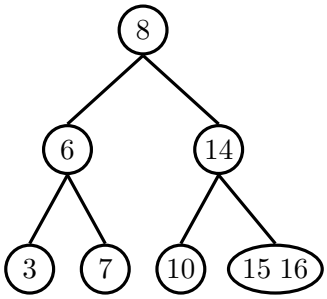
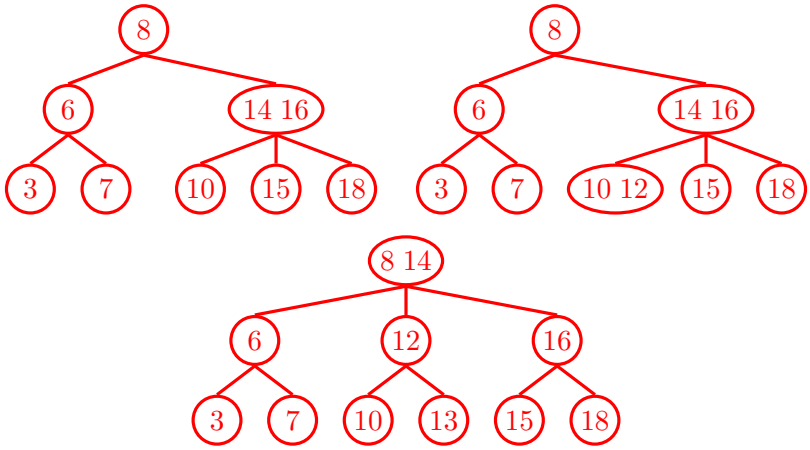
CSM 61B Heaps

Spring 2019 Mentoring 8: November 5, 2018

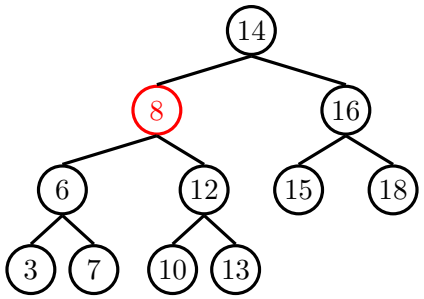
1 2-3 Forever



1.1 Draw what the 2-3 tree would look like after inserting 18, 12, and 13.



1.2 Now, convert the resulting 2-3 tree to a left-leaning red-black tree.



2 Min-Heapify This

2.1 In general, there are 4 ways to heapify. Which 2 ways actually work?

- Level order, bubbling up
- Level order, bubbling down

- Reverse level order, bubbling up
- Reverse level order, bubbling down

2.2 Are the values in an array-based min-heap sorted in ascending order?

2.3 Is an array that is sorted in descending order also a max-oriented heap?

3 K Largest Items

3.1 The largest item in a heap must appear in position 1, and the second largest must appear in position 2 or 3. Give the list of positions in a heap where the kth largest can appear for $k \in \{2, 3, 4\}$. Assume values are distinct.

4 Amortized Analysis

4.1 Give the *amortized runtime analysis* for push and pop for the priority queue below.

```
class TwinListPriorityQueue<E implements Comparable> {
    ArrayList<E> L1, L2;
    void push(E item) {
        L1.push(elem);
        if (L1.size() >= Math.log(L2.size())) {
            L2.addAll(L1);
            mergeSort(L2);
            L1.clear();
        }
    }
    E pop() {
        E min1 = getMin(L1);
        E min2 = L2.poll();
        if (min1.compareTo(min2) < 0) {
            L1.remove(min1);
            return min1;
        } else {
            L2.remove(min2);
            return min2;
        }
    }
}
```

5 Kelp!

5.1 You have been hired by Alan to help design a priority queue implementation for *Kelp*, the new seafood review startup, ordered on the timestamp of each `Review`.

Describe a data structure that supports the following operations.

- `insert(Review r)` a `Review` in $O(\log N)$.
- `edit(int id, String body)` any one `Review` in $\Theta(1)$.
- `sixtyOne()`: return the sixty-first latest `Review` in $\Theta(1)$.
- `pollSixtyOne()`: remove and return the sixty-first latest `Review` in $O(\log N)$.