Data Structures and Algorithms **Spring-Summer 2011/12**

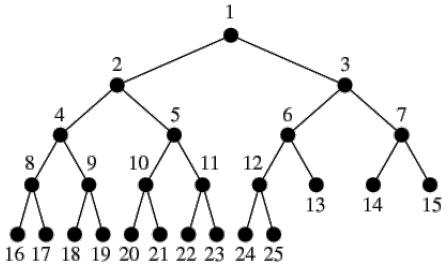
Assignment 7

Valeria Fionda, Mouna Kacimi,
Werner Nutt, Simon Razniewski

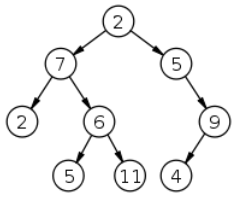
Binary Search Trees and Red-Black Trees

Instructions: Your assignment should represent your own effort. However, you are not expected to work alone. It is fine to discuss the exercises and try to find solutions together, but each student shall write down and submit his/her solutions separately. It is good academic standard to acknowledge collaborators, so if you worked together with other students, please list their names.
You must be prepared to present your solution at the lab. If you are not able to explain your solution, this will be considered as if you had not done your work at all.
You can write up your answers by hand (provided your handwriting is legible) or use a word processing system like Latex or Word. Experience shows that Word is in general difficult to use for this kind of task.
For a programming task, your solution must contain (i) an explanation of your solution to the problem, (ii) the Java code, in a form that we can run it, (iii) instructions how to run it. Also put the source code into your solution document. For all programming tasks, it is not allowed to use any external libraries ("import") or advanced built-in API functions (for example, `String.indexof("a")` or `String.substring(1,5)`), if not stated otherwise.
Please, include name, matriculation number and email address in your submission.

Preliminaries: These exercises are about binary and we need a way to specify binary trees. We will use the following schema for numbering the positions of complete binary tree:



That is, the nodes are numbered as one would encounter them when traversing the tree breadth-first.
Next, we want to specify arbitrary binary trees, whose nodes contain numbers and can be coloured. The specification of a tree is a sequence of triplets. We will use this notation to specify trees by which you should test the algorithms you develop. Each triplet has the form $\langle v, c, pos \rangle$, where v is a number, the value of the node, c is the colour, where $c = nil$ if the tree is a uncoloured binary tree, or $c \in \{r, b\}$ if the tree is a red-black tree. For example the sequence $\langle 2, nil, 1\rangle, \langle 7, nil, 2\rangle, \langle 5, nil, 3\rangle, \langle 2, nil, 4\rangle, \langle 6, nil, 5\rangle, \langle 9, nil, 7\rangle, \langle 5, nil, 10\rangle, \langle 11, nil, 11\rangle, \langle 4, nil, 12\rangle$ represents the following uncoloured binary tree:



1. Binary Search Trees

In this exercise you are asked to:

(1) Provide an algorithm in pseudo code that implements the insertion of an integer into a binary search tree. Then, use this algorithm as as part of another algorithm, `constructBST`, that reads in a sequence of integers and constructs a binary search tree containing exactly those integers.
Note: The input consists of integers, not the triples defined above.

(2) Provide an algorithm in pseudo code, `printBST`, that receives as input a binary search tree T and produces as output a sequence of integers that is a permutation of the values of the nodes of T. The output sequence should be such that, if it is fed as input to the algorithm `constructBST` in part (1), the obtained tree is equal to T. Explain why your algorithm has this property.

(3) Implement the two algorithms developed in part (1) and (2) in Java.

(4) Prove that one way to construct printBST is to let it traverse the binary search tree T using breadth first search.
Hint: There is an easier way than this to write an algorithm `printBST`.

(15 Points)

2. Red Black Trees

In this exercise you are asked to:

(1) Provide an algorithm in pseudo code that given a tree T, the nodes of which are coloured red or black, checks if T is a red-black tree. The proposed algorithm should be as fast as possible and, in case it detects that the input tree is not a red-black tree, it should provide the motivation of its finding (e.g., a particular node is coloured with the wrong colour).
Note: Such an algorithm would be useful if one wants to implement red-black trees.

(2) Implement the algorithm in part (1) in Java.

(3) Extend the Java version of `insertBST` to an algorithm `insertBSTC` (where "`C`" stands for "colour"), which takes as input a sequence of numbers and colours $c \in \{ r, b \}$. A possible input sequence could be

```
3 b, 7 b, 4 r, 5 b, 2 r, 9 r
```

Use `insertBSTC` to build the trees associated to the following sequences:

- $\langle 13, b, 1\rangle, \langle 8, r, 2\rangle, \langle 17, r, 3\rangle, \langle 1, b, 4\rangle, \langle 11, b, 5\rangle, \langle 25, b, 7\rangle$
- $\langle 13, b, 1\rangle, \langle 8, r, 2\rangle, \langle 17, r, 3\rangle, \langle 1, b, 4\rangle, \langle 11, b, 5\rangle, \langle 25, r, 7\rangle$
- $\langle 13, b, 1\rangle, \langle 8, r, 2\rangle, \langle 17, r, 3\rangle, \langle 1, b, 4\rangle, \langle 11, b, 5\rangle, \langle 25, b, 7\rangle, \langle 12, b, 11\rangle$
- $\langle 13, b, 1\rangle, \langle 8, r, 2\rangle, \langle 17, r, 3\rangle, \langle 1, b, 4\rangle, \langle 7, b, 5\rangle, \langle 25, b, 7\rangle$

Then, check with the algorithm developed in part (1) if they are red-black trees.

(12 Points)

3. Is Deletion Commutative?

Is the operation of deletion "commutative" in the sense that deleting x and then y from a binary search tree leaves the same tree as deleting y and then x? Argue why it this is the case or give a counterexample.

(3 Points)

Submission: Tue, 8 May 2012, 8:30 am. Preferably by email.

If you want to submit a hand-written solution, hand it over to your lab tutor or in the lecture. However, all code has to be submitted in electronic form.