COSC2436: AVL Trees

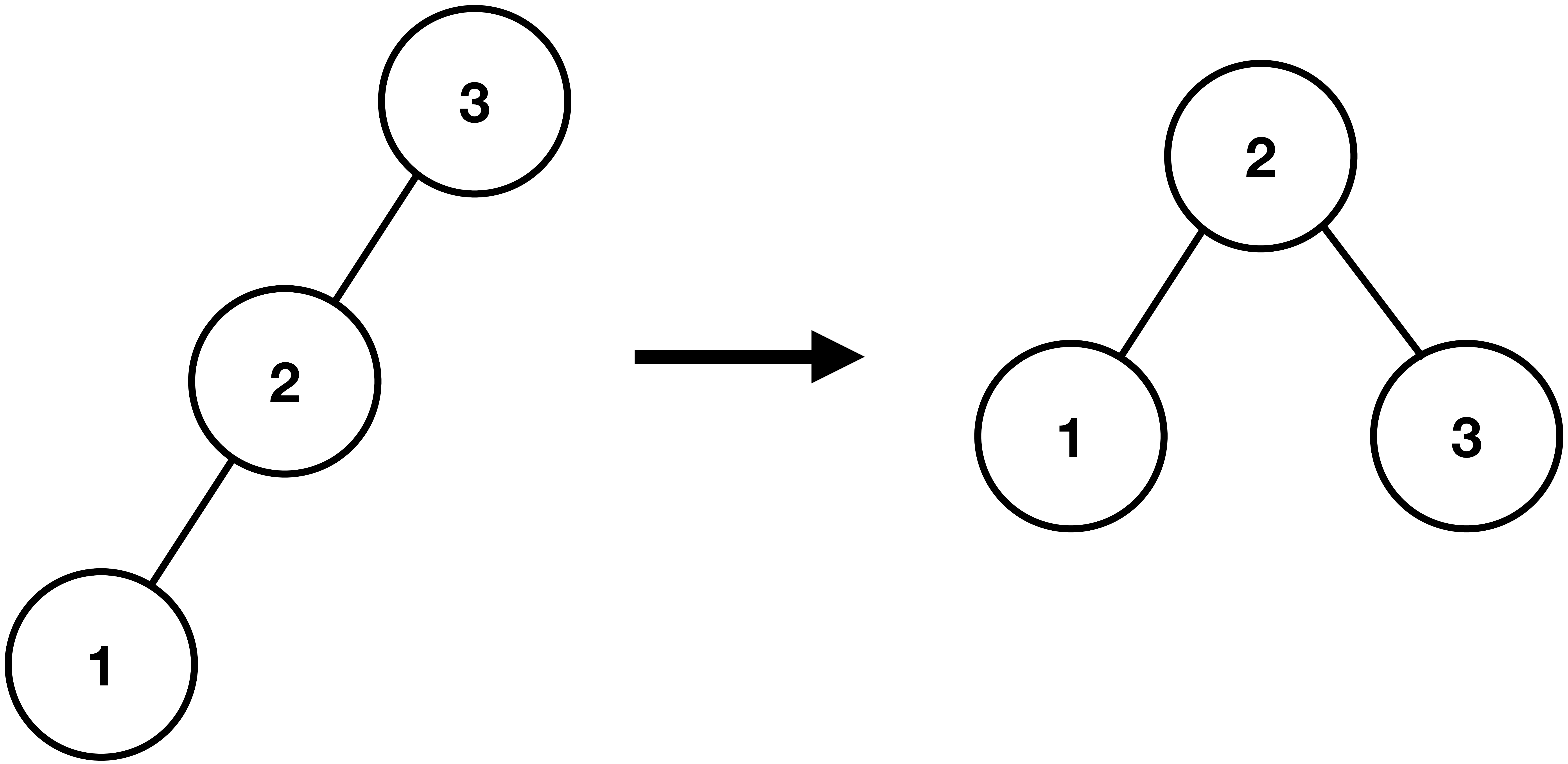
What is an AVL Tree?

- An **AVL Tree** is a BST with a height balance property
- A BST is height balanced if for any node, the heights of the node’s left and right subtrees differ by only 0 or 1
- Balance factor is the left subtree height minus the right subtree height (which is 1, 0, or -1 in an AVL Tree)
- If the balance factor is not -1, 0, or 1 you will have to perform a rotation on the tree in order to make it balanced
- A **rotation** is a local rearrangement of a BST that maintains the BST ordering property while rebalancing the tree
- The time complexity for an AVL Tree will always be **O(log(n))**

