AVL Tree

Adelson-Velskii and Landis

AVL Tree

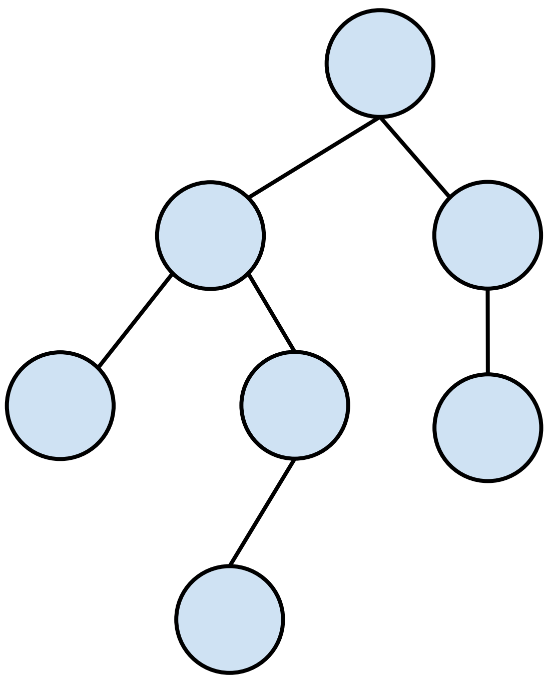
- BST
- Balanced to guarantee depth of O(log N)
- Balance condition

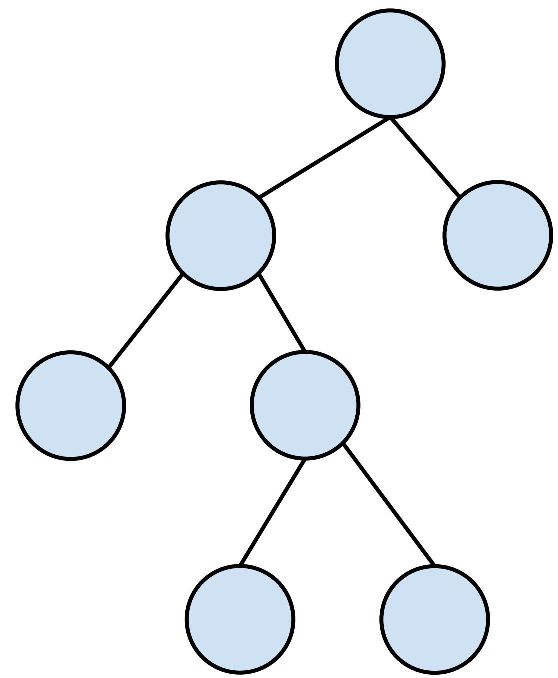
AVL Tree: Balance

- Height
  - The height of an empty tree is -1

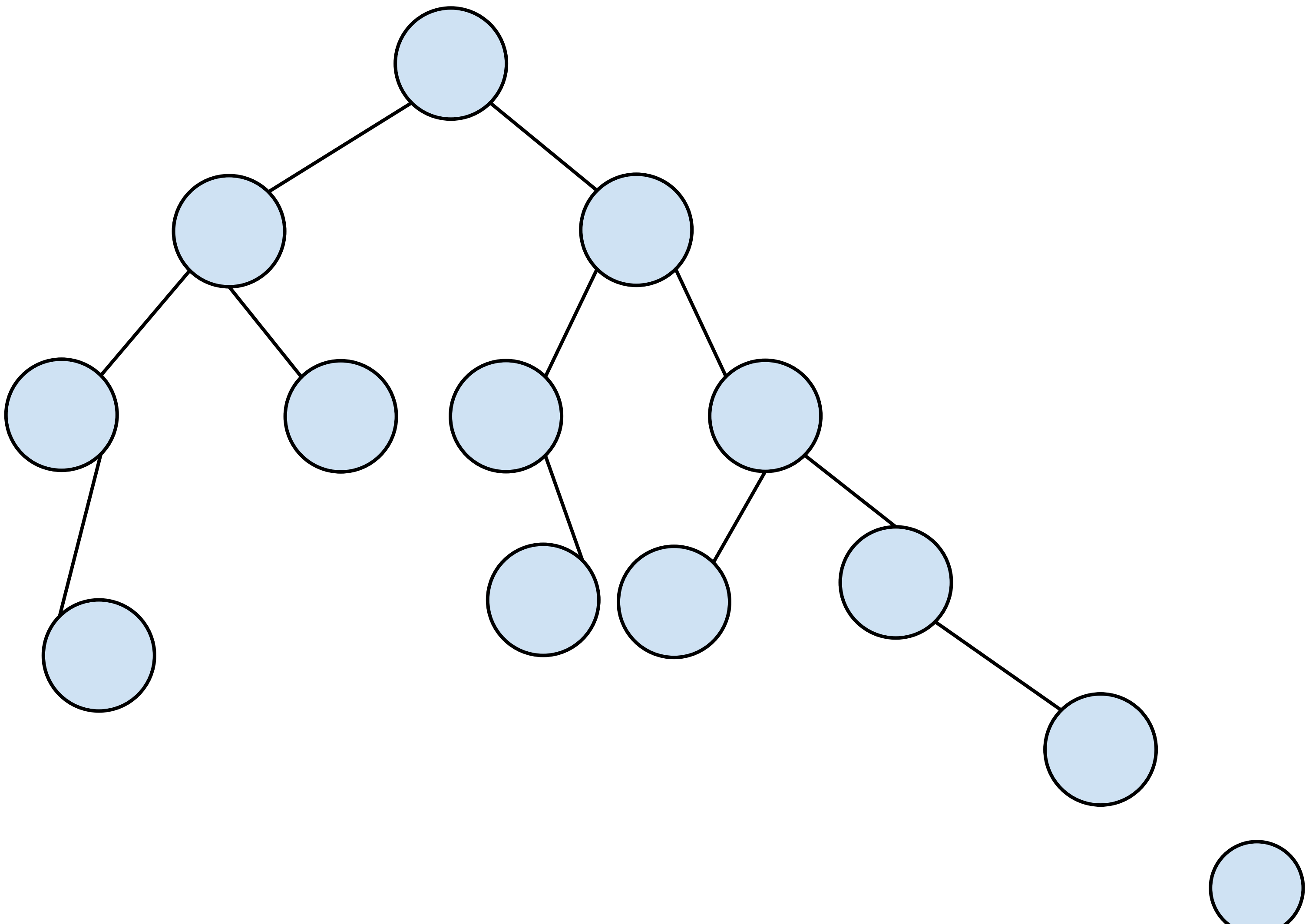
- For every node in the tree
  - The height of the left and right tree can differ by at most 1

