Objectives

- Introduction to Dynamic Programming
- Difference between Greedy Method and Dynamic Programming
- Memoization and Tabular method
- Examples of Memoization and Tabular method
- Matrix Chain multiplication problem

Dynamic programming (DP):

DP is used to solve problems which have certain properties: (i) optimal substructure (ii) overlapping subproblems.

Optimal substructure: to get the optimal solution of the main problem, we have broken down to subproblems. eg. If a problem is divided into number of subproblems say $\{p_1, p_2, \ldots, p_k\}$and if the optimal solution to subproblem p_1 is obtained, then to solve p_2 , we can use the result of p_1 to get the optimal solution of $p_2$and repeat the process until we solve the entire problem. Then we can say that the problem has optimal substructure property.

Overlapping subproblems: when we break a problem into subproblems, we need to calculate some task multiple times. If it happens then we can say that problem has overlapping subproblems.

The optimal solution to the problem is obtained by combining optimal solution to the subproblems.

Many problems using DP is solved in polynomial time to which a naive based approach could take exponential time. DP is a method to optimize the solution of a problem that is in many cases time complexity from exponential time down to polynomial time complexity.

DP is used to solve (i) Combinatorial problems or counting problems (ii) optimization problems.

For example, the problems (combinatorial) like:

1. How many ways to traverse a graph?
2. How many steps are needed to get from point A to point B?

Similarly if we want to find the minimum no. of steps required to get from point A to point B (optimization problem).

So,

- DP is a algorithmic technique to solve combinatorial and optimization problem utilizing the fact that the optimal solution of the overall problem depends upon the optimal solution to its overlapping subproblems.

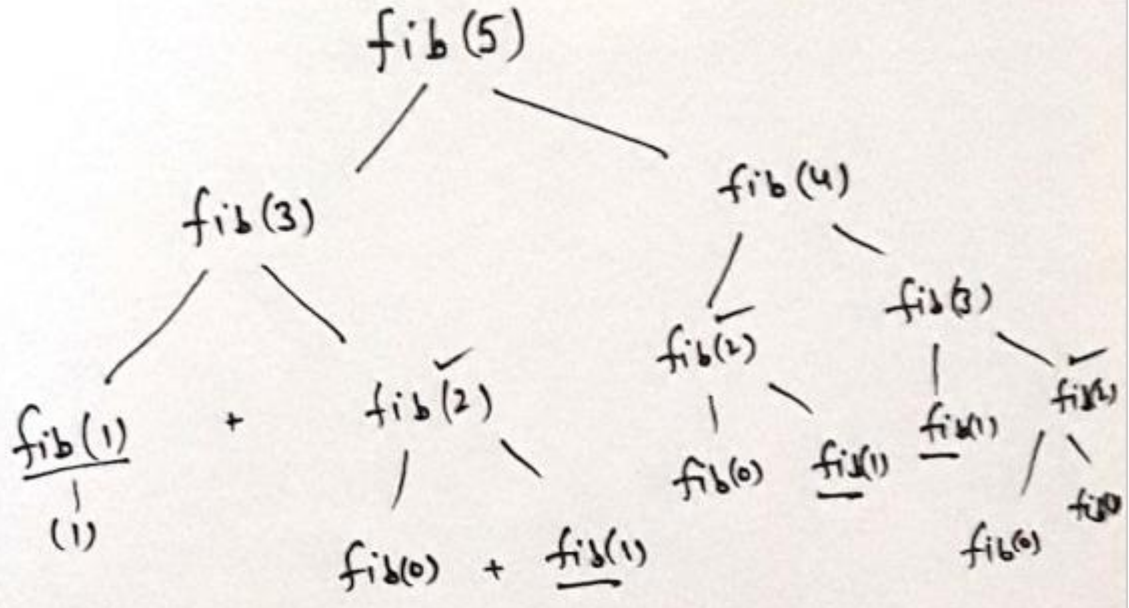
- Mostly the DP problems are solved by using recursive formulas. Although we will not use recursive programming but recursive formula is used. Recursion can also be used, but mostly DP problems can be simplified using iteration.
- DP follows the Principle of optimality, which means that a problem can be solved by sequence of decisions to get the optimal solution.

Dynamic programming adopts (i) Memoization (ii) Tabular method to solve problems.

Memoization:

Consider the recursive procedure of the Fibonacci series is:

```
int fibonacci(m)
{
if (m <= 1)
return 1
else
return fibonacci(m - 2) + fibonacci(m - 1)
}
```



If I assume $fib(n-2) = fib(n-1)$

then

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

By Master's Th'm of decreasing fn.

$$= O(2^n)$$

Time taken $= O(2^n)$ - Exponential time Complexity

Take a Global array

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

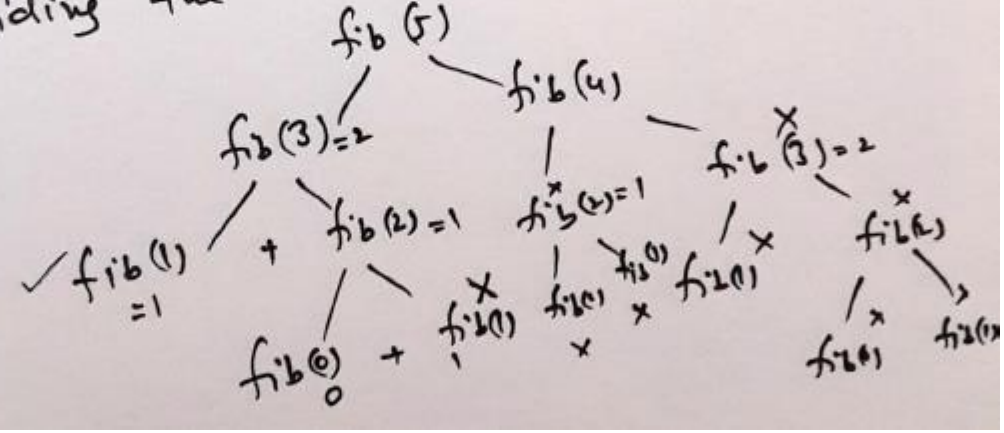
Total no. of Calls = 6

fib(n) = (n+1) Calls

= O(n)

This is the Memoization.

By storing the result of the fn., we are avoiding the same call once again.



Thus Memoization has reduce time from

$$2^n \longrightarrow n$$

In this we observed that tree is generated, fn. Calls are avoided & result is obtained back.

This is a Top-down approach.

So Memoization follows Top-down app

Generally we use iterative method in Dynamic Programming i.e. Tabulation method is used in DP.

Tabulation Method for Dynamic Programming.

fib(5)

```
int fib(n)
{ if (n <= 1)
        return n;
    f(0) = 0;  f(1)=1
  for (i=2; i<=n; i++)
    { f(i) = f(i-2) + f(i-1)

    } return f(n)
```

3^3

f →	0	1	1	2	3	5
	0	1	2	3	4	5
			i	i	i	i

Tabulation method follows bottom up approach.

THANKS !