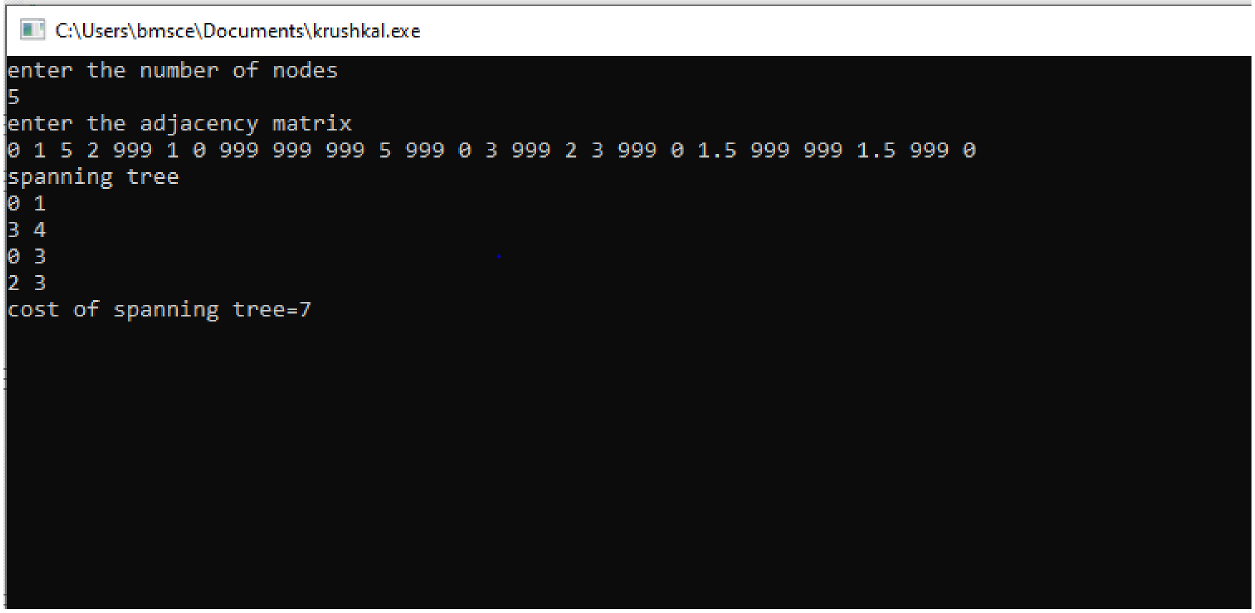
Krushkal’s algorithm:

```
#include<stdio.h>

#include<conio.h>


int find(int v,int parent[10])

{

  while(parent[v]!=v)

  {

    v=parent[v];

  }

  return v;

}


void union1(int i,int j,int parent[10])

{

  if(i<j)

    parent[j]=i;

  else

    parent[i]=j;

}


void kruskal(int n,int a[10][10])

{

  int count,k,min,sum,i,j,t[10][10],u,v,parent[10];

  count=0;
```

```
k=0;
sum=0;
for(i=0;i<n;i++)
  parent[i]=i;
while(count!=n-1)
{
  min=999;
  for(i=0;i<n;i++)
  {
        for(j=0;j<n;j++)
        {

          if(a[i][j]<min && a[i][j]!=0)
          {
                min=a[i][j];
                u=i;
                v=j;
           }
         }
   }
   i=find(u,parent);
   j=find(v,parent);
   if(i!=j)
   {
         union1(i,j,parent);
```

```
          t[k][0]=u;
          t[k][1]=v;
          k++;
          count++;
          sum=sum+a[u][v];
    }
    a[u][v]=a[v][u]=999;
  }
  if(count==n-1)
  {
    printf("spanning tree\n");
    for(i=0;i<n-1;i++)
    {
          printf("%d %d\n",t[i][0],t[i][1]);
    }
    printf("cost of spanning tree=%d\n",sum);
  }
  else
    printf("spanning tree does not exist\n");
 }

void main()
{
  int n,i,j,a[10][10];
  printf("enter the number of nodes\n");

  scanf("%d",&n);

  printf("enter the adjacency matrix\n");

  for(i=0;i<n;i++)

   for(j=0;j<n;j++)

          scanf("%d",&a[i][j]);

  kruskal(n,a);

  getch();


}
```



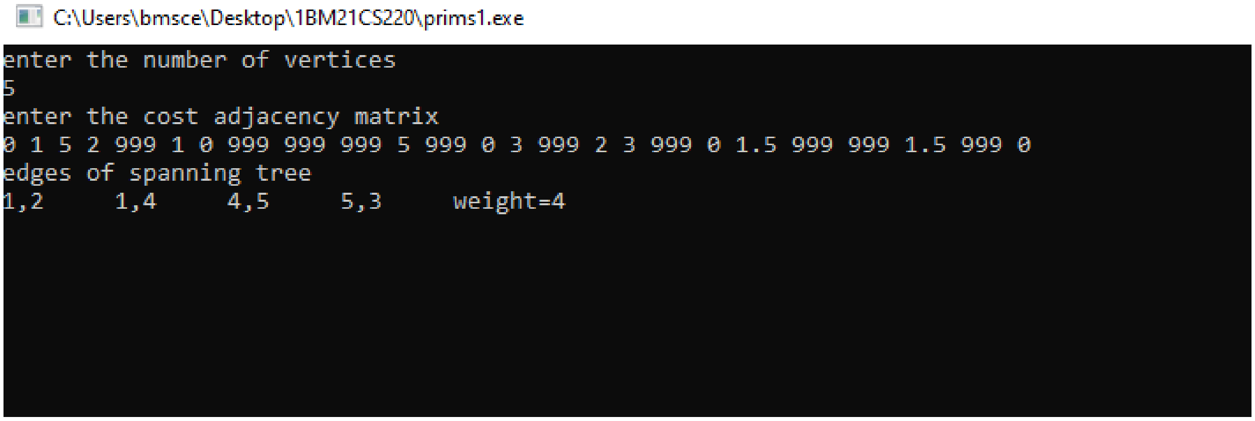
```
enter the number of nodes
5
enter the adjacency matrix
0 1 5 2 999 1 0 999 999 999 5 999 0 3 999 2 3 999 0 1.5 999 999 1.5 999 0
spanning tree
0 1
3 4
0 3
2 3
cost of spanning tree=7
