Data structure & algorithm

Data structure

- particular way of storing and organizing data in a computer so that it can be used efficiently

Ex:- arrays, linked lists, stacks, queue, trees, graphs etc.

DS → Linear, Non-Linear

Linear	Non-Linear
⇝ linear order	⇝ non-linear order
⇝ linked lists, stack etc.	⇝ Trees and graphs

Algorithm

Step by step fixed unambiguous instruction to solve a problem

properties ⇝

1) Each step must be a basic operator like we can't write directly in algo that sort the given numbers
2) Should terminate in finite amount of time
3) Should produce atleast 1 output
4) Should take 0 or more input
5) Proper mapping
6) Platform independent
7) Every statement must be unambigous

Steps required

- Problem Definition
- Development model
- Design
- Testing ⇝ To find error
- Analysis ⇝ Measuring S.C & T.C
- Optimize ⇝ minimize T.C & S.C
- implementation

Analysis of algorithm

it means not only just measuring T.C & S.C but also analysing the performance of algo.

⇝ How to analyze performance of algorithm

2 ways

Apriori analysis	Aposteriori analysis
⇝ Before implementation	⇝ After implementation
⇝ Can calculate approx S.C & T.C	⇝ Used to calculate exact S.C & T.C
⇝ We use this while analysing	⇝ Not used
	⇝ Bcz it gives credit to machine & not developer of algo

⇝ How to analyze algo before implementation

S-1 Hypothetical language

S-2 Hypothetical CPU with ∞ RAM (Each fundamental operation need 1 unit of time)

S-3 Select a matric for analyzing algo

S-4 Approach for categorizing the running of algo (Asymptotic notation)

Fundamental operations

Assignment	Arithmatic	Logical	Relational	I/O
=	+, -, *, /, %	AND, OR, NOT	>, <, ==, !=	read, write

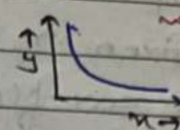
Ways of apriori analysis

- Step count method
- Order of magnitude
  - ↳ we use this
  - ⇝ Calculate only major operator
  - Ex:- $f(n) = n^2 + 3n + 2$
    - $\rightarrow (n^2)$ Ans

Time complexity
quantifies the amount of time (CPU time) taken by an algorithm to run as a funcⁿ of I/P size

Space complexity
Quantifies the space (RAM size) taken by an algorithm to run as a funcⁿ of I/P size

Tangents to the curve at ∞



→ x & y axis are asymptotes to this curve at ∞

→ curve is bounded by x & y axis

Asymptotic notation
- → O (Big-oh)
- → Ω (Big-omega)
- → Θ (theta)
- → o (Small-oh)
- → ω (Small-omega)

Asymptotic analysis of algorithms
- Analysing time & space needed by algorithm when $n \to \infty$
- Analysis of algorithm is generally done for very large input size
- Apriori analysis gives time needed by algorithm as a funcⁿ of Input size
- Algo 1: $f(n) = n^2 + n + 1$
  Algo 2: $f(n) = n^2$

  Mathematically A1 need more time but asymptotically ($n \to \infty$) both have same time complexity
- A1 : n^2 , A2 : $10000n$

→ Mathematically we will say till 9999 inputs, A2 need more time but beyond 10000 A1 need more time

Asymptotically A1 needs more time than A2.

Conclusion : 1) constants don't matter
2) Take dominating term

reasons : • $n \to \infty$
• machine independent

Date
Page No.

→ We should have ways to represent time and space complexity of algorithm which is independent of machine specification & these ways are asymptotic notations.

① Big-oh notation:
Given 2 f^n $f(n)$ & $g(n)$
we say $f(n) = O(g(n))$ if we were able to find atleast 1 +ve constant c & a value of n called n_0 [$n_0 \geq 1$] such that $f(n) \leq c \cdot g(n)$ $\forall$ $n \geq n_0$

$O \to \leq$

• $f(n) = O\,g(n)$ means $g(n)$ is greater than or equal to $f(n)$ by taking help of c (constant) i.e
$f(n) \leq c \cdot g(n)$

• if $g(n)$ is already bigger then no need to take help of c ($c=1$) but if $g(n)$ is smaller than $f(n)$ then we need to find c such that $c \cdot g(n)$ is always bigger or equal to $f(n)$ but this relationship holds starting from which value of n?? this value is n_0

n → no. of inputs always greater than or equal to 1

Q- $f(n) = n$, $g(n) = 3n+2$

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

Ans- Clearly
$g(n) \geq f(n)$
so take $c = 1$
and yes
$f(n) = O(g(n))$

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

Ans- here for any constant c greater than 3
let $c = 3.1$
$3n+2 \leq 3.1(n)$
$2 \leq 0.1n$
$n_0 = 20$ $\boxed{20 \leq n}$ ✓
$3n+2 \leq 3.1n$ $\forall n \geq 20$

n_0 should be an integer

Note

Q- $f(...$
(i...

