Fibonacci Series

fib(n)	0	1	1	2	3	5	8	13
n	0	1	2	3	4	5	6	7

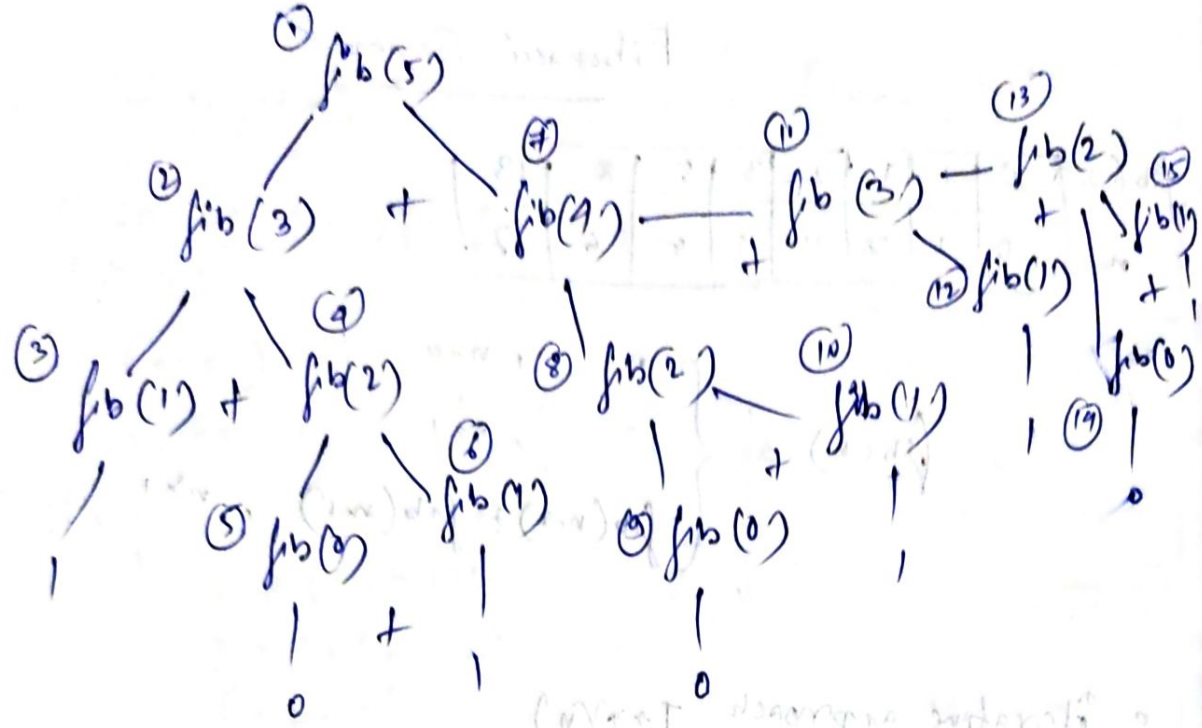
$$fib(n) = \begin{cases} 0 & , n=0 \\ 1 & , n=1 \\ fib(n-2) + fib(n-1) & , n>1 \end{cases}$$

- **Iterative approach** $T \to O(n)$

```
int fib (int n)
{   int term0=0, term1=1, sum;
    if (n<=1) return n;
    for(int i=2; i<=n; i++)
      {  sum = term0 + term1;
         term0 = term1
         term1 = sum
      }
    return sum;
}
```

- **Recursion**

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



fib(5) → 15 calls
fib(4) → 9 calls
fib(3) → 5 calls
} No pattern

So, fib(n-2) + fib(n-1) → Assuming this is also fib(n-1)

↳ 2 fib(n-1)

↳ When a func is calling 2 times by reduced value of one.

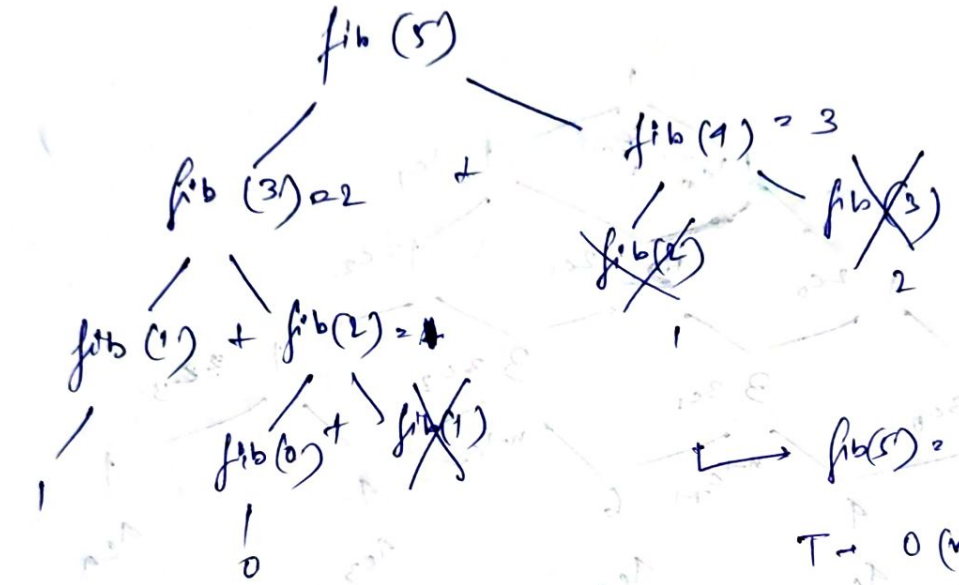
T → O(2^n) (approx)

• Optimized Approach (Memorization)

↳ we would store the value of some previous calls to stop making unnecessary extra calls again

Store

x_0	x_1	x_2	x_3	x_4	x_5
0	1	2	3	4	5



↳ fib(5) = 6 calls

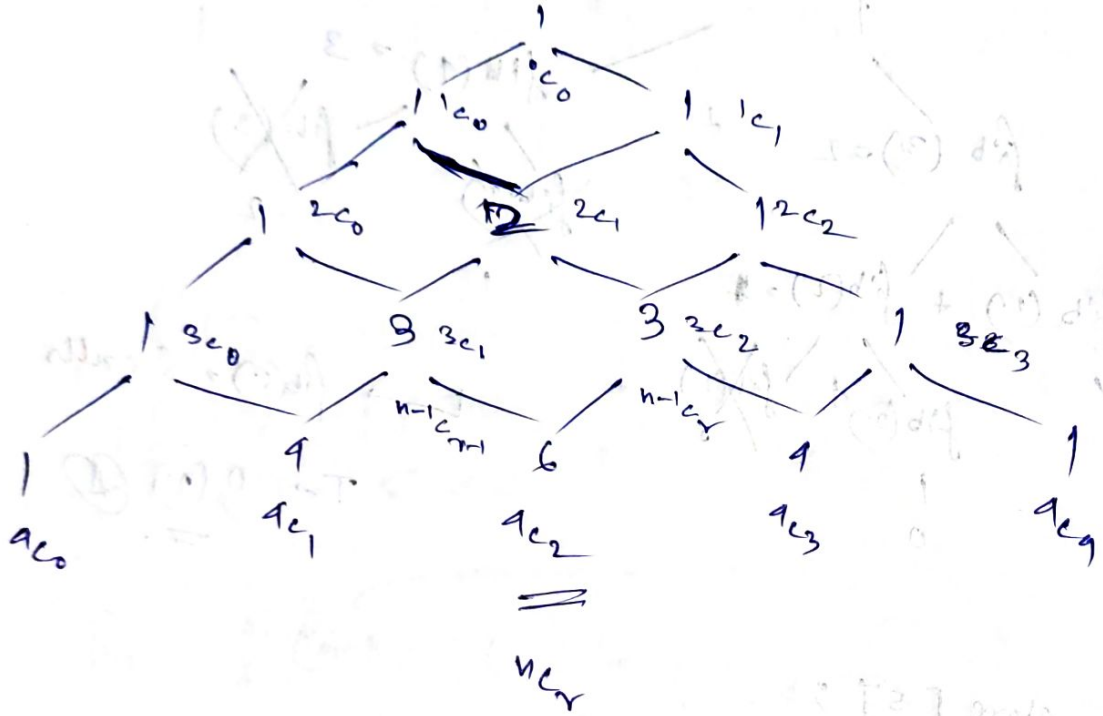
T → O(n) (A)

```
int store[5];
int fib(int n){
    if (n<=1){
        store[n] = n
        return n;
    }
    else {
        if (store[n-2] == -1)
            store[n-2] = fib(n-2);
        if (store[n-1] == -1)
            store[n-1] = fib(n-1);
        return store[n-2] + store[n-1];
    }
}
```

Combination formula ^nC_r

$$^nC_r = \frac{n!}{r!(n-r)!}$$

Pascal's triangle



So $^nC_r = {}^{n-1}C_{r-1} + {}^{n-1}C_r$

```
int combo (int n) {
    if (r==0 || r==n)
        return 1;
    return combo(n-1, r-1) + combo(n-1, r);
}
```