LAST DETAILS: OH, AND QUICKSORT

LECTURE 13–1

JIM FIX, REED COLLEGE CSCI 121

COURSE INFO

- **Today:** our last lecture(!!!)
  - Because of lab timing, I never finished covering sorting algorithms.
    - ➡ We will (quickly) look at **quick sort**.
  - Also, I want to give a quick overview of the CS major.
  - And then I want to reserve time for course evaluations.
- **Project 4:**
  - The completed project is due Friday, May 2nd, 11:59pm.
    - ➡ ***Please, please, please*** make a list of mods, with points, tallying a total.
  - I will run upstairs after lecture to field questions in my office.
- **Final Exam:**
    - ➡ Thursday, May 15th, 1-4pm, Library 204 (here).

FINAL EXAM

Details:

- ➡ Thursday, May 15th, 1-4pm, Library 204.
- Closed book, closed note, closed computer
- Written answers, similar to quizzes and midterms
- 3 hours long
- Usually 8-10 quiz-length questions; about two midterms
- Comprehensive; covers all the material (see next slide).

Preparation:

- I will post a practice exam during reading week.
- I will post my solutions to these by the end of reading week.

COURSE TOPICS

- scripting with `input` and `print`
- variables and assignment
- integer arithmetic, `%` and `//`, integer comparisons, boolean connectives
- strings and string operations
- conditional statements and loops
- function definitions
- printing versus returning, the `None` value
- lists and dictionaries
- object-orientation and inheritance
- linked lists and binary search trees
- sorting and searching, efficiency and running time
- higher-order functions and `lambda`
- recursive functions
- file I/O and error handling

