Shrihari K Mangle
CSB2-74 12010517

LAB-08

8. Write a C/C++ program to represent graph in its adjacency Matrix and Adjacency Lists representation format. Also Perform BFS and DFS traversal on same.

CODE :

```c
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#define maxNode 4



struct Node
{
    int data;
    struct Node *next;
};

struct Node *top = NULL;

int isEmptyStack(struct Node *top)
{
    if (top == NULL)
    {
        return 1;
    }

    return 0;
}

int isFullStack(struct Node *top)
{
    struct Node *p = (struct Node *)malloc(sizeof(struct Node));

    if (p == NULL)
    {
        return 1;
    }

    return 0;
}
```

```
struct Node *push(struct Node *top, int value)
{


        struct Node *n = (struct Node *)malloc(sizeof(struct Node));
        n->data = value;
        n->next = top;
        top = n;

        return top;

}

int pop()
{

        struct Node *n = top;
        int x = top->data;
        top = top->next;
        free(n);

        return x;

}

int stackTop()
{
    return top->data;
}


// N vertices and M Edges
int N, M;

// Function to create Adjacency Matrix
void createAdjMatrix(int Adj[][N + 1],
                     int arr[][2])
{

    for (int i = 0; i < N + 1; i++)
    {

        for (int j = 0; j < N + 1; j++)
        {
            Adj[i][j] = 0;
        }
    }

    for (int i = 0; i < M; i++)
    {
```

```
        int x = arr[i][0];
        int y = arr[i][1];

        Adj[x][y] = 1;
        Adj[y][x] = 1;
    }
}

void printAdjMatrix(int Adj[][N + 1])
{
    printf("\nADJECENCY MATRIX REPRESENTATION : \n");

    for (int i = 0; i < N; i++)
    {
        printf("%d : ", i);
        for (int j = 0; j < N; j++)
        {

            printf("%d ", Adj[i][j]);
        }
        printf("\n");
    }
}

typedef struct node
{
    int data;
    struct node *next;
} node;

typedef struct list
{
    node *head;
} list;

list *adjList[maxNode] = {0};

void addNode(int s, int d)
{
    node *dest, *temp, *src;

    if (adjList[s]->head == NULL)
    {
        src = (node *)malloc(sizeof(node));
        src->data = s;
        src->next = NULL;
        adjList[s]->head = src;
    }

    dest = (node *)malloc(sizeof(node));
```

```
    dest->data = d;
    dest->next = NULL;
    temp = adjList[s]->head;

    while (temp->next != NULL)
    {
        temp = temp->next;
    }
    temp->next = dest;
}

void printList()
{
    printf("\nADJECENCY LIST REPRESENTATION : \n");

    for (int i = 0; i < maxNode; i++)
    {
        node *p = adjList[i]->head;

        while (p)
        {
            printf("%d-> ", p->data);
            p = p->next;
        }
        printf("NULL");
        printf("\n");
    }
}

struct queue
{
    int size;
    int f;
    int r;
    int *arr;
};

int isEmpty(struct queue *q)
{
    if (q->r == q->f)
    {
        return 1;
    }
    return 0;
}

int isFullQueue(struct queue *q)
{
    if (q->r == q->size - 1)
    {
        return 1;
```

```
    }
    return 0;
}

void enqueue(struct queue *q, int val)
{
    if (isFullQueue(q))
    {
         printf("This Queue is full\n");
    }
    else
    {
        q->r++;
        q->arr[q->r] = val;

    }
}

int dequeue(struct queue *q)
{
    int a = -1;
    if (isEmpty(q))
    {
        printf("This Queue is empty\n");
    }
    else
    {
        q->f++;
        a = q->arr[q->f];
    }
    return a;
}

int visited[4] = {0, 0, 0, 0};
int a[4][4] =
    {
        /*0 -> */ {0, 1, 1, 1},
