CS 260 - Tree Worksheet

Professor Mark W. Boady

Assignment 9

1 Introduction

Content by Mark Boady, Drexel University
This worksheet is worth 100 points.
You may complete this worksheet using any of the following methods.

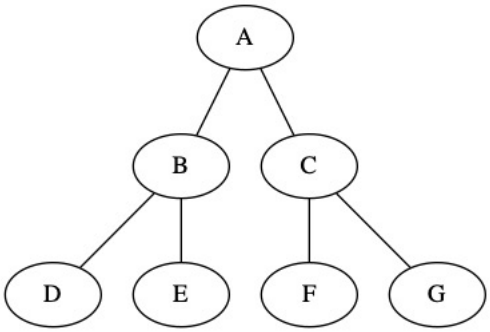
- Print out the PDF. Complete by hand. Scan the file.
- Write directly on the PDF with editing software
- Write the Answer in MS Word, Notepad, etc. You **do not** have to rewrite the questions. Just clearly label each answer in your file.

2 Tree Worksheet

Question 1 : 10 points

A tree is a special time of Data Structure.

A visual representation of a tree is shown below.



(a) (1 point) A **node** is a value stored in the tree. A node is drawn as a circle. How many nodes does this tree have?

7

(b) (2 points) An **edge** is a relationship between two values. It is drawn as a line connecting the nodes. For example, there is an edge (A,B). In a tree order matters, always write the higher node first.

What are all the edges in this tree?

(A,B), (A,C), (B,D), (B,E), (C,F), (C,G)

(c) (1 point) A node's **parent** is the node directly above it in the graph. For example, B is the parent of D.

What node is the parent of C?

A

(d) (2 points) A node's **child** is the node directly below it in the graph. For example D is the child of B.

What two nodes are children of C?

F, G

(e) (1 point) The **Root** of the tree is a node with no parent. What is the root node of this tree?

A

(f) (2 points) The **Leaves** of the tree are any nodes with no children. What are the leaves of this tree?

D, E, F, G

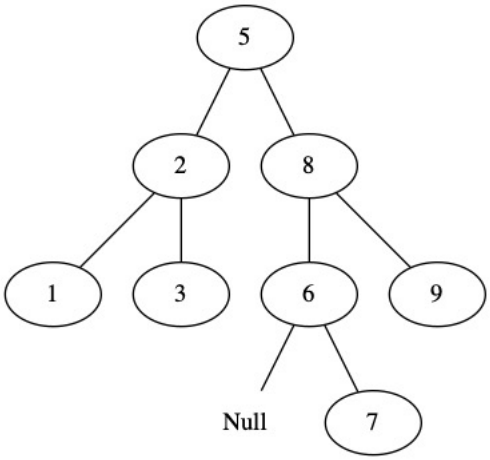
(g) (1 point) The **Height** of a tree is the maximum number of edges (parent to child) needed to get from the root to a leaf. What is the height of the tree?

2

Question 2 : 6 points

A Binary Search Tree is a special kind of tree. It has two special properties.

- Each Node has **at most** two children
- The tree is ordered
  - Left Children **must** be smaller then the parent
  - Right Children **must** be greater then the parent



A Binary Search Tree is a Data Structure used for quick searching.

```
function TreeSearch(target,node)
    if node == Null then
        return False
    end if
    if node.val == target then
        return True
    end if
    if node.val < target then
        return TreeSearch(target, node.right)
    else
        return TreeSearch(target, node.left)
    end if
end function
```

If we wanted to find 3. We would compare $3 < 5$, $3 > 2$, $3 == 3$. It takes us three comparisons to find out that 3 is in the tree. If we run out of nodes, we know the value is not in the tree.

(a) (2 points) If we want to find 1, what numerical comparisons would we make? (Not counting NULLs)

1<5, 1<2, 1==1

(b) (2 points) If we want to find 7, what numerical comparisons would we make? (Not counting NULLs)

7>5, 7< 8, 7> 6, 7==7

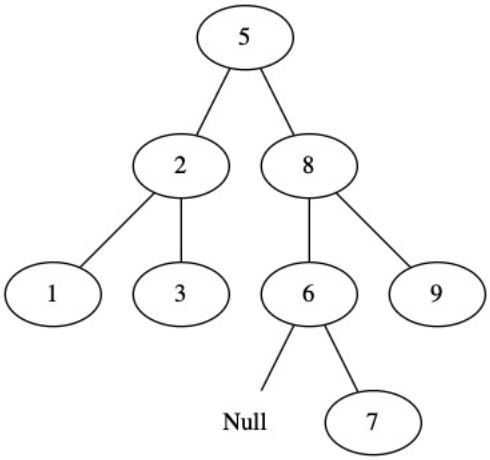
(c) (2 points) If we want to find 12, what numerical comparisons would we make? (Not counting NULLs)

12>5, 12>8, 12>9, end(no 12)

Question 3 : 8 points

The biggest advantage of a Binary Search Tree is how fast we can find items.

Review the below tree again.



(a) (1 point) What is the **height** of the tree? (Remember height is the longest path.)

3

(b) (1 point) What value in the tree requires the least comparisons?

5

(c) (1 point) How many comparisons are needed for the best case?

1

(d) (1 point) What value in the tree requires the most comparisons?

7

(e) (1 point) How many comparisons are needed for the worst case?

4

(f) (1 point) What is the maximum number of comparisons needed to determine a value is not in the tree? (Assume the tree can have decimal numbers, like 7.9)

4

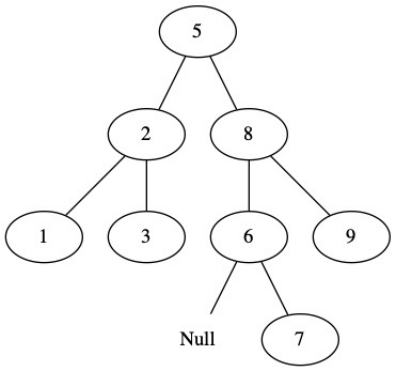
(g) (2 points) Assume the height of a tree is called h. Give a relationship between the height and worst case number of comparisons $c(h) =?$.

h + 1 **This is if we include a comparison for checking if there is a child to the last node. If not it would just be h**

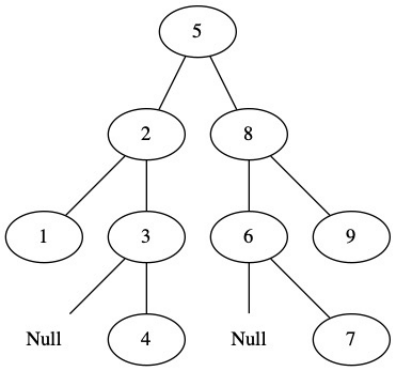
Question 4 : 4 points

A Binary Search Tree contains no duplicates. If we want to insert a new value into the tree, the algorithm is very similar to search.

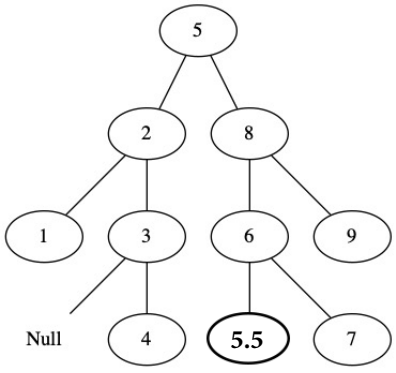
To insert an element follow the search pattern. If you find the node, then no changes are made. If the node is not found, insert the value as a child of the leaf you compare to.



We want to insert 4. We compare $4 < 5$, $4 > 2$,$4 > 3$. In a search, we would return false because there is no right child of 3 to compare to. With Insert, we add the node 4 at this location.



(a) (4 points) Insert 5.5 into the above Binary Search Tree. Draw the new tree below.



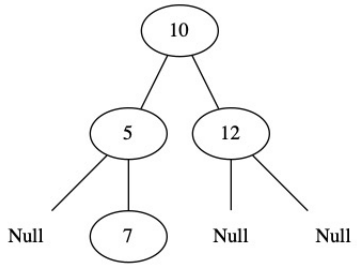
Question 5 : 24 points

Displaying Trees in text format can be a challenge. There are tree methods for traversing the nodes of a tree to display as a single string.

Since Null spaces are import to BSTs, we will write them as N.

- **Preorder**: Print a Node, Print Left Children, Print Right Children.
- **Inorder**: Print Left Children, Print a Node, Print Right Children.
- **Postorder**: Print Left Children, Print Right Children, Print a Node.

Examples will be shown using the below tree. **Note**: Obvious nulls are not shown in the image but still displayed.

