UNIVERSITY of WASHINGTON

CSS 343: Data Structures, Algorithms, and Discrete Mathematics II

AVL tree

Version 1

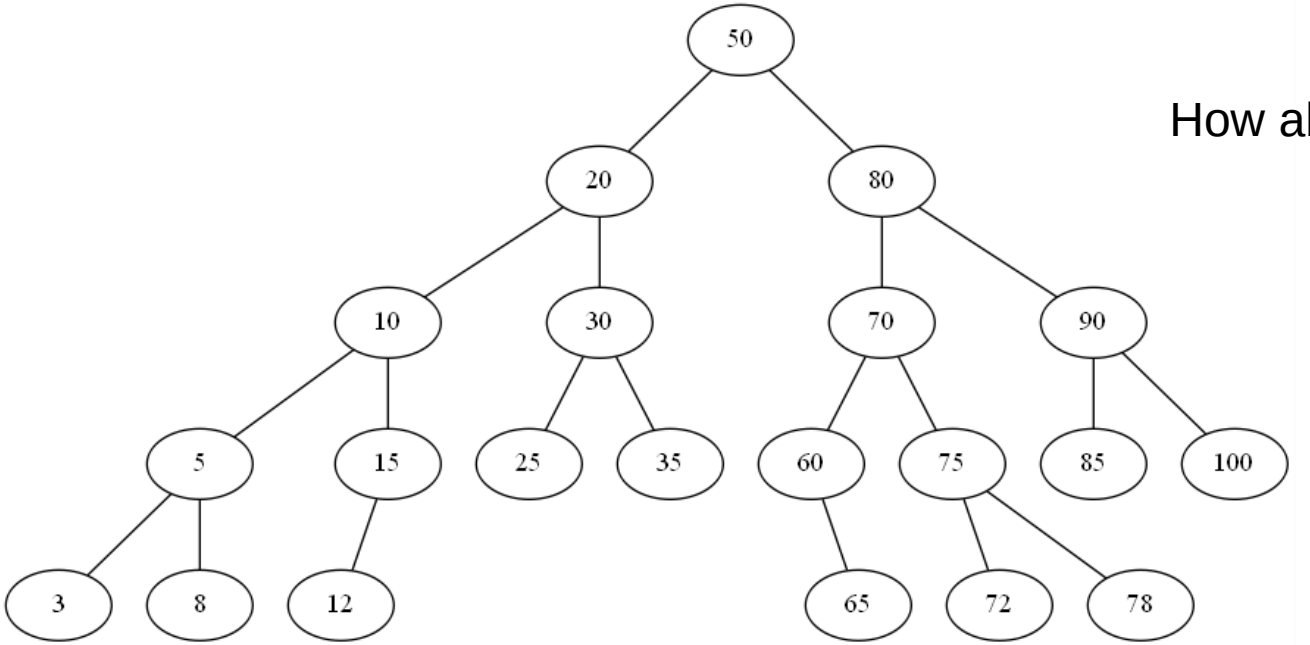
Wooyoung Kim

Balanced tree

- Worst case complexity of an ordinary binary search tree?
  - O(n)
- Can we achieve O(log n)?
  - Remain balanced or nearly balanced while updating
- Balanced tree
  - AVL
  - 2-3
  - B-tree
  - Etc

AVL

- AVL
  - Binary search tree AND
  - For each vertex in the tree, the height of the left and right subtrees differ by at most one
  - Named after two inventors, Adelson-Velsky and Landis

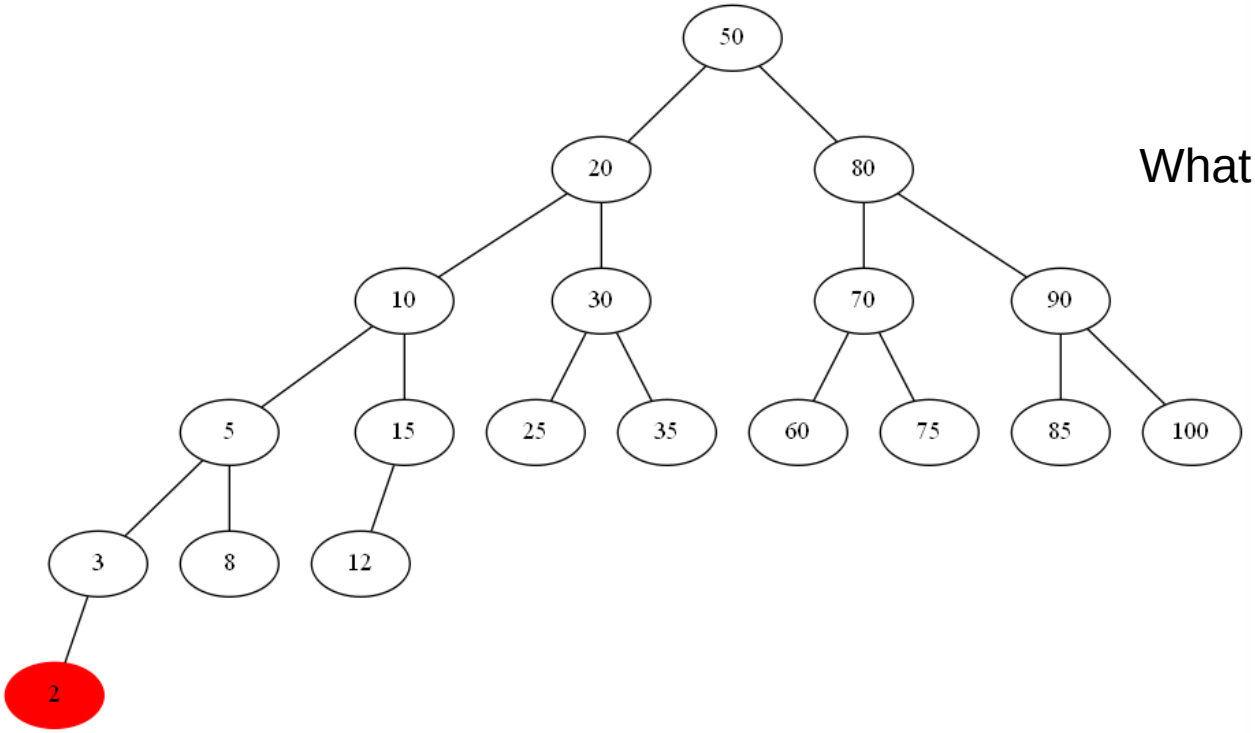


AVL

How about inserting 2?

AVL

- AVL
  - Binary search tree AND
  - For each vertex in the tree, the height of the left and right subtrees differ by at most one

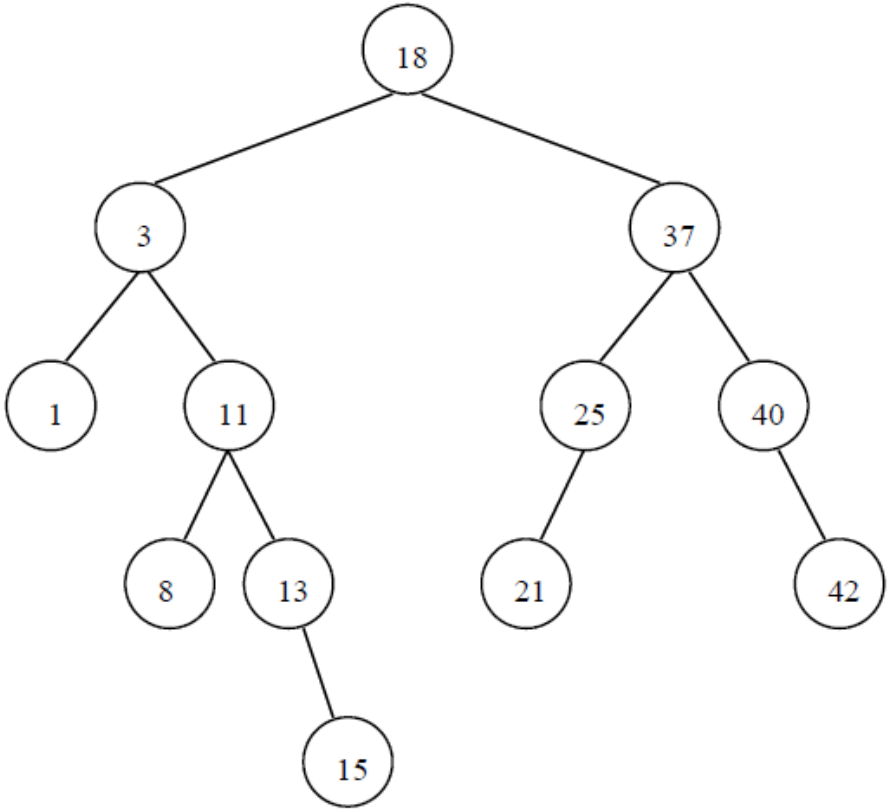


What node is unbalanced one?

