Assignment 2 Questions

You must solve all the problems below, preferably before your lab test. The lab evaluation questions may build upon these assignment questions, and therefore it is strongly recommended that you try these questions beforehand.

Question 1

This question has been made available on Moodle virtual lab for you to practice. You can find the code there.

Given an input of n integers, given one at a time, output the median of the resulting array after every input. Do so efficiently - total time complexity of handling the input of n integers should be O(n log n); that is, for every input, you should be able to output the median in O(log n) time.

If currently there are an odd number of elements in the array, output the element with index (n-1)/2 (0-indexed). If there are an even number of elements in the array, the median is the average of the two middle elements (at indices (n-2)/2 and n/2) (round down this value to the nearest integer if required).

The function you implement will take a single integer as input - one new element to be added, and you should output the median of the array formed after adding this element in O(log n) time. Your function will be run in a loop to insert all n elements.

Note that it is easy to handle every new insertion in O(n) time: just insert the new element in the right position (O(n)), and find the median of this array (O(1)). Hint: Instead of an array, use two heaps, as declared in the skeleton code, to solve the problem.

You will also have to implement methods to insert new elements into the heap and delete the topmost element from the heap. We have intentionally not provided you these methods directly so that you can practice implementing these methods, and get a feel for how heaps work.

Hint: Maintain an invariant, such that the median is always at top of one of the heaps (or between the two, if there are an even number of elements).

Please see the quiz titled "Additional Hint for Question 1" if you want more hints for this question. However, we urge you to spend some time on this question without looking at the hint.

Sample Test Case

Number of elements: 6
Elements: 31, 15, 13, 18, 84, 12

Medians: 31, 23, 15, 16, 18, 16
Explanation:

- After inserting first element, sorted array has one element: [31]; therefore median is 31
- After inserting second element, sorted array has two elements: [15, 31]; Since number of elements is even, median is (15+31)/2 = 23
- After inserting third element, sorted array has three elements: [13, 15, 31]; therefore median is 15
- After inserting fourth element, sorted array has four elements: [13, 15, 18, 31]; median is (15+18)/2 = 16.5, we always round down, therefore output is 16.
- After inserting fifth element, sorted array has five elements: [13, 15, 18, 31, 84]; median is 18.
- After inserting sixth element, sorted array has six elements: [12, 13, 15, 18, 31, 84]; median is (15+18)/2 = 16.5, therefore we output 16.

Question 2

This question has been made available on Moodle virtual lab for you to practice. You can find the code there.

Given an array of integers and a target sum, find the number of pairs of numbers that add up to the given target sum. If no such pairs exist, output 0. You may not use the same value twice. Give an O(n) solution.

Hint: You are expected to use hashing for this question. You can use the HashMap data structure inbuilt in Java for this purpose.

We have added several large test cases to allow you to test the time complexity of your solution as well. An O(n^2) solution will definitely fail, however, we are setting the time limits such that an O(n log n) solution may pass. Please try to work towards implementing the O(n) solution itself however.

The answer may become very large for the large test cases, therefore, please report the answer modulo 1000000007.
A hint for modulo arithmetic: (a+b+c+d) mod N = (((a+b) mod N + c) mod N + d) mod N. Basically, on every addition, take it modulo the given number.

Sample Test Case
Array: [2, 4, 1, 3, 5, 5, 4, 2, 3, 5]
Target: 6

Expected output: 7
Explanation: The pairs of numbers that could add up to 6 are: (1,5) - there is one 1 and three 5's, which means a total of 3 pairs; (2,4) - there are two 2's and two 4's, which means a total of 4 pairs; and (3,3) - we do not consider any of these pairs, because of the condition that you cannot use the same value twice.

Question 3

Consider the definitions provided below:

- The depth of every node in a binary tree is equal to the shortest path from the root.
- A tree that consists of a node and all of its descendants is called a subtree.
- If a node has the most depth that any node in the entire tree may have, it is referred to as the deepest node.

