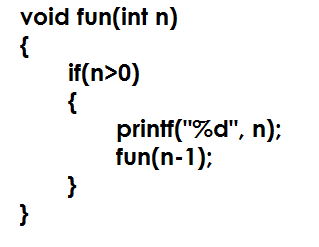
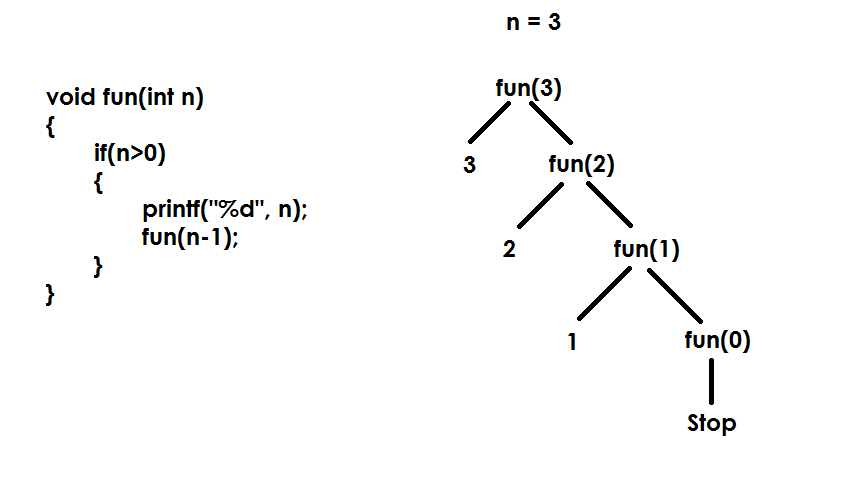
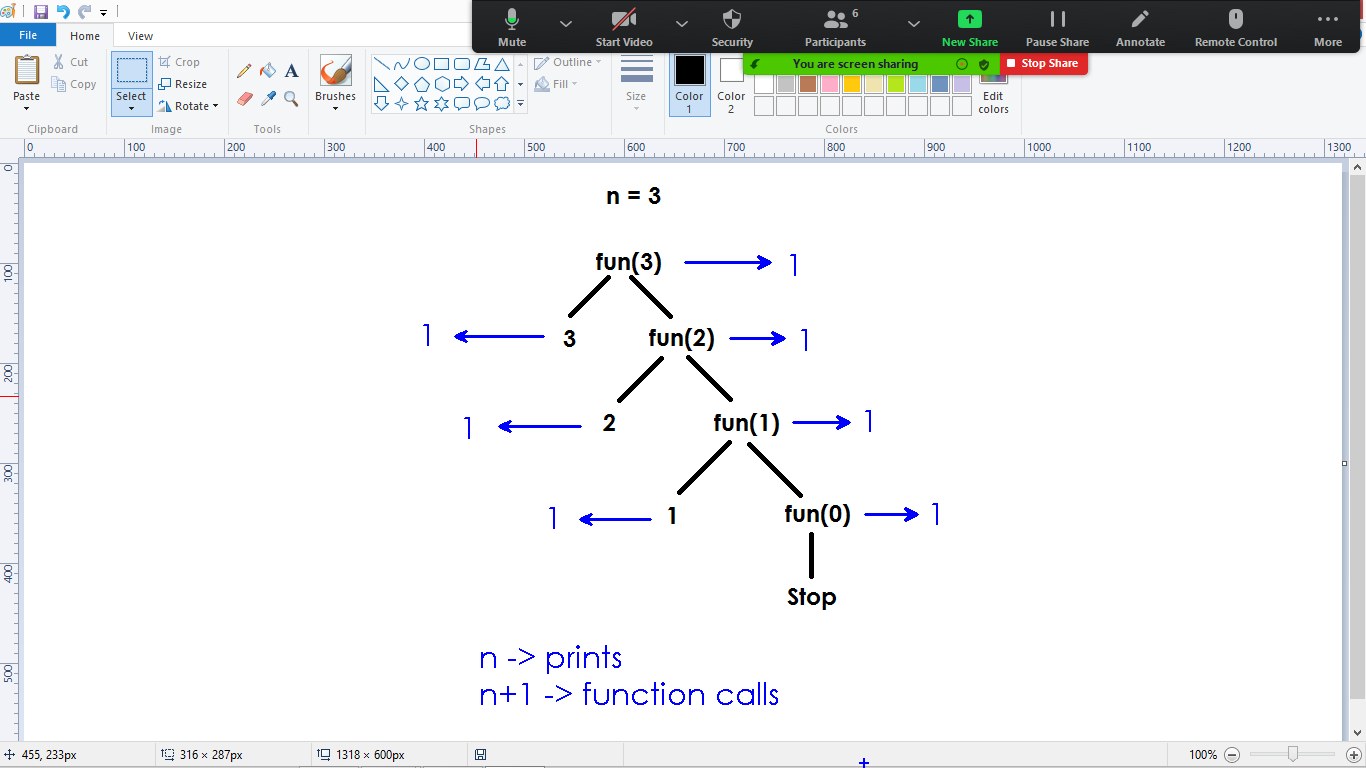
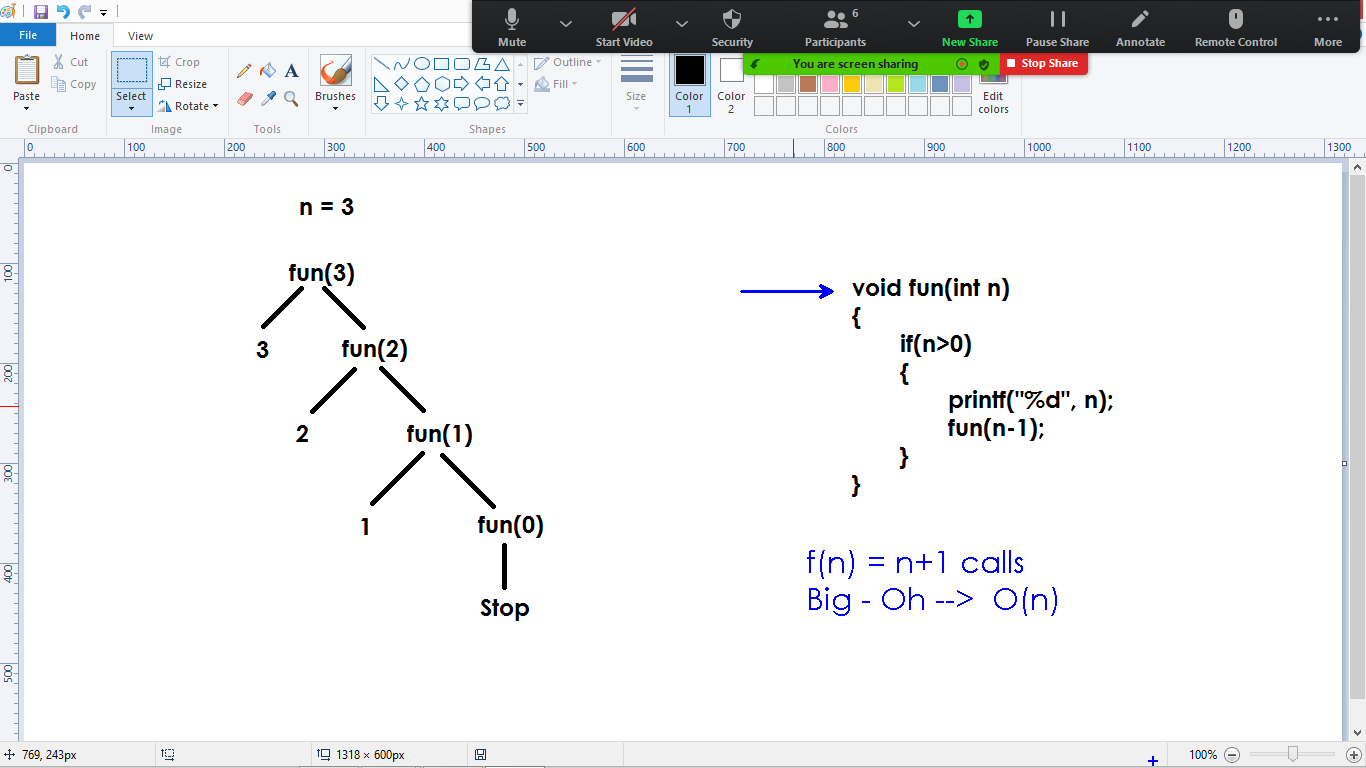
**Recurrence relation:**

