



Lecture 40

Wrapping Things Up

CS61B, Spring 2024 @ UC Berkeley

Slide Credit: Josh Hug

61B in 12 Minutes or Less

Lecture 40, CS61B, Spring 2024

61B in 12 Minutes or Less

Your Future

AMA

Course Reflections

Where We Started

Just 14 weeks ago:

[61B FA23] Lecture 1 - Intro to 61B, Java

```
1 x = 0
2
3 while (x < 10):
4     print(x)
5     x = x + 1
```



Press Esc to exit full screen

```
1 public class HelloNumbers {
2
3     int x;
4     x = 0;
5     while (x < 10) {
6         System.out.println(x);
7         x = x + 1;
8     }
9 }
10 }
```

```
0
1
2
3
4
5
6
7
8
9
```

I

What We've Learned about Programming Languages

- Object based programming: Organize around objects.
 - Object oriented programming:
 - Interface Inheritance. 
 - Implementation inheritance.
 - Dynamic vs. static typing.
 - Generic Programming, e.g. ArrayList<Integer>, etc.
 - The model of memory as boxes containing bits. 
 - Bit representation of positive integers.
 - Java.
 - Some standard programming idioms/patterns:
 - Objects as function containers (e.g. Comparators, [IntUnaryFunctions](#)).
 - Default method specification in interfaces ([Link](#)).
 - Iterators.
- Example: Programmer only needs to know List API, doesn't have to know that ArrayList secretly does array resizing.**
- Example: Array is a sequence of boxes. An array variable is a box containing address of sequences of boxes.**

Important data structure interfaces:

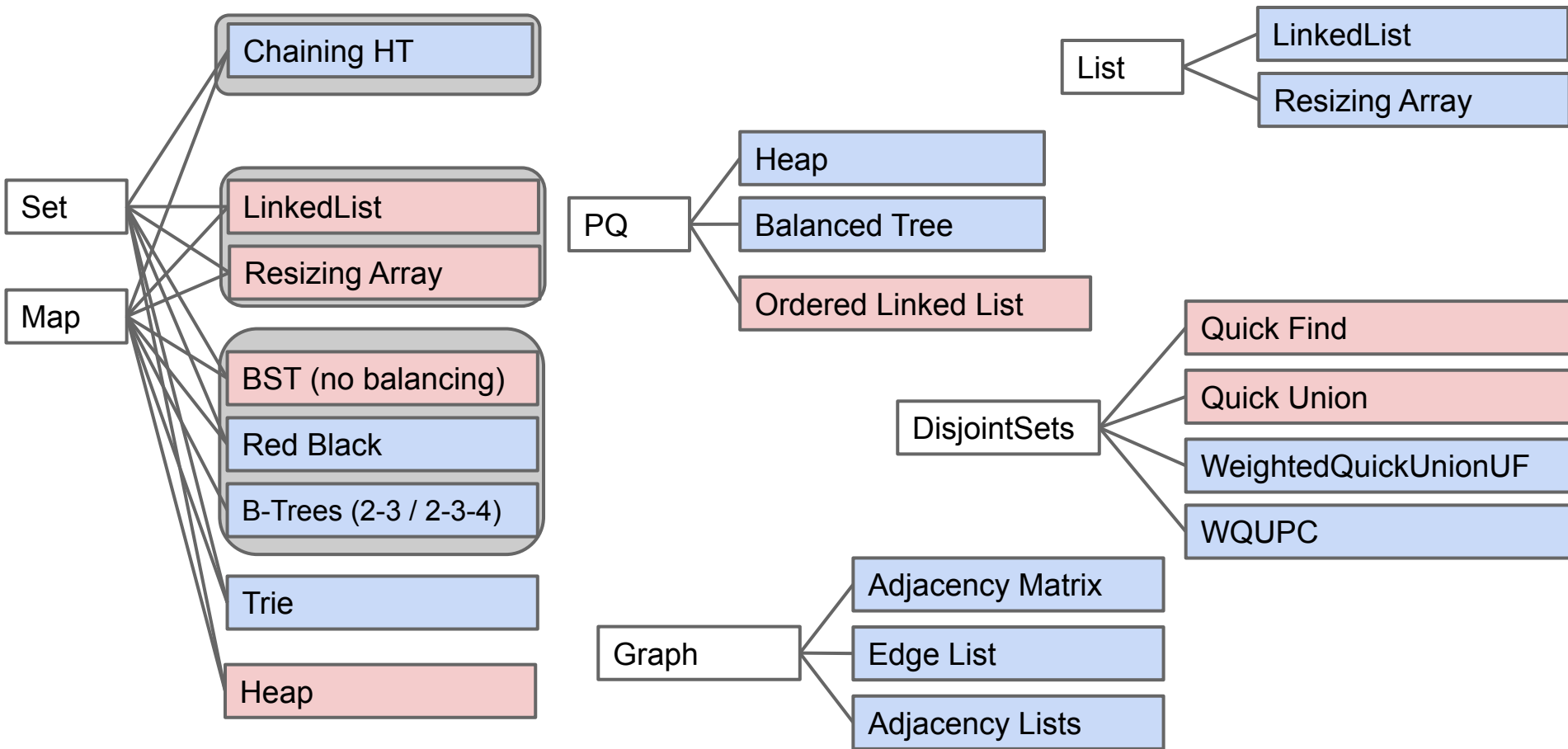
- `java.util.Collection` (and its subtypes).
 - With a special emphasis on `Map` (and its subtypes).
- Our own Collections (e.g. `Map61B`, `Deque`): Didn't actually extend `Collection`.

Concrete implementations of these abstract implementations:

- Examples: `ArrayDeque` implements `Deque`.

- Asymptotic analysis.
- $O(\cdot)$, $\Omega(\cdot)$, $\Theta(\cdot)$, and tilde notation.
- Worst case vs. average case vs. best case.
 - Exemplar of usefulness of average case: Quicksort
- Determining the runtime of code through inspection (often requires deep thought).
- Multiplicative resizing is much better than additive resizing.
 - Multiplicative resizing results in $\Theta(1)$ add runtime for ArrayLists.
 - Additive resizing results in $\Theta(N)$ add runtime for ArrayLists.

Some Examples of Implementations for ADTs



Array-Based Data Structures:

- ArrayLists and ArrayDeque
- HashSets, HashMaps, MyHashMap: Arrays of 'buckets'
- ArrayHeap (tree represented as an array)

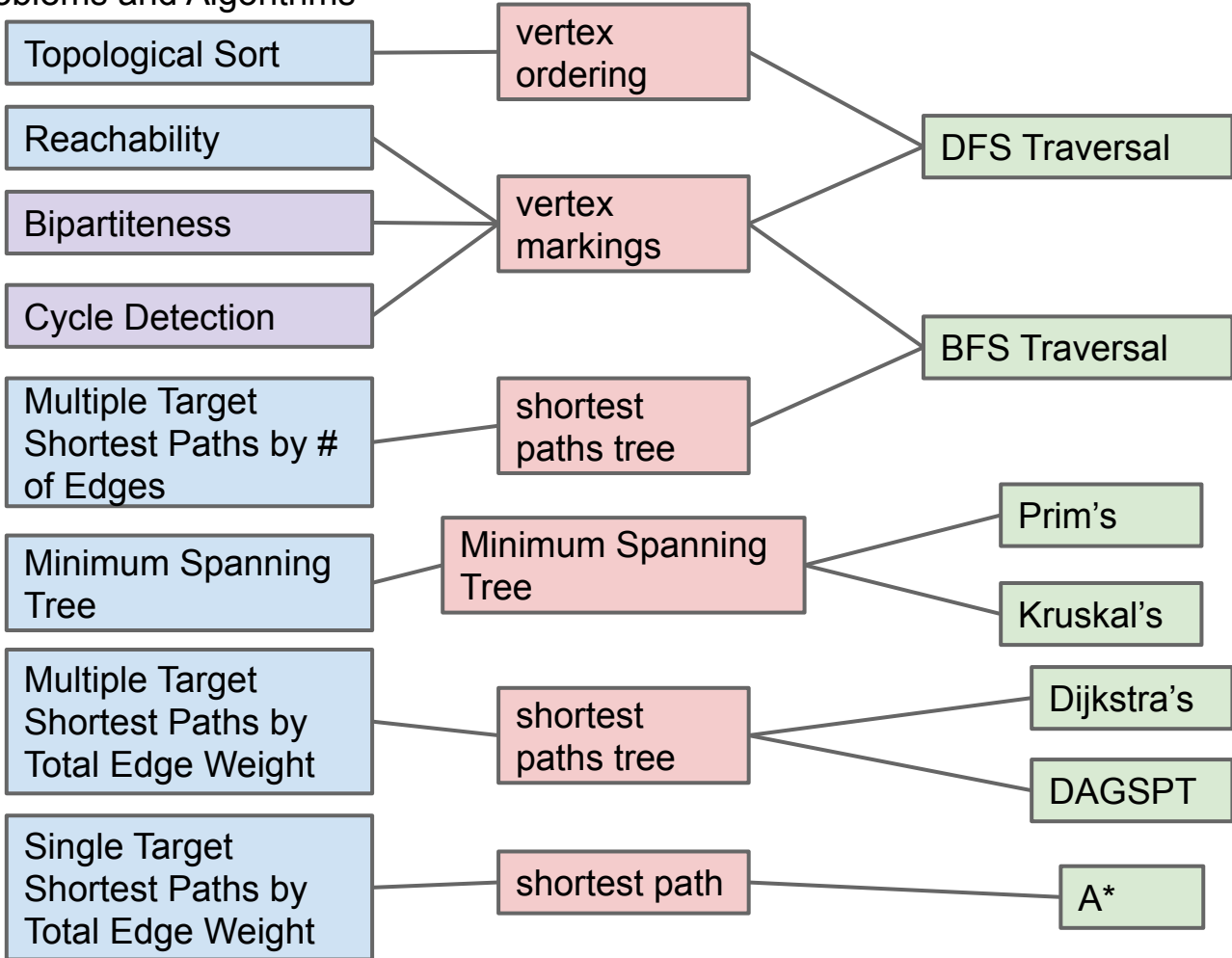
Linked Data Structures

- Linked Lists
 - LinkedList, IntList, LinkedListDeque, SLList, DLList
- Trees: Hierarchical generalization of a linked list. Aim for bushiness.
 - TreeSet, TreeMap, BSTMap, Tries (trie links often stored as arrays)
- Graphs: Generalization of a tree (including many algorithms).

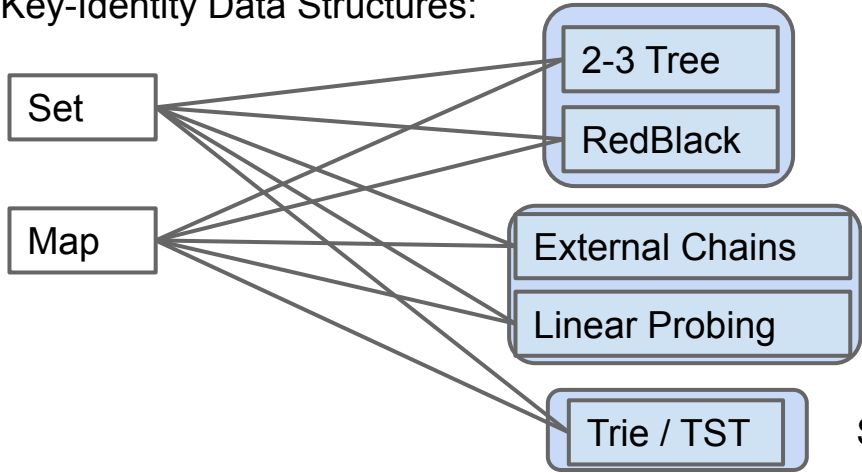
Tradeoffs of array-based vs. linked data structures.

Tractable Graph Problems and Algorithms

Discussed
in section



Search-By-Key-Identity Data Structures:

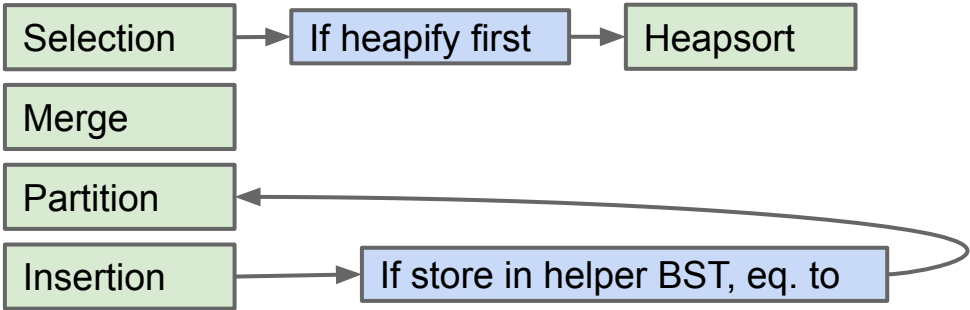


Searches using compareTo()
Analogous to **Comparison-Based**

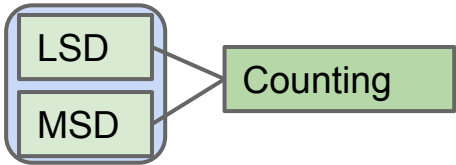
Searches using hashCode() and equals()
Roughly Analogous to **Integer Sorting**

Searches by digit. Analogous to **Radix Sorting**

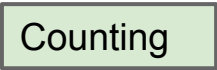
Comparison Based Sorting Algorithms: $\Omega(N \log N)$ worst case



Radix Sorting Algorithms: $\Theta(NW)$ worst case:
(require a sorting subroutine)



Small-Alphabet (Integer) Sorting Algorithms: $\Theta(N)$ worst case



Compression:

- Huffman Coding, and selection of data structures for Huffman Coding.
- Other approaches: LZW and Run Length Encoding (extra slides).

Compression, Complexity, and $P=NP$:

- Optimal compression is impossible.
- Bounded space/time compression is possible if $P=NP$.
 - Allows you to find arbitrarily short programs that produce a given output in a given runtime.
- If $P = NP$, we could also solve all kinds of other interesting problems.

- Java syntax and idioms.
- Unit testing (and its more extreme form: Test-driven development).
- Mining the web for code.
- Debugging:
 - Identify the simplest case affected by the bug.
 - Hunt it down, giving it no place to hide.
 - With the right methodology, can find bugs even when finding bug through manual code inspection is impossible.
- Tactical vs. Strategic Programming.
 - Avoid information leakage. Build deep modules. Minimize dependencies. Minimize obscurity.
- Real tools: IntelliJ, git, Truth, command line.
- Data structure selection (and API Design)
 - Drive the performance and implementation of your entire program.

Your Future

Lecture 40, CS61B, Spring 2024

61B in 12 Minutes or Less

Your Future

AMA

Course Reflections

Two immediate questions:

- What classes should I take next semester?
- How should I prepare for the final?

Two long-term questions:

- What am I capable of achieving?
- What should I do with my life?

Class Poll (Your Answer)

How many students are happy with the grade they are on track to receive?

What does a good grade in this class mean? How will a good grade in this class affect you?

- You understood the concepts in the class
- You can apply the concepts in this class in new and innovative exams
- Good grade -> leads to becoming a TA/jobs/Graduate School
- Lets you declare the major
- Proof of hard work
- Different number on resume

What does a good grade in this class mean? How will a good grade in this class affect you?

A grade in this course is not a measure of your worth/intelligence.

Several ways your grade could affect your future:

- Future job prospects/grad school?
 - While GPA is a factor, it's often not a very important one. Once you get into industry, job experience matters much more than college GPA.
- GPA requirement to enter the CS major
 - Unfortunately, this is a big determinator of getting into the major. Every faculty member and lecturer hates the GPA requirement because it incentivizes the wrong way to think about your grades
 - Fortunately, students admitted after FA23 won't have that GPA requirement.

What does your grade mean?

I would argue that the most important thing your grade tells you is **how hard your next semester will be.**

Most of the time, you don't truly master a concept until you're forced to use it to solve something new. A lot of the concepts we've covered in the last half of the course will only solidify when you start taking 61C, 170, or any other "next step" class.

The grade you get in this course is my message to you saying how difficult I predict it will be to develop that mastery in your next class.

What does your grade mean?

An A in this class is a signifier of **mastery**. If you take a sequel course, **you will likely do well even with other work on your plate**. So you can take more/harder courses.

A B in this class is a signifier of **comprehension**. If you take a sequel course, **you will likely do well if you dedicate sufficient time to the class**.

A C in this class is a signifier of **competence**. If you take a sequel course, **you will likely need to devote more time than usual to do well**. So try to avoid taking too many courses at the same time

Anything below a C is a signifier that **you will struggle in a sequel course**. Going directly to a sequel course will likely not be worthwhile. It may be good to retake the course so you get more time to digest concepts.

