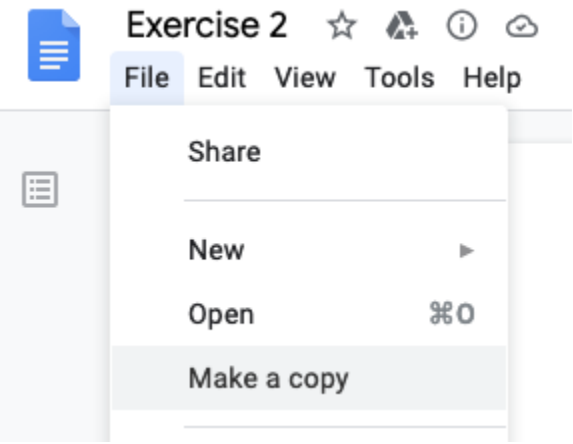
Week 9 Recitation Exercise

CS 261 – 10 points

In this exercise, you'll explore the differences between regular binary search trees and AVL trees. Follow the instructions below, and when you're finished, make sure to submit your completed exercise on Canvas. **Note that, unlike in previous recitations, this exercise will be done individually instead of in groups, so make sure to submit your own copy of the completed exercise on Canvas.** Remember that this exercise will be graded based on effort, not correctness, so don't worry if you don't get all the right answers. Do make sure you *try* to get the right answers though.

Step #1: Create your submission document

Start out by making a personal copy of this document so you're able to edit it:



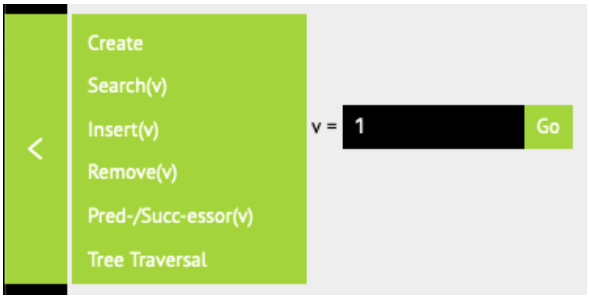
Step #2: Familiarize yourself with the VisuAlgo implementations of BSTs and AVL trees

In this exercise, you'll use the site VisuAlgo.net to explore the operations of BSTs and AVL trees. Take a minute to familiarize yourself with the way the BST/AVL tree implementation there works:

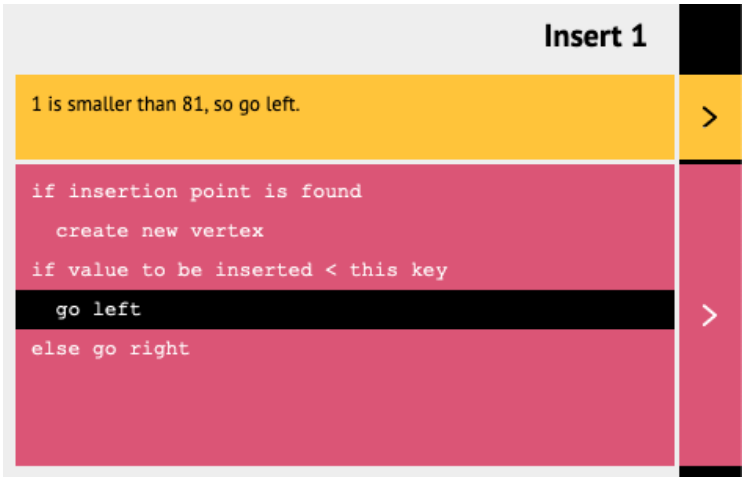
https://visualgo.net/en/bst

There are a couple things to note about this page. First, you can use the box at the lower left of the page to select and execute operations on a BST or AVL tree. For example, to insert a value into the current tree (which you should see visualized on the

main part of the page), you can select the “Insert” operation, enter the key to be inserted and click “Go”:



As the operation runs, you will see the site step through a pseudocode implementation of the operation in the box at the lower right of the page:



Each step of the operation will also be animated in the main visualization of the tree, and at the end, the tree will be modified with the result of the operation.

Finally, note that you can switch between a regular BST implementation and an AVL tree implementation in the navigation bar at the top of the page:

Play around a bit with the operations of both tree implementations so you understand how the site works.

