Lab 7: Binary Search Tree and its Applications

Objectives

- Understand the concept of binary search tree data structure and its implementation.
- Understand the basic operations of binary search tree.
- Write simple applications that use binary search tree.

Review

- A **binary search tree** is a data structure that is designed for quick searching.
- A BST has the following properties
  - For each node, all elements of its left subtree must be less than or equal to the root
  - For each node, all elements of its right subtree must be greater than the root

Basic Operations

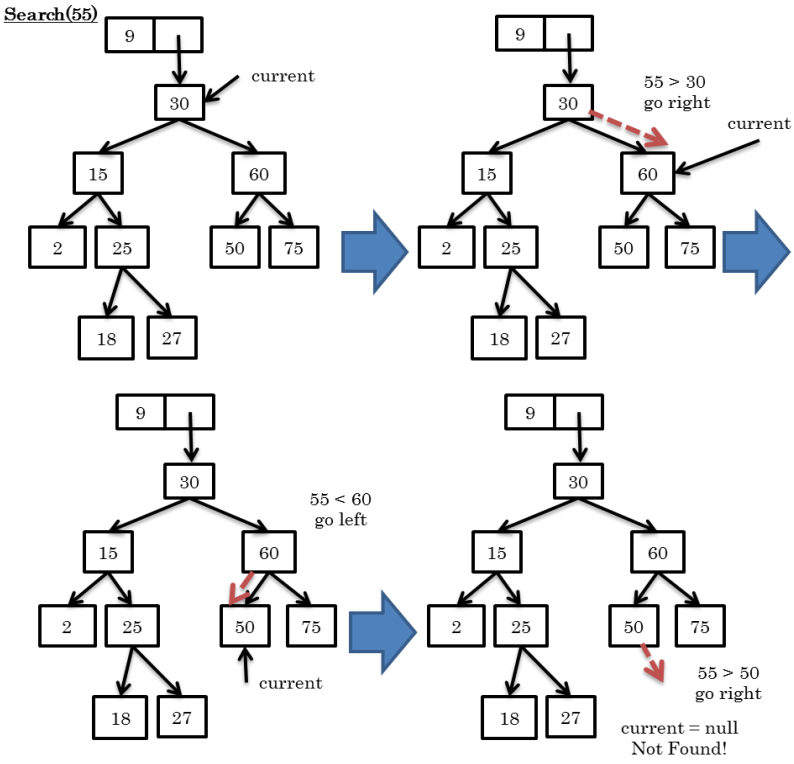
boolean search(T key) : find a specified key in a BST

Step 1: set current pointer at root

Step 2: If the key is less than current, make current now points to the left child of current.

If the key is greater than current, make current now point to the right child of current.

If the key is equal to current, stop and Repeat steps 1 and 2.

