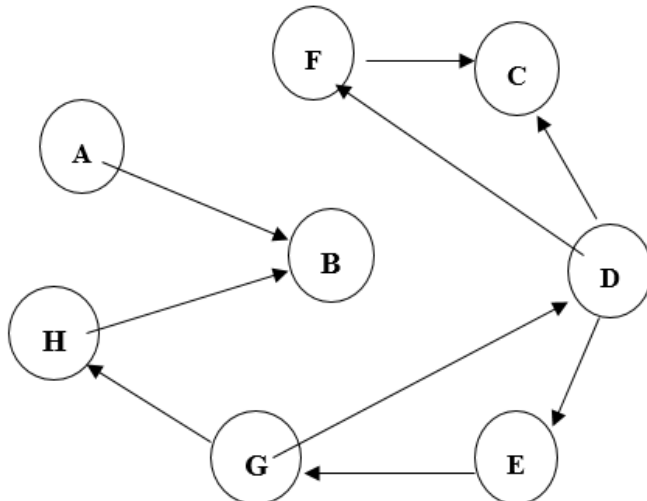


## CSE2001 (Data Structures & Algorithms) Lab-9

**KHAN MOHD OWAIS RAZA**

**20BCD7138**

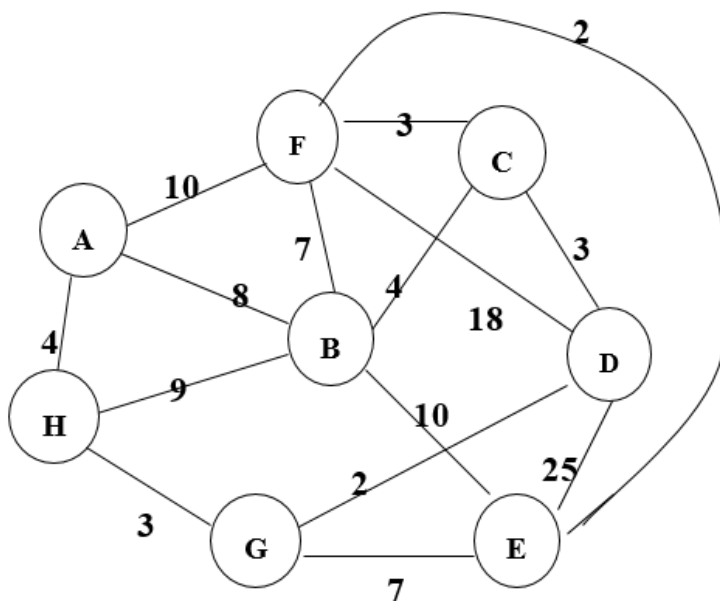
1. Write a Program to implement DFS algorithm and print the DFS sequence for below graph start with node D.



2. Write a Program to implement BFS Algorithm and print the BFS sequence start with node A.

Challenging:

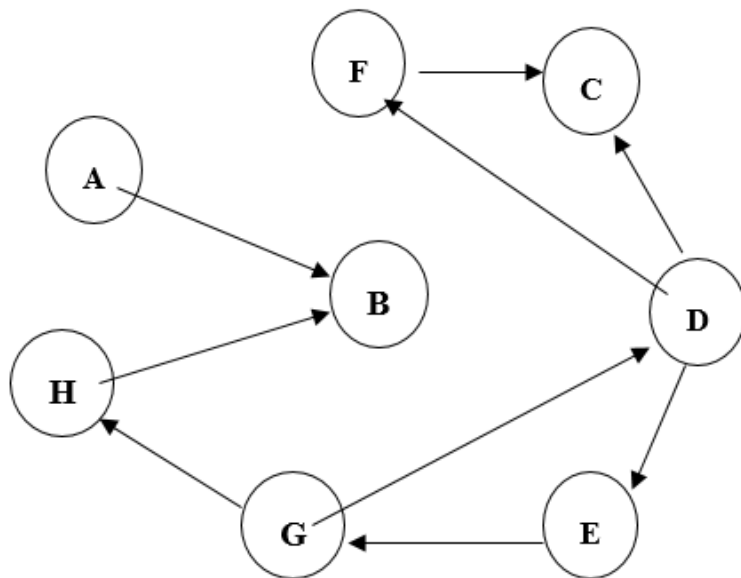
1. Write a Program to check the connectivity of a graph using BFS.
2. Write a program to implement Dijkstra's algorithm
3. Write a Program to Implement Prim's Algorithm and find the minimum spanning tree for the given graph



4. Write a Program to implement Kruskal's Algorithm and find the minimum spanning tree for the above graph

Q.1]

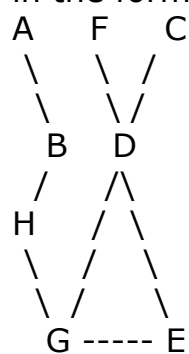
Write a Program to implement DFS algorithm and print the DFS sequence for below graph start with node D.



