Re-balance it through the tri-node operation

Fix the $x, y, z$ in the previous slides

- Let $a, b, c$ be the inorder sequence of $x, y, z$.
- Let $T_0, T_1, T_2, T_3$ be the four sub-trees such that the inorder traversal is $T_0, a, T_1, b, T_2, c, T_3$.
- Change the tree to the following shape.

```
        b
       /|
      / | 
     a  T1  c
    /     |   /
 T0     T2  T3
```
Cases: \( y \) in the middle

Let \( T_0, T_1, T_2, T_3 \) be subtrees.

- \( z, y, x \): full picture \( T_0, z, T_1, y, T_2, x, T_3 \) (in-order traversal).
Cases: $y$ in the middle after balance

Let $T_0, T_1, T_2, T_3$ be subtrees.

- $z, y, x$: full picture $T_0, z, T_1, y, T_2, x, T_3$ (in-order traversal).

Similarly for $x, y, z$ case. This is called **single rotation**!
Correctness

Claims

- Rotation design $\Rightarrow$ binary search tree.
Correctness

Claims

- Rotation design $\Rightarrow$ binary search tree. in-order order
Correctness

Claims

- Rotation design $\Rightarrow$ binary search tree. in-order order
- Tri-node operation rebalance $z$, and keep $y, x$ balanced.
Correctness

Claims

- Rotation design \( \Rightarrow \) binary search tree. in-order order
- Tri-node operation rebalance \( z \), and keep \( y, x \) balanced.
- The new root of the subtree \( b \) has the same height of \( z \) before the insertion operation. Thus, all the remaining nodes are also balanced.
Correctness

Claims

- Rotation design $\Rightarrow$ binary search tree. in-order order
- Tri-node operation rebalance $z$, and keep $y$, $x$ balanced.
- The new root of the subtree $b$ has the same height of $z$ before the insertion operation. Thus, all the remaining nodes are also balanced.
- This suggests that one only need to do the tri-node operation once.
Correctness

Claims

- Rotation design $\Rightarrow$ binary search tree. in-order order
- Tri-node operation rebalance $z$, and keep $y, x$ balanced.
- The new root of the subtree $b$ has the same height of $z$ before the insertion operation. Thus, all the remaining nodes are also balanced.
- This suggests that one only need to do the tri-node operation once.
Correctness: The single-rotation case!

Claims

- $z, y, x$: full picture $T_0, z, T_1, y, T_2, x, T_3$ (in-order traversal).
Correctness: The single-rotation case!

Claims

▸ $z, y, x$: full picture $T_0, z, T_1, y, T_2, x, T_3$ (in-order traversal).
▸ Tri-node operation rebalance $z$, and keep $y, x$ balanced.
▸ The new root of the subtree $b$ has the same height of $z$ before the insertion operation. Thus, all the remaining nodes are also balanced.
Unbalance Nodes: removal!

Analyze possible cases of unbalanced nodes

- Removal’s affect on the height reduces to case 1. Let \( w \) be the removed node. \( w \) must have an external child.
Analyze possible cases of unbalanced nodes

- Removal’s affect on the height reduces to case 1. Let $w$ be the removed node. $w$ must have an external child.
- Remove $w$ could decrease the height of some subtree. However, with at most 1 unbalanced node ($z$). Why?
Unbalance Nodes: removal!

Analyze possible cases of unbalanced nodes

- Removal’s affect on the height reduces to case 1. Let $w$ be the removed node. $w$ must have an external child.
- Remove $w$ could decrease the height of some subtree. However, with at most 1 unbalanced node ($z$). Why?
- Let $y$ be $z$’s child with higher height. Let $x$ be $y$’s child with higher height (could be a tie).
Unbalance Nodes: removal!

Analyze possible cases of unbalanced nodes

- Removal’s affect on the height reduces to case 1. Let $w$ be the removed node. $w$ must have an external child.
- Remove $w$ could decrease the height of some subtree. However, with at most 1 unbalanced node ($z$). Why?
- Let $y$ be $z$’s child with higher height. Let $x$ be $y$’s child with higher height (could be a tie).
- Claim: $x, y$ are not $w$’s ancestor. Why?
Cont’d: Removal

Implementation & Complexity

1 Find the unbalanced \( z \) if any and the \( y, x \).
Cont’d: Removal

Implementation & Complexity

1. Find the unbalanced $z$ if any and the $y, x$. $O(h_1)$.
Cont’d: Removal

Implementation & Complexity

1. Find the unbalanced $z$ if any and the $y, x$. $O(h_1)$.

2. Determine the in-order relationship of $z, y, x$. 

Correctness:
- Tri-node operation rebalance $z$, and keep $y, x$ balanced.
- The height of $b$, however, might decrease by one. Thus need to repeat the procedure until reach the root.
Cont’d: Removal

Implementation & Complexity

1. Find the unbalanced $z$ if any and the $y, x$. $O(h_1)$.
2. Determine the in-order relationship of $z, y, x$. $O(1)$
Cont’d: Removal

Implementation & Complexity

1. Find the unbalanced $z$ if any and the $y, x$. $O(h_1)$.
2. Determine the in-order relationship of $z, y, x$. $O(1)$
3. Perform tri-node operation.
Cont’d: Removal

Implementation & Complexity

1. Find the unbalanced $z$ if any and the $y, x$. $O(h_1)$.
2. Determine the in-order relationship of $z, y, x$. $O(1)$
3. Perform tri-node operation. $O(1)$. 

Correctness

▶ Tri-node operation rebalance $z$, and keep $y, x$ balanced.
▶ The height of $b$, however, might decrease by one. Thus need to repeat the procedure until reach the root.
Cont’d: Removal

Implementation & Complexity

1. Find the unbalanced $z$ if any and the $y, x$. $O(h_1)$.
2. Determine the in-order relationship of $z, y, x$. $O(1)$
3. Perform tri-node operation. $O(1)$.
4. Along the path of the new root $b$ of the subtree to its ancestors, repeat Step 1-3.
Cont’d: Removal

Implementation & Complexity

1. Find the unbalanced $z$ if any and the $y, x$. $O(h_1)$.
2. Determine the in-order relationship of $z, y, x$. $O(1)$.
3. Perform tri-node operation. $O(1)$.
4. Along the path of the new root $b$ of the subtree to its ancestors, repeat Step 1-3.
5. In total, $O(h_1 + h_2 + \cdots) = O(h) = O(\log(n))$. 

Correctness

▶ Tri-node operation rebalance $z$, and keep $y, x$ balanced.

▶ The height of $b$, however, might decrease by one. Thus need to repeat the procedure until reach the root.
Cont’d: Removal

Implementation & Complexity

1. Find the unbalanced $z$ if any and the $y, x$. $O(h_1)$.
2. Determine the in-order relationship of $z, y, x$. $O(1)$
3. Perform tri-node operation. $O(1)$.
4. Along the path of the new root $b$ of the subtree to its ancestors, repeat Step 1-3.
5. In total, $O(h_1 + h_2 + \cdots) = O(h) = O(\log(n))$.

Correctness

▶ Tri-node operation rebalance $z$, and keep $y, x$ balanced.
Cont’d: Removal

Implementation & Complexity

1. Find the unbalanced \(z\) if any and the \(y, x\). \(O(h_1)\).
2. Determine the in-order relationship of \(z, y, x\). \(O(1)\)
3. Perform tri-node operation. \(O(1)\).
4. Along the path of the new root \(b\) of the subtree to its ancestors, repeat Step 1-3.
5. In total, \(O(h_1 + h_2 + \cdots) = O(h) = O(\log(n))\).

Correctness

- Tri-node operation rebalance \(z\), and keep \(y, x\) balanced.
- The height of \(b\), however, might decrease by one. Thus need to repeat the procedure until reach the root.