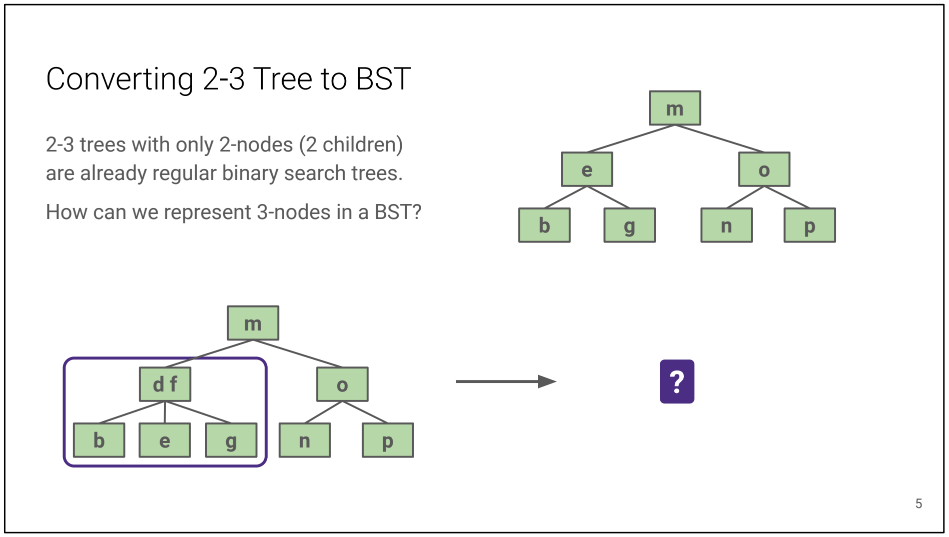
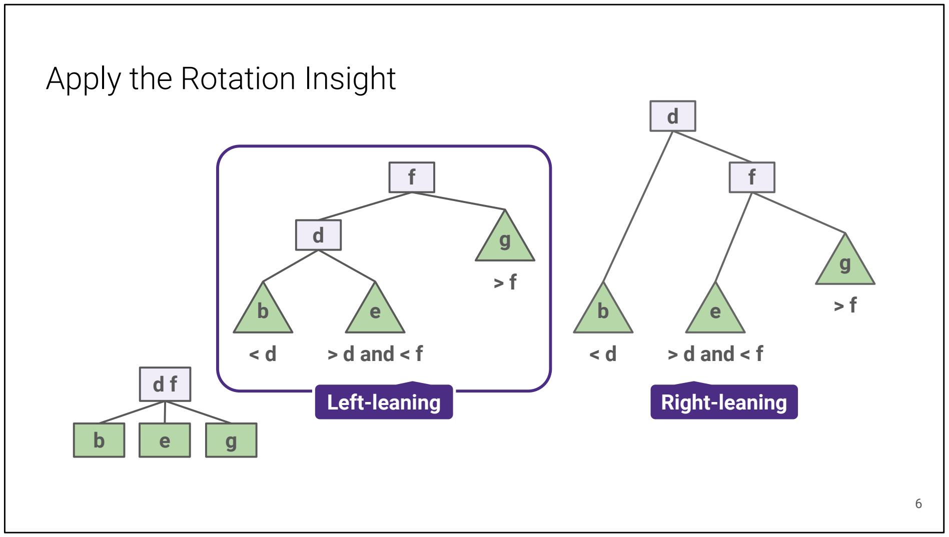


Converting 2-3 Tree to BST

2-3 trees with only 2-nodes (2 children) are already regular binary search trees.

How can we represent 3-nodes in a BST?

