Q1. Find the O notation for the following:

a) $f(n) = 5n^3 + n^2 + 6n + 2$

Solⁿ.

For $n \geq 2$ $\quad 5n^3 + n^2 + 6n + 2 \leq 5n^3 + n^2 + 6n + n \leq 5n^3 + n^2 + 7n$

For $n^2 \geq 7n$. $\quad 5n^3 + n^2 + 7n \leq 5n^3 + n^2 + n^2 \leq 5n^3 + 2n^2$

$5n^3 + 2n^2 \leq 5n^3 + n^3 \leq \underline{6}n^3$

thus $c = 6 \quad n_0 = 2$

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

b) $f(n) = 4n^3 + 2n + 3$

Solⁿ For $n \geq 3$ $\quad 4n^3 + 2n + 3 \leq 4n^3 + 2n + n \leq 4n^3 + 3n$.

For $n^3 \geq 3n$ $\quad 4n^3 + 3n \leq 4n^3 + n^3 \leq 5n^3$

Thus $c = 5 \quad n_0 = 3$

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

c) $f(n) = 10n^2 + 7$

for $n \geq 7$ $\quad 10n^2 + 7 \leq 10n^2 + n <$

for $n \leq n^2$ $\quad 10n^2 + n \leq 10n^2 + n^2 \leq 11n^2$

Thus $c = 11 \quad n_0 = 7$

So $f(n) = O(n^2$.

d) $f(n) = 2^n + 6n^2 + 3n$

For $n^2 \geq 3n$ $\quad 2^n + 6n^2 + 3n \leq 2^n + 6n^2 + n^2 \leq 2^n + 7n^2$

$2^n \geq n^2$ $\quad 2^n + 7n^2 \leq 2^n + 7 \cdot 2^n \leq 8 \cdot 2^n$.

$n_0 = 4 \quad C = 8$

Thus $f(n) = O(2^n)$ Ans

Q 2. Show that $27n^2 + 16n + 25 = \Omega(n^2)$

Solⁿ. Let $f(n) = 27n^2 + 16n + 25$

$27n^2 \leq 27n^2 + 16n + 25 \quad \forall n.$

For Ω notation

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

$g(n) = n^2$

$c = 27$ Ans

Q 3. Find the Ω notation for following function

a) $5n^3 + n^2 + 3n + 2.$

Solⁿ. $f(n) = 5n^3 + n^2 + 3n + 2.$

$5n^3 \leq 5n^3 + n^2 + 3n + 2 \quad \forall n.$

So $f(n) = \Omega(n^3)$

$c = 5$

b) $f(n) = 3^n + 6n^2 + 3n$

$3^n \leq 3^n + 6n^2 + 3n \quad \forall n$

So $f(n) = \Omega(3^n) \quad n \geq 1$

c) $f(n) = 4 * 2^n + 3n$

$4 * 2^n < 4 \cdot 2^n + 3n \quad \forall n.$

$c = 4 \quad f(n) = \Omega(2^n)$

Q. 4 Find Θ notation for the given exponential function.

$f(n) = 3 * 2^n + 4n^2 + 5n + 2$

Solⁿ.

$f(n) = 3 * 2^n + 4n^2 + 5n + 2.$

For finding Θ notation first we found lower bound for $f(n)$

$3 * 2^n \leq 3 * 2^n + 4n^2 + 5n + 2. \quad \forall n \geq n_0$

$c_1 = 3 \quad g(n) = 2^n \quad f(n) = \Omega(2^n)$

Now find the upper bound.

$3 * 2^n + 4n^2 + 5n + 2 \leq 3 * 2^n + 4n^2 + 5n + n \quad (n \geq 2)$

$3 * 2^n + 4n^2 + 6n \leq 3 * 2^n + 4n^2 + n^2 \quad (n^2 \geq 6n$ or $n \geq 6)$

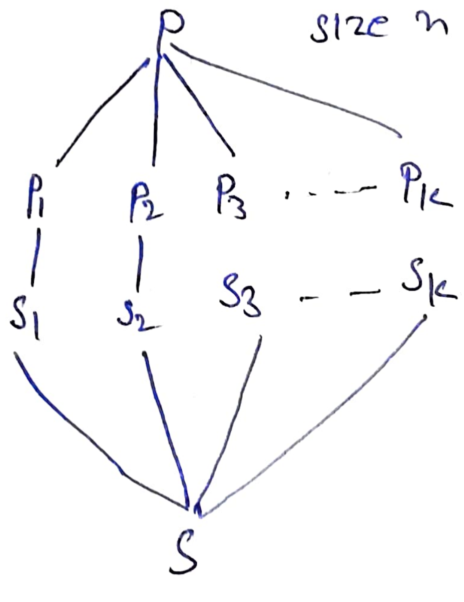
$3 * 2^n + 5n^2 \leq 3 * 2^n + 5 * 2^n \quad (n \geq 6)$

$\leq 8 \cdot 2^n$

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

Thus $f(n) = \Theta(2^n)$ Ans.

