B-Trees - 2-3 Trees and Red-Black Trees

Balanced BSTs, Insertions and Rotations

SoftUni Team

Technical Trainers

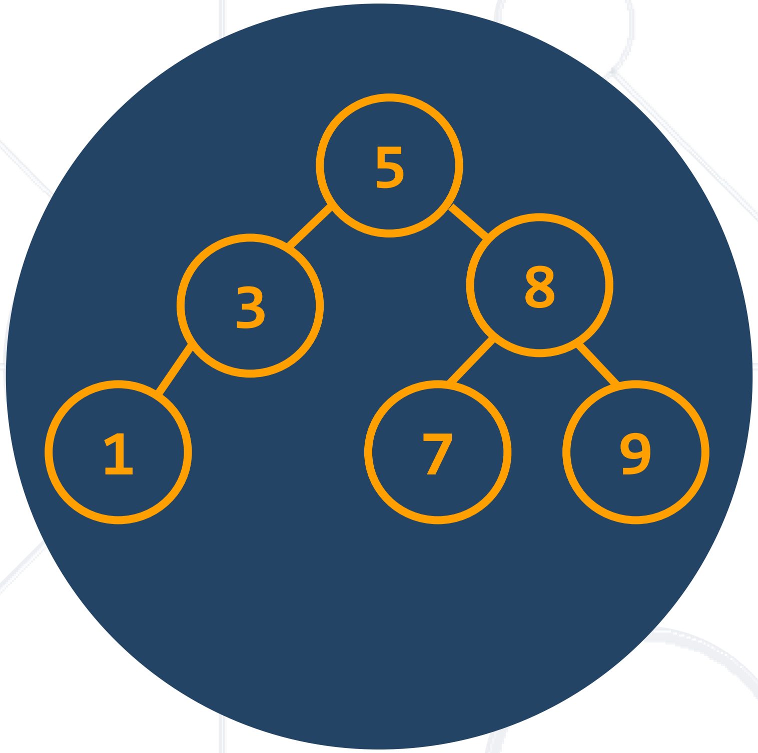
Software University

Software University

https://softuni.bg/

SoftUni

Table of Contents

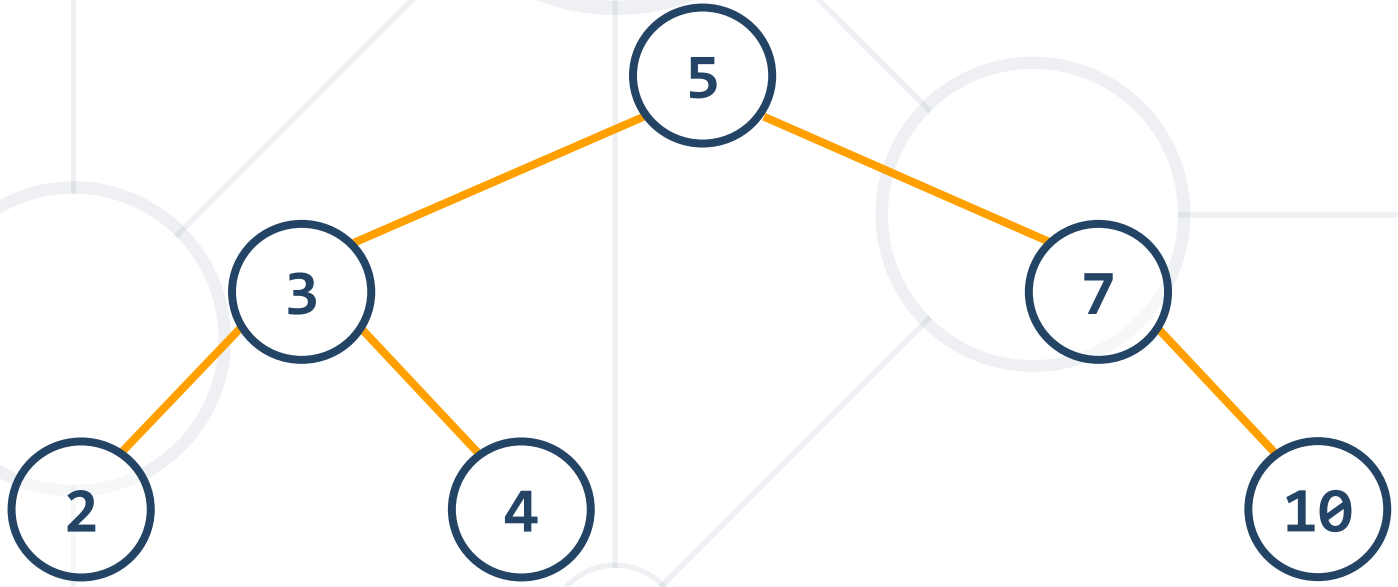


Balanced BSTs

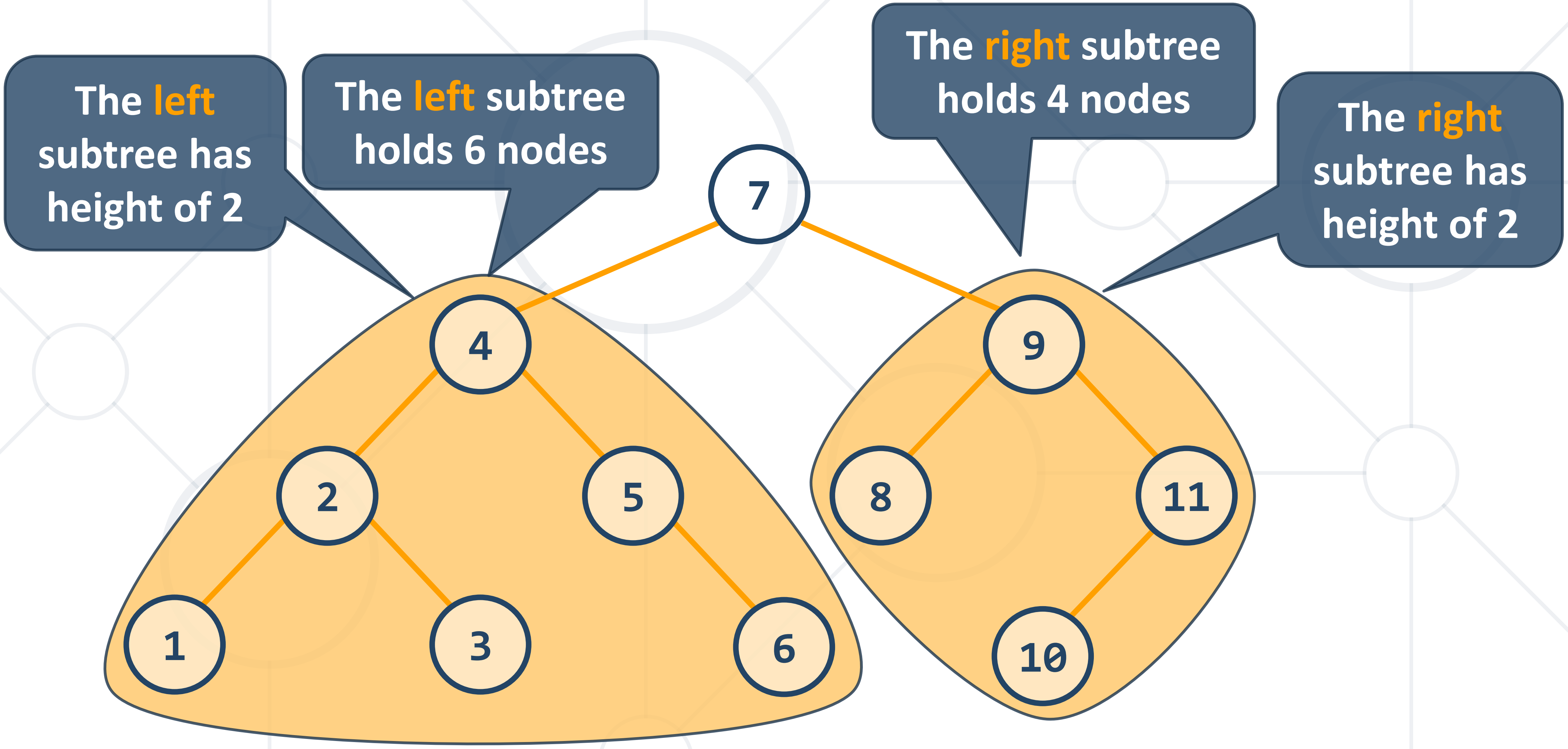
Balancing a BST

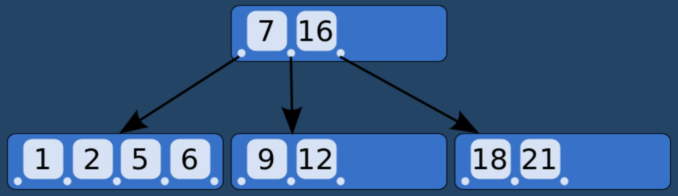
What is a Balanced Binary Search Tree?

- Binary search trees can be **balanced**
  - The left and right subtrees' heights differ by at most one
  - Left and right subtrees are balanced



Balanced Binary Search Tree – Example


The left subtree has height of 2
The left subtree holds 6 nodes
The right subtree holds 4 nodes
The right subtree has height of 2
7
4
2
5
1
3
6
9
8
11
10



B-Trees

B-Trees Concept

What is a B-Tree?

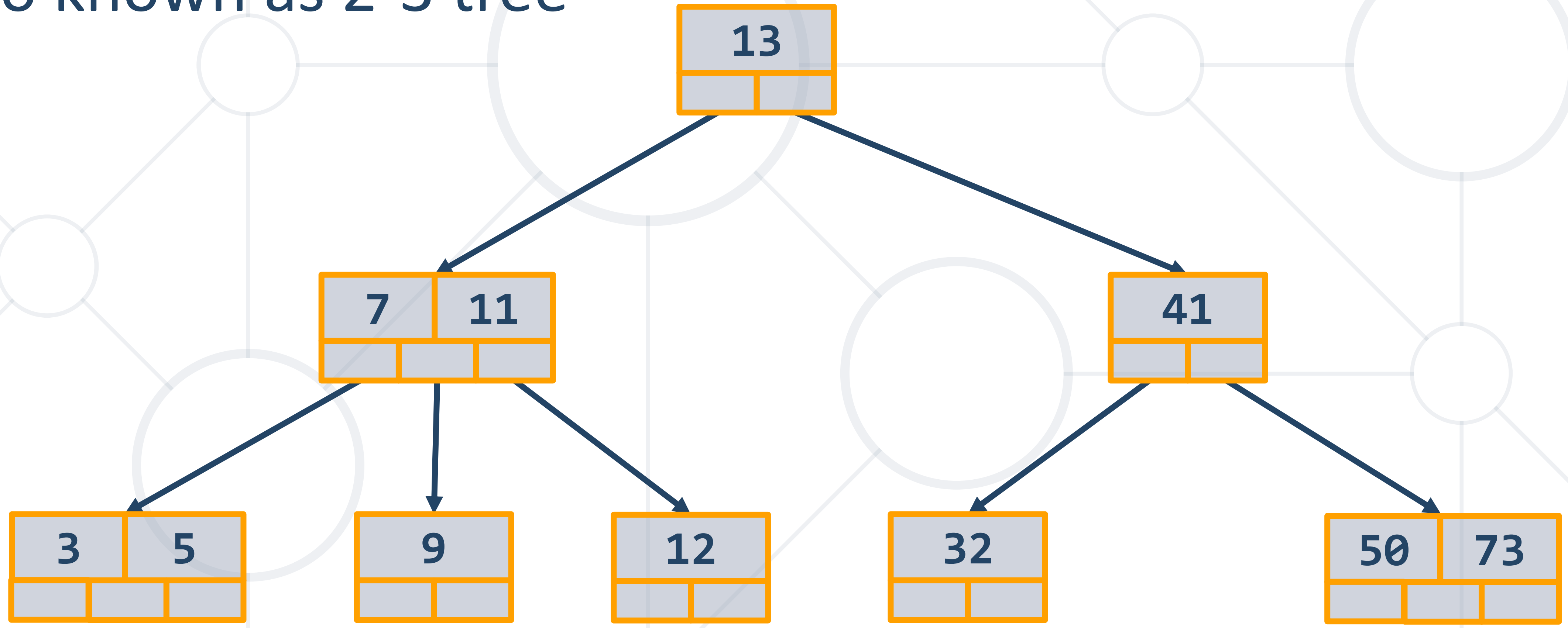
- B-trees are a generalization of the concept of ordered binary search trees – see the visualization
  - B-tree of order **b** has between **(b-1)/2** and **b-1** keys in a node and between **b/2+1** and **b** child nodes
  - The keys in each node are ordered increasingly
  - All keys in a child node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

B-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

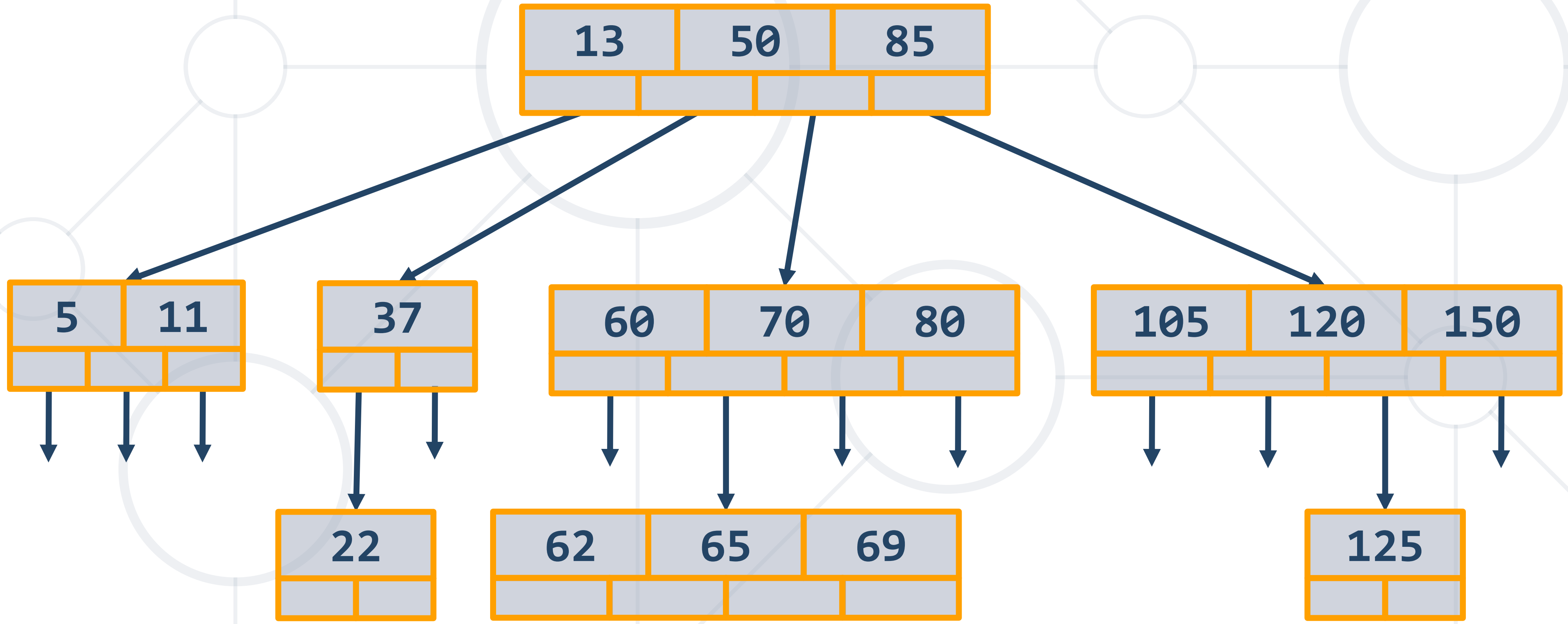
B-Tree – Example

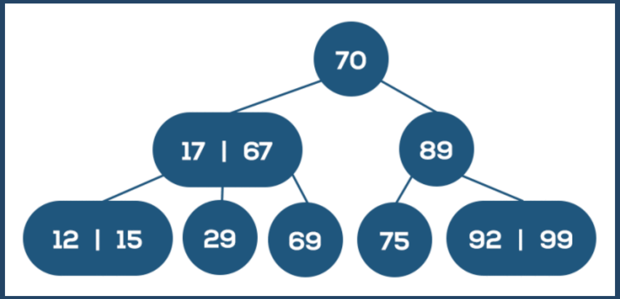
- B-Tree of order **3** (max count of child nodes), also known as 2-3 tree



