At this lab section, we will learn implementation of AVL Trees.

AVL Trees

Asst. Prof. Dr. Feriştah DALKILIÇ
Res. Asst. Fatih DİCLE

PART 1 – Course Questionnaire

Fill out the survey in the link below and paste the screenshot showing that you have completed it.

https://cscloud.deu.edu.tr/index.php/apps/forms/TjbkAwmn6DHKKgCt

Your Screenshot

PART 2 – AVL Tree

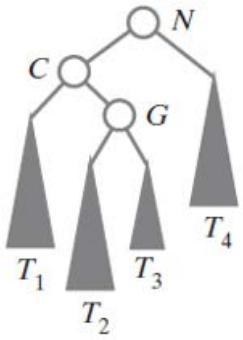
You can form several differently shaped binary search trees from the same collection of data. Some of these trees will be balanced and some will not. You could take an unbalanced binary search tree and rearrange its nodes to get a balanced binary search tree.

Every node in a balanced binary tree has subtrees whose heights differ by no more than 1.

The AVL tree is a binary search tree that rearranges its nodes whenever it becomes unbalanced. The balance of a binary search tree is upset only when you add or remove a node. Thus, during these operations, the AVL tree rearranges nodes as necessary to maintain its balance.

Steps to follow for insertion
Let the newly inserted node be W

1) Perform standard BST insert for W.
2) Starting from W, travel up and find the first unbalanced node. Let N be the first unbalanced node, C be the child of N that comes on the path from W to N and G be the grandchild of N that comes on the path from W to N.
3) Re-balance the tree by performing appropriate rotations on the subtree rooted with N.



There can be 4 possible cases that need to be handled as G, C and N can be arranged in 4 ways. Following are the possible 4 arrangements:

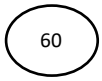
a) C is left child of N and G is left child of C **(Left Left Case) – (Single Right Rotation)**
b) C is left child of N and G is right child of C **(Left Right Case) – (Left Right Rotation)**
c) C is right child of N and G is right child of C **(Right Right Case) – (Single Left Rotation)**
d) C is right child of N and G is left child of C **(Right Left Case) – (Right Left Rotation)**

Example:

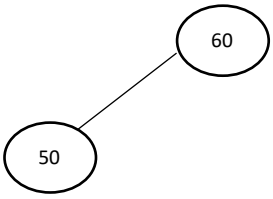
Insert items to the AVL tree.

Items = [60, 50, 20, 80, 90, 70, 55, 10, 40, 35]

