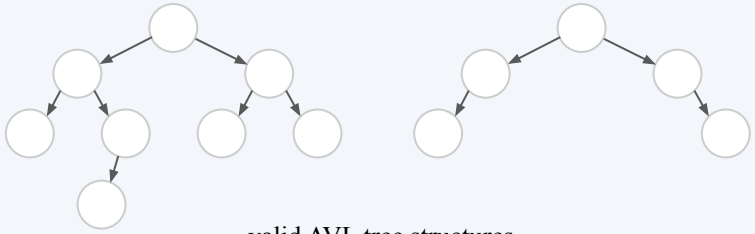
AVL Trees

AVL Trees



valid AVL tree structures

According to lecture:

An AVL Tree is a tree data structure with the following properties:

Structural Property

- *Each node has ≤ 2 children.*
- *Heights of the left and right subtrees of every node differ by at most 1.*

Order Property

- *All keys in the left subtree < node's key.*
- *All keys in the right subtree > node's key.*

Notice that this is just a Binary Search Tree (BST) with 1 additional property.

Problem 0

Problem 0

What are the constraints on the data types you can store in an AVL tree?

- Keys must be of a comparable data type.
- Values can be any data type.

When is an AVL tree preferred over another dictionary implementation, such as a HashMap?

- In an AVL tree, keys can be iterated over in sorted order.

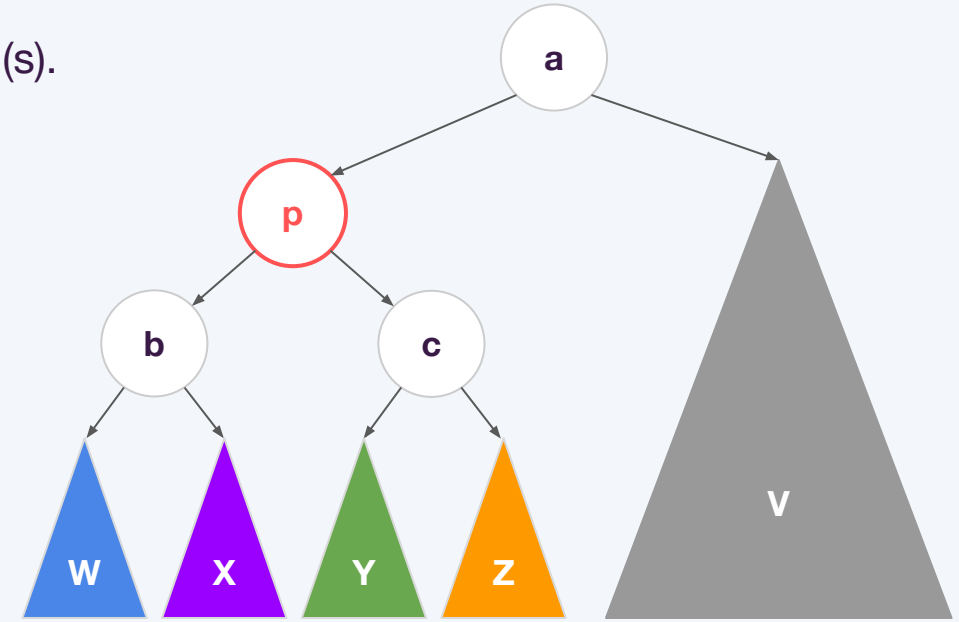
AVL Tree Rotations Microteach

AVL Tree Rotations

To fix an imbalance in an AVL tree after an insert:

1. Identify the *lowest* problem node **p** where the imbalance occurs.
2. Identify the insert location.
3. Execute the corresponding tree rotation(s).

Case	Insert Location	Tree Rotation(s)
1	Left subtree of Left child (W)	Single right rotation
2	Right subtree of Left child (X)	Double left-right rotation
3	Left subtree of Right child (Y)	Double right-left rotation
4	Right subtree of Right child (Z)	Single left rotation

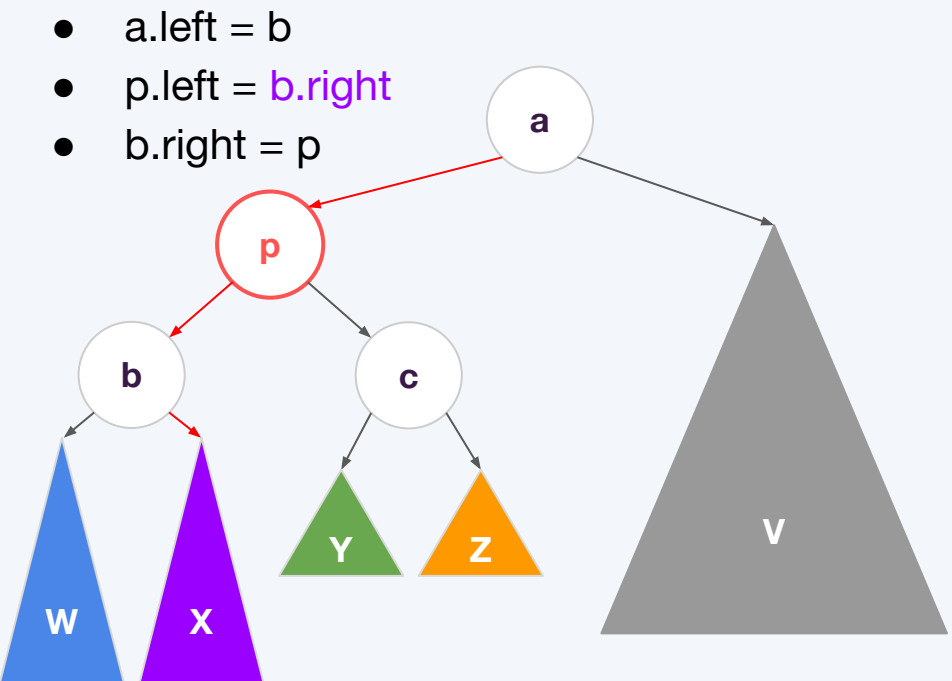


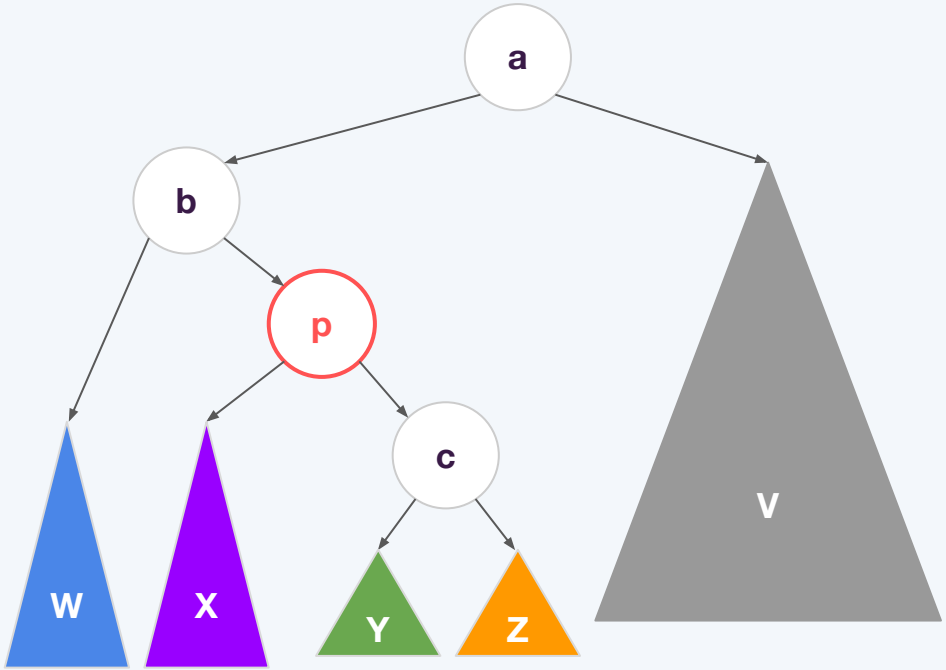
Right Rotation

p is the lowest problem node. Insert location was in w.

References that change:

- a.left = b
- p.left = b.right
- b.right = p



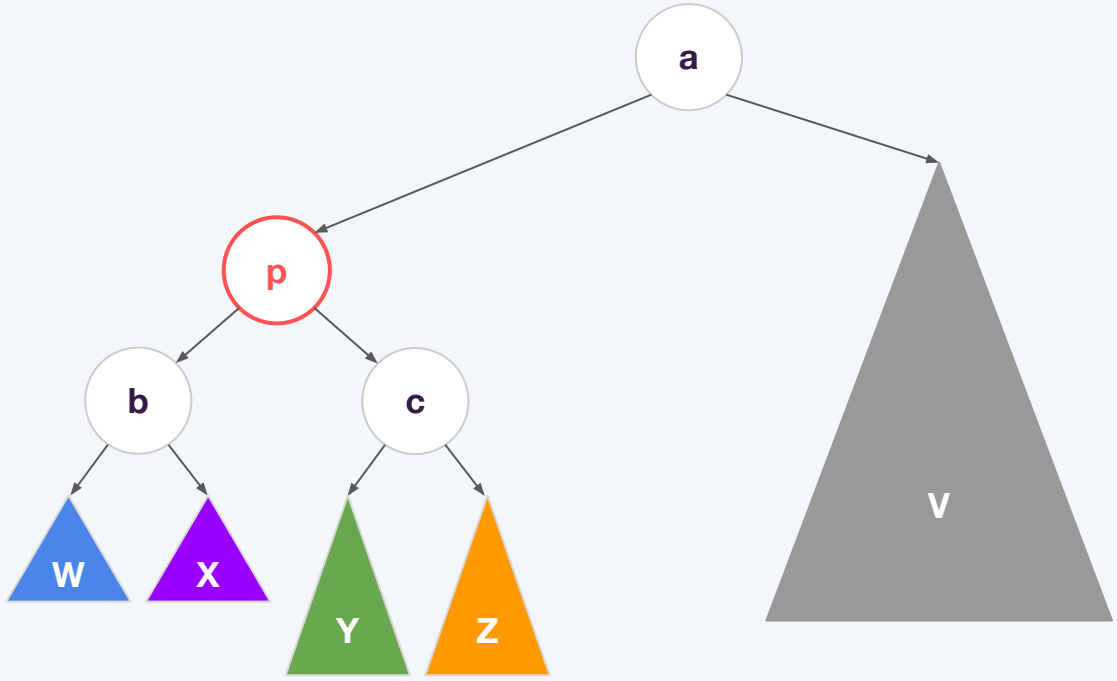


Problem 1 - Left Rotation

p is the lowest problem node. Insert location was in z.

References that change:

-

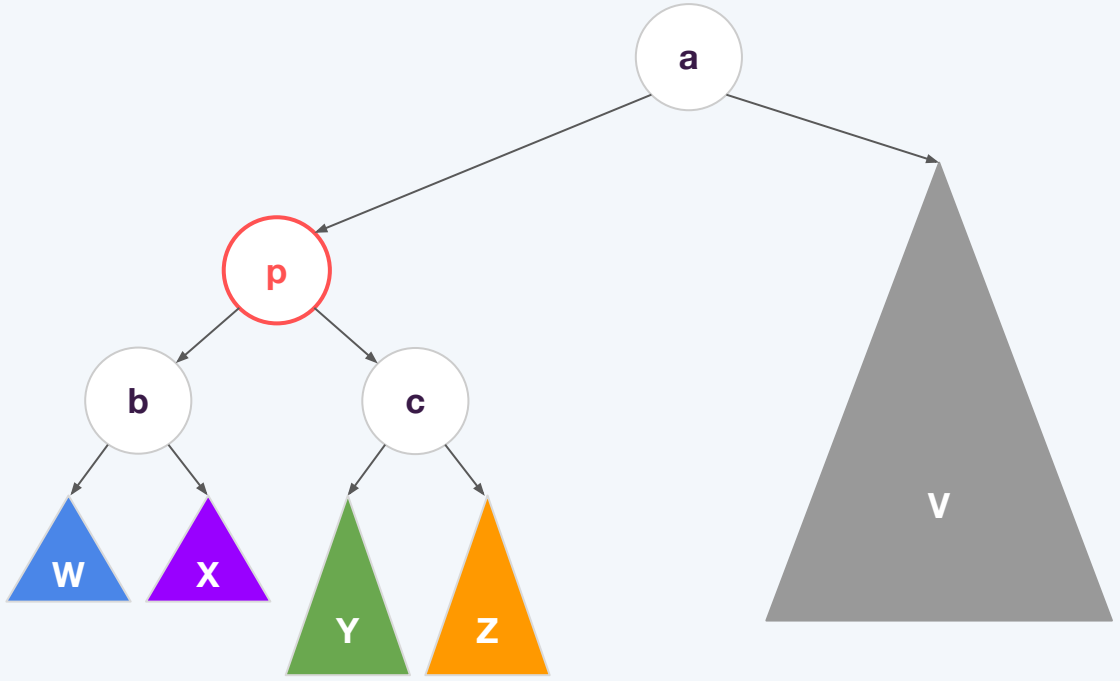


Problem 1 - Left Rotation

p is the lowest problem node. Insert location was in z.

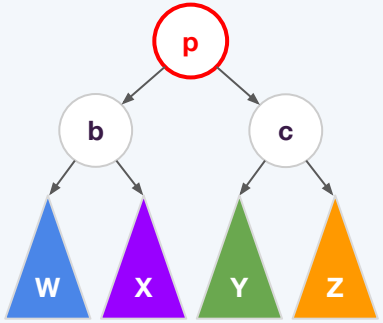
References that change:

- a.left = c
- p.right = c.left
- c.left = p



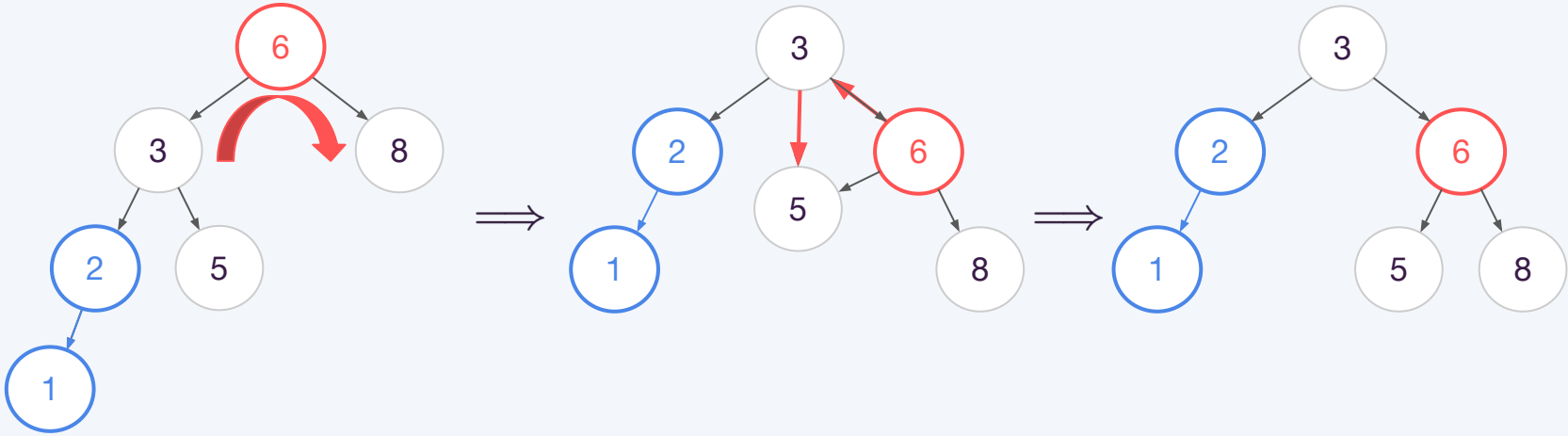
Case 1

Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation



Let's insert 1.

1. Identify the problem node **p**
2. Identify the insert location
3. Execute the tree rotation



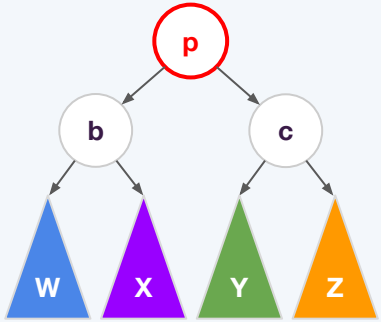
3.1. Identify bad edges

3.2. Remove bad edges

3.3. Replace edges

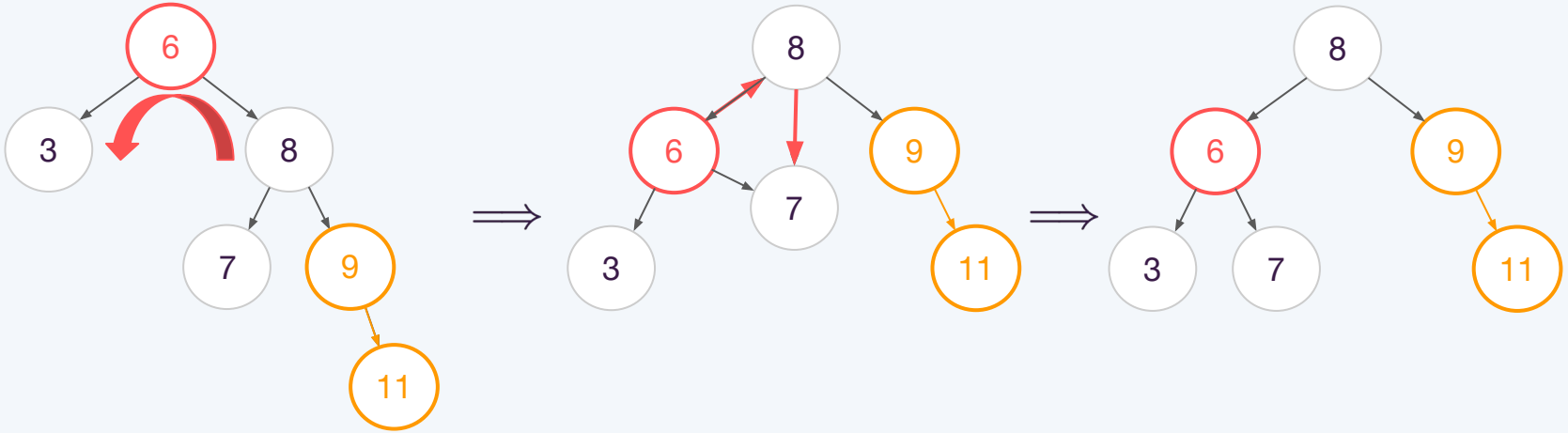
Case 4

Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation



Let's insert 11.

1. Identify the problem node **p**
2. Identify the insert location
3. Execute the tree rotation



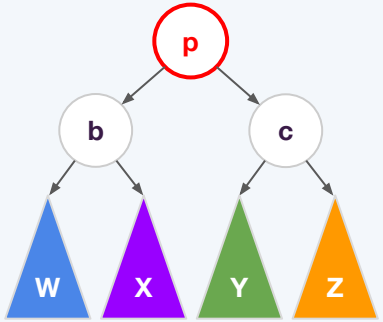
3.1. Identify bad edges

3.2. Remove bad edges

3.3. Replace edges

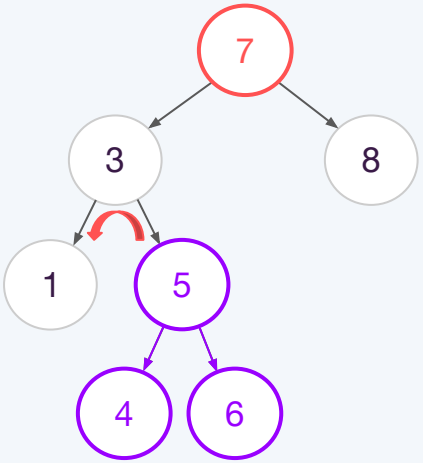
Case 2

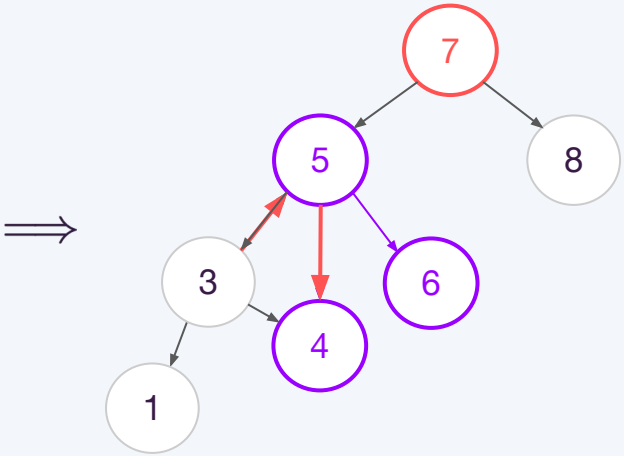
Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation

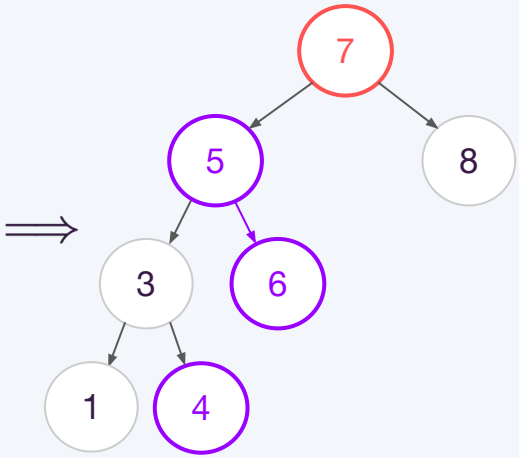


Let's insert either 4 or 6.

1. Identify the problem node **p**
2. Identify the insert location
3. Execute the first tree rotation
   For Case 2, we do the first rotation on node **b**







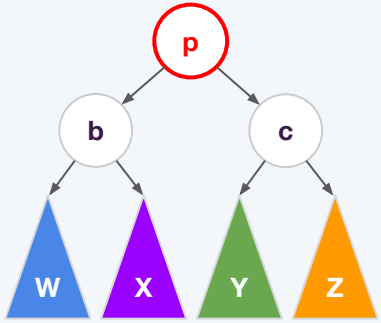
3.1. Identify bad edges

3.2. Remove bad edges

3.3. Replace edges

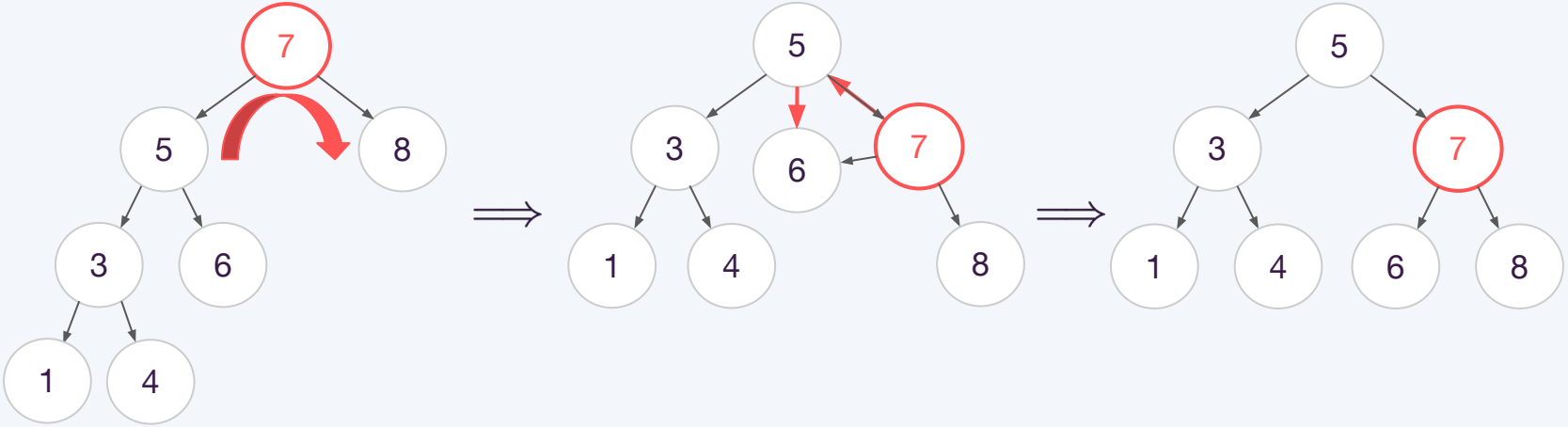
Case 2

Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation



This is now similar to Case 1!

4. Execute the second tree rotation
   We do the second rotation on node **p**



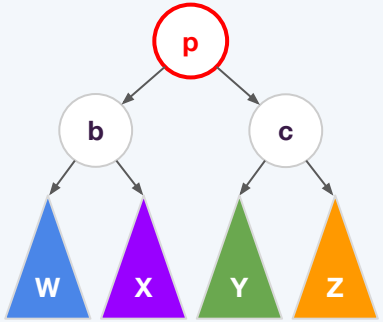
4.1. Identify bad edges

4.2. Remove bad edges

4.3. Replace edges

Case 3

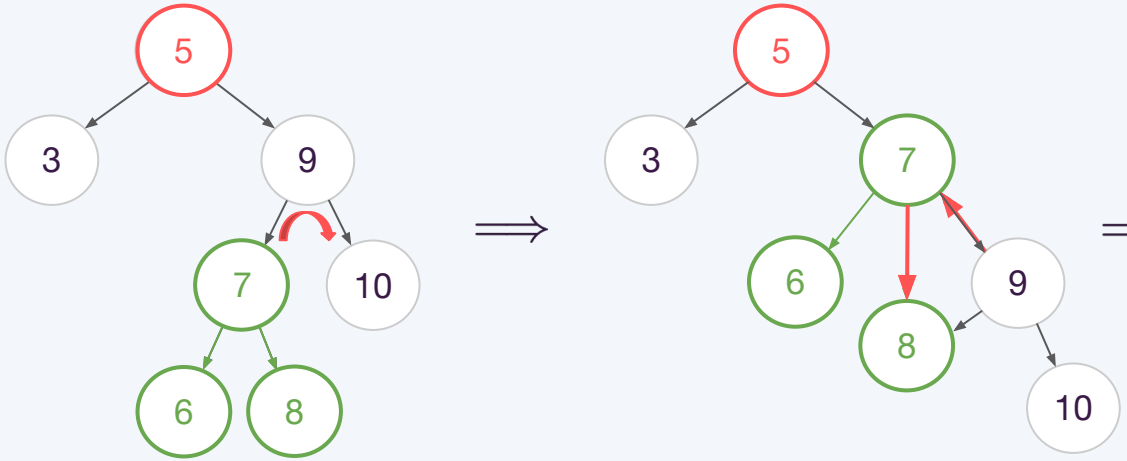
Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation



Let's insert either 6 or 8.

1. Identify the problem node **p**
2. Identify the insert location
3. Execute the first tree rotation

For Case 3, we do the first rotation on node **c**



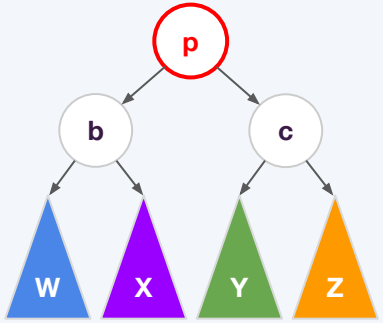
3.1. Identify bad edges

3.2. Remove bad edges

3.3. Replace edges

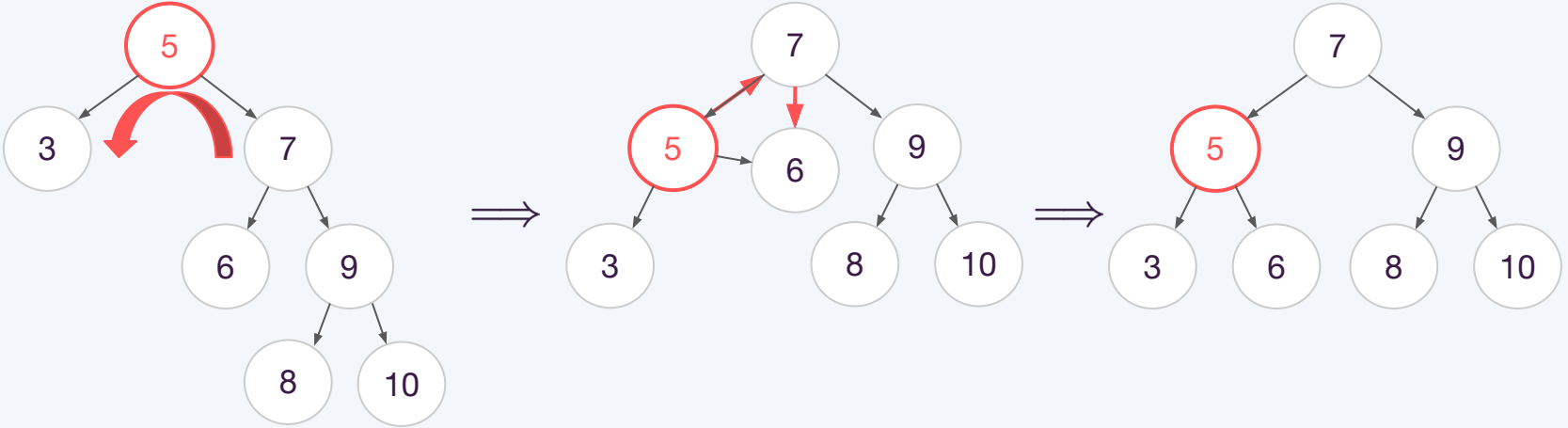
Case 3

Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation



This is now similar to Case 4!

4. Execute the second tree rotation
   We do the second rotation on node **p**



4.1. Identify bad edges

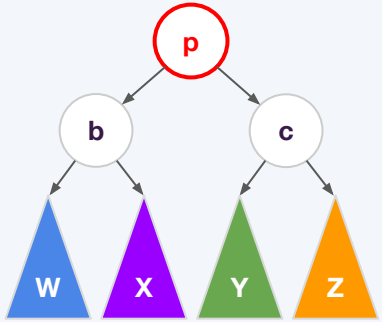
4.2. Remove bad edges

4.3. Replace edges

Problem 2

Problem 2

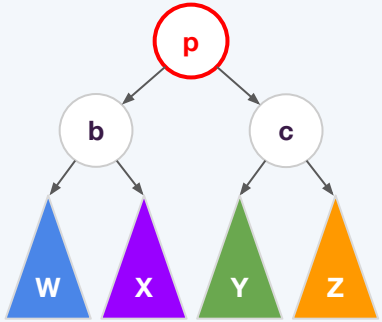
Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation



Insert 10, 4, 5, 8, 9, 6, 11, 3, 2, 1, 14 into an initially empty AVL Tree

Problem 2

Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation



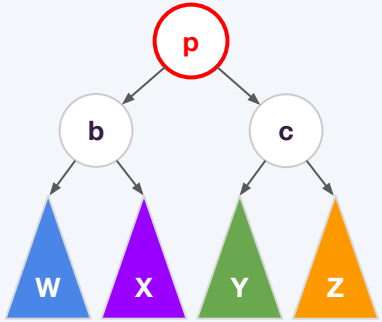
Insert **10**, 4, 5, 8, 9, 6, 11, 3, 2, 1, 14 into an initially empty AVL Tree

No imbalances

10

Problem 2

Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation

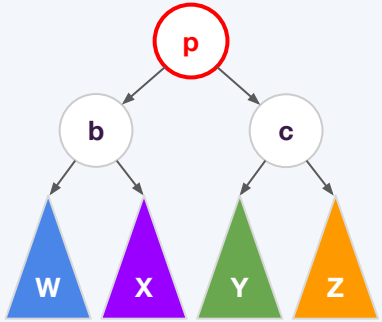


Insert 10, **4**, 5, 8, 9, 6, 11, 3, 2, 1, 14 into an initially empty AVL Tree

No imbalances

Problem 2

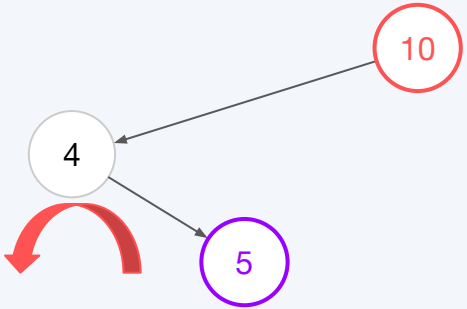
Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation



Insert 10, 4, **5**, 8, 9, 6, 11, 3, 2, 1, 14 into an initially empty AVL Tree

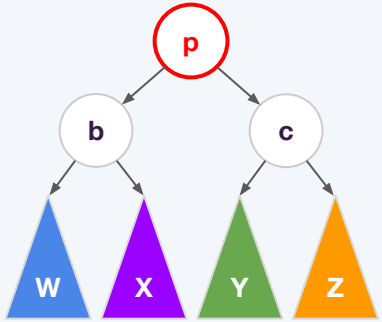
Imbalance at 10

- Case 2
- Rotate left at 4



Problem 2

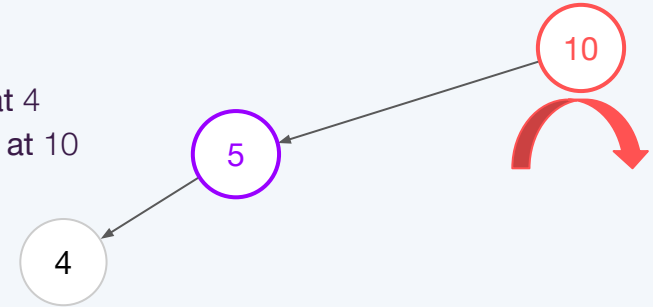
Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation



Insert 10, 4, **5**, 8, 9, 6, 11, 3, 2, 1, 14 into an initially empty AVL Tree

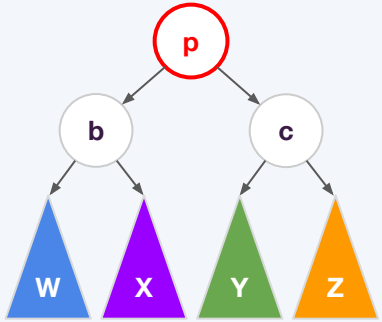
Imbalance at 10

- Case 2
- Rotate left at 4
- Rotate right at 10



Problem 2

Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation

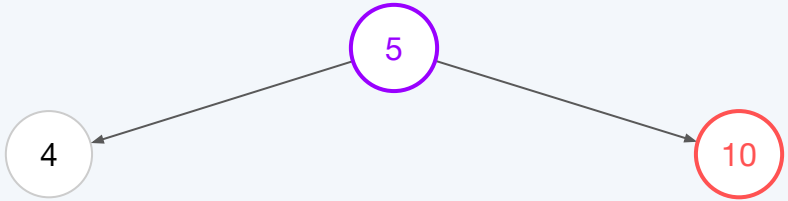


Insert 10, 4, **5**, 8, 9, 6, 11, 3, 2, 1, 14 into an initially empty AVL Tree

Imbalance at 10

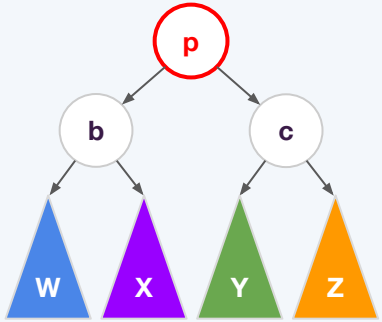
- Case 2
- Rotate left at 4
- Rotate right at 10

No more imbalances



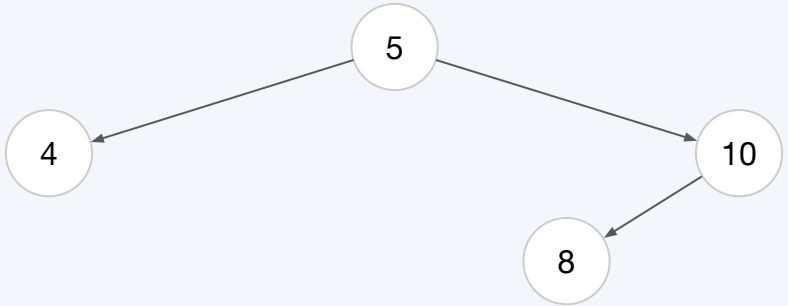
Problem 2

Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation



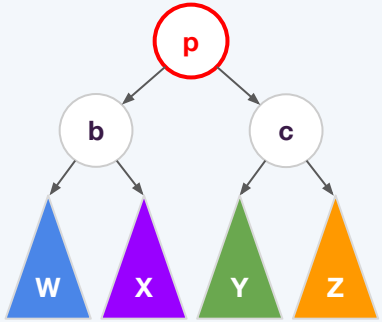
Insert 10, 4, 5, **8**, 9, 6, 11, 3, 2, 1, 14 into an initially empty AVL Tree

No imbalances



Problem 2

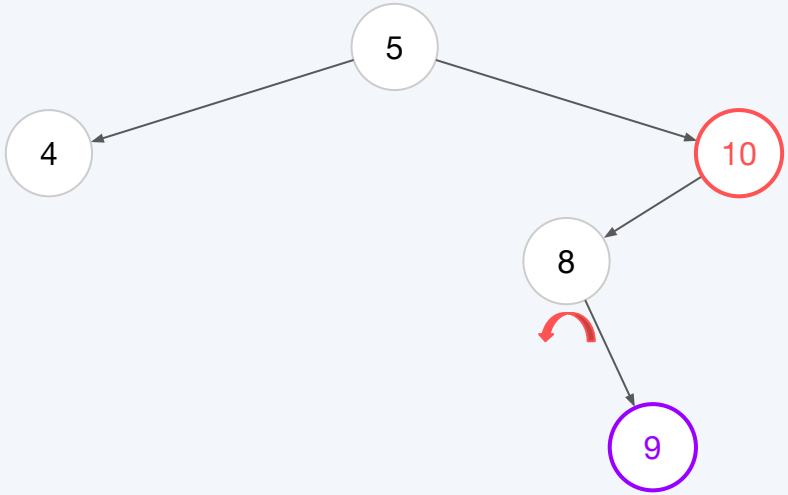
Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation



Insert 10, 4, 5, 8, **9**, 6, 11, 3, 2, 1, 14 into an initially empty AVL Tree

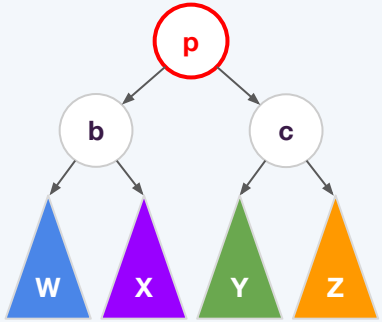
Imbalance at 10

- Case 2
- Rotate left at 8



Problem 2

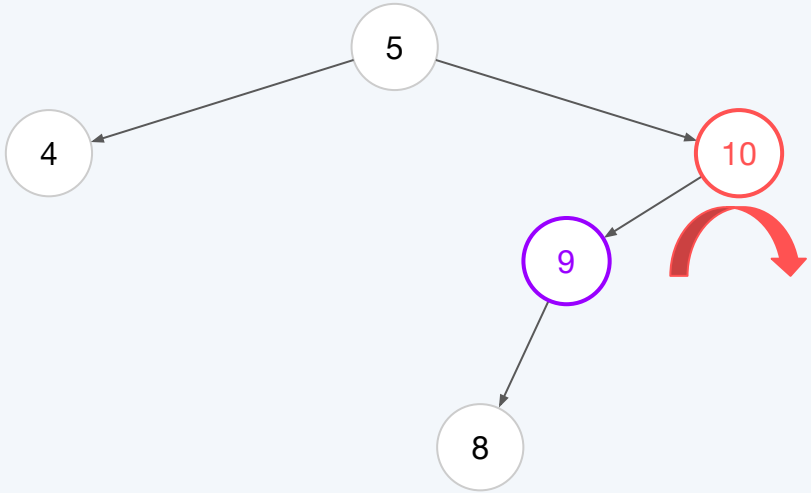
Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation



Insert 10, 4, 5, 8, **9**, 6, 11, 3, 2, 1, 14 into an initially empty AVL Tree

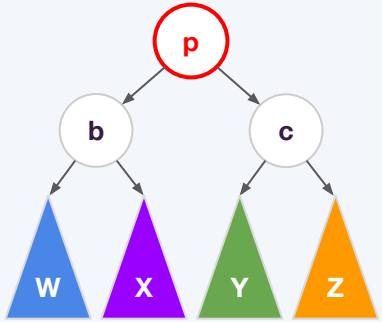
Imbalance at 10

- Case 2
- Rotate left at 8
- Rotate right at 10



Problem 2

Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation

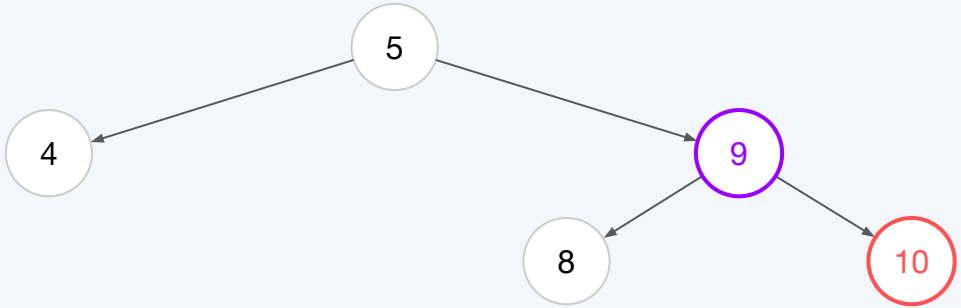


Insert 10, 4, 5, 8, **9**, 6, 11, 3, 2, 1, 14 into an initially empty AVL Tree

Imbalance at 10

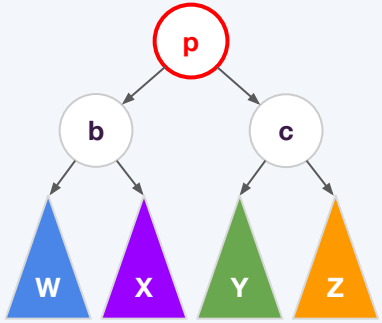
- Case 2
- Rotate left at 8
- Rotate right at 10

No more imbalances



Problem 2

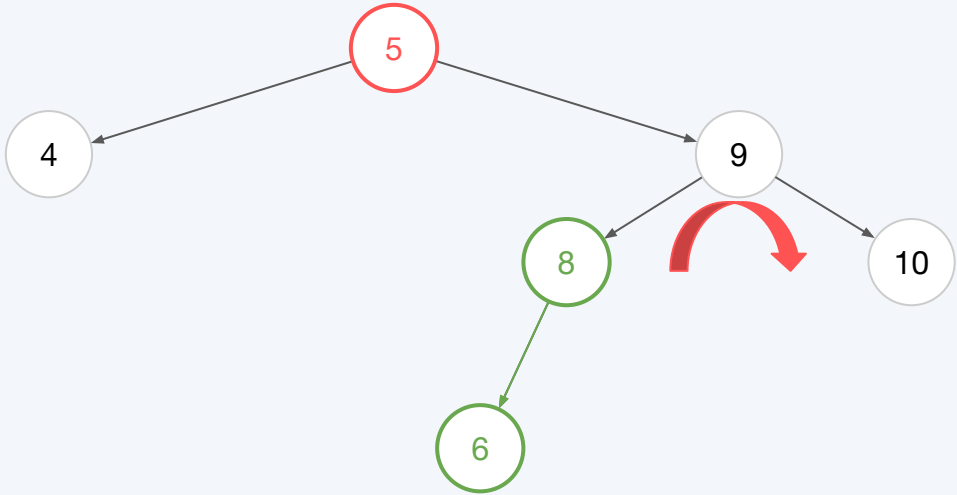
Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation



Insert 10, 4, 5, 8, 9, **6**, 11, 3, 2, 1, 14 into an initially empty AVL Tree

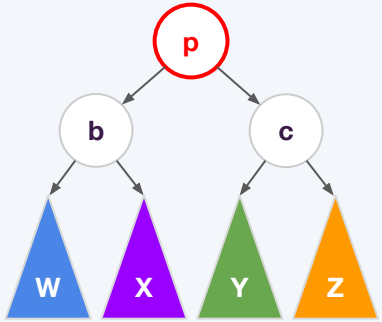
Imbalance at 5

- Case 3
- Rotate right at 9



Problem 2

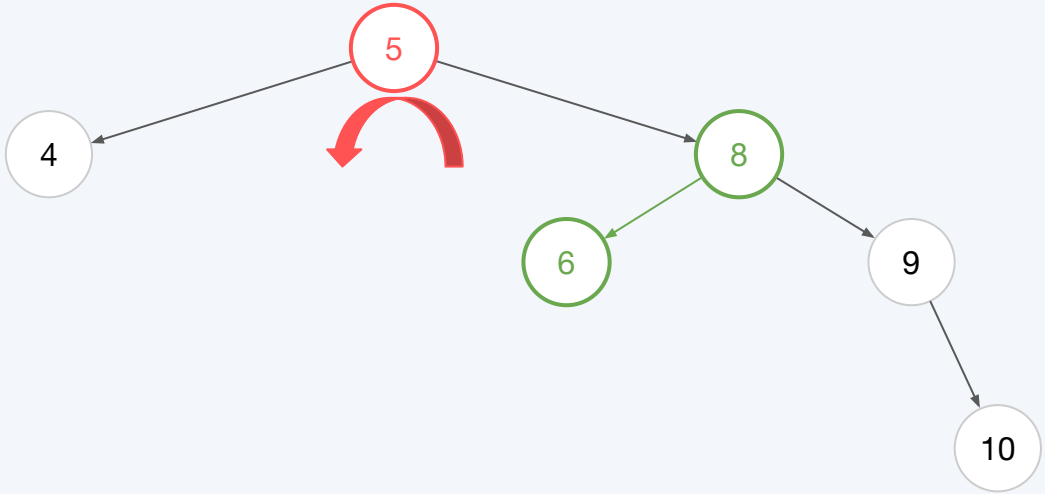
Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation



Insert 10, 4, 5, 8, 9, **6**, 11, 3, 2, 1, 14 into an initially empty AVL Tree

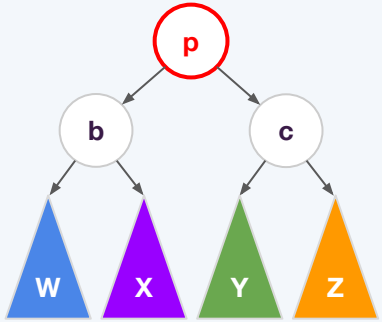
Imbalance at 5

- Case 3
- Rotate right at 9
- Rotate left at 5



Problem 2

Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation

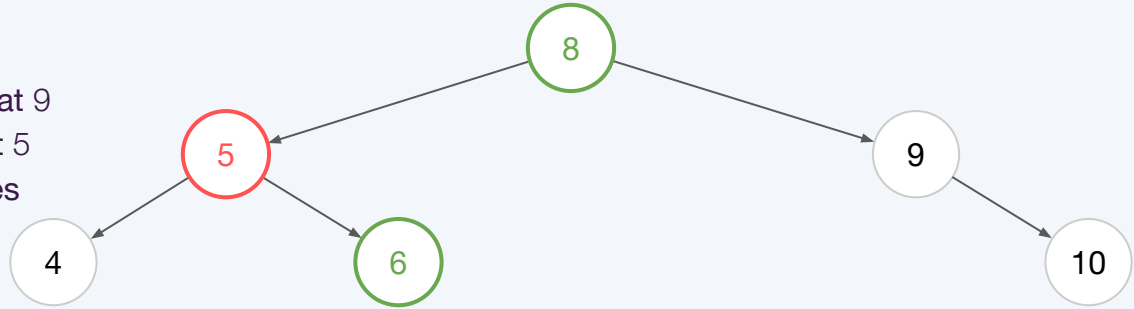


Insert 10, 4, 5, 8, 9, **6**, 11, 3, 2, 1, 14 into an initially empty AVL Tree

Imbalance at 5

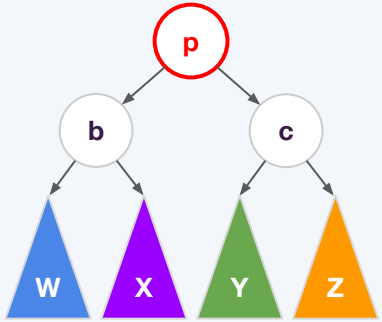
- Case 3
- Rotate right at 9
- Rotate left at 5

No more imbalances



Problem 2

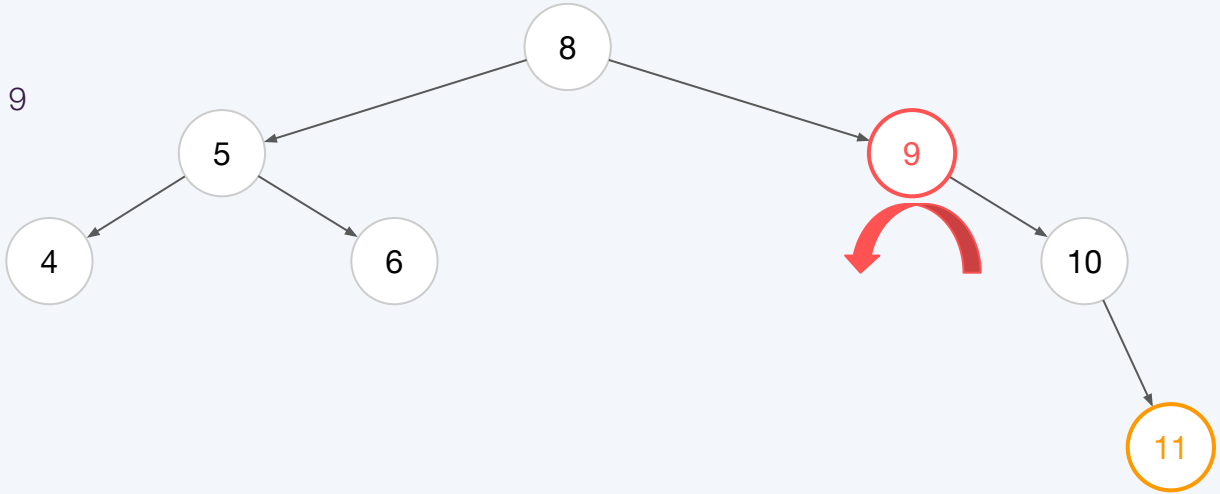
Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation



Insert 10, 4, 5, 8, 9, 6, **11**, 3, 2, 1, 14 into an initially empty AVL Tree

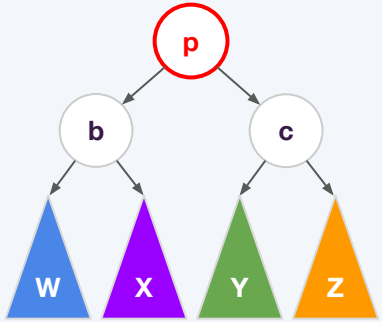
Imbalance at 9

- Case 4
- Rotate left at 9



Problem 2

Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation

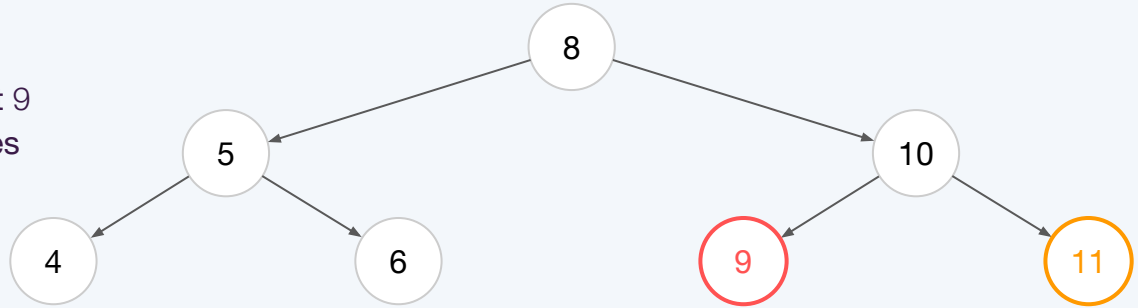


Insert 10, 4, 5, 8, 9, 6, **11**, 3, 2, 1, 14 into an initially empty AVL Tree

Imbalance at 9

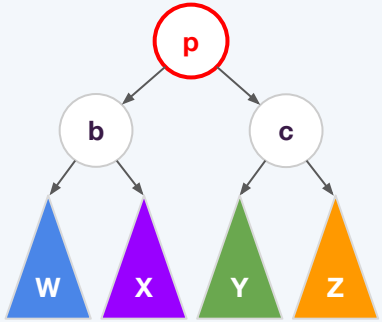
- Case 4
- Rotate left at 9

No more imbalances



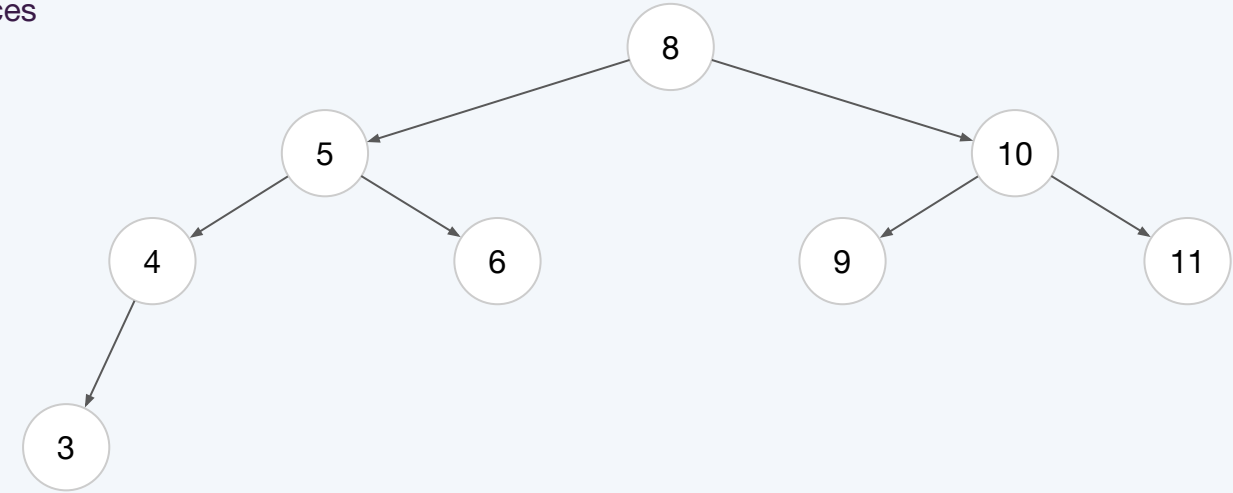
Problem 2

Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation



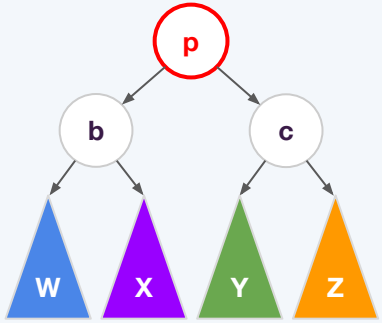
Insert 10, 4, 5, 8, 9, 6, 11, **3**, 2, 1, 14 into an initially empty AVL Tree

No imbalances



Problem 2

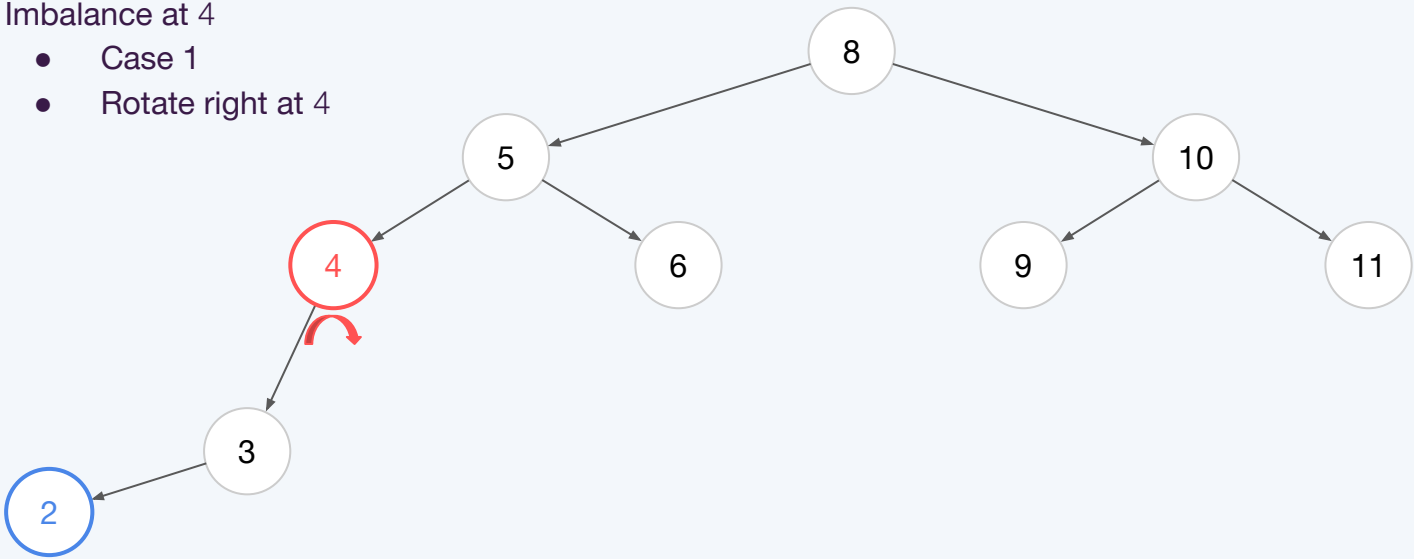
Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation



Insert 10, 4, 5, 8, 9, 6, 11, 3, **2**, 1, 14 into an initially empty AVL Tree

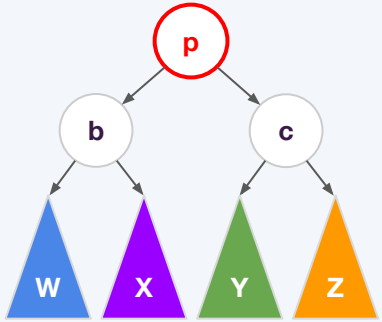
Imbalance at 4

- Case 1
- Rotate right at 4



Problem 2

Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation

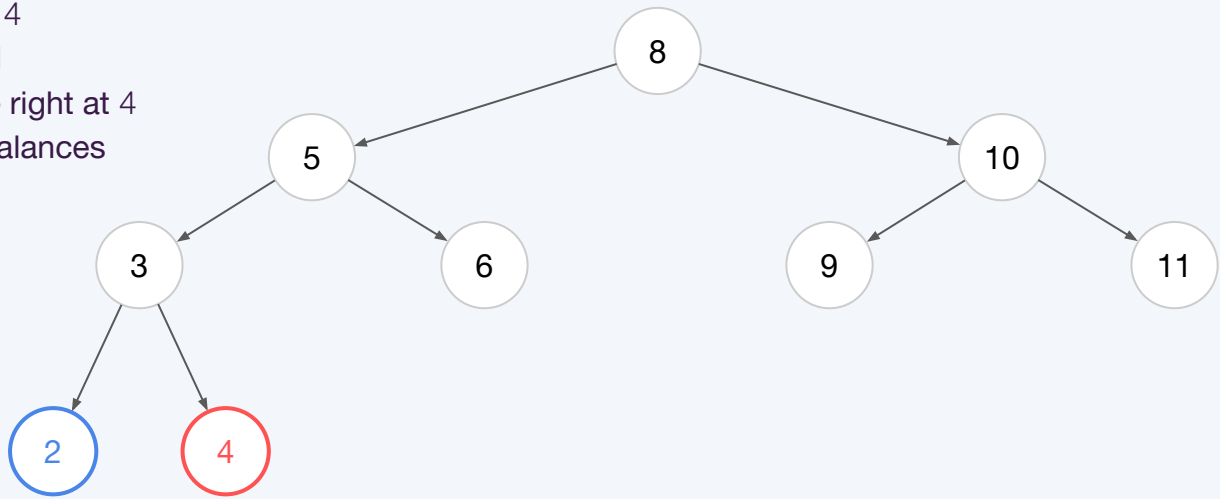


Insert 10, 4, 5, 8, 9, 6, 11, 3, **2**, 1, 14 into an initially empty AVL Tree

Imbalance at 4

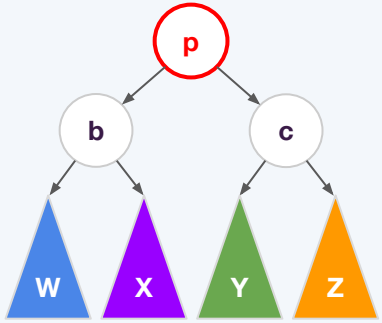
- Case 1
- Rotate right at 4

No more imbalances



Problem 2

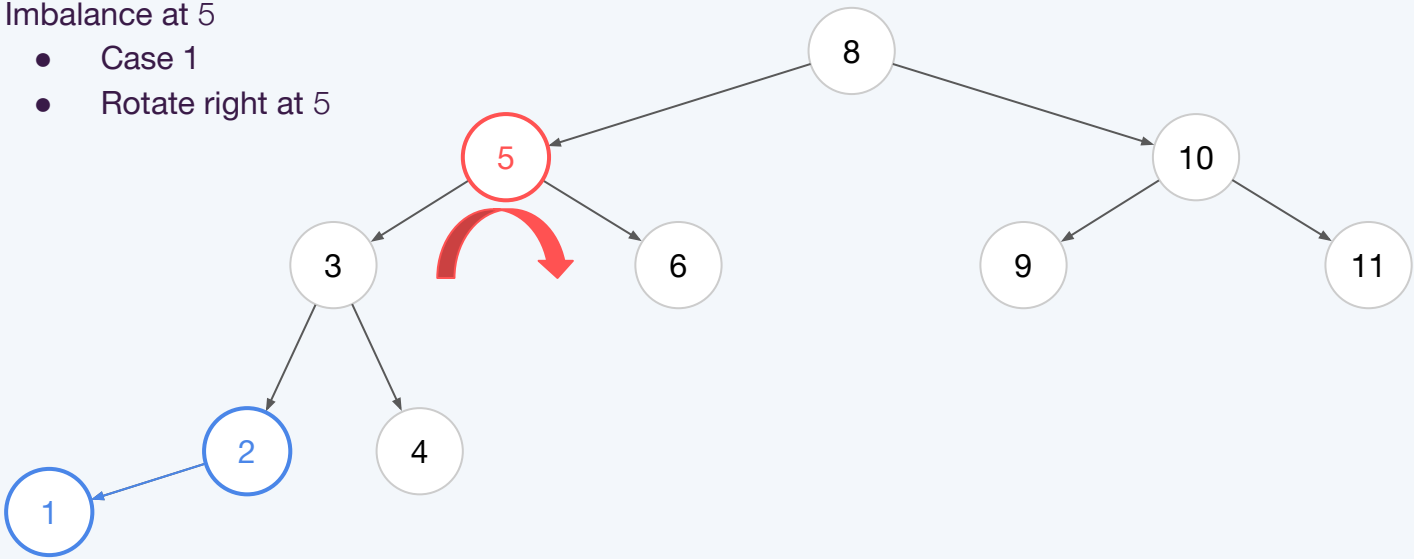
Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation



Insert 10, 4, 5, 8, 9, 6, 11, 3, 2, **1**, 14 into an initially empty AVL Tree

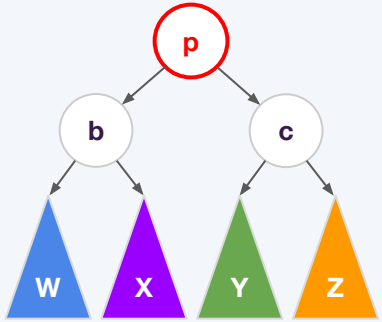
Imbalance at 5

- Case 1
- Rotate right at 5



Problem 2

Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation

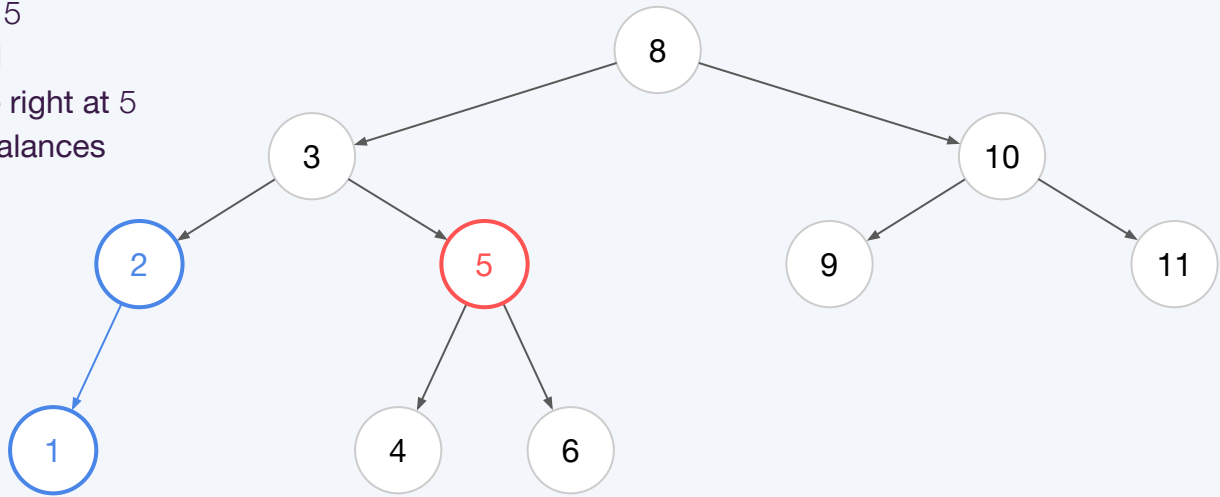


Insert 10, 4, 5, 8, 9, 6, 11, 3, 2, **1**, 14 into an initially empty AVL Tree

Imbalance at 5

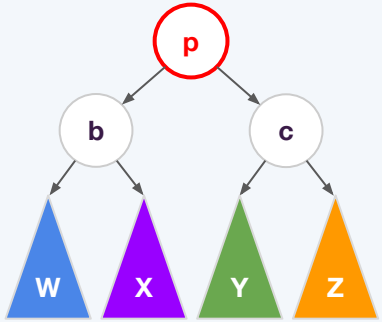
- Case 1
- Rotate right at 5

No more imbalances



Problem 2

Case	Insert Location	Tree Rotation(s)
1	Left of Left (W)	Single right rotation
2	Right of Left (X)	Double left-right rotation
3	Left of Right (Y)	Double right-left rotation
4	Right of Right (Z)	Single left rotation



Insert 10, 4, 5, 8, 9, 6, 11, 3, 2, 1, **14** into an initially empty AVL Tree

No imbalances