Week 2 Quiz

TOTAL POINTS 10

1. For a linked structure of edges and nodes to be a tree, which of the following is not required to be true? 1 point

   - ◯ If any two nodes are connected, they are connected by only one path of unique nodes and edges.
   - ◯ Every node has zero, one or two children.
   - ◯ Every node has a parent except for one single root node.
   - ◯ Every node is connected to every other node by some path of edges.

2. Which data structure below supports the fastest run time for finding an item in a sorted list of items? 1 point

   - ◯ Array
   - ◯ Linked List
   - ◯ Binary Search Tree
   - ◯ All of these data structures have the same run time complexity for finding an item in a sorted list of items.

3. What is the height of the binary search tree created by inserting the following values one at a time in the following order of insertion: **1 2 3 4 5 6 7** ? 1 point

   - ◯ 2
   - ◯ 3
   - ◯ 6
   - ◯ 7

4. Which of the following is NOT true of a perfect binary search tree of a list of n ordered items? 1 point

   - ◯ If the height of the tree is h, then n = 2^(h+1) - 1.
   - ◯ All of the leaf nodes are at the same level.
   - ◯ The worst-case run time to find an item is O(n).
   - ◯ Every non-leaf node has two children.

5. Which of the following is NOT a full binary tree? 1 point

   - ◯ A single node.
   - ◯ A perfect binary tree.
   - ◯ The binary tree consisting of the subtree of ancestors of any node in any perfect binary tree.
   - ◯ The binary search tree created by inserting the following values one at a time: **4 2 3 5 1**.

6. Which of the following is not a true statement about a complete binary tree? 1 point

   - ◯ No node in a complete binary tree has only a right child.
   - ◯ The worst-case run time for finding an object in a complete binary search tree of an ordered list of n items is O(lg n).

- Any tree that contains a node with a single child is not a complete binary tree.
- The height of a complete binary tree of n nodes is floor(lg n).

7. Which one of the following functions outputs the keys of a binary search tree in item order when the root node is passed to it as its parameter. 1 point

```
void print(TreeNode *node){
    if (!node) return;
    std::cout << node->key << " ";
    print(node->left);
    print(node->right);
}
```

```
void print(TreeNode *node){
    if (!node) return;
    print(node->left);
    std::cout << node->key << " ";
    print(node->right);
}
```

```
void print(TreeNode *node){
    if (!node) return;
    print(node->left);
    print(node->right);
    std::cout << node->key << " ";
}
```

- None of these outputs all of the keys of the binary search tree in item order.

8. Consider the binary search tree built by inserting the following sequence of integers, one at a time: **5, 4, 7, 9, 8, 6, 2, 3, 1** 1 point

Which method below will properly remove node **4** from the binary search tree?

- Set the *left* pointer of node **5** to point to the node pointed to by the *left* pointer of node **4**, and then delete node **4**.
- Find the in order predecessor (IOP) of node **4**, which is node **3**. Remove node **3** from the tree by setting the right pointer of its parent (node **2**) to point to the node pointed to by the left pointer of node **3**. Then copy the key and any data from node **3** to node **4**, turning node **4** into a new node **3**, and delete the old node **3**.
- Find the in order predecessor (IOP) of node **4**, which is node **3**. Remove node **3** from the tree by setting the right pointer of its parent (node **2**) to **nullptr**. Then copy the key and any data from node **3** to node **4**, turning node **4** into a new node **3**, and delete the old node **3**.
- Set the *left* pointer of node **5** to **nullptr**, and then delete node **4**.

9. Suppose that we have numbers between 1 and 1000 in a binary search tree and we want to search for the number **363**. Which of the following sequences can **not** be the sequence of nodes visited in the search? 1 point

- **925, 202, 911, 240, 912, 245, 363**
- **924, 220, 911, 244, 898, 258, 362, 363**
- **2, 399, 387, 219, 266, 382, 381, 278, 363**
- **2, 252, 401, 398, 330, 344, 397, 363**

10. Given any binary tree with 128 nodes where each node has a *left* pointer and a *right* pointer, how many of these pointers are set to **nullptr**? 1 point

Enter answer here

☐ I, **Jiarong Yang**, understand that submitting work that isn’t my own may result in permanent failure of this course or deactivation of my Coursera account.

Learn more about Coursera’s Honor Code

Save Submit