GRAPH

in data structures

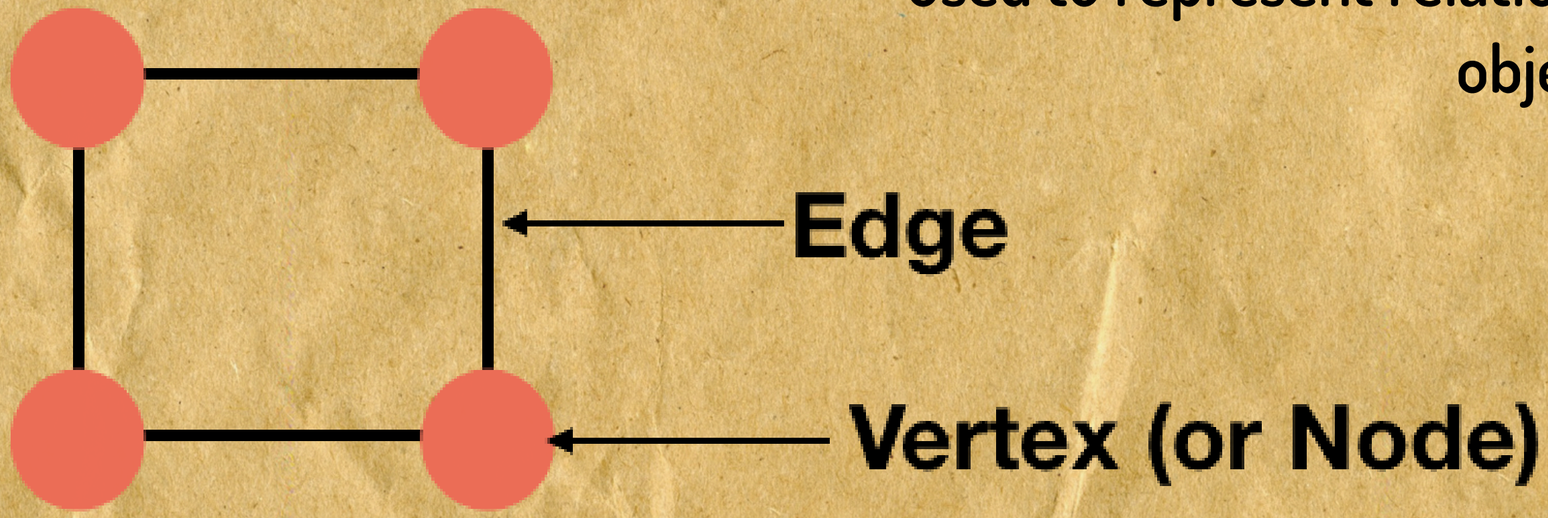
Presented by Brainmentors

content:

- Graph Basics and its Applications
- Directed Graph Undirected Graph
- Weighted Graph Unweighted Graph
- Cyclic Graph Acyclic Graph
- Different Types of Implementations Of graph
- Implementation Of graph using Adjacency List
- DFS and BFS graph
- What is Shortest Path between nodes.
- Spanning Tree And Minimum Spanning Tree

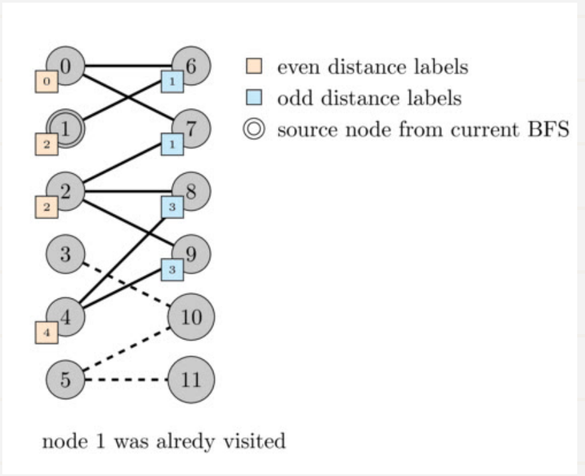
What is graph?

A graph is a collection of nodes (also called vertices) and edges that connect pairs of nodes. Graphs are used to represent relationships between different objects.

