ch - 1, 2, 3, 4. (T. H. cormen). [20 marks].

Algorithms :- finite set of instructions to accomplish a task.
The algorithm should be correct.
The Properties of an algorithm are as follows :-

1) finiteness (having an end or limit)
2) Input
3) output
4) definiteness (clear, fixed or certain)
5) effectiveness (Power to produce effective result).
6) correctness (produce correct result)
7) Generality :- applicable for all problems of the desired form, not just for a particular set of input values.

8. Algo to find GCD of two nos.

1. Begin.
2. Get two nos m & n.
3. Divide m by n.
4. If remainder == 0 return n as the GCD.
5. else
6. m ← n, n ← remainder.
7. Go to step 3.
8. Exit.

Life cycle of an Algorithm.

1) Design the Algorithm.
2) write (Implementation of the Algorithm).
3) Test the Algorithm.
4) Analyze the Algorithm.

Analysis of Algorithms

An algorithm when implemented, uses the computer's primary memory & central Processing unit. Analyzing the amount of resources needed for a particular solution of the Problem. The analysis is done at two stages :-

1) Priori Analysis :- Analysis done before implementation.

2) Posteriori Analysis :- Analysis is done after implementation.

Analysis of Algorithm refers to predicting the resources required by the Algorithm, based on size of the problem. The primary resources are Time and space.

Analysis based on Time taken to execute the algorithm is called Time Complexity of the Algorithm.

Analysis based on memory required to execute the algorithm is called space complexity of the algorithm.

Ex:- consider the code below :-

```
SumAll(array, n)
1.  Sum = 0
2.  for (i=0 ; i<n ; i++)
        2.1  sum = sum + array[i] ;
3.  return Sum
```

Time taken:

(initialization) 1 + (checking) n + (extra checking) 1 + (increments) n + n + 1 = 3n + 4

Basic mathematics

$$A.P = \sum_{i=0}^{n} i = 1+2+3+\ldots+n = \frac{n(n+1)}{2}$$

Logarithms

(1) $a = b^{\log_b a}$

(2) $\log_b a = (\log_c a)(\log_b c)$

(3) $\log_b a = \frac{1}{\log_a b}$

(4) $\log_b a = \frac{\log_c a}{\log_c b}$

(5) $\lg n = \log_2 n$ (Binary logs)

(6) $\ln n = \log_e n$ (natural logs)

(7) $\lg^k n = (\lg n)^k$ (exponentiation)

(8) $\lg \lg n = \lg(\lg n)$ (composition)

(9) $a = b^{\log_b a}$

(10) $\log_b a^n = n \log_b a$

(11) $\log_b a = \frac{\log_c a}{\log_c b}$.

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

(13) $\log_b\left(\frac{1}{a}\right) = -\log_b a$

(14) $\log_b a = \frac{1}{\log_a b}$

(15) $a^{\log_b c} = c^{\log_b a}$.

(16) $\log(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots$

(17) $\log_2 2 = 1$

(18) $\log_2 4 = 2$.

$1^2 + 2^2 + 3^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6}$

$1 + a + a^2 + \ldots + a^n = \frac{a^{n+1} - 1}{a - 1}$.

${}^nC_r = \frac{n!}{r!(n-r)!}$

floors ($\lfloor \; \rfloor$)

$\lfloor x \rfloor$ = for any real no 'x', we denote the greatest integer less than or equal to x by $\lfloor x \rfloor$.

eg:- $\lfloor 6 \rfloor = 6$

$\lfloor 6.1 \rfloor = 6$

$\lfloor 6.9 \rfloor = 6$

= ceilings ($\lceil \; \rceil$)

$\lceil x \rceil$:- Least integer greater than or equal to x.

≡ eg $\lceil 6 \rceil = 6$, $\lceil 6.1 \rceil = 7$, $\lceil 6.9 \rceil = 7$.

≠ Modular Arithmetic

$a \bmod n = a - \lfloor a/n \rfloor n$.

eg: $7 \,\%\, 2 = 7 - \lfloor 7/2 \rfloor . 2$

$= 7 - 3.2$

$= 7 - 6 = 1$.

exponentials

$a^0 = 1$, $(a^m)^n = a^{mn}$

$a^1 = 1$, $(a^m)^n = (a^n)^m$

$a^{-1} = \frac{1}{a}$, $a^m \cdot a^n = a^{m+n}$,

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

factorials

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

Thus $n! = 1.2.3.\ldots n.$

fibonacci Numbers :- are defined by the foll. recurrances

$f_0 = 0$, $f_1 = 1$, $f_i = f_{i-1} + f_{i-2}$ for $i \geq 2$

seq:- 0 1 1 2 3 5 8 13 ----

Analysis based on the nature of the Problem:-

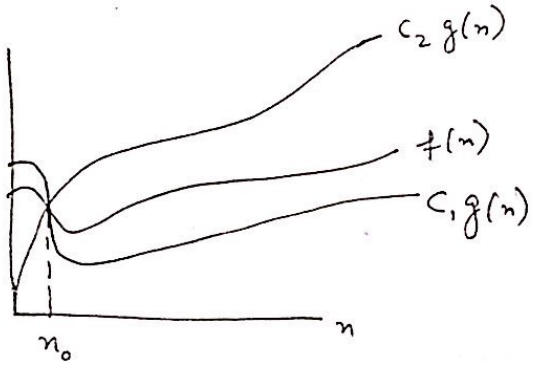
Worst case # Average case # Best case.

In sorting the array, the worst case is in which array is reversed & best case is in which input array is already sorted. we usually concentrate on worst case running time that is longest running time for any input of size n. The worst case running time of an algo is an upper bound on the running time for any input. Knowing it gives us a guarantee that the algorithm will never take any longer.

