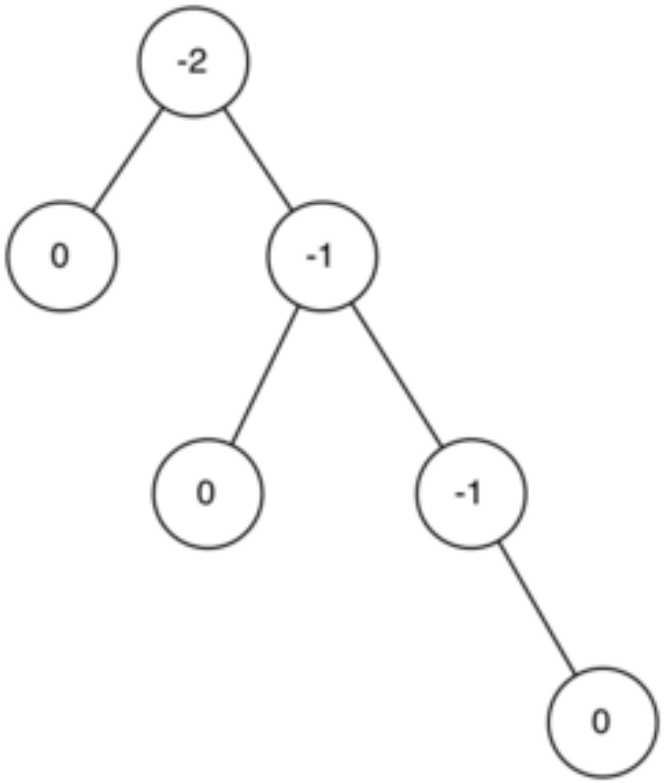
ต้นไม้เอวีแอล
AVL Tree

ADT

- มี method เหมือนกับ binary search tree
- ต่างกันที่การทำงานข้างใน method
  - คำนึงถึงค่าความสมดุล (balance factor)

Balance factor

- balanceFactor = height(lestSubTree) – height(rightSubTree)
- balance factor > 0 ต้นไม้หนักไปทางฝั่งซ้าย
- balance factor < 0 ต้นไม้หนักไปทางฝั่งขวา
- balance factor = 0 ต้นไม้สมดุลแบบสมบูรณ์
- AVL tree
  - balance factor = -1, 0, 1
  - ถ้าหลุดออกจากค่าพวกนี้ต้องทำการแก้ไข
  - ทำงานได้เร็วขึ้น



เพิ่ม node

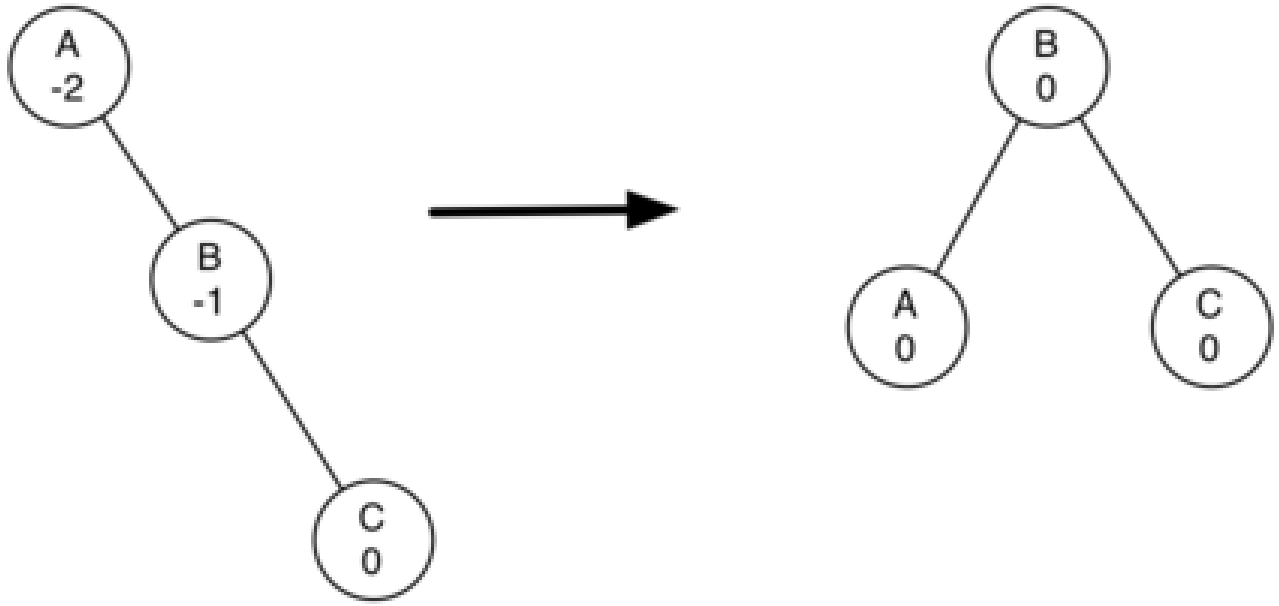
- เพิ่ม left node
  - parent และ ต้นตระกูลมีค่า balance factor เพิ่ม 1
- เพิ่ม right node
  - parent และ ต้นตระกูลมีค่า balance factor ลด 1

```
def _put(self,key,val,currentNode):
    if key < currentNode.key:
        if currentNode.hasLeftChild():
                self._put(key,val,currentNode.leftChild)
        else:
                currentNode.leftChild = TreeNode(key,val,parent=currentNode)
                self.updateBalance(currentNode.leftChild)
    else:
        if currentNode.hasRightChild():
                self._put(key,val,currentNode.rightChild)
        else:
                currentNode.rightChild = TreeNode(key,val,parent=currentNode)
                self.updateBalance(currentNode.rightChild)
```

