CS 473: Algorithms

Ruta Mehta

University of Illinois, Urbana-Champaign

Spring 2018

Administrivia, Introduction

Lecture 1

Jan 16, 2018

Some of the slides are courtesy Prof. Chekuri

Part I

Administrivia

Instructional Staff

1. Instructors: Ruta Mehta
2. Teaching Assistants: Shant Boodaghian and Vasilis Livanos
3. Graders: TBD
4. Office hours: See course webpage
5. Email: Use private notes on Piazza to reach course staff.

Online resources

1. Webpage: General information, lecture schedule/slides/notes, homeworks, course policies
   courses.engr.illinois.edu/cs473
2. Gradescope: HW submission, grading, regrade requests
3. Moodle: HW solutions, grades
4. Piazza: Announcements, online questions and discussion, contacting course staff (via private notes)

See course webpage for links

Important: check Piazza/course web page at least once each day

Prerequisites

1. Prerequisites: CS 173 (discrete math), CS 225 (data structures), CS 374 (algorithms and models of computation) or sufficient mathematical maturity
2. Concretely:
    1. Good ability to write formal proofs of correctness
    2. Comfort with recursive thinking/algorithms, reductions
    3. Comfort with basic data structures: balanced binary search trees, priority queues, heaps, etc.
    4. Basic graph algorithms: reachability (DFS/BFS), undirected vs directed, strong connected components, shortest paths and Dijkstra's algorithm, minimum spanning trees
    5. Probability: random variables, expectation, variance
    6. Exposure to models of computation and NP-Completeness (optional but will help)

Textbooks and Resources

No one specific textbook for the course.

1. Recommended books: (not required)
    1. Algorithms by Dasgupta, Papadimitriou & Vazirani. Available online for free!
    2. Algorithm Design by Kleinberg & Tardos
2. Lecture notes/slides/pointers: available on course web-page
3. Additional References
    1. Lecture notes of Jeff Erickson, Timothy Chan, Sariel HarPeled, and others
    2. Computers and Intractability: Garey and Johnson.

Grading Policy: Overview

1. Homeworks: 25%
2. Midterms: 45% (**2 × 22.5%**)
3. Finals: 30% (covers the full course content)

Midterms dates:

1. Midterm 1: Mon, Feb 26, 7–9pm, 1320 DCL
2. Midterm 2: Mon, Apr 9, 7–9pm, 1320 DCL
3. Final Exam: Fri, May 11

No conflict exam offered unless you have a valid reason (see course webpage).

Homeworks

1. One homework every week: Due on Wednesday at 8pm. To be submitted electronically in pdf form in *Gradescope*. Assigned at least a week in advance.
2. Homeworks can be worked on in groups of up to 3 and each group submits *one* written solution **(except Homework 0)**.
3. Important: academic integrity policies. See course web page.

More on Homeworks

1. No extensions or late homeworks accepted.
2. To compensate, five problems will be dropped. Homeworks typically have three problems each.
3. Important: Read homework faq/instructions on website.

Advice

1. Attend lectures, please ask plenty of questions.
2. Don't skip homework and don't copy homework solutions.
3. Study regularly and keep up with the course.
4. This is a course on problem solving. Solve as many as you can! Books/notes have plenty.
5. Ask for help promptly. Make use of office hours/Piazza.
6. This is an optional mixed undergrad/grad course. (Mathematical) maturity and independence are expected.

Homework 0

1. HW 0 is posted on the class website.
2. HW 0 due on Wednesday Jan 24 at 8pm
3. HW 0 to be done and submitted individually.

Miscellaneous

Please contact instructors if you need special accommodations.

Lectures are being taped. A link to the videos will be put up on course webpage.

