Dynamic programming (usually referred to as **DP**) is a very powerful technique to solve a particular class of problems. It demands very elegant formulation of the approach and simple thinking and the coding part is very easy. The idea is very simple, If you have solved a problem with the given input, then save the result for future reference, so as to avoid solving the same problem again.. shortly *'Remember your Past'* :) . If the given problem can be broken up in to smaller sub-problems and these smaller subproblems are in turn divided in to still-smaller ones, and in this process, if you observe some over-lappping subproblems, then its a big hint for DP. Also, the optimal solutions to the subproblems contribute to the optimal solution of the given problem (referred to as the Optimal Substructure Property).

There are two ways of doing this.

1.) Top-Down : Start solving the given problem by breaking it down. If you see that the problem has been solved already, then just return the saved answer. If it has not been solved, solve it and save the answer. This is usually easy to think of and very intuitive. This is referred to as ***Memoization***.

2.) Bottom-Up : Analyze the problem and see the order in which the sub-problems are solved and start solving from the trivial subproblem, up towards the given problem. In this process, it is guaranteed that the subproblems are solved before solving the problem. This is referred to as ***Dynamic Programming***.

Note that divide and conquer is slightly a different technique. In that, we divide the problem in to non-overlapping subproblems and solve them independently, like in mergesort and quick sort.