AKTU · 2ND YEAR · SEMESTER 3 · BCS301

Data Structure

Unit 5 — Graphs

- Terminology used with Graph
- Data Structure for Graph Representations: Adjacency Matrices, Adjacency List
- Graph Traversal: Depth First Search and Breadth First Search
- Connected Component, Spanning Trees
- Minimum Cost Spanning Trees: Prims and Kruskal algorithm
- Transitive Closure and Shortest Path algorithm: Warshal Algorithm and Dijikstra Algorithm



DECODED BY

Chirag Jain

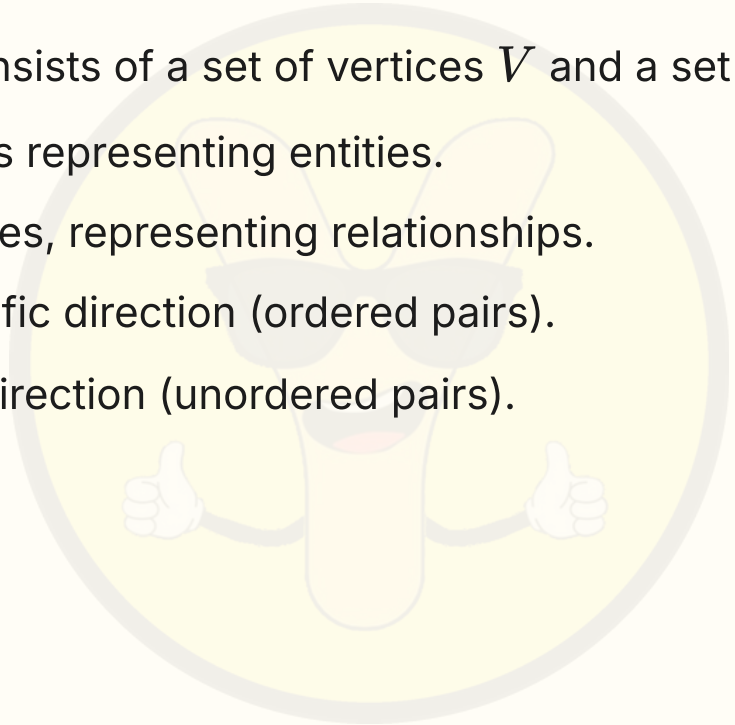
Founder, Ycotes · Gold Medalist in Artificial Intelligence, IIT Madras · taught 10,000+ students

theycotes.com

Introduction to Graphs

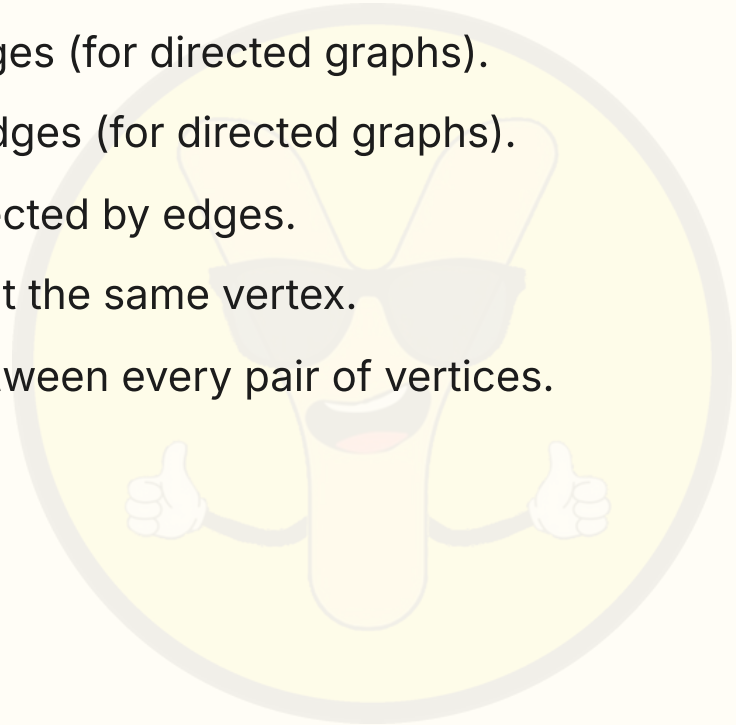
A graph is a non-linear data structure consisting of nodes and edges.

