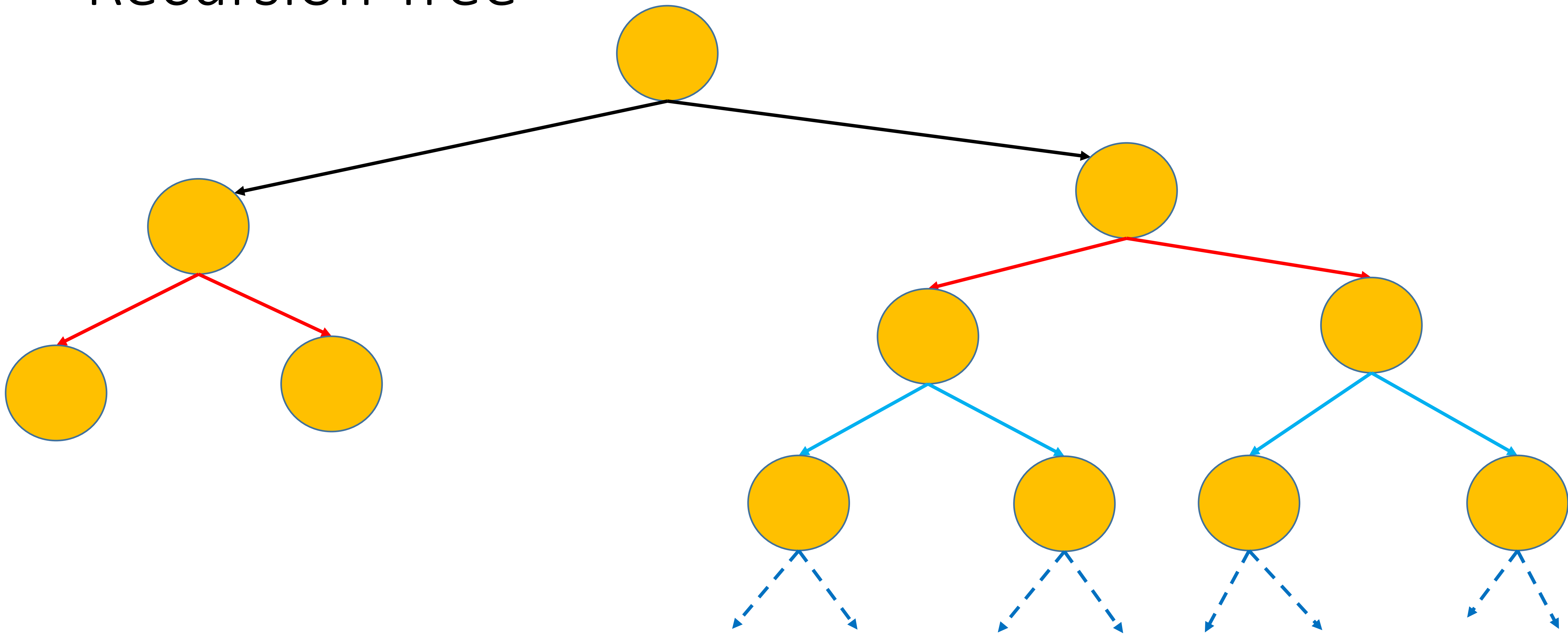
Analysing Complexity for Recursive Functions (cont.)