AVL Tree: Balance





AVL Tree: Insert



AVL Tree: Insert

- Easy Cases:
  - Insert into left tree
  - Insert into right tree

- Tree is balance and will remain balanced

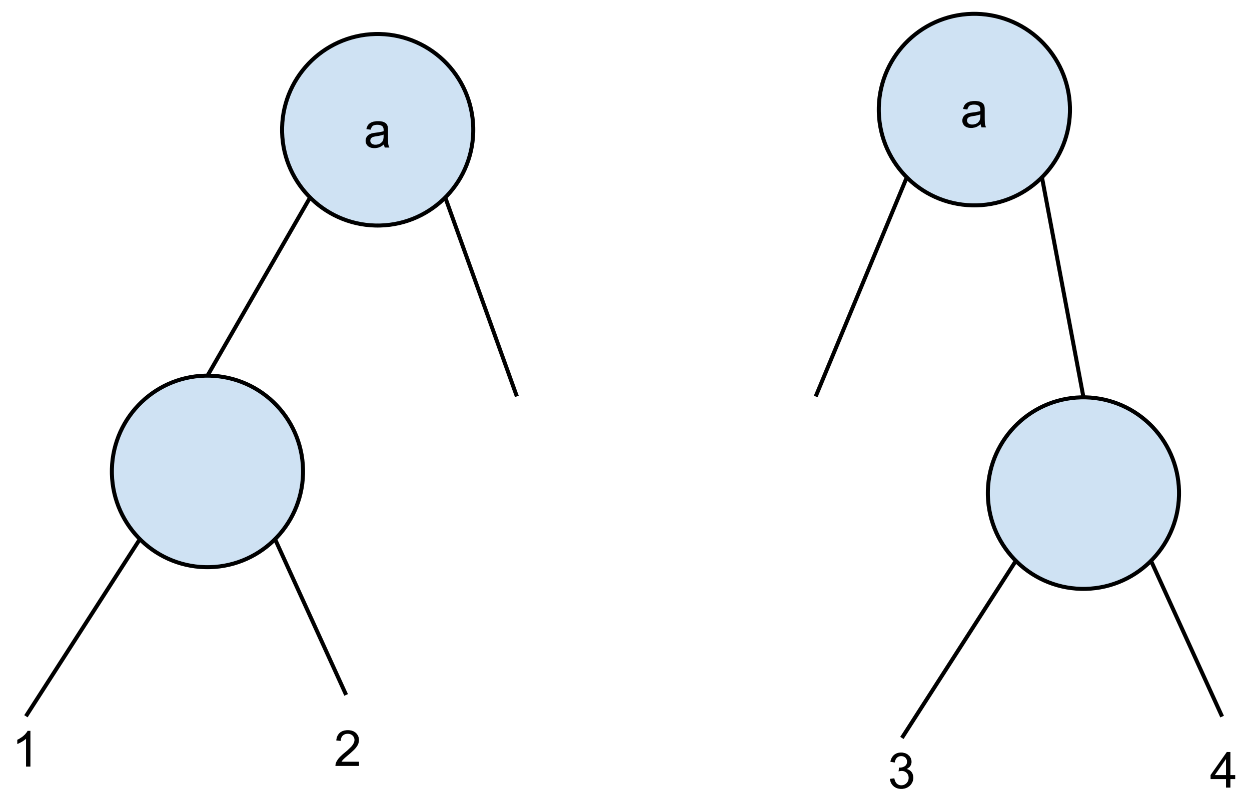
AVL Tree: Insert

- Hard Cases
  1. Insert into the left subtree of the left child
  2. Insert into the right subtree of the left child
  3. Insert into the left subtree of the right child
  4. Insert into the right subtree of the right child

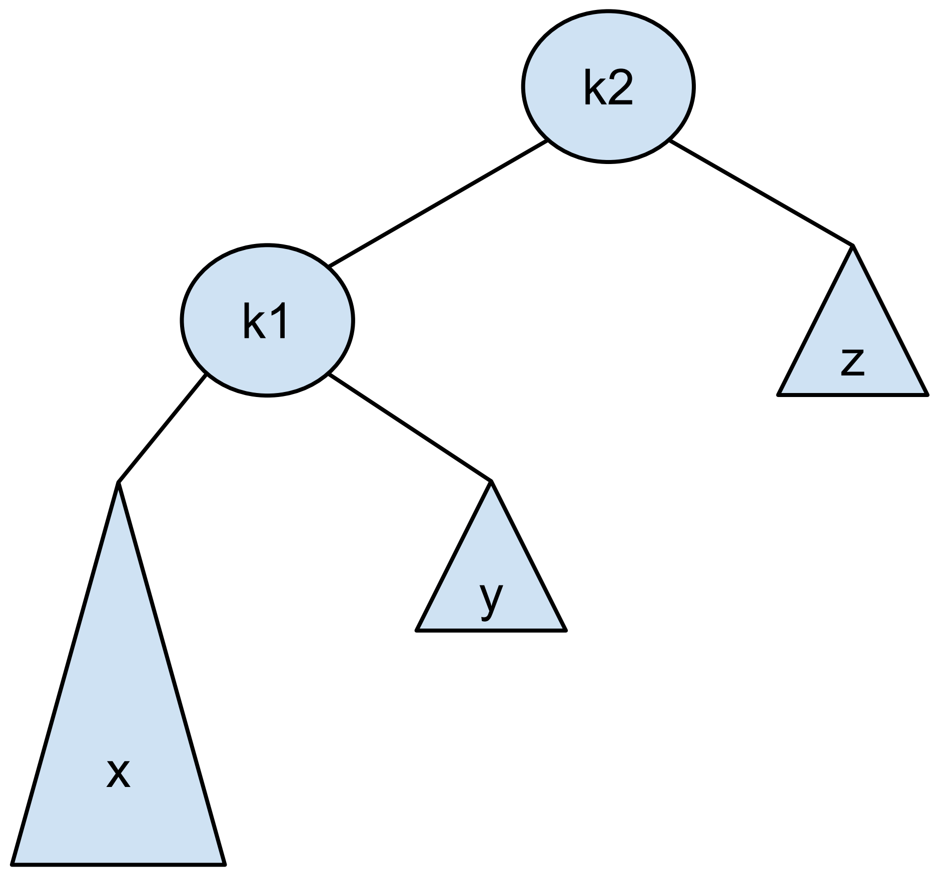
- 1 & 4, 2 & 3 are mirror images of each other

- Tree becomes unbalanced (dun dun dunnnnn)