B-Tree – Example

- B-Tree of order **4** (max count of child nodes) - 2-3-4 tree





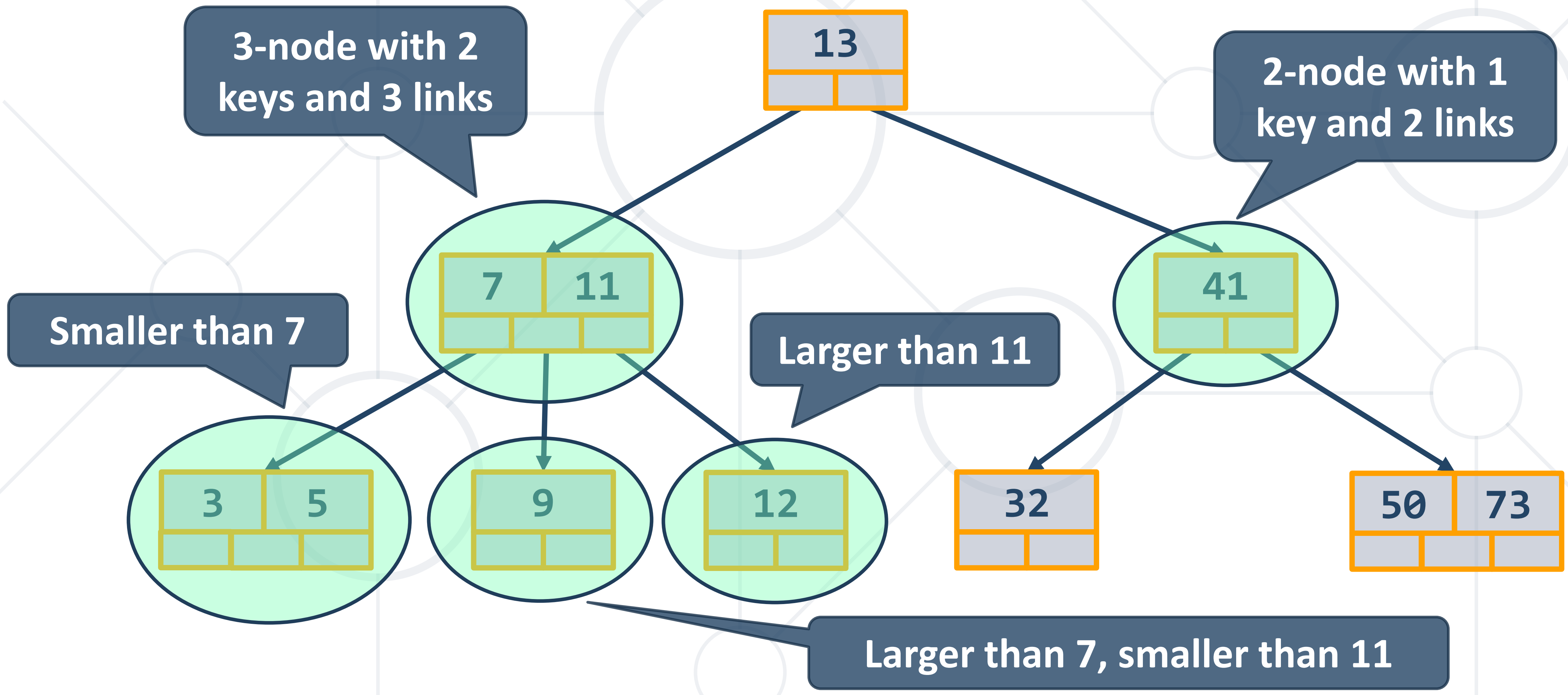
2-3 Trees

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.

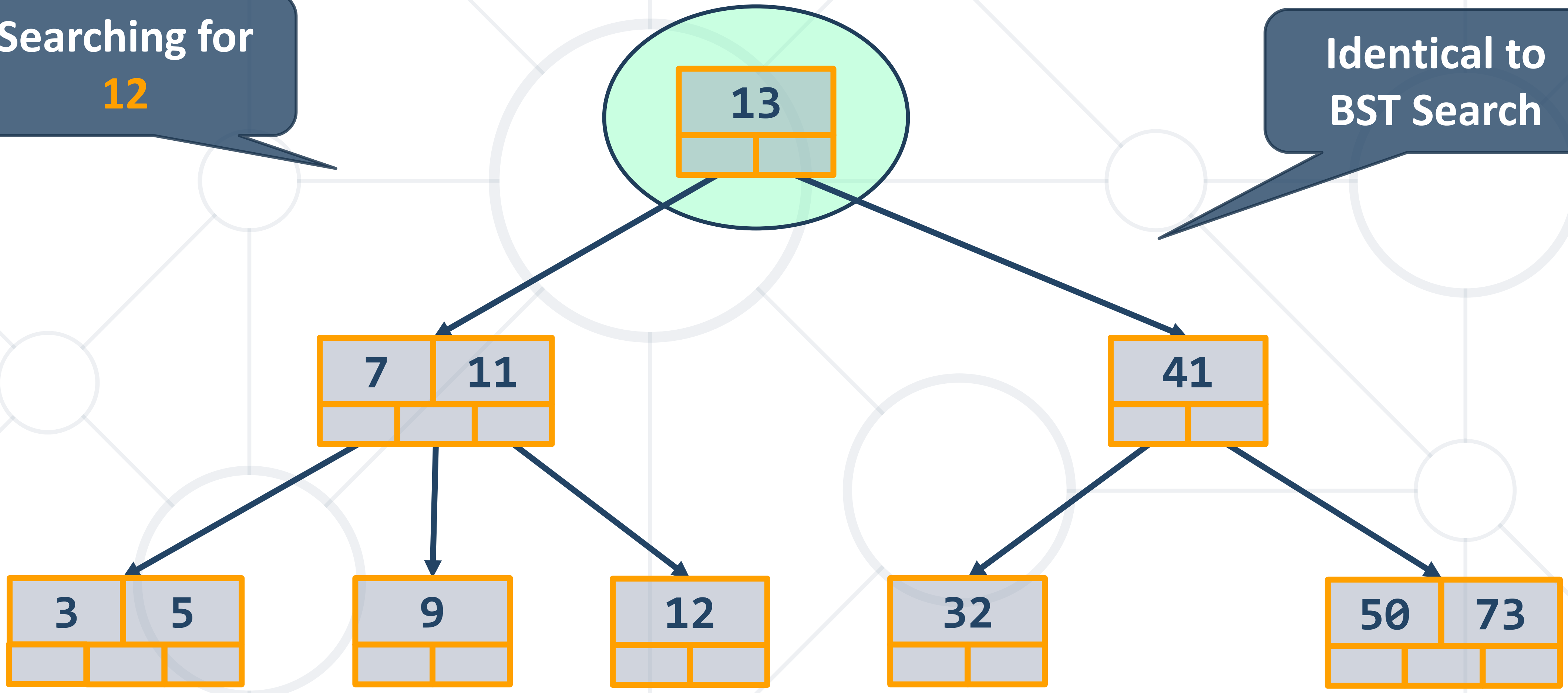
2-3 Tree Example


3-node with 2 keys and 3 links
13
2-node with 1 key and 2 links
7
11
41
Smaller than 7
Larger than 11
3
5
9
12
32
50
73
Larger than 7, smaller than 11

