Introduction to Computational Thinking

Lecture #1: Introduction

Qiao Xiang, Qingyu Song

https://sngroup.org.cn/courses/ct-xmuf25/index.shtml

9/24/2025

This deck of slides are heavily based on cs112 at Yale University and cs101 at UCAS, respectively, by courtesy of Dr. Y. Richard Yang and Dr. Zhiwei Xu.

Outline

- *What is Computer Science?*
- What is Programming?
- Administrative trivia's
- Summary

What are CS and CT?

- Computer science (CS) is the study of computational processes
  - for problem solving and creative expression
  - that are correct, smart, and practical
- CS combines
  - logic, algorithmic, systems thinking, and network thinking
- Computational thinking (CT) is the way of thinking underlying the computer science discipline

ALGORITHMIC THINKING?

- Late ~~lonely~~ nights in front of computers?
- Computers?

ALGORITHMIC THINKING

al·go·rithm:

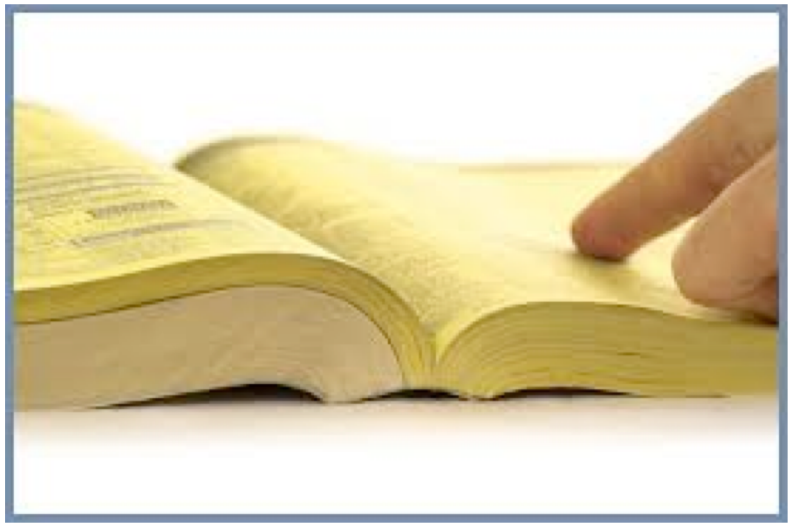
a step-by-step procedure for solving a problem or accomplishing some end, *especially by computers.*

Fields of Computer Science

- Artificial intelligence
- Databases
- Graphics and multimedia
- Mobile computing
- Systems/networks
- Theory
- ...

