UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

COS212: Practical 4

Release: Monday 4 March 2019, 18:00
Deadline: Tuesday 5 March 2019, 18:00

Objectives

The aim of this practical is to learn how to implement a self-adjusting tree using the *semi-splaying* strategy. (See section 6.8.2 of course textbook)

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: Splay Tree using the *semi-splaying* strategy [32]

A Splay Tree is a self-adjusting Binary Search Tree where accessed elements are moved to the root by means of *splaying*. This involves applying rotations in pairs based on the configuration of the accessed node, its parent, and its grandparent (see section 6.8.2 of textbook). This has the advantage of flattening the tree and moving the accessed element to the root. You will be required to implement a Splay Tree using the *semi-splaying* strategy as described on page 273 of the textbook.

Implement the following methods according to the given specification:

`void insert(T key)`

Insert the given key into the tree. Duplicate keys should be ignored. No Splaying should take place during insert.

`Boolean contains(T key)`

If the key exists in the tree, return `true`, otherwise return `false`.

`void access(T key)`

If the key exists, apply the *semi-splaying* strategy to move the key closer to the root. If the key does not exist, add it the tree without splaying.

`T findSuccessor(T key)`

Return the successor of the given key. If the key does not exist or does not have a successor, return `null`. This method should not modify the structure of the tree.

You may use your own helper functions to assist in implementing the specification. However you may not modify any of the given method signatures.

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. There should be no folders in your archive. You have 24 hours to finish this practical, regardless of which practical session you attend. Upload your archive to the *2019 Prac 4 - Tuesday* slot on the Assignment website. Submit your work before the deadline. **No late submissions will be accepted!**