CS 6301.002 IMPLEMENTATION OF ADVANCED DATA STRUCTURES AND ALGORITHMS

LONG PROJECT 0 - FINDING EULER TOURS

SUBMITTED BY : GROUP 05

ABSTRACT:

The project is implemented to find the Euler tour or Euler path in an undirected Eulerian Graph, using Hierholzer's algorithm which is an O(|E|) running time algorithm. We have implemented the two methods which finds an Euler Tour and verifies its correctness.

```
static List<Edge> findEulerTour(Graph g)  // Return an Euler tour of g
static boolean verifyTour(Graph g, List<Edge> tour, Vertex start)  //
verify tour is a valid Euler tour
```

INTRODUCTION:

A graph G is called Eulerian if it is connected and the degree of every vertex is an even number. It is known that such graphs aways have a tour (a cycle that may not be simple) that goes through every edge of the graph exactly once. Such a tour (sometimes called a circuit) is called an Euler tour. If the graph is connected, and it has exactly 2 nodes of odd degree, then it has an Euler Path connecting these two nodes, that includes all the edges of the graph exactly once. In this project, write code that finds an Euler tour or an Euler Path in a given graph. Initial code base: Edge.java, Vertex.java, Graph.java, CheckBipartiteNew.java.

In addition, write a driver program that reads the input graph, finds a tour in it and prints out the tour using the format given below.

METHODOLOGY:

Hierholzer's Algorithm:

- Choose any starting vertex v, and follow a trail of edges from that vertex until returning to v. It is not possible to get stuck at any vertex other than v, because the even degree of all vertices ensures that, when the trail enters another vertex w there must be an unused edge leaving w. The tour formed in this way is a closed tour, but may not cover all the vertices and edges of the initial graph.
- As long as there exists a vertex u that belongs to the current tour but that has adjacent edges not part of the tour, start another trail from u, following unused edges until returning to u, and join the tour formed in this way to the previous tour.

By using a data structure such as a doubly linked list to maintain the set of unused edges incident to each vertex, to maintain the list of vertices on the current tour that have unused edges, and to maintain the tour itself, the individual operations of the algorithm (finding unused edges exiting each vertex, finding a new starting vertex for a tour, and connecting two tours that share a vertex) may be performed in constant time each, so the overall algorithm takes linear time, O(|E|). [8]

DEVELOPMENT PLATFORM:

The project was implemented and testes using the following tools:

Language: Java SE 1.7
IDE: Eclipse

TEST RESULTS:

Sample Input:

```
5 6
1 2 1
2 3 1
3 1 1
3 4 1
4 5 1
5 3 1
```

Sample Output:

```
Size 6
 01
 4
 4
 4
 4
 (1,2)
 (2,3)
 (3,4)
 (4,5)
 (5,3)
 (3,1)
 Time: 1 msec.
 Memory: 2 MB / 128 MB.
```

CONCLUSION:

We have implemented the Hierholzer's algorithm and its running time is found to be O(IEI).

REFERENCES:

1. https://en.wikipedia.org/wiki/Eulerian_path.
2. https://www-m9.ma.tum.de/graph-algorithms/hierholzer/index_de.html