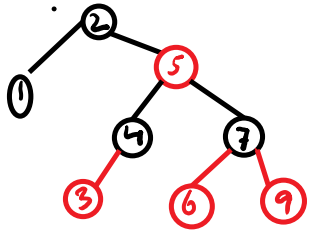


Habib University
shaping futures

CS 201 Data Structure II (L2 / L5)

AVL Trees

Muhammad Qasim Pasta

qasim.pasta@sse.habib.edu.pk

Slides are developed for discussion in the class, not for reference material.

Visual aids are taken from the slide deck of Dr. Vishnu Kotrajaras

Class Norms:

L2

Mark your attendance using biometric machines

- Chit-chat during the lectures – Don't
- Receiving calls – leave the class
- Ask questions – Do's
- Sleeping in the class – twice in a month
- Coming late – Sometimes
- Leaving the class and coming back – without causing disturbance
- Request for early leaves – 15 minutes , once in a month
- Working on laptop – for taking notes
- Checking phones – 3 to 5 times
- …

Class Norms:

L5

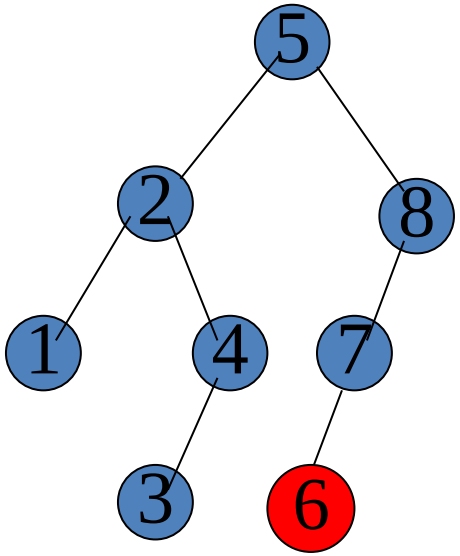
- Chit-chat during the lectures – Don’t
- Receiving calls – leave the class – Do’s
- Ask questions – Do’s
- Sleeping in the class – Do’s
- Coming late – Do’s
- Leaving the class and coming back – non disruptive manner
- Request for early leaves – not often, not more than 15 minutes
- Working on laptop – only taking for notes, not to solve assignment
- and checking phones – 2 to 4 times
- …

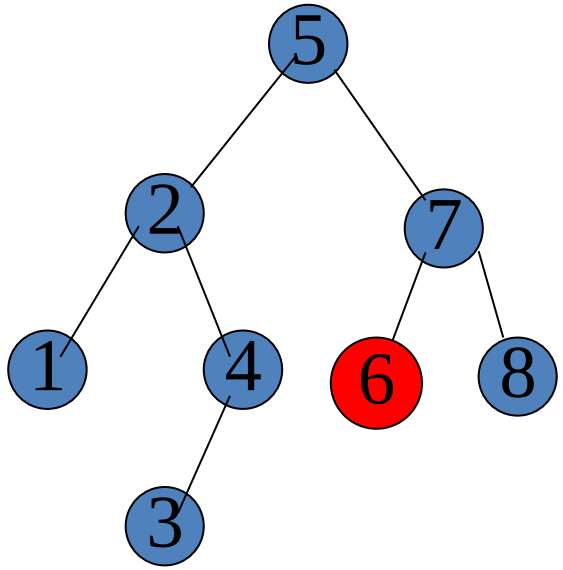
AVL Tree

- Georgy **Adelson-Velsky** and Evgenii Mikhailovich **Landis**
- Balanced Binary Search Tree
  - also known as height-balanced BST
- The height of each node from its left and right subtree can only differ at most 1.
- Search, Insertion, and Removal of a node can be achieve in O(log n)

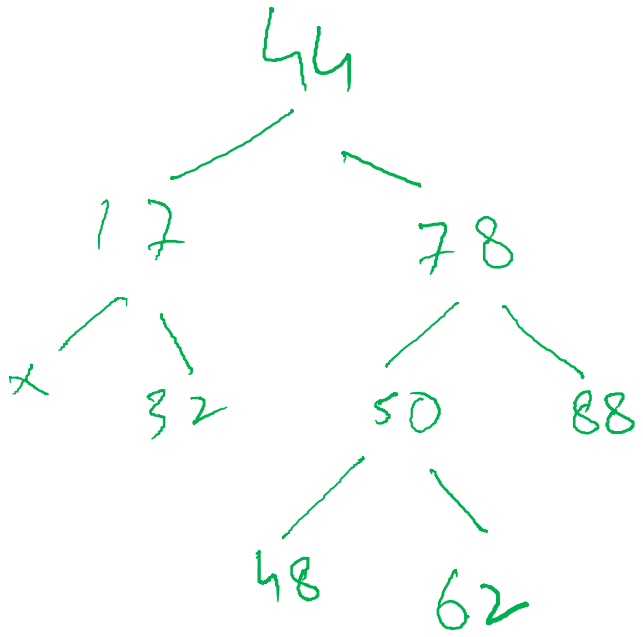
AVL Tree Example

- Balance factor of a node › height(left subtree) - height(right subtree)
- For every node, heights of left and right subtree can differ by no more than 1 (-1, 0, 1) aka Rank
  - Height and Rank are consider interchangeable, to differentiate:
    - R(n) = number of nodes (not edges) from the node to deepest leave
    - Or R(n) = H(n) + 1
- R(n) = R(L) – R(R)





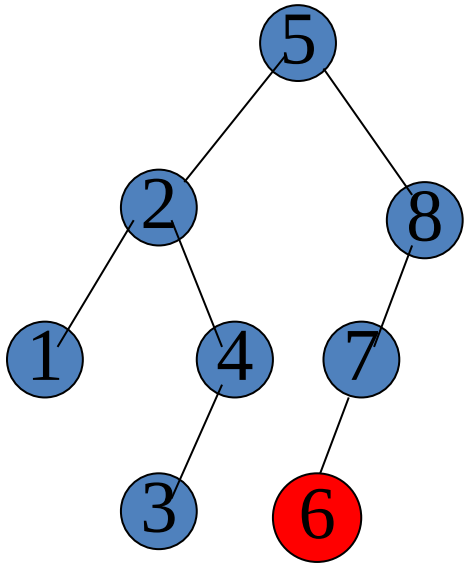
More Example

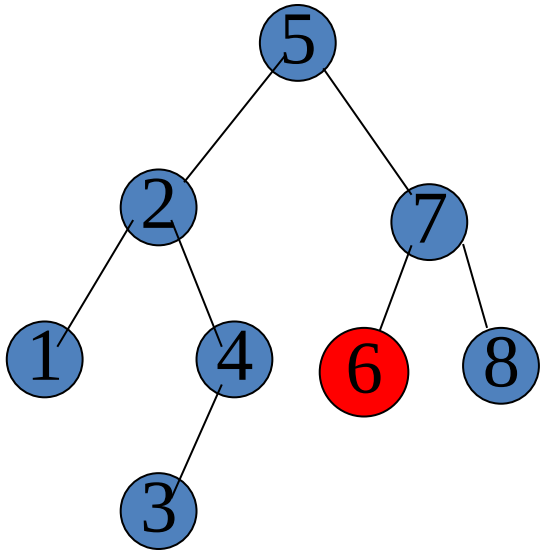


Node	Height	Rank
44		
17		
78		
32		
50		
88		
48		
62		

Insertion:

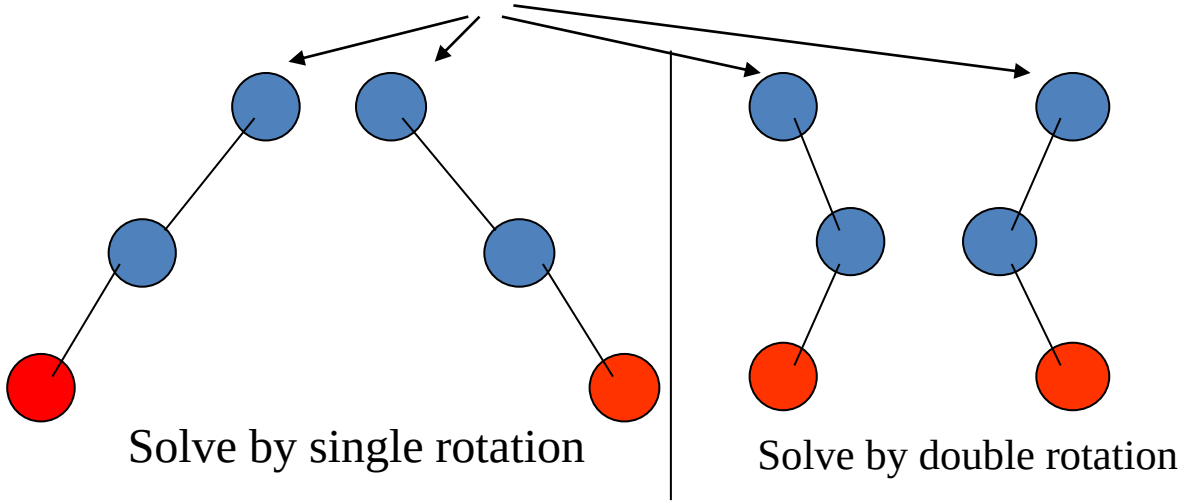
- Insert as in BST and then fix the height
- Only nodes on the path from the insertion point to the root have a chance of height change.
- Follow from inserted node to find the node violating height/rank property
- Fix by rotations





