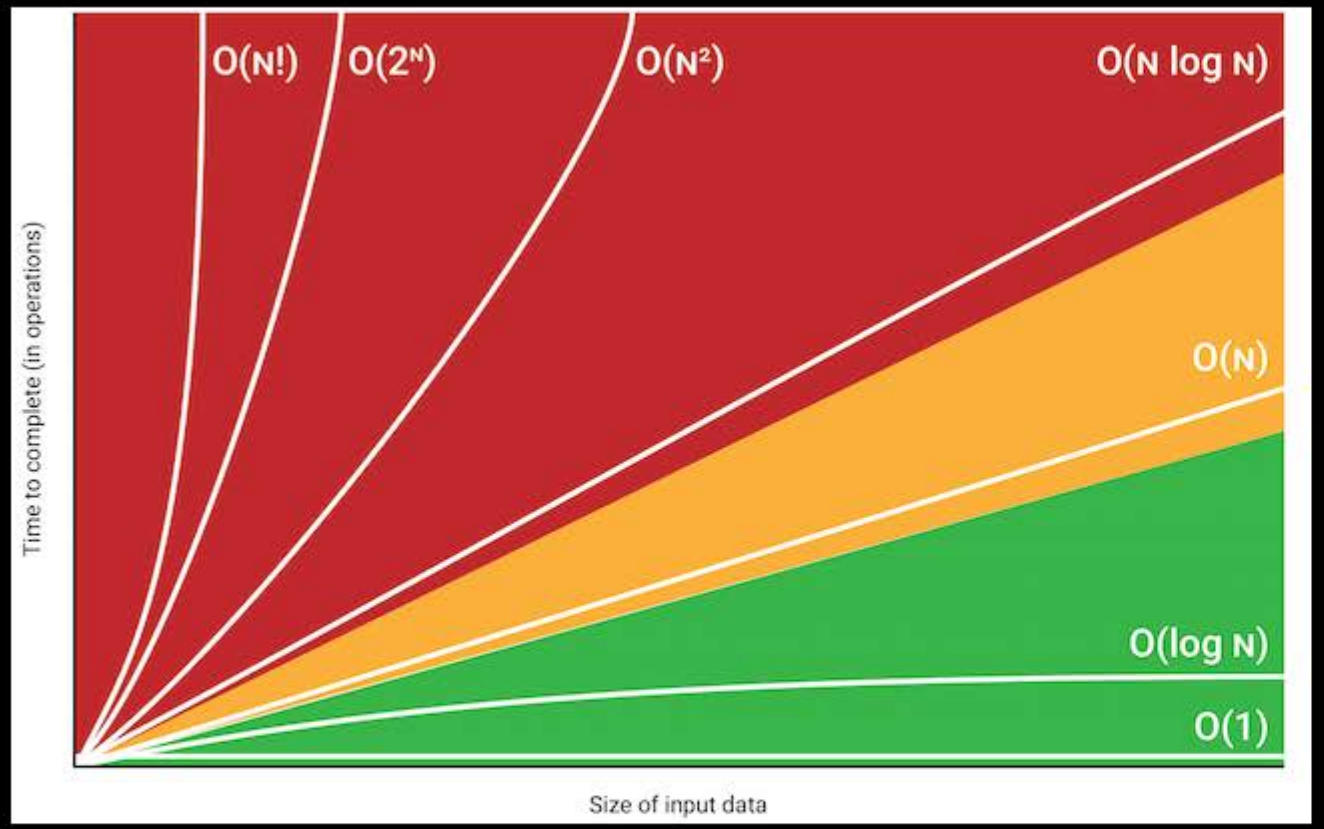


Properties of Asymptotic Notation

Mathematical Background

2pm – 8.30

7am – 1.30

5th

2pm – 8.30

7am - ✠ 1.30

7am - 1.30

4pm

7am – 1.30

Arithmetic Progression

If the initial term of an arithmetic progression is a_1 and the common difference of successive members is d, then the n-*th* term of the sequence is given by

Arithmetic Progression

If the initial term of an arithmetic progression is a_1 and the common difference of successive members is d, then the n-*th* term of the sequence is given by

$$a_n = a_1 + (n-1)d,\ n = 1, 2, \ldots$$

Arithmetic Progression

- The sum S of the first n numbers of an arithmetic progression is given by the formula:

Arithmetic Progression

- The sum S of the first n numbers of an arithmetic progression is given by the formula:
- $S = \frac{n}{2}(a_1 + a_n)$, where a_1 is the first term and a_n the last one.
- $S = \frac{n}{2}(2a_1 + d(n-1))$ ←

Some Analysis Required this calculation

Geometric Progression or GP

The geometric sequence has its sequence formation:

$$a_1, a_1r, a_1r^2, a_1r^3, a_1r^4 \ldots a_1r^{n-1}, a_1r^n$$

Analysis of tree

Recurrence Relation

Sum of Term Geometric Progression

$s_n =$ $r < 1$

$s_n =$ $r > 1$

Sum of Term Geometric Progression

$$s_n = \frac{a(r^n - 1)}{r - 1} \quad where\ r > 1$$

$$s_n = \frac{a(1 - r^n)}{1 - r} \quad where\ r < 1$$

n is No. of terms

a is the first term

Approximate

Sum To Infinity $r < 1$

$s_\infty =$

Sum To Infinity $r < 1$

$$s_\infty = \frac{1}{1-r} \qquad -1 < r < 1$$

Sum the following

- $1 + 2 + 2^2 + 2^3 + 2^4 + \cdots + 2^{40}$
- $1 + \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} + \cdots + \frac{1}{2^{40}}$
- $1 + \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} + \cdots$

