Binary Search Trees

CS400

Peyman Morteza

Summer 2023

Review: Trees

Review: Trees

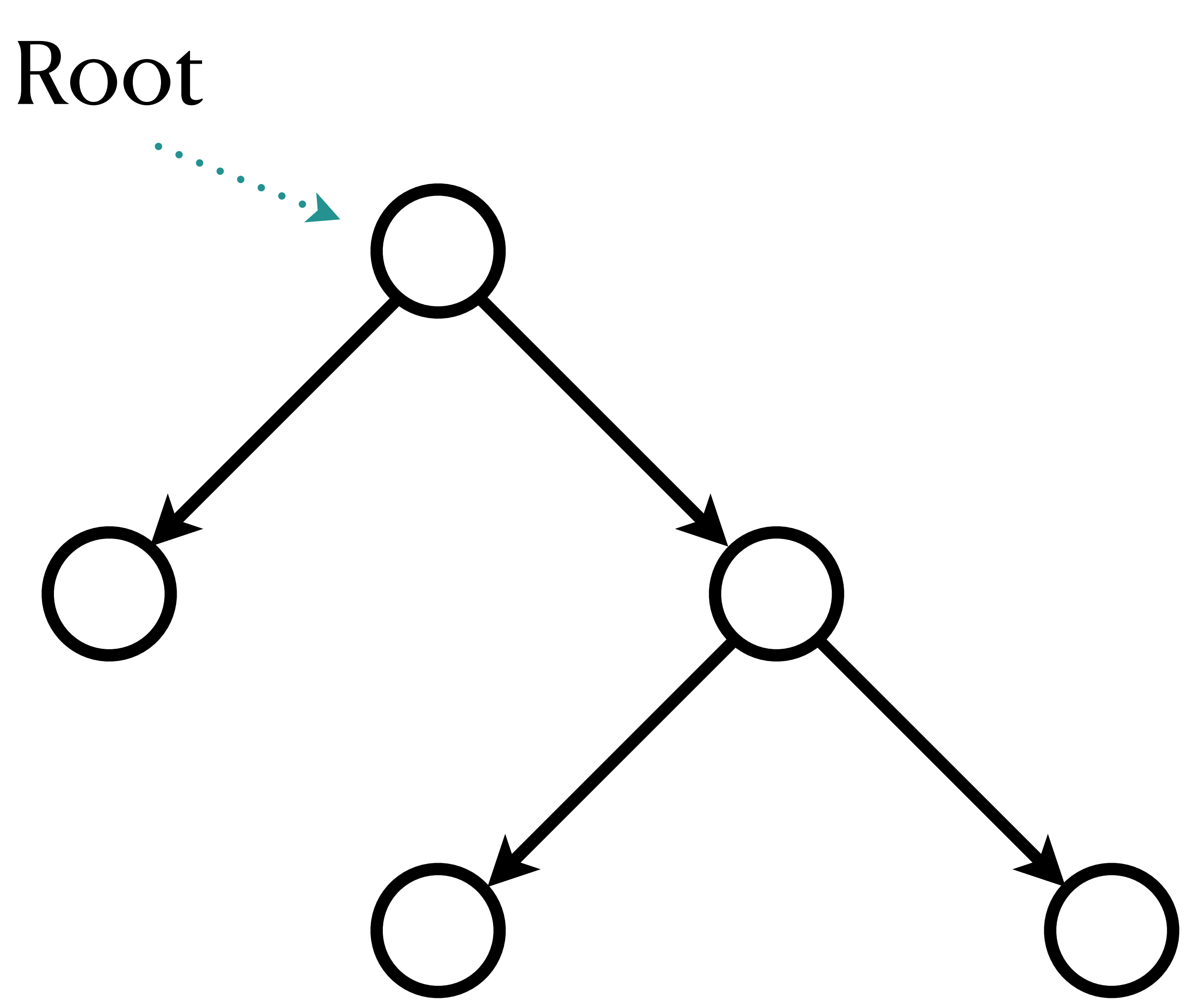
A tree is a data structure with the following properties:

- One distinguished node is designated as the root
- Every other node (except root node) has ***exactly*** one parent

Review: Trees

A tree is a data structure with the following properties:

- One distinguished node is designated as the root
- Every other node (except root node) has ***exactly*** one parent

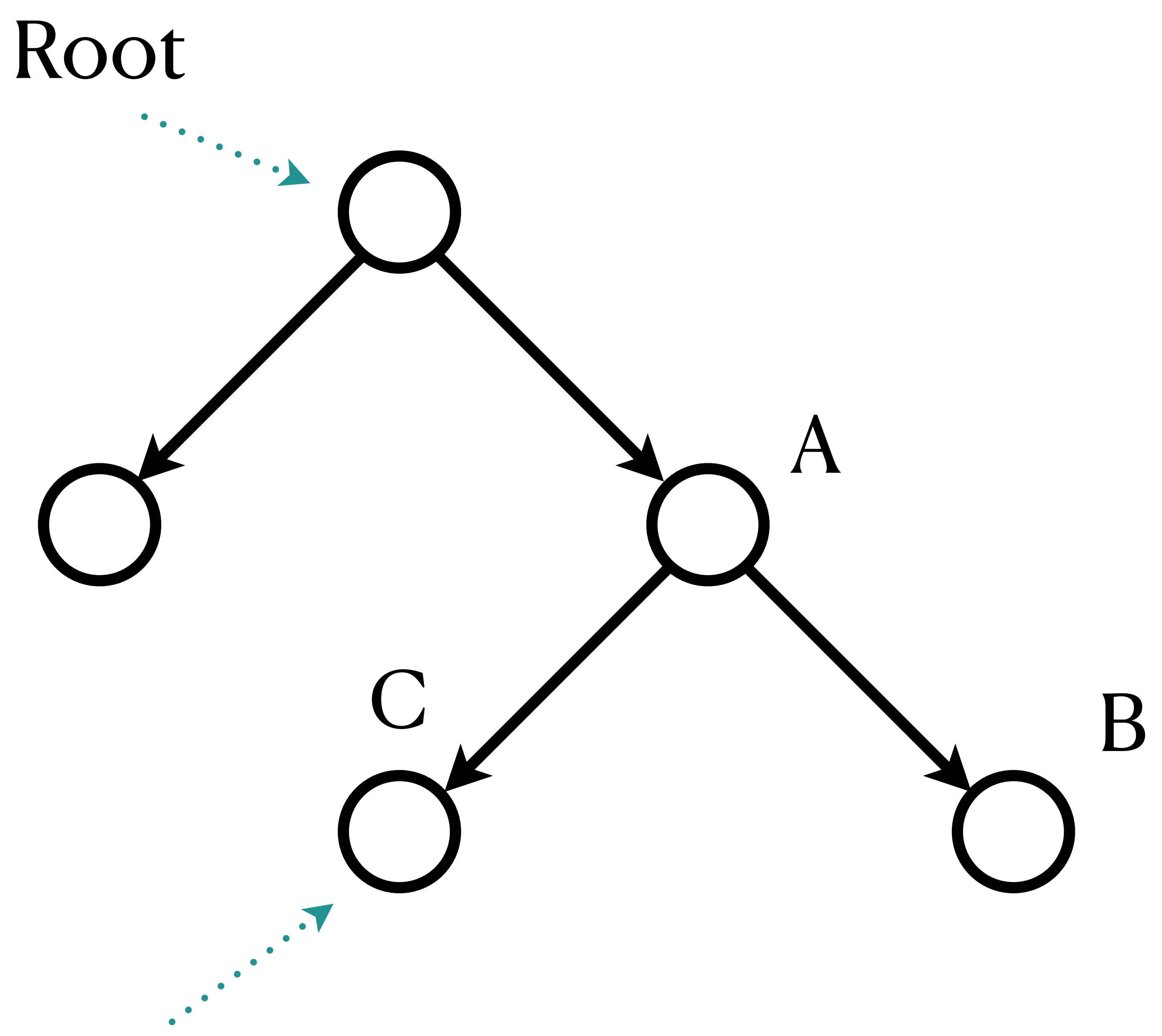


Every node has exactly one parent

Review: Trees

A tree is a data structure with the following properties:

- One distinguished node is designated as the root
- Every other node (except root node) has ***exactly*** one parent

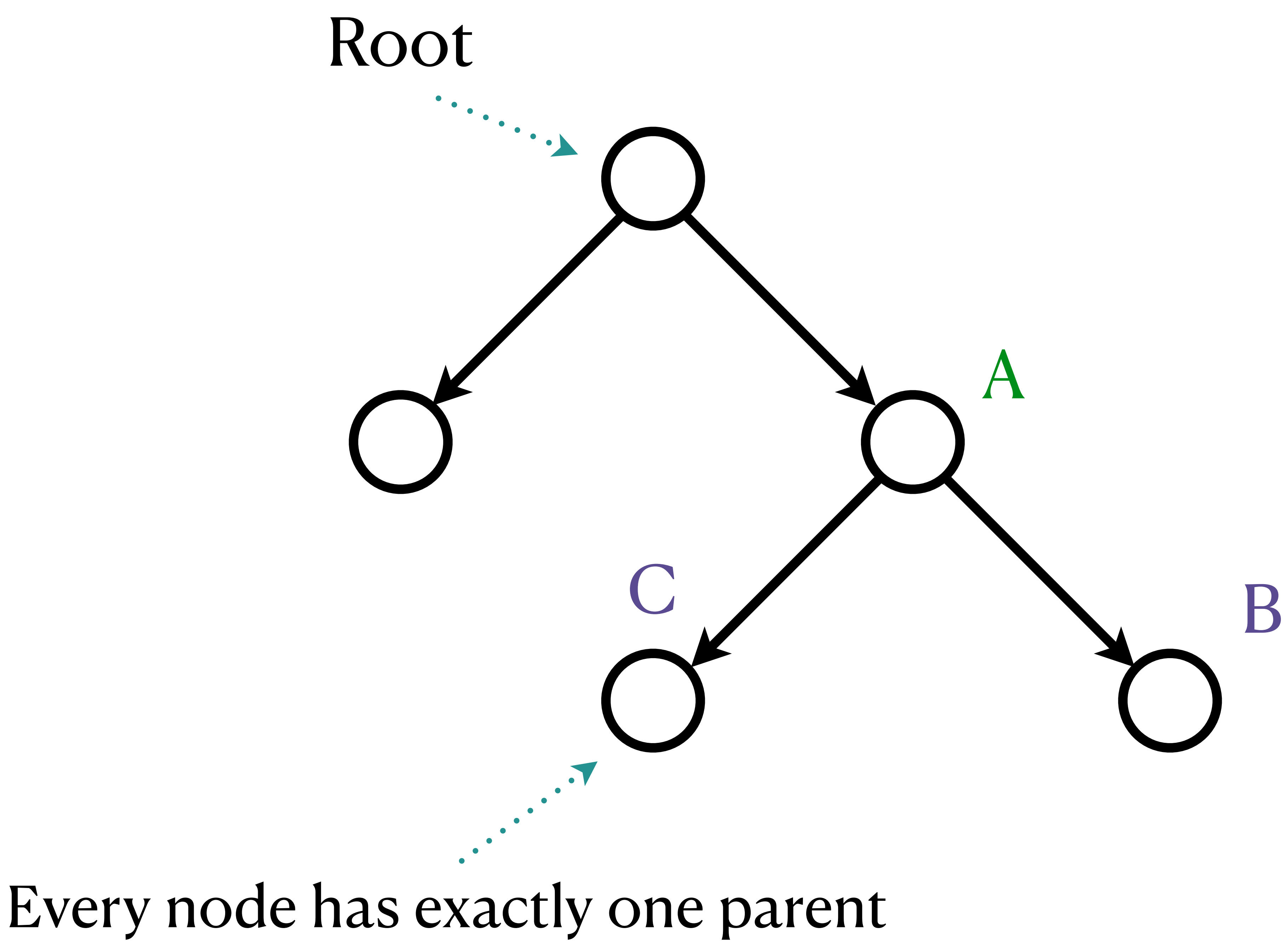


Every node has exactly one parent

Review: Trees

A tree is a data structure with the following properties:

- One distinguished node is designated as the root
- Every other node (except root node) has ***exactly*** one parent

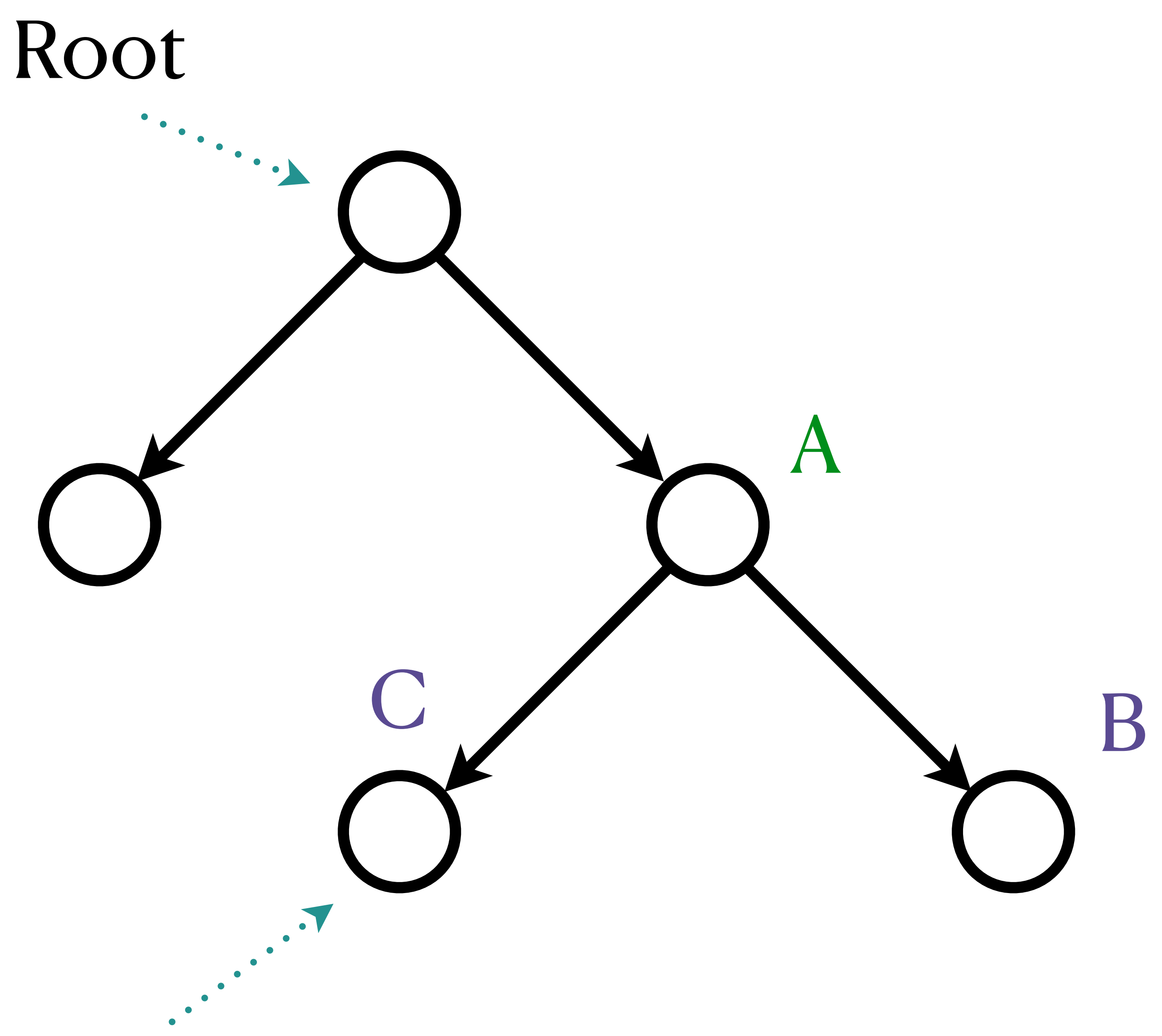


A is parent of B and C

Review: Trees

A tree is a data structure with the following properties:

- One distinguished node is designated as the root
- Every other node (except root node) has ***exactly*** one parent



Every node has exactly one parent

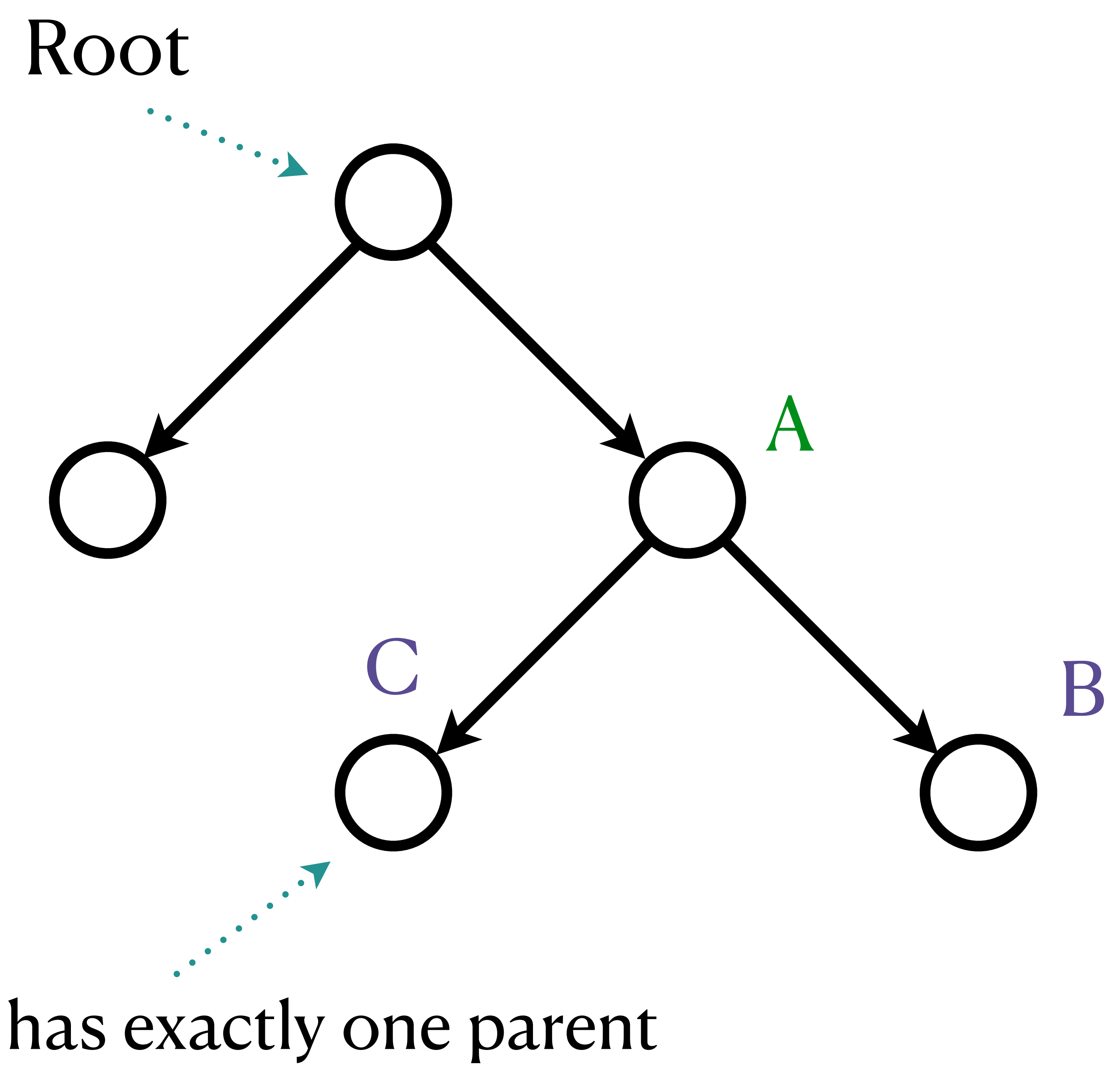
A is parent of B and C

B and C are children of A

Review: Trees

A tree is a data structure with the following properties:

- One distinguished node is designated as the root
- Every other node (except root node) has ***exactly*** one parent



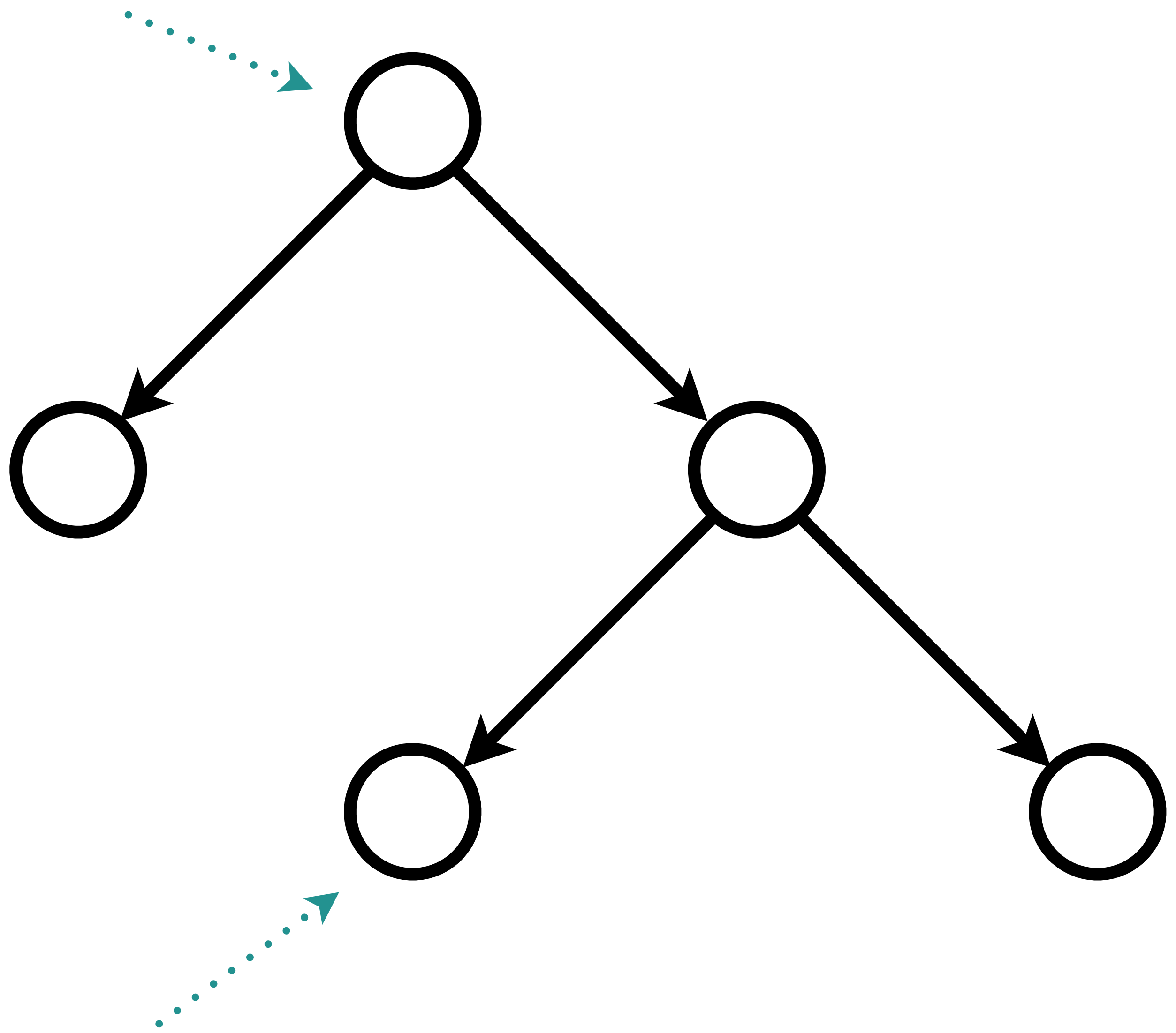
A is parent of B and C

Review: Trees

A tree is a data structure with the following properties:

- One distinguished node is designated as the root
- Every other node (except root node) has ***exactly*** one parent

Root



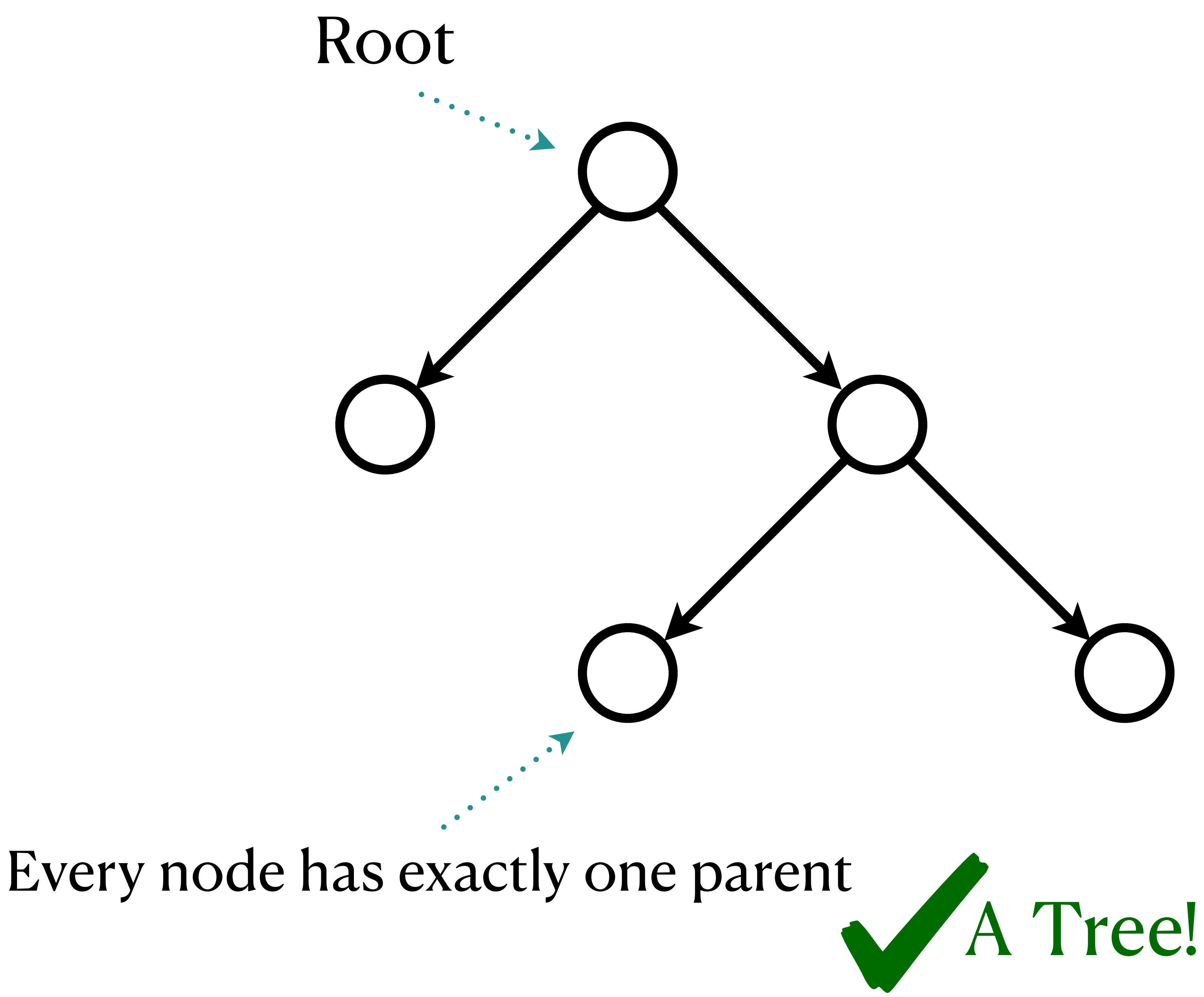
Every node has exactly one parent

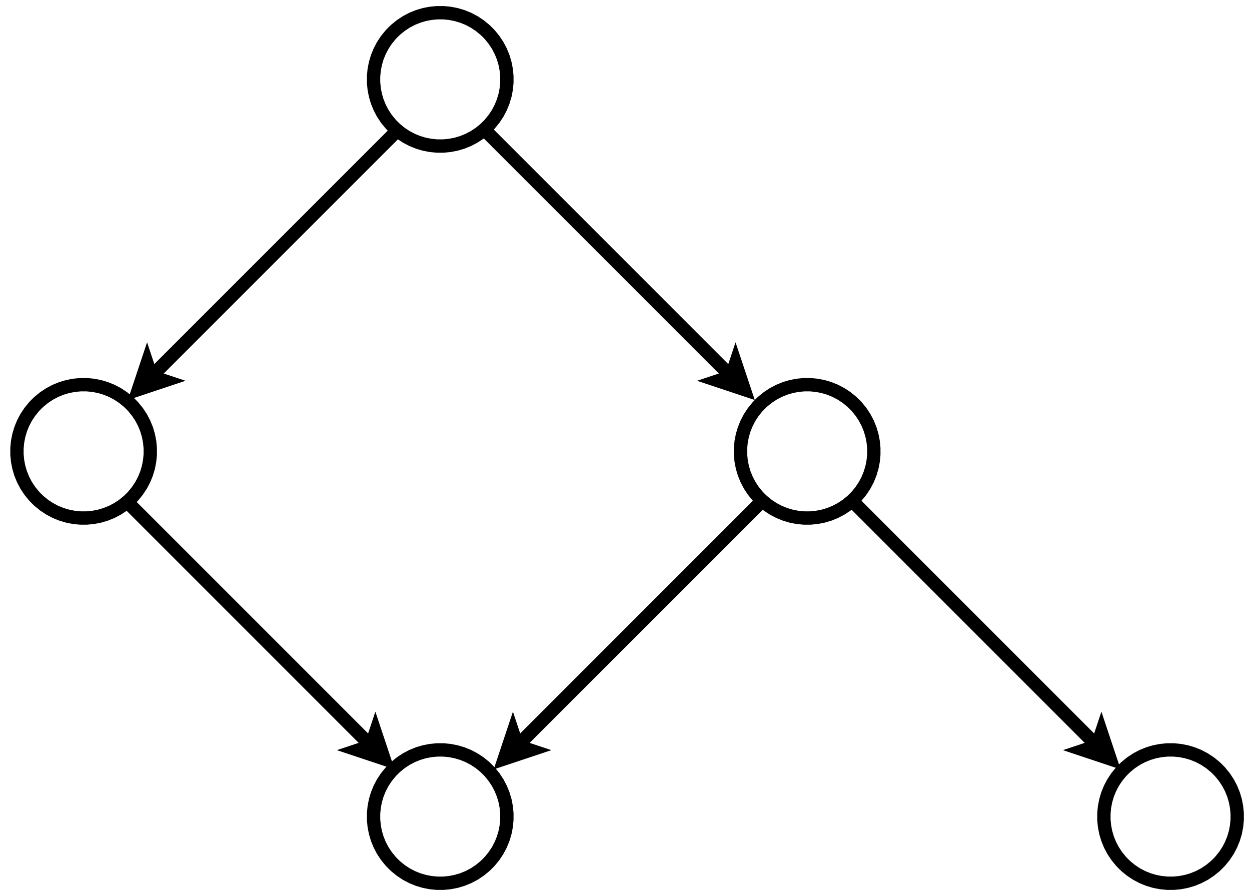
✓ A Tree!

Review: Trees

A tree is a data structure with the following properties:

- One distinguished node is designated as the root
- Every other node (except root node) has ***exactly*** one parent



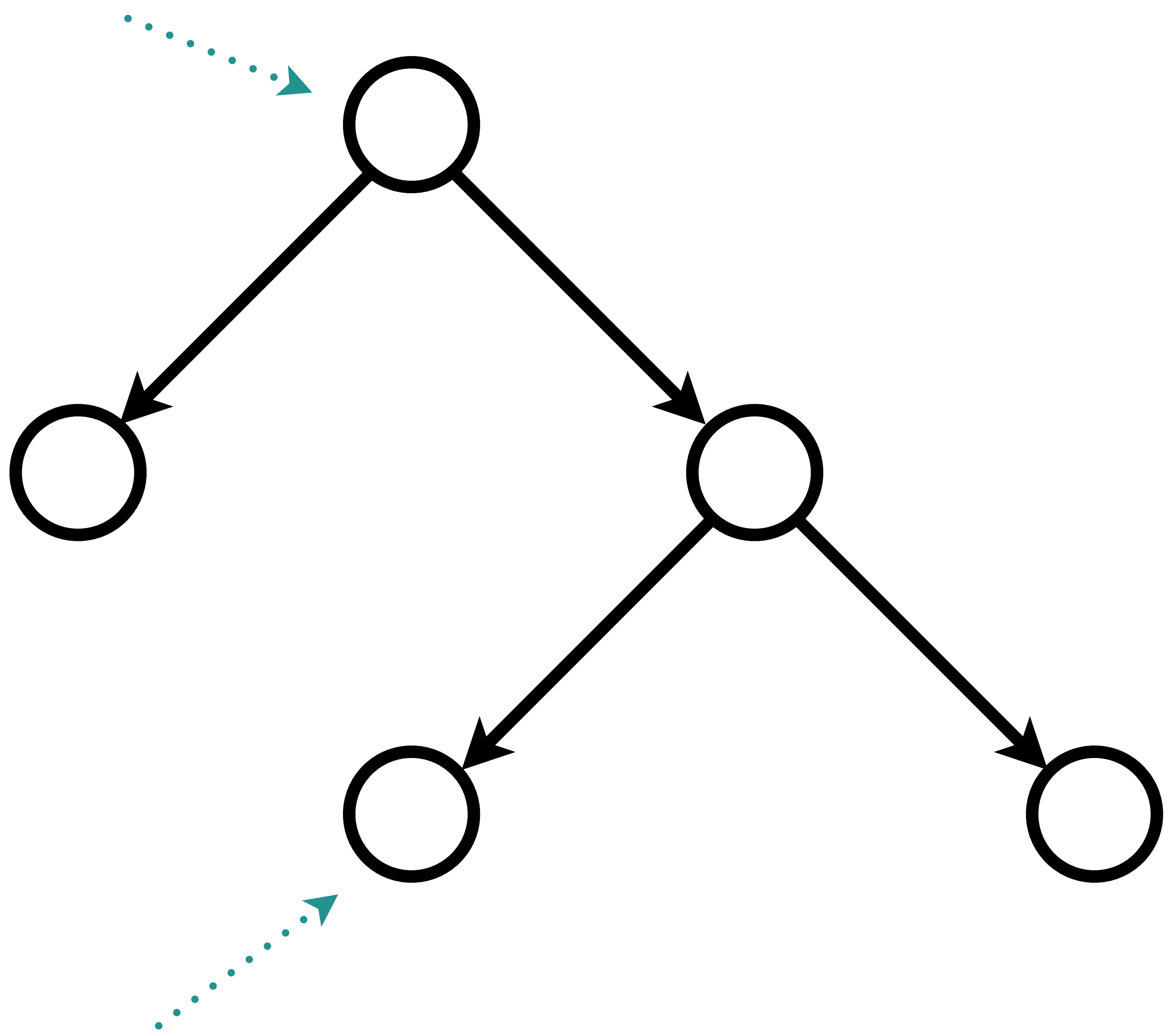


Review: Trees

A tree is a data structure with the following properties:

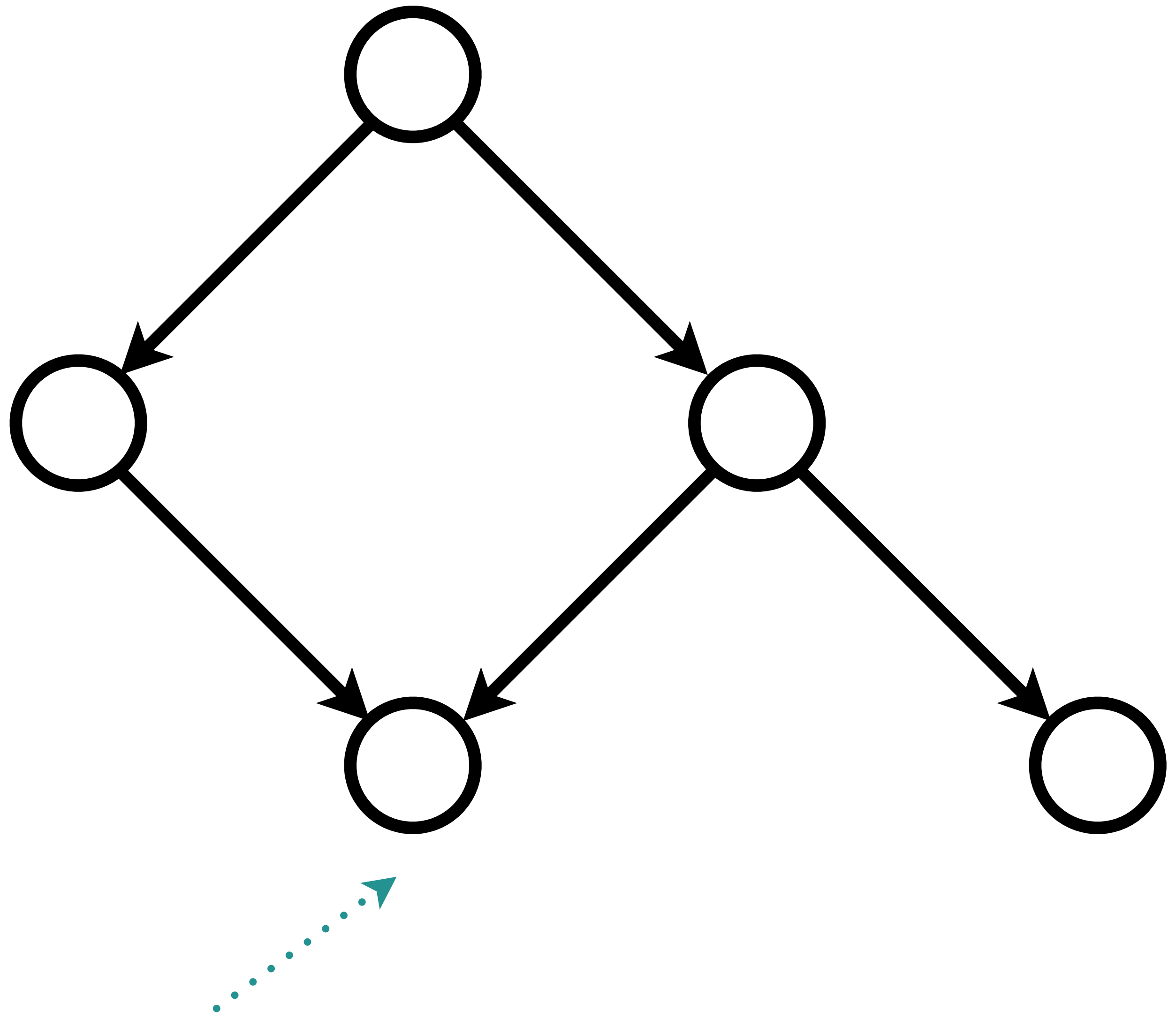
- One distinguished node is designated as the root
- Every other node (except root node) has ***exactly*** one parent

Root



Every node has exactly one parent

✓ A Tree!



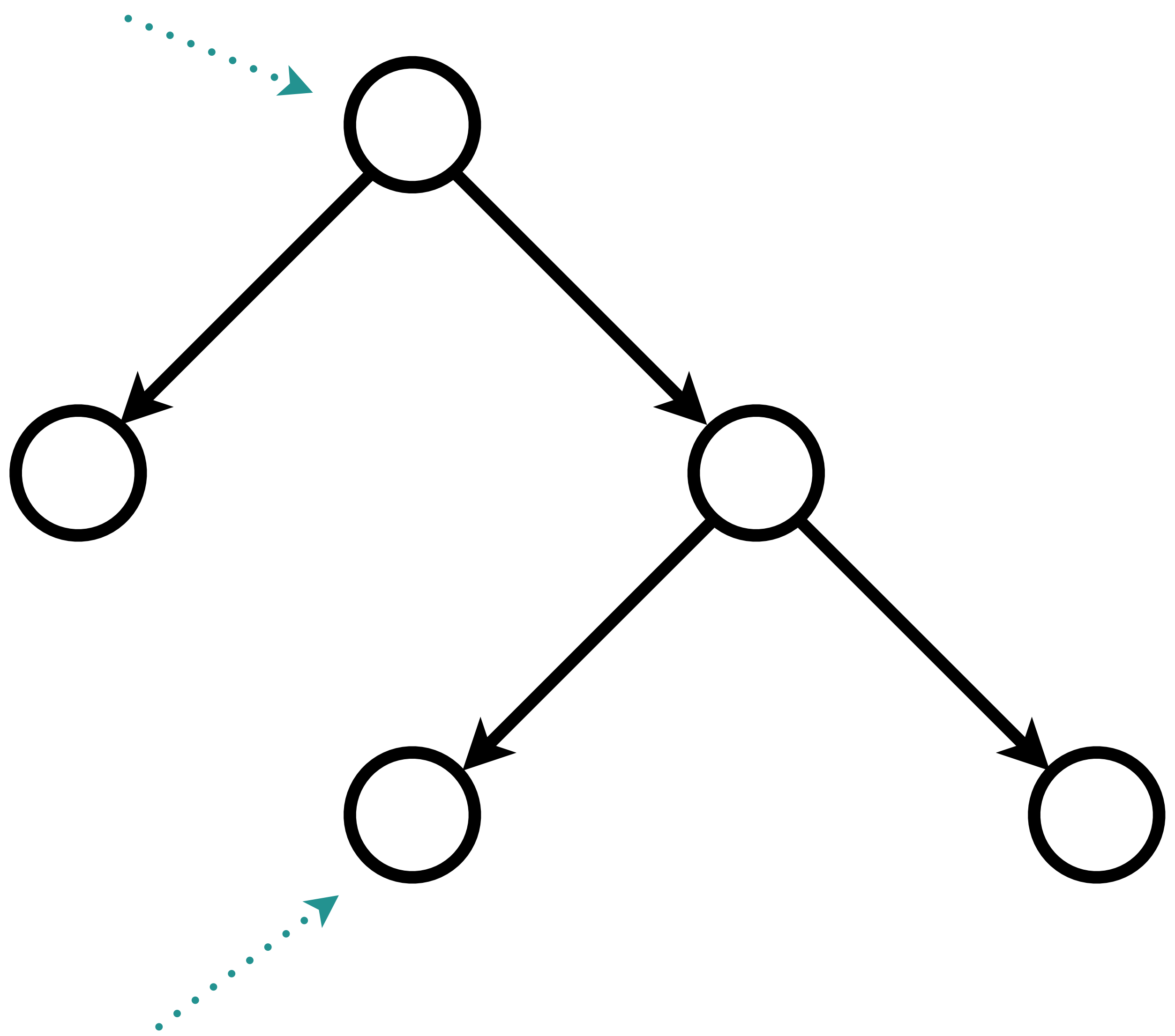
This node has multiple parents

Review: Trees

A tree is a data structure with the following properties:

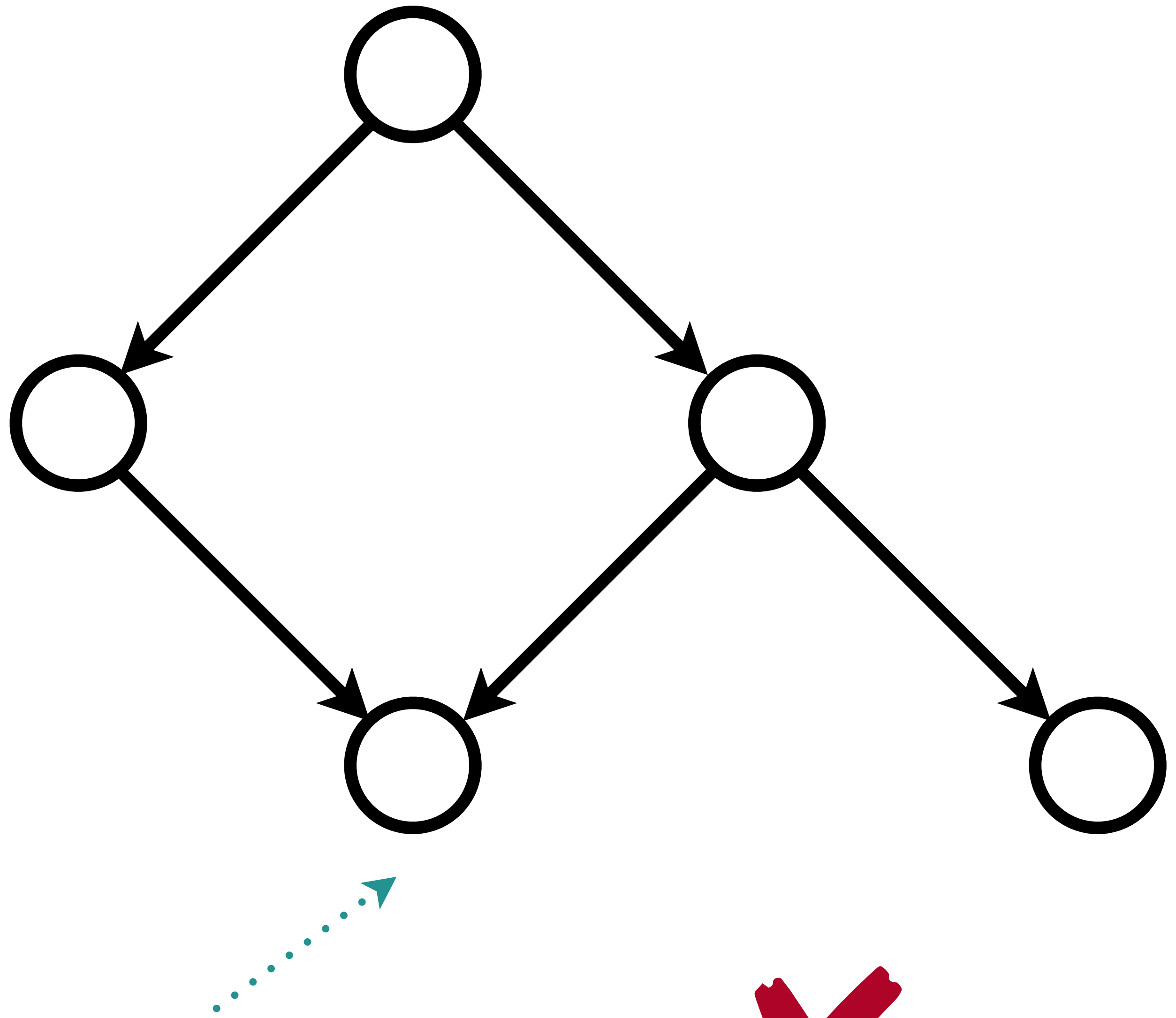
- One distinguished node is designated as the root
- Every other node (except root node) has ***exactly*** one parent

Root



Every node has exactly one parent

✓ A Tree!



