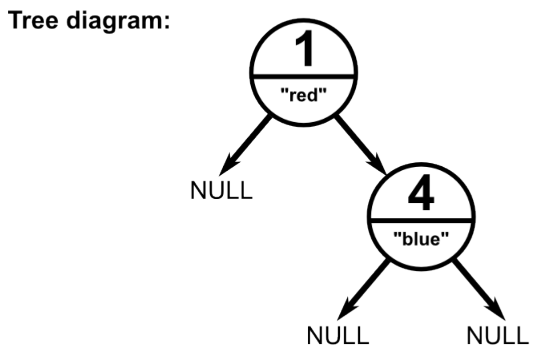
Exercise sheet 10

February 18, 2020

Exercise 1 *(14 points)*
Implement a class *BinarySearchTree* with a binary search tree, such that the keys are of type *int* (integer) and the elements of type *str* (string).
Please note that only *insert* (6 points) and *lookup* (4 points), but not *remove* needs to be implemented. This means that you can omit the doubly linked lists between the nodes in your tree.
As usual, write some (useful) tests for your methods. For this purpose write one *to_string* method (4 points) which outputs a string representation of your binary tree (see Figure 1 for illustration).
Exercise 2 *(6 points)*
Write a program that inserts the numbers $1, 2, \ldots, n$ in this order in a (at the beginning) empty *BinarySearchTree.*
Next, insert in a (again, at the beginning) empty *BinarySearchTree* the same n numbers in random order.
Measure the runtime and the depth of the tree for both cases (after each n insertions) for $n = 2^{10}, 2^{11}, 2^{12}, \ldots$. Increase n so that the runtime is still bearable on your machine. Shortly discuss the results in your *erfahrungen.txt.*
Commit
Commit your code into the SVN in a new subdirectory **uebungsblatt_10** and a PDF with the solutions of the theoretical tasks in the same folder. Commit your feedback in a text file *erfahrungen.txt* as usual. Please specify: the length of time needed for the exercise. Which tasks have been difficult for you and where did you have problems? How much time did you spend to solve the problems?



tree.to_string() returns:

```
[(1, "red"), left: null, right: [(4, "blue"), left: null, right: null]]
```

Figure 1: example tree diagram and corresponding output string of the to_string method.