CSE100: Algorithm Design and Analysis
Lecture 01 – Introduction

January 21st 2026

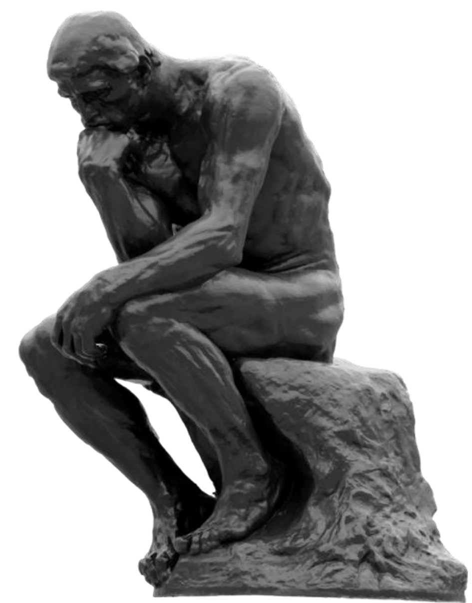
Logistics and Introduction

Let's start with some logistics

- CatCourses for Assignments, Files
- Piazza for Announcements, Discussion Forum
- Textbook(s):
  - CLRS: Introduction to Algorithms (main)
  - DPV: Algorithms (optional)
  - Kleinberg and Tardos: Algorithm Design (optional)
- Lab Assignments:
  - posted on Mondays, due in 7 days
  - See Late Policy in Syllabus for more details
- Exams:
  - 4 midterms and final, tentative dates on Syllabus.

The big questions

- Who are we?
  - Professor, TAs, students?
- Why are we here?
  - Why learn about algorithms?
- What is going on?
  - What is this course about?
  - Logistics?
- Wrap-up



Welcome to CSE 100!

Who are we?

- Instructors:
  - Ross Greer
  - Hua Huang
- Awesome TAs:
  - Shuo Chen
  - Wesley Ferreira Maia
  - Yi Liu

Who are you?

- CSE majors…
- Some Math

Acknowledgements

Thank you to Professors Cerpa, Chandrasekhar, Su, Huang, and others for slide and content contributions for this course (and all animal alliterations).

You are better equipped to answer this question than I am, but I'll give it a go anyway…

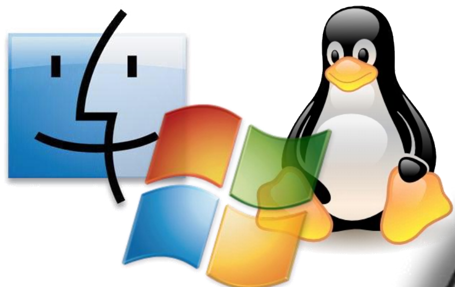
Why are you here?

- Algorithms are fundamental
- Algorithms are useful.
- Algorithms are fun!
- CSE100 is a required course.

Why is CSE100 required?

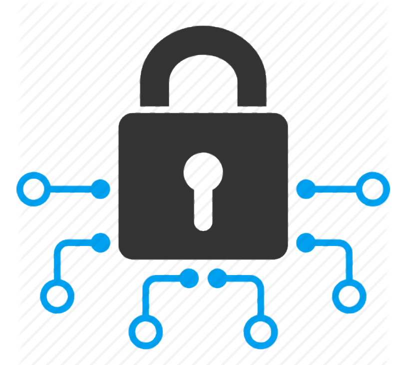
- Algorithms are fundamental.
- Algorithms are useful.
- Algorithms are fun!

Algorithms are fundamental



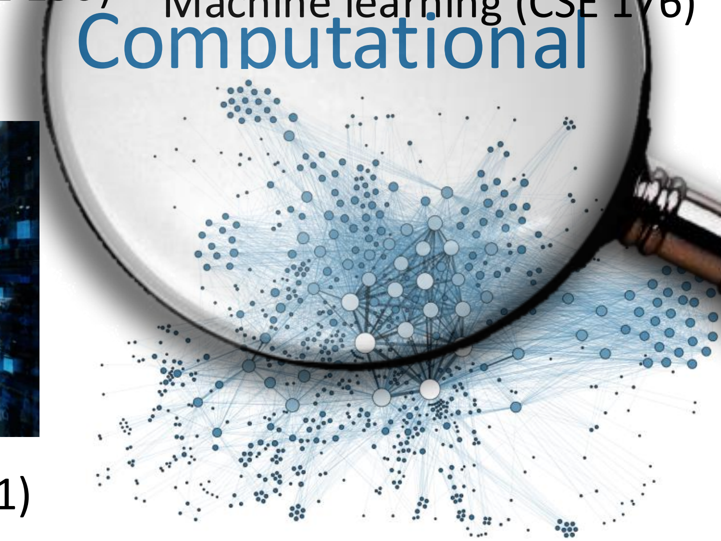
Operating Systems (CSE 150)

