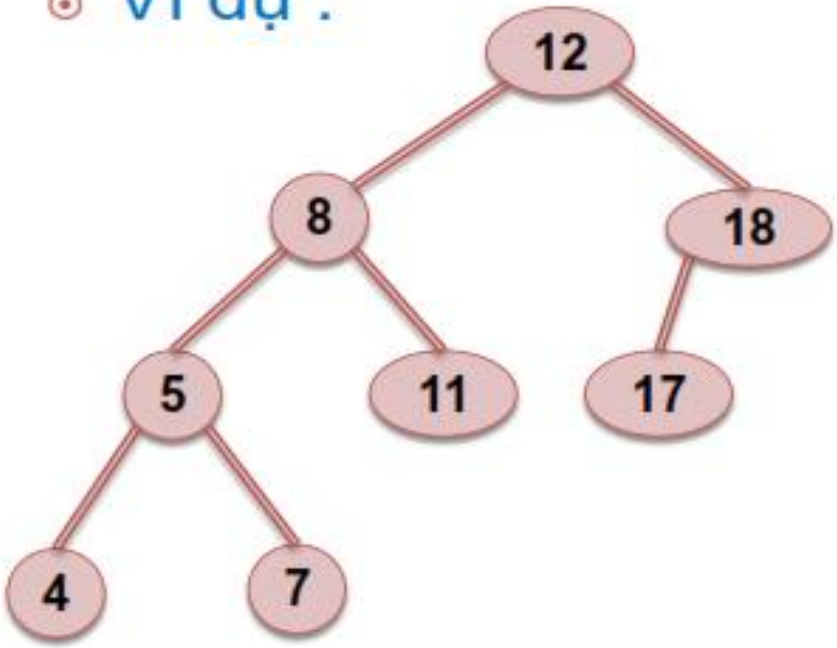
Cây AVL
(cây nhị phân tìm kiếm cân bằng)

Định nghĩa

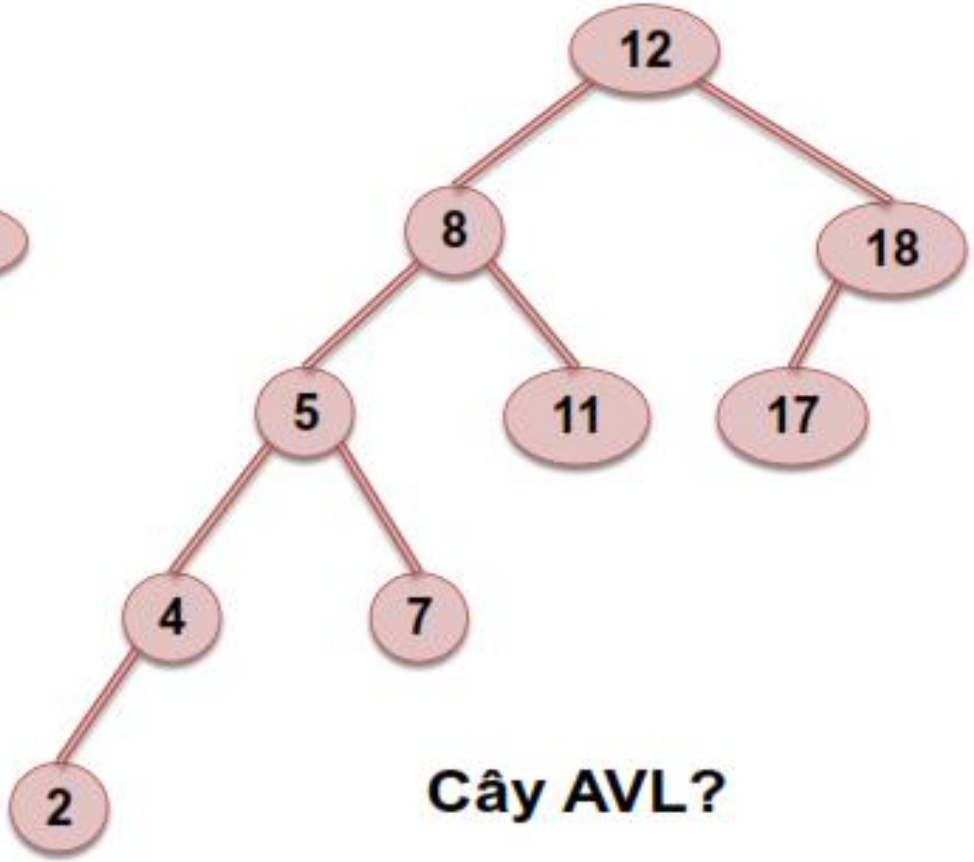
- Cây cân bằng AVL là cây nhị phân tìm kiếm mà tại mỗi đỉnh của cây, độ cao của cây con trái và cây con phải **không chênh lệch quá 1**.

Cây AVL

- Ví dụ :



Cây AVL?



Cây AVL?

Xây dựng cây cân bằng

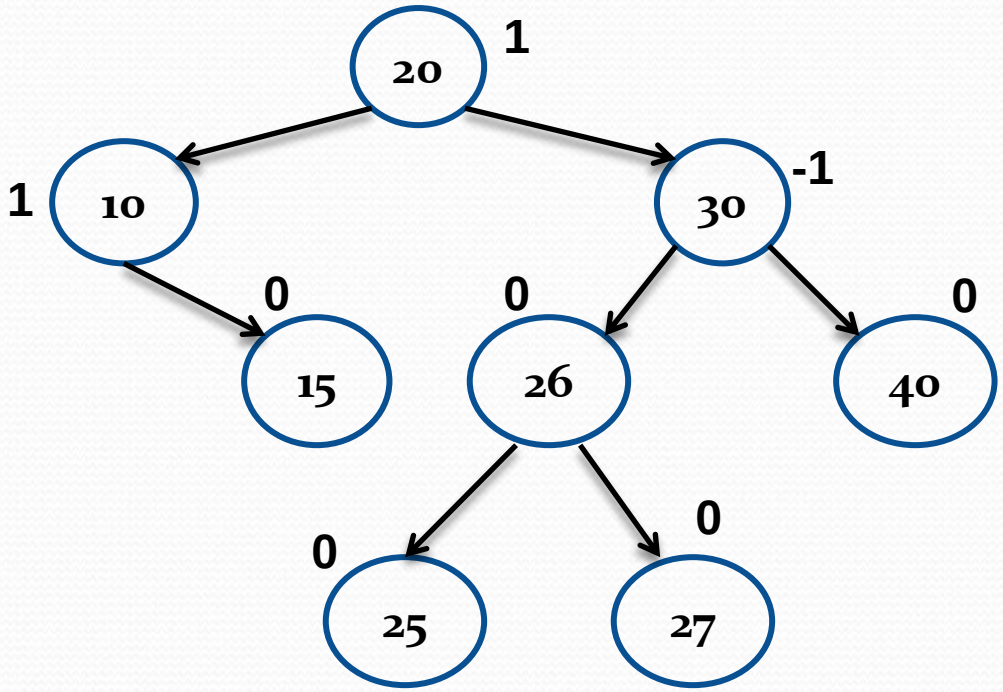
- Việc xây dựng cây cân bằng dựa trên cây nhị phân tìm kiếm, chỉ bổ sung thêm 1 giá trị cho biết sự cân bằng của các cây con như thế nào.
- Cách làm gợi ý:

```
struct NODE {
    Data key;
    NODE *pLeft, *pRight;
    int bal;
};
```

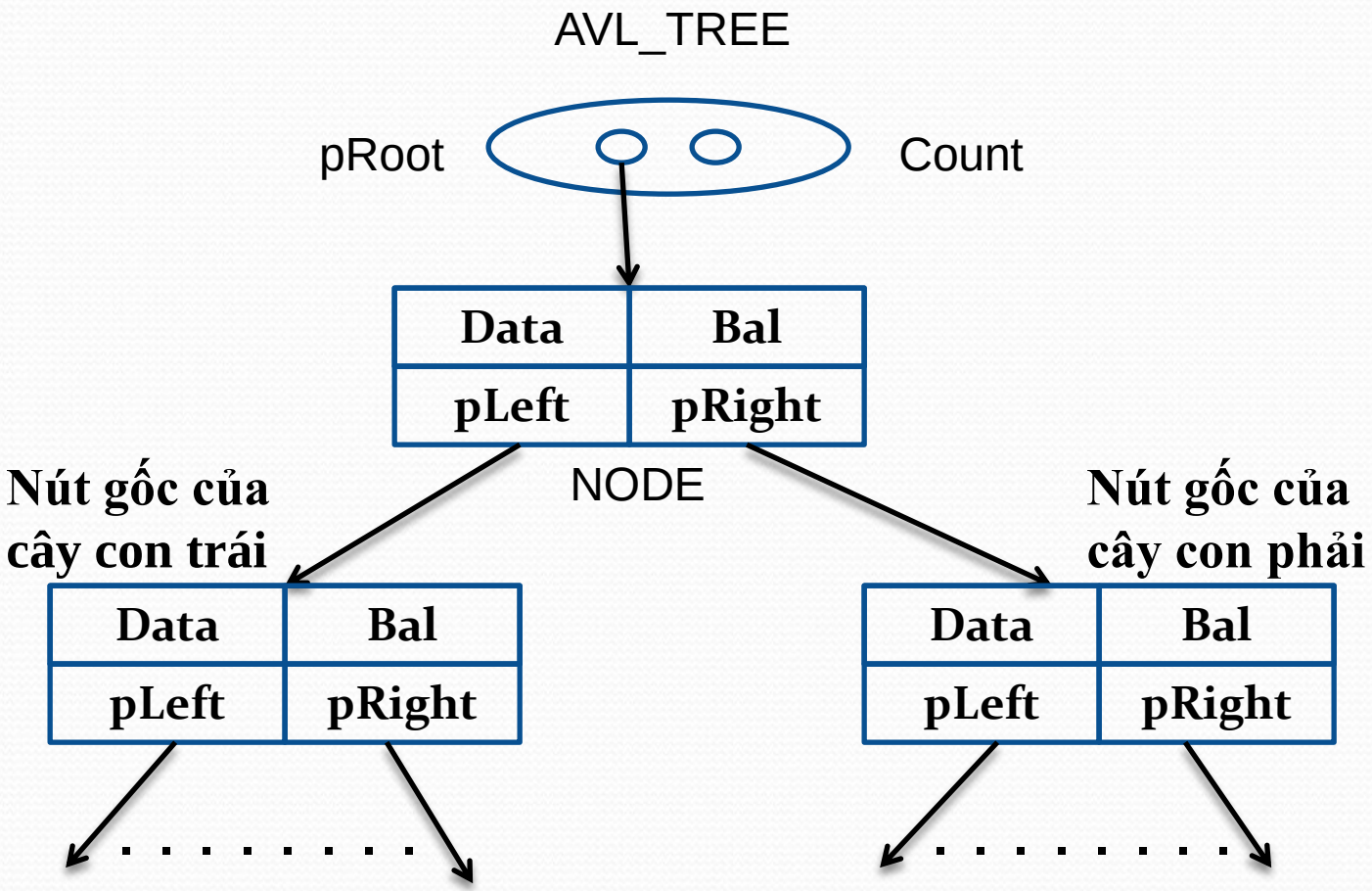
- Trong đó giá trị bal (balance, cân bằng) có thể là: 0: cân bằng; -1: lệch trái; 1: lệch phải