non-AVL

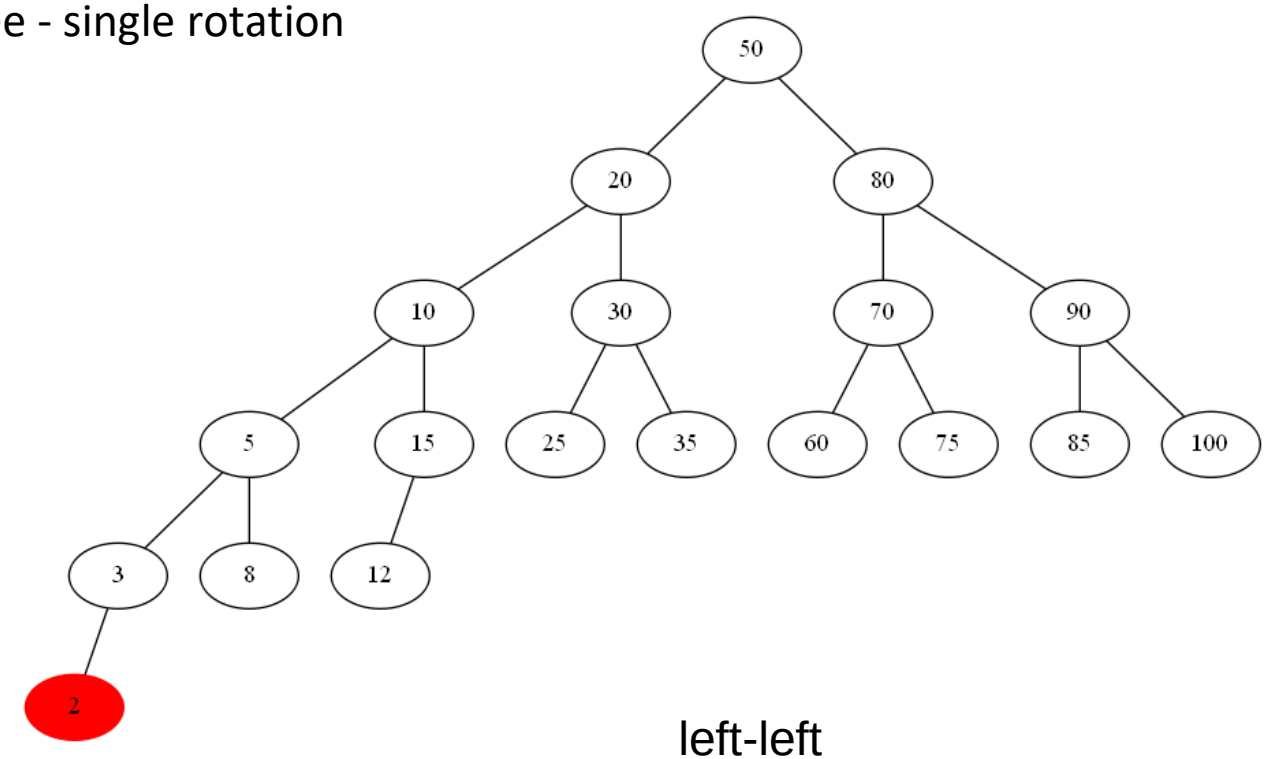
Height - review

- Height of a node: between node and its leaf (leaf node's height is 1)
- In assignment 2: getHeight
  - keep maximal values (longest path upwards)



Single rotation

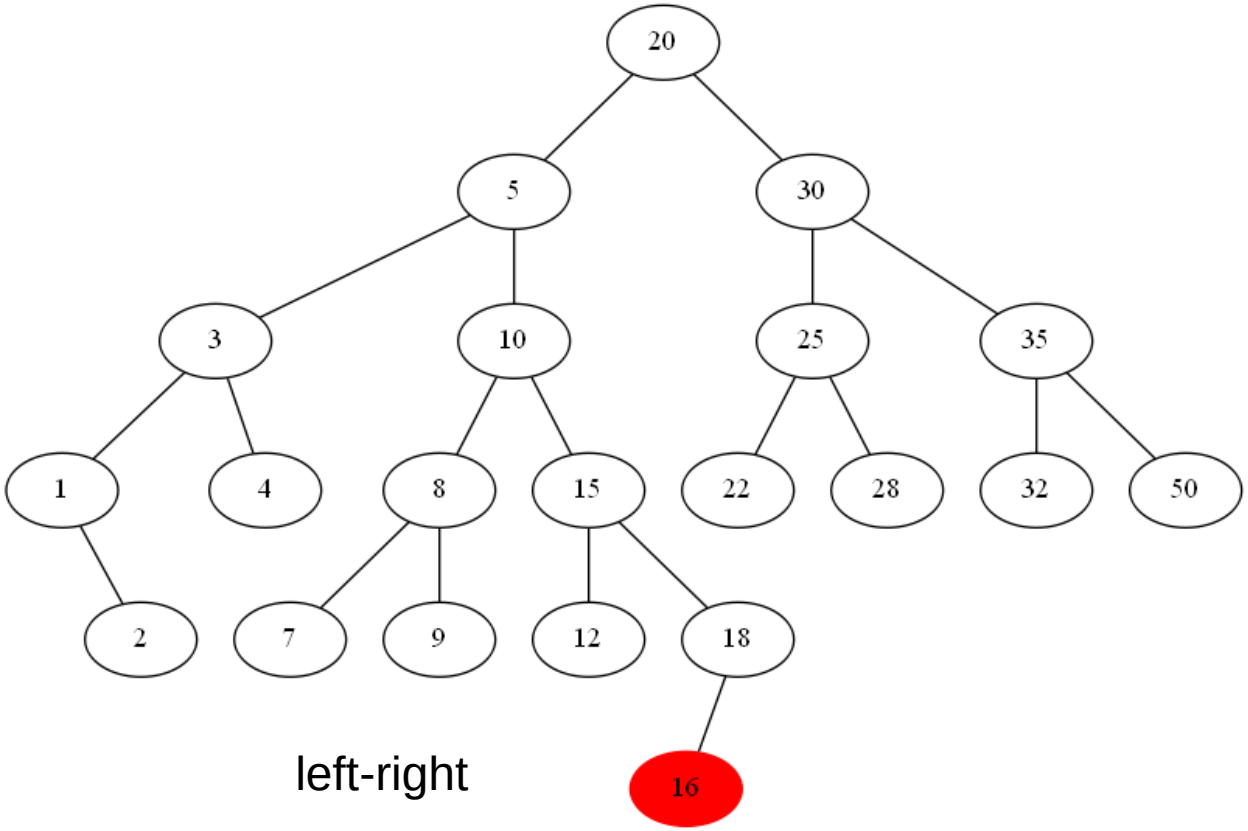
- Key to AVL: keep it balanced when update is performed
- Starting from AVL, four possible insertion cases leading to non-AVL and rotations to fix:
  - Left-left: insertion in (the left child of the unbalanced vertex)'s the left subtree – single rotation
  - Right-right: insertion in (the right child of the unbalanced vertex)'s right subtree - single rotation



left-left

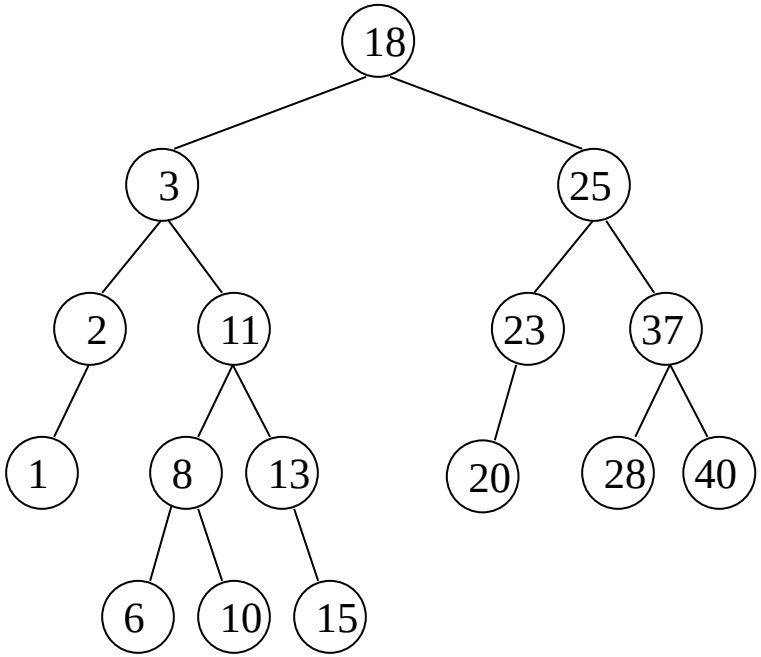
Double rotations

- Left-right: insertion in (the left child of the unbalanced vertex)'s right subtree – double rotation
- Right-left: insertion in (the right child of the unbalanced vertex)'s left subtree – double rotation



