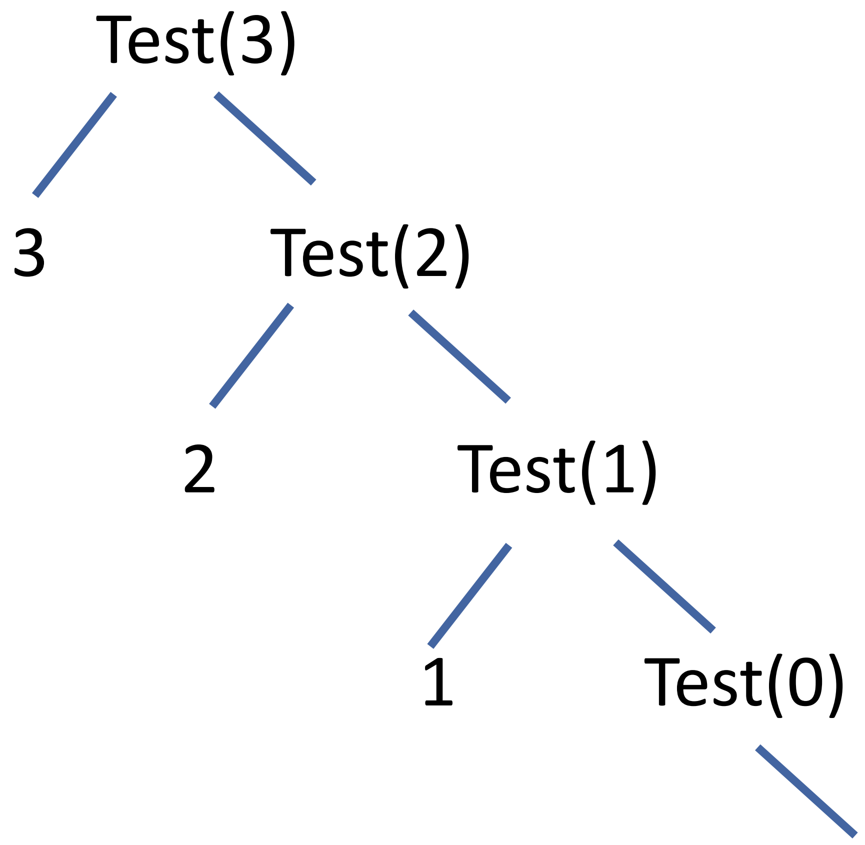
Recurrence Relation Examples: Substitution Method

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

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



f(n) = n+1 calls

Complexity O(n)

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

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

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

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 for k times

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

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

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 for k times

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

Assume n-k= 0

n=k

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

T(n) = T(0) + n

T(n) = 1 + n

O(n)