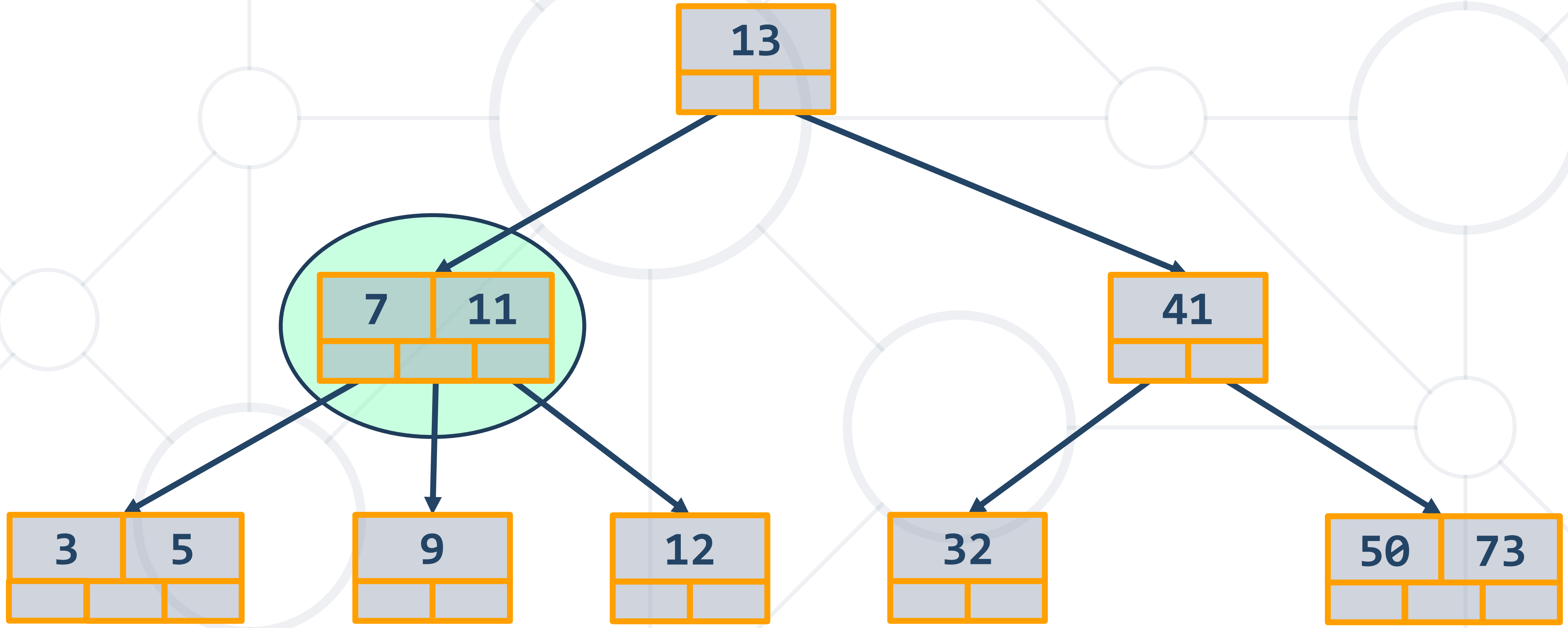
2-3 Tree Searching

Searching for 12

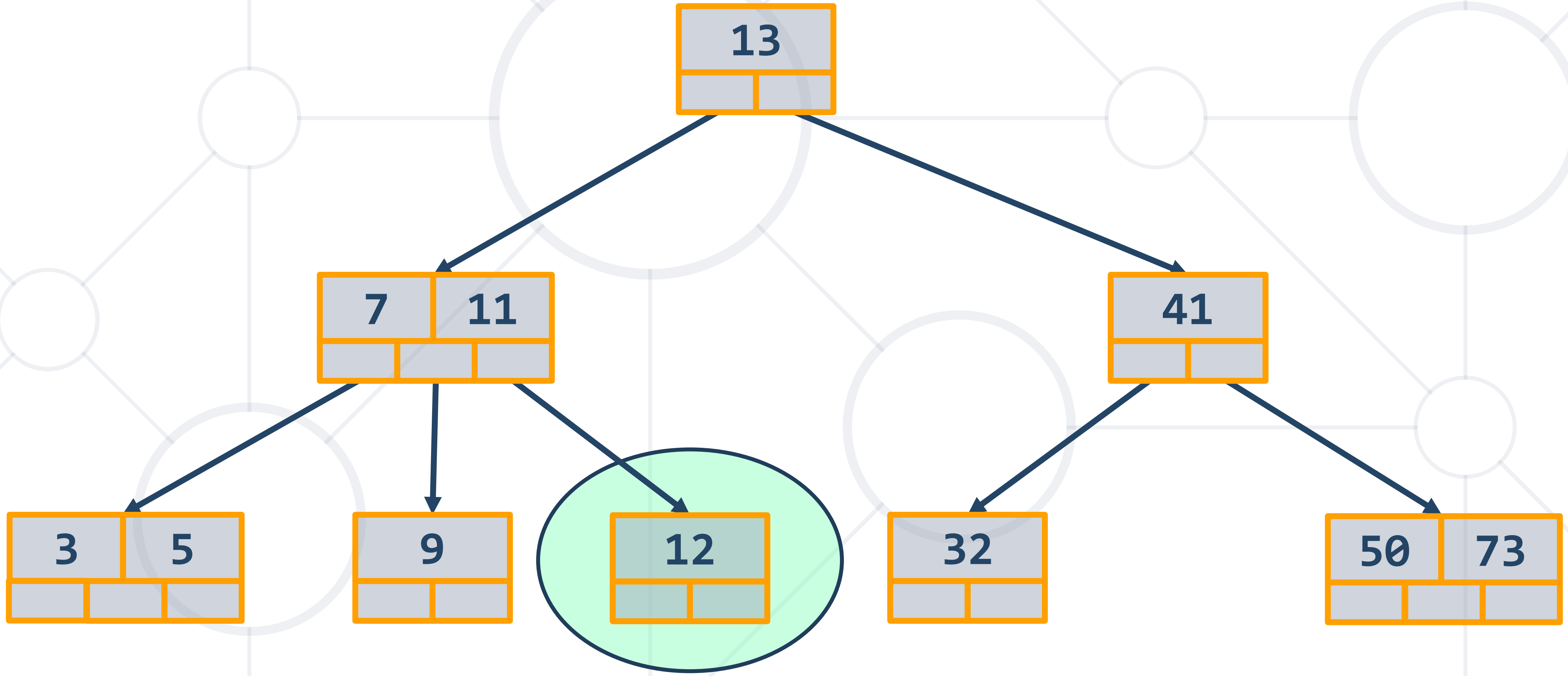
Identical to BST Search



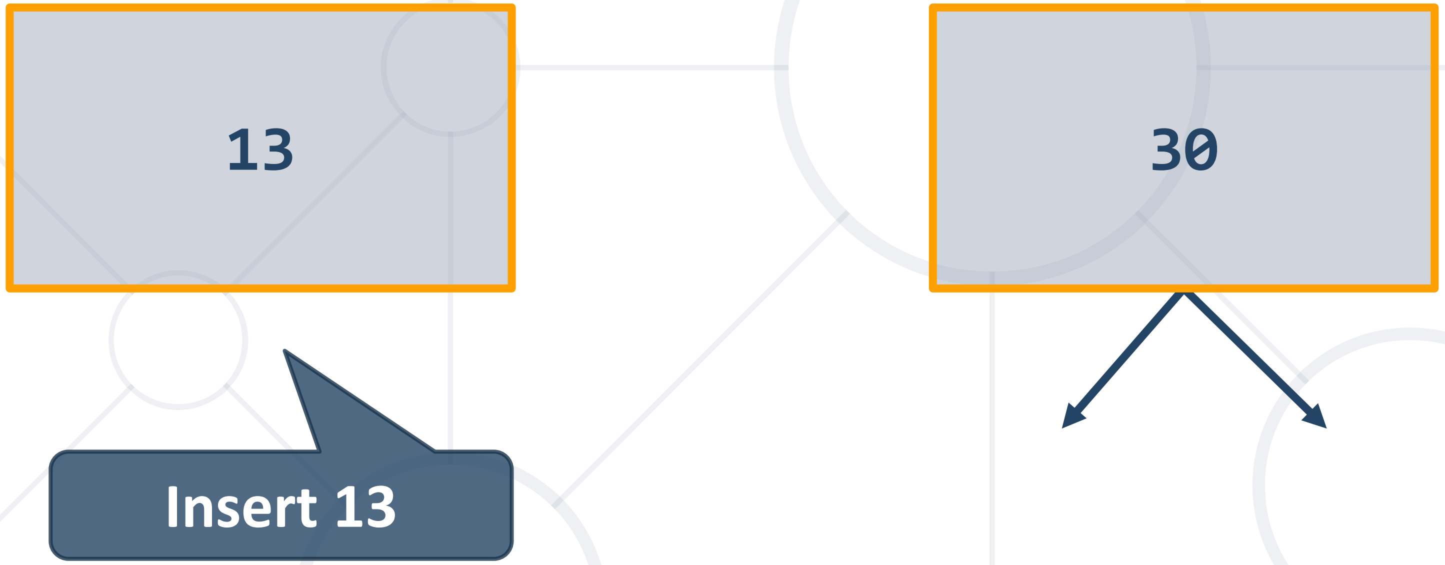
2-3 Tree Searching



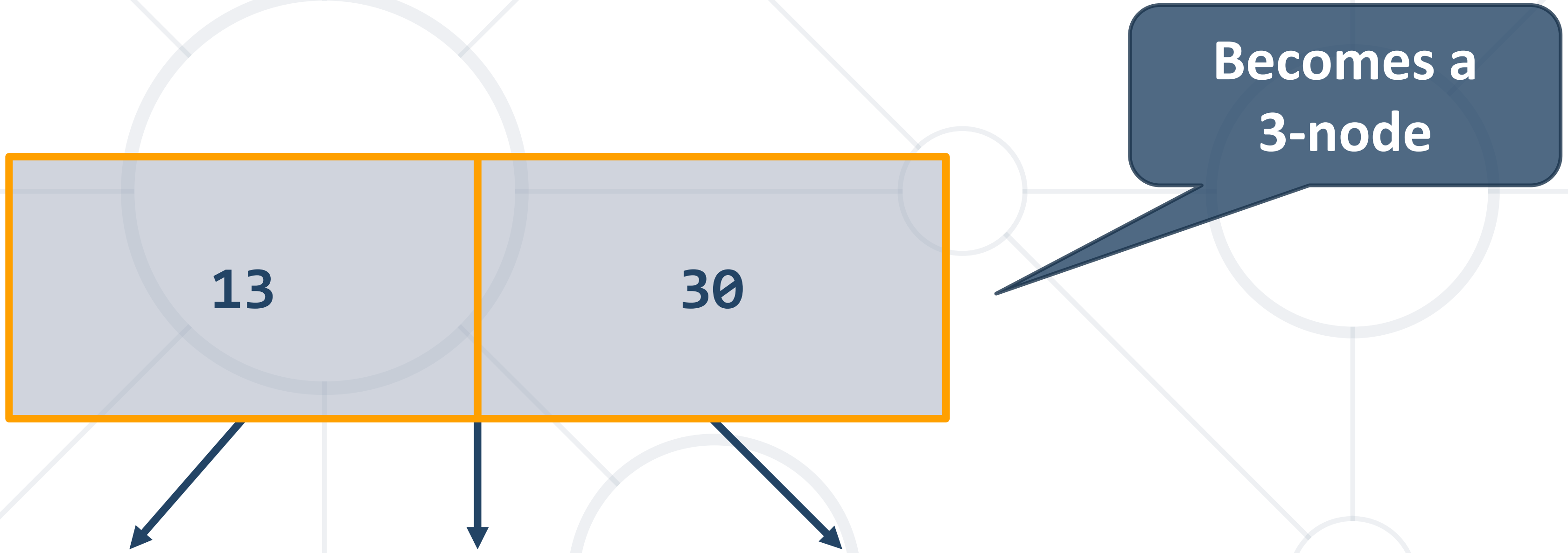
2-3 Tree Searching



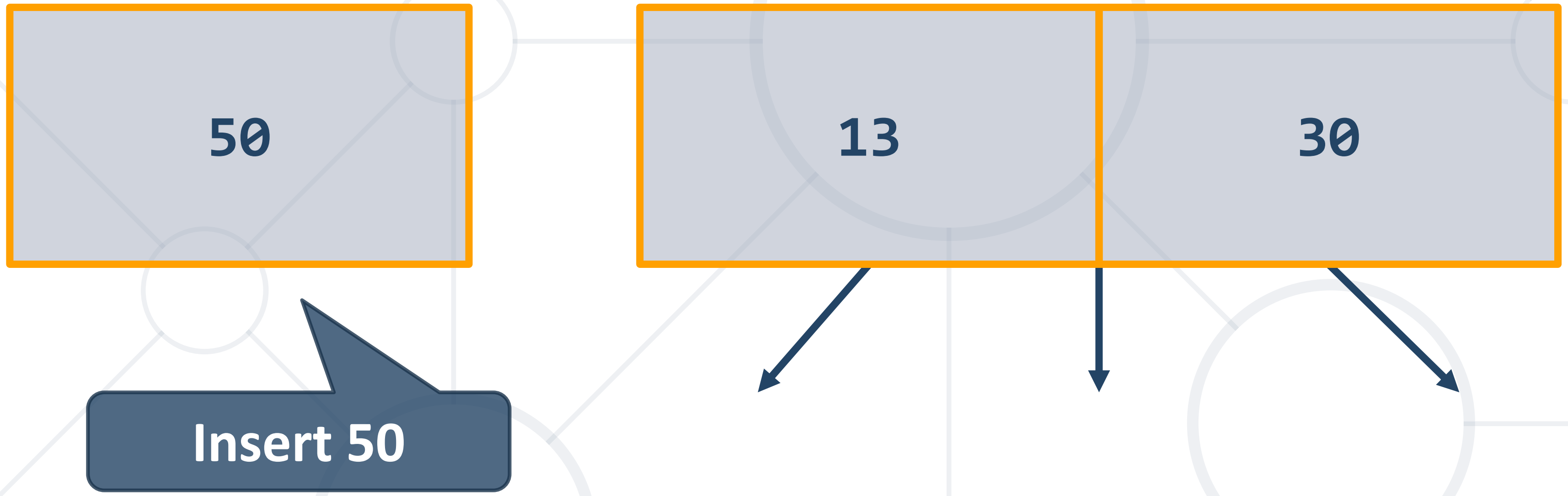
2-3 Tree Insertion (at 2-node)



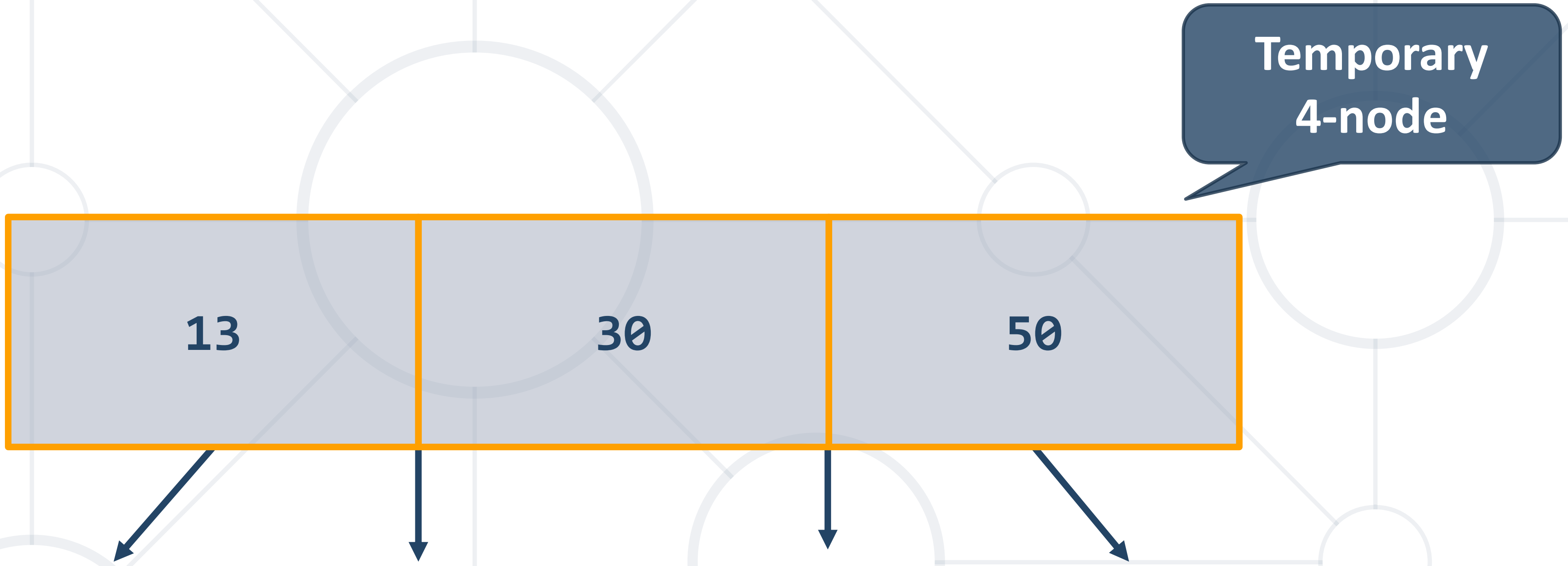
2-3 Tree Insertion (at 2-node)



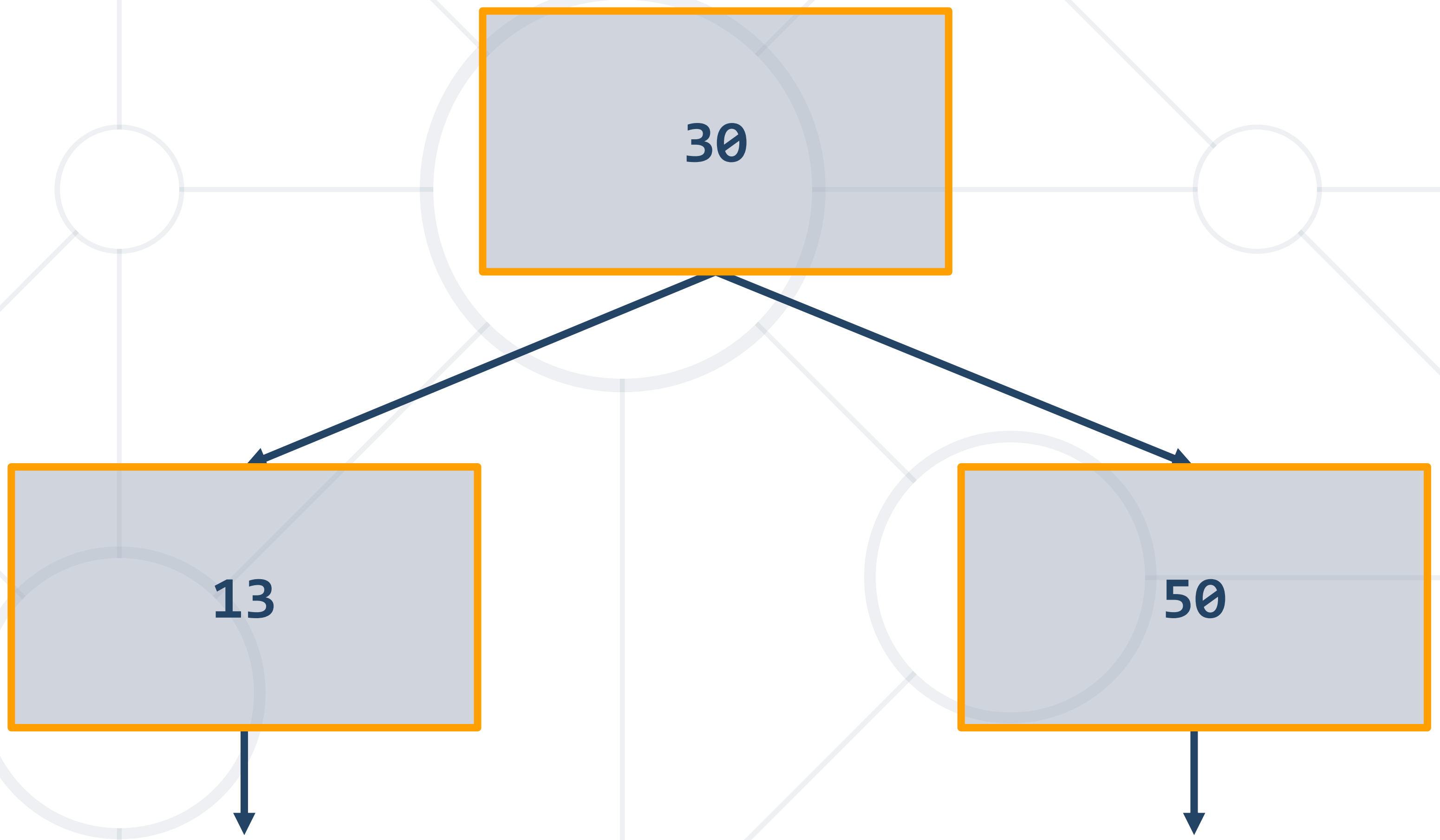
2-3 Tree Insertion (at 3-node)



2-3 Tree Insertion (at 3-node)

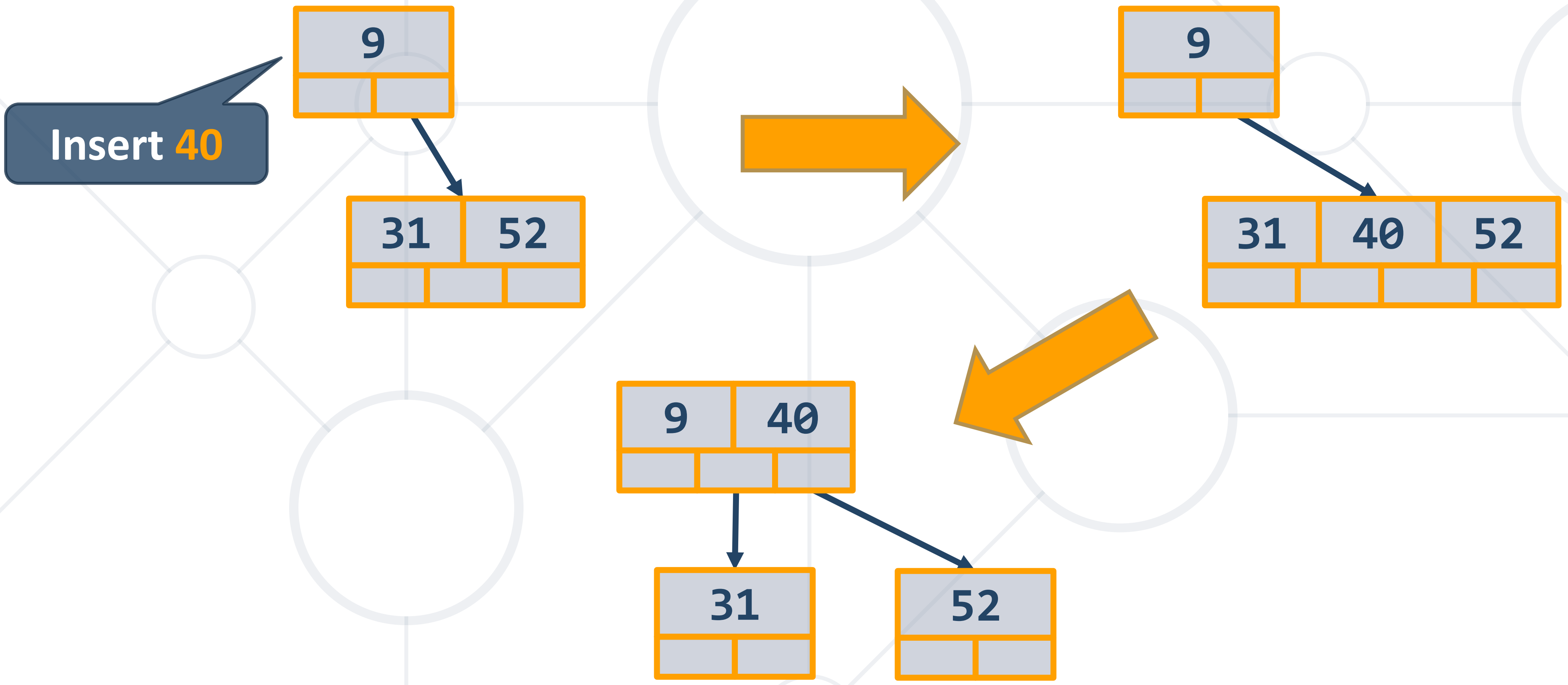


2-3 Tree Insertion (at 3-node)



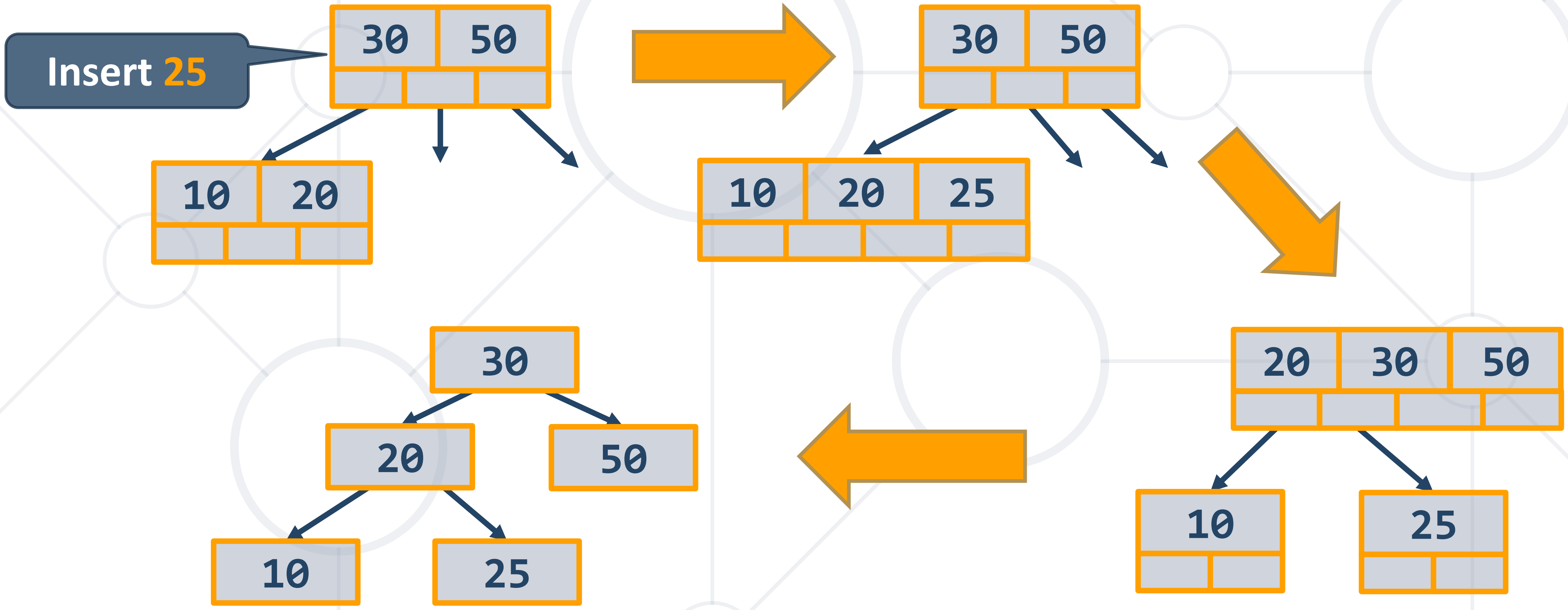
2-3 Tree Insertion

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



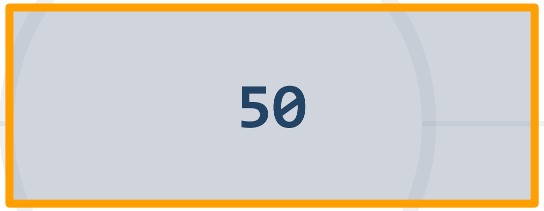
2-3 Tree Insertion (2)