No

(1)
(2)
(3)

②

Note: Big-oh represents upper bound
Big-omega represents lower bound

Date
Page No.

and
independent
asymptotic
find
called
$n \geq n_0$
equal
take
n
$c \cdot g(n)$
of
stant
s
o

Q- $f(n) = 1000n$, $g(n) = n^2$

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

$1000n = O(n^2)$

$1000n \leq c \cdot n^2$

let $c = 1$

$1000n \leq n^2$

$\boxed{n_0 = 1000}$

if $c = 2$

$n_0 = 500$

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

$n^2 = O(1000n)$

$n^2 \leq c \cdot 1000n$

analyse and you'll get to know that we can't find any value greater for which it is always

→ Not possible

Note: if time taken by an algorithm is independent of input size then its time complexity is $O(1)$

conclusion:

(1) always take dominating term for big-oh
(2) if $f(n)$ is $O(n)$ then $f(n)$ is $O(n^2)$, $O(n^3)$ & so on
(3) $O(n)$ represents ∞ func^n i.e. its not a single func^n → whose dominating term is $\leq n$

② Big-omega notation:

Given 2 func^n $f(n)$ & $g(n)$
we say $f(n) = \Omega\, g(n)$ if we are able to find atleast 1 +ve constant c & a value of n called n_0 s.t

$$f(n) \geq c \cdot g(n) \quad \forall n \geq n_0$$

Ex:- $f(n) = 3n+2$, $g(n) = n$

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

$3n+2 \geq c \cdot n$

let $c = 1$

$\forall n \geq 1$

$\boxed{n_0 = 1}$

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

$n \geq c \cdot (3n+2)$

let $c = 1/4$

$n \geq \frac{3}{4}n + \frac{1}{2}$

$\forall n \geq 2$

$\boxed{n_0 = 2}$

Conclusion:

(1) Always take dominating term for Ω

(2) If $f(n)$ is $\Omega(n^3)$ then $f(n)$ is $\Omega(n^2)$, $\Omega(n)$ & so on.

(3) $\Omega(n)$ represents ∞ functions i.e not a single func^n^

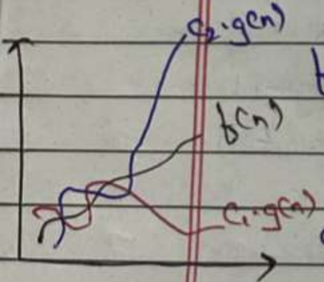
(3) Theta notation:

Given 2 f^n $f(n)$ & $g(n)$

we say $f(n) = \Theta(g(n))$ if we are able to find atleast 2 +ve constant c_1 & c_2 & a value of n called n_0 $(n_0 \geq 1)$ such that

$$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \quad \forall n \geq n_0$$

$f(n) \leq c_2 \cdot g(n) \quad \forall n \geq n_0$ - (i) big-oh



$f(n) \geq c_1 \cdot g(n) \quad \forall n \geq n_0'$ - (ii) big-omega

from (i) & (ii)

$$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \quad \forall n \geq \max(n_0, n_0')$$

Q- $f(n) = 3n+2$, $g(n) = n$

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

$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$

$c_1 \cdot n \leq 3n+2 \leq c_2 \cdot n$

let $c_1 = 1$ and $c_2 = 4$

$n \leq 3n+2 \leq 4n$

$n_0 = 2$

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

$c_1 \cdot f(n) \leq g(n) \leq c_2 \cdot f(n)$

$c_1 \cdot (3n+2) \leq n \leq c_2 \cdot (3n+2)$

let $c_1 = \frac{1}{4}$ let $c_2 = 1$

$\frac{(3n+2)}{4} \leq n \leq 3n+2$

$n = 2$

$n_0 = 2$

Q- $f(n) = 1000n$, $g(n) = n^2$

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

$c_1 \cdot n^2 \le f(n) \le c_2 \cdot n^2$

$c_1 \cdot n^2 \le 1000n \le c_2 \cdot n^2$

not possible bcz of dominating term (omega fails)

let $c_2 = 1$ (Big-oh pass)

