B-Trees - 2-3 Trees and AVL Trees

Balanced BSTs, Operations Insertions and Rotations

SoftUni Team

Technical Trainers

Software University

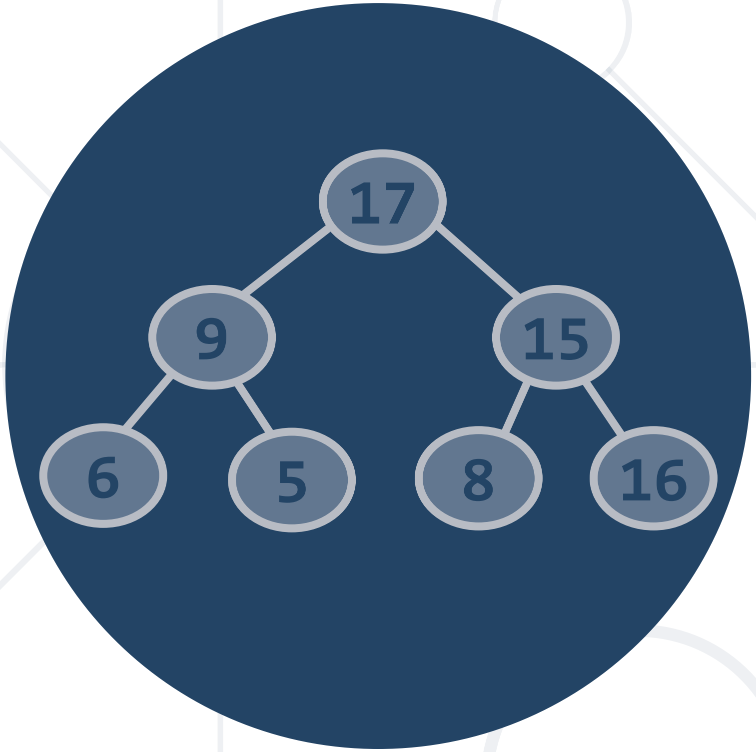
SoftUni

Software University

http://softuni.bg

Table of Contents

1. B-Trees
2. 2-3 Trees
   - Ordered Operations
   - Insertion
3. AVL Trees
   - Properties of AVL
   - Rotations in AVL (Double Left, Double Right)
   - AVL Insertion Algorithm

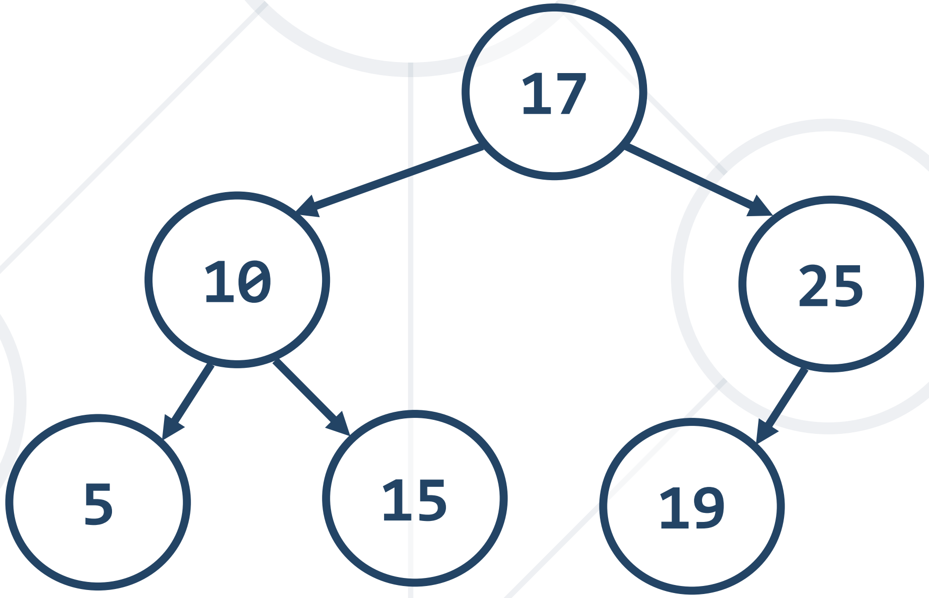


Balanced BSTs

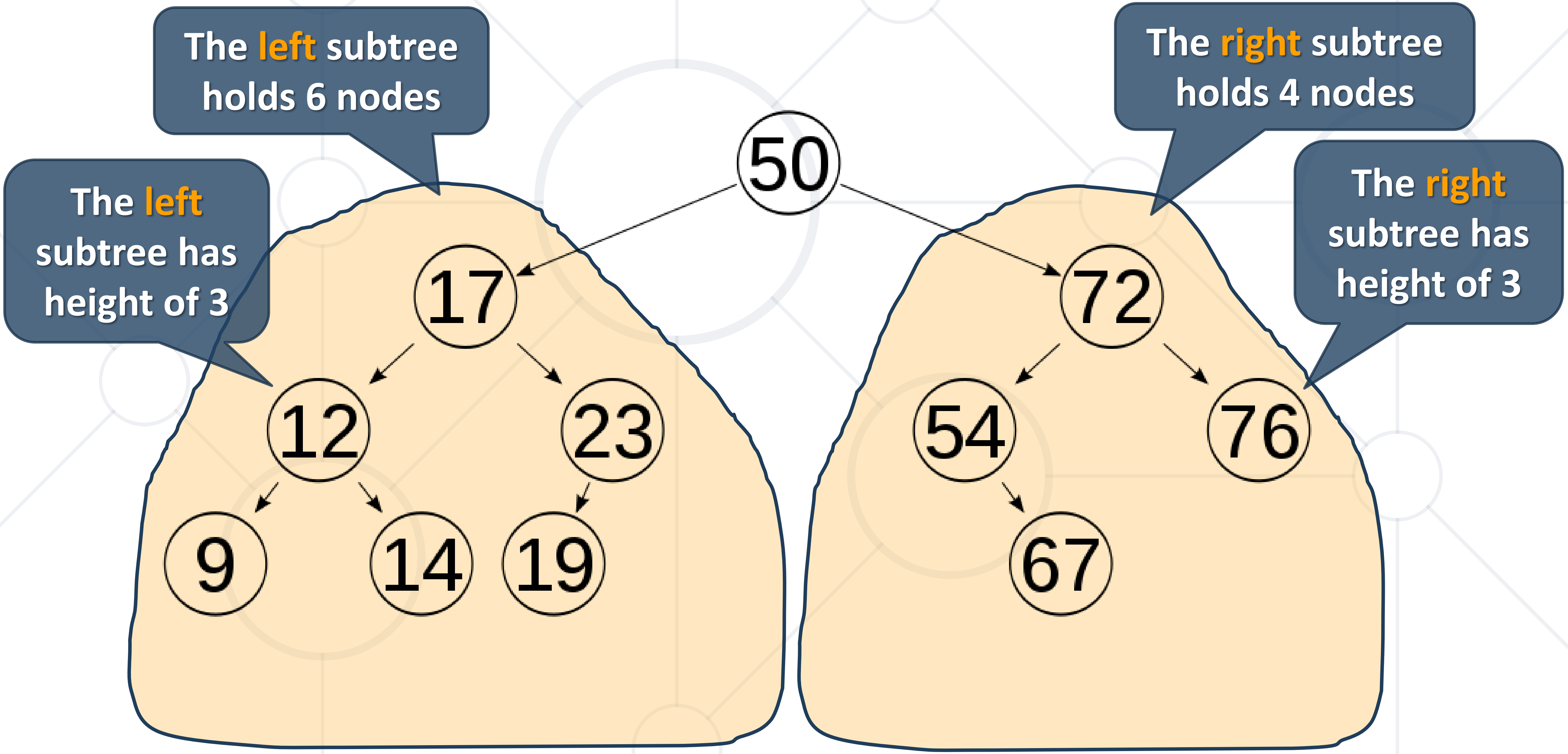
Balancing a BST

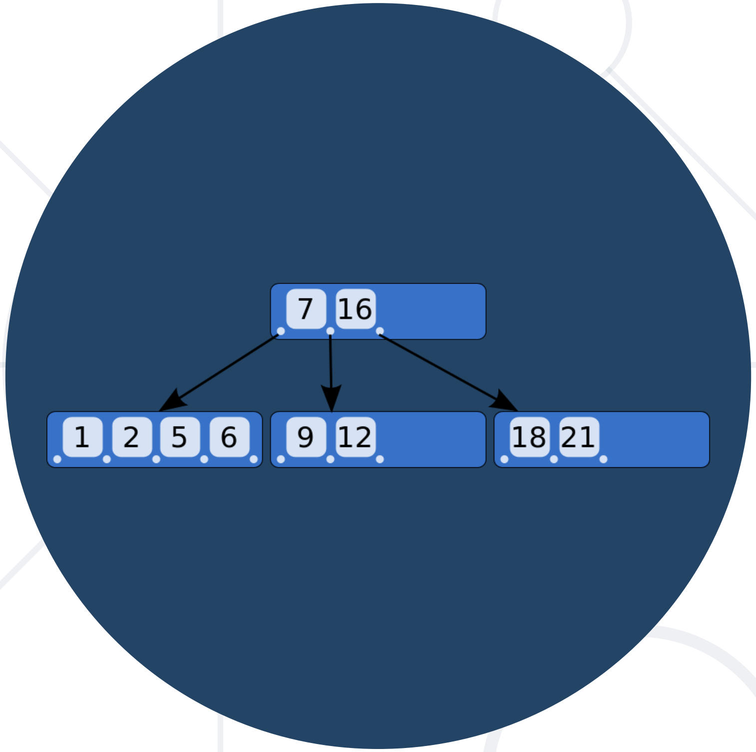
What is a Balanced Binary Search Tree?

- Binary search trees can be **balanced**
  - Subtrees hold nearly equal number of nodes
  - Subtrees are with nearly the same height



Balanced Binary Search Tree – Example





B-Trees

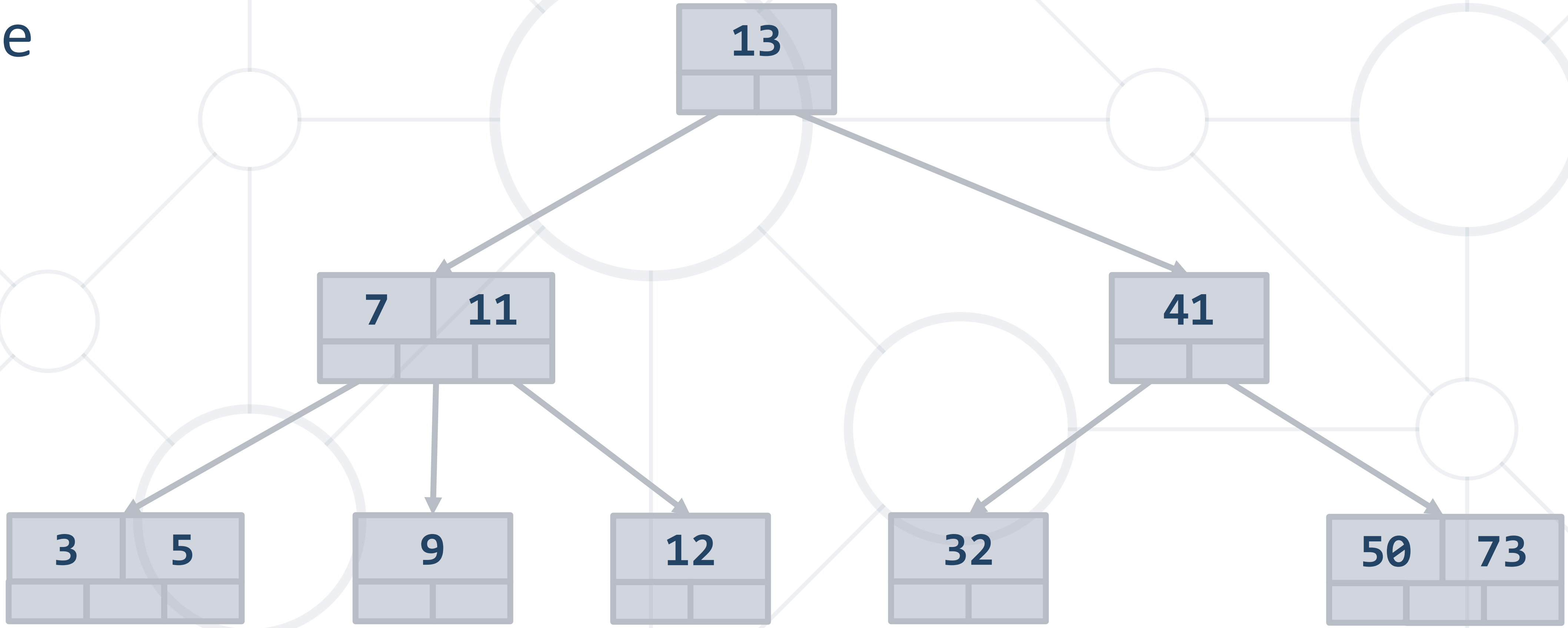
B-Trees Concept

What are B-Trees?

- B-trees are generalization of the concept of ordered binary search trees – see the visualization
  - B-tree of order **b** has between **b** and **2*b** keys in a node and between **b+1** and **2*b+1** child nodes
  - The keys in each node are ordered increasingly
  - All keys in a child node have values between their left and right parent keys
- B-trees can be efficiently stored on the hard disk

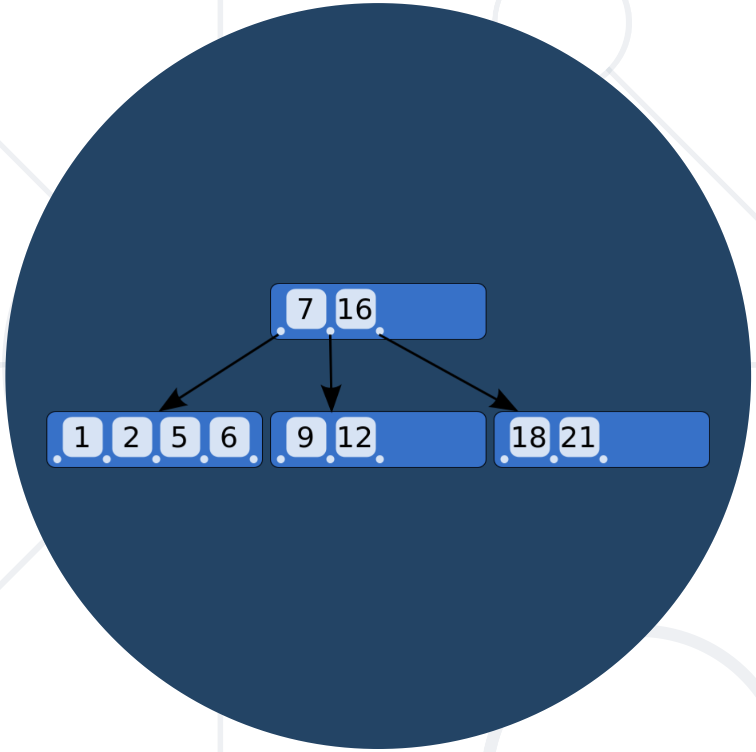
B-Tree – Example

- B-Tree of order **3** (max count of child nodes), also known as 2-3 tree



B-Trees vs. Other Balanced Search Trees

- B-Trees hold a **range of child nodes**, not single one
  - B-trees do not need re-balancing so frequently
- B-Trees are good for **database indexes**
  - Because a single node is stored in a single cluster of the hard drive
  - Minimize the number of disk operations (which are very slow)
- B-Trees are almost perfectly balanced
  - The count of nodes from the root to any **null** node is the same



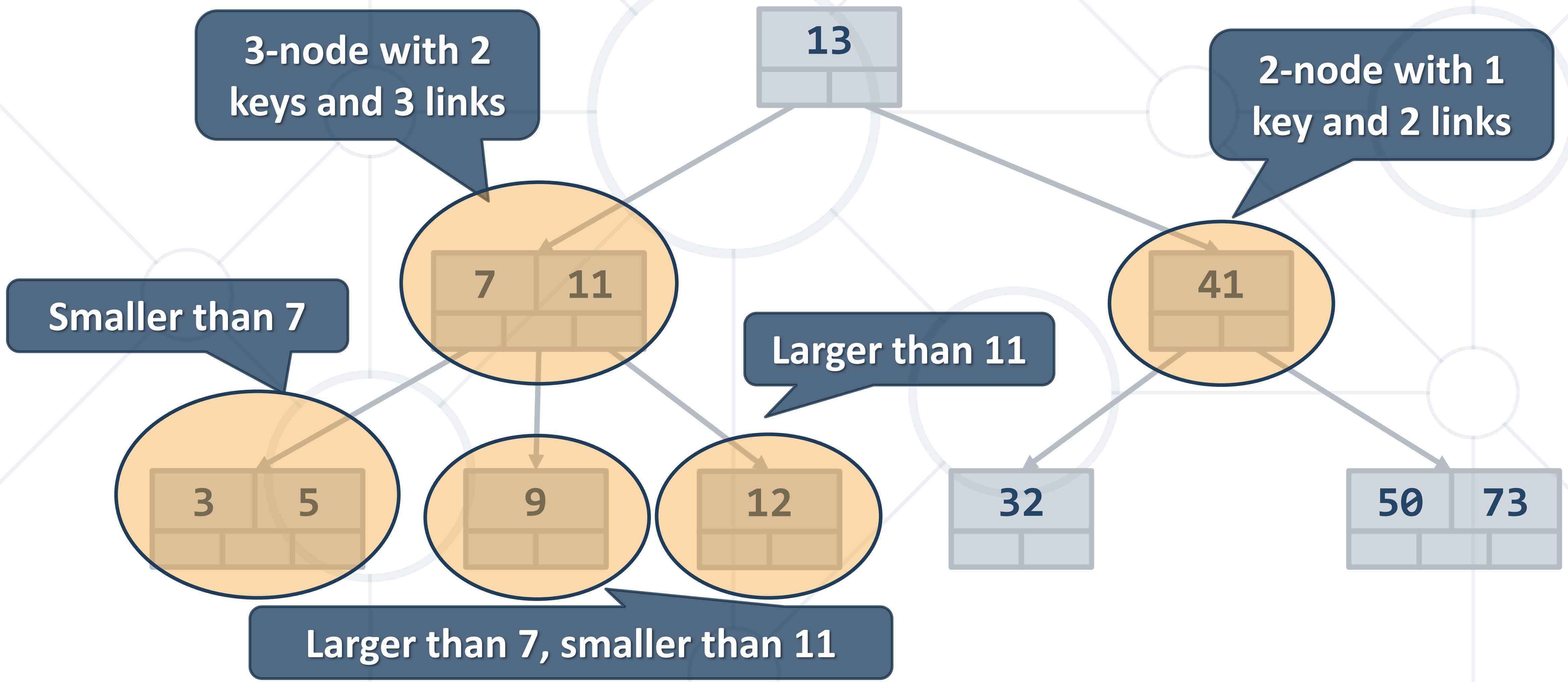
2-3 Trees

2-3 Trees Operations

Definition

- A 2-3 search tree can contain:
  - Empty node (**null**)
  - 2-node with **1 key** and **2 links** (children)
  - 3-node with **2 keys** and **3 links** (children)
- As usual for BSTs, all items to the left are smaller, all items to the right are larger.

