6.854 — Advanced Algorithms

David Karger and Aleksander Mądry

Handout #1, September 5, 2018 — Class Info

Summary: The need for efficient algorithms arises in nearly every area of computer science. But the type of problem to be solved, the notion of what algorithms are "efficient," and even the model of computation can vary widely from area to area.

This course is designed to be a capstone course in algorithms that surveys some of the most powerful algorithmic techniques and key computational models. We will cover a broad selection of topics including amortization, hashing, dimensionality reduction, bit scaling, network flow, linear programing, and approximation algorithms. Domains that we will explore include data structures; algorithmic graph theory; streaming algorithms; online algorithms; parallel algorithms; computational geometry; external memory/cache oblivious algorithms; and continuous optimization.

Goals: We hope that this class will confer

- some familiarity with several of the main lines of work in algorithms—sufficient to give you some context for formulating and seeking known solutions to an algorithmic problem;
- sufficient background and facility to let you read current research publications in algorithms;
- a set of tools for design and analysis of new algorithms for new problems that you encounter.

Class webpage: `http://courses.csail.mit.edu/6.854/` . **Sign up there for the mailing list.**

Instructors:
David Karger, 32-G592. `karger@mit.edu`. `http://people.csail.mit.edu/karger/`
Aleksander Mądry, 32-G666. `madry@mit.edu`. `http://people.csail.mit.edu/madry/`

Content: The goal is to be broad rather than deep. This list is approximate.

Data Structures: Persistent data structures, splay trees.

Network Flows: Augmenting paths. Min-cost flows. Bipartite matchings.

Linear Programming: Formulation of problems as linear programs. Duality. Simplex, Interior point, and Ellipsoid algorithms.

Continuous Optimization: Gradient descent. Newton's method.

Dimensionality Reduction: Johnson-Lindenstrauss lemma. Compressive sensing.

Online Algorithms: Ski rental. Paging. The k-server problem.

Approximation Algorithms: Greedy approximation algorithms. Dynamic programming and weakly polynomial-time algorithms. Linear programming relaxations. Randomized rounding. Scheduling, vertex cover, and TSP.

Parallel Algorithms: PRAMs. Circuits. Maximal independent set.

External-Memory Algorithms: The cost of accessing data from slow memory. Buffer trees. Cache-oblivious algorithms.

Computational Geometry: Convex hull. Orthogonal range search. Voronoi diagrams.

Streaming Algorithms: Sketching. Distinct and frequent elements.

Prerequisites: Strong performance in an undergraduate class in algorithms (e.g., 6.046/18.410), discrete mathematics and probability (6.042 is more than sufficient), and substantial mathematical maturity.

Requirements:

Scribing and/or grading (10%). Scribe a lecture in LaTeX and/or help grade a problem set.

Homework (70%). Weekly problem sets, with collaboration usually allowed.

Many of the problems already have solutions posted on the internet or in course bibles. **No preexisting solutions may be used.** Violations of this policy will be dealt with severely.

Independent Project (20%). You will carry out independent work to exercise your new mastery of algorithms. It can have several forms, or be a combination:

Read some new (not yet textbook) algorithms from the recent research literature, and provide an improved (of greater clarity) presentation/synthesis of the results

Research a new algorithm that improves upon the state of the art, either for a classical problem or one that arises naturally from your own work

Implement some interesting algorithms and study/compare their performance. Considerations include choice of algorithm, design of good tests, interpretation of results, and design and analysis of heuristics for improving performance in practice.

Collaboration (in groups of at most 3) is encouraged on these final projects.

Homework/Collaboration Policy.

1. Homework is due Wednesdays at the beginning of class. We'll have boxes or piles for dropping them off at the back of the room. Those boxes will be collected a few minutes in, and homework arriving after will be considered late.

2. Each student has a total budget of 15 "slack" points to accommodate his/her late problem submissions.

   - Each problem that is submitted late but in before Friday class will consume one slack point (and incur no grade penalty).
   - If that problem is submitted in before Monday class, it will consume two slack points (and, again, incur no grade penalty).
   - No late problem will be considered if submitted after the Monday class begins.

   So, for example, if there is a problem set with a total of five problems on it, submitting three of these problems on time, one of them before Friday class, and one of them before Monday class will consume three slack points in total.

3. Write each **sub**problem on a separate sheet of paper and include your name and email address. Also, make sure your name appears on each page.

4. Collaboration is encouraged, except where explicitly forbidden.

   - All collaboration (who and what) must be clearly indicated in writing on anything turned in.
   - Collaborators should discuss solutions, but must write up all solutions independently.
   - Groups must be small so that each member plays a significant role (usually 3 or 4 students).
   - For projects every collaborator must contribute significantly to reading, implementation, and writeup. To allow this, groups should limit their size to 3 unless the project is unusually large. All members should be involved with all parts of the project and writeup.

5. You may not seek out answers from other sources without prior permission. In particular, you may not use bibles or posted solutions to problems from previous years.

6. Each student is required to grade (at least) one problem in the semester. We will have a TA-supervised grading session each week. This session is used to make sure that the graders fully understand the solution, while they can grade the problems at home after this session.

7. For questions about grading, please contact the graders (emails listed on the website) first. Once you reach an agreement, the grader should send an email to the grading supervisor with a short explanation and a new grade.

8. All late psets should be sent electronically to the TA supervising the grading.

Textbooks. There are no textbooks covering a majority of the material we will be studying. Lectures will often draw from the following (optional) texts, all of which are nice to have.

1. Cormen, Leiserson, Rivest, and Stein. *Introduction to Algorithms.* MIT Press. 2001.

2. Ahuja, Magnanti, and Orlin. *Network Flows.* Prentice Hall, 1993.

3. Motwani and Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

4. Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, 1997.

5. Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.

6. Robert Tarjan. *Data Structures and Network Algorithms.* Society for Industrial and Applied Mathematics, 1983. A classic—no longer up to date, but outstanding writing.

7. Mark de Berg, Marc van Kreveld, Mark Overmars, Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications.* Springer Verlag, 2000.

8. Williamson and Shmoys. *The Design of Approximation Algorithms.* Cambridge University Press, 2011.