18.3 Inserting LLRB Trees

Some Tree Maintenance.

We can always insert into an LLRB tree by inserting into a 2-3 tree and converting it as we've done before. However, this would be contrary to our original purpose of creating LLRBs, which was to avoid the complicated implementation of a 2-3 tree!

Instead, we **insert into the LLRB as we would with a normal BST.** However, this could break its 1-1 mapping to a 2-3 tree, so we will use **rotations** to massage the tree back into a proper structure.

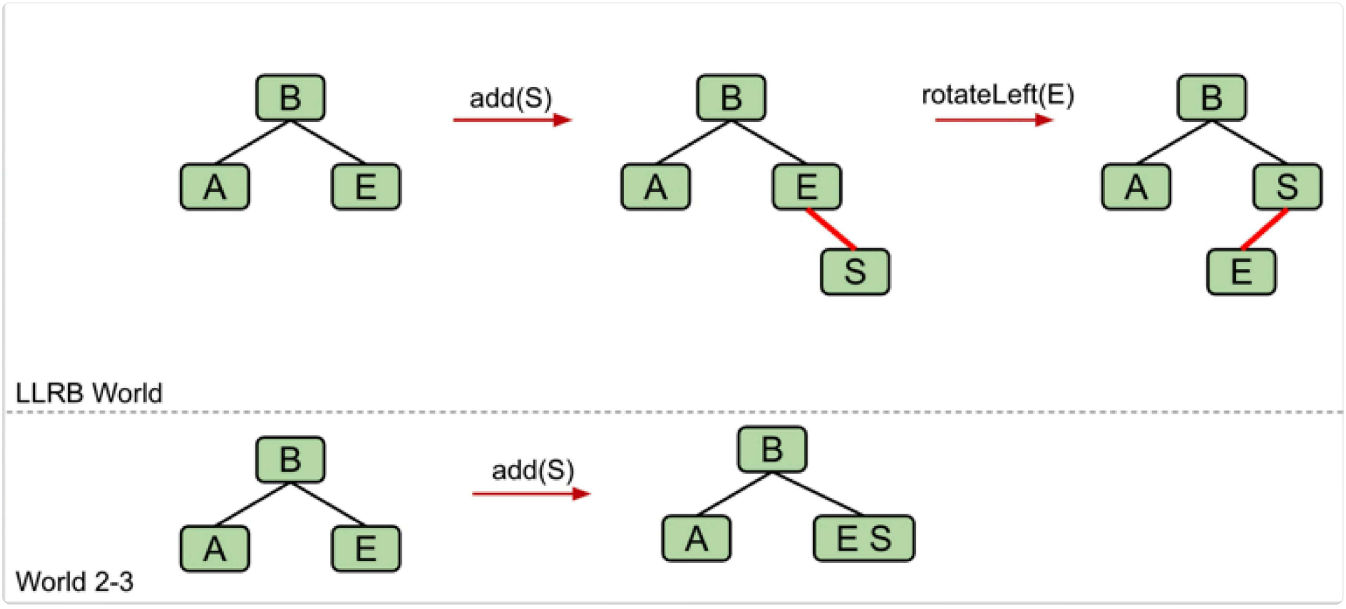
Maintaining Proper Structure

When inserting into an LLRB tree, we always insert the new node with a **red link** to its parent node. This is because in a 2-3 tree, we are always inserting by adding to a leaf node, the color of the link we add should always be red.

But sometimes, inserting red links at certain places might lead to cases where we break one of the invariants of LLRBs.

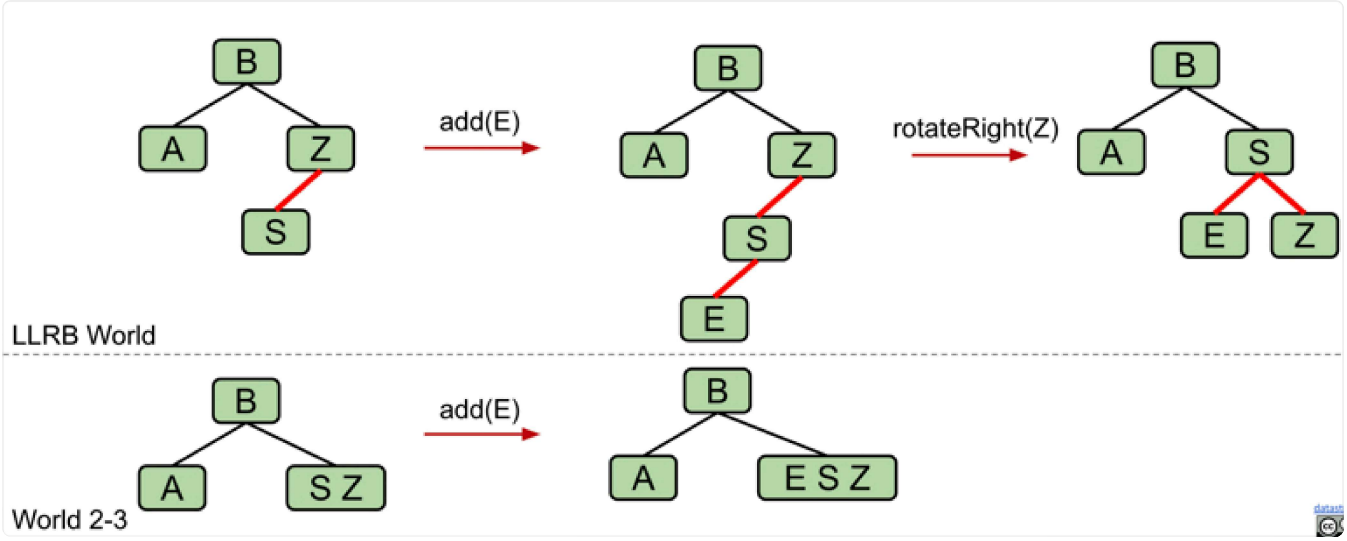
Below are three such cases where we need to perform certain tasks address in order to maintain the LLRB tree's proper structure:

- Case 1: **Insertion results in a right leaning "3-node → Left Leaning Violation**
  - **Task 1: rotateLeft(node)**
  - Recall, we are using *left-leaning* red black trees, which means we can never have a right red link.
  - To address this, we will need to use a **rotation** in order to maintain the LLRB invariant: "There are no right red links".
  - If the left child is *also* a red link, then we will temporarily allow it for purposes that will become clearer in case 2.