Mô tả cấu trúc cây AVL

Hệ số cân bằng của các nút trong cây AVL



Mô tả cấu trúc cây AVL



Mô tả cấu trúc cây AVL

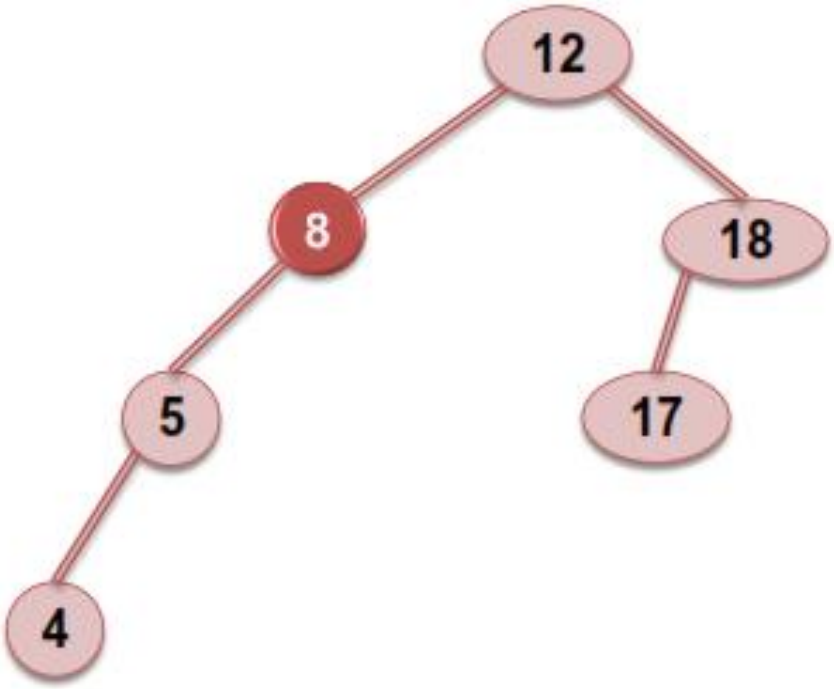
```
struct   NODE
{
      DataType  Data;
      NODE  * pLeft;
      NODE  *pRight;
      int  Bal;
};

Struct   AVLTREE
{
      int  Count;
      NODE  *pRoot;
};
```

Các trường hợp mất cân bằng

- Mất cân bằng trái-trái (L-L)
- Mất cân bằn trái-phải (L-R)
- Mất cân bằng phải-phải (R-R)
- Mất cân bằng phải-trái (R-L)

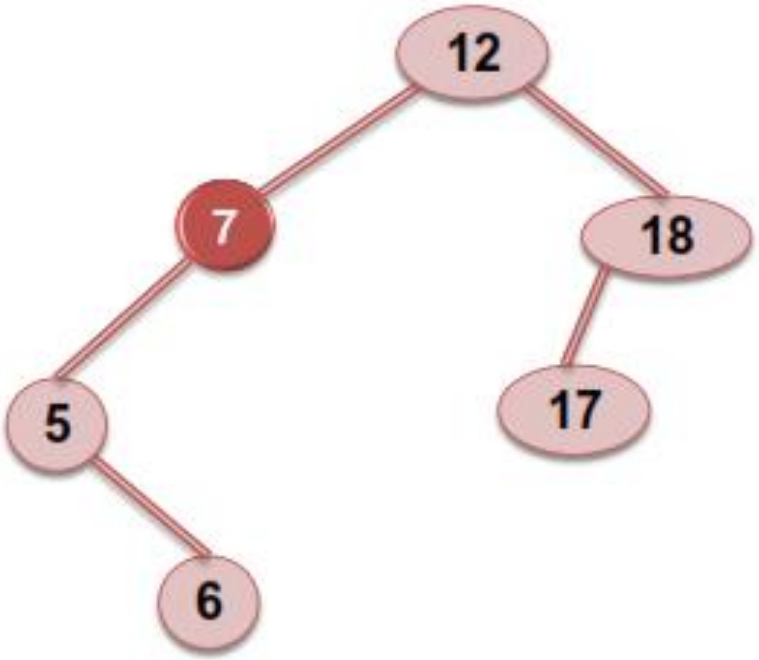
Các trường hợp mất cân bằng

- Mất cân bằng trái-trái (L-L)



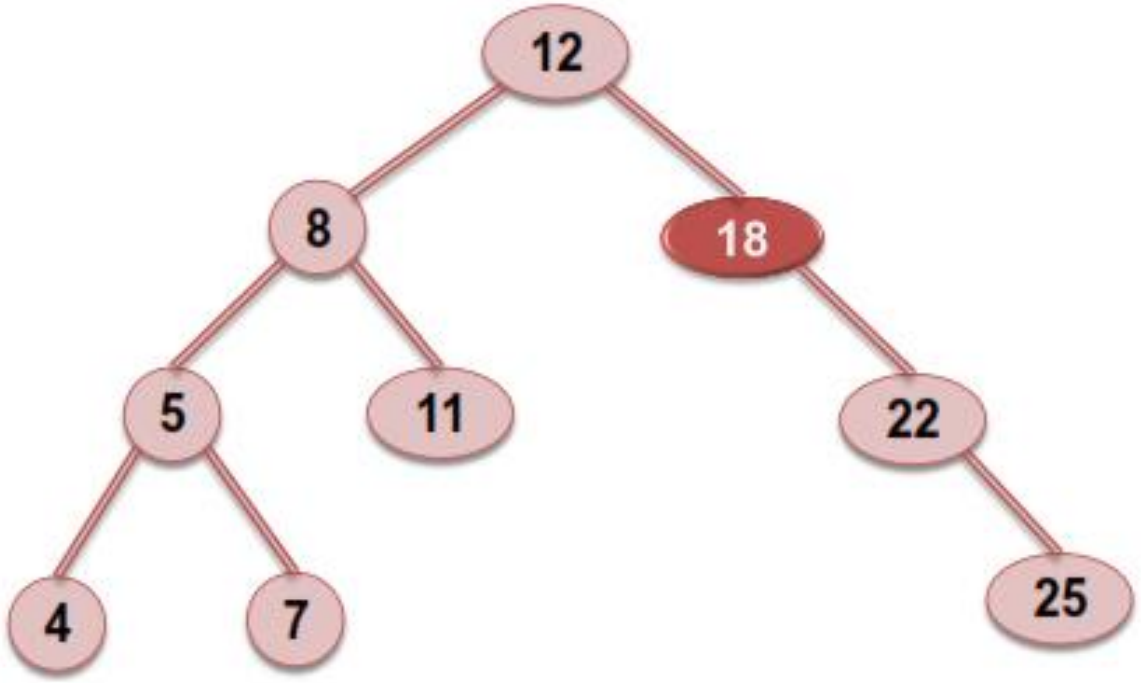
Các trường hợp mất cân bằng

- Mất cân bằng trái-phải (L-R)



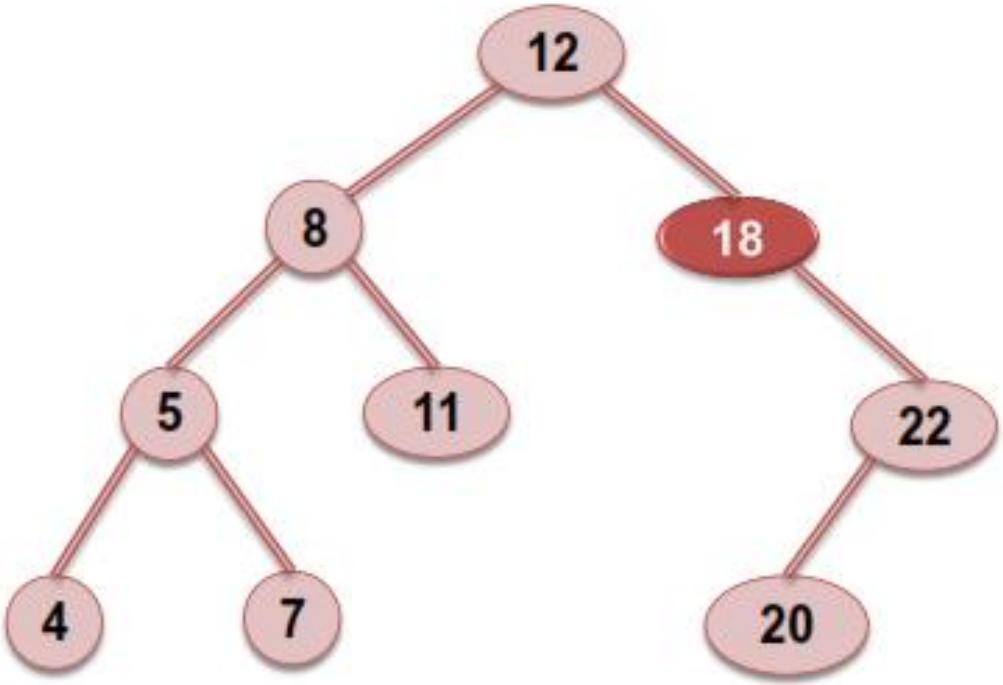
Các trường hợp mất cân bằng

- Mất cân bằng phải-phải (R-R)

