GATE

CRASH COURSE

CS & IT

Algorithms

Analysis of Algorithms

Lecture -02

By - Aditya Sir

Topics to be covered

1. Time Complexity
   Loop Complexities
2. 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 working professions in field of Data Science and Analytics
11. Have been mentoring 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.

Topic : (Lecture Schedule)

1. **Analysis of Algorithms**
   1. Asymptotic Notations $(O, \Omega, \Theta, o, \omega)$
   2. Analysing Non-Recursive Algorithms
   3. Analysing Loops
   4. Analysing Recursive Algorithms
   5. Space Complexity
   6. Problem Solving

* Loop Complexities

1) Single loop (while, for)

2) Multiple loops → a) Mutually exclusive (non-nested)

b) Nested loops

i) Dependent loops

ii) Independent loops.

1) Mutually Exclusive loops

```
Algo AJ(n)
{
    a=0
    for(i=1; i<=n; i++)
    {
        a=a+2
    }
    for(j=1; j<=√n; j++)
    {
        print(j)
    }
}
```

$O(n)$

$O(\sqrt{n})$

overall TC

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

$= \underline{\underline{O(n)}}$

Nested loops: loop within/inside another loop

```
Algo AJ(n, m)
{
  for(i=1; i<=n; i++)          → O(n)
  {
    for(j=1; j<=m; j++)        → O(m)
    {
      print("AJ Sir Algo")
    }
  }
}
```

overall TC

$= O(n * m)$

Algo AJ(n, m)

eg1)

```
{
  For(i=1; i<=√n; i++)
  {
    For(j=m; j>0; j--)
    {
      print("Hi")
    }
  }
}
```

independent nested loop

$\Rightarrow (1 \longrightarrow \sqrt{n}) : O(\sqrt{n})$

$\longrightarrow (m \longrightarrow 1) : O(m)$

Overall TC

$: O(m * \sqrt{n})$

2) Dependent Nested loops:

dependent

eg:

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

For $i : 1 \longrightarrow n$

Soln :- $\quad j : 1 \longrightarrow i$

		# runs
$i=1$:	$j : 1 \longrightarrow 1 \Rightarrow$	1 ✓
$i=2$:	$j : 1 \longrightarrow 2 \Rightarrow$	2 ✓
$i=3$:	$j : 1 \longrightarrow 3 \Rightarrow$	3 ✓
		⋮
$i=n$:	$j : 1 \longrightarrow n \Rightarrow$	n ✓

overall TC of inner and outer loop Combined:

$$\Rightarrow 1+2+3+ \cdots (n-1)+n$$

$$= \sum_{i=1}^{n} i = \boxed{\frac{n(n+1)}{2}} = \frac{n^2+n}{2} \Longrightarrow O(n^2)$$

eg:

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

dependent

$i: 1 \longrightarrow n$

$j: i \longrightarrow n$

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

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

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

$\vdots$

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

Overall TC

$$\Rightarrow n+(n-1)+(n-2) \cdots 2+1$$

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

Question

#Q. Consider the following code → independent nested loop

```
i = n;
while (i > 0)          → O(log_2 n)
{
  j = 1;
  while (j ≤ n)        → O(log_2 n)
  {
    j = 2 * j;
  }
    i = i/2;
}
```

Time complexity of above code in terms of Big-Oh?

A. $(\log n)^2$
B. $\sqrt{\log n}$
C. $n \log n$
D. $\log\log n$

Ans: A

$\log(\log n) + (\log n)^2$

overall TC

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

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

$$i = n \longrightarrow n/2 \longrightarrow n/2^2 \longrightarrow 1$$

$$\Rightarrow \underline{\underline{\log_2 n}}$$

$$j = 1 \longrightarrow 2 \longrightarrow 2^2 \cdots\cdots 2^k$$

$$n$$

$$2^k = n$$

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

Question

#Q. Consider the following code:

```
main()
 i = 1;
 while (i ≤ n)
 {
    i = i * 2;
 }
```

$1 \rightarrow 2 \rightarrow 2^2 \rightarrow 2^3 \cdots \rightarrow 2^k$

$2^k = n$

$\underline{k = \log_2 n}$

What is the time complexity of above code?