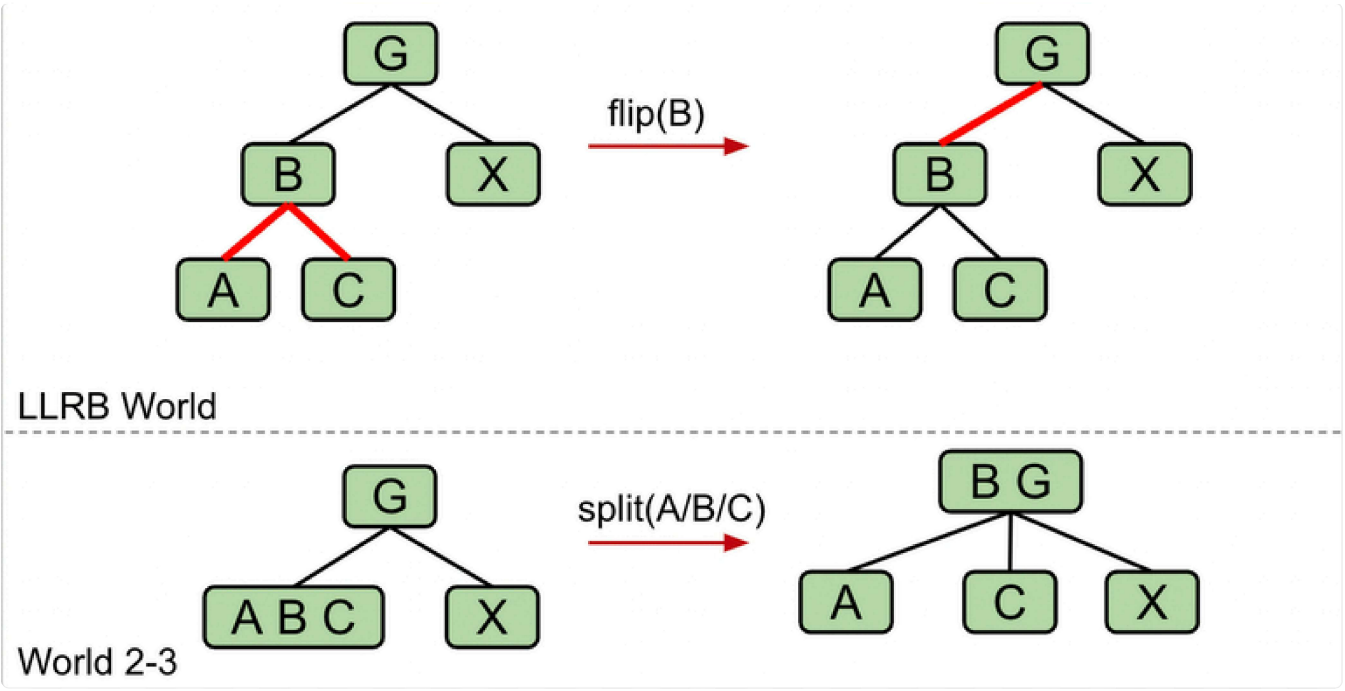
Case 1: Inserting a Right Red Link

- Case 2: **Double Insertion on the Left → Incorrect 4 Node Violation**
  - **Task 2: rotateRight(node)**
  - If there are 2 left red links, then we have a 4-node which is illegal.
  - To address this, we will first **rotate** to create the same tree seen in case 1 above. Then, in both situations, we will address the temporary issue of having an illegal 4-node in case 3.



Case 2: Double Insertion on the Left

- Case 3: **Node has Two Red Children → Temporary 4 Node**
  - **Task 3: Color Flip**
  - Finally, to address the issue of having illegal 4-node by **flipping the colors** of all edges touching "S" above.
  - This is equivalent to pushing up the middle node in a 2-3 tree.



Case 3: Two Red Links

Note: You may need to go through a series of rotations in order to complete the transformation. It's possible that one operation will result in additional LLRB rule violations, which we can fix with the corresponding operation.

The process is: while the LLRB tree does not satisfy the 1-1 correspondence with a 2-3 tree or breaks the LLRB invariants, perform task 1, 2, or 3 depending on the condition of the tree until you get a legal LLRB.

Summary of Maintenance Operations

- When inserting: Use a **red link**. Insert in the same way as inserting into a BST.
- If there is a right-leaning "3-node", we have a **Left Leaning Violation**.
  - **Rotate left** the appropriate node to fix.
- If there are two consecutive left links, we have an **Incorrect 4 Node Violation**.
  - **Rotate right** the appropriate node to fix.
- If there are any nodes with two red children, we have a **Temporary 4 Node**.
  - **Color flip** the node to emulate the split operation.

Previous
18.2 Creating LLRB Trees

Next
18.4 Runtime Analysis

Last updated 1 year ago