- Like surfing: Once you're behind a wave, it's difficult to climb it
 - But if you're ahead of the wave, it pushes you forward
 - If you're behind the wave, it may be best to wait for the next wave

A Supporting Experiment for “Everyone Can Do Well”

In Sp16, Josh gave students the option to fail intentionally to retake the course.

- On average, students were 2.5 letter grades higher the second time, e.g. someone in the middle of the B range got an A the second time.

My interpretation: Even if you're not on pace for the grade you want, **you have learned a lot from this course**. That foundation will make it much easier to relearn the course material at a later time.

Bottom line: You're capable of achieving more than you might think possible.

The core "goal" of 61B

Ultimately, 61A's main goal was to teach you *how* to program:

- Go from problem description to a code solution

61B's goal is to teach you *why* you program:

- You have a problem, and you need to decide that code is the best way to solve that problem

61C and 170 mainly focus on making the code you write *efficient* (170 by focusing on the theoretical algorithm design, and 61C by focusing on the reality of how a physical computer works).

After that, most classes in the CS major focus on one tiny facet of CS (specialization)

At this point, you have everything you need to build whatever you want from code.

The best way to learn CS from now is to make something by yourself, and to play with code.

How to have fun with CS: Competitive Programming and CTFs (Links in Speaker Notes)



Project Euler.net



the cryptopals crypto challenges



ZACHTRONICS

TIS-100
TESSELLATED INTELLIGENCE SYSTEM

 **Bombe**

How to have fun with CS: Hackathons (<https://www.calhacks.io/>)



[Home](#) [About](#) [Inclusion](#) [Sponsor Us](#)

[Contact Us](#) [Other Initiatives](#)



Cal Hacks 11.0

We are a team of students with a common goal – cultivating a platform where people can build, learn, and experiment without boundaries.

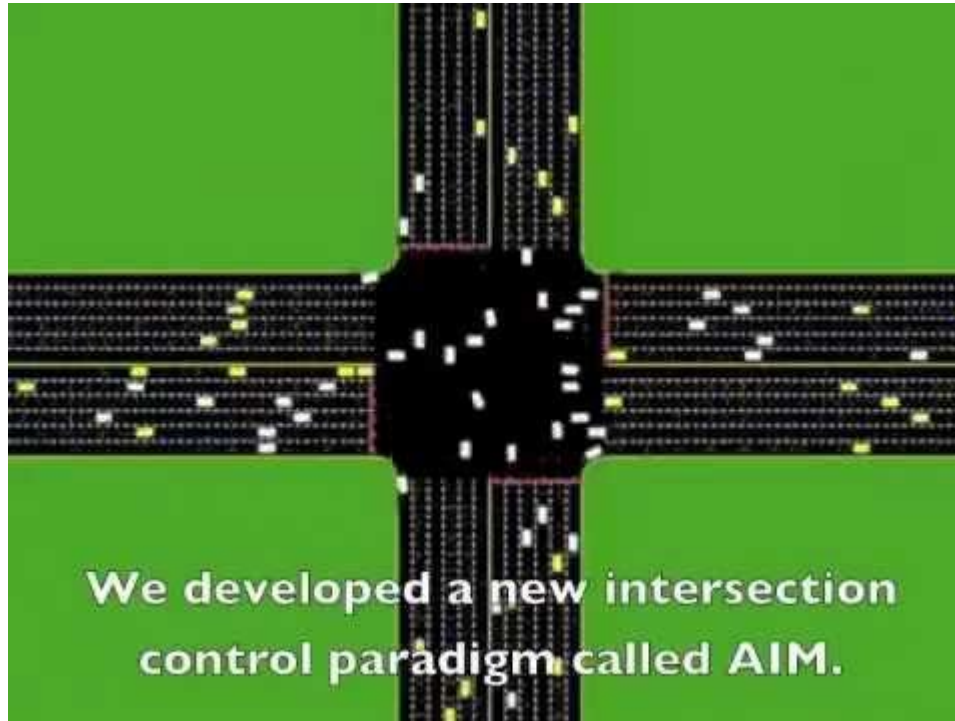
Fill out our **early interest form** and be the first to hear about our next event!

Two long-term questions:

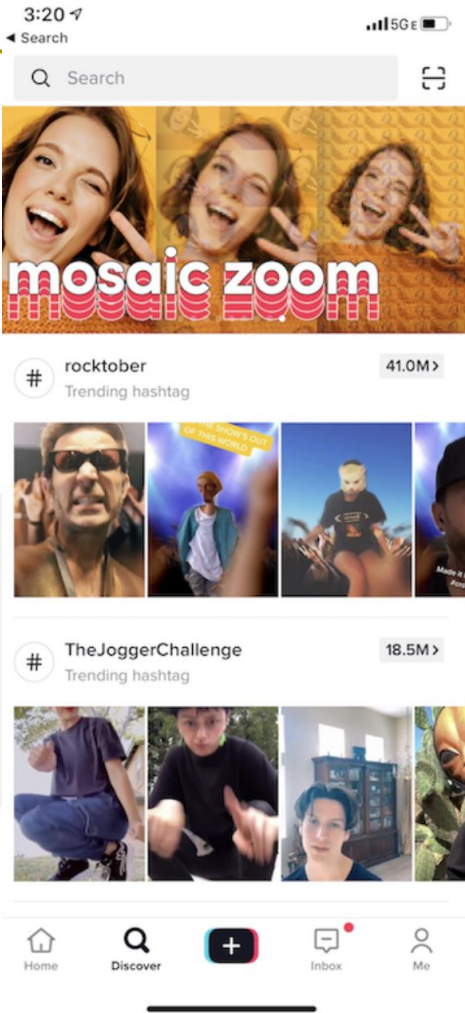
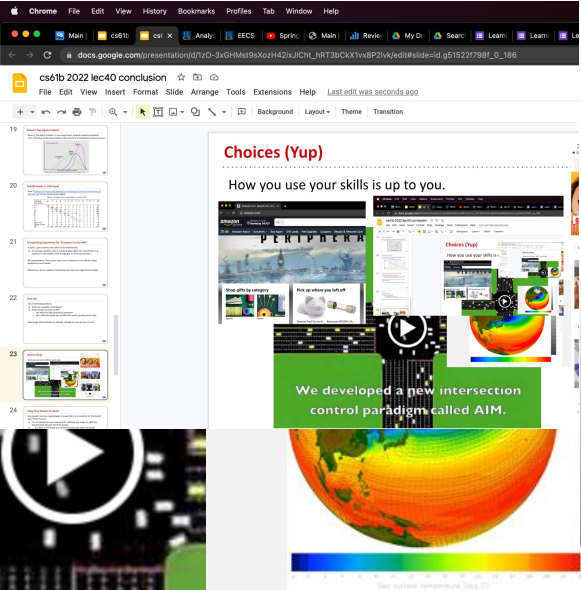
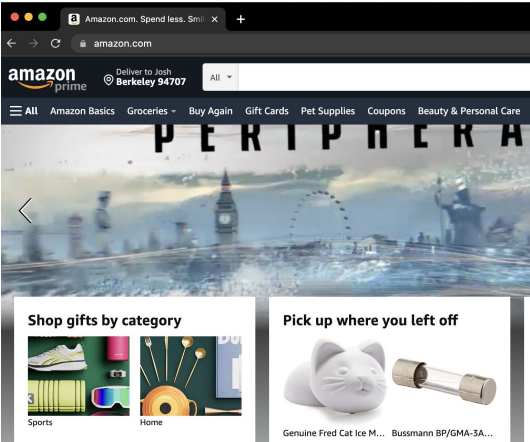
- What am I capable of achieving?
 - Almost anything given time and motivation
- What should I do with my life?
 - You will be in high demand by everyone.
 - Your skills will enable you to affect the world, possibly profoundly.

Technology will continually to radically reshape the way we live our lives.

How you use your skills is up to you.



How you use your skills is up to you.



My request: Use your superpowers in a way that is a net positive for the lives of your fellow humans.

- I'm not saying that you must work for relatively low wages to uplift the downtrodden (though that'd be great, of course).
- but keep in mind that your work will profoundly affect the world.

(Wanna talk about this stuff more? Take CS195!)

AMA

Lecture 40, CS61B, Spring 2024

61B in 12 Minutes or Less

Your Future

AMA

Course Reflections

Questions About Anything for Peyrin and Me?

Course Reflections

Lecture 40, CS61B, Spring 2024

61B in 12 Minutes or Less

Your Future

AMA

Course Reflections

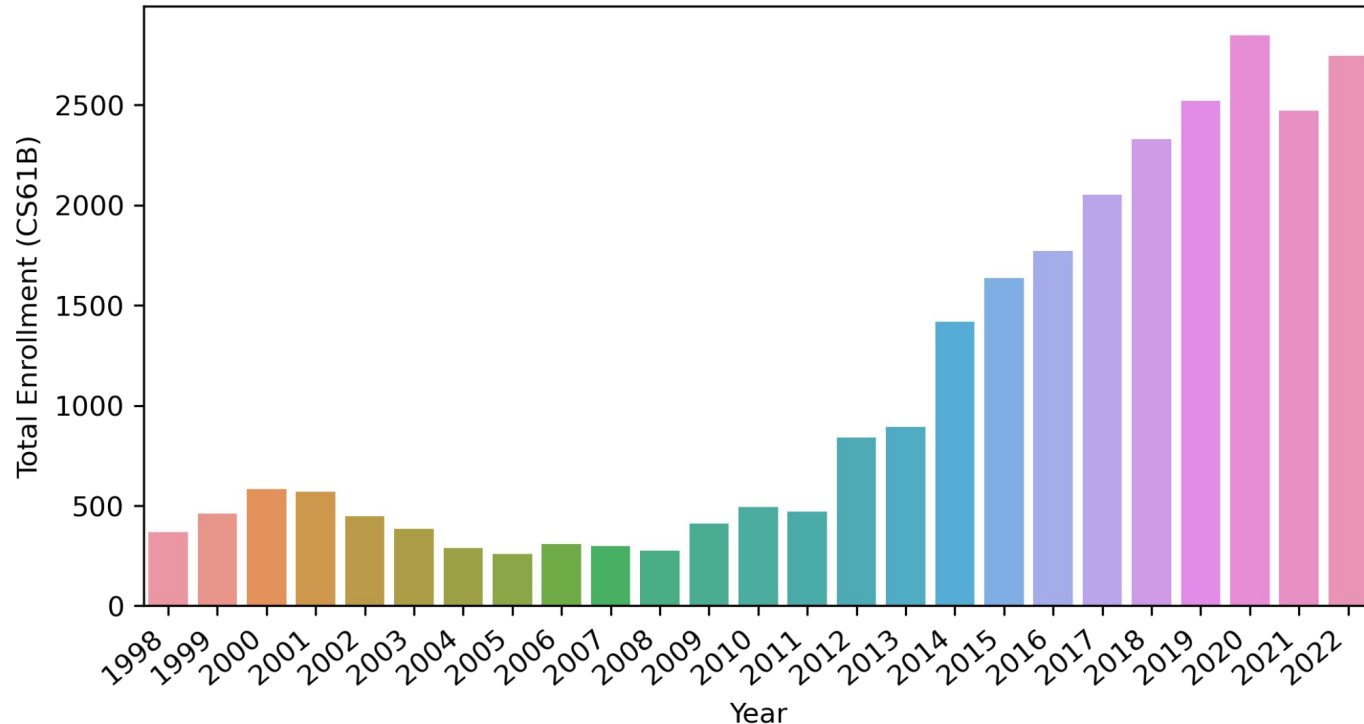
We read all of your feedback, especially on the end of course evaluations.

- There is an official form from the university, and an internal survey for 61B that we'll send out ourselves.

Course evaluations factor heavily into lecturer reviews, and let us know what bits of the class we can improve.

61B Would Love to Have You Next Fall

- Academic interning: Learn more, help others, get units.
- Watch out for an announcement on the Spring 2024 61B Ed.



Special Thanks to the Staff (who keep the wheels on the bus)

Closing Thoughts (Peyrin)

Closing Thoughts

It is a terrifying and awesome responsibility to decide how you should spend hundreds of your precious hours here at Berkeley.

Don't worry about how others are doing, or focus on perfectly optimizing your college experience. Chances are, the choices you make now will lead to a good outcome. All you need to do is to be better than the person you were a year ago.

Personally, I teach because I believe that I am doing the most good for the world in this position. Please, prove me right.