- Definition: A graph $G = (V, E)$ consists of a set of vertices V and a set of edges E.
- Vertices (Nodes): Fundamental units representing entities.
- Edges: Connections between vertices, representing relationships.
- Directed Graph: Edges have a specific direction (ordered pairs).
- Undirected Graph: Edges have no direction (unordered pairs).



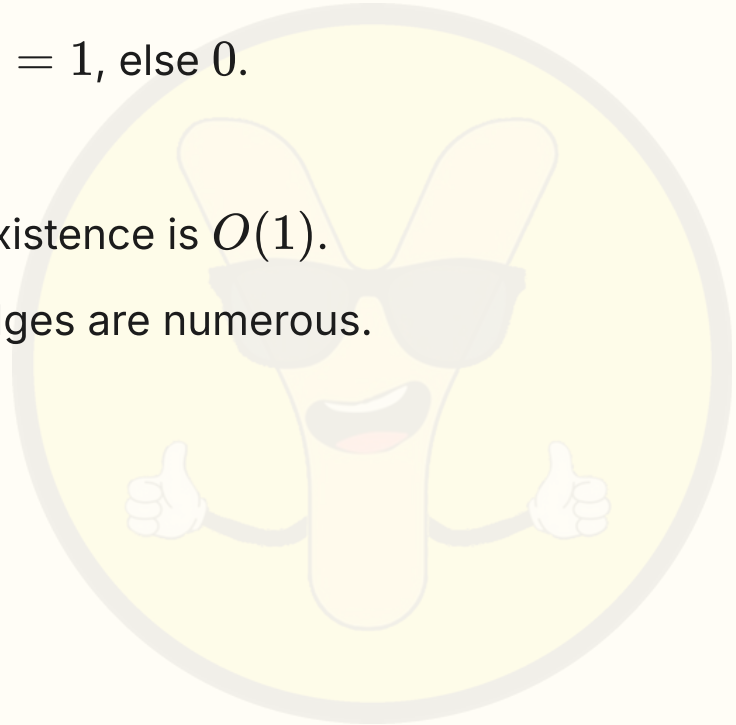
Graph Terminology

- Degree: Number of edges incident to a vertex.
- In-degree: Number of incoming edges (for directed graphs).
- Out-degree: Number of outgoing edges (for directed graphs).
- Path: A sequence of vertices connected by edges.
- Cycle: A path that starts and ends at the same vertex.
- Connected Graph: A path exists between every pair of vertices.



