EECS 281
Data Structures and Algorithms

Dr. Raed Almomani
Mr. Marcus Darden
Dr. David Paoletti

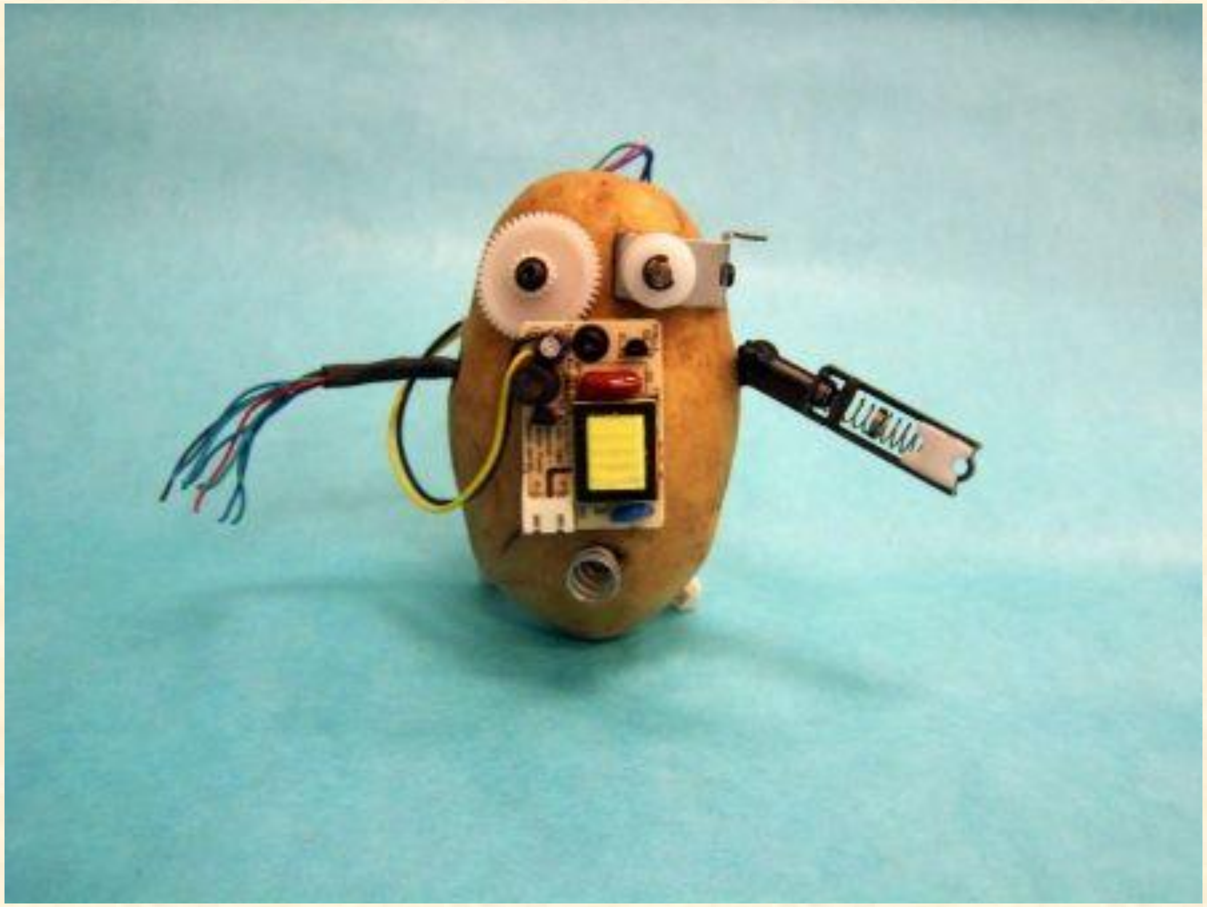
Fall 2018

Other Staff

Teaching Assistants

- Aman Goel (GSI)
- Yijun Hou (GSI)
- Leonard Lin (GSI)
- Saif Rahman (GSI)
- Andrew Zhou
- Austin Dudas
- Bing Schafer
- Chelsie Eiden
- Colten Williams
- Daniel Cayo
- Eric Winsor
- Fee Christoph
- Isabelle Wong
- Jacob Hage
- Joseph Nwabueze
- Karthik Bhandarkar
- Milo Hartsoe
- Noah Ostrowski
- Oliver Hill
- Paul Gossman
- Stephen Satarino
- Sukang Kim
- Susanne Sheng
- Wenyi Wu
- William Wendorf

Course Staff


Potatobot (Piazza)

Course Weekly Schedule

Lectures

- Tuesday & Thursday
- 9am-10:30am, 1670 BBB
- 10:30am-12pm, Chrys Aud
- 12pm-1:30pm, 1610 IOE
- 1:30pm-3pm, 1610 IOE
- 5pm-6:30pm, 2505 GGBL
- Important announcements in lectures

Labs

- See the course Google Calendar on Canvas
- All lectures and labs cover the same material
  - You can attend any, provided there is space

Syllabus

- Please read the syllabus on Canvas

Required Text Books

- *Introduction to Algorithms, 3rd Ed.*
  - MIT Press
  - Cormen, Leiserson, Rivest, and Stein
  - Online version available
- *The C++ Standard Library 2nd Ed, Tutorial and Reference*
  - Addison Wesley
  - Josuttis
  - Online version available

Office Hours

- Student Office Hours Etiquette
  - Come prepared with specific questions
    - Conceptual is fine
    - If code-specific, please have input that your program does work with, and input that it does not
  - Sign up at http://eecs.help
  - Please respect other students
    - Ask one question, then move to back of line
    - Can listen to other student’s questions, as long as not personal in nature
    - If you hear someone that has the same issue that you already solved, feel free to tell them, **in general**, about the problem and solution!

Office Hours

- Staff Office Hours Etiquette
  - Will be posted on the Google Calendar by the end of the first week
  - Please respect course staff availability, as TA’s are students too
- Professor Office Hours Etiquette
  - Always available during scheduled office hours
  - Sometimes available for quick questions (1-2 min) when office door is open
  - Can schedule time outside of posted OH for personal matters
  - Not available when office door is closed
  - Not available during undergrad advising hours

Grading

- Grading Policy
  - 20% Labs (10)
  - 40% Projects (4)
  - 20% Midterm Exam
  - 20% Final Exam

Policy on Deadlines

- Autograder: 2 Late Days per semester
- Use them as you want
- Project 0 late days are “free”, use them for practice!
  - Before any “real” assignment is due, everyone will be reset to 2 late days remaining
- For example: a Project was due Tuesday, today is Thursday; you didn’t submit yesterday = 2 late days to submit today (submitting 2 days late)

Labs (20%)

- 10 lab assignments
- **Can work with a partner**
- Prelab work due at start of lab section
- Submit paper, electronically via Canvas, and/or autograder machine
- Late submissions: must contact us via eecs281admin@umich.edu **BEFORE THE DUE DATE** and provide documentation (extreme medical/personal)

Lab Partners

- No need to “register” partnerships
- Both students must submit all parts individually to receive points!
  - The in-class written problem is done individually!
    - You CANNOT submit for your partner
  - This is practice for the exams
  - It is graded by effort

Projects (40%)

- 4 projects
- **Individual work**
- Submitted electronically to autograder
  - Details to follow
- Approximately 2 weeks per project
- Late submissions: USE LATE DAYS WISELY (see “Policy on Deadlines”)

Projects (40%)

- C++ (International C++17 Standard)
  - https://en.wikipedia.org/wiki/C++17
- CAEN Linux Computing Environment
  - g++ (GCC) 6.2.0
- Beware if you are doing development in any other environment
  - May compile/run perfectly for you, then not even compile on the autograder

Exams (40%)

- Midterm Exam (20%)
  - Wednesday 24 October, 6:30 - 8pm (sharp)
- Final Exam (20%)
  - Monday 17 December, 8 – 10am (sharp)
- Exams will have both a multiple choice section (must bring a number 2 pencil) and a long answer section
- Must notify instructor 2 weeks ahead if conflict
- Cannot miss exam without documented serious medical or personal emergency

