ENEE 641 FALL 2017
Project Specification

Due November 20, 2017, 11:59 pm

1 Objective

Articulation points, bridges, and biconnected components(c.f. Problem 22-2, CLRS)
Let G = (V, E) be a connected, undirected graph. An **articulation point** of G is a vertex whose removal disconnects G. A **bridge** of G is an edge whose removal disconnects G. A **biconnected component** of G is a maximal biconnected subgraph(a biconnected graph is a connected and "nonseparable" graph, meaning that if any vertex were to be removed, the graph will remain connected. Therefore, a biconnected graph has no articulation points). As discussed in class, we can determine articulation points, bridges, and biconnected components using **depth-first search**. Now, it is your turn to implement these algorithms in C, and analyze their time requirement. Your algorithms should have both the best complexity and the best constant factor.

Important Note:

1. In the class slide presentation of Chapter 22, slide 17 defines biconnectivity as a binary relation on edges, which is an equivalence relation. This extends to biconnected components by including all the vertices that touch on the edges of an equivalence class. This definition implies that a single bridge is an equivalence class of one edge (or a biconnected component of 2 vertices).

2. In contrast, Problem 22-2 defines biconnected component as "a maximal set of edges such that any two edges in the set lie on a common simple cycle"; indeed item g seeks proof that the biconnected components of a graph partition only its non-bridge edges.

   In the programming assignment, students should follow the definition in the slides, or the definition above(see also Wikipedia `http://en.wikipedia.org/wiki/Biconnected_component`), but not the one in the text.

2 Input 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 2 3
2 1 3
3 1 2 4 7
4 3 5 6
5 4 6
6 4 5
7 3

This input should be interpreted as follows: the undirected edges of G are (1, 2), (1, 3), (2, 3), (3, 4), (3.7), (4, 5), (4, 6) and (5, 6). It is to be noted that the vertices following the source vertex are in sorted order.

3 Output

The following is expected from you:

1. Your 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. The report should contain what the code does and what data structures are used.

2. Asymptotic analysis of the runtime of your code up to a constant factor, counting all the C commands in your code.

3. Experimental results for the inputs provided. Report the articulation points, bridges, biconnected components, as well as the actual runtime of the program, counting all C command executed. (See Output format for details. The extra code introduced just for counting of executed commands, or sorting required for output formats below should be properly marked, and not included in the counting.)

4 Deliverables

1. Source code compatible on glue (allowing us to run your code), with the output files for the provided inputs. Please name your file <your UID>.c

2. Report (at most 4 page report) illustrating points [1,2,3] under Output above. Please name your pdf file <your UID>.pdf

3. Submit your source code and output files to the TA by the due date (ahari1@terpmail.umd.edu). Please *do not* zip files. Your final submission should contain (1 source code file + 1 report pdf + 28 output files).

5 Output Format

A(*Three* separated files)

- **Aa.txt**(articulation points):
  The output of your articulation points should be the corresponding vertex numbers, sorted in increasing order, and put in a line. In the example above, the output will be:
  3 4

- **Ab.txt**(bridges):
  The output of your bridges should be like this: each bridge in a line, the smaller vertex number first, then the larger. And then sort bridges lexicographically. In the example above, the output will be:
  3 4
  3 7

- **Ac.txt**(biconnected components):
  The output of your biconnected components shoud be like this: for each biconnected component, output its vertex numbers, sorted in increasing order, and one line for each biconnected component. In the example above, the output will be:
  1 2 3
  3 4
  3 7
  4 5 6

B.txt

Only for the input graph number 1("1.in"), 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 Vertex 1 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.)
Total number of operations charged to all vertices is:...
Total number of operations charged to all edges is:...
Total number of operations is:...

C.txt

For input graphs number 2, . . . , 7 only("2.in" to "7.in"):

Total number of operations charged to all vertices is:...
Total number of operations charged to all edges 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, out files of 5th graph should be named *Aa5.txt, Ab5.txt, Ac5.txt* and *C5.txt*.

6   Notes

There are a total of 7 input graphs for this project. In your report, you only need to print out results for "1.in" in format A&B(4 files in total). For the rest of the input graphs, print out the results in format A&C(also 4 files in total, per graph).

7   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.