Machine learning (CSE 176)

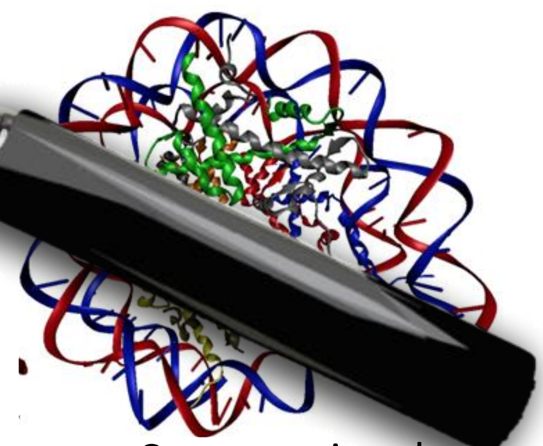


Computers and Network Security (CSE 178)

Database Systems (CSE 111)



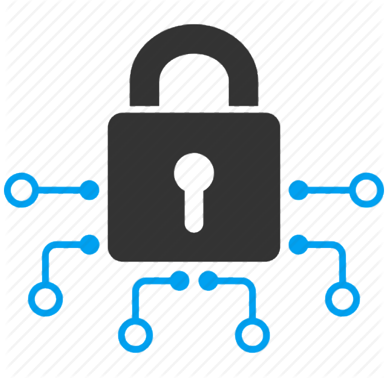
Networking (CSE 160)

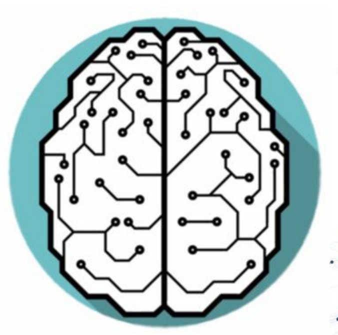


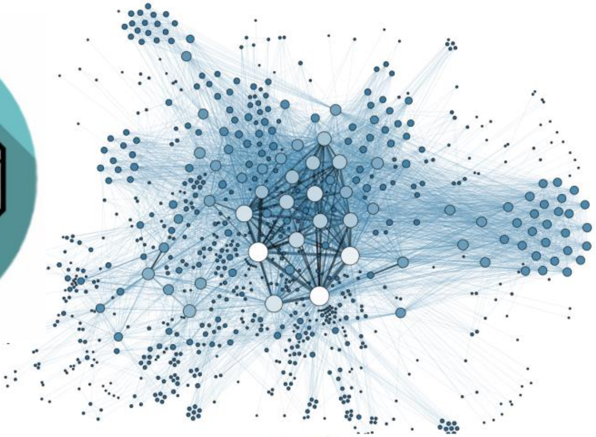
Computational Neuroscience (CSE 173)

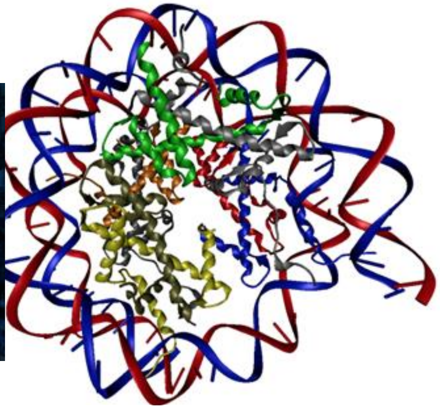
Algorithms are useful

- All those things, without UC Merced CSE course numbers
- As we get more and more data and problem sizes get bigger and bigger, algorithms become more and more important.
- Will help you get a job.









Algorithms are fun!

- Algorithm design is both an art and a science.
- Many surprises!
- A young field, many exciting research questions!

- (Will help you get a job you like!)

What’s going on?

