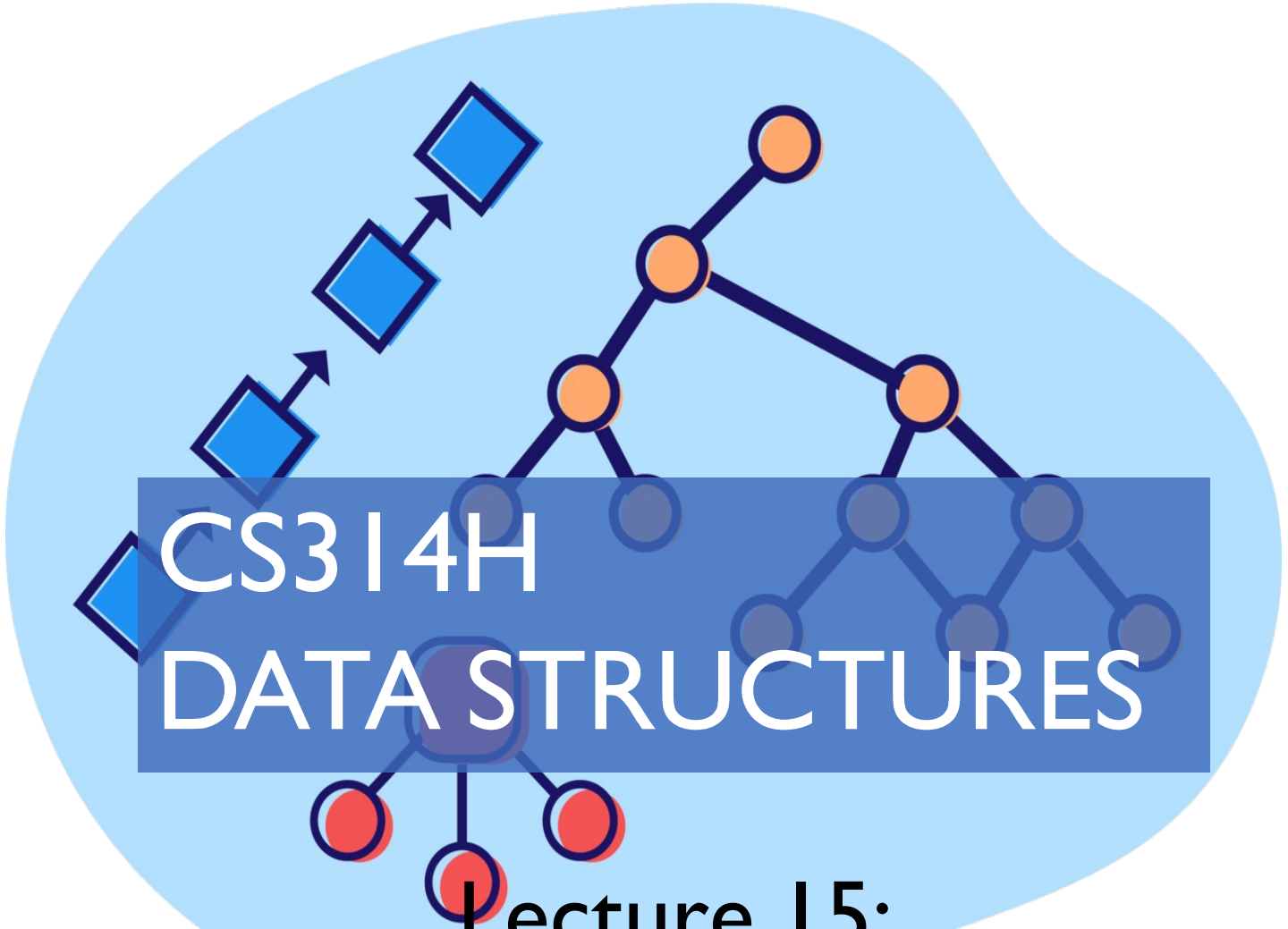


CS314H
DATA STRUCTURES

Lecture 15:
Binary Search Tree

Mikyung Han

Please, interrupt and ask questions AT ANY TIME !

Reminders

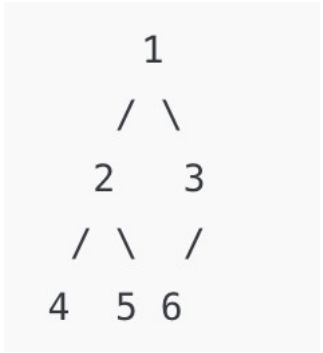
- Exam 1 is currently scheduled to Oct 20 6 - 9 PM
- Exam 1 availability survey: Are you available on Oct 21 6-9 PM?
- Exam 1 topics:
  - Everything we learned (from OOP till BST/AVL intro)
  - Project/testing related questions
  - Same style/format as Quiz 1-3 (these are your practice questions)

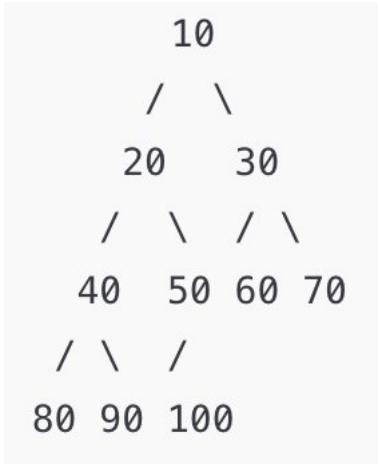
Outline

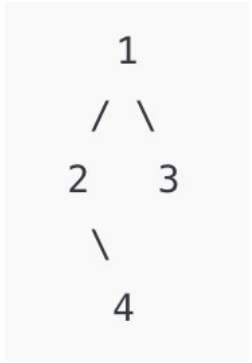
1. Administrative
2. Binary Tree Analysis

Complete Binary Tree

- **All the levels of the tree are filled completely except the lowest level**
  - At the lowest level, nodes are filled from as left as possible
- **Which is NOT a complete binary tree?**







  - Num nodes at level k?
  - Min num nodes at the lowest level?

Complete Binary Tree

- All the levels of the tree are filled completely except the lowest level
  - At the lowest level, nodes are filled from as left as possible
- Num nodes at level k?