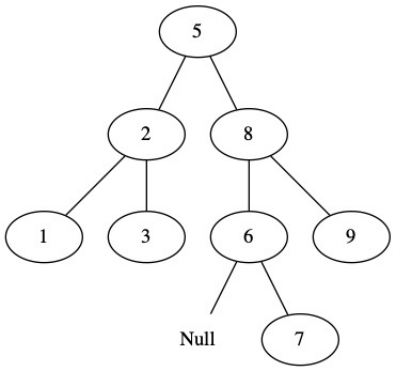


Preorder: 10 5 N 7 N N 12 N N

Inorder: N 5 N 7 N 10 N 12 N

Postorder: N N N 7 5 N N 12 10

Complete all three on the tree from the previous questions.



(a) (8 points) What is the preorder traversal of the tree?

5, 2, 1, N, N, 3, N, N, 8, 6, N, 7, N, N, 9, N, N

(b) (8 points) What is the inorder traversal of the tree?

N, 1, N, 3, N, 2, N, 5, N, 8, N, 6, N, 7, N, 9, N

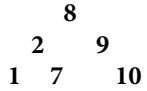
(c) (8 points) What is the postorder traversal of the tree?

N, N, 1, N, N, 3, 2, N,N, N, 7, 6, N, N, 9, 8, 5

Question 6 : 7 points

Create a Binary Search Tree by starting with an empty tree and inserting the following nodes in the order given.

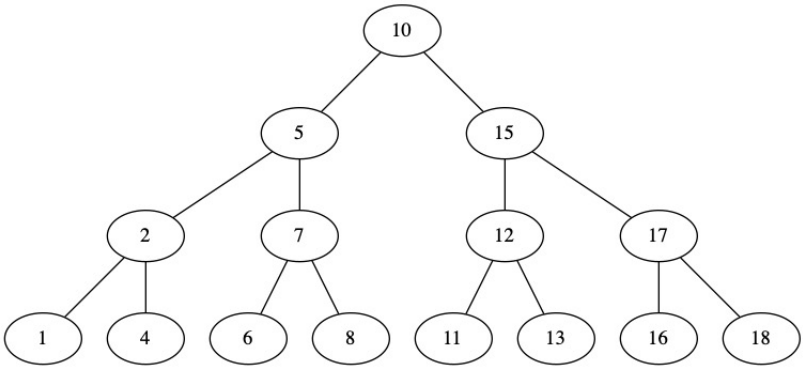
Insert in this order: 8, 2, 9, 1, 10, 5, 7



Question 7 : 16 points

Deleting from a BST is based on finding either the local min or the local max. We want to find the min or max, but only of a subset of nodes.

The global min of this tree is 1. The global max is 18. What if we want the minimum of just descendants of 7? There are only three descendants 7, 6 and 8. The min is 6.



(a) (2 points) What is the **min** of all **descendants** of 5?

1

(b) (2 points) What is the **max** of all **descendants** of 5?

8

(c) (2 points) What is the **min** of all **descendants** of 15?

11

(d) (2 points) What is the **max** of all **descendants** of 15?

18

(e) (2 points) What is the **min** of all **descendants** of 2?

1

(f) (2 points) What is the **max** of all **descendants** of 2?

4

(g) (2 points) What is the **min** of all **descendants** of 17?

16

(h) (2 points) What is the **max** of all **descendants** of 17?

18

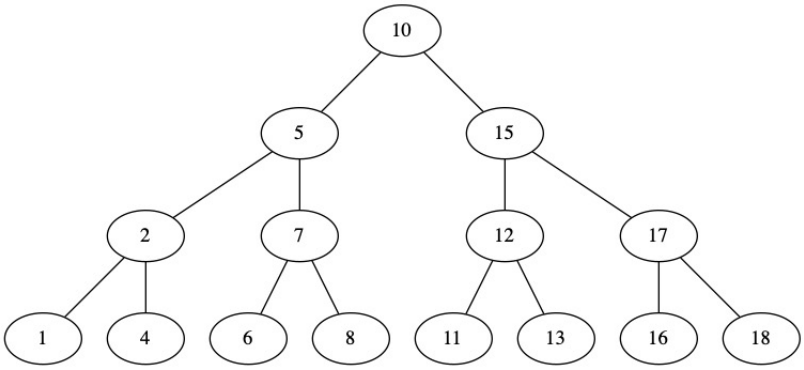
Question 8 : 8 points

Deleting the **root** of a large tree is the hardest thing to remove. We want to avoid rebuilding the whole tree.