**C code and output:-**

```
// KHAN MOHD OWAIS RAZA  
// 20BCD7138  
// CSE2001 Lab-9  
// Question-1  
#include <stdio.h>  
#include <stdlib.h>  
/*
```

According to question, the graph is in the form of alphabets.



Let it be:

A = 0, B = 1, C = 2

D = 3, E = 4, F = 5

G = 6, H = 7

As per the graph,  
number of edges = 9,  
number of vertices = 8;

Edges are:

0 -> 1 (A -> B)

1 -> 1 (H -> B)

6 -> 7 (G -> H)

4 -> 3 (E -> G)

3 -> 4 (D -> E)

6 -> 3 (G -> D)

3 -> 2 (D -> C)

3 -> 5 (D -> F)

5 -> 2 (F -> C)

\*/

```
int source,V,E,time,visited[20],G[20][20];
```

```
void DFS(int i){
```

```
int j;
```

```
visited[i]=1;
```

```
printf(" %d ",i+1);
```

```
for(j=0;j<V;j++){
```

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

```
DFS(j);
```

```
}}
```

```
int main(){
```

```
int i,j,v1,v2;
```

```
printf("Enter the no of edges:");
```

```
scanf("%d",&E);
```

```
printf("Enter the no of vertices:");
```

```
scanf("%d",&V);
```

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

```
for(j=0;j<V;j++){
```

```
G[i][j]=0;
```

```
}
```

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

```
printf("Enter the edges: ");
```

```
scanf("%d%d",&v1,&v2);
```

```
G[v1-1][v2-1]=1;
```

```
}
```

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

```
for(j=0;j<V;j++){
```

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

```
printf("\n");
```

```
}
```

```
printf("Enter the source: ");
```

```
scanf("%d",&source);
```

```
DFS(source-1);
```

```
return 0;
```

```
}
```

Question1.c

Question2.c

```
1 // KHAN MOHD OWAIS RAZA
2 // 20BCD7138
3 // CSE2001 Lab-9
4 // Question-1
5 #include <stdio.h>
6 #include <stdlib.h>
7 /*
8 According to question, the graph is
9 in the form of alphabets.
10 Let it be:
11 A = 0
12 B = 1
13 C = 2
14 D = 3
15 E = 4
16 F = 5
17 G = 6
18 H = 7
19 As per the graph,
20 number of edges = 9,
21 number of vertices = 8;
22 Edges are:
23 0 -> 1 (A -> B)
24 7 -> 1 (H -> B)
25 6 -> 7 (G -> H)
26 4 -> 6 (E -> G)
27 3 -> 4 (D -> E)
28 6 -> 3 (G -> D)
29 3 -> 2 (D -> C)
30 3 -> 5 (D -> F)
31 5 -> 2 (F -> C)
32 */
33 int source,V,E,time,visited[20],G[20][20];
34 void DFS(int i){
35     int j;
36     visited[i]=1;
37     printf(" %d ",i+1);
38     for(j=0;j<V;j++){
39         if(G[i][j]==1&&visited[j]==0)
40             DFS(j);
41     }
42 }
43 int main(){
44     int i,j,v1,v2;
45     printf("Enter the no of edges:");
46     scanf("%d",&E);
47     printf("Enter the no of vertices:");
48     scanf("%d",&V);
49     for(i=0;i<V;i++){
50         for(j=0;j<V;j++){
51             G[i][j]=0;
52         }
53     }
54     for(i=0;i<E;i++){
```

```

53 printf("Enter the edges: ");
54 scanf("%d%d",&v1,&v2);
55 G[v1-1][v2-1]=1;
56 }
57 for(i=0;i<V;i++){
58     for(j=0;j<V;j++){
59         printf(" %d ",G[i][j]);
60         printf("\n");
61     }
62     printf("Enter the source: ");
63     scanf("%d",&source);
64     DFS(source-1);
65     return 0;
66 }
67

```

C:\Users\Owais\Desktop\Question1.exe

Enter the no of edges:9

Enter the no of vertices:8

Enter the edges: 0 1

Enter the edges: 3 2

Enter the edges: 3 4

Enter the edges: 3 5

Enter the edges: 4 6

Enter the edges: 5 2

Enter the edges: 6 3

Enter the edges: 6 7

Enter the edges: 7 1

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 1 0 1 1 0 0 0

0 0 0 0 0 1 0 0

0 1 0 0 0 0 0 0

0 0 1 0 0 0 1 0

1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

Enter the source: 3

3 2 4 6 7 1 5

-----  
Process exited after 80.08 seconds with return value 0

Press any key to continue . . .

