**21.** **Write a C program to Graph traversal using Breadth First Search**

#include <stdio.h>

#include <stdlib.h>

#define MAX\_NODES 100

// Queue implementation for BFS

typedef struct Queue {

int items[MAX\_NODES];

int front;

int rear;

} Queue;

// Graph representation using adjacency matrix

typedef struct Graph {

int numNodes;

int adjMatrix[MAX\_NODES][MAX\_NODES];

} Graph;

// Queue operations

Queue\* createQueue() {

Queue\* q = (Queue\*)malloc(sizeof(Queue));

q->front = -1;

q->rear = -1;

return q;

}

int isEmpty(Queue\* q) {

return q->rear == -1;

}

void enqueue(Queue\* q, int value) {

if (q->rear == MAX\_NODES - 1) {

printf("Queue is full!\n");

} else {

if (q->front == -1)

q->front = 0;

q->rear++;

q->items[q->rear] = value;

}

}

int dequeue(Queue\* q) {

int item;

if (isEmpty(q)) {

printf("Queue is empty!\n");

item = -1;

} else {

item = q->items[q->front];

q->front++;

if (q->front > q->rear) {

q->front = q->rear = -1;

}

}

return item;

}

// Graph operations

Graph\* createGraph(int nodes) {

Graph\* graph = (Graph\*)malloc(sizeof(Graph));

graph->numNodes = nodes;

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

for (int j = 0; j < nodes; j++) {

graph->adjMatrix[i][j] = 0;

}

}

return graph;

}

void addEdge(Graph\* graph, int src, int dest) {

graph->adjMatrix[src][dest] = 1;

graph->adjMatrix[dest][src] = 1; // For undirected graph

}

// BFS traversal

void BFS(Graph\* graph, int startNode) {

int visited[MAX\_NODES] = {0};

Queue\* q = createQueue();

visited[startNode] = 1;

enqueue(q, startNode);

printf("BFS Traversal: ");

while (!isEmpty(q)) {

int currentNode = dequeue(q);

printf("%d ", currentNode);

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

if (graph->adjMatrix[currentNode][i] == 1 && !visited[i]) {

visited[i] = 1;

enqueue(q, i);

}

}

}

printf("\n");

free(q);

}

// Display adjacency matrix

void displayGraph(Graph\* graph) {

printf("\nAdjacency Matrix:\n");

printf(" ");

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

printf("%d ", i);

}

printf("\n");

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

printf("%d: ", i);

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

printf("%d ", graph->adjMatrix[i][j]);

}

printf("\n");

}

}

int main() {

int nodes, edges, choice, startNode;

int src, dest;

printf("Enter number of nodes in the graph: ");

scanf("%d", &nodes);

Graph\* graph = createGraph(nodes);

printf("Enter number of edges: ");

scanf("%d", &edges);

printf("Enter edges (source destination):\n");

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

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

if (src >= nodes || dest >= nodes) {

printf("Invalid edge! Node numbers should be between 0 and %d\n", nodes-1);

i--;

continue;

}

addEdge(graph, src, dest);

}

while (1) {

printf("\nGraph Operations Menu:\n");

printf("1. Perform BFS Traversal\n");

printf("2. Display Adjacency Matrix\n");

printf("3. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter starting node for BFS (0 to %d): ", nodes-1);

scanf("%d", &startNode);

if (startNode < 0 || startNode >= nodes) {

printf("Invalid starting node!\n");

} else {

BFS(graph, startNode);

}

break;

case 2:

displayGraph(graph);

break;

case 3:

free(graph);

printf("Exiting program.\n");

exit(0);

default:

printf("Invalid choice! Please try again.\n");

}

}

return 0;

}