

ACOL106: Data Structures and Algorithms

Ragesh Jaiswal, IIT Delhi

Data Structures: **Balanced** Binary Search Trees

Data Structures

Balanced Binary Search Trees → AVL Trees

- AVL Tree: An AVL tree is a binary search tree that satisfies the following property:
Height balance property: For every internal node of the tree, the heights of its children differ by at most 1.
- Claim: The height of any AVL tree storing n nodes is $O(\log n)$.

Data Structures

Balanced Binary Search Trees → AVL Trees

- AVL Tree: An AVL tree is a binary search tree that satisfies the following property:
Height balance property: For every internal node of the tree, the heights of its children differ by at most 1.
- Claim: The height of any AVL tree storing n nodes is $O(\log n)$.
 - Let $n(h)$ denote the minimum number of internal nodes in an AVL tree with height h .
 - Try writing a recurrence relation for $n(h)$ and solving it to get a lower bound.

Data Structures

Balanced Binary Search Trees → AVL Trees

- AVL Tree: An AVL tree is a binary search tree that satisfies the following property:
Height balance property: For every internal node of the tree, the heights of its children differ by at most 1.
- Question: How do we perform $\text{get}(k)$ operation on an AVL tree?

Data Structures

Balanced Binary Search Trees → AVL Trees

- AVL Tree: An AVL tree is a binary search tree that satisfies the following property:
Height balance property: For every internal node of the tree, the heights of its children differ by at most 1.
- Question: How do we perform $\text{get}(k)$ operation on an AVL tree?
The same as BST

Data Structures

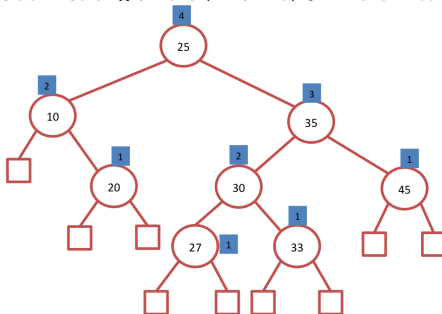
Balanced Binary Search Trees → AVL Trees

- AVL Tree: An AVL tree is a binary search tree that satisfies the following property:
Height balance property: For every internal node of the tree, the heights of its children differ by at most 1.
- Question: How do we perform $\text{put}(k, v)$ operation on an AVL tree?

Data Structures

Balanced Binary Search Trees → AVL Trees

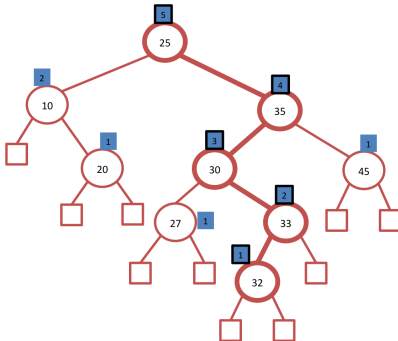
- AVL Tree: An AVL tree is a binary search tree that satisfies the following property:
Height balance property: For every internal node of the tree, the heights of its children differ by at most 1.
- Question: How do we perform $\text{put}(k, v)$ operation on an AVL tree?
 - Same as in BST. However, you also have to make sure that after insertion, the height balance property is maintained.
 - Consider inserting an entry with key 32 in the Tree below.



Data Structures

Balanced Binary Search Trees → AVL Trees

- AVL Tree: An AVL tree is a binary search tree that satisfies the following property:
Height balance property: For every internal node of the tree, the heights of its children differ by at most 1.
- Question: How do we perform $\text{put}(k, v)$ operation on an AVL tree?
 - Same as in BST. However, you also have to make sure that after insertion, the height balance property is maintained.
 - Consider inserting an entry with key 32 in the Tree below.



Data Structures

Balanced Binary Search Trees → AVL Trees

- Question: How do we perform $\text{put}(k, v)$ operation on an AVL tree?

Algorithm

// p denotes the node that is inserted.

$\text{BalanceAfterPut}(\text{Node } p)$

- While going up from p , let z denote the first node for which the height balance property is not satisfied.
- Let y be the child of z with greater height.
- Let x be the child of y with greater height.
- Perform a tri-node restructuring w.r.t. nodes x, y, z .

Data Structures

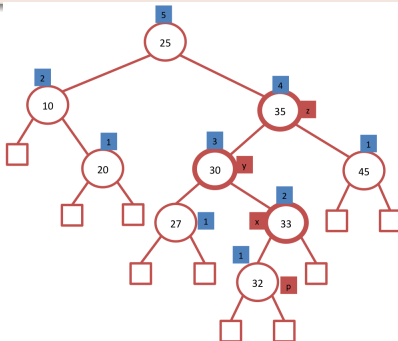
Balanced Binary Search Trees → AVL Trees

Algorithm

// p denotes the node that is inserted.

BalanceAfterPut(Node p)

- While going up from p , let z denote the first node for which the height balance property is not satisfied.
- Let y be the child of z with greater height.
- Let x be the child of y with greater height.
- Perform a tri-node restructuring w.r.t. nodes x, y, z .



Data Structures

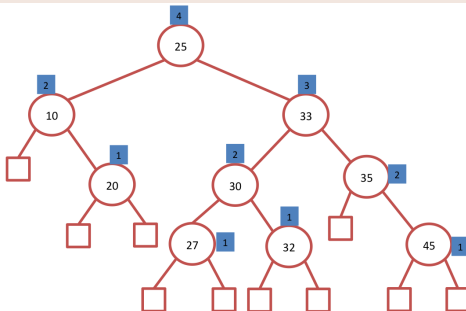
Balanced Binary Search Trees → AVL Trees

Algorithm (pseudocode)

//p denotes the node that is inserted.

