Middle East Technical University

Department of Computer Engineering

CENG 351

Data Management and File Structures

Fall 2022

PA2 – B+ Tree Index

Due: 08 January 2023 23:55

Submission: **via ODTUClass**

1 Introduction

Cengify is a rising audio streaming platform founded on 20 November 2022 by a student in Middle East Technical University. Platform is rapidly growing. Now, it already reaches several thousands users per day and you as a platform owner realized that the system does not work as fast as you want because you could not think that the platform would grow so fast. Before it becomes a serious problem, you should consider indexing.

In the assignment, you should store the music records by using both primary and secondary **B+ Trees**. You will implement only certain operations described below. Also, it is provided a nice graphical user interface such that you can check the correctness of your implementation.

In the primary B+ tree, you should implement a clustered B+ tree index on `audioId` field. On the other hand, in the secondary B+ tree you should implement an unclustered B+ tree on `genre` field which is a nonunique attribute.

2 Objectives

There are 4 main operations that you should implement in the assignment:

- **Add Audio**: Adds a new song on the trees. The key attribute of the audio files is `<audioId>`, so you should store the audio files according to the `<audioId>` field in the primary B+ Tree. When storing audio record to the secondary B+ tree, you should consider `<genre>` field of the audio records as an index. Note that several audio record may have same `<genre>` field. Therefore, you should use unclustered index structure in the leaf nodes just like in your lecture notes. In other words, in the leaf nodes, each genre bucket contains a list of songs. Also, when you face with **add** comment, you should update both primary and secondary B+ tree.

- **Search Audio**: Searches for a audio record with the given `<audioId>` in **primary B+ tree**, prints the visited nodes along the path. Output specifications is in the section 4.
- **Print Primary Tree**: Print primary tree in depth first order. Output details are in section 4
- **Print Secondary Tree**: Print secondary tree in depth first order. Output details are in section 4

3 Project Structure

3.1 GUI Classes

The following classes are implemented to provide you a better environment for debugging. It is not expected for you to modify those. They will only be executed if the program calling parameter `<guiOptions>` is set to greater than 0. Note that grading will be done without using GUI.

- CengGUI.java
- GUIInternalPrimaryNode.java
- GUIInternalSecondaryNode.java
- GUILevel.java
- GUIPrimaryLeafNode.java
- GUISecondaryLeafNode.java
- GUITreeNode.java
- WrapLayout.java

3.2 Main Files

The classes and files below are used to parse the input and determine the structure of out databases.

- CengPlaylist.java
- CengSong.java
- PlaylistNode.java
- PlaylistNodePrimaryLeaf.java
- PlaylistNodeSecondaryLeaf.java

- PlaylistNodePrimaryIndex.java
- PlaylistNodeSecondaryIndex.java
- PlaylistNodeType.java
- PlaylistParser.java
- PlaylistTree.java

3.3 Implementation

You can make some changes on only 3 files listed below.

- PlaylistTree.java : This class builds a B+ tree structure for both primary and secondary indexes. You are expected to implement the methods marked as `TODO` in this class file. You can define some methods and attributes if necessary.
- PlaylistNodePrimaryIndex.java : This class builds internal index nodes of primary B+ tree. You can define some methods and attributes if necessary.
- PlaylistNodeSecondaryIndex.java : This class builds internal index nodes of secondary B+ tree. You can define some methods and attributes if necessary.

4 Input/Output

The evaluation of the homework will be done without using the GUI. GUI is used to visualize the trees and call the necessary functions by clicking the buttons. There is an input file provided in order to allow GUI to load some examples. You can see how those are parsed when you open the GUI.

The order **d** of B+ trees will be given as an input. Note that, the order applies to leaf nodes as well, i.e., a leaf node can store N records, where $d \leq N \leq 2d$.

You should continuously listen for the inputs from the command line in an infinite loop until you read the string **quit**. When **quit** is read, just terminate the program. Here are some input/output examples.

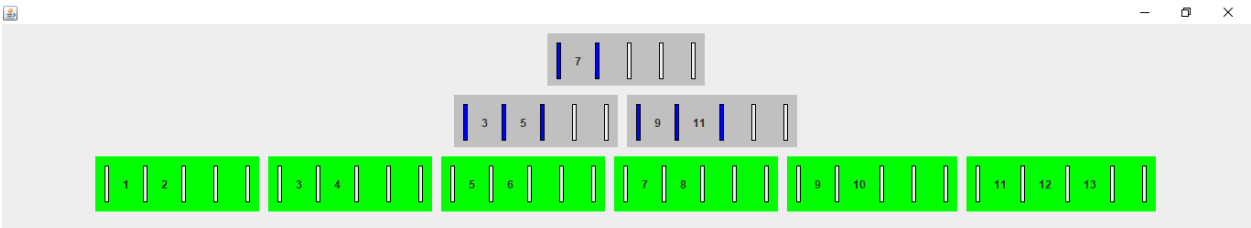
4.1 Add

Input format: `add|<audioId>|<genre>|<songName>|<artist>|`

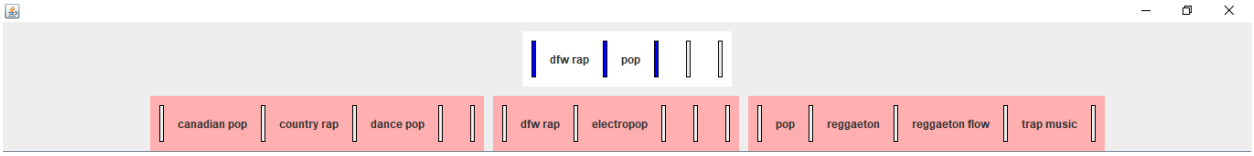
Example Input: `add|10|Turkish Rock|Ceviz Agaci|Cem Karaca`
Output: None

Note the use of the pipe character `|` and the lack of spaces between arguments, in order

to allow spaces within each argument.



Example primary B+ tree after first 13 add operations of example input file.



Example secondary B++ tree after 13 add operations of example input file.

4.2 Search

Input format: `search|<searchKey>`

Example Input: `search|7`
Output: Prints the visited nodes starting from root to data records in leaf nodes, indentation with tabs should be added while traversing down on each level in the tree. If the record is not found, prints "Could not find `<search key>`." All non-leaf and leaf nodes should be printed with a proper indentation (with **tabs**) for each level in the tree.

```
search|7
<index>
7
</index>
    <index>
    9
    11
    </index>
        <data>
        <record>7|trap music|Ransom|Lil Tecca</record>
        </data>
```

Output of search operation

4.3 Print

Should be triggered when **print1** or **print2** are read as input. If input is print1, then you should print primary B+ tree. If input is print2, then you should print secondary B+ tree.

Input format: print1
Output format:
All non-leaf and leaf nodes should be printed with a proper indentation (with **tabs**) for each level in the tree.
For non-leaf nodes, you should print the search key enclosed by `<index>` and `</index>` tags. For leaf nodes, you should print the content between `<data>` and `</data>` tags. Records should be printed between with `<record>` and `</record>` tags.

```
print1
<index>
7
</index>
    <index>
    3
    5
    </index>
        <data>
        <record>1|canadian pop|Señorita|Shawn Mendes</record>
        <record>2|reggaeton flow|China|Anuel AA</record>
        </data>
        <data>
        <record>3|dance pop|boyfriend (with Social House)|Ariana Grande</record>
        <record>4|pop|Beautiful People (feat. Khalid)|Ed Sheeran</record>
        </data>
        <data>
        <record>5|dfw rap|Goodbyes (Feat. Young Thug)|Post Malone</record>
        <record>6|pop|I Don't Care (with Justin Bieber)|Ed Sheeran</record>
        </data>
    <index>
    9
    11
    </index>
        <data>
        <record>7|trap music|Ransom|Lil Tecca</record>
        <record>8|pop|How Do You Sleep?|Sam Smith</record>
        </data>
        <data>
        <record>9|country rap|Old Town Road - Remix|Lil Nas X</record>
        <record>10|electropop|bad guy|Billie Eilish</record>
        </data>
        <data>
        <record>11|reggaeton|Callaita|Bad Bunny</record>
        <record>12|dance pop|Loco Contigo (feat. J. Balvin & Tyga)|DJ Snake</record>
        <record>13|pop|Someone You Loved|Lewis Capaldi</record>
        </data>
```

Output of print1.

```
<index>
dfw rap
pop
</index>
    <data>
    canadian pop
        <record>1|canadian pop|Señorita|Shawn Mendes</record>
    country rap
        <record>9|country rap|Old Town Road - Remix|Lil Nas X</record>
    dance pop
        <record>3|dance pop|boyfriend (with Social House)|Ariana Grande</record>
        <record>12|dance pop|Loco Contigo (feat. J. Balvin & Tyga)|DJ Snake</record>
    </data>
    <data>
    dfw rap
        <record>5|dfw rap|Goodbyes (Feat. Young Thug)|Post Malone</record>
    electropop
        <record>10|electropop|bad guy|Billie Eilish</record>
    </data>
    <data>
    pop
        <record>4|pop|Beautiful People (feat. Khalid)|Ed Sheeran</record>
        <record>6|pop|I Don't Care (with Justin Bieber)|Ed Sheeran</record>
        <record>8|pop|How Do You Sleep?|Sam Smith</record>
        <record>13|pop|Someone You Loved|Lewis Capaldi</record>
    reggaeton
        <record>11|reggaeton|Callaita|Bad Bunny</record>
    reggaeton flow
        <record>2|reggaeton flow|China|Anuel AA</record>
    trap music
        <record>7|trap music|Ransom|Lil Tecca</record>
    </data>
```

Output of print2.

5 Submission

You are going to submit the files given under Section 3.3. Make sure you covered your implementation only under those files. Also, you should not use any subdirectories. Submission will be done via ODTUClass. Archive the files using `tar -cvf <e123456>.tar.gz`.(`<e123456>` part is your student ID.)

6 Regulations

- Programming Language: Java(Version 11).
- Your submission can be extracted by using `tar -xvf <e123456>.tar.gz`
- Grading will be done by using blackbox testing. Therefore, before submitting make sure that your code runs with the following command:

`javac *.java`
`java CengPlaylist <order> <guiOptions> <inputFileName>`
`<guiOptions>`: 0: No GUI, 1: Only primary tree, 2: Only secondary tree, 3: Both primary and secondary tree

- Make sure that output format is exactly matched the output format in Section 4
- We have zero tolerance policy for cheating. People involved in cheating will be punished according to the university regulations.