Homework 4

Experimentation with BSTs and B-Trees

Version 1.0

The objective of this assignment is to reinforce concepts we learned about BSTs and B-Trees. The programming assignment should be implemented in Java. You are required to work alone on this assignment. The assignment accounts for 7.5% towards your cumulative course grade.

This assignment may be modified to clarify any questions (and the version number incremented), but the crux of the assignment and the distribution of points will not change. If there are any changes to the assignment, all changes will be documented in the "Change History" section of this assignment.

Due Date: Wednesday, May 1st, 2024 @ 5:00 pm

1 Description of Task

As part of this assignment, you will be responsible for designing programs that implement two different data structures common in computing. The programming assignment should be developed in Java, and your classes must reside in the package **cs250.hw4**.

You will be designing two separate classes that implement the same functionality of storing data in memory using two different data structures.

1.1 Testing your code

We will provide an input file containing X number of random integers separated by new lines. You should read in this file inserting each line as an entry into the tree.

1.2 Task 1: Build a Binary Search Tree (5 pts)

You should create a class called: **BinaryTree** that implements the **TreeStructure** interface that is provided below. Each of the methods part of the **TreeStructure** interface should be implemented using the same logic discussed in the lecture slides.

Command line execution: `java cs250.hw4.BinaryTree <filename>`

Points Breakdown

- **1 point** for correctly implementing the "**insert**" operation.
- **1 point** for correctly implementing the "**remove**" operation.
- **1 point** for correctly implementing the "**get**" operation.
- **1 point** for correctly implementing the "**findMaxDepth**" operation.
- **1 point** for correctly implementing the "**findMinDepth**" operation.

1.3 Task 2: Build a B Tree (2.5 pts)

You should create a class called: **BTree** that implements the **TreeStructure** interface that is provided below. Each of the methods part of the **TreeStructure** interface should be implemented using the same logic discussed in the lecture slides.

Command line execution: `java cs250.hw4.BTree <filename>`

Points Breakdown

0.5 points for correctly implementing the "**insert**" operation.
0.5 points for correctly implementing the "**remove**" operation.
0.5 points for correctly implementing the **"get"** operation.
0.5 points for correctly implementing the "**findMaxDepth**" operation.
0.5 points for correctly implementing the "**findMinDepth**" operation.

1.4 TreeStructure interface explanation

Each of your classes should implement the **TreeStructure** interface we have provided below.

Recall in Java you implement an interface by specifying so in the class declaration.

Example:

```
public class ExampleTree implements TreeStructure {}
```

There are 5 methods in the **TreeStructure** interface which will interact with your underlying data structure.

1. **Insert(num)**: When called this method should insert a node into the tree with the value **num** into the tree along with the current time from **System.nanoTime()**
2. **Remove(num)**: When called this method should find and remove the node in the tree with the value **num**. If **num** did not exist in the tree this method should return **False** and if **num** did exist in the tree this method should return **True**.
3. **Get(num)**: When called this method should find the node with the value **num** then return the time that the node was inserted at.
4. **FindMaxDepth()**: When called this method should return the maximum depth of the current tree.
5. **FindMinDepth()**: When called this method should return the minimum depth of the current tree.

Code:

```
package cs250.hw4;

public interface TreeStructure {

    public void insert(Integer num);

    public Boolean remove(Integer num);

    public Long get(Integer num);

    public Integer findMaxDepth();

    public Integer findMinDepth();

}
```

1.5 Example output

Binary Tree Driver Code

```
public static void main(String[] args) throws FileNotFoundException, IOException {
        File file = new File(args[0]);
        FileReader fReader = new FileReader(file);
        BufferedReader bufferedReader = new BufferedReader(fReader);
        TreeStructure tree = new BinaryTree();
        Random rng = new Random(42);
        ArrayList<Integer> nodesToRemove = new ArrayList<>();
        ArrayList<Integer> nodesToGet = new ArrayList<>();
        String line = bufferedReader.readLine();
        while (line != null) {
            Integer lineInt = Integer.parseInt(line);
            tree.insert(lineInt);
            Integer rand = rng.nextInt(10);
            if (rand < 5) nodesToRemove.add(lineInt);
            else if (rand >= 5) nodesToGet.add(lineInt);
            line = bufferedReader.readLine();
        }
        bufferedReader.close();
        for (int i = 0; i < 10; i++) {
            System.out.println(nodesToGet.get(i) + " inserted at " + tree.get(nodesToGet.get(i)));
        }
        System.out.println("Max depth: " + tree.findMaxDepth());
        System.out.println("Min depth: " + tree.findMinDepth());
        for (Integer num : nodesToRemove) {
            tree.remove(num);
        }
        for (int i = 0; i < 10; i++) {
            System.out.println(nodesToGet.get(i) + " inserted at " + tree.get(nodesToGet.get(i)));
        }
        System.out.println("Max depth: " + tree.findMaxDepth());
        System.out.println("Min depth: " + tree.findMinDepth());
    }
```

