Creating Heaps

- Heap is an array-based implementation of a binary tree used for implementing priority queues, supports:
  - insert, findmin, deletemin: complexities?

- Using array minimizes storage (no explicit pointers), faster too
  --- children are located by index/position in array

- Heap is a binary tree with shape property, heap/value property
  - shape: tree filled at all levels (except perhaps last) and filled left-to-right (complete binary tree)
  - each node has value smaller than both children
Array-based heap

- store “node values” in array beginning at index 1
- for node with index k
  - left child: index $2k$
  - right child: index $2k+1$

- why is this conducive for maintaining heap shape?
- what about heap property?
- is the heap a search tree?
- where is minimal node?
- where are nodes added? deleted?
Thinking about heaps

- Where is minimal element?
  - Root, why?
- Where is maximal element?
  - Leaves, why?
- How many leaves are there in an N-node heap (big-Oh)?
  - O(n), but exact?
- What is complexity of find max in a minheap? Why?
  - O(n), but ½ N?
- Where is second smallest element? Why?
  - Near root?
Adding values to heap

- to maintain heap shape, must add new value in left-to-right order of last level
  - could violate heap property
  - move value “up” if too small

- change places with parent if heap property violated
  - stop when parent is smaller
  - stop when root is reached

- pull parent down, swapping isn’t necessary (optimization)
Adding values, details (pseudocode)

```java
void add(Object elt)
{
    // add elt to heap in myList

    myList[mySize] = elt;
    mySize++;
    int loc = mySize - 1;

    while (1 < loc &&
        elt < myList[loc/2])
    {
        myList[loc] = myList[loc/2];
        loc = loc/2; // go to parent
    }
    // what's true here?

    myList[loc] = elt;
}
```

```
6 10 7 13 9 21 19 25
0 1 2 3 4 5 6 7 8 9 10

ArrayList myList
```
Removing minimal element

- Where is minimal element?
  - If we remove it, what changes, shape/property?
- How can we maintain shape?
  - “last” element moves to root
  - What property is violated?
- After moving last element, subtrees of root are heaps, why?
  - Move root down (pull child up) does it matter where?
- When can we stop “re-heaping”?
  - Less than both children
  - Reach a leaf