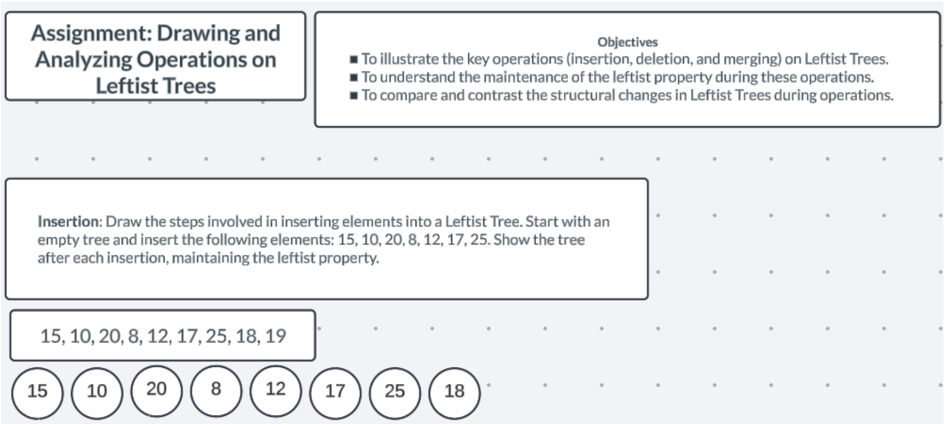
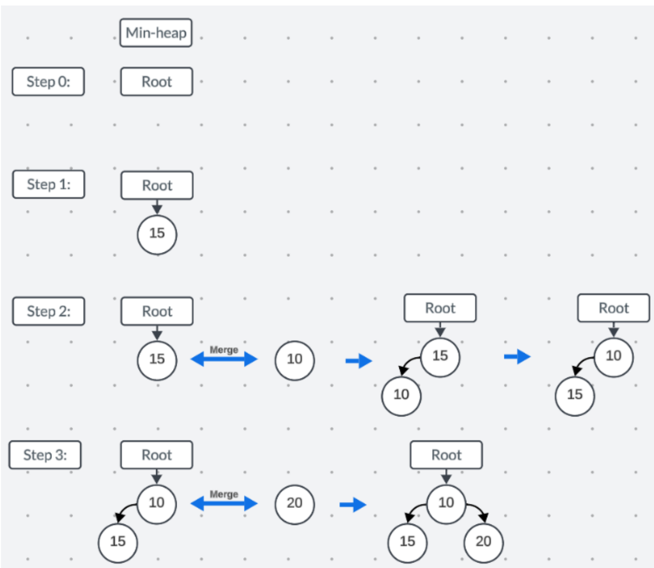
Keiffer Tan

Assignment: Drawing and Analyzing Operations on Leftist Trees



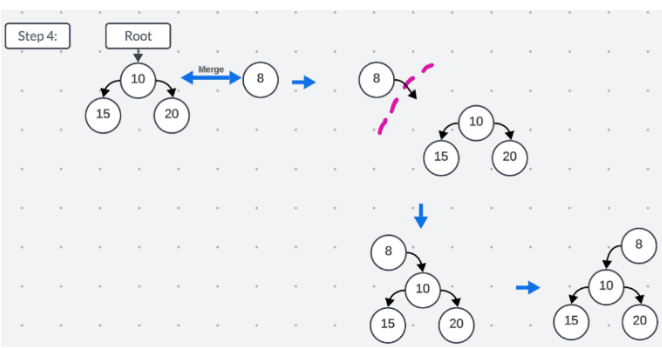


Start off with an empty left heap tree.

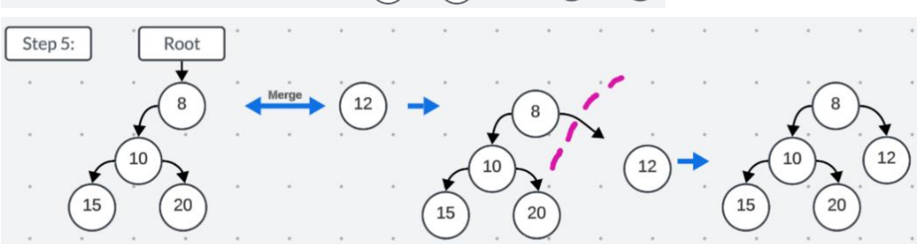
With every sequential insert, it almost acts like the node is a “new tree” and you want to merge it with the existing tree.