A $\theta(n)$

B $\theta\sqrt{n}$

C $\theta(\log n)$

D None

Ans : C

Question

∞ loop

#Q. What is the time complexity of the following code ?

```
for (a = 0; a <= n ; a =a*2)
{
    for (b = 0; b < 100 ; b = b +2)
    {
        for (c = 1; c < 8*n; c ++)
        {
            print("AJ Sir")
        }
    }
}
```

$\Rightarrow O(\log n)$

$\Rightarrow O(1)$

$\Rightarrow O(n)$

$\log n \times c * n$

$= \underline{O(n \log n)}$

When

for (a = 1; a <= n; a = a * 2)

A $O(n^3)$

B $O(n^2)$

C $O(\log n)$

D None of These (∞)

for $(\underline{a=0}; \underline{\underline{a<=n}}; a=a*2)$ $\longrightarrow$ <u>Infinite loop</u>

$a=0 \longrightarrow 0\times 2 \longrightarrow 0\times 2^2 \longrightarrow 0\times 2^3 \ldots\ldots \infty$

Question

#Q. Consider the following function:

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

The return value of the function is

i: $i = n/2 \longrightarrow n;\ i{+}{+} \Rightarrow O(n/2) = O(n)$

j: $j = 2 \longrightarrow n;\ * 2 \Rightarrow O(\log_2 n)$

$(k = k + n/2)$

A) $\theta(n^2)$

B) $\theta(n^3)$

C) $\theta(n^2 \log n)$

D) $\theta(n^3 \log n)$

Ans: C

(Q1) what is the TC of the code?

$\Rightarrow O(n * \log_2 n)$

(Q2) what is the value returned by the Algo?

each time $\to$ $\underline{k = k + n/2}$

no. of times $= \underline{(n \log n)}$

k at the end $= \underline{(n/2 * n \log n)}$

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

Question

#Q. Consider the following code.

```
main()
{
    i=1;
    while(i <= n)
    {
        i=10*i;
    }
}
```

$i: 1 \to 10 \to 10^2 \to 10^3 \cdots - 10^k$

$10^k = n$

$k = (\log_{10} n)$

What is the highest asymptotic worst case time complexity of above code fragment?

A $O(n^2)$ ✗

B $O(\sqrt{n})$ ✗

C $O(n)$ ✗

D $O(\log n)$ ✓

Ans D

Recursive Code

1) Recurrence relation (TC)

2) Solve TC Recurrence

3) TC ⇒ Notation

Solving recurrence

1. Back substitution
   - → value of recurrence as well
2. Master Method
   - → Shortcut
   - → Specific cases (Symm Rel)
3. Recursion tree
   - → Asymm Recur

Topic : Time Complexity Framework for Recursive Algorithms

```
1. Algorithm AJ(n)            → TC. T(n)
   {
         if (n = 1) return;   → O(1)
         else
         {
               AJ(n – 1);     → T(n-1)
               [redacted]
         }
   }
```

Ans: TC of AJ(n)
$= O(n)$

Step 1 :- TC Recurrence

$$T(n) = T(n-1) + C, \quad n \geq 1$$

$$T(n) = b, \quad n = 1 \longrightarrow \text{Terminating / Base condition}$$

Step-2: Solve TC Recurrence using Back-sub.

$$T(n) = T(n-1) + C \quad \text{— } ①$$

$$T(n-1) = \left[T(n-2) + C\right]$$

$$T(n) = T(n-2) + C + C$$

$$T(n) = T(n-2) + 2C \quad \text{— } ②$$

$$T(n-2) = T(n-3) + C$$

$$T(n) = T(n-3) + 3C \quad \text{— } ③$$

In general

$$T(n) = T(n-k) + k * C \quad \text{— } ④$$

$$T(n) = \underbrace{T(n-k)}_{} + k * c \quad \text{—— (4)}$$

For Base Condition, $\underline{(T(1))}$

$$n-k = 1$$

$$\boxed{k = n-1}$$

Put this in eqn (4)

$$T(n) = T(1) + (n-1) * c$$

$$\boxed{T(n) = b + (n-1) * c}$$

$\longrightarrow$ value of Recurrence

$$T(n) = b + nc - c$$

$$\boxed{T(n) = \Theta(n)}$$

(Q) TC of below code?

```
Algo AJ(n)            → T(n)
{
    if(n==1)
        return 1      } O(1) = b
    else
        return AJ(n/5)   → T(n/5)
}
```

