American University of Beirut

Department of Computer Science
CMPS 212- Summer 2021/22
Lab 07

Problem 1 – Array Lists & Iterators

1 - Write a method which takes in an ArrayList and two objects of its element type and an int position. The method must replace an occurrence of an object equal to the first object by the second object. For example, if a call takes in an ArrayList of integers [5, 12, 4, 16, 4, 2, 2] and the integers 4, 7 and position 1 you should change the actual ArrayList object into [5, 12, 7, 16, 4, 2, 2] and return the modified array. If the position doesn't exist, throw an OutOfBoundsOccurence exception.

2 - Write a method which takes an ArrayList of integers, extracts, using an Iterator, the ones that are prime and stores them in a regular array. The regular array should remain sorted as you're inserting the elements in it.

Problem 2 – Binary Search Trees

Given the implementation of BST provided to you, add the following functionalities:

a- **isBST:** a method that determines wether a given tree is a BST or not (use the swap function provided to you to change the tree from a BST to a BT to test your method)

b- **search**: write a method search that returns true if an element exists in the BST, false otherwise.

c- **height:** write a method that determines the height of a BST given the root. The method should also work if provided with the root of a subtree.

d- **Balanced BST:** A BT is considered height-balanced if the left and right subtrees of every node differ in height by no more than 1.

Examples:

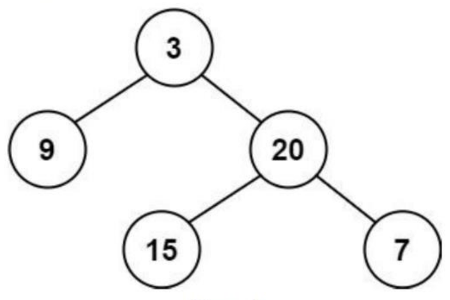


Figure 1

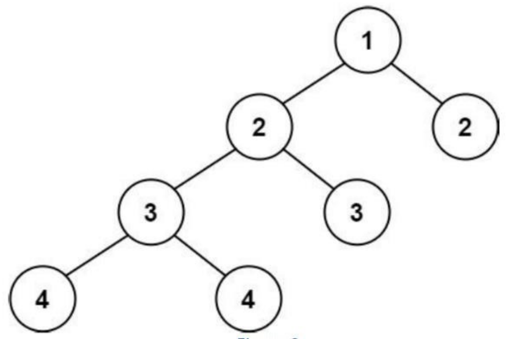


Figure 2

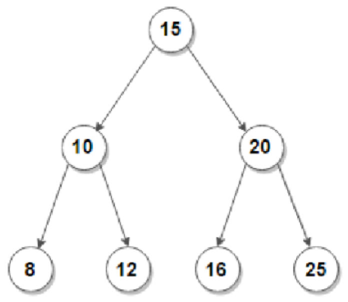
Figure 1 represents a height-balanced tree, since for every node in the tree the height difference between its left and right subtrees is less than or equal to 1.

Figure 2 does not represent a height-balanced tree, since node of value 1 has imbalanced subtrees. The left subtree's height is 3 while its right subtree's height is 1.

Add a boolean method to your implementation in problem 1 that returns true if the tree is height-balanced and false otherwise.

Hint: Make use of the height method.

e- **Kth largest element:** given a BST and an integer k, you have to find and print kth largest element in the tree. You are not allowed to modify the tree, nor use any extra auxiliary space.

	Given the BST on the left: For k=1, output=25 For k=4, output=15 For k=6, output=10 For k=7, output=8

Hint: In-order traversal gives you the values sorted in increasing order, think of a way to traverse the values in a decreasing order.

Problem 3 – Practice it!

Login to Practice it and access the following link https://practiceit.cs.washington.edu/problem/view/bjp4/chapter17/e15-trim to solve the problem.