Alessandro Artale, Werner Nutt,
Ognjen Savković, Francesco Sportelli

11. Algorithms for Trees

In this lab, you are going to develop algorithms for trees. Whenever, you work on an algorithm, proceed by the following steps:

1. Write down in words your idea for the algorithm.
2. Write up your algorithm in pseudocode.
3. Determine the asymptotic complexity of your algorithm.

1. Number of Nodes in a Tree

Develop code for a function

```
int numberOfNodes(),
```

which returns the number of non-null nodes in the current tree.

Hints: Solving this problem recursively may be the best approach. If you want to do that, you will have to define an auxiliary function.

2. Depth of a Tree

Develop code for a function

```
int depth(),
```

which returns the depth of the current tree. Remember that the depth of a tree is the length of the longest path from the root to a (non-null) leaf.

Hint: Note that we have not defined the depth of an empty tree, where the root is null. What would be a reasonable extension of the definition so that it covers also this case?

3. Breadth-first Traversal of a Tree

In the lecture, you have seen how to traverse a tree in preorder, inorder, and postorder.
An alternate technique to traverse a tree proceeds by levels, where the level of a node is defined as the length of the path from the root to that node. An algorithm for such a level-order traversal visits first all nodes a level 0 (i.e., the root), then those at level 1 (i.e., the children of the root), after that those at level 2 (i.e., the grandchildren of the root), and so on. Level-order traversal is also called *breadth-first* traversal.
Develop a method

```
void breadthFirstTraversal(),
```

that prints out the keys of a tree in the order of a breadth-first traversal.

Hint: Preorder, inorder, and postorder traversal can easily be expressed using recursion, which is executed by the Java engine using a stack with all the pending function calls. The reason is that children of a node are processed according to "last in, first out" queuing policy. What is the queuing policy for breadth-first search? How can one implement it?

4. Serialization of Binary Search Trees

Data structures with pointers are often difficult to store in a file. By a serialization of such a structure we understand a format that can easily be written to a file and easily be used to rebuild the original structure. Here, we want to serialize binary search trees (BSTs) that contain integer keys.
Clearly, if we are given an arbitrary file containing a sequence of integers, then we can construct a BST by starting with an empty tree and inserting the integers one after the other according to the order on integers.

Task: In this exercise, you are asked to develop a method for *writing* the keys in a BST T on a file F in such a way that by reading the integers in F and inserting them into an empty tree the original tree T is reconstructed.