PRESIDENCY UNIVERSITY

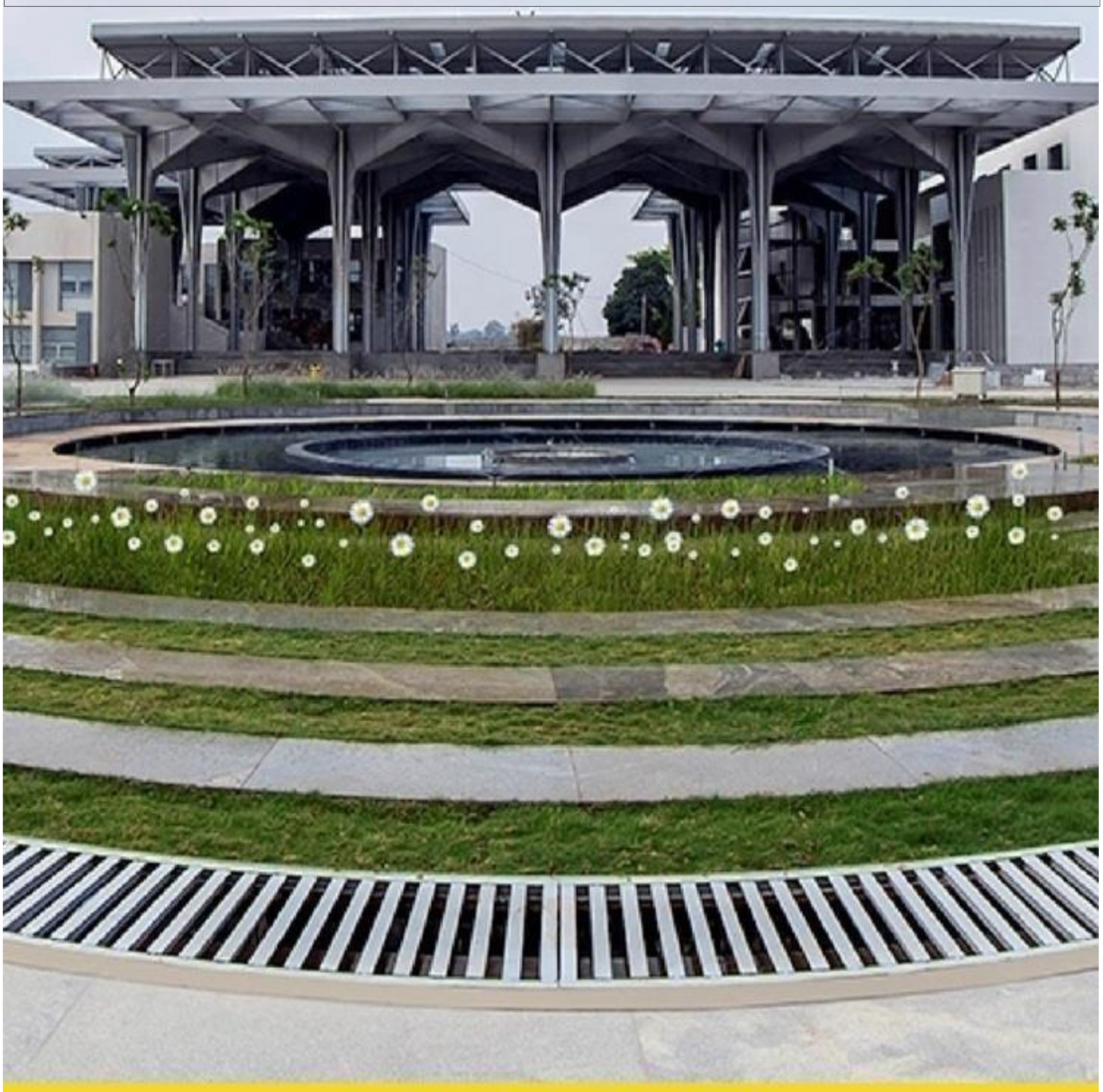
GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

Private University Estd. in Karnataka State by Act No. 41 of 2013

Department of Computer Science

CSE2007: Design and Analysis of Algorithm

4th Semester 2021-22



www.presidencyuniversity.in

Session 4

Greedy Techniques
a. Find Minimum Cost Spanning Tree of a given undirected graph using Kruskal's algorithm b. Find Minimum Cost Spanning Tree of a given undirected graph using Prim's algorithm
Algorithms

