BÀI TẬP CÓ HƯỚNG DẪN

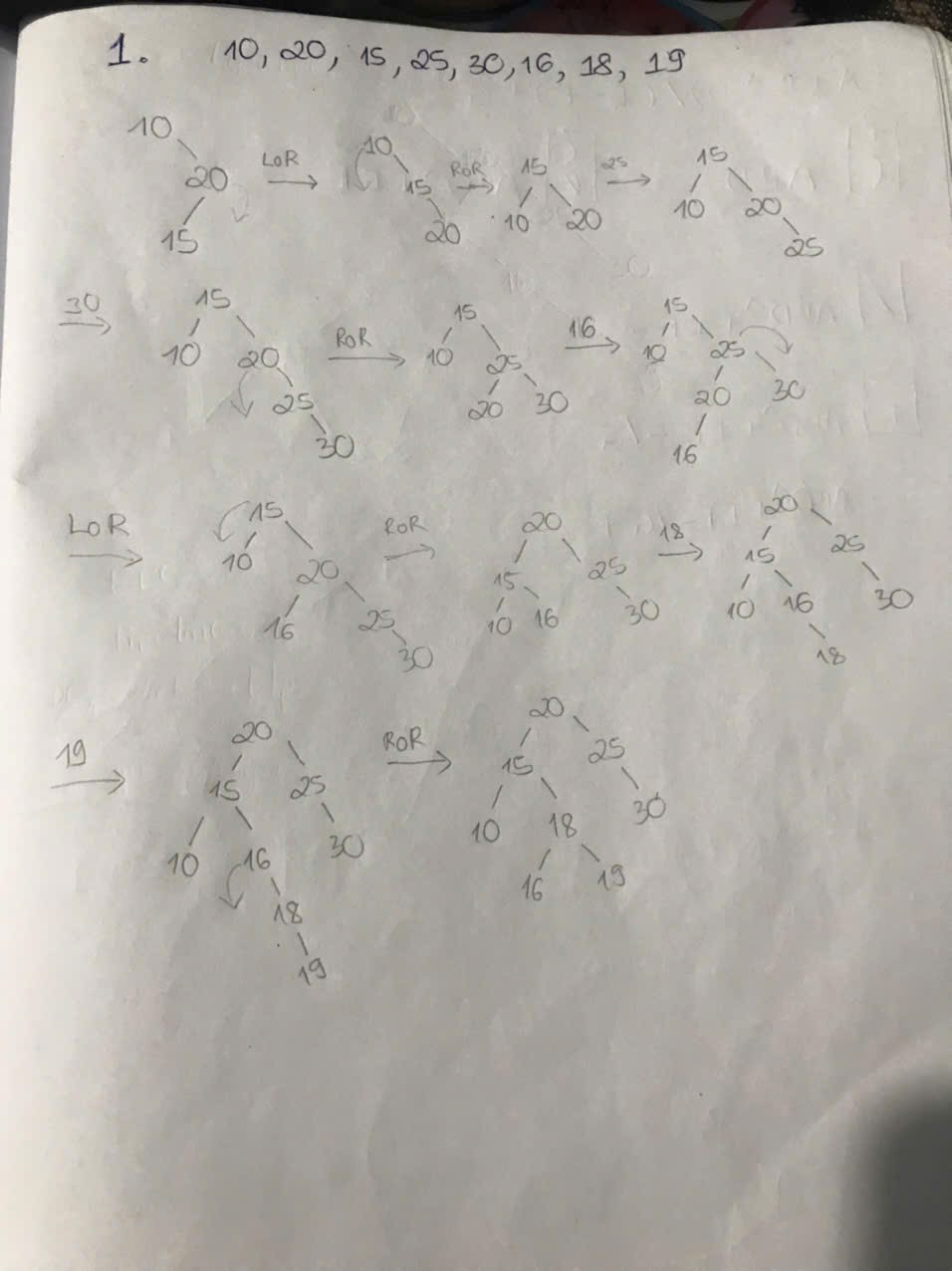
Nội dung: Advanced BST và B-Tree

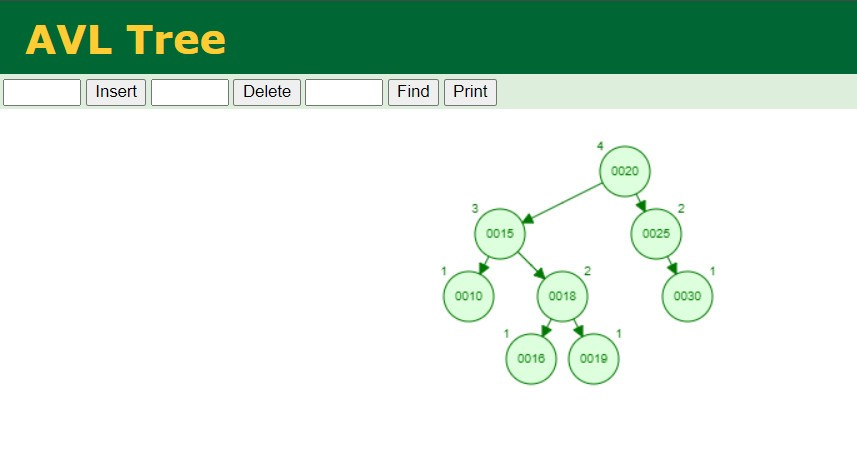
## Bài tập 1

Cho một cây AVL rỗng:

### Lần lượt thêm các giá trị: 10, 20, 15, 25, 30, 16, 18, 19 vào cây trên.

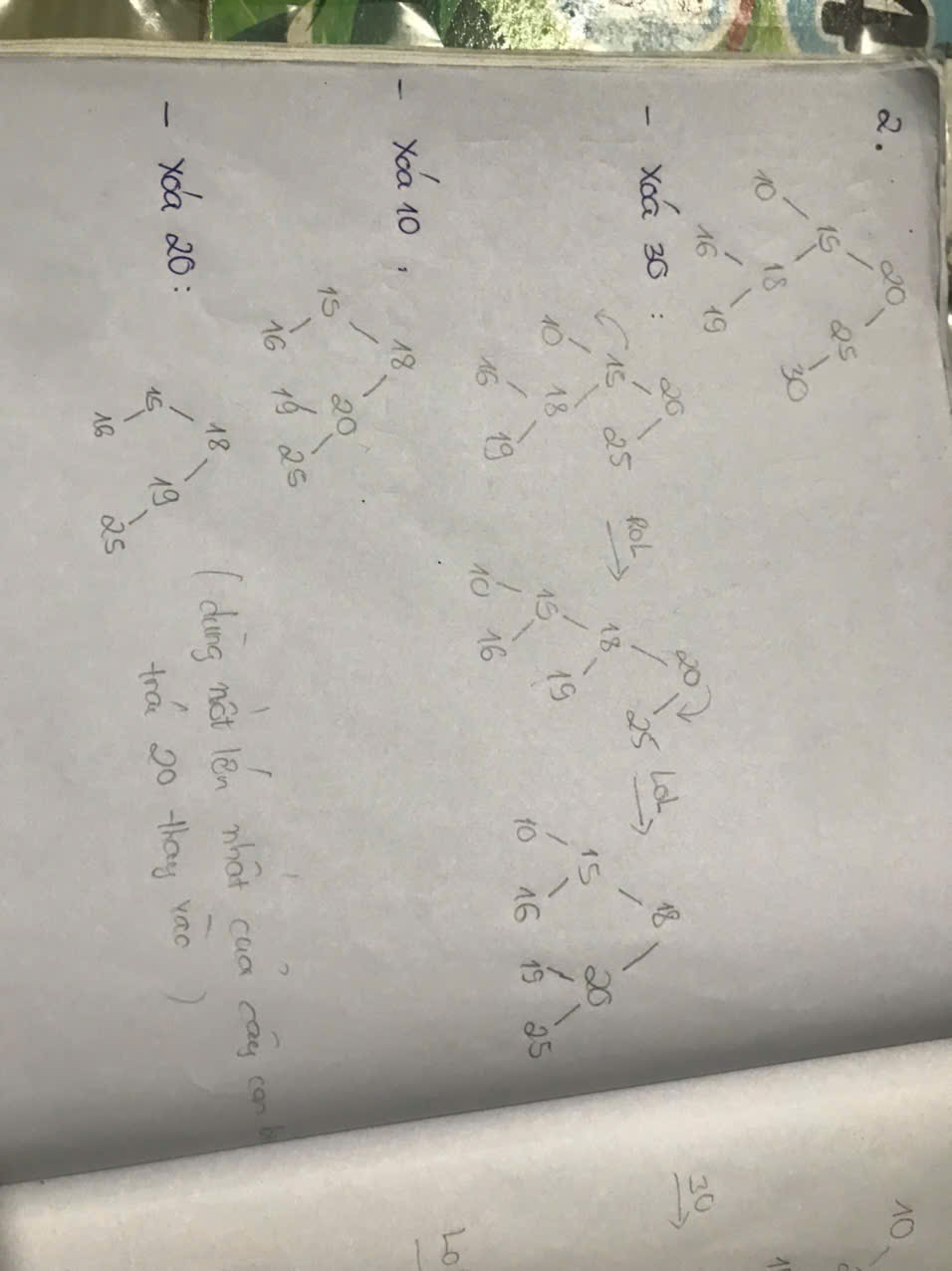
Nguồn: Tony

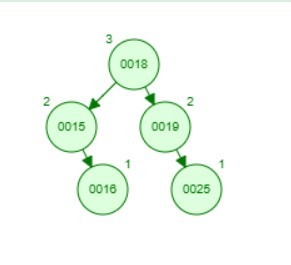




### 2. Lần lượt xóa các số: 30, 10, 20 ra khỏi cây đã tạo thành ở cây số 1.

Nguồn: Tony





### 3. Hãy cho biết, ở cây được tạo thành ở câu số 1. Nếu không muốn thực hiện thao tác tái cân bằng, ta có thể xóa những nút nào?

Hòa: 16,18,19,20

### 4. Hãy sắp xếp lại thứ tự của các giá trị ở câu 1 để việc thêm (lần lượt) các phần tử đó vào cây AVL mà không phải thực hiện thao tác xoay cây?

Hòa: 20,15,25,10,18,30,16,19

### 5. Hãy đề xuất một giải thuật sắp xếp sử dụng cây AVL làm trung gian. Đánh giá độ phức tạp của giải thuật đề xuất về không gian và thời gian.

Tham khảo: chatgpt

Giải thuật sắp xếp sử dụng cây AVL làm trung gian có thể được thực hiện theo các bước sau:

1. **Chèn các phần tử của mảng vào cây AVL**: Cây AVL là một loại cây tìm kiếm nhị phân cân bằng, đảm bảo chiều cao của cây luôn là $O(\log n)$. Khi chèn mỗi phần tử vào cây AVL, thời gian chèn là $O(\log n)$.
2. **Duyệt cây AVL theo thứ tự trung thứ tự (in-order traversal)**: Khi duyệt cây AVL theo thứ tự này, ta sẽ lấy được các phần tử theo thứ tự tăng dần. Thời gian duyệt cây là $O(n)$.

**Đánh giá độ phức tạp**

**Độ phức tạp thời gian**

* **Chèn các phần tử vào cây AVL**: Để chèn $n$ phần tử vào cây AVL, mỗi lần chèn tốn $O(\log n)$. Do đó, tổng thời gian chèn là $O(n \log n)$.