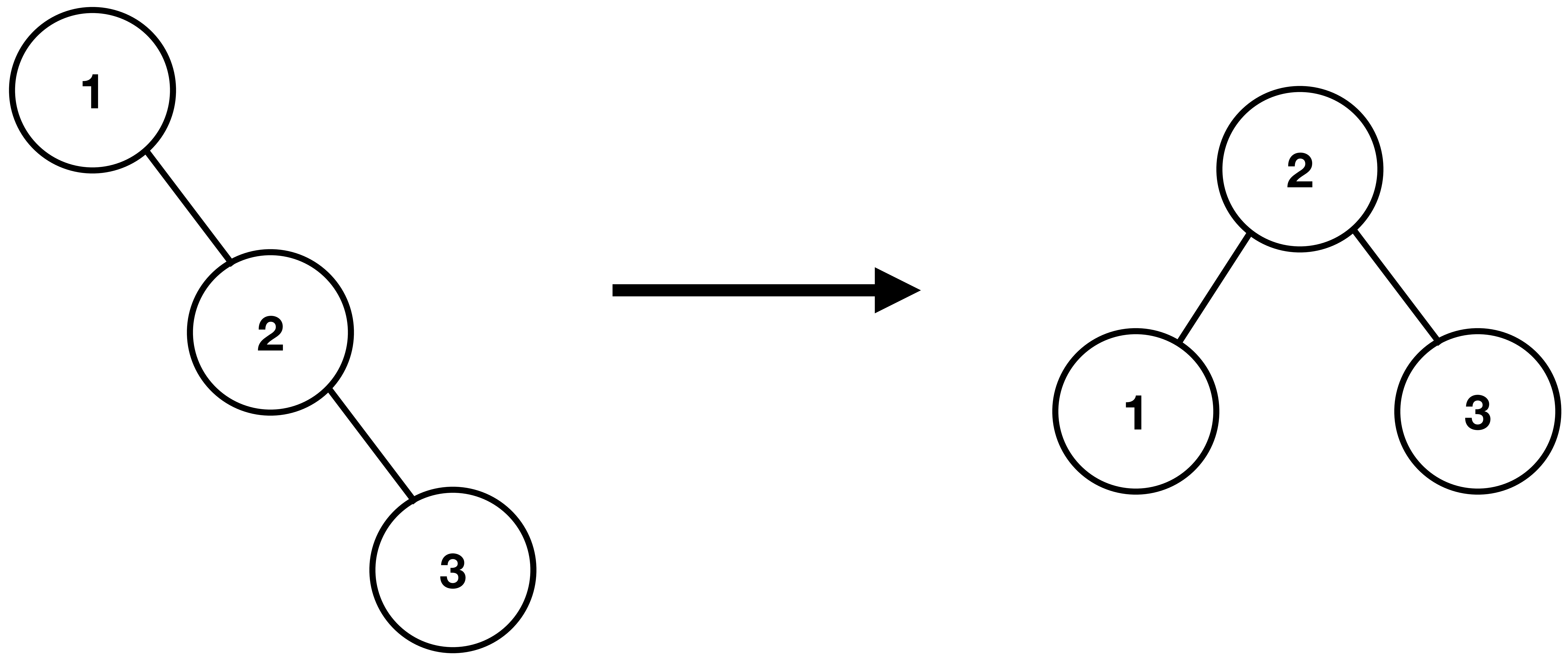
AVL Tree: Rotations

- The AVL Tree rotations include:
  - Single Right Rotation (srr)
  - Single Left Rotation (slr)
  - Right Left Rotation (rlr)
  - Left Right Rotation (lrr)

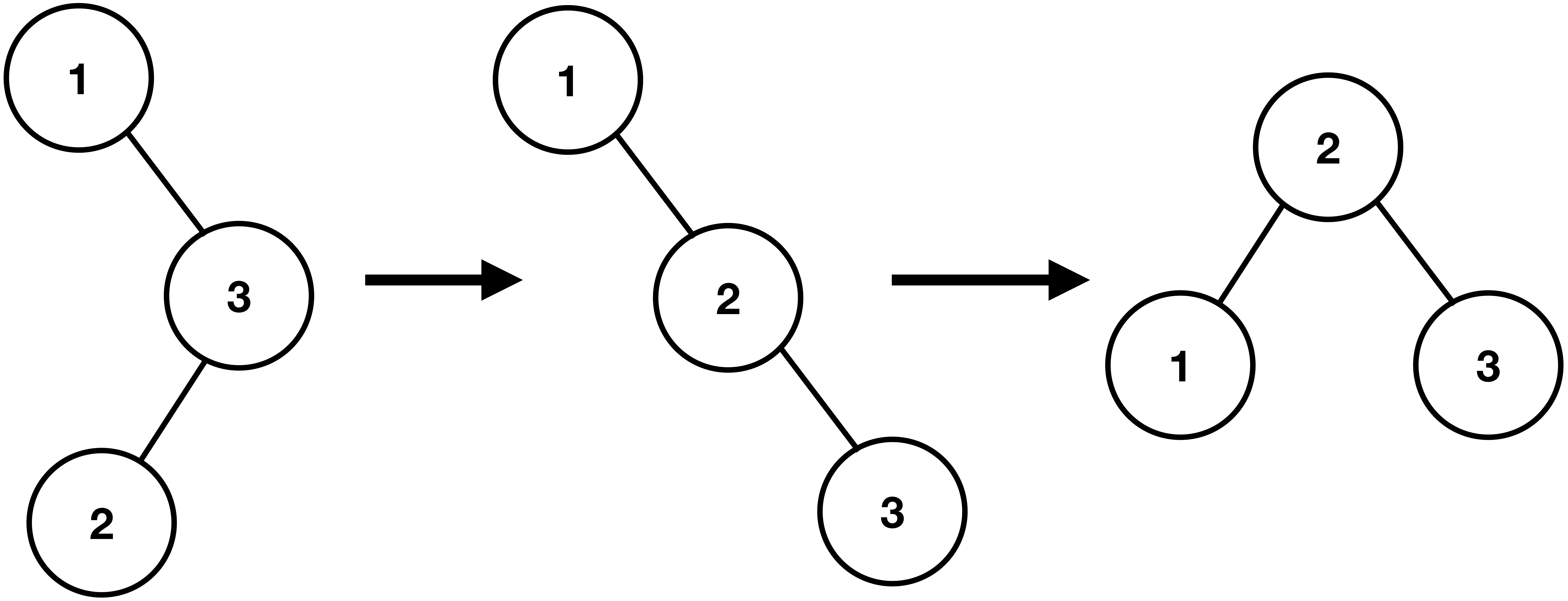
AVL Tree: Single Right Rotation (srr)



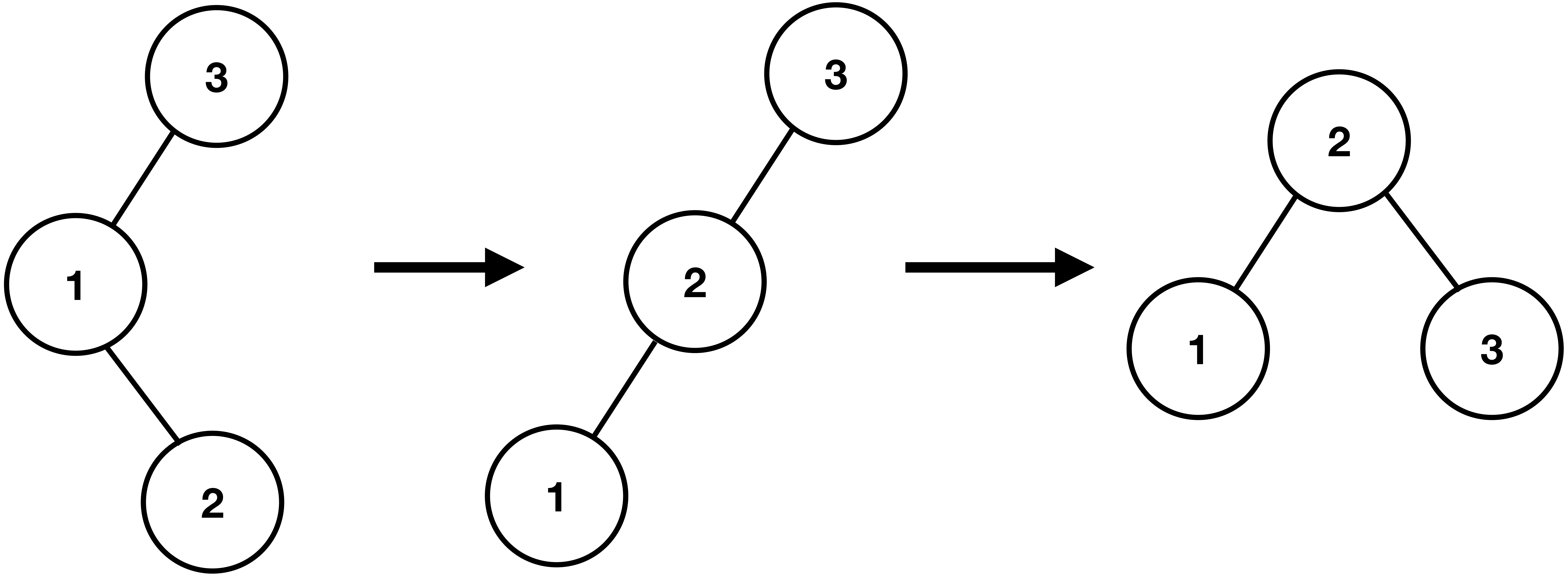
AVL Tree: Single Left Rotation (slr)