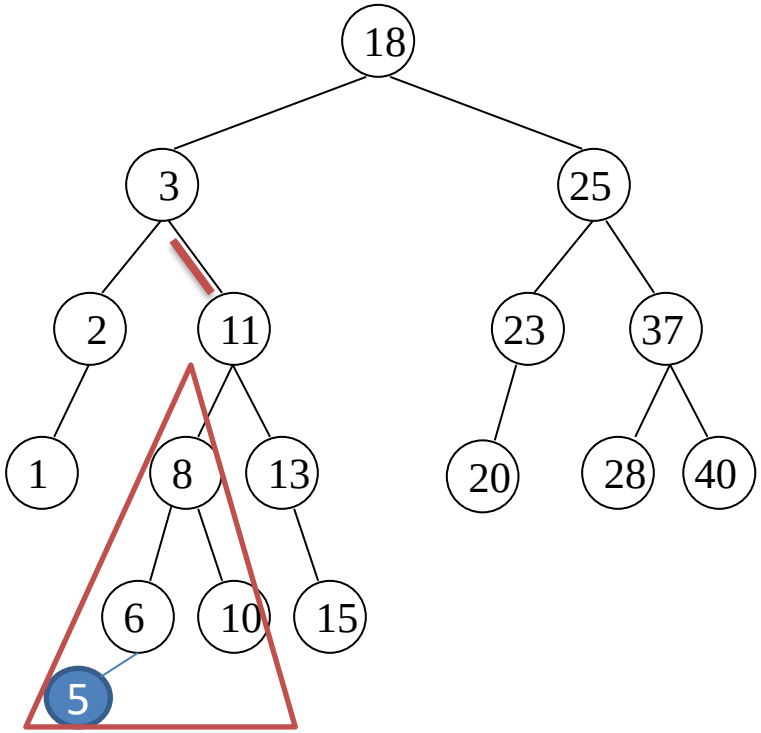
Practice – which insertion type

- After insertion, find the lowest unbalanced vertex
- E.g., insert 5:



Practice – which insertion type

- After insertion, find the lowest unbalanced vertex
- E.g., insert 5:



Unbalanced vertex: 3

right-left type

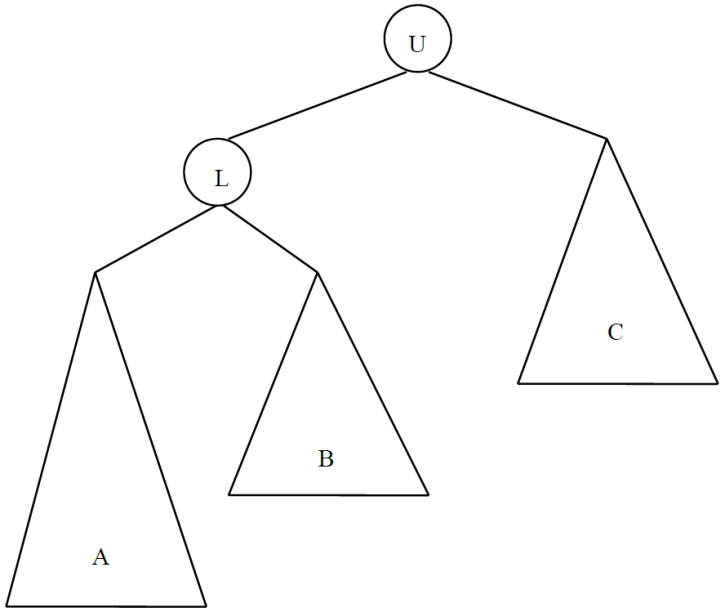
Steps

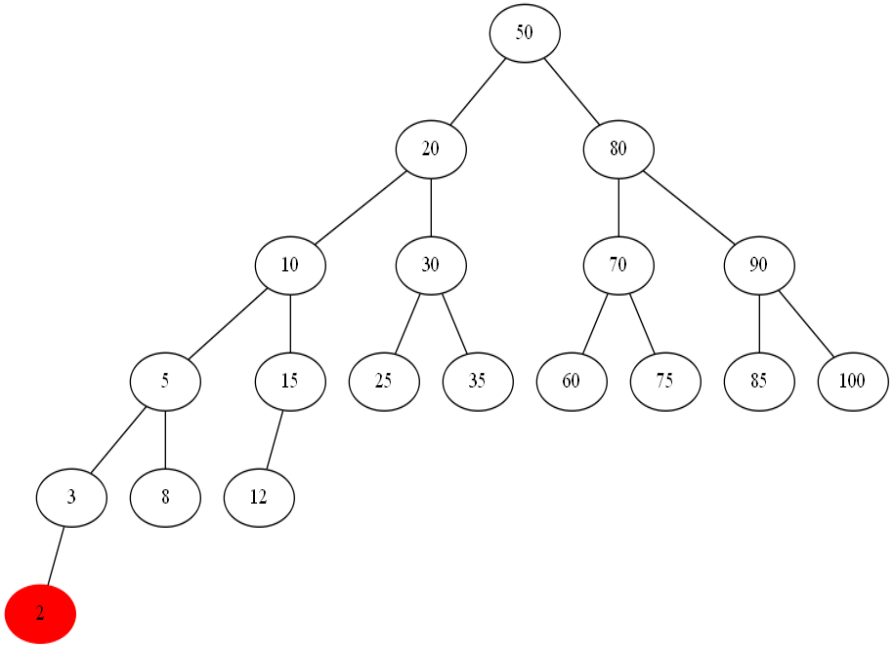
Precondition: AVL tree

- Step 1: After an insertion, find the lowest unbalanced vertex
- Step 2: Identify insertion type
- Step 3: Make rotation accordingly

Single rotation

- Left-left: find the lowest unbalanced node - U

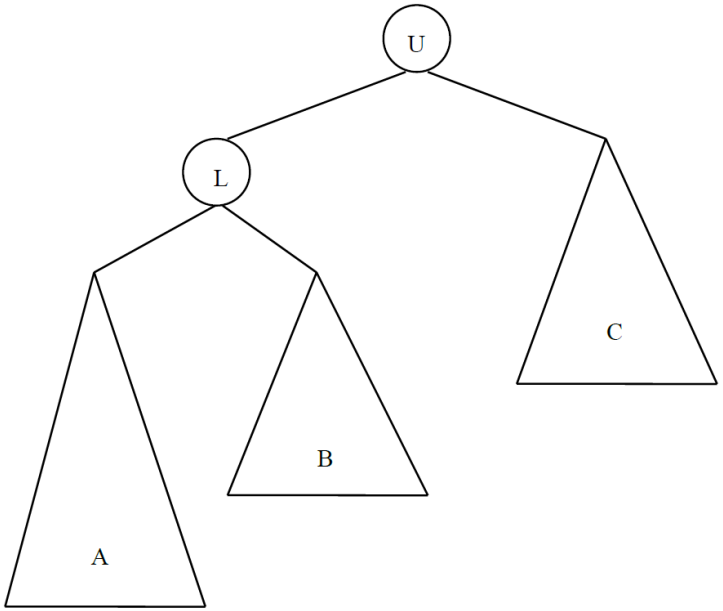


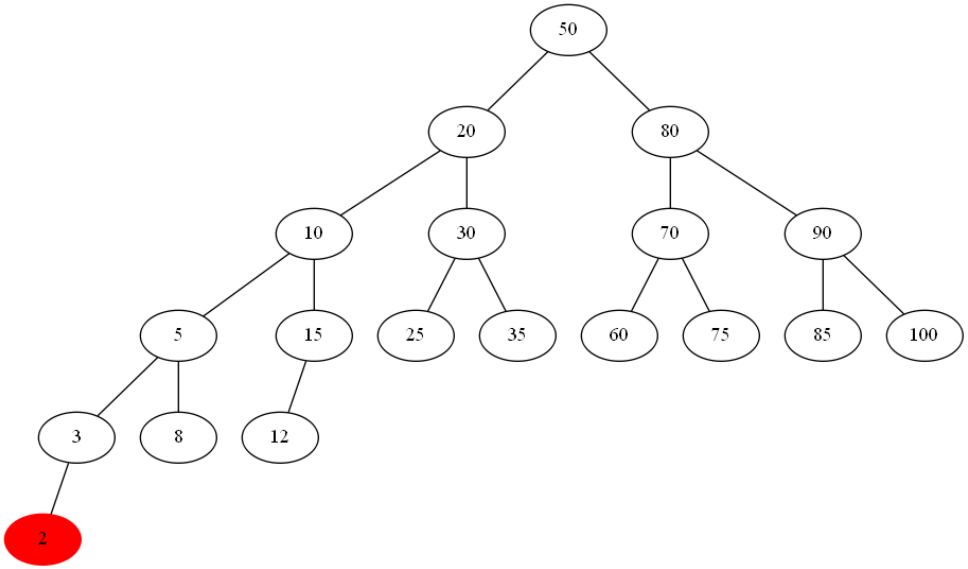


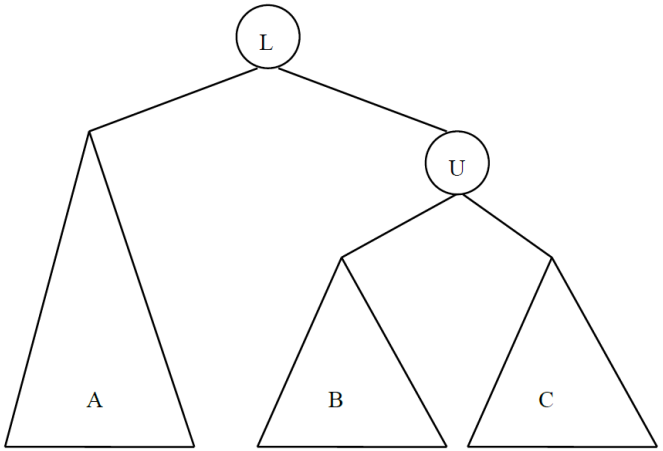
- What is the lowest unbalanced node in the right tree?
  - 20
  - Because its left subtree is two levels deeper than its right subtree

