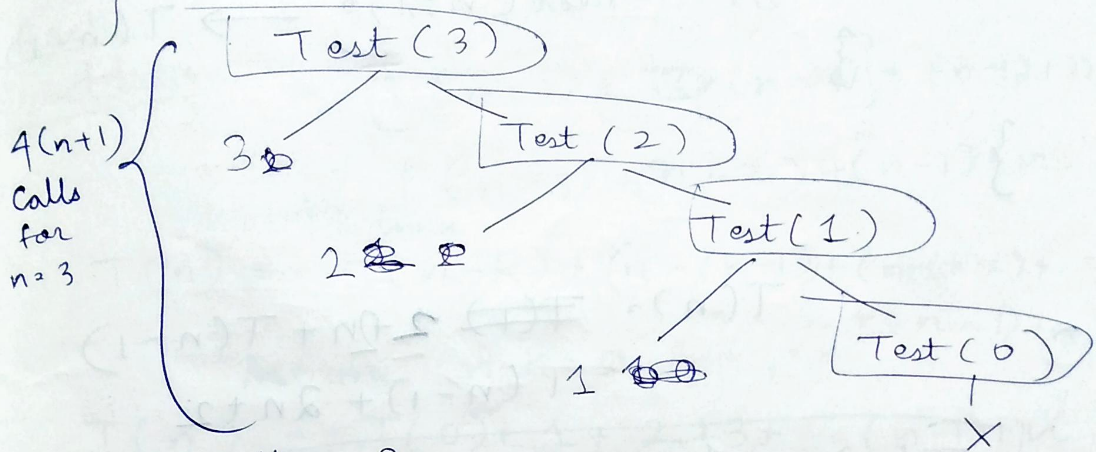
Recurrence Relation ($T(n) = T(n-1)+1$)

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

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

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



n+1 calls for n.

$f(n) = n+1$ calls

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

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

Substitute $T(n-1)$

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

⋮ continue for k times

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

Assume $n-k=0$, $n=k$

$$T(n) = T(0) + n$$
$$T(n) = 1+n.$$
$$O(n) \Rightarrow \boxed{O(n)}$$

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

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

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

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

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

<u>Tree Method</u>

$T(n) \to n,\ T(n-1)$

$T(n-1) \to n-1,\ T(n-2)$

$T(n-2) \to n-2,\ T(n-3)$

$\vdots$

$T(2) \to 2,\ T(1)$

$T(1) \to 1,\ T(0)$

$T(0) \to X$

n

n-1

n-2

⋮

2

1

0

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

$\boxed{O(n^2)}$

Another way

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

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

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

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

Back Substitution

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

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

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

k times

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

Assume $n-k=0$

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

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

$= O(n^2)$.