The Series

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

$2 + 2^1 + 2^2 + \cdots + 2^n$

$2(2^n - 1)$

$2^{n+1} - 2$

Summation

$\Theta(2^n)$

The Series

$$\sum_{i=1}^{n} 2^i = 2^{n+1} - 2$$

The Series

$$\sum_{i=1}^{n} 2^i = \quad 2^{n+1} - 2$$

The Series

$$\sum_{i=1}^{n} i\,2^i =$$

$$S = 2 + 2\times 2^2 + 3\times 2^3 + \cdots + (n-1)\times 2^{n-1} + n\,2^n = \boxed{\Theta(n2^n)}$$

$$\because \quad 2S = 2^2 + 2\times 2^3 + 3\times 2^4 + \cdots + (n-1)\times 2^n + n\times 2^{n+1}$$

$$(1-2)\times 2^2$$

$$\Rightarrow \quad 2S - S = -2 + 2^2 - 2\times 2^2 + 2\times 2^3 - 3\times 2^3 + \cdots + (n-1)2^n - n\times 2^n + n\times 2^{n+1}$$

$$\Rightarrow \quad S = -2 - 2^2 - 2^3 - 2^4 - \cdots - 2^n + n\,2^{n+1}$$

$$= -(2 + 2^2 + 2^3 + \cdots + 2^n)$$

$$= -(2^{n+1} - 2) + n\times 2^{n+1} \Rightarrow -\left[2^{n+1} + 2^{n+1}\times n\right] + 2$$

$$2^{n+1}(n-1) + 2$$

The Series

- $\sum_{i=1}^{n} i\,2^i = S$
- $S = 2 + 2 \times 2^2 + 3 \times 2^3 + 4 \times 2^4 + \ldots + n \times 2^n$
- $2S = 2^2 + 2 \times 2^3 + 3 \times 2^4 + 4 \times 2^5 + \ldots + n \times 2^{n+1}$
- $2S - S = -2 + 2^2 - 2 \times 2^2 + 2 \times 2^3 - 3 \times 2^3 + 3 \times 2^4 - 4 \times 2^4 \ldots + (n-1) \times 2^n - n \times 2^n + n \times 2^{n+1}$
- $S = -2 - 2^2 - 2^3 - 2^4 - 2^n + n \times 2^{n+1}$
- $S = -\frac{2(2^n - 1)}{(2-1)} + n \times 2^{n+1}$
- $S = -2^{n+1} + 2 + n \times 2^{n+1} = 2^{n+1}(n-1) + 2$

$n \times 2^{n+1}$

$\Theta(n\,2^n)$

The Series

$$\sum_{i=1}^{n} i\,2^i = 2^{n+1}(n-1) + 2$$

The Series

$$\sum_{i=1}^{n} i\,2^i = \quad 2^{n+1}(n-1)+2$$

The Series

$$\sum_{i=0}^{n} \frac{1}{2^i} =$$

$$\frac{1}{2^0} + \frac{1}{2} + \frac{1}{2^2} + \cdots + \frac{1}{2^n}$$

$(n+1)$

$$\frac{a(1-r^n)}{1-r}$$

$$\frac{1\left(1-\frac{1}{2^{n+1}}\right)}{1-1/2}$$

$$= 2\left(1-\frac{1}{2^{n+1}}\right)$$

$$= \left(2-\frac{1}{2^n}\right)$$ → Decreasing

$$\lim_{n\to\infty}\left(2-\frac{1}{2^n}\right) \qquad \frac{1}{\infty} = 0$$

≈ 2

O(1)

Bound for 1 ?

Constant

Summations With General Bounds

Summations With General Bounds

- When a summation does not start at the 1 or 0, as most of the above formulas assume, you can just split it up into the difference of two summations. For example, for $1 \leq a \leq b$

$$\sum_{i=a}^{b} f(i) = \sum_{i=0}^{b} f(i) - \sum_{i=0}^{a-1} f(i)$$

$69+70+71+72+\cdots+85$

$1+2+3+\cdots 68 + 69 + \cdots + 85$

$$\frac{85 \times 86}{2} - \frac{68 \times 69}{2} = 1305$$

$$\sum_{i=a}^{b} 1 \qquad b-a+1$$

The Series

- $a \neq 1$ $a>1$

$$\sum_{i=j}^{n} a^i = \sum_{i=0}^{n} a^i - \sum_{i=0}^{j-1} a^i$$

Solve

$$a^0+a^1+a^2+\cdots+a^n - (a^0+a^1+a^2+\cdots+a^{j-1})$$

$$\frac{a^0(a^{n+1}-1)}{a-1} - \frac{a^0(a^j-1)}{a-1}$$

$$\frac{a^{n+1}-1-a^j+1}{a-1} = \left(\frac{a^{n+1}-a^j}{a-1}\right)$$

The Series

- $a \neq 1$

$$\sum_{i=j}^{n} a^i = \frac{a^{n+1} - a^j}{a - 1} \quad (a \neq 1)$$

Mathematical foundation

Required for analysis

The Series

- $a \neq 1$

$$\sum_{i=j}^{n} a^i = \frac{a^{n+1} - a^j}{a - 1} \quad (a \neq 1)$$

Factorials

The notation $n!$ (read "n factorial") is defined for integers n ≥ 0 as

$$n! = \begin{cases} 1 & if\ n = 0 \\ n.(n-1) & if\ n > 0 \end{cases}$$

Thus $n!$=1.2.3……..n

n!

find bound for n!

* Remember

. Heap Sort

. Construction of BST

Best/ worst case

weak upper bound

$n! = \underbrace{n.(n-1)(n-2)(n-3)\cdots 1}_{(n-0)}$

$\approx n^n + \cdots$ ($n - (n-1)$)

How many n's getting multiplied

$n!$ bounded by $O(n^n)$

Week Upper Bound

- A weak upper bound on the factorial function is $n! \leq n^n$, since each of the n terms in the factorial product is at most n.

