AVL TREE

Đặng Quý Anh, Hoàng Anh Đức,
Lê Ngọc Lâm

Ngành: Toán Tin
Trường Đại học Khoa học Tự nhiên, ĐHQG Hà Nội

Ngày 22 tháng 12 năm 2021

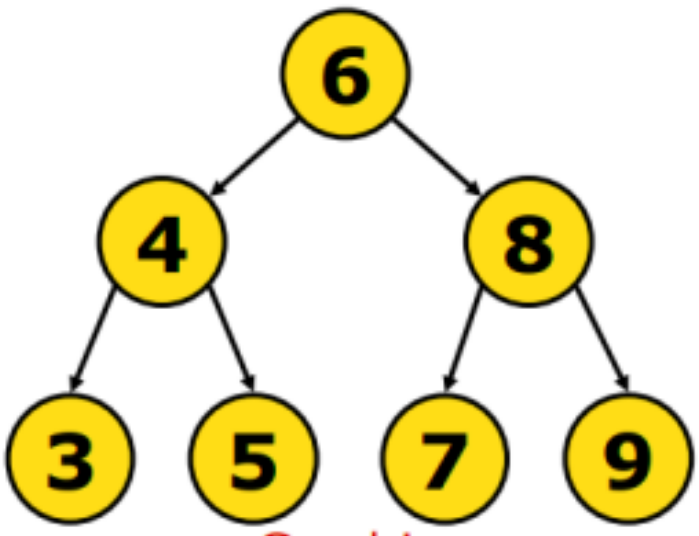
Nội dung

1. Vì sao là cây AVL?
2. Cây AVL
   - Tổng quát
   - Nhân tố cân bằng
   - Phép xoay
   - Tự cân bằng
3. Các phương thức cơ bản
   - Tìm kiếm phần tử nhỏ nhất
   - Tìm kiếm phần tử
   - Thêm một phần tử
   - Xóa đi một phần tử
4. Ứng dụng
5. Tài liệu tham khảo

Nội dung

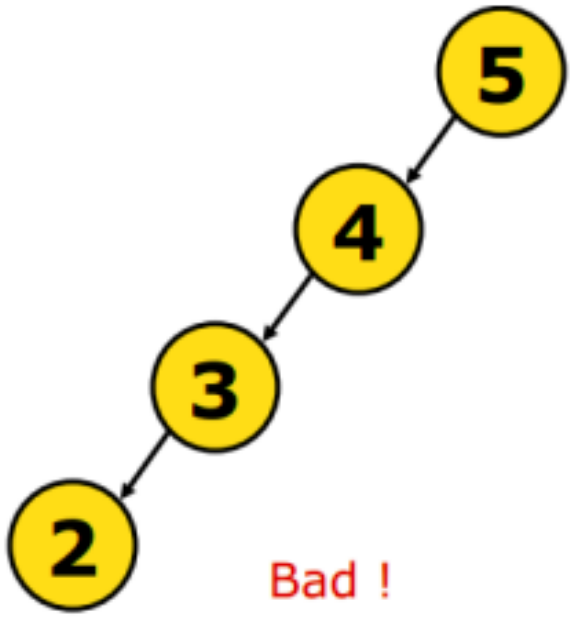
1. Vì sao là cây AVL?
2. Cây AVL
   - Tổng quát
   - Nhân tố cân bằng
   - Phép xoay
   - Tự cân bằng
3. Các phương thức cơ bản
   - Tìm kiếm phần tử nhỏ nhất
   - Tìm kiếm phần tử
   - Thêm một phần tử
   - Xóa đi một phần tử
4. Ứng dụng
5. Tài liệu tham khảo

Vì sao là cây AVL?



Good !

h = O(log n)



Bad !

h = O(n)

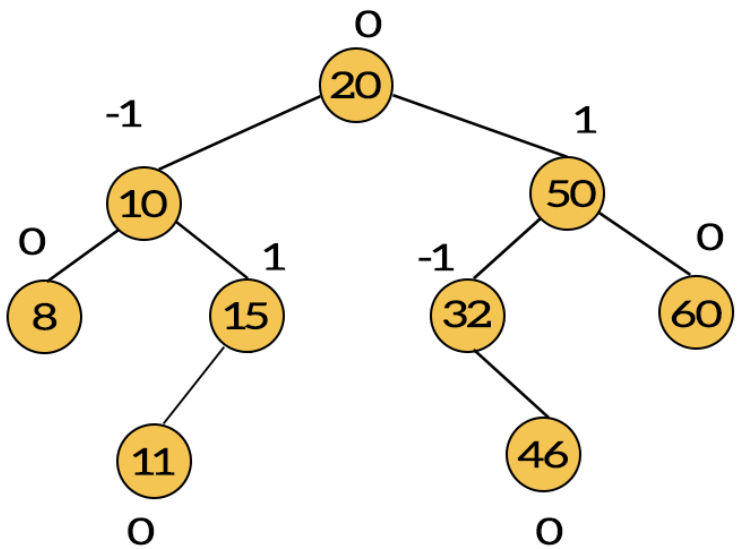
Nội dung

1. Vì sao là cây AVL?
2. Cây AVL
   - Tổng quát
   - Nhân tố cân bằng
   - Phép xoay
   - Tự cân bằng
3. Các phương thức cơ bản
   - Tìm kiếm phần tử nhỏ nhất
   - Tìm kiếm phần tử
   - Thêm một phần tử
   - Xóa đi một phần tử
4. Ứng dụng
5. Tài liệu tham khảo

Cây AVL

Tổng quát

- Cây AVL là cây tìm kiếm nhị phân tự cân bằng
- Được đặt tên theo Georgy Adelson-Velsky và Evgenii Landis
- Chiều cao của cây $h < 2logn + 2 \Rightarrow h = O(logn)$



