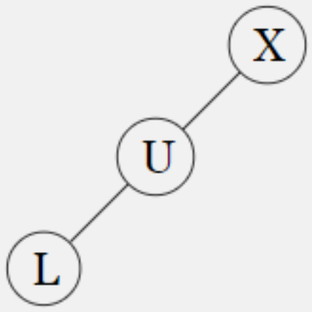
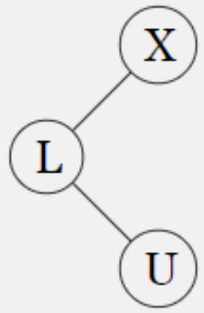
1. Rotations

In the following binary search trees, rotate the 'X' node to the right (that is, rotate it and its left child). Do these rotations make the tree more balanced, or less balanced?

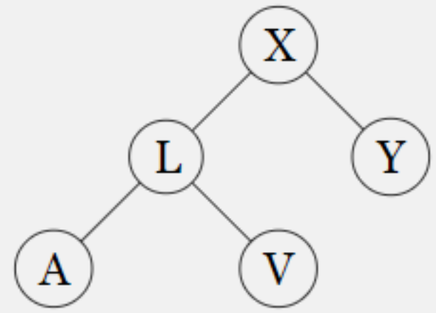
a.



b.

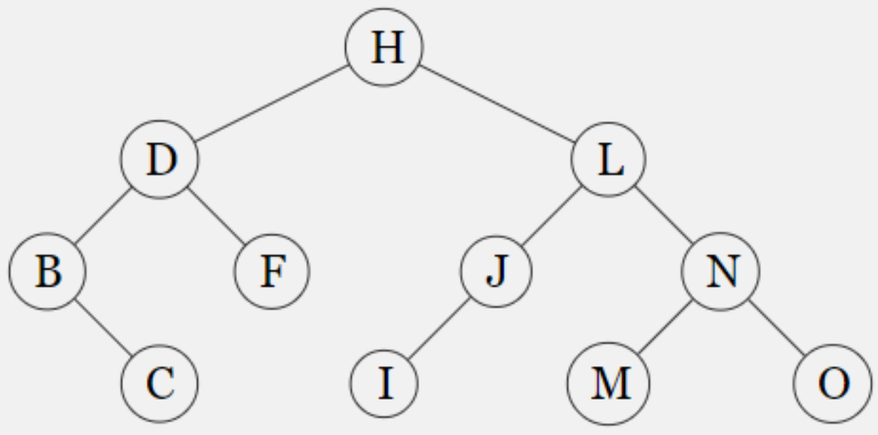


c.



2. Balance factor

A node's 'balance factor' is defined as the height of its right subtree minus the height of its left subtree. Calculate the balance factor of each node in the following binary search tree.



3. AVL Tree Insertion

Insert the following letters into an initially-empty AVL Tree.

A V L T R E X M P

4. AVL Tree Deletion

On the tree you built in exercise 3, perform the following deletions:

a. T

b. V

c. X

d. E

5. 2-3 Tree Insertion

Insert the following letters into an initially-empty 2-3 Tree.

A L G O R I T H M

6. (Optional Homework) 2-3 Tree Deletion

On the tree you built in exercise 5, perform the following deletions:

a. I

b. L

c. A

7. (Revision Homework) Mergesort Time Complexity

Mergesort is a divide-and-conquer sorting algorithm made up of three steps (in the recursive case):

1. Sort the left half of the input (using mergesort)
2. Sort the right half of the input (using mergesort)
3. Merge the two halves together (using a merge operation)

Construct a recurrence relation to describe the runtime of mergesort sorting n elements. Explain where each term in the recurrence relation comes from.

Use the Master Theorem to find the time complexity of Mergesort in Big-Theta terms.