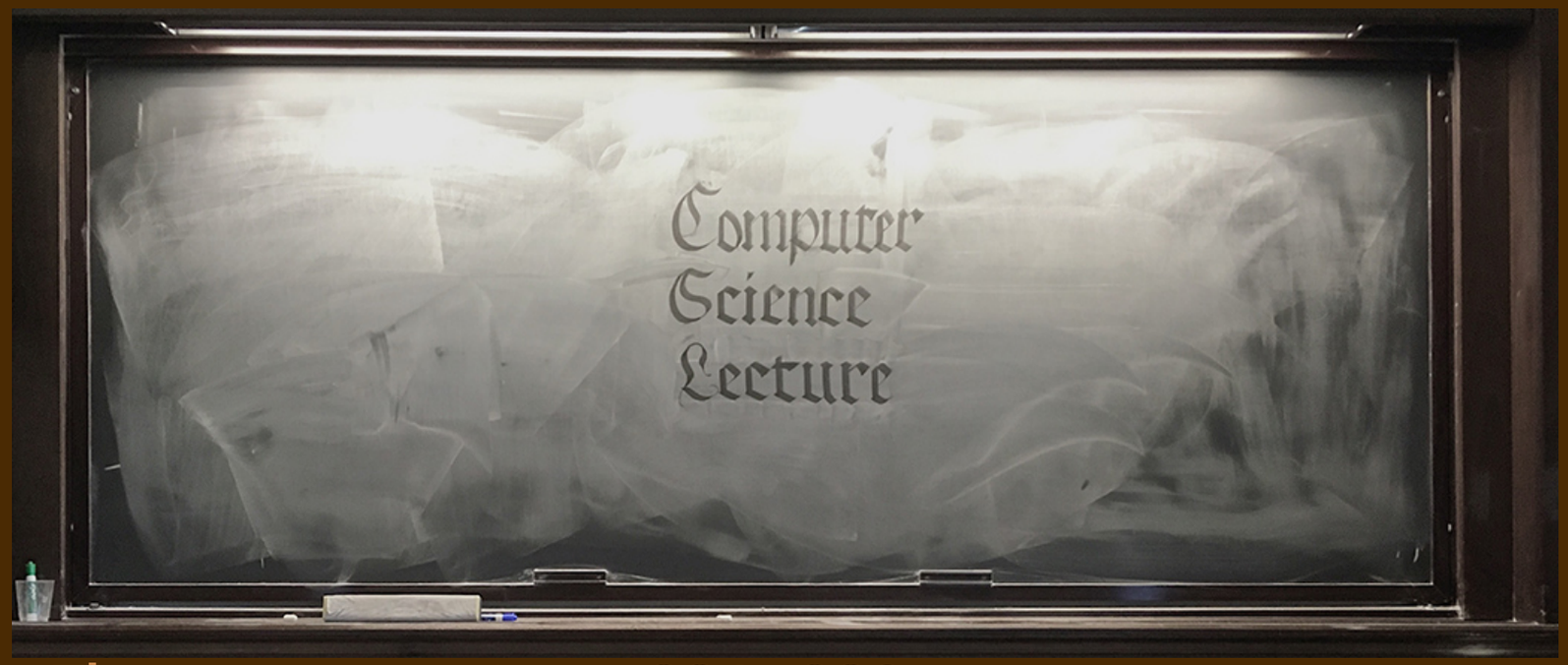
NEXT COURSES

- **CSCI 221 : CS Fundamentals II**
  - low level computer details
    - ➡ digital logic and circuits
    - ➡ processor machine language
    - ➡ program memory layout: registers, stack, heap
    - ➡ pointers/addresses
  - "industrial" level programming in C/C++
    - ➡ object-oriented language with "template" classes
    - ➡ sophisticated memory management
    - ➡ rich, complicated "standard template" library
  - more coding: short programs and larger projects
  - more experience using programmer tools: Unix, git, debuggers, profilers