3) $\boxed{T(n) \rightarrow T(n-1) + \log(n)}$

$T(n)$

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

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

~~T(~~

Tree

$T(n) \rightarrow \log n,\ T(n-1)$

$T(n-1) \rightarrow \log(n-1),\ T(n-2)$

$T(n-2) \rightarrow \log(n-2),\ T(n-3)$

$\vdots$

$T(2) \rightarrow \log 2,\ T(1)$

$T(1) \rightarrow \log 1,\ T(0)$

(i) $\log n +$

$\log(n-1) +$

— —

$\log 2 + \log 1$

$\log[n(n-1)(n-2) \ldots 2 \cdot 1]$

$= \log n!$ $\quad (n^n > n!)$

$O(n \log n)$

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

Back Substitution

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

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

| k times

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

$+ \ldots 1$

$\boxed{n-k=0}$

$= 1 + \log n!$

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

4) $T(n) = T(n-1) + 1$ ... $O(n)$

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

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

$T(n) = T(n-1) + n^2$ ... $O(n^3)$

$T(n) = T(n-2) + 1$ ... $\frac{n}{2}$ — $O(n)$

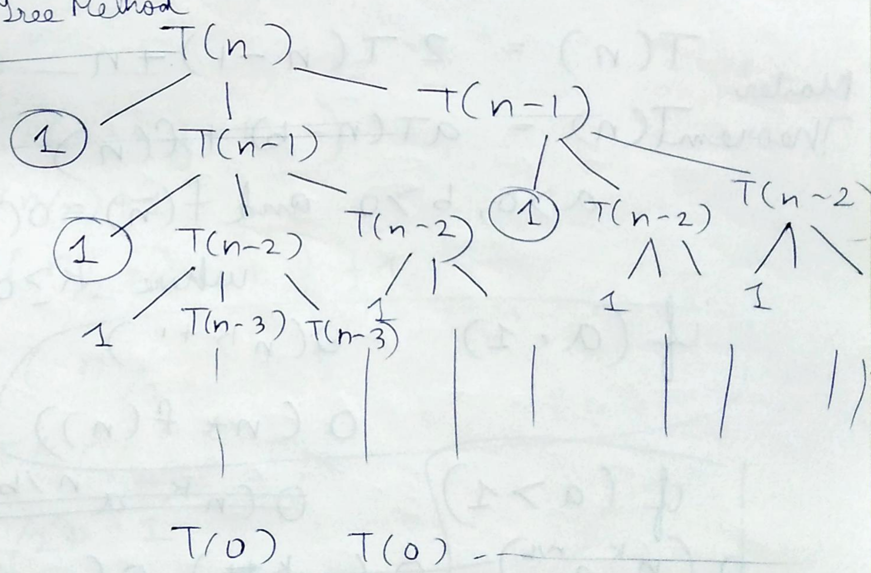
$T(n) = T(n-100) + n$ — $O(n^2)$

---

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

```
Algorithm Test (int n)
{    if (n>0)
     {  printf ("%d", n);
        Test (n-1);
        Test (n-1);
     }
}
```

Tree Method



(2^0) 1

(2^1) 2

(2^2) 4

8

⋮

k times

$= 2^0 + 2^1 + \dots 2^k$

$= 2^{k+1} - 1$

$O(2^n)$

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

$$T(n) = 2^k\,T(n-k) + 2^k + 2^{k-1} + \ldots\ 2 + 1$$

Assume $n-k=0$

$$= 2^n + 2^k - 1$$
$$= 2^{n+1} - 1$$

$$\boxed{O(2^n)}$$

$T(n) = 2\,T(n-1) + 1$ — $O(2^n)$

$T(n) = 3\,T(n-1) + 1$ — $O(3^n)$

$T(n) = 2\,T(n-1) + n$ — $O(n\,2^n)$

Master Theorem $T(n) = aT(n-b) + f(n)$

$a > 0$, $b > 0$ and $f(n) = O(n^k)$
where $k \geq 0$

if $(a = 1)$ $O(n^{k+1})$

$O(n * f(n))$

if $(a > 1)$ $\cancel{O(n^k\,a^{n/b})}$

$O(n^k\,a^{n/b})$ $\cancel{O(n^{k})}$ $\cancel{O(n * f(n))}$

if $(a < 1)$ $O(n^k)$ $O(f(n))$

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

Algorithm Test (int n) — $T(n)$

```
{   if (n > 1)
    {   printf ("%d", n);  — (1)
        Test (n/2);        — T(n/2)
    }
}
```

$$T(n) = \begin{cases} 1, & n = 1 \\ T(n/2) + 1, & n > 1 \end{cases}$$

Tree

$T(n)$ — 1

1 — $T(n/2)$ — 1

1 — $T\left(\frac{n}{2^2}\right)$ — 1

1 — $T\left(\frac{n}{2^3}\right)$ — 1

$T\left(\frac{n}{2^k}\right)$ — 1

$\boxed{\text{K steps}}$

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

$$n = 2^k$$

$$k = \log_2 n$$

$$O(\log n)$$

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

$$= \left[T(n/2^2) + 1\right] + 1$$

$$= T(n/2^3) + 3$$

$$= T(n/2^k) + k$$

$$n/2^k = 1$$

$$\therefore k = \log n$$

$$T(n) = T(1) + \log n \qquad \cancel{O(\log)}$$

$$O(\log n)$$

2) $\boxed{T(n) = T(n/2) + n}$

$$T(n) = \begin{cases} 1 & n = 1 \\ T(n/2) + n & n > 1 \end{cases}$$

$$T(n) \to T(n/2),\ n \quad \text{—} \quad (n)$$

$$T(n/2) \to T\left(\frac{n}{2^2}\right),\ n/2 \quad \text{—} \quad \left(\frac{n}{2}\right)$$

$$T\left(\frac{n}{2^2}\right) \to T\left(\frac{n}{2^3}\right),\ \frac{n}{2^2} \quad \text{—} \quad \frac{n}{2^2}$$

$$\vdots$$

$$T\left(\frac{n}{2^k}\right) \quad \frac{n}{2^{k-1}} \quad \text{—} \quad \frac{n}{2^k}$$

