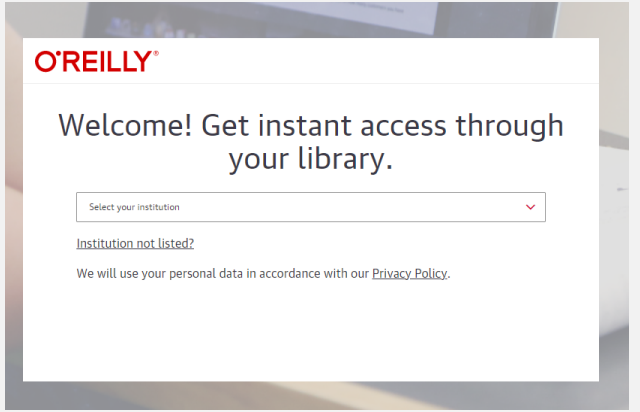
Course Logistics:

Have you checked your exam?

You have free access to O'Reilly Library
https://www.oreilly.com/library-access/?orpq



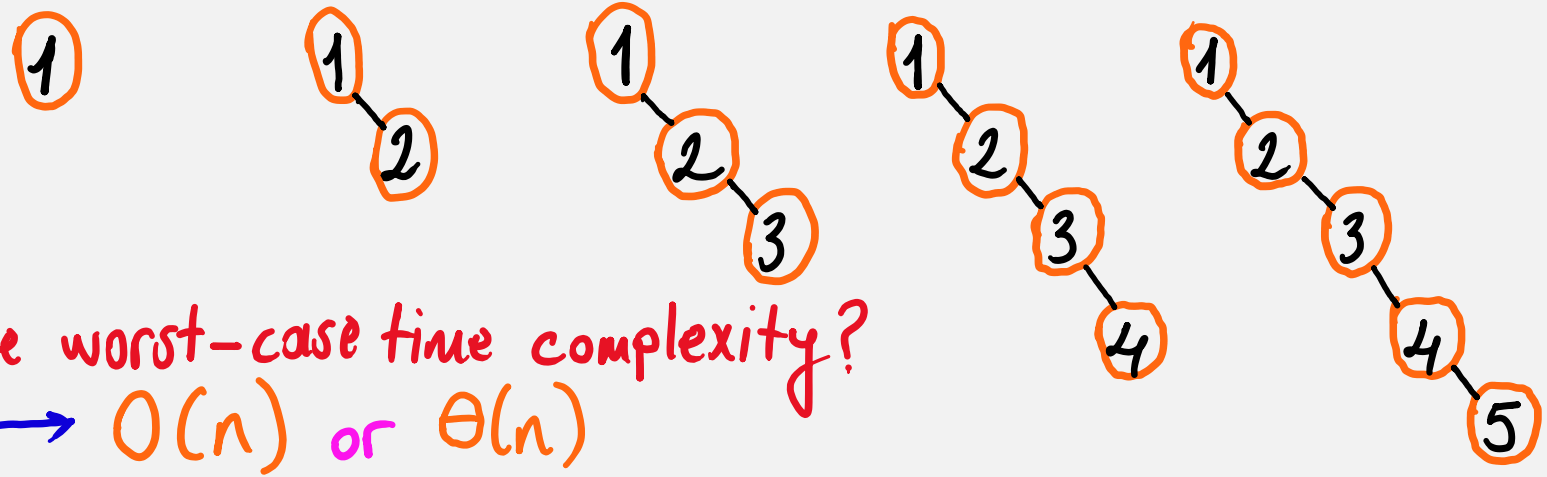
Good visualization tool:
https://www.cs.usfca.edu/~galles/visualization/Algorithms.html

Binary Search Trees

How unbalanced could the tree be?

↳ already sorted input, reverse-sorted, nearly sorted ...

1 2 3 4 5 →



- What could be the worst-case time complexity?

  O(depth) ⟶ O(n) or Θ(n)

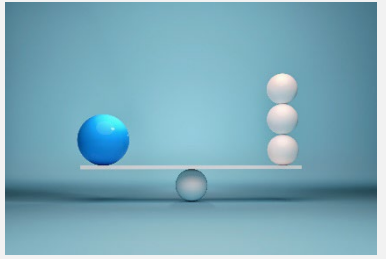
- What would be ideal? O(log n) or Θ(log n)

AVL Trees

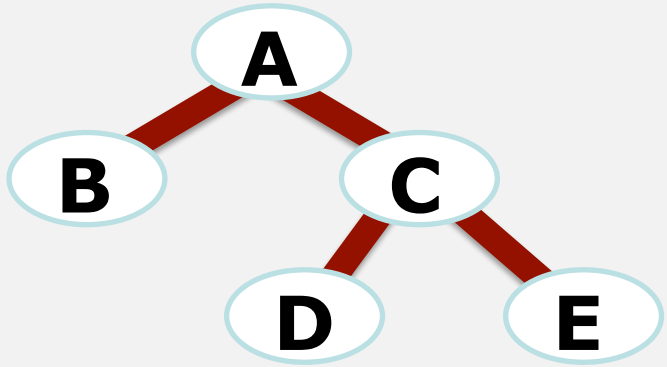
- A data structure, invented by Adelson-Velsky and Landis in 1962 to balance Binary Search Trees.

- Essentially AVL trees are BSTs with one extra invariant (rule).
  - ↳ tree is always balanced.

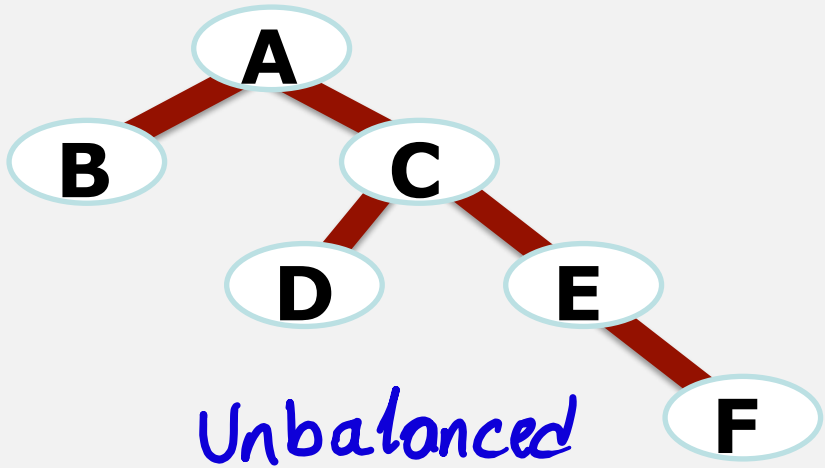
- What does balanced mean?
  - ↳ What is the criteria for that?



Which one is unbalanced? Who decides that?

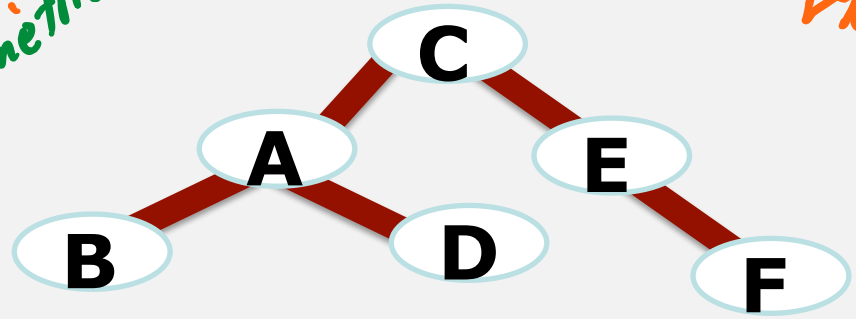


Balanced



Unbalanced

Is it a symmetry thing?
None of them are symmetric.

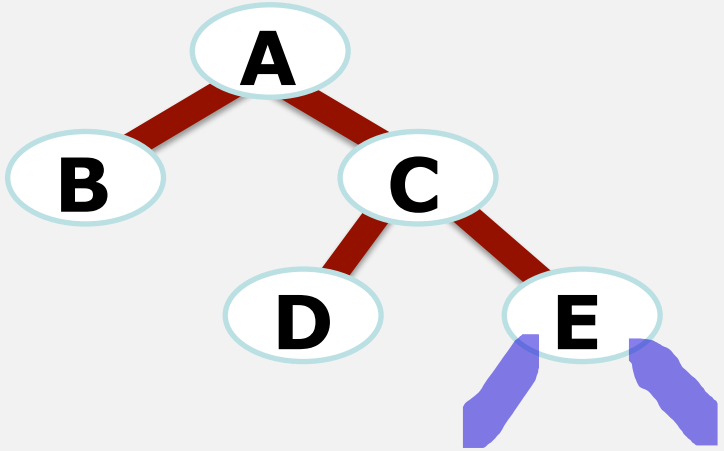


Balanced

Visually pleasing?
To whom?!

We need a deterministic approach!