NEXT COURSES (CONT'D)

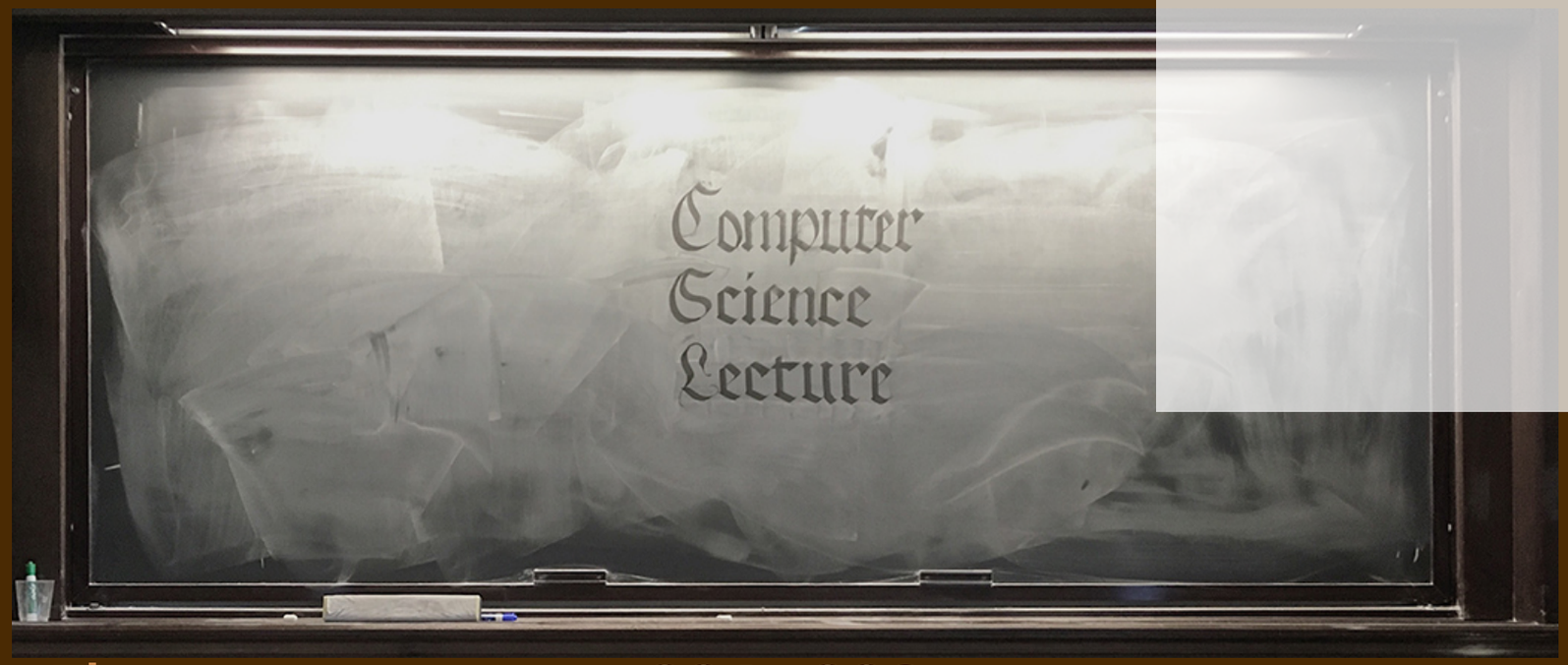
- **MATH/CSCI 382 : Algorithms & Data Structures**
  - careful, mathematical treatment of coding
  - runtime analysis; revisit sorting and searching
  - lots of nifty data structures
  - lots of nifty algorithms and their applications:
    - ➡ network/graph analysis
- Requires **MATH 112: Intro to Analysis**
  - ➡ teaches you to make careful mathematical arguments
- Requires **MATH 113: Discrete Structures**
  - ➡ teaches you "computer science" mathematics
  - ➡ develops problem-solving skills
  - ➡ more mathematical proofs, different than MATH 112

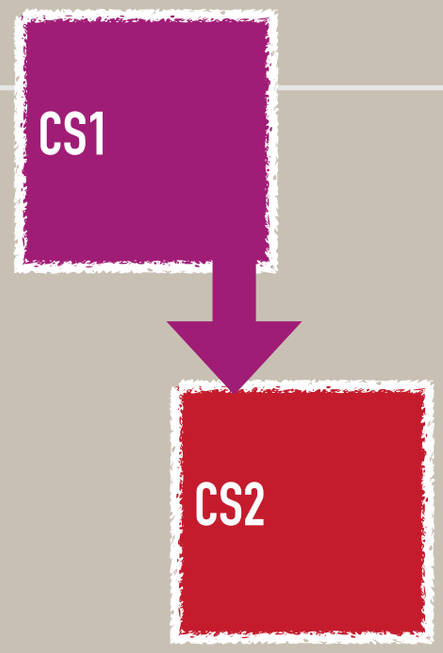
THE CS MAJOR AT A GLANCE



- Three entry points: CS1, CS2
- Three core requirements: A & DS, "Comp Comp", Systems
- Four or more advanced electives from a rotating menu.
  - ML, PLs, CG, Security, Arch, Networks, Crypto, Topics in ..., Ethics & PP, ...

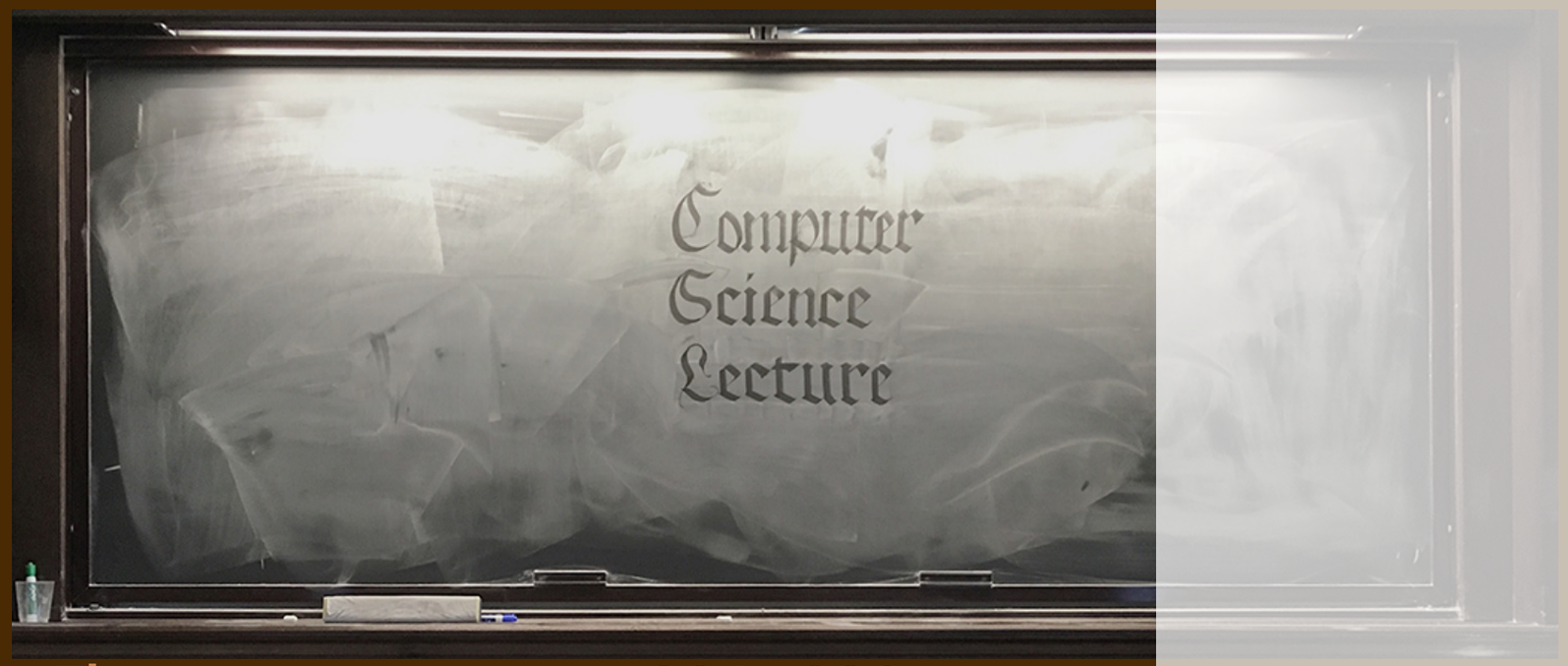
THE CS MAJOR AT A GLANCE

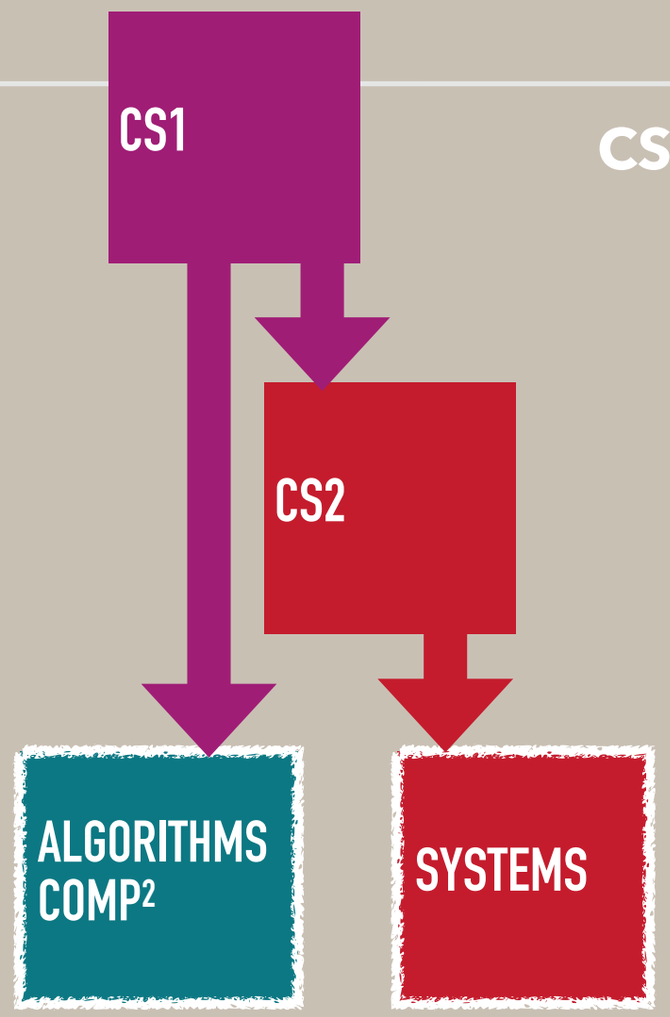




- Three entry points: **CS1, CS2**
- Three core requirements: A & DS, "Comp Comp", Systems
- Four or more advanced electives from a rotating menu including
  - ML, PLs, CG, Security, Arch, Networks, Crypto, Topics in ..., Ethics & PP, ...

