Syllabus for DS 592
Randomized Algorithms (with applications)

Moon Duchin @ Boston University

Spring 2024

What is this course?

Randomization can supercharge algorithms! They often get simpler and faster when their step-to-step behavior is allowed to include some randomness. But getting a handle on the associated guarantees (in terms of correctness and runtime) is highly nontrivial.

This course is proofy but also application-centered. You definitely have to write solid proofs. I don't assume you already know how, but I—and our TF, Kevin Quinn—will help you get there. You definitely have to write pseudocode. Actual code that runs and produces output will typically be optional.

But isn't there a course with the same title offered by CS?

There sure is! And to the extent that we're using a textbook, it's even the same one: Motwani and Raghavan, *Randomized Algorithms* — an excellent book that someone has been kind enough to put online in PDF form. But there are differences: (a) that is a big class and this is a small one, (b) this one is meant to be a topics class that jumps around and spends time on brand new ideas (whereas the textbook is from 1995), (c) this is CDS, and the D is for "data," which I will endeavor to take seriously as we explore some of these algorithms.

Who are you?

My name is Moon Duchin — I'm a pure mathematician by training with expertise in geometric group theory, geometric topology, and dynamical systems. My research these days is in mathematical data science, especially graph algorithms for analyzing political redistricting and broad topics in computational social choice. (You'll hear a lot about that as we go.) I've been teaching at Tufts for about 12 years and I'm visiting BU this year, in both Math/Stat and CDS. I split my time between the 4th floor and the 13th floor and I'll share my office hours timing and location as soon as both are set. Please be patient with my limited knowledge of BU.

Our TF is Kevin Quinn, who's a second-year PhD student in CDS.

What do you expect me to know in advance?

I assume most or all of you have had a probability course, a programming course (with basic exposure to Python), a discrete math/graph theory course, and some exposure to proof. If you're missing some of these ingredients, it's still possible to take the class, because I'll be calibrating to your level to the best of my ability.

With coding in particular, I don't expect that this is your strong suit, but there may be light Python assignments that assume at least "hello world" familiarity and a willingness to learn. I hope you've seen LaTeX and overleaf and I will expect you to TeX your written work.

What is the weekly rhythm for this course?

I've got a list of topics I want to cover, and rough timing in mind for each. But I want to adapt to the best pace for the actual students enrolled in the class, so the plan is subject to adjustment.

Class meets Tue/Thu 11-12:15, and there's a required discussion session on Wednesdays 12:20-1:10. Typically I teach class and Kevin runs discussion, but there are a few weeks that I'll be out of town on a Tuesday or Thursday and we'll swap it up. Basically, plan to be at all three meetings each week! This is an expectation of this class, and can impact your grade if you miss more than four meetings.

There is weekly homework — typically about five problems a week, sometimes with programming options attached. Homework will be given out Tuesdays, due Mondays by 11:59pm sharp, and graded by the following Monday (except in unusual weeks). There is no final exam, but there is a written final assignment that you'll work on in pairs (assuming the enrollment stays even) that's due in our final exam block but you're welcome to finish sooner. So no need to be here in person in our final exam block.

You can miss two assignments over the course of the term with no penalty, and you can be up to two days late on one more. After that it comes out of the grade.

In the past, I have found simple in-class smartphone quizzes to be a remarkably effective learning tool — no joke! I reserve the right to add these once I take the temperature of this class. Other than that, please stay off your phones and laptops in class.

How does grading work?

This is a letter-graded course. I like to grade problems on a simple four-point scale: 4 if you nailed it, 3 if it's mostly right, 2 means good start, 1 means you started! If a problem is substantial it might be worth double, or 8 points. This scale is meant so you have a sense of how you're doing, grade-wise: a 75% score will typically be keyed to an A-. Because this is a grad/undergrad topics course, I'm not planning to maintain an online gradebook like you see in lower-level classes. But talk to me at any point if you want to know how you're doing.

How will we communicate?

CDS has a slack workspace and I've created a channel for our class. However, it's currently an upaid account, so messages only last for 30 days. (Hopefully it will be converted to a paid workspace before long—if not, I'll plan to set up a Wordpress site to supplement the slack in with materials I want available more durably.)

Slack is a fantastic place to ask questions, so they can be answered and used collectively — I will definitely chip in where needed. It's also by far the best way to get in touch with me, because my email inbox is a radioactive thicket.

For submitting work, Kevin will be maintaining the submission mechanism; he's currently deciding between Blackboard and Gradescope.

Topics to be covered

There are only 26 class meetings for the semester, plus 11 discussions, if I've done the arithmetic right. There are five major topics I will cover at a pace of 3-6 classes each, which will account for most of the semester. I will mix in many other topics as one-offs along the way. I plan to make a day-by-day rough schedule available by Week 3 once I have a feeling for our pace.

Major topics.

- Efficent partitions (including mincut)
- Random walks
- Other Markov chains, and MCMC
- Spanning trees and matroids
- Linear programming

Other important topics. Sorting, coupon collecting, matching, probabilistic method, shortest paths, computational geometry

Integrity

I expect you to work together if you want (and most of you will). Your written work must credit your collaborators. You are responsible for familiarity with the Academic Code of Conduct.

It's the age of gray areas for academic integrity, now that powerful AI can write code and even write proofs. Heads up that both can be buggy. But your professors do know how gray the gray areas are. The CS department has put together a really helpful guide on academic integrity.

CDS also has an official policy on Generative AI.
TL;DR — just like with human collaborators, you must give credit to AI tools, however you use them! It's not an automatic deal-breaker to use chatbots but I will let you know if it's getting there. Please be open and honest about this! It helps us all.

Respect and atmosphere

I will try to set the tone here: a tone of support, respect, and care. This is expected of you as well, from calling each other what we want to be called to being generous with invitations to collaborate.

As part of an atmosphere of respect, I have a practice of adopting an extremely relaxed accommodations policy. You should be aware of your access to Disability & Access Services (bu.edu/disability) but you should also just talk to me about ways that I can help this class work for your working style and your learning style. Being open-minded about this is important to me.

Please do not record class or share materials like solution sets beyond the other members of this class community.