VIBHA HUGAR

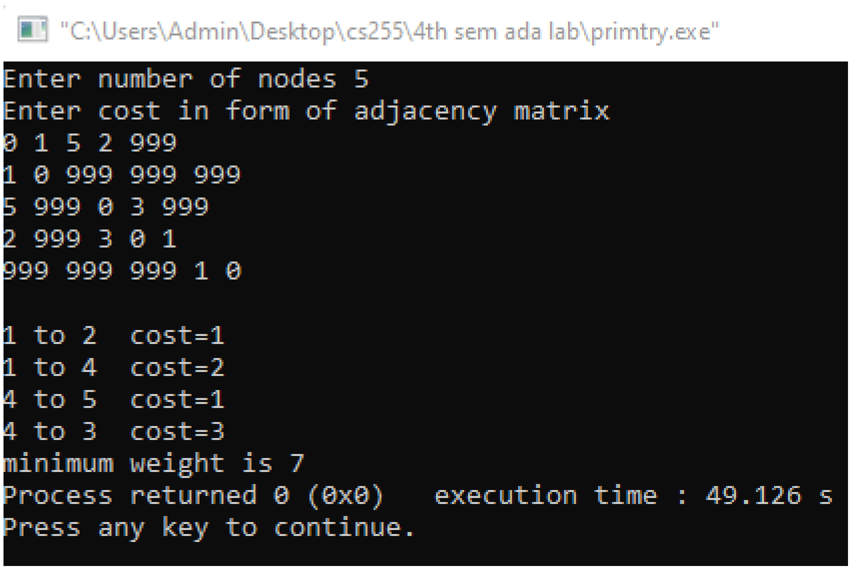
1BM21CS255

WEEK 7 ADA

CODE FOR PRIM'S ALGORITHM

```
#include<stdio.h>
int main()
{
  int cost[10][10],visited[10]={0},i,j,n,no_e=1,min,a,b,min_cost=0;
  printf("Enter the number of nodes:\n");
  scanf("%d",&n);
  printf("Enter the cost in form of 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]=1000;
    }
  }


  visited[1]=1;
  while(no_e<n)
  {
        min=1000;

        for(i=1;i<=n;i++)
        {
            for(j=1;j<=n;j++)
            {
                if(cost[i][j]<min)
                {
                    if(visited[i]!=0)
                    {
                        min=cost[i][j];
                        a=i;
                        b=j;
                    }
                }
            }
        }

        if(visited[b]==0)
        {
            printf("\n%d to %d  cost=%d",a,b,min);
            min_cost=min_cost+min;
            no_e++;
        }
        visited[b]=1;

        cost[a][b]=cost[b][a]=1000;
    }
    printf("\nminimum weight is %d",min_cost);
    return 0;

}
```

OUTPUT



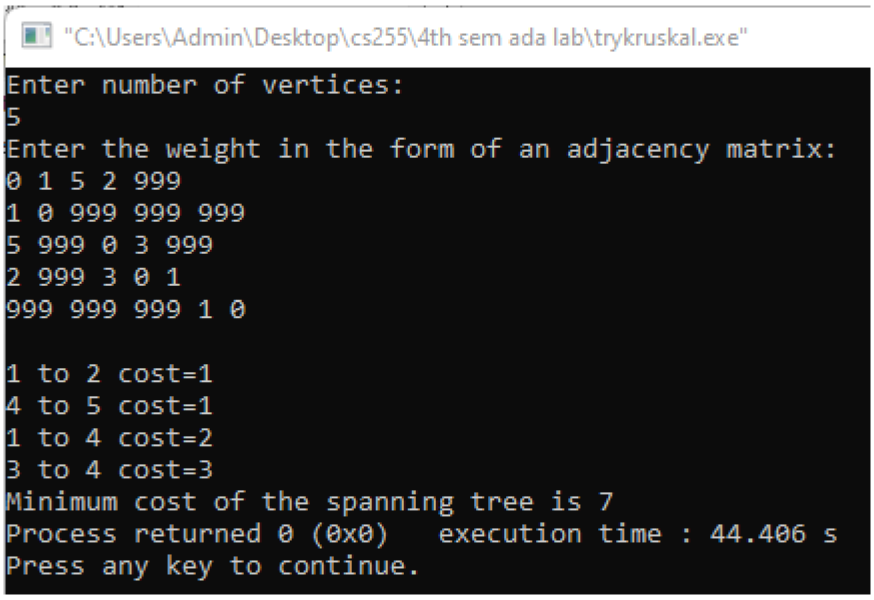
"C:\Users\Admin\Desktop\cs255\4th sem ada lab\primtry.exe"

```
Enter number of nodes 5
Enter cost in form of adjacency matrix
0 1 5 2 999
1 0 999 999 999
5 999 0 3 999
2 999 3 0 1
999 999 999 1 0

1 to 2   cost=1
1 to 4   cost=2
4 to 5   cost=1
4 to 3   cost=3
minimum weight is 7
Process returned 0 (0x0)   execution time : 49.126 s
Press any key to continue.
```

