### CSE 373: Selection

Miscellaneous
(bits of Chapters 1, 4, 6, 7)

---

### The Selection Problem

**Goal:** Given a list of $n$ numbers, find the $k^{th}$ smallest

**Special Cases:**
- $k = 1$: \textbf{FindMin()}
- $k = n$: \textbf{FindMax()}
- $k = n/2$: the median of the list

Any ideas?
Selection Brainstorming

Which of the data structures that we’ve studied would be appropriate for selection?

- List
- Stack
- Queue
- Tree
- BST
- Hash Table
- Heap

- must be able to store data
- must maintain some sort of ordering information

List-Based Selection

Naive algorithm:

- Insert each element into a second list using InsertSorted()
- Return the element in the kth position
- Running time?

Slightly improved algorithm:

- Store only the k smallest elements seen so far
- Running time?
Tree-Based Selection

Naive Algorithm:
- Insert() all elements into a BST
- Traverse the tree using an in-order traversal
- Count off until we reach the kth element
- Running time?

Improved Algorithm?

Heap-Based Selection

Naive Algorithm:
- Insert() all elements into a min-heap
- Perform DeleteMin() k-1 times
- The next DeleteMin() returns the target value
- Running time?

Improved Algorithm?
Relating Selection and Sorting

If we were to do selections for \( k = 1, 2, \ldots, n \), we would end up with a sorted list
- Running time?

Alternatively, if we were to sort our input list, we could do any selection in \( O(1) \) time
- Running time?

Motivation for Sorting

- Sorted arrays allow us to do binary searches
- They also allow us to do fast selection
- The mode could be computed trivially in \( O(n) \) time if the input was sorted

(but perhaps most importantly…)

- Humans tend to like things in sorted order

How could we use our data structures to sort?
Which would be appropriate? Efficient?