Recurrence Relation

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

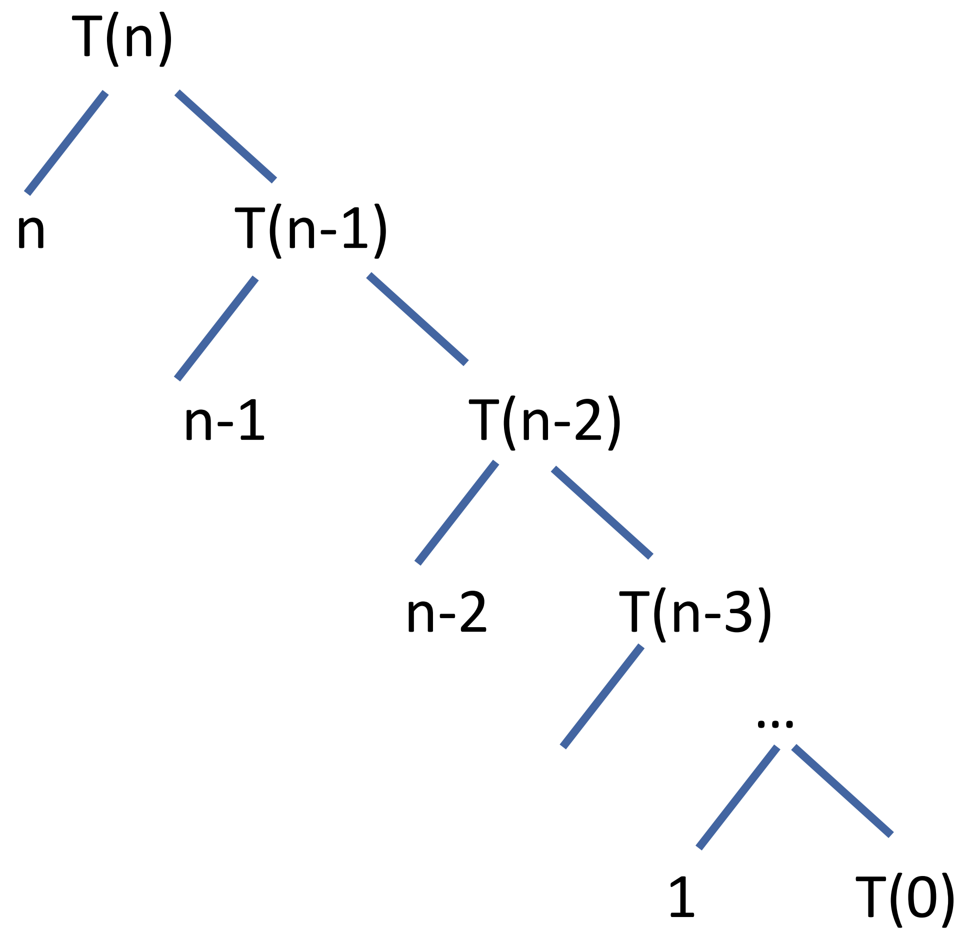
Recurrence Relation

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

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

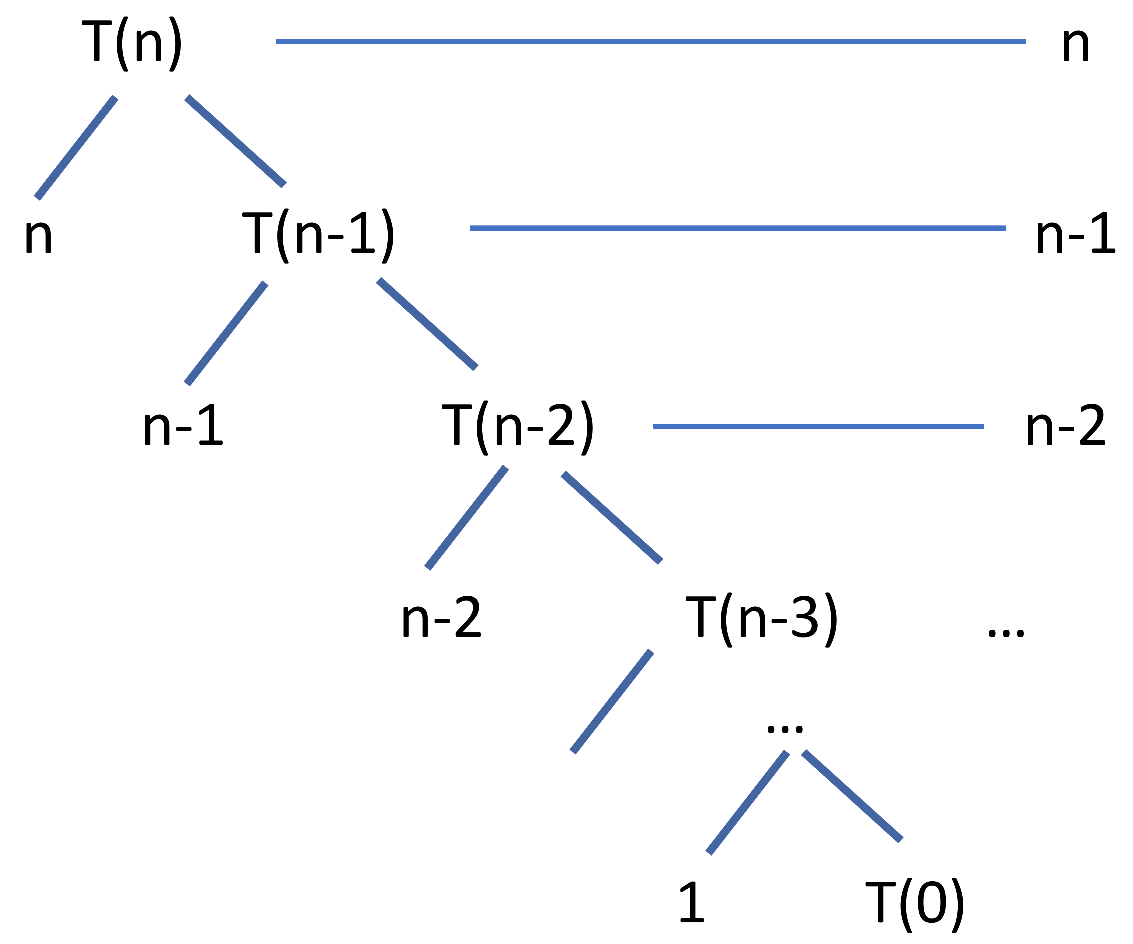
Recurrence Relation

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



Recurrence Relation

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

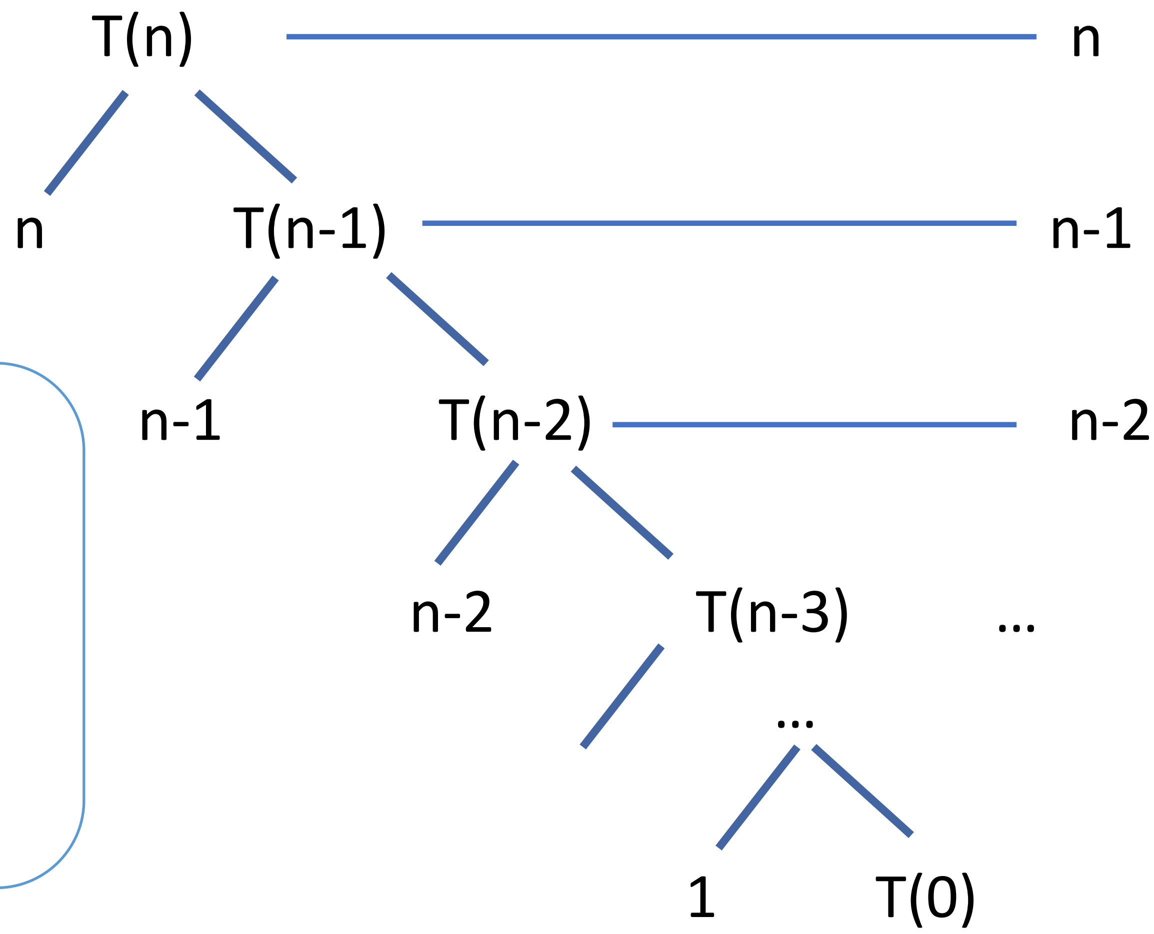


Recurrence Relation

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

0+1+2+.........+n = n(n+1)/2

= $\boldsymbol{O(n^2)}$



