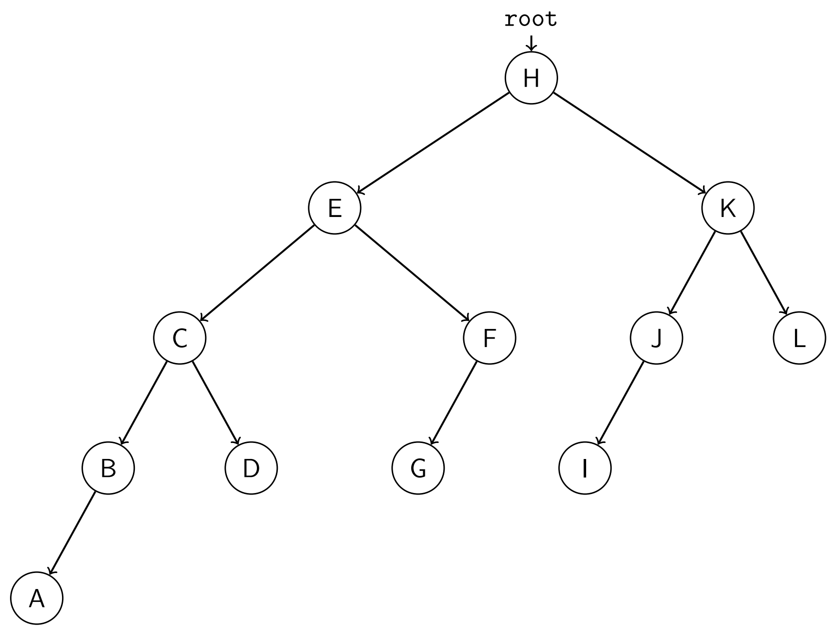
CSE 332: Data Structures and Parallelism

Section 4: Balanced Trees Solutions

0. MinVL Trees

Draw an AVL tree of height 4 that contains the minimum possible number of nodes.

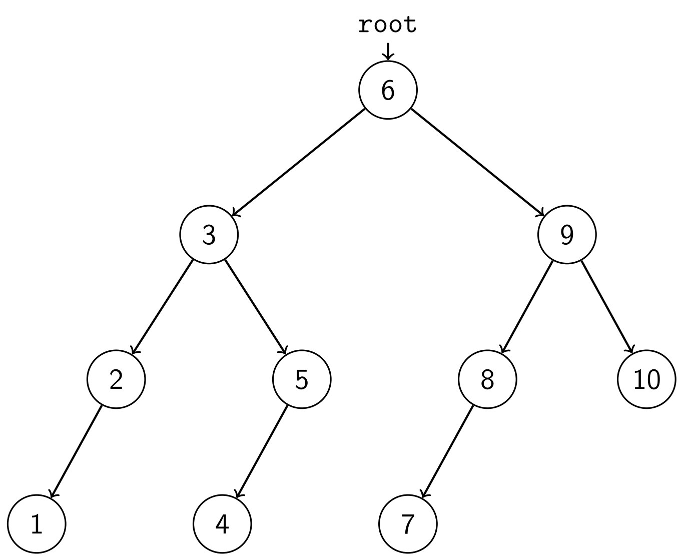
Solution:



1. AVL Trees

Insert 6, 5, 4, 3, 2, 1, 10, 9, 8, 7 into an initially empty AVL Tree.

Solution:



2. AVL Trees

Given a binary search tree, describe how you could convert it into an AVL tree with worst-case time $\mathcal{O}(n \lg(n))$. What is the best case runtime of your algorithm?

Solution:

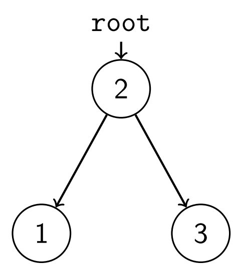
Since we already have a BST, we can do an in-order traversal on the tree to get a sorted array of nodes. We could now simply insert all of these nodes back into an AVL tree using rotations which would give us an $\mathcal{O}(n \lg(n))$ runtime.

3. HeapVL Trees

Is there an AVL Tree that isn't a binary min heap? Is there a binary min heap that isn't an AVL tree? Is there a binary search tree that is neither? Is there a binary search tree that is both?

