

Practical 10

AIM: GRAPH REPRESENTATION

Implement graph data structure using adjacency matrix and adjacency list representation. Perform the graph traversal such as breadth-first-search (BFS) and depth-first-search (DFS).

Program:-

```
#include <stdio.h>

#define MAX 10

int graph_matrix[MAX][MAX]; // Adjacency matrix
int graph_list[MAX][MAX]; // Array-based adjacency list
int list_size[MAX];

// Tracks the number of neighbors for each vertex

void add_edge_matrix(int u, int v) {
    graph_matrix[u][v] = 1;
    graph_matrix[v][u] = 1; // For undirected graph
}

void add_edge_list(int u, int v) {
    graph_list[u][list_size[u]++] = v;
    graph_list[v][list_size[v]++] = u;
}

void bfs_matrix(int start, int vertices) {
    int visited[MAX] = {0}, to_visit[MAX], front = 0, rear = 0;

    printf("BFS with Matrix: ");

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

    while (front < rear) {
        int curr = to_visit[front++];
        printf("%d ", curr);

        for (int i = 0; i < vertices; i++) {
            if ((graph_matrix[curr][i] == 1) && (visited[i] == 0)) {
                to_visit[rear++] = i;
                visited[i] = 1;
            }
        }
    }
}
```

```

    }

}

}

printf("\n");
}

void bfs_list(int start, int vertices) {

    int visited[MAX] = {0}, to_visit[MAX], front = 0, rear = 0;

    printf("BFS with List: ");

    to_visit[rear++] = start;

    visited[start] = 1;

    while (front < rear) {

        int curr = to_visit[front++];

        printf("%d ", curr);

        for (int i = 0; i < list_size[curr]; i++) {

            int neighbor = graph_list[curr][i];

            if (visited[neighbor] == 0) {

                to_visit[rear++] = neighbor;

                visited[neighbor] = 1;

            }

        }

    }

    printf("\n");

}

```

```

void dfs_matrix(int start, int visited[], int vertices) {

    visited[start] = 1;

    printf("%d ", start);

    for (int i = 0; i < vertices; i++) {

        if ((graph_matrix[start][i] == 1) && (visited[i] == 0)) {

```

```

        dfs_matrix(i, visited, vertices);
    }
}

void dfs_list(int start, int visited[]) {
    visited[start] = 1;
    printf("%d ", start);

    for (int i = 0; i < list_size[start]; i++) {
        int neighbor = graph_list[start][i];
        if (visited[neighbor] == 0) {
            dfs_list(neighbor, visited);
        }
    }
}

int main() {
    int vertices = 5, visited[MAX] = {0};

    // Add edges to adjacency matrix
    add_edge_matrix(0, 1);
    add_edge_matrix(0, 2);

    add_edge_matrix(1, 3);
    add_edge_matrix(1, 4);
    bfs_matrix(0, vertices);
    printf("DFS with Matrix: ");
    dfs_matrix(0, visited, vertices);
    printf("\n");

    // Reset adjacency list and visited array
    for (int i = 0; i < vertices; i++) {
        list_size[i] = 0;
        visited[i] = 0;
    }

    // Add edges to adjacency list

```

```

add_edge_list(0, 1);

add_edge_list(0, 2);

add_edge_list(1, 3);

add_edge_list(1, 4);

bfs_list(0, vertices);

printf("DFS with List: ");

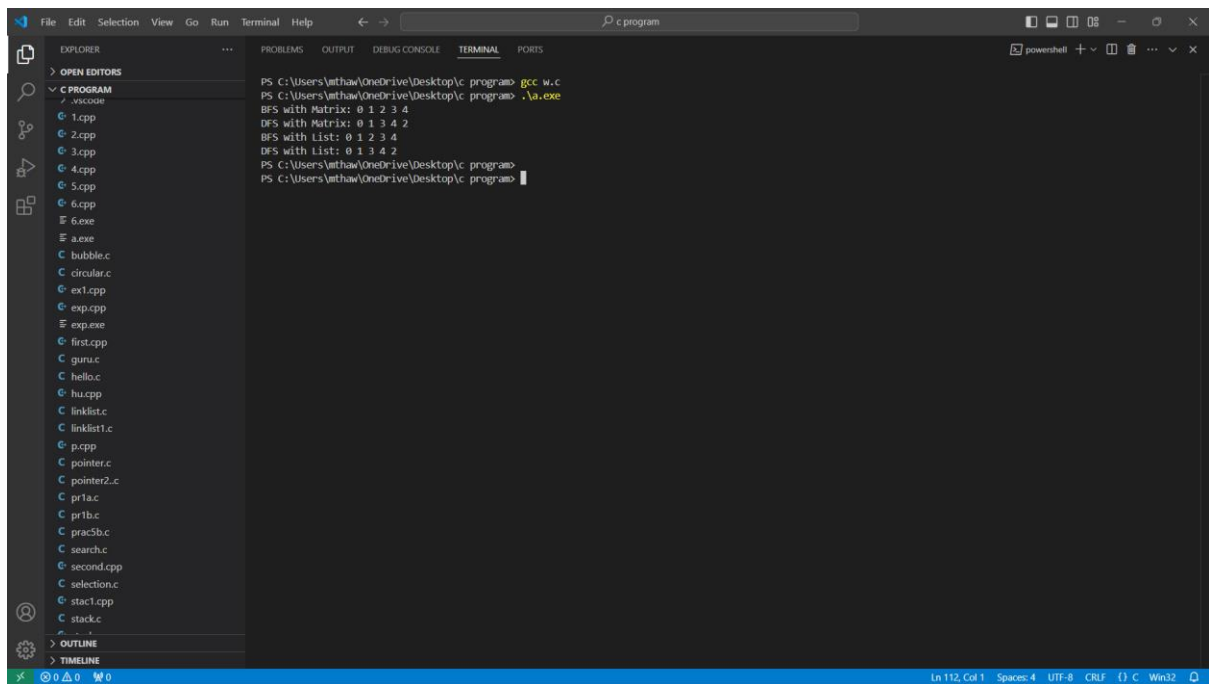
dfs_list(0, visited);

printf("\n");

return 0;
}

```

Output:-



```

PS C:\Users\mithaw\OneDrive\Desktop\c program> gcc w.c
PS C:\Users\mithaw\OneDrive\Desktop\c program> ./a.exe
BFS with Matrix: 0 1 2 3 4
DFS with List: 0 1 3 4 2
BFS with List: 0 1 2 3 4
DFS with List: 0 1 3 4 2
PS C:\Users\mithaw\OneDrive\Desktop\c program>
PS C:\Users\mithaw\OneDrive\Desktop\c program>

```

Github link :- <https://github.com/MayurThaware122/DSA>