9/10/2023

TIME & SPACE COMPLEXITY OF RECURSIVE SOLUTIONs

✅ What is time complexity:

Time is taken by any algorithm with respect to a function of its input N.

K = Process

M = byte space Allocated

Example:01

```
main() {

    fun(n);

    return;

}
```

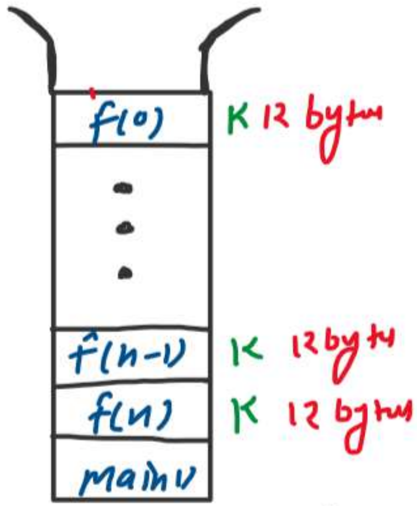
```
fun(n) {

    BASE
    if (n==0)
        return;

    Processing
        int a,b,c;

    Relation
        fun(n-1);

}
```



Call Stack (RAM)

Example 2 Print Array

```
PrintArray( int a[], int N){
        if(n==0) return; } K1 Process
        cout << *a << " ";
        PrintArray(a+1, N-1)
}
```

$\Rightarrow nK + K_1$

$\Rightarrow O(nK + K_1)$

$\Rightarrow \boxed{O(n)}$

① RECURSIVE TREE Time Complexity

$f(n) \rightarrow K$

$f(n-1) \rightarrow K$

$f(n-2) \rightarrow K$

$\vdots$

$f(0) \rightarrow K_1$

② FORMULA method

Time complexity

$$F(N) = K + F(\cancel{N}-1)$$

$$T(\cancel{N}) = K + T(\cancel{N}-1)$$

$$T(\cancel{N}-1) = K + T(\cancel{N}-2)$$

$$T(\cancel{N}-2) = K + T(\cancel{N}-3)$$

$$\vdots \qquad\qquad \vdots$$

$$T(\cancel{1}) = K + T(\cancel{0})$$

$$T(\cancel{0}) = K_1$$

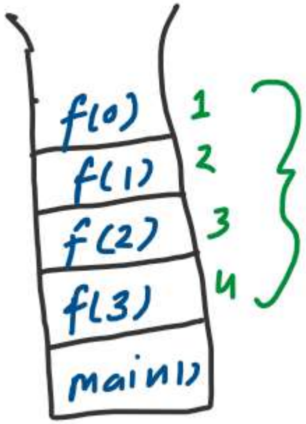
$$T(N) = nK + K_1$$

$$T(N) = O(nK + K_1)$$

$$= \boxed{O(N)}$$

SPACE COMPLEXITY

when $N = 3$ $S.C. = ?$



Total Entry $= 4$

$= 3 + 1$

$= N + 1$

$S.C. = O(N+1)$

$= \boxed{O(N)}$

Example 3 Factorial

```
int fact(int N) {
    if(N==1)
        return 1;
    return N*fact(N-1);
}
```

Time complexity M:1

$f(N) \rightarrow K$ process

$f(N-1) \rightarrow K$

$f(N-2) \rightarrow K$

$\vdots$

$f(1) \rightarrow K$

$\Rightarrow T(N) = NK$

$= O(N)$

Time complexity **M:2**

$$\boxed{\begin{aligned} F(N) &= N * F(N-1) \\ T(N) &= K_{Time} * T(N-1) \end{aligned}}$$

$$T(N) = K + T(\cancel{N-1})$$
$$T(\cancel{N-1}) = K + T(\cancel{N-2})$$
$$T(\cancel{N-2}) = K + T(\cancel{N-3})$$
$$\vdots \qquad\qquad \vdots$$
$$T(\cancel{1}) = K + T(0)$$
$$T(0) = K_1$$

