Note for me : <u>~~falut~~</u> (1)

Consider a recursive function :

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

[for recurrence solution function name will T]

→ We don't consider condition it will take 1 unit time, if you consider it will not set any impect on time compexity. if consider, $T(n) = T(n-1) + 2$

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

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

Apply substitute method

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

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

$= T(n-2) + 2$

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

$= T(n-3) + 3$

. . . .

continue for k times

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

Assume $(n-k) = 0$

$\therefore n = k$

{I will stop 'k' when → when base case reached

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

$= T(0) + n$

$T(n) = 1 + n$

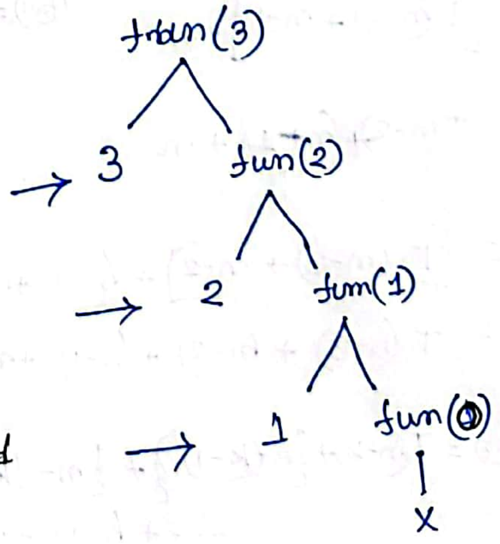
big O → $O(n)$

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

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

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

~~make a tree~~ Make a tree pass 3 to function



that means, $(1+n)$ time running the function.

Consider another example :

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

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

by using asymptotic anlaysis $T(n) = T(n-1) + n$

Prepare a recurrence equation

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

By using substitute method

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

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

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

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

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

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

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

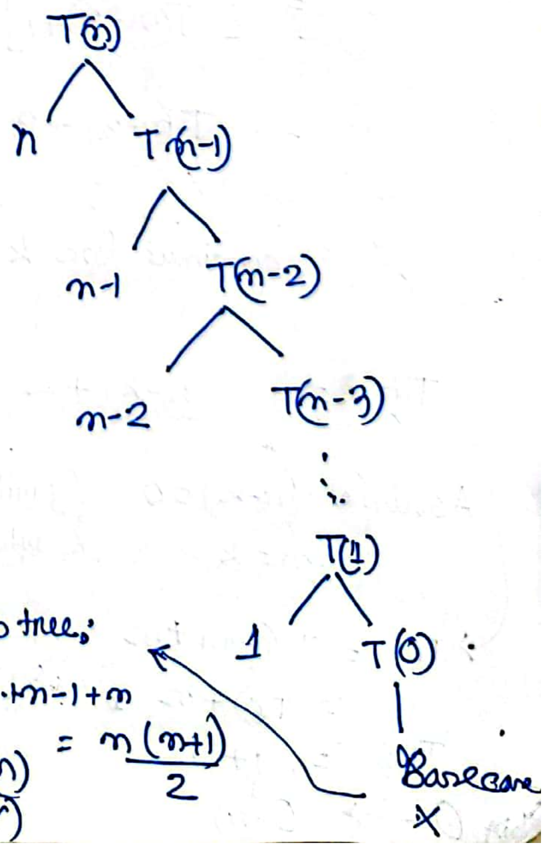
so, $T(n) = T(n-k)+\{n-(k-1)\}+\{n-(k-2)\}$
$\quad \cdots + (n-1)+n$

Assume $n-k=0$

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

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

Recurrence Tree



according To tree,

$0+1+2+\cdots+n-1+n$

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

$\Rightarrow O(n^2)$

here,
1 step more than Recurrence tree steps ~~log~~ but big (O) is $O(n^2)$ same

Another example :-

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

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

here ommit some steps accoding to Asymptotic analysis

here is the recurrence equation,

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

by substitution

$T(n) = T(n-1) + \log n$ ——— (1)

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

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

$T(n) = T(n-3) + \log(n-2) + \log(n-1) + \log n$ ——— (3)
running k times.