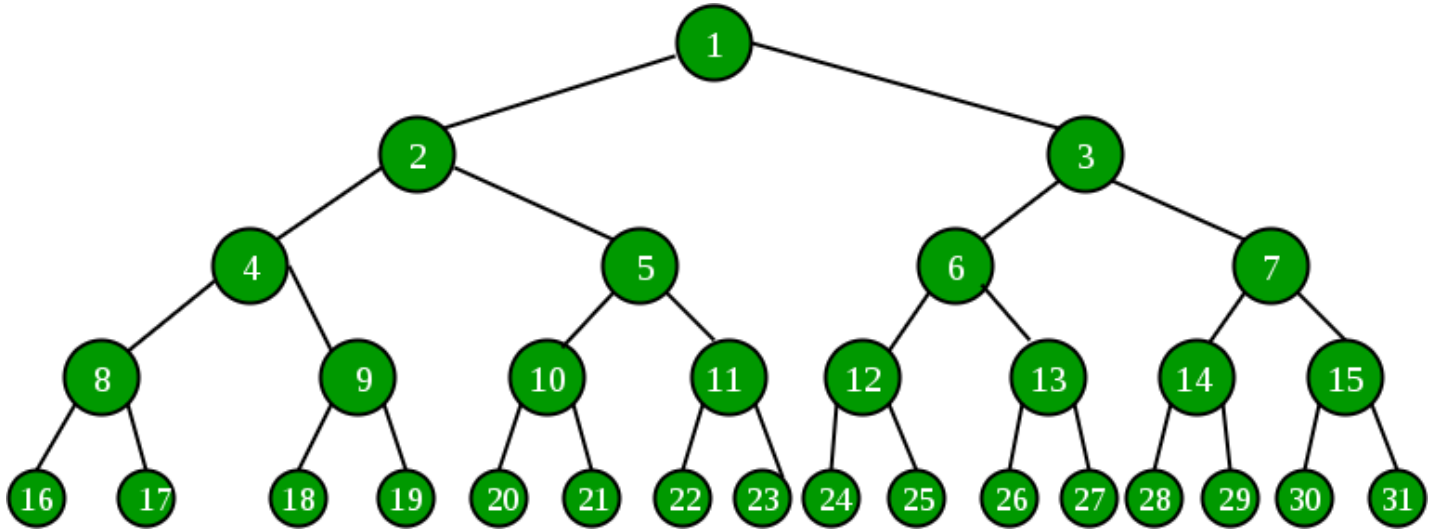
- Min/max num nodes at the lowest level?

Complete Binary Tree

- **All the levels of the tree are filled completely except the lowest level**
  - At the lowest level, nodes are filled from as left as possible
- **Num nodes at level k?**
  - Root is at level 0
  - 2^k
- **Min/max num nodes at the lowest level?**
  - Let k be the number of levels
  - Min: 1
  - Max: 2^{k-1}

Perfect Binary Tree

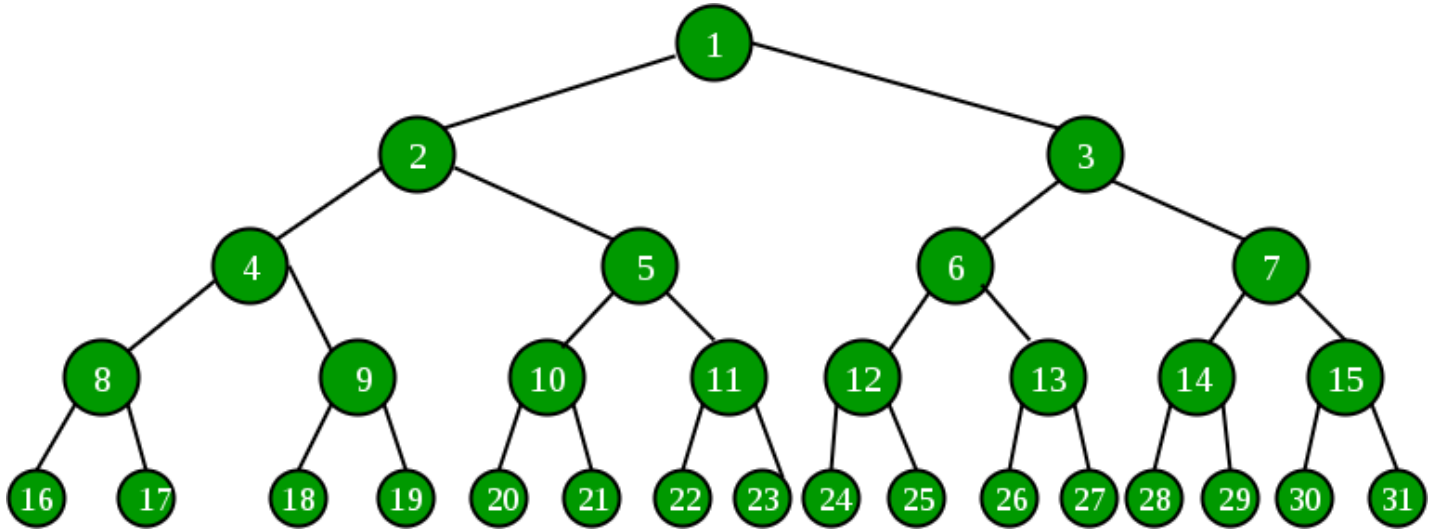
- All the levels of the tree are filled completely including the lowest level



- Total number of nodes given there are total k levels?
- How many leaf nodes?
- How many internal nodes?

Perfect Binary Tree

- All the levels of the tree are filled completely including the lowest level



Let k be the number of levels

- Total 2^k-1 nodes
- 2^{k-1} leaf nodes
- 2^{k-1}-1 internal nodes

Outline

1. Administrative
2. Binary Tree Analysis
3. Binary Search Tree

Suppose we need a dictionary that inserts, finds, deletes items efficiently

- How can we do this with binary tree?