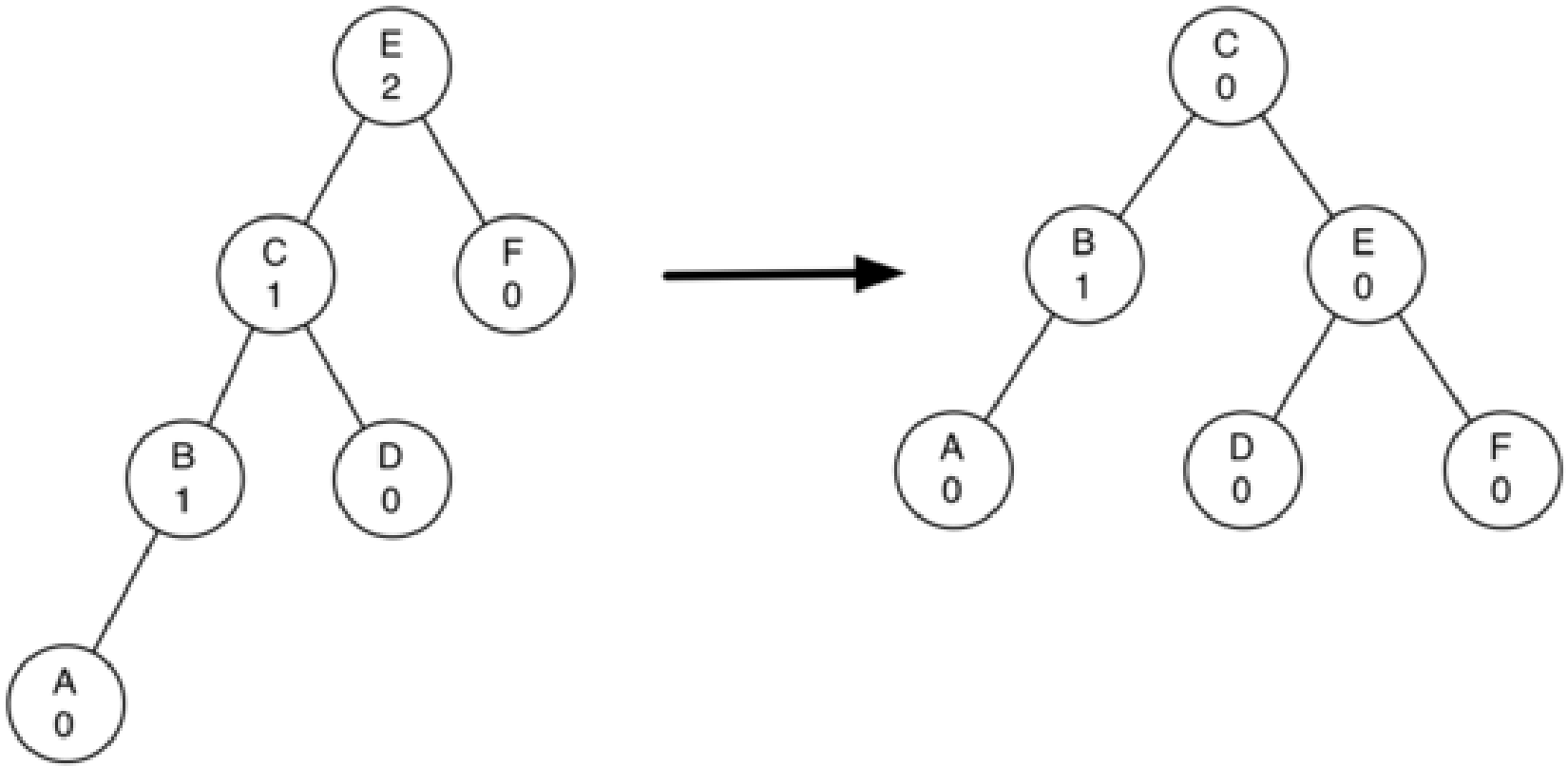
```
def updateBalance(self,node):
    if node.balanceFactor > 1 or node.balanceFactor < -1:
        self.rebalance(node)
        return
    if node.parent != None:
        if node.isLeftChild():
                node.parent.balanceFactor += 1
        elif node.isRightChild():
                node.parent.balanceFactor -= 1

        if node.parent.balanceFactor != 0:
                self.updateBalance(node.parent)
```

