

Recursive Description: root and subtree.
 Each subtree is itself a valid tree.
 A tree with zero subtrees is a leaf.

Relative Description: node and value.
 Each node has a value. A parent node is joined with an edge to each child node.

Tree Abstraction

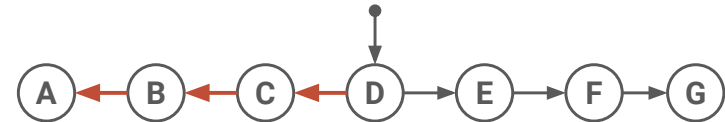
We oftentimes abuse the terminology a bit by saying things like, “each parent is the sum of its children”.

?: What does the “root node” refer to? What does the “root value” refer to?

Optimization: Move Entry Point, Flip Links

Problem: Search is slow even if we spend extra time adding keys to their sorted position.

Solution 2: Move the entry pointer to the middle. Flip the left links.



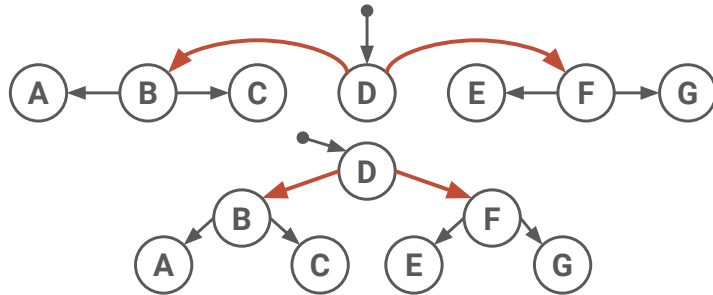
?: How does this change affect the worst-case search time?

?: How can we improve this optimization?

Optimization: Move Entry Point, Flip Links, Use Longer Hops

Problem: Search is slow even if we spend extra time adding keys to their sorted position.

Solution 2: Move the entry pointer to the middle. Flip the left links. Use longer hops.



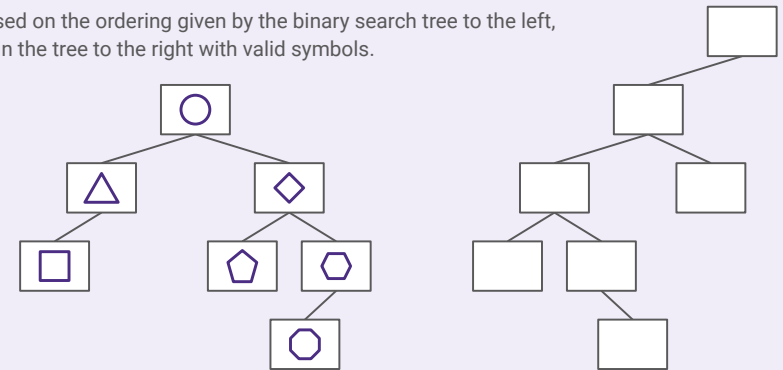
9

We saw this pattern of recursive subdivision in merge sort, and it's here again!

?: What is the worst-case search time?

Q Order Theory

Based on the ordering given by the binary search tree to the left, fill in the tree to the right with valid symbols.



10

?: Binary search trees are related to OrderedLinkedSets. What do we know about the relationship between the square symbol and the triangle symbol?

Q1: Based on the ordering given by the binary search tree to the left, fill in the tree to the right with valid symbols.

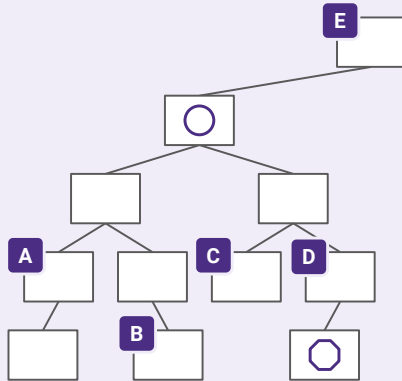
Q Applying Order Theory

Say we have the following total order.



Assume that there are several other symbols not shown above.

In which of the five labeled nodes can the pentagon symbol  reside?



12

Binary Search Tree Invariant

Say we have the following total order.



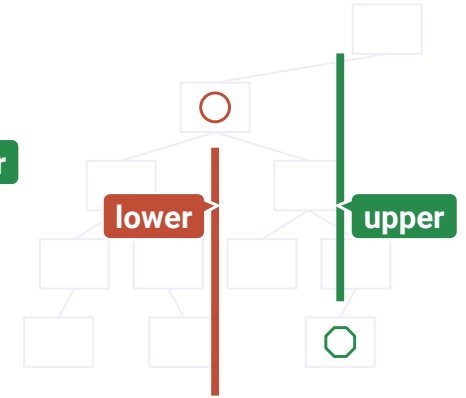
lower

upper

Binary Search Tree Invariant.

For every node X in the tree:

- All keys in the left subtree $<$ X's key.
- All keys in the right subtree $>$ X's key.



14

Q1: In which of the five labeled nodes can the pentagon symbol reside?

?: If we search a left subtree, how does that change the **lower** limit on the keys? The **upper** limit?

?: If we search a right subtree, how does that change the **lower** limit on the keys? The **upper** limit?

Key Comparison

Formally, ordering must be complete, transitive, and antisymmetric.

