Priority Queues (Heaps)

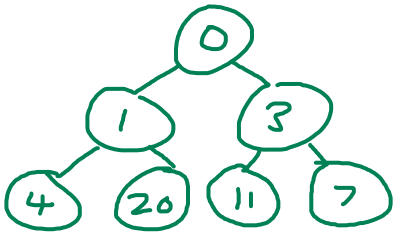
What is a priority queue?

It is a abstract data structure where inserted items are automatically ranked in terms of "importance." We as programmers get to determine what "importance" means.
For example, suppose we have a priority queue that takes in ints; we place highest priority on larger ints. Then when we ask the queue for the next item, it will return the largest int in the queue.

What is an efficient way to represent priority queues?

Heaps are binary tree where parent nodes have higher priority than child nodes (root = highest priority)

Example: (smaller int = higher priority)

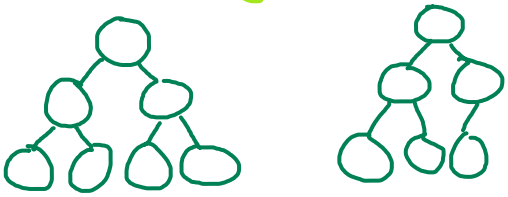


Properties

(1) Complete

a) Missing nodes only at leaf level
b) All leaf nodes as far left as possible

Correct!



Wrong!

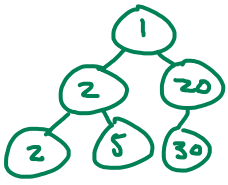
(missing node at non-leaf level)

(not all leaf nodes are as far left)

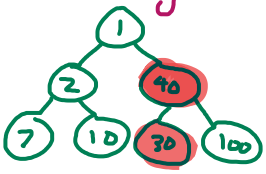
(2) Heap Priority

Every node has higher or equal priority to their children (higher height = higher priority)

Correct!



Wrong!



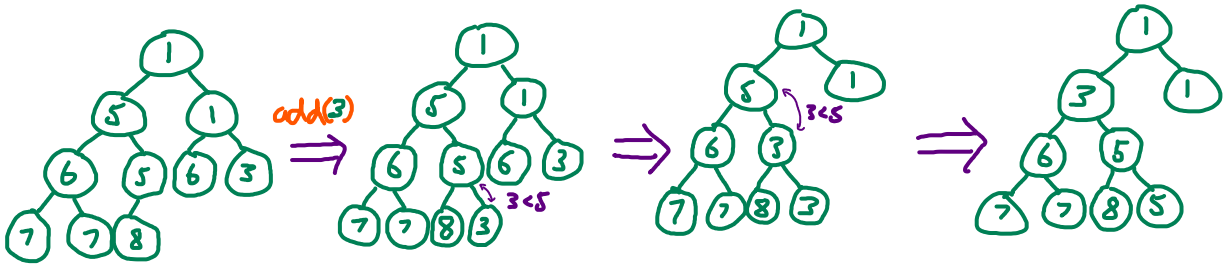
Some Definitions

end: bottom-most, right-most node
swim: a node swaps with parent node in a tree
sink: a node swaps with a child node in a tree

Heap Methods

Heap add(x): add x to heap

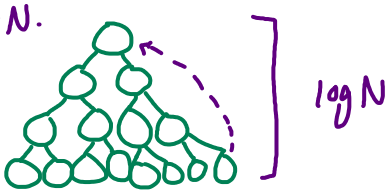
1. Put x at end of heap
2. Let x keep swimming until parent has more priority or no parents



add Runtime?

Best case: $\Theta(1)$ ← parent is greater priority in very first iteration
Worst case: $\Theta(\log N)$

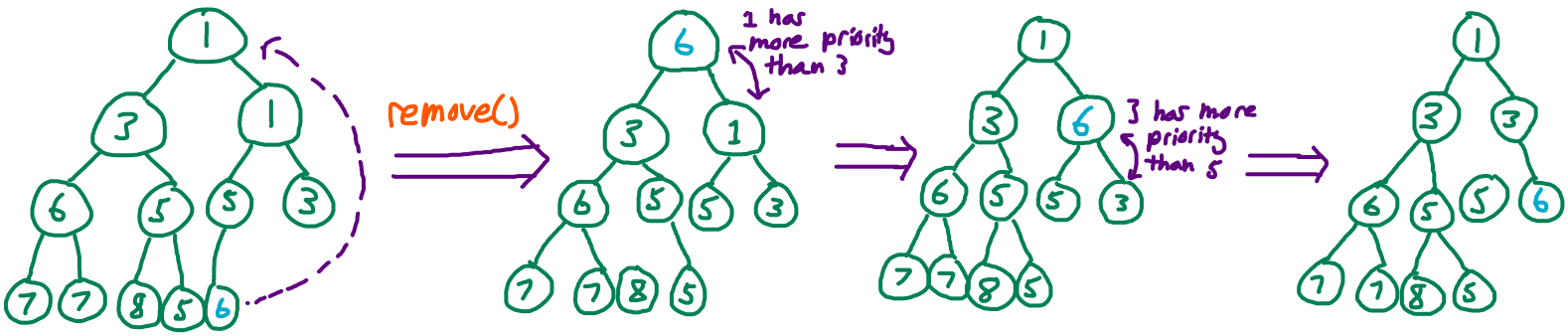
In the worst case, the new node swims up all the way to the root of the tree.
This means the number of swaps = height of tree = $\log N$.



remove(): removes and returns the highest priority item (root).

1. Replace root with end.
2. Sink new root until all children less priority or no children.

Which child to choose to swap? The one w/ highest priority.



remove() runtime?

Best case: $\Theta(1)$ ← new root already higher or equal priority than children
Worst case: $\Theta(\log N)$

In the worst case, new root sinks all the way to the bottom. Number swaps = height = $\log N$