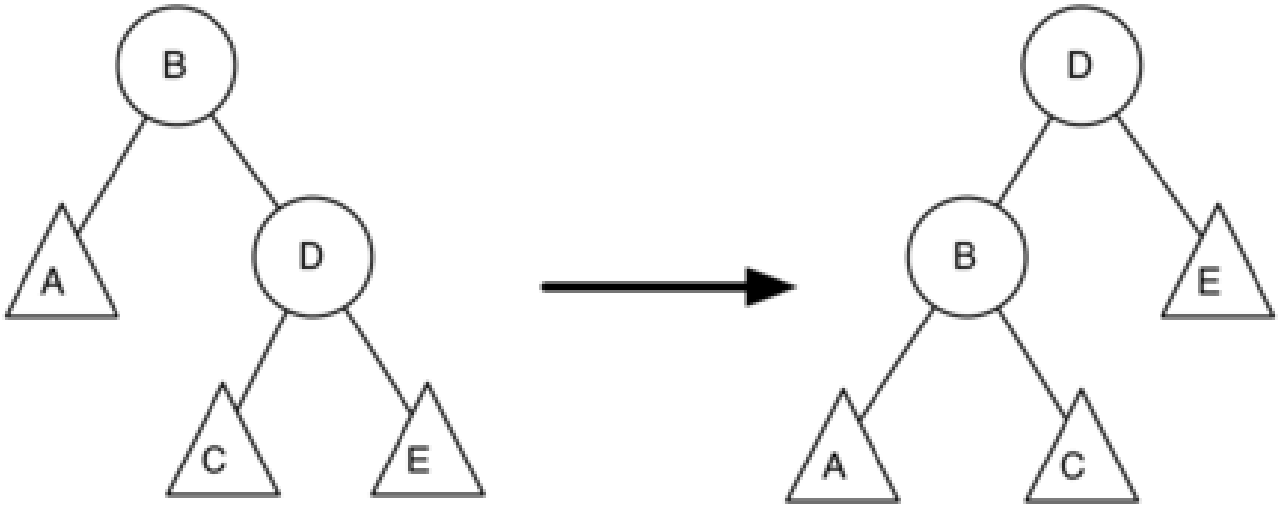
rebalance



- rotation
  - left rotation
    - ปรับ right child มาเป็น root ของ subtree
    - เอา root เดิมมาเป็น left child
    - ถ้า root ใหม่มี left child อยู่แล้วให้ปรับมาเป็น right child ของ left child ใหม่
  - right rotation
    - ปรับ left child มาเป็น root ของ subtree
    - เอา root เดิมมาเป็น right child
    - ถ้า root ใหม่มี right child อยู่แล้วให้ปรับมาเป็น left child ของ right child ใหม่