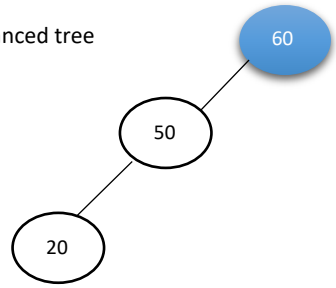
STEP 1
Add 60



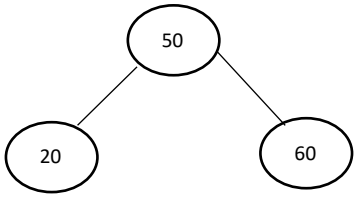
STEP 2
Add 50



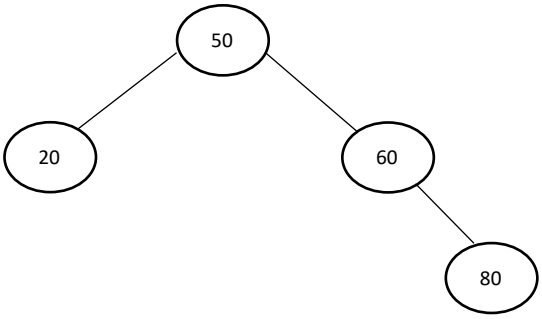
STEP 3
Add 20, unbalanced tree



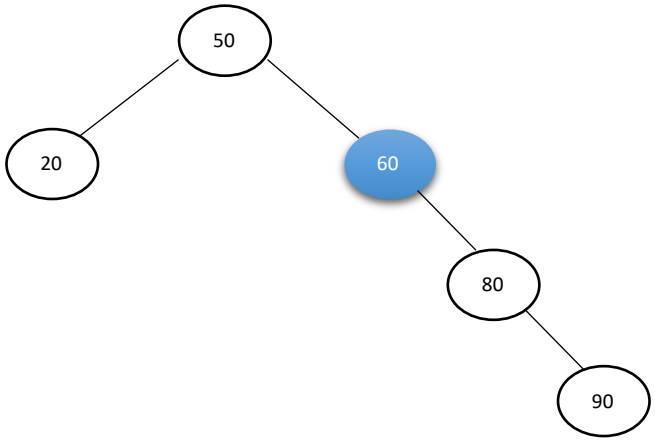
Right Rotation



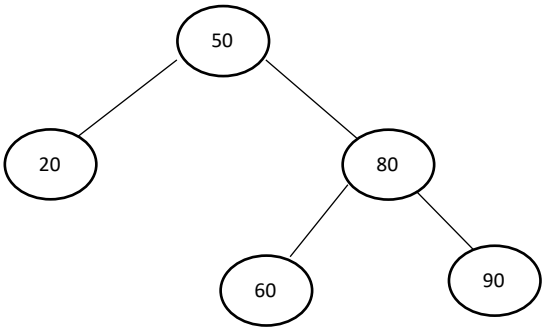
STEP 4
Add 80



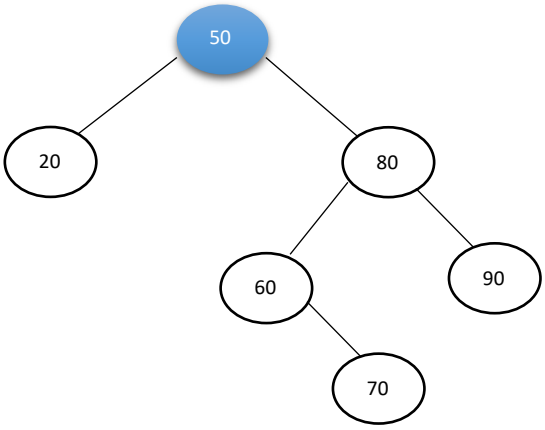
STEP 5
Add 90, unbalanced tree



Left Rotation

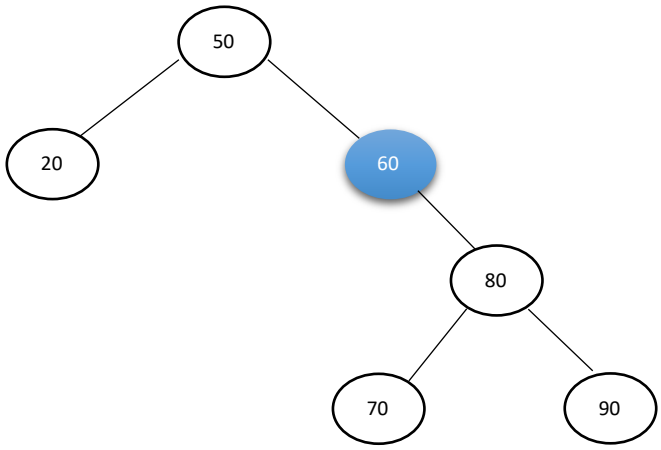


STEP 6
Add 70, unbalanced tree

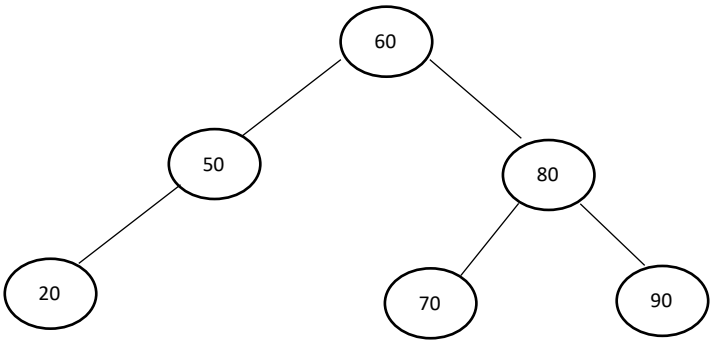


STEP 6
Right-left rotation

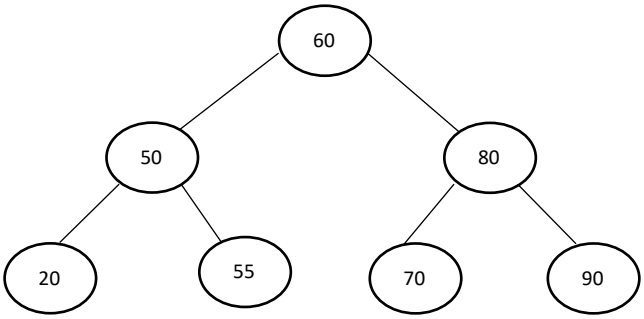
1) Right Rotation



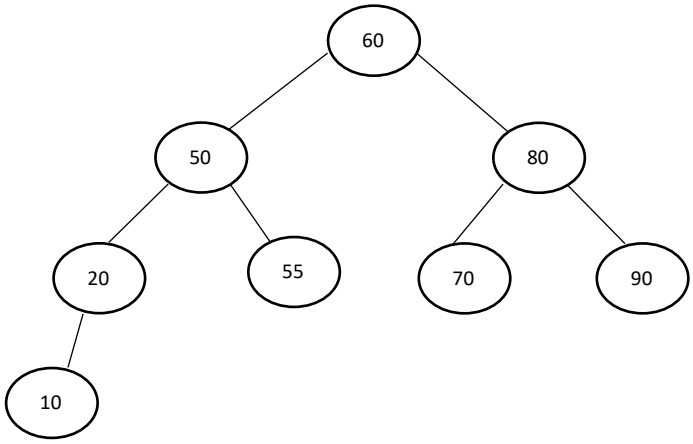
2) Left Rotation



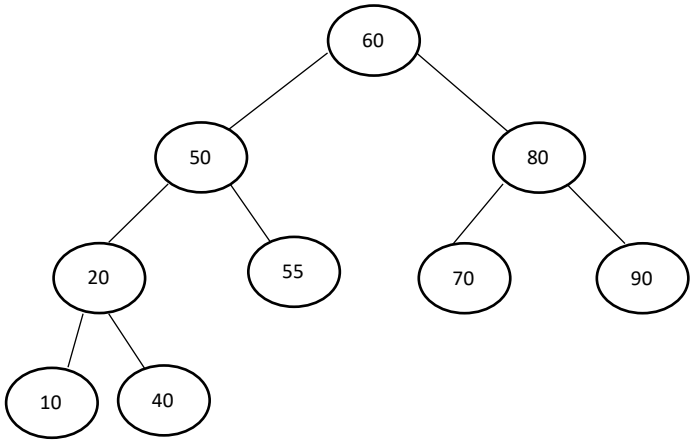
STEP 7
Add 55



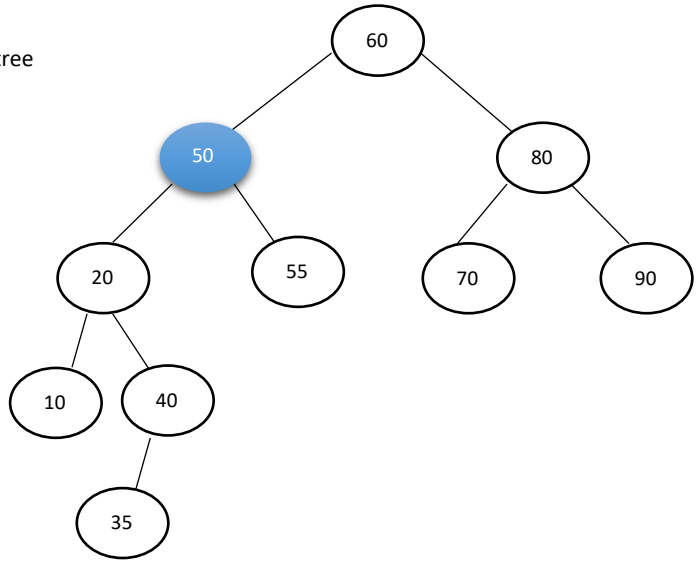
STEP 8
Add 10