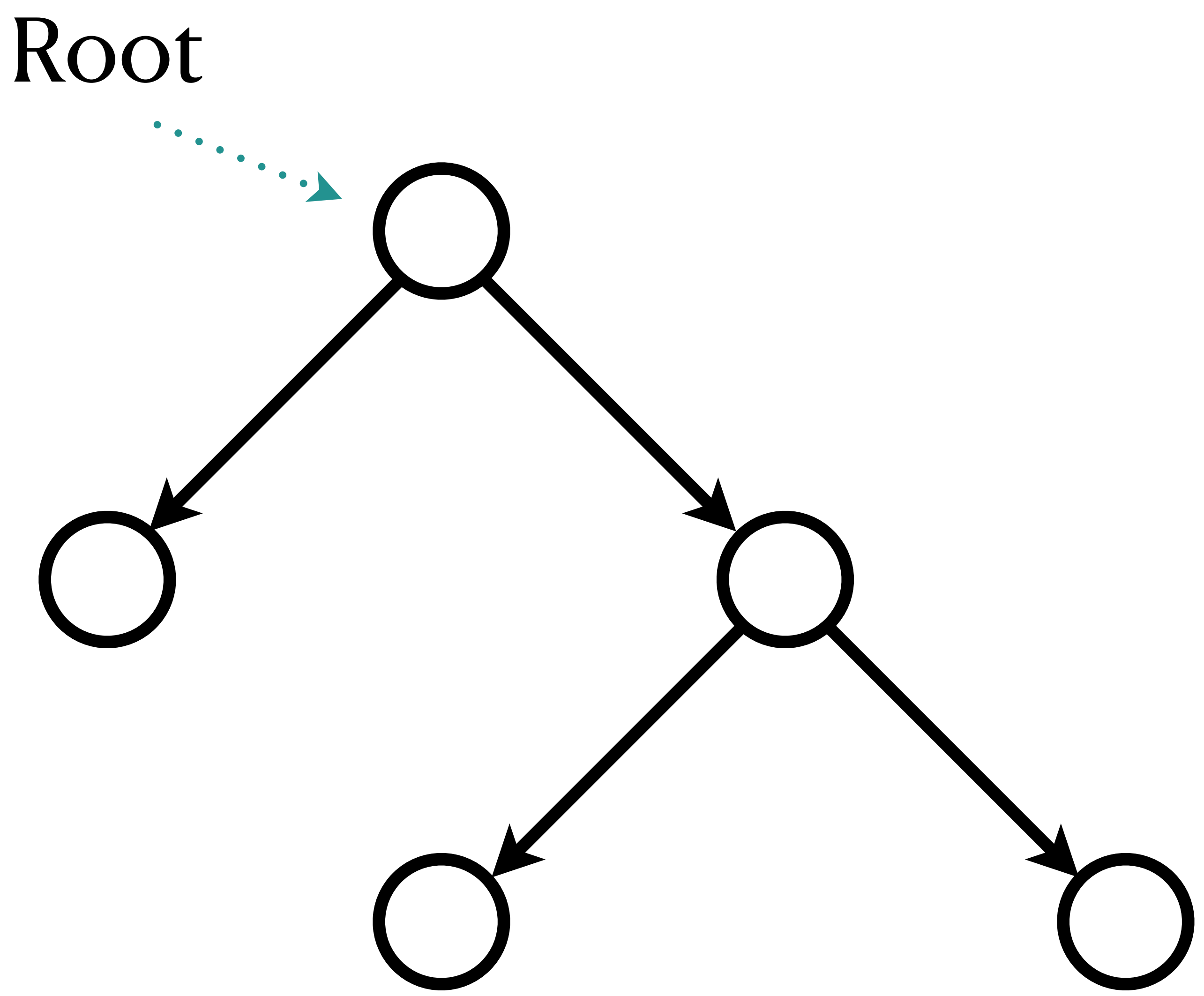
This node has multiple parents ✗ Not a Tree!

Review: Trees

- Two nodes are called ***siblings*** if they have same parent.
- A node is called ***leaf node*** if it has no children.

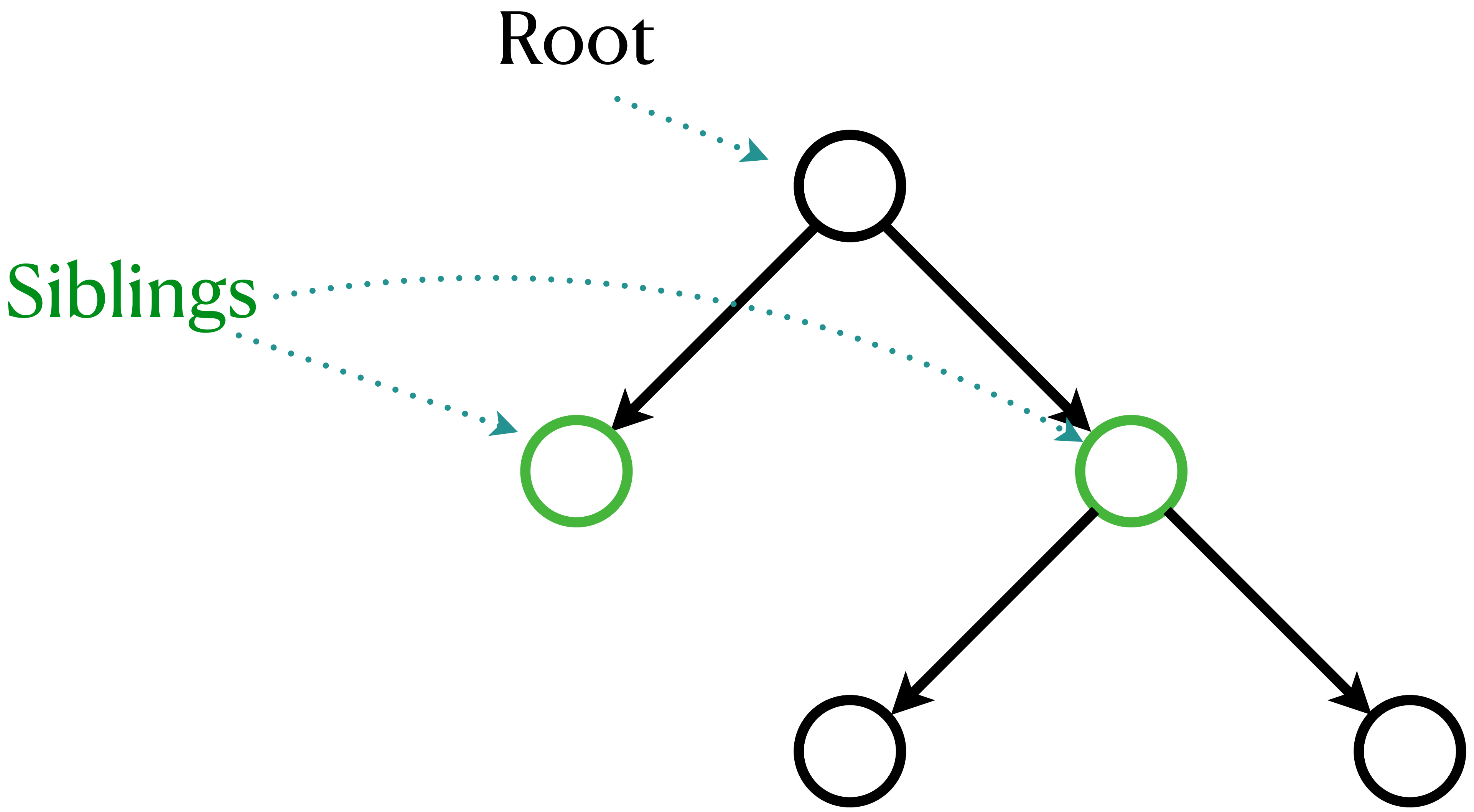
Review: Trees

- Two nodes are called ***siblings*** if they have same parent.
- A node is called ***leaf node*** if it has no children.



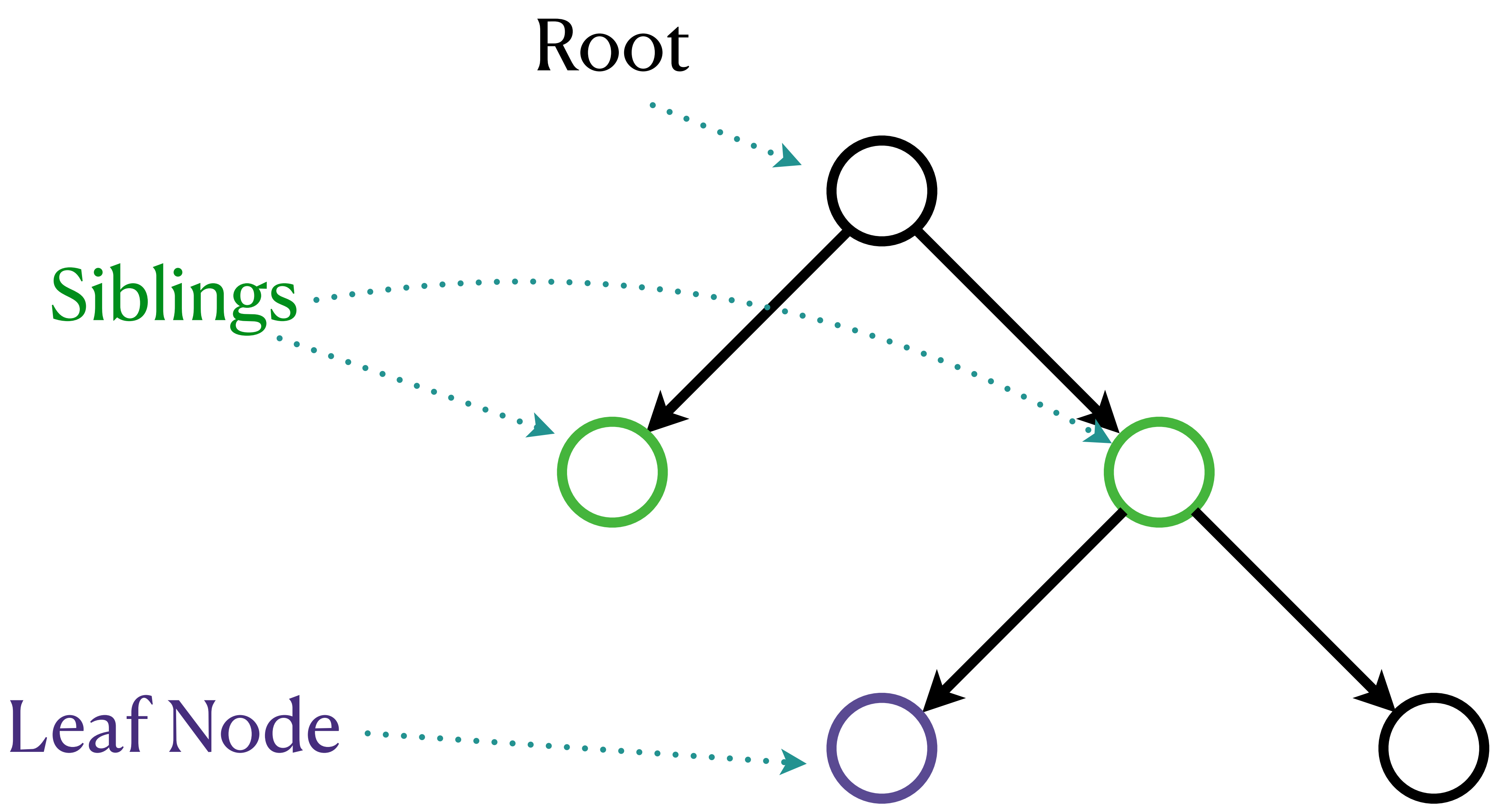
Review: Trees

- Two nodes are called ***siblings*** if they have same parent.
- A node is called ***leaf node*** if it has no children.



Review: Trees

- Two nodes are called ***siblings*** if they have same parent.
- A node is called ***leaf node*** if it has no children.

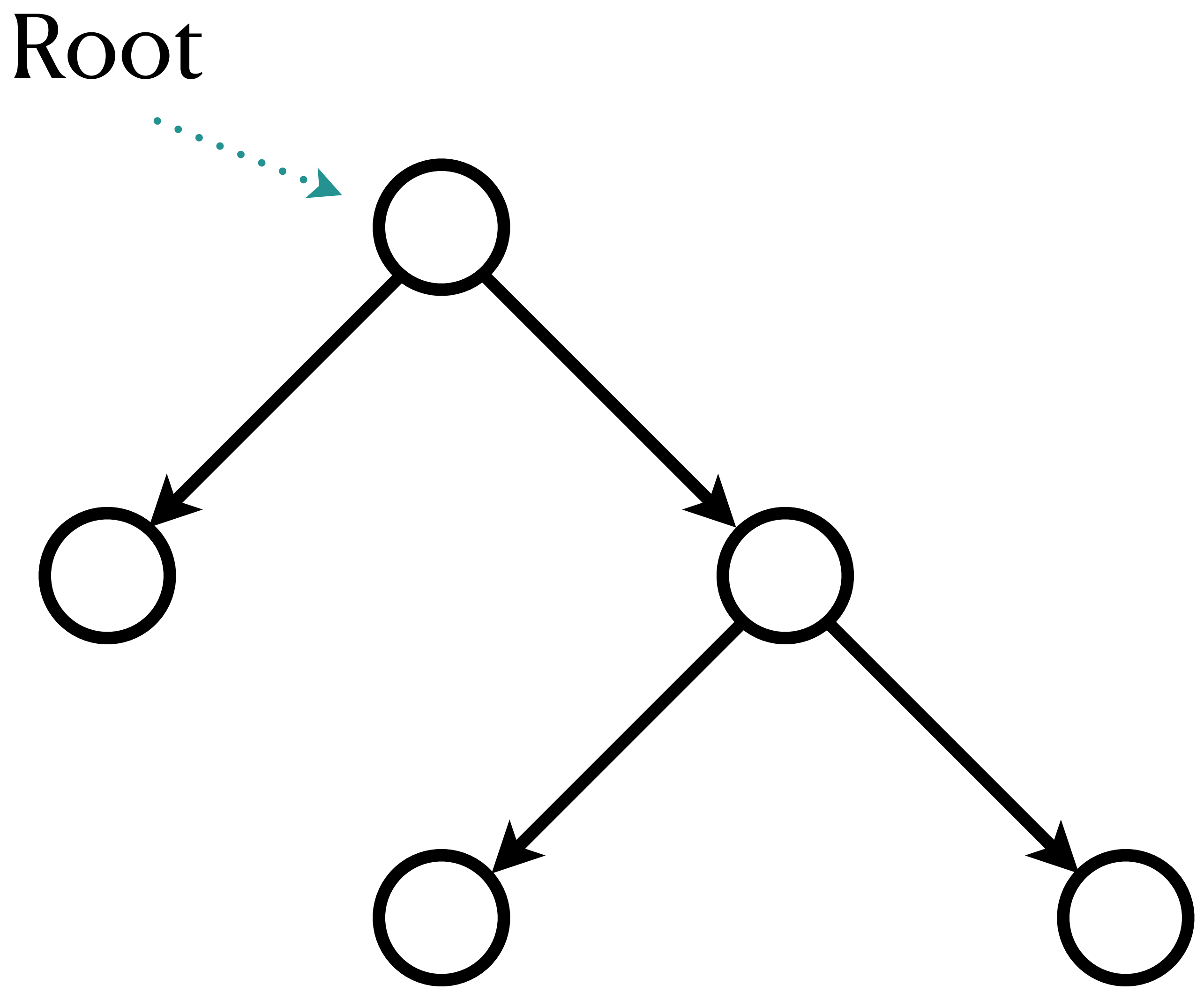


Review: Trees

Depth of a node is the number of nodes from root to that node
Height of a tree is depth of its deepest node.

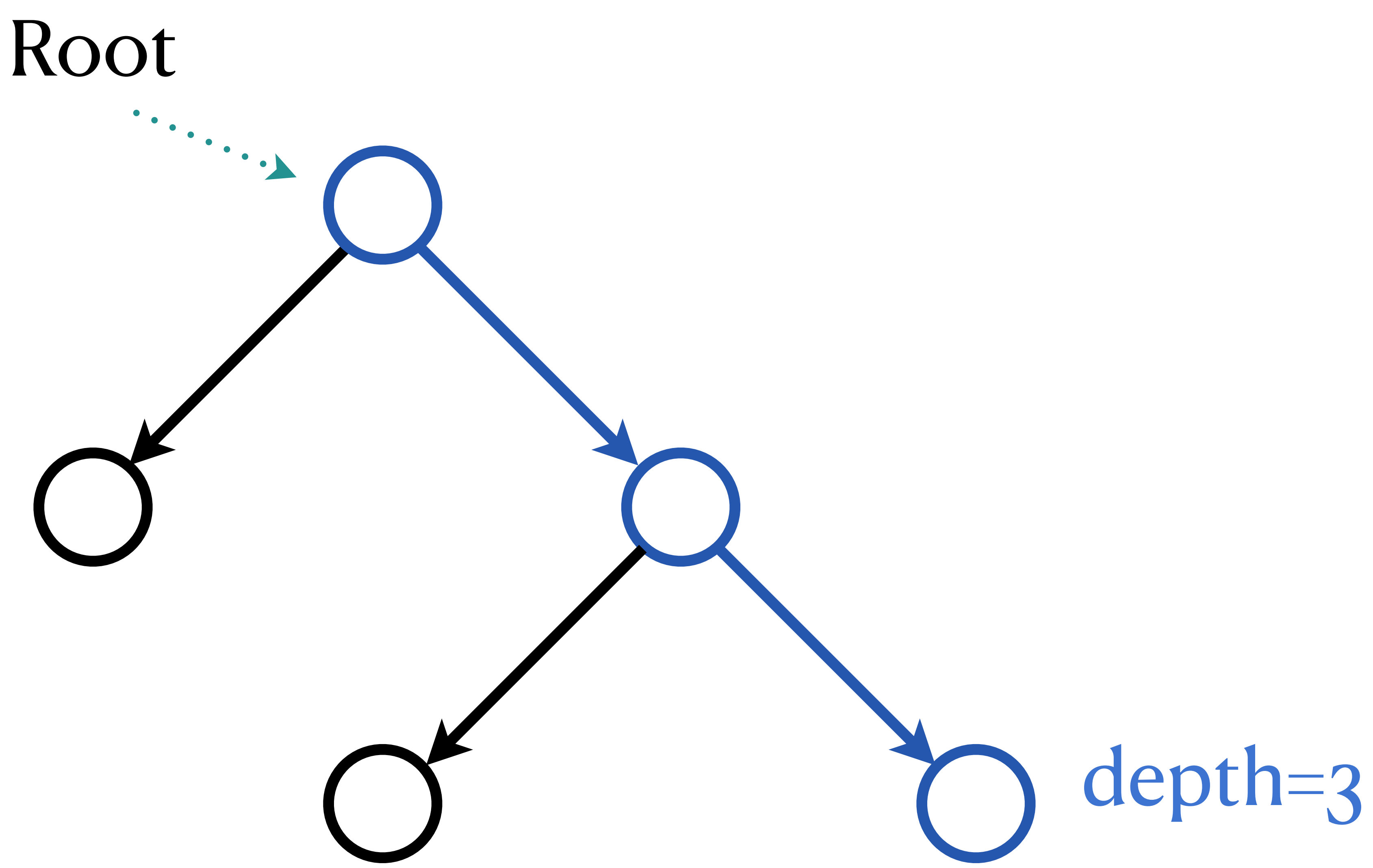
Review: Trees

Depth of a node is the number of nodes from root to that node
Height of a tree is depth of its deepest node.



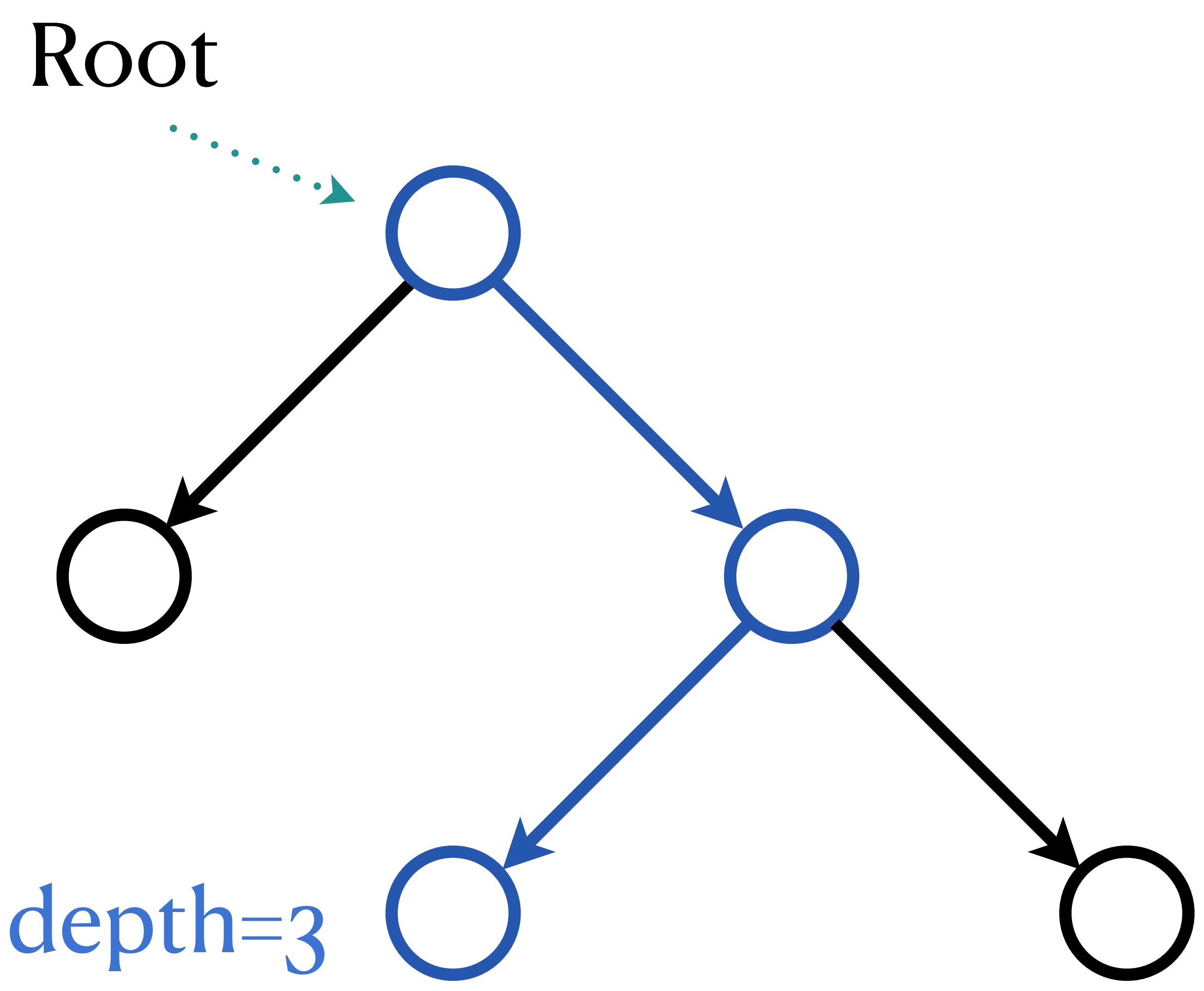
Review: Trees

Depth of a node is the number of nodes from root to that node
Height of a tree is depth of its deepest node.



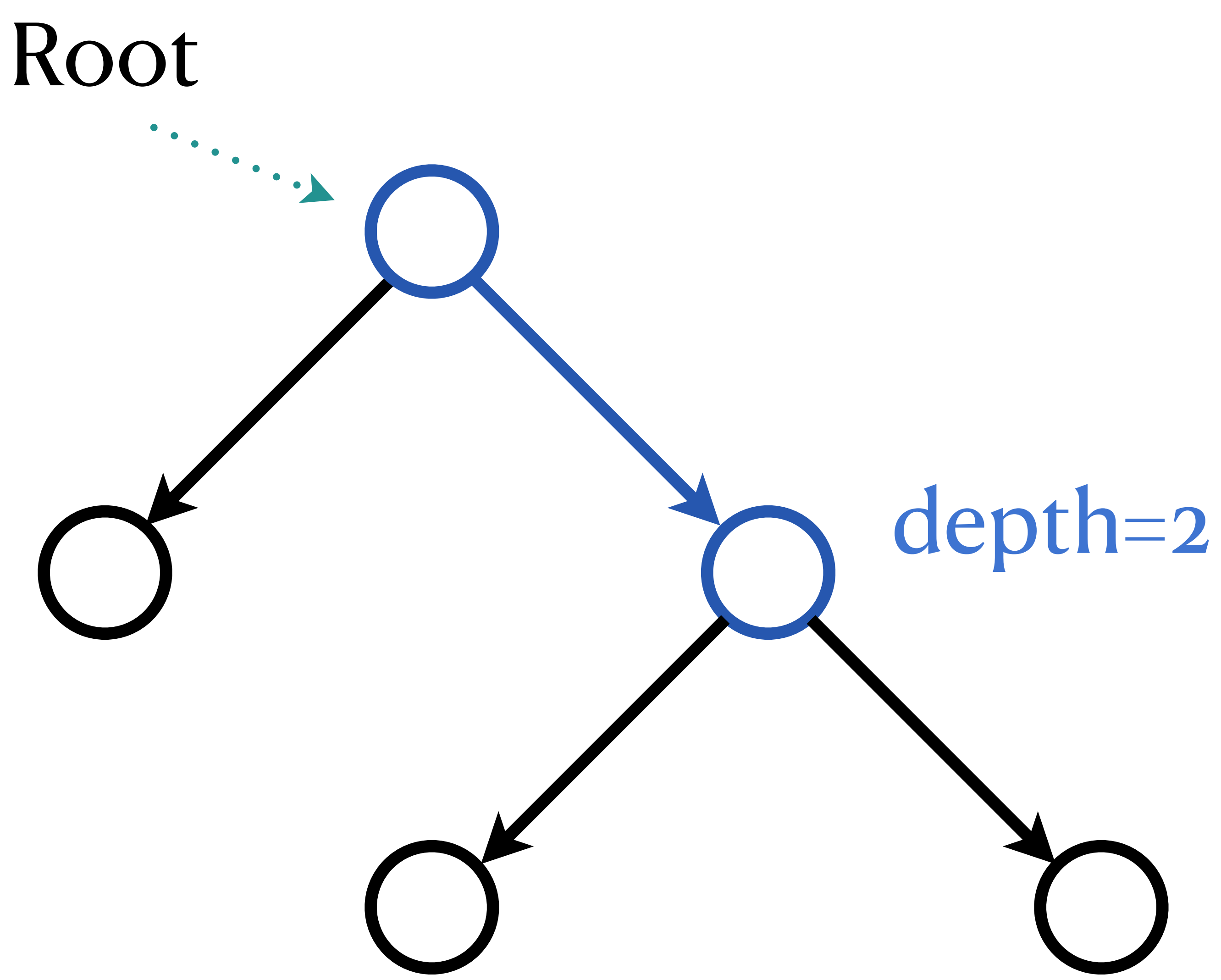
Review: Trees

Depth of a node is the number of nodes from root to that node
Height of a tree is depth of its deepest node.



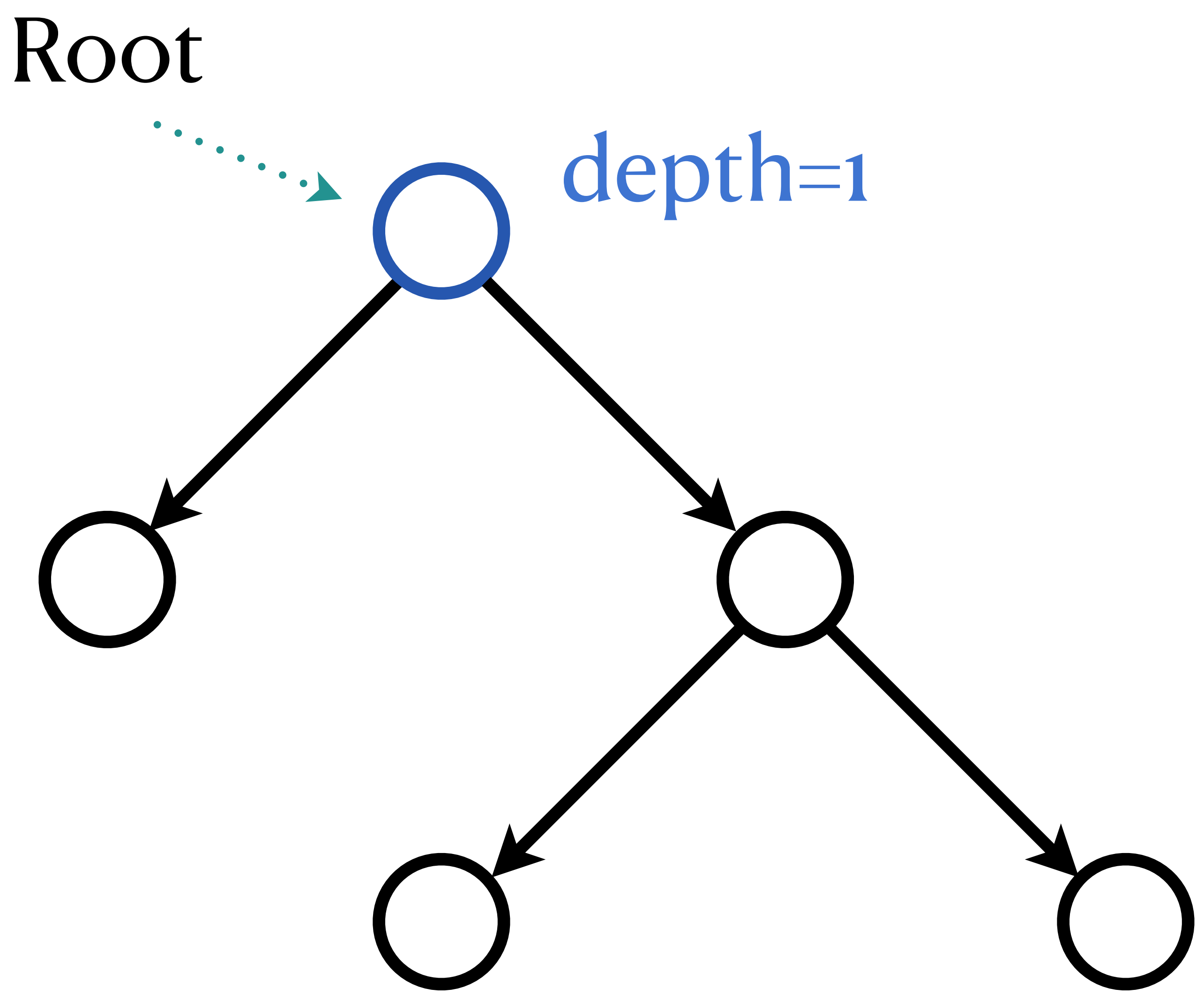
Review: Trees

Depth of a node is the number of nodes from root to that node
Height of a tree is depth of its deepest node.



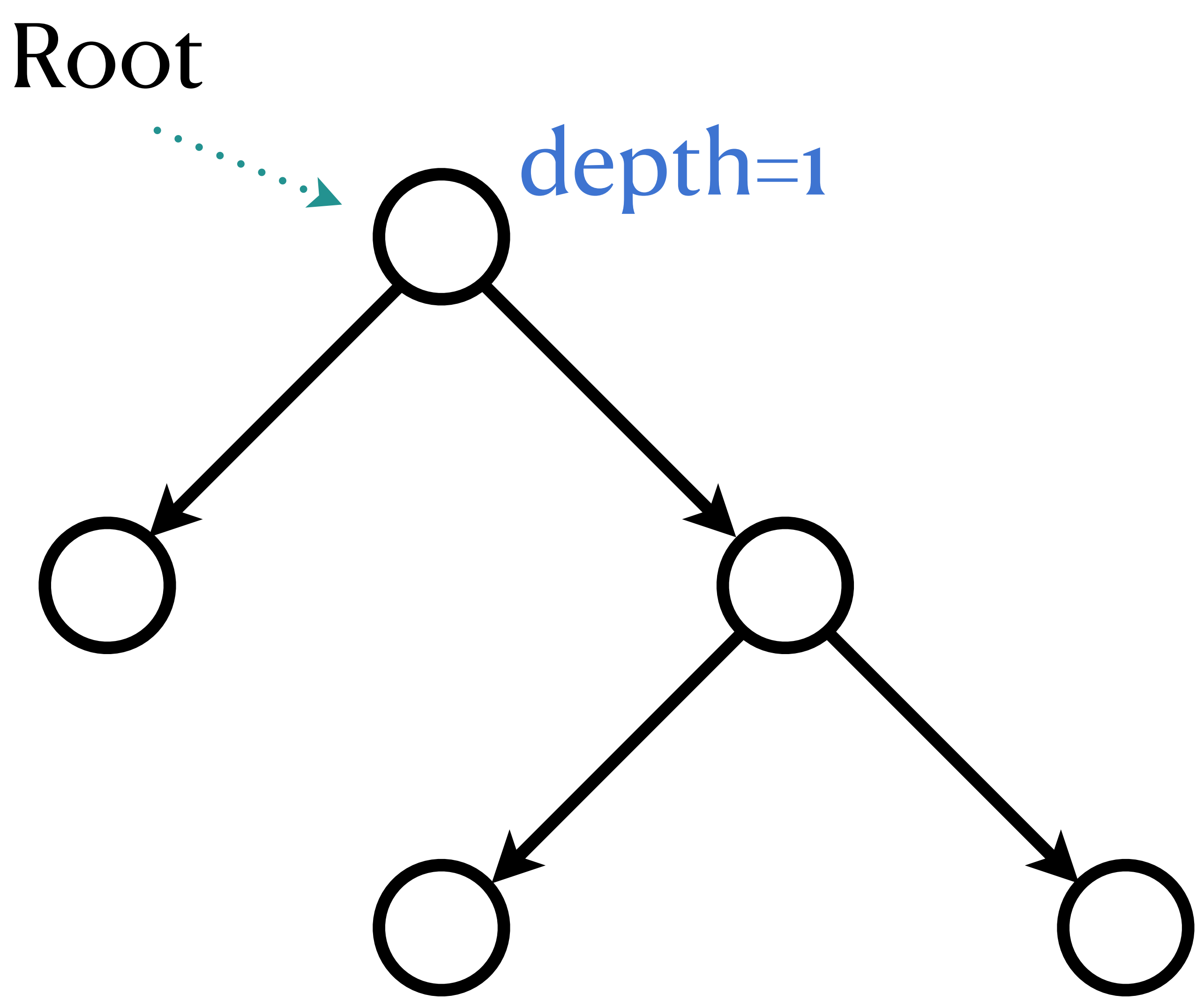
Review: Trees

Depth of a node is the number of nodes from root to that node
Height of a tree is depth of its deepest node.



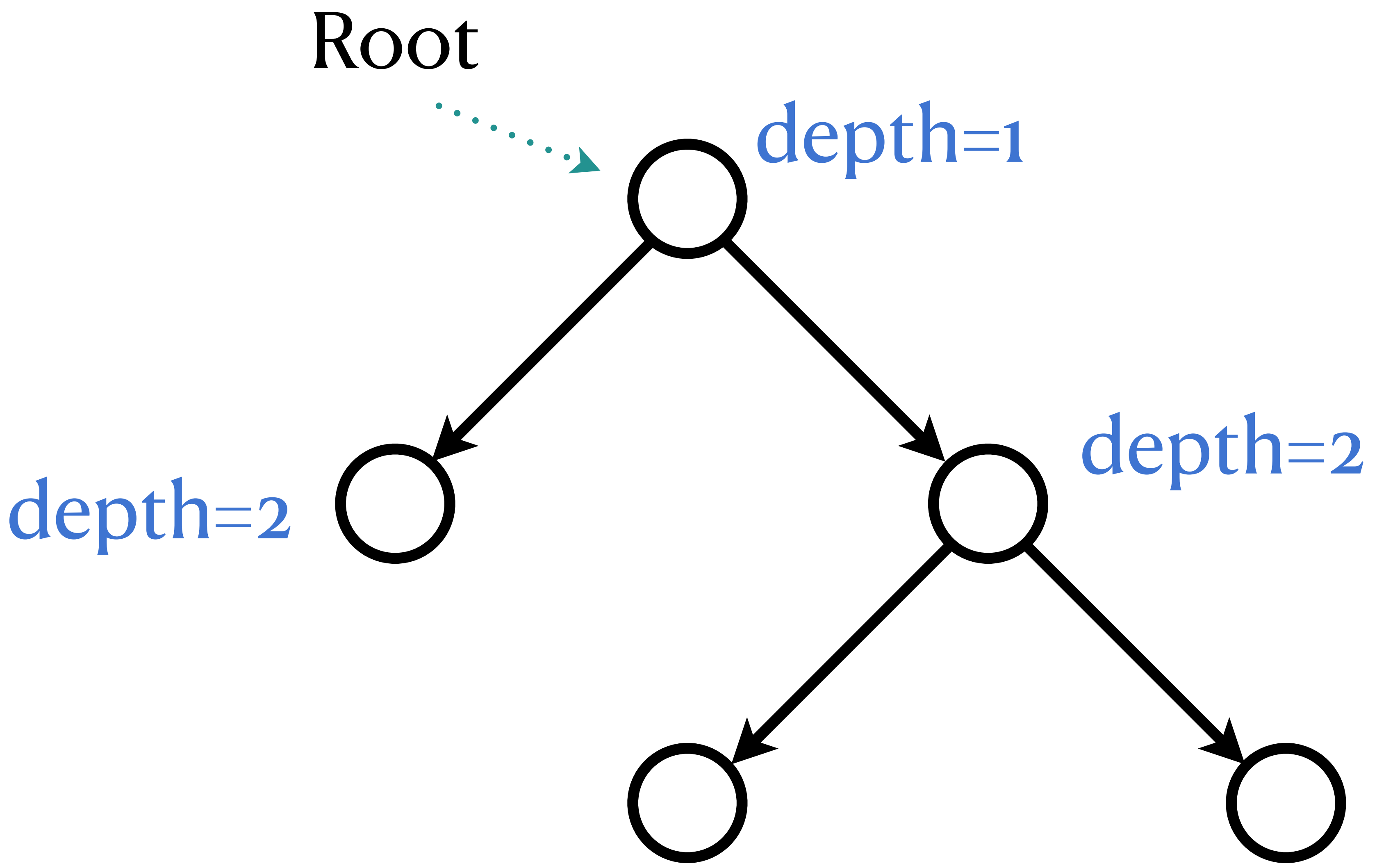
Review: Trees

Depth of a node is the number of nodes from root to that node
Height of a tree is depth of its deepest node.