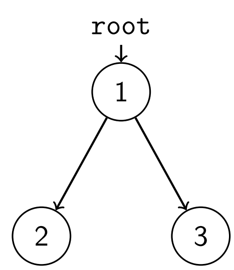
Solution:

Is there an AVL Tree that isn't a binary min heap? Yes. Consider the following:



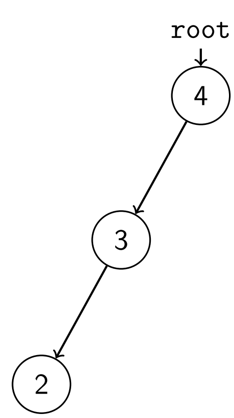
This tree meets the AVL properties - left children are smaller, right children are larger, and the tree is balanced. But it does not meet the ordering property of a binary min heap - no children can be smaller than the root.

Is there a binary min heap that isn't an AVL tree? Yes. Consider the following:



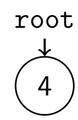
This meets the properties of a binary min heap, but is not in the correct format for an AVL tree since the left child should be smaller than the root value.

Is there a binary search tree that is neither? Yes. Consider the following:



This BST is unbalanced, so it is not an AVL tree. It also is not a complete tree (it is missing the right child of 4), so it cannot be a binary heap (min heap or max heap) since it doesn't meet the structure property.

Is there a binary search tree that is both? Yes. Consider the following:

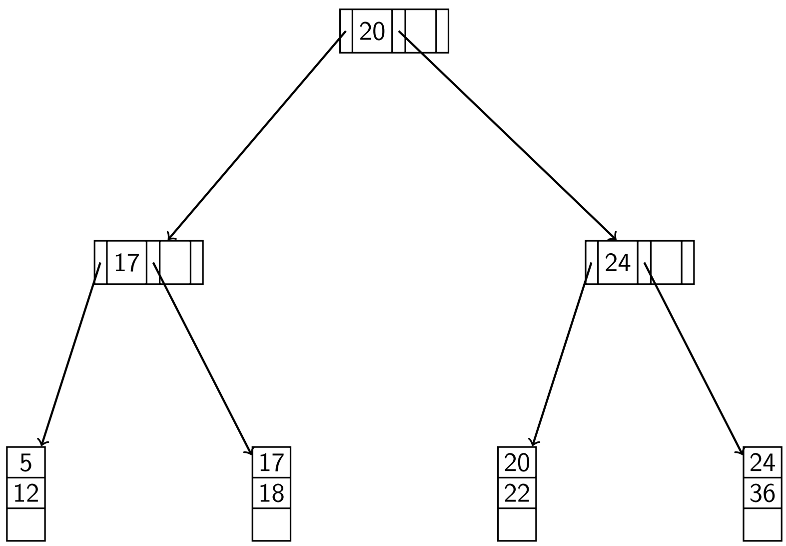


This is both a valid heap (min or max) and a valid AVL tree!

4. B-Trees

(a) Insert the following into an empty B-Tree with $M = 3$ and $L = 3$: $12, 24, 36, 17, 18, 5, 22, 20$.

Solution:



(b) Delete $17, 12, 22, 5, 36$

Solution:

18
20
24

(c) Given the following parameters for a B-Tree with $M = 11$ and $L = 8$

- Key Size = 10 bytes
- Pointer Size = 2 bytes
- Key/Value pair Size = 16 bytes per record

Assuming that M and L were chosen appropriately, what is the likely disk block size on the machine where this implementation will be deployed? Give a numeric answer and a short justification based on two equations using the parameter values above.

Solution:

We use the following two equations to find M and L to fit as best as possible in the block size, where:

- 1 block on disk is b bytes
- Keys are k bytes
- Pointers are t bytes
- Key/Value pairs are v bytes

We know that: $(M-1)*k + M*t \leq b$ and $L*v \leq b$, so:
$M = \left\lfloor \frac{b+k}{t+k} \right\rfloor$ and $L = \lfloor \frac{b}{v} \rfloor$

Plugging in the given values, we get:

$M = \left\lfloor \frac{b+10}{2+10} \right\rfloor$ and $L = \lfloor \frac{b}{16} \rfloor$

And solving for b gives us an answer of 128 bytes.