AVL Tree: Insert

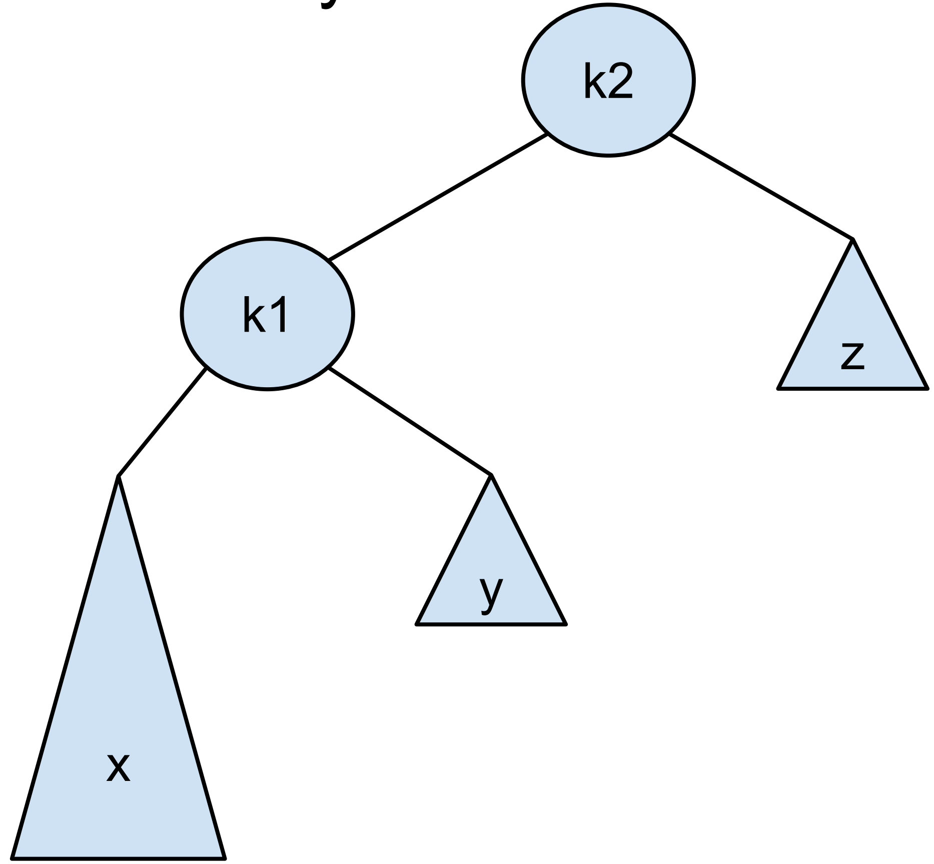


AVL Tree: Rotation Case 1



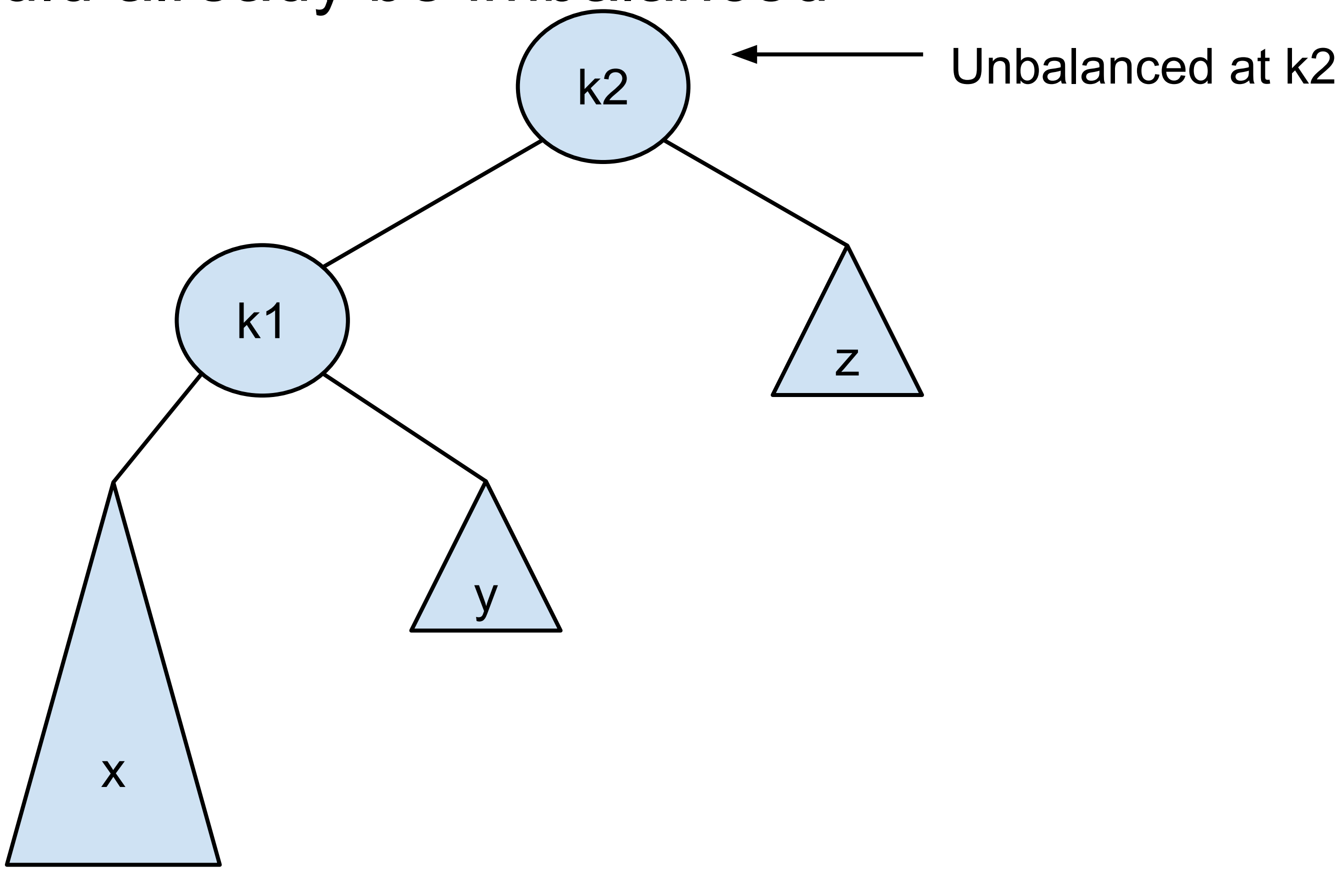
AVL Tree: Rotation Case 1

- This is the only situation that is possible
  - Before insertion x was the same size as y or else the tree would already be imbalanced



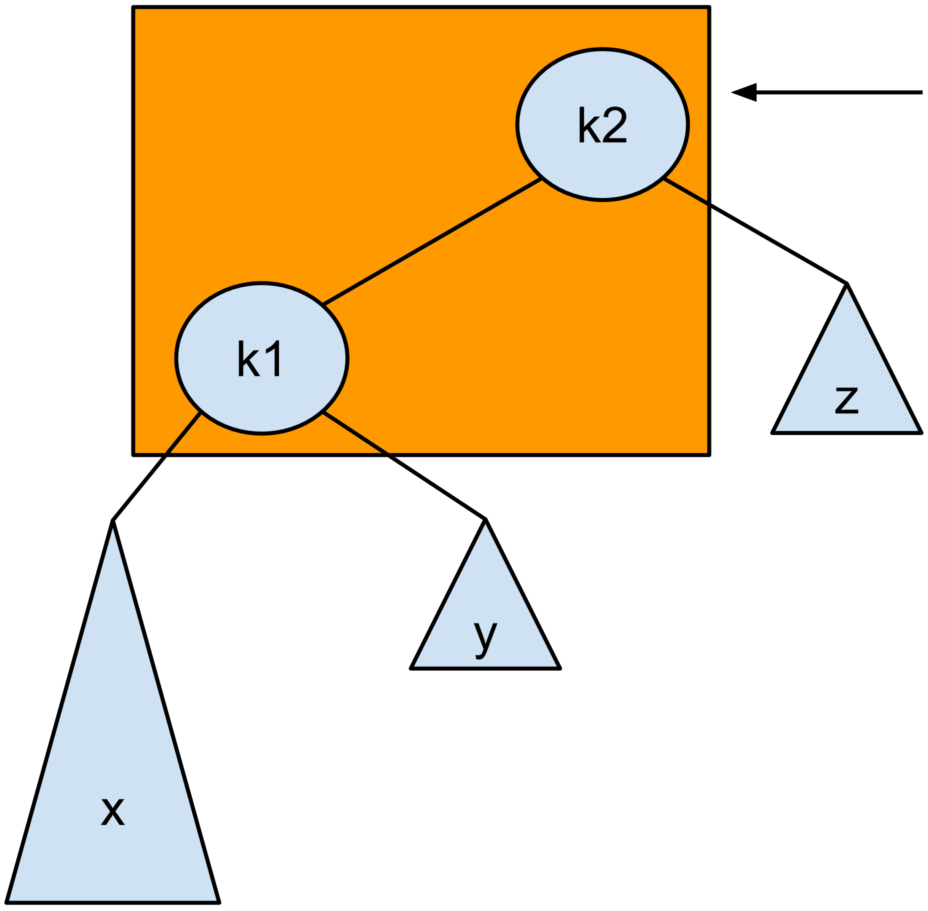
AVL Tree: Rotation Case 1

- This is the only situation that is possible
  - Before insertion x was the same size as y or else the tree would already be imbalanced



