Analysing Complexity for Recursive Functions

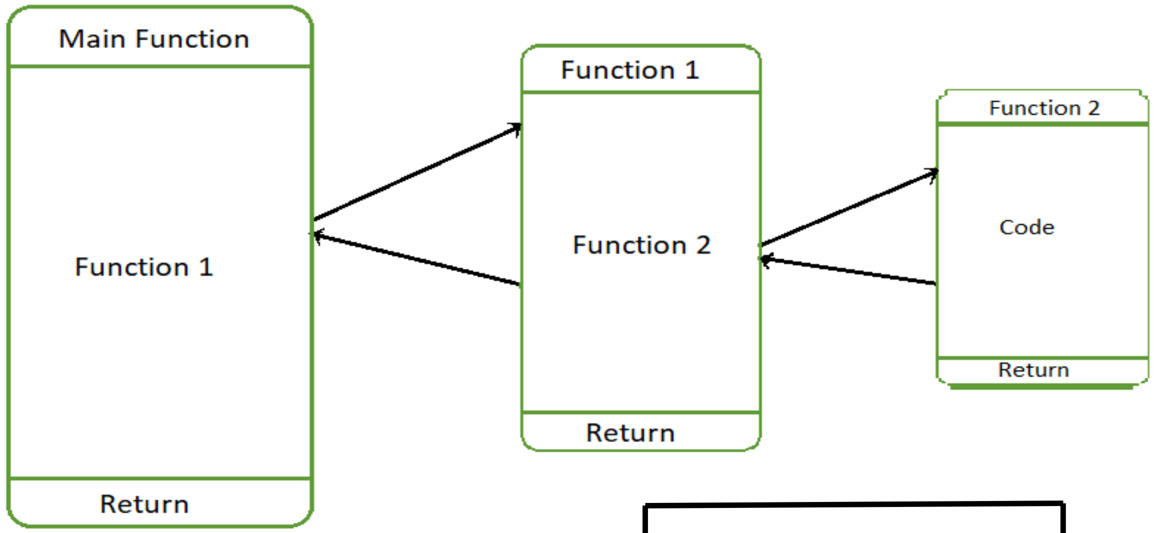
By

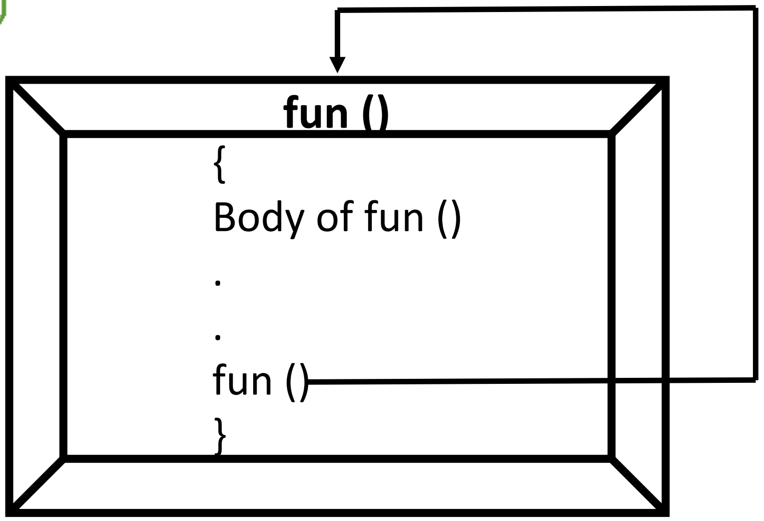
Arun Cyril Jose

Code Snippet

Snippet 9

```
void fun (int n)
{
    if ( n <= 1 )
        return
    for (int i = 0; i < n; i++)
        print("IIIT Kottayam")
    fun (n/2)
    fun (n/2)
}
```