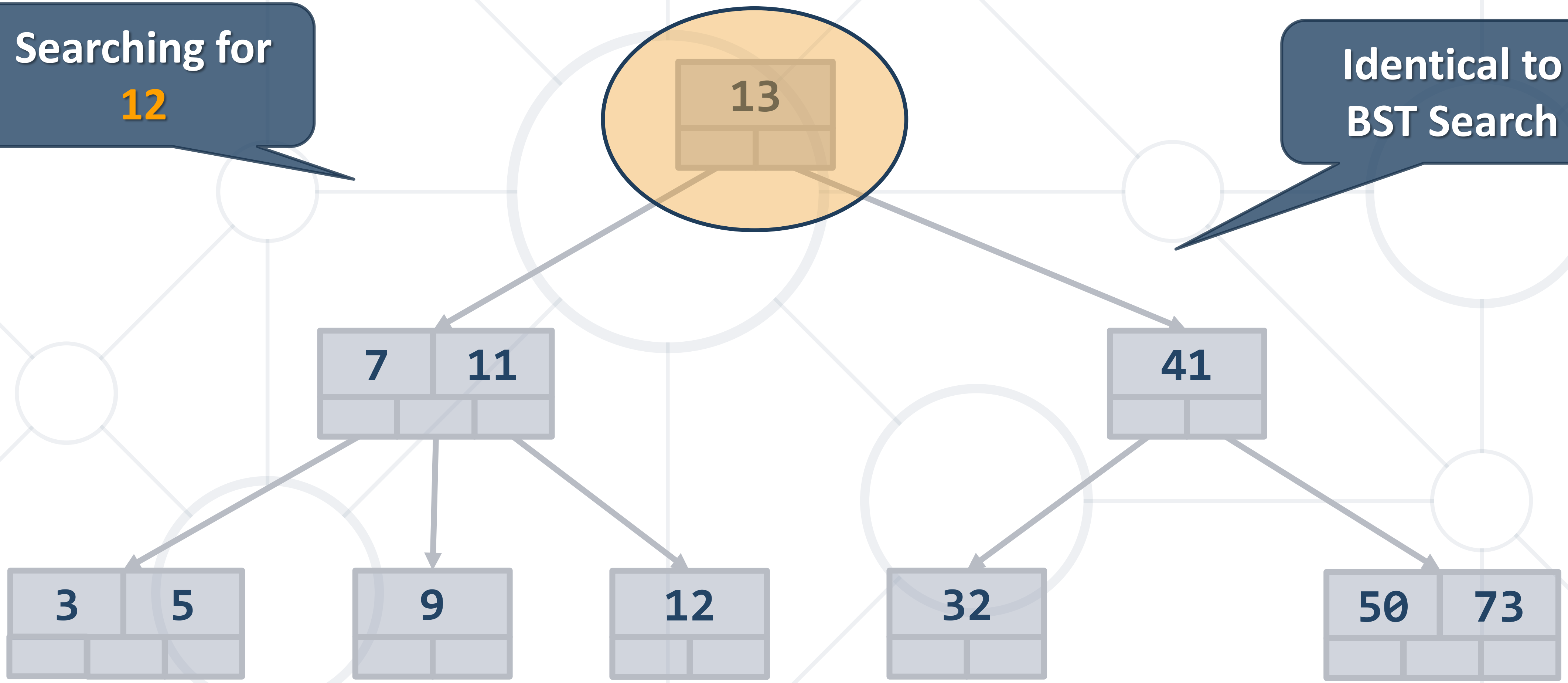
2-3 Tree Example



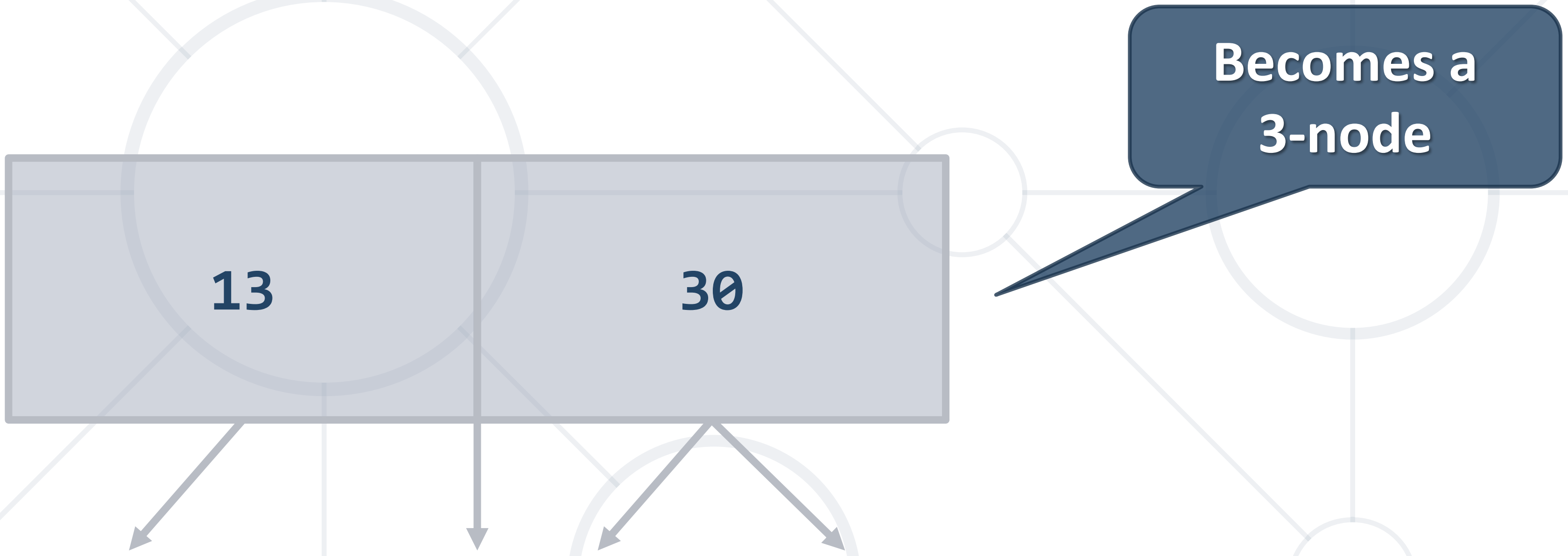
2-3 Tree Searching

Searching for 12

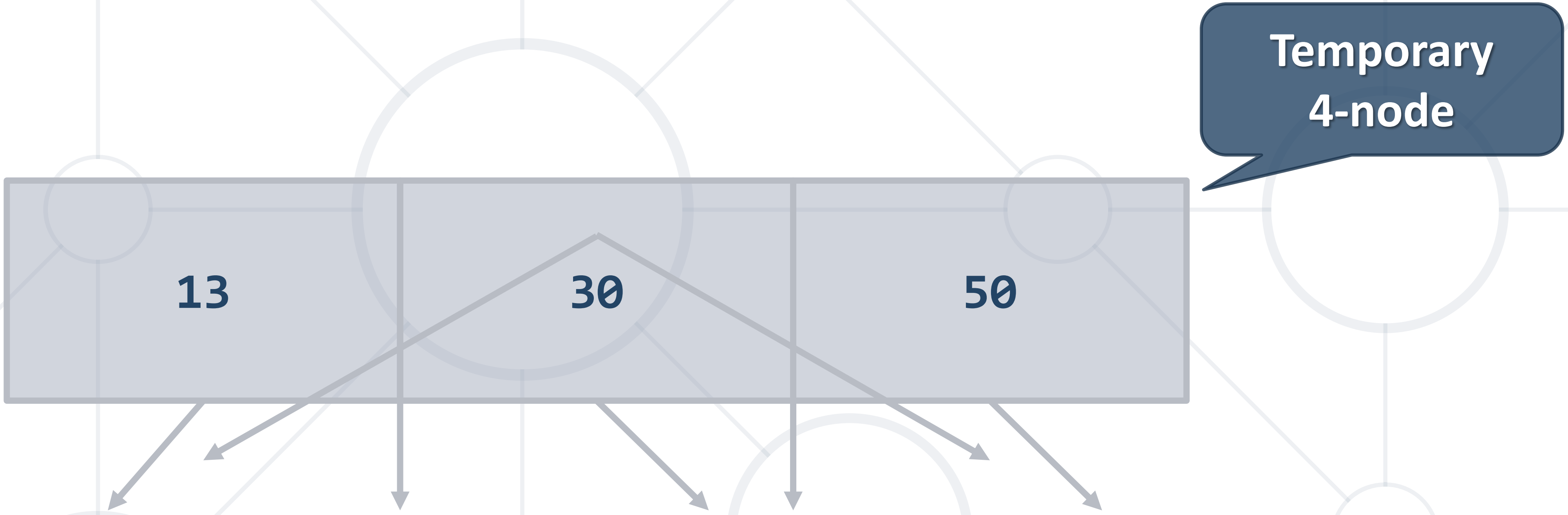
Identical to BST Search



2-3 Tree Insertion (at 2-node)

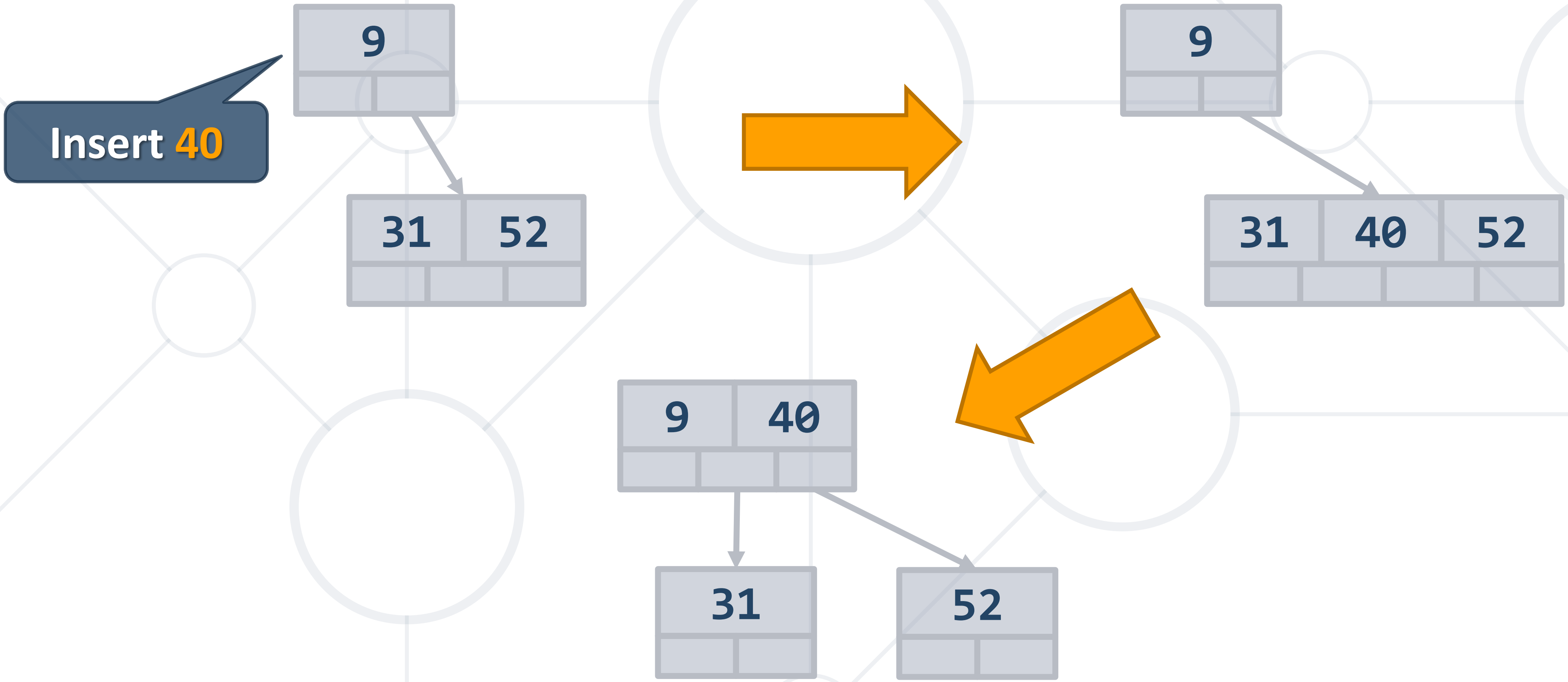


2-3 Tree Insertion (at 3-node)



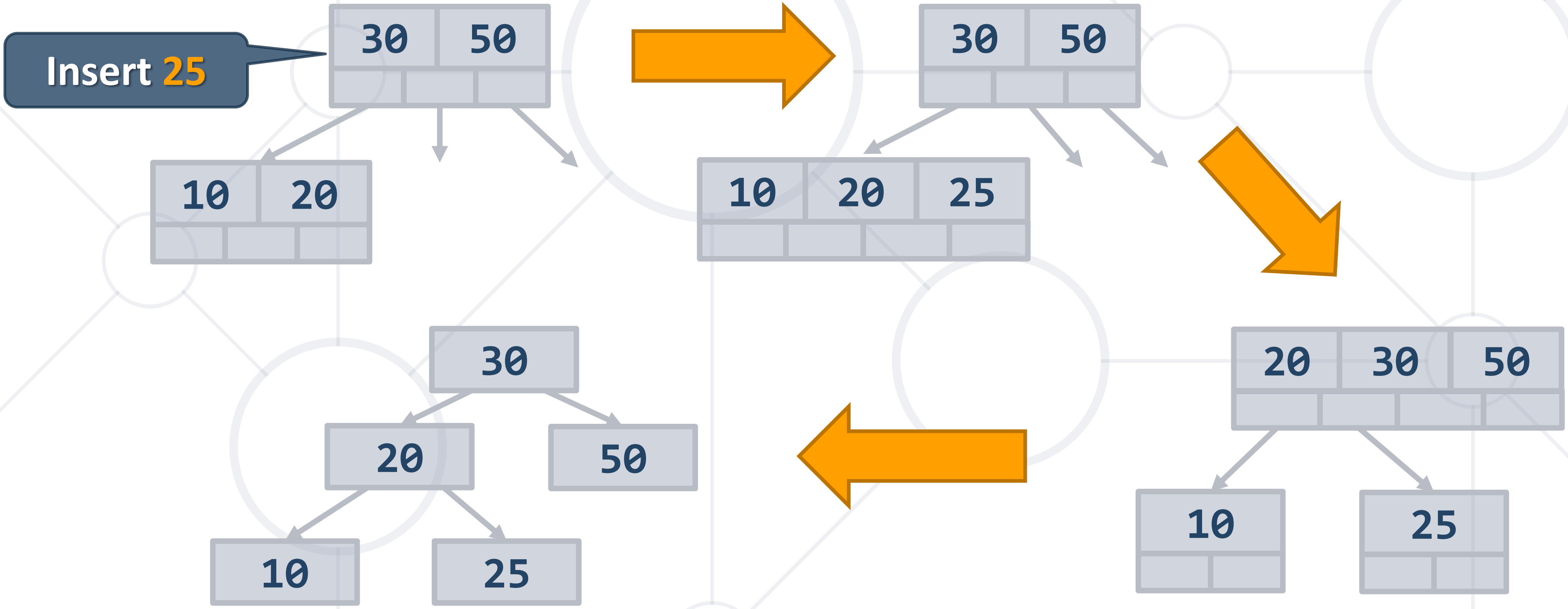
2-3 Tree Insertion

- Into a 3-node whose parent is a 2-node



2-3 Tree Insertion (2)

- Into a 3-node whose parent is a 3-node

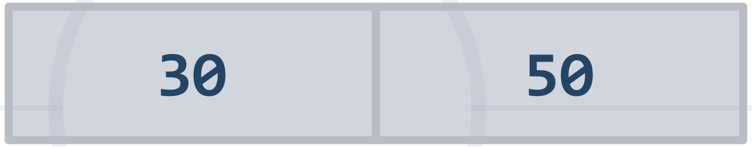


2-3 Tree Construction

- Insert **50**

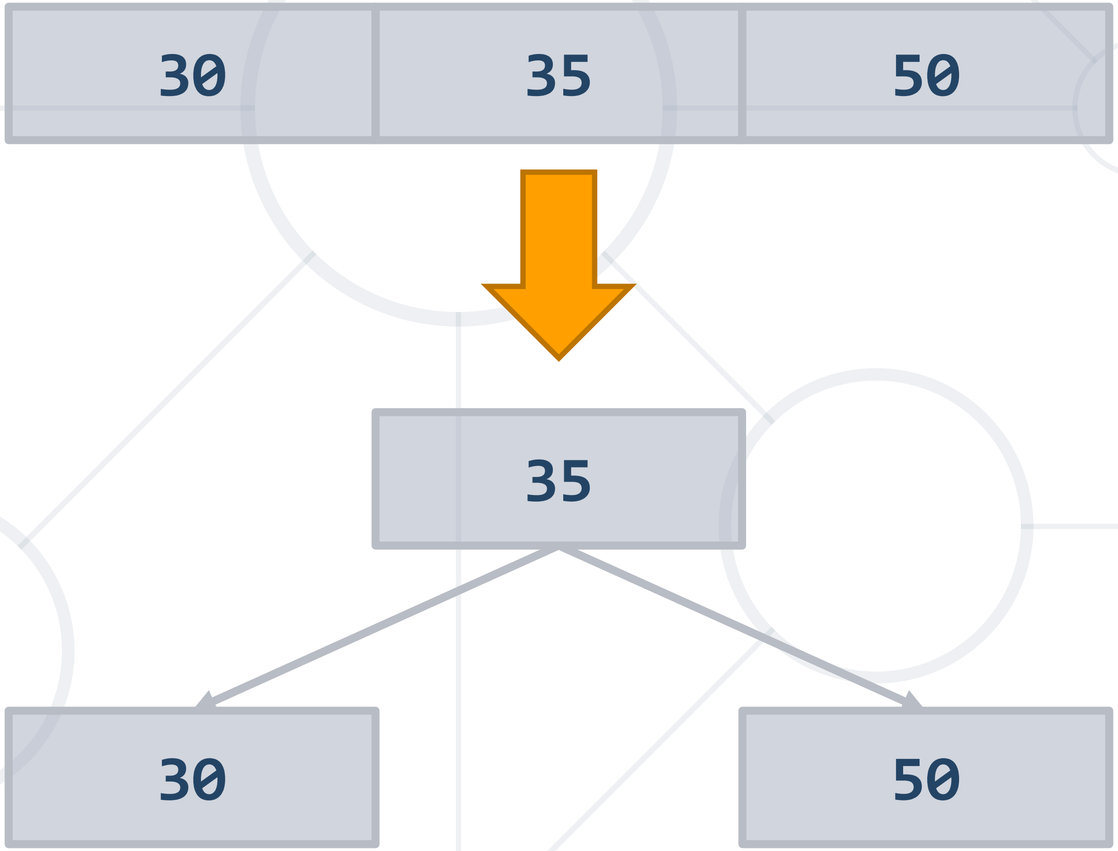
2-3 Tree Construction (2)

- Insert **30**



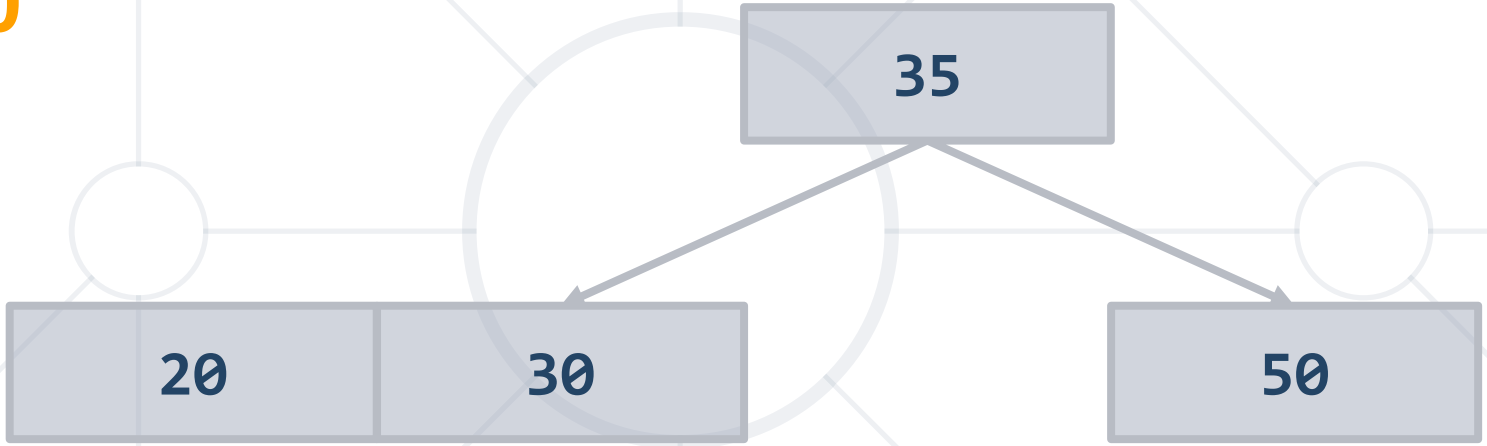
2-3 Tree Construction (3)

- Insert **35**



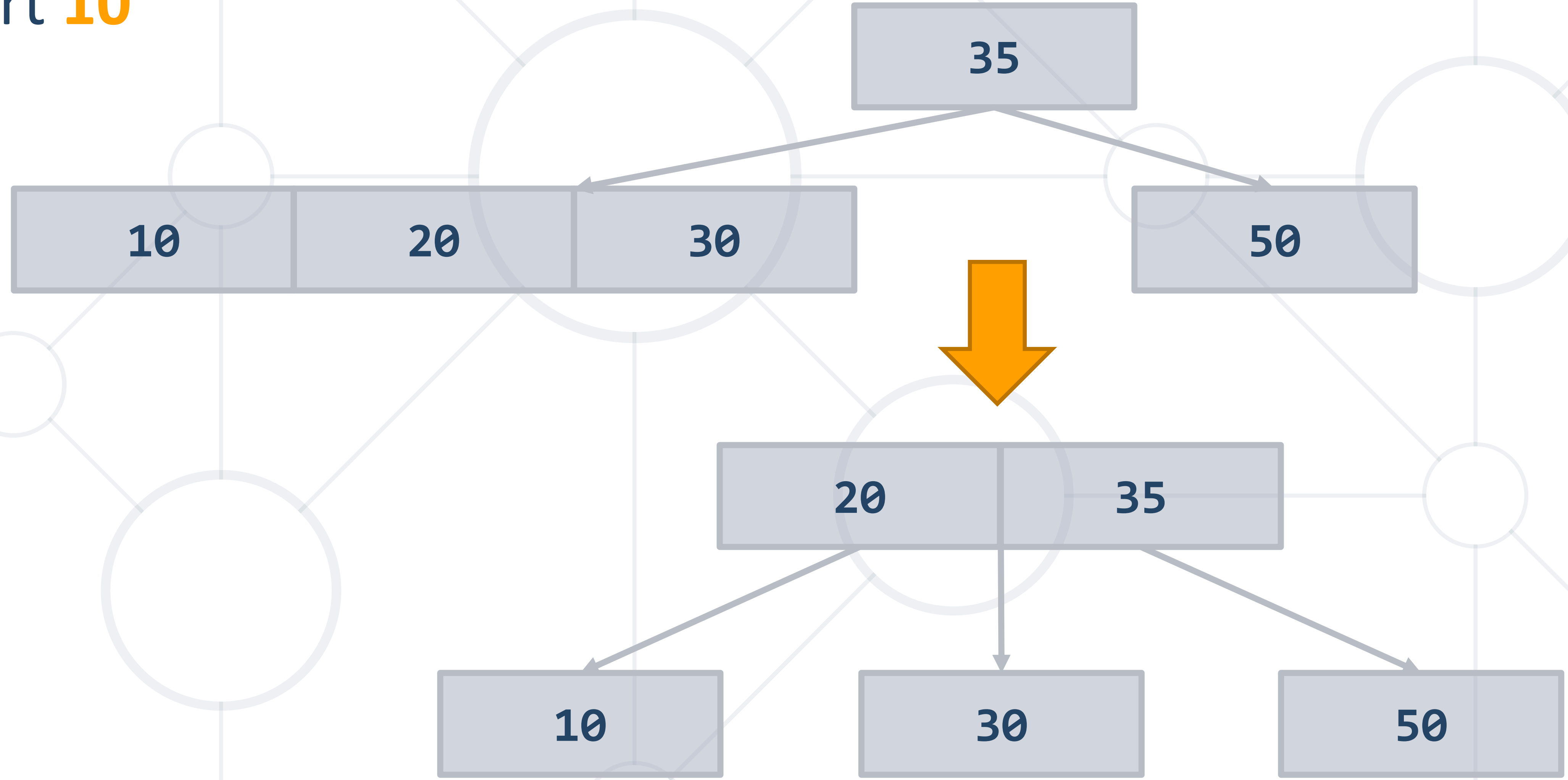
2-3 Tree Construction (4)

- Insert **20**



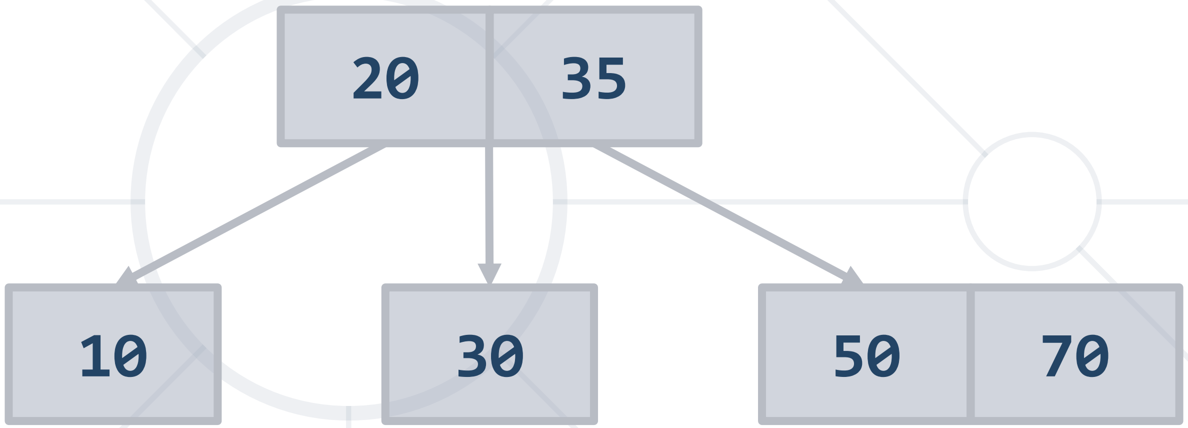
2-3 Tree Construction (5)

