CS & IT ENGINEERING

Algorithms

Analysis of Algorithms

Lecture No.- 08

By- Aditya Jain sir

Topics to be Covered

- Topic
- Topic
- Topic

Loop Complexity

Space Complexity

About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 12,000+ students & working professions in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.

Telegram

Telegram Link for Aditya Jain sir: https://t.me/AdityaSir_PW

Topic : Analysis of Algorithms

Noted loops complexity

Loop inside loop

2 types:-

1. Nested dependent loops
2. Nested independent loops

Topic : Analysis of Algorithms

Eg.1.

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

Nested independent loop

Topic : Analysis of Algorithms

Eg.1.

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

for (i = 1; i ≤ n; i++) ⟶ O(n)

for (j =1; j ≤ n; j++) ⟶ O(n)

overall TC : O(n*n) = O(n²)

Topic : Analysis of Algorithms

Mutually Exclusive loops:

(Not nested)

```
Algo AJ(n, m)
{
    for (i =1; i ≤ n ; i++)
    {
        printf (i);
    }
    for (j = 1; j ≤ m; j++)
    {
        printf(j);
    }
}
```

$\longrightarrow O(n)$

$\longrightarrow O(m)$

$\rightarrow$ overall TC

$= O(m+n)$

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

Topic : Analysis of Algorithms

```
Algo AJ(n)
{
    int i, j, k
    for (i=1; i ≤ n; i++)
    {
        print (i);
    }                                   → O(n)

    for (j =1; j ≤ n; j++)
    {
        print (i+j);                    → O(n)
    }

    for ((i=1; i ≤ n; i++)
      for (j =1; j ≤ (n/5); j++)
        for (k =1; k <=n; k++)          O(1)
        {
            break;
        }                               O(n²)
}
```

- A $O(n)$
- B $O(n^2)$
- C $O(n^3)$
- D $O(n^4)$

Topic : Analysis of Algorithms

1.
```
for (i =1; i ≤ n; i =i+1)
{
    printf(i);
}
```
→ n
O(n)

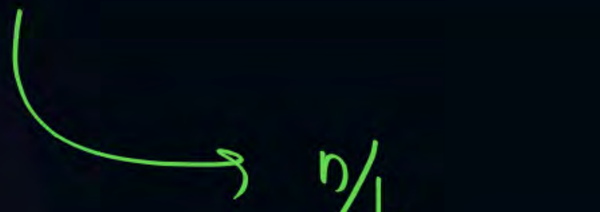
2.
```
for (i=1; i ≤ n; i=i+2)
{
    printf(i);
}
```
→ n/2
O(n/2) = O(n)

3.
```
for (i=1; i ≤ n; i =i+7)
{
    printf(i);
}
```
→ n/7 → O(n/7) = O(n)

Topic : Analysis of Algorithms

Generalized from

```
for (i=1; i ≤ n; i =i+b)  [where b  >0 ]
{
        printf(i);
}
```



n/b

TC: O(n)

Topic : Analysis of Algorithms

1.
```
for (i=1; i ≤ n; i=i*2)
{
    printf(i);
}
```

TC: $O(\log_2 n)$

Way-1

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

Let assume loop runs for k times

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

$\boxed{k = \log_2 n}$

Way-2

Let n = 16

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

$= \log_2 {}^{16}$

$= 4$

Topic : Analysis of Algorithms

2.
```
for (i=1; i ≤ n; i=i*5)
{
    printf(i);
}
```

TC: $O(\log_5 n)$

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

At k^{th} iteration, $5^k = n$

$k = \log_5 n$

Topic : Analysis of Algorithms

Generalized from

```
for (i=1; i ≤ n; i=i*b)
{
        printf(i);
}
```

$TC : O(\log_b n)$

Topic : Analysis of Algorithms

95%
↓
80%

#Q. Algo AJ(n) [AJ2(n) take $O(n^2)$]

```
{
    for (j=1, j ≤ (n/2); j++)
    {
        printf(j);
    }

    i=0;
    while (i ≤ n)
    {
        printf(i);
        i=i+3;
    }

    AJ2(n)
}
```

O(n)

O(n)

$O(n^2)$

A $O(n)$

B $O(\log_3 n)$

C $O(n^2)$

D $O(n\log_3 n)$

Topic : Analysis of Algorithms

#Q.

```
For(i=1; i ≤ n; i++)
{
    for (j=1; j ≤ n; j++)
    {
        for(k=n/2 ; k ≤ n; k=k+(n/2))
        {
            printf("AJ");
        }
    }
}
```

- A O(n)
- B $O(n^2)$
- C $O(n^3)$
- D $O(n^2 \log n)$

Topic : Analysis of Algorithms

6.
```
i=1;
while (i ≤ n)
{
    i=i*5;
        printf(i);
}
```

$O(\log_5 n)$

Topic : Analysis of Algorithms

7.