Code Snippet

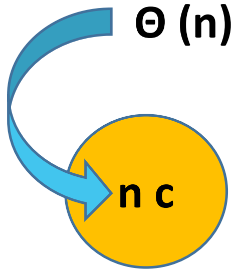
Snippet 10

```
void fun (int n)
{
    if ( n <= 1 )
        return
    print(“Hai")
    fun (n - 1)
}
```

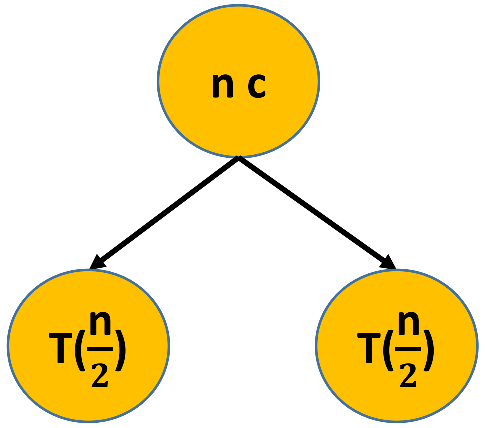
Recursion Tree

Snippet 9

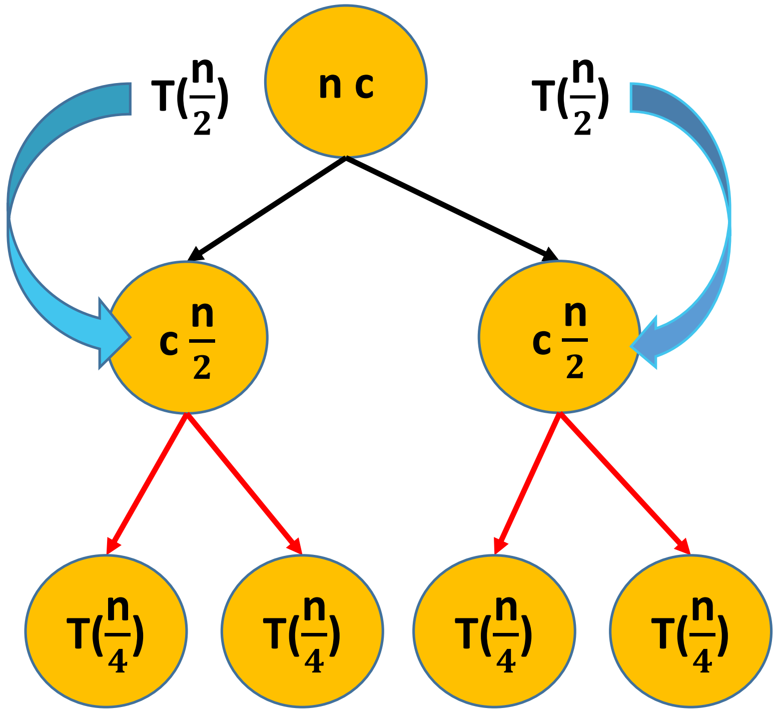
```
void fun (int n)
{
    if ( n <= 1 )
        return
    for (int i = 0; i < n; i++)
        print("IIIT Kottayam")
    fun (n/2)
    fun (n/2)
}
```



