You can view this report online at : https://www.hackerrank.com/x/tests/1121389/candidates/27123055/report

Full Name:	My Nguyen
Email:	nguyen_my@yahoo.com
Test Name:	**Binary Trees Assessment 2021**
Taken On:	27 Jun 2021 16:25:29 PDT
Time Taken:	89 min 18 sec/ 90 min
Personal Email Address:	nguyen_my@yahoo.com
Contact Number:	+14084096862
Resume:	https://hackerrank-resumes.s3.amazonaws.com/412894/JhbK9vK_4Bhc4Gvuv7s5hgcFJGeFCAThWIiNY1UGAfhwRPsrmVekT5ZtKXgX8QA2Ag/My_Nguyen_Resume.PDF
Linkedin:	https://www.linkedin.com/in/my-nguyen-87849
Invited by:	Curriculum
Skills Score:	
Tags Score:	Binary Search Trees 65/150 Binary Trees 115/200 Hard 10/50

56.5%

130/230

scored in **Binary Trees Assessment 2021** in 89 min 18 sec on 27 Jun 2021 16:25:29 PDT

Recruiter/Team Comments:

No Comments.

Plagiarism flagged

We have marked questions with suspected plagiarism below. Please review.

	Question Description	Time Taken	Score	Status
Q1	What is the postorder traversal of this binary tree? › **Multiple Choice**	32 sec	5/ 5	✓
Q2	Suppose we want to make a copy of a binary tree, what type of traversal wou › **Multiple Choice**	2 min 52 sec	0/ 5	⊗
Q3	Given that the inorder traversal of a complete binary tree is: 3, 7, 2, 5, 9, 1, ›	50 sec	5/ 5	✓

	Multiple Choice			
Q4	Knowing that the tree below is a binary search tree, which numbers would we p › Multiple Choice	8 sec	5/ 5	✓
Q5	The following code is meant to get the sum of all left leaf nodes in a binary tr › Multiple Choice	3 min 56 sec	0/ 5	⊗
Q6	The following code is meant to validate if a binary tree is actually a binary se › Multiple Choice	2 min 6 sec	0/ 5	⊗
Q7	Average of Levels in Binary Tree › Coding	6 min 18 sec	50/ 50	!
Q8	Serialize and Deserialize BST › Coding	26 min 1 sec	50/ 50	✓
Q9	Binary Search Tree Iterator › Coding	41 min 19 sec	5/ 50	✓
Q10	Merge Two BSTs › Coding	3 min 1 sec	10/ 50	✓
Q11	Free Response › Subjective	2 min	0/ 0	⊖

QUESTION 1

✓

Correct Answer

Score 5

Multiple Choice

QUESTION DESCRIPTION

What is the postorder traversal of this binary tree?

```
      A
     / \
    B   C
   /   / \
  D   E   F
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

○ A, B, C, D, E, F

✓ ● D, B, E, F, C, A

○ D, B, A, C, E, F

○ D, E, F, B, C, A

No Comments

QUESTION 2

☒ Wrong Answer

Score 0

Multiple Choice

QUESTION DESCRIPTION

Suppose we want to make a copy of a binary tree, what type of traversal would be best suited for this problem?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ✓ ○ Preorder
- ○ Postorder
- ● Inorder

No Comments

QUESTION 3

✓ Correct Answer

Score 5

Multiple Choice

QUESTION DESCRIPTION

Given that the inorder traversal of a complete binary tree is: 3, 7, 2, 5, 9, 1, 10 what is the BFS traversal of the same tree?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ○ 3, 7, 2, 5, 9, 1, 10
- ○ 1, 2, 3, 5, 7, 9, 10
- ○ 5, 2, 9, 3, 7, 1, 10
- ✓ ● 5, 7, 1, 3, 2, 9, 10
- ○ 5, 1, 7, 3, 2, 9, 10

No Comments

QUESTION 4

✓

Correct Answer

Score 5

Multiple Choice

QUESTION DESCRIPTION

Knowing that the tree below is a binary search tree, which numbers would we pass through while looking for 22 with the most efficient method?

