Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**Academic Year: 2023 - 24**

**COURSE CODE: DJS22ITL302** **CLASS: S. Y. B. Tech. Sem**

**COURSE NAME: Data Structures Lab** **SAP ID :60003220045**
**Name : Anish Sharma** **DATE: 8/12/2023**

## EXPERIMENT NO. 8

**CO/LO:**

Implement graph traversing techniques

**Objective:**

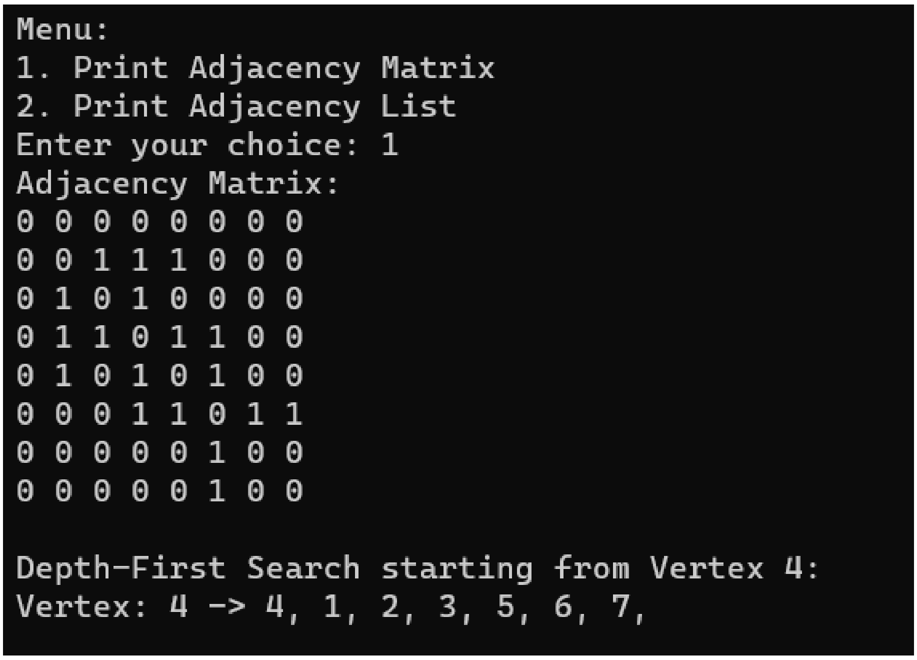
Write a program to implement DFS and BFS for graphs

**Code :**

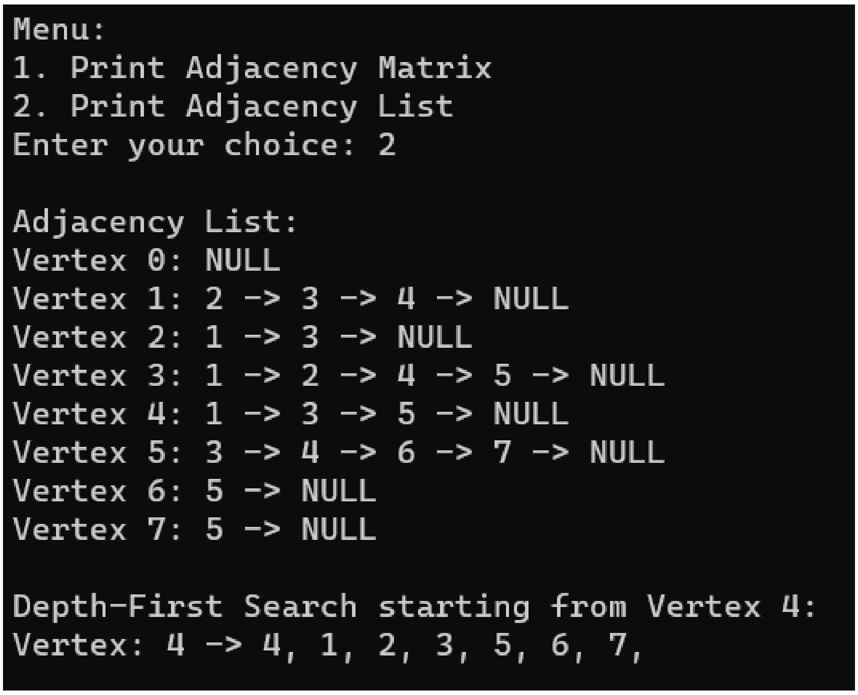
**This is DFS**

```
#include <stdio.h>

void printAdjMatrix(int A[8][8], int n) {
    printf("Adjacency Matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }
}


void printAdjList(int A[8][8], int n) {
    printf("\nAdjacency List:\n");
    for (int i = 0; i < n; i++) {
        printf("Vertex %d: ", i);
        for (int j = 0; j < n; j++) {
            if (A[i][j] == 1) {
                printf("%d -> ", j);
            }
        }
        printf("NULL\n");
    }
}

void DFS(int u, int A[8][8], int n, int visited[8]) {
    if (visited[u] == 0) {
        printf("%d, ", u);
        visited[u] = 1;
        for (int v = 0; v < n; v++) {
            if (A[u][v] == 1 && visited[v] == 0) {
                DFS(v, A, n, visited);
            }
        }
    }
}

int main() {
    int A[8][8] = {{0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 1, 1, 1, 0, 0, 0},
                   {0, 1, 0, 1, 0, 0, 0, 0},
                   {0, 1, 1, 0, 1, 1, 0, 0},
                   {0, 1, 0, 1, 0, 1, 0, 0},
                   {0, 0, 0, 1, 1, 0, 1, 1},
                   {0, 0, 0, 0, 0, 1, 0, 0},
                   {0, 0, 0, 0, 0, 1, 0, 0}};

    int choice;
    printf("Menu:\n");
    printf("1. Print Adjacency Matrix\n");
    printf("2. Print Adjacency List\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            printAdjMatrix(A, 8);
            break;
        case 2:
            printAdjList(A, 8);
            break;
        default:
            printf("Invalid choice\n");
            break;
    }

    int visited[8] = {0};

    printf("\nDepth-First Search starting from Vertex 4:\nVertex: 4 -> ");
    DFS(4, A, 8, visited);
    printf("\n");

    return 0;
}
```

