UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

COS212: Practical 3

Release: Monday 25 February 2019, 18:00
Deadline: Tuesday 26 February 2019, 18:00

Objectives

The aim of this practical is to learn how to implement and use binary search trees.

Instructions

Complete the task below. Certain classes have been provided for you in the *files* zip archive of the practical. You have also been given a main file which will test some code functionality, but it is by no means intended to provide extensive test coverage. You are encouraged to edit this file and test your code more thoroughly. Remember to test boundary cases. Upload **only** the given source files with your changes in a zip archive before the deadline. Please comment your name **and** student number in at the top of each file.

Task 1: Binary Search Trees [35]

A binary search tree is a variant of the binary tree that makes a binary search possible. For each node x of the tree, all values stored in its left subtree are less than the value of x, and all values stored in the right subtree are greater than the value of x. This allows a complexity between *O(log n)* and *O(n)* for search, insert and delete operations within a sequence of `n` elements. The shape of the tree determines the complexity, which can change when the tree is updated.

You have been given a partially implemented binary search tree class and a binary search tree node class to use. Your task is to implement the following methods in the binary search tree class according to the given specification:

```
boolean isEmpty()
```

This function should determine whether the binary search tree is empty or not. It returns true if the binary search tree is empty and false otherwise.

```
T minValue()
```

This function should determine the smallest element in the binary search tree and return it. If the tree is empty, the function should return `null`.

```
T maxValue()
```

This function should determine the greatest element in the binary search tree and return it. If the tree is empty, the function should return `null`.

```
void printPostorder()
```

This function should print out all the elements in the binary search tree as a space separated list that is terminated by a newline. This function should use the post-order tree traversal strategy.

`void insert(T elem)`

This function should insert the given `elem` in the binary search tree. `elem` should be inserted in the correct position in the binary search tree to maintain the sorted order required by a binary search tree.

`boolean deleteByCopy(T elem)`

This function should delete the given `elem` from the binary search tree. It returns true if `elem` is removed successfully and false otherwise. This function should use the delete by copying strategy.

`T search(T elem)`

This function should find the given `elem` in the binary search tree. If `elem` is found in the binary search tree it should return the element and otherwise it should return `null`.

Only implement the methods listed above. You can use both iterative and recursive approaches. Do not modify any of the other code that you were given for this task.

Submission

You need to submit your source files on the Assignment website (assignments.cs.up.ac.za). All methods need to be implemented (or at least stubbed) before submission. Place all the source files including a makefile in a zip or tar/gzip archive named uXXXXXXXX.zip or uXXXXXXXX.tar.gz where XXXXXXXX is your student number. You have 24 hours to finish this practical, regardless of which practical session you attend. Upload your archive to the *2019 Prac 3 - Tuesday* slot on the Assignment website. Submit your work before the deadline. **No late submissions will be accepted!**