CS 225 Spring 2019 :: TA Lecture Notes 2/27 AVL

By Wenjie

Four BST Rotation

- L, R , LR, RL
- With all rotations, BST properties are remain preserved
- Each rotation has constant time complexity
- We know that tree imbalance can be caused by a stick and by an elbow. We fix sticks in previous lecture, and in this part we we learn to fix elbow by transforming the elbow into a stick.
- Goal : create a tree with height diff small as possible
- Then we will have AVL Trees aka balanced BST

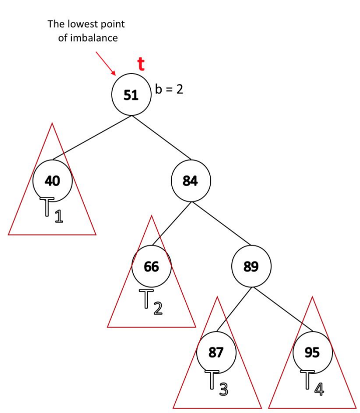
AVL Tree consideration

- Four rotations
    - Simple rotations: stick
    - Complex rotations: elbow
- Maintain height of tree
- Detect imbalance of tree

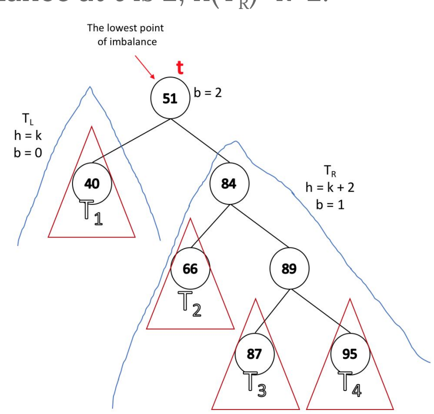
Rotations

- **<u>Theorem 1</u>: If an insertion occurred in subtrees t_3 or t_4, and an imbalance was detected at t, then a LEFT rotation about t restores the balance of the tree. We gauge this by noting the balance factor of t→right is 1.**

1. We identified the lowest point of imbalance, which means b=2.



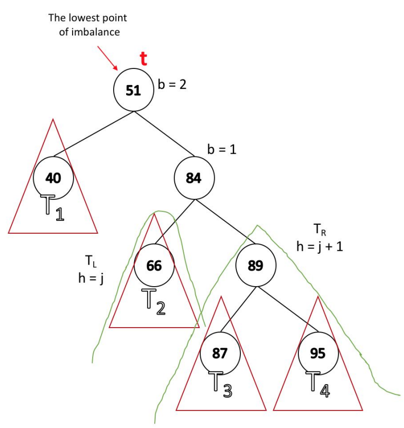
2. Let $h(T_L)=k$. Since we inserted at $\mathbf{t_3}$ or $\mathbf{t_4}$ and balance at **t** is 2, $h(T_R)=k+2$.



CS 225 Spring 2019 :: TA Lecture Notes 2/27 AVL

By Wenjie

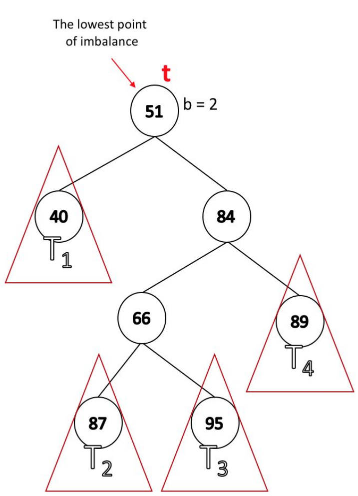
3. If b(t→right) = 2, then we would have detected it as the lowest point of imbalance. Therefore, b(t→right) < 2. If b(t→right) = -1, it would mean that the tree is leaning to the left, but we said we are inserting to the right. Therefore, b(t→right) > -1. Since we have b(t) = 2, we know that b(t→right) cannot be 0 because it is not balanced. Therefore, b(t→right) = 1.