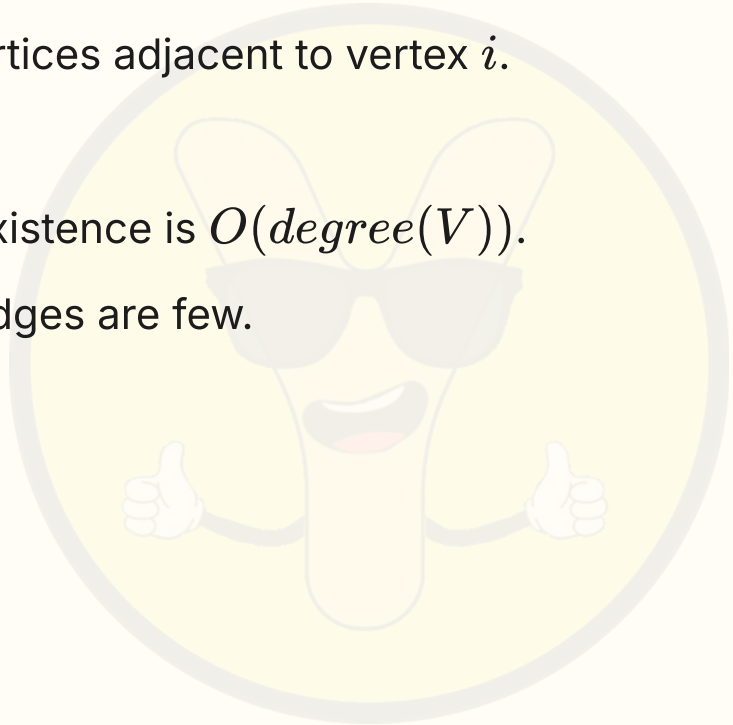
Adjacency Matrix Representation

- A 2D array $A[V][V]$ where V is the number of vertices.
- If an edge exists from i to j, $A[i][j] = 1$, else 0.
- Space Complexity: $O(V^2)$.
- Time Complexity: Checking edge existence is $O(1)$.
- Suitable for dense graphs where edges are numerous.



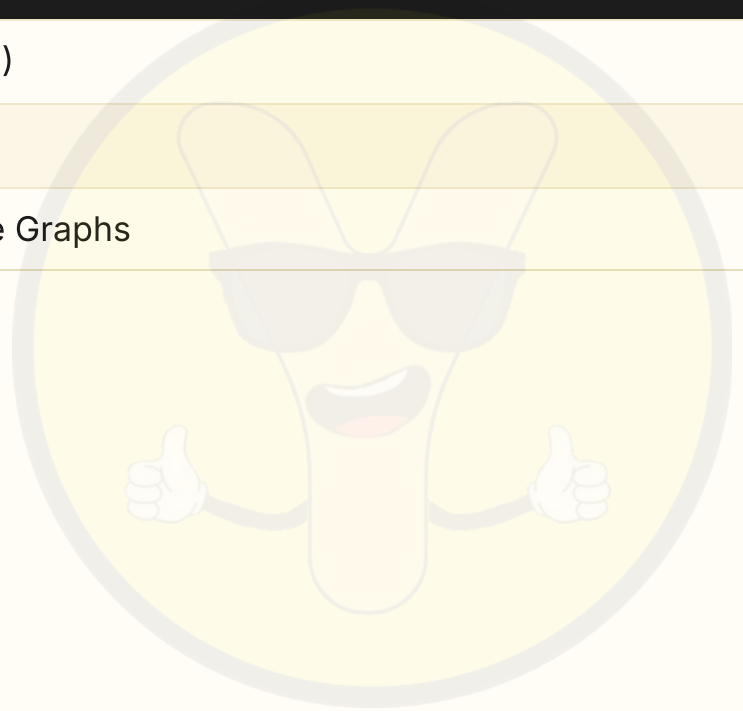
Adjacency List Representation

- An array of linked lists.
- Each index i contains a list of all vertices adjacent to vertex i.
- Space Complexity: $O(V + E)$.
- Time Complexity: Checking edge existence is $O(degree(V))$.
- Suitable for sparse graphs where edges are few.