E
2
C
1
F
0
B
1
D
0
A
0
C
0
B
1
E
0
A
0
D
0
F
0

```
def rotateLeft(self,rotRoot):
    newRoot = rotRoot.rightChild
    rotRoot.rightChild = newRoot.leftChild
    if newRoot.leftChild != None:
        newRoot.leftChild.parent = rotRoot
    newRoot.parent = rotRoot.parent
    if rotRoot.isRoot():
        self.root = newRoot
    else:
        if rotRoot.isLeftChild():
                rotRoot.parent.leftChild = newRoot
        else:
            rotRoot.parent.rightChild = newRoot
    newRoot.leftChild = rotRoot
    rotRoot.parent = newRoot
    rotRoot.balanceFactor = rotRoot.balanceFactor + 1 - min(newRoot.balanceFactor, 0)
    newRoot.balanceFactor = newRoot.balanceFactor + 1 + max(rotRoot.balanceFactor, 0)
```



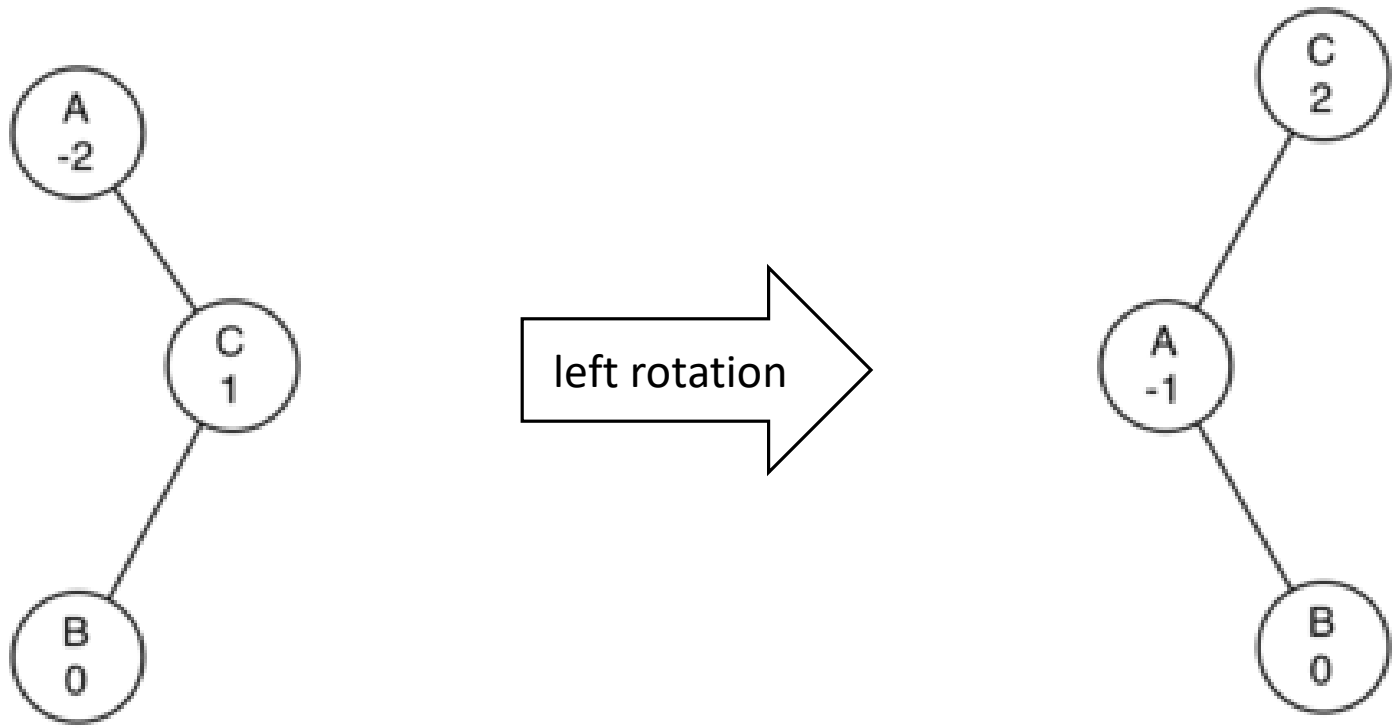
$$newBal(B) = h_A - h_C$$

$$oldBal(B) = h_A - h_D$$
$$oldBal(B) = h_A - (1 + \max(h_C, h_E))$$

$$newBal(B) - oldBal(B) = h_A - h_C - (h_A - (1 + max(h_C, h_E)))$$
$$newBal(B) - oldBal(B) = h_A - h_C - h_A + (1 + max(h_C, h_E))$$
$$newBal(B) - oldBal(B) = h_A - h_A + 1 + max(h_C, h_E) - h_C$$