Single rotation

- Make to AVL by single rotation

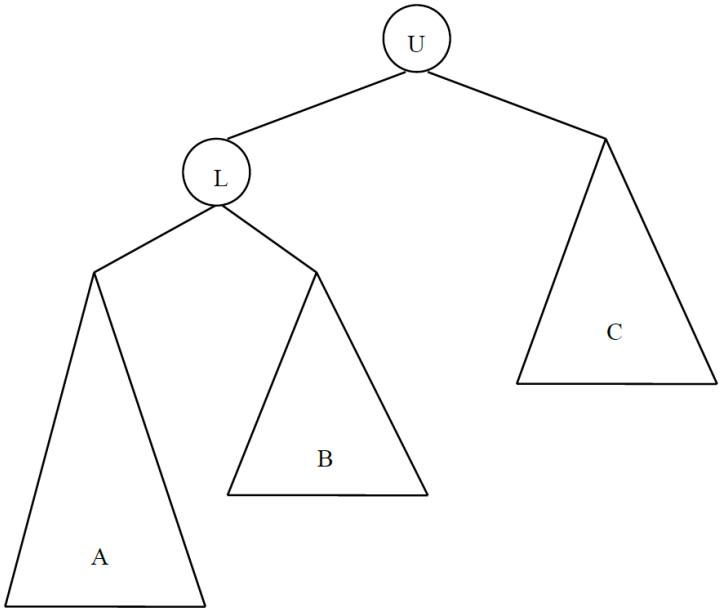


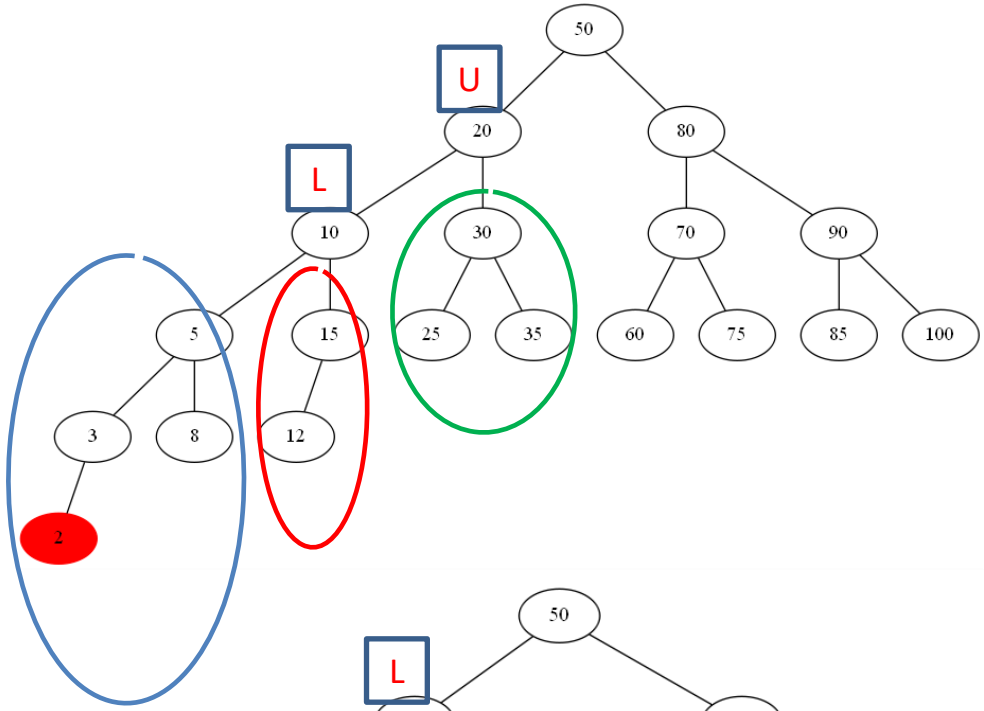


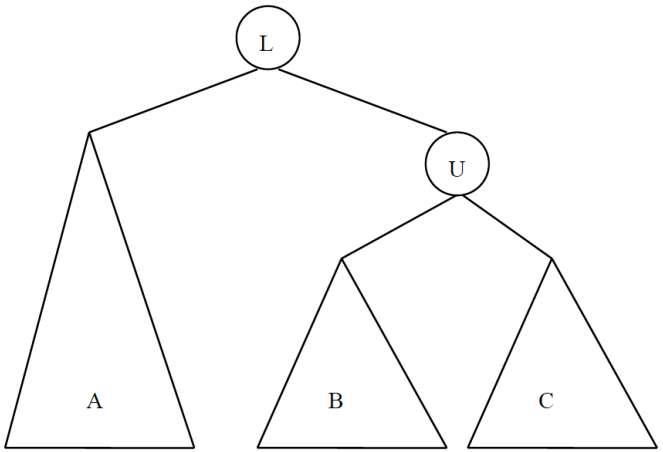