- Insert **10**



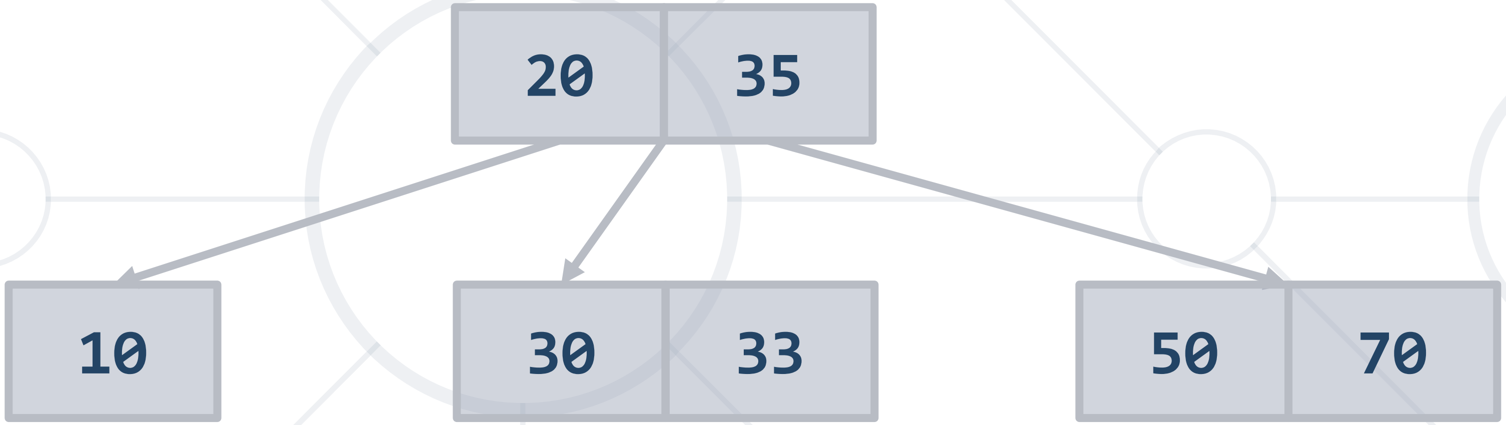
2-3 Tree Construction (6)

- Insert **70**



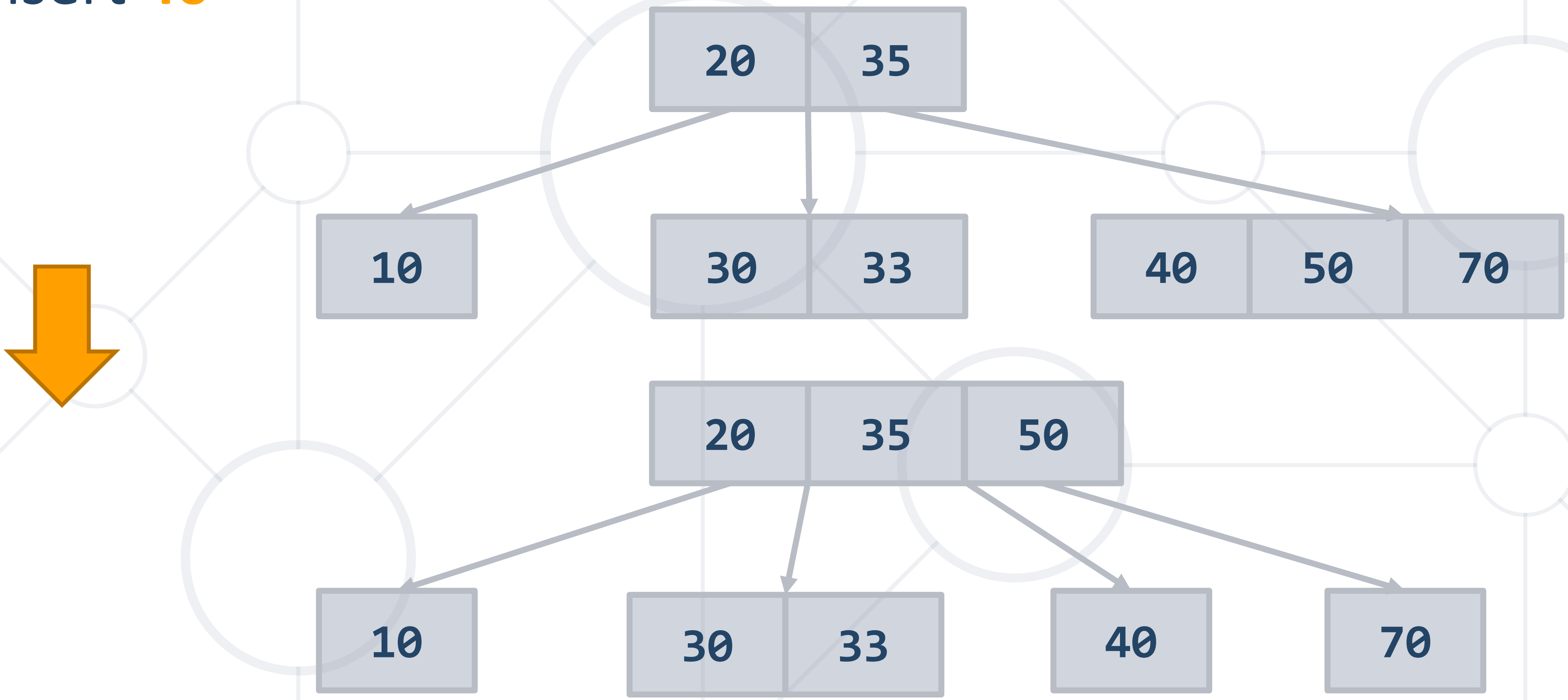
2-3 Tree Construction (7)

- Insert **33**

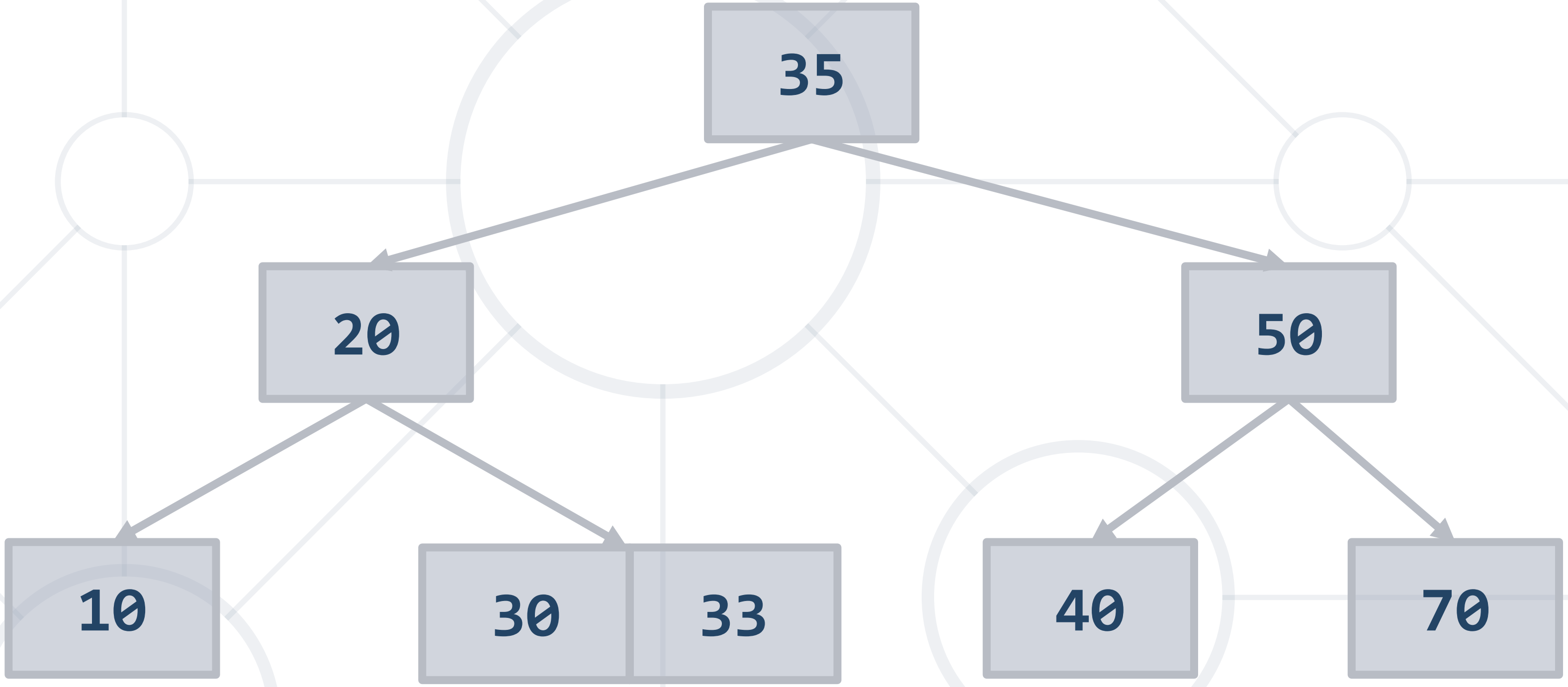


2-3 Tree Construction (8)

- Insert **40**



2-3 Tree Construction (9)



2-3 Tree Properties

- Unlike standard BSTs, 2-3 trees **grow from the bottom**
- The **number of links** from the root to any **null** node is the same
- Transformations are **local**
- Nearly **perfectly balanced**
- Inserting **10 nodes** will result with height of the tree **2**
  - For normal BSTs the height can be **9** in the worst case

2-3 Tree - Quiz

TIME'S

- Suppose that you are inserting a new node to a 2-3 tree. Under which of the following scenarios must the height of the 2-3 tree increase by one?

  A. Number of nodes is equal to power of 2

  B. Number of nodes is one less than a power of 2

  C. When the final node on a search path from the root is a 3-node

  D. When every node on the search path from the root is a 3-node

2-3 Tree - Answer

- Suppose that you are inserting a new node to a 2-3 tree. Under which of the following scenarios must the height of the 2-3 tree increase by one?

A. Number of nodes is equal to power of 2

B. Number of nodes is one less than a power of 2

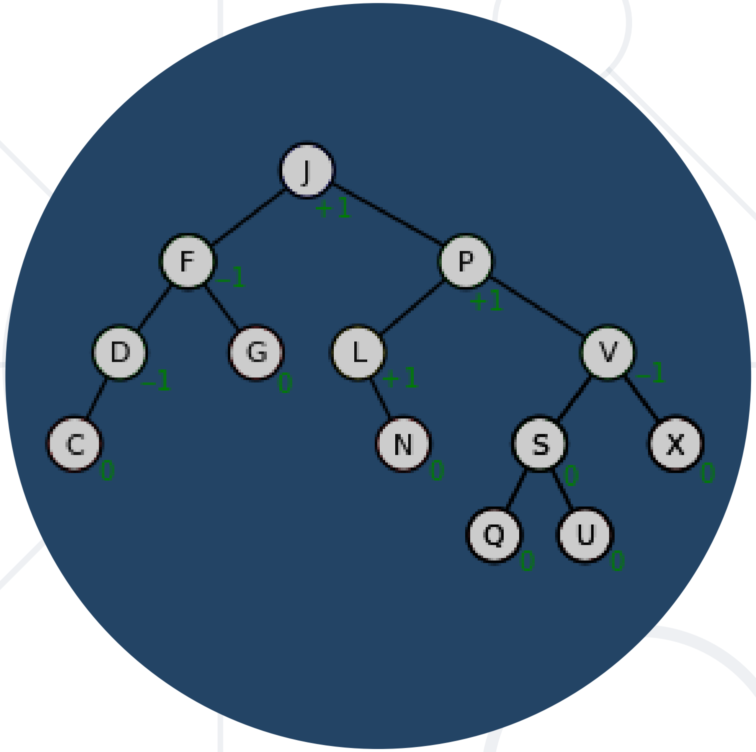
C. When the final node on a search path from the root is a 3-node

D. When every node on the search path from the root is a 3-node

2-3 Tree - Summary

Structure	Worst case			Average case	
	Search	Insert	Delete	Search Hit	Insert
BST	N	N	N	1.39 lg N	1.39 lg N
2-3 Tree	c lg N	c lg N	c lg N	c lg N	c lg N

Constants depend on implementation

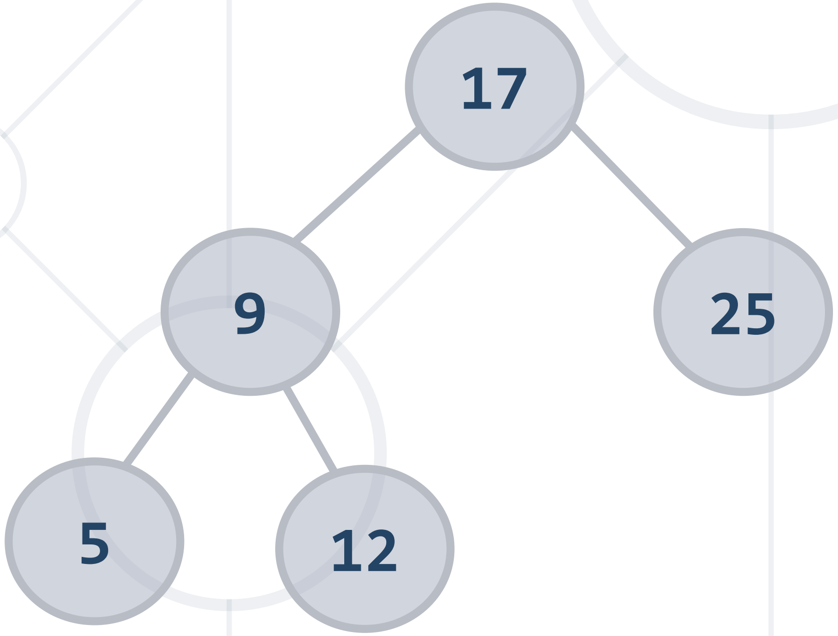


AVL Trees

Properties and Rotations

AVL Tree

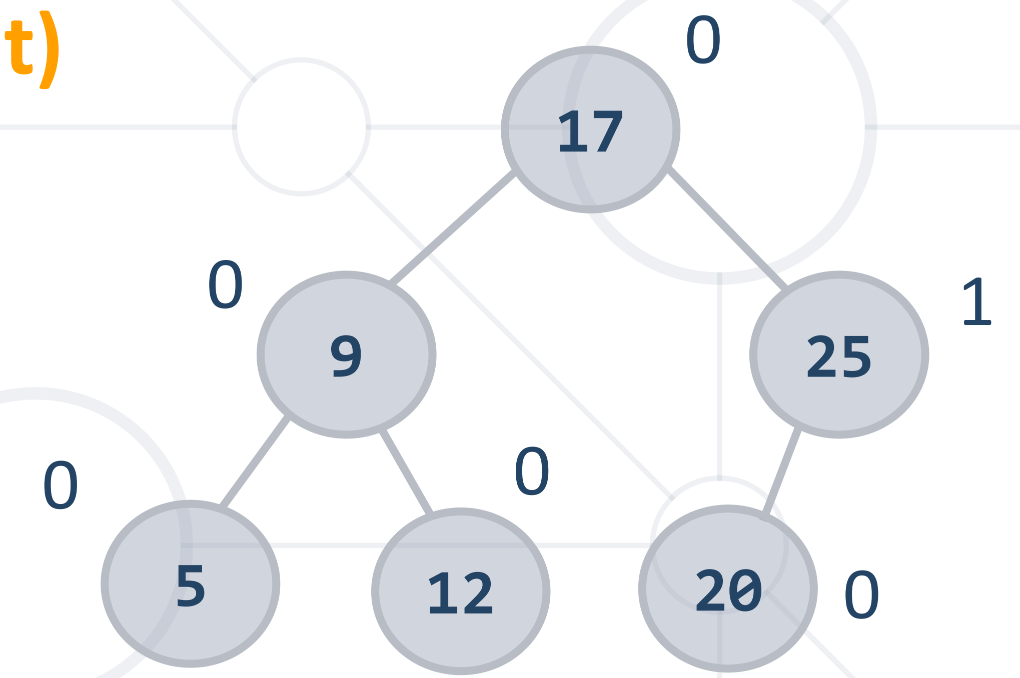
- AVL tree is a self-balancing binary-search tree (visualization)
  - Height of two subtrees can **differ by at most 1**



	Average	Worst case
Space	O(n)	O(n)
Search	O(log n)	O(log n)
Insert	O(log n)	O(log n)
Delete	O(log n)	O(log n)

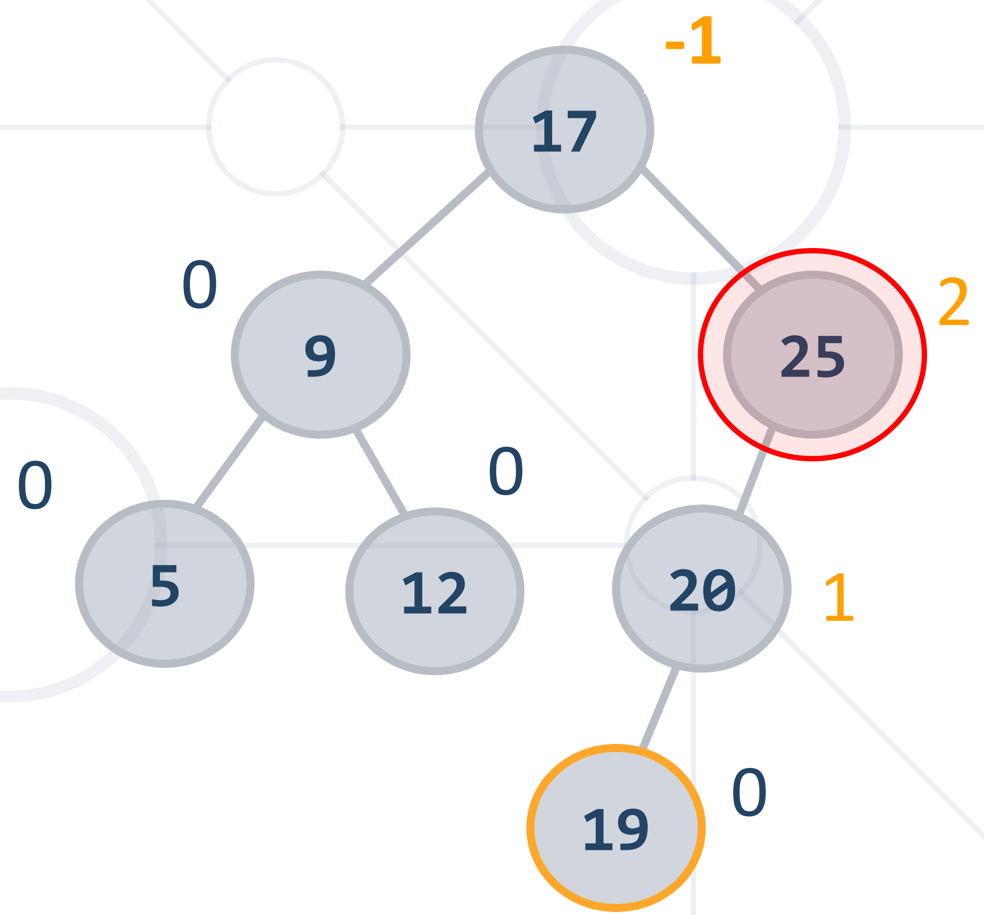
AVL Tree Rebalancing

- Height difference is measured by a balance factor (BF)
- **BF(Tree) = Height(Left) – Height(Right)**
  - BF of any node is in the range **[-1, 1]**
  - If BF becomes **-2** or **2** → rebalance



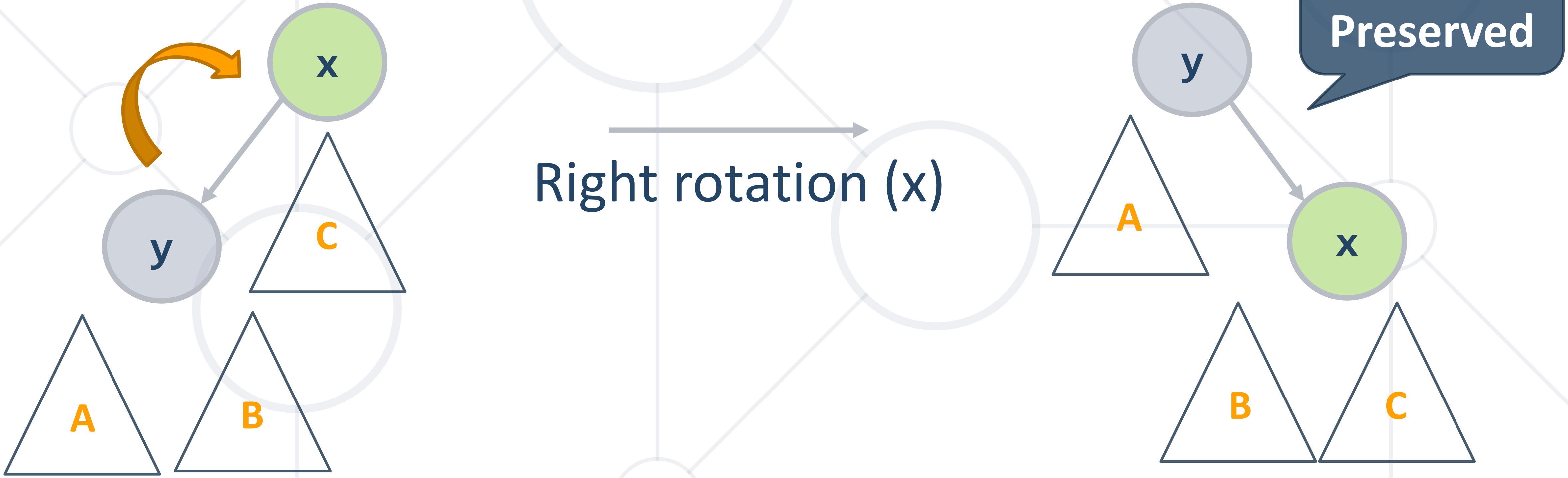
AVL Tree Rebalancing

- Rebalancing is done by **retracing**
  - Start from inserted node's parent and go up to root
  - Perform **rotations** to restore balance