$$newBal(B) - oldBal(B) = 1 + max(h_C, h_E) - h_C$$

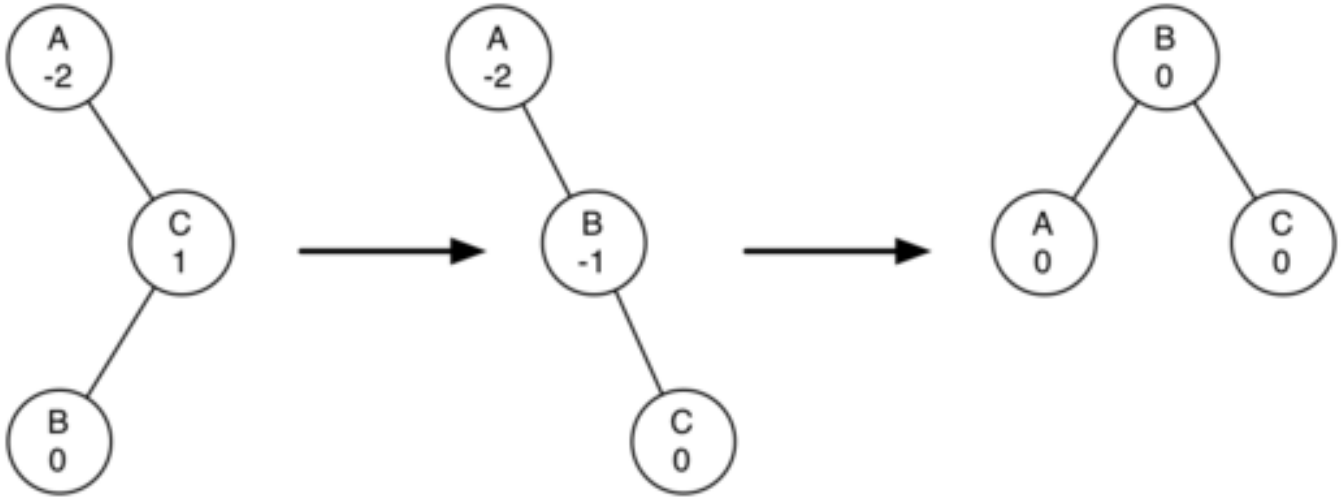
$$newBal(B) = oldBal(B) + 1 + max(h_C - h_C, h_E - h_C)$$

$$newBal(B) = oldBal(B) + 1 + max(0, -oldBal(D))$$
$$newBal(B) = oldBal(B) + 1 - min(0, oldBal(D))$$


A
-2
C
1
B
0
left rotation
C
2
A
-1
B
0

double rotation

- ทำ **left rotation**
  - ถ้า **right child** มี **balance factor > 0** (หนักไปทางฝั่งซ้าย) ให้ทำ **right rotation** ที่ **right child** ก่อน แล้วจึงทำ **left rotation** ปกติ
- ทำ **right rotation**
  - ถ้า **left child** มี **balance factor < 0** (หนักไปทางฝั่งขวา) ให้ทำ **left rotation** ที่ **left child** ก่อน แล้วจึงทำ **right rotation** ปกติ



```
def rebalance(self,node):
  if node.balanceFactor < 0:
        if node.rightChild.balanceFactor > 0:
           self.rotateRight(node.rightChild)
           self.rotateLeft(node)
        else:
           self.rotateLeft(node)
  elif node.balanceFactor > 0:
        if node.leftChild.balanceFactor < 0:
           self.rotateLeft(node.leftChild)
           self.rotateRight(node)
        else:
           self.rotateRight(node)
```