CSE 599-I: Algorithms through Geometric Lens (Fall'18)

Course Information

Instructor: *Ilya Razenshteyn*

1 Basic Information

Lectures:

- Time: Tue, Thu, at 4:30–5:50pm.
- Location: CSE 305.

Instructor:

- **Ilya Razenshteyn** (`ilyaraz@microsoft.com`)

Office hours are by appointment.

Website: There is no textbook, but there's a website with regular updates and links to lectures, and additional resources:

`https://ilyaraz.org/static/class_2018/`

(see also `https://ilyaraz.org/static/class/` for the previous offering of the class).

Courseworks: Class announcements, including homework assignments will be done via a mailing list.

2 Course Goals

The goals of the class are to introduce you to modern ways to model various algorithmic questions using geometry, and use geometric tools to solve them efficiently. Many of these geometric approaches have become central to modern algorithm design over the last two decades. The intent is to be broad, covering a diversity of algorithmic problems, and geometric techniques, rather than be deep. Many of the covered algorithms have been implemented and are used in industry. The ultimate goal of the class is to equip you with skills to:

- model many algorithmic questions using a geometric language;
- apply modern geometric tools to address such questions;
- be able to read research-level papers in the field of algorithms.

Tentative topics to be covered:

- Sketching/Streaming
- Dimension reduction

- Randomized numerical linear algebra
- Nearest Neighbor Search (NNS)
- Graph algorithms based on semidefinite programming
- Spectral Graph Algorithms
- Metric embeddings with applications
- Distance oracles
- Discrepancy minimization

3 Prerequisites

First and foremost, mathematical maturity is a must: the class is based on theoretical ideas and is proof-heavy. You are expected to be able to read and write formal mathematical proofs. Furthermore, some familiarity with algorithms and randomness will be assumed as well.

Here is a rough list of math/CS topics that you are expected to know or have background in:

- basics of probability theory, including: linearity of expectation, variance, Markov/Chebyshev bound;
- basic linear algebra (eigenvalues, eigenvectors);
- asymptotic analysis of algorithms, runtime analysis;
- most basic algorithms, such as hashing or binary search;
- graphs.

4 Evaluation and Grading

Your grade is based on the following three components:

- 2–3 homeworks: 50%;
- Project: 50%, including 5% for project proposal, 10% for oral presentation, and 35% for the final write-up.

5 Homeworks

Homeworks will be assigned roughly every two weeks and will be posted on Courseworks. They will be due in class on their due date before the lecture starts. Please follow the Homework Submission Guidelines below.

Late policy. You have a default 5 days of extension (fractions of a day are rounded up), over all the homeworks. Once you've used up the 5 days, late homeworks will be penalized at the rate of 10%, additively, per late day or part thereof (i.e. fractions of a day are rounded up), for up to 7 days. To

allow us to distribute the solutions in a timely fashion, homeworks submitted more than 7 days after the deadline will not be accepted. Exceptions will be made only for exceptional unforeseen circumstances (e.g., serious illness), in which case you will need to provide some additional documentation (e.g., doctor's note).

You are strongly encouraged to start working on the homeworks *early*: some problems may require you to sit on the problem for a while before you get your "aha" moment. Starting early also gives you time to ask questions and make effective use of the office hours of the teaching staff.

Writing up solutions: precise and formal proofs. The goal of the class, in part, is for you to learn to reason about algorithms, precisely describe them, and formally prove claims about their correctness and performance. Hence, it is important that you write up your assignments *clearly, precisely, and concisely.* Legibility of your write-up will be an important factor in its grading. When writing up (algorithmic) solutions, keep in mind the following:

- The best way for you to convey an algorithm is by using plain English description. A worked example can also help; but revert to pseudocode only if necessary. Generally, give enough details to clearly present your solution, but not so many that the main ideas are obscured.
- The analysis of the algorithm has to include both 1) proof of correctness, and 2) upper bound on performance (usually runtime, but sometimes space as well).
- You are encouraged (but not required) to type up your solutions using LaTeX. Latex is the standard package for typesetting and formatting mathematically-rich content. Since LaTeX knowledge is a good life skill, now may be a good chance to learn it. A short mini-course on LaTeX is available here: `http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf`. Macros to format pseudocode are available at `http://www.cs.dartmouth.edu/~thc/clrscode/`

Note that our lectures will generally be at a slightly lower level of formalism, in the interest of time.

Clarity points. To encourage clarity (and conciseness), for each problem, 20% of the points are given for the *clarity* of your presentation. In particular, you will be awarded a default 20% of the points for an *empty solution* (note that, if you submit no coversheet whatsoever, you get only 0%). Note that you can *lose these 20%* if you write something that is unintelligible, does not lead to a solution, or is excessively long (including scoring a 0%).

6 Collaboration and Academic Honesty

Collaboration: you are permitted to discuss the homework *assignments*. If you collaborate, you must write the solutions *individually* (without looking at anybody else's solutions), and acknowledge anyone with whom you have discussed the problems. It will be considered an honor code violation to consult solutions from previous years, from the web or elsewhere, in the event that homework problems have been previously assigned or solutions are available elsewhere.

You are expected to abide by the policies of academic honesty.

7 Homework Submission Guidelines

- Submit your homeworks electronically via e-mail *in pdf format*. You may write solution by hand, in which case you should either scan or photograph your solutions.

- Homeworks are due on the specified due date 10minutes before the class starts (i.e., at 4:20pm).
- Please identify yourself and each problem clearly at the top of each page. Write your name and e-mail, and the problem number. Collaborators must be mentioned for each problem.

8 Final Project

In the final project you will delve into a particular topic in more detail in a team of your own. The final projects can be of three types:

- Reading-based: read a few recent research papers on a concrete topic and summarize them.
- Implementation-based: implement some of the algorithms from the class (or from other theoretical literature), and perhaps apply to your area of interest/expertise, using real-world datasets. One aspect of such projects will be a comparison among a few algorithms.
- Research-based: investigate a research topic on your own (eg, develop an algorithm, and prove its properties; or prove an impossibility result). It may be more applied: e.g., perhaps in your area, certain theoretical algorithms can be modified to have even better performance, due to special properties of the datasets, etc.

You will have to submit a project proposal, of about 2 pages in length. Also, you will make a 10-15mins presentation at the end of the class.

Teams: you are allowed to have a team of up to 2 people in total per team. Single-person teams are not encouraged and need special permission from the instructor (the reason is that the topics are hard, and having a collaborating partner/s will qualitatively improve your experience).

You are encouraged to find a team early, and discuss potential topics with the instructor.

Topic: the topic of your project must be within the scope of Theoretical Computer Science, and preferrably algorithmic. In particular, the focus is on algorithms with provable guarantees (for the implementation type, you may compare such theoretical guarantees with heuristics though). More details and suggestions will be given later in the class.