Binary Search Tree

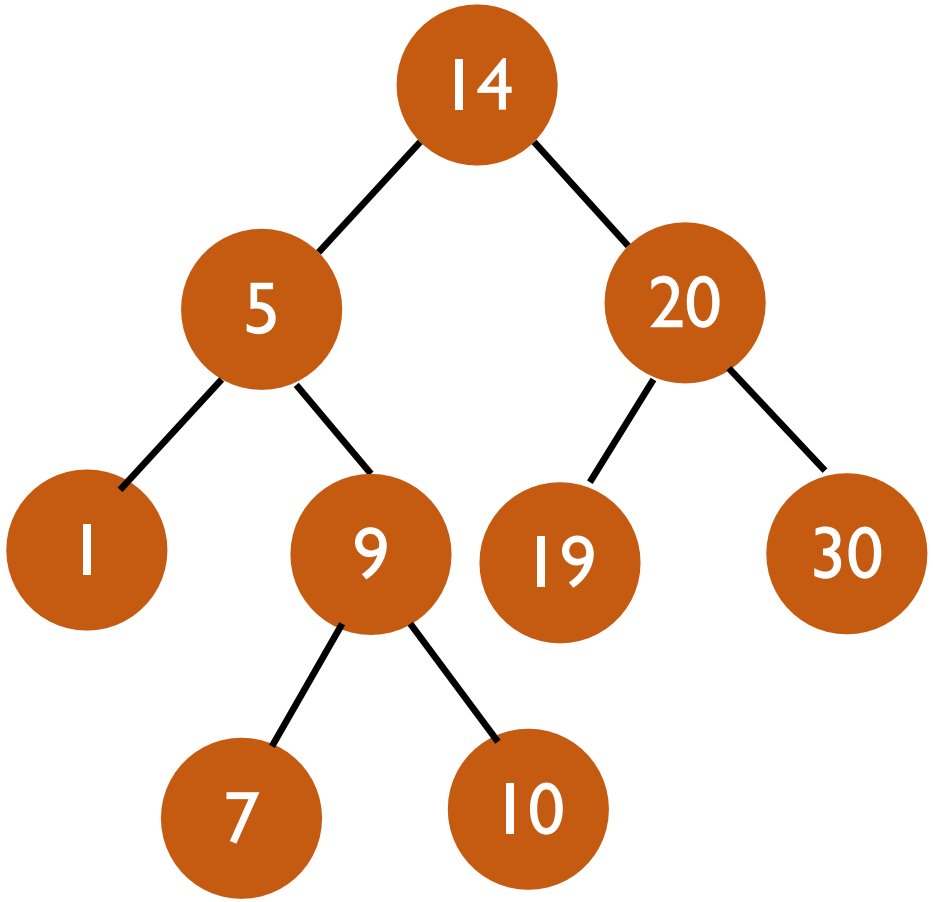
BST is a binary tree with BST property

BST Property
For every node x,
- all keys in x's left subtree have smaller values than the key in x
- all keys in x's right subtree have larger values than the key in x

Does it say anything about the tree being balanced?

Find on BST

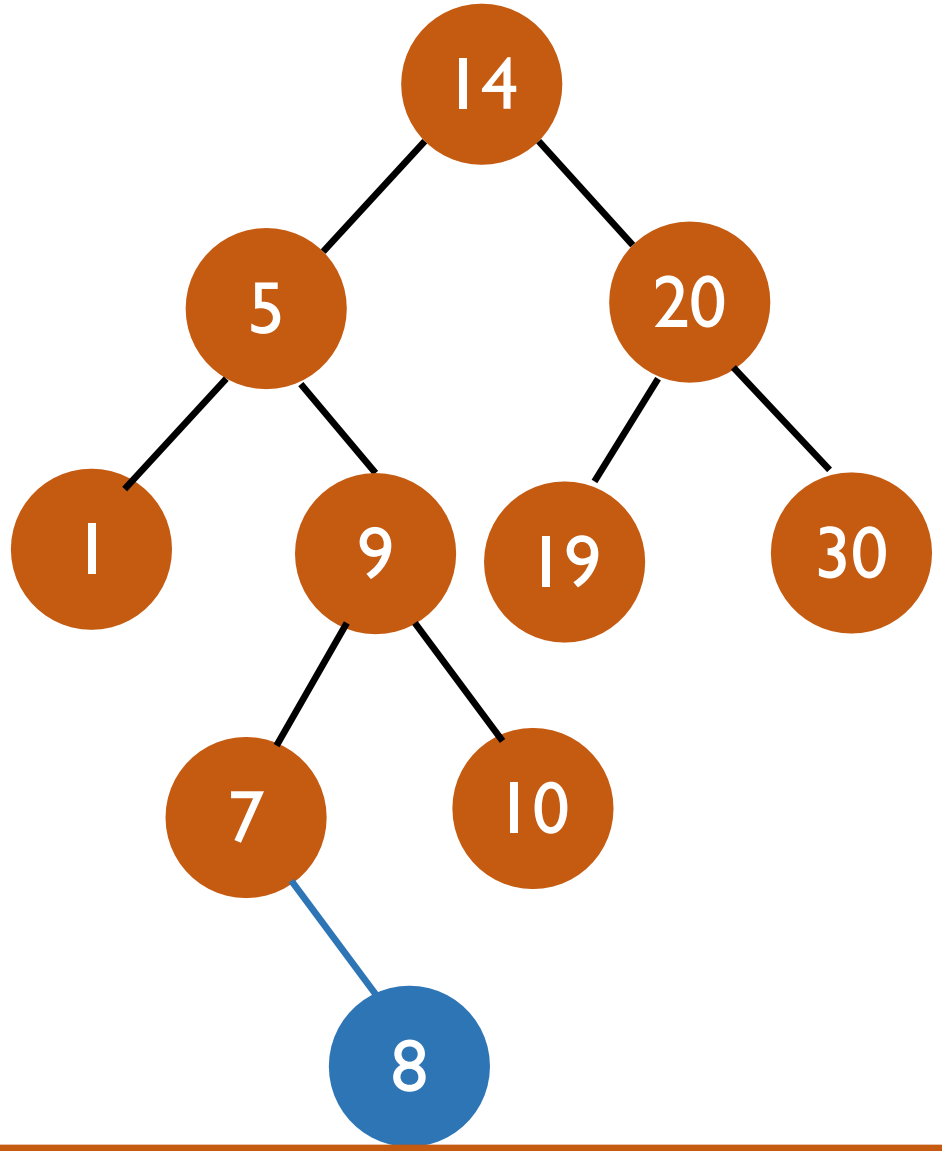
```
Find 8
Find 10
```



What is the time complexity of find?

