ARTIFICIAL INTELLIGENCE LAB

ASSIGNMENT – 9

TRAVELLING SALES PERSON PROBLEM

NAME : PRATHAPANI SATWIKA

REG.NO. : 20BCD7160

In this problem the salesman starts at one point from one place and travels all the places compare to this starting point or place. the objective of this problem is to minimize or optimize the cost. the main requirement communication between two nodes or cities.

CODE :

```
import java.util.*;  class
Travellingsalesmanproblem
{       static int findHamiltonianCycle(int[][] distance, boolean[] visitCity, int currPos, int cities, int count, int cost, int hamiltonianCycle)
{
if (count == cities && distance[currPos][0] > 0)
{
   hamiltonianCycle = Math.min(hamiltonianCycle, cost + distance[currPos][0]);
return hamiltonianCycle;
}  for (int i = 0; i < cities;
i++)
{    if (visitCity[i] == false && distance[currPos][i] >
0)
{    visitCity[i] =
true;
hamiltonianCycle =
```

```
findHamiltonianCyc
le(distance,
visitCity, i,
cities,count+  1, cost
+
distance[currPos][i],
hamiltonianCycle);
visitCity[i] = false;
} }  return
hamiltonianCycle;
}  public static void main(String[]
args)
{
   int cities;
Scanner sc = new Scanner(System.in);
System.out.println("Enter total number of cities : ");
cities = sc.nextInt();  int distance[][] = new
int[cities][cities];  for( int i = 0; i < cities; i++){
for( int j = 0; j < cities; j++){
System.out.println("Distance from city"+ (i+1) +" to city"+ (j+1) +": ");
distance[i][j] = sc.nextInt();
} }  boolean[] visitCity = new boolean[cities];  visitCity[0] = true; int
hamiltonianCycle = Integer.MAX_VALUE; hamiltonianCycle =
findHamiltonianCycle(distance, visitCity, 0, cities, 1, 0, hamiltonianCycle);
System.out.println(hamiltonianCycle);
}
```

}

```
import java.util.*;
class Travellingsalesmanproblem
{
    static int findHamiltonianCycle(int[][] distance, boolean[] visitCity, int currPos, int cities,
        int count, int cost, int hamiltonianCycle)
{
if (count == cities && distance[currPos][0] > 0)
{
    hamiltonianCycle = Math.min(hamiltonianCycle, cost + distance[currPos][0]);
    return hamiltonianCycle;
}  for (int i = 0; i < cities; i++)
{    if (visitCity[i] == false && distance[currPos][i] > 0)
{    visitCity[i] = true;
hamiltonianCycle = findHamiltonianCycle(distance, visitCity, i, cities,count+  1, cost +
    distance[currPos][i], hamiltonianCycle);
visitCity[i] = false;
}

}
return hamiltonianCycle;
}
public static void main(String[] args)
{
    int cities;
Scanner sc = new Scanner(System.in);
System.out.println("Enter total number of cities : ");
cities = sc.nextInt();
int distance[][] = new int[cities][cities];
for( int i = 0; i < cities; i++){
    for( int j = 0; j < cities; j++){
System.out.println("Distance from city"+ (i+1) +" to city"+ (j+1) +": ");
distance[i][j] = sc.nextInt();
}
```

```
 }
 boolean[] visitCity = new boolean[cities];
 visitCity[0] = true;
 int hamiltonianCycle = Integer.MAX_VALUE;
 hamiltonianCycle = findHamiltonianCycle(distance, visitCity, 0, cities, 1, 0, hamiltonianCycle);
 System.out.println(hamiltonianCycle);
 }
 }

```

OUTPUT :

```
Enter total number of cities : 3
Distance from city1 to city1:
14
Distance from city1 to city2: 15
Distance from city1 to city3:
18
Distance from city2 to city1:
20
Distance from city2 to city2:
22
Distance from city2 to city3:
26
Distance from city3 to city1: 25
Distance from city3 to city2:
33
Distance from city3 to city3:
18
66
```

```
Enter total number of cities : 4
Distance from city1 to city1:
14
Distance from city1 to city2:
15
Distance from city1 to city3:
88
Distance from city1 to city4:
45
Distance from city2 to city1:
19
Distance from city2 to city2:
33
Distance from city2 to city3:
25
Distance from city2 to city4:
17
Distance from city3 to city1: 19
Distance from city3 to city2:
13
Distance from city3 to city3:
55
Distance from city3 to city4:
30
Distance from city4 to city1:
90
Distance from city4 to city2: 86
Distance from city4 to city3:
22
Distance from city4 to city4:
14
73
```