ต้นไม้เอวีแอล

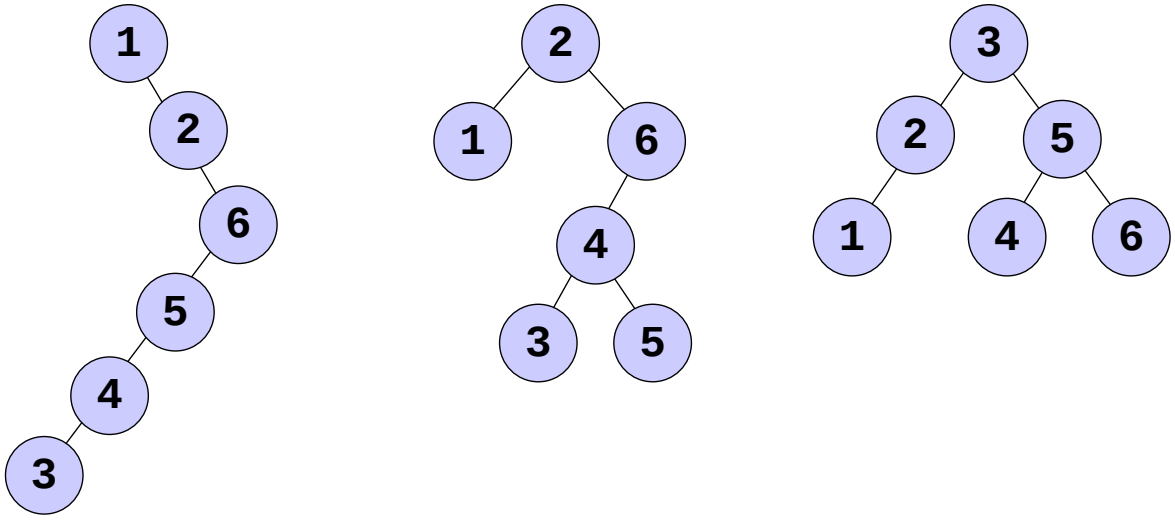
(AVL Tree)

หัวข้อ

- นิยามต้นไม้เอวีแอล
- การวิเคราะห์ความสูงของต้นไม้เอวีแอล
- การปรับต้นไม้เอวีแอลให้สูงสมดุล
- กระบวนการหมุนปม

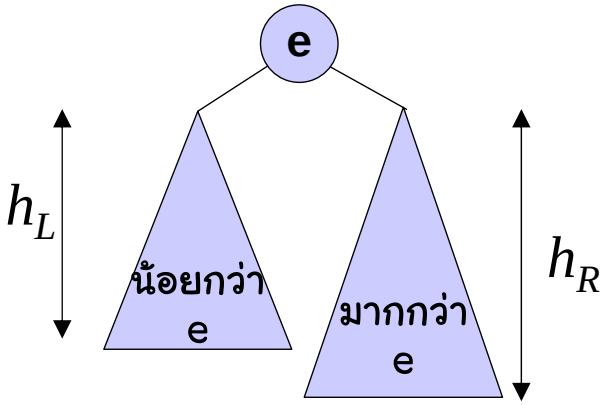
ต้นไม้ค้นหาแบบทวิภาค

- เวลาการทำงานเป็น $O(h)$
- $\lfloor \log_2 n \rfloor \leq h \leq n-1$
- โชคดีทำงานเร็ว $O(\log n)$, โชคร้ายทำงานช้า $O(n)$