When inserting (10), since it is the node with the lesser key, the (15) will become its left child.

When inserting (20), it just becomes the root’s (10) right child

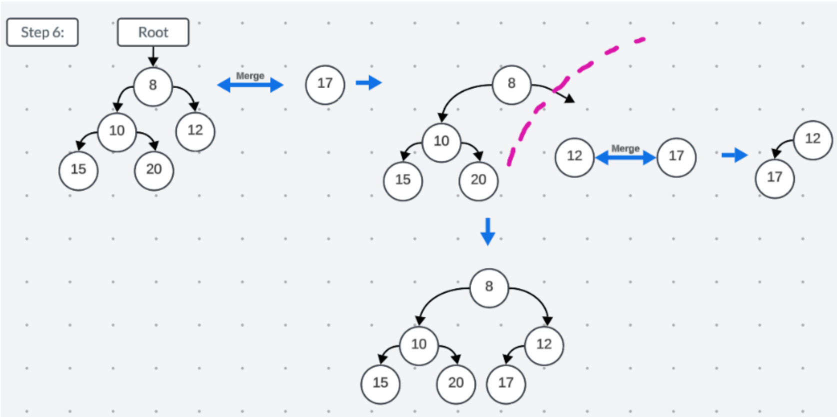


When inserting (8), since it is the lesser key of both “root” nodes. The (10) is becomes the (8)’s right child, but to meet the leftist tree property, we need to swap the (10) from the right to the left child



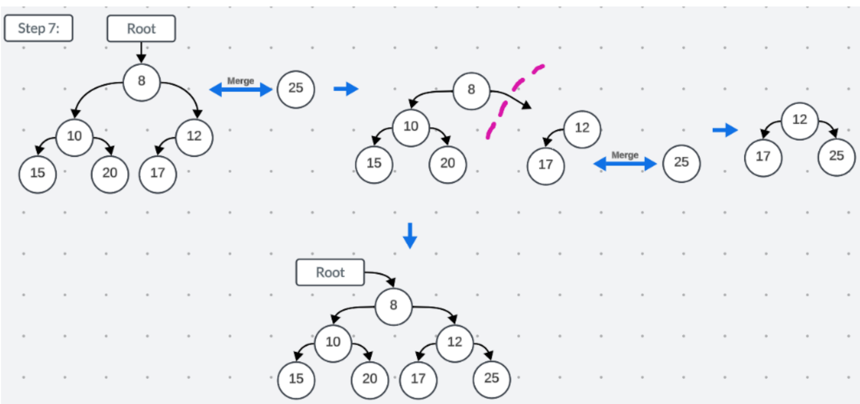
With inserting the (12), there is no right child, so we can just insert it in there.

Keiffer Tan



When inserting (17), we take the right sub-tree of (8) and merge it with (17). This makes a subtree of (12) with a left child of (17).

That subtree will become the new right subtree of (8)



Inserting (25), we take the right subtree of (8) and to a recursive merge.

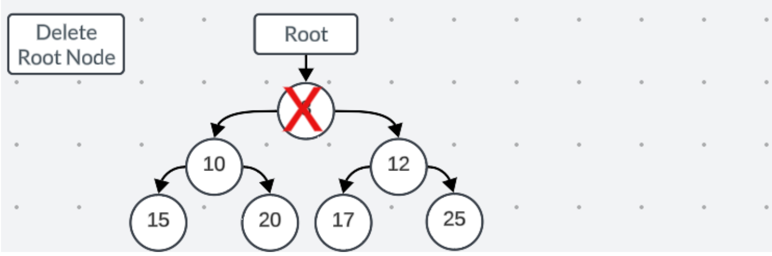
Taking the subtree, we can just insert (25) into the right child of (12).

From there, we can take the merged subtree, and append it to the right child of the root (8).