$n!$ is $O(n^n)$

- Stirling's approximation,

- $n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1 + \Theta\left(\frac{1}{n}\right)\right)$

$\log\sqrt{2\pi n} + \log\left(\frac{n}{e}\right)^n \Rightarrow n\log n/e$

$\Rightarrow n(\log n - \log e)$ $n\log n$

$\log n!$

Asymptotic Notation

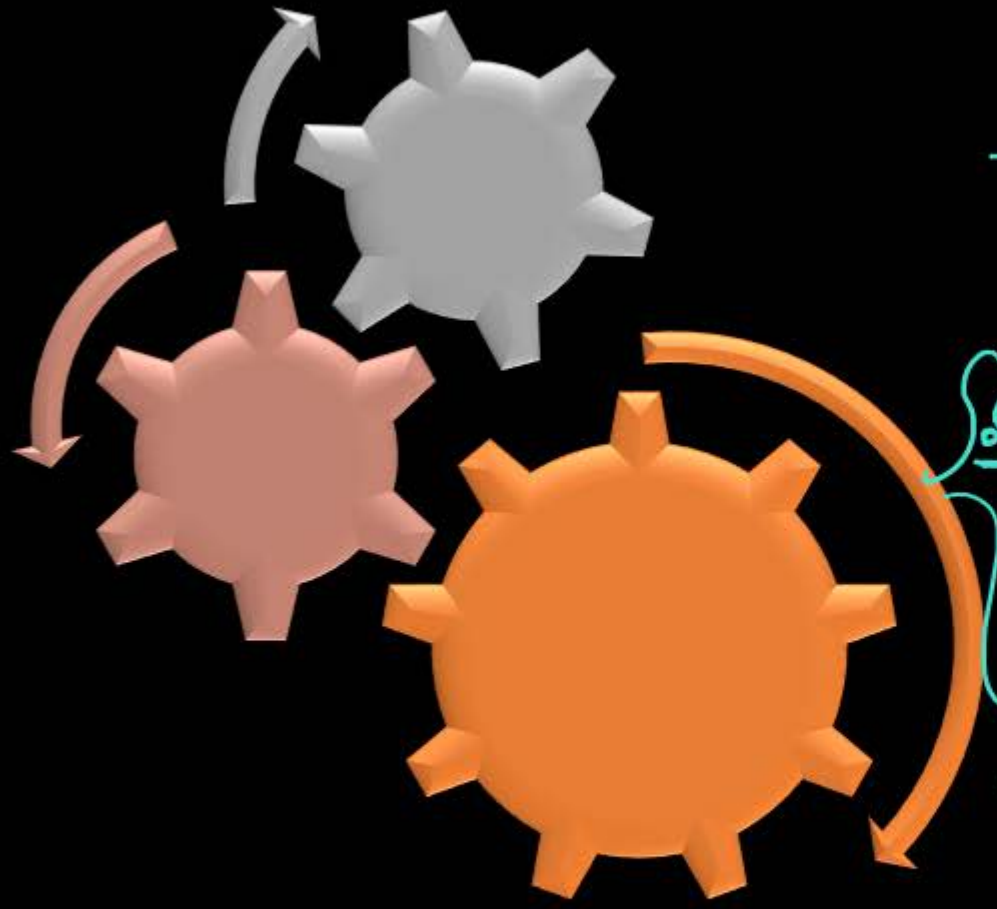
- $\log n! = \sum_{i=1}^{n} \log i$ *is* $O(n \log n)$

$$\log 1 + \log 2 + \log 3 + \cdots + \log n$$

$$= \log 1 \cdot 2 \cdot 3 \cdots n$$

$$= \log n! \text{ is } O(n \log n)$$

Important



1. Asymptotic Notation
   Comparison of function } 17 ← Text book
   . Walls book
   . GGB

2. Asymptotic Analysis
   (I) finding complexity of program
   (II) finding approximate value

for Loop / while loop

Problem Solving

Question

(a) Is $2^{n+1} = O(2^n)$?

(b) Is $2^{2n} = O(2^n)$?

For Exponential

Additive power is a constant , multiplicative power is not contant.

Question

Asymtotic C

For each of the following pairs of functions, either

I $f(n)$ *is in* $O(g(n))$ ✓

II $f(n)$ *is in* $\Omega(g(n))$ or ✓

III $f(n)$ *is in* $\Theta(g(n))$. Determine which relationship is correct.

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

$f(n) - \log\sqrt{n} = \frac{1}{2}\log n$

$g(n)$ is $\log n^6 = 6\log_2 n$

Question

Consider the following function

$$4n^2,\ \log_3 n,\ 3^n,\ 20n,\ 2,\ \log_2 n,\ n^{2/3}$$

Arrange the expressions by asymptotic growth rate from slowest to fastest.

*Never Compare log function considering the base of it .

$\log_2 n \quad - \quad \log_4 n$

$- \quad \frac{\log_2 n}{\log_2 4}$

$- \quad \frac{\log_2 n}{2}$

Question

For each of the following pairs of functions,

either $f(n)\ is\ in\ O\big(g(n)\big)$

$f(n)\ is\ in\ \Omega\big(g(n)\big)$ or $f(n)\ is\ in\ \Theta\big(g(n)\big)$.

Determine which relationship is correct and briey explain why.

$$f(n) = \log^2 n;\ g(n) = \log n$$

Question

For each of the following pairs of functions, either $f(n)$ *is in* $O(g(n))$ $f(n)$ *is in* $\Omega(g(n))$ or $f(n)$ *is in* $\Theta(g(n))$. Determine which relationship is correct and briey explain why.

$$f(n) = n;\ g(n) = \log^2 n$$

$x \quad x^2 < x^3$

$[2\log x \quad 3\log x]$ Same fun

as funchon funchon

(Both I and II)

polynomial is always bigger funchon than logarithmic

2^{1000}

$2^{2^{100}}$

$2^{10} \times 2^{100}$

$f(n) = n^{1/10}$

[log both sicle]

$\frac{1}{10}\log_2 n \leftarrow$

$g(n) = (\log n)^{100}$

[Compare the value]

$100 \log\log n$

Not compare them funchon

100×100

Question

For each of the following pairs of functions, either $f(n)$ *is in* $O(g(n))$

$f(n)$ *is in* $\Omega(g(n))$ or $f(n)$ *is in* $\Theta(g(n))$.

Determine which relationship is correct and briey explain why.