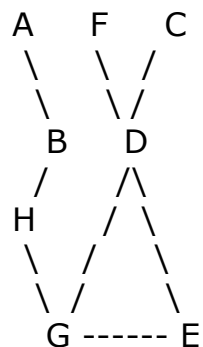
Q.2]

Write a Program to implement BFS Algorithm and print the BFS sequence start with node A.

**C code and output :-**

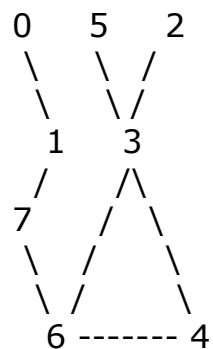
```
// KHAN MOHD OWAIS RAZA
// 20BCD7138
// CSE2001 Lab-9
// Question-2
/*
```

According to question, the graph is in the form of alphabets.



Let it be:

A = 0, B = 1, C = 2  
D = 3, E = 4, F = 5  
G = 6, H = 7



As per the graph,  
number of edges = 9,  
number of vertices = 8;

Edges are:

0 -> 1 (A -> B)  
7 -> 1 (H -> B)  
6 -> 7 (G -> H)  
4 -> 6 (E -> G)  
3 -> 4 (D -> E)  
6 -> 3 (G -> D)  
3 -> 2 (D -> C)

```

3 -> 5 (D -> F)
5 -> 2 (F -> C)
*/
#include<stdlib.h>
#define MAX 100
#define initial 1
#define waiting 2
#define visited 3
int n;
int adj[MAX][MAX];
int state[MAX];
void create_graph();
void BF_Traversal();
void BFS(int v);
int queue[MAX], front = -1, rear = -1;
void insert_queue(int vertex);
int delete_queue();
int isEmpty_queue();
int main(){
create_graph();
BF_Traversal();
return 0;
}
void BF_Traversal()
{
int v;
for(v=0; v<n; v++)
state[v] = initial;
printf("Enter starting vertex: \n");
scanf("%d", &v);
BFS(v);
}
void BFS(int v){
int i;
insert_queue(v);
state[v] = waiting;
while(!isEmpty_queue()){
v = delete_queue( );
printf("%d ",v);
state[v] = visited;
for(i=0; i<n; i++){
if(adj[v][i] == 1 && state[i] == initial){
insert_queue(i);
state[i] = waiting;
}}}
printf("\n");
}

```

```

void insert_queue(int vertex){
if(rear == MAX-1)
printf("Queue Overflow\n");
else{
if(front == -1)
front = 0;
rear = rear+1;
queue[rear] = vertex ;
}}
int isEmpty_queue(){
if(front == -1 || front > rear)
return 1;
else
return 0;
}
int delete_queue(){
int delete_item;
if(front == -1 || front > rear){
printf("Queue Underflow\n");
exit(1);
}
delete_item = queue[front];
front = front+1;
return delete_item;
}
void create_graph(){
int count,max_edge,origin,destin;
printf("Enter number of vertices : ");
scanf("%d",&n);
max_edge = n*(n-1);
for(count=1; count<=max_edge; count++){
printf("Enter edge %d (or type 0 0 to stop) : ",count);
scanf("%d %d",&origin,&destin);
if((origin == 0) && (destin == 0))
break;
if(origin>=n || destin>=n || origin<0 || destin<0){
printf("Invalid edge!\n");
count--;
}
else{
adj[origin][destin] = 1;
}}
}
}

```



```

1 // KHAN MOHD OWAIS RAZA
2 // 20BCD7138
3 // CSE2001 Lab-9
4 // Question-2
5 /*
6 According to question, the graph is
7 in the form of alphabets.
8 A      F      C
9  \    / \    /
10   \  /   \  /
11    B     D
12   / \   / \
13  H   /   \
14   \  /   \
15    G ----- E
16
17 Let it be:
18 A = 0, B = 1, C = 2
19 D = 3, E = 4, F = 5
20 G = 6, H = 7
21
22 0      5      2
23  \    / \    /
24   \  /   \  /
25    1     3
26   / \   / \
27  7   /   \
28   \  /   \
29    6 ----- 4
30
31
32 As per the graph,
33 number of edges = 9,
34 number of vertices = 8;
35 Edges are:
36 0 -> 1 (A -> B)
37 7 -> 1 (H -> B)
38 6 -> 7 (G -> H)
39 4 -> 6 (E -> G)
40 3 -> 4 (D -> E)
41 6 -> 3 (G -> D)
42 3 -> 2 (D -> C)
43 3 -> 5 (D -> F)
44 5 -> 2 (F -> C)
45 */
46 #include<stdlib.h>
47 #define MAX 100
48 #define initial 1
49 #define waiting 2
50 #define visited 3
51 int n;
52 int adj[MAX][MAX];
53 int state[MAX];
54 void create_graph();
55 void BF_Traversal();

