Algorithms in the Real World (15-853), Fall 10
Project Due: April 26

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.
I have a preference for you doing a parallel algorithm, but this is not required. I can get you access to a multicore machine.
The project proposal is due April 5th, and the project is due April 26. 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. Calculate the edit distance between two strings in parallel using $O(nm)$ work and $O(n+m)$ span and space.

4. Implement the Callahan-Kosaraju algorithm and use it to find all nearest neighbors. Ideally this should be parallel.

5. Implement cover trees and use it to find all nearest neighbors. Ideally this should be parallel.

6. Implement Collision detection using Bounded Volume Hierarchies (we did not do these in class, but they are similar to K-d trees). Ideally this should be parallel.

7. 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.

8. Implement an I/O efficient priority queue and use it to find shortest paths on a DAG. You can assume the DAG is already topologically sorted.