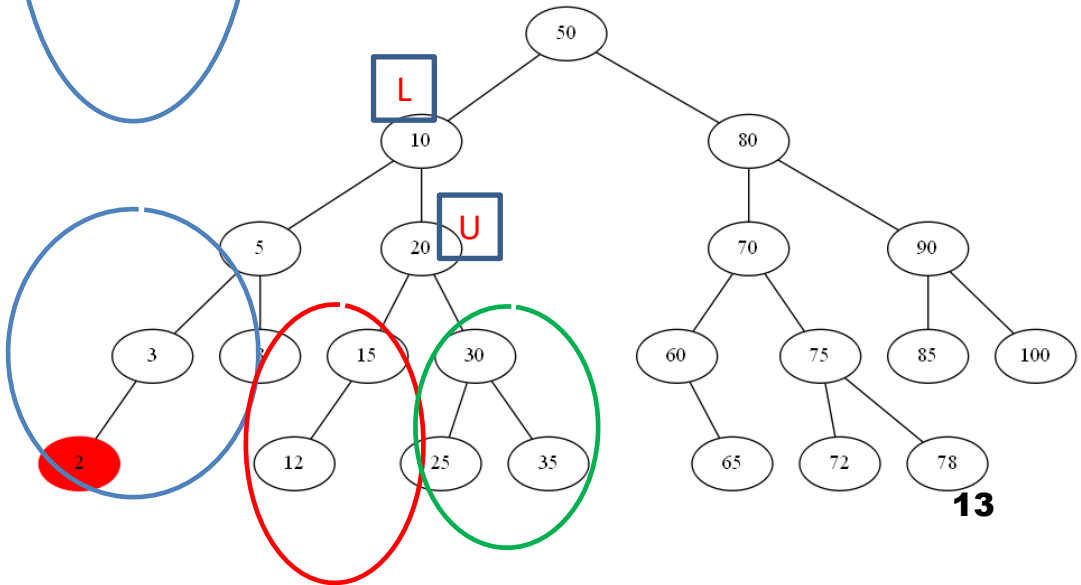
Single rotation

- Make to AVL by single rotation



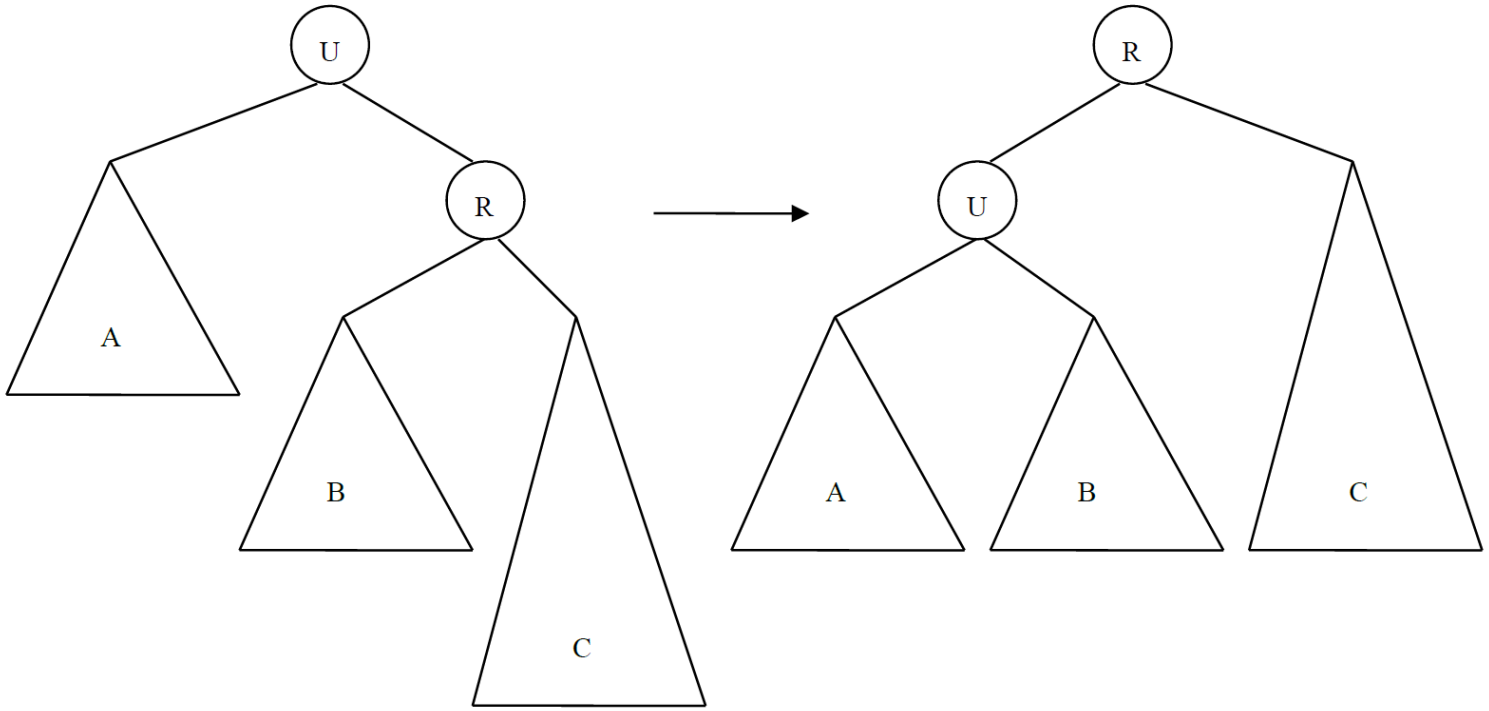




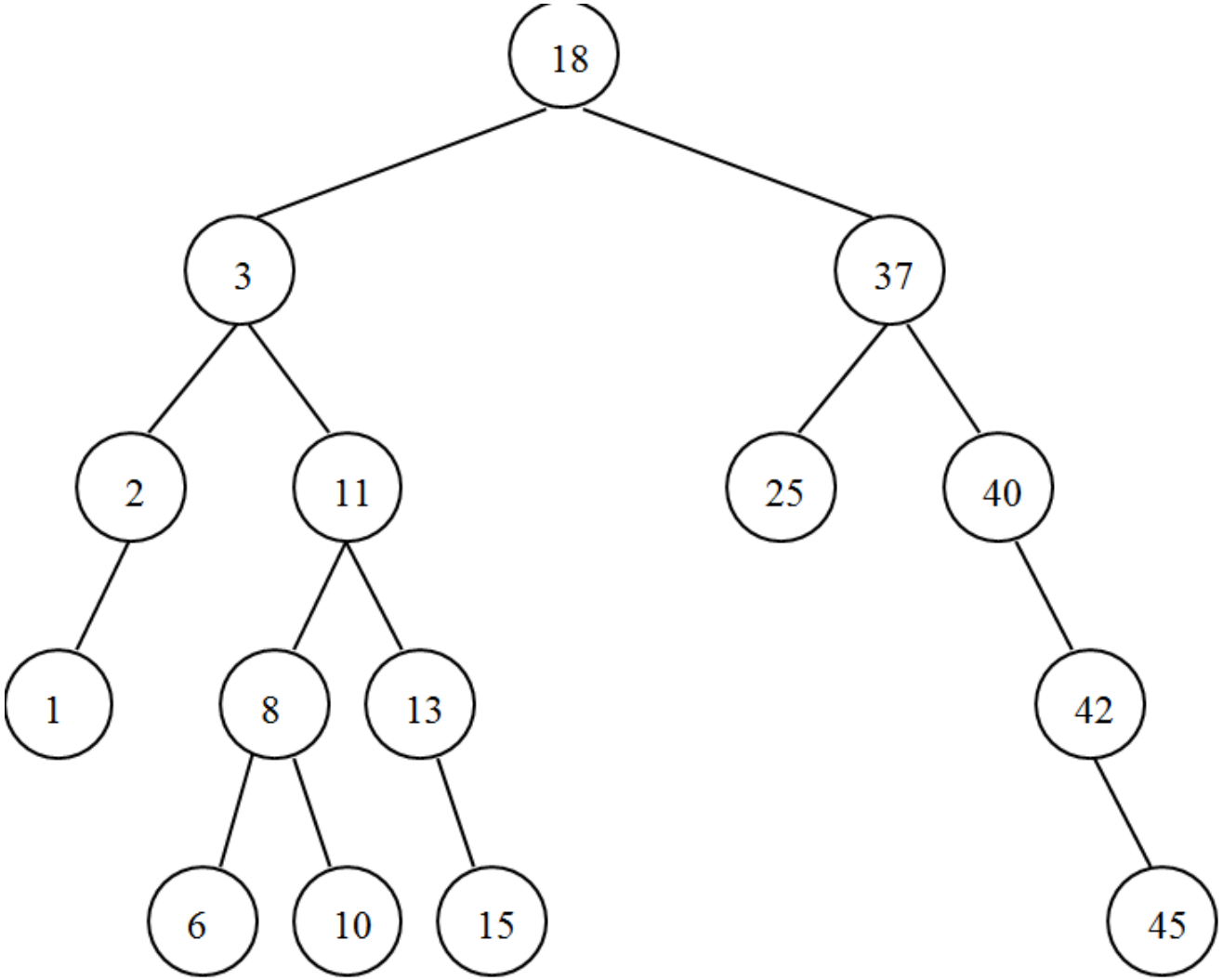


Single rotation (cont.)