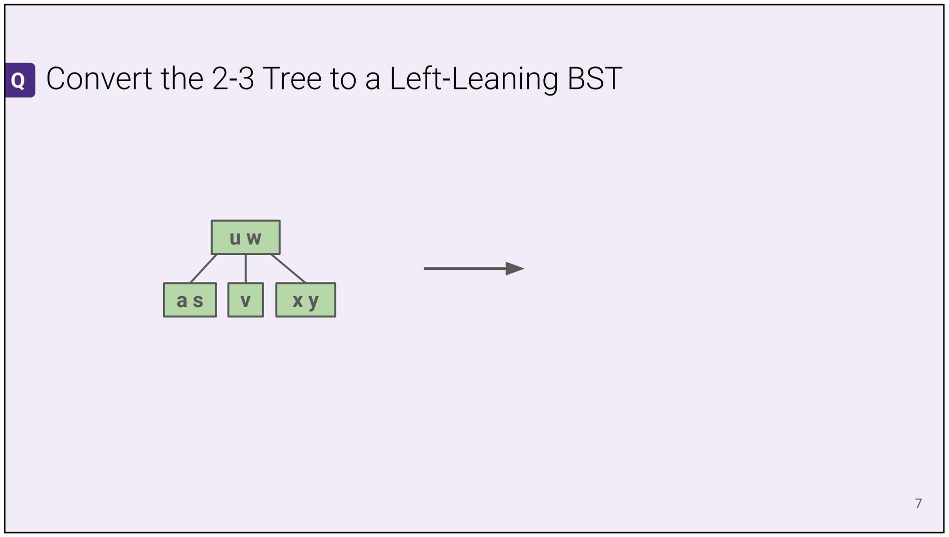
?: How can we represent 3-nodes in a BST?



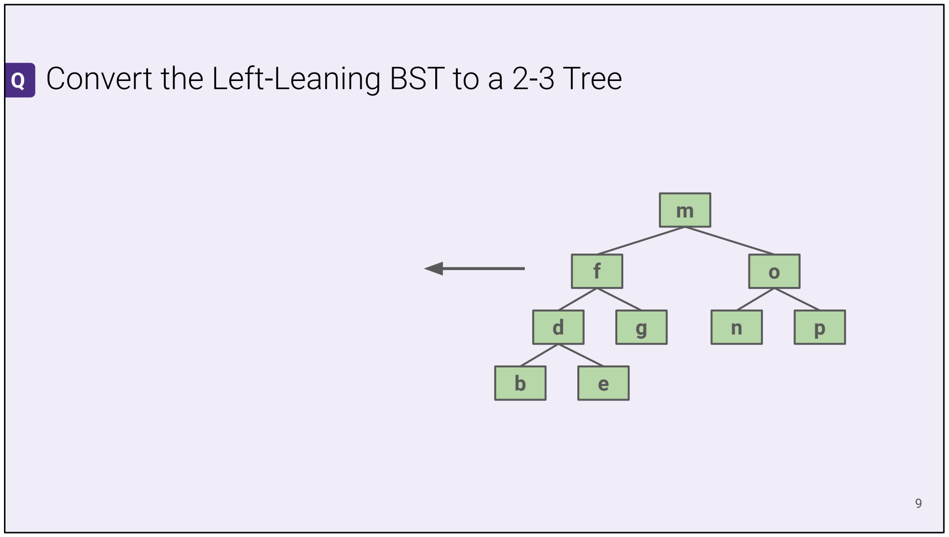
Apply the Rotation Insight

We learned in the reading that rotations, in effect, combine two nodes together and then split them up again into one of these two configurations. However, writing code to handle both cases is unnecessary since we only need one representation. Let's (arbitrarily) choose to use only the **left-leaning representation**.

?: Why does handling both representations take more work?