Insert on BST

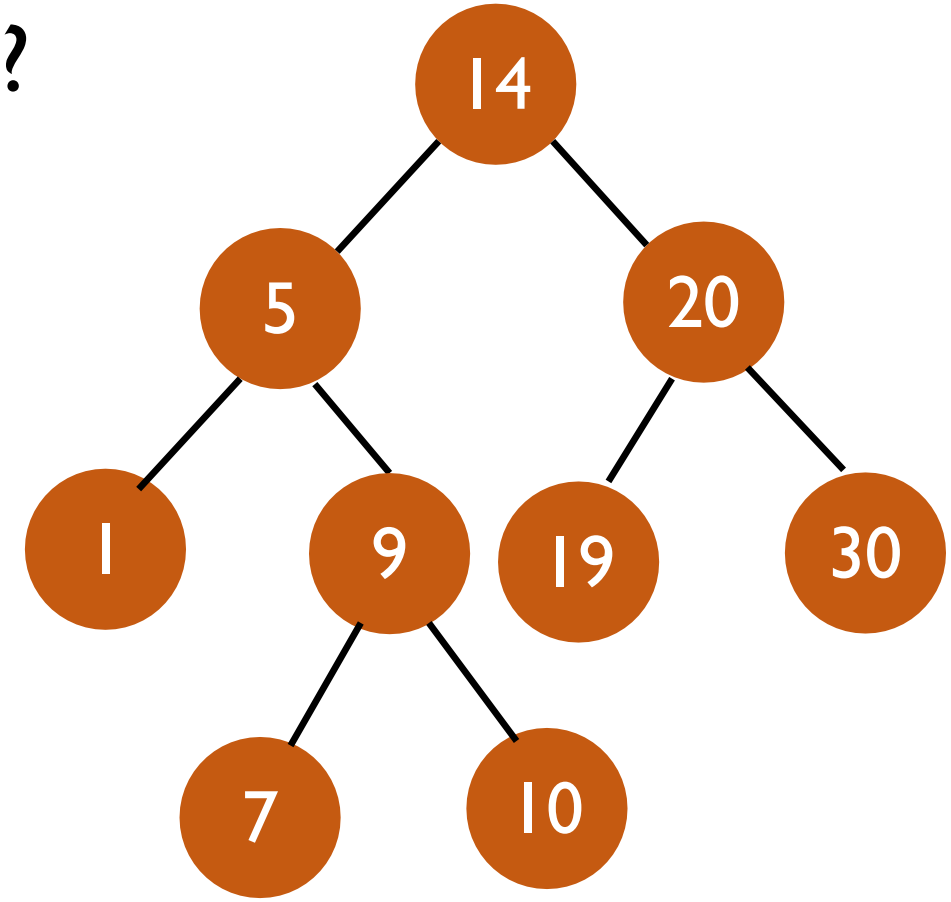
```
Insert 8
```



What is the time complexity of insert?

Duplicate on BST?

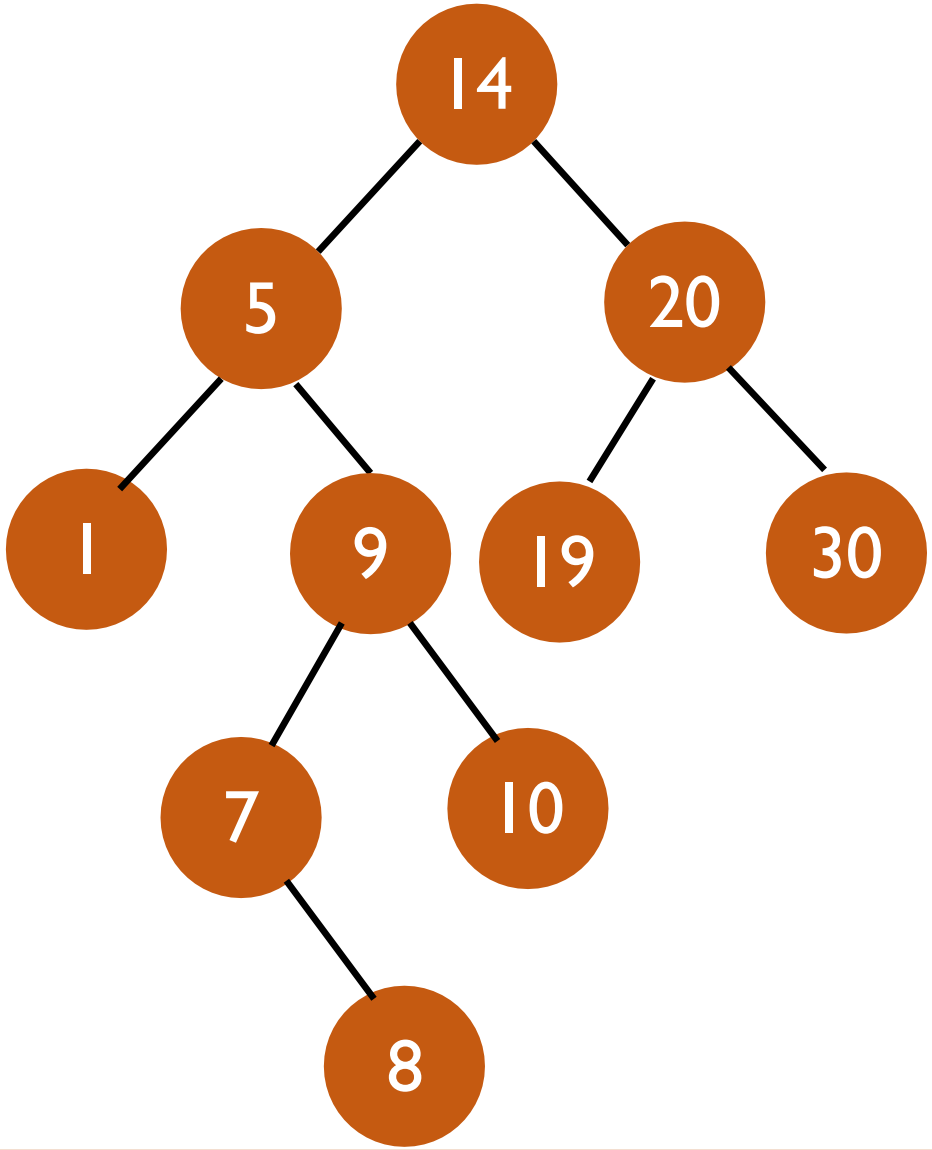
```
Insert 10
```



Flag an error, ignore, or allow duplicate – all possible options

Delete on BST

```
Delete 19
```



Case 1: Deleting on a leaf node – simple!

