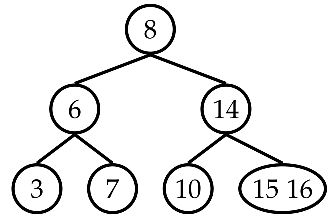
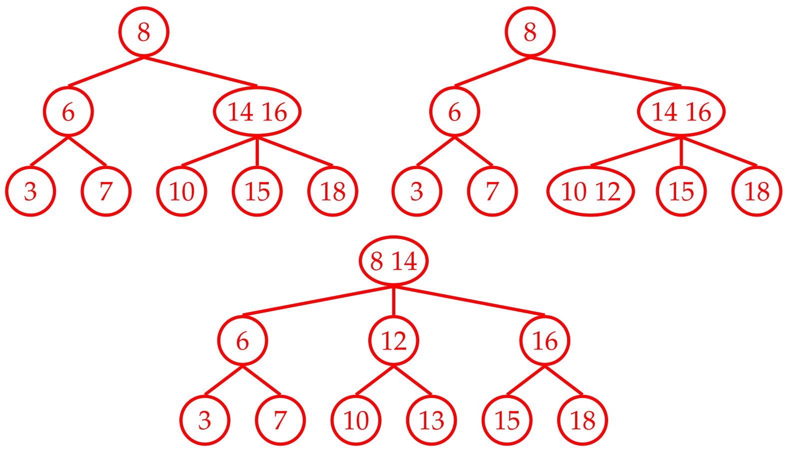
CS 61B Spring 2017	Disjoint Sets & Trees Discussion 11: April 5, 2017



1 2-3-Forever

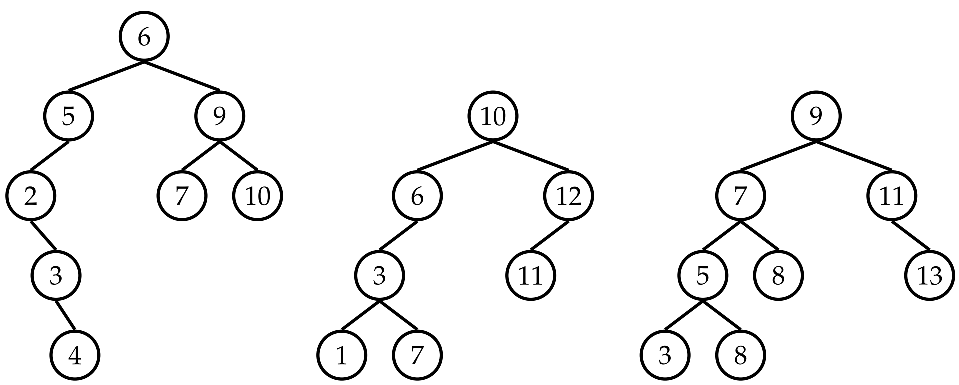
1.1 Draw what the 2-3 tree would look like after each step in inserting 18, 12, and 13.



Make sure to know how to convert each of these into left-leaning red-black trees!

2 Unbalanced Search Trees

2.1 Given the following binary trees, determine if each is a binary search tree, and whether the height of the tree is the same as the height of the optimal binary search tree containing the given elements.



1. Valid but not balanced
2. Invalid and not balanced
3. Invalid but balanced

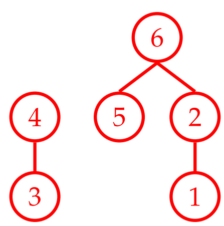
2.2 Provide tight asymptotic runtime bounds in terms of N, the number of nodes in the tree, for the following operations and data structures.

Operations	BST	Red-black tree
`boolean contains(T o);`	$\Omega(1)$ $O(N)$	$\Omega(1)$ $O(\log N)$
`void insert(T o);`	$\Omega(1)$ $O(N)$	$\Theta(\log N)$
`T get(int i);`	$\Omega(1)$ $O(N)$	$\Omega(1)$ $O(\log N)$

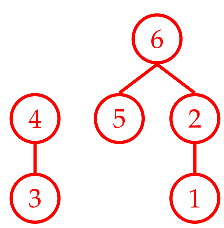
3 Disjointed Sets

3.1 Suppose we have a `WeightedQuickUnionUF` disjoint set with path compression. Show the tree structure in the union-find algorithm as the following sequence of commands is executed.

(a)
```
connect(1, 2);
connect(3, 4);
connect(5, 6);
connect(1, 6);
```



(b) `find(6);`



3.2 Describe how to construct a `WeightedQuickUnionUF` tree of maximum height.

To construct a max-height tree of N nodes, union two max-height trees with $\frac{N}{2}$ nodes each. The height of the final tree is 1 greater than each of the max-height $\frac{N}{2}$-node trees.