Recurrence Relation

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

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

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

T(n) = T(n-k) + (n-(k-1)) + (n-(k-2)) + ….+(n-1)+n

Assume (n-k)=0

n=k

T(n) = T(0) + 1 + 2 + 3 + …..+(n-1)+n

= 1 + n(n+1)/2

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

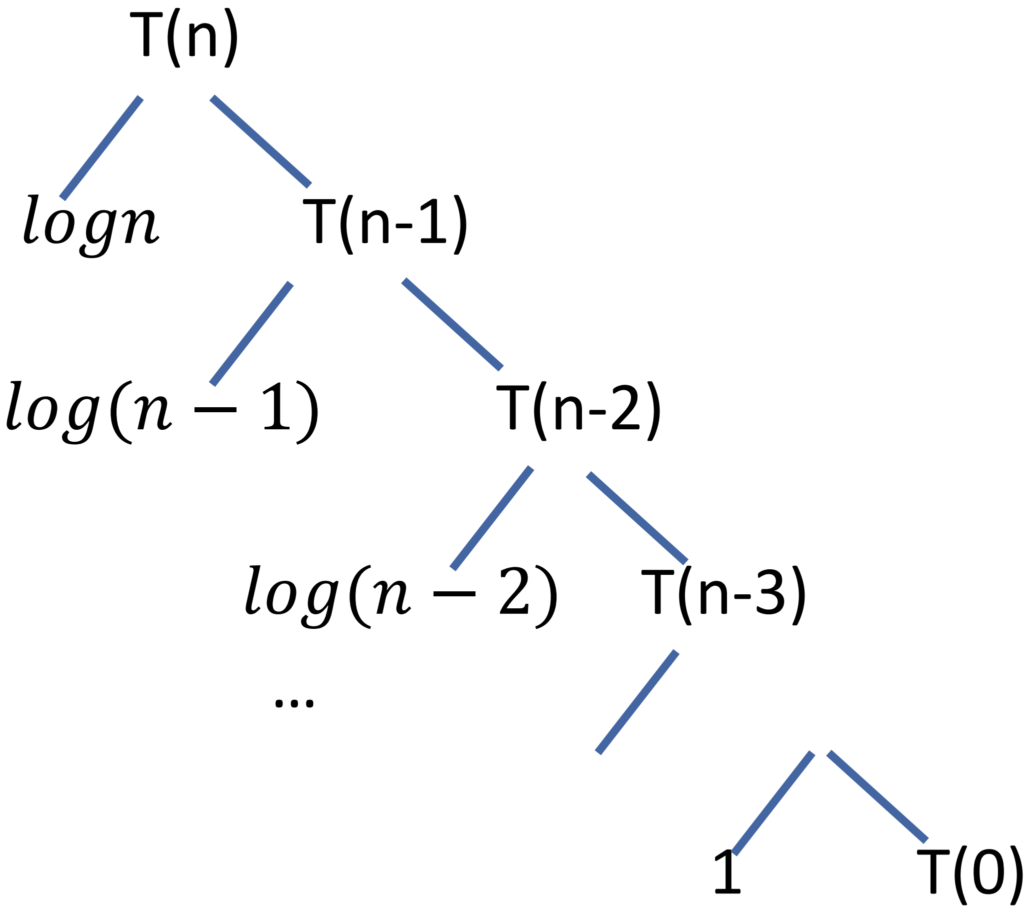
Recurrence Relation

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

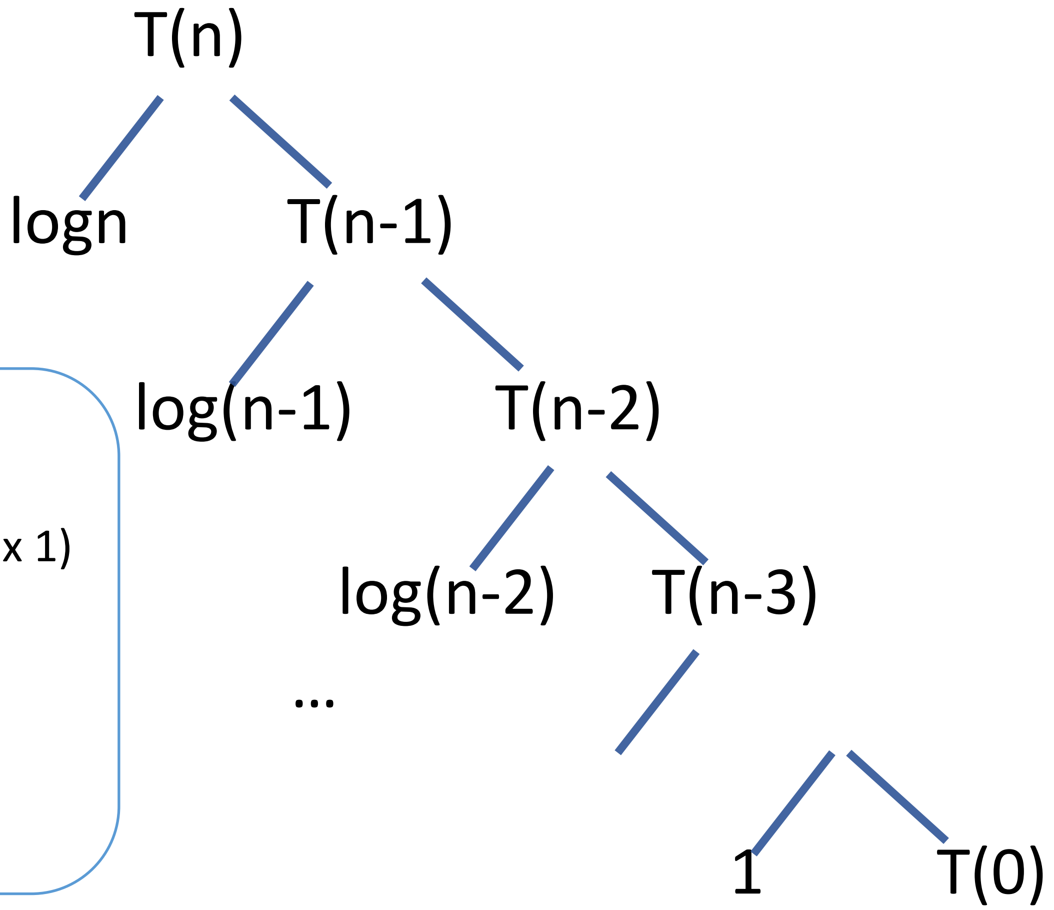
```
void Test(int n)
{
    if(n > 0)                                   1
     {
       for(i=0; i < n ; i*2)                    n + 1
       {
           printf("%d", n);                     logn
       }
        Test(n-1)                               T(n - 1)
    }
}
```

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

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



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



log1+log2+..........log(n-1)+logn =log(n x n-1 x ...x 2 x 1)
=logn!

$O(nlogn)$

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

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

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

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

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

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

$$T(n) = T(n-k) + \log 1 + \log 2 + \ldots. + \log(n-1) + logn$$

Putting $n - k = 0$

$T(n) = T(0) + logn!$
$T(n) = 1 + logn!$
$O(nlogn)$

$T(n) = T(n-1) + 1$ —— $O(n)$

$T(n) = T(n-1) + n$ —— $O(n^2)$

$T(n) = T(n-1) + logn$ —— $O(nlogn)$

$T(n) = T(n-1) + n$^2 —— $O(n^3)$

$T(n) = T(n-2) + 1$ —— $O(n)$

$T(n) = T(n-100) + n$ —— $O(n^2)$

For recurrence like

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

??