Q Convert the 2-3 Tree to a Left-Leaning BST

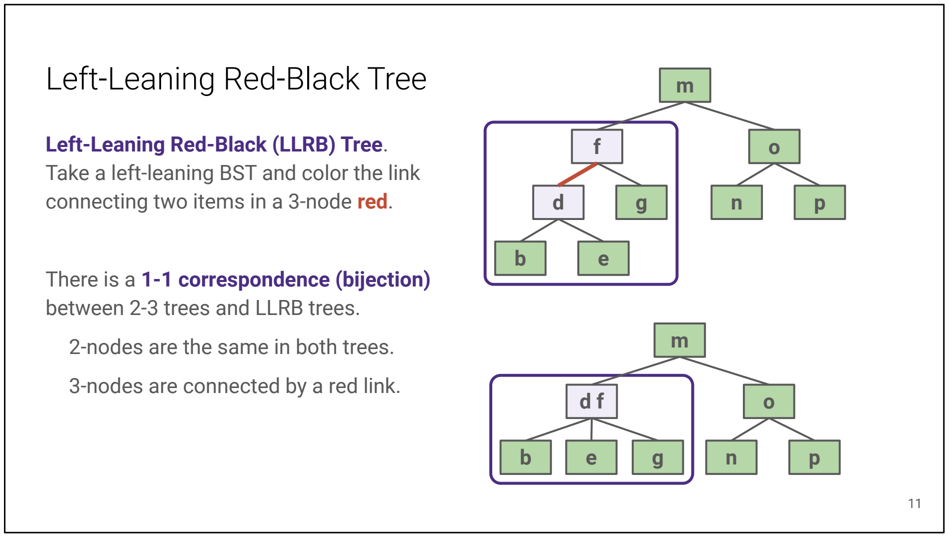


Q Convert the Left-Leaning BST to a 2-3 Tree

Q1: Convert the 2-3 Tree to a Left-Leaning BST.

Q1: Convert the Left-Leaning BST to a 2-3 Tree.

?: How did you determine which nodes were 2-nodes? 3-nodes?

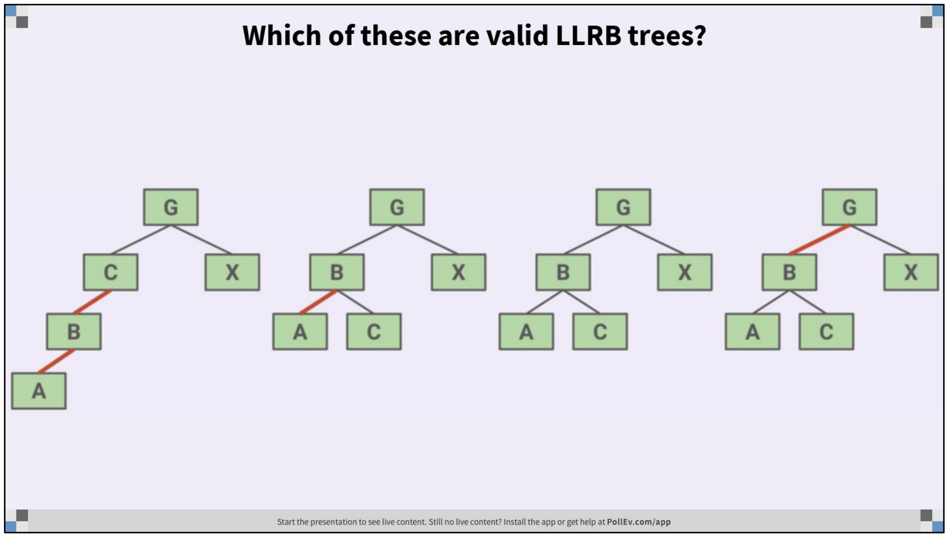


Left-Leaning Red-Black Tree

Left-Leaning Red-Black (LLRB) Tree. Take a left-leaning BST and color the link connecting two items in a 3-node **red**.

There is a **1-1 correspondence (bijection)** between 2-3 trees and LLRB trees.

- 2-nodes are the same in both trees.
- 3-nodes are connected by a red link.



Which of these are valid LLRB trees?

Start the presentation to see live content. Still no live content? Install the app or get help at **PollEv.com/app**

Note that “left-leaning binary search trees” don’t actually exist, but it’s useful to know that the red links role is to make it easier to figure out which nodes are 3-nodes.

