CS/ECE 374: Algorithms & Models of Computation

Andrew Miller, Haitham Hassanieh

University of Illinois, Urbana-Champaign

Spring 2020

Administrivia, Introduction

Lecture 1
Wednesday, January 22, 2020

LaTeXed: January 19, 2020   04:07

Part I

Administrivia

Instructional Staff

1. **Instructor:** Andrew Miller, Haitham Hassanieh
2. 10^9 students.
3. 3 Teaching Assistants
4. ?? Undergraduate Course Assistants
5. **Office hours:** See course webpage
6. **Contacting us:** Use *private notes* on Piazza to reach course staff. Direct email only for sensitive or confidential information.

Online resources

1. **Webpage:** General information, announcements, homeworks, course policies
   http://courses.engr.illinois.edu/cs374/sp2020/B
2. **Gradescope:** Homework submission and grading, regrade requests
3. **Piazza:** Announcements, online questions and discussion, contacting course staff (via private notes)

See course webpage for links

Important: check Piazza 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 Har-Peled, Mahesh Viswanathan and others
   2. Introduction to Algorithms: Cormen, Leiserson, Rivest, Stein.
   3. Computers and Intractability: Garey and Johnson.

Grading Policy: Overview

1. Homeworks: 30%
2. Midterm exams: 40% (**2 × 20%**)
3. Final exam: 30% (covers the full course content)

Midterm exam dates:

1. Midterm 1: Monday February 24: 7-9 pm, in 1002 ECEB.
2. Midterm 2: Monday April 13: 7-9 pm, in 1002 ECEB.

No conflict exam offered unless you have a valid excuse.

Homeworks

1. One homework every week: Due on Monday at 11pm on *Gradescope*. Assigned at least 6 days in advance.
2. Homeworks can be worked on in groups of up to 3 and each group submits *one* written solution.
3. Important: academic integrity policies. See course web page.

More on Homeworks

1. No extensions or late homeworks accepted.
2. To compensate, six to 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 70% 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 rigorous proofs of correctness. Refresh your 173 background on proofs.
8. Ask for help promptly. Make use of office hours/Piazza.

Homework 1

1. HW 1 is posted on the class website.
2. HW 1 due Monday, January 27, 2019 at 11pm on Gradescope.
3. Groups of size up to 3.

Miscellaneous

Please contact instructors if you need special accommodations.

Lectures are being taped. See course webpage.

Part II

Course Goals and Overview

```
532     continue;
        } while (++iN < iLength]) {
533         if (isdigit(sInput[iN]) ) {
534             continue;
                /*iN == (iLength - 3) ) {
```

Computer Programming To Be Officially Renamed "Googling Stackoverflow"

Washington DC - The IEEE have produced a report today where they strongly recommend that from now on, the discipline of Computer Programming should be officially renamed to "Googling Stackoverflow".

"We are recommending a root-and-branch name change to this discipline", said President of the IEEE, Thomas M. Conte. "We are even going to change the official name of the IEEE Computer Society to the IEEE Quick Look At StackOverflow Society".

"We are furthermore recommending that all universities across the planet should cease to award Bachelors degrees, Masters or PhDs in computer programming or software engineering and instead they should award these degrees in Googling Stackoverflow."

"We are also considering renaming "P-values" to "Reviewer Pacifiers"."

High-Level Questions

1. Modeling: States/Graphs/Recursion/Algorithms.
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. What is a mathematical definition of a computer?
   1. Is there a formal definition?
   2. Is there a "universal" computer?
4. What can computers compute?
   1. Are there tasks that our computers cannot do?

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

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

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!

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!