ANS ⇝ not correct eqn

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

$c_1 \cdot 1000n \le n^2 \le c_2 \cdot 1000n$

let $c_1 = 1$ (omega pass)

not possible bcz of dominating term (Big-oh fails)

ANS ⇝ Not a correct reln

Note : If time taken by algorithm is independent of input size, then its time complexity is $\Theta(1)$

Conclusion :

(1) Always take dominating term for Θ

(2) If $f(n)$ is $\Theta(n^3)$ then $f(n)$ is $\Theta(n^2)$, $\Theta(n)$ & so on — wrong statement only valid for equal

(3) $\Theta(n)$ represents ∞ funcn i.e. not a single funcn

(4) Ω is lower bound, O is upper bound, Θ is tight bound

Note : Theta notation is the most suitable and accurate among all because

if we say $\Theta(n)$ ⇝ it is guranteed that 'n' is dominating term

but

if we say $O(n)$ ⇝ not guranteed bcz 'n' se niche wali power bhi dominating term hoge

and

if we say $\Omega(n)$ ⇝ not guranteed bcz n^2, n^3 & so on ye sab dominating term hoge

④ Small-oh notation

Given 2 funcⁿ $f(n)$ & $g(n)$
we say $f(n) = o(g(n))$ if for every +ve constant c
we are able to find a value of n called n_0 s.t.

$$f(n) < c \cdot g(n) \quad \forall n \geq n_0$$

agar $f(n) = n^2$ toh $g(n)$ ko hamesha bdi power lena like $g(n) = n^3, n^4, \ldots$

$f(n) = 3n+2$, $g(n) = n^2$

~~f(n) = o(~~ $f(n) = o(g(n))$ ✓ $\qquad g(n) = o(f(n))$ ✗

secondry definition

$f(n) = o(g(n))$ iff

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

simply we can say that when $n \to \infty$

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

⑤ Small omega notation

Given 2 funcⁿ $f(n)$ & $g(n)$
we say $f(n) = \omega(g(n))$ if for every +ve constant c
we are able to find a value of n called n_0 s.t.

$$f(n) > c \cdot g(n) \quad \forall n \geq n_0$$

agar $f(n) = n^3$ toh $g(n)$ ko hamesa choti power lena like $g(n) = n^2, n, n\log n$ so on

another definition

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

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$$

when $n \to \infty$

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

Q- why small Θ never exists

Ans- Theta notation is comprised of big-oh and big-omega

$$C_1 \cdot g(n) \le f(n) \le C_2 \cdot g(n)$$

here this equal sign
is the reason for
existing of theta notation
and not existing for small Θ

bcz according to small Θ

$$C_1 \cdot g(n) < f(n) < C_2 \cdot g(n)$$

Yaha pe dominating term
equal ho hi nhi skte bcz
ek mai hme power bde chahiye
or ek mai power choti chahiye

㊟ Some important topics to build background for properties

① $f(n) = 3n+2$
$g(n) = 6n-4$

we can say $f(n) = O(n)$ -(i)
& also say $g(n) = O(n)$ -(ii)

Now we can't conclude $f(n) = g(n)$
bcz big-oh represents set of ∞ funcⁿ

technically we need to write
$f(n) \in O(n)$
$g(n) \in O(n)$

but for better we write
$f(n) = O(n)$

that's why we cant equate $f(n) = g(n)$

$\Theta(n) \rightarrow$ set of ∞ asymptotically +ve func^n whose dominating term is $\leq n$

Date

Page No.

② Asymptotically -ve func^n

$f(n) = -n^2 + 4n + 1$

curve reaches -ve after some value and then remains -ve !

⤳ we don't use big-oh for asymptotically -ve func^n bcz time can't be negative

③ $f(n) \rightarrow a$, $g(n) \rightarrow b$

$f(n) = O(g(n)) \rightsquigarrow a \leq b$

$f(n) = \Omega(g(n)) \rightsquigarrow a \geq b$

$f(n) = \Theta(g(n)) \rightsquigarrow a = b$

$f(n) = o(g(n)) \rightsquigarrow a < b$

$f(n) = \omega(g(n)) \rightsquigarrow a > b$

⤳ ignore constants

⤳ Take only dominating term

④ All func^n are asymptotically comparable ?

