Lecture 02/15/16

Lecturer: Xiaodi Wu

February 15th, 2016
AVL tree: Update

Updates like BST
Except, we might break the height-balance property. Need additional effort to re-balance it!
AVL tree: Update

Updates like BST

Except, we might break the height-balance property. Need additional effort to re-balance it!
AVL tree: Update

Updates like BST

Except, we might break the height-balance property. Need additional effort to re-balance it!
Unbalance Nodes: insertion!

Analyze possible cases of unbalanced nodes

- Let $w$ be the inserted node. Follow the path from $w$ to the root (update height). How?
Unbalance Nodes: insertion!

Analyze possible cases of unbalanced nodes

- Let $w$ be the inserted node. Follow the path from $w$ to the root (update height). How?
- Find the first unbalanced node $z$. Let $y$ be $z$’s child with higher height. Let $x$ be $y$’s child with higher height.
Unbalance Nodes: insertion!

Analyze possible cases of unbalanced nodes

- Let $w$ be the inserted node. Follow the path from $w$ to the root (update height). How?
- Find the first unbalanced node $z$. Let $y$ be $z$'s child with higher height. Let $x$ be $y$'s child with higher height.
- Claim: $x, y, z$ are $w$'s ancestor. ($x$ could be $w$ itself.) Why?
Unbalance Nodes: insertion!

Analyze possible cases of unbalanced nodes

- Let $w$ be the inserted node. Follow the path from $w$ to the root (update height). How?
- Find the first unbalanced node $z$. Let $y$ be $z$’s child with higher height. Let $x$ be $y$’s child with higher height.
- Claim: $x, y, z$ are $w$’s ancestor. ($x$ could be $w$ itself.) Why?
- Understand how heights are updated. Insertion can only increase the height of some subtree.
Unbalance Nodes: insertion!

Analyze possible cases of unbalanced nodes

- Let \( w \) be the inserted node. Follow the path from \( w \) to the root (update height). How?
- Find the first unbalanced node \( z \). Let \( y \) be \( z \)'s child with higher height. Let \( x \) be \( y \)'s child with higher height.
- Claim: \( x, y, z \) are \( w \)'s ancestor. (\( x \) could be \( w \) itself.) Why?
- Understand how heights are updated. Insertion can only increase the height of some subtree.
- Could be more unbalanced nodes. \( z \) is the first. Why?
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 $x, y, z$. 
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 $T_0, a, T_1, b, T_2, c, T_3$. 
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.
Tri-node operation based on $x, y, z$

Properties of $x, y, z$

- $z$: the first unbalanced node. Let $y$ be $z$’s child with higher height. Let $x$ be $y$’s child with higher height (could be a tie).
Tri-node operation based on $x, y, z$

Properties of $x, y, z$

- $z$: the first unbalanced node. Let $y$ be $z$’s child with higher height. Let $x$ be $y$’s child with higher height (could be a tie).
- Then there are four possible relative relations of $x, y, z$ in the in-order traversal. Why not 6?
Tri-node operation based on $x, y, z$

Properties of $x, y, z$

- $z$: the first unbalanced node. Let $y$ be $z$'s child with higher height. Let $x$ be $y$'s child with higher height (could be a tie).
- Then there are four possible relative relations of $x, y, z$ in the in-order traversal. Why not 6? $z$ cannot be in the middle.
Tri-node operation based on $x, y, z$

Properties of $x, y, z$

- $z$: the first unbalanced node. Let $y$ be $z$’s child with higher height. Let $x$ be $y$’s child with higher height (could be a tie).
- Then there are four possible relative relations of $x, y, z$ in the in-order traversal. Why not 6? $z$ cannot be in the middle.
  - $z, y, x$.
  - $x, y, z$.
  - $z, x, y$.
  - $y, x, z$. 
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

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).

```
                      z
                     /   \
                    /     \
                   /       \
                  /         \
                 /           \
                /             \
               /               \
              /                 \
             /                   \
            /                     \
           /                       \
          /                         \
         /                           \
        /                             \
       /                               \
      /                                 \
     /                                   \
    /                                     \
   /                                       \
  /                                         \
 /                                           \
/                                             \

T_0  z  T_1  y  T_2  x  T_3
```
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).
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**!
Cases: $x$ in the middle

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

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

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

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

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

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

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

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

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

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

Similarly for $y$, $x$, $z$ case. This is called **double rotation**!
Find the unbalanced $z$ if any and the $y, x$. 

$O(h) = \Theta(\log(n))$. 

$O(1)$. 

Perform tri-node operation. 

$O(1)$. 

Only need to perform once. Update the heights. 

In total, $O(\log(n))$. 

Implementation & Complexity
Find the unbalanced $z$ if any and the $y, x$. $O(h) = O(\log(n))$. 
Implementation & Complexity

- Find the unbalanced $z$ if any and the $y, x$. $O(h) = O(\log(n))$.
- Determine the in-order relationship of $z, y, x$. 
Implementation & Complexity

- Find the unbalanced $z$ if any and the $y, x$. $O(h) = O(\log(n))$.
- Determine the in-order relationship of $z, y, x$. $O(1)$
Find the unbalanced $z$ if any and the $y, x$. $O(h) = O(\log(n))$.

Determine the in-order relationship of $z, y, x$. $O(1)$

Perform tri-node operation.
Implementation & Complexity

- Find the unbalanced \( z \) if any and the \( y, x \). \( O(h) = O(\log(n)) \).
- Determine the in-order relationship of \( z, y, x \). \( O(1) \)
- Perform tri-node operation. \( O(1) \).
Implementation & Complexity

- Find the unbalanced $z$ if any and the $y, x$. $O(h) = O(\log(n))$.
- Determine the in-order relationship of $z, y, x$. $O(1)$
- Perform tri-node operation. $O(1)$.
- Only need to perform once. Update the heights.
Implementation & Complexity

- Find the unbalanced $z$ if any and the $y, x$. $O(h) = O(\log(n))$.
- Determine the in-order relationship of $z, y, x$. $O(1)$
- Perform tri-node operation. $O(1)$.
- Only need to perform once. Update the heights. $O(1)$.
Implementation & Complexity

- Find the unbalanced $z$ if any and the $y, x$. $O(h) = O(\log(n))$.
- Determine the in-order relationship of $z, y, x$. $O(1)$
- Perform tri-node operation. $O(1)$.
- Only need to perform once. Update the heights. $O(1)$.
- In total, $O(\log(n))$. 