COMS 4232: Advanced Algorithms (Spring'23) Jan 18, 2023

Course Information

Instructor: *Alex Andoni*

1 Basic Information

Lectures:

- Time: MW 2:40-3:55pm.
- Location: Mudd 1127.

Instructor:

- **Alex Andoni** (`andoni@cs.columbia.edu`)

Teaching Assistants:

- Imanol Uribe Echevarria, `iu2155@columbia.edu`;
- Yiming Fang, `yf2484@columbia.edu`;
- Hantao Yu, `hy2751@columbia.edu`.

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

`http://www.cs.columbia.edu/~andoni/advancedS23`

Office hours: see calendar linked from Courseworks (requires LionMail login).

Self evaluation test: to confirm for yourself that you have the right background, you should complete the self-evaluation test asap (definitely, by the end of the first week). It will help you identify potential parts to brush up before the class starts. See the website for the test and the solution key.

Courseworks: Class announcements, including homework assignments will be posted on Courseworks.

Edstem: There is also a Edstem forum setup for the class (accessible through the Courseworks). You are encouraged to discuss class lectures and related topics, as well as ask questions or clarifications. (But you *cannot* discuss homework solutions on Edstem.)

2 Course Goals

The goals of the class are to introduce you to classic and modern algorithmic ideas that are central to many areas of Computer Science. The focus is on most powerful paradigms and techniques of how to design algorithms, and how to measure their efficiency. The intent is to be broad, covering a diversity of algorithmic techniques, rather than be deep. The covered topics have all been implemented and are widely used in industry. At the end of the class, expect to:

- know a diversity of classic algorithmic tools,
- be able to design algorithms for computational problems arising in your area,
- be able to read research-level papers in the field of algorithms.

Topics to be covered (one bullet point corresponds to approx 0.5—2 lectures):

- **Hashing**:
  - universal hashing, perfect hashing
  - power of 2 choices, concentration bounds (Chernoff/Hoeffding).
- **Sketching/Streaming**:
  - approximate counting, Morris algorithm
  - distinct elements count, impossibility results
  - heavy hitters, countSketch algorithm
  - frequency moments, dimension reduction.
- **Nearest Neighbor Search**:
  - sketching, nearest neighbor search
  - Locality Sensitive Hashing scheme.
- **Graph algorithms**:
  - flows decomposition and augmenting paths
  - capacity scaling algorithms
  - max-flow algorithms.
- **Spectral Graph Theory**:
  - linear algebra overview, and graph Laplacian
  - spectrum of graph Laplacians
  - graph cuts, Cheeger inequality
  - expander graphs, applications.
- **Linear Programming**:
  - introduction to Linear Programming (LP), structure of optima
  - strong duality, complementary slackness
  - simplex and ellipsoid algorithms
  - gradient descent, iterative methods
  - interior point method
  - multiplicative weights update, online algorithms.

- **Models for large-scale computation**:
  - external memory, cache-oblivious algorithms
  - parallel algorithms.
- **Other tentative topics**:
  - Fast Fourier Transform;
  - Algorithms to beat exhaustive search algorithms (e.g., 3SAT);
  - Error correcting codes;
  - Compressed Sensing.

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. COMS 4231 (Analysis of Algorithms) or equivalent is recommended, but not required if you have solid math background.

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

- basic linear algebra (eigenvalues, eigenvectors);
- basics of probability theory (linearity of expectation, variance, Markov bound);
- asymptotic analysis of algorithms, runtime analysis;
- hashing, or binary search trees;
- graphs.

4 Evaluation and Grading

Your grade is based on the following three components (TENTATIVE 1/12):

- 5 homeworks: 50%;
- Midterm: 25%;
- Project (including a preliminary report): 25% (including 5% for a preliminary report, and 20% 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 (e.g., using overleaf). 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. The CS department web page lists the department's academic honesty policies: `http://www.cs.columbia.edu/education/honesty`.

7 Homework Submission Guidelines

- Submit your homeworks electronically via GradeScope. 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.
- Please identify yourself and each problem clearly at the top of each page. Write your name and UNI, 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.

Teams: you are allowed to have a team of up to 3–4 people in total per team. Single-person teams are possible but discouraged 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 with the instructor the potential topics. There will be a project proposal due. It will be of 1–4 pages long (exact details tbd).

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.

9 Callendar (tentative)

Below is the schedule of assignments. Information regarding what each lecture covers will be periodically updated on the course website.

Lecture	Date		HW out	HW/P due
1	1/18			
2	1/23		HW1 out	
3	1/25			
4	1/30			
5	2/1			
6	2/6		HW2 out	HW1 due
7	2/8			
8	2/13			
9	2/15			
10	2/20		HW3 out	HW2 due
11	2/22			
12	2/27			
13	3/1			HW3 due
14	3/6			
	3/8	Midterm	HW4 out	
	3/13	NO CLASSES: Spring Break		
	3/15	NO CLASSES: Spring Break		
15	3/20			
16	3/22			
17	3/27			
18	3/29			HW4 due
19	4/3			
20	4/5			
21	4/10			
22	4/12		HW5 out	Project proposal due
23	4/17			
24	4/19			
25	4/24			
26	4/26			
27	5/1			HW5 due
	5/10	Final projects due		Project due