ENEE641 Fall 2020: Project description

Due: December 15, 2020, 11:59 pm

1 Problem description

Euler path of an undirected (or directed) graph is a path that traverses all edges in the graph, exactly once. An *Euler tour* is an Euler path which starts and ends at the same vertex. A graph is said to be *simple* if it has no self-loops at any vertex and it does not have more than one edge between two vertices. An Euler tour exists in an undirected, simple graph if and only if:

- The graph is connected, and
- Each vertex has an even degree. (That is, each vertex has even number of edges incident on it.)

2 Objective

Let G = (V, E) be an undirected, connected, simple graph. The goal of this project is to find if an **Euler tour**(not Euler path) exists for given **undirected, connected, simple graphs**, and to print such a tour, if they do indeed have one. You have to implement the Euler tour algorithm in **C**, and analyze its run time complexity. Your algorithm should have both the best asymptotic time complexity and the best constant factors. (If you run into tension between the best constant factor of V as opposed to the constant factor for E, minimize the latter, since the set E is larger than or equal to V.

3 Input Format

The graph is stored in a adjacent list format. In each of the input graphs, vertices are numbered from 1 to $|V|$. Each row includes a vertex, followed by zero or more other vertices. For example:
1 3 6
2 3 5
3 1 2 4 5
4 3 5
5 2 3 4 6
6 1 5

This input should be interpreted as follows: the undirected edges of G are (1,3), (1,6), (2,3), (2,5), (3,4), (3,5), (4,5) and (5,6). The vertices following the source vertex are in arbitrary order. There are 5 input graphs provided.

4 Output

The following is expected from you:

1. Asymptotic analysis of the runtime of your code up to a constant factor, counting all the C commands in your code.
2. Experimental results for the inputs provided. Print the Euler tour (if it exists), the actual runtime of the program (using *clock* command), and the count of all C commands executed. (See Output format 6 for details. The extra code introduced just for counting of executed commands should be properly marked and should not be included in the counting.)

5 Deliverables

1. Source code (coupled, of course, with documentation) along with a separate overview. That is, your code should contain comments on what a function or a block does. Your code must be compatible with Glue (allowing us to run your code). Please name your file <your UID>.c. More information on accessing Glue: `https://terpconnect.umd.edu/login.html`.

2. Report (at most 4 page) illustrating the algorithm used, analysis of runtime complexity (in Θ, Ω, O notation) of the algorithm, and points [1,2] under 'Output' section above. The report should also contain **what the code does** and what **data structures** are used. Please name your pdf file <your UID>.pdf.

3. Submit your source code and output files to the TA (mayu1996@terpmail.umd.edu) by the due date.

4. You can submit a zipped file that contains all the related files. Your final submission should contain 1 source code file + 1 report pdf + 10 output files. If you are submitting a zipped file, name it

   <your UID>.zip.

6 Output Format

A.txt

The first line of **A.txt** should be either 0 or 1. It should be 0, if Euler tour does not exist for the given input graph and 1, if it exists.

If the first line is 1, the next line should print the Euler tour starting from **any** vertex in the graph. Make sure that your tour starts and ends at the same vertex!

For the example graph given in section 3, a correct output looks like this:

1
1 3 2 5 3 4 5 6 1

B.txt

Only for the input graph number 1("in1.txt"), output the following:
Vertex 1: total # operations(C commands) charged to Vertex 1 in decimal.
...
Vertex $|V|$: same
Edge (x, y): total # operations(C commands) charged to Edge (x, y) in decimal.
...
Edge (z, w): same
(Namely, for each of the vertices and edges, print out in decimal the total number of operations that have been charged to that item. The printed out numbers should match as closely as possible your asymptotic and constant factor analysis.)
Maximum number of operations charged to any single vertex is:...
Maximum number of operations charged to any single edge is:...
Total number of operations is:...

C.txt

For input graphs number 2, . . . , 5 only("in2.txt" to "in5.txt"):

Maximum number of operations charged to any single vertex is:...
Maximum number of operations charged to any single edge is:...
Total number of operations is:...

Note: All output files corresponding to an input graph have to be suffixed by the index number of the graph. For example, output files of 4^{th} graph should be named $A4.txt$ and $C4.txt$. These numbered output files need not be generated automatically, i.e., you can generate generic output files $A.txt$ and $C.txt$, and then, rename them manually according to the input graph number.

7 Notes

There are a total of 5 input graphs for this project. In your report, you only need to print out results for "in1.txt" in format A&B. For the rest of the input graphs, print out the results in format A&C.

8 Grading Policy

The grading will account for issues such as: correctness of code & explanation, complexity of code (use as few C commands as possible, constant factors matter!), correctness of asymptotic analysis, and correctness of counting.