

Divide and Conquer v/s Dynamic ProgrammingDivide & Conquer

Break the problem into smaller subproblems, solve them independently, then combine results

Results are not reused

May repeat the same calculations multiple times

Divide  $\rightarrow$  Solve  $\rightarrow$  Combine

Eg: Merge sort, Quick sort, Matrix Multiplication

Dynamic Programming

Break the problem into smaller overlapping subproblems, solve each subproblem once, and store results for reuse

Stores results in a table to avoid recomputation

Avoid repeated calculations

Break  $\rightarrow$  Store  $\rightarrow$  Reuse

Eg: Floyd-Warshall Algorithm, Knapsack problem

Backtracking v/s Branch & BoundBacktracking

Explore all possible solutions by building a solution step-by-step, abandoning paths that are infeasible

Branch & Bound

Solve optimization problems by exploring branches but pruning paths that cannot lead to a better solution than the current best

Find all solutions

Checks validity of partial solutions

May stop after first valid solution if only feasibility is required

Eg: N-Queens Problem

Find the optimal solution

Checks both validity and bound value of partial solutions

Continues until optimal solution is guaranteed.

Eg: Travelling Salesman Problem, Job Scheduling