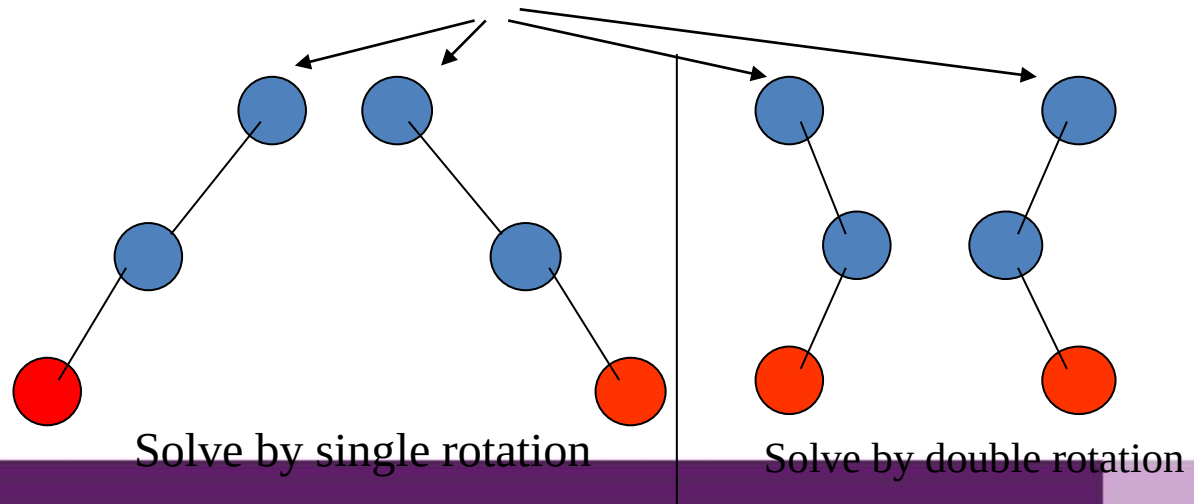
Fixation:

- There can be four cases:
  - Left-Left: Insertion into left subtree of left child
  - Right-Right: Insertion into right subtree of right child
  - Left-Right: Insertion into right subtree of left child
  - Right-Left: Insertion into left subtree of right child

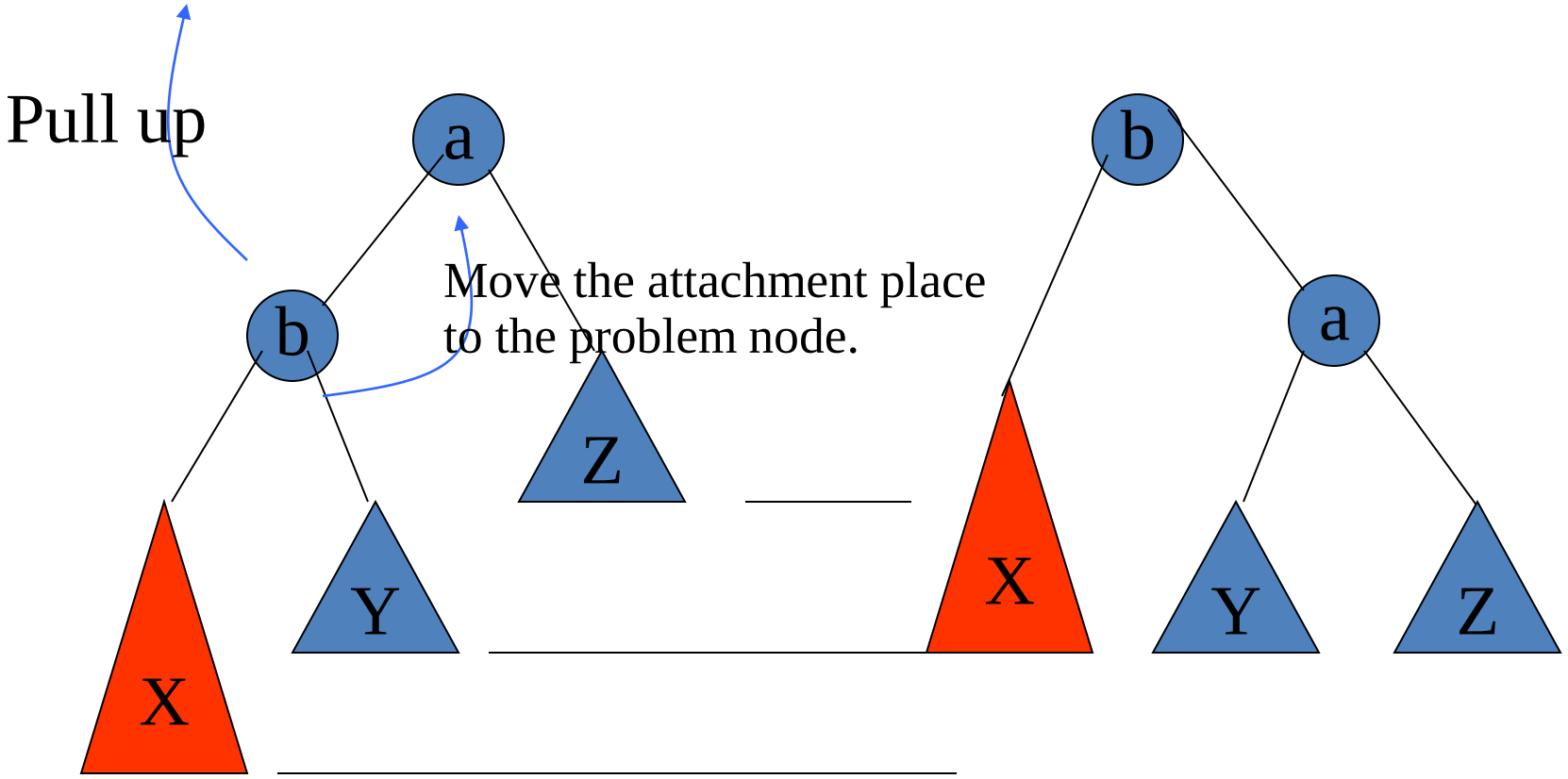


Four Cases

- Left – Left: perform **Right** rotation at violation node
- Right – Right: perform **Left** rotation at violation node
- **Left-Right**:
  - Perform **Left** rotation at violation node's child
  - Perform **Right** rotation at violation node
- **Right-Left**
  - Perform **Right** rotation at violation node's child
  - Perform **Left** rotation at violation node

