Heap operations

* last week: definition (complete binary tree ordered so that each node's value is <= its children's values, for a min-heap)

* insert()

First, add the item at the "next" leaf position-- at the end of the list used to store the heap items.

Second: fix the heap order by swapping the new value with its parent, repeatedly, as long as the new value is smaller than its parent. ("percolating up" or "bubbling up") This stops when either (1) the new value ends up at the root, or (2) the parent of the new value is less than or equal to the new value.

Running time? O(log n)
- constant number of operations on each level, and there are only log n many levels in a complete binary tree;
- equivalently, the index n can be divided in half at most O(log n) times before reaching 1 or 0.

* get_min(): return the item at index 0, in time O(1).
First: remove the last element and copy it into the root position (at index 0).

Second: repeatedly swap the value at the root with its child of smallest priority, as long as the child's priority is smaller.

Running time? $O(\log n)$, just like before.
Heap Sort

To sort list L:
1. Create a heap; insert the items from L into the heap, one by one.
2. Extract elements from the heap one by one and copy them back into L, in order. The smallest element will be removed first, followed by the second-smallest, etc.

Running time?
O(n log n) to copy the elements of L into the heap
O(n log n) to copy the heap elements back into L