Height



- Height of an empty node
  - ↳ non-existing child of node E → −1

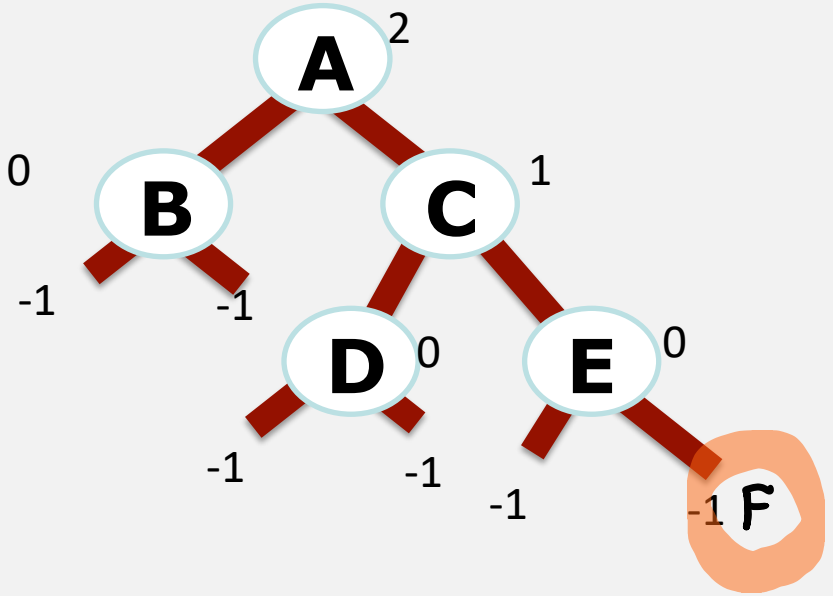
- Height of a node
  - ↳ longest shortest path to a descendant leaf

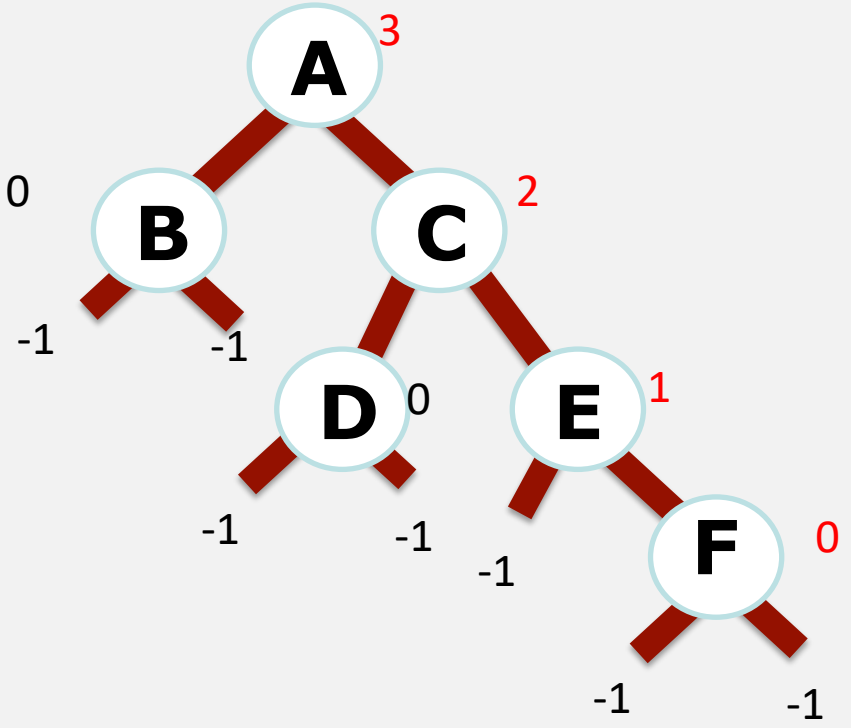
example:
- Node E → 0
- Node C → 1
- Node A → 2
- Node B → 0

- Hight of a tree
  - ↳ The leaf which is furthest away from the root.

example: starting from E or D to A → 2

Computing Height:





- Recursion:

height(node) = max(heigh of children) + 1;

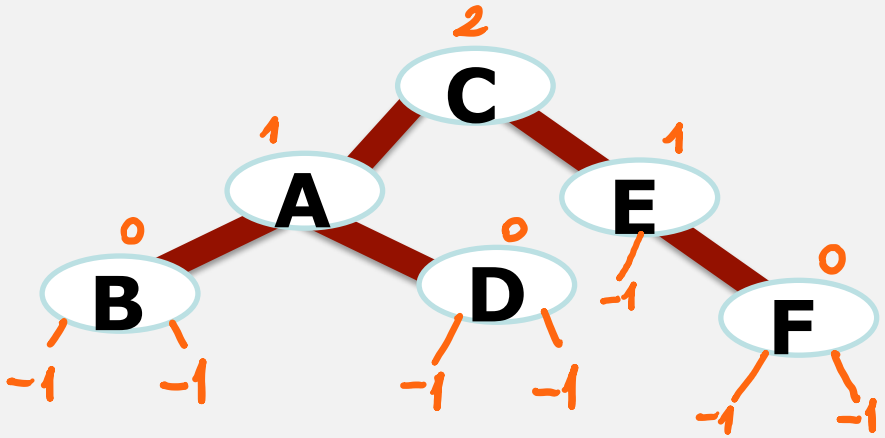
How can we maintain the height?

example: Let's add F

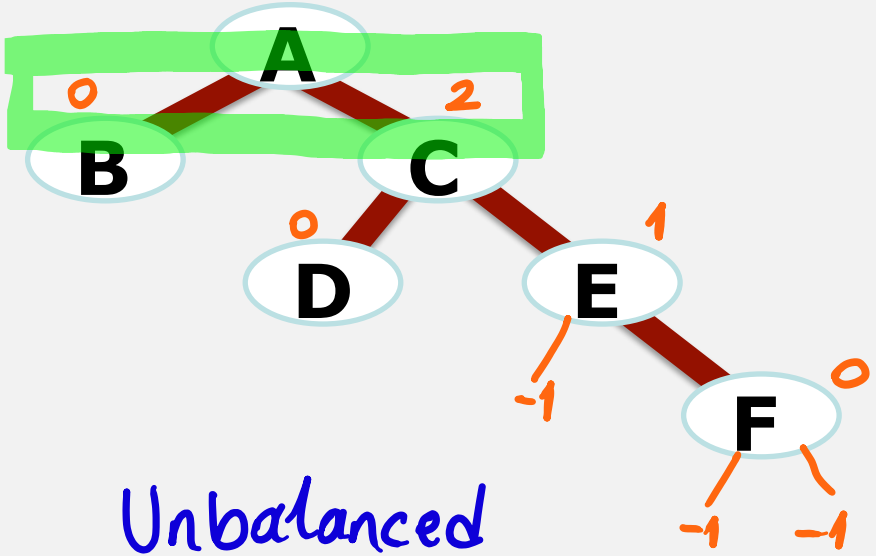
- Only nodes in the path to root are affected.
- If we inserted recursively, we can update the tree free.

Which one is unbalanced? Who decides that?

Height difference between siblings should not be more than 2



Balanced



Unbalanced

What kind of tree traversal algorithm do we need?

Pre-order

??

In-order

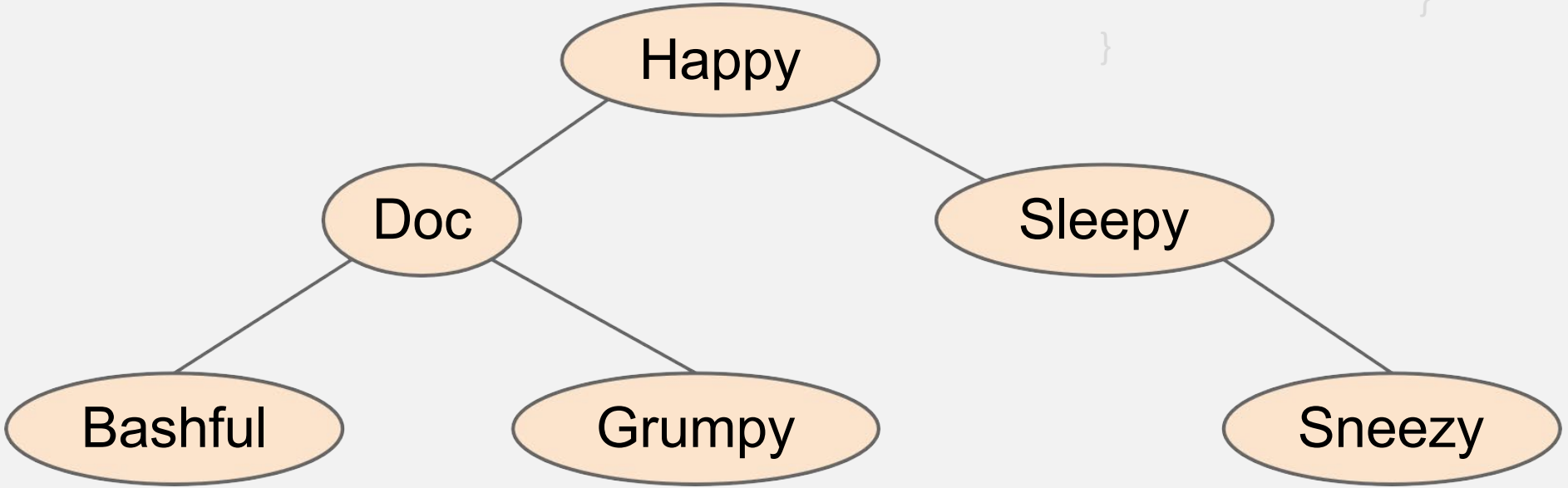
??

Post-order

??

