DAA

Unit - 1

Algorithm :- An algorithm is a step by step method to solve a problem and is independent of hardware and software.

Characteristics of Algorithm :-

1) Uniqueness :- Each step of an algorithm should be unique and self-explanatory.
2) Input :- Algorithm must receive atleast one input.
3) Output :- Algorithm must ~~stop~~ obtain atleast one output
4) finiteness :- Algorithm should stop after finite number of steps.
5) ~~Uma~~ Unambigous :- It should have single and clear interpretion.

Algorithm Analysis :- Analysis of an algorithm is required to detect the correctness & used to measure the quantitative ~~analysis~~ of efficiency of an algorithm.

Complexity :- Complexity of an algorithm is a function describing the efficiency of the algorithm in terms of amount of data or time it requires.

The performance of algorithm is measured on basis of following properties :-

1) Time Complexity
2) Space Complexity.

1) Space Complexity :- It represents the amount of memory space required by an algorithm in its life cycle.

2) Time Complexity :- The time complexity is a function which gives the amount of time required by an algorithm to run to completion.

3 Cases :-

1) Best Case :- It is defined as ~~max~~ minimum number of steps and minimum amount of time required to complete the execution of an algorithm.

2) Worst Case :- It is defined as maximum number of steps and maximum amount of time required to complete the execution of an algorithm.

3) Average Case :- It is defined as average number of steps and average time required to complete the execution of an algorithm.

Asymptotic Notation :-

It is basically a mathematical tool to represent the complexity of an algorithm.

Following are the asymptotic Notation :-

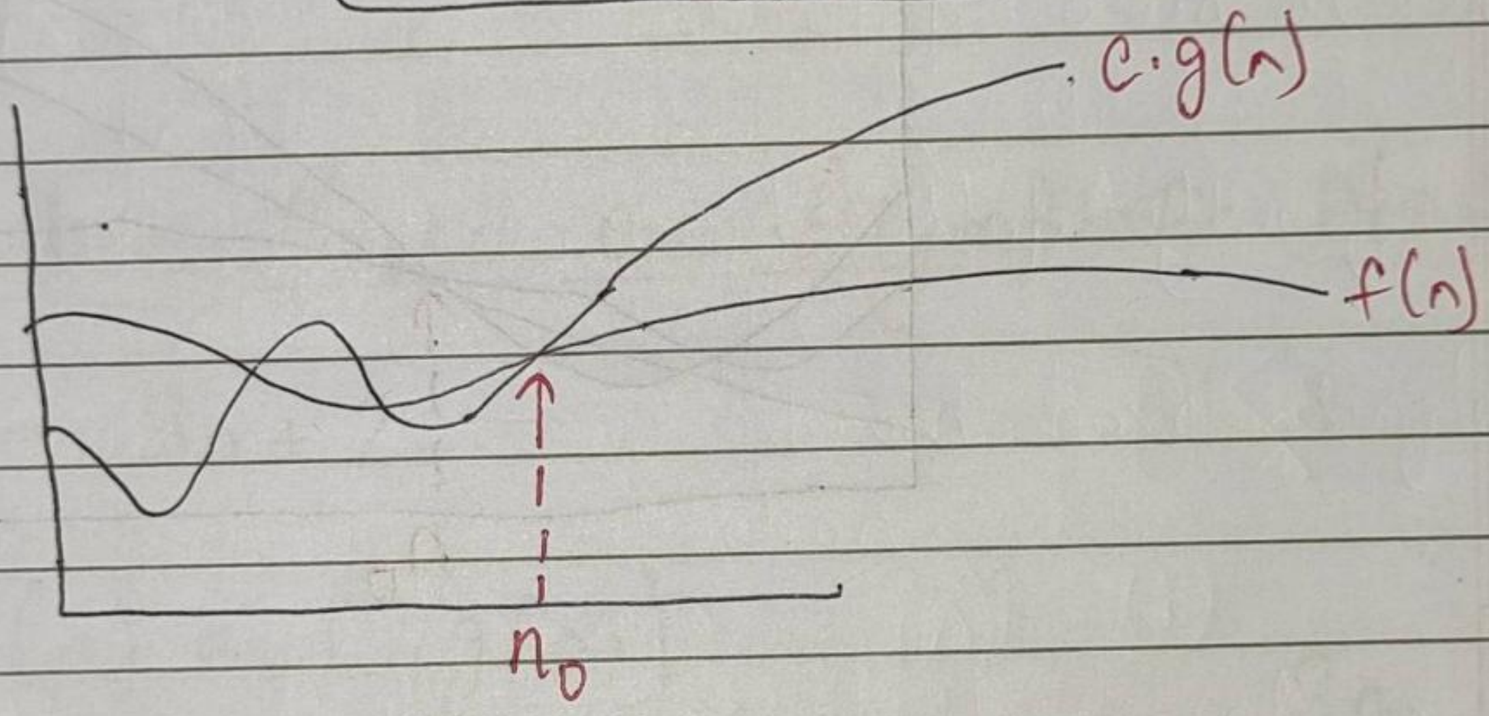
(i) Big O Notation :-

The Big O notation defines the upper bound of an algorithm. It also defines worst case of the algorithm.

Condition :-

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

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

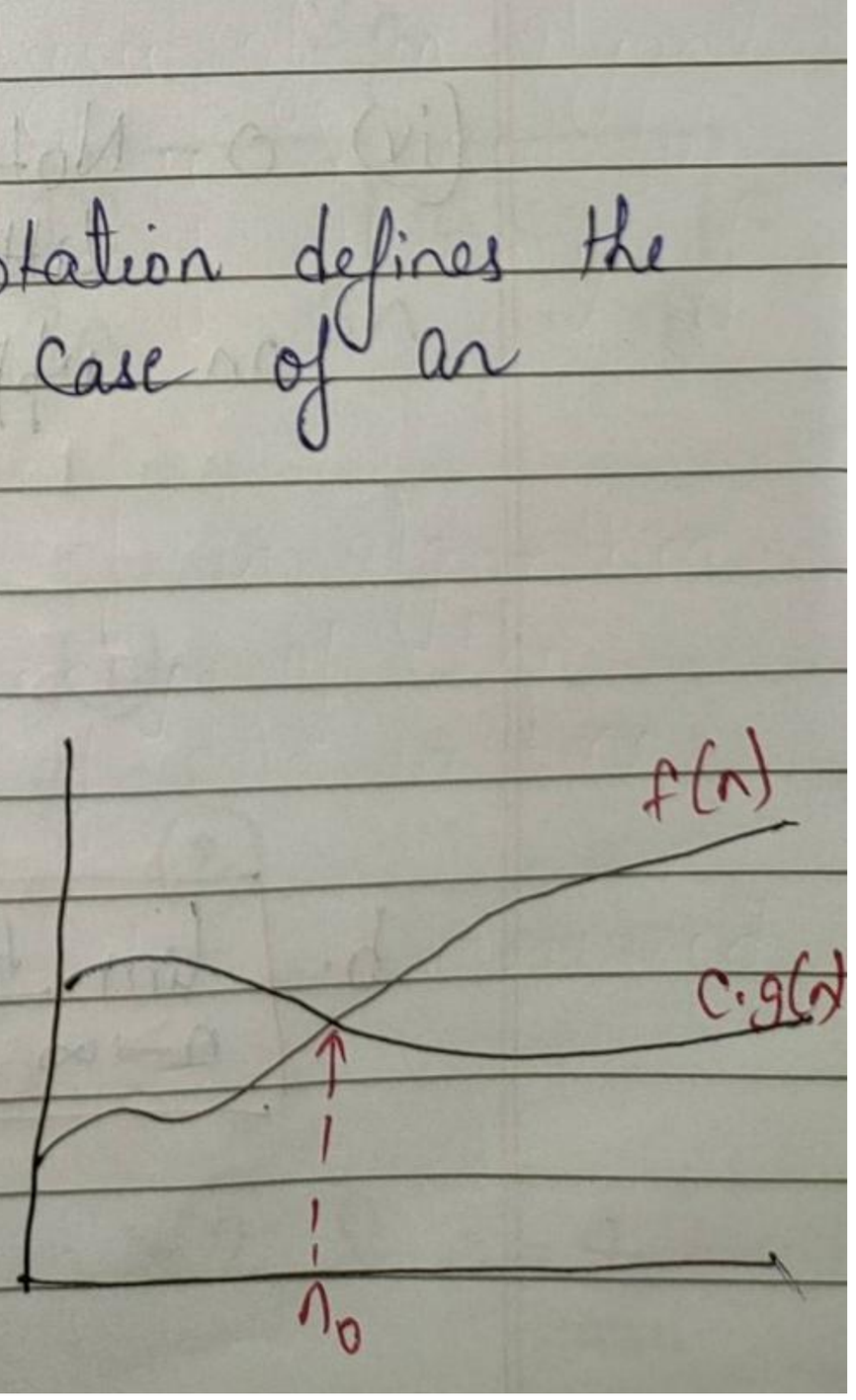


(ii) Ω Notation :-

The Big Omega (Ω) notation defines the lower bound and the best case of an algorithm.

