CSE 522
Assignment 1:
Code Reuse in BSTs
Assigned: Sept 6, 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

***Preliminaries*:** Refer to `BST_Part1.java` posted at `Resources → Assignments`. It provides the starting point for Part 1.

A protected field `Tree parent` field has been added to class `Tree`. 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 `Tree` node with the minimum value, the `Tree` node with the maximum value, and the `Tree` node containing the value `n`. If `n` is not present in the tree, `find(n)` should return `null`.

***Defining delete*:** The `delete(n)` method should remove value `n` from the tree if it is present in the tree while ensuring that the remaining values maintain the binary search tree property. A good explanation of `delete` is given at:

http://www.algolist.net/Data_structures/Binary_search_tree/Removal

There are four main cases to `delete` depending upon whether the value to be deleted is at:

(i) A leaf node (but not the root); or
(ii) A non-leaf node (not the root) with only one non-null subtree; or
(iii) A root node with one non-null subtree; or
(iv) A node with both non-null subtrees.

Note that we do not allow the root node to be deleted if it has both null subtrees.

The top-level definition of the `delete(n)` method in class `Tree` has been provided to you – do not modify this definition. The method makes use of four helper methods, called `case1`, `case2`, `case3L`, and `case3R`. The methods `case1` and `case2` correspond to cases (i) and (ii) above while case (iii) is implemented using two methods, called `case3L` and `case3R`, depending upon whether the missing tree is on the left (`case3R`) or the right (`case3L`). Case (iv) can be handled using `case3L` or `case3R`.

The file BST_Part1.java provides the outline of class Tree and the definition of the driver class, BST_Part1, which contains method main. Your task is to provide definitions of the four helper methods for delete in addition to the methods min, max, and find.

A screen-cast BST_Part1.mp4 will be posted clarifying the cases of delete.

Run your program under JIVE and save the object and sequence diagrams in files called obj.png and seq.png, respectively, at the point when all insert and delete operations have been performed by main, but main has not yet exited. For the object diagram, choose the 'Stacked with Tables' option.

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 both subtrees null.

What to Submit**:** Prepare a top-level directory named *A1_Part1_UBITId1_UBITId2* if the assignment is done by two students (list the *UBITIds* in alphabetic order); otherwise, name it as *A1_Part1_UBITId* if the assignment is done solo.

In this directory, place BST_Part1.java, obj.png and seq.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.

End of Part 1

Part 2: Define delete in class AbsTree

Will be posted soon.