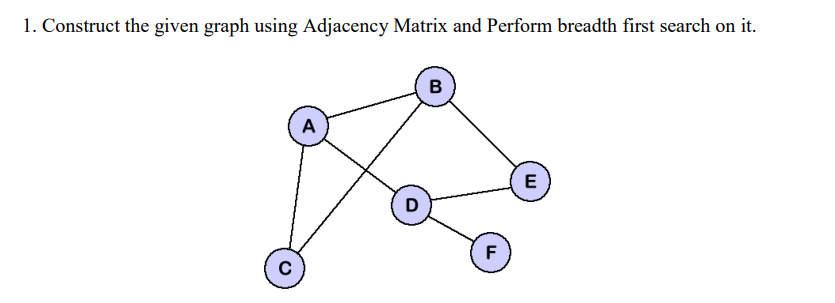
**BCSE202P -**

**Data Structures and Algorithms**

Digital Assignment – 4

**Name:** Dhruv Rajeshkumar Shah

**Registration No –** 21BCE0611



**CODE**

// DA-4

// Dhruv Rajeshkumar Shah

// 21BCE0611

#include <stdio.h>

#include <stdlib.h>

// BFS

void BFS(int n, int a[10][10], int source, int visited[10])

{

    int queue[10], front = -1, rear = -1, i, j;

    queue[++rear] = source;

    visited[source] = 1;

    while (front != rear)

    {

        i = queue[++front];

        printf("%c ", i + 64);

        for (j = 1; j <= n; j++)

        {

            if (a[i][j] == 1 && visited[j] == 0)

            {

                queue[++rear] = j;

                visited[j] = 1;

            }

        }

    }

}

int main()

{

    int n, a[10][10], i, j, source, visited[10];

    char source\_letter;

    printf("Enter the number of vertices: ");

    scanf("%d", &n);

    printf("Enter the adjacency matrix: \n");

    printf("    A B C D E F\n");

    printf("   -------------\n");

    for (i = 1; i <= n; i++)

    {

        printf("%c | ", i + 64);

        for (j = 1; j <= n; j++)

        {

            scanf("%d", &a[i][j]);

        }

    }

    getchar();

    printf("Enter the source vertex: ");

    scanf("%c", &source\_letter);

    source = (int)source\_letter - 64;

    for (i = 1; i <= n; i++)

    {

        visited[i] = 0;