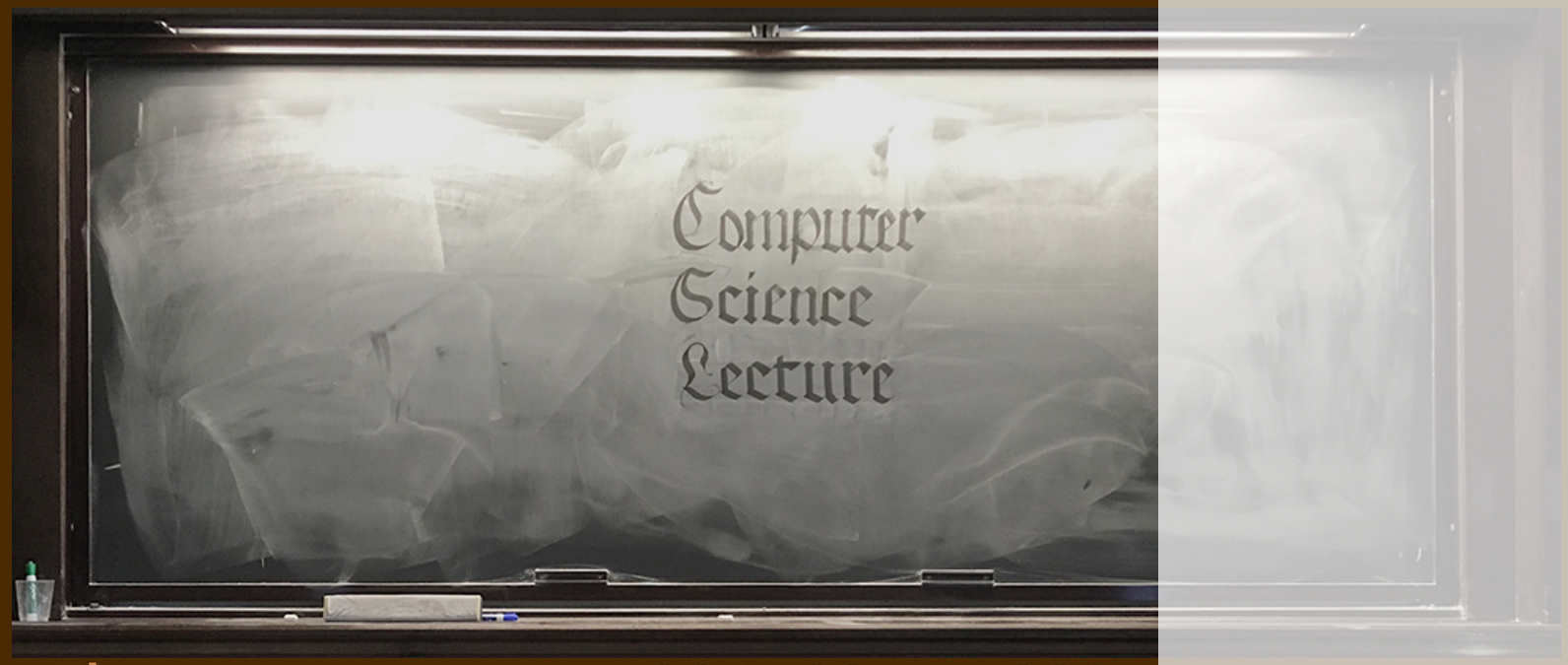
THE CS MAJOR AT A GLANCE



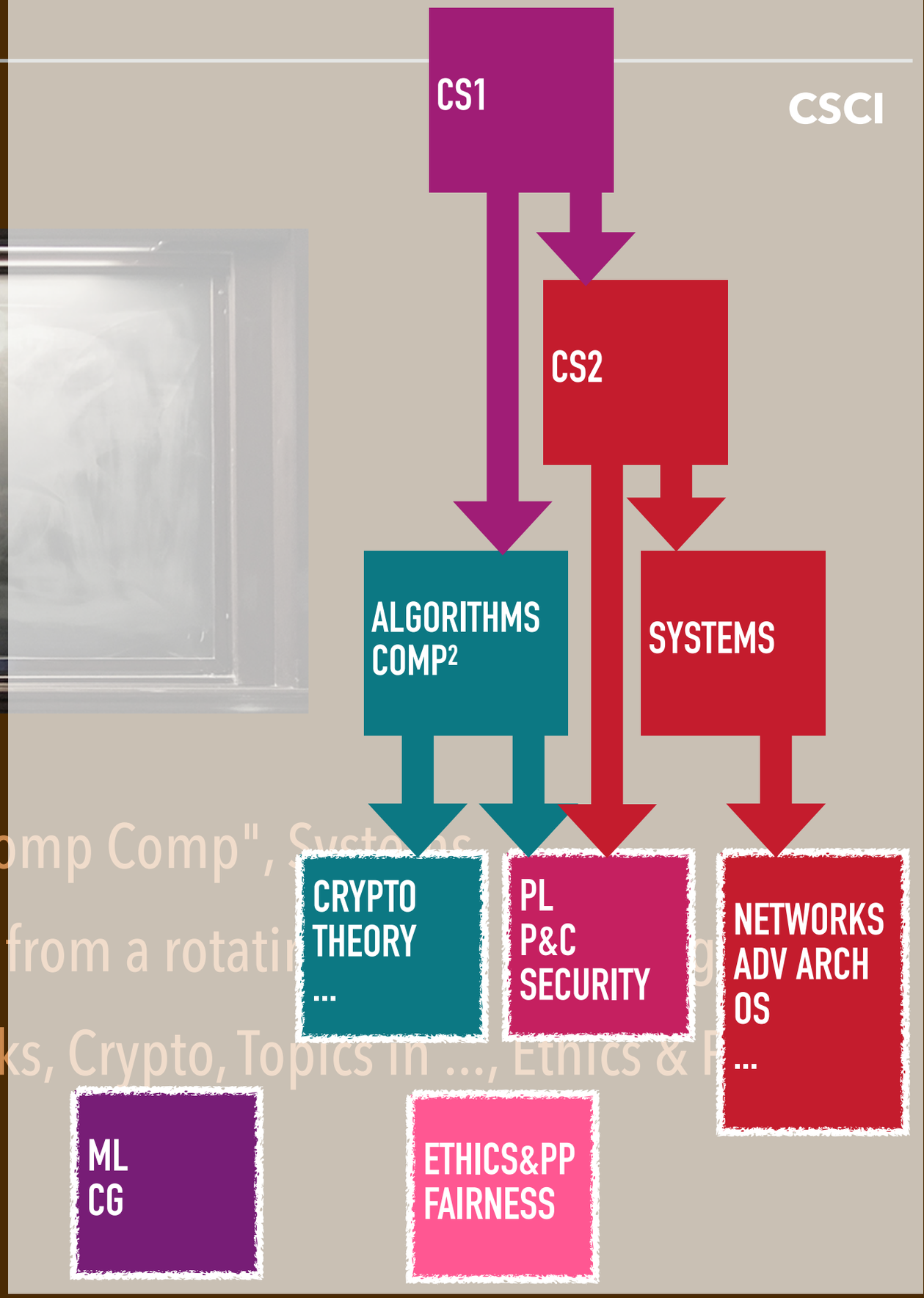


- Three entry points: CS1, CS2
- **Three core requirements: A & DS, "Comp Comp", Systems**
- Four or more advanced electives from a rotating menu including
  - ML, PLs, CG, Security, Arch, Networks, Crypto, Topics in ..., Ethics & PP, ...

