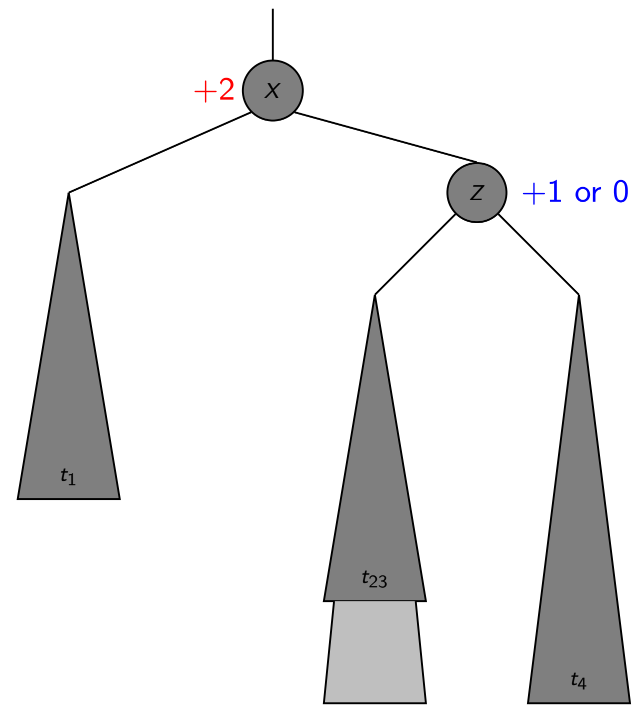
AVL Tree: Insertion and Deletion

Subhabrata Samajder

IIIT, Delhi
Winter Semester,
14^{th} April, 2023

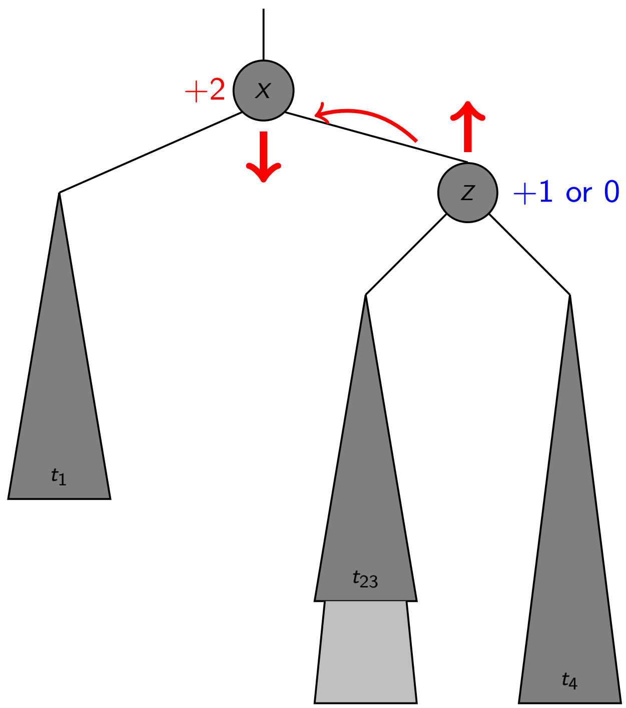
AVL Trees

The Balancing Act: Single Rotation (Recap)



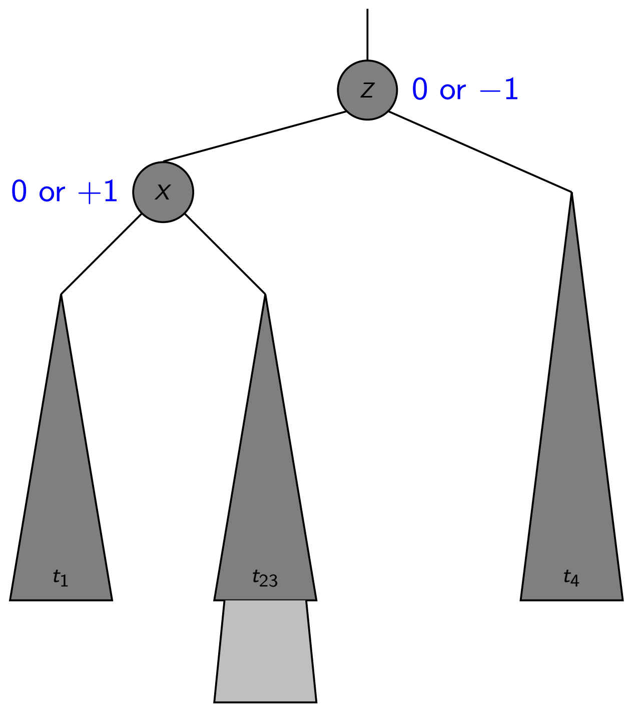
- Node X has two child trees with a balance factor of $+2$.
- The left child t_{23} of Z is not higher than its sibling t_4.
  - Can happen by a height increase of t_4 or by a height decrease of t_1.
- **Note:** t_{23} can have the same height as t_4.
- The mirror case is easily derived.

The Balancing Act: Single Rotation (Recap)



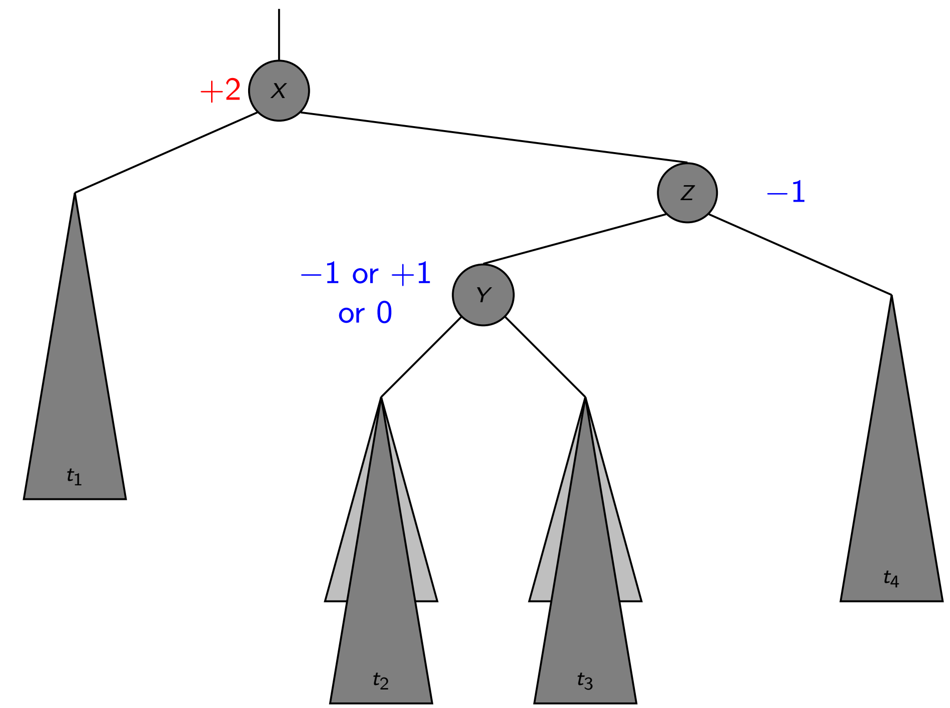
- Node X has two child trees with a balance factor of $+2$.
- The left child t_{23} of Z is not higher than its sibling t_4.
    - Can happen by a height increase of t_4 or by a height decrease of t_1.
- **Note:** t_{23} can have the same height as t_4.
- The mirror case is easily derived.

The Balancing Act: Single Rotation (Recap)



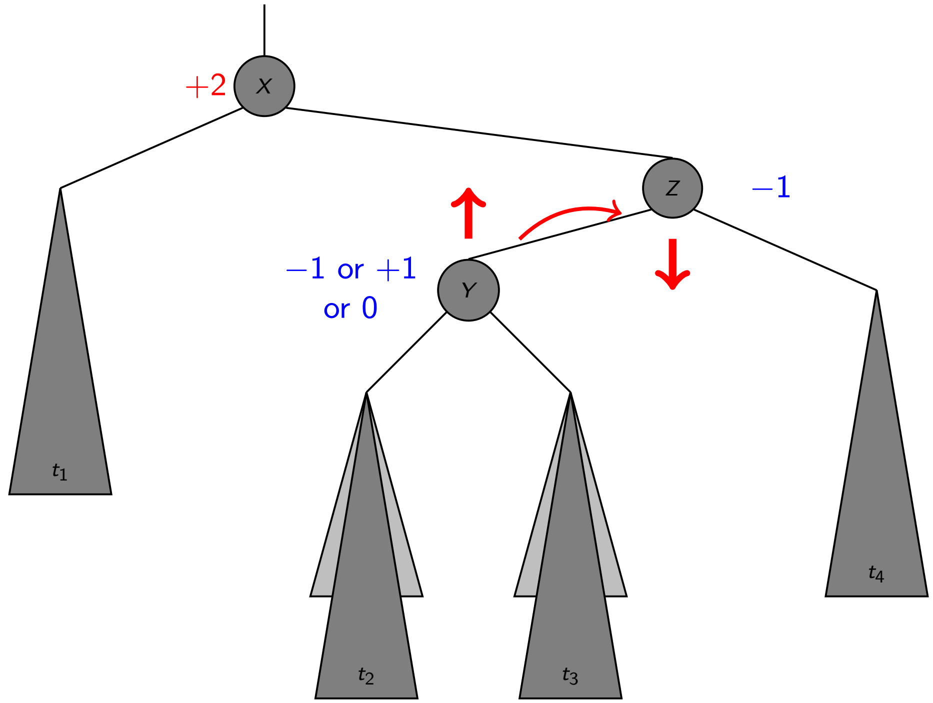
- Node X has two child trees with a balance factor of $+2$.
- The left child t_{23} of Z is not higher than its sibling t_4.
  - Can happen by a height increase of t_4 or by a height decrease of t_1.
- **Note:** t_{23} can have the same height as t_4.
- The mirror case is easily derived.

The Balancing Act: Double Rotation (Recap)



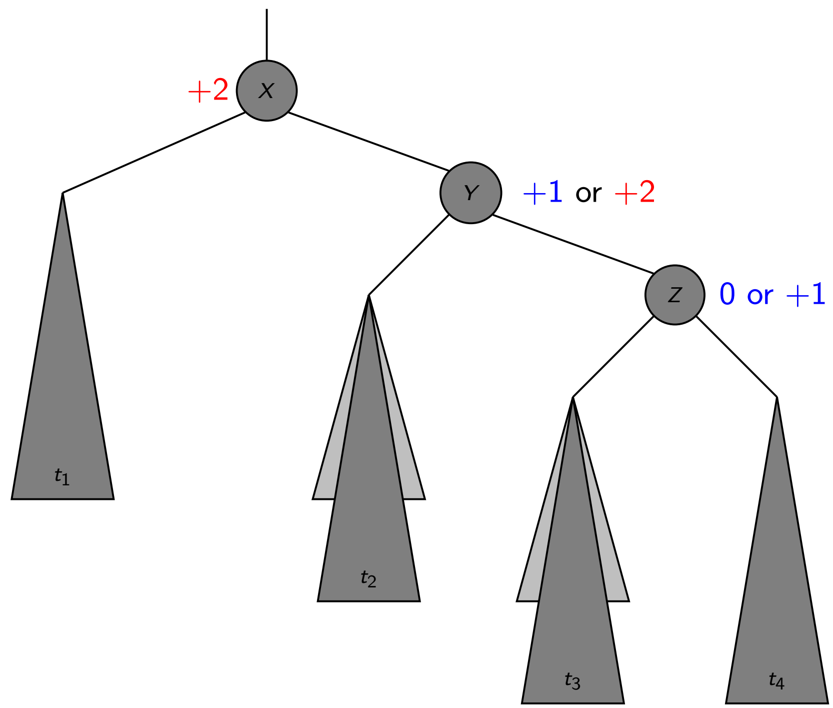
- Node X has two child trees with a balance factor of $+2$.
- The left child Y of Z is higher than its sibling t_4.
  - Can happen by the insertion of Y itself or a height increase of one of its subtrees t_2 or t_3 or by a height decrease of subtree t_1.
- **Note:** t_2 and t_3 may also be of same height.
- The mirror case is easily derived.