```

Prim’s algorithm:

```
#include<stdio.h>

#include<conio.h>


int cost[10][10],vt[10],et[10][10],vis[10],j,n;

int sum=0;

int x=1;

int e=0;

void prims();


void main()

{

  int i;

  printf("enter the number of vertices\n");

  scanf("%d",&n);

  printf("enter the cost adjacency matrix\n");

  for(i=1;i<=n;i++)

  {

      for(j=1;j<=n;j++)

      {

               scanf("%d",&cost[i][j]);

      }

      vis[i]=0;

   }

   prims();
```

```
    printf("edges of spanning tree\n");
    for(i=1;i<=e;i++)
    {
              printf("%d,%d\t",et[i][0],et[i][1]);
    }
    printf("weight=%d\n",sum);
    getch();
}

void prims()
{
   int s,min,m,k,u,v;
   vt[x]=1;
   vis[x]=1;
   for(s=1;s<n;s++)
   {
       j=x;
       min=999;
       while(j>0)
       {
                 k=vt[j];
                 for(m=2;m<=n;m++)
                 {
                   if(vis[m]==0)
                   {
```

```
                if(cost[k][m]<min)

                {

                  min=cost[k][m];

                  u=k;

                  v=m;

                }

            }

        }

        j--;

  }

  vt[++x]=v;

  et[s][0]=u;

  et[s][1]=v;

  e++;

  vis[v]=1;

  sum=sum+min;

 }

}
```

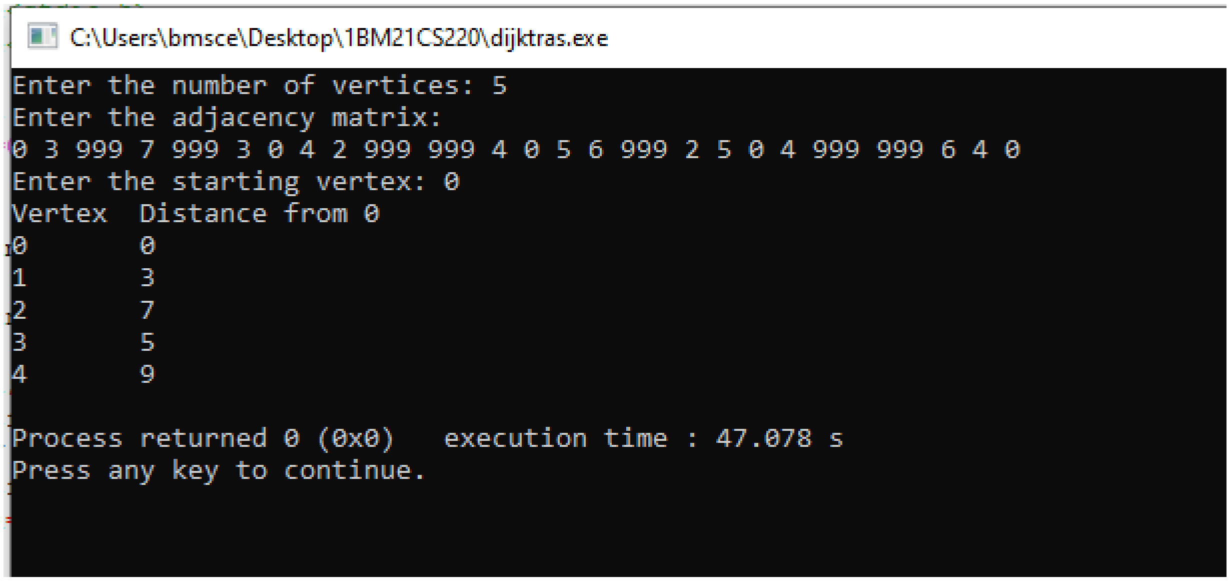


C:\Users\bmsce\Desktop\1BM21CS220\prims1.exe

```
enter the number of vertices
5
enter the cost adjacency matrix
0 1 5 2 999 1 0 999 999 999 5 999 0 3 999 2 3 999 0 1.5 999 999 1.5 999 0
edges of spanning tree
1,2     1,4     4,5     5,3     weight=4
```

```
#include <stdio.h>

#include <stdbool.h>


#define MAX_VERTICES 100

#define INF 9999999


int graph[MAX_VERTICES][MAX_VERTICES];

int numVertices;


