Greedy algorithms make locally optimal choices at each step in th hope of finding the global optimum. Some classic greedy algorithms and their complexities are:

Kruskal's Algorithm

Example: Finding the minimum spanning tree of a graph. Time Complexity: O(E log E) Space Complexity: O(V) Data Structures Used: Disjoint-set for cycle detection.

Prim's Algorithm

Example: Another way to find a minimum spanning tree. Time Complexity: O(V^2) for dense graphs using adjacency matrix Space Complexity: O(V^2) Data Structures Used: Priority Queue

Dijkstra's Algorithm

Example: Finding the shortest path in a weighted graph. Time Complexity: O(V^2) or O(E + V log V) with a Min-Heap Space Complexity: O(V) Data Structures Used: Priority Queue

Fractional Knapsack

Example: Maximizing profit when items can be divided. Time Complexity: O(n log n) Space Complexity: O(1) Data Structures Used: Array

Algorithm	Time Complexity	Space Complexity	Data Structure Used
Kruskal's Algorithm	O(E log E)	O(V)	Disjoint-set
Prim's Algorithm	O(V^2)	O(V^2)	Priority Queue
Dijkstra's Algorithm	O(V^2) or O(E + V log V)	O(V)	Priority Queue
Fractional Knapsack	O(n log n)	O(1)	Array

Dyanmic programming:

Dynamic programming (DP) is all about breaking a problem into smaller subproblems, solving each just once, and storing their solutions. This way, when the same subproblem comes up again, you don't have to compute it from scratch. You can think of it as a nifty way to optimize recursive algorithms.

Types of Dynamic Programming

Top-Down Approach (Memoization):

This involves writing a recursive function and storing the results of subproblems (in a table or dictionary) to avoid redundant calculations.

Bottom-Up Approach (Tabulation):

Here, you solve subproblems in a systematic way, starting from the smallest subproblems and storing their results in a table until you solve the original problem.

Algorithm	Time Complexity	Space Complexity	Data Structure Used
Fibonacci Sequence	O(n)	O(n) / O(1)	Array / Variables
Knapsack Problem	O(nW)	O(nW)	2D Array
Longest Common Subsequence	O(m*n)	O(m*n)	2D Array
Matrix Chain Multiplication	O(n^3)	O(n^2)	2D Array