Lecture1: Overview

9/9/2015

Daniel Bauer

Contents

Instructors

- Section 1:
  - Daniel Bauer (bauer@cs.columbia.edu)
  - Office hours: Mon 4:00pm-5:00pm
    7LW3 CEPSR
    (or by appointment)
- Section 2:
  - Larry Stead (lss2168@columbia.edu)
  - Office hours: After class or by appointment

Class Coordination

- The two sections will be mostly synced
- TAs are shared between the sections
    - can go to anybody's office hours or recitation
- Same homework for each section
- One piazza account shared between the sections

Course Overview

- Lectures:
  Section 1: Mon & Wed, 5:40pm - 6:55pm, 301 Pupin
  Section 2: Tue & Thu, 5:40pm - 6:55pm, 614 Schermerhorn
- Course Website:
  http://www.cs.columbia.edu/~bauer/cs3134
- GitHub used for homework, code examples.
- piazza used for questions / discussions / announcements.
- Slides and Gradebook on Courseworks.
- Task for this week: Make sure you can access all resources!

Instructional Assistants

Linan Qiu (lq2137@columbia.edu)
Anna Lawson
Evan Tarrh
Joshua Keough
Nick Mariconda
Ken Aizawa

Kunal Jasty
Ruicong Xie
Harsha Vemuri
Lily Wang
Zeynep Ejder

- IA hours, starting next week: Announced on course website.
- Will probably have three recitation sessions. Office hours (at least) on all other days of the week.

Prerequisites

- **Knowledge of Java** (e.g. W1004 - Introduction to Computer Science and Programming in Java)
- Basic discrete math.
- Have some method for developing Java code - whatever works for you
  - Eclipse, IntelliJ, SublimeText, Emacs, Vim

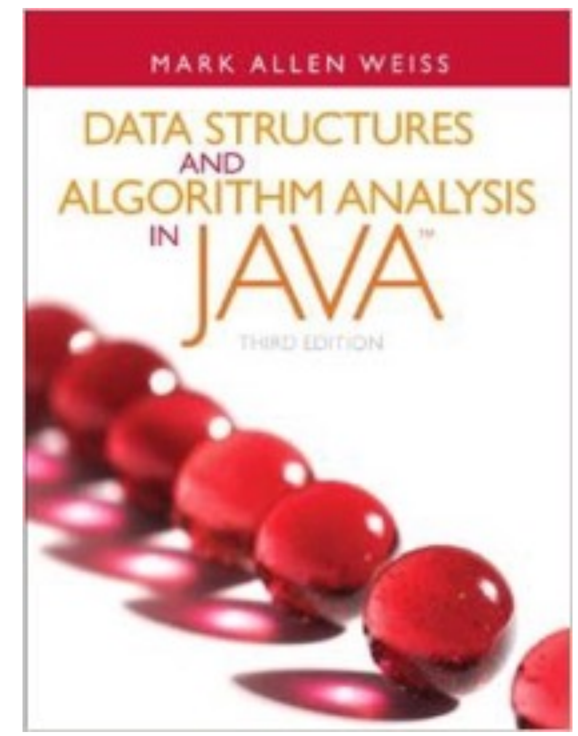
Textbook

- Weiss, Mark Allen (2012).
  Data structures and Algorithm Analysis in Java.
  3rd ed. Prentice Hall.

