UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

COS212: Practical 5

Release: Monday 01 April 2019, 18:00
Deadline: Tuesday 02 April 2019, 18:00

Objectives

The aim of this practical is to learn how B-Trees work and how to implement this functionality.

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 zip archive before the deadline. Please comment your name **and** student number in at the top of each file.

B-Trees

A B-Trees is a kind of balanced search tree. Like binary search trees it allows data to be kept sorted while allowing searches, insertions and deletions. Unlike a BST, it allows a node to have more than two children. Each node also contains a number of keys that contain pointers to the actual data. The B-Tree for this practical will have a maximum of *k-1* keys and *k* child nodes where k is defined as follows: m $\leq$ k $\leq$ 2m.

You are required to implement some of the B-Tree methods. Every time that a key is inserted, searching for the correct position should start at the root and propagate down to the correct location. Inserts are only performed on leaf nodes. Each full node encountered on the way down should be split. This is called the top-down pre-splitting insertion strategy, as mentioned on page 323 of the textbook. You have been given a functional B-Tree class and a partially implemented B-Tree node class to use. Your task is to implement the following methods in the B-Tree node class according to the given specification.

Task 1: Insert Key [33]

```
public BTreeNode<T> insert(T key)
```

This function should insert the key *key* into the tree at the correct position. If this node is not a leaf node, then the correct child node needs to be determined. The root node of the changed tree should be returned.

Task 2: Traverse Tree [5]

```
public void traverse()
```

This function should traverse all the nodes in a sub tree rooted in this node and print out all the keys in the correct order.

You should use your own helper functions to assist in implementing these methods as per 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 tasks 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 (you need to compress your tar 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 what practical session you attend. Upload your archive to the *2019 Prac5 - Tuesday* slot on the Assignment website. Submit your work before the deadline. **No late submissions will be accepted!**