SORTING

Input: a list of numbers  $8, \sqrt{2}, 5, 21, \frac{1}{3}$

Output: a permutation that is in increasing order  $\frac{1}{3}, \sqrt{2}, 5, 8, 21$

An algorithm should do the following:

1) produce correct output for all possible input
2) terminate quickly
3) not use a lot of space
4) be described clearly

etc
Issues that affect algorithmic design:

• input type: e.g., are numbers distinct? integer/real/irrational/etc? Do they have bounded size?

• allowed operations: compare, add, truncate, etc

• data structure: array? linked list? tree? etc

• model of computation: time/space complexity of operations

We focus on:

• comparison-based algorithms

• constant time for basic ops (more later)
Insertion sort

- Start with a sorted prefix of size 1.
- Extend size of sorted prefix by 1
- Repeat

In general:

Before: ![Sorted prefix with an element to be inserted]

After: ![Sorted prefix extended by one element]

Use the element next to the prefix to extend the prefix...
If prefix size = \( j \) then we can insert \( X \) after at most \( j \) comparisons.
\[ \leq j \text{ comparisons} \rightarrow \text{increase the sorted prefix size from } j \text{ to } j+1 \]

Terminate when prefix size = \(n\) (entire array)

\[
\text{comparisons} \leq \sum_{j=1}^{n-1} j = 1 + 2 + 3 + \cdots + (n-1) = \frac{n(n-1)}{2}
\]

\[= \frac{1}{2}n^2 - \frac{1}{2}n = \text{worst case comparisons} \]

To actually implement, we need extra time & space

but just a constant amount per comparison e.g., \(5 \cdot \left(\frac{1}{2}n^2 - \frac{1}{2}n\right)\)
At an introductory level:

- we don't focus on whether it takes 1 or 2 or 5 operations to compare, swap & iterate: \(5 \cdot \left(\frac{1}{2} n^2 - \frac{1}{2} n\right)\)

- we don't really care about non-leading terms: \(5 \cdot \left(\frac{1}{2} n^2 - \frac{1}{2} n\right)\)

When \(n\) is HUGE, both of the above are unimportant.

\[
\begin{cases}
\text{see } 5 \cdot \left(\frac{1}{2} n^2 - \frac{1}{2} n\right) \\
\text{interpret } n^2
\end{cases}
\]

This leads to \(\Theta\)-notation aka big-O notation