Binary Tree Output

```
-1643699032 inserted at 13017969196679
1614093892 inserted at 13017969230068
-1000272748 inserted at 13017969239743
-1907667521 inserted at 13017969249045
1912735862 inserted at 13017969258249
-1142304848 inserted at 13017969303508
1564342588 inserted at 13017969321896
624763728 inserted at 13017969341361
-200006731 inserted at 13017969372713
-284011743 inserted at 13017969401098
Max depth: 33
Min depth: 5
-1643699032 inserted at 13017969196679
1614093892 inserted at 13017969230068
-1000272748 inserted at 13017969239743
-1907667521 inserted at 13017969249045
```

```
1912735862 inserted at 13017969258249
-1142304848 inserted at 13017969303508
1564342588 inserted at 13017969321896
624763728 inserted at 13017969341361
-200006731 inserted at 13017969372713
-284011743 inserted at 13017969401098
Max depth: 28
Min depth: 6
```

B Tree Driver Code

```
public static void main(String[] args) throws FileNotFoundException, IOException {
        File file = new File(args[0]);
        FileReader fReader = new FileReader(file);
        BufferedReader bufferedReader = new BufferedReader(fReader);
        TreeStructure tree = new BTree();
        Random rng = new Random(42);
        ArrayList<Integer> nodesToRemove = new ArrayList<>();
        ArrayList<Integer> nodesToGet = new ArrayList<>();
        String line = bufferedReader.readLine();
        while (line != null) {
            Integer lineInt = Integer.parseInt(line);
            tree.insert(lineInt);
            Integer rand = rng.nextInt(10);
            if (rand < 5) nodesToRemove.add(lineInt);
            else if (rand >= 5) nodesToGet.add(lineInt);
            line = bufferedReader.readLine();
        }
        bufferedReader.close();
        for (int i = 0; i < 10; i++) {
            System.out.println(nodesToGet.get(i) + " inserted at " + tree.get(nodesToGet.get(i)));
        }
        System.out.println("Max depth: " + tree.findMaxDepth());
        System.out.println("Min depth: " + tree.findMinDepth());
        for (Integer num : nodesToRemove) {
            tree.remove(num);
        }
        for (int i = 0; i < 10; i++) {
            System.out.println(nodesToGet.get(i) + " inserted at " + tree.get(nodesToGet.get(i)));
        }
        System.out.println("Max depth: " + tree.findMaxDepth());
        System.out.println("Min depth: " + tree.findMinDepth());
    }
```

B Tree Output

```
-1643699032 inserted at 15715236838213
1614093892 inserted at 15715236870816
-1000272748 inserted at 15715236880460
-1907667521 inserted at 15715236889689
1912735862 inserted at 15715236906432
-1142304848 inserted at 15715236948494
1564342588 inserted at 15715236967429
624763728 inserted at 15715236988701
-200006731 inserted at 15715237022938
```

```
-284011743 inserted at 15715237052359
Max depth: 2
Min depth: 2
-1643699032 inserted at 15715236838213
1614093892 inserted at 15715236870816
-1000272748 inserted at 15715236880460
-1907667521 inserted at 15715236889689
1912735862 inserted at 15715236906432
-1142304848 inserted at 15715236948494
1564342588 inserted at 15715236967429
624763728 inserted at 15715236988701
-200006731 inserted at 15715237022938
-284011743 inserted at 15715237052359
Max depth: 2
Min depth: 2
```

2 What to Submit

Use the CS250 *Canvas* to submit a single .zip file that contains:

- Your Java source codes, matching this directory structure:
    - cs250
        - hw4
- A README.txt file containing a description of each file and any information you feel the TAs need to grade your program.

Filename Convention: The archive file should be named as <FirstName>-<LastName>-HW4.zip . E.g., if you are Cameron Doe then the zip file should be named Cameron-Doe-HW4.zip.

3 Grading

The assignments must compile and function correctly on machines in the CSB-120 Lab. Assignments that work on your laptop on your particular flavor of Linux, but not on the Lab machines are considered unacceptable.

This assignment will contribute a maximum of 7.5 points towards your final grade. The grading will also be done on a 7.5 point scale. There will be multiple submissions each building upon the previous for this assignment to ensure steady progress is made throughout the assignment duration. The points breakdown is as follows:

Points Breakdown

5 points for correctly performing Task 1.
2.5 points for correctly performing Task 2.

Deductions:

2 points each for not naming your classes what we specified.
2 points for any error requiring manual intervention including:
- Incorrect package name
- Incorrect class name
- Not implementing the interface provided
- Etc.

You are required to **work alone** on this assignment.

4 Late Policy

Please check the class policy on submitting late assignments. You are allowed to submit assignments up to 2 days with a per-day deduction of 7.5%.

5 Version Change History

This section will reflect the change history for the assignment. It will list the version number, the date it was released, and the changes that were made to the preceding version. Changes to the first public release are made to clarify the assignment; the spirit or the crux of the assignment will not change.

Version	Date	Comments
1.0	4/6/2024	First public release of the assignment.