Week 14

Celebrating Illinois Computer Science

- December 8th (Thursday)
- 3pm
- Siebel Center
- Free crewneck sweatshirts and tasty treats

Chandra Chekuri
Theory and Algorithms

CS 100
Theory and Algorithms

Chandra Chekuri

Acknowledgements: Timothy Chan and Jeff Erickson for past slides/outline

The Area

- Theory A (algorithms & computational complexity) and Theory B (logic, automata, formal methods, PL etc). Theory B separate area (PL/FM/SE)

- Research driver
  - Foundational topics and questions
  - Applications, connections and collaborations inside and outside CS

- Education driver
  - Core theory classes in undergraduate program (CS 173, 225, 374, 473) and graduate curriculum (CS 473, 579, 583, 584, ….)
  - Research frontier, new developments, interdisciplinary (topics classes)

Faculty

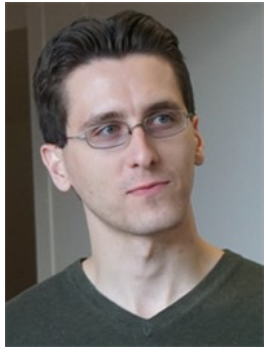
Timothy Chan

Karthik C

Chandra Chekuri

Payam Delgosha

Jeff Erickson



Michael Forbes

Jugal Garg

Sariel Har-Peled

Sheldon Jacobsen

Dakshita Khurana

Ruta Mehta

Matus Telgarsky

Research Interests

- **Timothy** (computational geometry, data structures, algorithms)
- **Karthik** (combinatorial optimization, graphs, math programming)
- **Chandra** (approx. algorithms, graphs, comb opt, math programming)
- **Payam** (information theory, graphs, ML)
- **Jeff** (computational geometry/topology, data structures, algorithms)
- **Michael** (complexity theory, algebraic computation, pseudorandomness)
- **Jugal** (algorithmic game theory, market algorithms, math programming)
- **Sariel** (comp geometry, geometric approximation, algorithms)
- **Sheldon** (operations research, stochastic methods, applications)
- **Dakshita** (cryptography, complexity, quantum)
- **Ruta** (algorithmic game theory, equilibrium computation, algorithms)
- **Matus** (ML theory, deep learning theory, optimization)

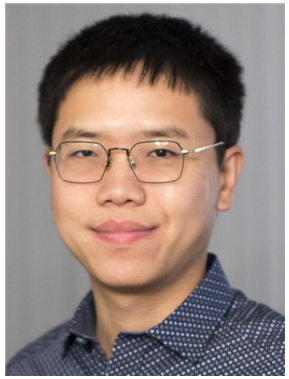
Other Faculty

Nancy Amato

Mohammed El-Kabir

Brighten Godfrey

Nan Jiang



Ling Ren

Edgar Solomonik

Mahesh V

Tandy Warnow

Research Interests

- **Nancy** (geometry/robotics, parallel algorithms, comp bio)
- **Mohammed** (algorithms for computational biology)
- **Brighten** (networking and related algorithms)
- **Nan** (reinforcement learning, ML, sample complexity)
- **Ling** (crypto, distributed algorithms, blockchains, security)
- **Edgar** (parallel algorithms, scientific and numerical computing)
- **Mahesh** (logic, formal verification, automata, algorithms)
- **Tandy** (algorithms for computational biology)

Many Related Faculty/Areas on Campus

- **Math (combinatorics, graph theory, geometry, logic, quantum, …)**
- **Statistics (probability, …)**
- **ISE (discrete/continuous optimization, OR, ML, applied probability, queuing)**
- **ECE (applied probability, information theory, crypto, ML, coding theory, quantum, …)**
- **Physics (quantum computing)**

What is theoretical computer science (TCS)?

Mathematical and formal aspects of computing

Study of formal problem solving on a computer/computational device/system

What is theoretical computer science (TCS)?

- *What is a **computer**? More generally, what is a model of computation? (Sequential, Parallel, Distributed, Quantum, State machines, Streaming …)*
- *What is a **problem?** What kind of **problems** do we want to solve?*
- *What is an **algorithm?** How can solve a computational problem with least amount of **resources (time, space, energy, randomness etc)?***
- *Can we **solve** all our computational problems? Can we prove that some problems cannot be solved? **Impossibility results and lower bounds***
- ***Computational** aspects of communication, cryptography, intelligence/AI, game theory, physics, biology, social sciences, …. Is the brain a computer? Does nature compute? What kind? How can we harness it?*
- ***Mathematics** necessary for computing*

Motivation/Impact

Practical/pragmatic: design *better*, *faster*, *correct*, *and robust* computers/algorithms/mechanisms to solve real-world problems, and understand limitations

Foundations: science/discipline of computing

Connections: mathematics, natural & social sciences, life …

Beauty:

Multiplication

Given two n digit numbers how fast can you multiply them?

```
            1891350210
          x 2019031847
          ------------
           13239451470
           7565400840
          15130801680
          1891350210
         5674050630
     17022151890
     1891350210
+ 3782700420
--------------------
  3818696307820137870
```

Grade school method: $O(n^2)$ steps

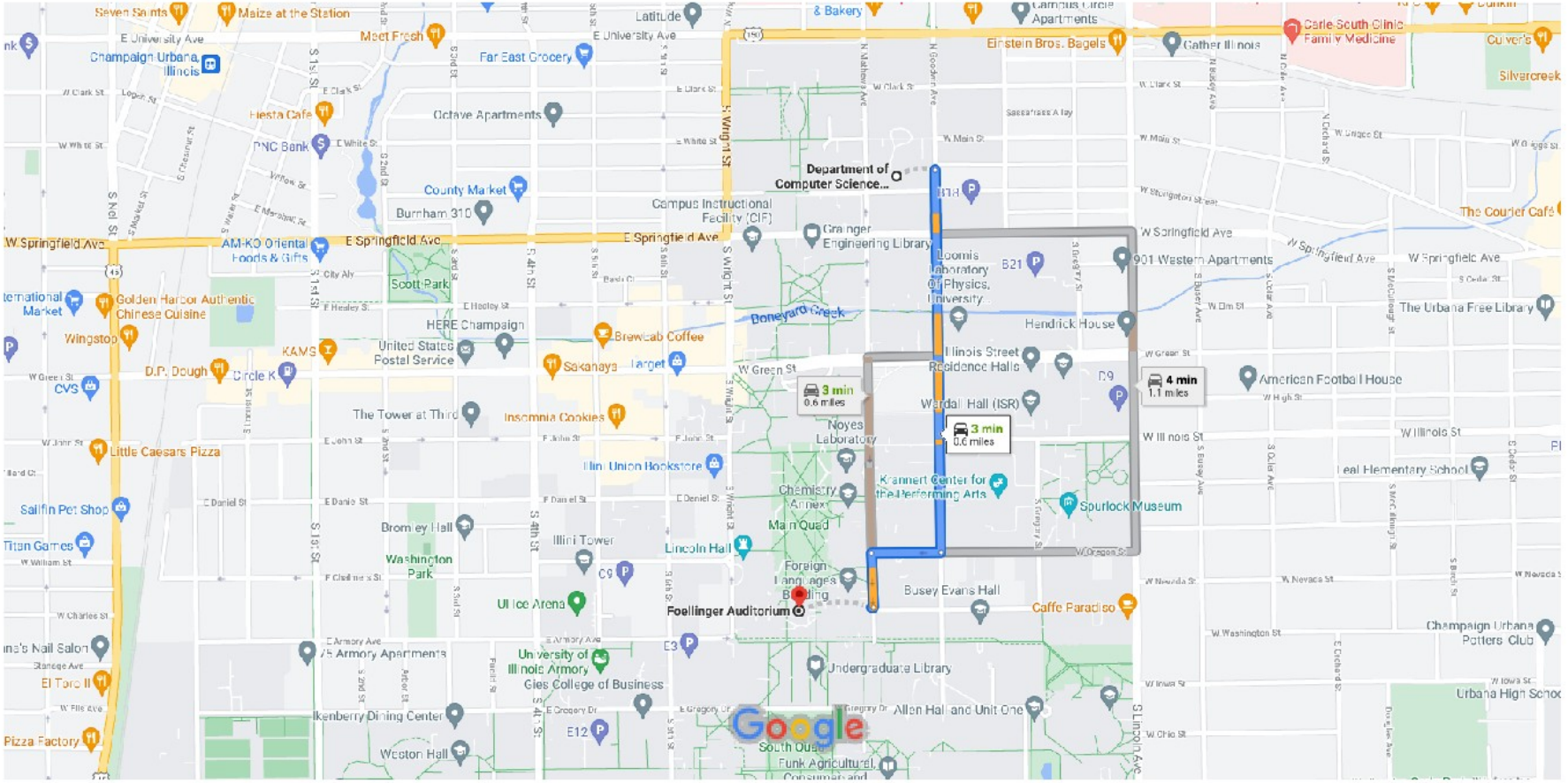
Not optimal! Can do in O(n log n) steps!
Claimed in 2019, before that *almost* O(n log n)

Primality and Factoring

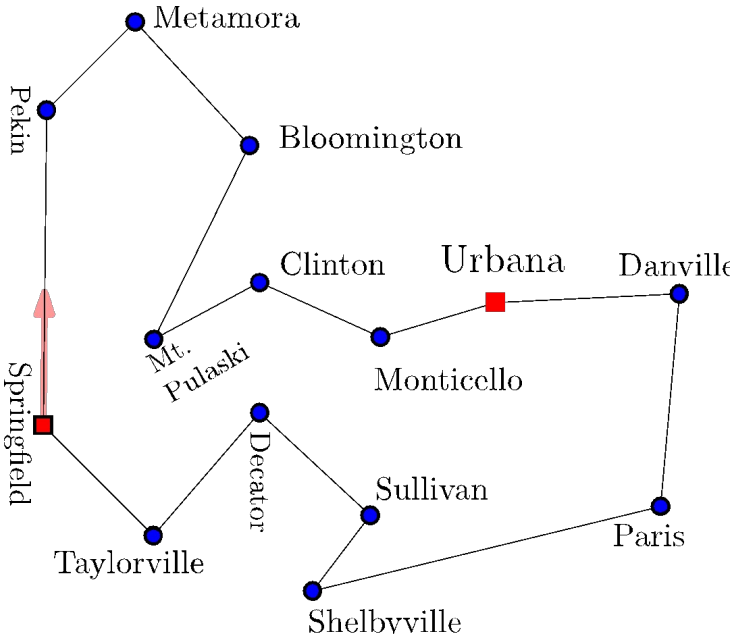
Given n digit number 10092003300140014003

- Is it a prime number?
- Can you find factor if it is not a prime?

- Primality can be checked efficiently [Miller-Rabin'80, Agarwal-Kayal-Saxena'02]
- Factoring – no known efficient algorithm. Modern cryptosystems (RSA) are based on hardness of factoring/discrete log
- Factoring has efficient algorithm on *quantum computers*! [Shor'97]


Department of Computer Science...
Foellinger Auditorium
3 min
0.6 miles
3 min
0.6 miles
4 min
1.1 miles
Google

Lincoln's Problem



Lincoln was a Circuit Court Judge

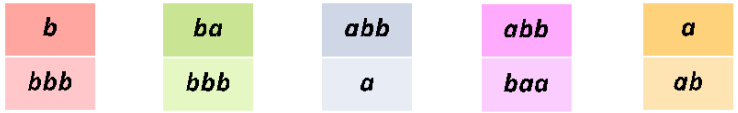
Travel to towns/villages for a few days each and return

Find shortest tour to visit all cities and come back (Traveling Judge Problem, popularly known as TSP)

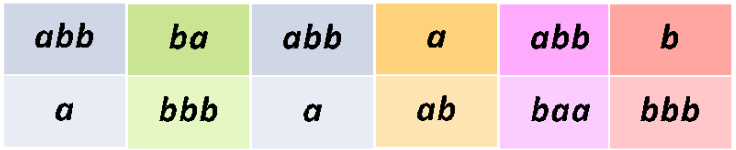
NP-Hard! No efficient algorithm known for exact solution.

Tiling Puzzle

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?



Undecidable! No computer algorithm that works correctly on all instances!

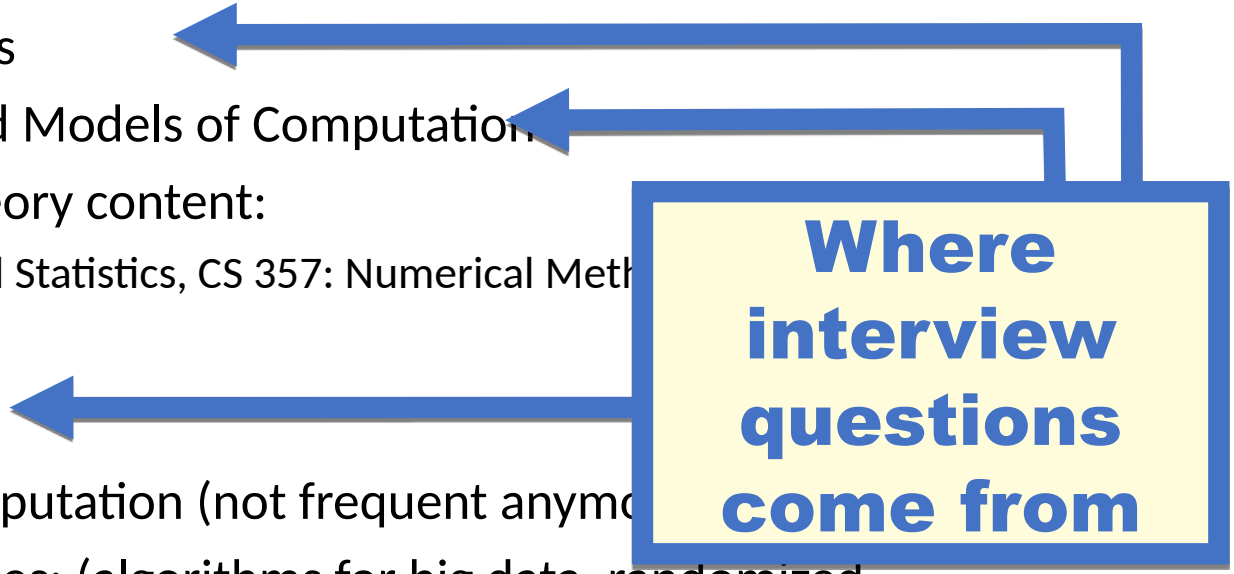
Undergrad Classes

Required

- CS 173: Discrete Mathematics
- CS 225: Data Structures
- CS 374: Algorithms and Models of Computation
- Other courses with theory content:
  - CS 361: Probability and Statistics, CS 357: Numerical Meth

Elective:

- CS 473: Algorithms
- CS 475: Theory of Computation (not frequent anymo
- CS 498s and new courses: (algorithms for big data, randomized, computational geometry, cryptography, …)



Standard/Semi-Regular Grad Classes

- Complexity Theory (every year)
- Approximation Algorithms (every other year)
- Randomized Algorithms (every other year)
- Combinatorial Optimization (every other year)
- Algorithmic Game Theory (every year)
- (Applied/Advanced) Cryptography (every year)
- Deep Learning Theory (every year)
- Many other *special topics* classes

Some Recent/Future Topics Courses

- Algorithms for 1-D structures (Jeff)
- Fine-grained Complexity Theory (Timothy)
- Algorithms for Big Data (Chandra)
- Computational Geometry (Timothy/Jeff/Sariel)
- Pseudorandomness (Michael)
- Geometric Data Structures (Timothy)
- Mathematical Methods and Algorithms in Large Graphs (Payam)
- Quantum Cryptography (Dakshita)
- Algebraic Computation (Michael)

Undergrad Opportunities

- We are **always** looking for qualified, motivated, and responsible course assistants for theory classes
- Grad theory classes: take CS 374/473 (or equivalent) first
- Individual study/research/senior thesis:
  - take 374 first, also additional ones, do well, show enthusiasm
  - approach faculty, talk to grad theory students
  - attend some theory seminars/talks, read independently, find a topic that you enjoy
- Please talk to us if you are interested

Seminars and Mailing Lists

- **Weekly theory seminar**
  - Mondays, 10:00 - 11:00 am
  - Talks by students/faculty/external speakers
- **Website: https://publish.illinois.edu/theory-cs/**
- theorycs@lists.cs.illinois.edu for generic theory related mailings including theory seminar
- Student mailing list and Slack channel

Some advice

Differentiate yourself from the many CS professionals with your theory skills!

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 [person] by himself [themselves] out of its husk, with such help as he can get, but not without stern labour of his [their] own.

– John Ruskin

Charith Mendis

Architecture, Compilers, and Parallel Computing