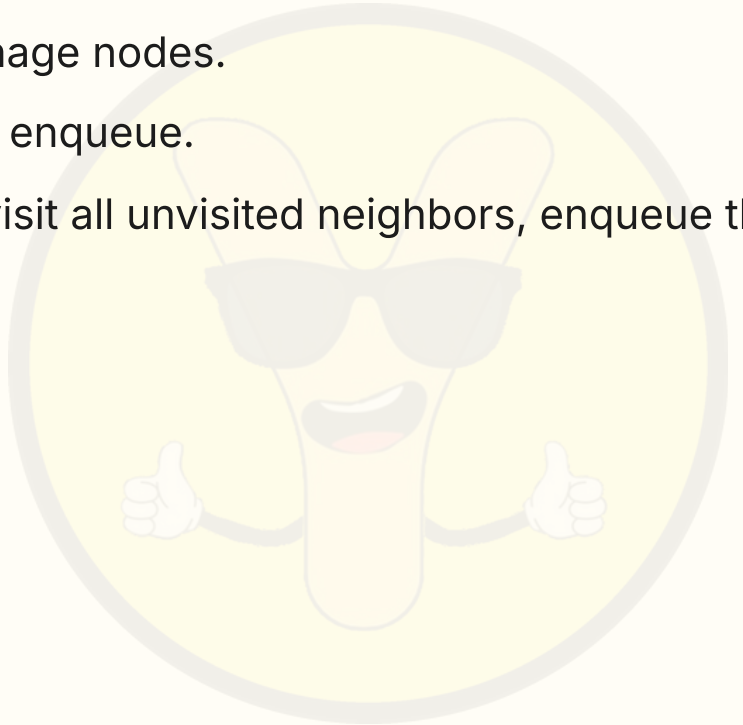
Matrix vs List Comparison

Feature	Adjacency Matrix	Adjacency List
Space	O(V^2)	O(V+E)
Edge Check	O(1)	O(degree)
Best For	Dense Graphs	Sparse Graphs



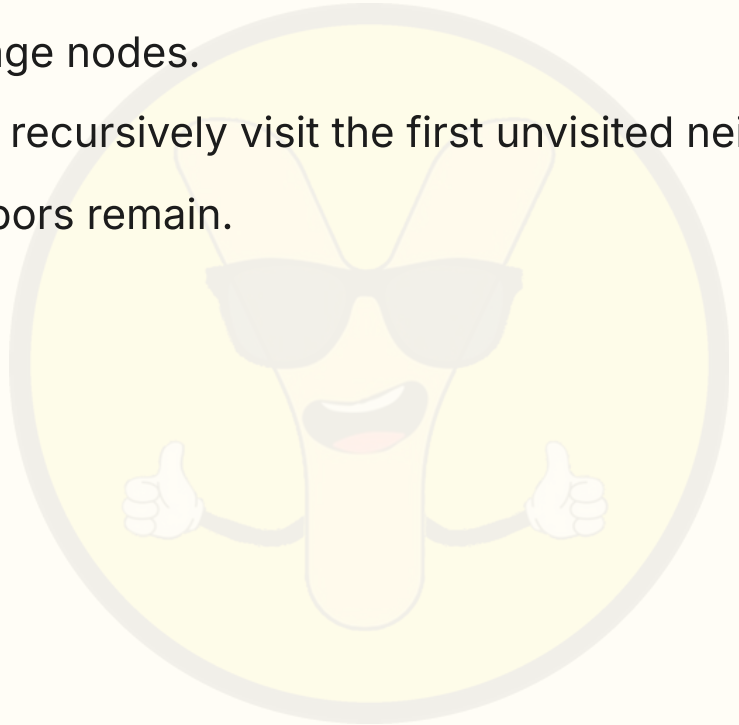
Graph Traversal: BFS

- Breadth First Search explores neighbors level by level.
- Uses a Queue data structure to manage nodes.
- Steps: Start at source, mark visited, enqueue.
- While queue not empty: Dequeue, visit all unvisited neighbors, enqueue them.
- Time Complexity: $O(V + E)$.



