# **Algorithms and Data Structures**

## Homework 8

### Dragi Kamov

### 14 April 2018

## Problem 1

1. These are all the valid red-black trees that store the numbers 1, 2, 3, 4, 5

2B 1B 4B 3R 5R

3B 2B 1R 5B 4R

3B 2B 1R 4B 5R

4B 2B 1R 3R 5B

3B 1B 2R 5B 4R

3B 1B 2R 4B 5R

1. Adding 35

Current color red

35 changed to black

Adding 17

Current color red

35 changed to black

Adding 26

Current color red

Rotating left 17

26 changed to black

35 changed to red

Rotating right 35

26 changed to black

Adding 7

Current color red

17 changed to black

35 changed to black

26 changed to red

26 changed to black

Adding 42

Current color red

26 changed to black

Adding 14

Current color red

Rotating left 7

14 changed to black

17 changed to red

Rotating right 17

26 changed to black

26B 14B 7R 17R 35B 42R

1. Let’s consider that we have a red-black tree with only one node (black, because it is the root). When we would add the second node the default inserting color would be red according to my algorithm, and even if we didn’t have the red as a default color according to property 5 for red-black trees (For each node all paths from the node to a leaf have the same number of black nodes) the second node would have to be a red node. Inserting the third node again would have the default color red. Now let’s say that we don’t have the default color as red, and let’s also say that with this insertion we would need to rotate and the red color goes up in the root (which would be fixed later to black, because the root has to be black according to property 2) we could come to a tree where all nodes are black, but that’s why we have the default color as red so in this situation we would have two red nodes and black root.

## Problem 2

My code for this problem is in “rb-tree.py”. In the printing of the tree I am also printing before each node a letter “L” or “R” indicating whether the node is a left node or a right node.