CS 61B Spring 2018

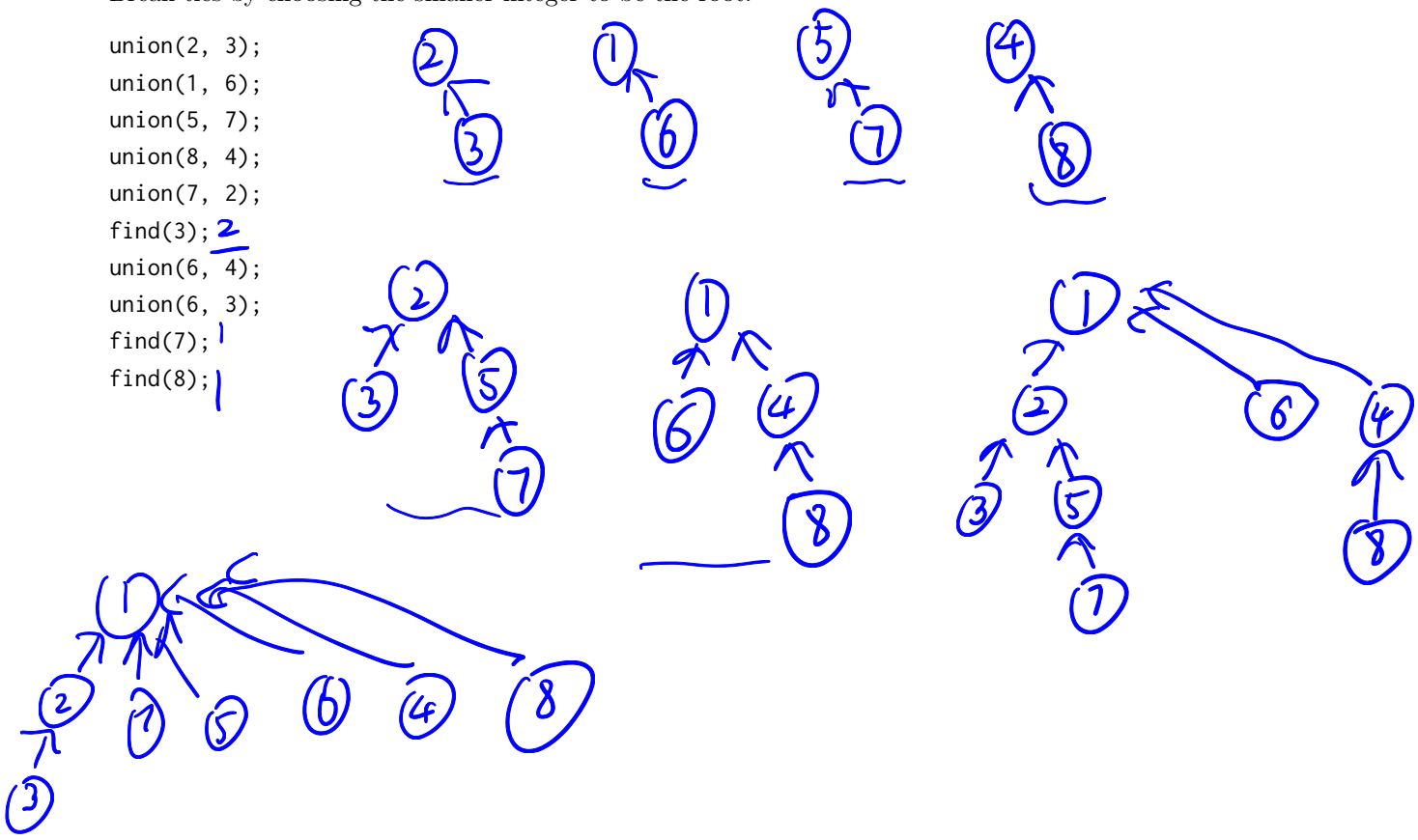
Disjoint Sets, Trees, Hashing

Discussion 9: March 12, 2018

1 WQU and Path Compression

Assume we have eight sets, represented by integers 1 through 8, that start off as completely disjoint sets. Draw the WQU Tree after the series of `union()` and `find()` operations with path compression. Write down the result of `find()` operations. Break ties by choosing the smaller integer to be the root.

```
union(2, 3);
union(1, 6);
union(5, 7);
union(8, 4);
union(7, 2);
find(3); 2
union(6, 4);
union(6, 3);
find(7); 1
find(8); 1
```

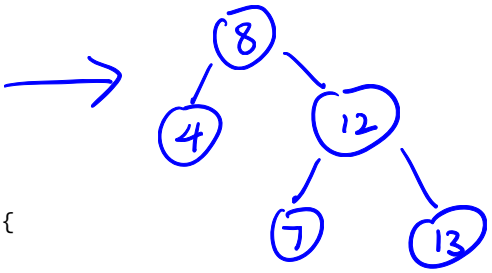


2 Is This a BST?

The following method `isBSTBad` is supposed check if a given binary tree is a BST, though for some binary trees, it is returning the wrong answer. Think about an example of a binary tree for which `isBSTBad` fails. Then, write `isBSTGood` so that it returns the correct answer for any binary tree. The `TreeNode` class is defined as follows:

```
class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;
}
```