(I) $f(n) = n\log n + n;\ g(n) = \log n$

$f(n)$ is bigger

$f(n) = n\log n$

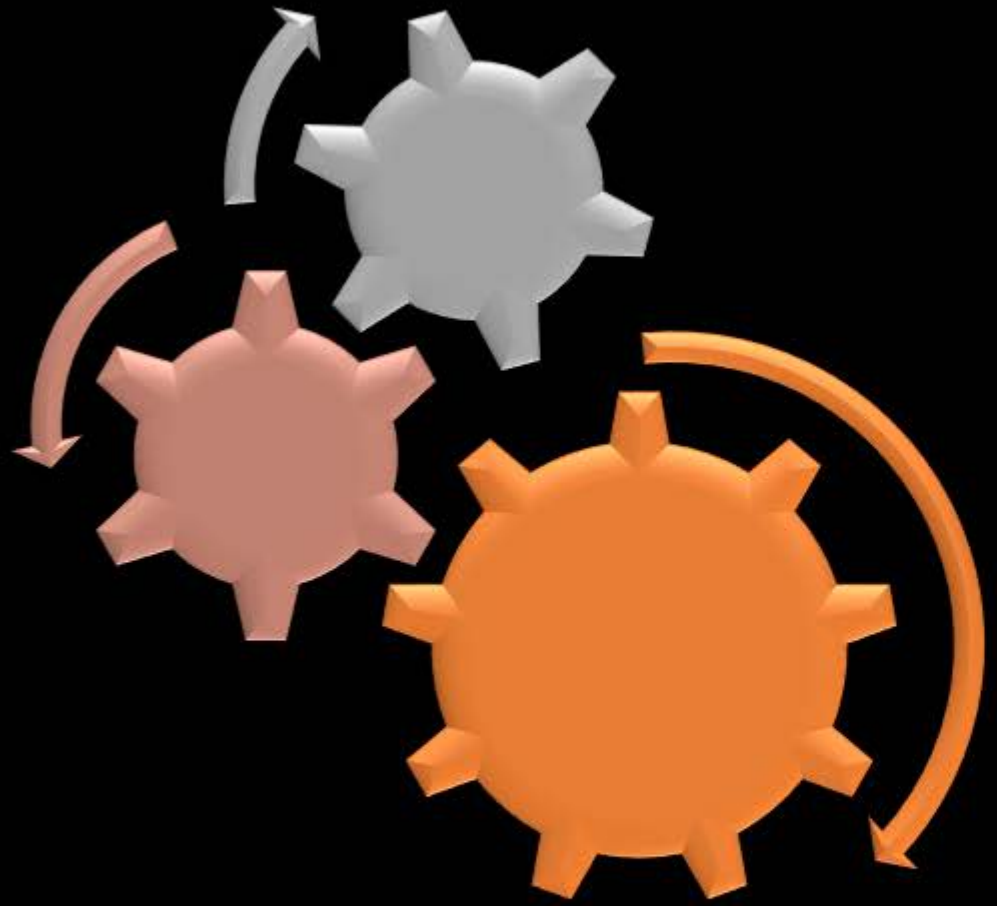
$g(n) = \log$

(II) $f(n) = n\log n$ — $f(n)$ is bigger Punch

$g(n) = n$

$n * \log n$ $\quad$ $n * 1$

divid by n



TRUE/FALSE

$f(n) = 100n\log n$ and $g(n) = \frac{n\log n}{100}$

Which of the following is TRUE?

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

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

Never compare function based on constant

True/False

- I only
- II only
- Both

$f(n) = \sqrt{\log n}$ and $g(n) = \log\log n$

Which of the following is TRUE?

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

II. $g(n) = O(f(n))$ ← (II)

$f(n) \cdot \sqrt{\log n}$

$\log\log n$

$\log n = x$

$\sqrt{x}$

$x^{1/2}$

polynomial of x

$\log x$

logarithmic function

$f(n) = n\log n$ and $g(n) = n^{1.5}$

$n \to 1$ $n \to \infty$

Which of the following is TRUE?

$n \log n$ $n \cdot n^{1/2}$

I. $f(n) = O(g(n))$ ← I

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

divide by n

$f(n)$ is smaller function
$g(n)$ is bigger function

$\log_2 n$ $n^{1/2}$

$f(n)$ $g(n)$

$f(n) = \log\sqrt{n}$ and $g(n) = \log n$

Which of the following is TRUE?

$\frac{1}{2}\log_2 n$ $\log_2 n$

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

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

$f(n) = \log_4 n$ and $g(n) = \log_2 n$

Both

Never compare log b considering then base

Which of the following is TRUE?

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

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

$f(n) = \log 2^n$ and $g(n) = n$

Which of the following is TRUE?

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

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

$f(n) = 2^{2\log n}$ and $g(n) = n^2$

Which of the following is TRUE?

Both

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

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

$f(n) = n^2 2^{3\log n}$ and $g(n) = n^5$

n^5

Which of the following is TRUE?

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

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

} Both

$f(n) = n^2 2^{\log_2 n^3}$

$= n^2 \cdot n^3$

$= n^5$

$f(n) = n^2 \log n$ and $g(n) = n^{\log n}$

$n^{\log n}$ – Not polynomial

Which of the following is TRUE?

I. $f(n) = O(g(n))$ – (I)

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

$f(n)$

$n^2 \cdot \log_2 n$

$\log_2$ · Both side

$2 \log n + \log \log n$

$n^c \leftarrow$ constant

$\frac{n}{n^2 \cdot n^{\log \log} \cdot n^{200}}$

$(2^{10})^{\times 2}$

$g(n)$ polynomial

$n^{\log n} \leftarrow$ constant

$\log n * \log n$ –

$2^{128} \leftarrow$

2^{100}

$n^3 < n^{\log n}$

1 ↑

$2 \times 128 + 7$

$256 + 7$

263

128×128

2^{14}

$f(n) = n^{\log_2 n}$ $g(n) = 2^n$

$f(n)$ is smaller function than $g(n)$

Not polynomial

exponential function

$n \to \infty$

2^{128} $\log_{10}$

10^{1000}

Log Both side

$\log_2 n \cdot \log_2 n - \log n \log 2^n$

$= n \cdot \log_2 2$

$= n$

$= 2^{128}$

$128 * 128$

$= 2^{14}$

$f(n) = \frac{4^n}{2^n}$ and $g(n) = 2^n$

Which of the following is TRUE?