        /*1 -> */ {1, 0, 1, 0},
        /*2 -> */ {1, 1, 0, 1},
        /*3 -> */ {1, 0, 1, 0},
};

void BFStraversal(struct queue *q, int i)
{
    int node;
    int visited1[4] = {0, 0, 0, 0};
    visited1[i] = 1;
    enqueue(q, i); // Enqueue i for exploration

    while (!isEmpty(q))
    {
```

```
        int node = dequeue(q);
        printf("%d ", node);

        for (int j = 0; j < 4; j++)
        {
            if (a[node][j] == 1 && visited1[j] == 0)
            {
                // printf("%d ", j);
                visited1[j] = 1;
                enqueue(q, j);
            }
        }
    }
}

void DFStraversal(int i)
{
    // int visited[4] = {0, 0, 0, 0};
    printf("%d ", i);
    visited[i] = 1;
    for (int j = 0; j < 4; j++)
    {
        if (a[i][j] == 1 && !visited[j])
        {
            DFStraversal(j);
        }
    }

}

void BFSAdjLists(struct queue *q, int n)
{
    int v1,visited[4];
    struct node *p;
    for(v1=0;v1<n;v1++)
    {
        visited[v1]=0;
    }

    printf("Enter start vertex : ");
    scanf("%d",&v1);

    enqueue(q, v1);
    visited[v1]=1;
    printf("BFS traversal : ");

    while(!isEmpty(q))
    {
        v1=dequeue(q);
        printf("%d ",v1);
        p=adjList[v1]->head;
```

```
        while(p!=NULL)
        {
            if(visited[p->data]==0)
            {
                enqueue(q, p->data);
                visited[p->data]=1;
            }
            p=p->next;
        }
    }
}

void DFSadjList(struct Node *top, int n)
{
    struct node *p;
    int v, visited[4], f;
    for (int i = 0; i < n; i++)
    {
        visited[i] = 0;
    }

    printf("Enter starting vertex: ");
    scanf("%d", &v);
    visited[v] = 1;
    top = push(top, v);
    printf("%d ", v);

    do
    {
        p = adjList[v]->head;
        while (p!=NULL)
        {
            if(visited[p->data]==0)
            {
                visited[p->data] = 1;
                top = push(top, p->data);
                printf("%d ", p->data);
                v = p->data;
                break;
            }
            else
            {
                p = p->next;
            }

        }

        if (p==NULL)
        {
            f = pop();
```

```
            if (!isEmptyStack(top))
            {
               v = stackTop();
            }

        }

    } while (!isEmptyStack(top));


}

void LinkedListDisplay(struct Node *ptr)
{
    while (ptr != NULL)
    {
        printf("Element : %d\n", ptr->data);
        ptr = ptr->next;
    }
}

int main()
{
    printf("\nGiven graph :\n\n");

    printf("0---1\n");
    printf("| \\ |\n");
    printf("3---2\n");

    int v, z;
    // Number of vertices
    N = 4;

    // Given Edges
    int arr[][2] = {

        {0, 1}, {0, 2}, {0, 3}, {1, 0}, {1, 2}, {2, 0}, {2, 1}, {2, 3}, {3,
0}, {3, 2}

    };

    M = sizeof(arr) / sizeof(arr[0]);
    int Adj[N][N];

    struct queue q;
    q.size = 100;
    q.f = q.r = -1;
    q.arr = (int *)malloc(q.size * sizeof(int));

    int ch;
    while (1)
```

```
    {
        printf("\n1.Adjacency Matrix\n2.Adjacency List\n3.BFS Traversal
using Adjacency Matrix\n4.BFS traversal using adjacency list\n5.DFS
Traversal\n6.DFS traversal using adjcency list\n7.Exit\n ");
        printf("\nEnter the choice : ");
        scanf("%d", &ch);

        switch (ch)
        {
        case 1:
            // Number of Edges
            // printf("%d\n", sizeof(arr));
