**CS 564: Project checkpoint 3: Understanding B+-tree index**

**Hand-in Instructions**

1. This homework assignment includes one programming problem in Java.
2. Hand in all parts electronically to your Canvas assignments page.
3. For the programming question, submit a zip file containing all the Java code necessary to run your program, whether you modified the provided code or not.
4. Submit the file with the name (with exactly these names):

|  |
| --- |
| <wiscNetId>.zip |

For example, for someone with UW NetID ggupta7@wisc.edu the file name must be: ggupta7.zip

1. In case of any doubt with the skeleton code given or the questions. Please contact Grishma Gupta ( [ggupta7@wisc.edu](mailto:ggupta7@wisc.edu) ) or TA.

**Problem: Understanding the B+-tree index**

In this checkpoint, each group will implement a B+-tree library and use this library to

(i) create a B+-tree index file for the *Student* table on the *StudentId* field. The leaf node will contain <StudentID, recordID>

(ii) Search for an existing student given a *studentId,* return *recordId* if found. Otherwise, print out a message that the given *studentId* has not been found in the table.

(iii) insert a new student with a new *studentId*. Use a random generator to generate *recordId* for this student. Update both the B+ tree index and the Student table.

(iv) delete an existing student given a *studentId.* Return true if deletion is complete successfully. Return false otherwise.

We assume that all the values being indexed are unique. The structure of the *Student* table is as follow:

StudentId: bigint

StudentName: varchar(255)

Major: varchar(255)

Level: char(2)

Age: int

RecordId:bigint

(v) Print the leaf nodes of B+Tree from left to right using the pointer. Return the list of recordIDs.

RecordId is supposed to represent the address of each row in this table. The data for this table is posted under *student.csv* file.

For the B+ tree insert algorithm, please use the pseudo-code provided in the reference book with split function.

For the B+ tree delete algorithm, please also use the pseudo-code provided in the reference book with the merge function.

Each leaf node in your B+ tree should contain (key, rid). In this application, the key corresponds to StudentId while rid corresponds to RecordId.

Your library should allow users to create a B+tree with degree/order as an input variable.

Due to time constraints, we could repeatedly call insert function while creating a B+ tree from a table.

**Code:**

You must use the code skeleton provided. You are to complete the code by implementing B+Tree() methods insert, search, and delete in the class **BTreeMain** and getStudents() method of the class **BTree**. For your help, already ***“TODO”*** is added. You only need to fill that for a working code.

You are permitted to add or modify the classes and methods, but we require you to keep the IO class as-is for automatic grading.

Always verify the student.csv at the end to confirm if all the entries were added or not.

**Testing**

Test your code on the sample test input file: input.txt and check your results are added onto student.csv and your b+tree. Make sure the results are correct on CSL machines.

**Deliverables**

Put all .java files needed to run your program, including ones you wrote, modified, or were given and are unchanged, into a folder called <wiscNetId>. Compress this folder to create <wiscNetId>.zip and upload it to Canvas. For example, for someone with UW NetID ggupta7@wisc.edu the file name must be: ggupta7.zip

**Rubric: [ For Grading ]**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criteria** | **Unsatisfactory**  **(0-2 points)** | **Fair**  **3 points** | **Good**  **4-5 points** | **Excellent**  **6 points** |
| B+ tree created for the original Student.csv | Code still has compiling errors or running time errors | Code is compiled and run but B+ tree is incorrectly created | B+ created correctly for the most part except 1-2 minor errors/or things needed improvement | B+ file created correctly for a given degree/order of B+ tree |
| Insert | Code still has compiling errors or running time errors | The code is compiled and insert may work without split function. | Successful insert for the most parts. Have some minor errors or things needed improvement | Successfully inserted (tested correctly with split/without split) |
| Search | Code still has compiling errors or running time errors | The code is compiled and run. The search may not run correctly | Search is done and works for the most part. Have some minor errors or things could be improved in some parts | Successfully search |
| Delete | Code still has compiling errors or running time errors | The code is compiled and run. Delete may work for the case without merge but don’t work for the case with merge | Delete is done for the most part. Have some minor things needed to improve | Successfully delete (with/without merge) |
| Readability | The code is poorly organized and very difficult to read | The code is readable only by someone who knows what it is supposed to be doing | The code is fairly easy to read | The code is exceptionally well organized and very easy to follow. Comments are plentiful |