**Output :**



```
Menu:
1. Print Adjacency Matrix
2. Print Adjacency List
Enter your choice: 1
Adjacency Matrix:
0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0
0 1 0 1 0 0 0 0
0 1 1 0 1 1 0 0
0 1 0 1 0 1 0 0
0 0 0 1 1 0 1 1
0 0 0 0 0 1 0 0
0 0 0 0 0 1 0 0

Depth-First Search starting from Vertex 4:
Vertex: 4 -> 4, 1, 2, 3, 5, 6, 7,
```



```
Menu:
1. Print Adjacency Matrix
2. Print Adjacency List
Enter your choice: 2

Adjacency List:
Vertex 0: NULL
Vertex 1: 2 -> 3 -> 4 -> NULL
Vertex 2: 1 -> 3 -> NULL
Vertex 3: 1 -> 2 -> 4 -> 5 -> NULL
Vertex 4: 1 -> 3 -> 5 -> NULL
Vertex 5: 3 -> 4 -> 6 -> 7 -> NULL
Vertex 6: 5 -> NULL
Vertex 7: 5 -> NULL

Depth-First Search starting from Vertex 4:
Vertex: 4 -> 4, 1, 2, 3, 5, 6, 7,
```

**This is BFS**

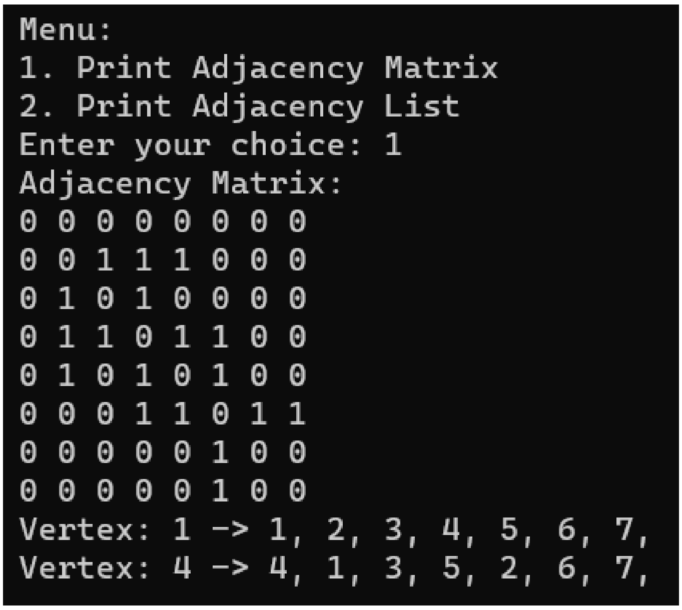
**Code :**

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

void printAdjMatrix(int A[][8], int n) {
    printf("Adjacency Matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }
}

void printAdjList(int A[][8], int n) {
    printf("\nAdjacency List:\n");
    for (int i = 0; i < n; i++) {
        printf("Vertex %d: ", i);
        for (int j = 0; j < n; j++) {
            if (A[i][j] == 1) {
                printf("%d -> ", j);
            }
        }
        printf("NULL\n");
    }
}

void BFS(int vtx, int A[][8], int n) {
    int *visited = (int *)calloc(n, sizeof(int));
    if (visited == NULL) {
        printf("Memory allocation failed.\n");
        exit(EXIT_FAILURE);
    }