- Right-right: symmetric to left-left

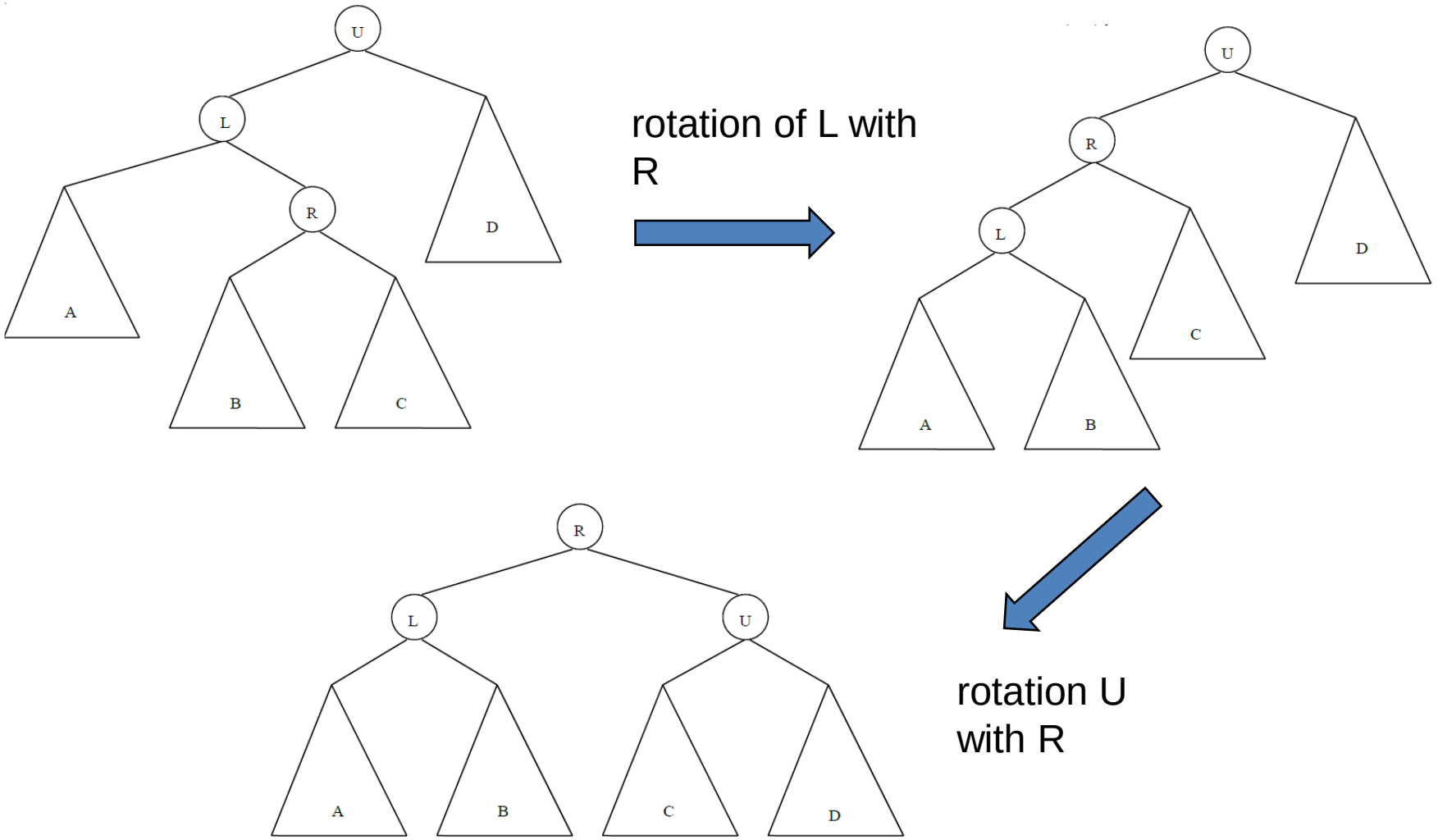


Single rotation - exercise



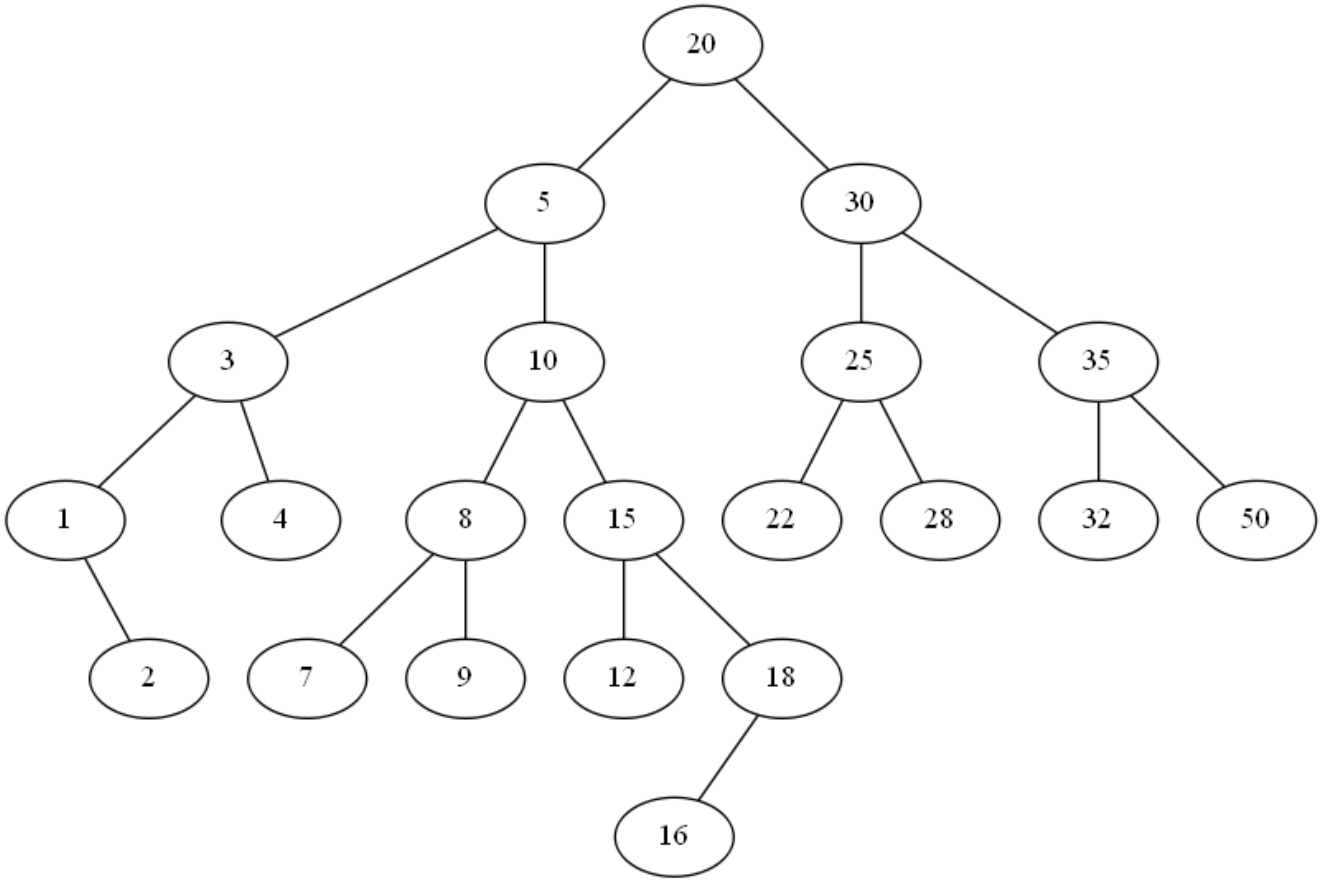
Double rotation

- Left-right



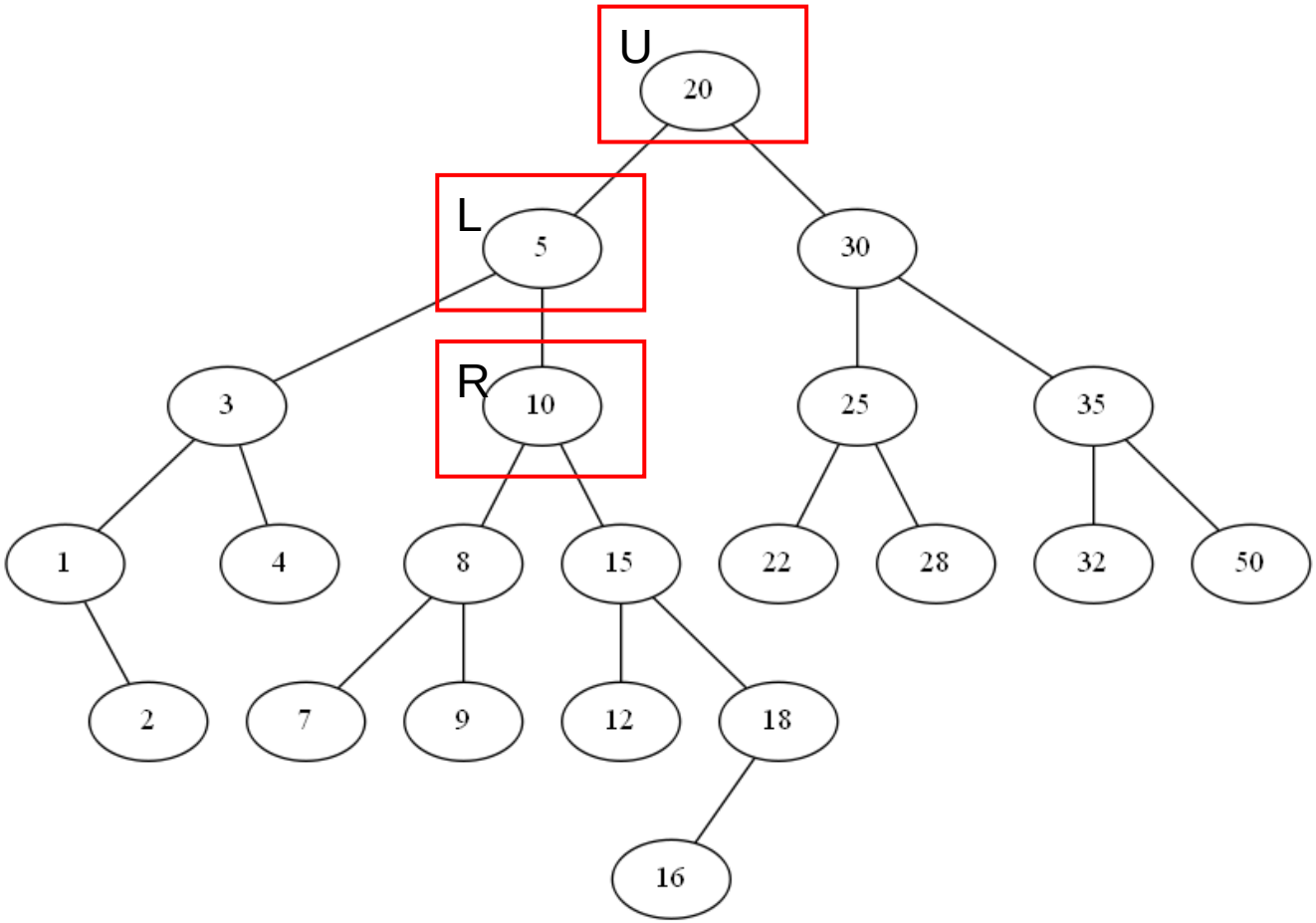
Double rotation-practice

- What is U? L? R?