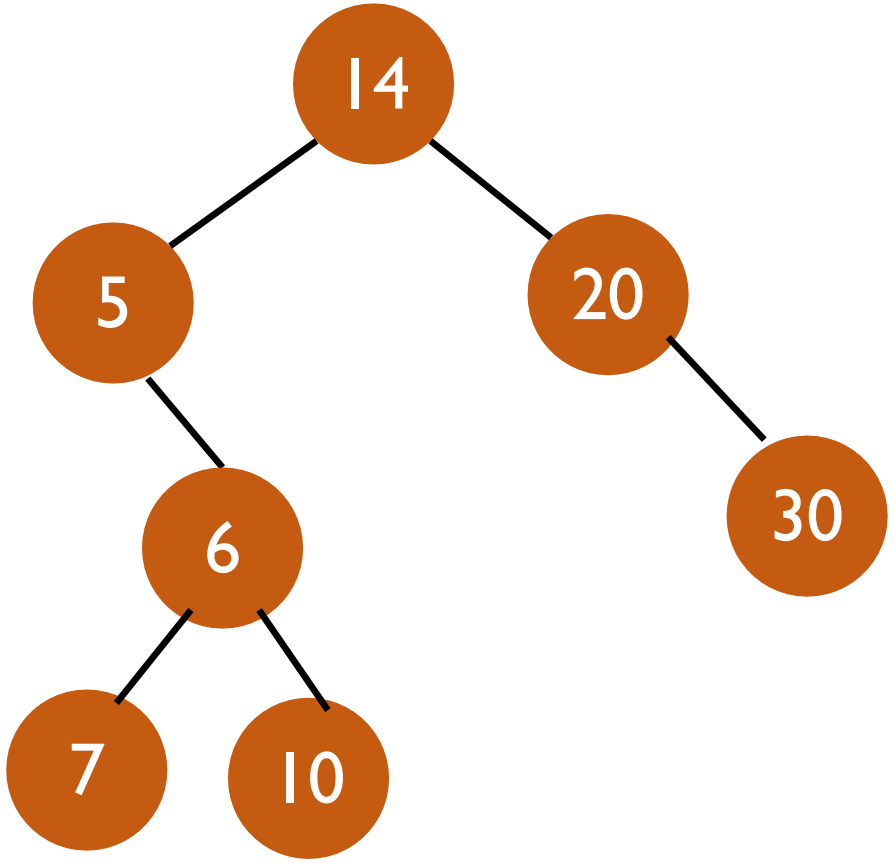
Deleting an internal node

Case 2: when the internal node has either left or right subtree

Case 3: when the internal node has both left and right subtree

Case 2: Deleting an internal node with either left/right subtree

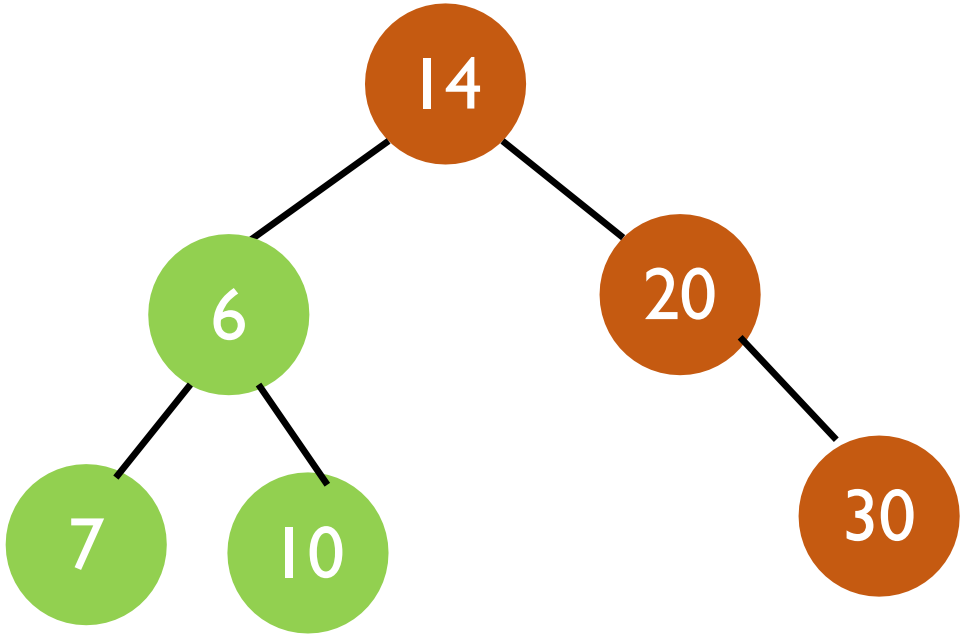
```
Delete 5
```



Which node should replace 5 to preserve BST property?

Case 2: Deleting an internal node with either left/right subtree

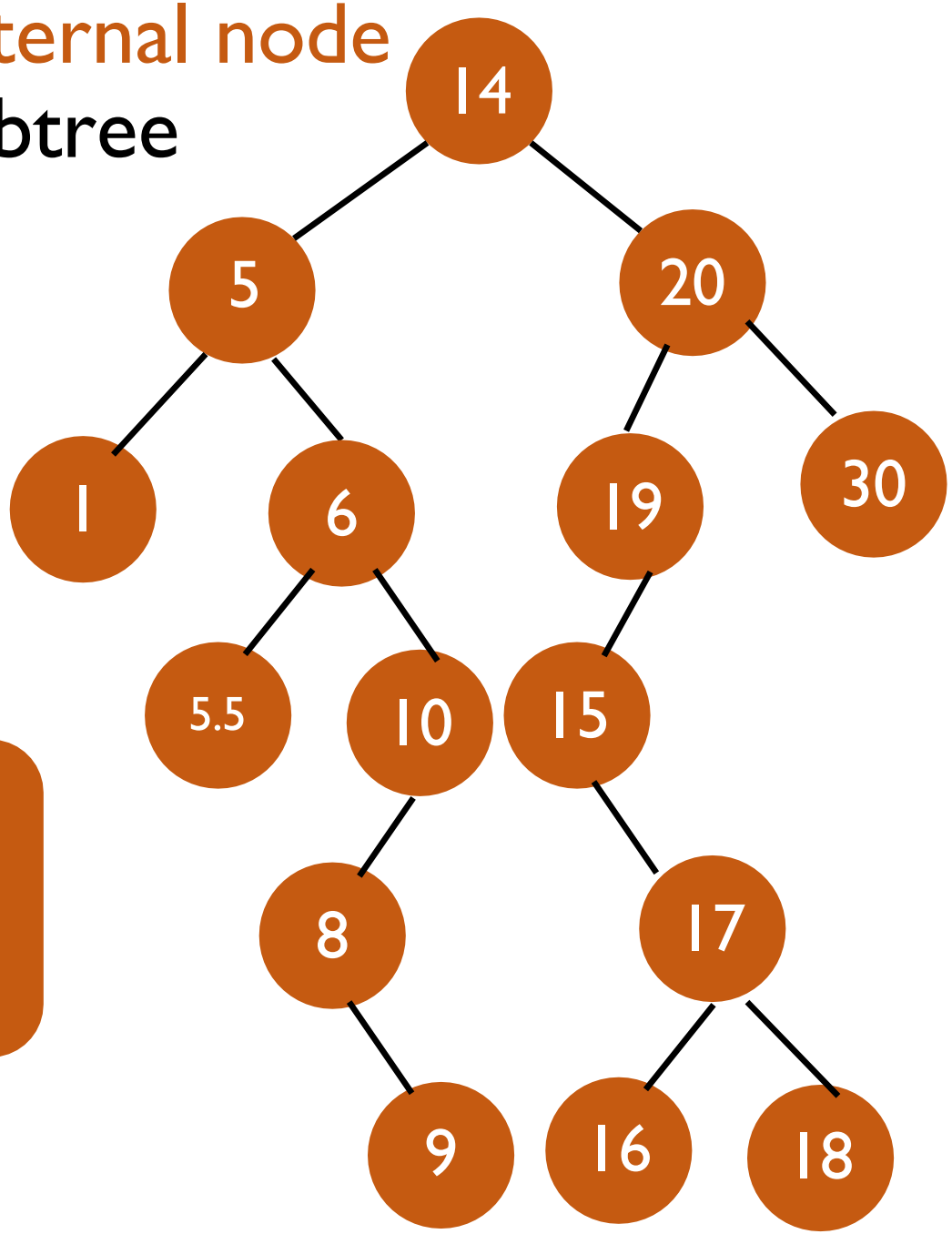
```
Delete 5
```



Its subtree can replace the internal node. Why?