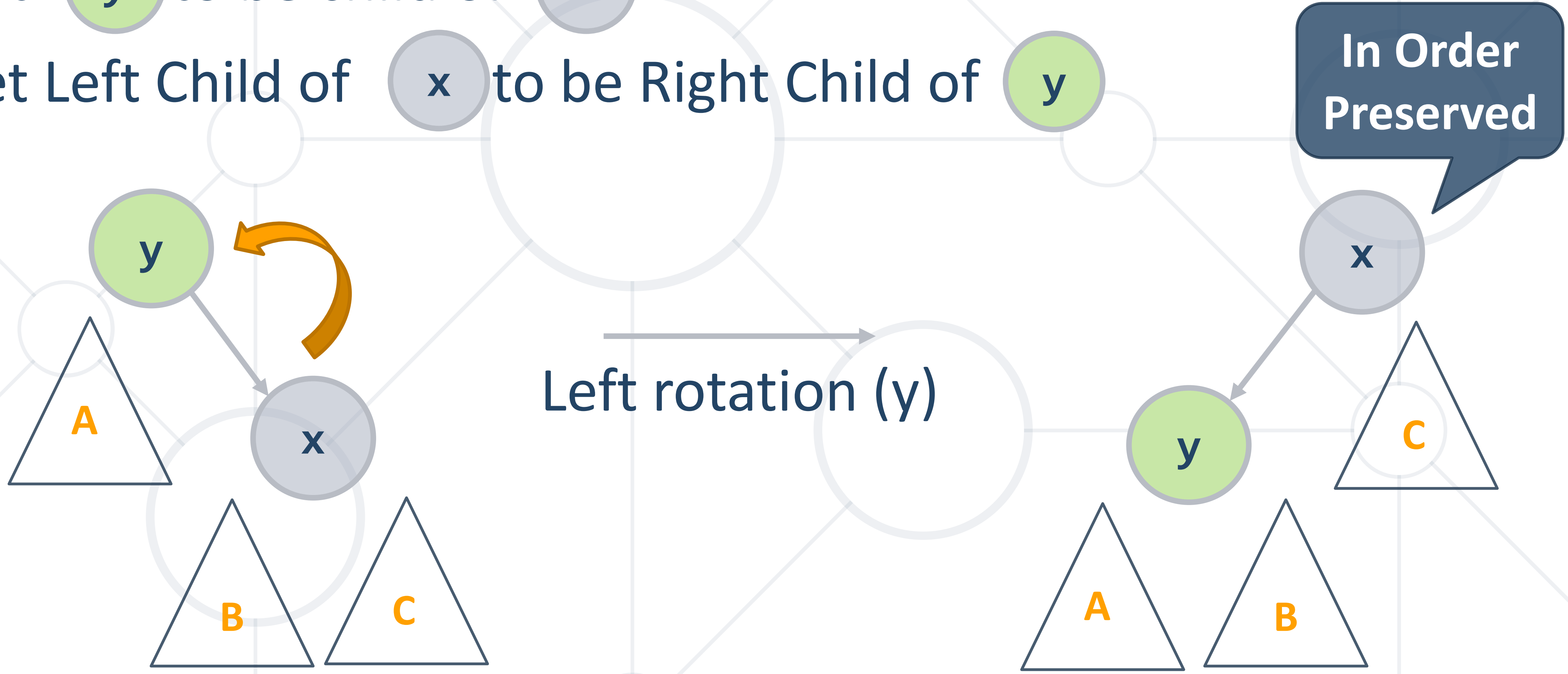
Right Rotation

- Set x to be child of y
- Set Right Child of y to be Left Child of x



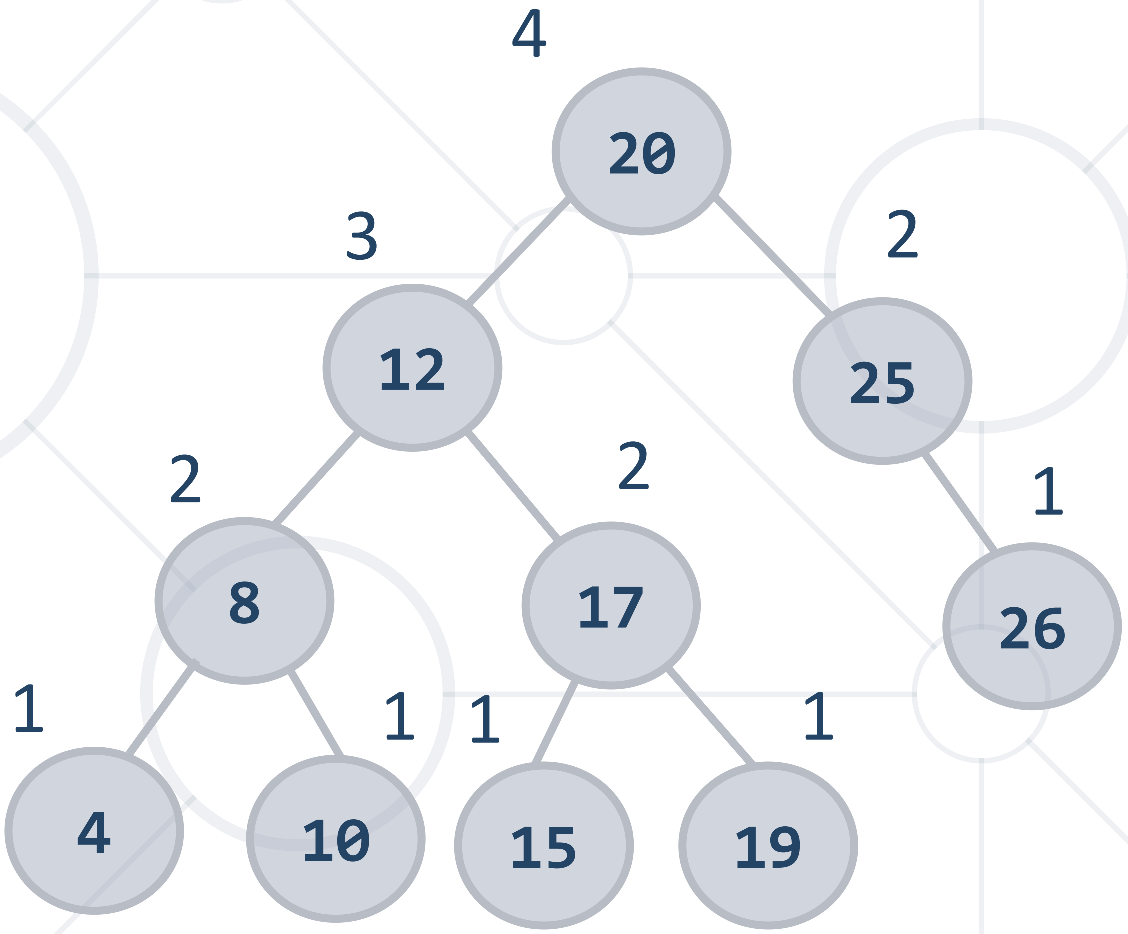
Left Rotation

- Set y to be child of x
- Set Left Child of x to be Right Child of y



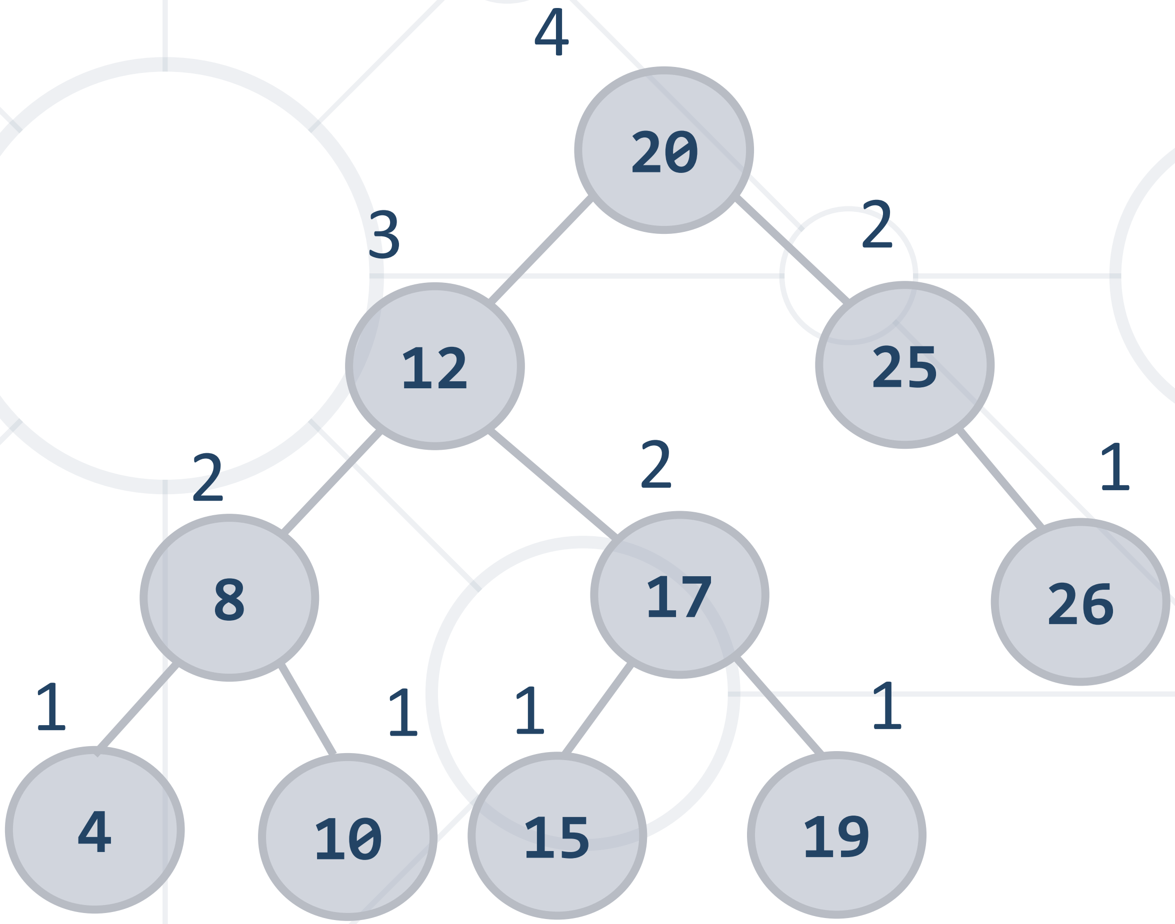
AVL Tree Insertion Algorithm

- Insert like in **ordinary BST**
- Retrace **up** to root
  - Modify balance / height
  - If balance factor **∉ [-1,1]** → rebalance