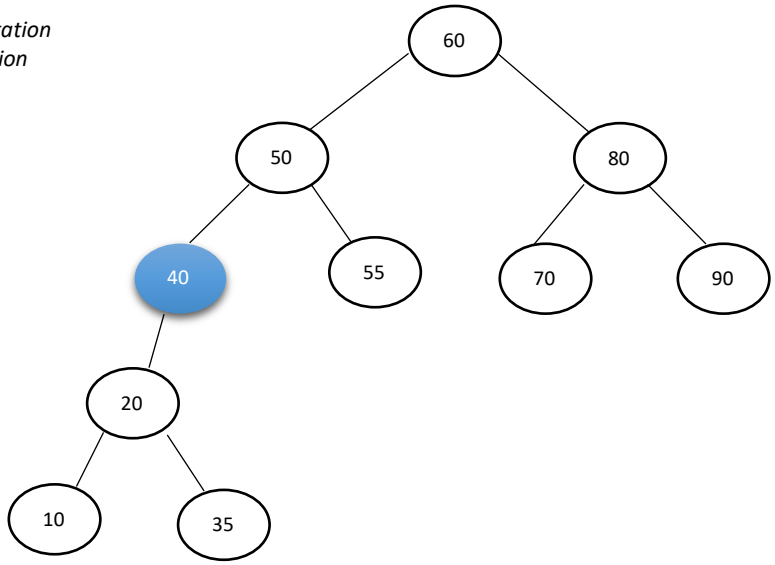
STEP 9
Add 40



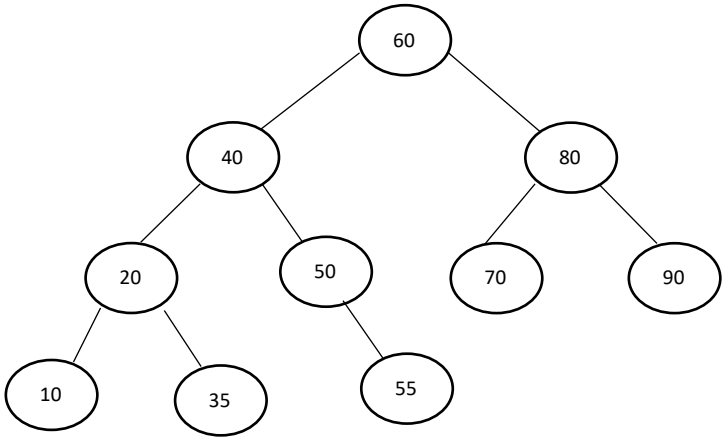
STEP 10
Add 35, unbalanced tree



STEP 10
Left-right Rotation
1) Left Rotation



2) Right Rotation



Exercise - 1

At this section, you will experiment some operations on both a binary search tree and an AVL Tree in Java.

Step – 1

Create a new Java project and add the given classes in *src* folder into your project.

Step – 2

Fill in the missing parts in *rotateLeft* and *rotateLeftRight* methods of *AVLTree.java*.

Step – 3

Add a new class with the name Test.java. Create a *BinarySearchTree* instance. Read the given input file *shuffled_numbers_100K.txt* and insert all the numbers into BST.

Step – 4

Create an *AVLTree* instance. Read the given input file *shuffled_numbers_100K.txt again* and insert all the numbers into AVL Tree.

Step – 5

Read the file *search_1K.txt* and search all the numbers in the BST.

Step – 6

Search for the same 1K numbers in the AVL Tree, as well.

Step – 7

Complete the table given below by measuring the performance of specified operations.

Hint: You can use the following code to measure the time spent (in millisecond) for an operation.

```
long StartTimer = System.nanoTime();
//perform your operation here
double timeDiff = (System.nanoTime() - StartTimer)/1000000;
```

Data Structure	Total Indexing Time for 100K Numbers	Final Height of the Tree	Total Searching Time for 1K Numbers	Successful Search Count	Unsuccessful Search Count
BST	?	?	?	?	?
AVL Tree	?	?	?	?	?

Step – 8

Paste the content of Test.java and your final output.

Your Test.java

Your Output