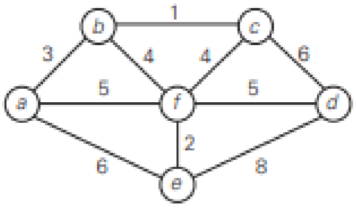
ALGORITHM Kruskal(G) //Kruskal's algorithm for constructing a minimum spanning tree
//Input: A weighted connected graph G = (V, E)
//Output: ET , the set of edges composing a minimum spanning tree of G
sort E in nondecreasing order of the edge weights w(ei 1) ≤ . . . ≤ w(ei| E|)
ET ←NULL;
ecounter ←0 //initialize the set of tree edges and its size
k←0 //initialize the number of processed edges
while ecounter < | V| − 1 do
k← k + 1
if ET ∪ { eik} is acyclic
ET← ET ∪ { eik};
ecounter ← ecounter + 1
return ET.
 	
ALGORITHM Prim(G) //Prim's algorithm for constructing a minimum spanning tree
//Input: A weighted connected graph *G* = _ *V*, *E* _
//Output: *ET* , the set of edges composing a minimum spanning tree of *G*
 	
VT←{ *v* 0} //the set of tree vertices can be initialized with any vertex
ET←NULL
for *i* ←1 to | *V*| − 1 do
find a minimum-weight edge *e** = *(v*, u*)* among all the edges *(v, u)*
such that *v* is in *VT* and *u* is in
V − *VT* *VT*← *VT* ∪ { *u**}
ET← *ET* ∪ { *e**}
return *ET*.

Coding using C Language

a) Using Kruskal's Algorithm

```
#include<stdio.h>
int i,j,k,a,b,v,u,n,ne=1;
int min,mincost=0,cost[9][9],parent[9];
void main()
{
printf("\nEnter the number of vertices\n");
scanf("%d",&n);
printf("Enter 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("\nThe edges of spanning treeare:\n\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;
                        }
                }
                while(parent[u])
                        u=parent[u];
                while(parent[v])
                        v=parent[v];
                if(u!=v)
                {
                        printf("\n%d\tEdge(%d,%d)=%d",ne++,a,b,min);
                        mincost+=min;
                        parent[v]=u;
                }
                cost[a][b]=cost[b][a]=999;
        }
        printf("\n\tMiINCOST=%d\n",mincost);

}
```

Undirected Graph

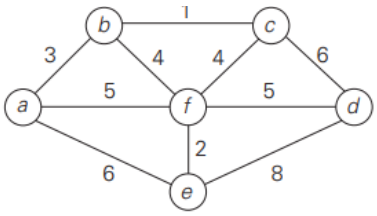


Input to be given in the form of Adjacency Matrix as shown below for above graph:

```
Enter the number of vertices
6
Enter the adjacency matrix::
0 3 0 0 6 5
3 0 1 0 0 4
0 1 0 6 0 4
0 0 6 0 8 5
6 0 0 8 0 2
5 4 4 5 2 0
```

Output: Minimum Spanning Tree

```
The edges of spanning treeare:


1        Edge(2,3)=1
2        Edge(5,6)=2
3        Edge(1,2)=3
4        Edge(2,6)=4
5        Edge(4,6)=5
         MiINCOST=15
```



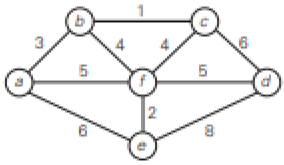
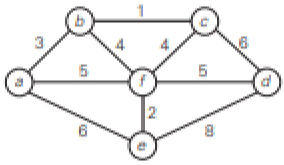
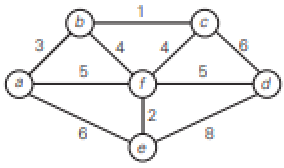
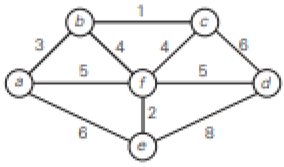
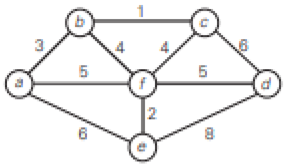
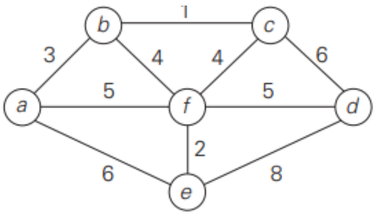
Tree edges	Sorted list of edges	Illustration
	bc ef ab bf cf af df ae cd de 1 2 3 4 4 5 5 6 6 8	
bc 1	bc **ef** ab bf cf af df ae cd de 1 2 3 4 4 5 5 6 6 8	
ef 2	bc ef **ab** bf cf af df ae cd de 1 2 3 4 4 5 5 6 6 8	
ab 3	bc ef ab **bf** cf af df ae cd de 1 2 3 4 4 5 5 6 6 8	
bf 4	bc ef ab bf cf af **df** ae cd de 1 2 3 4 4 5 5 6 6 8	
df 5		

FIGURE 9.5 Application of Kruskal's algorithm. Selected edges are shown in bold.

b) USING PRIM'S ALGORITHM

```
#include<stdio.h>
int i, j, a, b, v, u,n, ne=1;
int min,mincost=0, cost[9][9], visited[9];
void main()
{
        printf( "The no of vertices=\t");
        scanf("%d",&n);
        printf("Enter the adjacency matrix=\t");
        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 spanning tree are \t");
        visited[1]=1;
        while(ne<n)
        {
                for(i=1,min=999;i<=n; i++)
                {
                        for(j=1;j<=n;j++)
                        {
                                if(cost[i][j]<min)
                                {
                                        if(visited[i]==0)
                                                continue;
                                        else
                                        {
                                                min=cost[i][j];
                                                a=u=i;
                                                b=v=j;
                                        }
                                }
                        }
                }

                if(visited[v]==0)
                {
                        printf("\n%d\t Edge \t(%d, %d)=%d\n",ne++, a, b, min);
                        mincost+=min;
                        visited[b]=1;
                }
                cost[a][b]=cost[b][a]=999;
        }
 printf("\n\t mincost=%d\n",mincost);
}
```



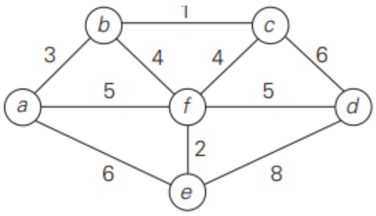
Input to be given in the form of Adjacency Matrix as shown below for above graph:

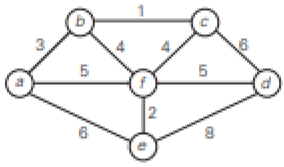
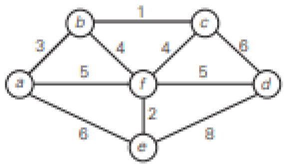
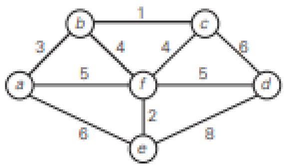
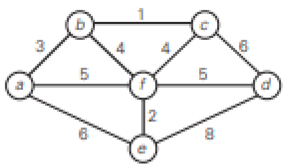
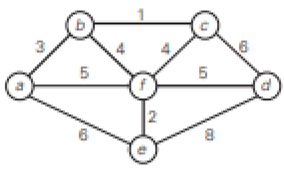
```
The no of vertices=     6
Enter the adjacency matrix=
0 3 0 0 6 5
3 0 1 0 0 4
0 1 0 6 0 4
0 0 6 0 8 5
6 0 0 8 0 2
5 4 4 5 2 0
```

Output: Minimum Spanning Tree

```
The edges of spanning tree are
1         Edge   (1, 2)=3

2         Edge   (2, 3)=1

3         Edge   (2, 6)=4

4         Edge   (6, 5)=2

5         Edge   (6, 4)=5

          mincost=15
```

Illustration of Prims Algorithm :



Tree vertices	Remaining vertices	Illustration
a(−, −)	**b(a, 3)** c(−, ∞) d(−, ∞) e(a, 6) f(a, 5)	
b(a, 3)	**c(b, 1)** d(−, ∞) e(a, 6) f(b, 4)	
c(b, 1)	d(c, 6) e(a, 6) **f(b, 4)**	
f(b, 4)	d(f, 5) **e(f, 2)**	
e(f, 2)	**d(f, 5)**	
d(f, 5)		