**DSA Assignment: 11**

# EXP 11: Implementation of BFS and DFS on a directed graph using adjacency matrix.

Shashwat Tripathi

D10A Roll No: 60

**AIM:** In this experiment, we will implement BFS and DFS on a directed graph using adjacency matrix.

**CODE:**

**BFS-**

#include <stdio.h>

#include <stdlib.h>

int vertex = 5;

int adjacencyMatrix[20][20] = {

    {0, 1, 1, 1, 0},

    {1, 0, 1, 0, 0},

    {1, 1, 0, 0, 1},

    {1, 0, 0, 0, 0},

    {0, 0, 1, 0, 0}};

int queue[20], front = -1, rear = -1;

int visited[20];

int deleted;

int indexVisited = 0;

void insert(int item)

{

    if (front == -1)

    {

        front++;

    }

    rear++;

    queue[rear] = item;

}

void del(int \*deleted)

{

    if (front == -1 || front > rear)

    {

        return;

    }

    \*deleted = queue[front];

    front++;

}

int isPresentVisited(int num)

{

    int i;

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

    {

        if (visited[i] == num)

        {

            return 1;

        }

    }

    return 0;

}

int isPresentQueue(int num)

{

    for (int i = front; i <= rear; i++)

    {

        if (queue[i] == num)

        {

            return 1;

        }

    }

    return 0;

}

void bfs(int start, int vertex)

{

    visited[indexVisited++] = start;

    int i = 0;

    while (i < vertex)

    {

        if (adjacencyMatrix[start][i] && !isPresentVisited(i))

        {

            insert(i);

        }

        i++;

    }

    while (front <= rear)

    {

        del(&deleted);

        visited[indexVisited++] = deleted;

        for (i = 0; i < vertex; i++)

        {

            if (adjacencyMatrix[deleted][i] && !isPresentVisited(i) && !isPresentQueue(i))

            {

                insert(i);

            }

        }

        printf("\n");

    }

}

int main()

{

    printf("D10A\_60\_Shashwat Tripathi\n");

    bfs(0, vertex);

    for (int i = 0; i < vertex; i++)

    {

        printf("%d\t", visited[i]);

    }

}

**DFS-**

#include <stdio.h>

#include <stdlib.h>

#define MAX 50 // Max size of stack

int popped;

int stack[MAX]; // Defining stack

int vertex = 6;

int adjacencyMatrix[20][20] = {

    {0, 1, 1, 1, 0},

    {1, 0, 1, 0, 0},

    {1, 1, 0, 0, 1},

    {1, 0, 0, 0, 0},

    {0, 0, 1, 0, 0}};

int top = -1;

int visited[10]; // for printing and preventing repetition

int deleted;

int indexVisited = 0; // index of visited array

void push(int elem)

{

    top++;

    stack[top] = elem;

}

void pop(int \*popped)

{

    \*popped = stack[top];

    top--;

}

int isPresentVisited(int num)

{ // check presence of number in visited

    int i;

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

    {

        if (visited[i] == num)

        {

            return 1;

        }

    }

    return 0;

}

void dfs(int start, int vertex)

{

    push(start);

    visited[indexVisited++] = start;

    while (top >= 0)

    {

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

        {

            if (adjacencyMatrix[stack[top]][i] && !isPresentVisited(i))

            {

                visited[indexVisited++] = i;

                push(i);

                break;

            }

            if (i == vertex - 1)

            {

                pop(&popped);

            }

        }

    }

}

int main()

{

    printf("D10A\_60\_Shashwat Tripathi\n");

    dfs(2, vertex);

    for (int i = 0; i < vertex; i++)

    {

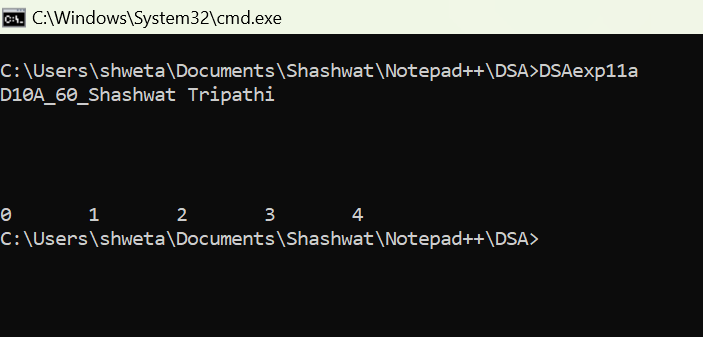
        printf("%d\t", visited[i]);

    }

}

**OUTPUT:**

**BFS-**

****

**DFS-**

**Text

Description automatically generated**