Cây AVL

Nhân tố cân bằng

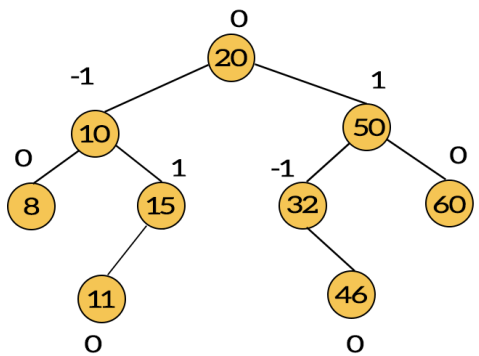
- $BF(T) = Height(LeftSubtree(T)) - Height(RightSubtree(T))$
  trong đó,
  - BF (balance factor) là nhân tố cân bằng
  - T là một cây nhị phân tìm kiếm
  - Height là chiều cao của cây
  - RightSubtree(T) là cây con phải của T
  - LeftSubtree(T) là cây con trái của T
- T là cây AVL thì $BF(T) \in \{-1, 0, 1\}$
- Chiều cao của cây
  - Là level cao nhất của nút gốc được tính từ lá
  - Nút lá có level bằng 1

Cây AVL

Nhân tố cân bằng

```
getBalance(T) {
    if (T is empty)
        return 0
    return getHeight(T.left) - getHeight(T.right)
}
```

- $getHeight(46) = 1, getBalance(46) = 0$
- $getHeight(32) = 2, getBalance(32) = -1$
- ...
- $getHeight(20) = 4, getBalance(20) = 0$
- ...

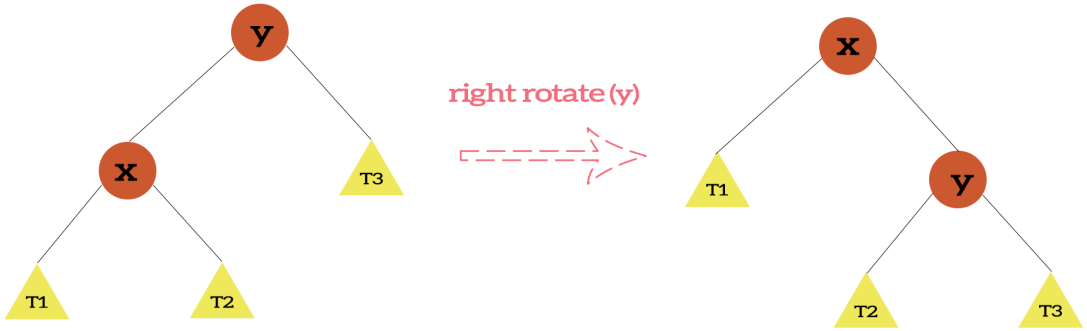


Cây AVL

Phép xoay

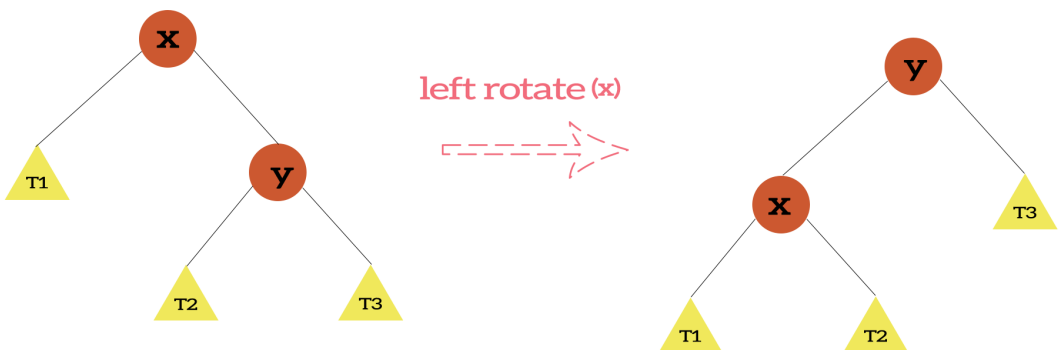
I. Xoay phải

```
rightRotation(y) {
    x = y.left
    T2 = x.right
    // Perform rotation
    x.right = y
    y.left = T2
    // Update heights
    y.height = getMax(getHeight(y.left), getHeight(y.right)) + 1
    x.height = getMax(getHeight(x.left), getHeight(x.right)) + 1
    return x
}
```



II. Xoay trái

```
leftRotation(x) {
    y = x.right
    T2 = y.left
    // Perform rotation
    y.left = x
    x.right = T2
    //  Update heights
    x.height = getMax(getHeight(x.left), getHeight(x.right)) + 1
    y.height = getMax(getHeight(y.left), getHeight(y.right)) + 1
    return y;
}
```

