Module 8

Dynamic programming (DP) is a powerful algorithmic technique used to solve optimization problems by breaking them into smaller sub-problems. Unlike divide-and-conquer, DP stores the results of sub-problems to avoid redundant computations, making it highly efficient for problems with overlapping sub-problems. Below, we discuss key examples from the uploaded document, provide analogies, and include C++ programs without using vectors.

Fibonacci Number Series

A naive recursive solution has exponential time complexity due to repeated calculations. Using DP, we store intermediate results in an array to reduce the time complexity to O(n).

Knapsack Problem

he knapsack problem involves selecting items with given weights and values to maximize the total value without exceeding the weight capacity. The DP approach builds a table where each entry dp[i][w] represents the maximum value achievable with the first i items and weight w.

Tower of Hanoi

The Tower of Hanoi puzzle involves moving disks between three pegs under specific rules. The DP approach uses recursion to solve the problem in 2n −1 moves.

Floyd-Warshall Algorithm

The Floyd-Warshall algorithm finds the shortest paths between all pairs of vertices in a weighted graph. It uses a dynamic programming table dp[i][j] to store the shortest path between vertices i and j.