Announcements

- Homework 2 solutions posted
- Homework 3 due 10/20
- Midterm Exam, open book/notes 10/22
  - see theory problems for examples
- My office hours this week 4-6 PM
Review

- Amortized Running time
- Splay Trees
- Tries
Priority Queues

- New abstract data type Priority Queue
  - Insert: add node with key
  - deleteMin: delete the node with smallest key
  - findMin: access the node with smallest key
  - (increase/decrease priority)
Tradeoffs

- Binary search trees contain full order information (inorder returns sorted list)
- Priority queues only maintain efficient method to find minimum element
- Loss in functionality is worth it for gain in speed
Simple Implementations

- Use a list
  - $O(1)$ insert, $O(N)$ deleteMin/findMin
- Use a balanced BST
  - $O(\log N)$ insert/deleteMin*/findMin
  - deleting min from BST leads to imbalance
Heap Implementation

- Binary tree with special properties
- Heap Structure Property: all nodes are full*
- Heap Order Property: any node is smaller than its children

*Note: The structure assumes a complete binary tree with full nodes.
Array Implementation

- A full tree is regular: we can store in an array
  - Root at $A[1]$
  - Node $i$ has children at $2i$ and $(2i+1)$
  - Parent at $\text{floor}(i/2)$
- No links necessary, so much faster (but only constant speedup)
Array Implementation

- A full tree is regular: we can easily store in an array
  - Root at $A[0]$
  - Node $i$ has children at $2(i+1)-1$ and $2(i+1)$
  - Parent at $\text{floor}((i+1)/2)-1$
- No links necessary, so faster (in most languages)
Insert

- To insert key $X$, create a hole in bottom level
- **Percolate up**
  - Is hole’s parent is less than $X$
    - If so, put $X$ in hole, heap order satisfied
    - If not, swap hole and parent and repeat
DeleteMin

- Save root node, and delete, creating a hole
- Take the last element in the heap X
- **Percolate down:**
  - is X is less than hole’s children?
    - if so, we’re done
    - if not, swap hole and smallest child and repeat
Changing a key

- Assuming you allow direct access to elements in heap
- decreaseKey: lower key, percolate up
- increaseKey: raise key, percolate down
Running times

- Insert/deleteMin $O(\log N)$
- findMin $O(1)$
- Where’s the big gain?
  - buildHeap: given $N$ items, creates a heap in linear time
Reading

• This class and next: Weiss 6.1-6.3