ต้นไม้เอวีแอล

- AVL = Binary Search Tress + กฎความสูงสมดุล

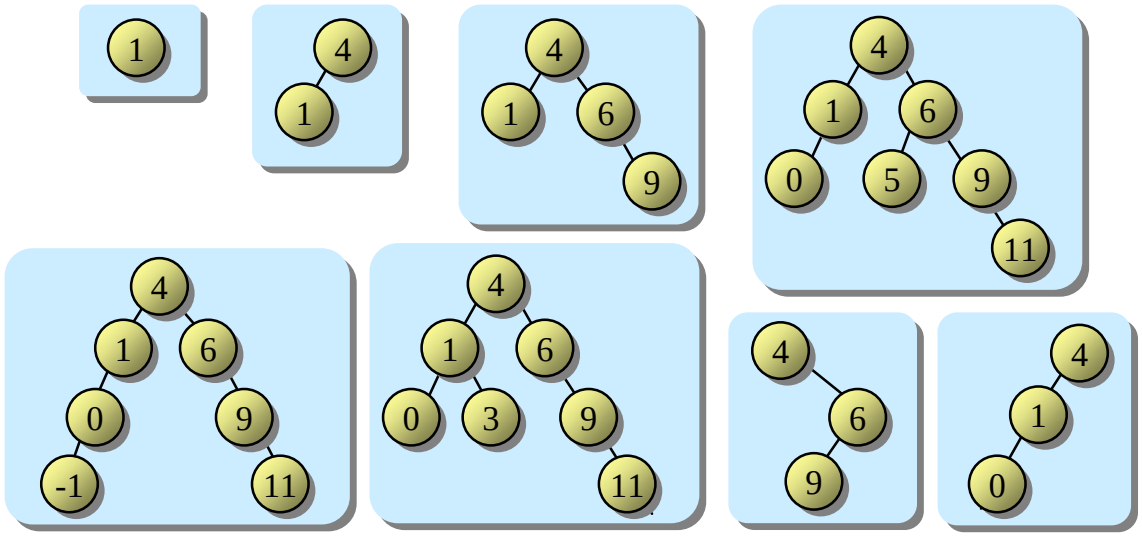


$$\left| h_L - h_R \right| \leq 1$$

ต้นไม้ย่อยทุกต้นต้องเป็นไปตามกฎ

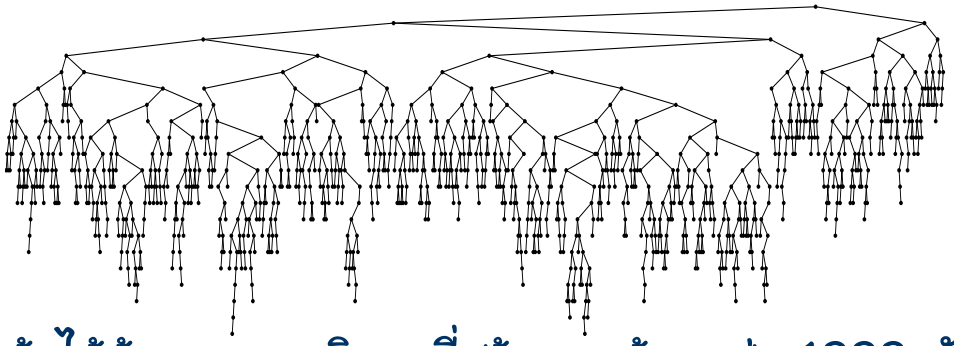
AVL : Adelson-Velskii and Landis

ตัวอย่างต้นไม้เอวีแอล

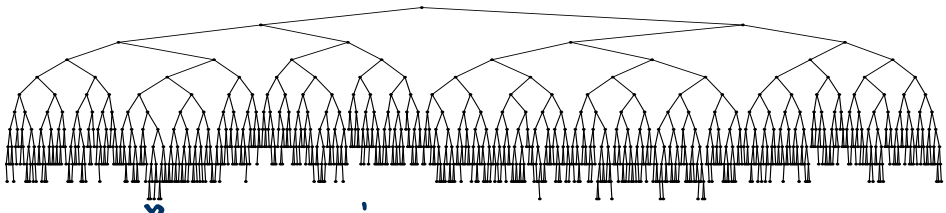


ต้นไม้ว่าง (null) สูง -1

ต้นไม้ค้นหาแบบทวิภาคกับเอวีแอล



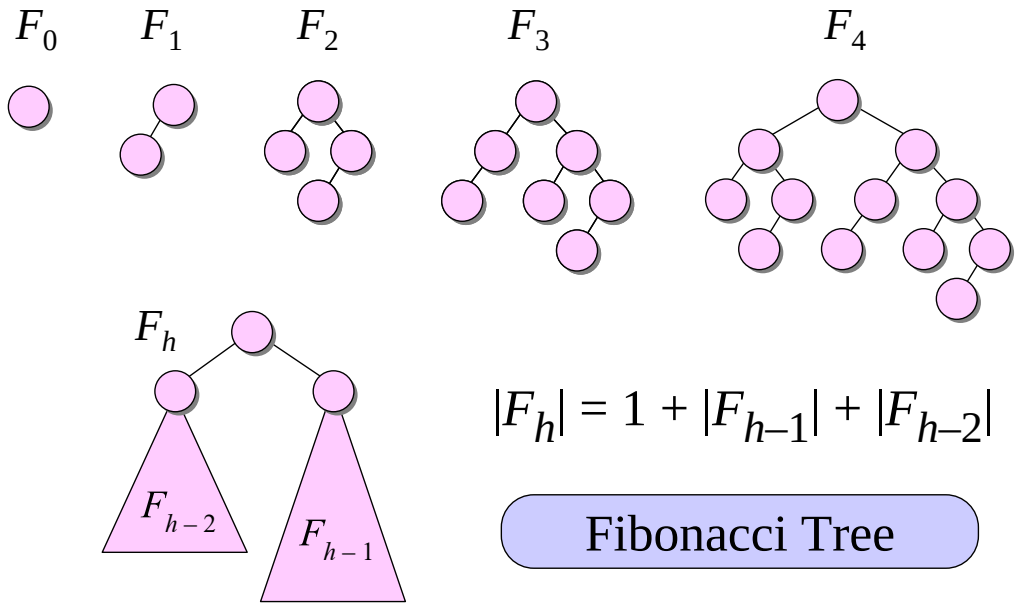
ต้นไม้ค้นหาแบบทวิภาคที่สร้างจากข้อมูลสุ่ม 1000 ตัว



ต้นไม้เอวีแอลที่สร้างจากข้อมูลสุ่ม 1000 ตัว

ต้นไม้เอวีแอลสูงเท่าใด ?

- ให้ F_h คือต้นไม้เอวีแอลซึ่งสูง h ที่มีจำนวนปมน้อยสุด



