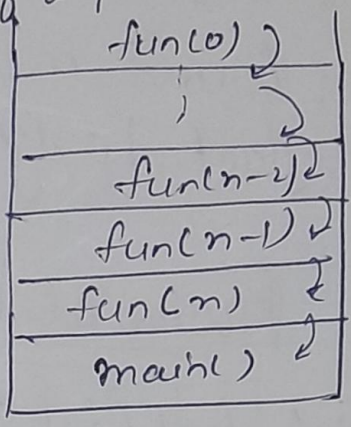
Time & Space complexity of Recursive

Solution :-

$T(n)$ → Time Taken as a function of input 'n'.

Recursive function maintain stack.

Base case reach hoga toh ~~stack~~ fun() ko unbind ~~kar~~stack unbind karega




```
void printArray(int a[], int n){
    if(n==0) return;
    cout<< *a<<;
    printArray(a+1, n-1);
}
```

T.C
Time taken by any algorithm ~~to~~ with respect to it's func^n

① $F(n) = k + F(n-1)$ — k

$F(n-1) = k + F(n-2)$

$F(n-2) = k + F(n-3)$

⋮

$F(0) = k + F(0)$

+ $F(0) = k_1$

$F(n) = nk + k_1$

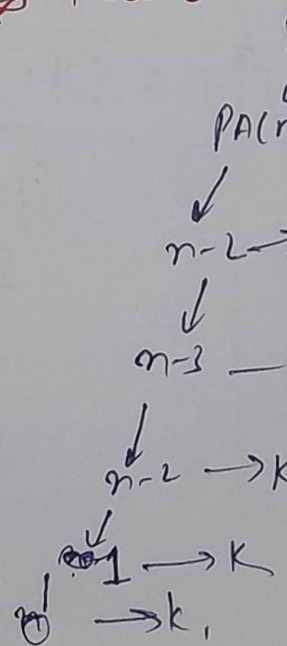
$T(n) = nk + k_1$

* Formula Based Method

$T(n) = O(nk + k_1)$

$= O(n)$

* Recursive Tree Method :-



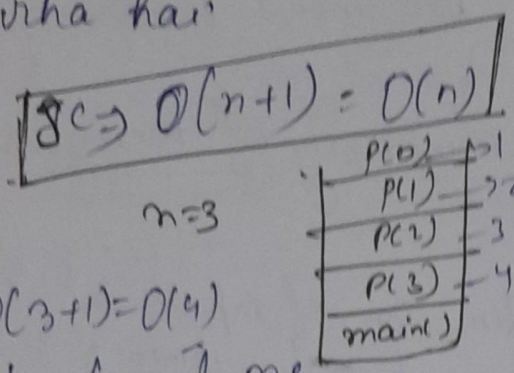
$T(n) = nk + k$

$T_n = O(n)$

Space complexity :-

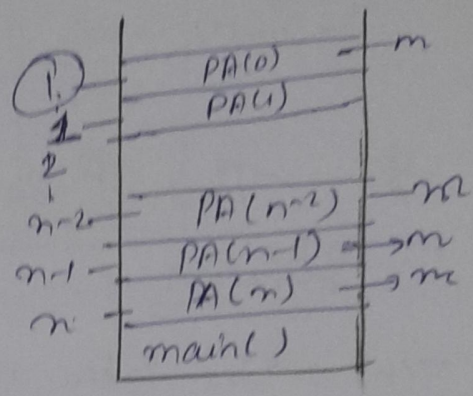
khud se kuch hum memory allocation nhi kar rhe hai OS stack me khud se manage kar rha hai

