Programming Paradigms

CT331 Week 8 Lecture 2

Finlay Smith

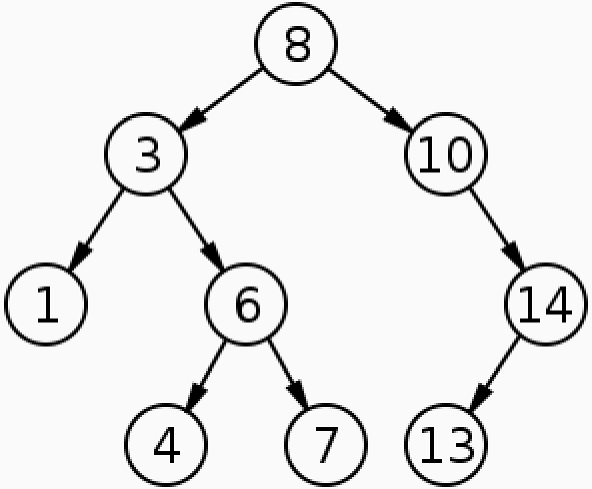
Finlay.smith@universityofgalway.ie

Binary Search Tree (BST)

Binary Search Trees

Data structure to store an ordered set of items.

- Fast lookup
- Fast addition and removal of items
- Can be used to implement:
  - Dynamic sets of items
  - Lookup tables that allow finding an item by its key



Binary Search Tree rules

Three rules for all BSTs:

1. No duplicates
2. Maximum of 2 child nodes.
3. Left child is always smaller than parent.
   (and right is always larger)

Binary Search Trees

Each child, or leaf node is also a BST.

We call these sub-trees.

Smallest possible sub-tree: null

Binary Search Trees

Binary search trees keep their keys in sorted order, so that lookup and other operations can use the principle of binary search.

When looking for a key in a tree (or a place to insert a new key):

Traverse the tree from root to leaf, making comparisons to keys stored in the nodes of the tree and deciding, based on the comparison, to continue searching in the left or right subtrees.

Binary Search Trees

On average, this means that each comparison allows the operations to skip about **half of the tree.**

- Much better than linear search through unsorted list
- Not as fast as a hash table operations.

Binary Search Trees: Operations

- Searching
- Insertion
- Deletion
- Traversal
- Verification

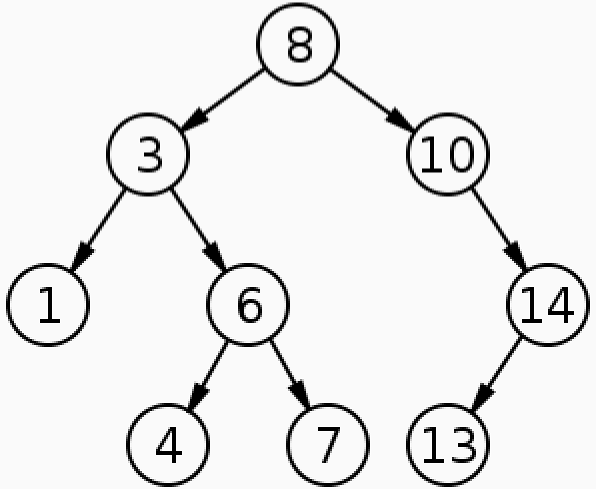
Use the following slides as algorithms for the assignment

Binary Search Trees: Search

Find the node with value 6:

Start at head (8)

1. Is current node = null?
    a. Y: item not in tree
    b. N: continue
2. Is current node = 6?
    a. Y: Return, item found.
    b. N: Continue
3. Is 6 < current node?
    a. Y: Recurse on left child/leaf
    b. N: Recurse on right child/leaf

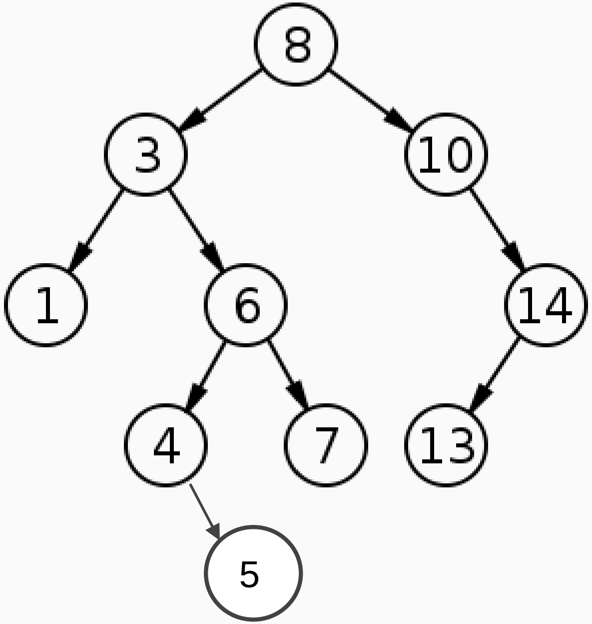


Binary Search Trees: Insert

Insert the value: 5

Start at head (8)