Hint: You will find `Integer.MIN_VALUE` and `Integer.MAX_VALUE` helpful when writing `isBSTGood`.




```
public static boolean isBSTBad(TreeNode T) {
    if (T == null) {
        return true;
    } else if (T.left != null && T.left.val > T.val) {
        return false;
    } else if (T.right != null && T.right.val < T.val) {
        return false;
    } else {
        return isBSTBad(T.left) && isBSTBad(T.right);
    }
}
```

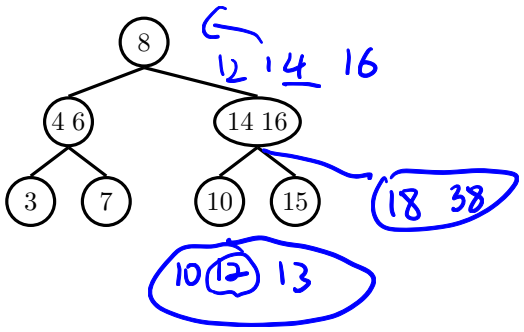
```
public static boolean isBSTGood(TreeNode T) {
    return isBSTHelper( T, Integer.MIN_VALUE, Integer.MAX_VALUE );
}

public static boolean isBSTHelper(TreeNode T, int min, int max ) {
  if ( T == null )
{ return true };
if ( T.val < min || T.val ≥ max )
{ return false}
return. isBSTHelper ( T.left , Integer.MIN_VALUE , T.val )
    && isBSTHelper ( T.right , T.val , Integer.MAX_VALUE )
}
```

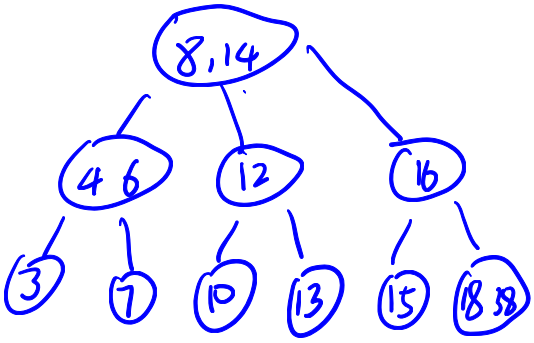
min

max

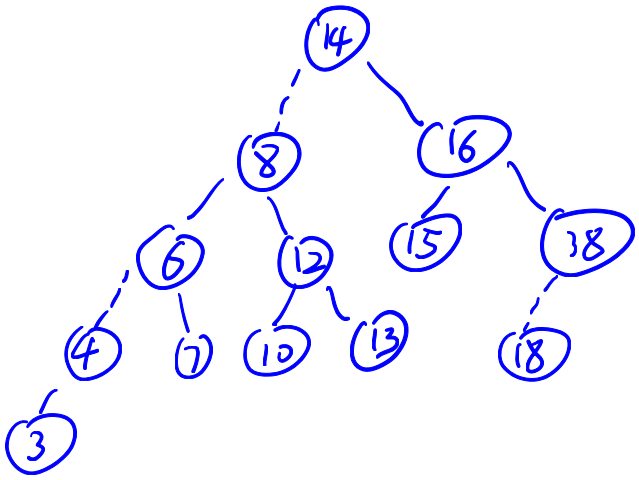
3 2-3 Trees and LLRB's

3.1 Draw what the following 2-3 tree would look like after inserting 18, 38, 12, and 13.





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



3.3 *Extra:* If a 2-3 tree has depth `H` (that is, the leaves are at distance `H` from the root), what is the maximum number of comparisons done in the corresponding red-black tree to find whether a certain key is present in the tree?

2H

4 Hashing

4.1 Here are three potential implementations of the `Integer`'s `hashCode()` function. Categorize each as either a valid or an invalid hash function. If it is invalid, explain why. If it is valid, point out a flaw or disadvantage.

```
1 public int hashCode() {
2     return -1;
3 }
```

valid only one bucket (H) (n)

collision. 100%

```
1 public int hashCode() {
2     return intValue() * intValue();
3 }
```

valid -1, 1 are the same hashcode.

```
1 public int hashCode() {
2     return super.hashCode();
3 }
```

invalid two values equals, may not have the same hash value

4.2 *Extra, but highly recommended:* For each of the following questions, answer **Always**, **Sometimes**, or **Never**.

(a) When you modify a key that has been inserted into a `HashMap` will you be able to retrieve that entry again? Explain.

Sometimes hash use memory id → Yes

others may be No.

(b) When you modify a value that has been inserted into a HashMap will you be able to retrieve that entry again? Explain.

Always. the key not changed.