HOMEWORK 8

ALGORITHMS AND DATA STRUCTURES (CH08-320201)

Spring 2018

Prof. Dr. Michael Sedlmair
Computer Science & Electrical Engineering
Jacobs University

Due on Monday, Apr 16, 2018, 23:55.

Problem 1: Understanding Red Black Trees *(3+3+3* = 9 points)*

Make sure you submit your solution in the specified format. Points will be deducted if your solution does not follow the format. For a short explanation of pre-order tree traversal, check this assignment's last page.
FORMAT FOR A) AND B): Write your solutions into the text files `solution_a.txt` and `solution_b.txt`, respectively. Each tree is to be represented using pre-order traversal. Separate individual values within trees with single spaces. Separate different trees with new lines. Each node's value needs to be immediately followed by 'B' or 'R', representing the node's color. Do not add `nil` nodes. Example: `17B 12R 7R`. You are welcome to also upload your detailed solution (drawn trees), into `solution_detailed.pdf`. Grading will be based on the `.txt` files that you submit, but in case they are incorrect, we can give you partial points for the detailed solution.
FORMAT FOR C): Please submit your argumentation in the file `solution_c.pdf`.

(a) Describe all valid red-black trees that store the numbers 1,2,3,4,5.

(b) Describe the red-black trees that result after successively inserting the keys [14, 42, 35, 7, 26, 17] into an empty red-black tree. You are required to describe the tree after each insertion, as well as any additional recoloring and balancing.

(c) **Bonus.** Consider a red-black tree formed by inserting n nodes with the algorithm described in class. Argue that if $n > 1$, the tree contains at least one red node.

Problem 2: Implementing Red Black Trees *(16 points)*

Implement a Red Black Tree (with integer nodes), closely following the specifications and algorithms from the lecture. Write your code either in C++ or Python - no other languages will be accepted. Save your solution to `rbt.py` for Python and `rbt.cpp` (header in `rbt.h`) for C++. Make sure you handle errors appropriately.
Grading will be based on case-tests, therefore make sure that you use the exact naming of functions and classes below. We will test all functions, classes, structs that are listed below. Feel free to define any number of helper functions.

```cpp
enum Color { RED, BLACK };

struct Node
{
    int data;
    Color color;
    Node *left, *right, *parent;
};

class RedBlackTree
{
    private:
        Node *root;
    protected:
        void rotateLeft(Node*&);
        void rotateRight(Node*&);
    public:
        RedBlackTree();
        void insert(int);
        void delete(Node*&);
        Node* predecessor(Node*&);
        Node* successor(Node*&);
        Node* getMinimum();
        Node* getMaximum();
        Node* search(int);
};
```

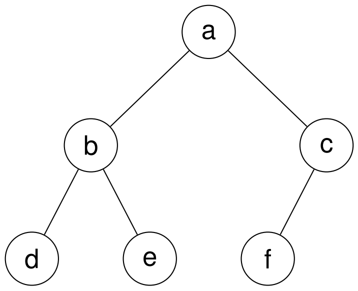
Listing 1: Methods and classes to implement for a Red Black Tree

Background: Pre-order tree traversal

Trees can be represented with the pre-order traversal.

```
function preorder(tree):
     1. Write down tree.value.
     2. Call preorder(tree.left_child)
     3. Call preorder(tree.right_child)
```

The following tree is represented via pre-order traversal by: `a b d e c f`.



Remarks

- Solutions have to be handed in via Moodle by the due date. For late submissions you need to get in contact with the TAs directly.
- You need to upload the files specified in the problem statements. The PDF file `solution_detailed.pdf` can contain the solutions for your entire homework. Programming assignments need to be handed in as **either C++ or Python code**. Please write your own code. It is ok to take snippets from online resources, but they need to be clearly marked.
- Exercises marked with a * are bonus problems. These count towards your number of points for this homework. The maximum number of official points can not be exceeded.