Week 0

This is CS50. Harvard University. Fall 2014.

Cheng Gong

Table of Contents

This is CS50.

- This is CS50 2014[1] plays highlights from past years' events, including the CS50 Fair and CS50 Hackathon.
- CS50 is "an introduction to the intellectual enterprises of computer science and the art of programming."
- Remember that 78% of students who take CS50 have no prior experience.
    # The 22% with prior experience will also be challenged by different sections and problem set editions.
- "what ultimately matters in this course is not so much where you end up relative to your classmates but where you, in Week 12, end up relative to yourself in Week 0"
- Let's call computer science "the science of computation": running some algorithm, or set of instructions, on inputs, to produce some output, or solution.
- Susan Wojcicki[2] of YouTube speaks of her experience with CS50 and how it "changed how [she] think[s] about everything."
- Computer science isn't just programming; programming is a tool to be applied to any field.

[1] http://www.youtube.com/watch?v=zLo7CoiKIto
[2] http://www.youtube.com/watch?v=y1121-De4o4

- What Most Schools Don't Teach[3] by Code.org stars Bill Gates, Mark Zuckerberg, and other creators of technology, who talk about their beginnings with programming.

Binary

- Computers only understand zeroes and ones, an alphabet called **binary**. While humans use **decimal**, which has 10 digits 0-9, computers only understand binary, which has two digits, 0 and 1.
- With just two digits, we can still represent almost every possible piece of information.
- In decimal, `123` is one hundred and twenty-three, with each digit in a column:

```
    100         10          1

      1          2          3

100 x 1     10 x 2      1 x 3
```

- Binary represents numbers in the same pattern, but using powers of 2 instead of powers of 10 that decimal uses. The first row shows the value of each column, like the 100, 10, and 1 above, and the second row is our current binary number.

```
      4          2          1

      0          0          0
```

- To represent a 1, we simply place a `1` in the ones column:

```
      4          2          1

      0          0          1

   1 x 1
```

- And a 2 like so:

```
      4          2          1

      0          1          0
```

[3] http://www.youtube.com/watch?v=nKIu9yen5nc

```
        2 x 1
```

- And a 3 by combining the previous two steps:

```
    4       2       1

    0       1       1

        2 x 1   1 x 1
```

- We continue incrementing:

```
    4       2       1

    1       0       0

4 x 1
```

```
    4       2       1

    1       0       1

4 x 1           1 x 1
```

```
    4       2       1

    1       1       0

4 x 1   2 x 1
```

```
    4       2       1

    1       1       1

4 x 1   2 x 1   1 x 1
```

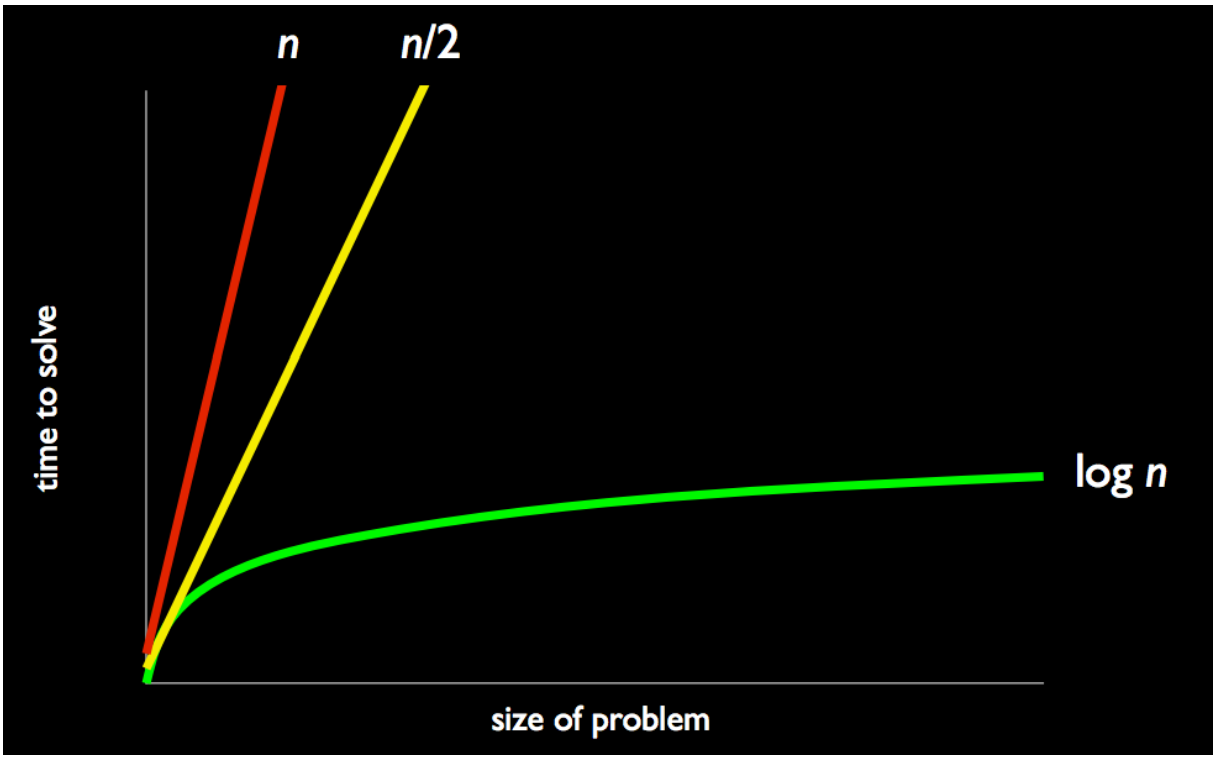
- But once we have used up all the places, we need more **bits**, or binary digit, which stores a `0` or `1`. With additional digits, we can represent larger numbers.