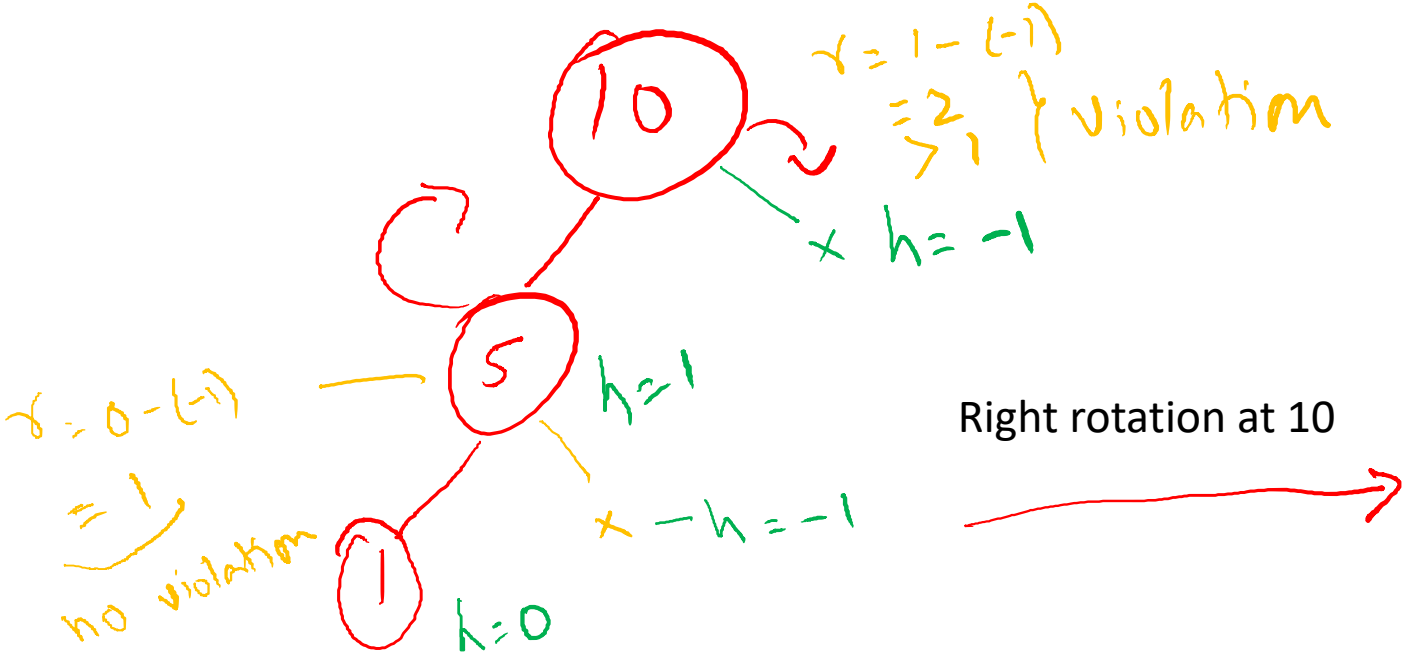


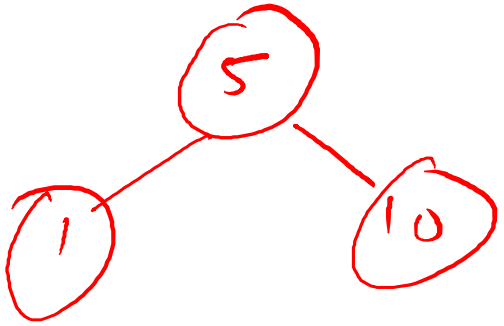
Single rotation after insert



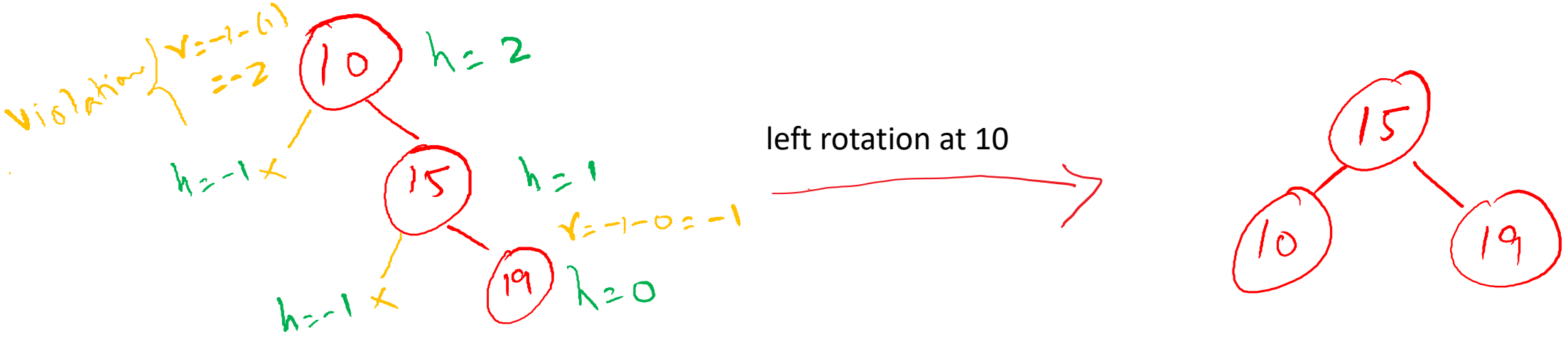
If it is fixed now, no further rotation is needed.