Step 1

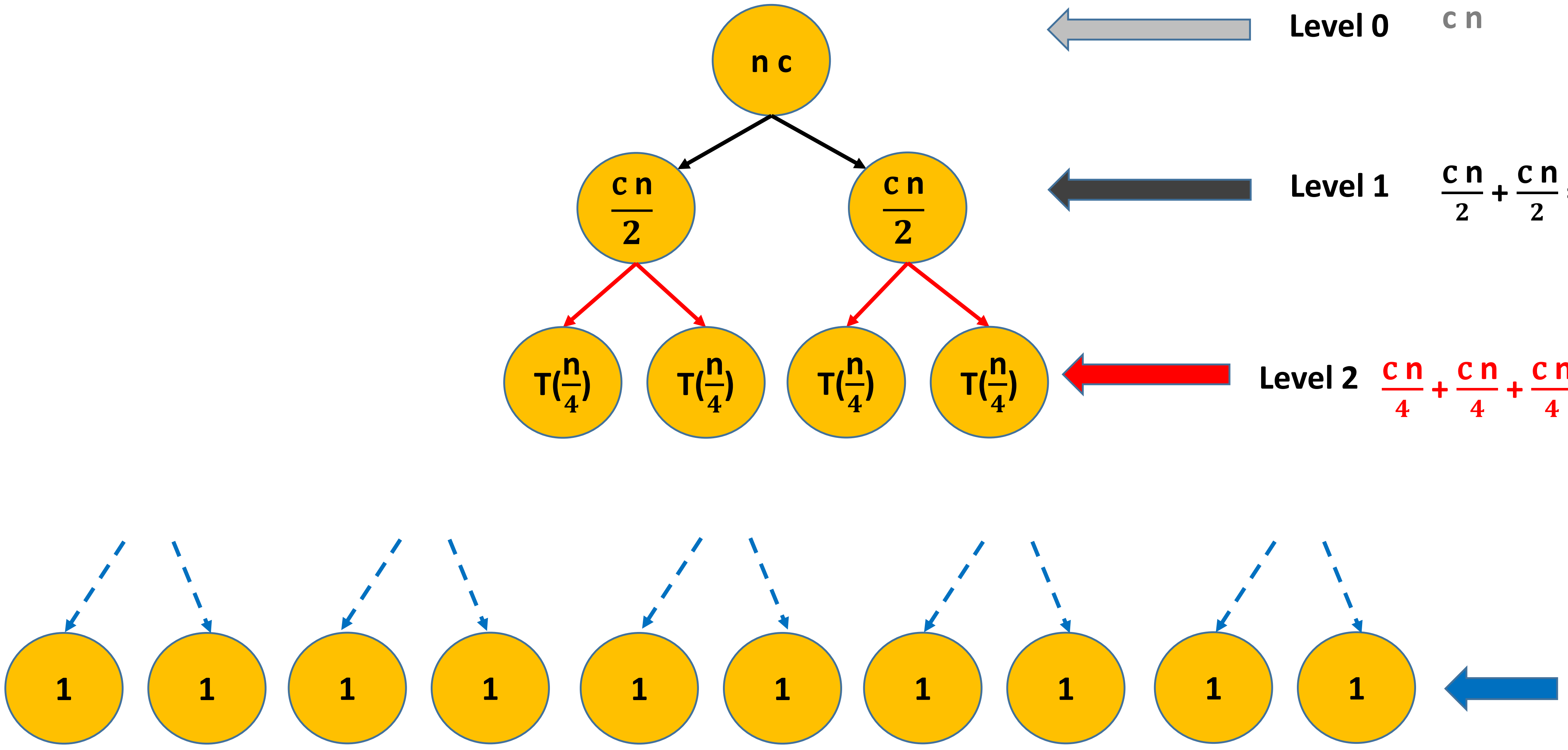


Step 2



Step 3

Recursion Tree



Level 0 c n

Level 1 $\frac{c\,n}{2} + \frac{c\,n}{2} = c\,n$

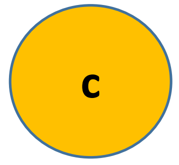
Level 2 $\frac{c\,n}{4} + \frac{c\,n}{4} + \frac{c\,n}{4} + \frac{c\,n}{4} = c\,n$

