Class: SE_Computer Subject: AOA RollNo_Name: 55_AdnanShaikh

EXPERIMENT NO- 06

AIM: Implementation of travelling salesman problem using dynamic programming.

PROBLEM STATEMENT : Write a program to implement the Travelling Salesman Problem (TSP) using dynamic programming.

RESOURCE REQUIRED: Pentium IV, Turbo C, Printer, Printout Stationary

THEORY:

Travelling Salesman Problem: - Definition: -
Given a set of cities and the distance between each possible pair, the Travelling Salesman Problem is to find the best possible way of „visiting all the cities exactly once and returning to the starting point‟.
The (original) problem can be formulated as:"If there are n cities a salesman must visit, and the distance between each pair of these cities is given, find the shortest tour where each city is visited exactly once and returning to your starting point."

Dynamic Programming
Dynamic programming is typically applied to optimization problems. For each given problem, we may get any number of solutions we seek for optimum solution (i.e. minimum value or maximum value solution). And such an optimal solution becomes the solution to the given problem. It is guaranteed that the dynamic programming will generate optimal solution using principle of optimality.
Problem Description: Let G be directed graph denoted by (V, E) where V denotes set of vertices and E denotes set of edges. The edges are given along with their cost Cij. The cost Cij>0 for all i and j. If there is no edge between i and j then Cij=∞.
A tour for a graph should be such that all the vertices should be visited only once and cost of the tour is sum of the cost of edges on the tour. The travelling salesperson problem is to find the tour of minimum cost.

Dynamic programming is used to solve this problem.
Step 1: Let the function C (1, V-{1}) is the total length of the tour terminating at 1. The objective of TSP problem is that the cost of this tour should be minimum.
Let d [i, j] be the shortest path between two vertices i and j.
Step 2: Let V1, V2… Vn be the sequence of vertices followed in optimal tour. Then (V1, V2,… Vn) must be a shortest path from V1 to Vn which passes through each vertex exactly
once.
Here the principle of optimality is used. The path Vi, Vi+1,…Vj must be optimal for all paths beginning at v(i), ending at v(j),and passing through all the intermediate vertices{V(i+1)..V(j-1)} once.
Step 3: Following formula can be used to obtain the optimum cost tour. Cost(i,S) = min{d[i,j] + Cost(j,S-{j})} where j€S and i€S

ALGORITHM:

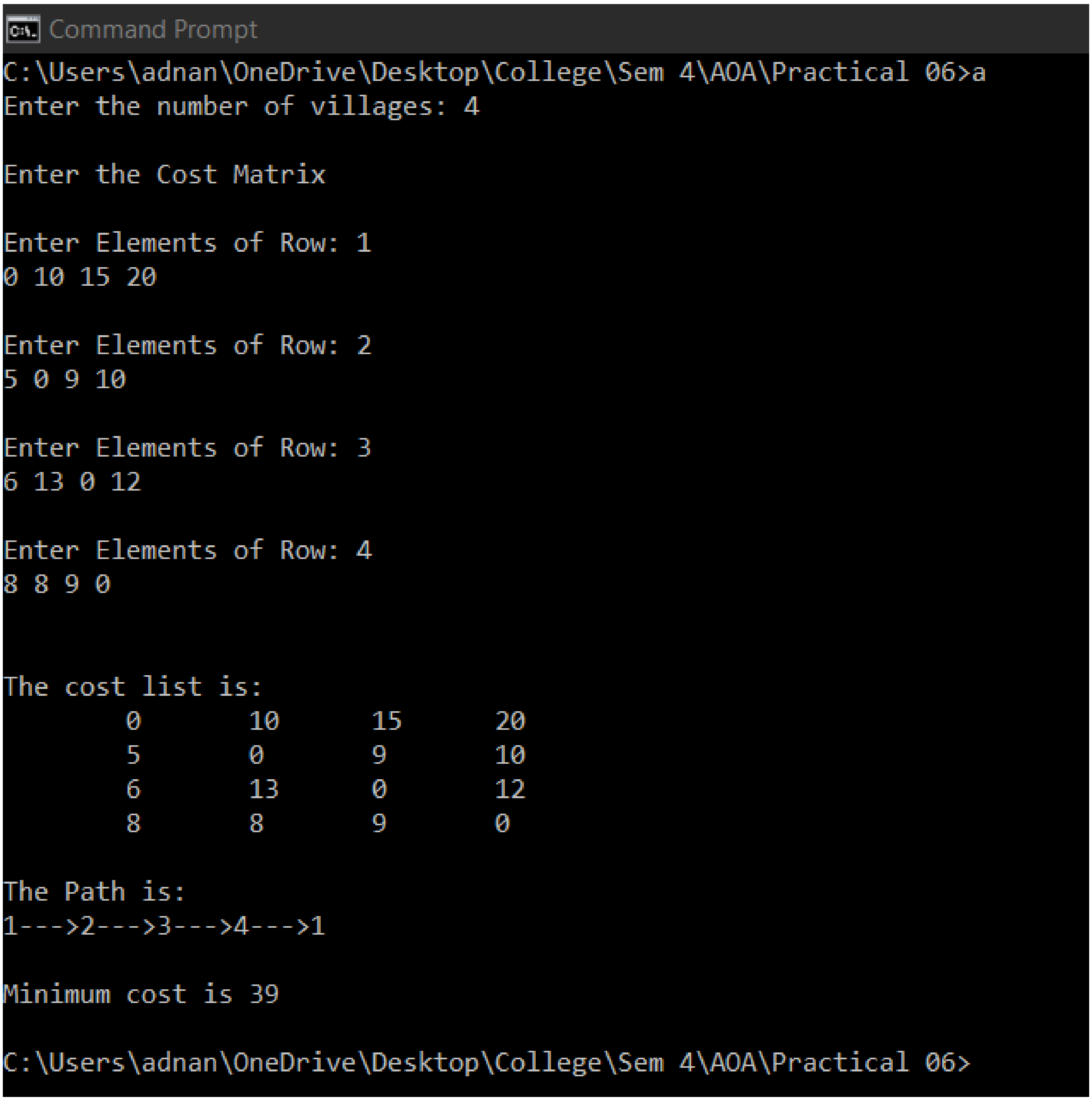
1. Scan the number of vertices to be traversed.
2. Maintain the list of distances from every vertex to every other possible vertex.
3. Store the source vertex from the list.
4. If there is no path between two cities then put zero.

5. Go through all the vertices.
6. Recursively record the path of traversal.
7. Calculate the minimum path each time and update it.
8. Select the new path with the minimum cost as your optimal path.
9. Display the path and the cost related to it.

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;
	int least();
	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;
}
```

OUTPUT:



```
C:\Users\adnan\OneDrive\Desktop\College\Sem 4\AOA\Practical 06>a
Enter the number of villages: 4

Enter the Cost Matrix

Enter Elements of Row: 1
0 10 15 20

Enter Elements of Row: 2
5 0 9 10

Enter Elements of Row: 3
6 13 0 12

Enter Elements of Row: 4
8 8 9 0


The cost list is:
        0       10      15      20
        5       0       9       10
        6       13      0       12
        8       8       9       0

The Path is:
1--->2--->3--->4--->1

Minimum cost is 39

C:\Users\adnan\OneDrive\Desktop\College\Sem 4\AOA\Practical 06>
```

CONCLUSION: Given n is the number of cities to be visited, the total number of possible routes covering all cities can be given as a set of feasible solutions of the TSP and is given as (n-1)! /2.