

## **PROGRAM-1**

**AIM: Write a program to perform DFS**

```
#include<stdio.h>

void dfs(int);

int g[10][10],visited[10], n;

void main()
{
    int i, j;

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

    scanf ("%d", &n);

    printf ("\n enter the adjacnecy matrix:");

    for(i = 0; i < n; ++i)
    {
        for(j = 0; j < n; ++j)
        {
            printf("\n edge exist between vertices %d-%d :", i, j);

            scanf("%d", &g[i][j]);
        }
    }

    for(i = 0; i < n; ++i)
    {
        visited[i] = 0;
    }

    dfs(0);
}

void dfs(int i)
{
    int j;

    printf ("\n %d", i);

    visited[i] = 1;

    for (j = 0; j < n; j++)
```

```

{
if(!visited[j] && g[i][j] == 1)
{
dfs(j);
}
}
}

```

### OUTPUT:

The screenshot shows the Visual Studio Code interface with the following content:

- EXPLORER:** A list of files and folders in the project, including `question1.c`, `dfs.c`, `launch.json`, `.dist`, `.vscode`, `css`, `form`, `frames`, `registration`, `resume`, `a.exe`, `color change.html`, `dfs.c` (selected), `doubly.c`, `electricity.php`, `formvalidation.html`, `pgm1_CO1.html`, `qp.pdf`, `question1.c`, `series1.c`, `series1.exe`, `test.c`, `validation.html`, `validation1.php`, and `web assignment1.docx`.
- TERMINAL:** The output of the program execution:
 

```

PS E:\mca s1\web development> gcc dfs.c
PS E:\mca s1\web development> ./a
enter the number of vertices:6

enter the adjacency matrix:
edge exist between vertices 0-0 :0
edge exist between vertices 0-1 :1
edge exist between vertices 0-2 :1
edge exist between vertices 0-3 :0
edge exist between vertices 0-4 :0
edge exist between vertices 0-5 :0
edge exist between vertices 1-0 :0
edge exist between vertices 1-1 :0
edge exist between vertices 1-2 :0
edge exist between vertices 1-3 :1
edge exist between vertices 1-4 :1
edge exist between vertices 1-5 :0
edge exist between vertices 2-0 :0
edge exist between vertices 2-1 :0
edge exist between vertices 2-2 :0
edge exist between vertices 2-3 :0

```

Visual Studio Code interface showing the Explorer, Search, and Run and Debug views. The Explorer view displays the file structure of the 'dfsc' project, including files like 'question1.c', 'dfsc.c', 'launch.json', and various HTML and PDF files. The Search view shows 16 results for the pattern 'edge exist between vertices'. The Run and Debug view shows the output of the program, which is a list of 16 statements: 'edge exist between vertices 2-3 :0', 'edge exist between vertices 2-4 :0', 'edge exist between vertices 2-5 :1', 'edge exist between vertices 3-0 :0', 'edge exist between vertices 3-1 :0', 'edge exist between vertices 3-2 :0', 'edge exist between vertices 3-3 :0', 'edge exist between vertices 3-4 :0', 'edge exist between vertices 3-5 :0', 'edge exist between vertices 4-0 :0', 'edge exist between vertices 4-1 :0', 'edge exist between vertices 4-2 :0', 'edge exist between vertices 4-3 :0', 'edge exist between vertices 4-4 :0', 'edge exist between vertices 4-5 :0', 'edge exist between vertices 5-0 :0', and 'edge exist between vertices 5-1 :0'. The status bar at the bottom indicates the current file is 'dfsc.c' and the active file is 'dfsc.c'.

