

PDF generated at: 18 Jul 2024 03:03:20 UTC

View this report on HackerRank \Box

Score

94.6% • 175 / 185

scored in CodePath TIP103: Unit 5 Assessment - Summer 2024 in 67 min 23 sec on 17 Jul 2024 18:54:09 PDT

Candidate Information

Email tanveerm176@gmail.com

Test CodePath TIP103: Unit 5 Assessment - Summer 2024

Candidate Packet View ℃

Taken on 17 Jul 2024 18:54:09 PDT

Time taken 67 min 23 sec/ 90 min

Personal Email Address tanveerm176@gmail.com

Invited by CodePath

Suspicious Activity detected

Code similarity



Code similarity

2 questions

Skill Distribution

Candidate Report Page 1 of 24



There is no associated skills data that can be shown for this assessment

Tags Distribution

Binary Trees 100% Binary Search Trees 100%

Questions

Status	No.	Question	Time Taken	Skill	Score
8	1	What is the postorder traversal of this binary tree? Multiple Choice	3 min 15 sec	-	5/5
8	2	Binary Tree Traversal Multiple Choice	44 sec	-	5/5
⊗	3	Inorder Traversal of a complete binary tree Multiple Choice	1 min 22 sec	-	0/5
8	4	Binary Search Tree Efficiency Multiple Choice	2 min 3 sec	-	5/5

Candidate Report Page 2 of 24

5	Get the sum of all left leaf nodes in a binary tree Multiple Choice	3 min 36 sec	-	5/5
6	Validate if a binary tree is actually a binary search tree Multiple Choice	12 min 28 sec	-	0/5
7	Average of Levels in Binary Tree Coding	6 min 29 sec	-	50/50 🏳
8	Serialize and Deserialize Binary Tree Coding	21 min 11 sec	-	50/50
9	Binary Search Tree Iterator Coding	6 min 29 sec	-	50/50 🏳
10	What is the missing piece of code found in the push function? Multiple Choice	9 min 36 sec	-	5/5
	6 7 8	 in a binary tree Multiple Choice Validate if a binary tree is actually a binary search tree Multiple Choice Average of Levels in Binary Tree Coding Serialize and Deserialize Binary Tree Coding Binary Search Tree Iterator Coding What is the missing piece of code found in the push function? 	5 in a binary tree Multiple Choice 8 Validate if a binary tree is actually a binary search tree Multiple Choice 7 Average of Levels in Binary Tree 29 sec 8 Serialize and Deserialize Binary Tree Coding 9 Binary Search Tree Iterator 29 sec 9 What is the missing piece of code found in the push function? 9 What is the missing piece of code found in the push function?	5 in a binary tree Multiple Choice Validate if a binary tree is actually a binary search tree Multiple Choice 7 Average of Levels in Binary Tree Coding Serialize and Deserialize Binary Tree Coding Page 11 Sec

1. What is the postorder traversal of this binary tree?

Multiple Choice

Question description

What is the postorder traversal of this binary tree?

A /\

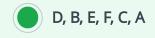
Candidate Report Page 3 of 24

ВС			
/ /\ D E F			
DEF			

Candidate's Solution

Options: (Expected answer indicated with a tick)









! No comments.

2. Binary Tree Traversal

 \bigcirc

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 Report Page 4 of 24

Candidate's Solution

Options: (Expected answer indicated with a tick)

Preorder	\otimes
Postorder	
Postorder	
Inorder	
• No comments.	

3. Inorder Traversal of a complete binary tree

Incorrect

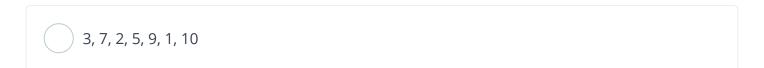
Multiple Choice

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?

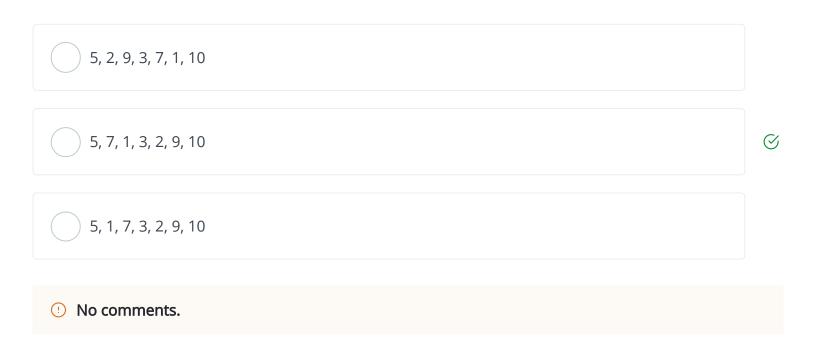
Candidate's Solution

Options: (Expected answer indicated with a tick)





Candidate Report Page 5 of 24



4. Binary Search Tree Efficiency

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

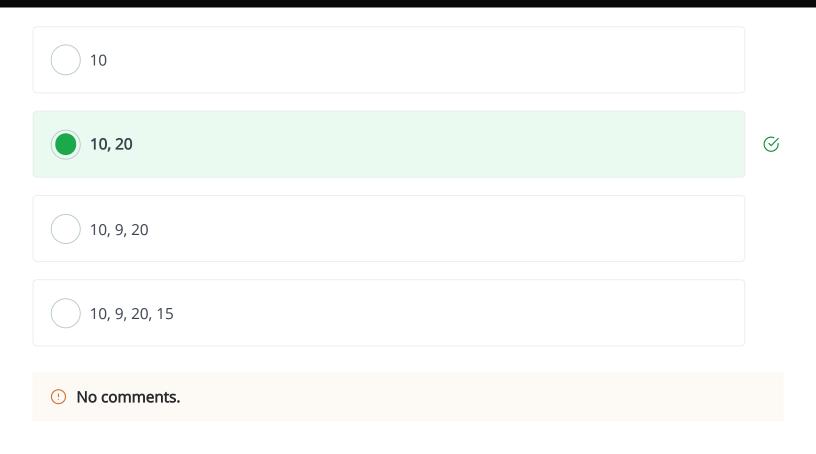
/ \

15 22
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

Candidate Report Page 6 of 24



5. Get the sum of all left leaf nodes in a binary tree

⊘ Correct

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:

```
/**

* 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) {
```

Candidate Report Page 7 of 24

```
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 Report Page 8 of 24

Candidate's Solution

Options: (Expected answer indicated with a tick)	
9	
24	\otimes
31	
51	
114	
① No comments.	
6. Validate if a binary tree is actually a binary search tree	⊗ Incorrect
Multiple Choice	
Question description	
The following code is meant to validate if a binary tree is actually a binary search tree. Will wright output with the following code? If not, which lines need to be amended?	e produce the
Java:	

Candidate Report Page 9 of 24

```
1 /**
2
  * Definition for a binary tree node.
3
  * public class TreeNode {
4
       int val;
       TreeNode left;
5
       TreeNode right;
6
7
   *
       TreeNode(int x) { val = x; }
8
   * }
9 */
10 class Solution {
     public boolean isValidBST(TreeNode root) {
11
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
        }
        if (root.val >= max | | root.val <= min) {
19
          return false;
20
21
        }
        return isValidBST(root.left, Math.min(min, root.val), Math.max(min, root.val)) &&
22
          isValidBST(root.right, Math.min(min, root.val), Math.max(min, root.val));
23
24
     }
25 }
```

Python:

```
1
  111111
2 class TreeNode:
3
     def __init__(self, x):
4
     self.val = x
5
     self.left = self.right = None
6
14 def isValidBST(root):
      def helper(root, min, max):
15
        if not root:
16
17
           return True
18
19
        if root.val >= max or root.val <= min:
```

Candidate Report Page 10 of 24

20 21	return False	
22 23	return helper(root.left, min(min, root.val), max(min, root.val)) and helper(root.right, min(min, root.val), max(min, root.val))	
24 25		
25	return helper(root, -float("inf"), float("inf"))	
Candid	ate's Solution	
Options	s: (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	\otimes
	Lines 22 and 23 needs to be fixed	
	More than 2 lines need to be fixed	
<u> </u>	lo comments.	

Candidate Report Page 11 of 24

7. Average of Levels in Binary Tree

Correct

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, 1 4.5, 11].
```

Candidate's Solution

Language used: Python 3

```
1 #!/usr/bin/env python
 2
 3 class TreeNode:
4
       def __init__(self, x):
 5
            self.val = x
            self.left = self.right = None
6
7
8
   def input_binary_tree():
9
       input_values = input().split()
       index = 0
10
       num nodes = int(input values[index])
11
12
       index += 1
13
       if (num nodes == 0):
14
            return None
15
16
       nodes = []
17
        current parent index = 0
```

Candidate Report Page 12 of 24

```
18
19
        root = TreeNode(int(input values[index]))
20
        index += 1
21
        nodes.append(root)
22
        for i in range(1, num nodes, 2):
23
            left val = int(input values[index])
24
25
            index += 1
            if (left val != -1):
26
27
                left = TreeNode(left val)
28
                nodes.append(left)
29
                nodes[current parent index].left = left
30
31
            right val = int(input values[index])
            index += 1
32
33
            if (right val != -1):
34
                right = TreeNode(right val)
35
                nodes.append(right)
36
                nodes[current parent index].right = right
37
38
            current parent index += 1
39
40
        return root
41
42
   class TreeNode:
43
        def __init__(self, x):
44
            self.val = x
45
            self.left = self.right = None
   0.000
46
47
48
   from collections import deque
49
   def average of levels(root):
        0.00
50
51
        Write your code here
52
        :type root: TreeNode
53
        :rtype: List[double]
54
55
56
        # if the tree is empty return an empty list
57
        if not root:
58
            return []
59
        result = []
60
61
        queue = deque([root])
62
63
        while queue:
```

