Red-black Trees (Arbres rouge-noir / bicolores)

1 From 2-4 Trees to Red-black Trees

« *The red-black trees, invented by Guibas and Sedgewick, are data structures which are types of self-balancing binary search trees. Their particularity is that their nodes are either red or black. These trees are a representation of 2-4 trees and are used only in order to facilitate the implementation of the previous ones.*
In a 2-4 tree, two elements belonging to the same node are called twins. In a binary tree, nodes can only contain one element. This is the usefulness of the color of the node. It allows to distinguish the hierarchy of elements together. If a child node (in the red-black tree) contains a twin element of the one contained in the parent node, ***the child node will be red****. If these elements belong to two different nodes in the 2-4 tree,* ***the child node (in the red-black tree) will be black****.* » (Christophe "Krisboul" Boullay –wiki algo–)

Exercise 1.1 (Properties)

With the above definition, we will try to deduce the properties of red-black trees.

1. Red-black trees corresponding to each kind of nodes in 2-4 trees:
   (a) What does a **2-node** become?
   (b) Give 2 different ways to represent **3-nodes**.
   (c) How **4-nodes** can be represented in order to minimise height?

2. Infer, from 2-4 trees properties and the previous drawings, red-black tree properties.

3. From these properties, describe how the height and the size of a 2-4 tree represented by a red-black tree can be computed.

Exercise 1.2 (Converting to and from red-black trees)

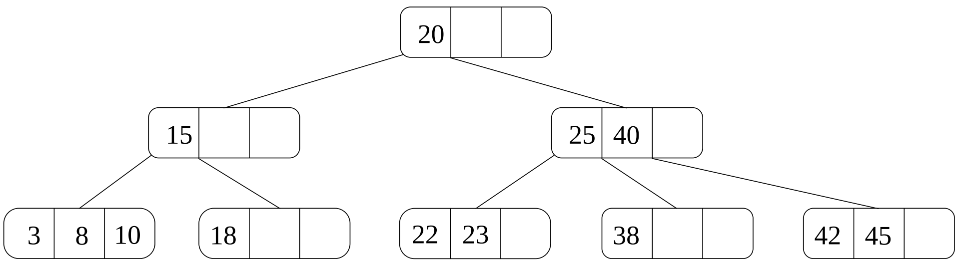

Figure 1: 2-4 tree

1. Draw the red-black tree corresponding to the tree in figure 1. 3-nodes will be balanced on the left.

2. Write the conversion function from 2-4 tree to red-black tree.

3. And the other way?

Exercise 1.3 (Full Test – *Final test - Dec. 2014*)

We want to know if a 2-4 tree represented as a red-black tree (rb-tree) is a full 2-4 tree, meaning each node is a 4-node. We will consider that the red-black tree, given as a parameter, is representing a valid 2-4 tree.

2. Write the function to check if a 2-4 tree represented as a rb-tree is full.

2 Modifications

Exercise 2.1 (Insertions)

1. Insert, in the 2-4 tree from figure 1, keys: $17, 48, 5, 16, 62$ using bottom-up transformation principle and perform the same operation concurrently in the corresponding red-black tree (exercise 1.2). While inserting, answer the following:
   (a) Where, in the red-black tree, are the new keys inserted?
   (b) How could the split operation be translated into a red-black tree?
   (c) What other transformations do we need?
2. Infer the insertion principle for red-black trees.
3. Write the full insertion fonction.

Exercise 2.2 (Deletion (*bonus*))

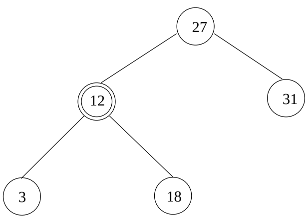

Figure 2:

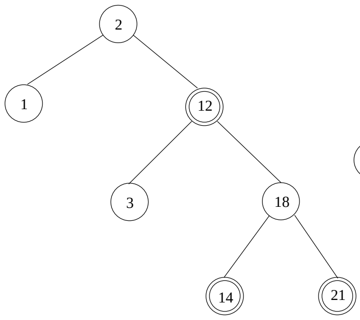

Figure 3:

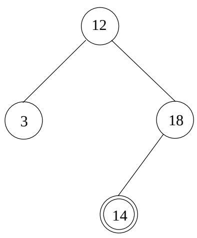

Figure 4:

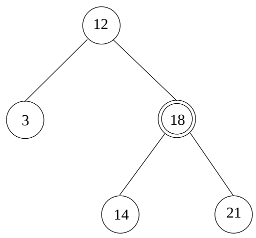

Figure 5:

1. Describe issues when deleting in red-black tree. Linked these problems with deleting in 2-4 trees.
2. Delete key 3 in figure 2, how could you fix the tree? Compare with the solution for 2-4 tree. Generalize this transformation.
3. Delete key 3 in figure 3, how could you fix the tree? Compare with the solution for 2-4 tree. Generalize this transformation.
4. Delete key 3 in figure 4, which transformation can lead to the same configuration as previous case? Generalize.
5. Delete key 3 in figure 5, find which transformation can lead to the same configuration as one of the three previous cases ? Generalize.
6. From all those cases, infer the idea of a delete function in red-black tree.