UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science
Faculty of Engineering, Built Environment & IT
University of Pretoria

COS212 - Data structures and algorithms

Practical 5 Specifications: B-Trees

Release Date: 17-04-2023 at 06:30

Due Date: 21-04-2023 at 23:59

Total Marks: 148

Contents

1 General instructions:

- This assignment should be completed individually, no group effort is allowed.
- Be ready to upload your assignment well before the deadline as no extension will be granted.
- You may **not** import any provided Java library. Importing any library or data structure will result in a mark of zero. You may only make use of 1-dimensional native arrays where applicable. If you require additional data structures, you will have to implement them yourself.
- If your code does not compile, you will be awarded a mark of zero. Only the output of your program will be considered for marks, but your code may be inspected for the presence or absence of certain prescribed features.
- All submissions will be checked for plagiarism.
- Read the entire specification before you start coding.
- You will be afforded three upload opportunities.
- Submissions that result in a compilation failure or a run time error will also receive a mark of zero.

2 Plagiarism

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.

3 Outcomes

The primary goal of this assignment is to implement a simplistic B-Tree and perform traversals and lookups on the tree.

4 Introduction

Complete the task below. Certain classes have been provided for you alongside this specification in the Student files folder. A very basic main has been provided. **Please note this main is not extensive and you will need to expand on it**. Remember to test boundary cases. Submission instructions are given at the end of this document.

5 Task:

Your task for this practical will be to implement the following class diagram as described in later sections.

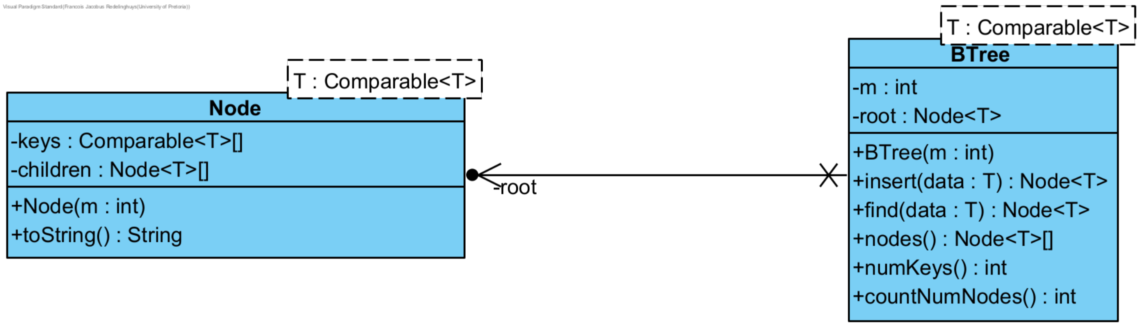

Figure 1: Class diagram

5.1 Node

This class will act as the nodes in the BTree.

You are strongly recommended to add helper functions to this class

- Members:
  - keys: Comparable<T>[]
    - This variable will contain all of the keys in the leaf.
  - children: Node<T>[]
    - This variable will contain all of the children of the leaf.
- Functions:
  - Node(m: int)
    - This is the constructor for the Node class.
    - This constructor should initialize the keys array with a size of m-1 and the children array with a size of m.
  - toString(): String
    - This function is provided.
    - Do not alter this function.
    - Altering this function may negatively impact your final mark.

5.2 BTree

This class will act as the BTree for the practical.

- Members:
  - m: int
    - This is the m value of the tree.
  - root: Node<T>
    - This is the root of the tree.
- Functions:
  - BTree(m: int)
    - This is the constructor for the class.
    - This function should initialize the appropriate member variable.
  - insert(data: T): Node<T>
    - This function will insert the passed-in parameter into the tree.
    - This function is used to determine the appropriate leaf the data will need to be inserted into.
    - If the leaf is full, the function will need to split the leaf as discussed in the lectures.
    - The key that will be moved to the parent during splitting can be determined as follows:

      $$pos = \left\lceil \frac{m-1}{2} \right\rceil$$

    - *Hint: Use the following site to assist you with inserting:* *`https://www.cs.usfca.edu/~galles/visualization/BTree.html`*
    - The function should return the leaf that the passed-in parameter was inserted into.

    - If the passed-in parameter already exists in the tree, then return the leaf that the passed-in parameter can be found in.
    - **IMPORTANT: Ensure that this function is fully working as a failure in this function will negatively impact the rest of your functions**
- find(data: T): Node<T>
    - This function will try and find the leaf that the passed-in parameter can be found in and return it.
    - If the passed-in parameter is not in the tree, the function should return null.
- nodes(): Node<T>[]
    - This function should return an array containing all the nodes in the tree.
    - The order of the nodes in the array is not important. The testing main will compensate for this.
    - If there are no nodes in the tree, the function should return null.
    - Note the array should not contain any nulls and be exactly sized to the number of nodes in the tree.
    - Note: **actual** number of nodes and not the **theoretical** number of nodes.
- numKeys(): int
    - This function should count the number of keys in the tree.
    - Note: **actual** number of keys and not the **theoretical** number of keys.
    - If the tree is empty, return 0.
- countNumNodes(): int
    - This function should count the number of nodes in the tree.
    - Note: **actual** number of nodes and not the **theoretical** number of nodes.

- If the tree is empty, return 0.

6 Helper functions and variables

You are allowed to add helper functions to any of the classes. You are strongly encouraged to add your own helper functions to the Node class.

7 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. The following java files should at least be in a zip archive named uXXXXXXXX.zip where XXXXXXXX is your student number:

- Node.java
- BTree.java

You may add any other custom classes that you created. Your code should be able to be compiled with the following command:

```
make *.java
```

and run with the following command:

```
java App
```

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!**