Welcome to CS50 section! This is Week 3.

Please open your CS50 IDE and run this in your console:

```
cd ~/workspace/cs50-section ↵
git reset --hard ↵
git pull
```

If new to this section, visiting, or want to “start over”, run this in your console:

```
rm -r -f ~/workspace/cs50-section/ ↵
cd ~/workspace ↵
git clone https://github.com/bw/cs50-section.git
```

Fun fact! Banging your head against the wall burns 150 calories an hour.
(This will come in handy later on in the term. Just kidding!)

Cumulative concepts for this week

- **Arrays**
- debug50
- **Asymptotic notation**
  - **O and sigma**

- **Linear search**
- **Binary search**

- **Bubble sort**
- **Insertion sort**
- **Selection sort**
- **Merge sort, in theory only**

- **Recursion**

- Using distribution code

More introductions!

Arrays, revisited

- For all data types

- Be very comfortable with common array operations
  - Indexing into an array (i.e. get each element one at a time)
  - Comparing across elements of an array
  - Making changes to an array

- Only strings need `\0` at the end
  - Why?

Asymptotic notation

- Big O notation
  - Describes an upper bound on algorithm run time

Asymptotic notation

- Big O notation
  - Describes an upper bound on algorithm run time

In this notation we simplify and ignore lower-order terms:

- Ignore all constants. Why?
- If $x^2 + x$, ignore x. Why?
- If $x^3 + x^2 + x$, ignore $x^2 + x$. Why?
- If x log(x), leave as is. Why?

Asymptotic notation

- Little O notation (□)
  - Describes lower bound on algorithm run time
  - Think of as, what's the best case scenario?

- Think about the algorithm *and* the implementation

Asymptotic notation

Think about these actions:

- Swaps
- Comparisons
- For loops

Searching

- Linear search
  - Prerequisites
  - Benefits
  - Disadvantages
  - Upper bound
  - Lower bound
- Binary search
  - Prerequisites
  - Benefits
  - Disadvantages
  - Upper bound
  - Lower bound

Before we talk about bubble sort...

Let's talk about break.

Before we talk about bubble sort...

Let's talk about break.

- We are iterating (i.e. using a for loop)
- We're checking for something
- We want to stop iterating

Before we talk about bubble sort…

Let's talk about break.

```
for (int i = 0; i < n; i++) {
  …

  if (we should stop) {
    break;
  }
}
```

More about break

A very useful concept.

- Works in for and while loops
- Will always jump out of the inner-most loop

Bubble sort (pset 3)

Let’s craft the pseudo code for bubble sort.

Bubble sort is on pset 3.

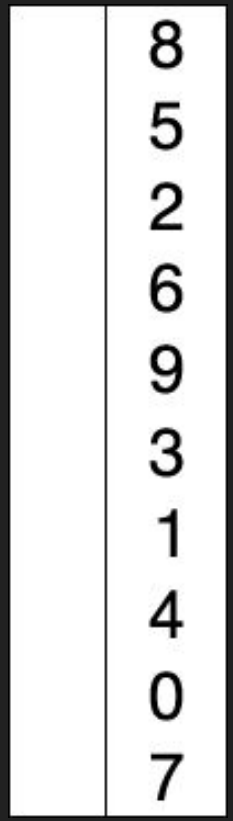
Bubble sort (pset 3)

What are the nuances to consider?

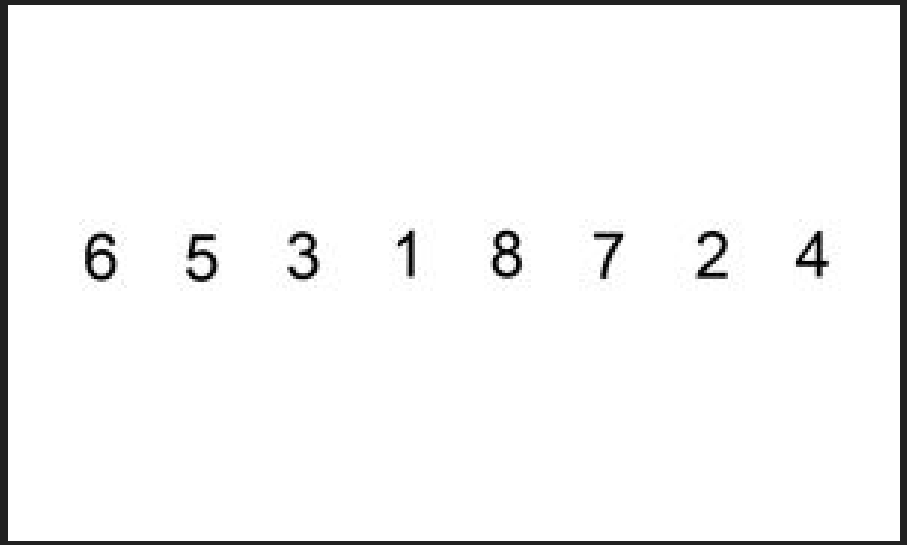
- Is this an efficient implementation of bubble sort?
  - How do you know when to stop sorting?
  - How many loops are you doing?

Hint: You'll lose design (and potentially correctness) points if your code always runs with the worst-case scenario in mind.

Selection sort



Insertion sort



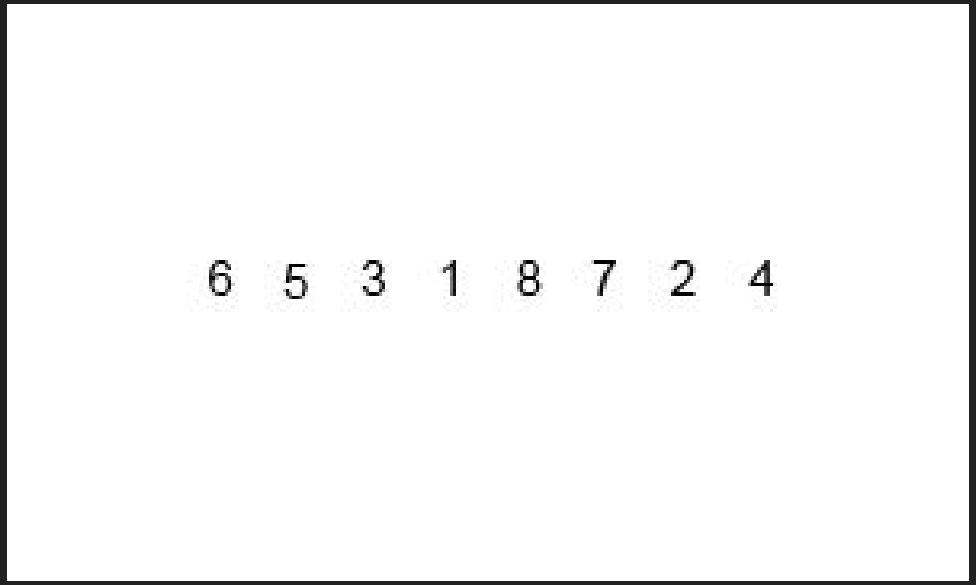
Merge sort

For purposes of section, understand--

- Divide and conquer

- Sort the left
- Sort the right
- Put them together
  - Look through, from the leftmost element
  - Which one is smaller? Grab that one first
  - Rinse and repeat

Merge sort



Merge sort

Let’s craft the *basic* pseudo code for merge sort.

- This can get complicated, so we’ll keep things simple.
- You’ll probably be tested on merge sort.

Recursion

More on that “divide and conquer”

What is recursion?

Recursion

All recursions have--

- Base case
  - This is the end
- Recursive case
  - Do it again!

Recursion

- Recursion has upsides
  - Beautiful code
  - Sometimes easier to understand

- Recursion has downsides
  - Can be memory-intensive
  - Can be harder to understand (bummer…)

Recursion

An exercise! Let’s write a recursive function that finds the Fibonacci number given the number of terms.

Load up your IDE!

Using distribution code

- Much of computer science involves wrangling other people's code
  - With all of their idiosyncrasies, annoyances, etc.
  - Get used to it!

- In CS50, the code is generally written pretty well
  - So think about what your piece is contributing and how

Problem set 3

I can’t say too much here! :(

But, thinking about this pset broadly,

- Do the pset in chunks, not all at once
- Game of fifteen
  - What are the allowed moves?
  - What happens during each move?
  - How do you check if the user has won?

That’s all for today!