Recurrence Relation

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

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

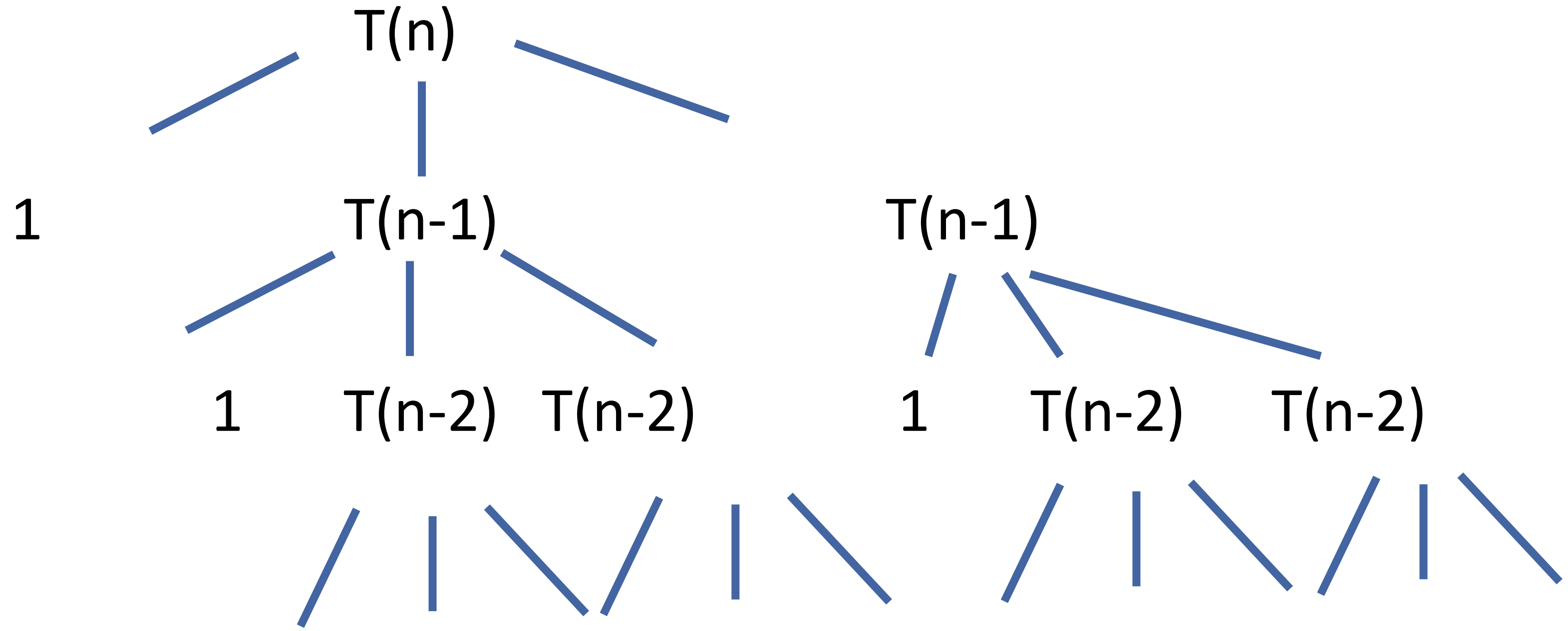
Recurrence Relation

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

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

Recurrence Relation

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



Recurrence Relation

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

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

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

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

Assume n-k=0

$O(2^n)$

Recurrence Relation

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

By substitution,

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

Assume $n - k = 0$

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

O(2^n)

Recurrence Relation for Root Function

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

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

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

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

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

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

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

$\vdots$

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

Assume n = 2^m

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

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

Therefore, $\frac{m}{2^k}$ = 1

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

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

k = log $\log_2 n$

θ(log $\log_2 n$)