Give back the root node of the smallest subtree that has every deepest node in the complete parent tree.

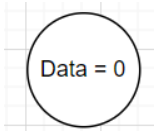
The drivercode has implemented the following test cases in the main function. Feel free to write methods of your own for convenience in testing, such as those for insertion and deletion of nodes.

There are two methods that you have to implement:

1. public int maxNodeDepth(Node root): This returns the maximum depth (with respect to ‘root’) of any node in the subtree rooted at ‘root’. You may use this function in the second method given below if you wish to do so.
2. public Node deepestSubtreeNode(Node root): This returns the root node of the smallest subtree that contains all the deepest nodes (with respect to ‘root’) in the complete parent tree rooted at ‘root’.

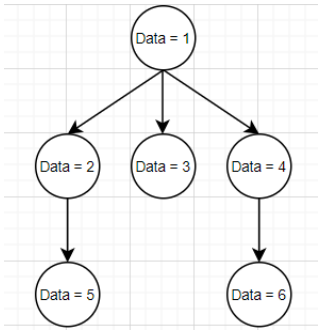
Test Cases:

Test 1:



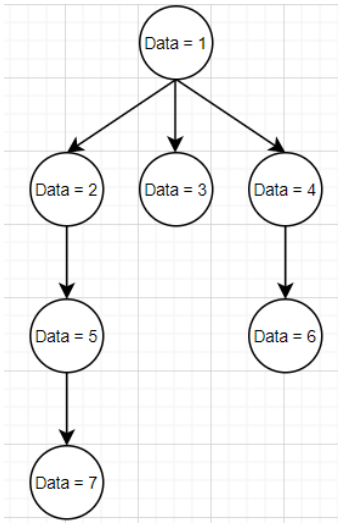
Expected node returned is node with data = 0

Test 2:



Expected node returned is the node with data = 1

Test 3:



Expected node returned is the node with data = 7

Question 4

You are provided an array ‘queries’ of size n made up of positive numbers and the root of a binary search tree.

Determine a 2D array ‘answer’ such that answer[i] = [min_i, max_i]:

The greatest value in the tree that is equal to or less than queries[i] is called min_i. In the event that such a value is absent, add -1.
The lowest value in the tree that is equal to or larger than queries[i] is called max_i. In the event that such a value is absent, add -1.
Give back the array ‘answer’.

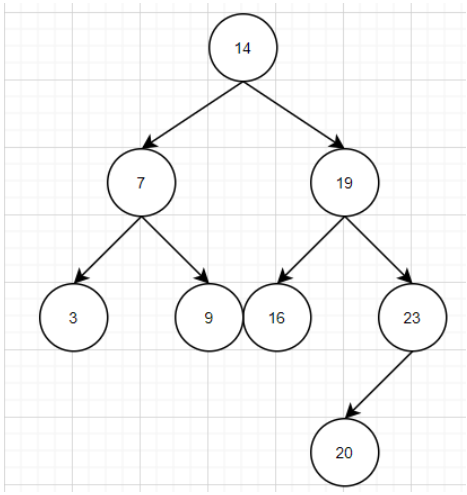
The drivercode has implemented the following test cases in the main function. Feel free to write methods of your own for convenience in testing, such as those for insertion and deletion of nodes.

You have to implement the method:

1. public Vector<Vector<Integer>> minMaxResult(Vector<Integer> queries): This returns the ‘answer’ vector with respect to the ‘queries’ vector.

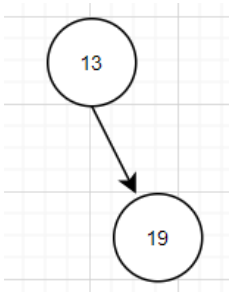
Test Cases:

Test 1:



Queries = [7, 12, 24]
Expected Output = [[7, 7], [9, 14], [23, -1]]

Test 2:



Queries = [9]
Expected Output = [[-1. 13]]

Instructions

- Do not change the accessibility, names or signatures of the attributes and methods in the driver code.