**EXPERIMENT N0-8**

**AIM: Write a program to find the single source shortest path using Dynamic Programming.**

**CODE:**

#include<stdio.h>

#include<stdlib.h>

int Bellman\_Ford(int DistMat[20][20], int numVertex, int E, int edge[20][2])

{

int i,u,v,k,distance[20],parent[20],S,flag=1;

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

{

distance[i]=1000,parent[i]=-1;

}

printf("Enter Source Vertex Number :-->");

scanf("%d",&S);

distance[S-1]=0;

for(i=0;i<numVertex-1;i++)

{

for(k=0;k<E;k++)

{

u=edge[k][0], v=edge[k][1];

if(distance[u]+DistMat[u][v]<distance[v])

{

distance[v]=distance[u]+DistMat[u][v] ,

parent[v]=u;

}

}

}

for(k=0;k<E;k++)

{

u=edge[k][0], v=edge[k][1];

if(distance[u]+DistMat[u][v]<distance[v])

{

flag=0;

}

}

if(flag)

{

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

{

printf("VERTEX %d -> COST = %d, PARENT= %d\n",i+1,distance[i],parent[i]+1);

}

}

return flag;

}

int main()

{

int numVertex,edge[20][2],DisMat[20][20],i,j,k=0;

printf("ENTER THE NUMBER OF VERTICES: ");

scanf("%d",&numVertex);

printf("ENTER DISTANCE MATRIX:\n");

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

{

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

{

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

if(DisMat[i][j]!=0)

{

edge[k][0]=i,edge[k++][1]=j;

}

}

}

if(Bellman\_Ford(DisMat,numVertex,k,edge))

{

printf("\nNO NEGATIVE WEIGHT CYCLE FOUND ");

}

else{

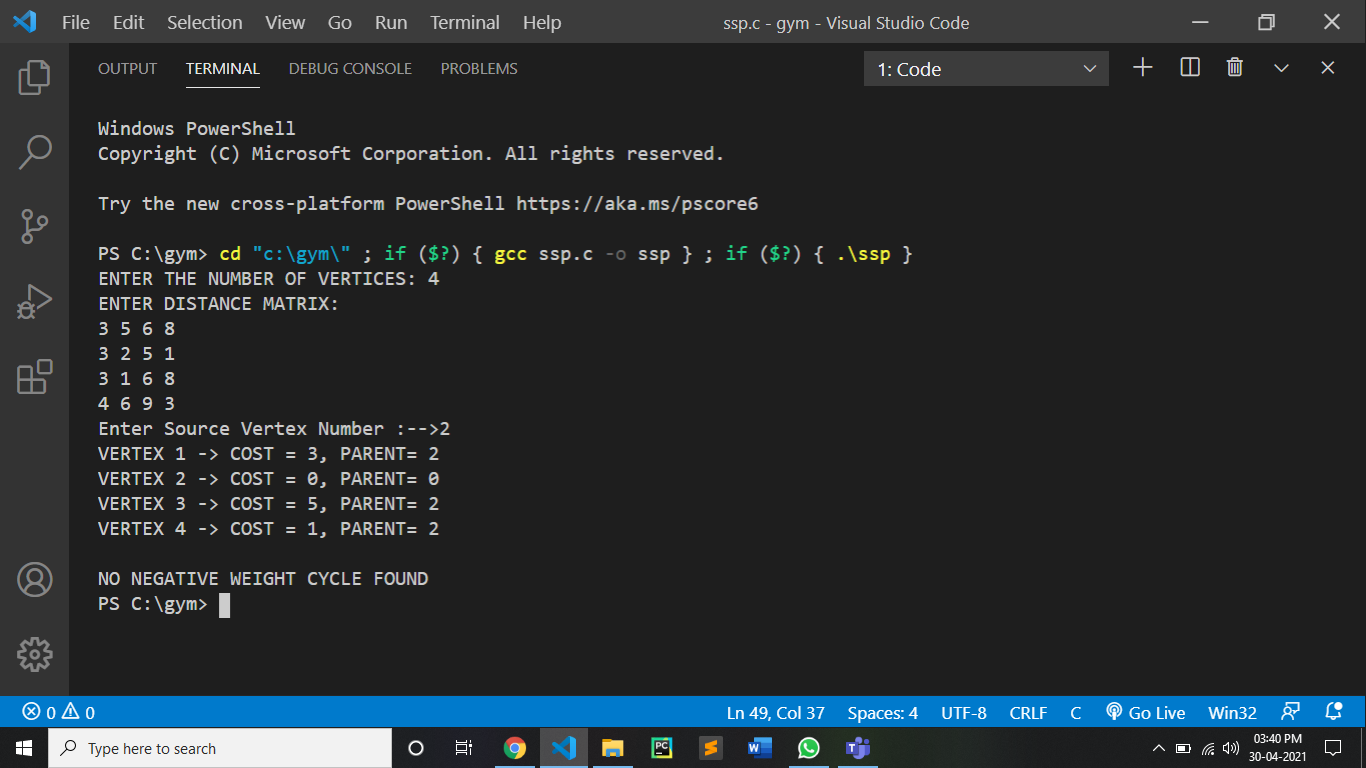
printf("\nNEAGTIVE WEIGHT CYCLE EXISTS");

}

return 0;

}

**OUTPUT:**



CONCLUSION:

By performing the above experiment we can conclude it is slower than dijkstra’s algorithm but it can handle negative weight cycle.

The time complexity of algorithm is O(|V|.|E|) where the V is the number of vertices and E is the number of Edges. And the space complexity is O(V)