By

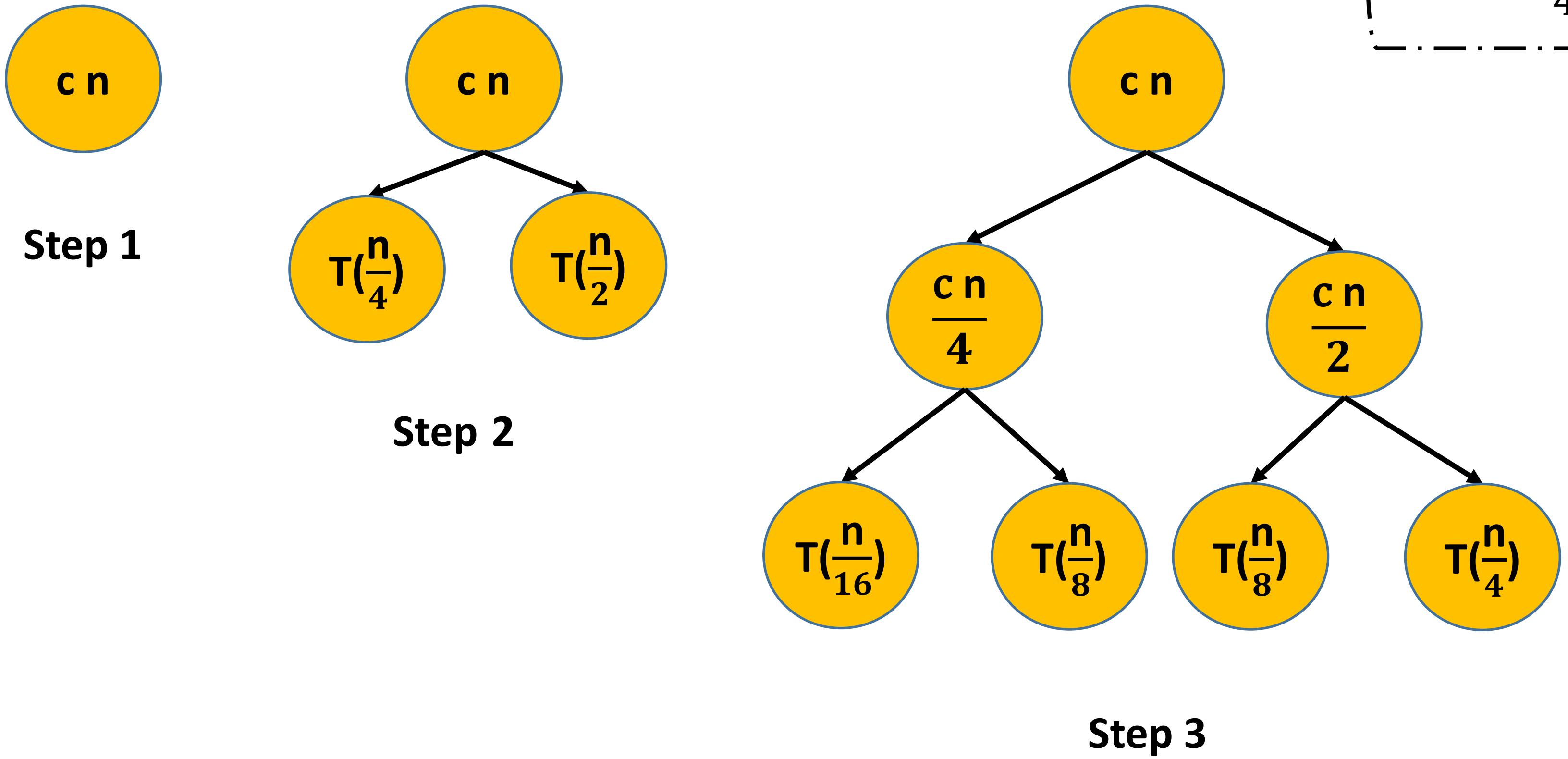
Arun Cyril Jose

Recursion Tree



Recursion Tree

- $T(n) = T(\frac{n}{4}) + T(\frac{n}{2}) + cn$



$$\frac{cn}{4} + \frac{cn}{2} = \frac{3cn}{4}$$

$$\frac{cn}{16} + \frac{cn}{8} + \frac{cn}{8} + \frac{cn}{4} = \frac{9cn}{16}$$

Total Complexity =

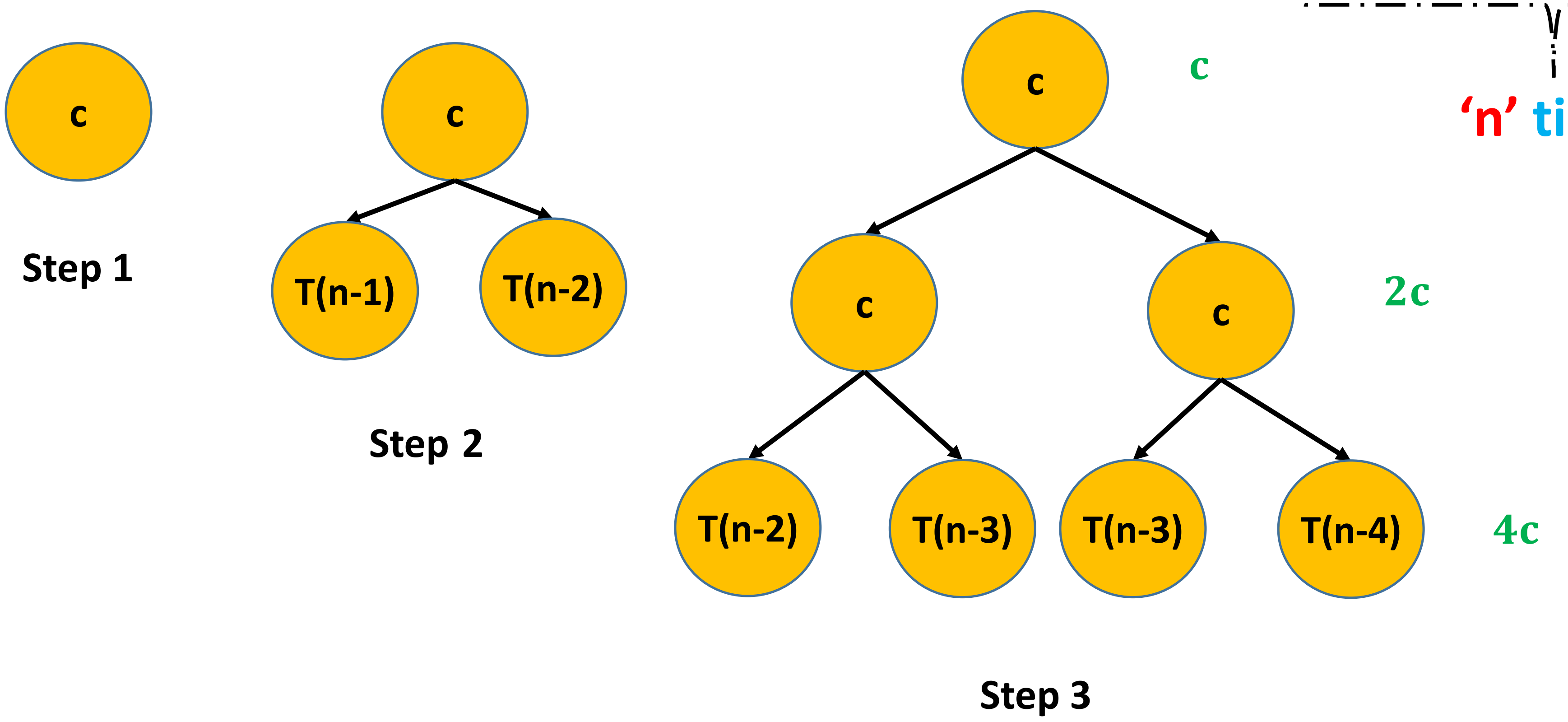
$$\underbrace{cn + \frac{3cn}{4} + \frac{9cn}{16} + \frac{27cn}{64} + \ldots\ldots}_{\text{'log n' times}}$$

Recursion Tree

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

Total Complexity =

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



Space Complexity and Auxiliary Space

By

Arun Cyril Jose

Space Complexity

- Order of growth of memory space in terms on input size.

Snippet 1

```
int fun1(int n)
{
    return (n * (n+1) / 2)
 }
```

Snippet 2

```
int fun2 (int n)
{
    int sum = 0;
    for(i = 1; i <=n; i++)
       sum = sum + i;
    return sum;
}
```

Space Complexity

- Order of growth of memory space in terms on input size.

