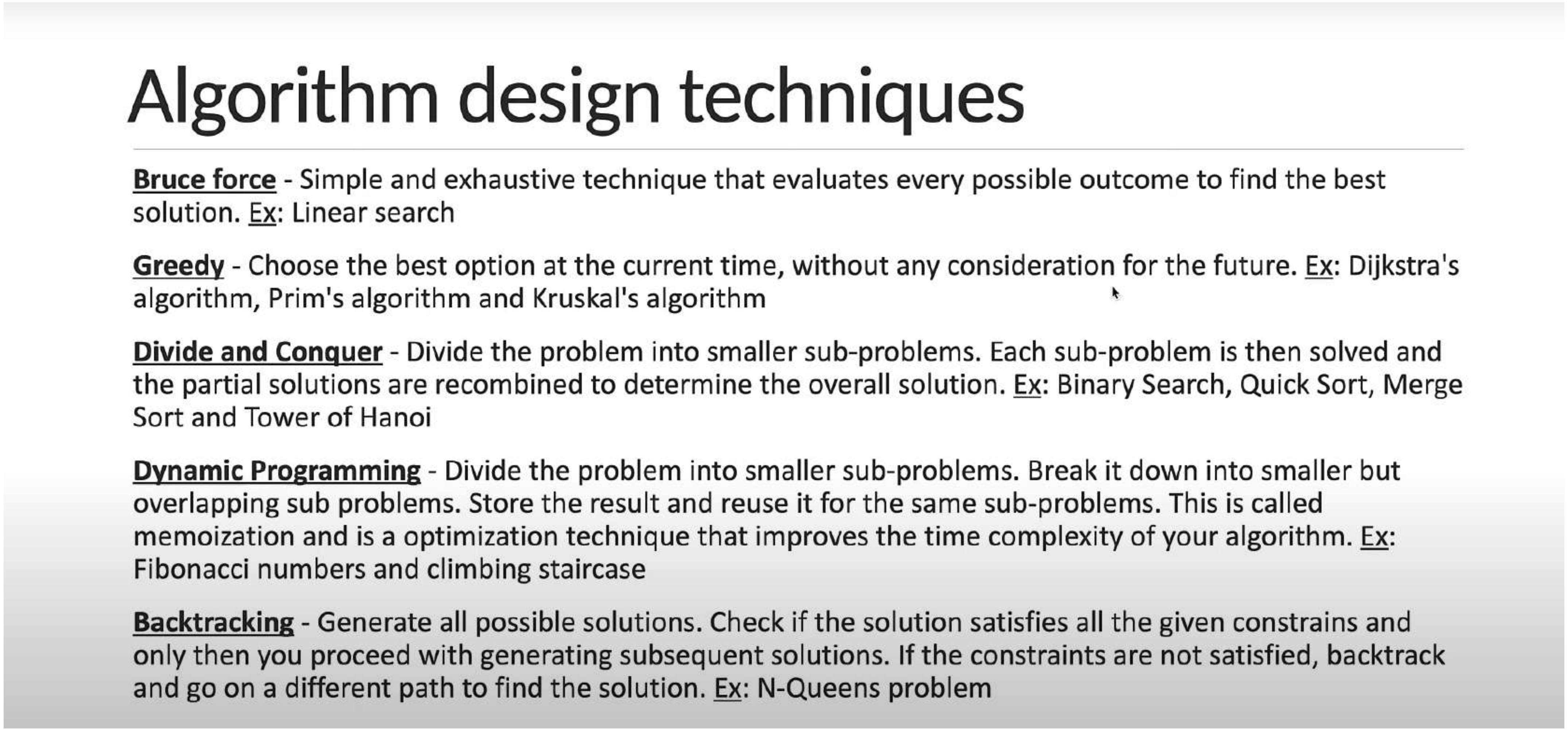


Algorithm design techniques

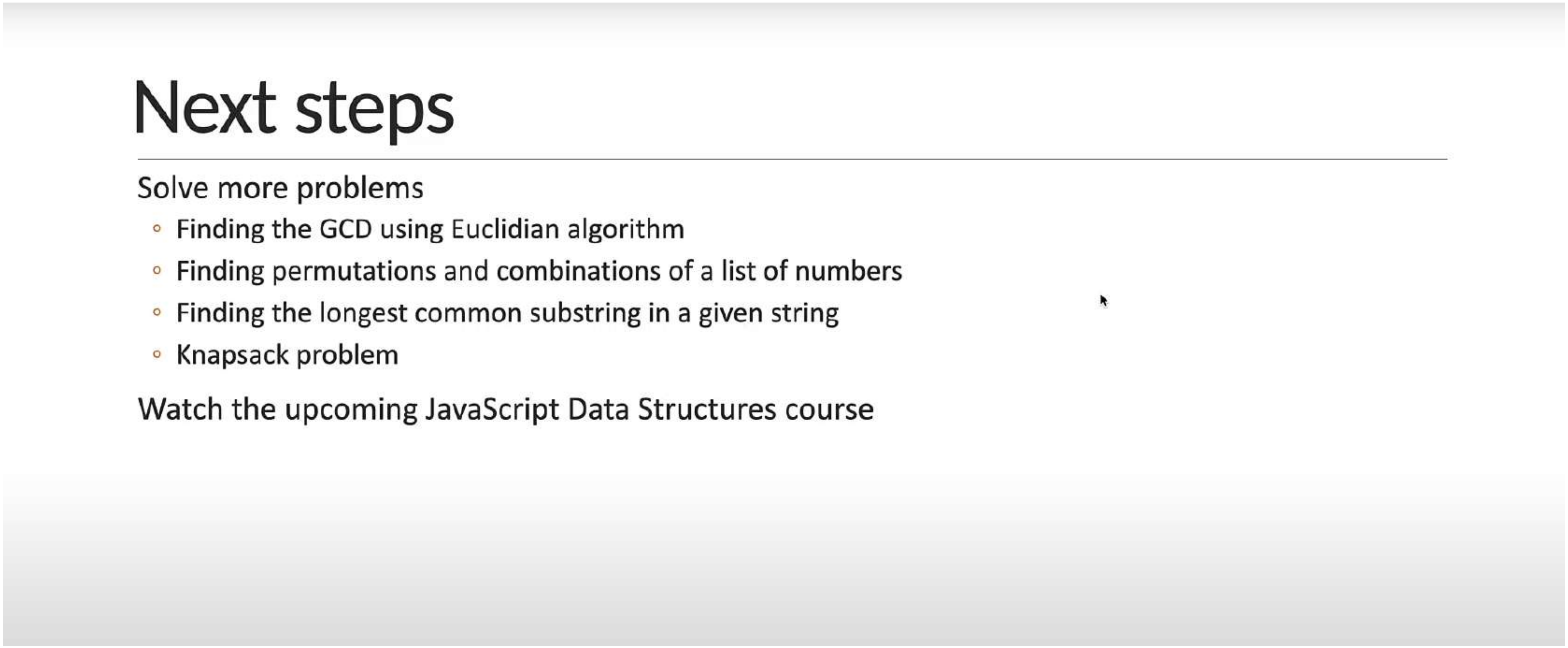
Bruce force - Simple and exhaustive technique that evaluates every possible outcome to find the best solution. Ex: Linear search

Greedy - Choose the best option at the current time, without any consideration for the future. Ex: Dijkstra's algorithm, Prim's algorithm and Kruskal's algorithm

Divide and Conquer - Divide the problem into smaller sub-problems. Each sub-problem is then solved and the partial solutions are recombined to determine the overall solution. Ex: Binary Search, Quick Sort, Merge Sort and Tower of Hanoi

Dynamic Programming - Divide the problem into smaller sub-problems. Break it down into smaller but overlapping sub problems. Store the result and reuse it for the same sub-problems. This is called memoization and is a optimization technique that improves the time complexity of your algorithm. Ex: Fibonacci numbers and climbing staircase

Backtracking - Generate all possible solutions. Check if the solution satisfies all the given constrains and only then you proceed with generating subsequent solutions. If the constraints are not satisfied, backtrack and go on a different path to find the solution. Ex: N-Queens problem



Next steps

Solve more problems

- Finding the GCD using Euclidian algorithm
- Finding permutations and combinations of a list of numbers
- Finding the longest common substring in a given string
- Knapsack problem

Watch the upcoming JavaScript Data Structures course