→ Masters theorem of Decreasing functions

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

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

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

So, we can say

$O(n * f(n))$

If $a>1$ $\quad O(n^k a^{n/b})$

So,

Case 1:

if $(a<1)$ $\Rightarrow O(n^k)$

Case 2:

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

$O(n * f(n))$

Case 3:

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

→ Dividing function

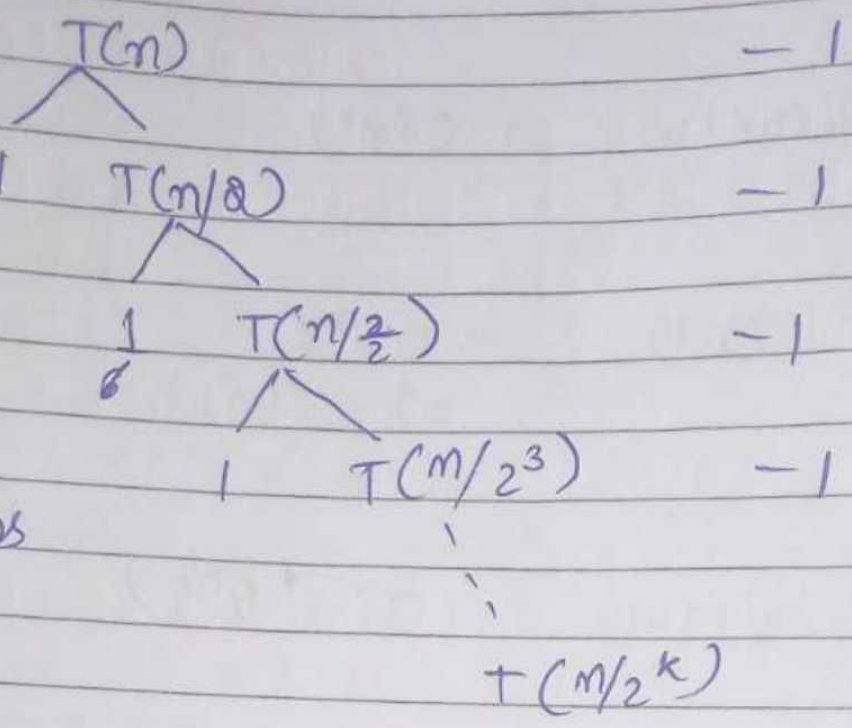
. Recurrence Relation Dividing function $T(n) = T(n/2) + 1$

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

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

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

Recursion Tree method



K steps

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

$n = 2^k$

$k = \log_2 n$

$[a^b = c \quad b = \log_a c]$

$O(\log n)$

Substitution Method

$T(n) = T(n/2) + 1$ — (1)

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

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

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

$T(n) = [T(n/2^2)] + 2$ — (2)

$T(n) = [T(n/2^3)] + 3$ — (3)

⋮

$T(n) = T(n/2^k) + k$ — (4)

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

assume $\frac{n}{2^k} = 1$ $\quad [\because n = 2^k \text{ and } k = \log n]$

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

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

$O(\log n)$

- Recurrence Relation Dividing $[T(n) = T(n/2) + n]$

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

Recursion tree Method

$T(n) = n + \frac{n}{2} + \frac{n}{2^2} + \frac{n}{2^3} + \cdots + \frac{n}{2^k}$

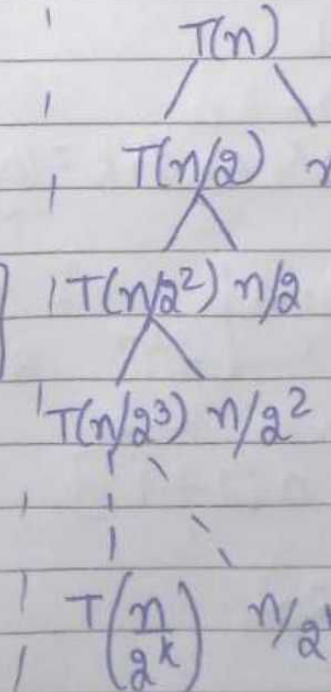
$= n\left[\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \cdots + \frac{1}{2^k}\right]$

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

$= n \times 1$

$= n$

$O(n)$



Substitution Method

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

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

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

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

$$\vdots$$

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

$$\because T\left(\frac{n}{2^k}\right) = 1$$

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

$\therefore n = 2^k$ and $k = \log n$

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

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

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

$\therefore O(n)$

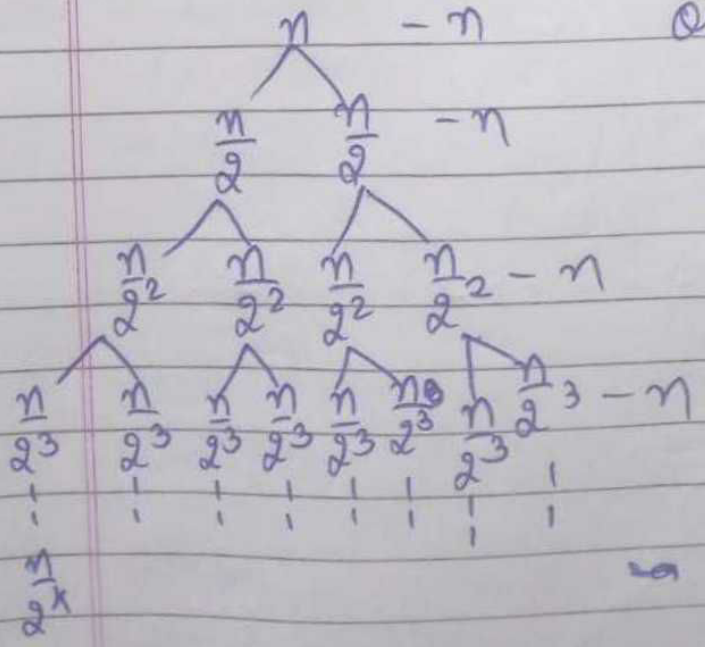
- Recurrence Relation $[T(n) = 2T(n/2) + n]$

```
void Test (int n)                    - T(n)
{
  if (n>1)
  {
    for (i=0 ; i<n ; i++)            - n
    {
      stmt ;
    }
    Test (n/2) ;                     - T(n/2)
    Test (n/2) ;                     - T(n/2)
  }
}
```

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

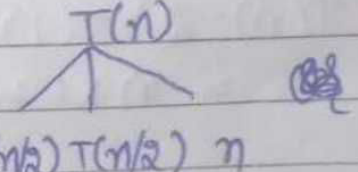
$$T(n) = \begin{cases} 1 & n=1 \\ 2T(n/2) + n & n>1 \end{cases}$$

Recursion tree Method



K-steps

or



Total = n * k ∴

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

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

$n * k = n \log n$

$O(n \log n)$

Substitution Method

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

$= 2\left[2T(n/2^2) + \frac{n}{2}\right] + n$ $\left[\because T(n) = 2T(n/2) + n,\ T(n/2) = 2T(n/2^2) + \frac{n}{2}\right]$

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

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

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

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

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

$\left[T(n/2^k) = T1,\ n/2^k = 1 \Rightarrow n = 2^k,\ k = \log n\right]$

$T(n) = 2^k T(1) + kn$

$= n \times 1 + n \log n$

$\therefore O(n \log n)$