NO

$f(n) = n$

$g(n) = n^{1+\sin n}$ ⤳ lies b/w (n^0 to n^2)

$f(n) = O(g(n))$ ✗

$f(n) = \Omega(g(n))$ ✗

Date
Page No.

Properties of Asymptotic notation

① Reflexivity

$R \subseteq A \times A$

$x R x \ \forall x \in A$

$f(n) = O(f(n))$ ✓
$f(n) = \Omega(f(n))$ ✓
$f(n) = \Theta(f(n))$ ✓
$f(n) = o(f(n))$ ✗
$f(n) = \omega(f(n))$ ✗

$f(n) = n$

$n = O(n)$ ✓
$n = \Omega(n)$ ✓
$n = \Theta(n)$ ✓
$n = o(n)$ ✗
$n = \omega(n)$ ✗

but if $n = o(n^2)$ ✓
$n^2 = \omega(n)$ ✓

func

constants
dominatin

② Symmetric

$R \subseteq A \times A$

$\forall a, b \in A \quad a R b \rightarrow b R a$

$f(n) = O(g(n))$ ✗ $\rightarrow (a \le b) \neq (b \le a)$
$f(n) = \Omega(g(n))$ ✗ $(a \ge b) \neq (b \ge a)$
$f(n) = \Theta(g(n))$ ✓ $(a = b) = (a = b)$
$f(n) = o(g(n))$ ✗ $(a < b) \neq (b < a)$
$f(n) = \omega(g(n))$ ✗ $(a > b) \neq (b > a)$

③ Transitive

$R \subseteq A \times A$

$\forall a, b, c \in A : a R b R c \rightarrow a R c$

$\rightarrow a \le b \ \& \ b \le c \Rightarrow a \le c$

$f(n) = O(g(n)) \ \& \ g(n) = O(h(n)) \rightarrow f(n) = O(h(n))$ ✓
$f(n) = \Omega(g(n)) \ \& \ g(n) = \Omega(h(n)) \rightarrow f(n) = \Omega(h(n))$ ✓
$f(n) = \Theta(g(n)) \ \& \ g(n) = \Theta(h(n)) \rightarrow f(n) = \Theta(h(n))$ ✓
$f(n) = o(g(n)) \ \& \ g(n) = o(h(n)) \rightarrow f(n) = o(h(n))$ ✓
$f(n) = \omega(g(n)) \ \& \ g(n) = \omega(h(n)) \rightarrow f(n) = \omega(h(n))$ ✓

Page No.

Summary

		R	S	T
$\le$	O	✓	✗	✓
$\ge$	Ω	✓	✗	✓
$=$	Θ	✓	✓	✓
$<$	o	✗	✗	✓
$>$	ω	✗	✗	✓

(4) Transpose symmetry

$$f(n) = O(g(n)) \text{ iff } g(n) = \Omega(f(n))$$

$a \le b$ & $b \ge a$ ⇝ yes, its true

(5)

$$\overset{a}{f(n)} = O(\overset{b}{g(n)})$$
$$\underset{c}{d(n)} = O(\underset{d}{h(n)})$$

$$f(n) + d(n) = O(\max(g(n), h(n)))$$

yes, its true

(6)

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

$$f(n) * d(n) = O(g(n) * h(n))$$

(7) $n^2 + \Theta(n) = \Theta(n^2)$

$n^2 + C_1 * n + C_2$ ⇝ we don't know C_1 & C_2 ⇝ also complicated to find

that's why

we ANS ⇝ $\Theta(n^2)$ by property

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

also a true identity

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

$\{3n+2,\ 5n-6,\ \vdots,\ \log n,\ \sqrt{n},\ \vdots\}$

(9) $O(n) + o(n) = O(n)$

$\{3n+2,\ 5n-9,\ \vdots,\ \log n,\ \sqrt{n},\ \vdots\}$ $\{\log n,\ \sqrt{n},\ \vdots\}$ $= \times$

not a true identity

(10) $\sum_{i=1}^{n} O(i) = O(1) + O(2) + O(3) + \cdots + O(n) = O(n)$ ✗

bcz its a set of infinite funcⁿ that why we can't expand it like that.