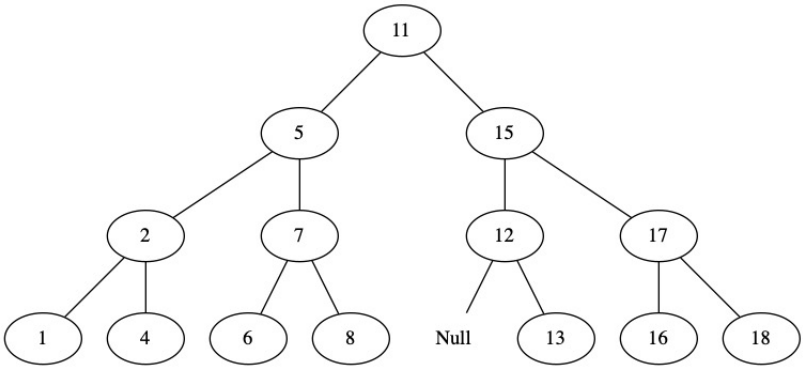
The easiest solution is to **not delete the root**! Instead, we delete the **min** of the **descendants** of the 15. (15 is the root's right child) Then we overwrite the root with this value.

We have solved the delete problem without rebuilding the entire tree.

Given this tree



We delete the leaf node containing 11, then we overwrite the root 10 with that value.



(a) (4 points) Is the resulting tree still a valid Binary Search Tree? Why did 11 fit in the root position?

Yes it is a valid tree, that is because the min decedent of the right side of the tree is the lowest value that is still greater then the head of the left side of the tree meaning it is still in the middle of each child of the previous head.

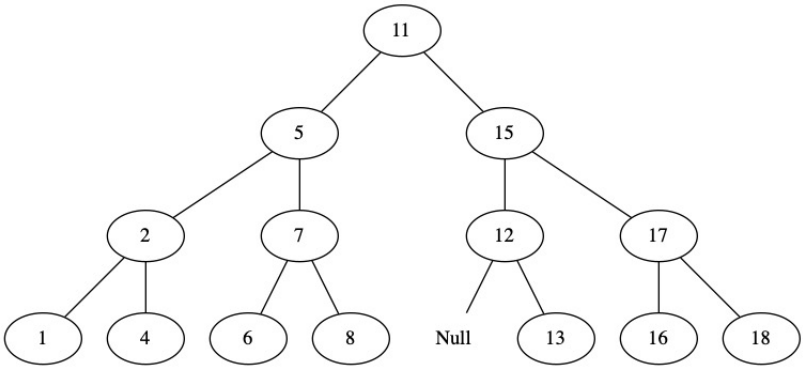
(b) (4 points) Would the **max** of the 5's **descendants** also have worked? Why

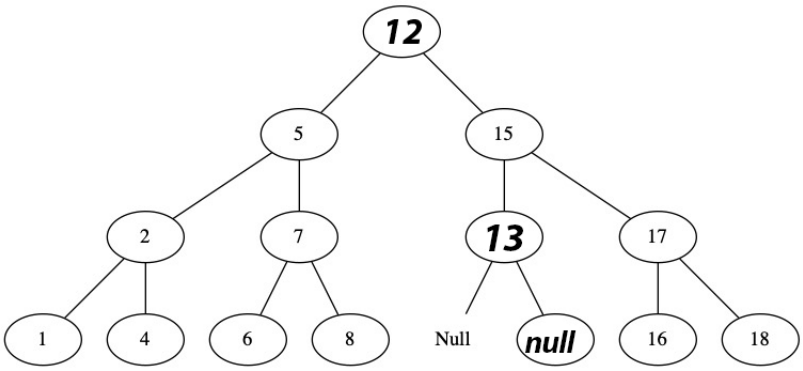
Yes, it would also have worked since it is similar to the min of the right side, just a different number.

Question 9 : 5 points

Delete the root from this tree. Use the **min** of the **right side** as your replacement. Draw the new tree.

Hint: You will have to come up with a way to deal with deleting the value.





Question 10 : 10 points

Draw all possible BST that can be created by inserting 1,2,3 in different orders.

Hint: There are 5.

1.) 1, 2,3

```
1
  2
    3
```

2.) 3, 2, 1

```
  3
2
1
```

3.) 1, 3, 2

```
  1
    3
  2
```

4.) 2, 3, 1

```
  2
1   3
```

5.) 3, 1, 2

```
  3
1
  2
```

Question 11 : 2 points

What is the average height? (Sum of all 5 heights / 5)

9 / 5 = 1.8