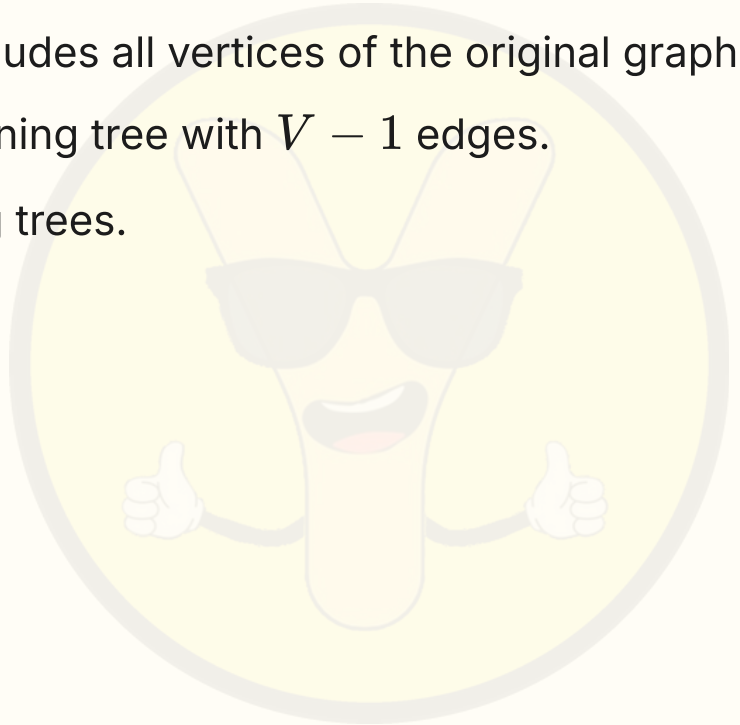
Graph Traversal: DFS

- Depth First Search explores as far as possible along each branch.
- Uses a Stack (or recursion) to manage nodes.
- Steps: Start at source, mark visited, recursively visit the first unvisited neighbor.
- Backtrack when no unvisited neighbors remain.
- Time Complexity: $O(V + E)$.



Connected Components & Spanning Trees

- Connected Component: A maximal subgraph where any two vertices are connected.
- Spanning Tree: A subgraph that includes all vertices of the original graph and is a tree (no cycles).
- A graph with V vertices has a spanning tree with $V - 1$ edges.
- A graph can have multiple spanning trees.



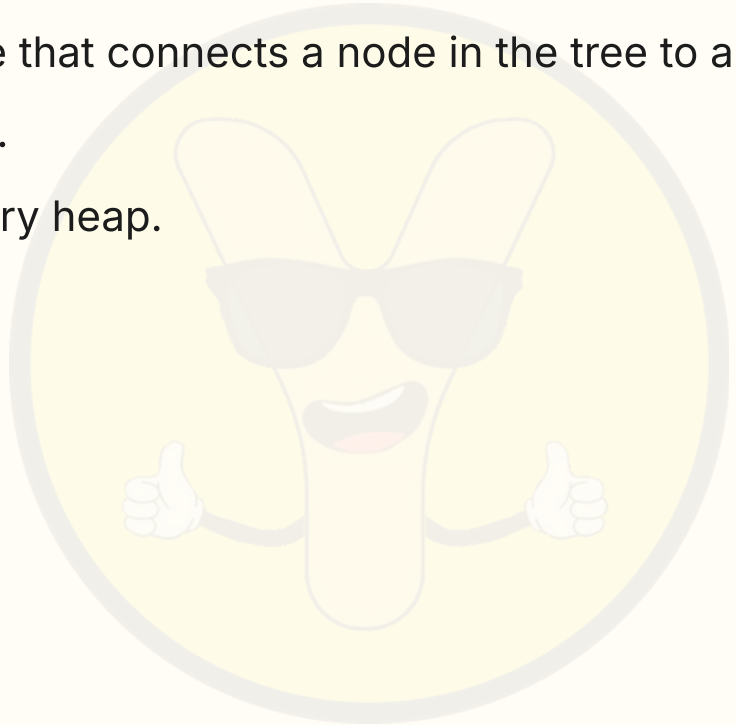
Minimum Spanning Tree (MST)

- A spanning tree where the sum of edge weights is minimized.
- Applications: Network design, circuit layout, road infrastructure.
- Two famous algorithms: Prim's and Kruskal's.
- Both are Greedy algorithms.



