15CSE201 : Data Structures and Algorithms

Lecture 12 :Problem solving session

Ritwik M

Based on the reference materials by Prof. Goodrich and Dr. Vidhya Balasubramanian

Topics Covered

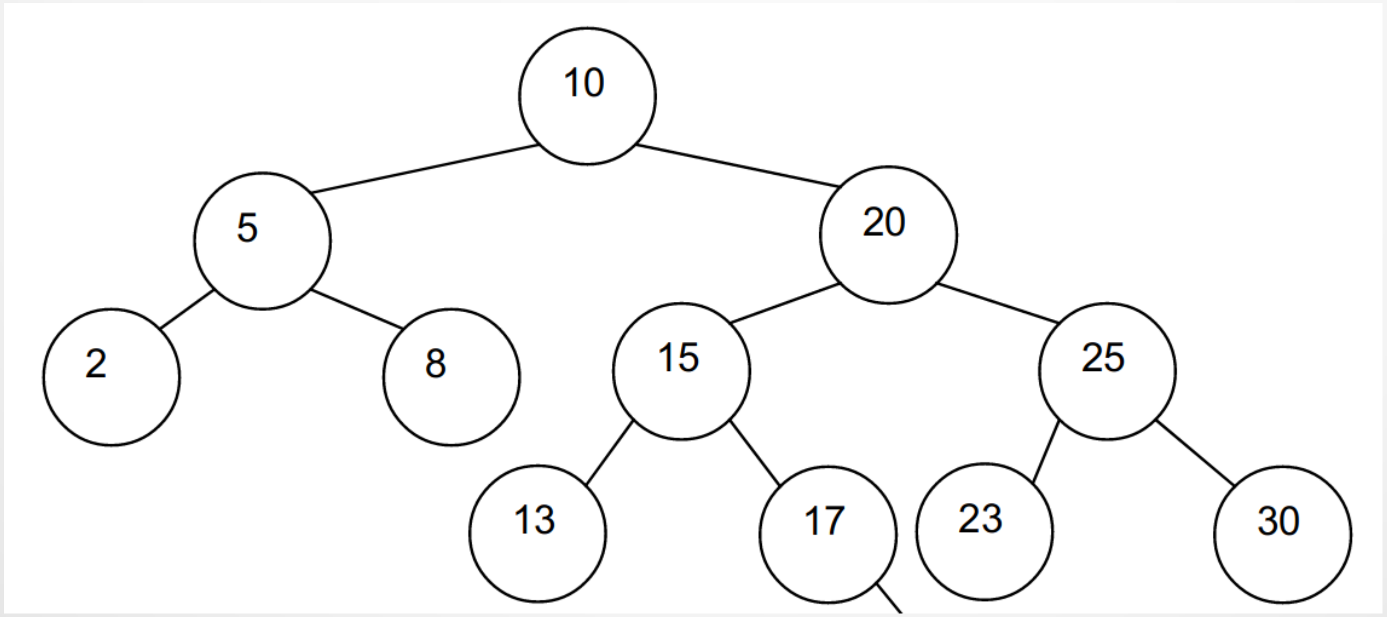
- Height Balanced Trees
- Binary Search Trees

Problems

1. What 3 properties must an AVL tree have?
2. Draw an AVL tree of height 4 that contains the minimum possible number of nodes.
3. In a typical BST, inserting keys in order result in a worst-case height. Show the result when an initially empty AVL tree has keys 1 through 7 inserted in order.