YE PART NAHI BAN SKtaHai because **N=1 return 1** Ho Rha Hai

$$T(N) = NK$$
$$= \boxed{O(N)}$$

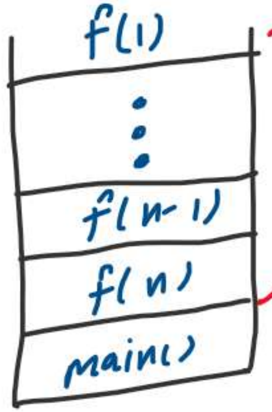
Space complexity

Ex $N! = 5!$

$$5! = \underbrace{5 \times 4 \times 3 \times 2 \times 1}_{5 \text{ ENTRY}}$$

S.C. $\boxed{O(N)}$

M = space Allocated to each Entry



Total Entry = M

N * M

S.C $\Rightarrow O(NM)$

$\Rightarrow O(N)$

Example 4 Binary search

```
// ✅Binary Search RE
int BS(int arr[], int k, int start, int end){
    // Base Case
    if(start > end){
        return -1;
    }

    int mid = start + (end - start)/2;
    if(arr[mid] == k){
        return mid;
    }
    else if(arr[mid] < k){
        return BS(arr, k, mid + 1, end);
    }
    else{
        return BS(arr, k, start, mid - 1);
    }
}
```

$$F(N) = K_{Time} + F(N/2)$$

$$\Rightarrow \quad T(N) = K + F(N/2)$$

$$T(N/2) = K + F(N/4)$$

$$T(N/4) = K + F(N/8)$$

$$\vdots \qquad \qquad \vdots$$

$$T(2) = K + F(1)$$

$$T(1) = K + \boxed{F(0)} \rightarrow \text{YE NAHI BANEGA}$$

$$T(N) = a * K$$

what is a ?