Emergencies: see information at link http://police.illinois.edu/dpsapp/wp-content/uploads/2017/12/CEOP-2017.pdf

Part II

Course Goals and Overview

Course Structure

Course divided into four parts:

1. Recursion, dynamic programming.
2. Randomization in algorithms
3. Combinatorial and Discrete Optimization: flows/cuts, matchings, introduction to linear and convex programming
4. Intractability and heuristics

Course Goals

Mostly algorithms:

1. Some fundamental problems and algorithms
   1. FFT, Hashing, Flows/Cuts, Matchings, LP, approximation, ...
2. Broadly applicable techniques in algorithm design
   1. Recursion, Divide and Conquer, Dynamic Programming
   2. Randomization in algorithms and data structures
   3. Optimization via convexity and duality
   4. Approximation and heuristics
   5. Role of mathematics in algorithm design: graph theory, (linear) algebra, geometry, convexity ···

Complexity and Algorithms

- ***P***: class of problems that can solved in polynomial time
- ***EXP***: class of problems that can be solved in exponential time
- ***NP***: non-deterministic polynomial-time.
- ***DECIDABLE***: class of problems that have an algorithm

Complexity and Algorithms

- ***P***: class of problems that can solved in polynomial time
- ***EXP***: class of problems that can be solved in exponential time
- ***NP***: non-deterministic polynomial-time.
- ***DECIDABLE***: class of problems that have an algorithm

Theorem

There exist (many) undecidable problems.

Complexity and Algorithms

- ***P***: class of problems that can solved in polynomial time
- ***EXP***: class of problems that can be solved in exponential time
- ***NP***: non-deterministic polynomial-time.
- ***DECIDABLE***: class of problems that have an algorithm

Theorem

There exist (many) undecidable problems.

Theorem

P *is a strict subset of* ***EXP****.*

Complexity and Algorithms

- P: class of problems that can solved in polynomial time
- EXP: class of problems that can be solved in exponential time
- NP: non-deterministic polynomial-time.
- $DECIDABLE$: class of problems that have an algorithm

Theorem

There exist (many) undecidable problems.

Theorem

P is a strict subset of EXP.

$P \subseteq NP \subseteq EXP$. Major open problem: Is $P = NP$?

Complexity and Algorithms

- ***P***: class of problems that can solved in polynomial time
- ***EXP***: class of problems that can be solved in exponential time
- ***NP***: non-deterministic polynomial-time.
- ***DECIDABLE***: class of problems that have an algorithm

Theorem

There exist (many) undecidable problems.

Theorem

P *is a strict subset of* ***EXP****.*

$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{EXP}$. Major open problem: Is $\mathbf{P} = \mathbf{NP}$?
Many useful and important problems are *intractable*: ***NPComplete***, ***EXPComplete***, ***UNDECIDABLE***.

Complexity and Algorithms

Goals of algorithm desgin.

- find the "best" possible algorithm for some spefic problems of interest
- develop broadly applicable techniques for algorithm design

Goals of complexity:

- prove lower bounds for specific problems
- develop broadly applicable techniques for proving lower bounds
- develop complexity classes to characterize many problems

Rich interplay between the two areas.

Important Ingredients in Algorithm Design

1. What is the problem (really)?
   1. What is the input? How is it represented?
   2. What is the output?
2. What is the model of computation? What basic operations are allowed?
3. Algorithm design
   - Understand the structure of the problem
   - Relate it to standard and known problems via reductions
   - Try algorithmic paradigms: recursion, divide and conquer, dynamic programming, greedy, convex optimization, $\cdots$
   - Failing, try to prove a lower bound via reduction to existing hard problems or settle for approximation, heuristics.
4. Proving correctness of algorithm
5. Analysis of time and space complexity
6. Algorithmic engineering

Recall: Time (Space) Complexity and Notations

Representing running time of an algorithm on an n sized input:

- Upper bound $O(f(n))$: Takes at most $c \cdot f(n)$ time for some constant $c \in \mathbb{R}_+$.
- Lower bound $\Omega(f(n))$: Takes at least $c \cdot f(n)$ time for some constant $c \in \mathbb{R}_+$.
- Tight bound $\Theta(f(n))$: Takes at least $c \cdot f(n)$ and at most $c' \cdot f(n)$ time for some $c, c' \in \mathbb{R}_+$.