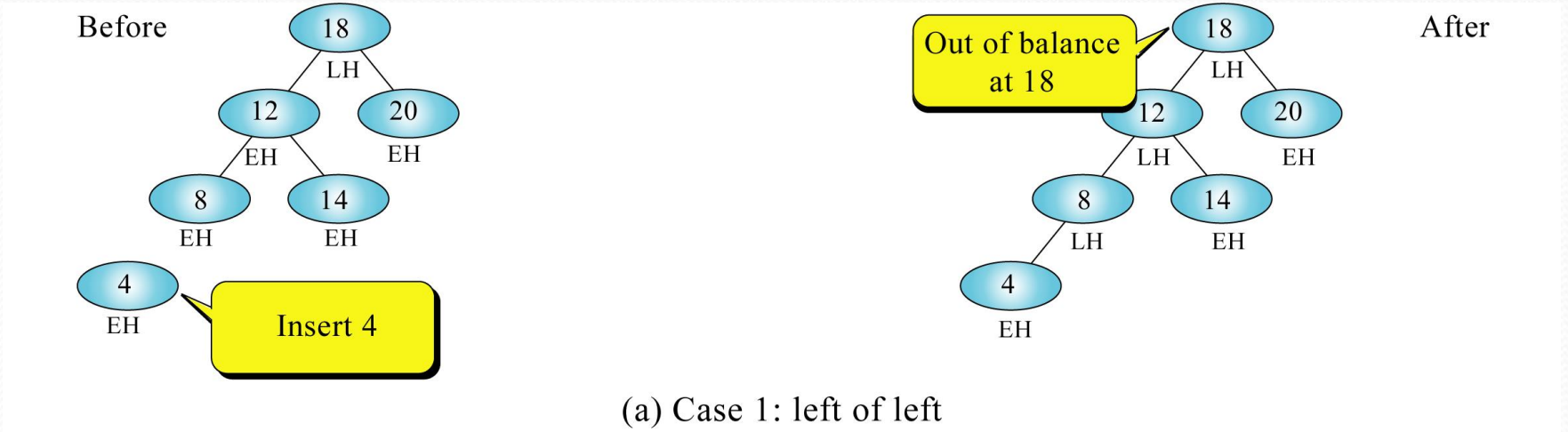


Các trường hợp mất cân bằng

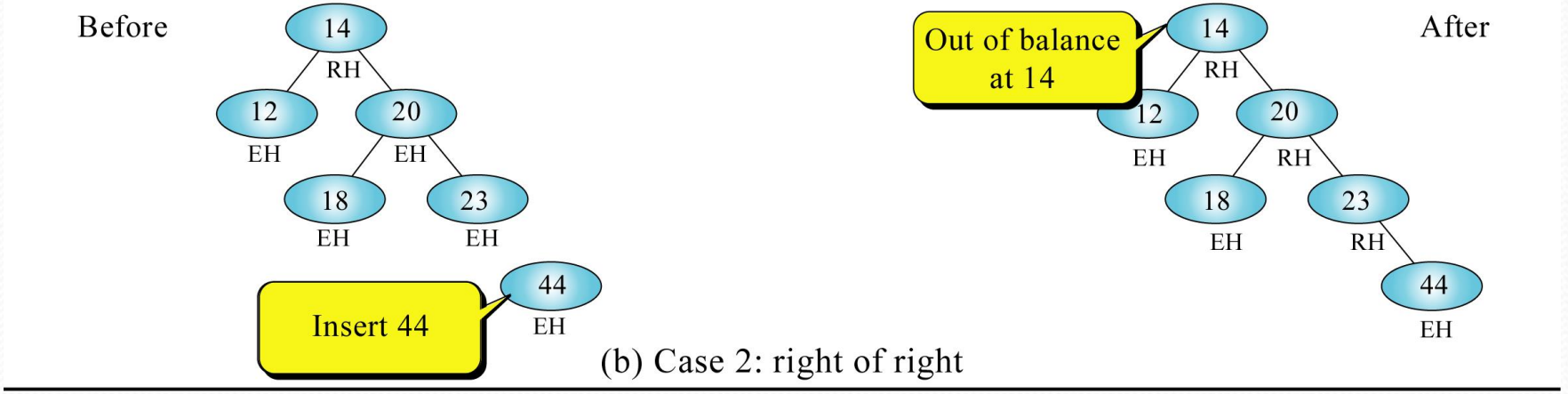
- Mất cân bằng phải-trái (R-L)



Ví dụ: Các trường hợp mất cân bằng

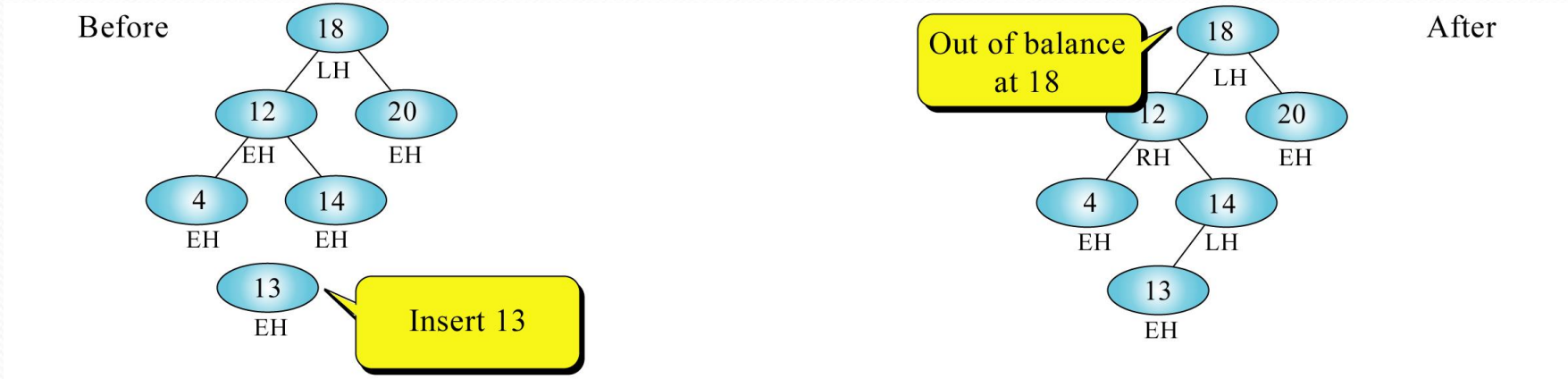


(a) Case 1: left of left

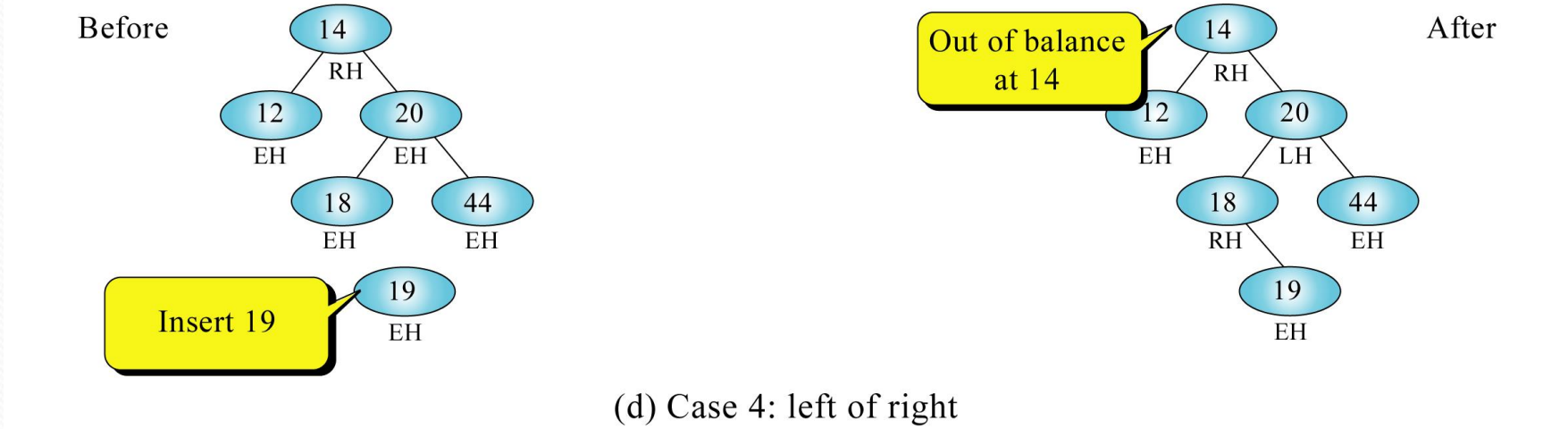


(b) Case 2: right of right

Ví dụ: Các trường hợp mất cân bằng



(c) Case 3: right of left



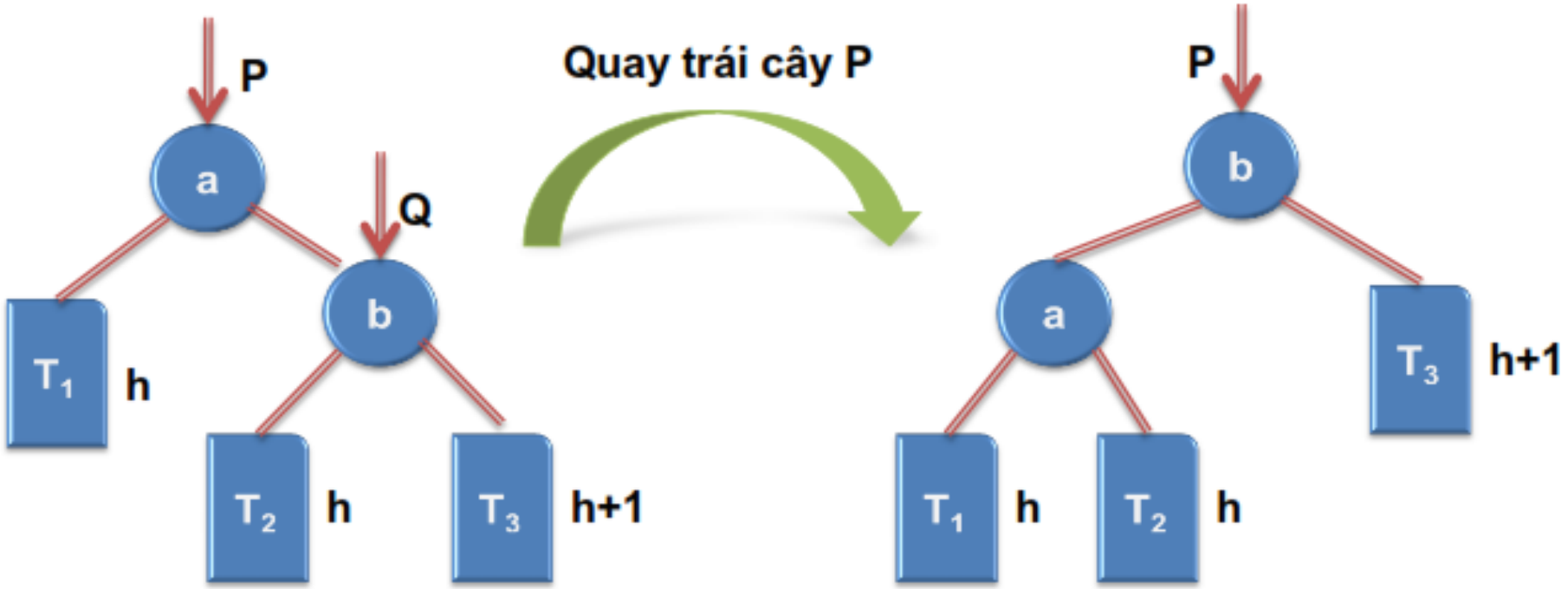
(d) Case 4: left of right

Xử lý mất cân bằng

- Giả sử tại một node cây xảy ra mất cân bằng bên phải (cây con phải chênh lệch với cây con trái hơn một đơn vị):
  - Mất cân bằng phải-phải (RR)
    - Quay trái
  - Mất cân bằng phải-trái (R-L)
    - Quay phải
    - Quay trái

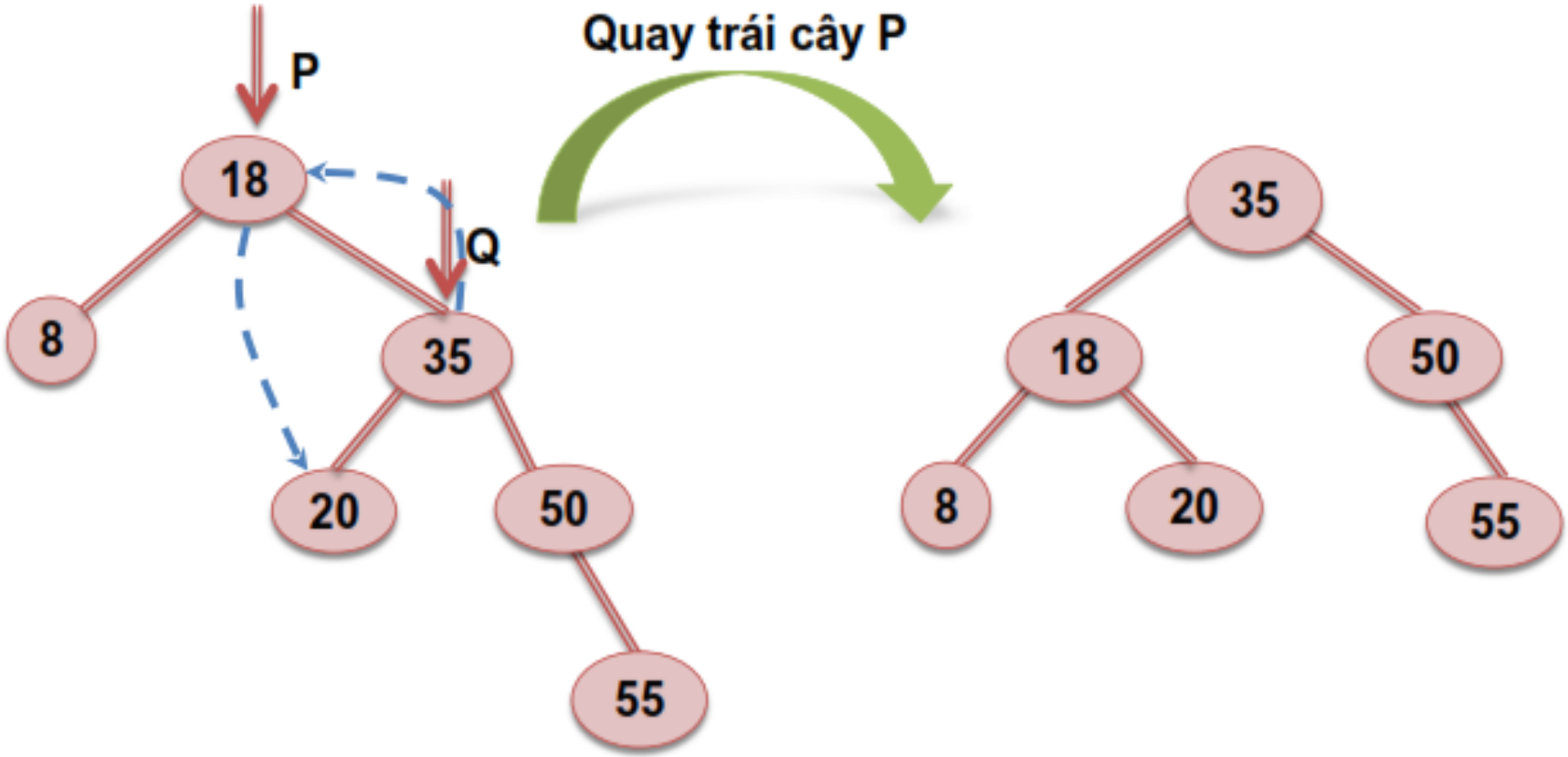
Xử lý mất cân bằng

- P mất cân bằng phải-phải (RR):

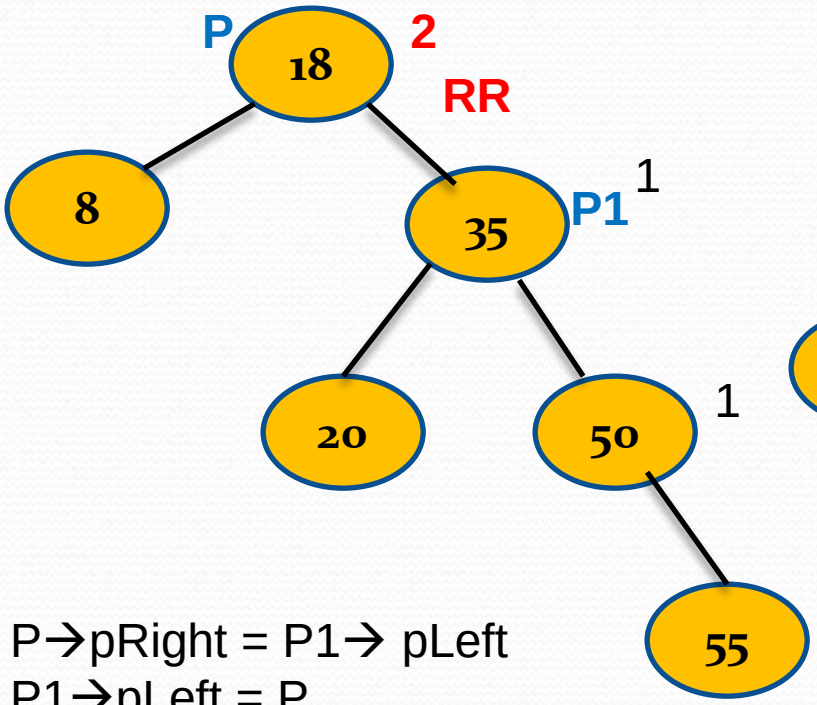


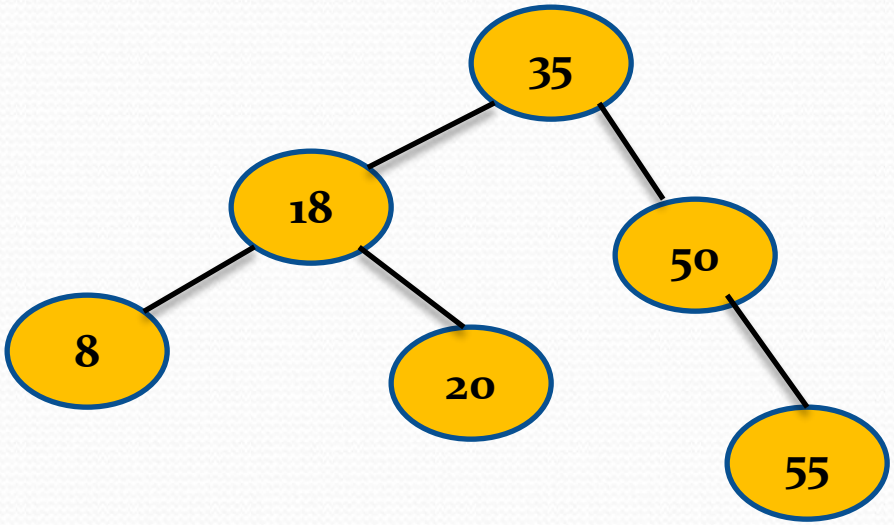
Xử lý mất cân bằng

- P mất cân bằng phải-phải (RR):



Phép quay trái

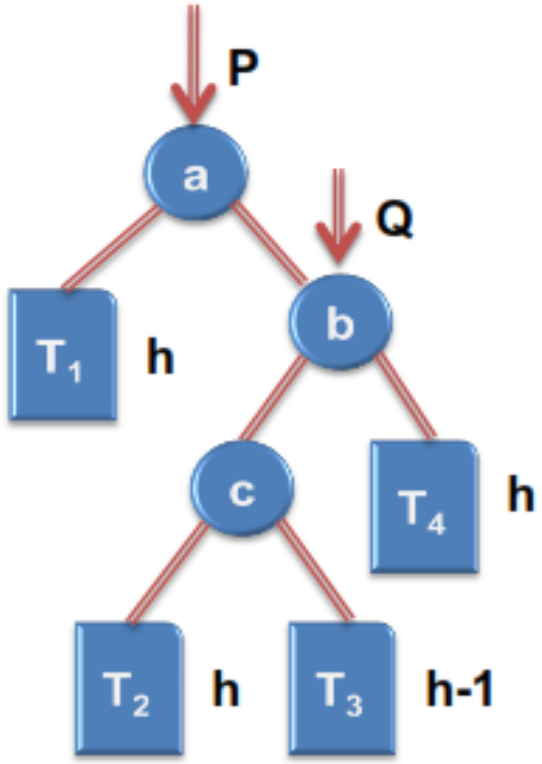




P→pRight = P1→ pLeft
P1→pLeft = P

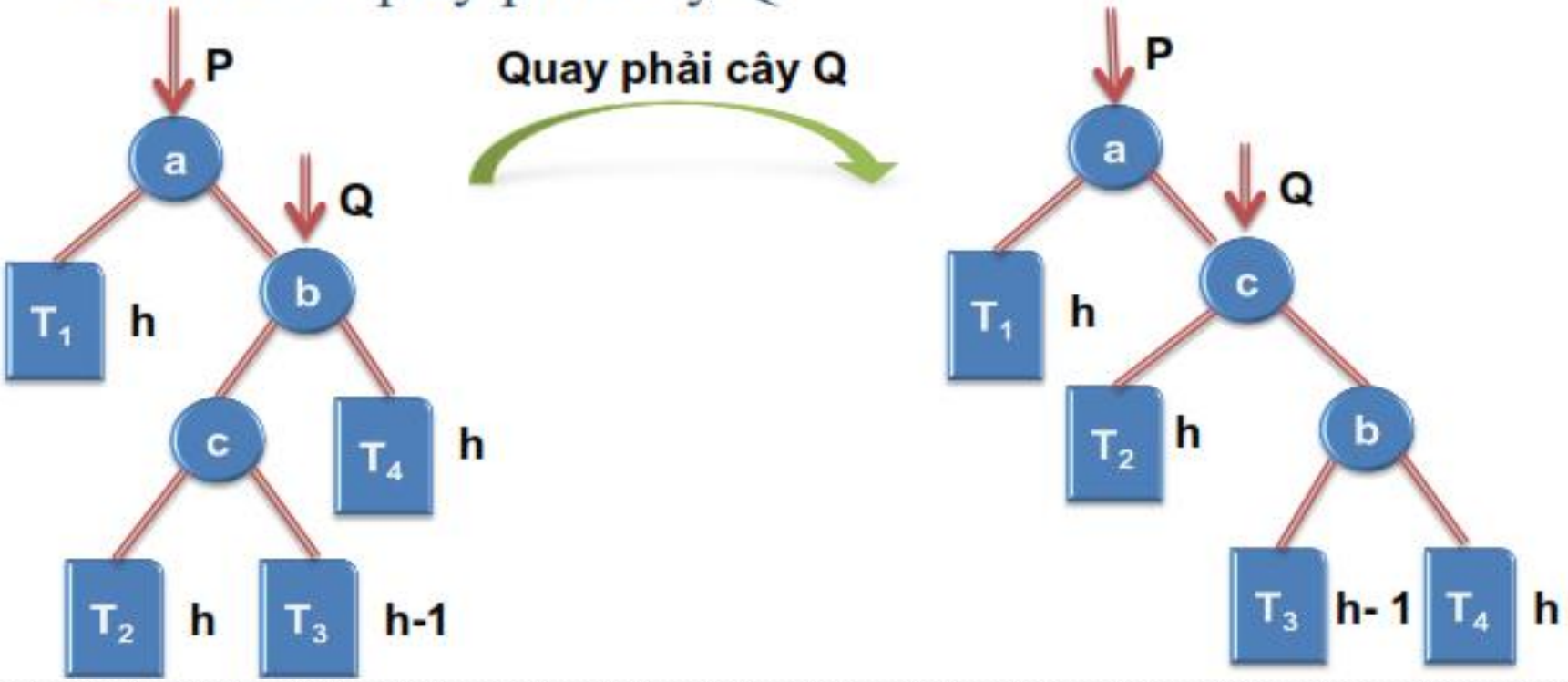
Xử lý mất cân bằng

- P mất cân bằng phải-trái (RL):
  - Bước 1: quay phải Q
  - Bước 2: quay trái cây P



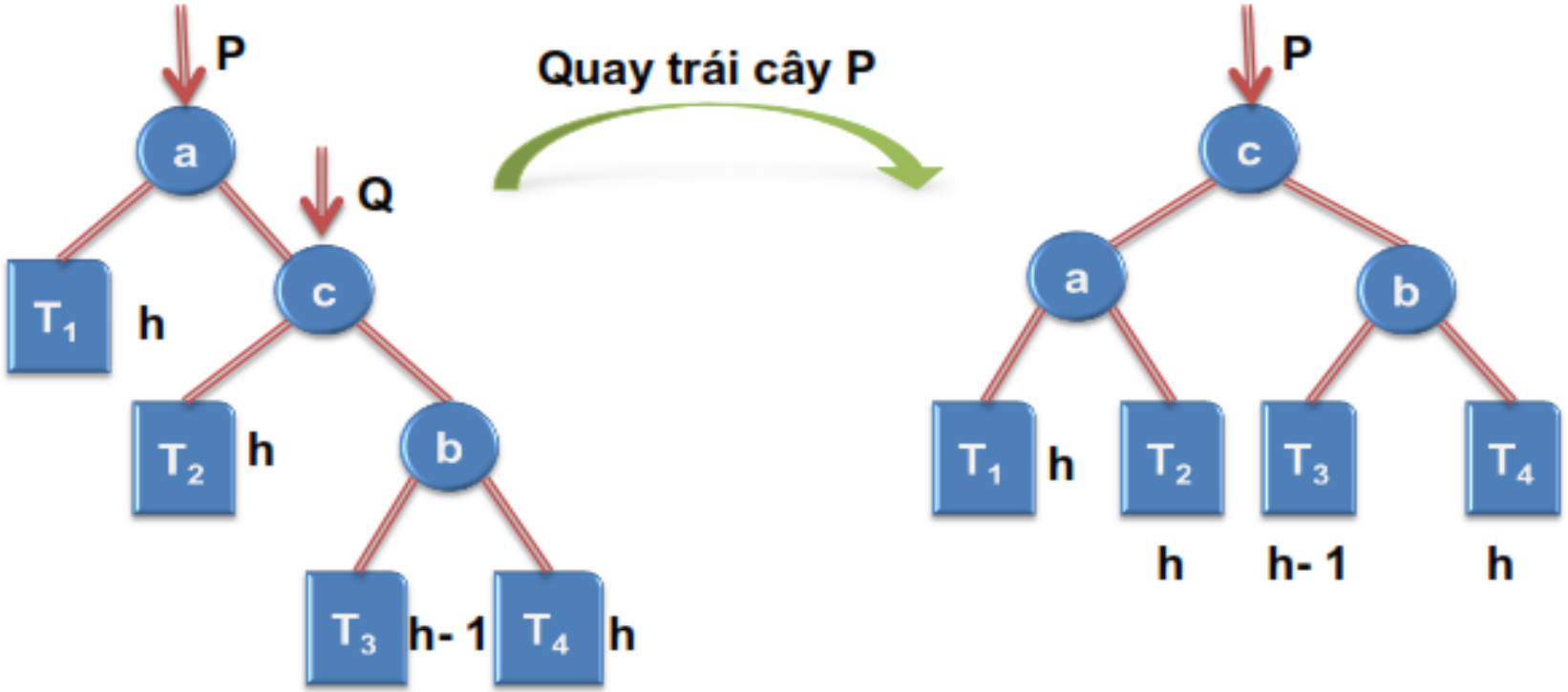
Xử lý mất cân bằng

- P mất cân bằng phải-trái (RL):
  - Bước 1: quay phải cây Q



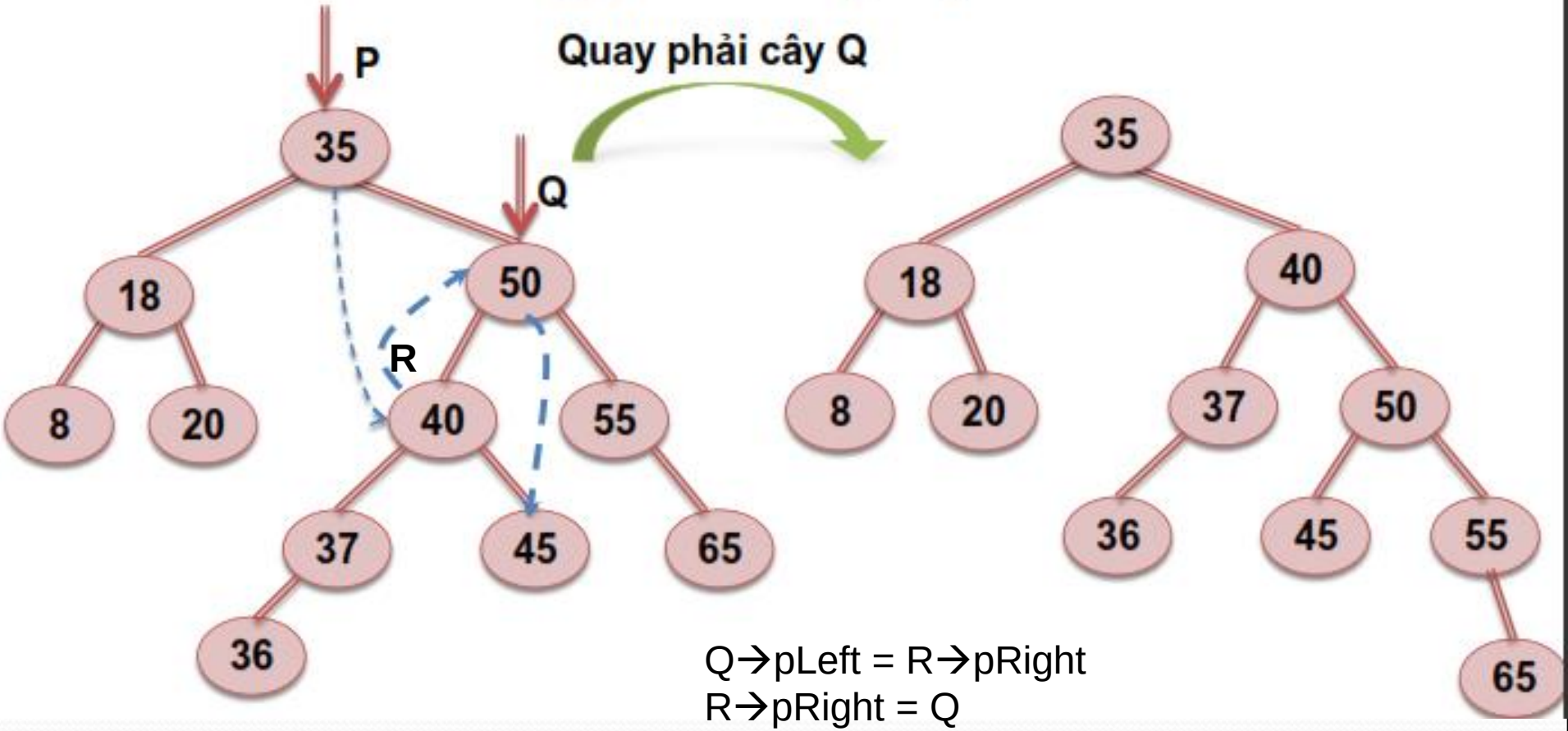
Xử lý mất cân bằng

- P mất cân bằng phải-trái (RL):
  - Bước 2: quay trái cây P



Xử lý mất cân bằng

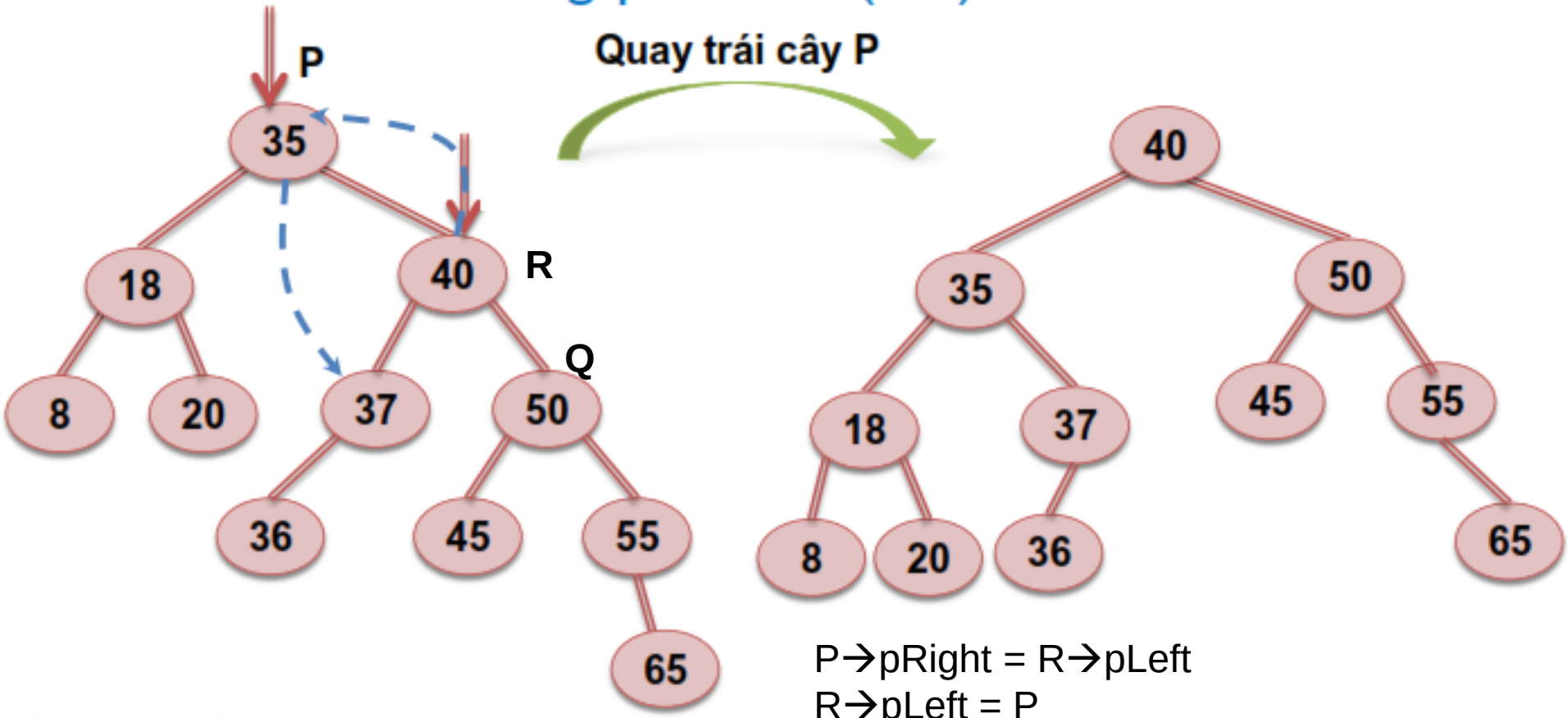
- P mất cân bằng phải-trái (RL) – Bước 1:



Q→pLeft = R→pRight
R→pRight = Q

Xử lý mất cân bằng

- P mất cân bằng phải-trái (RL) - Bước 2:



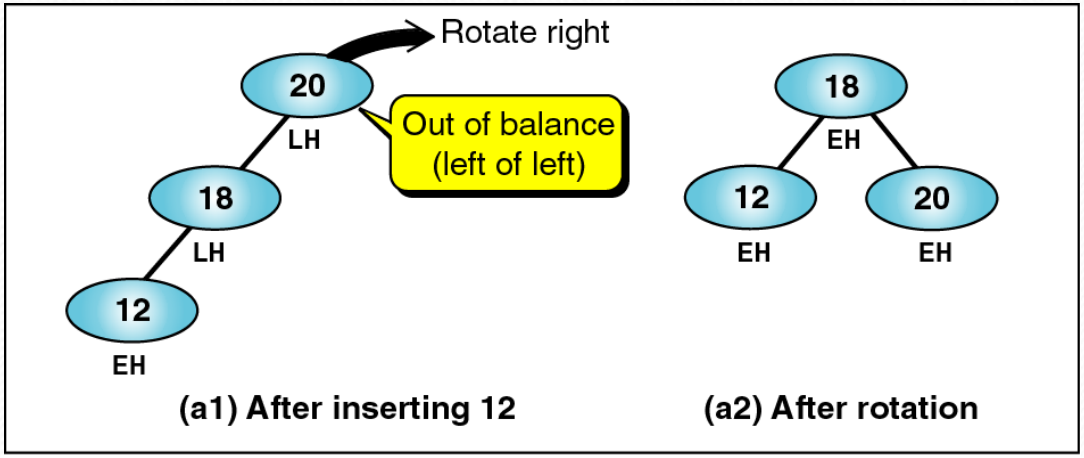
P→pRight = R→pLeft
R→pLeft = P

Xử lý mất cân bằng

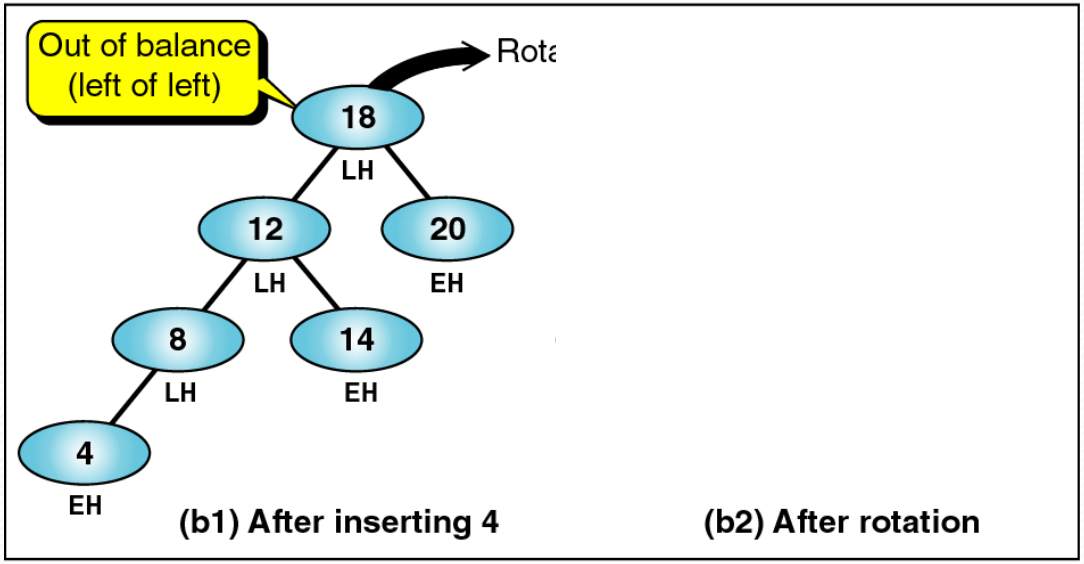
- Khi một node cây xảy ra mất cân bằng bên trái (cây con trái chênh lệch với cây con phải hơn một đơn vị): (thực hiện đối xứng với trường hợp mất cân bằng bên phải)
  - Mất cân bằng trái-trái (LL)
    - Quay phải
  - Mất cân bằng trái-phải (L-R)
    - Quay trái
    - Quay phải