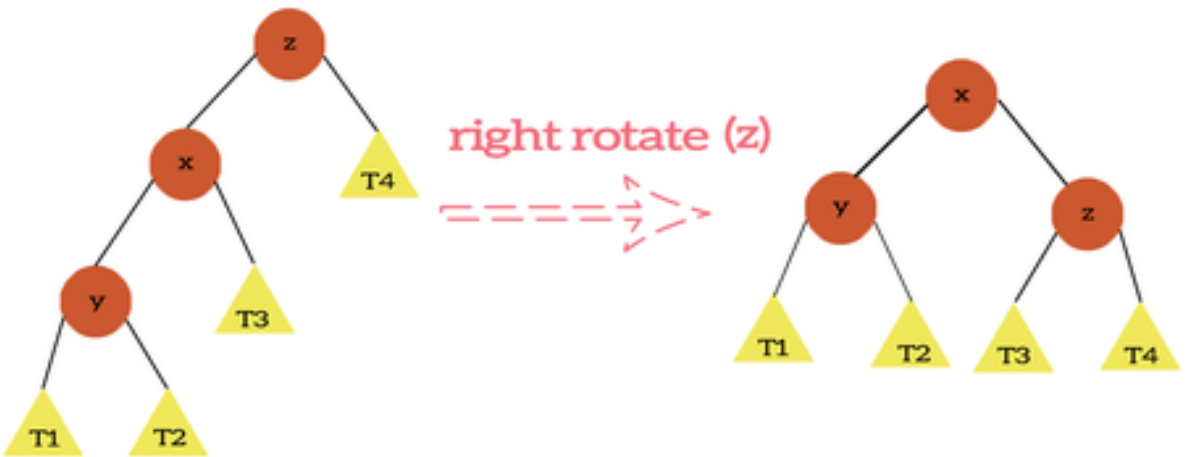


Cây AVL

Tự cân bằng

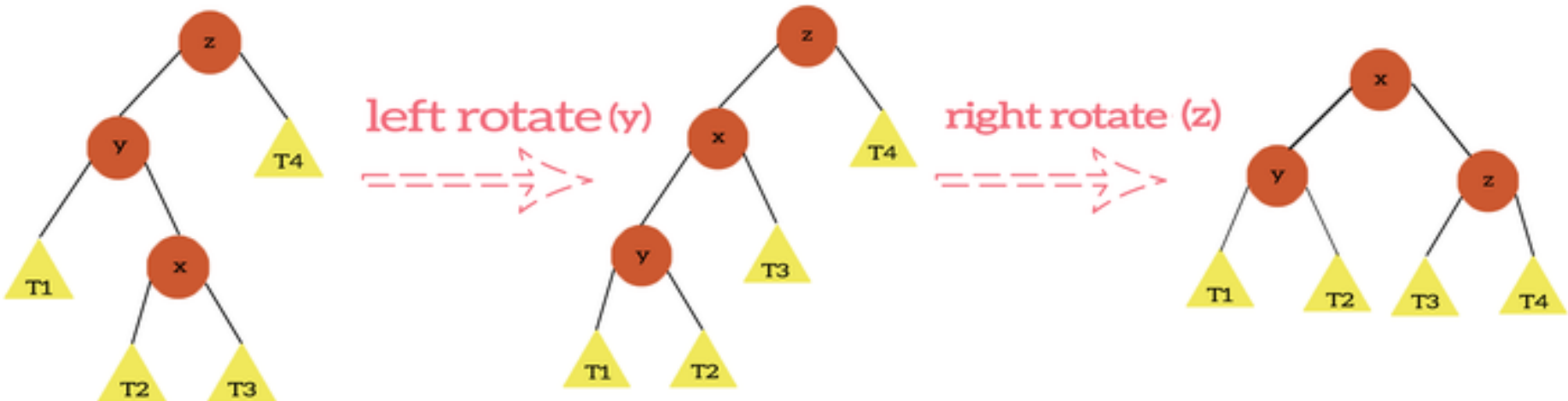
I. Trái - Trái

- $BF(z) > 1$
- $BF(z.left) \geq 0$



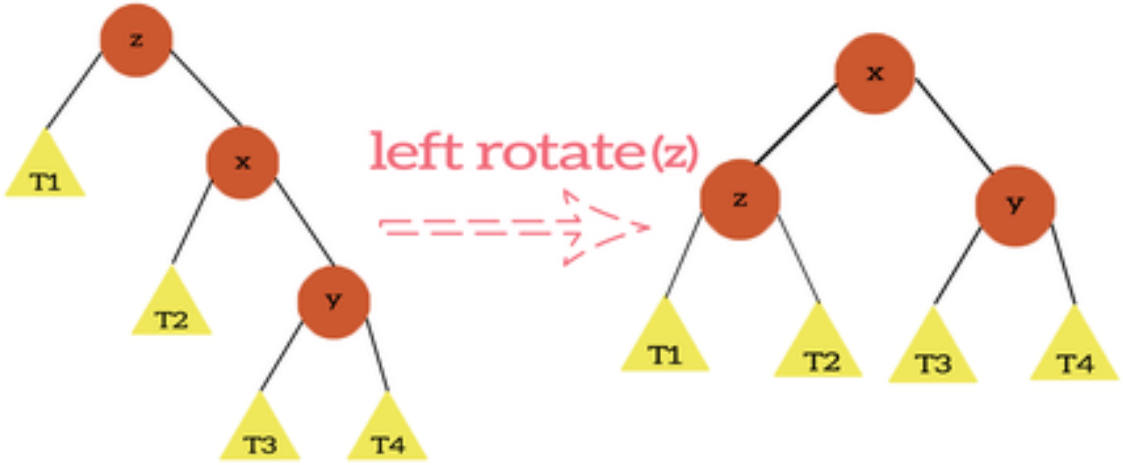
II. Trái - Phải

- $BF(z) > 1$
- $BF(z.left) < 0$



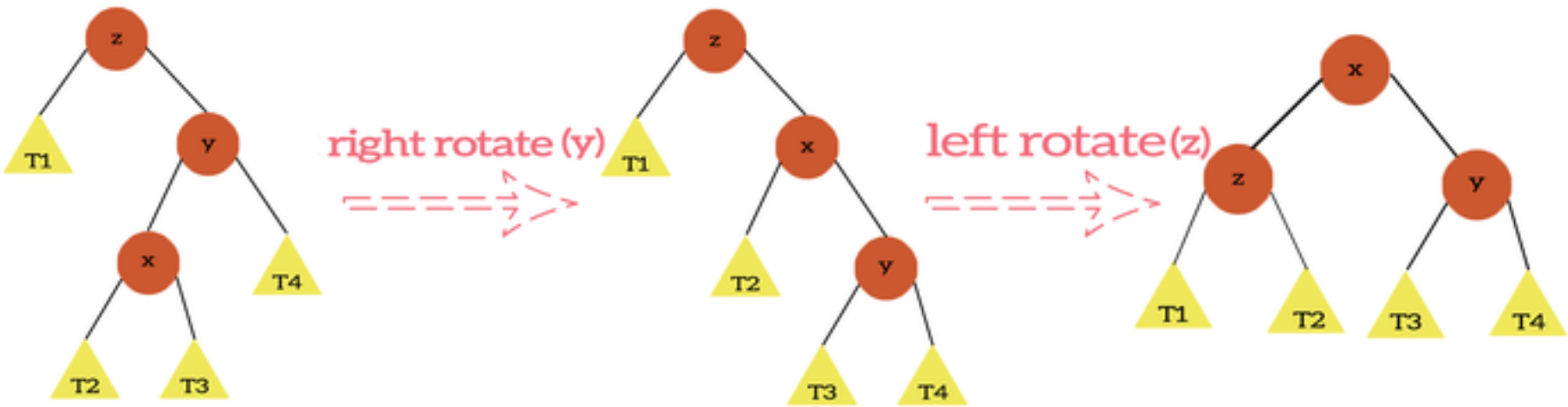
III. Phải - Phải

- $BF(z) < -1$
- $BF(z.right) \leq 0$



IV. Phải - Trái

- $BF(z) < -1$
- $BF(z.right) > 0$



IV. Cân bằng

```
rebalance(node) {
    balance = getBalance(node)
    // If this node becomes unbalanced, then there are 4 cases
    // Left Left Case
    if (balance > 1 && getBalance(node.left) >= 0)
        return rightRotation(node)
    // Left Right Case
    if (balance > 1 && getBalance(node.left) < 0)
        node.left = leftRotation(node.left)
        return rightRotation(node)
    // Right Right Case
    if (balance < -1 && getBalance(node.right) <= 0)
        return leftRotation(node)
    // Right Left Case
    if (balance < -1 && getBalance(node.right) > 0)
