Balanced Search Tree

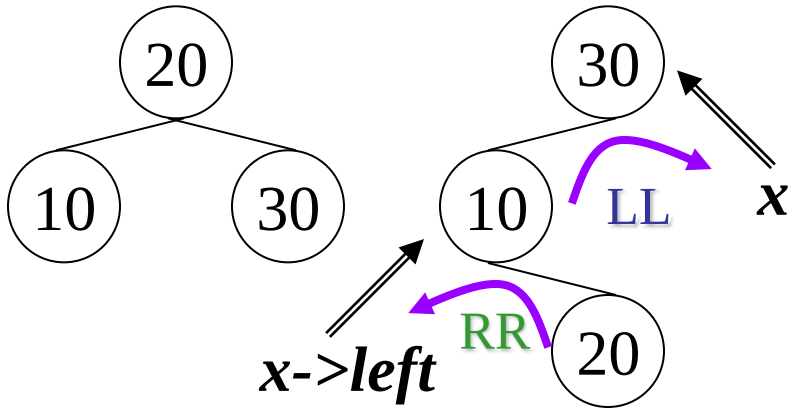
AVL *Tree*

Outline

- 2-3 tree
- 2-3-4 tree
- **AVL tree**
  - [Adelson-Velskii & Landis, 1962]
- **Red-black tree**
  - [Rudolf Bayer, 1972]… **B-tree**

AVL Tree: *Double Rotations*

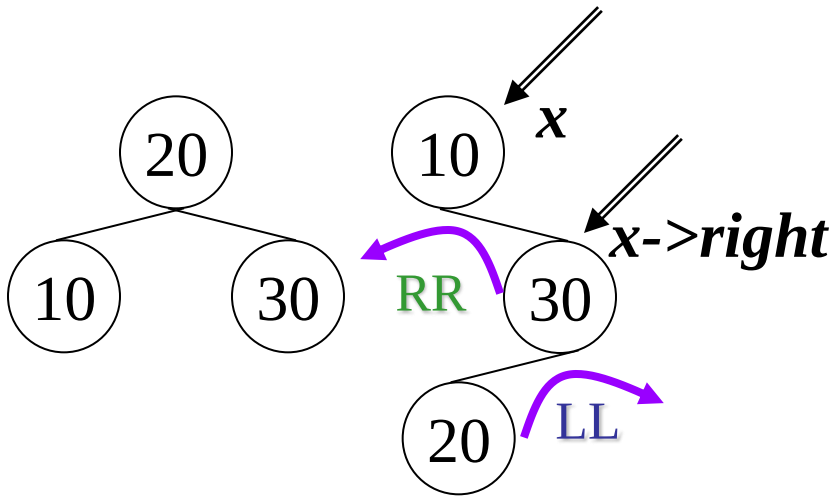
- **Let *x* be the node at which *x->left* and *x->right* differ by more than 1; Assume that the height of *x* is 3**
  - Height of *x->left* is **2** (i.e. height of *x->right* is 0)
    3. **Height of *x->left->right*: 1 ⇒ double rotation with the left child (LR)**
       - **BF(x) = +2**
       - **BF(x->left) = -1**



Tree height: shorter

AVL Tree: *Double Rotations*

- **Let *x* be the node at which *x->left* and *x->right* differ by more than 1; Assume that the height of *x* is 3**
  - Height of *x->right* is **2** (i.e. height of *x->left* is 0)
    4. **Height of *x->right->left*: 1 ⇒ double rotation with the right child (RL)**
       - **BF(x) = -2**
       - **BF(x->right) = +1**