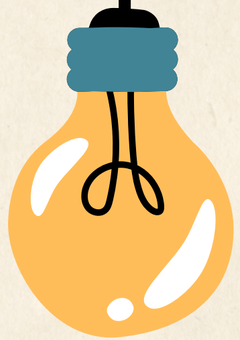


Application of graph

- IoT data analyzed via graphs for patterns, optimization, anomaly detection.
- Graphs identify shortest paths, essential in computer science, logistics, transportation.
- Graphs model complex relationships in machine learning for recommendations, fraud detection.



TYPES OF GRAPH

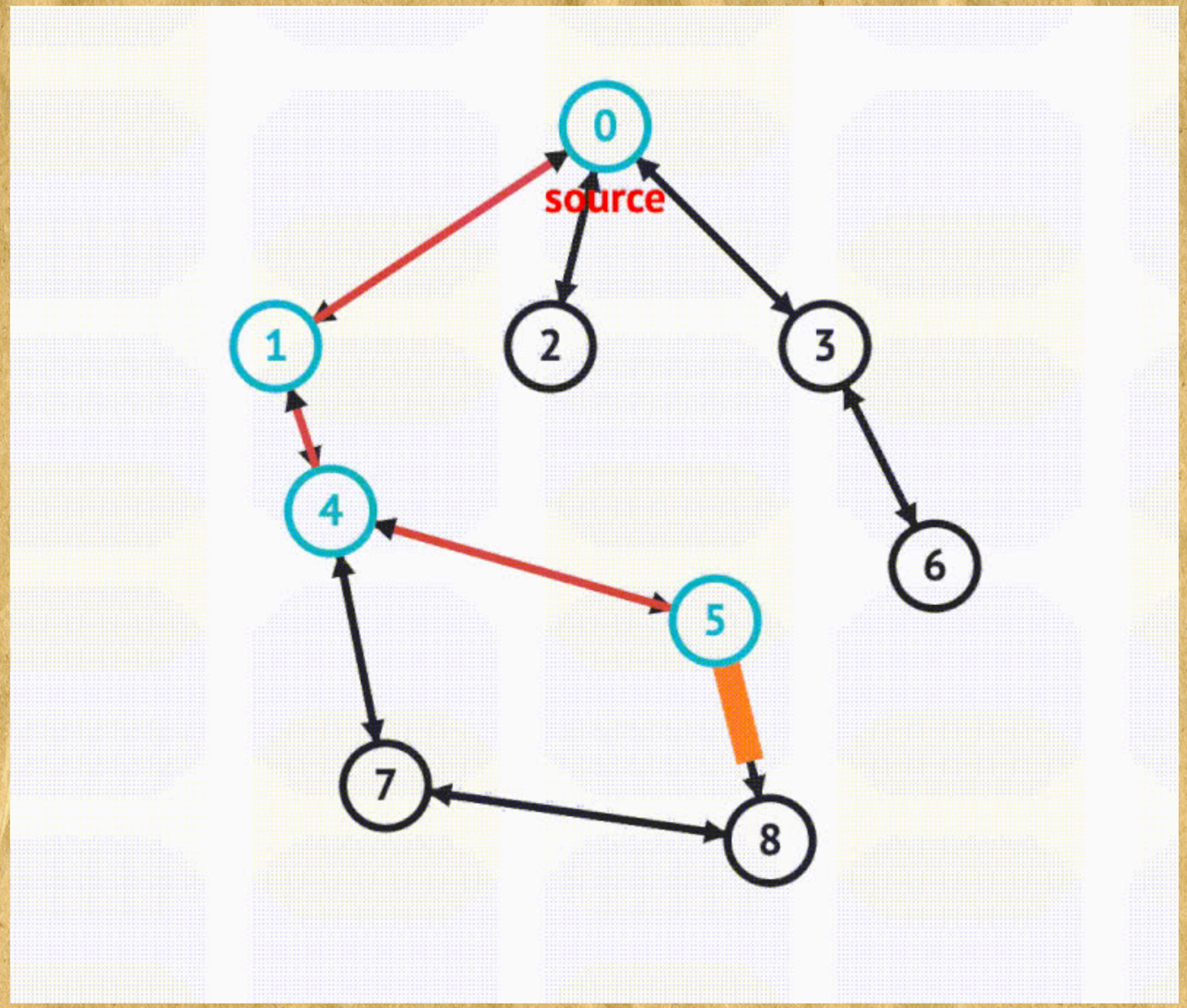


1. DIRECTED
2. UNDIRECTED
3. WEIGHTED
4. UNWEIGHTED
5. CYCLIC
6. ACYCLIC

Directed Graph

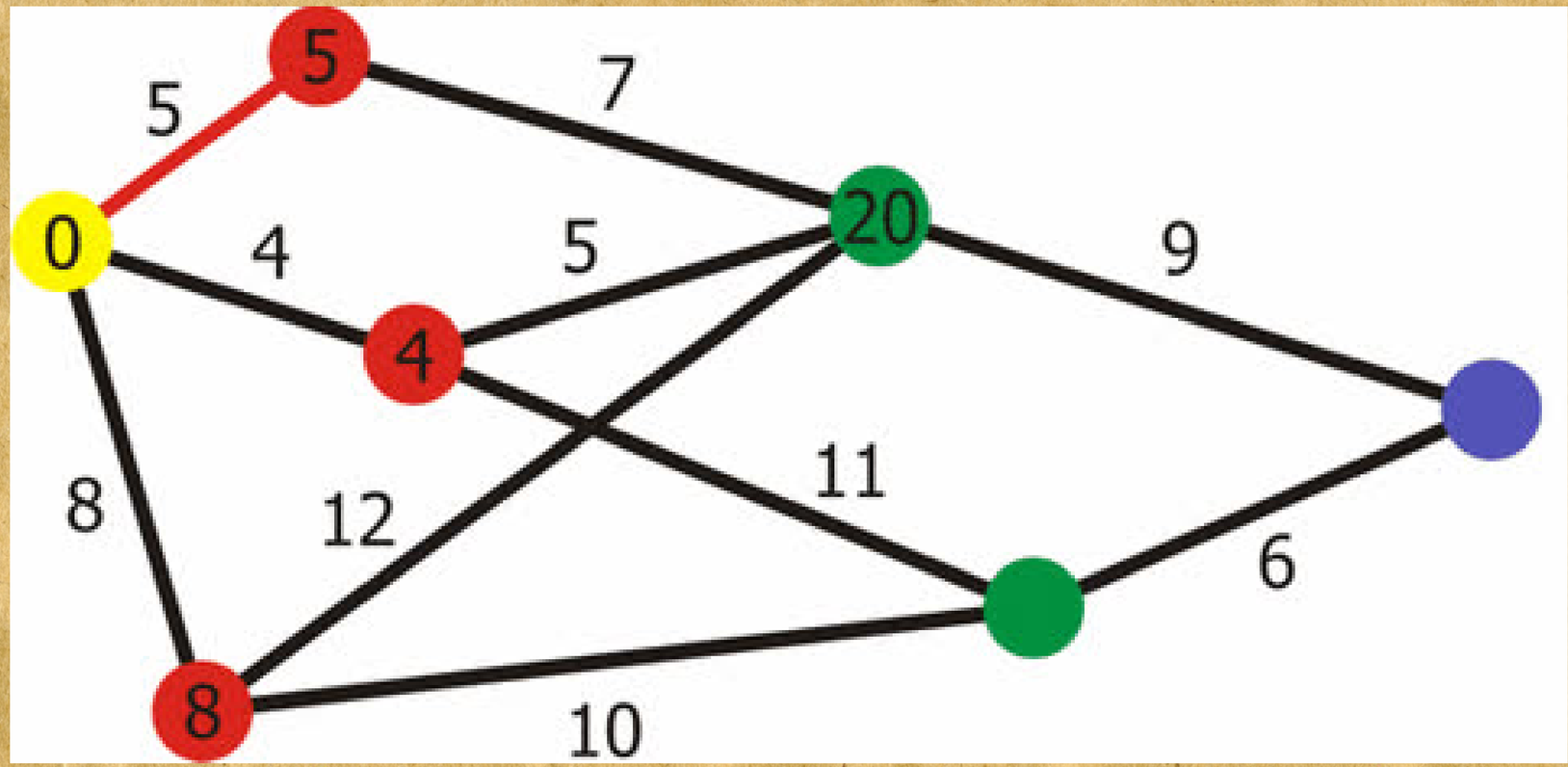
A directed graph (or digraph) is a graph where edges have a direction, indicating a one-way relationship between nodes. Here are some key points:

- Definition: A directed graph G=(V,E)G = (V, E)G=(V,E) consists of a set of vertices VVV and a set of edges EEE where each edge is an ordered pair (u,v)(u, v)(u,v), representing a direction from vertex uuu to vertex vvv.
- Representation: Directed edges are often represented with arrows pointing from the start vertex to the end vertex.



