Data structures

Graphs:

graph data structure consists of a set of nodes (vertices) and a set of edges connecting them. Operations on graphs include adding or removing nodes/edges, finding neighbors of a node, determining connectivity, and traversing the graph.

Major real-world applications of graphs include:

Social Networks: Representing users as nodes and friendships/connections as edges.

Transportation Networks: Modeling roads, railways, and flight routes with nodes as junctions/destinations and edges as connections between them.

Internet and Web Graphs: Representing web pages as nodes and hyperlinks as edges.

Recommendation Systems: Analyzing user-item interactions to suggest products, movies, or music.

Network Routing: Finding the shortest path between two points in a network.

Computer Networks: Modeling routers, switches, and connections between devices.

Bioinformatics: Analyzing genetic interactions, protein-protein interactions, and metabolic pathways.

Circuit Design: Representing components and connections in electronic circuits.

Example of a graph represented in adjacent list is as follows: 0 -> [1, 3]

1 -> [0, 2, 3]

2 -> [1]

3 -> [0, 1]

Operations such as insert and deleting nodes from a graph is shown in the below psuedo code

Function to insert a node into the graph

void insertNode(Graph \*graph, int newNode) {

// Check if the node already exists in the graph

if (!isNodeExists(graph, newNode)) {

// Add the node to the graph's node list

graph->nodes[graph->numNodes++] = newNode;

// Optionally, you might want to initialize the adjacency list for this node

// graph->adjList[newNode] = NULL; // Initialize as empty adjacency list

} else {

printf("Node already exists in the graph.\n");

}

}

Deletion of a Node:

c

// Function to delete a node from the graph

void deleteNode(Graph \*graph, int delNode) {

// Check if the node exists in the graph

if (isNodeExists(graph, delNode)) {

// Remove the node from the graph's node list

for (int i = 0; i < graph->numNodes; i++) {

if (graph->nodes[i] == delNode) {

// Shift elements to fill the gap

for (int j = i; j < graph->numNodes - 1; j++) {

graph->nodes[j] = graph->nodes[j + 1];

}

// Decrement the number of nodes in the graph

graph->numNodes--;

// Remove all edges associated with the deleted node

for (int j = 0; j < graph->numNodes; j++) {

removeEdge(graph, graph->nodes[j], delNode);

}

printf("Node %d has been deleted from the graph.\n", delNode);

return;

}

}

} else {

printf("Node does not exist in the graph.\n");

}

}

There are two ways of graph representation

1: Adjacent Matrix

2:Adjacent list

For Graph traversal we use

1:BFS(breadth first search )

2:DFS(depth first search)