```

```

56 void BFS(int v);
57 int queue[MAX], front = -1, rear = -1;
58 void insert_queue(int vertex);
59 int delete_queue();
60 int isEmpty_queue();
61 □ int main(){
62     create_graph();
63     BF_Traversal();
64     return 0;
65 }
66 void BF_Traversal()
67 □ {
68     int v;
69     for(v=0; v<n; v++)
70         state[v] = initial;
71     printf("Enter starting vertex: \n");
72     scanf("%d", &v);
73     BFS(v);
74 }
75 □ void BFS(int v){
76     int i;
77     insert_queue(v);
78     state[v] = waiting;
79 □ while(!isEmpty_queue()){
80     v = delete_queue( );
81     printf("%d ", v);
82     state[v] = visited;
83 □ for(i=0; i<n; i++){
84 □ if(adj[v][i] == 1 && state[i] == initial){
85     insert_queue(i);
86     state[i] = waiting;
87     }}}
88     printf("\n");
89 }
90 □ void insert_queue(int vertex){
91     if(rear == MAX-1)
92         printf("Queue Overflow\n");
93 □ else{
94     if(front == -1)
95         front = 0;
96     rear = rear+1;
97     queue[rear] = vertex ;
98     }}
99 □ int isEmpty_queue(){
100     if(front == -1 || front > rear)
101         return 1;
102     else
103         return 0;
104 }
105 □ int delete_queue(){
106     int delete_item;
107 □ if(front == -1 || front > rear){
108     printf("Queue Underflow\n");
109     exit(1);

```

```

110 }
111 delete_item = queue[front];
112 front = front+1;
113 return delete_item;
114 }
115 void create_graph(){
116     int count,max_edge,origin,destin;
117     printf("Enter number of vertices : ");
118     scanf("%d",&n);
119     max_edge = n*(n-1);
120     for(count=1; count<=max_edge; count++){
121         printf("Enter edge %d (or type 0 0 to stop) : ",count);
122         scanf("%d %d",&origin,&destin);
123         if((origin == 0) && (destin == 0))
124             break;
125         if(origin>=n || destin>=n || origin<0 || destin<0){
126             printf("Invalid edge!\n");
127             count--;
128         }
129         else{
130             adj[origin][destin] = 1;
131         }
132     }

```

```

C:\Users\Owais\Desktop\Question2.exe
Enter number of vertices : 8
Enter edge 1 (or type 0 0 to stop) : 0 1
Enter edge 2 (or type 0 0 to stop) : 3 2
Enter edge 3 (or type 0 0 to stop) : 3 4
Enter edge 4 (or type 0 0 to stop) : 3 5
Enter edge 5 (or type 0 0 to stop) : 4 6
Enter edge 6 (or type 0 0 to stop) : 5 2
Enter edge 7 (or type 0 0 to stop) : 6 3
Enter edge 8 (or type 0 0 to stop) : 6 7
Enter edge 9 (or type 0 0 to stop) : 7 1
Enter edge 10 (or type 0 0 to stop) : 0 0
Enter starting vertex:
0
0 1

-----
Process exited after 52.07 seconds with return value 0
Press any key to continue . . .

```

### Challenging Q.1]

Write a Program to check the connectivity of a graph using BFS.

