50001 - Algorithm Analysis and Design - Lecture 12

Oliver Killane

18/11/21

Lecture Recording

Lecture recording is available here

Red-Black Trees

AVL trees worked by storing an extra integer (height) to use in rebalancing, **red-black trees** use an extra bit to determine if a node is red or black.

In practice they are less balanced than **AVL trees** however the insertion is faster and the data structure is a little bit smaller.

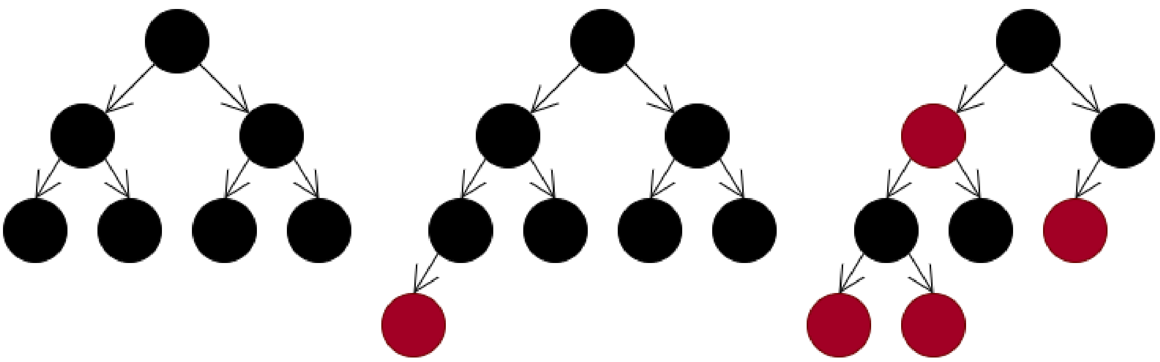
```haskell
data Colour = Red | Black

data RBTree a = Empty | Node Colour (RBTree a) a (RBTree a)
```

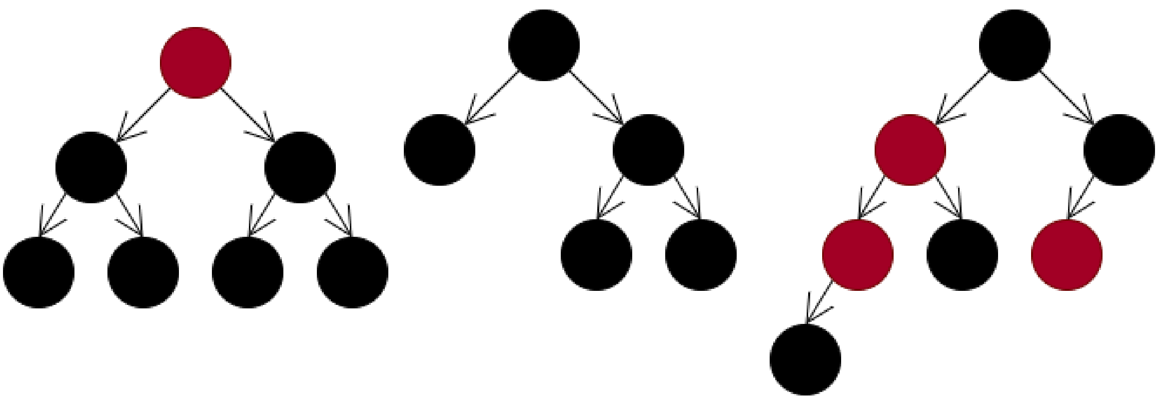
The structure relies on two invariances:

1. Every Red node must have a Black parent node.
2. Every path from the root to leaf must have the same number of black nodes.