Undirected Graph

An undirected graph is a type of graph where the edges have no direction, indicating a mutual relationship between the nodes they connect.



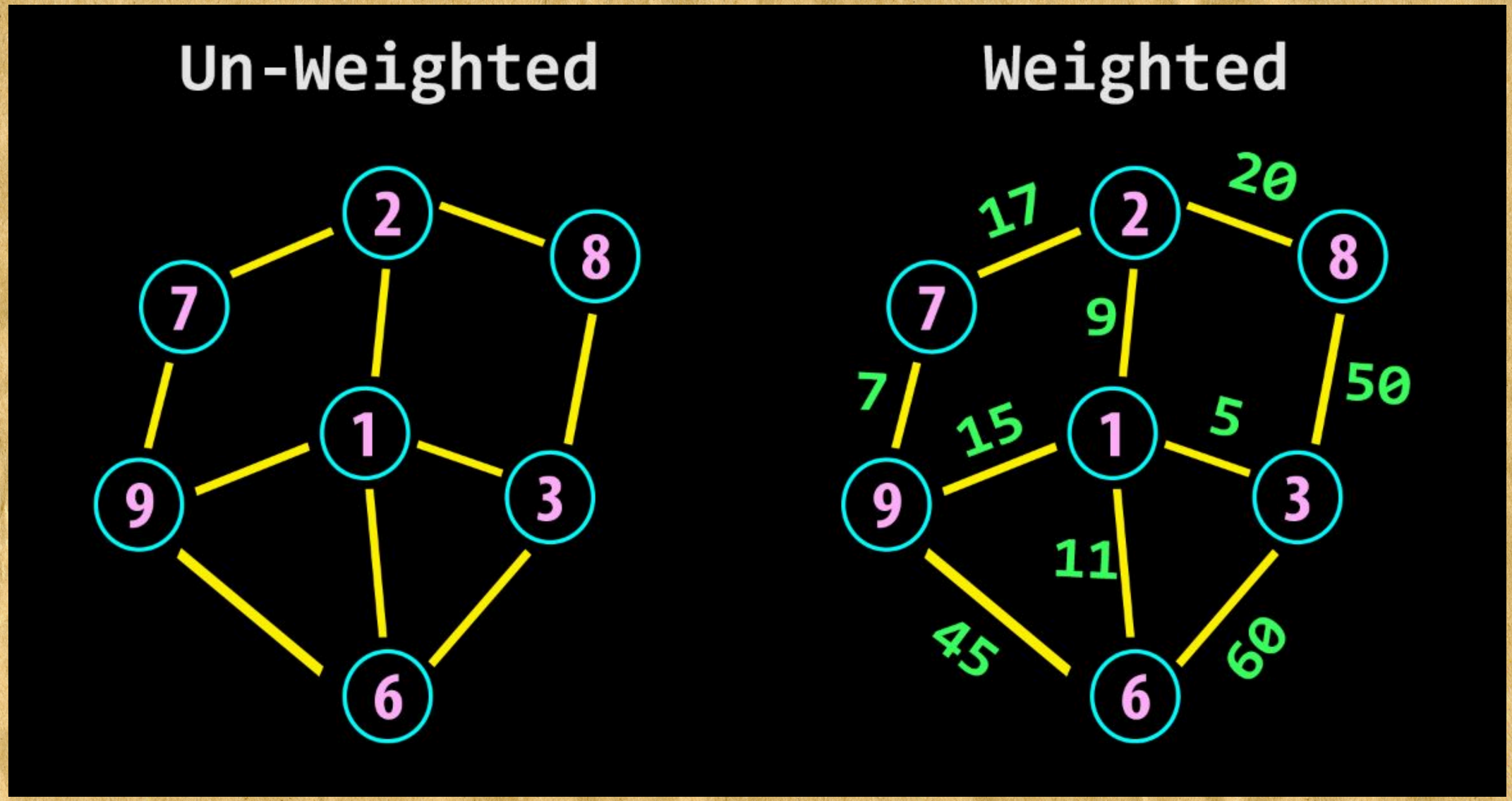
- **Weighted Graph**

A weighted graph is a type of graph in which each edge is assigned a weight or cost. These weights can represent various metrics such as distances, times, costs, or any other quantifiable measure.

- **Unweighted Graph**

An unweighted graph is a type of graph where all edges have the same weight or are considered to have equal weight.