Asymptotic Notations

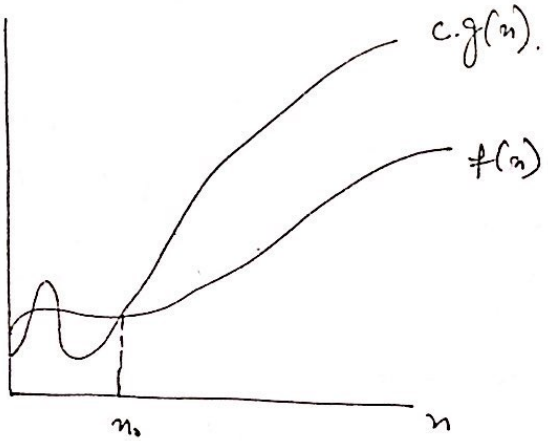
⇒ **Θ Notation** :- $f(n) = \Theta(g(n))$ [Theta Notation]

$\Theta(g(n)) = \{ f(n) :$ there exist positive constants c_1, c_2 & n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \; \forall \; n \geq n_0 \}$



(II) **O-Notation** :- $f(n) = O(g(n))$ [Big-oh Notation]

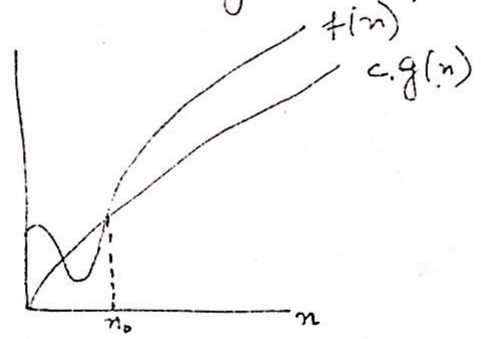
$O(g(n)) = \{ f(n)$: there exist some +tive constants c & n_0 such that $0 \leq f(n) \leq c.g(n) \; \forall \; n \geq n_0$



$f(n)$ grows no faster than $g(n)$.

(II). Ω Notation. $\therefore f(n) = \Omega(g(n))$.

$\Omega(g(n)) = \{ f(n)$: there exist Positive constants c & n_0 such that $0 \leq c.g(n) \leq f(n) \quad \forall \; n \geq n_0$



It means $f(n)$ grows atleast at fast as $g(n)$.

(I) o Notation (small-oh notation)

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

$o(g(n)) = \{ f(n)$: for any +tive constant $c > 0$, there exist a constant $n_0 > 0$ such that

$0 \leq f(n) < c.g(n) \quad \forall \; n \geq n_0$.

The relation $f(n) = o(g(n))$ implies that,

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

(III) ω Notation (small omega Notation).

$\omega(g(n)) = \{ f(n)$: for any +tive constant $c > 0$, there exist a constant $n_0 > 0$ such that $0 \leq c.g(n) < f(n) \quad \forall \; n \geq n_0$.

The relation $f(n) = \omega(g(n))$ implies that,

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

Θ is like $=$

O is like $\leq$

Ω is like $\geq$

o is like $<$

ω is like $>$

The def^n of O-Notation & o-Notation are similar. The main difference is that in $f(n) = O(g(n))$, the bound $0 \leq f(n) \leq c.g(n)$ holds for <u>some constant</u> $c > 0$, but, in $f(n) = o(g(n))$, the bound $0 \leq f(n) < c.g(n)$ holds for <u>all constants</u> $c > 0$.

eg: Is $2n = o(n^2)$?

$0 \leq 2n < cn^2$, Div by n^2,

$\frac{2}{n} < c$ or $n > \frac{2}{c}$

It should be true for all + values of c ∴ take $n = 3$ & bigger values

∴ $2n = o(n^2)$.

$\lim_{n \to \infty} \frac{2n}{n^2} = \frac{2}{n} = \frac{2}{\infty} = 0$

$2n^2 \neq o(n^2)$

$0 \leq 2n^2 < cn^2$

True for all values of $c > 2$ but not for 1 & 2.

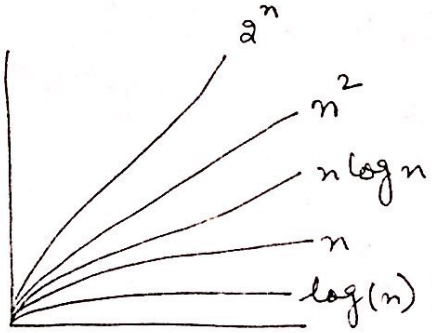
∴ $2n^2 \neq o(n^2)$.

$\lim_{n \to \infty} \frac{2n^2}{n^2} = 2 \neq 0$

∴ $2n^2 \neq o(n^2)$

Growth of functions

Algorithm complexity will be represented in terms of mathematical functions.



Comparison of functions

> Transitivity :- $f(n) = \Theta(g(n))$ & $g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$.

$f(n) = O(g(n))$ & $g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$.

$f(n) = \Omega(g(n))$ & $g(n) = \Omega(h(n)) \Rightarrow f(n) = \Omega(h(n))$.

$f(n) = o(g(n))$ & $g(n) = o(h(n)) \Rightarrow f(n) = o(h(n))$.

$f(n) = \omega(g(n))$ & $g(n) = \omega(h(n)) \Rightarrow f(n) = \omega(h(n))$.

> Reflexivity :- If $f(n) = \Theta(f(n))$, then

$f(n) = O(f(n))$ & $f(n) = \Omega(f(n))$.

> symmetry :- $f(n) = \Theta(g(n))$ iff $g(n) = \Theta(f(n))$.

> Transpose symmetry :-

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

$f(n) = o(g(n))$ iff $g(n) = \omega(f(n))$.