#### C code and output :-

```
// KHAN MOHD OWAIS RAZA
// 20BCD7138
// CSE2001 Lab-9
// Challenging Question-1
#include<stdio.h>
int a[20][20], q[20], visited[20], n, i, j, f = 0, r = -1;
void bfs(int v) {
    for (i = 1; i <= n; i++)
        if (a[v][i] && !visited[i])
            q[++r] = i;
            if (f <= r) {
                visited[q[f]] = 1;
                bfs(q[f++]);
            }
}
int main(int argc, char **argv) {
    int v = 1, count = 0;
    printf("\n Enter the number of vertices:");
    scanf("%d", &n);
    for (i = 1; i <= n; i++) {
        q[i] = 0;
        visited[i] = 0;
    }
    printf("\n Enter graph data in matrix form:\n");
    for (i = 1; i <= n; i++)
        for (j = 1; j <= n; j++)
            scanf("%d", &a[i][j]);
    bfs(v);
    for (i = 1; i <= n; i++)
        if (visited[i])
            count++;
    if (count == n)
        printf("\n Graph is connected");
    else
        printf("\n Graph is not connected");
    return 0;
}
```

```

1 // KHAN MOHD OWAIS RAZA
2 // 20BCD7138
3 // CSE2001 Lab-9
4 // Challenging Question-1
5 #include<stdio.h>
6 int a[20][20], q[20], visited[20], n, i, j, f = 0, r = -1;
7 void bfs(int v) {
8     for (i = 1; i <= n; i++)
9         if (a[v][i] && !visited[i])
10             q[++r] = i;
11     if (f <= r) {
12         visited[q[f]] = 1;
13         bfs(q[f++]);
14     }
15 }
16 int main(int argc, char **argv) {
17     int v = 1, count = 0;
18     printf("\n Enter the number of vertices:");
19     scanf("%d", &n);
20     for (i = 1; i <= n; i++) {
21         q[i] = 0;
22         visited[i] = 0;
23     }
24     printf("\n Enter graph data in matrix form:\n");
25     for (i = 1; i <= n; i++)
26         for (j = 1; j <= n; j++)
27             scanf("%d", &a[i][j]);
28     bfs(v);
29     for (i = 1; i <= n; i++)
30         if (visited[i])
31             count++;
32     if (count == n)
33         printf("\n Graph is connected");
34     else
35         printf("\n Graph is not connected");
36     return 0;
37 }

```

C:\Users\Owais\Desktop\Challenging\_Question1.exe

Enter the number of vertices:8

Enter graph data in matrix form:

```

0 1 0 0 0 0 0 0
1 0 0 0 0 0 0 1
0 0 0 1 0 1 0 0
0 0 1 0 1 0 1 0
0 0 0 1 0 0 1 0
0 0 1 1 0 0 0 0
0 0 0 1 1 0 0 1
0 1 0 0 0 0 1 0

```

Graph is connected

-----  
Process exited after 127 seconds with return value 0  
Press any key to continue . . .

Challenging Q.2]

Write a program to implement Dijkstra's algorithm

C program and output :-

```
// KHAN MOHD OWAIS RAZA
// 20BCD7138
// CSE2001 Lab-9
// Challenging Question-2
#include<stdio.h>
#define INFINITY 9999
#define MAX 10
void dijkstra(int G[MAX][MAX],int n,int startnode);
int main(){
int G[MAX][MAX],i,j,n,u;
printf("Enter no. of vertices: ");
scanf("%d",&n);
printf("\nEnter the adjacency matrix:\n");
for(i=0;i<n;i++)
for(j=0;j<n;j++)
scanf("%d",&G[i][j]);
printf("\nEnter the starting node: ");
scanf("%d",&u);
dijkstra(G,n,u);
return 0;
}
void dijkstra(int G[MAX][MAX],int n,int startnode){
int cost[MAX][MAX],distance[MAX],pred[MAX];
int visited[MAX],count,mindistance,nextnode,i,j;
for(i=0;i<n;i++)
for(j=0;j<n;j++)
if(G[i][j]==0)
cost[i][j]=INFINITY;
else
cost[i][j]=G[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=INFINITY;
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%d = %d",i,distance[i]);
printf("\nPath: %d",i);
j=i;
do{
j=pred[j];
printf(" <- %d",j);
}
while(j!=startnode);
}}

```

Challenging\_Question2.c

```

1 // KHAN MOHD OWAIS RAZA
2 // 20BCD7138
3 // CSE2001 Lab-9
4 // Challenging Question-2
5 #include<stdio.h>
6 #define INFINITY 9999
7 #define MAX 10
8 void dijkstra(int G[MAX][MAX],int n,int startnode);
9 int main(){
10     int G[MAX][MAX],i,j,n,u;
11     printf("Enter no. of vertices: ");
12     scanf("%d",&n);
13     printf("\nEnter the adjacency matrix:\n");
14     for(i=0;i<n;i++)
15         for(j=0;j<n;j++)
16             scanf("%d",&G[i][j]);
17     printf("\nEnter the starting node: ");
18     scanf("%d",&u);
19     dijkstra(G,n,u);
20     return 0;
21 }
22 void dijkstra(int G[MAX][MAX],int n,int startnode){
23     int cost[MAX][MAX],distance[MAX],pred[MAX];
24     int visited[MAX],count,mindistance,nextnode,i,j;
25     for(i=0;i<n;i++)
26         for(j=0;j<n;j++)
27         if(G[i][j]==0)