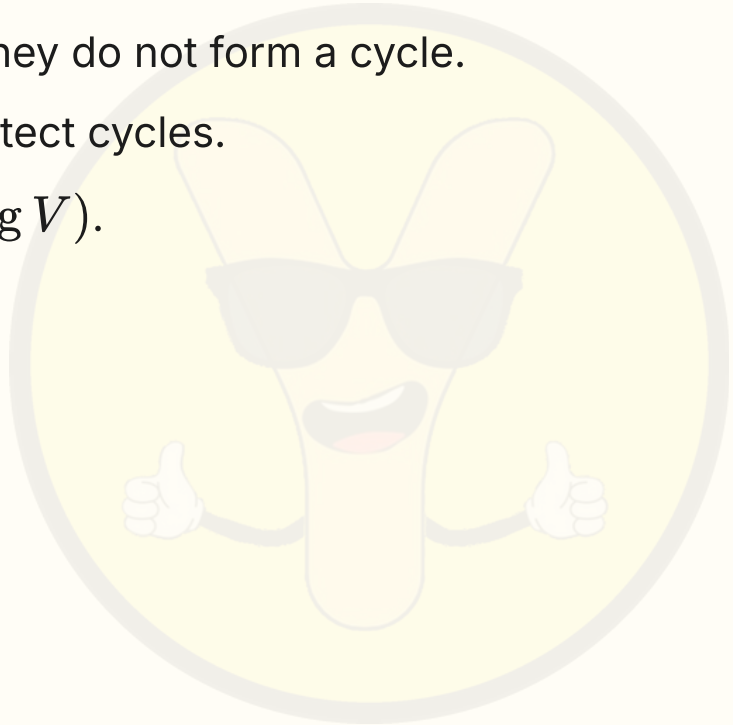
Prim's Algorithm

- Starts with an arbitrary node.
- Repeatedly adds the cheapest edge that connects a node in the tree to a node outside the tree.
- Maintains a priority queue of edges.
- Complexity: $O(E \log V)$ using binary heap.



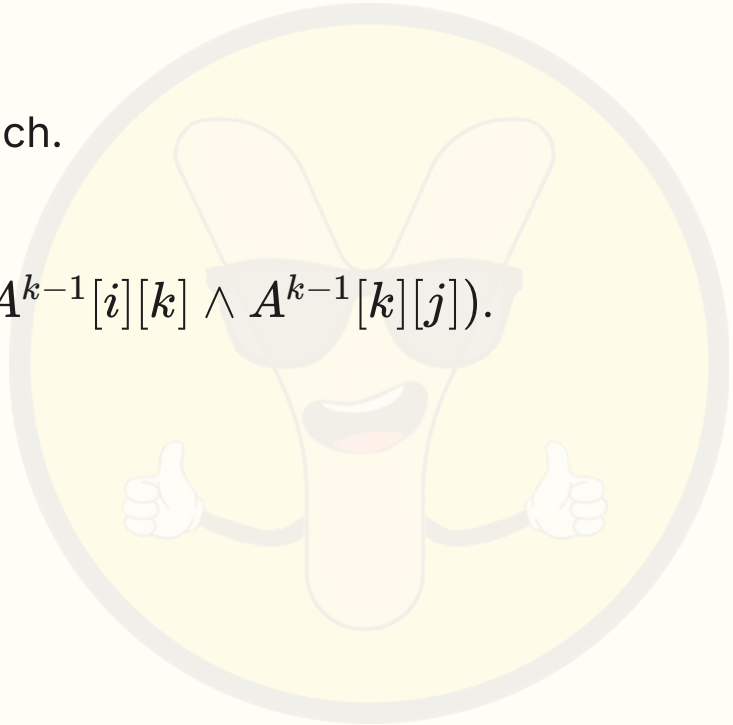
Kruskal's Algorithm

- Sorts all edges in non-decreasing order of weight.
- Adds edges one by one, provided they do not form a cycle.
- Uses Disjoint Set Union (DSU) to detect cycles.
- Complexity: $O(E \log E)$ or $O(E \log V)$.



