GATE

CRASH COURSE

CS & IT

Algorithms

Analysis of Algorithms

By - Aditya Sir

Topics to be covered

1. Types of Analysis
2. Asymptotic Notations

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
    2. Analysing Non-Recursive Algorithms
    3. Analysing Loops
    4. Analysing Recursive Algorithms
    5. Space Complexity
    6. Problem Solving

Nagpur $\longrightarrow$ Noida

time money

1) Flight less high

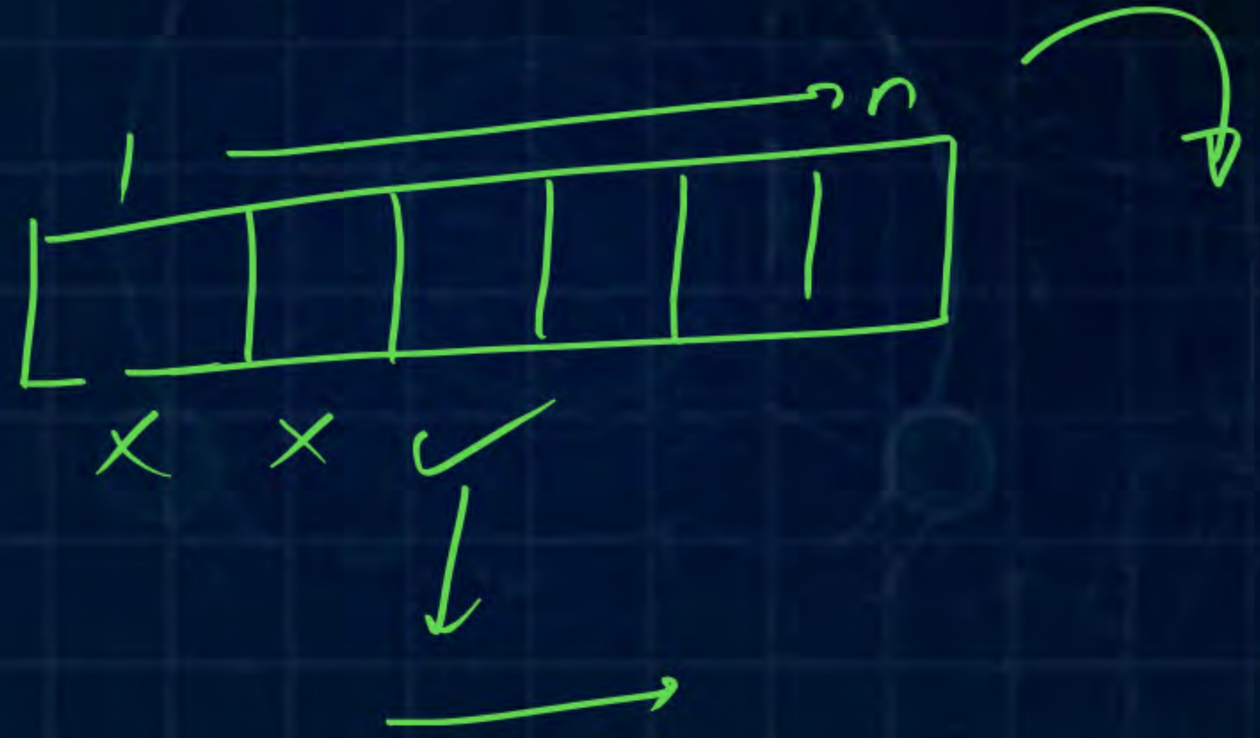
2) Train

3) Car

4) Cycle High least

Linear Search

```
Algorithm LS(A, n, x)
integer n, A[n]
{
      int i;
      for i ← 1 to n
{
      if (x = A[i])
      {
      print (i);
      Exit;
      }
{
      Print(element not found);
}
```



Linear Search

eg1) A = | 2 | 7 | 3 | 10 | 5 | $x = 2$

Best Case i/p: 1 Comp → Best Case TC

eg2) A = | 2 | 7 | 3 | 10 | 5 | (1 → n)