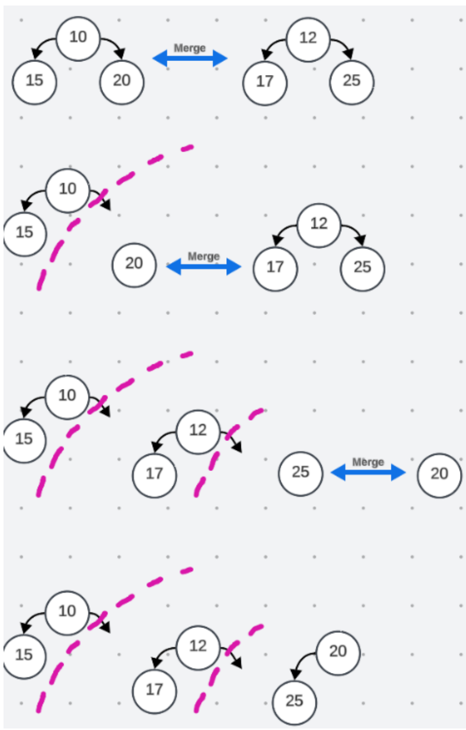
Keiffer Tan

Deletion: Continue from the final Leftist Tree in the insertion task and show the steps for deleting the minimum element three times. Demonstrate how the leftist property is maintained after each deletion.

- Start with the final tree from the insertion task.
- Delete the minimum element (the root) three times.
- After each deletion, show the tree structure and explain how the leftist property is restored (e.g., by merging the children of the deleted root)



When deleting the root, we take both right and left subtrees and merge those 2 together.

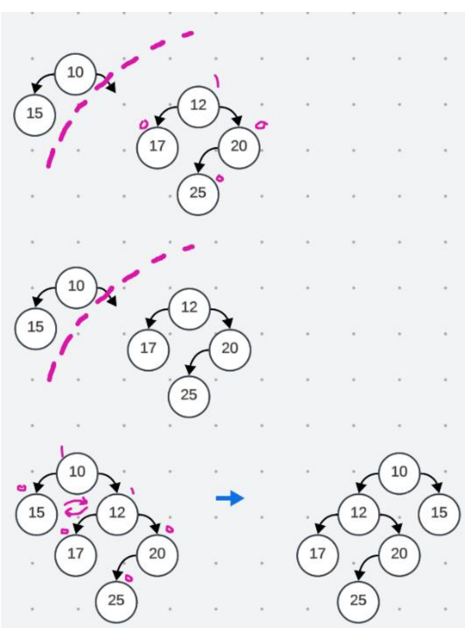


When merging two trees, we take the right subtree of the tree with the lesser key value (8) and merge that with the 2nd tree.

Since the (10) tree has the smaller key value, that will be the root of the new tree.

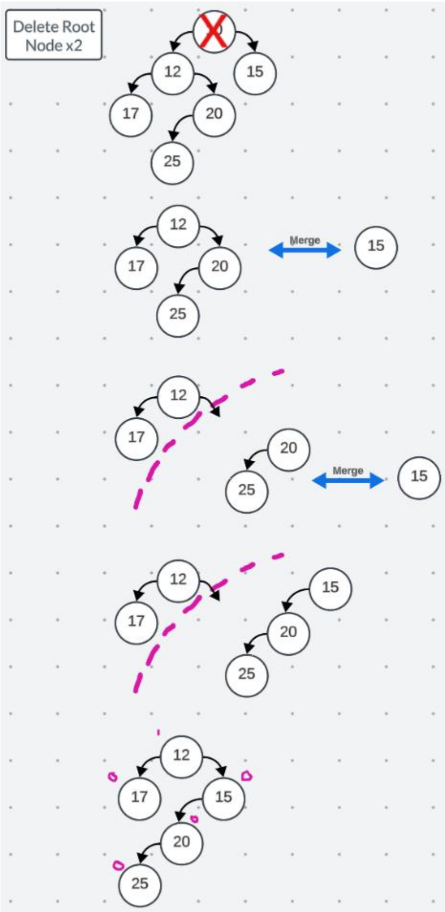
Thus we, merge (20) with the tree of root (12). Which we merge (20) with the right subtree of (12).

We bring all those merges back and reassess the level.



Everything looks fine until we append the right subtree back to the main tree. Since the levels are off, we need to switch the right and left child to maintain a leftist tree.