Divide and Conquer → A stategy is a approach or design for solving a problem. For solving any problem we can use this approach. If a problem of some size n is given, then we can break this problem into smaller k subproblems $P_1, P_2, P_3 \ldots P_k$. These subproblems can be solved to obtain their solution $S_1, S_2, S_3 \ldots S_k$ and finally these solutions can be combined to get the solution for the original problem. If the subproblem is large then do the same thing i.e divide it into subproblems, solve them and combined the result.



the subproblems should be same as original problem.

It is recursive in nature i.e we recursively solve it.

There should be some method to combine the results.

```
DAC (P)
{
   if small(P)
   { S(P)
   }
   else
     { divide P into P1, P2, P3 -- Pk
       Apply DAC(P1) DAC(P2) --- DAC(Pk)
       Combine ( DAC(P1), DAC(P2) -- DAC(Pk))
     }
```

1. Binary Search
2. Finding Maximum and Minimum
3. Merge Sort
4. Quick Sort
5. Strassen's Matrix multiplication.

Recurrence Relation →

when an algorithm contains a recursive call to itself, its running time can be described by a recurrence equation.

e.g $T(n) = \begin{cases} O(1) & \text{if } n \le 1 \\ 2T(n/2) + n & \text{otherwise} \end{cases}$

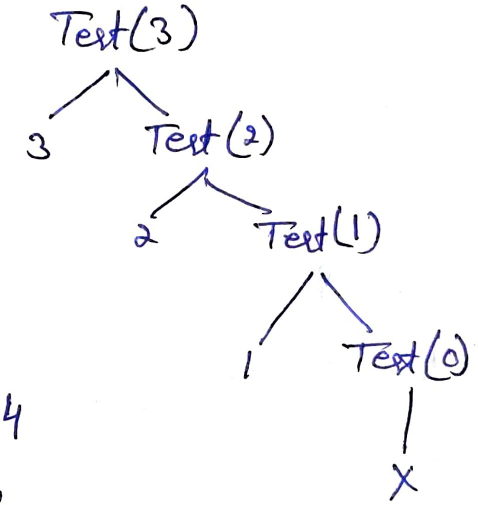
Solving Recurrence Relation —

1. Substitution Method / Iteration Method.
2. Master's Method
3. Recurrence Tree method.

① Substitution Method

```
void Test(int n)
{
    if(n>0)
    {
        Printf("%d", n)  —
        Test (n-1);
    }
}
```

Test(3) $\Rightarrow$



No of times it is printing 3

No of time it is calling itself 4

i.e n+1 calls

$f(n) = n+1$

$\Rightarrow O(n)$

$T(n)$ –

```
Void Test(int n)
{
    if (n>0)
    {
        Printf("%d", n);      — 1
        Test (n-1)            — T(n-1)
    }
}
```