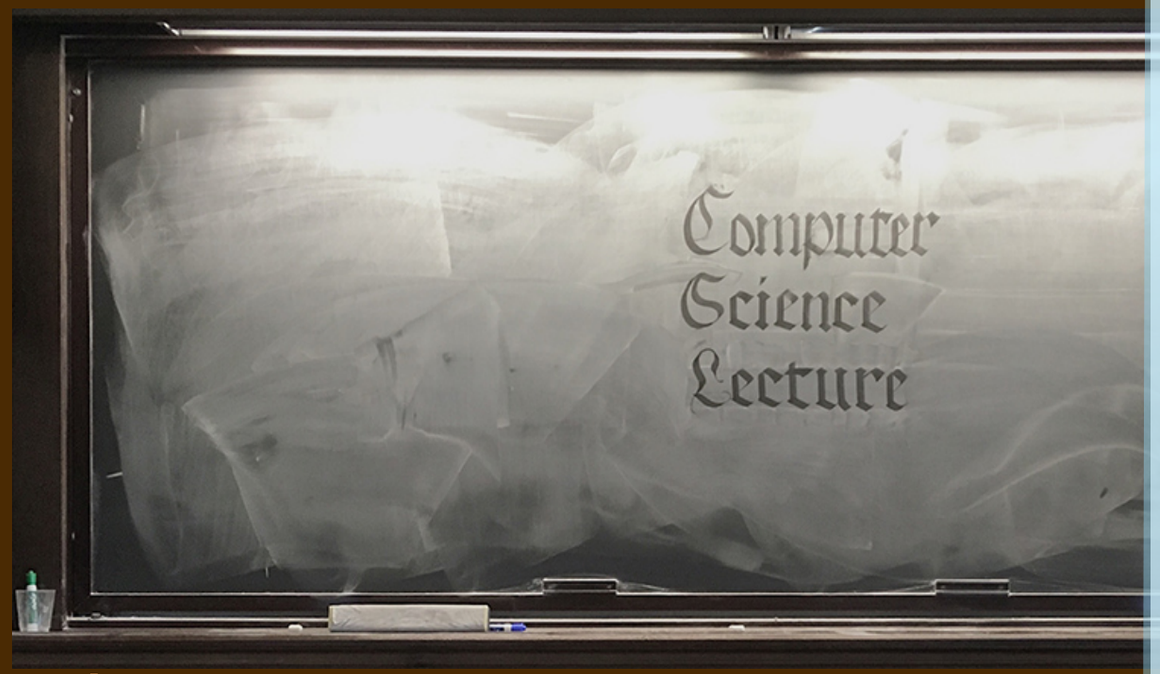
THE CS MAJOR AT A GLANCE



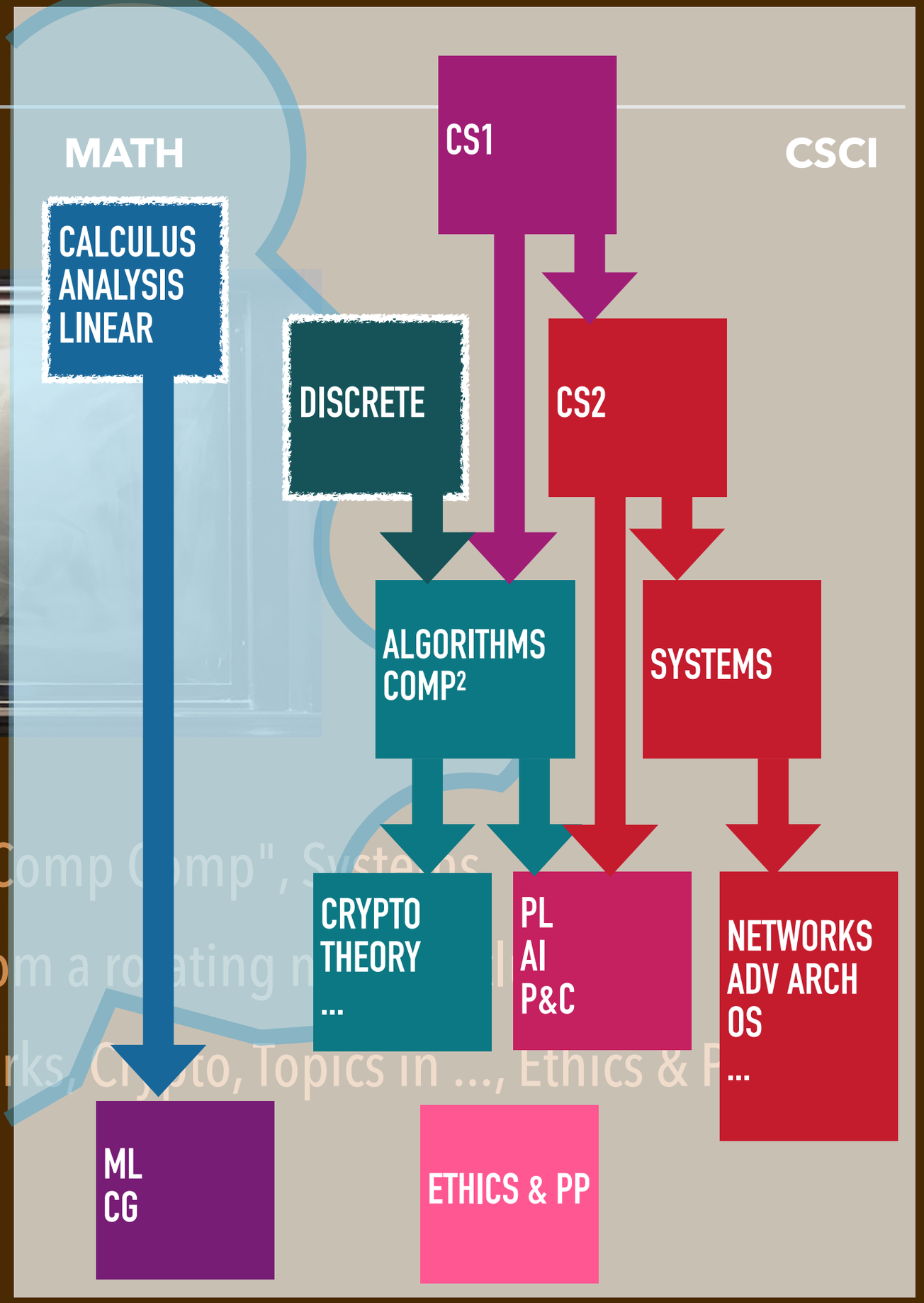
- Three entry points: CS1, CS1+, CS2
- Three core requirements: A & DS, "Comp Comp", Systems
- **Four or more advanced electives** from a rotating
  - ML, PLs, CG, Security, Arch, Networks, Crypto, Topics in ..., Ethics & P



THE CS MAJOR AT A GLANCE



- Three entry points: CS1, CS1+, CS2
- Three core requirements: A & DS, "Comp Comp", Systems
- Four or more advanced electives from a rotating menu
  - ML, PLs, CG, Security, Arch, Networks, Crypto, Topics in ..., Ethics & PP
- MATH plays a critical role.



QUICKSORT

- A sorting algorithm that partitions then recursively sorts.

```
def quicksort(someList):
    if len(someList) == 0:
        # It's already sorted! BASE CASE.
        return []
    else:
smaller,pivot,larger = partition(someList)
smallerSorted = quicksort(smaller)
largerSorted = quicksort(larger)
return smallerSorted + [pivot] + largerSorted
```

PARTITIONING A LIST "AROUND" A PIVOT VALUE

- Here is the code for partitioning a list:

```
def partition(someList):
    smallers = []
    pivot = someList[0]  # pick some value from the list
    largers = []
    for x in someList[1:]:
        if x <= pivot:
            smallers.append(x)
        else:
            largers.append(x)
    return smallers, pivot, largers
```

PARTITIONING A LIST "AROUND" A PIVOT VALUE

- Here is the code for partitioning a list:

```
def partition(someList):
    smallers = []
    pivot = someList[0]  # pick some value from the list
    largers = []
    for x in someList[1:]:
        if x <= pivot:
            smallers.append(x)
        else:
            largers.append(x)
    return smallers, pivot, largers
```

- This always picks the left element as the pivot. Other pivot choices:
  - Find the median.
  - Pick a random element.
  - Choose the median of the left, middle, and right.