$x = 5$, $x = 25$

Worst Case i/p: n Comp → Worst Case TC

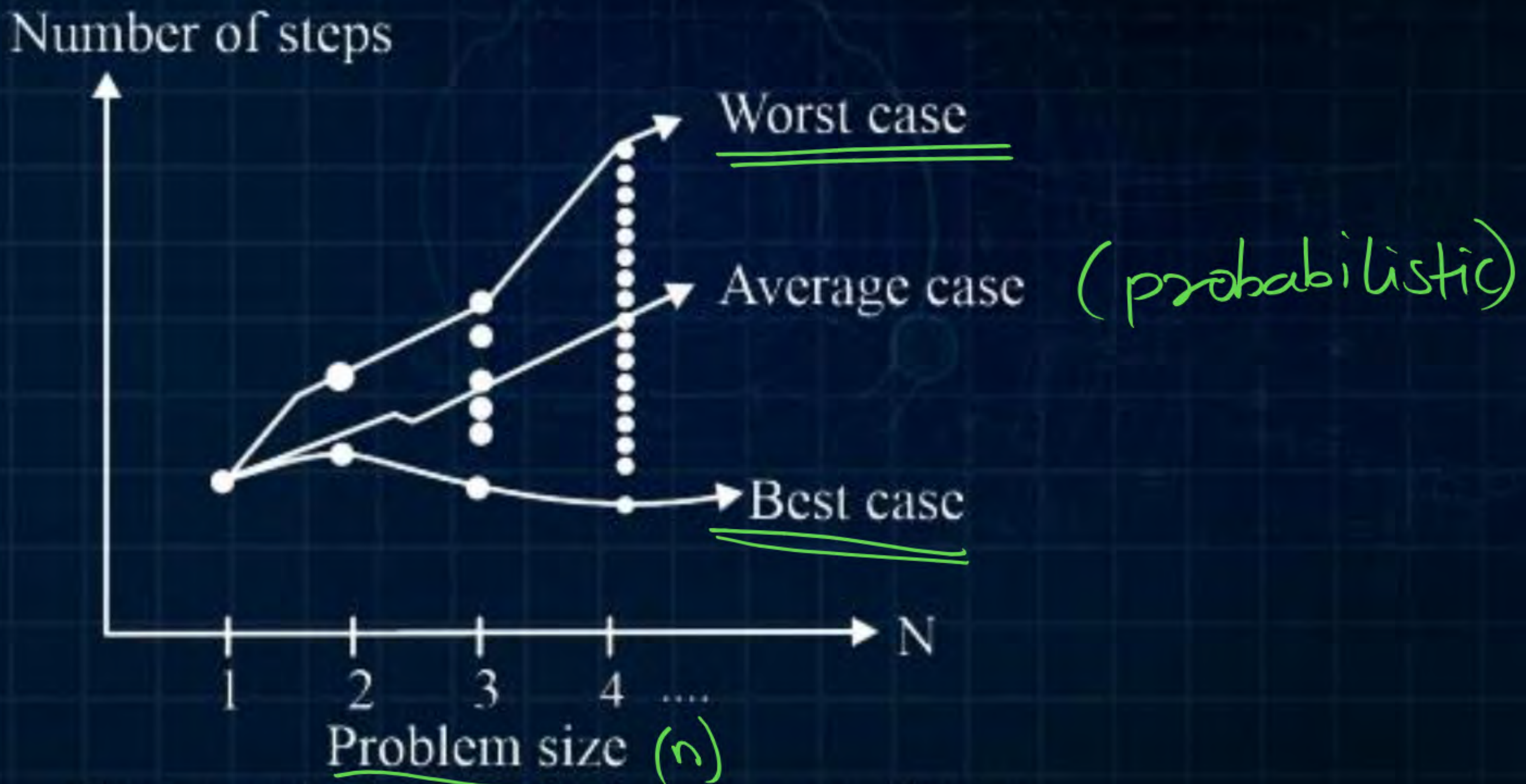
Topic : Analysis of Algorithms

Linear search A(n)

1. Best Case : 1 : ~~O(1)~~ $\Omega(1)$
2. Worst case : n : O(n)
3. Average case: (for a successful linear search)

   $\rightarrow O(n)$

Topic : Analysis of Algorithms



Best, worst, and average-case complexity

TC → function of i/p size 'n'

Oh → UB

Omega → LB

Asymptotic Notations

Big Notations

Lose + Tight Bound

1) Big-Oh (O)
2) Big-Omega (Ω)

Small/little Notations

loose Bound

1) Small-oh (o)
2) small omega (ω)

5) Theta (θ) → Tight Bound

Topic : Asymptotic Notations

→ Let ‘f’ and ‘g’ be functions from the set g integers/real to reals number;

1. **Big-Oh(O):** Upper bound

   $f(n)$ is $O(g(n))$ if there exists some constant $c > 0$ and $n_0 > 0$ such that $f(n) \le c.g(n)$, whenever $n \ge n_0$