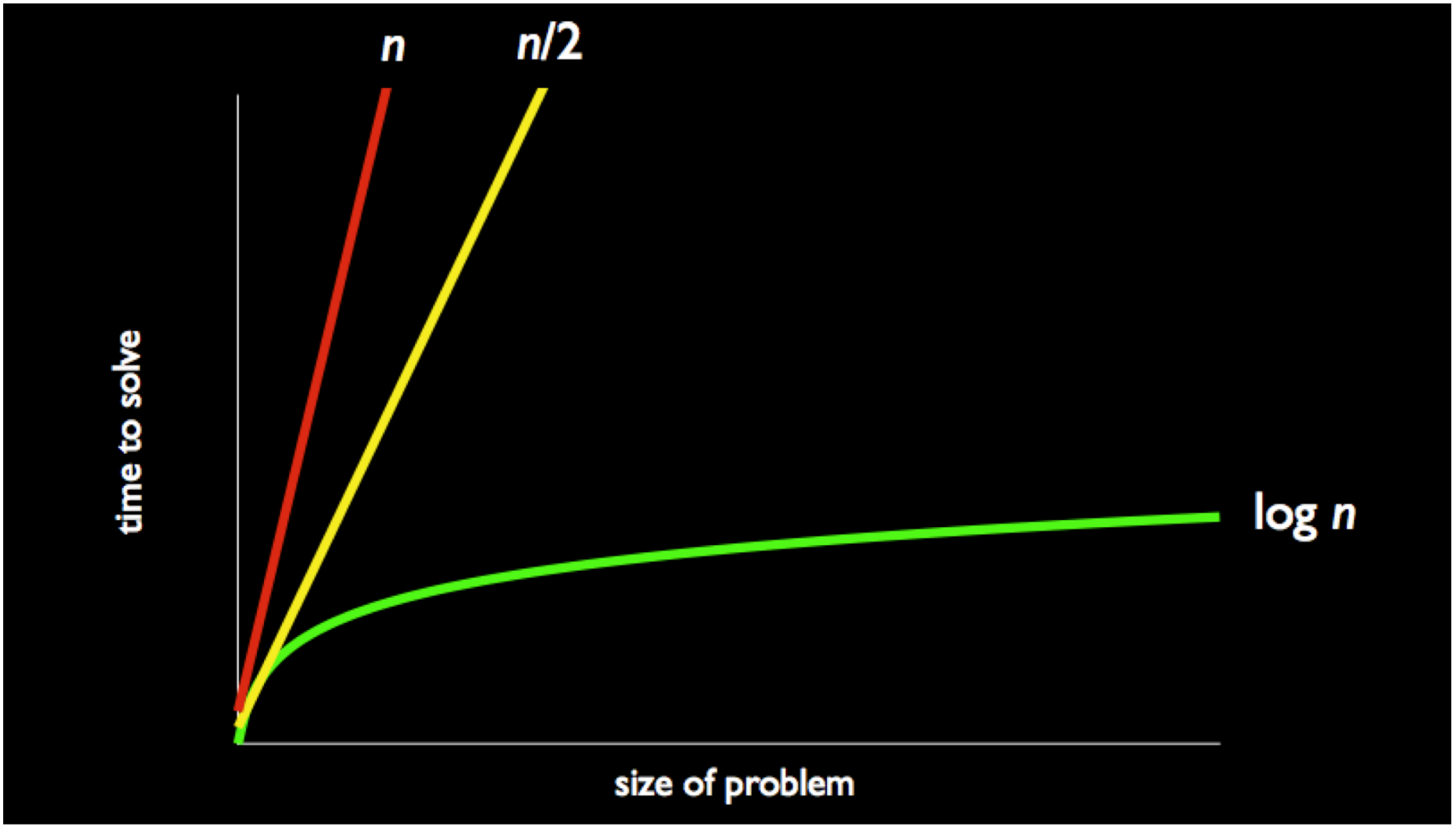
Algorithm Example: Phonebook

- Problem: How to look up a name in the telephone directory?



Src: http://www.123rf.com/photo_3534051_phone-directory-on-isolated-white-background.html

Comparison of Complexity



http://d2o9nyf4hwsci4.cloudfront.net/2013/fall/lectures/0/w/notes0w/notes0w.html

Algorithm Example: Counting

- Problem: How to count the number of students in a classroom?