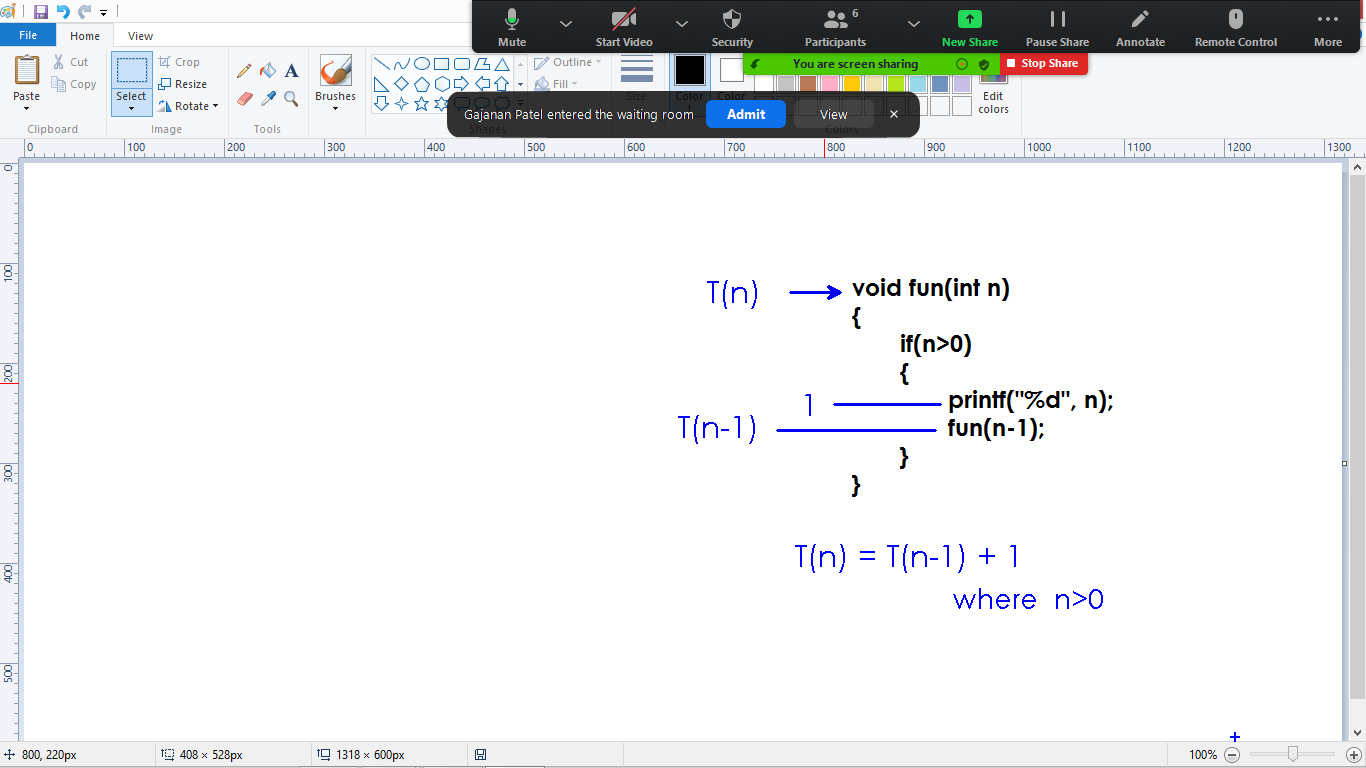
* Finding the time of a function which is re-occurring.
* We can understand the recurrence relation using a recursive function.



