1. **Intro to Dynamic Programming : DP**
   Easy + Hard problems can be solved by DP. we will solve much problems to recognize the pattern of problems that can be solved by dp.

   Prerequisite : i. Recursion ii. Complexity Analysis

   Complexity is more important here. We need to analyze the problem's complexity which is matched with dp patterns. Then, we can approach it with the pseudocode of it.

2. **Back to Fibonacci!! :**
   fib (50) isn't showing any output, because normal recursion method is very slow. We should improve this situation by solving this problem with dp.

3. **Simulation :** Complexity of fibonacci in recursion method.
   (i),

```
int fun(int x){
    if(x == 1)return 1;
    return fun(x-1);
}
```

   Time Complexity of this recursinon function is O(n).
   Space complexity for function is O(n) [**Call Stack]**

   (ii),

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

   Node in a specific level in a complete binary tree : 2^{n-1} [root level is 1] and Total node in that tree : 2^n - 1. And function will be called at every node.
   SO,
   Time Complexity of this recursion function is O(2^n)
   Space complexity for function is O(n)

4. **Lets Optimize :** By Dynamic Programing (DP)
   In Recursive method of fibonacci series, there are lots of same subtree, which are called and calculate multiple times, and it is unnecessary.

   By saving the repetitive same value in memory can reduce nodec all for fibonacci in recursion method. So it will become n call for n int.
   So, Time complexity will be O(N).

5. **Optimized Fibonacci Recursion Method : Implementation**