UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

COS212: Practical 4

Release: Monday 3 May 2021
Deadline: Friday 7 May 2021, 18:00

PLAGIARISM POLICY

UNIVERSITY OF PRETORIA

The Department of Computer Science considers plagiarism as a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with consent) and copying material (such as text or program code) from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to `http://www.library.up.ac.za/plagiarism/index.htm` (from the main page of the University of Pretoria site, follow the *Library* quick link, and then choose the *Plagiarism* option under the *Services* menu). If you have any form of question regarding this, please ask one of the lecturers, to avoid any misunderstanding. Also note that the OOP principle of code re-use does not mean that you should copy and adapt code to suit your solution.

Objectives

The aim of this practical is to learn how self-adjusting trees work and how to implement this functionality using different strategies.

Instructions

Complete the tasks 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 compressed archive before the deadline. Please comment your name **and** student number in at the top of each file. Do not import any library. No makefile required.

Splay Trees - Splaying and Semi-splaying

Splay trees are a kind of self-adjusting binary search tree. The structure of a splay tree is modified during use so that elements that are accessed frequently are either near the root or at the root, and elements that are accessed infrequently are further down. This is done by either splaying, performing a series of rotations that eventually moves a given node to the root of the tree, or by semi-splaying, performing one or more rotations that moves a given node towards the root of the tree. You are required to implement a splay tree. Every time that a node is accessed, that node should be moved to the root by means of splaying, as described on page 268 of the textbook, or that node should be moved up towards the root of the tree by means of semi-splaying, as described on page 273 of the textbook.

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

Task 1: Insert Element [10]

```
boolean insert(T elem)
```

This function should insert the given element into the tree, but only if it is not already in the tree. Returns true if the element was successfully inserted into the tree and false otherwise. The insert should follow normal binary search tree rules of having larger elements on the right and smaller elements on the left.

Task 2: Contains Element [4]

```
boolean contains(T elem)
```

This function should determine whether the given element is present in the tree. Returns true if the element is found in the tree and false otherwise.

Task 3: Access Element [36]

```
void access(T elem, SplayType type)
```

This function should first determine whether the given element is in the tree. If so, the function should use the provided enumeration `SplayType` to determine which self-adjusting strategy type has been given to use on the tree to move the node with the given element. The appropriate method below should then be invoked to adjust the tree. If the given element is not found in the tree, it should simply be inserted into the tree (with no splaying or semi-splaying).

Task 3.1: Splay Tree

```
void splay(Node<T> node)
```

This function should splay the tree so that the given node is the root of the tree when the function returns.

Task 3.2: Semi-splay Tree

```
void semisplay(Node<T> node)
```

This function should semi-splay the tree so that the given node is moved towards the root of the tree when the function returns.

Only implement the methods listed above. You may use your own helper functions, such as methods for rotating left and rotating right, to assist in implementing the specification. However, you may not modify any of the given method signatures. Also 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 (https://ff.cs.up.ac.za/). All tasks need to be implemented (or at least stubbed) before submission. Place all the source files in a zip or tar/gzip archive (you need to compress your tar archive) named uXXXXXXXX.zip or uXXXXXXXX.tar.gz where XXXXXXXX is your student number. You have 4 days to complete this practical, regardless of which practical session you attend. Upload your archive to the *Practical 4* slot on the Assignment website. Submit your work before the deadline. No late submissions will be accepted.