$1+2+3+\cdots+n = \frac{n(n+1)}{2} = O(n^2)$ ✓

$\sum_{i=1}^{n} O(i) = O(n)$ ⇝ wrong statement

Q- Let $f(n)$, $g(n)$ & $h(n)$ be 3 +ve funcⁿ defined as follows

$\overset{a}{f(n)} = O(g(n))$ but $\overset{b}{g(n)} \neq O(f(n))$ $a \le b$ & $b \not\le a$ ⇓ $\boxed{a < b}$ —(I)

$g(n) = O(h(n))^{c}$ and $h(n) = O(g(n))$ $b \le c$ & $c \le b$ ⇓ $\boxed{b = c}$ (II)

which are True

a) $f(n) + g(n) = O(h(n))$ ✓

b) $f(n) = o(h(n))$ ✓

c) $h(n) * g(n) = O(h(n) * h(n))$ ✓

d) $f(n) * g(n) = O(h(n) * g(n))$ ✗

a) $a + b \le c$ (yes)

b) $a < c$ (yes)

c) $c * b = c * c$ (yes)

d) $a * \cancel{b} = c * \cancel{b}$ (No)

Page No.

a) $O(n)$ ~~b) $\Omega(n)$~~
c) ~~$\Theta(n)$~~ d) ~~$\omega(n)$~~

Q- $f(n) = \Omega(n)$, $g(n) = O(n)$, $h(n) = \Theta(n)$
then $(f(n) * g(n)) + h(n)$

Ans-

$f(n) \geq n$ $\quad$ $g(n) \leq n$ $\quad$ $h(n) = n$

$a \geq n$ $\quad$ $b \leq n$ $\quad$ $c = n$

$(a * b) + c$

$\geq n * \leq n + n \Rightarrow n^2 * \sqrt{n} + n$

n^2 $\quad$ $\sqrt{n}$ $\quad$ $n * c + n = n$

ANS ⇝ b

Q-

a) $O(f(n)) + O(g(n)) = O(f(n) + g(n))$ ✓
b) $O(f(n)) * O(g(n)) = O(f(n) * g(n))$ ✓
c) $n^2 + 6n + n^3 = O(n^4)$ ✓
d) $n^2 + 6n + n^3 = \Theta(n^4)$ ✓

a) $O(n) + O(n^2)$ $\quad$ $O(n + n^2)$ $O(n^2)$

$O(n^2)$ $\quad$ Yes, True

b) $O(n) * O(n^2)$ $O(n * n^2)$

$O(n^3) \equiv O(n^3)$

imp. log properties

(1) $\log_2 10 = \dfrac{\log_{10} 10}{\log_{10} 2}$

(2) $\log_x 4 = \dfrac{\log_e 4}{\log_e x}$

$$\boxed{\log_{10} n = \frac{\log_2 n}{\log_2 10} = \frac{\log_e n}{\log_e 10}}$$

$\log_{10} n = \Theta(\log_2 n) = \Theta(\log_e n) = \Theta(\log_{50} n) \ldots$

a)
b)
c)
d)
e)
f)
g

Asymptotic notations ke andr hume log ka base lagane ki koye jrurat nhi pdte bcz hum ek base se dusre base mai change krdete hai constant se divide krke

Date	
Page No.	

a) $O(n)$ b) $\Omega(n)$
c) $\Theta(n)$ d) $\omega(n)$

Growth of func^n

	f(n)	g(n)
a)	n	n^2
b)	$\sqrt{n}$	n
c)	$\log n$	n
d)	$\log(\log N)$	$\log N$
e)	n	$n\log N$
f)	n	2^n
g)	n^{1000}	2^n

we can do this changement

$< \Rightarrow \leq$

$> \Rightarrow \geq$

$o \Rightarrow O$

$\omega \Rightarrow \Omega$

but not

$\leq \not\Rightarrow <$

$\geq \not\Rightarrow >$

$O \not\Rightarrow o$

ANSWERS

a) $f(n) = o(g(n))$
$f(n) = O\, g(n)$
$g(n) = \omega f(n) \Rightarrow g(n) = \Omega f(n)$

b) $f(n) = o(g(n))$
$f(n) = O(g(n))$
$g(n) = \omega f(n) \Rightarrow g(n) = \Omega f(n)$

c) Same like option a & b

d) Same like option a & b

e) Same like option a & b

f) Same like option a & b

g) Same like option a & b

n^{1000} 2^n

let $n = 2^{15}$

$(2^{15})^{1000}$ $2^{2^{15}}$

2^{15000} 2^{32768} ✓

★★★ $n^{finite} <<<< 2^n$

$\frac{\log_e n}{\log_e 10}$

Date: 52:43
Page No.

$n >>> \log n$
$n^2 >>> \log n$
$n^3 >>> \log n$
$n^{1/2} >> \log n$

Positive no:
(1) $n > \log n$
(2) $(\log n)^{\text{finite}} < n$

① $f(n)$ $g(n)$
$\sqrt{n}$ $\log n$

$\rightsquigarrow$ $f(n) = \omega(g(n)) \Rightarrow f(n) = \Omega(g(n))$
$g(n) = o(f(n)) \Rightarrow g(n) = O(f(n))$

② $f(n)$ $g(n)$
$\sqrt{\log n}$ $\log(\log n)$

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

$\rightsquigarrow$ $f(n) = \omega(g(n)) \Rightarrow f(n) = \Omega(g(n))$
$g(n) = o(f(n)) \Rightarrow g(n) = O(f(n))$

③ $f(n)$ $g(n)$
2^n $<<$ $n!$

$\rightsquigarrow$ $f(n) = O(g(n)) \Leftarrow f(n) = o(g(n))$
$g(n) = \omega(f(n)) \Rightarrow g(n) = \Omega(f(n))$

④ $f(n)$ $g(n)$
$n!$ $<<$ n^n

$n * (n-1) * (n-2) \ldots$
$n * n * n * n$

Same result like 3rd

⑤ $f(n)$ $g(n)$
2^{n+1} 2^n

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

$f(n) = \Theta(g(n))$
$\hookrightarrow$ $f(n) = O(g(n))$
$f(n) = \Omega(g(n))$

Page No.

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

⑥ $2^{2n} >> 2^n$

2^{2*n} 2^n

$(2^2)^n$ 2^n

$4^n >>> 2^n$

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

$g(n) = o(f(n)) \Rightarrow g(n) = O(f(n))$

⑦ $f(n)$ $g(n)$

$(n+k)^m$ n^m where k, m are constants

$k=2, m=3$

$(n+2)^3$ n^3

$\cancel{n^3} + \ldots$ $\cancel{n^3}$

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

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

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

⑧ $f(n)$ $g(n)$

$2^{n^2} >> n!$

Same like result 6

Q. $f(n) = 10n\log n$

$g(n) = 0.0001n^2$

let $f(n) < g(n)$ $\forall$ $n \geq 10^x$. Find smallest value of x

Ans- $f(n) = o(g(n)) \Rightarrow f(n) < \underset{1}{c} \cdot g(n)$ $\forall$ $n \geq 10^x$

$10 * 10^x \log 10^x < 0.0001(10^x)^2$

$x * 10^{x+1} < 10^{-4}(10^{2x})$

$x * 10^{x+1} < 10^{2x-4}$

$x < \frac{10^{2x-4}}{10^{x+1}}$

$x < 10^{(2x-4)-(x+1)}$

$x < 10^{x-5}$

using hit and trial

$\boxed{x=6}$

Tip: Jab dono taraf log mai comparison ho toh 2 ki power lagake try kro

$(1.001)^n >>> n^{1000000}$

Q- $f(n) << g(n)$

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

$<$ let $n = 2^{10}$ $\quad 10^{\log_2 2^{10}}$

$2^{10} \qquad (\log_2 2^{10})^{\log_2 2^{10}}$

$2^{10} \qquad 10^{10}$

$f(n) = o(g(n)) \Rightarrow f(n) = O(g(n))$

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

Q- $f(n) \qquad g(n)$

$n^{\sqrt{n}} \qquad (\log n)^n$

let $n = 2^{10}$

Same result as above

$(2^{10})^{(2^{10})^{1/2}} \qquad (\log_2 2^{10})^{2^{10}}$

$(1024)^{32} << (10)^{1024}$

Q- $f_1(n) = n^2$

$f_2(n) = n\log n$

$f_3(n) = n\sqrt{n}$

$f_4(n) = e^n$

$f_5(n) = n$

$f_6(n) = 2^n$

$f_7(n) = 1/n$

$e^n > 2^n > n^2 > n\sqrt{n} > n\log n > n > \frac{1}{n}$ Ans

arrange them

Q- $f_1(n) = 2^n$

$f_2(n) = n\log n$

$f_3(n) = n^{3/2}$

$f_4(n) = n^{\log n}$

$2^n > n^{\log n} > n^{3/2} > n\log n$

$2^n \qquad n^{\log n}$

take log both side

$n\log_2 2 > \log n \log n$

$\{ \log(n!) = \Theta(n\log n)$

Date	
Page No.	

