CS221: Algorithms and Data Structures Lecture #3 Mind Your Priority Queues

Steve Wolfman
2014W1

Learning Goals

- Provide examples of appropriate applications for priority queues.
- Describe efficient implementations of priority queues.
- Relate heaps to binary trees, binary search trees, and arrays.

It is **not** a goal for this term to be able to manipulate heaps by hand.

Today's Outline

- Priority Queue ADT
- Solutions So Far?
- 10km View of Heaps

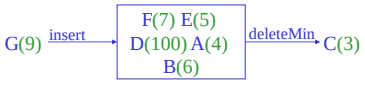
Back to Queues

- Some applications
  - ordering CPU jobs
  - simulating events
  - picking the next search site
- Problems?
  - short jobs **should go first**
  - earliest (simulated time) events **should go first**
  - most promising sites **should be searched first**

Remember ADTs?

Priority Queue ADT

- Priority Queue operations
  - create
  - destroy
  - insert
  - deleteMin
  - is_empty



- Priority Queue property: for two elements in the queue, *x* and *y*, if *x* has a lower priority value than *y*, *x* will be deleted before *y*

Applications of the Priority Q

Your call will ***not*** be answered in the order it was received.

- Call queues for help lines (or don't you think so?)
- Hold jobs for a printer in order of length
- Simulate events (hold in order of time)
- Sort numbers
- Store packets on network routers in order of urgency
- Select symbols for compression (hold in order of frequency)
- Anything *greedy*: an algorithm that makes the "locally best choice" at each step (hold in order of quality)

Naïve Priority Q Data Structures

- Unsorted list:
  - *insert:*
  - *deleteMin:*
- Sorted list:
  - *insert:*
  - *deleteMin:*

a. O(lg n)
b. O(n)
c. O(n lg n)
d. O(n^2)
e. Something else

Today's Outline

- Priority Queue ADT
- Solutions So Far?
- 10km View of Heaps

How Can We Efficiently Implement a PQ?

- Stack?
- Queue?
- Linked List
  - Singly, doubly, …
- Array
  - Circular, resizing, …
- BST
- AVL Tree

Priority value is _____

`insert` is _____

`delete_min` is _____

AVL Tree as PQ

How well does this work in terms of the number of priority values/data in the tree, *n*?

Runtime of `insert`?

Runtime of `delete_min`?

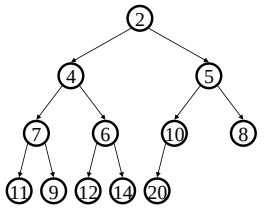
Today's Outline

- Priority Queue ADT
- Solutions So Far?
- 10km View of Heaps

Binary Heap Priority Q Data Structure

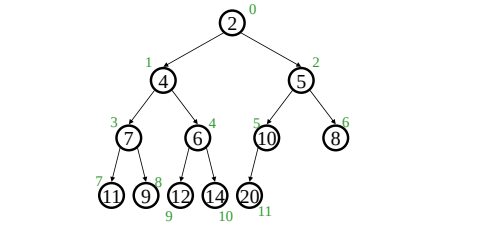
Look! Invariants!

- Heap-order property
  - parent's key is less than or equal to children's keys
  - result: minimum is always at the top
- Structure property
  - "nearly complete tree"
  - result: depth is always O(log n); next open location always known



WARNING: this has *NO SIMILARITY* to the "heap" you hear about when people say "things you create with `new` go on the heap". And, this **is** a binary tree but is **NOT** a binary search tree.

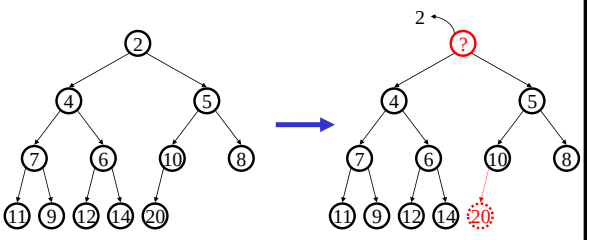
Nifty Storage Trick



So, we get the speed and "spatial locality" of arrays.
(Also the occasional expensive resizes of arrays.)

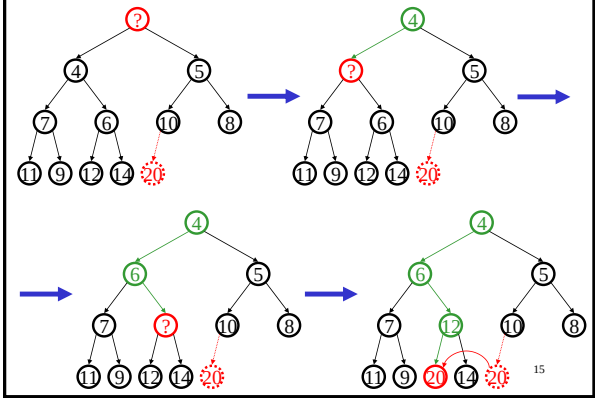
DeleteMin

```
pqueue.deleteMin()
```

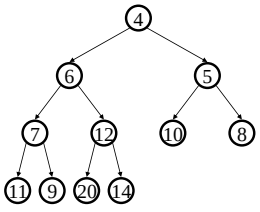


Invariants violated! DOOOM!!!