Case 3: Deleting an internal node with both left/right subtree

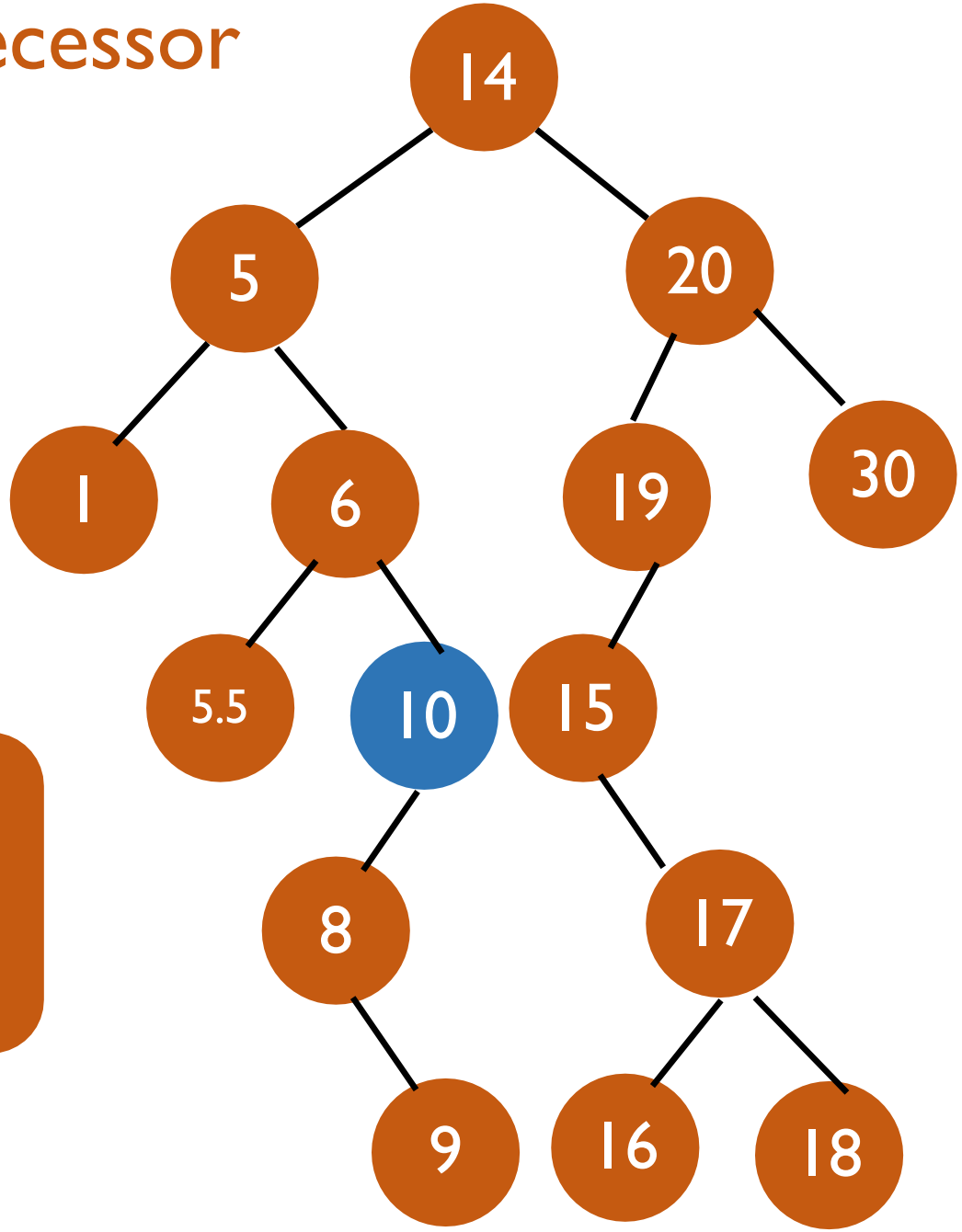
```
Delete 14
```



Which node should replace 14 to preserve BST property?

