Jaeseung Lee

CS241

Project 1: BST Implementation

Due 04/24/2016 11:59PM

Section1 : ADT description

In the Project1 class, where the main method is, I used scanner to recieve the String commands from user, and then convert those into appropriate data type. Some of the examples for possible cases of inputs are: First, "32 22 44 55 13.."for initial sequence of values with unknown length. Second, "I 2", D 22, I 112..", which has 3+ length. Third, "H" with 1 length. First case wasn't repetitive case where you do it only once, so I didn't put it in the while loop. For others, I did put it in while loop, to give a permission to quit to user. Appropriate comments are there to explain for each codes in main method.

In the BST class, first of all, I do have nested BSTNode class at the end, to have better efficiency rather than having one more extra class file. I have int type variable "data", and then BSTNode type "left", "right", and "parent". I didn't make "parent" variable until I make deleteNode() method, where I actually have to keep track the parent node class of the node that I want to delete. For findNode(), it is useful method that I have used to find the location of the wanted-node, and then return the found node. In the insertNode(), I used recursion to find the appropriate spot to insert the desired node, otherwise, it will overwrite the existing nodes. In the deleteNode(), it was more complicated then insertNode(), where we have 3 cases: the node has two children(left,right), the node has one child(It is either its parent's left-child or right-child), and no-child(it is a leaf node). I used recursion to find the successor of the node and node that we want to delete. For the 3 cases above, I had to use local variables, to temporarily store the BSTNode, in order to swap the nodes and nullify the links or link to other nodes. And then I have printTraverse(), which just prints out the result of in-order, pre-order, and post-order traversals to the screen. Next, I have getPredecesssor(), getSuccessor(), to find the desired node of predecessor and successor. Return type is data type, since we want to actually print-out the data to the screens.

Section2: Testing Methodology

```
Please enter the initial sequence of values:
51 29 68 90 36 40 22 59 44 99 77 60 27 83 15 75 3
Pre-order: 3 15 22 27 29 36 40 44 51 59 60 68 75 77 83 90 99
In-order: 51 3 15 22 27 29 36 40 44 59 60 68 75 77 83 90 99
Post-order: 51 3 15 22 27 29 36 40 44 59 60 68 75 77 83 90 99
Command? H
I  Insert a value
D  Delete a value
P  Find predecessor
S  Find successor
E  Exit the program
H  Display this message
Command? I 88
In-order: 3 15 22 27 29 36 40 44 51 59 60 68 75 77 83 88 90 99
Command? I 42
In-order: 3 15 22 27 29 36 40 42 44 51 59 60 68 75 77 83 88 90 99
Command? D 44
99 doesn't exist!
NullPointerExcpetion occured.
NullPointerExcpetion occured.
NullPointerExcpetion occured.
In-order: 3 15 22 27 29 36 40 42 44 51 59 60 68 75 77 83 88 90 99
Command? D 90
3 doesn't exist!
NullPointerExcpetion occured.
NullPointerExcpetion occured.
NullPointerExcpetion occured.
NullPointerExcpetion occured.
In-order: 3 15 22 27 29 36 40 42 44 51 59 60 68 75 77 83 88 90 99
Command? D 70
3 doesn't exist!
NullPointerExcpetion occured.
NullPointerExcpetion occured.
NullPointerExcpetion occured.
NullPointerExcpetion occured.
In-order: 3 15 22 27 29 36 40 42 44 51 59 60 68 75 77 83 88 90 99
Command? S 75
59
Command? P 99
44
Command? E
Thank you for using my program!
```

I was having trouble on deleteNode(), where only way I can think of implementing this was, adding one more BSTNode variable, "parent". I have tried to store the parent's nodes when I use the recursion to insert. I thought while I inserting value, it will be perfect to store each nodes' reference to their parent. But somehow, when I tried to use each nodes' parent references, they were shown as null. If each nodes' parent references were properly stored, I wouldn't have any errors for using deleteNode method, where I thought about all the possible cases: when its leaf node, when it has a child, when it have two children. Other than that, the function of insert works fine, used recursion calls to find the appropriate spot to insert. Printing-out the value for the node's predessor and successor works fine, I have tried to make interface clean and organized as possible, and tried to code possible code into the seperate method, in that way, I can use it when I implement other methods, such as findNode(). Or sometimes, it looks cleaner to make extra method where you can group relevant codes into one method, such as printTraverse(). Pre-order, In-order and Post-order also looks fine as well. Alternatively for the interface in Project1 class, I also thought using StringTokenizer. Although I finally decided to use String array with split(), it was good to review about it, and I think it is always good to know alternative way to code.

Section3: Lessons learned

It was the first time where my expected style of code didn't work, to be specific, deleteNode method. I am dying to know why it was impossible to store each node's parent while I perform recursion to traverse in insertNode(). And if it is possible to perform deleteNode without adding parent node variable, I want to know. Although I heard it is really inefficient to use recursion in real life coding, I think it is really important to understand deeply-understanding how the recursion works.