Citing the above recursion tree we can easily spot the presence of duplicate recursive calls for instance: fib(n-2), fib(n-3) ... etc. Its the presence of these duplicate recursive calls that presents us the opportunity to apply the concept of dynamic programming to a problem as then we could just cache the solutions to these duplicate sub-problems and use it when we encounter them next.

NOTE: Whenever there are multiple recursive calls being spawned there is a high likelihood that there might exist duplicate recursive calls presenting us an opportunity to apply dynamic programming.

2) The second way to identify the applicability of DP as a technique to solve a specific problem is that the problem would hint us to find an optimal solution. What we mean by this is firstly the problem could first be modelled as a function "f" on some variables and our job would be to figure out either the maximum or minimum value of that function which will be the optimal value depending on the problem.

- From all the discussion we have had above it leads us to the following 2 important points before we start writing a dynamic programming solution to a problem:

1) Always always, always first try to model the problem as a recursive function. This is the function that will provide the solution to the original problem. For instance in case of fibonacci series:

$$fib(n) = fib(n-1) + fib(n-2).$$