        node.right = rightRotation(node.right)
        return leftRotation(node)

    return node
}
```

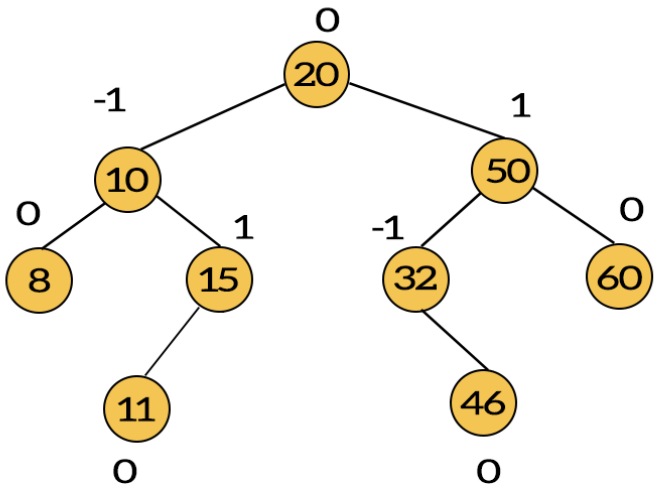
Nội dung

1. Vì sao là cây AVL?
2. Cây AVL
   - Tổng quát
   - Nhân tố cân bằng
   - Phép xoay
   - Tự cân bằng
3. Các phương thức cơ bản
   - Tìm kiếm phần tử nhỏ nhất
   - Tìm kiếm phần tử
   - Thêm một phần tử
   - Xóa đi một phần tử
4. Ứng dụng
5. Tài liệu tham khảo

Các phương thức cơ bản

Tìm kiếm phần tử nhỏ nhất

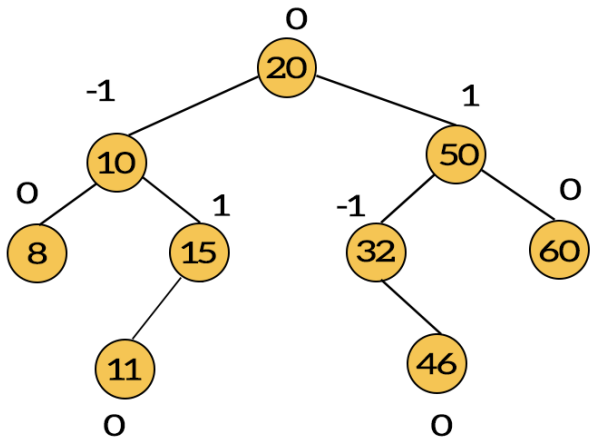
```
findMin(T) {
    while (T.left is not empty)
        T = T.left
    return T.element
}
```



Các phương thức cơ bản

Tìm kiếm phần tử

```
search(x, T) {
    if (T is empty )
        return null
    else ( x == T.element)
        return T
    else if ( x < T.element)
        return search(x, T.left)
    else
        return search(x, T.right)
}
```



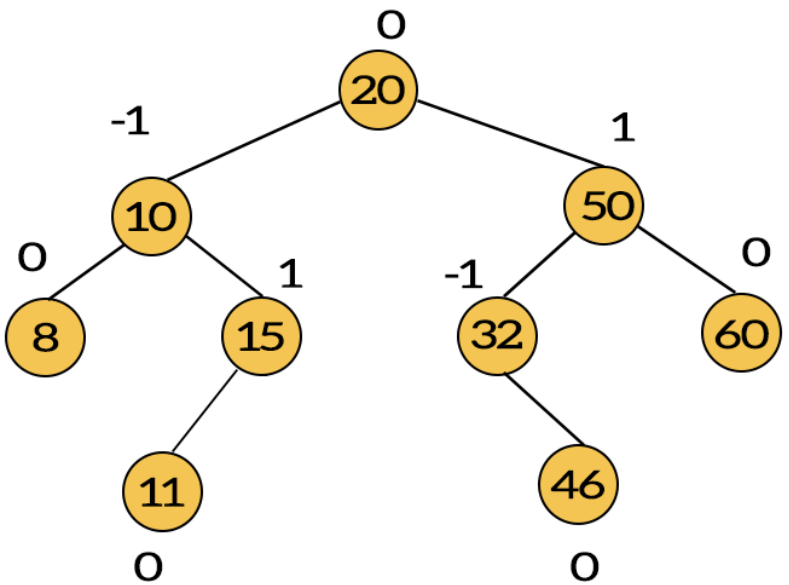
Running time: $O(logn)$

Ví dụ: search(10, T)