$$\boxed{f(n) \le c * g(n)}$$

for some $c > 0$

when $n \ge n_0$

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

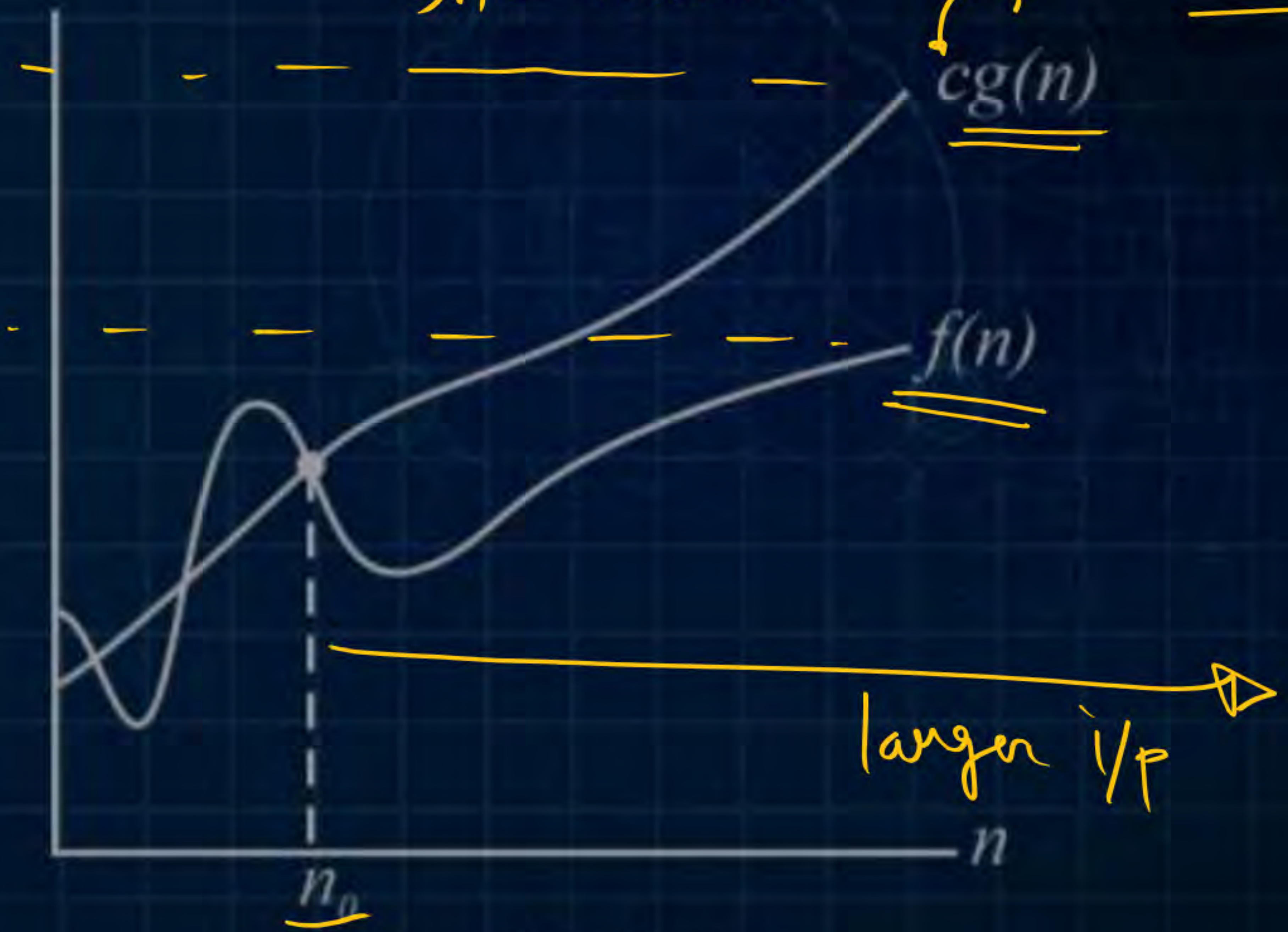
Topic : Asymptotic Notations

$2^n \to$ Expo
$3n \to$ linear

asymptotic:
$2^n > 3n$
A

$2^n < 3n$
$n = 1$
$2^1 = 2$ $\qquad$ $3 \times 1 = 3$

$n \to$ large



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