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

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

$O(\log_5 n)$

$\frac{n}{5^k} = 1$

$5^k = n \rightarrow k = \log_5 n$

Topic : Analysis of Algorithms

8. where $m=2^n$

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

$n/2$ times

$\log_2 m$

$\frac{n}{2} * \log_2 m = \frac{n}{2} * \log(2^n)$

$= \frac{n}{2} * n = \frac{n^2}{2}$

Topic : Analysis of Algorithms

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

Topic : Analysis of Algorithms

#Q.

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

A. $O(n)$

B. $O(n^3)$

C. $O(n \log n)$

D. $O(n^2)$

$i=1$	$j: 1 \rightarrow n$:	n
$i=2$	$j: 2 \rightarrow n$:	$(n-1)$
$i=3$	$j: 3 \rightarrow n$:	$(n-2)$
	⋮	⋮
$i=n$		1

Total

$= n + (n-1) \dots 1$

$= \frac{n(n+1)}{2} = \underline{O(n^2)}$

Topic : Analysis of Algorithms

#Q.

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

A $O(n)$

B $O(\sqrt{n})$

C $O(n^2)$

D $O(\log n)$

$b = 1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 4 \longrightarrow \cdots \longrightarrow k$

$a = 1 \rightarrow 2+1 = 3 \rightarrow (1+2+3) \rightarrow (1+2+3+4) \rightarrow (1+2+3 \cdots k) \downarrow n$

$\underline{1+2+3 \cdots k} = n$

$$\frac{k(k+1)}{2} = n$$

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

$$k^2 \approx n$$

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

Topic : Space Complexity

We define the space used by an algorithm to be the number of function 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.

Recursive Algo → Recursion stack

Topic : Space Complexity

```
Algo (input)
{
        ..............
}
```

Space complexity

Auxiliary space

(Additional space except the input)

Topic : Space Complexity

Eg.3. Linear search

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

Input

SC:

variable i ⟶ O(1)

Topic : Space Complexity

Eg.4. Algo sum AJ (A[n] , n)

```
{
    sum = 0;
    for (i = 1; i ≤ n; i++)
    {
        sum = sum +A[i];
    }
}
```

X SC

SC : O(1)

Topic : Space Complexity

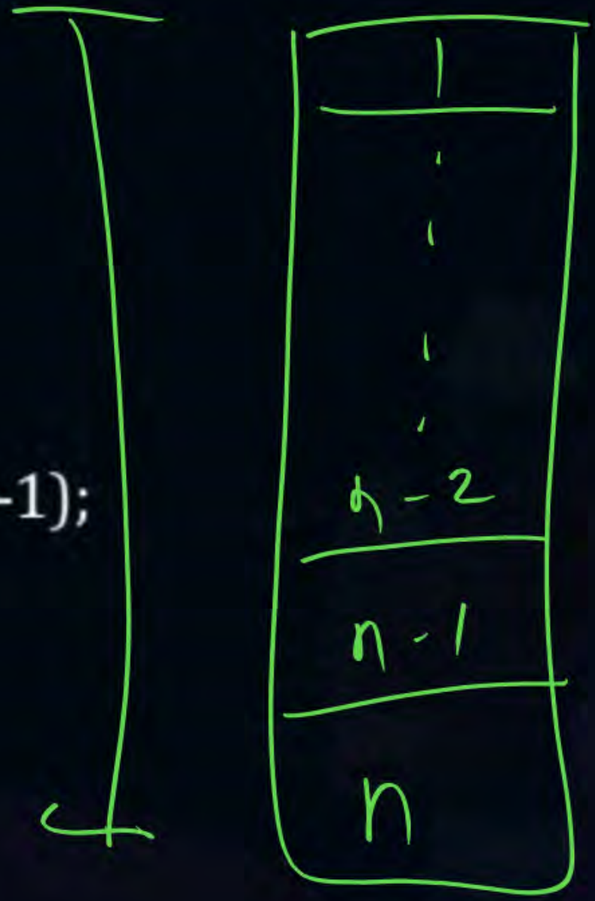
Eg.4. Algo recursive sum (A, n)

```
{
    it (n ==1)
        return A[n]
    else
        return (A[n] + recursive sum (A, n-1);
}
```