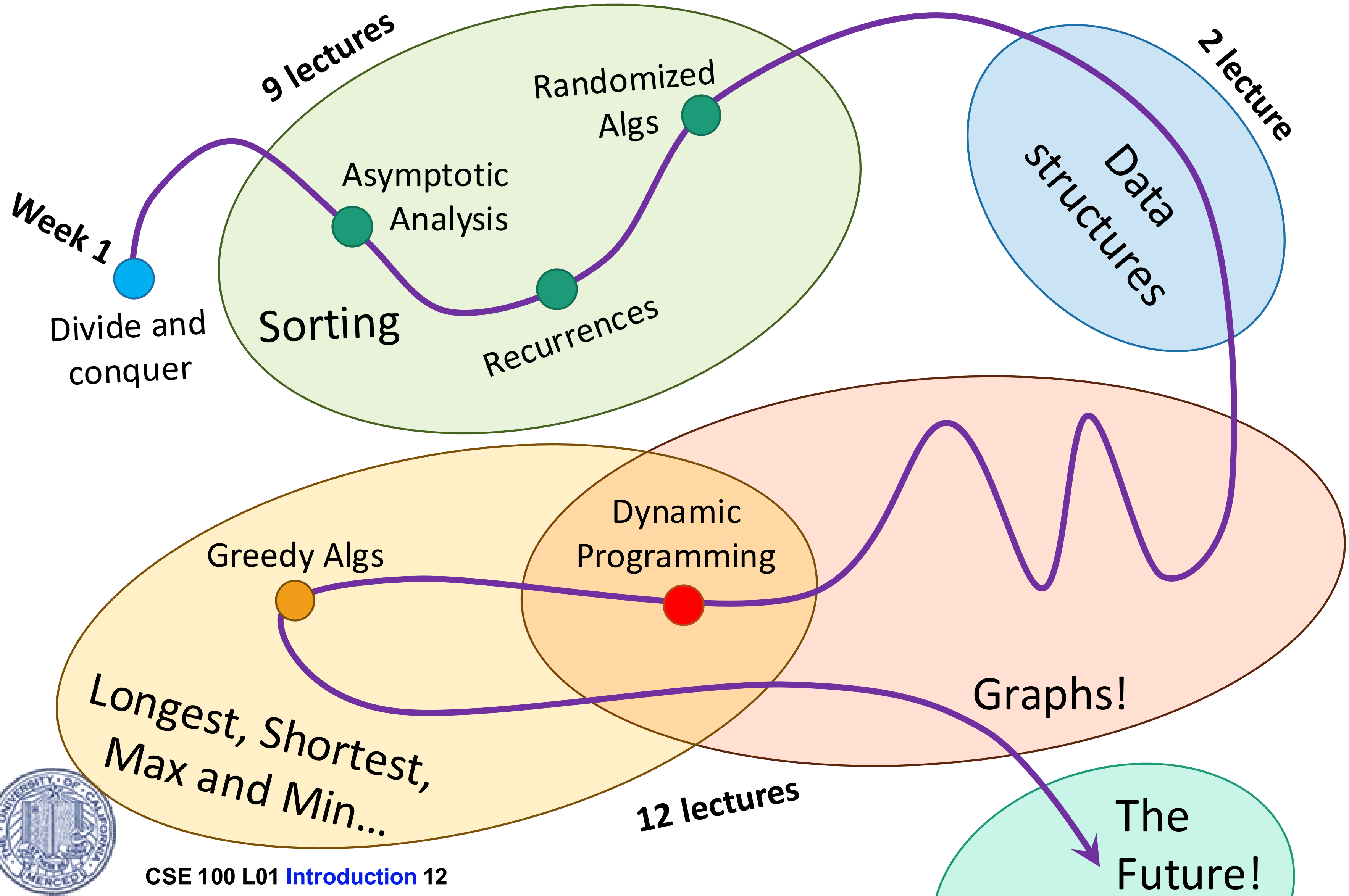
- Course goals/overview
- Logistics

Course goals

- The design and analysis of algorithms
  - These go hand-in-hand

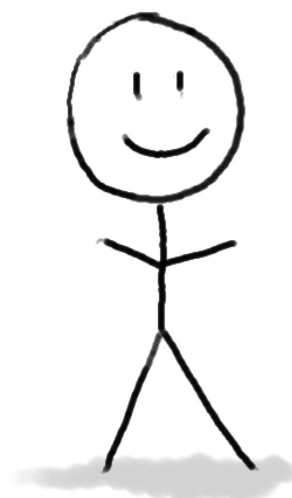
- In this course you will:
  - Learn to think analytically about algorithms
  - Flesh out an "algorithmic toolkit"
  - Learn to communicate clearly about algorithms

Roadmap


9 lectures
Randomized Algs
Asymptotic Analysis
Week 1
Divide and conquer
Sorting
Recurrences
2 lecture
Data structures
Dynamic Programming
Greedy Algs
Graphs!
Longest, Shortest, Max and Min…
12 lectures
The Future!