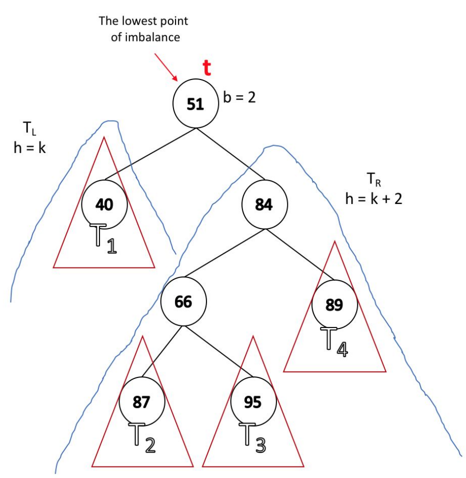


- **<u>Theorem 2</u>: If an insertion occurred in subtrees t_2 or t_3, and an imbalance was detected at t, then a RIGHT-LEFT rotation about t restores the balance of the tree.**
  - **We gauge this by noting the balance factor of t→right is -1.**

1. We identified the lowest point of imbalance, which means b=2.

2. Let $h(T_L)$=k. Since we inserted at $\mathbf{t_2}$ or $\mathbf{t_3}$ and balance at **t** is 2, $h(T_R)$=k+2.

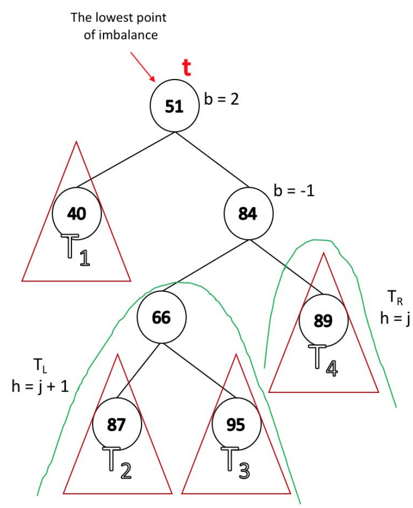




CS 225 Spring 2019 :: TA Lecture Notes
2/27 AVL

By Wenjie

3. Similarly to the third step in the Theorem 1, the balance of t→left cannot be 2 because t→left is not the lowest point of imbalance. Furthermore, t→left cannot be 1 because the insertion is done on the left side of the t→left. Therefore, the only value that shows imbalance is b(t→left) = -1.

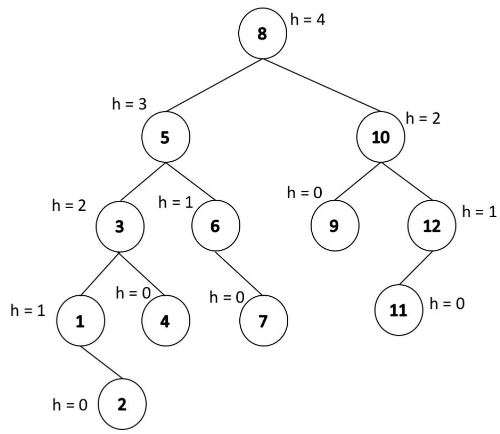


- Both of these theorems involve two steps:
  - Identify the point of imbalance.
  - Apply rules to determine what kind of rotation to use.

- The two theorems we introduced offer solutions for only two rotations, but we learned that there are four rotations. Luckily, the other two rotations (R and LR) are just mirrors of the L and RL.
  - In fact, if we know one rotation we know all four: For example, if we know L, then R is a mirror of L; RL and LR are just combinations of L and R.

CS 225 Spring 2019 :: TA Lecture Notes
2/27 AVL

By Wenjie



Insertion of AVL

- How do we maintain the tree height?
- Use BST insert
- As we recurse back:
    - Check for imbalance.
    - Correct it (do rotations).
    - Update height.

AVLTree.cpp

```
AVLTree<T>::_insert(const T & x, treeNode<T> * & t )
     Base case: if t == NULL then insert;

     Case 1: x < t→key   // we are going to the left subtree
          _insert(x, t→left)  // recursive call on the left
subtree
          if balance == -2    // detecting imbalance point
               if leftBalance == -1 then rotate to the right
;
               else rotate left-right;

     Case 2: x > t→key  // we are going to the right subtree
          _insert(x, t→right) // recursive call on the right
subtree
          if balance == 2    // detecting imbalance point
               if rightBalance == 1 then rotate to the left;
               else rotate right-left;
```

```
    update height;
}
```