Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Dmytro Chibisov
Sandeep Sadanandan

Winter Semester 2007/08
Solution Sheet 1
December 05, 2007

Fundamental Algorithms

Problem 1 (10 Points)

Suppose we have a binary search tree with keys in the range from 1 to 1000. We search for key 363. Which of the following cannot represent the sequence of keys of nodes visited during this search?

a $2, 252, 401, 398, 330, 344, 397, 363$

b $924, 220, 911, 244, 898, 258, 362, 363$

c $925, 202, 911, 240, 912, 245, 363$

d $2, 399, 387, 219, 266, 382, 381, 278, 363$

e $935, 278, 347, 621, 299, 392, 358, 363$

Solution

The wrong ones are **(c)** and **(e)**

Reason: While searching in a binary search tree, once we reach a node and decide to continue the search in the left subtree (we are looking for a smaller element than the node), then in the following search, no element which is larger than the node will occur. Similarly, once we have taken a right subtree at a node, a value smaller than that node will never occur in the sequence.

In the case **(c)**, we had already reached 911 once and decided to go for the smaller elements. Then by no means, **912** can occur in the search.

In the case **(e)**, we once take a right subtree at 347. But later in the sequence, we come across **299** which is less than 347. This cannot occur in a binary search tree.

Problem 2

After `insert(x)` or `delete(x)` operations on an AVL tree, the tree needs to be rebalanced using single/double rotations. Show that all the rotations of an AVL tree are made out of two simple operations.