Representation



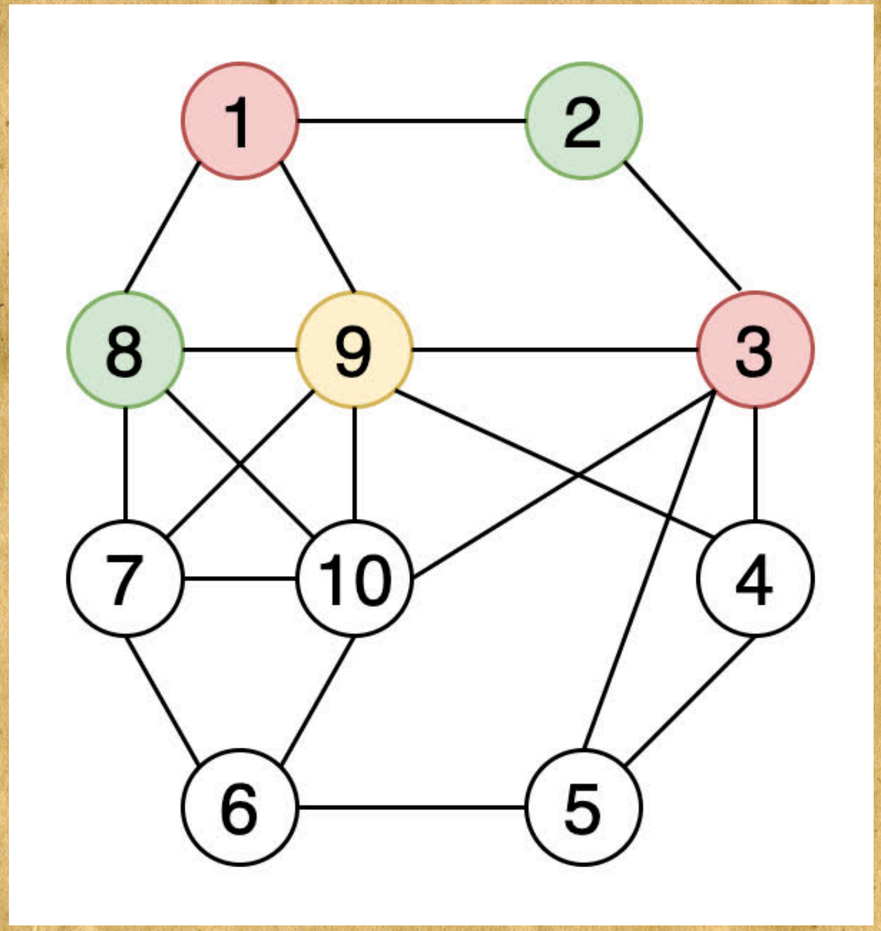
- **Cyclic Graph**

A cyclic graph is a graph that contains at least one cycle. A cycle is a path of edges and vertices wherein a vertex is reachable from itself, with no repetition of vertices or edges except for the starting/ending vertex.