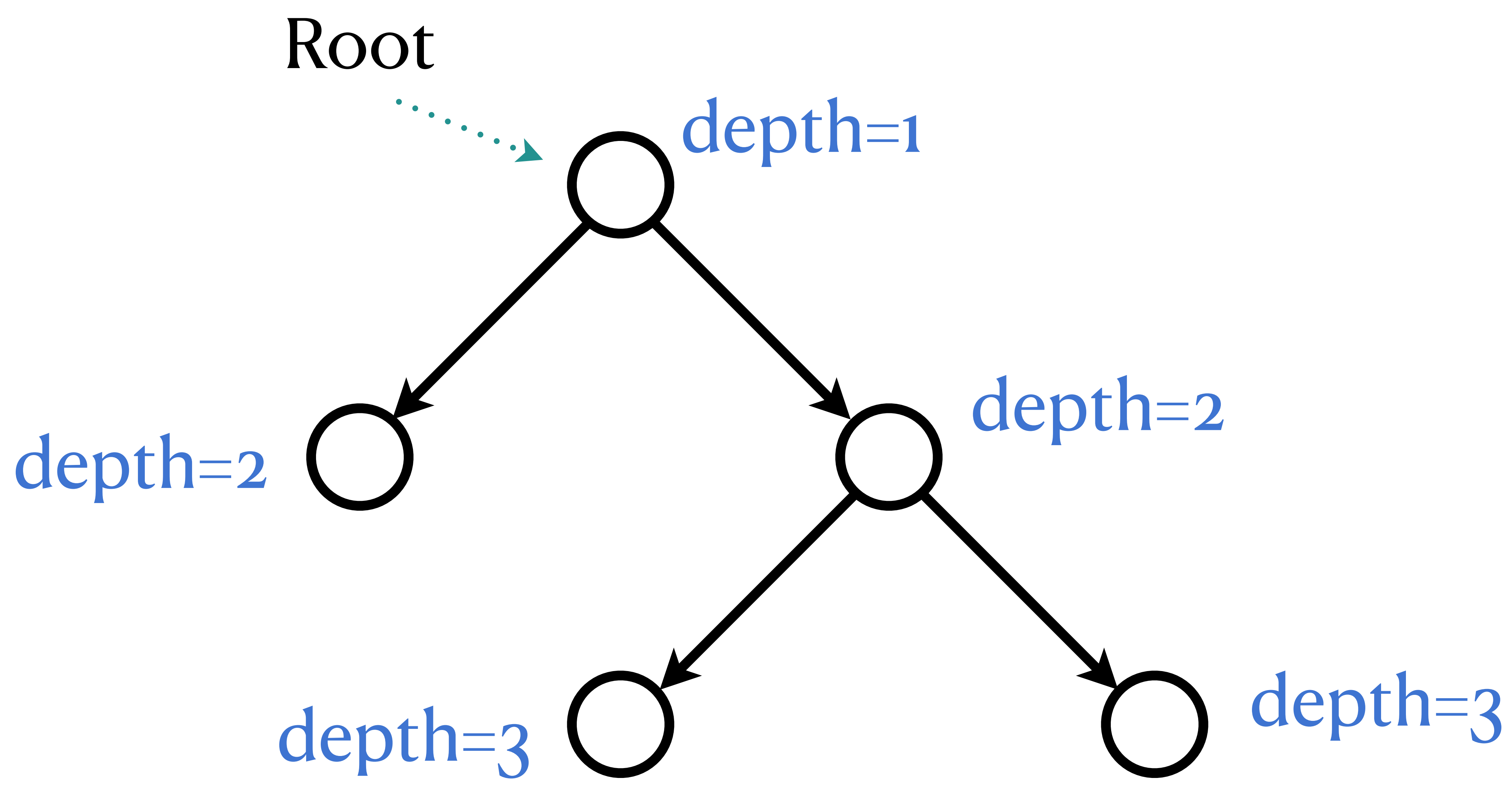
Review: Trees

Depth of a node is the number of nodes from root to that node
Height of a tree is depth of its deepest node.



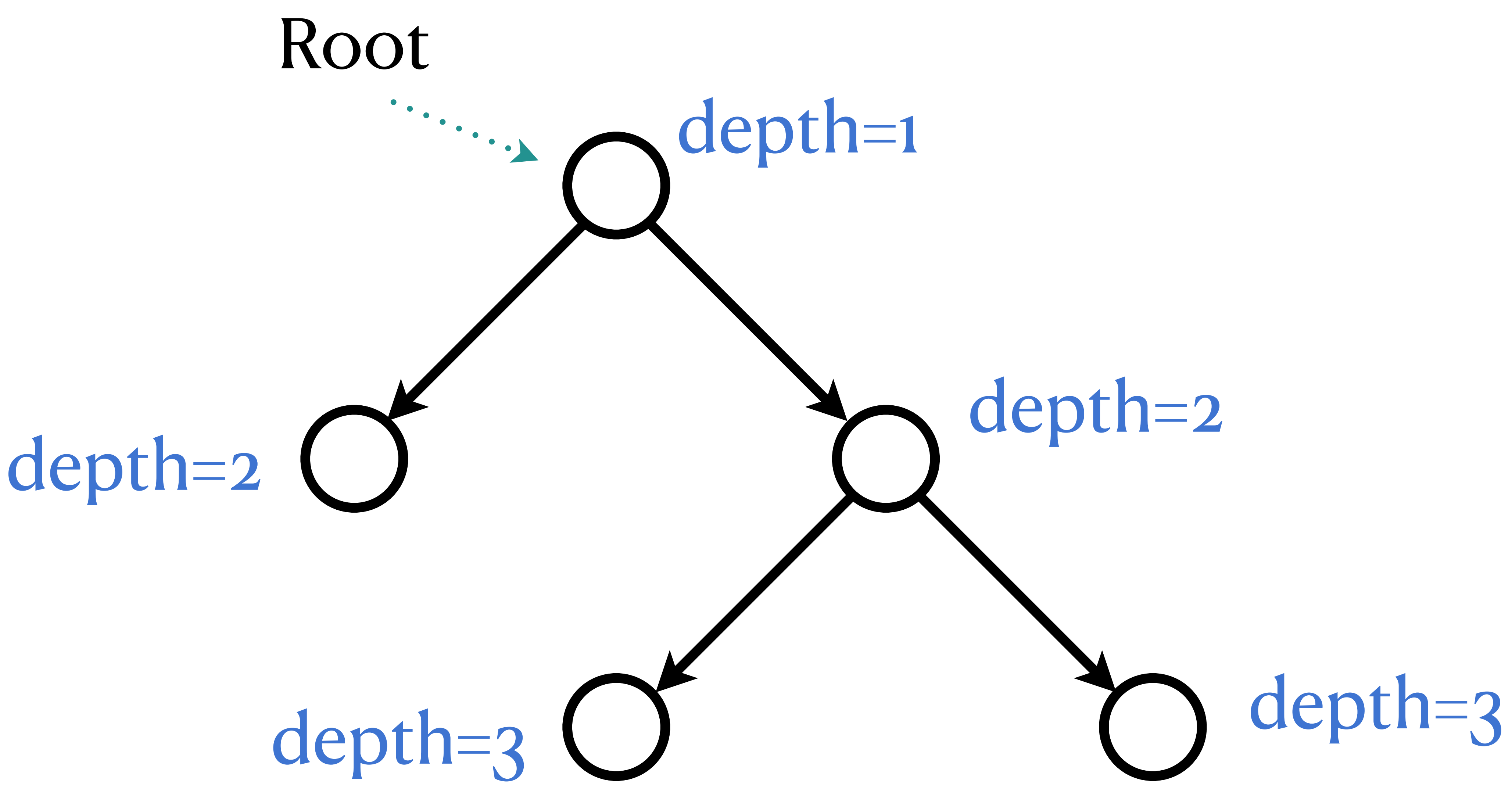
Review: Trees

Depth of a node is the number of nodes from root to that node
Height of a tree is depth of its deepest node.



Review: Trees

Depth of a node is the number of nodes from root to that node
Height of a tree is depth of its deepest node.



Height of the tree is 3

Review: Trees

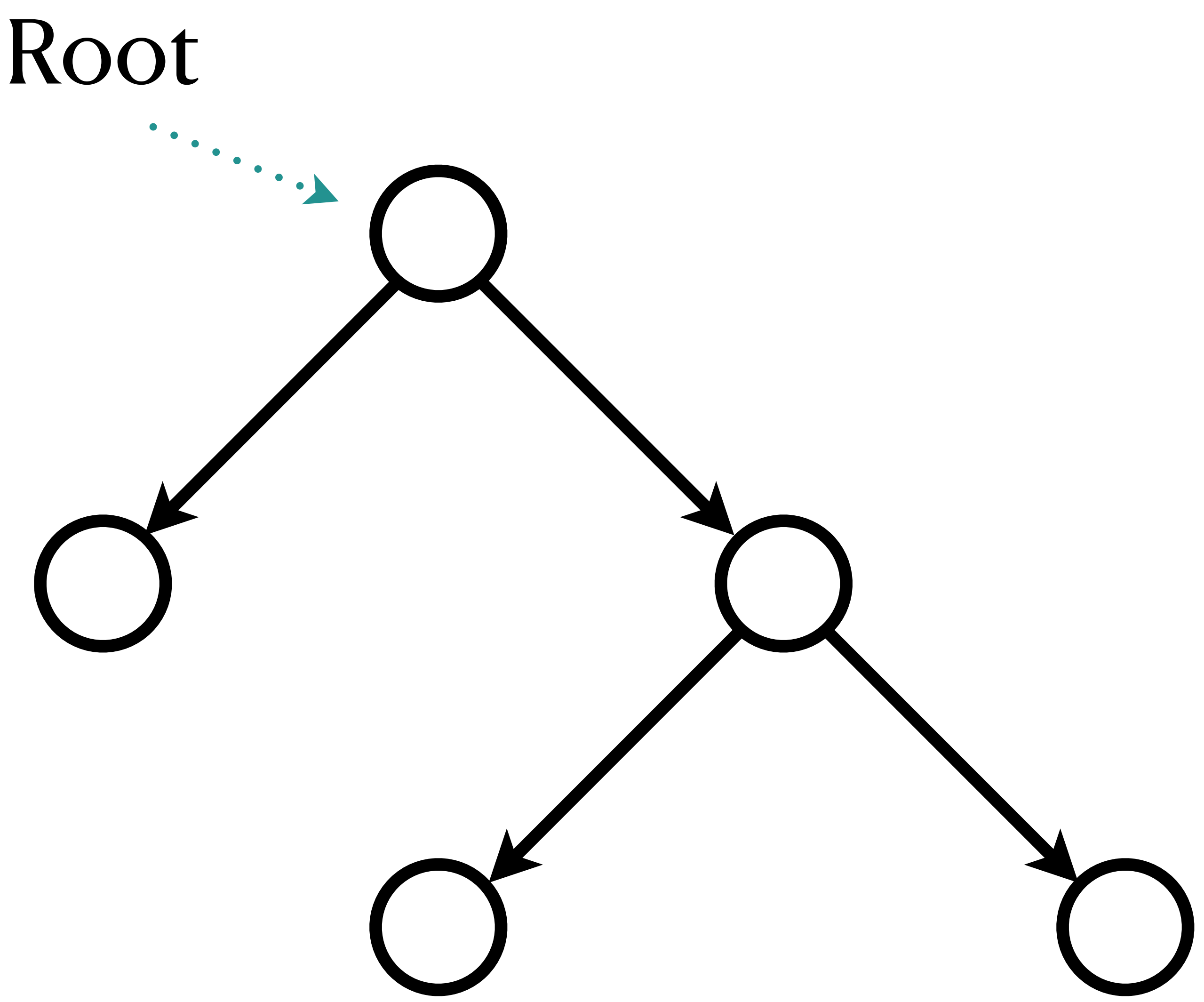
A Binary tree is a tree such that:

- Every node has ***at most*** two children

Review: Trees

A Binary tree is a tree such that:

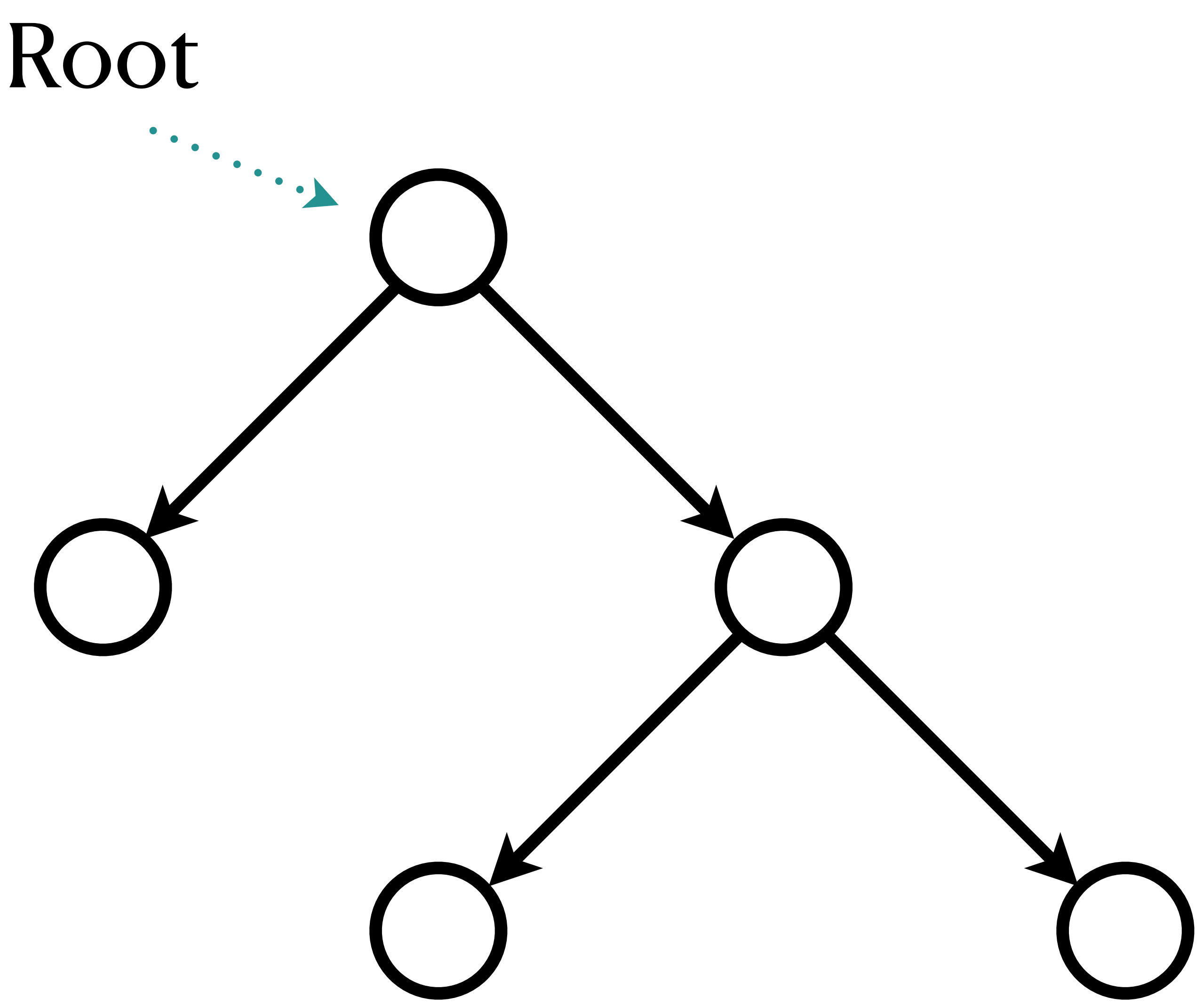
- Every node has ***at most*** two children



Review: Trees

A Binary tree is a tree such that:

- Every node has ***<u>at most</u>*** two children

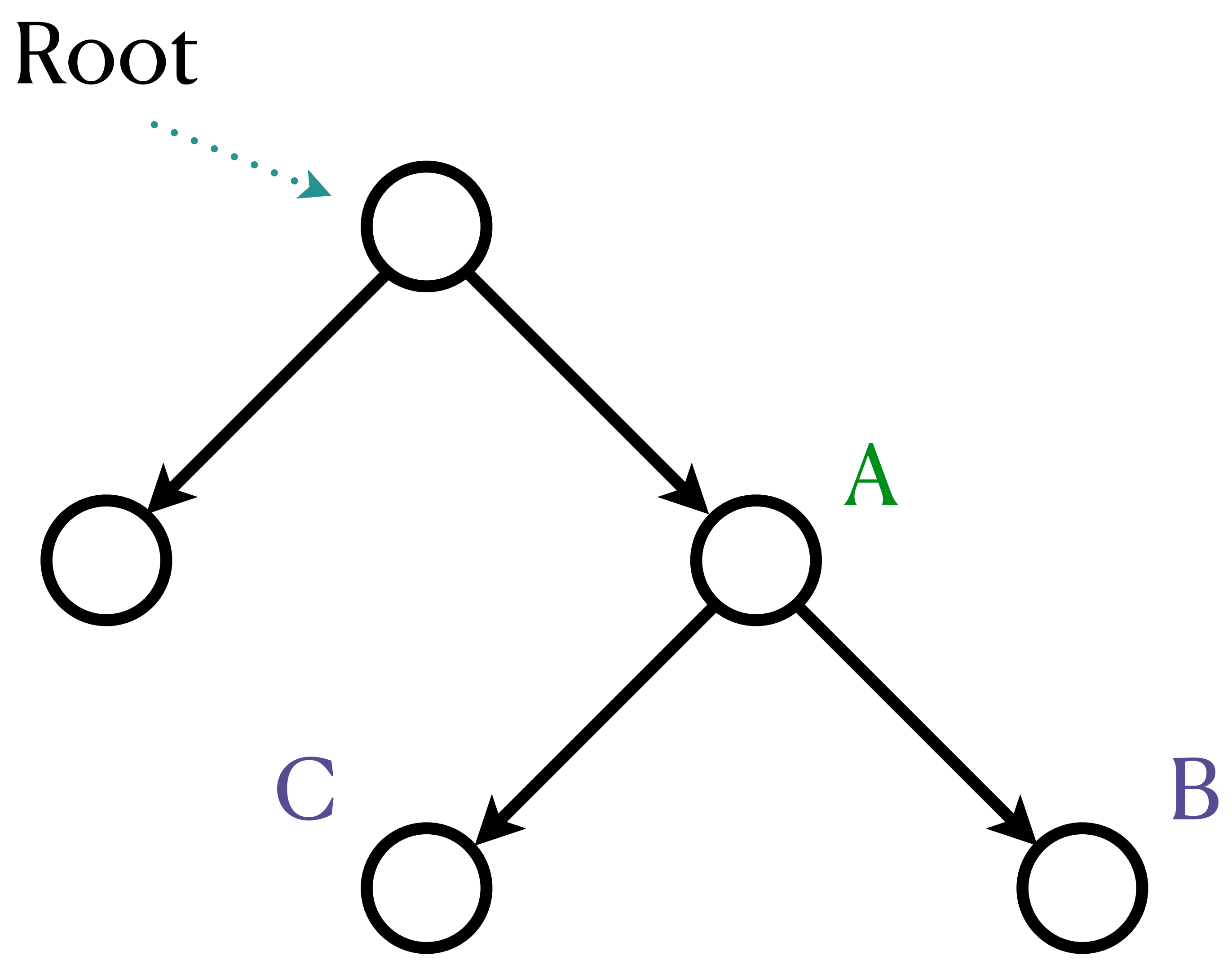


Every node has 0,1 or 2 children

Review: Trees

A Binary tree is a tree such that:

- Every node has ***at most*** two children



A is parent of B and C

B is right child of A

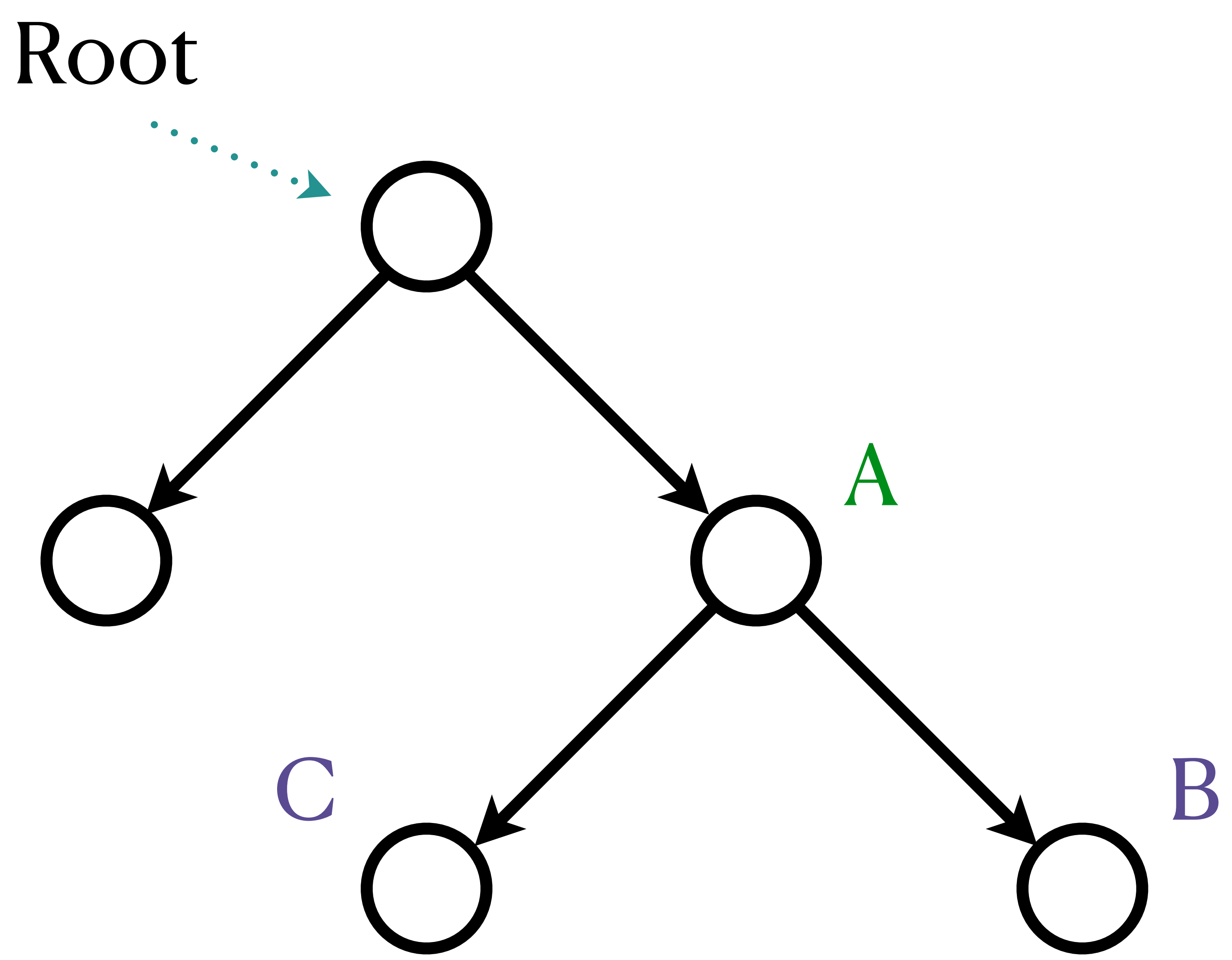
C is left child of A

Every node has 0,1 or 2 children

Review: Trees

A Binary tree is a tree such that:

- Every node has ***at most*** two children



A is parent of B and C

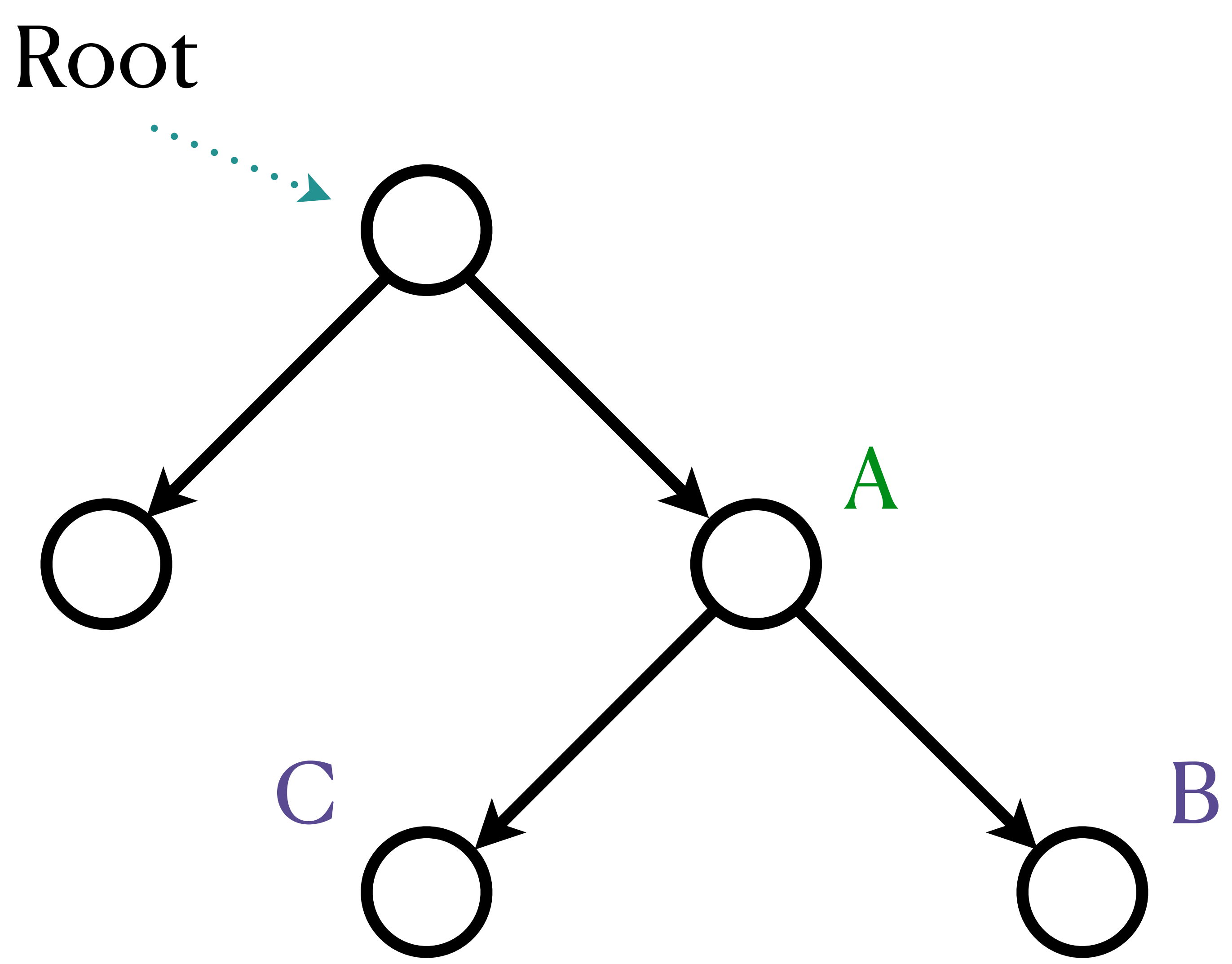
B is right child of A

Every node has 0,1 or 2 children

Review: Trees

A Binary tree is a tree such that:

- Every node has ***<u>at most</u>*** two children



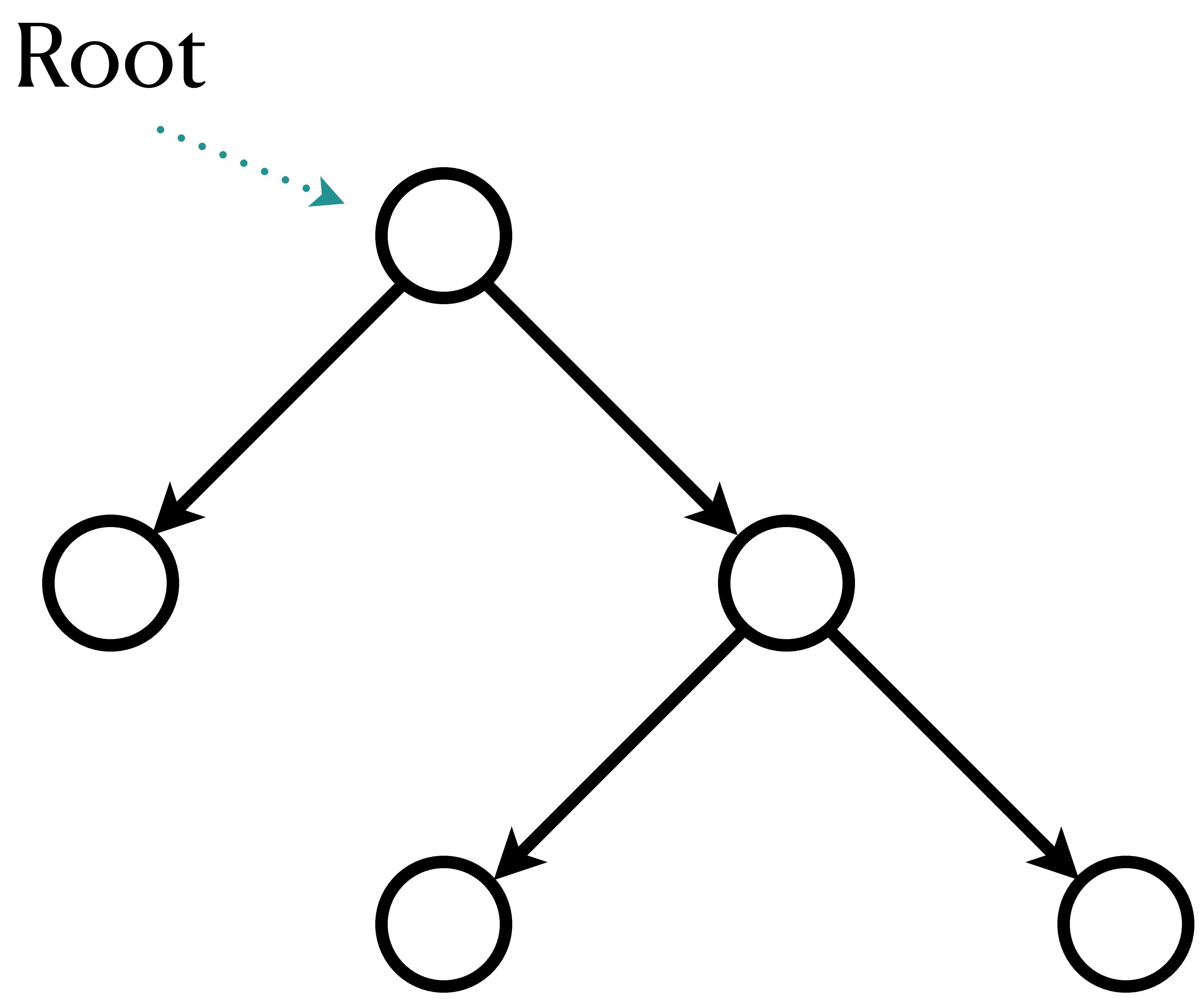
A is parent of B and C

Every node has 0,1 or 2 children

Review: Trees

A Binary tree is a tree such that:

- Every node has ***at most*** two children

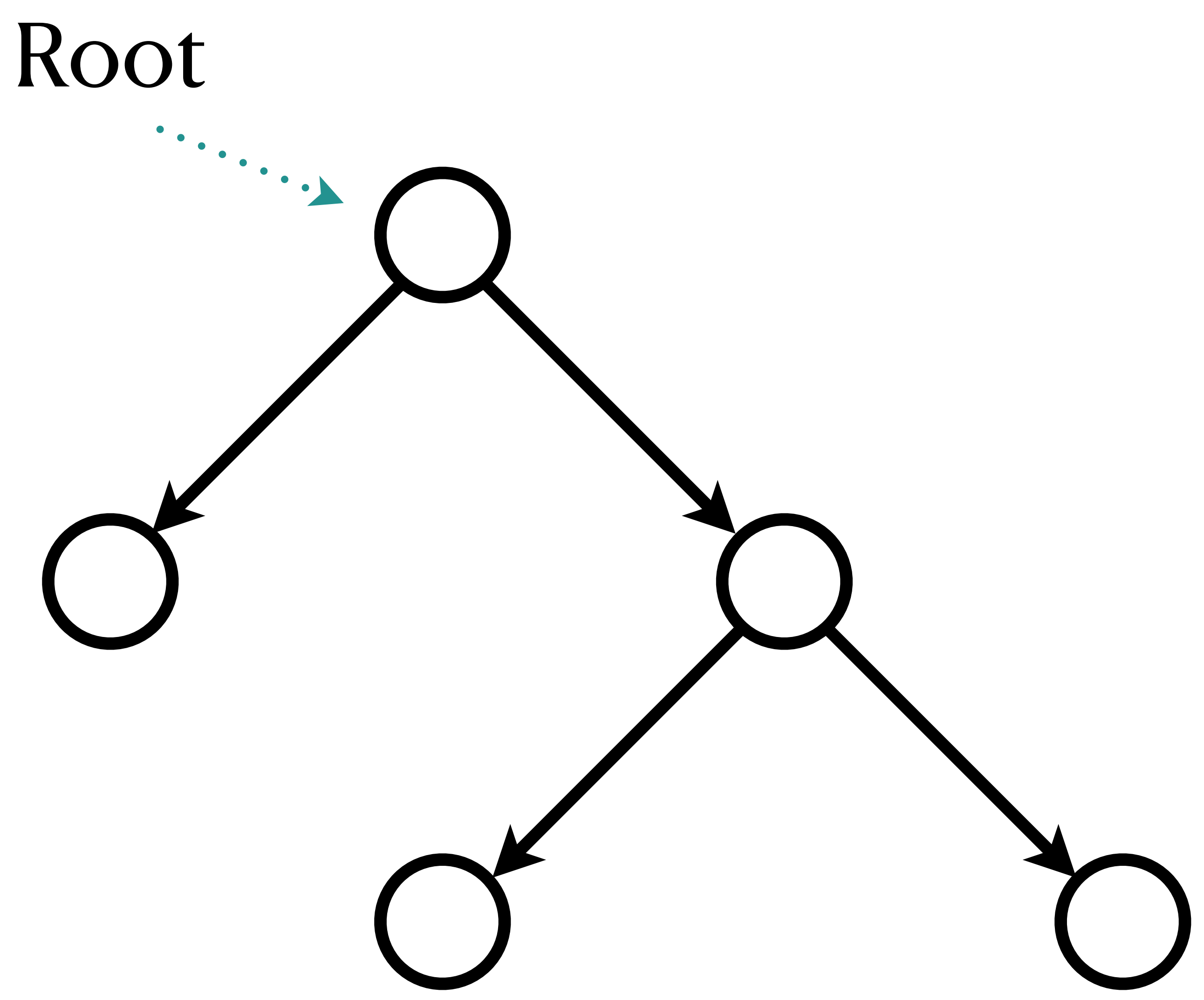


Every node has 0,1 or 2 children ✓ A binary tree!

Review: Trees

A Binary tree is a tree such that:

- Every node has ***at most*** two children

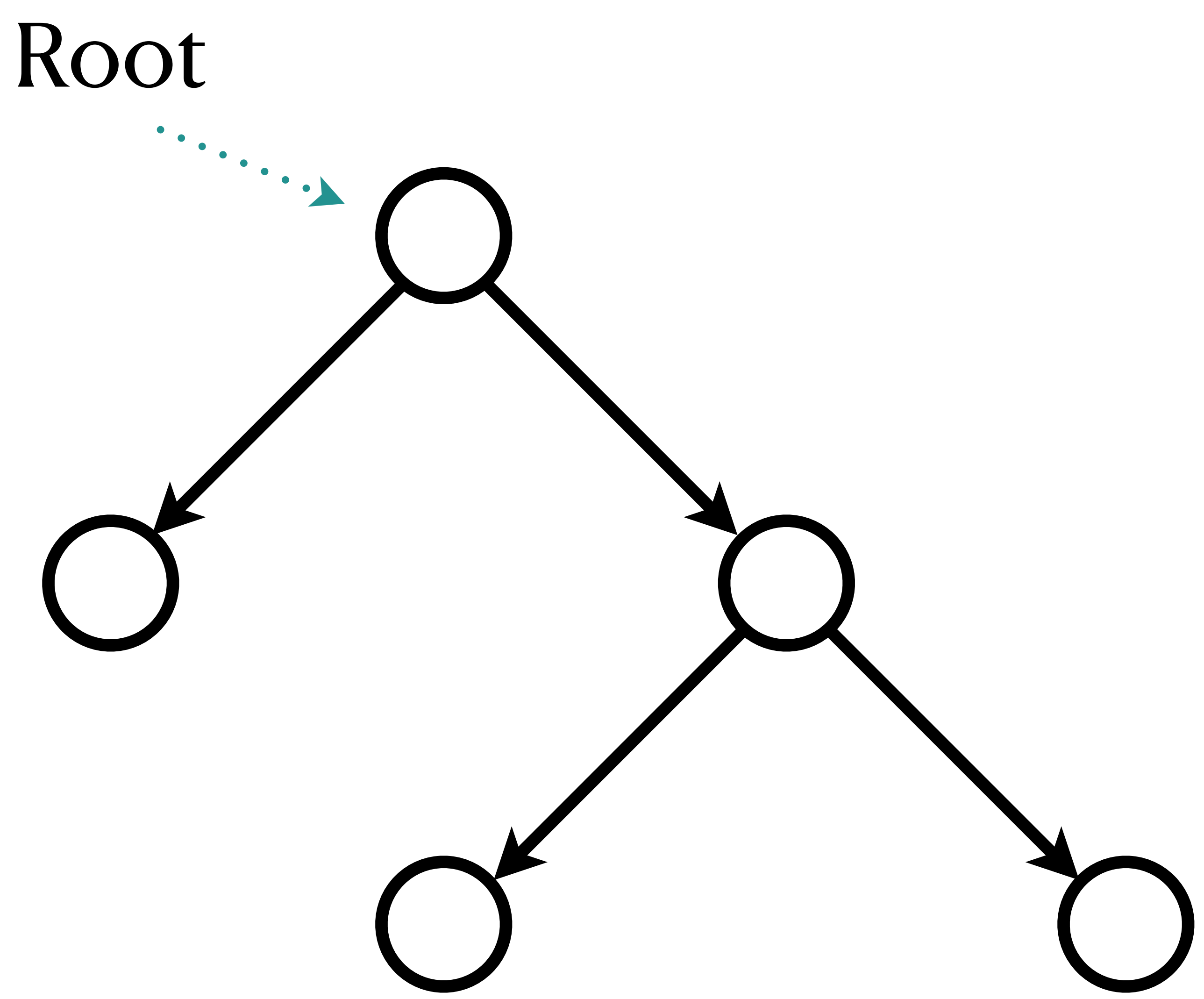


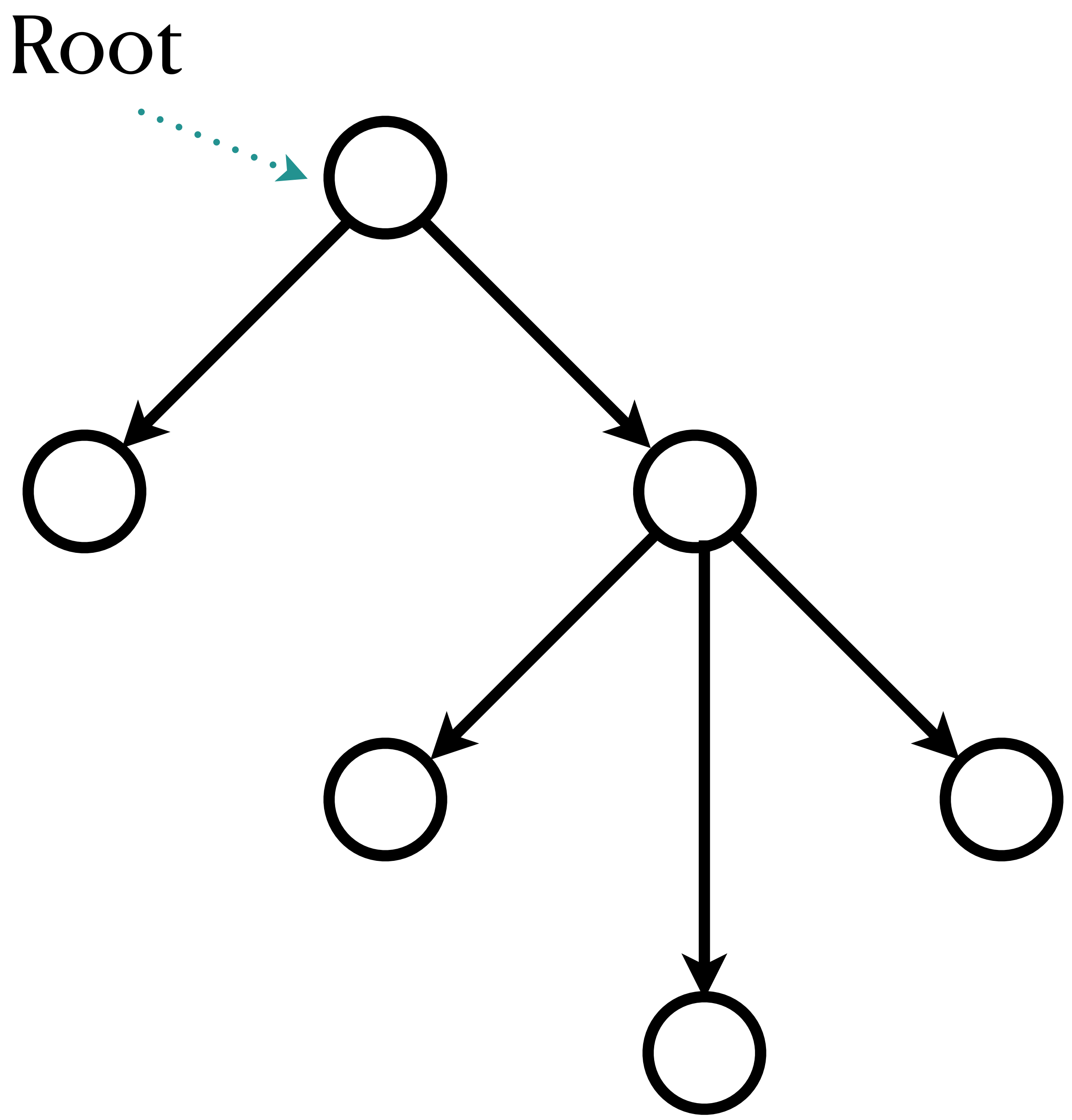
Every node has 0,1 or 2 children ✓ A binary tree!

Review: Trees

A Binary tree is a tree such that:

- Every node has ***at most*** two children



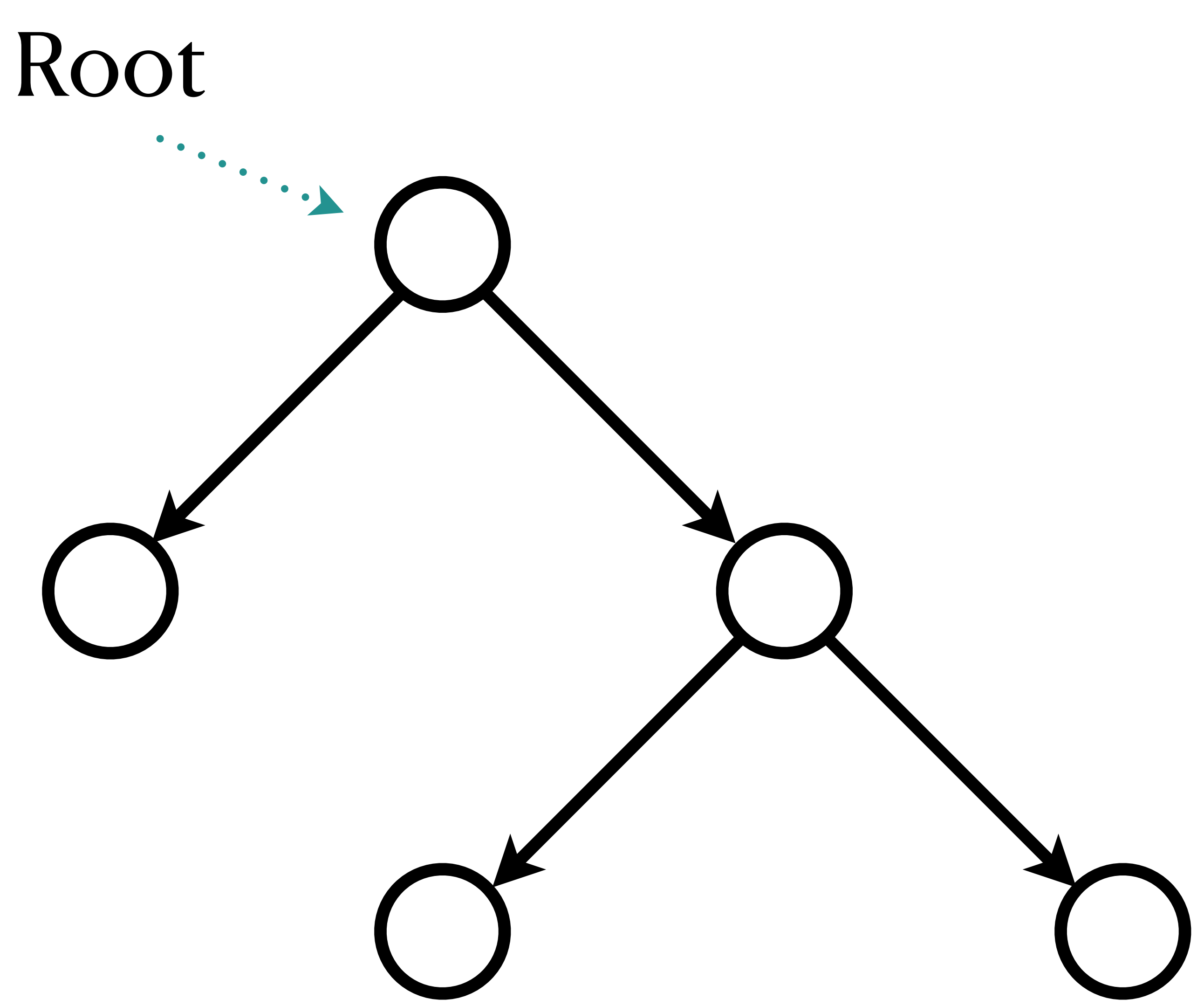


Every node has 0,1 or 2 children ✓ A binary tree!