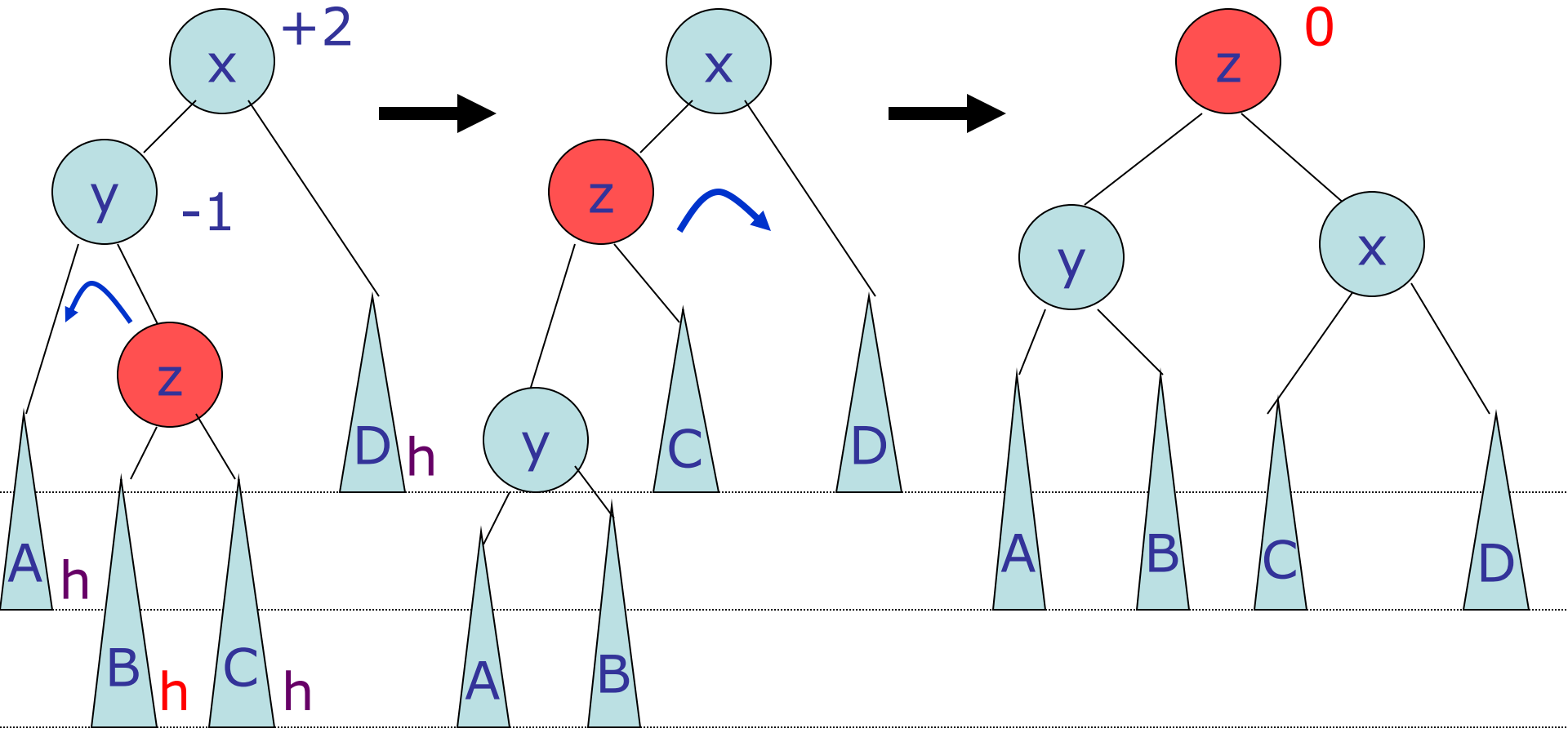
Tree height: shorter

AVL Tree: *Double Rotations*

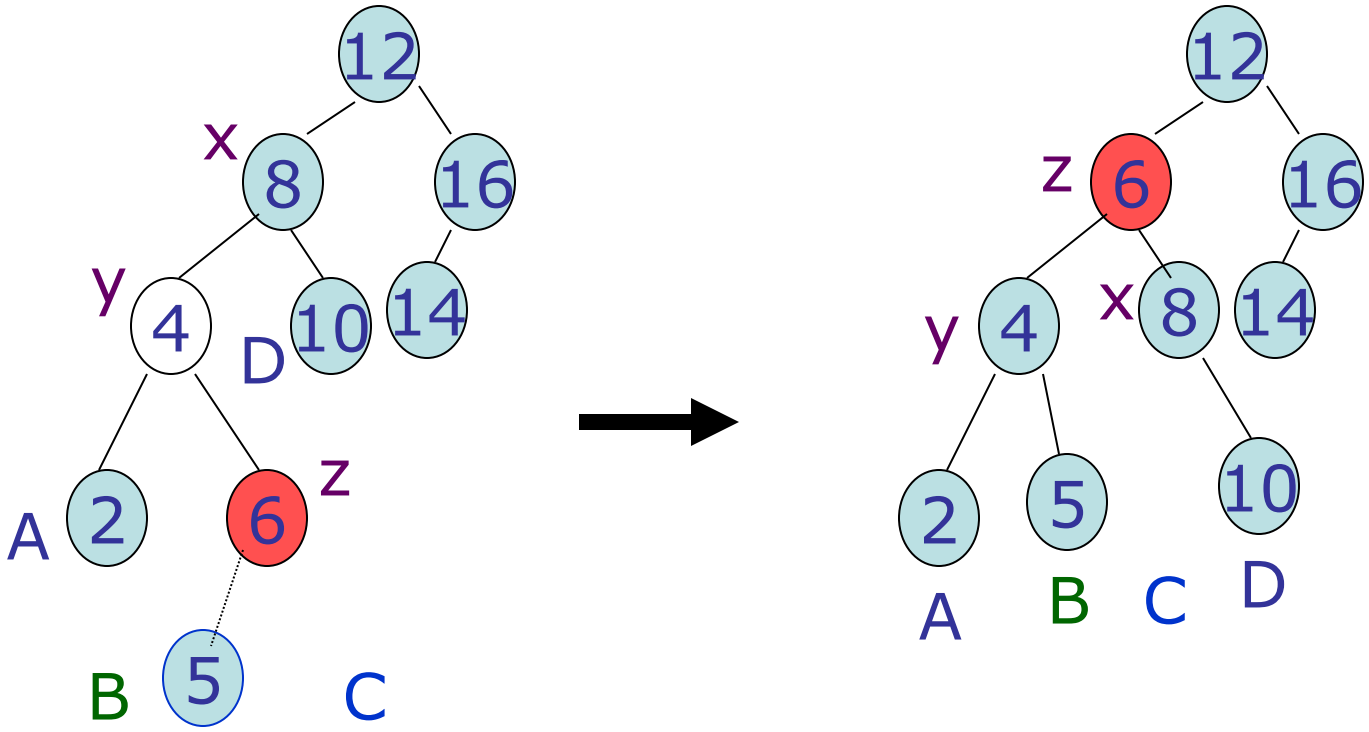
- **Let *x* be the node at which *x->left* and *x->right* differ by more than 1; Assume that the height of *x* is h+3**
  - Heights of two subtrees: **h+2**, h
    3. **Height of *x->left->right*: h+1 ⇒ double rotation with the left child (LR):** RR ➔ LL
       - **BF(x) = +2, BF(x->left) = -1**
    4. **Height of *x->right->left*: h+1 ⇒ double rotation with the right child (RL):** LL ➔ RR
       - **BF(x) = -2, BF(x->right) = +1**

Q&A: What are the conditions for double rotations?

Double Rotations: *LR*



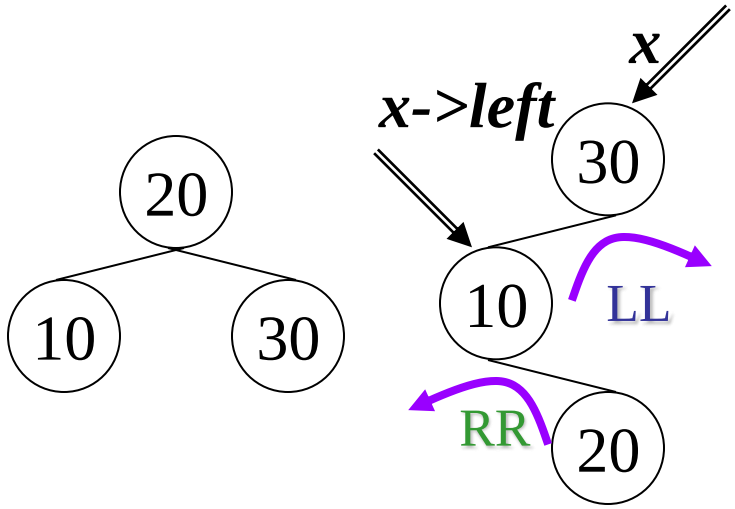
Double Rotations: *LR*



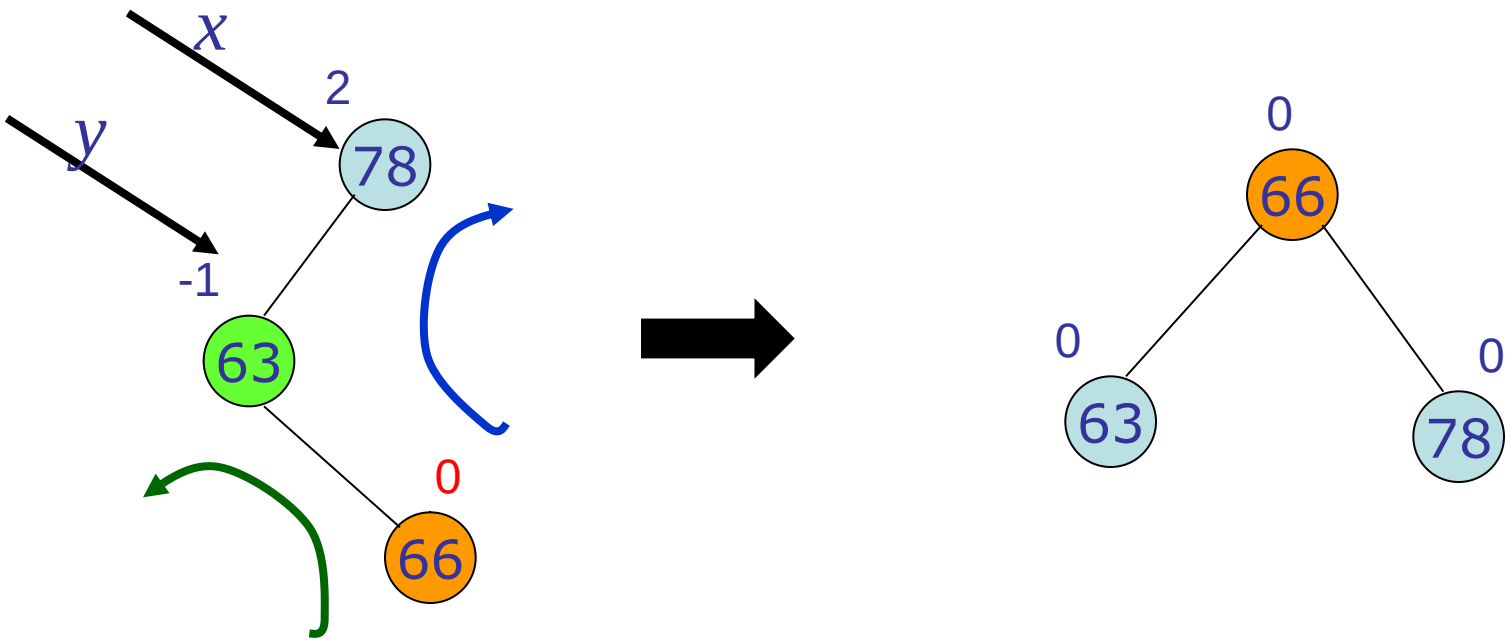
Insert “5”

LR Rotation: *Pseudocode*


```
// first rotate left child with its right child; RR
// then, rotate node x with its new left child. LL
nodeType rotateLR(nodeType x)
{
    x->left = rotateRR (x->left);
    return rotateLL(x);
}
```

