Applied Algorithms Lec 1: Welcome (and maybe some C)

Sam McCauley

October 21, 2021

Williams College

Welcome!

- Welcome back to campus.
- Can everyone see me and the projector?

Admin

- Colloquium Fridays at 2:30
- Some attendance required for majors
- Welcome colloquium this week

About the Class

- Goal: bridge the gap between theory and practice
- How can theoretical models better predict practice?
- Useful algorithms you may not have seen
- Using algorithmic principles to become better coders!

Pantry Algorithms

- Algorithms that you should always have handy because they are incredibly useful
- Bloom filters, linear programming, suffix trees
- What drives the course
- Algorithmic understanding of these ideas!

Coding

- We'll be doing some coding practice each week
- Code review from time to time
- Collaboration highly encouraged
- Optional, friendly competition for those who want to optimize code (with some bonus points)

About Me

- Call me Sam
- Research is in algorithms
    - Some experimental algorithms
- Office is TCL 306
- Office Hours Mon 3-5pm TCL 306, Tue 3-5pm TCL 312

One TA!

- Chris Chung
- Took the course last year
- He'll help with some back end stuff and also hold office hours
- Wed 8-10 TCL 312

About the Course

- No course textbook; some suggested readings
- Textbooks for background will be left in TCL 312.
- Was taught in Spring 2020
- ...probably will need to make some pacing adjustments in the second half
- Questions *particularly* welcome!

Help, Questions, Comments, Etc.

- Slack, email
- During or after class
- Stop by the lab during (or not during) office hours
- Stop by my office (no promises!)

Help, Questions, Comments, Etc.

- Slack, email
- During or after class
- Stop by the lab during (or not during) office hours
- Stop by my office (no promises!)

Covid stuff

- Purell while going into lab
- Masks whenever you're in an academic building (certainly in labs)
- We'll try to do everything in person for now, but we'll play it by ear. Let me know if you have questions.
- Keep in touch, especially if something changes about your situation.
- Questions?

Lab

- TCL 312
- Passcode (write it down)
- Office hours will generally be in TCL 312
- Feel free to stop by.
    - No one else has reserved it
    - But others use it—keep an eye out for occupancy
- Purell going in, masks on.
- Don’t spray down keyboards (it will brick them)
- No food or drink this semester!

Theory assessment

- A small number of problems each week
- Don't fall behind! (Or get too distracted by coding)
- Goal: Understanding how the algorithms work
- Especially important on the final

Coding assessment

- (Almost) all in C
- Weekly assignments
- Assignment 1 is designed to give you an opportunity to catch up
- Grading generally not too strict
- (Mostly) no parallelism in this course

Why C?

```
register short *to, *from;
register count;
{
    register n = (count + 7) / 8;
   switch (count % 8) {
   case 0: do { *to = *from++;
   case 7:      *to = *from++;
   case 6:      *to = *from++;
   case 5:      *to = *from++;
   case 4:      *to = *from++;
   case 3:      *to = *from++;
   case 2:      *to = *from++;
   case 1:      *to = *from++;
        } while (--n > 0);
}
```

- Familiarity
- Low-level
  - Course is about how design decisions affect performance
- Fast, useful to know
- A couple specific features we'll be using

Summary of Policies and Assessments

Mini-midterms

- 3 during the semester
- Look like assignments, handed in like assignments
- All work must be *entirely* your own!
    - No instructor or TA help;
    - No help from other students; no online resources
    - Contact me with any questions or if issues come up

Take-home final

- 24 hour take home final
- No help whatsoever (of course)
- Some coding, but main focus is on understanding: both how algorithms work, and why certain implementations have certain effects

Weekly Assignments

- Due Wednesday 10pm
- Released one week before
- Late penalty 20% per day
  - Let me know if there is some reason why you cannot make it!
  - I have no problems giving late days if the need arises
  - (Seriously do this 🙂)