- ISBN: 9780132576277

- Errata:
  http://users.cis.fiu.edu/~weiss/dsaajava3/errata.html

- Source code:
  http://users.cis.fiu.edu/~weiss/dsaajava3/code/



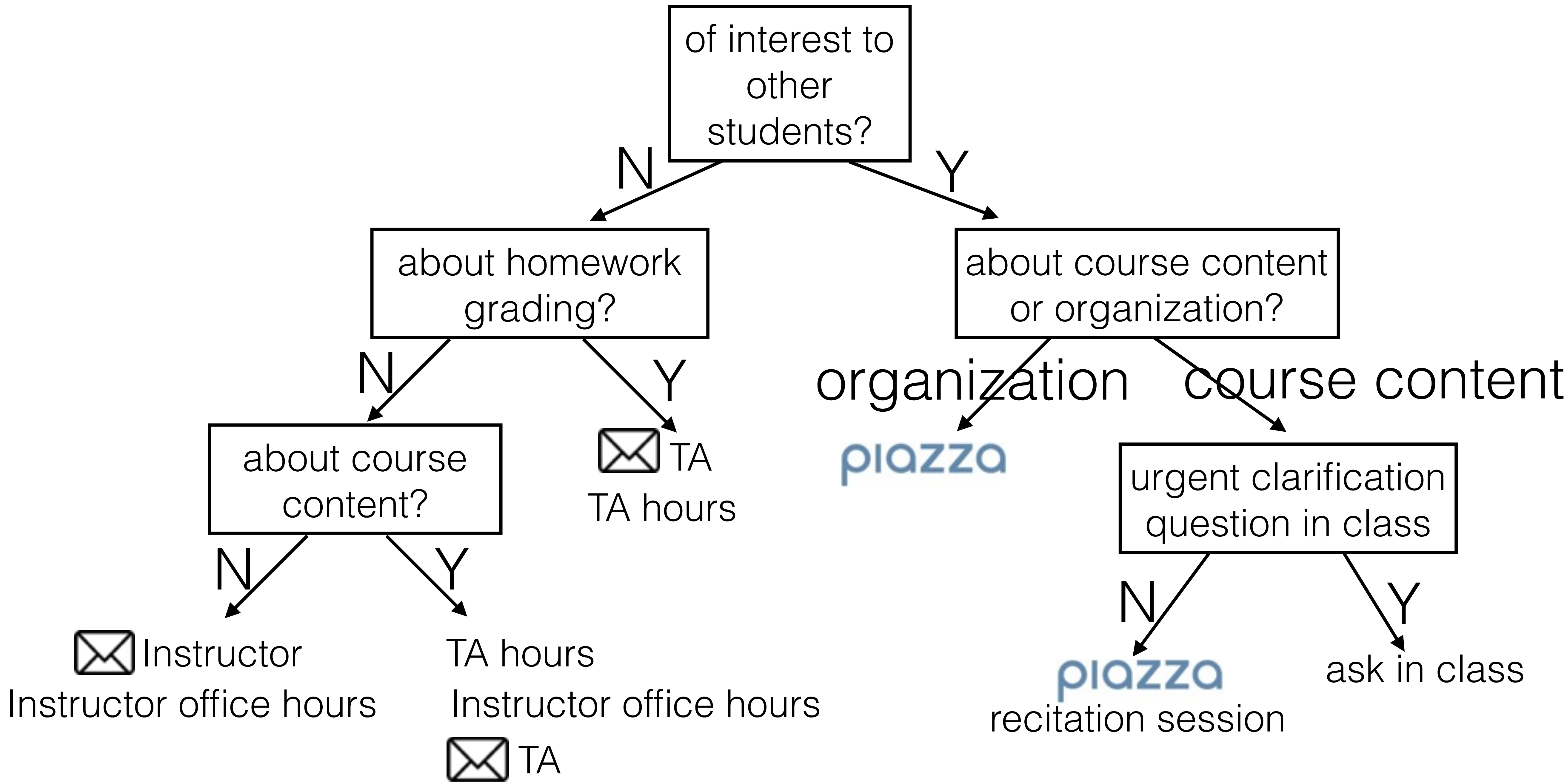
Deliverables and Grading

- 50% - 7 homework assignments, weakest dropped
  + homework 0 (ungraded).
- 20% - In-class midterm (late October).
- 25% - Final exam.
- 5% - Participation
  (class attendance/participation,
  activity on Piazza).
- Grading Range A+ to F

Expectations

- Attend class, participate!
- Do reading assignments.
- Start homework early.
- Get help when you need it and help others on Piazza.
- Make sure you have the registration status you want!
- Academic Honesty - read the statement on course website carefully. You MUST submit original work!

Asking Questions



Contents

1. Organizational Overview
2. **Introduction to Data Structures**
3. Abstract Data Types

Why should I take Data Structures?

- Requirement.
- Core software engineering skills.
- Develop problem solving skills using a “top-down” approach.
- Coding interviews usually focus on data structures and algorithms.

