Technische Universität München
Institut für Informatik
Theoretical Computer Science

WS 2017/18

Fundamental Algorithms 6 - Solution Examples

Exercise 1 (Flat Trees)

Write an algorithm that copies all keys that are stored in a binary search tree into an array of appropriate size. In the resulting array, the keys shall be sorted in descending order.

Solution:
The main idea of the algorithm is to rely on recursion, processing the input as follows:

1. Copy all keys of the right subtree to the array
2. Copy the root of the tree into the array
3. Copy all keys of the left subtree to the array

The algorithm can be written as:

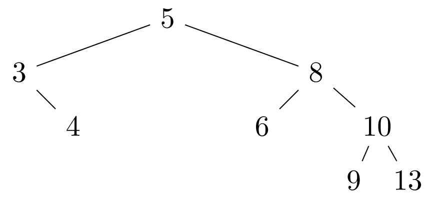
```
Algorithm 1: Tree2Array
  Input: $T$: BinaryTree
         $A$: Array[1..$n$]
         $pos$: Current write position
  Result: New write position
  if isEmpty($T$) then
      return pos;
  end
  $pos \leftarrow$ Tree2Array(right($T$), $A$, $pos$);
  $A[pos] \leftarrow$ key($T$);
  $pos \leftarrow$ Tree2Array(left($T$), $A$, $pos + 1$);
```

Exercise 2 (Growing Trees)

Consider the binary tree given by the expression

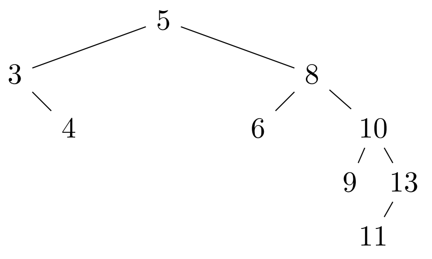
$$x = (5, (3, \emptyset, (4, \emptyset, \emptyset)), (8, (6, \emptyset, \emptyset), (10, (9, \emptyset, \emptyset), (13, \emptyset, \emptyset))))$$

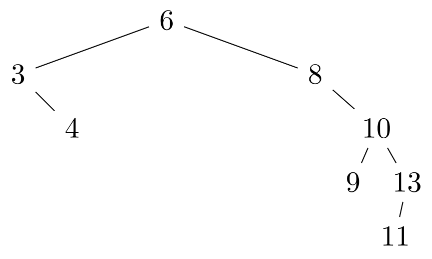
1. Draw a diagram of this binary tree and decide whether it is a binary search tree.

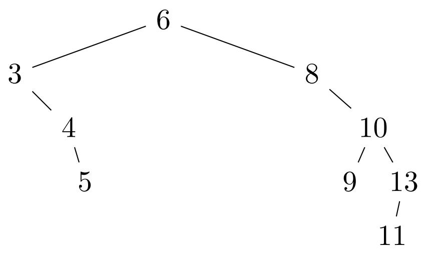


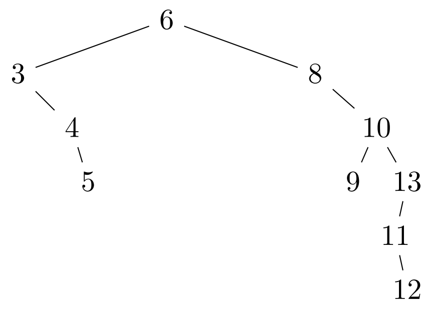
For each node, all keys in the left subtree are smaller than that in the node, and all keys in the right subtree are larger. Hence, the tree is a binary search tree.

2. Perform the following operations (using the algorithms from the lecture) and draw a diagram of the search tree after each operation: INSERT$(x, 11)$; DELETE$(x, 5)$; INSERT$(x, 5)$; INSERT$(x, 12)$









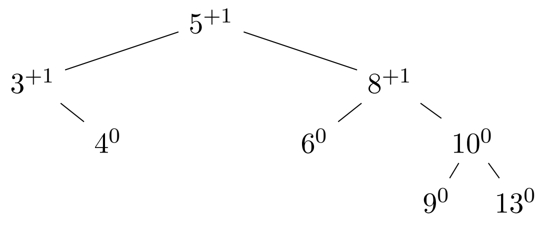
Exercise 3 (AVL Trees)

Decide whether the binary tree given in Exercise 2 is an AVL tree

- before the insert/delete operations, and
- after each of the regular insert and delete operations.