Assignment Honor Code Policies: Problem Set Questions

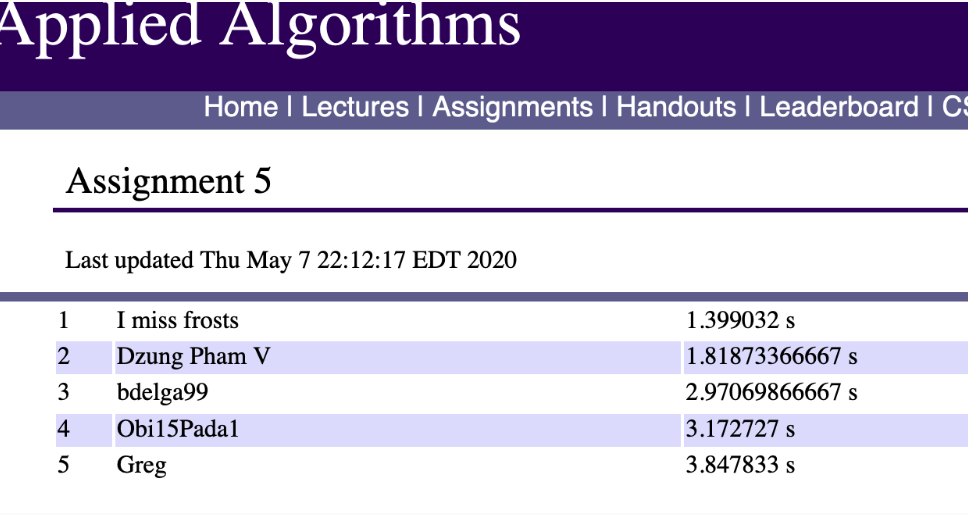
- Normal CS department assignment rules
- You must do by yourself
- Instructor and TA can help
- Can discuss high-level strategies with other students ("hands-in-pockets" rule)
- Can ask other students about debugging and syntax issues

Assignment Honor Code Policies: Code

- You can collaborate with other students and use online resources
- You may share code! (But you **must cite!!!**)
- You have to understand anything you submit.
    - I may actually ask you about code you've written—possibly because what you've done is interesting (though it may also be to ensure you're keeping up)
- Details in syllabus; let me know if you have questions

“Leaderboard” extra credit

- On some assignments we’ll have a fun competition to see who can write the fastest implementation
- Totally optional!
- First–third fastest will get 30, 25, 20 extra points
- +5 if you are faster than last year’s best
- Current 5 fastest times will be (anonymously) posted on website, along with last year’s and my implementation


Applied Algorithms

Home | Lectures | Assignments | Handouts | Leaderboard | C

Assignment 5

Last updated Thu May 7 22:12:17 EDT 2020

1	I miss frosts	1.399032 s
2	Dzung Pham V	1.81873366667 s
3	bdelga99	2.97069866667 s
4	Obi15Pada1	3.172727 s
5	Greg	3.847833 s

Grading

- Assignments: 20%
- Mini-midterms: 20% each
- Final: 20%

Let’s look over the syllabus quickly

Course Website

“Assignment” 0

- Not worth points
- Due next Wednesday
- Just asks for your name and Github
- You can’t do Assignment 1 without it!

Coding in C

Plan for this section

- Quick review of some key concepts
- Emphasize some particularly important areas for this course
- Use the first week as an opportunity to catch up!
- Instructor, TA, other students, even stackexchange (etc.) are all good resources for questions you may have[1]

[1] Just remember to be sure that you can explain anything you submit.

About C

- Lifetime of information to learn
- I am not an expert (though I've used it a lot)
- Many interesting features, many interesting behind-the-scenes effects
- Close connection between your code and the computer's actions

Arrays

- Really just pointers
- No bounds checking
- Can use `sizeof` for fixed-size array (compiler replaces with size at compile time). Also works with variables

Structs

- What C has instead of classes
    - No member functions
    - Still uses `.` operator to access member variables
- Sequence of variables stored contiguously in memory
- Semicolon after declaration
- Need to use `struct` or `typedef` to refer to structs.

Two Examples

- `struct.c`
  - `typedef` to make things easier
- `pointers.c`
  - Local variables different local vs remote
  - Access out of bounds
  - Values change(?) with different optimizations
  - `valgrind` to catch these issues

Memory Allocation

- `malloc` and `free`
  - Also use `calloc` and `realloc`
  - Need `stdlib.h`
- If you call `C++` code, be careful with mixing `new` and `malloc`
- Use useful library functions like `memset` and `memcpy`
- Example: `memory1.c`

Sorting in C

- `qsort()` from `stdlib.h`
- Takes as arguments array pointer, size of array, size of each element, and a comparison function
- What's a downside to this in terms of efficiency?
- Many ways to get better sorts in `C`:
    - Nicely-written homemade sort
    - `C++ boost` library
    - Third-party code
- Instructions to get this to work in handouts on the website (**strictly optional**)

Makefiles

Notes on `C` and compilation

- We use `gcc` in this course
- Macs tell you they have `gcc` but it is not; it is actually `clang`
- Unlikely to make too much of a difference, but one reason to use lab computers if you're running into issues

Architecture

- x86 architecture (not AMD, not M1)
- This *is* likely to have an effect on fine-grained performance in some cases
- Your home computers are fine for correctness and coarse optimization; use lab computers for fine-grained optimization
- If I ask you to do a performance comparison, either is fine—just make sure it's consistent, and make sure you write what you're using.