CODE FOR KRUSKAL’S ALGORITHM

```
#include<stdio.h>

int parent[10]={0};

int find_parent(int);

int is_cyclic(int,int);

int main()

{

    int cost[10][10],min_cost=0,min,i,j,n,no_e=1,a,b,u,v,x;

    printf("Enter number of vertex\n");

    scanf("%d",&n);

    printf("Enter weight in form of 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;

        }

    }




    while(no_e<n)

    {

        min=999;


        for(i=1;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_parent(u);
        v=find_parent(v);
        x=is_cyclic(u,v);
        if(x==1)
        {
            printf("\n%d to %d",a,b);
            no_e++;
            min_cost+=min;
        }
        cost[a][b]=cost[b][a]=999;
    }
    printf("\nMinimum cost of the spanning tree is %d",min_cost);
    return 0;
}



int find_parent(int a)
{
    while(parent[a]!=0)
      a=parent[a];
    return a;
}



int is_cyclic(int a ,int b)
{
    if(a!=b)
    {
        parent[b]=a;
        return 1;
    }
    return 0;
}
```

OUTPUT



"C:\Users\Admin\Desktop\cs255\4th sem ada lab\trykruskal.exe"

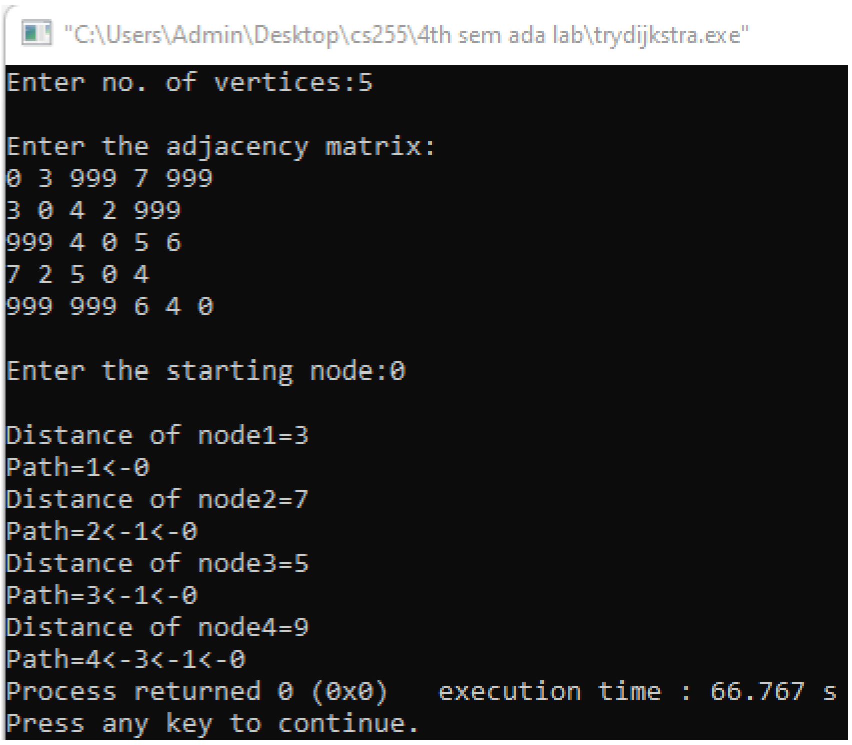
```
Enter number of vertices:
5
Enter the weight in the form of an adjacency matrix:
0 1 5 2 999
1 0 999 999 999
5 999 0 3 999
2 999 3 0 1
999 999 999 1 0

1 to 2 cost=1
4 to 5 cost=1
1 to 4 cost=2
3 to 4 cost=3
Minimum cost of the spanning tree is 7
Process returned 0 (0x0)   execution time : 44.406 s
Press any key to continue.
```

CODE FOR DIJKSTRA'S ALGORITHM

```
#include<stdio.h>

#include<conio.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);
}
}
```

OUTPUT


"C:\Users\Admin\Desktop\cs255\4th sem ada lab\trydijkstra.exe"

```
Enter no. of vertices:5

Enter the adjacency matrix:
0 3 999 7 999
3 0 4 2 999
999 4 0 5 6
7 2 5 0 4
999 999 6 4 0

Enter the starting node:0

Distance of node1=3
Path=1<-0
Distance of node2=7
Path=2<-1<-0
Distance of node3=5
Path=3<-1<-0
Distance of node4=9
Path=4<-3<-1<-0
Process returned 0 (0x0)   execution time : 66.767 s
Press any key to continue.
```