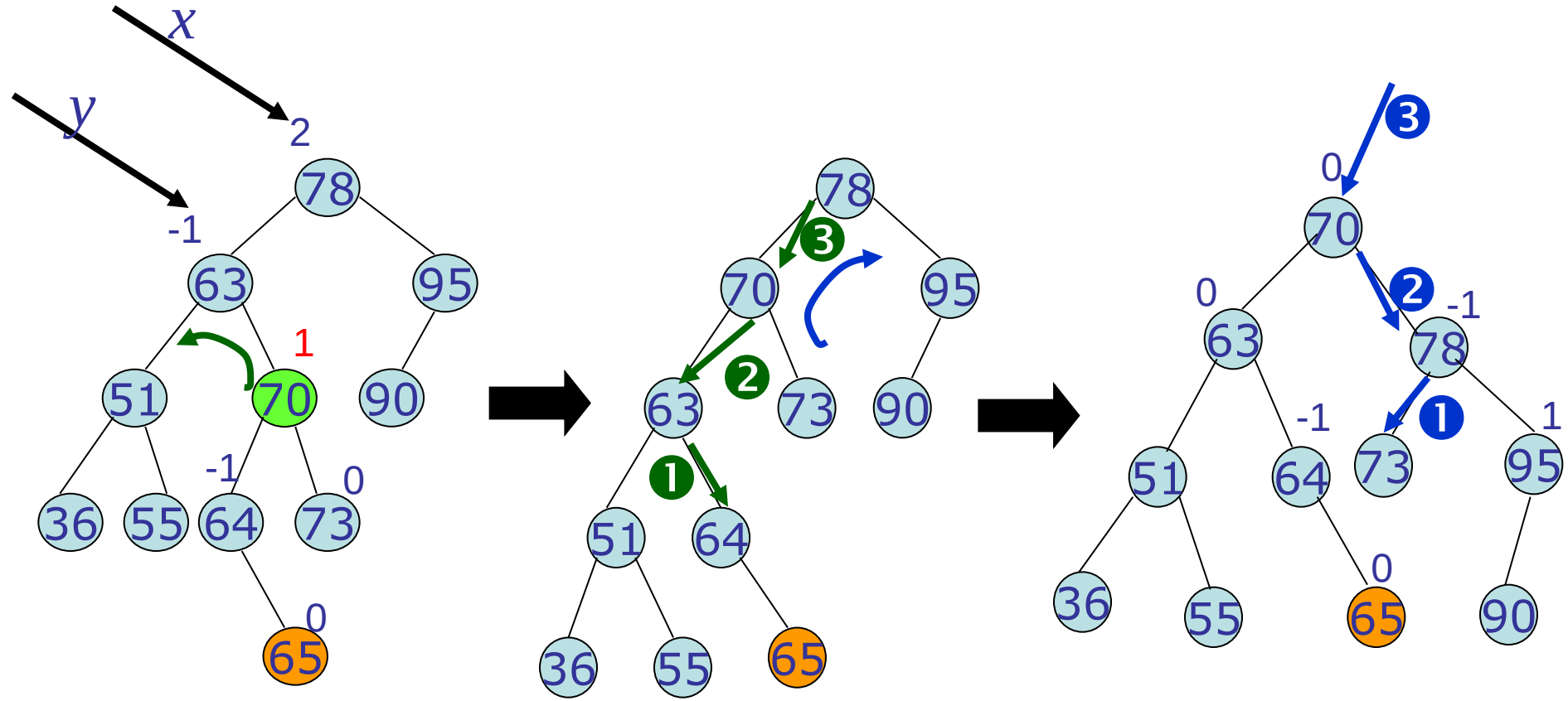


LR Rotation: *Examples*



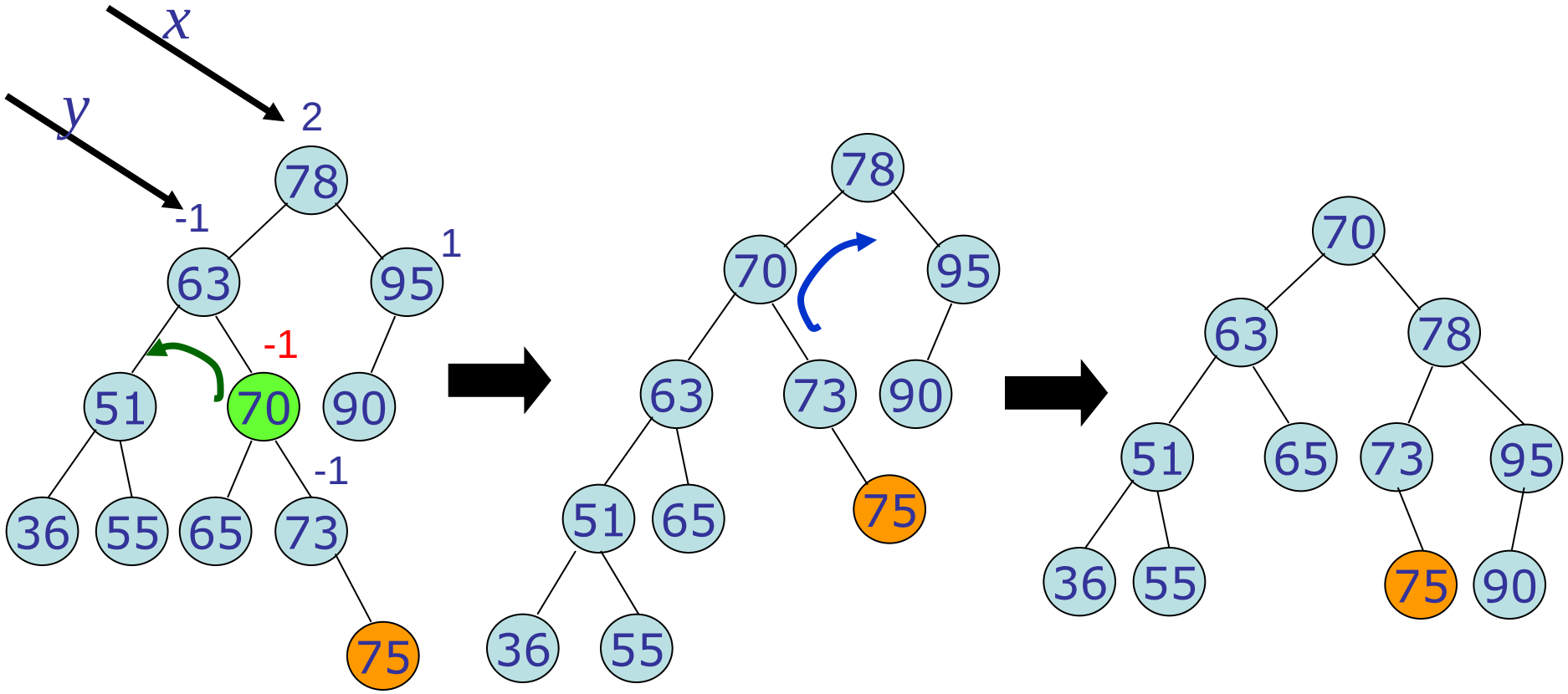
LR: case 1

LR Rotation: *Examples*



LR: case 2

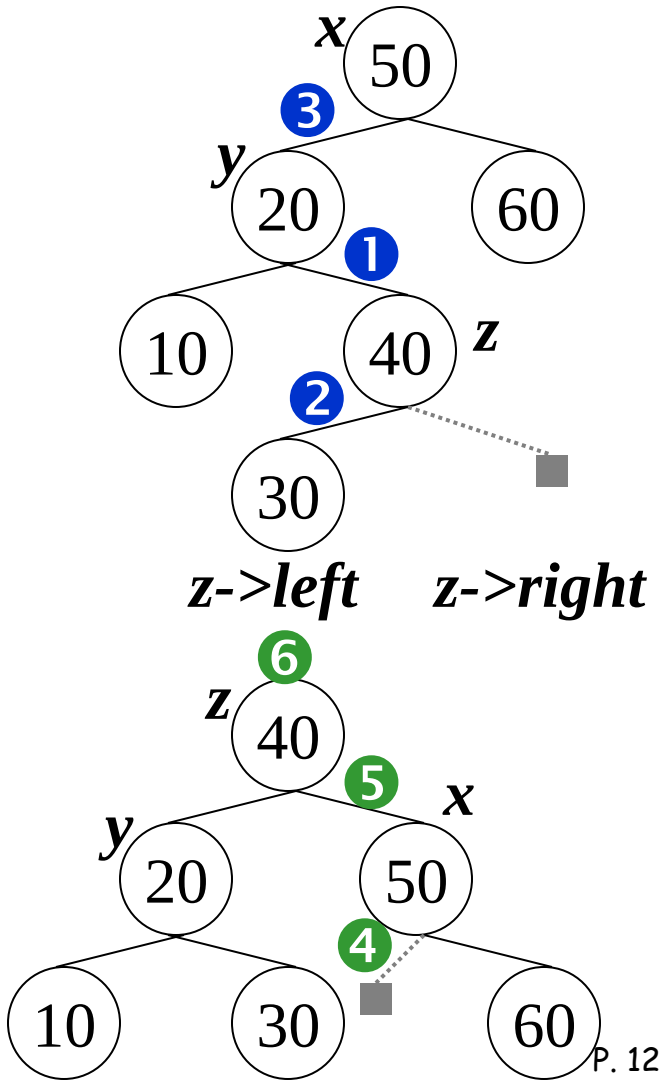
LR Rotation: *Examples*



LR: case 3

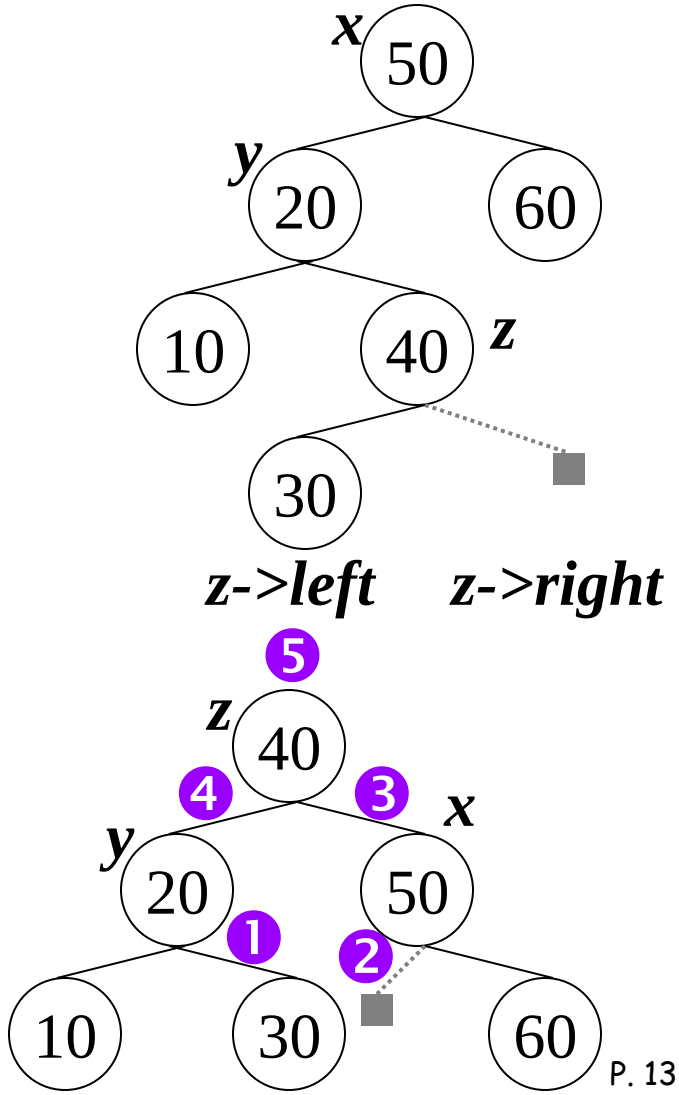
LR Rotation: *Version I.*


```
nodeType rotateLR1(nodeType x)
{   nodeType y = x->left;
    nodeType z = y->right;
// RR rotation on y
    y->right = z->left;          ❶
    z->left = y;                 ❷
    x->left = z;                 ❸
// LL rotation on x
    x->left = z->right;          ❹
    z->right = x;                ❺
    return z;                    ❻
}
```

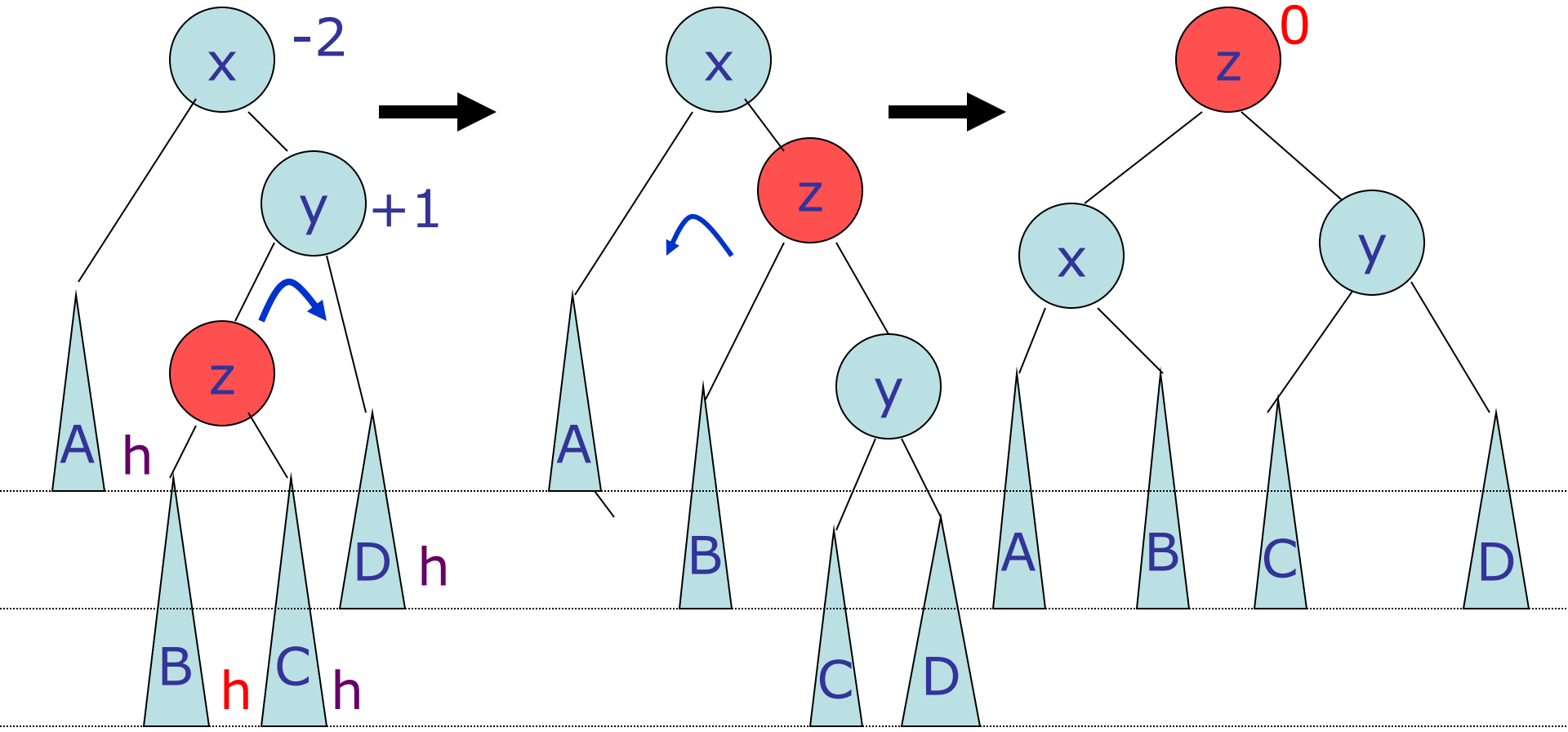


LR Rotation: *Version II.*

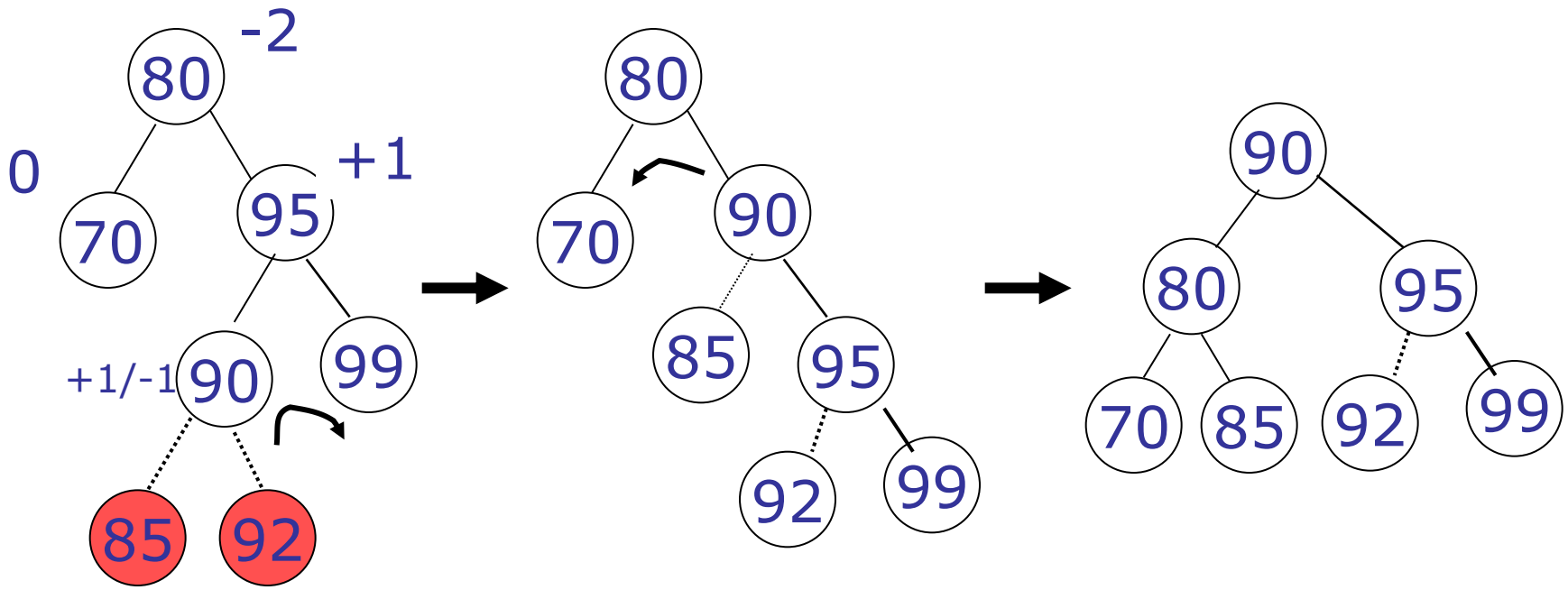

```
nodeType rotateLR2(nodeType x)
{   nodeType y = x->left;
    nodeType z = y->right;
    y->right = z->left;          ❶
    x->left = z->right;          ❷
    z->right = x;                ❸
    z->left = y;                 ❹
    return z;                    ❺
}
```



Double Rotations: *RL*



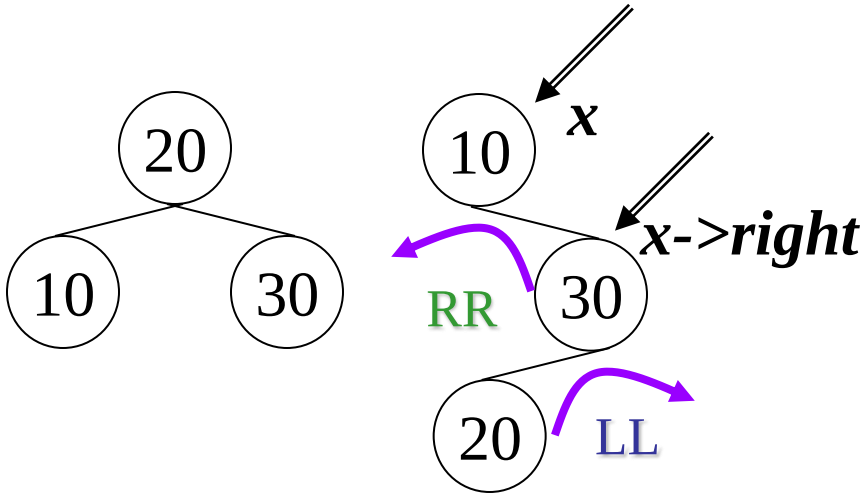
Double Rotations: *RL*



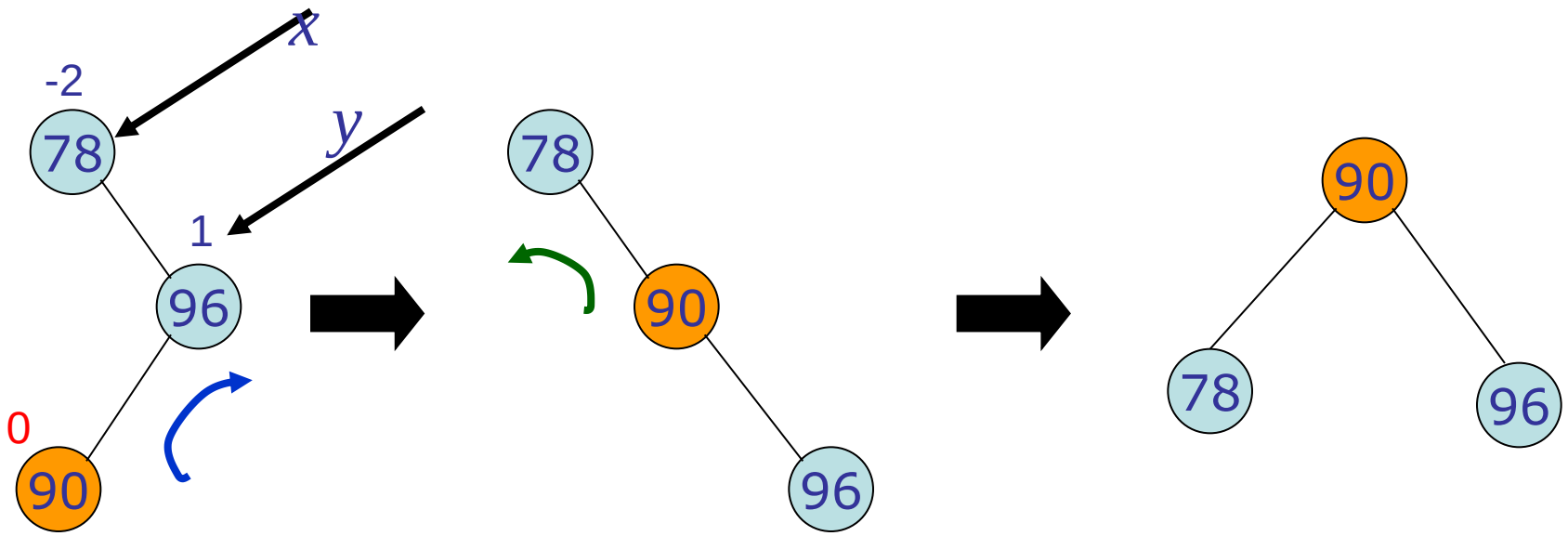
Insert 85 or 92

RL Rotation: *Pseudocode*


```
// first rotate right child with its left child; LL
// then, rotate node x with its new right child. RR
nodeType rotateRL(nodeType x)
{
    x->right = rotateLL (x->right);
    return rotateRR(x);
}
```