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

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

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

$\boxed{T(n) = n}$ $O(n)$

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

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

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

$$= T(n/2^k) + n/2^{k-1} + \cdots \frac{n}{2^2} + \frac{n}{2^1}$$

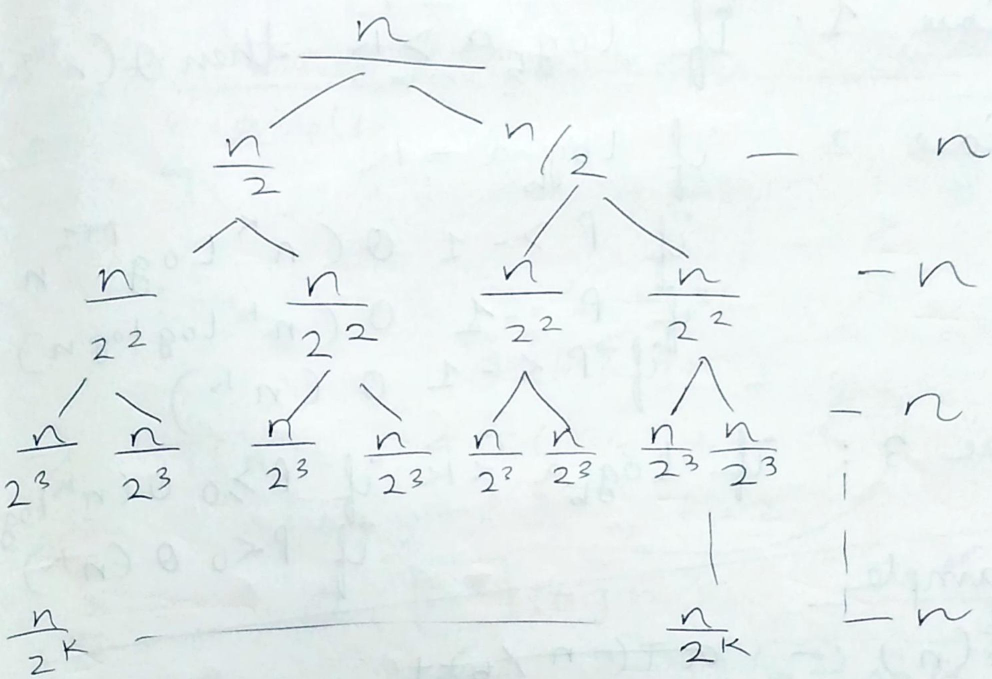
$n = 2^k$, $k = \log n$

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

$= 1 + 2n$

---

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



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

$(n = k)$

Masters

$T(n) = aT(n/b) + f(n)$

$a \geq 1,\ b > 1,\ f(n) = \theta(n^k \log^p n)$

Find TC :-

```
p = 0
for(i = 1; p <= n; i++)
    p = p + i
```

i	p
1	0+1=1
2	1+2=3
3	1+2+3
:	
k	1+2+3+4+ .. +k

Assume $p > n$,

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

$k^2 > n$ $\quad O(\sqrt{n})$

$k > \sqrt{n}$

Master's Theorem

$$T(n) = aT(n/b) + f(n)$$

$a \geq 1$ $\quad f(n) = \theta(n^k \log^p n)$

$b > 1$

Case 1: if $\log_b a > k$ then $\theta(n^{\log_b a})$

Case 2: if $\log_b a = k$

if $p > -1$ $\theta(n^k \log^{p+1} n)$

if $p = -1$ $\theta(n^k \log \log n)$

if $p < -1$ $\theta(n^k)$

Case 3: if $\log_b a < k$ if $p \geq 0$ $\theta(n^k \log^p n)$

if $p < 0$ $\theta(n^k)$

Example

$\boxed{\text{Case 1}}$

$T(n) :-$ $\cancel{aT(n/b)} +$

$2T(n/2) + 1$

$a = 2, b = 2,$

$1 = n^0 \log^0 n$

$k = 0, p = 0$

$\Rightarrow \log_b a = \log_2 2 = 1 > k \to 0$

$\theta(n^{\log_2 2}) \approx \theta(n)$

$T(n) = 8T(n/2) + n$

$= \theta(n^3)$

$T(n) = 8T(n/2) + n^2$

$\theta(n^3)$

$T(n) = 8T(n/2) + n\log n$

$a = 8,\ b = 2,\ k = 1,\ p = 1$

$\log_b a = 3 > 1(k)$

$\theta(n^3)$

Case 2 Examples

$p > -1$

→ $T(n) = 2T(n/2) + n$

$\log_2 2 = 1 = 1(k)$

$\theta(\boxed{n}\log n)$

→ $T(n) = 4T(n/2) + n^2$

$\log_2 4 = 2,\ k = 2$

$\theta(\boxed{n^2}\log n)$

→ $T(n) = 4T(n/2) + \boxed{n^2\log n}$

$\log_2 4 = 2,\ k = 2,\ p = 1$

$\theta(\boxed{n^2\log n}\log n)$

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

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

Case 1 :-

$T(n) = 2T(n/2) + 1 \quad n^0 \qquad \Theta(n)$

$T(n) = 4T(n/2) + 1 \qquad \Theta(n^2)$

$T(n) = 4T(n/2) + n \qquad \Theta(n^2)$

$T(n) = 8T(n/2) + n^2 \qquad \Theta(n^3)$

$T(n) = 16T(n/2) + n^2 \qquad \Theta(n^4)$

Case 3

$T(n) = T(n/2) + n \qquad \Theta(n)$

$T(n) = 2T(n/2) + n^2 \qquad \Theta(n^2)$

$T(n) = 2T(n/2) + n^2 \log n \qquad \Theta(n^2 \log n)$

$T(n) = 4T(n/2) + n^3 \log^2 n \qquad \Theta(n^3 \log^2 n)$

$T(n) = 2T(n/2) + \dfrac{n^2}{\log n} \qquad \Theta(n^2)$

Case 2

$T(n) = T(n/2) + 1 \qquad \Theta(\log n)$

$T(n) = 2T(n/2) + n \qquad \Theta(n \log n)$

$T(n) = 2T(n/2) + n \log n \qquad \Theta(n \log^2 n)$

$T(n) = 4T(n/2) + n^2 \qquad \Theta(n^2 \log n)$

$T(n) = 4T(n/2) + (n \log n)^2 \qquad \Theta(n^2 \log^3 n)$

$T(n) = 2T(n/2) + \dfrac{n}{\log n} \qquad \Theta(n \log \log n)$

$T(n) = 2T(n/2) + \dfrac{n}{\log^2 n} \qquad \Theta(n)$

Root Function

$$T(n) = \begin{cases} 1 & n = 2 \\ T(\sqrt{n}) + 1 & n > 2 \end{cases}$$

$$T(n) = T(\sqrt{n}) + 1$$

$$T(n) = T(n^{1/2}) + 1 \quad (1)$$

$$T(n) = T(n^{1/2^2}) + 2 \quad (2)$$

$$T(n) = T(n^{1/2^3}) + 3 \quad (3)$$

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

Assuming $n = 2^m$

$$T(2^m) = T(2^{m/2^k}) + k$$

Assume $T(2^{\frac{m}{2^k}}) = T(2^1)$

$\therefore \frac{m}{2^k} = 1$

$m = 2^k$ and $k = \log_2 m$

$n = 2^m$, $m = \log_2 n$

$\boxed{k = \log \log_2 n}$

```
void Test (int n)        — T(n)
{ if (n > 2)
  { Task                 — 1
    Test(√n);            — T(√n)
  }
}
```

$$T(n) = T(\sqrt{n}) + 1$$