$$|F_h| = 1 + |F_{h-1}| + |F_{h-2}|$$

Fibonacci Tree

ความสูงของต้นไม้ฟิโบนักชี

$$|F_h| = 1 + |F_{h-1}| + |F_{h-2}|$$

$$n_h = 1 + n_{h-1} + n_{h-2} \quad h \geq 2,\ n_0 = 1, n_1 = 2$$

$$n_h = \alpha_1 \phi^h + \alpha_2 \hat{\phi}^h - 1,\ \phi = 1.618..., \hat{\phi} = -0.618$$

$$n_h \approx \alpha_1 \phi^h$$

$$h \approx \frac{1}{\log_2 \phi}\left(\log_2 n_h\right)$$

สรุป :
ต้นไม้เอวีแอลที่มี n ปม
สูงไม่เกิน 1.44 $\log_2 n$

$$h \approx 1.44\left(\log_2 n_h\right)$$

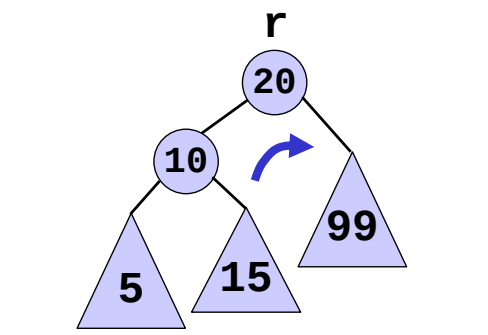
$$\lfloor \log_2 n \rfloor \leq h_{AVL} \leq 1.44 \log_2 n$$