AVL Tree: Right Left Rotation (rlr)

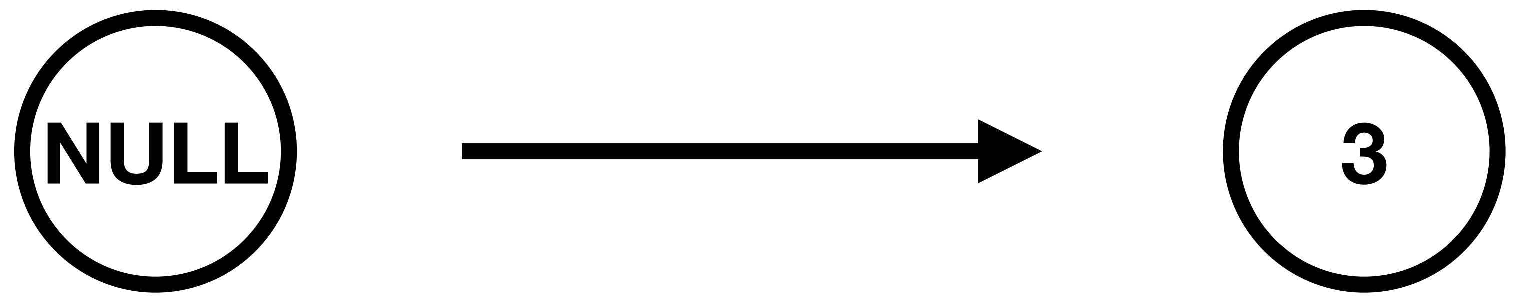


AVL Tree: Left Right Rotation (lrr)



