CS/ECE 374: Algorithms & Models of Computation

Chandra Chekuri

University of Illinois, Urbana-Champaign

Fall 2018

Administrivia, Introduction

Lecture 1

August 28, 2018

Part I

Administrivia

Instructional Staff

1. Instructors: Chandra Chekuri (A section) and Nikita Borisov (B section)
2. 11 Teaching Assistants
3. ?? Undergraduate Course Assistants
4. Office hours: See course webpage
5. Contacting us: Use *private notes* on Piazza to reach course staff. Direct email only for sensitive or confidential information.

Section A vs B

Only lectures different for the sections.

Home work, exams, labs etc will be common.

Homework groups can be across sections.

Online resources

1. Webpage: General information, announcements, homeworks, course policies `courses.engr.illinois.edu/cs374`
2. Gradescope: Homework submission and grading, regrade requests
3. Moodle: Quizzes, solutions to homeworks, 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

Prereqs and Resources

1. Prerequisites: CS 173 (discrete math), CS 225 (data structures)
2. Recommended books: (not required)
    1. Introduction to Theory of Computation by Sipser
    2. Introduction to Automata, Languages and Computation by Hopcroft, Motwani, Ullman
    3. Algorithms by Dasgupta, Papadimitriou & Vazirani. Available online for free!
    4. Algorithm Design by Kleinberg & Tardos
3. Lecture notes/slides/pointers: available on course web-page
4. Additional References
    1. Lecture notes of Jeff Erickson, Sariel HarPeled, Mahesh Viswanathan and others
    2. Introduction to Algorithms: Cormen, Leiserson, Rivest, Stein.
    3. Computers and Intractability: Garey and Johnson.

Grading Policy: Overview

1. Quizzes: 0% for self-study
2. Homeworks: 24%
3. Midterm exams: 44% (**2 × 22%**)
4. Final exam: 32% (covers the full course content)

Midterm exam dates:

1. Midterm 1: Mon, October 1, 7–9.30pm
2. Midterm 2: Mon, November 12, 7–9.30pm

No conflict exam offered unless you have a valid excuse.

Homeworks

1. Self-study quizzes each week on *Moodle*. No credit but stronlgy recommended.
2. One homework every week: Due on Wednesdays at 10am on *Gradescope*. Assigned at least a week in advance.
3. Homeworks can be worked on in groups of up to 3 and each group submits *one* written solution (except Homework 0).
4. Important: academic integrity policies. See course web page.

More on Homeworks

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

Discussion Sessions/Labs

1. 50min problem solving session led by TAs
2. Two times a week
3. Go to your assigned discussion section
4. Bring pen and paper!

Advice

1. Attend lectures, please ask plenty of questions.
2. Attend discussion sessions.
3. Don't skip homework and don't copy homework solutions. Each of you should think about *all* the problems on the home work - do not divide and conquer.
4. Use pen and paper since that is what you will do in exams which count for 76% of the grade. Keep a note book.
5. Study regularly and keep up with the course.
6. This is a course on problem solving. Solve as many as you can! Books/notes have plenty.
7. This is also a course on providing rigourous proofs of correctness. Refresh your 173 background on proofs.
8. Ask for help promptly. Make use of office hours/Piazza.

On Learning

Without seeking, truth cannot be known at all. It can neither be declared from pulpits, nor set down in articles nor in any wise be prepared and sold in packages ready for use. Truth must be ground for every man by himself out of its husk, with such help as he can get, but not without stern labour of his own.

– John Ruskin

Homework 0

1. HW 0 is posted on the class website. Quiz 0 available on Moodle.
2. HW 0 due on Wednesady September 5th at 10am on Gradescope
3. HW 0 to be done and submitted *individually*.

Miscellaneous

Please contact instructors if you need special accommodations.

Lectures are being taped. See course webpage.

Part II

Course Goals and Overview

High-Level Questions

1. Computation, formally.
    1. Is there a formal definition of a computer?
    2. Is there a "universal" computer?
2. Algorithms
    1. What is an algorithm?
    2. What is an *efficient* algorithm?
    3. Some fundamental algorithms for basic problems
    4. Broadly applicable techniques in algorithm design
3. Limits of computation.
    1. Are there tasks that our computers cannot do?
    2. How do we prove lower bounds?
    3. Some canonical hard problems.

Course Structure

Course divided into three parts:

1. Basic automata theory: finite state machines, regular languages, hint of context free languages/grammars, Turing Machines
2. Algorithms and algorithm design techniques
3. Undecidability and NP-Completeness, reductions to prove intractability of problems

Goals

1. Algorithmic thinking
2. Learn/remember some basic tricks, algorithms, problems, ideas
3. Understand/appreciate limits of computation (intractability)
4. Appreciate the importance of algorithms in computer science and beyond (engineering, mathematics, natural sciences, social sciences, ...)

Historical motivation for computing

1. Fast (and automated) *numerical calculations*
2. Automating mathematical theorem proving

Models of Computation vs Computers

1. Model of Computation: an "idealized mathematical construct" that describes the primitive instructions and other details
2. Computer: an actual "physical device" that implements a very specific model of computation

Models of Computation vs Computers

1. Model of Computation: an "idealized mathematical construct" that describes the primitive instructions and other details
2. Computer: an actual "physical device" that implements a very specific model of computation

Models and devices:

1. Algorithms: usually at a high level in a model
2. Device construction: usually at a low level
3. Intermediaries: compilers
4. How precise? Depends on the problem!
5. Physics helps implement a model of computer
6. Physics also inspires models of computation

Adding Numbers

Problem Given two n-digit numbers x and y, compute their sum.

Basic addition

$$\begin{array}{r} \mathbf{3141} \\ \mathbf{+7798} \\ \hline \mathbf{10939} \end{array}$$

Adding Numbers

```
c = 0
for i = 1 to n do
    z = x_i + y_i
    z = z + c
    If (z > 10)
        c = 1
        z = z - 10       (equivalently the last digit of z)
    Else c = 0
    print z
End For
If (c == 1) print c
```

Adding Numbers

```
c = 0
for i = 1 to n do
    z = x_i + y_i
    z = z + c
    If (z > 10)
        c = 1
        z = z − 10        (equivalently the last digit of z)
    Else c = 0
    print z
End For
If (c == 1) print c
```

1. Primitive instruction is addition of two digits
2. Algorithm requires $O(n)$ primitive instructions

Multiplying Numbers

Problem Given two n-digit numbers x and y, compute their product.

Grade School Multiplication

Compute "partial product" by multiplying each digit of y with x and adding the partial products.

$$
\begin{array}{r}
3141 \\
\times 2718 \\
\hline
25128 \\
3141 \\
21987 \\
6282 \\
\hline
8537238
\end{array}
$$

Time analysis of grade school multiplication

1. Each partial product: $\Theta(n)$ time
2. Number of partial products: $\leq n$
3. Adding partial products: n additions each $\Theta(n)$ (Why?)
4. Total time: $\Theta(n^2)$
5. Is there a faster way?

Fast Multiplication

Best known algorithm: $O(n \log n \cdot 2^{O(\log^* n)})$ time [Furer 2008]

Previous best time: $O(n \log n \log \log n)$ [Schonhage-Strassen 1971]

Conjecture: there exists an $O(n \log n)$ time algorithm

Fast Multiplication

Best known algorithm: $O(n \log n \cdot 2^{O(\log^* n)})$ time [Furer 2008]

Previous best time: $O(n \log n \log \log n)$ [Schonhage-Strassen 1971]

Conjecture: there exists an $O(n \log n)$ time algorithm

We don't fully understand multiplication!
Computation and algorithm design is non-trivial!

Post Correspondence Problem

Given: Dominoes, each with a top-word and a bottom-word.

b	***ba***	***abb***	***abb***	***a***
bbb	***bbb***	***a***	***baa***	***ab***

Can one arrange them, using any number of copies of each type, so that the top and bottom strings are equal?

abb	***ba***	***abb***	***a***	***abb***	***b***
a	***bbb***	***a***	***ab***	***baa***	***bbb***

Halting Problem

Debugging problem: Given a program M and string x, does M halt when started on input x?

Halting Problem

Debugging problem: Given a program M and string x, does M halt when started on input x?

Simpler problem: Given a program M, does M halt when it is started? Equivalently, will it print "Hello World"?

Halting Problem

Debugging problem: Given a program M and string x, does M halt when started on input x?

Simpler problem: Given a program M, does M halt when it is started? Equivalently, will it print "Hello World"?

One can prove that there is no algorithm for the above two problems!