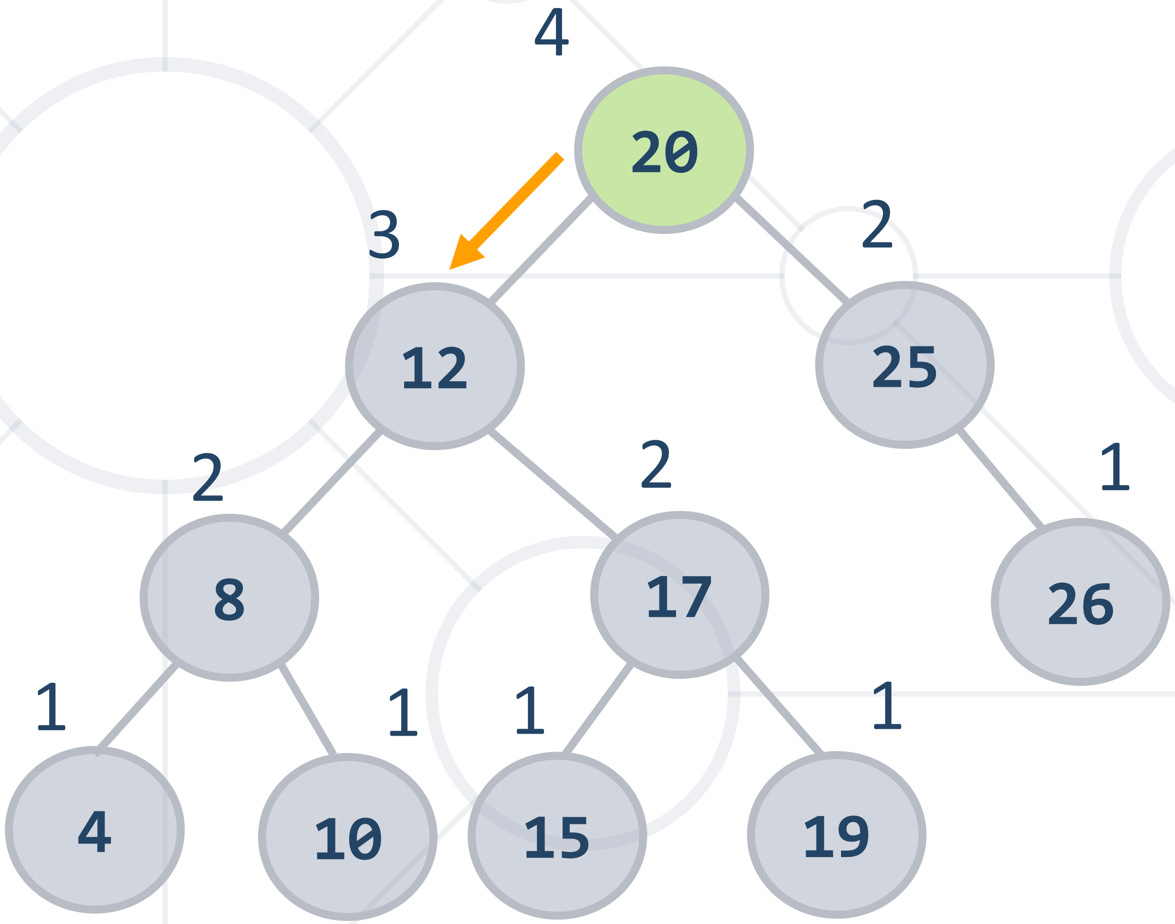
Insertion - #1

- Insert **11**



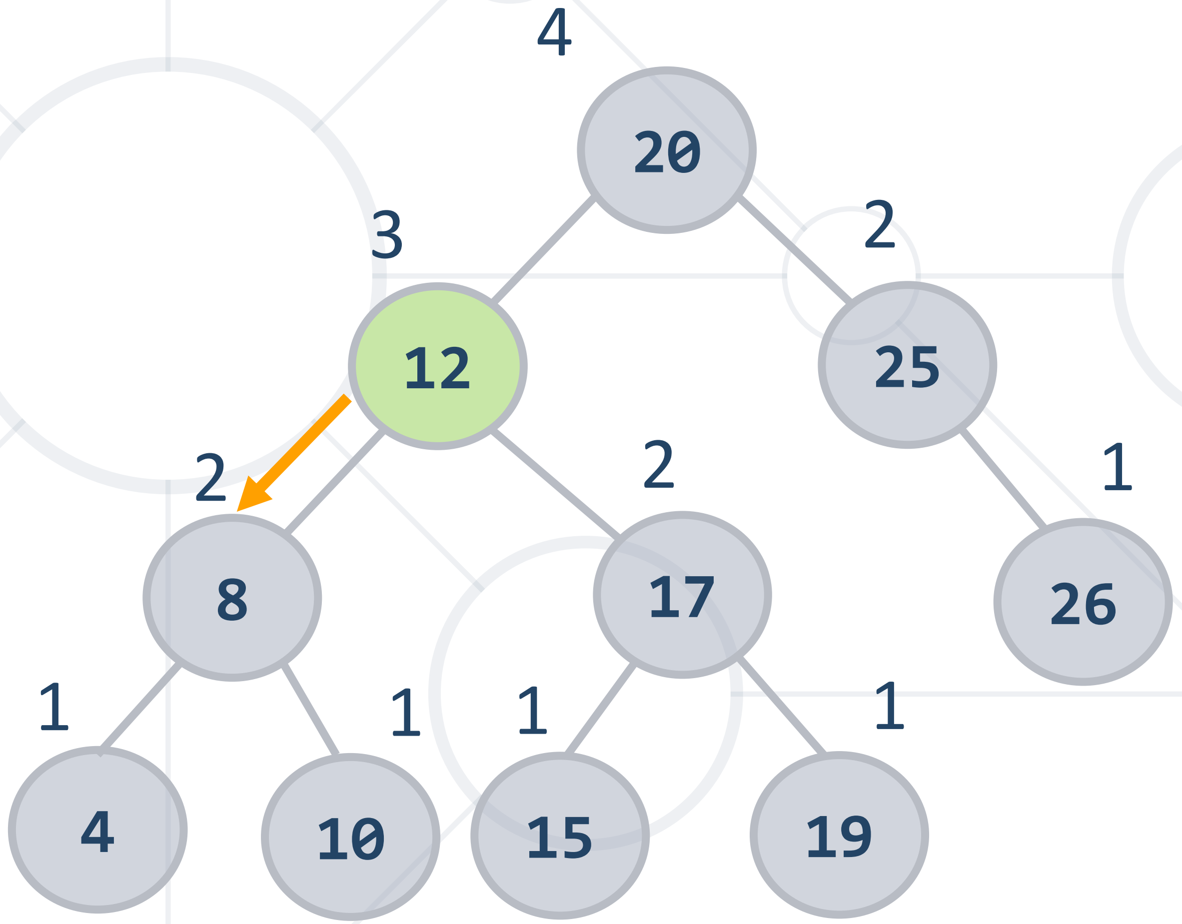
Insertion - #2

- 11 < 20 → go left



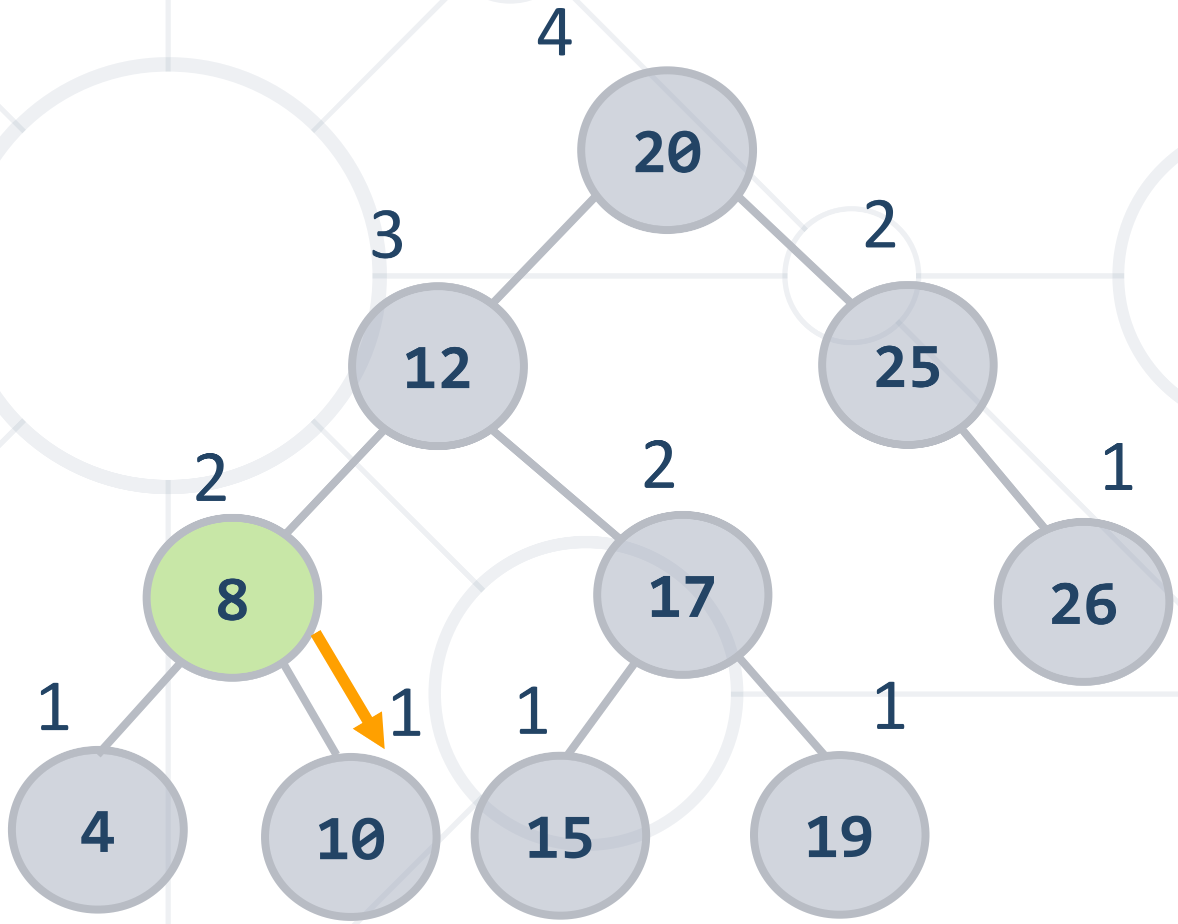
Insertion - #3

- 11 < 12 → go left



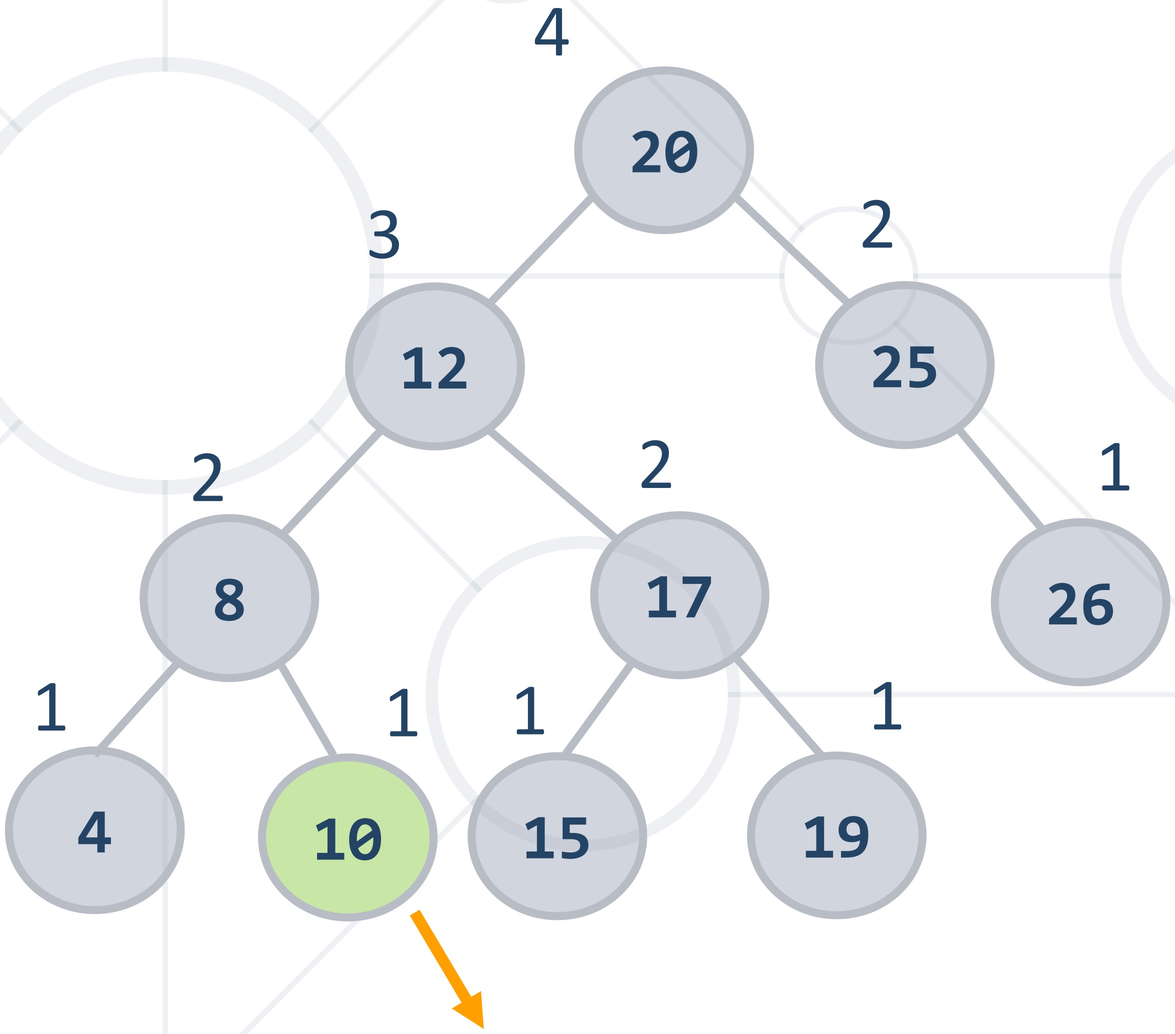
Insertion - #4

- 11 > 8 → go right



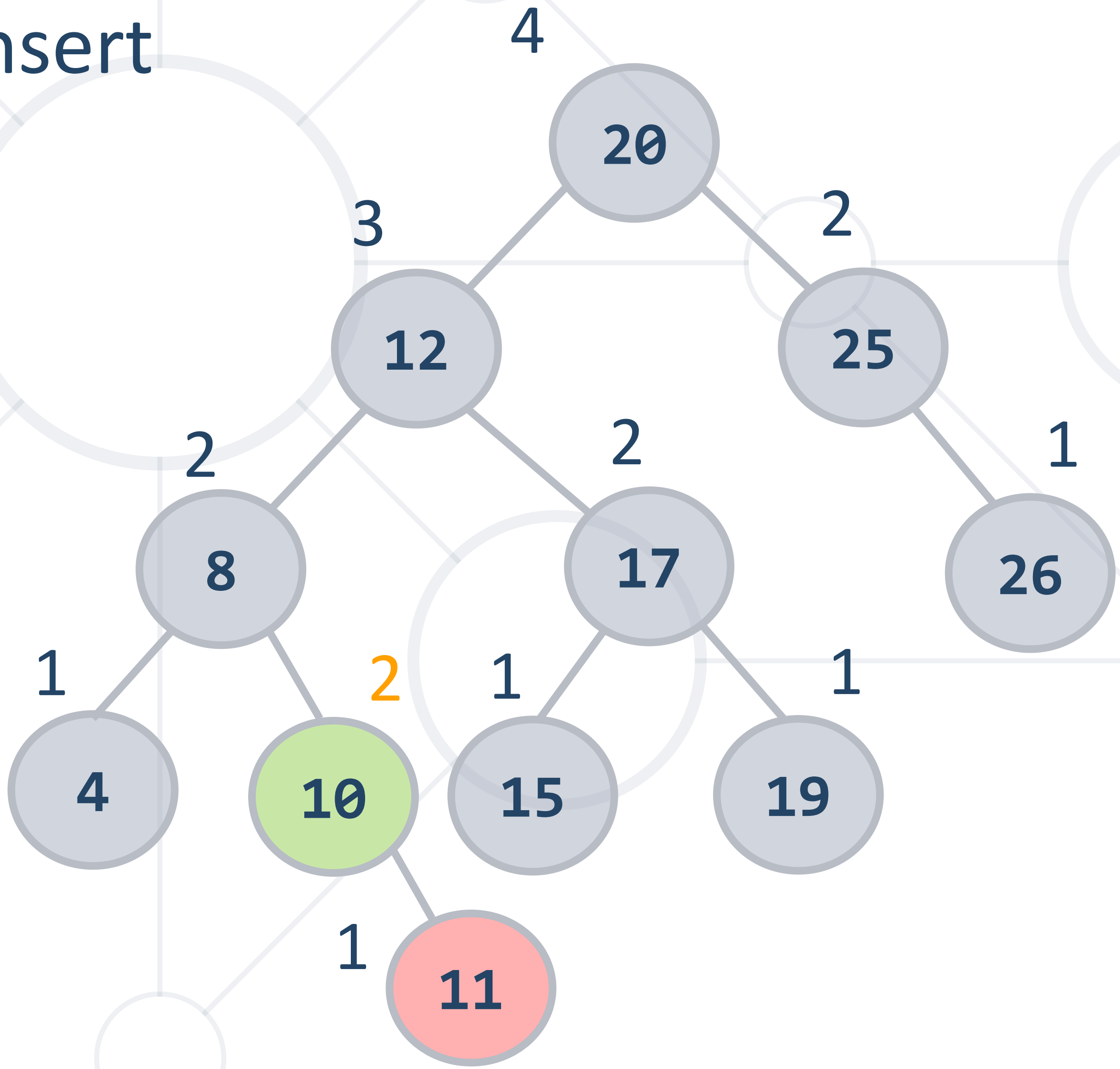
Insertion - #5

- 11 > 10 → go right



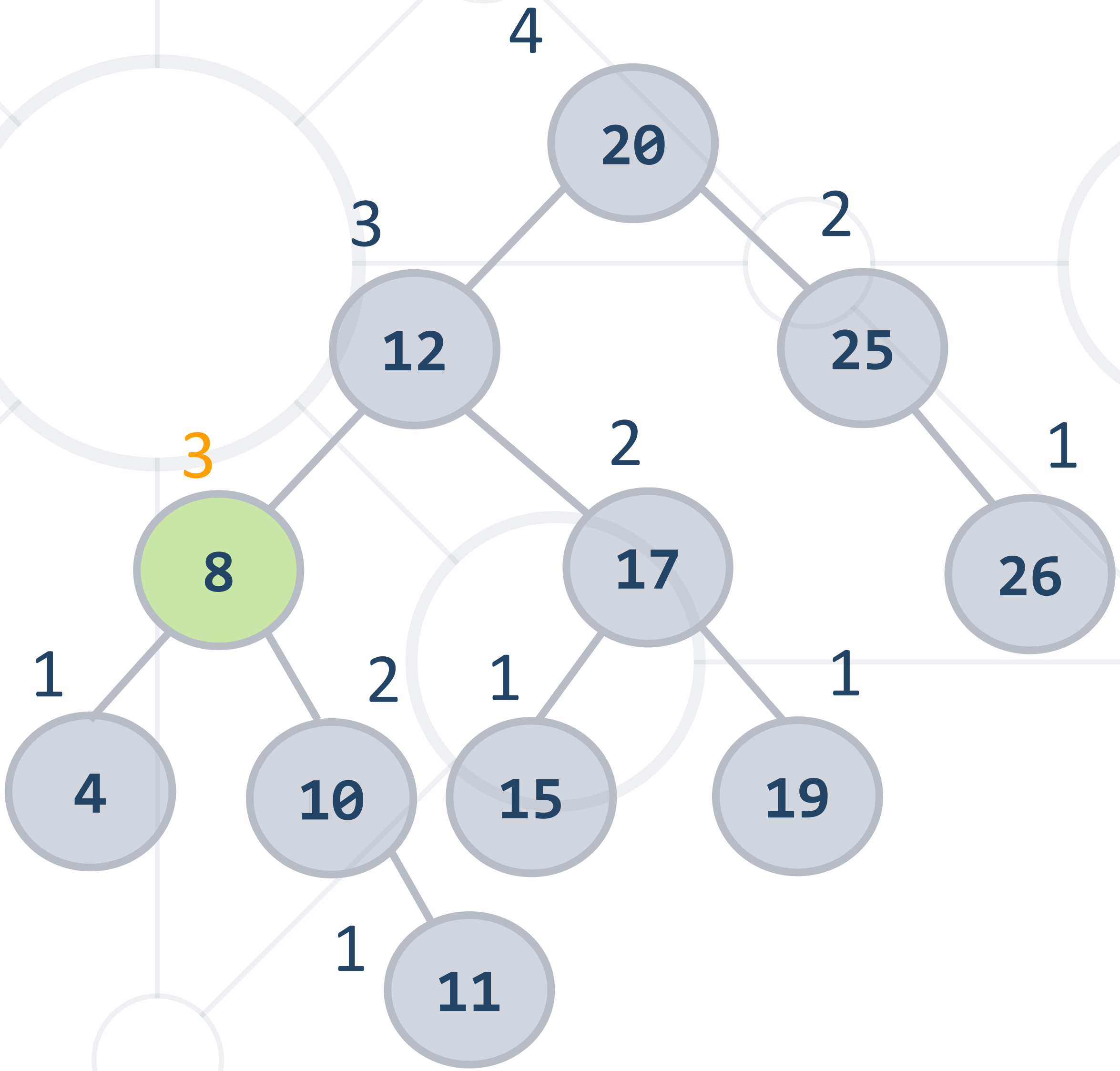
Insertion - #5

- Right node is **null** → insert
- Update 10 height
- 10 balance is **-1**



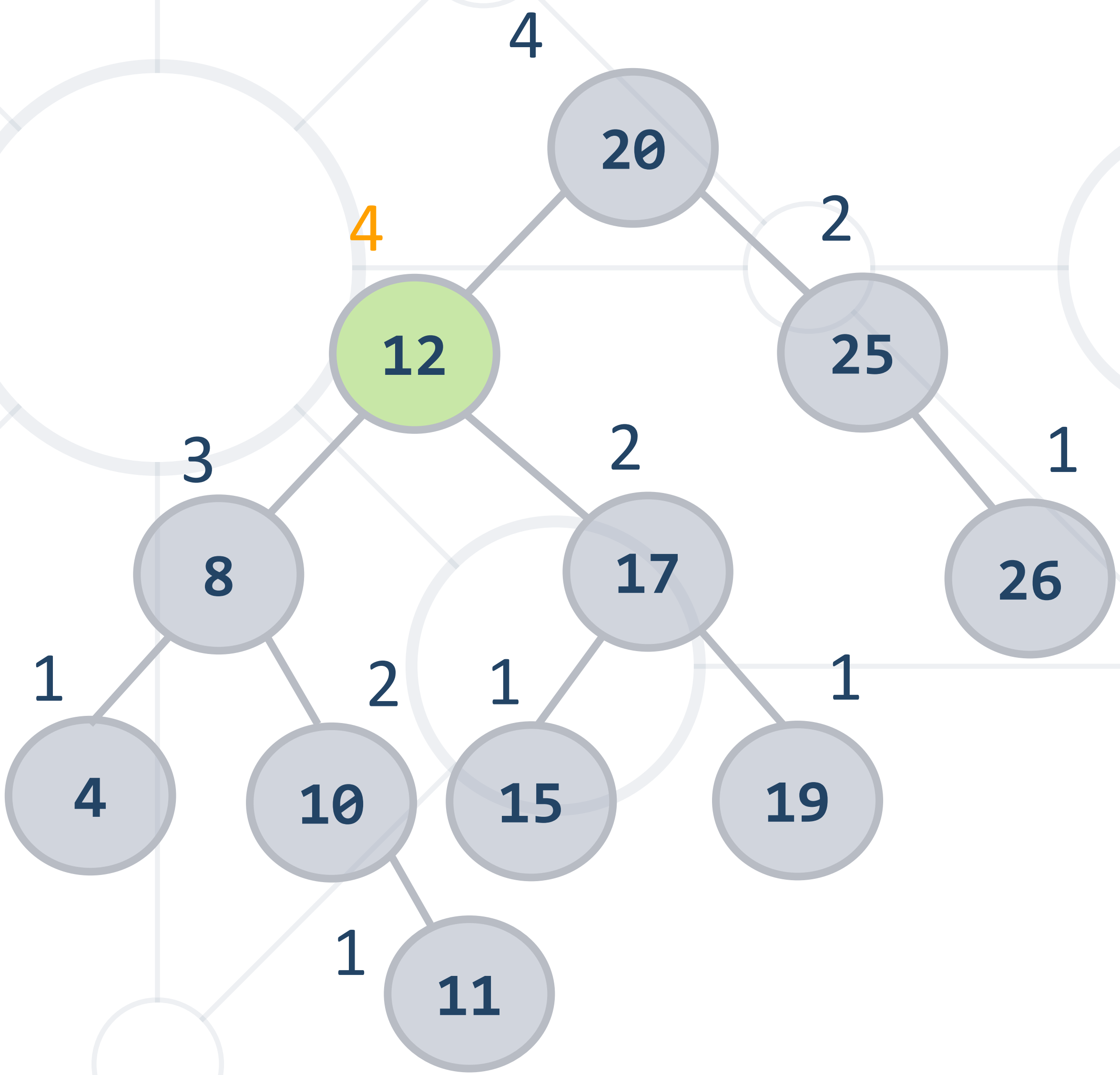
Insertion - #5

- Update 8 height
- 8 balance is -1



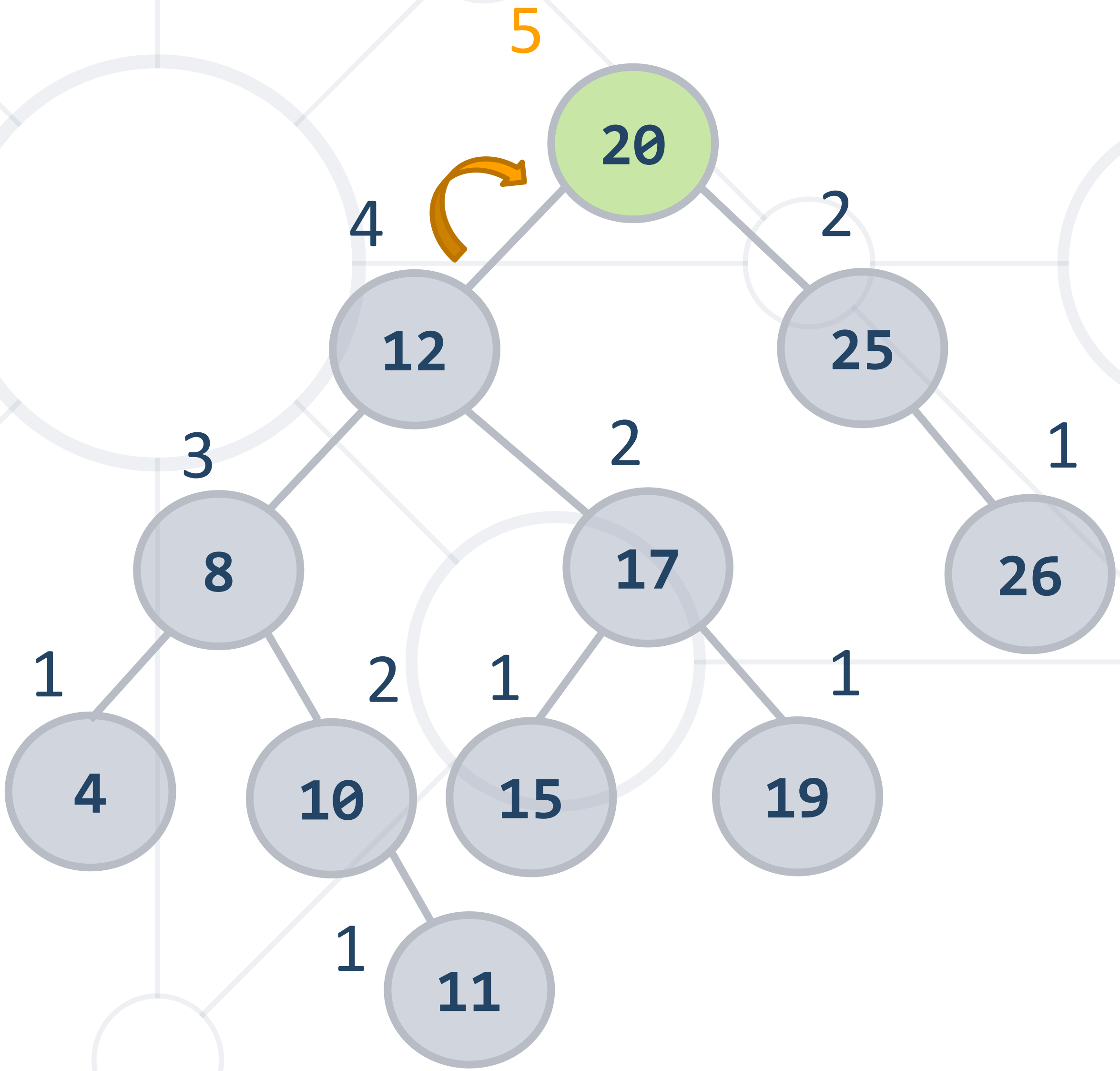
Insertion - #5

- Update 12 height
- 12 balance is 1



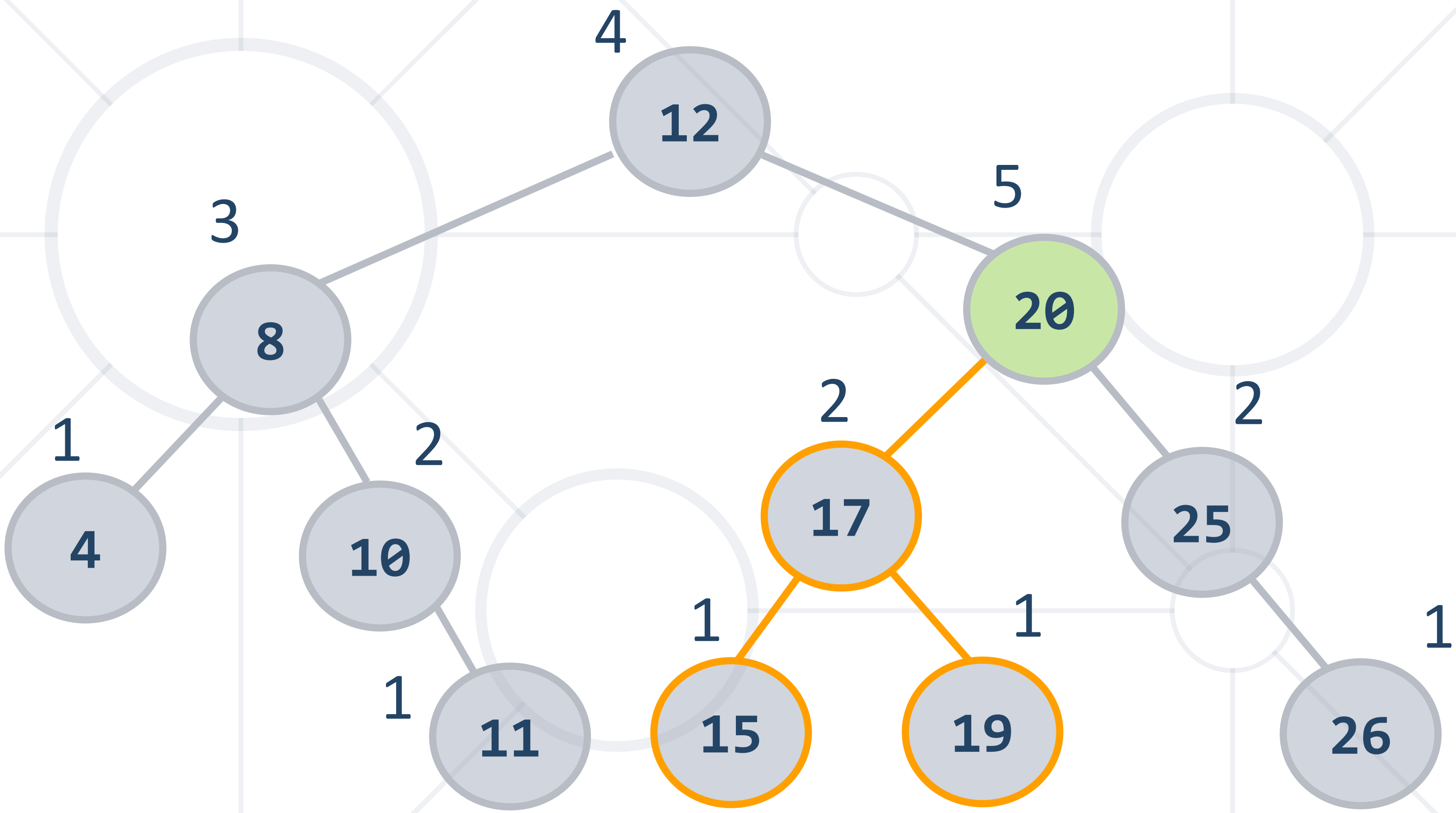
Insertion - #5

- Update 20 height
- 20 balance is **2**
- 20 is left heavy
- rotate 20 right