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

$$f(n) \geq c \cdot g(n)$$

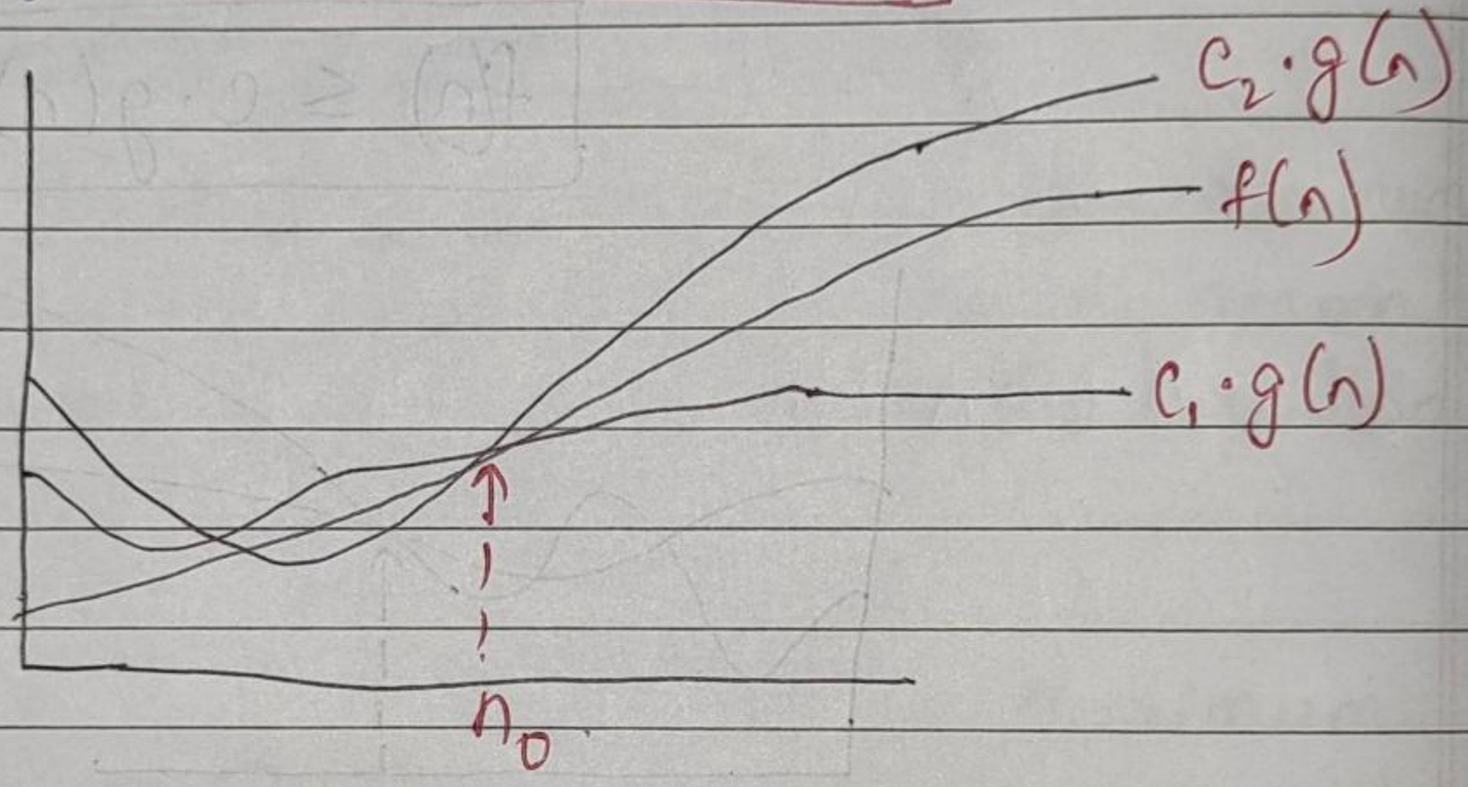


(iii) Θ Notation :-

The theeta Notation (Θ) defines the exact asymptotic behaviour of the algorithm. It is also used to define average case & average bound of algorithm.

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

$$\boxed{c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)}$$



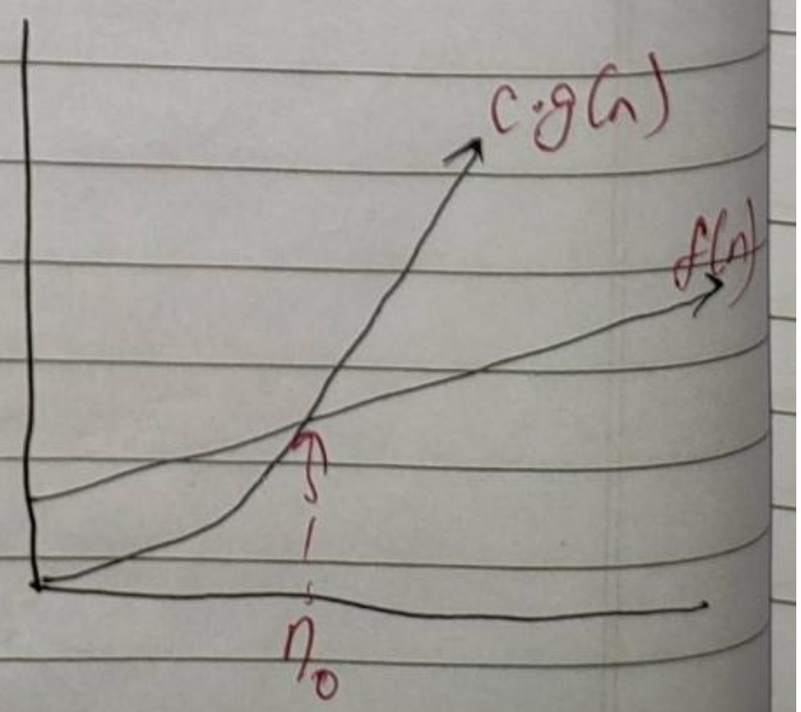
(iv) o - Notation :-

Little o Notation is used to describe an upper-bound that cannot be tight.

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

a. $\boxed{f(n) < c \cdot g(n)}$

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



(v) Small - ω Notation :-

Little - ω (also known as small ω) is used to denote a lower bound that is not a asymptotic tight.

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

a. $\boxed{f(n) > c \cdot g(n).}$

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

Question Based on Asymptotic Notation :-

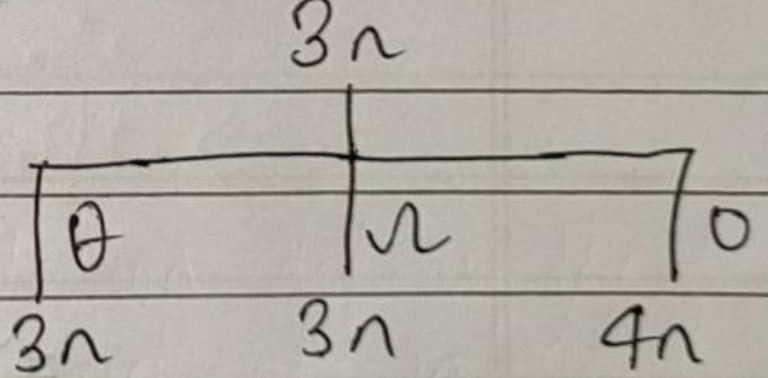
1) $f(n) = 3n + 2.$

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

$f(n) \leq c \cdot g(n)$ ---(i)

$g(n) = n$ [highest power of n]

$c = 4.$



Putting these values in eq. (i)

$3n + 2 \leq 4n$ ---(2)

for $n = 1$ $\quad 5 \leq 4$ (false)

$n = 2$ $\quad 8 \leq 8$ (True).

