UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

COS212: Practical 5

Release: Monday 17 May 2021, 18:00
Deadline: Friday 21 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 to implement insertion into B$^+$-trees.

Instructions

Complete the task below. Certain classes have been provided for you alongside this specification in the *Student files* folder. You have 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. Submission instructions are given at the end of this document.

Task 1: B$^+$-trees [52]

A B$^+$-tree is a variant of B-tree which allows for fast in-order traversal through its leaf nodes. B$^+$-trees store all of their data elements in the **leaves** of the tree. Leaves are linked sequentially in a linked-list structure (called a *sequence set).* Traversing the sequence set yields all the elements in the tree in ascending order. The nodes which are **not** leaves contain indices (and thus form part of the *index set*). The index set utilises conventional B-tree nodes.

The parents of the leaf nodes may contain copies of the keys in their children. If a parent contains a copy of a key, then it is the first key in the child to the right of the separator key. This arrangement **only** holds between the leaf nodes and their parents. As mentioned previously, the index set is a normal B-Tree and therefore parent nodes of non-leaf nodes do not contain copies of keys from their children. Refer to **Section 7.1.3** in the textbook for the necessary details. In addition, you may utilize the B$^+$-tree visualizer provided by the University of San Francisco to assist you in understanding the B$^+$-tree (click here for link). No guarantees are made that the online visualizer is entirely correct. You may assume that only B$^+$-trees of order $m \geq 3$ will be tested. Consult **Section 7.1.1** in the textbook for what the order of a B-tree implies (which is also relevant for the B$^+$-tree).

You have been given a partially implemented B$^+$-tree class and a B$^+$-tree node class to use. Only `int` elements will be used in this practical. Your task is to implement the following methods in the B$^+$-tree class according to the given specification:

```
void insertElement(int element)
```

This function should insert the given integer `element` into the B$^+$-tree. You may assume that duplicates will not be inserted into your tree. Remember to maintain the references in the sequence set correctly and ensure that the `root` field of the `BPlusTree` class remains up-to-date as the tree expands. Importantly, the B$^+$-tree should be **left-biased** on splitting nodes in the *index set*. This means if a node in the *index set* can not be split perfectly, the new right node will have one less node than the left node (i.e. the left node in the split

retains the extra element). Imperfect splits which happen in the *sequence set* are **right-biased** because the left-most element of the new right node in the split needs to be copied to the parent node. An extensive example output is given in the **Main.java** in order for you to test your code. Some examples are given in Figure 1 and 2 below.

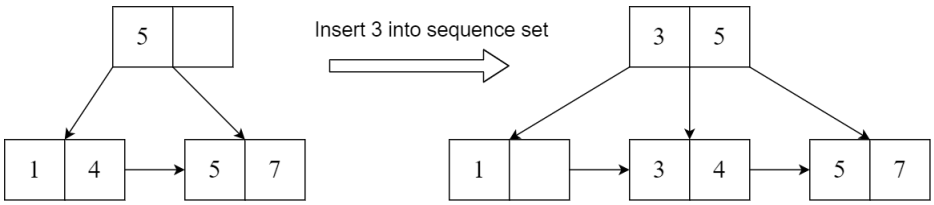

Figure 1: B^+-tree of order 3. Notice how the extra element in the split is in the new right node in this imperfect split. If the tree had an even order, there would be the same number of elements in both the left node and the new right node in the split (with the first element in the new right node still copied to the parent).

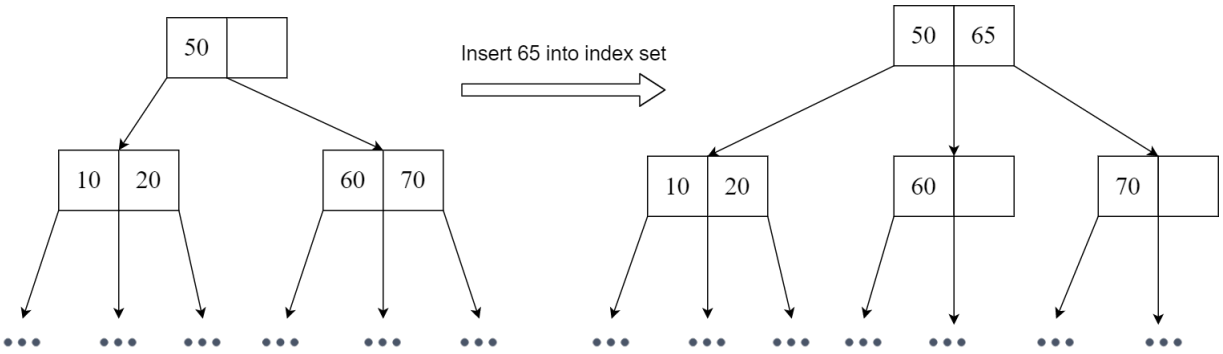

Figure 2: B^+-tree of order 3. Notice how keys are not copied into their parents in the index set because the index set is a normal B-tree. This is a perfect split because the order is odd. If the tree had an even order, the split would be imperfect and the **left** node in the split would retain the extra element. The nodes further down in the tree are omitted in this example.

`BPlusNode getFirstLeaf()`

This function should find and return the first leaf (i.e. the left-most leaf) in the tree. If the tree is empty, return `null`.

Implement the methods listed above. It is **highly recommended** that you use your own helper functions 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. You may create subclasses of the BPlustNode class if you desire (if you do, you should also upload these new files). Importing will result in a mark of zero.

Submission

You need to submit your source files on the Fitch Fork website (https://ff.cs.up.ac.za/). All methods need to be implemented (or at least stubbed) before submission. Place your **BPlusTree.java** and **BPlusNode.java** file in a zip archive named uXXXXXXXX.zip where XXXXXXXX is your student number. There is no need to include any other files in your submission. You have 4 days to complete this practical, regardless of which practical session you attend. You have 5 submissions and your best mark will be your final mark. Upload your archive to the *Practical 5* slot on the Fitch Fork website. Submit your work before the deadline. **No late submissions will be accepted!**