Src: http://www.flickr.com/photos/usag-yongsan/5587562543/

Google's Map-Reduce Scheme

- Initialize all inputs

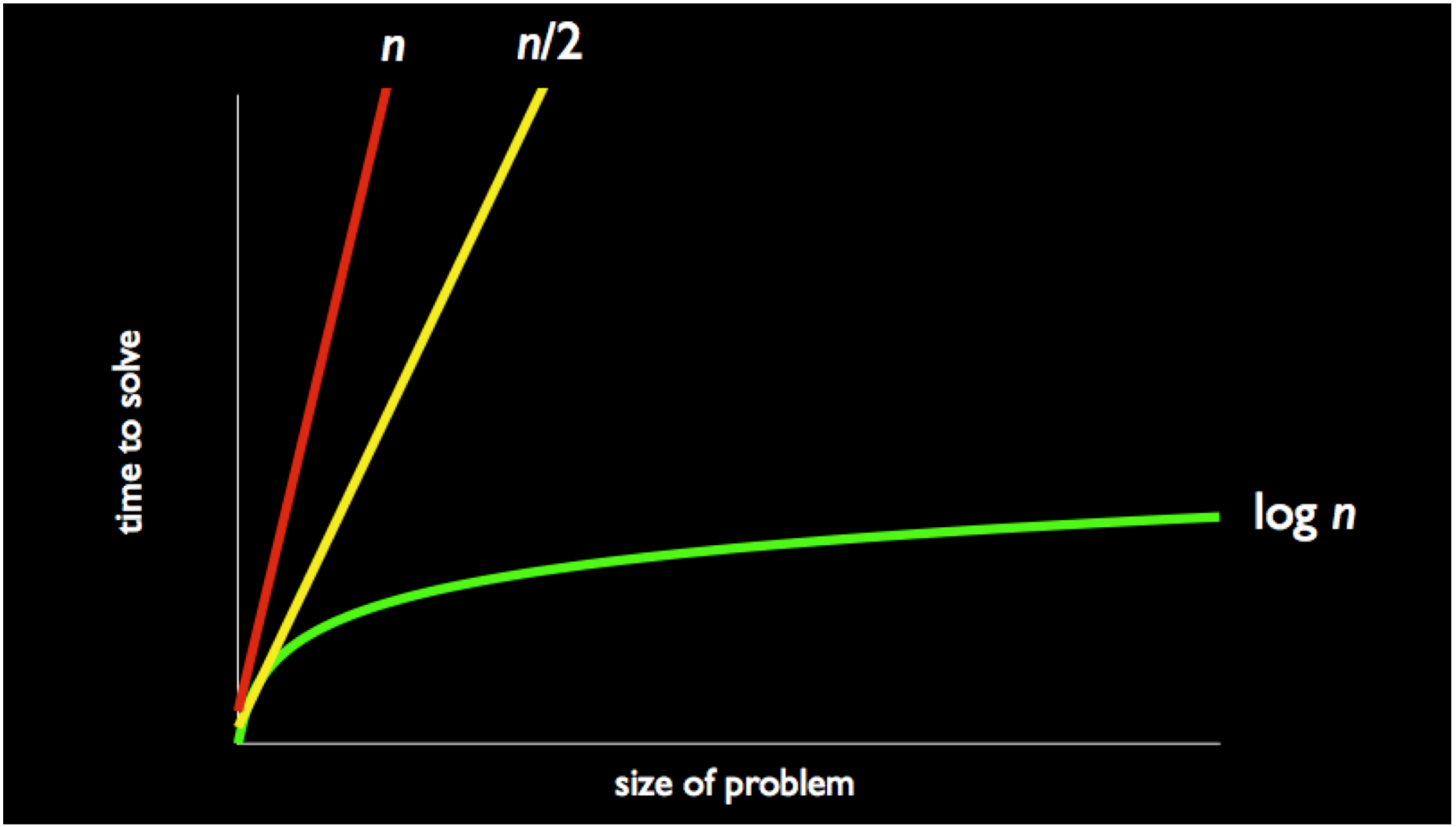
- Repeat until no input to process
  - Map: partition inputs into buckets
  - Reduce: process the inputs in the same bucket

A Map-Reduce Counting Algorithm

- Initialize inputs
  - every student finds a piece of paper, writes 1 on it, and stands up
- Repeat
  - Map:
    - Pairing standing-up students
  - Reduce:
    - For each pair, one student gives his/her number to the other and sits down; The other student updates its number to be the sum and remains standing up

Q: How long does the alg run?

Comparison of Complexity



http://d2o9nyf4hwsci4.cloudfront.net/2013/fall/lectures/0/w/notes0w/notes0w.html

Outline

- What is Computer Science?
- *What is Programming?*
- Administrative trivia's
- Summary

What is Programming?

- **Programming:** Design algorithms and express algorithms precisely in programs

- **Program**: A set of instructions written in a computer language to be carried out by a computer.
  - We will be focusing on computer languages including Java and JS

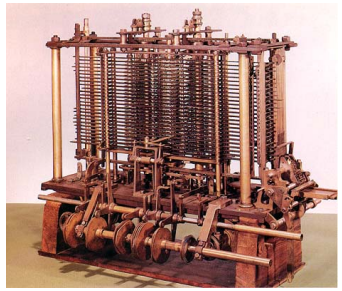
- **Program execution**: The act of carrying out the instructions contained in a program.

