CS 313
- Kangmei Yang
AVL Tree

- It’s a binary search Tree

- Height-Balance Property:
  - For every node n in tree, the heights of n’s children differ by at most 1.
Sample AVLT

A

B

C

D
self-balancing

- if add or remove, result in violating AVL property
  - one of the ancestor is not balanced
  - make a rotation to balance
Unbalance from Add

- add 1

- Not in tree
- Add 1 to the left subtree of 4
- One of it’s ancestor might be unbalanced
- node 7 is not balanced
Rotate to balance

- How do we rotate to balance the tree?
- There’re 4 cases.
  - First 2 cases, the left subtree’s height is much bigger than the right subtree’s height
    - 1. The left child of this unbalance node, it’s left subtree is heavier
    - 2. The left child of this unbalance node, it’s right subtree is heavier

Case 1

Case 2
Rotate for First Case

Node A is Unbalance
It’s left subtree start at B is heavier than it’s right subtree

For left child Node B, it’s left subtree is heavier than it’s right subtree. (same side as A)
- Node A perform a single rotate with it’s left child B
- Node A becomes right child of Node B, Node B becomes parent of Node A
For left child Node B, it’s left subtree is heavier than it’s right subtree. (same side as A)
- Node A perform a single rotate with it’s left child B
- Node A becomes right child of Node B, Node B becomes parent of Node A
Rotate for Second Case

Node A is Unbalance
It’s left subtree start at B is heavier than it’s right subtree.

For left child Node B, it’s right subtree is heavier than it’s left subtree.
(different side as A)
- Node A perform a double rotate with it’s left child B
  - First Node B perform single rotate with right child C
  - Then Node A perform single rotate with new left child C
- Node A becomes right child of Node C, Node B becomes left child of Node C, Node C becomes parent of Node A and B
Double Rotate with Left Child Child

For left child Node B, it’s right subtree is heavier than it’s left subtree. (different side as A)
- Node A perform a double rotate with it’s left child B
- Node A becomes right child of Node C, Node B becomes left child of Node C, Node C becomes parent of Node A and B
Double Rotate with Left Child

For left child Node B, it’s right subtree is heavier than it’s left subtree. (different side as A)
- Node A perform a double rotate with it’s left child B
  - First Node B perform single rotate with right child C
  - Then Node A perform single rotate with new left child C
Rotate to balance

- How do we rotate to balance the tree?
- There’re 4 cases.
  - Last 2 cases, the right subtree’s height is much bigger than the left subtree’s height
    - 3. The right child of this unbalance node, it’s right subtree is heavier
    - 4. The right child of this unbalance node, it’s left subtree is heavier

Case 3

Case 4
Rotate for Third Case

Node A is Unbalance
It’s right subtree start at B is heavier than it’s left subtree

For right child Node B, it’s right subtree is heavier than it’s left subtree. (same side as A)
- Node A perform a single rotate with it’s right child B
- Node A becomes left child of Node B, Node B becomes parent of Node A
Single Rotate with Right Child

For right child Node B, it’s right subtree is heavier than it’s left subtree. (same side as A)
- Node A perform a single rotate with it’s right child B
- Node A becomes left child of Node B, Node B becomes parent of Node A
Rotate for Fourth Case

Node A is Unbalance
It’s right subtree start at B is heavier than it’s left subtree

For right child Node B, it’s left subtree is heavier than it’s right subtree. (different side as A)
- Node A perform a double rotate with it’s right child B
  - First Node B perform single rotate with left child C
  - Then Node A perform single rotate with new right child C
- Node A becomes left child of Node C, Node B becomes right child of Node C, Node C becomes parent of Node A and B
For left child Node B, it’s right subtree is heavier than it’s left subtree. (different side as A)
- Node A perform a double rotate with it’s right child B
- Node A becomes left child of Node C, Node B becomes right child of Node C, Node C becomes parent of Node A and B
self-balancing

- If the unbalanced node is on the “inside” of the subtree, perform double rotation
- If the unbalanced node is on the “outside” of the subtree, perform single rotation