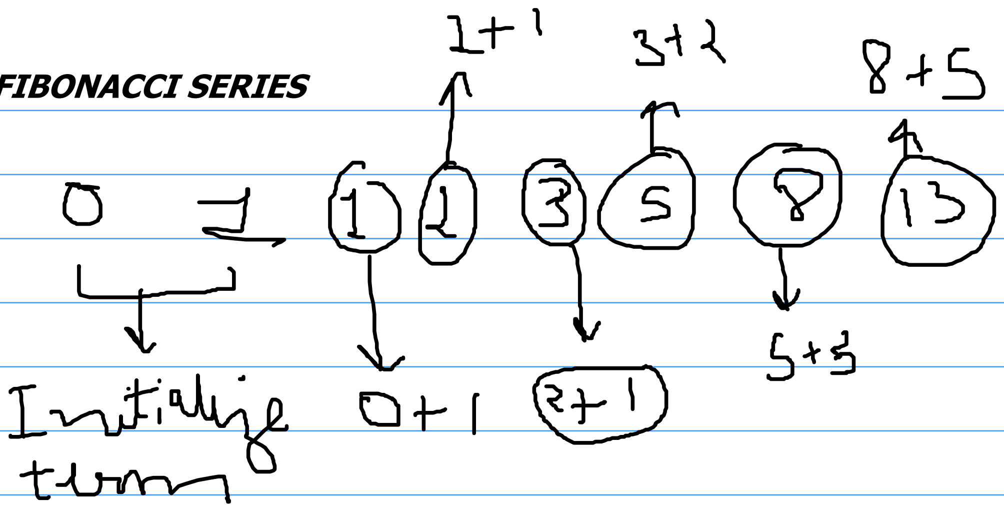
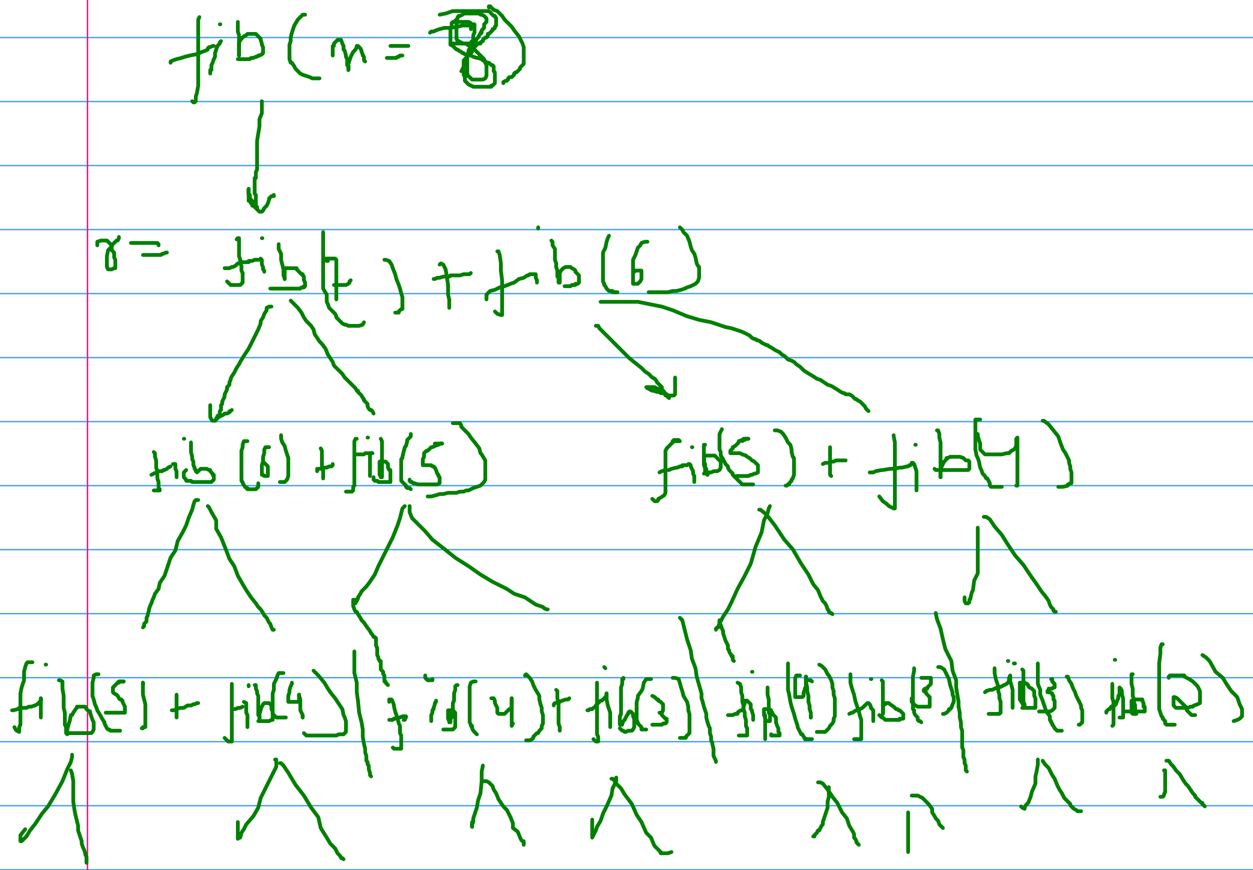
FIBONACCI SERIES



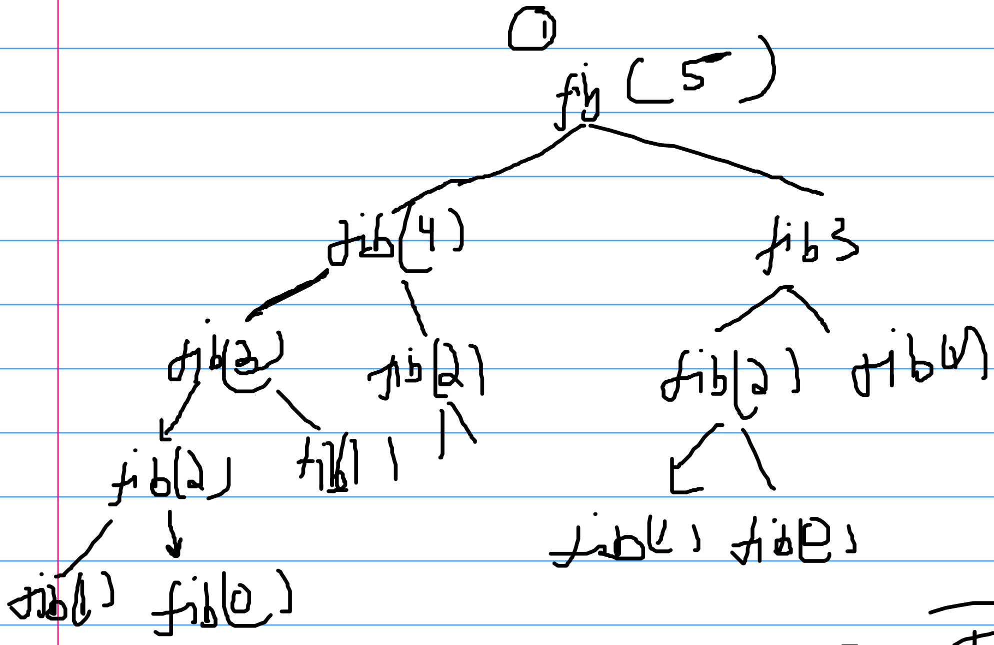
$$fib = \begin{cases} 1 & n = 1 \\ 0 & n = 0 \\ fib(n-1)+fib(n-2) & n > 1 \end{cases}$$



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

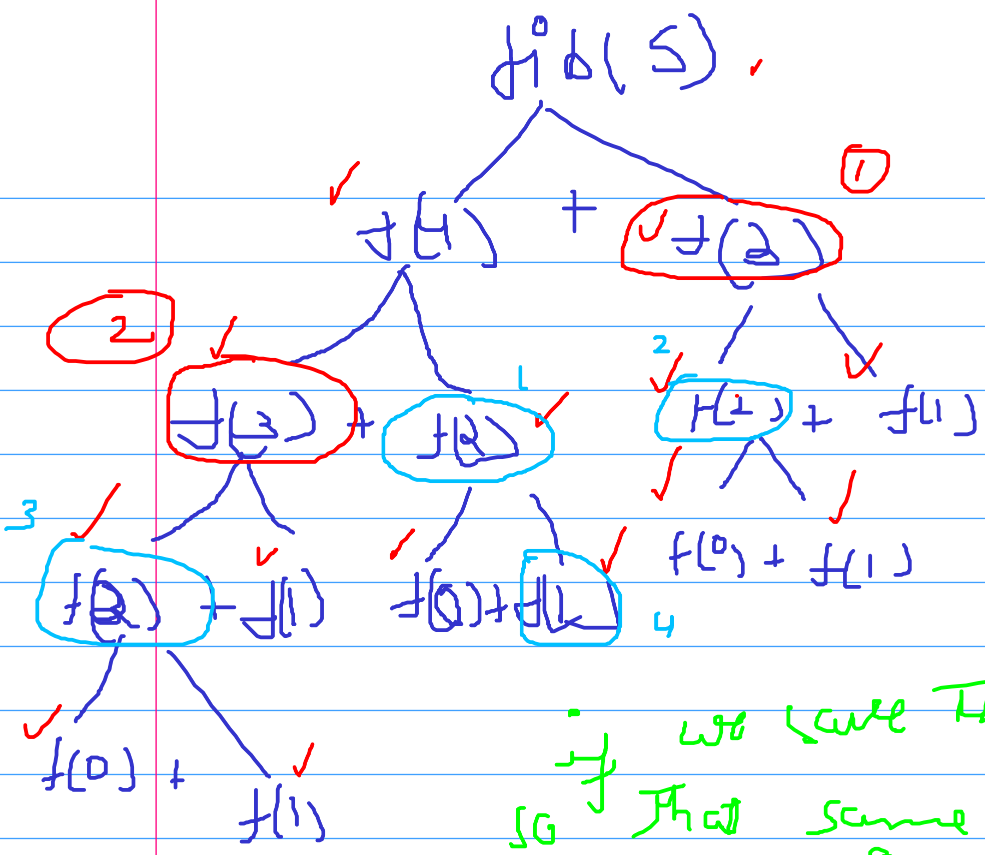
```
for ( i = 0 ; i < n ; i ++ )
{
        cout << fib ( i ) ;
}
```

you can see that the no of calls are large and also same value of n there are multiple calls



Iterative version : faster than the recursive version

```
t1 = 0 , t2 = 1 , next = t1 + t2 = 1
for ( i = 0 ; i < n ; i ++ )
{   if ( i < 1 )
        cout << i << " " ;
    else
        cout << next << " " ;
        t1 = t2 ;
        t2 = next ;
        next = t1 + t2 ;
}
```



f(5) -> 15 calls
f(3) -> 5 calls
f(2) -> 2 calls

if we save the result of fib
so that same no. of calls no happen again.

for same process f(3) call twice

$f_6 \rightarrow$

0	1	1	2	3	5
0	1	2	3	4	5

let take an array (static) have (n) element

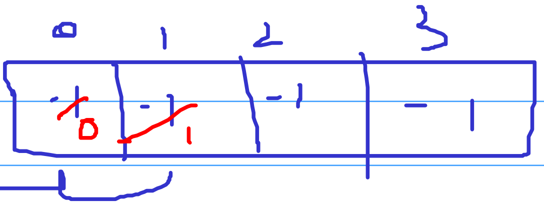
arr → memoization

-1	-1	-1	-1	-1	-1	-1
0	1	2	3	4	5	6

values are stored in these element.

```
for (i=0 ; i<(n); i++)
{
    cout << fib(i) << " ";
}
```

```
fib (int n)
{
    if (n <= 1)
    { arr[n] = n;
      return n;
    }
    else {
        if ( arr[n-2] == -1)
        { Arr[n-2] = = fib(n-2)
        }
        if (arr[n-1] = = -1)
        { arr[n-1] = fib(n-1);}
    return arr[n-1] + arr[n-2];
}
```



0
1
2
3
~~-1~~ 0
~~-1~~ 1
-1
-1