COSC–254 Data Mining
Project 02 — Counting Triangles
Due: Monday, April 15, 2019, 1.59pm

Collaboration reminder: Please do not share your code with anyone and do not ask to see anyone's code. Please use the forum for *all* discussions, so any doubt can be clarified for everybody, and so everybody is on the same page.
No extensions will be given on this project.

Implementation

You will implement and evaluate two data stream algorithms for triangle counting, TRIÈST-BASE and TRIÈST-IMPR. The algorithms were described in class (see the slides on counting triangles). Additional information can be found in Sects. 4.1 and 4.2 of the journal article on TRIÈST.

You do not have to use Java for this project, but:

1. the input, output, and command line interface of your program must be *exactly the same* as the ones of the given support code;

2. it must be possible to compile (if necessary) and run your program on `romulus` without any special setup;

3. The `README.txt` file in your submission must give very precise instructions on how to compile, if necessary, and run your program.

Implementation details First of all, please download the support code and extract it from the archive. You are allow to modify *only* the files `TriestBase.java` and `TriestImpr.java` in the `src` directory, but *do not modify the signature of the methods in these classes.* Do not alter the code in any of the other provided files. You are allowed to add methods to the `TriestBase` and `TriestImpr` class, and to add new files for new classes that may be needed by your implementation.

By running `java Main` with different arguments, you can select at runtime which variant of the algorithm you want to use. Run `java Main -h` (even without any modification to the files) for details.

Input and Output Formats The input edge stream is represented as a plaintext file containing one undirected edge per line. Each edge is represented by two non-negative integers separated by a *single* white space, such as

```
1 5
4 9
5 10
```

For an example, see the input file ca-AstroPh.txt[1] in the input directory of the support files. You will use this file in the evaluation part of the project. We strongly suggest to *not* use this file for debugging and testing your code during development.

The output is a stream of estimates of the number of triangles in the graph, one for each time step, i.e., one for each new edge inserted in the graph. The output functionality is already implemented in the Main class, and you *must not modify it*.

Evaluation

You will create figures showing the behavior of the estimation of the number of triangles by plotting the estimation as a function of the time t (in the stream) (i.e., the x axis is the time t, and the y axis is the estimation of the number of triangles).

Using the ca-AstroPh.txt input file (or a larger input file, see below), run each algorithm 20 times *for each* of the following 5 values of the sample size parameter: 5,000, 10,000, 20,000, 30,000, 40,000. For each of the values, plot a curve for each of the minimum, maximum, median, and first and third quartile over the 20 runs (so you will create 10 figures in total, five for each algorithm, and of these five, there is one for each of the 5 different values of the sample size parameter, each containing 5 curves). Please remember to add a legend to your figures, and to label the axis and the figures appropriately.

You will write a report containing the figures and a comment on the behavior of the algorithms, analyzing how the min, max, median, and various quartiles change with the sample size value for each algorithm, and then discuss the difference between the two algorithms. We expect about $1/2$ a page to 1 page of comments. When preparing your submission, place your report in the same directory of your source code.

Larger input files A (very?) good implementation may be able to process the ca-AstroPh.txt in a few (tens of?) seconds. Indeed, as far as modern graphs go, this file is pretty small. **For extra credit** you can use a different larger input file, and report on it *instead* of using ca-AstroPh.txt. Please mention in your report which input graph you are using. We would greatly appreciate if you evaluate your implementation on the largest graph it can handle in a reasonable time. When using a larger graph, you may need to adapt the values of the sample size parameter: please choose your own values carefully so they are representative of a variety of behaviors and you can really evaluate your implementation. Choosing the right values is part of the art and practice of algorithm testing.

Here are some links to larger input files (each page reports the number of vertices and edges of the graph, so you can decide how large of a graph you may want to handle; the input file is linked at the bottom of each page):

- http://snap.stanford.edu/data/soc-Epinions1.html
- http://snap.stanford.edu/data/roadNet-CA.html

[1] You have to uncompress the file first.

- http://snap.stanford.edu/data/web-Google.html
- http://snap.stanford.edu/data/soc-LiveJournal1.html
- Any other large *undirected* graph at http://snap.stanford.edu/data/index.html

How to submit

Submit your work at https://www.cs.amherst.edu/submit or via cssubmit from romulus or remus, as a *single* archive file with name username.ext where username is your user name and ext is one of .zip, .tar.bz2, or .tar.gz (no .rar, please). The archive must contain a *single* directory with name username. This directory must contain a subdirectory with name X for each Exercise X, or a single subdirectory with name 1 in case of projects. All files (source code or otherwise) for each exercise (or project) must be in the directory for that exercise. Directories containing source code should contain a README.txt file explaining how to run the code in that directory. For non-code answers, please submit a .pdf (no .txt, .rtf, or .doc(x), please). You can find an example archive at http://bit.ly/DM19sub. Please post to the Moodle forum if you have problems with the submission.