```
  10
 /  \
9    20
    /  \
  15    22
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

○ 10

✓ ● 10, 20

○ 10, 9, 20

○ 10, 9, 20, 15

No Comments

QUESTION 5

☒

Wrong Answer

Score 0

Multiple Choice

QUESTION DESCRIPTION

The following code is meant to get the sum of all left leaf nodes in a binary tree. Given the code and the tree below, what will the code output?

Java:

```java
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {

    public int sumOfLeftLeaves(TreeNode root) {
        return helper(root, 0);
    }

    public int helper(TreeNode root, int sumSoFar) {
        if (root == null) {
            return sumSoFar;
        }

        if (root.left != null) {
            sumSoFar += root.left.val;
        }

        return helper(root.left, sumSoFar) + helper(root.right, sumSoFar);
    }
}
```

Python:

```
"""
class TreeNode:
  def __init__(self, x):
    self.val = x
    self.left = self.right = None
"""

def sumOfLeftLeaves(root):
    def helper(root, sumSoFar):
        if not root:
            return sumSoFar

        if root.left:
            sumSoFar += root.left.val

        return helper(root.left, sumSoFar) + helper(root.right,sumSoFar)

    return helper(root, 0)
```

```
  3
 / \
9   20
   /  \
 15    7
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

○ 9

● 24

○ 31

○ 51

✓ ○ 114

No Comments

QUESTION 6

⊗ Wrong Answer

Score 0

Multiple Choice

QUESTION DESCRIPTION

The following code is meant to validate if a binary tree is actually a binary search tree. Will we produce the right output with the following code? If not, which lines need to be amended?

Java:

```
1   /**
2    * Definition for a binary tree node.
3    * public class TreeNode {
4    *     int val;
5    *     TreeNode left;
6    *     TreeNode right;
7    *     TreeNode(int x) { val = x; }
8    * }
```

```
8     * }
9     */
10  class Solution {
11      public boolean isValidBST(TreeNode root) {
12          return isValidBST(root, Integer.MIN_VALUE, Integer.MAX_VALUE);
13      }
14
15      public boolean isValidBST(TreeNode root, int min, int max) {
16          if (root == null) {
17              return true;
18          }
19          if (root.val >= max || root.val <= min) {
20              return false;
21          }
22          return isValidBST(root.left, Math.min(min, root.val),
Math.max(min, root.val)) &&
23              isValidBST(root.right, Math.min(min, root.val),
Math.max(min, root.val));
24      }
25  }
```

Python:

```
1   """
2   class TreeNode:
3       def __init__(self, x):
4       self.val = x
5       self.left = self.right = None
6   """
.
.

14  def isValidBST(root):
15      def helper(root, min, max):
16          if not root:
17              return True
18
19          if root.val >= max or root.val <= min:
20              return False
21
22          return helper(root.left, min(min, root.val), max(min,
root.val)) and
23              helper(root.right, min(min, root.val), max(min, root.val))
24
25      return helper(root, -float("inf"), float("inf"))
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ○ It will produce the right code
- ○ Line 19 needs to be fixed
- ○ Line 22 needs to be fixed
- ✓ ○ Line 23 needs to be fixed
- ● Lines 22 and 23 needs to be fixed
- ○ More than 2 lines need to be fixed

No Comments

QUESTION 7

Needs Review

Score 50

Average of Levels in Binary Tree › Coding Binary Trees

QUESTION DESCRIPTION

Given a non-empty binary tree, return the average value of the nodes on each level in the form of a list.