ake try KTO

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

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

Q- $f_1(n) = \log n$

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

$f_3(n) = \sqrt{n}$

$f_4(n) = n$

arrange

$\log n < \sqrt{n} < n < (\log n)^{\log n}$

$n \qquad (\log n)^{\log n} \sim \text{let } n = 2^{10}$

$2^{10} << 10^{10}$

Q- $f_1(n) = n^n$

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

$f_3(n) = n^{\sqrt{n}}$

$f_4(n) = (\log n)^n$

$f_5(n) = n!$

$f_6(n) = 2^n$

$f_7(n) = \log(n!)$

$f_8(n) = n\log n$

as above

$n^n > n! > (\log n)^n > 2^n > n^{\sqrt{n}} > n^{\log n} > \log(n!)$

★★★★

Q- if $a<b$ then $f(n^a) = O f(n^b)$ where $a, b \geq 1$

Ans- let $f(n) = \frac{1}{n}$

$gn > n > \frac{1}{n}$

$f(n) = 3n+2$

$f(n^2) = 3n^2+2$

$f(n^a) = 3n^a+2$

$f(n^a) = \frac{1}{n^a}$, $f(n^b) = \frac{1}{n^b}$

$a < b$

$\frac{1}{n^a} > \frac{1}{n^b}$

~~but~~

so $f(n^a) \neq O f(n^b)$

$n! = O(n^n)$

Date
Page No.

(2) Polynomial bounded function

Given any funcⁿ $f(n)$. if we have any polynomial funcⁿ greater than or equal to $f(n)$ then we say $f(n)$ is polynomial bounded.

every polynomial funcⁿ is polynomial bounded funcⁿ

n is bounded by n^2

Polynomial Fⁿ → P.B.F

P F → P.B.F (one way)

~PF → ~PBF ⇝ False

PBF → PF ⇝ False

~PBF → ~PF ⇝ True

funcⁿ	Polynomial Bounded funcⁿ
n	n^2
n^2	n^3
n^5+n^2+1	n^6
$6n^3+4n+5$	n^4
$n\log n$	n^2
$\log n$	n
$n\log(\log n)$	n^2
$n^2\log n$	n^3
2^n	×
$n^{\log n}$	×
n^n	×

not a polynomial funcⁿ but still bounded

$f(n)$ is polynomially bounded iff

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

Q- which of the funcⁿ are polynomially bounded

a) $n!$ → $O(n\log n)$ × No

b) $\log(n!)$ → $O(n\log n) \approx n\log n \Rightarrow \log n + \log\log = O(\log n)$ yes

c) $(\log n)!$ → ×

d) $\log(\log n)!$ → yes lee agar $\log(n!)$ aarha toh ye bhi ayega

e) $(\log(\log n))!$ → yes

use above formula

Note : $(n!)$ is the perfect example of a func^n which is not polynomially bounded but $\log(f(n))$ is polynomially bounded

Date	
Page No.	