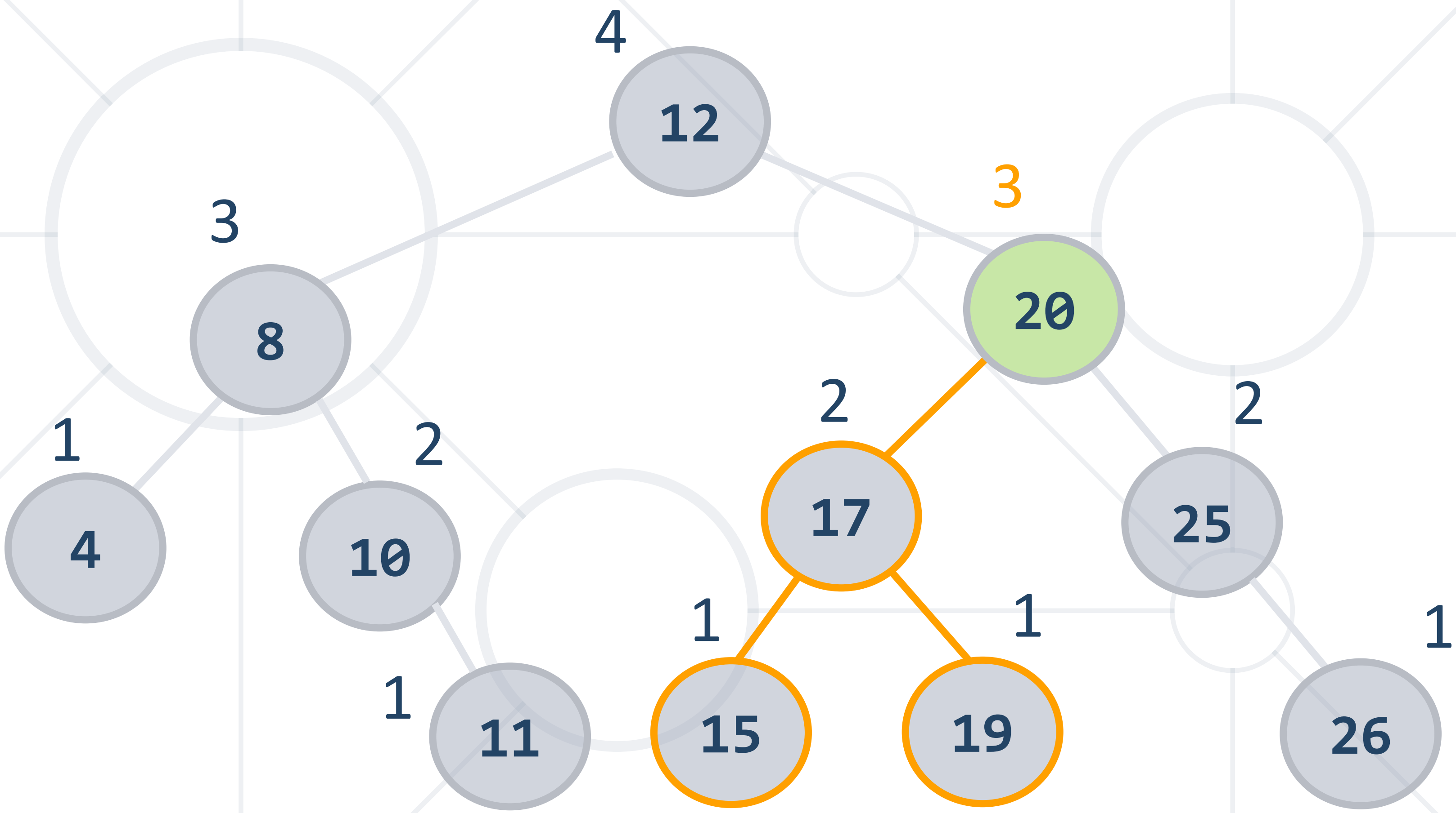
Insertion - #5

- 17 Switch parent



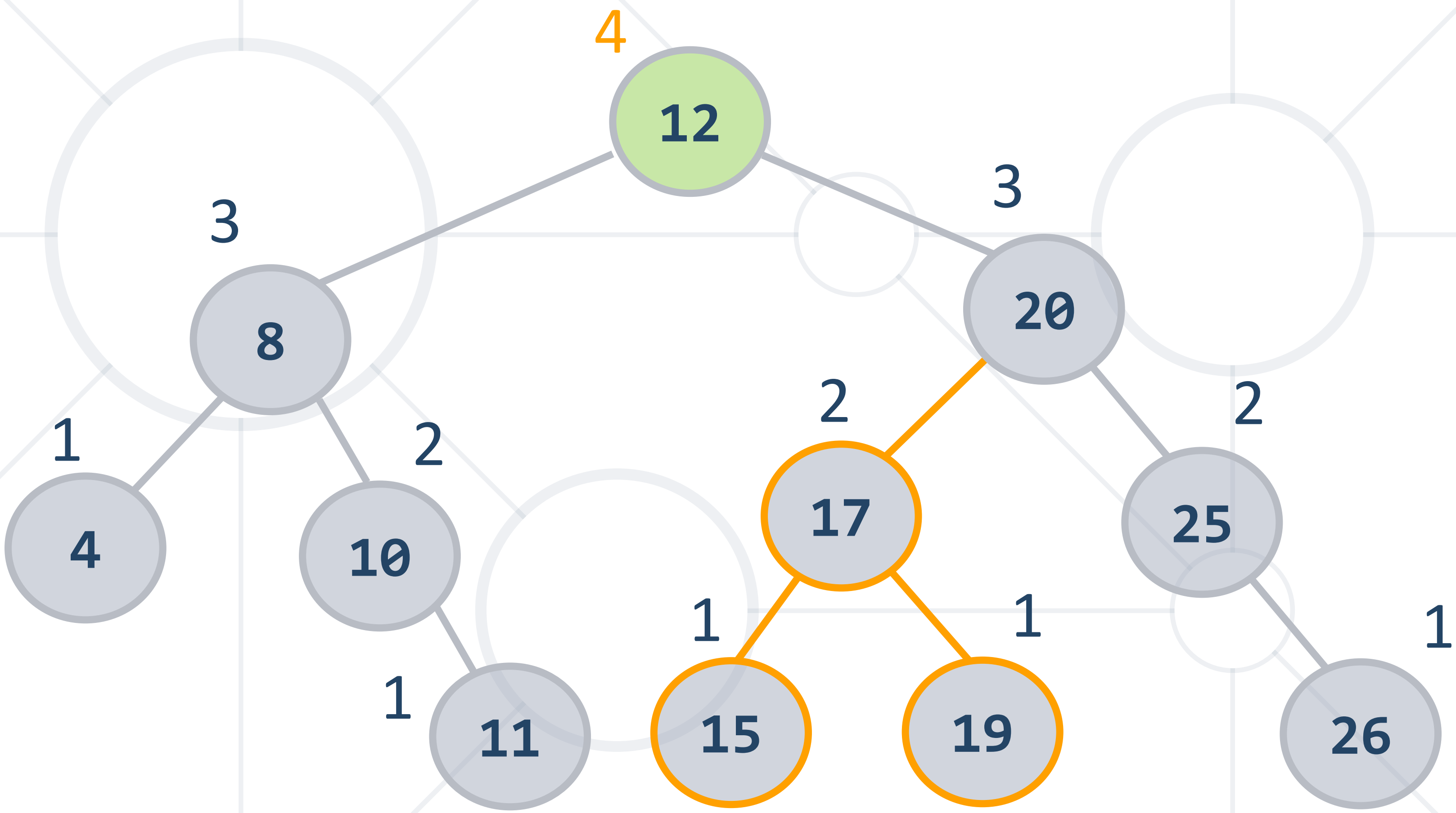
Insertion - #5

- 20 Update height



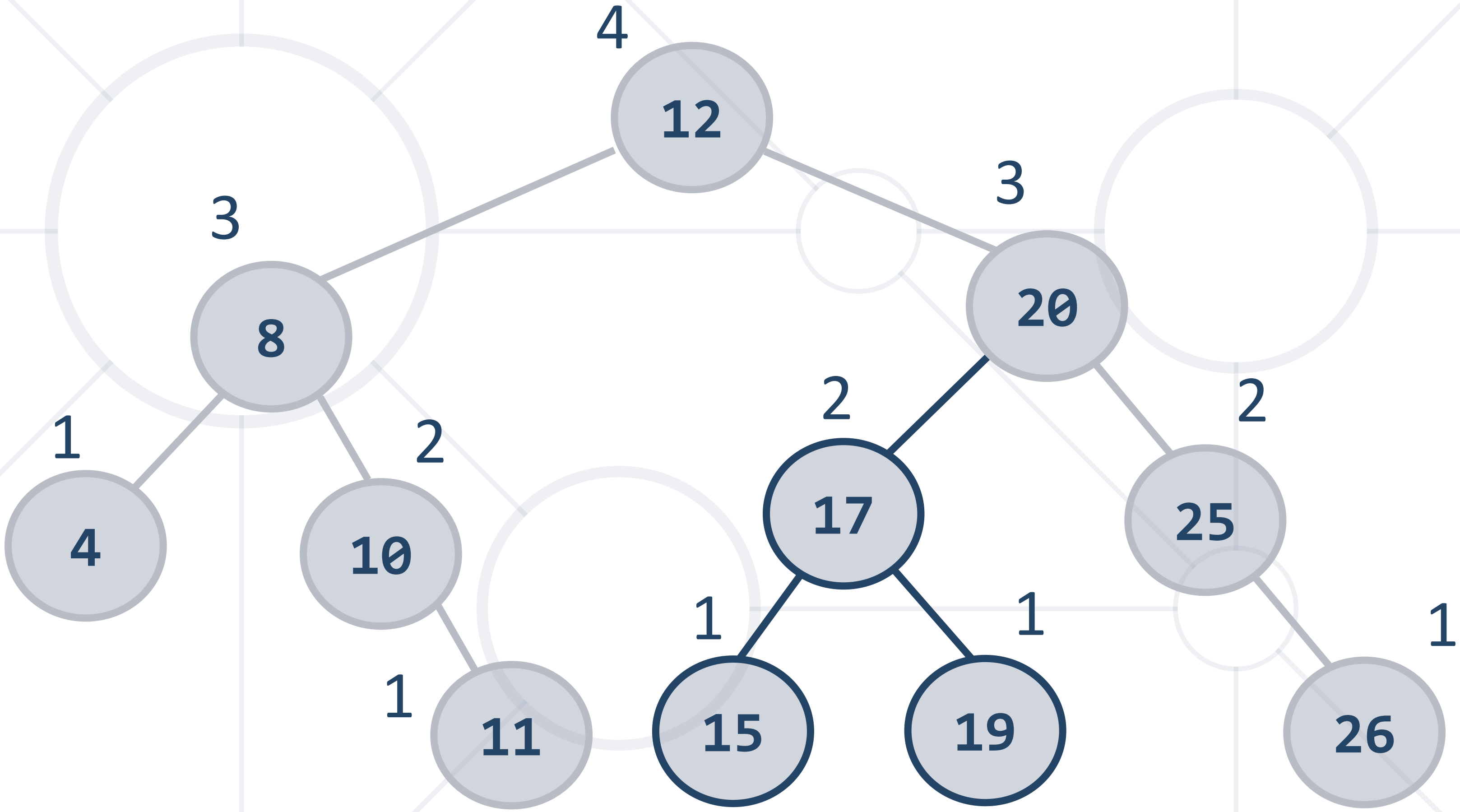
Insertion - #5

- 12 Update height
- 12 Balance is 0



Insertion - #5

- Over

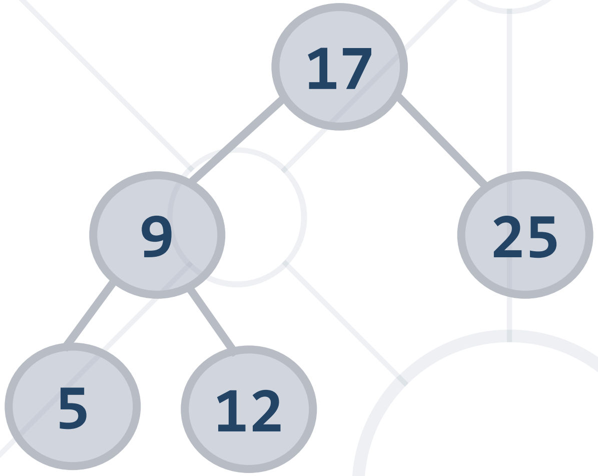


- More: https://en.wikipedia.org/wiki/AVL_tree

AVL Tree - Quiz

TIME'S

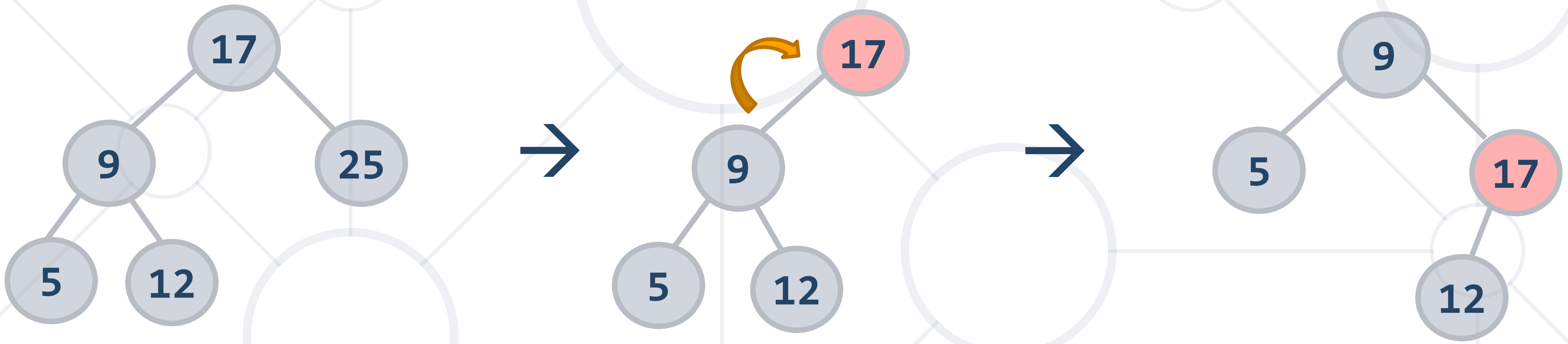
- Delete **25**. What will be the **resulting tree**?

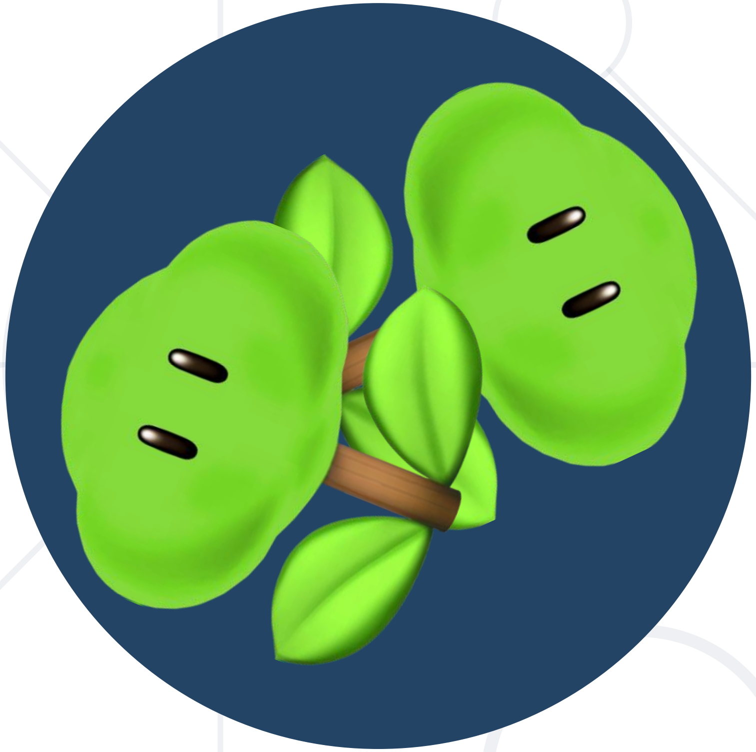


AVL Tree - Quiz

TIME'S UP!

- Delete **25**. What will be the **resulting tree**?



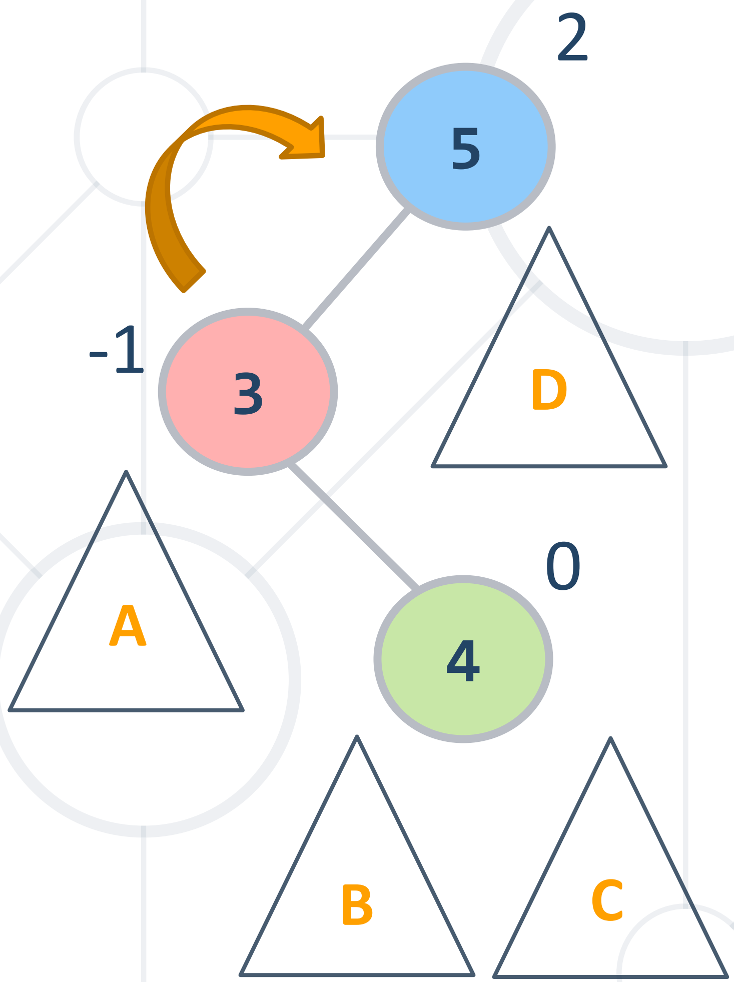


Double Rotations

Double Left, Double Right Rotation

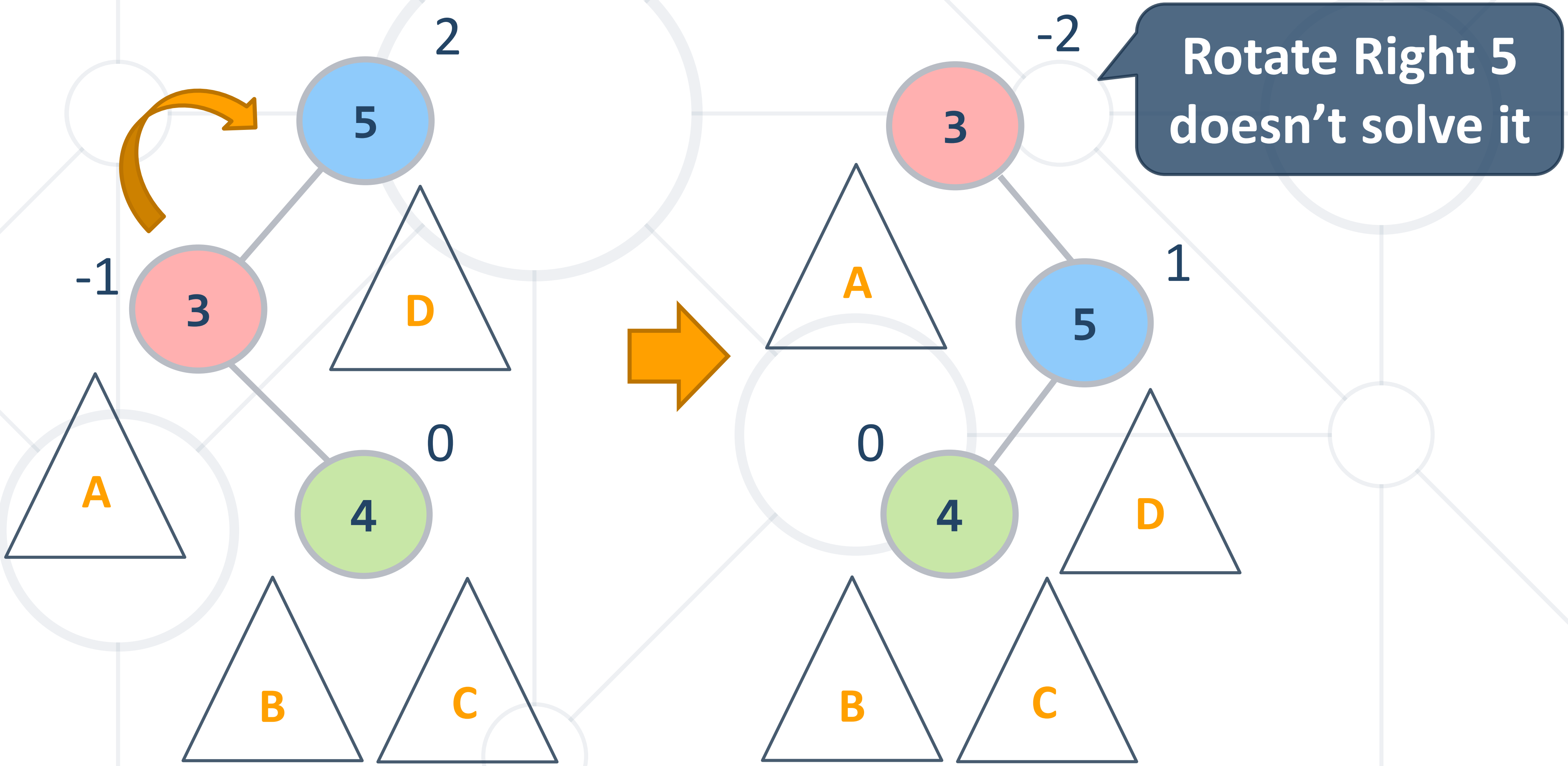
Single Rotation Problem

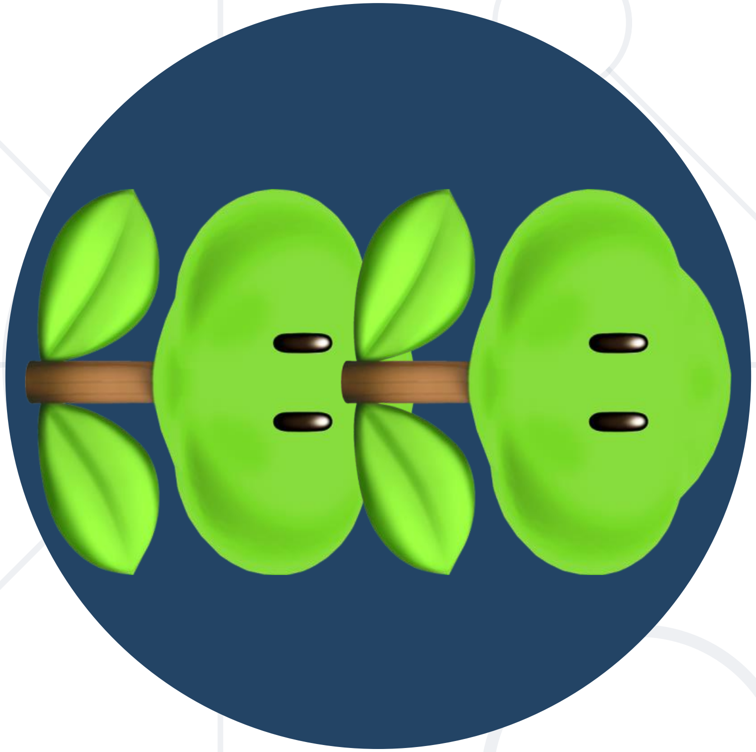
- Insert **4**



Single Rotation Problem (2)

