15/02 Mathematical analysis of recursive function

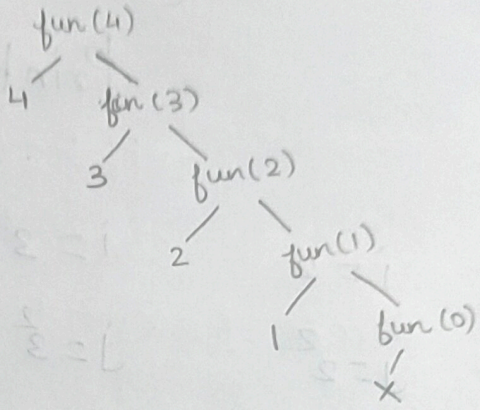
3 Methods to calculate complexity in recursive function

→ Recursive Tree → Back Substitution → Using Masters Theorem (To cross check the complexity)

Eg:1

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

This is a recursive function because the function (fun) calls its self



How to find Time complexity T(n)?

if n=4, no. of comparisons = 5, So for the given (n) the no of compis n+1

$O(n+1)$ [Ignoring the constant]

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

Recurrance relation (Total time complexity of the given function)

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

$T(n) = T(n-1) + 1 \rightarrow ①$

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

$T(n-1) = T(n-2) + 1 \rightarrow ②$

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

Sub ③ in ②

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

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

$= T(n-3) + 2 \rightarrow ④$

Sub ④ in ①

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

$T(n) = T(n-3) + 3 \rightarrow ⑤$

At some k ⇒ $T(n) = T(n-k) + k$

Let $n-k = 0$

$\boxed{n = k}$

$T(n) = T(n-n) + n = T(0) + n = 1 + n = n$

$T(n) = O(n)$

Eg:2

```
fun(n)
{
  if(n>1)
  printf("%d", n);
  fun(n-1);
}
```

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

$T(n) = T(n-1) + 1 \rightarrow ①$

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

$= T(n-2) + 1 \rightarrow ②$

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

$= T(n-3) + 1 \rightarrow ③$

Sub ③ in ②

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

$= T(n-3) + 2 \rightarrow ④$

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

$T(n) = T(n-3) + 3 \rightarrow ⑤$

At some k

$T(n) = T(n-k) + k \rightarrow ⑥$

Let $n - k = 1 \Rightarrow (n-1) = k$

Sub (k) in eq ⑥

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

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

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

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

Eg:3

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

$T(n) = n + T(n-1) \rightarrow ①$

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

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

Recurrance Relation

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

Sub ② in ①

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

Sub ③ in ④

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

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

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

let $n-(k+1) = 1$

$n-k-1 = 1$

$n-k = 1+1$

$n-k = 2$

$k = n-2$ //

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

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

$= n + (n-1) + (n-2) + \ldots 2 + 1$

$= n(n+1)/2$ => Formula for sum of series

$= n^2/2 + n/2$

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

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

Eg:4

```
void ex(int n) → T(n)
{
  if (n>1)
  {
    for(i=0; i<n; i++)
      printf("Crescent"); → n times
    ex(n-1); → T(n-1) times
  }
}
```

Recurrance Relation

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

$T(n) = n * T(n-1)$ → ①

$T(n-1) = (n-1) * T(n-2)$ → ②

$T(n-2) = (n-2) * T(n-3)$ → ③

$T(n) = n * (n-1) * T(n-2)$ → ④

Sub ③ in ④

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

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

At some k

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

Let $n-(k+1) = 1 \Rightarrow n-k-1 = 1 \Rightarrow \boxed{k = n-2}$

$T(n) = n * (n-1) * (n-2) * \ldots * (n-(n-2)) * T(n-((n-2)+1))$

$= n * (n-1) * (n-2) * \ldots * (n-n+2) * T(n-n+2-1)$

$= n * (n-1) * (n-2) * \ldots * 2 * 1$

$T(n) = n^2$

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

Eg:5

```
void ex (int n) → T(n)
{
  if (n>1)
  {
    print ("%d", n);
    ex (n-1); → T(n-1)
    ex (n-1); → T(n-1)
  }
}
```

Recurrance Relation

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

$T(n) = 1 + 2T(n-1) \rightarrow ①$

$T(n-1) = 1 + 2T(n-2) \rightarrow ②$

$T(n-2) = 1 + 2T(n-3) \rightarrow ③$

Sub ② in ①

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

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

$= 1 + 2 + 4[1 + 2T(n-3)]$

$= 1 + 2 + 4 + 8 * T(n-3)$

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

$= 2^0 + 2^1 + 2^2 + 2^3 + 2^4 * T(n-4)$

$\vdots$

$= 2^0 + 2^1 + 2^2 + \ldots + 2^k T(n-k)$

Let $n-k = 1$

$\boxed{k = n-1}$

$T(n) = 2^0 + 2^1 + 2^2 + \ldots 2^{n-1} T(n-(n-1))$

$= 2^0 + 2^1 + 2^2 + \ldots 2^{n-1} T(n-n+1)$

$= 2^0 + 2^1 + 2^2 + \ldots 2^{n-1} T(1)$

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

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

Eg:6

```
void ex(int n) → T(n)
{
  if(n>1)
  {
    for(i=1;i<=n;i++)
      print("%d",n); → 1 time
    ex(n/2); → T(n/2)
    ex(n/2); → T(n/2)
  }
}
```

Recurrance Relation

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

$T(n) = n + 2 * T(n/2) \rightarrow (1)$

$T(n/2) = n/2 + 2 * (T(n/4)) \rightarrow (2)$

$T(n/4) = n/4 + 2T\left(\frac{n}{8}\right) \rightarrow (3)$

Sub (2) in (1)

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

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

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

Sub (3) in (4)

$T(n) = 2n + 4\left(\frac{n}{4} + 2T\left(\frac{n}{8}\right)\right)$

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

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

At some k, $T(n) = kn + 2^k T\left(\frac{n}{2k}\right)$, Let $\frac{n}{2^k} = 1$, $n = 2k$

$\log n = \log 2k$

$\log_2 n = k$

$T(n) = n \log_2 n + n T(1) \Rightarrow n \log_2 n + n$

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

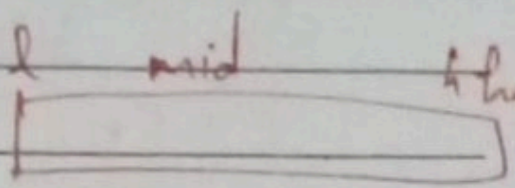
Time Recursive algorithm

SRSMES

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

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

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



Binary search algo

```
binarysearch(l, h, key)
if (l==h)  ------------> 1
{ if(a[i]==key)
    return l;
  else
    return 0;
}
else
{
  mid = (l+h)/2;          ----> 1
  if(key == a[mid])       ----> 1
      return mid;
  if (key < a[mid])
    return binarysearch(l, mid-1, key);
  else
    return binarysearch(mid+1, h, key);
}
```

$T(n/2)$

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