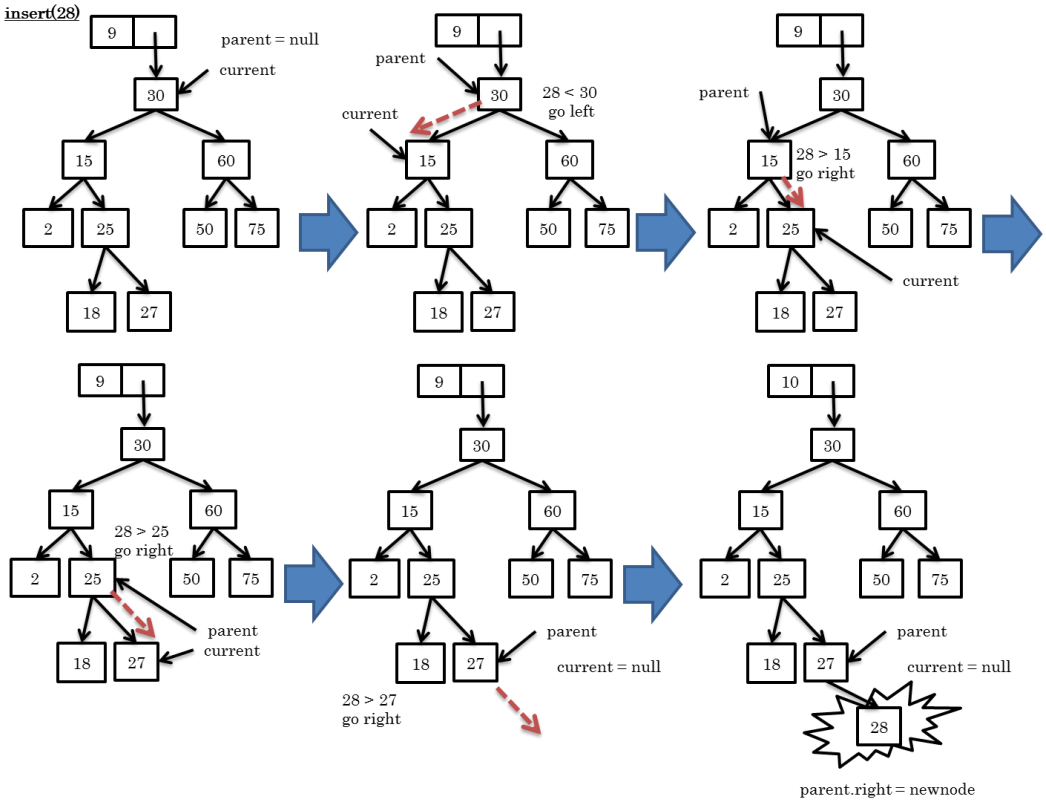


Searching for 55 in a binary search tree with 9 nodes

void insert(T key): insert a key into a BST

Step1: find location of the parent of the key

Step2: add the new node in the appropriate location of the parent found in step 1



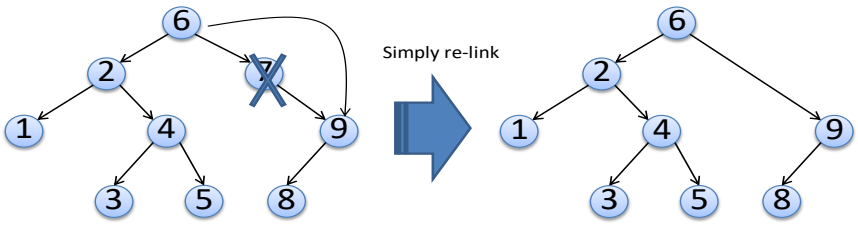
Insertion of 28 to a binary search tree with 9 nodes

Deletion

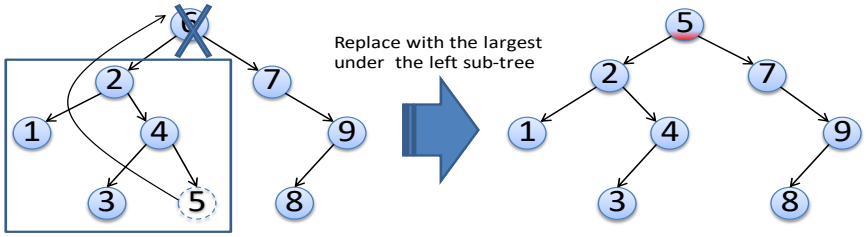
T delete(T key): To delete a key from a binary search tree,

1. The node containing the key has no children. In this case, all we need to do is delete the node.
2. The node containing the key has only one child. We delete the node and attach its only subtree to the deleted node's parent
3. The node containing the key has both left and right subtrees. Find the largest node from the left subtree of the node containing the key. This node is so called maxleft. Replace the node containing the key with maxleft's value, and then remove maxleft from the tree.

Deleting a node with one child



Deleting a node with two children



void FindSmallest() : find the smallest element in a BST

As the smallest element is always located in the left most of the tree, we can find it by keep traversing the tree from the root to the left repeatedly until the node no longer has a left child.

void Findlargest() : find the largest element in a BST

As the largest element is always located in the right most of the tree, we can find it by keep traversing the tree from the root to the right repeatedly until the node no longer has a right child.

Exercises

Exercise 1 Implement void insert(T newdata). This method inserts a new data to a suitable place in the BST. Test the result in the main() method. **Hint:** To compare two comparable generic-type variables x and y, use x.compareTo(y).

Exercise 2 Implement delete(T data). This method deletes an element from the binary tree and returns the element in the node that is deleted. Use the given template source code to complete this exercise.

Exercise 3 Implement search(T searchedData). This method returns true if the searched data is found and return false otherwise. The search method is similar to finding locations for insertion in the previous exercises.

Exercise 4 Implement findSmallest() that returns the BTNode<T> containing smallest data in the binary search tree. If the tree is empty, return null.

Exercise 5 Implement findLargest() that returns the BTNode<T> containing largest data in the binary search tree. If the tree is empty, return null.