Các phương thức cơ bản

Thêm một phần tử

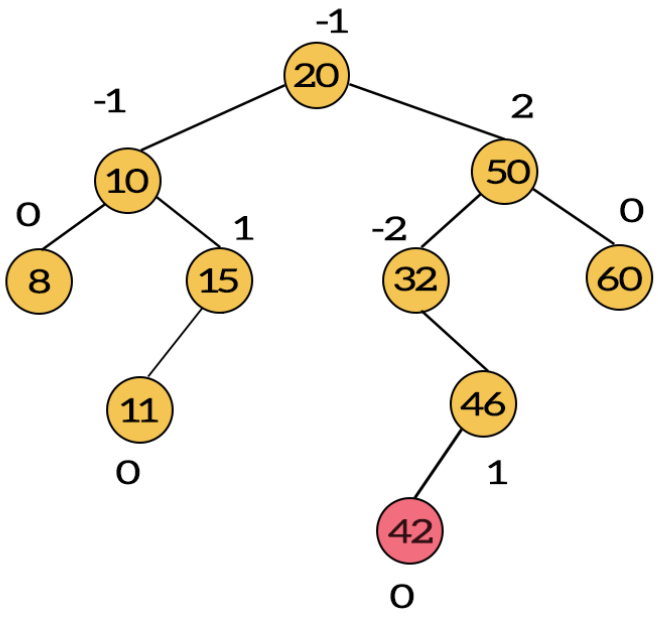
Thêm phần tử 42 vào cây AVL như thế nào?

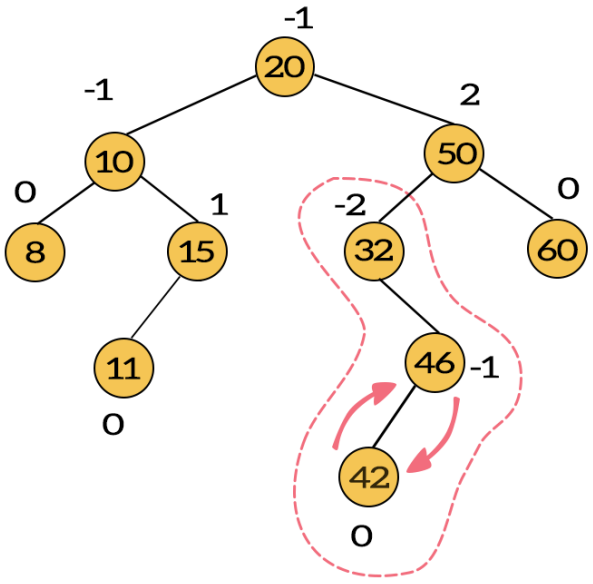


Các phương thức cơ bản

Thêm một phần tử

Trường hợp: Phải-Trái

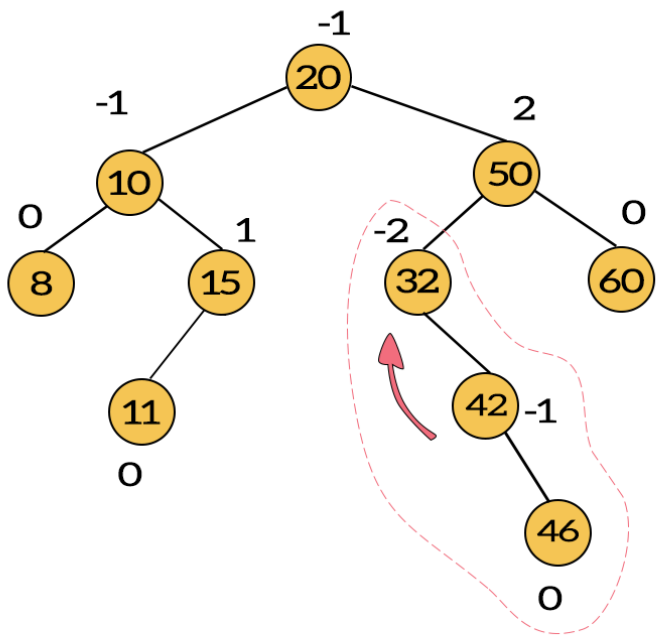


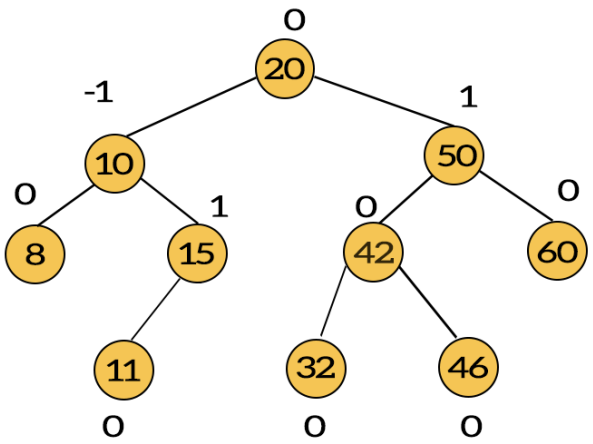


Các phương thức cơ bản

Thêm một phần tử

Cân bằng: Xoay phải -> Xoay trái





Các phương thức cơ bản

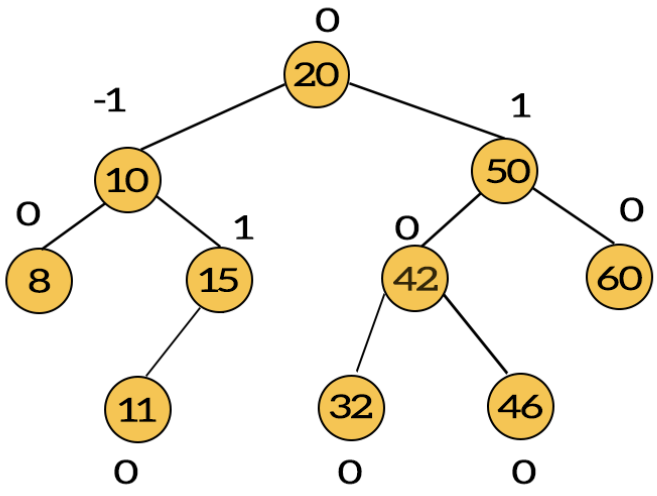
Thêm một phần tử

```
insert(x, T) {
    if (T is empty)
        // create new AVL tree with root x
        return new AVLTree(x)
    else if (x < T.element)
        T.left = insert(x, T.left)
    else if
        (x > T.element)
        T.right = insert(x, T.right)
    else
        Error! // Element is already in tree
    // Update height
    T.height = getMax(getHeight(root.left), getHeight(root.right)) + 1
    T = rebalance(T)
    return T
}
```

