**EXPERIMENT N0-6**

**AIM: Implementation of Prim's and Kruskal's algorithm for Minimum Cost Spanning Tree.**

PRIM’S ALGORITHM:

**CODE:**

#include<stdio.h>

#include<conio.h>

int a,b,u,v,n,i,j,ne=1;

int visited[10]= {

0

}

,min,mincost=0,cost[10][10];

void main() {

printf("\nEnter the number of nodes:");

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;

}

visited[1]=1;

printf("\n");

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) {

min=cost[i][j];

a=u=i;

b=v=j;

}

if(visited[u]==0 || visited[v]==0) {

printf("\nEdge %d:(%d %d) cost:%d",ne++,a,b,min);

mincost+=min;

visited[b]=1;

}

cost[a][b]=cost[b][a]=999;

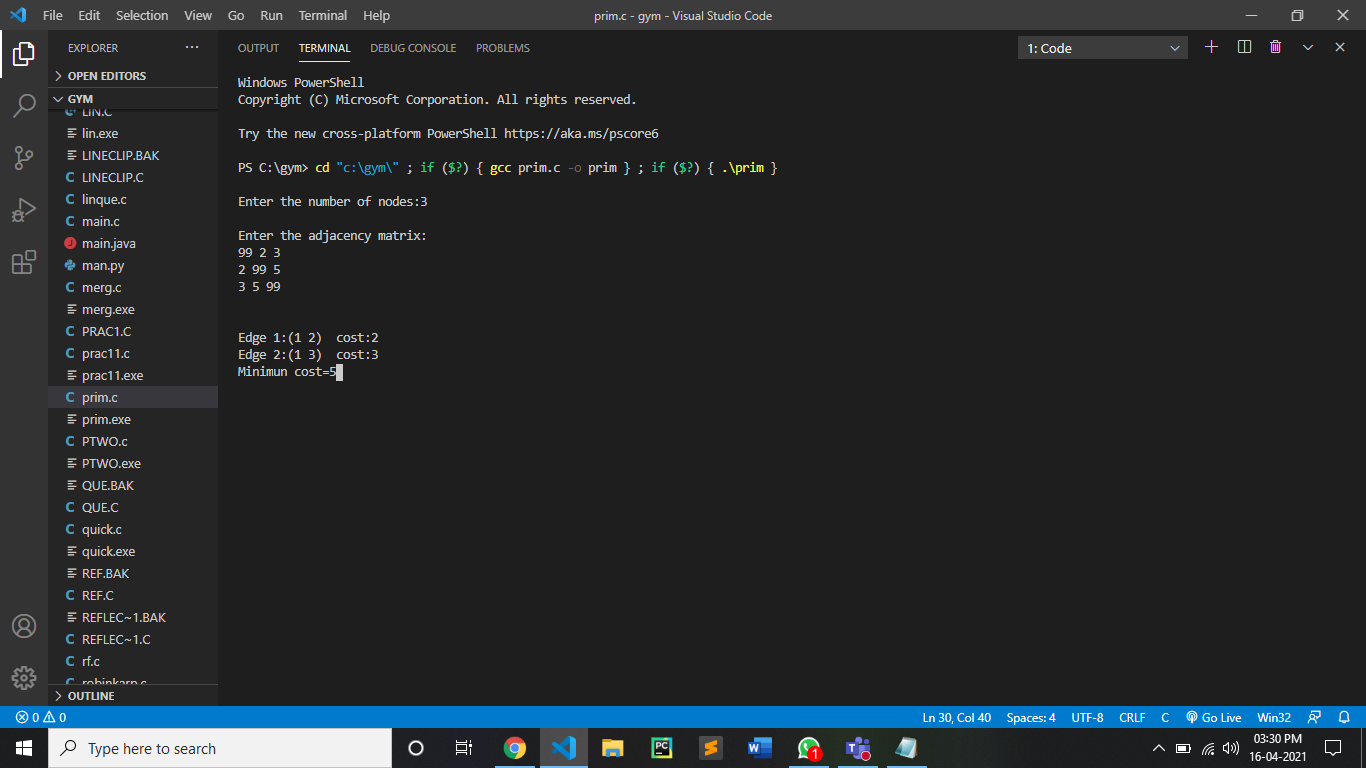
}

printf("\nMinimun cost=%d",mincost);

getch();