Tree: 6 of the 7 Dwarves

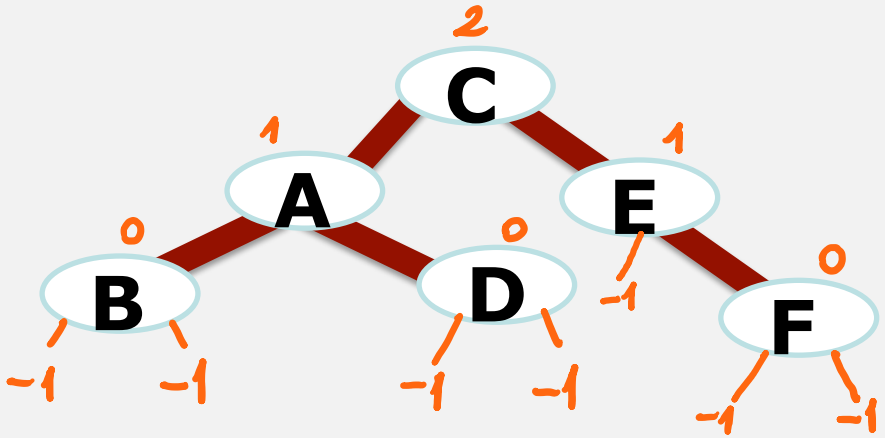
```
void traverse(struct node* root)
{
if (root != NULL) {
        traverse(root->left);
        traverse(root->right);
        printf("%s", root->key);
                }
}
```



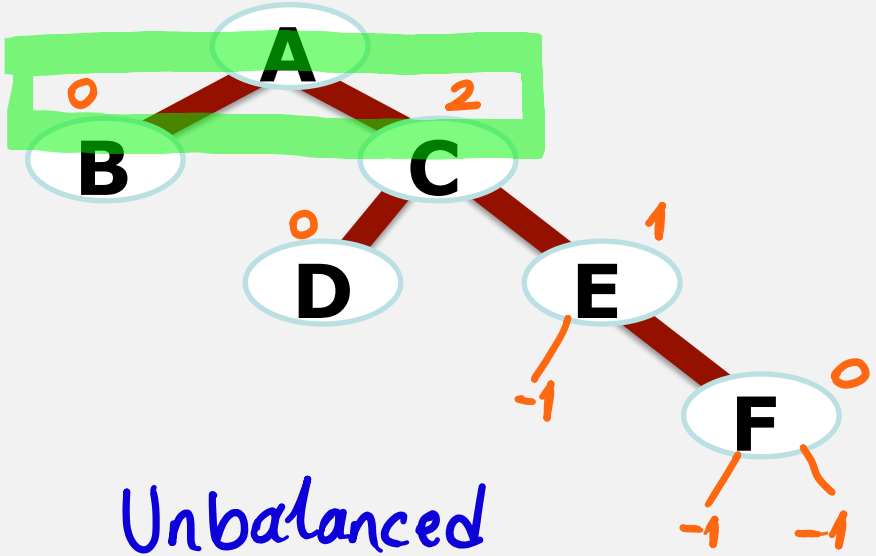
Post-order: Bashful, Grumpy, Doc, Sneezy, Sleepy, Happy

Which one is unbalanced? Who decides that?

Height difference between siblings should not be more than 2



Balanced



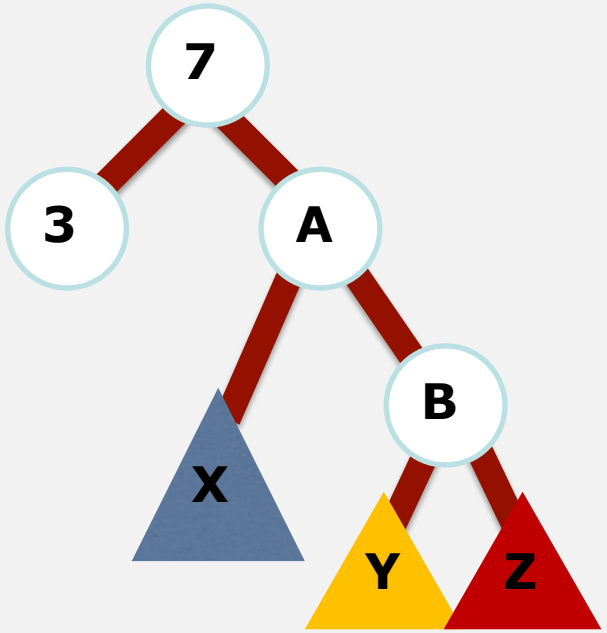
Unbalanced

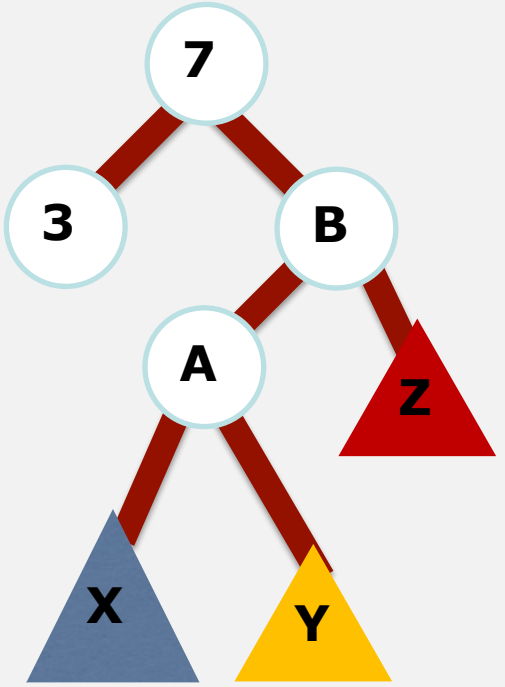
What kind of tree traversal algorithm do we need?

Post-order traversal is what we need here!

Rotations:

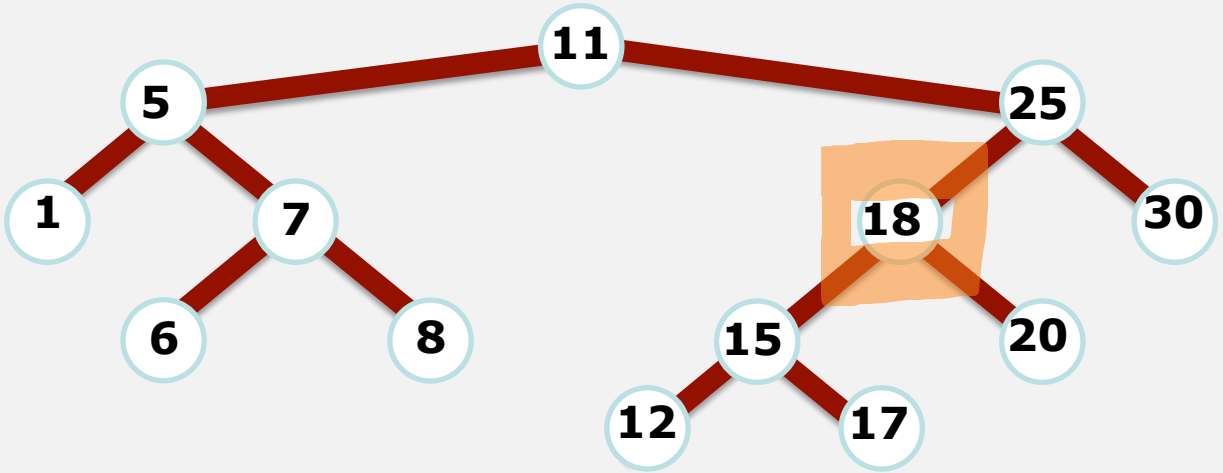
Let's rotate A to the left



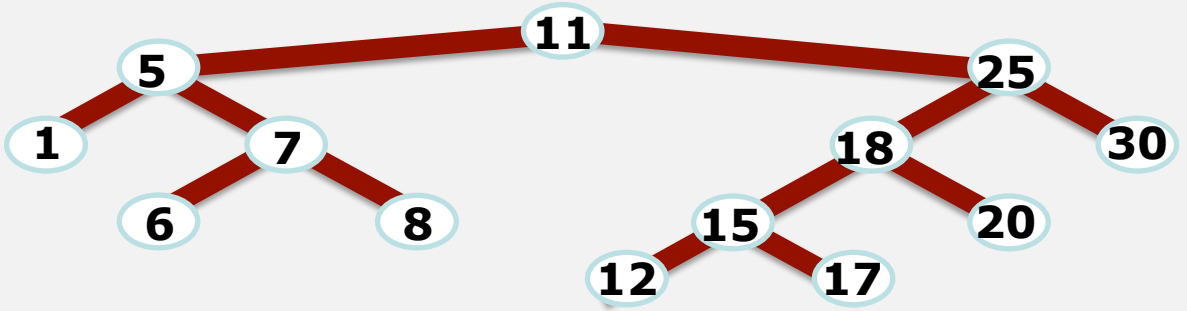


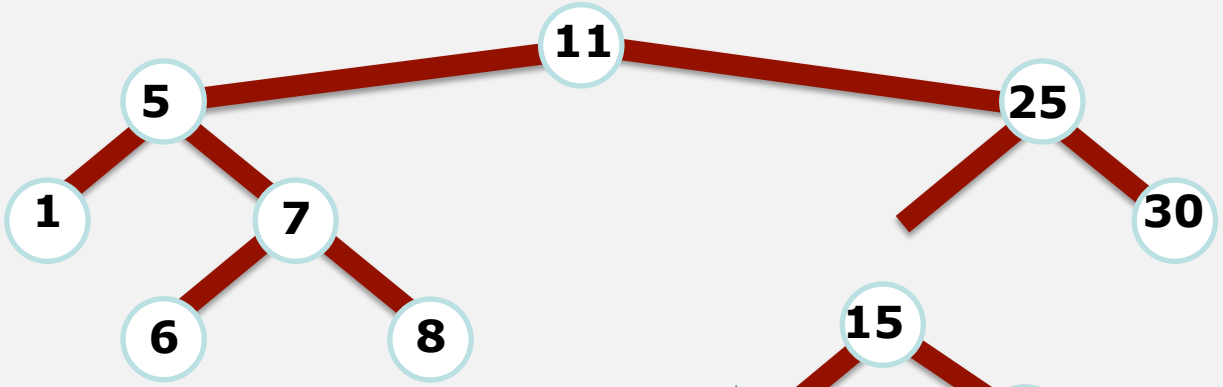
$$X<a<Y<B<Z$$

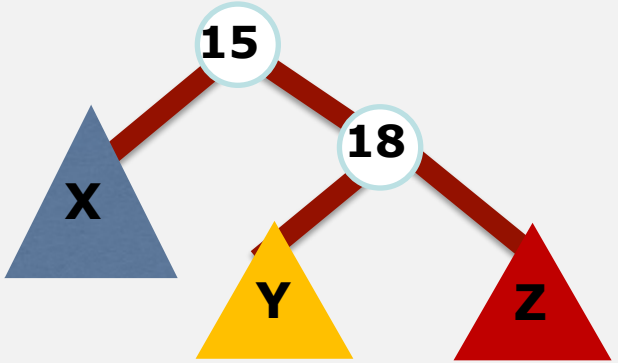
Exercise : Let's rotate 18 right



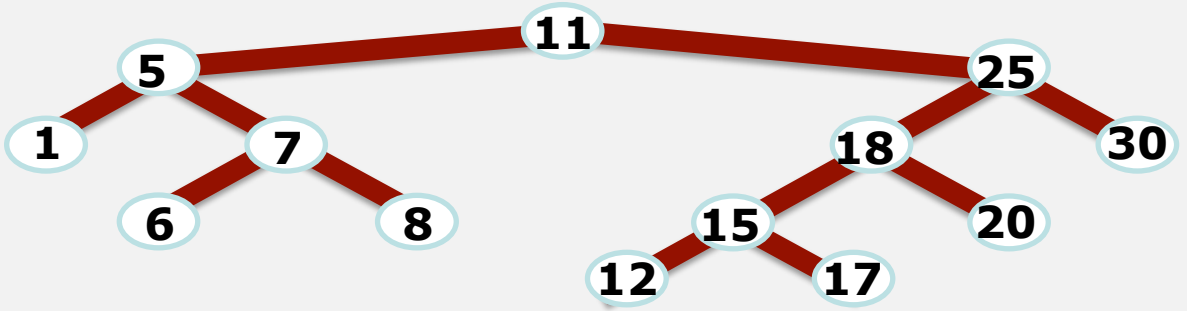
Exercise : Let's rotate 18 right

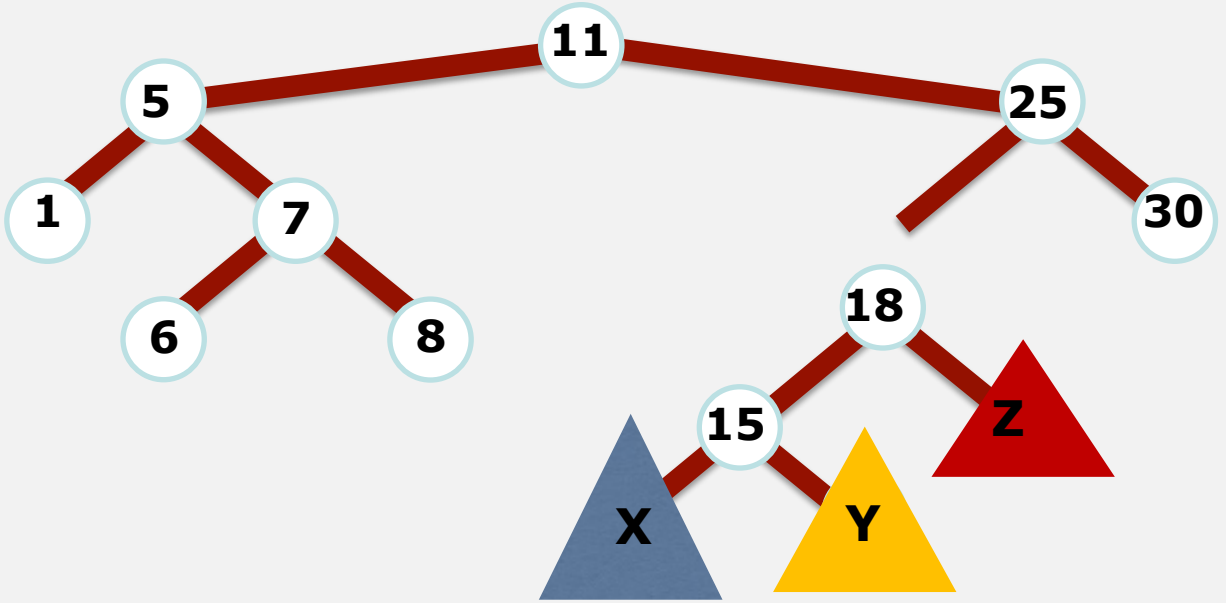




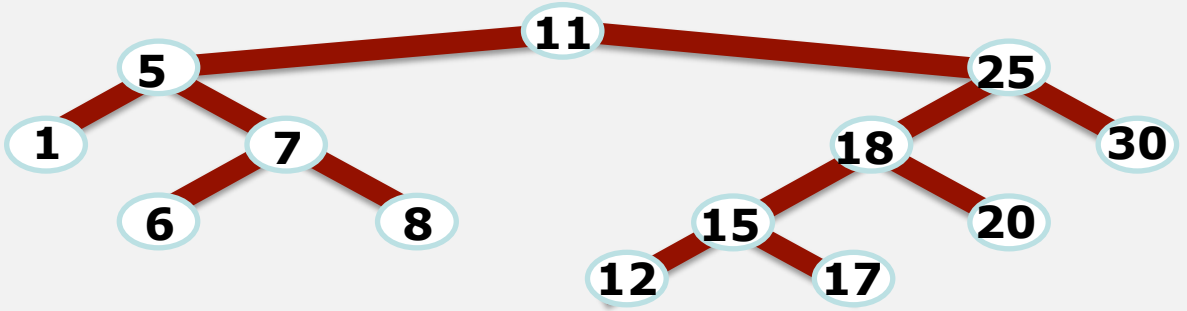


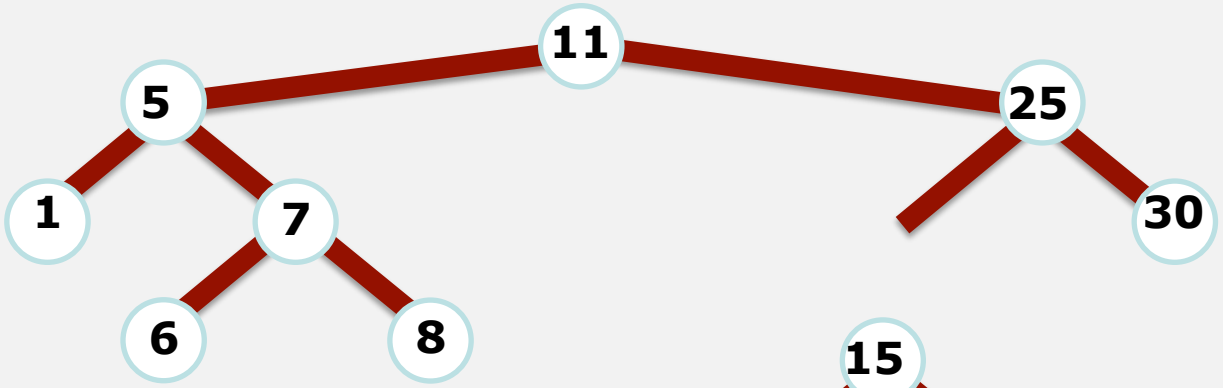
Exercise : Let's rotate 18 right

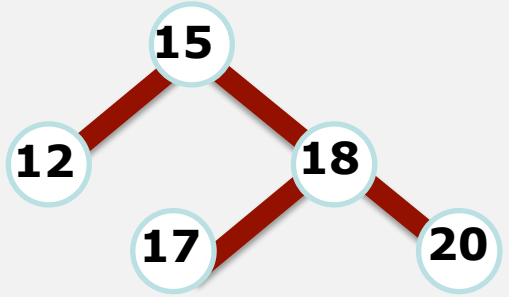




Exercise : Let's rotate 18 right



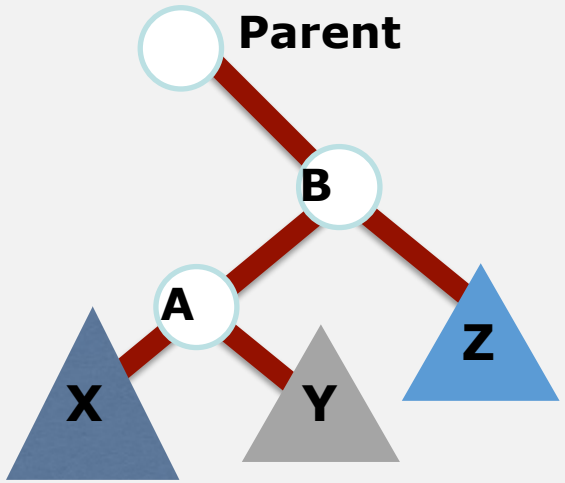


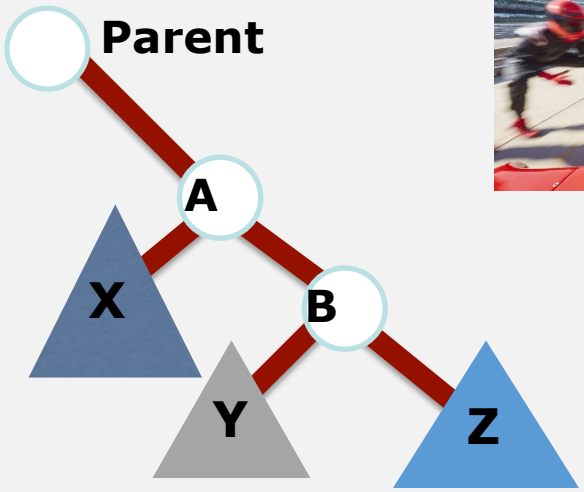


Implementation:

How fast are rotations?

O(1) } Super fast



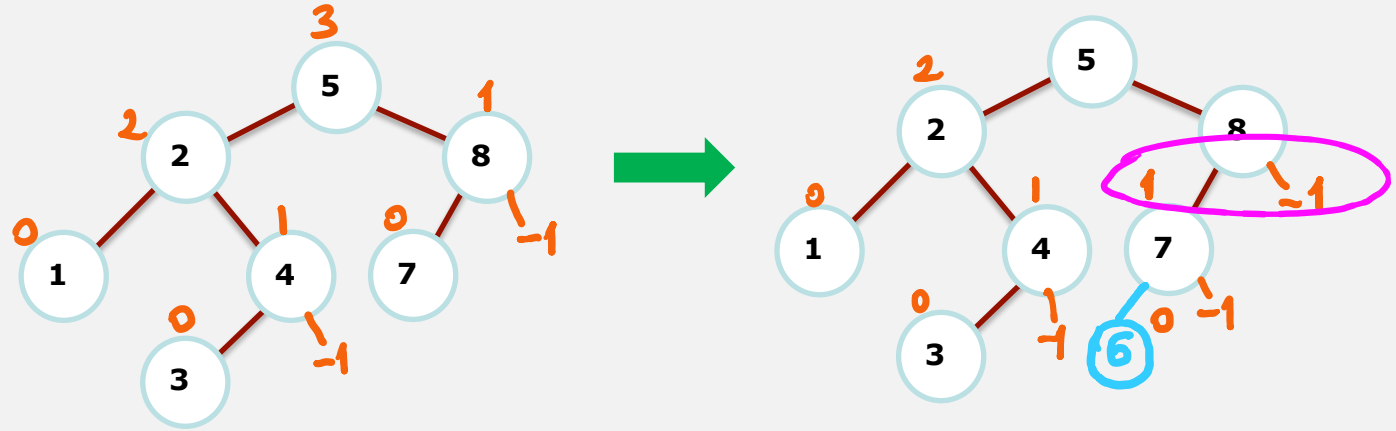



```
void rightRotateRightChild(avlNode *parent){
    avlNode *B;
    avlNode *A;

    B = parent->right;
    A = B->left;
    parent->right=A;          //Right child of parent is A
    B->left = A->right;       //Left child of B is right child of A
    A->right= B;              //A is parent of B now

}
```

Insert 6 into the tree :

↳ Let's check if the tree is a valid BST

↳ Then, let's check if the tree satisfies the AVL invariant



- We need to ask some questions:

  Which node is responsible for this? 8

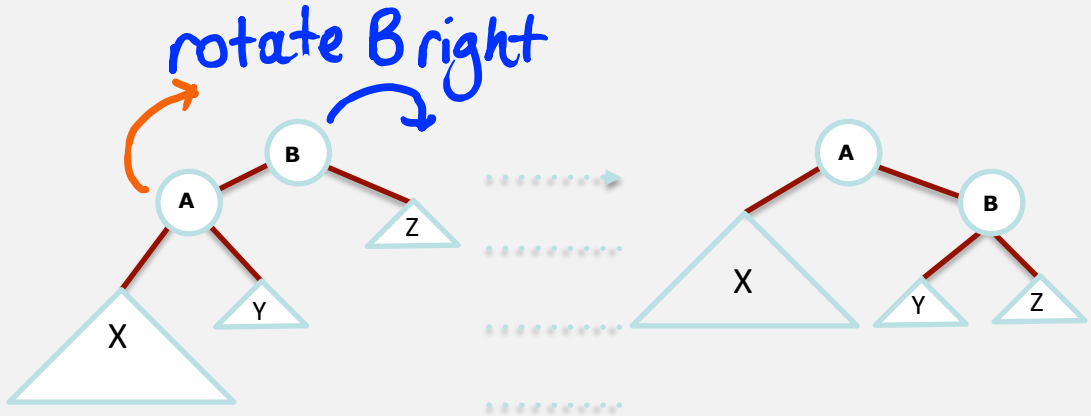
  Explain what happened from the eyes of 8?

  all possibilities : 8's left child's left subtree → single rotation

  8's left child's right subtree → double rotation

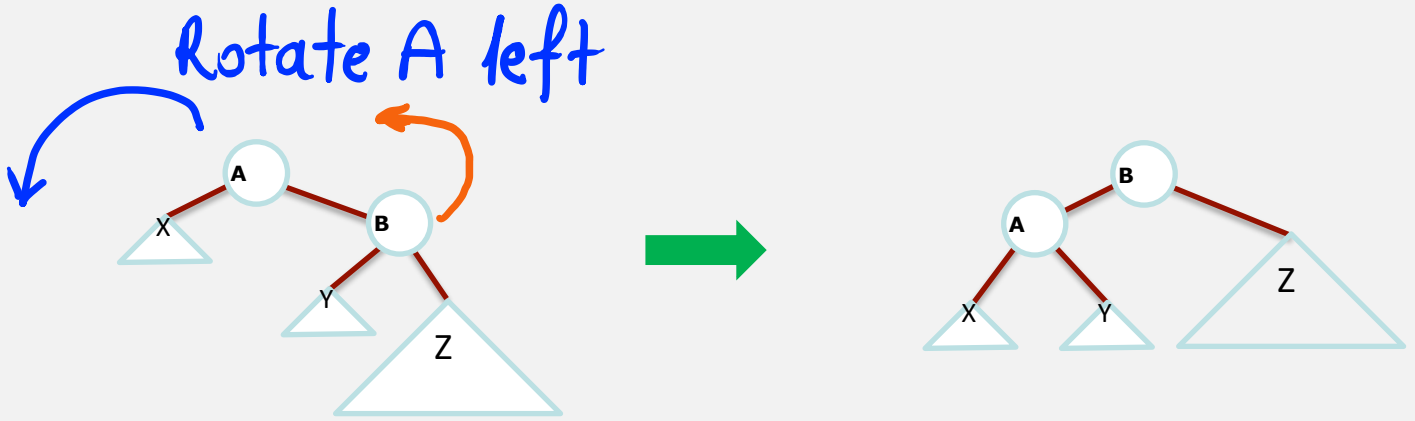
Insertion at Left Child's Left Subtree :

- Single rotation is sufficient



Insertion at Right Child's Right Subtree :

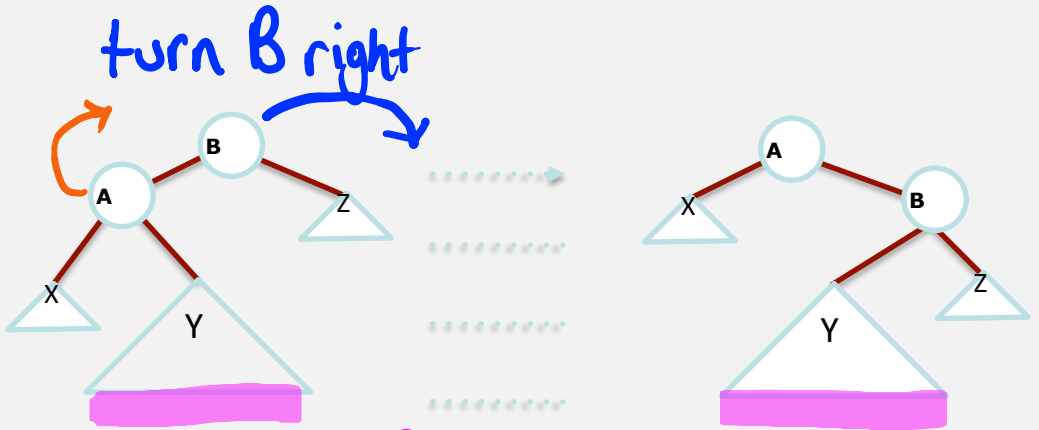
- Single rotation is sufficient



Insertion at Left Child's Right Subtree:

Single rotation will not be enough. Double rotation is needed.

Example: Assume some node is added to Y subtree and let Node B is responsible for this problem.



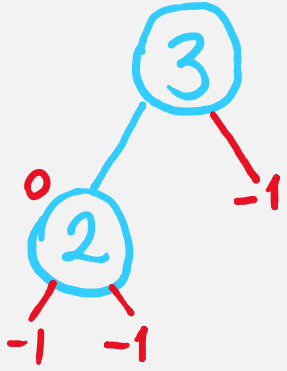
Problem exists

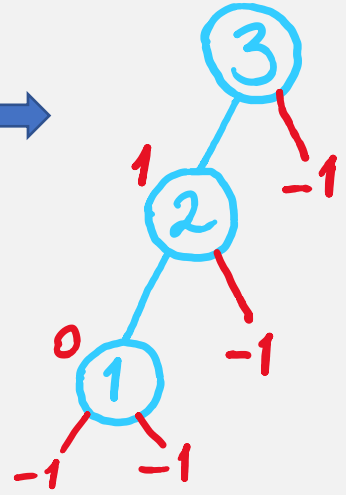
- Same problem would also occur if the insertion type was right child's left subtree because of the symmetry.

- The task is too difficult for B. Maybe B should seek help from A first. After A rotates to the lighter side, B can rotate right then.

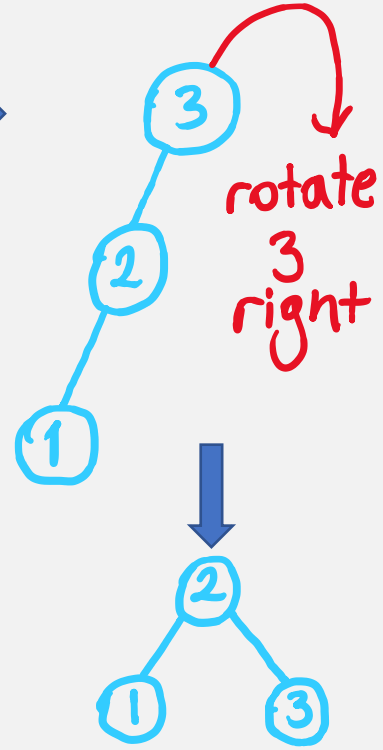
- Exercise: Insert nodes into an AVL tree : 3, 2, 1, 4, 5, 6, 7

- Steps:
  - Insert node into a BST → 
  - check height difference of the children → 

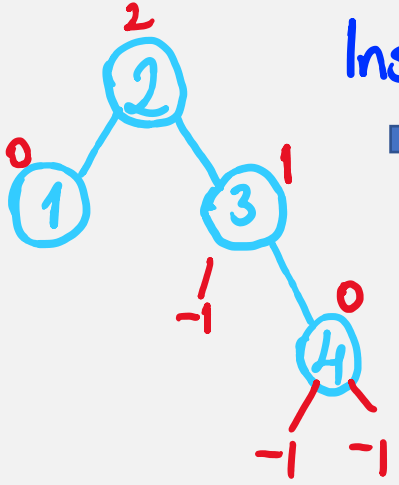




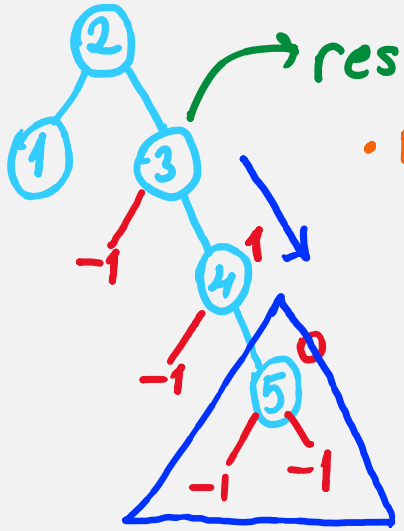
- Steps continued
  - find the responsible parent
  - analyze the type of insertion (left-right, left-left)??
  - if the parent needs a single rotation, rotate parent to the opposite side of insertion



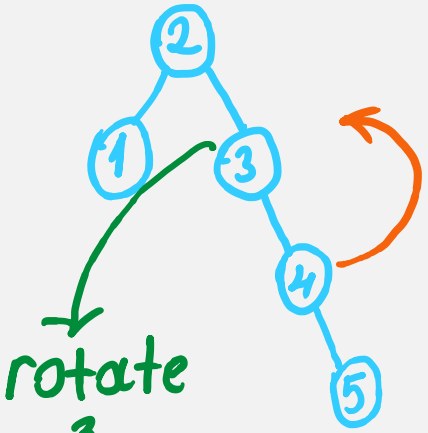
• Exercise: Insert nodes into an AVL tree : 3, 2, 1, 4, 5, 6, 7



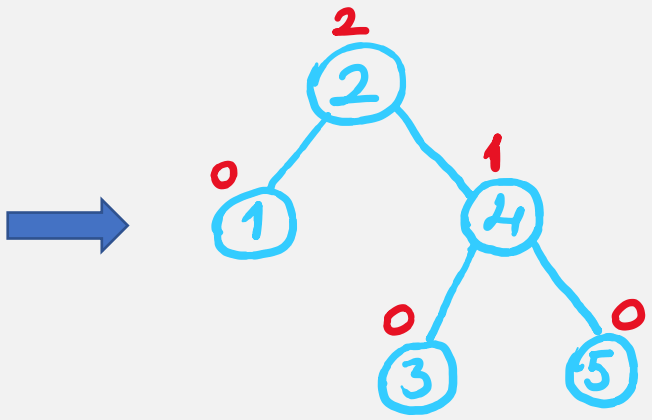
Insert 5



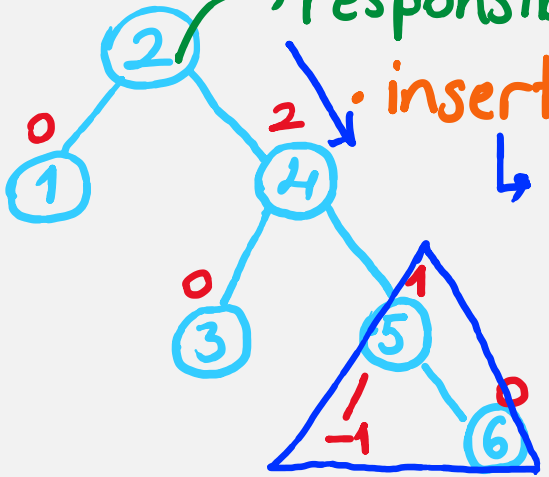
responsible parent
· insertion type
↳ right child's right subtree



rotate 3 left

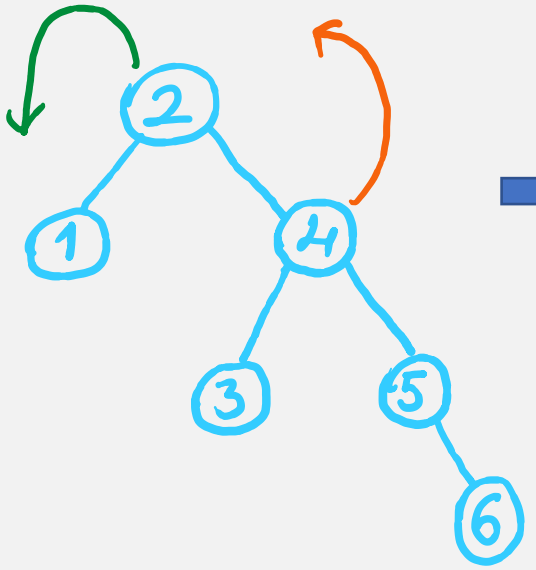


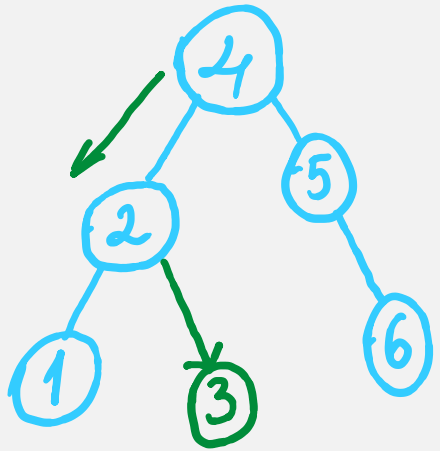
Insert 6



responsible parent
· insertion type
↳ right child's right subtree
↳ Rotate 2 left (Single Rotation)

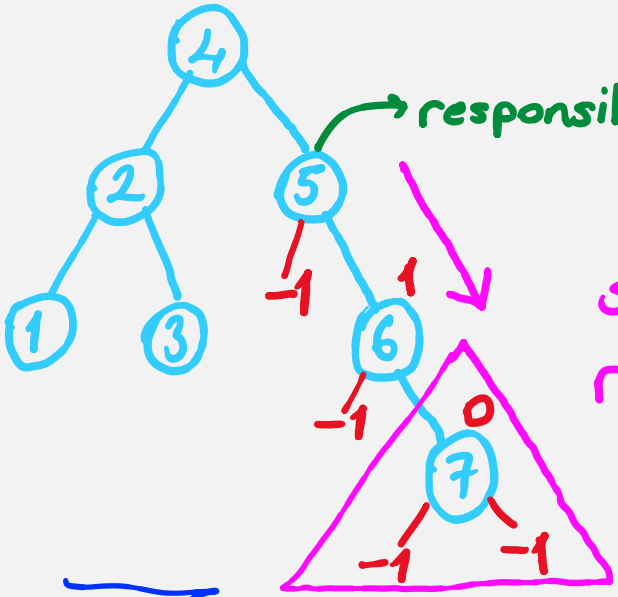
• Exercise: Insert nodes into an AVL tree : 3, 2, 1, 4, 5, 6, 7



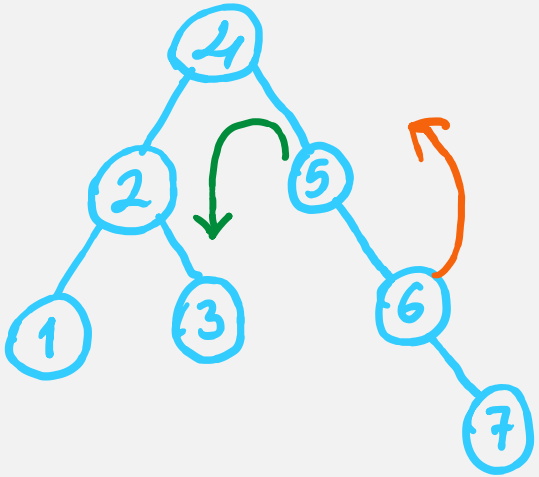


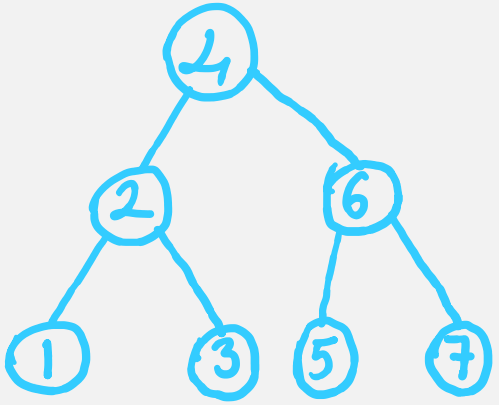
How about 3?