Why Programming?

- Avoid programming: using prepackaged software solutions.

- Programming: Enables you to make a computer do much more.

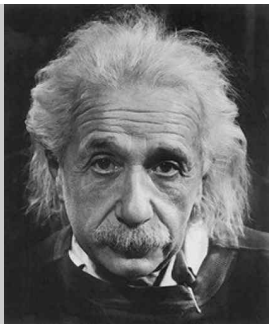
Ada Lovelace



Analytic Engine

Why Programming?

" Computers are incredibly fast, accurate, and stupid; humans are incredibly slow, inaccurate, and brilliant; together they are powerful beyond imagination. " *– Albert Einstein*

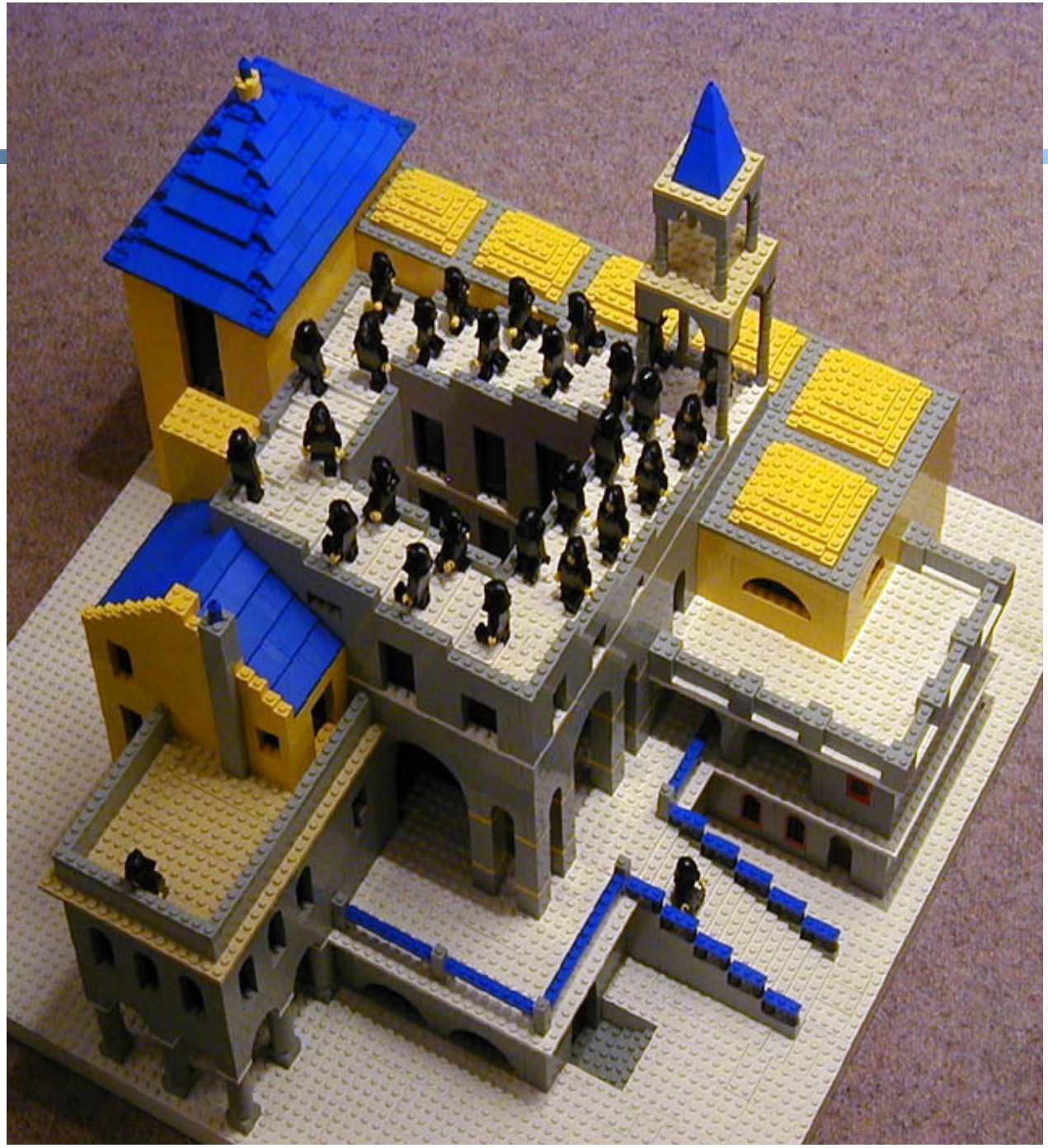


Learning a Programming Language

- Just like learning any new language
  - syntax: "new words"
  - grammar: how to put them together
  - programming: telling a coherent story
  - library: use plots already written

- Initially needs efforts, but pays off in the end !

Programming is like Legos…




BENSON JR.

Programming Languages are Easier than Natural Languages

- Programming languages have simpler structure

- Programming languages do not have ambiguities

Kids Make Nutritious Snacks.
Local High School Dropouts Cut in Half.
Police Squad Helps Dog Bite Victim.
Red Tape Holds Up New Bridge.

[real newspaper headlines, compiled by Rich Pattis]

Comment on Course Eval

- "We only work with java, but honestly the class isn't about learning a language so much as it is learning how to think like a programmer. This class was great for that purpose, indeed, I would argue that such a class is actually what a QR requirement should be -- not merely quantitative reasoning but thinking in terms of a formal system and how to translate complex tasks into simple tasks that can be accomplished with formal statements."

What Kind of Class is This?

- Sort of foreign language (how to communicate with a computer effectively)
- Sort of engineering (building things that realize a desired objective)
- Sort of philosophy (formal logic, deductive systems and semantics, model of world)
- Sort of psychology (given some output behavior, what is going on internally?)

Comment on Course Eval

- "Econ major? History? English? Political Science? It matters not, my friend! You use a computer. Now is the time to take the reigns and truly control that machine that captures your attention for 35% of your day."

What are learned in this course

- Learn the elementary and the way of thinking
  - **Overview** of the CS discipline and the IT field
  - **Ability** to **create** computer applications
    - Not just **use** computer applications
  - **Principles** of computational thinking
    - Logic thinking, algorithmic thinking, systems thinking
    - Network thinking

Topics of this course

Intended mainly as your ***first CS and programming course !***

- program problem solving techniques:
  - how to model real world and manage complexity, e.g.,
    - structural programming: top-down vs. bottom-up vs. stepwise refinement
    - recursive programming
    - object-oriented programming: abstraction and modularization
    - event-driven programming
  - how to solve a problem efficiently, elegantly
  - debugging: figure out why the computer did not do what you wanted
- good programming style, e.g.,
  - robust, efficient program, readable code; documentation
- use important libraries and data structures, e.g.,
  - Multimedia, GUI, Android, Google App Engine, thread
- programming tools
  - Java/Javascript/Web programming language and supporting tools

Outline

- What is Computer Science?
- What is Programming?
- *Administrative trivia's*
- Summary

Course Personnel: Instructor

- Instructor
  - Qiao Xiang, qiaoxiang@xmu.edu.cn
    - office hours: by appointment
  - Qingyu Song, qingyusong@xmu.edu.cn
- Teaching assistant
  - Mengqi Fu, fumengqi0725@gmail.com
  - Ziheng Zhang, ziheng.zhang.xmu@outlook.com

Instructor: Qiao Xiang

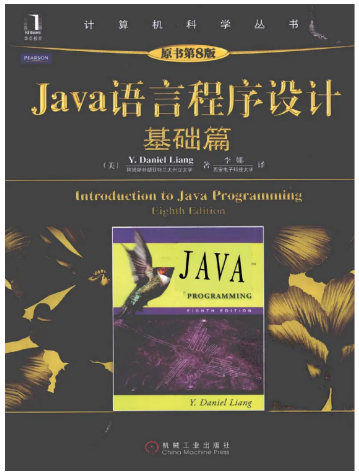
- Joined XMU as a professor in January 2021
- Research: Computer Networks and Systems
- Previously,
  - Research assistant professor, Yale University, US., 2019-2020
  - Postdoctoral fellow, Yale University, US. 2016-2018
  - Postdoctoral fellow, McGill University, Canada, 2014-2015
  - Ph.D. in Computer Science, Wayne State University, US, 2014
  - B.E. in Information Security and B.Econ., NKU, 2007

Instructor: Qingyu Song

- Joined XMU as an assistant professor September 2025
- Research: Computer Network + AI
- Previously,
  - Ph.D. in Computer Science & Engineering, CUHK, 2025
  - Research Intern, Huawei HK, Hong Kong SAR. 2025
  - Visiting Research Intern, Huawei HK, Hong Kong SAR. 2023
  - M.E. in Control Engineering, THU, 2021
  - B.E. in Software Engineering, HITWH, 2018

Textbook Information

- Java语言程序设计(基础篇)



- 计算机科学导论——以计算思维为舟

Other recommended readings listed online

Information, lecture notes, & assignments are available on class server

- we won't use much paper !
- *please read online information at least once every two days !*

Course Workload

- Participant 10%
- A one-hour exams (20%)
  - one exam == 1.5 assignments
- Others (70%) :
  - Assignments: 8 -- 9 in all
    - all assignments are posted online (no paper handout)
    - all assignments must be submitted online
    - due on the date specified in the schedule page
    - grace period policies
  - One or two bonus [optional] assignments to replace lowest grades
  - Final project (== 2 assignments)?

Assignments (pset)

- All assignments are programming assignments

- You learn more from doing the assignments
  - we learn [Dr. William Glasser]
    - 10% of what we read
    - 20% of what we hear
    - 50% of what we read and hear
    - 70% of what we discuss
    - 80% of what we experience personally
    - 95% of what we teach others

http://principalcowart.blogspot.com/2009/02/how-we-learn-william-glasser.html

Assignments (pset)

- Assignments start small and easy, and then turn large and more interesting

- Each assignment intends to
  - introduce one key programming concept
  - demonstrate the application of computer programming to diverse fields (text, graphics, audio, problem solving, scientific computation)

Example Past Assignments (pset): Processing Texts

- Generate (simple) poems
- Madlib
- Hangman

Example Past Assignments (pset): Media

- Recursive graphics (static image)

- Bouncing animation (dynamic image)
- Countdown
- Generating illustration

- Guitar hero (music + visualization)
  - demo/ps8-guitar/

Example Past Assignments (pset): Problem Solving

- Sudoku (ps7)
  - http://mrt-sudokusolver.appspot.com/
- Critter tournament (ps8)

Example Past Assignments (pset): Scientific Computation

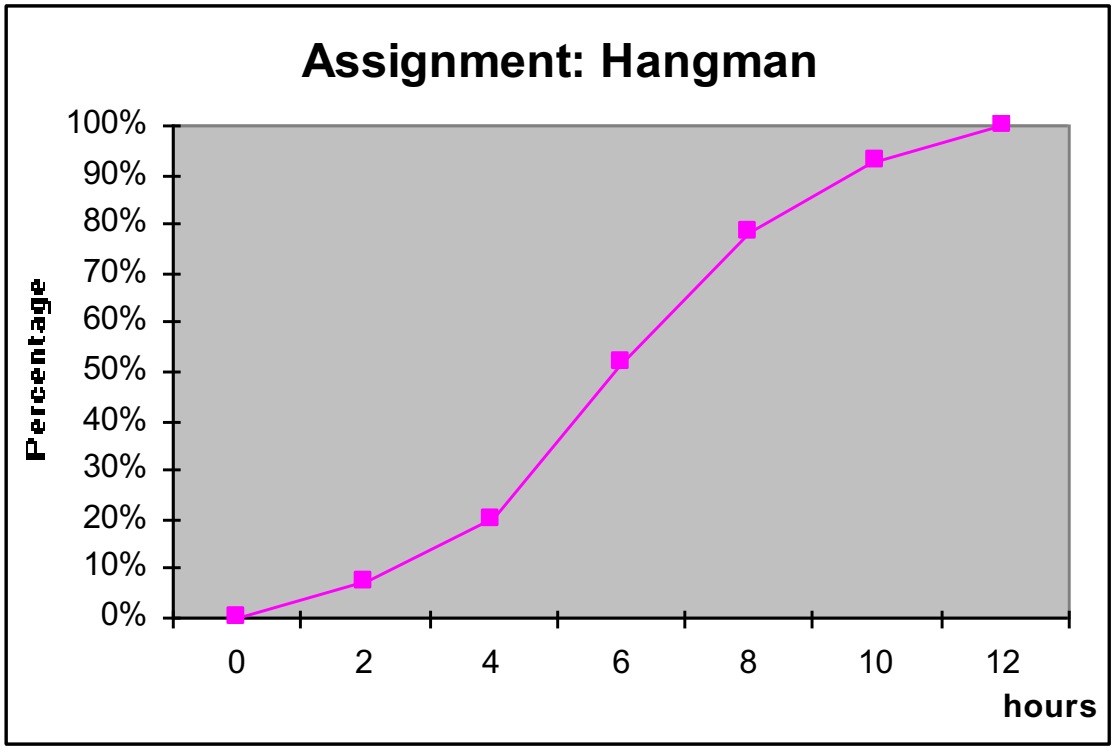
- N-body
  - demo/ps5-nbody/

Assignments this Semester

- We are still finalizing the programming assignments for this semester: expect both overlapping and new assignments

Assignments Workload

- Time spent on assignments reported by students



Comment on Course Eval

- "[The problem sets] often take 6-7 hours to complete (especially a few towards the end of the semester). The final project [...] took me at least 12 hours to get right, but the feeling of accomplishment after making a working program is like none other."

Comment on Course Eval

- "Basically, programming is just problem solving. If you think you're someone who likes to solve problems, then definitely take this class. […] I do have to warn you about the problem sets, though. They start out very easy, but get pretty hard because you have to build on your prior knowledge.

  Debugging your program can be the most annoying thing in the world, since hitting the shift key while typing the equals sign can cost you about 5 hours' worth of time looking for it (personal experience). So start on the problem sets early, and don't hesitate to ask the TA's or help--you'll probably need it at least once, such as on the last problem set."

How to do well in this course

- Keep up with the assignments
  - The course material is cumulative
  - From a former student: "Procrastination will eventually come around to bite you in the ass!"
- If you don't understand something, ask questions (especially "WHY?").
  - "There's no such thing as a dumb question."
  - Computers are neither magical nor mysterious. Everything can be explained!
- Discuss your design with others

Should you Take this Course?

- "I hate computers."
- "I like fuzzy solutions."
  - Programming forces you to solve problem precisely.
- "I refuse to think logically."
- "I want to take an easy class."
  - Hard for those who find difficulty in logical thinking and who don't pay attention to precision.

- Answer: No

Should you Take this Course?

- "I want free gourmet meals and to make lots of money by working for Facebook/Google/LinkedIn/Apple"
- "World of Warcraft rocks hardcore!"
- "Everyone, look at my Facebook farm!"

- Answer: Yes/no

Should you Take this Course?

- "I have to take this class."
- "Computers and robots are going to take over the world. I want to befriend them so that my life will be spared."
- "I know that 65% of us will have to talk to software developers on a regular basis in whatever profession I choose after I graduate, I need to communicate with them."
- "I like to solve problems."
- "I like to learn new perspectives on how to solve problems."
- ...

- Answer: Yes

Questions for You

- Please return the class survey at the end of the class

Questions?