Computer Science 136
Data Structures
Lecture #20 (November 5, 2021)

Notes.

1. Questions?
2. How might you convert infix (standard) math formulas to postfix?
   (a) Idea: store expressions in a tree.
   (b) Interior nodes: operators. Leaves: values.
   (c) Higher nodes are evaluated *after* lower nodes.
   (d) Inorder traversal: infix. Postorder traversal: postfix.
   (e) Hmm. How do we perform step 1, given an infix expression? Postfix expression?
3. Priority queues.
   (a) A structure that delivers a smallest item next (via `getFirst` (nonmutating) and `remove` (mutating)). Items are `Comparable`.
   (b) Could be implemented using an `OrderedStructure`, for example, an `OrderedVector`. Problems? How do we use this to build a `PriorityVector`?
   (c) New concept: A heap.
      i. Heap is a binary tree structure.
      ii. Root is smallest (in the natural ordering of the values).
      iii. Subtrees are heaps.
      iv. How do we insert values?
      v. How do we remove values?
      vi. Everything is logarithmic. Cool.
   (d) Vector-based heap implementation.
      i. Notion: `percolateUp` and `pushDownRoot`.
      ii. Uses no extra space.
      iii. Basis for a vector sorting operation.
   (e) Skew-heap implementation.
      i. Notion: a `merge` of two heaps.
      ii. Has amortized logarithmic cost even though the tree is not necessarily very balanced. Very cool.