CSE 522
Assignment 1:
Code Reuse in BSTs
Assigned (Part 1): Sept 6, 2018
Assigned (Part 2): Sept 7, 2018

Due: Sept 17, 2018 (11:59 pm)

You may work on this assignment as a team of at most two students.

In the class lectures, we discussed two object-oriented definitions of the binary search tree (BST), called class `Tree` and class `AbsTree` respectively. In this assignment, you are to complete the definition of a `delete` operation for both versions of the binary search tree. Part1 pertains to defining `delete` for `Tree` and Part2 pertains to defining `delete` for `AbsTree`.

Part 1: Define delete in class Tree

See file A1_Part1.pdf.

Part 2: Define delete in class AbsTree

Preliminaries: Refer to `BST_Part2.java` posted at `Resources → Assignments`. It provides the starting point for Part 2.

A protected field `AbsTree parent` field has been added to class `AbsTree`. Revise the `insert` method so that the `parent` field is correctly set when a value is inserted into the tree. Define three procedures, `min()`, `max()`, and `find(n)` which return, respectively, the `AbsTree` node with the minimum value, the `AbsTree` node with the maximum value, and the `AbsTree` node containing the value `n`. If `n` is not present in the tree, `find(n)` should return `null`.

Defining delete: Define the `delete` method so that it works for ordinary trees as well as duptrees. Similar to `insert`, the code for `delete` should be kept in class `AbsTree` and should capture what is common to trees and duptrees. The differences in `delete`’s behavior between `Tree` and `DupTree` should be expressed in terms of one or more *protected abstract methods* which are implemented in the classes `Tree` and `DupTree`.

The four cases of delete described in Part 1 also apply here. Additionally, when a value `n` in a duptree has a `count > 1`, the method `delete(n)` should decrement the `count` field associated with `n` but should not delete the node. If a value `n` is associated with a `count == 1`, the method `delete(n)` should remove the node containing value `n` from the duptree.

Watch the posted screen-cast, A1_Part2.mp4, for a clarification of delete’s behavior.

Run your program under JIVE and save the object and sequence diagrams in files called `obj2.png` and `seq2.png`, respectively, at the point when all `insert` and `delete` operations have been performed by `main`, but `main` has not yet exited. Choose the “Objects with Tables” option for the Object Diagram.

Note**:** Print out error messages on the Console when the value to be deleted is either not present in the tree or is present at the root of the tree with count == 1 and both subtrees are null.

What to Submit**:** Prepare a top-level directory named *A1_Part2_UBITId1_UBITId2* if the assignment is done by two students (list the *UBITIds* in alphabetic order); otherwise, name it as *A1_Part2_UBITId* if the assignment is done solo.

In this directory, place `BST_Part2.java, obj2.png` and `seq2.png`. Compress the directory and submit the compressed file using the Linux command `submit_cse522`. Details regarding online submission will be posted in due course at

```
Resources → Assignments → Online_Submission_CSE522.pdf
```

Only one submission per team is required.

Important Note for Parts 1 and 2:

Do not change the names of classes, fields, or methods given in the Assignment.

Do not change the names or the number of parameters for the methods.

You are free to have any number of local variables in the methods that you are asked to define: min, max, find, case1, case2, case3L, and case3R.

End of Assignment 1