Left-Left: perform **Right** rotation

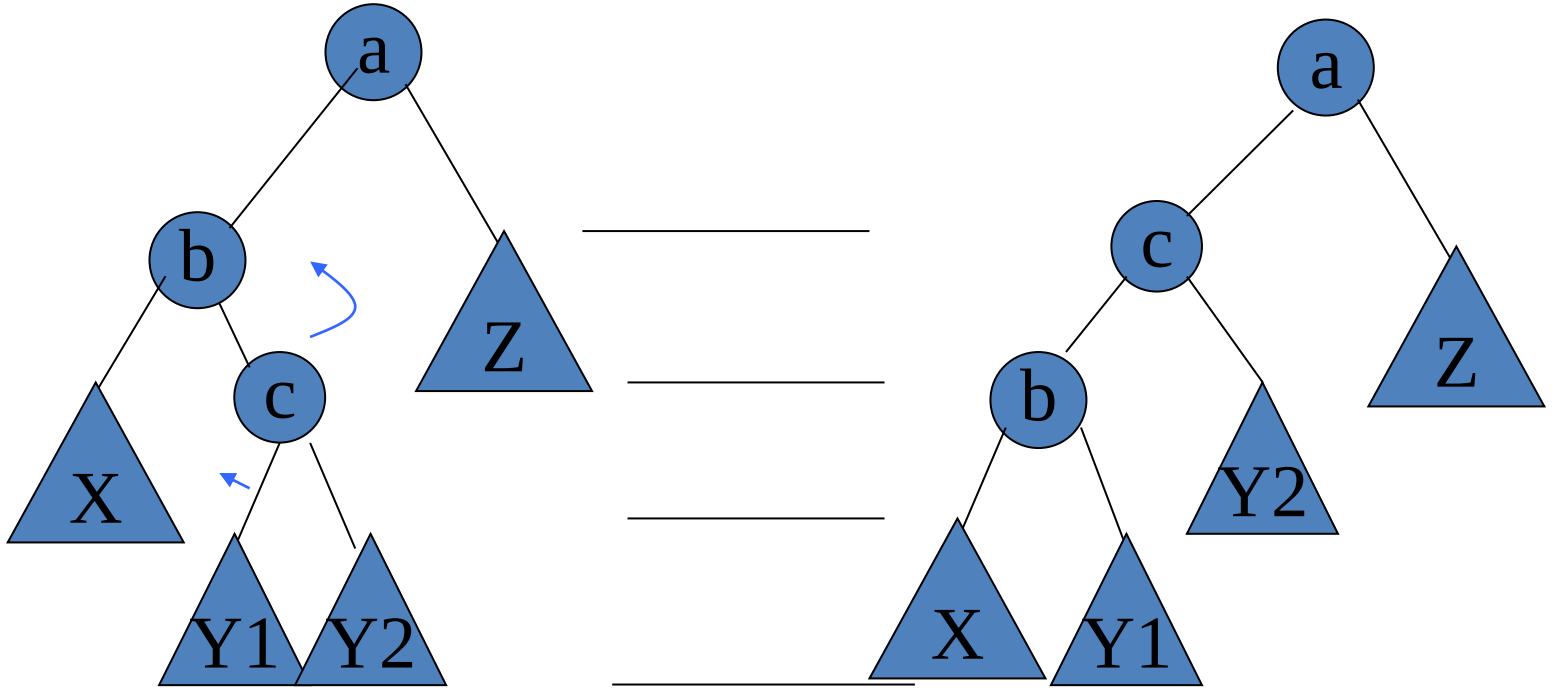




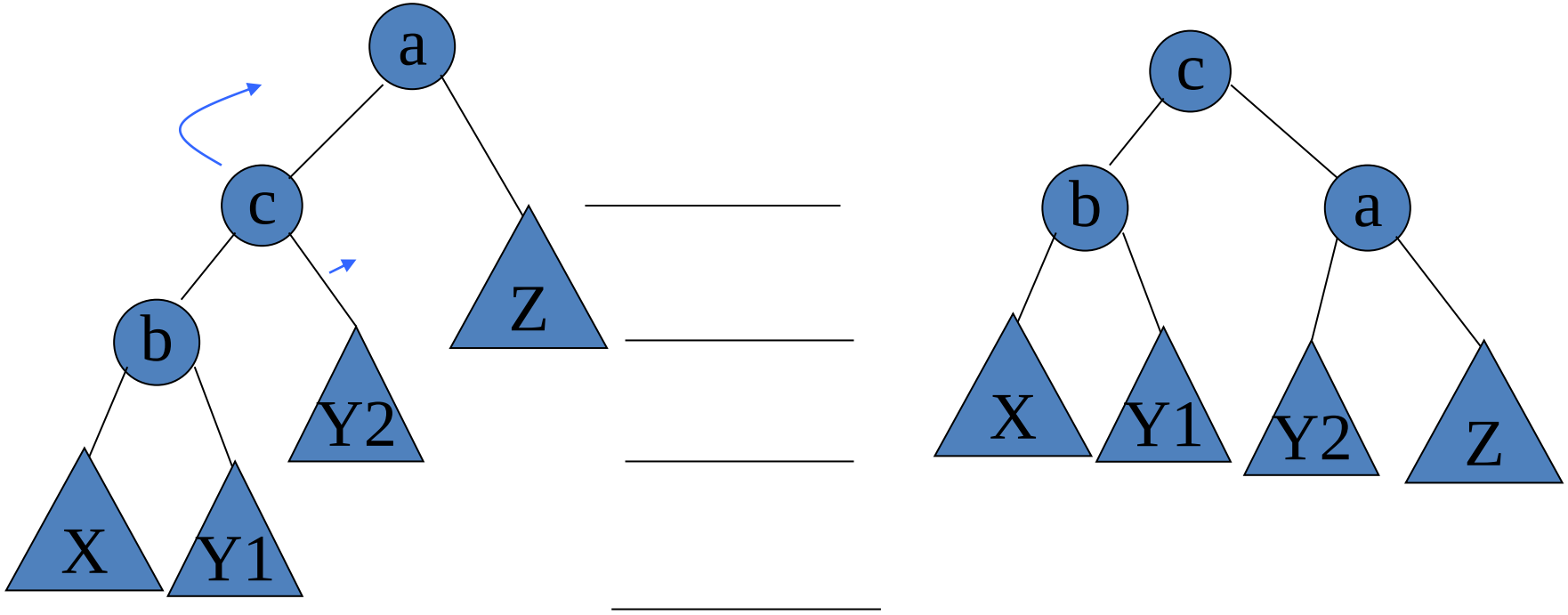
Right-Right: perform **Left** rotation



Double Rotation: first

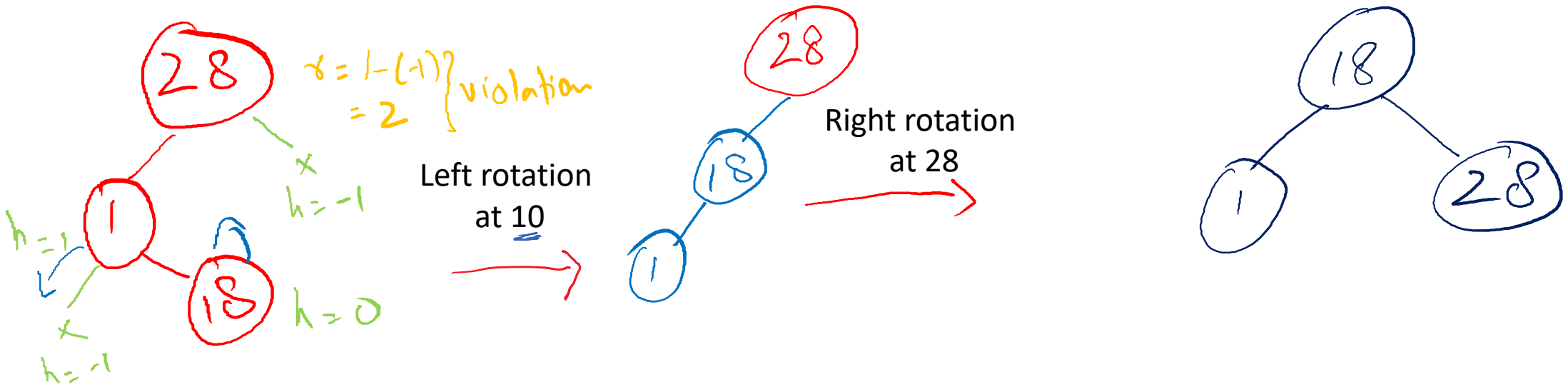


Double Rotation: second



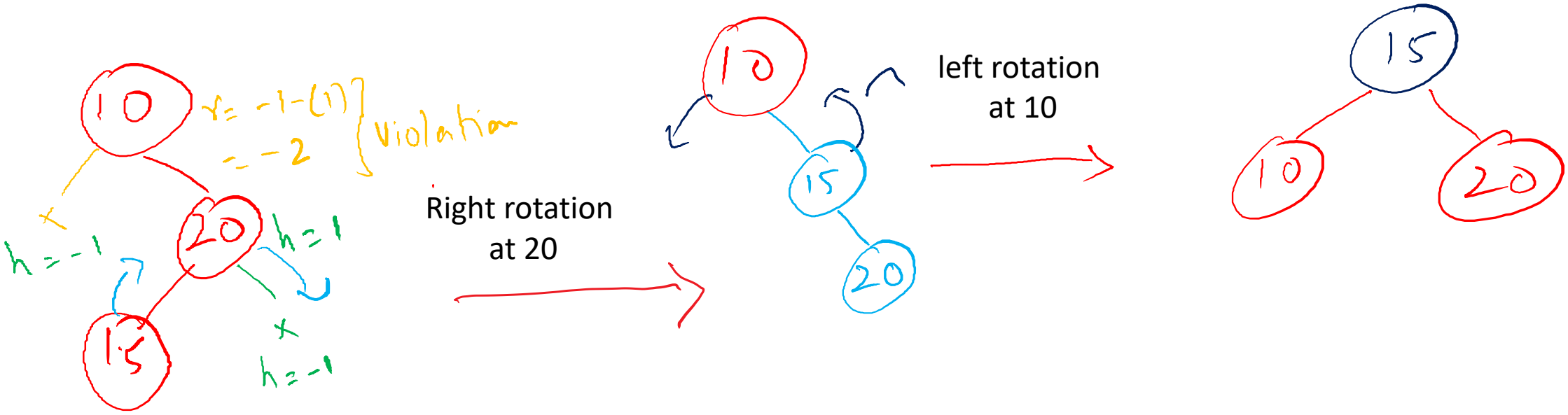
Left-Right: double rotations

- **Left-Right**:
  - Perform **Left** rotation at violation node's child
  - Perform **Right** rotation at violation node



Right-Left: double rotations

- **Right-Left**
  - Perform **Right** rotation at violation node's child
  - Perform **Left** rotation at violation node

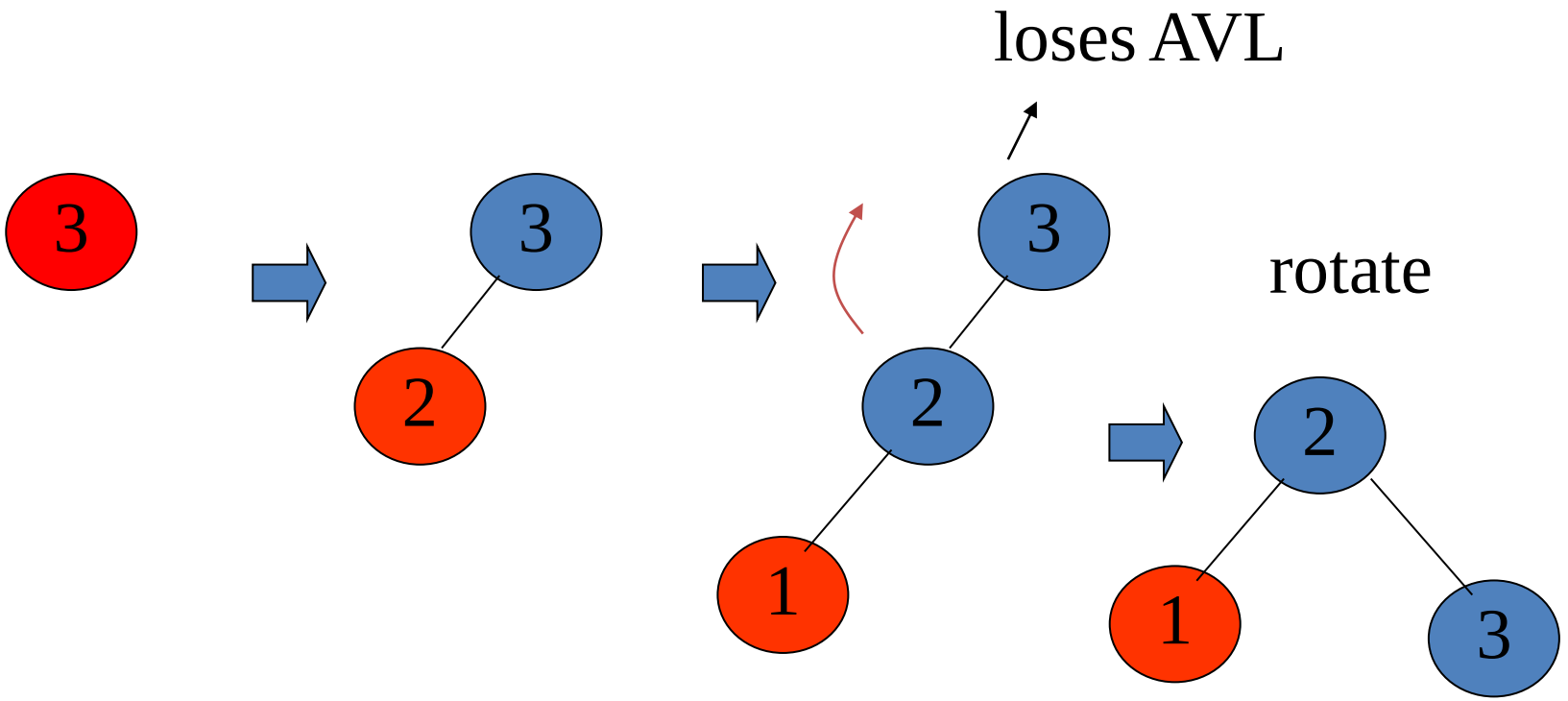


Example

- AVL Tree for sequence: 28, 1, 18 14, 3, 35, 39

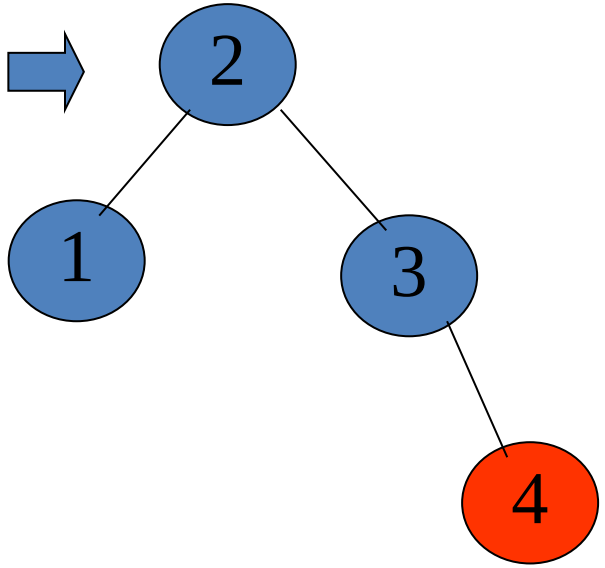
Example

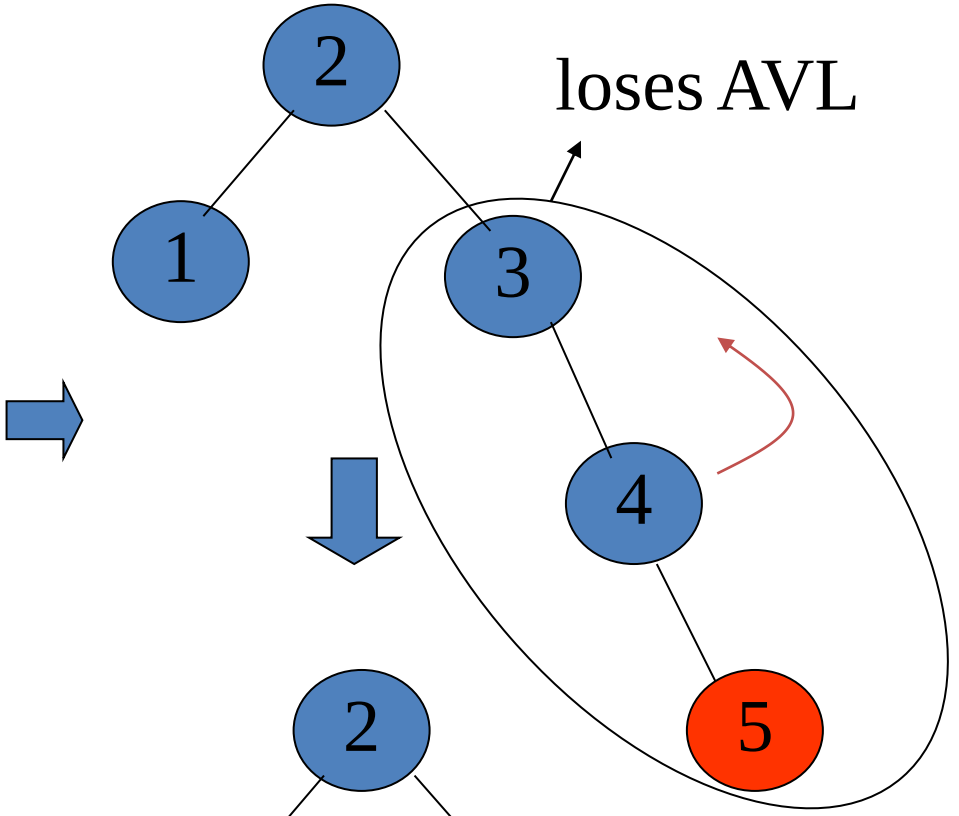
- Starting from nothing, we insert 3,2,1