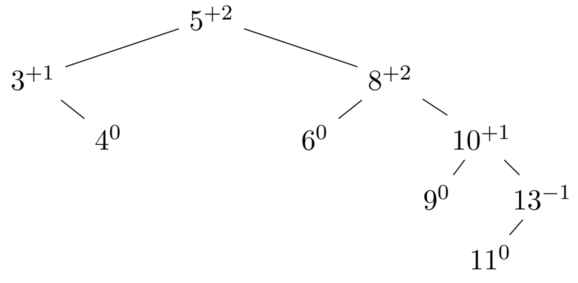
Again, perform the insert/delete operations given in Exercise 4, and, if required, perform the rotation(s) to restore the AVL property after each step. Draw a diagram of the search tree after each of your insert, delete, and rotation operations.

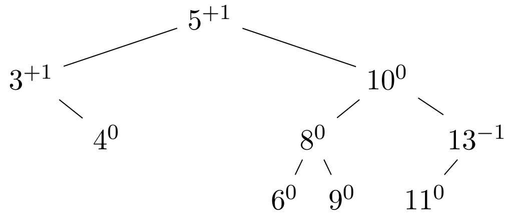
Solution:
Before the insert/delete operations, the height balances for the nodes are:



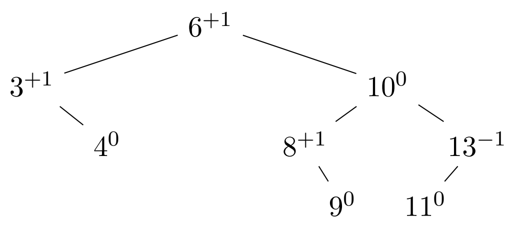
Therefore, the binary search tree is also an AVL tree.

- INSERT$(x, 11)$ Violated in node 8 $\rightarrow$ Left-rotation:

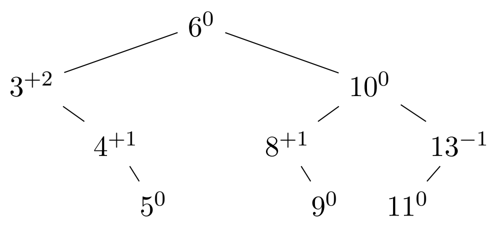


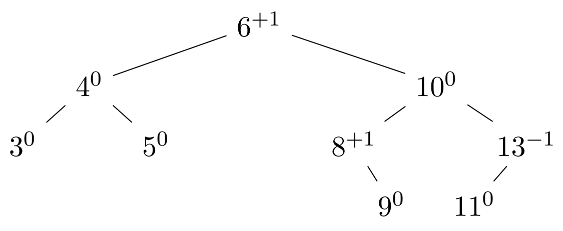


- DELETE$(x, 5)$: Satisfied → No rotation required:

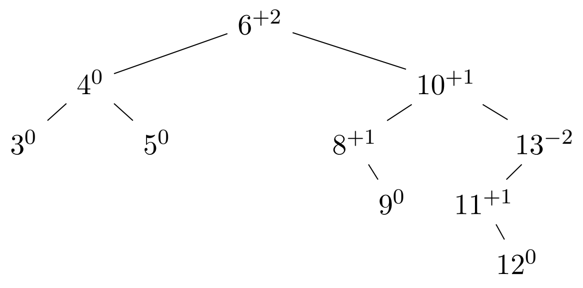


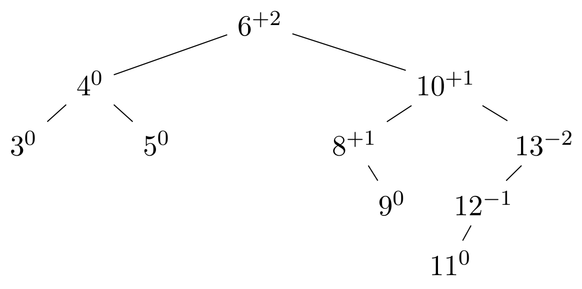
- INSERT$(x, 5)$: Violated in node 3 → Left-rotation:

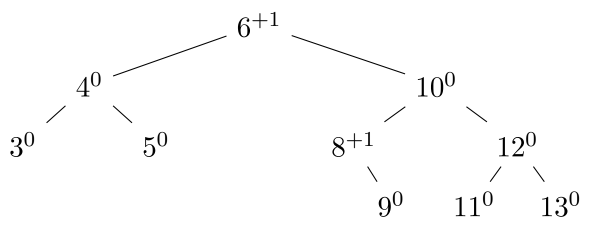




- INSERT$(x, 12)$: Violated in node 13 → Left-right-rotation:


6^{+2}
4^{0}
10^{+1}
3^{0}
5^{0}
8^{+1}
13^{-2}
9^{0}
11^{+1}
12^{0}


6^{+2}
4^{0}
10^{+1}
3^{0}
5^{0}
8^{+1}
13^{-2}
9^{0}
12^{-1}
11^{0}


6^{+1}
4^{0}
10^{0}
3^{0}
5^{0}
8^{+1}
12^{0}
9^{0}
11^{0}
13^{0}