Example:

```
Input:
    3
   / \
  9  20
    /  \
   15   7

Output: [3, 14.5, 11]

Explanation:
The average value of nodes on level 0 is 3, on level 1 is 14.5, and on
level 2 is 11. Hence return [3, 14.5, 11].
```

CANDIDATE ANSWER

Language used: **Java 7**

```java
/**
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
static private List<Double> averageOfLevels(TreeNode root) {
    List<Double> averages = new ArrayList<>();
    if (root == null)
        return averages;

    Queue<TreeNode> queue = new LinkedList<>();
    queue.add(root);
    while (!queue.isEmpty()) {
        int size = queue.size();
        double sum = 0.0;
        for (int i = 0; i < size; i++) {
            TreeNode node = queue.poll();
            sum += node.val;

            if (node.left != null)
                queue.add(node.left);
            if (node.right != null)
                queue.add(node.right);
        }
        averages.add(sum / size);
    }
    return averages;
}


```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	✓ Success	20	0.0725 sec	24.5 KB
Testcase 2	Easy	Hidden case	✓ Success	10	0.1066 sec	24.3 KB
Testcase 3	Easy	Hidden case	✓ Success	5	0.3009 sec	24.1 KB
Testcase 4	Easy	Hidden case	✓ Success	5	0.0705 sec	24.4 KB
Testcase 5	Easy	Hidden case	✓ Success	10	0.0751 sec	24.4 KB

No Comments

QUESTION 8

✓ Correct Answer

Score 50

Serialize and Deserialize BST › Coding Binary Trees Binary Search Trees

QUESTION DESCRIPTION

Design an algorithm to serialize and deserialize a binary search tree.

In this problem, we want to ensure that if we serialize a binary tree into a string, the string can be deserialized back to the original tree

For example, you may serialize the following tree

```
    1
   / \
  2   3
     / \
    4   5
```

into a string seen as: `"[1,2,3,*,*,4,5]"` , with * representing a null node.

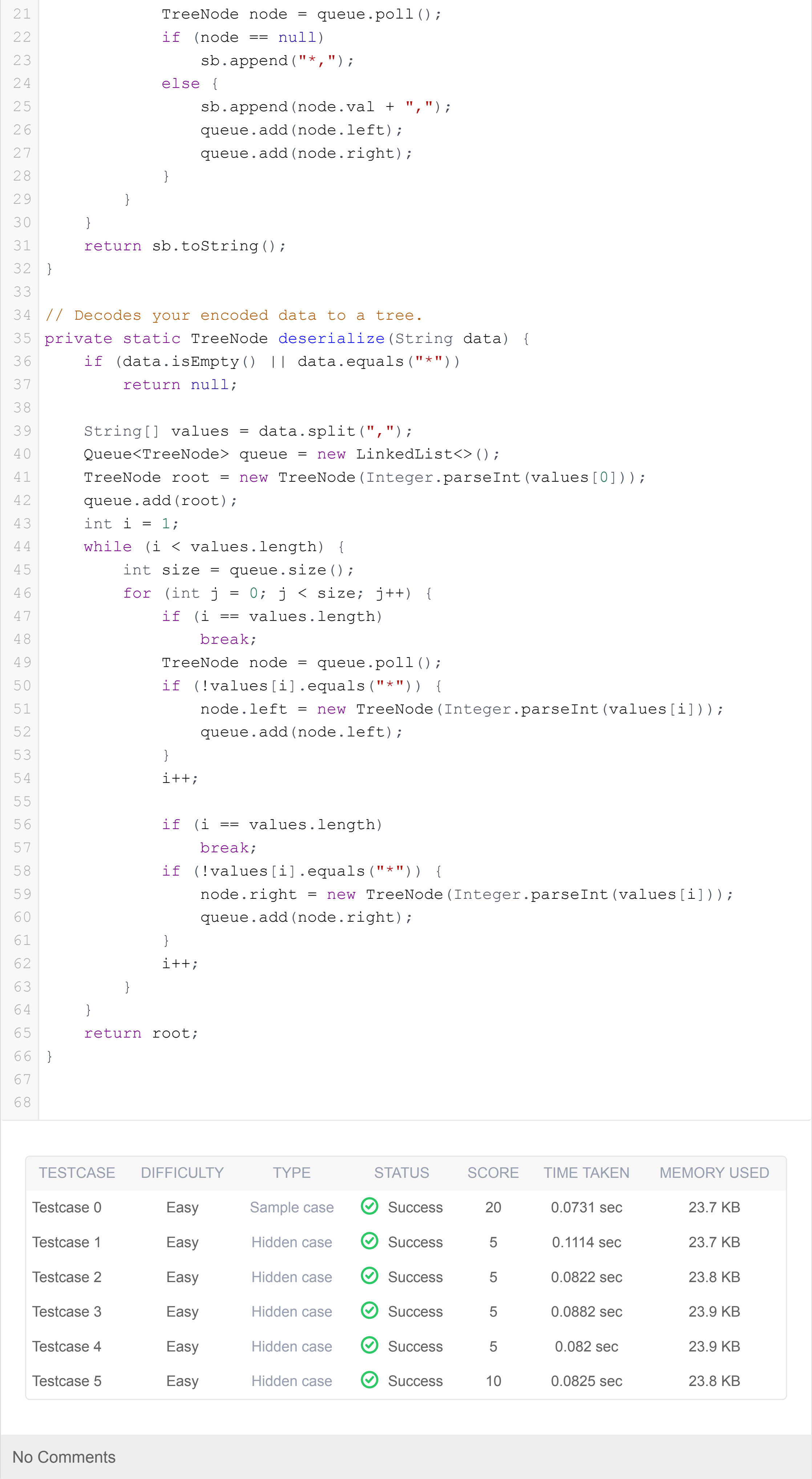
You don't need to follow the format seen above, so feel free to serialize the tree in a way that makes sense to you.

CANDIDATE ANSWER

Language used: **Java 7**

```
/**
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
// Encodes a tree to a single string.
private static String serialize(TreeNode root) {
    if (root == null)
        return "*";

    Queue<TreeNode> queue = new LinkedList<>();
    queue.add(root);
    StringBuilder sb = new StringBuilder();
    // sb.append(root.val);
    while (!queue.isEmpty()) {
        int size = queue.size();
        for (int i = 0; i < size; i++) {
```



```
            TreeNode node = queue.poll();
            if (node == null)
                sb.append("*,");
            else {
                sb.append(node.val + ",");
                queue.add(node.left);
                queue.add(node.right);
            }
        }
    }
    return sb.toString();
}

// Decodes your encoded data to a tree.
private static TreeNode deserialize(String data) {
    if (data.isEmpty() || data.equals("*"))
        return null;

    String[] values = data.split(",");
    Queue<TreeNode> queue = new LinkedList<>();
    TreeNode root = new TreeNode(Integer.parseInt(values[0]));
    queue.add(root);
    int i = 1;
    while (i < values.length) {
        int size = queue.size();
        for (int j = 0; j < size; j++) {
            if (i == values.length)
                break;
            TreeNode node = queue.poll();
            if (!values[i].equals("*")) {
                node.left = new TreeNode(Integer.parseInt(values[i]));
                queue.add(node.left);
            }
            i++;

            if (i == values.length)
                break;
            if (!values[i].equals("*")) {
                node.right = new TreeNode(Integer.parseInt(values[i]));
                queue.add(node.right);
            }
            i++;
        }
    }
    return root;
}


```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	20	0.0731 sec	23.7 KB
Testcase 1	Easy	Hidden case	✓ Success	5	0.1114 sec	23.7 KB
Testcase 2	Easy	Hidden case	✓ Success	5	0.0822 sec	23.8 KB
Testcase 3	Easy	Hidden case	✓ Success	5	0.0882 sec	23.9 KB
Testcase 4	Easy	Hidden case	✓ Success	5	0.082 sec	23.9 KB
Testcase 5	Easy	Hidden case	✓ Success	10	0.0825 sec	23.8 KB

No Comments

QUESTION 9

✓ Correct Answer

Score 5

Binary Search Tree Iterator › Coding

Binary Trees | Binary Search Trees

QUESTION DESCRIPTION

Implement an iterator over a binary search tree. Your iterator will be initialized with the root node of a BST. Calling `next()` should return the next smallest number in the BST.

The common iterator methods `next()` and `hasNext()` should run in average O(1) time and uses O(h) memory, where h is the height of the tree.

CANDIDATE ANSWER

Language used: **Java 7**

```
/**
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
static class BSTIterator {
    private TreeNode root;
    // private TreeNode parent;
    private TreeNode current;

    public BSTIterator(TreeNode root) {
        this.root = root;
        current = null;
    }

    /** @return whether we have a next smallest number */
    public boolean hasNext() {
        if (root == null)
            return false;

        if (current == null) {
            TreeNode parent = root;
            if (root.left != null)
                current = root.left;
            else if (root.right != null)
                current = root.right;
            else {
                current = root;
                return true;
            }

            while (current.left != null) {
                parent = current;
                current = current.left;
            }
        } else {
            TreeNode tmp = root;
            while (tmp != current) {
                if (tmp.val < current.val) {
                    tmp = tmp.right;
                } else {
                    tmp = tmp.left;
                }
            }
        }
        return true;
    }

    /** @return the next smallest number */
    public int next() {
        return current.val;
    }
}


```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	⊗ Terminated due to timeout	0	4.031 sec	87.7 KB
Testcase 1	Easy	Hidden case	⊗ Terminated due to timeout	0	4.0101 sec	85.6 KB
Testcase 2	Easy	Hidden case	✓ Success	5	0.0715 sec	23.8 KB
Testcase 3	Easy	Hidden case	⊗ Terminated due to timeout	0	4.038 sec	85.3 KB
Testcase 4	Easy	Hidden case	⊗ Terminated due to timeout	0	4.0702 sec	88.7 KB
Testcase 5	Easy	Hidden case	⊗ Terminated due to timeout	0	4.1043 sec	83.5 KB

No Comments

QUESTION 10

✓

Correct Answer

Score 10

Merge Two BSTs › Coding

Hard | Binary Search Trees | Binary Trees

QUESTION DESCRIPTION

Merge two binary search trees into a single balanced binary search tree.

Example:

```
Input:
   100          80
   /           /  \
  50          40   120
 /  \
20   70

Output:
           70
          /   \
       40      100
      /  \    /   \
     20   50  80   120
```

See if you can do it in linear time. i.e. if there are *m* elements in first tree and *n* elements in the other tree, your merge function should take *O*(*m*+*n*) time.

CANDIDATE ANSWER

Language used: **Java 7**

```java
/**
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
static TreeNode merge(TreeNode root1, TreeNode root2) {
    /* Write your code here.*/
    return root1;
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	⊗ Wrong Answer	0	0.0743 sec	23.6 KB
Testcase 1	Easy	Hidden case	✓ Success	5	0.0695 sec	23.7 KB
Testcase 2	Easy	Hidden case	✓ Success	5	0.074 sec	23.8 KB
Testcase 3	Easy	Hidden case	⊗ Wrong Answer	0	0.0802 sec	23.7 KB
Testcase 4	Easy	Hidden case	⊗ Wrong Answer	0	0.1473 sec	23.6 KB
Testcase 5	Easy	Hidden case	⊗ Wrong Answer	0	0.0842 sec	23.9 KB

No Comments

QUESTION 11

⊖

Self Evaluation

Score 0

Free Response › Subjective

QUESTION DESCRIPTION

Answer any of the following:
1. Can you think of an application of binary trees that you interact with every day?
2. What do you think are the implications of self-driving cars?
3. Do you ever feel that tech has gone too far? (e.g. with "smart" home appliances like Juicero, Google's AlphaGo AI, etc.)

CANDIDATE ANSWER

2. self-driving cars, if perfected, free drivers from the chore of driving so they can attend to some other task, even if it's sleep. the world is increasingly automated, from manufacture to airplanes, to scanning, to OCR, so why not self-driving cars?

No Comments

PDF generated at: 29 Jun 2021 06:57:32 UTC