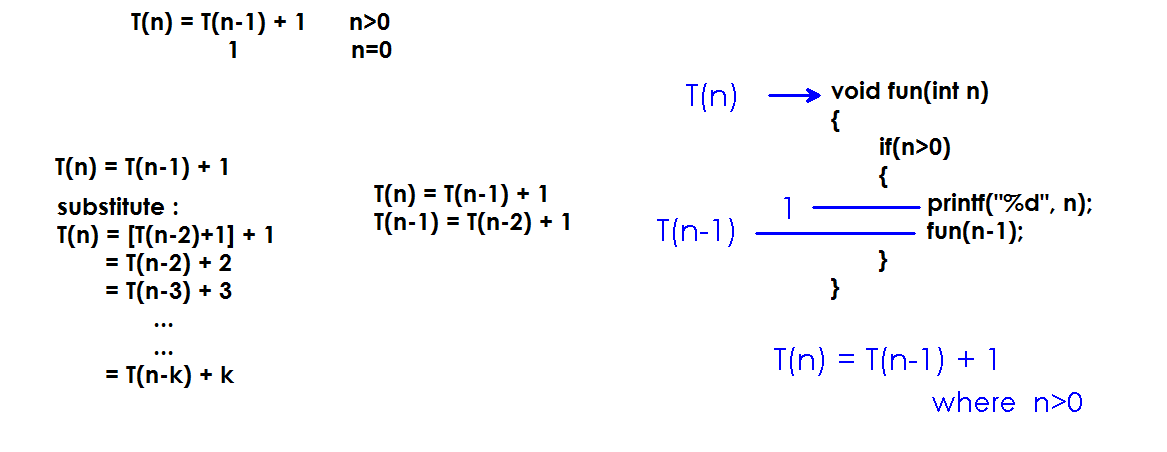






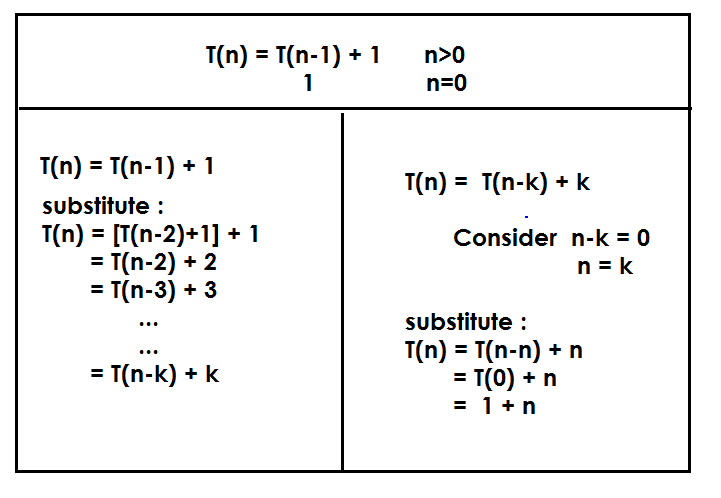


Recurrence relation:



Recurrence ends with final call where n value is 0.

Hence kth iteration result is 0



**Binary Search:**

**Bi\_Se(arr, low, upp, key)**

**{**

**mid = (low + upp)/2;**

**if(arr[mid] == key){**

**return mid;**

**}**

**else{**

**if(arr[mid] > key)**

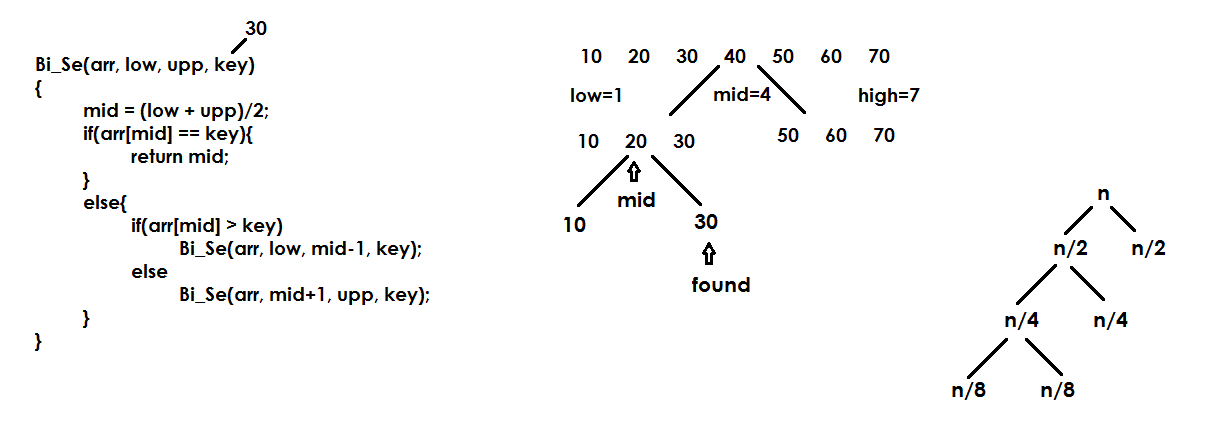
**Bi\_Se(arr, low, mid-1, key);**

**else**

**Bi\_Se(arr, mid+1, upp, key);**

**}**

**}**





**Dynamic programming:**

* In normal method of solving a problem, we follow one algorithm blindly to get the output.
* We cannot overcome the problems of taking memory and taking time while following any algorithm.
* Dynamic programming is an idea of changing the algorithm dynamically while solving the problem.