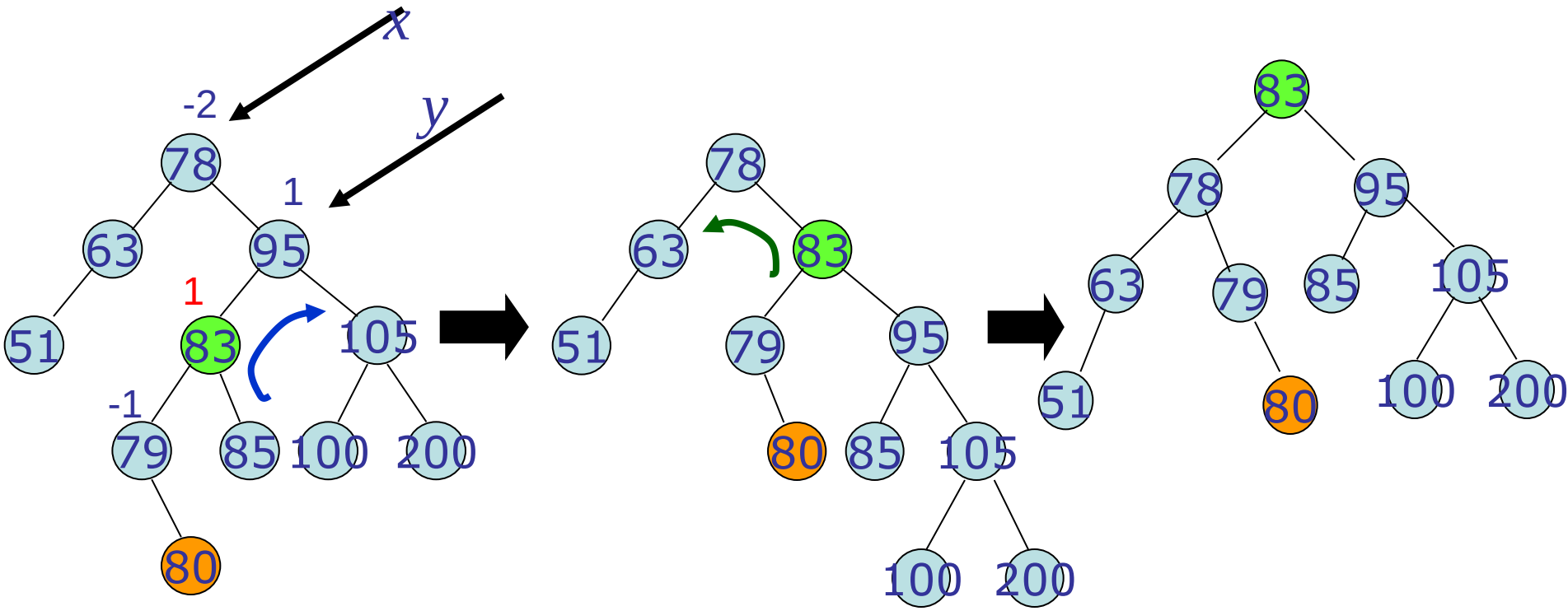


RL Rotation: *Examples*



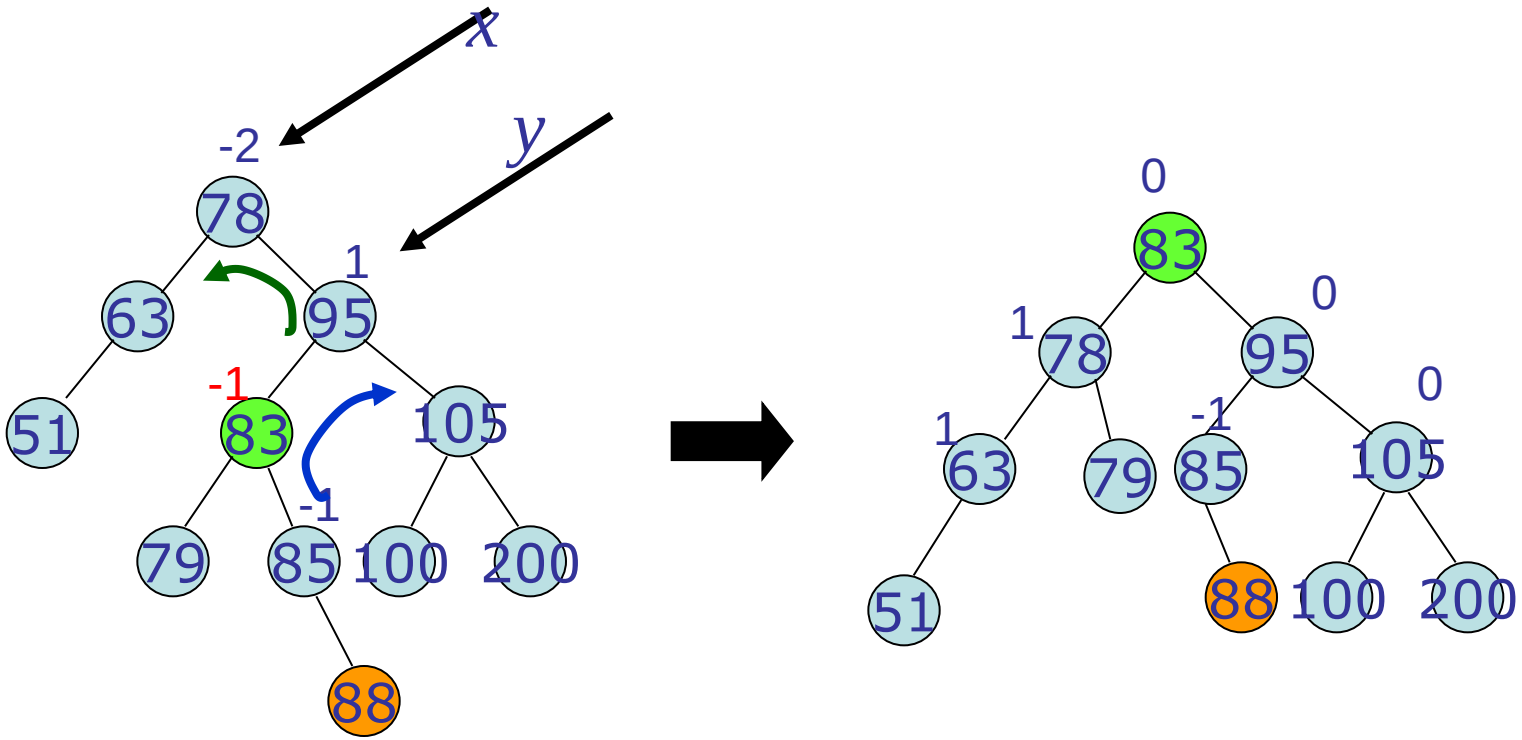
RL: case 1

RL Rotation: *Examples*



RL: case 2

RL Rotation: *Examples*



RL: case 3

Try it!