Percolate Down



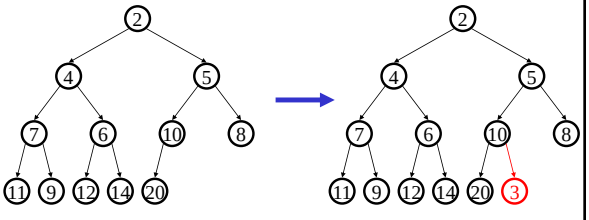
Finally…



Algorithm intuition:
move the rightmost, bottom element to the root;
then, fix the invariant downward until stable.

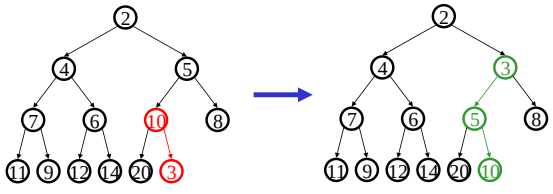
Insert

```
pqueue.insert(3)
```



Invariant violated! What will we do?
Intuition: insert at the bottom-right;
then, fix invariant upward until stable.

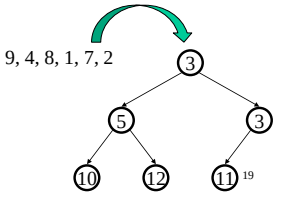
Insert Result



Closer Look at Creating Heaps

To create a heap given a list of items:
Create an empty heap.
For each item: insert into heap.

Time complexity?
a. O(lg n)
b. O(n)
c. O(n lg n)
d. $O(n^2)$
e. None of these



A Better BuildHeap

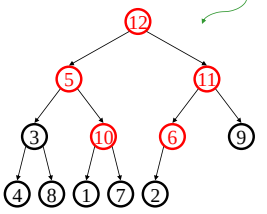
Floyd's Method. Thank you, Floyd.

12	5	11	3	10	6	9	4	8	1	7	2

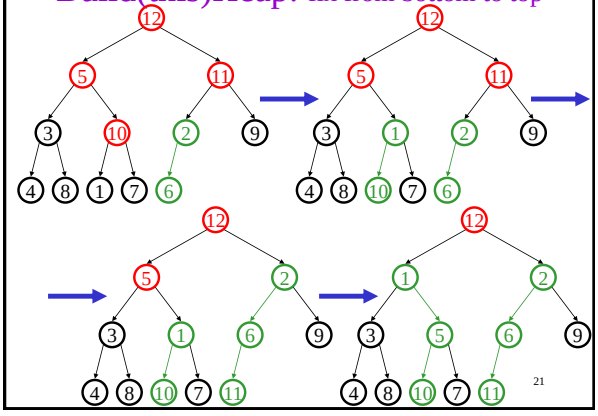
pretend it's a heap and fix the heap-order property!

Invariant violated!

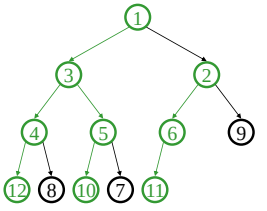
Where can the order invariant be violated in general?
a. Anywhere
b. Non-leaves
c. Non-roots



Build(this)Heap: fix from bottom to top

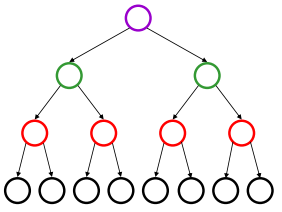


Finally…



runtime:

Build(any)Heap



This is as many violations as we can get.
How do we fix them? Let's play colouring games!
"amortized analysis!"

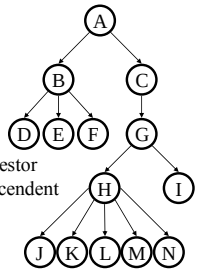
Other Uses for Trees

(besides BSTs and heaps)

- Family Trees
- Organization Charts
- Classification trees
  - what kind of flower is this?
  - is this mushroom poisonous?
- File directory structure
  - folders, subfolders in Windows
  - directories, subdirectories in UNIX
- Function call tree (i.e., a record of everything that goes in the call stack)

Tree Terminology Reference

root: the single node with no parent
leaf: a node with no children
child: a node pointed to by me
parent: the node that points to me
sibling: another child of my parent
ancestor: my parent or my parent's ancestor
descendent: my child or my child's descendent
subtree: a node and its descendents



We sometimes use degenerate versions of these definitions that allow NULL as the empty tree. (This can be *very* handy for recursive base cases!)

25

To Do

- Read: KW Section 8.5

26

Coming Up

- Sorting, sorting, and more sorting!

27