$SC \Rightarrow O(n+1) = O(n)$



$O(3+1) = O(4)$

Upper bound me constant ko ignore kar dete hai



Factorial RE :-

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

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

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

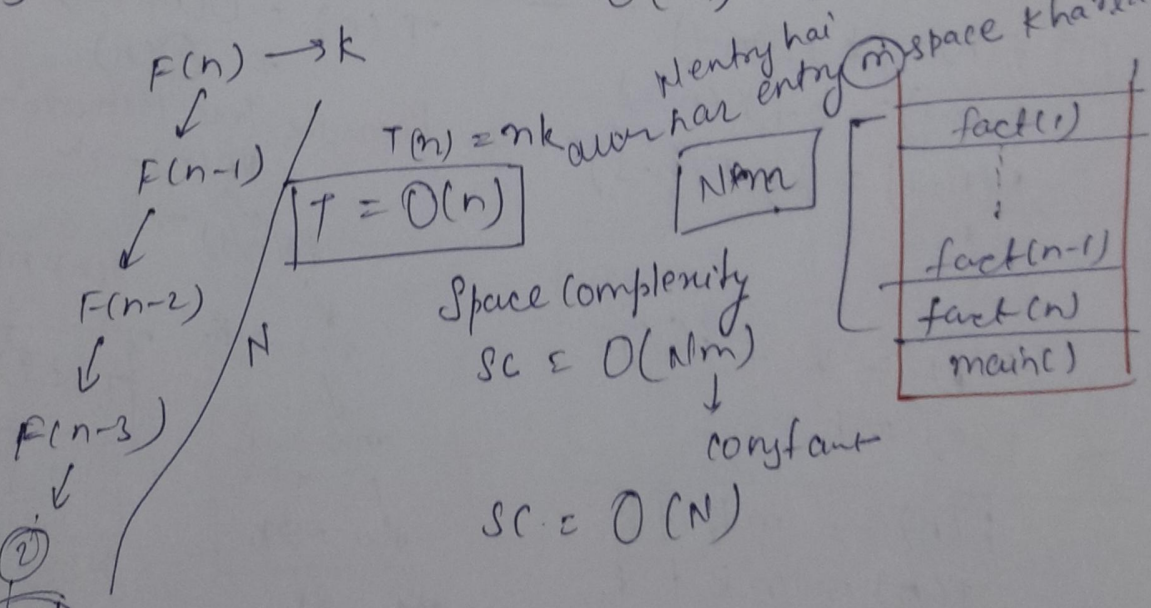
$T(n-2) = k + T(n-3)$

$T(1) = k$

$T(n) = nk$

$O(T_n) = O(nk)$

3

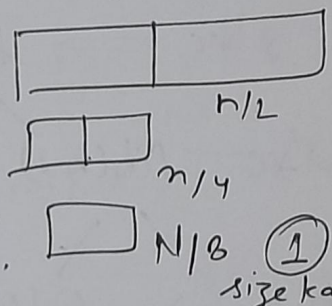


Binary Search RE

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

Iterative BS is no better quality than Recursive because it takes less memory.

kuch k processing lgti hai and fir dubara se call kar dete hai n/2 ke liye.



$F(n) = k + F(n/2)$

$T(n) = k + T(n/2)$

$T(n/2) = k + T(n/4)$

$T(n/4) = k + T(n/8)$

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

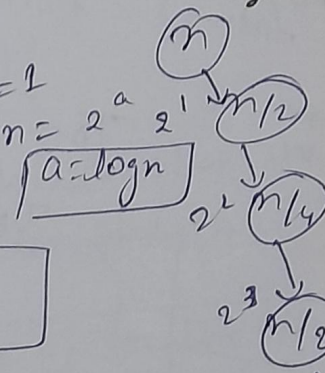
$T(1) = k$

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

$n = 2^a$

2^1

$a = \log n$



$T(n) = \log n$

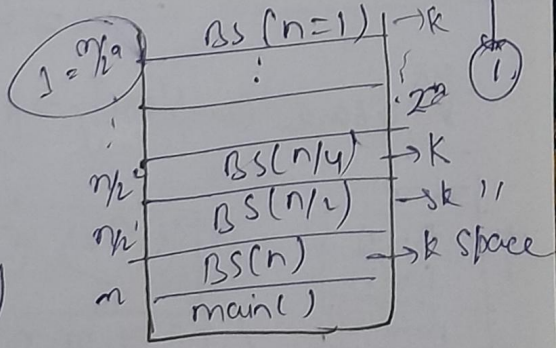
$T(n) = a + k$

Space-Complexity

$a = \log n$

$O(Tn) = O(k \log n)$;