Visual Studio Code interface showing the Explorer, Search, and Run and Debug views. The Explorer view displays the file structure of the 'dfsc' project, including files like 'question1.c', 'dfsc.c', 'launch.json', and various HTML and PDF files. The Search view shows 16 results for the pattern 'edge exist between vertices'. The Run and Debug view shows the output of the program, which is a list of 16 statements: 'edge exist between vertices 3-3 :0', 'edge exist between vertices 3-4 :0', 'edge exist between vertices 3-5 :0', 'edge exist between vertices 4-0 :0', 'edge exist between vertices 4-1 :0', 'edge exist between vertices 4-2 :0', 'edge exist between vertices 4-3 :0', 'edge exist between vertices 4-4 :0', 'edge exist between vertices 4-5 :0', 'edge exist between vertices 5-0 :0', 'edge exist between vertices 5-1 :0', 'edge exist between vertices 5-2 :0', 'edge exist between vertices 5-3 :0', 'edge exist between vertices 5-4 :0', 'edge exist between vertices 5-5 :0', and a list of numbers: '0', '1', '3', '4', '2', '5'. The status bar at the bottom indicates the current file is 'dfsc.c' and the active file is 'dfsc.c'.

## **PROGRAM-2**

**AIM: Write a program to perform BFS**

```
#include<stdio.h>

int a[20][20],q[20],visited[20],n,i,j,f=0,r=-1;

void bfs(int v);

void main() {
    int v; //call the value of starting vertex
    printf("\n Enter the number of vertices:");
    scanf("%d",&n);
    printf("enter the adjecency matrix");
    for (i=0;i<n;i++)
    {
        for (j=0;j<n;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    printf("\n Enter the starting vertex:");
    scanf("%d",&v);
    for (i=0;i<n;i++)
    {
        q[i]=0;
        visited[i]=0;
    }
    bfs(v);
    printf("\n The node which are reachable are:\n");
    for (i=1;i<=n;i++)
    {
        if(visited[i])
        {
```

```

        printf("%d\t",i);
    }
}

void bfs(int v)
{
    for (i=0;i<n;i++)
    {
        if(a[v][i] && !visited[i])
            q[++r]=i;
    }
    if(f<=r)
    {
        visited[q[f]]=1;
        bfs(q[f++]);
    }
}

```

### **OUTPUT:**

```

Enter the number of vertices:4
enter the adjacency matrix  0 1 0 1
1 0 1 0
0 1 0 1
1 0 1 0

Enter the starting vertex:0

The node which are reachable are:
1      2      3

```

### **PROGRAM-3**

**AIM: Write a program to implement PRIMS**

```
#include<stdio.h>

#include<stdbool.h>

#define infinity 1000

//#define v 5

int graph[20][20];

int v;

/*int graph[v][v] = {

    {0, 9, 75, 0, 0},

    {9, 0, 95, 19, 42},

    {75, 95, 0, 51, 66},

    {0, 19, 51, 0, 31},

    {0, 42, 66, 31, 0}};

*/

/*void display(){

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

        for(int j=0;j<v;j++){

            printf("%d",graph[i][j]);

        }

    }

}*/

void mst(bool span[]){

    int edge_count=0,total=0,x,y;

    span[0]=1;

    printf("\nEdge : Weight\n");

    while(edge_count<v-1){
```

```

        int cost=infinity;
        for(int i=0;i<v;i++){
            if(span[i]){
                for(int j=0;j<v;j++){
                    if(!span[j] && graph[i][j]){
                        if(graph[i][j] < cost){
                            cost=graph[i][j];
                            x=i;
                            y=j;
                        }
                    }
                }
            }
        }
        printf("%d - %d : %d\n", x, y, graph[x][y]);
        total+=graph[x][y];
        span[y]=1;
        edge_count++;
    }
    printf("\nTotal Cost=%d\n",total);
}

```

```

void main(){
    printf("\nEnter the number of vertices ");
    scanf("%d",&v);
    printf("\nEnter the Adjacency Matrix \n");
    for(int i=0;i<v;i++){
        for(int j=0;j<v;j++){
            printf("Enter the edge weight of %d to %d ",i,j);
            scanf("%d",&graph[i][j]);
        }
    }
}