1. Is current node = null?
   a. Y: set current node to 5, return.
   b. N: continue
2. Is current node = 5?
   a. Y: Return, item found.
   b. N: Continue
3. Is 5 < current node?
   a. Y: Recurse on left child/leaf
   b. N: Recurse on right child/leaf

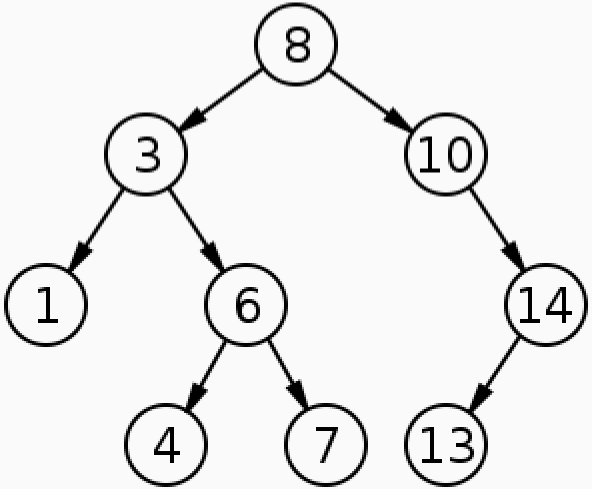


Binary Search Trees: Delete

Delete the node with value 3:

Start at head (8)

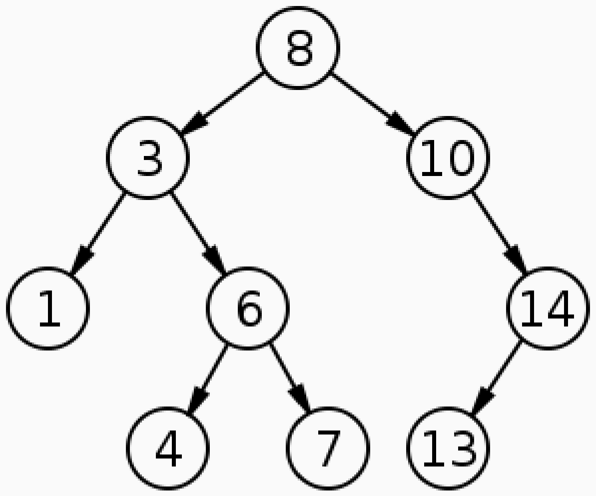
1. Is current node = null?
   a. Y: item not in tree
   b. N: continue
2. Is current node = 3?
   a. Y: Return, item found.
   b. N: Continue
3. Is 3 < current node?
   a. Y: Recurse on left child/leaf
   b. N: Recurse on right child/leaf



Binary Search Trees: Delete

Deletion is not so simple…

- If node has no children:
    - Just set to null.
- If node has 1 child:
    - Set node value to child value
    - Delete child.
- If node has 2 children:
    - ???



Binary Search Trees: Delete

If node has 2 children:

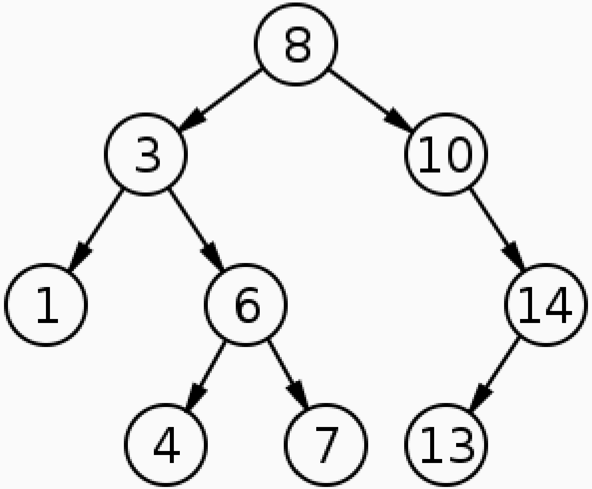
1. Find in-order successor (IOS).

   (the next highest value in the tree)

   (the smallest value on the right sub-tree)

2. Copy IOS to node

   Delete old IOS.

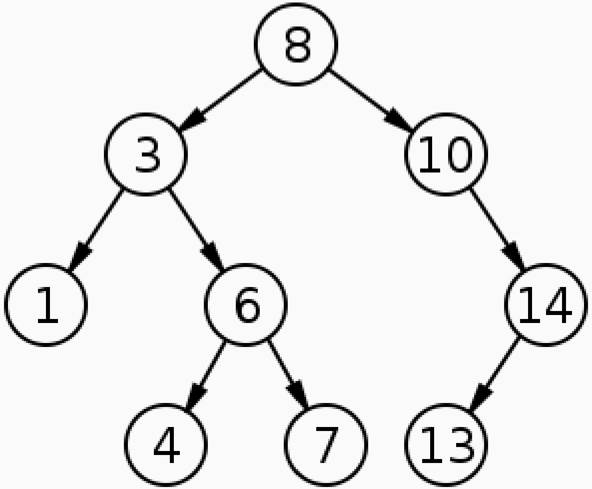


Note: you can also use the in-order predecessor, or the next lowest value, the highest value on the left sub-tree.