PARTITION

- Here is the code for partitioning a list:

```
def partition(someList):
    smallers = []
    pivot = someList[0] # pick some value from the list
    largers = []
    for x in someList[1:]:
        if x <= pivot:
            smallers.append(x)
        else:
            largers.append(x)
    return smallers, pivot, largers
```

- This always picks the left element as the pivot. Other pivot choices:
  - Find the median. ***Ideal, but expensive.***
  - Pick a random element. ***Good, but has some overhead.***
  - Choose the median of the left, middle, and right. ***Usually good enough.***

ANIMATION OF QUICKSORT

A BAD CASE FOR QUICKSORT

PROBALISTIC ANALYSIS OF QUICKSORT

CAN WE DO BETTER THAN THETA(N LOG(N))?

SORTING AND SEARCHING SUMMARY

- Sorting a list makes information retrieval faster:
  - can use binary search to check membership in $\mathbf{O}(\log_2(n))$ time.

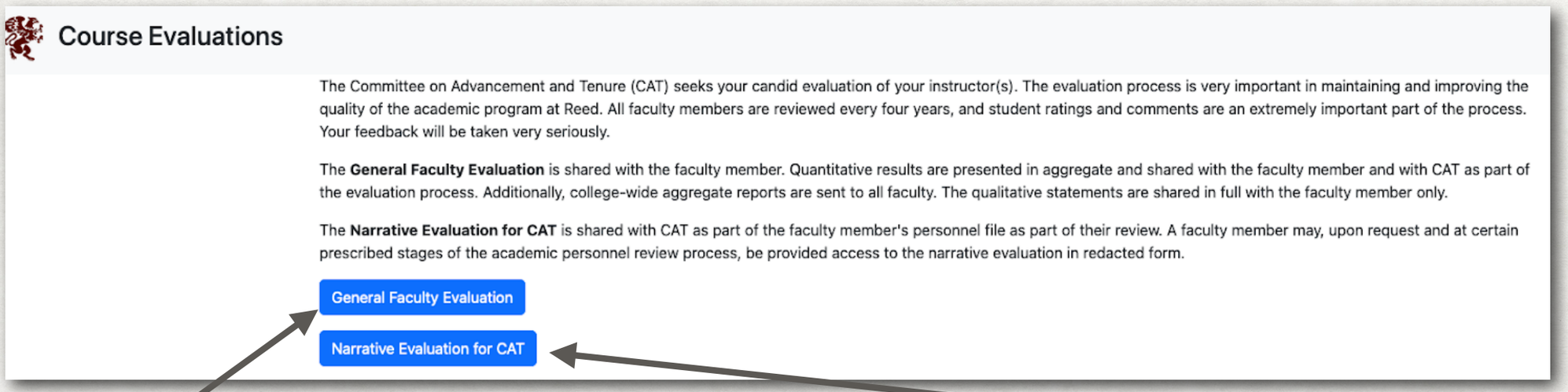
- "First try" sorting algorithms typically sort in quadratic time.
  - bubble sort, insertion sort, selection sort, etc.
  - They essentially (in the worst case) compare every item to every other.
  - This means they might perform $1 + 2 + 3 + \ldots + (n\text{-}1)$ comparisons.
    - ➡ That sum is $n\,(n\text{-}1)\,/\,2$ and so that leads to $\boldsymbol{\Theta}(n^2)$ comparisons.

- Faster sorts use recursion:
  - Merge sort sorts in $\boldsymbol{\Theta}(n \log_2(n))$ time.
  - Quick sort typically sorts in $\boldsymbol{\Theta}(\text{n } \log_2(n))$ time.
    - ✦ With bad pivot choices, can take $\boldsymbol{\Theta}(n^2)$ time. Can be avoided with randomness.
  - Can't do better than $\boldsymbol{\Theta}(n \log_2(n))$ time if comparison-based.

COURSE EVALUATIONS

FACULTY EVALUATION INSTRUCTIONS

- Step 1: Open up this course on Moodle
- Step 2: Click on "Course Evaluations" link (under the 'General' category at the top of the page)
- A new Moodle page opens that looks like this:



Step 3: Click top link to fill out a survey and provide feedback. (Survey results go to me and the college, feedback only goes to me.)

Step 4: Click bottom link to give feedback to the Committee on Advancement and Tenure (I do not see this.)