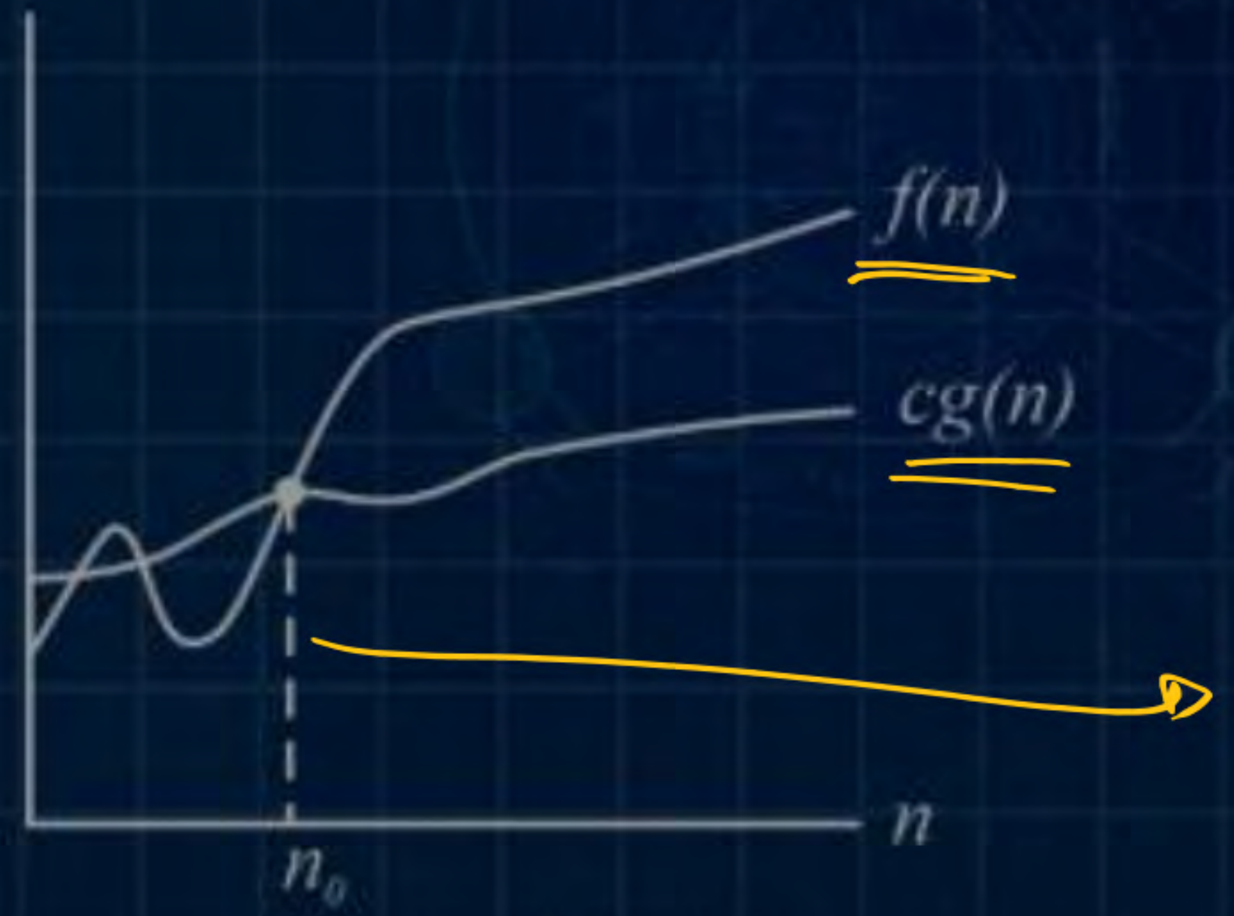
2. **Big-Omega(Ω):** Lower Bound

f(n) in Ω(g(n)) iff there exists ~~consist~~ a +ve constant 'c' and 'n_0' such that

$f(n) \geq c.g(n)$, whenever $n \geq n_0$

$$\boxed{f(n) \geq c * g(n)}$$

Topic : Asymptotic Notations



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

3. **Theta(θ):** Tight Bound:

f(n) is θ(g(n) iff f(n) is O(g(n)) and f(n) is Ω(g(n))