Review: Trees

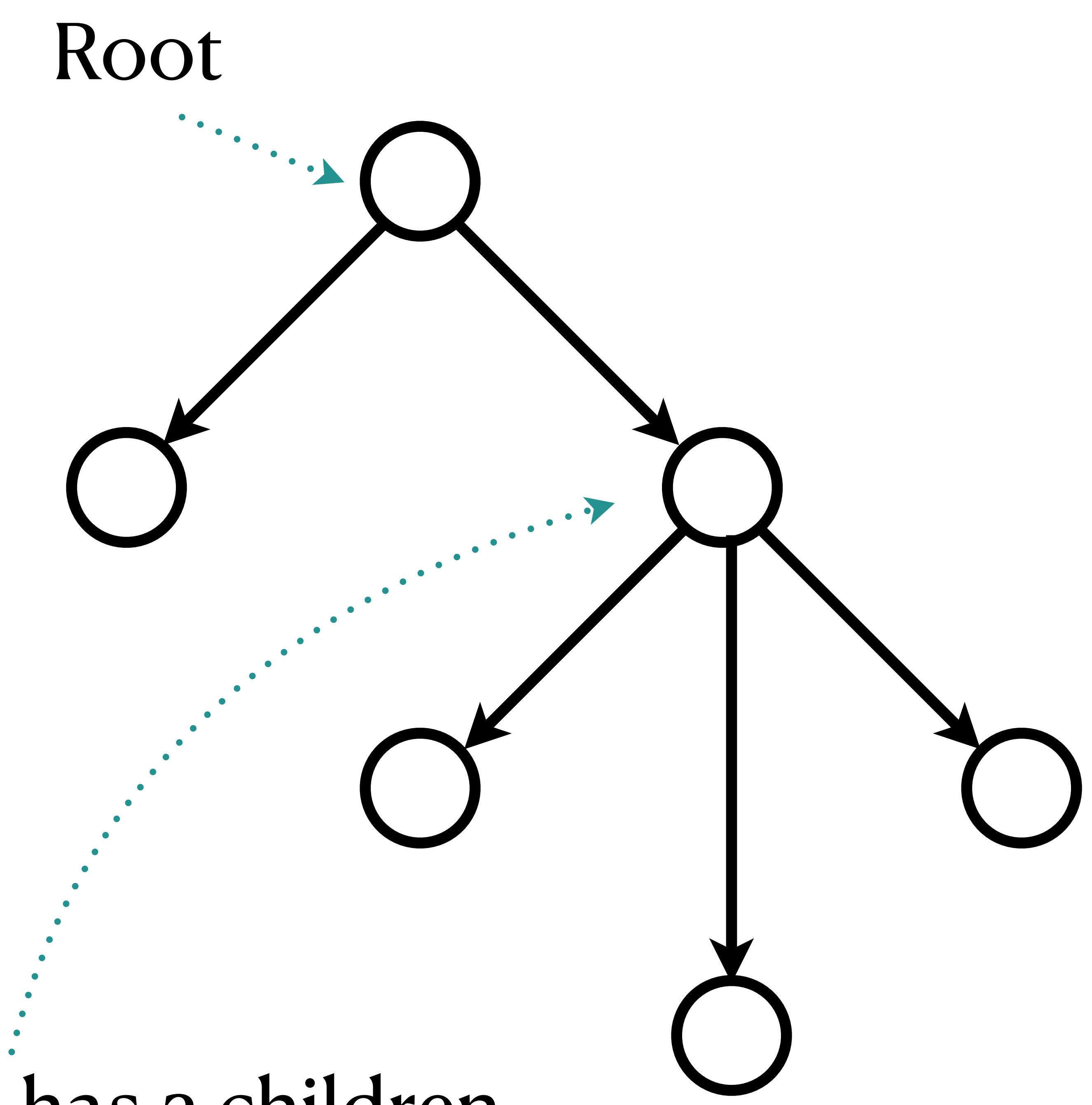
A Binary tree is a tree such that:

- Every node has ***<u>at most</u>*** two children



Root

Every node has 0,1 or 2 children ✓ A binary tree!



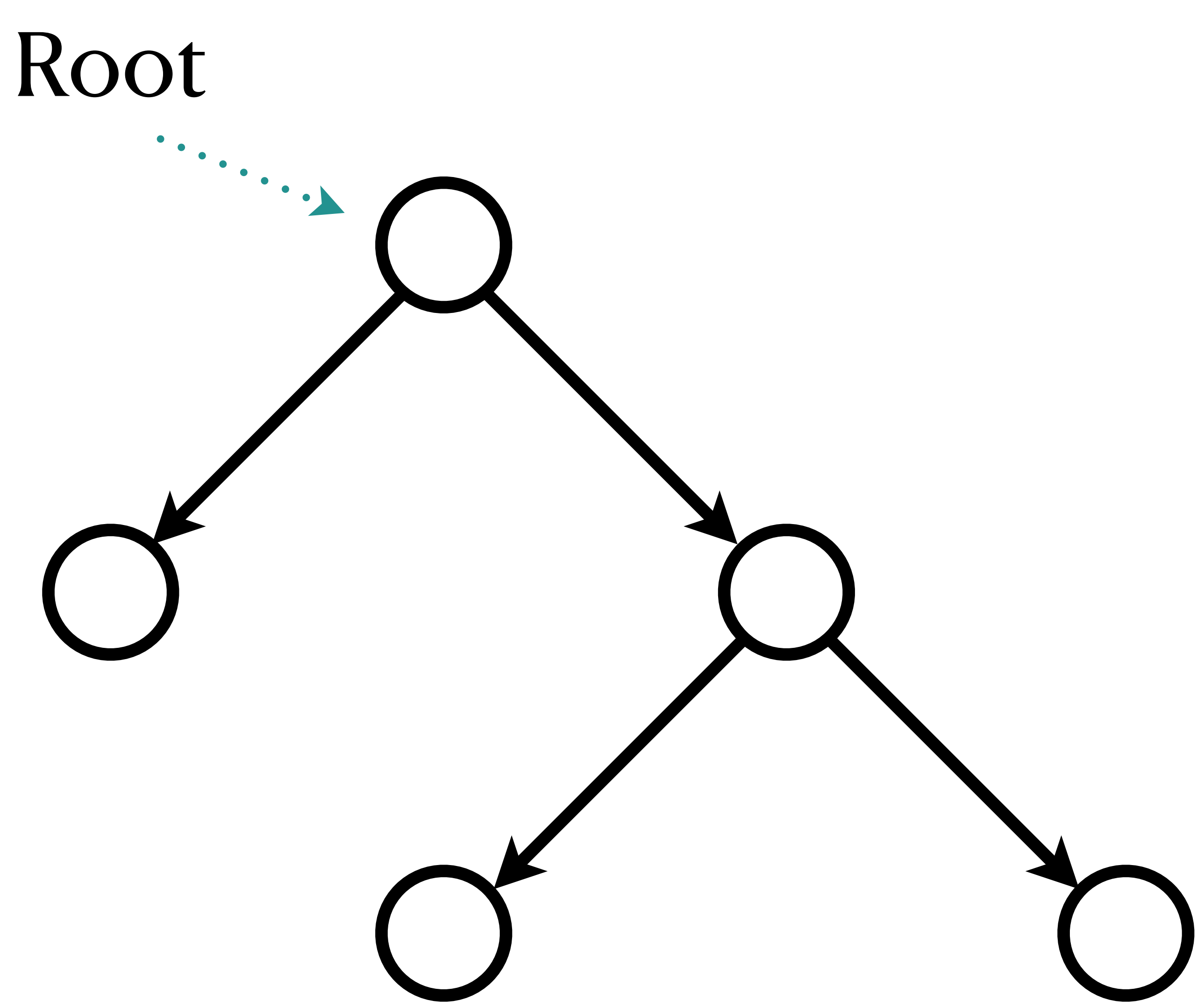
Root

This node has 3 children

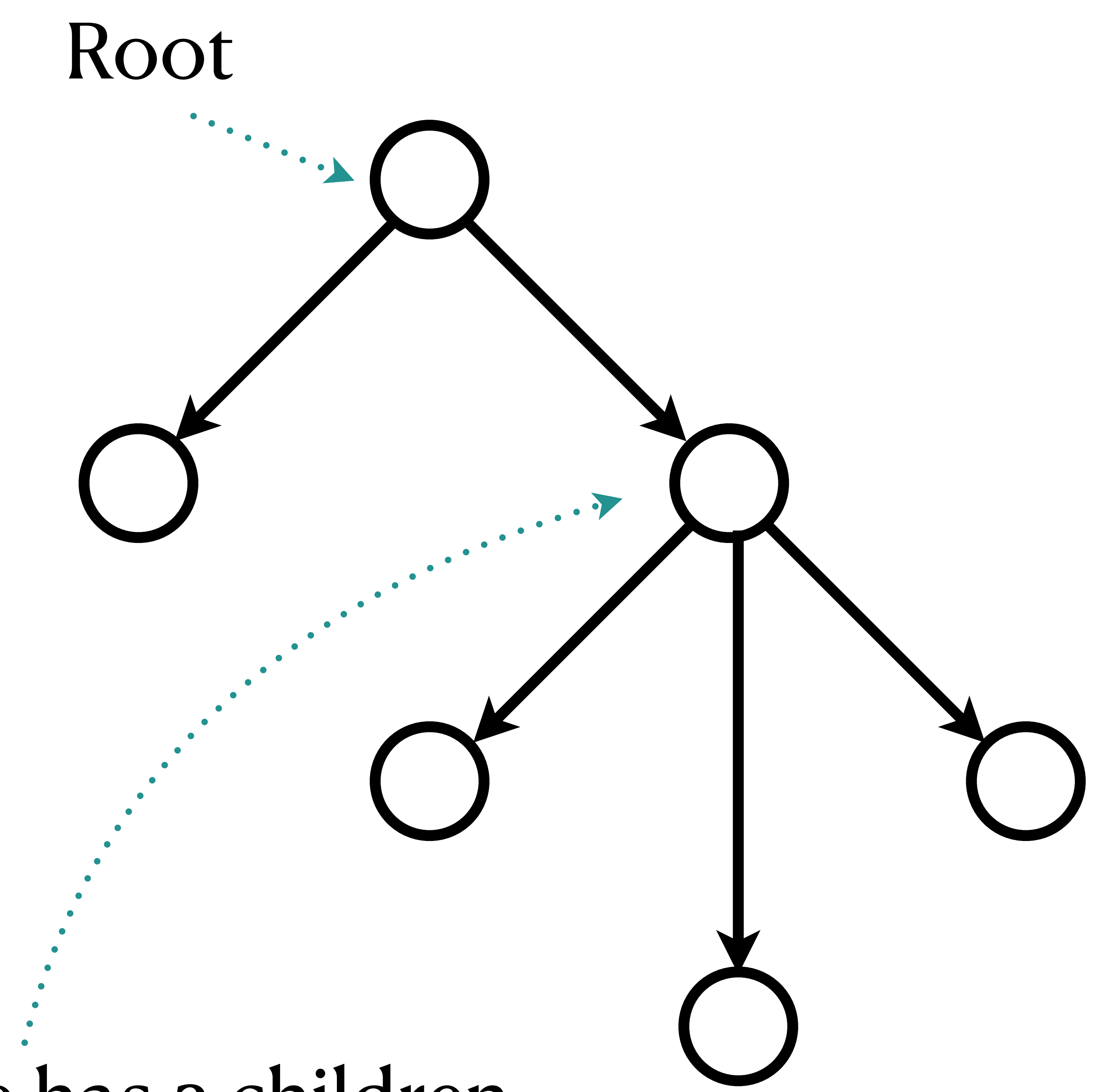
Review: Trees

A Binary tree is a tree such that:

- Every node has ***at most*** two children



Every node has 0,1 or 2 children ✓ A binary tree!



This node has 3 children

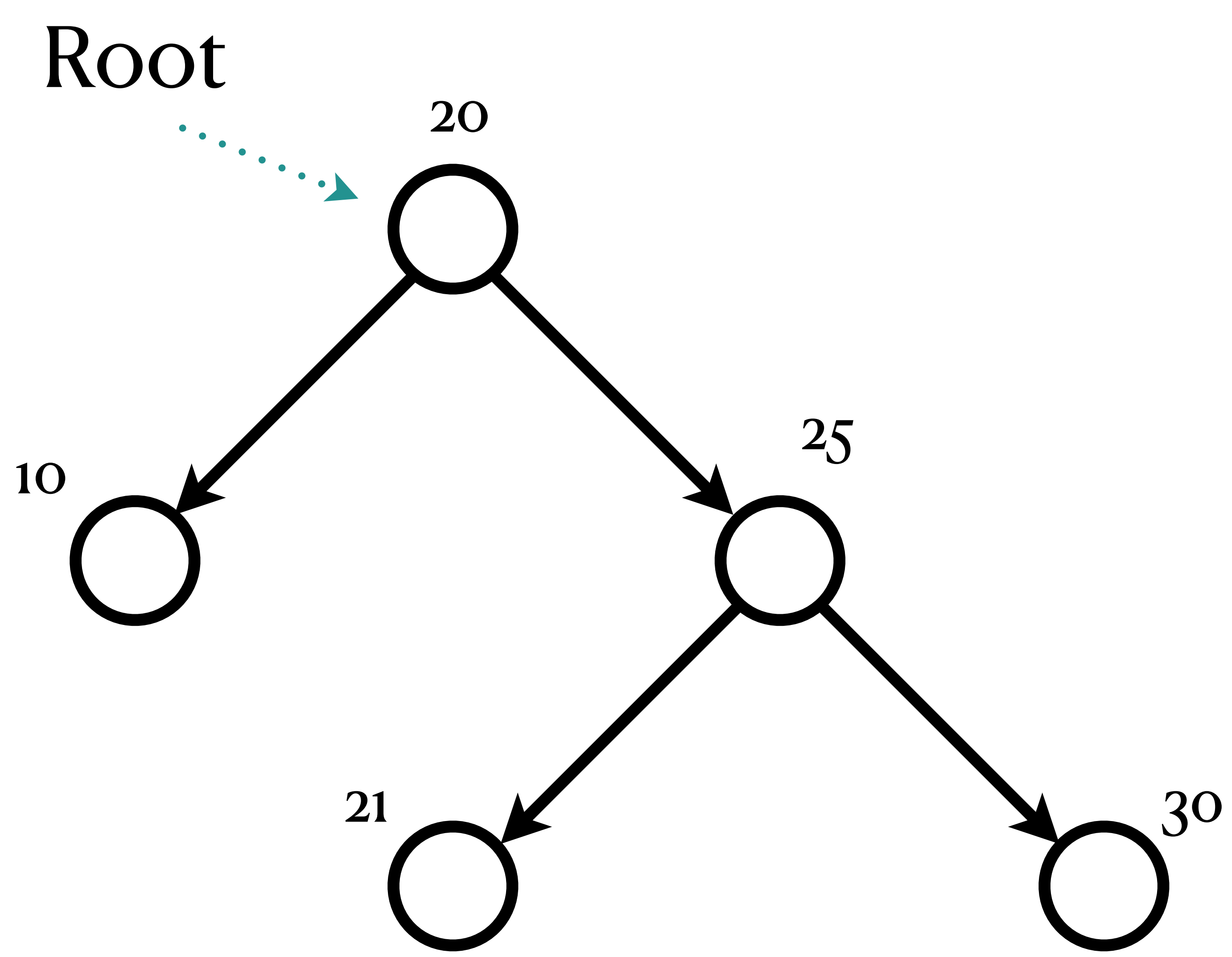
✗ Not a binary tree!

Binary Search Tree

A binary search tree is a binary tree with added property that:
Each node holds a value greater than all values in its left subtree and less than all values in its right subtree

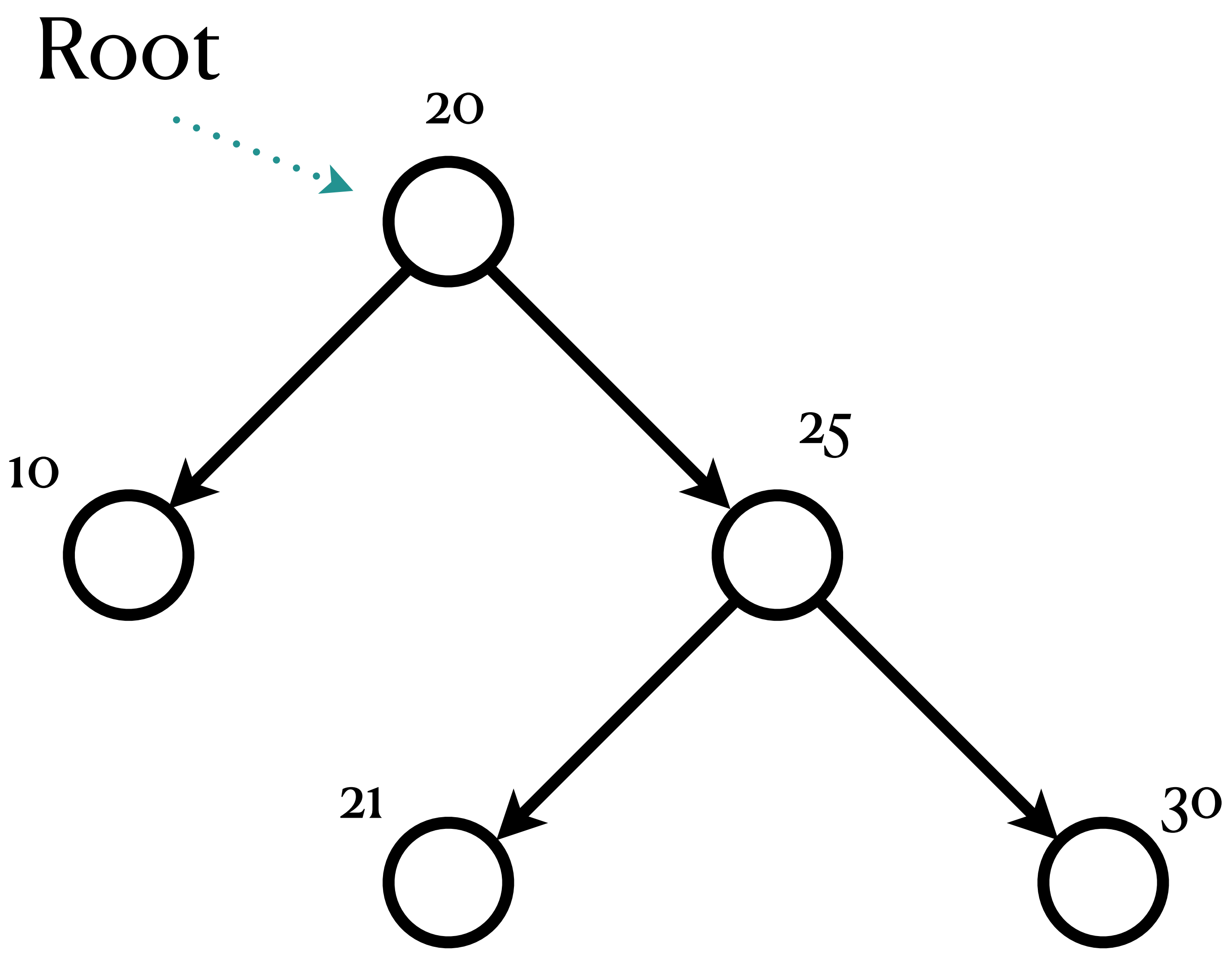
Binary Search Tree

A binary search tree is a binary tree with added property that:
Each node holds a value greater than all values in its left subtree and less than all values in its right subtree



Binary Search Tree

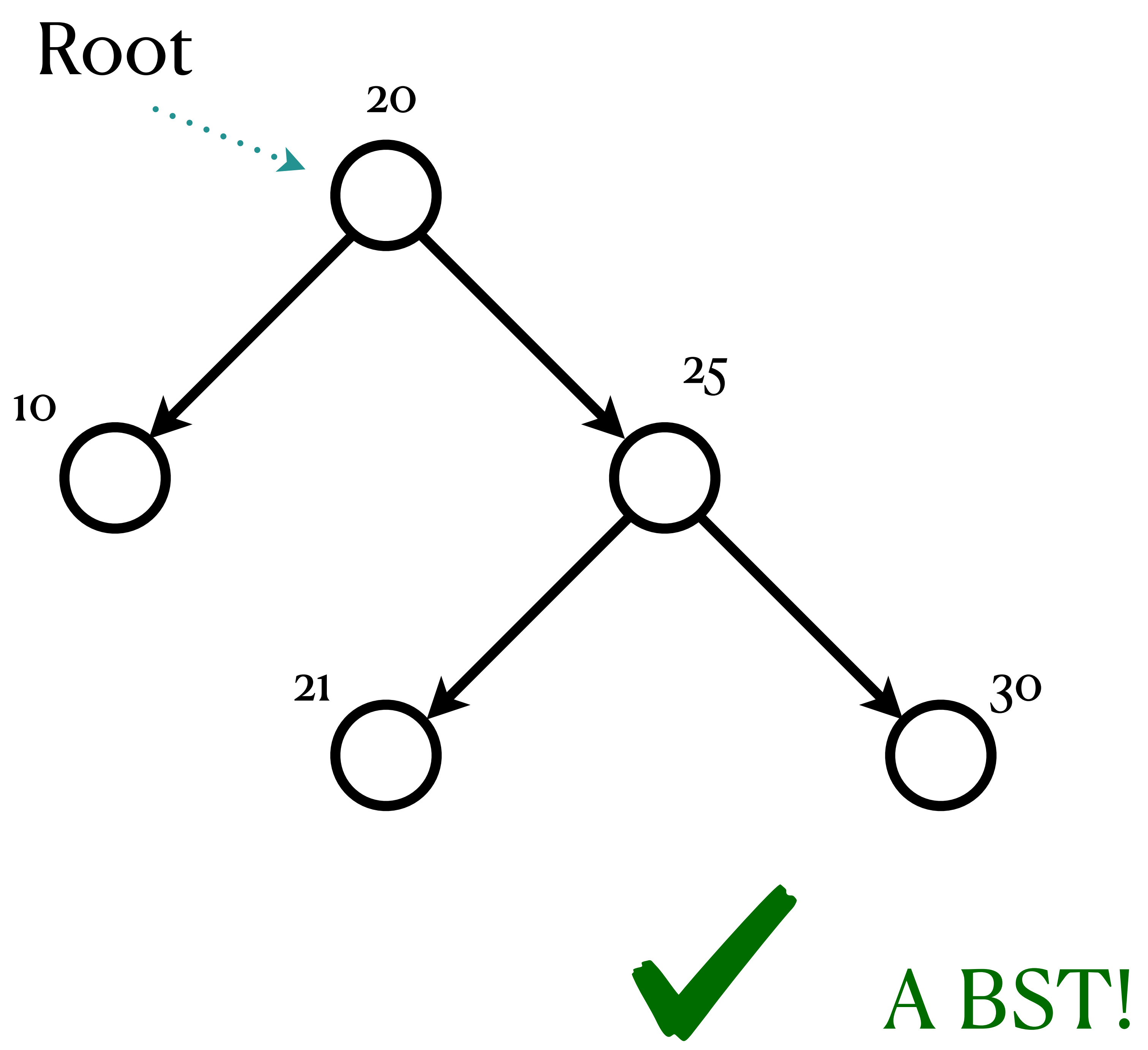
A binary search tree is a binary tree with added property that:
Each node holds a value greater than all values in its left subtree and less than all values in its right subtree

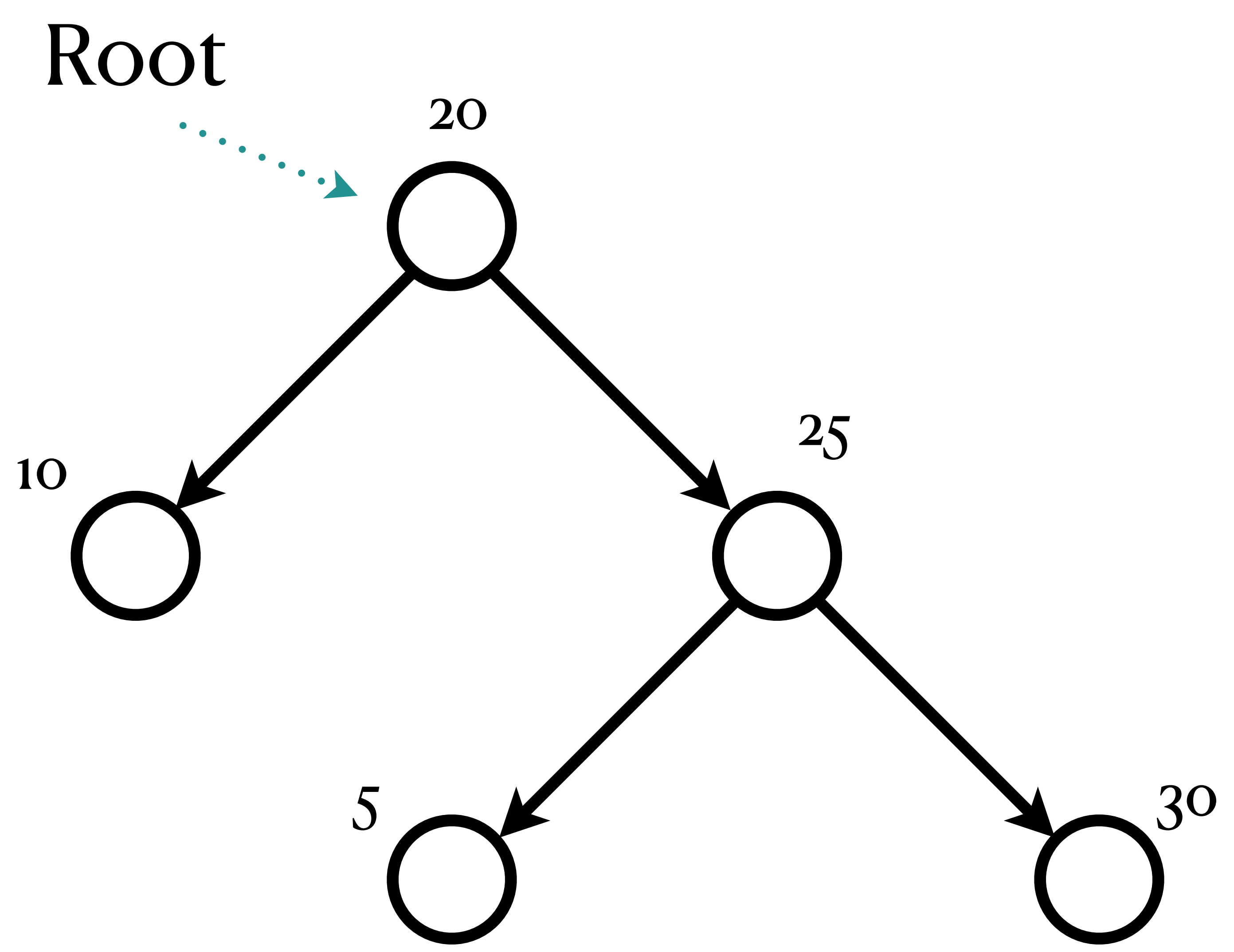


✓ A BST!

Binary Search Tree

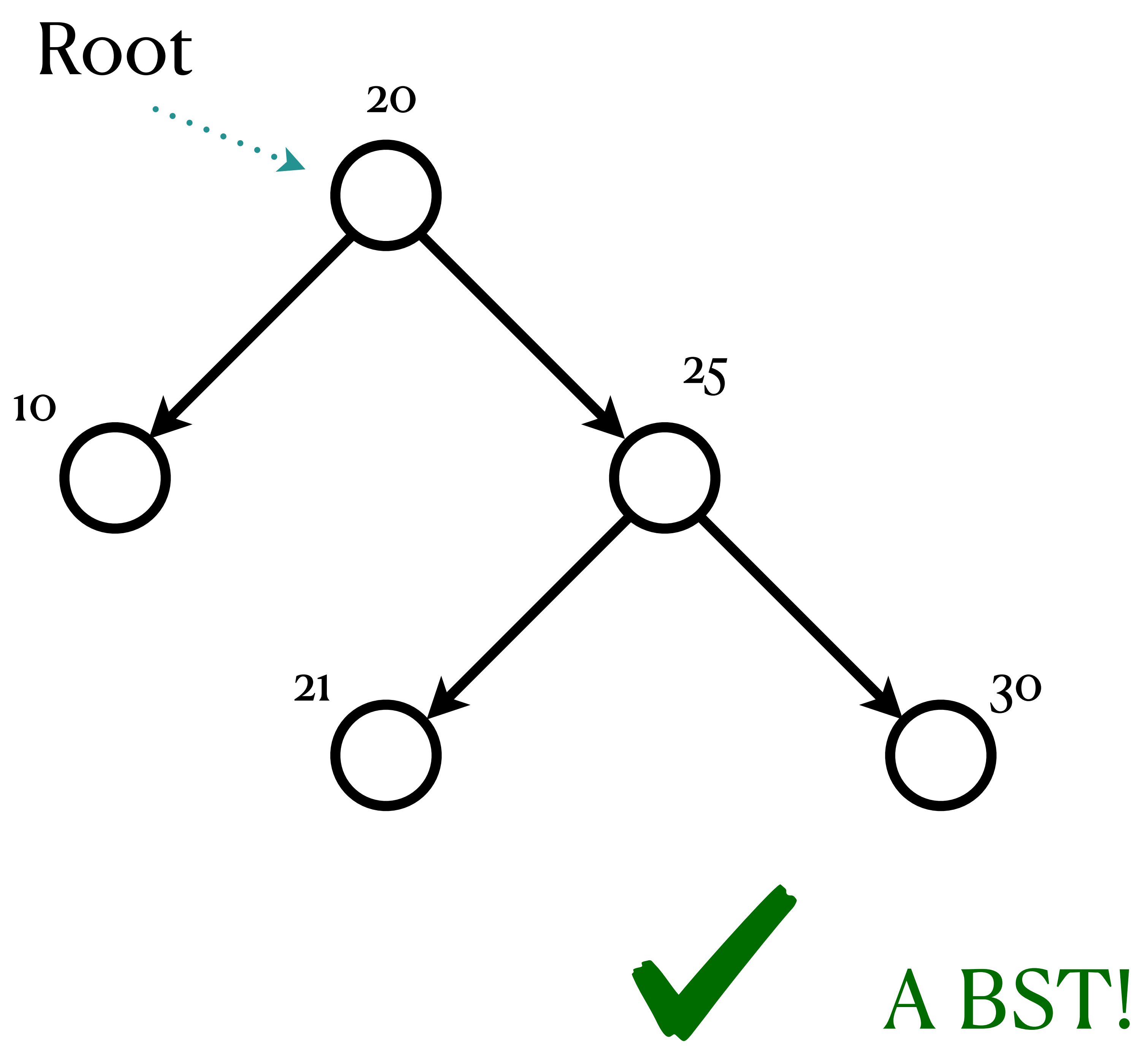
A binary search tree is a binary tree with added property that:
Each node holds a value greater than all values in its left subtree and less than all values in its right subtree

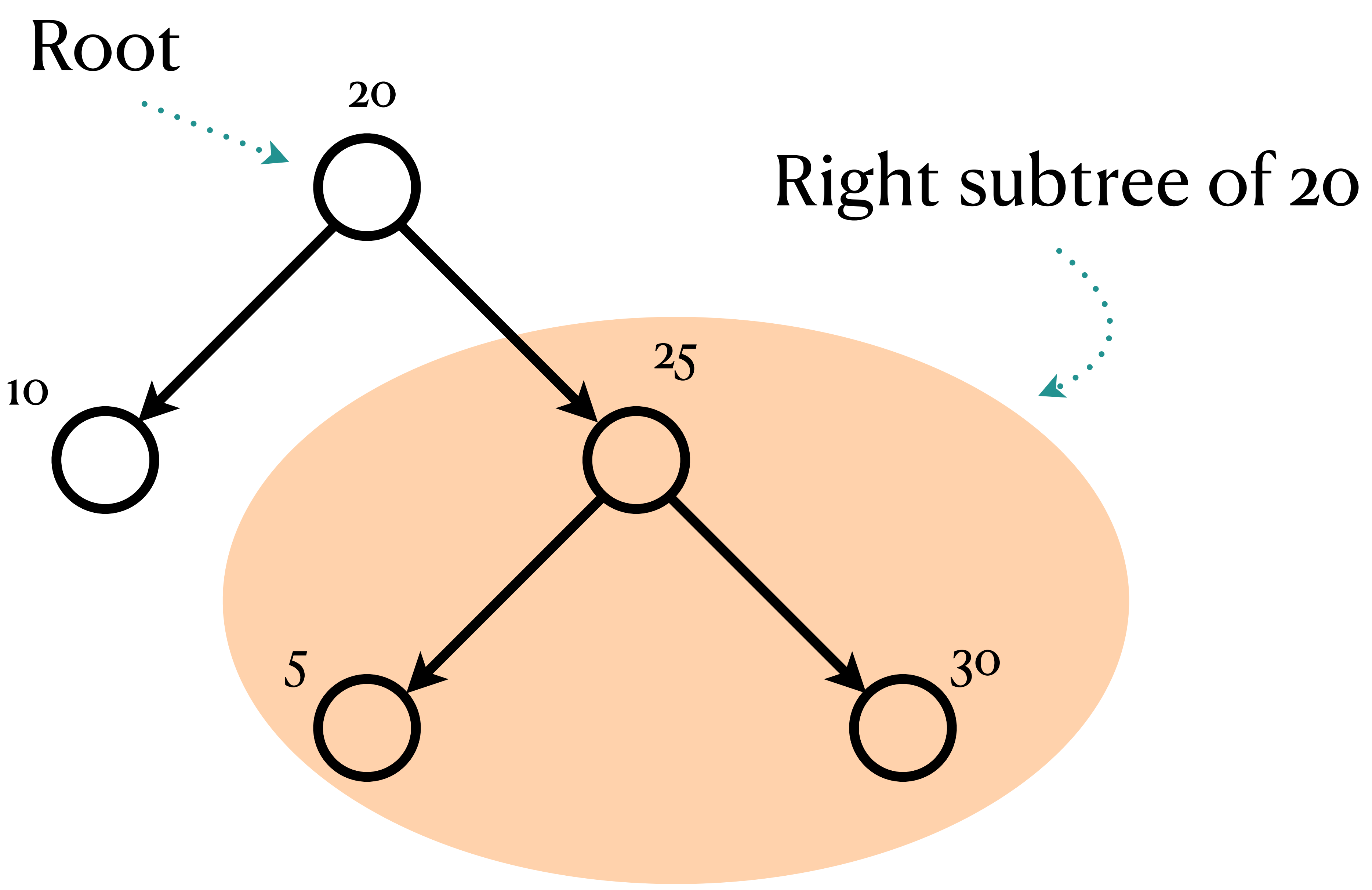




Binary Search Tree

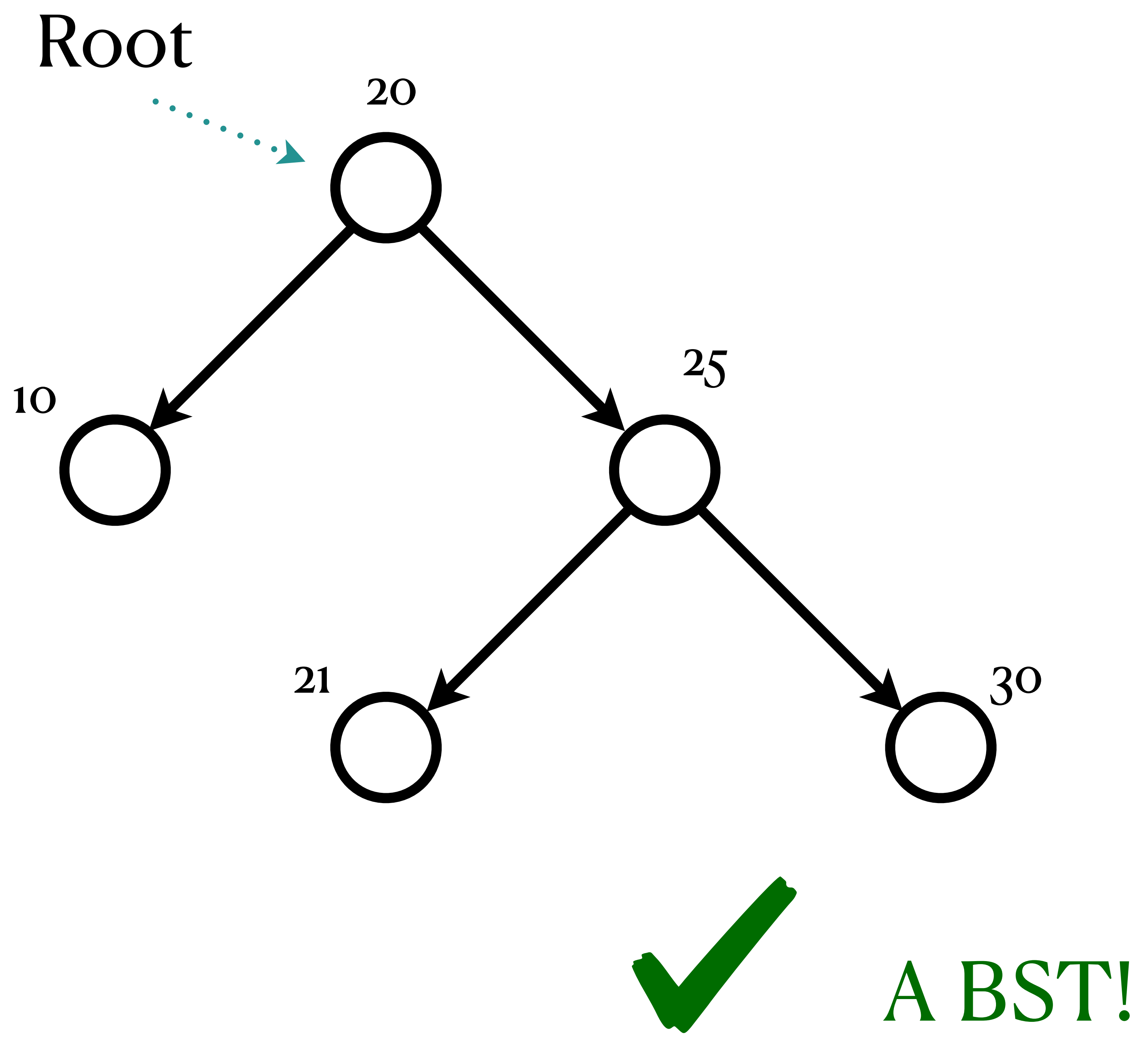
A binary search tree is a binary tree with added property that:
Each node holds a value greater than all values in its left subtree and less than all values in its right subtree



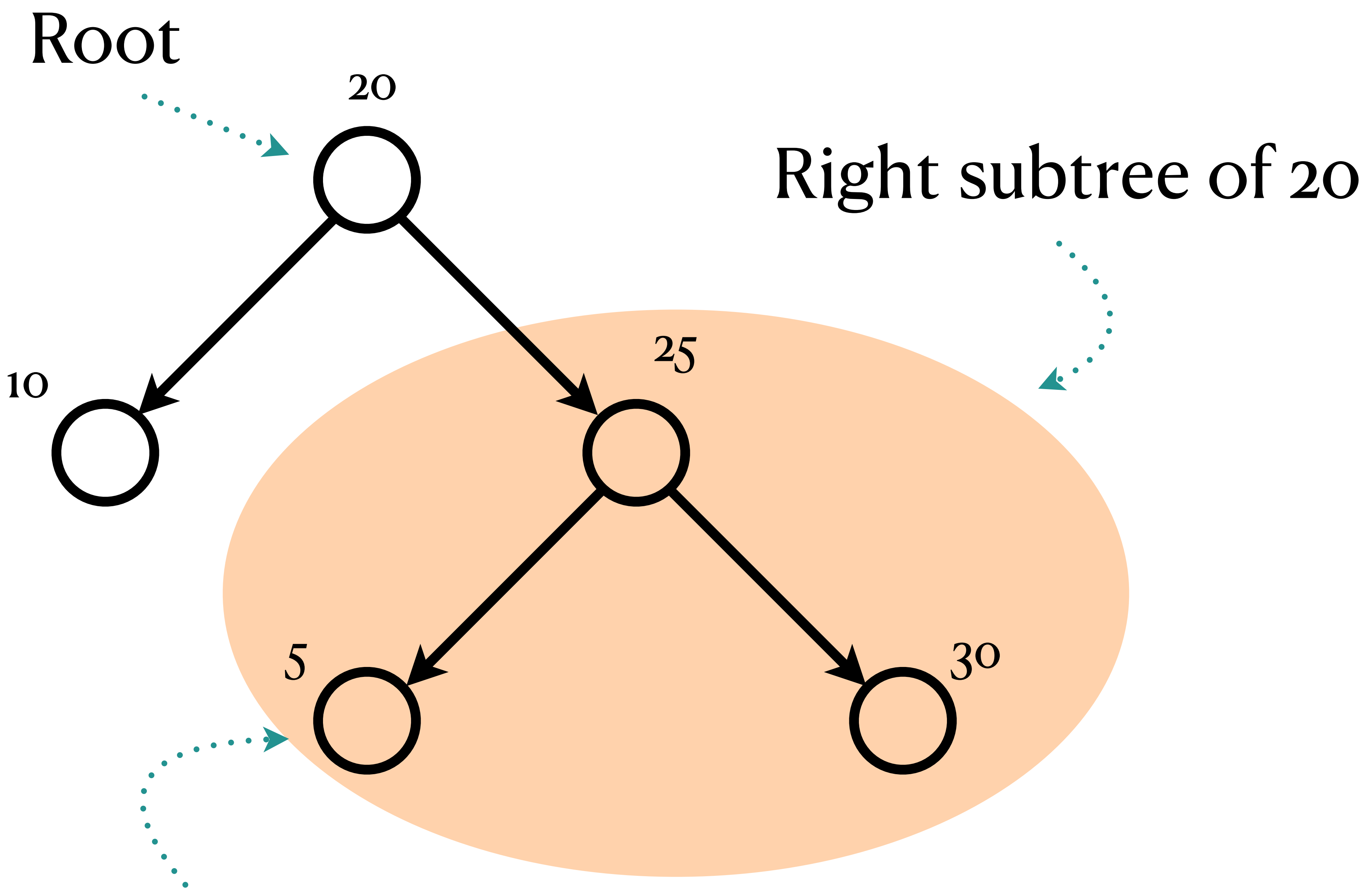


Binary Search Tree

A binary search tree is a binary tree with added property that:
Each node holds a value greater than all values in its left subtree and less than all values in its right subtree



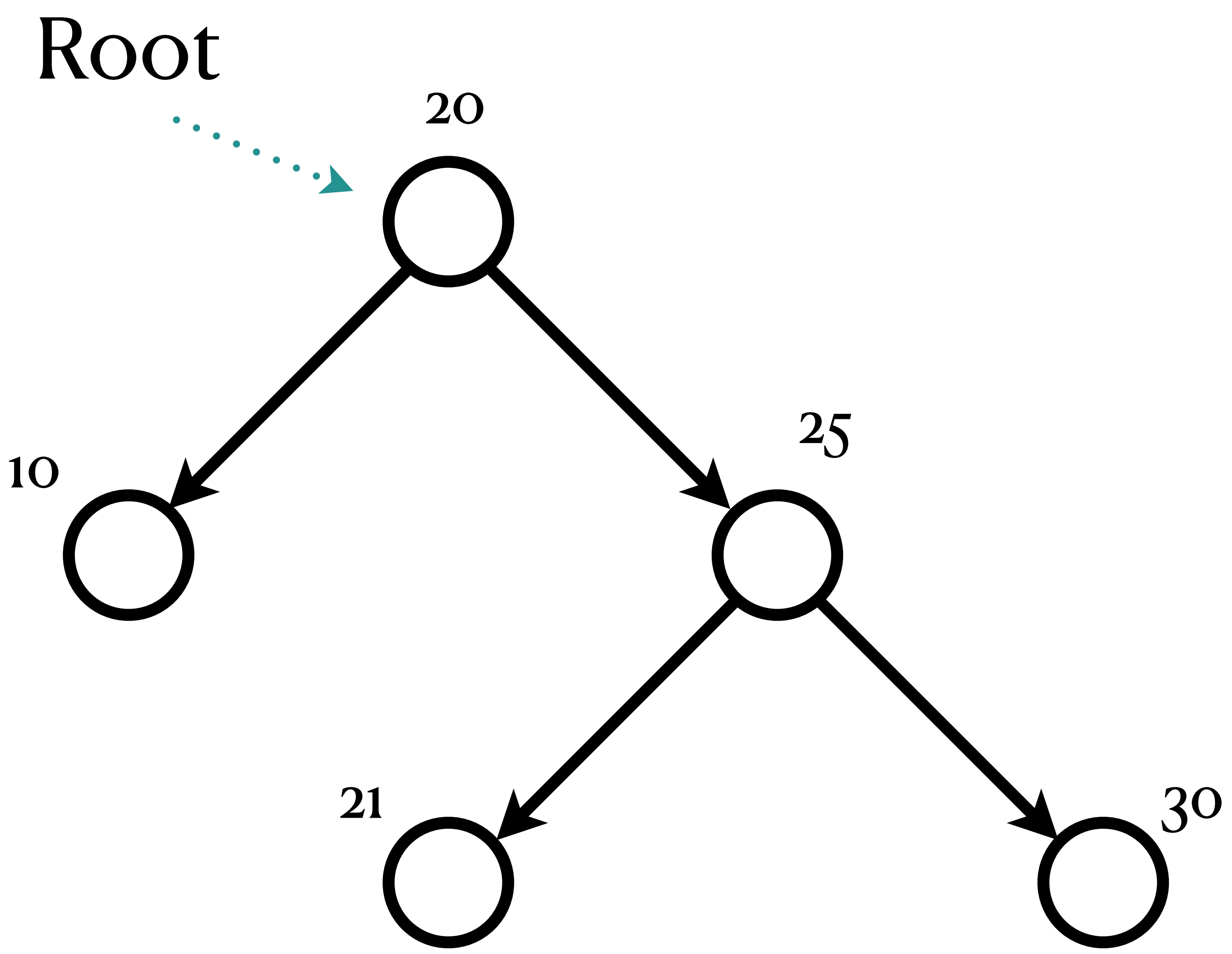
A BST!



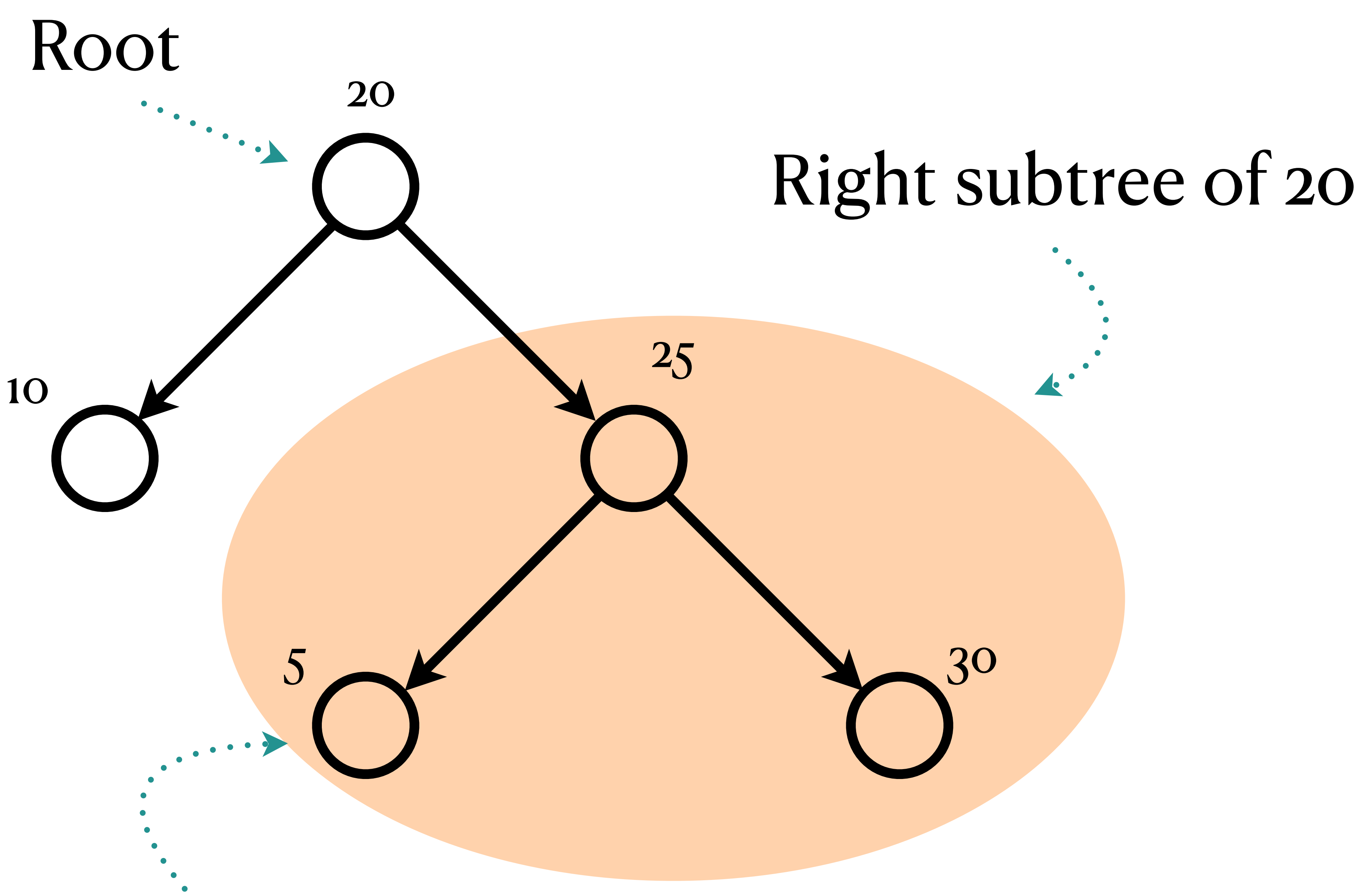
5 is in right subtree of 20
and has smaller value

Binary Search Tree

A binary search tree is a binary tree with added property that:
Each node holds a value greater than all values in its left subtree and less than all values in its right subtree



✓ A BST!



Right subtree of 20

5 is in right subtree of 20
and has smaller value

✗ Not a BST!

