ECE430.217 Data Structures

1.A - Introduction

Byoungyoung Lee

https://compsec.snu.ac.kr

byoungyoung@snu.ac.kr

Lecture slides were prepared based on Douglas W. Harder's notes (**dwharder@alumni.uwaterloo.ca).**

About Instructor: Byoungyoung Lee

- Research Area: Hacking, Systems Security, Software Security
  - Microsoft Research, Research Intern (2012)
  - Google, Software Engineering Intern (2014)
  - Purdue University, Assistant Professor (2016-2018)
  - Seoul National University, Assistant/Associate Professor (2018-Current)

- Three times DEFCON CTF Finalist (2007,2009, and 2011)
- Internet Defense Prize by Facebook and USENIX (2015)
- DARPA Cyber Grand Challenge (CGC) Finalist (2016)
- Google ASPIRE Awards (2019)
- Found 100++ vulnerabilities from Windows kernel, Linux kernel, Chrome, Firefox, etc.

Data Structures

In this course, we will look at:

- ***Data structures*** for efficiently storing, accessing, and modifying data
- *Some* ***Algorithms*** for solving problems efficiently

We will see that all data structures have trade-offs

- There is no *ultimately good* data structure...
- The choice depends on your requirements

Good Data Structures? Bad?

- Consider accessing the k^{th} entry in an array or linked list
  - In an array, we can access it using an index `array[k]`
    - `Fast`
  - In a linked list, we must step through the first $k-1$ nodes
    - Slow

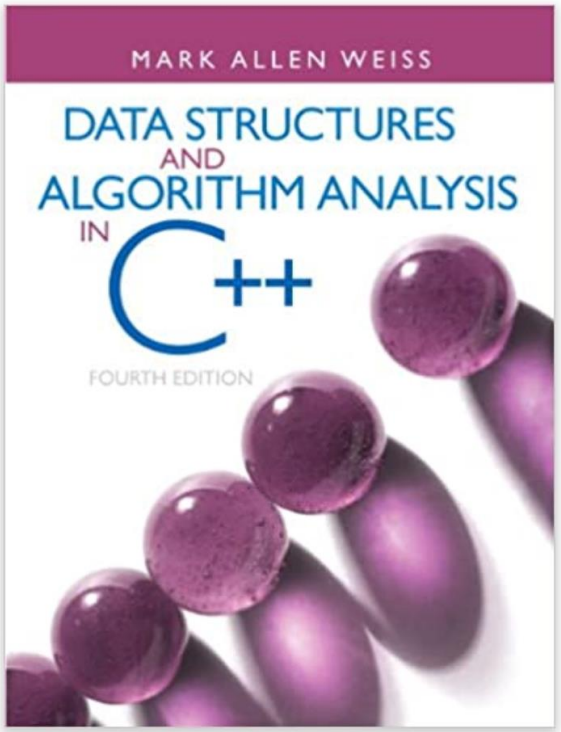
- Consider searching for an entry in a sorted array or linked list
  - In a sorted array, we use a fast binary search
    - Very fast
  - In a linked list, we must step through all entries less than the entry we're looking for
    - Slow

People

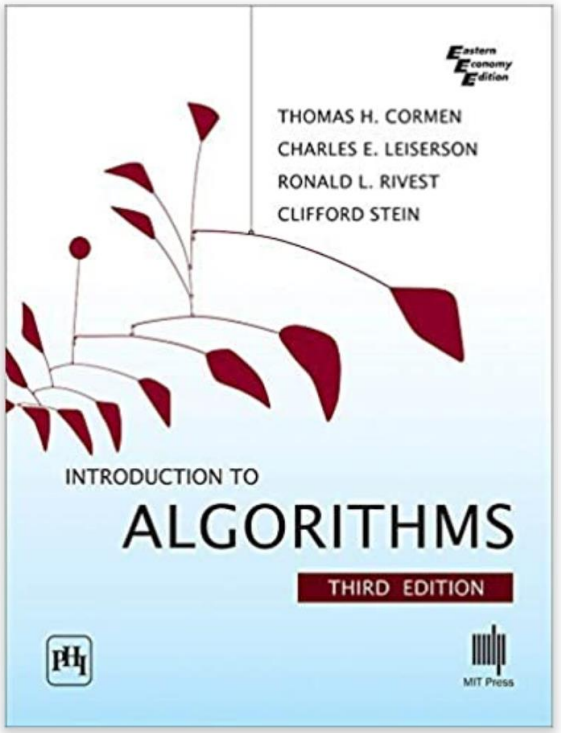
- Instructor
  - Byoungyoung Lee (이병영)
  - byoungyoung@snu.ac.kr
  - Systems security, software security

- TAs
  - Sangyoon Kim (김상윤): ksu880@snu.ac.kr
  - Byeongwook Kim (김병욱): rlaquddnr904@snu.ac.kr
  - Hyokyung Kim (김효경): hyokyung0808@gmail.com

