Red-Black Tree Implementation: Bugs

No Collaborators

1. findNodeFindsNode test failed

`Assert.assertEquals(expectedResult, actualResult) failed`

Input tree: Output tree:


findNode was returning null if the root was *not* nil, and returning the same thing recursively otherwise, which would end up always returning null.

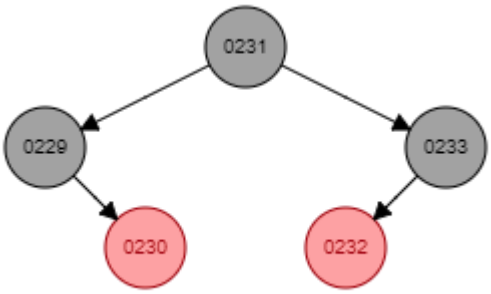
Expected output: node with input key, instead returned null

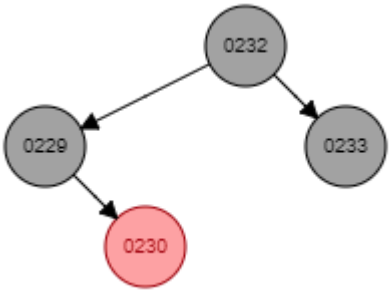
Fix: compare input parameter to current node key, recurse depending on comparison, returning node in the case that key is equal

2. deleteNodeDeletesRootWithGrandchildren test failed

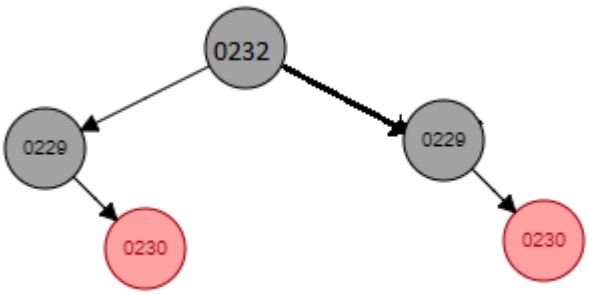
`Assert.assertTrue(tree.isValid()); failed`

Input tree: Output tree, expected:





Output tree, actual:



Problem: deleting root causes tree to become invalid

Bug: deleteNode had a bug where replacement node's right child would be set to the original's left child, fixing this fixed the test

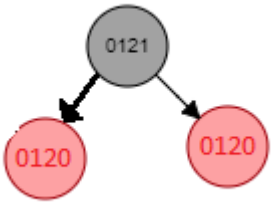
3. insertNodeInsertsSecondChildAsLeft

```
Assert.assertTrue(tree.isValid()); failed
```

Input tree:

Output tree, expected:

Output tree, actual:



Problem: inserting into left node would also insert it into the right node

Bug: caused by missing else statement, once added the problem is fixed

4. findMaximumFindsMaximumNode

```
Assert.assertEquals(expectedResult, actualResult); fails
```

Input tree:

Output tree:

Problem: findMaximum function returns the node with key 8 instead of the node with max key 9

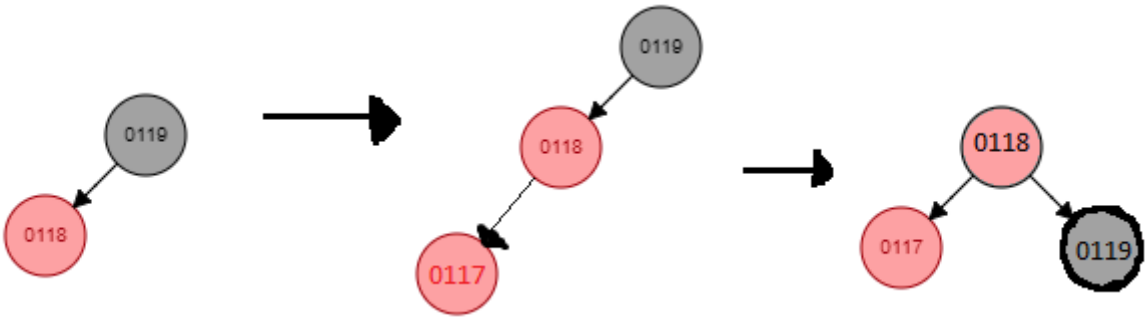
Bug: findMaximum was doing the same thing as find minimum, traversing left down the tree, fixed when changed to traverse down the right side.