More Problems

4. Insert 18 into the following AVL tree. What type of imbalance does it cause? Show the result after balancing.



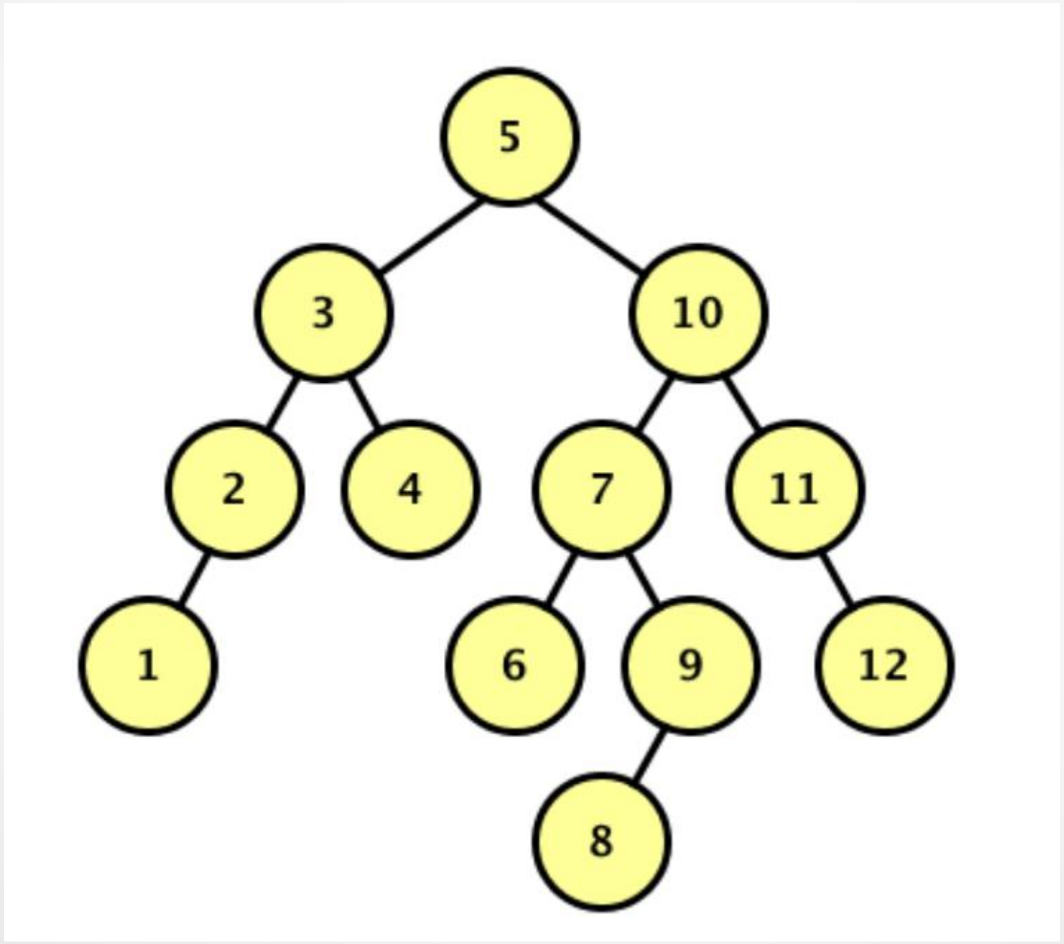
Some more problems

5. Insert the following keys, in order, into an initially empty AVL tree: 12, 8, 9, 20, 10, 15, 3,11, 5

6. Given a binary search tree, describe how you could convert it into an AVL tree with worst-case time O(nlogn) and best case O(n).

Even More Problems

7. Given the following AVL Tree



Problems Continued

7.a. Draw the resulting BST after 5 is removed, but before any rebalancing takes place. Label each node in the resulting tree with its balance factor. Replace a node with both children using an appropriate value from the node's left child

7. b. Now rebalance the tree that results from (a). Draw a new tree for each rotation that occurs when rebalancing the AVL Tree (you only need to draw one tree that results from an RL or LR rotation). You do not need to label these trees with balance factors.

Still More Problems

8. Insert the following sequence of elements into an AVL tree, starting with an empty tree: 10, 20, 15, 25, 30, 16, 18, 19. Delete 30 in the AVL tree that you got.

9. Given an AVL tree T, is it always possible to build the same tree by a sequence of BST-insert and delete operations (with no rotations)?

Problems Galore

10. Design an ADT containing integers that supports the following actions. Explain how these actions are implemented.

Init()	Initialize the ADT	O(1)
Insert(x)	Insert x into the ADT, if it is not in ADT yet	O(log n)
Delete(x)	Delete x from the ADT, if exists	O(log n)
Delete_in_place(i)	Delete from the ADT an element, which is in the ith place (as determined by the order of insertion) among the elements that are in the ADT at that moment.	O(log n)
Get_place(x)	Return the place (which is determined by the order of insertion) of x among the elements that are in the ADT at that moment. If x does not exist, return -1.	O(log n)

The Final Problem..(for this set)

11. An electrician wants to represent a list of her clients' records (by their ID).For each client we would like to mark whether he is a man or she is a woman.
Design a data structure that supports the following operations in O(log n) time in the worst case, where n is the number of persons (men and women)in the data structure when the operation is executed:

1.Insert(k,c)-Insert a new client c with id = k to the data structure, at first mark the client as a woman.

