The Art of Data Structures

Introduction

Alan Beadle
CSC 162: The Art of Data Structures

Acknowledgment:
Slides and course materials originally prepared by Richard E Sarkis for the previous offering of this course

UNIVERSITY of ROCHESTER

Agenda

- To review the ideas of computer science, programming, and problem-solving
- To understand abstraction and the role it plays in the problem-solving process
- To understand and implement the notion of an abstract data type

Class Overview

Course Instructor

- Alan Beadle (he/him/his)
- hbeadle@cs.rochester.edu
- PhD student in computer science
- Contact me via email, can meet over zoom

Class Organization

- Class content provided mostly through a class website, and the rest through Blackboard
- Course components are:
  - **Lectures**
  - **Lab Assignments**
  - **Workshops**

Lectures

- **See class website for schedule…**
- Course will include
  - Lecture slides (i.e. my lecture notes)
  - Code samples and some live coding
  - Discussions from the class
    - Class participation is *vital*
    - We are a small class (v.s. 150+ people)

Lectures

- Per current University policy, online lectures will be recorded and uploaded.
- Please email me after class if you have concerns about this.

Syllabus, schedule, etc

(On website)

What is Computer Science?

Why Computer Science?

- Everything is bits and computation!
  - Astronomy - simulation and observation data
  - Biology – Protein folding
  - Music – Sound processing
  - Mechanical Engineering – Load bearing
  - Literature – Text analysis
- What are you studying?

Lies, damned lies...

- Common misconceptions about Computer Science:
  - *”Computer science is the study of computers”*
  - *”Computer science is the study of how to write computer programs”*
  - *”Computer science is the study of the uses and applications of computers and software”*

“The possibility of a science in which all the world is thought of computationally casts the study of computers in an important new light. As its practitioners are fond of saying, computer science is not about computers, any more than astronomy is about telescopes, or biology about microscopes. These devices are tools for observing worlds otherwise inaccessible. The computer is a tool for exploring the world of complex processes, whether they involve cells, stars, or the human mind.”

1986, Machinery of the Mind: Inside the New Science of Artificial Intelligence by George Johnson, Chapter 4: The Art of Programming, Quote Page 81 and 82, Times Books: A Division of Random House Inc., New York.

Computer science is…

- …the study of computation.
- Feasibility, structure, expression, and mechanization of the methodical processes (or algorithms) that underlie the acquisition, representation, processing, storage, communication of, and access to information

What a tool

The computer is just a tool, described and realized by the concepts computer science

Why Programming?

“The art of programming is the art of organizing complexity, of mastering multitude and avoiding its bastard chaos as effectively as possible.”

Edsger W. Dijkstra (1970) “Notes On Structured Programming” (EWD249), Section 3 (“On The Reliability of Mechanisms”), p. 7.

Why Programming?

- Using off-the-shelf applications will move you a along most days, e.g.:
  - Spreadsheets, Photo editors, word processors
- However, if you are doing novel, creative work, sooner or later, you will need to solve a *new* problem
- Learning good principles of programming will make it possible with more fun and less frustration

Programming

- Take a computing problem and create an executable program

1. Analyze and understand the problem
2. Design a data model and an algorithm
   - Data model: way to represent the information
   - Algorithm: formal computation steps
3. Implement the algorithm in a computer language

Programming

- Programming != solving a problem
- Programming is teaching somebody else how to solve a problem
  - “Somebody else” is a computer
  - A computer is a device that can be programmed to carry out a finite set of operations on binary numbers
  - Computers can’t do anything that they are not programmed to do

Why Python?

Why Python?

- *It's powerful, accessible and used in many fields*

Why Python?

- Python is a language of choice for:
  - Scientists (NASA)
  - Web site builders (Reddit)
  - Researchers in the humanities (Somebody...)
  - Application developers (Dropbox)

Why Python?