Prerequisites

- We enforce prerequisites: 203 and 280
  - If you enrolled in EECS 281 and then received a C- in EECS 203 or 280, you must drop EECS 281
- For EECS 203, we count Math 465 and 565 (graph theory, combinatorics, etc)
- Per Departmental Policy, grad students cannot register for or audit EECS courses below 400-level (including EECS 281)

Topic Preparedness Self-survey

- There is a short survey on Canvas (Quizzes)
  - Multiple choice
  - Assessment of prerequisite material
  - Will not affect your course grade
  - Gives you practice on the Canvas “Quiz” tool
- Will help you decide whether you are adequately prepared for EECS 281

Sunday 9/9, 6:30-9pm
Stamps

- Introduce faculty and TAs
- How to:
  - Using Gitlab (https://gitlab.eecs.umich.edu)
  - Build a project with Xcode
  - Build a project with Visual Studio
  - Connect Sublime Text / Atom to CAEN
  - Use `getopt_long()`
  - Submit to the AG and read results
  - Work on CAEN (module load, etc)

Lectures and Labs

- You can print out (or use digital version) lecture notes, and must go over them
  - Before the lecture – to prepare questions
  - After the lecture – to make sure everything was clear
- If you are <u>not</u> following lecture material, don’t wait until just before the exam
  - Ask questions, attend office hours

Lectures and Exams

- Not all material presented in lecture will appear in the lecture slides
  - Explanations on the board
  - Additional practice questions
- Exam questions
  - Will test your **understanding** of material and **problem-solving skills**

Curving

- We will curve the Midterm and Final
  - Will only curve up, not down
- Not planning on an overall course curve
  - May curve up if needed, but again never down

- You can calculate what you need on the final to pass.
  - Have to pass both exams and projects

- **Let us know of any concerns <u>early</u>**

What Do I Need To Do To Pass?

- Achieve minimal competency
- If you earn **ALL OF**:

  (>= 50% on Exams) **AND** (>= 55% on Projects) **AND** (>= 75% on Labs)

  – Curved exam score
- You will pass this course
- A 75% average, with 30% projects and 90% exams is **NOT PASSING**

What Do I Need To Do To Succeed?

- Be serious and organized, stay sharp
- Allocate sufficient time for this course
- Be proactive
- Prioritize tasks -- don’t waste your time
- Don’t get stuck, do ask for help
- Practice writing code by hand! To prepare for the exams, treat Labs, projects, etc. as exam questions

Computing CARES

- View their website here:

http://www.eecs.umich.edu/eecs/about/articles/2015/Computing_CARES.html

- 281 Video:

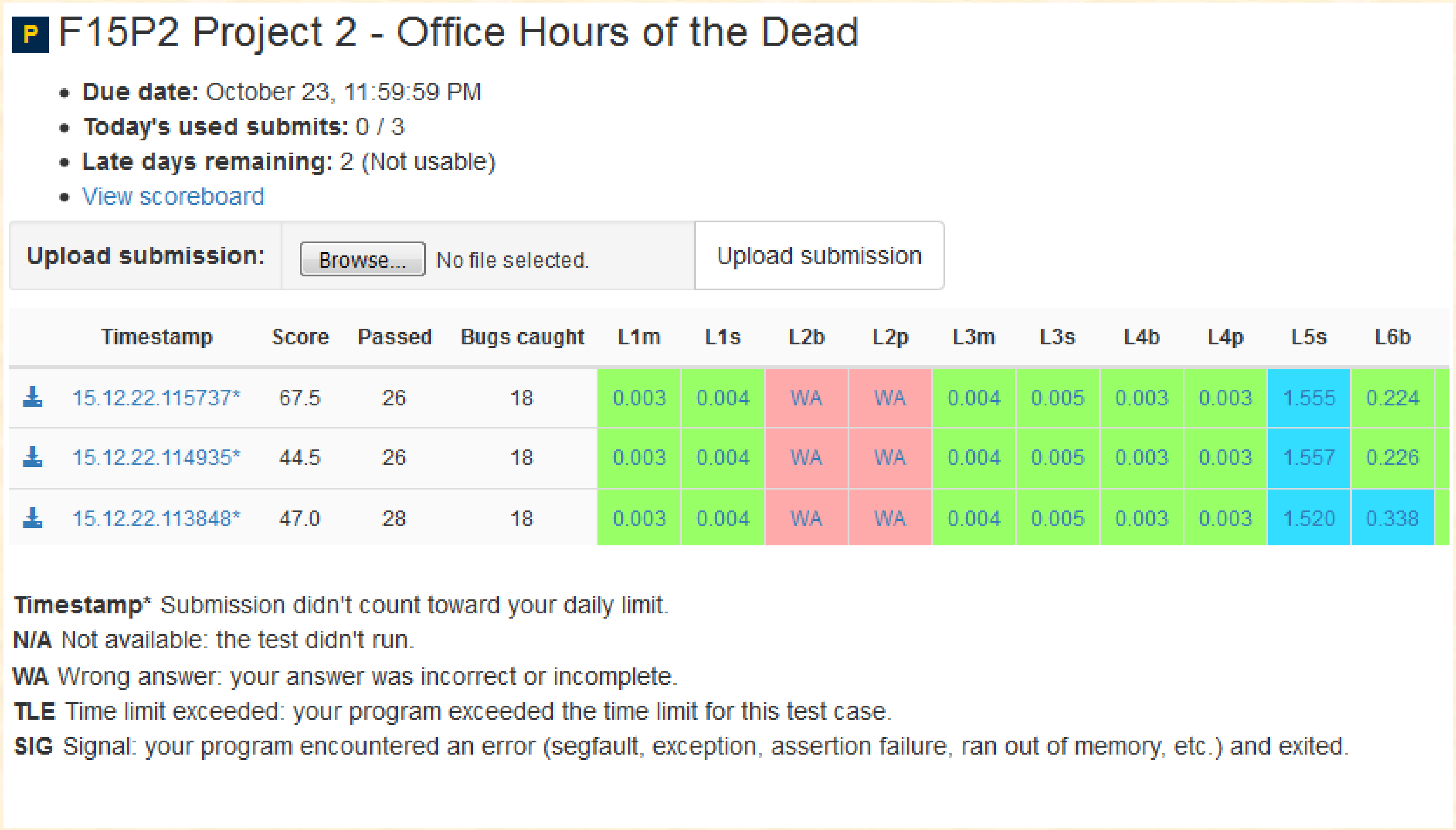
https://youtu.be/5MkRjP9qpKY

How Many Hours Per Week?

- **It varies widely**, based on
  - How well you remember EECS 280 material
    - Makefiles, library functions, debugging, using headers properly, etc.
  - Same with EECS 203 material
    - Counting, induction, complexity, summations, graphs
  - Following our directions
  - How well you plan?
  - Do you need to redo things?

Autograder

- We will grade projects with an autograder
  - Correctness
  - Timing and memory usage
- Immediate feedback on most testcases
- ~3 submissions per day
  - 3 for Projects 1-3; 4 for Project 4
  - Double in Spring
- We grade the best submission before the deadline



F15P2 Project 2 - Office Hours of the Dead

- **Due date:** October 23, 11:59:59 PM
- **Today's used submits:** 0 / 3
- **Late days remaining:** 2 (Not usable)
- View scoreboard

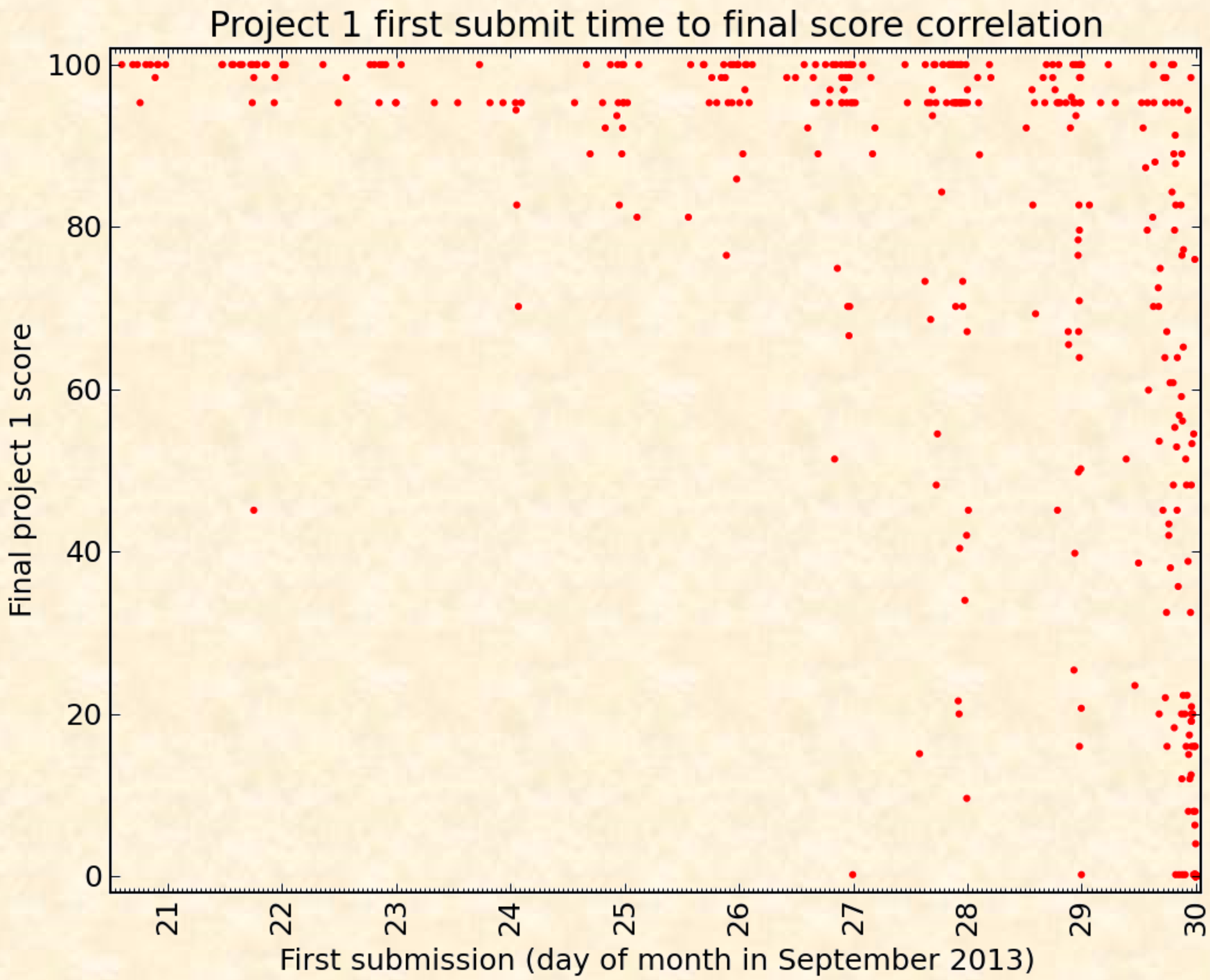
Upload submission: Browse... No file selected. Upload submission

	Timestamp	Score	Passed	Bugs caught	L1m	L1s	L2b	L2p	L3m	L3s	L4b	L4p	L5s	L6b
	15.12.22.115737*	67.5	26	18	0.003	0.004	WA	WA	0.004	0.005	0.003	0.003	1.555	0.224
	15.12.22.114935*	44.5	26	18	0.003	0.004	WA	WA	0.004	0.005	0.003	0.003	1.557	0.226
	15.12.22.113848*	47.0	28	18	0.003	0.004	WA	WA	0.004	0.005	0.003	0.003	1.520	0.338

Timestamp* Submission didn't count toward your daily limit.
N/A Not available: the test didn't run.
WA Wrong answer: your answer was incorrect or incomplete.
TLE Time limit exceeded: your program exceeded the time limit for this test case.
SIG Signal: your program encountered an error (segfault, exception, assertion failure, ran out of memory, etc.) and exited.

- Red tests failed (wrong output or exit status)
- Blue tests are over time and/or memory

Submission time vs. score



*data from EECS 281, Fall 2013

Emphasis: Get Things Done

- EECS 281 emphasizes your ability to:
  - Find the correct answer
  - Design the best algorithm
  - Design efficient data structure(s)
  - Fix bugs
  - Produce working code
  - Pass our test cases
  - Find our bugs
- Rather than getting partial solutions to two problems, solve one problem fully

Policy on Collaboration

- All work submitted must be your own
- You may not collaborate on projects
- You may use source code provided by EECS 281 instructors
- You may reuse YOUR OWN CODE if you are retaking EECS 281
- If you use other code and try to obscure it, we have automated ways to detect that

Policy on Collaboration

- Do not show your project code to others
  - Do not post code on Piazza
  - Do not use open online repositories (github, etc.)
- You may exchange project test files, but may only submit your own test files
- When in doubt, ask us, use Piazza, or come to office hours

Honor Code

- Read Honor Code (link is on Canvas)
- Please know that we take this very seriously
- We automatically check electronic submissions for violations of Honor Code

- We (teaching staff) are the ‘traffic cops’. Honor Council is the ‘court of law’

Regrades

- Your score may go up or down
- Midterm
  - Regrade requests are due 1 week from when the exam is handed back
  - Regrade requests are done via Gradescope
- For coding questions on the midterm
  - Email to eecs281admin@umich.edu
    a working / tested C++ program
    similar to what you wrote at the exam
  - We will run and test it, see how similar it is

Will Solutions Be Posted?

- Yes - for labs (see Canvas after it is scored)
- No
  - For in-class exercises
  - For projects
  - For exams
- Midterm solutions may be outlined in class
- Clarifications on Piazza and office hours
- We recommend study groups
  - After exams and project deadlines, discuss your solutions in groups

Study Groups

- Generally, a great idea
  - You will not overlook important material
  - Someone can fill you in on a lecture you missed

- Downside: your project codes may look similar and will attract extra scrutiny

OK to use Wikipedia, Google, etc.?

- Yes, it is – to understand algorithms and data structures covered in lecture
  - External sources must be mentioned in labs & projects for credit assignment reasons
  - We do not accept external references *to justify answers* on labs, exams, etc.
  - Don't copy+paste from GitHub!

Course = 1st Half + 2nd Half

- Different styles
  - 1st half – developing skills and basic knowledge; more concrete
  - 2nd half – learning sophisticated algorithms and data structures; more conceptual
- Both types of material very practical
  - We will discuss typical job interview questions in both
  - 2nd half won't be very useful without 1st half

Before the Midterm: Generic Techniques and Skills

- Complexity analysis of algorithms
- Building blocks – elementary algorithms & data structures
  - Sorting, searching, stacks and queues, priority queues (+ possibly more)
- Implementation in C++17 using STL
  - How to be efficient, what to avoid
- Time measurement and optimization
- Algorithmic problem-solving
- Examples for how to select the best algorithm for a problem

After the Midterm: Sophisticated Algorithms

- Binary search trees (dictionaries)
- Hashing and hash tables
- Graph algorithms
- Algorithm types
  - Divide-and-conquer
  - Greedy
  - Dynamic programming
  - Backtracking and branch-and-bound

Useful Software

- Tools (editors, version control, etc.)
- IDEs (Integrated Development Environments)
- Plotting/visualization

Useful tools

- Automated compilations
  - make
- Editors for “power users”
  - Vim, Emacs
- Version control system
  - Git (http://git-scm.com/), (gitlab.eecs.umich.edu)
  - CVS (http://www.nongnu.org/cvs/)
  - Subversion (http://subversion.tigris.org/) http://en.wikipedia.org/wiki/Subversion_(software)

Making Copies of your Code

- Suppose you DON’T do any of the following things (the first 3 which we suggested):
  - – Upload to the autograder
  - – Upload to CAEN to test building with g++
  - – Upload your code to the gitlab server
  - – Copy your files to a flash drive
- Then your computer dies…

Don’t let This be You

IDE

- One platform allows the use of multiple tools through a single interface
  - Text editor
    - Many have tooltip popups for method parameters
    - Some detect errors while typing
  - Advanced code browsing (look up method definitions, jump directly to them from a call)
  - Project management/make
  - Compiler, debugger, profiler
  - Some include version control

Partial List of IDEs

Multiple Platforms

- NetBeans* (free)
  - netbeans.org
  - C++, Java, etc.
  - PC, Mac, Linux
- Eclipse* (free)
  - eclipse.org
  - C++, Java, etc.
  - PC, Mac, Linux

*Need a separate g++ compiler such as Cygwin or Min-GW

Proprietary

- Visual Studio 2017, Enterprise or Community
  - Enterprise edition
  - Community edition
  - C++, C#
  - PC only
- Xcode (free)
  - apple.com
  - C++, Swift, Objective-C
  - Mac only

Plotting Tools

- Useful for plotting algorithm statistics
  - Runtimes
  - Memory Usage
  - Other parameters

- Gnuplot
  - http://www.gnuplot.info/
- Excel
  - http://www.usd.edu/trio/tut/excel/
- Matlab
  - http://www.math.ufl.edu/help/matlab-tutorial/

Why Algorithms so Early?

- CLRS Textbook
  - Simply put, it is THE textbook for Algorithms and Complexity
  - Normally used in graduate level courses
  - One of our required textbooks

Pragmatic Reasons

- Algorithms are easier to remember than the exact code
- Does not lock you in to a specific language, data structure, etc.
- Introduce earlier
  - Integrated with programming
  - Learn it better
  - Better prepared for EECS 477, 586, etc.

Complexity

- Complexity is considered in general, not using empirical data
- Analyze the algorithm, not the implementation

Algorithm Engineering

- For a given application, is it better to use:
  - Algorithm A or B?
  - Data structure X or data structure Y?
- Often you can tell before writing any code
- Sometimes you must do an empirical comparison
- Sometimes the answer is surprising
- For a given piece of code:
  - Where is the bottleneck?
  - How do you speed it up?

Algorithm Exercise

1. Write this function
2. How many multiplications will it take if size = 1 million?

```
//REQUIRES: in and out are arrays with size elements
//MODIFIES: out
//EFFECTS:  out[i] = in[0] *...* in[i-1] *
//   * in[i+1] *...* in[size-1]
void f(int *out, const int *in, int size);
```

Developing Your Skills

- Problem-solving
- Algorithm analysis
- Software development
- Practice, repetition, and rewriting
  - Building skills
- Memorization
  - Not necessarily rote!
  - Required for speed in programming

Software Engineering Issues

- When is a given technique appropriate
  - Pointers (or references), classes, STL
- Good code versus bad code
  - Modular, concise, readable, debuggable
- Functional robustness
  - Input checking, assertions, etc.
- Code reuse: less work, less debugging
- How to avoid and minimize bugs

Getting Help & Contacting Us

- For *urgent & personal* issues
  - eecs281admin@umich.edu goes to all instructors and TAs
- For *really personal* issues –
  - Make an appointment
- http://cppreference.com
- http://www.cplusplus.com/
- http://piazza.com
  - Do not post code from lab and project solutions
  - Do not ask if your solution is correct
  - You can post anonymously to other students (but we will know your name)
  - Students can answer questions of other students
  - Instructors endorse good answers
- **Please "close" your questions once answered**

Office Hours Help

- Come to office hours prepared to get help
- Bring specific questions
- Attend soon after the project is assigned and get conceptual questions answered before you start coding
- Try to fix compiler errors (Piazza, Google) before coming to office hours
  - We're happy to help, but try to solve yourself

Before Debugging Help

- Before getting help debugging, you should have:
  - Submitted to the autograder
    - Included test files of your own
    - The autograder will tell you if your own test file reveals your solution as buggy!
  - Tested all provided examples using valgrind
  - Found as small a test as possible that reveals your bug