Level k

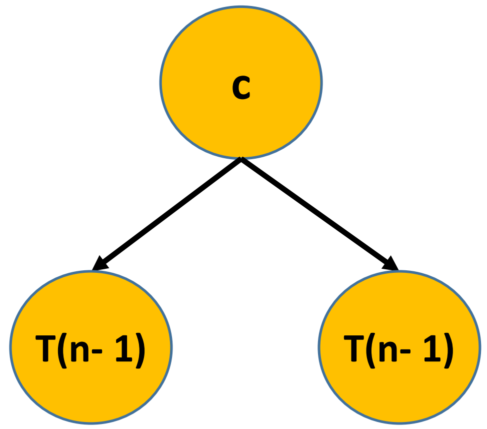
Recursion Tree

Snippet 11

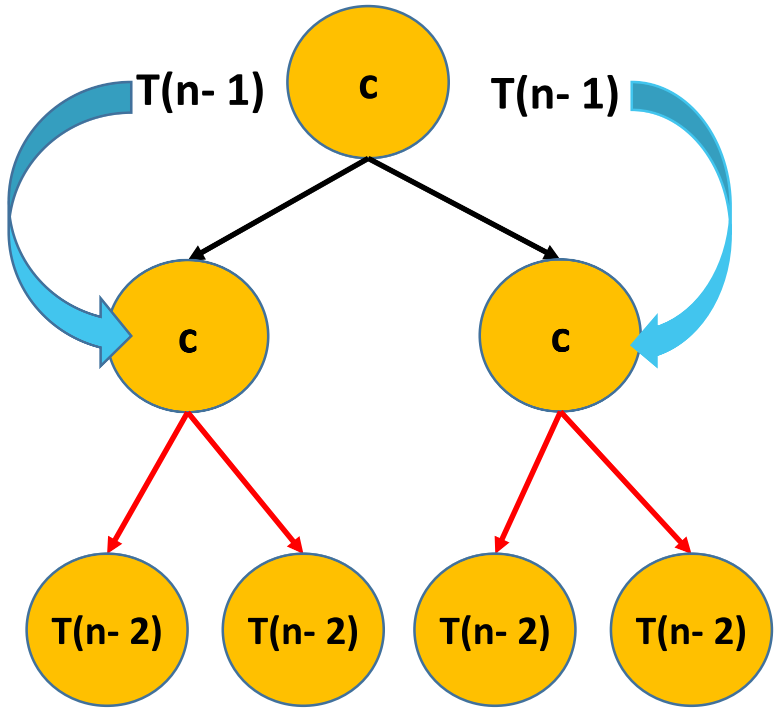
```
void fun (int n)
{
    if ( n <= 1 )
        return
    print(“Hai")
    fun (n - 1)
    fun (n - 1)
}
```



