Lecture 0
Data Structures and Algorithms

Welcome!

This course is about *fundamental data structures and algorithms for organizing and processing information*

- "Classic" data structures / algorithms and how to analyze rigorously their efficiency and when to use them
- Queues, dictionaries, graphs, sorting, etc.

Course materials

- All lecture and section materials will be posted
  – They are visual aids although not a complete description
- Textbook: Cormen, Leiserson, Rivest, Stein
  – Good read, but quite detailled and complete
- A good Java reference of your choosing
  – Google is only so helpful

What is this course about

- Deeply understand the basic structures used in all software
  - Understand the data structures and their trade-offs
  - Rigorously analyze the algorithms that use them
  - Learn how to pick "the right thing for the job"
  - More thorough and rigorous take on topics introduced in just a programming course

- Practice design, analysis, and implementation
  - The elegant interplay of "theory" and "engineering" at the core of computer science

- More programming experience (as a way to learn)

Goals

- Be able to make good design choices as a developer, project manager, etc.
  - Reason in terms of the general abstractions that come up in all non-trivial software (and many non-software) systems
- Be able to justify and communicate your design decisions
  - Key abstractions used almost every day in just about anything related to computing and software
  - It is a vocabulary you are likely to internalize permanently

Assumed Background

- Fundamentals of computer science and object oriented programming
  – Variables, conditionals, loops, methods, fundamentals of defining classes and inheritance, arrays, single linked lists, simple binary trees, recursion, some sorting and searching algorithms, basic algorithm analysis

Topics Outline

- Introduction to Algorithm Analysis
- Lists, Stacks, Queues
- Trees, Dictionaries
- Binary Search Trees, AVL Tress
- Heaps, Priority Queues
- Sorting
- Disjoint Sets
- Graph, Topological Sorting
- Minimum Spanning Tree
- Shortest Paths

Terminology

- Abstract Data Type (ADT)
  – Mathematical description of a "thing" with set of operations
- Algorithm
  – A high level, language-independent description of a step-by-step process
- Data structure
  – A specific organization of data and family of algorithms for implementing an ADT
- Implementation of a data structure
  – A specific implementation in a specific language

Data structures

(Often highly *non-obvious*) ways to organize information to enable *efficient* computation over that information

A data structure supports certain *operations*, each with a:

- Meaning: what does the operation do/return
- Performance: how efficient is the operation

Examples:

- ***List*** with operations **`insert`** and **`delete`**
- ***Stack*** with operations **`push`** and **`pop`**

Trade-offs

A data structure strives to provide many useful, efficient operations

But there are unavoidable trade-offs:

- Time vs. space
- One operation more efficient if another less efficient
- Generality vs. simplicity vs. performance

We ask ourselves questions like:

- Does this support the operations I need efficiently?
- Will it be easy to use, implement, and debug?
- What assumptions am I making about how my software will be used? (E.g., more lookups or more inserts?)

ADT vs. Data Structure vs. Implementation

"Real life" Example (not perfect)

ADT: Automobile

- Operations: Accelerate, decelerate, etc...

Data Structure: Type of automobile

- Car, Motorcycle, Truck, etc...

Implementation (of Car):

- 2009 Honda Civic, 2001 Subaru Outback, ...

Example: Stacks

- The ***Stack*** ADT supports operations:
  - **`isEmpty`**: have there been same number of pops as pushes
  - **`push`**: takes an item
  - **`pop`**: raises an error if empty, else returns most-recently pushed item not yet returned by a pop
  - … (possibly more operations)

- A Stack data structure could use a linked-list or an array or something else, and associated algorithms for the operations

- Many implementations are provided in Java