Q- which of the following are True

(1) $f(n) = O\, f(n/2)$

let $f(n) = 4^n$ — False

$f(n) = 4^n$, $f(n/2) = 4^{n/2} = (4^n)^{1/2} = (2^{2n})^{1/2} = 2^n$

$f(n) \neq O(f(n/2))$

(2) $f(n) = O\, f(n^2)$

let $f(n) = \frac{1}{n}$ — False

$f(n^2) = \frac{1}{n^2}$ $\quad \frac{1}{n} > \frac{1}{n^2}$

$f(n) \neq O(f(n^2))$

(3) $f(n) = O(f(n)^2)$

let $f(n) = \frac{1}{n}$ — False

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

$f(n) \neq O((f(n))^2)$

(4) if $f(n) = O(g(n))$ then $2^{f(n)} = O(2^{g(n)})$

Ans- Let $f(n) = n$ $g(n) = n/2$ — False

$2^n \neq O(2^{n/2})$

(5) $1000^* n\log n = O\left(\frac{n\log n}{1000}\right)$ — True

ignore constant

Q-

$$f(n) = \begin{cases} n^4 & 0 < n < 1000 \\ n^2 & n \geq 1000 \end{cases}$$

$$g(n) = \begin{cases} n' & 0 < n < 100 \\ n^3 & n \geq 100 \end{cases}$$

as we know, we calculate for big terms

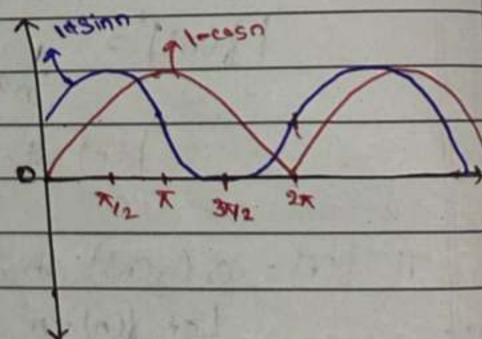
$f(n) = n^2$
$g(n) = n^3$

$f(n) = o(g(n)) \Rightarrow f(n) = O(g(n))$
$g(n) = \omega(f(n)) \Rightarrow g(n) = \Omega(f(n))$

Q-

$f(n) = n^{\overset{-1 \text{ to } +1}{1+\sin n}} \rightsquigarrow n^0$ to n^2

$g(n) = n^{\underset{-1 \text{ to } 1}{1-\cos n}} \rightsquigarrow n^0$ to n^2



we can't conclude any notation here bcz it is changing

Q-

$f(n) = n^{\sin n}$ ↗ -1 to $+1$

$g(n) = n^{2+\cos n}$ ↗ -1 to $+1$.

$f(n) \rightsquigarrow n^{-1}$ to n'

$g(n) \rightsquigarrow n'$ to n^3

$f(0)=0$	$g(0)=3$
$f(\pi/2)=1$	$g(\pi/2)=2$
$f(\pi)=0$	$g(\pi)=1$
$f(3\pi/2)=-1$	$g(3\pi/2)=2$
$f(2\pi)=0$	$g(2\pi)=3$

$f(n) = O(g(n)) \leftarrow f(n) = o(g(n))$
$g(n) = \omega(f(n)) \rightsquigarrow g(n) = \Omega(f(n))$

as we can see g(n) is always greater than f(n) but when we look at quesn it seems they are equal at some point

Note: Jab bhi compare krte time thoda bhi doubt hai toh value leke compare kro kro.

Date	
Page No.	

Q- $f(n) = n!$
$g(n) = (n-1)!$
$h(n) = (n-1)! + n$

$n! = n * (n-1)!$

$g(n) = o f(n) \Rightarrow g(n) = O(f(n))$

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

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

become equal to g(n) bcz of negotiating constant

$h(n) = g(n) < f(n)$

Q- $f(n) = n!$
$f(n) = (\log n)^{\log n}$
$f(n) = n^{\log n}$ $\rightarrow n^{\log n} > (\log n)^{\log n}$
$f(n) = \log(n!)$
$f(n) = \sqrt{n}$
$f(n) = (\log(\log n))!$
$f(n) = (\log n)!$ $\rightarrow (\log n)! > (\log(\log n))!$
$f(n) = 2^n$
$f(n) = n^{\sqrt{n}}$ $\rightarrow 2^n > n^{\sqrt{n}}$

$\rightsquigarrow$ n! is biggest

$n! > 2^n > n^{\sqrt{n}}$

ANS $\rightsquigarrow$ $n! > 2^n > n^{\sqrt{n}} > n^{\log n} > (\log n)^{\log n} > (\log n)! > \log(n!) > \sqrt{n} > (\log(\log n))!$

Q- $f(n) = \sqrt{n}$ $\quad g(n) = n \log n$

Same part ko uda do

$\sqrt{n} \times 1$ $\quad \sqrt{n} \times \sqrt{n} \times \log N$

$\sqrt{n} \log n >> 1$

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

$f(n) = o(g(n)) \Rightarrow f(n) = O(g(n))$
$g(n) = \omega(f(n)) \Rightarrow g(n) = \Omega(f(n))$

★★★ Note: first remove common part then take log