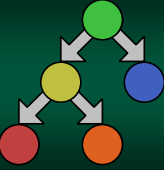




Binary Search Trees

Part 10

1



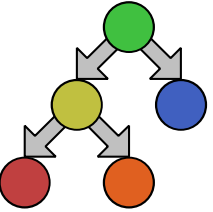
Binary Search Trees

Climbing a tree has never been so fun

2

Binary Search Trees


- Binary Search Tree (BST) is a special type of binary tree that sorts nodes by value
- For each node...
 - all the nodes on the **left** branch are **less than** it
 - all the nodes on the **right** branch are **greater than** it



3

Searching the Tree


- Since the tree divides the problem progressively by two, the time complexity is only $O(\log n)$
- Which gives that all the benefits of a sorted array
- Worst case is $O(n)$ – if the tree is a list-like chain



4

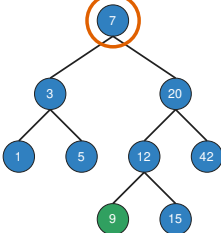
Search logic (looking for S)

- Looking for If **S** is equal to the current node, you found it
- If **S** is **smaller** than the current node, take the **left branch**
- If **S** is **bigger** than the current node, take the **right branch**
- If there are no branches, **S** was not found



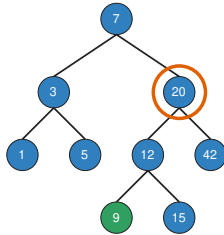
5

Searching for 9



6

Searching for 9



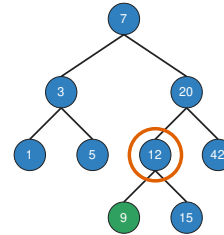
Fall 2024

Sacramento State - Oak - CS130

7

7

Searching for 9



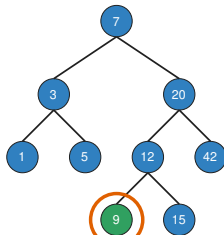
Fall 2024

Sacramento State - Oak - CS130

8

8

Searching for 9



Fall 2024

Sacramento State - Oak - CS130

9

9

Inserting into the Tree

- Inserting is handled exactly like a search
- The only difference is that if the item is not found, the node is added
- If we reach a leaf...
 - we are already at the max-depth of the tree
 - we are at the node that needs to be updated
 - so, add a left or right node (based on value)
 - ... wow, this is easy!

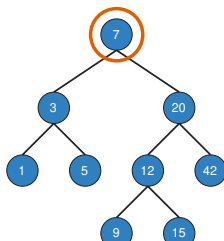
Fall 2024

Sacramento State - Oak - CS130

10

10

Inserting 4



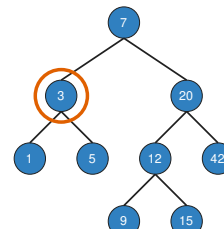
Fall 2024

Sacramento State - Oak - CS130

11

11

Inserting 4



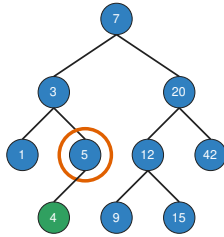
Fall 2024

Sacramento State - Oak - CS130

12

12

Inserting 4



Fall 2024

Sacramento State - Oak - CS110

13

13

Binary Search Trees vs. Arrays

- Insert into a Binary Search Tree
 - traverses down the tree
 - then add itself as a leaf
 - it requires only $O(\log n)$
- Insert into an Array
 - must expanded when new elements are added
 - ...and compacted when elements are removed
 - these requires $O(n)$

Fall 2024

Sacramento State - Oak - CS110

14

14



Deleting From a BST

Not as easy as Heaps

15

15

Deleting From a BST



- Deleting from a BST is a tad more tricky
- When the node is deleted, the tree needs to be re-linked to *still preserve the ordering*

Fall 2024

Sacramento State - Oak - CS110

16

16

Deleting From a BST



- Fortunately, while the logic might seem a tad hard, it is fairly straight forward
- There are 3 different scenarios that must be taken into account based on the node being deleted

Fall 2024

Sacramento State - Oak - CS110

17

17

Case 1: The Node is a Leaf

- If the node (to be deleted) is a leaf, it can simply be removed from the parent
- This is due to, the fact that, nothing is linked from the node



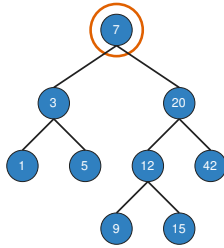
Fall 2024

Sacramento State - Oak - CS110

18

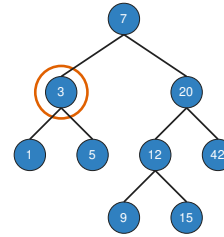
18

Case 1: Deleting 5



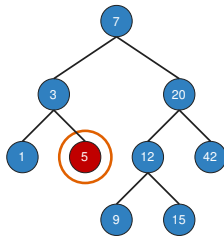
19

Case 1: Deleting 5



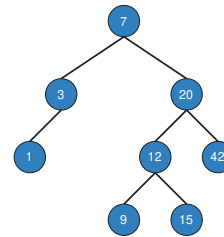
20

Case 1: Deleting 5



21

Case 1: Deleting 5



22

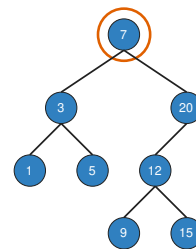
Case 2: One Child

- If the deleted node has a single child, then that child can be "*promoted*"
- The deleted node's child is moved into its position
- ... so, the subtree moves into the space created by the deleted node



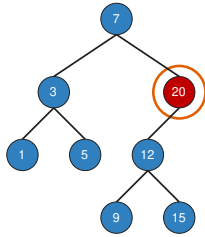
23

Case 2: Deleting 20



24

Case 2: Deleting 20



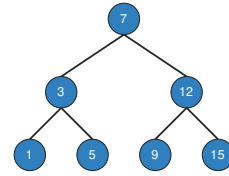
Fall 2024

Sacramento State - Oak - CS110

25

25

Case 2: Deleting 20



Fall 2024

Sacramento State - Oak - CS110

26

26

Case 3: Two Children

- However, if the deleted node has two children, then things get a tad more complex
- We need to find a node to "*promote*" to the deleted node's position



Fall 2024

Sacramento State - Oak - CS110

27

27

Case 3: Two Children

- We need to find a value that is mathematically next to the deleted value
- There are two equally valid options
- Choose one:
 - find the **maximum** node of the **left** (smaller) branch
 - ... or find the **minimum** node of the **right** (larger) branch

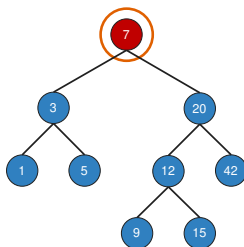
Fall 2024

Sacramento State - Oak - CS110

28

28

Case 3: Delete 7 – Max of Minimums



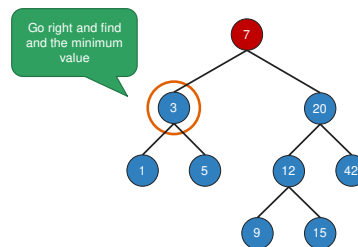
Fall 2024

Sacramento State - Oak - CS110

29

29

Case 3: Delete 7 (Going Right Once)

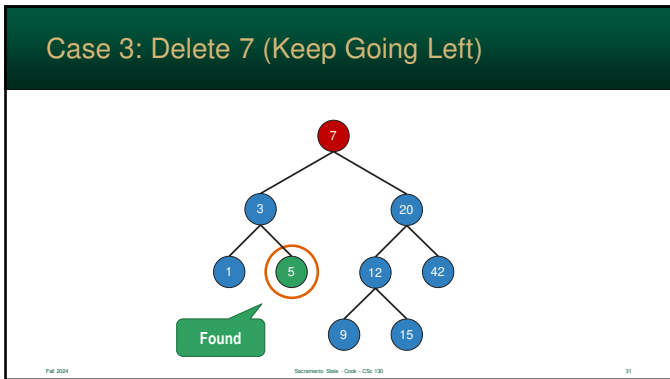


Fall 2024

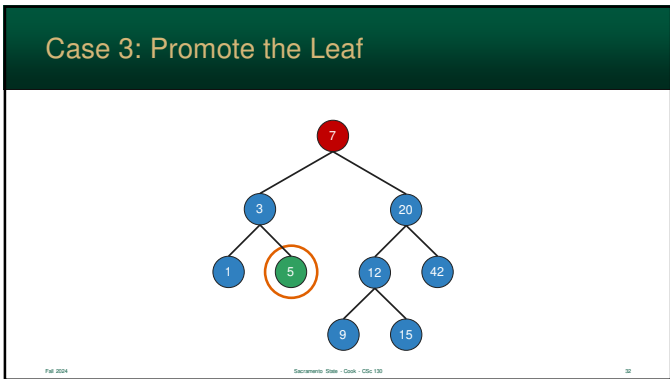
Sacramento State - Oak - CS110

30

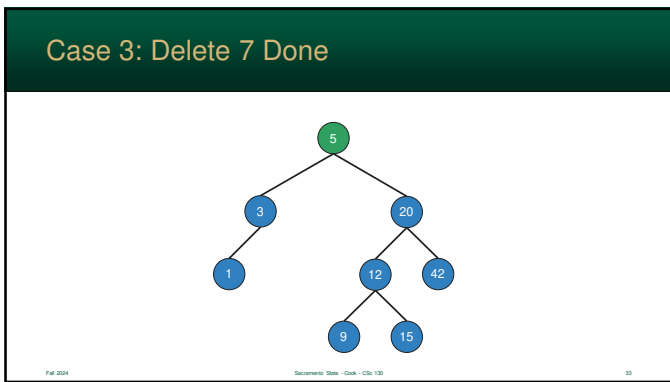
30



31



32

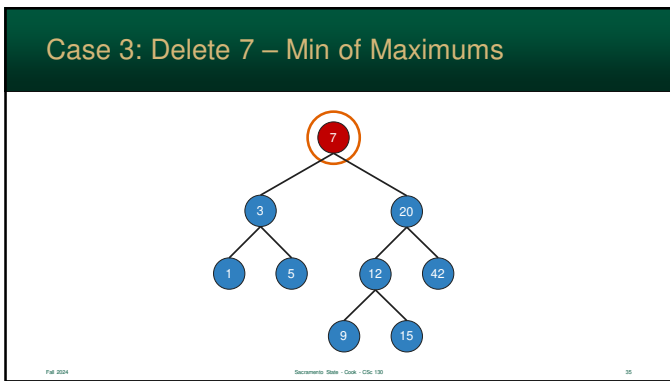


33

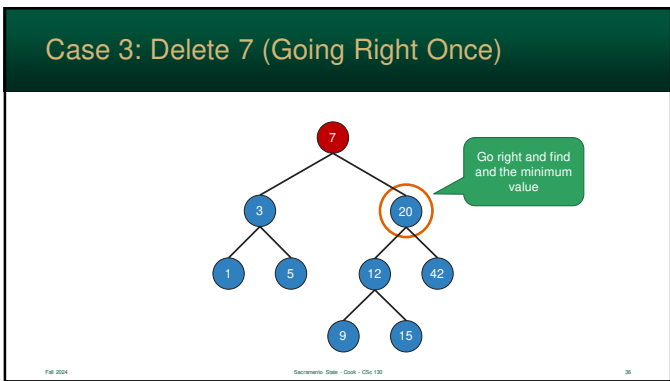
Case 3: Two Children

- It doesn't matter which approach you use
- As long as you find a value that fits mathematically into the old node's position it works.

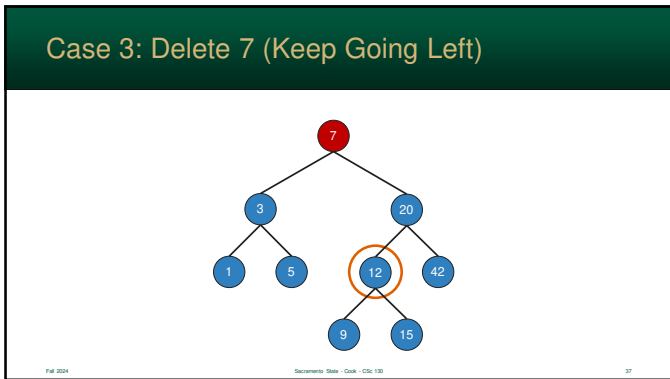
34



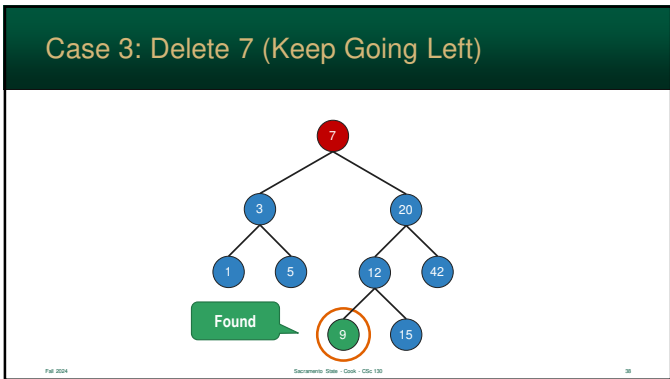
35



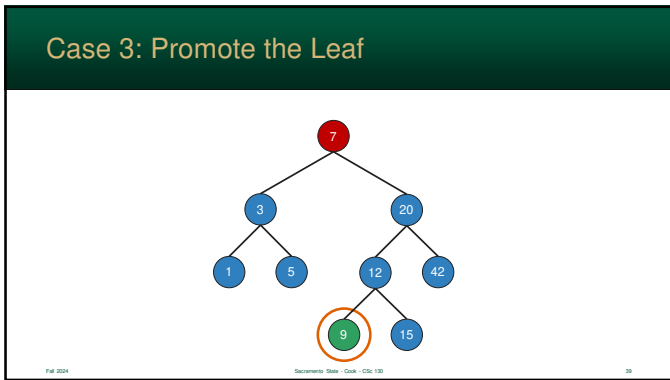
36



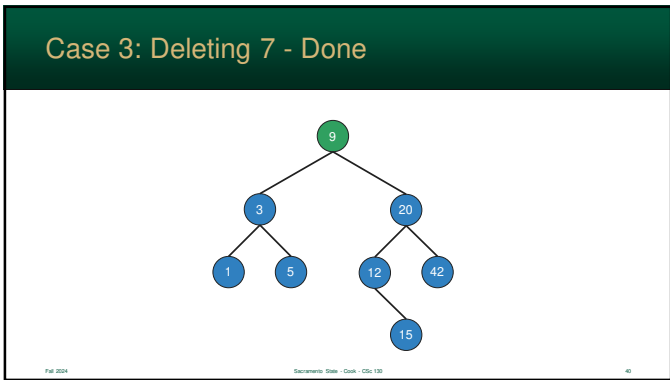
37




38



39



40




The Problem with BSTs

Climbing a tree has never been so fun

41

The Problem with BSTs

- In a BST, internal nodes never change – and have a profound affect on the rest of the tree
- There are cases where the tree is unbalanced – one particular path contains most of the data



42

The Problem with BSTs

- This can easily occur if the data is not truly random (which is generally the case)
- When this happens, **the time complexity slowly deteriorates to $O(n)$**



Fall 2024

Sacramento State - Oak - CS130

43

43

Let's Try Adding...

- Let's add the following number to a binary search tree
- How did this tree look?

○	
○	7, 8, 21, 43, 76
○	

Fall 2024

Sacramento State - Oak - CS130

44

44

Let's add: 7, 8, 21, 43, 76



Fall 2024

Sacramento State - Oak - CS130

45

45

Let's add: 7, 8, 21, 43, 76



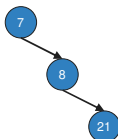
Fall 2024

Sacramento State - Oak - CS130

46

46

Let's add: 7, 8, 21, 43, 76



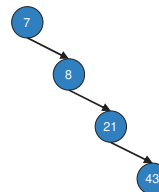
Fall 2024

Sacramento State - Oak - CS130

47

47

Let's add: 7, 8, 21, 43, 76



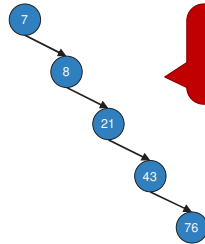
Fall 2024

Sacramento State - Oak - CS130

48

48

Let's add: 7, 8, 21, 43, 76



Our tree is unbalanced & no better than a linked list!

Fall 2024

Sacramento State - Oak - CS 130

49

49

Sort in a Binary Search Tree?

- Unfortunately, while this might seem like a good idea, this is not a great solution
- Binary Search Trees can deteriorate into linked lists
- So...
 - $O(\log n)$ search can quickly deteriorate to $O(n)$
 - ...and $O(n \log n)$ sort can deteriorate to $O(n^2)$

Fall 2024

Sacramento State - Oak - CS 130

50

50