EXPERIMENT-06

NAME-ATHARVA PALIWAL

ROLL NO. - 40

Problem statement:

Implement a program that prints all the Hamiltonian cycles present in a connected graph.

CODE-

```
package daa_lab;
/**
 *
 * @author Atharva Paliwal
 */

import java.util.Scanner;
import java.util.Arrays;

public class exp6
{
    private int numberOfVertices, pathCount;
    private int[] path;
    private int[][] graph;
```

```
//This function finds the cycle
    public void findHamiltonianCycle(int[][] g)
    {
        numberOfVertices = g.length;
        path = new int[numberOfVertices];

        Arrays.fill(path, -1);
        graph = g;
        try
        {
            path[0] = 0;
            pathCount = 1;
            findPathrecursively(0);
            System.out.println("--- No solution found!! ---");
        }
        catch (Exception e)
        {
            System.out.println(e.getMessage());
            display();
        }
    }
    //This function finds the paths recursively
    public void findPathrecursively(int vertex) throws Exception
    {

        if (graph[vertex][0] == 1 && pathCount == numberOfVertices)
            throw new Exception("\n--- Solution found !! ---\n");
```

```
    if (pathCount == numberOfVertices)
        return;

    for (int v = 0; v < numberOfVertices; v++)
    {
        //if vertex is connected
        if (graph[vertex][v] == 1 )
        {
            //add to path
            path[pathCount++] = v;
            //remove connection
            graph[vertex][v] = 0;
            graph[v][vertex] = 0;

            //if vertex not already selected  solve recursively
            if (!isPathSelected(v))
                findPathrecursively(v);

            //restore connection
            graph[vertex][v] = 1;
            graph[v][vertex] = 1;
            //remove path
            path[--pathCount] = -1;
        }
    }
}

// This function checks if path is already selected
```

```
public boolean isPathSelected(int v)
{
    for (int i = 0; i < pathCount - 1; i++)
        if (path[i] == v)
            return true;
    return false;
}

// This function prints the path
public void display()
{
    System.out.println("------- HAMILTONIAN PATH -------");
    System.out.print("\nPath : ");
    for (int i = 0; i <= numberOfVertices; i++){
        System.out.print((path[i % numberOfVertices] + 1 ));
        if(i != numberOfVertices)
            System.out.print(" --> ");
    }
    System.out.println();
}


public static void main (String[] args)
{
    Scanner sc = new Scanner(System.in);

    exp6 obj = new exp6();
```

```
        System.out.println("Enter number of vertices : ");
        int numberOfVertices = sc.nextInt();

        System.out.println("\nEnter adjacency matrix\n");
        int[][] graph = new int[numberOfVertices][numberOfVertices];

        for (int i = 0; i < numberOfVertices; i++)
            for (int j = 0; j < numberOfVertices; j++)
                graph[i][j] = sc.nextInt();

        long startTime = System.nanoTime();
        obj.findHamiltonianCycle(graph);
        long endTime = System.nanoTime();
        long timeElapsed = endTime - startTime;
        System.out.println("----------------------------------------
--------");
        System.out.println("Execution time in seconds : " +timeElapsed
+ " nanoseconds");
        System.out.println("----------------------------------------
--------");
    }
}
```

OUTPUT –

```
Enter number of vertices :
4
Enter adjacency matrix
0 1 0 1
1 0 0 1
1 1 0 0
1 1 1 1
                --- Solution found !! ---
------- HAMILTONIAN PATH -------
Path : 1 --> 2 --> 4 --> 3 --> 1
-------------------------------------------------
Execution time in seconds : 17341600 nanoseconds
-------------------------------------------------

Enter number of vertices :
6
Enter adjacency matrix
1 0 1 0 0 1
0 1 0 0 0 1
0 1 1 0 0 1
0 0 0 0 0 1
1 0 0 0 0 0
1 1 1 1 1 1
           --- No solution found!! ---
-------------------------------------------------
Execution time in seconds : 1114800 nanoseconds
-------------------------------------------------
```

```
Enter number of vertices :
10
Enter adjacency matrix
0 1 0 1 1 0 1 0 1 0
1 0 1 0 1 0 1 0 0 0
0 1 0 0 0 1 1 1 0 1
1 0 0 0 1 0 0 0 1 0
1 1 0 1 0 1 0 0 0 0
0 0 1 0 1 0 0 1 0 0
1 1 1 0 0 0 0 0 1 1
0 0 1 0 0 1 0 0 0 1
1 0 0 1 0 0 1 0 0 0
0 0 1 0 0 0 1 1 0 0
            --- Solution found !! ---
------- HAMILTONIAN PATH -------
Path : 1 --> 2 --> 3 --> 6 --> 8 --> 10 --> 7 --> 9 --> 4 --> 5 --> 1
------------------------------------------------
Execution time in seconds : 336800 nanoseconds
------------------------------------------------
```

NUMBER OF VERTICES	TIME TAKEN(IN NS)
4	17341600
6	1114800
10	336800

ANALYSIS :

In the above Java Program we implemented Hamiltonian Cycle Algorithm. Hamiltonian cycle is a path in a graph that visits each vertex exactly once and back to starting vertex. This program is to determine if a given graph is a hamiltonian cycle or not. This program assumes every vertex of the graph to be a part of hamiltonian path.