$\therefore f(n) = O(n).$

$\Rightarrow 3n+2 = O(n)$ for all $n \geq 2$ & $c = 4$

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

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

$f(n) \geq c \cdot g(n)$ ——— (i)

$g(n) = n$ (highest power of n)

$c = 3$.

$\Rightarrow 3n + 2 \geq 3n$

$n = 1$ $\quad 5 \geq 3$ (True).

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

$\Rightarrow 3n + 2 = \Omega(n)$ for all $n \geq 1$ & $c = 3$.

3) $f(n) = 2n + 5$

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

$f(n) \geq c \cdot g(n)$ ——— (i)

$g(n) = n$

$c = 3$

$\Rightarrow 2n + 5 \geq 3n$

$n = 1$	$7 \leq 3$	false
$n = 2$	$9 \leq 6$	false
$n = 3$	$11 \leq 9$	false
$n = 4$	$13 \leq 12$	false
$n = 5$	$15 \leq 15$	True

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

$\Rightarrow 2n + 5 = \Omega(n)$ for $n \geq 5$ & $c = 3$

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

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

$g(n) = n$

$c = 2$

$\Rightarrow 2n+5 \geq 2n$

$n=1 \quad 7 \geq 2$. True

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

$\Rightarrow 2n+5 = \Omega(n)$ for $n \geq 1$ & $c = 2$.

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

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

$g(n) = n$

$c_1 = 2, \quad c_2 = 3$.

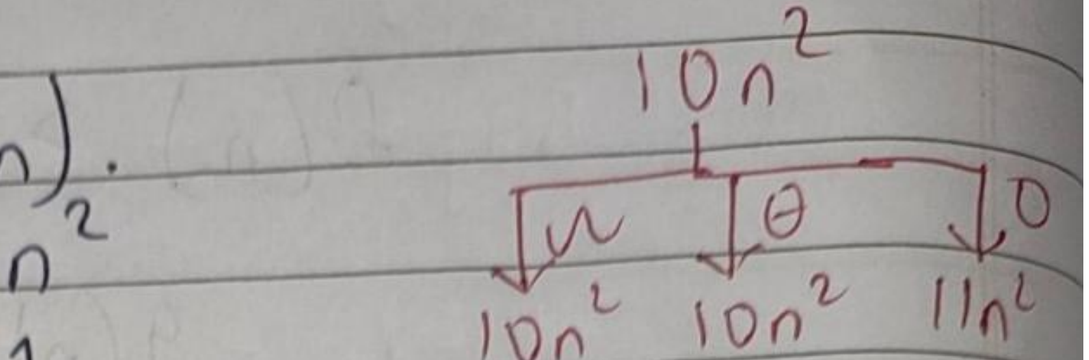
$\Rightarrow 2n \leq 2n+5 \leq 3n$

$n=1$	$2 \leq 7 \leq 3$
$n=2$	$4 \leq 9 \leq 6$
$n=3$	$6 \leq 11 \leq 9$
$n=4$	$8 \leq 13 \leq 12$
$n=5$	$10 \leq 15 \leq 15$

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

$\Rightarrow 2n+5 = \Theta(n)$ for $n \geq 5$ & $c_1 = 2, c_2 = 3$

4) $f(n) = 10n^2 + 4n^2 + 2$



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

$g(n) = n^2$

$c = 11$

$10n^2 + 4n^2 + 2 \leq 11n^2$

$n=1$	$10\times(1)^2 + 4\times1 + 2 \leq 11\times(1)^2$
	$\Rightarrow 16 \leq 11$ false
$n=2$	$10\times(2)^2 + 4\times2 + 2 \leq 11\times(2)^2$
	$50 \leq 44$ false
$n=3$	$10\times(3)^2 + 4\times3 + 2 \leq 11\times(3)^2$
	$104 \leq 99$ false
$n=4$	$10\times(4)^2 + 4\times4 + 2 \leq 11\times(4)^2$
	$178 \leq 176$ false
$n=5$	$10\times(5)^2 + 4\times5 + 2 \leq 11\times(5)^2$
	$272 \leq 275$ True.

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

$\Rightarrow 10n^2 + 4n + 2 = O(n^2)$ for $n \geq 5$ & $c = 11$.

(ii) $f(n) \geq c \cdot g(n)$.

$g(n) = n^2$

$c = 10$

$\Rightarrow 10n^2 + 4n + 2 \geq 10n^2$

$n=1$ | $10\times(1)^2 + 4\times1 + 2 \geq 10\times(1)^2$

$16 \geq 10$

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

$\Rightarrow 10n^2 + 4n + 2 = \Omega(n)$ for $n \geq 1$ & $c = 10$.

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

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

$g(n) = n^2$ $c_1 = 10$, $c_2 = 11$

$\Rightarrow 10n^2 \leq 10n^2 + 4n + 2 \leq 11n^2$

$n=1$	$10(1)^2 \leq 10(1)^2 + 4\times1 + 2 \leq 11\times(1)^2$
	$10 \leq 16 \leq 11 \rightarrow$ ~~Tove~~ false
$n=2$	$10(2)^2 \leq 10(2)^2 + 4\times2 + 2 \leq 11\times(2)^2$
	$40 \leq 50 \leq 44 \rightarrow$ false
	$10(3)^2 \leq 10(3)^2 + 4\times3 + 2 \leq 11\times(3)^2$
	$90 \leq 104 \leq 99 \rightarrow$ false

$10(4)^2 \leq 10(4)^2 + 4\times4 + 2 \leq 11\times(4)^2$

$160 \leq 178 \leq 176 \rightarrow$ false

$10(5)^2 \leq 10(5)^2 + 4\times5 + 2 \leq 11\times(5)^2$

$250 \leq 272 \leq 275 \rightarrow$ Tove.

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

$\Rightarrow 10n^2 + 4n + 2 = \Theta(n^2)$ for $n \geq 5$ & $c_1 = 10$, $c_2 = 11$

5) $3n + 2 = o(n^2)$ is true or false?

for $o(g(n))$:-

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

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

$$\lim_{n\to\infty} \frac{3n+2}{n^2} = \lim_{n\to\infty} \frac{3}{n} + \frac{2}{n^2} = 0 + 0$$

$$= 0$$

So, $3n + 2 = o(n^2)$ is true

6) $3n+2 = O(n)$ is true or false?

$\lim_{n\to\infty} \frac{f(n)}{g(n)}$ should be equal to 0.

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

$$\lim_{n\to\infty} \frac{3n+2}{n} = \lim_{n\to\infty} 3 + \frac{2}{n} = 3+0 = 3$$

So, since $\lim_{n\to\infty} \frac{f(n)}{g(n)} \neq 0$.

So, $3n+2 = O(n)$ is false.

7) $\frac{n^3}{1000} = O(n^3)$.

Here $f(n) = \frac{n^3}{1000}$. $g(n) = n^3$.

$$\lim_{n\to\infty} \frac{\frac{n^3}{1000}}{n^3} = \frac{1}{1000} \neq 0$$

So, $\frac{n^3}{1000} \neq O(n^3)$

8) $2n+3 = \omega(n)$

for $\omega(n)$, $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$.

Here, $f(n) = 2n+3$. $g(n) = n$

$$\lim_{n\to\infty} \frac{2n+3}{n} = 2 + \frac{3}{\infty} = 2 \neq \infty$$

So, $2n+3 \neq \omega(n)$.