$C_1.g(n) \leq f(n) \leq C_2.g(n)$

and

$f(n) = O(g(n)) \longrightarrow f(n) \leq c_1 * g(n)$

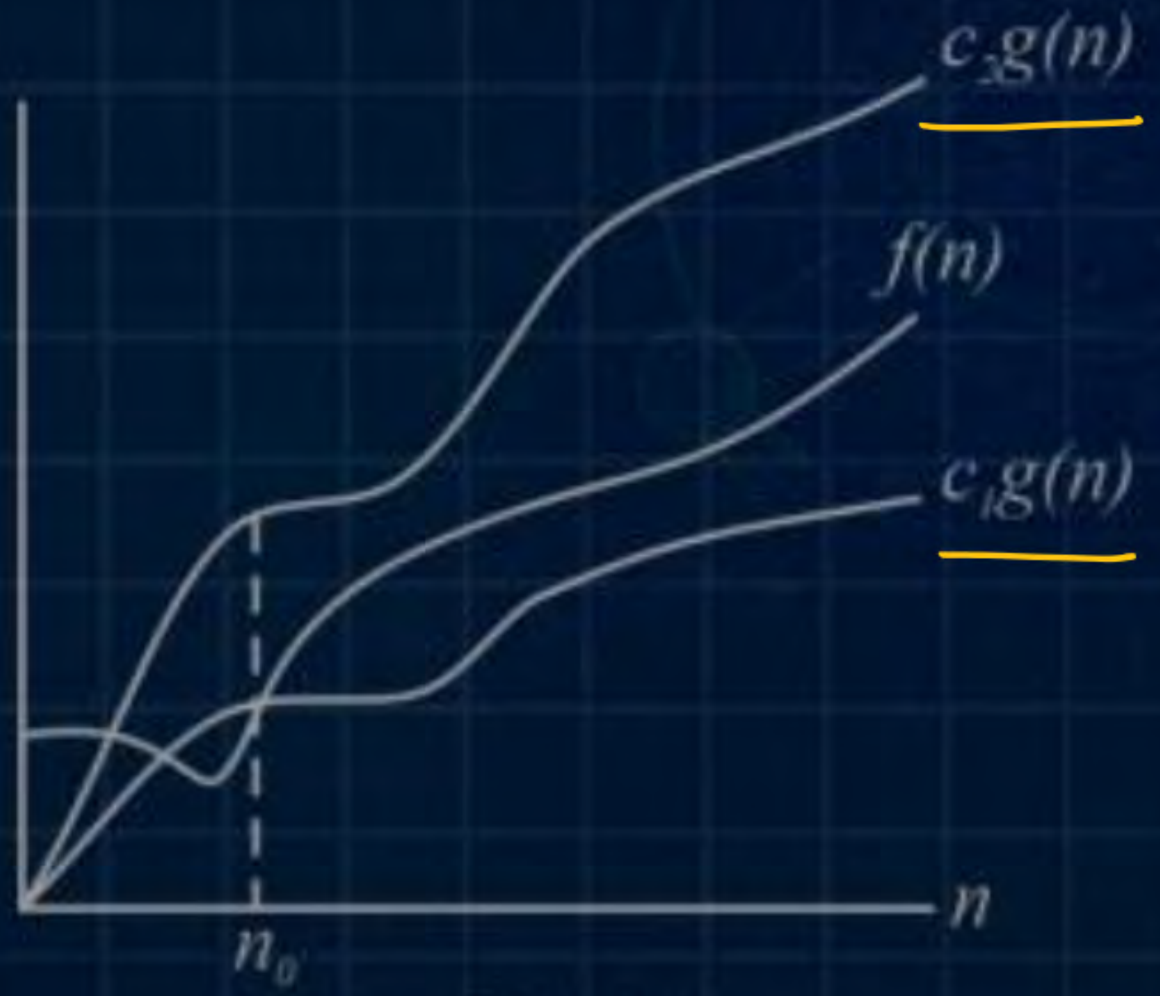
$f(n) = \Omega($ $\longrightarrow f(n) \geq c_2 * g(n)$

$\longrightarrow f(n) = \Theta(g(n))$

$c_2 * g($

$f(n) \leq c_1 * g(n)$

Topic : Asymptotic Notations



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

$$\boxed{n^2+n+10 = \Theta(n^2)}$$

eg:- $f(n) = n^2 + n + 10$

$$n^2+n+10 \leq 100 * n^2$$

$$f(n) \leq c_1 * g(n)$$

$$n^2+n+10 = O(n^2)$$

$$n^2+n+10 \geq 1 * n^2$$

$$f(n) \geq c_2 * g(n)$$

$$f(n) = \Omega(n^2)$$

Topic : Asymptotic Notations

loose

(1) Small – oh (o) : Proper upper Bound

f(n) is o (g (n)) iff for all c > 0.

$f(n) < c.\, g(n)$, whenever $(n > n_0)$, $n_0 > 0$.

$$\boxed{f(n) < c \times g(n)}$$

loose

(2) Small omega (ω) : Proper lower Bound

f(n) is ω (g (n)) iff for all c > 0.

f(n) > c. g (n) , whenever (n > no), no > 0.

$$\boxed{f(n) > c * g(n)}$$

$$f(n) = 3n + 5$$

UB

$3n+5 \leq 10 * n$

$3n+5 = \underline{O(n)}$

Loose UB

$$\left.\begin{aligned} &= O(n^2) \\ &= O(n^3) \\ &\ \vdots \end{aligned}\right.$$

tight Bound

$\underline{\underline{\Theta(n)}}$

LB

$3n+5 \geq 1 * n$

$3n+5 = \underline{\Omega(n)}$

$3n+5 = \Omega(\sqrt{n})$

$3n+5 = \Omega(1)$

Loose LB

Topic : Discrete Properties of ASN

Properties \ ASN	O	Ω	Θ	o	ω
Reflexive	✓	✓	✓	✗	✗
Symmetric	✗	✗	✓	✗	✗
Transitive	✓	✓	✓	✓	✓
Transpose Symmetry	If f(n) is O(g(n)) Then g(n)	is Ω (f(n))	✗	If f(n) is o(g(n)) Then g(n)	is ω (f(n))