ทำอย่างไรให้เป็นไปตามกฎของ AVL

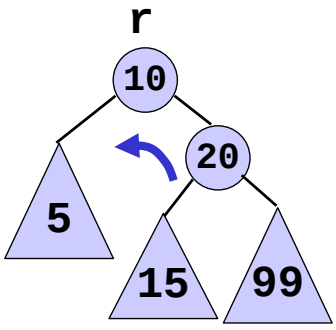
- การเพิ่ม/ลบข้อมูลทำเหมือน BSTree
- แต่หลังการเพิ่ม/ลบ อาจทำให้ผิดกฎสูงสมดุล
- ถ้าผิดกฎ ต้องปรับต้นไม้

การหมุนปม

- การปรับต้นไม้อาศัยการหมุน (rotation)
- การหมุนปมยังคงรักษาความเป็นต้นไม้ค้นหา



`rotateLeftChild(r)`

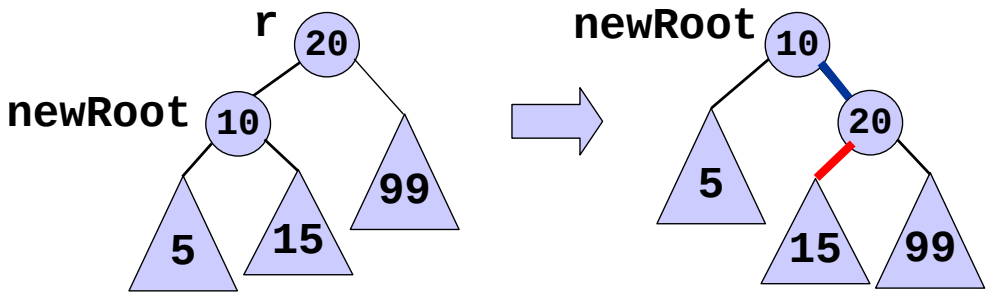


`rotateRightChild(r)`

rotateLeftChild(r)

```
public class BSTree extends BinaryTree {
  ...
  Node rotateLeftChild(Node r) {
    Node newRoot = r.left;
    r.left = newRoot.right;
    newRoot.right = r;
    return newRoot;
  }
  ...
```

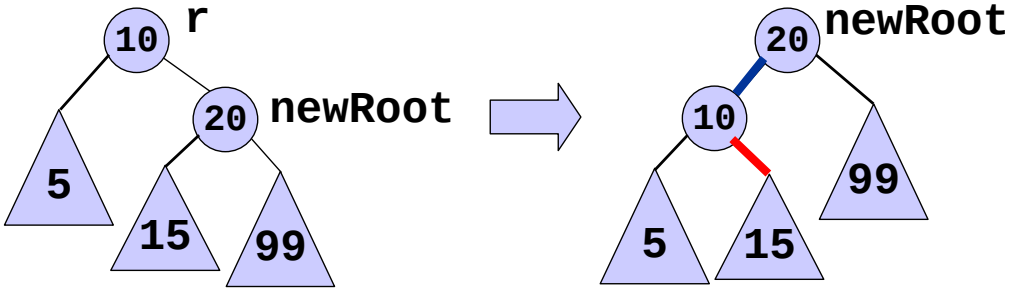
$\Theta(1)$



rotateRightChild(r)

```
public class BSTree extends BinaryTree {
  ...
  Node rotateRightChild(Node r) {
    Node newRoot = r.right;
    r.right = newRoot.left;
    newRoot.left = r;
    return newRoot;
  }
  ...
```

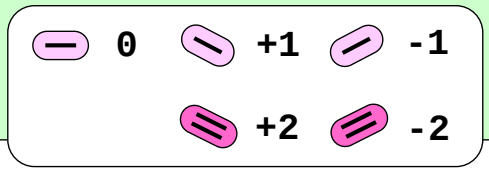
$\Theta(1)$



โครงสร้างปมของต้นไม้เอวีแอล

```
public class AVLTree extends BSTree {
  private static class AVLNode extends Node {
    private int height;
    AVLNode (Object e, Node left, Node right) {
      super(e, left, right);
      setHeight();
    }
    void setHeight() {
      height = 1+Math.max(height(left),height(right));
    }
    int height(Node n) {
      return (n == null ? -1 : ((AVLNode) n).height);
    }
    int balanceValue() {
      return height(right) - height(left);
    }
  }
  ...
```