Binary Search Trees: Traversal

Once the binary search tree has been created, its elements can be retrieved in-order by:

- Recursively traversing the left subtree of the root node.
- Accessing the node itself
- Then recursively traversing the right subtree of the node



Binary Search Trees in Scheme

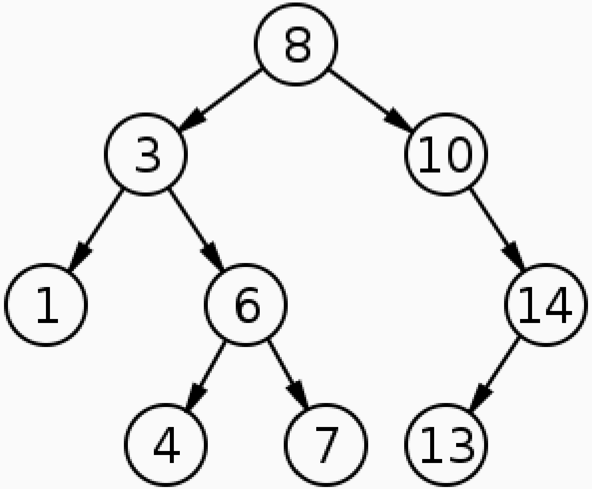
How do we represent a BST in scheme?

Minimum sub-tree:

Null or ‘()

Typical sub-tree:

(Left child, value, right child)

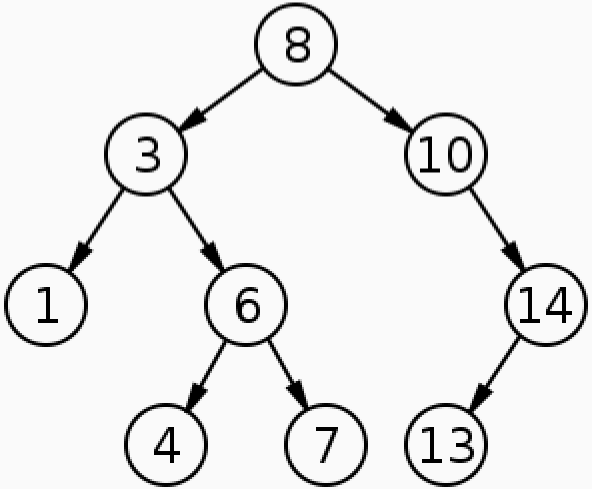


Binary Search Trees in Scheme

How do we get the left child?

How do we get the right child?

How do we get the value?



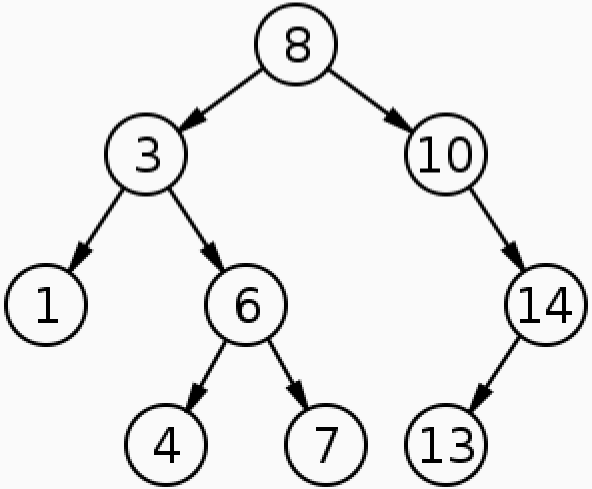
Binary Search Trees in Scheme

How do we get the left child?

(car sub-tree)

How do we get the right child?

How do we get the value?



Binary Search Trees in Scheme

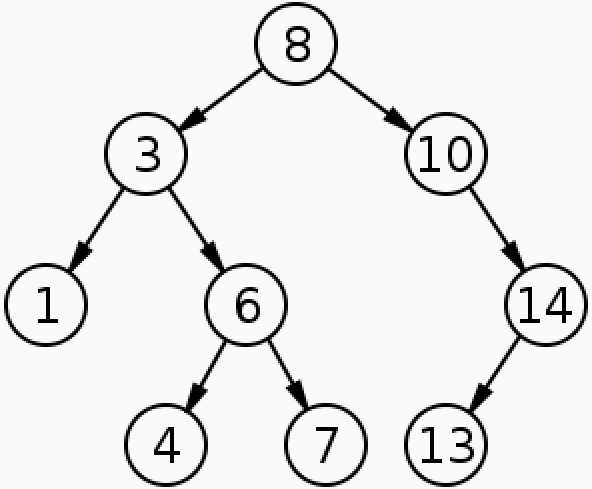
How do we get the left child?

(car sub-tree)

How do we get the right child?

(caddr sub-tree)

How do we get the value?



Binary Search Trees in Scheme

How do we get the left child?

(car sub-tree)

How do we get the right child?

(caddr sub-tree)

How do we get the value?

(cadr sub-tree)