CS: Basic Sorting Algorithms

In Java, total order is specified by compareTo and compare methods.
An inversion is a pair of elements not in order.

Goal of sorting:
- Given a sequence w/ Z inversions
- perform a sequence of ops that reduces inversions to 0.

Selection sort
- Find smallest item
- Swap this item to front and "fix" it.
- Repeat for unfixed items until all items are fixed.

$\Theta(N^2)$

Naive Heap Sort
- ~~Insert~~ all items into a max heap & discard input array. Create output array.
- Repeat N times:
  - Delete largest item from max heap
  - Put largest item at the end of the unused part of the output array.

Runtime:

$O(N\log N)$

Getting items in heap: $O(N\log N)$

Selecting largest: $\Theta(1)$

Removing largest: $O(\log N)$ for each removal.

$O(N\log N) + O(N\log N) + \Theta(N) = O(N\log N)$.

In place Heapsort:
- Bottom-up heapify input array:
  - Sink nodes in reverse level order
  - After sinking, guaranteed that ~~tree~~ rooted at position k is a heap.
- Repeat N times:
  - Delete largest item, swapping root w/ last item in heap.

RT: $O(N\log N) \rightarrow \Theta(N\log N)$ Space complexity: $\Theta(1)$ no extra array.

Mergesort

- Split items into 2 roughly even pieces.
- Mergesort each half
- Merge the 2 sorted halves to form the final result.
  - Compare input[i] < input[j] if needed
  - Copy smaller item & increment p and i or j.

$\Theta(N \log N)$ runtime

$\Theta(N)$ memory.

Insertion Sort

Starting w/ empty output,
add each item from input, inserting into output at the right spot.

Better Insertion Sort:

- Repeat for $i=0$ to $N-1$.
  - Designate item i as the travelling item.
  - Swap item backward until traveler is in the right place among all previously examined items.

$\Omega(N)$, $O(N^2)$ runtime.

Situation: Say you have a sorted array and someone changes just one value. What sort would you use?

Insertion sort → $\Omega(N)$ minimizes swaps.

Insertion sweet spots:

$\Theta(N+K)$ where K is number of inversions.

Insertion sort is very good for "almost" sorted arrays.

Also for small arrays, insertion sort is fastest.
$(N < 15)$