$f(n) \Rightarrow 2^0$ call

$f(n/2) \Rightarrow n/2^1$ call

$f(\frac{n}{4}) \Rightarrow n/2^2$ call

$f(\frac{n}{8}) \Rightarrow n/2^3$ call

$\vdots$

$f(2) \Rightarrow n/2^1$ call

$f(1) \Rightarrow n/2^a$ call

$2^a \Rightarrow \frac{n}{2^a} = 1$

$a = \log N$

$TC = a * K$

$= \log N * K$

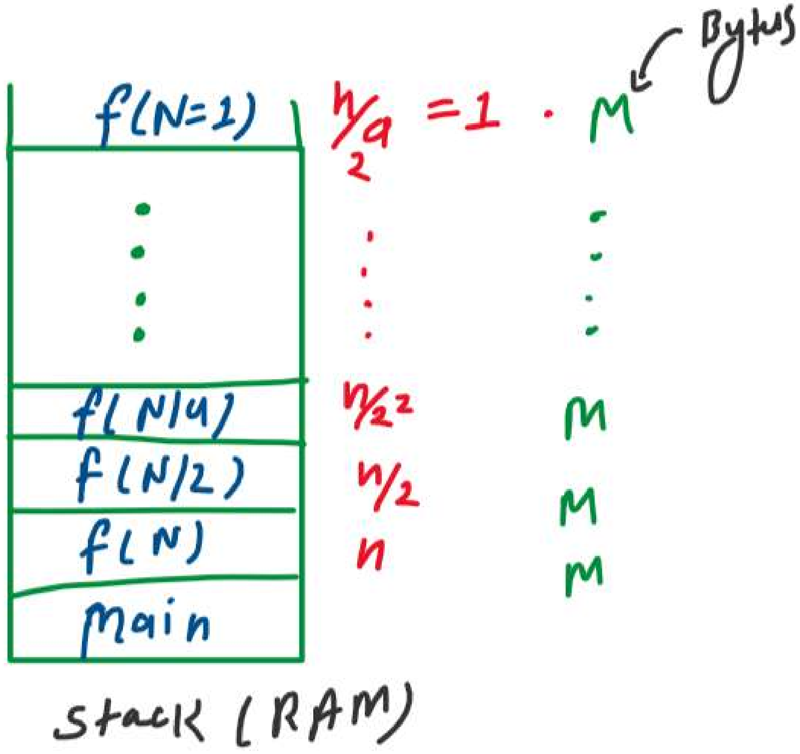
$= \boxed{O(\log N)}$

Space complexity

SoCo $\Rightarrow O(M * a)$

$\Rightarrow O(M \log N)$

$\Rightarrow \boxed{O(\log N)}$



Programs

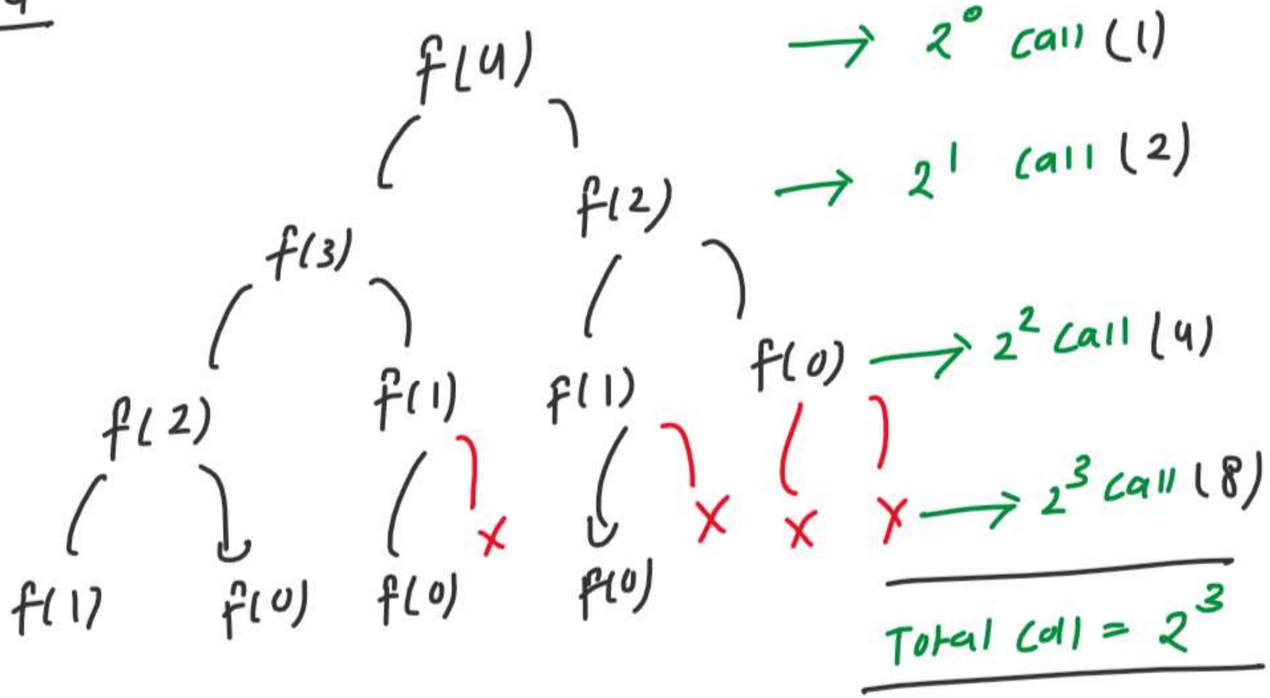
FEBONACCI Series

0 1 1 2 3 5 8 13 21 . . . . . .

```
// ✅Febonacci series RE
int fib(int N){
    //Base Case
    if(N==0 || N==1){
        return N;
    }

    return fib(N-1) + fib(N-2);
}
```

let N=4

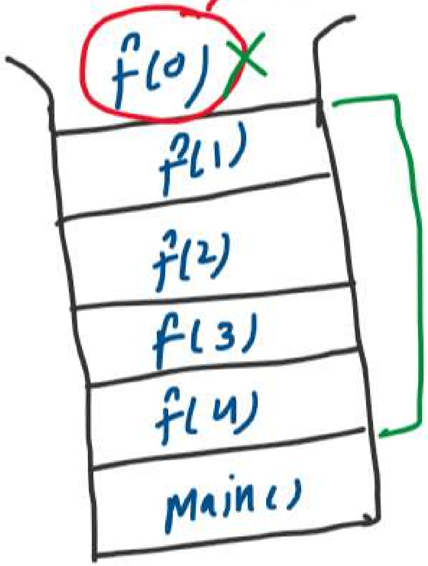


T.C $\Rightarrow O(2^{n-1})$

$\Rightarrow O(2^n)$

Space complexity

$\boxed{N=4}$



f(0) को f(1) आने नहीं देगा

Total Entry $= 4 * M$ ← M = bytes

$= N * M$

S.C. $= O(N * M)$

$= \boxed{O(N)}$

✅ Drawback of RE:

$\boxed{RE}$ solution Always stack (RAM) पर space leta Hai.