**Recurrence relation of another code:**

void Test(int n)

{

if(n>0)

{

for(i=n ; i>0 ; i--)

{

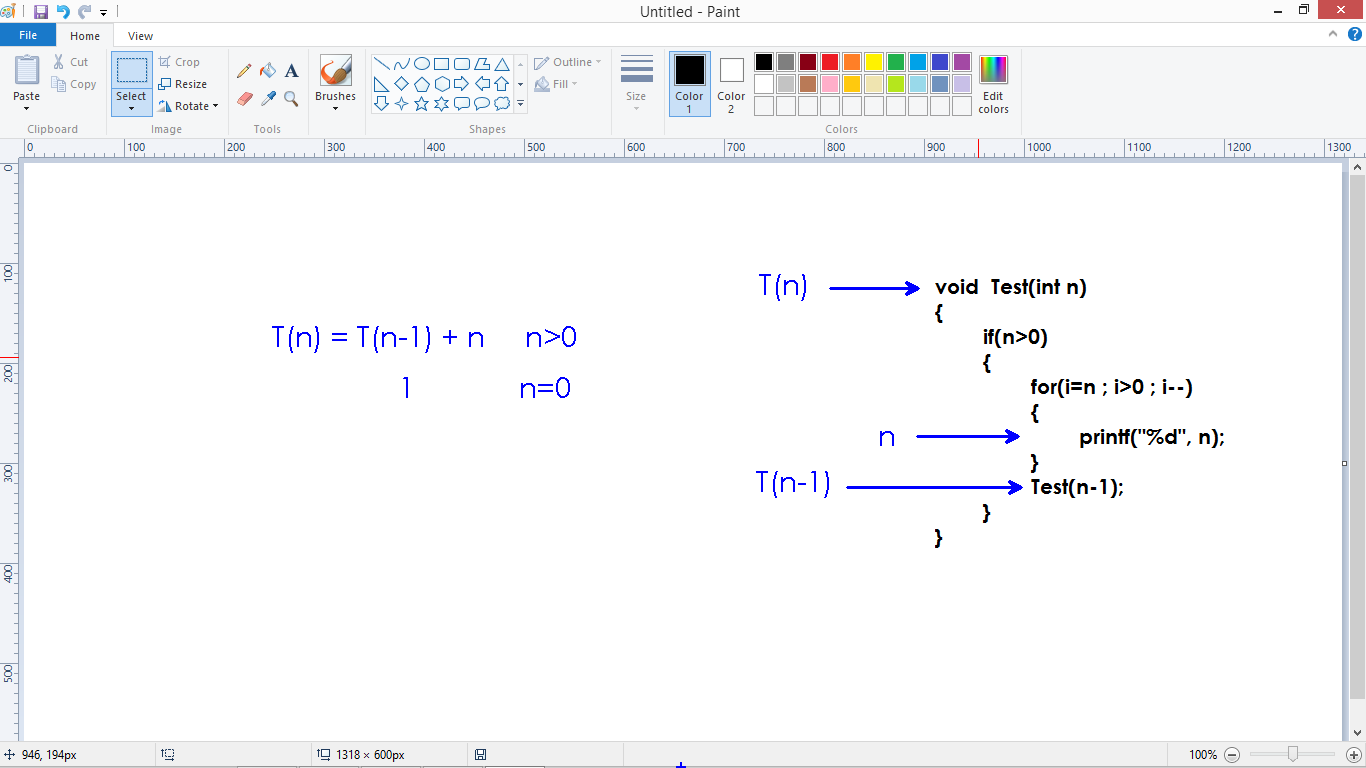
printf("%d", n);