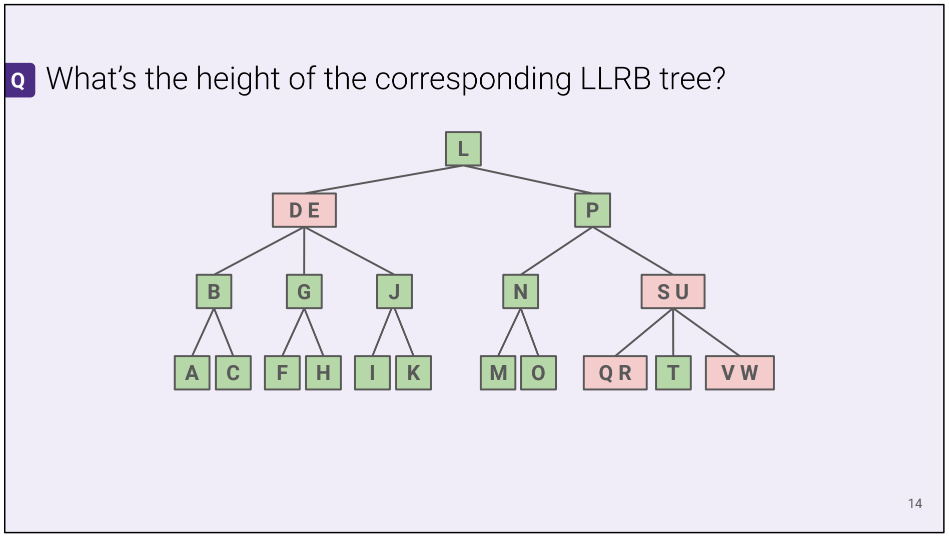
?: Why do red links lean left? Can a red link connect to a right child in an LLRB?

?: What do the black links in an LLRB connect in the analogous 2-3 tree?

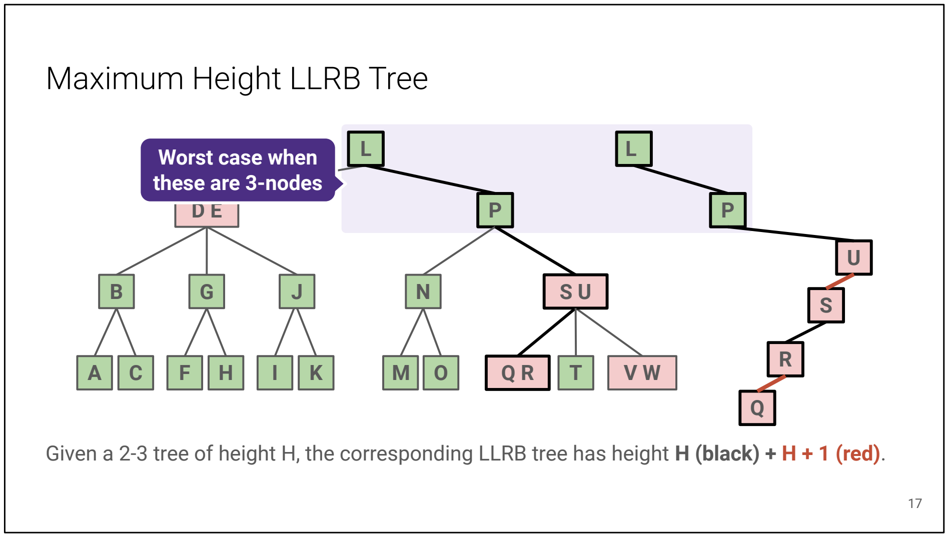
?: What does it mean to have 1-1 correspondence?

Use the fact that LLRB trees have a 1-1 correspondence with 2-3 trees.

Q1: Which of these are valid LLRB trees?



Q What’s the height of the corresponding LLRB tree?



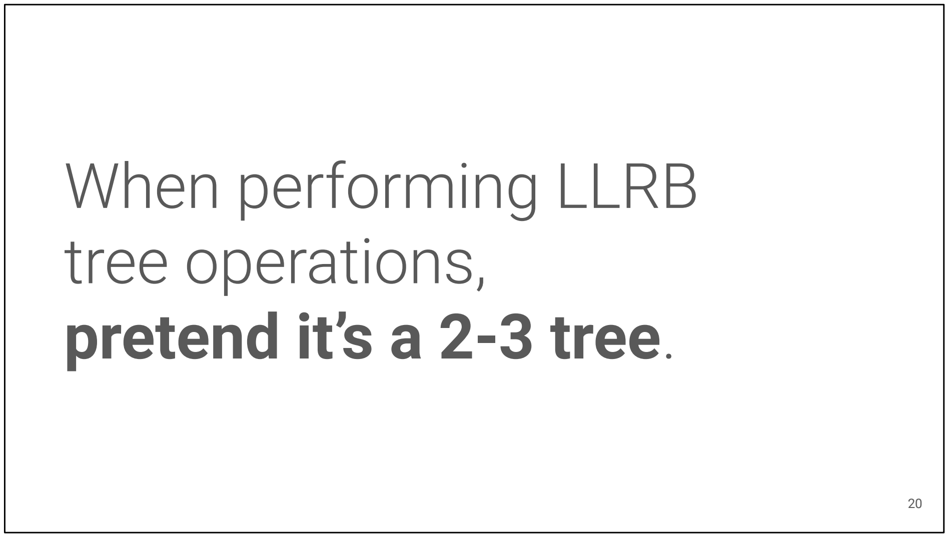
Maximum Height LLRB Tree

Worst case when these are 3-nodes

Given a 2-3 tree of height H, the corresponding LLRB tree has height **H (black)** + **H + 1 (red)**.

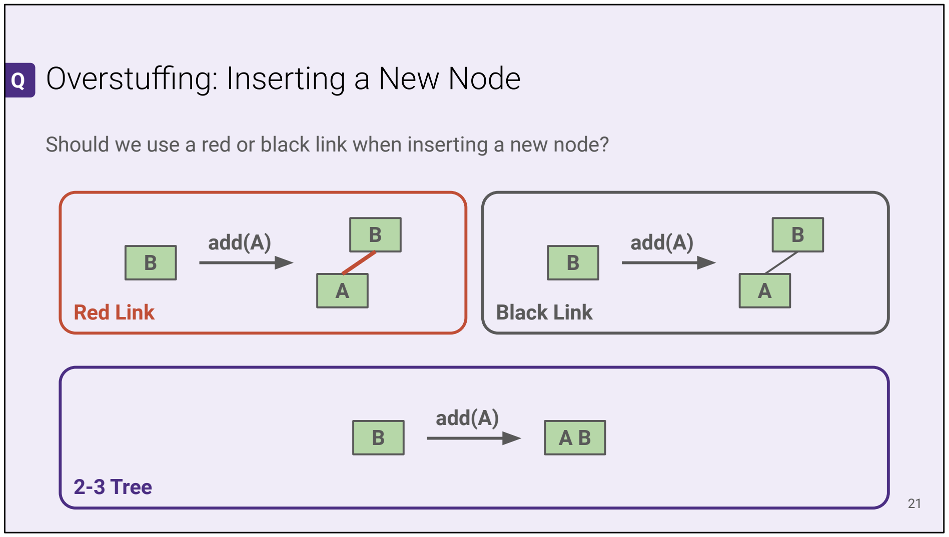
The total height for the corresponding LLRB is 5.

?: If the height of a 2-3 tree is H, what is the maximum possible height of its corresponding LLRB tree?



When performing LLRB tree operations,
pretend it’s a 2-3 tree.

Preservation of the correspondence will involve tree rotations. We want our LLRB tree to function like a 2-3 tree, so let’s design it so that it behave this way!