Data Structures

- Data Structures: Ways of representing and organizing data **so that they can be used efficiently.**
- Tasks: organize, search, filter, update, add, delete, combine

Digital Data

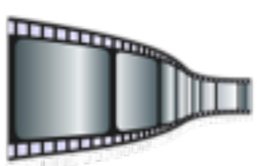

Audio


Social
Networks

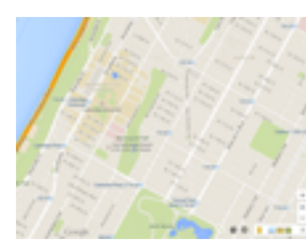

Tables

Text


Images


Video

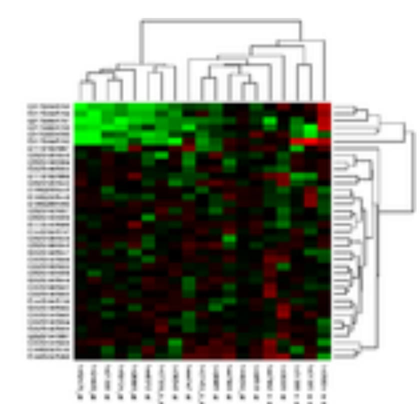

Bio data

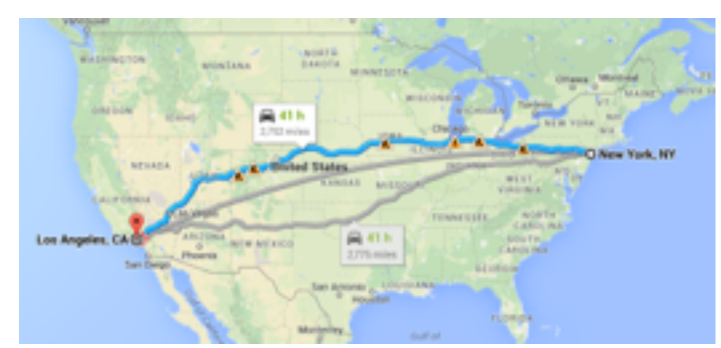

Maps

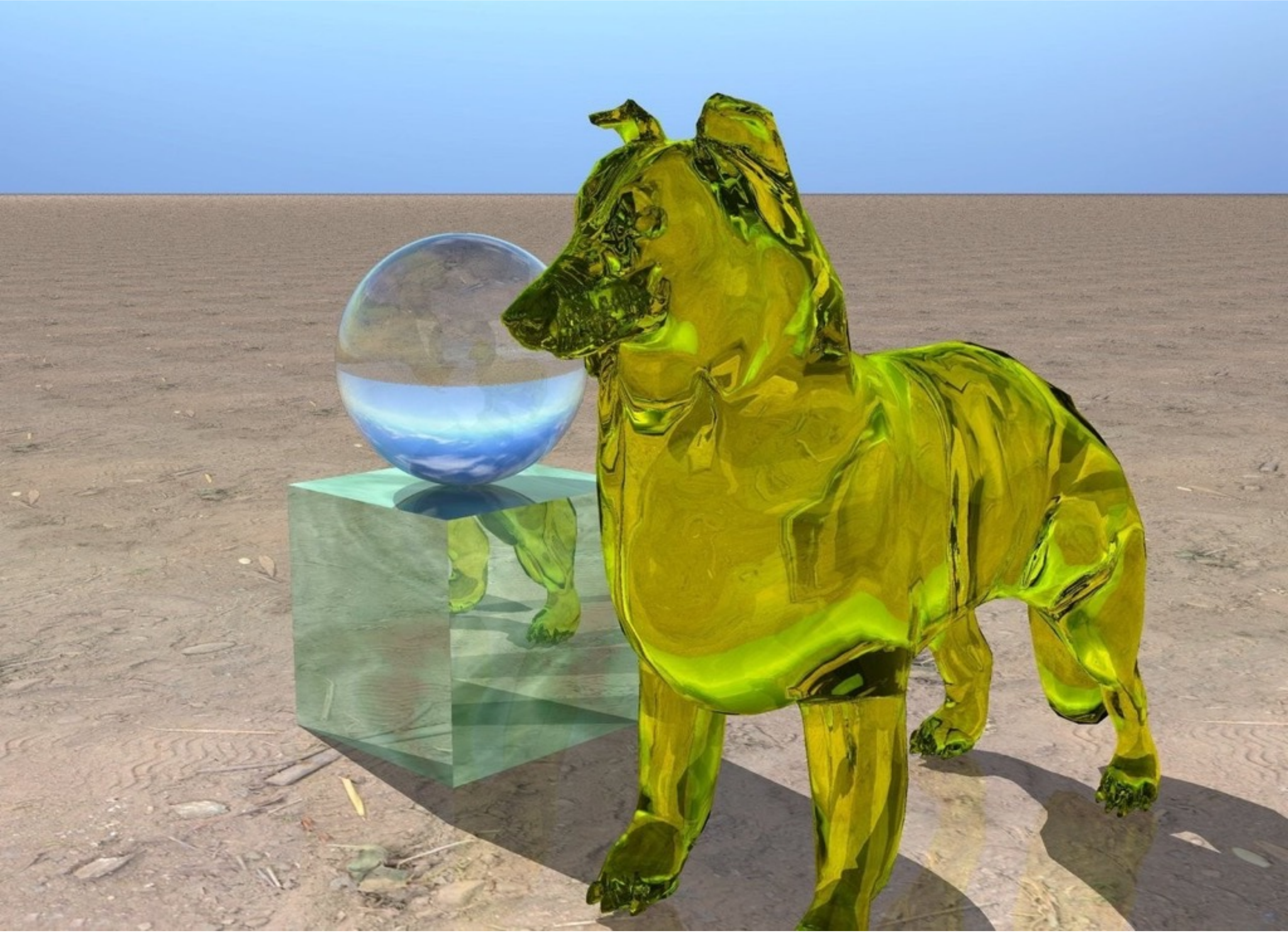
...

