Mandatory Project 1: Finding Euler Tours

Project Report

Niveditha Varadha Chandrasekaran

10/2/2016

CONTENTS

1. ABSTRACT

The aim of this project is to find an Euler tour for a given undirected graph G = (V, E), if the graph is Eulerian.

2. PROBLEM STATEMENT

For a graph G = (V, E), a tour is a sequence of edges starting and ending at the same place. A tour is called as **Euler tour** if every edge of the graph G appears exactly once in it. Such a graph is called as **Eulerian graph**.

A graph G is called Eulerian and has an Euler tour if and only if:

a. Graph G is connected.
b. The degree of every vertex of the graph is even (i.e. number of edges incident to it.)

This project aims at writing a program that takes an undirected graph G = (V, E) as input and outputs its Euler tour

3. METHODOLOGY

a. Finding if the given graph is Eulerian or not:
The degree for every vertex of the graph is calculated. The graph is Eulerian if the degree of the vertices of the graph is non-zero and is even. If this condition is not satisfied for any vertex of the graph then it is not Eulerian.

b. Breaking the graph into sub tours:
Once the graph is verified to be Eulerian, we have to break the graph into sub tours. Each sub tour is stored as a circularly linked list where every node of the list has the vertex and the edge used to traverse to the next vertex. The sub tours are maintained as a list of circularly linked list. The sub tours are found by starting at the first vertex of the graph and maintaining a Queue for newly discovered vertices. During the process of finding sub tours, as we encounter an unseen vertex it is added to the queue and if the vertex has unvisited edges then the tour takes the next unvisited edge of the vertex, marks it as seen and adds the vertex and the edge used to the circular list which has the sub tour and proceeds to vertex that is at the other end of the edge just visited. Each sub tour ends when entering a node for which there are no more unused edges left.

c. Stitching the sub tours to get the Euler tour:
In this phase, we stitch the sub tours obtained from the previous step into a single tour. This is done by maintaining an index for nodes. The first sub tour is taken from the list of sub tours. Iterate through the rest of the list and merge each of the sub tour with the first tour by using the index to get O(E) time for the stitch.

d. Verification of the Euler tour:

The Euler tour is verified by iterating through the nodes of the circular list that has the tour. As we iterate through the nodes, we get the edge used and the existence of that edge in the graph is verified using the name of the edge and is marked as seen and we maintain a count for the number of edges. If an already seen edge is encountered again or if an edge that does not exist in the graph given is traversed or if the total number of edges encountered does not match with number of edges in the graph then the tour is not valid.

4. DEVELOPMENT PLATFORM

IDE: Eclipse. Version: Mars.1 Release (4.5.1)
Java Version: 1.8
Operating System: Windows 10

5. SAMPLE INPUT/OUTPUT

Sample input:

```
6 10
1 2 1
1 3 1
1 4 1
1 6 1
2 3 1
3 6 1
3 4 1
4 5 1
4 6 1
5 6 1
```

Sample output:

```
Time taken to read graph:Time: 25 msec.
Memory: 1 MB / 123 MB.
Time taken to test if the given graph has Euler tour or not:Time: 1 msec.
Memory: 1 MB / 123 MB.
Number of Sub tours: 2
Time taken to break the graph into sub tours:Time: 6 msec.
Memory: 1 MB / 123 MB.
Time taken to stitch the sub tours:Time: 0 msec.
Memory: 1 MB / 123 MB.
Euler Tour:
1
2
3
1
4
5
6
```

```
4
3
6
Time taken to print the Euler tour:Time: 1 msec.
Memory: 1 MB / 123 MB.
The tour is a valid Eulerian tour.
Time taken to verify the Euler tour:Time: 4 msec.
Memory: 1 MB / 123 MB.
```

6. ANALYSIS OF RESULT

mp1-ck			
	Running time (ms)	**Memory Used(MB)**	**Memory Total(MB)**
Read graph	112	3	123
Check for Eulerian	1	3	123
Break into sub tours	13	4	123
Stitch tours	0	4	123
Validate tour	9	4	123
mp1-k5k			
	Running time (ms)	**Memory Used(MB)**	**Memory Total(MB)**
Read graph	252	14	123
Check for Eulerian	2	14	123
Break into sub tours	22	15	123
Stitch tours	1	15	123
Validate tour	25	15	123
mp1-big (if memory is not allocated)			
	Running time (ms)	**Memory Used(MB)**	**Memory Total(MB)**
Read graph	48790	663	996
Check for Eulerian	50	663	996
Break into sub tours	11286	787	1235
Stitch tours	0	787	1235
Validate tour	14643	893	1458
mp1-big (if memory is allocated)			
	Running time (ms)	**Memory Used(MB)**	**Memory Total(MB)**
Read graph	49483	1026	1364
Check for Eulerian	64	1026	1364
Break into sub tours	4739	1026	1364
Stitch tours	0	1061	1364
Validate tour	6752	1061	1818

mp1-big (if executed on another PC)			
	Running time (ms)	**Memory Used(MB)**	**Memory Total(MB)**
Read graph	26815	990	1351
Check for Eulerian	50	990	1351
Break into sub tours	3154	1062	1352
Stitch tours	0	1062	1352
Validate tour	4692	924	1855

7. CONCLUSION

An Euler tour for a graph can be obtained only if the graph is connected and every vertex of the graph has even number of edges. Also, appropriate use of data structures, efficient usage of iterators and indexes plays a key role in getting efficient running times as the input size increases.

8. REFERENCES

- https://en.wikipedia.org/wiki/Eulerian_path