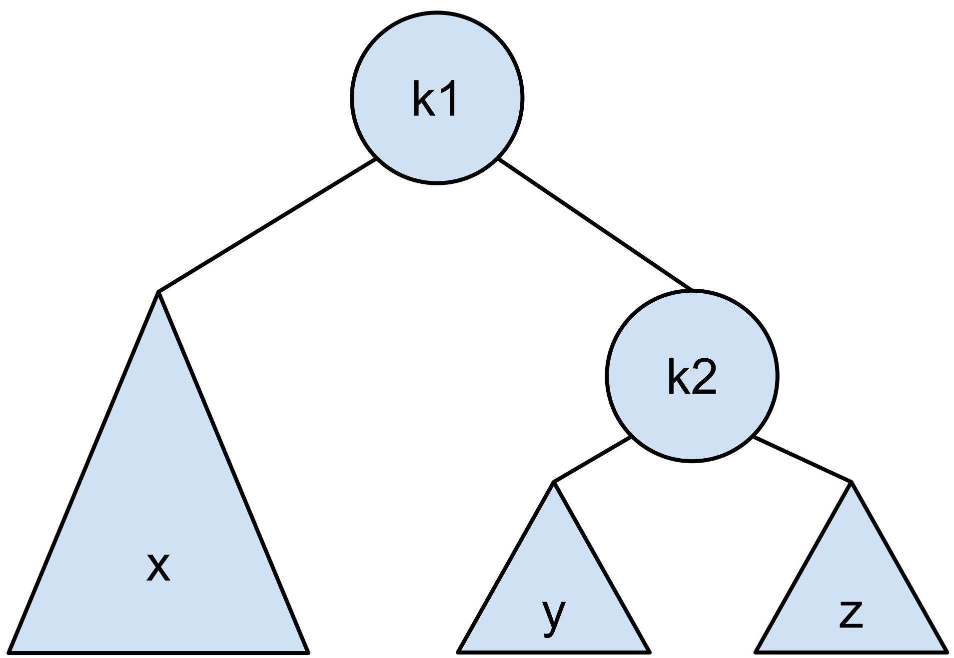
AVL Tree: Rotation Case 1

- Do a single rotation of k2 and k1



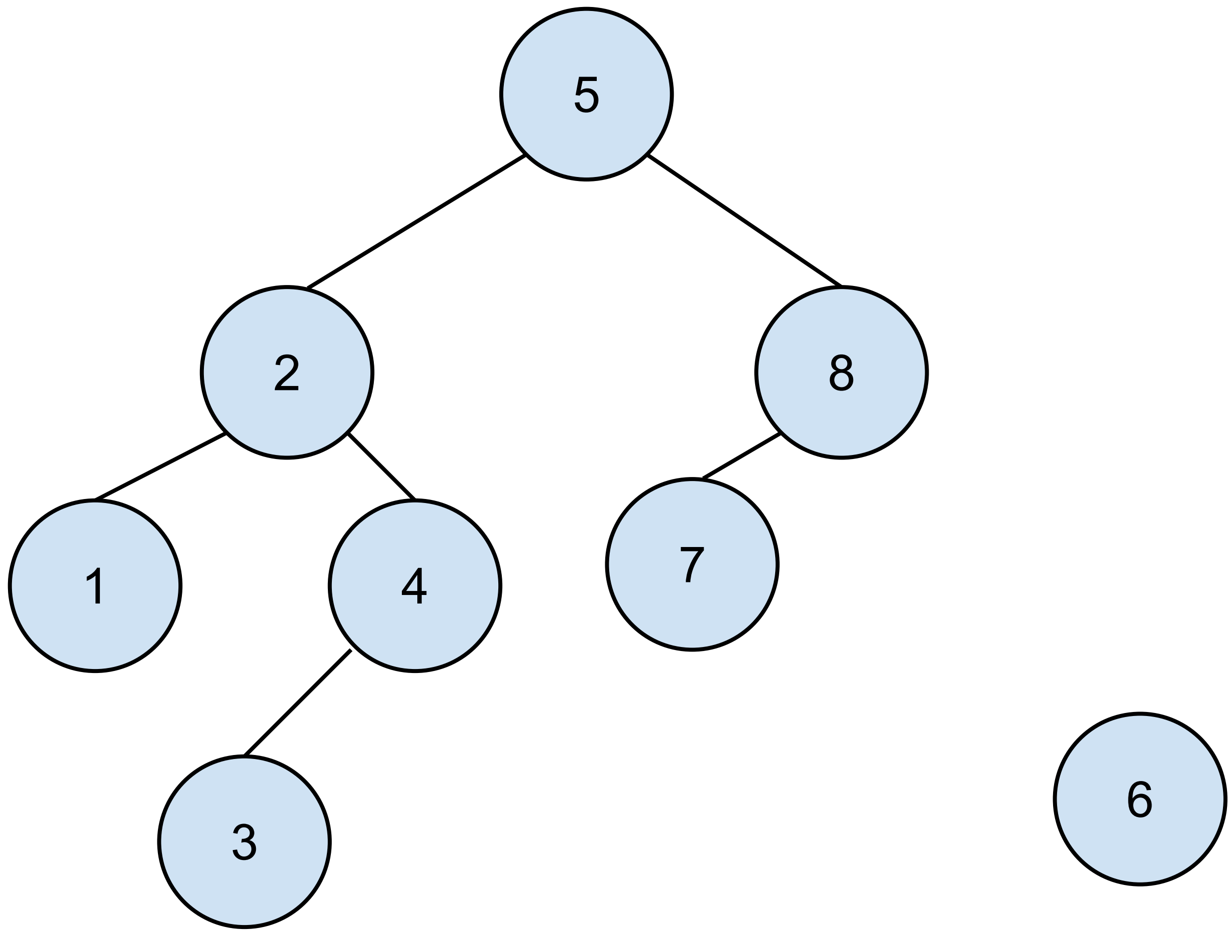
AVL Tree: Rotation Case 1

- Make k1 the new root
  - adjust the other nodes