I. $f(n) = O(g(n))$ ✓ true

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

$$f(n) = \frac{4^n}{2^n}$$

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

$$= 2^{2n-n}$$

$$= 2^n$$

$n \log n < 2^n$

Not polynomial function

$g(n) = 2^n$

Practice

$f(n) = \sum_{i=1}^{n} \log_2 i$ and $g(n) = n \log n$ $= \log n!$

$n \to \infty$ $\log n!$ Asymtotic

Which of the following is TRUE?

$\log 1 + \log 2 + \log 3 \cdots \log n$

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

$\log 1 \cdot 2 \cdot 3 \cdot \cdots \cdot n$ Both

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

$\log n!$

$\log n! = n \log n$

$\approx O(n \log n)$

Problem Solving

GATE-1994

Consider the following two functions:

$$g_1(n) = \begin{cases} n^3 \text{ for } 0 \le n \le 10,000 \\ n^2 \text{ for } n \ge 10,000 \end{cases}$$

$$g_2(n) = \begin{cases} n \text{ for } 0 \le n \le 100 \\ n^3 \text{ for } n > 100 \end{cases}$$

Which of the following is TRUE?

(A) $g_1(n)$ is $O(g_2(n))$

(B) $g_1(n)$ is $O(n^3)$

(C) $g_2(n)$ is $O(g_1(n))$

(D) $g_2(n)$ is $O(n)$

Asymtotically $n \to \infty$

Q7 Text book Q.6

n^2 n^3

A, B

$g_1(n)$ is $O(g_2(n))$

True

n^2

True

A, B

Later Look into option (D)

$f(n)$ is $O(g(n))$

$f(n) \le c \cdot g(n)$ $c > 0$

$n > n_0$

GATE 2017

Consider the following functions from positive integers to real numbers:

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

Constant, Logarithmic, polynomial, Exponential

Work sheet - Q.3

GQB - 24

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

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

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

~~1000~~ $\frac{100}{n}$, 10 $\log_2 n$ $\sqrt{n}$, n

"Constant function is upper bound for decreasing function"

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

$\left(2 + \frac{1}{2^n}\right)$ ← Decreasing

$n \to \infty$

Bound — 12

GATE 2017

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

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

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

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

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

$\frac{100}{n}$

GATE 2015

Let $f(n)=n$ and $g(n)=n^{(1+\sin n)}$, where n is a positive integer. Which of the following statements is/are correct?

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

a) Only I

b) Only II

c) Both I and II

d) Neither I nor II

decreasing function – upper bound

GQB-28

f(n) is Linear function

$g(n) = n^{1+\sin n}$

Sinn - oscillation function

(D)

g(n) function

$n^{1+1} = n^2$ ($n \to \infty$)

n^2 ___ n oscillation

$n^{1-1} = n^0 = 1$

Lesser

GATE 2011

Which of the given options provides the increasing order of asymptotic complexity of functions f_1, f_2, f_3 and f_4? $f_1(n) = 2^n$; $f_2(n) = n^{3/2}$; $f_3(n) = n\log_2 n$; $f_4(n) = n^{\log_2 n}$

(A) f_3, f_2, f_4, f_1 (B) f_3, f_2, f_1, f_4 (C) f_2, f_3, f_1, f_4 (D) f_2, f_3, f_4, f_1

[A] f_3 f_2, f_4, f_1

$f_2(n) = n^{3/2}$

$= n^{1.5}$

$= n * n^{0.5}$

$n^{0.5}$

Completed in parts

$f_3(n) = n \log n$

$n \log n$

$n * \log n$

$\log n$

Queshon - 1 text book

GGB - 23

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

$\log_2$ both sicle *

$\log_2 n \cdot \log_2 n - n \cdot \log 2$

put value of n and compare - 2^{10}

$10 * 10 - 2^{10} * 1$

100 1024

GATE 2008 IT

Arrange the following functions in increasing asymptotic order.

a. $n^{1/3}$ b. e^n c. $n^{7/4}$ d. $n\log^9 n$ e 1.0000001^N

(A) a, d, c, e, b — [A]

(B) d, a, c, e, b

(C) a, c, d, e, b

(D) a, c, d, b, e

$n \to \infty$

Constant n >1 $\Rightarrow$ expon

$n^{0.33}$ | Constant | | Logarithm ~~polynomia~~ | | poly nomil | | exponential |

$$\frac{n^{1/3}}{a} < \frac{n\cdot(\log n)^9}{d} < \frac{n^{7/4}}{c} < e\ 1.000000001^n < e^n$$

$n^{1+3/4}$ $\quad n^1$ $\quad$ e $\quad$ b

$$\frac{n\cdot(\log n)^8}{(\log n)^9} \cdot \frac{n^1 \times n^{0.75}}{n^{0.75}}$$

GATE 2008 IT

(A) a, d, c, e, b

(B) d, a, c, e, b

(C) a, c, d, e, b

(D) a, c, d, b, e

GATE 2001

Let $f(n) = n^2 \log n$ and $g(n) = n(\log n)^{10}$ be two positive functions of n. Which of the following statements is correct?

(a) f(n) = O(g(n) and g(n) ≠O(f(n))

(b) g(n) = O(f(n) and f(n) ≠O(g(n))

(c) f(n)≠O(g(n)) and g(n) ≠O(f(n))

(d) f(n)=O(g(n)) and g(n) =O(f(n))

GATE 2003

Consider the following three claims

I. $(n+k)^m = \Theta(n^m)$ where k and m are constants

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

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

Which of these claims are correct?

