NAME: **Ayush Sanjay Dhangar**

# ROLL NO: 42 CLASS: IT - A BATCH: 02

PROGRAM:

#include <stdio.h> #include <stdlib.h>

struct Node { int data;

struct Node \*next;

};

struct Queue {

struct Node \*front, \*rear;

};

struct Node \*createNode(int data) {

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

newNode->next = NULL; return newNode;

}

struct Queue \*createQueue() {

struct Queue \*queue = (struct Queue \*)malloc(sizeof(struct Queue)); queue->front = queue->rear = NULL;

return queue;

}

void enqueue(struct Queue \*queue, int data) { struct Node \*newNode = createNode(data); if (queue->rear == NULL) {

queue->front = queue->rear = newNode; return;

}

queue->rear->next = newNode; queue->rear = newNode;

}

int dequeue(struct Queue \*queue) { if (queue->front == NULL)

return -1;

struct Node \*temp = queue->front; int data = temp->data;

queue->front = queue->front->next; if (queue->front == NULL)

queue->rear = NULL; free(temp);

return data;

}

int isEmpty(struct Queue \*queue) { return queue->front == NULL;

}

void BFS(int N, int Adj[][N + 1], int start) {

int \*visited = (int \*)malloc((N + 1) \* sizeof(int)); for (int i = 1; i <= N; ++i)

visited[i] = 0;

struct Queue \*queue = createQueue(); visited[start] = 1;

printf("BFS traversal starting from vertex %d: ", start); enqueue(queue, start);

while (!isEmpty(queue)) {

int vertex = dequeue(queue);

printf("%d ", vertex);

for (int i = 1; i <= N; ++i) {

if (Adj[vertex][i] && !visited[i]) { visited[i] = 1;

enqueue(queue, i);

}

}

}

free(visited);

}

void DFSUtil(int N, int Adj[][N + 1], int vertex, int \*visited) { visited[vertex] = 1;

printf("%d ", vertex);

for (int i = 1; i <= N; ++i) {

if (Adj[vertex][i] && !visited[i]) { DFSUtil(N, Adj, i, visited);

}

}

}

void DFS(int N, int Adj[][N + 1], int start) {

int \*visited = (int \*)malloc((N + 1) \* sizeof(int)); for (int i = 1; i <= N; ++i)

visited[i] = 0;

printf("DFS traversal starting from vertex %d: ", start); DFSUtil(N, Adj, start, visited);

free(visited);

}

int main() { int N, M;

printf("Enter the number of vertices: "); scanf("%d", &N);

printf("Enter the number of edges: "); scanf("%d", &M);

int Adj[N + 1][N + 1];

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

for (int j = 0; j <= N; j++) { Adj[i][j] = 0;

}

}

printf("Enter the edges (source destination):\n"); for (int i = 0; i < M; i++) {

int x, y;

scanf("%d %d", &x, &y);

Adj[x][y] = 1;

Adj[y][x] = 1;

}

printf("\nGraph representation using adjacency matrix\n"); for (int i = 1; i <= N; i++) {

for (int j = 1; j <= N; j++) { printf("%d ", Adj[i][j]);

}

printf("\n");

}

int choice; int start;

while (1) {

printf("\nChoose traversal type:\n"); printf("1. BFS traversal\n"); printf("2. DFS traversal\n"); printf("3. Exit\n");

printf("Enter your choice: "); scanf("%d", &choice);

if (choice == 3) break;

printf("Enter the starting vertex: "); scanf("%d", &start);

switch (choice) { case 1:

BFS(N, Adj, start); break;

case 2:

DFS(N, Adj, start); break;

default:

printf("Invalid choice\n"); break;

}

}

return 0;

}

OUTPUT:

Enter the number of vertices: 5 Enter the number of edges: 6

Enter the edges (source destination):

|  |  |
| --- | --- |
| 1 | 3 |
| 1 | 2 |
| 1 | 4 |
| 2 | 4 |
| 3 | 5 |
| 4 | 5 |

Graph representation using adjacency matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |

Choose traversal type:

1. BFS traversal
2. DFS traversal
3. Exit

Enter your choice: 1

Enter the starting vertex: 1

BFS traversal starting from vertex 1: 1 2 3 4 5 Choose traversal type:

1. BFS traversal
2. DFS traversal
3. Exit

Enter your choice: 2

Enter the starting vertex: 1

DFS traversal starting from vertex 1: 1 2 4 5 3 Choose traversal type:

1. BFS traversal
2. DFS traversal
3. Exit

Enter your choice: 3