$T(n) = T(n-1) + a$

$\hookrightarrow O(n)$

$S.C: O(n)$



Topic : Space Complexity

TC $\Rightarrow T(n) = T(n-1) + a$

$\Rightarrow O(n)$

Space complexity = $O(n)$

Auxiliary Space

Recursion stack

Space complexity

The max size of recursion stack at any time during recursion.

Topic : Analysis of Algorithms

#Q. Algo AJ(n)

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

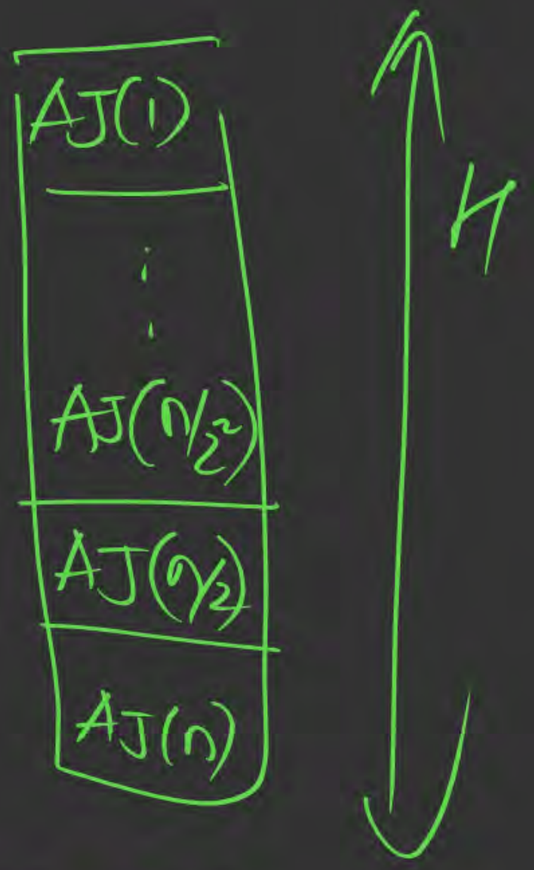
What is the space complexity ?

A. $O(n)$

B. $O(n^2)$

C. $O(n \log n)$

D. $O(\log n)$



$H = SC = O(\log_2 n)$

$n \to n/2 \to n/2^2 \cdots \overset{H}{n/2^h}$

$\frac{n}{2^h} = 1 \Rightarrow n = 2^h$

$h = \log_2 n$

Topic : Analysis of Algorithms

#Q.

```
Algo AJ(n)
{
    for (i=1, i ≤ n; i=i+1)
    {
        for (j=1; j ≤ n; j=j+i)
        {
            printf(i,j);
        }
    }
}
```

TC :

