Discrete Mathematics and Algorithms

CME 305 / MS&E 316

Professor: Aaron Sidford (sidford@stanford.edu / Huang 358)
TAs: Ananthakrishnan Ganesan (ananthg@stanford.edu), Arun Jambulapati (jmblpati@stanford.edu), and Nolan Skochdopole (naskoch@stanford.edu)
Lectures: Tuesday / Thursday - 10:30AM – 11:50AM in School of Education 128
Website: Canvas (go here for further course information, e.g. office hours, psets, announcements, etc.)

Course Overview

This class will introduce the theoretical foundations of discrete mathematics and algorithms. Emphasis will be on providing mathematical tools for combinatorial optimization, i.e. how to efficiently optimize over large structured finite sets. The course will provide a rigorous understanding of both discrete mathematical structures, e.g. graphs, matroids, submodular functions, set systems, etc., as well algorithmic paradigms for optimizing over them, e.g. greedy methods, linear programming, randomized methods, etc. The class constitutes a fast-paced proof-based introduction to the broad area of designing and analyzing algorithms for combinatorial problems. By course completion you should have a solid foundation to enable future theoretical research and practical work on discrete problems.

The material of the course will be split up roughly into 4 units that we describe below. The progression loosely follows the history of the field and is designed to give you a broad set of tools.

- **Unit 1 – Combinatorial Methods**: here we will cover some of the most fundamental and classic problems in combinatorial optimization and the combinatorial algorithms that solve them. Example topics include graphs, distances, connectivity, minimum spanning trees, minimum cut, maximum flow, bipartite matching, greedy algorithms, blocking flows, DFS, BFS, etc.

- **Unit 2 – Algebraic / Randomized Methods**: here we cover the use of algebraic techniques, random methods, and random walk based approaches for solving combinatorial problems. Example topics include non-bipartite matching, random walks, Markov chains, random spanning trees, PageRank, Laplacians, spectral methods, graph clustering, cover times, etc.

- **Unit 3 – NP-Hardness / Approximation Algorithms**: here we cover the fundamental complexity theory of NP-hardness for showing when combinatorial optimization problems may be computationally intractable. Furthermore, we will show how to devise efficient algorithms to approximately solve for many of these problems.

- **Unit 4 – Polyhedral / Primal-Dual Techniques**: here we cover algorithms based on linear programming and duality for designing fast combinatorial algorithms and approximation algorithms and understanding the structure of combinatorial problems.

In addition, there may be some advanced bonus material covered as there is time. Example topics include sparsification and approximate almost linear time algorithms.

Prerequisites

This class will assume strong foundations in mathematics, i.e. linear algebra, probability, set theory, etc.; having had an undergraduate course in algorithms is not required. However, we will introduce several algorithmic concepts early in the course and use them extensively. If you are unfamiliar with these, we encourage you to go to office hours and consult the suggested reading material. The sooner you internalize these, the easier the course will be. If you have questions about prerequisites or suspect that lectures are assuming too much prior knowledge, please do not hesitate to contact the professor or TAs.

Course Material

There are a few sources of course material that you are responsible for and will be tested on in the homeworks, midterm, and final. Here we cover each in slightly greater detail.

- **Lecture Notes**: We will provide lecture notes associated with each lecture. These will serve as the primary source of course material and are required reading. If you have any questions about the notes, find any typos, or any further information would be helpful please do not hesitate to post a discussion on Canvas (ideally) or email the course staff (if more appropriate).

- **Lectures**: The material from class is also required. If you miss class, it is your responsible to discuss with the other students and TAs to make sure you did not miss anything. That said, the lecture notes are intended to converge to a superset of the lecture material, so if you find lecture material missing from the lecture notes again please feel free to let us know.

- **Required Textbook**: The textbook for the course is "Algorithm Design" by Kleinberg and Tardos. I will assign readings from this book to complement the lecture notes. The readings will be particularly critical to solidify introductory algorithmic concepts in the beginning of the course.

- **Additional Reading**: Throughout the course I will provide pointers to additional reading that might be helpful. I will try to make clear what is optional, suggested, and required. If you have any questions about this please contact the course staff.

Note: the material in the course may differ slightly from previous years. Lecture notes from previous years may help to get ahead and obtain a more complete understanding of the course material, but they do not serve as a replacement for the lecture notes we will be providing.

Grading

Grades for the course will be based primarily on problem sets (30%), a midterm (30%), and a final (40%). There will be 4 equally weighted problems sets assigned and the midterm will occur in-class. In addition, minor consideration in grades may be given to class participation. Class participation can occur both by participating in in-class discussion, participating in discussions on the Canvas forum, and asking good questions about the lecture notes (e.g. pointing out typos or missing material).

More Course Expectations

I will pass out comment cards at the beginning of each lecture to get quick feedback on the course pace and ensure material is not lost. Please fill them out; it takes less than a minute and they are invaluable.

Tentative Schedule

(See Canvas for the latest version)

- **Week 1: T 1/9**: class overview and global minimum cut
- **Week 1: Th 1/11**: intro to graph theory (connectivity, distances, BFS, DFS)

- **Week 2: T 1/16**: spanning trees and matroids
- **Week 2: Th 1/18**: maximum flow

- **Week 3: T 1/23**: bipartite matching
- **Week 3: Th 1/25**: algebraic methods for general matching **[pset #1 due]**
  (Friday 1/26: add/ drop - deadline)

- **Week 4: T 1/30**: algebraic / randomized methods
- **Week 4: Th 2/1**: algebraic / randomized methods

- **Week 5: T 2/6**: algebraic / randomized methods
- **Week 5: Th 2/8**: algebraic / randomized methods **[pset #2 due]**

- **Week 6: T 2/13**: graph approximation – spanners and sparsifiers
- **Week 6: Th 2/15**: **[in class midterm]**

- **Week 7: T 2/20**: NP-hardness and approximation algorithms
- **Week 7: Th 2/22**: NP-hardness and approximation algorithms

- **Week 8: T 2/27**: NP-hardness and approximation algorithms
- **Week 8: Th 3/1**: polyhedral / primal-dual techniques **[pset #3 due]**

- **Week 9: T 3/6**: polyhedral / primal-dual techniques
- **Week 9: Th 3/8**: polyhedral / primal-dual techniques

- **Week 10: T 3/13**: polyhedral / primal-dual techniques
- **Week 10: Th 3/15**: bonus material - recent fast algorithms **[pset #4 due]**

The final exam is on Wednesday, March 21