Running time: $O(logn)$

Các phương thức cơ bản

Xóa đi một phần tử

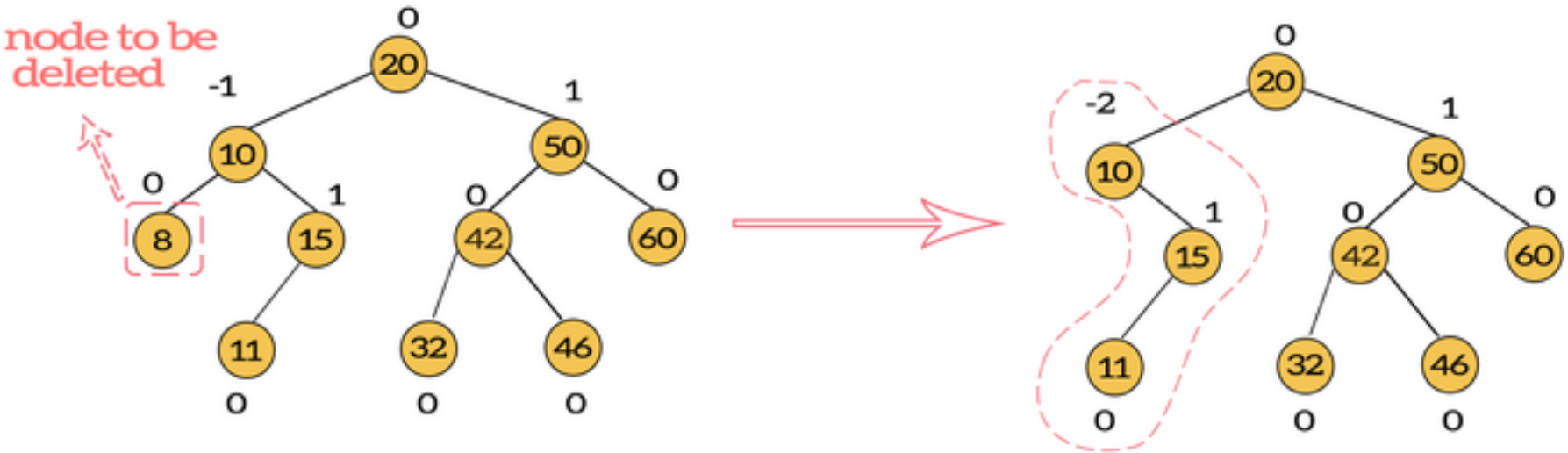


Làm thế nào để xóa đi một phần tử từ cây AVL mà vẫn đảm bảo điều kiện cân bằng?