- Rotate a node with opposite balanced child



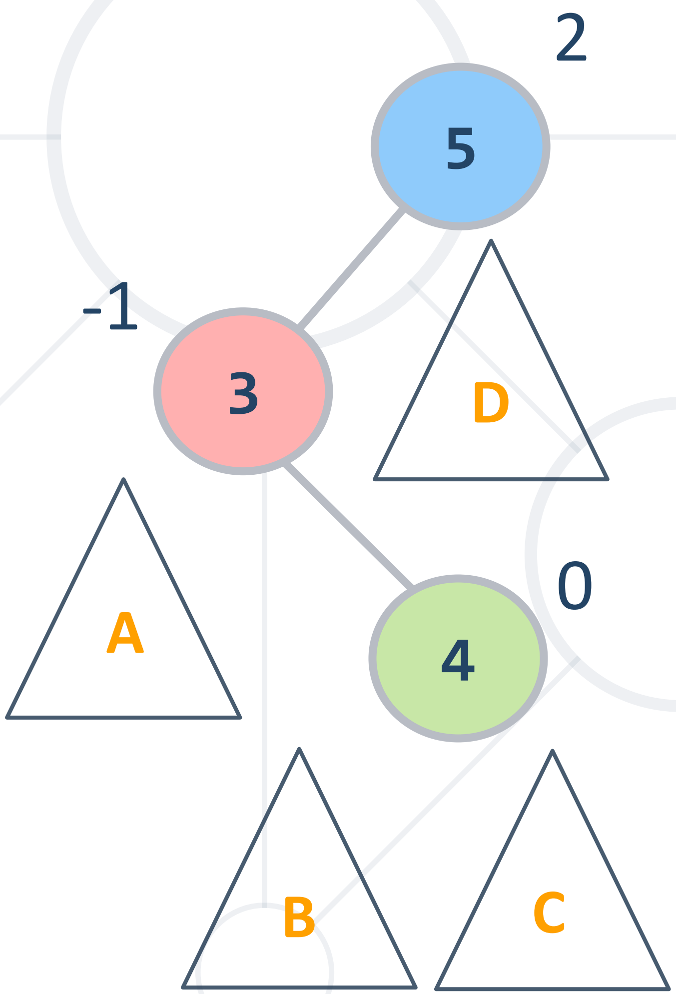


Double Right Rotation

Right-Left

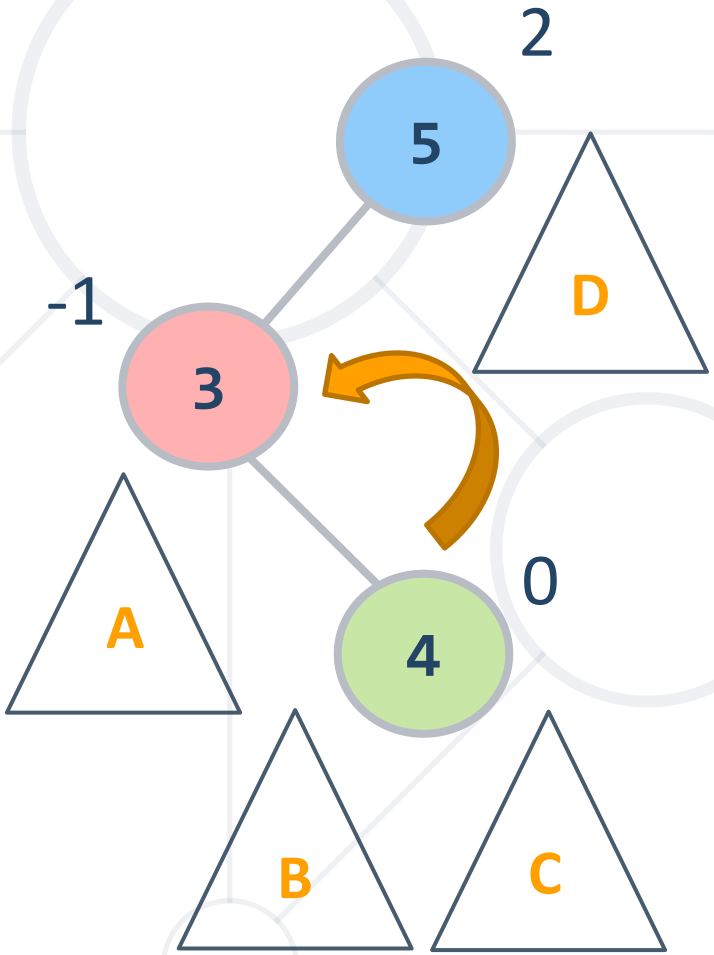
Double Right Rotation

- Rotate Right (node) with negatively balanced Left Child



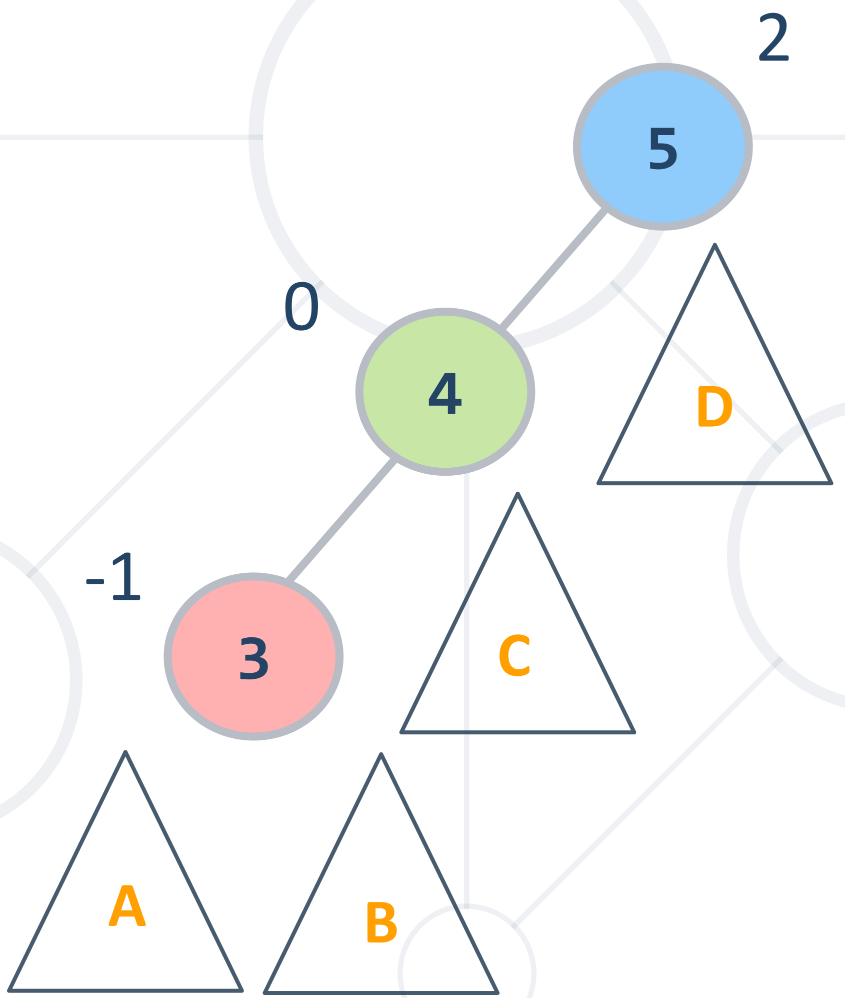
Double Right Rotation

- Left Rotate **3**



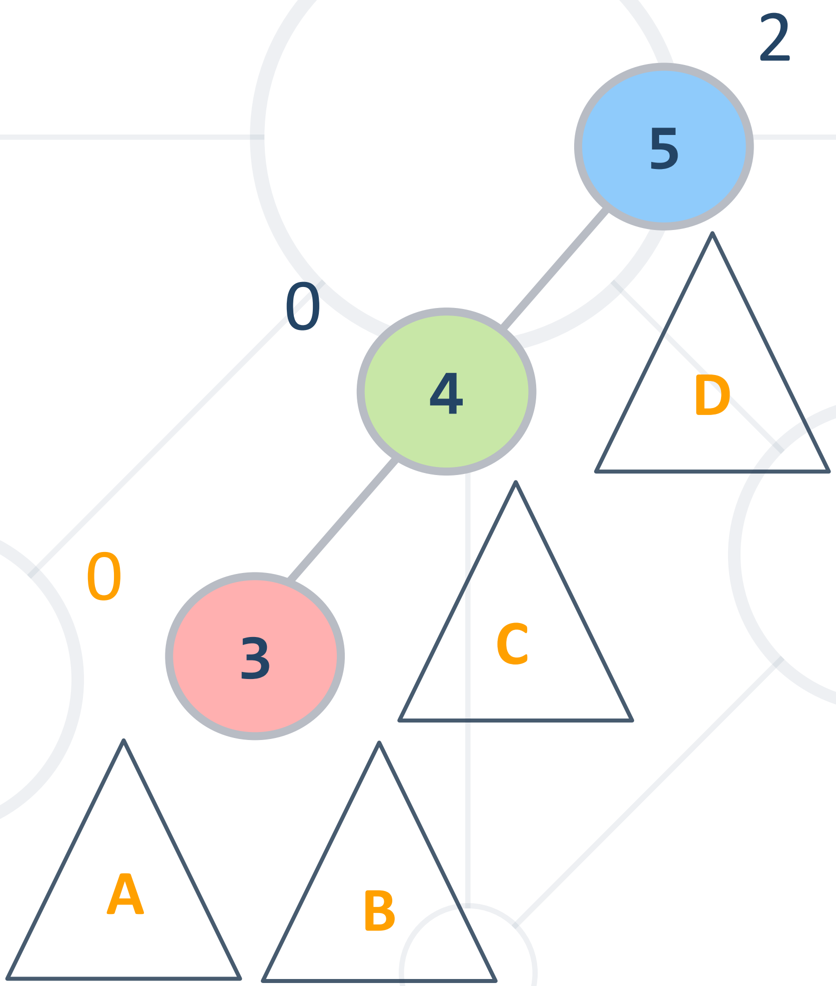
AVL Tree - Double Rotations

- Update Balance **3**



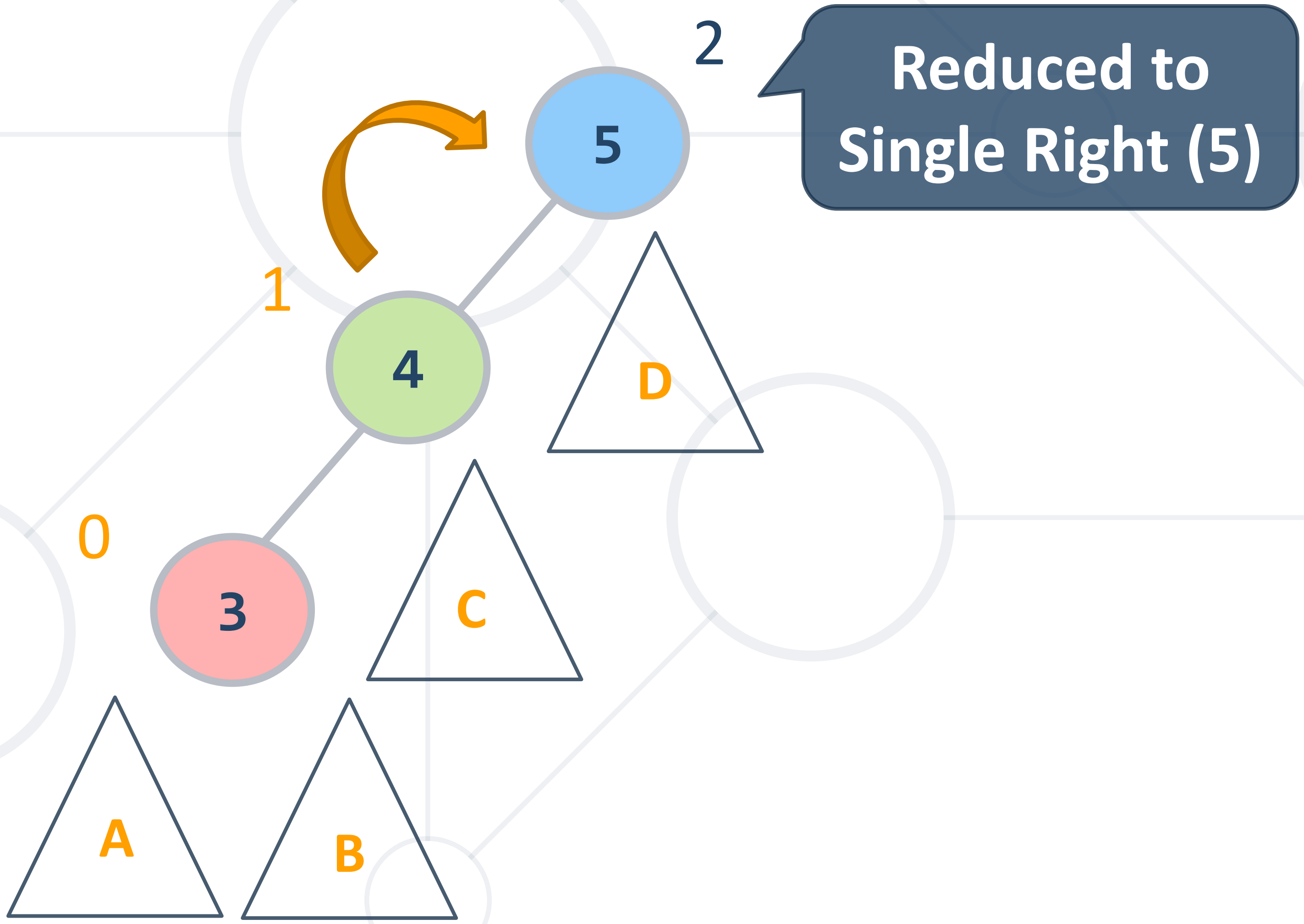
AVL Tree - Double Rotations

- Update Balance **3**



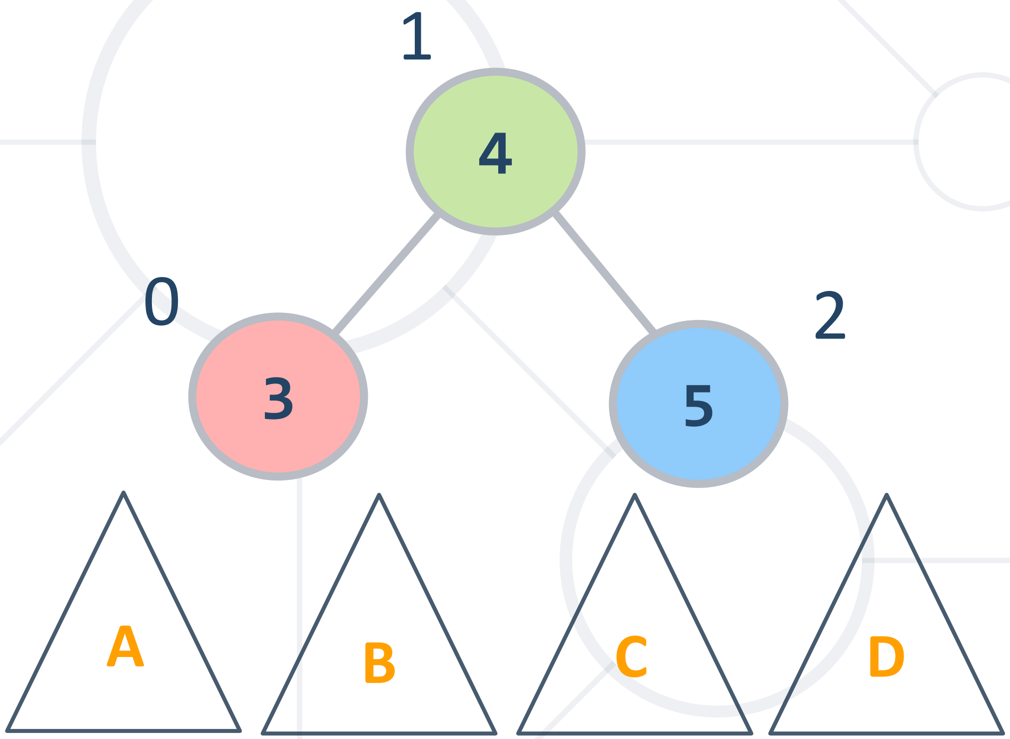
AVL Tree - Double Rotations

- Right Rotate **(5)**



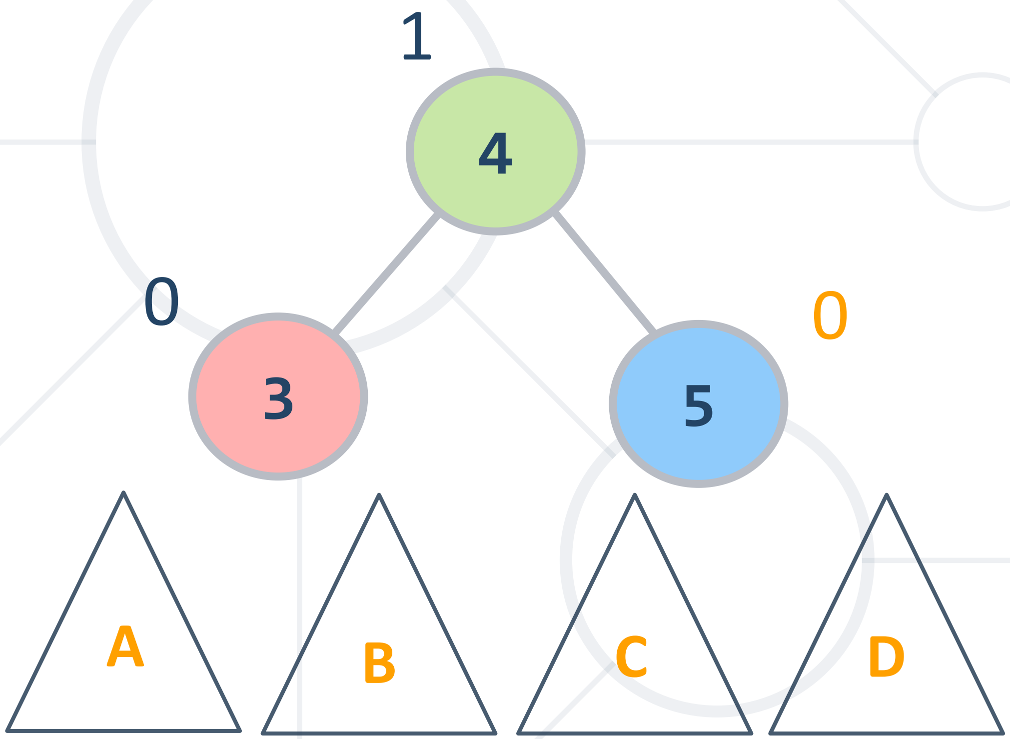
AVL Tree - Double Rotations

- Update Balance 5



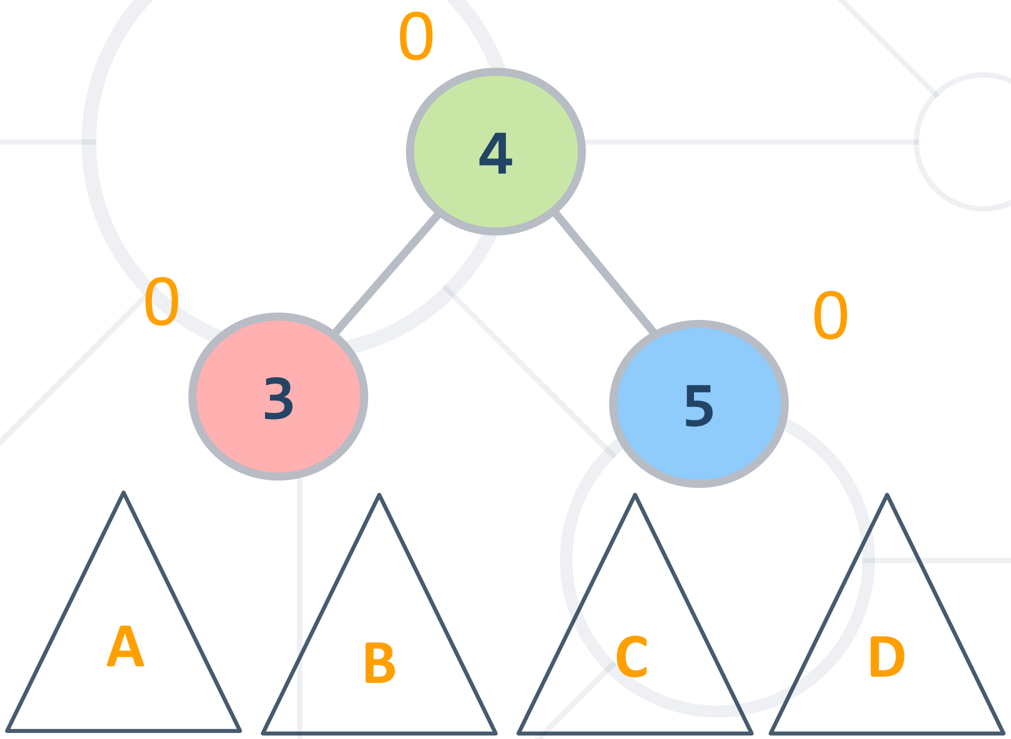
AVL Tree - Double Rotations

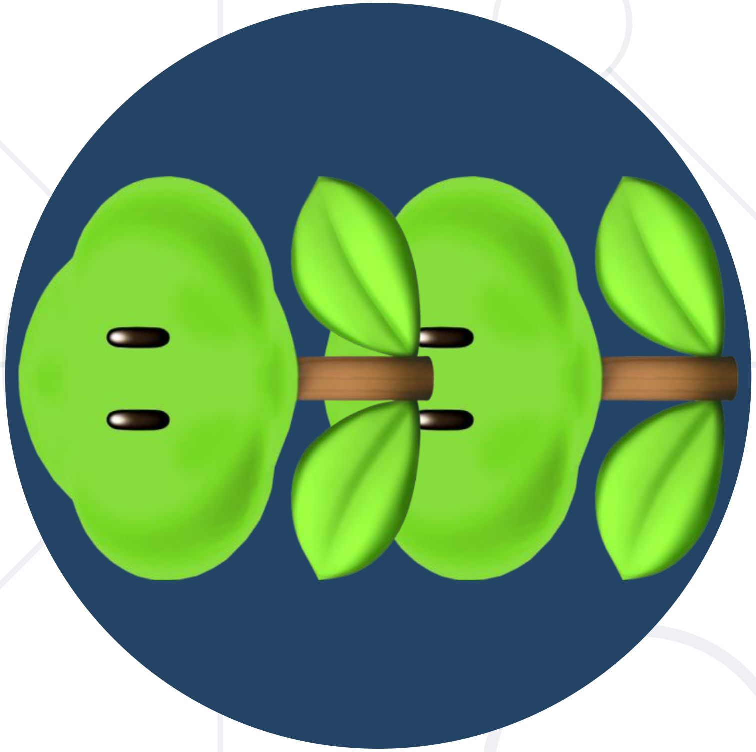
- Update Balance **4**



AVL Tree - Double Rotations

- Balance Restored!



