Lecture 1:
Introduction to the course

Instructors:
Venkatesan Guruswami
Anil Ada

August 30th, 2016
What is computer science?

Is it:

“Writing programs that do certain tasks.”

What is theoretical computer science?
Motivational Quote of the Course

“Computer Science is no more about computers than astronomy is about telescopes.”

- Edsger Dijkstra
What is computer science?

Is it branch of:
- science?
- engineering?
- math?
- philosophy?
- sports?
Physics

Theoretical physics
- come up with mathematical models
- derive the logical consequences

Nature’s language is mathematics

Experimental physics
- make observations about the universe
- test mathematical models with experiments

Applications/Engineering
The role of theoretical physics

Real World

Observed Phenomenon

Test Consequences
Applications

Abstract World

Mathematical Model

Explore Consequences
Theoretical Physics

- science?
- engineering?
- math?
- philosophy?
- sports?
The science that studies **computation**.

**Computation**: manipulation of information/data.

**Algorithm**: description of how the data is manipulated.

**Computational problem**: the input-output pairs.

![Diagram of computer process](input → “Computer” → output)
Computer Science

The science that studies computation.

**Computation**: manipulation of information/data.

**Algorithm**: description of how the data is manipulated.

**Computational problem**: the input-output pairs.

Usually

Input → **Calculator** → Output
Computer Science

The science that studies computation.

**Computation**: manipulation of information/data.

**Algorithm**: description of how the data is manipulated.

**Computational problem**: the input-output pairs.

![Diagram](Usually)

- **Input** → **Laptop** → **Output**
Computer Science

The science that studies computation.

Computation: manipulation of information/data.

Algorithm: description of how the data is manipulated.

Computational problem: the input-output pairs.
“Computers” in early 20th century
Computer Science

The science that studies **computation**.

**Computation**: manipulation of information/data.

**Algorithm**: description of how the data is manipulated.

**Computational problem**: the input-output pairs.

![Diagram](attachment:image.png)

Usually

Input → **Evolution** → Output
The computational lens

- Computational physics
- Computational biology
- Computational chemistry
- Computational neuroscience
- Computational economics
- Computational finance
- Computational linguistics
- Computational statistics
- …
Defining computer science

“Computer Science deals with the theoretical foundations of information and computation, together with practical techniques for the implementation and application of the foundations.”

- Wikipedia
The role of theoretical computer science

Build a mathematical model for computation.

Explore the logical consequences.  
Gain insight about computation.

Look for interesting applications.

CMU undergrad  
CMU Prof.  
OK, we don’t have everybody
The role of theoretical computer science

Real World

Computation

Applications

Abstract World

Mathematical Model

Explore Consequences

Only done recently
Simple examples of computation

We have been using algorithms for thousands of years.

\[
5127 \\
\times 4265 \\
\hline
25635 \\
307620 \\
1025400 \\
20508000 \\
\hline
21866655
\]
Simple examples of computation

We have been using algorithms for thousands of years.

Euclid’s algorithm (~ 300BC):

```python
def gcd(a, b):
    while (a != b):
        if (a > b):
            a = a - b
        else:
            b = b - a
    return a
```

We have been using algorithms for thousands of years.

Algorithm/Computation was only formalized in the 20th century!

Someone had to ask the right question.
The Problems of Mathematics

“Who among us would not be happy to lift the veil behind which is hidden the future; to gaze at the coming developments of our science and at the secrets of its development in the centuries to come? What will be the ends toward which the spirit of future generations of mathematicians will tend? What methods, what new facts will the new century reveal in the vast and rich field of mathematical thought?”
2 of Hilbert’s Problems

Hilbert’s 10th problem (1900)
Is there a finitary procedure to determine if a given multivariate polynomial with integral coefficients has an integral solution?

e.g. \[ 5x^2yz^3 + 2xy + y - 99xyz^4 = 0 \]

Entscheidungsproblem (1928)
Is there a finitary procedure to determine the validity of a given logical expression?

\[ \neg \exists x, y, z, n \in \mathbb{N} : (n \geq 3) \land (x^n + y^n = z^n) \]
(Mechanization of mathematics)
Fortunately, the answer turned out to be NO.
Gödel (1934):
Discusses some ideas for mathematical definitions of computation. But not confident what is a good definition.

Church (1936):
Invents lambda calculus. Claims it should be the definition of an “algorithm”.

Gödel, Post (1936):
Arguments that Church’s claim is not justified.

Meanwhile… in New Jersey… a certain British grad student, unaware of all these debates…
2 of Hilbert’s Problems

Alan Turing (1936, age 22):
Describes a new model for computation, now known as the Turing Machine.

Gödel, Kleene, and even Church:
“Umm. Yeah. He nailed it. Game over. “Algorithm” defined.”

Turing (1937):
TMs $\equiv$ lambda calculus
Formalization of computation: Turing Machine

Turing Machine:

Infinite Tape

1 0 0 0 1 1 1 1 0

Read / Write Head

Control Unit

State: Y

...
Church-Turing Thesis:

The intuitive notion of “computable” is captured by functions computable by a Turing Machine.

(Physical) Church-Turing Thesis

Any computational problem that can be solved by a physical device, can be solved by a Turing Machine.
Back to Hilbert’s Problems

Hilbert’s 10th problem (1900)
Is there a finitary procedure to determine if a given multivariate polynomial with integral coefficients has an integral solution?

\[ 5x^2yz^3 + 2xy + y - 99xyz^4 = 0 \]

Entscheidungsproblem (1928)
Is there a finitary procedure to determine the validity of a given logical expression?

\[ \neg \exists x, y, z, n \in \mathbb{N} : (n \geq 3) \land (x^n + y^n = z^n) \]

(Mechanization of mathematics)
Back to Hilbert’s Problems

Hilbert’s 10th problem (1900)

Is there an algorithm (a TM) to determine if a given multivariate polynomial with integral coefficients has an integral solution?

e.g. \( 5x^2yz^3 + 2xy + y - 99xyz^4 = 0 \)

Entscheidungsproblem (1928)

Is there an algorithm (a TM) to determine the validity of a given logical expression?

e.g. \( \neg \exists x, y, z, n \in \mathbb{N} : (n \geq 3) \land (x^n + y^n = z^n) \)

(Mechanization of mathematics)
Hilbert’s 10th problem (1900)

Matiyasevich-Robinson-Davis-Putnam (1970):

There is no algorithm to solve this problem.

Entscheidungsproblem (1928)

Turing (1936):
There is no algorithm to solve this problem.
Computer science

- science?
- engineering?
- math?
- philosophy?
- sports?
2 Main Questions in TCS

**Computability** of a problem:
Is there an algorithm to solve it?

**Complexity** of a problem:
Is there an *efficient* algorithm to solve it?
- time
- space (memory)
- randomness
- quantum resources
Computational Complexity

Complexity of a problem:

Is there an efficient algorithm to solve it?

- time
- space (memory)
- randomness
- quantum resources

2 camps:

- trying to come up with efficient algorithms (algorithm designers)
- trying to show no efficient algorithm exists (complexity theorists)
Computational Complexity

2 camps:

- trying to come up with efficient algorithms (algorithm designers)
- trying to show no efficient algorithm exists (complexity theorists)

- multiplying two integers
- factoring integers
- sorting a list
- protein structure prediction
- simulation of quantum systems
- computing Nash Equilibria of games
Some other interesting questions

If a problem has a space-efficient solution does it also have a time-efficient solution?

Can every randomized algorithm be derandomized efficiently?

Can we use quantum properties of matter to build faster computers?

P vs NP
What will you learn in this course?
Topics Overview

**Part 1:** Formalizing the notions of problems, algorithms, and computability.

**Part 2:** Efficient computation: basic algorithms and complexity

**Part 3:** Highlights of theoretical CS and the mathematics behind them.
This is a “big picture” course

Finite automata

Turing machines

Graph theory

NP-completeness

Combinatorial games

Approximation algorithms

Group theory

Probability

Error correcting codes

Cryptography

Interactive proofs

Fields and polynomials

Communication complexity

Generating functions

Markov chains

Randomized algorithms

Basic number theory
Goals

1. Learn about the theoretical foundations of computation.
2. Learn the mathematical language and tools we need.
4. Become better at rigorous, logical, abstract thinking.
5. Become better at expressing yourself clearly.
6. Become better at working with other people.
This is a challenging course

What Kind of Mindset Do You Have?

**Growth Mindset**
- I can learn anything I want to.
- When I'm frustrated, I persevere.
- I want to challenge myself.
- When I fail, I learn.
- Tell me I try hard.
- If you succeed, I'm inspired.
- My effort and attitude determine everything.

**Fixed Mindset**
- I'm either good at it, or I'm not.
- When I'm frustrated, I give up.
- I don't like to be challenged.
- When I fail, I'm no good.
- Tell me I'm smart.
- If you succeed, I feel threatened.
- My abilities determine everything.
A review of the course syllabus
A quick review of the course syllabus

Course webpage: www.cs.cmu.edu/~15251
A quick review of the course syllabus

Grading:

11 homework assignments, lowest 2 half-weighted
30%

2 midterm exams
20% + 20% = 40%

1 final exam
25%

Participation (attending classes and recitations)
5%
What is your favorite TV show?
- Game of Thrones
- Breaking Bad
- Seinfeld
- Friends
- The Wire
- Sesame Street
- None of the above
- I don’t watch TV!
Homeworks

Most important part of the course!

They are meant to be challenging.

Make use of the office hours!!!

Homeworks prepare you for the exams. Seriously!
Homework System:

3 types of questions:
SOLO, GROUP, OPEN COLLABORATION

SOLO - work by yourself

GROUP - work in groups of 3 or 4

OPEN - work with anyone you would like from class
Homework System:

3 types of questions:
   SOLO, GROUP, OPEN COLLABORATION

Don’t share written material with anyone.

Erase public whiteboard when done.

Can search books to learn more about a subject.

**Can’t** Google specific keywords from the homework.

Always cite your sources!

Think about a problem before you collaborate.
Homeworks

Homework System:

Homework writing sessions:
   Wednesdays 6:30pm to 7:50pm at DH 2315

Write the solutions to a random subset of the problems.

You must practice writing the solutions beforehand!!!

You will lose points for poor presentation.

You get 25% of the credit for the question if you write:
   - nothing
   - “I don’t know”, or
   - “WTF!”
Homework System:

Feedback/grading:
Done by recitation on Friday.

You will know who graded which question.

Go see TA if:
- you think there has been a mistake in grading
- you don’t understand why you lost points
Everyone must sign up.

Course announcements will be made on Piazza. You have to check it every day.

Great resource, make use of it.

Please be polite.

Don’t give away any hints.
Office hours

See course webpage.

You have to use the OHs!
A typical week

<table>
<thead>
<tr>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
</table>

Lecture 1
Office hour (Anil)
A typical week

<table>
<thead>
<tr>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
</table>

Lecture 1.5 (6:30 - 7:50pm)
A typical week

<table>
<thead>
<tr>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
</table>

Lecture 2
Office hour (Anil)
Review that week’s material.
Homework comes out.
Maybe start working on the SOLO problems.
<table>
<thead>
<tr>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
</table>

Recitation

Make progress on SOLO problems.

Start thinking about the GROUP problems.

Make appointments to meet with your group over the weekend.
A typical week

Meet with your group.
Make some progress on the questions.
Maybe solve some of them.
Go to office hours.
A typical week

<table>
<thead>
<tr>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Meet with your group.
Go to office hours, get some help.
Solve some more problems.
A typical week

<table>
<thead>
<tr>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
</table>

Finish up GROUP problems.

Go to office hours.
A typical week

Realize that you still need to do the OPEN problem(s)!

Express hate towards the professors.

Lecture

Rush to OH to get help.

Don’t sleep until you solve the hardest problem.
A typical week

<table>
<thead>
<tr>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
</table>

Practice writing up the solutions to the problems.
Realize you have a mistake in one of the questions.

Express hate towards the professors.

*Learning moment:* write solution down once you think you figured it out.
Keys to success in this course

- Be awake during lectures, and review them on time.
- Use office hours. Use Piazza.
- Find good group members.
- If you are not happy with your group, break up.
- Take the “writing up the proof” part seriously.
- Make sure you understand the mistakes you make.
- Embrace the challenge.