Textbook



Primary



Reference

Topics to be covered

- The course is divided into numerous topics
  - Basics
    - C++
    - Algorithms
    - Time complexity and space analysis
  - List/Stack/Queue
  - Tree
  - Hash
  - Priority Queue
  - Sorting
  - Graph

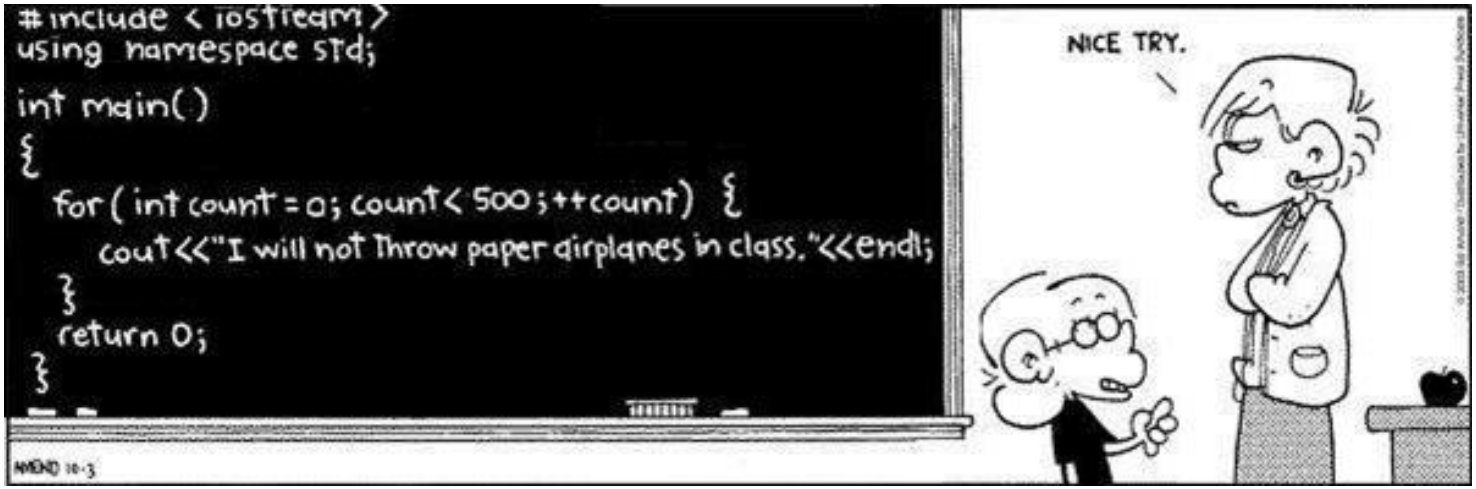
  - Assignments
    - C++
    - Linux

Evaluation

- Your evaluation in this course is based on following components:
  - Quiz + written homework: 10%
  - Midterm exam: 20%
  - Final exam: 30%
  - Programming Assignments (7~9): 40%

C++

- You will be using the C++ programming language in this course



Modified for C++ from http://www.foxtrot.com/

C++

- This course does not teach C++ programming
  - You will use C++ to demonstrate your knowledge in this course
- There will be summary lectures for C++
  - If you want to learn more, please read through on-line materials

C++

- Again, this class assumes you are familiar with C++

- Other sources of help in C++ are:
  - TAs
  - The instructor
  - Other online tutorials: **`http://www.cplusplus.com/`**

Linux

- You will be exposed to the Linux environment
  - We will help you get familiar with Linux
  - Will be providing a short tutorial on Docker or Vagrant
    - Docker provides a containerized environment to build/run a program
    - Vagrant provides a virtualized environment to build/run a program

Academic Integrity

- If you happen to do
  - **Cheating**
  - **Plagiarism**

- You will **very likely** get **F**
  - And your mis-conduct will be reported to the student council
  - Do NOT ask for an excuse: graduation, scholarship, etc.

Plagiarism

- All projects must be done individually:
  - You may not copy code directly from any other source
  - Plagiarism detection software will be used on all of the projects
  - If you viewed another code (from books or lecture notes), you must include a reference in your project
    - Leave a comment!
  - You should not share code with any other students by transmitting completed functions to your peers
    - This restriction includes—but is not limited to—electronic and hard-copy sharing
    - Both students (who showed or copied the code) will get the same penalty
  - You may discuss projects together and help another student debug his or her code; however, you cannot give the exact solution

Plagiarism

- Collaboration with other students must be limited to
  - Discussions
  - High-level pseudocode
  - Assistance with debugging (only through the offering of advice)

- All such collaborations **must** be documented in your source code

Plagiarism

- The best way to avoid plagiarism is:
  - review the C++ tutorial
  - read the project as soon as it is available
  - start the project so that there is sufficient time to contact the T.A. or myself if you have difficulty
  - do not give your code to anyone

Summary

- In this topic, we have:
  - Outlined the course
  - Discussed C++
  - Linux
  - Academic Integrity