



CANTHO UNIVERSITY

Cấu trúc dữ liệu CÂY AVL (G.M. Adelson-Velsky và E.M. Landis)

Bộ môn Công Nghệ Phần Mềm



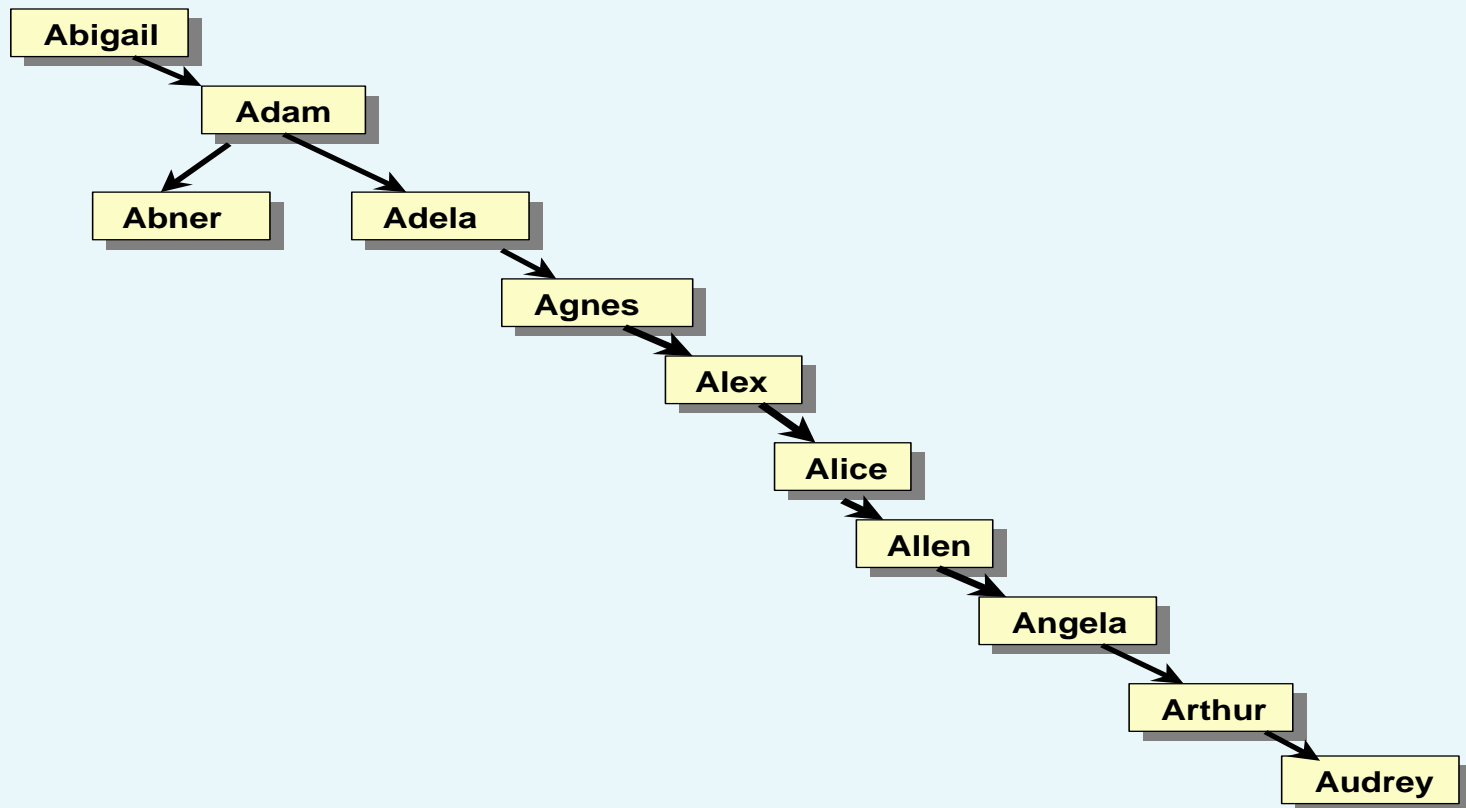
Nội dung

- Khái niệm về cây tìm kiếm nhị phân cân bằng
- Khái niệm về cây AVL
- Các thuật toán trên cây AVL
- Ý tưởng cài đặt cây bằng con trỏ



KIẾN THỨC BỔ SUNG

- Bên dưới là một cây BST “không cân bằng”





KIẾN THỨC BỔ SUNG

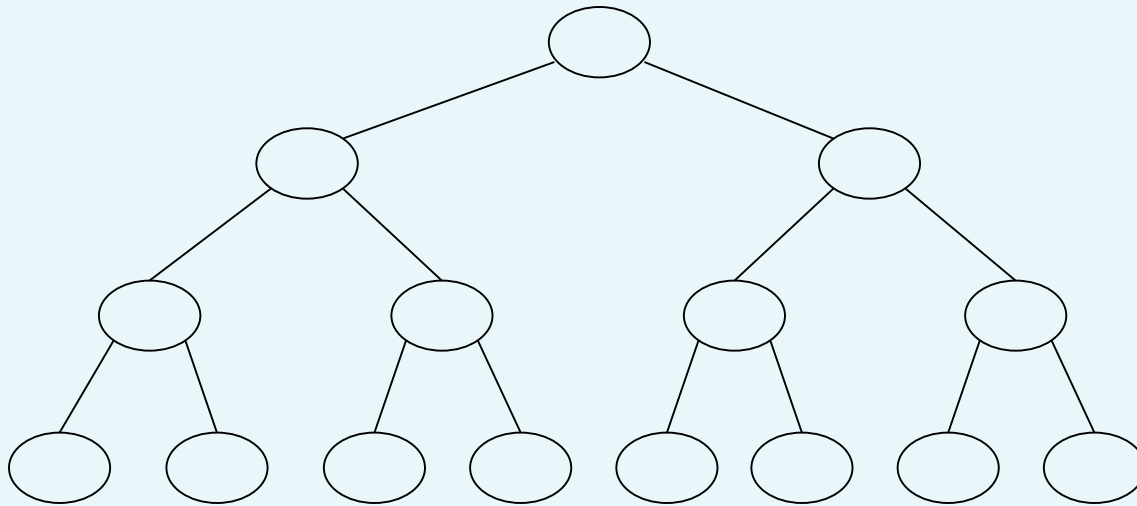


- Tìm kiếm nút có khóa “Audrey” là trường hợp xấu nhất
 - Ta phải duyệt qua hầu như toàn bộ n nút của cây.
 - Thời gian tìm kiếm: $O(n)$.



CÂY NHỊ PHÂN ĐẦY ĐỦ (full binary tree)

- Một cây nhị phân là “**cây nhị phân đầy đủ**” nếu và chỉ nếu
 - Mỗi nút không phải lá có chính xác 2 nút con.
 - Tất cả các nút lá có mức bằng nhau.

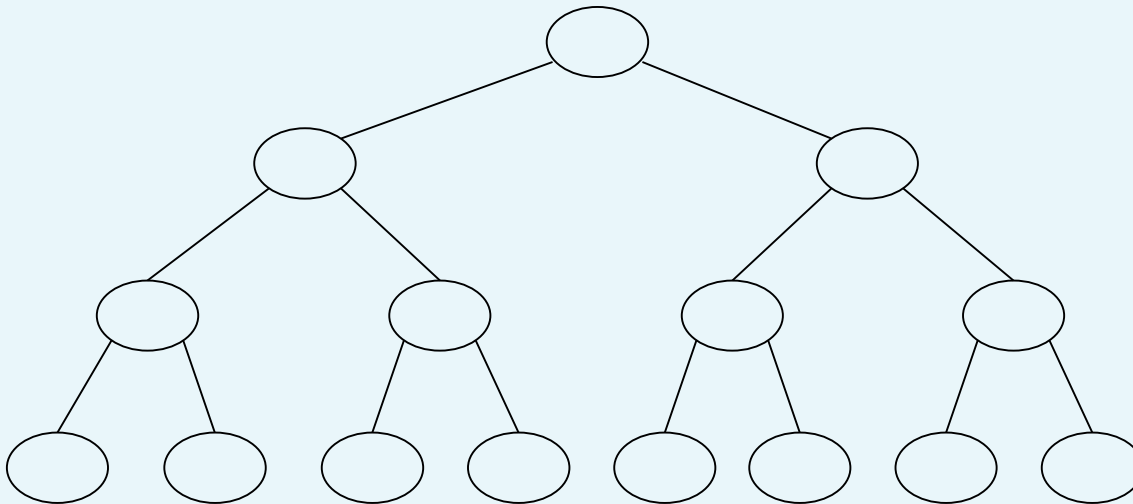


Ví dụ: một cây nhị phân đầy đủ



CÂY NHỊ PHÂN ĐẦY ĐỦ

- **Câu hỏi về cây nhị phân đầy đủ:**
 - Một cây nhị phân đầy đủ chiều cao h sẽ có bao nhiêu nút lá? 2^h
 - Một cây nhị phân đầy đủ chiều cao h sẽ có tất cả bao nhiêu nút? $2^{(h+1)} - 1$



Ví dụ: một cây
nhị phân đầy đủ



CÂY NHỊ PHÂN HOÀN CHỈNH (complete binary tree)

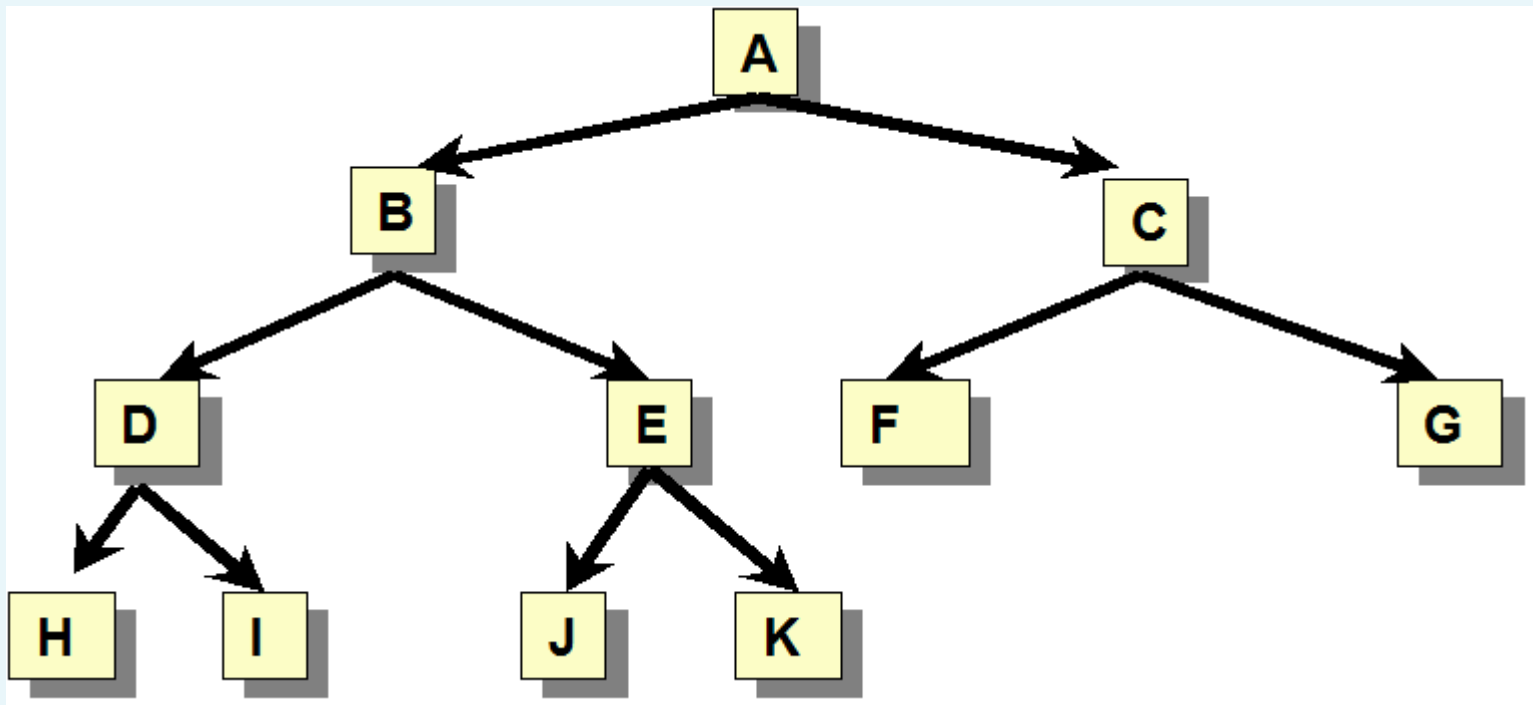
- Một **cây nhị phân hoàn chỉnh** (về chiều cao) thỏa mãn các điều kiện sau:
 - Mức 0 đến $h-1$ là trình bày một cây nhị phân đầy đủ chiều cao $h-1$.
 - Một hoặc nhiều nút ở mức $h-1$ có thể có 0, hoặc 1 nút con.
 - Nếu j, k là các nút ở mức $h-1$, khi đó j có nhiều nút con hơn k nếu và chỉ nếu j ở bên trái của k .



CANTHO UNIVERSITY

CÂY NHỊ PHÂN HOÀN CHỈNH

- Ví dụ



Một cây nhị phân hoàn chỉnh



CÂY NHỊ PHÂN HOÀN CHỈNH

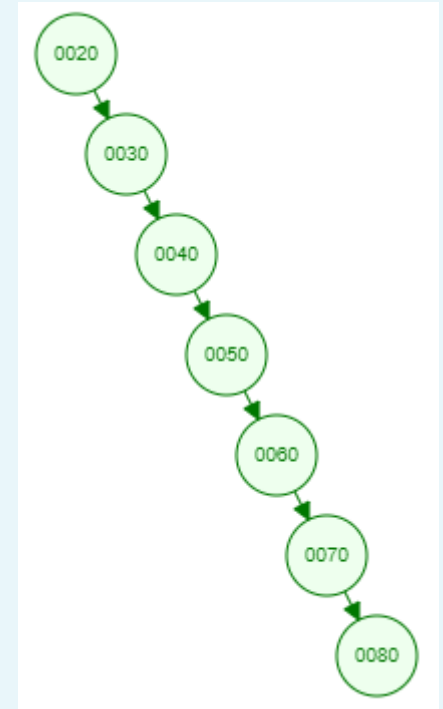
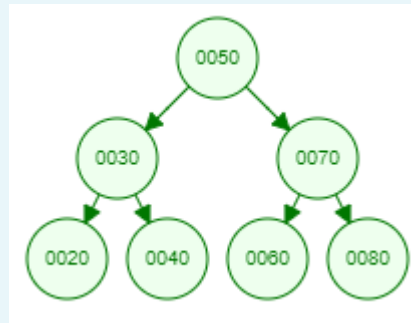
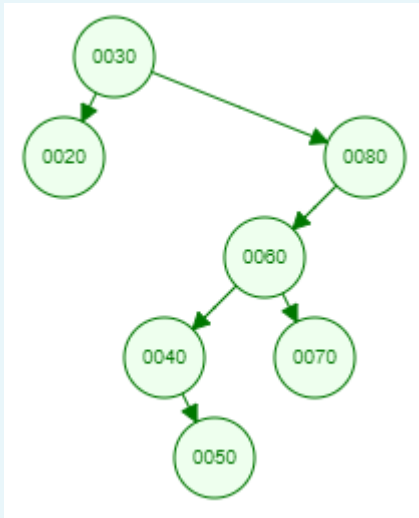
- Được cho một tập hợp n nút, một cây nhị phân hoàn chỉnh của những nút này cung cấp số nút lá nhiều nhất - với chiều cao trung bình của mỗi nút là nhỏ nhất.
- Cây hoàn chỉnh n nút phải chứa ít nhất một nút có chiều cao là $\lfloor \log n \rfloor$.



CANTHO UNIVERSITY

Khái niệm về cây BST cân bằng

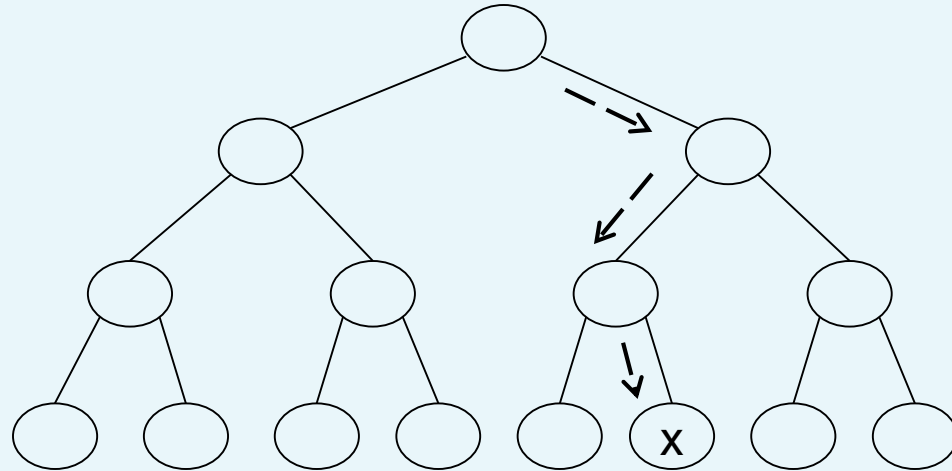
- Cây BST được dựng từ các khóa:
20, 30, 40, 50, 60, 70, 80



- Nhận xét: độ phức tạp trong các giải thuật trên cây tìm kiếm nhị phân trường hợp xấu nhất là $O(n)$ và trung bình $O(\log n)$.



Khái niệm về cây BST cân bằng



- Cây có n nút.
- Thời gian tìm x trong trường hợp xấu nhất khi x là lá
 - Ta phải đi qua đường đi độ dài h để tìm x (h là chiều cao cây)
 - $2^{(h+1)} - 1 = n$
 $\Rightarrow h = \log(n+1) - 1$
 $\Rightarrow O(\log(n+1) - 1) \approx O(\log(n))$



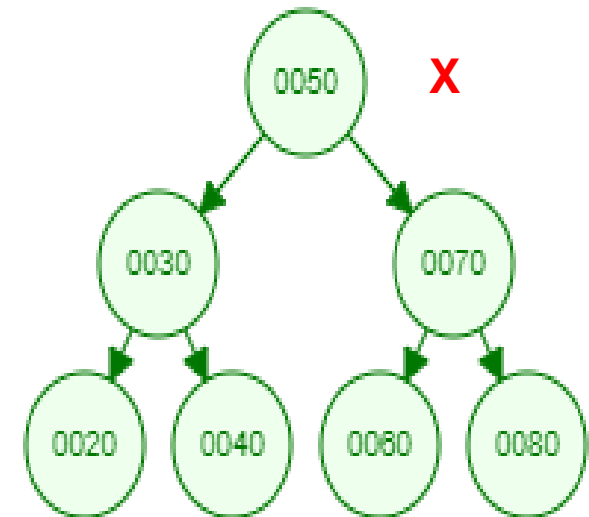
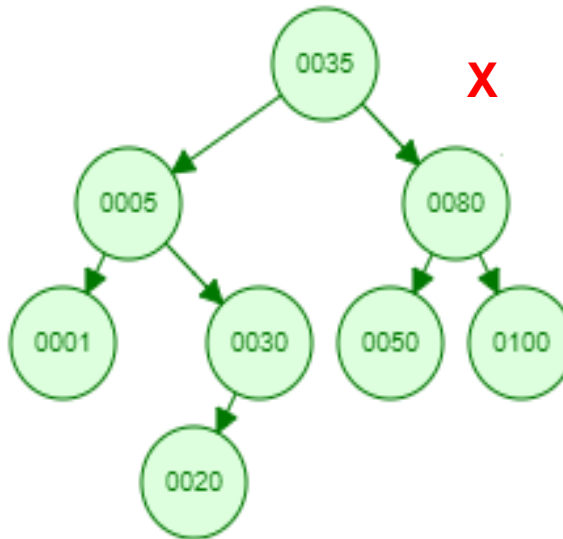
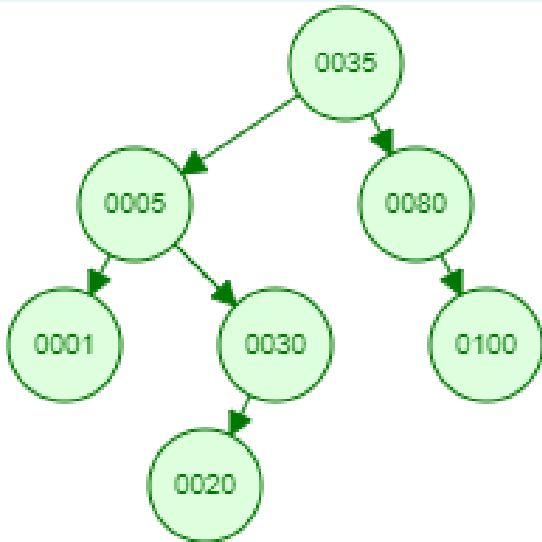
Khái niệm về cây BST cân bằng

- Quá trình tìm kiếm khóa, thêm nút, xóa nút trên cây BST là *quá trình di chuyển từ nút gốc ra nút lá*. → **Cây càng cao thì giải thuật càng kém hiệu quả.**
- Do vậy rất cần thiết phải xây dựng cây BST mà trong đó chiều cao của cây càng nhỏ càng tốt để cho các giải thuật được hiệu quả nhất. Cây ở dạng này được gọi là **cây cân bằng**.



Cây BST cân bằng hoàn toàn

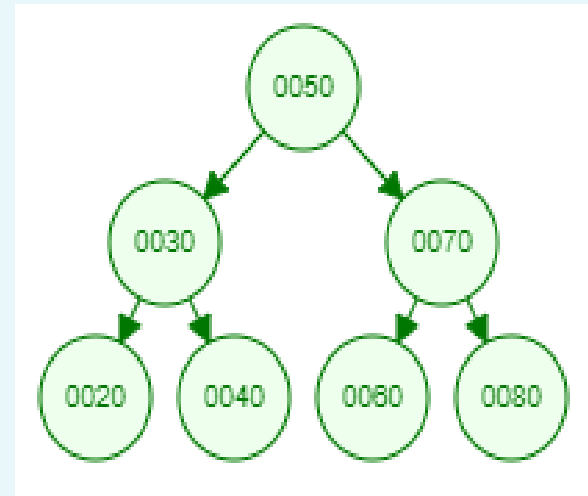
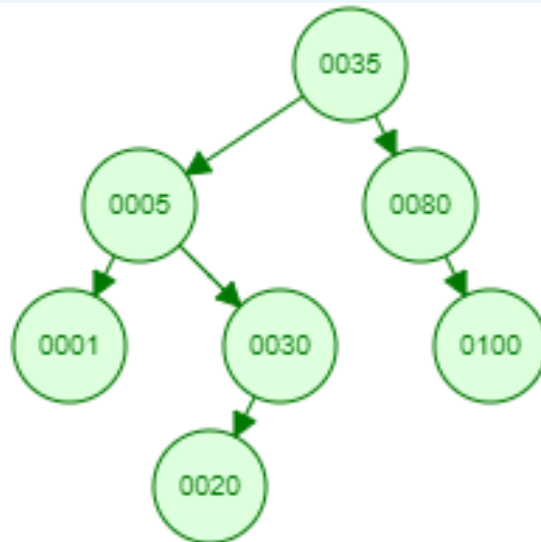
- **Cây BST cân bằng hoàn toàn** là cây BST mà tại mỗi nút có tổng số nút của cây con trái và con phải lệch nhau không quá một.
- Cây nào là cây cân bằng hoàn toàn?





Cây BST cân bằng tương đối (cân bằng về chiều cao)

- Cây BST cân bằng về chiều cao là cây BST mà trong đó mỗi nút đều có chiều cao con trái và con phải lệch nhau tối đa là 1.
- Cây cân bằng về chiều cao còn gọi là **cân bằng tương đối**.
- Ví dụ:





Khái niệm về cây AVL

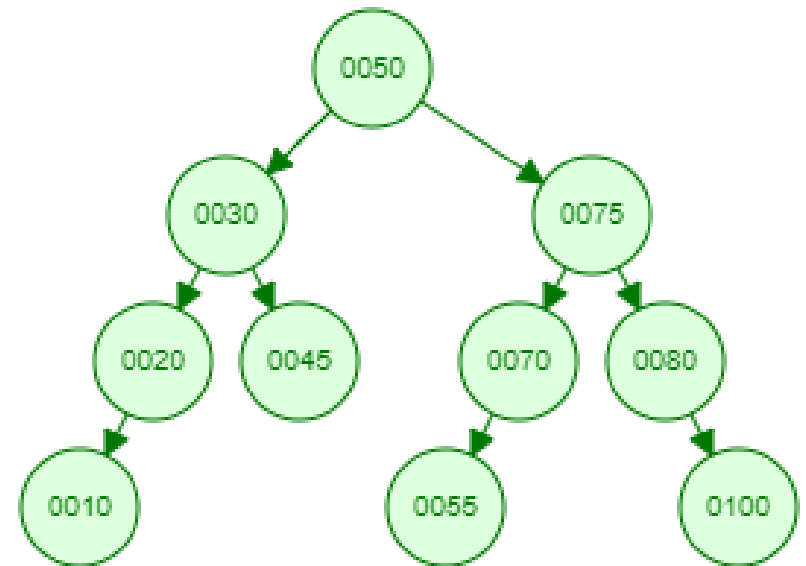
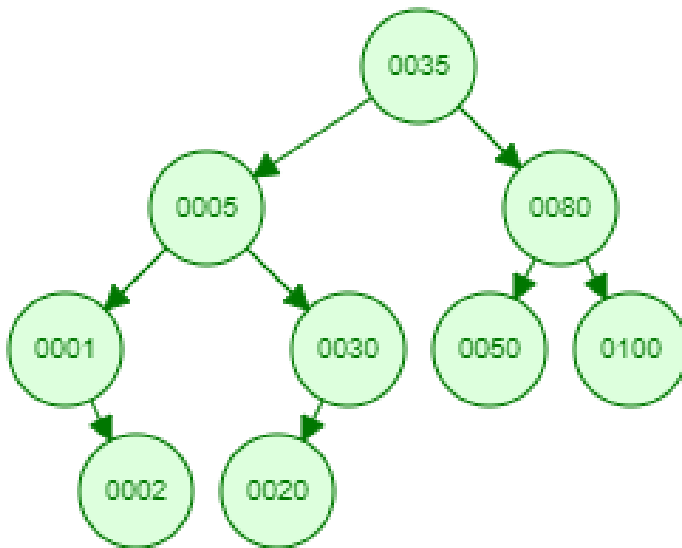
- **Cây AVL** được gọi theo tên của hai người đề xuất chúng, G.M. Adelson-Velsky và E.M. Landis, được công bố trong bài báo của họ vào năm 1962: "An algorithm for the organization of information." (Một thuật toán về tổ chức thông tin)
- **Cây AVL** là cây **BST** mà chiều cao của hai cây con của mọi nút chênh lệch tối đa là 1.



CANTHO UNIVERSITY

Khái niệm về cây AVL

- Ví dụ:





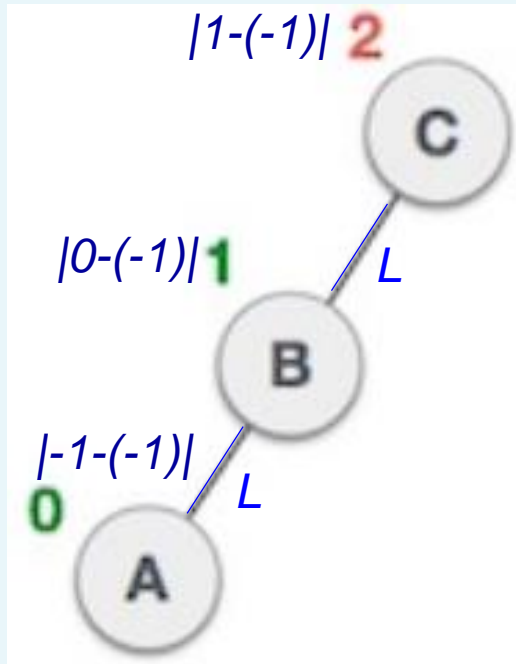
Các thao tác trên cây AVL

- Cây AVL là cây tìm kiếm nhị phân nên *các thao tác cơ bản trên cây AVL tương tự như các thao tác cơ bản trên cây BST.*
- Trong đó, thao tác thêm và xóa nút trên cây có thể làm cây mất cân bằng nên *phải có thao tác cân bằng lại cây* sau khi thực hiện thêm hoặc xóa nút.

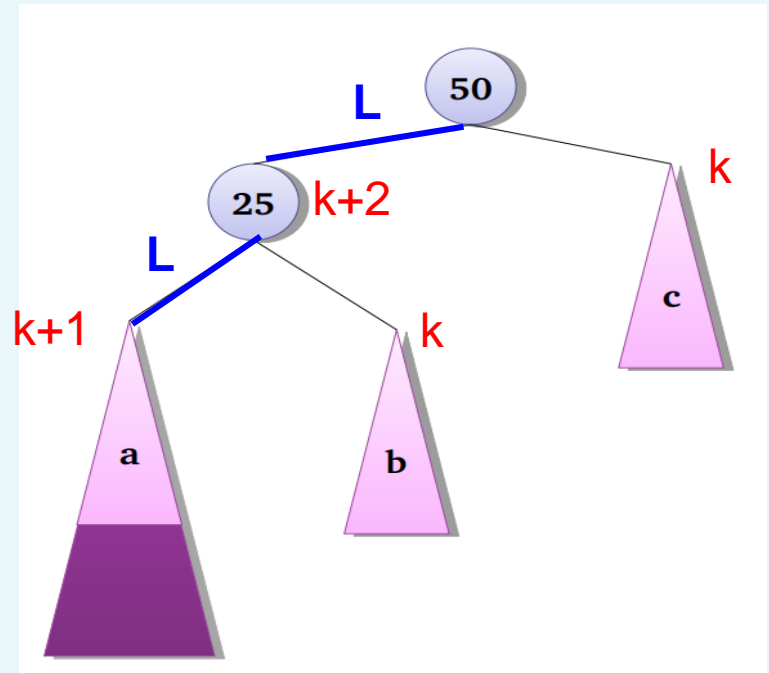


Các trường hợp cây mất cân bằng

- Trường hợp 1: Cây mất cân bằng **bên trái của con trái (L-L)**



BalanceFactor = $|\text{height}(\text{left-sutree}) - \text{height}(\text{right-sutree})|$

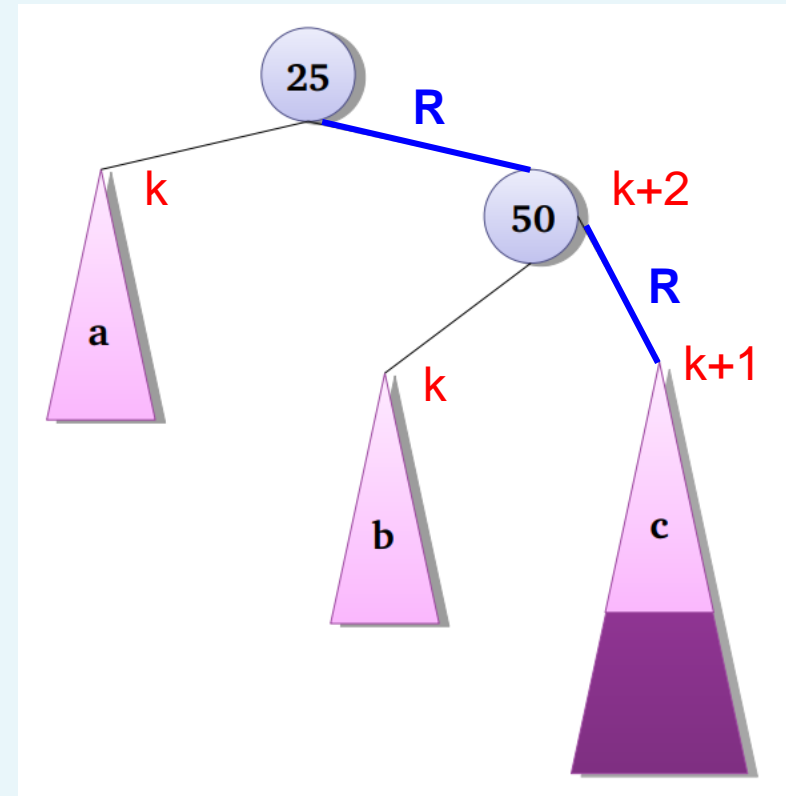
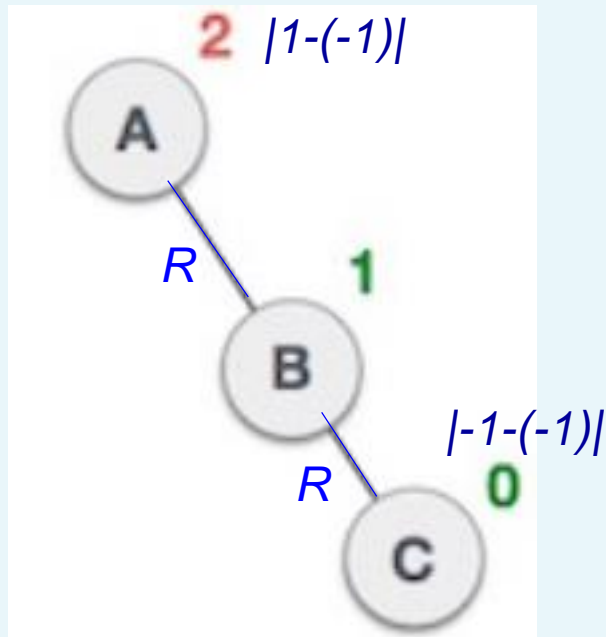


$k, k+1, k+2$: chiều cao các cây con



Các trường hợp cây mất cân bằng

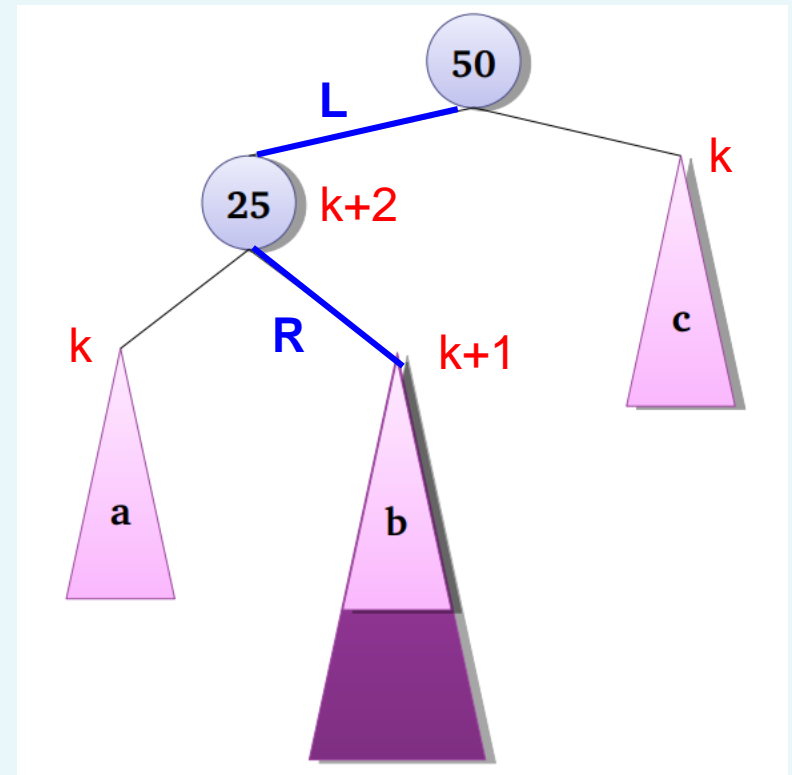
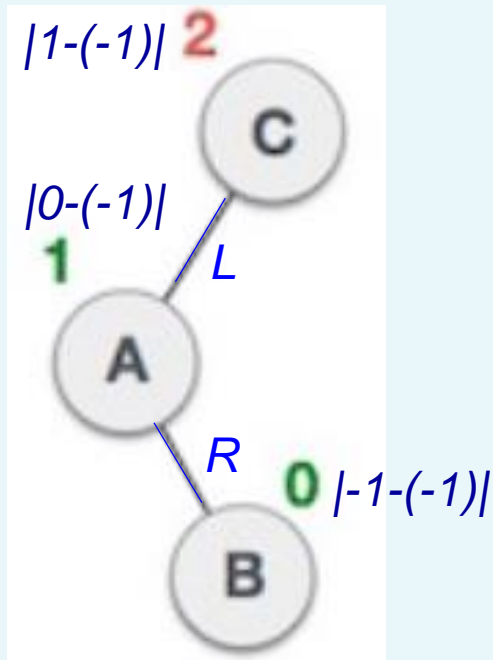
- Trường hợp 2: Cây mất cân bằng **bên phải của con phải (R-R)**





Các trường hợp cây mất cân bằng

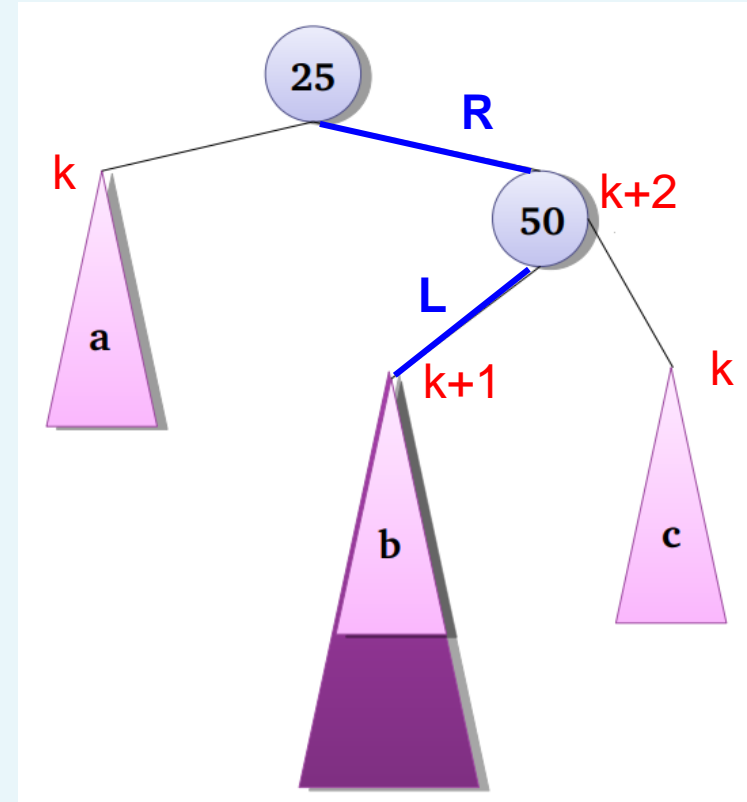
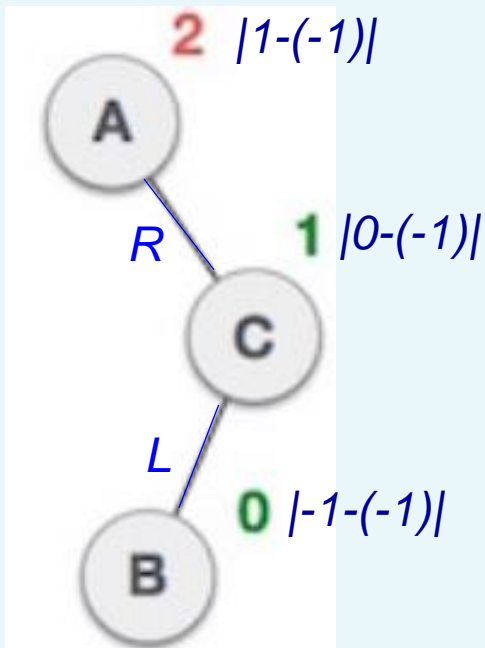
- Trường hợp 3: Cây mất cân bằng **bên phải của con trái (R-L)**





Các trường hợp cây mất cân bằng

- Trường hợp 4: Cây mất cân bằng **bên trái của con phải (L-R)**

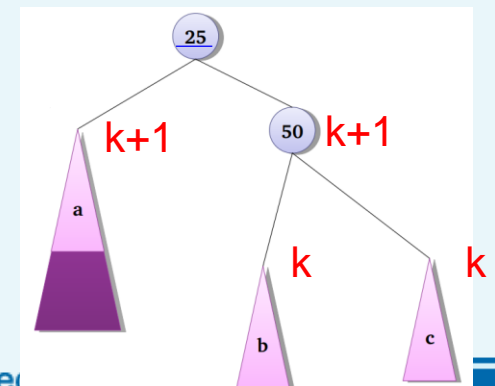
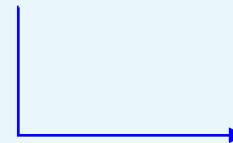
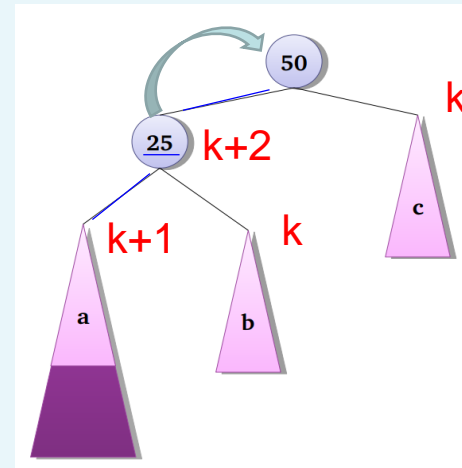
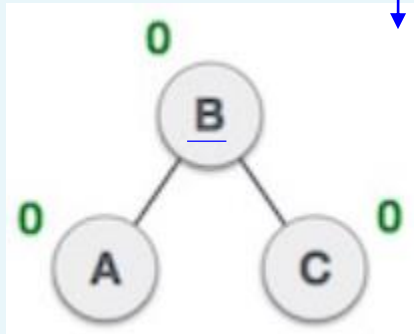
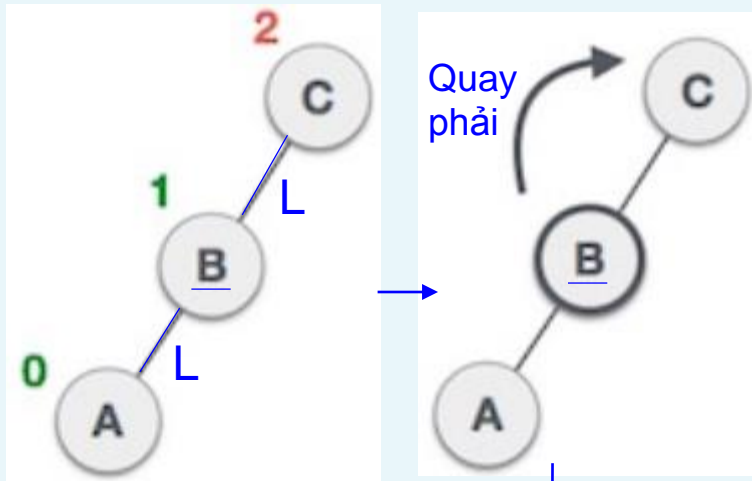




CANTHO UNIVERSITY

Các quy tắc xử lý khi cây mất cân bằng

- Đối với trường hợp 1 (L-L) ta thực hiện **quay đơn qua phải** (right rotate) như sau:

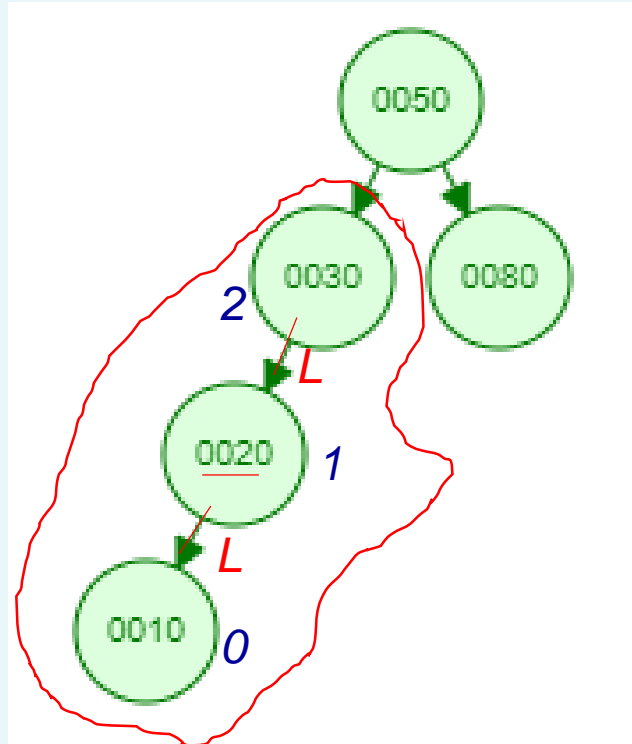
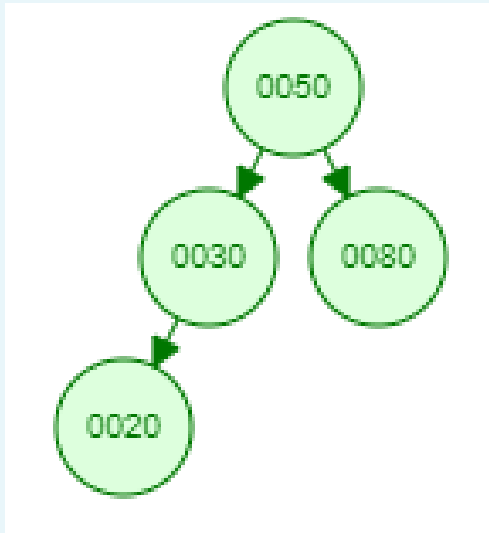




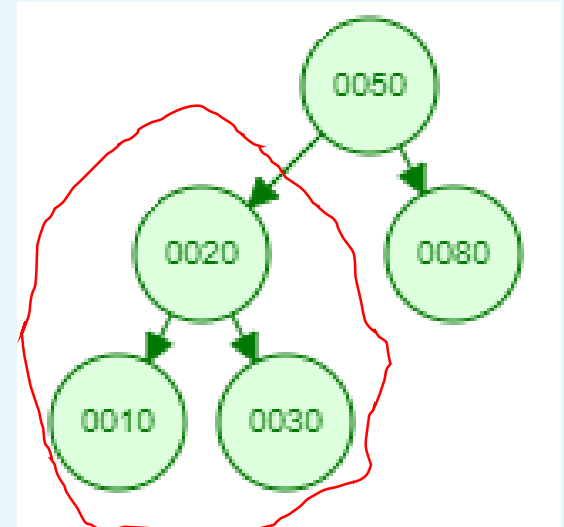
Ví dụ

- Thêm nút 10

Mất cân bằng bên trái
của con trái



Quay qua phải

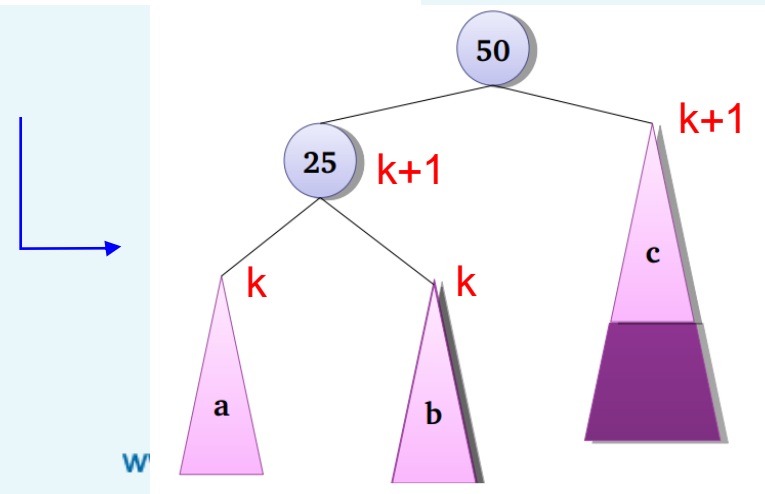
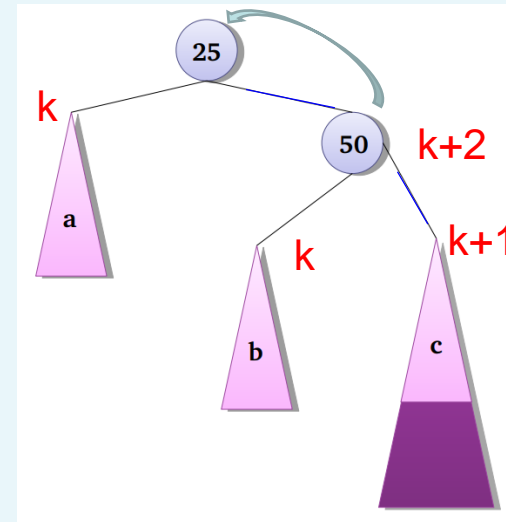
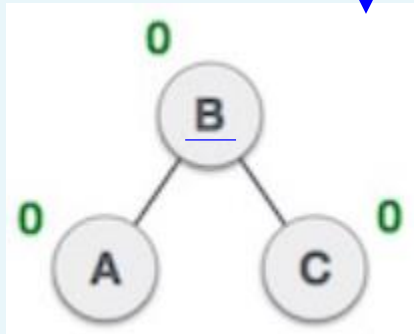
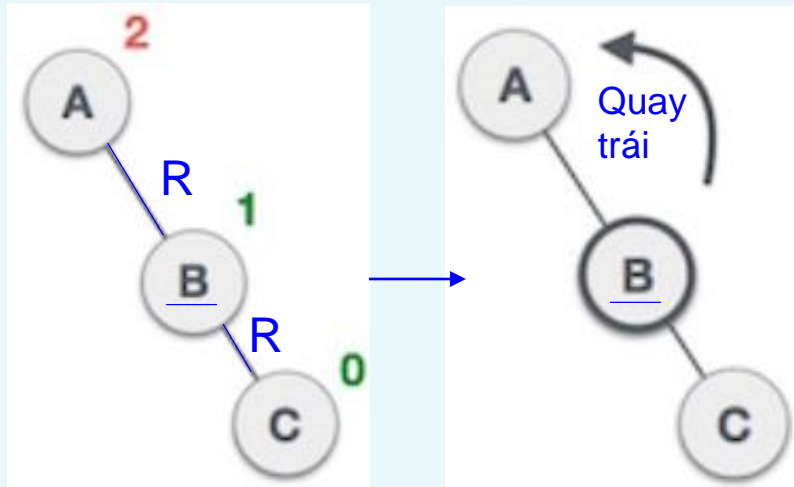




CANTHO UNIVERSITY

Các quy tắc xử lý khi cây mất cân bằng

- Trường hợp 2: Ta thực hiện quay đơn qua trái (Left Rotate) như sau:



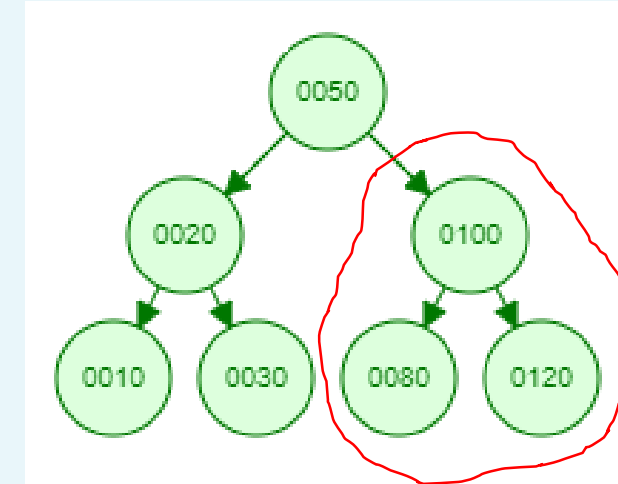
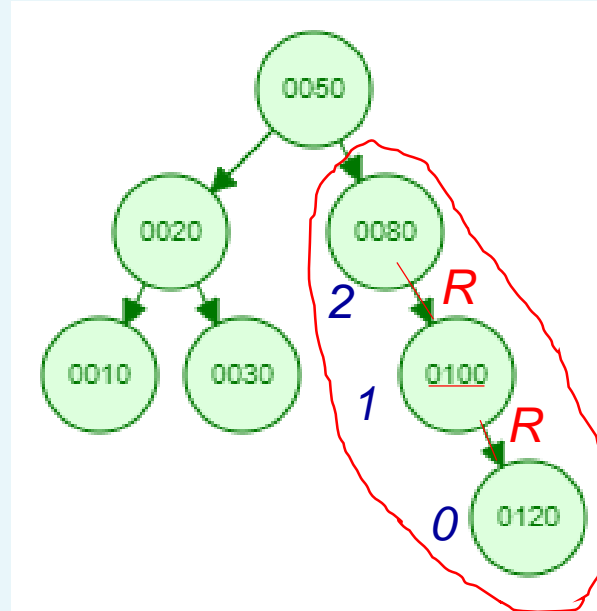
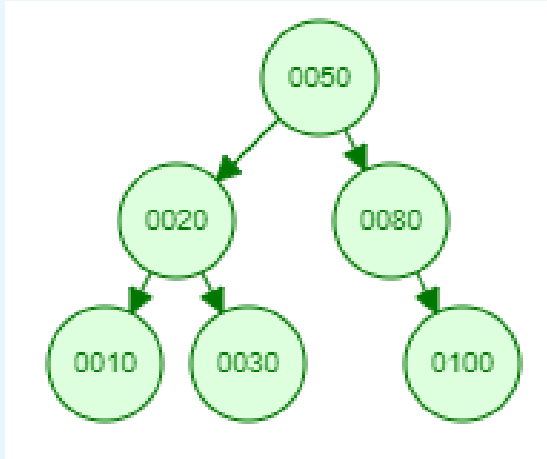


Ví dụ

- Thêm nút 120

Mất cân bằng bên phải
của con phải

Quay qua trái

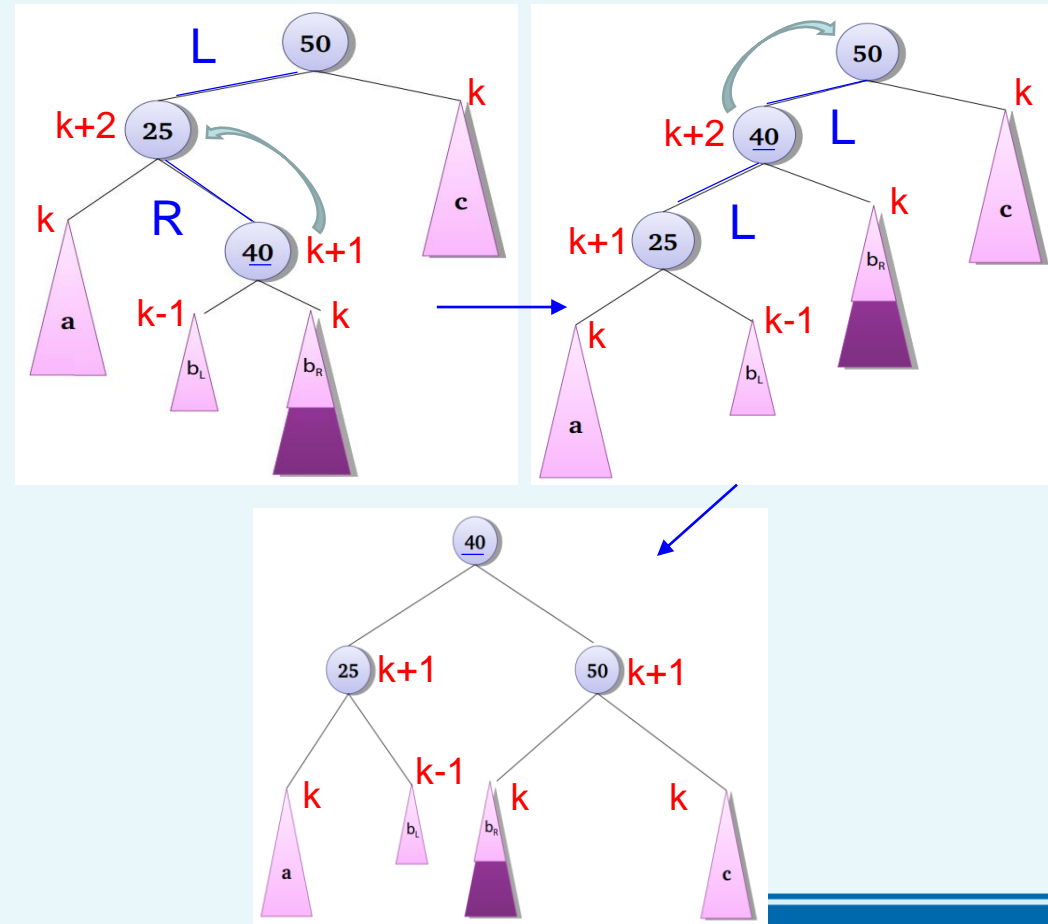
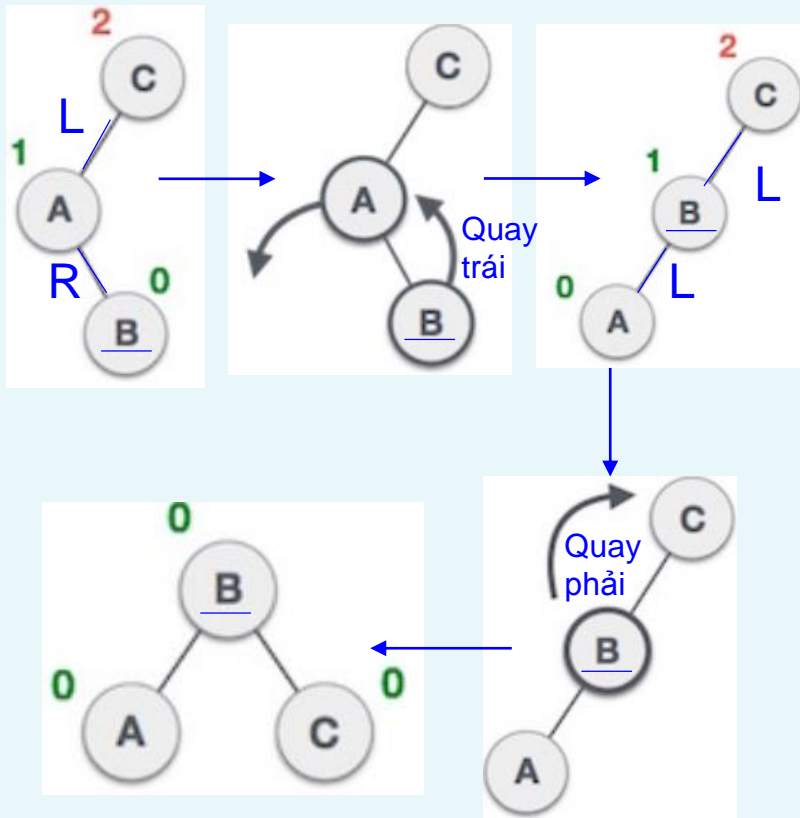




CANTHO UNIVERSITY

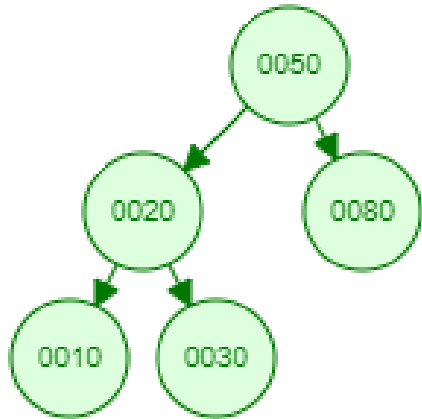
Các quy tắc xử lý khi cây mất cân bằng

- Trường hợp 3: Cây mất cân bằng **bên phải của con trái** ta thực hiện **quay kép trái-phải** (left-right rotate)

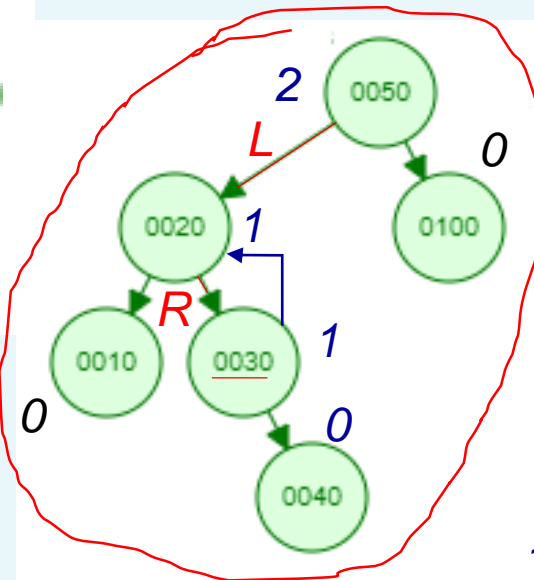




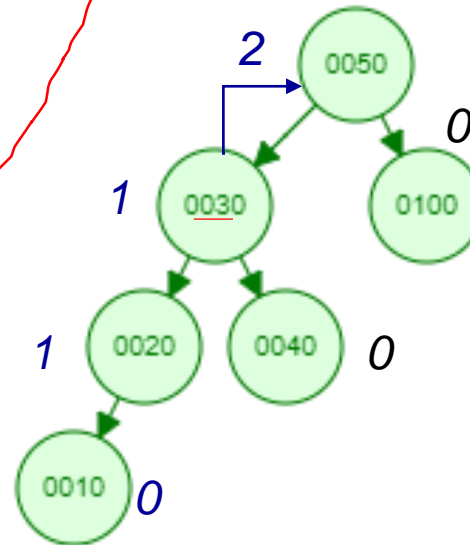
Ví dụ: Thêm nút 40 vào cây



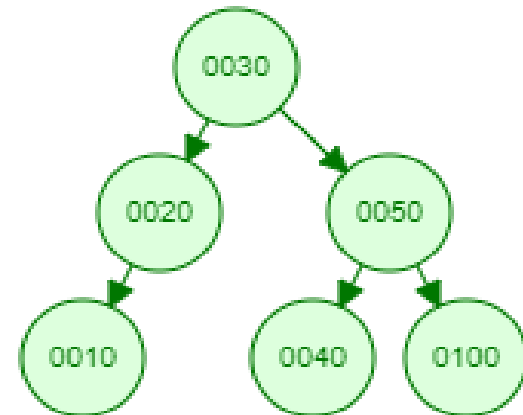
Mất cân bằng bên phải
của con trái



Quay qua trái



Quay qua phải

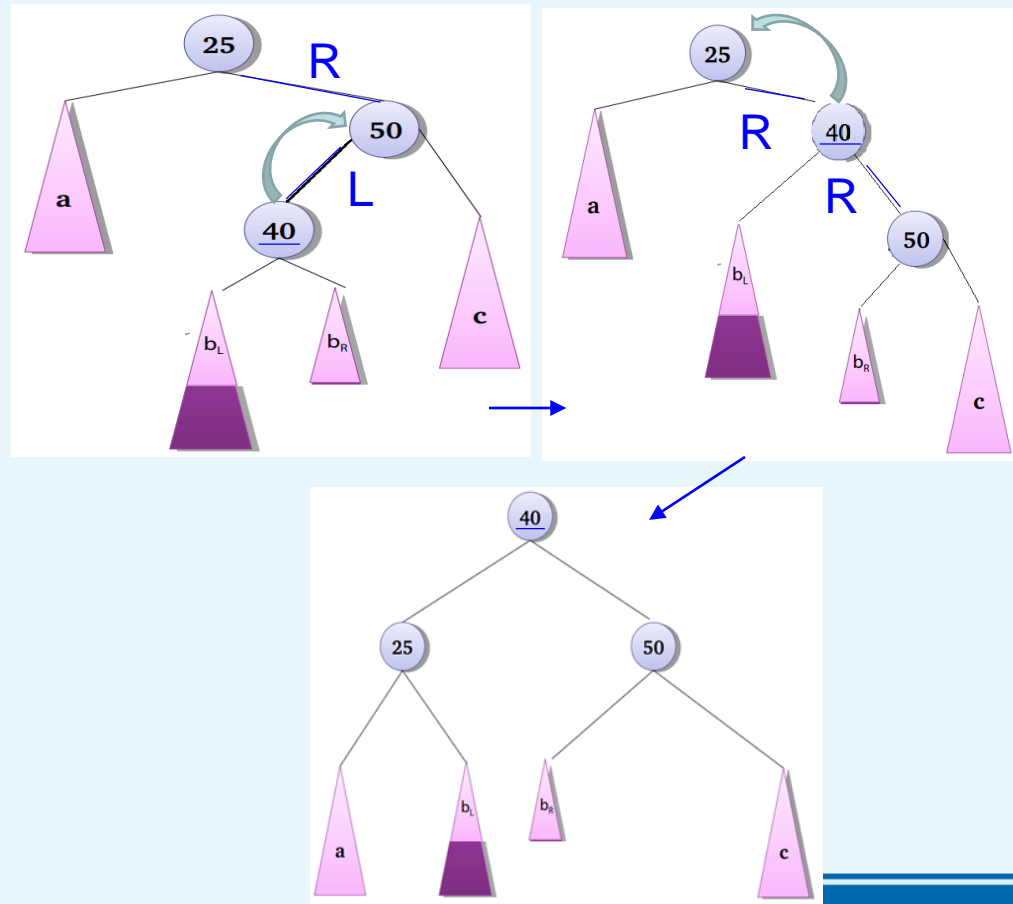
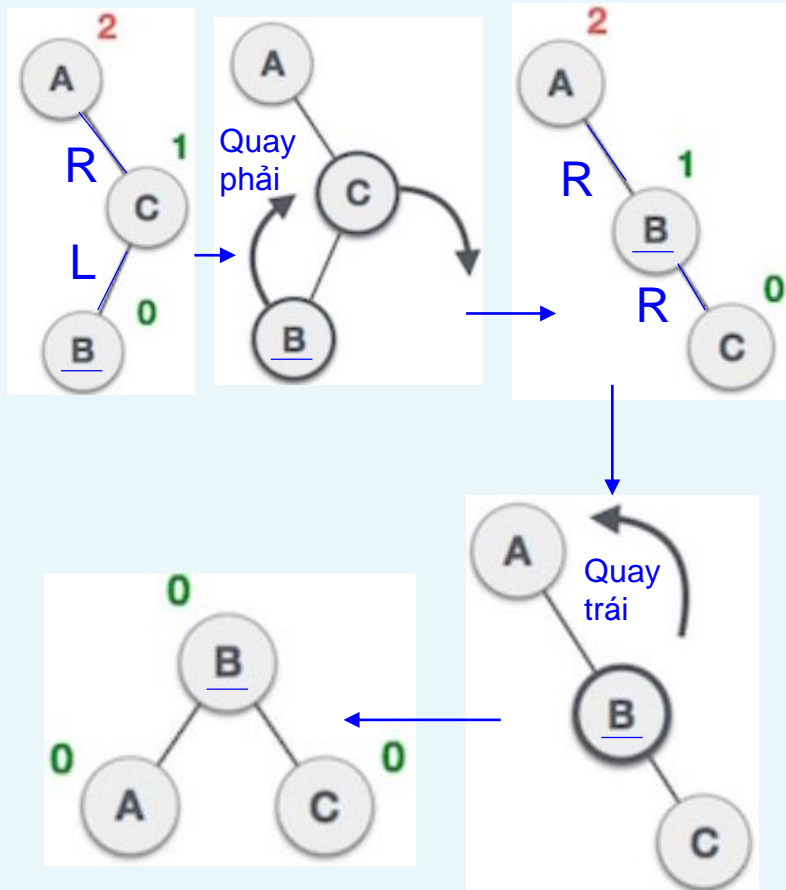




Các quy tắc xử lý khi cây mất cân bằng

CANTHO UNIVERSITY

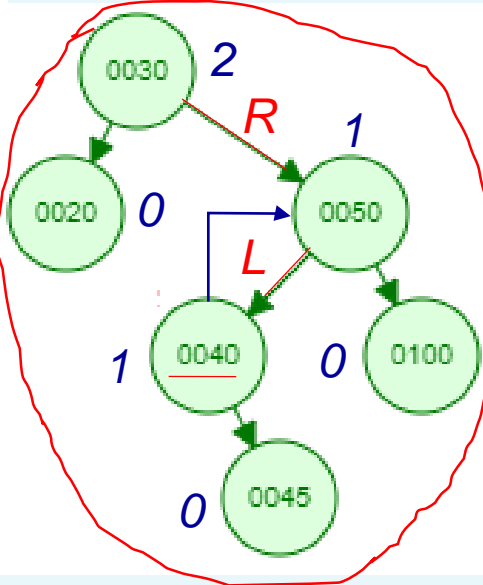
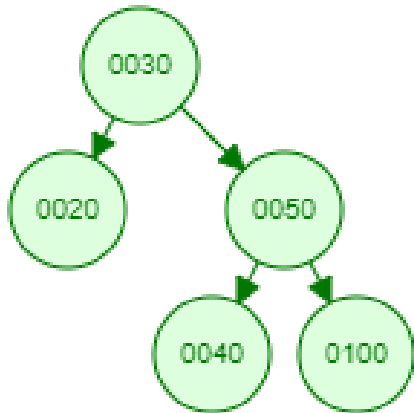
- Trường hợp 4: Cây mất cân bằng **bên trái của con phải**. Trường hợp này ta thực hiện **quay kép phải-trái** như sau:



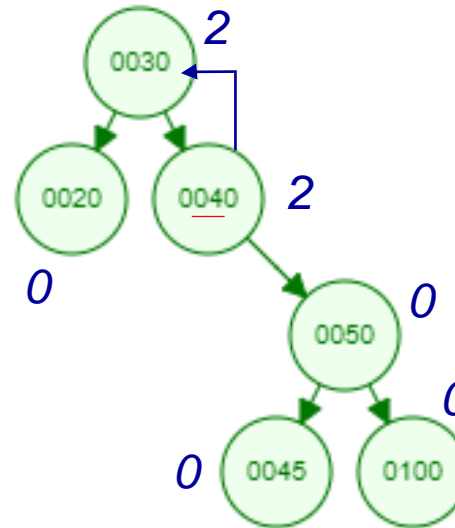


Ví dụ: Thêm nút 45

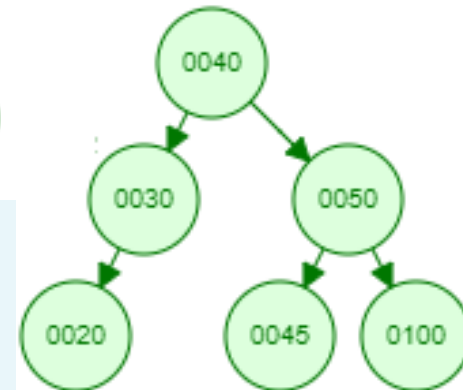
Mất cân bằng bên trái
của con phải



Quay qua phải



Quay qua trái

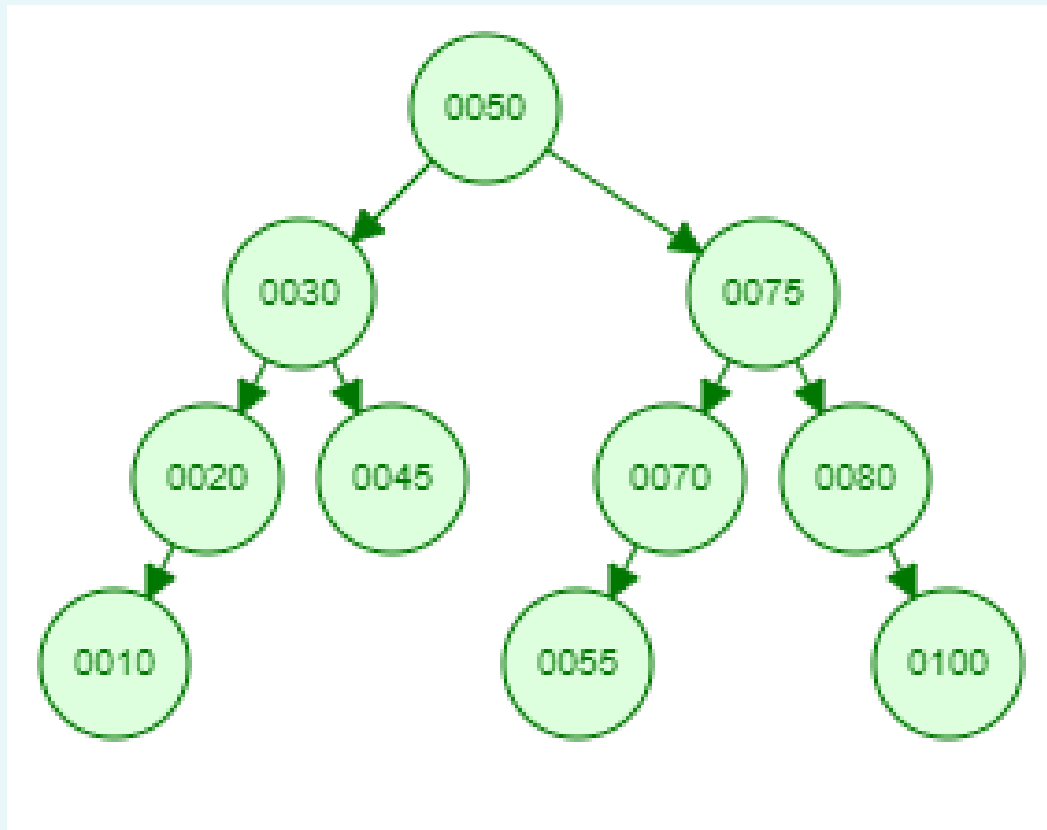




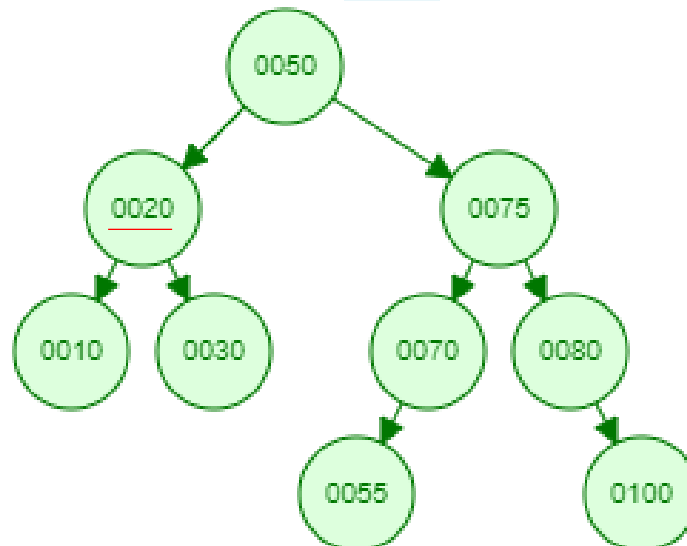
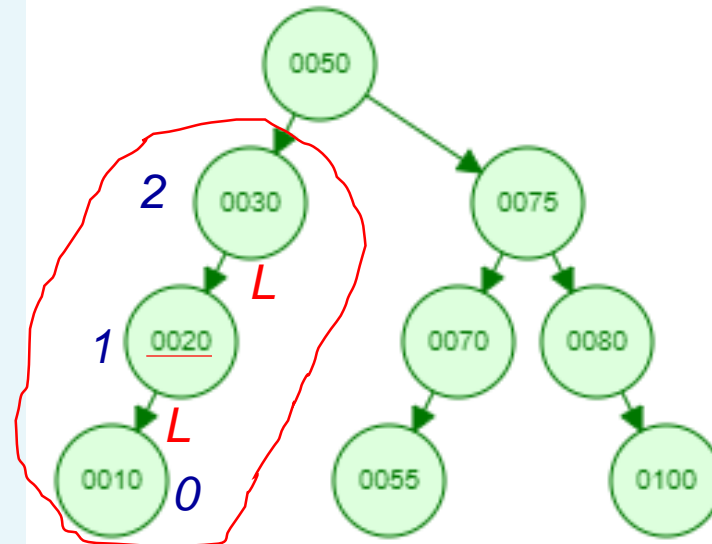
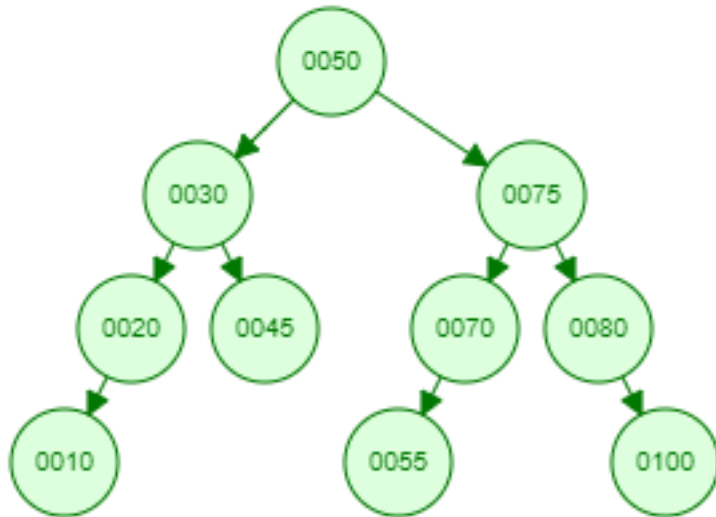
CANTHO UNIVERSITY

Vẽ cây AVL cho bởi danh sách

50, 70, 30, 10, 20, 45, 80, 75, 100, 55



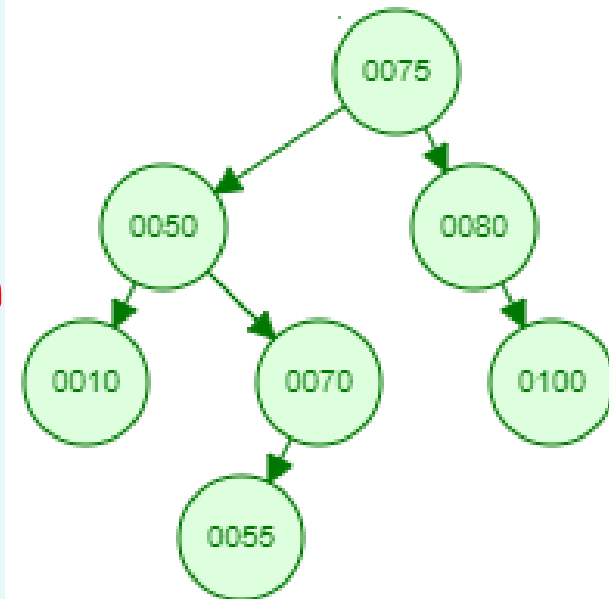
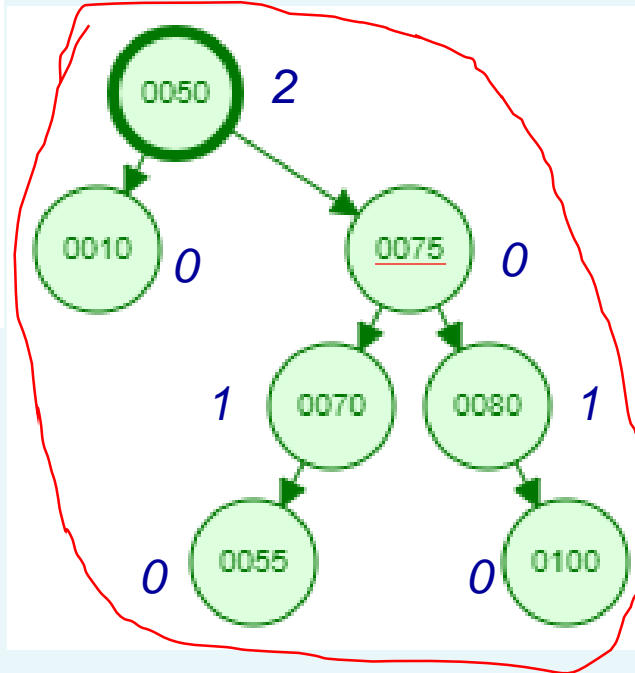
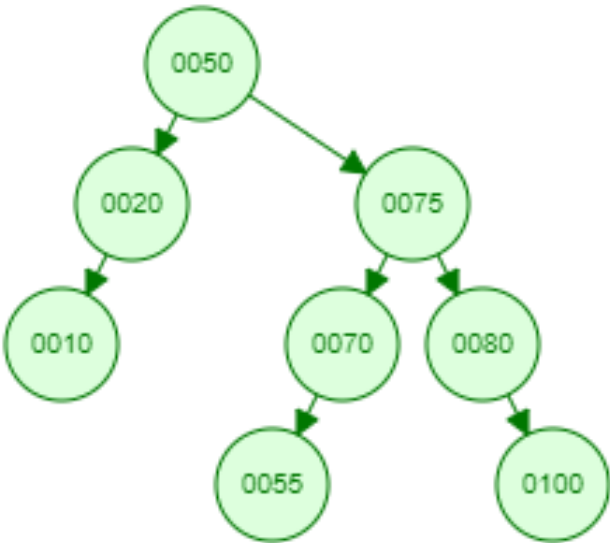
Xóa nút 45





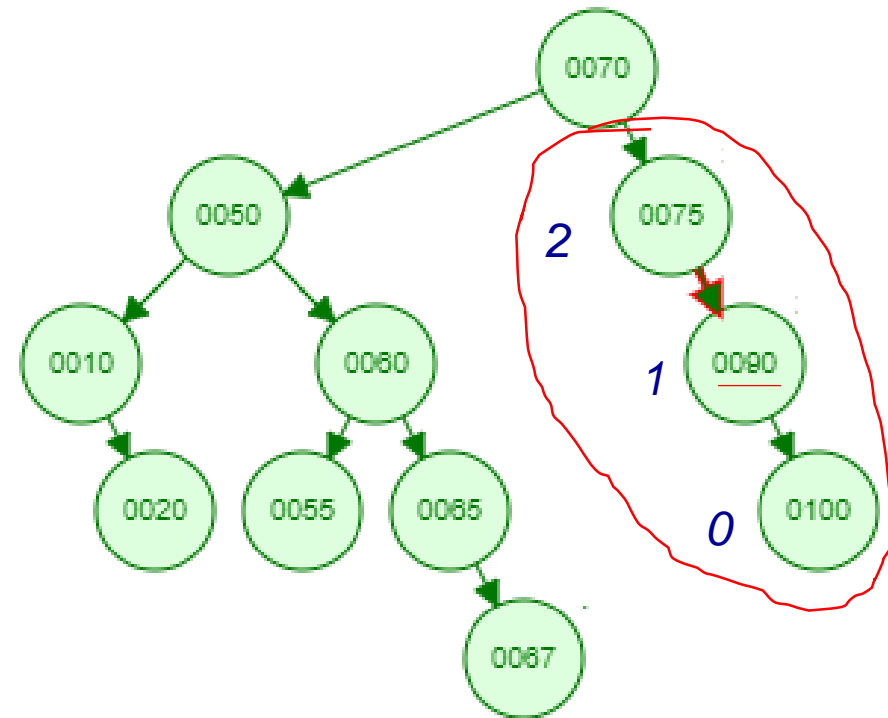
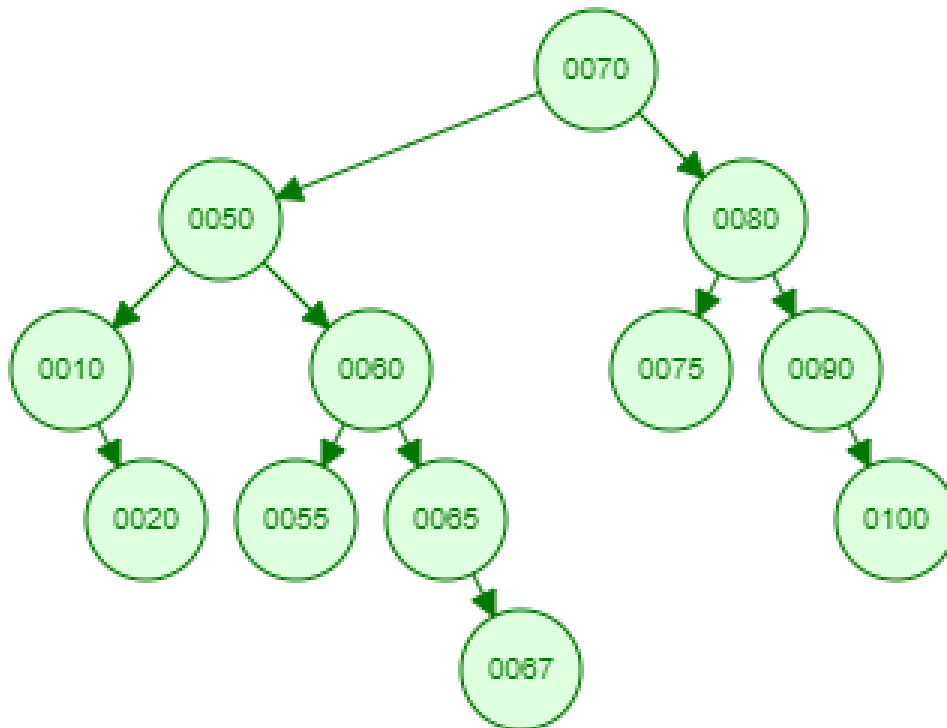
CANTHO UNIVERSITY

Xóa nút 20 trên cây





Xóa nút 80





CANTHO UNIVERSITY

Ý tưởng cài đặt cây

Mỗi nút trên cây cần có các thông tin:

- Khóa của nút
- Cây con trái
- Cây con phải
- Chỉ số cân bằng



Thêm nút vào cây AVL

- Thêm nút vào AVL tương tự như thêm nút vào cây BST thông thường. Sau đó ta xét xem cây có bị mất cân bằng hay không để thực hiện cân bằng lại cây theo các quy tắc xoay và cập nhật lại chỉ số cân bằng của các nút.



Xóa nút ra khỏi cây AVL

- Xóa nút ra khỏi AVL tương tự như xóa nút ra khỏi cây BST thông thường. Sau đó ta xét xem cây có bị mất cân bằng hay không để thực hiện cân bằng lại cây theo các quy tắc xoay và cập nhật lại chỉ số cân bằng của các nút.
- Quá trình xóa nút có thể dẫn đến cây mất cân bằng dây chuyền liên tiếp đến các nút tiền bối của nó. Theo đó, nguyên tắc thực hiện là phải cân bằng từ nút con trước rồi đến lên nút cha.



Khai báo cây AVL

```
typedef ... KeyType;
struct Node
{
    KeyType      Key;
    int          Bal;
    struct Node  *Left;
    struct Node  *Right;
};
typedef struct Node*  AVLTree;
```



Tài liệu tham khảo

- Adam Drozdek, Data structures and Algorithms Analysis in C++ 4th Edition, Cengage Learning, 2012
- Introduction to Algorithms, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest & Clifford Stein, MIT Press, 2012
- <https://www.cs.usfca.edu/~galles/cs245S08/lecture/lecture23.pdf>
- <http://www.cse.chalmers.se/edu/year/2018/course/DAT037/slides/6c-avl-trees.pdf>
- <https://www.cs.usfca.edu/~galles/visualization/AVLtree.html>
- https://www.tutorialspoint.com/data_structures_algorithms/avl_tree_algorithm.htm

Q&A?