Replace with predecessor

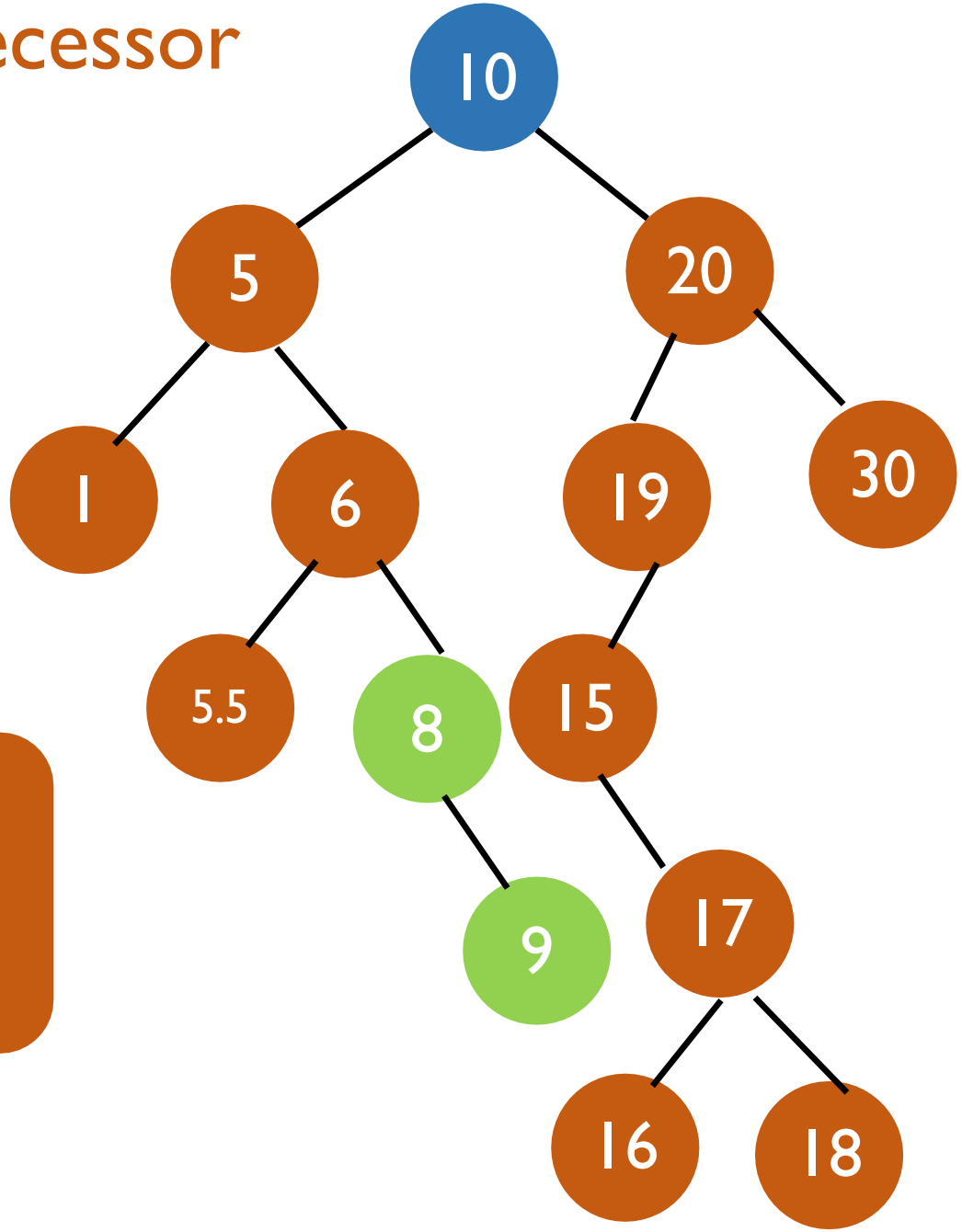
```
Delete 14
```



Replace with the largest key in the left subtree

Replace with predecessor

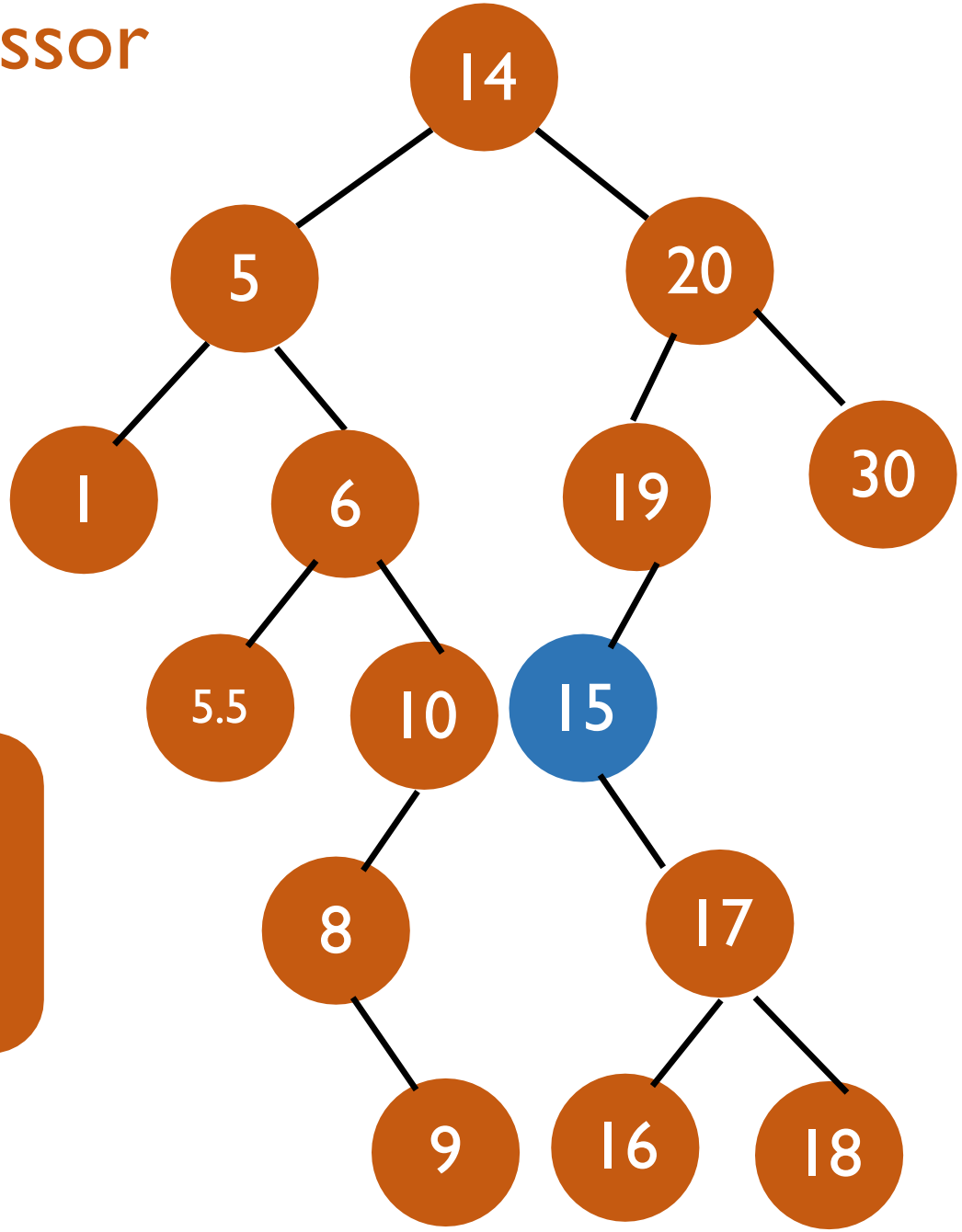
```
Delete 14
```



Replacing
will always result in
case 1 or case 2

Replace with successor

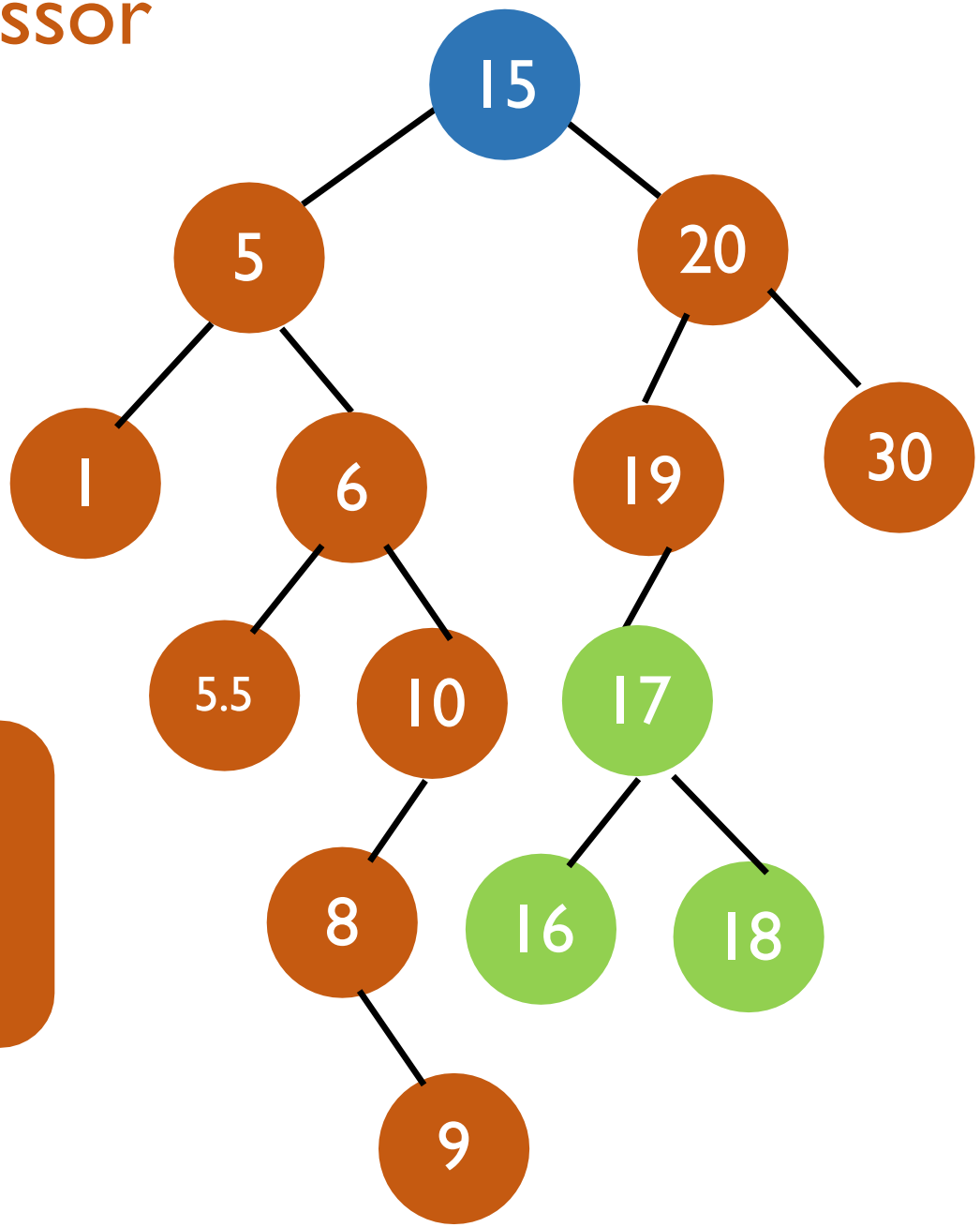
```
Delete 14
```



Replace with the smallest key in the right subtree

Replace with successor

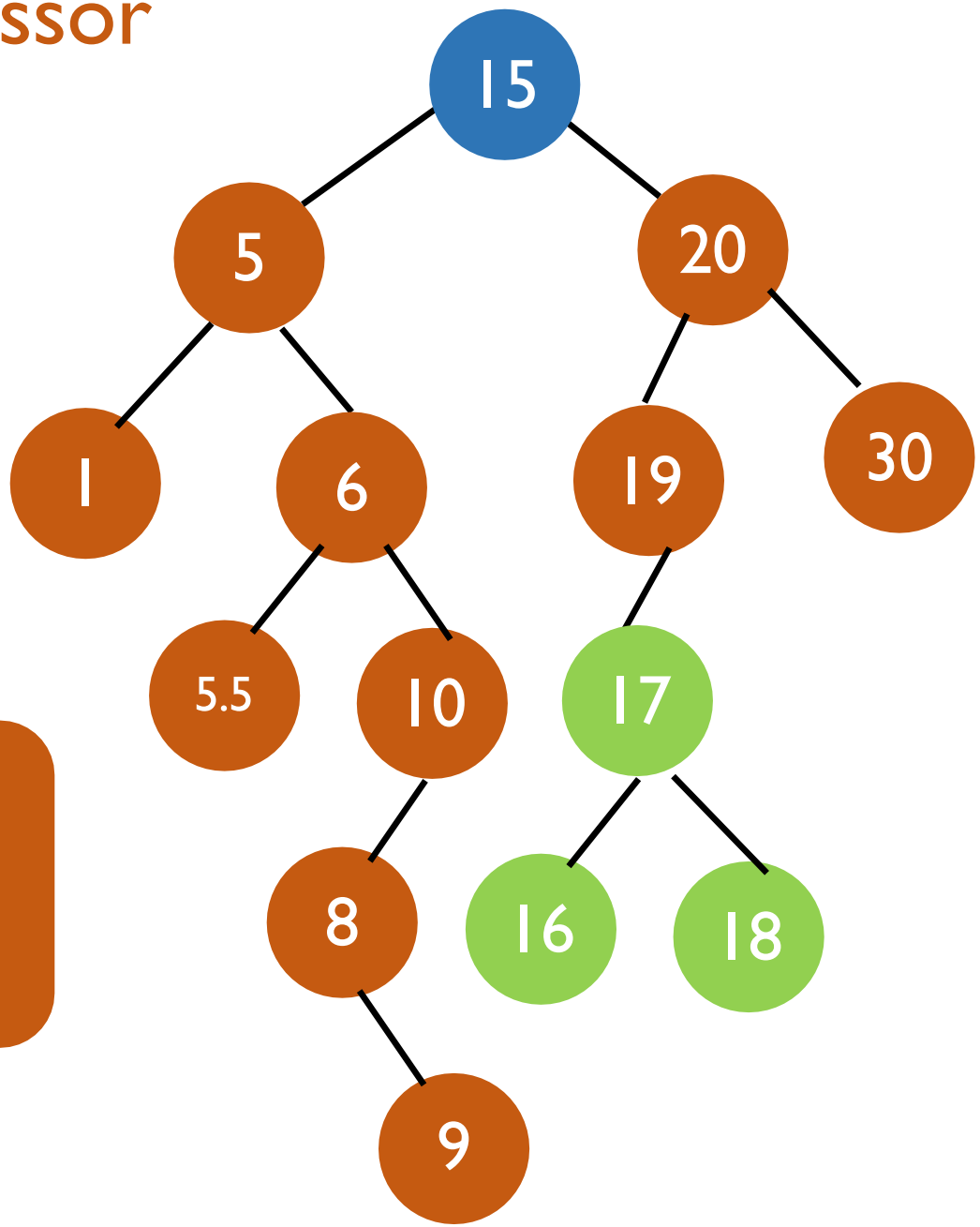
Delete 14



Replacing
will always result in
case 1 or case 2

Replace with successor

Delete 14



Replacing
will always result in
case 1 or case 2