Step 1



Step 2

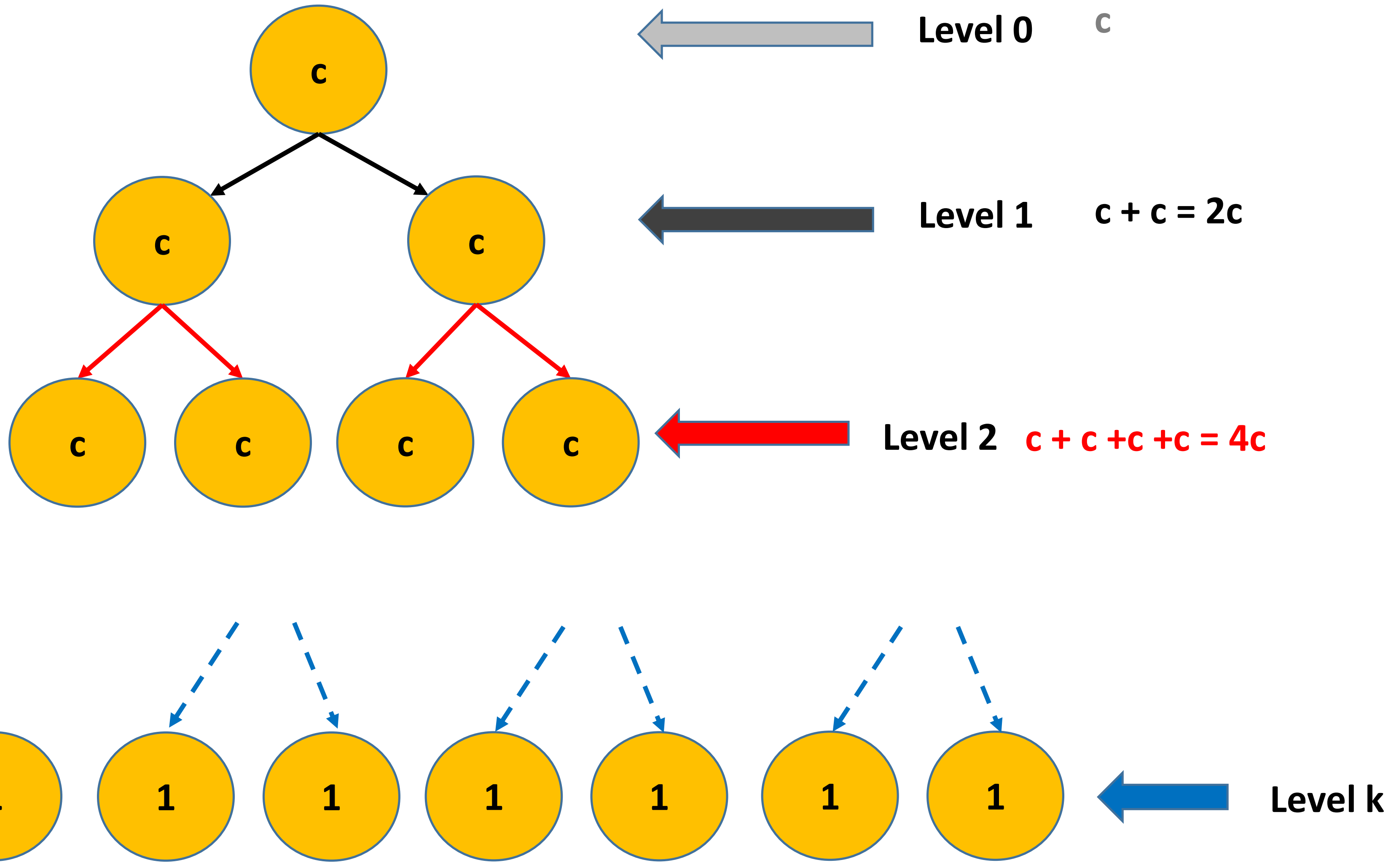


Step 3

Recursion Tree

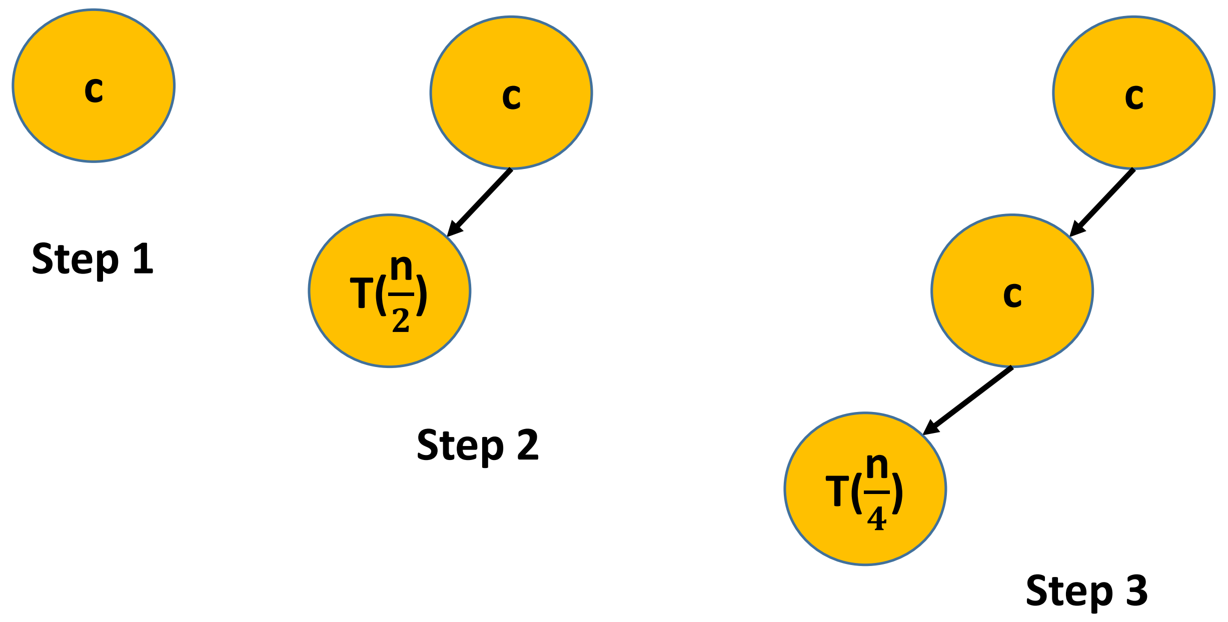
Total Complexity =

$$\underbrace{c + 2c + 4c + 8c + \ldots\ldots\ldots}_{\text{'n' times}}$$



Recursion Tree

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



Total Complexity =

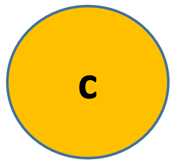