5. insertNodeInsertsLeftTwice

```
tree.insertNode(newNode); fails, NullPointerException
```

Input tree: Output tree, expected:

Output tree, actual:



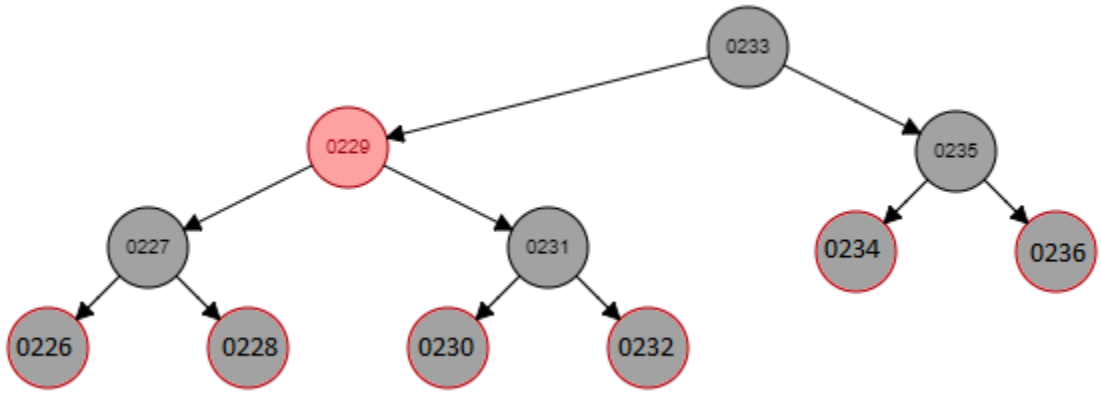
Problem: insertFixup has a NullPointerException

Bug: this happens because insertFixup messes up when assigning colors, sets z.parent to black, then sets it to red right after, causing the fixup go loop one more time and fail because it tries to reference a null parent. Fixed by setting parent.parent to red instead of the parent.

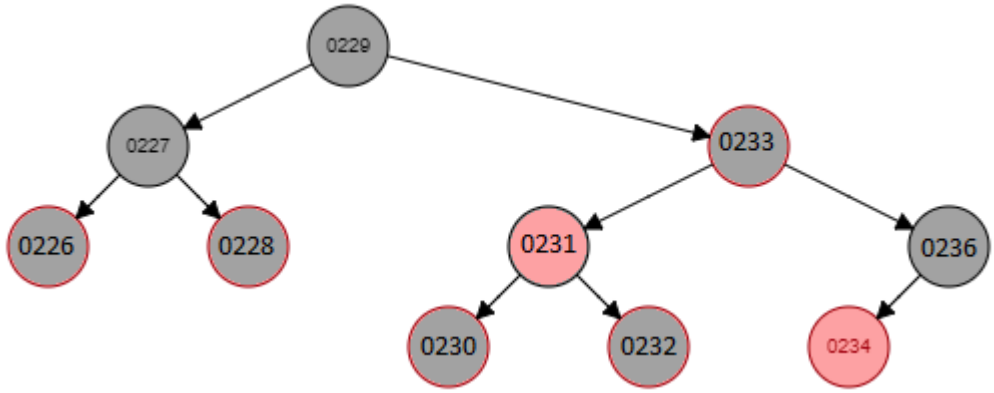
6. deleteNodePassesComplexTest2

```
Assert.assertTrue(tree.isValid()); fails
```

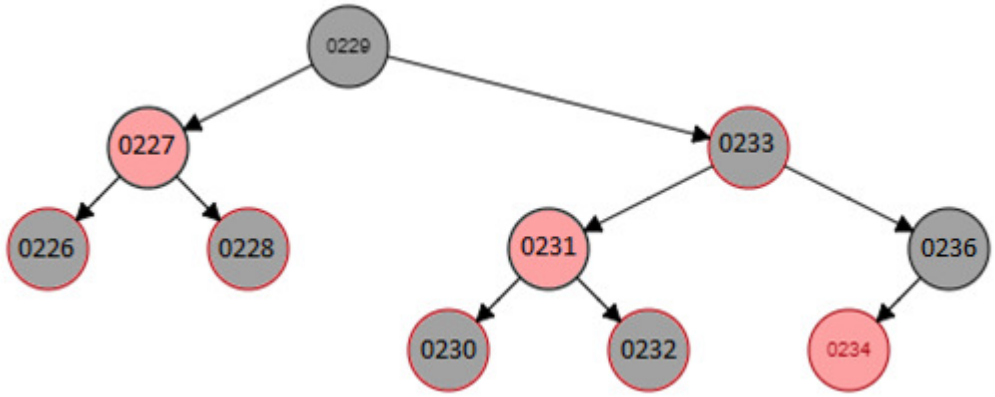
Input tree:



Output tree, expected:



Output tree, actual:



Problem: black-height is incorrect after deletion, node is made red, but the rest of the nodes on the other side of the tree are still black which causes unbalanced black-heights

Bug: looking through the code, in the delete function it checked if y.parent is y, which should be z. While this is a bug, it does not fix the problem. The bug was in deleteFixup, where x.parent has its color set to black when it should be red. Fixing this caused the test to pass.