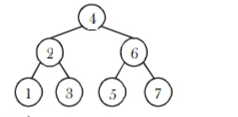
# Balanced Binary Search Trees

* Độ phức tạp thời gian O(n): Trong trường hợp phức tạp nhất của cây tìm kiếm nhị phân cân bằng đầy đủ, thời gian thực hiện các thao tác có độ phức tạp là O(n), với n là số nút trong cây.
* Ưu điểm O(logn) và hạn chế đối với cây nghiêng: Trong các phần trước, đã thấy rằng trường hợp xấu nhất của cây tìm kiếm là khi cây nghiêng, dẫn đến độ phức tạp O(n). Để giảm thiểu điều này, có thể áp đặt các hạn chế để đảm bảo thời gian thực hiện các thao tác có thể đạt đến độ phức tạp tốt hơn, chẳng hạn như O(logn).
* Cây cân bằng và chiều cao: Các cây cân bằng được đặc trưng bằng chiều cao HB(k), trong đó k là sự khác biệt giữa chiều cao của cây con trái và cây con phải. Hệ số cân bằng này giúp đánh giá mức độ cân bằng của cây.
* Hệ số cân bằng HB(k) và cây tìm kiếm nhị phân đầy đủ: Nếu hệ số cân bằng HB(k) bằng 0, tức là k = 0, thì cây được gọi là cây tìm kiếm nhị phân cân bằng đầy đủ. Trong trường hợp này, chênh lệch chiều cao giữa cây con bên trái và cây con bên phải là tối đa bằng 0, đảm bảo tính đầy đủ của cây.

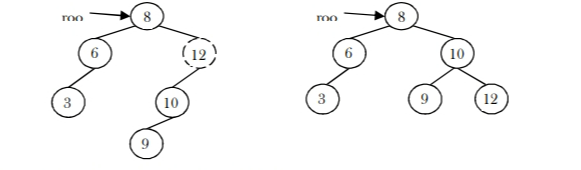
Ví dụ,



note: Để xây dựng cây HB(0), hãy tham khảo phần Vấn đề.

6.11 Cây AVL (Adelson-Velskii và Landis)

* Cây AVL là một dạng cây tìm kiếm nhị phân với hệ số cân bằng k = 1.
* Điều kiện cân bằng của cây AVL là chênh lệch giữa chiều cao của cây con bên trái và chiều cao của cây con bên phải nhiều nhất là 1.
* Một cây nhị phân được gọi là cây AVL nếu nó là cây tìm kiếm nhị phân và đối với bất kỳ nút X nào, chiều cao của cây con bên trái của X và chiều cao của cây con bên phải của X khác nhau nhiều nhất là 1.



số nút tối thiểu và tối đa trong cây AVL:

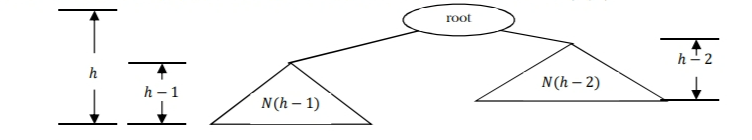
* Số nút tối thiểu trong cây AVL có chiều cao h, ký hiệu là N(h), được tính bằng cách lấp đầy cây với số lượng nút tối thiểu có thể. Điều này có nghĩa là nếu cây con bên trái có chiều cao h - 1 thì cây con bên phải sẽ có chiều cao h - 2. Số nút tối thiểu có chiều cao h là:

N(h)=N(h−1)+N(h−2)+1

* Số nút tối đa trong cây AVL được tính bằng cách lấp đầy cả cây con trái và cây con phải với chiều cao h-1. Kết quả là:

N(h)=2N(h−1)+1

* Trong cả hai trường hợp, thuộc tính của cây AVL đảm bảo rằng chiều cao của cây AVL có n nút là O(logn).



Khai báo cây AVL

Vì cây AVL là BST nên việc khai báo AVL tương tự như BST. Nhưng để đơn giản hóa các thao tác, chúng tôi cũng đưa chiều cao vào như một

phần của phần khai báo.

class AVLTreeNode:

    def \_\_init\_\_(self):

        self.data = None

        self.height = None

        self.left = None

        self.right = None

    def get\_data(self):

        return self.data

    def set\_data(self, data):

        self.data = data

    def get\_height(self):

        return self.height

    def set\_height(self, height):

        self.height = height

    def get\_left(self):

        return self.left

    def set\_left(self, left):

        self.left = left

    def get\_right(self):

        return self.right

    def set\_right(self, right):

        self.right = right

## Cây nhị phân cân bằng

là một loại cấu trúc dữ liệu cây nhị phân, trong đó độ sâu (chiều cao) của hai nhánh con của mọi nút không chênh lệch quá nhiều. Điều này giúp đảm bảo hiệu suất của các hoạt động như tìm kiếm, thêm và xoá dữ liệu được cải thiện so với cây nhị phân thông thường. Các ví dụ của cây nhị phân cân bằng bao gồm:

## Cây AVL (Adelson-Velsky và Landis Tree):

Một trong những loại cây nhị phân cân bằng đầu tiên, nơi mà độ chênh lệch chiều cao của hai nhánh con của mọi nút không quá 1.

## Cây Đỏ-Đen (Red-Black Tree):

Loại cây này đảm bảo cân bằng bằng cách sử dụng các quy tắc đặc biệt trong việc sắp xếp các nút màu đỏ và đen.

## Cây B-tree và B+ tree:

Được sử dụng chủ yếu trong các hệ thống quản trị cơ sở dữ liệu, nơi cần lưu trữ và truy cập dữ liệu một cách hiệu quả trên đĩa cứng.

## Cây Splay:

Loại cây nhị phân cân bằng tự điều chỉnh, nơi các nút được truy cập gần đây nhất sẽ được di chuyển lên gần gốc của cây để cải thiện hiệu suất truy cập tiếp theo.

## Cây AVL (Adelson-Velsky và Landis Tree)

là một loại cây nhị phân cân bằng, nơi mỗi nút duy trì một số cân bằng (balance factor) để đảm bảo rằng độ chênh lệch chiều cao của hai nhánh con không quá 1. Dưới đây là một ví dụ về việc mô phỏng cây AVL bằng Python, bao gồm các phương thức cho việc chèn và cân bằng lại cây:

## Định nghĩa Cấu trúc Nút:

Mỗi nút của cây chứa dữ liệu, tham chiếu đến nút con trái, nút con phải, và một yếu tố cân bằng.

## Chèn Một Nút:

Quá trình này gồm việc chèn một nút mới và sau đó cân bằng lại cây nếu cần.

## Cân Bằng Lại Cây:

Cân bằng lại cây bằng cách sử dụng các phép quay (rotation) để đảm bảo cây luôn cân bằng.

## Cập Nhật Yếu Tố Cân Bằng:

Cập nhật yếu tố cân bằng sau mỗi lần chèn hoặc xoá.

## Phép Quay:

Các phép quay bao gồm quay trái, quay phải, quay trái-phải và quay phải-trái để giữ cho cây cân bằng.