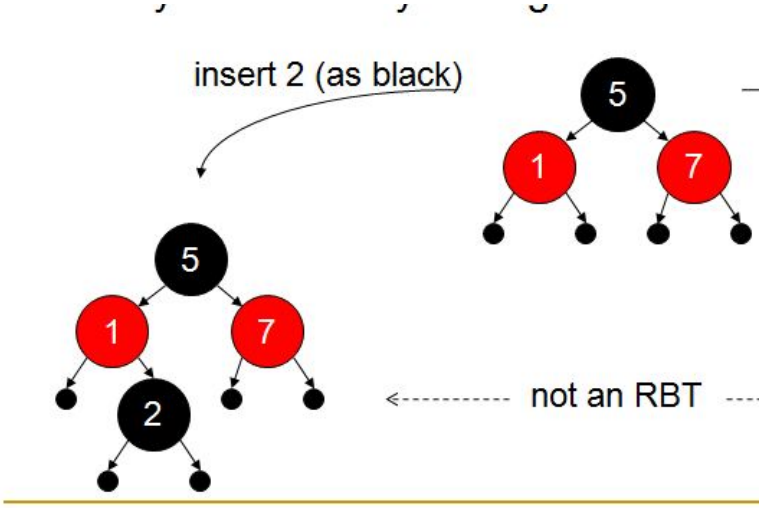
Inserting into red black tree

I am taking an algorithms course and in my course slides, there is an example of insertion into a red-black tree:



My question is, why don't we let "2" be a leaf node here? It looks like if we let it be a leaf node, then no condition of a red black tree is violated. What am I missing here?

algorithm insert tree red-black-tree

edited Mar 22 '13 at 19:12
dckrooney
1,705 8 23

asked Mar 22 '13 at 15:58
user2110714

3 Answers

All the leaves of a Red Black tree have to be `NIL` Check property 3

answered Mar 22 '13 at 16:05
noMAD
3,038 5 32 60

That was what i have been trying to ask, so thank you. – user2110714 Mar 22 '13 at 16:07

@noMAD: I am afraid your answer did not answer the question. – sowrov Mar 22 '13 at 18:01

@sowrov: I read your answer, you explain insertion but you didn't answer the OP's question. He asked, `why don't we let "2" be a leaf node here?` I just answered to that. – noMAD Mar 22 '13 at 18:56

The Problem is not with the position of 2 the the second tree of your image but the color of different nodes. Here is the explanation:

1st Rule of insertion in Red-Black tree is: the newly inserted node has to be always Red. You fall in case 3 insertion where both the father and uncle of node 2 is Red. So they are needed to be recolored to Black, and the grandfather will become Red but as the grandfather is root so it will become Black again.

So the new tree (after inserting 2) should be like this (r and b indicate color, .b is Nil node):

```
      5b
    /    \
  1b      7b
```

```
  /   \    / \
.b    2r .b  .b
      / \
     .b  .b
```

And why we always need to insert red node in RBT, you may ask? Answer is, 1st we know every NIL nodes are always Black in RBT, 2nd we have rule 5. `Every simple path from a given node to any of its descendant leaves contains the same number of black nodes.` Now if we insert a black node at the end the tree will violate this rule, just put 2b in above tree instead of 2r and keep color of 1 and 7 red, then count black node from root to any Nil node, you will see some path have 2 back nodes and some path have 3 black nodes.

edited Mar 22 '13 at 18:03

answered Mar 22 '13 at 17:38

sowrov

691 7 12

The wikipedia article, based on the same idea, explains it as follow:

> In many of the presentations of tree data structures, it is possible for a node to have only one child, and leaf nodes contain data. It is possible to present red–black trees in this paradigm, but it changes several of the properties and complicates the algorithms. For this reason, this article uses "null leaves",

So clearly nothing prevents you to do it your way, but you have to take it in account in your algorithms, which make them significantly more complex. Perhaps this issue can be somewhat alleviated by using OOP, where leaves contain elements, but behave as nodes with empty leaves.

Anyway, it's a trade off: what you would gain in space (roughly two pointers set to `NULL` in C), you'd probably lose in code complexity, computation time, or in the object runtime representation (specialized methods for the leaves).

edited Mar 22 '13 at 18:17

answered Mar 22 '13 at 17:53

didierc

11.5k 2 14 40