Our guiding questions:

Does it work?
Is it fast?
Can I do better?



Algorithm designer

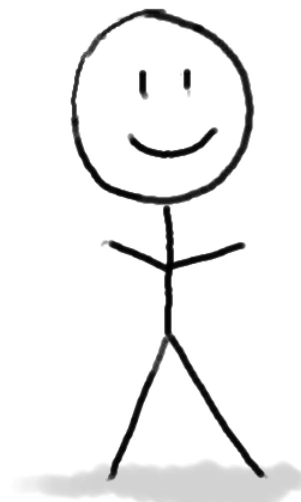
Our internal monologue…

What exactly do we mean by better? And what about that corner case? Shouldn't we be zero-indexing?

Plucky the
Pedantic Penguin

Detail-oriented
Precise
Rigorous

Does it work?
Is it fast?
Can I do better?



Dude, this is just like that other time. If you do the thing and the stuff like you did then, it'll totally work real fast!

Lucky the
Lackadaisical Lemur

Big-picture
Intuitive
Hand-wavey

Both sides are necessary!

We will feel this tension throughout the course

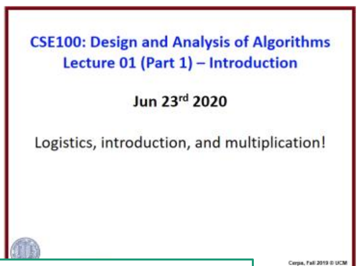
- In lecture, I will channel Lucky in presenting ideas.
- On Problems and Labs, you should lean a bit more towards Plucky.

Course elements and resources

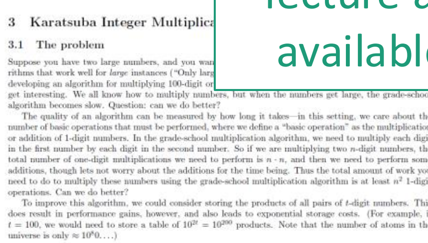
- Course website: CatCourses
- Piazza
- Lectures and Notes
- Textbook
- Lab Discussions
- Lab Assignments
- Exams
- Office hours (posted by end of Week to Syllabus) or meetings by appointment.

Lectures

- Resources available:
  - Slides, Notes, Book, Discussion, Lab Assignments



Slides are the slides from lecture.



3 Karatsuba Integer Multiplica

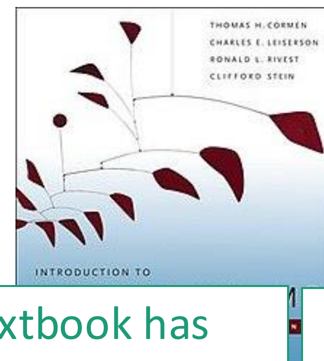
3.1 The problem

Suppose you have two large numbers, and you wan rithms that work well for *large* instances ("Only larg developing an algorithm for multiplying 100-digit or get interesting. We all know how to multiply numbers, but when the numbers get large, the grade-school algorithm becomes slow. Question: can we do better?

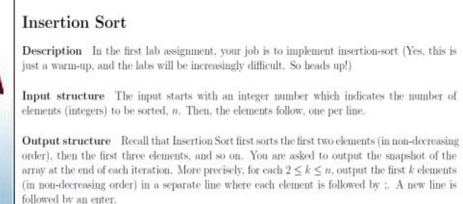
The quality of an algorithm can be measured by how long it takes—in this setting, we care about the number of basic operations that must be performed, where we define a "basic operation" as the multiplication or addition of 1-digit numbers. In the grade-school multiplication algorithm, we need to multiply each digit in the first number by each digit in the second number. So if we are multiplying two n-digit numbers, the total number of one-digit multiplications we need to perform is $n \cdot n$, and then we need to perform some additions, though lets not worry about the additions for the time being. Thus the total amount of work you need to do to multiply these numbers using the grade-school multiplication algorithm is at least n^2 1-digit operations. Can we do better?