BalanceAfterPut(Node p)

- While going up from p , let z denote the first node for which the height balance property is not satisfied.
- Let y be the child of z with greater height.
- Let x be the child of y with greater height.
- Perform a tri-node restructuring w.r.t. nodes x, y, z .



Data Structures

Balanced Binary Search Trees $\rightarrow$ AVL Trees

Algorithm

// p denotes the node that is inserted.

BalanceAfterPut(Node p)

- While going up from p , let z denote the first node for which the height balance property is not satisfied.
- Let y be the child of z with greater height.
- Let x be the child of y with greater height.
- Perform a tri-node restructuring w.r.t. nodes x, y, z .

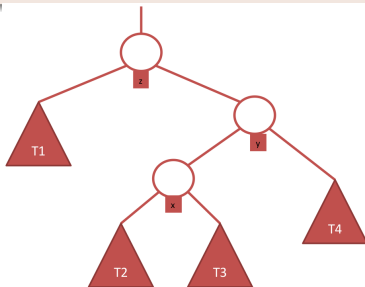


Figure: Suppose the insertion happens in the right sub-tree of node labeled x .

Data Structures

Balanced Binary Search Trees $\rightarrow$ AVL Trees

Algorithm

// p denotes the node that is inserted.

BalanceAfterPut(Node p)

- While going up from p , let z denote the first node for which the height balance property is not satisfied.
- Let y be the child of z with greater height.
- Let x be the child of y with greater height.
- Perform a tri-node restructuring w.r.t. nodes x, y, z .

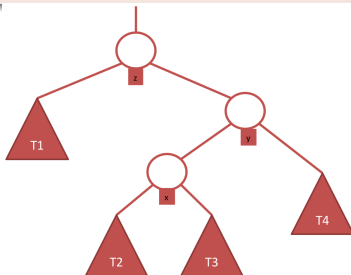


Figure: Suppose the insertion happens in $T3$ and x, y, z are as defined in the pseudocode.

Data Structures

Balanced Binary Search Trees → AVL Trees

Algorithm

// p denotes the node that is inserted.

BalanceAfterPut(Node p)

- While going up from p , let z denote the first node for which the height balance property is not satisfied.
- Let y be the child of z with greater height.
- Let x be the child of y with greater height.
- Perform a tri-node restructuring w.r.t. nodes x, y, z .

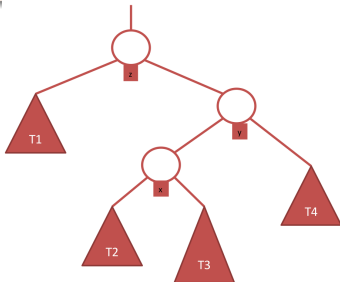


Figure: Suppose the insertion happens in $T3$ and x, y, z are as defined in the pseudocode.

Data Structures

Balanced Binary Search Trees $\rightarrow$ AVL Trees

Algorithm

// p denotes the node that is inserted.

BalanceAfterPut(Node p)

- While going up from p , let z denote the first node for which the height balance property is not satisfied.
- Let y be the child of z with greater height.
- Let x be the child of y with greater height.
- Perform a tri-node restructuring w.r.t. nodes x, y, z .

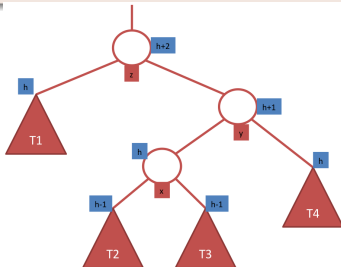


Figure: Suppose the insertion happens in $T3$ and x, y, z are as defined in the pseudocode. For some h the height of the nodes before insertion will be as shown above.

Data Structures

Balanced Binary Search Trees $\rightarrow$ AVL Trees

Algorithm

// p denotes the node that is inserted.

BalanceAfterPut(Node p)

- While going up from p , let z denote the first node for which the height balance property is not satisfied.
- Let y be the child of z with greater height.
- Let x be the child of y with greater height.
- Perform a tri-node restructuring w.r.t. nodes x, y, z .

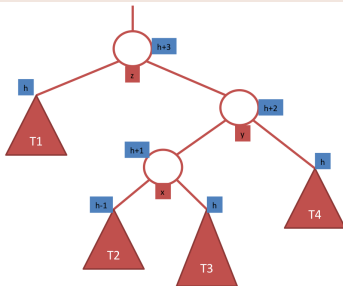


Figure: The height of the nodes after inserting the new node are as shown above.

Data Structures

Balanced Binary Search Trees $\rightarrow$ AVL Trees

Algorithm

// p denotes the node that is inserted.

BalanceAfterPut(Node p)

- While going up from p , let z denote the first node for which the height balance property is not satisfied.
- Let y be the child of z with greater height.
- Let x be the child of y with greater height.
- Perform a tri-node restructuring w.r.t. nodes x, y, z .

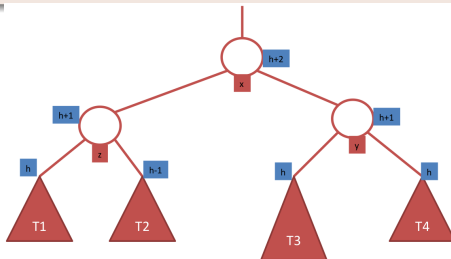


Figure: The height of the nodes after inserting and performing tri-node restructuring.

End