Insert 7



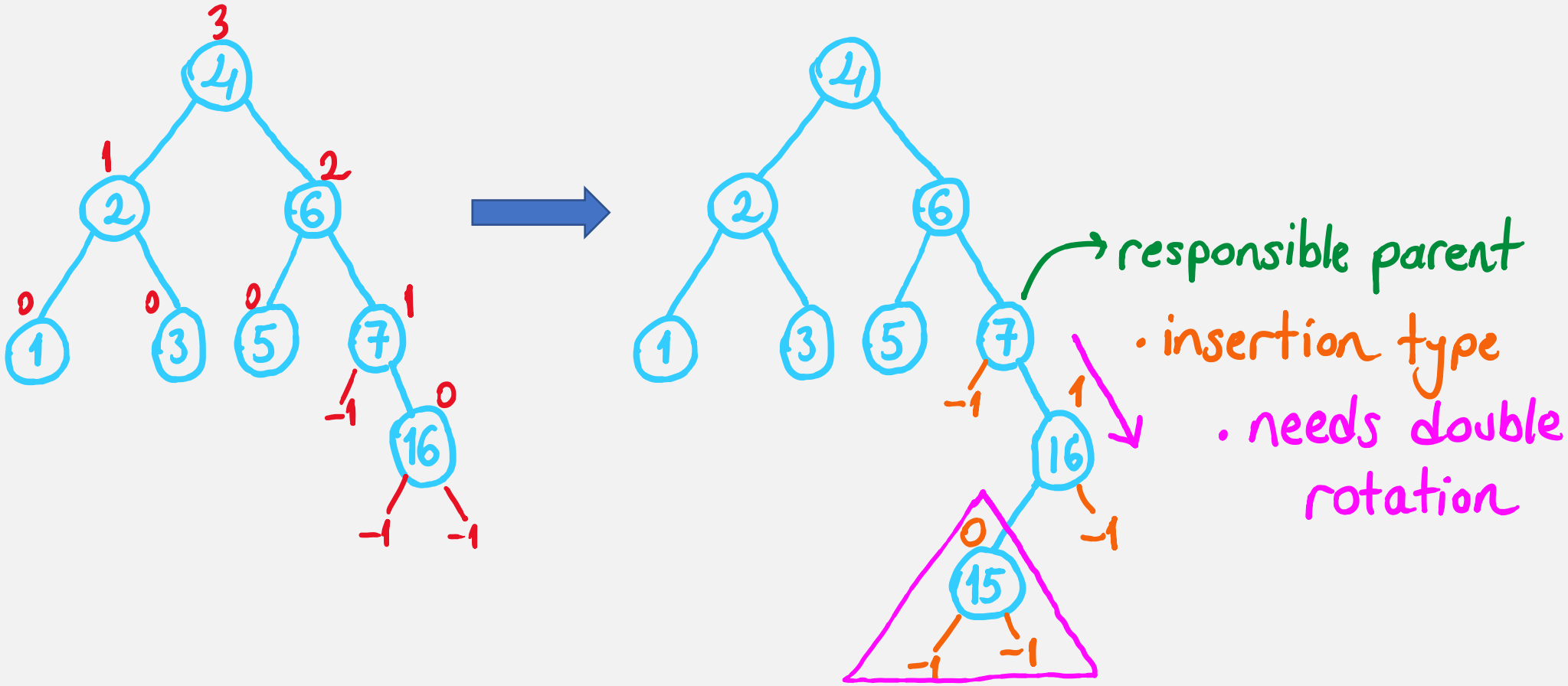
Rotate 5 left



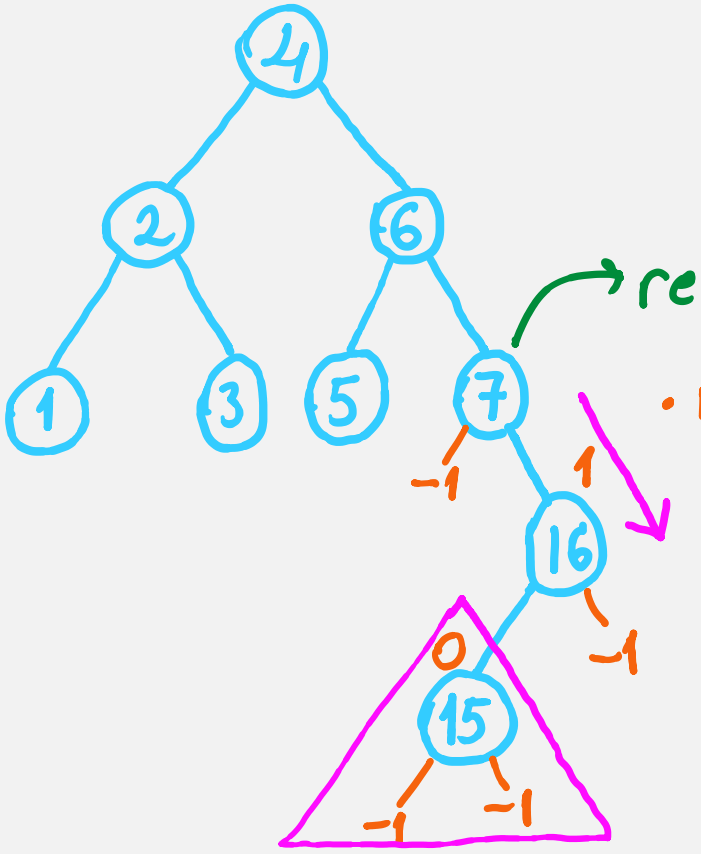


Ideal balance

Exercise: Double Rotations: Add more nodes: 16, 15, 14

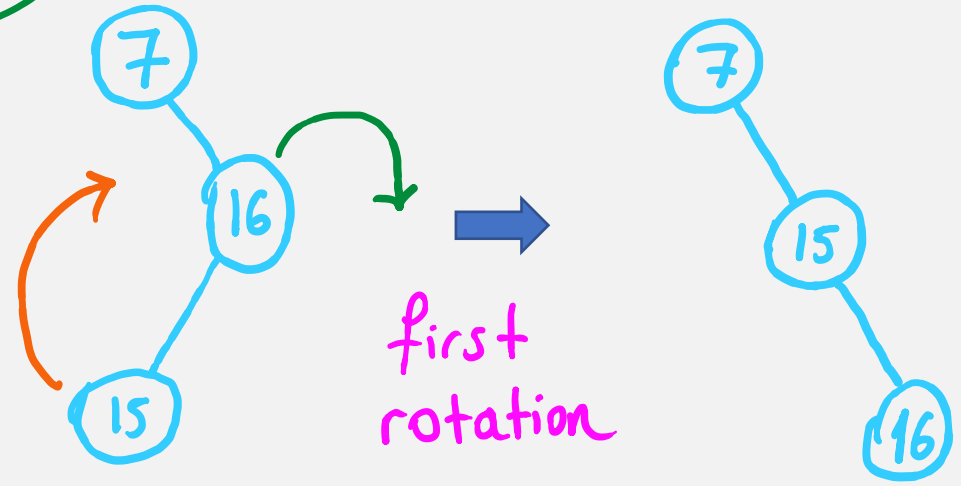


Exercise: Double Rotations: Add more nodes: 16, 15, 14

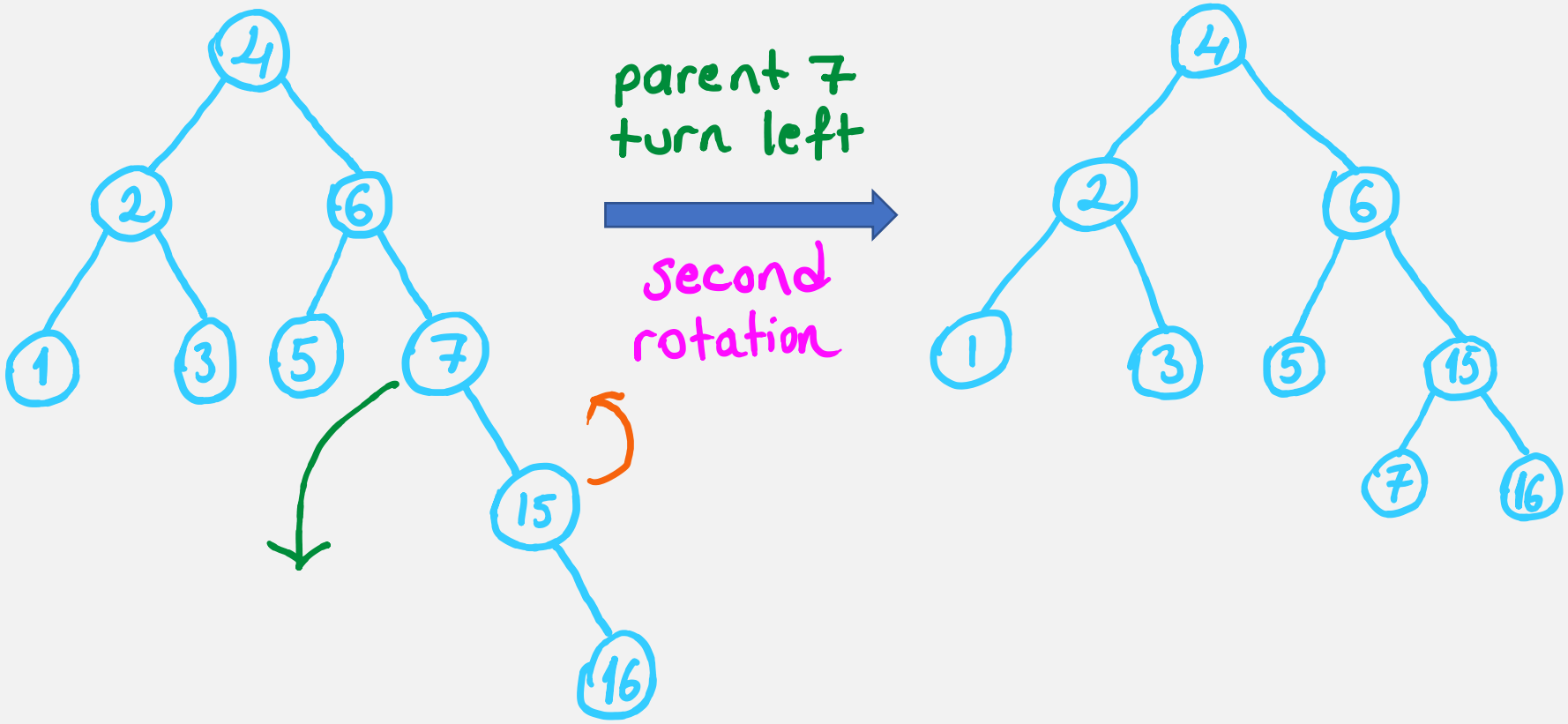


→ responsible parent
- insertion type
- needs double rotation

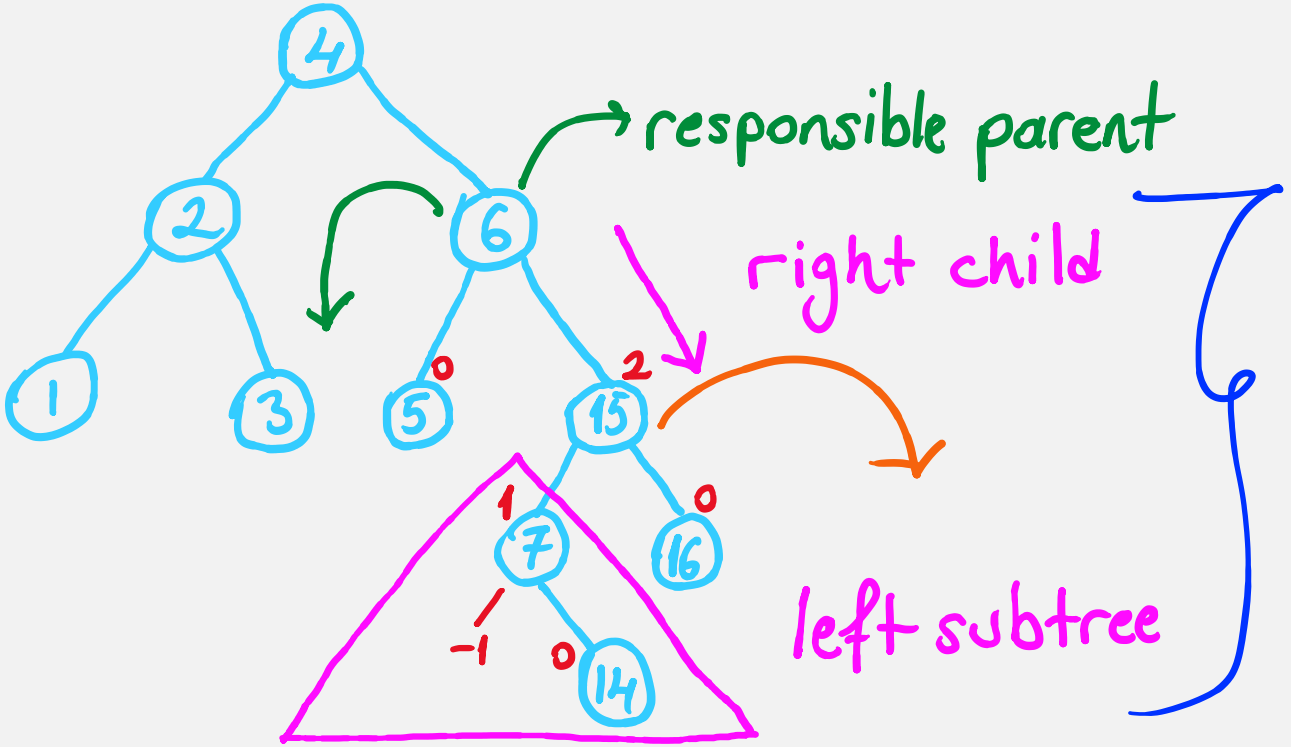
- 7 needs help from the child 16
- 16 should turn right because the insertion was to its left



Exercise: Double Rotations: Add more nodes: 16, 15, 14



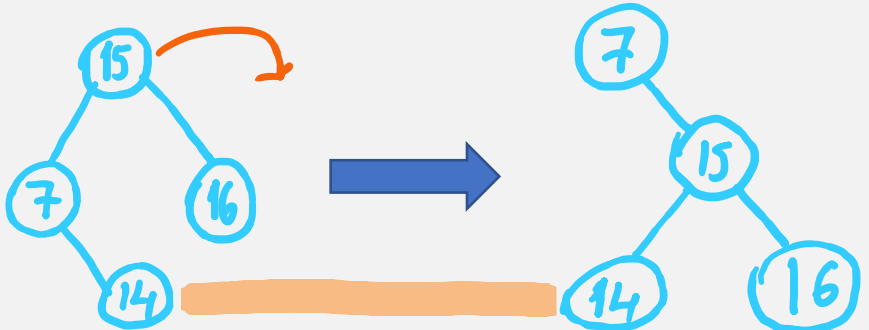
Exercise: Double Rotations: Add more nodes: 16, 15, 14

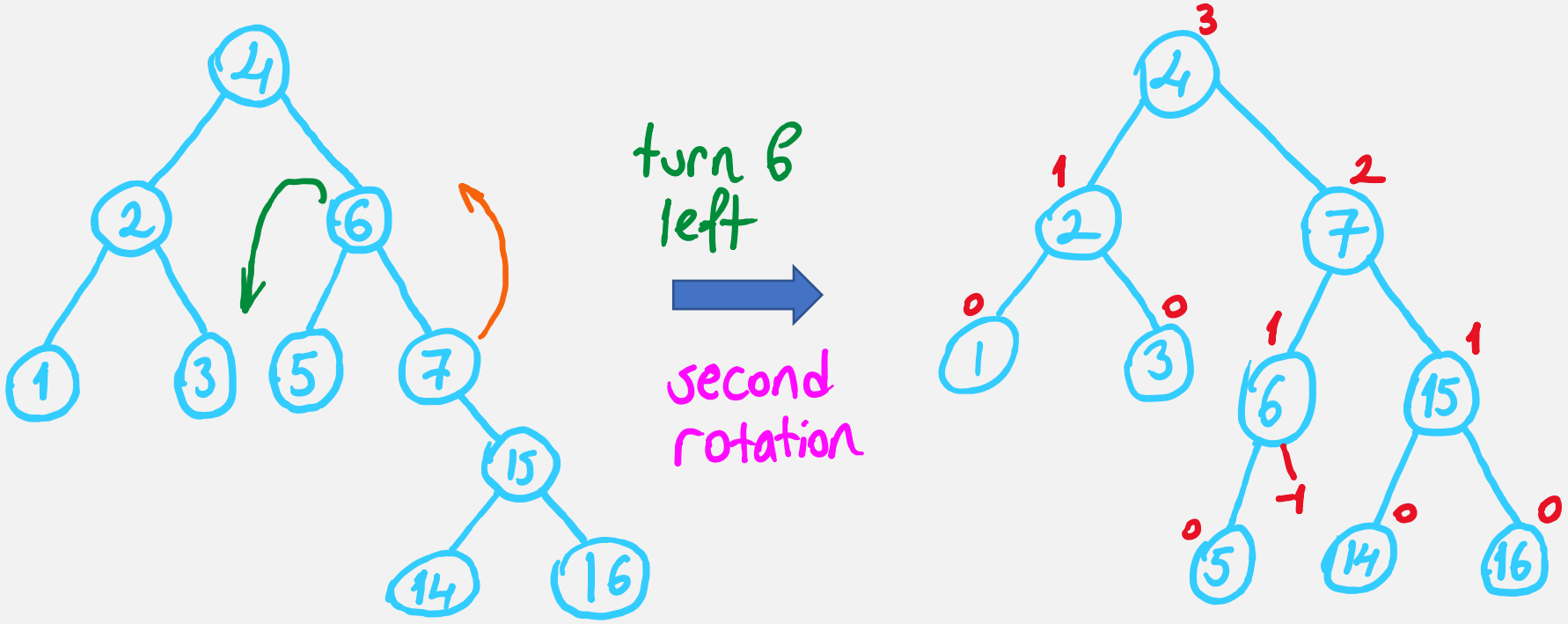


Double rotation

1. 6 gets help from 15
2. 15 turns right
3. 6 turns left

first rotation





Practice Homework: Add 13 to the tree!