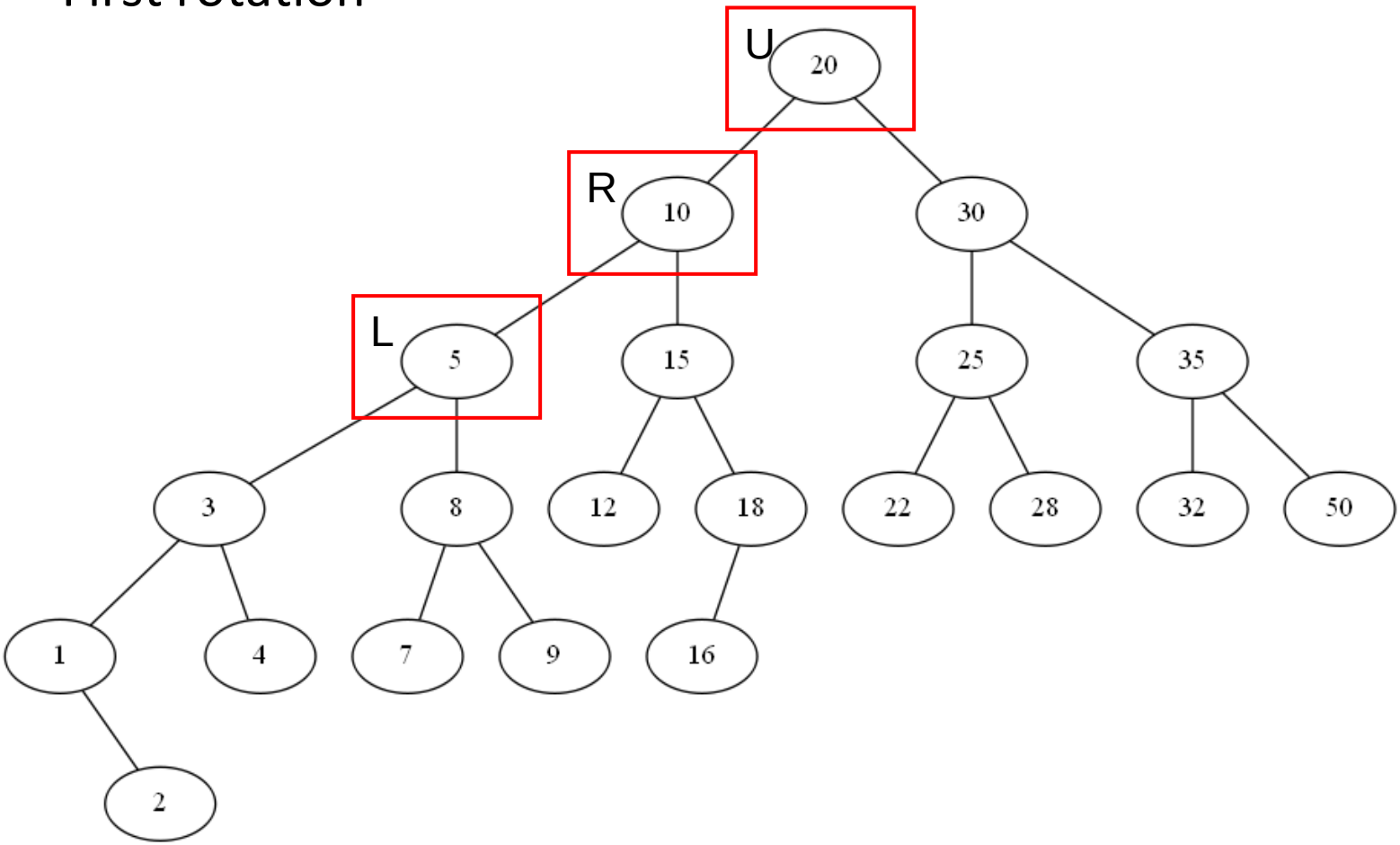
Double rotation-practice

- What is U? L? R?



