**Aim:** Implement Graph Traversal techniques: a) Depth First Search b) Breadth First Search

1. Depth First Search #include <stdio.h> #include <stdlib.h> #include <stdbool.h>

struct Node {

int data;

struct Node\* next;

};

struct List {

struct Node\* head;

};

struct Graph {

int vertices; struct List\* array;

};

struct Node\* createNode(int data) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node)); newNode->data = data;

newNode->next = NULL; return newNode;

}

struct Graph\* createGraph(int vertices) {

struct Graph\* graph = (struct Graph\*)malloc(sizeof(struct Graph)); graph->vertices = vertices;

graph->array = (struct List\*)malloc(vertices \* sizeof(struct List));

for (int i = 0; i < vertices; i++) { graph->array[i].head = NULL;

}

return graph;

}

void addEdge(struct Graph\* graph, int src, int dest) { struct Node\* newNode = createNode(dest); newNode->next = graph->array[src].head;

graph->array[src].head = newNode;

newNode = createNode(src);

newNode->next = graph->array[dest].head; graph->array[dest].head = newNode;

}

void DFS(struct Graph\* graph, int vertex, bool visited[]) { visited[vertex] = true;

printf("%d ", vertex);

struct Node\* currentNode = graph->array[vertex].head; while (currentNode) {

int adjacentVertex = currentNode->data; if (!visited[adjacentVertex]) { DFS(graph, adjacentVertex, visited);

}

currentNode = currentNode->next;

}

}

specified order

void DFSTraversal(struct Graph\* graph, int\* order, int orderSize) { bool\* visited = (bool\*)malloc(graph->vertices \* sizeof(bool));

for (int i = 0; i < graph->vertices; i++) { visited[i] = false;

}

for (int i = 0; i < orderSize; i++) { if (!visited[order[i]]) { DFS(graph, order[i], visited);

}

}

free(visited);

}

int main() {

int vertices = 4;

struct Graph\* graph = createGraph(vertices); addEdge(graph, 2, 0);

addEdge(graph, 0, 2);

addEdge(graph, 1, 2);

addEdge(graph, 0, 1);

addEdge(graph, 3, 3);

addEdge(graph, 1, 3);

int order[] = {2, 0, 1, 3};

int orderSize = sizeof(order) / sizeof(order[0]); printf("Following is Depth First Traversal (starting from vertex 2):\n");

DFSTraversal(graph, order, orderSize); return 0;

}

1. Breadth First Search #include <stdio.h> #include <stdlib.h> #include <stdbool.h> #define MAX 100

void bfs(int adj[MAX][MAX], int V, int s) {

int q[MAX], front = 0, rear = 0;

bool visited[MAX] = { false };

visited[s] = true;

q[rear++] = s;

printf("BFS Traversal starting from node %d: ", s);

while (front < rear) {

int curr = q[front++]; printf("%d ", curr);

for (int i = 0; i < V; i++) {

if (adj[curr][i] == 1 && !visited[i]) { visited[i] = true;

q[rear++] = i;

}

}

}

printf("\n");

}

void addEdge(int adj[MAX][MAX], int u, int v) { adj[u][v] = 1;

adj[v][u] = 1; // For undirected graph

}

int main() {

int V = 5;

int adj[MAX][MAX] = {0};

addEdge(adj, 0, 1);

addEdge(adj, 0, 2);

addEdge(adj, 1, 3);

addEdge(adj, 1, 4);

addEdge(adj, 2, 4);

bfs(adj, V, 0);

return 0;

}