Unit-4: Dynamic Programming (DP)

◆ **What is DP?**

- DP is used for **optimization problems**.
- Solves by breaking problem into **overlapping subproblems** and storing results (to avoid recomputation).
- Two approaches:
  - **Top-down (Memoization)** → recursion + cache.
  - **Bottom-up (Tabulation)** → iterative filling of a DP table.

◆ **Principle of Optimality**

- "An optimal solution contains optimal sub-solutions."
- Example: Shortest path from A to C via B = (Shortest A→B) + (Shortest B→C).

◆ **Problems in DP**

1. Binomial Coefficient (nCr)

- Formula: C(n, k) = C(n-1, k-1) + C(n-1, k).
- Base case: C(n, 0) = C(n, n) = 1.

---

2. Make Change Problem (DP approach)

- Instead of greedy, we use DP table: c[n][N], where
  - n = number of denominations,
  - N = amount.
- Gives minimum coins required.

---

3. 0/1 Knapsack Problem

- Each item can either be **taken or not**.
- DP Table V[i][w] = maximum value using first i items with capacity w.
- Formula:
  - If wi ≤ W: V[i][w] = max(V[i-1][w], V[i-1][w-wi] + vi).
  - Else: V[i][w] = V[i-1][w].
- Time Complexity: O(nW).

---

4. All-Pairs Shortest Path (Floyd-Warshall Algorithm)

- Iteratively improve distance matrix.
- Formula: D[i][j] = min(D[i][j], D[i][k] + D[k][j]).
- Time Complexity: $O(n^3)$.

---

5. Matrix Chain Multiplication

- Goal: Find order of multiplying matrices to minimize scalar multiplications.
- DP formula: m[i][j] = min(m[i][k] + m[k+1][j] + pi-1·pk·pj).
- Time Complexity: $O(n^3)$.

---

6. Longest Common Subsequence (LCS)

- Find longest subsequence common to two strings.
- DP Formula:
  - If x[i] = y[j] → L[i][j] = 1 + L[i-1][j-1].
  - Else → L[i][j] = max(L[i-1][j], L[i][j-1]).
- Time Complexity: O(mn).