```

```

28 cost[i][j]=INFINITY;
29 else
30 cost[i][j]=G[i][j];
31 for(i=0;i<n;i++){
32 distance[i]=cost[startnode][i];
33 pred[i]=startnode;
34 visited[i]=0;
35 }
36 distance[startnode]=0;
37 visited[startnode]=1;
38 count=1;
39 while(count<n-1){
40 mindistance=INFINITY;
41 for(i=0;i<n;i++)
42 if(distance[i]<mindistance&&!visited[i]){
43 mindistance=distance[i];
44 nextnode=i;
45 }
46 visited[nextnode]=1;
47 for(i=0;i<n;i++)
48 if(!visited[i])
49 if(mindistance+cost[nextnode][i]<distance[i]){
50 distance[i]=mindistance+cost[nextnode][i];
51 pred[i]=nextnode;
52 }
53 count++;
54 }
55 for(i=0;i<n;i++)
56 if(i!=startnode){
57 printf("\nDistance of node%d = %d",i,distance[i]);
58 printf("\nPath: %d",i);
59 j=i;
60 do{
61 j=pred[j];
62 printf(" <- %d",j);
63 }
64 while(j!=startnode);
65 } } |

```

C:\Users\Owais\Desktop\Challenging\_Question2.exe

Enter no. of vertices: 5

Enter the adjacency matrix:

```

0 10 0 30 100
10 0 50 0 0
0 50 0 20 10
30 0 20 0 60
100 0 10 60 0

```

Enter the starting node: 0

Distance of node1 = 10

Path: 1 <- 0

Distance of node2 = 50

Path: 2 <- 3 <- 0

Distance of node3 = 30

Path: 3 <- 0

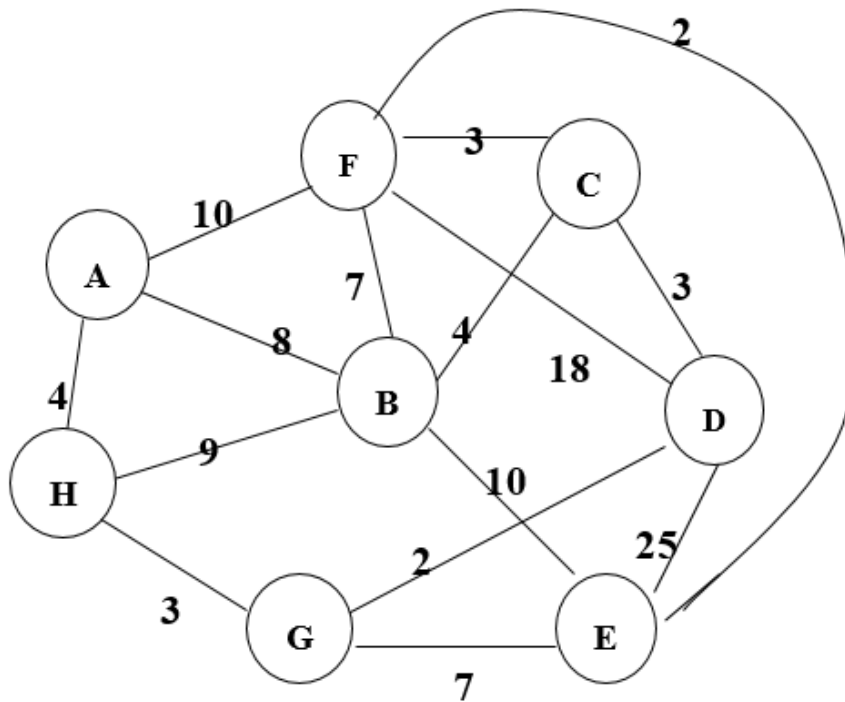
Distance of node4 = 60

Path: 4 <- 2 <- 3 <- 0



Challenging Q.3]

3. Write a Program to Implement Prim's Algorithm and find the minimum spanning tree for the given graph



C code and output :-

```
// KHAN MOHD OWAIS RAZA
// 20BCD7138
// CSE2001 Lab-9
// Challenging Question-3
#include <stdio.h>
#include <limits.h>
#define V 5
/*
```