(A) I and II (B) I and III

(C) II and III (D) I, II, and III

true

true

$\frac{(n+1)^2}{(n+1)^3} = \frac{n^2+1+2n}{(n+1)(n+1)}$ $\Theta(n^2)$

2^{n+1} is $O(2^n)$

Additive constant

(II) $2^{n+1} \leq 2^n$

$2 \cdot 2^n \leq C \cdot 2^n$

(III) $2^{2n+1} \leq 2^n$

$2 \cdot 2^{2n} \leq 2^n$

*Remember

GATE 2021 Set-I, Question Number 3,

GATE Question Bank - 26

Consider the following three functions.

$$f_1 = 10^n \quad f_2 = n^{\log n} \quad f_3 = n^{\sqrt{n}}$$

Which one of the following options arranges the functions in the increasing order of asymptotic growth rate?

(A) f_3, f_2, f_1

(B) f_2, f_1, f_3

(C) f_1, f_2, f_3

(D) f_2, f_3, f_1

[D]

Last year ←

$$\underset{f_2}{n^{\log n}} < \underset{f_3}{n^{\sqrt{n}}} < \underset{f_1}{10^n}$$

Standard of Asymptotic Notation

$\sqrt{n}/\log n < \sqrt{n}/\sqrt{n}$

10^6

$n^{\sqrt{n}} < 10^n$

take $\log_{10}$

$\sqrt{n} \log_{10} n \quad n \cdot \log_{10} 10$

$10^3 \cdot 6 \quad \frac{n}{10^6}$

Algorithmic Complexity

for and while loop

Algorithmic Analysis

tools

Mathematical foundation

1. Algorithmic Complexity
2. value approximated

loop

for and while loop

Constant time algorithm

Constant time

Algorithm whose running time is independent of input size

assignment / funchon call

Condition check / funchon call

assignment / funchon call

```
for(i =0; i≤ 1000;i++){
  body of the loop;}
}
```

⇒

```
i=0;
while (i<=1000) {
  body of the loop
  i++;
}
```

one time

"body of the loop"

Last statement

O(1)

Size of Input

1000

Condition checking is done one extra time when condition is false

```
fun( int n...) {
  for (i=0; i<=1000; i++)
  {
  }
}
```

≈ 1000 — n = 1000

≈ — n = 10^6

Constant time algorithm

Question

final value of a is?

```
int a =1
for(i = 19; i≤ 84;i++){
a=a+2 a=a*2;
}
```

① 67 66 times

66 times

Analysis:

UB − LB + 1

84 − 19 + 1

65 + 1 = 66

2^{66}

a = 1 * 2 = 2

a = 2 * 2 = 2^2

a = 2^2 * 2 = 2^3

a = 2^{66} — 66

Constant time algorithm

```
i=0;
while (i ≤1000){
   body of the loop;
   i++;
}
```

Constant time Algorithm

Linear time algorithm

```
fun(int n, ...){
    for (i =0; i < n; i++){
      body of the loop;
    }
}
```

for loop

← θ(n)

Complexity of function fun

i = 0 ... i = n-1

n times

Linear time algorithm

fun (int n . . .) {

```
i=0;
while (i<=n){
   body of the loop;
   i++;
}
```

}

Same effect

i = 0 - - - i = n

n - 0 + 1 ⟹ n+1

$\Theta(n)$

Linear time algorithm

- `for (i =0; i < n/2; i++){`
  `body of the loop;`
  `}`

$\Theta(n)$

for loop execute?

$i = 0 \quad \ldots \quad i = n/2 - 1$

$UB - LB + 1 = n/2 - 0 + 1$

$= n/2$

- `for (i =0; i < n/4; i++){`
  `body of the loop;`
  `}`

$\Theta(n)$

$i = 0 \quad \ldots \quad i = n/4 - 1$

$n/4 - 0 + 1$

$n/4$

$\Theta(n)$

```
for ( i=0; i<= n/10 ; i++)
    body of the loop
```

Linear time algorithm

- `for (i =0; i < n; i+=2){`
  `body of the loop;`
  `}`

$\Theta(n)$ $\approx n/2$ $i = i+2$

1 — 2 — 3 — 4 - - - — 100

1 — 3 — 5 — 7 - — 100

- `for (i =0; i < n; i+=4){`
  `body of the loop;`
  `}`

$\Theta(n)$ $\approx n/4$

$O(n/2)$ κ

$O(n)$

$\approx n$ ← `for (i=0; i<n; i++)`

Constant ← `for (j=0; j<=1000; j++)`

Body of the loop

} time Complex

$1000n$

$O(n)$

Mute | Stop Video | Participants | Chat | Q&A | Polls | New Share | Pause Share | Annotate | More

