# EECS 281
# Data Structures and Algorithms

Mr. Marcus Darden

Dr. David Paoletti

Fall, 2015

# Course Staff

**Faculty**

- Mr. Marcus Darden
  – [mmdarden@umich.edu](mailto:mmdarden@umich.edu)

- Dr. David Paoletti
  – [paoletti@umich.edu](mailto:paoletti@umich.edu)

**Teaching Assistants**

- Chun Wu
- Laura Wendlandt
- Morteza Noushad
- Spencer Kim
- Waleed Khan
- David Spencer

- Nikhill Rao
- Anna Hua
- Jessica Opaleski
- Luum Habtermariam
- Aaryman Sagar

# Course Weekly Schedule

## Lectures

- Tuesday, Thursday
- S1: 9-10:30, 133 Chrysler
- S2: 10:30-12, 1013 DOW
- S3: 12-1:30, 1610 IOE
- S4: 1:30-3, 1610 IOE
- Important announcements in lectures

## Discussions

- See the course Google Calendar on CTools

- All discussions and lectures cover the same material
  - You can attend any, provided there is space

3

# Syllabus

- Please read the syllabus on CTools

# 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://eecshelp.appspot.com/
  - 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
    - Homework          10%
    - Projects          40%
    - Midterm Exam      25%
    - Final Exam        25%

# Homework (10%)

- 4 homework assignments
- Individual work
- Submit electronically via CTools and/or autograder machine
- Late submissions: must contact instructors before due date via [eecs281admin@umich.edu](mailto:eecs281admin@umich.edu) and provide documentation (extreme medical/personal)

# Projects (40%)

- Four projects
- Individual work
- Submitted electronically to autograder
  - Details to follow
- 10-14 days per project
- Late submissions: must contact instructors before due date via eecs281admin@umich.edu and provide documentation (extreme medical/personal)

# Projects (40%)

- C++ (International C++ 11 Standard)
  - **http://en.wikipedia.org/wiki/C++11**
- CAEN Linux Computing Environment
  - g++ (GCC) 4.8 (either 4.8.3 or 4.8.2)
  - May switch to g++ 5.1.0 after autograder machines updated (allows C++ 14 standard)
- Beware if you are doing development in any other environment

# Exams (50%)

- Midterm Exam (25%)
  – Wednesday 27 October, 7-9pm
- Final Exam (25%)
  – Friday 18 December, 8-10am
- 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)
- This is not a course for graduate students who wish to learn programming

# Topic Preparedness Self-survey

- There will be a short survey on Ctools (to be posted soon)
- Multiple choice
- Assessment of prerequisite material
- Will not affect your course grade
- Will help you decide whether you are adequately prepared for EECS 281

# Lectures and Discussions

- You must print out (or use digital version) lecture notes, and 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 the midterm
  - 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, extra-credit questions
- Exam questions
  - Will test your **understanding** of material and **problem-solving skills**

# Course Grade is Curved

- The final is cumulative with a heavy focus on the 2$^{nd}$ half
- Curving is done for the entire course grade, not per assignment

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

# What Do I Need To Do To Pass?

- Achieve minimal competency
- If you earn **BOTH**:

  >= 50% on Exams

  >= 50% on Projects

- You will pass this course

# 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

# 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 some testcases
- ~3 submissions per day with feedback
- Unlimited submissions without feedback
- We grade the last submission before the deadline

# P W15 Project 4 - Gotta Catch 'Em All!

- **Due date:** April 21, 11:59:59 PM
- **Today's used submits:** 0 / 4
- View scoreboard

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

| | Timestamp | Score | Passed | Bugs caught | AA | AB | AC | AD | AE | A_CN1 | A_CN2 | A_CN3 | A_CN |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ⬇ | 15.04.30.073346* | 0.0 | 0 | 0 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| ⬇ | 15.04.29.134944* | 67.0 | 26 | 6 | 1.067 | 1.367 | 1.929 | 2.406 | 3.077 | 2.835 | 6.985 | 10.352 | 9.08 |
| ⬇ | 15.04.23.150435 | 56.6 | 25 | 6 | 0.706 | 0.981 | 1.338 | 1.773 | WA | 2.086 | 5.052 | 7.999 | 6.55 |
| ⬇ | 15.04.22.115608* | 60.2 | 22 | 6 | 0.701 | 0.981 | 1.336 | 1.766 | WA | 2.082 | 5.031 | N/A | 6.54 |
| ⬇ | 15.04.15.101244* | 61.7 | 25 | 5 | 0.676 | 0.917 | 1.343 | 1.790 | 2.109 | 1.952 | 5.362 | N/A | 7.13 |

**Timestamp*** Submission didn't count toward your daily limit.

**N/A** Not available: the test didn't run.

**HID** Hidden: this was a blind submit and the results are hidden.

**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.

# Policy on Deadlines

- No late days, no personal extensions
  - For extraordinary events, we cannot act on information received after the deadline
- Metaphor: airport departures
  - Assignments should be completed before stated deadlines