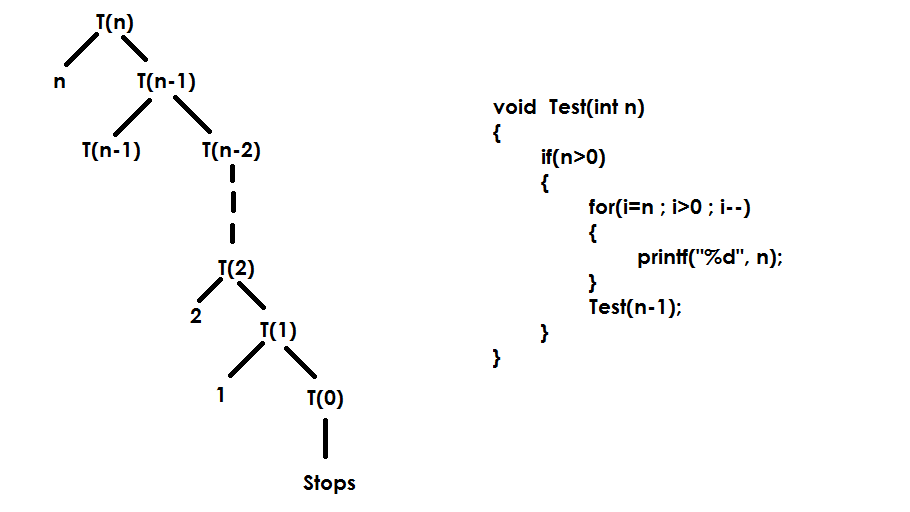
}

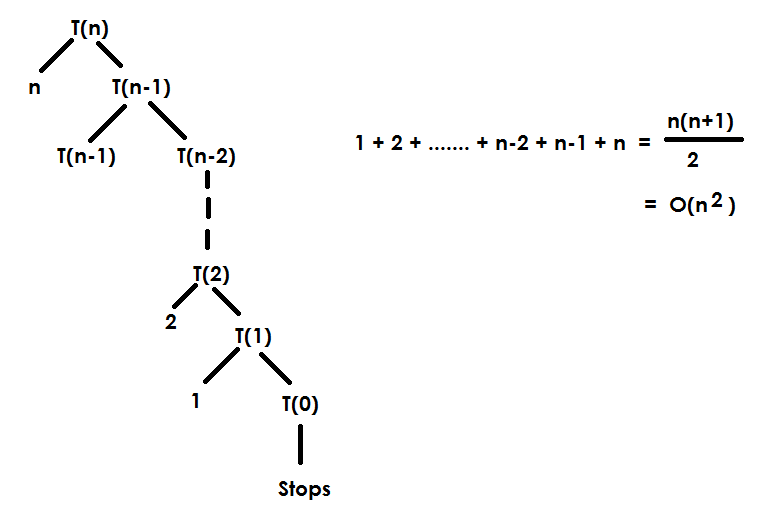
Test(n-1);