Snippet 3

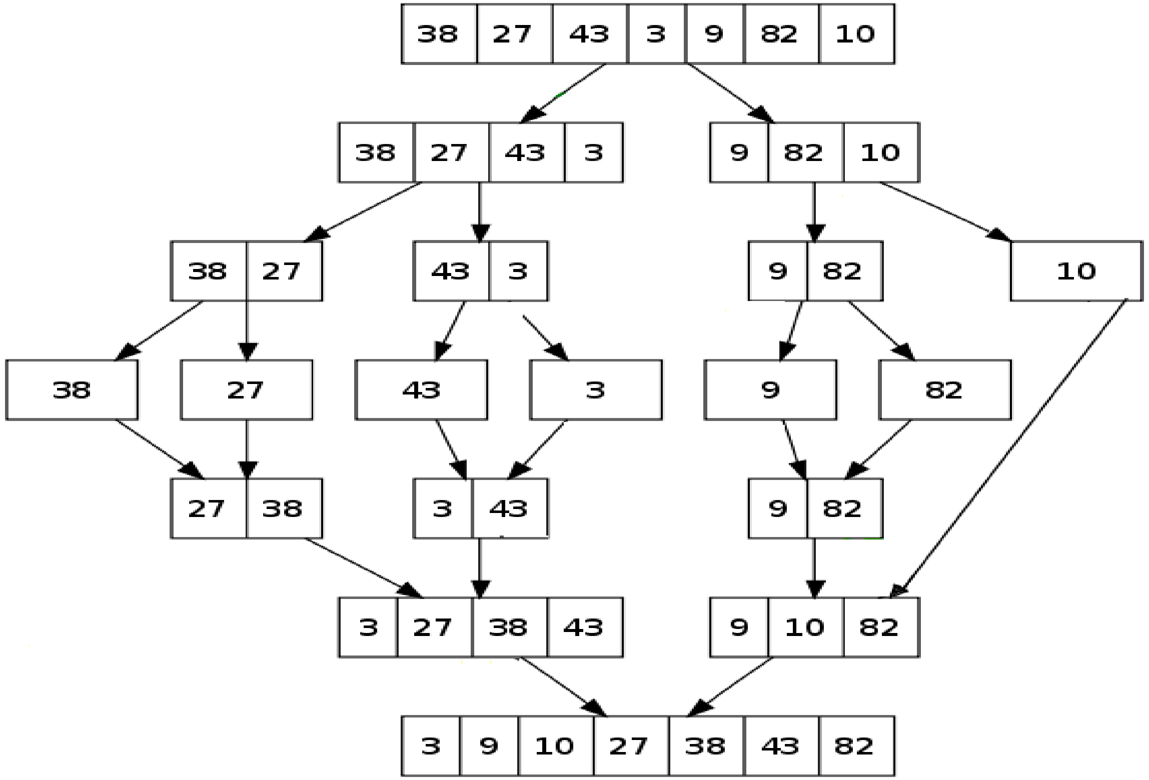
```
int fun3 (int arr[], int n)
{
    int sum = 0;
    for(i = 1; i <=n; i++)
       sum = sum + arr[i];
    return sum;
}
```

Space Complexity

- Order of growth of memory space in terms on input size.

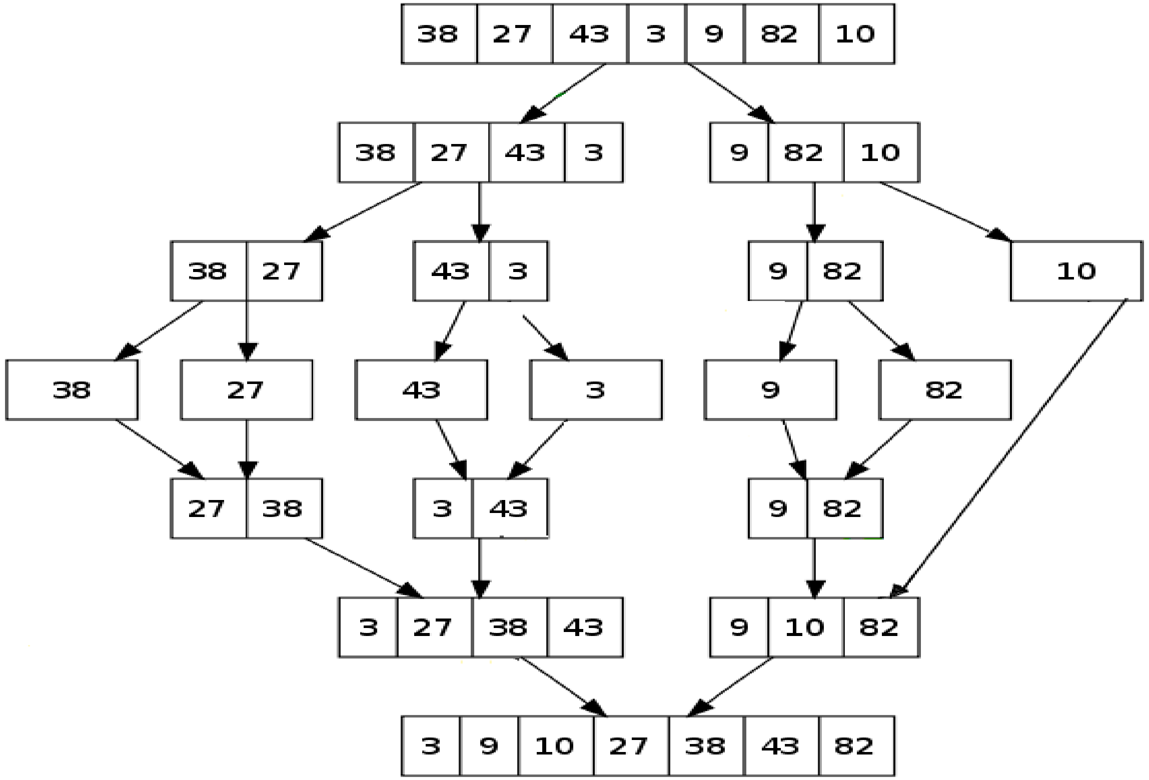
Snippet 3

```
int fun3 (int arr[], int n)
{
    int sum = 0;
    for(i = 1; i <=n; i++)
        sum = sum + arr[i];
    return sum;
}
```

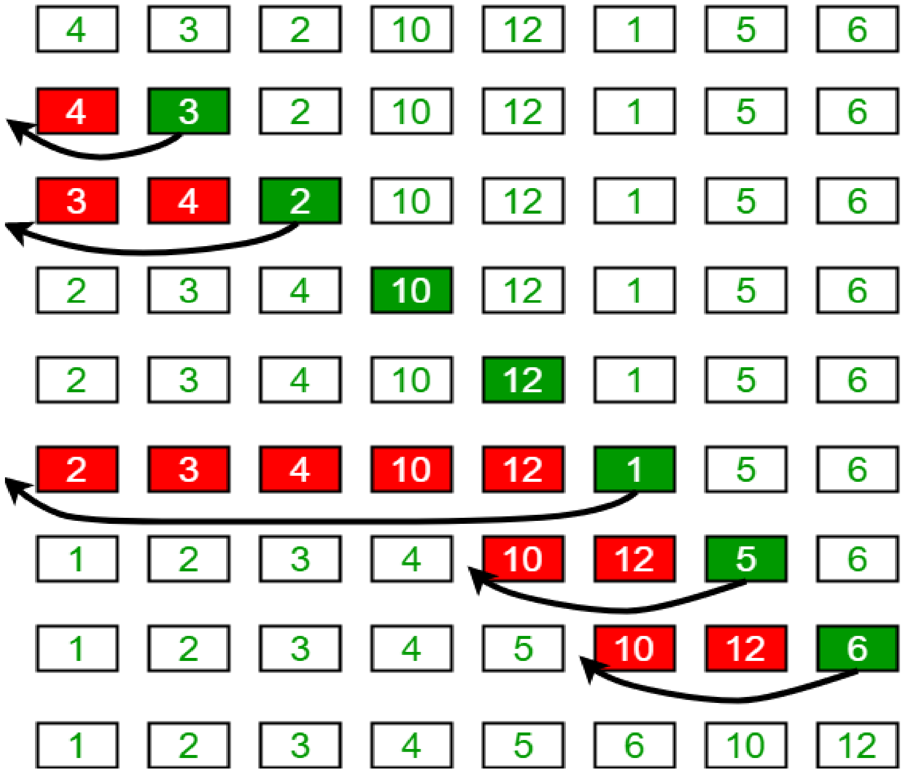


Auxiliary Complexity

- Order of growth of **temporary** space in terms on input size.



Insertion Sort Execution Example


4 3 2 10 12 1 5 6

4 3 2 10 12 1 5 6

3 4 2 10 12 1 5 6

2 3 4 10 12 1 5 6

2 3 4 10 12 1 5 6

2 3 4 10 12 1 5 6

1 2 3 4 10 12 5 6

1 2 3 4 5 10 12 6

1 2 3 4 5 6 10 12

Auxiliary Complexity

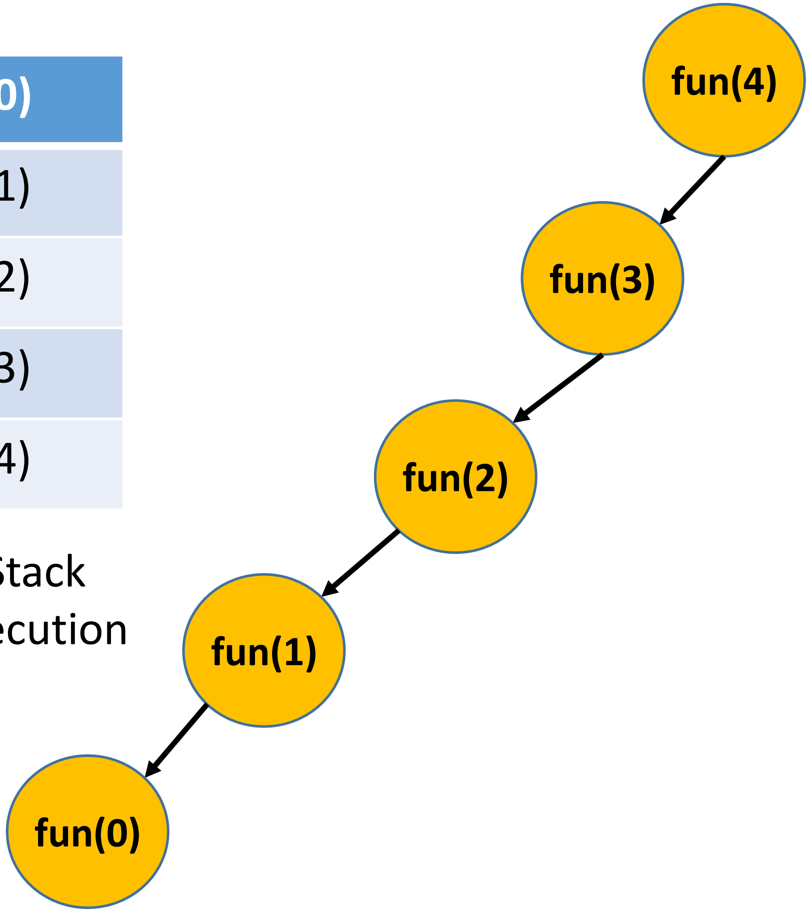
- Order of growth of **temporary** space in terms on input size.

<u>Snippet 4</u>

```
int fun (int n)
{
    if (n <= 0)
        return 0;
    return (n + fun (n-1));
}
```

fun (0)
fun (1)
fun (2)
fun (3)
fun (4)

Function Stack
during execution



Auxiliary Complexity

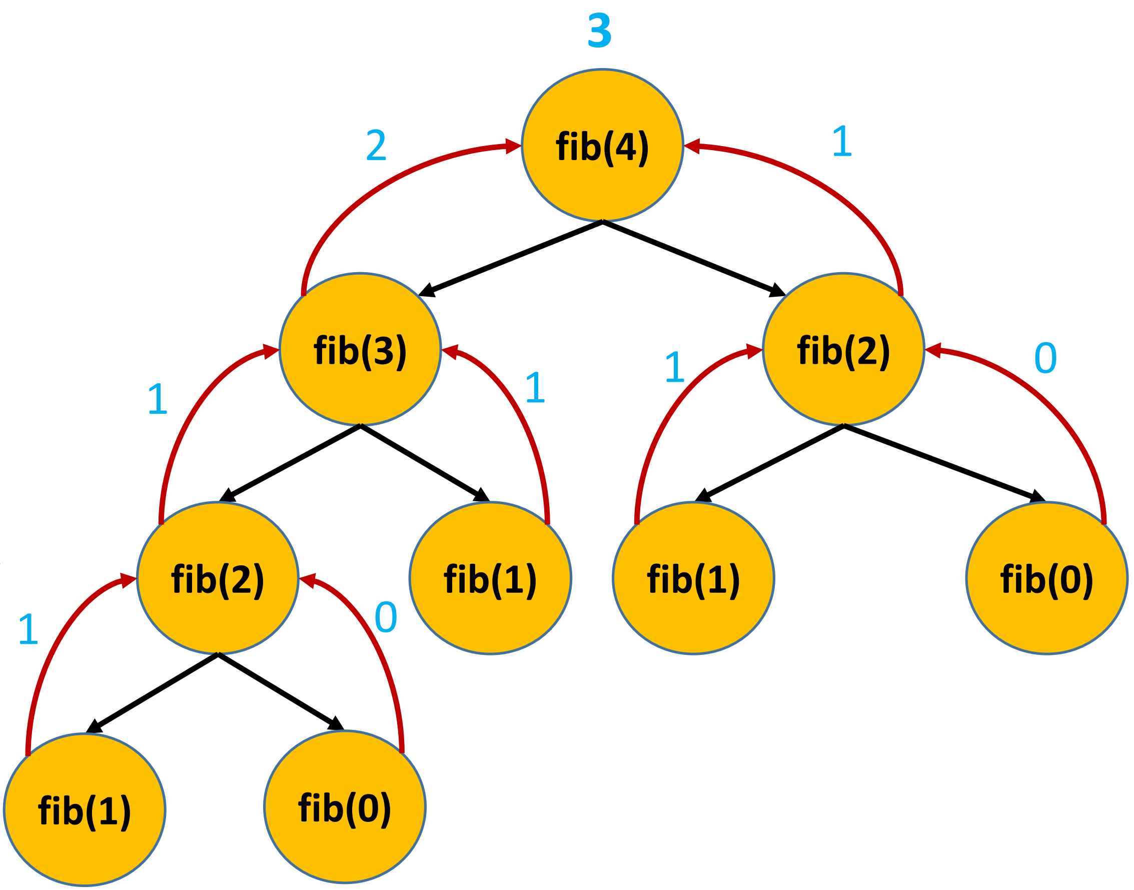
- Fibonacci Series

<u>Snippet 5</u>

```
int fib (int n)
{
    if (n == 0 || n == 1)
        return n;
    return (fib (n – 1) + fib (n – 2));
}
```

fib (0)
fib (1)
fib (2)
fib (3)
fib (4)

Function call stack at some point during execution



3
fib(4)
2
1
fib(3)
fib(2)
1
1
1
0
fib(2)
fib(1)
fib(1)
fib(0)
1
0
fib(1)
fib(0)