CS 5V81.012 IMPLEMENTATION OF DATA STRUCTURES AND ALGORITHMS
MANDATORY PROJECT 1- EULER TOUR

ABOUT THE PROBLEM:

To find the Euler tour for the graph and to verify the tour.

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 that goes through every edge of the graph exactly once. Such a tour is called an Euler tour.

ABOUT THE ALGORITHM:

Heiholzer's Algorithm:

This algorithm for finding Euler tour works as follows:

- Choosing any arbitrary starting vertex, say v, follow the trail of edges from that vertex until returning to v. Since, Eulerian graph requires the degree of each vertex in the graph to be an even number, whenever the trail enters a vertex there must be an unvisited edge leaving from that vertex (except for v). This trail may or may not include all the vertices.

- 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.

DETAILS ABOUT IMPLEMENTATION:

Requirement #1:

CircularLinkedList.java implements public class CircularLinkedList<T> implements Iterable<T> {…..

Requirement #2:

- Graph.readGraph() reads the input graph either from the console or from the file

- To read the input from file, filename is to be given as command-line argument.

Requirement #3:

List<CircularLinkedList<Vertex>> breakGraphIntoTours(Graph g) is a function in EulerTour.java that takes the input graph and decomposes it to list of sub-tours.

Requirement #4:

CircularLinkedList<Vertex> stitchTours(List<CircularLinkedList<Vertex>> listOfTours) is a function in EulerTour.java that takes the list of tours and stitches them into one.

Requirement #5:

boolean verifyTour(Graph g, CircularList<Vertex> tour) is function in EulerTour.java that verifies if the given tour is a valid Euler tour for the given graph.

Requirement #6:

Driver.java reads the input graph, finds and verifies an Euler Tour for the graph.

Data Structure Decisions:

- To find the unvisited edge at each vertex in constant time, i.e., to avoid traversing through the entire list of edges at the vertex to find an unvisited edge, the edges are initially copied to the queue named 'adjacency' which at any point in time will hold the unvisited edges at that vertex.

- While generating the list of sub-tours, for every tour i,the exact place in tour (i-1) is to be remembered to stitch the tours, where i>0. A Queue is maintained to remember the points in the previous tour where the next tour is to be stitched.

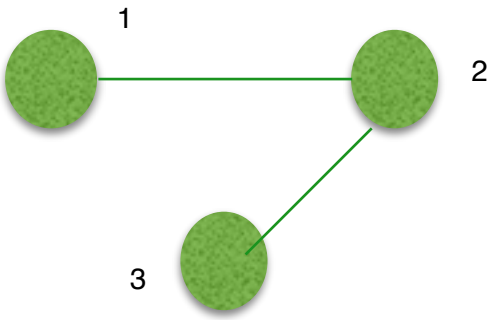
- To verify the tour, an edge between vertex i and vertex i+1 in the list is to be retrieved in constant time. A hash table is maintained on edges which stores the edge as both key and value. The key is generated only on from and to vertex of the edge, in order to retrieve the exact object while verification. hashCode() and equals() are methods overridden from java.lang.object.

ANALYSIS ON RUNNING TIME FOR LARGE INPUTS:

INPUT	NUMBER OF NODES	NUMBER OF EDGES	EULERIAN (Yes / No)	TIME FOR FINDING TOUR (secs)	TIME FOR VERIFYING THE TOUR (secs)
mp1-ck.txt	100	1050	Yes	0	0
mp1-k5k.txt	1000	5506	Yes	0	0
mp1-big.txt	500000	5249924	Yes	1	3

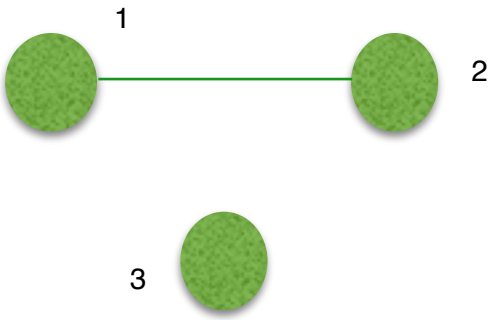
The program takes 4 seconds to find and verify an Euler tour for the graph with 500000 nodes and 5249924 edges. (Excluding the time to red the input and to print the output)

SAMPLE SCENARIOS:

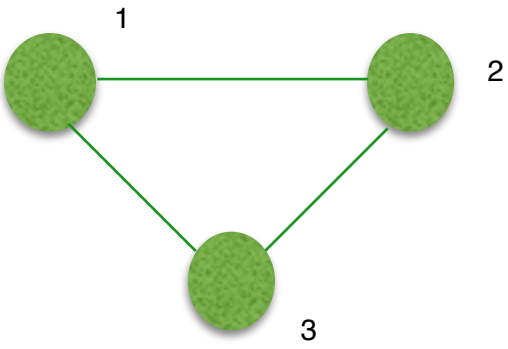


OUTPUT:

GRAPH IS NOT EULERIAN



GRAPH IS NOT EULERIAN



1

2

3