แต่ละปมมีความสูงกำกับ



add และ remove ใช้ rebalance

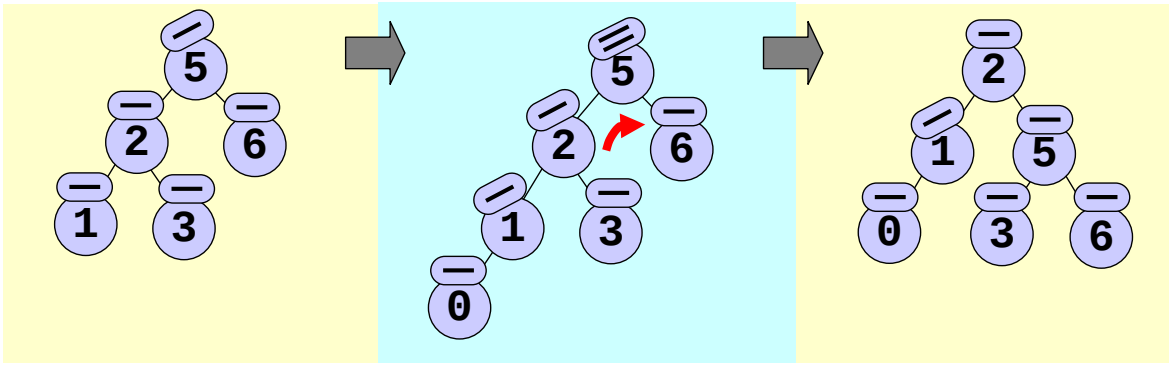
```
public class AVLTree extends BSTree {
  ...
  Node add(Node r, Object e) {
    if (r == null) {
      r = new AVLNode(e, null, null);
      ++size;
    } else {
      r = super.add(r,e);
      r = rebalance(r);
    }
    return r;
  }
  Node remove(Node r, Object e) {
    r = super.remove(r,e);
    r = rebalance(r);
    return r;
  }
```