Các phương thức cơ bản

Xóa đi một phần tử - Không có con

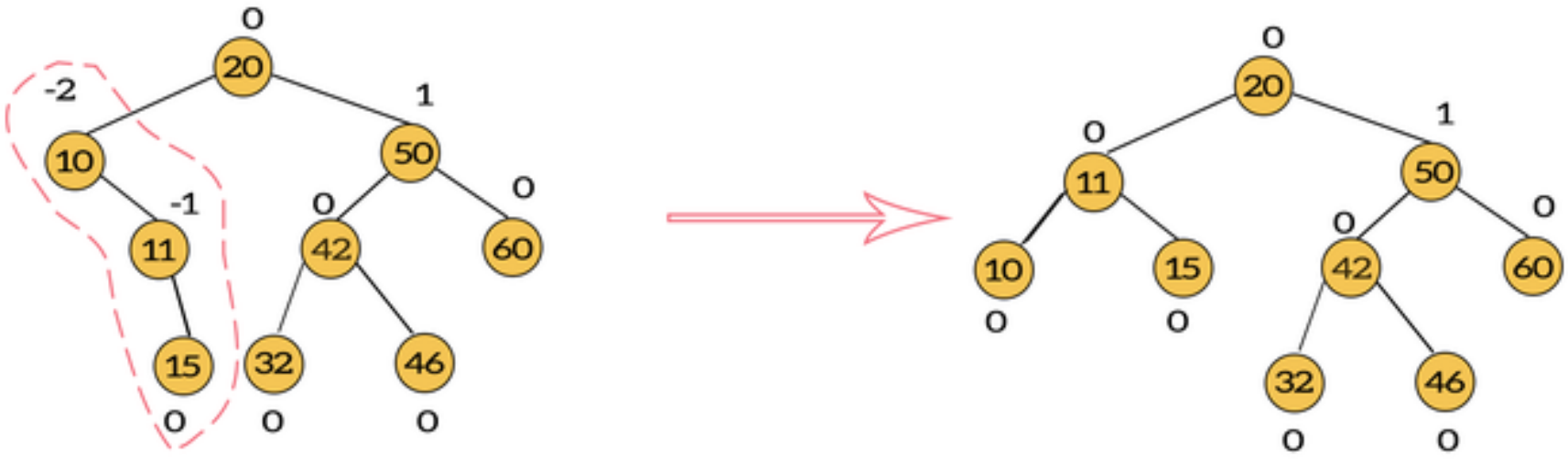
Xoá đi phần tử 8?
Trường hợp: Phải-Trái



Các phương thức cơ bản

Xóa đi một phần tử - Không có con

Cân bằng: Xoay phải - Xoay trái



Các phương thức cơ bản

Xóa đi một phần tử - Không có con

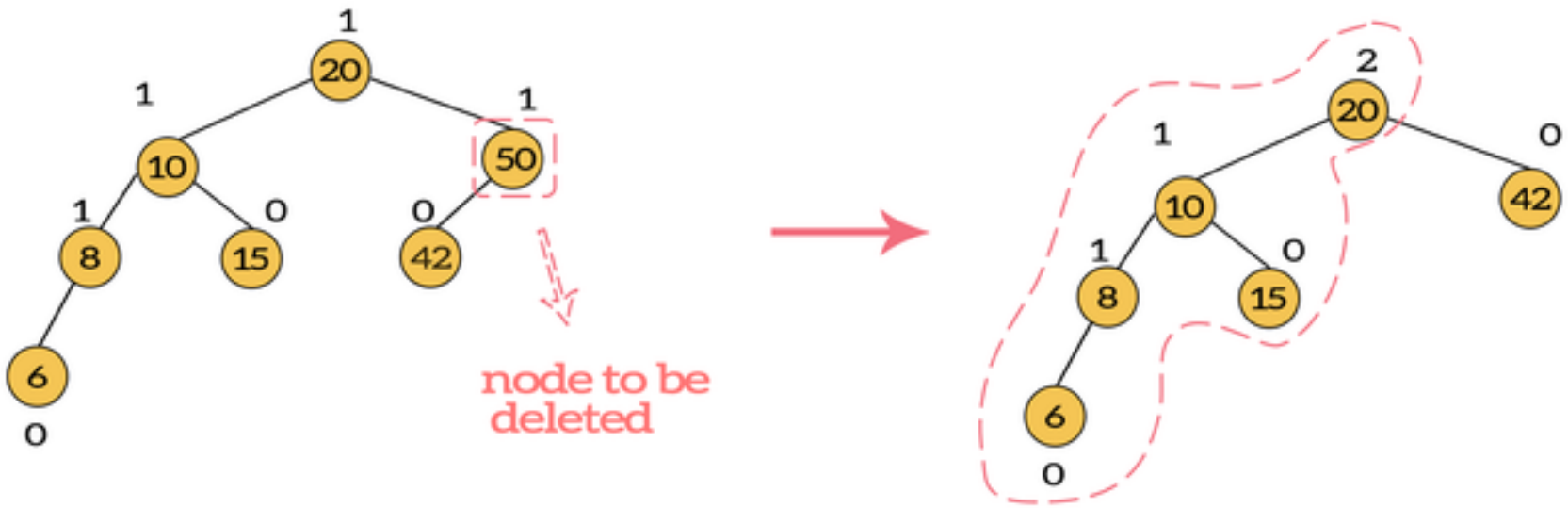
```
delete (x,T) {
    if (T has no children)
        if (T.element == x)
            return empty tree
        else
            NOT FOUND
    T.height = getMax(getHeight(T.left), getHeight(T.right)) + 1
    T = rebalance(T)
    return T
}
```

Running time: $O(logn)$

Các phương thức cơ bản

Xóa đi một phần tử - Có một con

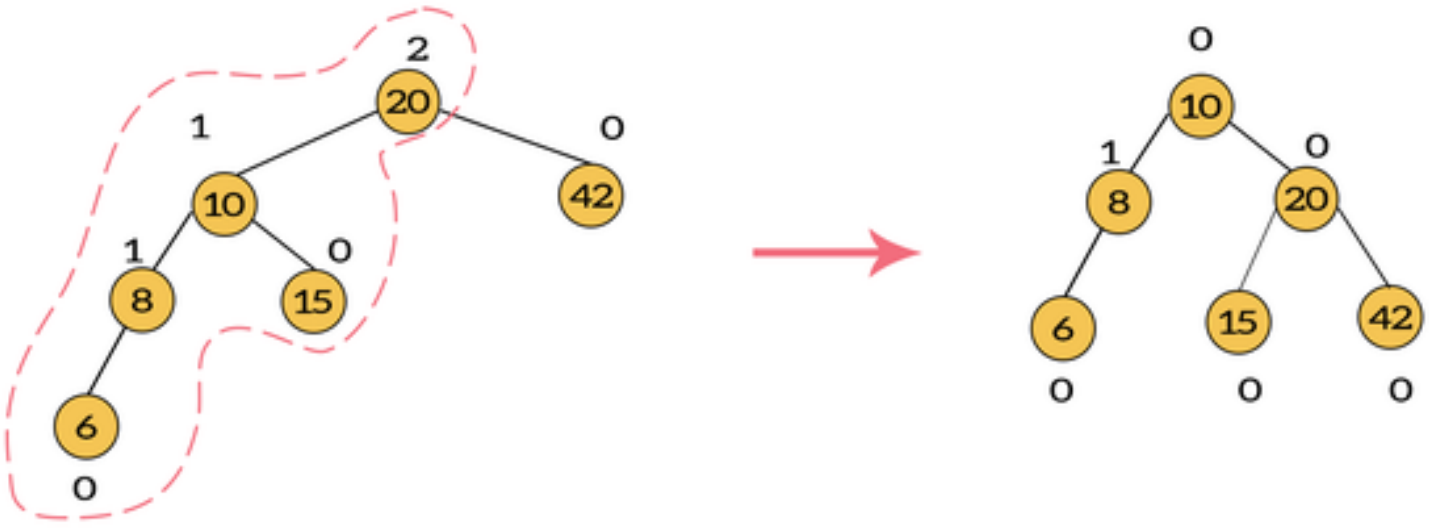
Xóa đi phần tử 50?
Trường hợp: Phải - Phải