$$TC = \Theta(\log_5 n)$$

Step1 : TC Recurrence

$$T(n) = T(n/5) + c, \quad n \geq 1$$

$$T(n) = b, \quad n = 1$$

Step 2 : Solve

$$T(n) = T(n/5) + c \quad \text{--- (1)}$$

$$T(n/5) = T(n/5^2) + c$$

$$T(n) = T(n/5^2) + 2c \quad \text{--- (2)}$$

$$T(n/5^2) = T(n/5^3) + c$$

$$T(n) = T(n/5^3) + 3c \quad \text{--- (3)}$$

General,

$$T(n) = \underline{T(n/5^k)} + k * c \quad (4)$$

for Base Condition, $T(1)$

$$n/5^k = 1$$

$$5^k = n$$

$$\boxed{k = \log_5 n}$$

$$T(n) = T(1) + c * \log_5 n$$

$$\boxed{T(n) = b + c * \log_5 n}$$

$$\boxed{T(n) = O(\log_5 n)}$$

7. The given diagram represents the flowchart of recursive algorithm A(n). Assume that all statement except for the recursive calls have order (1) time complexity. Then the best case and worst case time of this algorithm is _____.

V. Imp

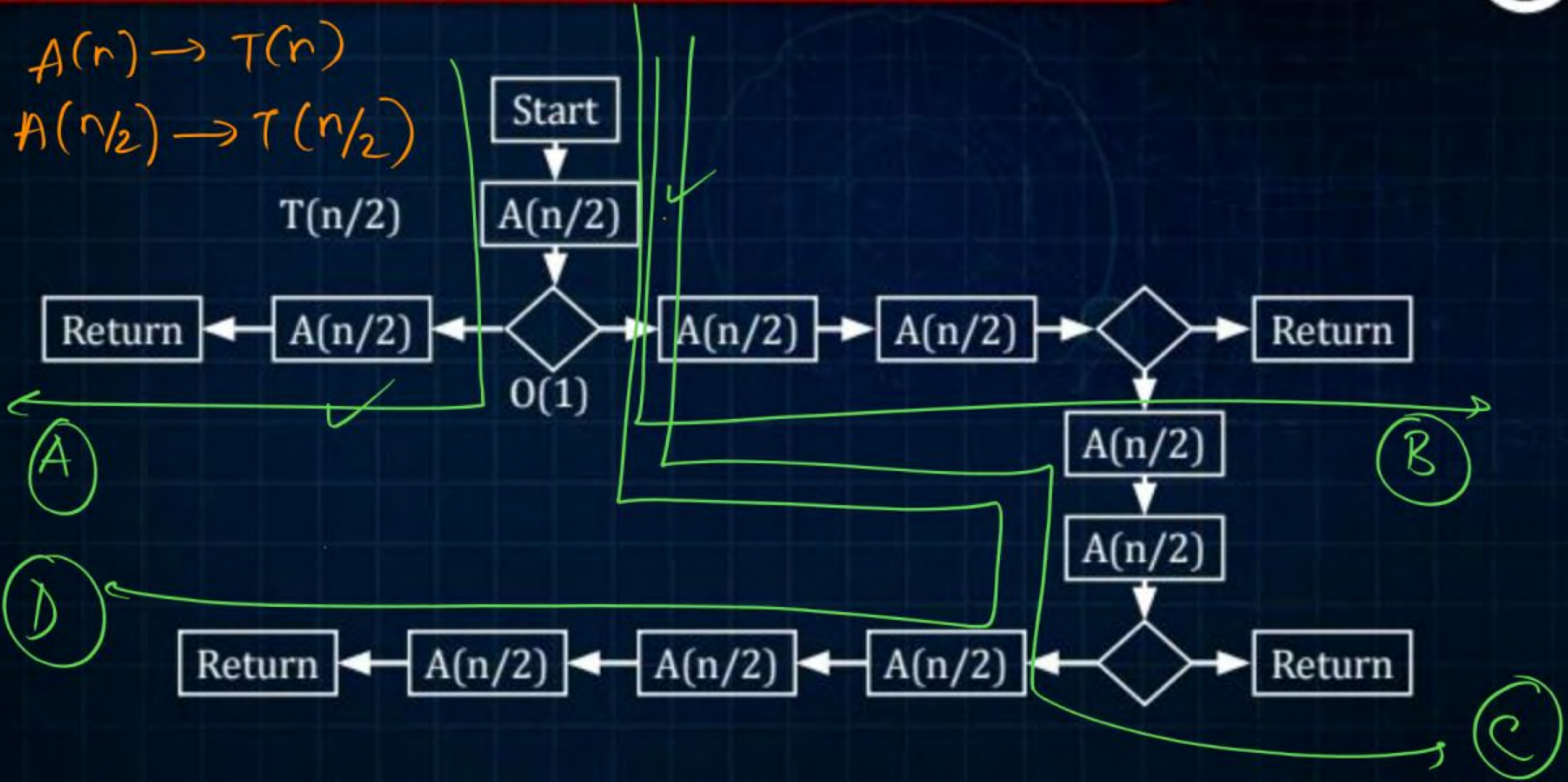
Best Case : least TC / steps

Worst Case : Max TC / steps

Topic : Time Complexity Framework for Recursive Algorithms

$A(n) \rightarrow T(n)$

$A(n/2) \rightarrow T(n/2)$



A) $T(n) = \underline{2}T(n/2) + c$ Best Case

B) $T(n) = \underline{3} * T(n/2) + c$

H.W

C) $T(n) = \underline{5} * T(n/2) + c$

D) $T(n) = \underline{8}T(n/2) + c$ Worst Case