Insert *15,14*

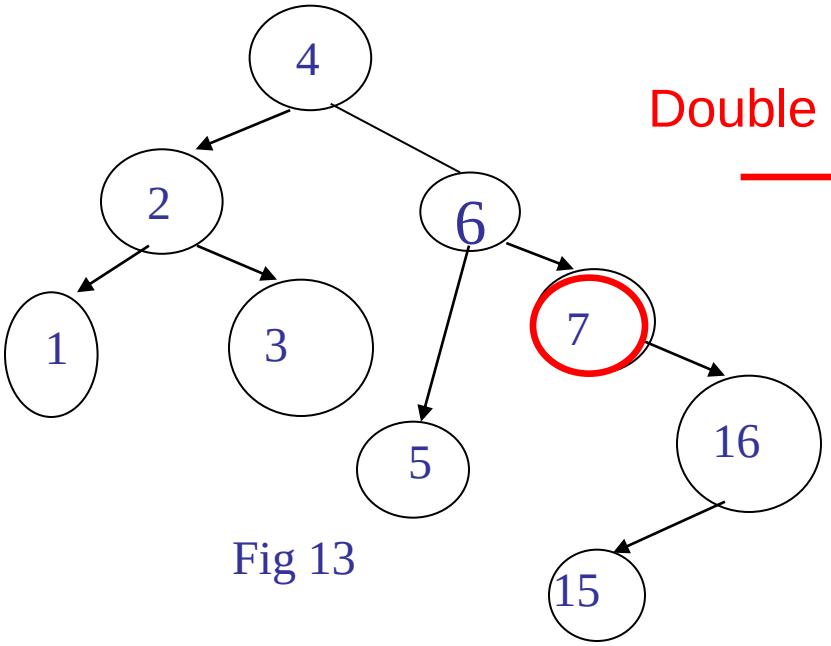


Fig 13

Double rotation

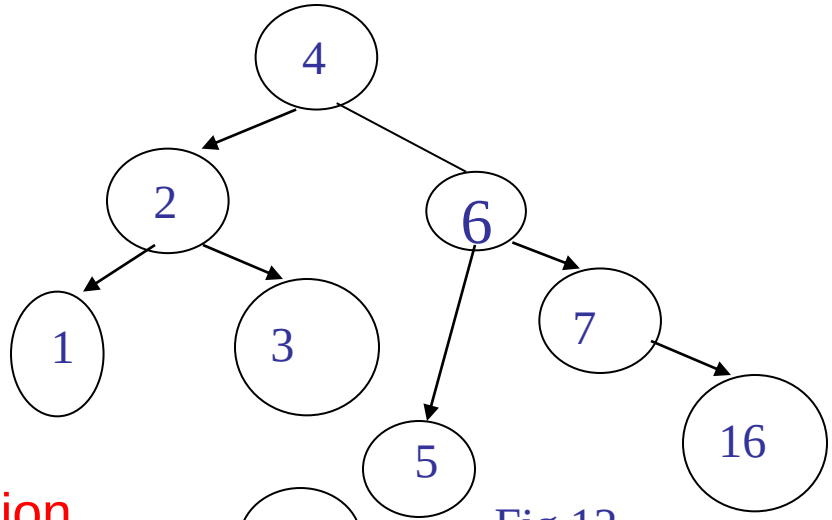


Fig 12

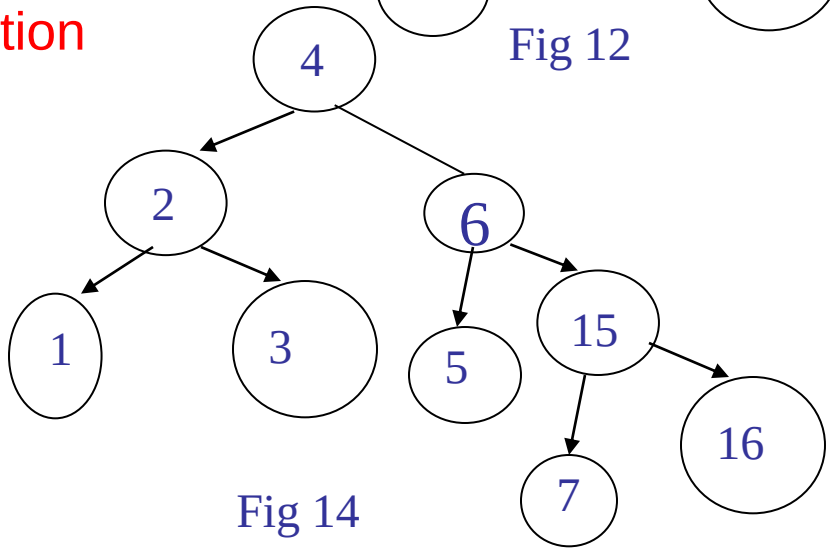


Fig 14

Try it!