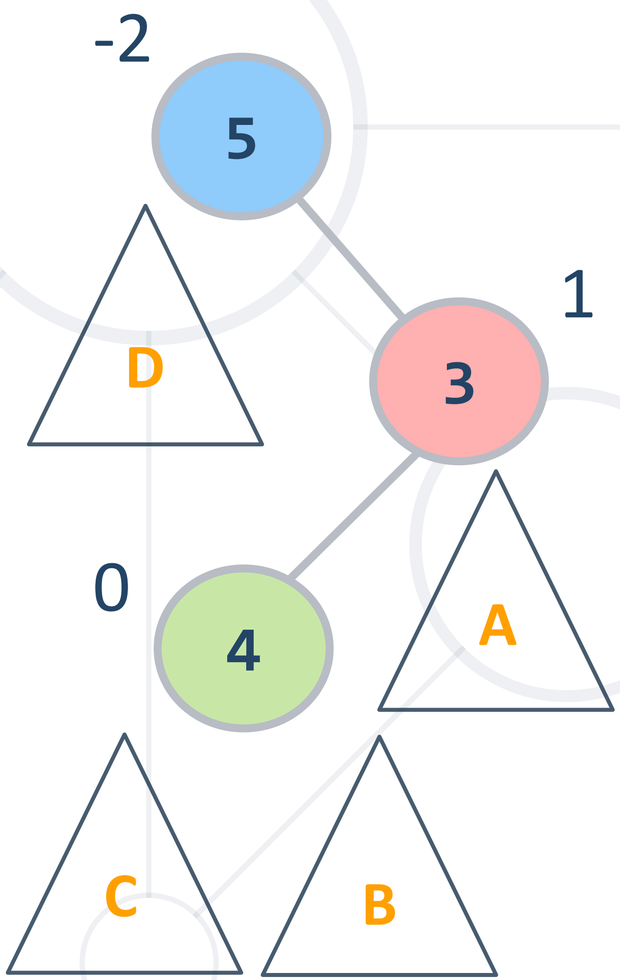


Double Left Rotation

Left-Right

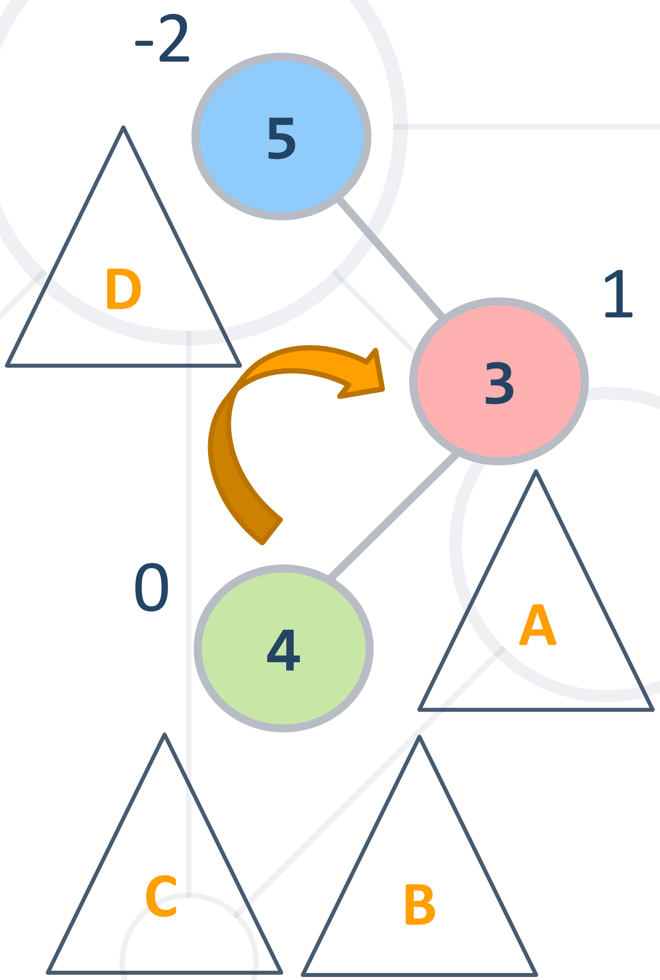
AVL Tree - Double Rotations

- Rotate **Left (node)** with positively balanced **Right Child**



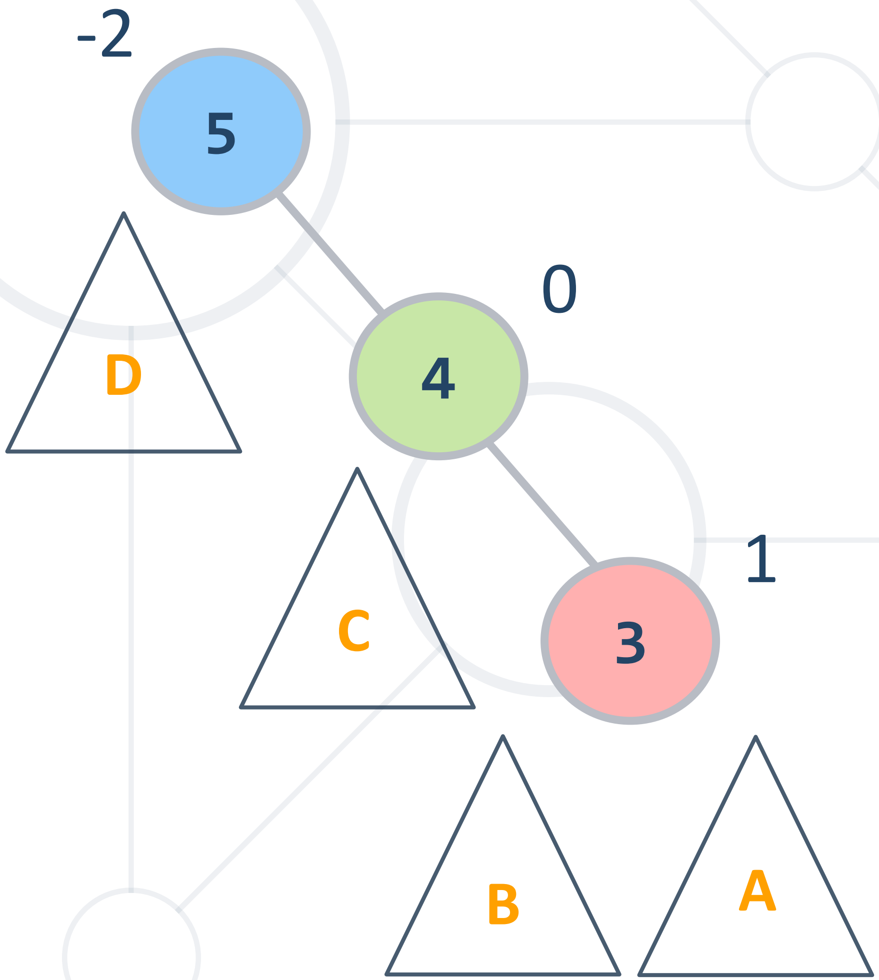
AVL Tree - Double Rotations

- Rotate Right **(3)**



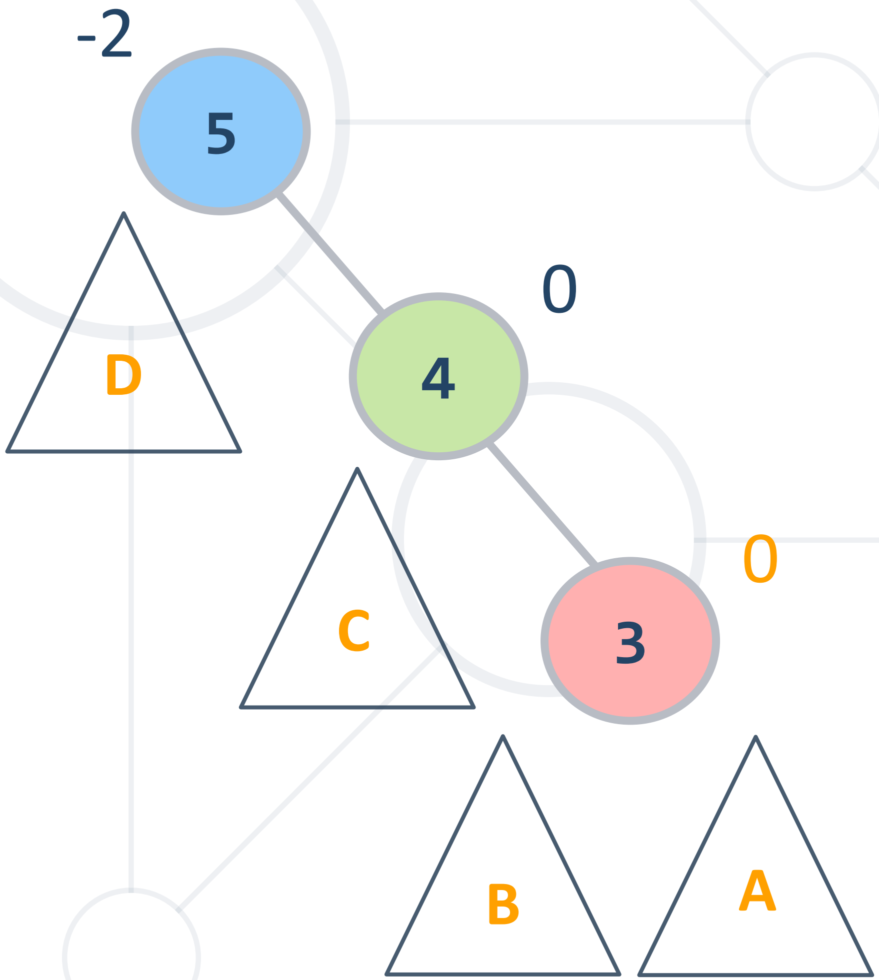
AVL Tree - Double Rotations

- Update Balance **(3)**



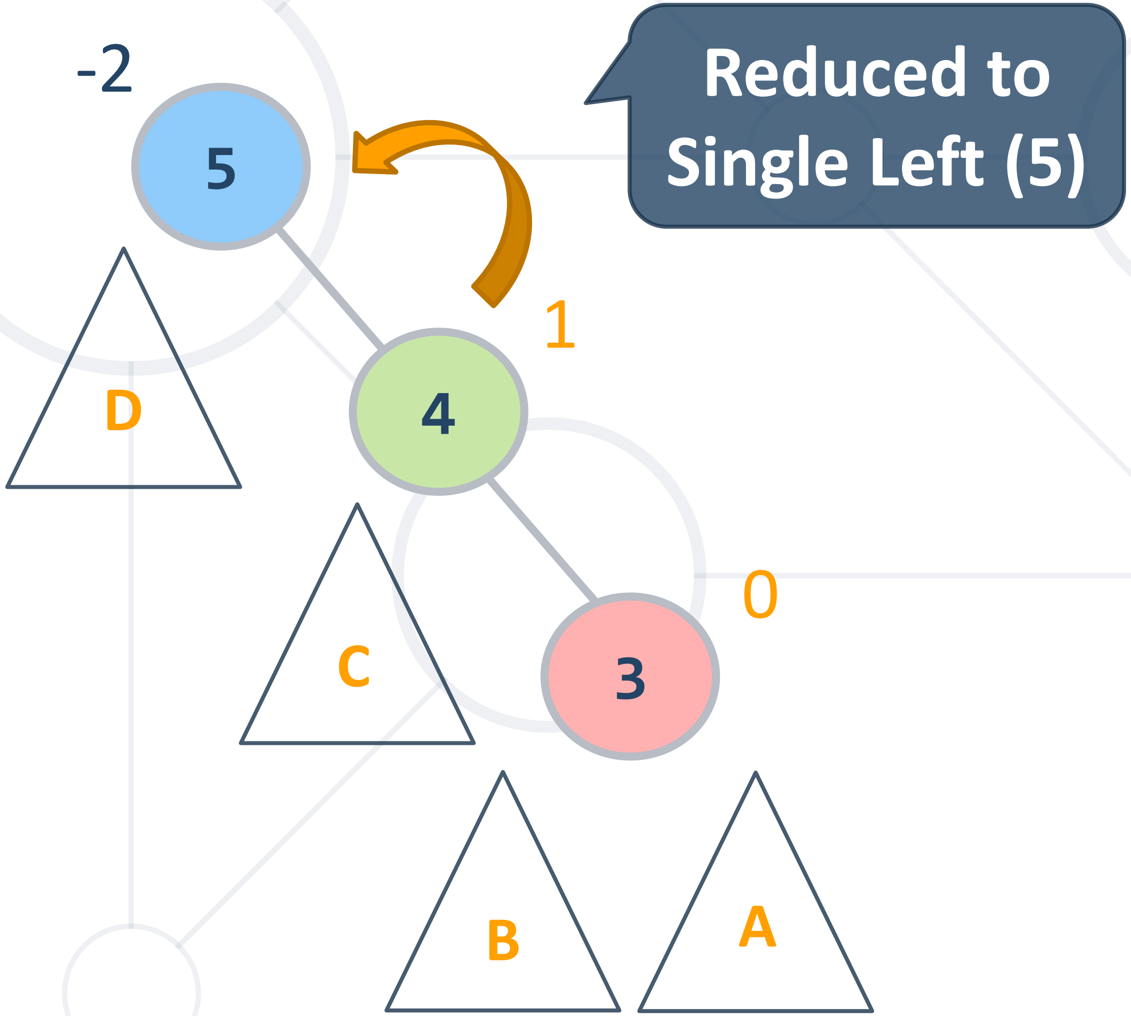
AVL Tree - Double Rotations

- Update Balance **(3)**



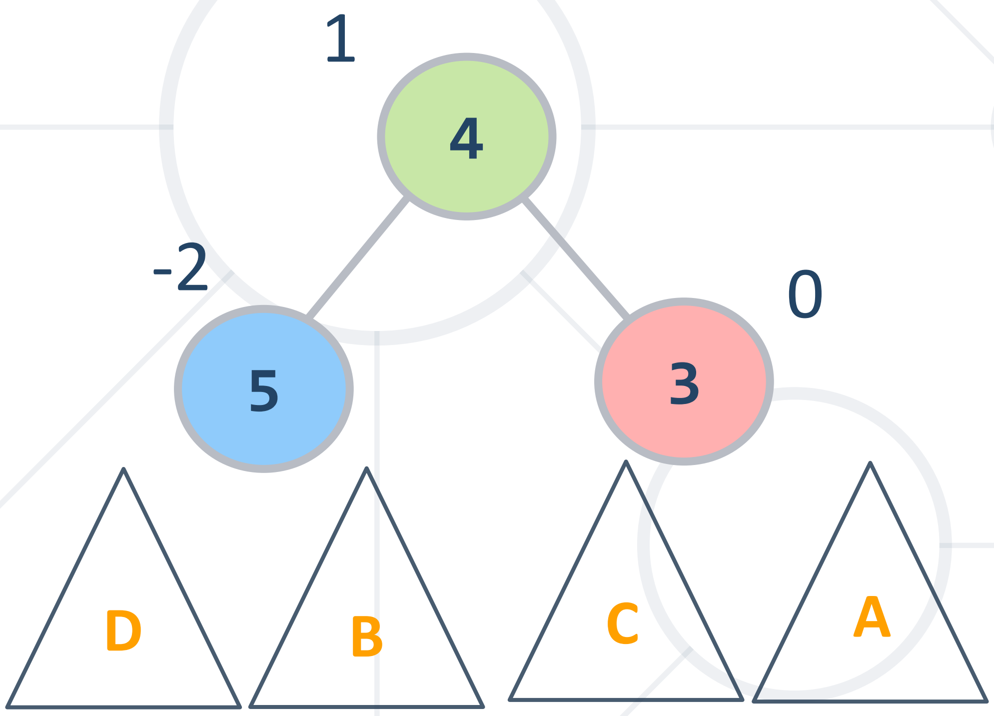
AVL Tree - Double Rotations

- Rotate Left **(5)**



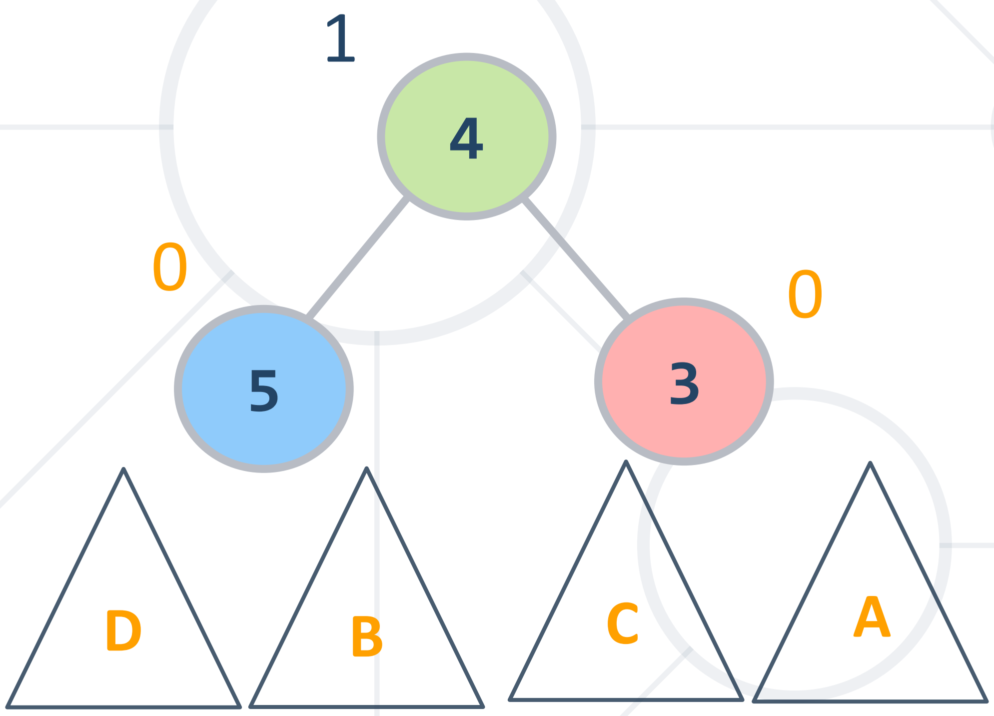
AVL Tree - Double Rotations

- Update Balance **(5)**



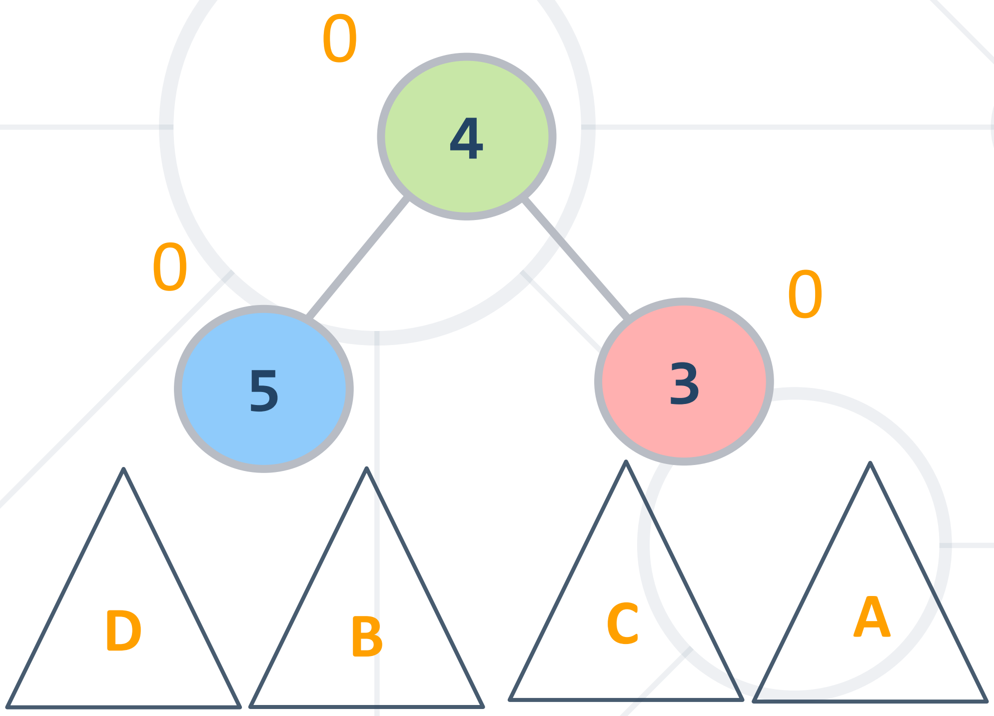
AVL Tree - Double Rotations

- Update Balance **(5)**



AVL Tree - Double Rotations

- Update Balance **(4)**

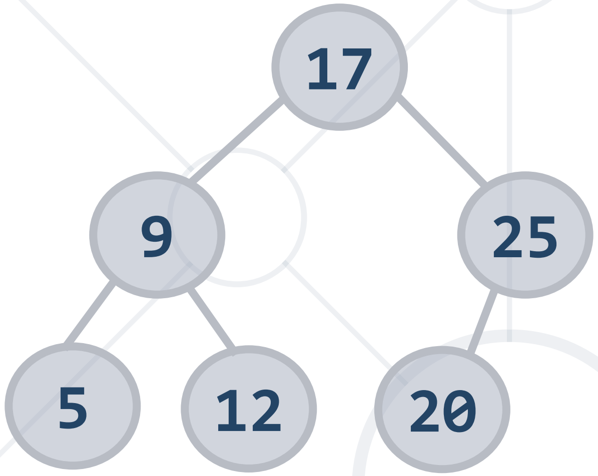


AVL Tree - Quiz

Software University

TIME'S

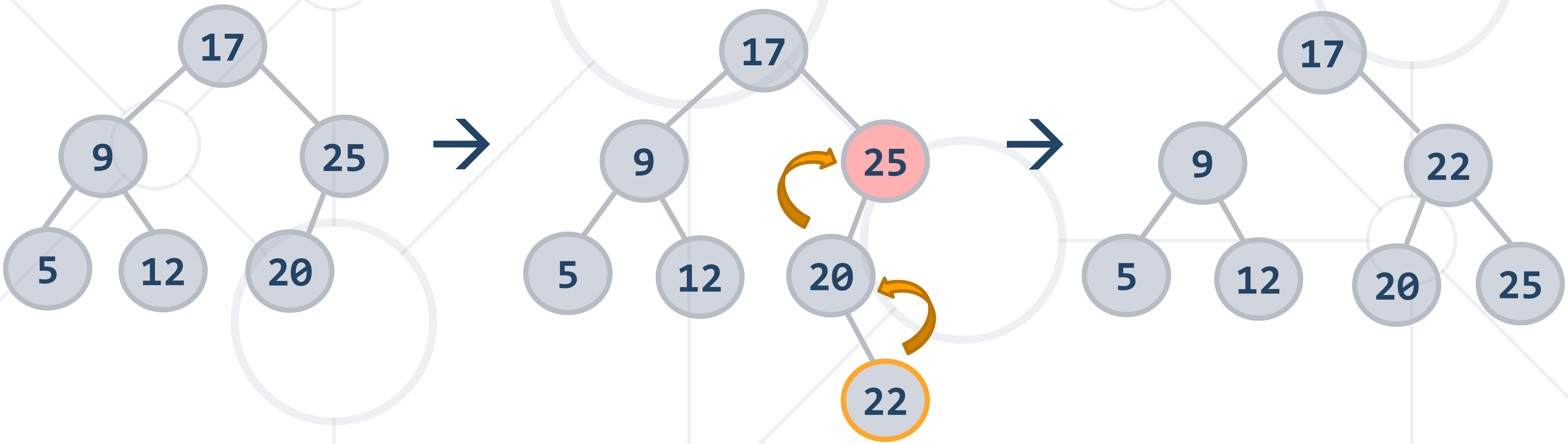
- Insert **22**. What will be the **resulting tree**?



AVL Tree - Quiz

TIME'S UP!

- Insert **22**. What will be the **resulting tree**?

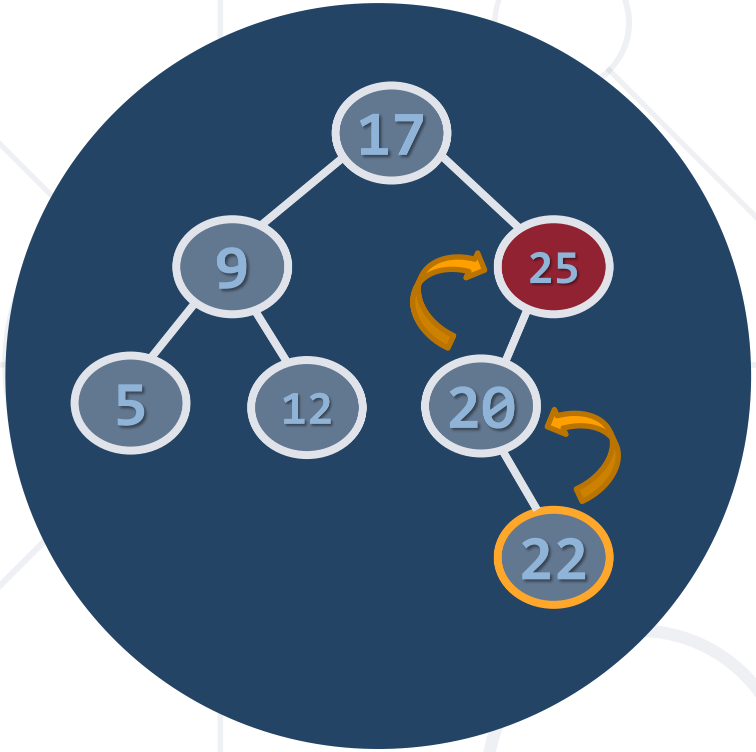


AVL Tree - Summary

Software University

Structure	Worst case			Average case	
	Search	Insert	Delete	Search Hit	Insert
BST	N	N	N	1.39 lg N	1.39 lg N
2-3 Tree	c lg N	c lg N	c lg N	c lg N	c lg N
Red-Black	2 lg N	2 lg N	2 lg N	lg N	lg N
AVL Tree	1.44 lg N	1.44 lg N	1.44 lg N	lg N	lg N

Insert/Delete perform O(lgN) rotations



AVL Tree

Balancing Implementation

Rotate Right

```java
public Node<T> rotateRight(Node<T> node) {
  Node<T> left = node.left;
  node.left = node.left.right;
  left.right = node;

  updateHeight(node);

  return left;
}
```

Balance Node

```
private Node<T> balance(Node<T> node) {
  int balance = height(node.Left) - height(node.Right);
  if (balance < -1) // Right child is heavy {
    balance = height(node.right.left) - height(node.right.right);
    if (balance <= 0) { return rotateLeft(node); }
    else { node.right = rotateRight(node.right); return rotateLeft(node); }
  }
  else if (balance > 1) // Left child is heavy {
    balance = height(node.Left.Left) - height(node.Left.Right);
    if (balance >= 0) { return rotateRight(node); }
    else { node.left = rotateLeft(node.left); return rotateRight(node); }
  }

  return node;
}
```

Summary

- B-Trees can be **efficiently** sored on disks
- 2-3 tree is **B-Tree** of order **3**
- Not **perfectly** balanced
- Performs **local** transformations
- AVL Trees
  - Rotations right and left
  - Double rotations

Software University

Questions?

SoftUni

- Software University
- SoftUni Creative
- SoftUni Digital
- SoftUni Foundation
- SoftUni Kids
- Finance Academy

SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

Flutter International

INDEAVR Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

BOSCH

Postbank Решения за твоето утре

PHAR VISION

SmartIT

DXC TECHNOLOGY

createX

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License