Analysis of Algorithms

Bubble Sort

```
for (i = n-1 ; i >=1 ; i--)
 {
   for (j = 1; j <= i; j++)
    {
      If (a[j] >= a[j+1])
       {
          temp = a[j];
          a[j] = a[j+1];
          a[j+1] = temp;
       }
    }
 }
```

steps:-

1) start with the 1st element.
2) compare subsequent elements & push the larger one to the back.

S.No.	Index	elements	I st iter.	II nd iter.	III rd iter.	IV th iter.
	1	50	40	40	30	25
	2	40	50	30	25	30
	3	60	30	25	40	40
	4	30	25	50	50	50
	5	25	60	60	60	60

Analysis:

No of elements		comparisons
5	⟶	4
4	⟶	3
3	⟶	2
$\therefore n$	⟶	$(n-1)$

$\therefore$ Total no. of comparisons $= 1 + 2 + \dots + (n-1) = \frac{n(n+1)}{2} - n = \frac{n^2-n}{2} = O(n^2)$.

In all three cases, the no. of comparisons will be same, $\therefore$ we can also write $\Omega(n^2)$. [$\because$ Because min. no. of comparisons are also (n^2) $\therefore$ $T(n) = \Theta(n^2)$ [In all three cases].

Selection Sort

```
for (i=1 ; i<=n-1 ; i++)
{
    min = i;
    for (j = i+1 ; j <= n ; j++)
    {
        If (a[j] <= a[min])
            min = j;
    }
    If (i != min)
        swap a[i] && a[min]
}
```

Steps

1) consider first element as the smallest element.
2) compare it with rest of the array.
3) find min. element from rest of the array & swap it with min. elt. chosen.

idx.	elements	I st iter.	II nd iter	III rd iter	IV th iter
1	[50]	25	25	25	25
2	40	[40]	30	30	30
3	60	60	[60]	40	40
4	30	30	40	[40]	50
5	25	50	50	50	60

Analysis :

No. of elements		comparisons
5	→	4
4	→	3
3	→	2
⋮		
2	→	1
n	→	n-1.

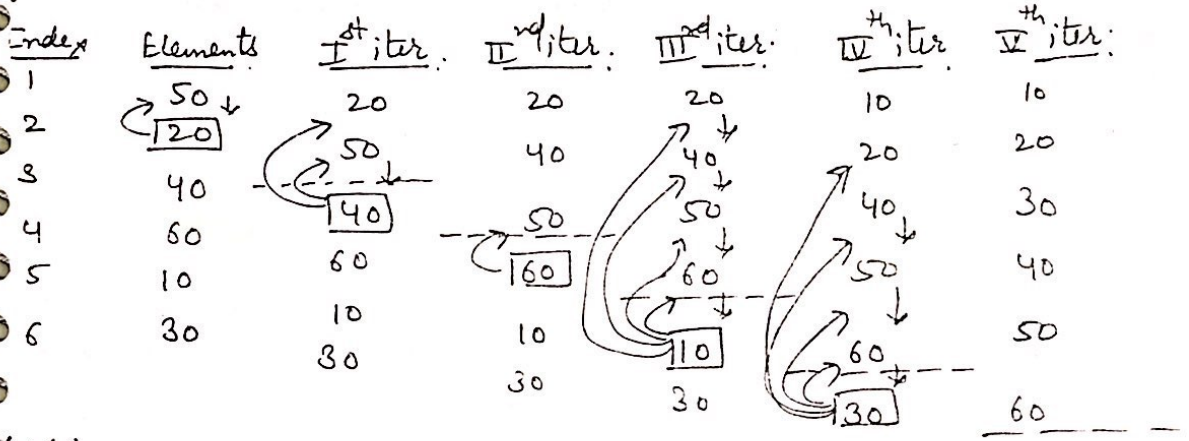
∴ Total no. of comparisons are :-

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

Minimum no. of comparisons are also $\Omega(n^2)$, no matter what the case is. ∴ $T(n) = \theta(n^2)$ [in all the three cases].

Insertion sort

```
for (i = 2 ; i <= n ; i++)
{
    v = a[i];
    j = i-1;
    while ((j >= 1) && v <= a[j])
    {
        a[j+1] = a[j];
        j = j - 1;
    }
    a[j+1] = v;
}
```



Logic :-

⇒ start with 2nd element & compare with above elements until u find an elements ie less than element chosen.

⇒ write the element on the position u find & shift all elements down.

The array is partially sorted after each iteration.

Analysis :-

⇒ In the worst case eg :- 50 40 30 20 no. of comparisons will be

	elements		comparisons
I st iter →	2	→	1
II nd iter →	3	→	2
ie	n	→	n-1.

or Total no. of comparisons = $1 + 2 + 3 + \ldots + (n-1)$

$= O(n^2)$.

[Maximum no. of comparisons]

2) In the best case, for eg :-

10
20 , the no. of comparisons are
30
40
50.

	elements	comparisons
Ist iteration	2	1
IInd iteration	3	1
IIIrd iteration	4	1
ie	n	(n-1) [Total no. of comparisons].

$T(n) = \underset{O(n)}{\Omega}(n)$ [min. no. of comparisons to sort sorted array].

[Also $T(n) = \Theta(n) \rightarrow$ Best case].

3). In average case, we consider the upperbound, to find maximum time taken to sort n elements.

$\therefore T(n) = O(n^2)$.

$\therefore$ Insertion sort lies between $\Omega(n)$ to $O(n^2)$.

∴ Insertion sort Running time

$T(n) \neq \Omega(n^2)$ [Because it can be $\Omega(n)$ & $O(n)$ in the Best case] ie $T(n) = \Theta(n)$.

$T(n) = \Omega(n^2)$ or $O(n^2)$ ie $\Theta(n^2)$ in worst case.