9) $2n+5 = \omega(n^2)$.

Here, $f(n) = 2n+5$, $g(n) = n^2$

$$\lim_{n\to\infty} \frac{2n+5}{n^2} = \lim_{n\to\infty} \frac{2}{n} + \frac{5}{n^2} = 0.$$

$\therefore 2n+5 \neq \omega(n^2)$

10) $10n^2+7 = \omega(n)$.

Here $f(n) = 10n^2+7$ $g(n) = n$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{10n^2+7}{n} = \lim_{n\to\infty} 10n + \frac{7}{n}$$

$= \infty$

$\therefore 10n^2+7 = \omega(n)$

11) $2n+5 = o(n^2)$

Here, $f(n) = 2n+5$

$g(n) = n^2$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{2n+5}{n^2}$$

$$= \lim_{n\to\infty} \frac{2n}{n^2} + \frac{5}{n^2}.$$

$$= \lim_{n\to\infty} \frac{2}{n} + \frac{5}{n^2} = 0$$

$\therefore 2n+5 = o(n^2)$

Algorithm Design Paradigms :-

There are various methods to design an algorithm. An algorithmic paradigms or algorithm design paradigm is a generic model or framework which underline the design of algorithm.

Following are some common algorithm design paradigm which is frequently used :-

1) Brute force :- It is the easiest approach to apply which is usually useful for ~~each~~ solving small size instances of problem.

It involves solving a problem based on the problem's statements and definitions of the concept involved.

Examples :-

1) Computing n!
2) Selection Sort, Bubble Sort.
3) Sequential Search

2) Divide & Conquer :-

It basically involves 3 steps which are as follows :-

1. Divide :- It means splitting the given problem into sub-problem.
2. Conquer :- It means solving subproblem independently.
3. Combine :- Combine sub-problem solution.

Examples :-
1. Binary search
2. Merge Sort
3. Quick Sort, etc

3) Greedy Algorithm :- It is basically based on "take whatever you can get now" strategy. In this at each step the choice must be locally optimal. This algorithm works better on optimization problem.

Examples :-
1) Minimal Spanning Tree
2) Shortest distance in graph
3) Coin exchange problem

4) Dynamic Programming :-
It is almost similar to Divide and conquer technique.
Difference between the two is that in dynamic ~~prob~~ programming solution of subproblem is stored and can be ~~plced~~ used later.

Examples :-
1) Fibonacci numbers using recursion
2) Warshall's ~~alg~~ algorithm.

5) Backtracking :- This method is used for state-space search problem. Now, you can ask what exactly is state-space problem? It is basically a problem where the representation consist of an initial state, goal state, a set of intermediate state, a set of operations that transform one state into another a cost function and a utility function.

Example :-
1) DFS problem
2) Maze problem

6) Branch-and-Bound :-
It is used when we can evaluate each node using the cost and utility function.
At each step, we choose the best node to proceed further. Branch and bound algorithm are implemented using priority queue.

Example :-
1) 8-puzzle problem
2) N queens problem.

Mathematical Induction

It involves two steps :-
1) Base step
2) Inductive step.

Q1. Prove $1+2+3+4 \dots +n = \frac{n(n+1)}{2}$ using Mathematical Induction.

Sol → $S_n = 1+2+3+4+ \dots +n = \frac{n(n+1)}{2}$

first, Show that S_n is true for $n=1$.

$$1 = \frac{1\cdot(1+1)}{2} \Rightarrow 1 = 1$$

$\Rightarrow$ LHS = RHS

$\therefore S_n$ is true for $n=1$.

Now, Let S_n be true for $n=k$.

$$S_k = 1+2+3+4+ - - +k = \frac{k(k+1)}{2} \quad --(1)$$

Now, we show that S_n is true for $n=k+1$

$$S_{k+1} = 1+2+3+4+ - - +k+k+1 = \frac{(k+1)(k+2)}{2} \quad --(2)$$

LHS of (2)

$$1+2+3+4+ - - +k+(k+1)$$

$$\Rightarrow \frac{k(k+1)}{2} + (k+1) \quad [\text{using (1) in this}]$$

$$\Rightarrow (k+1)\left[\frac{k}{2}+1\right]$$

$$\Rightarrow \frac{(k+1)(k+2)}{2} = RHS.$$

So, LHS = RHS

$\therefore$ S_n is true for $n=(k+1)$

$\therefore$ By using Mathematical Induction, we have proved S_n is true for all $n \in N$.

Q2. Proove $1^2+3+3^2+ - - - +3^{n-1} = \frac{3^n-1}{2}$ using Mathematical Induction.

Sol → $S_n = 1+3+3^2+ - - - +3^{n-1} = \frac{3^n-1}{2}$.

Show that S_n is true for $n=1$

$$\Rightarrow 1 = \frac{3^1-1}{2} = 1 = RHS$$

$\therefore$ S_n is true for $n=1$.

Let S_n be true for $n=k$.

$$\Rightarrow S_k = 1+3+3^2+\dots+3^{k-1} = \frac{3^k-1}{2}$$

Now, we show that S_n is true for $n=(k+1)$

$$\Rightarrow S_{k+1} = 1+3+3^2+\dots+3^{k-1}+3^k = \frac{3^{k+1}-1}{2}$$

$$= \frac{3^k-1}{2} + 3^k$$

$$= \frac{3^k-1+2\times 3^k}{2}$$

$$= \frac{3\cdot 3^k-1}{2}$$

$$= \frac{3^{k+1}-1}{2} = \text{RHS.}$$

$\therefore$ By Mathematical Induction, S_n is true for all $n \in N$.

Q3. $1+3+5+7+\dots+(2n-1) = n^2$.

Sol → $S_n = 1+3+5+7+\dots+(2n-1) = n^2$

Show that S_n is true for $n=1$

$$\Rightarrow 1 = (1)^2 = 1 = \text{RHS}$$

$\therefore$ S_n is true for $n=1$.

Let S_n be true for $n=k$.

$$S_k = 1+3+5+7+\dots+(2k-1) = k^2$$

Now, Show that S_n is true for $n=(k+1)$.

$$\Rightarrow S_{k+1} = 1+3+5+7+---+(2k-1)+(2k+1)=(k+1)^2$$

$$= k^2+(2k+1)$$

$$= (k^2+2k+1)$$

$$= (k+1)^2 = RHS$$

$\therefore$ By Mathematical Induction, S_n is true for all $n \in N$.

Q4. $1^3+2^3+3^3+---+n^3 = \left(\frac{n(n+1)}{2}\right)^2$

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

Show that S_n is true for $n=1$.

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

$$\Rightarrow 1^3 = (1)^2$$

$$\Rightarrow 1 = 1$$

$$\Rightarrow LHS = RHS$$

$\therefore$ S_n is true for $n=1$.

Let S_n be true for $n=k$.

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

Show that S_n is true for $n=(k+1)$

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

$$\left(\frac{k(k+1)}{2}\right)^2 + (k+1)^3$$

$$= (k+1)^2\left[\frac{k^2}{4} + (k+1)\right]$$

$$= (k+1)^2\left[\frac{k^2+4k+4}{4}\right]$$

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

$$= \left[\frac{(k+1)(k+2)}{2}\right]^2 = RHS$$

$\therefore$ S_n is true for $n = (k+1)$

$\therefore$ By Mathematical Induction, S_n is true for all $n \in N$.

Q5. $1^2 + 2^2 + 3^2 + -- + n^2 = \frac{n(n+1)(2n+1)}{6}$

Sol → $S_n = 1^2 + 2^2 + 3^2 + -- + n^2 = \frac{n(n+1)(2n+1)}{6}$

Show that S_n is true for $n = 1$

$\Rightarrow 1^2 = \frac{1(1+2)(2+1)}{6} = 1 = RHS$

$\therefore$ LHS = RHS

$\therefore$ S_n is true for $n = 1$.

Let S_n be true for $n = k$.

