Thuật toán cây AVL

Cây AVL là một cấu trúc dữ liệu tương tự như cây đỏ đen, được thiết kế để giảm thiểu chi phí của các phép cập nhật, bao gồm cả chèn và xóa. Các nút trong cây AVL được đánh số thứ tự theo giá trị các nút, và phải tuân thủ một số quy tắc nhất định để đảm bảo rằng cây luôn cân bằng và có độ cao O(log n).

Quy tắc

Các quy tắc sau đây phải được tuân thủ để đảm bảo rằng cây AVL luôn cân bằng:

1. Chiều cao của cây con trái và cây con phải của một nút không quá chênh lệch 1.
2. Cả cây con trái và cây con phải của một nút đều là cây AVL.

Các thao tác trên cây AVL

Các thao tác trên cây AVL bao gồm:

- Chèn (insert)
- Xóa (delete)
- Tìm kiếm (search)

Các thao tác này được triển khai bằng các hàm phù hợp. Dưới đây là đoạn mã Python mẫu cho các thao tác trên cây AVL:

```
class AVLTree:
    def __init__(self):
        self.root = None

    def height(self, node):
        if node is None:
            return 0
        return node.height

    def balance_factor(self, node):
        if node is None:
            return 0
        return self.height(node.left) - self.height(node.right)

    def rotate_left(self, x):
        y = x.right
```

```
        z = y.left
        y.left = x
        x.right = z
        x.height = max(self.height(x.left), self.height(x.right)) + 1
        y.height = max(self.height(y.left), self.height(y.right)) + 1
        return y

    def rotate_right(self, y):
        x = y.left
        z = x.right
        x.right = y
        y.left = z
        y.height = max(self.height(y.left), self.height(y.right)) + 1
        x.height = max(self.height(x.left), self.height(x.right)) + 1
        return x

    def insert(self, key):
        self.root = self._insert(self.root, key)

    def _insert(self, node, key):
        if node is None:
            return Node(key)
        elif key < node.key:
            node.left = self._insert(node.left, key)
        else:
            node.right = self._insert(node.right, key)

        node.height = max(self.height(node.left), self.height(node.right)) + 1

        balance = self.balance_factor(node)

        if balance > 1 and key < node.left.key:
            return self.rotate_right(node)

        if balance < -1 and key > node.right.key:
            return self.rotate_left(node)

        if balance > 1 and key > node.left.key:
            node.left = self.rotate_left(node.left)
            return self.rotate_right(node)
```

```
        if balance < -1 and key < node.right.key:
            node.right = self.rotate_right(node.right)
            return self.rotate_left(node)
```

Revision #2
Created 27 June 2023 10:59:58 by Laptrinh.vn
Updated 27 June 2023 11:00:49 by Laptrinh.vn