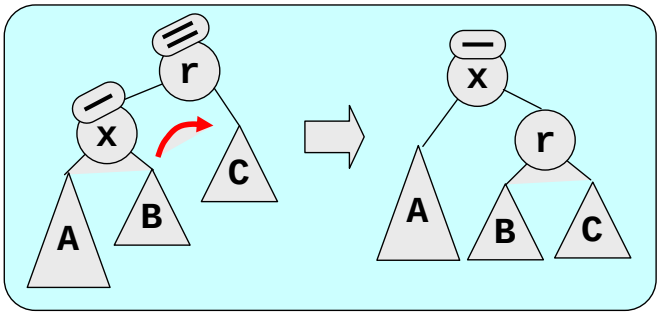
เพิ่มตามปกติ แล้วค่อยปรับ

rebalance มี 4 กรณี

ลบตามปกติ แล้วค่อยปรับ

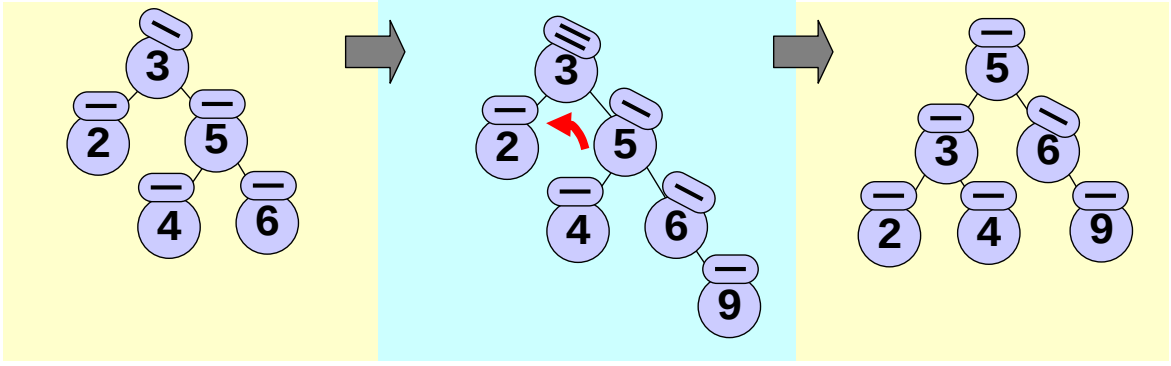
rebalance : กรณีที่ 1

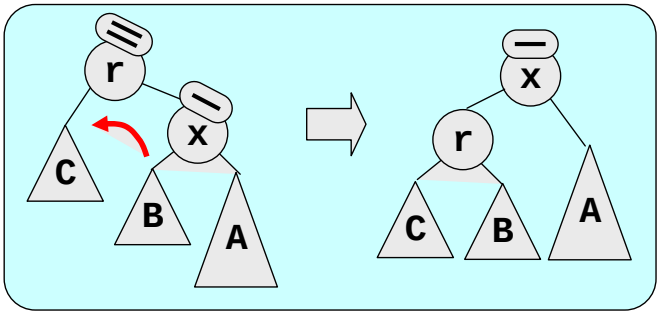




`rotateLeftChild(r)`

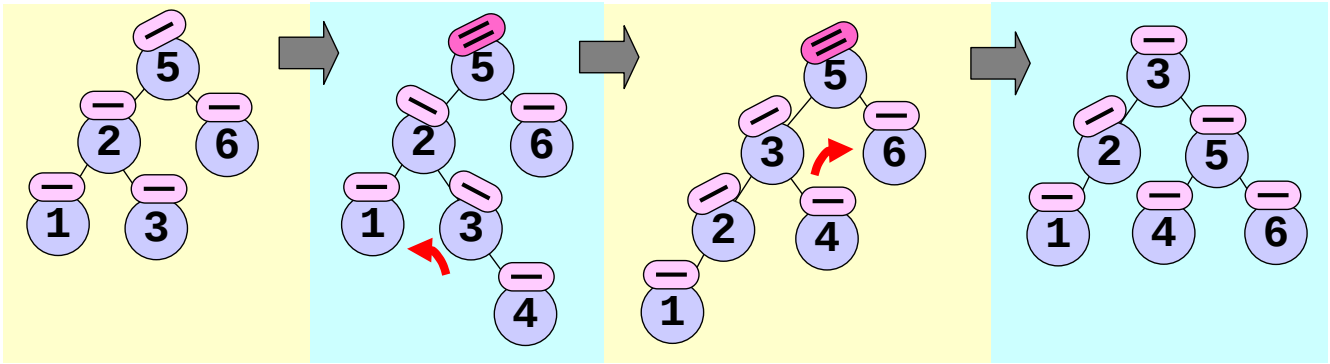
rebalance : กรณีที่ 2

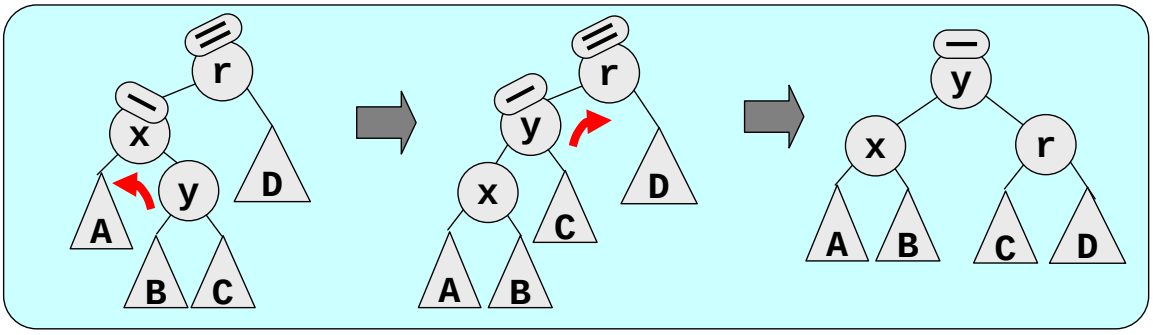




`rotateRightChild(r)`

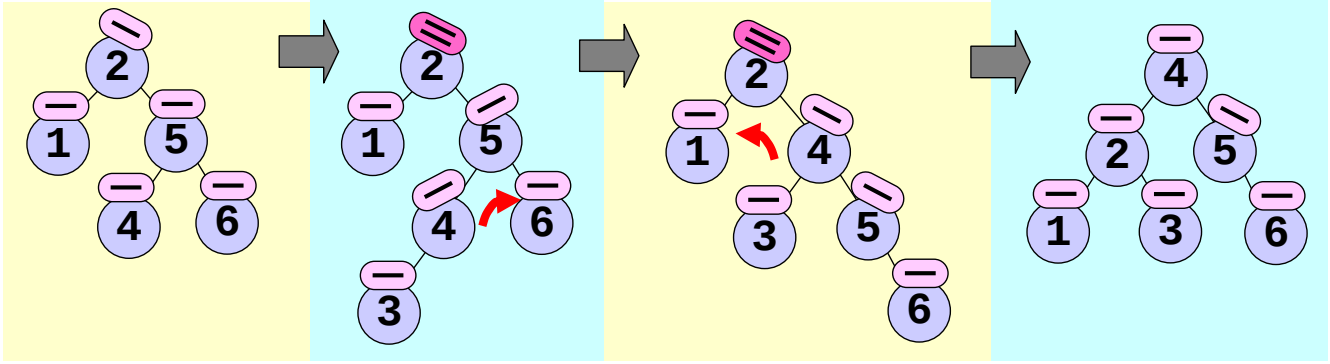
rebalance : กรณีที่ 3

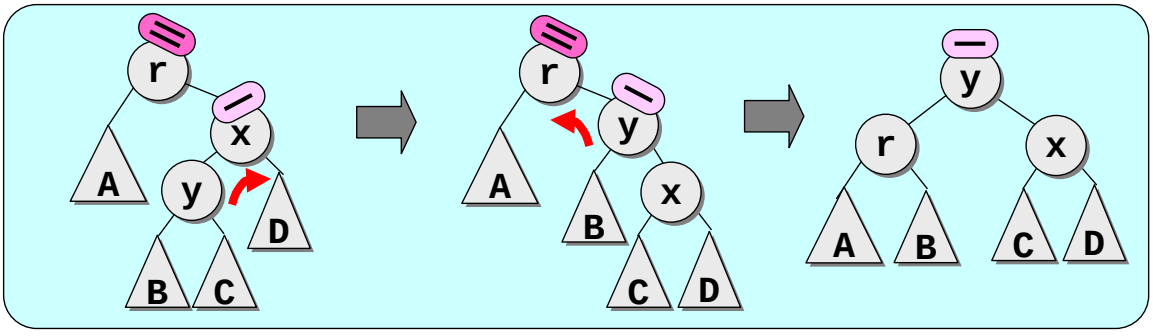




`rotateRightChild(r.left)` `rotateLeftChild(r)`

rebalance : กรณีที่ 4





`rotateLeftChild(r.right)` `rotateRightChild(r)`

rebalance