$S_k = 1^2 + 2^2 + 3^2 + -- + k^2 = \frac{k(k+1)(2k+1)}{6}$

Show that S_n be true for $n=(k+1)$

$\Rightarrow 1^2+2^2+3^2+ \cdots k^2+(k+1)^2 = \frac{(k+1)(2k+3)(k+2)}{6}$

$\Rightarrow \frac{k(k+1)(2k+1)}{6} + (k+1)^2$

$\Rightarrow (k+1)\left[\frac{k(2k+1)}{6} + (k+1)\right]$

$\Rightarrow (k+1)\left[\frac{2k^2+k+6k+6}{6}\right]$

$\Rightarrow \frac{(k+1)(2k^2+7k+6)}{6}$

$\Rightarrow \frac{(k+1)\left[2k^2+4k+3k+6\right]}{6}$

$\Rightarrow \frac{(k+1)\left[2k(k+2)+3(k+2)\right]}{6}$

$\Rightarrow \frac{(k+1)(2k+3)(k+2)}{6} = RHS$

Hence, S_n is true for $n=(k+1)$.

So, S_n is true for all $n \in N$.

Recurrence Relation

It is an equation which recursively defines a sequence where next function is a function of its previous term.

$$a_1 = a_0 + 2$$
$$a_2 = a_1 + 2$$
$$a_3 = a_2 + 2$$
$$\vdots$$
$$a_n = a_{n-1} + 2$$

Example :- fibonacci Series :-

0 1 1 2 3 5 8 13 - - .

```
if (n == 0) return 0;
if (n == 1) return 1;
else return (n-1) + (n-2)
```

Here for n it depend on (n-1) & (n-2).
This is an example of Recurrence Relation.

Methods to Solve :-

1) Substitution Method :-
2) Iteration Method
3) Recursion Tree Method
4) Master's Method.

2) Iteration Method :-

Q. If $T(n) = 1$ if $n = 1$ & $T(n) = T(n-1) + n$. Then find complexity.

Sol → $T(n) = T(n-1) + n$ ---. (1)

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

Putting this in eq. (1).

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

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

Putting this in eq. (2)

$T(n) = T(n-3) + (n-2) + (n-1) + n$ - (3)

$\vdots$

$= T(n-k) + (n-(k-1)) + (n-(k-2)) + \dots + n.$

We have to do this until we find $T(1)$. So for $n-k = 1$

$T(n) = T(1) + (n-k+1) + (n-k+2) + \dots$ -- n.

$= 1 + (1+1) + (1+2) + \dots + n$

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

So, $T(n) = O(n^2)$.

Q. If $n=1$, $T(1)=1$, else $T(n)=T(n-1)*n$. Then find complexity.

Sol → $T(n) = T(n-1) * n$ ---①

$T(n-1) = T(n-2)*(n-1)$.

Substituting it in eq. ①, we get

$T(n) = T(n-2)*(n-1)*n$ ---②

$T(n-2) = T(n-3)*(n-2)$.

Substituting it in eq. ② we get

$T(n) = T(n-3)*(n-2)*(n-1)*n$ ---③

$\vdots$

$T(n) = T(n-k)*(n-(k-1))*(n-(k-2))*--*n$.

$= T(1)*(n-k+1)*(n-k+2)*--*n$

$= 1*2*3*4*--*n$

$= n!$

So, $T(n) = O(n!)$

Q. find complexity for $T(n) = T(\frac{n}{2}) + 1$ if $n>1$.

Sol → $T(n) = T(\frac{n}{2}) + 1$ ---①

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

Substituting it in eq. ①

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

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

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

Substituting it in eq. (2), we get

$$T(n) = T\left(\frac{n}{2^3}\right) + 3 \quad --- (3)$$

$$\vdots$$

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

We have to make problem size = 1.

$$\Rightarrow \frac{n}{2^k} = 1$$

$$\Rightarrow n = 2^k \Rightarrow \log n = k \log 2$$

$$\Rightarrow k = \log n$$

$$T(n) = T(1) + k$$

$$= 1 + \log n$$

$$\Rightarrow T(n) = O(\log n)$$

Q. $T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + n & \text{if } n > 1 \\ 1 & \text{if } n = 1. \end{cases}$

Sol → $T(n) = 2T\left(\frac{n}{2}\right) + n \quad --- (1)$

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

Substituting it in eq. ①,

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

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

$$T(n) = 2^2 T\left(\frac{n}{2^2}\right) + 2n \quad \text{---} \quad ②$$

$$T\left(\frac{n}{2^2}\right) = 2T\left[\frac{n}{2^3}\right] + \frac{n}{4}$$

Substituting it in eq. ②.

$$T(n) = 2^2\left[2T\left(\frac{n}{2^3}\right) + \frac{n}{4}\right] + 2n$$

$$= 2^3 T\left(\frac{n}{2^3}\right) + n + 2n$$

$$T(n) = 2^3 T\left(\frac{n}{2^3}\right) + 3n$$

⋮

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kn$$

$$\frac{n}{2^k} = 1 \Rightarrow n = 2^k$$

$$\Rightarrow k = \log n$$

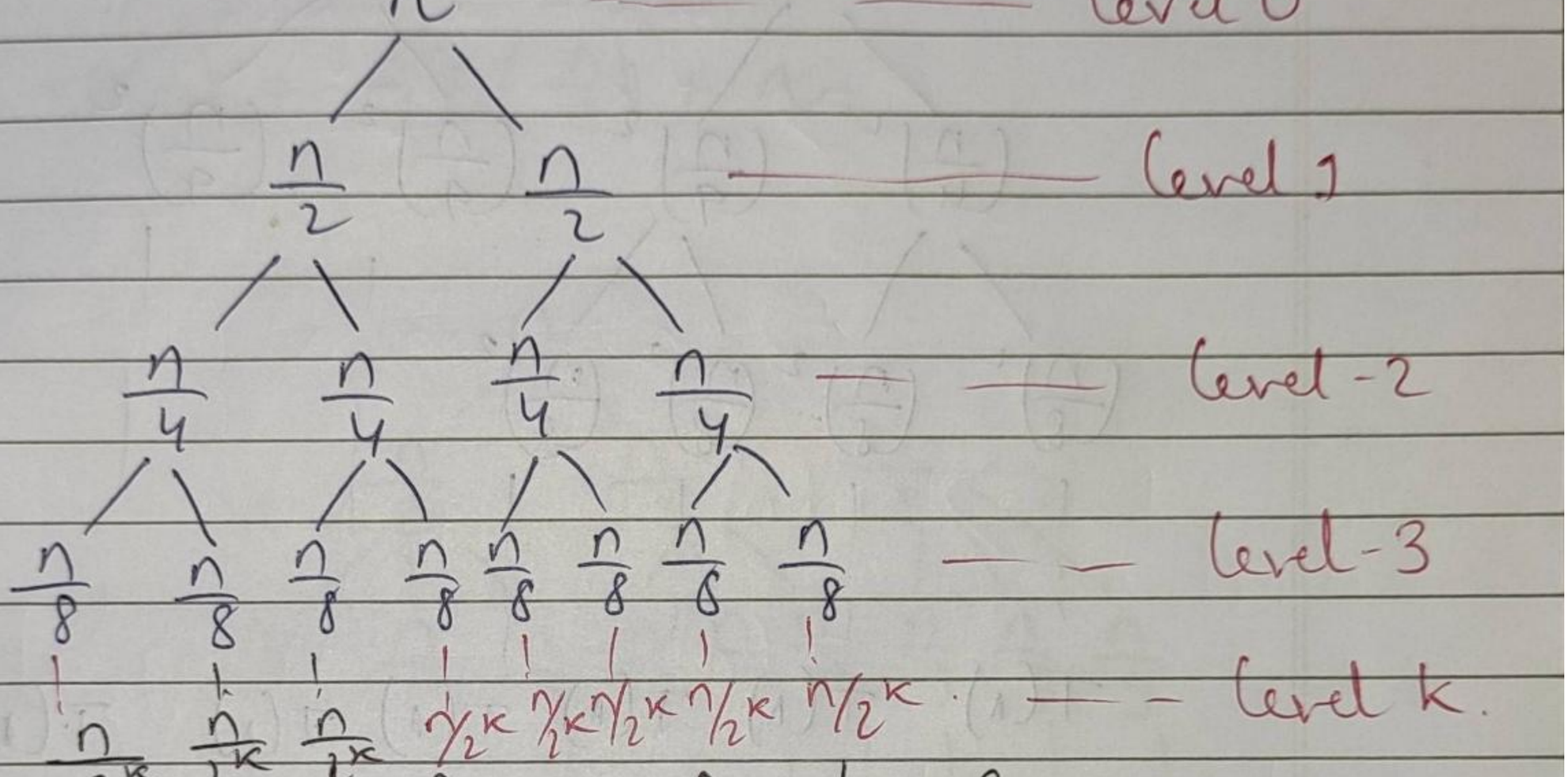
$$\Rightarrow T(n) = n \cdot T(1) + \log n \cdot n$$

$$T(n) = O(n \log n).$$

3) Recursion Tree Method :-

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

Sol →



Level	No. of nodes	Problem size	Cost
0	$1 = 2^0$	n	n
1	$2 = 2^1$	$n/2$	n
2	$4 = 2^2$	$n/2^2$	n
3	$8 = 2^3$	$n/2^3$	n
!	!	!	n
!	!	!	!
k	2^k	$n/2^k$	n