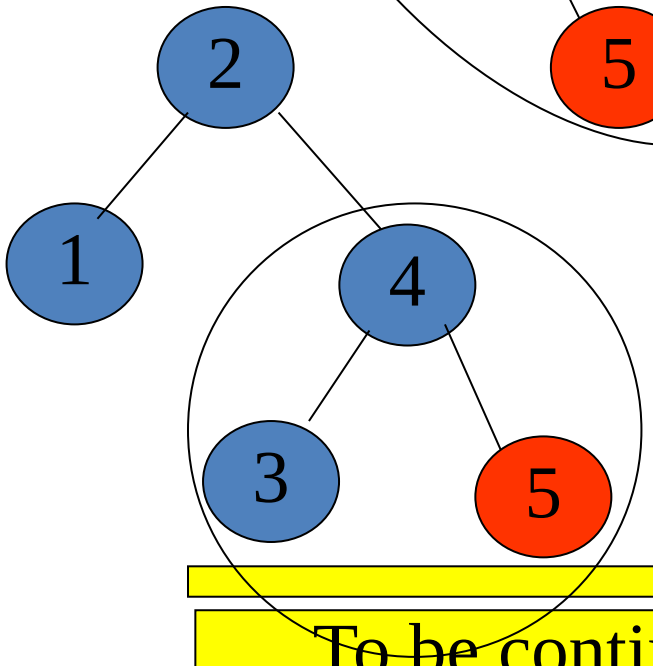
To be continued

Then we insert 4,5



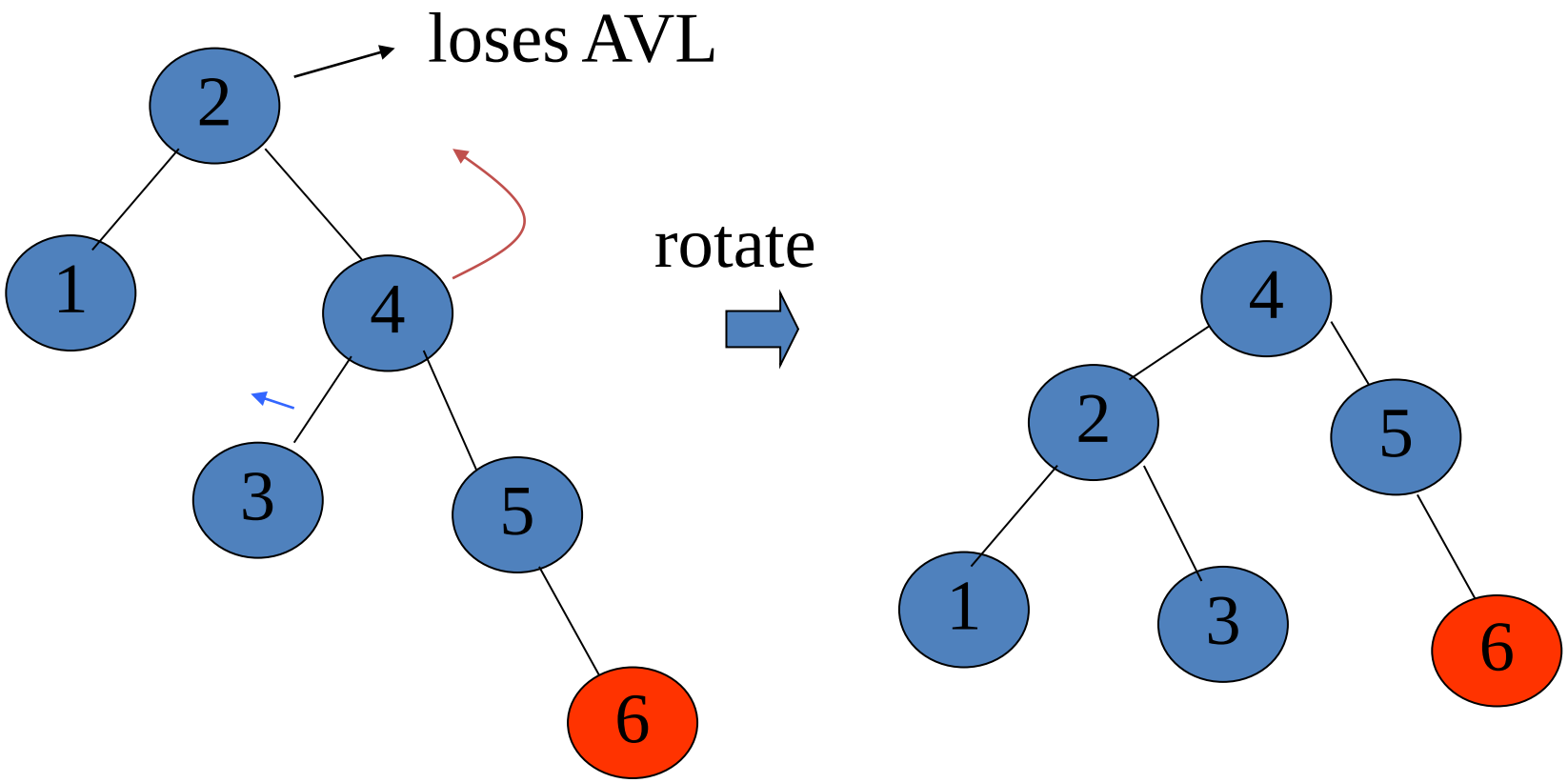


loses AVL



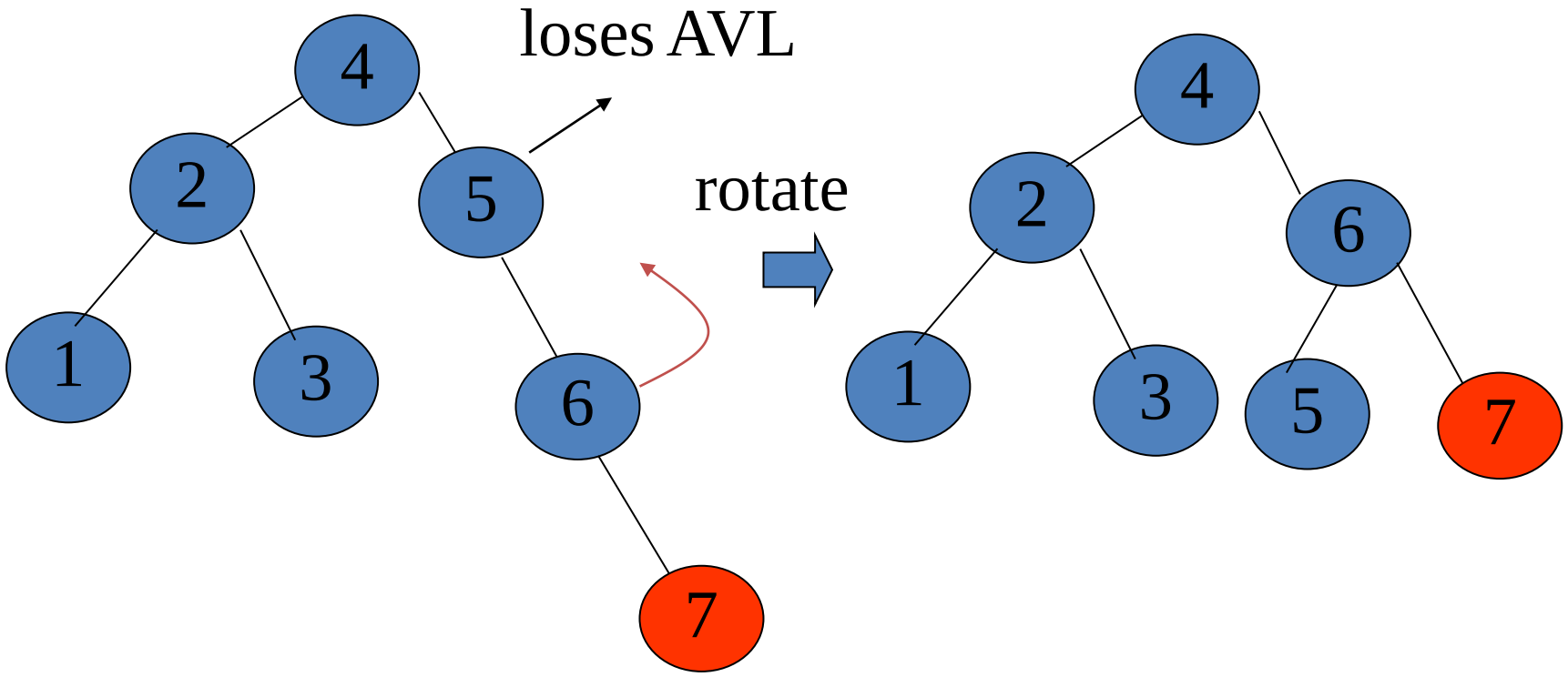
To be continued

- Then we insert 6



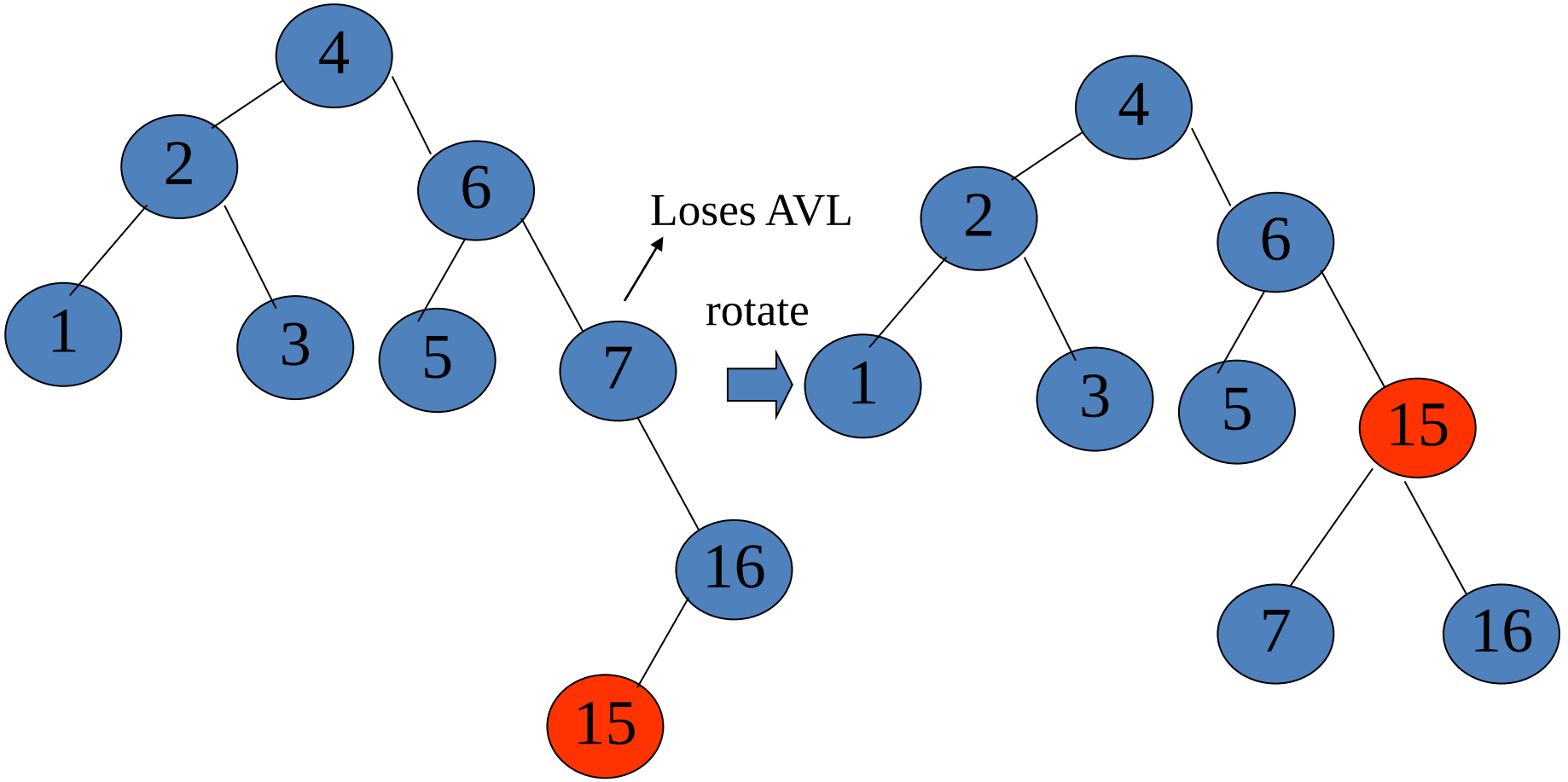
To be continued

Then we insert 7

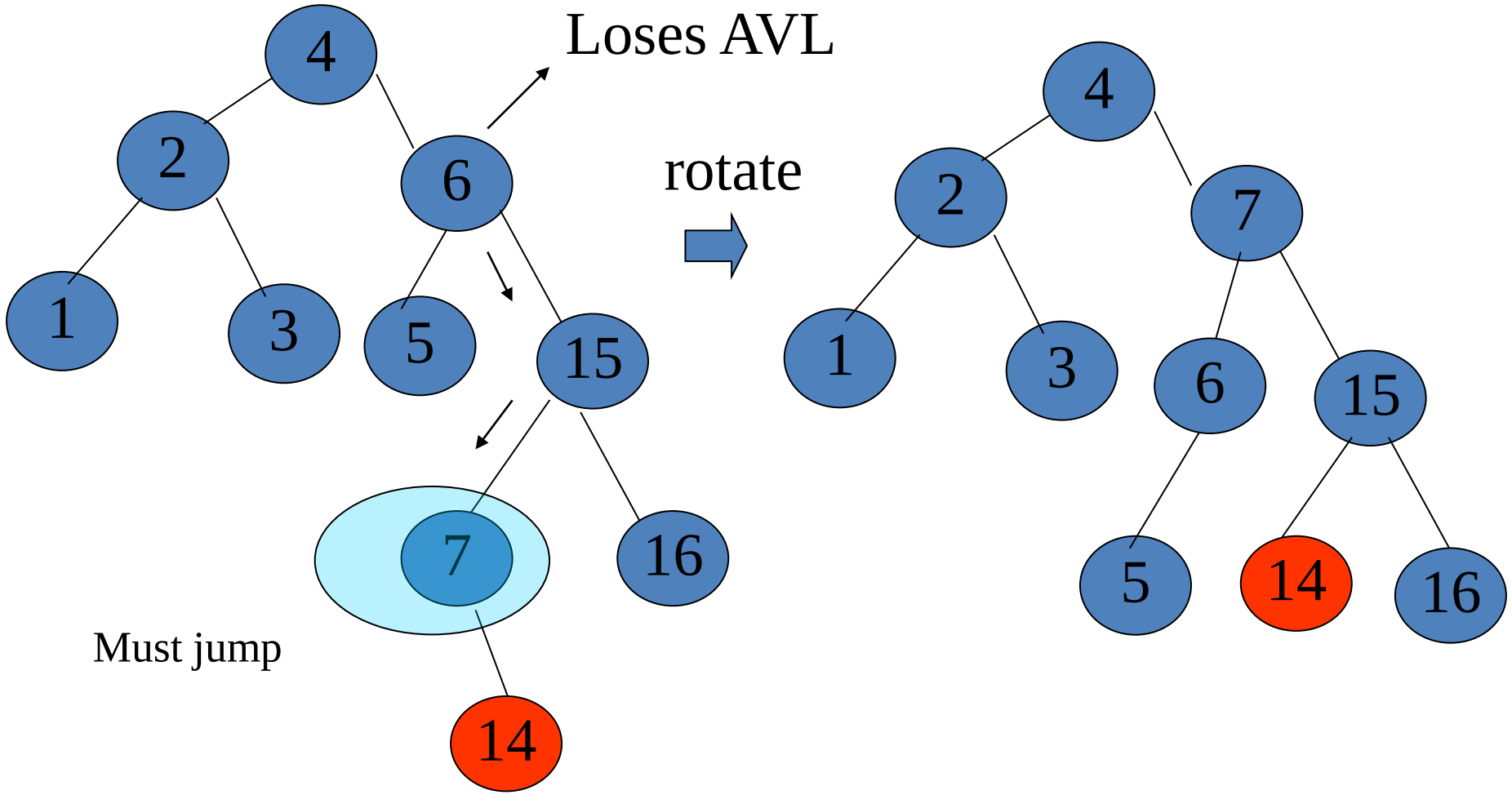


