COMS W6998-9: Algorithms for Massive Data (Spring'19)

Course Information

Instructor: *Alex Andoni*

1 Basic Information

Lectures:

- Time: Tue, Thu, at 2:40–3:55pm.
- Location: Mudd 825.

Instructors:

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

Teaching Assistants:

- **Marshall Ball** (`mmb2249@columbia.edu`).
- **Peilin Zhong** (`pz2225@columbia.edu`).

Office hours will be announced on the Courseworks.

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

`http://www.mit.edu/~andoni/algoS19/algoS19.html`

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

Piazza: There is also a Piazza 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 Piazza.)

2 Course Description

Modern Data presents both a big promise but also a big challenge — how are we to extract that promise? The classic algorithms for processing data are often insufficient to deal with the datasets of modern sizes. For example, a quadratic-time algorithm means that 10x increase in data size requires a 100x increase in resources!

This class will focus on algorithmic techniques to tackle massive datasets efficiently. We will cover 1) some novel computational models for how to think of massive data processing, and 2) core algorithmic techniques. For example, a common thread will be how to summarize complex data into a smaller core, via sampling and small sketches techniques, so that later processing is more efficient. The class is theoretical in nature, and hence we will seek to prove correctness and efficiency guarantees of our algorithms.

The ultimate goal of the class is to equip you with skills to:

- model algorithmic questions on massive data;
- apply modern techniques to address such questions, and analyze resulting algorithms;
- be able to read research-level papers in theorical computer science, in particular in sublinear algorithms.

Tentative topics to be covered (about 1-2 lectures per line):

- **Sketching/Streaming**:
  - approximate counting, distinct elements count;
  - impossibility results;
  - frequency moments, "Tug of War" sketch;
  - heavy hitters, CountMin and CountSketch algorithms;
- **Dimension reduction:**
  - random projections (Johnson–Lindenstrauss lemma);
  - fast dimension reduction;
- **Numerical linear algebra (via sketching):**
  - regression via Sketch-And-Solve, subspace embeddings;
  - low-rank approximation;
- **Compressed sensing:**
  - Restricted Isometry Property (RIP), ℓ_1 minimization;
  - iteration hard thresholding;
- **Streaming for graphs:**
  - spanners, triangle counting;
  - dynamic graph algorithms via ℓ_0 sampling;
- **Distribution (statistical hypothesis) testing:**
  - uniformity testing, closeness;
- **Property testing / sublinear-time algorithms:**
  - monotonicity testing;
  - graph property estimation;
  - geometric problems;
- **Parallel computation:**
  - MapReduce-like parallel models, sorting;
  - graph algorithms.

3 Prerequisites

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 highly recommended, but not required if you have a solid math background.

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;
- basic algorithms, such as hashing, binary search, connectivity in graph;
- graphs.

4 Evaluation and Grading

Your grade is based on the following three components:

- Scribing (1 lecture): 10%;
- 3 homeworks: 45%;
- Project: 45%, including 5% for project proposal, 10% for oral presentation, and 30% for the final write-up.

5 Scribing

Each one of you will have to scribe one lecture.

Scribes are due by midnight next day after lecture. You have to use the LaTeX template available on the class website. The scribed lecture will be posted immediately after it is received so that the rest of the class can use it before the following lecture. The staff will review the scribe, and, if necessary, the scriber(s) will be asked to improve the write-up.

6 Homeworks

Homeworks will be posted on Courseworks. 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/`
  You are encouraged to use the template available on Coursework.

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%).

7 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`.

8 Homework Submission Guidelines

- Submit your homeworks electronically via Courseworks *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 10 minutes before the class starts (i.e., at 2:30pm).

- 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.

9 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:

- Survey-based: read a few recent research papers on a concrete topic and summarize the research on that topic.
- 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 2-3 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.

10 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	1/22			
2	1/24			
3	1/29		HW1 out	
4	1/31			
5	2/5			
6	2/7			
7	2/12			
8	2/14		HW2 out	HW1 due
9	2/19			
10	2/21			
11	2/26			
12	2/28			
13	3/5			HW2 due
14	3/7			
15	3/12			
16	3/14			
	3/19	NO CLASSES: Spring Break		
	3/21	NO CLASSES: Spring Break		
17	3/26		HW3 out	Proj proposal due
18	3/28			
19	4/2			
20	4/4			
21	4/9			
22	4/11			HW3 due
23	4/16			
24	4/18			
25	4/23			
26	4/25	Project presentations (if necessary)		
27	4/30	Project presentations		
28	5/2	Project presentations		
	5/10	Final project due		Final project due