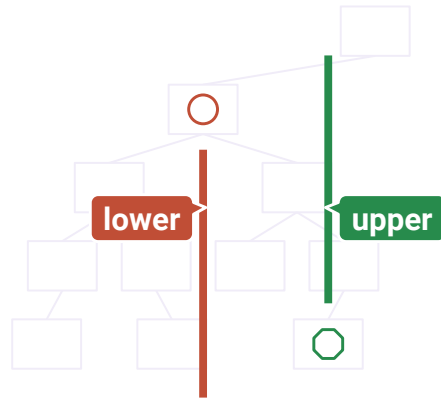
Given keys p and q :

Exactly one of $p < q$ and $q < p$ are true.

$p < q$ and $q < r$ imply $p < r$.

One consequence of these rules:

No duplicate keys allowed!



15

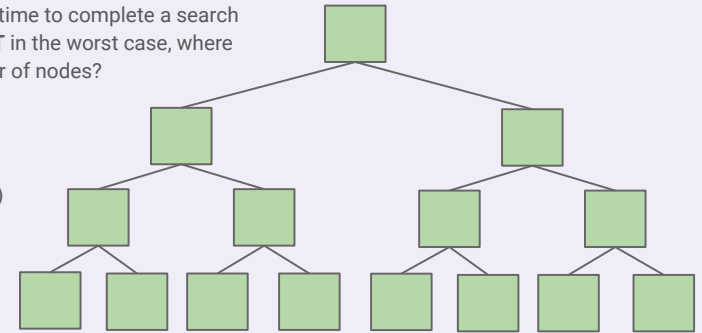
?: What is the purpose of this formal definition of key comparison? How do we apply these rules to numbers vs. arbitrary objects?

?: How might we allow duplicate keys in our binary search tree in spite of these rules? What are the potential problems that arise?

Q Search Algorithm Analysis

What is the runtime to complete a search on a **bushy BST** in the worst case, where N is the number of nodes?

- A. $\Theta(\log N)$
- B. $\Theta(N)$
- C. $\Theta(N \log N)$
- D. $\Theta(N^2)$
- E. $\Theta(2^N)$



18

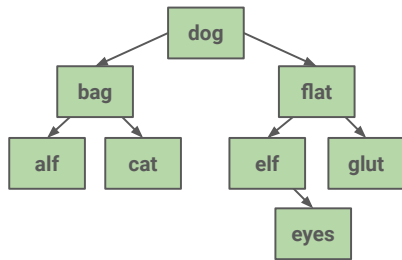
We don't yet have a formal definition for the concept of bushiness. Use the example as a visual aid.

Q1: What is the runtime to complete a search on a **bushy BST** in the worst case, where N is the number of nodes?

?: What is the best case runtime?

Adding a New Key

Check if the tree already has the key. If found, do nothing. Else, create a new node and set the appropriate reference.



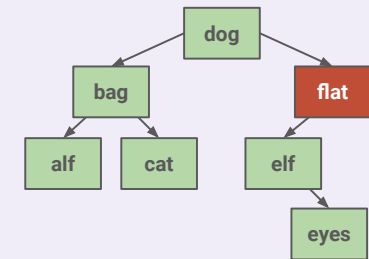
```
static BST add(BST T, Key ik) {  
    if (T == null)  
        return new BST(ik);  
    if (ik < T.key)  
        T.left = add(T.left, ik);  
    else if (ik > T.key)  
        T.right = add(T.right, ik);  
    return T;  
}
```

21

Q Removing: One Child

Remove the key containing the value **flat**.

What simple modification can we make to the tree to remove the value **flat**?



24

You might sometimes see code that exhibits “arm’s-length recursion.” Consider these two unnecessary base cases.

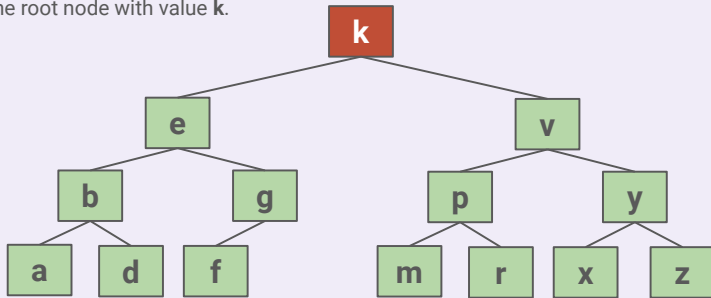
```
if (T.left == null)  
    T.left = new BST(ik);  
else if (T.right == null)  
    T.right = new BST(ik);
```

?: How does the code given in the slide handle the arm’s-length recursion scenario?

Q1: What simple modification can we make to the tree to remove the value **flat**?

Q Key Removal Challenge

Delete the root node with value k.



26

Q1: Delete the root node with value k.

Implementer's Design Decision Hierarchy

Abstract Data Type

Which ADT is the best fit?

Set

Map

Data Structure

Which data structure offers the best performance for our input/workload?

Binary Search Tree

Linked Nodes

Implementation Details

How do we maintain invariants?

For every node X in the tree:
All keys in the left subtree $<$ X's key.
All keys in the right subtree $>$ X's key.

35

As the ADT implementer, we always had to keep in mind our invariants when thinking through the problem.

?: How does the Binary Search Tree Invariant affect the implementation of contains, add, and remove?