- A **transistor** in a computer can be represented by a light bulb or a switch. A light bulb turned off is like a `0` and a light bulb on is like a `1`.
- Companies now make light bulbs that change color, and you can control them over a network using an **API**, application programming interface, or the way through which the light bulbs can be programmed. You can send them messages, and even change color based on weather or time.
- Ansel Duff '15 and Dan Bradley '14 built binary bulbs, a row of eight light bulbs that can be controlled by an iPad app.
    # Jackie, a volunteer from the audience, uses the app to turn on the correct light bulbs to represent the number 15.
    # Alex, another volunteer, participates in the "Hacker edition", with a larger number, 50, to represent, and without the assistance of the place-value magnets.
- Now we can do more, by using **ASCII**, a standard that maps numbers to letters. For example, `A` is mapped to `65`, `B` is `66`, etc. Though bits can only store numbers, programs can translate those bits to letters for humans to easily read.

Algorithms

- **Algorithms** are sets of instructions to solve particular problems, taking inputs and producing outputs that are hopefully correct, but also efficient.
- The famous phone book example:
    # We can open to the first page, and look for someone, say, Mike Smith. Then we continue to the second, third, and so on until we find Mike Smith. That algorithm is correct because we will find him eventually, but isn’t very fast.
    # We can optimize and flip two pages at a time, and it is twice as fast as the previous algorithm.
    # Humans reasonably open to the middle, and since the phone book is alphabetical and the middle is the M section, we can literally tear this problem in half by ripping the phone book in half and keeping only the half that we think Mike Smith is in. We repeat this again and again, and eventually find one page. With 1000 pages, it would only take about 10 steps of division to reach that one page.

- Graphically, we can see the differences in efficiency as below. With `n` as the size of the problem, the red line represents the first algorithm in which time to solve increases with the size of the problem:



  # The yellow line is the second algorithm, which, though twice as fast, still increases linearly with the size of the problem.

  # The green line will have a **logarithmic slope** that doesn't increase in height as much as the other lines. With the phone book, even if the size of the phone book doubled, it would only take one more step to solve the problem.

- We have to assume that the phone book is alphabetized, since we wouldn't be sure where to go in a phone book with random names.

- **Pseudocode** is describing code for a problem in English:

```
pick up phone book
open to middle of phone book
look at names
if "Smith" is among names
    call Mike
else if "Smith" is earlier in book
    open to middle of left half of book
    go to line 3
else if "Smith" is later in book
    open to middle of right half of book
    go to line 3
else
    give up
```

1. Note that line 4 is a decision point, or branch, and we indent line 5 to convey the idea that we only `call Mike` if that branch is followed.
1. Line 6, `else if`, is another branch, in which we go to the left half of the book.
1. With line 8, `go to line 3`, we introduce a loop, in which we reuse the same instructions since we can continue solving the problem the same way.
1. If we reach line 12, then Mike Smith isn't in the phone book, and we should `give up`.

Syllabus

- **SAT/UNS** is a grading option for students who are a bit uneasy, and students are encouraged to start the course SAT/UNS and continue SAT/UNS or switch to letter-graded later in the semester, if at all on the fence.
    1. Even David took CS50 pass/fail, the equivalent of SAT/UNS back in the day, and now he's here!
- **Simultaneous enrollment** is allowed for CS50. See the **syllabus**[4] for more details.
- **Lectures**[5] will generally only meet for an hour, 1pm - 2pm, to provide a conceptual overview and demonstrations.

[4] http://cs50.harvard.edu/syllabus
[5] http://cs50.harvard.edu/lectures

- **Sections**[6] will be weekly, with different tracks for students less comfortable, more comfortable, and in between.
- **Problem sets** come in standard and hacker editions, which offer no extra credit but a more challenging approach to the same ideas. Walkthroughs provide hints to starting and completing the problem sets, and postmortems will be explanations of the problem sets after the fact.
- 5 late days are given to you, and the lowest grade will be dropped. More details in the **syllabus**[7].
- The first problem set last year introduced Scratch, a graphical programming language. Then we proceeded to a problem set with cryptography, scrambling messages. Another problem set implemented a game of Breakout, with a ball bouncing around breaking bricks. Later in the semester we'll write a program that spell checks a document with a given dictionary. At the end of the semester we'll look at web programming, and how we can apply concepts to the web in making dynamic websites that solve problems.
- The **final project** is almost any project you choose to implement, with the **CS50 Hackathon** an overnight event to work on it, and the **CS50 Fair** a time to show off your project to thousands of students and staff from across campus.
- **Office hours**[8] will be four nights a week, 8pm - 11pm, to provide support for problem sets.
- **Tutoring** will be available for students who are least comfortable.
- **Staff**[9] include Daven, Rob, and Gabriel, this year's heads, who can be reached at heads@cs50.harvard.edu[10]
- **CS50 Puzzle Day** is this Saturday, with problem-solving rather than programming being the focus. Register at http://cs50.harvard.edu/register.
- Finally, Colton Ogden demonstrates the applicability of computer science to domains like music, playing a song for Week 0[11].

[6] http://cs50.harvard.edu/sections
[7] http://cs50.harvard.edu/syllabus
[8] http://cs50.harvard.edu/hours
[9] http://cs50.harvard.edu/staff
[10] mailto:heads@cs50.harvard.edu
[11] http://www.youtube.com/watch?v=1uIN0I8VCeE