$$T(n) = \begin{cases} 1 & n=0 \\ T(n-1)+1 & n>0 \end{cases}$$

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

Ans

$\cancel{T(n) = (T(n-1)+P)}$

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

$\because T(n) = T(n-1) + 1$

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

substitute $T(n-1)$

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

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

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

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

continue k times

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

Assume $n-k=0$

i.e $n=k$

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

$T(n) = T(0) + n$

$\boxed{T(n) = 1 + n} \Rightarrow O(n)$

② Decreasing function

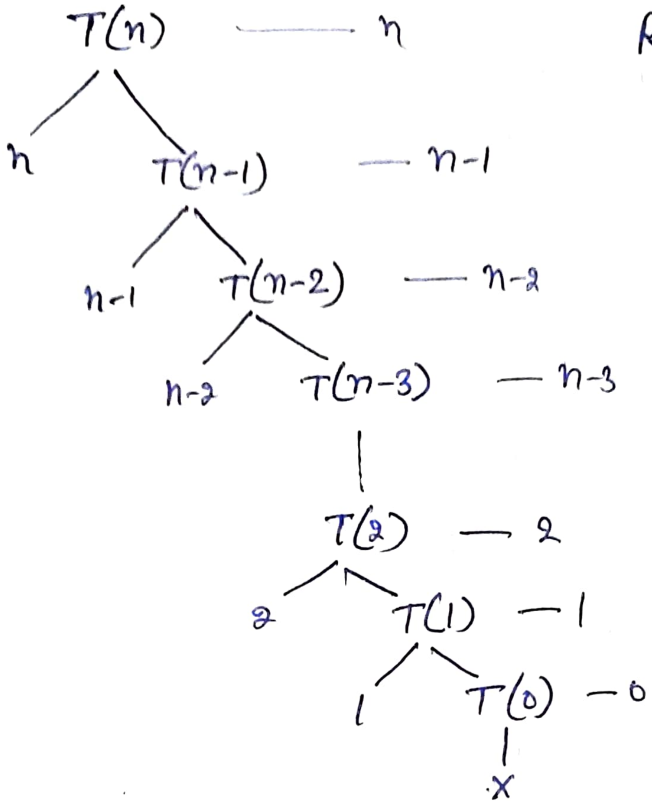
$T(n) = T(n-1) + n$

$$T(n) = \begin{cases} 1 & n=0 \\ T(n-1)+n & n>0 \end{cases}$$

```
T(n) <- Void Test (int n)
        {
1           if(n>0)
            {
n+1             for (i=0; i<n; i++)
                {
n                   Printf("%d", n)
                }
T(n-1)          Test(n-1)
            }
        }
```

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

Recurrence Tree



$$0+1+2+\dots+(n-1)+n. = \frac{n(n+1)}{2}.$$

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

Substitution Method

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

$$T(n) = T(n-1)+n \quad -(1)$$

$\because T(n) = T(n-1)+n$

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

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

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

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

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

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

k times

$$T(n) = T(n-k)+(n-(k-1))+(n-(k-2))+\dots(n-1)+n \quad -(4)$$

Assume $n-k$ become 0 so $n=k$

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

$$T(n) = T(0)+1+2+3+\dots(n-1)+n$$

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

$$\Rightarrow \theta(n^2)$$

Q
$$T(n) = \begin{cases} 1 & \text{if } n=1 \\ T(n-1) * n & \text{if } n>1 \end{cases}$$

$T(n) = T(n-1) * n$ $\quad\because T(n) = T(n-1) * n$

$T(n) = T(n-2) * (n-1) * n$ $\quad\therefore T(n-1) = [T(n-2) * (n-1)] * n$

$\quad T(n-1) = T(n-2) * (n-1) * n$

$T(n) = T(n-3) * (n-2) * (n-1) * n.$

$\vdots$

$= T(n-(n-1)) * (n-(n-2)) * (n-(n-3)) * \cdots * (n-1) * n.$

$= T(1) * 2 * 3 * 4 \times \cdots (n-1) * n$

$= n!$

$= O(n^n)$

$\because \quad n * (n-1) * (n-2) * (n-3) * \cdots * (n-(n-1))$

$n \quad n \quad n \quad n \quad n.$

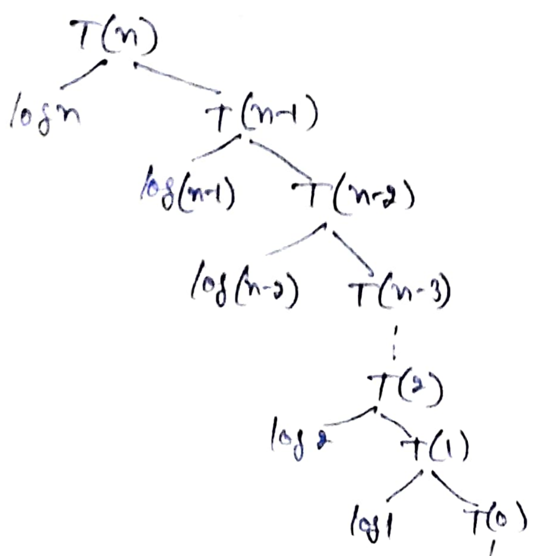
every where write max value $\Rightarrow n^n$.

(this is called upper bound.)

