

LAB – 9

PROGRAM 1 : Write a program to traverse a graph using BFS method

SOURCE CODE:

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

#define MAX_VERTICES 100

typedef struct {
    int matrix[MAX_VERTICES][MAX_VERTICES];
    int numVertices;
} Graph;

Graph* createGraph(int numVertices) {
    Graph* graph = (Graph*)malloc(sizeof(Graph));
    graph->numVertices = numVertices;

    for (int i = 0; i < numVertices; i++) {
        for (int j = 0; j < numVertices; j++) {
            graph->matrix[i][j] = 0;
        }
    }

    return graph;
}

void addEdge(Graph* graph, int src, int dest) {
    graph->matrix[src][dest] = 1;
}

void bfs(Graph* graph, int startVertex) {
    int visited[MAX_VERTICES] = {0};
    int queue[MAX_VERTICES];
    int front = 0, rear = 0;

    queue[rear++] = startVertex;
    visited[startVertex] = 1;

    printf("BFS traversal starting from vertex %d: ", startVertex);
    while (front < rear) {
        int currentVertex = queue[front++]; // Dequeue a vertex
        printf("%d ", currentVertex);

        for (int i = 0; i < graph->numVertices; i++) {
            if (graph->matrix[currentVertex][i] && !visited[i]) {
```

```

        queue[rear++] = i;
        visited[i] = 1;
    }
}
}
printf("\n");
}

int main() {
    int numVertices, numEdges, src, dest, startVertex;

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

    Graph* graph = createGraph(numVertices);

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

    printf("Enter the edges (source destination):\n");
    for (int i = 0; i < numEdges; i++) {
        scanf("%d %d", &src, &dest);
        addEdge(graph, src, dest);
    }

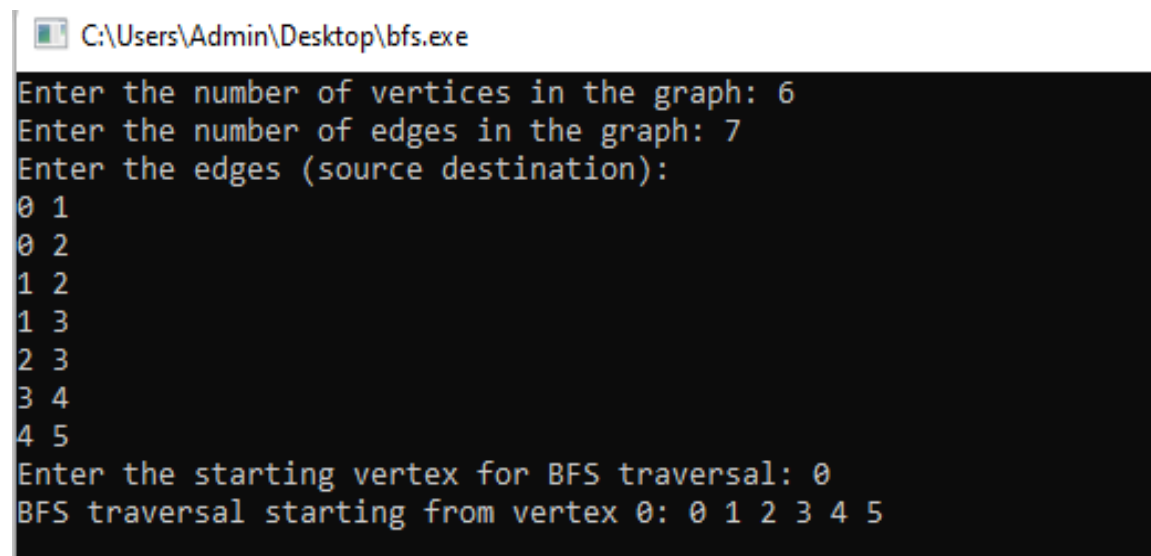
    printf("Enter the starting vertex for BFS traversal: ");
    scanf("%d", &startVertex);

    bfs(graph, startVertex);

    return 0;
}

```

OUTPUT:



```

C:\Users\Admin\Desktop\bfs.exe
Enter the number of vertices in the graph: 6
Enter the number of edges in the graph: 7
Enter the edges (source destination):
0 1
0 2
1 2
1 3
2 3
3 4
4 5
Enter the starting vertex for BFS traversal: 0
BFS traversal starting from vertex 0: 0 1 2 3 4 5

```

PROGRAM 2 : Write a program to check whether given graph is connected or not using DFS method

SOURCE CODE:

```
#include <stdio.h>

#include <stdlib.h>

struct AdjListNode {

    int dest;

    struct AdjListNode* next;

};

struct AdjList {

    struct AdjListNode *head;

};

struct Graph {

    int V;

    struct AdjList* array;

};

struct AdjListNode* newAdjListNode(int dest) {

    struct AdjListNode* newNode = (struct AdjListNode*)malloc(sizeof(struct AdjListNode));

    newNode->dest = dest;

    newNode->next = NULL;

    return newNode;

}

struct Graph* createGraph(int V) {

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

    graph->V = V;
```

```

graph->array = (struct AdjList*)malloc(V * sizeof(struct AdjList));

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

    graph->array[i].head = NULL;

return graph;
}

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

    struct AdjListNode* newNode = newAdjListNode(dest);

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

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


    newNode = newAdjListNode(src);

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

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

void DFSUtil(struct Graph* graph, int v, int* visited) {

    visited[v] = 1; // Mark the current vertex as visited

    struct AdjListNode* temp = graph->array[v].head;


    while (temp != NULL) {

        int adjVertex = temp->dest;

        if (!visited[adjVertex])

            DFSUtil(graph, adjVertex, visited);

        temp = temp->next;

    }

}

```

```

int isConnected(struct Graph* graph, int V) {

    int* visited = (int*)malloc(V * sizeof(int));

    int i;

    // Mark all vertices as not visited

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

        visited[i] = 0;

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

        if (graph->array[i].head != NULL) {

            DFSUtil(graph, i, visited);

            break;

        }

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

        if (visited[i] == 0)

            return 0;

    return 1;

}

int main() {

    int V, E; // Number of vertices and edges

    printf("Enter the number of vertices: ");

    scanf("%d", &V);

    struct Graph* graph = createGraph(V);

    printf("Enter the number of edges: ");

    scanf("%d", &E);

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

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

        int src, dest;

```

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

addEdge(graph, src, dest);

}
if (isConnected(graph, V))

    printf("The graph is connected.\n");

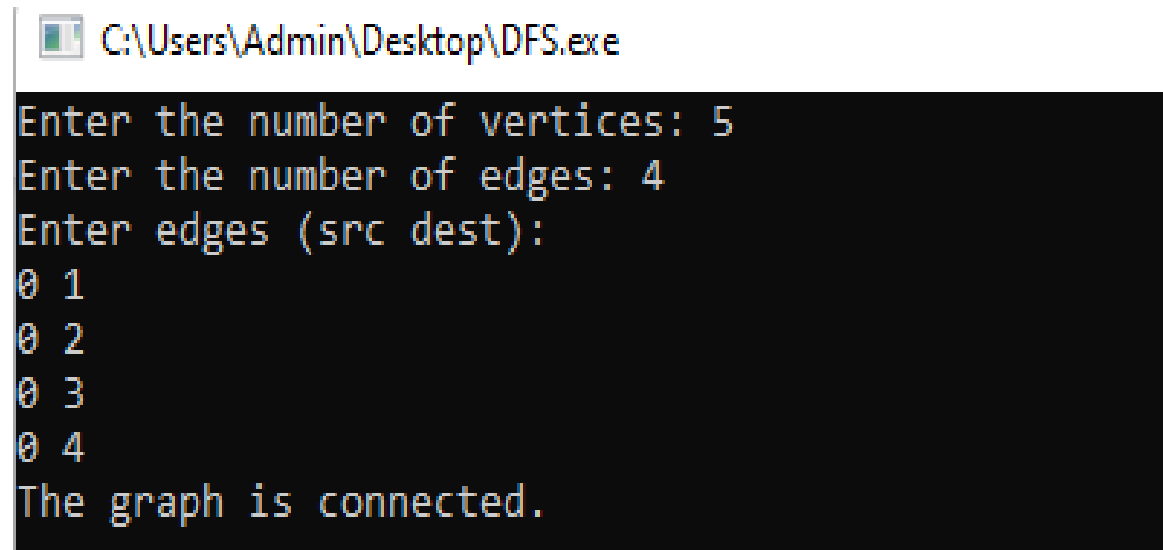
else

    printf("The graph is not connected.\n");

return 0;

}
```

OUTPUT:



```
C:\Users\Admin\Desktop\DFS.exe
Enter the number of vertices: 5
Enter the number of edges: 4
Enter edges (src dest):
0 1
0 2
0 3
0 4
The graph is connected.
```