Ví dụ: Tái cân bằng

1. left of left



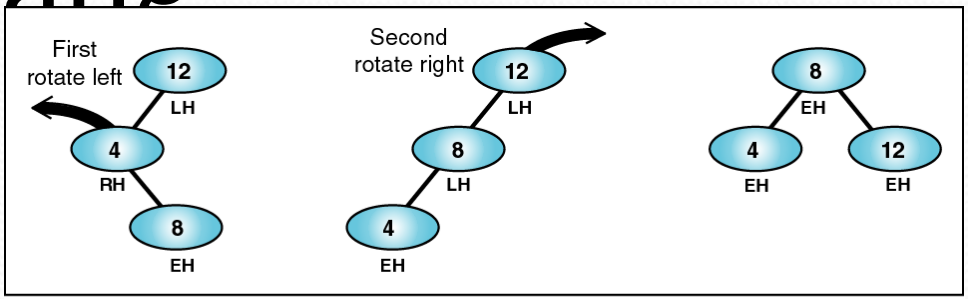
(a) Simple right rotation



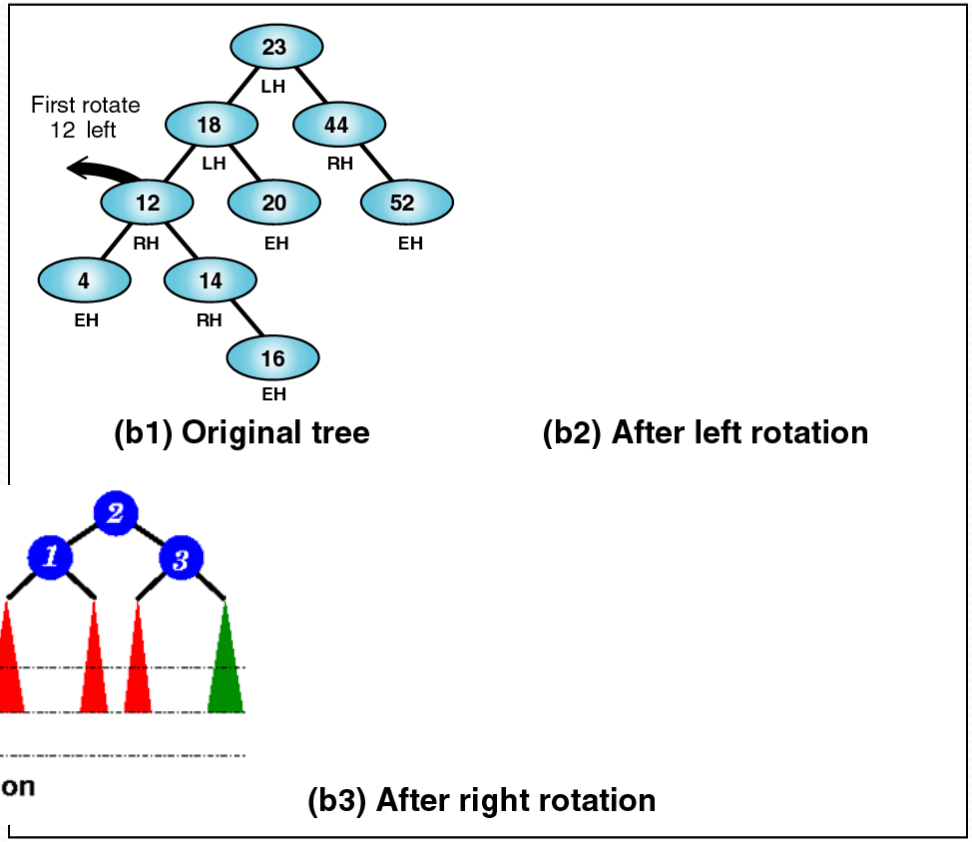
(b) Complex right rotation

Ví dụ: Tái cân bằng

2. Left of right

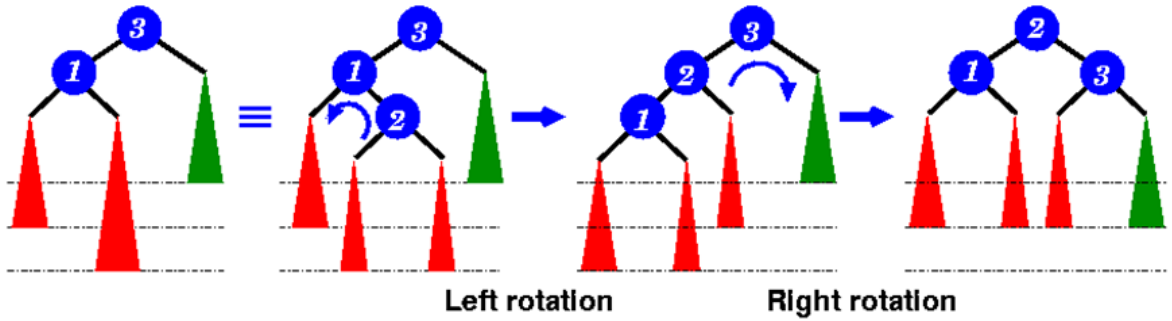


(a) Simple double rotation right



(b1) Original tree

(b2) After left rotation

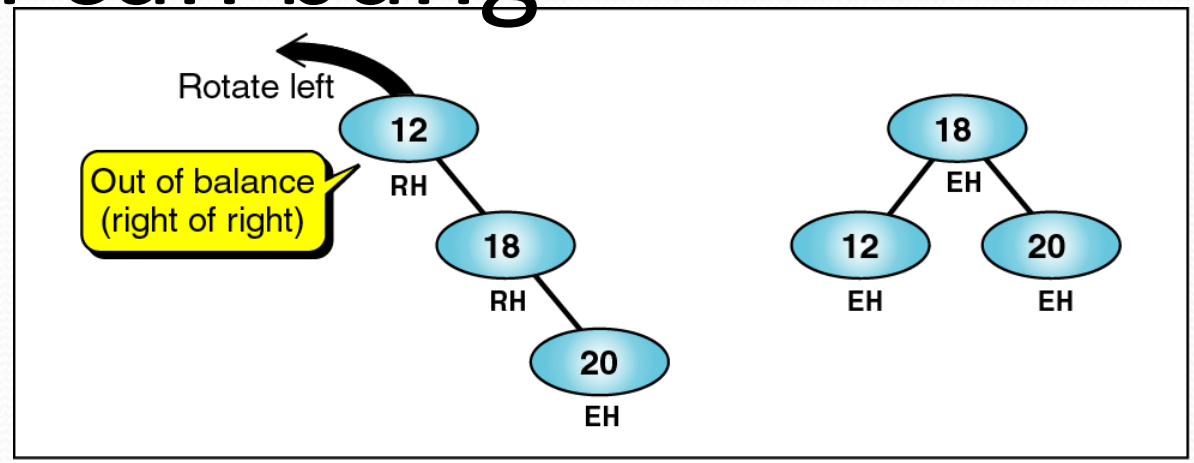


(b3) After right rotation

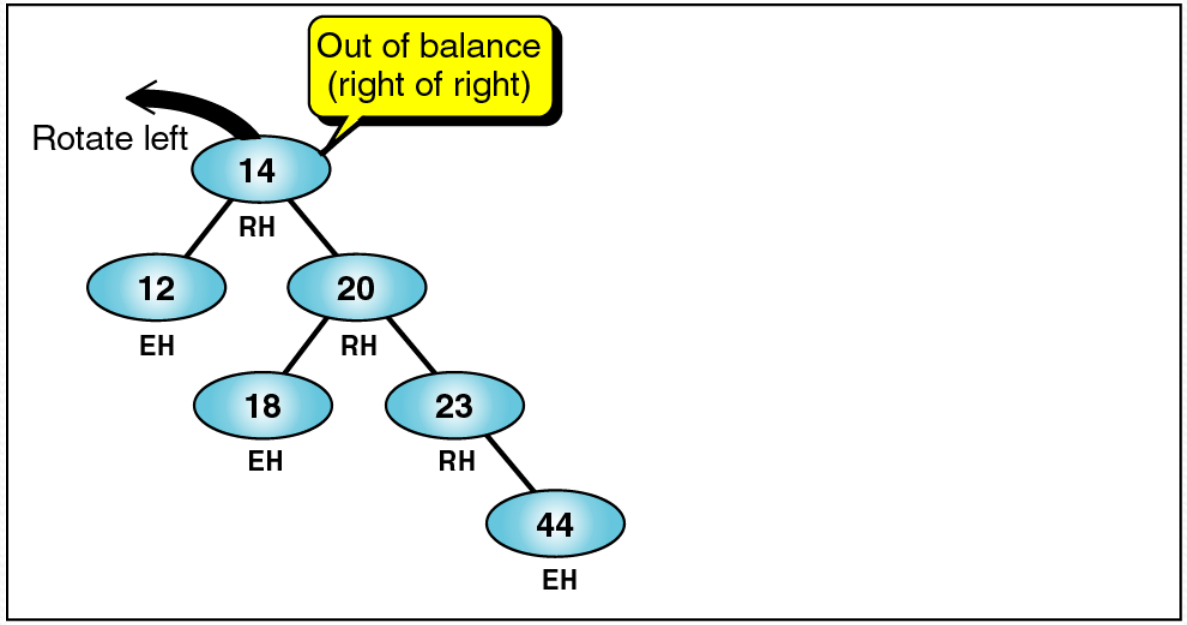
(b) Complex double rotation right

Ví dụ: Tái cân bằng

3. right of right



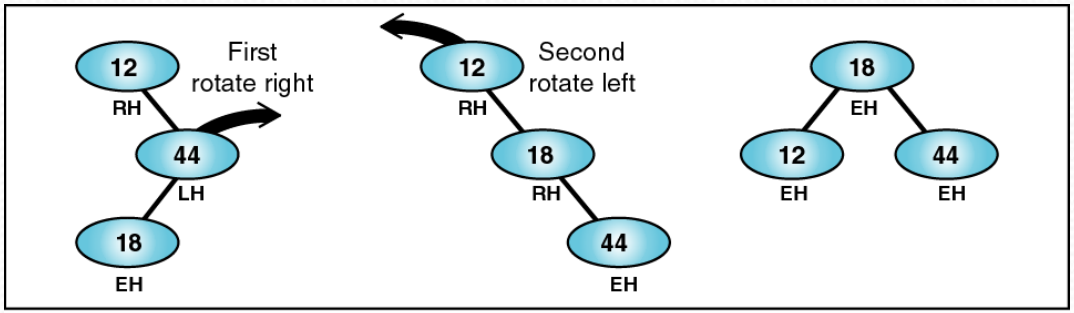
(a) Simple left rotation



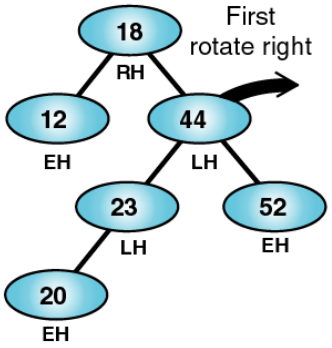
(b) Complex left rotation

Ví dụ: Tái cân bằng

4. left of right



(a) Simple double rotation right



(b1) Original tree

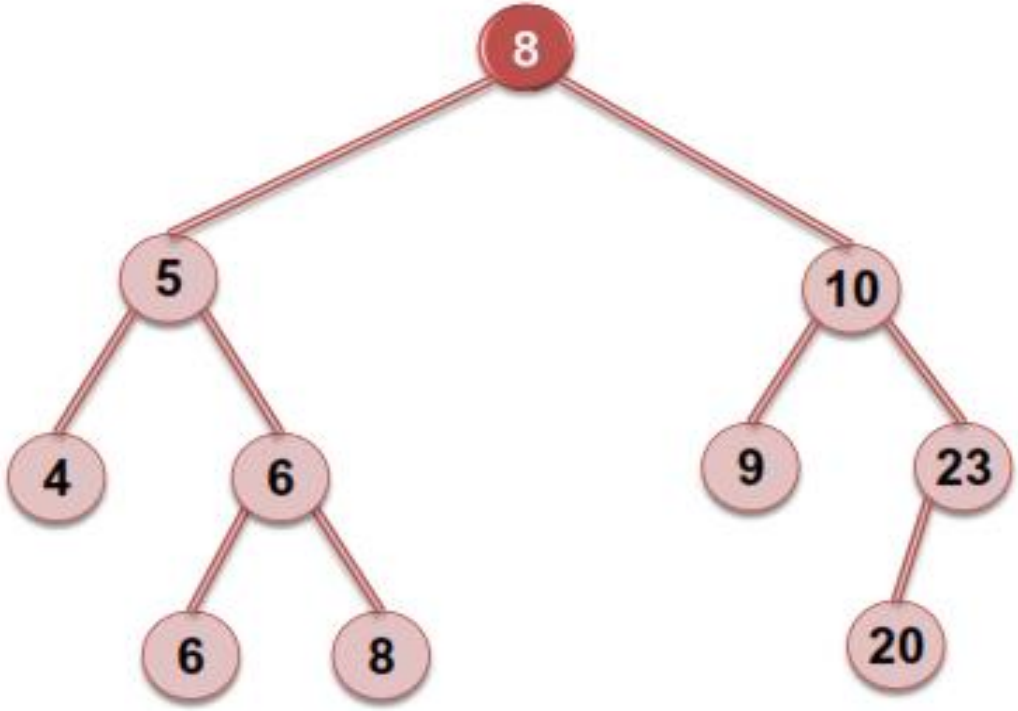
(b2) After right rotation

(b3) After left rotation

(b) Complex double rotation right

Thao tác tìm kiếm

- Thực hiện hoàn toàn tương tự cây nhị phân tìm kiếm.



Thao tác thêm phần tử

- Thực hiện tương tự với việc thêm phần tử của cây nhị phân tìm kiếm.

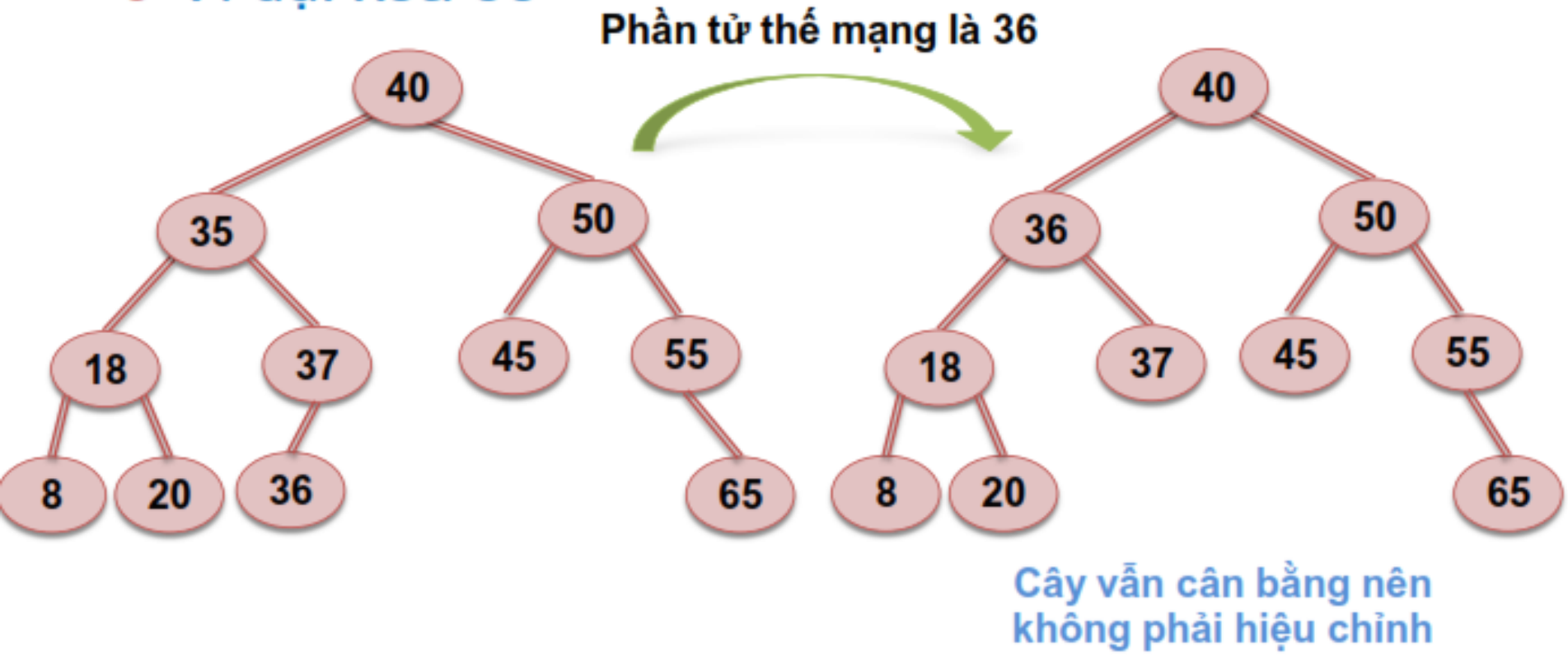
- Nếu xảy ra việc mất cân bằng thì xử lý bằng các trường hợp mất cân bằng đã biết.

Thao tác xóa phần tử

- Thực hiện tương tự cây nhị phân tìm kiếm: xét 3 trường hợp, và tìm phần tử thế mạng nếu cần.
- Sau khi xóa, nếu cây mất cân bằng, thực hiện cân bằng cây.
- Lưu ý: *việc cân bằng sau khi hủy có thể xảy ra dây chuyền*.

Thao tác xóa phần tử

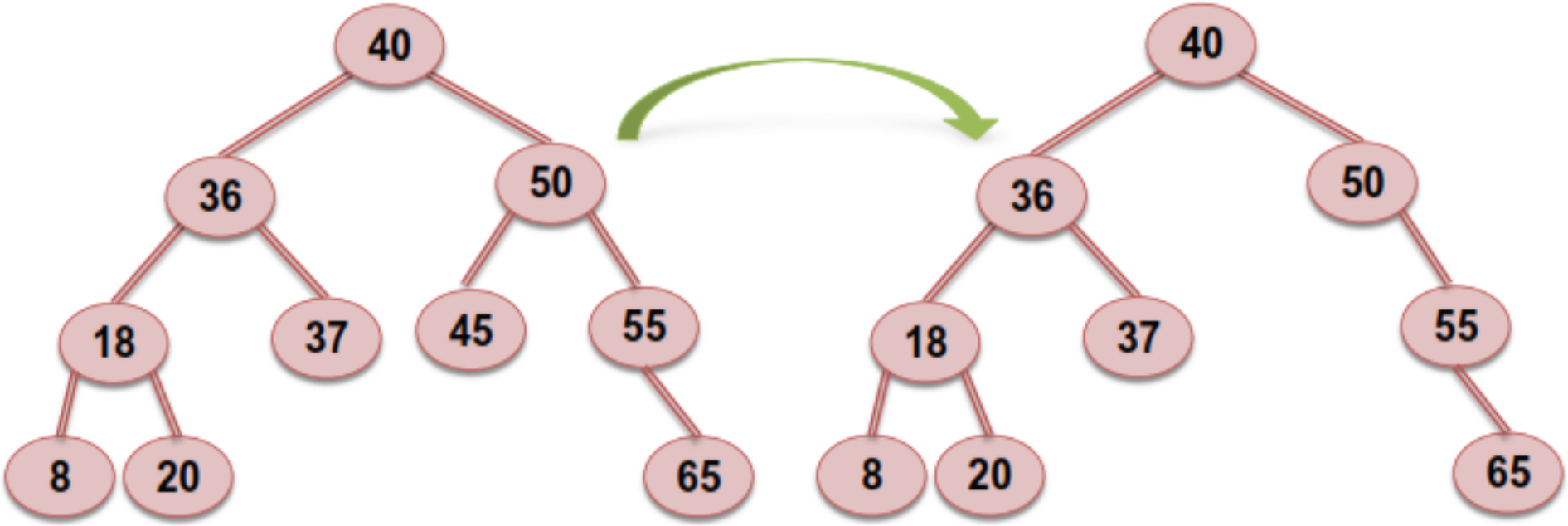
- Ví dụ: xóa 35



Cây vẫn cân bằng nên
không phải hiệu chỉnh

Thao tác xóa phần tử

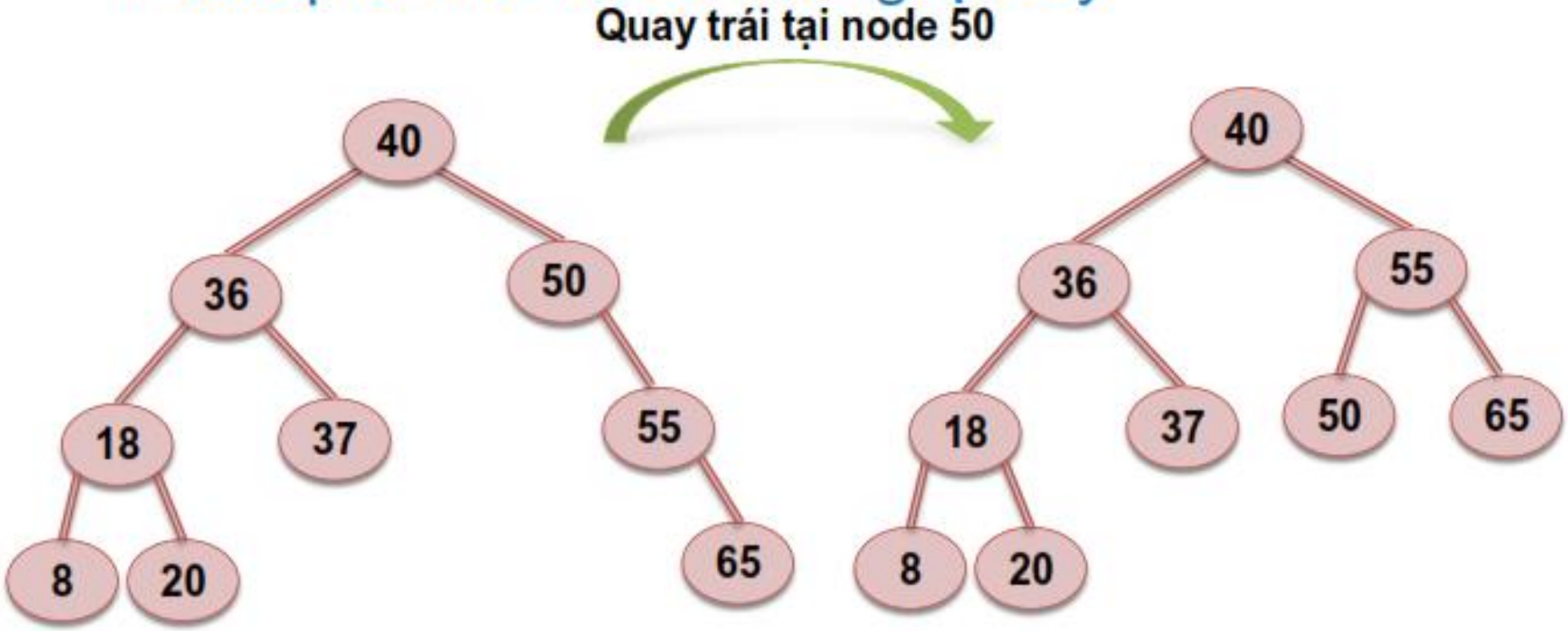
- Xóa phần tử 45



Node 50 bị lệch phải !!!

Thao tác xóa phần tử

- Xóa phần tử 45: cân bằng lại cây



Bài tập

- **Tạo cây AVL với các khóa lần lượt là: 30, 20, 10. Sau đó thêm lần lượt các khóa: 15, 40, 25, 27, 26, 5, 13, 14 vào cây trên.**
-

Bài tập

BT3>Tổ chức một cây cân bằng AVL trong dó mỗi node trên cây chứa thông tin dữ liệu nguyên. Người dùng sẽ nhập các giá trị nguyên từ bàn phím. Với mỗi giá trị nguyên dược nhập vào, phải tạo cây AVL theo dúng tính chất của nó. Nếu người dùng nhập -1 quá trình nhập dữ liệu sẽ kết thúc. Sau dó, xuất thông tin các node trên cây. Khi chuong trình kết thúc, tất cả các node trên cây bị ***xóa bỏ khỏi bộ nhớ***