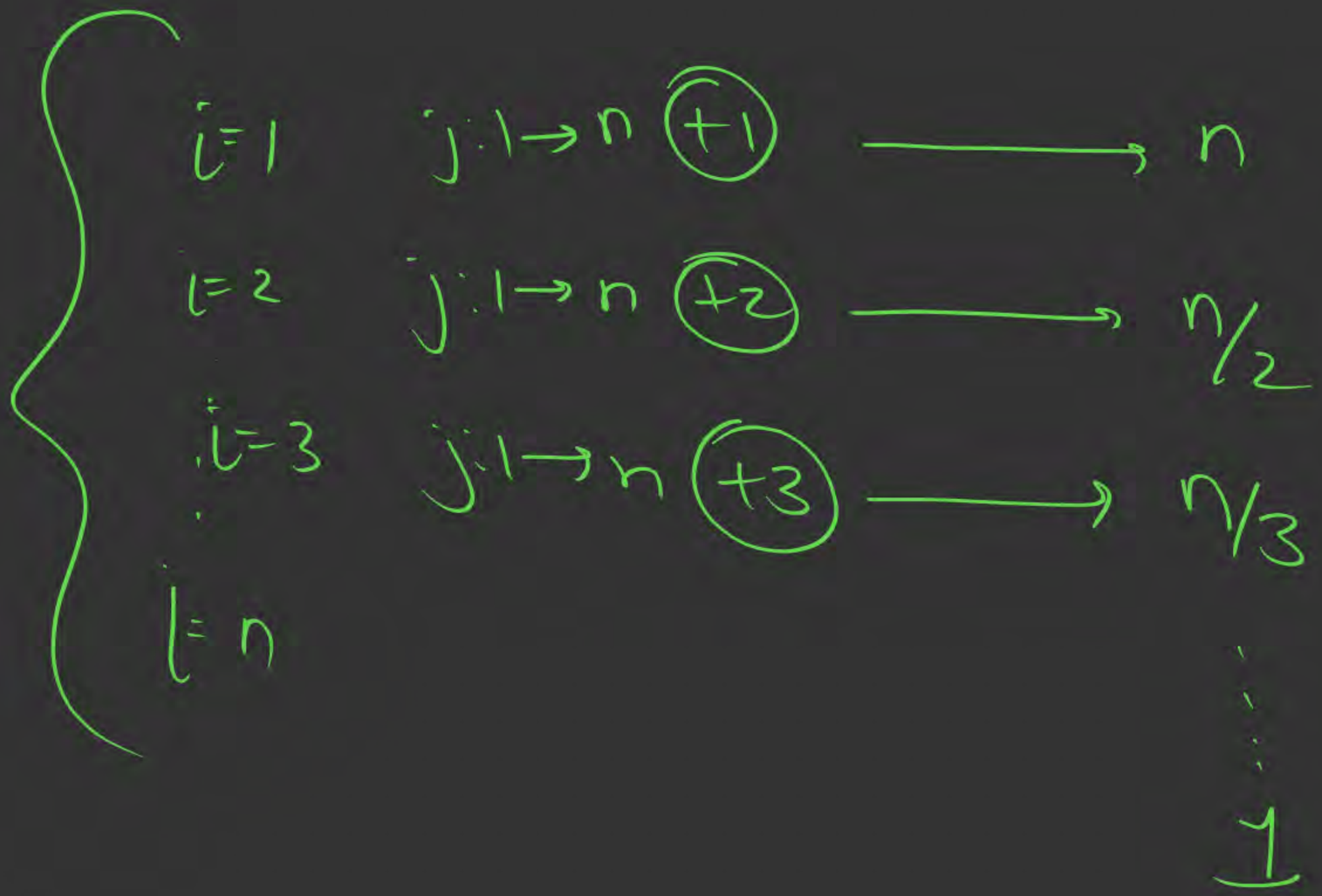
What is the time complexity ?

A. $O(n)$

B. $O(n^2)$

C. $O(n \log n)$

D. $O(\log n)$


i=1
j:1→n
+1
n
i=2
j:1→n
+2
n/2
i=3
j:1→n
+3
n/3
i=n
1

overall TC: $n + \frac{n}{2} + \frac{n}{3} \cdots 1$

$$\int \frac{1}{i} \qquad n\left[\underbrace{1 + 1/2 + 1/3 \cdots 1/n}\right] = n \log n \checkmark$$

Topic : Analysis of Algorithms

#Q.
```
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);
}
```

The return value of the function is ?

A. O(n)

B. $O(n^3 \log n)$

C. O(n log n)

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

Topic : Space Complexity

Time complexity $\Rightarrow O(n \log n)$

Topic : Space Complexity

Value of K $\Rightarrow \frac{n\log n}{2} \times \frac{n}{2} \Rightarrow O(n^2 \log n)$

Topic : Analysis of Algorithms

MSQ

HW

#Q. Consider functions function_1 and function_2 expressed in pseudocode as follow:

Let $f_1(n)$ and $f_2(n)$ denote the number of times the statement "x=x+1" is executed in function_1 and function_2, respectively. Which of the following statement is/are TRUE?

Function_1	Function_2

While n > 1 do
for i = 1 to n do
x = x + 1;
end for
n = [n/2];
end while
	for i = 1 to 100*n
	do
	x = x + 1;
	end for

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

B $f_1(n) \in o(f_2(n))$

C $f_1(n) \in \omega(f_2(n))$

D $f_1(n) \in O(n)$

THANK - YOU