**EXPERIMENT N0- 9**

**AIM: Implementation of Travelling Salesman Problem using Branch & Bound.**

**CODE:**

#include<stdio.h>

#include<conio.h>

int DistanceMatrix[10][10],VisitedCities[10],n,cost=0;

void getData()

{

int i,j;

printf("\n\nEnter Number of Cities :- ");

scanf("%d",&n);

printf("Enter (%d x %d) Distance Matrix : \n",n,n);

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

{

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

{

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

}

VisitedCities[i]=0;

}

printf("\n\nThe Distance Matrix is : \n");

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

{

printf("\n\n");

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

{

printf("\t%d",DistanceMatrix[i][j]);

}

}

}

void mincost(int city){

int i,ncity;

VisitedCities[city] = 1;

printf("%d--> ",city+1);

ncity=least(city);

if(ncity==999){

ncity=0;

printf("%d",ncity+1);

cost += DistanceMatrix[city][ncity];

return;

}

mincost(ncity);

}

int least(int c){

int i,nc=999;

int min=999,kmin;

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

if((DistanceMatrix[c][i]!=0) && (VisitedCities[i]==0))

if(DistanceMatrix[c][i]<min){

min = DistanceMatrix[i][0] + DistanceMatrix[c][i];

kmin=DistanceMatrix[c][i];

nc=i;

}

}

if(min != 999)

{

cost += kmin;

}

return nc;

}

void DisplayPath(){

printf("\n\nMinimum Cost:");

printf("%d",cost);

}

int main(){

getData();

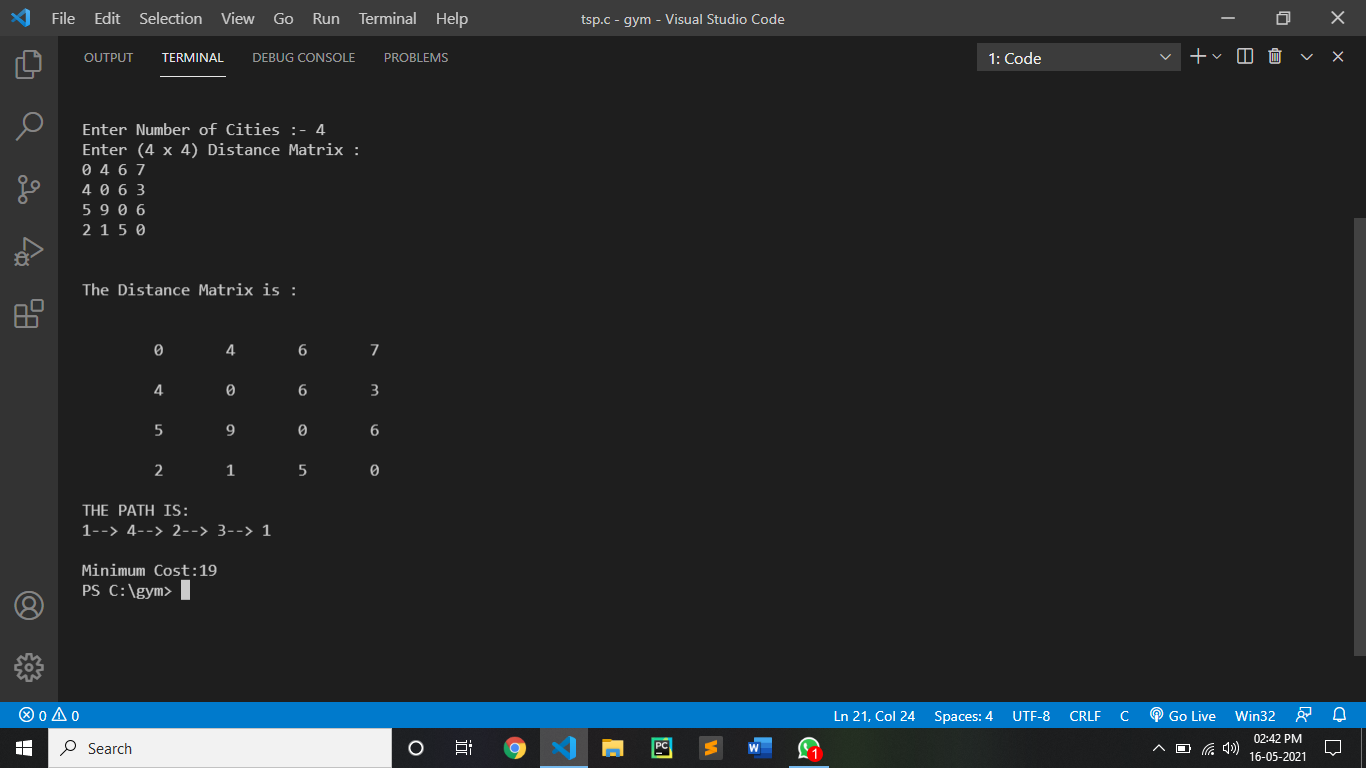
printf("\n\nTHE PATH IS: \n");

mincost(0);

DisplayPath();

}

**OUTPUT:**



CONCLUSION:

By performing the Travelling Salesman problems we can that:

•The worst case complexity of Branch and Bound remains same as that of the Brute Force clearly because in worst case, we may never get a chance to prune a node.

•But in practice it performs very well depending on the different instance of the TSP.

•The complexity also depends on the choice of the bounding function as they are the ones deciding how many nodes to be pruned.

•The time complexity of the program is O(n^2) as explained above for the row and column reduction functions.