Experiment 6

Aim: Implement Travelling Salesman Problem (TSP) using Dynamic Programming.

6.1 CO Attained:
CO2, CO3 and CO4

6.2 Objective:

Given a set of cities and the distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point.

6.3 Resources:
Turbo c/Dev C++

6.4 Program Logic:

traveling salesman problem, an optimization problem in graph theory in which the nodes (cities) of a graph are connected by directed edges (routes), where the weight of an edge indicates the distance between two cities.

```
Data: $s$: starting point; $N$: a subset of input cities; $dist()$:
      distance among the cities
Result: $Cost$ : TSP result
$Visited[N] = 0$;
$Cost = 0$;
Procedure TSP($N$, $s$)
    $Visited[s] = 1$;
    if $|N| = 2$ and $k \neq s$ then
        $Cost(N, k) = dist(s, k)$;
        Return Cost;
    else
        for $j \in N$ do
            for $i \in N$ and $visited[i] = 0$ do
                if $j \neq i$ and $j \neq s$ then
                    $Cost(N, j) = \min$ ( $TSP(N - \{i\}, j) + dist(j, i)$)
                    $Visited[j] = 1$;
                end
            end
        end
    end
    Return $Cost$;
end
```

6.5 Procedure:

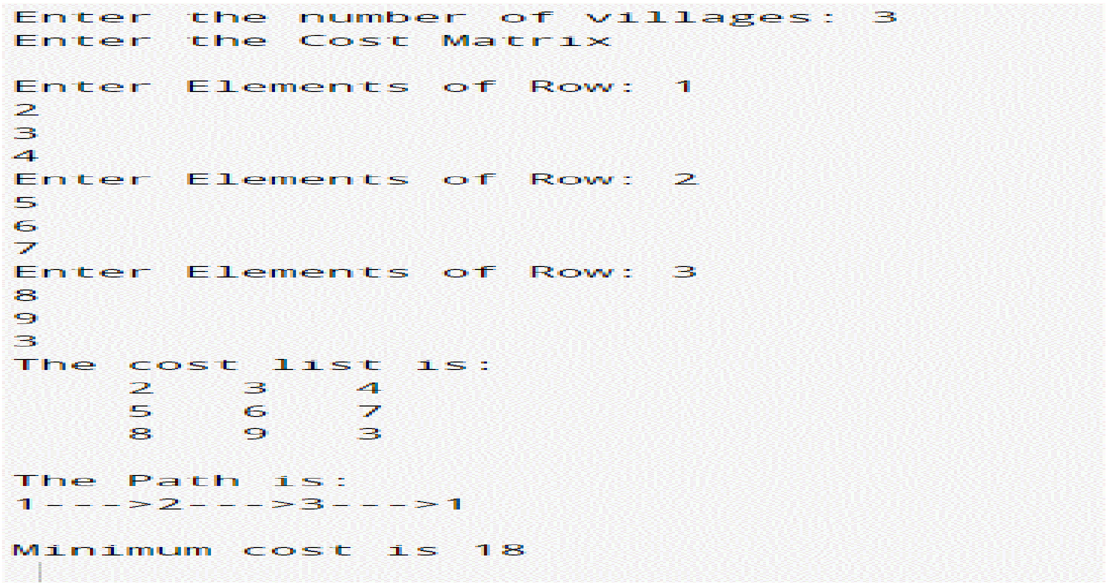
1. Create: Open Dev C++/C and write a program after that save the program with the .c extension.
2. Compile: Alt + F9

3. Execute: Ctrl + F10

6.6 Program Code:

```
#include<stdio.h>
int ary[10][10], completed[10], n, cost=0;
void takeInput()
{
int i, j;
printf("Enter the number of villages: ");
scanf("%d",&n);
printf("\nEnter the Cost Matrix\n");
for(i=0;i <n;i++)
{
printf("\nEnter Elements of Row: %d\n",i+1);
for( j=0;j <n;j++)
scanf("%d",&ary[i][j]);
completed[i]=0;
}
printf("\n\nThe cost list is:");
for( i=0;i < n;i++)
{
printf("\n");
for(j=0;j <n;j++)
printf("\t%d",ary[i][j]);
} }
void mincost(int city)
{
int i, ncity;
completed[city]=1;
printf("%d--->",city+1);
ncity=least(city);
if(ncity==999)
{
ncity=0;
printf("%d",ncity+1);
cost+=ary[city][ncity];
return;
}
mincost(ncity);
}
int least(int c)
{
int i,nc=999;
int min=999,kmin;
for(i=0;i <n;i++)
{
if((ary[c][i]!=0)&&(completed[i]==0))
if(ary[c][i]+ary[i][c] < min)
{
min=ary[i][0]+ary[c][i];
kmin=ary[c][i];
nc=i;
}}
if(min!=999)
cost+=kmin;
return nc;
}
int main()
{
takeInput();
printf("\n\nThe Path is:\n");
mincost(0); //passing 0 because starting vertex
printf("\n\nMinimum cost is %d\n ",cost);
return 0;}
```

6.7Conclusion:



```
Enter the number of villages: 3
Enter the Cost Matrix

Enter Elements of Row: 1
2
3
4
Enter Elements of Row: 2
5
6
7
Enter Elements of Row: 3
8
9
3
The cost list is:
        2      3      4
        5      6      7
        8      9      3

The Path is:
1--->2--->3--->1

Minimum cost is 18
```

6.8Analysis:

Dynamic programming creates $n.2^n$ subproblems for n cities. Each sub-problem can be solved in linear time. Thus the time complexity of TSP using dynamic programming would be $O(n^2 2^n)$. It is much less than n! but still, it is an exponent. Space complexity is also exponential.

6.9Lab Viva Questions:

1. What is the most efficient approach to solving this problem?
2. Define the Dynamic approach.
3. Define Travelling Salesman Problem.