Question

#Q. Consider the following recurrence relation

$$T(n) = 9T\left(\frac{n}{3}\right) + C$$

What is the time complexity of above recurrence relation?

A $\theta(n^3)$

B $\theta(n^2)$

C $\theta(n^2 \log n)$

D $\theta(\log n)$

Soln :-

$$T(n) = 9T(n/3) + C \quad \text{—} \quad ①$$

$$T(n/3) = \underline{9T(n/3^2) + C}$$

$$T(n) = 9\left[9T(n/3^2) + C\right] + C$$

$$T(n) = 9^2 T(n/3^2) + 9C + C \quad \text{—} \quad ②$$

$$T(n) = 9^2 T(n/3^2) + 9c + c$$

$$9^0 = 1$$

$$T(n/3^2) = 9T(n/3^3) + c$$

$$T(n) = 9^3 T(n/3^3) + 9^2 c + 9c + c \quad \text{— } ③$$

general

$$T(n) = 9^k T(n/3^k) + \left(9^{k-1} + 9^{k-2} \cdots\cdots 9^0\right) c \quad \text{— } ④$$

$$9^0 + 9^1 \cdots \quad 9^{k-1} = \frac{(9^k - 1)}{8}$$

$$\underline{\underline{GP}} : \frac{a(r^n - 1)}{r - 1} = \frac{1 \times (9^k - 1)}{9 - 1} = \frac{9^k - 1}{8}$$

general Term

$$T(n) = 9^k T(n/3^k) + \left(\frac{9^k - 1}{8}\right) * c \quad \text{—— } 5$$

For Base Conditn $T(1)$

$n/3^k = 1$

$3^k = n$

$9^k = n^2$

$$T(n) = n^2 T(1) + \frac{(n^2 - 1)}{8} * c$$

$$T(n) = n^2 * b + \frac{(n^2 - 1)}{8} * c \longrightarrow \underline{\Theta(n^2)}$$

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.

Space Complexity ⟶ Auxilary / Additional Space

a) Iterative Algo ⟶ variables / arrays, etc

b) Recursive Algo ⟶ Recursion Stack

6. Algorithm AJ(n)

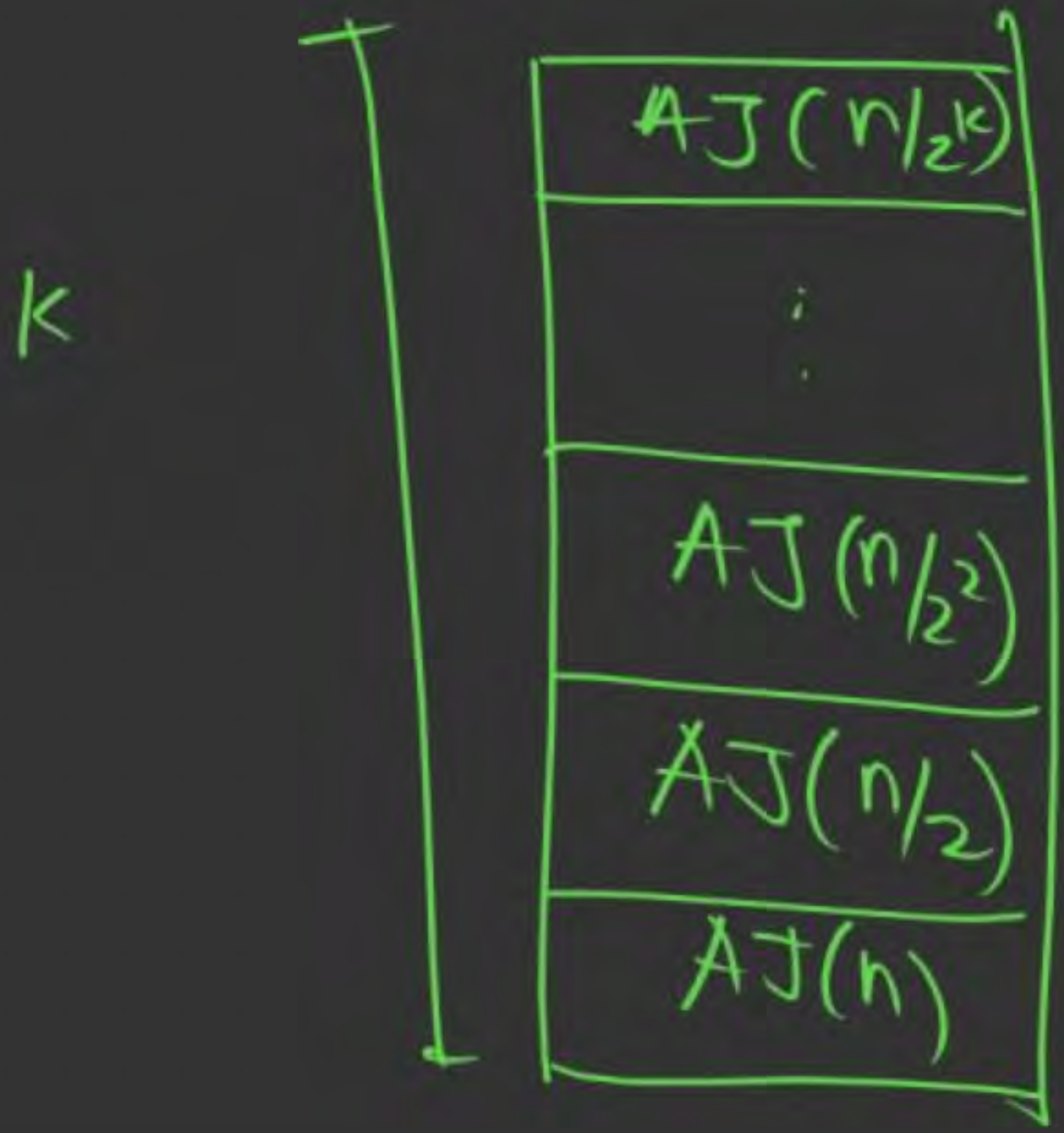
```
{
    if(n = 1) return;
    else
    {
        AJ(n/2);
    }
}
```

What is the Space Complexity of AJ(n)?

Recursion → Recursion Stack

$$\boxed{SC: O(\log_2 n)}$$

Recursion Stack



Height of Recursion Stack = K

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

$$2^K = n \qquad \boxed{K = \log_2 n}$$

Question

#Q. Consider the following code

```
i = n;
while (i > 0)
{
        for(j = 1; j<=i; j = j+3)
        {
            print("AJ Sir Algo")
    }
            i = i - 1
}
```

dependent

H.W

Post on Telegram group "Aditya Jain Sir"

Time complexity of above code in terms of Big-Oh?

A $(\log n)^2$

B $\sqrt{\log n}$

C n*n

D loglogn

Question

#Q. Consider the following C-code

```
void foo (int n)
{
    for (a=1; a <= n; a=a*5)
        {
        for(b = n; b>0; b = b/3)
        {
            printf("AJ Sir Algo");
        }
        }
}
```

$a : 1 \longrightarrow n \quad *5 \longrightarrow (\log_5 n)$

$b : n \longrightarrow 1 \quad /3 \longrightarrow \log_3 n$

Overall $O(\log_5 n * \log_3 n)$

What is the worst time complexity of above program?

A O(1)

B O(n)

C O(logn*log n)

D O$\sqrt{n}$

Question

#Q. Consider the following asymptotic functions:

$f_1 = 2^n$

$f_2 = 1.001^n \longrightarrow$ incr

$f_3 = e^n$

$f_4 = 200$

$f_5 = (0.8)^n \longrightarrow$ decr

Which of the following is correct increasing order of above functions?

$2 < 2^2 < 2^3$

$\frac{1}{2} > \frac{1}{2^2} > \frac{1}{2^3}$

$[0.8^n < 200 < 1.001^n < 2^n < e^n]$

(A) f_4, f_5, f_2, f_1, f_3 ✗

(B) f_2, f_4, f_5, f_1, f_3 ✗

(C) f_5, f_4, f_2, f_1, f_3 ✓

(D) f_5, f_2, f_1, f_3, f_4

$a^n \longrightarrow$ incr $\quad a > 1$

$a^n \longrightarrow$ decr $\quad a < 1$

$a^n \longrightarrow$ const, $\quad a = 1$

$e^n \qquad 2^n$

$e > 2$

$\underline{e^n > 2^n}$

Question

#Q. Arrange following function in the descending order growth rate.

