CS/ECE 374: Algorithms & Models of Computation

Chekuri and Hulett

University of Illinois, Urbana-Champaign

Spring 2025

Administrivia, Introduction

Lecture 1

January 21, 2025

Part I

Administrivia

Section A vs B

Independent courses. Content very similar but run **separately**.

This is Section A

Instructional Staff

1. Instructors: Chandra Chekuri and James Hulett
2. Approximately 475 students in Section A
3. 11 Teaching Assistants.
4. 24 Undergraduate Course Assistants
5. Office hours: See course webpage/Ed
6. Contacting us: Use *private notes* on EdStem to reach course staff. Direct email only for sensitive or confidential information.

Online resources

1. Webpage: General information, schedule, lecture slides, homeworks, course policies
   https://courses.engr.illinois.edu/cs374al1/sp2025/
2. EdStem: Announcements, online questions and discussion, contacting course staff (via private notes)
3. Mediaspace: Channel for lecture videos
4. Gradescope: Written homework submission & grading, regrades
5. PrairieLearn: Autograded assessments with short questions
6. Discord: banter

See course webpage for links

Important: check Ed/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 Jeff Erickson
   4. Algorithms by Dasgupta, Papadimitriou & Vazirani.
   5. 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. Homeworks: 28%
2. Midterm exams: 44% ($2 \times 22\%$)
3. Final exam: 28% (covers the full course content)

Midterm exam dates:

1. Midterm 1: Mon, Feb 24, 7 to to 9.30pm
2. Midterm 2: Mon, April 14 , 7 to 9.30pm
3. Final exam: Thur, May 15, 7 to 10pm (tentative)

Potential new policy: a PL based pre-test quiz for 10% of grade

No conflict exam offered unless you have a valid excuse.

Homework

Every week except during exams: total of 11

1. Short questions on *PrarieLearn*: Due Tuesday, 9am.
   - Individually done and submitted.
   - Multiple attempts with goal of mastering basic material
2. Written homework with 2 problems: Due Wednesday, 9am on *Gradescope*.
   - Written 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 to homeworks accepted.
2. To compensate:
   - **four** problems in written homework and **two** PrairieLearn quizzes will be dropped (corresponds to two whole home works).
   - For 25% penalty, can submit PL and home work till midnight of the deadline day.
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/pencil and paper!**

Advice

1. Attend lectures and labs. Ask questions.
2. Don't skip homework and don't copy homework solutions. Each of you should *think about all* the problems on the homework — do not divide and conquer.
3. Keep a note book/tablet. **Write** solutions since that is what you will do in exams which count for 70% of the grade.
4. Study regularly and keep up with the course.
5. This is a course on problem solving. Solve as many as you can! Books/notes have plenty.
6. This is also a course on providing rigourous proofs of correctness. Refresh your 173 background on proofs.
7. Have fun! Try to enjoy the material beyond grades/interviews. What is *computing*? What if some one asks you on a date?
8. Ask for help promptly. Make use of office hours/Ed/Discord.

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 1

1. HW 1 is posted on the class website. GPS1 available on PrarieLearn.
2. GPS 1 is due on Tuesday Jan 28th at 9am.
3. HW 1 due on Wednesady Jan 29th at 9am on Gradescope.

Miscellaneous

Please contact instructors if you need special accommodations (DRES and others). See course web page on accommodations and health/well-being.

Lectures are being taped. See course webpage.

Avoid the temptation to watch videos "later"

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 and Rigorous analysis
2. Formal models for computation

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

Historical motivation for computing

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

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: classical, quantum, biological, ...

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

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

Adding Numbers

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

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

- $O(n^{1.58})$ time [Karatsuba 1960] disproving Kolmogorov's belief that $\Theta(n^2)$ is the right answer
- $O(n \log n \log \log n)$ [Schonhage & Strassen 1971].
  Conjecture: $O(n \log n)$ time possible
- $O(n \log n \cdot 2^{O(\log^* n)})$ time [Furer 2008]
- $O(n \log n)$ [Harvey & van der Hoeven 2019]

Fast Multiplication

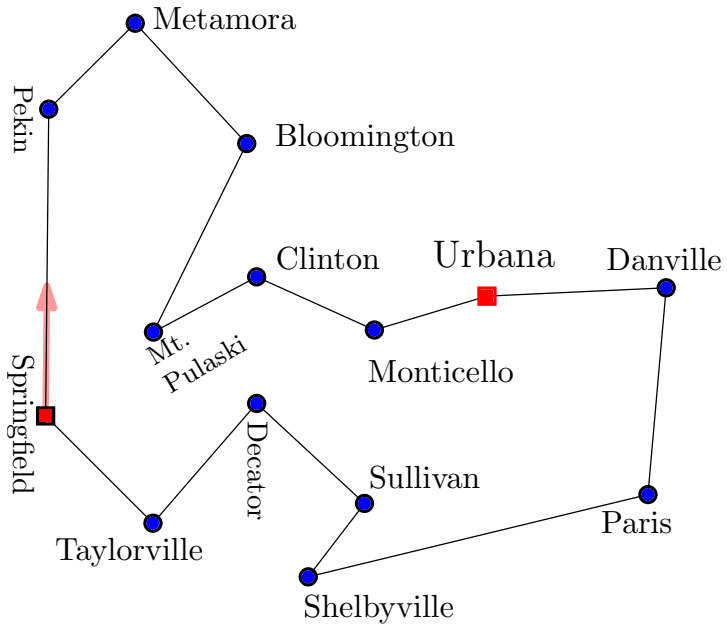
- $O(n^{1.58})$ time [Karatsuba 1960] disproving Kolmogorov's belief that $\Theta(n^2)$ is the right answer
- $O(n \log n \log \log n)$ [Schonhage & Strassen 1971].
  Conjecture: $O(n \log n)$ time possible
- $O(n \log n \cdot 2^{O(\log^* n)})$ time [Furer 2008]
- $O(n \log n)$ [Harvey & van der Hoeven 2019]

Can we achieve $O(n)$? No lower bound beyond trivial one!

Computation and algorithm design is non-trivial!

Lincoln and Traveling Judge Problem

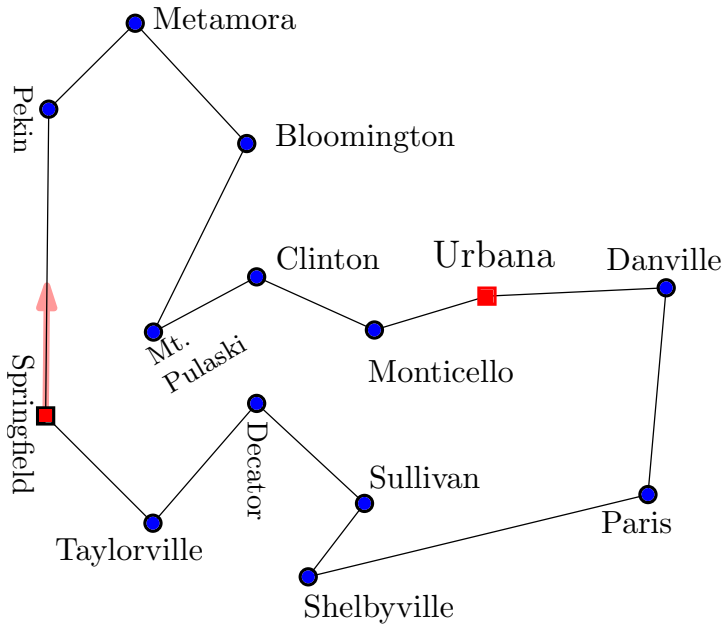
Lincoln was a circuit court judge. Had to visit towns.



Traveling Salesman Problem (TSP): Given a set of cities with distances, what is the shortest tour that visits all the cities?

Lincoln and Traveling Judge Problem

Lincoln was a circuit court judge. Had to visit towns.

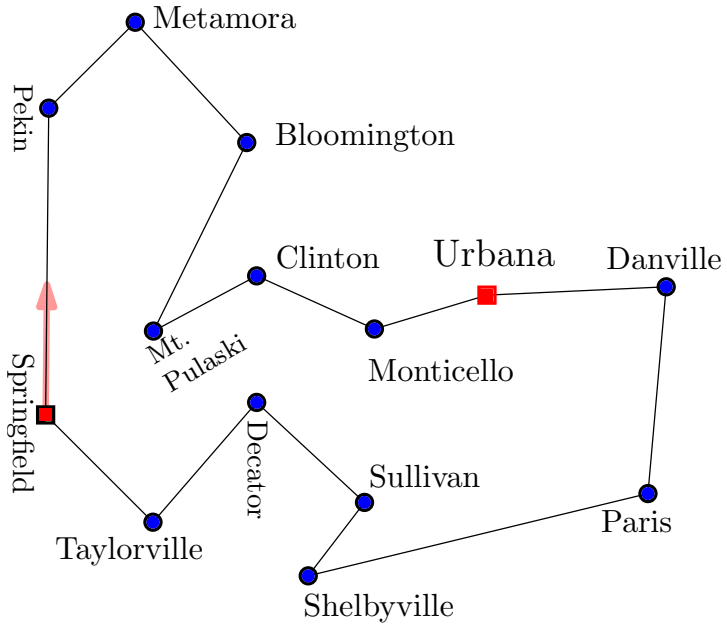


Traveling Salesman Problem (TSP): Given a set of cities with distances, what is the shortest tour that visits all the cities?

Easy to try all possible $n!$ permutations but exponential time algorithm. Is there an *efficient* algorithm?

Lincoln and Traveling Judge Problem

Lincoln was a circuit court judge. Had to visit towns.

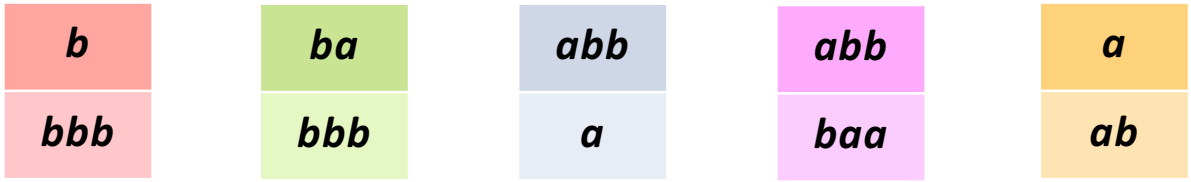


Traveling Salesman Problem (TSP): Given a set of cities with distances, what is the shortest tour that visits all the cities?

Easy to try all possible $n!$ permutations but exponential time algorithm. Is there an *efficient* algorithm? The famous P vs NP problem! One of the Clay millennium open problems.

Post Correspondence Problem

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



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! And also the Post Coresspondence Problem is as hard as Halting!