Practical no 8 1.Implementaition of DFS and BFS search algorithms.

CODE:

#include <stdio.h>

#include <stdlib.h>

#define size 8

int matrix[size][size];

int visited[size];

int queue[size];

int Rear = -1;

int Front = -1;

void enqueue(int item)

{

    if (Rear == size)

        printf("Overflow \n");

    else

    {

        if (Front == -1)

            Front = 0;

        Rear = Rear + 1;

        queue[Rear] = item;

    }

}

int isFull()

{

    if (Rear == size)

    {

        printf("Full!!\n");

        return 1;

    }

    return 0;

}

int isEmpty()

{

    if (Front == -1 || Front > Rear)

    {

        return 1;

    }

    return 0;

}

int dequeue()

{

    int val = 0;

    if (Front == -1 || Front > Rear)

    {

        printf("Underflow \n");

        exit(0);

    }

    else

    {

        val = queue[Front];

        Front = Front + 1;

    }

    return val;

}

void edge(int f, int s)

{

    matrix[f][s] = 1;

}

void bfs(int root)

{

    printf("%d ", root);

    visited[root] = 1;

    enqueue(root);

    while (!isEmpty())

    {

        int prev = dequeue();

        for (int i = prev; i < size; i++)

        {

            if (matrix[prev][i] == 1 && !visited[i])

            {

                printf("%d ", i);

                visited[i] = 1;

                enqueue(i);

            }

        }

    }

}

int main()

{

    int val = 0;

    printf("\nEnter the elements: \n");

    for (int i = 0; i < size; i++)

    {

        visited[i] = 0;

        for (int j = 0; j < size; j++)

        {

            matrix[i][j] = 0;

        }

    }

    edge(0, 1);

    edge(1, 0);

    edge(1, 5);

    edge(2, 5);

    edge(5, 1);

    edge(5, 2);

    edge(5, 7);

    edge(7, 5);

    printf("\nBFS Traversal:\n");

    bfs(0);

    printf("\nElements of Matrix: \n");

    for (int i = 0; i < size; i++)

    {

        for (int j = 0; j < size; j++)

        {

            printf("%d ", matrix[i][j]);

        }

        printf("\n");

    }

}

OUTPUT:

BFS Traversal:

0 1 5 7

Elements of Matrix:

0 1 0 0 0 0 0 0

1 0 0 0 0 1 0 0

0 0 0 0 0 1 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 1 1 0 0 0 0 1

0 0 0 0 0 0 0 0

0 0 0 0 0 1 0 0

DFS:

#include <stdio.h>

#include <stdlib.h>

#define size 8

int matrix[size][size];

int visited[size];

int queue[size];

int Rear = -1;

int Front = -1;

void edge(int f, int s)

{

    matrix[f][s] = 1;

}

void dfs(int root)

{

    printf("%d ", root);

    visited[root] = 1;

    for (int j = 0; j < size; j++)

    {

        if (matrix[root][j] == 1 && !visited[j])

        {

            dfs(j);

        }

    }

}

int main()

{

    int val = 0;

    printf("\nEnter the elements: \n");

    for (int i = 0; i < size; i++)

    {

        visited[i] = 0;

        for (int j = 0; j < size; j++)

        {

            matrix[i][j] = 0;

        }

    }

    edge(0, 1);

    edge(1, 0);

    edge(1, 5);

  edge(2, 5);

    edge(5, 1);

    edge(5, 2);

    edge(5, 7);

    edge(7, 5);

    printf("\nDFS Traversal:\n");

    dfs(0);

    printf("\nElements of Matrix: \n");

    for (int i = 0; i < size; i++)

    {

        for (int j = 0; j < size; j++)

        {

            printf("%d ", matrix[i][j]);

        }

        printf("\n");

    }

}

OUTPUT:

DFS Traversal:

0 1 5 2 7

Elements of Matrix:

0 1 0 0 0 0 0 0

1 0 0 0 0 1 0 0

0 0 0 0 0 1 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 1 1 0 0 0 0 1

0 0 0 0 0 0 0 0

0 0 0 0 0 1 0 0