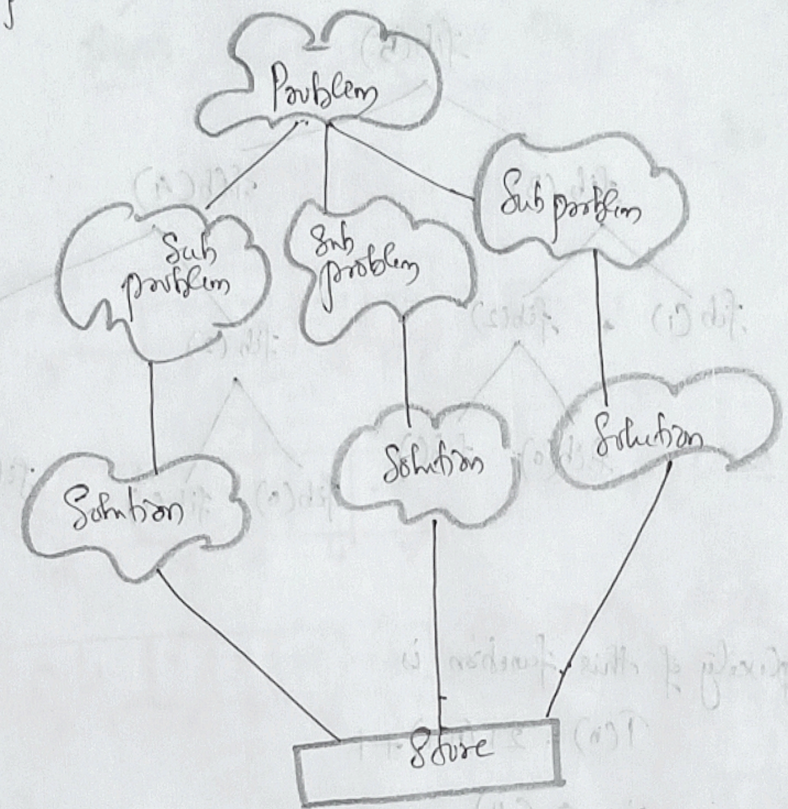
Dynamic Programming

→ It is one of the algorithm design technique.
→ It is used when problem breaks down into recurring smaller sub problems.
→ It is typically applied to optimization problem.
→ In such optimization problems there can be many solutions and each solution has a value and we wish to find a solution with optimal value.



Every dynamic problem has 2 properties:

1) Overlapping Subproblem → If the problem can be broken down into subproblems which are reused several times rather than generating new subproblems.
{ Solutions are stored in a table }

2) Optimal Substructure.
A given problem is said to have optimal substructure, if the solution of the problem can be obtained from the optimal solutions of its subproblems.

[Note: dynamic programming is not useful in problems which do not have overlapping subproblems.]

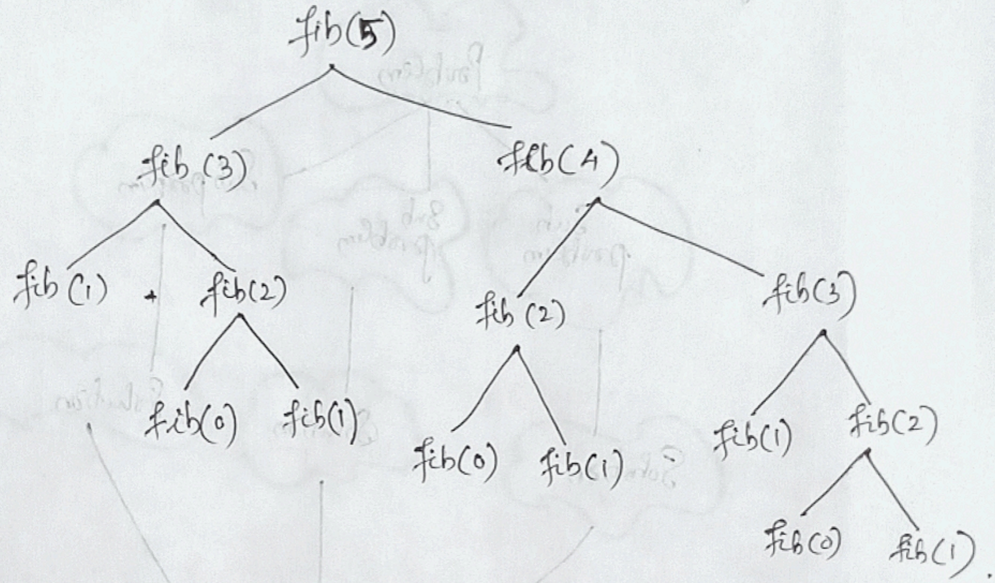
Fibonacci series
0, 1, 1, 2, 3, 5, 8, ...

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

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

}.
```

Suppose we want to find the 5th term of this fibonacci series, the recursive tree is as shown below.



Complexity of this function is

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

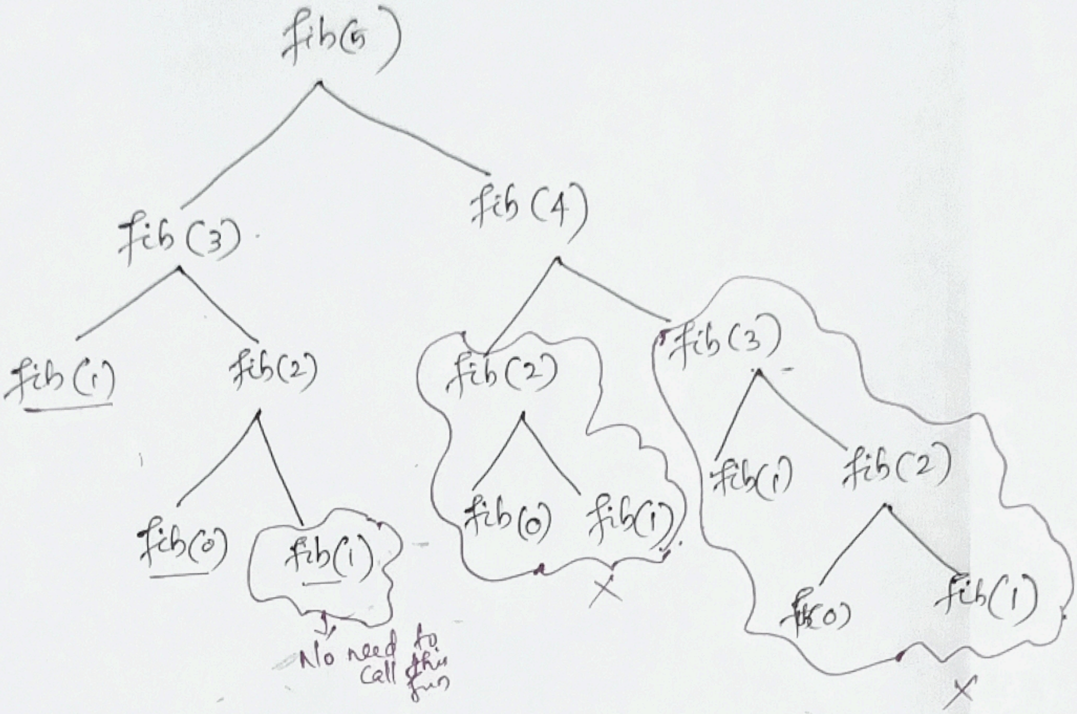
ie, $\underline{O(2^n)}$

By using dynamic programming we can reduce the no: of recursive calls and there by the time required for its execution.
For that we use a global array for storing the subsolutions.

F	0	1	2	3	4	5
	-1	-1	-1	-1	-1	-1

Initially the values are set -1. Whenever we get a solution for a value, it is filled and reused later.

Considering the recursion trace,



F

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

fib(1) = 1

F

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

fib(0) = 0.

F

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

fib(2) = fib(0) + fib(1) = 0 + 1 = 1

fib(3) = fib(1) + fib(2)

= 1 + 1 = 2

F

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

fib(4) = fib(2) + fib(3)

= 1 + 2 = 3

fib(5) = fib(3) + fib(4)

= 2 + 3 = 5

fib(n) = n+1 calls.

i, Complexity O(n).