DATA STRUCTURES

Division	**CS(AIML) -A**
Batch	**2**
GR-no	**12311493**
Roll no	**54**
Name	**Atharva Kangralkar**

Assignment 11:

WAP to implement DFS and BFS traversal on Graph using Adjacency Matrix and Adjacency Lists.

Code:-

1. Adjacency Matrix

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

#define MAX 100

int adjMatrix[MAX][MAX];
int visited[MAX];
int queue[MAX];
int front = -1, rear = -1;

void addEdge(int u, int v) {
  adjMatrix[u][v] = 1;
  adjMatrix[v][u] = 1;
}

void dfs(int v, int n) {
  visited[v] = 1;
  printf("%d ", v);
  for (int i = 0; i < n; i++) {
    if (adjMatrix[v][i] == 1 && !visited[i]) {
      dfs(i, n);
    }
  }
}

void enqueue(int val) {
  if (rear == MAX - 1) return;
  if (front == -1) front = 0;
  queue[++rear] = val;
}

int dequeue() {
  if (front == -1 || front > rear) return -1;
  return queue[front++];
}

void bfs(int start, int n) {
```

```
    for (int i = 0; i < n; i++) visited[i] = 0;

    enqueue(start);
    visited[start] = 1;

    while (front <= rear) {
        int current = dequeue();
        printf("%d ", current);

        for (int i = 0; i < n; i++) {
            if (adjMatrix[current][i] == 1 && !visited[i]) {
                enqueue(i);
                visited[i] = 1;
            }
        }
    }
}

int main() {
    int n, e, u, v;
    printf("Enter the number of vertices: ");
    scanf("%d", &n);
    printf("Enter the number of edges: ");
    scanf("%d", &e);

    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            adjMatrix[i][j] = 0;
    printf("Enter the edges (u v) for an undirected graph:\n");

    for (int i = 0; i < e; i++) {
        scanf("%d %d", &u, &v);
        addEdge(u, v);
    }


    printf("DFS (Adjacency Matrix): ");
    for (int i = 0; i < n; i++) visited[i] = 0;
    dfs(0, n);

    printf("\nBFS (Adjacency Matrix): ");
```

```
    bfs(0, n);

    return 0;
}
```

2. Adjacency List

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

#define MAX 100

typedef struct Node {
    int vertex;
    struct Node* next;
} Node;

Node* adjList[MAX];
int visited[MAX];
int queue[MAX];
int front = -1, rear = -1;

Node* createNode(int v) {
    Node* newNode = malloc(sizeof(Node));
    newNode->vertex = v;
    newNode->next = NULL;
    return newNode;
}

void addEdge(int u, int v) {
    Node* newNode = createNode(v);
    newNode->next = adjList[u];
    adjList[u] = newNode;

    newNode = createNode(u);
    newNode->next = adjList[v];
    adjList[v] = newNode;
}

void dfs(int v) {
    visited[v] = 1;
    printf("%d ", v);
```

```
    Node* temp = adjList[v];
    while (temp != NULL) {
        int connected = temp->vertex;
        if (!visited[connected]) {
            dfs(connected);
        }
        temp = temp->next;
    }
}

void enqueue(int val) {
    if (rear == MAX - 1) return;
    if (front == -1) front = 0;
    queue[++rear] = val;
}

int dequeue() {
    if (front == -1 || front > rear) return -1;
    return queue[front++];
}

void bfs(int start) {
    for (int i = 0; i < MAX; i++) visited[i] = 0;

    enqueue(start);
    visited[start] = 1;

    while (front <= rear) {
        int current = dequeue();
        printf("%d ", current);

        Node* temp = adjList[current];
        while (temp != NULL) {
            int connected = temp->vertex;
            if (!visited[connected]) {
                enqueue(connected);
                visited[connected] = 1;
            }
            temp = temp->next;
        }
```

```
    }
}

int main() {
    int n, e, u, v;
    printf("Enter the number of vertices: ");
    scanf("%d", &n);
    printf("Enter the number of edges: ");
    scanf("%d", &e);

    // Initialize adjacency list
    for (int i = 0; i < n; i++)
        adjList[i] = NULL;

    // Input edges
    printf("Enter the edges (u v) for an undirected graph:\n");
    for (int i = 0; i < e; i++) {
        scanf("%d %d", &u, &v);
        addEdge(u, v);
    }

    // DFS and BFS
    printf("DFS (Adjacency List): ");
    for (int i = 0; i < n; i++) visited[i] = 0;
    dfs(0);  // Start DFS from vertex 0

    printf("\nBFS (Adjacency List): ");
    front = rear = -1;
    bfs(0);  // Start BFS from vertex 0

    return 0;
}
```

Output:-

1. **Adjacency matrix:-**

```
Enter the number of vertices: 5
Enter the number of edges: 4
Enter the edges (u v) for an undirected graph:
0 1
0 2
1 3
1 4
DFS (Adjacency Matrix): 0 1 3 4 2
BFS (Adjacency Matrix): 0 1 2 3 4
```

2. **Adjacency list:-**

```
Enter the number of vertices: 4
Enter the number of edges: 3
Enter the edges (u v) for an undirected graph:
0 2
0 3
1 2
DFS (Adjacency List): 0 3 2 1
BFS (Adjacency List): 0 3 2 1
```