# Submission time vs. score



Project 1 first submit time to final score correlation

*data from EECS 281, Fall 2013
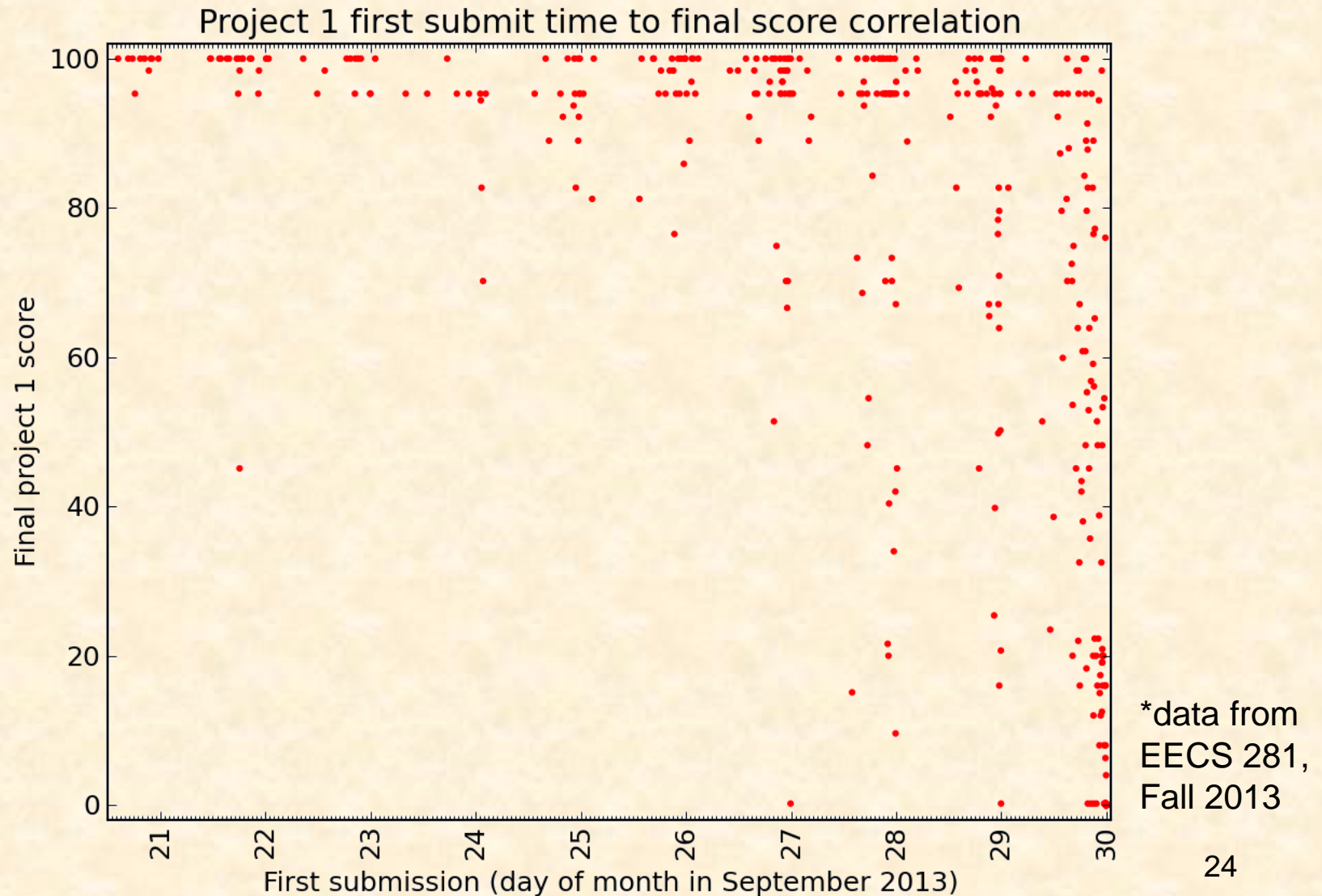
# Emphasis: Get Things Done

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

# Honor Code

- Read Honor Code (link is on CTools)
- 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 (rain or shine) from when the exam is handed back
  - Regrade requests are due in writing ***
- 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 homeworks (see CTools 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
- <u>We recommend study groups</u>
  - 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 HWs & Projects for credit assignment reasons
  - We do not accept external references *to justify answers* on HWs, exams, etc.

# Course = 1$^{st}$ Half + 2$^{nd}$ Half

- Different styles
  - 1$^{st}$ half – developing skills and basic knowledge; more concrete
  - 2$^{nd}$ half – learning sophisticated algorithms and data structures; more conceptual
- Both types of material very practical
  - We will discuss typical job interview questions in both
  - 2$^{nd}$ half won't be very useful without 1$^{st}$ 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++11 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

34

# 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
  - CVS (http://www.nongnu.org/cvs/)
  - Subversion (http://subversion.tigris.org/)
    http://en.wikipedia.org/wiki/Subversion_(software)
  - Git (http://git-scm.com/)

# 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

## Cross Platform

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

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

## Proprietary

- Visual Studio 2015 Professional (students: free)
  - [CAEN Dreamspark](CAEN Dreamspark)
  - [dreamspark.com](dreamspark.com)
  - C++, C#
  - PC only
- Xcode (free)
  - [apple.com](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, 480, 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://piazza.com
  - <u>Do not post code from HW and project solutions</u>
  - <u>Do not ask if your solution is correct</u>
  - You can post anonymously to other students
    (but we will know your name)
  - Students can answer questions of other students
  - Instructors endorse good answers

- **Make sure to "close" your questions where you have an answer**