[min no. & max no.] of comparisons will be n^2 in worst case

$\therefore \Theta(n^2)$.

Insertion sort Analysis [on the basis of code]

	cost	times
`for(i=2; i<=n; i++)`	c_1	n
`{`		
`v = a[i];`	c_2	$n-1$
`j = i-1;`	c_3	$n-1$
`while (j>=1 && v<=a[j])`	c_4	$\sum_{i=2}^{n} t_i$
`{`		
`a[j+1] = a[j];`	c_5	$\sum_{i=2}^{n} t_i - 1$
`j = j-1;`	c_6	$\sum_{i=2}^{n} t_i - 1$
`}`		
`a[j+1] = v;`	c_7	$n-1$
`}`		

$$\therefore T(n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4 \sum_{i=2}^{n} t_i + c_5 \sum_{i=2}^{n} (t_i - 1) + c_6 \sum_{i=2}^{n} (t_i - 1) + c_7 (n-1).$$

Best case :- In the Best case, while loop is not entered as $v \le a[j]$ will be false, so the line get executed for $(n-1)$ times as other for loop lines:-

$$\therefore c_1 n + c_2(n-1) + c_3(n-1) + c_4(n-1) + c_7(n-1)$$
$$= (c_1 + c_2 + c_3 + c_4 + c_7) n - (c_2 + c_3 + c_4 + c_7)$$
$$= an + b \text{ [Linear function]} = \Omega(n).$$

Worst case :- In the worst case, we must compare each element $a[i]$ with each element in the entire sorted subarray $a[1 \ldots i-1]$ & so $t_i = i$ for $i = 2, 3, \ldots n$. Also,

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

$$\therefore c_1 n + c_2(n-1) + c_3(n-1) + c_4\left(\frac{n(n+1)}{2} - 1\right) + c_5\left(\frac{n(n-1)}{2}\right) + c_6\left(\frac{n(n-1)}{2}\right) + c_7(n-1).$$

[polynomial of n].

$\Rightarrow (\quad) n^2 + (\quad) n + (\quad) = an^2 + bn + c = O(n^2)$

Linear search

To find which element [if any] of the given array a[1..n] equals the target element.

```
for (i=1 ; i<=n ; i++)
    {
      If (target = a[i])
        {
          cout << " Position: " << i;
          break;
        }
    }
```

Analysis :-

Best case :- O(1)
Worst case :- O(n)
Average case :- O(n).

Binary search :-

To find which element [if any] of the given array [1..n] equals the target element.

```
left = 1; right = n;
while ( left <= right) do
    {
      m = floor ((left + right)/2);
      if (target == a[m] end with o/p as m;
      if ( target < a[m]) then right = m-1 ;
      if ( target > a[m]) then left = m+1 ;
    }.
```

condition :- The array should be sorted array (to be searched)

Analysis.

Time to divide the array into 2 subarrays + 1 checking.

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

$$T\left(\frac{n}{2}\right) = T\left(\frac{n}{4}\right) + 1$$

$$T\left(\frac{n}{4}\right) = T\left(\frac{n}{8}\right) + 1$$

$$\vdots$$

$$T(4) = T(2) + 1$$

$$T(2) = T(1) + 1.$$

Backtrack,

$$T(n) = T(1) + \log n$$

$$T(n) = 1 + \log n$$

$$= O(\log n)$$

eg. $T(8) = T(4) + 1$

$T(4) = T(2) + 1$

$T(2) = T(1) + 1$

Backtrack

$T(8) = T(1) + 3$

$= T(1) + \log_2 8$ (no. of ele.) $\Rightarrow n$

$= 1 + \log 8$

$= O(\log 8).$

or

$O(\log n)$ for n terms.

Q. find the values of c & n_0.

1) $f(n) = 7n+8$, $g(n) = n$. Is $f(n) = O(g(n))$?

Sol^n.

$$0 \leq f(n) \leq c \cdot g(n).$$

i.e. $f(n)$ grows no faster than $g(n)$.

$\therefore$ $7n+8 \leq n$ Take $C = 15$

$n=1$, $7+8 \leq 1$, $15 \leq 1.15$ $\therefore$ $15 = 15$.

$n=2$, $14+8 \leq 2$, $22 \leq 2$, $22 \leq 2.15$ or $22 < 30$.

$\therefore$ $n_0 = 1$, $C = 15$.

OR

$f(n) = 7n+8$

$\leq 7n+8n$

$\leq 15n$.

$\leq c \cdot g(n)$. $\therefore$ $C = 15$, $n_0 = 1$.

OR

$0 \leq f(n) \leq c.g(n)$

$7n+8 \leq c.n$, Div. by n, we get,

$7 + \frac{8}{n} \leq c$, Put $n=1$, $C=15$, the equality becomes True, i.e.

$7 + \frac{8}{1} \leq 15$, $(15 = 15)$

$\therefore$ $n_0 = 1$, $C = 15$

(2) $f(n) = 10n + 5$, $g(n) = n$. Is $f(n) = O(g(n))$.

$0 \leq 10n + 5 \leq c.n$, Div by n,

$10 + \frac{5}{n} \leq c$, $\boxed{c = 15,\ n_0 = 1.}$

$15 = 15.$

(3) $f(n) = 3n^2 + 4n + 1$, $g(n) = n^2$? Is $f(n) = O(g(n))$?

