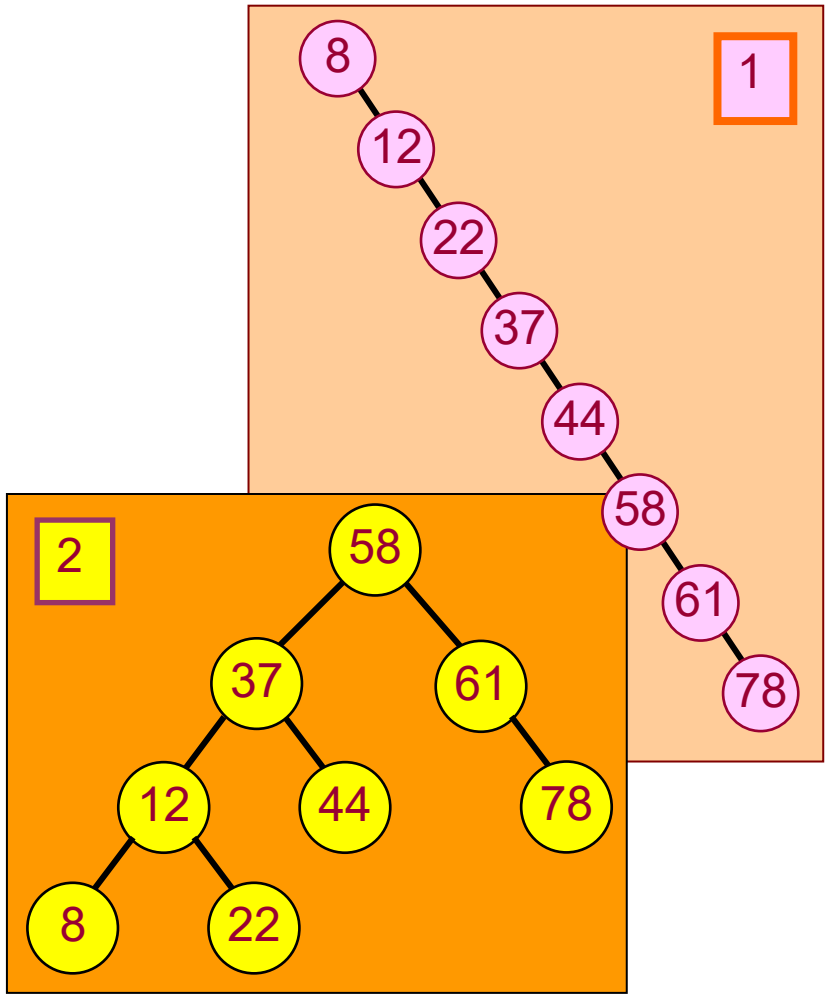
Chapter 5

AVL Tree

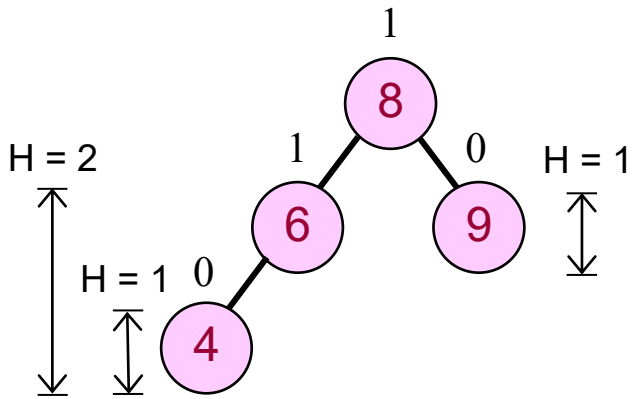
Binary Search Tree

- ปัญหา : BST อาจจะไม่สมดุล (เบ้ซ้าย-ขวา) ขึ้นกับลำดับของการแทรกข้อมูล
- ความไม่สมดุลนี้ ทำให้ประสิทธิภาพการค้นหาข้อมูลแย่ลง เช่น ถ้าต้องการค้นหาข้อมูล 78
  - ต้นไม้แบบแรก จะต้องเปรียบเทียบโหนดถึง 8 ครั้ง จึงพบโหนดที่มีค่า 78
  - ต้นไม้แบบที่สอง จะต้องเปรียบเทียบโหนดเพียง 3 ครั้ง จึงพบโหนดที่มีค่า 78
  - *แบบแรกค้นหาได้ช้ากว่าแบบที่สอง*



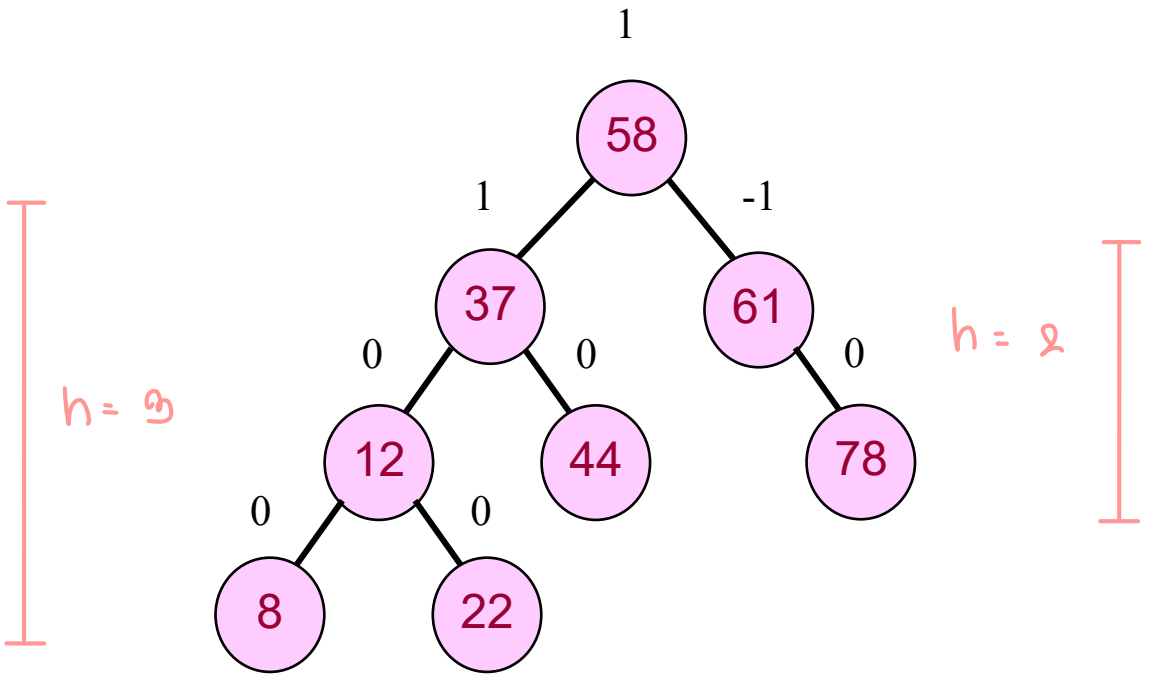
AVL Tree

- AVL Tree : เป็น BST ที่มีความสมดุล (Balance)
- ต้นไม้สมดุล : ความสูงของต้นไม้ย่อยข้างซ้ายและขวาต่างกันไม่เกิน 1 ระดับในทุกๆ โหนด ซึ่งสามารถกำหนดน้ำหนักความสูงเป็น -1, 0, 1
- น้ำหนักความสูง = ความสูงของ Left Subtree – ความสูงของ Right Subtree



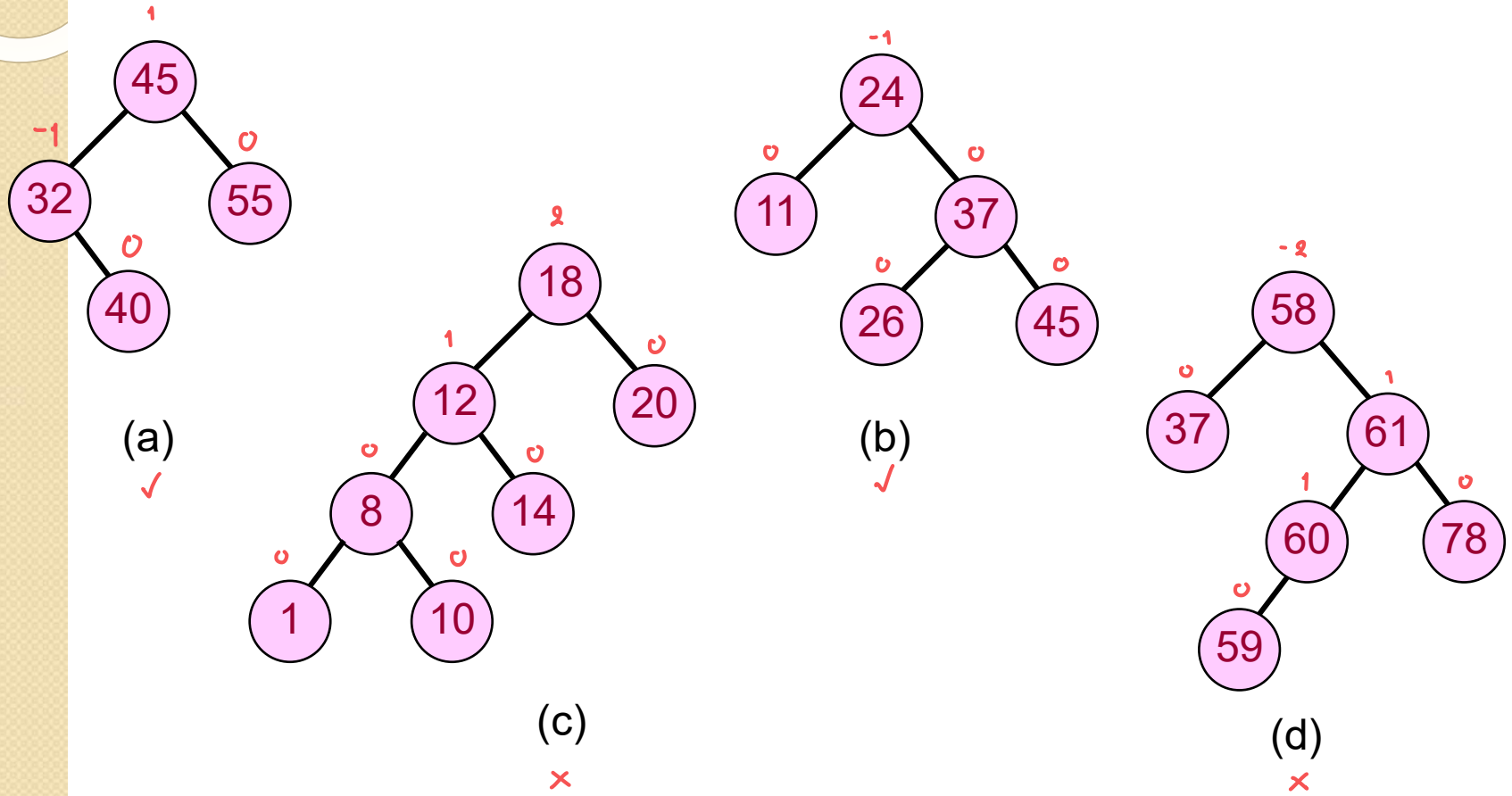
Example : AVL Tree

- ทุกโหนดมีค่าน้ำหนักความสูง -1 หรือ 0 หรือ 1



Example : AVL Tree

- ให้เขียนค่าน้ำหนักความสูงในทุกโหนด



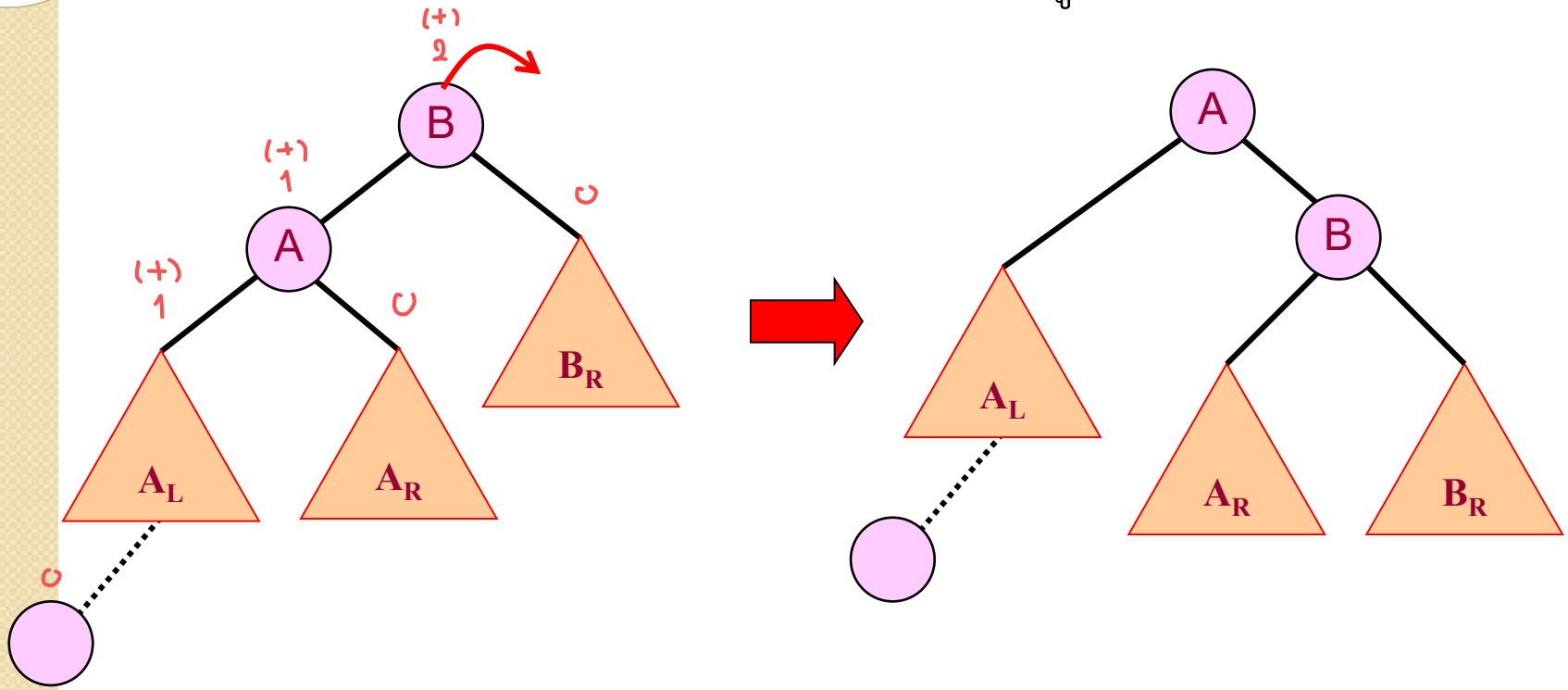
Balancing Trees

- เมื่อแทรกหรือลบโหนดที่ BST อาจเป็นทำให้ BST ไม่สมดุลได้
- เมื่อ BST ไม่สมดุลแล้ว เราสามารถปรับสมดุลของ BST ได้โดยการหมุนโหนดที่มีค่าน้ำหนักไม่เหมาะสม แบ่งเป็น 4 แบบ
  - หมุนขวา 1 ครั้ง
  - หมุนซ้าย 1 ครั้ง
  - หมุนซ้ายที่โหนดลูก แล้วหมุนขวาโหนดที่มีค่าน้ำหนักไม่เหมาะสม
  - หมุนขวาที่โหนดลูก แล้วหมุนซ้ายโหนดที่มีค่าน้ำหนักไม่เหมาะสม

Case 1 : Left to Left

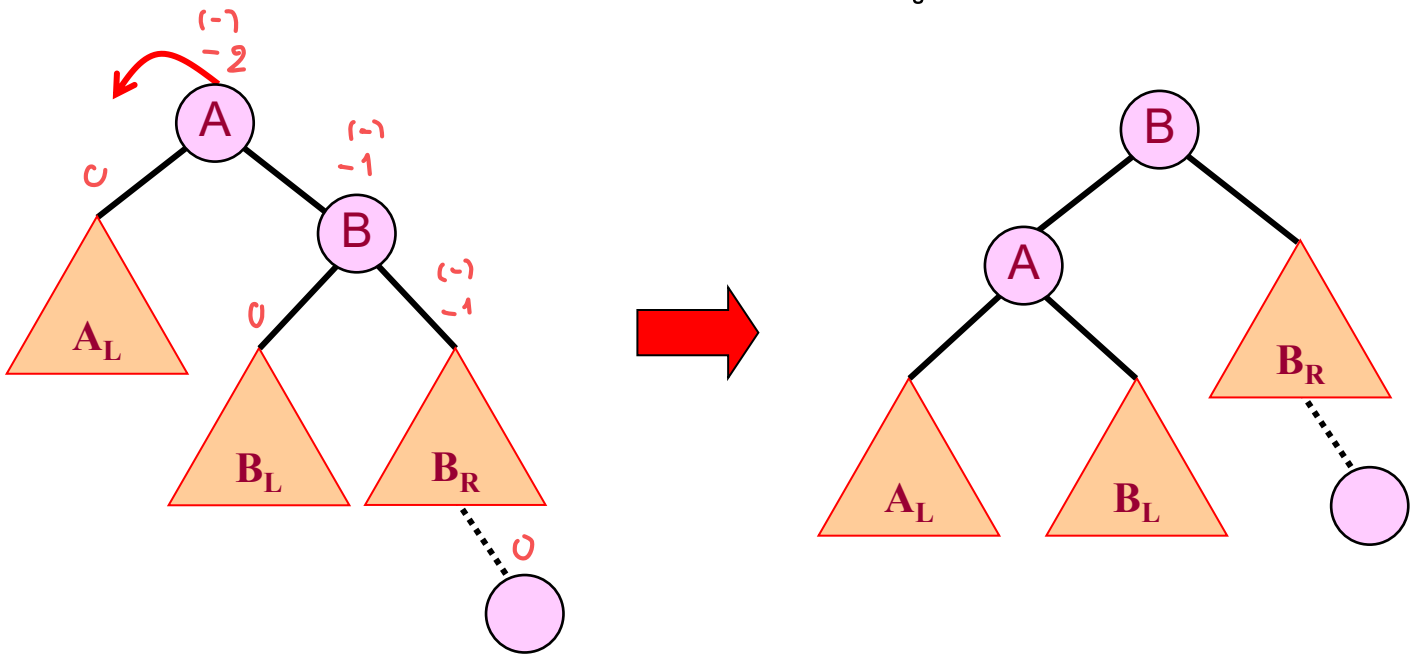
★ พิจารณา node ที่มีปัญหา
พิจารณาลูกของฝั่งที่เสียของ node ที่มีปัญหา

- หมุนขวา 1 ครั้ง (หมุนโหนด B)
  - โหนด B และ A เบ้ซ้าย (ค่าน้ำหนักเป็นบวกทั้งคู่)



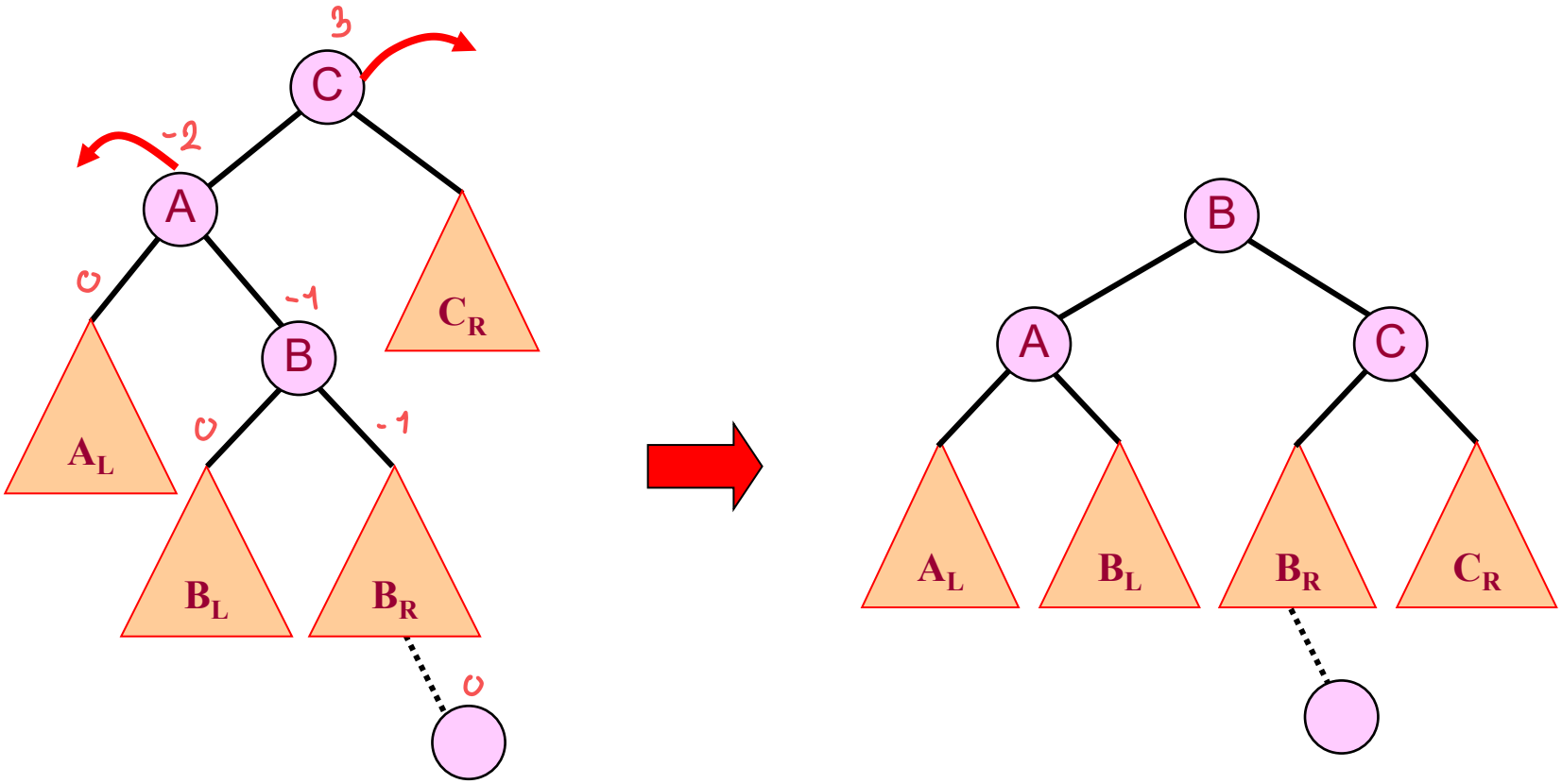
Case 2 : Right to Right

- หมุนซ้าย 1 ครั้ง (หมุนโหนด A)
  - โหนด A และ B เบ้ขวา (ค่าน้ำหนักเป็นลบทั้งคู่)



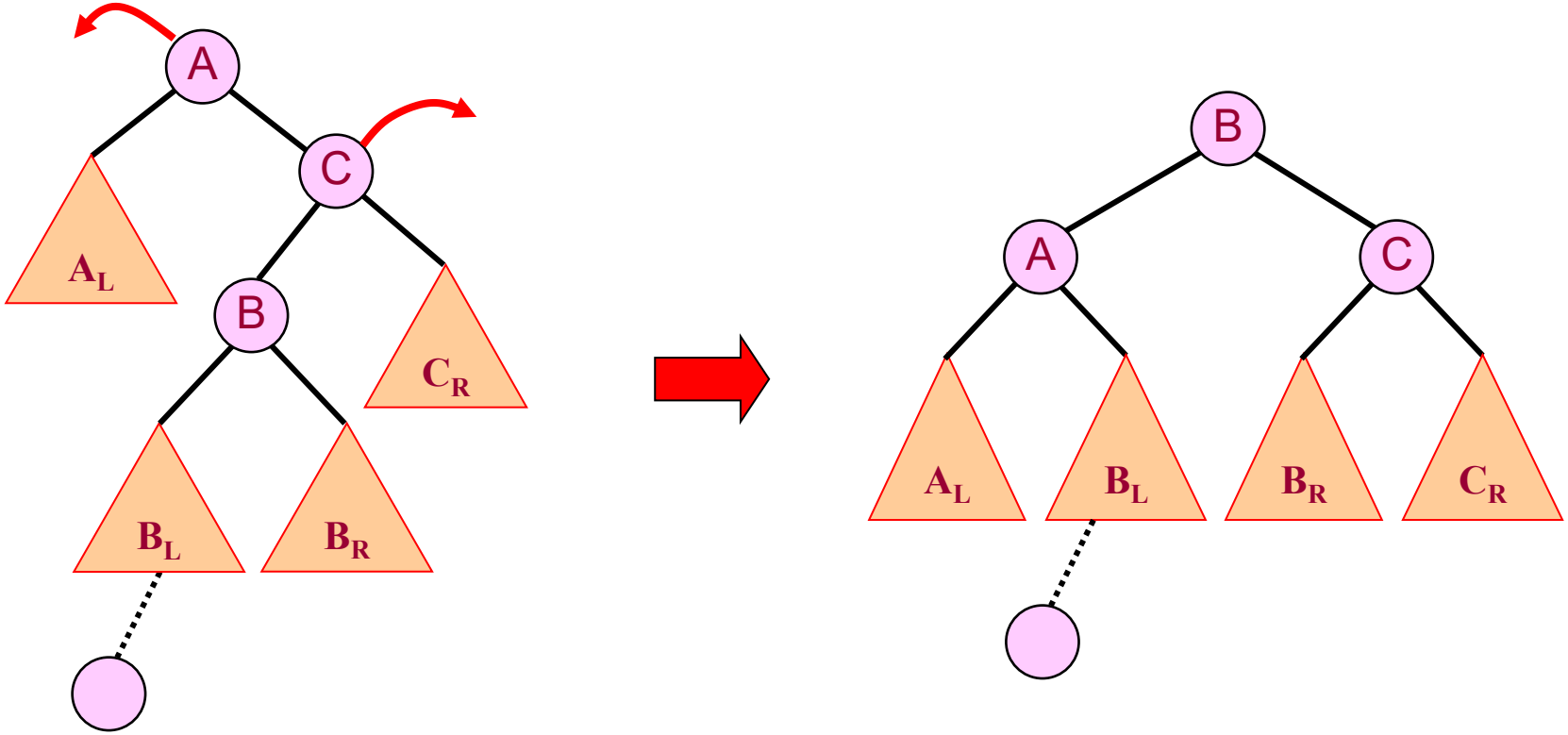
Case 3 : Right to Left

- หมุนซ้าย 1 ครั้ง (หมุนโหนด A) หมุนขวา 1 ครั้ง (หมุนโหนด C)
  - โหนด C เบ้ซ้าย และ A เบ้ขวา (ค่าน้ำหนัก C เป็นบวก A เป็นลบ)

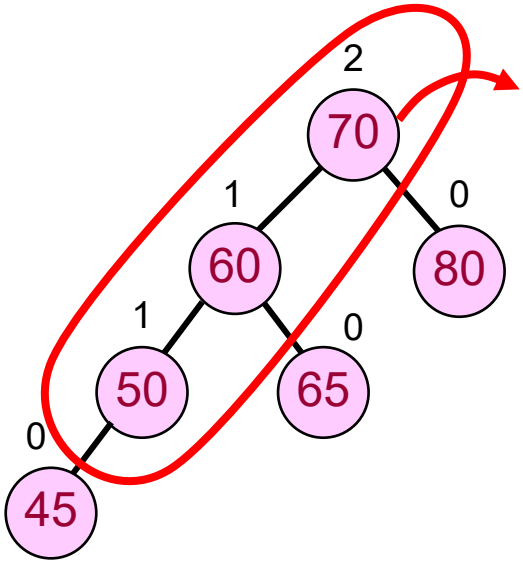


Case 4 : Left to Right

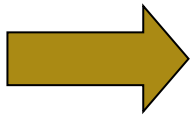
- หมุนขวา 1 ครั้ง (หมุนโหนด C) หมุนซ้าย 1 ครั้ง (หมุนโหนด A)
  - โหนด A เบ้ขวา และ C เบ้ซ้าย (ค่าน้ำหนัก C เป็นบวก A เป็นลบ)

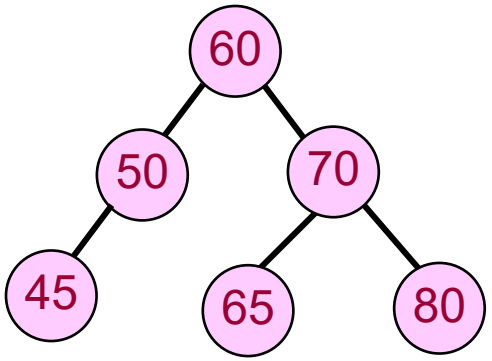


Example : Balancing Tree



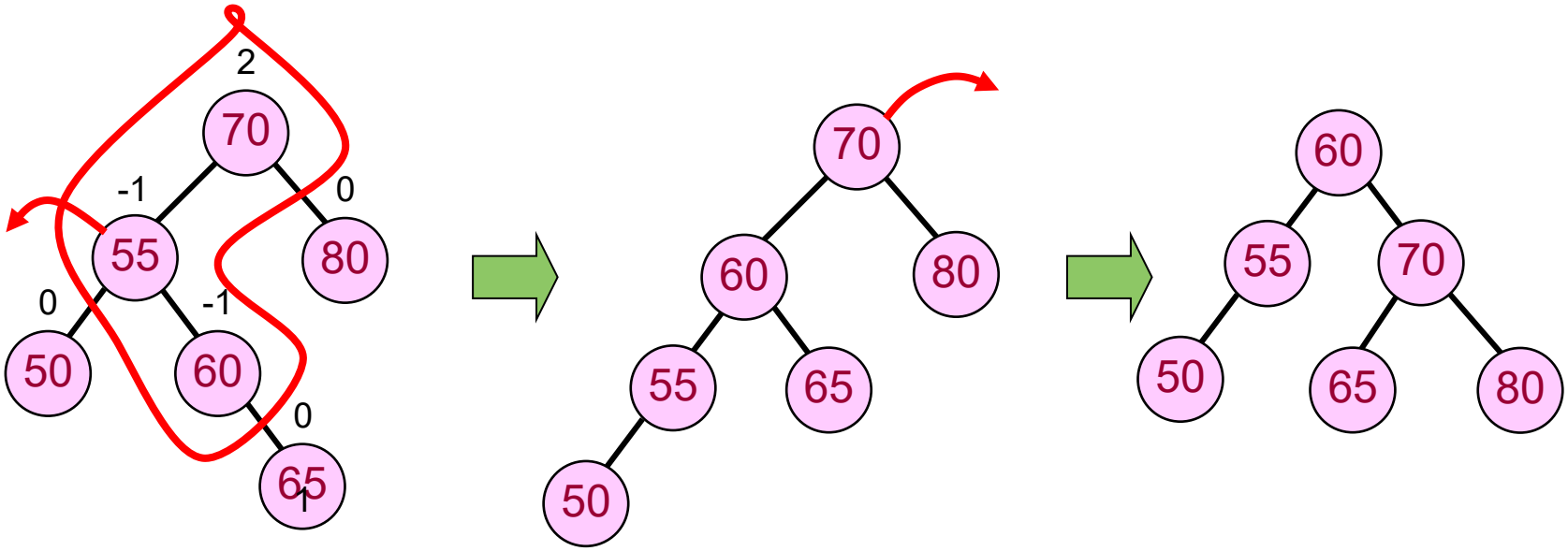
Case 1 : Left to Left





Example : Balancing Tree

Case 3 : Right to Left



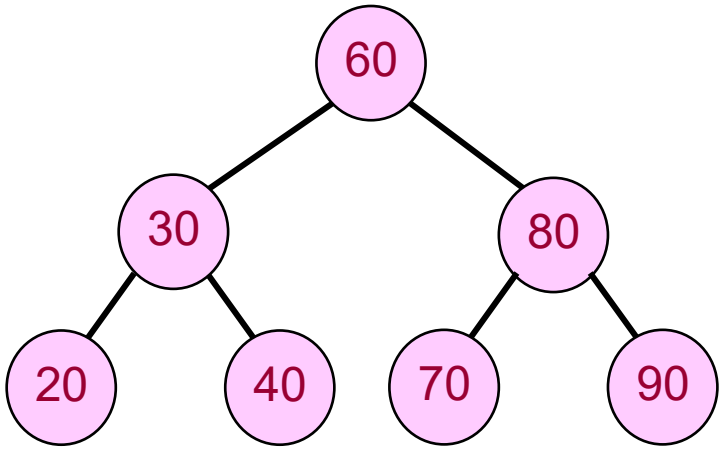
Insert in AVL Tree

- แทรกเหมือนที่ทำใน BST เพียงแต่เมื่อแทรกเสร็จแล้ว ถ้า BST ไม่สมดุล ก็ให้ปรับใหม่ตามกฎ

```
Algorithm AVLInsert (root, newData)
Using recursion, insert a node into an AVL tree.
   Pre    root is pointer to first node in AVL tree/subtree
          newData is pointer to new node to be inserted
   Post   new node has been inserted
   Return root returned recursively up the tree
 1 if (subtree empty)
   Insert at root
   1   insert newData at root
   2   return root
 2 end if
 3 if (newData < root)
   1   AVLInsert (left subtree, newData)
   2   if (left subtree taller)
       1   leftBalance (root)
   3   end if
 4 else
   New data >= root data
   1   AVLInsert (right subtree, newPtr)
   2   if(right subtree taller)
       1   rightBalance (root)
   3   end if
 5 end if
 6 return root
end AVLInsert
```

Quiz

- ให้เพิ่มค่า 25, 66 และ 97 เข้าไปใน AVL Tree ดังรูป แล้ววาดภาพผลลัพธ์ พร้อมแสดงค่าน้ำหนักความสูงในแต่ละโหนดด้วย



Quiz

- จงสร้าง AVL Tree เมื่อได้รับข้อมูลตามลำดับต่อไปนี้ (วาดภาพการแทรกข้อมูลทีละตัว พร้อมแสดงค่าน้ำหนักความสูงในแต่ละโหนดด้วย)

  11 27 5 9 44 59 88 70 75

- ให้ลบโหนดที่มีค่า 70 ออกจาก AVL Tree แล้ววาดภาพ AVL Tree ผลลัพธ์ พร้อมแสดงค่าน้ำหนักความสูงในแต่ละโหนดด้วย