```

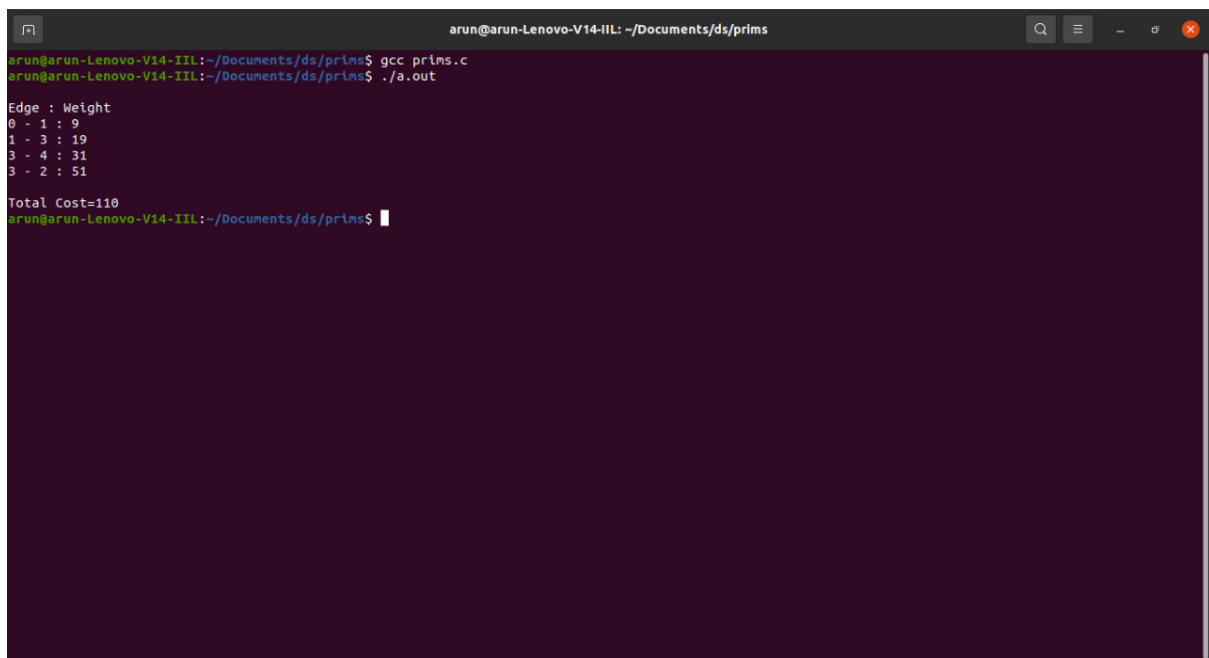
```
}

for(int i=0;i<v;i++){
    graph[i][i]=0;
}

bool span[v];
for(int i=0;i<v;i++){
    span[i]=0;
}

mst(span);
}
```

### OUTPUT:



```
arun@arun-Lenovo-V14-IIL: ~/Documents/ds/prims
arun@arun-Lenovo-V14-IIL:~/Documents/ds/prims$ gcc prims.c
arun@arun-Lenovo-V14-IIL:~/Documents/ds/prims$ ./a.out
Edge : Weight
0 - 1 : 9
1 - 3 : 19
3 - 4 : 31
3 - 2 : 51
Total Cost=110
arun@arun-Lenovo-V14-IIL:~/Documents/ds/prims$
```



#### **PROGRAM-4**

**AIM:** Write a program to implement KRUSKALS

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

/*int graph[v][v]={{0,2,3,0},

                {2,0,2,1},

                {3,2,0,4},

                {0,1,4,0}}

*/

int i,j,a,b,u,v,n,ne=1;

int min,cost=0,graph[9][9],parent[9];

int find(int);

int uni(int,int);

void main()

{

    printf("\nEnter the no. of vertices:");

    scanf("%d",&n);

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

    for(i=1;i<=n;i++)

    {

        for(j=1;j<=n;j++)

        {

            printf("Enter the edge weight of %d to %d ",i,j);

            scanf("%d",&graph[i][j]);

            if(graph[i][j]==0)

                graph[i][j]=999;