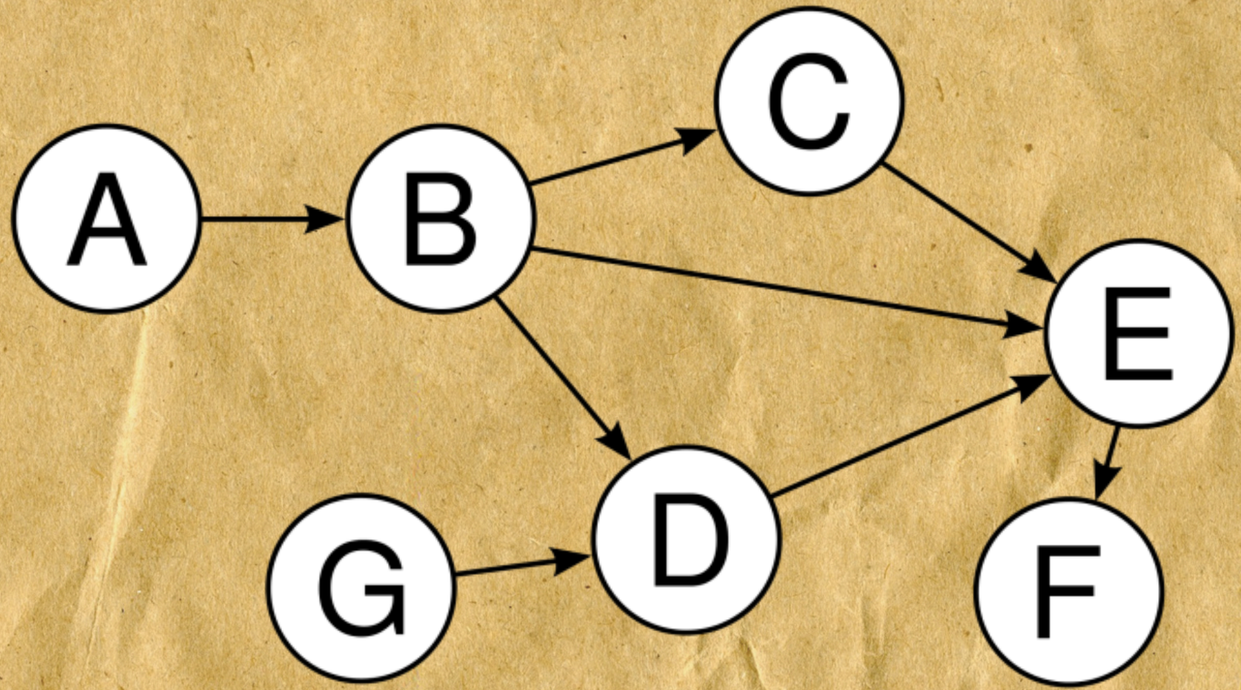
- **Acyclic Graph**

An acyclic graph, also known as a directed acyclic graph (DAG), is a directed graph that does not contain any cycles.

Cyclic Graph



Acyclic Graph



Implementation of Graph

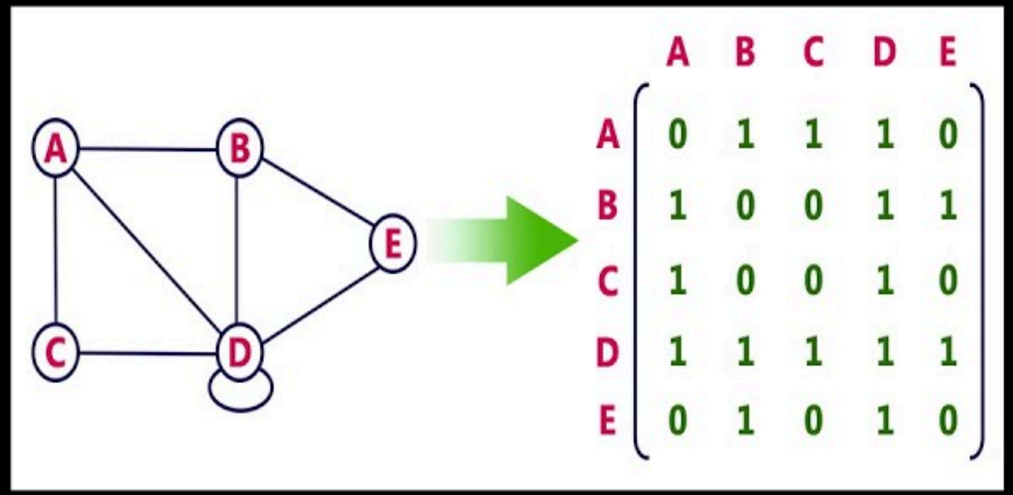
Adjacency List

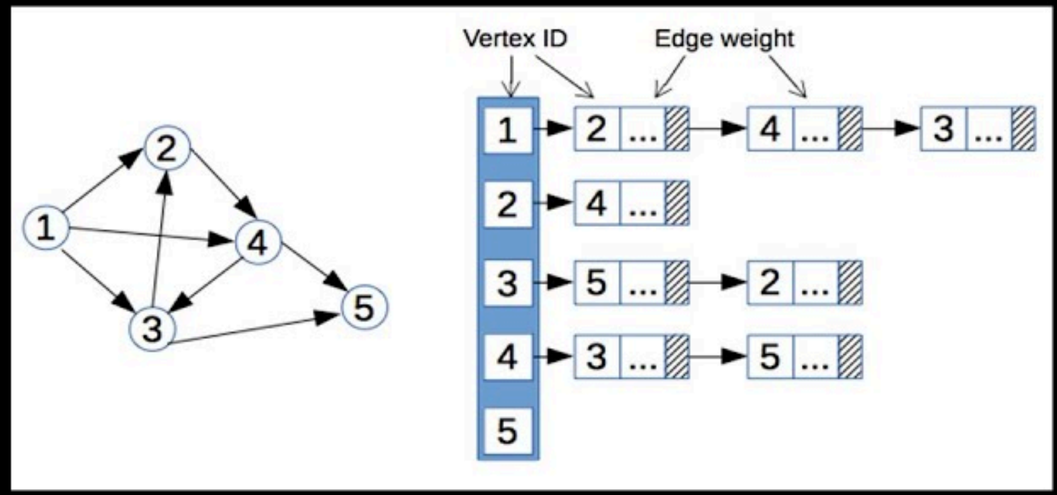
- An adjacency list represents the graph as an array of linked lists.
- Each index in the array represents a vertex, and the linked list at that index contains all adjacent vertices.