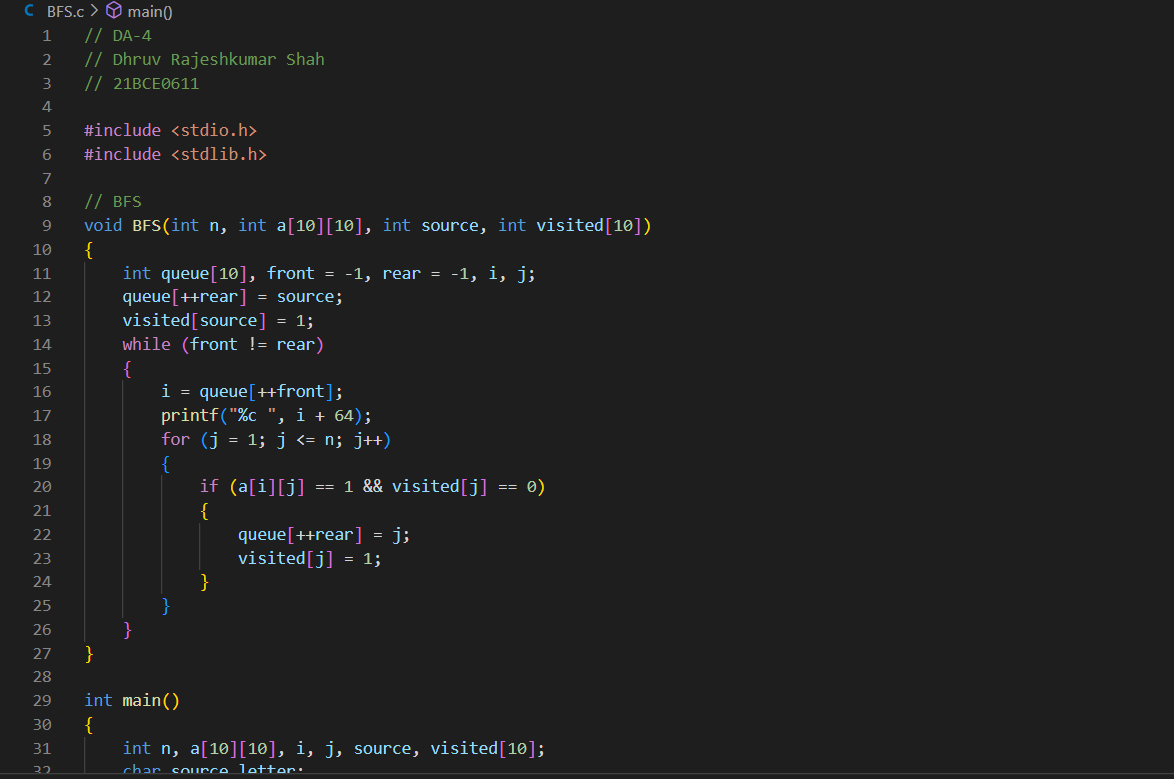
    }

    printf("The BFS traversal is: ");

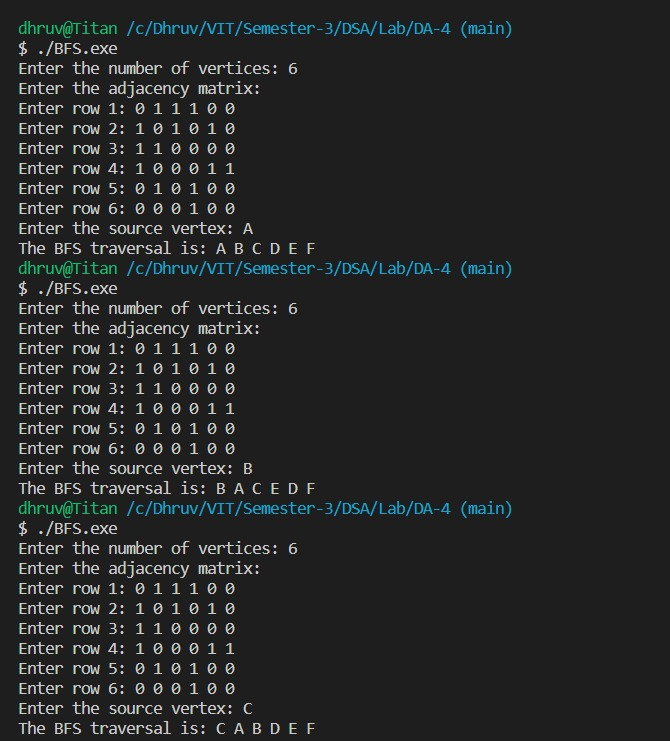
    BFS(n, a, source, visited);

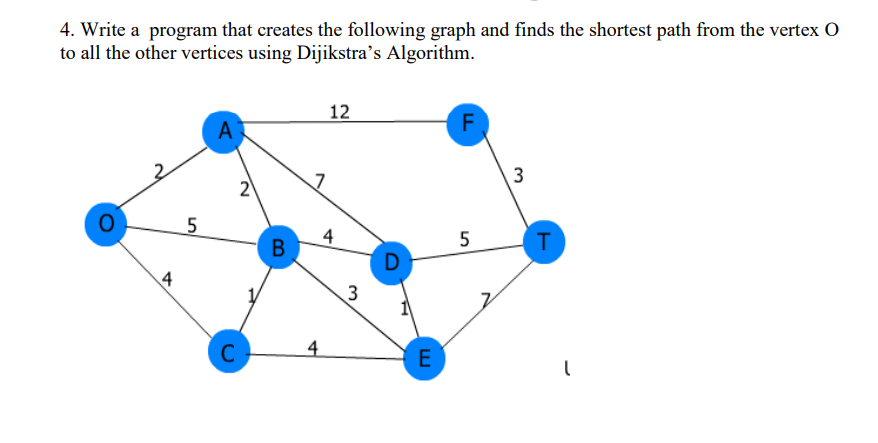
    return 0;