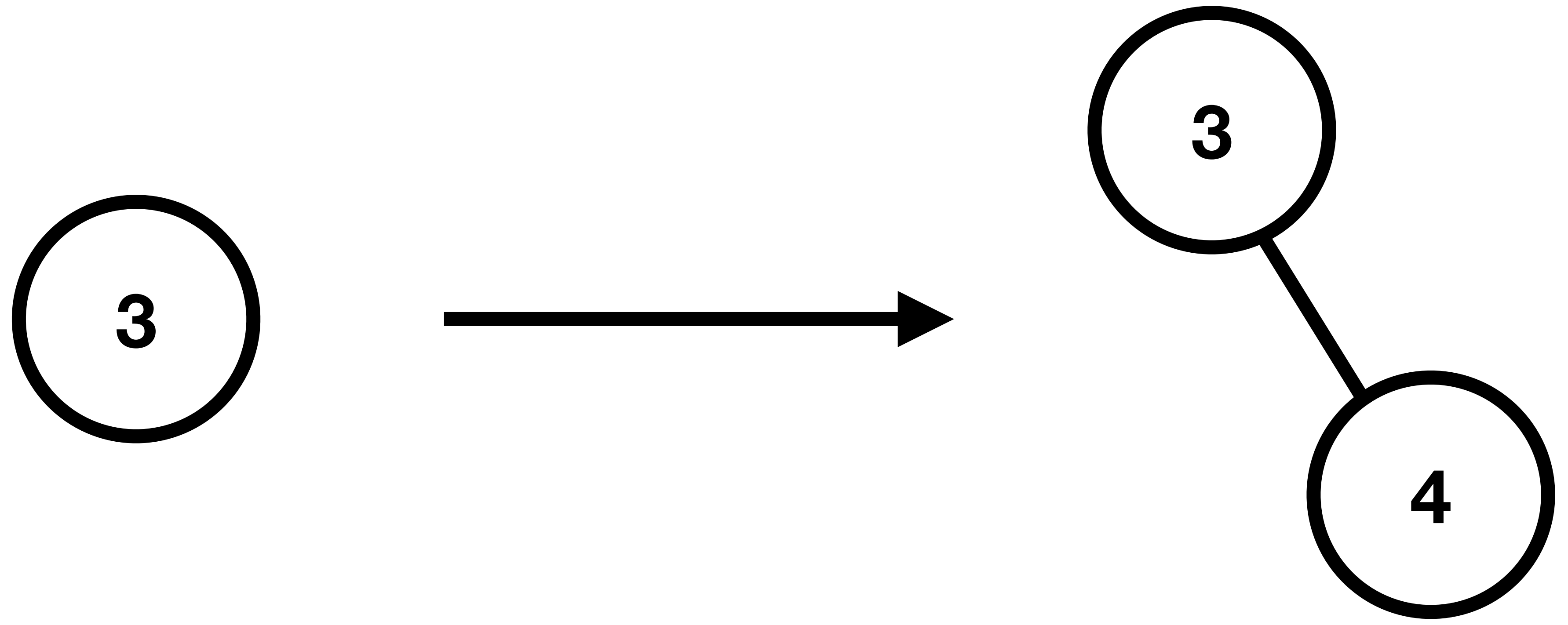
AVL Tree: Insertion

Insert 3



AVL Tree: Insertion

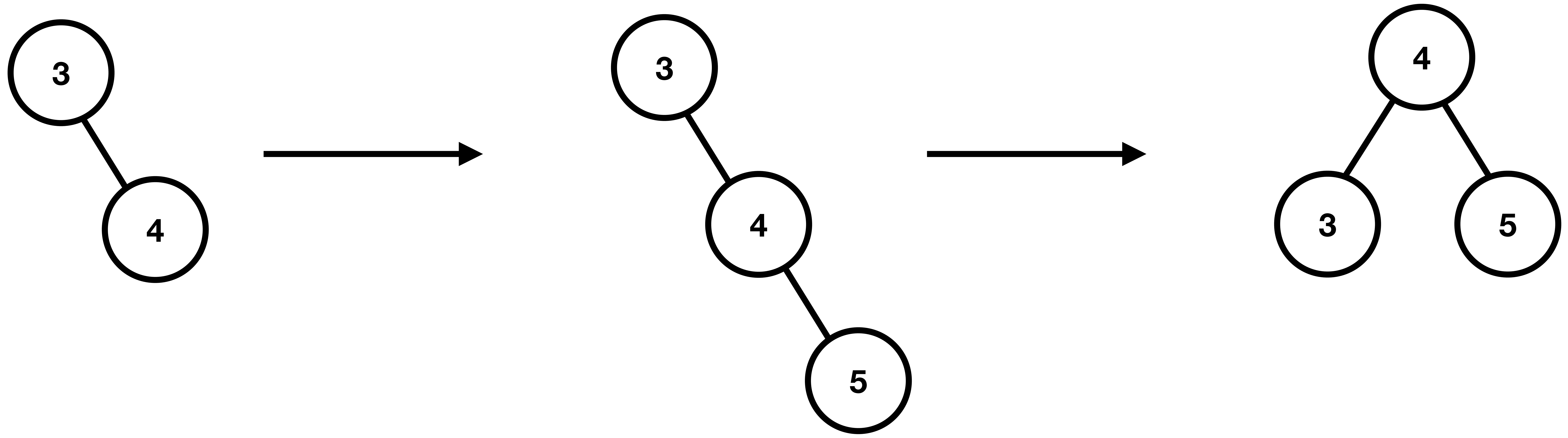
Insert 4



AVL Tree: Insertion

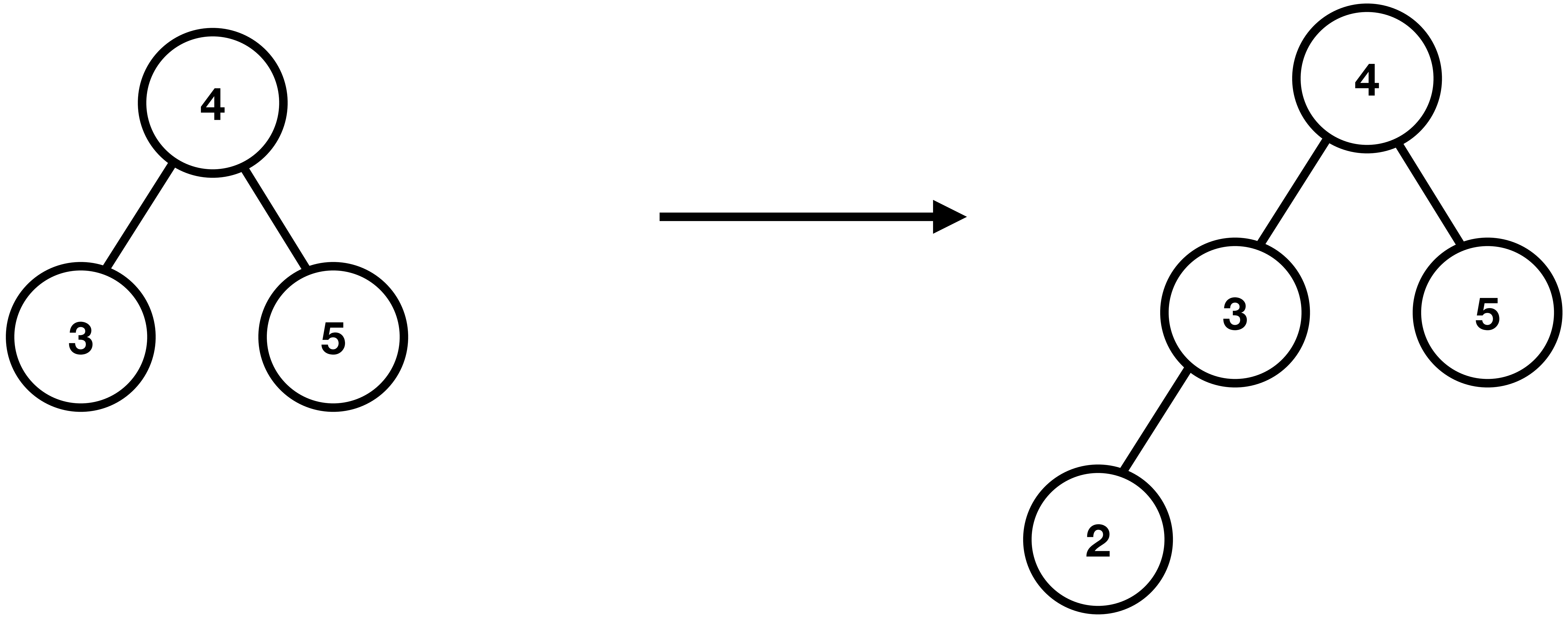
Insert 5

Balance factor is -2 at *node 3* so we do single left rotation