Adjacency Matrix

- An adjacency matrix is a 2D array of size $V \times V$V×V where VV is the number of vertices in the graph.
- The matrix element at row ii and column jj is 1 (or a weight value) if there is an edge from vertex ii to vertex jj, and 0 if there is no edge.

Graph Representation





Code of Graph

```java
public static void main(String[] args) {
    int V = 4;
    ArrayList<ArrayList<Integer>> graph = new ArrayList<>();
    for (int i = 0; i <= V; i++) {
        graph.add(new ArrayList<Integer>());
    }

    graph.get(index:1).add(e:2);
    graph.get(index:1).add(e:4);
    graph.get(index:2).add(e:1);
    graph.get(index:2).add(e:3);
    graph.get(index:3).add(e:2);
    graph.get(index:3).add(e:4);
    graph.get(index:4).add(e:1);
    graph.get(index:4).add(e:3);

    for (int i = 0; i <= V; i++) {
        System.out.print(i + " → [ ");
        for (int j = 0; j < graph.get(i).size(); j++) {
            System.out.print(graph.get(i).get(j) + " ");
        }
        System.out.println(x:" ]");
    }
```

DFS (Depth First Search)

Depth-First Search (DFS) is a graph traversal algorithm that explores as far as possible along each branch before backtracking. Here's a basic overview of how DFS works in a graph:

Algorithm:

1. Start with a graph GGG and a starting vertex vvv.
2. Mark the starting vertex vvv as visited.
3. For each neighbor uuu of vvv that has not been visited:
    - Recursively apply DFS starting from uuu.
4. Repeat step 3 for all unvisited neighbors of vvv.
5. When all neighbors of vvv have been visited, backtrack to the previous vertex and repeat step 3 if there are any unvisited neighbors.