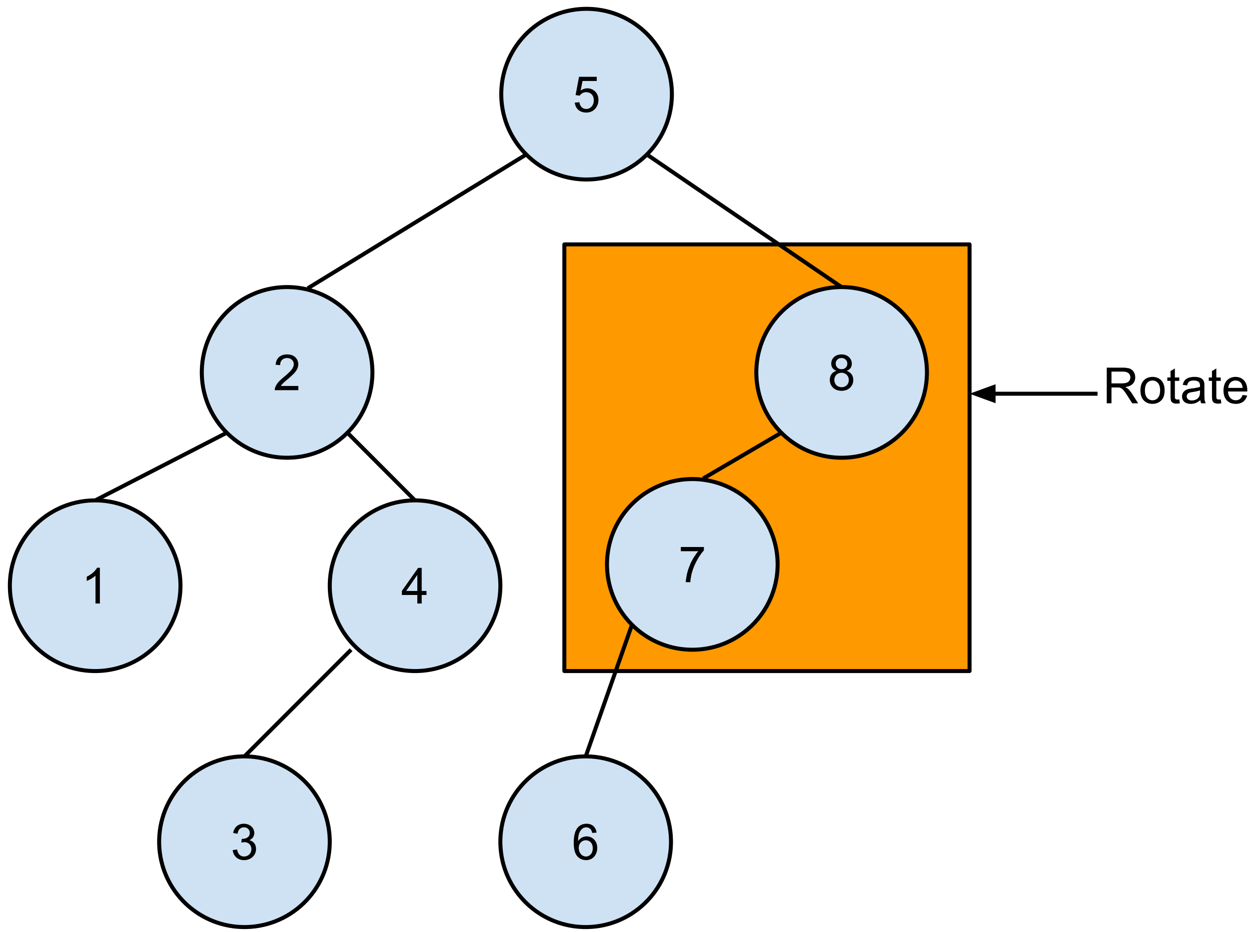
AVL Tree: Rotation Case 1

- Real example



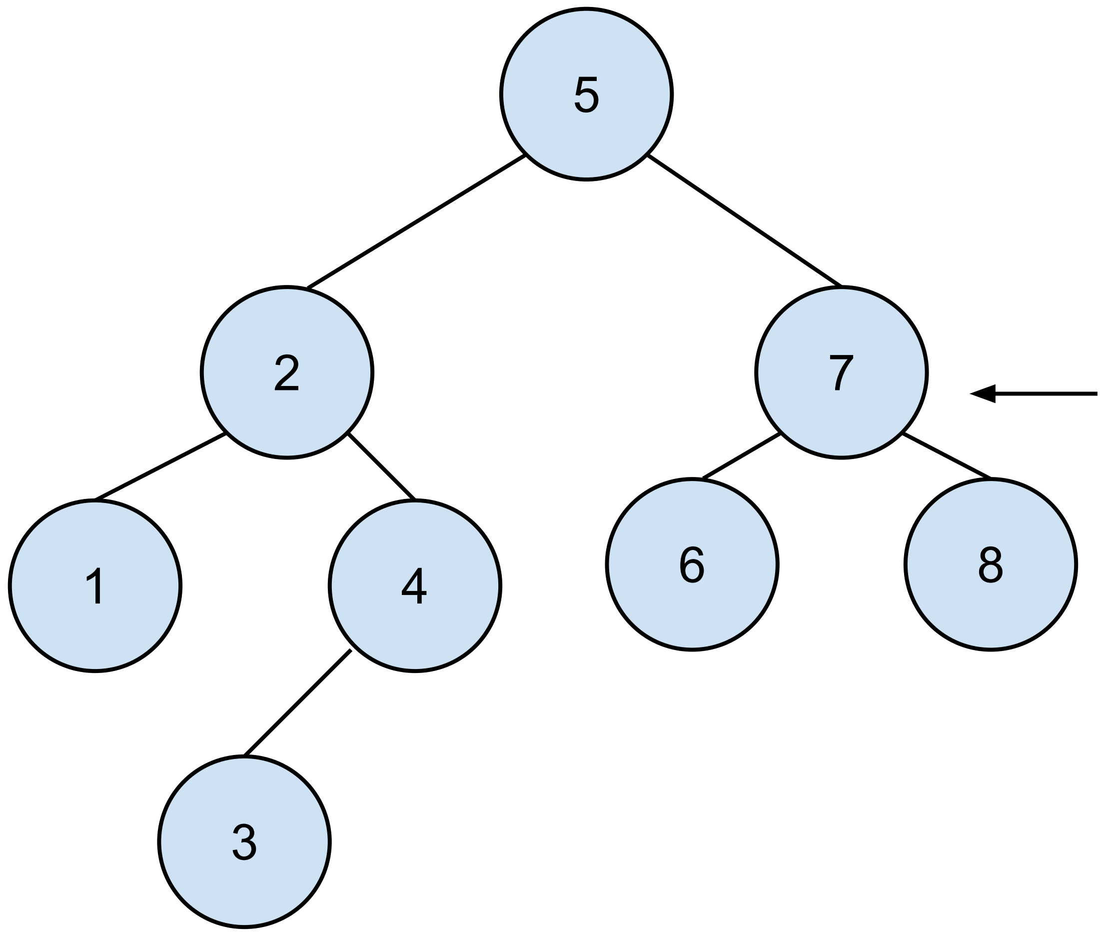
AVL Tree: Rotation Case 1

- Real example



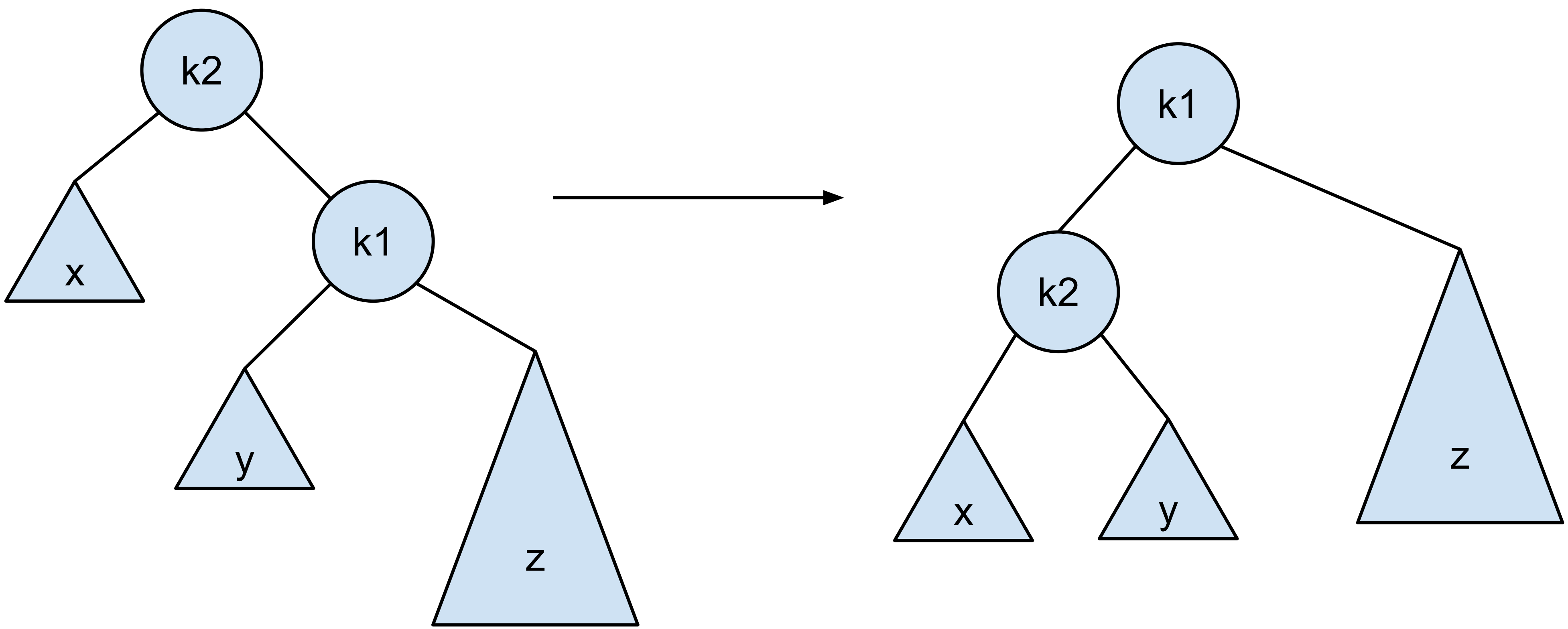
AVL Tree: Rotation Case 1

- Real example