AVL Tree: Insertion

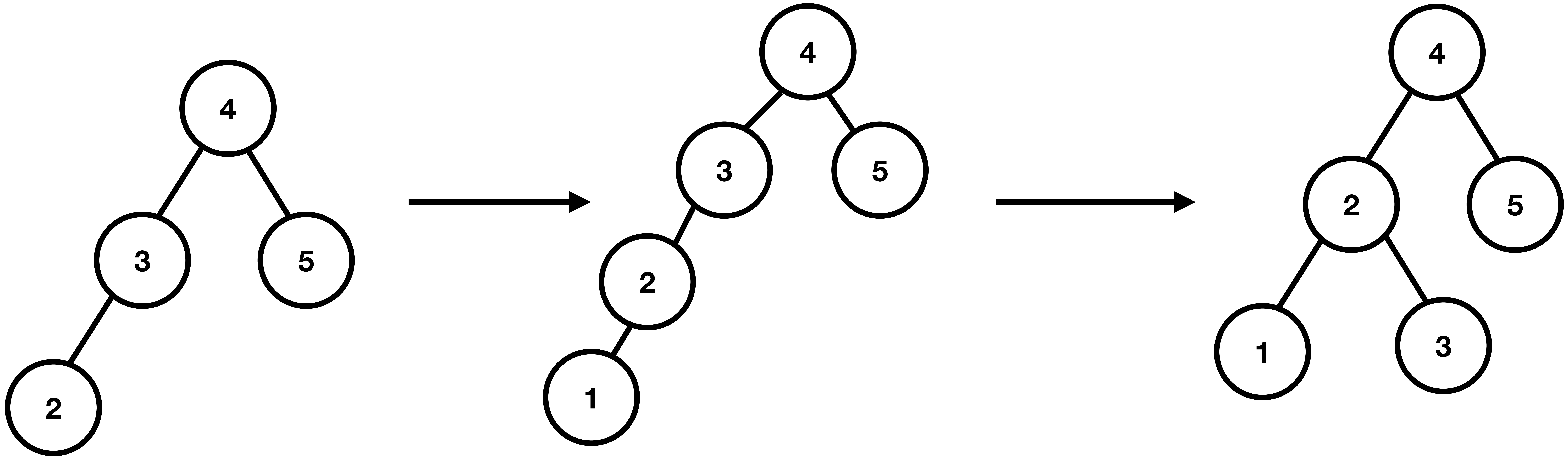
Insert 2



AVL Tree: Insertion

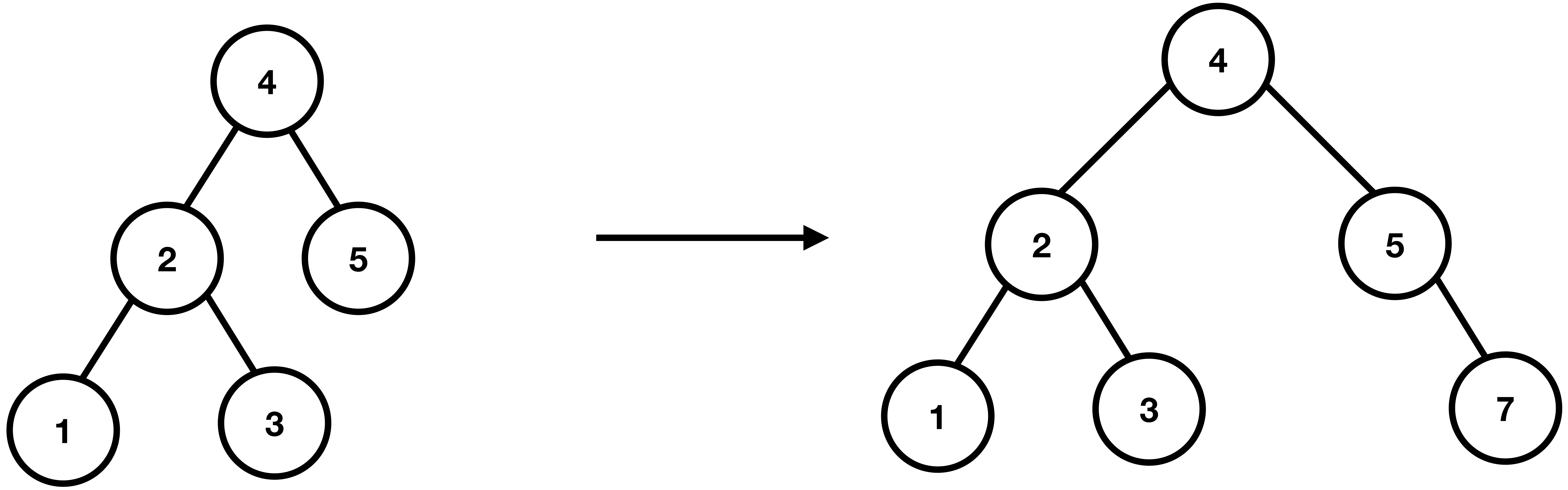
Insert 1

Balance factor is 2 at *node 3* so we do single right rotation



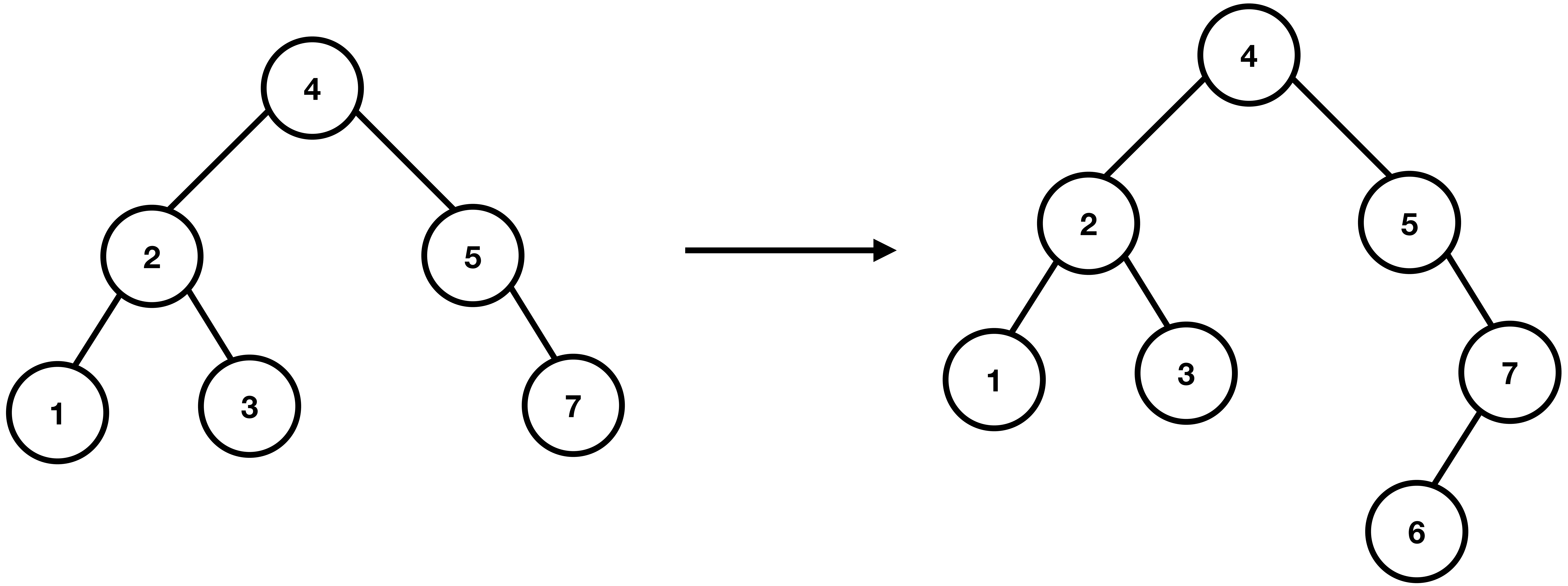