$f_1 = (e)^n, f_2 = \sqrt{n}^{\log n}, f_3 = (2)^n, f_4 = (\log n)^n, f_5 = (n)^{\log n}$

(A) f_2, f_5, f_3, f_1, f_4

(B) f_3, f_4, f_2, f_5, f_1

(C) f_2, f_5, f_1, f_3, f_4

(D) f_4, f_1, f_3, f_5, f_2

f_4 f_5

$f_1 = e^n$

$f_2 = (\sqrt{n})^{\log n}$

$f_3 = 2^n$

$f_4 = (\log n)^n$

$f_5 = n^{\log n}$

$e^n > 2^n$

$f_1 \quad f_3$

$(\log n)^n > n^{\log n}$

$\underline{n \log(\log n)} \quad (\log n)^2$

Question

which

#Q. ~~How many~~ of the following statements is/are True?

MSQ

A $10\sqrt{n} + \log n = O(n)$

B $\sqrt{n} + \log n = O(\log n)$

C $\sqrt{n} + \log n = \theta(n)$

D $\sqrt{n} + \log n = \theta(\sqrt{n})$

$10\sqrt{n} + \log n = \theta(\sqrt{n}) = O(\sqrt{n})$

$= O(n)$

$= O(n^2)$

A, D

Question

MSQ

#Q. Consider two function $f(n) = 10n + 2\log n$ and $g(n) = 2(\log(n^3)) + 5n$, then which of the following is correct option?

(A) $f(n) = \theta(g(n))$ ✓

(B) $f(n) = O(g(n))$ ✓

(C) $g(n) = O(f(n))$ ✓

(D) $g(n) = O(\log n)$ ✗

Conclusion: $f(n) = \theta(n)$
$g(n) = \theta(n)$

$f = \theta(g(n))$

$O(g(n))$ $\Omega(g(n))$

A, B, C

$$f(n) = 10n + 2\log n \longrightarrow \theta(n)$$

$$g(n) = 2(\log(n^3)) + 5n$$

$$= 2(3 + \log n) + 5n$$

$$= 6\log(n) + 5n \longrightarrow \theta(n)$$

Question

#Q. Suppose,

$$f(n)=\sum_{i=1}^{n}=O\left(n^2\right), g(n)=\sum_{i=1}^{n^2}=O(n)$$

Which of the following is/are corrected?

$f = \theta(n^3)$

$g = \theta(n^3)$

(A) $f(n) = \theta(g(n))$

A, B, C

(B) $f(n) = O(g(n))$

(C) $f(n) = \Omega(f(n))$ → True

$f = \Omega(f)$

$f \geq c * f$

D) $f = \omega(f)$

$f < c * f$ ✗

(D) $f(n) = \omega(f(n))$ → False

$$f(n) = \sum_{i=1}^{n} O(n^2)$$

$$= \sum_{i=1}^{n} c_1 * n^2$$

$$= c_1 n^2 + c_1 n^2 \cdots c_1 n^2$$

(n time)

$$= n * c_1 n^2 = \Theta(n^3)$$

$$g(n) = \sum_{i=1}^{n^2} O(n)$$

$$= \sum_{i=1}^{n^2} c_2 * n$$

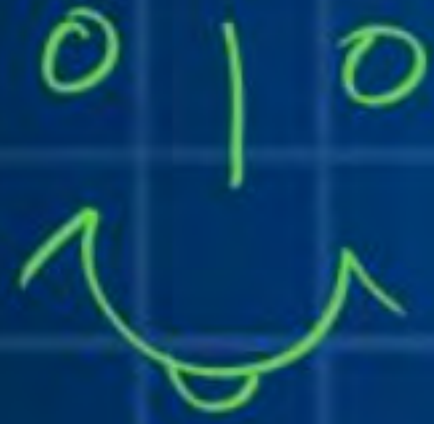
$$= \underbrace{c_2 * n + c_2 * n \cdots c_2 * n}_{n^2 \text{ times}}$$

$$= n^2 * c_2 * n$$

$$= c_2 * n^3 = \Theta(n^3)$$

Thank

THANK



Keep Hustling!