}

**OUTPUT:**



KRUSKAL’S ALGORITHM:

**CODE:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

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 cost 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 Minimum Cost Spanning Tree are\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("\nMinimum 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;

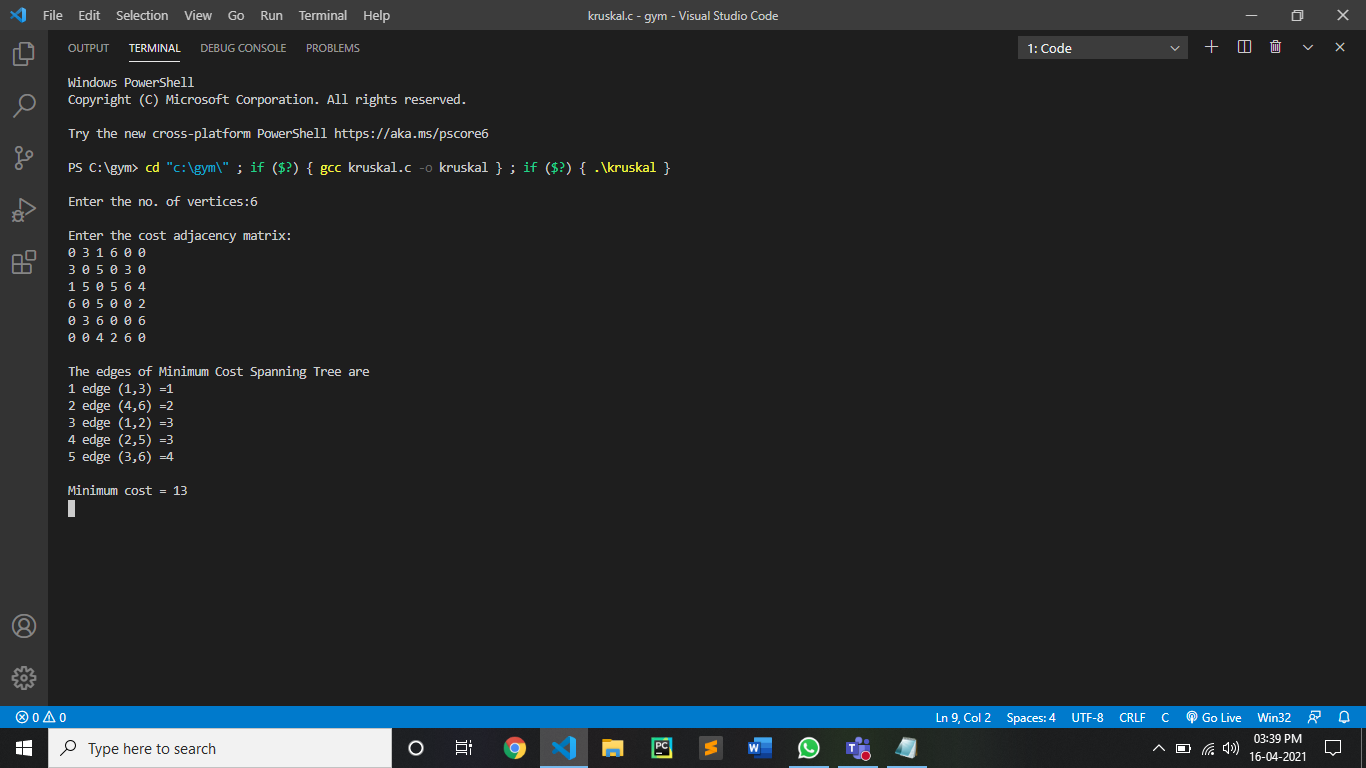
return 1;

}

return 0;

}

**OUTPUT:**



CONCLUSION:

By performing above practical we can conclude the following points:

* Prim’s algorithm has a time complexity of O(V2), V being the number of vertices and can be improved up to O(E + log V) using Fibonacci heaps.
* Kruskal’s algorithm’s time complexity is O(E log V), V being the number of vertices.
* Prim’s algorithm gives connected component as well as it works only on connected graph.
* Kruskal’s algorithm can generate forest(disconnected components) at any instant as well as it can work on disconnected components
* prim's algorithm does not need sorted edges
* kruskal's algorithm need sorted edges
* Prim’s algorithm runs faster in dense graphs.
* Kruskal’s algorithm runs faster in sparse graphs.