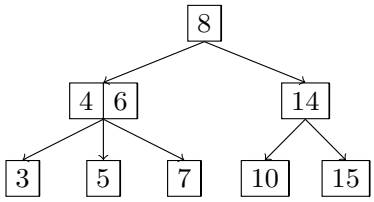
CS 61B Spring 2019

LLRBs, Hashing, Heaps

Discussion 8: March 11, 2019

1 2-3 Trees and LLRB's

1.1 Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.



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

1.3 *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?

2 Hashing

2.1 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 }
```

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

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

2.2 *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.

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

3 Heaps of Fun

3.1 Assume that we have a binary min-heap (smallest value on top) data structure called `Heap` that stores integers, and has properly implemented `insert` and `removeMin` methods. Draw the heap and its corresponding array representation after each of the operations below:

```
1 Heap<Character> h = new Heap<>();
2 h.insert('f');
3 h.insert('h');
4 h.insert('d');
5 h.insert('b');
6 h.insert('c');
7 h.removeMin();
8 h.removeMin();
```

3.2 Your friendly TA Tina challenges you to quickly implement an integer max-heap data structure. "Hah! I'll just use my min-heap implementation as a template to write MaxHeap.java," you think to yourself. Unfortunately, due to following the instructions of a shady stackoverflow post, you manage to permanently delete your `MinHeap.java` file. Luckily, you notice that you still have `MinHeap.class`. Can you still complete the challenge before time runs out?

Hint: Although you cannot alter them, you can still use methods from `MinHeap`.