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



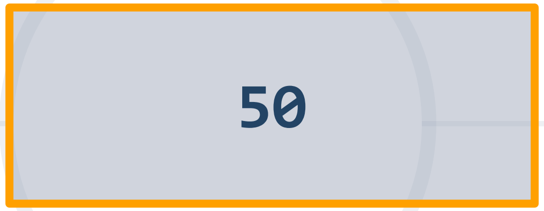
2-3 Tree Construction

- Insert **50**



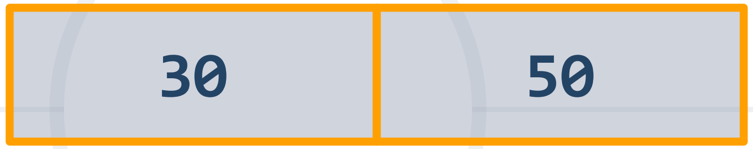
2-3 Tree Construction (2)

- Insert **30**



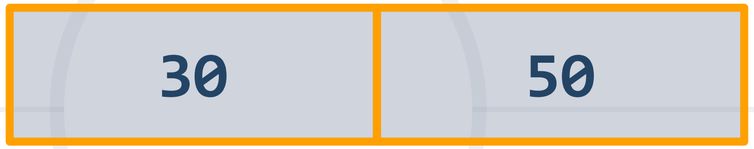
2-3 Tree Construction (2)

- Insert **30**



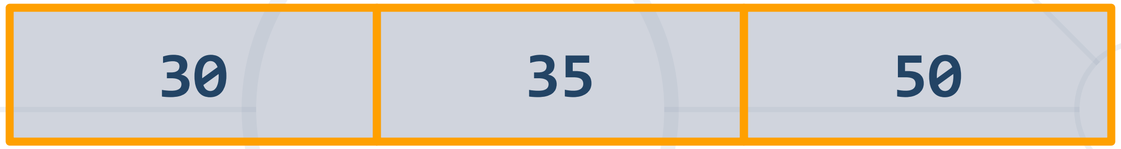
2-3 Tree Construction (3)

- Insert **35**



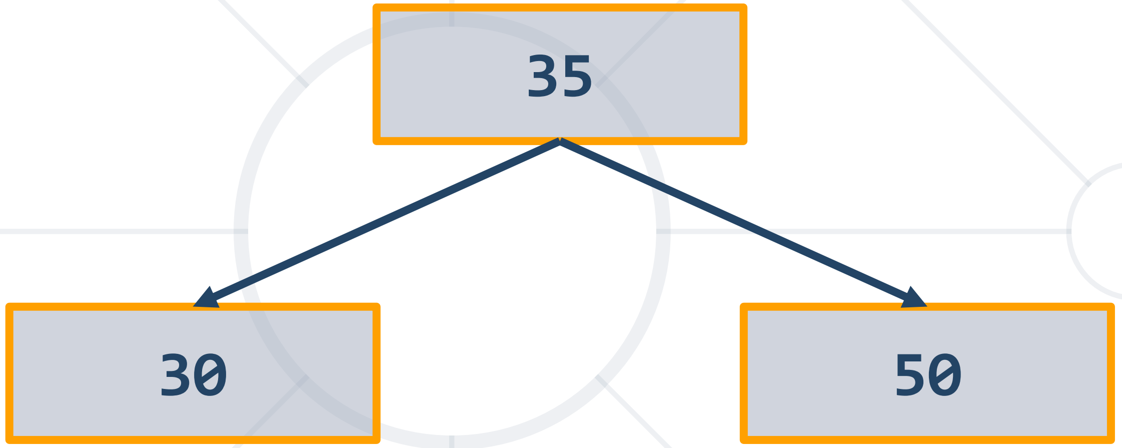
2-3 Tree Construction (3)

- Insert **35**



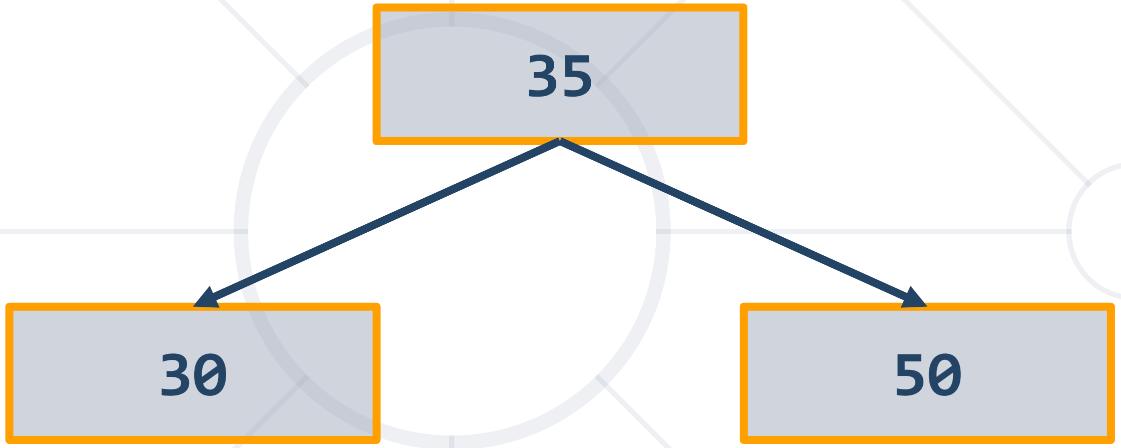
2-3 Tree Construction (3)

- Insert **35**



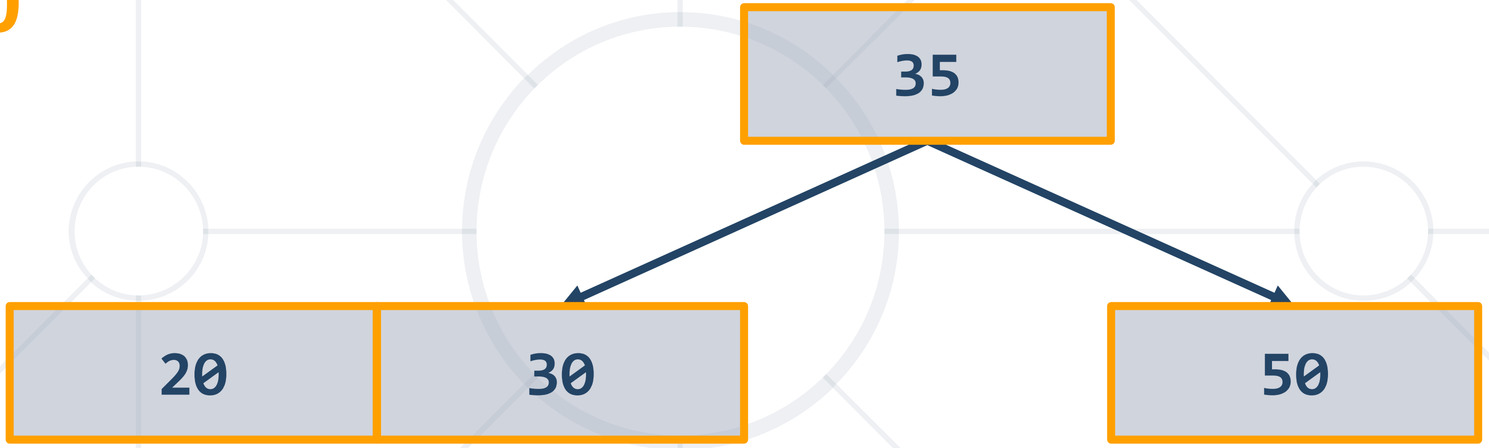
2-3 Tree Construction (4)

- Insert **20**



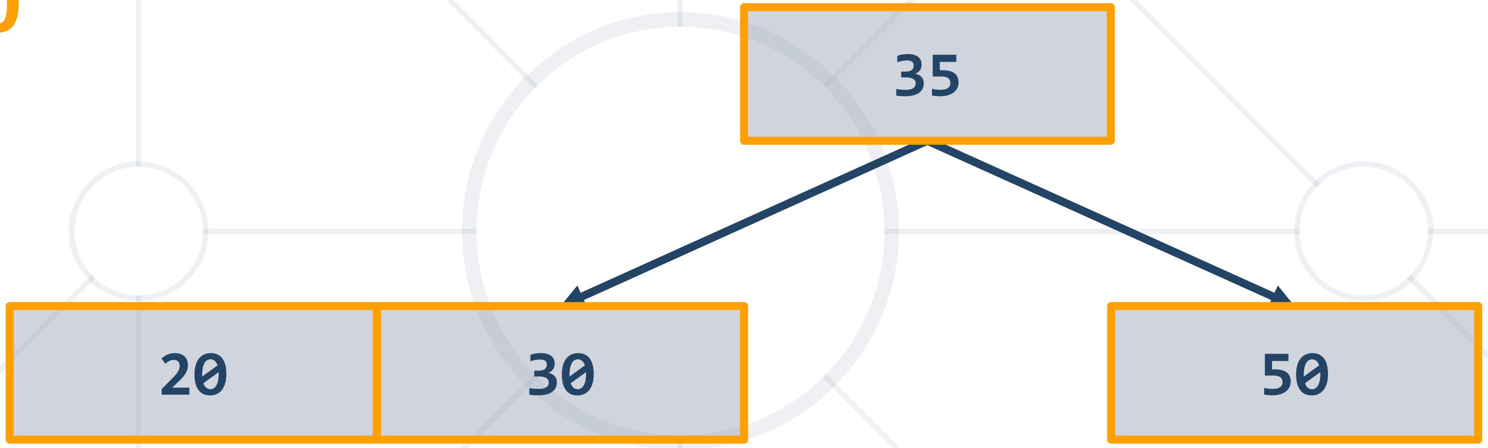
2-3 Tree Construction (4)

- Insert **20**



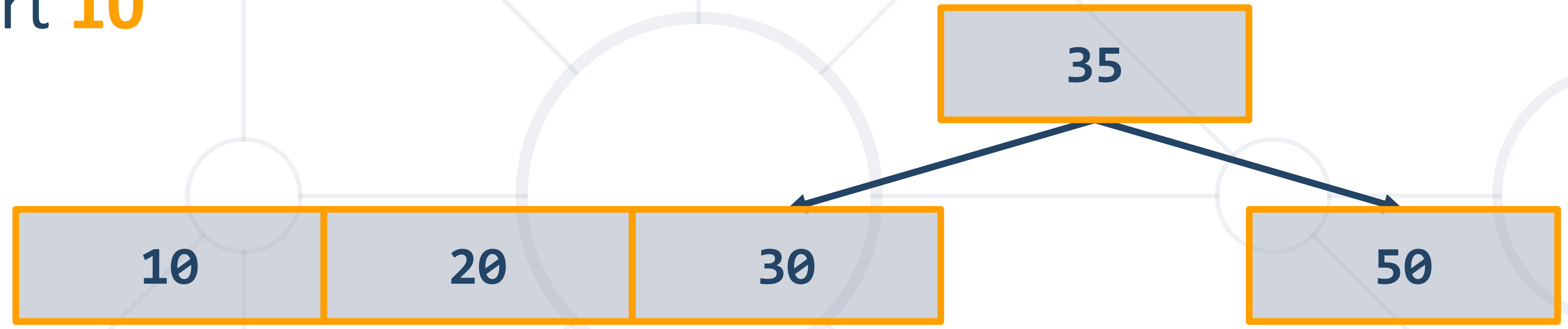
2-3 Tree Construction (5)

- Insert **10**



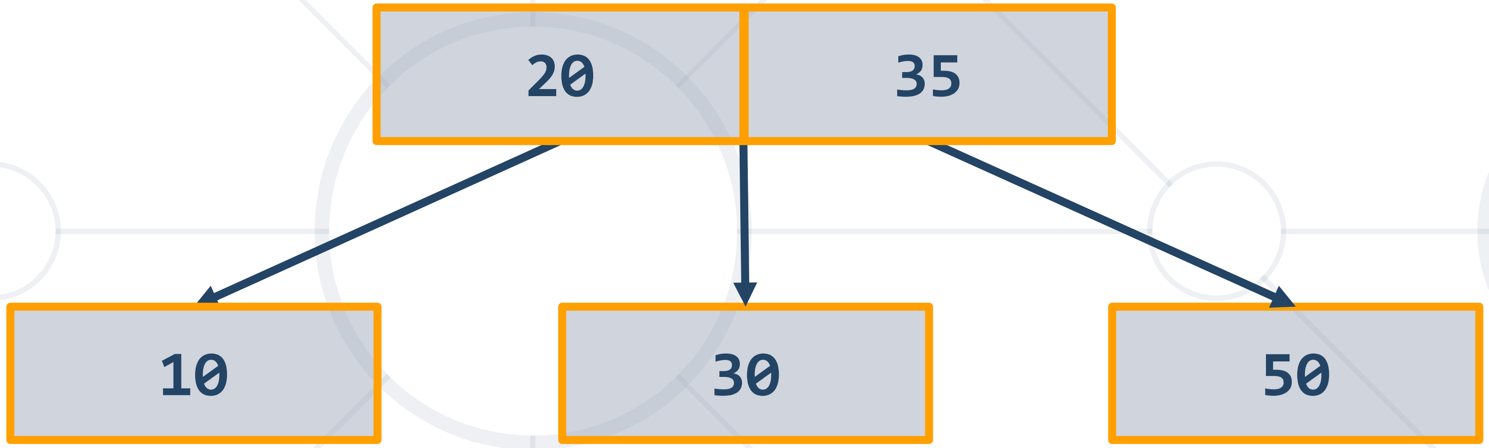
2-3 Tree Construction (5)

- Insert **10**



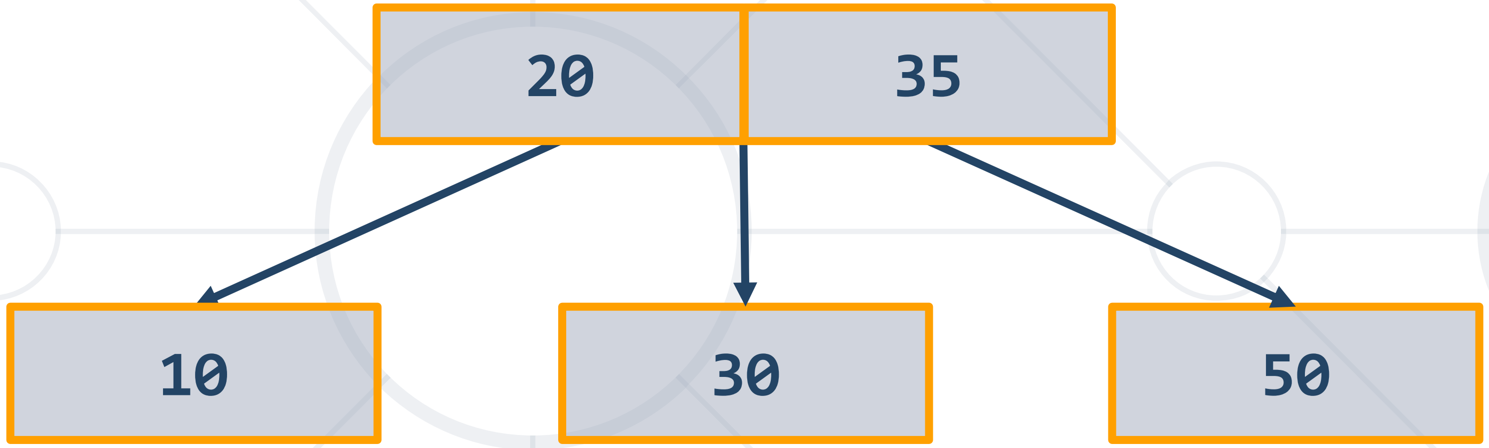
2-3 Tree Construction (5)

- Insert **10**



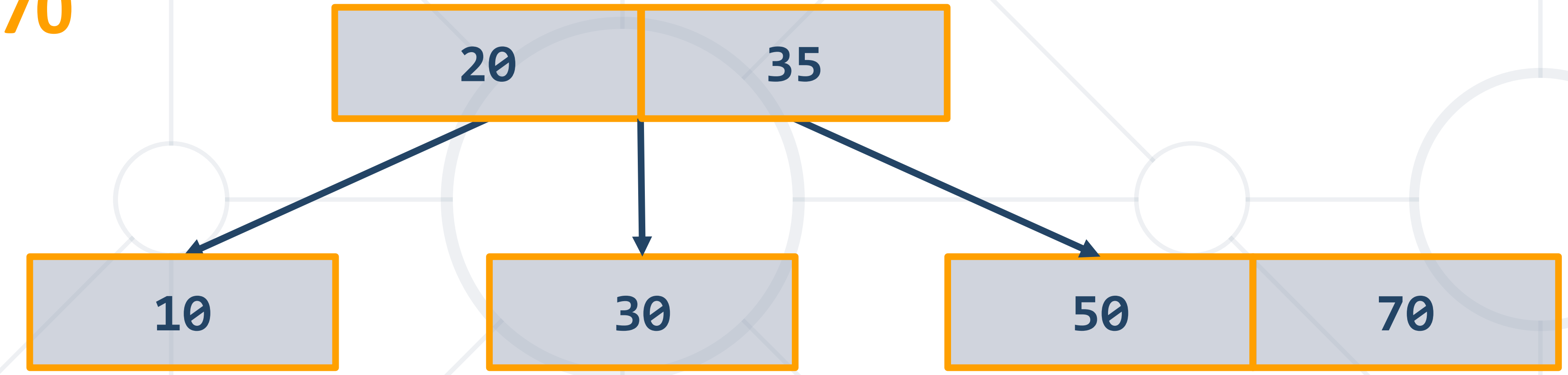
2-3 Tree Construction (6)