AVL Tree: Insertion

Insert 7



AVL Tree: Insertion

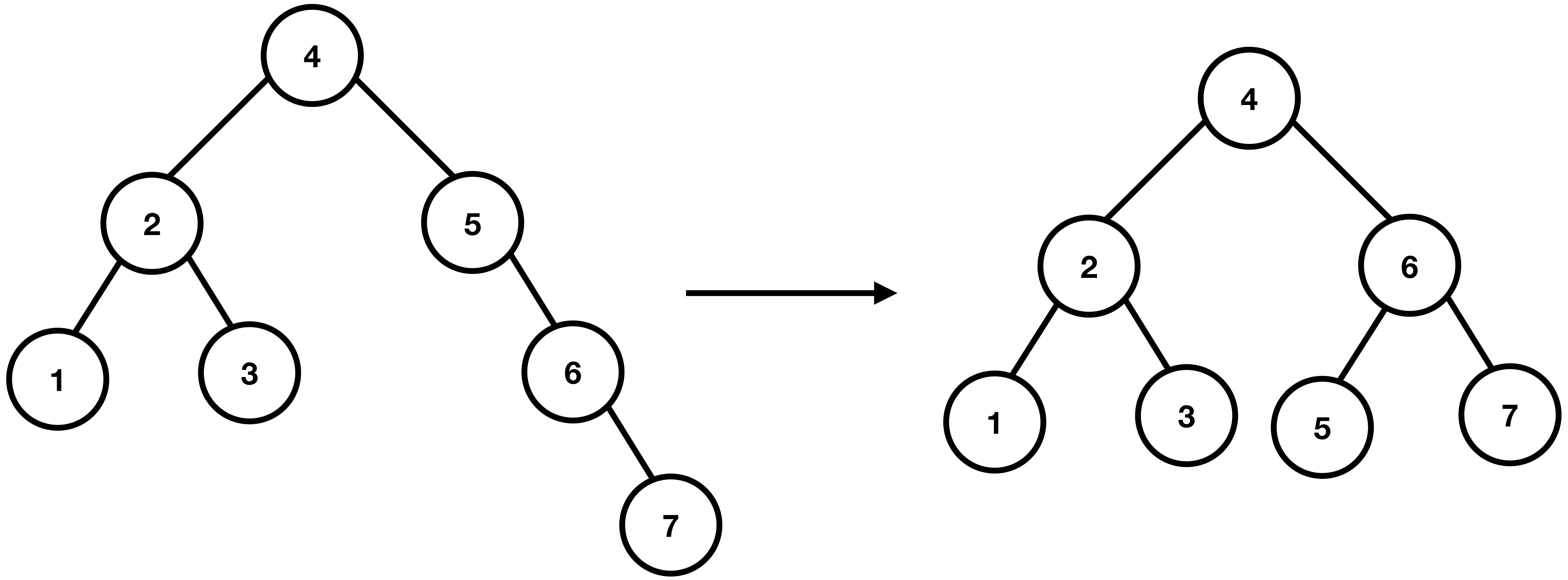
Insert 6
Balance factor is -2 at *node 5* so we do a right left rotation



continued on next page

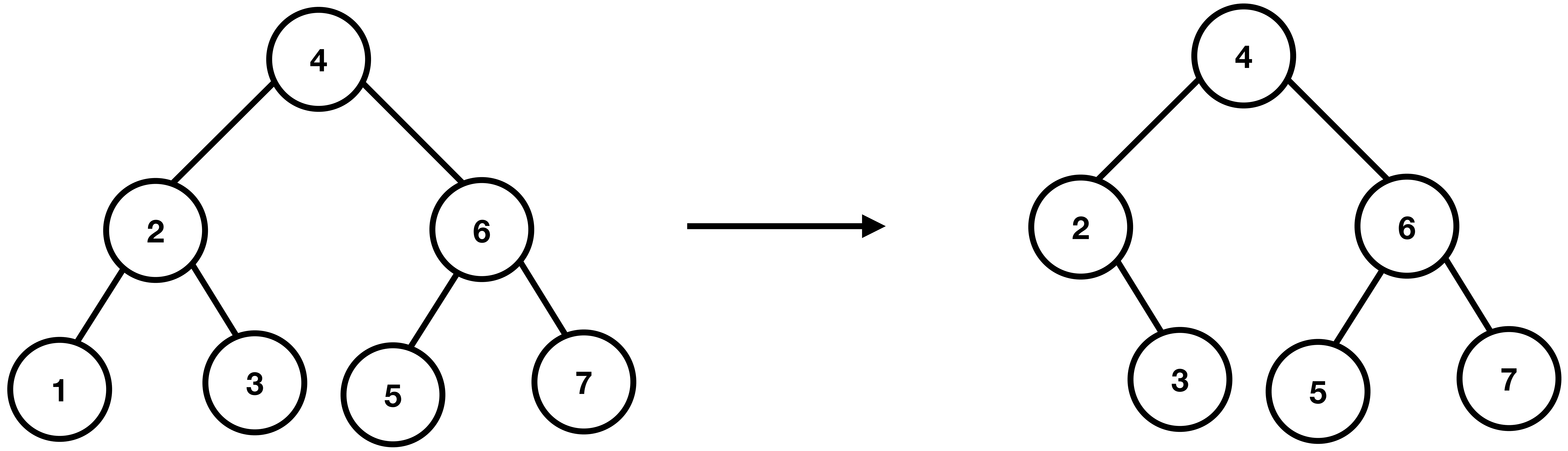
AVL Tree: Insertion

Insert 6 (continued)