2.Update(k) –Update client with ID = k to be a man.

3.FindDiff(k) –Find the difference between the number of women and the number of men (| #of women -#of men|) among all the clients with ID smaller than k.

end

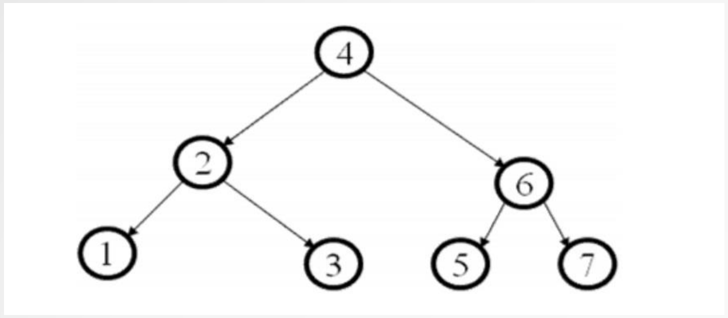
A few Sample Solutions

1.
   a. Be a binary tree
   b. Have Binary Search Tree ordering property (left children < parent, right children > parent)
   c. Be a balanced tree:
      $|n.left.height – n.right.height| \leq 1$
      for all nodes n in the tree.

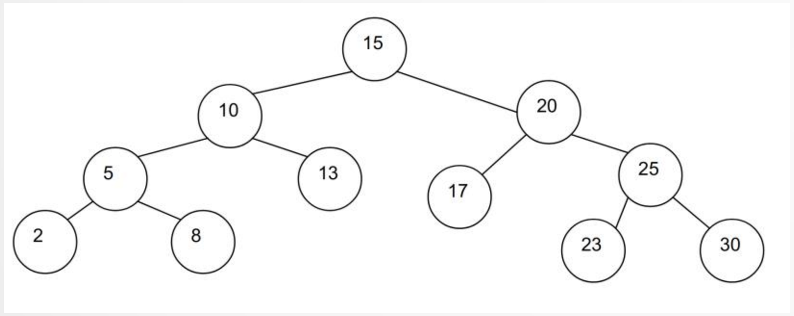
2. Construct a minimum size AVL tree of height h by creating a new root and making one of its children a minimum AVL tree of height h-1, and the other a minimum AVL tree of h-2. So to get a minimum AVL tree of height 4, we need to build up minimum AVL trees of heights 0-3 first. Note that there are many different possible orderings of branches(left versus right). The minimum number of nodes is 12.

Some More Solutions

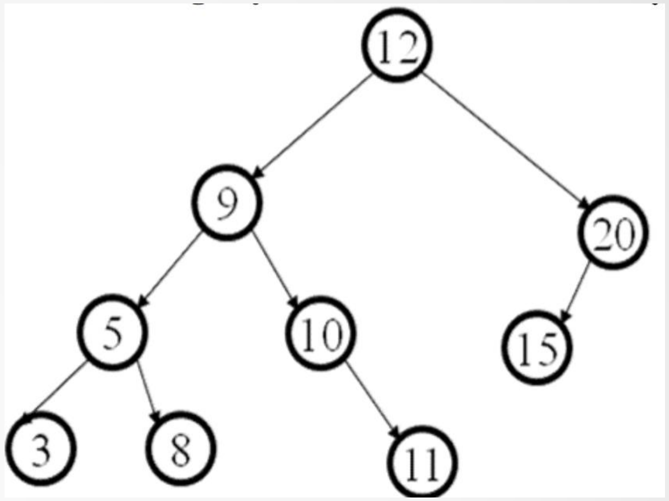
3.



4.



5.



The Final Solutions

11. The data structure is an AVL tree T where each node x represents a person and has the following fields (in addition to the regular fields of a node in an AVL tree):
    1.x.ID -ID number (the searchkey for T)
    2.x.client -the client record
    3.x.gender-1 (woman) or 0 (man)
    4.x.women-the number of all women in the subtree rooted at x (including x).
    5.x.men -the number of all men in the sub tree rooted at x (including x).

•Insert (k, c)

```
#create new node x
x.ID ←k
x.client ←c
x.gender←1 //(woman)
x.women ←1 // (a new node is always inserted as a leaf)
x.men ←0
AVL-Insert(x)// With rotations that update new fields// where necessary.
for every node v in the path fromx to the root do:
    v.women ←v.women + 1Time complexity: O(logn)
```

The End?