Massachusetts Institute of Technology
6.854J/18.415J: Advanced Algorithms
David Karger

Handout 21
November 15, 2011

Course Project

This handout provides a brief description of the kinds of things we are looking for in the course project. The basic goal of the project is for you to independently apply some of the advanced algorithmic thinking you have (hopefully) been developing in this class. The course project involves a combination of goals—encountering and independently absorbing new material, doing some thinking about it, and presenting what you have done. Different projects will attain these goals in different proportions.

There are 3 main ways to approach the project.

Reading project. Find a few interesting but challenging theoretical papers on a coherent topic, and write up an exposition that synthesizes their ideas. You will be graded on how much clearer/more informative your exposition is than that of the original papers (so the original papers should be pretty tough). The paper should be new—not yet digested into the journal and textbook literature. While you need not give all proofs in full detail, the reader should come away with a good understanding of why it all works. Unless the paper is extensive, one is usually not sufficient; two or three often are.

Theoretical research. Develop an interesting new solution to an algorithmic problem. "Interesting" may mean more efficient or may mean simpler. While you will presumably need to do some background reading, there is no set lower bound on it so long as you are able to advance beyond what is known. This kind of project will be graded on the extent of the improvement on previous results. As with all research, you need to be prepared for the risk of not making any progress—what is your backup plan? Often, it can be a reading project built around your background reading for the research.

Implementation project. We have studied many algorithms in theoretical form. Their behavior when implemented can be quite surprising. Grab one that interests you, implement and test it as above. This kind of project will be graded on how well you explain the algorithm you are implementing, what kind of interesting heuristics you developed, what interesting test cases you ran them against, and how well you interpret the results.

Algorithms papers tend to leave out a lot of little implementation details that turn out not to be so little when the time comes to implement. You may also explore implementation of heuristics that have "no theoretical value" but may lead to enormous improvements in practice. Once you have an implemented algorithm, you can test it to see how it performs in practice. This involves devising interesting "hard" inputs that make the algorithm perform poorly. Study of the algorithm's behavior may lead you to make changes in the algorithm and test them. A typical implementation paper says "we implemented algorithm X and it was awful, then we added heuristic Y and it was great!"

Be cautious—implementations can take a lot of time. Make sure what you are trying to implement is a reasonably "small" algorithm. Give serious thought to test inputs—ideally, they will come from real-world problems, or will be specially designed to "stress" some aspect

of the algorithm. You probably also need a control (i.e. dumb) algorithm to compare yours to.

Also, realize that your reader (me) may well not be familiar with the algorithms you are implementing; thus, your paper must give an adequate description of the algorithm, not just jump to the implementation results.

Warning: it is very easy to get a B on an implementation project, by just diving in and implementing. If you don't read an exemplary implementation paper (e.g. by goldberg) you won't understand what is needed to do one well. It isn't any harder than doing other kinds of papers, you just have to know what you are doing.

Regardless of which route you take, the end result should be a roughly 10 page paper describing the results. Presentation quality will be a factor in your grade. The paper should be written to be understood **by any other student in this advanced algorithms class**: you should not assume the reader has more knowledge than an undergraduate CS curriculum plus this class. Based on many papers that missed this mark, I **strongly recommend** that you trade readings with someone else in the class—you'll be surprised at how incomprehensible other students find your writeup, and will hopefully be able to make changes to fix that problem.

Some obvious questions:

What topic should I work on? The best topic to pick is one you are interested in anyway. Many of you are already involved in some research project; some thought may reveal an algorithmic component. Perhaps your system is presently using a naive algorithm for X and could be improved by using a more sophisticated one. You can do background reading on the topic (reading project), develop a theoretical model of the problem and devise a solution (theoretical project) or take some extant algorithm and try it out (experimental project).

If you aren't working on anything, try browsing through what we've covered in class as well as papers (see below) to identify an area that sounds like fun.

You are **not** limited to the topics we have covered in class. Anything that uses ideas from combinatorial algorithms (numeric algorithms are out of scope) and a theoretical framework to devise more efficient solutions is fair game.

Where can I find papers? There are numerous sources of papers on algorithms. The best work in the area is published in three conferences, each with yearly proceedings:

- The ACM Symposium on Theory of Computing (STOC).
- The IEEE Symposium on Foundations of Computer Science (FOCS).
- The ACM-SIAM Symposium on Discrete Algorithms (SODA).

There are lots of other sources, of course. There are also journals but they tend to be rather behind.

Is collaboration allowed? Yes, even encouraged—group projects are much more fun, and you will be able to get more done (more will be expected of groups). As with problem sets, group size should be limited to three so all can participate fully. All members should participate in the entire project. One collaboration strategy that has frequently *failed* in the past is to have each collaborator read and write up a single paper and staple the writeups together. The writeup needs to be coherent, drawing together the different elements of the project.