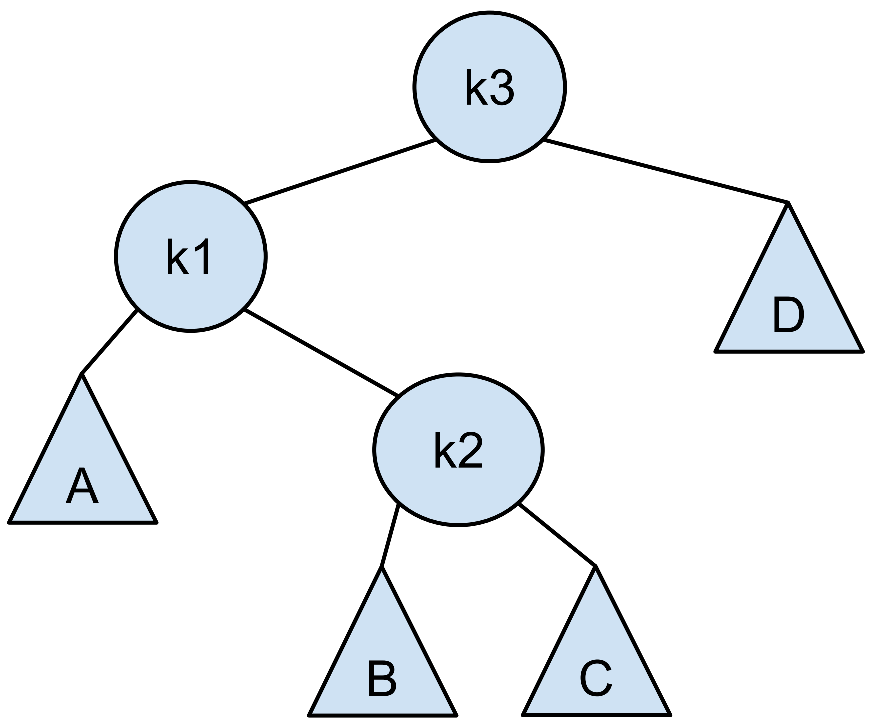
AVL Tree: Rotation Case 4

- Just the reverse of case 1



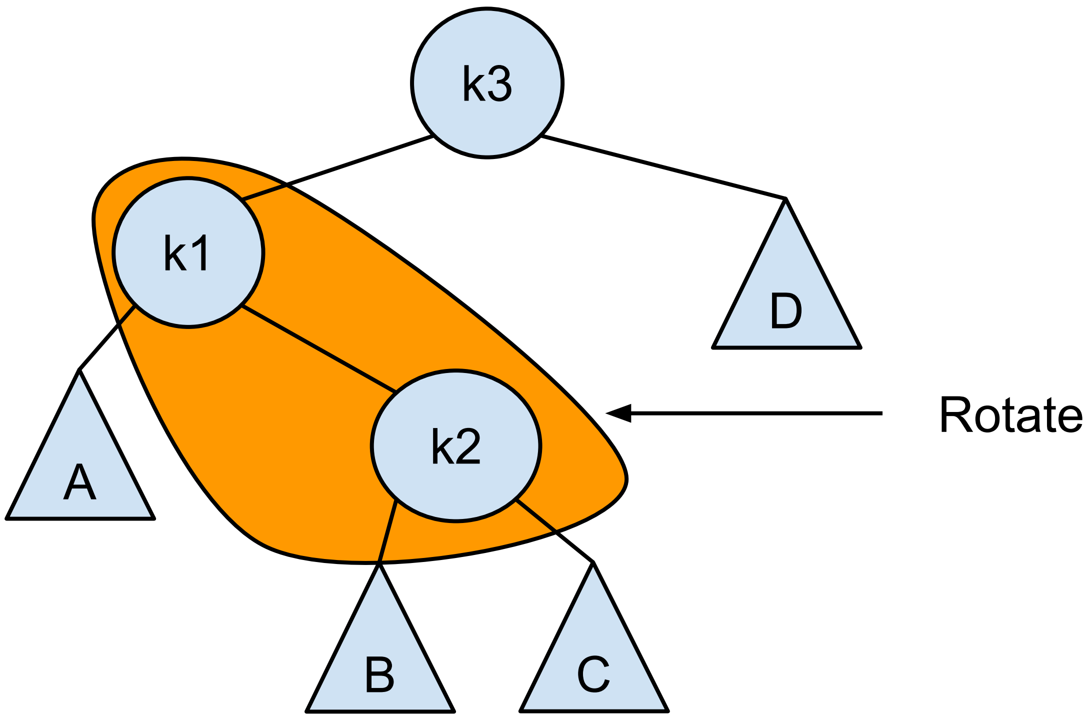
AVL Tree: Double Rotation

- A single rotation will not cover it.
- Who can be the new root?



