Correct

1 / 1 points

1.

Dynamic median. Design a data type that supports insert in logarithmic time, find-the-median in constant time, and remove-the-median in logarithmic time.

Note: these interview questions are ungraded and purely for your own enrichment. To get a hint, submit a solution.

Thank you for your response.

Hint: maintain two binary heaps, one that is max-oriented and one that is min-oriented.

Correct

1 / 1 points

2.

Randomized priority queue. Describe how to add the methods 𝚜𝚊𝚖𝚙𝚕𝚎() and 𝚍𝚎𝚕𝚁𝚊𝚗𝚍𝚘𝚖() to our binary heap implementation. The two methods return a key that is chosen uniformly at random among the remaining keys, with the latter method also removing that key. The 𝚜𝚊𝚖𝚙𝚕𝚎() method should take constant time; the 𝚍𝚎𝚕𝚁𝚊𝚗𝚍𝚘𝚖() method should take logarithmic time. Do not worry about resizing the underlying array.

Thank you for your response.

Correct

1 / 1 points

3.

Taxicab numbers. A taxicab number is an integer that can be expressed as the sum of two cubes of integers in two different ways: a3+b3=c3+d3. For example, 1729=93+103=13+123. Design an algorithm to find all taxicab numbers with a, b, c, and d less than n.

Version 1: Use time proportional to n2logn and space proportional to n2.

Version 2: Use time proportional to n2logn and space proportional to n.

Thank you for your response.

Hints:

Version 1: Form the sums a3+b3 and sort.

Version 2: Use a min-oriented priority queue with n items.

Correct

1 / 1 points

1.

Java autoboxing and equals(). Consider two 𝚍𝚘𝚞𝚋𝚕𝚎 values 𝚊 and 𝚋 and their corresponding <tt>Double</tt> values 𝚡 and 𝚢.

Find values such that (𝚊==𝚋) is 𝚝𝚛𝚞𝚎 but 𝚡.𝚎𝚚𝚞𝚊𝚕𝚜(𝚢) is 𝚏𝚊𝚕𝚜𝚎.

Find values such that (𝚊==𝚋) is 𝚏𝚊𝚕𝚜𝚎 but 𝚡.𝚎𝚚𝚞𝚊𝚕𝚜(𝚢) is 𝚝𝚛𝚞𝚎.

Note: these interview questions are ungraded and purely for your own enrichment. To get a hint, submit a solution.

Thank you for your response.

Hint: IEEE floating point arithmetic has some peculiar rules for 𝟶.𝟶, −𝟶.𝟶, and 𝙽𝚊𝙽. Java requires that 𝚎𝚚𝚞𝚊𝚕𝚜() implements an equivalence relation.

Correct

1 / 1 points

2.

Check if a binary tree is a BST. Given a binary tree where each 𝙽𝚘𝚍𝚎 contains a key, determine whether it is a binary search tree. Use extra space proportional to the height of the tree.

Thank you for your response.

Hint: design a recursive function 𝚒𝚜𝙱𝚂𝚃(𝙽𝚘𝚍𝚎𝚡,𝙺𝚎𝚢𝚖𝚒𝚗,𝙺𝚎𝚢𝚖𝚊𝚡) that determines whether 𝚡 is the root of a binary search tree with all keys between 𝚖𝚒𝚗 and 𝚖𝚊𝚡.

Correct

1 / 1 points

3.

Inorder traversal with constant extra space. Design an algorithm to perform an inorder traversal of a binary search tree using only a constant amount of extra space.

Thank you for your response.

Hint: you may modify the BST during the traversal provided you restore it upon completion.

Correct

1 / 1 points

4.

Web tracking. Suppose that you are tracking n web sites and m users and you want to support the following API:

User visits a website.

How many times has a given user visited a given site?

What data structure or data structures would you use?

Thank you for your response.

Hint: maintain a symbol table of symbol tables.