}

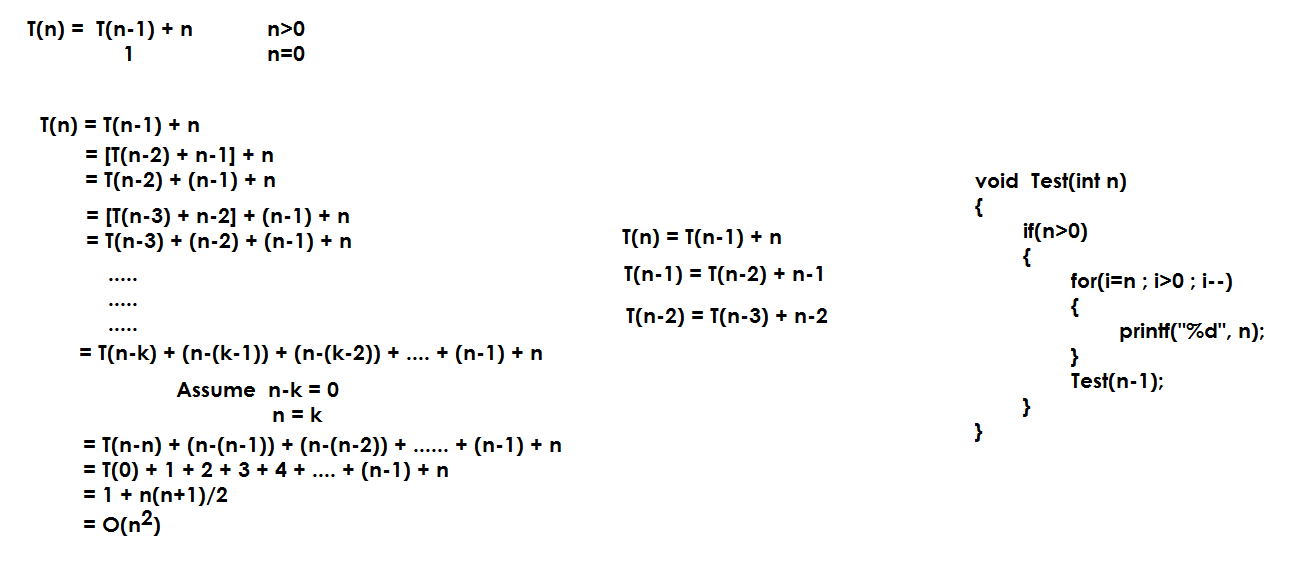
}





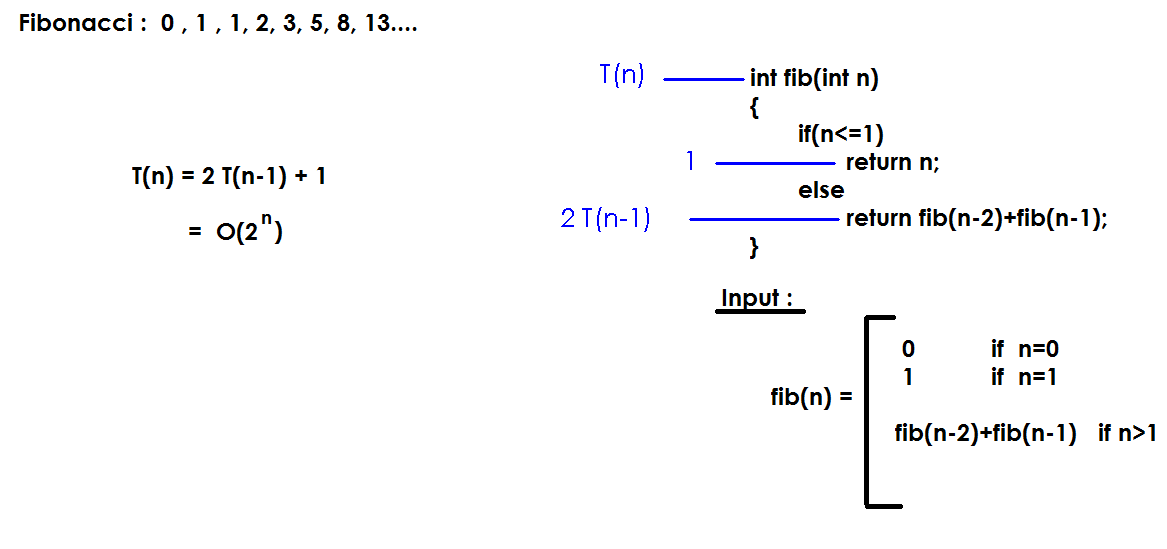


**Substitution method:**

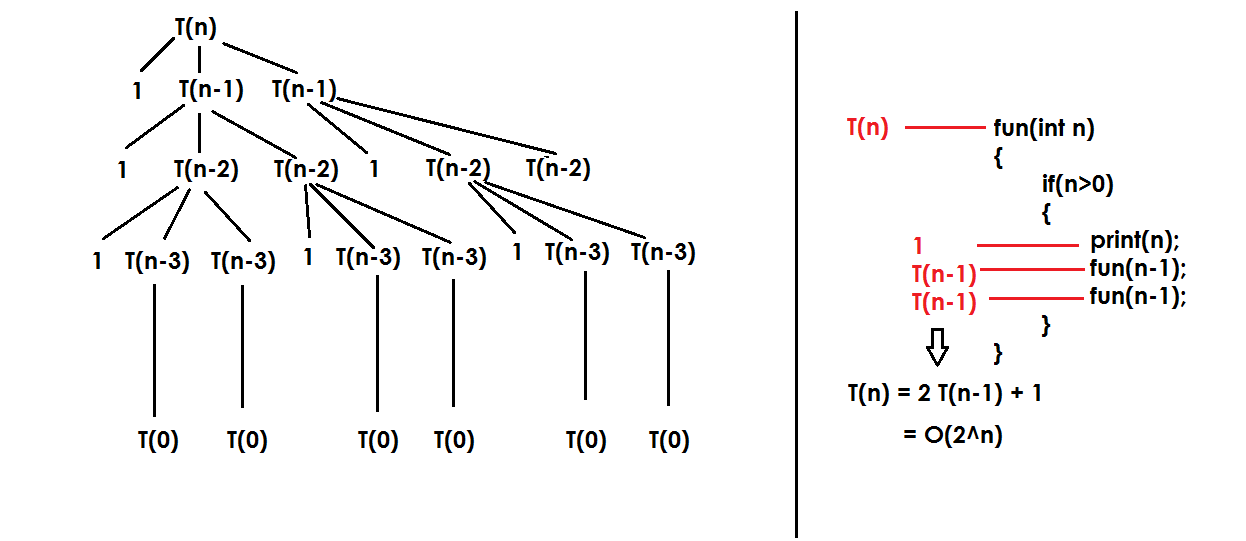


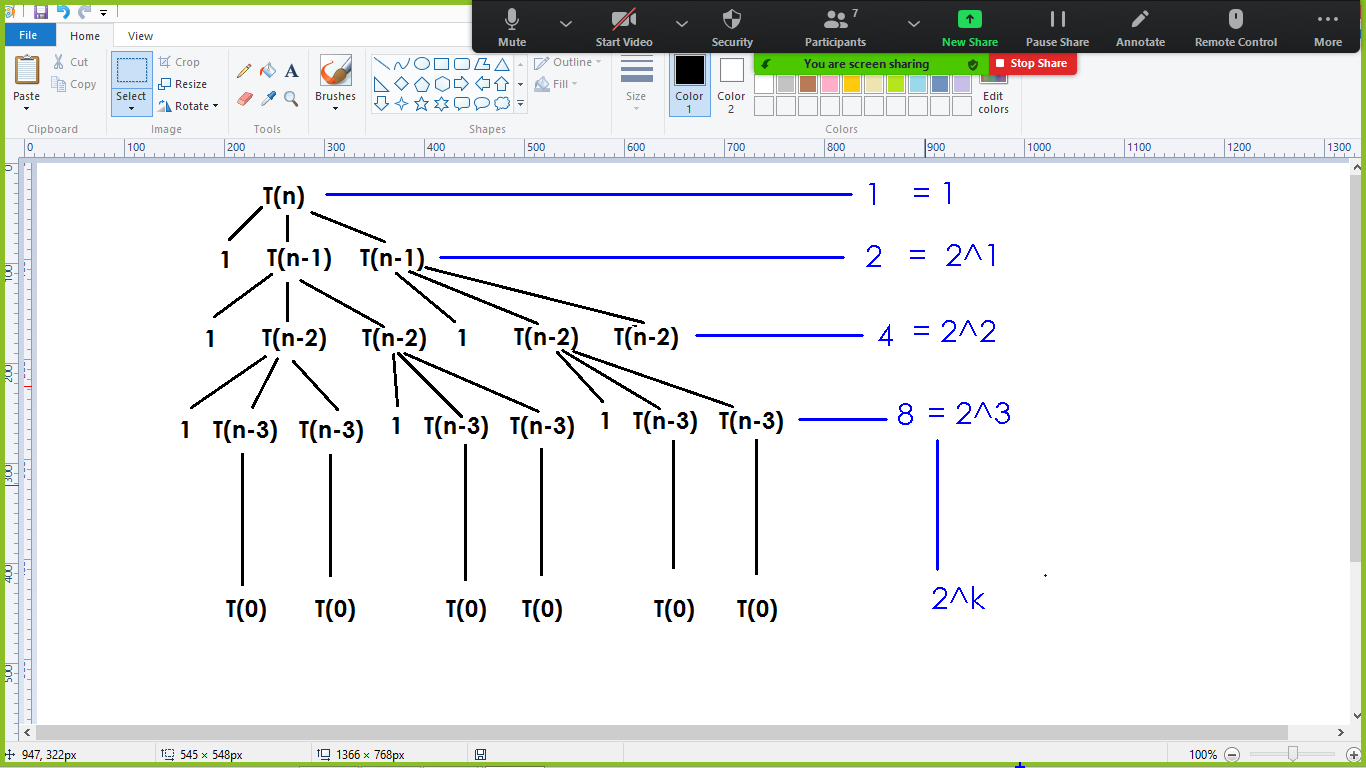
**Optimization Problem:**

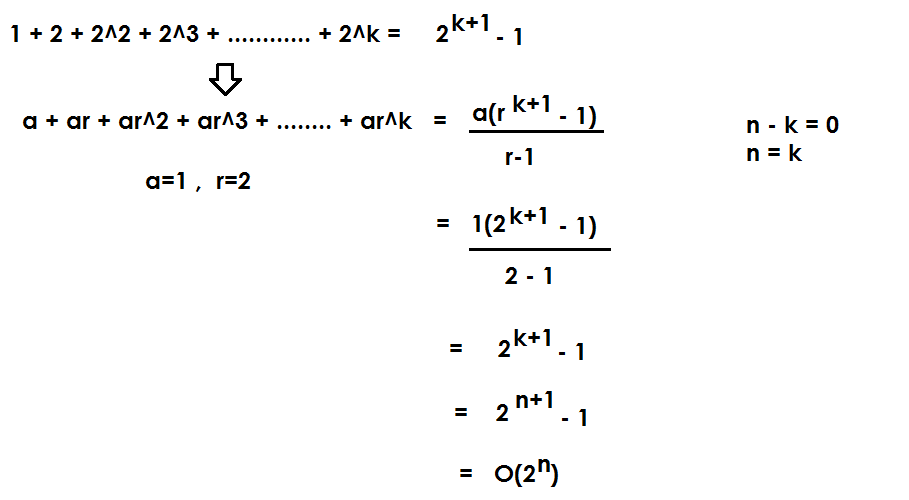
* Consider a problem.
* Either maximizing or minimizing the function using available options(situations) in the process of finding a solution.
* The traditional approach of solving a problem using methodology is called Greedy method.
* For example, finding a Fibonacci series using recursion.



