DS Lab Assignment 10

Pranav Joshi
CS-B Batch 2
Roll no: 43

Title: WAP to Perform BFS and DFS traversals on Graph using Adjacency Matrix and Adjacency Lists.

Code:

1. BFS Traversal on graph using Adjacency Matrix

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

#define MAX_NODES 100

int adj[MAX_NODES][MAX_NODES];
int visited[MAX_NODES];
int queue[MAX_NODES];
int weight[MAX_NODES][MAX_NODES];

int front = -1, rear = -1;

void add_edge(int u, int v, int w) {
    adj[u][v] = 1;
    adj[v][u] = 1;
    weight[u][v] = w;
    weight[v][u] = w;
}

void bfs(int start, int num_nodes) {
    int i, curr_node, curr_weight;

    visited[start] = 1;
    queue[++rear] = start;

    while (front != rear) {
        curr_node = queue[++front];
        printf("%d ", curr_node);

        for (i = 0; i < num_nodes; i++) {
```

```c
            if (adj[curr_node][i] == 1 && visited[i] == 0) {
                visited[i] = 1;
                queue[++rear] = i;
                curr_weight = weight[curr_node][i];
            }
        }
    }
}

int main() {
    int num_nodes, num_edges, i, u, v, w, start;

    printf("Enter the number of nodes: ");
    scanf("%d", &num_nodes);

    printf("Enter the number of edges: ");
    scanf("%d", &num_edges);

    for (i = 0; i < num_nodes; i++) {
        visited[i] = 0;
        for (int j = 0; j < num_nodes; j++) {
            adj[i][j] = 0;
            weight[i][j] = 0;
        }
    }

    for (i = 0; i < num_edges; i++) {
        printf("Enter the endpoints and weight of edge %d: ", i+1);
        scanf("%d%d%d", &u, &v, &w);
        add_edge(u, v, w);
    }

    //          0
    //        /   \
    //      1      2
    //     / \   /
    //    3   4
    //     \  /
    //       5

    printf("Enter the starting node for BFS: ");
    scanf("%d", &start);

    printf("BFS traversal using Adjacency Matrix: ");
    bfs(start, num_nodes);

    return 0;
}
```

Output:

```
PS C:\Code\C\Code> cd "c:\Code\C\Code\" ; if ($?) { gcc Sem4Assignment10.c
Enter the number of nodes: 6
Enter the number of edges: 7
Enter the endpoints and weight of edge 1: 0 1 1
Enter the endpoints and weight of edge 2: 0 2 1
Enter the endpoints and weight of edge 3: 1 3 2
Enter the endpoints and weight of edge 4: 1 4 2
Enter the endpoints and weight of edge 5: 2 4 3
Enter the endpoints and weight of edge 6: 3 5 4
Enter the endpoints and weight of edge 7: 4 5 4
Enter the starting node for BFS: 0
BFS traversal using Adjacency Matrix: 0 1 2 3 4 5
PS C:\Code\C\Code>
```

2. BFS Traversal on graph using Adjacency List

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

#define MAX_NODES 100

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

struct node* adj[MAX_NODES];

int visited[MAX_NODES];

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

void add_edge(int u, int v, int weight) {
    struct node* new_node = (struct node*)malloc(sizeof(struct node));
    new_node->data = v;
    new_node->weight = weight;
    new_node->next = adj[u];
    adj[u] = new_node;

    new_node = (struct node*)malloc(sizeof(struct node));
    new_node->data = u;
```

```
    new_node->weight = weight;
    new_node->next = adj[v];
    adj[v] = new_node;
}

void bfs(int start) {
    int i, curr_node, curr_weight;
    struct node* temp;

    visited[start] = 1;
    queue[++rear] = start;

    while (front != rear) {
        curr_node = queue[++front];
        printf("%d ", curr_node);

        temp = adj[curr_node];
        while (temp != NULL) {
            i = temp->data;
            curr_weight = temp->weight;
            if (visited[i] == 0) {
                visited[i] = 1;
                queue[++rear] = i;
            }
            temp = temp->next;
        }
    }
}

int main() {
    int num_nodes, num_edges, i, u, v, weight, start;

    printf("Enter the number of nodes: ");
    scanf("%d", &num_nodes);

    printf("Enter the number of edges: ");
    scanf("%d", &num_edges);

    for (i = 0; i < num_nodes; i++) {
        adj[i] = NULL;
        visited[i] = 0;
    }

    for (i = 0; i < num_edges; i++) {
        printf("Enter the endpoints and weight of edge %d: ", i+1);
        scanf("%d%d%d", &u, &v, &weight);
        add_edge(u, v, weight);
    }
```

```
    //          1
    //        /   \
    //      3      2
    //     /      /  \
    //    6      5    4

    printf("Enter the starting node for BFS: ");
    scanf("%d", &start);

    printf("BFS traversal using Adjacency List: ");
    bfs(start);

    return 0;
}
```