Matrix table for the given graph:-

```
    A   B   C   D   E   F   G   H
*****
A   0   8   0   0   0  10   0   4
B   8   0   4   0  10   7   0   9
C   0   4   0   3   0   3   0   0
D   0   0   3   0  25  18   2   0
E   0  10   0  25   0   2   7   0
F  10   7   3  18   2   0   0   0
G   0   0   0   2   7   0   0   3
H   4   9   0   0   0   0   3   0
*/
```

```

int minKey(int key[], int mstSet[]) {
    int min = INT_MAX, min_index;
    int v;
    for (v = 0; v < V; v++)
        if (mstSet[v] == 0 && key[v] < min)
            min = key[v], min_index = v;
    return min_index;
}

int printMST(int parent[], int n, int graph[V][V]) {
    int i;
    printf("Edge   Weight\n");
    for (i = 1; i < V; i++)
        printf("%d - %d   %d \n", parent[i], i, graph[i][parent[i]]);
}

void primMST(int graph[V][V]) {
    int parent[V];
    int key[V], i, v, count;
    int mstSet[V];
    for (i = 0; i < V; i++)
        key[i] = INT_MAX, mstSet[i] = 0;
    key[0] = 0;
    parent[0] = -1;
    for (count = 0; count < V - 1; count++) {
        int u = minKey(key, mstSet);
        mstSet[u] = 1;
        for (v = 0; v < V; v++)
            if (graph[u][v] && mstSet[v] == 0 && graph[u][v] < key[v])
                parent[v] = u, key[v] = graph[u][v];
    }
    printMST(parent, V, graph);
}

int main() {
    int graph[V][V] = {
        {0, 8, 0, 0, 0, 10, 0, 4},
        {8, 0, 4, 0, 10, 7, 0, 9},
        {0, 4, 0, 3, 0, 3, 0, 0},
        {0, 10, 0, 25, 0, 2, 7, 0},
        {0, 10, 25, 0, 2, 7, 0},
        {10, 7, 3, 18, 2, 0, 0, 0},
        {0, 0, 0, 2, 7, 0, 0, 3},
        {4, 9, 0, 0, 0, 0, 3, 0}};
    primMST(graph);
    return 0;
}

```

```

1 // KHAN MOHD OWAIS RAZA
2 // 20BCD7138
3 // CSE2001 Lab-9
4 // Challenging Question-3
5 #include <stdio.h>
6 #include <limits.h>
7 #define V 5
8 /*
9 Matrix table for the given graph:-
10      A  B  C  D  E  F  G  H
11 *****
12 A   0  8  0  0  0 10  0  4
13 B   8  0  4  0 10  7  0  9
14 C   0  4  0  3  0  3  0  0
15 D   0  0  3  0 25 18  2  0
16 E   0 10  0 25  0  2  7  0
17 F  10  7  3 18  2  0  0  0
18 G   0  0  0  2  7  0  0  3
19 H   4  9  0  0  0  0  3  0
20 */
21 int minKey(int key[], int mstSet[]) {
22     int min = INT_MAX, min_index;
23     int v;
24     for (v = 0; v < V; v++)
25         if (mstSet[v] == 0 && key[v] < min)
26             min = key[v], min_index = v;
27     return min_index;
28 }
29 int printMST(int parent[], int n, int graph[V][V]) {
30     int i;
31     printf("Edge    Weight\n");
32     for (i = 1; i < V; i++)
33         printf("%d - %d    %d \n", parent[i], i, graph[i][parent[i]]);
34 }
35 void primMST(int graph[V][V]) {
36     int parent[V];
37     int key[V], i, v, count;
38     int mstSet[V];
39     for (i = 0; i < V; i++)
40         key[i] = INT_MAX, mstSet[i] = 0;
41     key[0] = 0;
42     parent[0] = -1;
43     for (count = 0; count < V - 1; count++) {
44         int u = minKey(key, mstSet);
45         mstSet[u] = 1;
46         for (v = 0; v < V; v++)
47             if (graph[u][v] && mstSet[v] == 0 && graph[u][v] < key[v])
48                 parent[v] = u, key[v] = graph[u][v];
49     }
50     printMST(parent, V, graph);
51 }
52 int main() {

```

```

53 int graph[V][V] = {
54     {0, 8, 0, 0, 0, 10, 0, 4},
55     {8, 0, 4, 0, 10, 7, 0, 9},
56     {0, 4, 0, 3, 0, 3, 0, 0},
57     {0, 10, 0, 25, 0, 2, 7, 0},
58     {0, 10, 25, 0, 2, 7, 0},
59     {10, 7, 3, 18, 2, 0, 0, 0},
60     {0, 0, 0, 2, 7, 0, 0, 3},
61     {4, 9, 0, 0, 0, 0, 3, 0}};
62 primMST(graph);
63 return 0;
64 }

```

C:\Users\Owais\Desktop\Challenging\_Question3.exe

Edge Weight

```

0 - 1      8
1 - 2      4
2 - 3      0
1 - 4     10

```

```

-----
Process exited after 0.04776 seconds with return value 0
Press any key to continue . . .