	$f(n)$	$g(n)$
$n = 1$	8	$1.8(c)$
$n = 2$	$12 + 8 + 1 = 21$	$4.8 = 32$

or

$3n^2 + 4n + 1 \leq c.n^2$, Div by n^2,

$3 + \frac{4}{n} + \frac{1}{n^2} \leq c$

Put $n = 1$,

$3 + 4 + 1 \leq 8.$

$\therefore$ $\boxed{c = 8,\ n_0 = 1}$

(4) $f(n) = n^3 + 20n + 1$, $g(n) = n^3$

$n^3 + 20n + 1 \leq cn^3$, Div. by n^3,

$1 + \frac{20}{n^2} + \frac{1}{n^3} \leq c$,

Put $n = 1$

$1 + 20 + 1 \leq 22$

$\therefore$ $\boxed{c = 22,\ n_0 = 1}$

5. $f(n) = n^2$, $g(n) = n^2 - n$. Is $f(n) = O(g(n))$?

$$f(n) \leq c \cdot g(n).$$

$$n^2 \leq c(n^2 - n), \text{ Div by } n^2,$$

$$1 \leq c\left(1 - \frac{1}{n}\right)$$

Put $n = 1$, $1 \leq c.0$ (False).

$n = 2$, $1 \leq c\left(1 - \frac{1}{2}\right)$

$1 \leq c\left(\frac{1}{2}\right)$ Put $c = 2$ (True)

$\therefore$ $\boxed{c = 2,\ n = 2}$

6. $f(n) = n^2$, $g(n) = n^2 - n$? Is $g(n) = O(f(n))$?

$$g(n) \leq c f(n).$$

$$n^2 - n \leq c\, n^2, \text{ Div. by } n^2,$$

$1 - \frac{1}{n} \leq c$, Put $n = 1$,

$1 - 1 \leq c$ ie $0 \leq 1$

or:

$$n^2 - n \leq n^2$$

Put $n = 1$, $1 - 1 \leq 1$ ie $0 \leq 1$

$n = 2$, $4 - 2 \leq 4$ ie $2 \leq 4$

$n = 3$, $9 - 3 \leq 9$ ie $6 \leq 9$. True;

$\therefore$ $\boxed{n = 1,\ C = 1}$

2) $f(n) = n^3$, $g(n) = n^3 - n^2 - n$? Is $f(n) = O(g(n))$?

$$n^3 \leq n^3 - n^2 - n$$

Put $n=1$: $1 \leq -1$ (False).

$n=2$: $8 \leq (2).\underset{\uparrow c}{4}$ (True).

$\therefore$ $\boxed{n=2,\ C=4}$

$n=3$: $27 \not\leq (27-9-3).4 = 60$

3) $f(n) = n^3$, $g(n) = n^3 - n^2 - n$? Is $g(n) = O(f(n))$?

$$n^3 - n^2 - n \leq n^3$$

Put $n=1$, $-1 \leq 1$ (True)

$n=2$, $2 \leq 8$ (True)

$\therefore$ $\boxed{n=1,\ C=1}$

1) $f(n) = n^2$, $g(n) = n^2 + n$. Is $f(n) = O(g(n))$?

$$n^2 \leq n^2 + n$$

Put $n=1$, $1 \leq 2$

$n=2$, $4 \leq 6$

$n=3$, $9 \leq 12$

$n=4$, $16 \leq 20$ (True)

$\therefore$ $\boxed{n=1,\ C=1}$

2) $f(n) = n^2$, $g(n) = n^2 + n$. Is $f(n) = \Omega(g(n))$?

$$n^2 + n \leq n^2 \qquad (c.g(n) \leq f(n))$$

Put $n=1$, $\frac{1}{2}.2 \leq 1$ (True) (Take $c = \frac{1}{2}$)

$n=2$, $\frac{1}{2}.6 \leq 4$ (True)

$n=3$, $\frac{1}{2}.12 \leq 9$ (True)

$\boxed{C=\frac{1}{2},\ n=1}$

$0 \leq c(n^2+n) \leq n^2$, Div. by n^2,

$c\left(1+\frac{1}{n}\right) \leq 1$, Put $n=1$,

$c=\frac{1}{2}$

$\boxed{C=\frac{1}{2},\ n=1}$

(11) $f(n) = n^3$, $g(n) = n^3 + 4n^2 - 5n$. Is $f(n) = \Omega(g(n))$?

$$n^3 + 4n^2 - 5n \leq n^3$$

Put $n=1$, $1 + 4 - 5 \leq 1$ ie $\frac{1}{2} \cdot 0 \leq 1$ ie $0 \leq 1$

$n=2$, $8 + 16 - 10 \leq 8$ ie $\frac{1}{2} \cdot 14 \leq 8$ ie $7 \leq 8$ $c = \frac{7}{7}$

$n=3$, $27 + 36 - 15 \leq 27$ ie $\frac{1}{2} \cdot 48 \leq 27$ ie $24 \leq 27$ (True)

Take $c = \frac{1}{2}$, we get, $n=1$, $c = \frac{4}{11}$ $c = \frac{1}{2}$ $c > 0$

(10) $f(n) = n^3$, $g(n) = n^3 + 4n^2 - 5n$. Is $g(n) = \Omega(f(n))$?

$$c \cdot n^3 \leq n^3 + 4n^2 - 5n$$

Put $n=1$ $c \cdot 1 \leq 0$

$n=2$, $\frac{7}{4} c \cdot 8 \leq 14$ (True) $\Rightarrow 8 \leq 14 \cdot x$

$x = \frac{8}{14} = \frac{4}{7}$

$27 \leq 27 + 36 - 15 = 48 \times \frac{4}{7}$ $\quad 69.$ $\quad 51$ / $63 - 15 = 48$

Take $c=1$, $n=2$

or

$c = \frac{2}{4}$, $n=2$

(9) $2n^2 = O(n^2)$

$2n^2 \leq c \cdot n^2$, Div. by n^2,

$2 \leq c$.