Output:

```
PS C:\Code\C\Code> cd "c:\Code\C\Code\" ; if ($?) { gcc Sem4Assignment10.c
Enter the number of nodes: 6
Enter the number of edges: 5
Enter the endpoints and weight of edge 1: 1 2 1
Enter the endpoints and weight of edge 2: 1 3 1
Enter the endpoints and weight of edge 3: 2 4 2
Enter the endpoints and weight of edge 4: 2 5 2
Enter the endpoints and weight of edge 5: 3 6 3
Enter the starting node for BFS: 1
BFS traversal using Adjacency List: 1 3 2 6 5 4
PS C:\Code\C\Code>
```

3. DFS Traversal on graph using Adjacency Matrix

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

#define MAX_NODES 100

int adj[MAX_NODES][MAX_NODES];
int weights[MAX_NODES][MAX_NODES];

int visited[MAX_NODES];

void add_edge(int u, int v, int w) {
    adj[u][v] = 1;
    adj[v][u] = 1;
    weights[u][v] = w;
    weights[v][u] = w;
}
```

```c
void dfs(int node, int num_nodes) {
    int i;

    visited[node] = 1;
    printf("%d ", node);

    for (i = 0; i < num_nodes; i++) {
        if (adj[node][i] && !visited[i]) {
            dfs(i, num_nodes);
        }
    }
}

int main() {
    int num_nodes, num_edges, i, u, v, w, start;

    printf("Enter the number of nodes: ");
    scanf("%d", &num_nodes);

    printf("Enter the number of edges: ");
    scanf("%d", &num_edges);

    for (i = 0; i < num_nodes; i++) {
        visited[i] = 0;
        for (int j = 0; j < num_nodes; j++) {
            adj[i][j] = 0;
            weights[i][j] = 0;
        }
    }

    for (i = 0; i < num_edges; i++) {
        printf("Enter the endpoints and weight of edge %d: ", i+1);
        scanf("%d%d%d", &u, &v, &w);
        add_edge(u, v, w);
    }

    //          0
    //        /   \
    //      1      2
    //     / \   /
    //    3   4
    //     \  /
    //       5

    printf("Enter the starting node for DFS: ");
    scanf("%d", &start);
```

```
    printf("DFS traversal using Adjacency Matrix: ");
    dfs(start, num_nodes);

    return 0;
}
```

Output:

```
PS C:\Code\C\Code> cd "c:\Code\C\Code\" ; if ($?) { gcc Sem4Assignment10.c
Enter the number of nodes: 6
Enter the number of edges: 7
Enter the endpoints and weight of edge 1: 0 1 1
Enter the endpoints and weight of edge 2: 0 2 1
Enter the endpoints and weight of edge 3: 1 3 2
Enter the endpoints and weight of edge 4: 1 4 2
Enter the endpoints and weight of edge 5: 2 4 3
Enter the endpoints and weight of edge 6: 3 5 4
Enter the endpoints and weight of edge 7: 4 5 4
Enter the starting node for DFS: 0
DFS traversal using Adjacency Matrix: 0 1 3 5 4 2
PS C:\Code\C\Code>
```

4. DFS Traversal on graph using Adjacency List

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

#define MAX_NODES 100

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

struct node* adj[MAX_NODES];

int visited[MAX_NODES];

void add_edge(int u, int v, int w) {
    struct node* new_node = (struct node*)malloc(sizeof(struct node));
    new_node->data = v;
    new_node->weight = w;
    new_node->next = adj[u];
    adj[u] = new_node;
```

```
    new_node = (struct node*)malloc(sizeof(struct node));
    new_node->data = u;
    new_node->weight = w;
    new_node->next = adj[v];
    adj[v] = new_node;
}

void dfs(int node) {
    struct node* temp = adj[node];

    visited[node] = 1;
    printf("%d ", node);

    while (temp != NULL) {
        int adj_node = temp->data;
        if (!visited[adj_node]) {
            dfs(adj_node);
        }
        temp = temp->next;
    }
}

int main() {
    int num_nodes, num_edges, i, u, v, w, start;

    printf("Enter the number of nodes: ");
    scanf("%d", &num_nodes);

    printf("Enter the number of edges: ");
    scanf("%d", &num_edges);

    for (i = 0; i < num_nodes; i++) {
        adj[i] = NULL;
        visited[i] = 0;
    }

    for (i = 0; i < num_edges; i++) {
        printf("Enter the endpoints and weight of edge %d: ", i+1);
        scanf("%d%d%d", &u, &v, &w);
        add_edge(u, v, w);
    }

    //          1
    //        /   \
    //      3      2
    //     /      /  \
    //    6      5    4
```

```c
    printf("Enter the starting node for DFS: ");
    scanf("%d", &start);

    printf("DFS traversal using Adjacency List: ");
    dfs(start);

    return 0;
}
```

Output:

```
PS C:\Code\C\Code> cd "c:\Code\C\Code\" ; if ($?) { gcc Sem4Assignment10.c
Enter the number of nodes: 6
Enter the number of edges: 5
Enter the endpoints and weight of edge 1: 1 2 1
Enter the endpoints and weight of edge 2: 1 3 1
Enter the endpoints and weight of edge 3: 2 4 2
Enter the endpoints and weight of edge 4: 2 5 2
Enter the endpoints and weight of edge 5: 3 6 3
Enter the starting node for DFS: 1
DFS traversal using Adjacency List: 1 3 6 2 5 4
PS C:\Code\C\Code>
```