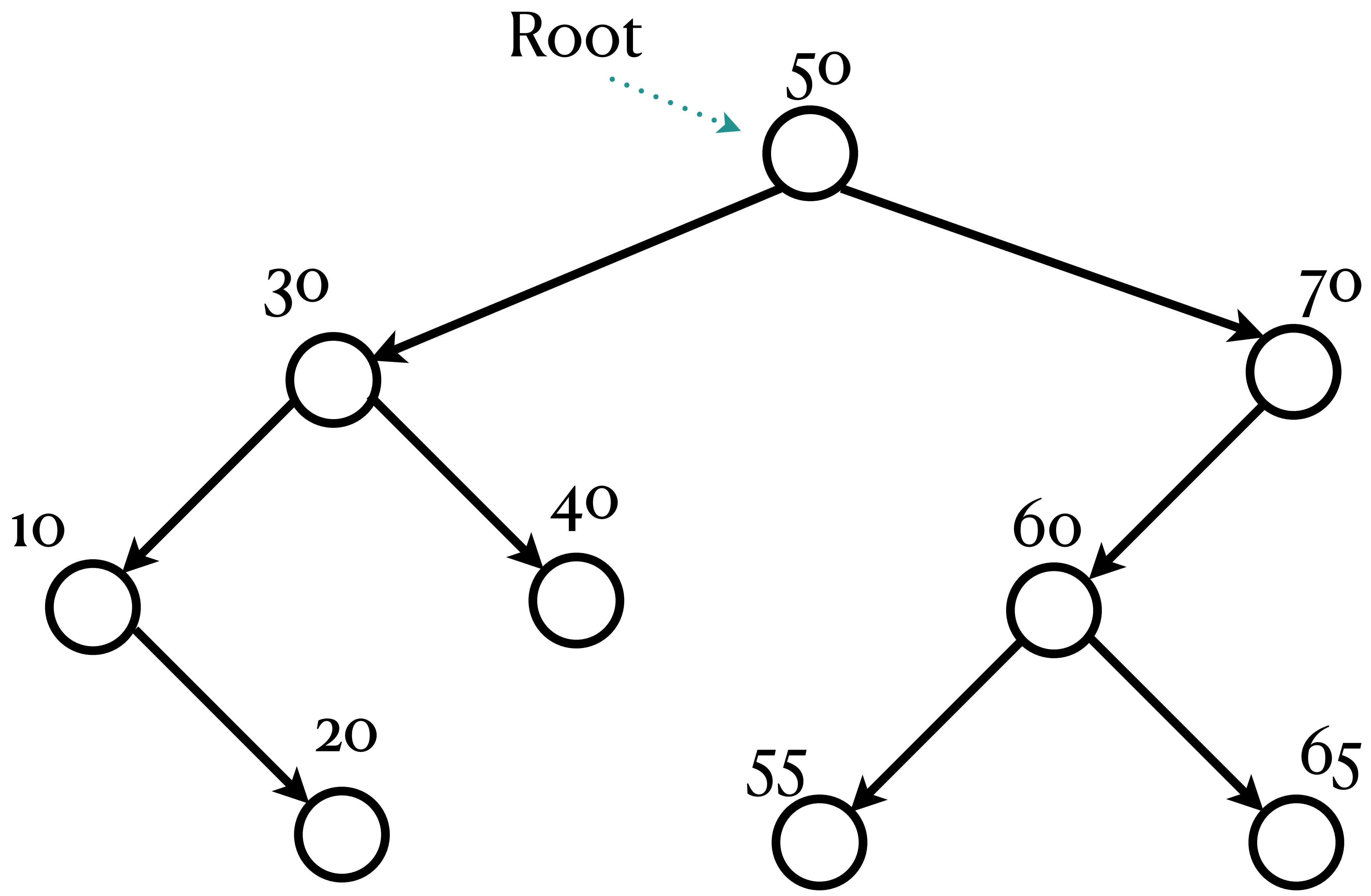
BSTs: Operations

Search

Insertion

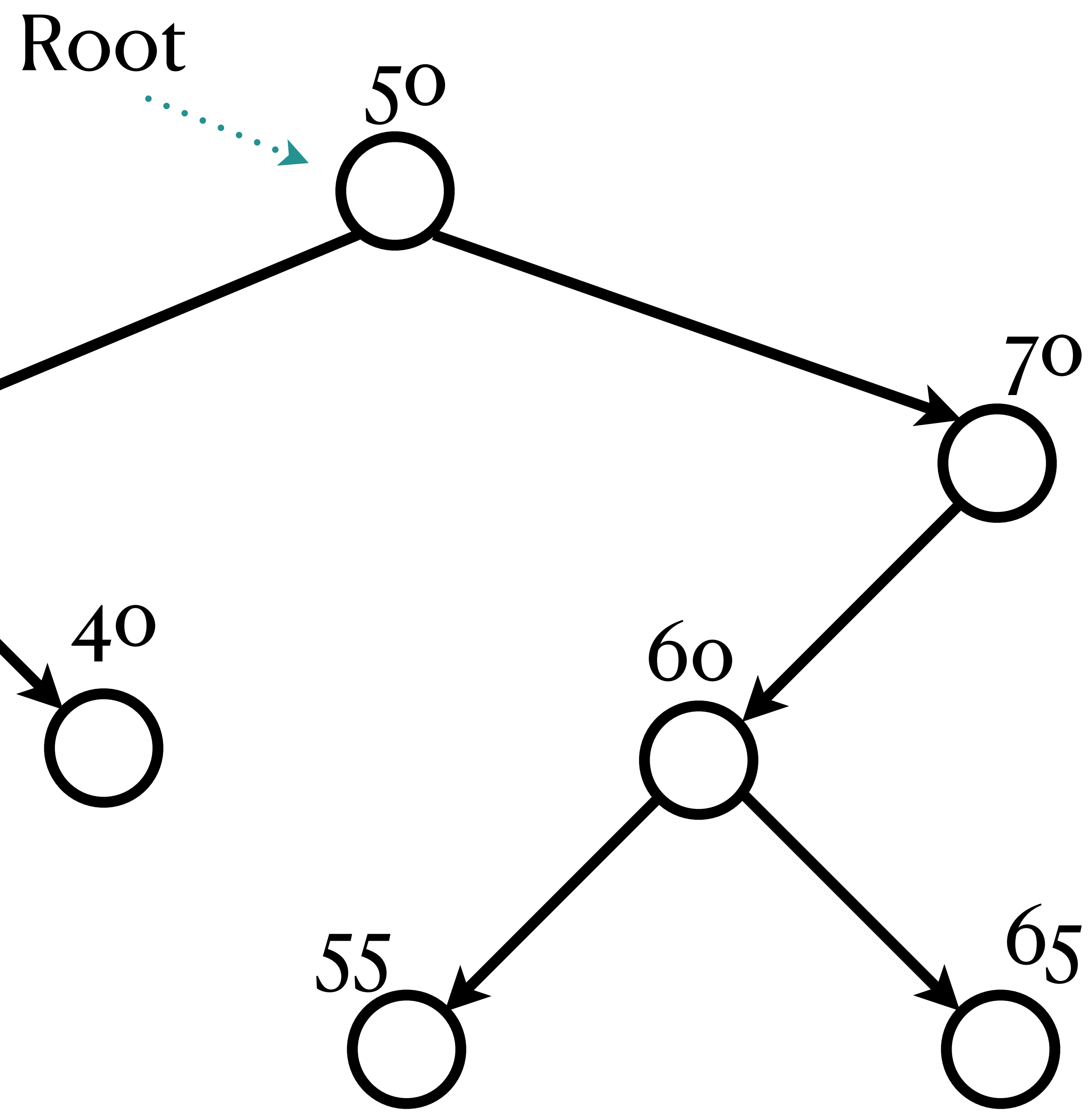
Deletion

Search in BSTs

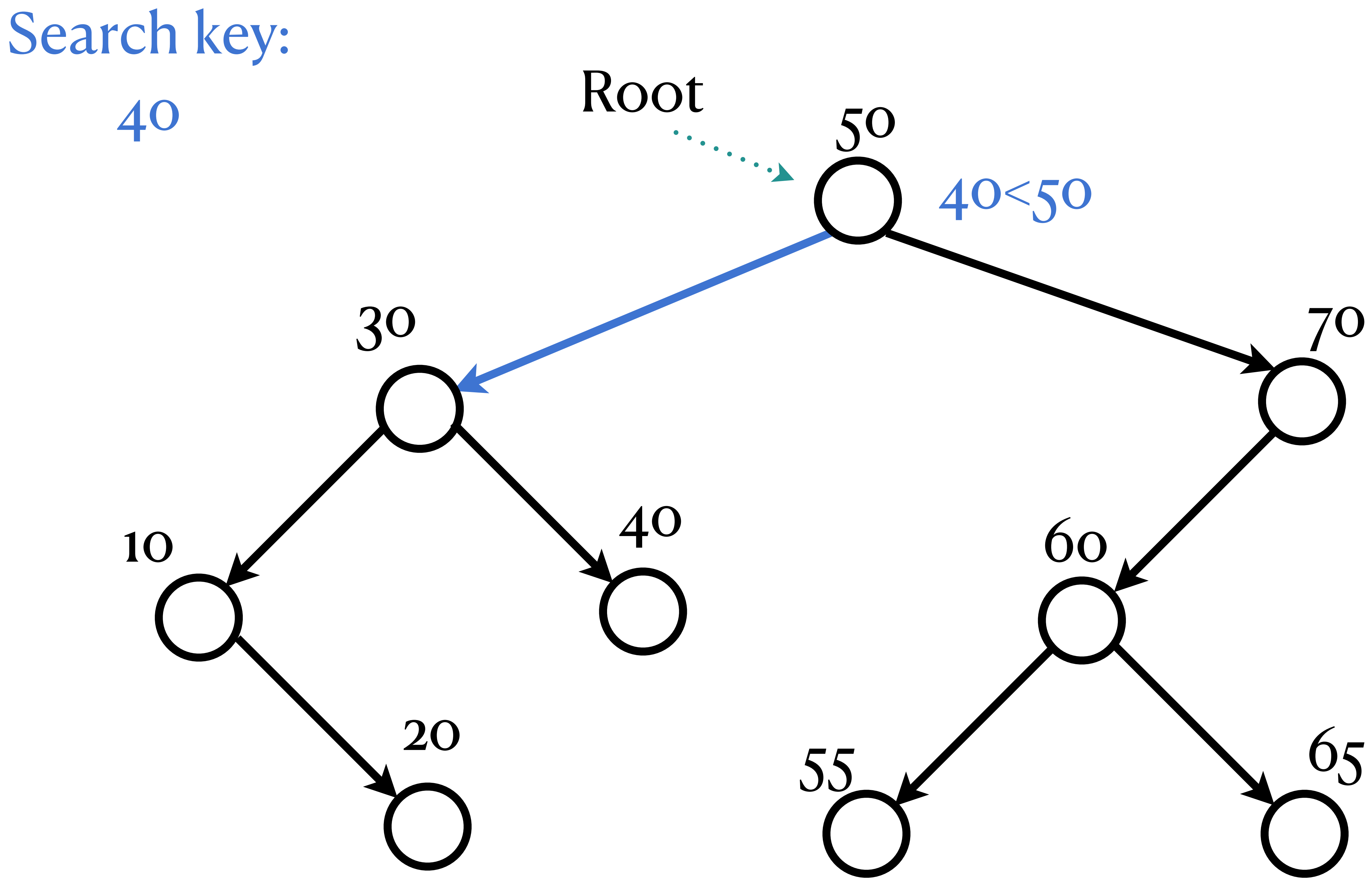


Search in BSTs

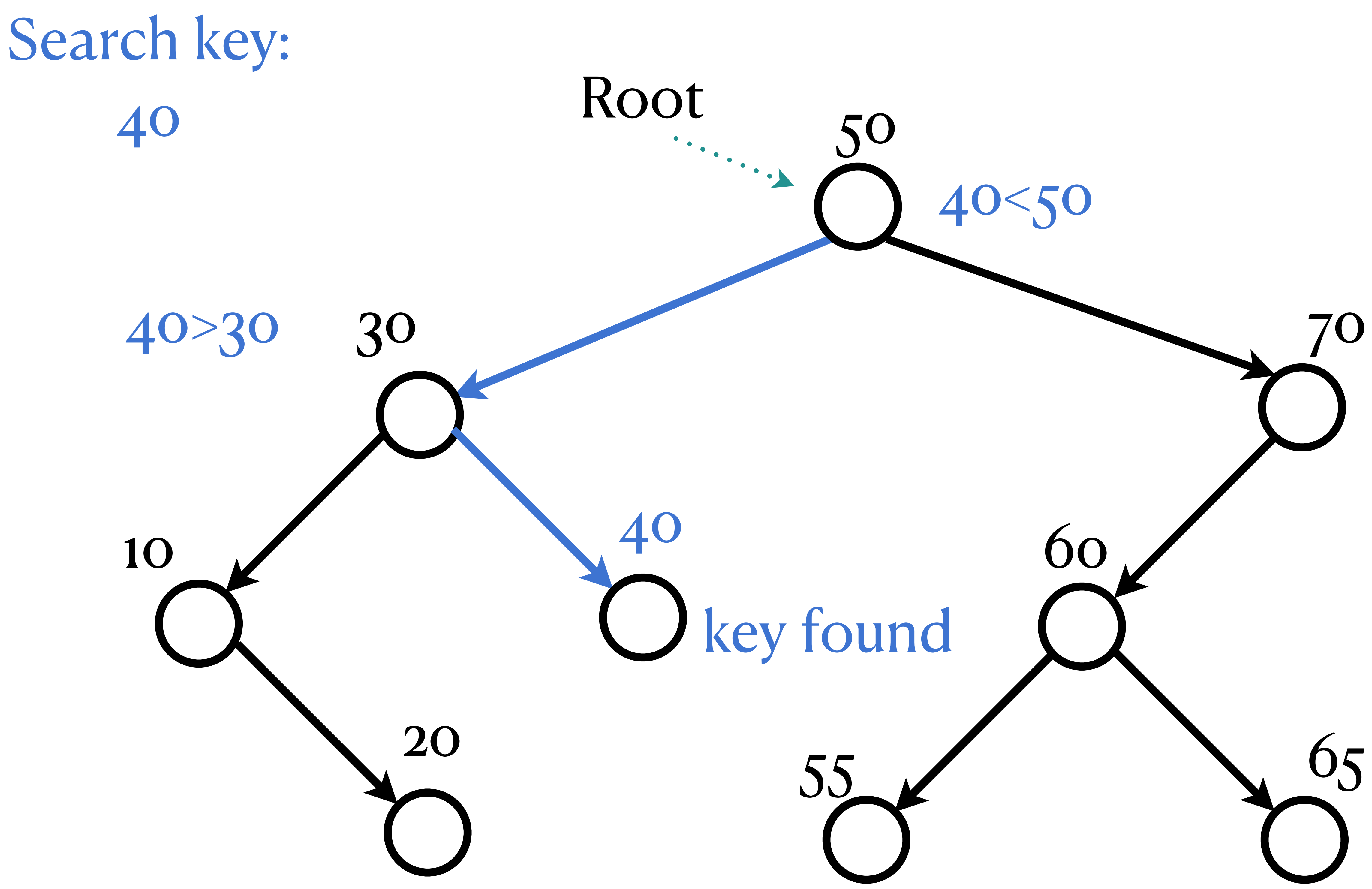
Search key:
40



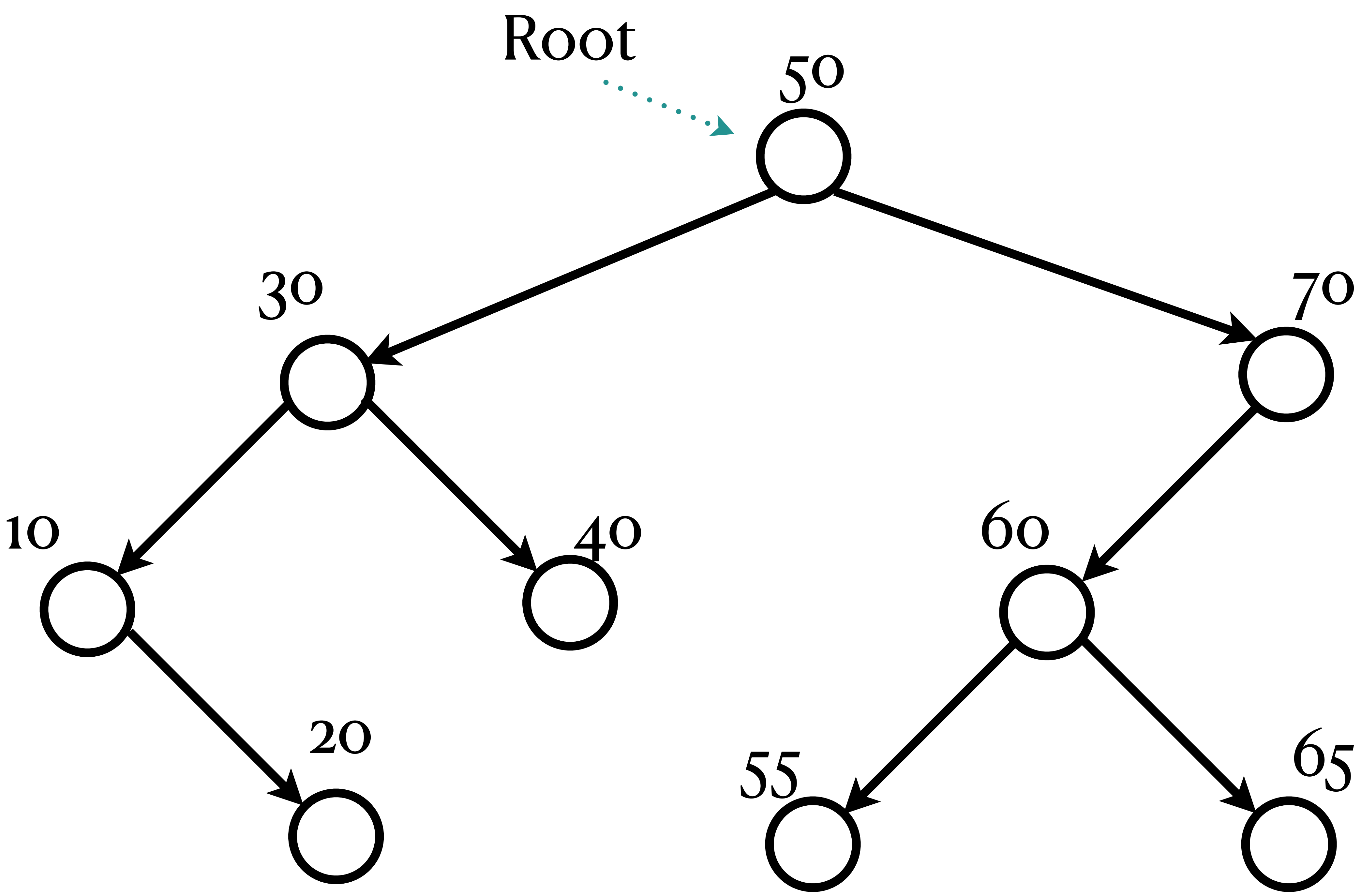
Search in BSTs



Search in BSTs

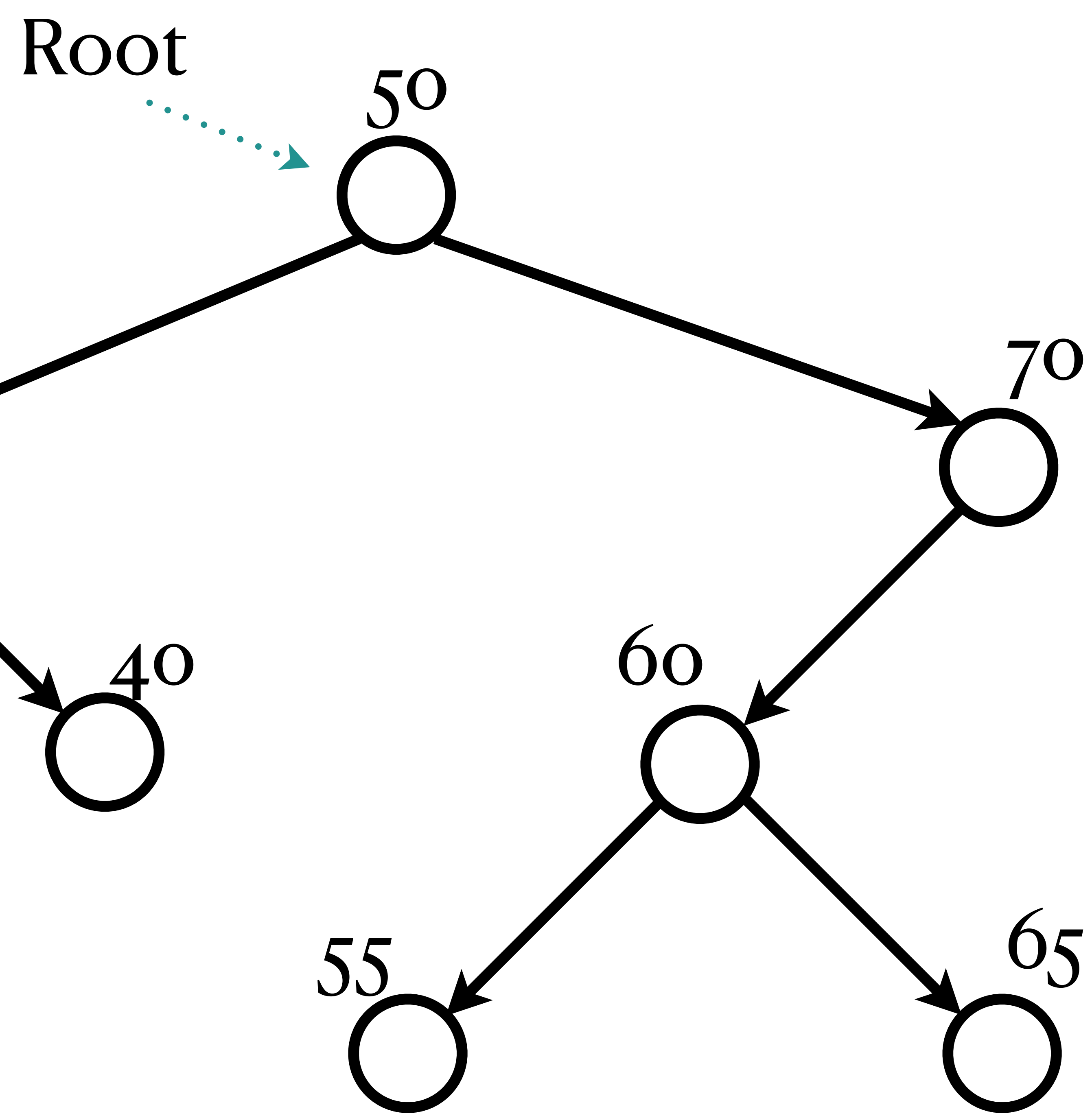


Search in BSTs

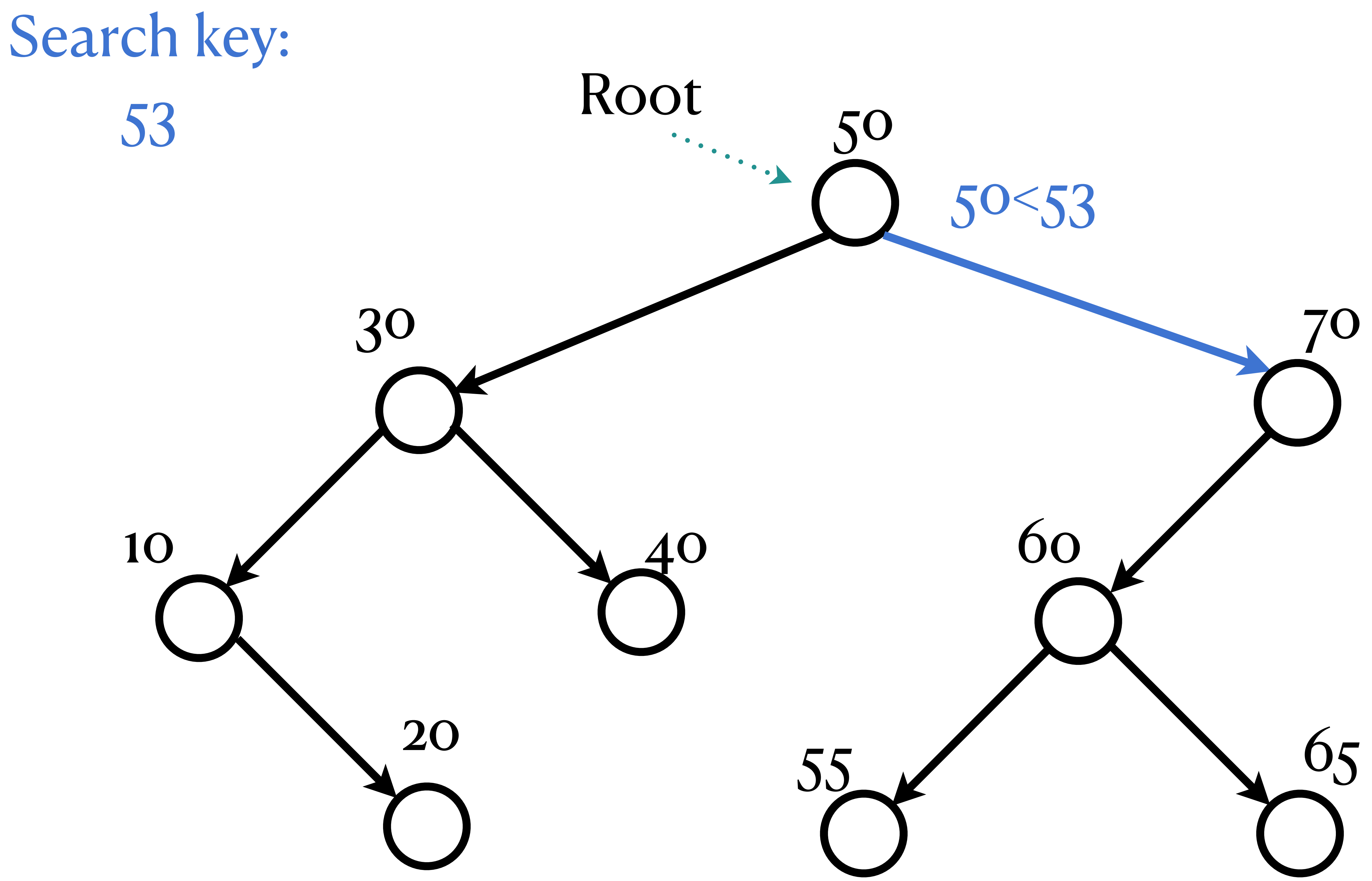


Search in BSTs

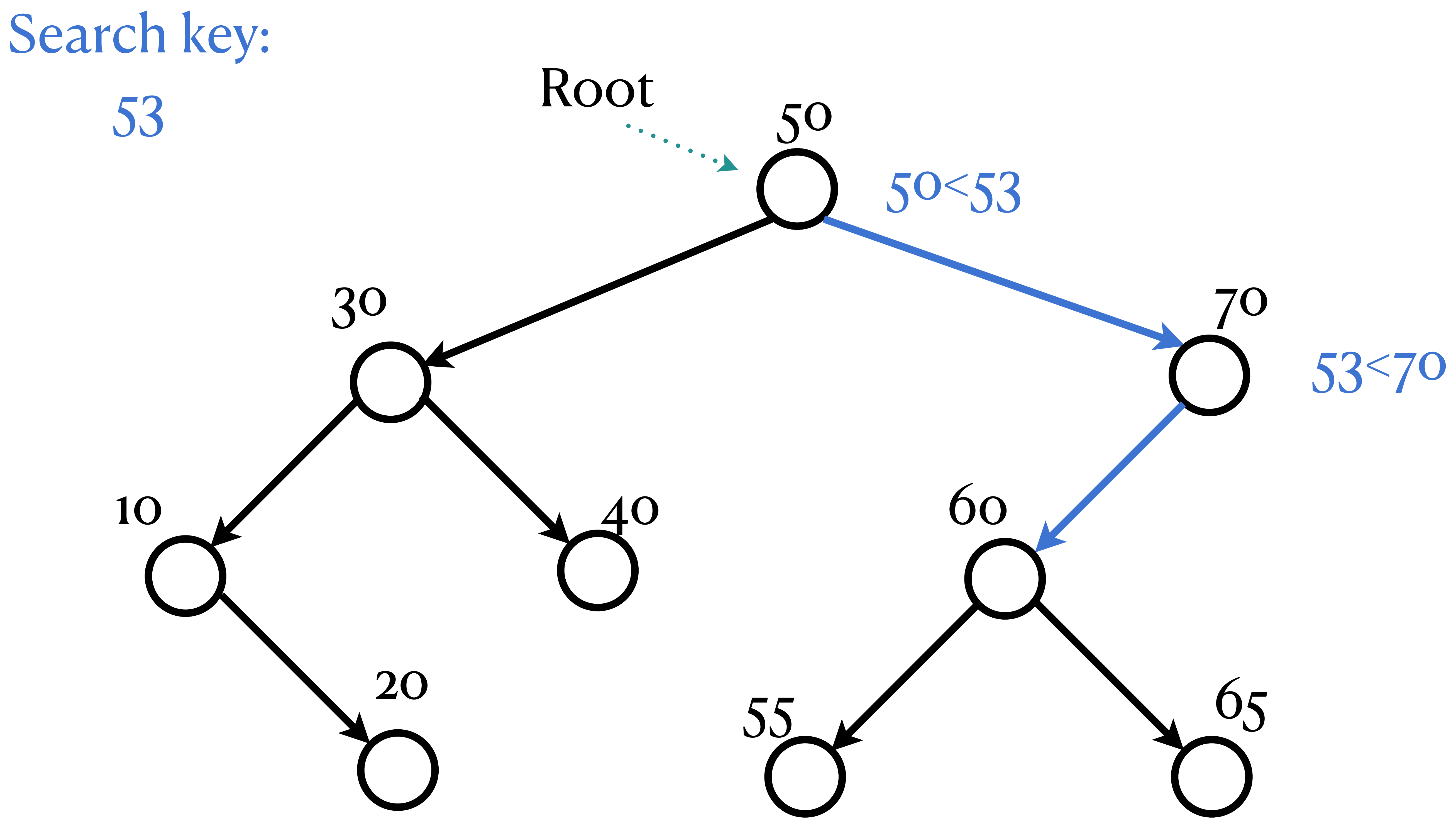
Search key:
53



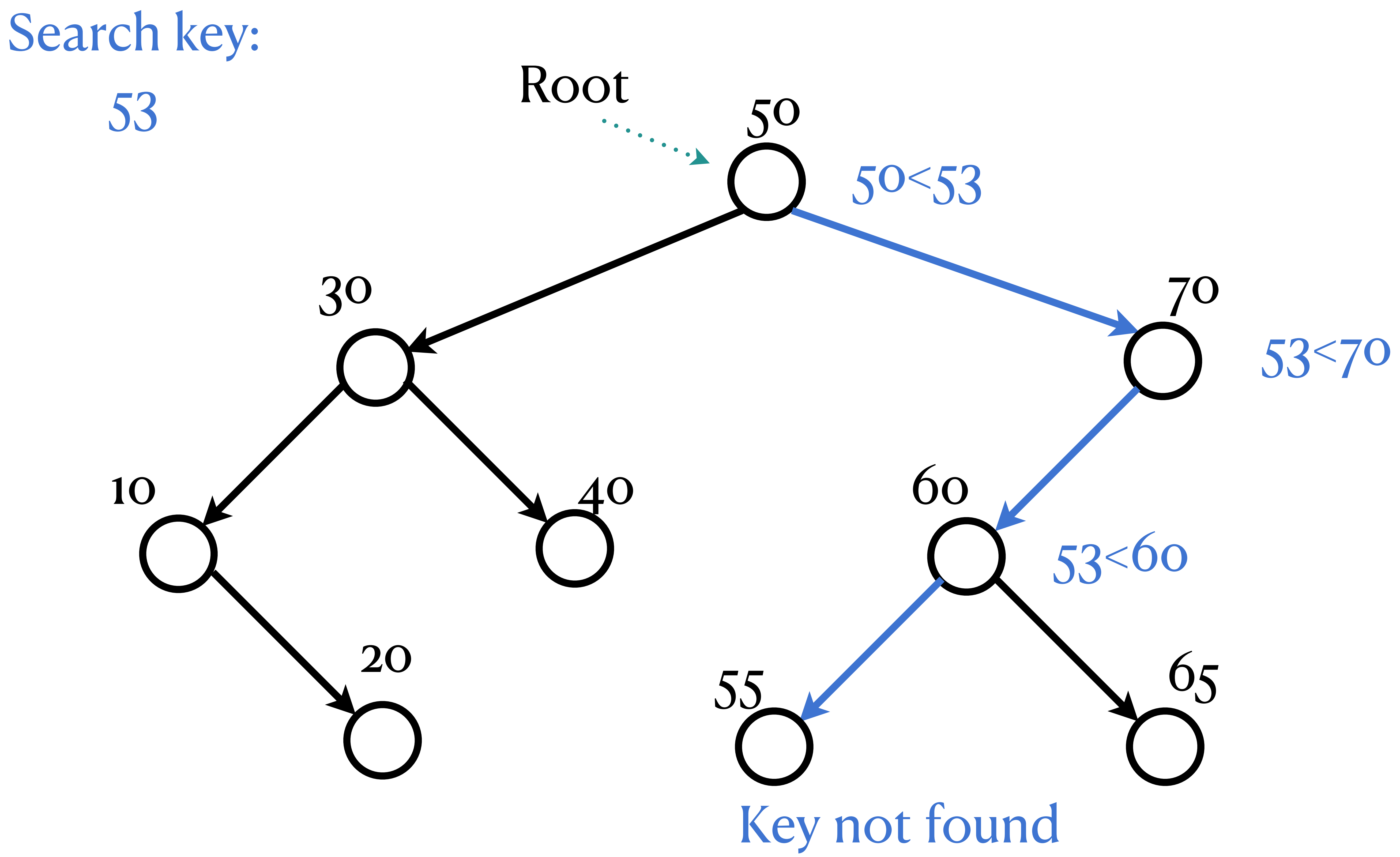
Search in BSTs



Search in BSTs

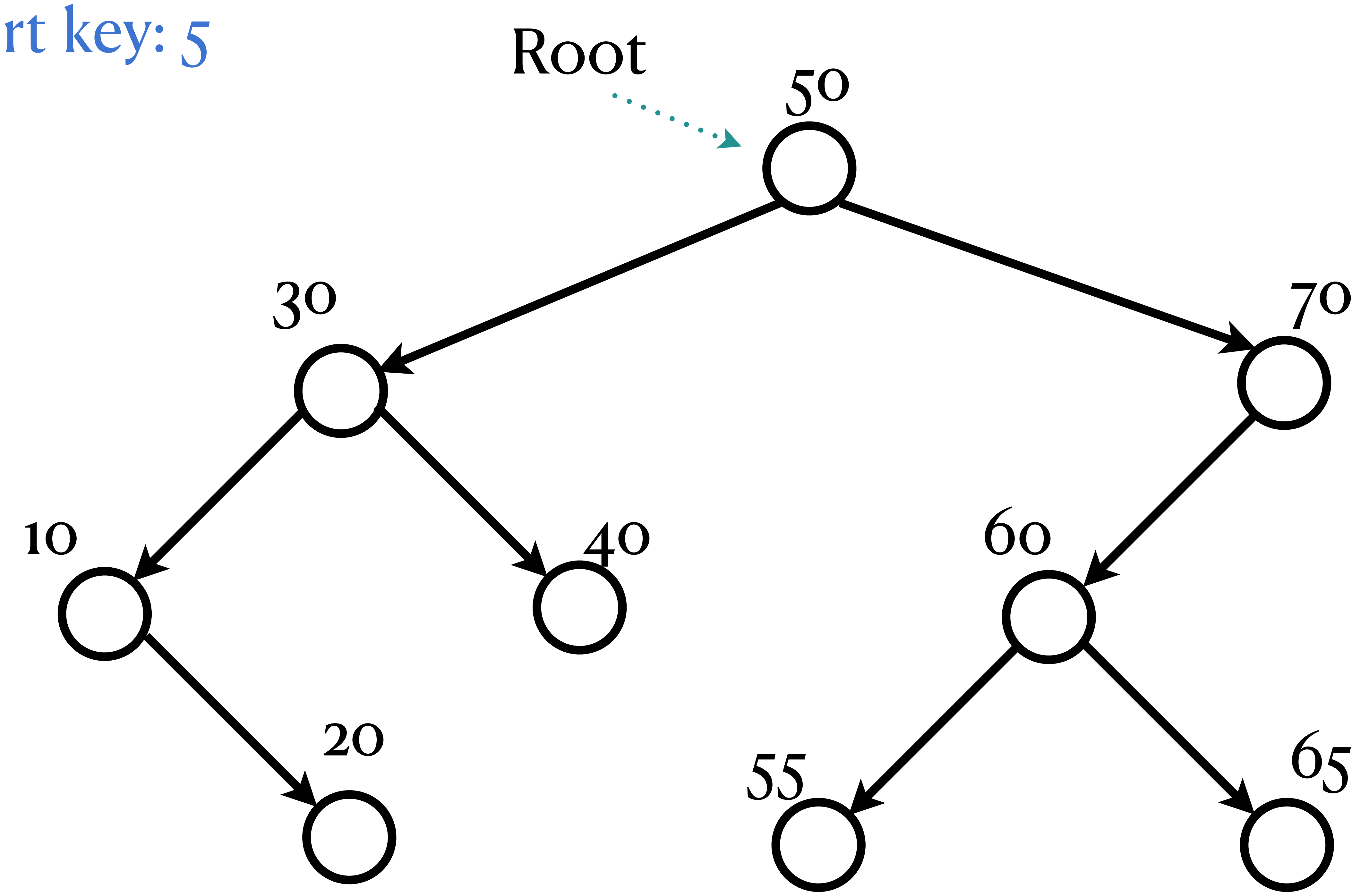


Search in BSTs

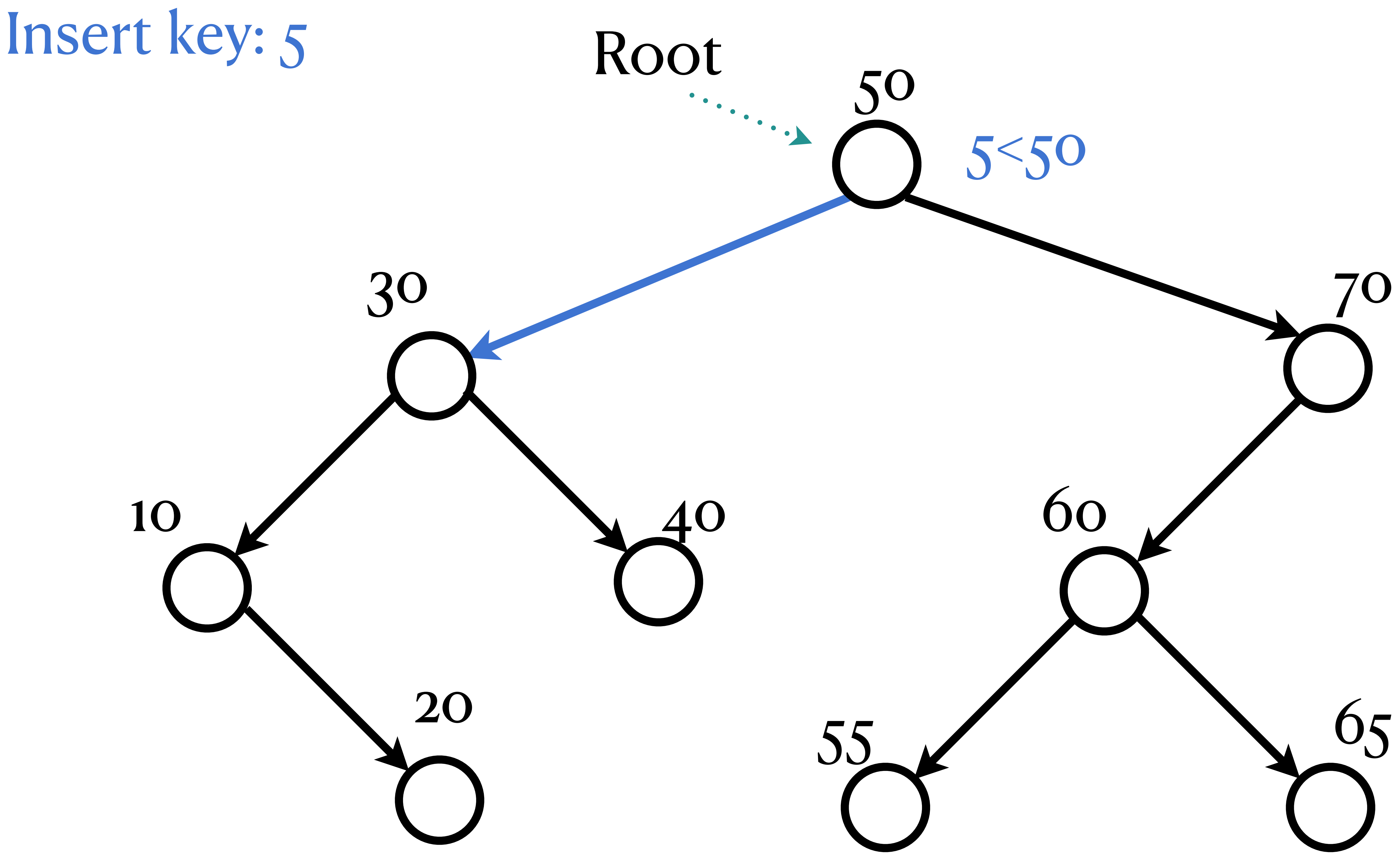


BST: Insertion

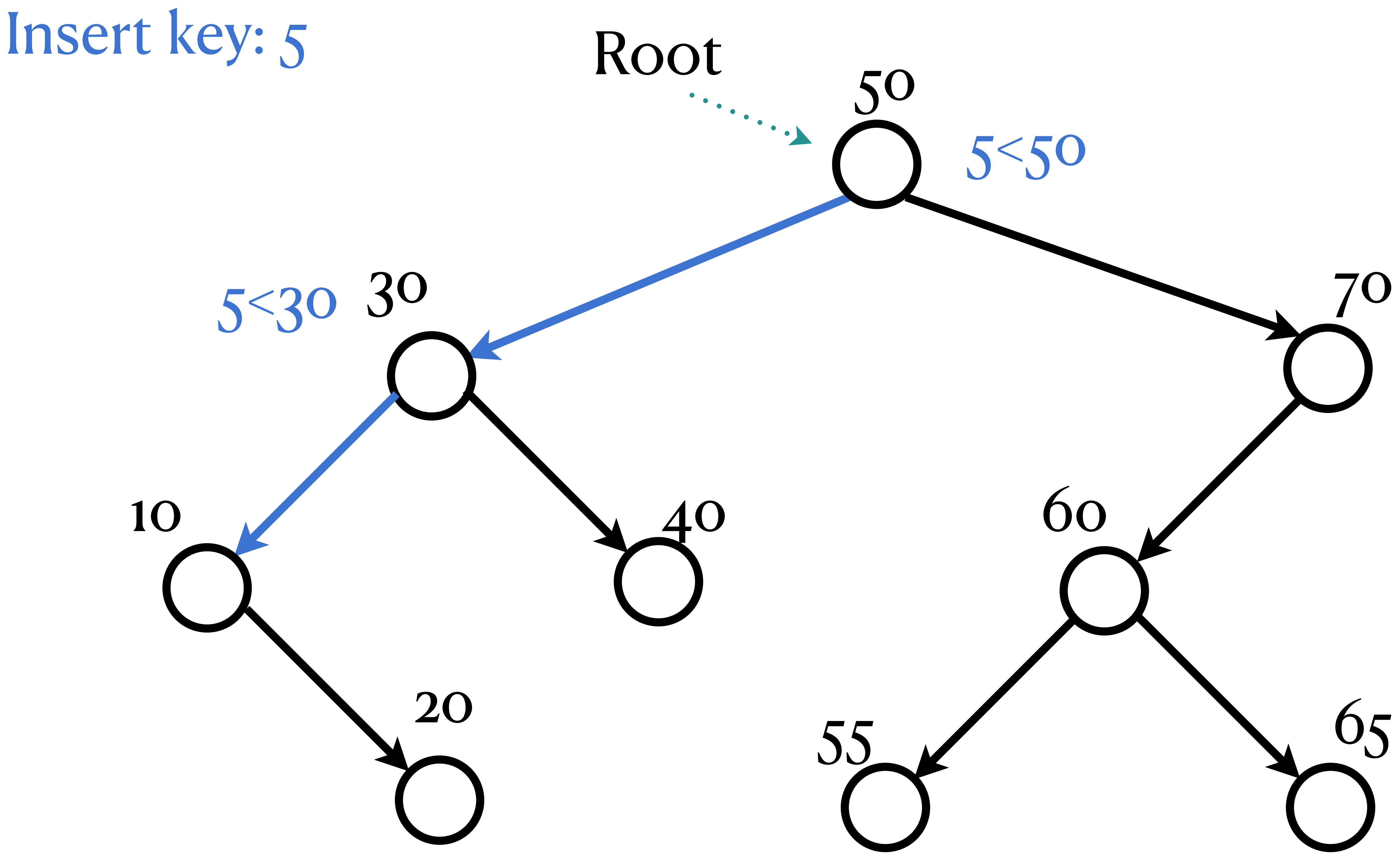
Insert key: 5



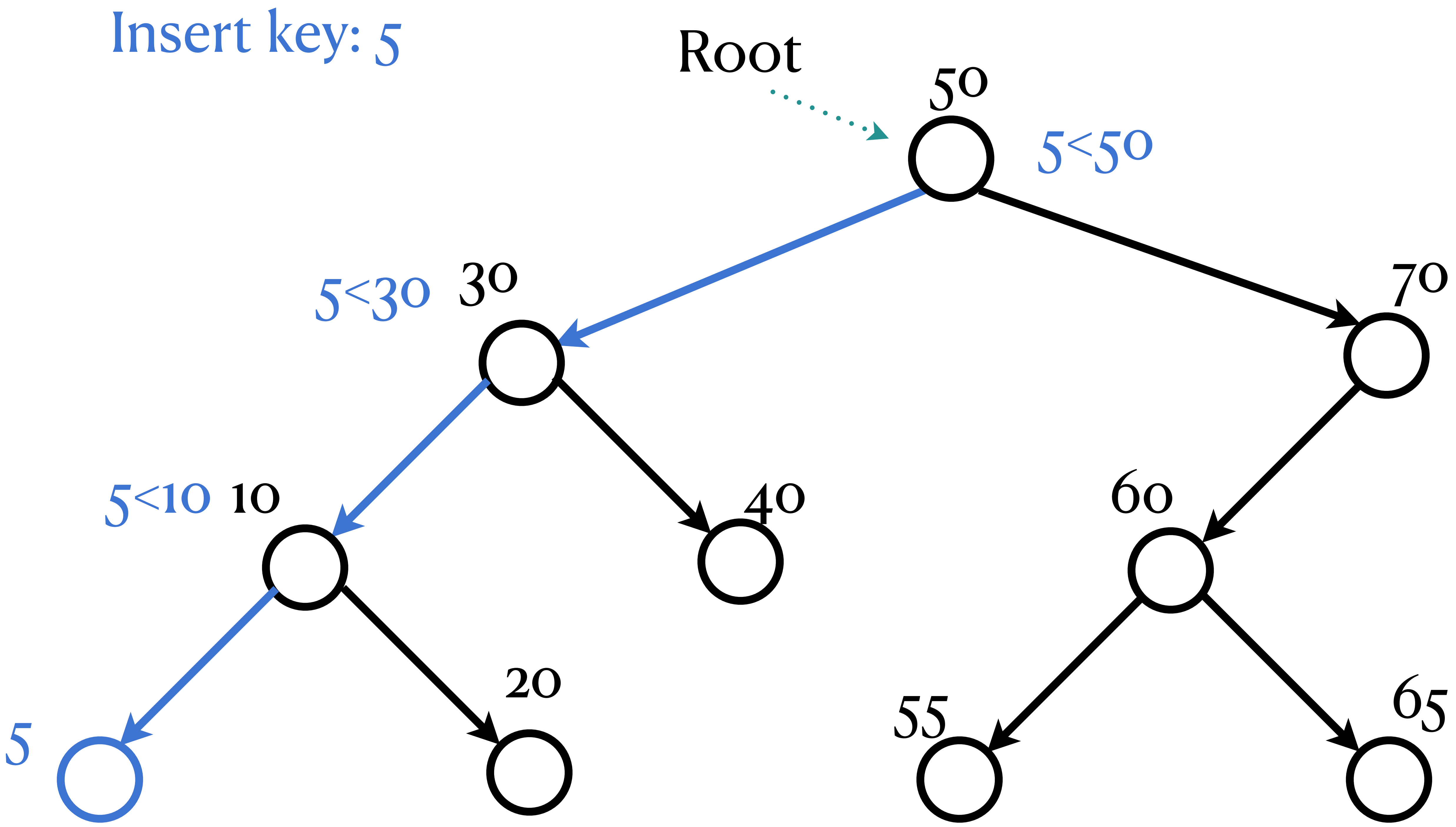
BST: Insertion



BST: Insertion

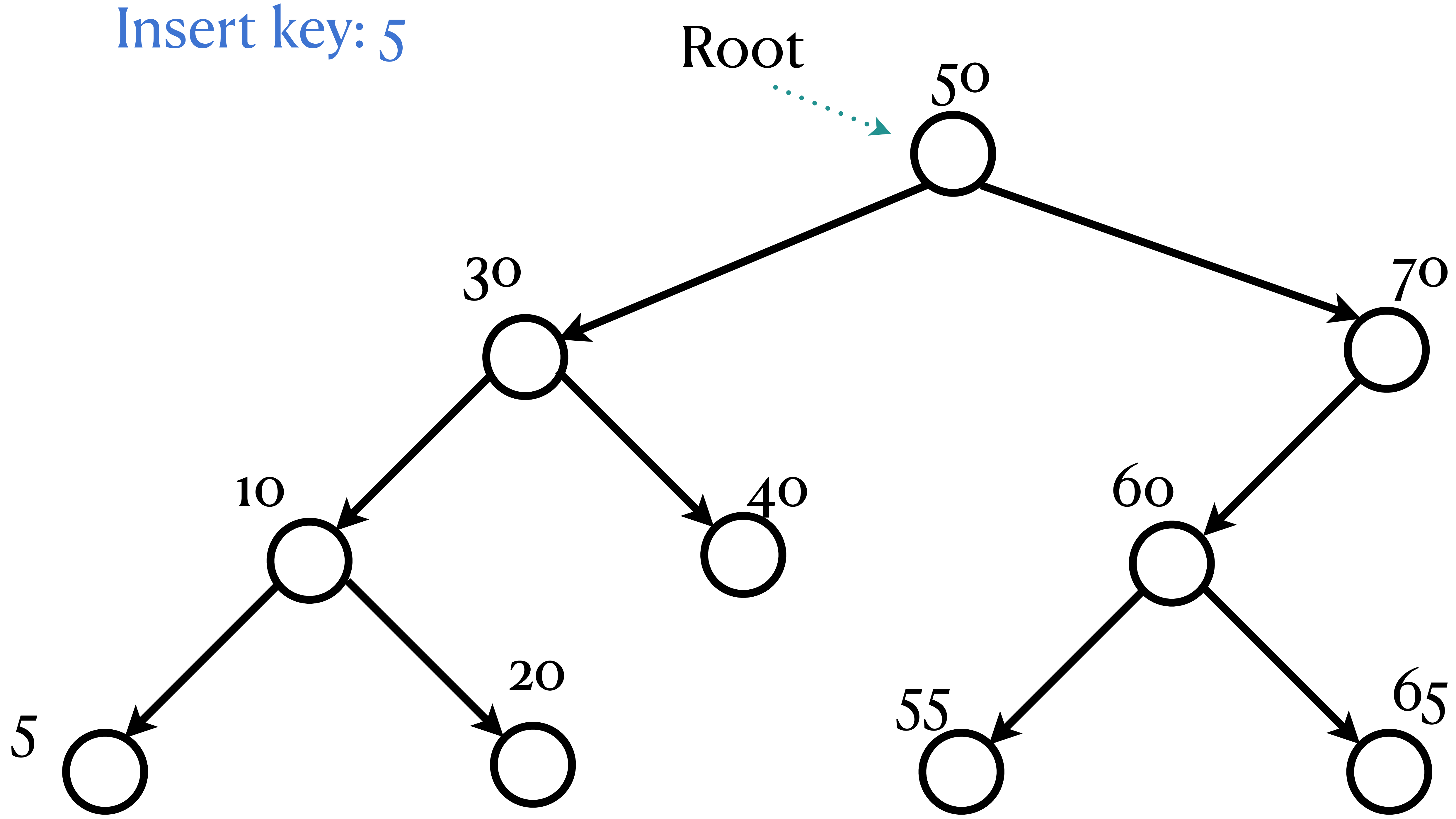


BST: Insertion



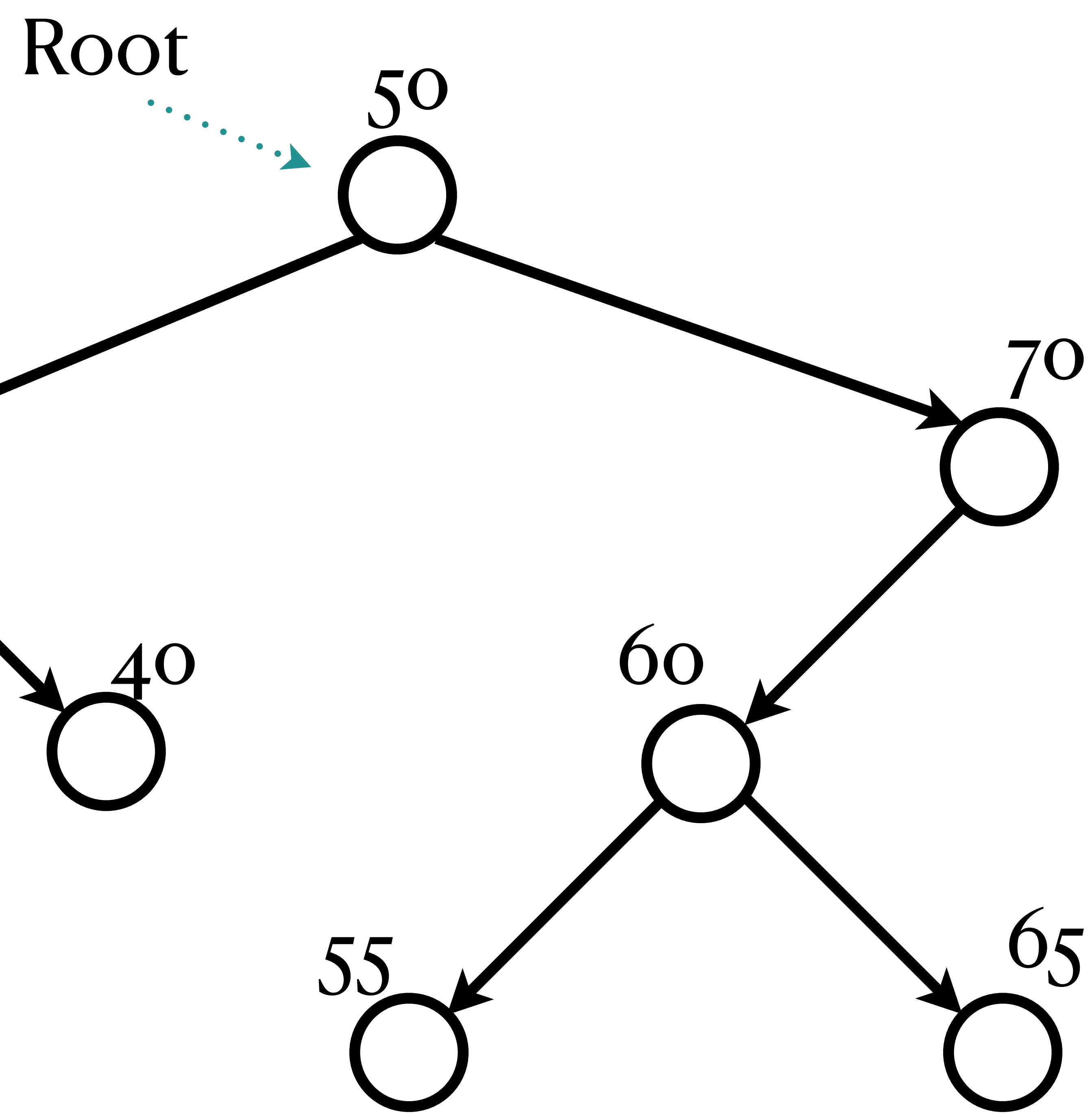
BST: Insertion

Insert key: 5

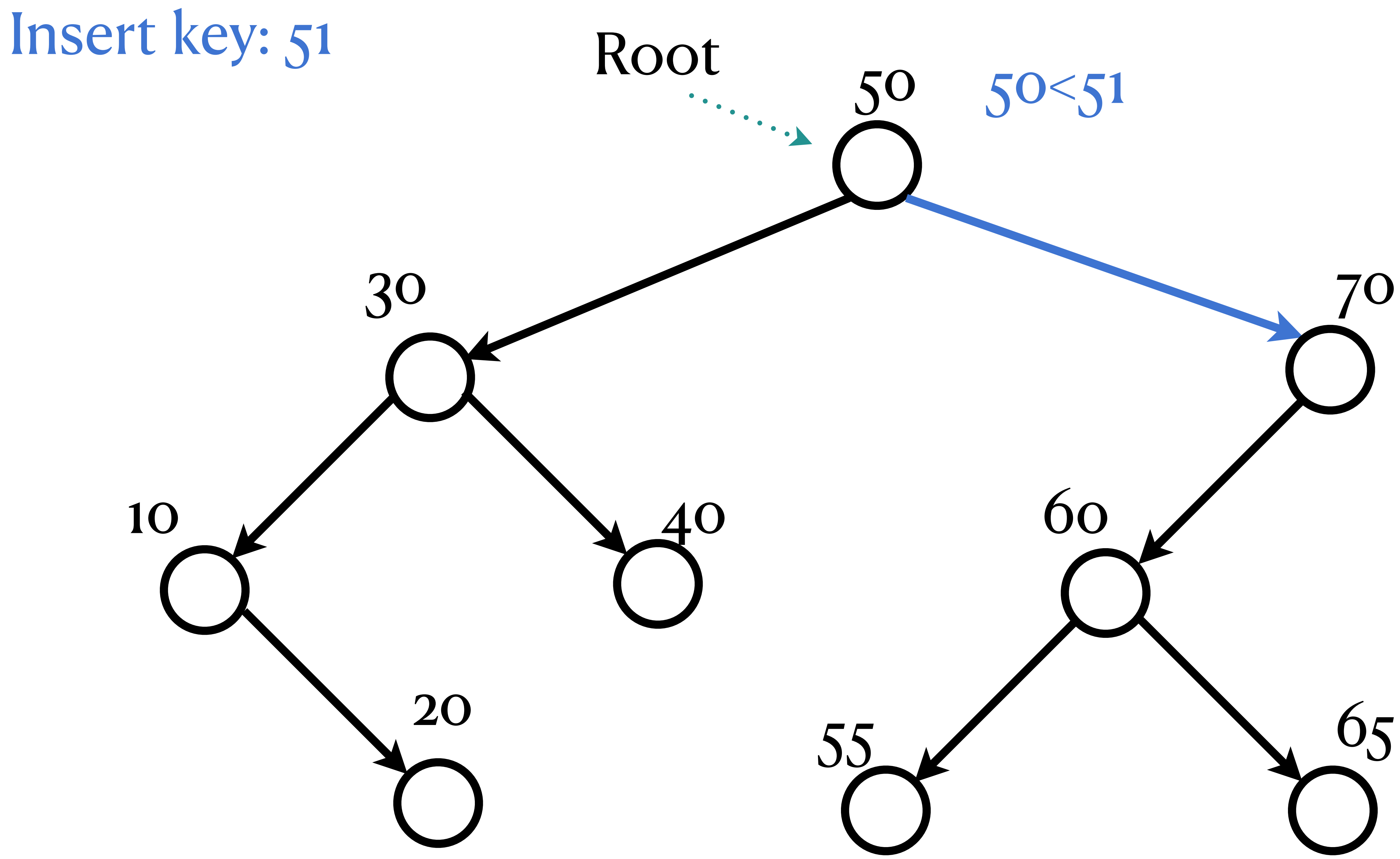


BST: Insertion

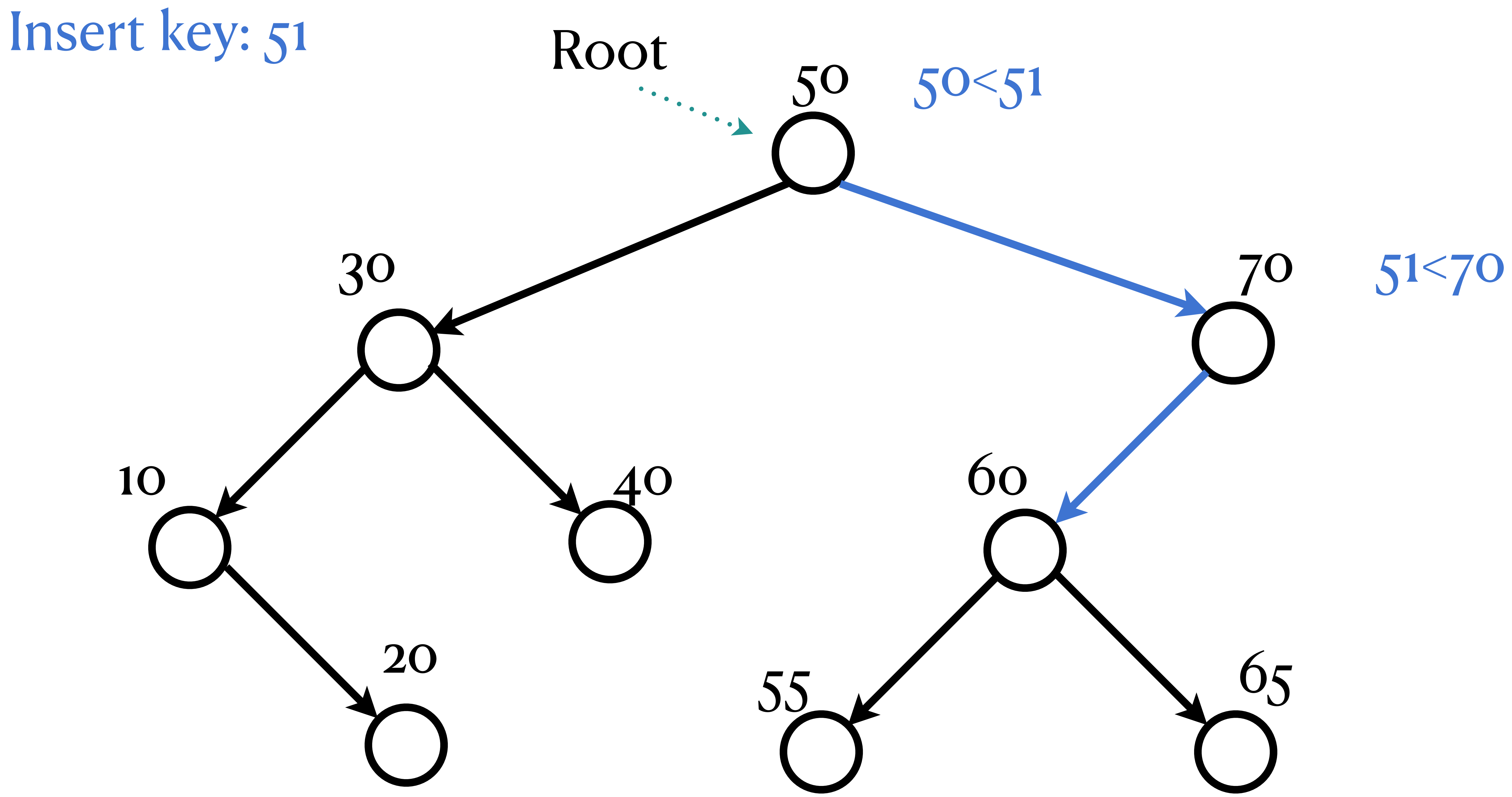
Insert key: 51



BST: Insertion

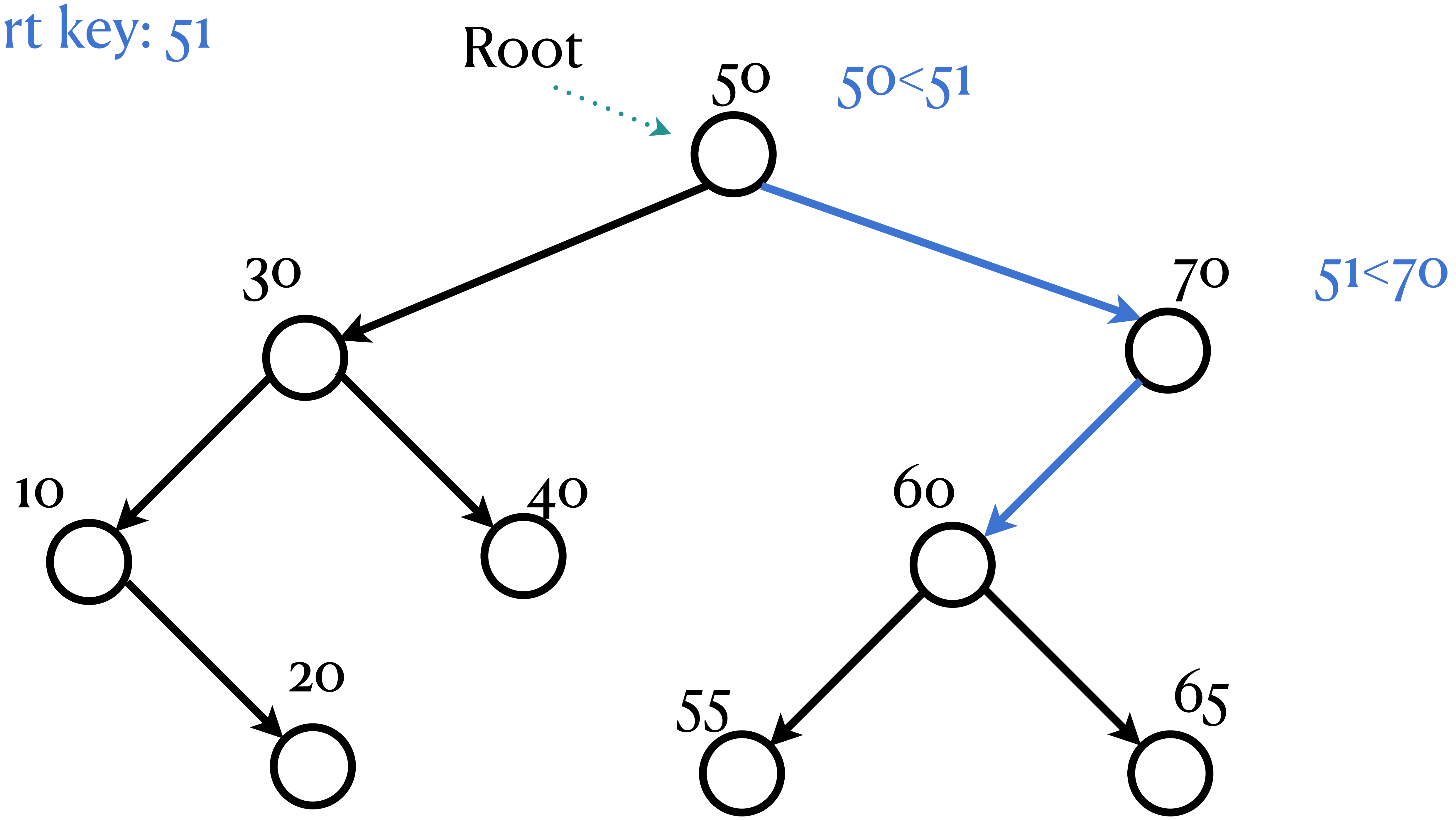