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

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

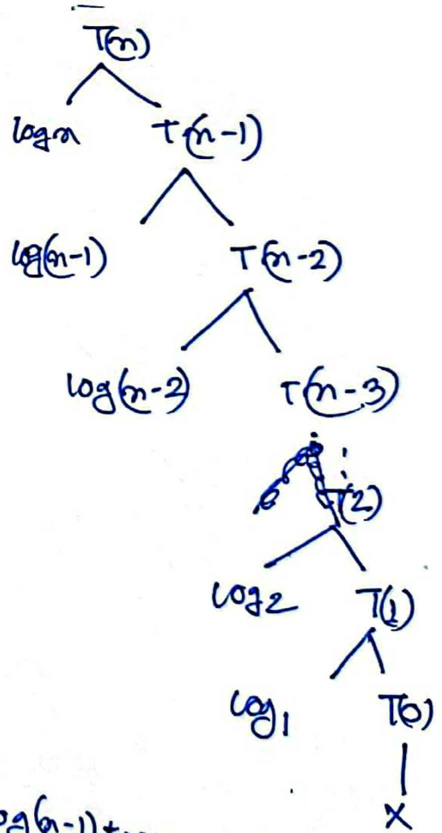
$n-k=0 \therefore n=k$ ——— (4)

$T(n) = T(0) + \log n!$

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

$O(n \log n)$

Recurence Tree



$\log n + \log(n-1) + \cdots + \log(2) + \log 1$

$= \log[n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1]$

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

So, we can write.

$T(n) = T(n-1) + 1$ ———— $O(n)$.

$T(n) = T(n-1) + n$ ———— $O(n^2)$

$T(n) = T(n-1) + n^2$ ———— $O(n^3)$

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

$T(n) = T(n-2) + 1$ ———— $O(n)$ ∵ $\boxed{n/2} \rightarrow$ No. of steps decreased.

$T(n) = T(n-100) + n$ ———— $O(n^2)$

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

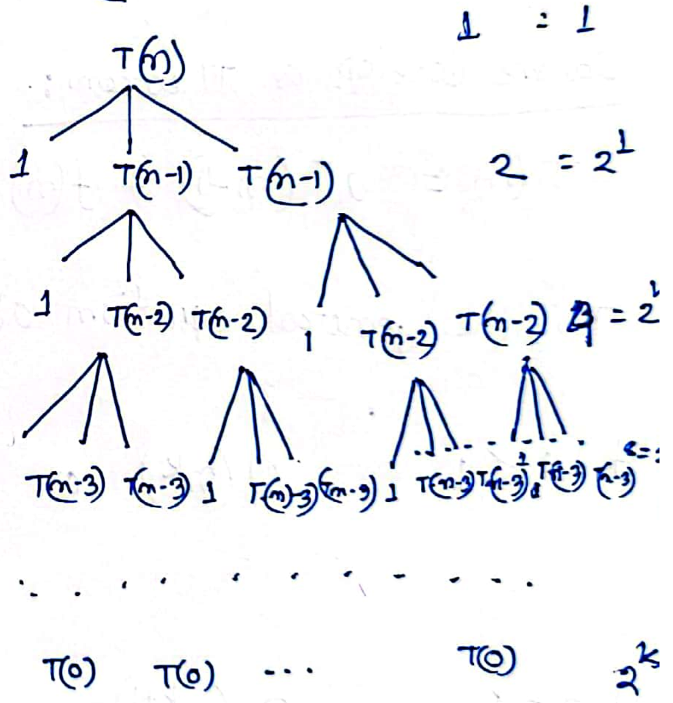
```
void Test(int n)              ——— T(n)?
{ if (n>0)
   { printf("%d",n);          ——— 1
     Test(n-1);               ——— T(n-1)
     Test(n-1);               ——— T(n-1)
   }
}
```

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

so recurrence equation;

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

corresponding tree



by using substitution method :

$T(n) = 2T(n-1)+1$ ——— (1)

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

$T(n) = 2^2T(n-2)+2+1$ ——— (2)

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

$T(n) = 2^3T(n-3)+2^2+2+1$ ——— (3)

continue for k times

$T(n) = 2^kT(n-k)+2^{k-1}+2^{k-2}+\cdots+2^2+2^1+1$ ——— (4)

$n-k=0,\ n=k$

$= 2^nT(0)+1+2+2^2+\cdots+2^{k-1}$

$= 2^n\times1+2^k-1.$

$= 2^n+2^n-1 \quad = 2^{n+1}-1 \simeq O(2^n)$

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

সিরিজের ফর্মুলা (note)