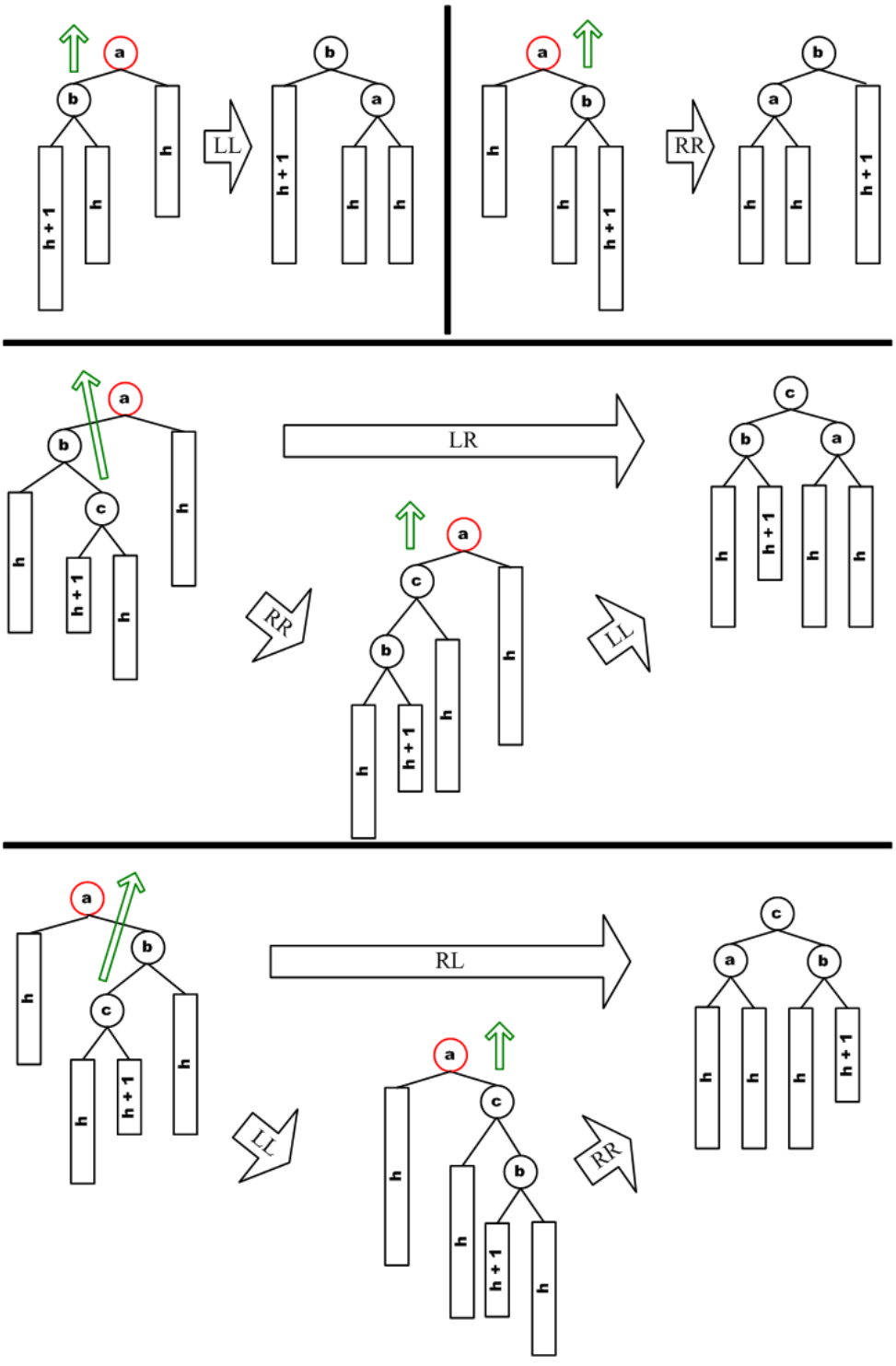


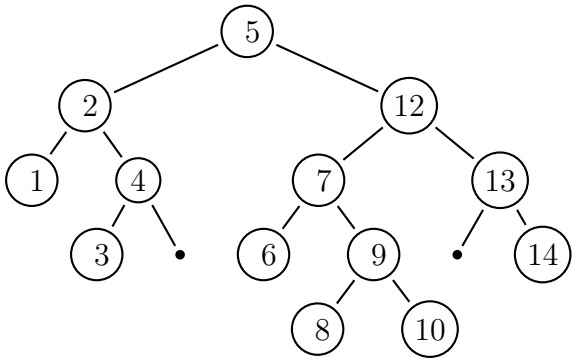
Abbildung 1: Rotations in an AVL tree

Solution

The different types of rotations are given in the picture. From observation it is possible to see that **RR** rotation is the symmetric image of **LL**. Similarly, **RL** and **LR** are also mirror images. The figure also shows how the double rotations can be done by two successive single rotations.

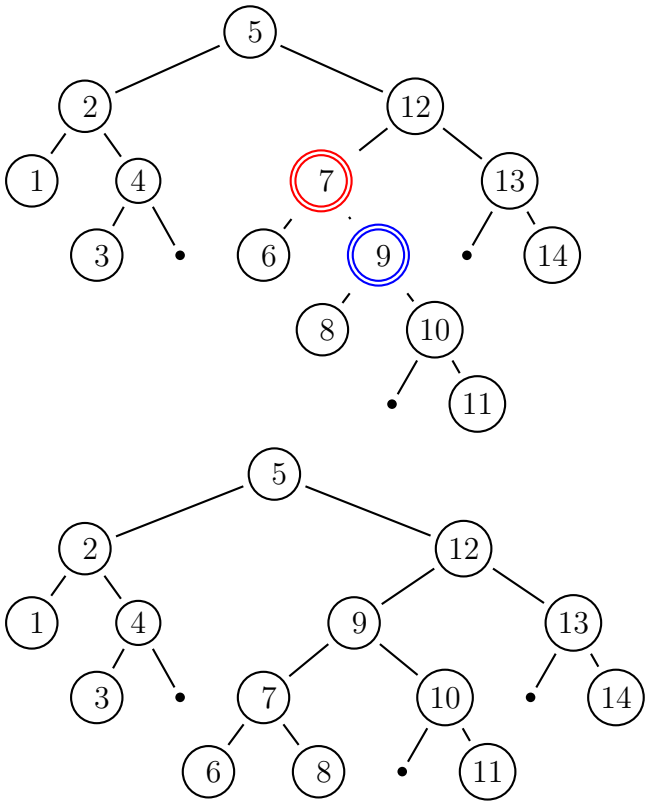
Problem 3

Given is an AVL tree. Perform the operation `insert(11)` on it. Balance the tree.



Solution

The operation `insert(11)` is shown in the figure - step by step.



Problem 4

Prove that an AVL-tree containing n nodes is of height $\Theta(\lg n)$.

Solution

What could be the minimum height of an AVL-tree with n nodes? The tree will be of minimum height when the tree is *balanced*[1].
In that case we know that the height of the tree is of $\Theta(\lg n)$.
So when would be the height maximum for a given number of nodes? This will happen when the tree is more sparse, but satisfy the AVL properties. To find out a relation between the height and the number of nodes let's do the following.
A tree of height h satisfying the above condition can be made by joining a $h-1$ tree and a $h-2$ tree by a single node at the root. So the number of nodes for such a tree of height h is

$$m_h = m_{h-1} + m_{h-2} + 1$$

From observation it could be seen that $m_h = F_{h+2} - 1$ where F_n represents the n^{th} Fibonacci number.
Using the result of the **Problem 2** in **Tutorial 1**, we can see that

$$2^{\frac{h}{2}} \leq m_h < 2^h$$

which implies $h \leq 2 \lg m_h$ and $h > \lg m_h$. Hence h is $\Theta(\lg m_h)$, where m_h is the number of nodes. Hence $h \in \Theta(\lg n)$.

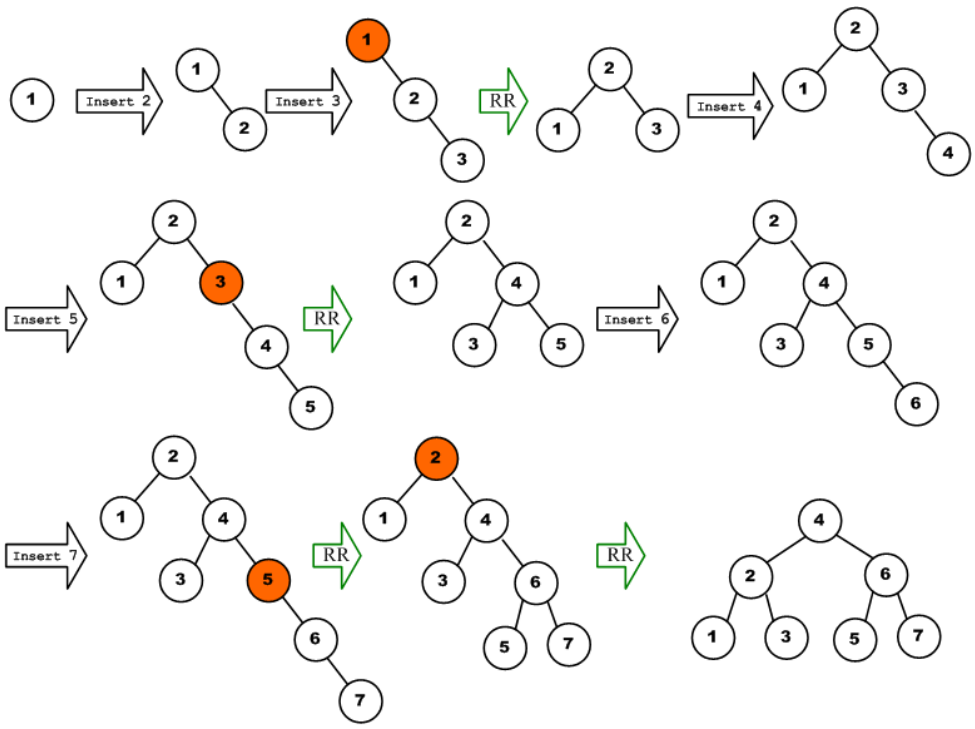
---

[1]

- A Rooted binary tree is a rooted tree in which every node has at most two children.
- A Full binary tree, or proper binary tree, is a tree in which every node has zero or two children.
- A Perfect binary tree (sometimes Complete binary tree) is a full binary tree in which all leaves are at the same depth.
- A Complete binary tree is a tree with n levels, where for each level $d \geq n-1$, the number of existing nodes at level d is equal to $2d$. This means all possible nodes exist at these levels. An additional requirement for a complete binary tree is that for the n^{th} level, while every node does not have to exist, the nodes that do exist must fill from left to right. (This is ambiguous with perfect binary tree.)
- A Balanced binary tree is where the depth of all the leaves differs by at most 1.
- An Almost Complete binary tree is a tree in which each node that has a right child also has a left child. Having a left child does not require a node to have a right child. Stated alternately, an almost complete binary tree is a tree where for a right child, there is always a left child, but for a left child there may not be a right child.
- A Degenerate tree is a tree where for each parent node, there is only one associated child node. This means that in a performance measurement, the tree will behave like a linked list data structure.

  The number of nodes n in a perfect binary tree can be found using this formula: $n = 2^{h+1} - 1$ where h is the height of the tree.

  The number of leaf nodes n in a perfect binary tree can be found using this formula: $n = 2^h$ where h is the height of the tree.



And Finally after inserting till 12, along with RR rotations when necessary, we get..

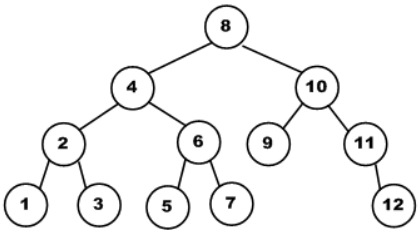


Abbildung 2: Changes happening in inserting 2, ..., 12 on the given AVL tree

Problem 5

On an AVL tree with a single node 1, insert the numbers $2, 3, \ldots, 12$ one by one. Show the balancing.

Solution

The step by step operations are shown in the figure.