if $f(n) \le c * g(n)$

then $g(n) \ge c * f(n)$

$f > g$

$g < f$

Topic : Exponentials

For all real a> 0, m, n

$a^0 = 1$

$a^1 = a$

$a^{-1} = 1/a$

$(a^m)^n = a^{mn}$

$(a^m)^n = (a^n)^m$

$a^m * a^n = a^{m+n}$

$$\boxed{(a^m)^n = (a^n)^m = a^{(n*m)}}$$

$$(2^2)^3 = (2^3)^2 = 2^{2\times 3}$$

$$4^3 = 8^2 = 2^6$$

$$= 64 \quad = 64 \quad = 64$$

$$a^m * a^n = a^{(m+n)}$$

Topic : Analysis of Algorithms

$\log X^y = y \log x$

$\log xy = \log x + \log y$

$\log \log n = \log (\log n)$

$a^{\log_b^x} = x^{\log_b^a}$

$a = b^{\log_b^a}$

$\log_b a^n = n. \log_b a$

$\log_b a = \frac{1}{\log_a b}$

$\log n = \log_{10}^n$

$\log^k n = (\log)^k$

$\log \frac{x}{y} = \log x - \log y$

$\log_b^x = \frac{\log_a^x}{\log_a^b}$

$\log_c (ab) = \log_c a + \log_c b$

$\log_b^{1/a} =- \log_b a$

$a^{\log_b c} = c^{\log_b a}$

$f(n) \cdot 3n+5 \qquad g(n) = 4n+2$

Asymptotic

1) $f = O(g) \Longrightarrow f \leq_A g$

2) $f = \Omega(g) \Longrightarrow f \geq_A g$

3) $f = \Theta(g) \Longrightarrow f =_A g$

4) $f = o(g) \Longrightarrow f <_A g$

5) $f = \omega(g) \Longrightarrow f >_A g$

$3n < 5n^2$

$5n^2 > 3n$

Mathematically

$3n+5 \neq 4n+2$

$3n+5 \longrightarrow \Theta(n)$

$4n+2 \longrightarrow \Theta(n)$

Question

#Q. Consider the following functions $f(n) = n.2^n$ and $g(n) = 4^n$ then which of the following is correct?

$f \underset{A}{\leq} g$

$f \underset{A}{\leq} g$

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

$f \underset{A}{\geq} g$

(B) $f(n) = \Omega(g(n))$

$f \overset{A}{=} g$

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

(D) None of these

$f(n) = n * 2^n$

f	g
$n * 2^n$	$(2^n)^2$
$n \times \cancel{2^n}$	$2^n * \cancel{2^n}$
n <	2^n

$g(n) = 4^n$

$\downarrow$

$= (2^2)^n$

$g(n) = (2^n)^2$

$\frac{1}{2} > \frac{1}{4} > \frac{1}{8} > \frac{1}{10}$

Rate of Growth

In general:

(Decreasing fn < Constant < Logarithmic < Polynomial < Exponential)

eg:- $\frac{1}{n} < 10 < \log(n) < \sqrt{n} < n < n^2 \dots < 2^n < 3^n < 4^n \dots < n^n$

Question

#Q. Consider the following functions:

$f_1 = 2^{2^n}$

$f_2 = n!$

$f_3 = 4^n$

$f_4 = 2^n$

What is the correct increasing order of above functions?

A. $f_1\ f_4\ f_3\ f_2$

Ans: D

B. $f_4\ f_2\ f_3\ f_1$

C. $f_1\ f_2\ f_3\ f_4$

D. $f_4\ f_3\ f_2\ f_1$

$n! < n^n$

Soln :-

$n! = O(n^n)$

$f_1 = 2^{2^n}$

$f_2 = n! = n \times (n-1) \times (n-2) \cdots 1$

$\approx n^n$

$f_3 = 4^n$

$f_4 = 2^n$

$2^n < 4^n < n! < 2^{2^n}$

$f_4 < f_3 < f_2 < f_1$

2^{2^n} > vs n^n

Take $\log_2()$ both sides

$\log_2(2^{2^n})$

$\log_2(n^n)$

$\underline{2^n}$ > $\underline{n \log_2 n}$

Question

#Q. Consider the following functions from positive integers to real number:

$f_1(n) = 2^{100}$

$f_2(n) = n$

$f_3(n) = n\log_2 n$

$f_4(n) = \frac{2^{100}}{n}$

The correct arrangement of the above functions in increasing order of asymptotic complexity is:

(A) $f_3\, f_4\, f_1\, f_2$ ✗

(B) $f_4\, f_1\, f_2\, f_3$ ✓

(C) $f_1\, f_4\, f_2\, f_3$ ✗

(D) $f_4\, f_1\, f_3\, f_2$ ✗

Ans: B

$\frac{1}{n} < 1$

$f_1 = 2^{100} \longrightarrow$ Const

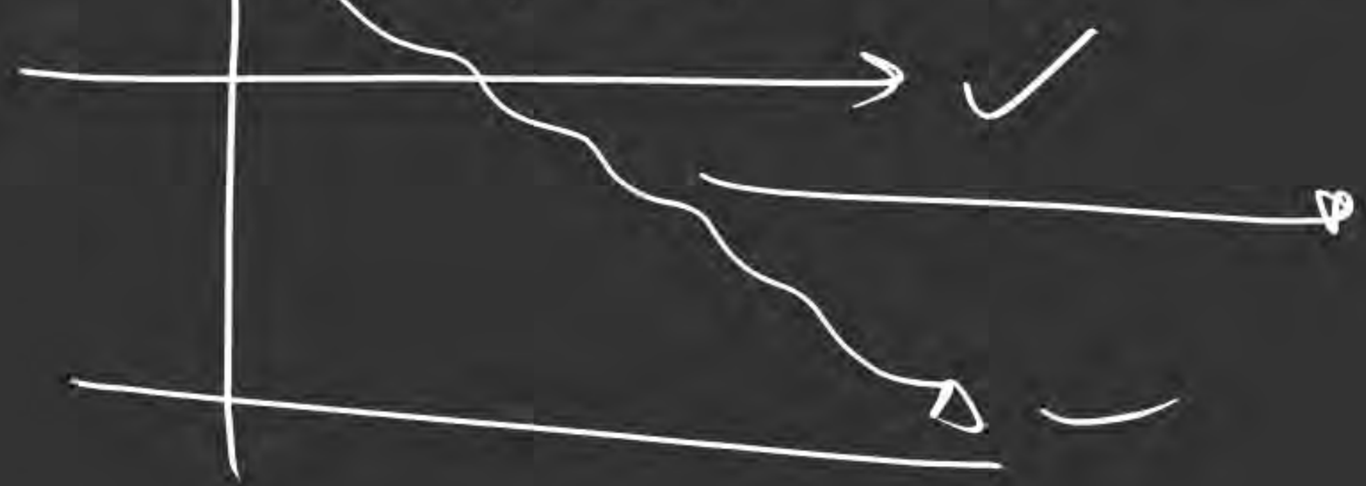
$f_2 = n \longrightarrow$ Linear

$f_3 = n\log_2 n \longrightarrow$ Polylog

$f_4 = \dfrac{2^{100}}{n} \longrightarrow$ Decr

$\dfrac{2^{100}}{n} < 2^{100} < n < n\log_2 n$

$4 < 1 < 2 < 3$



$f_4 < f_1$

$\cancel{n}$

$\cancel{n}\log_2 n$

$1 < \log n$

Question

$n = 64$ $\quad \log_2 n < \sqrt{n}$ $\quad \log_2 64 = 6$ $\quad \sqrt{64} = 8$

#Q. Consider the following functions from positive integers to real number:

$10, \sqrt{n}, n, \log_2 n, \frac{100}{n}$

The correct arrangement of the above functions in increasing order of asymptotic complexity is:

(A) $\log_2 n, \frac{100}{n}, 10, \sqrt{n}, n$ ✗

(B) $10, \frac{100}{n}, \sqrt{n}, \log_2 n, n$ ✗

(C) $\frac{100}{n}, 10, \log_2 n, \sqrt{n}, n$ ✓

(D) $\frac{100}{n}, \log_2 n, 10, \sqrt{n}, n$ ✗

Ans: C

10 → Const

$\sqrt{n}$ → Poly

n → Poly

$\log n$ → log

$\frac{100}{n}$ → Decr

Loop Complexities

1> For loop
2> while loop

Iterative / non-Recursive

3> Nested loop

Loop : TC → no. of time loop runs and TC of code within loop

$= n * O(AJ(n))$

```
For( i=1 ; i<=n ; i++ )
{
    a=a+1
    AJ(n)
}
```

→ runs n times $= O(n)$

$a=a+1$ → $O(1)$

$\boxed{AJ(n)}$

loop : $O(n)$

1)

```
a = 0
for (i = 1; i <= n ; i++)   ⟶ n times
{
    a = a + 5               ⟶ O(1)
}
```

$$TC = \underline{\underline{O(n)}} \quad \underbrace{O(1) + O(1) - - - - O(1)}_{n \text{ times}}$$

$a = 1$

2>

```
For (i = n; i > 0 ; i--)
{
    a = a * 2
}
```

$$i = n \downarrow n-1 \downarrow n-2 \;\vdots\; 0$$

$TC = O(n)$

$a = 100$

3> for (i=1 ; i <= $\sqrt{n}$; i++)

{

a = a * 0 → const time

}

TC : $O(\sqrt{n})$

4)

```
aj = 0
for(i=1; i<=n; i++)
{
    aj = aj + i
}
return aj
```

TC : $\underline{O(n)}$

Value Returned is $O(\underline{n^2})$

$a_j = 0$

$i = 1 \longrightarrow a_j = 0 + 1 = 1$

$i = 2 \longrightarrow a_j = 1 + 2$

$i = 3 \longrightarrow a_j = (1+2) + 3$

$\vdots$

$i = n \longrightarrow a_j = (1 + 2 \ldots (n-1)) + n$

After loop ends

$a_j = 1 + 2 + 3 \ldots (n-1) + n$

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

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

5)

$a=0$

```
for(i=1; i<=√n ; i++)
{
    if(i==3)
        break;
    a=a+2
}
```

TC = ?