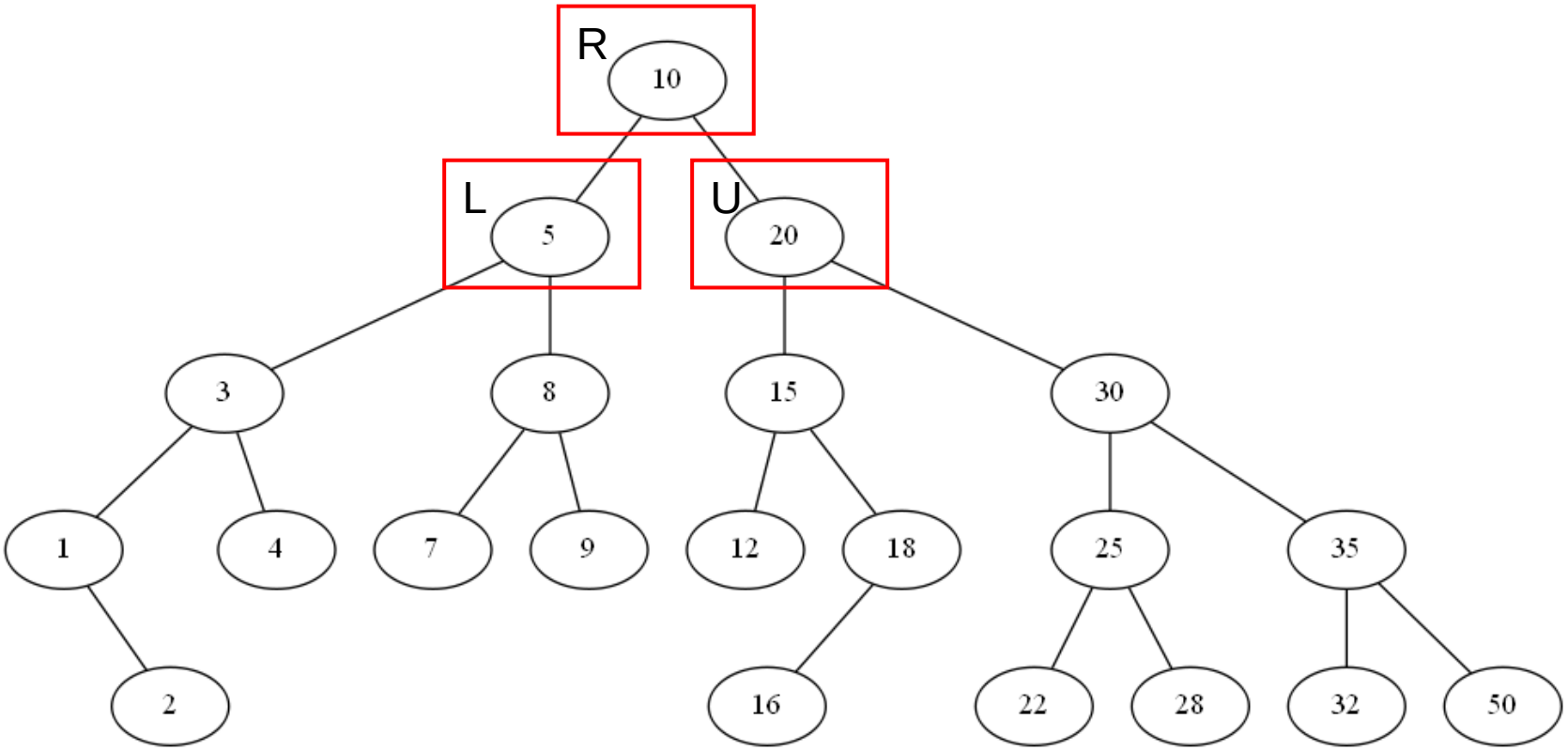
Double rotation-practice

- First rotation



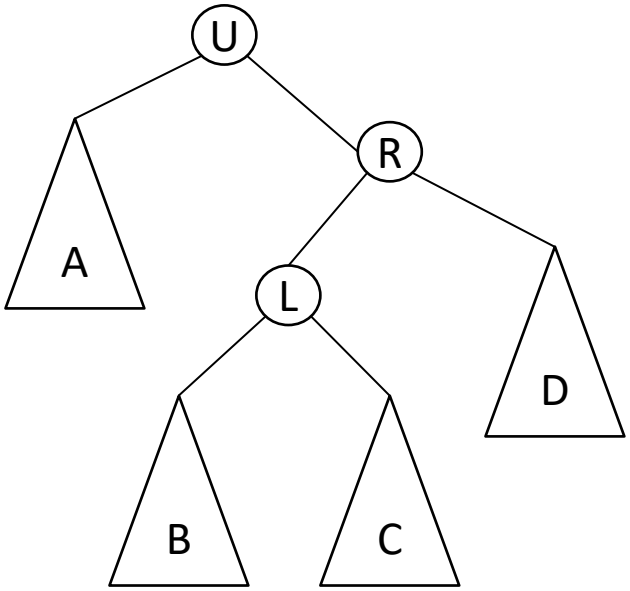
Double rotation-practice

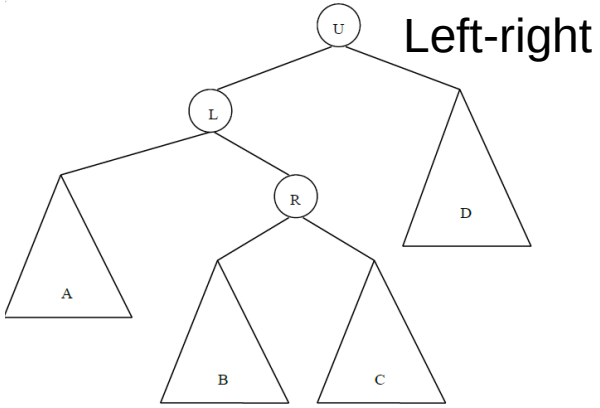
- Second rotation



Double rotation

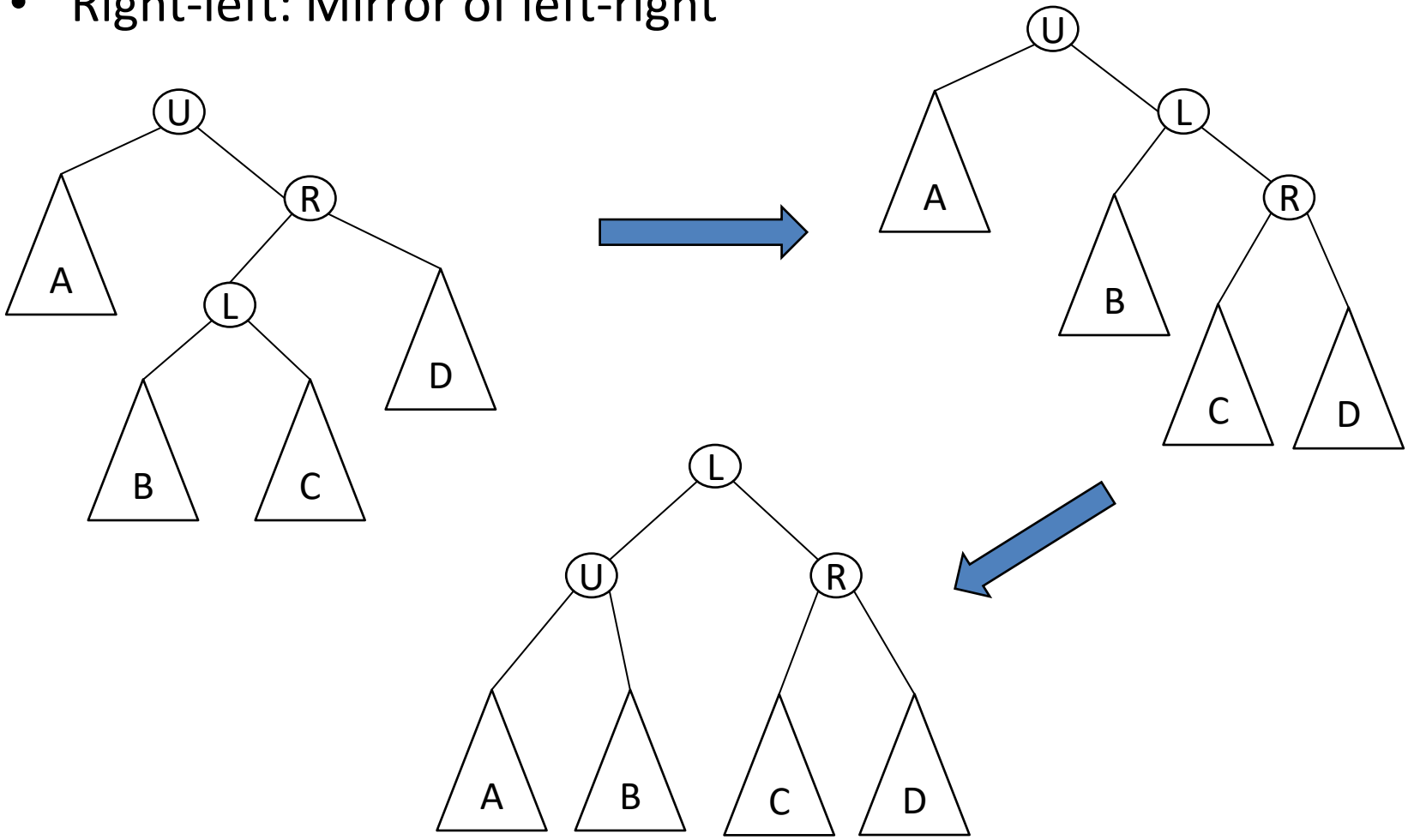
- Right-left: Mirror of left-right





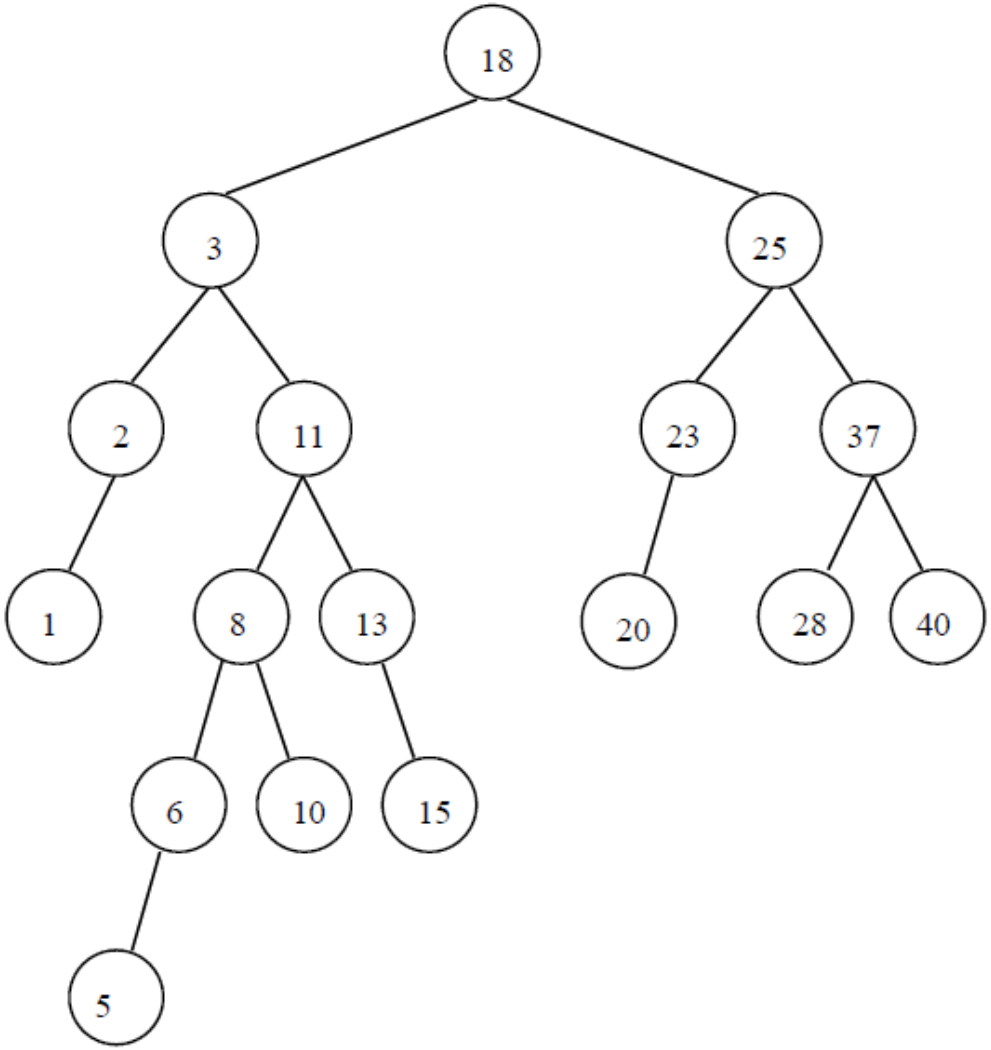
Double rotation

- Right-left: Mirror of left-right



Double rotation - exercises

- Right-left: Mirror of left-right(See lecture note)



AVL Tree- complexity

- The rotation routines are all O(1)
- Insertion takes O(log N), when N is the number of nodes, and log N is the height of the tree
- Insert: O(log N), requires rotation
- Search: O(log N)
- Remove: O(log N), requires rotation