Keiffer Tan

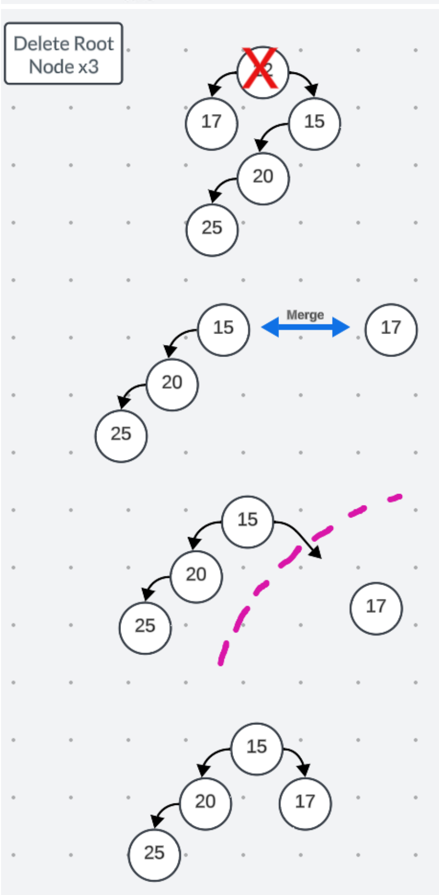


Deleting the root a second time is much simpler than the first time.

We take the right subtree of (12) and merge it with (15).

Since (15) is the lesser key value, we make the (20) the left child of (15)

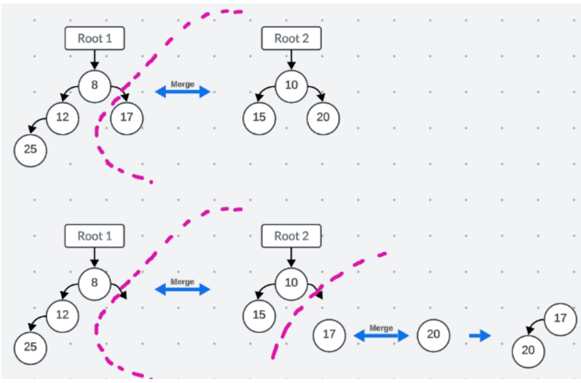
Assessing the levels, there is no issue, so we can move to the 3rd deletion.



The 3rd deletion is the simplest, as (15) is the lesser key value and has an open right child, we can just append (17) to the right child of (15)

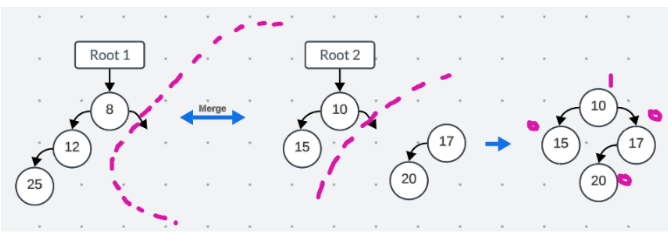
Keiffer Tan

> **Mergin**: Illustrate the merging of two Leftist Trees. Show the steps to merge these two trees into a single Leftist Tree, ensuring the leftist property is maintained throughout the process.
>
> - Tree 1: 15, 10, 20
> - Tree 2: 8, 12, 17, 25

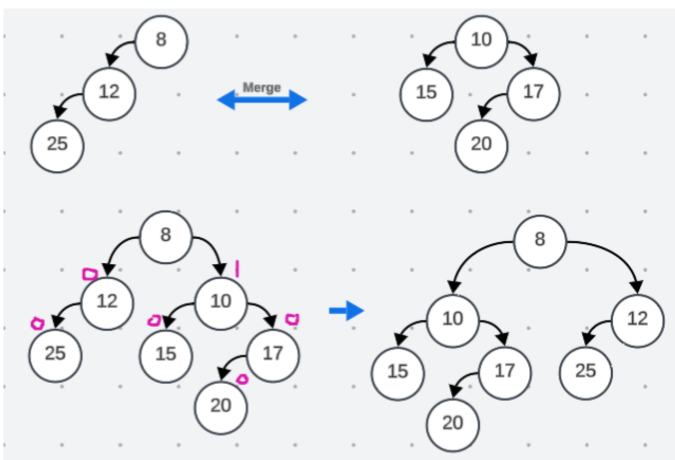


When merging 2 trees, we need to assess which one will be the new root, and that new root will be the tree with (8). We need to merge the right subtree of (8) tree with the 2nd tree.

We need to recursively merge as the root of the 2nd tree is smaller than (17). We merge the (17) with the right child of (10).



Bringing it back, we need to append the (17) as the right child, and everything still seems balanced, so no extra actions needed.



Once, we append the merged right subtree back to the main tree, there are issues with balance, so we do need to swap the left and right child to restore leftist tree property.

The structure and height of the tree really change with insertions, deletions, and merging for a leftist tree. There are very specific rules, and technically it is all “merging” operations. There will definitely be more time complexity with leftist trees as there are more technical operations needed to keep a leftist balance after every operation. Especially with merging recursions.