Valid Red Black Trees



Invalid Red Black Trees

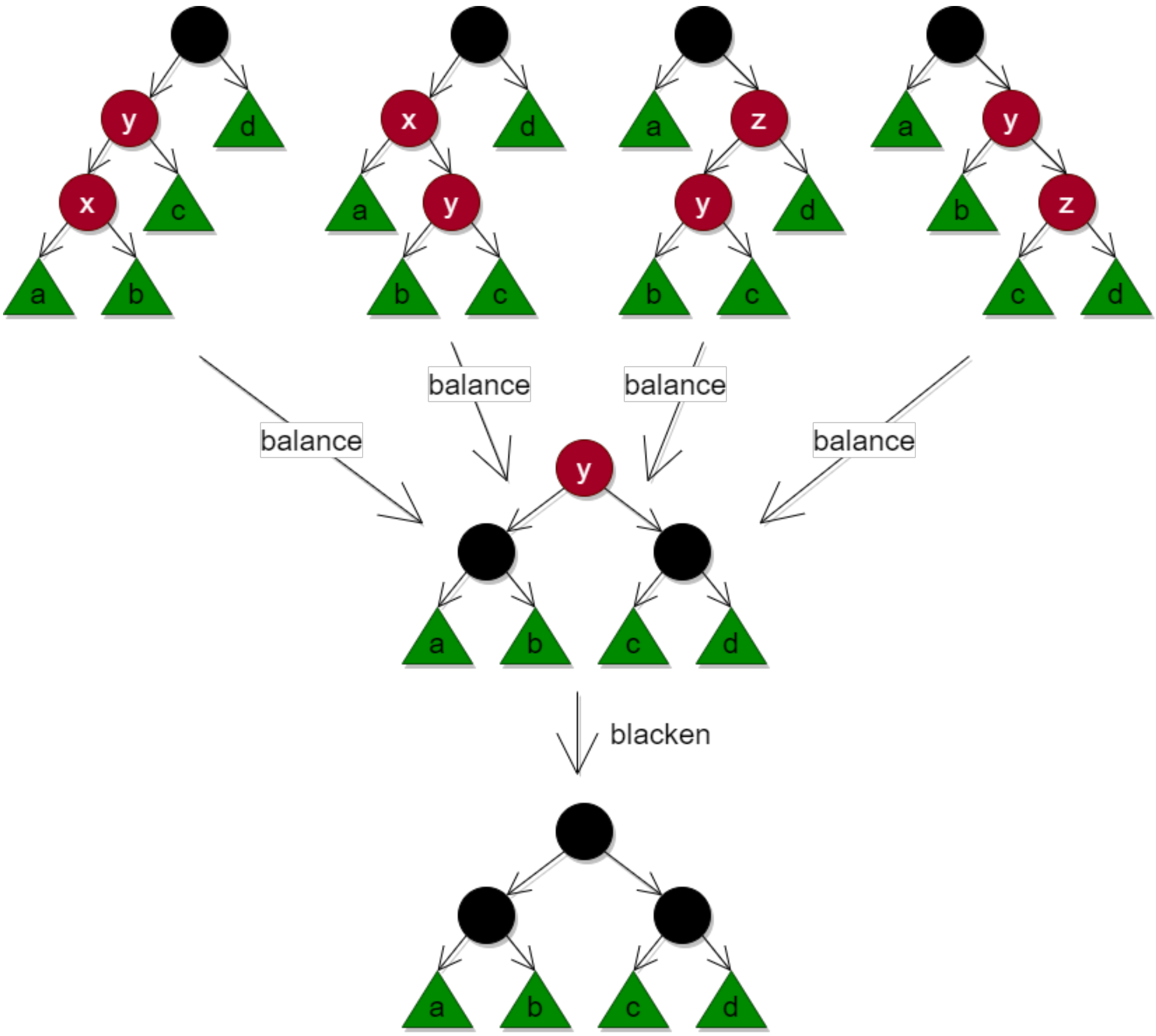


We have an insert function that needs to rebalance the tree:

```
blacken :: Ord a => RBTree a -> RBTree a
blacken (Node Red l x r) = Node Black l x r
blacken t                = t

balance :: Ord a => Colour -> RBTree a -> a -> RBTree a -> RBTree a
balance c l v r = case Node c l v r of
  Node Black (Node Red (Node Red a x b) y c) z d -> bal x y z a b c d
  Node Black (Node Red a x (Node Red b y c)) z d -> bal x y z a b c d
  Node Black a x (Node Red (Node Red b y c) z d) -> bal x y z a b c d
  Node Black a x (Node Red b y (Node Red c z d)) -> bal x y z a b c d
  t                                              -> t
  where
    bal x y z a b c d = Node Red (Node Black a x b) y (Node Black c z d)

insert :: Ord a => a -> RBTree a -> RBTree a
insert = (blacken .) . ins
  where
    ins :: Ord a => a -> RBTree a -> RBTree a
    ins x Empty = Node Red Empty x Empty
    ins x t@(Node c l y r)
      | x < y     = balance c (ins x l) y r
      | x == y    = t
      | otherwise = balance c l y (ins x r)
```



Counting

We can exploit the analogy we used with counting and trees for **RALists** here, with a difference.

Imagine a counting system that lacks zeros. We can count to 10 as:

Normal:	1	2	...	9	10	...	11	12	...	19	20	...	101	102	...	110	111
Special:	1	2	...	9	X	...	11	12	...	19	1X	...	X1	X2	...	XX	111

In this was we can count with binary:

UNFINISHED!!!