- Insert **70**



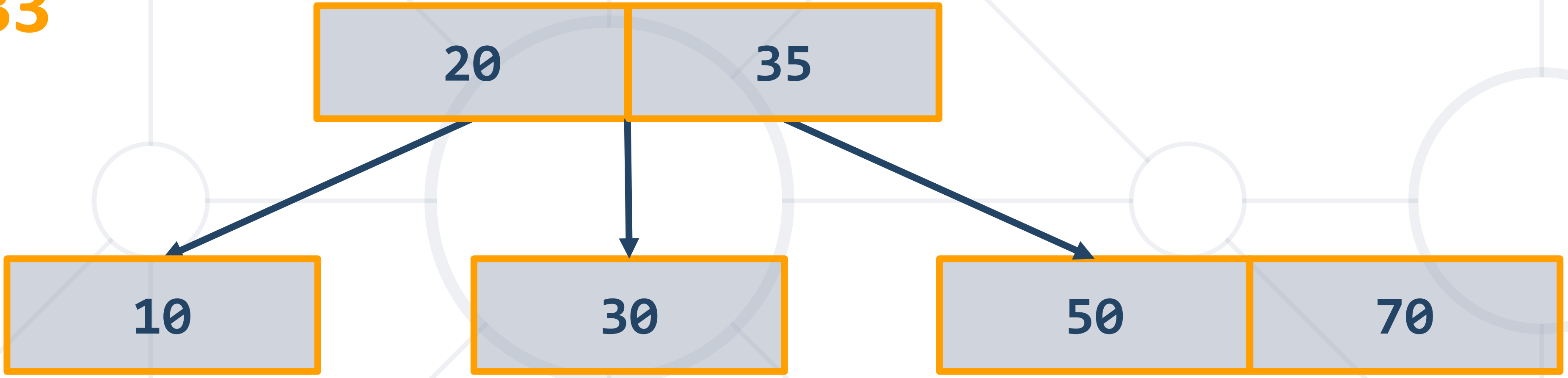
2-3 Tree Construction (6)

- Insert **70**



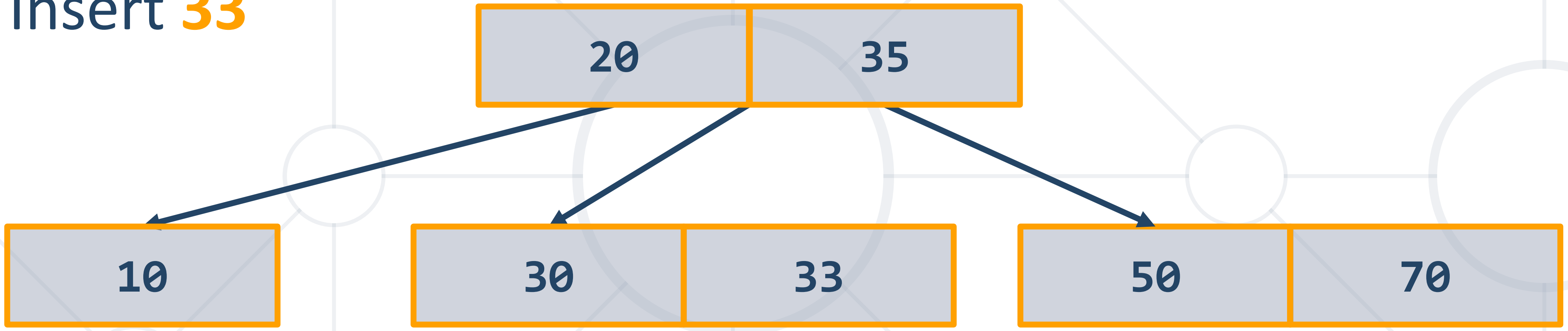
2-3 Tree Construction (7)

- Insert **33**



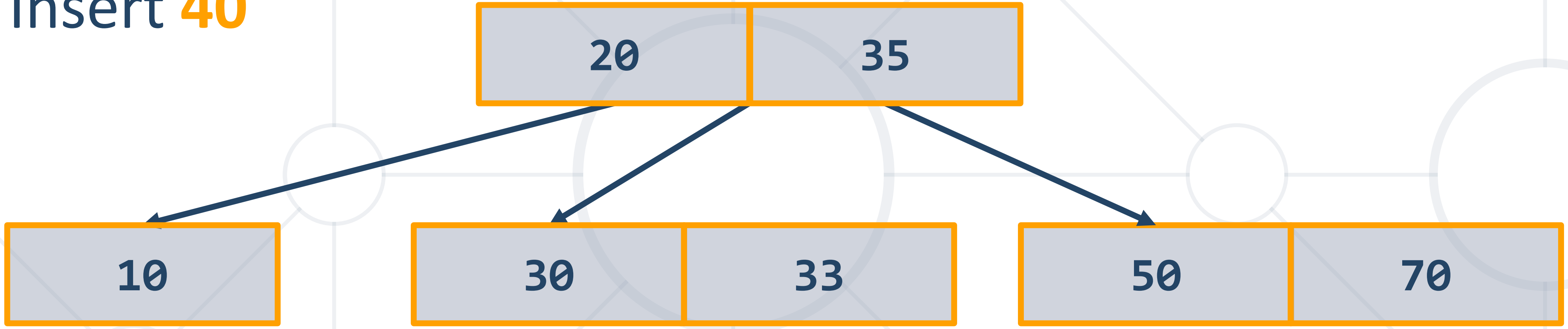
2-3 Tree Construction (7)

- Insert **33**



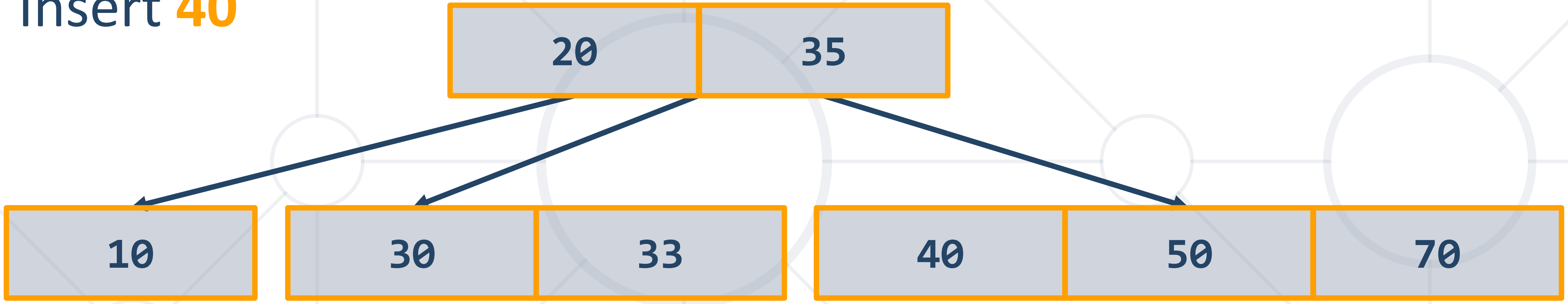
2-3 Tree Construction (8)

- Insert **40**



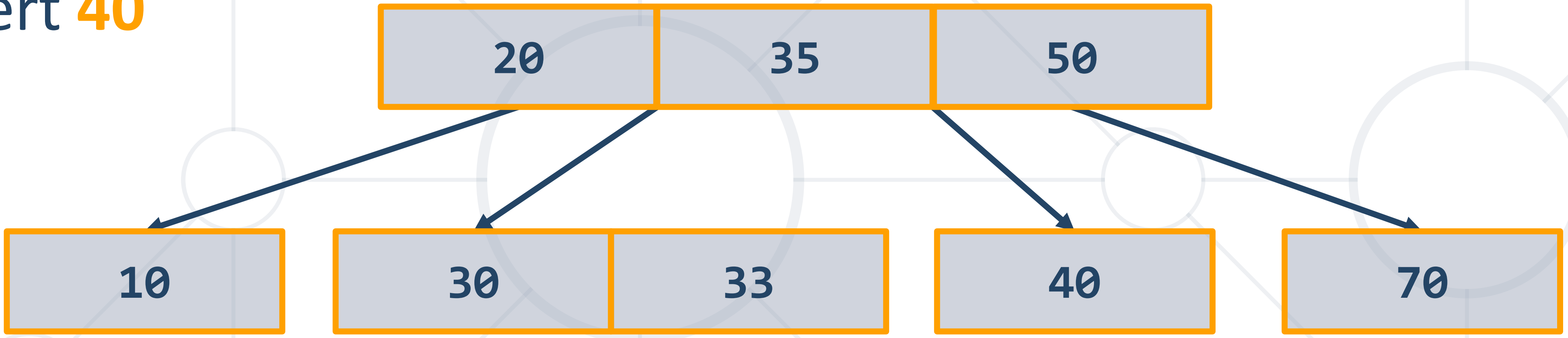
2-3 Tree Construction (8)

- Insert **40**



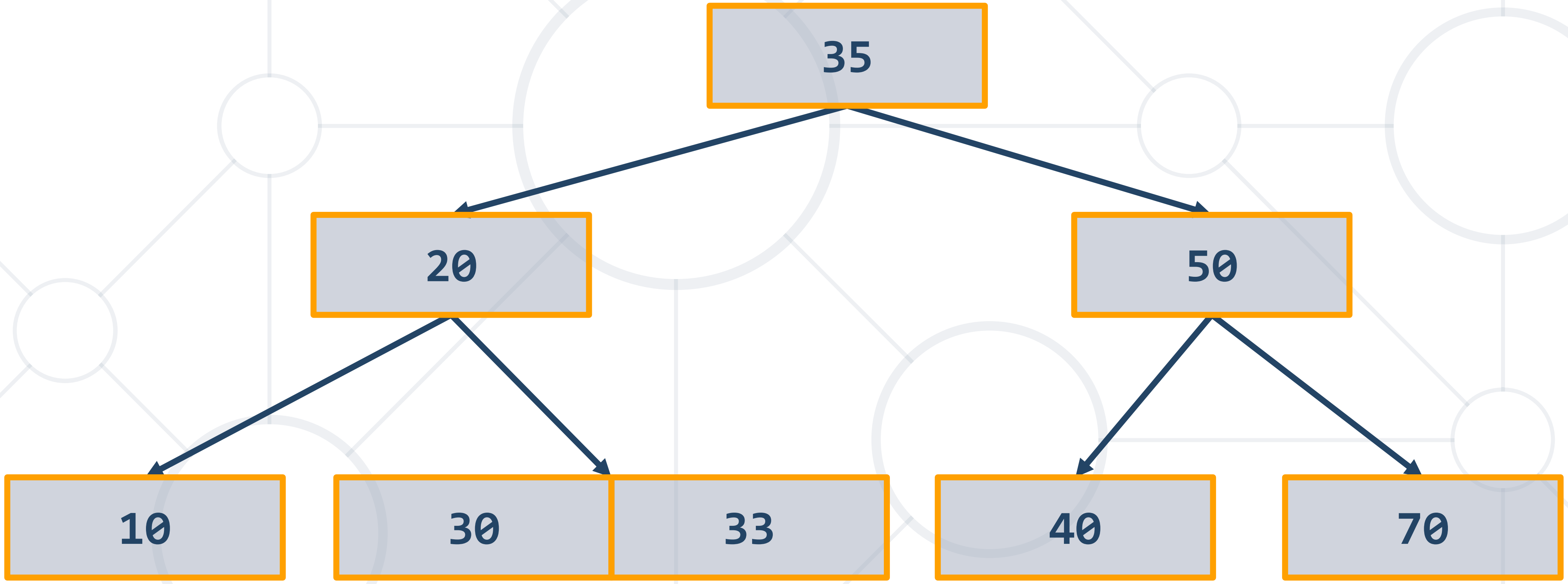
2-3 Tree Construction (8)

- Insert **40**



2-3 Tree Construction (8)

- Insert **40**



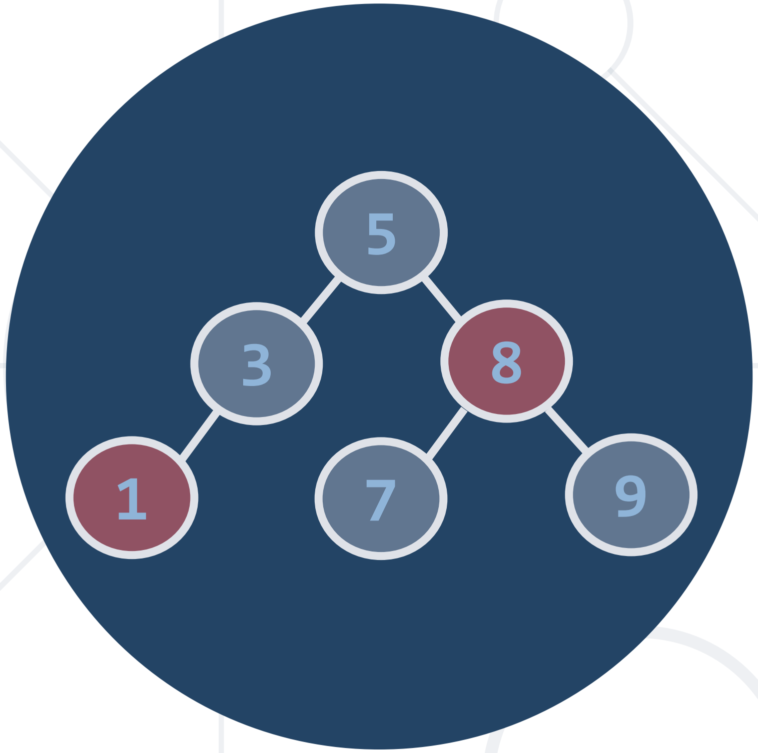
2-3 Tree Properties

- 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 - 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

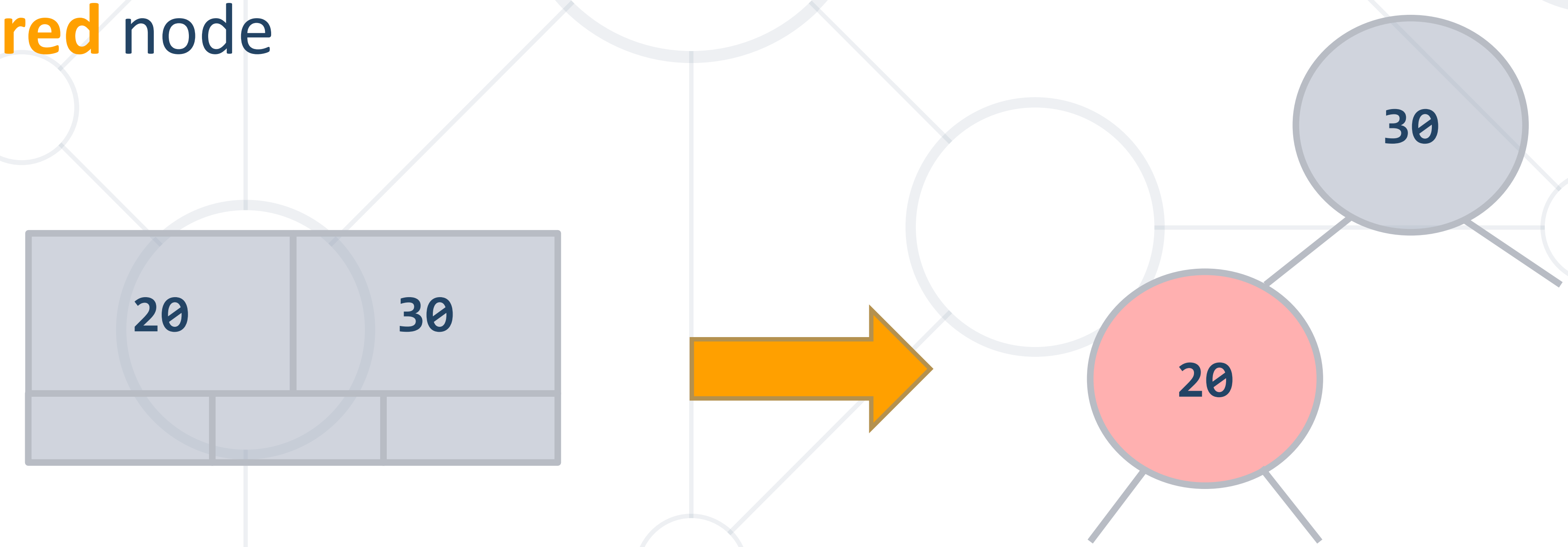


Red-Black Tree

Simple Representation of a 2-3 Tree

Red-Black Tree Definition

- Represent 2-3 tree as BST
- Use "internal" **left-leaning** links as "glue" for 3-nodes
- Nodes with values between the 2 nodes will be to the **right** of the **red** node



