DSA lab

**Assignment 9**

**Name:** Madhav Jha

**Roll no.:** E3-48

**Branch:** CSE (AI & ML)

# Graph Traversal

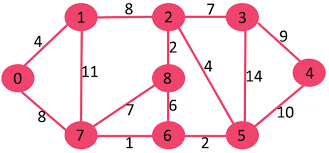
Write a menu driven program for graph traversal:

1. Create graph

2. Display using BFS

3. Display Using DFS

## Example graph:



## Code:

#include <stdio.h>

#include <stdlib.h>

int edges[1000][1000];

void edgePrint(int n) {

    printf("\n\n  | ");

    for (int i = 0;i < n;++i) {

        printf("%d ", i);

    }

    printf("\n--|");

    for (int i = 0;i < n;++i) {

        printf("--");

    }

    for (int i = 0;i < n;i++) {

        printf("\n%d | ", i);

        for (int j = 0;j < n;++j) {

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

        }

    }

    printf("\n");

}

int q[1000], s = 0, e = 0;

void pushQ(int n) {

    q[e++] = n;

}

int popQ() {

    return q[s++];

}

int isEmptyQ() {

    return ((s >= e) ? 1 : 0);

}

void bfsPrint(int n) {

    pushQ(1);

    int count = 0;

    int visited[n];

    for (int i = 0;i < n;++i) {

        visited[i] = 0;

    }

    visited[1] = 1;

    while (isEmptyQ() != 1) {

        int node = popQ();

        printf("%d ", node);

        for (int i = 0;i < n;i++) {

            if (edges[node][i] == 1 && visited[i] == 0) {

                pushQ(i);

                visited[i] = 1;

            }

        }

    }

}

int stack[1000], se = -1;

void pushS(int n) {

    stack[++se] = n;

}

int popS() {

    return stack[se--];

}

int topS() {

    return stack[se];

}

int isEmptyS() {

    return ((se < 0) ? 1 : 0);

}

void dfsPrint(int n) {

    pushS(1);

    int count = 0;

    int visited[n];

    for (int i = 0;i < n;++i) {

        visited[i] = 0;

    }

    visited[1] = 1;

    printf("1 ");

    while (isEmptyS() != 1) {

        int node = topS();

        int flag = 0;

        for (int i = 0;i < n;i++) {

            if (edges[node][i] == 1 && visited[i] == 0) {

                pushS(i);

                visited[i] = 1;

                printf("%d ", i);

                flag = 1;

                break;

            }

        }

        if (flag == 0) {

            popS();

        }

    }

}

int main(void) {

    printf("\n\n:::::Graph Traversal:::::\n\n");

    int n = 0;

    printf("Enter number of nodes: ");

    scanf("%d", &n);

    // put all edges as 0

    for (int i = 0;i < n;i++) {

        for (int j = 0;j < n;++j) {

            edges[i][j] = 0;

        }

    }

    int e = 0;

    printf("\nEnter number of edges: ");

    scanf("%d", &e);

    if (e < n - 1) {

        printf("\n\nERROR: Edges should be >= %d\n\n", n - 1);

        return 0;

    }

    printf("\nEnter edges in following format: from to \n(e.g.: 2 4)");

    for (int i = 0;i < e;i++) {

        int a, b;

        printf("\nEdge: ");

        scanf("%d", &a);

        scanf("%d", &b);

        edges[a][b] = 1;

        edges[b][a] = 1;

    }

    printf("\n\nEdge matrix: ");

    edgePrint(n);

    printf("\nBFS traversal: ");

    bfsPrint(n);

    printf("\n\nDFS traversal: ");

    dfsPrint(n);

    printf("\n\n");

    return 0;

}

## Output:

:::::Graph Traversal:::::  
  
Enter number of nodes: 9  
  
Enter number of edges: 14  
  
Enter edges in following format: from to   
(e.g.: 2 4)  
Edge: 0 7  
Edge: 0 1  
Edge: 1 7  
Edge: 1 2  
Edge: 7 8  
Edge: 7 6  
Edge: 2 8  
Edge: 8 6  
Edge: 2 5  
Edge: 6 5  
Edge: 2 3  
Edge: 3 4  
Edge: 5 4  
Edge: 3 5  
  
Edge matrix:  
 | 0 1 2 3 4 5 6 7 8  
--|------------------  
0 | 0 1 0 0 0 0 0 1 0  
1 | 1 0 1 0 0 0 0 1 0  
2 | 0 1 0 1 0 1 0 0 1  
3 | 0 0 1 0 1 1 0 0 0  
4 | 0 0 0 1 0 1 0 0 0   
5 | 0 0 1 1 1 0 1 0 0  
6 | 0 0 0 0 0 1 0 1 1  
7 | 1 1 0 0 0 0 1 0 1  
8 | 0 0 1 0 0 0 1 1 0  
  
BFS traversal: 1 0 2 7 3 5 8 6 4   
  
DFS traversal: 1 0 7 6 5 2 3 4 8  
  
PS E:\Google Drive\Classroom\SEM-3\DSA\_lab-sem3\lab-9\_(graph)>