Dynamic Programming

- Similar to divide-and-conquer, it breaks problems down into smaller problems that are solved recursively.
- In contrast, DP is applicable when the sub-problems are not independent, i.e. when sub-problems share sub-sub-problems. It solves every sub-sub-problem just once and save the results in a table to avoid duplicated computation.

Elements of DP Algorithms

- **Sub-structure:** decompose problem into smaller sub-problems. Express the solution of the original problem in terms of solutions for smaller problems.
- **Table-structure**: Store the answers to the sub-problem in a table, because sub-problem solutions may be used many times.
- **Bottom-up computation:** combine solutions on smaller sub-problems to solve larger sub-problems, and eventually arrive at a solution to the complete problem.

Applicability to Optimization Problems

- Optimal sub-structure (principle of optimality): for the global problem to be solved optimally, each sub-problem should be solved optimally. This is often violated due to sub-problem overlaps. Often by being "less optimal" on one problem, we may make a big savings on another sub-problem.
- Small number of sub-problems: Many NP-hard problems can be formulated as DP problems, but these formulations are not efficient, because the number of sub-problems is exponentially large. Ideally, the number of sub-problems should be at most a polynomial number.

Example of DP:

- Knapsack Problem
- The All-Pairs Shortest Paths Problem- Floyd-Warshall
- Longest common subsequence
- Sum of subset probelm