$i=1$ ✓

$i=2$ ✓

$i=3$ ✓

exit

$\therefore$ $\Rightarrow TC = O(n)$

6)

```
i = 1
while(i <= n)
{
    print(i)
    i = i+2
}
```

$$\underbrace{i : 1 \to 3 \to 5 \to 7 \cdots n}$$

$\approx n/2$ times

$$O(n/2)$$

$$= O(n) \checkmark$$

7)

```
for (i=1; i<=n; i=i+10)
  {
     print(i)
  }
```

$i = 1 \xrightarrow{+10} 11 \rightarrow 21 \ldots n$

runs $\approx$ $n/10$ times

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

8) For (i=1; i<=n; i=i+20)
{ print(i)
}

$\approx$ runs $n/20$ times

$TC \approx \underline{O(n)}$

★ For (i=1; i<=n; i=i+aj)
{
print(i)
}

i=1
while(i<=n)
{ i=i+10
}

$\rightarrow \approx \underline{O(n)}$

$\approx$ runs = $n/(aj)$

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

9)

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

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

loop ends after k iter.

$2^k = n$

$\log_2(2^k) = \log_2 n$

$\boxed{k = \log_2 n}$

$\boxed{TC = O(\log_2 n)}$

10)

```
For (i=1; i<=√n ; i=i*10)
  { print(i)
  }
```

$i = 1 \rightarrow 10 \rightarrow 10^2 \rightarrow 10^3 \ldots\ldots 10^k$ (k^{th} iter)

Assum loop runs k times

$10^k = \sqrt{n}$

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

$\log(\sqrt{n}) = \log_{10}(n^{1/2})$

$= \frac{1}{2}\log_{10}(n)$

$$\boxed{\begin{aligned} TC &= O(k) \\ &= O(\log_{10}(\sqrt{n})) \\ &= O(\log_{10}(n)) \end{aligned}}$$

```
i=1
while(i<=n)
{
    i=i*20
}
```

$i=1$

$\downarrow$

20

$\downarrow$

20^2

$\downarrow$

20^3

$\downarrow$

20^k

$20^k = n$

$k = \log_{20}(n)$

$TC = O(\log_{20} n)$

genralised

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

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

$TC: O(\log_x(n))$

$n \rightarrow n \times 2 \rightarrow n \times 2^2 \cdots$

```
for(i=n; i>0; i=i*2)
{
    print(i)
}
```

infinite loop

```
for (i=n; i>0; i=i/2)
{
    print(i)
}
```

$32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$

$n \rightarrow n/2 \rightarrow n/2^2 \rightarrow n/2^3 \cdots 1$

$i=1 \rightarrow n, *2$

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

$\log_2 n$

$TC = O(\log_2 n)$

12)

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

$O(n)$ ✗

$O(\sqrt{n})$ ✗

$i * i \le \sqrt{n}$

$(i)^2 \le \sqrt{n}$

$i \le (\sqrt{n})^{1/2}$

$(i \le n^{1/4})$

$\boxed{TC: O(n^{1/4})}$

Nested loops :

↳ loop within a loop

not nested

1) Algo AJ(n)

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

$\rightarrow \theta(n)$

$\rightarrow \theta(\sqrt{n})$

$n^2 + 2n + 100$

$= \theta(n^2)$

Overall TC of AJ(n) : $O(n+\sqrt{n})$

dominating term

$= O(n)$

nested loop

2) for (i=1; i<=n; i++) → n times

{

for (j=1; j<=√n; j++) → √n times

{

print("AJ Sir")

}

}

The no. of times "AJ Sir" gets printed is O(____)?

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

$$= O(n^{1+1/2})$$

$$= O(n^{3/2})$$

Question

H.W

#Q. Consider the following code

```
i = n;
while (i > 0)
{
  j = 1;
  while (j ≤ 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. $\log\log n$

Summary

1) Best Case, Worst Case

2) Asymptotic Notations

$(O, \Omega, \Theta, o, \omega)$

a) Asymptotic Comparison

b) Notation

3) Loop Complexities

Thank

THANK

Keep Hustling!