Q1 Fun with AVL Trees!

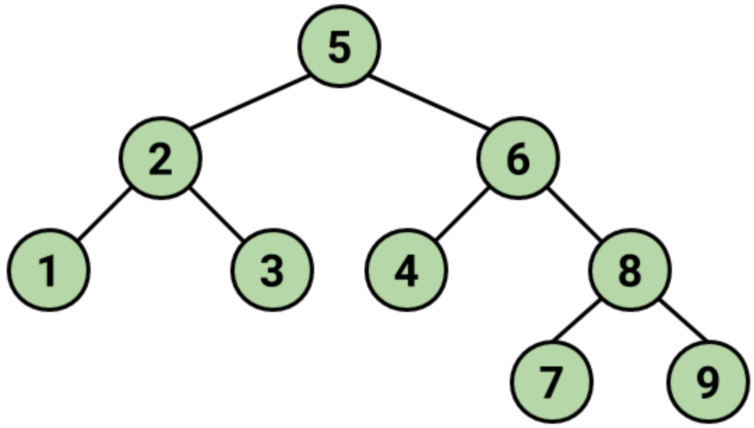
4 Points

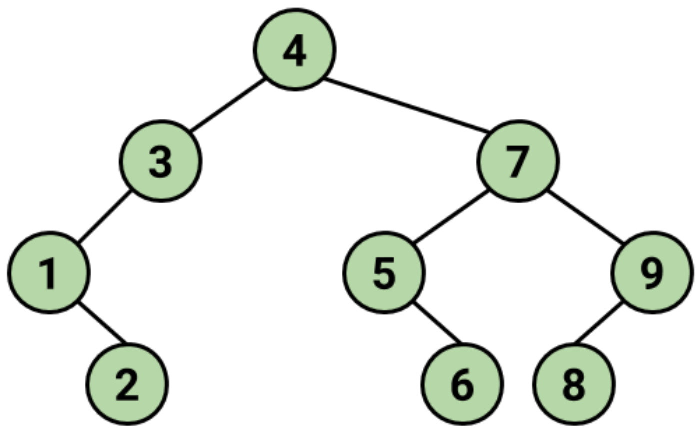
From lecture, we learned that an AVL tree is a binary search tree that respects the AVL tree invariant:

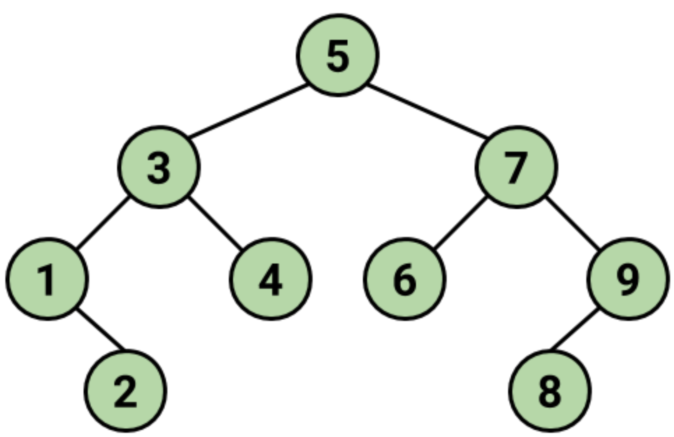
For every node, the height of its left and right subtrees may only differ by at most 1 (balance factor is either -1, 0, or 1).

Q1.1 Valid? Or Nah
1 Point

Which of the following is a valid AVL tree?


