Binary Search Tree

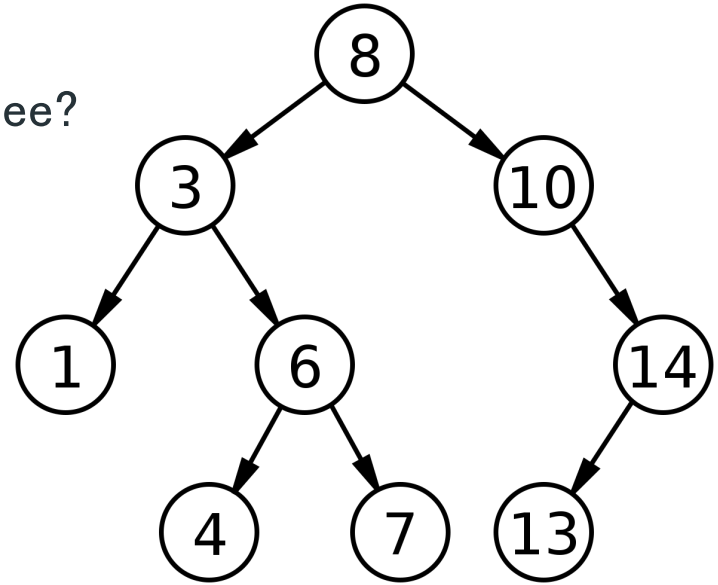
CS112 Recitation

Coding Files:
https://drive.google.com/drive/folders/1v1vauUBGWSUTpb3W0DVrXQc3OkIsH-zb?usp=sharing

Hoai-An Nguyen
hnn14@scarletmail.rutgers.edu

Warm-up 1

1. In the worst case, how long does it take to search in a binary search tree? Why?
2. What about the best case?
   a. What is the structure of that tree?



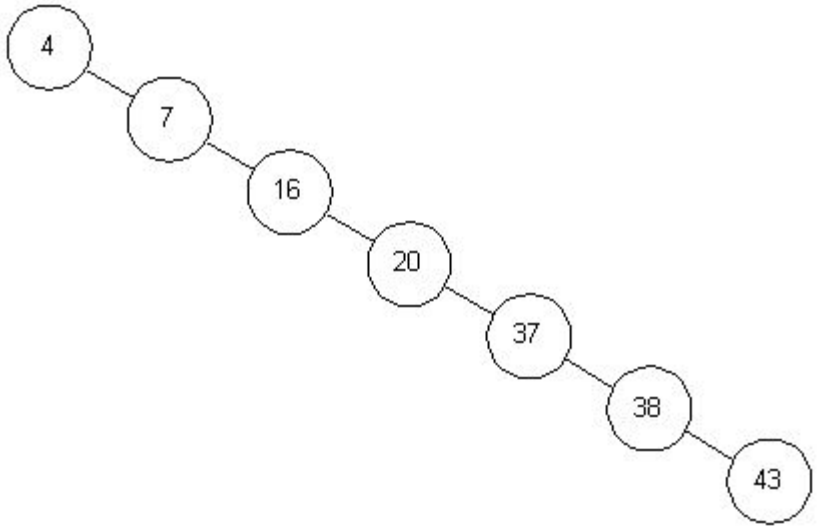
Warm-up 1 Answer

1. In the worst case, how long does it take to search in a binary search tree? Why?

A: O(n), potentially your tree could look like the following (there is no guaranteed balancing)

For example, if you build your tree in order using the following array:

[7, 6, 5, 3, 1] or [2. 6. 8. 9]



Warm-up 2

1. Suppose that we have an estimate ahead of time of how often search keys are to be accessed in a BST, and the freedom to insert them in any order that we desire. Should the keys be inserted into the tree in increasing order, decreasing order of likely frequency of access, or some other order? Explain your answer.

Warm-up 2 Answer

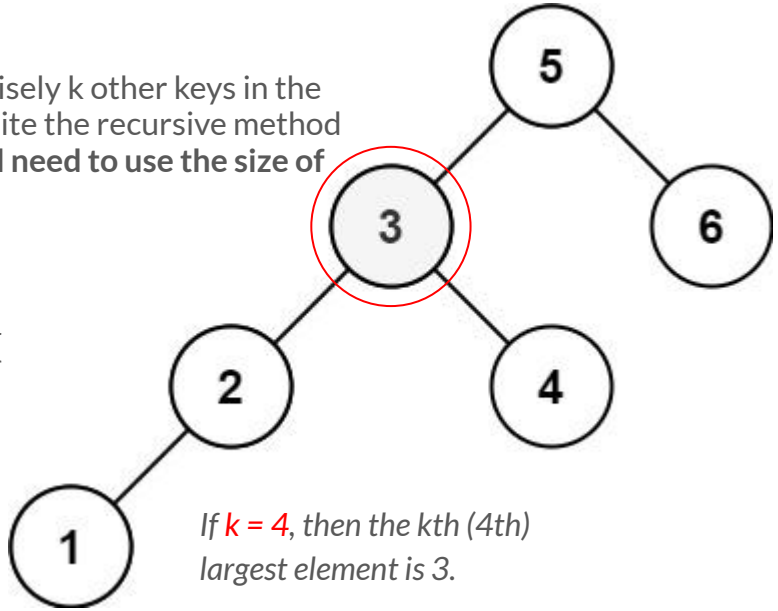
- The keys should be inserted into the tree in decreasing order of likely frequency of access.
- This will make the keys with high frequency of access stay above in the tree than keys with a low frequency of access, requiring less compares to be found.

Kth Largest Element in a BST

You have seen **select(k)** in class, which finds the key such that precisely k other keys in the BST are smaller. Assuming the following Node implementation, write the recursive method **kthLargest** that finds the kth largest item in the BST. **Hint: You will need to use the size of the right subtree!**

Use the following Node class:

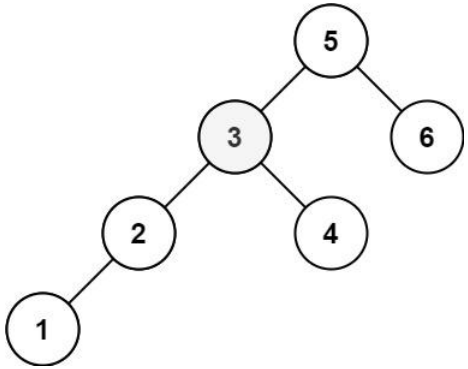
```
public class Node <Key extends Comparable<Key>,Value> {
        Key key;  // key
        Value val; // associated value
        Node left, right;  // links to subtrees
        int n;  // number of nodes in subtree rooted here
}
```



If k = 4, then the kth (4th) largest element is 3.

Use the header:

```
public Key kthLargest( Node root, int k )
```

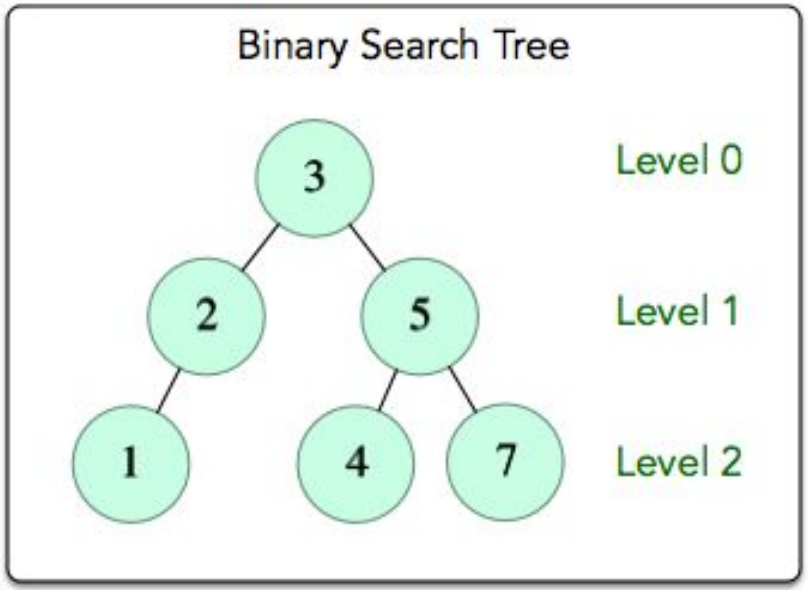

5
3
6
2
4
1

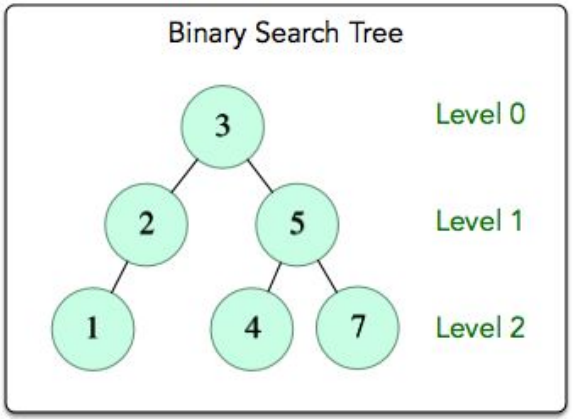
Kth Largest Element in a BST Solution

```
1   public Key kthLargest( Node root, int k) {
2       int rightCount = size(root.right);
3       if (rightCount == (k-1)) {
4         return root.key;
5       }
6       if (rightCount >= k) {
7         return kthLargest(root.right, k);
8       }
9       return kthLargest(root.left, k-rightCount-1);
10  }
11
12  public int size(Node root){
13    if(root == null){
14      return 0;
15    }
16    return 1 + size(root.left) + size(root.right);
17  }
```

BST - Level-Order Traversal

Write a method printLevel() that takes a Node as argument and prints the keys in the subtree rooted at that node in level order (in order of their distance from the root, with nodes on each level in order from left to right). Hint: Use a Queue.




Binary Search Tree
3
2
5
1
4
7
Level 0
Level 1
Level 2

BST - Level-Order Traversal Solution

```
private void printLevel (Node node) {
    Queue<Node> queue = new Queue<Node>();
    queue.enqueue(node);

    while (!queue.isEmpty()) {
        Node current = queue.dequeue();
        System.out.print(current.key + " ");

        if (current.left != null) {
            queue.enqueue(current.left);
        }

        if (current.right != null) {
            queue.enqueue(current.right);
        }
    }
}
```

Good Work!

Go to https://dynrec.cs.rutgers.edu/live/

Enter the Quiz Code: