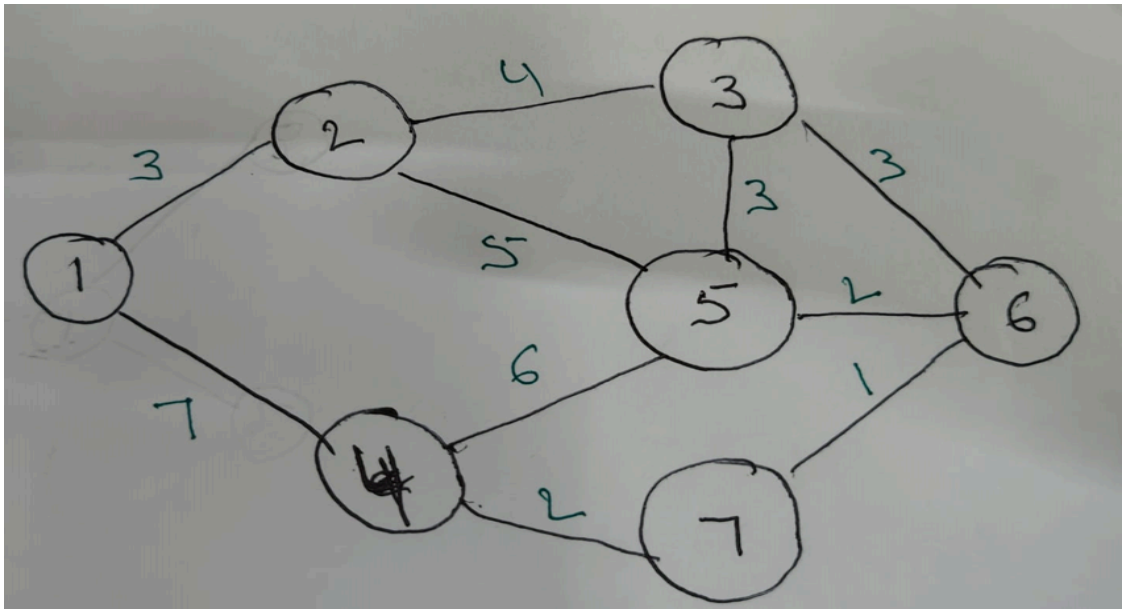
S.No	Date	IDX	PROGRAM
3	16-07-2015	3.1	Implementation of Single Source Shortest Paths using Greedy method when the graph is represented by adjacency matrix .
		3.2	Compare the performance of Single Source Shortest Paths using Greedy method when the graph is represented by adjacency lists.

Week 3 - Program 3.1

AIM : Implementation of Single Source Shortest Paths using Greedy method when the graph is represented by adjacency matrix

Consider Example Given Graph with weights



The Adjacency Matrix is

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

Adjacency Lists are

1 □ 2 □ 4
2 □ 1 □ 3 □ 5
3 □ 2 □ 5 □ 6
4 □ 1 □ 5 □ 7
5 □ 2 □ 3 □ 4 □ 6 □ 7
6 □ 3 □ 5 □ 7
7 □ 4 □ 5 □ 6

Starting from node 1

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#define INF 999

void dijkstra(int [][10],int,int);

void main()
{
int G[10][10],i,j,n,s;
printf(" Enter number of vertices : ");
scanf("%d",&n);

printf(" Enter the cost  matrix \n");
for(i=1;i<=n;i++)
{
        for(j=1;j<=n;j++)
        {
                scanf("%d",&G[i][j]);
        }
}

printf(" Enter source vertex :  ");
scanf("%d",&s);

dijkstra(G,n,s);
}

void dijkstra(int G[][10],int n,int s)
{
        int cost[10][10],dist[10],visit[10],pred[10];
        int next,min,i,j,count;
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                {
                        if(G[i][j]==0)
```