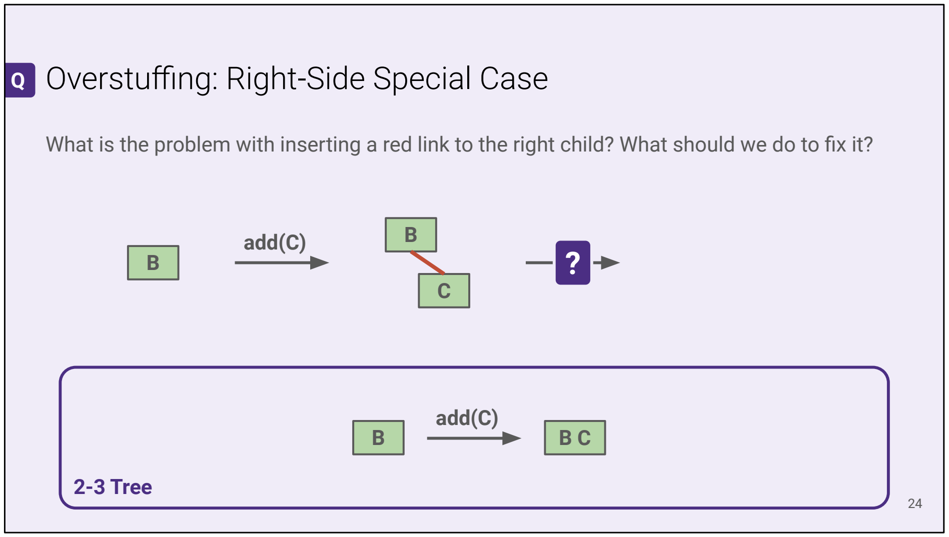


Q Overstuffing: Inserting a New Node

Should we use a red or black link when inserting a new node?

Our 2-3 tree adds the key to the left side since A is smaller than B.

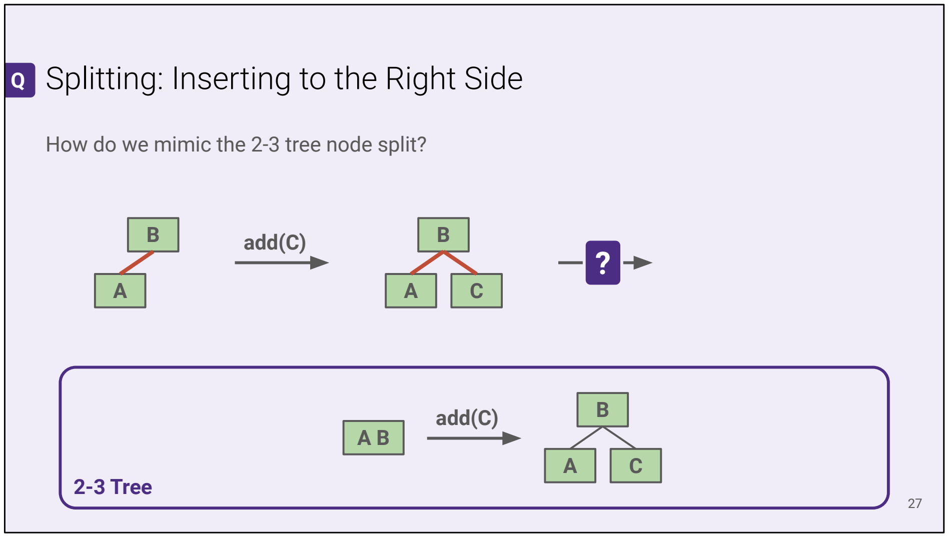
Q1: Should we use a red or black link when inserting a new node?



Q Overstuffing: Right-Side Special Case

What is the problem with inserting a red link to the right child? What should we do to fix it?