5
2
6
1
3
4
8
7
9

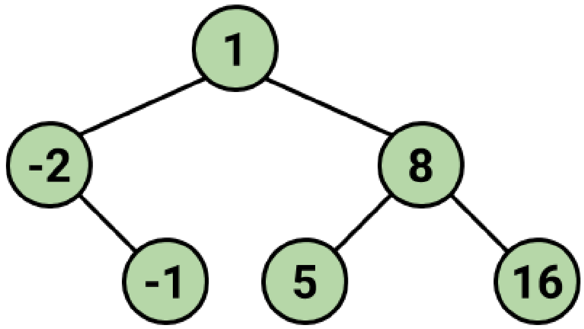

4
3
7
1
5
9
2
6
8


5
3
7
1
4
6
9
2
8

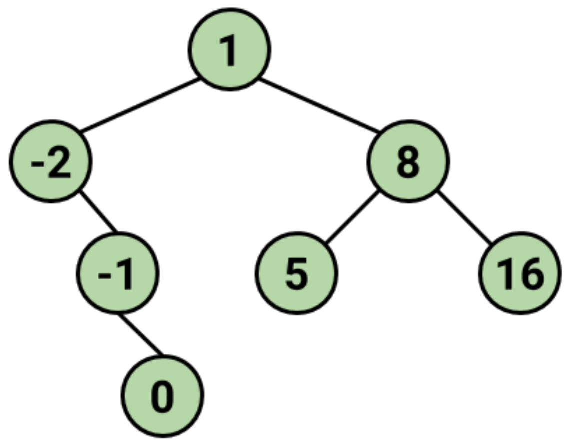
Save Answer

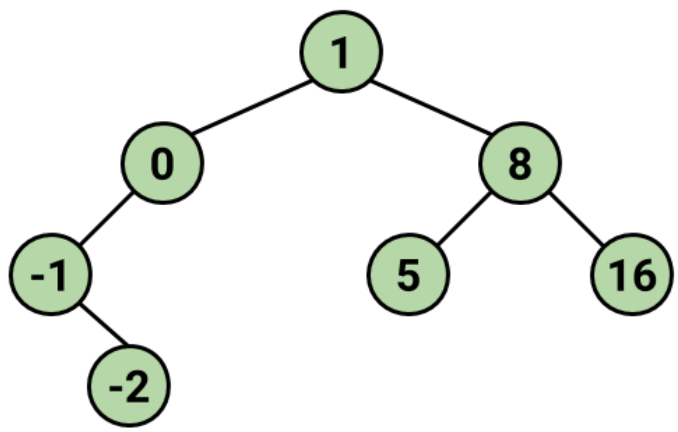
Q1.2 Simple Insertion
1 Point

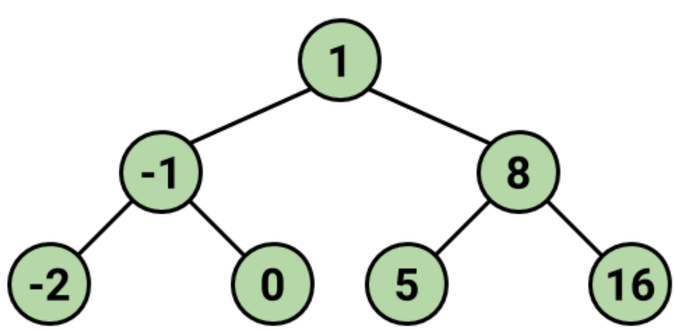
Given the following AVL Tree:

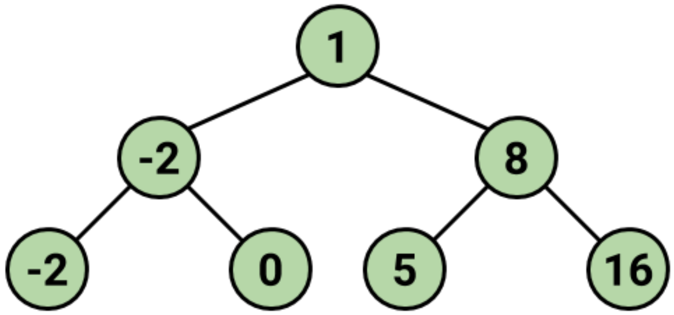


Which tree represent the result of inserting the value 0?