$\therefore$ $c=2$, $n=1$

$2n^2 \leq n^2$

Put $n=1$, $2 \leq 1 \cdot 2$

$n=2$, $8 \leq 4 \cdot 2$

$n=3$, $18 \leq 9 \cdot 2$ (True)

Take $c=2$

$\therefore$ $c=2$, $n=1$

$\frac{8}{14} = \frac{4}{7}$

4) $2n = O(n^2)$

Put $2n \leq n^2$

$n=1$, $2 \leq 1 \cdot 2$

$n=2$, $4 \leq 4 \cdot 2$

$n=3$, $6 \leq 9 \cdot 2$ (True)

Take $C=2$ $\therefore$ $\boxed{n=1,\ C=2}$

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

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

Put $n=1$, $\frac{1}{2} \leq 1 \cdot \frac{1}{2}$

$n=2$, $2 \leq 4 \cdot \frac{1}{2}$

$n=3$, $4.5 \leq 9 \cdot \frac{1}{2}$

$n=4$, $8 \leq 16 \cdot \frac{1}{2}$ (True).

Take $\boxed{C = \frac{1}{2},\ n=1}$

$\frac{n^2}{2} \leq c \cdot n^2$, Div. by n^2,

$\frac{1}{2} \leq c$

$\therefore c = \frac{1}{2}$

6) $2^{n+1} = O(2^n)$

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

Put $n=1$ $4 \leq 2 \cdot 2(c)$

$n=2$ $8 \leq 4 \cdot 2$

$n=3$ $16 \leq 8 \cdot 2$

$n=4$ $32 \leq 16 \cdot 2$.

$\therefore$ $\boxed{C=2,\ n=1}$

$2^{n+1} \leq c \cdot 2^n$

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

$\boxed{C=2,\ n=1}$

(17) $2n^2 = O(n^3)$

$2n^2 \leq n^3$

$n=1$ $\quad 2 \leq 1 \cdot 2^{(c)}$

$n=2$ $\quad 8 \leq 8 \cdot 2$

$n=3$ $\quad 18 \leq 27 \cdot 2$

$\therefore$ $n=1,\ C=2$

$2n^2 \leq C \cdot n^3$, Div by n^3,

$\frac{2}{n} \leq C$

Put $n=1,\ C=2$

(18) $2n^2 = O(n^2)$

$2n^2 \leq n^2$

$n=1$, $\quad 2 \leq 1 \cdot 2^{(c)}$

$n=2$, $\quad 8 \leq 4 \cdot 2$

$n=3$, $\quad 18 \leq 9 \cdot 2$

$n=4$, $\quad 32 \leq 16 \cdot 2$

$\therefore$ $n=1,\ C=2$

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

$C=2,\ n=1$

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

$\log n^2 \leq \log n$

$2 \log n \leq \log n$

$n=1$, $\quad 2.0 \leq 0 \cdot 2^{(c)}$

$n=2$, $\quad 2.1 \leq 1 \cdot 2^{(c)}$

$n=3$, $\quad 2.\log 3 \leq \log 3 \cdot 2^{(c)}$

$C=2,\ n=1$

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

$2 \log n \leq C \log n$

$C=2,\ n=1$

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

at $n^2 - n \leq n^2 + n$

$n=1$, $0 \leq 2.1(c)$

$n=2$, $2 \leq 6.1$

$n=3$, $6 \leq 12.1$

$\therefore$ $\boxed{c=1,\ n=1}$

$n^2 - n \leq c(n^2 + n)$, Div by n^2,

$1 - \frac{1}{n} \leq c\left(1 + \frac{1}{n}\right)$

Put $n=1$

$1 - 1 \leq c(1+1)$

$0 \leq 2.c.$

$\therefore$ $\boxed{c=1,\ n=1}$

(21) $3n^2 - 100n + 6 = O(n^2)$. , $c=109$, $n=1$

$3n^2 - 100n + 6 \leq n^2$

$-91 \leq 1$

$3n^2 - 100n + 6 \leq cn^2$,

$3 - \frac{100}{n} + \frac{6}{n^2} \leq c$

$n=1$

$n=10$ at end.

$=100$

(22) $3n^2 + 8n = O(4n^2)$

$3n^2 + 8n \leq 4n^2$

$n=1$, $3 + 8 = 11 \leq 4.\left(\frac{11}{4}\right)$ $\therefore 11 = 11$

$n=2$, $28 \leq 16.\frac{11}{4}$ $\therefore 28 \leq 44$

$n=3$, $81 \leq 36.\frac{11}{4}$ $\therefore 81 \leq 99$

$n=5$, $115 \leq 100.\frac{11}{4}$ $\therefore 115 \leq 275$

$\therefore$ $\boxed{n=1,\ c=\frac{11}{4}}$

$3n^2 + 8n \leq c.4n^2$, Div. by n^2,

$3 + \frac{8}{n} \leq 4c$

Put $n=1$ $\therefore 3+8 \leq 4.c$

$11 \leq 4$ $\therefore c = \frac{11}{4}$

$f(n) = 3n^2 + 8n$

$f(n) \leq 3n^2 + 8n^2$

$\leq 11n^2$

$\leq 4\left(\frac{11}{4}\right)n^2$ or $4n^2\left(\frac{11}{4}\right)$

$g(n)$ c.

(23). Is $2n = O(n^2)$?

$2n \le cn^2$

$n=1$, $2 \le 2.1$ $\therefore$ $c=2$

$n=2$, $4 \le 2.4$ True

$n=3$, $6 \le 2.9$ True.

$\therefore$ $c=2$, $n \ge 1$

$0 \le f(n) \le c\,g(n)$.

$2n \le cn^2$,

Div by n^2,

$\frac{2}{n} \le c$

Put $n=1$, $c=2$

$f(n) = 2n$

$f(n) \le 2n^2$

$\le c.g(n)$

$\therefore$ $c=2$, $n=1$

(24) $2^{2n} \neq O(2^n)$

$0 \le 2^{2n} \le c\,2^n$

$0 \le 2^n \cdot 2^n \le c \cdot 2^n$

i.e. $2^n \le c$.

But no constant is greater than all 2^n & so the assumption leads to a contradiction.

$2^{2n} \le c.2^n$

$n=1$, $4 \le 2.2$ $\therefore$ $c=2$

$n=2$, $16 \le 2.4$ (False).

$\therefore$ $2^{2n} \neq O(2^n)$.

(25) $f(n) = 2^n$, $g(n) = 3^n$. Is $f(n) = O(g(n))$?

$0 \le 2^n \le c.3^n$

$n=1$, $2 \le 3.\frac{2}{3}$ $\therefore$ $c = \frac{2}{3}$

$n=2$, $4 \le \frac{2}{3}.9$ i.e. $4 \le 6$ True

$n=3$, $8 \le \frac{2}{3}.27$ i.e. $8 \le 18$ True

$\therefore$ $n_0 = 1$, $c = \frac{2}{3}$

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

$0 \le g(n) \le c\,f(n)$

i.e. $0 \le 3^n \le c\,2^n$

$n=1$, $3 \le 2.\frac{3}{2}$

$\therefore$ $c = \frac{3}{2}$

$n=2$, $9 \le \frac{3}{2}.4^2$

i.e. $9 \le 6$ (False)

$\therefore$ $g(n) \neq O(f(n))$.

(26) Is $\frac{n^2}{2} = \omega(n)$?

To check. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$

ie $\lim_{n\to\infty} \frac{n^2}{2} \times \frac{1}{n} = \frac{n}{2} = \frac{\infty}{2} = \infty$

$\therefore \frac{n^2}{2} = \omega(n)$.

$0 \le cn < \frac{n^2}{2}$

or $2c < n$

$\therefore \boxed{n > 2c}$ & $\boxed{c > 0}$

(27) Is $\frac{n^2}{2} = \omega(n^2)$?

To check :- $\lim_{n\to\infty} \frac{n^2}{2} \times \frac{1}{n^2} = \frac{1}{2} \neq \infty$

$\therefore \frac{n^2}{2} \neq \omega(n^2)$.

or suppose. $0 \le c.n^2 < \frac{n^2}{2}$ [True for $\forall\, c > 0$].

$c < \frac{1}{2}$

suppose $c = 1$ $\therefore$ $1 < \frac{1}{2}$ (false)

$\therefore \frac{n^2}{2} \neq \omega(n^2)$.

(28) Is $2n = o(n^2)$? $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$

To check: $\lim_{n\to\infty} \frac{2n}{n^2} = \frac{2}{n} = \frac{2}{\infty} = 0$ (True).

Inequality is $\Rightarrow$ $0 \le 2n < c.n^2$

or $2 < cn$

or $c > \frac{2}{n}$ or $n > \frac{2}{c}$

~~$0 \le 2n < cn^2$~~

~~$n \in k, \ldots \varepsilon \frac{2}{1} 1^2$ (false)~~

~~$n \in 2, \ldots n \ldots \frac{2}{2}$~~

$0 \le 2n < cn^2$

or $0 \le 2 < cn$

Put $c=1$, $2 < 1.\left(\frac{2}{1}+\varepsilon\right)$, $\varepsilon > 0$ True.

$c=2$, $2 < 2\left(\frac{2}{2}+\varepsilon\right)$ $\varepsilon > 0$ (True)

$c=3$, $2 < 3\left(\frac{2}{3}+\varepsilon\right)$ True.

$\therefore$ $c > 0$ & $n > \frac{2}{c}$

(29) Is $2n^2 = o(n^2)$?

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

$\lim_{n\to\infty} \frac{2n^2}{n^2} = 2 \ne 0$ $\therefore$ $2n^2 \ne o(n^2)$.

Assuming the contradiction,

$0 \le 2n^2 < c.n^2$

True for $c > 3$ (but not for $c=1,2$.).

$\therefore$ $2n^2 \ne o(n^2)$

$2n^2 < cn^2$

$n=1$, $2 < c$

Take $c = 3$ (True)

But not for $c = 1, 2$.

$\therefore$ $2n^2 \ne o(n^2)$,

(30) $n! = o(n^n)$.

$$\lim_{n\to\infty} \frac{n!}{n^n} = 0$$

$f(n) = n!$

$f(n) \leq n^n \quad \forall\, n \geq 1$

ie $n! \leq n^n \quad \forall\, n \geq 1$

But o (small-oh) inequality is

$$0 \leq f(n) < c.g(n).$$

$\therefore f(n) = n!$

$f(n) < n^n . c \;\forall\; n > 1$ (True)

ie $n! < n^n . c \;\forall\; n > 1$ (True)

$\therefore$ $c > 0$ & $n > 1$

(31) $n! = \omega(2^n)$

$f(n) = n! \qquad g(n) = 2^n$

$0 \leq cg(n) < f(n)$

$0 \leq c2^n < n!$

$n = 1, \quad c.2 < 1$ (False)

$g(n) = 2^n$

Important Big-Oh Results

1) If $f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$, where $a_0, a_1, \ldots a_n$ are real numbers then $f(x) = O(x^n)$.