void dijkstra(int startVertex) {

   int distance[MAX_VERTICES];

   bool visited[MAX_VERTICES];


   for (int i = 0; i < numVertices; i++) {

      distance[i] = INF;

      visited[i] = false;

   }

   distance[startVertex] = 0;
```

```
    for (int count = 0; count < numVertices - 1; count++) {
        int u = -1;
                for (int v = 0; v < numVertices; v++) {
            if (!visited[v] && (u == -1 || distance[v] < distance[u])) {
                u = v;
            }
        }
        visited[u] = true;


        for (int v = 0; v < numVertices; v++) {
            if (!visited[v] && graph[u][v] && distance[u] + graph[u][v] < distance[v]) {
                distance[v] = distance[u] + graph[u][v];
            }
        }
    }
    printf("Vertex\tDistance from %d\n", startVertex);
    for (int i = 0; i < numVertices; i++) {
        printf("%d\t%d\n", i, distance[i]);
    }
}

int main() {
    printf("Enter the number of vertices: ");
    scanf("%d", &numVertices);


    printf("Enter the adjacency matrix:\n");
    for (int i = 0; i < numVertices; i++) {
        for (int j = 0; j < numVertices; j++) {
            scanf("%d", &graph[i][j]);
        }
    }


    int startVertex;
    printf("Enter the starting vertex: ");
    scanf("%d", &startVertex);


    dijkstra(startVertex);
    return 0;
}
```



```
Enter the number of vertices: 5
Enter the adjacency matrix:
0 3 999 7 999 3 0 4 2 999 999 4 0 5 6 999 2 5 0 4 999 999 6 4 0
Enter the starting vertex: 0
Vertex  Distance from 0
0       0
1       3
2       7
3       5
4       9

Process returned 0 (0x0)   execution time : 47.078 s
Press any key to continue.
```