```

### Challenging Q.4]

Write a Program to implement Kruskal's Algorithm and find the minimum spanning tree for the above graph

```
// KHAN MOHD OWAIS RAZA
// 20BCD7138
// CSE2001 Lab-9
// Challenging Question-3
#include <stdio.h>
#include <stdlib.h>
/*
Matrix table for the given graph:-
  A   B   C   D   E   F   G   H
*****
A   0   8   0   0   0  10   0   4
B   8   0   4   0  10   7   0   9
C   0   4   0   3   0   3   0   0
D   0   0   3   0  25  18   2   0
E   0  10   0  25   0   2   7   0
F  10   7   3  18   2   0   0   0
G   0   0   0   2   7   0   0   3
H   4   9   0   0   0   0   3   0
*/
int i, j, k, a, b, u, v, n, ne = 1;
int min, mincost = 0, cost[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 adjacency matrix:\n");
for (i = 1; i <= n; i++) {
for (j = 1; j <= n; j++) {
scanf("%d", &cost[i][j]);
if (cost[i][j] == 0)
cost[i][j] = 999;
}}
printf("The edges of minimum spanning tree:\n");
while (ne < n) {
for (i = 1, min = 999; i <= n; i++) {
for (j = 1; j <= n; j++) {
if (cost[i][j] < min) {
min = cost[i][j];
a = u = i;
b = v = j;
}}}
u = find(u);
```

```

v = find(v);
if (uni(u, v)) {
printf("%d edge (%d,%d) = %d\n", ne++, a, b, min);
mincost += min;
}
cost[a][b] = cost[b][a] = 999;
}
printf("\n\tMinimum cost = %d\n", mincost);
getch();
}
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;
}
}

```

Challenging_Question3.c	Challenging_Question4.c
<pre> 1 // KHAN MOHD OWAIS RAZA 2 // 20BCD7138 3 // CSE2001 Lab-9 4 // Challenging Question-3 5 #include &lt;stdio.h&gt; 6 #include &lt;stdlib.h&gt; 7 /* 8 Matrix table for the given graph:- 9      A  B  C  D  E  F  G  H 10 ***** 11 A    0  8  0  0  0 10  0  4 12 B    8  0  4  0 10  7  0  9 13 C    0  4  0  3  0  3  0  0 14 D    0  0  3  0 25 18  2  0 15 E    0 10  0 25  0  2  7  0 16 F   10  7  3 18  2  0  0  0 17 G    0  0  0  2  7  0  0  3 18 H    4  9  0  0  0  0  3  0 19 */ 20 int i, j, k, a, b, u, v, n, ne = 1; 21 int min, mincost = 0, cost[9][9], parent[9]; 22 int find(int); 23 int uni(int, int); 24 void main() { 25     printf("\nEnter the no. of vertices:"); 26     scanf("%d", &amp;n); 27     printf("\nEnter the adjacency matrix:\n"); 28     for (i = 1; i &lt;= n; i++) { 29         for (j = 1; j &lt;= n; j++) { </pre>	

```

30 scanf("%d", & cost[i][j]);
31 if (cost[i][j] == 0)
32 cost[i][j] = 999;
33 }
34 printf("The edges of minimum spanning tree:\n");
35 while (ne < n) {
36 for (i = 1, min = 999; i <= n; i++) {
37 for (j = 1; j <= n; j++) {
38 if (cost[i][j] < min) {
39 min = cost[i][j];
40 a = u = i;
41 b = v = j;
42 }}}
43 u = find(u);
44 v = find(v);
45 if (uni(u, v)) {
46 printf("%d edge (%d,%d) =%d\n", ne++, a, b, min);
47 mincost += min;
48 }
49 cost[a][b] = cost[b][a] = 999;
50 }
51 printf("\n\tMinimum cost = %d\n", mincost);
52 getch();
53 }
54 int find(int i) {
55 while (parent[i])
56 i = parent[i];
57 return i;
58 }
59 int uni(int i, int j) {
60 if (i != j) {
61 parent[j] = i;
62 return 1;
63 }
64 return 0;
65 }

```

C:\Users\Owais\Desktop\Challenging\_Question4.exe

Enter the no. of vertices:8

Enter the adjacency matrix:

```

0 8 0 0 0 10 0 4
8 0 4 0 0 10 7 0 9
0 4 0 3 0 3 0 0
0 0 3 0 25 18 2 0
0 10 0 25 0 2 7 0
10 7 3 18 2 0 0 0
0 0 0 2 7 0 0 3
4 9 0 0 0 0 3 0

```

The edges of minimum spanning tree:

```

1 edge (4,7) =2
2 edge (5,6) =2
3 edge (3,4) =3
4 edge (3,6) =3
5 edge (7,8) =3
6 edge (1,8) =4
7 edge (2,3) =4

```

Minimum cost = 21

-----  
Process exited after 175 seconds with return value 13