S.C $\Rightarrow O(\log n)$

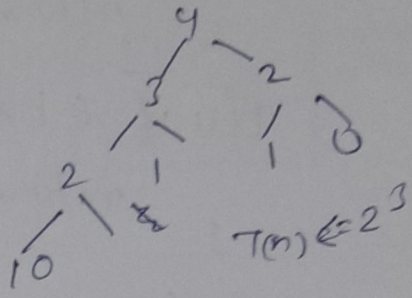


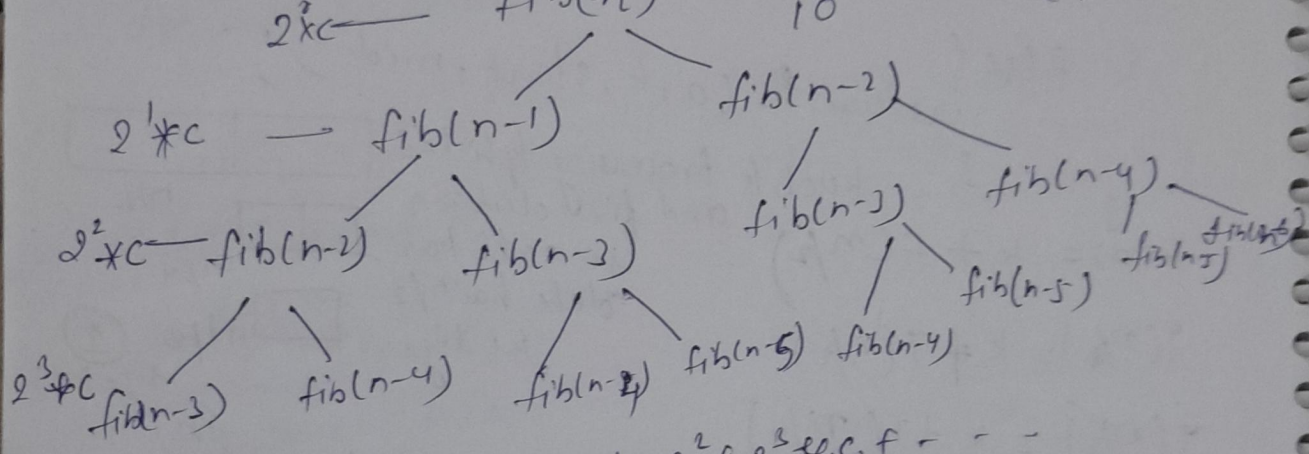
Recursive Algor ka drawback hota hai vo stack pe space le ke aa jate hai.

Two method To find complexity of RE
① Ek Visual Tree
② Mathematical Tarika fun^
Bikar Bekar Tarika hai ye

fibonacci Series RE

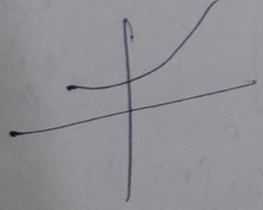
```
int fib(int n){
  if(n==0||n==1)
    return n;
  return fib(n-1)+fib(n-2);
}
```





$T(n) \le 2^k c + 2^1 c + 2^2 c + 2^3 c + \dots$

$T(n) \le [2^0 + 2^1 + 2^2 + \dots + 2^{n-1}]$

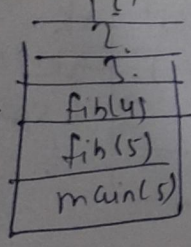


$T(n) \le 2^n - 1$

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

Crazy high T.C.

Space Complexity → kitna depth me jaa rha hai



BC ⇒ O(n)

(n==1 or n==0) return n..