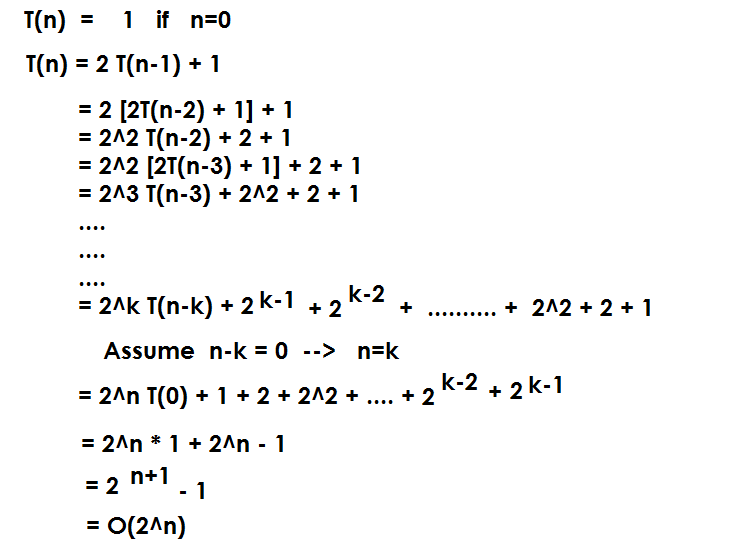
**Recurrence relation:**

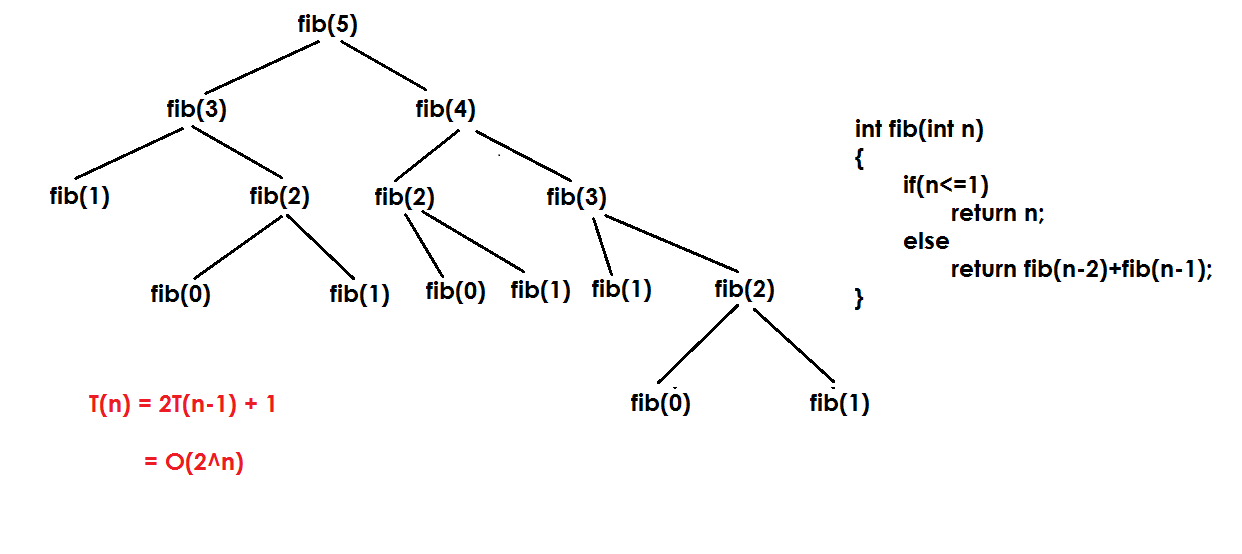


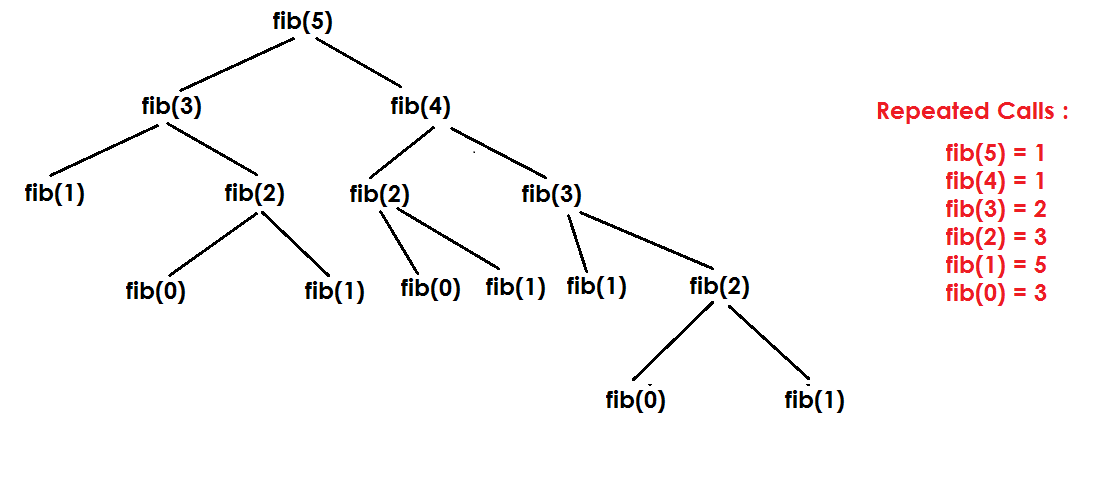


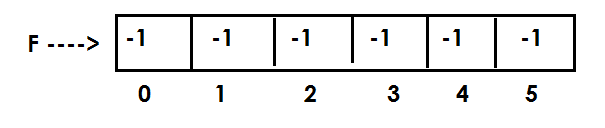


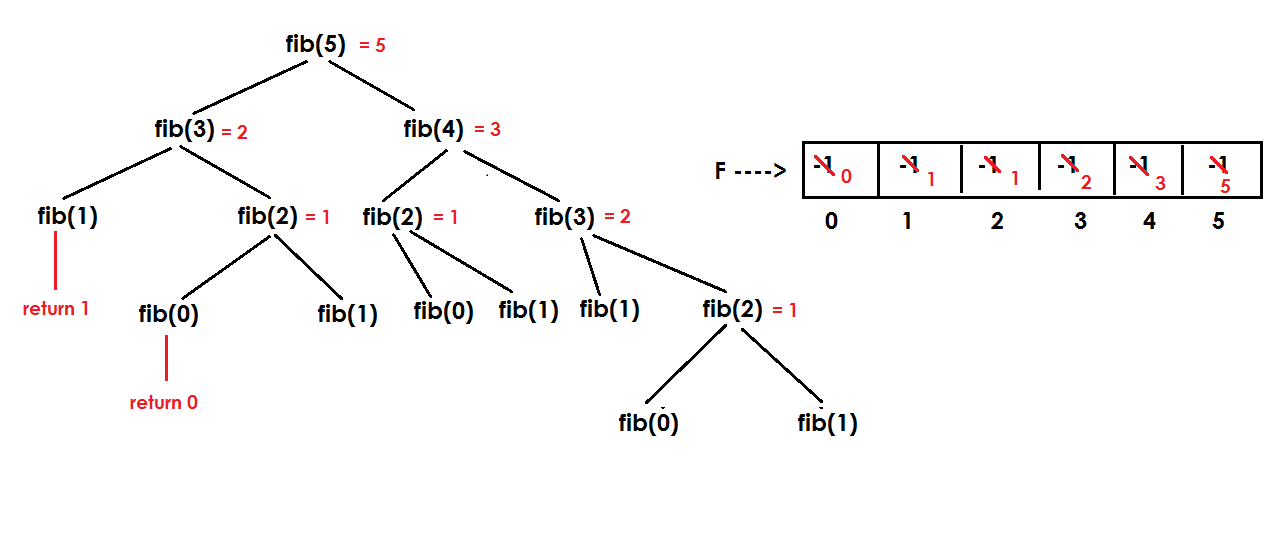
**Substitution method:**



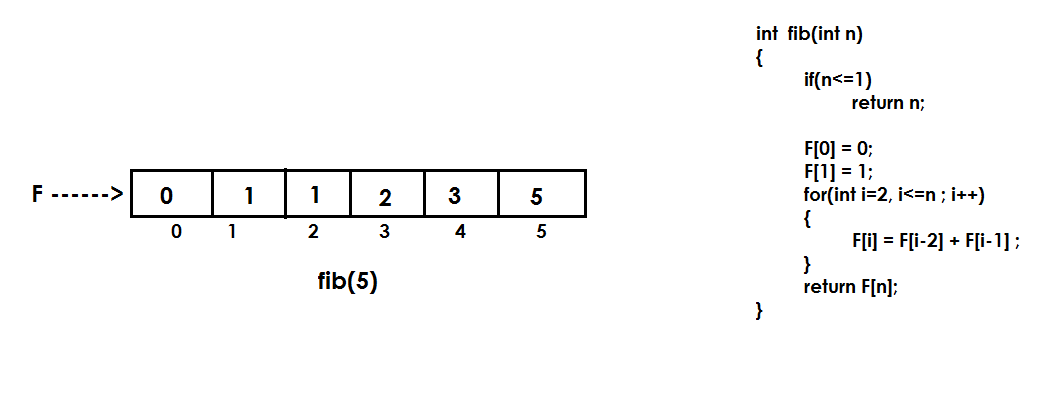








**Fibonacci without recursion:**

****