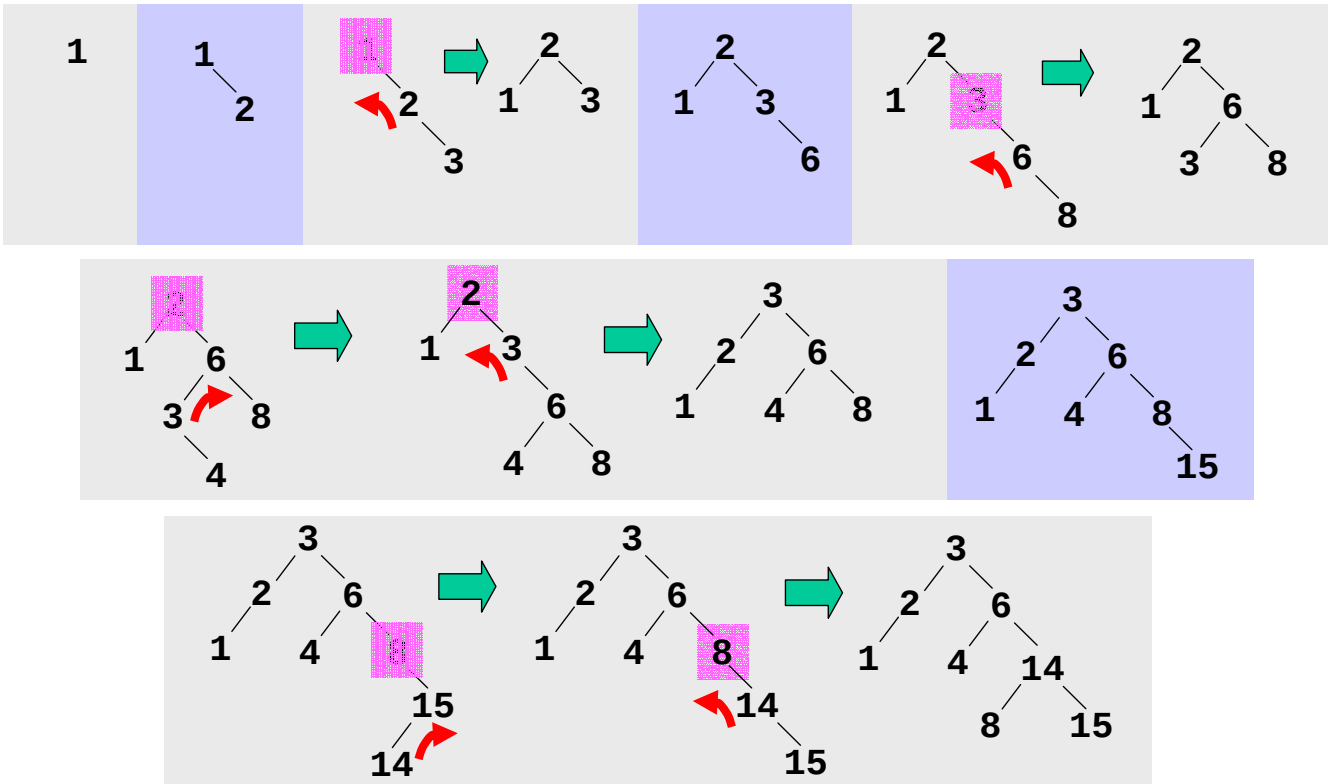
```
  private Node rebalance(Node r) {
    if (r == null) return r;
    int balance = ((AVLNode)r).balanceValue();
    if (balance == -2) {
      if (((AVLNode)r.left).balanceValue() == 1)
        r.left = rotateRightChild(r.left);
      r = rotateLeftChild(r);
    } else if (balance == 2) {
      if (((AVLNode)r.right).balanceValue() == -1)
        r.right = rotateLeftChild(r.right);
      r = rotateRightChild(r);
    }
    ((AVLNode)r).setHeight();
    return r;
  }
```


อย่าลืมปรับความสูงหลังการหมุน

```
public class AVLTree extends BSTree {
  ...
  Node rotateLeftChild(Node r) {
    r = super.rotateLeftChild(r);
    ((AVLNode) r.right).setHeight();
    ((AVLNode) r).setHeight();
    return r;
  }
  Node rotateRightChild(Node r) {
    r = super.rotateRightChild(r);
    ((AVLNode) r.left).setHeight();
    ((AVLNode) r).setHeight();
    return r;
  }
}
```

ตัวอย่าง

1, 2, 3, 6, 8, 4, 15, 14



สรุป

- ต้นไม้เอวีแอลคือต้นไม้ค้นหาที่ถูกควบคุมความสูง
- ผลต่างความสูงของลูกสองข้างห้ามเกินหนึ่ง
- พิสูจน์ได้ว่า $\lfloor \log_2 n \rfloor \leq h < 1.44 \log_2 n$
- แต่ละปมเก็บความสูงไว้ตรวจสอบ
- ถ้าหลังเพิ่ม/ลบข้อมูลแล้วผิดกฎ, ให้ปรับต้นไม้
- การปรับต้นไม้อาศัยการหมุนปม
- เวลาการทำงานของการเพิ่ม ลบ และค้นเป็น $O(\log n)$