To improve this algorithm, we could consider storing the products of all pairs of t-digit numbers. This does result in performance gains, however, and also leads to exponential storage costs. (For example, if $t = 100$, we would need to store a table of $10^{2t} = 10^{200}$ products. Note that the number of atoms in the universe is only $\approx 10^80 \ldots$.)

Notes from lecture are available!



Textbook has mathy details that slides may omit



Insertion Sort

Description In the first lab assignment, your job is to implement insertion-sort (Yes, this is just a warm-up, and the labs will be increasingly difficult. So heads up!)

Input structure The input starts with an integer number which indicates the number of elements (integers) to be sorted, n. Then, the elements follow, one per line.

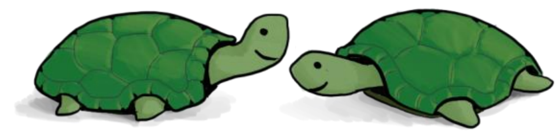
Output structure Recall that Insertion Sort first sorts the first two elements (in non-decreasing order), then the first three elements, and so on. You are asked to output the snapshot of the array at the end of each iteration. More precisely, for each $2 \le k \le n$, output the first k elements (in non-decreasing order) in a separate line where each element is followed by ;. A new line is followed by an enter.

Discussion and Lab Assignments have implementation details that slides may omit.

- Goal of lectures:
  - Hit the most important points of the week's material
    - Sometimes high-level overview
    - Sometimes detailed examples

How to get the most out of lectures

- **During lecture:**
  - Show up, ask questions.
  - Engage with in-class questions.
- **Before lecture:**
  - Do ***the reading suggested*** on the website.
- **After lecture:**
  - Go through the exercises on the slides.



Think-Pair-Share Terrapins (in-class questions)

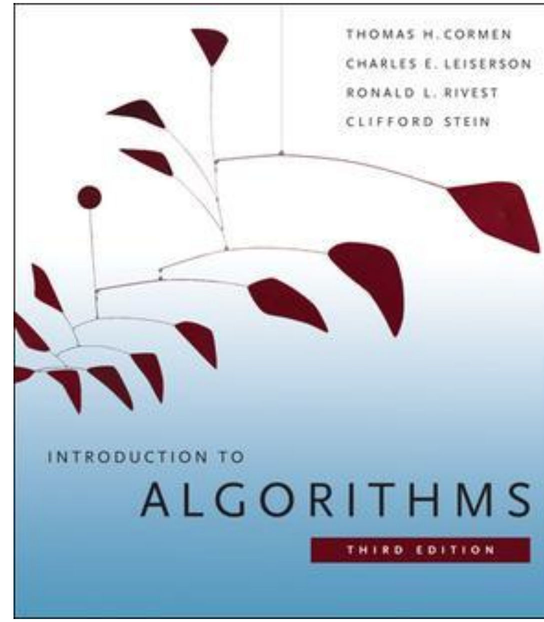
Siggi the Studious Stork (recommended exercises)

Ollie the Over-achieving Ostrich (challenge questions)

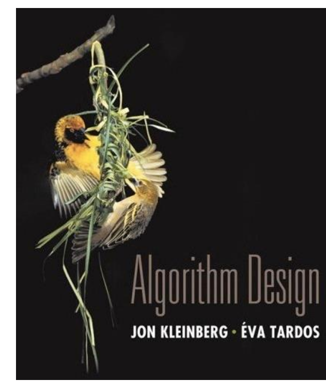
- ***Do the reading***
  - either before or after lecture, whatever works best for you.
  - **do not wait to “catch up” the week before the exam.**

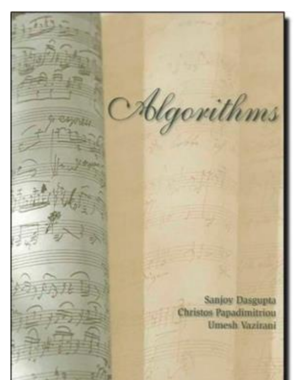
Textbook(s)

- **CLRS**:
  - Introduction to Algorithms, by Cormen, Leiserson, Rivest, and Stein.
- **Notes**:
  - We will provide summary notes for many lectures
  - They are not intended to replace the book!

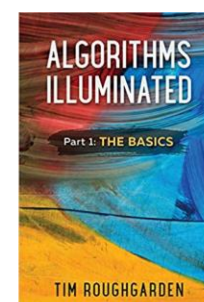


We will also sometimes refer to “Algorithm Design” by Kleinberg and Tardos or "Algorithms" by Papadimitriou, Dasgupta, and Vazirani.





“Algorithms Illuminated” by Tim Roughgarden is also a great reference.



Lab Discussion

Weekly Lab Discussion in two parts:

1. Exercises:
   - Check-your-understanding and computations
   - Should be pretty straightforward
   - We recommend you do these on your own

2. Problems:
   - Proofs and algorithm design
   - Not straightforward
   - You may collaborate with your classmates
     (WITHIN REASON: See website for collaboration policy!)

They will help you prepare for the midterms

Lab Assignments

- Lab assignments are algorithm implementations in C++.
  - You will write C++ code, and test it with test cases.
- You will need to learn ***some*** C++ if you don't know.
  - For next week, the ***Lab Assignment 00*** is to get started with lab assignments and the grader system.
  - See course website for details (will be posted this week).

- The goal is to make the algorithms (and their runtimes) more tangible.
- It is not the goal to become a super C++ programmer.
  - (Although if that happens that's cool).

How to get the most out of discussions

- Do the exercises on your own.
- Try the problems on your own **before** talking to a classmate.
- If you get help from TA:
  - **Try the problem first**.
  - Ask: **"I was trying this approach and I got stuck here."**
  - After you've figured it out, **write up your solution from scratch**, without the notes you took during office hours.

Exams

- There will be 4 **midterms** and a **final**. All during class.
- We will release discussion solutions before each.
- Cannot be rescheduled. Your lowest midterm score will be dropped.

Course elements and resources

- Course website: CatCourses
- Piazza
- Lectures and Notes
- Textbook
- Lab Discussions
- Lab Assignments
- Exams
- Office hours (by appointment or [illegible] sessions)

Bug bounty! 

- We hope all slides and notes will be bug-free.
- However, I sometmes maek typos.
- **If you find a typo** (that affects understanding*) on slides, notes, discussion or lab assignments:
    - Let us know! (Piazza post).

Bug Bounty Hunter

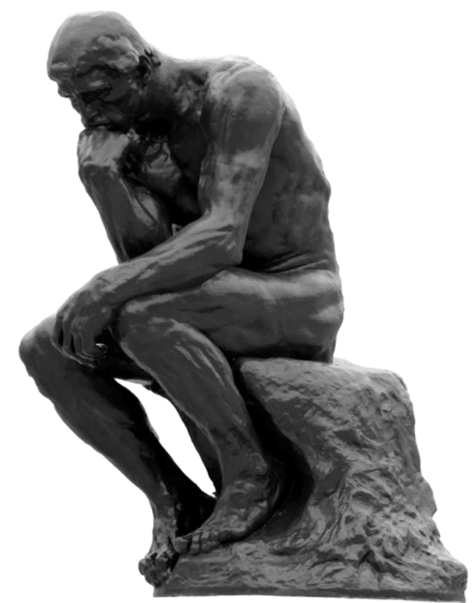
*So, typos lke thees onse don’t count, although please point those out too. Typos like 2 + 2 = 5 do count, as does pointing out that we omitted some crucial information.

A note on course policies

- Course policies are listed on the website.
- Read them and adhere to them.
- That’s all I’m going to say about course policies.

The big questions

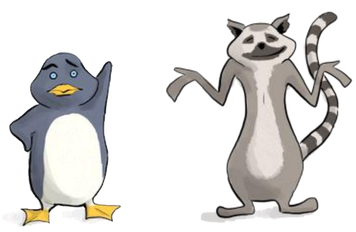
- Who are we?
  - Professor, TA's, students?
- Why are we here?
  - Why learn about algorithms?
- What is going on?
  - What is this course about?
  - Logistics?
- Wrap-up



Wrap up

- Make sure you are on CatCourses and Piazza.
- Algorithms are fundamental, useful and fun!

- In this course, we will develop both algorithmic intuition and algorithmic technical chops



Next time

- Karatsuba Integer Multiplication
- Algorithmic Technique:
  - Divide and conquer
- Algorithmic Analysis tool:
  - Intro to asymptotic analysis