The Balancing Act: Double Rotation (Recap)



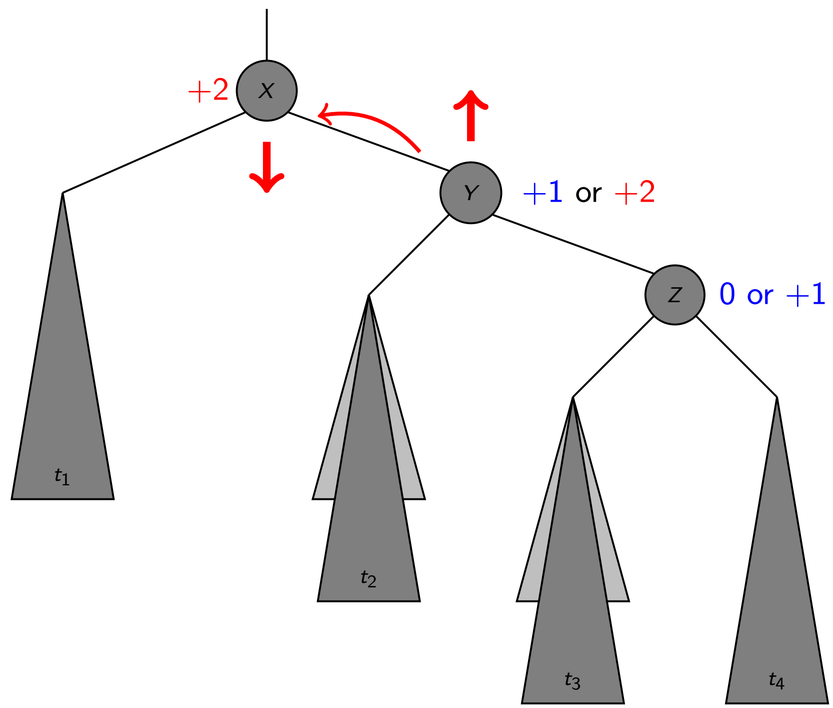
- Node X has two child trees with a balance factor of $+2$.
- The left child Y of Z is higher than its sibling t_4.
  - Can happen by the insertion of Y itself or a height increase of one of its subtrees t_2 or t_3 or by a height decrease of subtree t_1.
- **Note:** t_2 and t_3 may also be of same height.
- The mirror case is easily derived.

The Balancing Act: Double Rotation (Recap)



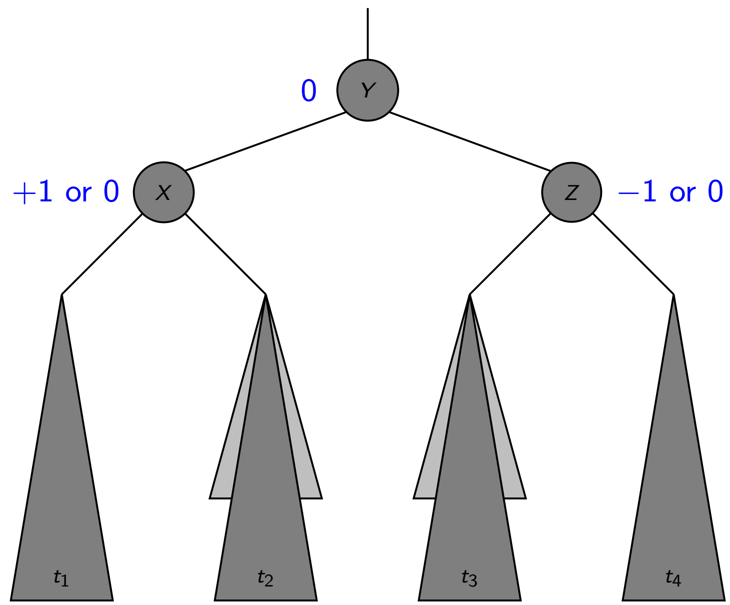
- Node X has two child trees with a balance factor of $+2$.
- The left child Y of Z is higher than its sibling t_4.
  - Can happen by the insertion of Y itself or a height increase of one of its subtrees t_2 or t_3 or by a height decrease of subtree t_1.
- **Note:** t_2 and t_3 may also be of same height.
- The mirror case is easily derived.

The Balancing Act: Double Rotation (Recap)



- Node X has two child trees with a balance factor of $+2$.
- The left child Y of Z is higher than its sibling t_4.
  - Can happen by the insertion of Y itself or a height increase of one of its subtrees t_2 or t_3 or by a height decrease of subtree t_1.
- **Note:** t_2 and t_3 may also be of same height.
- The mirror case is easily derived.

The Balancing Act: Double Rotation (Recap)



- Node X has two child trees with a balance factor of $+2$.
- The left child Y of Z is higher than its sibling t_4.
  - Can happen by the insertion of Y itself or a height increase of one of its subtrees t_2 or t_3 or by a height decrease of subtree t_1.
- **Note:** t_2 and t_3 may also be of same height.
- The mirror case is easily derived.

Insertion in an AVL Tree

Rotations

- When the tree structure changes (e.g., insertion or deletion), we need to transform the tree to restore the AVL tree property.
- This is done using single rotations or double rotations.
- An insertion/deletion involves adding/deleting a single node.
- This may increase/decrease the height of some subtree by 1.
- Thus, if the AVL tree property is violated at a node x, it means that the heights of $left[x]$ and $right[x]$ differ by exactly 2.
- Rotations are applied to x to restore the AVL tree property.

Insertion

- Insert the new node as leaf like ordinary BST.
- Trace the path from the new leaf towards the root.
- For each node x encountered, check if heights of $left[x]$ and $right[x]$ differ by at most 1.
  - If yes, proceed to the $parent[x]$.
  - If not, restructure by doing either a single or a double rotation.
- **Note:** Once a rotation at a node x is performed, no further rotation is needed for any ancestor of x.

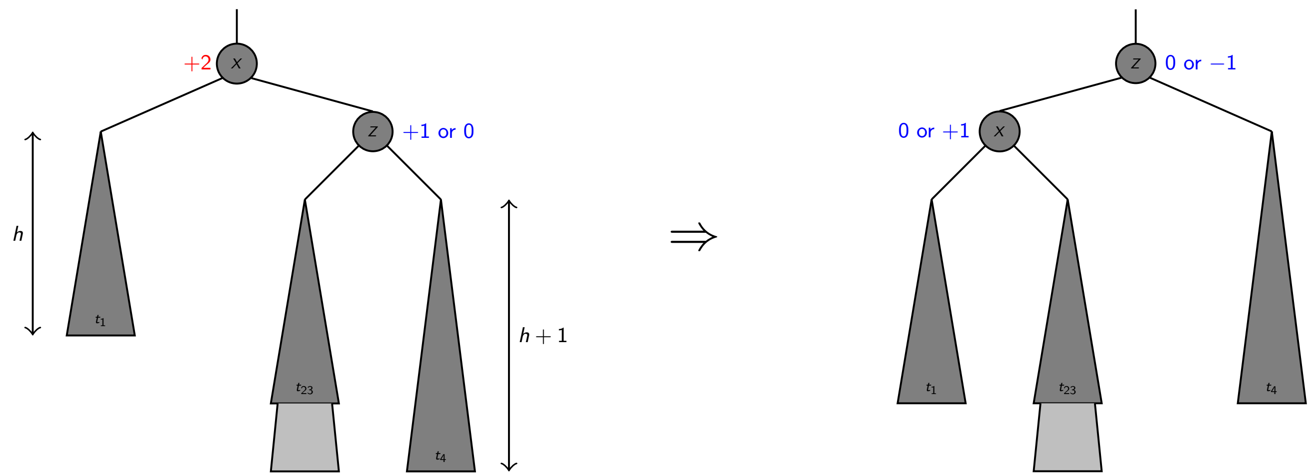
Insertion (Cont.)

- Let x be the node at which $left[x]$ and $right[x]$ differ by more than 1.

- Assume that the height of x is $h + 3$.

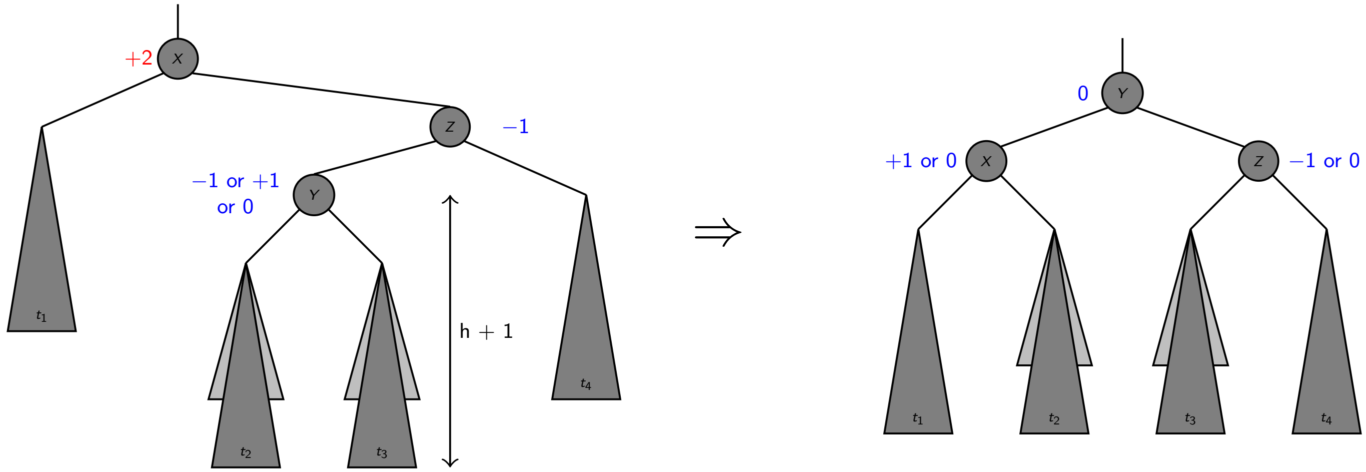
- Four cases may arise:
  - $Height[left[x]] = h + 2$ and $Height[right[x]] = h$.
    - $Height[left[left[x]]] = h + 1 \Rightarrow$ single rotate with left child.
    - $Height[right[left[x]]] = h + 1 \Rightarrow$ double rotate with left child.
  - $Height[right[x]] = h + 2$ and $Height[left[x]] = h$.
    - $Height[right[right[x]]] = h+1 \Rightarrow$ single rotate with right child.
    - $Height[left[right[x]]] = h+1 \Rightarrow$ double rotate with right child.

Single Rotation



- The new key is inserted in the subtree t_4.
- The AVL-property is violated at x:
  - height of x is $h + 3$.
  - height of $right[x]$ is $h + 2$ and height of $left[x]$ is h.
  - height of $right[right[x]] = h + 1$.
- Rotate with right child.
- **Mirror condition:** Rotate with left child.
- Single rotation takes $\mathcal{O}(1)$ time.
- **Worst-case complexity:** $\mathcal{O}(\log N)$.

Double Rotation



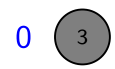
- The new key is inserted in the subtree t_2/t_3.
- The AVL-property is violated at x:
  - height of x is $h+3$.
  - height of $right[x]$ is $h+2$ and height of $left[x]$ is h.
  - height of $left[right[x]] = h+1$.
- Execute a right-left rotate.
- Similarly execute a left-right rotate for the mirror condition.
- Single rotation takes $\mathcal{O}(1)$ time.
- **Worst-case complexity:** $\mathcal{O}(\log N)$.

An Extended Example

Insert: 3, 2, 1, 4, 5, 6, 7, 16, 15, 14,

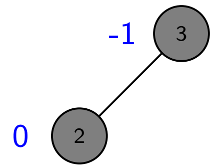
An Extended Example

Insert: 3, 2, 1, 4, 5, 6, 7, 16, 15, 14,



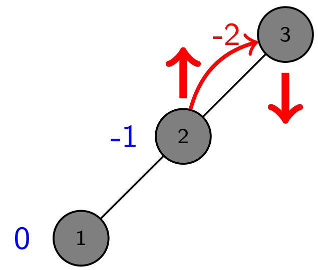
An Extended Example

Insert: 3, 2, 1, 4, 5, 6, 7, 16, 15, 14,



An Extended Example

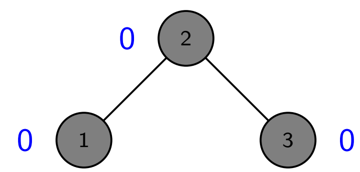
Insert: 3, 2, 1, 4, 5, 6, 7, 16, 15, 14,



Single Rotation

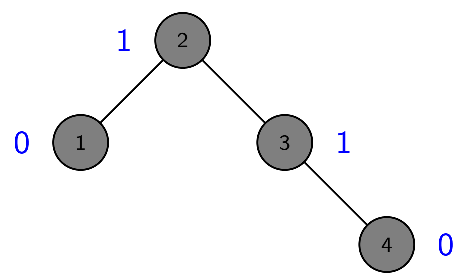
An Extended Example

Insert: 3, 2, 1, 4, 5, 6, 7, 16, 15, 14,



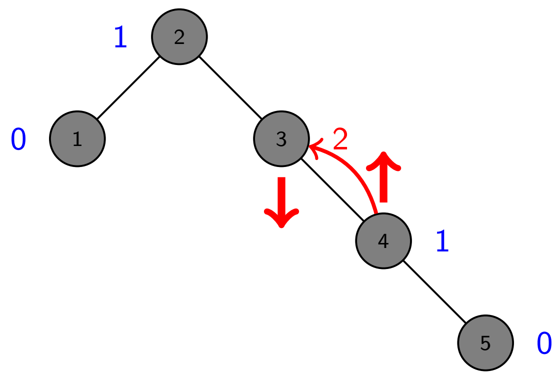
An Extended Example

Insert: 3, 2, 1, 4, 5, 6, 7, 16, 15, 14,



An Extended Example

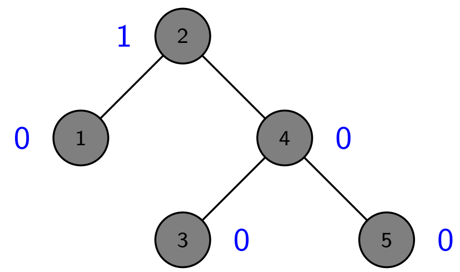
Insert: 3, 2, 1, 4, 5, 6, 7, 16, 15, 14,



Single Rotation

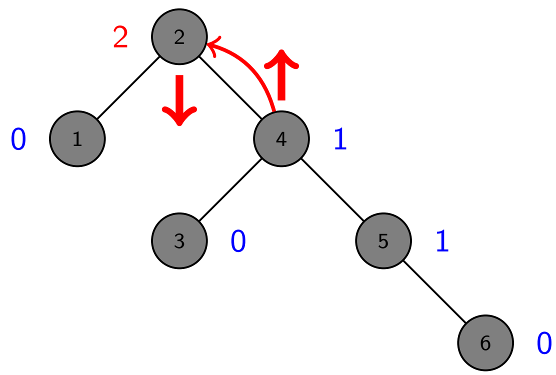
An Extended Example

Insert: 3, 2, 1, 4, 5, 6, 7, 16, 15, 14,



An Extended Example

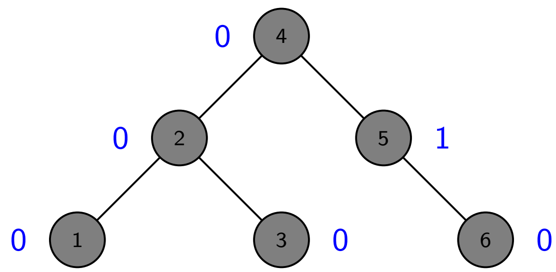
Insert: 3, 2, 1, 4, 5, 6, 7, 16, 15, 14,



Single Rotation

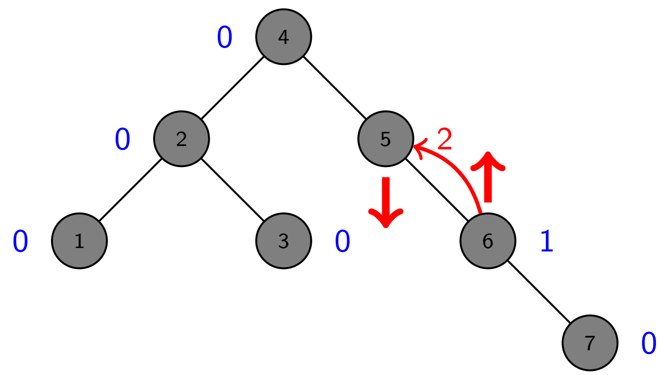
An Extended Example

Insert: 3, 2, 1, 4, 5, 6, 7, 16, 15, 14,



An Extended Example

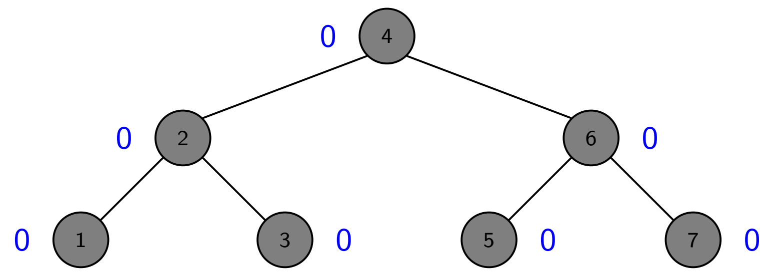
Insert: 3, 2, 1, 4, 5, 6, 7, 16, 15, 14,



Single Rotation

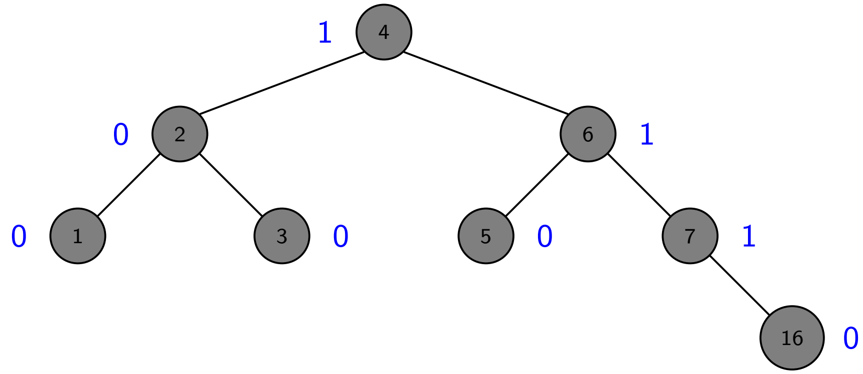
An Extended Example

Insert: 3, 2, 1, 4, 5, 6, 7, 16, 15, 14,



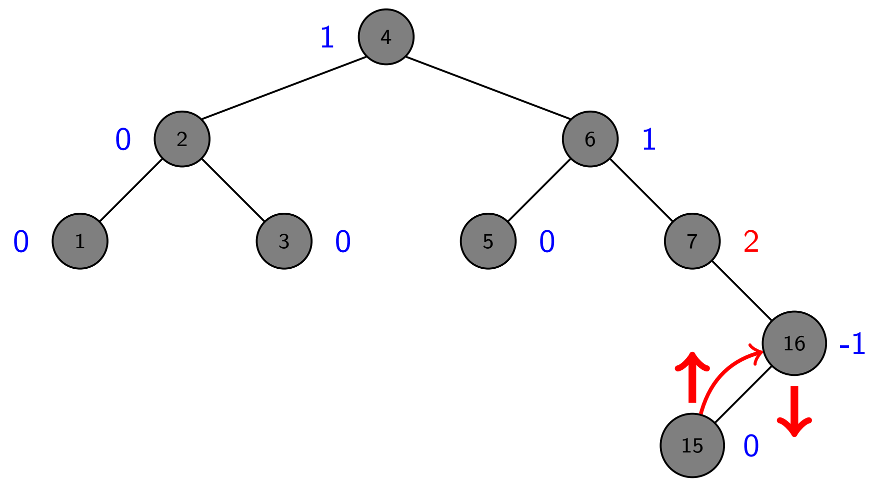
An Extended Example

Insert: 3, 2, 1, 4, 5, 6, 7, 16, 15, 14,



An Extended Example

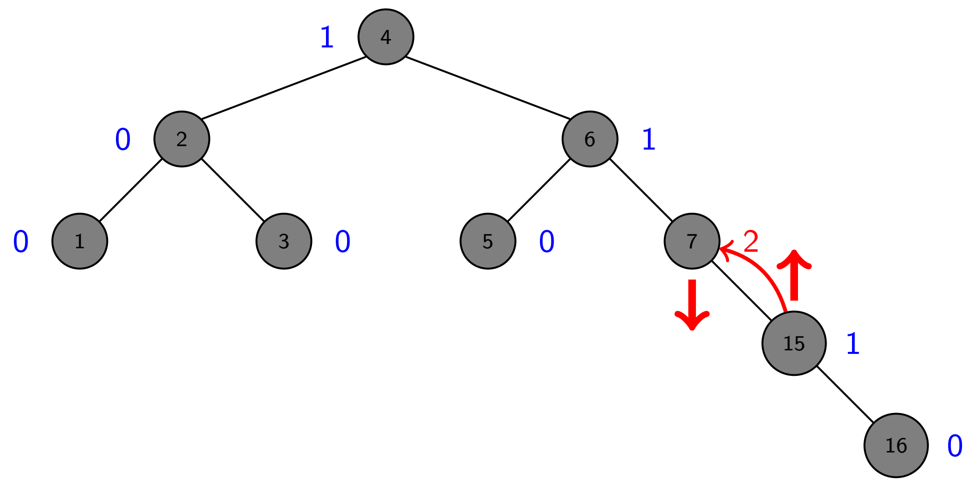
Insert: 3, 2, 1, 4, 5, 6, 7, 16, 15, 14,



Double Rotation

An Extended Example

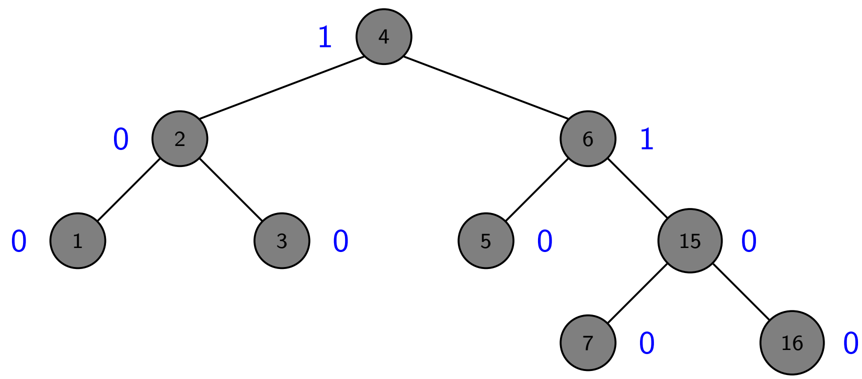
Insert: 3, 2, 1, 4, 5, 6, 7, 16, 15, 14,



Double Rotation

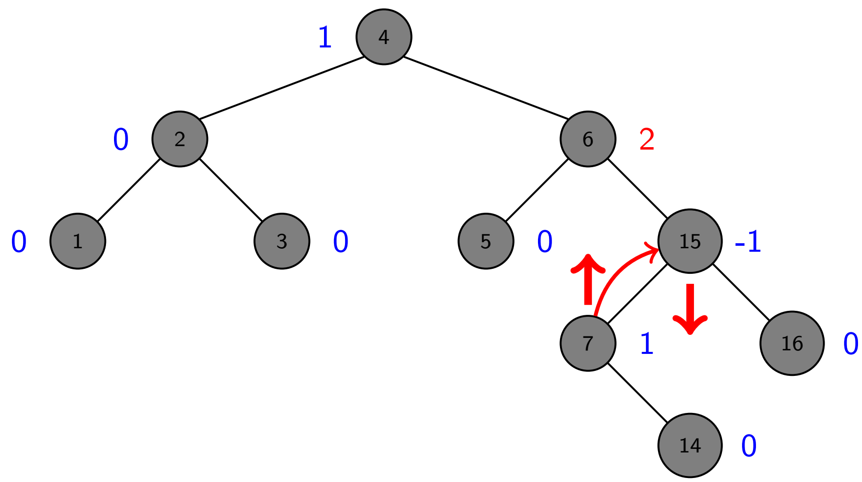
An Extended Example

Insert: 3, 2, 1, 4, 5, 6, 7, 16, 15, 14,



An Extended Example

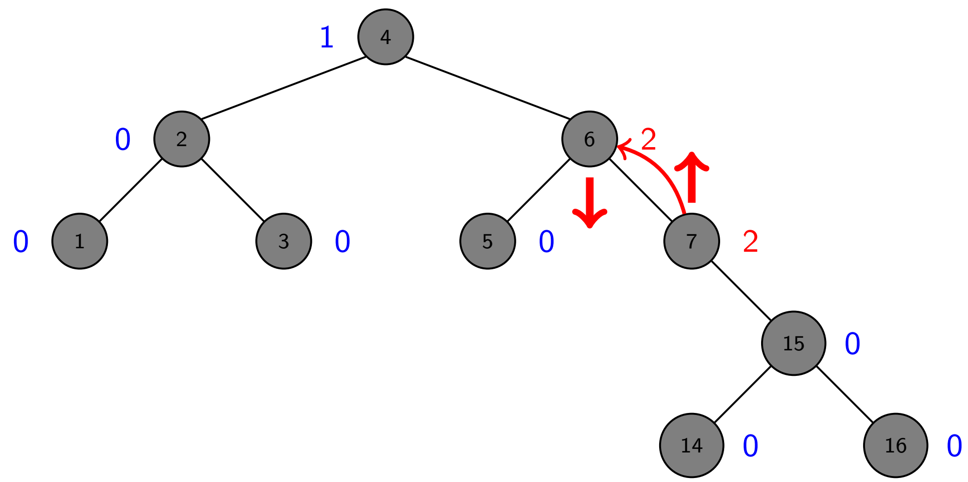
Insert: 3, 2, 1, 4, 5, 6, 7, 16, 15, 14,



Double Rotation

An Extended Example

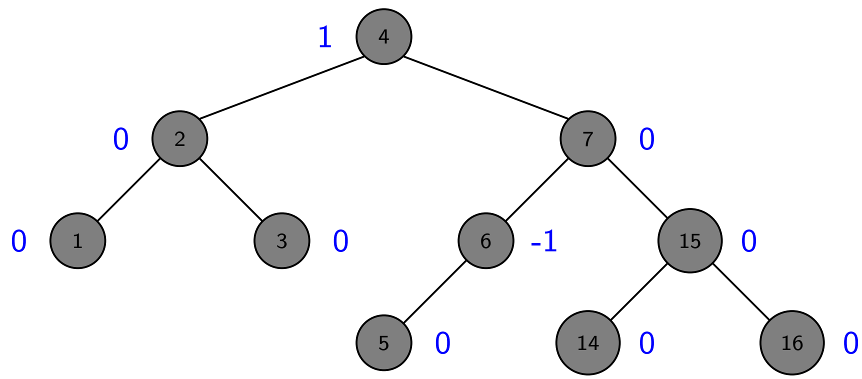
Insert: 3, 2, 1, 4, 5, 6, 7, 16, 15, 14,



Double Rotation

An Extended Example

Insert: 3, 2, 1, 4, 5, 6, 7, 16, 15, 14,



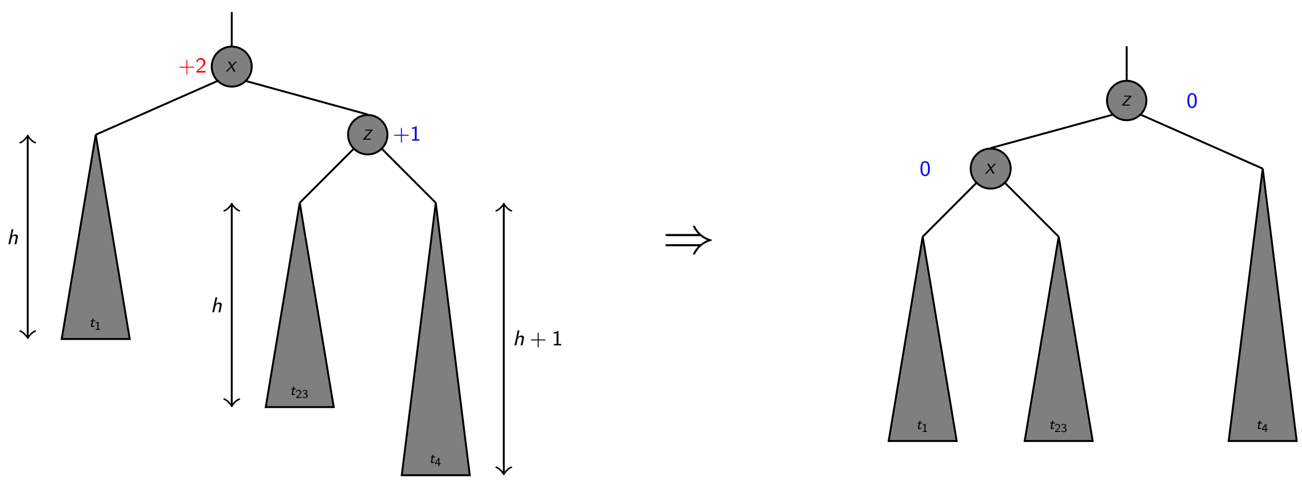
Deletion in an AVL Tree

Deletion

- Delete a node x as in ordinary BST.
- Trace the path from the deleted node towards the root.
- For each node x encountered, check if heights of $left[x]$ and $right[x]$ differ by at most 1.
  - If yes, proceed to $parent[x]$.
  - If not, perform an appropriate rotation at x.
- There are 4 cases as in the case of insertion.
- **Note:** For deletion, after we perform a rotation at x, we may have to perform a rotation at some ancestor of x.
- $\therefore$ continue to trace the path until we reach the root.

Single Rotations in Deletion

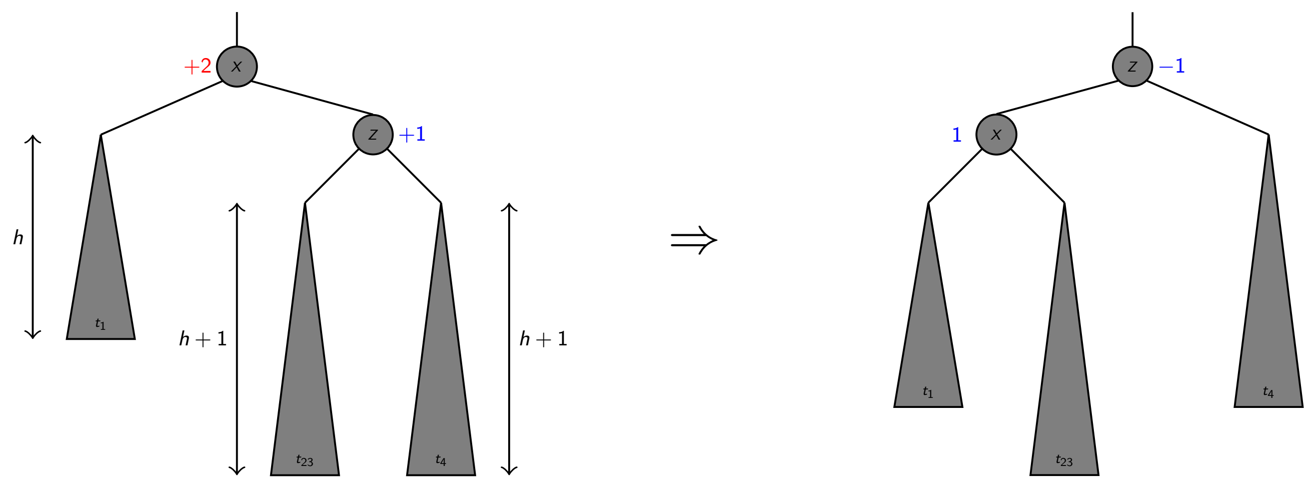
- A node is deleted in t_1, causing the height to drop to h.
- The height of z is $h + 2$.
- The height of t_4 is $h + 1$; the height of t_{23} can be h or $(h + 1)$.
- A single rotation can correct both cases.
- The mirror case is handled similarly.



Rotate with Right Child

Single Rotations in Deletion

- A node is deleted in t_1, causing the height to drop to h.
- The height of z is $h+2$.
- The height of t_4 is $h+1$; the height of t_{23} can be h or $(h+1)$.
- A single rotation can correct both cases.
- The mirror case is handled similarly.



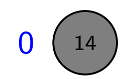
Rotate with Right Child

Rotations in Deletion

- There are 4 cases for single rotations, but we do not need to distinguish among them.
- There are exactly two cases for double rotations (as in the case of insertion).
- Therefore, we can reuse exactly the same procedure for insertion to determine which rotation to perform.

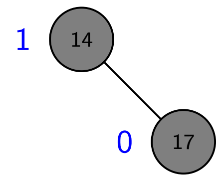
AVL Tree Example

Insert: 14



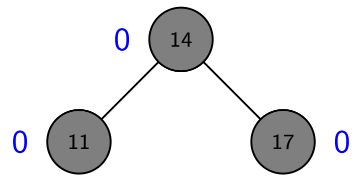
AVL Tree Example

Insert: 14, 17



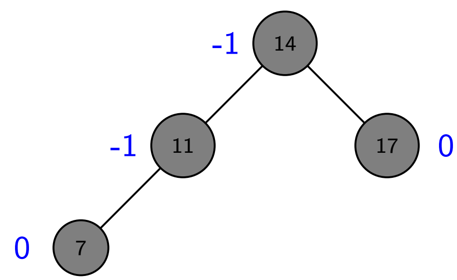
AVL Tree Example

Insert: 14, 17, 11



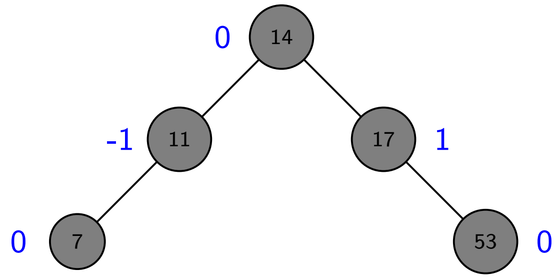
AVL Tree Example

Insert: 14, 17, 11, 7



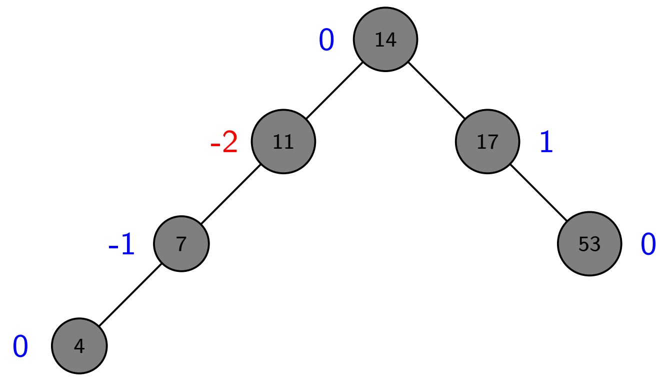
AVL Tree Example

Insert: 14, 17, 11, 7, 53



AVL Tree Example

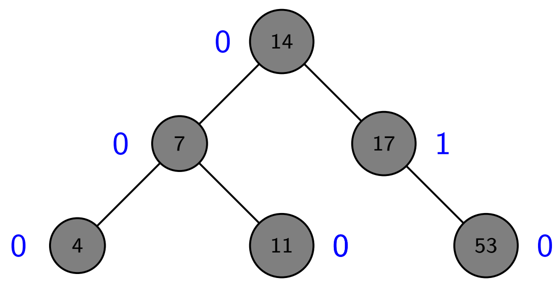
Insert: 14, 17, 11, 7, 53, 4



Single Rotation

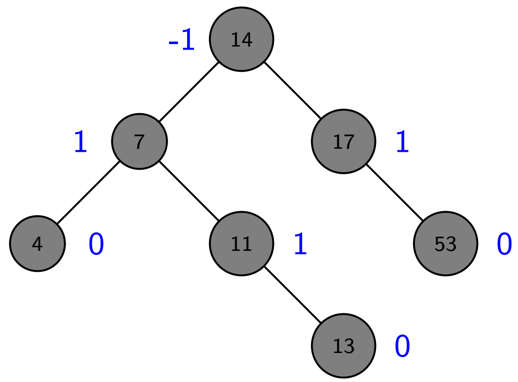
AVL Tree Example

Insert: 14, 17, 11, 7, 53, 4



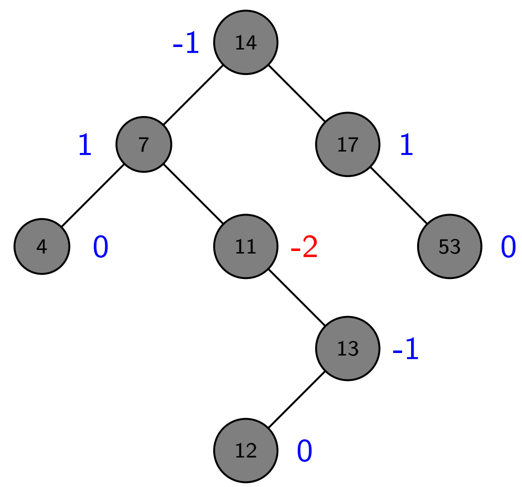
AVL Tree Example

Insert: 14, 17, 11, 7, 53, 4, 13



AVL Tree Example

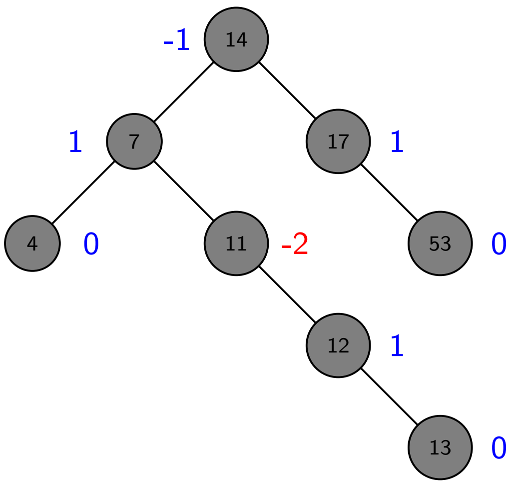
Insert: 14, 17, 11, 7, 53, 4, 13, 12



Double Rotation

AVL Tree Example

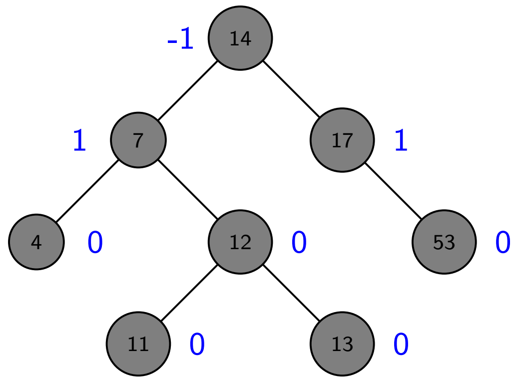
Insert: 14, 17, 11, 7, 53, 4, 13, 12



Double Rotation

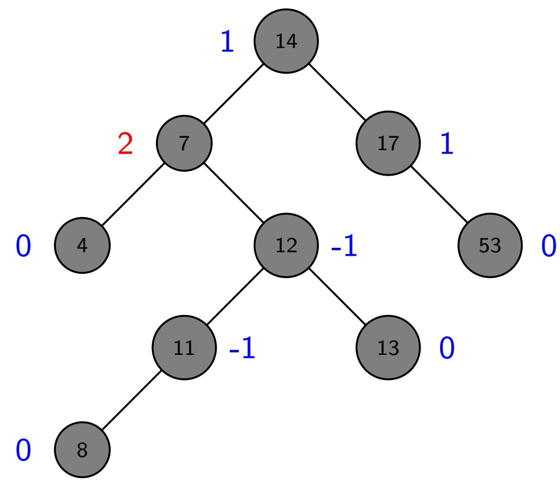
AVL Tree Example

Insert: 14, 17, 11, 7, 53, 4, 13, 12



AVL Tree Example

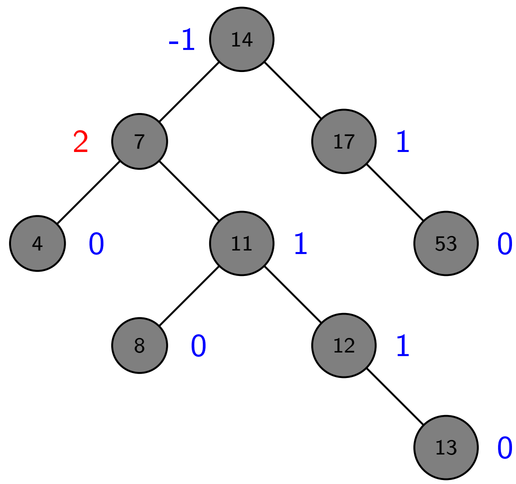
Insert: 14, 17, 11, 7, 53, 4, 13, 12, 8



Double Rotation

AVL Tree Example

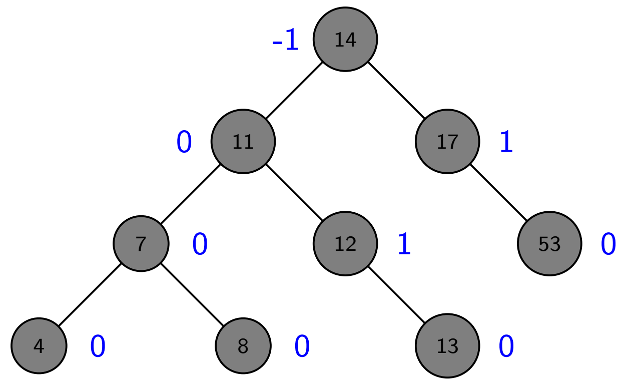
Insert: 14, 17, 11, 7, 53, 4, 13, 12, 8



Double Rotation

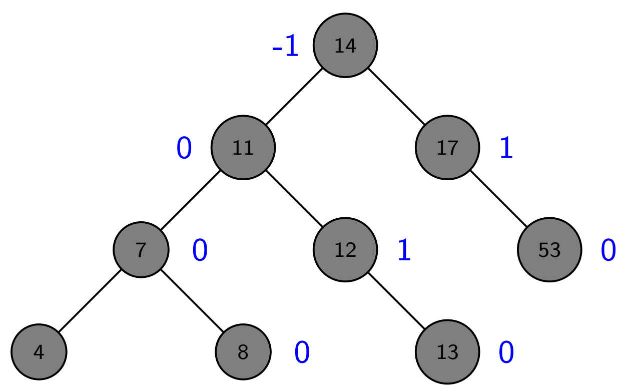
AVL Tree Example

Insert: 14, 17, 11, 7, 53, 4, 13, 12, 8



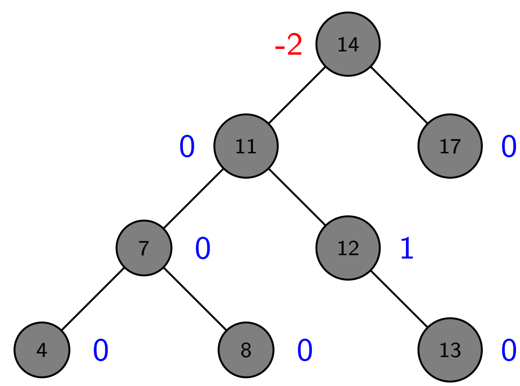
AVL Tree Example

Insert: 14, 17, 11, 7, 53, 4, 13, 12, 8
Delete: 53



AVL Tree Example

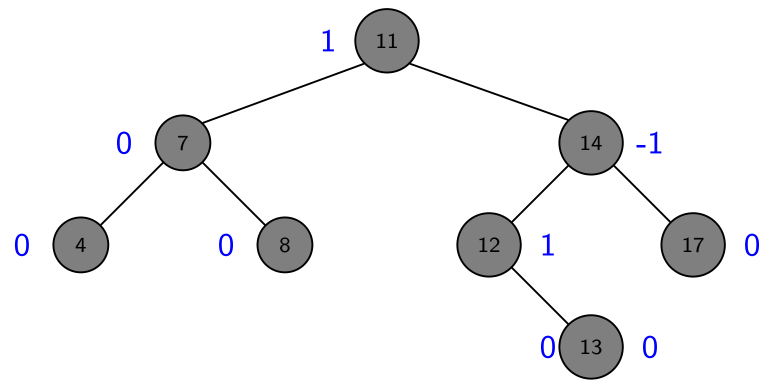
Insert: 14, 17, 11, 7, 53, 4, 13, 12, 8
Delete: 53



Single Rotation

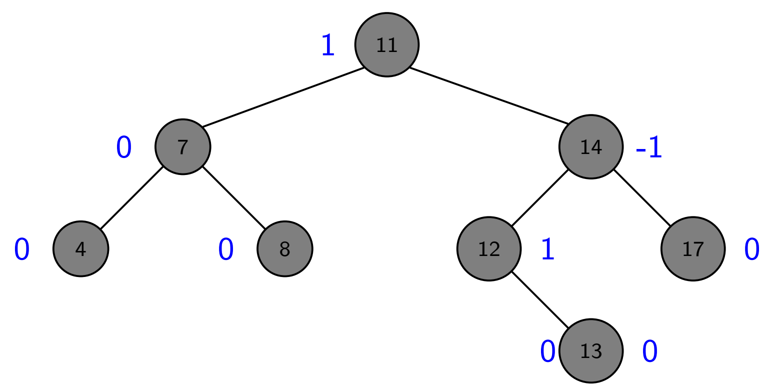
AVL Tree Example

Insert: 14, 17, 11, 7, 53, 4, 13, 12, 8
Delete: 53



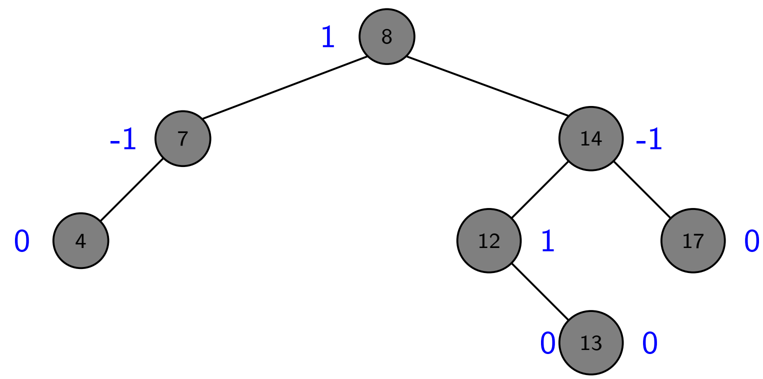
AVL Tree Example

Insert: 14, 17, 11, 7, 53, 4, 13, 12, 8
Delete: 53, 11



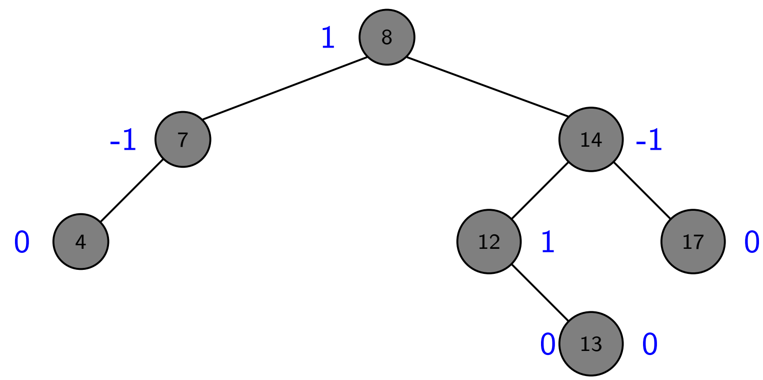
AVL Tree Example

Insert: 14, 17, 11, 7, 53, 4, 13, 12, 8
Delete: 53, 11 (Replace 11 by its pre-decessor!!)



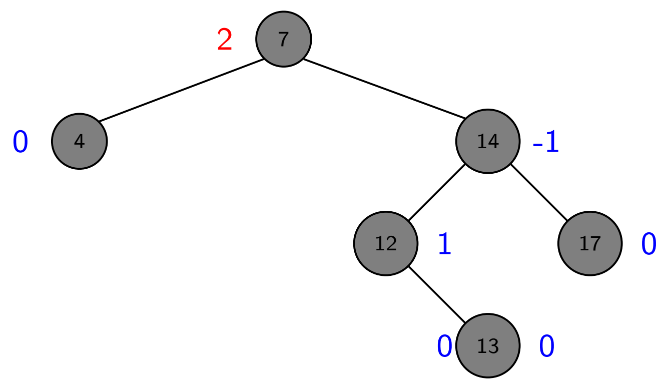
AVL Tree Example

Insert: 14, 17, 11, 7, 53, 4, 13, 12, 8
Delete: 53, 11, 8



AVL Tree Example

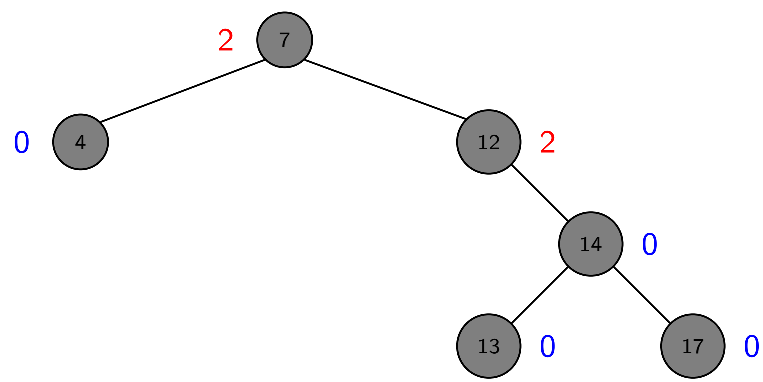
Insert: 14, 17, 11, 7, 53, 4, 13, 12, 8
Delete: 53, 11, 8 (Replace 8 by its pre-decessor!!)



Double Rotation

AVL Tree Example

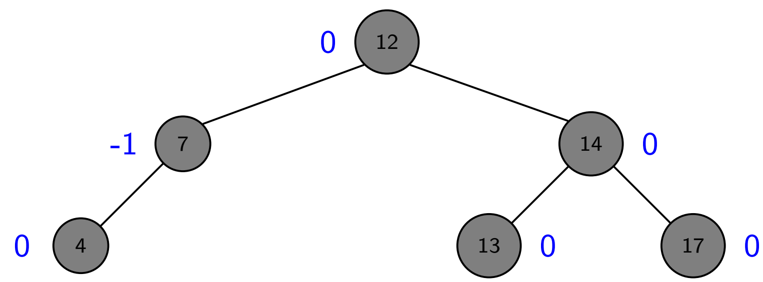
Insert: 14, 17, 11, 7, 53, 4, 13, 12, 8
Delete: 53, 11, 8



Double Rotation

AVL Tree Example

Insert: 14, 17, 11, 7, 53, 4, 13, 12, 8
Delete: 53, 11, 8



Class Exercises

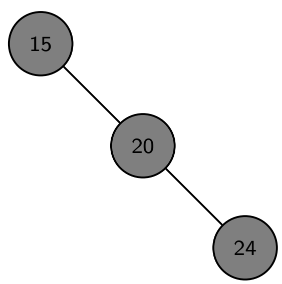
- Build an AVL tree with the following values:

$$15, 20, 24, 10, 13, 7, 30, 36, 25$$

Class Exercises

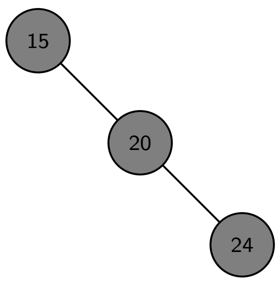
- Build an AVL tree with the following values:

$$15, 20, 24, 10, 13, 7, 30, 36, 25$$

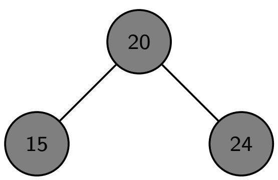


- Build an AVL tree with the following values:

15, 20, 24, 10, 13, 7, 30, 36, 25



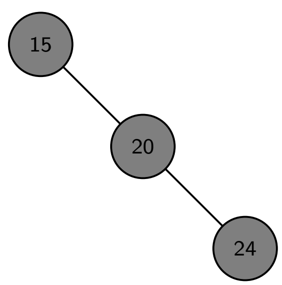
$\Rightarrow$



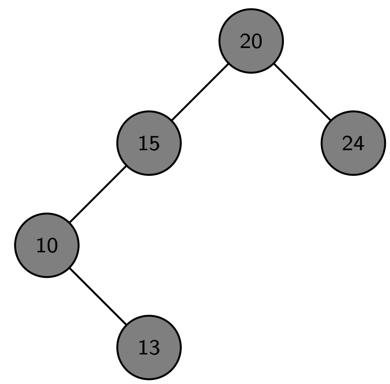
Class Exercises

- Build an AVL tree with the following values:

$$15, 20, 24, 10, 13, 7, 30, 36, 25$$



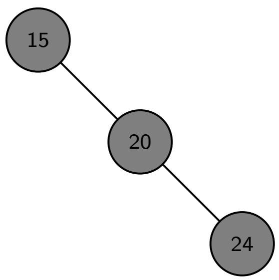
$\Rightarrow$

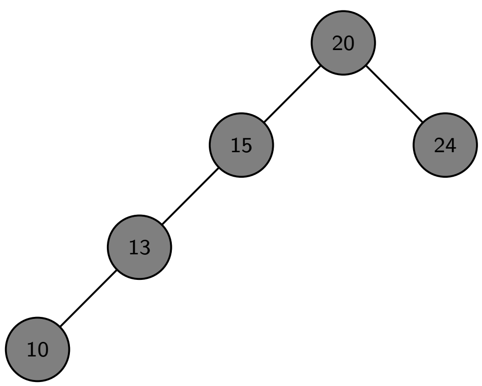


Class Exercises

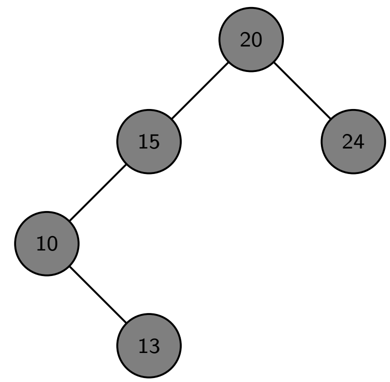
- Build an AVL tree with the following values:

$$15, 20, 24, 10, 13, 7, 30, 36, 25$$





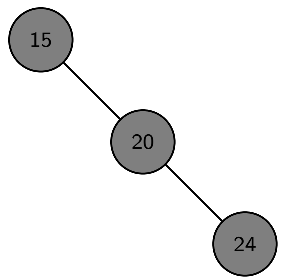
$\Rightarrow$



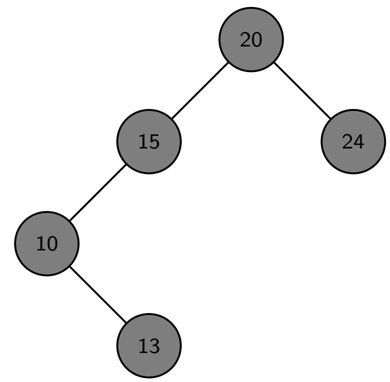
Class Exercises

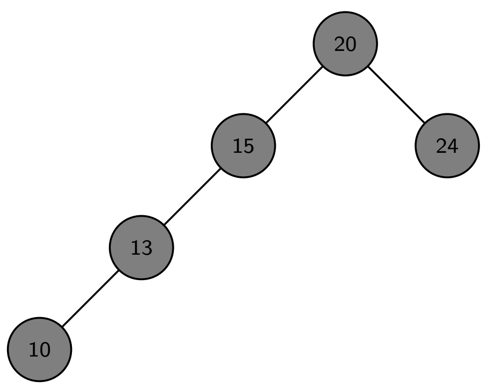
- Build an AVL tree with the following values:

$$15, 20, 24, \textcolor{red}{10, 13}, 7, 30, 36, 25$$

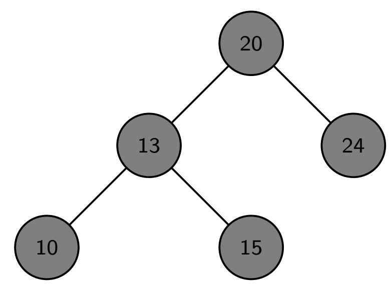


$\Rightarrow$





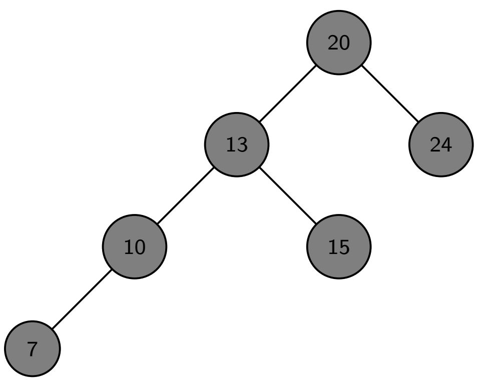
$\Rightarrow$



Class Exercises

- Build an AVL tree with the following values:

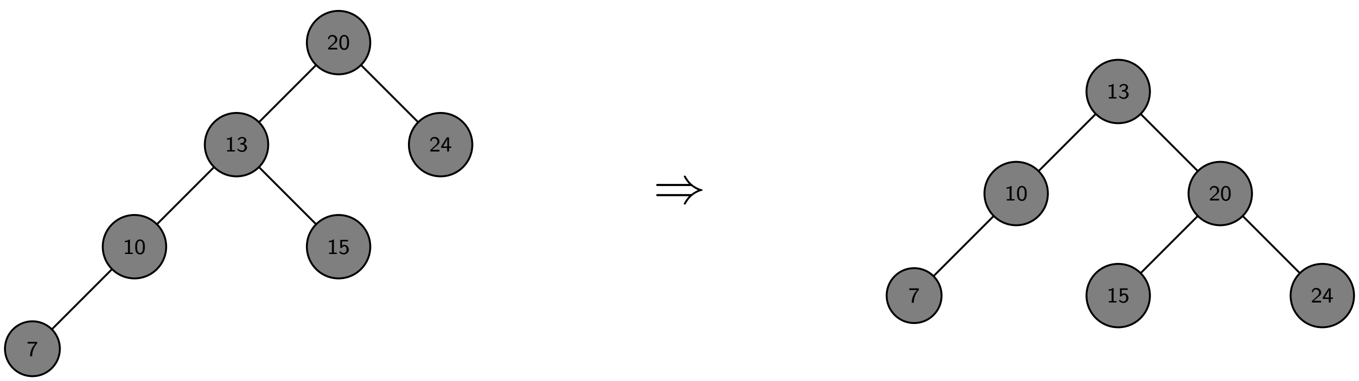
$$15, 20, 24, 10, 13, 7, 30, 36, 25$$



Class Exercises

- Build an AVL tree with the following values:

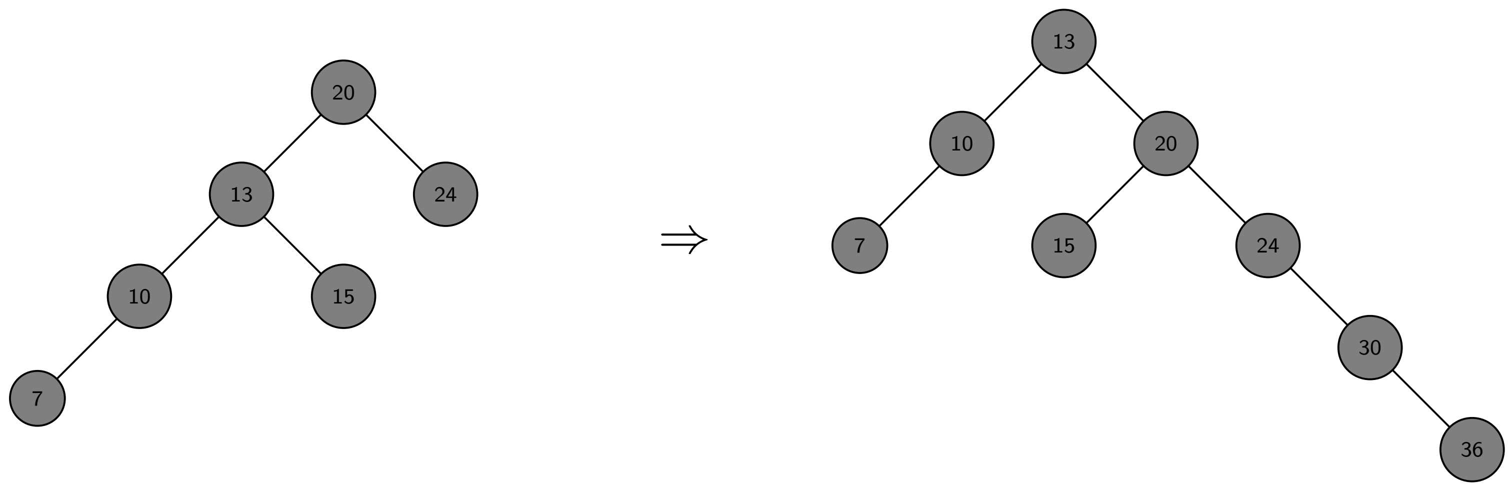
$$15, 20, 24, 10, 13, 7, 30, 36, 25$$



Class Exercises

- Build an AVL tree with the following values:

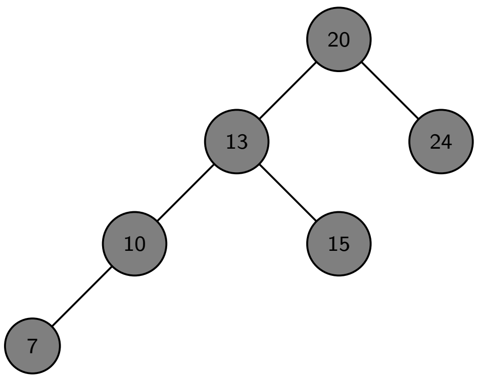
$$15, 20, 24, 10, 13, 7, 30, 36, 25$$



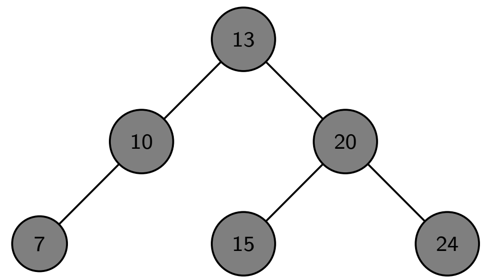
Class Exercises

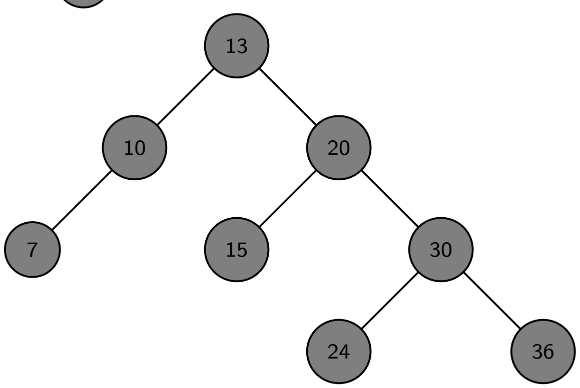
- Build an AVL tree with the following values:

$$15, 20, 24, 10, 13, 7, 30, 36, 25$$



$\Rightarrow$

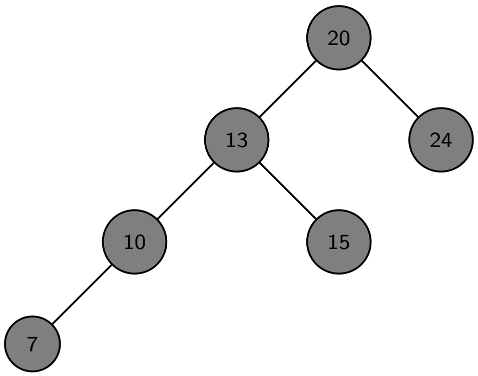




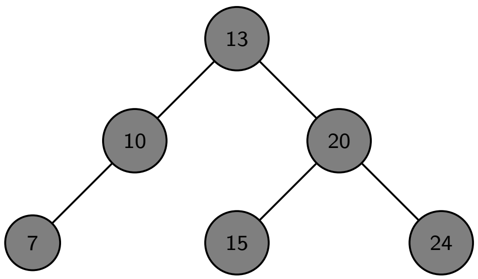
Class Exercises

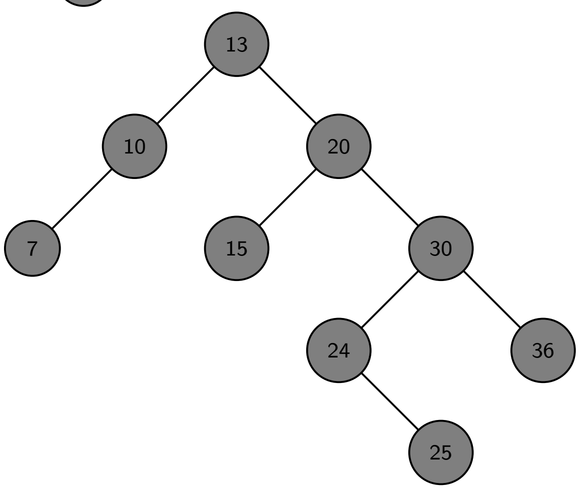
- Build an AVL tree with the following values:

$$15, 20, 24, 10, 13, 7, 30, 36, 25$$



$\Rightarrow$

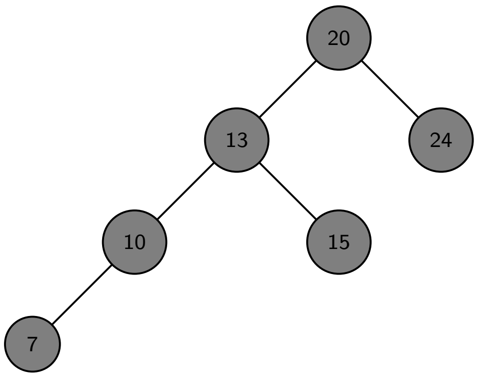




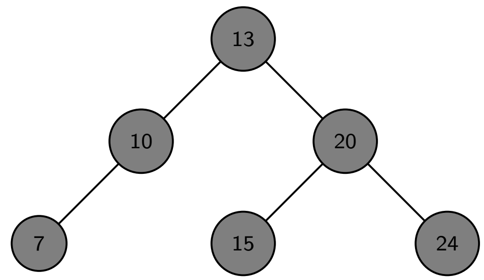
Class Exercises

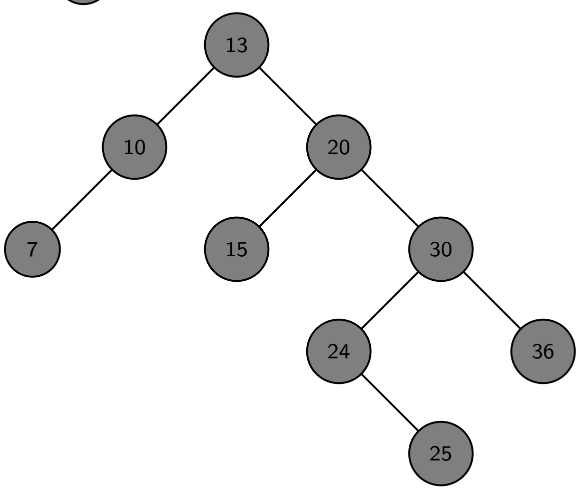
- Build an AVL tree with the following values:

$$15, 20, 24, 10, 13, 7, 30, 36, \color{red}{25}$$

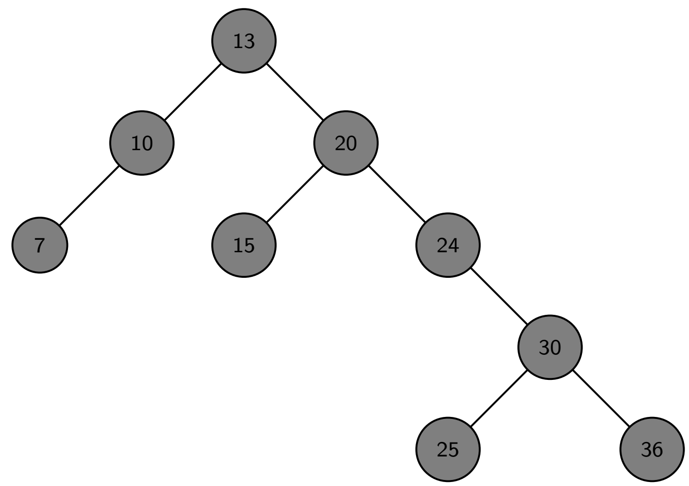


$\Rightarrow$





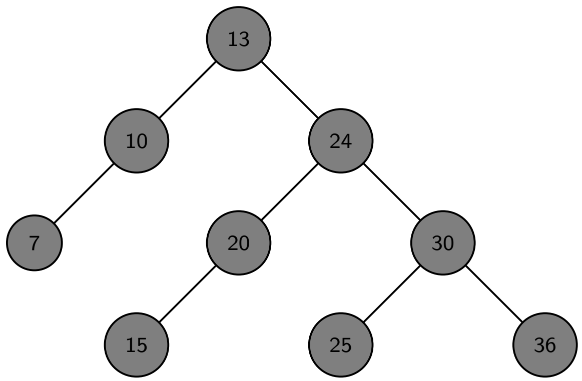
$\Rightarrow$



Class Exercises

- Build an AVL tree with the following values:

$$15, 20, 24, 10, 13, 7, 30, 36, 25$$

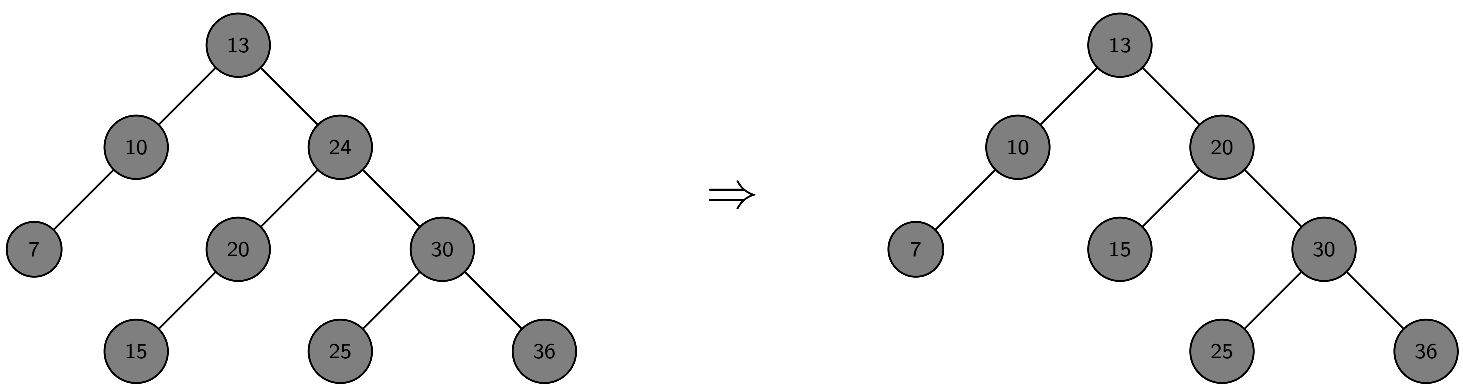


Class Exercises

- Build an AVL tree with the following values:

$$15, 20, 24, 10, 13, 7, 30, 36, 25$$

- Delete: 24,

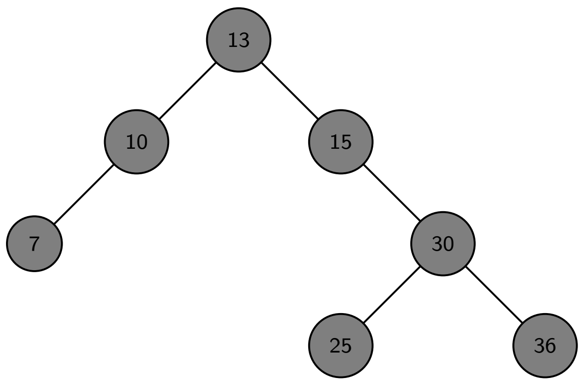


Class Exercises

- Build an AVL tree with the following values:

$$15, 20, 24, 10, 13, 7, 30, 36, 25$$

- Delete: 24, 20

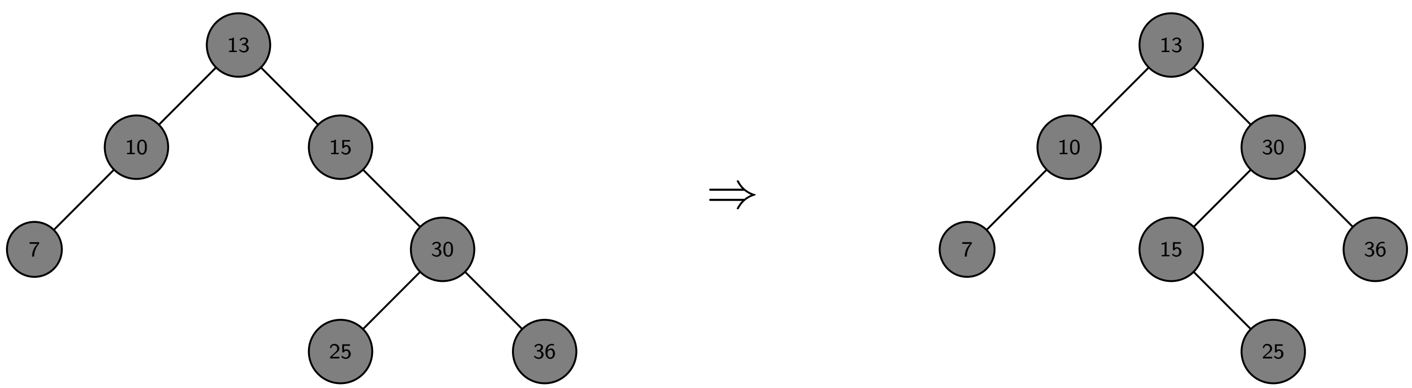


Class Exercises

- Build an AVL tree with the following values:

$$15, 20, 24, 10, 13, 7, 30, 36, 25$$

- Delete: 24, 20



Worst Case Complexity

- Let N_h denote the minimum number of nodes in an AVL tree of height h
- Clearly, $N_i \geq N_{i-1}$ by definition.
- Therefore, we have

$$\begin{aligned} N \geq N_h &\geq N_{h-1} + N_{h-2} + 1 \\ &> N_{h-1} + N_{h-2} \quad \text{[Fibonacci Series!]} \\ &\approx \frac{\phi^h}{\sqrt{5}} \\ \Rightarrow \log_2 N &> h \log_2 \phi - \frac{1}{2} \log_2 5 \\ \Rightarrow h &< 1.4404 \log_2 N + c, \end{aligned}$$

where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ration and $c = \frac{\log_2 5}{2 \log_2 \phi}$.

Thank You for your kind attention!

Books and Other Materials Consulted

1. The Class Exercise on AVL tree was taken from Prof. Daisy Tang webpage.
2. *Introduction to Algorithms* by Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein.
3. Portion on the AVL Trees taken from Prof. Roy P. Pargas's webpage.
4. The Part on Insertion and Deletion in an AVL Tree is taken from Prof. Roy P. Pargas's webpage.

Questions!!