i = 5^1, 5^2, 5^3 \dots 5^k = n$

$k = \log_5 n$

7) Decr

```
j=n
while(j>0)
{
    j=j/5
    print(j)
}
```

$n = 5^k$

$k = \log_5 n$

last iter

$j = n, n/5, n/5^2, n/5^3 \dots n/5^k = 1$

8) 

```
c=0
for(i=1; i<=n; i=i+2)
    for(j=1; j<=m; j=j*2)
    {
        c=c+1
    }
```

$n/2$

$\rightarrow O(n)$

$\rightarrow O(\log_2 m)$

$\rightarrow \log_2 m$

If $n = 12$, then the Final value of $c = ?$

72 ✓

144 ✗

$\Rightarrow \frac{n^2}{2} = \frac{12 \times 12}{2} = \boxed{72}$

Assume $m = 2^n$ then value of c in terms of n?

$1 \longrightarrow m$ $*2$

$\underline{(\log_2 m)}$

$c \rightarrow \frac{n}{2} * \log_2 m$

$\Rightarrow \frac{n}{2} * \log_2 2^n$

$\Rightarrow n/2 * n = \boxed{\frac{n^2}{2}} \rightarrow \underline{\underline{O(n^2)}}$

$85\%$

$64.5\%$ ✓

## ☆ Dependent nested loops

(9)

```
c=0
for(i=1; i<=n; i++)
{ for(j=i; j<=n; j++)
    { c=c+1
    }
}
```

$O(n^2)$

After the code ends the value of c is ____ ?

A) $O(n)$

B) $O(n^3)$

C) $O(n\log n)$

D) $O(n^2)$

Soln:

$i \to n$

$i=1 \quad j: 1 \longrightarrow n \quad : \quad n$

$i=2 : \quad j: 2 \longrightarrow n \quad : \quad (n-1)$

$i=3 : \quad j: 3 \longrightarrow n \quad : \quad (n-2)$

$\vdots$

$i=n : \quad j: n \longrightarrow n \quad$ 1 time

Total iterations

$= n + (n-1) + (n-2) \ldots 1$

$= \dfrac{n(n+1)}{2}$

$= \dfrac{n^2+n}{2}$

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

Imp

(Q)

```
a=1, b=1
while( a<=n)
{
    b = b+1
    a = a+b
}
```

TC of this code ?

A) $O(n)$ → 61.5 %

B) $O(\sqrt{n})$ → 13.2 %

C) $O(n^2)$

D) $O(\log n)$

13.2 %

Soln:-

| initial | 1st | 2nd | 3rd | ...... $k^{th}$ iteratn. |
|---|---|---|---|---|
| $b=1$ | $b=2$ | $b=3$ | $b=4$ | $b=k+1$ |
| $a=1$ | $a=1+2$ | $a=3+3$ | $a=6+4$ | $a=1+2+3\cdots(k+1)$ |
| | $=3$ | $=6$ | $=10$ | $=\dfrac{(k+1)(k+1)}{2}$ |
| $\downarrow$ | $\downarrow$ | $\downarrow$ | $\downarrow$ | |
| $1$ | $1+2$ | $1+2+3$ | $1+2+3+4$ | $\approx O(k^2)$ |

If loop ends after <u>k iterations</u>.

$$\frac{(k+1)(k+2)}{2} = n \longrightarrow \boxed{O(\sqrt{n})}$$

$$\simeq k^2 = O(n)$$

$$\boxed{k = O(\sqrt{n})}$$

# Space Complexity

## Topic : Space Complexity

We define the space used by an algorithm to be the number of memory calls (or words) needed to carry out the computation steps required to solve an instance of the problem excluding the space allocated to hold the input. In other word, it is only the work space required by the algorithm.

```
Algo( input)
{
    ____
    ____
    ____
}
```

Space Complexity

$\Longrightarrow$ Auxilary Space

(Additional space except the input)

eg: linear search.

input

```
Algo AJSearch(A[n], x)
{
    for(i=1; i<=n; i++)
        if(A[i]==x)
            return i
}
```

TC $\Rightarrow$ $O(n)$

SC $\Rightarrow$ A[n] ✗   x ✗

variables 'i'

$O(1)$ ↳ constant space.

eg2:

```
Algo SumAJ(A[n], n)
{   sum = 0
    for (i = 1; i <= n; i++)
    {
        sum = sum + A[i]
    }
}
```

TC = $O(n)$

SC =

↳ variables

sum

i

SC = $O(1)$

## Recursion

egs:

```
Algo RecursiveSum(A, n)
{
    if(n==1)
        return A[n]
    else
        return (A[n] + RecursiveSum(A, n-1))
}
```

TC ⇒ $T(n) = T(n-1) + a$

⇒ $O(n)$ ✓

Space Complexity $= \boxed{O(n)}$

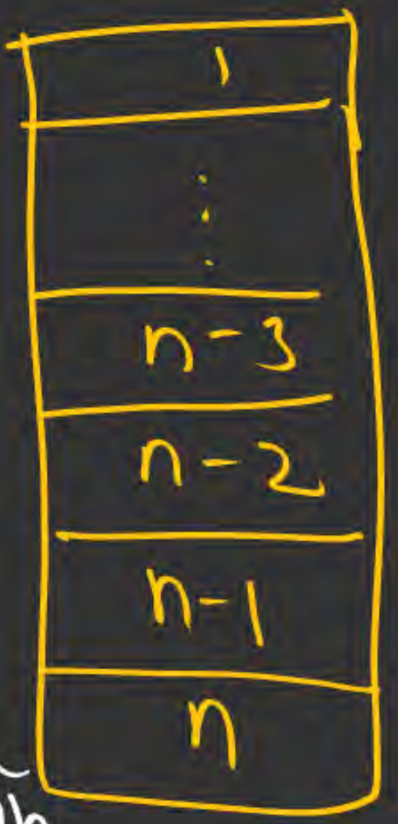
## Auxilary Space

Recursion Stack

↓

Space Complexity

⇒ the max size of Recursion stack at any time during Recursion.




Size of Recursion Stack → 'n'

85%

eg:-

```
Algo AJ(n)
{ if(n==1)
      return ,
  else
    return AJ(n/2)
}
```

SC = ?

A) $O(n)$ → 47%

B) $O(n^2)$

C) $O(n\log n)$

D) $O(\log n)$ → 34%

Soln:

**Recursion Stack:**

| |
|---|
| $AJ(n/2^k)$ |
| ⋮ |
| $AJ(n/2^3)$ |
| $AJ(n/2^2)$ |
| $AJ(n/2)$ |
| $AJ(n)$ |

Size of Recursion stack is $k$

Here For Termination

$$\frac{n}{2^k} = 1$$

$$2^k = n$$

$$\boxed{k = \log_2 n}$$

$$SC \rightarrow O(\log_2 n)$$

GATE : 2017 TC = ? (2 marks) 54%

Dependent nested

Algo AJ(n)

```
{
  for(i=1; i<=n; i=i+1)
  {
    for(j=1; j<=n; j=j+i)
    {
      print(i, j)
    }
  }
}
```

A) $O(\sqrt{n})$

B) $O(n\log n)$

C) $O(n^2)$

D) $O(n^2\log n)$

Soln:-

$i=1$: `for(j=1; j<=n; j=j+1)` $\longrightarrow n$

$i=2$: `for(j=1; j<=n; j=j+2)` $\longrightarrow n/2$

$i=3$: `for(j=1; j<=n; j=j+3)` $\longrightarrow n/3$

$\vdots$

$i=n$ `for(j=1; j<=n; j=j+n)` $\longrightarrow n/n$

Total Complexity

$$= \frac{n}{1} + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} \cdots \frac{n}{n}$$

$$= n\left[\frac{1}{1} + \frac{1}{2} + \frac{1}{3} \cdots \frac{1}{n}\right]$$

$$= n * \sum_{i=1}^{n} \frac{1}{i} \rightarrow \log n$$

$$TC = O(n * \log n)$$

$\underline{80\%} \longrightarrow \underline{40\%}$

PYQ: GATE 2013

```
Algo AJ(n)
{
  int i, j, k=0
  for(i=n/2; i<=n; i=i+1)
  {
    for(j=2; j<=n; j=j*2)
    {
      k=k+n/2
    }
  }
  return(k)
}
```

(Q) The value returned by function is order of ___?

A) $O(n)$

~~B) $O(n\log n)$~~

C) $O(n^3\log n)$

D) $O(n^2\log n)$

**Soln:-**

for (i = n/2 ; i <= n ; i = i+1) $\longrightarrow$ $\underline{n/2 \text{ times}}$

for (j = 2 ; j <= n ; j = j * 2) $\longrightarrow$ $\log_2 n$ times

$2 \to 2^2 \to 2^3 \to 2^k$

$2^k = n$

$\boxed{k = \log_2 n}$

{

k = k + n/2 $\Longrightarrow$ Runs $\left(n/2 * \log n\right)$ times.

}

(Q.1) Time Complexity $\Rightarrow$ $O(n\log_2 n)$

(Q2) value of K $\Rightarrow$ $\frac{n}{2}\log n * n/2 \Rightarrow O(n^2 \log n)$

[MSQ]

HW

#Q. Consider functions Function_1 and Function_2 expressed in pseudocode as follows:

| Function_1 | Funtion_2 |
| --- | --- |
| While n > 1 do<br>  for i = 1 to n do<br>    x = x + 1;<br>  end for<br>  n = [n/2];<br>end while | for i = 1 to 100*n<br>do<br>  x = x + 1;<br>end for |

Let $f_1(n)$ and $f_2(n)$ denote the number of times the statement "x = x + 1" is executed in Funtion_1 and Function_2, respectively.
Which of the following statement is/are TRUE?

A. $f_1(n) \in \theta(f_2(n))$

B. $f_1(n) \in \omega(f_2(n))$

C. $f_1(n) \in 0(f_2(n))$

D. $f_1(n) \in 0(n)$

# THANK - YOU

Telegram Link: https://t.me/AdityaSir_PW