example

- Insert 16,15: loses AVL when inserting 15



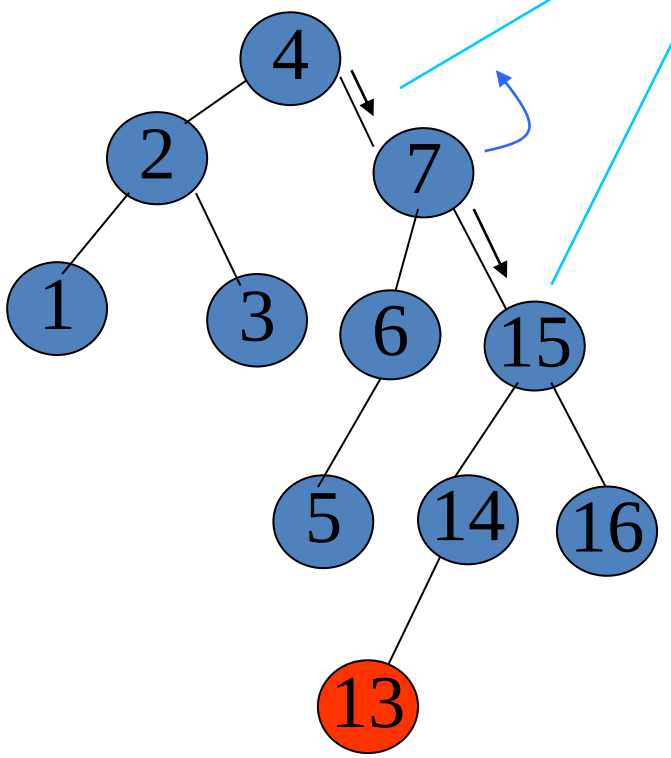
- Insert 14



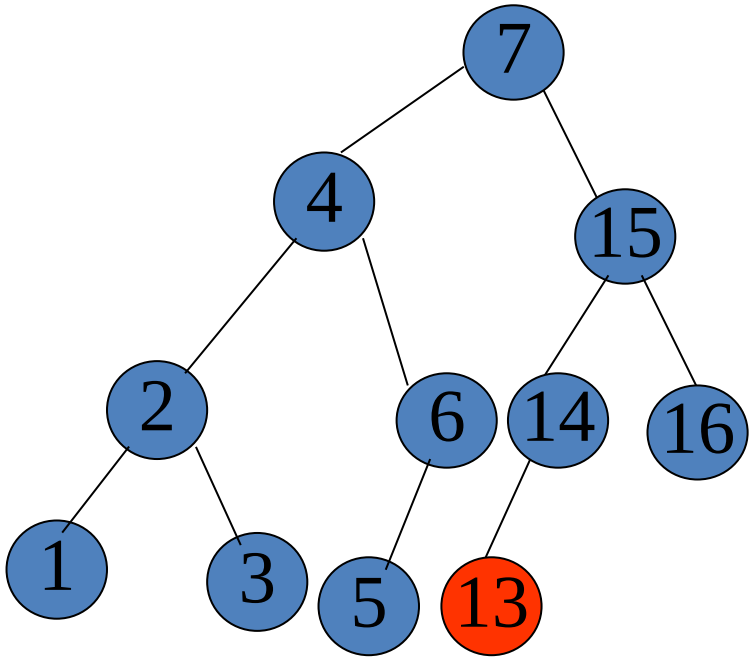
- Insert 13

Loses AVL

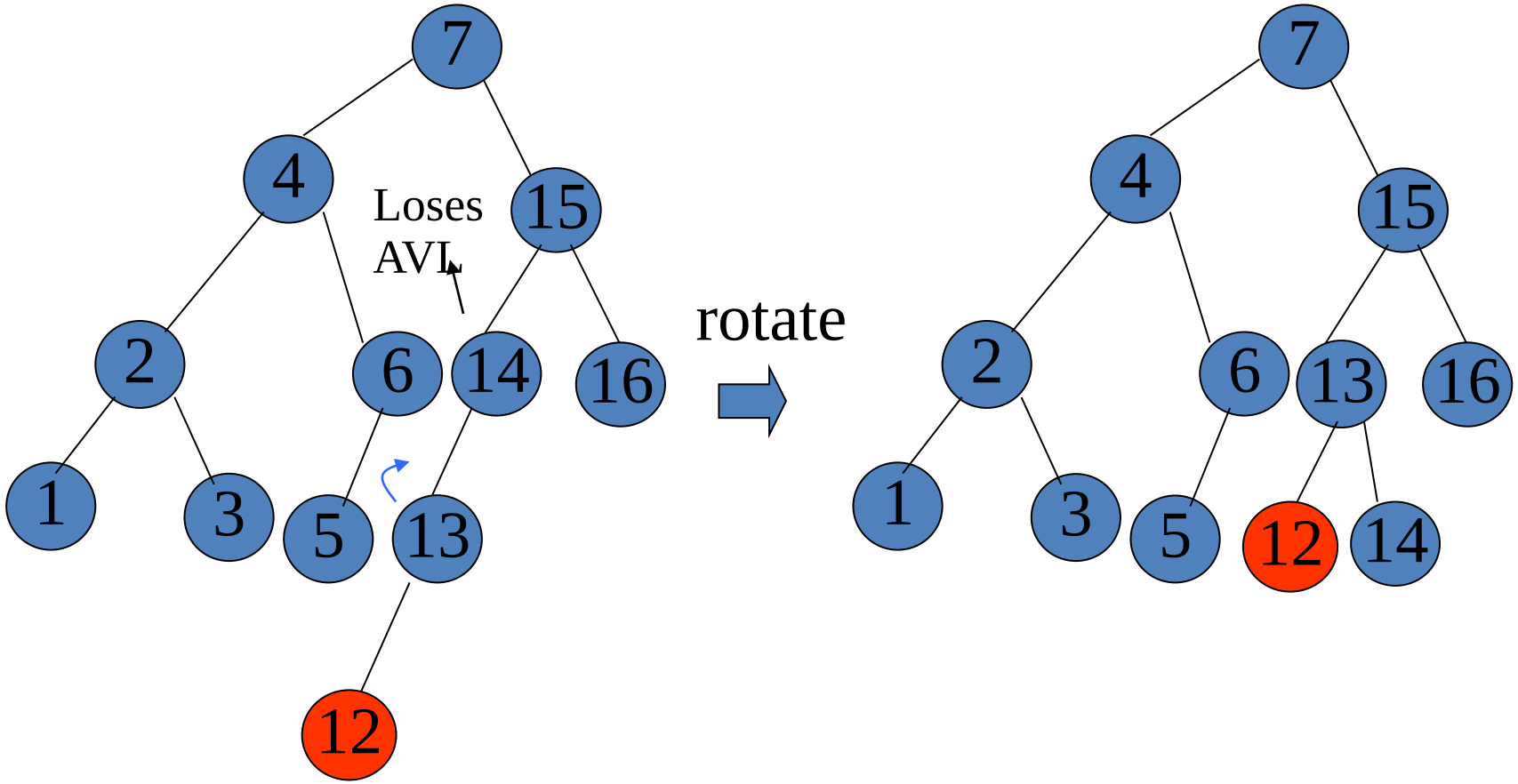
Two right moves: single rotation



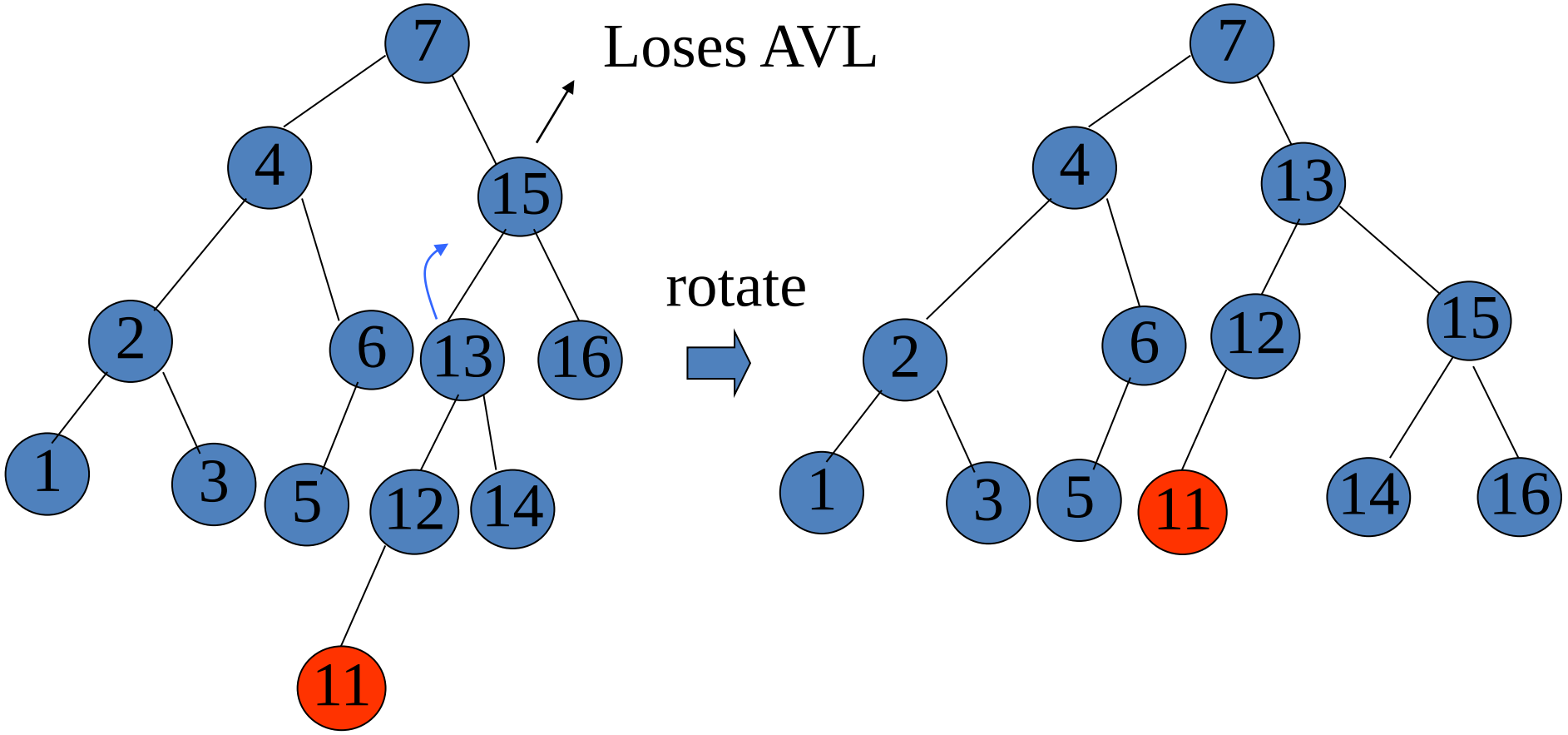
rotate



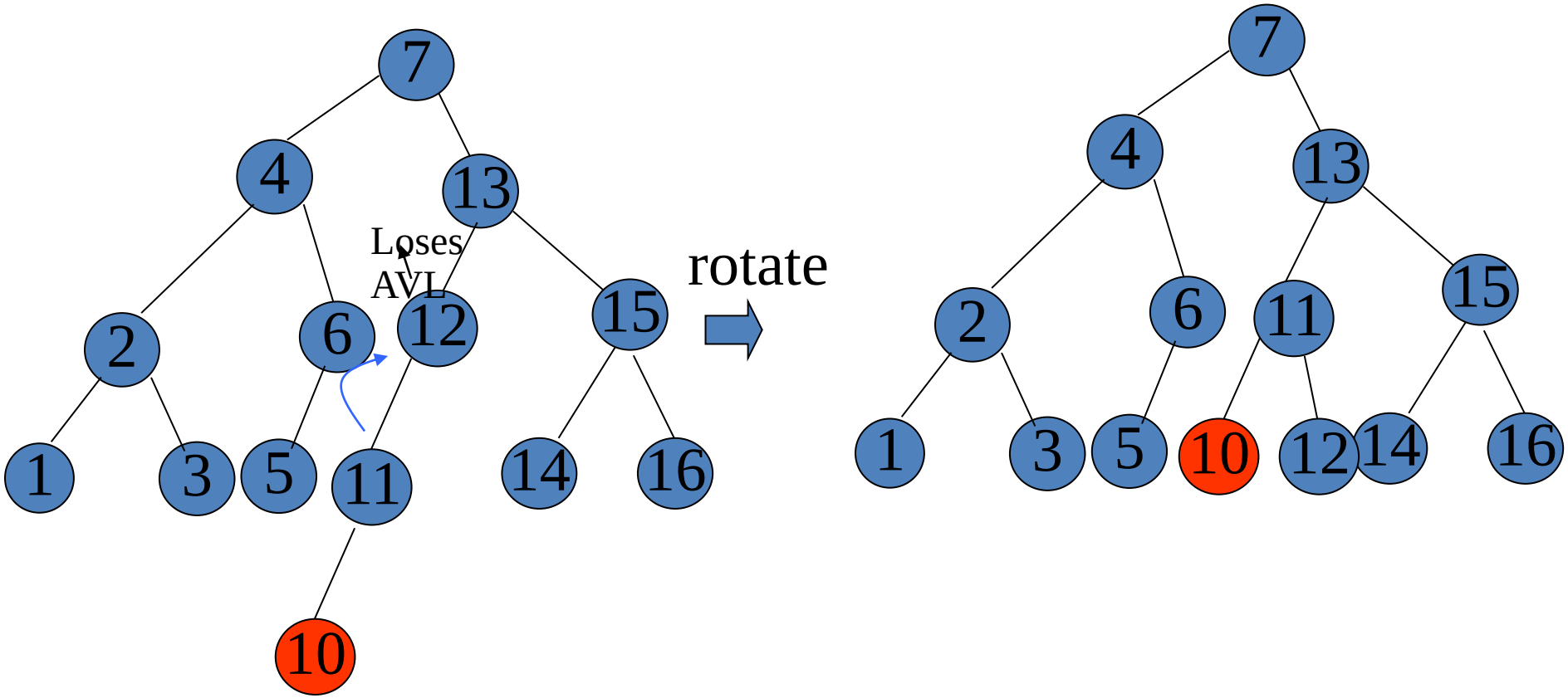