Red-Black Tree Properties

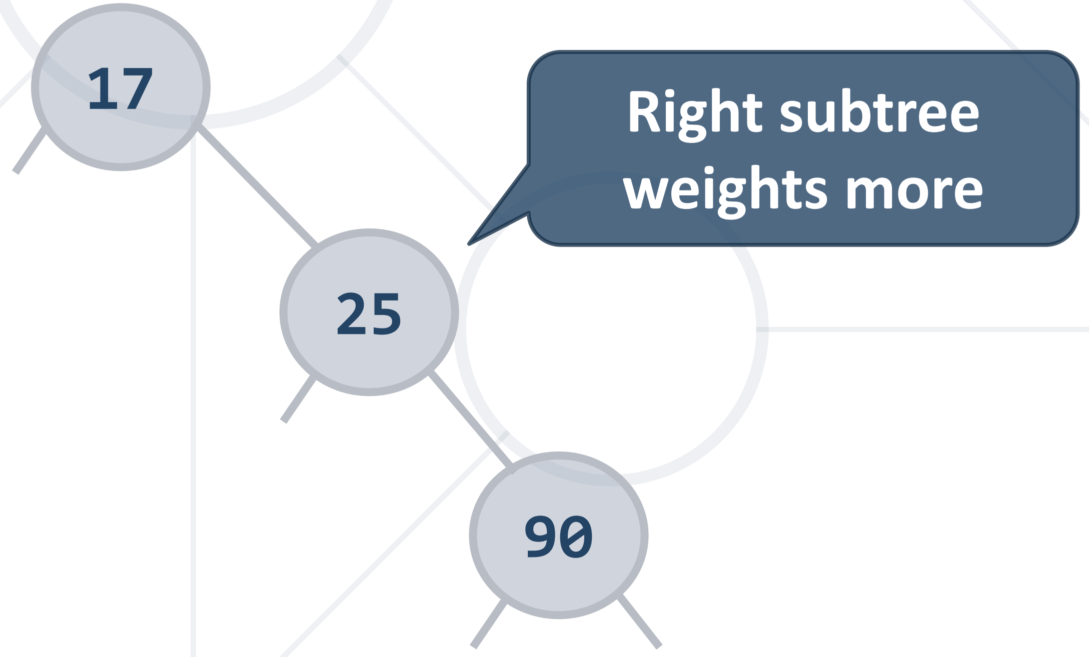
- No node has **two red links** connected to it
- Every path from the **root** to its **null leaf** nodes contains the **same** number of **black** nodes/links
- Red links **lean** left
- The root is **black**
- No path from the root to the bottom contains **two consecutive red links**
- **Visualization**

Rebalancing Trees

Rotations

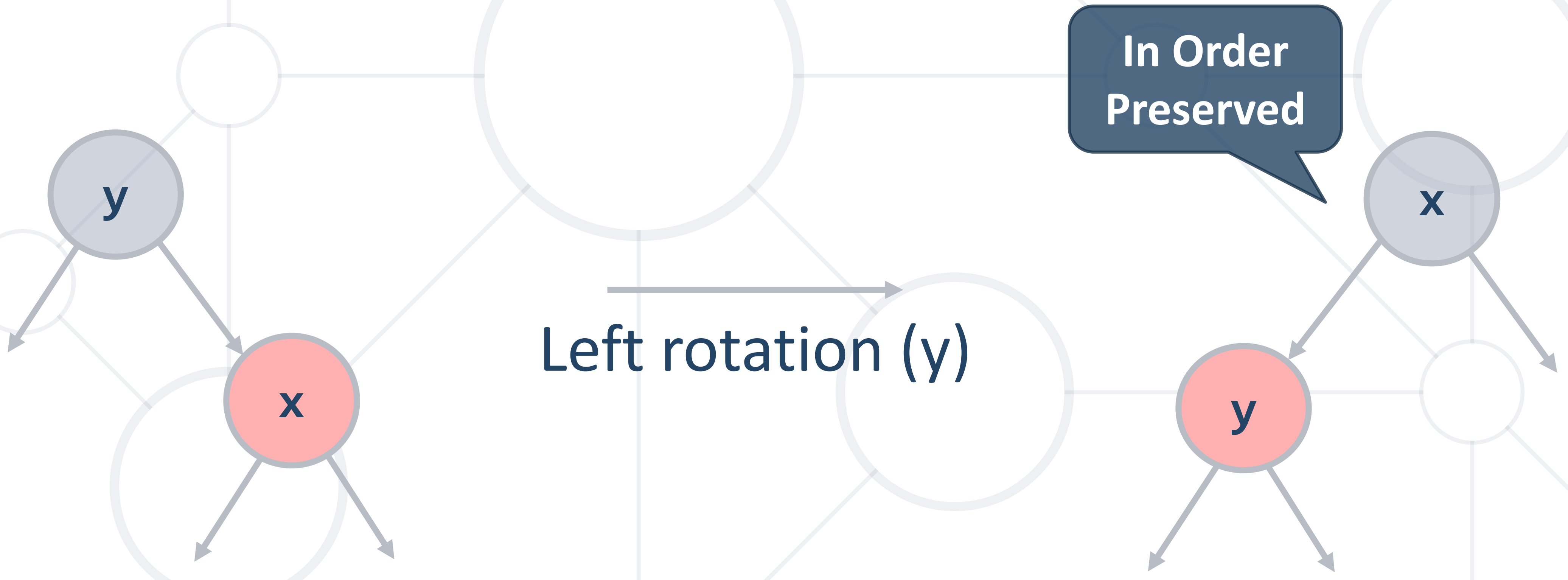
Rotations

- Rotations are used to correct the balance of a tree
- Balance can be measured in height, depth, size etc. of subtrees



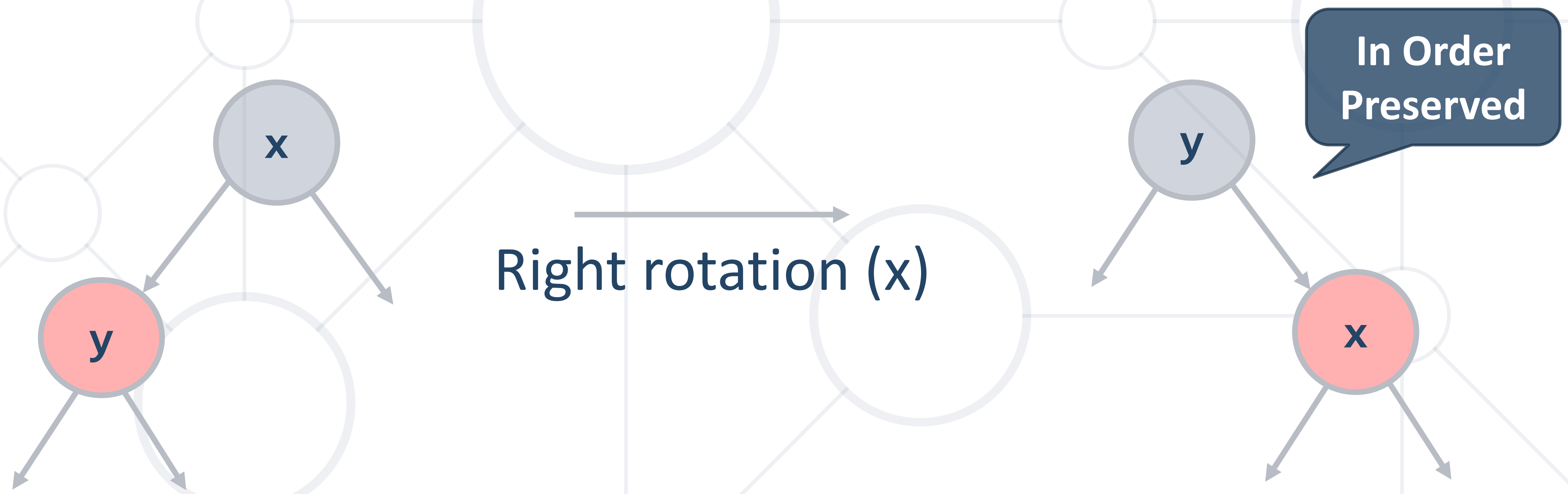
Left Rotation

- Orient a right-leaning red link to lean left



Right Rotation

- Orient a left-leaning red link to lean right (temporarily)



