BST:
- contains
- insert: \( O(n) \) worst
- remove: \( O(\log n) \) average

If we could guarantee depth is always \( O(\log n) \),
could get worst-case \( O(\log n) \) times.

Balanced Binary Search Tree (AVL Trees) -
BST with worst-case depth of \( O(\log n) \)

Specifically: for every node \( n \), heights of left and
right children of \( n \) differ by at most 1

Ex: AVL Tree  Not AVL Tree (Why not? 6)

Operations are same as in BST:
If after operation, tree becomes imbalanced, tree is
rebalanced to have depth \( O(\log n) \)
Since depth is worst-case \( O(\log n) \), all operations take \( O(\log n) \) in the worst-case.

\[
\begin{align*}
\text{Time} &= O(\text{depth}) = O(\log n) \quad \text{(worst \& avg)} \\
\text{insert} & \quad \text{remove}
\end{align*}
\]

How to rebalance?

**Example: insert e**

\[
\text{before insert} \\
\begin{array}{c}
(a) \\
(b) \\
(c) \\
(d)
\end{array}
\]

\[
\text{after insert} \\
\begin{array}{c}
(a) \\
(b) \\
(c) \\
(d) \\
(e)
\end{array}
\]

**Step 1:** Identify \( \alpha = \) deepest imbalanced node

- deepest node whose left + right subtrees have height difference \( > 1 \)

**Step 2:** Identify type of insert.

4 Types of inserts

1. insert into right subtree of \( \alpha \)'s right child \( \text{"outer" insert} \)

2. "left " \( \text{left } \)

(Above example was "outer" insert)

3. "left " \( \text{right } \)

(\text{inner } \text{insert})

4. "right " \( \text{left } \)
"inner" example

1. Insert into left subtree of α's right child.

Step 3: Perform swap(s) to rebalance

For outer: Single swap
(1) Swap α + α's child

For inner: Double swap
(1) Swap α's child + grandchild
(2) Swap α + α's new child.

ex 1D:

```
    20
   /  
  10   30
 /     /     
5      25   35
       /     /
      33    40
```

AVL
insert (45)

Step 4: α? = 30
Step 2: outer insert (single swap).
Step 3: swap $30 + 35$

insert(45)

3 pointer changes:
- $a$'s parent (right ptr) $\rightarrow a$'s child $20 \rightarrow 35$
- $a$'s right child (left ptr) $\rightarrow a$ $35 \rightarrow 30$
- $a$'s (right ptr) $\rightarrow a$'s successor $30 \rightarrow 33$

left/right switched for other side

ex.2 insert(4)

swap $a$ (10) + $a$'s child (5)

- $a$'s parent (left ptr) $\rightarrow a$'s child $20 \rightarrow 5$
- $a$'s left child (right ptr) $\rightarrow a$ $5 \rightarrow 10$
- $a$'s new (left ptr) $\rightarrow a$'s successor $10 \rightarrow$ null
AVL? ✓

\text{insert(34)}

AVL? No. α = 20

Which type of insert? inner:
→ left subtree of α's right child.

double swap: - swap α's child and grandchild (on side of imbalance)
            - swap α and α's new child

After 1st swap:

swap 35 \leftrightarrow 30

3 ptr-changes: <from original tree>  
α's (right) \rightarrow α's right child's left child \quad 20 \rightarrow 30  
α's grandchild (right ptr) \rightarrow α's child \quad 30 \rightarrow 35  
α's child (left ptr) \rightarrow it's (prev) predecessor \quad 35 \rightarrow 33
After 2nd swap

Swap 20 + 30

3 ptr changes

- a's parent's (right ptr) → a's child  root → 30
- a's child's (left ptr) → a  30 → 20
- a's (right ptr) → a's (prev) successor  20 → 25

Another example

Type of insert? inner
(right subtree of a's left child)
Double swap:

(i) Swap a's child and grandchild.

(ii) Swap a and a's new child.

![Diagram of a tree structure with nodes labeled 5, 7, 9, 12, 13, 15, 16, and 14. The tree is rooted at the top node 14. The network of children and grandchildren is shown, with 14 being the root, 15 as one of its children, and 12, 13, 16 as grandchildren of 15.]