Algorithms in the Real World (15-853), Spring 2014
Project Proposal — Due: April 8
Project — Due: May 1

This assignment is a programming project. The goal is to implement some 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 running code and a writeup describing the algorithm you used, any serious difficulties you came across, any optimizations, 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 8**, and the project is due **May 1**. 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 some form of optimization algorithm.

5. 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.

6. 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.

7. Randomized approximate and/or parallelized SVDs: there's a bunch of options out there, try implementing a few and exploring the speed/memory/accuracy tradeoffs for different kinds of data.

8. Streaming SVD-like algorithms: see, e.g., this paper by Liberty, or these two papers (1, 2) by Clarkson and Woodruff.

9. Locality sensitive hashing for different classes of metrics.

10. An exploration of techniques to compute the number of distinct elements. (There are several approaches, see a paper of Bar-Yossef, Jayram, Kumar, Sivakumar, and Trevisan. And an optimal algorithm of Nelson, Kane, and Woodruff.)

11. 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.)

12. Learning Mixtures of Gaussians. A paper of Santosh Vempala and Grant Wang gives a way to learn mixtures of k Gaussians: use SVD to get the best k-dimensional subspace followed by a special purpose clustering algorithm in those k dimensions. One can instead use Lloyd's k-means heuristic when down to k dimensional space.

13. An experimental analysis of cuckoo hashing and its variants.