$$\underbrace{c + c + c + c + \ldots\ldots\ldots}_{\text{'log n' times}}$$

Recursion Tree

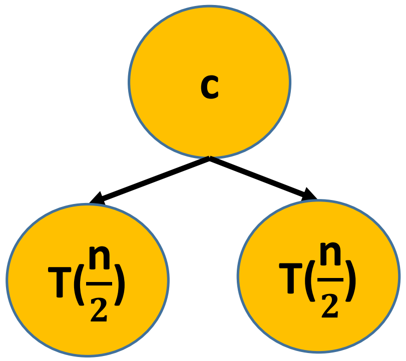
- $T(n) = 2T(\frac{n}{2}) + c$

Total Complexity =

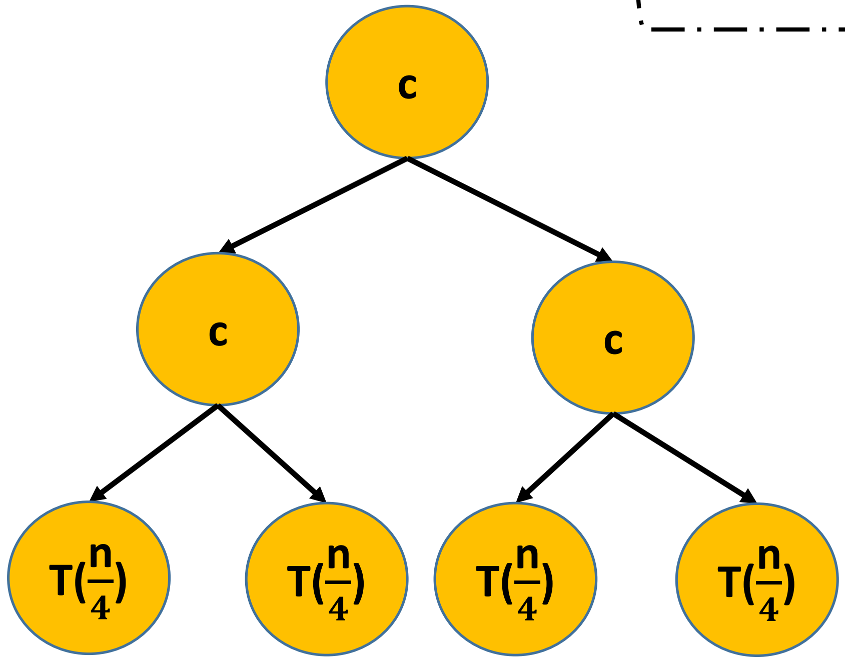
$$\underbrace{c + 2c + 4c + 8c + \ldots\ldots\ldots}_{\text{'}\log_2 n\text{' times}}$$



Step 1



Step 2



Step 3