Insert *15*,*14*

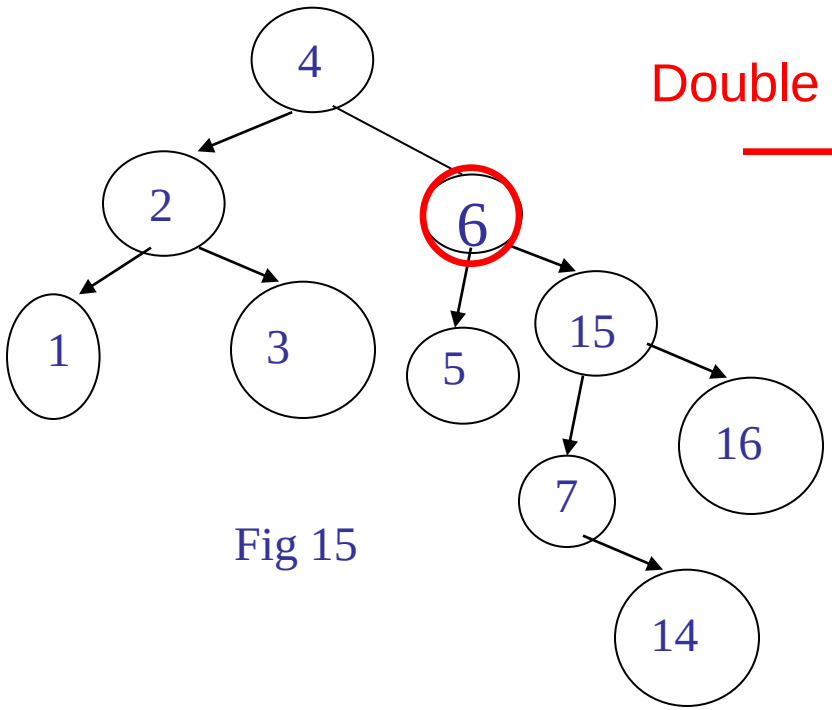


Fig 15

Double rotation

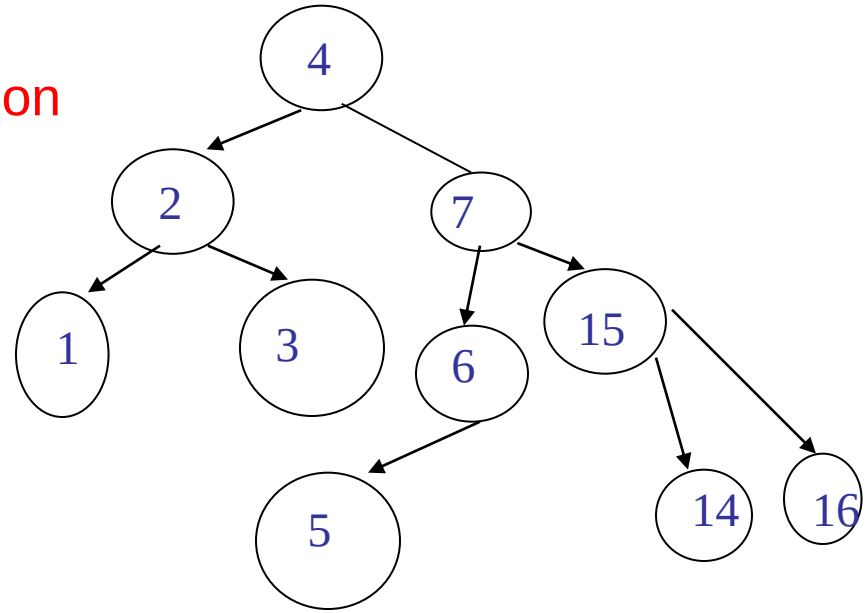


Fig 16

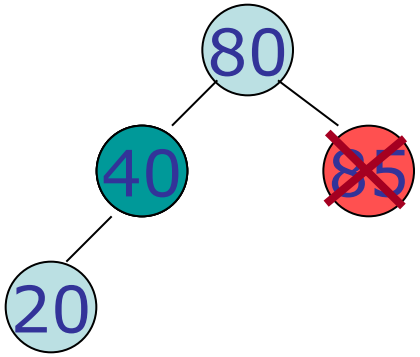
Practice 1: *Inserting* into AVL Tree

- **Input order: 10 12 30 8 60 40 70**

AVL Tree: *Deletion*

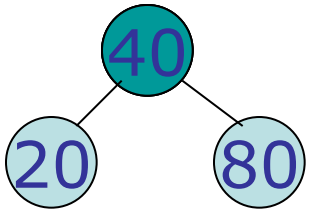
1. **Delete a node *x* as in a *binary search tree* and the last node deleted is a leaf.**
2. **Trace the path *from the new leaf towards the root*.**
   - For each node *x* encountered, check if the heights of *left(x)* and *right(x)* differ by at most 1.
     - **If NOT, restructure by either a single rotation or a double rotation**
3. **After we perform a rotation at x, we may have to perform a rotation at some ancestor of x.**
   - We must **continue** to trace the path until the root.

After delete 85,…



BF(y)=1

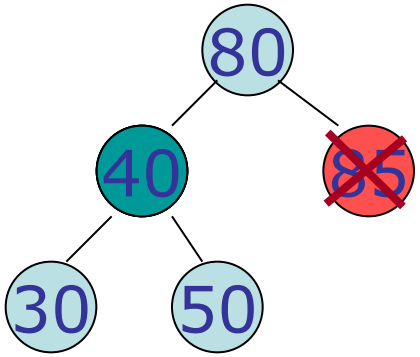
Single LL rotations



BF(y)=0

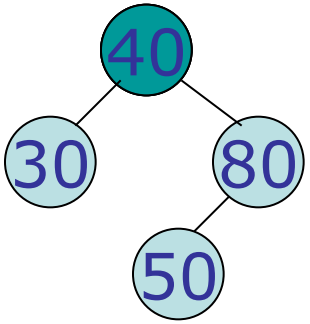
height=2

After delete 85,…



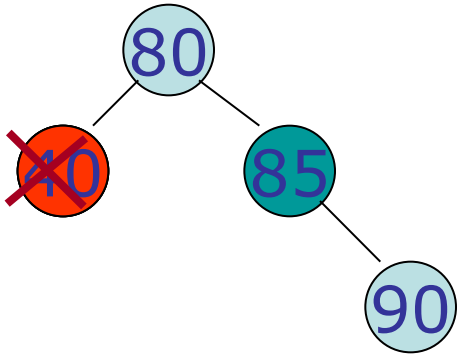
BF(y)=0

Single LL rotations

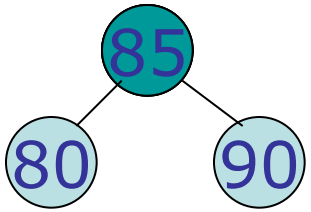


BF(y)= -1
height=3

After delete 40,…



Single RR rotations



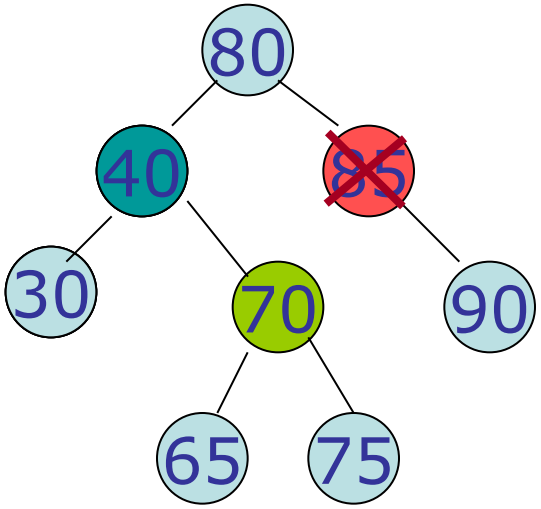
BF(y)= -1

BF(y)=0
height=2

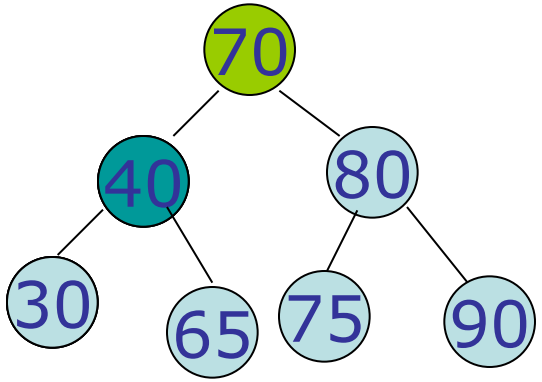
Rotations in Deletion

- There are *4 cases* for single rotations, but we do not need to distinguish among them.
- There are exactly *two cases* for double rotations (*as in the cases of insertion*)
- Therefore, we can reuse exactly the same procedure for insertion to determine which rotation to perform!

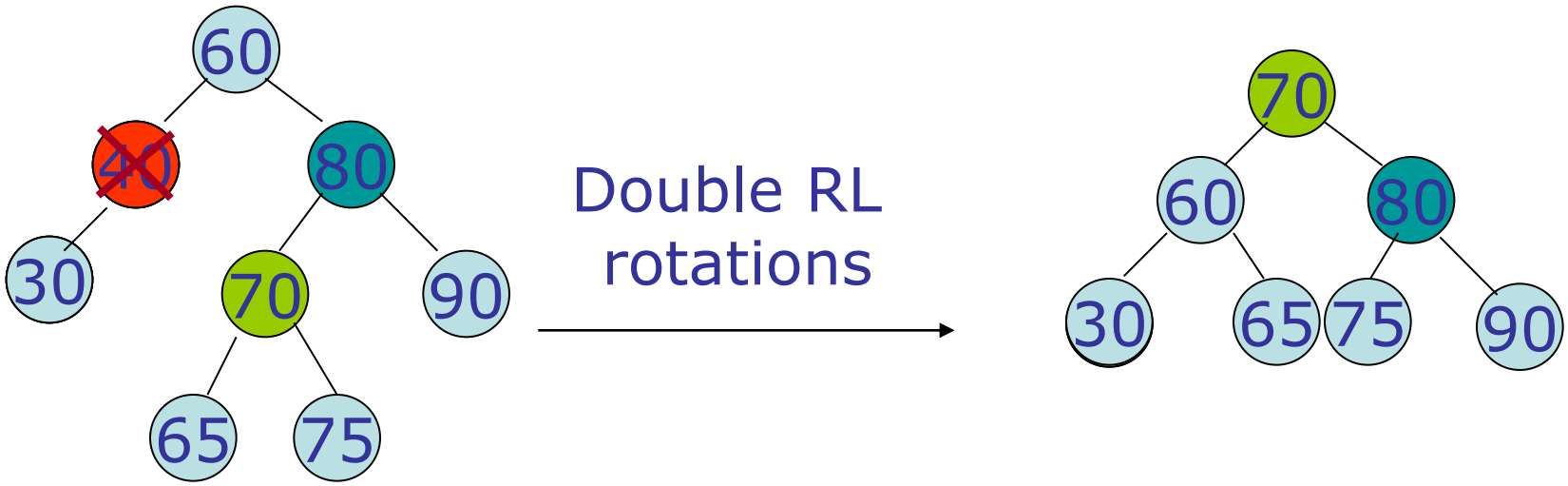
After delete 85,…



Double LR rotations



After delete 40,…



Practice 2: *Deleting* from AVL Tree

- After the deletions of 30, 10, 60