Searching in BST

Nimra Javed(SP22-BSCS-065)

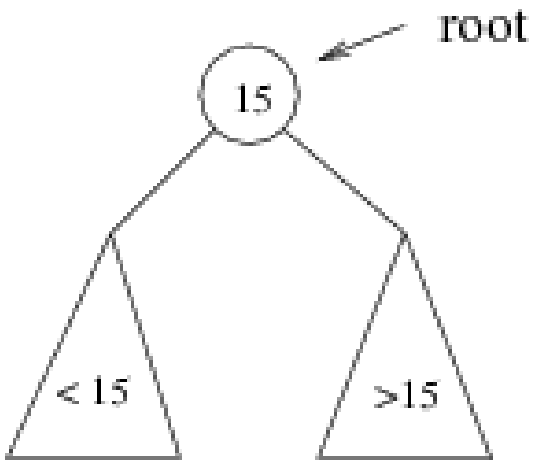
BST

Finding or locating some specific node within a tree.

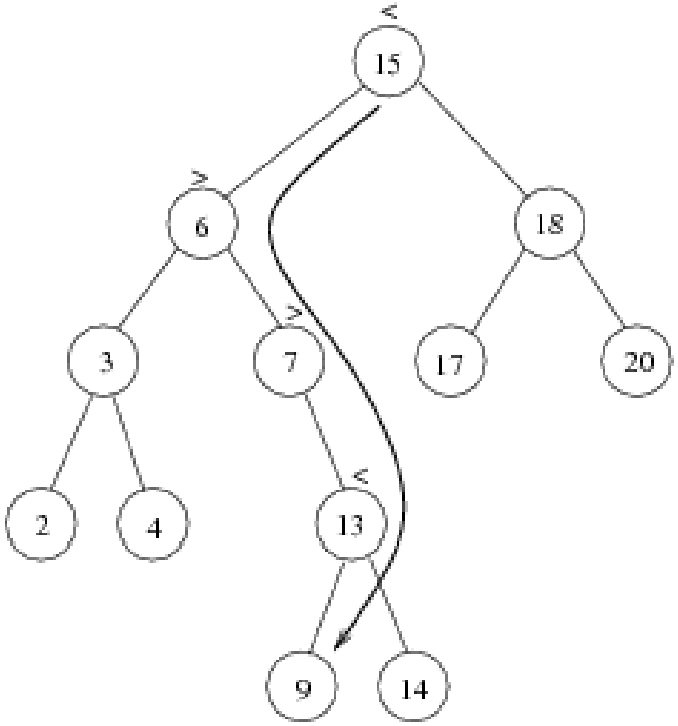
- Compare the element with the root of the tree.
- If the key is matched then return the location of the node.

Searching BST

- If we are searching for 15, then we are done.
- If we are searching for a key < 15, then we should search in the left subtree.
- If we are searching for a key > 15, then we should search in the right subtree.



Example: Search for 9 ...

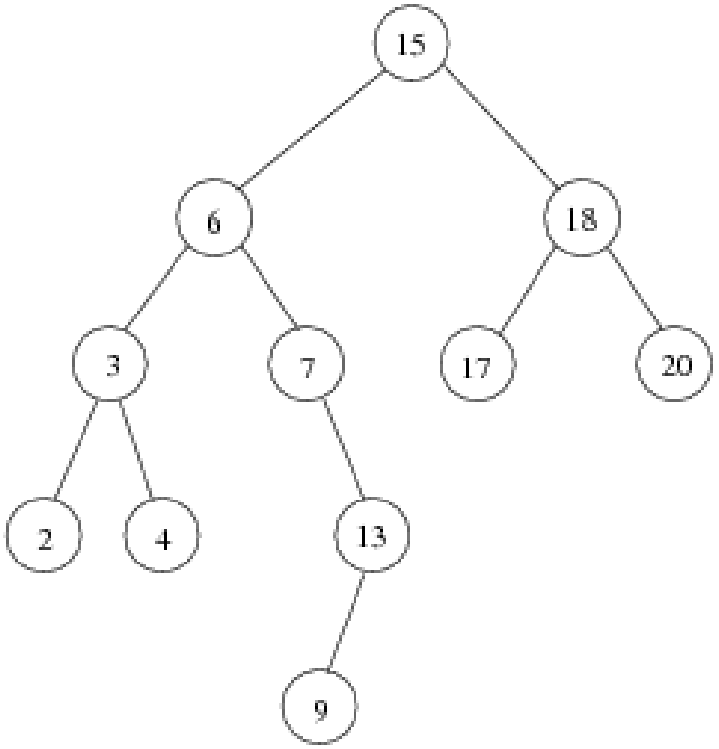


Search for 9:

1. compare 9:15(the root), go to left subtree;
2. compare 9:6, go to right subtree;
3. compare 9:7, go to right subtree;
4. compare 9:13, go to left subtree;
5. compare 9:9, found it!

Inorder traversal of BST

- Print out all the keys in sorted order



Inorder: 2, 3, 4, 6, 7, 9, 13, 15, 17, 18, 20

findMin/ findMax

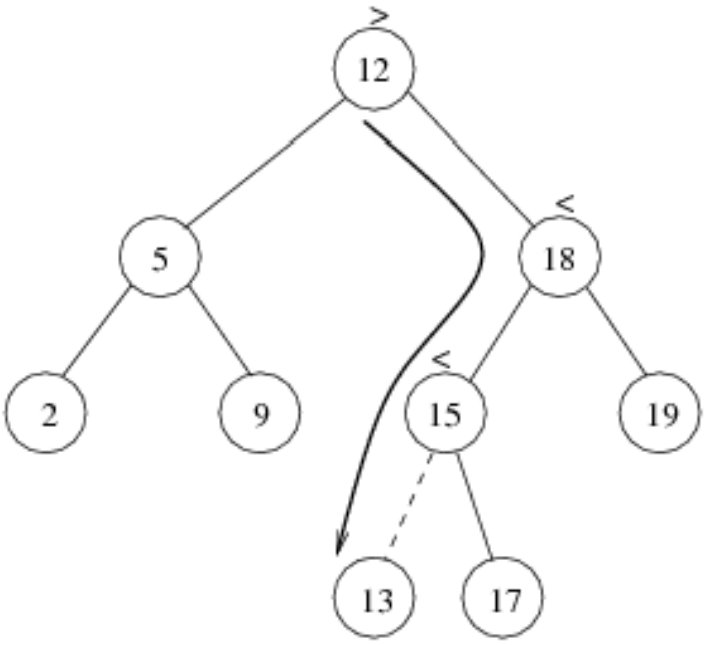
- Return the node containing the smallest element in the tree
- Start at the root and go left as long as there is a left child. The stopping point is the smallest element

```
template <class Comparable>
BinaryNode<Comparable> *
BinarySearchTree<Comparable>::findMin( BinaryNode<Comparable> *t ) const
{
    if( t == NULL )
        return NULL;
    if( t->left == NULL )
        return t;
    return findMin( t->left );
}
```

- Similarly for findMax
- Time complexity = O(height of the tree)

insert

- Proceed down the tree as you would with a find
- If X is found, do nothing (or update something)
- Otherwise, insert X at the last spot on the path traversed



- Time complexity = O(height of the tree)

delete

- When we delete a node, we need to consider how we take care of the children of the deleted node.
  - This has to be done such that the property of the search tree is maintained.

delete

Three cases:

(1) the node is a leaf

- Delete it immediately

(2) the node has one child

- Adjust a pointer from the parent to bypass that node

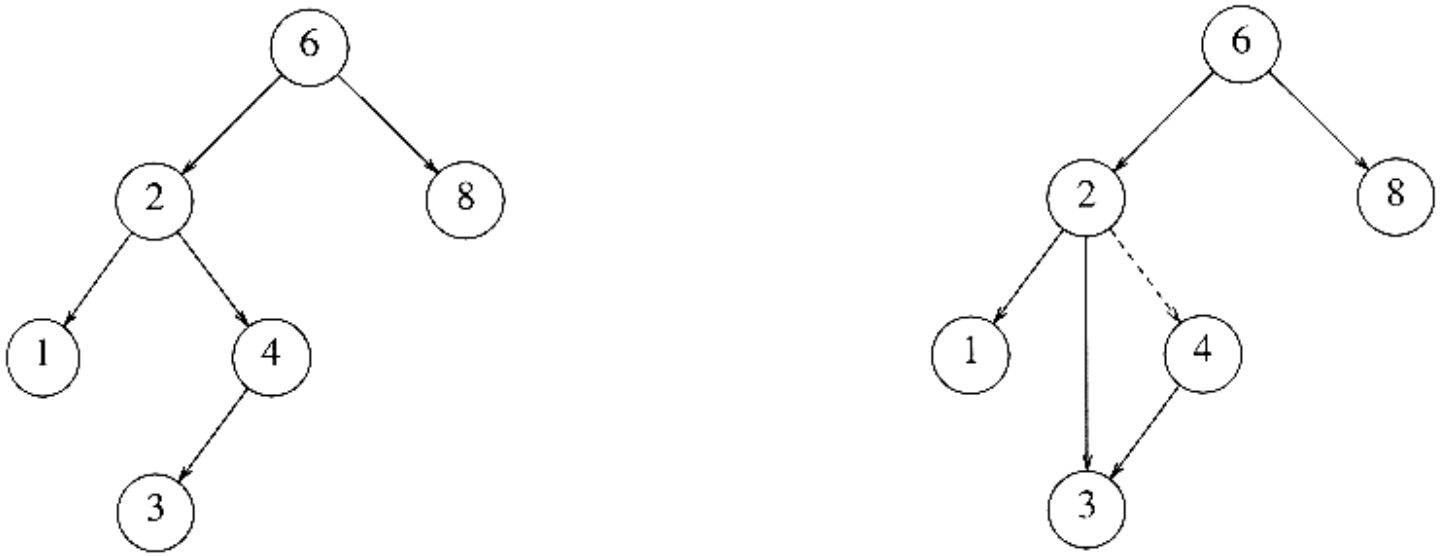


Figure 4.24 Deletion of a node (4) with one child, before and after

delete

(3) the node has 2 children

- replace the key of that node with the minimum element at the right subtree
- delete the minimum element
  - Has either no child or only right child because if it has a left child, that left child would be smaller and would have been chosen. So invoke case 1 or 2.

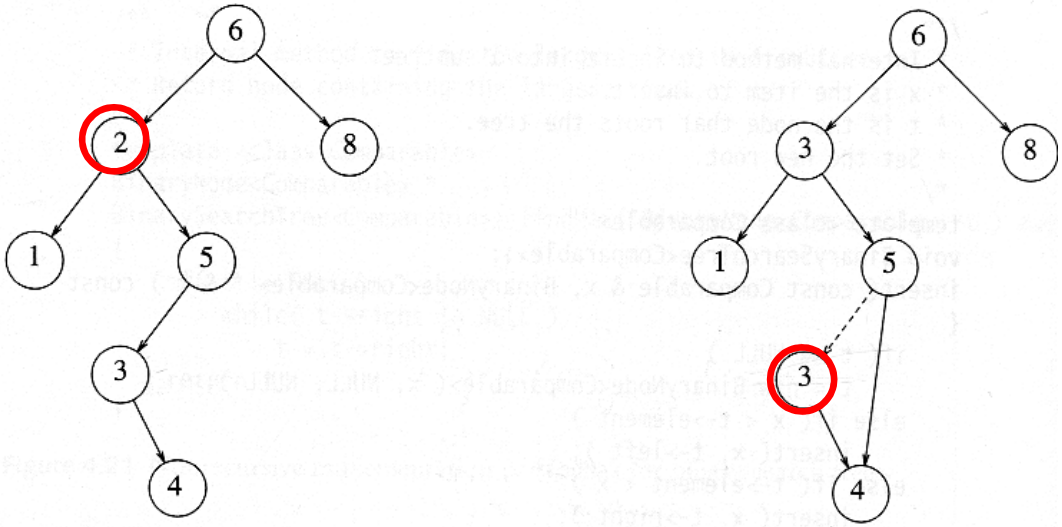


Figure 4.25 Deletion of a node (2) with two children, before and after

Thanks