Dynamic Programming

The idea is very simple,

- *It is a technique for solving a complex problem by first breaking into a collection of simpler subproblems,*
- *solving each subproblem just once, and then storing their solutions for future reference, to avoid solving the same problem again.*
- *Simply, we need to remember the past.*

Properties of Dynamic Programming Strategy

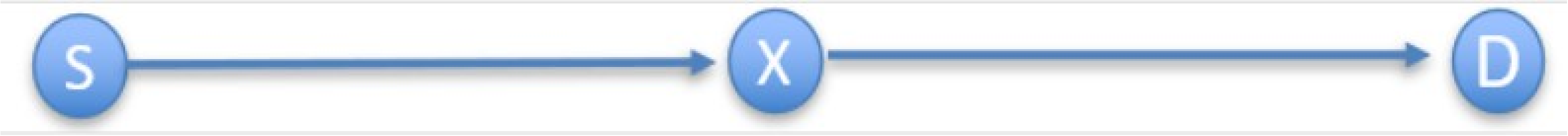
The two main properties of a problem that suggests that the given problem can be solved using Dynamic programming.

1. ***Overlapping Subproblems*** *:*

- *Similar to Divide-and-Conquer approach, Dynamic Programming also combines solutions to subproblems. It is mainly used where the solution of one sub-problem is needed repeatedly. The computed solutions are stored in a table, so that these don't have to be re-computed. Hence, this technique is needed where overlapping sub-problem exists.*
- ***For example****, Binary Search does not have overlapping sub-problem. Whereas recursive program of Fibonacci numbers have many overlapping sub-problems.*

2. Optimal Substructure:

- *A given problem has Optimal Substructure Property, if the optimal solution of the given problem can be obtained using optimal solutions of its sub-problems.*
- ***For example,*** *the Shortest Path problem has the following optimal substructure property − If a node x lies in the shortest path from a source node S to destination node D, then the shortest path from S to D is the combination of the shortest path from S to X, and the shortest path from X to D.*



When to use Dynamic Programming?

*If the given problem can be broken up into smaller sub-problems and these smaller subproblems are in turn divided into still-smaller ones, and in this process, if you observe some **over-lapping** subproblems, then it's a big hint for DP. Also, the optimal solutions to the subproblems contribute to the **optimal solution** of the given problem*

Note: When a problem has two main properties - Overlapping Sub problem and Optimal Substructure

*Basically, there are **two approaches for solving DP problems:***

- ***Top-down approach [Memoization]***
- ***Bottom-up approach [Tabulation]***

Optimization problem *is the problem of finding the best solution from all feasible solutions.*

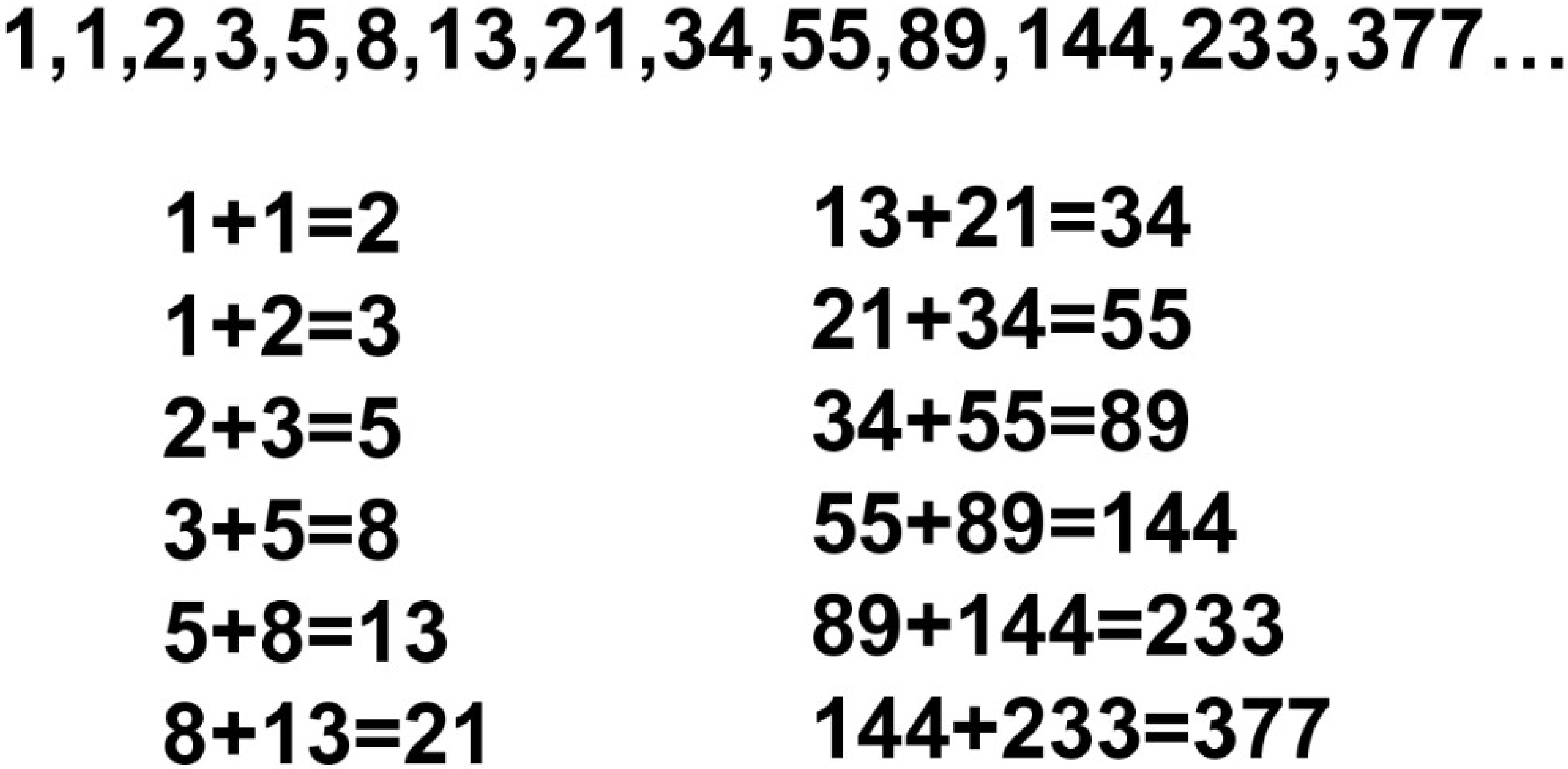
Like finding the Minimal cost, maximal profit, minimal error, minimal time, maximum speed etc.,

For example, companies often want to minimize production costs or maximize revenue. In manufacturing, it is often desirable to minimize the amount of material used to package a product with a certain volume.

Optimal Solution *- best solution from all feasible solutions.*

DP Problem 1 – Fibonacci Series

Fibonacci Series



1,1,2,3,5,8,13,21,34,55,89,144,233,377…

1+1=2

1+2=3

2+3=5

3+5=8

5+8=13

8+13=21

13+21=34

21+34=55

34+55=89

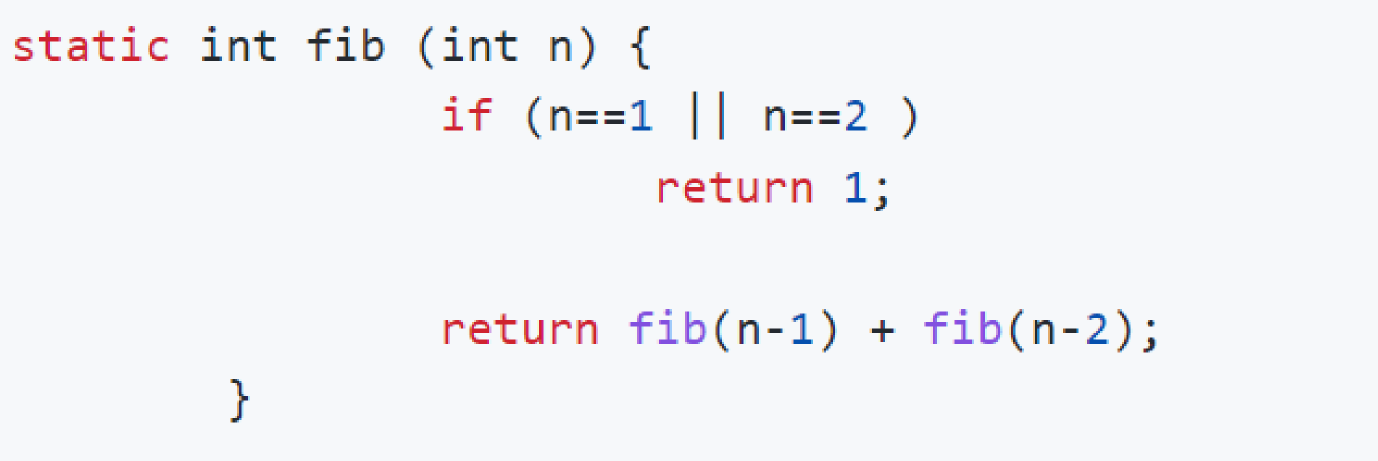
55+89=144

89+144=233

144+233=377

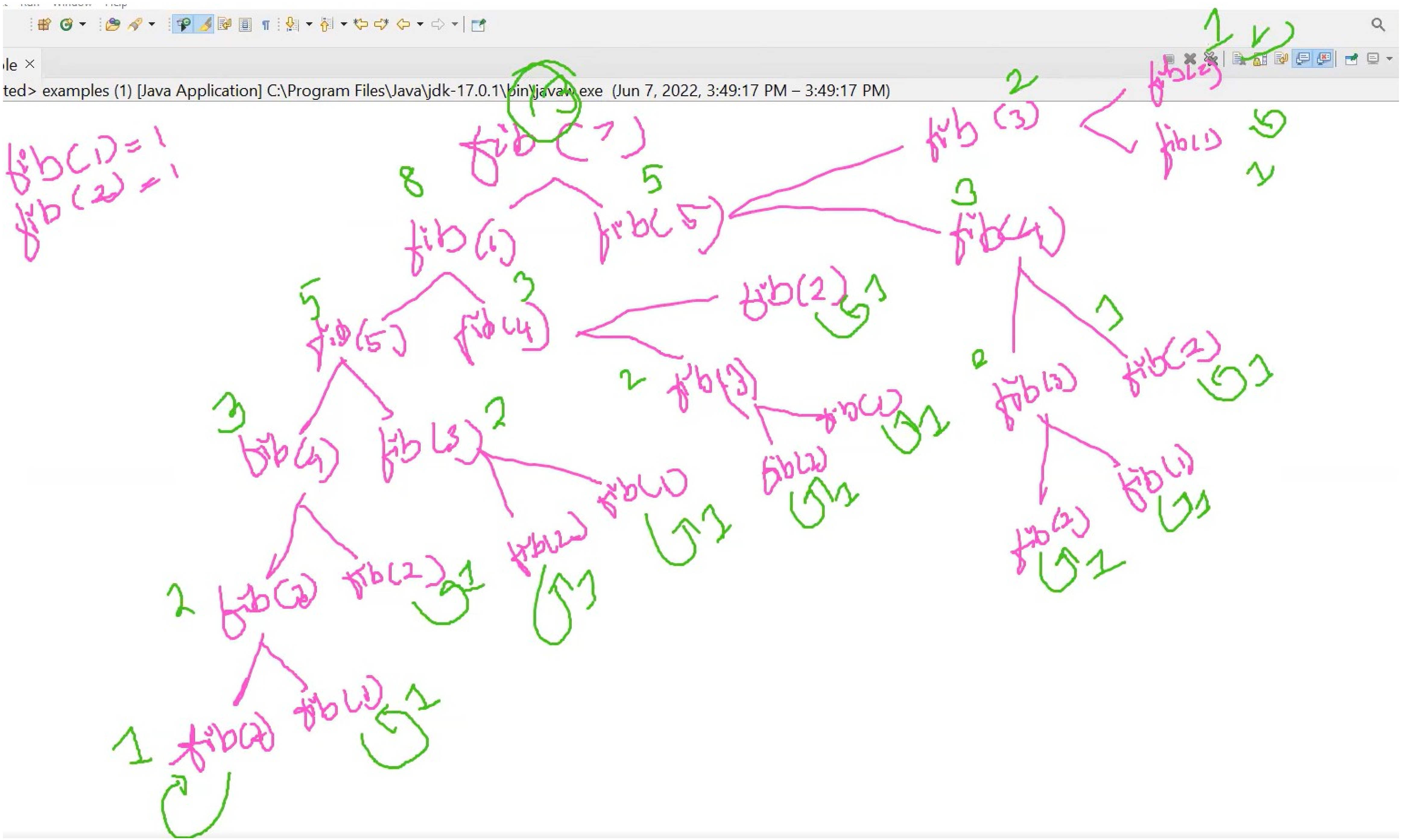
Recall - Fibonacci Series Using Recursion

- *nth term in the Fibonacci series is the addition of previous two terms i.e.,* ***nth term = (n-1)th term + (n-2)th term***
- ***Base condition:*** *first term and second term in the series is 1*
- ***Recursive Code:***



```
static int fib (int n) {
                if (n==1 || n==2 )
                        return 1;

                return fib(n-1) + fib(n-2);
        }
```



```
ted> examples (1) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe  (Jun 7, 2022, 3:49:17 PM – 3:49:17 PM)
```

fib(1) = 1
fib(2) = 1

Remember and Relate

Remember: When to use Dynamic Programming?

If the given problem can be broken up into smaller sub-problems and these smaller subproblems are in turn divided into still-smaller ones, and in this process, if you observe some over-lapping subproblems, then it's a big hint for DP. Also, the optimal solutions to the subproblems contribute to the optimal solution of the given problem

Relate with Fibonacci Series

- ***Is the given problem can be broken into smaller sub problem?***
- ***Is there any overlapping sub problem?***
- ***Can I solve it using DP?***

DP Approach

- ***For a problem, fib (6) we divide into 2 sub problems as fib(4) and fib(5)***
- ***How do we solve each sub problem?***
  - *Check whether we solved that same sub problem previously, if so take the result form the stored array*
  - *If not, apply the logic and find the solution*

Let’s solve Fibonacci using DP with Top down approach

Let’s solve Fibonacci using DP with bottom up approach