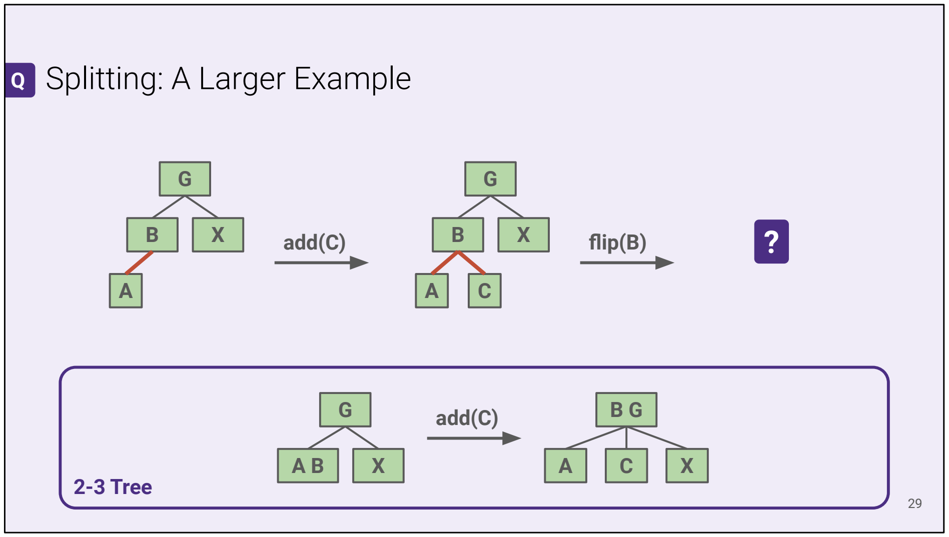
Q1: What is the problem with inserting a red link to the right child? What should we do to fix it?



Q Splitting: Inserting to the Right Side

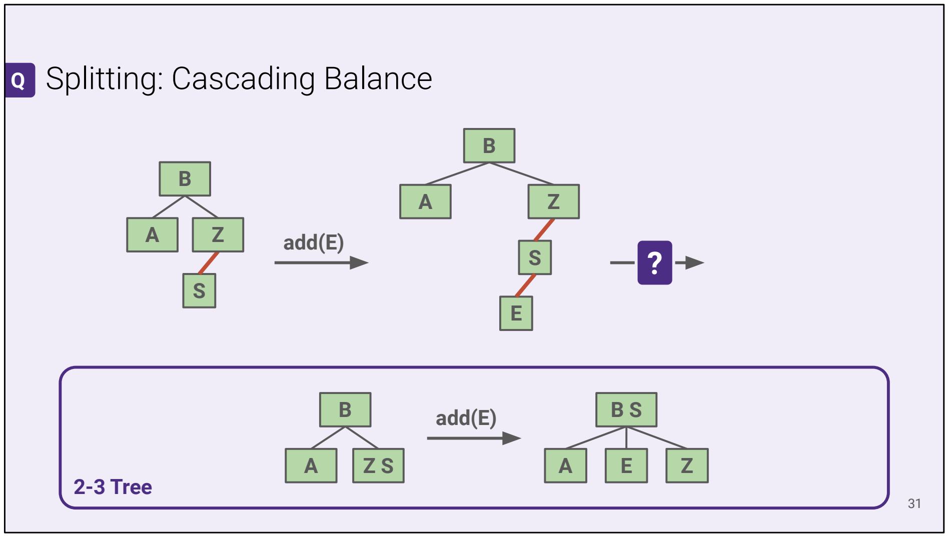
How do we mimic the 2-3 tree node split?

Q1: How do we mimic the 2-3 tree in this case?



