Computer Science 136
Data Structures
Lecture #13 (October 13, 2021)

Notes:

1. Announcements:
   (a) I have office hours today and tomorrow 1-2:30, and 10-11:15 tomorrow.
   (b) Sample exam solutions posted on the website.
   (c) Questions?
2. Getting it done: Sorting.
   (a) Recall: Bubble sort.
   (b) Recall: Selection sort. Halloween sorting technique.
   (c) Recall: Insertion sort. Poker hand sorting technique.
   (d) Quicksort. A partitioning approach.
      i. Based on this important fact: any value can be moved to its ultimate sorted location.
      ii. *Partitioning* moves this *pivot* value by putting smaller values to the left, and larger values to the right.
      iii. Once the pivot is located, you have two subarrays that contain smaller and larger values, respectively. Those can be sorted using any technique you desire. Typically: it's quicksort.
      iv. Best case behavior is $O(n \log n)$; worst case behavior is $O(n^2)$.
      v. Think carefully about the best and worst cases and consider solutions.
   (e) Mergesort. The ultimate divide-and-conquer approach.
      i. Divide the array in half and sort each half.
      ii. Merge the two ordered lists (this can be done in $O(n)$ time).
      iii. Result: $O(n \log n)$ sort, independent of value distribution.
   (f) The use of *comparators*.
      i. We can instrument our sorting methods to make use of externally specified comparison functions.
      ii. The `Comparator` class is a class that contains one abstract method, `compare(a,b)`.
      iii. It's relatively easy to construct a new class (and instance) that compares, say, two `Integers`.
      iv. Shorthand: the use of *lambdas*....