- Insert 12: this is an obvious single rotation



- Insert 11: another single rotation



- Insert 10: another single rotation



Deletion

- Delete as per BST
- Validate the AVL property from parent node to root node and fix by rotations, if required.

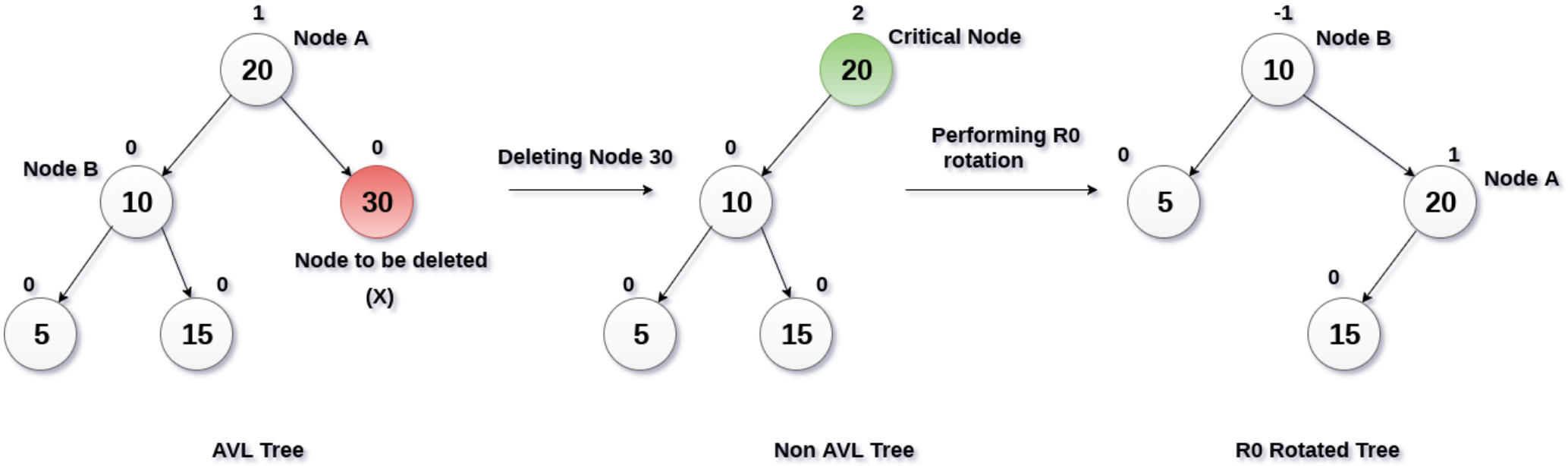


Image: Java Tutorial Point