* **Duyệt cây AVL**: Duyệt cây theo thứ tự trung thứ tự tốn $O(n)$.

Tổng hợp lại, độ phức tạp thời gian của giải thuật là $O(n \log n)$.

**Độ phức tạp không gian**

* **Không gian cho cây AVL**: Cây AVL chứa $n$ phần tử, mỗi phần tử cần một nút trong cây. Do đó, không gian cần thiết cho cây AVL là $O(n)$.
* **Bộ nhớ đệ quy cho duyệt cây**: Duyệt cây theo thứ tự trung thứ tự sử dụng bộ nhớ đệ quy với chiều cao tối đa của cây là $O(\log n)$.

Tổng hợp lại, độ phức tạp không gian của giải thuật là $O(n)$.

**Mã giả của giải thuật**

text

Copy

function AVLTreeSort(array):

AVLTree tree = new AVLTree()

// Chèn tất cả phần tử vào cây AVL

for element in array:

tree.insert(element)

sortedArray = []

// Duyệt cây AVL theo thứ tự trung thứ tự

inOrderTraversal(tree.root, sortedArray)

return sortedArray

function inOrderTraversal(node, sortedArray):

if node is not null:

inOrderTraversal(node.left, sortedArray)

sortedArray.append(node.value)

inOrderTraversal(node.right, sortedArray)

**Kết luận**

Giải thuật sắp xếp sử dụng cây AVL làm trung gian có độ phức tạp thời gian là $O(n \log n)$ và độ phức tạp không gian là $O(n)$. Đây là một phương pháp hiệu quả và đáng tin cậy để sắp xếp các phần tử, đặc biệt khi cần duy trì cân bằng của cây để đảm bảo hiệu suất tốt.