Example Applications

- Document Retrieval
- Machine Learning
- Microarray Data Analysis
- Route Planning
- Computer Graphics

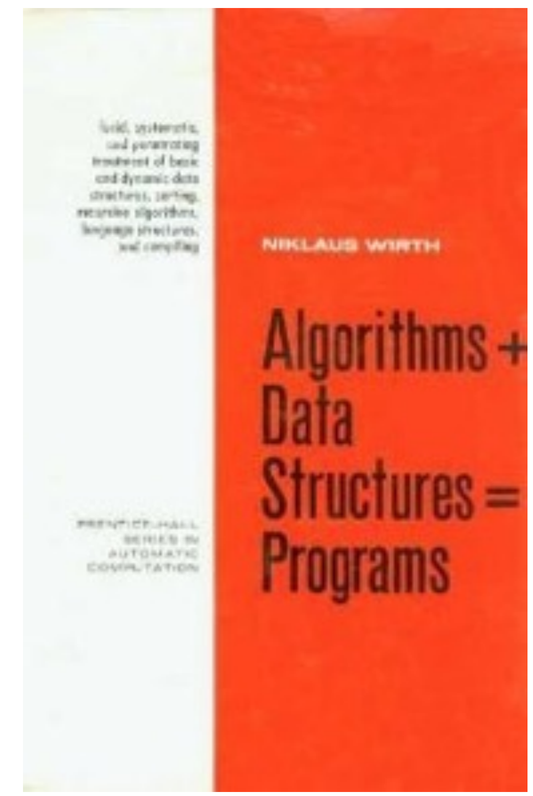




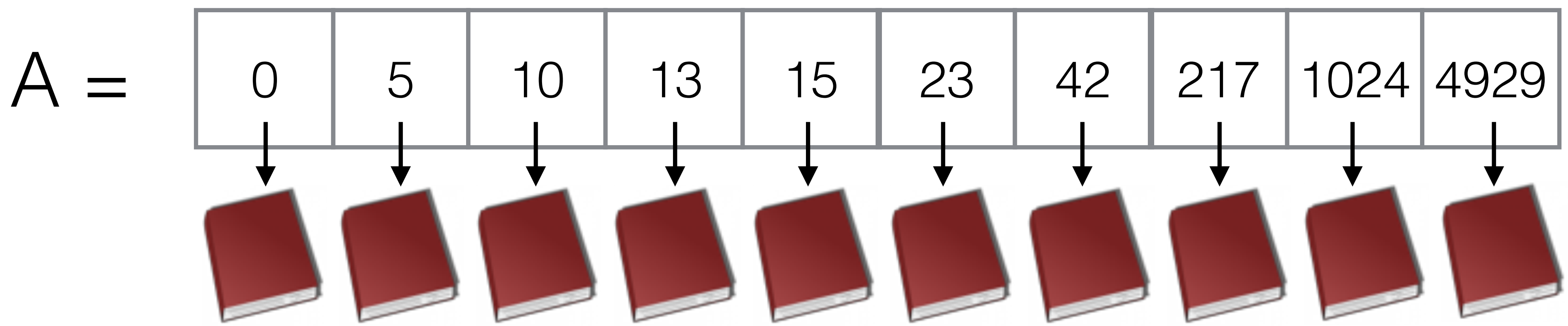


Algorithms + Data Structures = Programs

- Niklaus Wirth, 1976 (inventor of Pascal)
- Problem solving requires:
  1. Creating the right data model for thinking about a problem.
  2. Devising the appropriate methods to solve the problem.



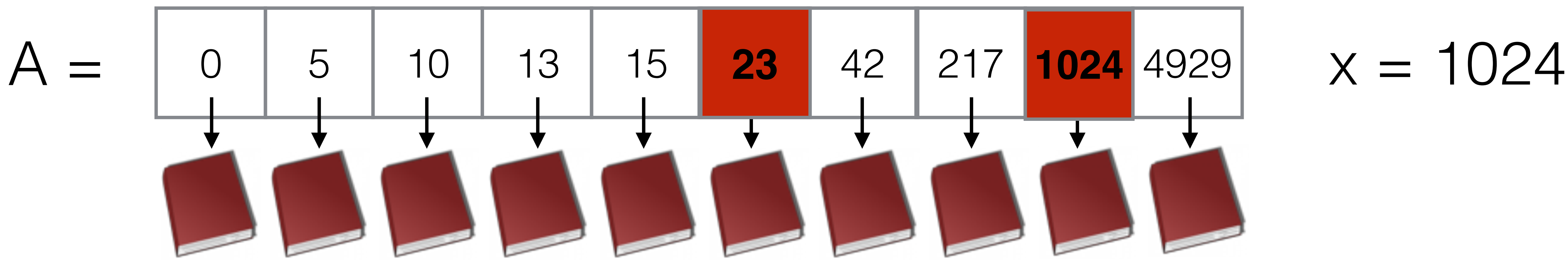
Example: Finding an Entry *x* in a Sorted List



- Approach 1: Linear search. Start at position 0, scan list until we reach the correct entry.

- In the worst case, we need length(A) steps to find the entry!