Step #3: Build a BST and an AVL tree from the same data

Now that you're comfortable with the BST and AVL tree implementations on VisuAlgo, you'll build a tree from the same data using each implementation. Begin by creating a new, empty BST. Then, run the following sequence of operations in that BST:

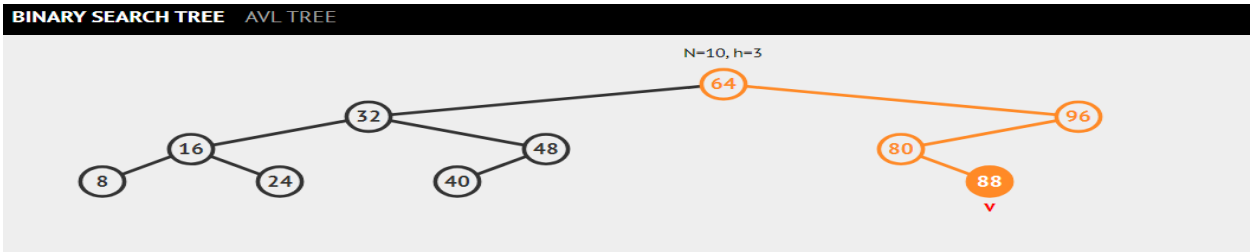
```
insert(64)
insert(96)
insert(32)
insert(16)
insert(80)
insert(24)
insert(48)
insert(8)
insert(40)
insert(88)
```

Make note of how the tree changes as it's being built and what it looks like when all the operations are complete. Take a screenshot of the final tree for later.

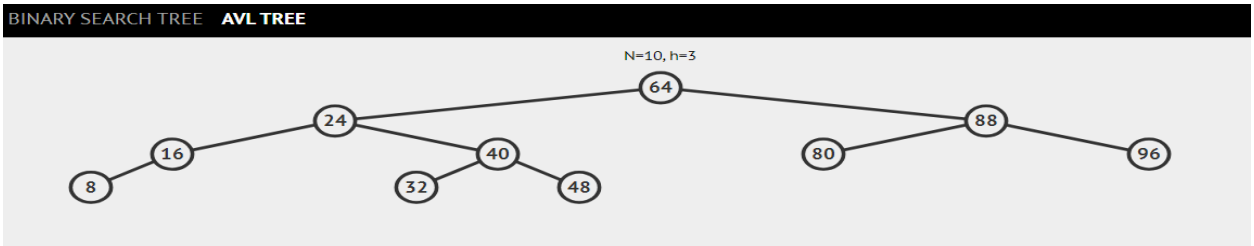
Once you're finished building your BST, create a new, empty AVL tree, and run the same sequence of operations in that tree. Again, make note of how the tree changes as it's being built and what it looks like when all the operations are complete. Pay particular attention to the number of times a rotation must be performed during the sequence of operations. Again, take a screenshot of the final tree.

When you're finished building your BST and your AVL tree, describe below what were the similarities and differences in the way each different kind of tree executed the same sequence of instructions. How did the finished trees differ in the end? How many rotations were performed by the AVL tree implementation? Make sure to include the screenshots you took above.

[



- Binary Search Tree After operations



- AVL tree After operations

Both BST and AVL have a same insertion process like depending on the number that would go left if it is lower value and go right if it is higher value. However, a main difference between BST and AVL is that AVL has rotations for making the tree balanced. Unlike BST, AVL does extra work to check the value of balance factor and make its tree balanced, which means the depth of left and right children is balanced. During the insertion in AVL, it rotates 6 times to be balanced.]

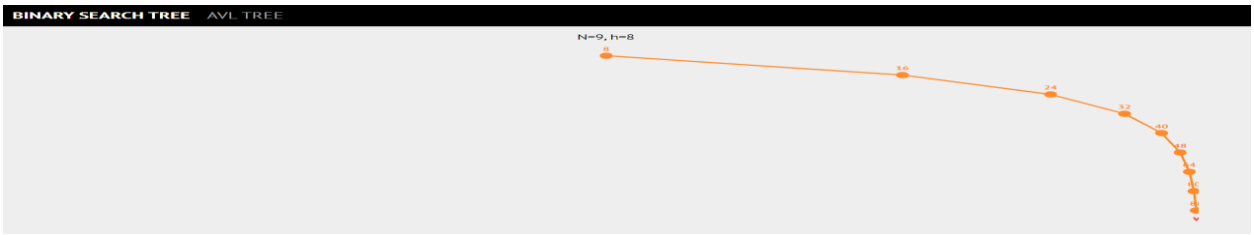
Step #4: Run the same operations in a different order

Now, you'll again insert the same data as above into both a BST and an AVL tree but using a different ordering of the operations. Specifically, starting from an empty tree, run the following sequence of operations in both a BST and an AVL tree:

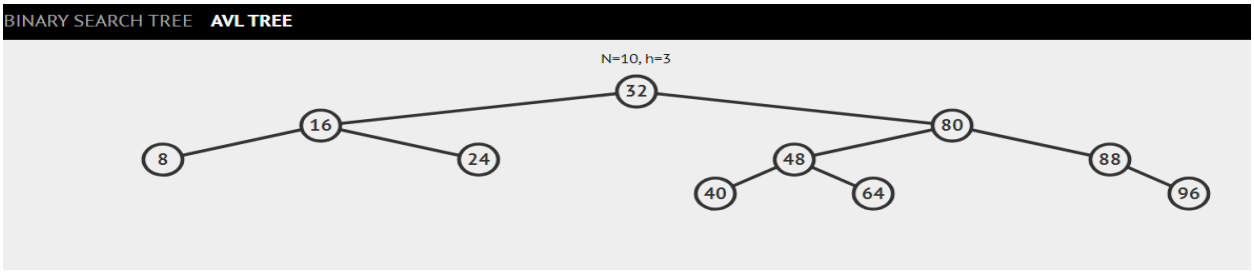
```
insert(8)
insert(16)
insert(24)
insert(32)
insert(40)
insert(48)
insert(64)
insert(80)
insert(88)
insert(96)
```

Again, describe below what were the similarities and differences in the way each different kind of tree executed the same sequence of instructions. How did the finished trees differ in the end? How many rotations were performed by the AVL tree implementation? How did the finished trees you constructed here differ from the finished trees you constructed above in step #3? Include screenshots of both trees again here.

[



- Binary Search Tree After Insertions

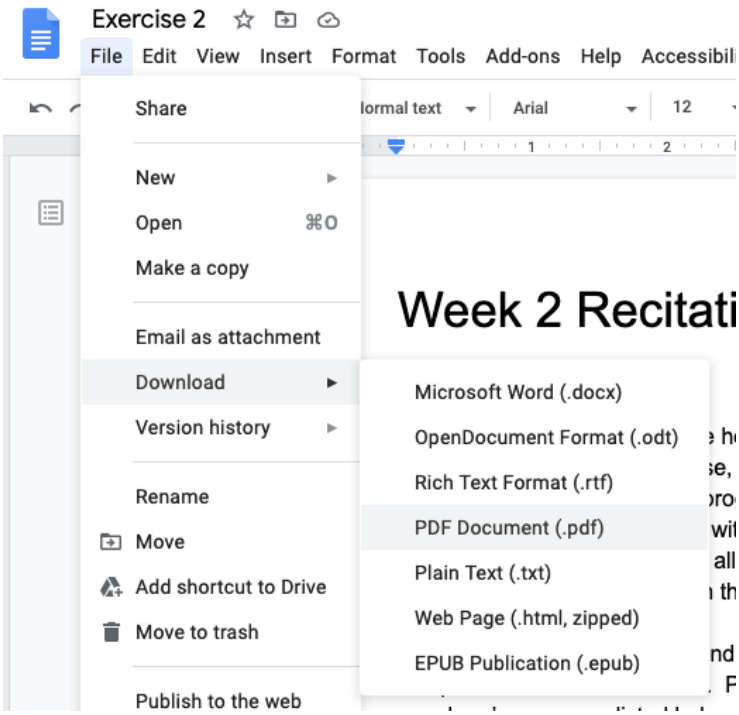


- AVL tree After Insertions

As the pictures above, the results of both BST and AVL are totally different. After whole insertions, BST shows the right-skewed tree whereas AVL tree shows balanced tree. To achieve this balanced tree in AVL, it rotates 6 times to be balanced. Unlike #3, this exercise can be explained that when insertion operation is skewed, which means regular BST has weakness about the numbers inserted.]

Submission and wrap-up

Once you've completed this exercise, download your completed worksheet as a PDF and submit that PDF on Canvas (refer to the image below to help find how to download a PDF from Google Docs). Again, since you're working individually on this exercise, make sure you submit your own copy of this document on Canvas.


Exercise 2
File Edit View Insert Format Tools Add-ons Help Accessibil
Share
New
Open ⌘O
Make a copy
Email as attachment
Download
Version history
Rename
Move
Add shortcut to Drive
Move to trash
Publish to the web
Microsoft Word (.docx)
OpenDocument Format (.odt)
Rich Text Format (.rtf)
PDF Document (.pdf)
Plain Text (.txt)
Web Page (.html, zipped)
EPUB Publication (.epub)
Arial
12
Week 2 Recitati