Các phương thức cơ bản

Xóa đi một phần tử - Có một con

Cân bằng: Xoay trái



Các phương thức cơ bản

Xóa đi một phần tử - Có một con

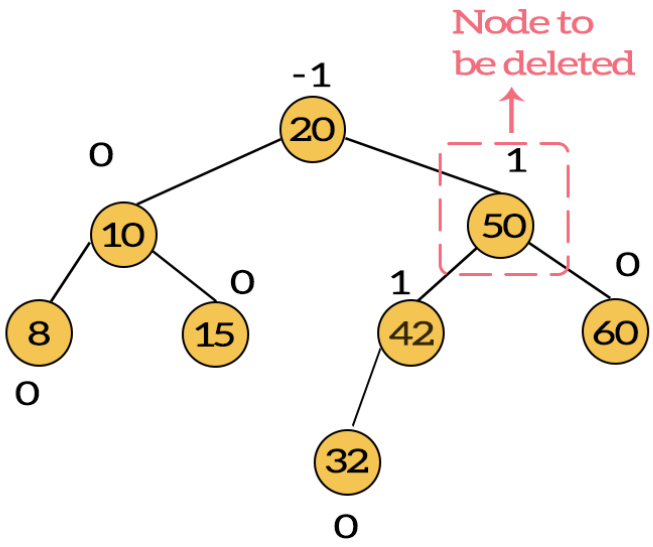
```
delete (x,T) {
    if (T has only 1 child (left)) // right
        if (x==T.element)
        return T.left // T.right
    else
        T.left=delete(x,T.left) // T.right
    T.height = getMax(getHeight(T.left), getHeight(T.right)) + 1
    T = rebalance(T)
    return T
}
```

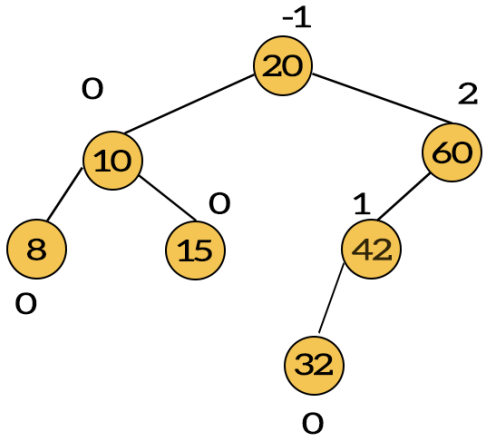
Running time: $O(logn)$

Các phương thức cơ bản

Xóa đi một phần tử - Có hai con

Xóa phần tử 50 khỏi cây AVL?
Trường hợp: Trái - Trái

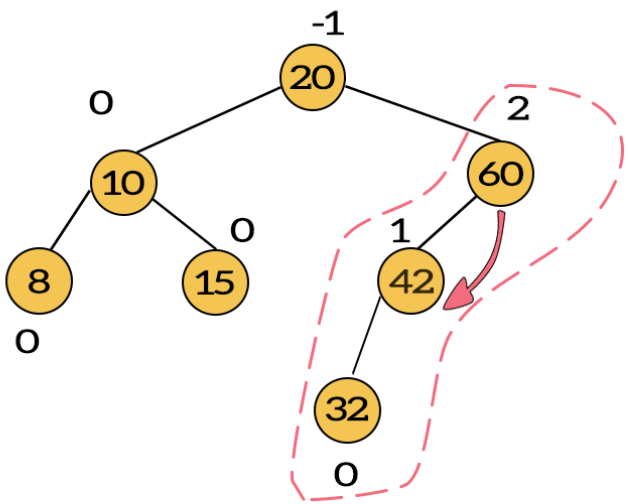


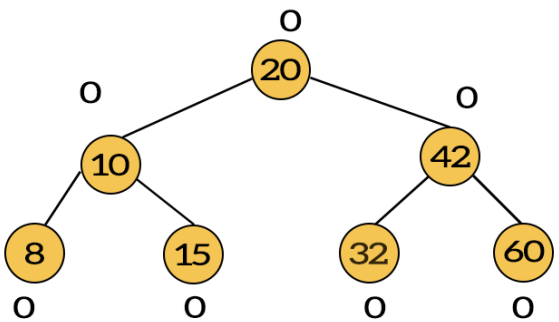


Các phương thức cơ bản

Xóa đi một phần tử - Có hai con

Cân bằng: Xoay phải





Các phương thức cơ bản

Xóa đi một phần tử - Có hai con

```
delete (x,T) {
    if (T has two children)
        if (T.element == x)
            T.element=findMin(T.right) //replace T.element by min of right
            T.right = delete(T.element, T.right)
        else if (x<T.element)
            T.left=delete(x,T.left)
        else
            T.right=delete(x,T.right)
    T.height = getMax(getHeight(T.left), getHeight(T.right)) + 1
    T = rebalance(T)
    return T
}
```

Running time: $O(logn)$

Nội dung

1. Vì sao là cây AVL?
2. Cây AVL
   - Tổng quát
   - Nhân tố cân bằng
   - Phép xoay
   - Tự cân bằng
3. Các phương thức cơ bản
   - Tìm kiếm phần tử nhỏ nhất
   - Tìm kiếm phần tử
   - Thêm một phần tử
   - Xóa đi một phần tử
4. Ứng dụng
5. Tài liệu tham khảo

Ứng dụng

- Tạo ứng dụng trực quan hóa cho cây AVL
- Mô hình quản lý sách áp dụng cấu trúc cây AVL
- Mô hình từ điển áp dụng cấu trúc cây AVL

Nội dung

1. Vì sao là cây AVL?
2. Cây AVL
   - Tổng quát
   - Nhân tố cân bằng
   - Phép xoay
   - Tự cân bằng
3. Các phương thức cơ bản
   - Tìm kiếm phần tử nhỏ nhất
   - Tìm kiếm phần tử
   - Thêm một phần tử
   - Xóa đi một phần tử
4. Ứng dụng
5. Tài liệu tham khảo

Tài liệu tham khảo

- Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser. "Data Structures and Algorithms in Java".
- Geeksforgeeks website. "AVL Tree - Insertion" and "AVL Tree - Deletion".
- Wikipedia website. "AVL tree"
- Techvidvan website. "AVL Tree in Data Structure".