You are screen sharing | Stop Share

$n+1 + n^2 + n + n^2$

$2n^2 + 2n + 1$ $\quad \Theta(n^2)$

for (i = 1 to n { $\quad$ $n+1$

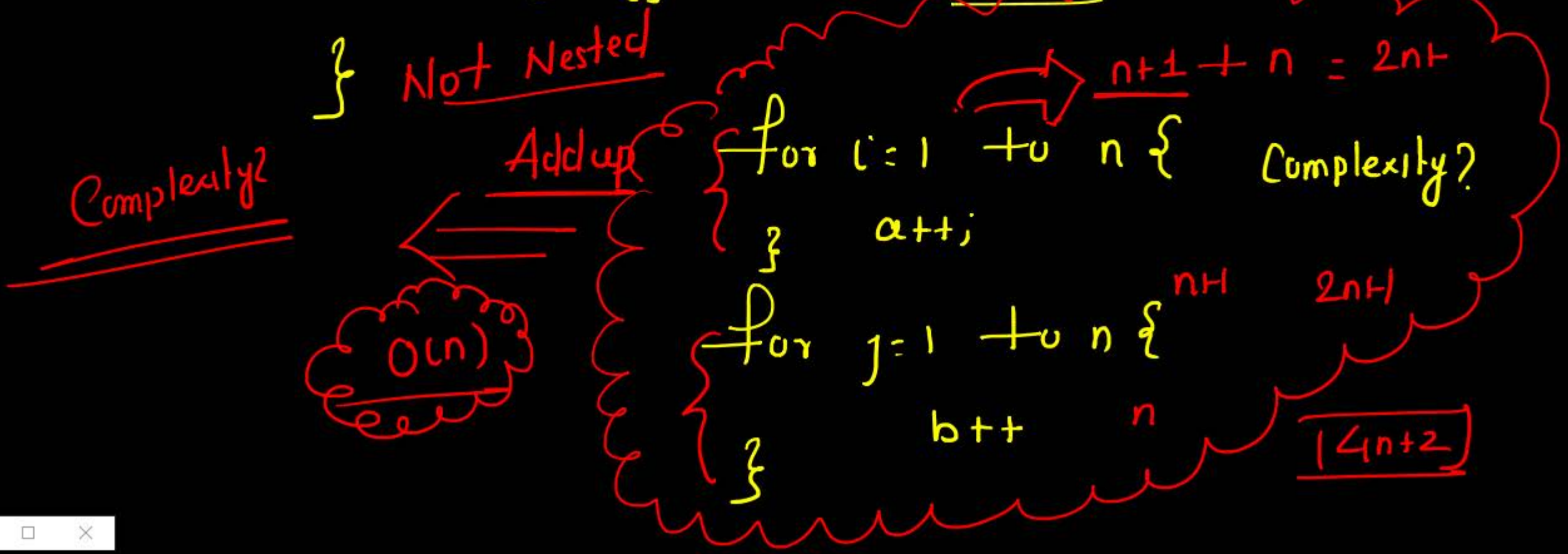
for j = 1 to n ← $\quad$ $n*(n+1)$

C[i,j] = a[i,j] + B[i,j] $\quad$ $n*n$

} Not Nested

Addup

Complexity?



Quadratic time algorithm

n^2

```
for (i =0; i < n; i++){
    for(j =0; j<n; j++){
        body of the loop;
    }
}
```

$\Theta(n^2)$

	j=0	j=1	j=2	j=3	j=4
i=0	*	*	*	*	*
i=1	*	*	*	*	*
i=2	*	*	*	*	*
i=3	*	*	*	*	*
i=4	*	*	*	*	*

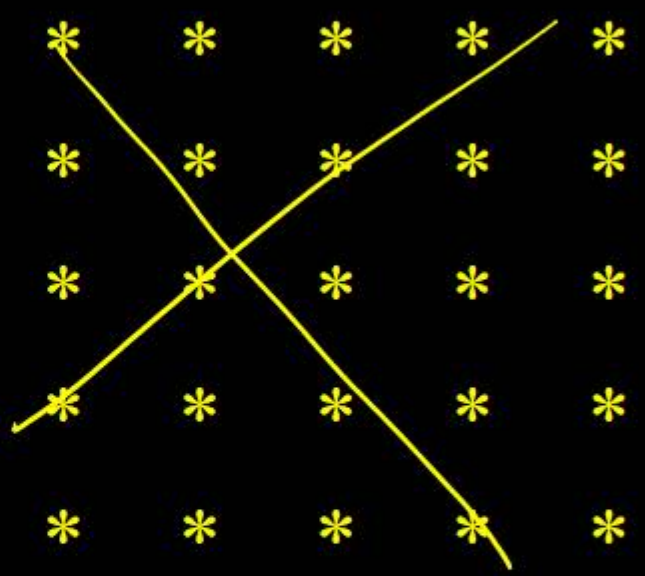
No. of star = n^2

Quadratic time algorithm

```
for (i =1; i <= n; i++){
    for(j =0; j<= n; j++){
        body of the loop;
    }
}
```

for i:1 to n ← n+1

for j:0 to n ← n+2



$(n+1)(n+2)$ $O(n^2)$

Quadratic time algorithm

$O(n)$

```
for (i =1; i <= n; i++){
    for(j =1; j<= i; j++){
        body of the loop;
    }
}
```

$\frac{n(n+1)}{2} \approx \Theta(n^2)$

$$\sum_{i=1}^{n} \sum_{j=1}^{i} 1$$

	j=1	j=2	j=3	j=4	j=5
i=1	*				
i=2	*	*			
i=3	*	*	*		
i=4	*	*	*	*	
i=5	*	*	*	*	*

No. of star - $1+2+3+\cdots+n$

$= \frac{n(n+1)}{2} \approx \Theta(n^2)$

2014 – IIT KGP

Triple Summation

for Loop analysed using Summation

$$\sum_{i=1}^{n} \sum_{j=1}^{i} 1$$

$$\left(\frac{UB - LB + 1}{i - 1 + 1}\right) \times 1$$

$$= \sum_{i=1}^{n} (i - 1 + 1) \times 1$$

$$= \sum_{i=1}^{n} i = 1 + 2 + 3 + \cdots + n$$

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

Quadratic time algorithm

```
for (i =1; i <= n; i++){
    for(j =i; j<= n; j++){
        body of the loop;
    }
}
```

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

	j=1	j=2	j=3	j=4	j=5	
i=1	*	*	*	*	*	1 – 5
i=2	*	*	*	*		= 2 – 5
i=3	*	*	*			= 3 – 5
i=4	*	*				— 4 – 5
i=5	*					— 5

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

Quadratic time algorithm

```
i=0;
j=0;
while (i<=n){
    j=0;
    while(j<=n){

      body of the loop;

    j++;

}

i++;

}
```

while loop quadratic time algorithm

Quadratic time algorithm

```
for (i =1; i <= n*n; i++) {
    Body of the loop
}
```

n^2 — $i=i+2$

Single for Loop

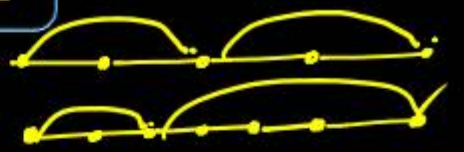
$\Theta(n^2)$

```
for (i=1; i<=n; i++)
    for (j=1; j<= n*n; j++)
```

n — n^2

n^3

Logarithmic time algorithm



$i = i+2$

$i <= n$

$i = 1$

```
for (i =1; i<n; i=i*2){
    body of the loop;
}
```

$\log_2 n$

$1 <= n$	—	1 iteration
$2 <= n$	—	2nd iteration
$2^2 <= n$	—	3rd iteration
$2^k <= n$	—	k+1 iteration