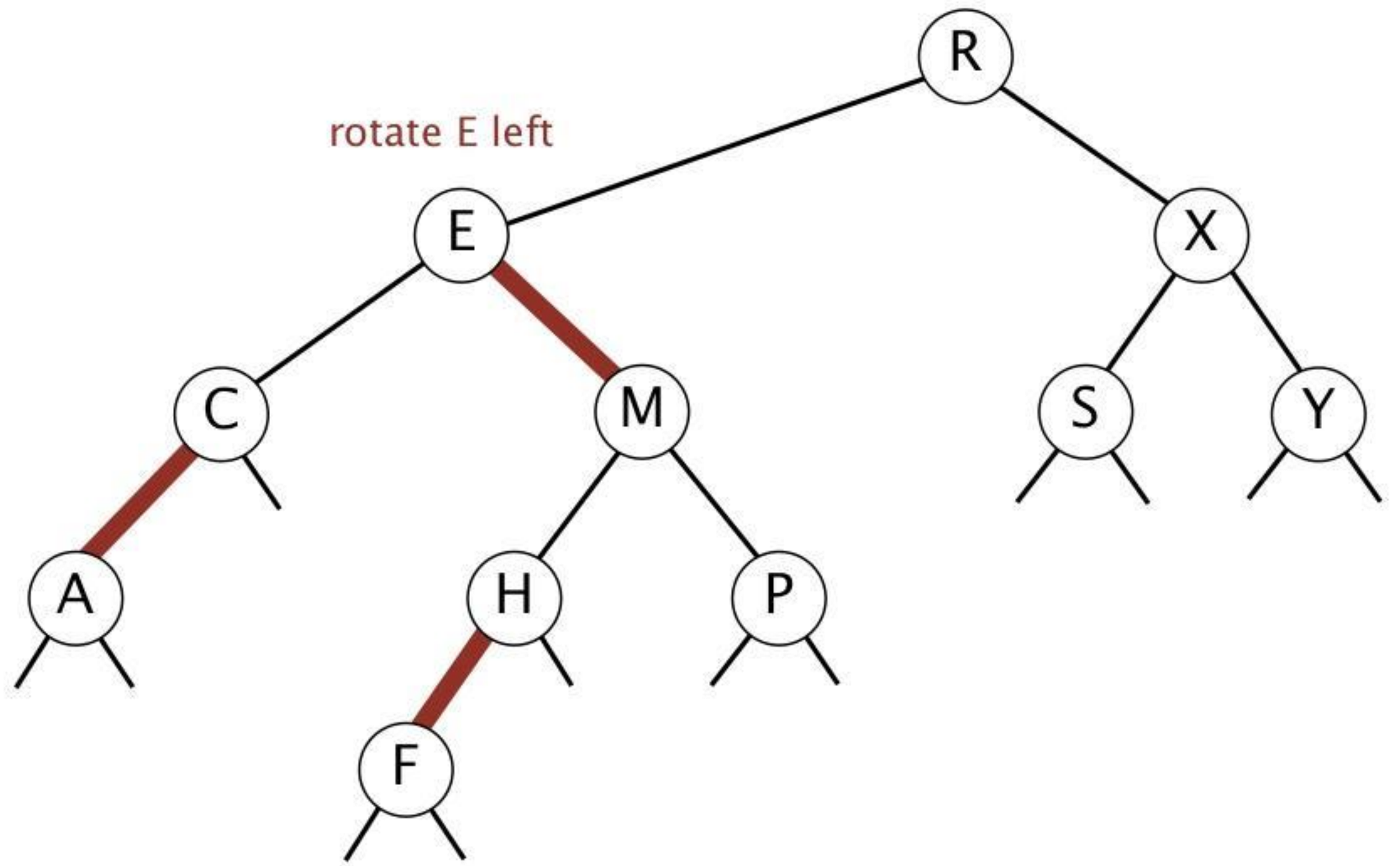
Rotations – Quiz

A. R E X C M S Y A H P F

B. R M X E H S Y C F P A

C. R M X E P S Y C H A F

D. R C X A E S Y M H P F



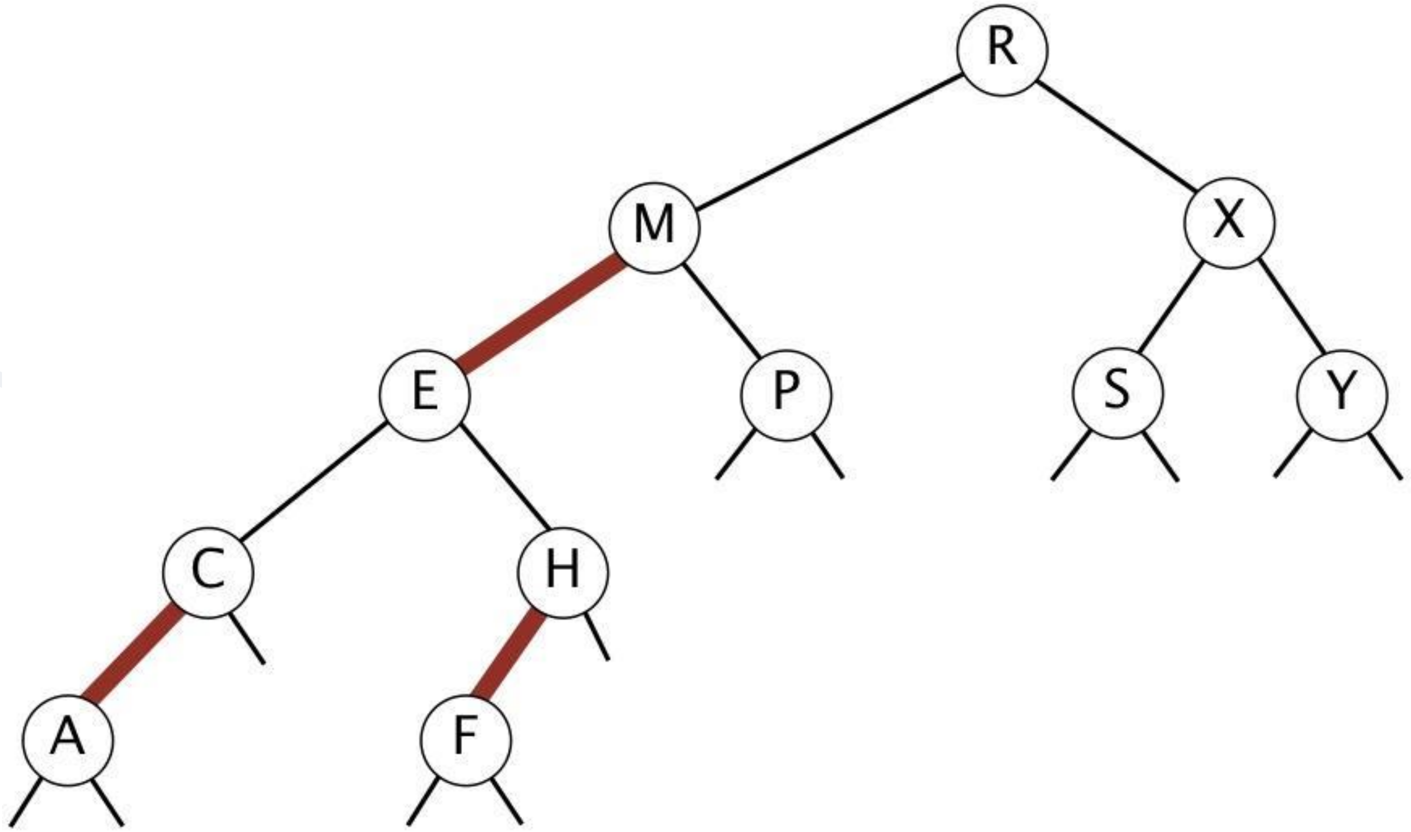
Rotations – Answer

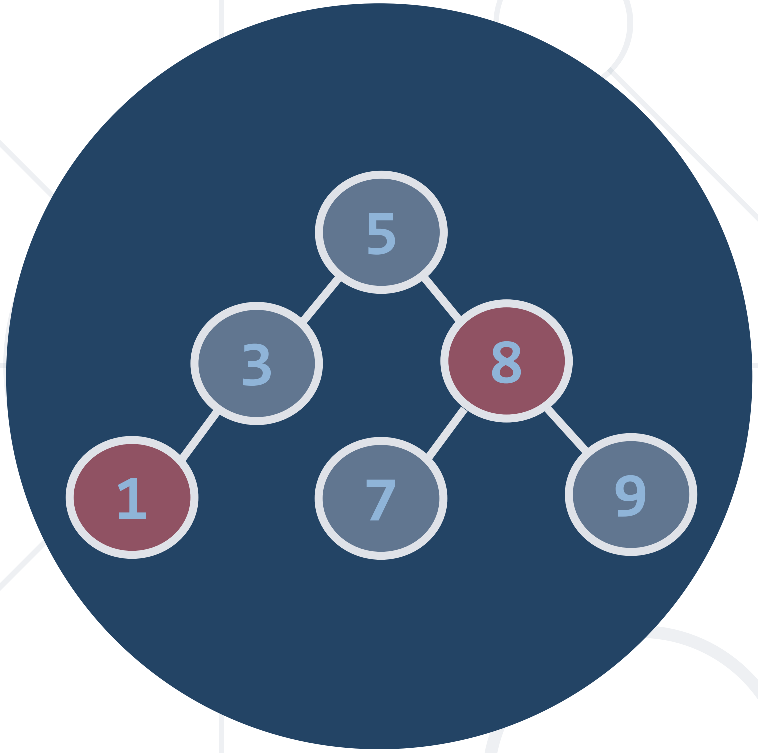
A. R E X C M S Y A H P F

B. R M X E H S Y C F P A

C. R M X E P S Y C H A F

D. R C X A E S Y M H P F





Red-Black Tree

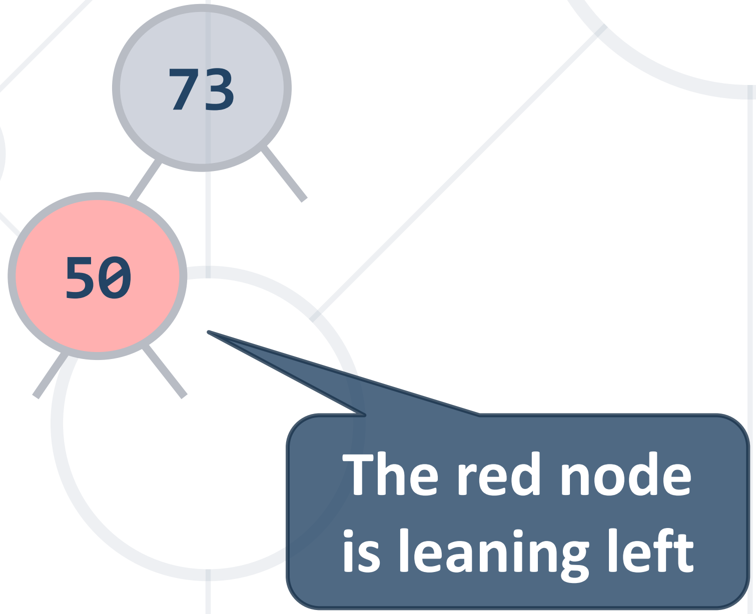
Insertion Algorithm

Insertion Algorithm

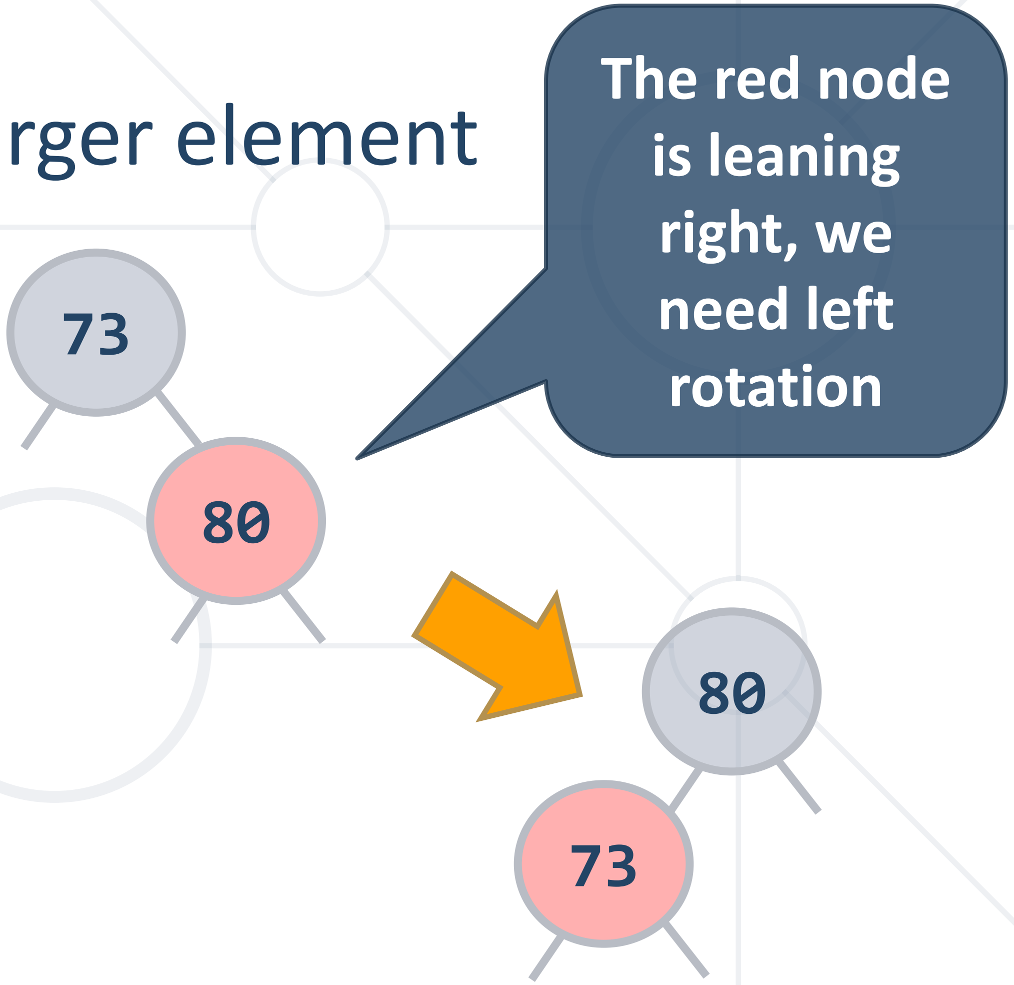
- **Locate** the node position
- Create new **red** node
- **Add** the new node to the tree
- **Balance** the tree if needed

Insertion

- Insert into a single 2-node:
- Smaller element

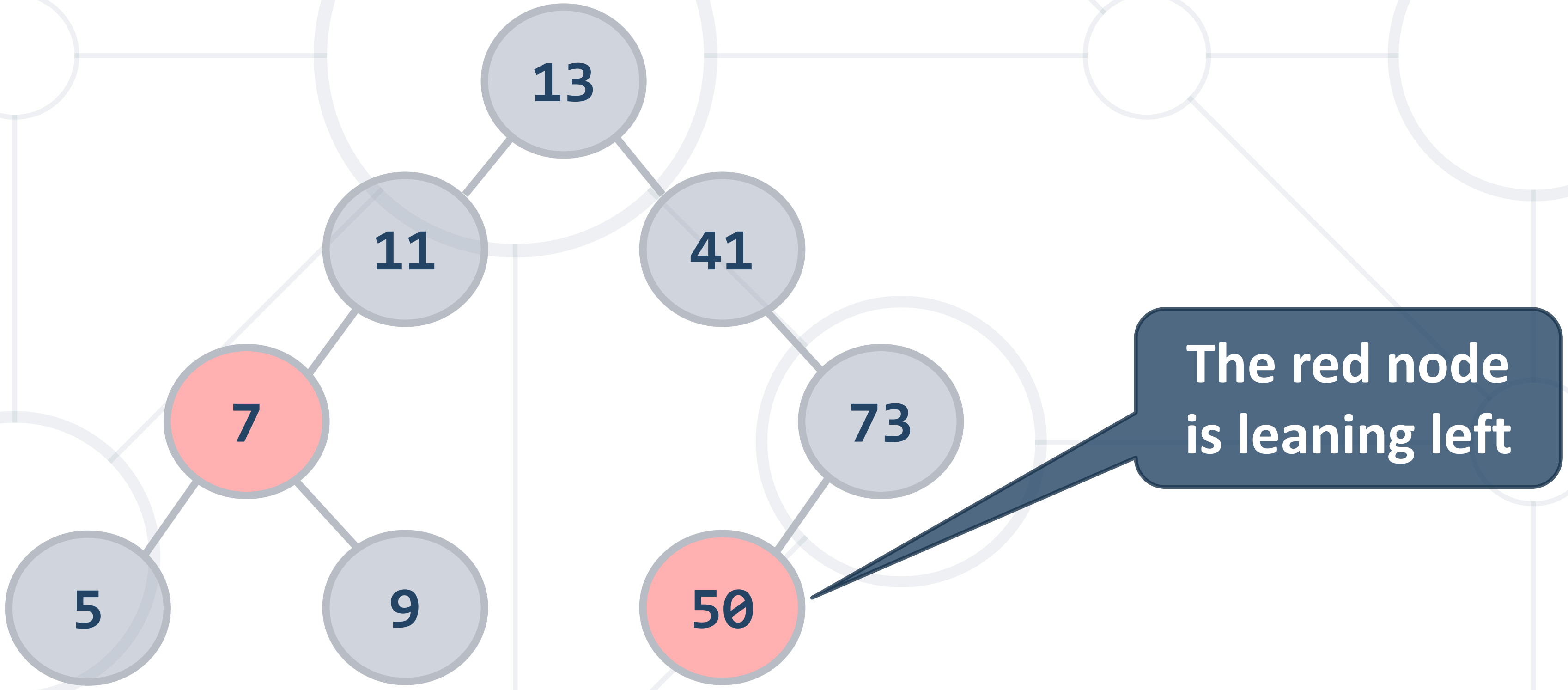


- Larger element



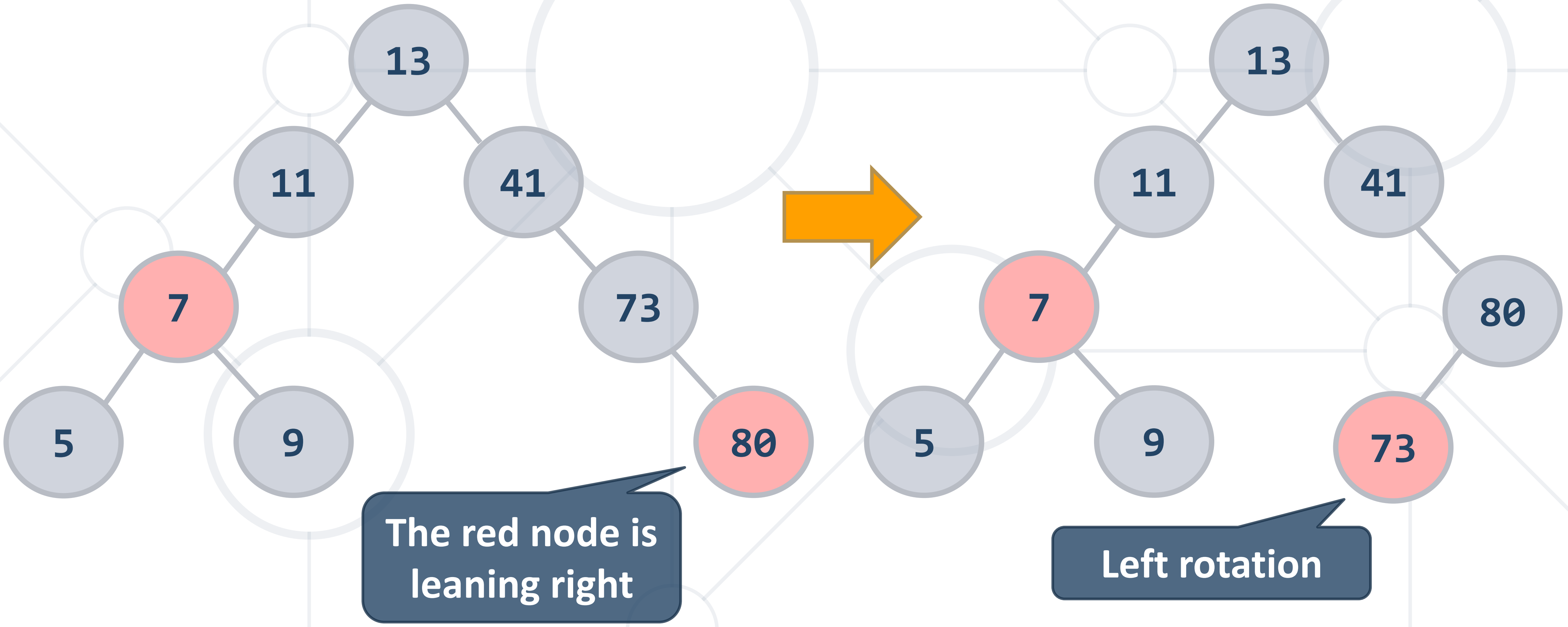
Insertion (2)

- Insert **smaller** item into a 2-node at the bottom:



Insertion (3)

- Insert **larger** item into a 2-node at the bottom:

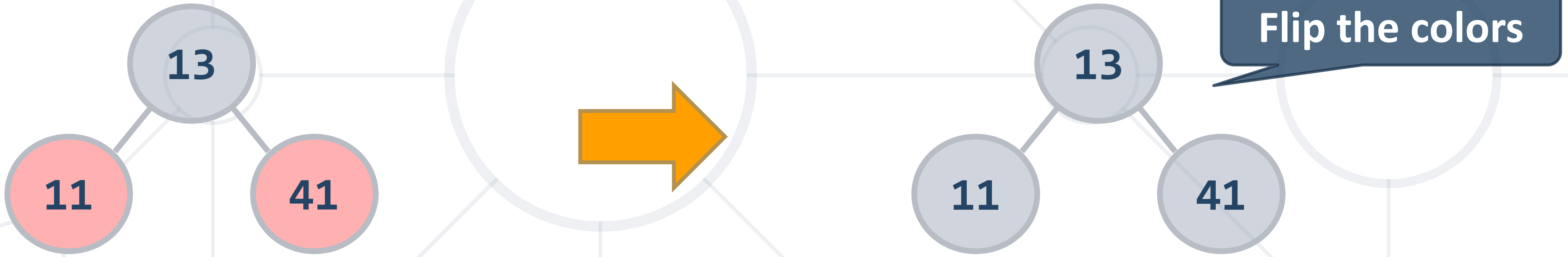


Insertion into 3-Node

- 3 cases:
  - The element is **larger** than both keys
  - The element is **smaller** than both keys
  - The element is **between** the 2 keys

Insertion into 3-Node (1)

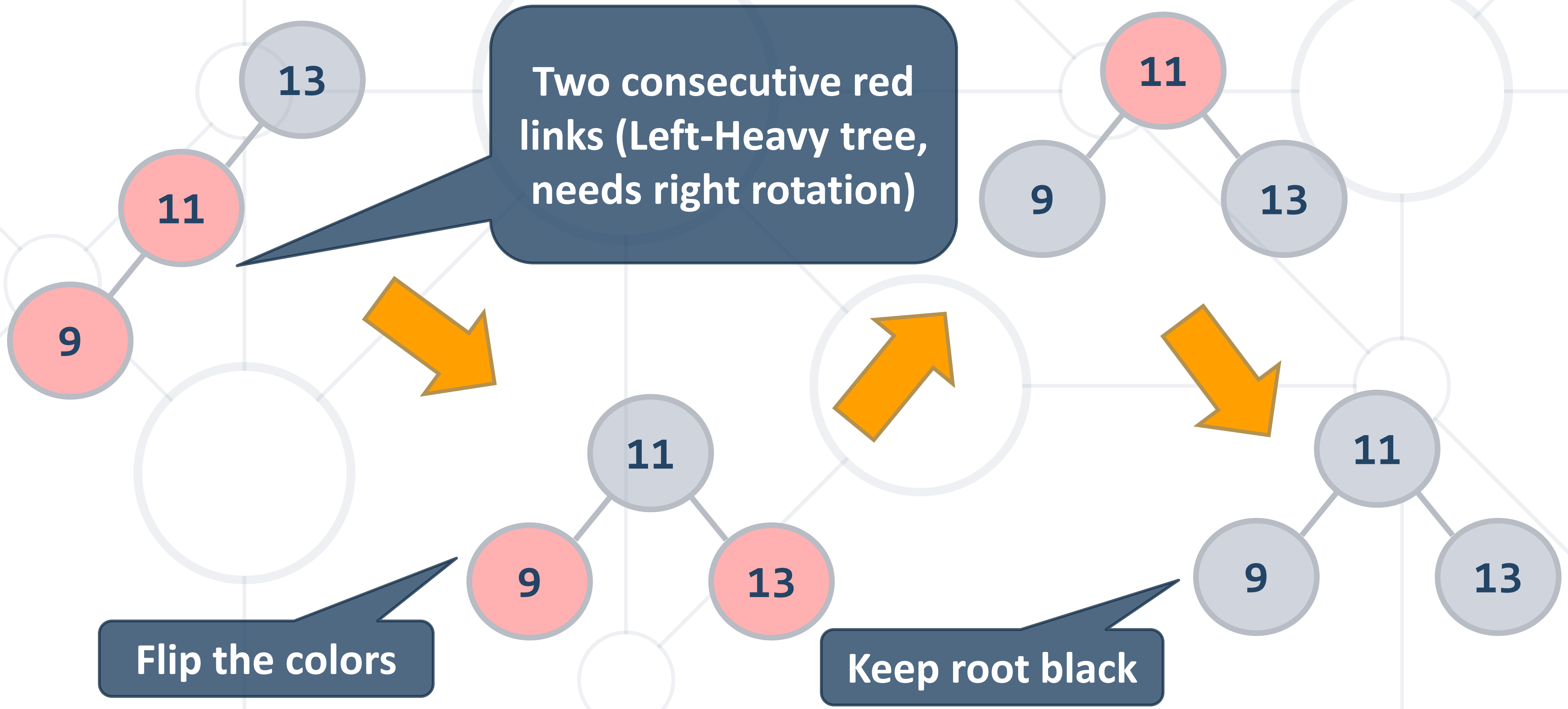
- Larger than both keys:



- Flipping the colors increases the tree height, which maintains the 1-1 correspondence to 2-3 trees

Insertion into 3-Node (2)

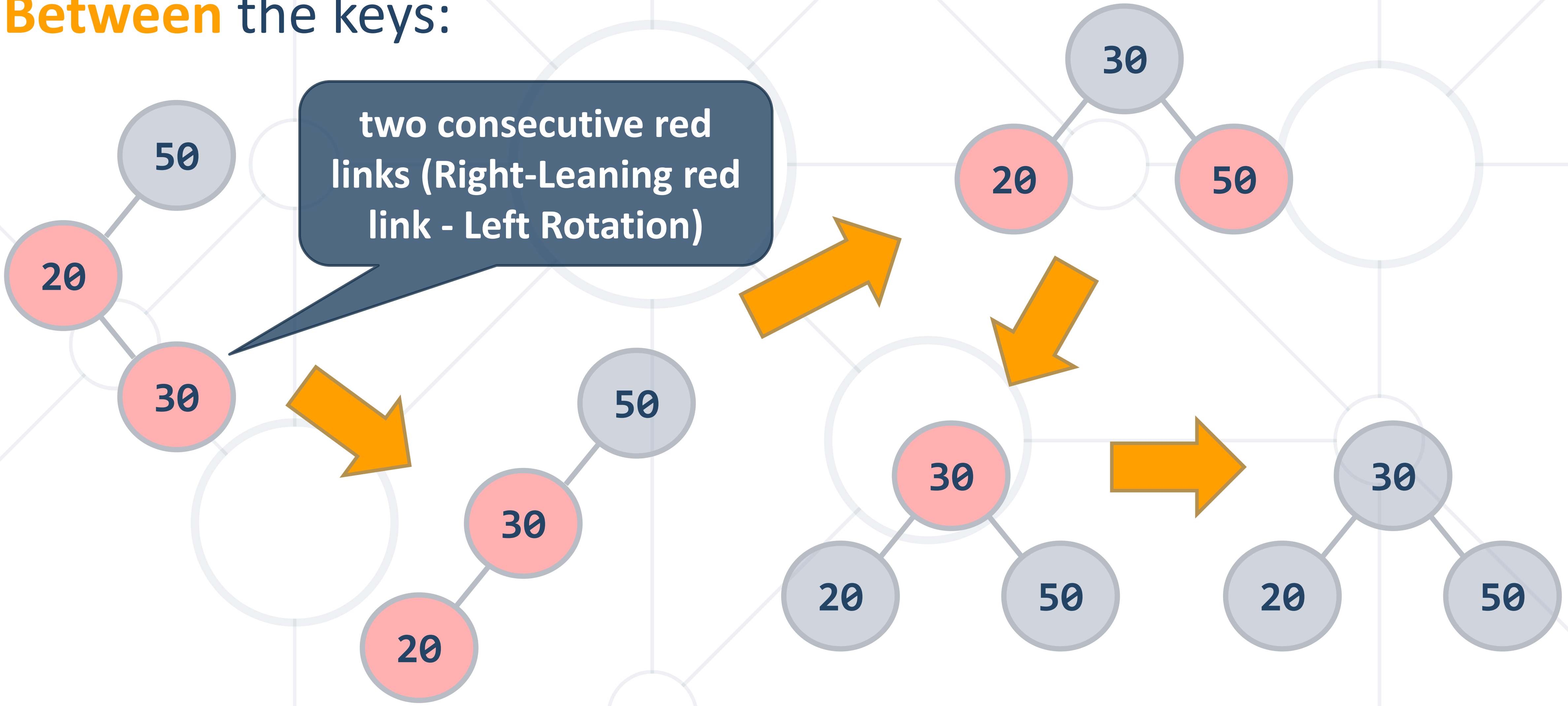
- **Smaller** than both keys:



13
11
9
Two consecutive red links (Left-Heavy tree, needs right rotation)
11
9
13
11
9
13
11
9
13
Flip the colors
Keep root black

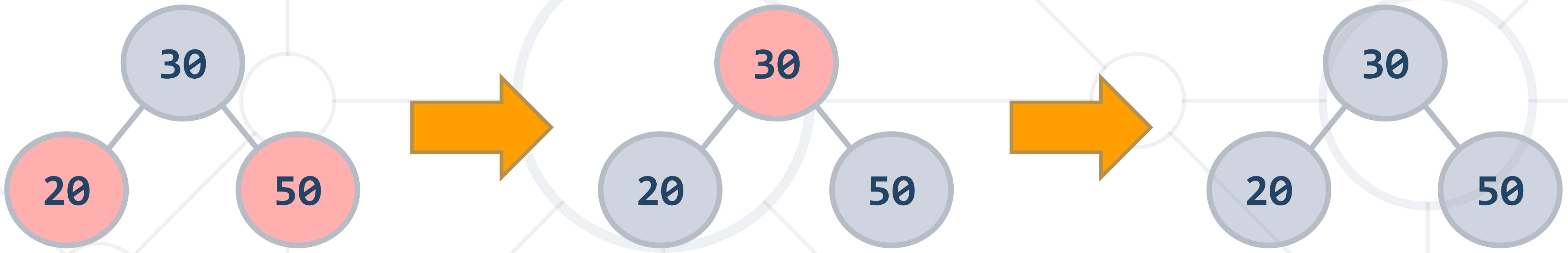
Insertion into 3-Node (3)

- **Between** the keys:


50
20
30
two consecutive red links (Right-Leaning red link - Left Rotation)
50
30
20
30
20
50
30
20
50
30
20
50

Keeping Black Root

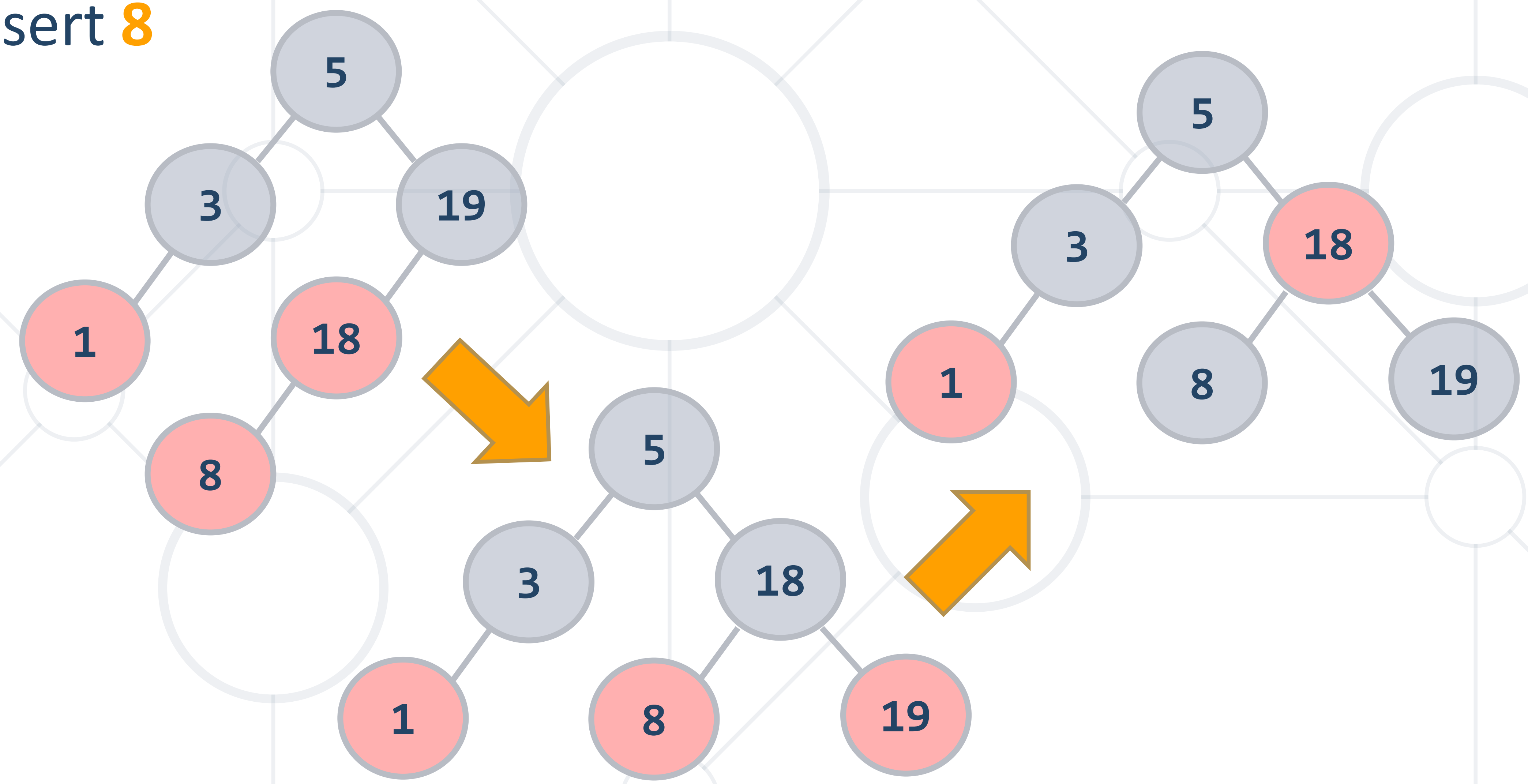
- Insert on a single node (root):



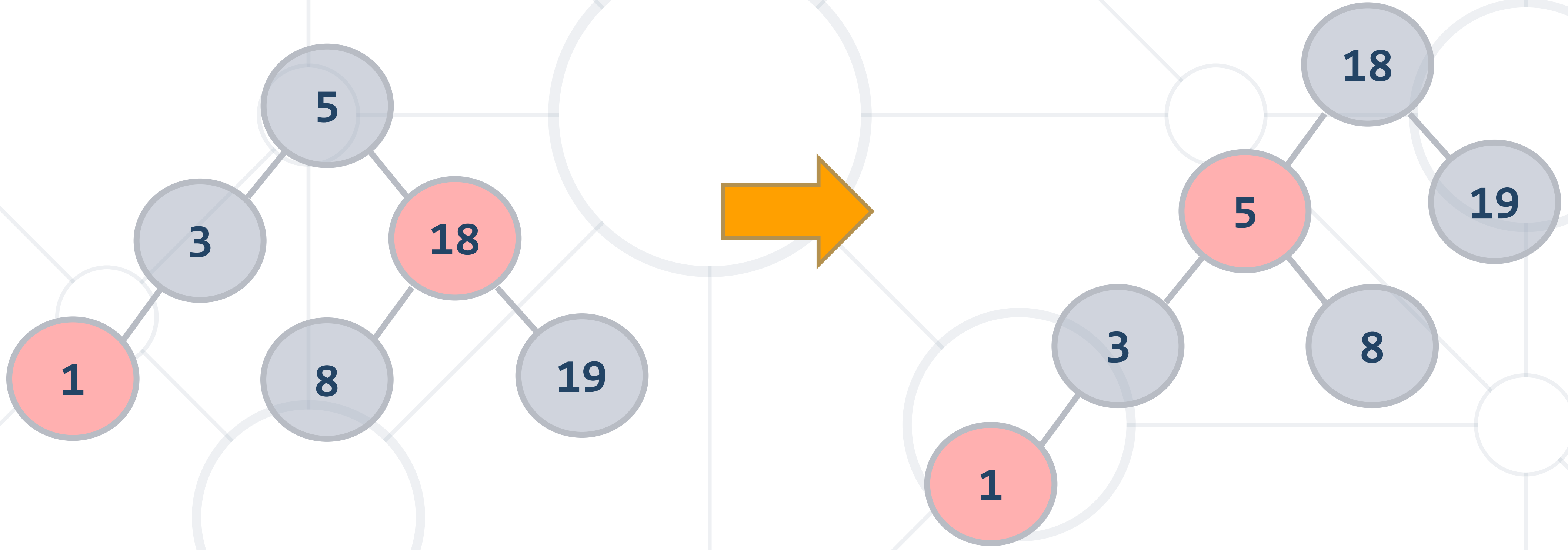
- Each time the root switches colors, the height of the tree is increased

Insert into 3-Node at the Bottom

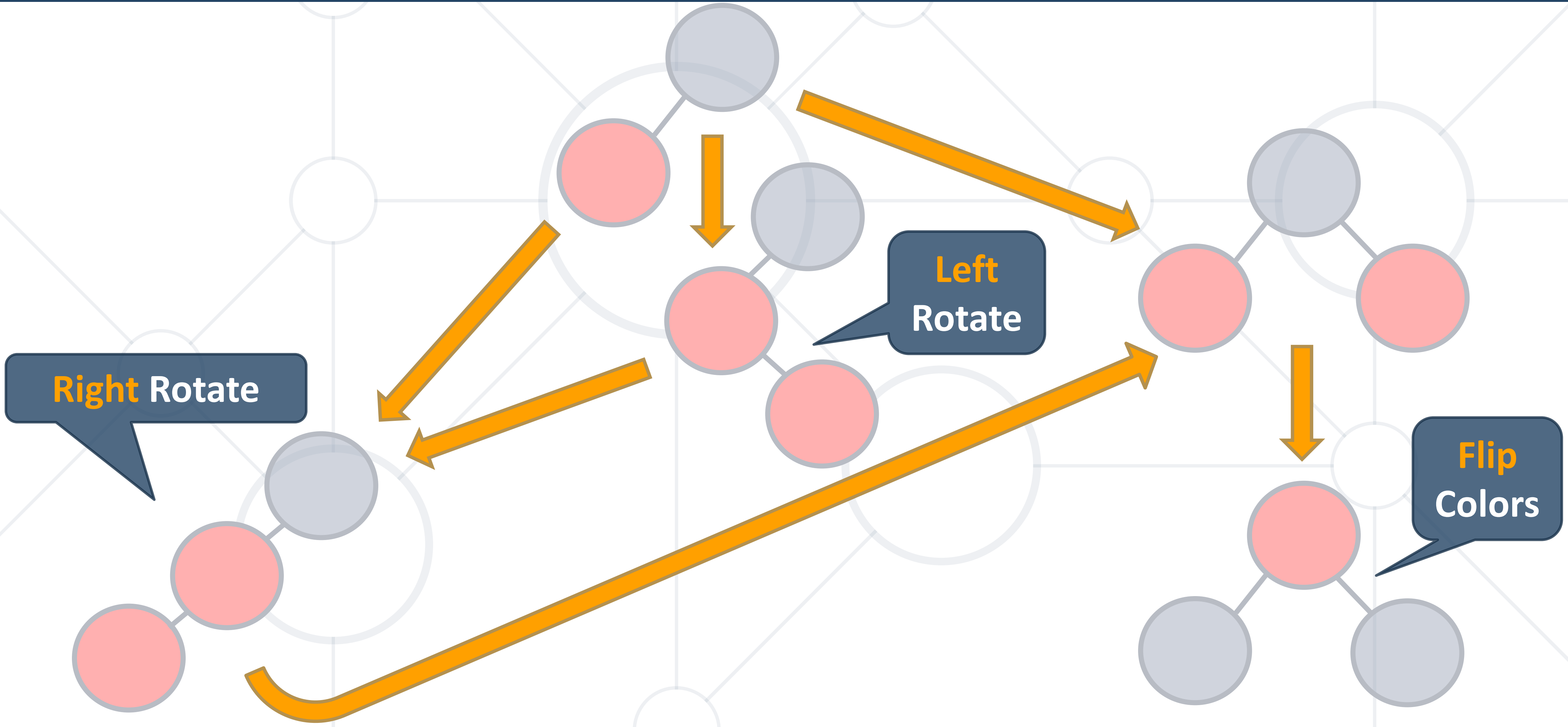
- Insert **8**



Insert into 3-Node at the Bottom (2)



Overall Insertion Process



Red-Black Tree – Quiz

TIME:

- Suppose that you insert **n** keys in ascending order into a red-black BST. What is the height of the resulting tree?
  - Constant
  - Logarithmic
  - Linear
  - Linearithmic

Red-Black Tree – Answer

- Suppose that you insert **n** keys in ascending order into a red-black BST. What is the height of the resulting tree?
  - Constant
  - Logarithmic ✓
  - Linear
  - Linearithmic

The height of any red–black BST on n keys (regardless of the order of insertion) is guaranteed to be between lg n and 2 lg n

Red-Black 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
Red-Black	2 lg N	2 lg N	2 lg N	lg N	lg N

Summary

- B-Trees can be **efficiently** stored on disks
- 2-3 tree is **B-Tree** of order **3**
  - Not **perfectly** balanced
  - Performs **local** transformations
- Red-Black tree is a simple representation of a 2-3 tree
  - Performs local 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