Candidate Report Page 13 of 24

```
64
            level sum = 0
            level count = len(queue)
65
66
            # process all nodes at the current level
67
            for _ in range(level_count):
68
                node = queue.popleft()
69
70
71
                level_sum += node.val
72
73
                if node.left:
74
                    queue.append(node.left)
75
76
                if node.right:
77
                    queue.append(node.right)
78
            result.append(level_sum/level_count)
79
80
81
        return result
82
83
   root = input_binary_tree()
   averages = average_of_levels(root)
84
85
86
   for average in averages:
       print(round(average,1))
87
```

TESTCASE	DIFFICULTY	ТҮРЕ	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample	Success	20	0.0241 sec	9.37 KB
Testcase 2	Easy	Hidden	Success	10	0.0308 sec	9.5 KB
Testcase 3	Easy	Hidden	Success	5	0.0431 sec	9.45 KB
Testcase 4	Easy	Hidden	Success	5	0.0335 sec	9.38 KB

Candidate Report Page 14 of 24

Testcase Easy Hidden Success 10 0.0252 sec 9.44 KB

! No comments.

8. Serialize and Deserialize Binary Tree



Language used: Python 3

Coding Binary Trees Binary Search Trees

Question description

Design an algorithm to serialize and deserialize a binary 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's Solution

```
1 #!/usr/bin/env python
2
3 class TreeNode:
4   def __init__(self, x):
5    self.val = x
```

Candidate Report Page 15 of 24

```
self.left = self.right = None
 6
 7
 8
 9
   def insert(val, root):
10
        if val < root.val:</pre>
11
            if root.left is None:
12
                 root.left = TreeNode(val)
13
            else:
14
                insert(val, root.left)
15
        else:
16
            if root.right is None:
17
                 root.right = TreeNode(val)
18
            else:
19
                 insert(val, root.right)
20
21
22
   def input bst():
23
        input_values = map(int, input().split())
        num nodes = next(input_values)
24
25
        if num nodes == 0:
26
            return None
27
28
        root = TreeNode(next(input values))
29
30
        for i in range(1, num nodes):
31
            insert(next(input values), root)
32
33
        return root
34
35
36
   def description(root):
37
        if root is None:
38
            return " "
39
40
        queue = []
41
42
        output = str(root.val)
43
        queue.append(root)
44
        cursor = 0
45
46
        while cursor < len(queue):</pre>
47
            node = queue[cursor]
            cursor += 1
48
49
            if node.left is not None:
50
51
                 output += " " + str(node.left.val)
```

Candidate Report Page 16 of 24

```
52
                queue.append(node.left)
53
54
            if node.right is not None:
                output += " " + str(node.right.val)
55
56
                queue.append(node.right)
57
58
        return output
59
60
61
   from collections import deque
62
   0.000
63
64 class TreeNode:
       def __init__(self, x):
65
66
            self.val = x
67
            self.left = self.right = None
   0.000
68
69
70 def serialize(root):
71
        """Encodes a tree to a single string.
72
73
        :type root: TreeNode
74
        :rtype: str
75
76
        def serialize helper(node):
77
            if node is None:
78
                return result.append('None')
79
80
            else:
81
                result.append(str(node.val))
                serialize helper(node.left)
82
83
                serialize helper(node.right)
84
85
        result = []
        serialize helper(root)
86
        return ','.join(result)
87
88
89
90
91
92
   def deserialize(data):
93
        """Decodes your encoded data to tree.
94
95
        :type data: str
96
        :rtype: TreeNode
        0.00
97
```

Candidate Report Page 17 of 24

```
98
 99
         def deserialize helper():
100
             if data[0] == 'None':
                 data.popleft()
101
102
                 return None
103
             node = TreeNode(int(data.popleft()))
104
             node.left = deserialize_helper()
105
             node.right = deserialize helper()
106
107
             return node
108
        data = deque(data.split(','))
109
110
         return deserialize helper()
111
112
113
114
115
116
117
     root = input bst()
118
    new root = deserialize(serialize(root))
119
    print(description(new root))
120
121
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample	Success	20	0.0263 sec	9.36 KB
Testcase 1	Easy	Hidden	Success	5	0.0288 sec	9.43 KB
Testcase 2	Easy	Hidden	Success	5	0.0287 sec	9.36 KB
Testcase 3	Easy	Hidden	Success	5	0.0429 sec	9.46 KB

Candidate Report Page 18 of 24

Testcase 4	Easy	Hidden	Success	5	0.0293 sec	9.32 KB	
Testcase 5	Easy	Hidden	Success	10	0.033 sec	9.29 KB	

No comments.

9. Binary Search Tree Iterator

Correct

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's Solution

Language used: Python 3