Proof. $f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$

$$= x^n \left(a_n + \frac{a_{n-1}}{x} + \ldots + \frac{a_1}{x^{n-1}} + \frac{a_0}{x^n} \right)$$

$$\leq |a_n| x^n + |a_{n-1}| x^n + \ldots + |a_1| x^n + |a_0| x^n$$

$$\leq x^n \left(|a_n| + |a_{n-1}| + \ldots + |a_1| + |a_0| \right)$$

$$\leq g(n) \cdot c. \quad \text{where } c = |a_n| + \ldots + |a_0| \text{ \& } n_0 = 1$$

2) $n! = O(n^n)$.

$$f(n) = n!$$
$$= 1 \cdot 2 \cdot 3 \cdot \ldots n$$
$$\leq n \cdot n \cdot n \cdot \ldots n$$
$$\leq n^n$$
$$f(n) \leq g(n) \cdot c. \qquad \boxed{c = 1, \; n_0 = 1}$$

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

$$f(n) = \log n!$$
$$\leq \log n^n$$
$$\leq n \log n$$
$$f(n) \leq g(n) \cdot c; \qquad \boxed{c = 4, \; n_0 = 1}$$

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

$n \leq c.2^n$

Put $n = 1$, $1 \leq c.2$, $1 \leq 2$ ∴ Take $c=1, n_0=1$

$n = O(2^n)$.

Also, $n \leq c.2^n$

Take log. both sides,

$\log_2 n \leq c.n \log_2 2$

$\log_2 n \leq c.n$

or

$\log_2 n \leq O(n)$

5) $\log_b n = O(n)$

$f(n) = \log_b n = \frac{\log_2 n}{\log_2 b} \leq c.n$

$= \frac{\log_2 n}{\log_2 b} \leq \left(\frac{1}{\log_2 b}\right).n$ $\left[\log_2 n \leq n\right]$.

$c = \frac{1}{\log_2 b}$, $n_0 = 1$

Growth of Combinations of functions

Th.1 If $f_1(x) = O(g_1(x))$ and $f_2(x) = O(g_2(x))$ then, $(f_1 + f_2)(x) = O(\max(g_1(x), g_2(x)))$.

Proof $f_1(x) = O(g_1(x))$ [given]. , Also $f_2(x) = O(g_2(x))$ (given)

$\therefore f_1(x) \leq c_1 \cdot g_1(x)$ $\quad\quad$ $\therefore f_2(x) \leq c_2 \cdot g_2(x)$.

L.H.S. $(f_1 + f_2)(x)$.

$\Rightarrow f_1(x) + f_2(x)$ [using f^n].

$\Rightarrow \therefore f_1(x) + f_2(x) \leq c_1 \cdot g_1(x) + c_2 g_2(x)$.

$\leq c_1 g(x) + c_2 g(x)$ [$\because g(x) = \max(g_1(x), g_2(x))$

$\leq (c_1 + c_2) g(x)$. $\quad$ $\because f(x) = n^2 + n + 1$

$\leq c \cdot g(x)$ $\quad$ $f(x) \leq n^2 + n^2 + n^2$]

[$\because c = c_1 + c_2$]

$\leq O(g(x))$

Corollary If $f_1(x) = O(g(x))$ & $f_2(x) = O(g(x))$.

Then $(f_1 + f_2)(x) = O(g(x))$.

Proof $f_1(x) \leq c \cdot g(x)$ & $f_2(x) \leq c\, g(x)$.

$(f_1 + f_2)(x) = f_1(x) + f_2(x)$

$\leq c_1 g(x) + c_2 g(x)$

$\leq (c_1 + c_2) g(x)$

$\leq c \cdot g(x)$ [$c = c_1 + c_2$]

$= O(g(x))$

Th 2 If $f_1(x) = O(g_1(x))$ & $f_2(x) = O(g_2(x))$.
Then $(f_1 f_2)(x) = O(g_1(x) \cdot g_2(x))$.

eg. Give Big-O estimate for
$f(n) = 3n(\log n!) + (n^2 + 3)\log_2 n$, where n is positive integer.

Ans. first estimate the product,

$3n(\log(n!))$

we know, $3n = O(n)$ &

$\log(n!) = O(n \log_2 n)$

$\therefore$ Acc. to Th. 2.

$$(f_1 f_2)(x) = O(n \cdot n \log_2 n)$$
$$= O(n^2 \log_2 n).$$

Next, estimate the product $\rightarrow (n^2+3)\log_2 n$.

we know, $n^2 + 3 = O(n^2)$ &

$\log_2 n = O(n)$

$$\therefore (f_1 f_2)(n) = O(n^2 \cdot n)$$
$$= O(n^3).$$

$\therefore f_1(n) = O(g_1(n))$ & $f_2(n) = O(g_2(n))$

using Th 1,

$$(f_1 + f_2)(n) = O(\max(g_1(n), g_2(n)))$$
$$= O(\max(n^2 \log_2 n, n^3))$$
$$= O(n^3)$$ Ans

Q2. Give a Big-O estimate for

$$f(x) = (x+1)\log(x^2+1) + 3x^2$$

Ans. first estimate the product,

$$(x+1)(\log(x^2+1))$$

we know, $\boxed{x+1 = O(x)}$

Also, ~~$x+1 = O(x)$~~ $\log(x^2+1) = O(\log_2 x)$.

Because $x^2+1 = O(x^2)$.

if $x^2+1 \leq C x^2$

$\therefore C = 2, x_0 = 1$

Taking log both sides.

$$\log(x^2+1) \leq \log(2x^2)$$

$$\leq \log 2 + \log_2 x^2 \quad (\log_c(ab) = \log_c a + \log_c b)$$

$$\leq 1 + 2\log_2 x$$

$$\boxed{\log(x^2+1) = O(\log_2 x)}$$

Now, $(f_1 \cdot f_2)(x) = O(g_1(x)\, g_2(x))$

$$= O(x \cdot \log_2 x)$$

Now $3x^2 = O(x^2)$.

$\therefore (f_1 + f_2)(x) = O(\max(g_1(x), g_2(x)))$

$$= O(\max(x \log_2 x, x^2)) = O(x^2)$$

$$\boxed{\therefore f(x) = O(x^2)}$$