Total cost $= n + n + n + \dots + n$

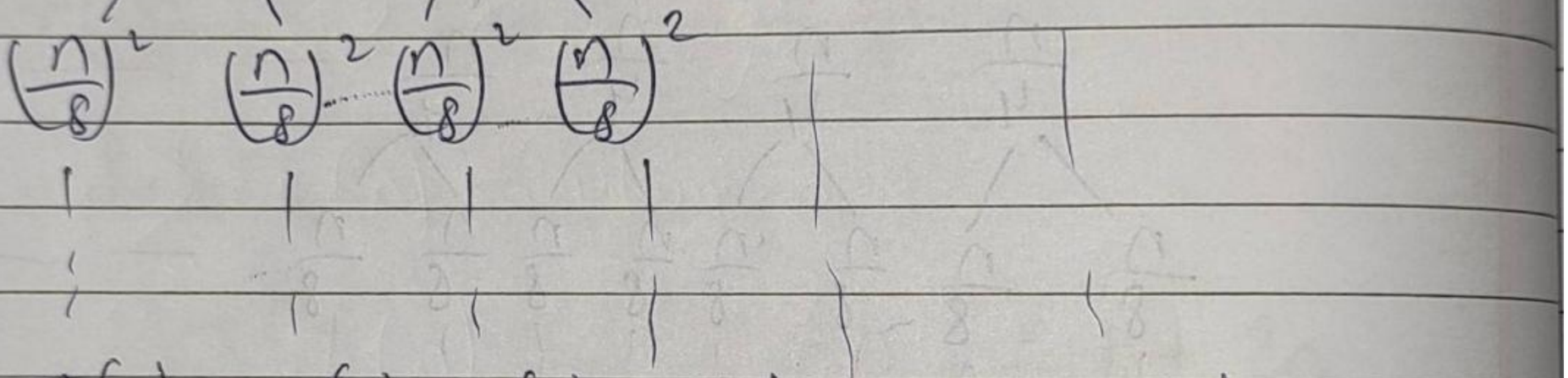
$= k \cdot n$

$\frac{n}{2^k} = 1 \Rightarrow k = \log n$

$\Rightarrow T(n) = O(n \log n)$ Ans

Q. $T(n) = 2T(\frac{n}{2}) + n^2$

Sol →



Level	No. of node	Problem size	Cost
0	$1 = 2^0$	n^2	n^2
1	$2 = 2^1$	$n^2/4$	$n^2/2$
2	$4 = 2^2$	$n^2/16$	$n^2/4$
3	$8 = 2^3$	$n^2/64$	$n^2/8$
!	!	!	!
i	2^{*i}	$n^2/4i$	$n^2/2i$

Total cost = $n^2 + \frac{n^2}{2} + \frac{n^2}{4} + \cdots \frac{n^2}{2^i}$

$= n^2\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots \frac{1}{2^i}\right)$.

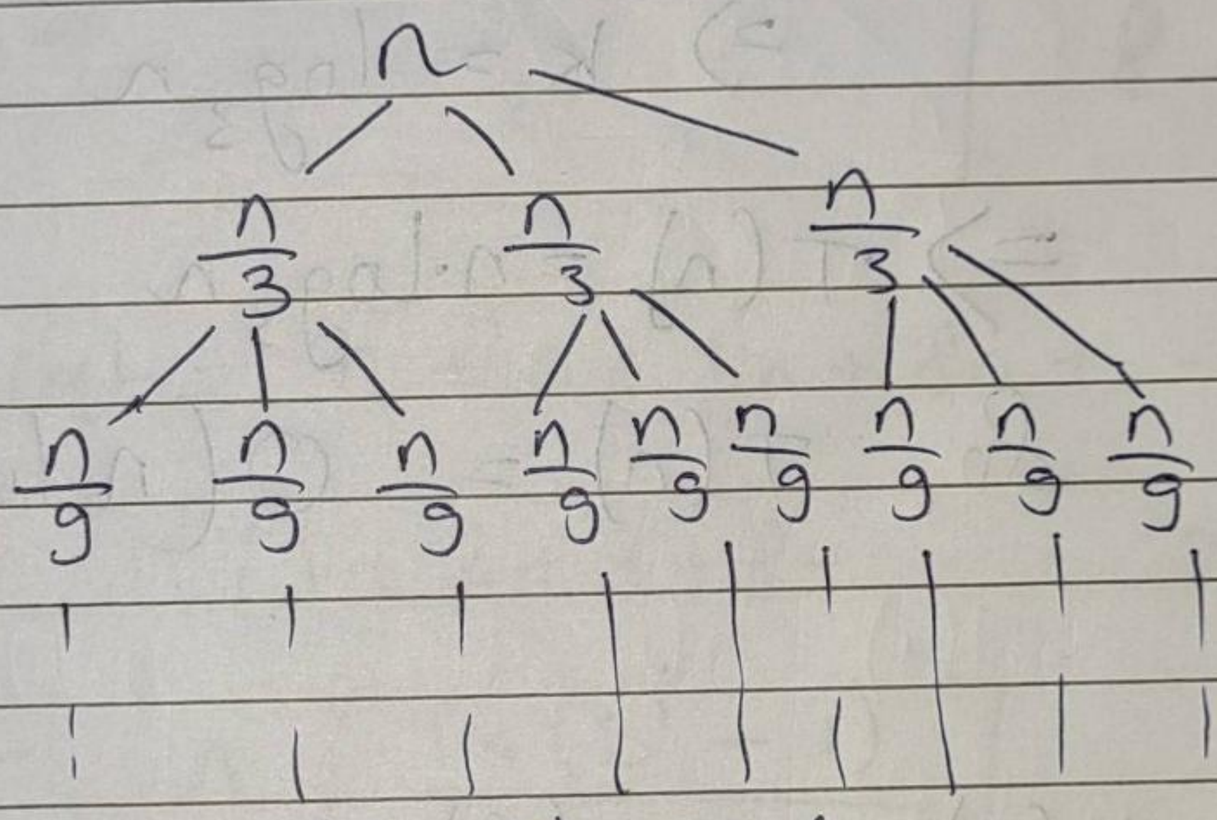
$$= n^2\left[\frac{1}{1-\frac{1}{2}}\right]$$

$$= 2n^2$$

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

Q3. $T(n) = 3T\left(\frac{n}{3}\right) + n$

Sol →



Level	No. of Node	Problem size	Cost
0	$1 = 3^0$	n	n
1	$3 = 3^1$	$\frac{n}{3}$	n
2	$9 = 3^2$	$\frac{n}{9}$	n
3	$27 = 3^3$	$\frac{n}{27}$	n
!	!	!	!
i	3^i	$\frac{n}{3^i}$	n