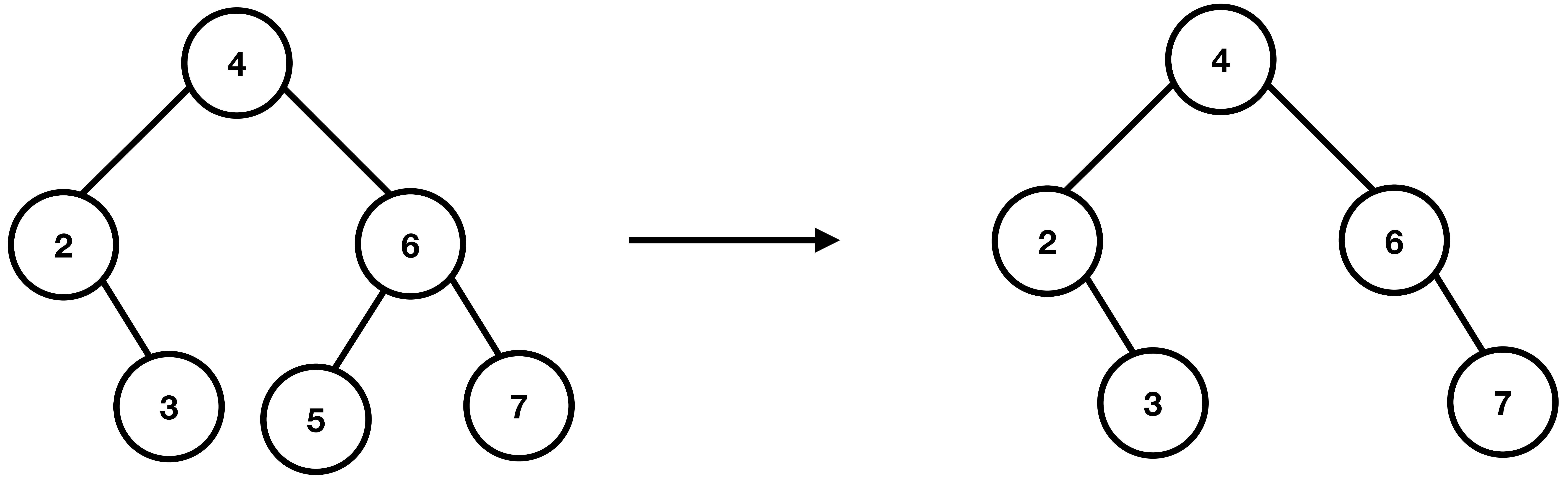
AVL Tree: Deletion

Delete 1



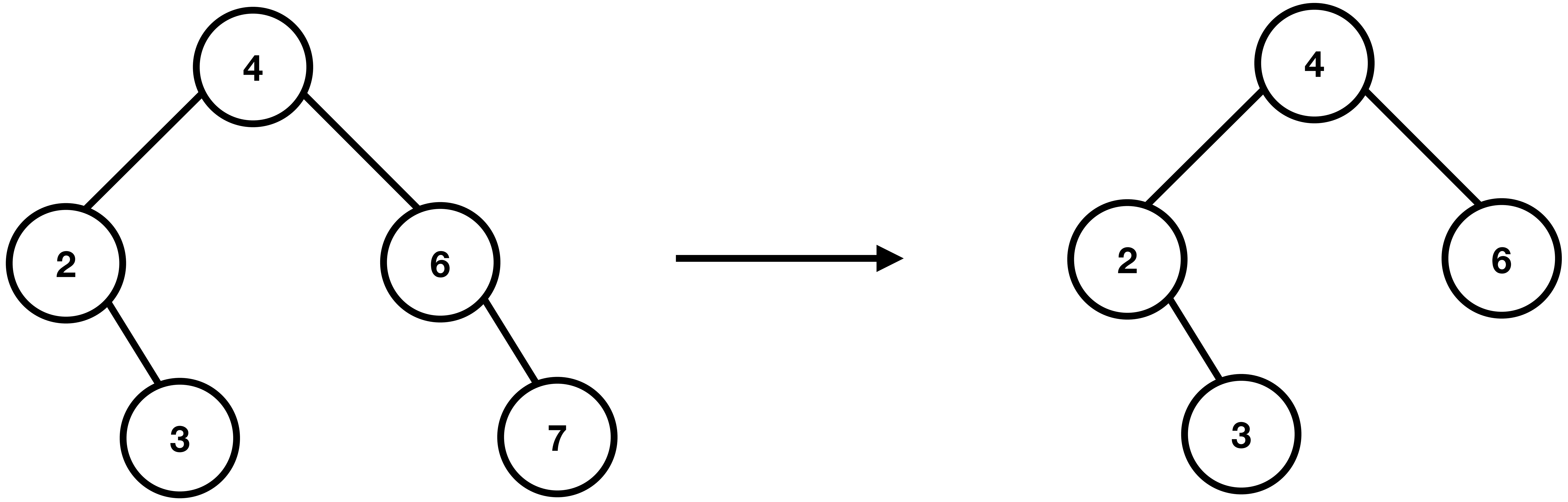
AVL Tree: Deletion

Delete 5



AVL Tree: Deletion

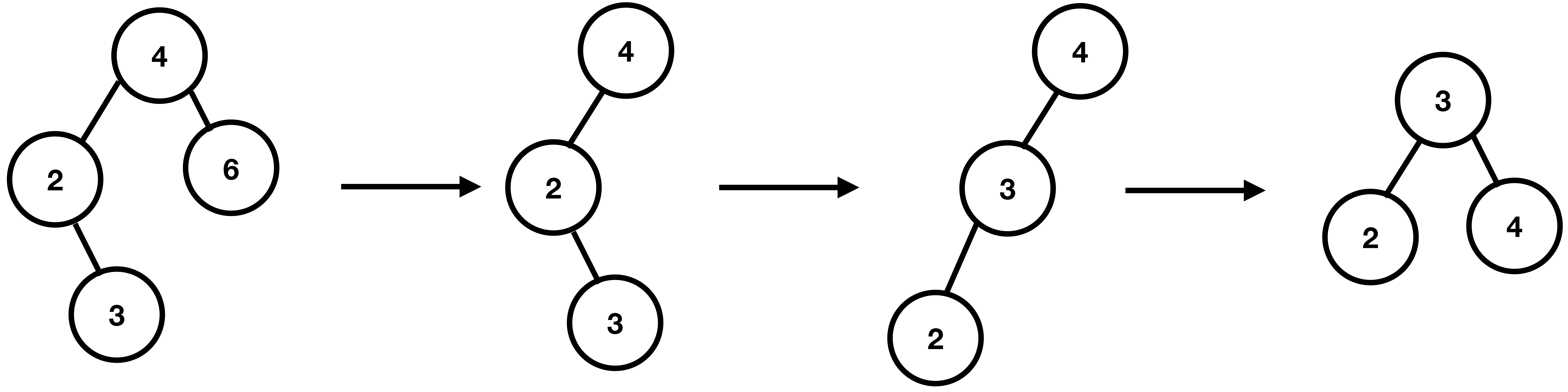
Delete 7



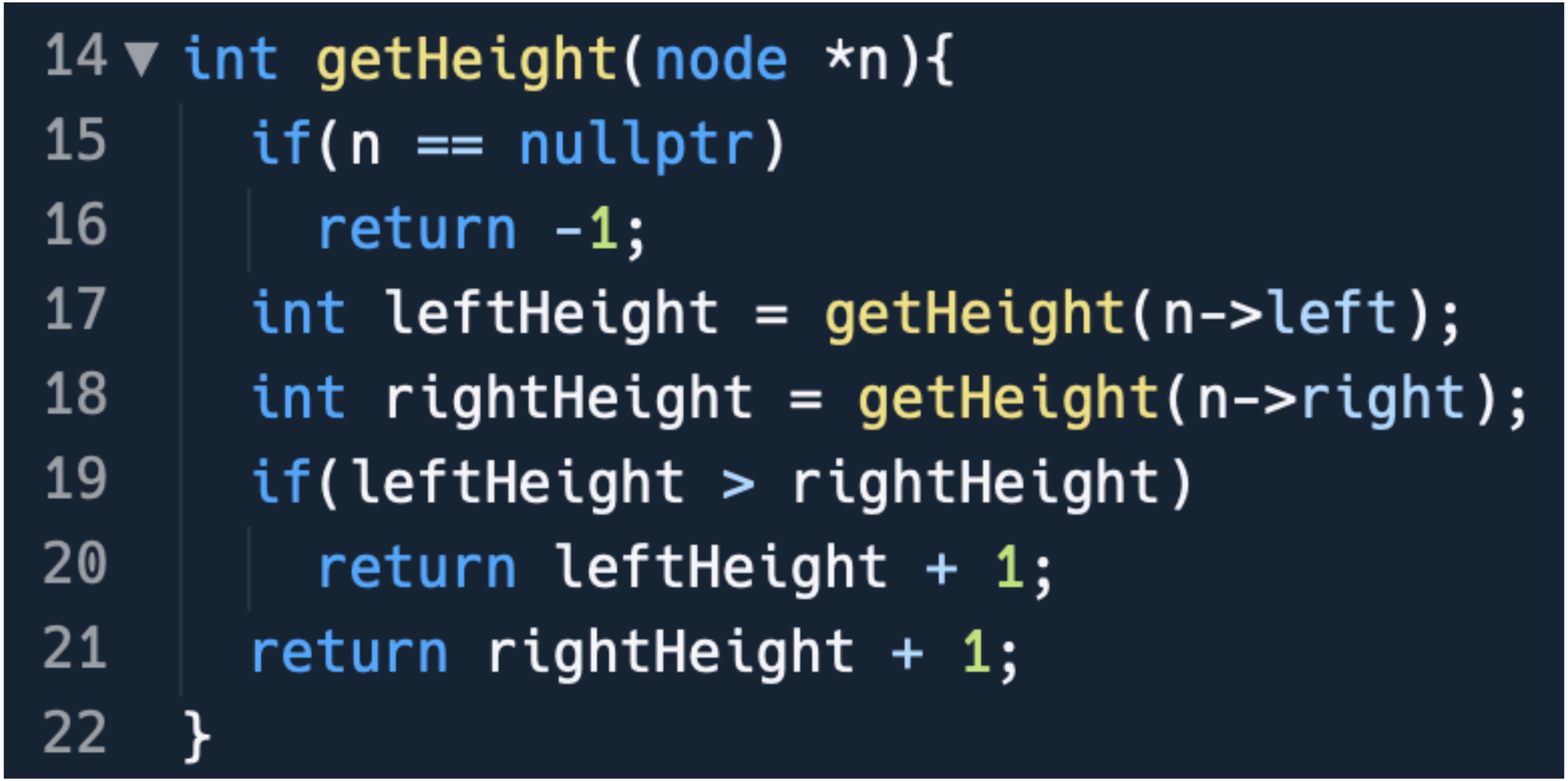
AVL Tree: Deletion

Delete 6

Balance factor is 2 at *node 4* so do left right rotation

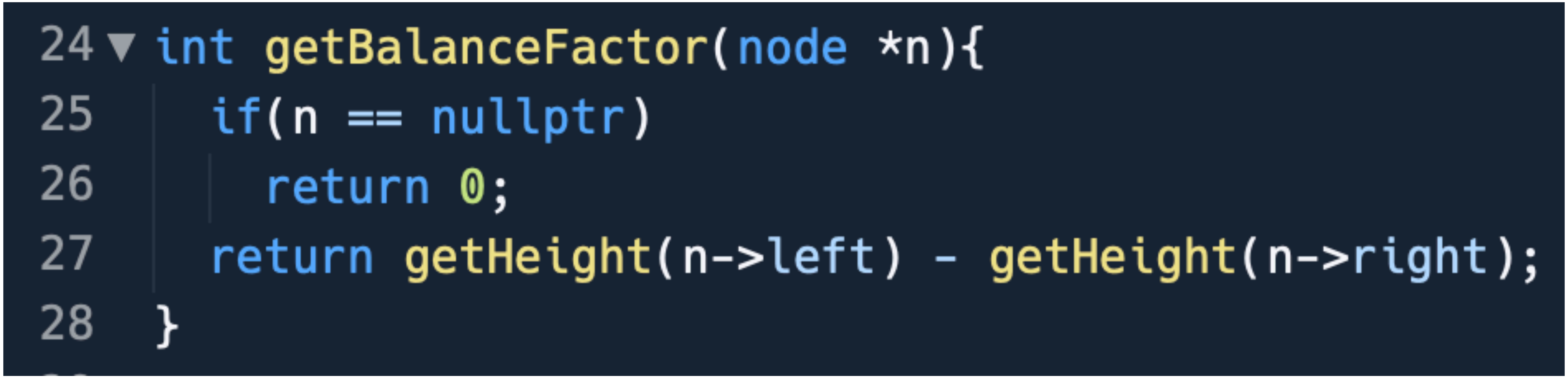


AVL Tree: Height Function



```cpp
int getHeight(node *n){
    if(n == nullptr)
        return -1;
    int leftHeight = getHeight(n->left);
    int rightHeight = getHeight(n->right);
    if(leftHeight > rightHeight)
        return leftHeight + 1;
    return rightHeight + 1;
}
```

AVL Tree: Balance Factor Function



```
int getBalanceFactor(node *n){
  if(n == nullptr)
    return 0;
  return getHeight(n->left) - getHeight(n->right);
}
```