

Priority Queues and Heaps

Bryce Boe

2013/08/13

CS24, Summer 2013 C

Outline

- Thursday Recap
- More Tree Properties
- Priority Queue
- Heaps

$O(n^2)$ Sorting Algorithms

- Bubble sort
 - Bubble the largest element to the end in each pass
- Insertion sort
 - Insert the next element into the sorted portion of the list
- Selection sort
 - Find the smallest item and put it in its proper location

$O(n \log(n))$ Sort Algorithms

- Merge Sort
 - Break the problem up until you have 1 or 2 items and put them in order
 - Merge the sorted lists $O(k)$ where k is the size of the small lists
 - $T(n) = 2T(n/2) + O(n) \implies O(n * \log(n))$ (masters theorem)

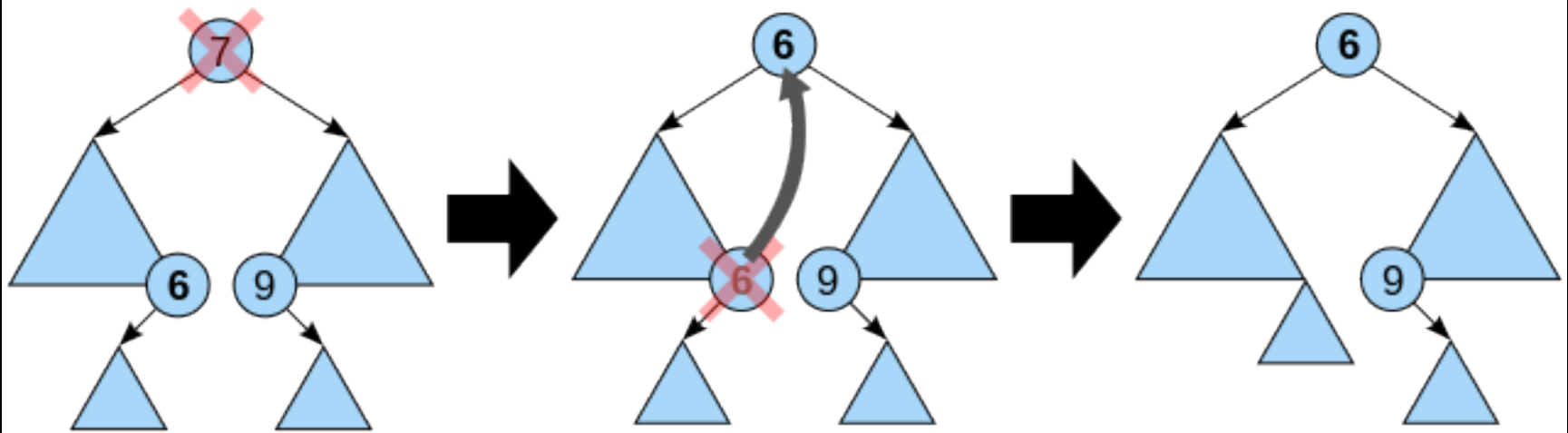
BST Remove

- If the node has no children simply remove it
- If the node has a single child, update its parent pointer to point to its child and remove the node

Removing a node with two children

- Replace the value of the node with the largest value in its left-subtree (right-most descendant on the left hand side)
- Then repeat the remove procedure to remove the node whose value was used in the replacement

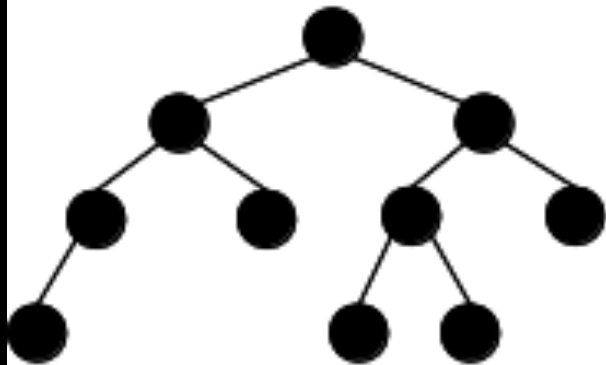
Removing a node with two children



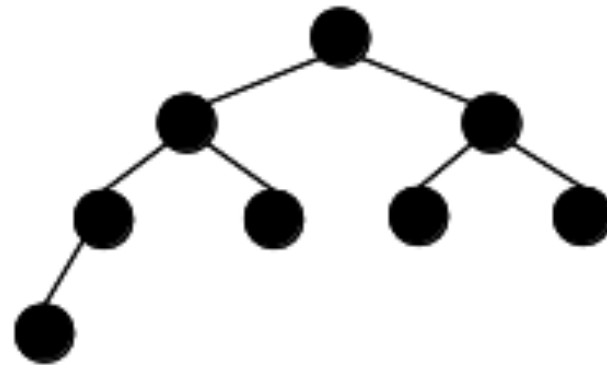
Full Binary Tree

- Full Binary Tree
 - A binary tree in which every node has either 0 or 2 children
- Complete Binary Tree
 - A binary tree in which every level is completely filled save for the last where all items must be filled from the left-hand-side

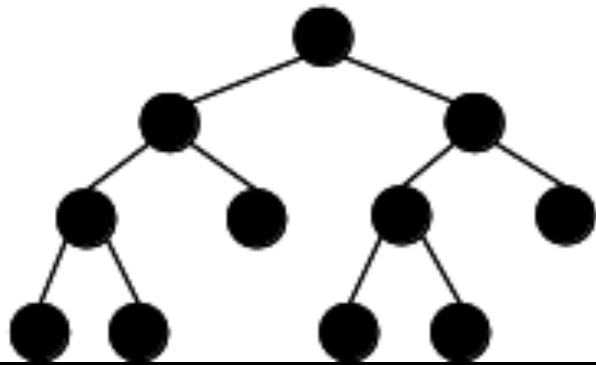
Neither complete nor full



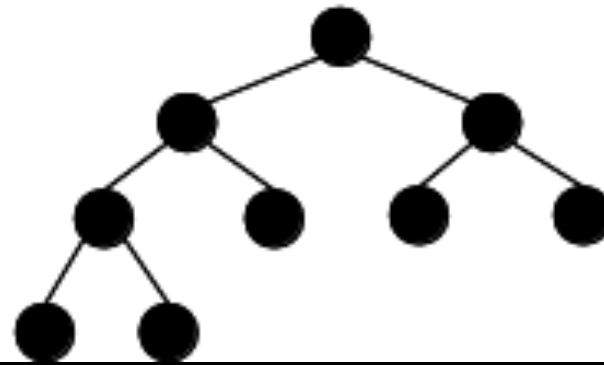
Complete but not full



Full but not complete



Complete and full



Priority Queue

- Abstract data type with operations similar to the queue
 - enqueue(item, **priority**)
 - Add an item to the queue
 - dequeue()
 - Remove the item with highest priority from the queue (FIFO order for items with same priority)

Brainstorm

- Discuss for a few minutes with those near you:
 - Come up with two functionally distinct ways to implement a priority queue

Bounded Priority Queue

- Simple trick if the number of priorities is bounded
- Provide one FIFO queue for each priority
- Always look for items starting from the highest queue before proceeding to the next
- **enqueue: $O(1)$**
- **dequeue: $O(1)$**

Sorted List Implementation

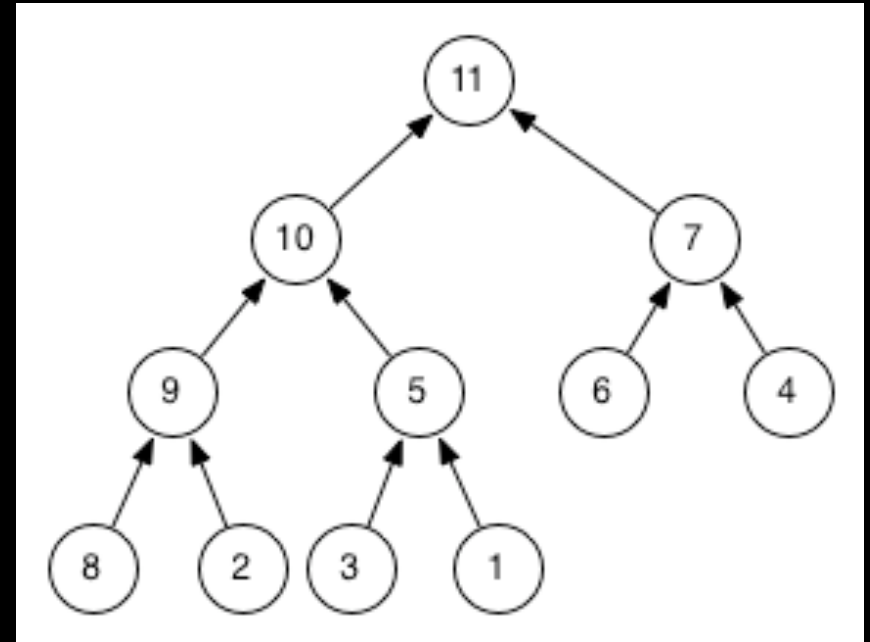
- Use a standard linked-list implementation of a queue
- Modify enqueue such that it places the item in the appropriate location in the list
- **enqueue: $O(n)$**
- **dequeue: $O(1)$**

Tree Implementation

- Construct a tree such that its root is always the highest priority
- Additionally every subtree has the same property (parent is of equal or higher priority than children)

Heap

- A **complete** tree where each node's parent has a higher or equal priority



Think about it

- Are there any trees that can be both a heap and a BST?

Heap Insertion (enqueue)

- Place item in the next free location (obey the completeness property)
- Continue to swap it with its parents until the ordering property is valid (bubble up)
- **enqueue:** $O(\log(n))$ – worst cases traverses up the entire height of the tree

Heap removal (dequeue)

- Store the value at the root to return at the end
- Swap the last item in the tree with the root
- Continually swap the current node with its child of highest priority until it is of higher priority than both children (bubble-down)
- **dequeue:** $O(\log(n))$ – worst cases traverses down the entire height of the tree

Lab 7

- Write function to test if array is in heap-order
- Arrays are great for storing complete trees

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