- We'll be using Version 3.7 or later

- http://www.python.org

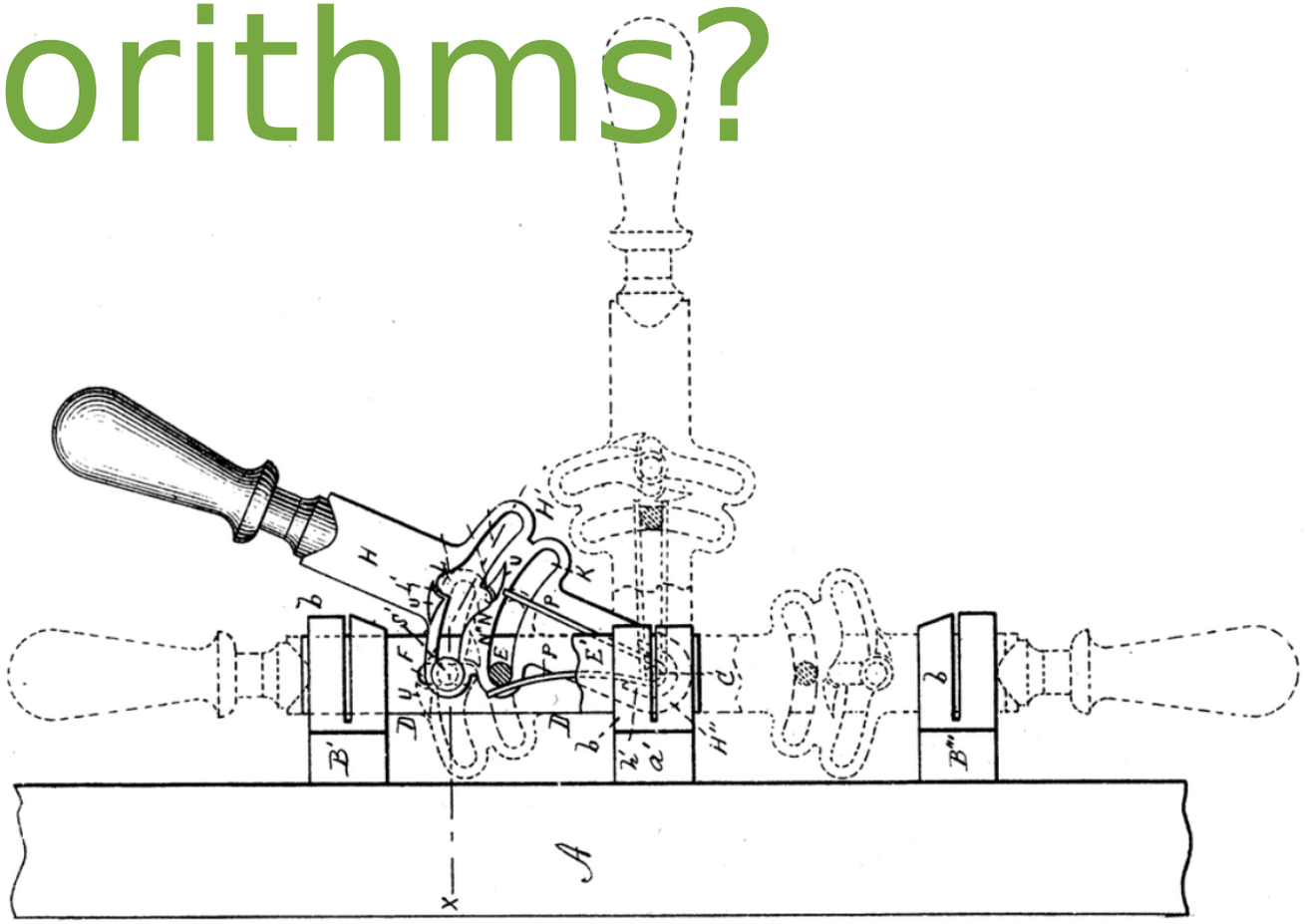
Why Python?

- Python can be used in a variety of contexts, we'll be using *Jupyter* notebooks
  - https://jupyter.org/
- This class will use *Jupyter* notebooks exclusively!

Why Python?

- Python installations can be complex with the large number of third-party packages (*Jupyter*, *NumPy*, *matplotlib*, etc.)
- *Anaconda* is a great re-packaging of Python and a variety of packages

Why Data Structures and Algorithms?



Why Data Structures and Algorithms?

- Our goal is to write *efficient*, and *correct* programs
- Write computer programs in a language, e.g. Python
- Develop mathematical maturity to understand analytical proofs
- Implement good abstractions

Why Data Structures and Algorithms?

Imagine it’s 1943. You’re a law clerk, charged with organizing the records of a newly created law firm.

How do you expect people to use the files? What sorts of questions will they want to answer — what sorts of things will they want to look up?

Why Data Structures and Algorithms?

- Records of a particular case, by:
- Date
- Plaintif
- Defendant
- client name Court docket #

- All cases with given:
- Plaintiff
- Defendant
- Judge
- Point of law
- Size of award

Why Data Structures and Algorithms?

- What other indices would you keep?
- How hard would it be to find things for which you don't have an index?
- What shortcuts might be possible?

Why Data Structures and Algorithms?

- The list of operations you want the files to support is analogous to an **Abstract Data Type (ADT)**
- The physical organization is analogous to a **Data Structure (DS)**

Why Data Structures and Algorithms?

- It's *very important* to distinguish between these two concepts
- We will look at three main things this semester: ADTs, DSes, and algorithms

Why Data Structures and Algorithms?

- An ADT represents a particular set of behaviors
- You can formally define, perhaps using mathematical logic what an ADT is/does
- For example, a *Stack* is a list implements a LIFO policy on additions/deletions

Why Data Structures and Algorithms?

- A DS is more concrete
- Typically, it is a technique or strategy for implementing an ADT
- Use a linked list or an array to implement a stack class

Why Data Structures and Algorithms?

Some common ADTs that all trained programmers know about:

- stack
- queue
- priority queue
- dictionary
- sequence
- set

Why Data Structures and Algorithms?

Some common DSs used to implement those ADTs:

- Array
- Linked list
- Hash table (open, closed, circular
- Trees (binary search trees, heaps, AVL trees, 2-3, tries, red/black trees, B-trees)

Conclusion

Topics covered...

- This course will cover a variety of intermediate topics:
- Analytical Process
- Standard (Pythonic) Techniques

Conclusion

Topics covered…

- Data Structures
  - Lists, Stacks, Queues, Trees, Heaps, Hashes, Graphs, Sets
- Algorithms
  - Sorting, Greedy, Backtracking, Randomized, Dynamic Programming

Questions?