BST: Insertion

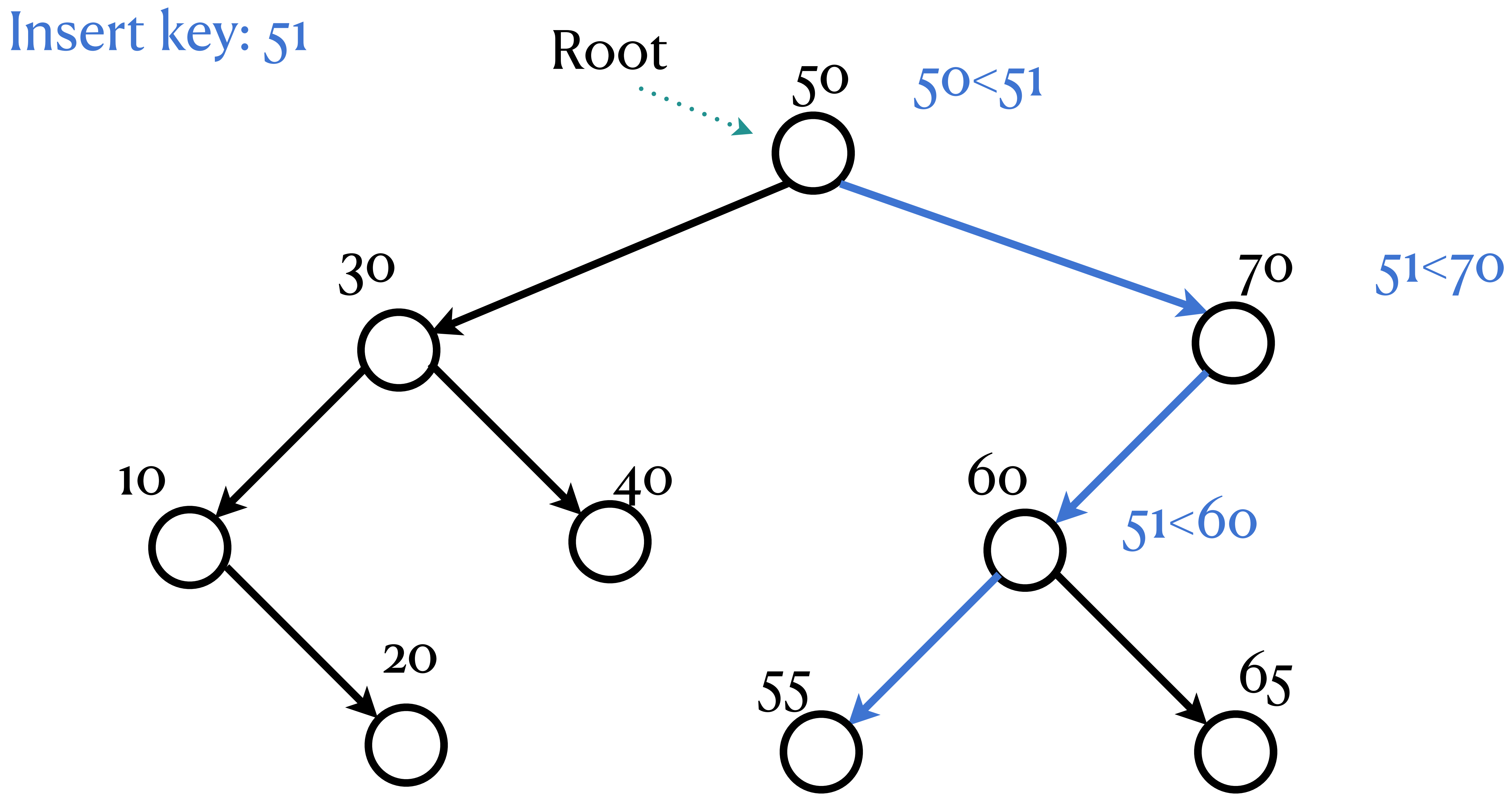


BST: Insertion

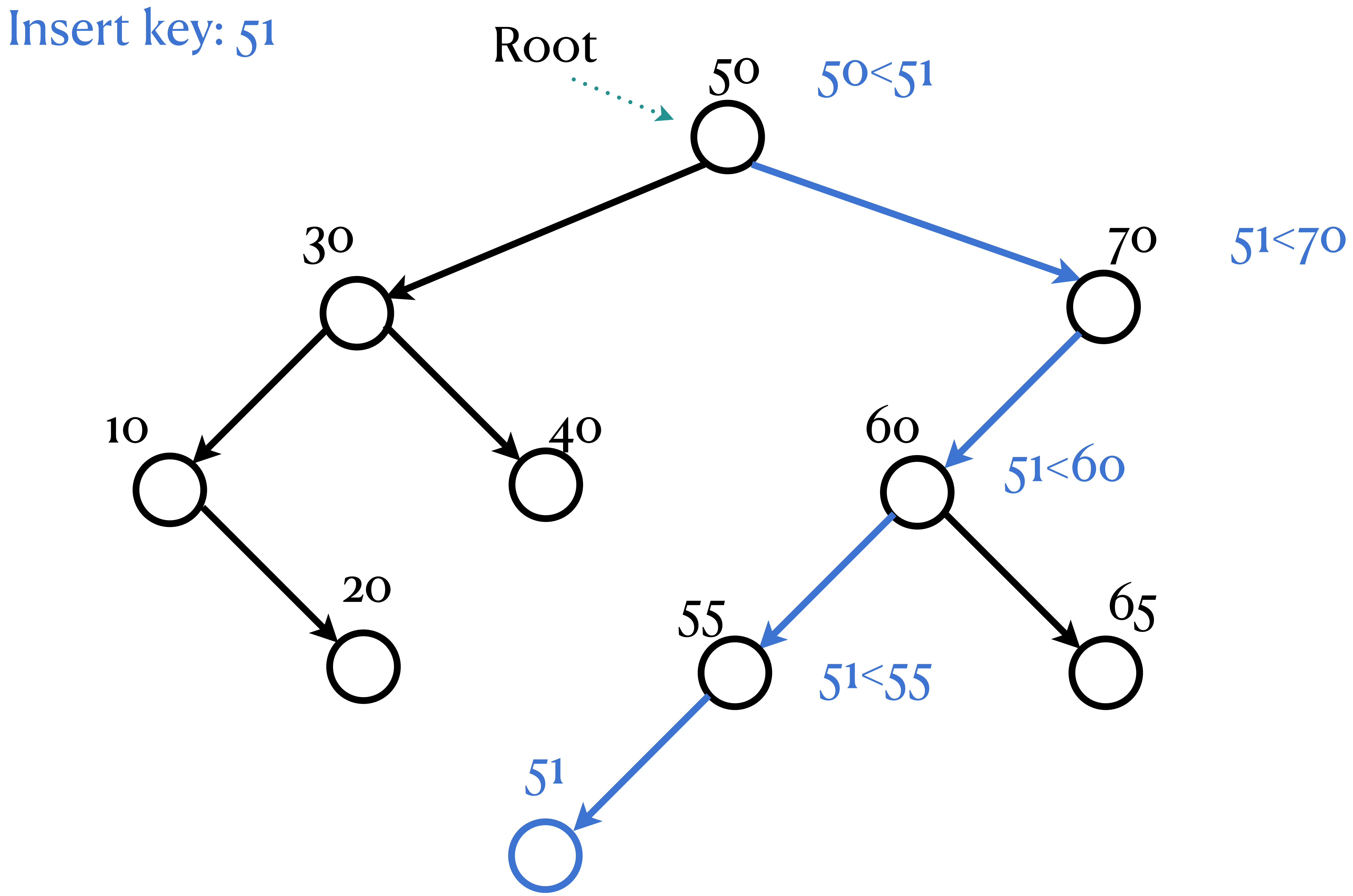
Insert key: 51



BST: Insertion

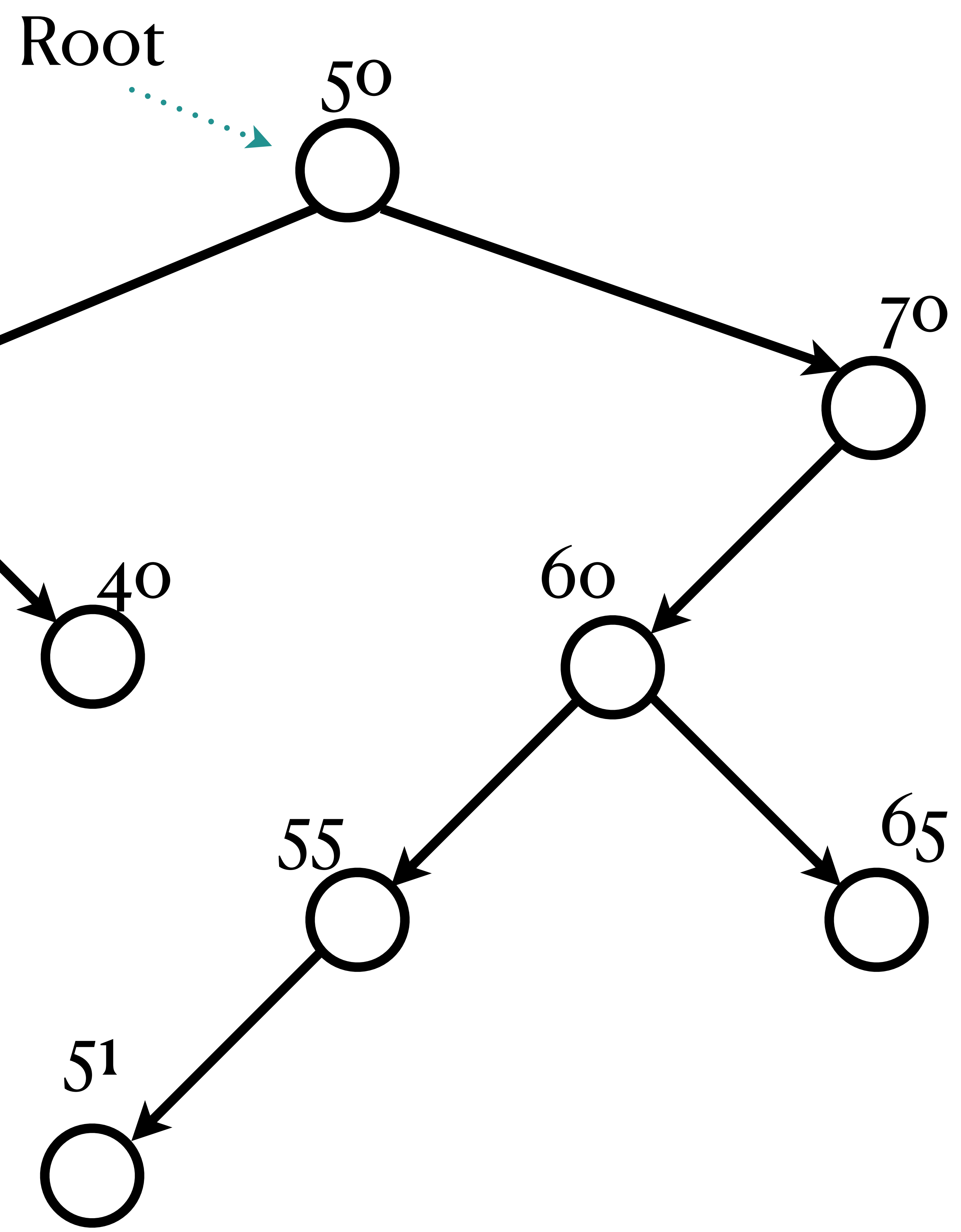


BST: Insertion

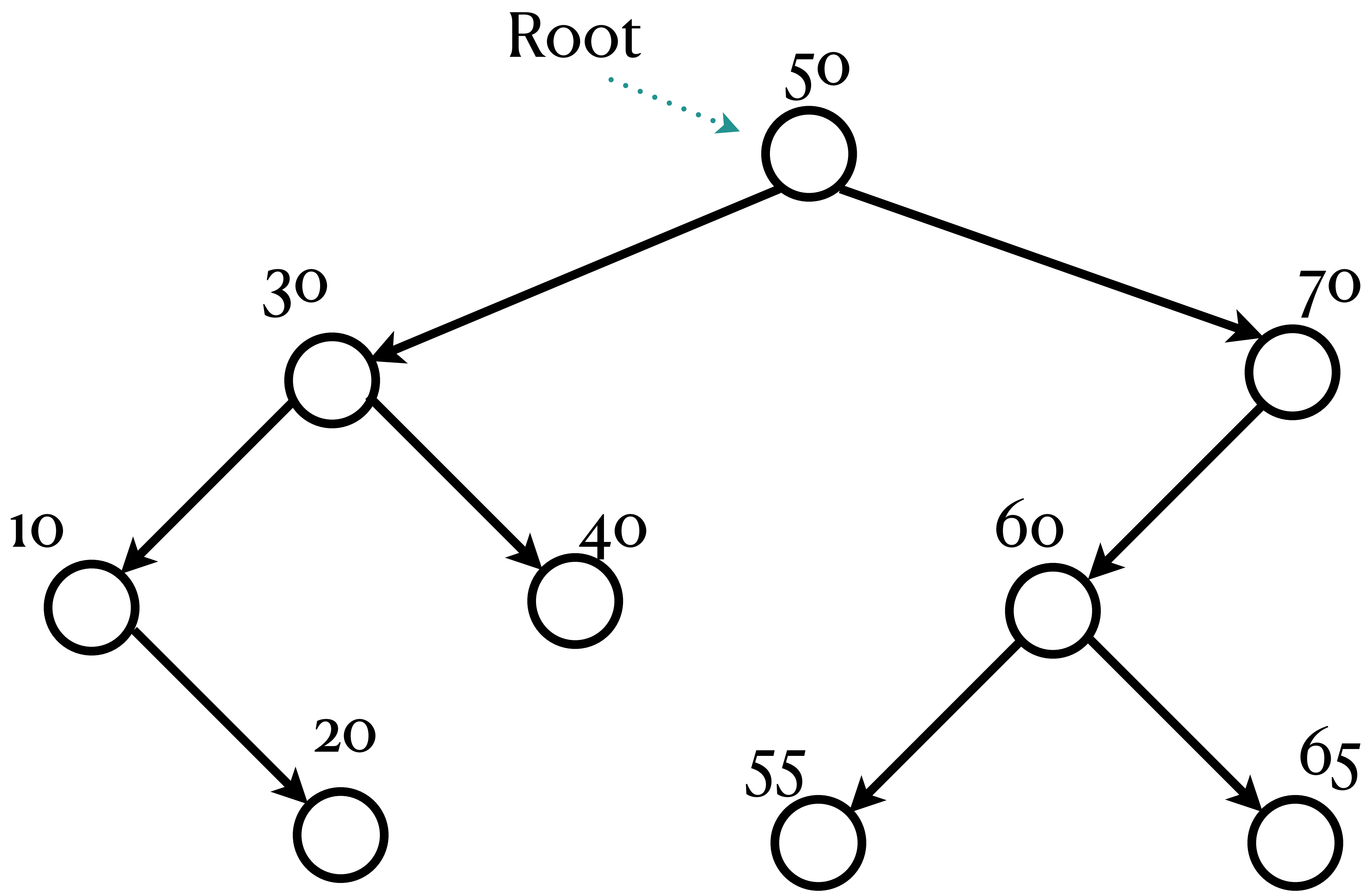


BST: Insertion

Insert key: 51

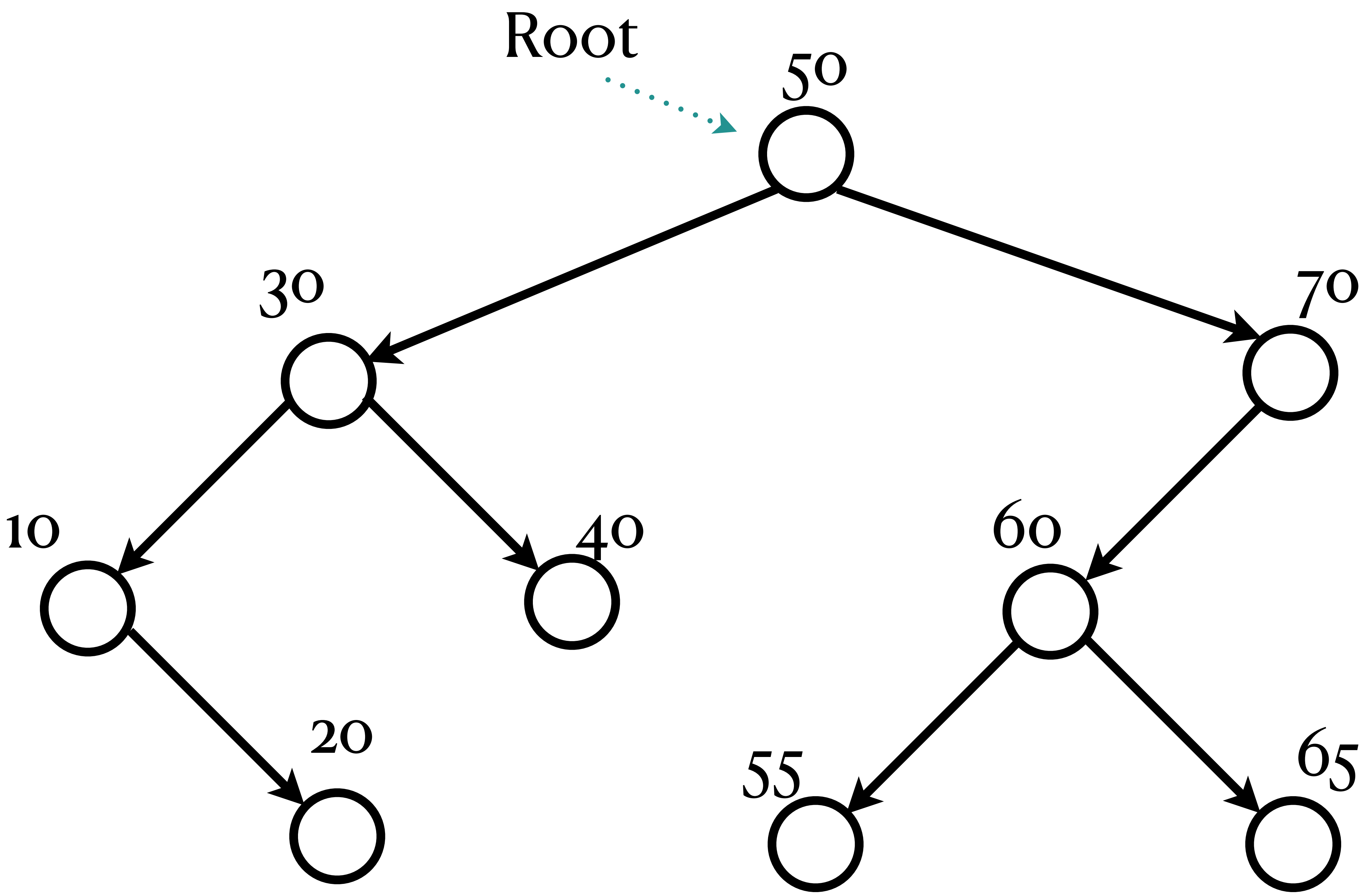


BST: Deletion



BST: Deletion

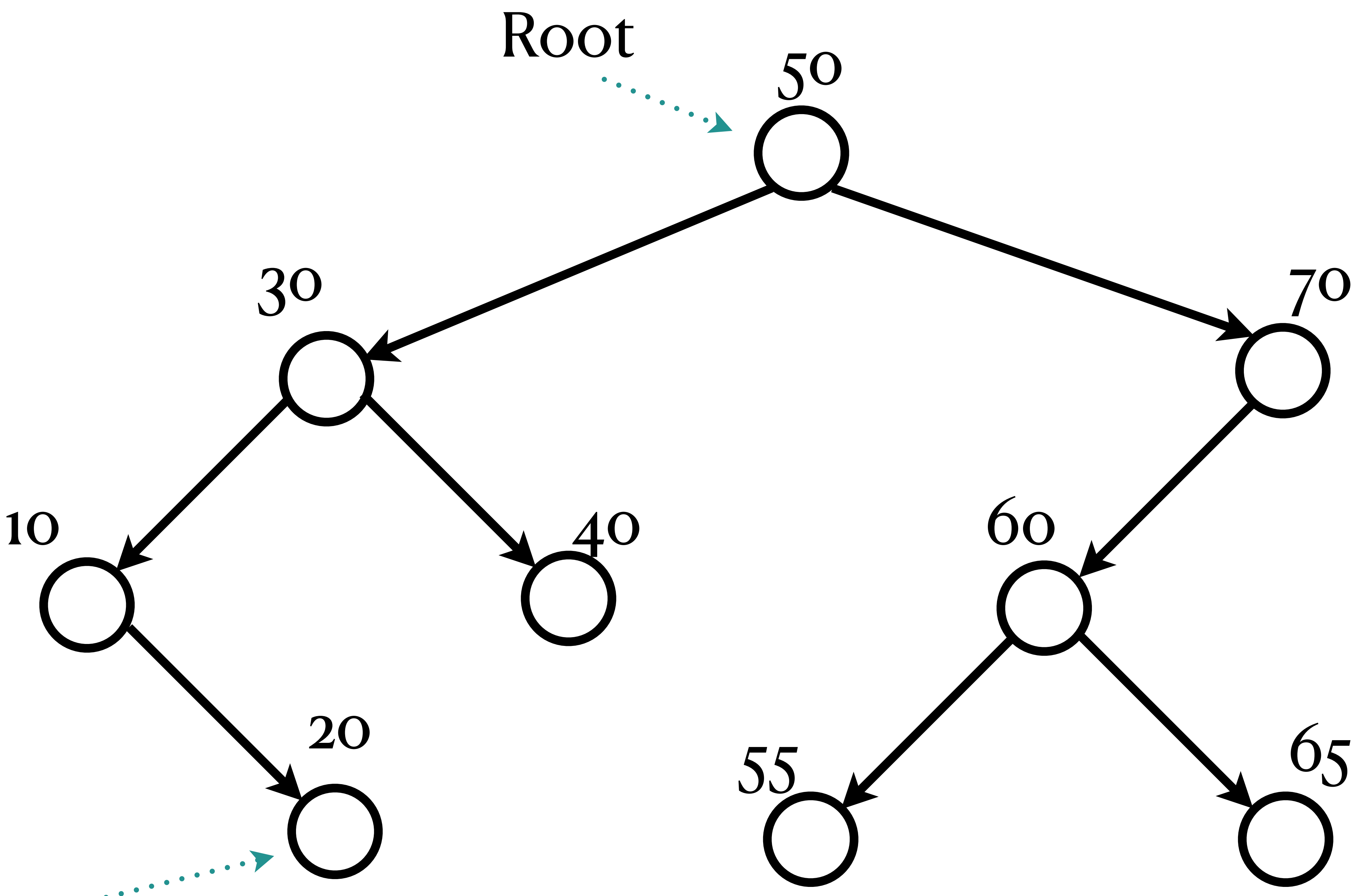
Case 1: Deleting a leaf Node



BST: Deletion

Case 1: Deleting a leaf Node

Deletion key: 20



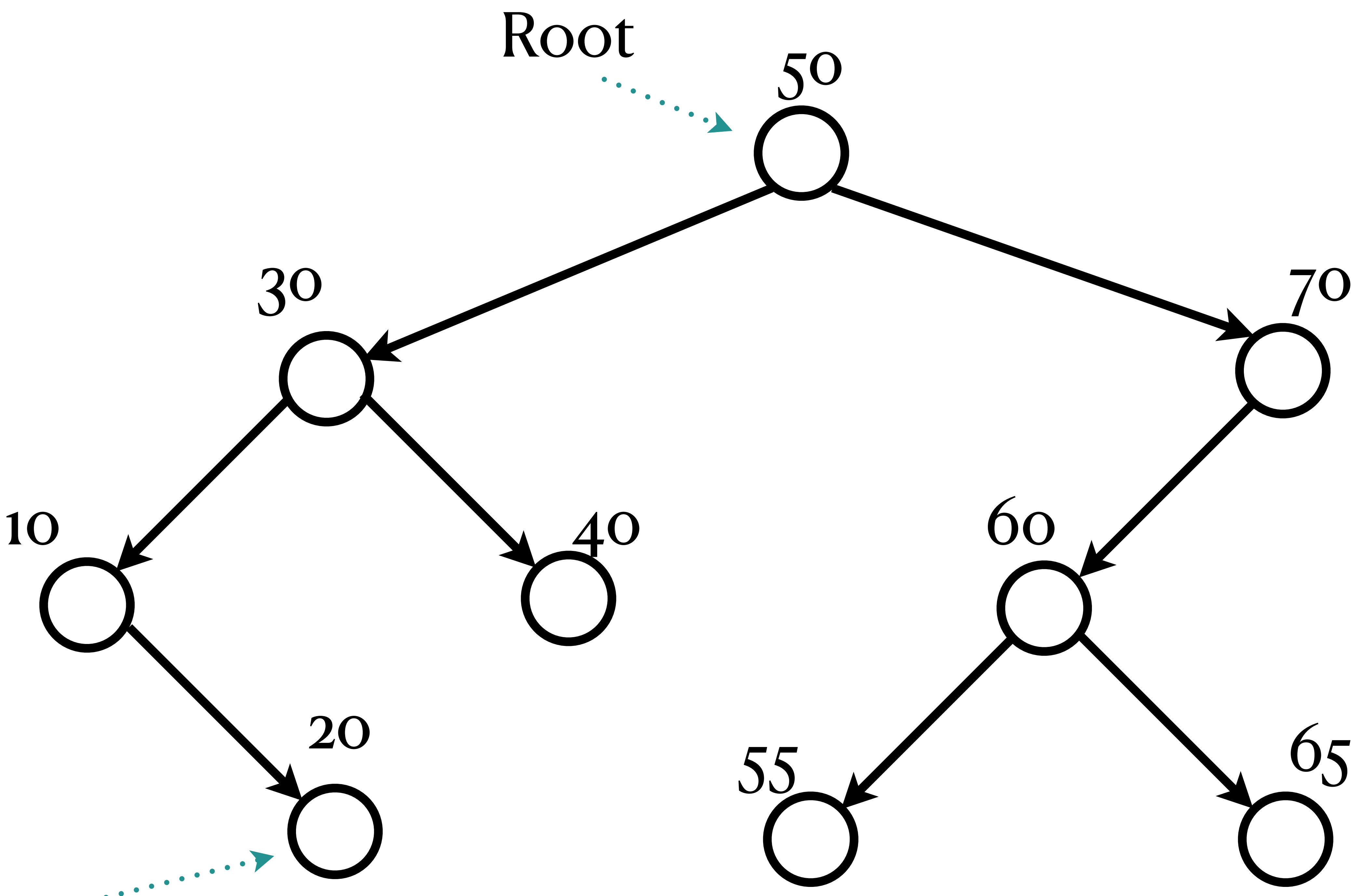
20 is a leaf node (it has no children)

BST: Deletion

Case 1: Deleting a leaf Node

Deletion key: 20

Step 1: Search for the node



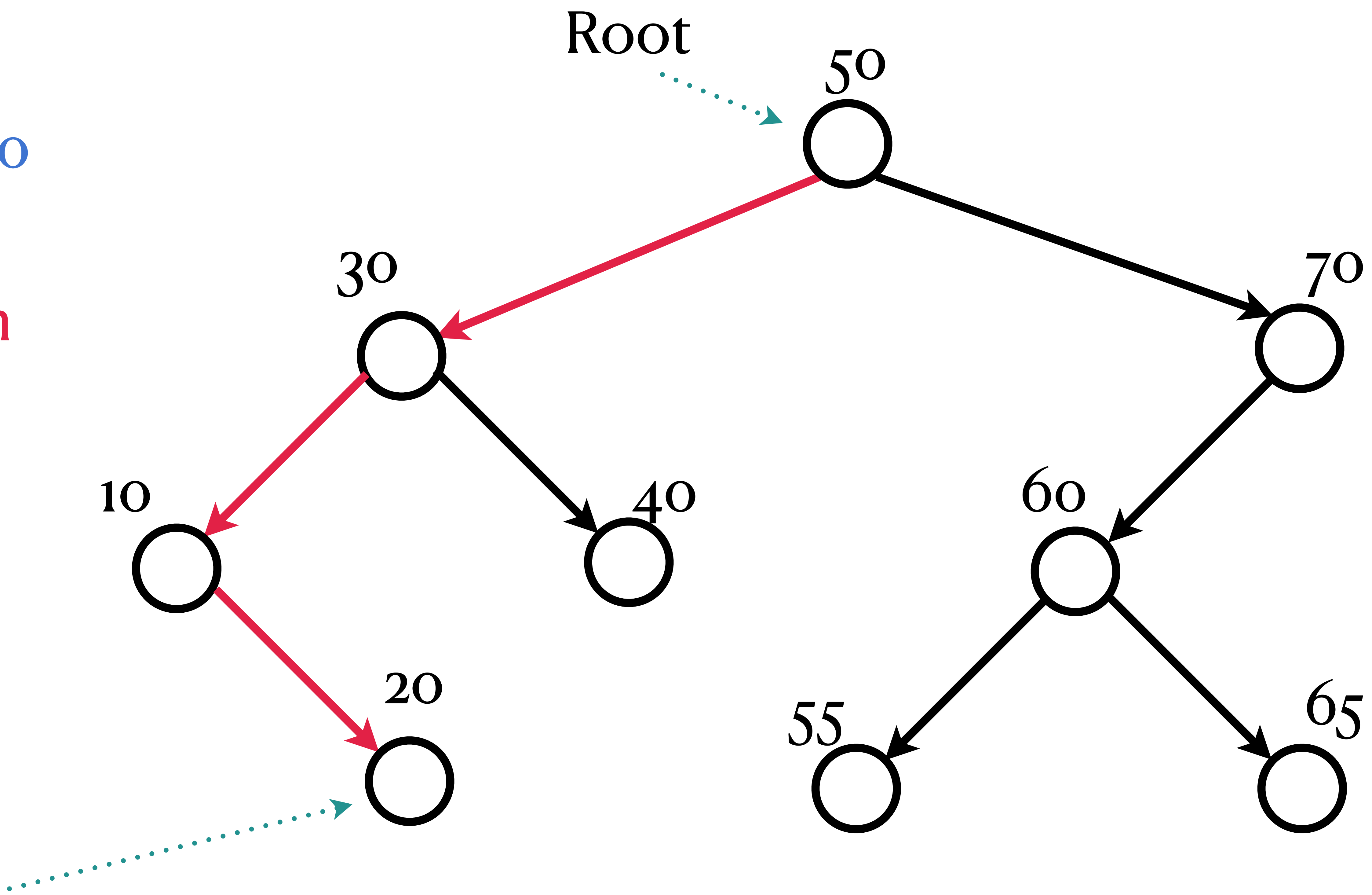
20 is a leaf node (it has no children)

BST: Deletion

Case 1: Deleting a leaf Node

Deletion key: 20

Step 1: Search for the node



20 is a leaf node (it has no children)

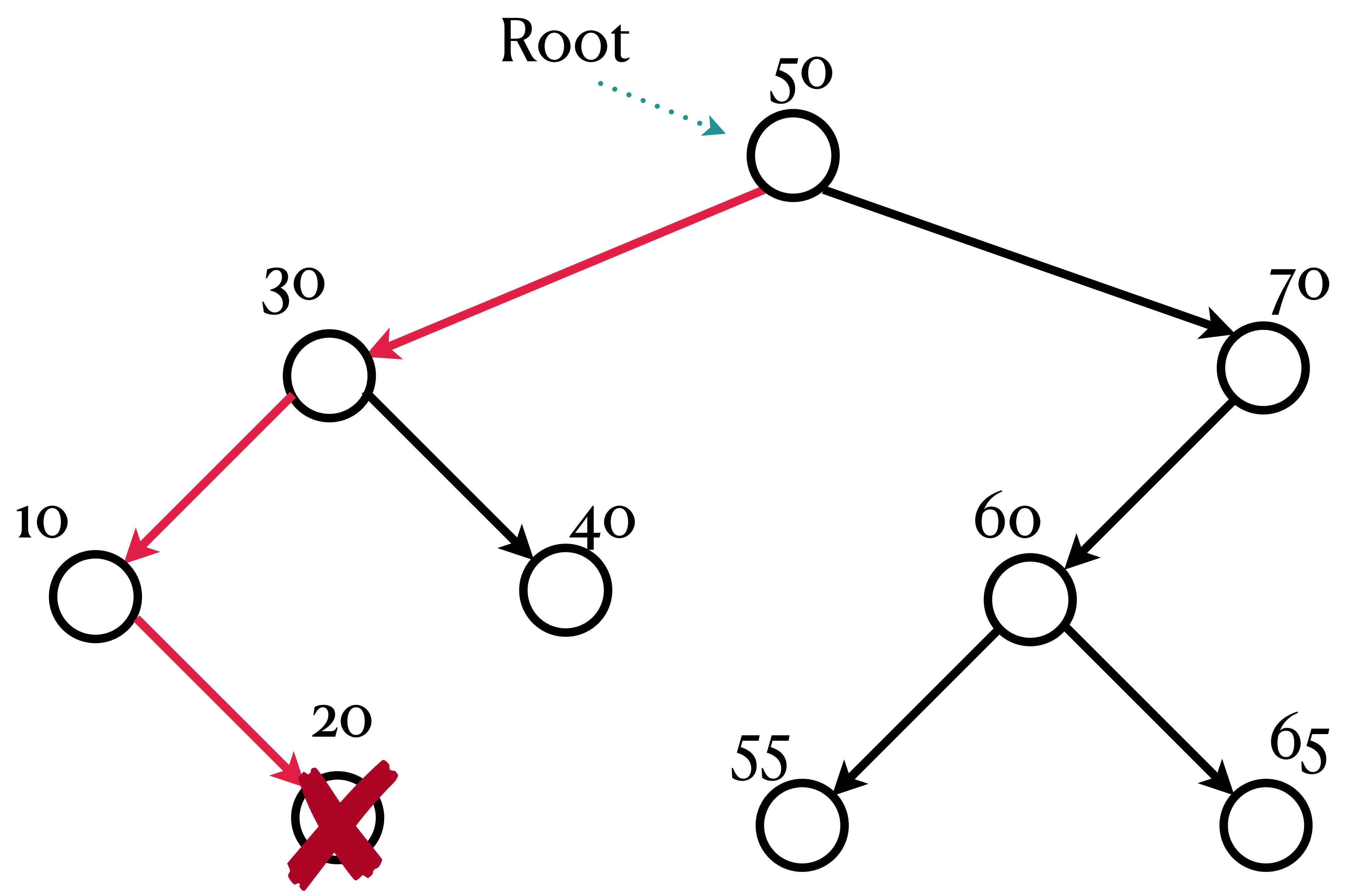
BST: Deletion

Case 1: Deleting a leaf Node

Deletion key: 20

Step 1: Search for the node

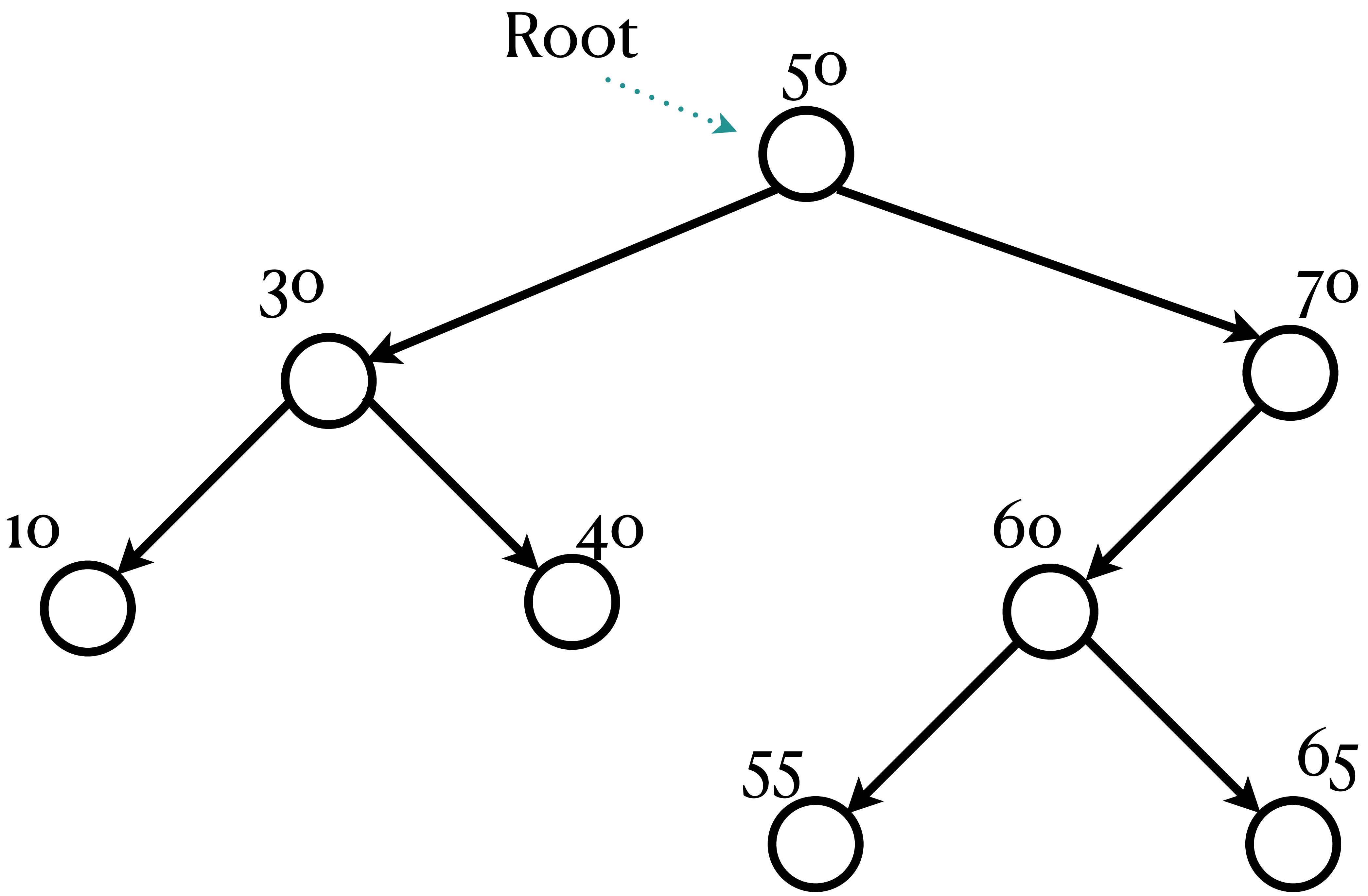
Step 2: Remove the node and update the parent



BST: Deletion

Case 1: Deleting a leaf Node

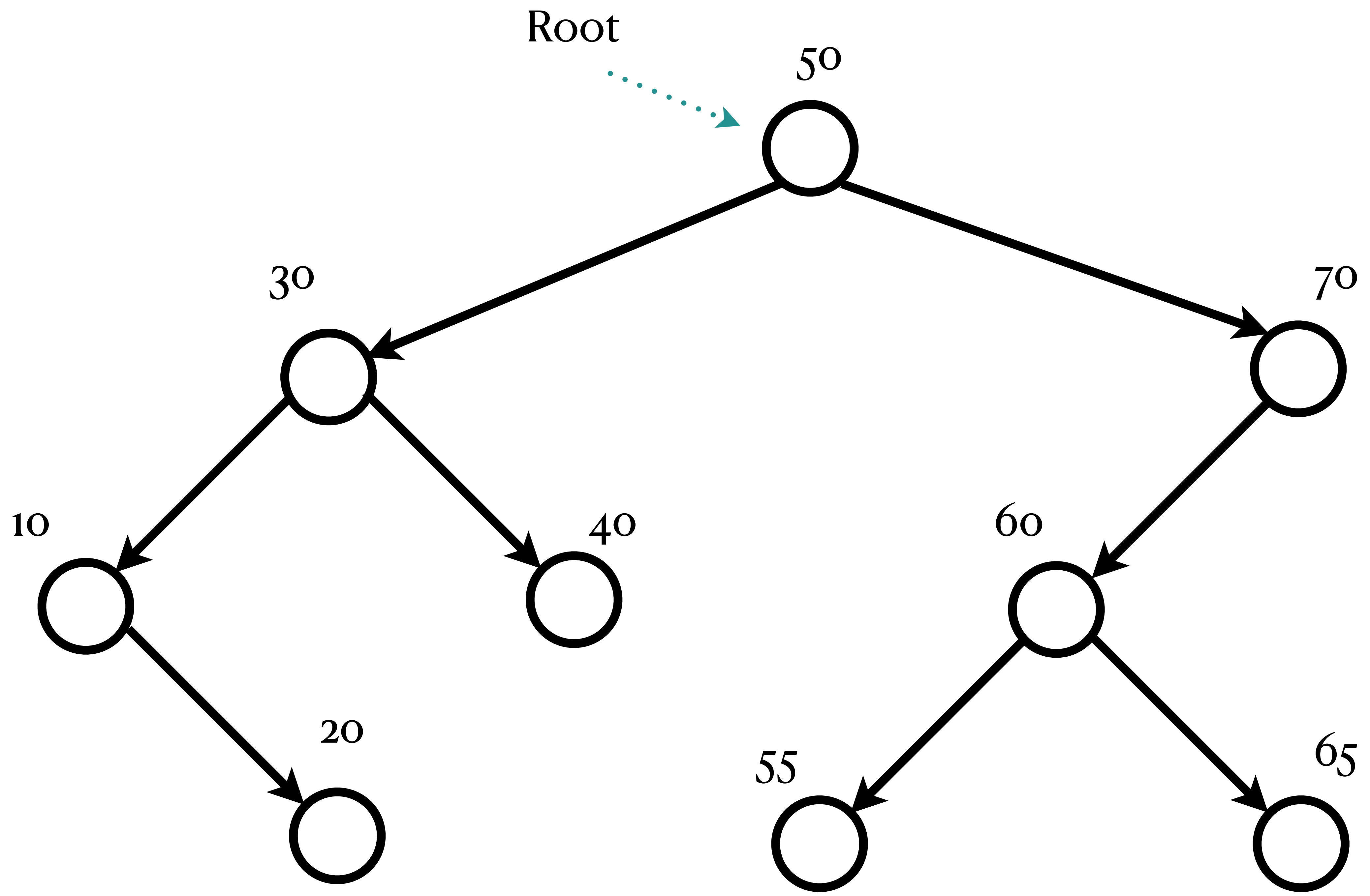
Deletion key: 20



BST: Deletion

Case 2: Deleting a node with exactly one child

Deletion key: 10

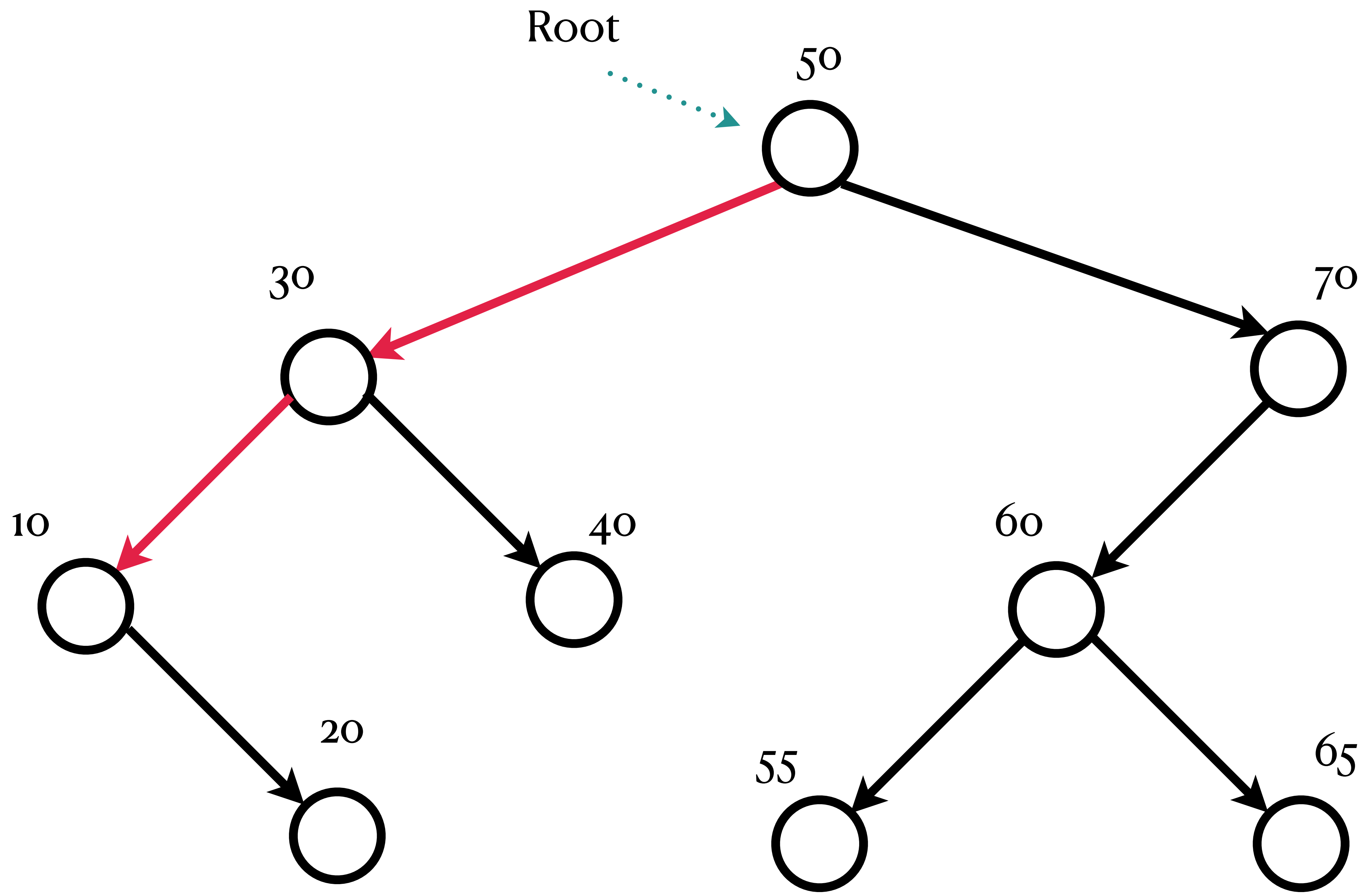


BST: Deletion

Case 2: Deleting a node with exactly one child

Deletion key: 10

Step 1: Search for the node

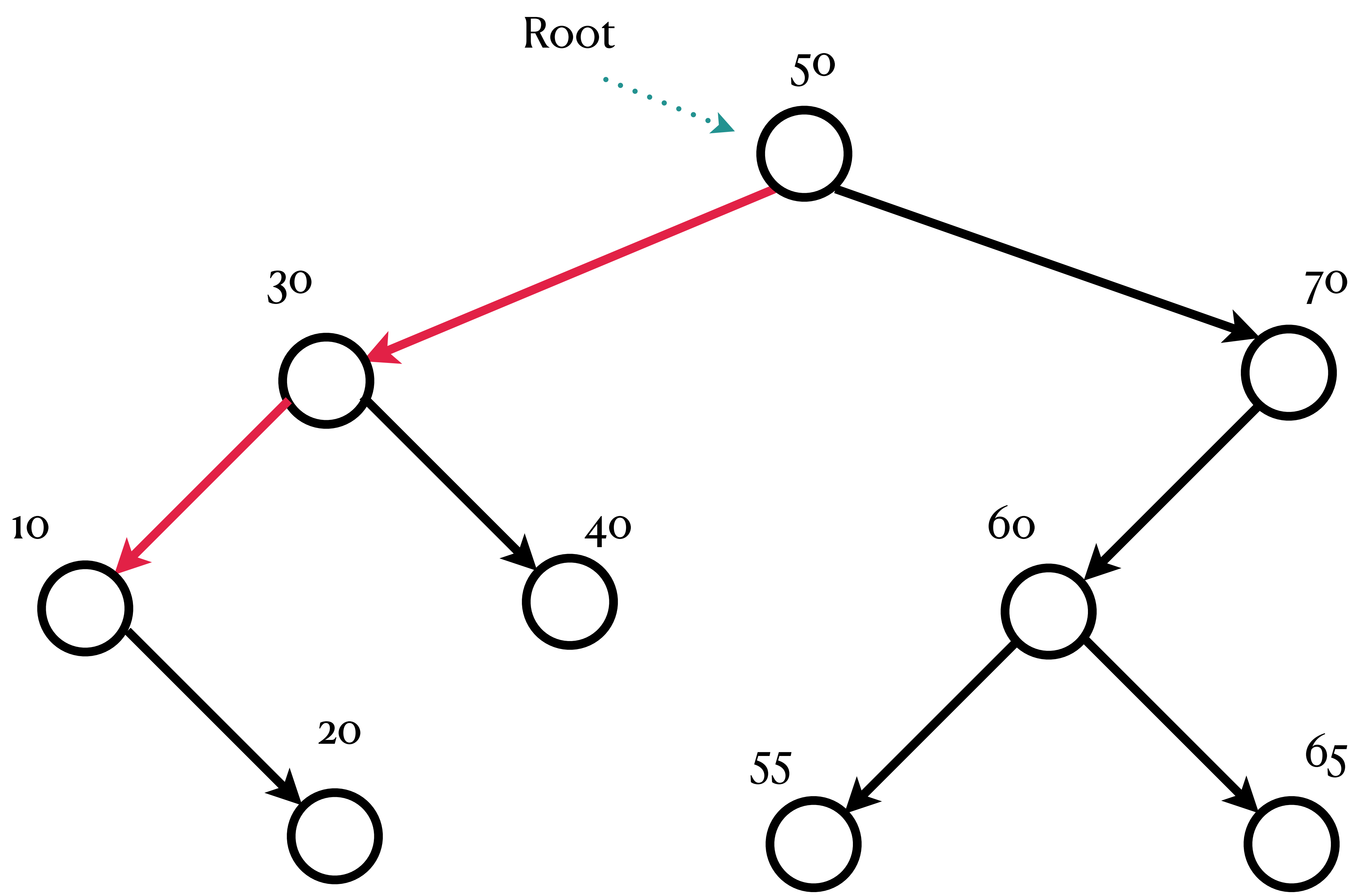


BST: Deletion

Case 2: Deleting a node with exactly one child

Deletion key: 10

Step 1: Search for the node



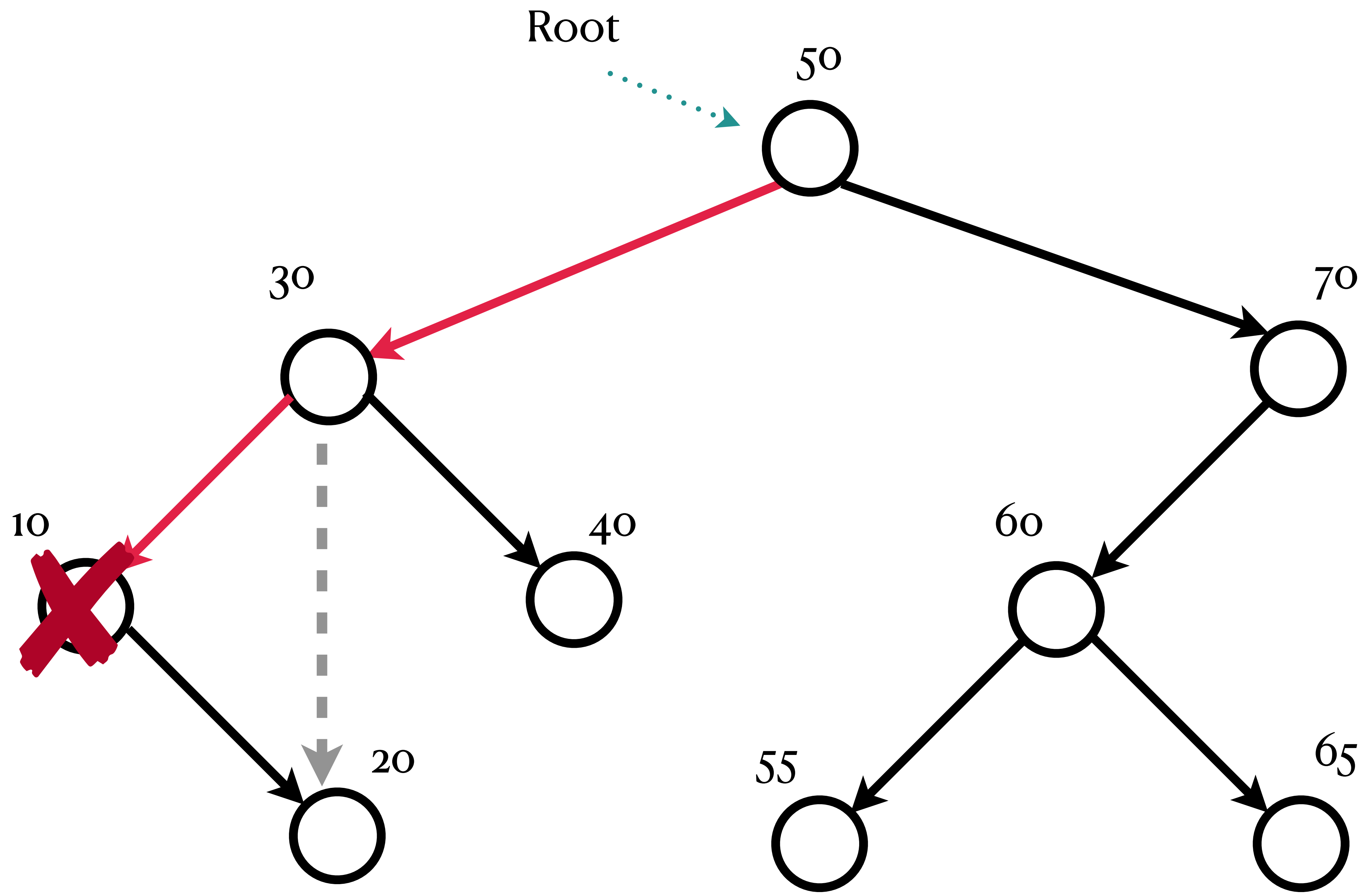
BST: Deletion

Case 2: Deleting a node with exactly one child

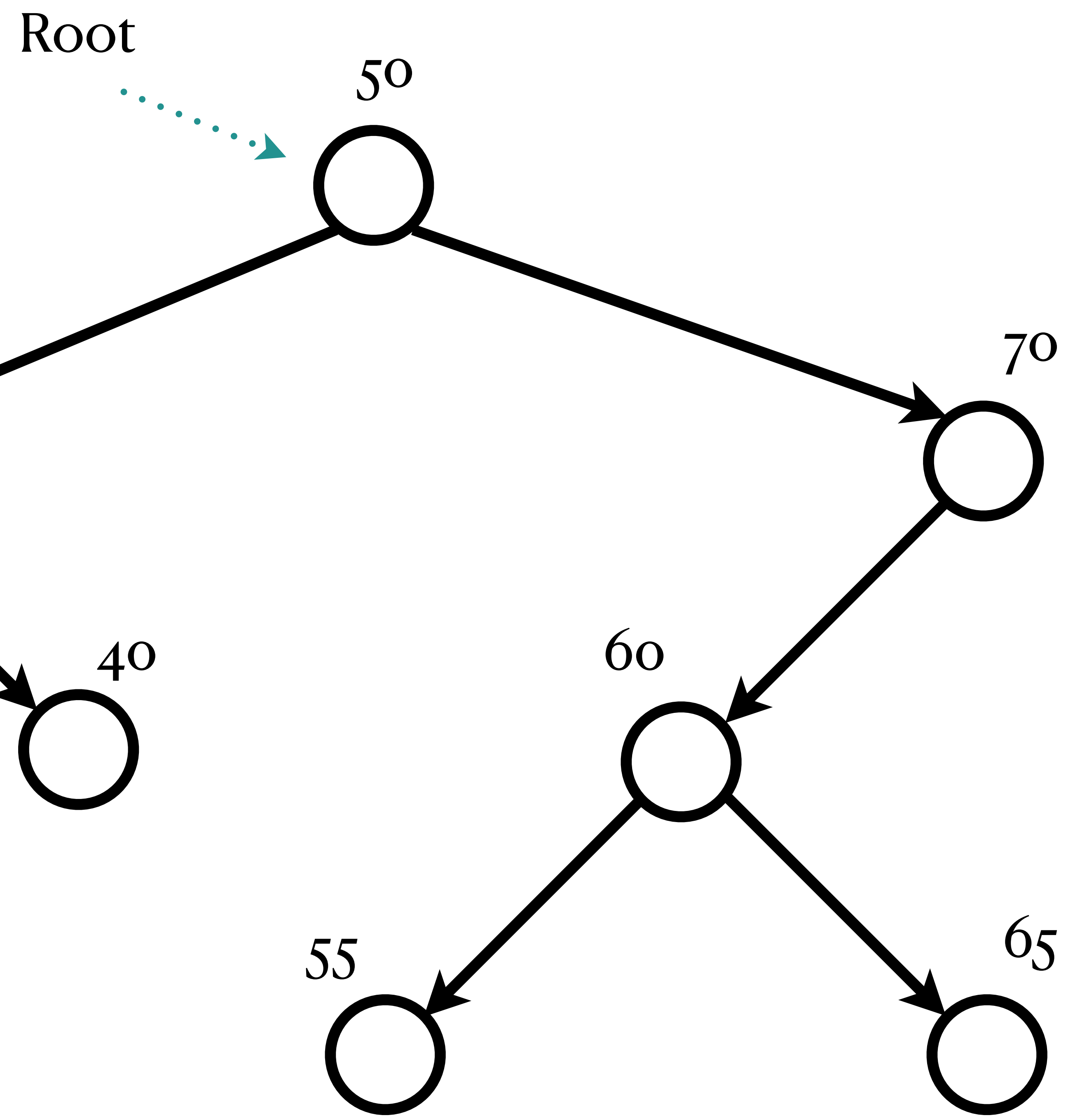
Deletion key: 10

Step 1: Search for the node

Step 2: Remove the node and update the parent



BST: Deletion

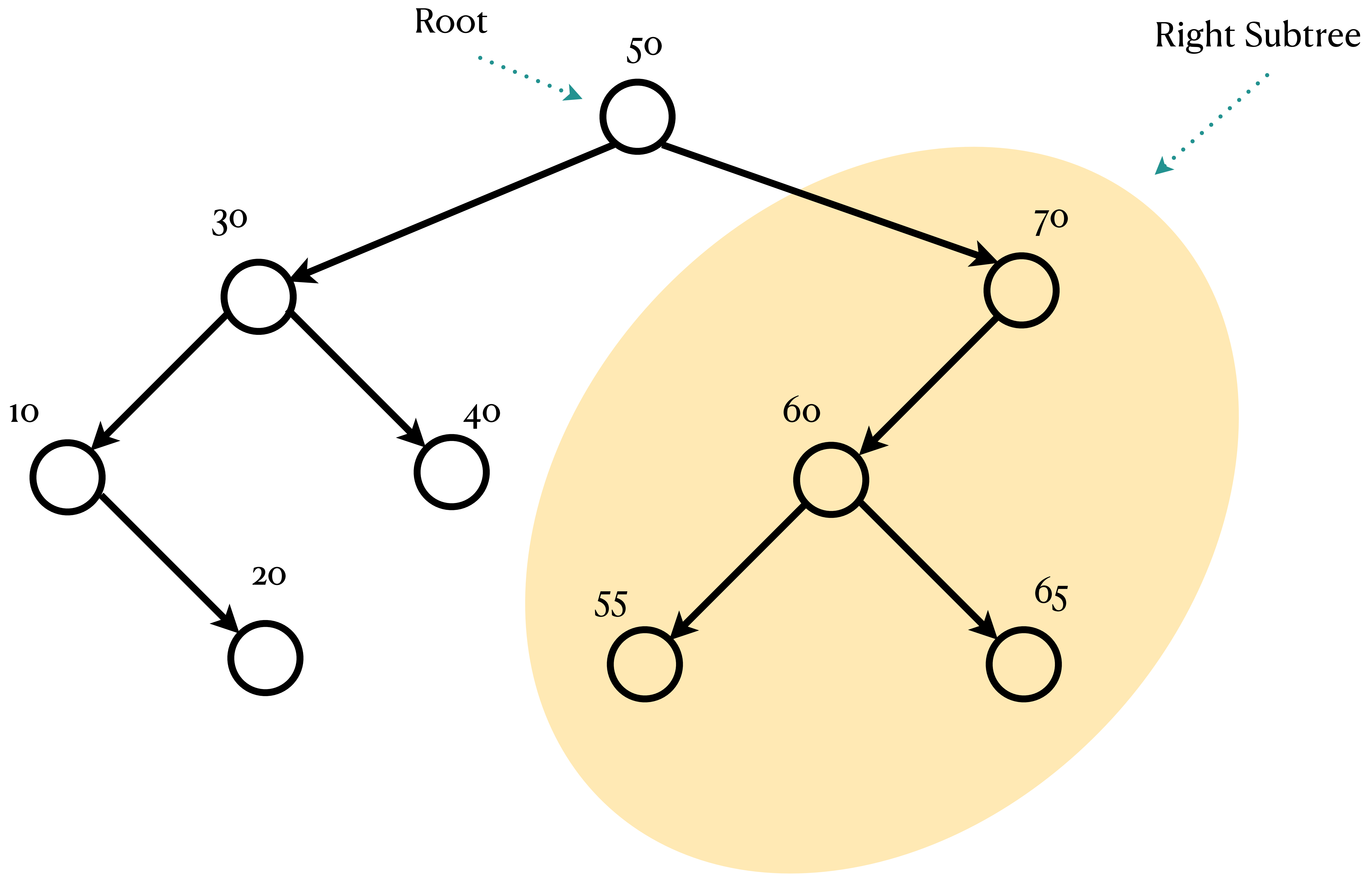


BST: Deletion

Case 3: Deleting a Node with two children

Deletion key: 50

Option 1: Find the smallest element in right subtree



BST: Deletion

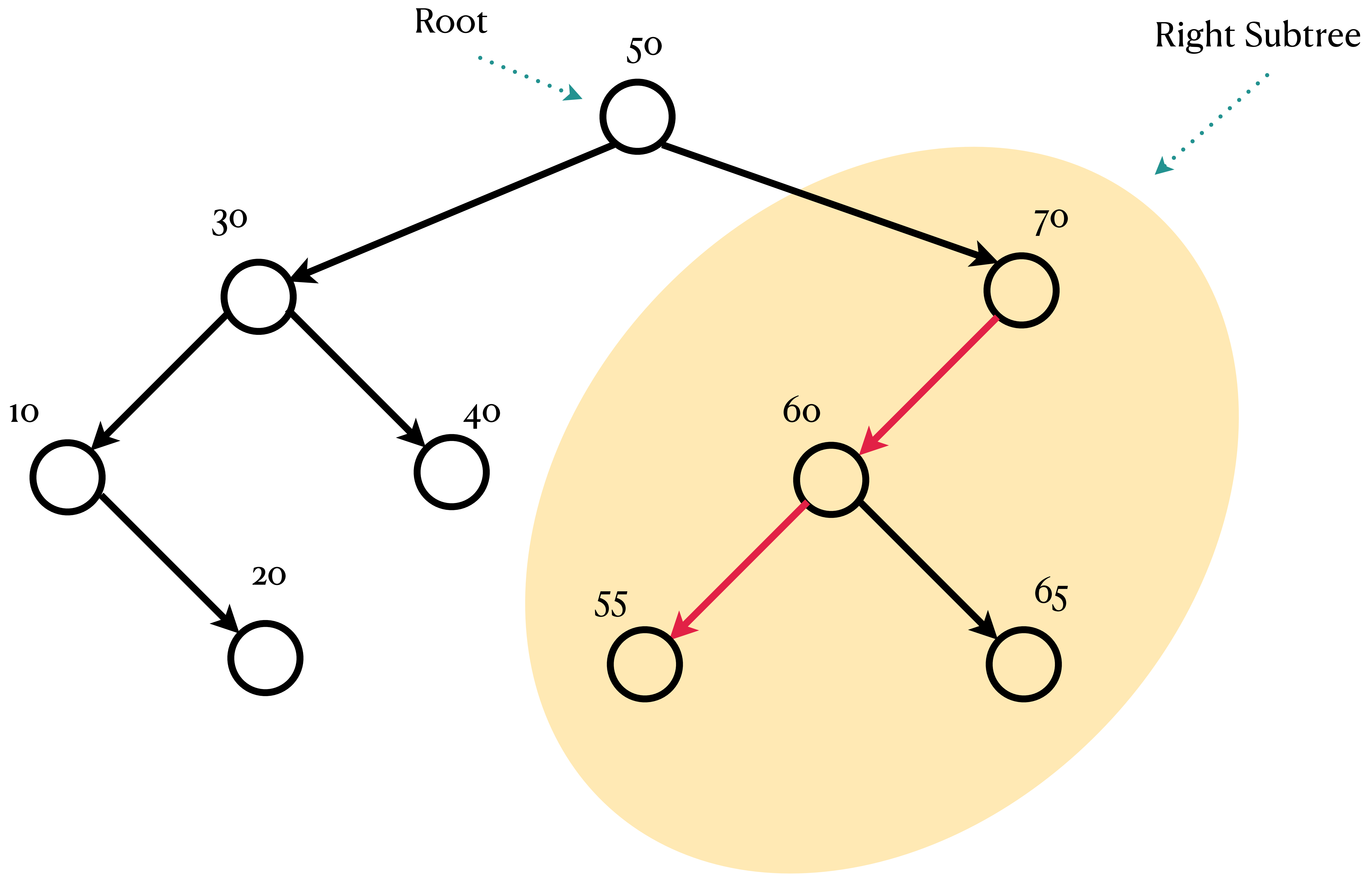
Case 3: Deleting a Node with two children

Deletion key: 50

Option 1: Find the smallest element in right subtree

How? Start from root of the right subtree and keep moving left

Smallest node in the right subtree has at most one children (Why?)



BST: Deletion

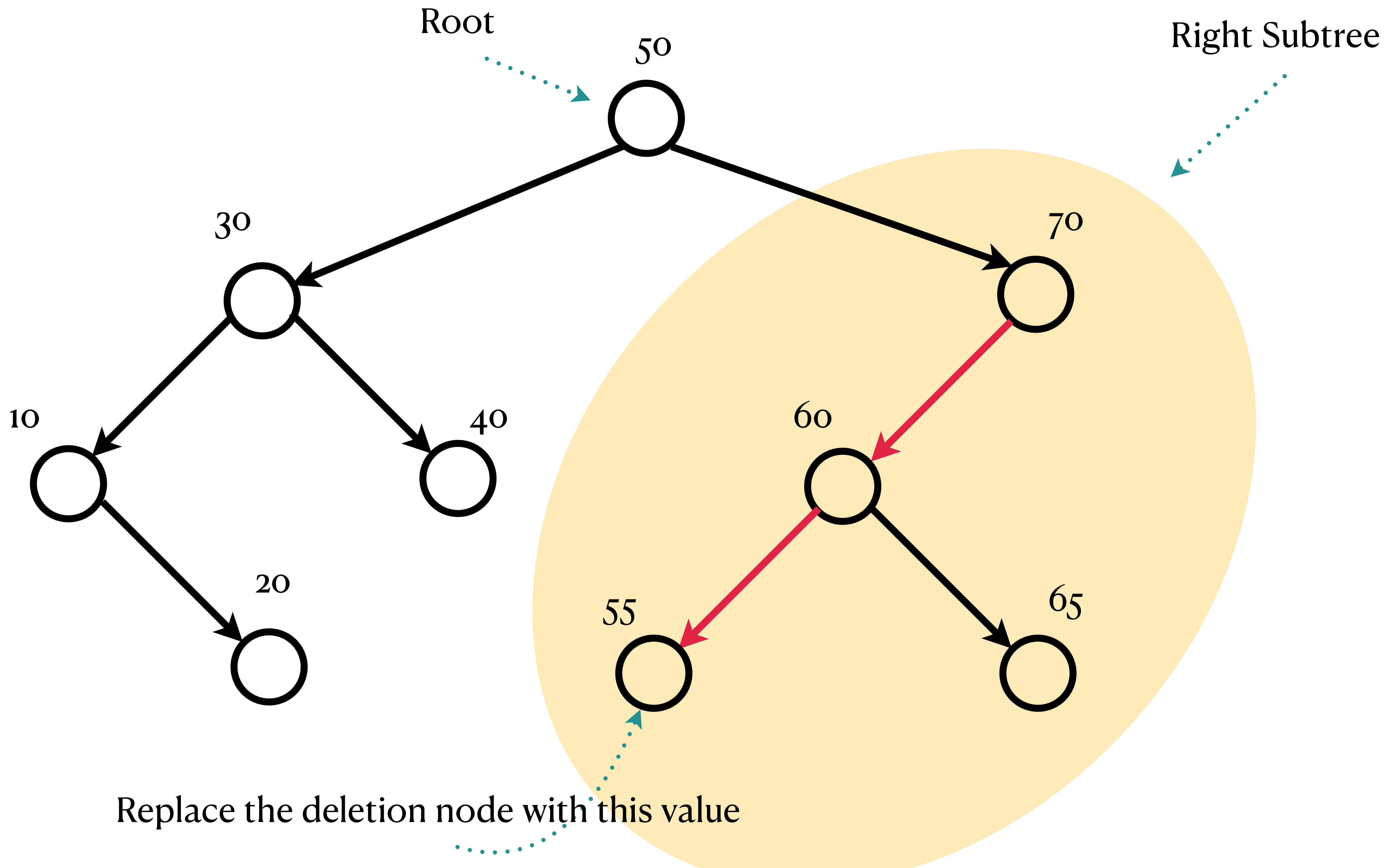
Case 3: Deleting a Node with two children

Deletion key: 50

Option 1: Find the smallest element in right subtree

How? Start from root of the right subtree and keep moving left

Smallest node in the right subtree has at most one children (Why?)



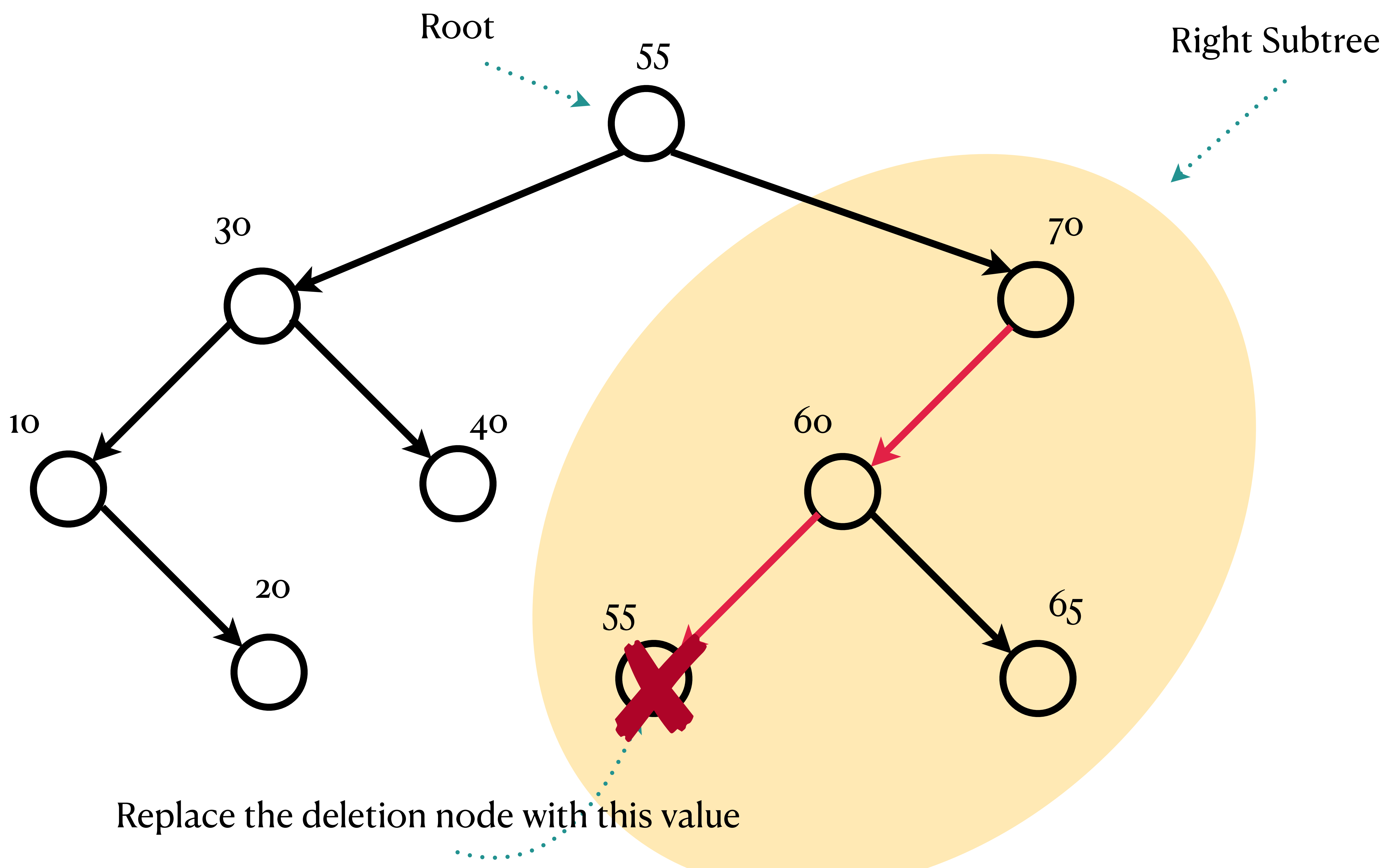
BST: Deletion

Case 3: Deleting a Node with two children

Deletion key: 50

Option 1: Find the smallest element in right subtree

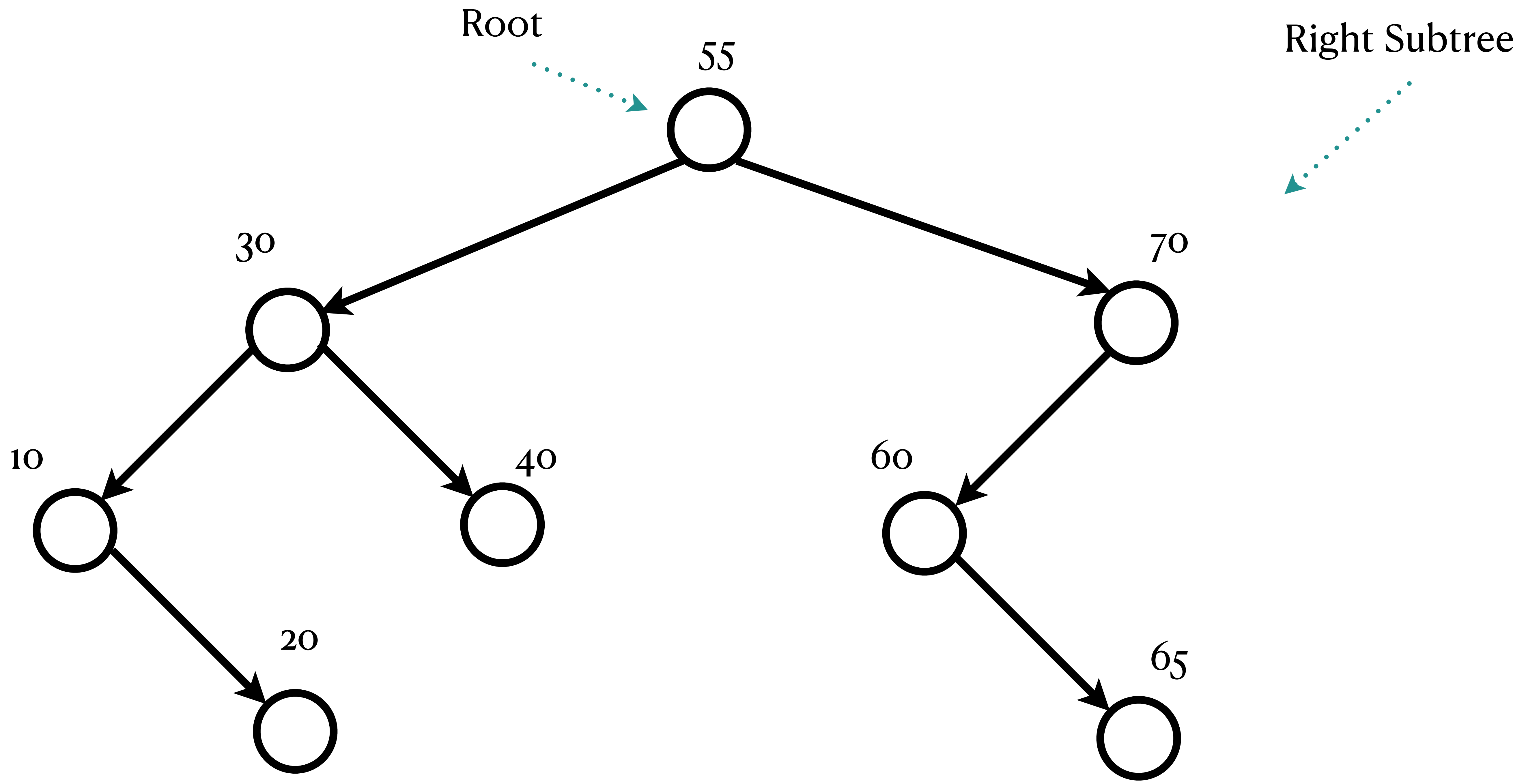
How? Start from root of the right subtree and keep moving left



Replace the deletion node with this value

Smallest node in the right subtree has at most one children (Why?)

BST: Deletion



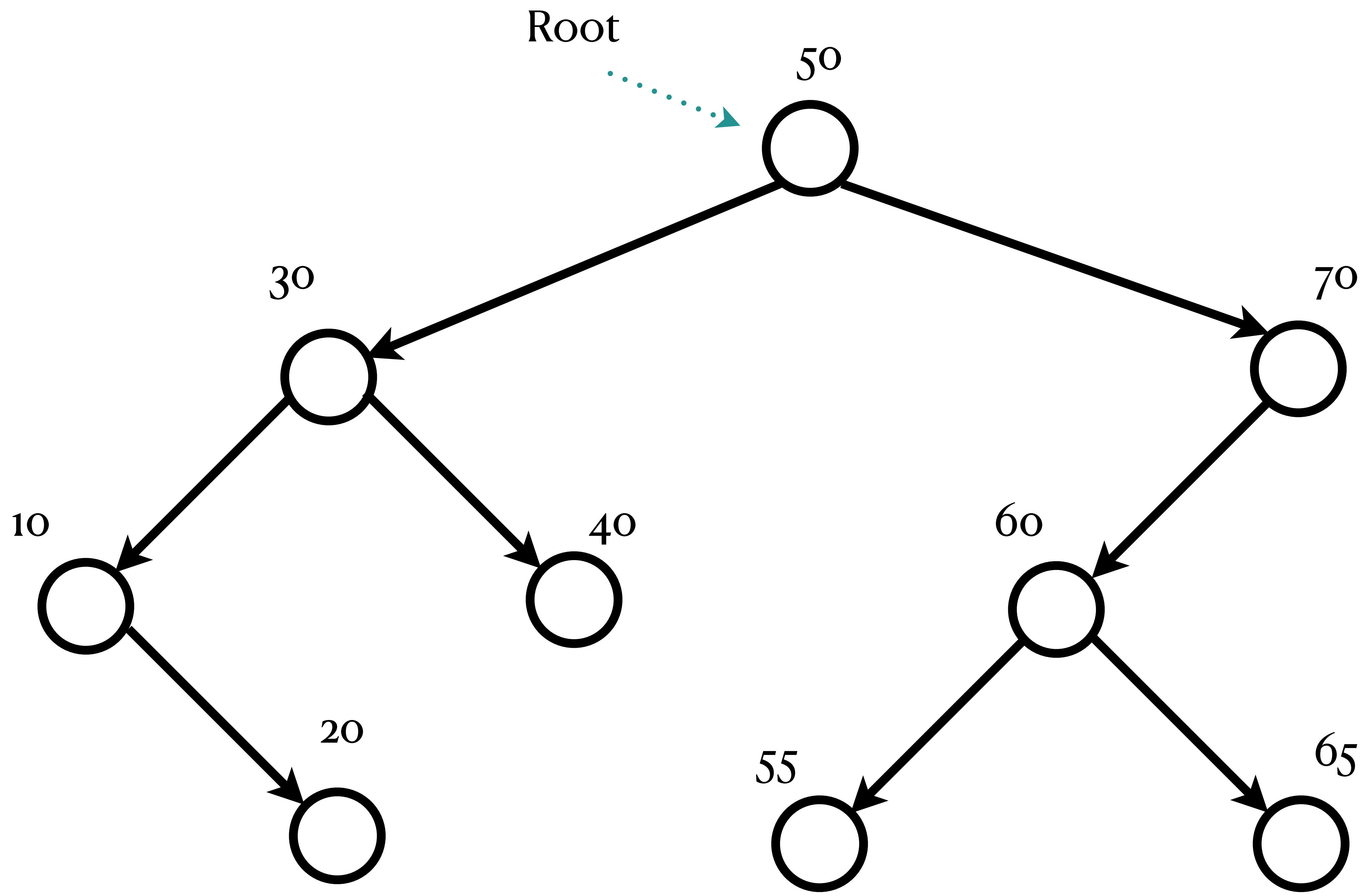
BST: Deletion

Case 3: Deleting a Node with two children

Deletion key: 50

Option 2: Find the largest element in left subtree

How? Start from root of the left subtree and keep moving right



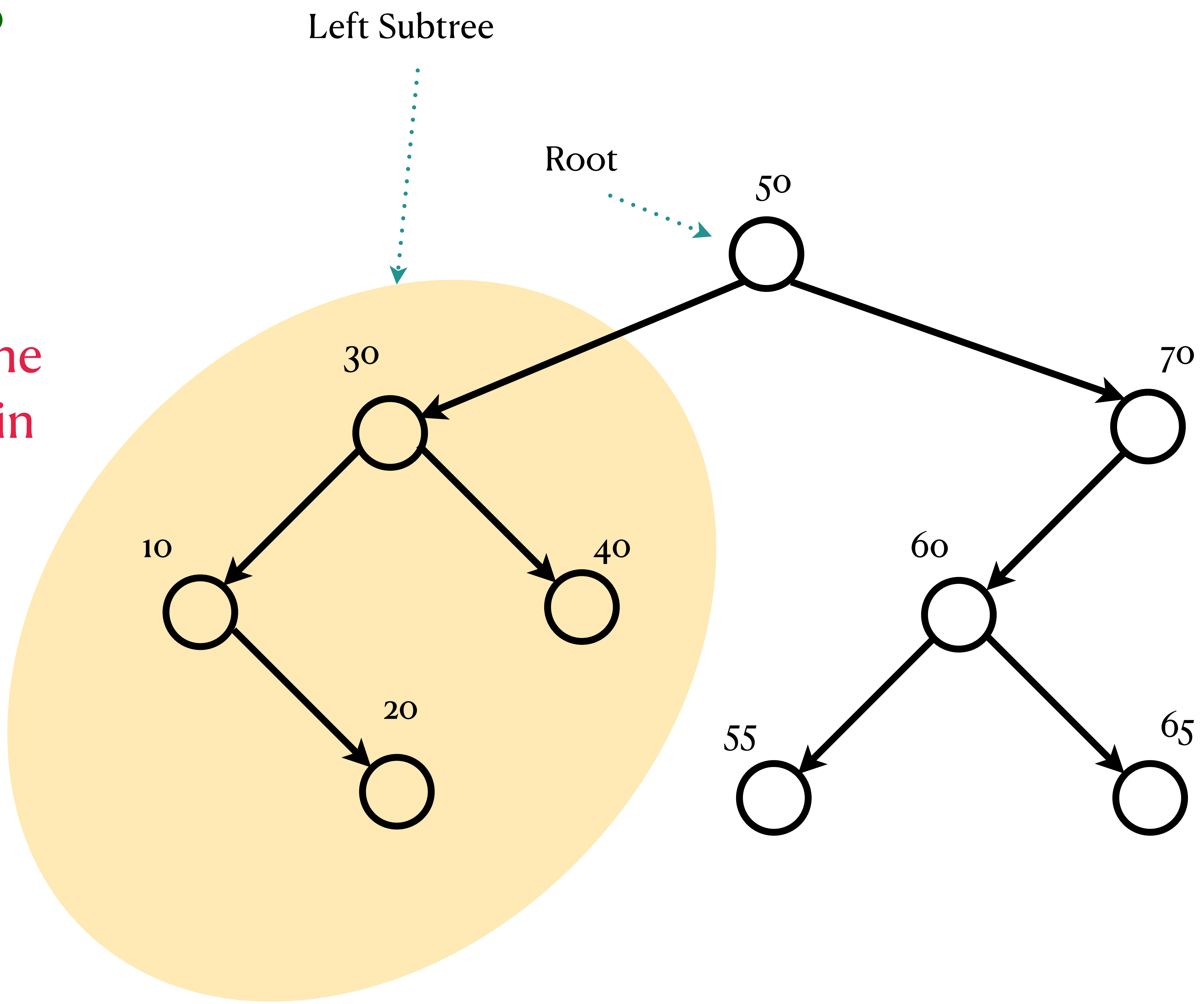
Largest node in the left subtree has at most one children (Why?)

BST: Deletion

Case 3: Deleting a Node with two children

Deletion key: 50

Option 2: Find the largest element in left subtree



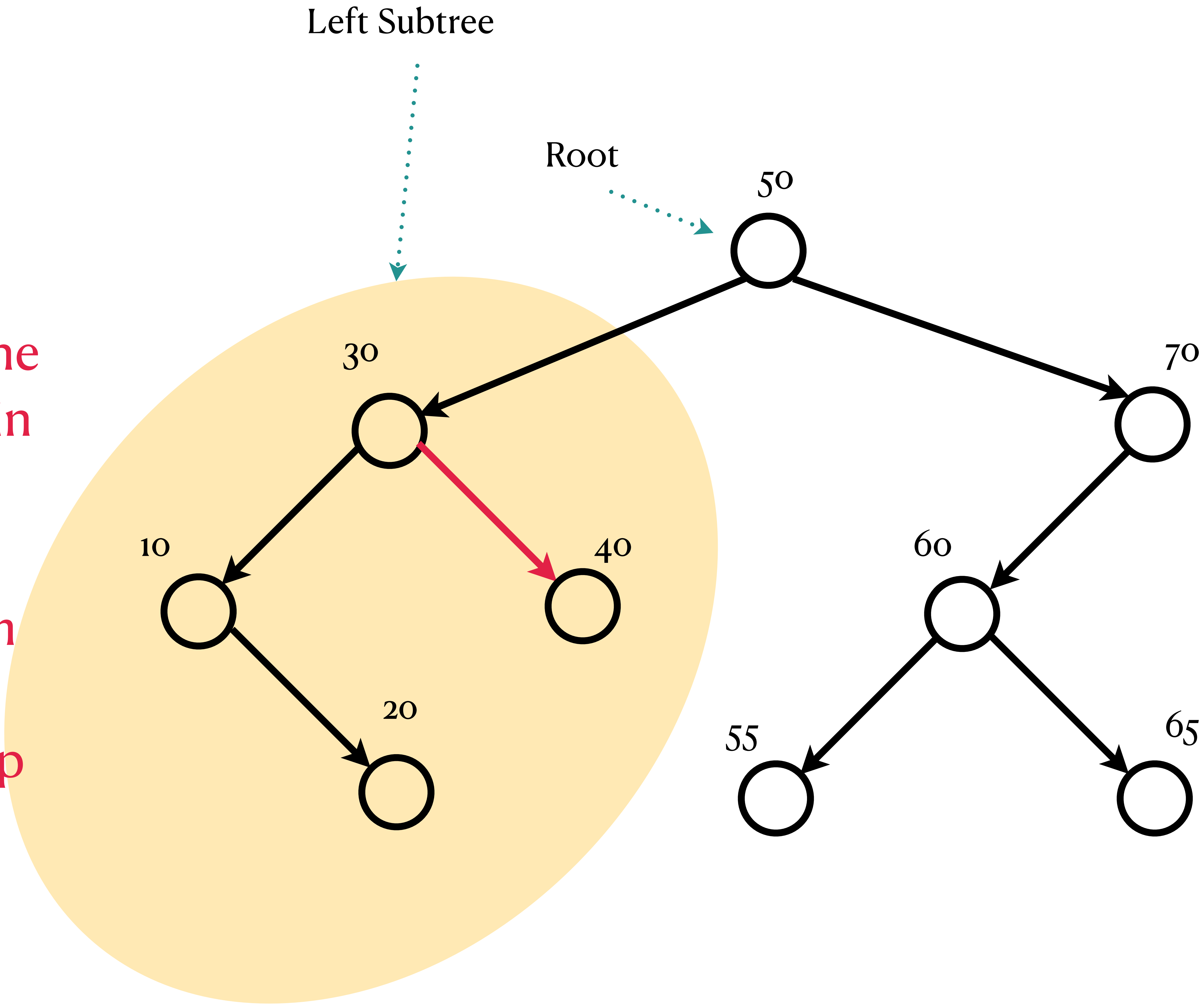
BST: Deletion

Case 3: Deleting a Node with two children

Deletion key: 50

Option 2: Find the largest element in left subtree

How? Start from root of the left subtree and keep moving right



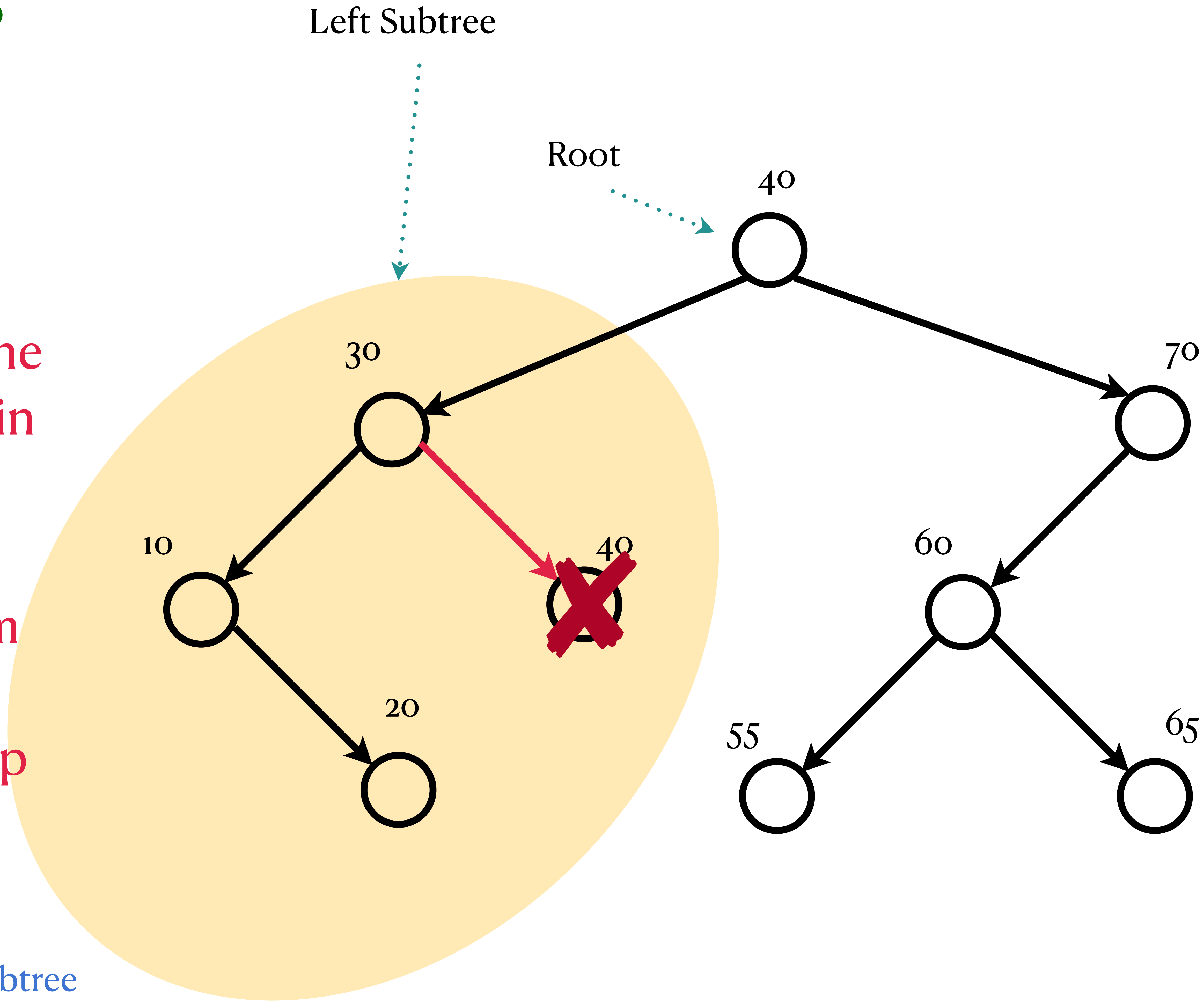
BST: Deletion

Case 3: Deleting a Node with two children

Deletion key: 50

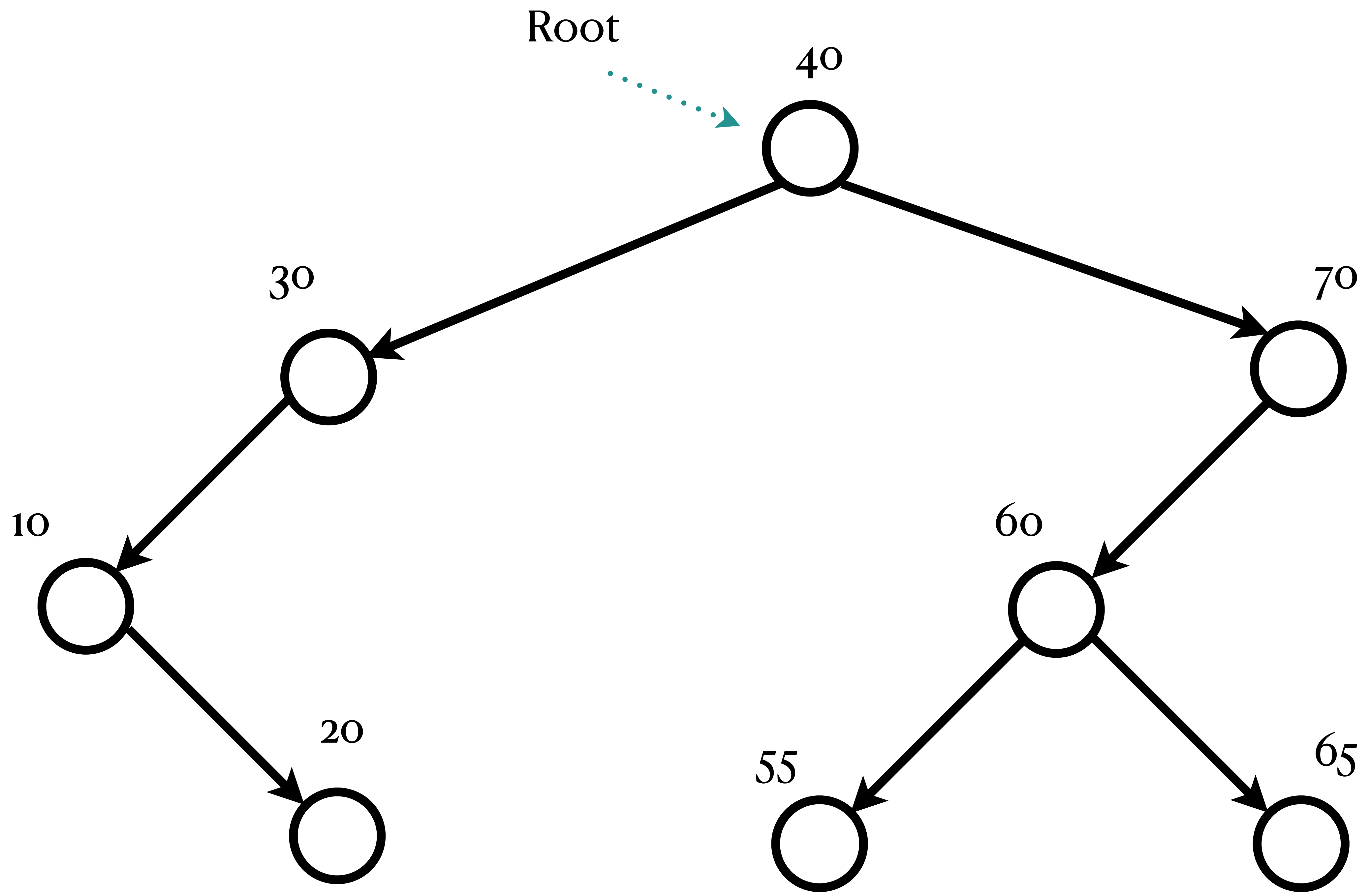
Option 2: Find the largest element in left subtree

How? Start from root of the left subtree and keep moving right



Largest node in the left subtree has at most one children (Why?)

BST: Deletion


Root
40
30
70
10
60
20
55
65