What is sorting?

• Input: sequence of \( n \) comparable values

• Reorder the input to be non-descending.

• Items we wish to sort are called “keys”

• Not here: retain associated information

Why discuss sorting?

• Standard library has sorting

• Why not use that and move on?

In this class, sorting is:

• a good intro for techniques

• a good intro to comparative algorithms

**SelectionSort**

**Idea:** Swap min into first spot, second-min to second, etc.

```plaintext
for i ← 1 to n - 1 do
    min ← i
    for j ← i + 1 to n do
            min ← j
        end if
    end for
    Swap A[i] and A[min]
end for
```

Let’s talk about SelectionSort.

• Does it waste memory?

• Does it only work for numbers?

• What other info do we need?

• Are there inputs that are sorted faster?

• Is there a lot of data movement?
Bubble Sort

**Idea:** Think globally act locally

```plaintext
for i ← 1 to n − 1 do
    for j ← 1 to n − i do
            Swap A[j] and A[j + 1]
        end if
    end for
end for
```

InsertionSort

```plaintext
for j ← 2 to n do
    key ← A[j]
    i ← j − 1
    while i > 0 and A[i] > key do
        A[i + 1] ← A[i]
        i = i − 1
    end while
    A[i + 1] ← key
end for
```

- What is the worst-case running time of InsertionSort?
- Why is InsertionSort correct?
- What is true *every time* we check the for loop? (including the time we find $j > n$ and stop)
HeapSort

**Idea 1:** Insert all \( n \) elements into an (initially empty) max heap. Then, repeatedly extract and place the maximum element from the heap into the last spot of the vector into which we have yet to place.

How long does this take?

**Idea 2:** Bottom-up heap construction. We know which locations will be leaf nodes.

\[
\begin{matrix}
4 & 1 & 3 & 2 & 16 & 9 & 10 & 14 & 8 & 7
\end{matrix}
\]

**Question:** Once we have the array turned into a max-heap, what do we do? Where do you place the result of a \texttt{remove-max} operation?