$\therefore$ Total cost = $n \times i$

$\propto$ ~~$n \times$~~

Also, $\frac{n}{3^k} = 1$

$\Rightarrow n = 3^k$

$\log n = k \log 3$

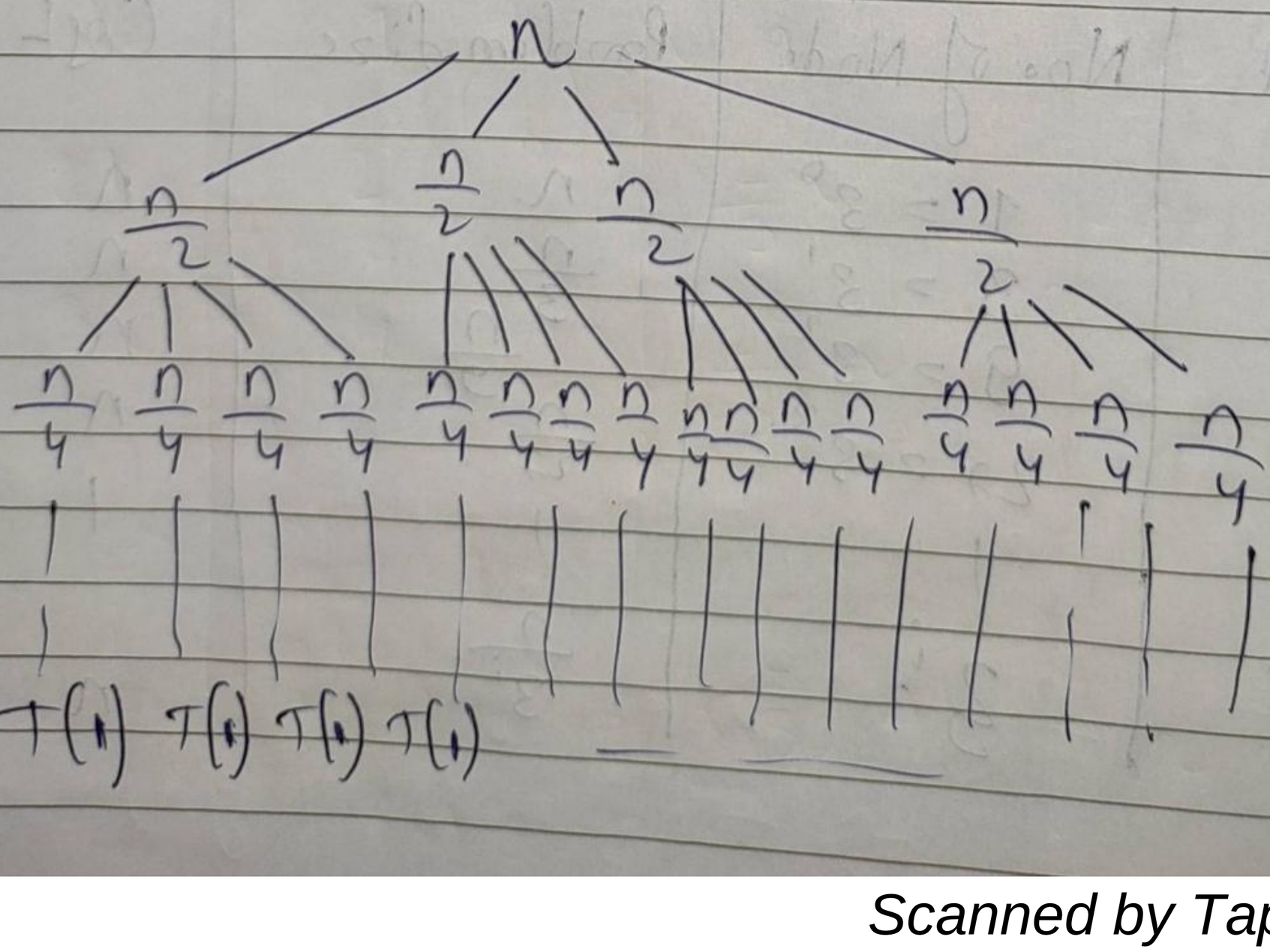
$\Rightarrow k = \frac{\log n}{\log 3}$

$\Rightarrow k = \log_3 n$

$\Rightarrow T(n) = n \cdot \log_3 n$

So, $T(n) = O(n \log_3 n)$

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



Level	No. of Node	Problem Size	Cost
0	$1 = 2^0$	n	n
1	$4 = 2^2$	$\frac{n}{2}$	$2n$
2	$16 = 2^4$	$\frac{n}{4}$	$4n$
⋮	⋮	⋮	⋮
i	2^{2i}	$\frac{n}{2^i}$	$2^i n$

$$\text{Total cost} = n + 2n + 4n + 8n + \cdots 2^i n$$

$$= n\left(1 + 2 + 4 + 8 + \cdots 2^i\right)$$

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

$$= n \times [2^i - 1]$$

$$= n * (2^{\log n} - 1)$$

$$= n * (n^{\log_2 2} - 1)$$

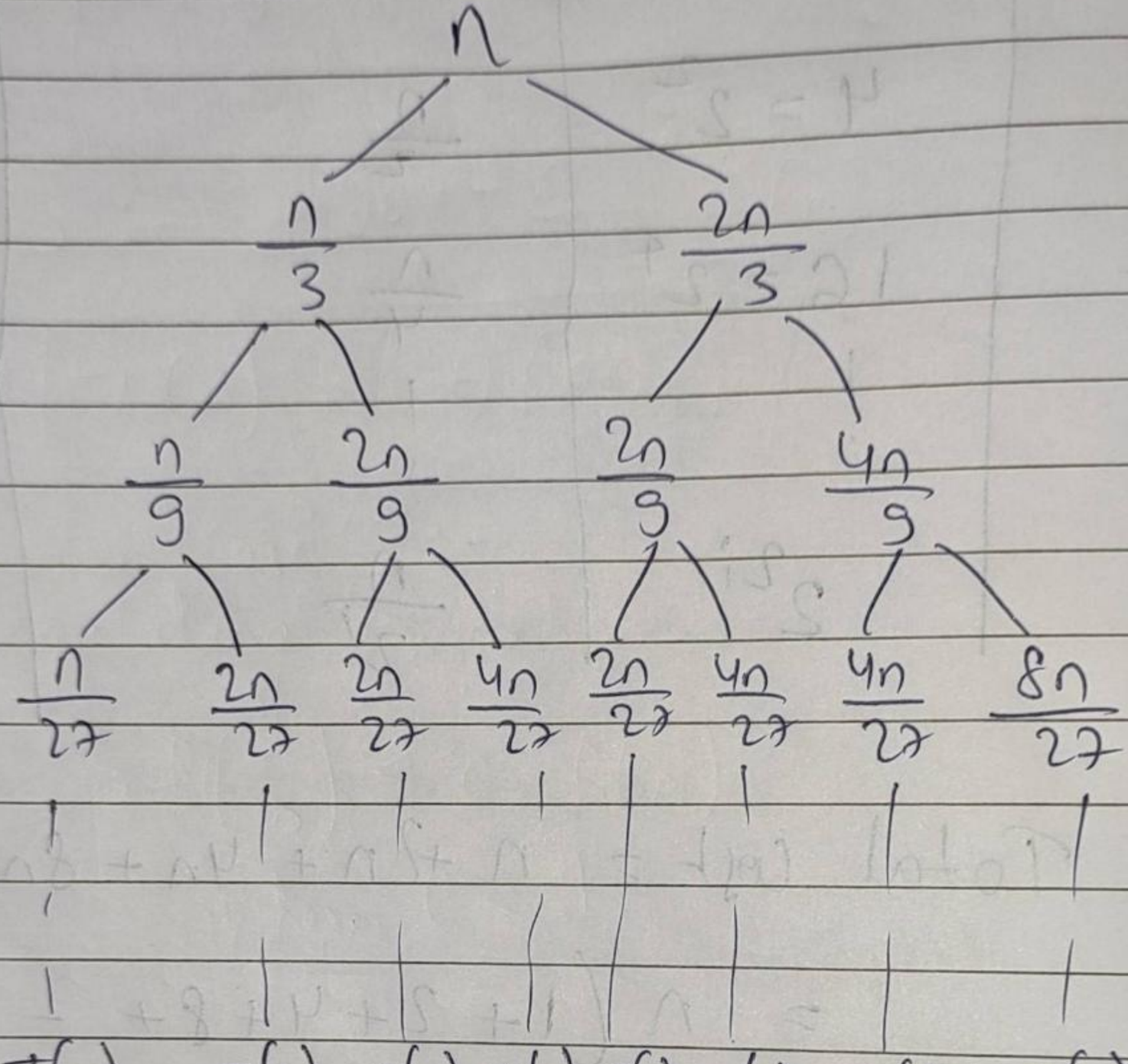
$$= n * (n - 1)$$

$$= n^2 - n$$

$$\Rightarrow T(n) = O(n^2)$$

Q. $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n$

Sol →



Level	No. of Node	Problem size	Cost
0	1	n	n
1	2	$\frac{2}{3}n$	n
2	4	$\frac{2^2}{3^2}n$	n
⋮	⋮	⋮	⋮
i	2^i	$\left(\frac{2}{3}\right)^i n$	n

