CIT 5940 - Module 6 Extra Credit Programming Assignment
Splay Trees

CIT5940 Staff

Contents

Assignment Overview

In the Splay Tree video, we learned a variant of the Binary Search Tree (BST) which aims to achieve better performance by moving elements to the root. Although it is outside the scope of this course, splay trees have comparable performance to AVL trees and Red Black trees with the advantage that their implementation is easier. This assignment asks you to implement a Splay Tree.

Learning Objectives

- Apply what you have learned about how Splay Trees work.
- Gain deeper understanding on recursion vs iteration.

Advice

We highly encourage you to utilize the course discussion forum to discuss the project with classmates and TAs. Before asking the question, please search in the forum to see if there are any similar questions asked before. Please check *all* suggestions listed there.

For this and all other assignments in this course, we recommend that you work on your local computer using an integrated development environment (IDE) such as Eclipse, IntelliJ, Visual Studio Code, or whatever IDE you used for any prior Java courses that you are familiar with. Although you will need to upload your solution to Codio for grading (see the Submission Section below) and could develop your solution on that platform, we recommend using industry standard tools such as Eclipse or IntelliJ so that you become more used to them and can take advantage of their features.

If you have trouble setting up your IDE or cannot get the starter code to compile, please post a note in the discussion forum and a member of the course staff will try to help you get started.

We expect that you will create your own JUnit tests to validate your solution before submission.

1 Setup

1. Begin by downloading the `CIT594-SplayTrees.zip` archive from Canvas, then extract the contents. The main files you will work in are `SplayTree.java`. This contains the unimplemented methods for the code that you will write in this assignment.
2. Compile this code before making any changes. This will ensure your local environment is configured correctly.
3. Review the instructions before implementation.

2 Requirements

2.1 Structure and Compiling

1. You MUST use a JDK version of 17 or higher.
2. You MAY use a JDK version higher than 17, but you MUST set the Java language level to 17 for compatibility with Codio.
3. You MUST NOT change any method signatures for any of the provided methods that you must implement. This includes the parameter lists, names, and return value types, etc.
4. You MUST leave all classes in the default package.
5. You MUST NOT create additional `.java` files for your solution.
6. You SHOULD create JUnit tests to help validate your code. These tests SHOULD NOT be a part of your solution. We do expect you to test your code thoroughly with JUnit tests before submission.
7. You MUST follow our directives in the starter code comments. This means if we ask you to not change a variable's value, you MUST NOT change that variable's value.
8. You MAY create additional helper functions, as long as they meet the other requirements of the assignment (e.g. doesn't crash with invalid inputs, etc.).
9. You MAY add to the `SplayTree` class or the inner `Node` class if desired. But DO NOT remove existing fields.
10. You MAY add additional classes, but you MUST define them in `SplayTree.java`
11. You MUST fill out the required Academic Integrity signature in the comment block at the top of your submission files.

2.2 Functionality Specifications

2.2.1 General Requirements

1. Your method MUST NOT crash if a user enters an invalid input. Remember you do not have control over what a user inputs as arguments. For example, all methods should handle `null` inputs gracefully.
2. You MUST adhere to these definitions:
   (a) A **parent** `Node` is a `Node` with successors.
   (b) A **child** `Node` is a `Node` that is a successor of a parent `Node`.
   (c) The **root** `Node` is the highest `Node` in the structure and has no parent `Node`.
   (d) We will follow the steps to splay as shown in the video.
   (e) **Find** - If the element is present, we splay the element. If the element is not present, we splay the last element that was accessed.
   (f) **Insert** - If the element is present, we splay the element. If the element is not present, insert the element and splay it.
   (g) **Delete** - We delete in the following manner: If the element to be deleted has 0 or 1 children, deletion is straightforward. If the element has 2 children, we swap the element to be deleted with its in order predecessor. The element to be deleted now has 0 or 1 children, and can be deleted. For splaying, if the element is present, we delete the element and splay the parent. If the element is not present, we splay the last element accessed.
3. We will be running performance tests for this assignment. As we will be testing on large inputs, recursion may end up causing stack overflow. If that occurs, use iteration. Note that there is a difference between ! = and *equals*, which may cause a bug when comparing integers. Refer https://stackoverflow.com/a/3637978

2.2.2 `boolean addNoSplay(E)`

1. this method MUST take a generic value as the argument.
2. it MUST return true or false depending on whether the value is added or not.
3. it MUST not splay.
4. the size of the tree must be updated accordingly.

2.2.3 `boolean add(E)`

1. this method MUST take a generic value as the argument.
2. it MUST return true or false depending on whether the value is added or not.

3. it MUST splay as defined above.
4. the size of the tree must be updated accordingly.
5. null values should NOT be added.

2.2.4 `boolean findNode(E)`

1. this method MUST take a generic value as the argument.
2. it MUST return true if a `Node` with the value exists, and false otherwise.
3. it MUST splay as defined above.

2.2.5 `boolean remove(E)`

1. this method MUST take a generic value as the argument.
2. it MUST return true or false depending on whether the value is deleted or not.
3. it MUST splay as defined above.
4. the size of the tree must be updated accordingly.

2.2.6 `List<E> inOrder()`

1. this method MUST NOT take any arguments
2. it MUST return a list of elements corresponding to inorder traversal.

2.2.7 `List<E> preOrder()`

1. this method MUST NOT take any arguments
2. it MUST return a list of elements corresponding to preorder traversal.

2.2.8 `List<E> postOrder()`

1. this method MUST NOT take any arguments
2. it MUST return a list of elements corresponding to postorder traversal.

3 Submission

3.1 Pre-submission Check

Before you submit, please double check that you followed all the instructions, especially the items in the Structure and Compiling section above.

3.2 Codio Submission

1. **This will be a one-time submission only!** Make sure you test your code thoroughly before submitting.
2. Upload your solution and any JUnit test files to the "submit" folder.
3. In the menu bar, select Run JUnit Tests. This will run any unit test files that you have uploaded with your submission. Note that there are no pre-submission checks provided. However you can use this feature as a basic validation check to be sure your code compiles.
4. When you're ready to submit, go to Education and select Mark As Completed. Confirm at the prompt.
5. You will see quite a bit of output, even if all the tests pass. At the bottom of the output, starting at `YOUR AUTOGRADING RESULTS BELOW` you will see the number of successful and failed test cases.

4 Grading

4.1 Grading Overview

1. There is no peer review for this assignment. You will not be marked on code style. However, we do expect you to use best practices (comments, indentation, etc), especially if you want to review with a TA.
2. There are no hidden tests. You will be able to see the name of each test, if it passed or failed, and the associated point value.
3. For this assignment, failed tests will have minimal feedback.
4. We will not provide the exact test cases.
5. The score you see is final. Scores will sync to Canvas relatively immediately.

4.2 Rubric Items

1. SplayTreeBasicTest: 10 points
2. SplayTreeTest: 30 points
3. SplayTreePerformanceTest: 4 points
4. Total points: 44

Good Luck!