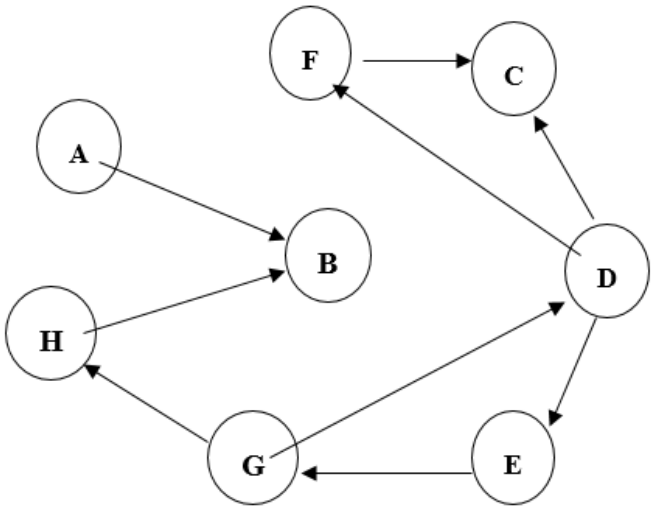
CSE2001 (Data Structures & Algorithms) Lab-9

KHAN MOHD OWAIS RAZA
20BCD7138

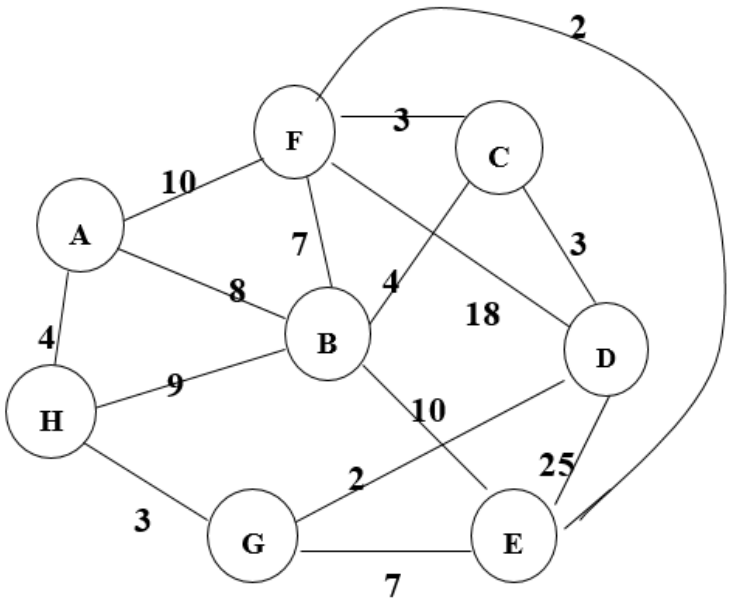
1. Write a Program to implement DFS algorithm and print the DFS sequence for below graph start with node D.



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

Challenging:

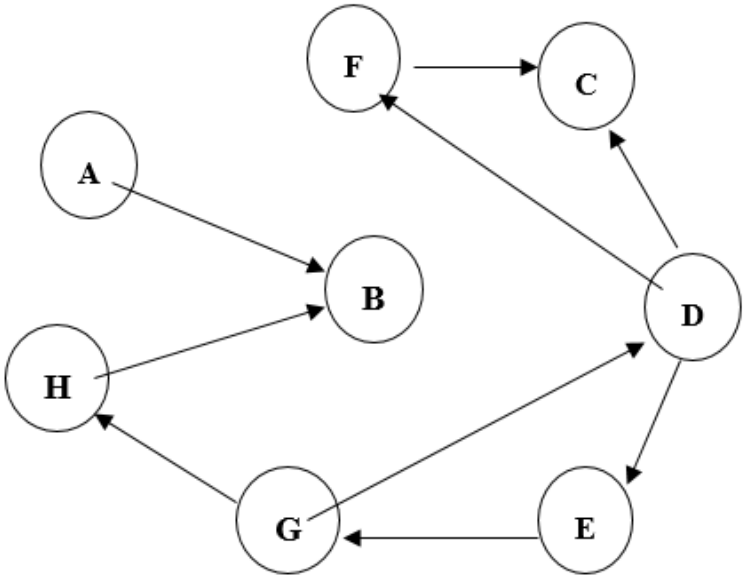
1. Write a Program to check the connectivity of a graph using BFS.
2. Write a program to implement Dijkstra's algorithm
3. Write a Program to Implement Prim's Algorithm and find the minimum spanning tree for the given graph