        }

    }
```

```

}

printf("The edges of Minimum cost Spanning Tree are\n");

while(ne < n)
{
    min=999;
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            if(graph[i][j] < min)
            {
                min=graph[i][j];
                a=u=i;
                b=v=j;
            }
        }
    }
    u=find(u);
    v=find(v);
    if(uni(u,v))
    {
        printf("edge (%d,%d) =%d\n",a,b,min);
        cost +=min;
        ne++;
    }
    graph[a][b]=graph[b][a]=999;
}

printf("\nMinimum cost = %d\n",cost);
}

```

```

int find(int i)
{
    while(parent[i])
    {
        i=parent[i];
    }
    return i;
}

int uni(int i,int j)
{
    if(i!=j)
    {
        parent[j]=i;
        return 1;
    }
    return 0;
}

```

### **OUTPUT:**

```

Enter the no. of vertices:4

Enter the cost adjacency matrix:
Enter the edge weight of 1 to 1 0
Enter the edge weight of 1 to 2 2
Enter the edge weight of 1 to 3 3
Enter the edge weight of 1 to 4 0
Enter the edge weight of 2 to 1 2
Enter the edge weight of 2 to 2 0
Enter the edge weight of 2 to 3 2
Enter the edge weight of 2 to 4 1
Enter the edge weight of 3 to 1 3
Enter the edge weight of 3 to 2 2
Enter the edge weight of 3 to 3 0
Enter the edge weight of 3 to 4 4
Enter the edge weight of 4 to 1 0
Enter the edge weight of 4 to 2 1
Enter the edge weight of 4 to 3 4
Enter the edge weight of 4 to 4 0

The edges of Minimum cost Spanning Tree are
edge (2,4) =1
edge (1,2) =2
edge (2,3) =2

Minimum cost = 5

```

## **PROGRAM-5**

**AIM: Write a program to implement Topological Sorting**

```
#include <stdio.h>

void main()
{
    int n = 0;

    printf("enter how many vertex are there - ");

    scanf("%d", & n);

    int a[n][n], tp[n], f[n], x = 0;

    printf("\nEnter 1 if an edge exists or otherwise\n");

    for (int i = 1; i <= n; i++) {

        f[i - 1] = 0;

        for (int j = 1; j <= n; j++) {

            printf("Does an edge exists from %d to %d - ", i, j);

            scanf("%d", & a[i - 1][j - 1]);

        }

    }

    while (x < n) {

        int in = 0, ind[n];

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

            for (int j = 0; j < n; j++) {

                if (a[j][i] == 1) {

                    in ++;

                }

            }

            ind[i] = in ; in = 0;

        }

        int t = 0;

        for (t = 0; t < n; t++) {

            if (ind[t] == 0 && f[t] == 0) {
```

```

    f[t] = 1;

    printf("%d ", t + 1);

    break;
}
}

printf("\n");

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

    if (a[t][i] == 1) {

        a[t][i] = 0;

    }

}

x++;

}

}

```

#### **OUTPUT:**

```

enter how many vertex are there - 4

Enter 1 if an edge exists or otherwise
Does an edge exists from 1 to 1 - 0
Does an edge exists from 1 to 2 - 1
Does an edge exists from 1 to 3 - 1
Does an edge exists from 1 to 4 - 0
Does an edge exists from 2 to 1 - 0
Does an edge exists from 2 to 2 - 0
Does an edge exists from 2 to 3 - 0
Does an edge exists from 2 to 4 - 1
Does an edge exists from 3 to 1 - 0
Does an edge exists from 3 to 2 - 0
Does an edge exists from 3 to 3 - 0
Does an edge exists from 3 to 4 - 1
Does an edge exists from 4 to 1 - 0
Does an edge exists from 4 to 2 - 0
Does an edge exists from 4 to 3 - 0
Does an edge exists from 4 to 4 - 0
Topological Sort : -
1
2
3
4

```