Unit 19 - Week 6 Quiz

Week 6 Quiz

The due date for submitting this assignment has passed. **Due on 2020-03-11, 23:59 IST.**
As per our records you have not submitted this assignment.

All questions carry equal weightage. You may submit as many times as you like within the deadline. Your final submission will be graded.

1) For a pair of nodes v_1 and v_2 in a binary search tree, the least common ancestor of v_1 and v_2 is the lowest level node in the tree from where there are paths to both v_1 and v_2. ***2 points***

Suppose we implement a binary search tree without a parent pointer in each node. So each node v has an associated value, value(v), and pointers left(v) and right(v) to its left and right children, respectively.

What is the worst case complexity of finding the least common ancestor of an arbitrary pair of nodes in a balanced search tree with n nodes?

- O(n log n)
- O(n)
- O(log n)
- O(log n + d), where d = value(v_2) - value(v_1)

No, the answer is incorrect.
Score: 0

Feedback:
We start from the root and search for value(v_1) and value(v_2) in parallel. The least common ancestor is the first node where the two parallel searches go in different directions. So this takes at most O(log n) steps.

Accepted Answers:
O(log n)

2) We have n distinct values stored in a height balanced (AVL) binary search tree. Which of the following statements is ***always*** true? ***2 points***

- The value at each node is the median of the values in the subtree rooted at that node.
- The shortest path between any pair of nodes is at most O(log n).

Week 4 : Weighted graphs

Week 4 Quiz

Week 4 Programming Assignment

Week 5: Data Structures: Union-Find and Heaps

Week 5 : Divide and Conqure

Week 5 Quiz

Week 6: Data Structures: Search Trees

Week 6: Greedy Algorithms

Week 6 Quiz

Quiz : Week 6 Quiz (assessment?name=116)

Week 6 Programming Assignment

Week 7: Dynamic Programming

Week 7 Quiz

Week 7 Programming Assignment

Week 8: Linear Programming and Network Flows

Week 8: Intractability

Week 8 Quiz

For any node, the difference between the size of the left subtree and the right subtree is at most 3.

The number of leaf nodes is greater than or equal to the number of internal nodes.

No, the answer is incorrect.
Score: 0

Feedback:

A height balanced tree has overall height at most O(log n), so the shortest path between any pair of nodes is always at most O(log n).

The following AVL tree is a counterexample for all other statements.

```
         3
        / \
       2   7
      /   / \
     1   5   8
        / \   \
       4   6   9
```

Accepted Answers:

The shortest path between any pair of nodes is at most O(log n).

3) Consider the following function to traverse a search tree t with integer values, where even(m) returns True if m is an even number and False otherwise. ***2 points***

```
function strangeOrder(t) {
  if (t != NIL) {
    if (even(t.value)){
      strangeOrder(t.left);
      strangeOrder(t.right);
      print(t.value);
    }else{
      print(t.value);
      strangeOrder(t.right);
      strangeOrder(t.left);
    }
  }
}
```

What is the complexity of this traversal for a binary search tree with n nodes?

O(n) whether the tree is balanced or unbalanced.

O(n log n) whether the tree is balanced or unbalanced.

O(log n) if the tree is balanced, O(n) otherwise.

O(n) if the tree is balanced, O(n log n) otherwise.

No, the answer is incorrect.
Score: 0

Feedback:

The traversal visits and lists each element once, so it is O(n).

Accepted Answers:

O(n) whether the tree is balanced or unbalanced.

4) Postorder traversal lists out the nodes in a binary tree as follows. ***2 points***

- First do a postorder traversal of the left subtree of the root.
- Then do a postorder traversal of the right subtree of the root.
- Finally list out the value at the root.

Text Transcripts

Books

Download Videos

The postorder traversal of a binary search tree with integer values produces the following sequence: 7, 12, 25, 47, 89, 55, 42, 17. What is the value of the left child of the root of the tree?

- 42
- 25
- 12
- 7

No, the answer is incorrect.
Score: 0

Feedback:

The inorder traversal of a search tree is always the sorted sequence. In this case: 7, 12, 17, 25, 42, 47, 55, 89. From the postorder traversal, we know that 17 is the root of the tree, so the segment 7, 12 corresponds to the left subtree and the segment 25, 42, 47, 55, 89 corresponds to the right subtree. The postorder traversal of the left subtree ends with 12, so this is the left child of the root node.

Accepted Answers:

12

5) Suppose we build a Huffman code over the four letter alphabet {a,b,c,d}, where f(a), f(b), f(c) ***2 points*** and f(d) denote the frequencies (probabilities) of the letters and f(a) > f(b) > f(c) > f(d). Which of the following is a valid conclusion?

- The Huffman code will assign two bits to every letter always.
- If f(c)+f(d) > f(b), the Huffman code will assign two bits to every letter, otherwise the letters will be encoded with varying numbers of bits.
- If f(c)+f(d) > f(a), the Huffman code will assign two bits to every letter, otherwise the letters will be encoded with varying numbers of bits.
- If f(b) > f(c)+f(d), the Huffman code will assign two bits to every letter, otherwise the letters will be encoded with varying numbers of bits.

No, the answer is incorrect.
Score: 0

Feedback:

The first iteration will group {c,d} as {cd}. To assign a variable number of bits, the second iteration should combine {b,cd} to form {bcd}. For this, we need f(a) > f(b) and f(a) > f(c)+f(d).

- *If f(c)+f(d) > f(b), we may still have f(a) > f(c)+f(d) > f(b), so we end up with a variable number of bits for a, b and {cd}.*
- *If f(c)+f(d) > f(a), the second iteration will definitely group the letters as {ab}, {cd}, so we get 2 bits per letter.*
- *If f(b) > f(c)+f(d), we definitely have f(a) > f(b) > f(c)+f(d), so we end up with a variable number of bits for a, b and {cd}.*

Accepted Answers:

If f(c)+f(d) > f(a), the Huffman code will assign two bits to every letter, otherwise the letters will be encoded with varying numbers of bits.