}

**SCREENSHOT**

**OUTPUT**

****



**CODE**

// DA-4

// Dhruv Rajeshkumar Shah

// 21BCE0611

#include <stdio.h>

#include <stdlib.h>

// Index of element in array

int getIndex(char a[], int n, char c)

{

    for (int i = 0; i < n; i++)

    {

        if (a[i] == c)

        {

            return i;

        }

    }

    return -1;

}

// Djitskar's shortest path alaorithm

void dijikstra(int a[10][10], int n, int startnode, char nodes[])

{

    int cost[10][10], distance[10], pred[10];

    int visited[10], count, mindistance, nextnode, i, j;

    for (i = 0; i < n; i++)

        for (j = 0; j < n; j++)

            if (a[i][j] == 0)

                cost[i][j] = 999;

            else

                cost[i][j] = a[i][j];

    for (i = 0; i < n; i++)

    {

        distance[i] = cost[startnode][i];

        pred[i] = startnode;

        visited[i] = 0;

    }

    distance[startnode] = 0;

    visited[startnode] = 1;

    count = 1;

    while (count < n - 1)

    {

        mindistance = 999;

        for (i = 0; i < n; i++)

            if (distance[i] < mindistance && !visited[i])

            {

                mindistance = distance[i];

                nextnode = i;

            }

        visited[nextnode] = 1;

        for (i = 0; i < n; i++)

            if (!visited[i])

                if (mindistance + cost[nextnode][i] < distance[i])

                {

                    distance[i] = mindistance + cost[nextnode][i];

                    pred[i] = nextnode;

                }

        count++;

    }

    for (i = 0; i < n; i++)

    {

        if (i != startnode)

        {

            printf("\nDistance of node %c = %d", nodes[i], distance[i]);

            printf("\nPath = %c", nodes[i]);

            j = i;

            do

            {

                j = pred[j];

                printf("<-%c", nodes[j]);

            } while (j != startnode);

        }

        printf("\n");

    }

}

void main()

{

    int a[10][10], j, n, startnode;

    char source;

    char nodes[] = {'A', 'B', 'C', 'D', 'E', 'F', 'O', 'T'};

    printf("\nEnter the number of vertices: ");

    scanf("%d", &n);

    printf("\nEnter the adjacency matrix:\n");

    printf("    A B C D E F O T\n");

    printf("   -----------------\n");

    for (int i = 0; i < n; i++)

    {

        printf("%c | ", nodes[i]);

        for (int j = 0; j < n; j++)

        {

            scanf("%d", &a[i][j]);

        }

    }

    getchar();