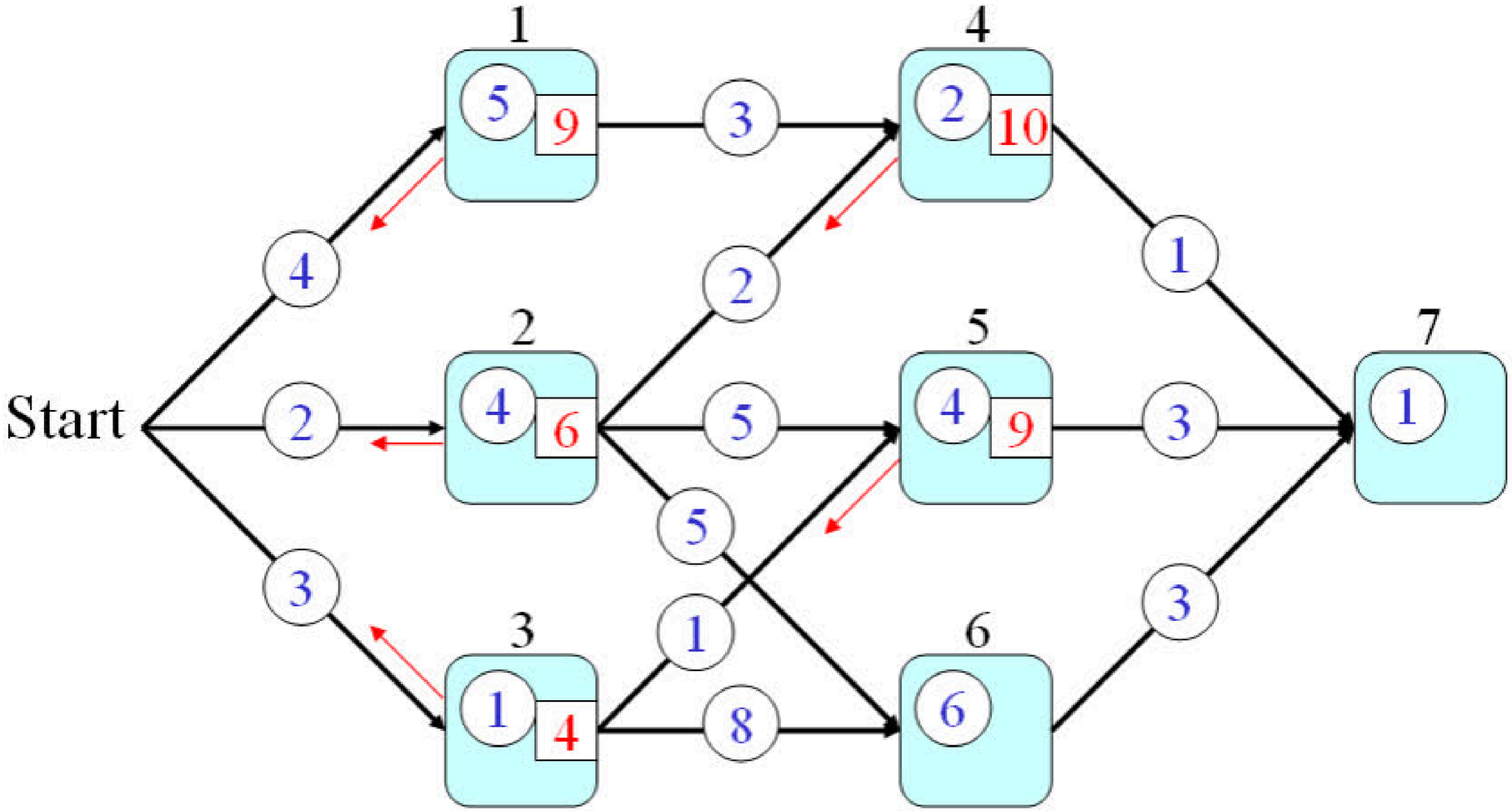
BFS (Breadth First Search)

Breadth-First Search (BFS) is a graph traversal algorithm that explores a graph level by level, starting from a chosen vertex. Here's how BFS works:

Algorithm:

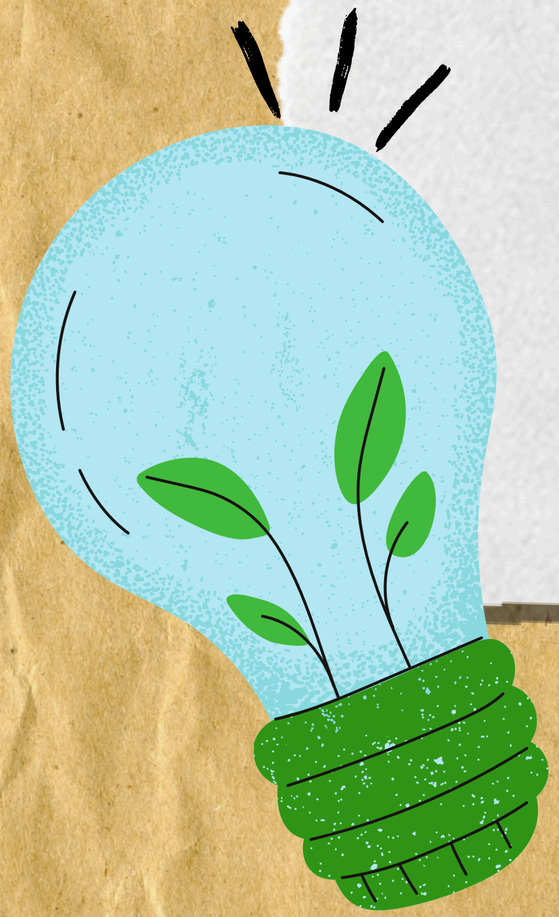
1. Start with a graph G and a starting vertex v.
2. Enqueue the starting vertex v into a queue and mark it as visited.
3. Repeat the following steps until the queue is empty:
    - Dequeue a vertex u from the queue.
    - Process u (e.g., print or store its value).
    - For each neighbor w of u that has not been visited:
        - Enqueue w into the queue and mark it as visited.
4. Continue until all vertices reachable from the starting vertex have been visited.

Finding shortest path




This is the right move.

Thank you!



www.reallygreatsite.com