Q. $$T(n) = \begin{cases} 1 & \text{if } n = 0 \\ T(n-1) + \log(n) & \text{if } n \geq 1 \end{cases}$$

$$T(n) = T(n-1) + \log(n)$$
$$= T(n-2) + \log(n-1) + \log(n)$$
$$= T(n-3) + \log(n-2) + \log(n-1) + \log(n)$$
$$\downarrow \; \underline{n-1}$$
$$= T(n-(n-1)) + \log(n-(n-2)) + \log(n-(n-3)) + \log(n-(n-4)) + \dots + \log(n-1) + \log(n)$$
$$= T(1) + \log(2) + \log(3) + \log(4) + \dots + \log(n-1) + \log(n)$$
$$= 1 + \log(2 \times 3 \times 4 \times \dots \times n-1 \times n)$$
$$T(n) = 1 + \log(n!)$$
$$= 1 + \log n^n \qquad \because \text{ for } n! \; n^n \text{ is is the upper bound.}$$
$$= 1 + n \log n$$
$$= O(n \log n)$$

$T(n) = 1$ if $n = 0$



T(n) Void Test (int n)

```
{
  if (n>0)
  {  for(i=1; i<n; i=i*2)         log n
     {  printf("%d", i);
     }
     Test(n-1);                    T(n-1)
  }
}
```

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

$$T(n) = \begin{cases} 1 & \text{if } n=0 \\ T(n-2)+\log(n) & \text{if } n>0 \end{cases}$$

$T(n) = T(n-2) + \log(n)$ $\quad$ $T(n-2) = T(n-4) + \log(n-2)$

$T(n) = T(n-4) + \log(n-2) + \log(n)$

$T(n) = T(n-6) + \log(n-4) + \log(n-2) + \log(n)$

| k time

$T(n) = T(n-2k) + \log(n-(2k-2)) + \log(n-(2k-4)) + \log(n-(2k-6))$
$+ \dots + \log(n-0)$

$T(n) = T(n-n) + \log(n-(n-2)) + \log(n-(n-4)) + \log(n-(n-6))$
$+ \dots + \log n.$

$\boxed{\begin{aligned} n-2k&=0 \\ n&=2k \\ k&=n/2. \end{aligned}}$

$T(n) = T(0) + \log(2) + \log(4) + \log(6) + \dots + \log(n)$

$= 1 + \log(2\times1) + \log(2\times2) + \log(2\times3) + \dots + \log(2*k)$

$= 1 + (\log 2 + \log 1) + (\log 2 + \log 2) + (\log(2) + \log(3)) + (\log(2) + \log(4))$
$+ \dots + (\log(2) + \log(k))$

$= 1 + k\underbrace{\log_2 2}_{=1} + \log 1 + \log 2 + \log 3 + \log 4 + \dots + \log k$

$= 1 + k + \log(1+2+3+\dots k)$ $\quad$ $\boxed{\log(a*b) = \log a + \log b}$

$= 1 + k + \log(k!)$

$= 1 + k + \log k^k$

$= 1 + k + k\log k$

$= 1 + \frac{n}{2}\log\frac{n}{2} \Rightarrow O(n\log n)$

Q/ $$T(n) = \begin{cases} 1 & \text{if } n = 0 \\ T(n-2)+n^2 & \text{if } n > 0 \end{cases}$$

$$T(n) = T(n-\overset{2\times1}{2}) + n^2$$

$$= T(n-\overset{2\times2}{4}) + (n-2)^2 + (n-0)^2$$

$$= T(n-\overset{2\times3}{6}) + (n-4)^2 + (n-2)^2 + (n-0)^2$$

⋮ k times.

$\boxed{\begin{array}{l} n-2k=0 \\ n=2k \\ k=n/2 \end{array}}$

$$= T(n-2k) + (n-(2k-2))^2 + (n-(2k-4))^2 + (n-(2k-6))^2$$
$$+ \cdots (n-2)^2 + (n-0)^2$$

$$= T(n-n) + (n-(n-2))^2 + (n-(n-4))^2 + (n-(n-6))^2$$
$$+ \cdots (n-2)^2 + (n-0)^2$$

$$= T(0) + 2^2 + 4^2 + 6^2 + 8^2 + \cdots + \left(2 * \frac{n}{2}\right)^2$$

$$= 1 + 2^2\left(1 + 2^2 + 3^2 + 4^2 + \cdots \left(\frac{n}{2}\right)^2\right)$$

$$= 1 + 2^2(1 + 2^2 + 3^2 + 4^2 + \cdots k^2)$$

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

$$= k^3 = \left(\frac{n}{2}\right)^3 = \frac{1}{8}n^3$$

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