$$\left(\frac{2}{3}\right)^i n = 1$$

$$n = \left(\frac{3}{2}\right)^i$$

$$\log n = i \log \frac{3}{2}$$

$$i = \frac{\log n}{\log \frac{3}{2}}$$

$$i = \log_{\frac{3}{2}} n$$

Total cost $= n + n + n + \dots n$

$= i \cdot n$

$= n \cdot \log_{\frac{3}{2}} n$

$\therefore T(n) = O\left(n \cdot \log_{\frac{3}{2}} n\right)$

4) Master's Method :-

$$\boxed{T(n) = aT\left(\frac{n}{b}\right) + f(n)}$$

$a \geq 1$, $b > 1$ are constant

Case 1 :- $n^{\log_b a - \epsilon} = f(n)$.

$$\boxed{T(n) = \Theta\left(n^{\log_b a}\right)}$$

Case 2 :- $n^{\log_b a} = f(n)$

$$\boxed{T(n) = \theta(n^{\log_b a} \cdot \log n).}$$

Case 3 :- $n^{\log_b a + \varepsilon} = f(n)$

$$\boxed{T(n) = \theta(f(n))}$$

for Case 3 :-

Regulating Condition :-

$$a \cdot f\left(\frac{n}{b}\right) \leq c \cdot f(n)$$

where $c < 1$.

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

Sol → On Comparing it with

$aT\left(\frac{n}{b}\right) + f(n)$, we get

$a = 4, \quad b = 2, \quad f(n) = n$

$n^{\log_b a} = n^{\log_2 4} = n^2$

⇒ Case 1 will be applied

⇒ $T(n) = \theta(n^{\log_2 4})$

$T(n) = \theta(n^2)$

Q $T(n) = 8T\left(\frac{n}{2}\right) + 3n^2$

Sol → $a = 8, \quad b = 2, \quad f(n) = 3n^2$

$$n^{\log_b a} = n^{\log_2 8} = n^3$$

$\Rightarrow$ Case #1 will be applied

$\Rightarrow T(n) = \theta(n^{\log_2 8})$

$$T(n) = \theta(n^3)$$

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

Sol → $a = 4, \quad b = 2, \quad f(n) = n^2$

$$n^{\log_2 4} = n^2 = f(n)$$

$\Rightarrow$ Case 2 will be applied.

$\Rightarrow T(n) = \theta(n^{\log_2 4} \cdot \log n)$

$$T(n) = \theta(n^2 \log n)$$

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

Sol → $a = 2, \quad b = 2, \quad f(n) = n$

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

$\Rightarrow$ Case 2 will be applied

$\Rightarrow T(n) = \theta(n^{\log_2 2} \cdot \log n) \Rightarrow T(n) = \theta(n \cdot \log n)$.

Q. $T(n) = 4T\left(\frac{n}{2}\right) + n^3$.

Sol → $a = 4, \; b = 2, \; f(n) = n^3$

$$n^{\log_b a} = n^{\log_2 4} = n^2$$

⇒ Case 3 will be applied

$$a \cdot f\left(\frac{n}{b}\right) \leq c \cdot f(n).$$

$$\Rightarrow 4 \cdot f\left(\frac{n}{2}\right) \leq c \cdot f(n)$$

$$\Rightarrow 4 \cdot \left(\frac{n}{2}\right)^3 \leq c \cdot n^3$$

$$= 4 \cdot \frac{n^3}{8} \leq c \cdot n^3$$

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

So, ~~ret~~ regulating condition is satisfied

So, $T(n) = \theta(n^3)$.

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

Sol → $a = 9, \; b = 3, \; f(n) = n^3$

$$n^{\log_b a} = n^{\log_3 9} = n^2$$

⇒ Case 3 will be applied

$$\Rightarrow a \cdot f\left(\frac{n}{b}\right) \leq c \cdot f(n)$$

$$\Rightarrow 9 \cdot f\left(\frac{n}{3}\right) \leq c \cdot f(n).$$

$$\Rightarrow 9 \cdot \left(\frac{n}{3}\right)^3 \leq c \cdot n^3$$

$$\Rightarrow 9 \cdot \frac{n^3}{27} \leq c \cdot n^3$$

$$\Rightarrow c = \frac{1}{3} < 1$$

So, regulating condition is satisfied.

$$\Rightarrow T(n) = \theta(n^3)$$

Insertion Sort

Algorithm :-

	Cost	Times.
1. for $j = 2$ to n.lenght {	C_1	n
2. Key = $n[j]$	C_2	$n-1$
3. $i = j-1$	C_3	$n-1$
4. while ($i > 0$ && $A[i] \geq key$){	C_4	$\sum_{j=2}^{n} t_j$
5. $A[i+j] = A[i]$.	C_5	$\sum_{j=2}^{n} t_{j-1}$
6. $i = i-1$; }	C_6	$\sum_{j=2}^{n} t_{j-1}$
7. $A[i+1] = key$ }.	C_7	$(n-1)$

Time Complexity :-

$$C_1 n + C_2(n-1) + C_3 \cdot (n-1) + C_4 \cdot \sum_{j=2}^{n} t_j +$$

$$C_5 \sum_{j=2}^{n} t_{j-1} + C_6 \sum_{j=2}^{n} t_{j-1} + C_7 (n-1).$$

Q. Perform the insertion sort on array
A = [6, 3, 5, 7, 2, 4].

Sol →

6	3	5	7	2	4
1	2	3	4	5	6

j = 2 :-

key = A[2] i.e. 3

i = 1 :-

i > 0 && A[i] > key.

A[2] ← A[i].

i = 0 :-

i > 0 && A[i] > key → false

A[i] ← key.

3	6	5	7	2	4
1	2	3	4	5	6

j = 3 :-

key = A[3] i.e. 5

i = 2 :-

i > 0 && A[2] > key.

A[3] ← A[2].

i = 1 :-

i > 0 && A[i] > key → false

A[2] ← key.

3	5	6	7	2	4
1	2	3	4	5	6

j = 4 :-

key = A[4] = 7

i = 3 :-

3 > 0 && A[3] > key ← false

A[4] ← key

3	5	6	7	2	4
1	2	3	4	5	6

j = 5 :-

key = A[5] = 2

i = 4 :-

4 > 0 && A[4] > key

A[5] ← A[4].

i = 3 :-

3 > 0 && A[3] > key

A[4] ← A[3]

i = 2 :-

2 > 0 && A[2] > key

A[3] ← A[2]

i = 1 :-

1 > 0 && A[1] > key

A[2] ← A[1]

i = 0 :

0 > 0 false

A[1] ← key.

2	3	5	6	7	4

j = 6 :-

i = 5 :-

5 > 0 && A[5] > key:
A[6] ← A[5]

i = 4 :-

4 > 0 && A[4] > key
A[5] ← A[4]

i = 3 :-

3 > 0 && A[3] > key
A[4] ← A[3]

i = 2

2 > 0 && A[2] > key ← false
A[3] ← key.

2	3	4	5	6	7

So, Insertion Sort is performed & array is sorted in ascending order.