1
-2
8
-1
5
16
0


1
0
8
-1
5
16
-2


1
-1
8
-2
0
5
16



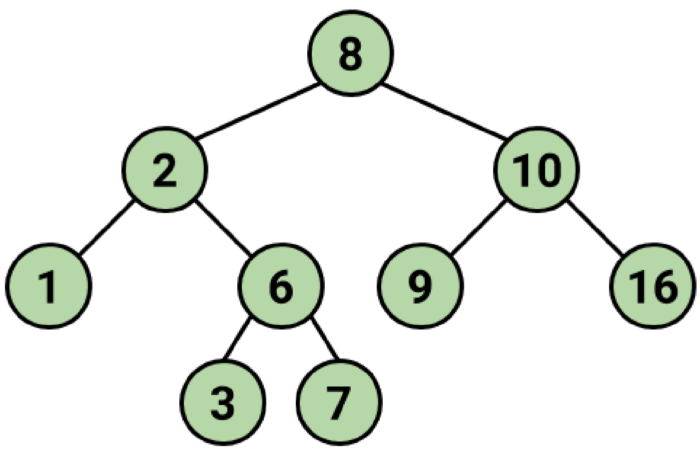
Save Answer

Q1.3 Thinking about Insertions...
1 Point

Suppose we followed the BST insertion procedure to add the value 4 to the AVL tree below. Which series of nodes violate the AVL tree invariant, where the first node in the sequence represents the line or kink that triggers a height imbalance?

Essentially, what this question is asking is when we add the node 4 into our given AVL Tree, immediately after adding the node 4 (with no rotations done yet) what is the 3-node series immediately preceding the added 4 node that violates the kink / line invariant?

In other words, after we add the node 4, what is 4's great-grandparent --> grandparent --> parent node series that is a kink / line?



8-2-1

8-2-6

2-1-4

2-6-3

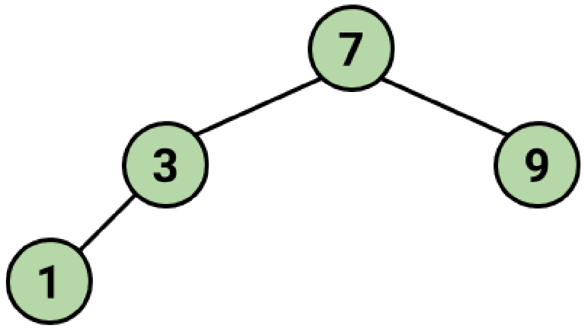
2-6-7

None of the above.

Save Answer

Q1.4 Let's break this case.... WIDE OPEN!
1 Point

What integer(s), if inserted into the tree below, would initially break the AVL invariant and require self-balancing rotations?



- [ ] 0
- [ ] 2
- [ ] 4
- [ ] 6
- [ ] 8

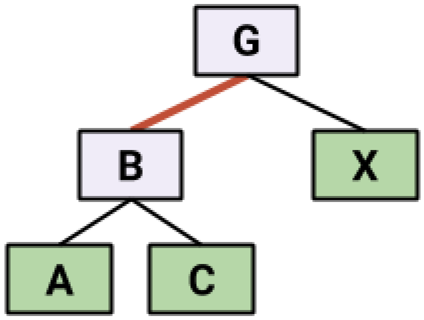
Save Answer

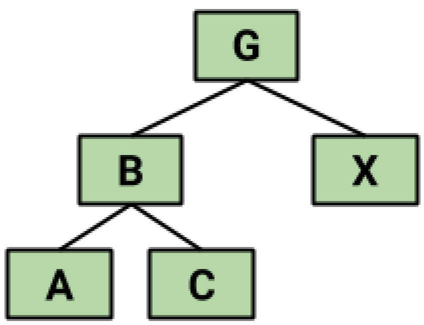
Q2 LLRB Trees 😻
3 Points

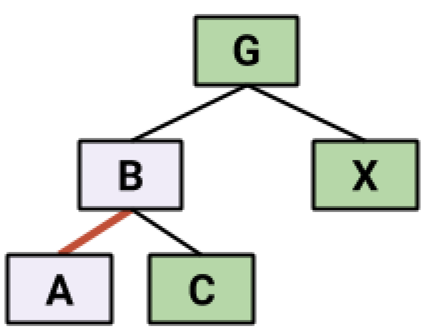
Let's talk about ❤️ 🖤 🌳 s!

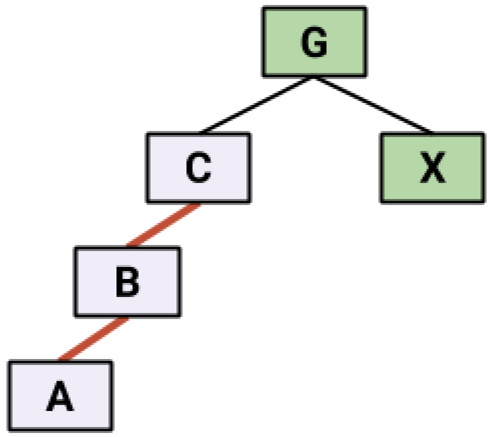
Q2.1 Valid???
1 Point

Which of the following are valid LLRB trees?


G
B
X
A
C


G
B
X
A
C

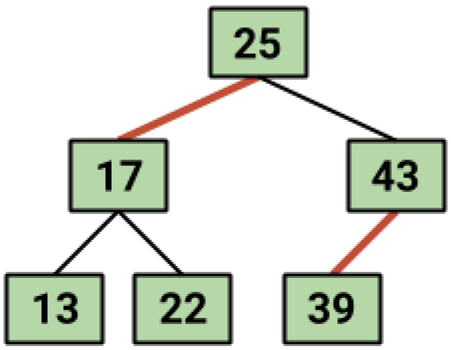

G
B
X
A
C



Save Answer

Q2.2 How do we operate?
1 Point

If we add 15 to the LLRB tree below, what operation will we need to perform to maintain LLRB invariants? Select a method call below to either `rotateLeft(n)` or `rotateRight(n)` where `n` is replaced with the value of the node.

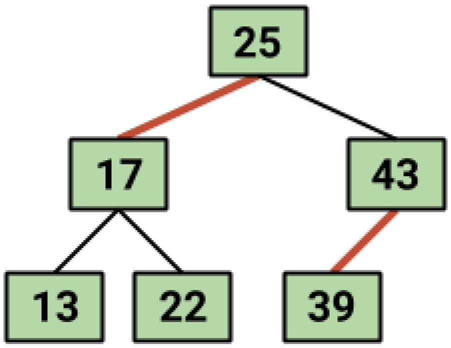


- `rotateLeft(39)`
- `rotateRight(13)`
- `rotateLeft(13)`
- `rotateRight(39)`

Save Answer

Q2.3 Let's rotate
1 Point

Which of the following values requires an immediate `rotateRight` operation when inserted into the LLRB tree below?



0

15

19

24

36

40

45

Save Answer

Q3 AVL vs. LLRB Tree Go Head to Head 🥊

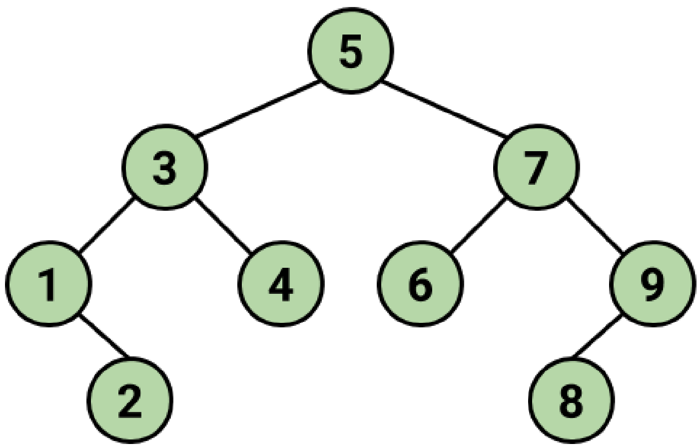
1 Point

Let's compare and contrast AVL trees with left-leaning red-black trees.

Q3.1 AVL to LLRB Tree?

0 Points

Consider the valid AVL tree below. Is it possible to color some edges red so that it is a valid LLRB tree?



True

False

Save Answer

Q3.2
1 Point

We know that worst-case search trees are as unbalanced as possible between the left and right subtrees. Are worst-case AVL trees more unbalanced than worst-case LLRB trees? Think about this question in terms of H, the height of the AVL or LLRB tree.

Worst-case AVL trees are more unbalanced than worst-case LLRB trees.

Worst-case LLRB trees are more unbalanced than worst-case AVL trees.

Worst-case AVL trees are about as unbalanced as worst-case LLRB trees.

Save Answer

Q4 Try some Tries!
3 Points

Q4.1 Sonic the Hedgehog's Family Trie
1 Point

Say you were making a `Trie` based on Sonic the Hedgehog's family. First you insert the `String` type `Sonic` term into your `Trie`. Then you inserted Sonic's sister, `Sonia` into your `Trie`. After insertion of both terms into your `Trie`, which node will be the only node in your `Trie` with ***exactly*** two children?

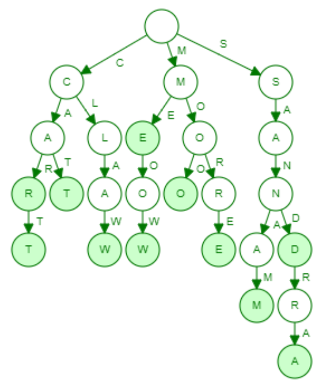
- s
- o
- n
- i
- a
- c

Save Answer

Q4.2 How many?
1 Point

Observe the `Trie` below. In total, how many keys are in this `Trie`?

NOTE: You may view the Trie in fullscreen by hovering over the image and clicking on the expanding arrows button.



Animation Completed

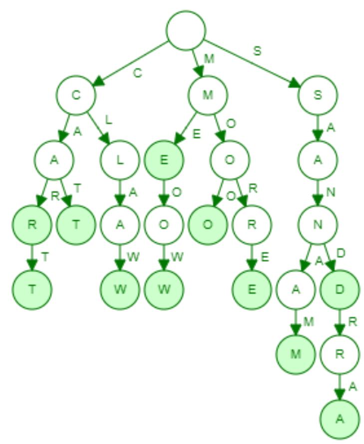
Please enter your answer as an integer.

Save Answer

Q4.3 Harvesting Tries
1 Point

Observe the `Trie` below. What is the result of calling `collect()` on this `Trie`?

NOTE: You may view the Trie in fullscreen by hovering over the image and clicking o the expanding arrows button.



Animation Completed

- ["car", "cart", "cat", "claw", "me", "meow", "moo", "more", "sanam", "sand", "sandr
- ["car", "cart", "cat", "law", "me", "meow", "moo", "moore", "sanam", "sand", "sandr
- ["cart", "cat", "claw", "me", "mew", "mo", "more", "sanam", "sand", "sandra"]

Save Answer

Q5 Battle of the Trees

5 Points

The first step in parsing a language often involves matching an input string against a set of a patterns.

In this problem, we allow a pattern to be any sequence of alphabetic letters (e.g., 'a', 'b', ... , 'z') and the special character '?' (which can be any single alphabetic character). For example, the pattern `ab?d?f` would match the inputs “abcdef”, “abbddf”, and “abzdzf”, but it would not match “bbbbbb” or “abc”.

We call a set of patterns unambiguous if no input string can be matched by two or more patterns in the set. For example, the set of patterns `{a, bc, ?a}` is unambiguous, but the set of patterns `{a, bc, ?c}` is ambiguous because the patterns `bc` and `?c` both match the input string “bc”.

Given a set of patterns, briefly describe an efficient algorithm that determines whether the set is ambiguous. You may use one of the following data structures to help you: **BST, AVL Tree, Trie, LLRB Tree**.

Your description should take 1-2 sentences. Be sure to also justify the efficiency of your answer by providing the **worst case running time** of your algorithm in $O(\cdot)$ notation. Use the variable n to represent the number of patterns in your set.

Save Answer

Save All Answers

Submit & View Submission >