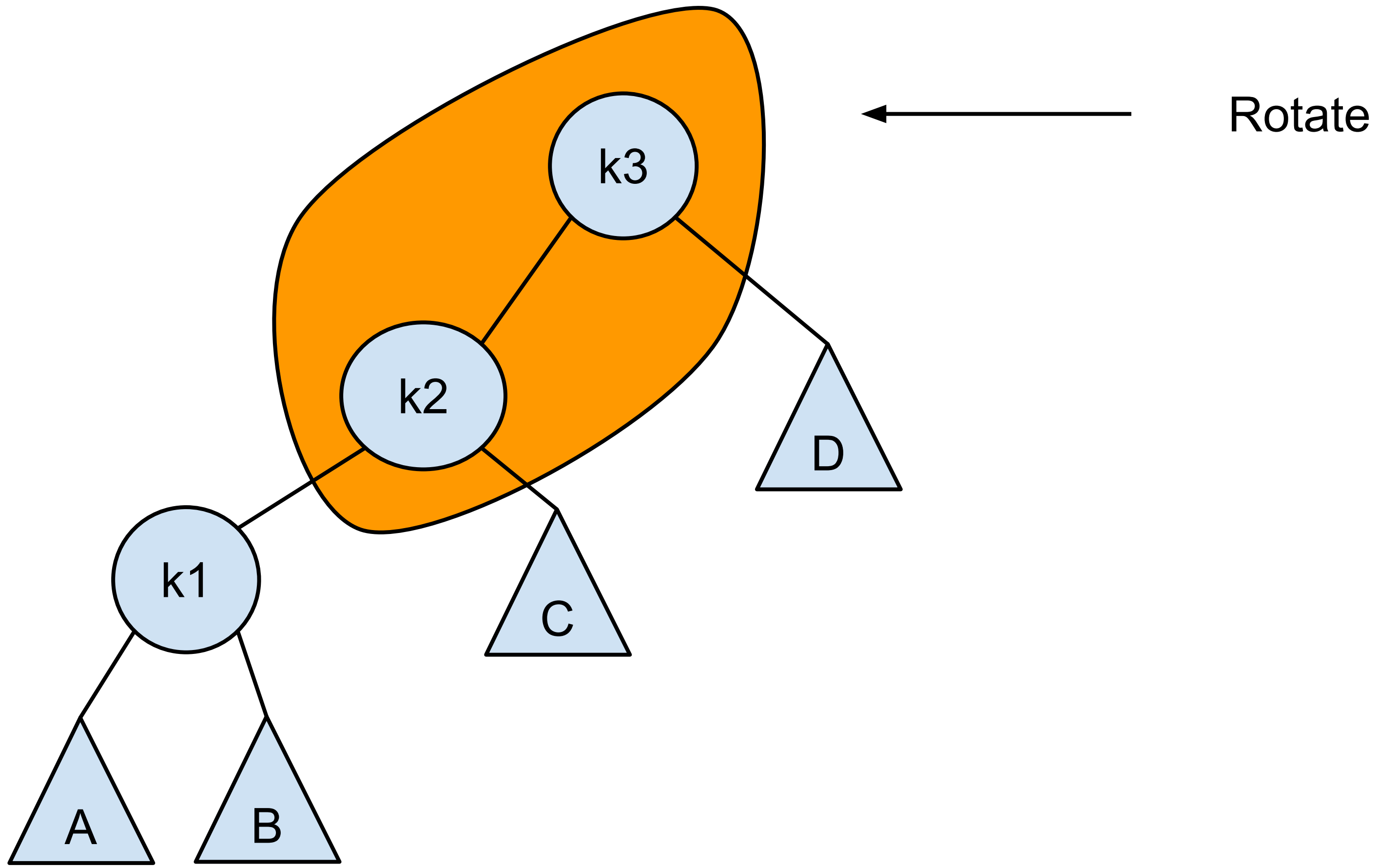
AVL Tree: Double Rotation

- Rotate k2 and k1

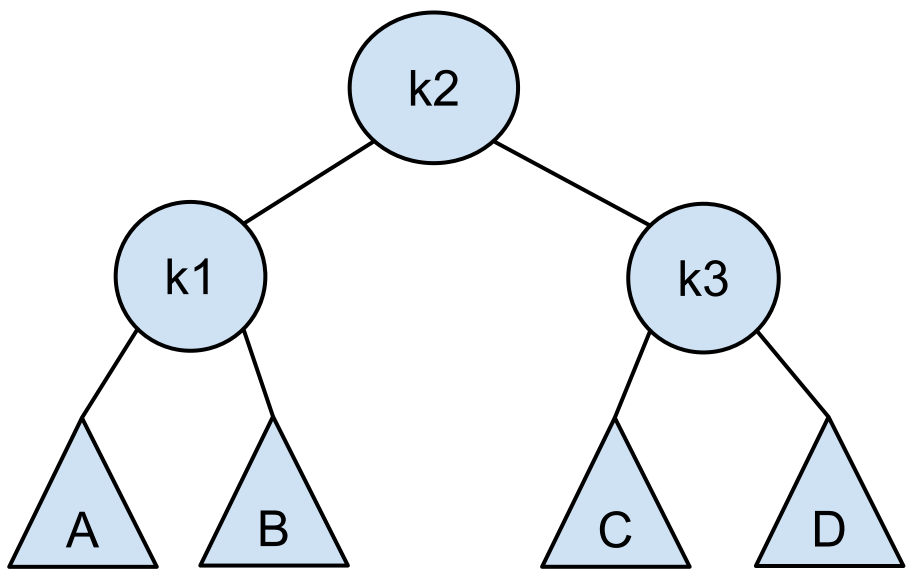


AVL Tree: Double Rotation

- Rotate k2 and k3



AVL Tree: Double Rotation

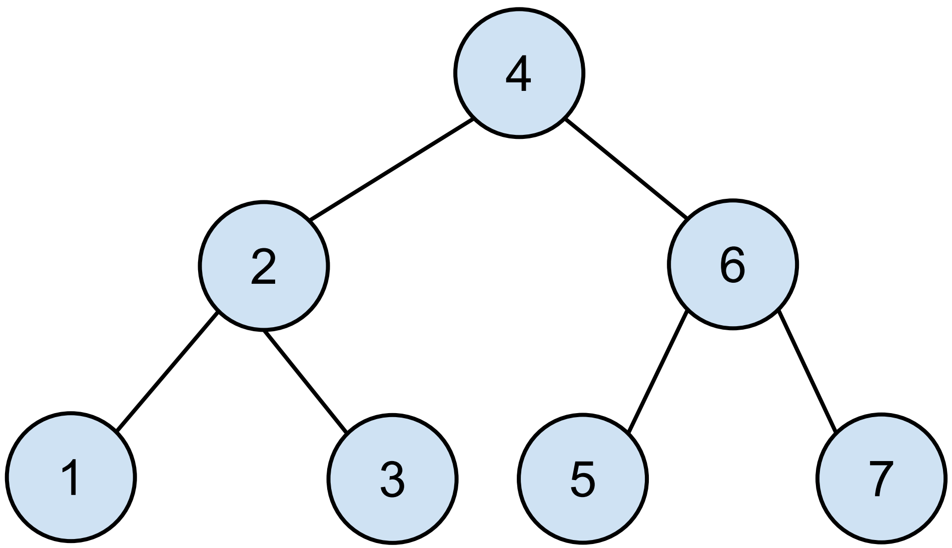


AVL Tree: Rotation

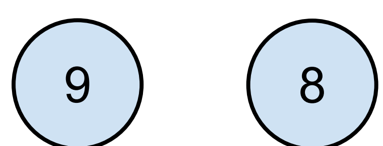
- The double rotation case 3 also works, but in reverse

- There are many ways of looking at these problems
- The solution is basic

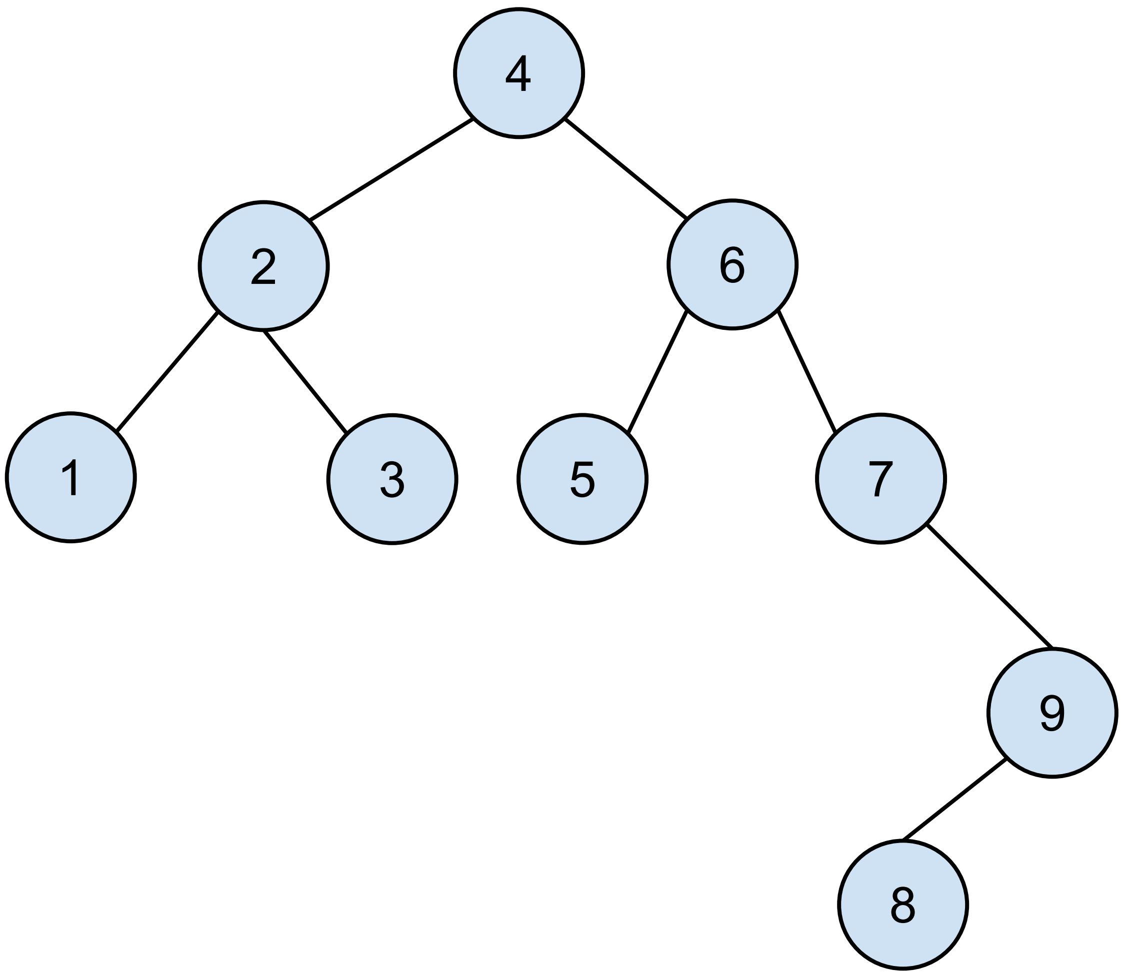
AVL Tree: Insert



Insert:

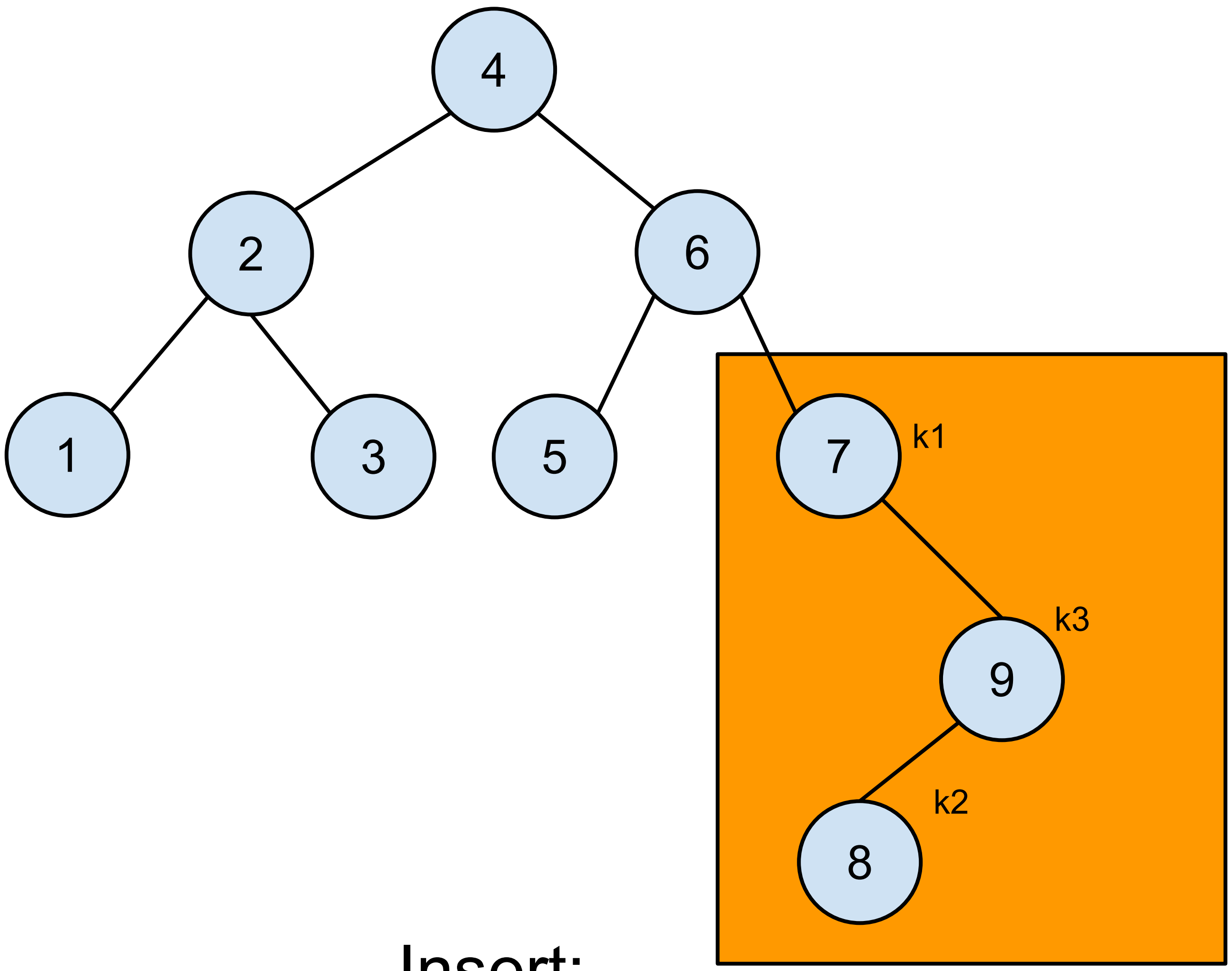


AVL Tree: Insert



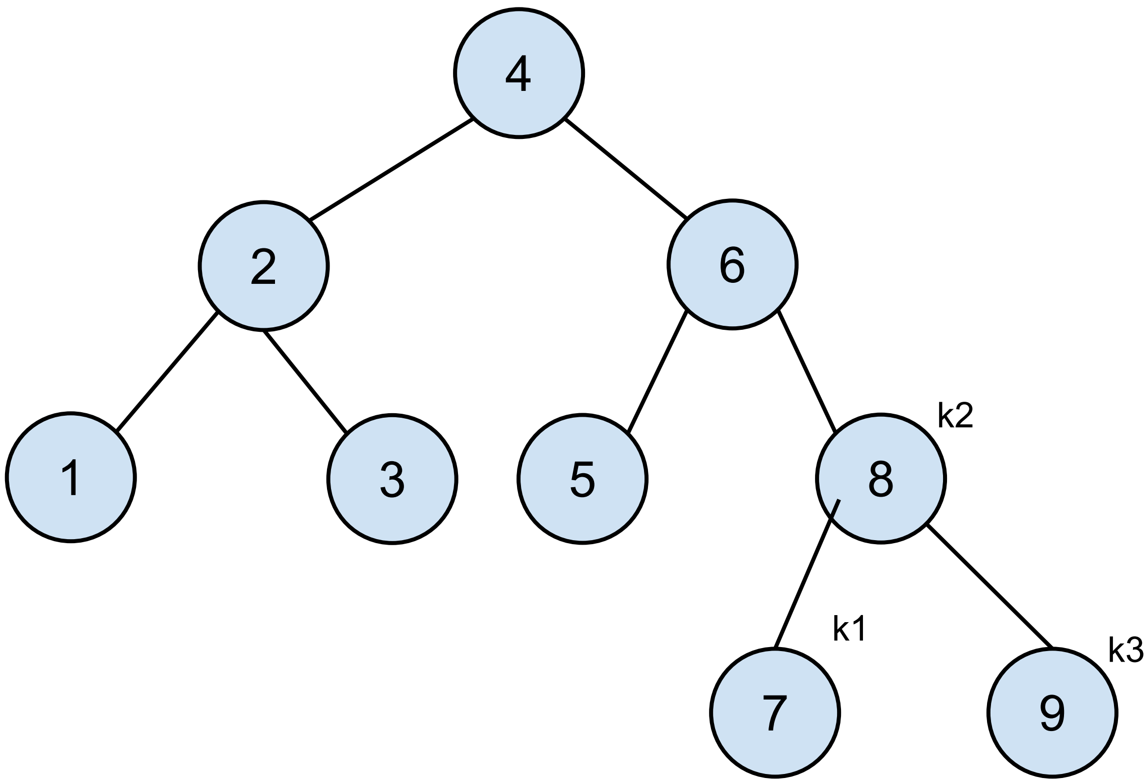
Insert:

AVL Tree: Insert



Insert:

AVL Tree: Insert



Insert: