***NAME : JANHAVI GATTANI***

***BATCH: 2***

***PRN : 12311291***

***ROLL NO : 37***

***LAB ASSIGNMENT 11***

**QUESTION:** **WAP to implement DFS and BFS traversal on Graph using Adjacency Matrix and Adjacency Lists.**

**CODE:**

#include <stdio.h>

#include <stdlib.h>

#define MAX 100

int visited[MAX];

int queue[MAX], front = -1, rear = -1;

void enqueue(int v) {

if (rear == MAX - 1) return;

if (front == -1) front = 0;

queue[++rear] = v;

}

int dequeue() {

if (front == -1 || front > rear) return -1;

return queue[front++];

}

void dfsMatrix(int graph[MAX][MAX], int n, int start) {

visited[start] = 1;

printf("%d ", start);

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

if (graph[start][i] && !visited[i]) {

dfsMatrix(graph, n, i);

}

}

}

void bfsMatrix(int graph[MAX][MAX], int n, int start) {

visited[start] = 1;

enqueue(start);

while (front <= rear) {

int current = dequeue();

printf("%d ", current);

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

if (graph[current][i] && !visited[i]) {

visited[i] = 1;

enqueue(i);

}

}

}

}

struct Node {

int vertex;

struct Node\* next;

};

struct Node\* createNode(int v) {

struct Node\* newNode = malloc(sizeof(struct Node));

newNode->vertex = v;

newNode->next = NULL;

return newNode;

}

void dfsList(struct Node\* adjList[], int vertex, int n) {

visited[vertex] = 1;

printf("%d ", vertex);

struct Node\* temp = adjList[vertex];

while (temp) {

if (!visited[temp->vertex]) {

dfsList(adjList, temp->vertex, n);

}

temp = temp->next;

}

}

void bfsList(struct Node\* adjList[], int start, int n) {

visited[start] = 1;

enqueue(start);

while (front <= rear) {

int current = dequeue();

printf("%d ", current);

struct Node\* temp = adjList[current];

while (temp) {

if (!visited[temp->vertex]) {

visited[temp->vertex] = 1;

enqueue(temp->vertex);

}

temp = temp->next;

}

}

}

int main() {

int choice, n, i, j, start;

printf("Enter number of vertices: ");

scanf("%d", &n);

printf("\nChoose Traversal Type:\n");

printf("1. DFS using Adjacency Matrix\n");

printf("2. BFS using Adjacency Matrix\n");

printf("3. DFS using Adjacency List\n");

printf("4. BFS using Adjacency List\n");

printf("Enter your choice (1-4): ");

scanf("%d", &choice);

if (choice == 1 || choice == 2) {

// Adjacency Matrix input

int graph[MAX][MAX];

printf("Enter adjacency matrix:\n");

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

scanf("%d", &graph[i][j]);

}

}

printf("Enter starting vertex: ");

scanf("%d", &start);

for (i = 0; i < n; i++) visited[i] = 0;

if (choice == 1) {

printf("DFS Traversal: ");

dfsMatrix(graph, n, start);

} else {

printf("BFS Traversal: ");

bfsMatrix(graph, n, start);

}

} else if (choice == 3 || choice == 4) {

struct Node\* adjList[MAX];

for (i = 0; i < n; i++) adjList[i] = NULL;

int edges, src, dest;

printf("Enter number of edges: ");

scanf("%d", &edges);

printf("Enter edges (src dest):\n");

for (i = 0; i < edges; i++) {

scanf("%d %d", &src, &dest);

struct Node\* newNode = createNode(dest);

newNode->next = adjList[src];

adjList[src] = newNode;

// For undirected graph, uncomment this:

// newNode = createNode(src);

// newNode->next = adjList[dest];

// adjList[dest] = newNode;

}

printf("Enter starting vertex: ");

scanf("%d", &start);

for (i = 0; i < n; i++) visited[i] = 0;

front = rear = -1; // reset queue

if (choice == 3) {

printf("DFS Traversal: ");

dfsList(adjList, start, n);

} else {

printf("BFS Traversal: ");

bfsList(adjList, start, n);

}

} else {

printf("Invalid choice!\n");

}

return 0;

}

**OUTPUT:**