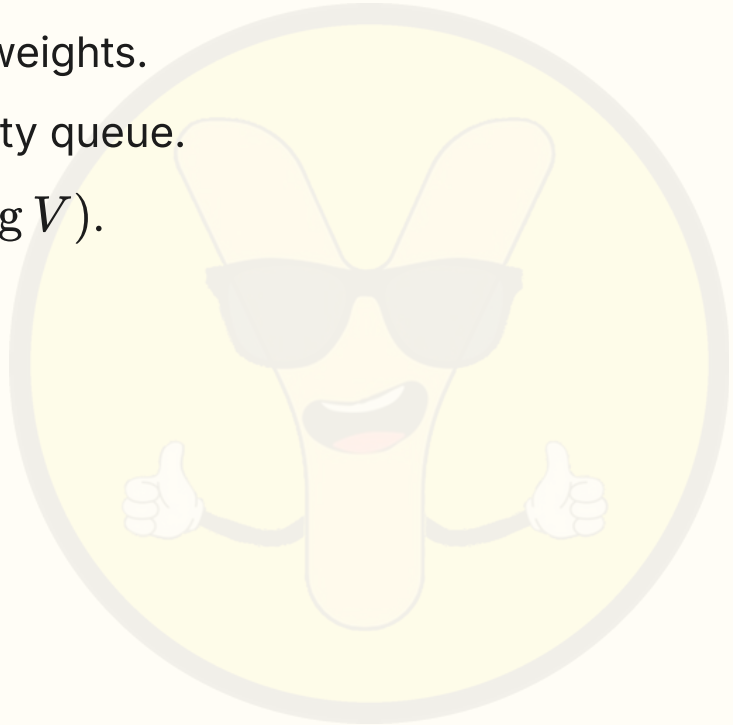
Transitive Closure: Warshall's Algorithm

- Determines if a path exists between any two vertices.
- Result is a reachability matrix.
- Uses dynamic programming approach.
- Complexity: $O(V^3)$.
- Formula: $A^k[i][j] = A^{k-1}[i][j] \vee (A^{k-1}[i][k] \wedge A^{k-1}[k][j])$.



Shortest Path: Dijkstra's Algorithm

- Finds shortest path from a source node to all other nodes.
- Works only for non-negative edge weights.
- Uses a greedy strategy with a priority queue.
- Complexity: $O(V^2)$ or $O(E + V \log V)$.

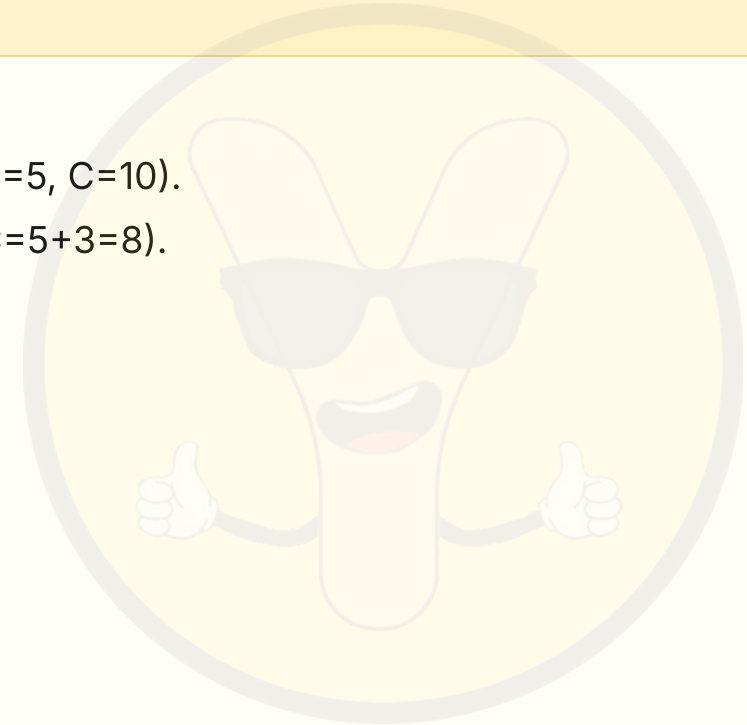


Dijkstra’s Logic

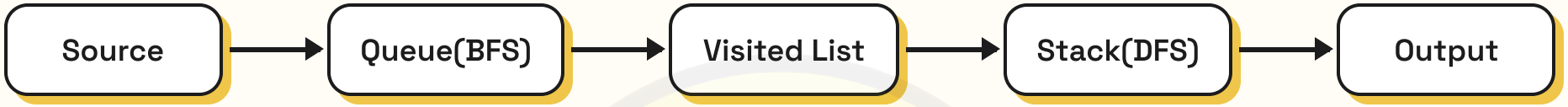
Find shortest path from A to C.

1. Initialize distances: A=0, others=infinity.
2. Extract min node (A), relax neighbors (B=5, C=10).
3. Extract min node (B), relax neighbors (C=5+3=8).
4. Final shortest distance to C is 8.

Shortest path A → B → C = 8



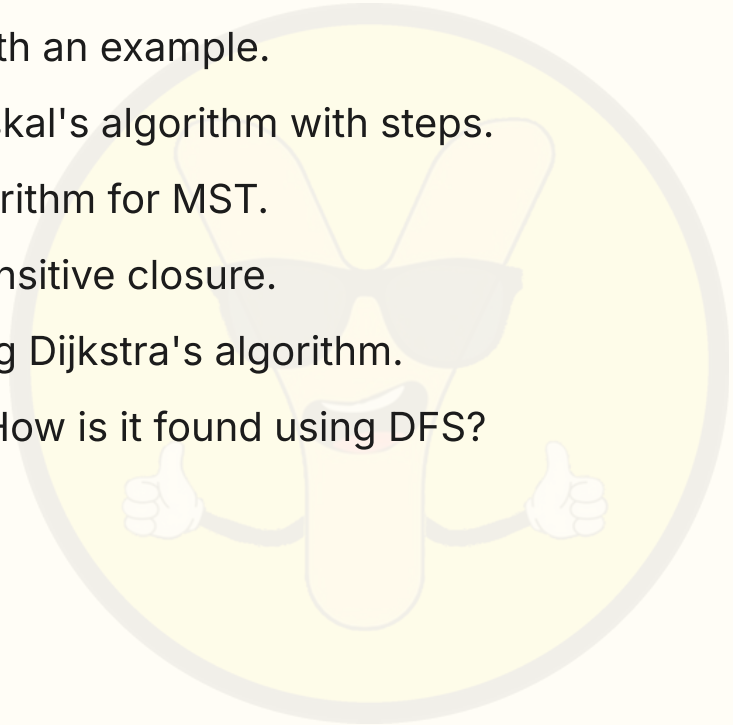
Graph Traversal Comparison



Traversal logic flow

Exam corner

- **Q1** Differentiate between Adjacency Matrix and Adjacency List with space complexity.
- **Q2** Explain BFS and DFS algorithms with an example.
- **Q3** Define Spanning Tree. Explain Kruskal's algorithm with steps.
- **Q4** Compare Prim's and Kruskal's algorithm for MST.
- **Q5** Explain Warshall's algorithm for transitive closure.
- **Q6** Solve a shortest path problem using Dijkstra's algorithm.
- **Q7** What is a connected component? How is it found using DFS?



Unit 5 done. You are exam-ready.

Revise the Exam corner once more, then open Unit 1 on theycotes.com. Every AKTU subject, unit by unit, free. Notes by Chirag Jain, Ycotes.