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.
A     F     C
 \     \   /
  \     \ /
   B     D
  /     /\
H     /  \
 \   /    \
  \ /      \
   G ----- E

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

```c
// 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)
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)
*/
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++){
```

```
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.
A     F     C
 \     \   /
  \     \ /
   B     D
  /     /\
H      /  \
 \    /    \
  \  /      \
   G ------ E
Let it be:
A = 0, B = 1, C = 2
D = 3, E = 4, F = 5
G = 6, H = 7

0     5     2
 \     \   /
  \     \ /
   1     3
  /     /\
7      /  \
 \    /    \
  \  /      \
   6 ------- 4

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;
}}}
```

Question1.c Question2.c

```c
// KHAN MOHD OWAIS RAZA
// 20BCD7138
// CSE2001 Lab-9
// Question-2
/*
According to question, the graph is
in the form of alphabets.
A     F     C
 \     \   /
  \     \ /
   B     D
  /     /\
H      /  \
 \    /    \
  \  /      \
   G ------- E
Let it be:
A = 0, B = 1, C = 2
D = 3, E = 4, F = 5
G = 6, H = 7

0     5     2
 \     \   /
  \     \ /
   1     3
  /     /\
7      /  \
 \    /    \
  \  /      \
   6 ------- 4

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();
```

```c
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);
```

```
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;
}
```

Question1.c | Question2.c | Challenging_Question1.c

```c
// 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;
}
```

```
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

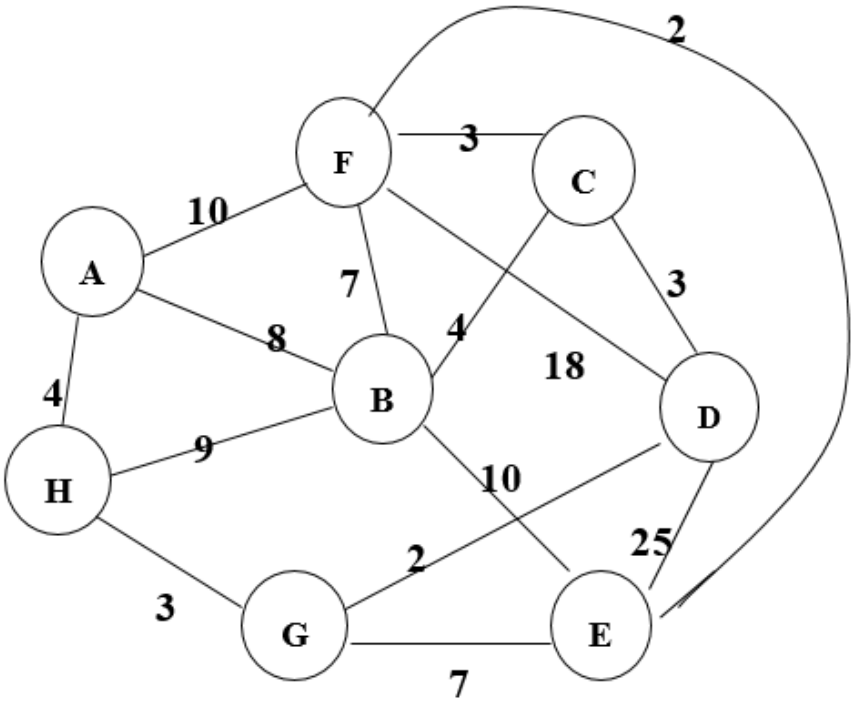
```c
// KHAN MOHD OWAIS RAZA
// 20BCD7138
// CSE2001 Lab-9
// Challenging Question-2
#include<stdio.h>
#define INFINITY 9999
#define MAX 10
void dijkstra(int G[MAX][MAX],int n,int startnode);
int main(){
int G[MAX][MAX],i,j,n,u;
printf("Enter no. of vertices: ");
scanf("%d",&n);
printf("\nEnter the adjacency 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)
```

```
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;
}
```

Challenging_Question3.c | Challenging_Question4.c

```c
// 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() {
```

```
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

```c
// 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++) {
```

```c
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;
}
```

```
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 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
```