```
1 #!/usr/bin/env python
 2
3 class TreeNode:
 4
        def init (self, x):
 5
            self.val = x
6
            self.left = self.right = None
7
8
9
   def insert(val, root):
       if val < root.val:</pre>
10
            if root.left is None:
11
                root.left = TreeNode(val)
12
13
            else:
```

Candidate Report Page 19 of 24

```
14
                insert(val, root.left)
15
        else:
16
            if root.right is None:
                root.right = TreeNode(val)
17
18
            else:
19
                insert(val, root.right)
20
21
22
   def input bst():
23
        input values = map(int, input().split())
24
        num_nodes = next(input_values)
        if num nodes == 0:
25
            return None
26
27
28
        root = TreeNode(next(input values))
29
30
        for i in range(1, num nodes):
31
            insert(next(input_values), root)
32
33
        return root
34
35
   0.000
36
37
   class TreeNode:
38
        def __init__(self, x):
39
            self.val = x
40
            self.left = self.right = None
    0.000
41
42
   class BSTIterator(object):
43
        def __init__(self, root):
44
45
            :type root: TreeNode
            0.00
46
47
            self.stack = []
48
            self.leftmost inorder(root)
49
        def leftmost inorder(self, root: TreeNode):
50
            # Helper fucntion to push all the left most nodes of the tree to the
51
   stack
52
            while root:
53
                self.stack.append(root)
54
                root = root.left
55
56
57
        def has next(self):
            0.00
58
```

Candidate Report Page 20 of 24

```
59
            :rtype: bool
            0.000
60
61
            return len(self.stack) > 0
62
63
64
        def next(self):
65
66
            :rtype: int
            0.00\,0
67
            topmost_node = self.stack.pop()
68
            if topmost_node.right:
69
                self.leftmost_inorder(topmost_node.right)
70
71
72
            return topmost_node.val
73 root = input_bst()
74 iter = BSTIterator(root)
75 while iter.has next():
        print(iter.next())
76
77
```

TESTCASE	DIFFICULTY	ТҮРЕ	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample	Success	20	0.0266 sec	9.28 KB
Testcase 1	Easy	Hidden	Success	10	0.0319 sec	9.25 KB
Testcase 2	Easy	Hidden	Success	5	0.0248 sec	9.13 KB
Testcase 3	Easy	Hidden	Success	5	0.0247 sec	9.2 KB
Testcase 4	Easy	Hidden	Success	5	0.025 sec	9.11 KB

Candidate Report Page 21 of 24

Testcase 5 0.0262 sec 9.23 KB

Onumber 19 On

10. What is the missing piece of code found in the push function?



Multiple Choice

Question description

Given two queues, implement a last-in-first-out (LIFO) stack. The implemented stack supports all the functions of a normal stack (push , top , pop , and empty).

The MyStack class consists of the following functions:

- void push(int x) Pushes element x to the top of the stack.
- int pop() Removes the element on the top of the stack and returns it.
- int top() Returns the element on the top of the stack.
- boolean empty() Returns true if the stack is empty, false otherwise.

Notes:

- You must use **only** standard operations of a queue, which means that only push to back, peek/pop from front, size and is empty operations are valid.
- Depending on your language, the queue may not be supported natively. You may simulate a queue using a list or deque (double-ended queue) as long as you use only a queue's standard operations.

Example 1:

```
Input
["MyStack", "push", "push", "top", "pop", "empty"]
[[], [1], [2], [], []]
Output
[null, null, null, 2, 2, false]

Explanation
MyStack myStack = new MyStack();
myStack.push(1);
myStack.push(2);
myStack.top(); // return 2
```

Candidate Report Page 22 of 24

```
myStack.pop(); // return 2
myStack.empty(); // return False
```

What is the missing piece of code found in the push function?

```
class MyStack:
  def __init__(self):
    self.q = deque()
    self.t = None
  def push(self, x: int) -> None:
    if self.t:
       // add missing line of code here
    self.t = x
  def pop(self) -> int:
    result = self.t
    newq = deque()
    while len(self.q) > 1:
       newq.append(self.q.popleft())
    self.t = self.q.popleft() if self.q else None
    self.q = newq
    return result
  def top(self) -> int:
    return self.t
  def empty(self) -> bool:
    return self.t is None
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

Candidate Report Page 23 of 24

self.q = deque()	
self.q.append(self.q.popleft())	
self.q.append(self.t)	\otimes
return self.l[-1]	
none of the above	
• No comments.	

Candidate Report Page 24 of 24