

Announcements

1. Exam 2 Review in Lab!
 - a. New FRQs
 - b. New Practice Coding Questions
2. Exam 2 Next Week (M-W)



Join Code: **225**

Warm-Up Question: How would you explain the binary search tree property? Put your answer in the discord!





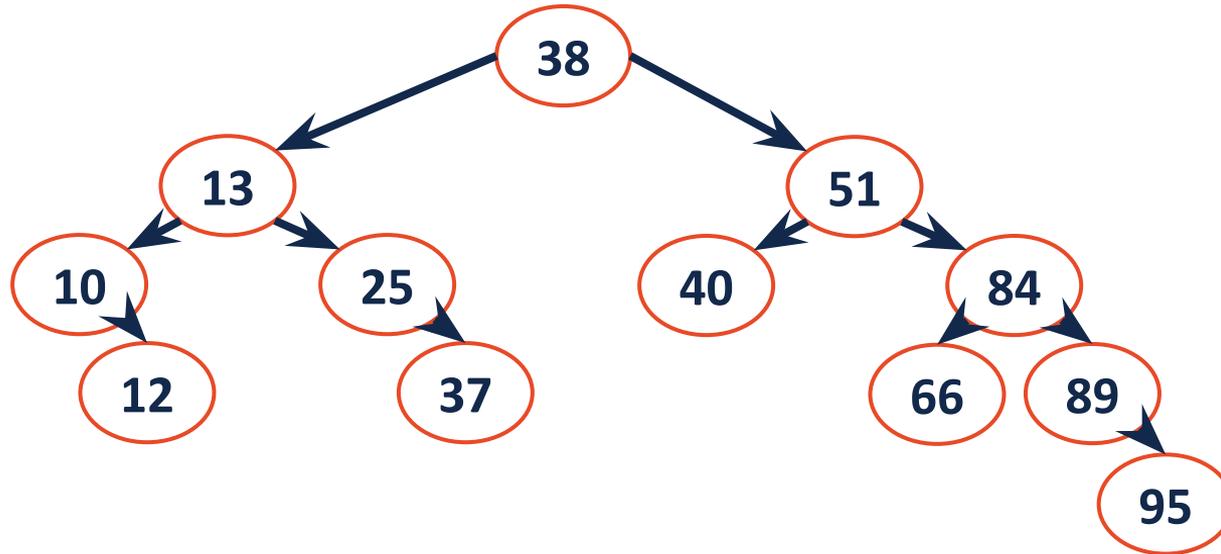
Binary Search Tree Implementations

Learning Objectives

1. Implement the BST Remove Algorithm
2. Analyze the different runtimes of dictionary implementations



Remove(12)



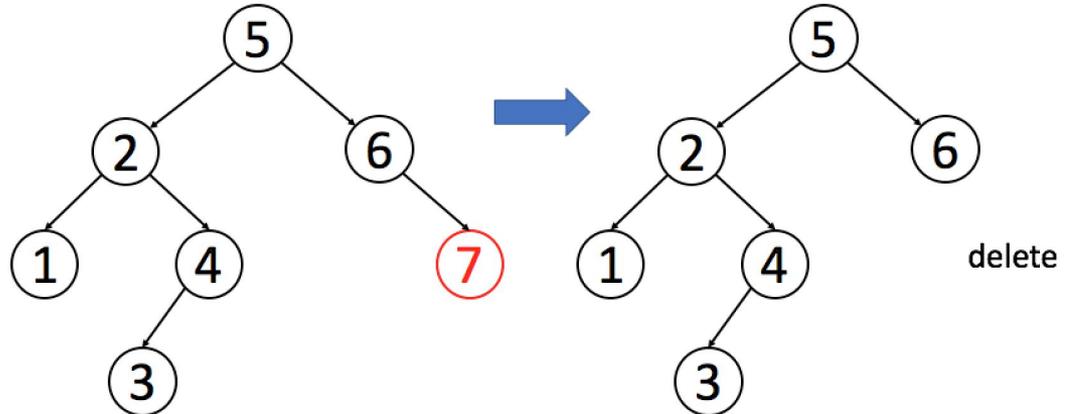
Remove

Case 1 (No child):

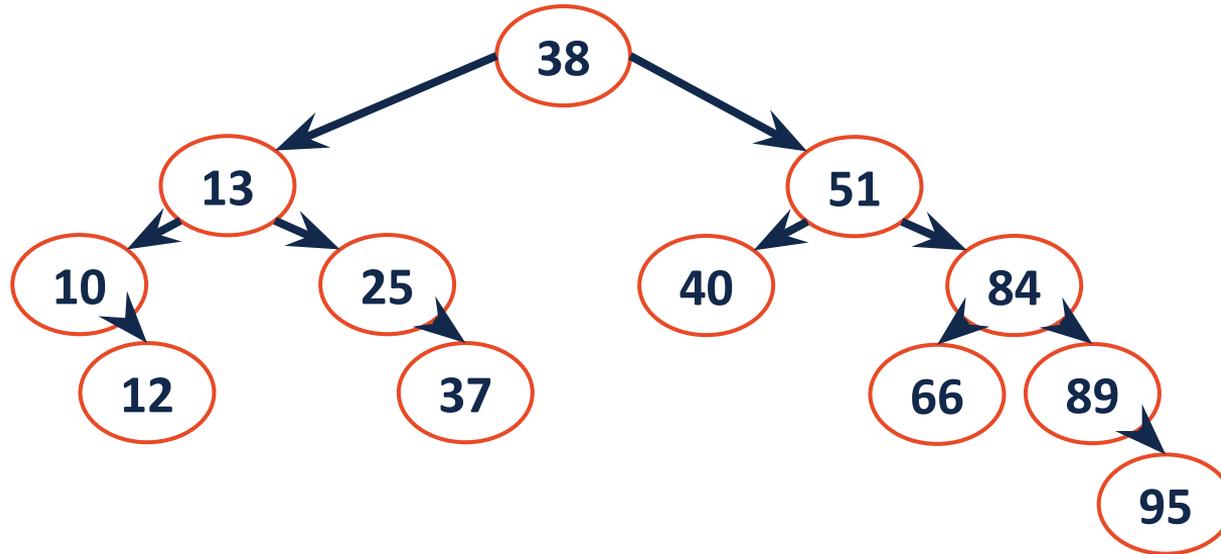
1. Find the target node (passed as a reference to the parent's pointer)
2. Delete the node
3. Set child pointer of parent to NULL

Case 1: No Child

Delete 7



Remove(10)

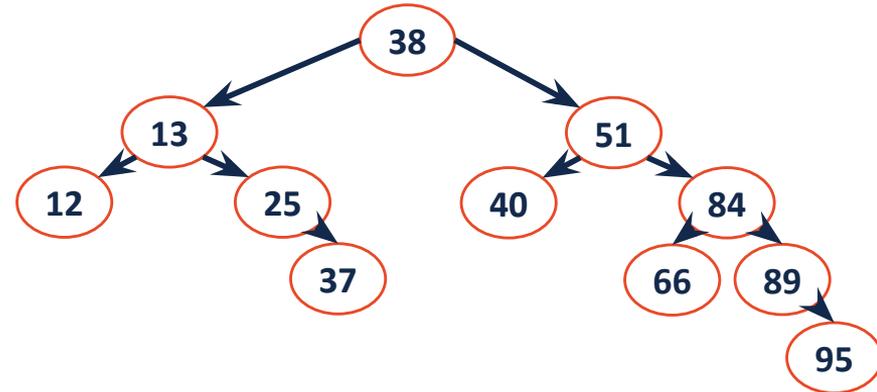
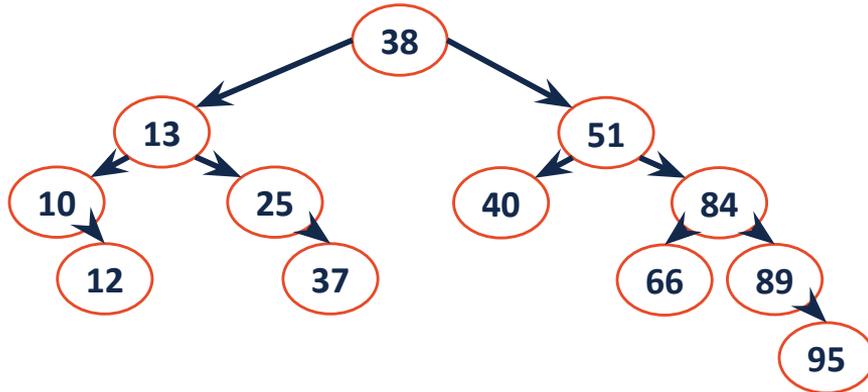


Remove

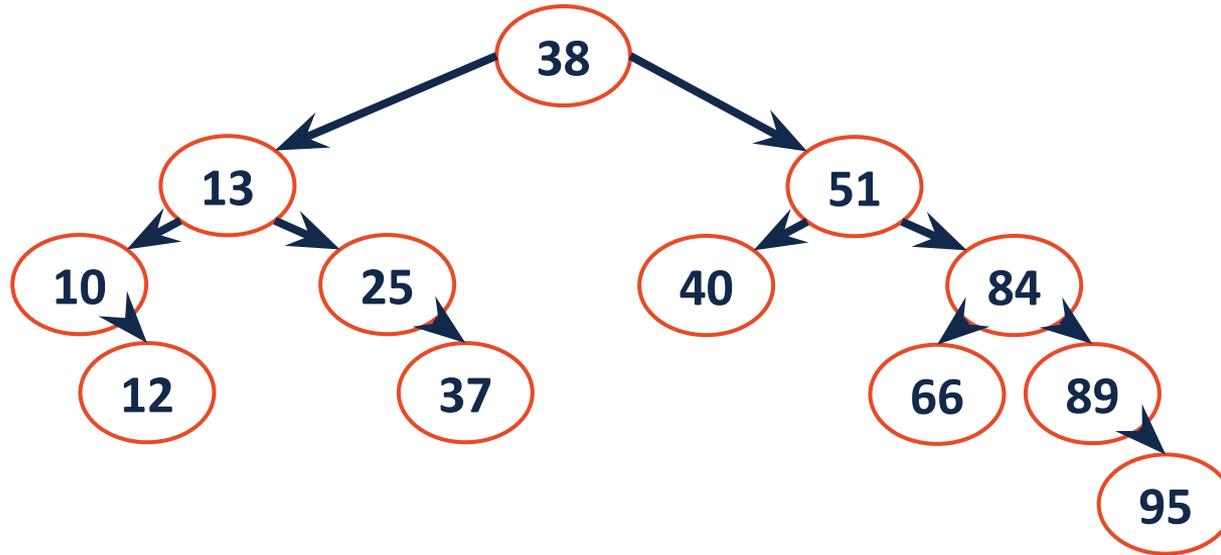
Case 2 (1 child):

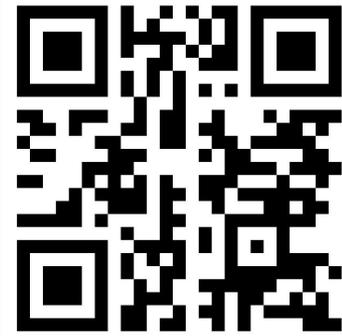
1. Find the target node (passed as a reference to the parent's pointer)
2. Make a temporary pointer to the target
3. Set the parent's node to the target's child
4. Delete the target

Remove(10)



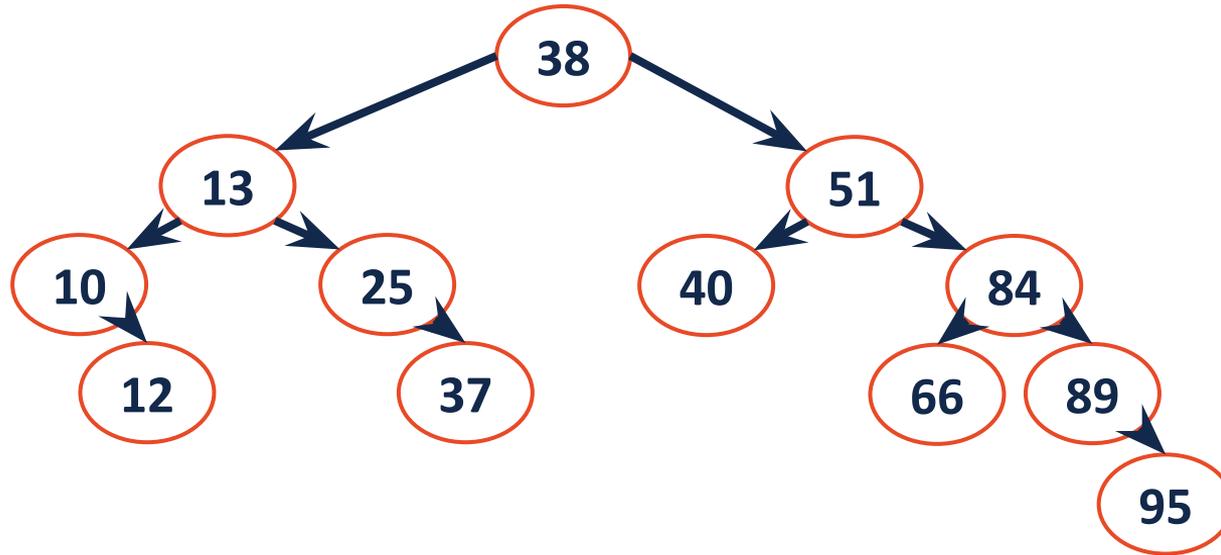
Remove(38)





Join Code: 225

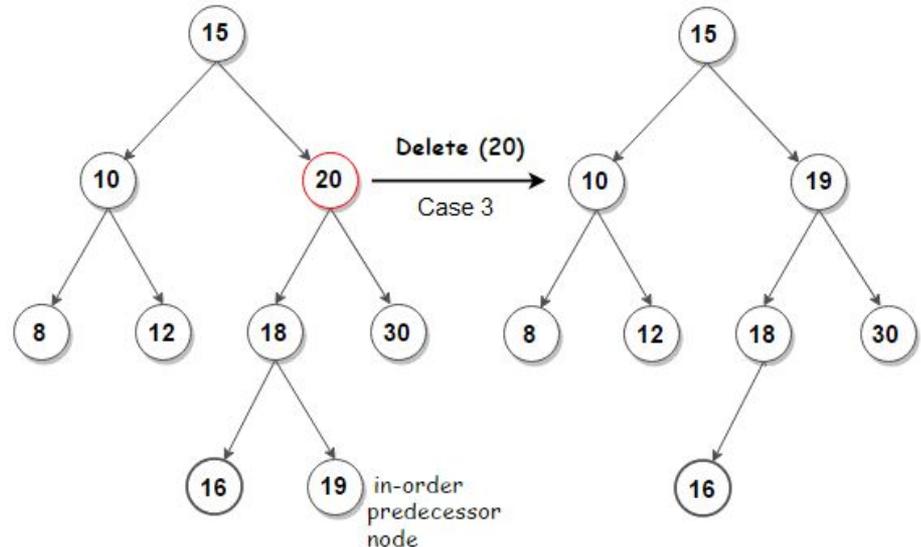
Remove(38) - What node(s) could take 38's place?



Remove

Case 3 (2 children):

1. Find the target node (passed as a reference to the parent's pointer)
2. Find the target's In Order Predecessor (IOP)
3. Swap the target with the IOP
4. Recursively call on the target's new location



Dictionary ADT

Insert (key, value)

Remove (key)

value Find (key)



Operation	BST	Binary Tree	Sorted Array	Unsorted Array	Sorted Linked-List	Unsorted Linked-List
find						
insert						
delete						
traverse						

Fill out the Big O for each of these functions!



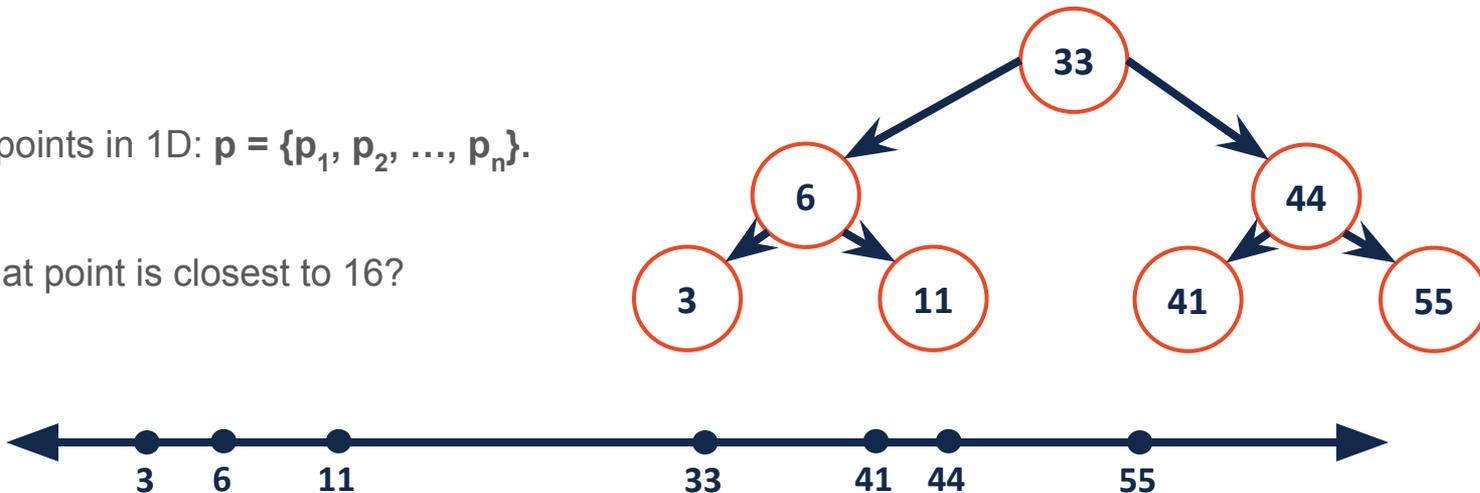
Closest Element

BSTs are useful structures for range-based and nearest-neighbor searches.

Q: Consider points in 1D: $\mathbf{p} = \{p_1, p_2, \dots, p_n\}$.

...what point is closest to 16?

Ex:



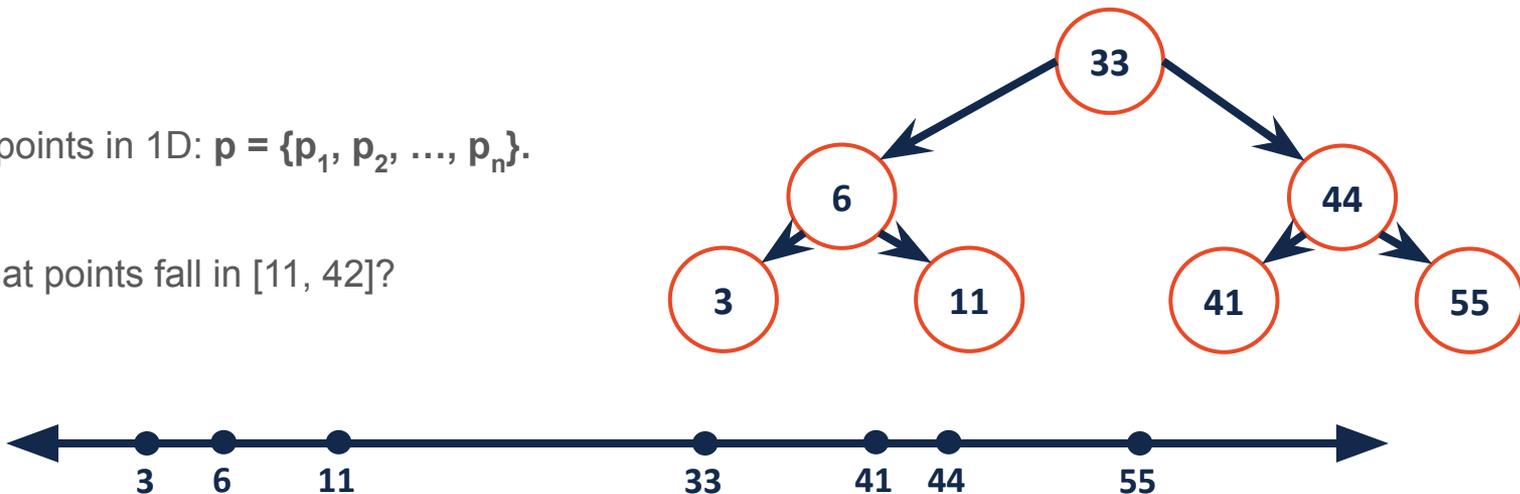
Range-based Searches

Balanced BSTs are useful structures for range-based and nearest-neighbor searches.

Q: Consider points in 1D: $\mathbf{p} = \{p_1, p_2, \dots, p_n\}$.

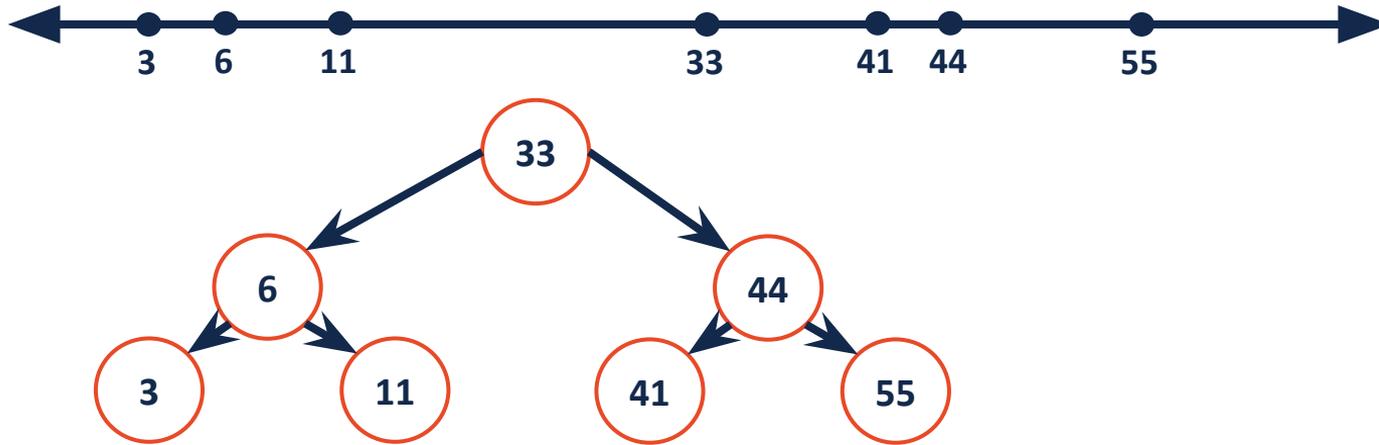
...what points fall in $[11, 42]$?

Ex:



Range-based Searches

Query: [11,42]



Red-Black Trees in C++ (map)

iterator std::map<K, V>::lower_bound(const K &);

- Returns an iterator pointing to the first element in the container whose key is not considered to go before k (i.e., either it is equivalent or goes after).

iterator std::map<K, V>::upper_bound(const K &);

- Returns an iterator pointing to the first element in the container whose key is considered to go after k .



Example

Given a set of points:



lower_bound(12)?

lower_bound(18)?

lower_bound(14)?

upper_bound(40)?

upper_bound(44)?

upper_bound(45)?