            // printf("%d\n", sizeof(arr[0]));

            createAdjMatrix(Adj, arr);

            printAdjMatrix(Adj);
            break;

        case 2:
            for (int i = 0; i < maxNode; i++)
            {
                adjList[i] = (list *)malloc(sizeof(list));
                adjList[i]->head = NULL;
            }

            addNode(0, 1);
            addNode(0, 2);
            addNode(0, 3);
            addNode(1, 0);
            addNode(1, 2);
            addNode(2, 0);
            addNode(2, 1);
            addNode(2, 3);
            addNode(3, 0);
            addNode(3, 2);

            printList();
            break;

        case 3:
            printf("\nENTER THE VERTEX : ");
            scanf("%d", &v);
            printf("BFS traversal : ");
            BFStraversal(&q, v);
            break;

        case 4:
            BFSAdjLists(&q,4);
            break;
```

```
        case 5:
            printf("\nENTER THE VERTEX : ");
            scanf("%d", &z);
            printf("DFS traversal : ");
            DFStraversal(z);
            break;

        case 6:

            //printf("DFS traversal using adjcency list : \n");
            DFSadjList(top, 4);
            //LinkedListDisplay(top);
            break;

        case 7:
            printf("Exiting.....\n\n");
            exit(0);
        }
        printf("\n");
    }



//      0---1
//     / \ /
//    3---2

// 0-> 1->2->3
// 1-> 0->2
// 2-> 0->1->3
// 3-> 0->2

    return 0;
}
```

OUTPUT :

```
PS E:\> cd "e:\VIT\SECOND YEAR(SY)\SEM 2\DATA STRUCTURES(DS)\DATA
\Graph_Assgn }

Given graph :

0---1
| \ |
3---2

1.Adjacency Matrix
2.Adjacency List
3.BFS Traversal using Adjacency Matrix
4.BFS traversal using adjacency list
5.DFS Traversal
6.DFS traversal using adjcency list
7.Exit

Enter the choice : 1

ADJECENCY MATRIX REPRESENTATION :
0 : 0 1 1 1
1 : 1 0 1 0
2 : 1 1 0 1
3 : 1 0 1 0
```

```
1.Adjacency Matrix
2.Adjacency List
3.BFS Traversal using Adjacency Matrix
4.BFS traversal using adjacency list
5.DFS Traversal
6.DFS traversal using adjcency list
7.Exit

Enter the choice : 2

ADJECENCY LIST REPRESENTATION :
0-> 1-> 2-> 3-> NULL
1-> 0-> 2-> NULL
2-> 0-> 1-> 3-> NULL
3-> 0-> 2-> NULL


1.Adjacency Matrix
2.Adjacency List
3.BFS Traversal using Adjacency Matrix
4.BFS traversal using adjacency list
5.DFS Traversal
6.DFS traversal using adjcency list
7.Exit

Enter the choice : 3

ENTER THE VERTEX : 2
BFS traversal : 2 0 1 3
```

```
1.Adjacency Matrix
2.Adjacency List
3.BFS Traversal using Adjacency Matrix
4.BFS traversal using adjacency list
5.DFS Traversal
6.DFS traversal using adjcency list
7.Exit
Enter the choice : 4
Enter start vertex : 2
BFS traversal : 2 0 1 3

1.Adjacency Matrix
2.Adjacency List
3.BFS Traversal using Adjacency Matrix
4.BFS traversal using adjacency list
5.DFS Traversal
6.DFS traversal using adjcency list
7.Exit

Enter the choice : 5

ENTER THE VERTEX : 0
DFS traversal : 0 1 2 3

1.Adjacency Matrix
2.Adjacency List
3.BFS Traversal using Adjacency Matrix
4.BFS traversal using adjacency list
5.DFS Traversal
6.DFS traversal using adjcency list
7.Exit

Enter the choice : 6
Enter starting vertex: 0
0 1 2 3
```

```
1.Adjacency Matrix
2.Adjacency List
3.BFS Traversal using Adjacency Matrix
4.BFS traversal using adjacency list
5.DFS Traversal
6.DFS traversal using adjcency list
7.Exit

Enter the choice : 7
Exiting.....

PS E:\VIT\SECOND YEAR(SY)\SEM 2\DATA STRUCTURES(DS)\DATA STRUCTURES LAB>
```