Example: Finding an Entry *x* in a Sorted List



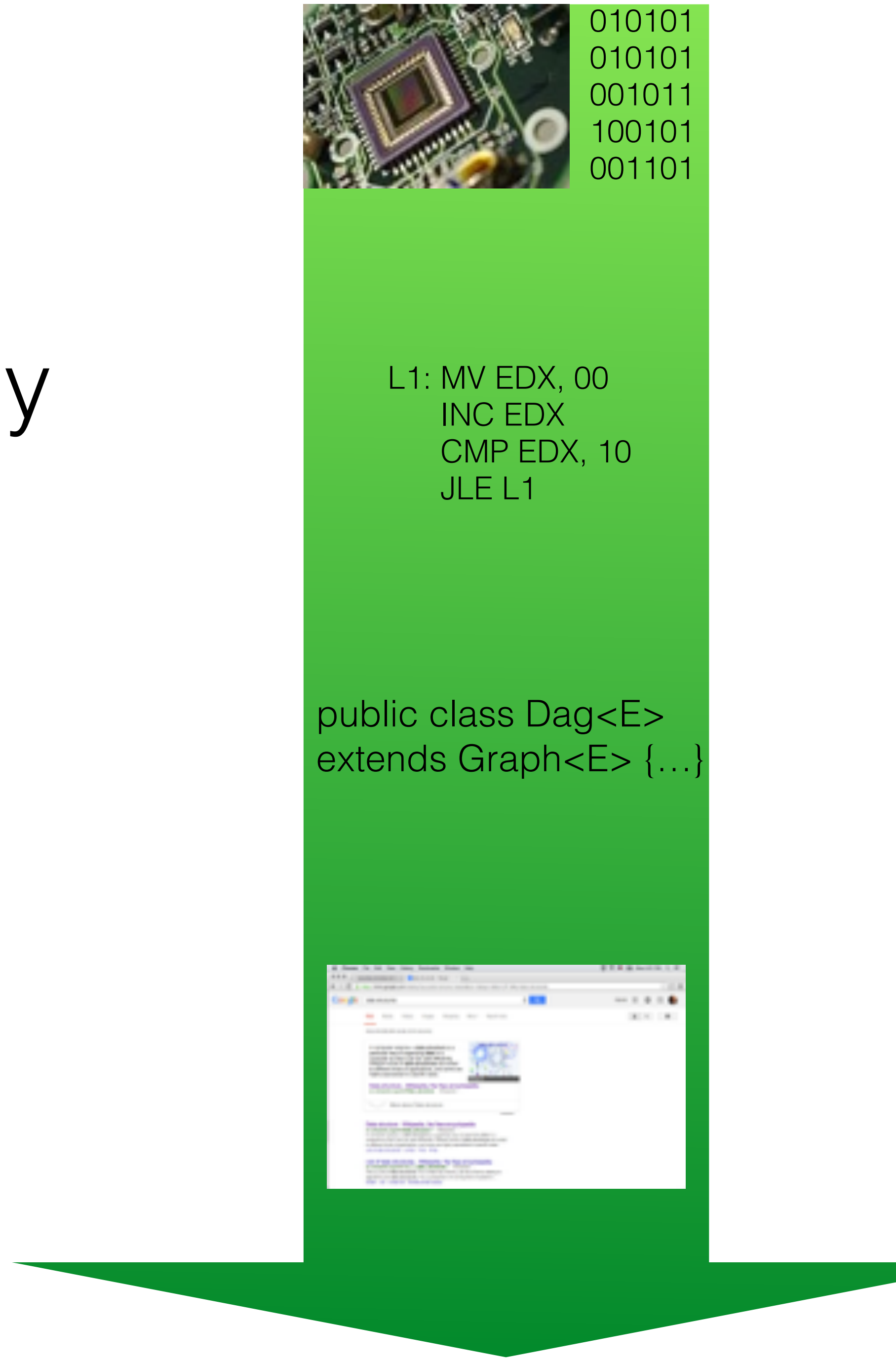
- Approach 2: **Use the property that the list is sorted.**
  - Find entry y in the middle if A: y = A[A.length/2]
  - if (y == x) we found the entry.
  - if (y < x) continue search on second half of A.
  - if (y > x) continue search on first half of A.
- In the worst case we need $\log_2(\text{length}(A))$ steps.

Big Data

- Ever faster hardware available, more memory.
  - Amazon cloud has machines with 244Gigs of main memory!
- Can now store and process huge data sets.
- Ironically, algorithmic efficiency matters even more!

Levels of Abstraction

- Hardware, bit representations
- Assembly language, registers, memory abstraction
- *Higher-level languages (Java, C++, Python, …)*
- Applications, User interfaces



Contents

Data Types

- Basic data types: booleans, bytes, integers, floats, characters…
- Simple abstractions: `Array`, `String`
- More complex data types (**this class**): Lists, Stacks, Trees, Sets, Graphs

Abstract Data Types

- An Abstract Data Type (ADT) is an collection of data together with a set of operations.

- ADT specification *does not mention how* operations are implemented.

- Example:

  - `Set` ADT might provide “add”, “remove”, “contains”, “union”, and “intersection” operations.

ADTs vs. Data Structures

- An abstract data type is a well-defined collection of data with a well-defined set of operations on it.
- A data structure is an actual implementation of a particular abstract data type.

ADTs in Java

- ADTs can be specified as interfaces. Interfaces define behavior, but say nothing about implementation or performance issues

- ADTs can be implemented as classes. Careful class design hides implementation details from users, enabling “plug and play”.

    - Encourage re-usability of components!

ADTs in Java

- Example: Java `Strings`
  - We can call methods such as `length()` and `concat(String str)`.
  - We don't have to know how Strings are stored in memory or how methods are implemented.
  - How many bytes does it take to represent a character?

Some ADTs we will study

- Lists
- Stacks
- Queues
- Priority Queues (Heaps)
- Search Trees
- Graphs