1 | Why do we sort?

Super important problem, lot’s of applications

1.1 | Problems where sorting *is* the problem

- Organizing a library
- Sorting directories
- Leader boards, tables, and rankings

1.2 | Problems where sorting makes it fast

- Calculating median
- Finding closest pair
- Binary searching

1.3 | Other stuff

- Data compression: sorting makes duplicate removal easy
- Lazy scene rendering: sorting by z-index and figure out what to render

2 | Basic Sorting Algorithms

2.1 | Insertion Sort

Compare each thing until we find where it goes, until we can find that spot.
Take, for instance, array [4,2,6,3]; we will go through and create sorted sub-arrays.
[4*,2,6,3]
We take the pointer at 4, we notice 2 < 4, we swap and we move our pointer rightwards
[2,4*,6,3]
We now notice that 6 > 4, we simply move our pointer rightwards.
[2,4,6,3*]
We then compare 3 < 6, we swap:
[2,4,3*,6]
We then compare 3 < 4, we swap:
[2,3*,4,6]
We then check rightwards, and then return sorted list
[2,3,4,6]

2.1.1 | **Properties**

- Adaptive: fast when sub-array is already sorted
- Stable: ties will be always sorted the same way — earlier tied values will be returned earlier
- In place: requires O(1) of memory
- Online (can work with constantly new input to sort)

2.2 | **Counting Sort**

Take our input, and then find minimum and maximum. Then, we will count how many of each element there were. Then, we create a whole new array with the same of the original length, then dump the sorted array out.

This is *not* a comparison based algorithm: you never needed to compare values—you are simply counting. We need to figure out all the buckets of integers, and figure a reference key of buckets. This is, of course, not memory efficient.

Furthermore, this algorithm does *not* work for floats or anything but clear integers. To address this, we introduce Radix sort.

2.3 | **Radix Sort**

Radix sort is creating buckets with a stem and leaf plot and sorting by counting sort.

We first sort by the first digit via counting sort: given we are only sorting 10 buckets of possible digits, its going to work.

Then, we sweep towards the left, and sort the larger digits sequentially.