```
                cost[i][j]=INF;
            else
                cost[i][j]=G[i][j];
        }
}
for(i=1;i<=n;i++)
{
        dist[i]=cost[s][i];
        visit[i]=0;
        pred[i]=s;
}
dist[s]=0;
visit[s]=1;
count=1;

while(count<n)
{
        min=INF;
        for(i=1;i<=n;i++)
        {
                if(visit[i]==0 && min>dist[i])
                {
                        min=dist[i];
                        next=i;
                }
        }
        visit[next]=1;
        for(i=1;i<=n;i++)
        {
                if(visit[i]==0&&dist[i]>dist[next]+cost[next][i])
                {
                        dist[i]=dist[next]+cost[next][i];
                        pred[i]=next;
                }
        }
count++;
}

for(i=1;i<=n;i++)
{
        if(i!=s)
```

```
            {
                printf("\n Distance of node %d = %d\n",i,dist[i]);
                printf(" Path= %d",i);

                j=i;

                do
                {
                    j=pred[j];
                    printf(" <- %d",j);
                } while(j!=s);
            }
        }
}
```

OUTPUT:

```
Enter number of vertices : 7
Enter the cost  matrix
 0   3   999   7  999   999   999
 3   0    4   999  5    999   999
999  4    0   999  3     3    999
 7  999  999   0   6    999    2
999  5    3    6   0     2    999
999 999   3   999  2     0     1
999 999  999   2  999    1     0
Enter source vertex :  1

Distance of node 2 = 3
Path= 2 <- 1
Distance of node 3 = 7
Path= 3 <- 2 <- 1
Distance of node 4 = 7
Path= 4 <- 1
Distance of node 5 = 8
Path= 5 <- 2 <- 1
Distance of node 6 = 10
Path= 6 <- 3 <- 2 <- 1
Distance of node 7 = 9
Path= 7 <- 4 <- 1
```

Week 4- Program 4.2

AIM : Compare the performance of Single Source Shortest Paths using Greedy method when the graph is represented by adjacency lists.

PROGRAM :

```
#include<stdio.h>
#include<conio.h>
int shortest(int ,int);
int cost[10][10],dist[20],i,j,n,k,l,m,v,totcost,S[20],path[20],pre[20],p=0;
int main()
{
   int c,min;

   printf("enter no of vertices\t");
   scanf("%d",&n);
   for(i=1;i<=n;i++)
      for(j=1;j<=n;j++)
         cost[i][j]=31999;
   printf("\nenter no of edges\t");
   scanf("%d",&m);
   printf("\nenter EDGE & Cost\n");
   for(k=1;k<=m;k++)
   {
      scanf("%d %d %d",&i,&j,&c);
      cost[i][j]=c;
   }
   printf("\nenter initial vertex\t");
   scanf("%d",&v);
   for(i=1;i<=n;i++)
   {
      S[i]=0;
      dist[i]=cost[v][i];
      if( dist[i] != 31999 )
        pre[i] = v;
      else
        pre[i] = 0;
```

```
}
S[v]=1;
dist[v]=0;
for(i=2;i<=n;i++)
{
   min=31999;
   for(j=1;j<=n;j++)
   {
      if(dist[j]<min && S[j]!=1)
      {
         min=dist[j];
          k=j;
      }
   }
   S[k]=1;
   for(j=1;j<=n;j++)
      if(cost[k][j]!=31999 && dist[j]>=dist[k]+cost[k][j] && S[j]!=1)
      {
          dist[j]=dist[k]+cost[k][j];
          pre[j]=k;
      }
}
printf("\n shortest paths and pathlength  are \n ");
for(i=1; i<=n; i++)
{
   if(pre[i] == 0 )
    {
      if(i != v)
        printf("\n no path to %d\n",i);
    }
   else
   {
      l=i;
      do
      {
         path[++p] = l;
         l= pre[l];
      }while(l != v);
      printf("%d -> ",v);
      for(l=p; l>=1; l--)
         printf("%d -> ",path[l]);
```

```
        printf("\b\b\b");
        printf( " \t\t length = %d\n",dist[i]);
      }
      p=0;
   }
   return 0;
}
```

OUTPUT:

```
enter no of vertices    7

enter no of edges       10

enter EDGE & Cost
1 2 3
1 4 7
2 3 4
2 5 5
3 5 3
3 6 3
4 5 6
4 7 2
5 6 2
6 7 1

enter initial vertex    1

 shortest paths and pathlength  are
 1 -> 2  >                length = 3
1 -> 2 -> 3  >            length = 7
1 -> 4  >                 length = 7
1 -> 2 -> 5  >            length = 8
1 -> 2 -> 5 -> 6  >               length = 10
1 -> 4 -> 7  >            length = 9
```