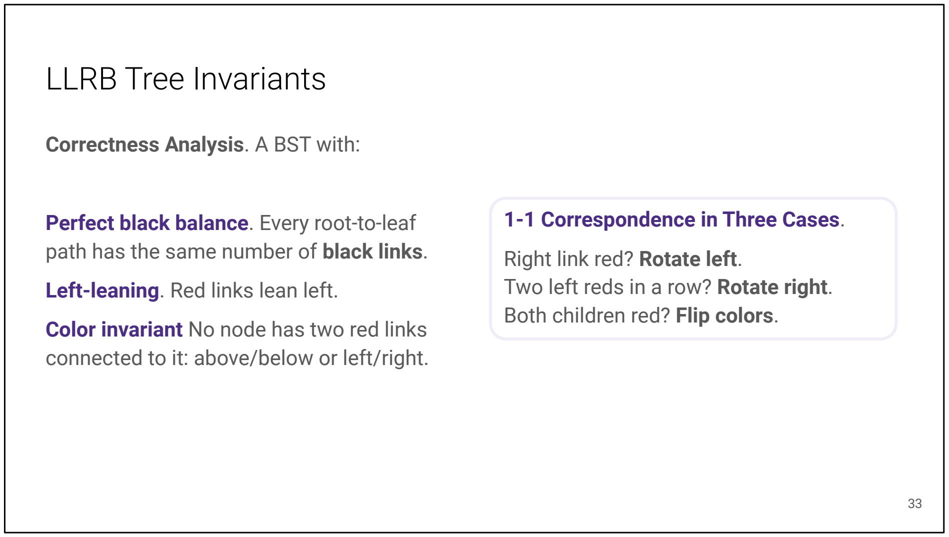
Q Splitting: A Larger Example

Q1: What should the result of flipping B’s links look like, based on the corresponding 2-3 tree?



Q Splitting: Cascading Balance

Q1: What rotation or color flip should we use to coerce this into a better form?



LLRB Tree Invariants

Correctness Analysis. A BST with:

Perfect black balance. Every root-to-leaf path has the same number of **black links**.

Left-leaning. Red links lean left.

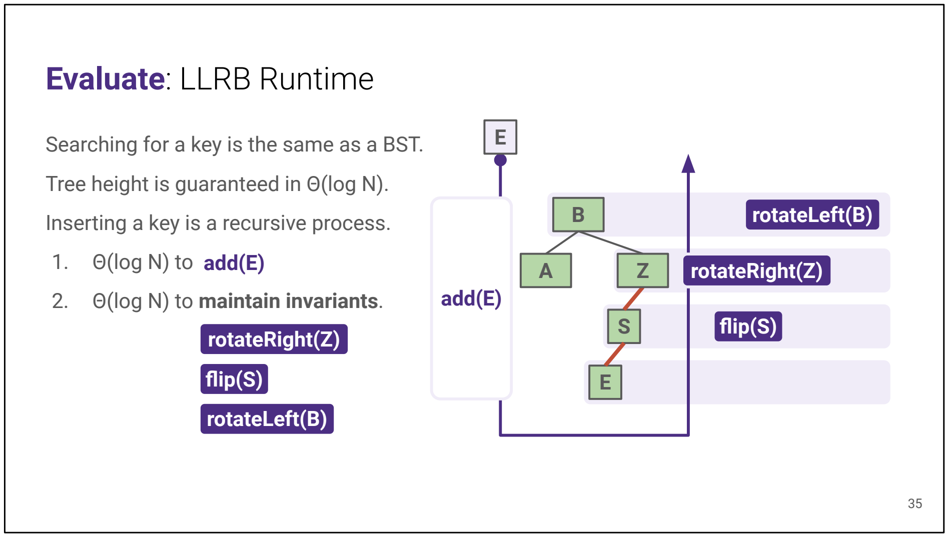
Color invariant No node has two red links connected to it: above/below or left/right.

1-1 Correspondence in Three Cases.

Right link red? **Rotate left**.
Two left reds in a row? **Rotate right**.
Both children red? **Flip colors**.

We now have a working left-leaning red-black tree!

?: How do we know that these three cases are enough to maintain the invariants?



Evaluate: LLRB Runtime

Searching for a key is the same as a BST.

Tree height is guaranteed in Θ(log N).

Inserting a key is a recursive process.

1. Θ(log N) to **add(E)**
2. Θ(log N) to **maintain invariants**.

rotateRight(Z)
flip(S)
rotateLeft(B)

Recall that add is a recursive method. Each row in the diagram is a call to add(E) on a different node in the tree. There's a cost to recurse downwards (to find the right place to add the leaf) and then a cost as we return from each recursive frame and maintain invariants.

?: Why is the runtime to execute add(E) Θ(log N) rather than O(log N)?