Unit-5

🔹 Graph Basics

- Graph = Vertices (nodes) + Edges (links).
- **Directed Graph** → edges have direction.
- **Undirected Graph** → edges no direction.

🔹 Graph Traversal

- **DFS** (Depth First Search) → goes deep using stack/recursion. O(V+E).
- **BFS** (Breadth First Search) → level order using queue. O(V+E).
- **Topological Sort** → ordering of nodes in DAG (tasks scheduling).
- **Articulation Point** → removing it disconnects graph.

🔹 Backtracking

- Tries all possibilities, **backtracks** if solution fails.
- Example: **N-Queens Problem** (place N queens without attacking each other).
- Time Complexity: O(N!).

🔹 Branch & Bound

- Improves backtracking using **bounds** to prune bad paths.
- Example: **0/1 Knapsack Problem** → compute upper bound on profit before exploring branch.

🔹 Minimax Algorithm (Game Theory)

- Used in games like Chess, Tic-Tac-Toe.
- One player = MAX (tries to maximize score).
- Other player = MIN (tries to minimize).
- Works by recursively exploring all moves.

🔹 String Matching

1. **Naive** → Check pattern at each shift. O((n-m+1)·m).
2. **Rabin-Karp** → Uses hashing to check substrings. O(n+m) average.

3. **Finite Automata Matching** → Precompute transition table. O(n).
4. **KMP (Knuth-Morris-Pratt)** → Uses prefix function (π table). O(n+m).

🔹 **NP-Completeness**

- **P** → Problems solvable in polynomial time.
- **NP** → Problems verifiable in polynomial time.
- **NP-Complete** → Hardest problems in NP (SAT, TSP). No polynomial-time solution known.
- **NP-Hard** → At least as hard as NP-Complete, not necessarily in NP.

Examples:

- Hamiltonian Cycle (NP-Complete).
- Travelling Salesman Problem (NP-Hard).

✅ **In Simple Words for Viva:**

- **Greedy** → Take best local choice (Kruskal, Prim, Dijkstra, Knapsack, Huffman).
- **DP** → Store & reuse results (Knapsack, Floyd, Matrix Chain, LCS).
- **Graphs** → DFS, BFS, Topo sort, articulation.
- **Backtracking** → Try all + backtrack (N-Queens).
- **Branch & Bound** → Prune bad solutions (Knapsack).
- **String Matching** → Naive, Rabin-Karp, FA, KMP.
- **NP** → P ⊆ NP, NP-Complete hardest, NP-Hard beyond NP.