    printf("%d, ", vtx); // Visit vertex
    visited[vtx] = 1;

    int *queue = (int *)malloc(n * sizeof(int));
    if (queue == NULL) {
        printf("Memory allocation failed.\n");
        free(visited);
        exit(EXIT_FAILURE);
    }

    int front = 0, rear = 0;
    queue[rear++] = vtx;

    while (front < rear) {
        int u = queue[front++]; // Vertex u for exploring
        for (int v = 0; v < n; v++) { // Adjacent vertices of vertex u
            if (A[u][v] == 1 && visited[v] == 0) { // Adjacent vertex and not visited
                printf("%d, ", v); // Visit vertex
                visited[v] = 1;
                queue[rear++] = v;
}}}
    free(visited);
    free(queue);
    printf("\n");
}
int main() {
```

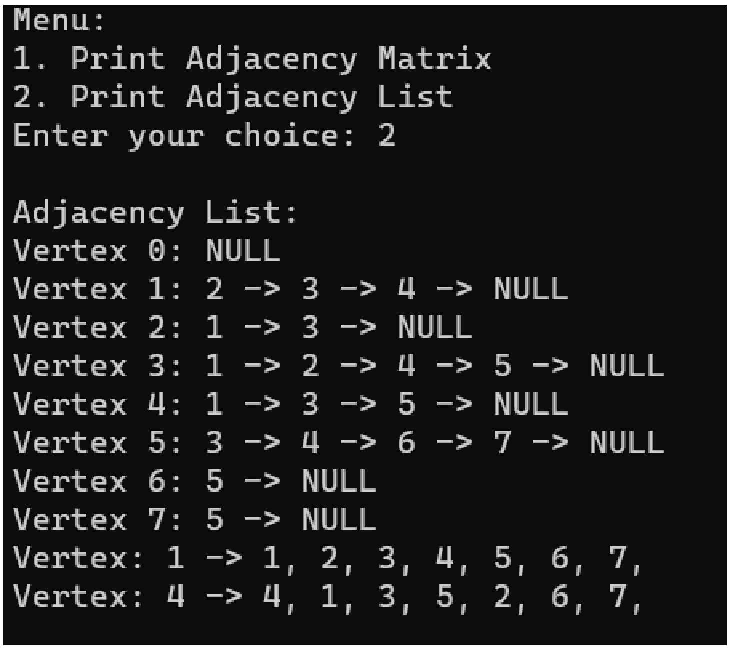
```
int A[8][8] = {{0, 0, 0, 0, 0, 0, 0, 0},
               {0, 0, 1, 1, 1, 0, 0, 0},
               {0, 1, 0, 1, 0, 0, 0, 0},
               {0, 1, 1, 0, 1, 1, 0, 0},
               {0, 1, 0, 1, 0, 1, 0, 0},
               {0, 0, 0, 1, 1, 0, 1, 1},
               {0, 0, 0, 0, 0, 1, 0, 0},
               {0, 0, 0, 0, 0, 1, 0, 0}}
int choice;
printf("Menu:\n");
printf("1. Print Adjacency Matrix\n");
printf("2. Print Adjacency List\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch (choice) {
    case 1:
        printAdjMatrix(A, 8);
        break;
    case 2:
        printAdjList(A, 8);
        break;
    default:
        printf("Invalid choice\n");
        break;
}
printf("Vertex: 1 -> ");
BFS(1, A, 8);
printf("Vertex: 4 -> ");
BFS(4, A, 8);
return 0;
```

}

**Output :**



```
Menu:
1. Print Adjacency Matrix
2. Print Adjacency List
Enter your choice: 1
Adjacency Matrix:
0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0
0 1 0 1 0 0 0 0
0 1 1 0 1 1 0 0
0 1 0 1 0 1 0 0
0 0 0 1 1 0 1 1
0 0 0 0 0 1 0 0
0 0 0 0 0 1 0 0
Vertex: 1 -> 1, 2, 3, 4, 5, 6, 7,
Vertex: 4 -> 4, 1, 3, 5, 2, 6, 7,
```



```
Menu:
1. Print Adjacency Matrix
2. Print Adjacency List
Enter your choice: 2

Adjacency List:
Vertex 0: NULL
Vertex 1: 2 -> 3 -> 4 -> NULL
Vertex 2: 1 -> 3 -> NULL
Vertex 3: 1 -> 2 -> 4 -> 5 -> NULL
Vertex 4: 1 -> 3 -> 5 -> NULL
Vertex 5: 3 -> 4 -> 6 -> 7 -> NULL
Vertex 6: 5 -> NULL
Vertex 7: 5 -> NULL
Vertex: 1 -> 1, 2, 3, 4, 5, 6, 7,
Vertex: 4 -> 4, 1, 3, 5, 2, 6, 7,
```