Name, SID, Date .....

In Class Assignment 21: Binary Tree Data Structure

Benjamin Sanders, MS November 25, 2020

1 Introduction

You will need to work individually to complete this assignment. Write your name at the top of all pages for this assignment. Turn in all work to Blackboard on or before the deadline to receive credit.

You may use additional libraries and online resources, if you get them approved in writing, over email, from the instructor first. If you have received approval from the instructor, write the approved libraries and any references in the space below.

.....
.....
.....
.....

2 Assignment Description

2.1 Big Picture

This data structure stays sorted. Similar, more complex versions of this data structure are what an Operating System uses to implement a filesystem on a hard drive.

2.2 Algorithm Implementation

Implement the following algorithms in Java, using the Vector data structure for any 1-D array, 2-D array, or linear algebra purposes.

ADD-RECURSIVE(R, v)
1 **if** $R == null$
2 **return** *new* NODE(v)
3 **if** $v < R.v$
4 $R.left =$ ADD-RECURSIVE($R.left, v$)
5 **elseif** $v > R.v$
6 $R.right =$ ADD-RECURSIVE($R.right, v$)

DELETE-RECURSIVE(R, v)
1 **if** $R == null$
2 **return** $null$
3 **if** $v == R.v$
4 **if** $R.left == null$ and $R.right == null$
5 **return** $null$
6 **if** $R.right == null$
7 **return** $R.left$
8 **if** $R.left == null$
9 **return** $R.right$
10 $s =$ SMALLEST($R.right$)
11 $R.v = s$
12 $R.right =$ DELETE-RECURSIVE($R.right, s$)
13 **return** R
14 **if** $v < R.v$
15 $R.left =$ DELETE-RECURSIVE($R.left, v$)
16 **return** R
17 $R.right =$ DELETE-RECURSIVE($R.right, v$)
18 **return** R

Name, SID, Date .....

```
SMALLEST(R)
1  if R.left == null
2      return R.v
3  else
4      return SMALLEST(R.left)
```

```
TRAVERSE-LEVEL-ORDER(R)
1  if R == null
2      return
3  A[0] = R
4  while A ≠ empty
5      N = A[0]
6      A.DELETE(A[0])
7      PRINT(N.v)
8      if N.left ≠ null
9          A.ADD-AT-END(N.left)
10     if N.right ≠ null
11         A.ADD-AT-END(N.right)
```

These algorithms assume the existence of a $Node$ class, being composed of integer v, Node $left$ and Node $right$.

2.3 Time Complexity Analysis

Where n is the number of data points to be stored, analyze the time complexity of the given algorithms with respect to n. Write the result of your analysis in big-O notation, i.e. $O(n^2)$ in the space below.

.....

2.4 Space Complexity Analysis

Where n is the number of data points to be stored, analyze the space complexity of the given algorithms with respect to n. Write the result of your analysis in big-O notation, i.e. $O(n \cdot log(n))$ in the space below.

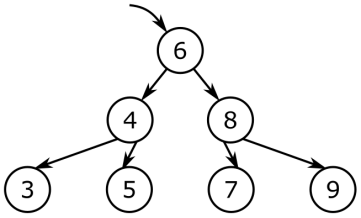
.....

Name, SID, Date .....

2.5 New Algorithm Design and Implementation

In the space below, design an algorithm that prints the tree to the console, in level-order, per level. Given inputs `6, 4, 8, 3, 5, 7, 9`, in that order, the output should look like the below screenshot at left, which represents a collapsed-left version of the tree drawing, at right. Use pseudocode written in a style similar to the given algorithm, and implement it in Java. You may use as many additional pages as necessary for this purpose.

```
6,
4,  8,
3,  5,  7,  9,
.,  .,  .,  .,  .,  .,  .,  .,
```



3 What to Turn In

Turn in one PDF or Word document on Blackboard, containing the following items.

1. All pages scanned or photographed of the In Class Assignment completed document.
2. Any additional pages you used to complete the assignment.
3. All code created for the assignment, along with test cases.
4. One statement indicating which parts of your implementation(s) are working, and which parts are not.
5. Screenshots demonstrating the code working, if it is working.