Algorithms in the Real World (15-853), Spring 2018

Project Proposal — Due: Tuesday, April 3

Project — Due: Thursday, May 3

This assignment is a programming project. The goal is to implement an algorithm related to the algorithms we have discussed in class. You should work on your own or in groups of two. You need to submit working code that can be readily compiled (use a makefile or a build system) and a writeup describing algorithm details, any serious difficulties you came across, any optimizations you implemented, and some timings. The report should be 5-10 pages including any tables and figures. If you use any code or code snippets from elsewhere they need to be cited in the report. The project cannot be anything you are using or have used for another class. If it is related to your research it cannot be something you are doing anyway, but trying some variant or new algorithm for a research problem is fine, and even encouraged.

The project proposal is due **Tuesday April 3**, and the project is due **Thursday, May 3**. The project proposal should be a paragraph or two overview of what you plan to do.

Here are a list of possible projects, as ideas. You are not restricted to this list.

1. A parallel version of the PPM algorithm for text compression. The algorithm does not need to generate exactly the same compressed file as the sequential algorithm. For example you can batch the updates. You don't need to implement your own arithmetic coder.

2. Generate Huffman Trees in parallel.

3. A parallel version of Burrows Wheeler compression.

4. Implement an I/O efficient sort and run it on data that does not fit in memory. Use this to implement list-ranking or some other pointer-based algorithm.

5. Examine the I/O efficiency of an algorithm for something else we've talked about in class (empirically and/or theoretically), and try to come up with a more I/O efficient version.

6. Implement the connectivity algorithm described in lectures, and compare it with other approaches to parallel connectivity.

7. Implement a graph compression algorithm. Ligra contains several existing compression formats based on difference-encoding and run-length encoding; one could try taking advantage of SIMD. See this library.

8. Compare the performance and compression-quality of different graph reordering algorithms on real-world inputs, e.g. Distributed Balanced Partitioning via Linear Embedding or Compressing Graphs and Indexes with Recursive Graph Bisection.

9. Implement and evaluate an interesting parallel graph algorithm on large real-world graphs (should be more involved than just coding up a parallel graph-traversal algorithm).

10. Using the Fast Johnson-Lindenstrauss Transform to speed up dimension reduction. (There are newer papers on sparse J-L transforms, many by Jelani Nelson and his co-authors. One can compare these approaches.)