Dynamic Programming

Dynamic Programming

1. **Definition**
    - A technique for solving complex problems by breaking them into overlapping subproblems, solving each only once, and storing results to avoid redundant computation.
2. **Key Requirements**
    - **Optimal substructure**: an optimal solution can be built from optimal solutions to subproblems.
    - **Overlapping subproblems**: the same subproblems recur multiple times (unlike divide and conquer, where subproblems are typically distinct).
3. **Two Main Approaches**
    - **Memoization (top-down)**: recursive solution that caches results as they're computed.
    - **Tabulation (bottom-up)**: iteratively fills a table of subproblem solutions from the smallest up.
4. **Classic Examples**
    - Fibonacci sequence computation, the 0/1 Knapsack Problem, Longest Common Subsequence, and shortest path algorithms like Bellman-Ford.
5. **Why It Matters**
    - Dynamic programming can turn exponential-time brute-force solutions into polynomial-time solutions by eliminating redundant work.