$a+ar+ar^2+\cdots+ar^k = \dfrac{a(r^{k+1}-1)}{r-1}$

here $a=1,\ r=2$ $= \dfrac{1(2^{k+1}-1)}{2-1}$

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

$n-k=0,\ n=k,\ \Rightarrow 2^{n+1}-1 \quad O(2^n)$

Scanned with CamScanner

So,

$T(n) = T(n-1) + 1$ —— $O(n)$

$T(n) = T(n-1) + n$ —— $O(n^2)$

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

$T(n) = 2T(n-1) + 1$ —— $O(2^n)$

$T(n) = 3T(n-1) + 1$ —— $O(3^n)$

$T(n) = 2T(n-1) + n$ —— ~~$O(n2^n)$~~

$O(n2^n)$

Master ~~Has~~ method for solving recurrence

$$T(n) = a\,T(n/b) + \theta(n^k \log^p n)$$

Where, $a >= 1$, $b > 1$, $k \geq 0$ & p is a real no.

case-1 :- if $a > b^k$ $\quad T(n) = \theta(n^{\log_b a})$

case-2 : if $a = b^k$

$p < -1 \quad T(n) = \theta(n^{\log_b a} \log n)$

$p = -1 \quad T(n) = \theta(n^{\log_b a} \log^2 n)$

$p > -1 \quad T(n) = \theta(n^{\log_b a} \log^{p+1} n)$

case-3 : if $a < b^k$

$p < 0 \quad T(n) = \theta(n^k)$

$p \geq 0 \quad T(n) = \theta(n^k \log^p n)$

example:

$T(n) = 3T\left(\frac{n}{2}\right) + n^2$ $\qquad$ compare with base equation'

$T(n) = a\,T(n/b) + \theta(n^k \log^p n)$

$a = 3$, $b = 2$, $k = 2$, $p = 0$

$\therefore b^k = 2^2 = 4 \quad \therefore a < b^k$ $\quad$ so case 3 accepted. & $p \geq 0$

$\therefore T(n) = \theta(n^k \log^p n)$

$= \theta(n^2 \log^0 n) = \theta(n^2)$

Example : 2: $T(n) = 2T(n/2) + n\log n$. ①

base equation

$T(n) = aT(n/b) + \theta(n^k \log^p n)$

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

so, $b^k = 2^1 = 2$, $a = b^k$ case -2 accepted & $p > -1$

so, $T(n) = \theta(n^{\log_b a} \log^{p+1} n)$

$= \theta(n^{\log_2 2} \log^{1+1} n)$

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

example : 3

$T(n) = 8T(n/2) + cn$. , c = constant

so, $a = 8,\ b = 2\ k = 1,\ p = 0$

$b^k = 2^1 = 2$ so, $a > b^k$, case I accepted.

so, $T(n) = \theta(n^{\log_b a})$

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

$= \theta(n^{\log_2 2^3})$

$= \theta(n^{3 \log_2 2})$

$= \theta(n^3)$

Dividing function:-

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

$T(n) = 2$

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

Exercise:-

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

So, equation:-

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

Substitution:-

$T(n) = T(n/2)+1$ —— ①

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

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

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

$\vdots \quad \vdots \quad \vdots$

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

Assume $\cancel{2^k}$

$n/2^k = 1 \Rightarrow n = 2^k$

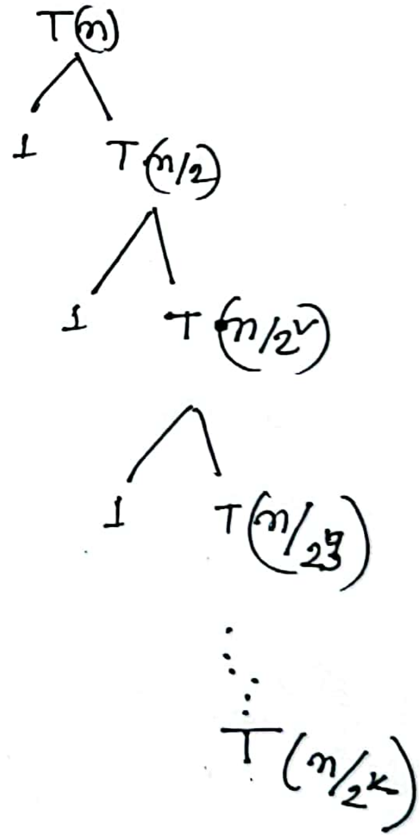
$\therefore k = \log_2 n$

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

$= 1 + \log n$

$\simeq O(\log n)$

Recurence Tree.



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

$\therefore n = 2^k$

$\therefore k = \log_2 n$

$\simeq O(\log(n))$

Practice problem on master theorem

(i) $T(n) = 2T(n/4) + n^{.50}$

(ii) $T(n) = 8T(n/4) + n^2 \log n$

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

(iv) $T(n) = 3T(n/3) + \sqrt{n}$