$\Theta(\log_2 n)$

Last iteration ← $2^k <= n$

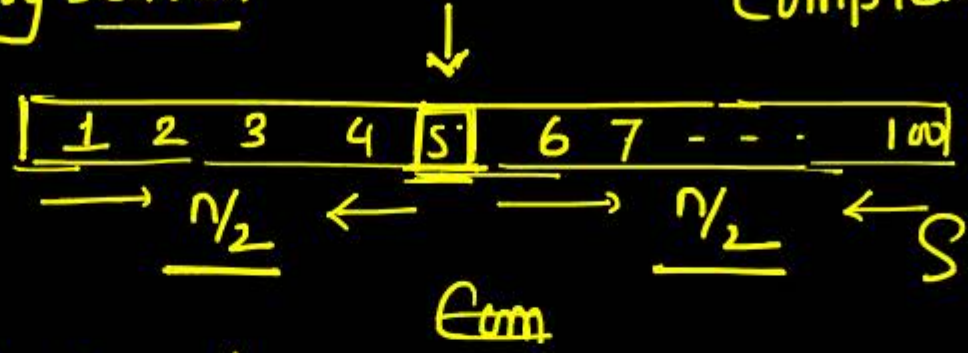
$$2^k = n$$

take $\log_2$ both sides

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

$$k = \log_2 n$$

Sorted array – Binary Search



Complexity of binary search $O(\log_2 n)$

Searching an element x

Compare the mid element

1 comparison — $n/2$

2nd comparison — $n/2^2$

⋮

kth Comparison — $n/2^k$

$n/2^k = 1$ element

$n = 2^k$

$\log_2$ both side

$\log_2 n = \log_2 2^k$

$k = \log_2 n$

Logarithmic time algorithm

* Important

```
for (i =n; i > 0; i=i/2){
    body of the loop;
}
```

i/=2

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

body of loop

$t=3 - 2^3 = 8$

$t=4 = 2^4 = 16$

$t=5 = 2^5 - 32$

$t=6 - 2^6 - 64$

$t=7 - 128$

$t=0$

$t=0 - 1$

at what time

$t=1 -$ oo

$t=2$ oo oo $- 2^2$

n

n/2

$n/2^2$

:

$n/2^k = 1$

$k = \log_2 n$

128 objects $t=0$

Every second

$t=0 - 128$

$t=1 - 64$

$t=2$ 32

$t=6 - 2$

$t=7$ 1

a duplicate of object is produced

$t=3 - 16$

$t=4 - 8$

$t=5 = 4$

Question

```
for (i =n; i > 0; i=i+2){
    body of the loop;
}
```

Logarithmic time algorithm

```
for (i =1; i<n; i=i*2){
    body of the loop;
}
```

You can change 2 and get different base of log

Logarithmic time algorithm

```
i=1;
while (i<=n){
    body of the loop;
 i = i*2;
}
```

Logarithmic time algorithm

```
i=n;
while (i>0){
    body of the loop;
 i = i/2;
}
```

Complexity??

16 25

i*i<=16

```
for (i =1; i*i<n; i++){
    body of the loop;
}
```

$i^2 <= n$

$i <= \sqrt{n}$

Question Complexity is $O(\sqrt{n})$ $O(n^2)$

for (i=1; i <= n*n; i++)

n = 16

$1 <= 16$ ✓

$2*2 \leq 16$

$3*3 \leq 16$ ✓

$4*4 \leq 16$

$5*5 \leq 16$ — false

1. Constant time
2. Linear time
3. Quadratic time
4. Logarithmic time
5. Square Root
   - I - part
   - II - part

for Loop / while Loop

Complexity??

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

    body of the loop;

}
```

Square Root time algorithm

```
S = 0; i=1;
while (S<=n){
 S =S+i;
 i = i+1;
}
```

Problem Solving

Question

```
#include<stdio.h>
int main ( ){
int x,sum;
for ( x = 0, sum = 0; x <= 500; x+=10 )
      sum += x ;
printf("%d",sum);
return 0;
}
```

what is the value

0 + 10 + 20 + 30 + - - - + 500

10 (1 + 2 + 3 + - - + 50) → Arithmetic Series

$$\frac{10 \times 50 \times 51}{2} = 12750$$

Output of the C-program is 12750

Question

Consider the following code

```
int P = 0;
for(i= 1; i < 2n; i++) {
        for(j = 1, j<=n; j++){
                if(j < i) P = P+1;
        }
}
printf( "%d", P);
```

n = 4

i	j	P
1	1 →	0
2	2	1
3	3	2
4	4	3
5 →		4
6		4
7		4

6+12 = 18

What is the output printed by the above code in terms of n?

$3 \times 16 - 12$

$48 - 12 = \frac{36}{2} = 18$

(A) $\frac{4n^2 - n}{2}$

(C) $\frac{n^2 - 4n}{2}$

(B) $\frac{3n^2 - 3n}{2}$ ✓

(D) $\frac{n^2 - 3n}{2}$

(B)

Question

Consider the following code

```
int P = 0;
for(i= 1; i < 2n; i++) {
        for(j = 1, j<=n; j++){
                if(j < i) P = P+1;
        }
}
printf( "%d", P);
```

What is the output printed by the above code in terms of n?

(A) $\frac{4n^2 - n}{2}$

(C) $\frac{n^2 - 4n}{2}$

(B) $\frac{3n^2 - 3n}{2}$

(D) $\frac{n^2 - 3n}{2}$

Summation form

UB − LB + 1

2n − 1 − n + 1 = (n − 1)

i	j	—
i = 1	j = 1	0
i = 2	j = 2	1
i = 3	j = 3	2
⋮	⋮	
i = n	j = n	n − 1
i = n + 1	—	n
i = n + 2	—	
⋮		
i = 2n − 1	—	n

$0 + 1 + 2 + 3 + \cdots + (n-1) + \underbrace{n + n + \cdots + n}_{n-1}$

$\frac{n(n-1)}{2} + n(n-1)$

$\frac{n(n-1)}{2}\left[\frac{1}{2} + 1\right] = \frac{3(n^2 - n)}{2}$

$= \frac{3n^2 - 3n}{2}$