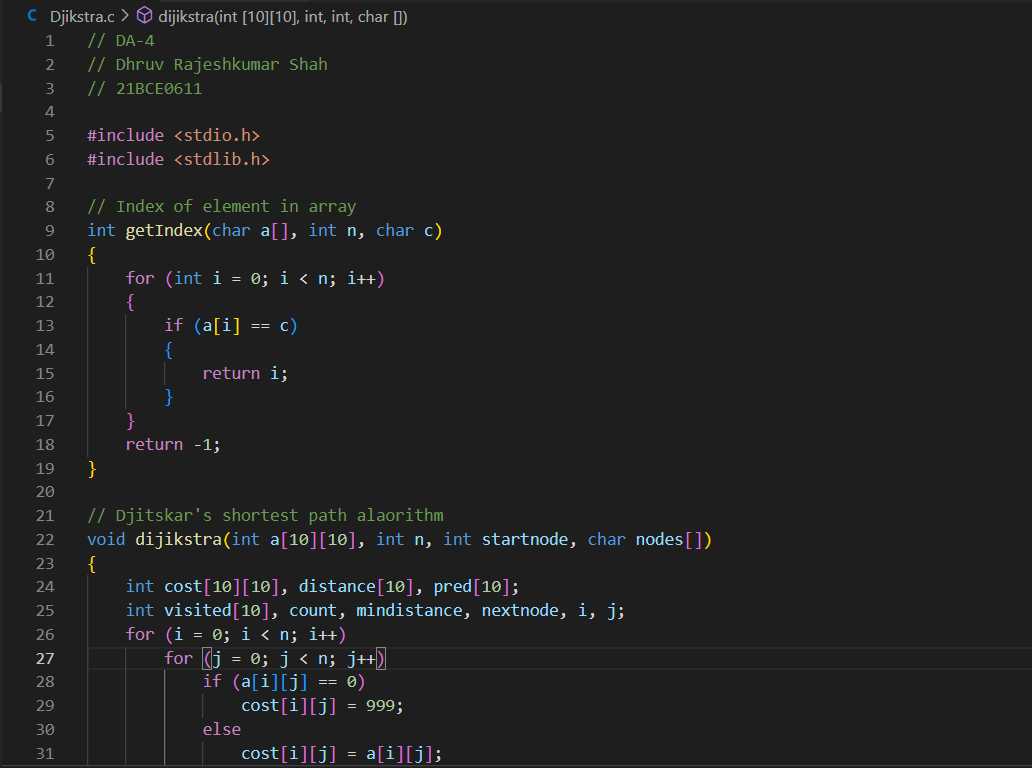
    printf("\nEnter the source: ");

    scanf("%c", &source);

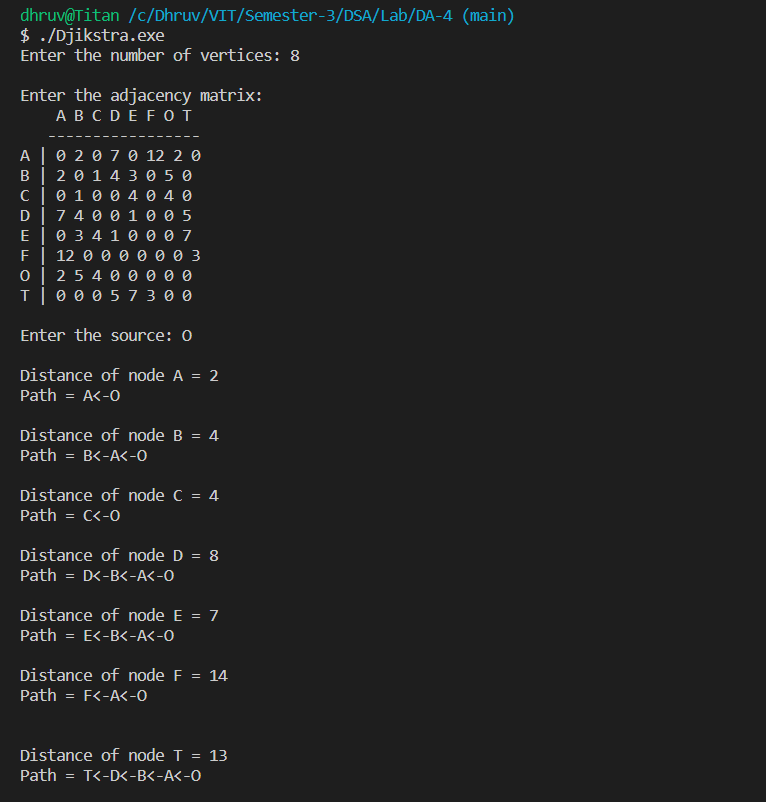
    startnode = getIndex(nodes, n, source);

    dijikstra(a, n, startnode, nodes);

}

**SCREENSHOT**

**OUTPUT**

****