CS 101 Fall 2023 - Quiz 6

11/20/2023 - 20 Minutes **Name:** **ID number:**

---

1. (8 points) True or False

For each statement, choose T if the statement is correct, otherwise, choose F.

Note: You should write down your answers in the box below.

(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
F	F	F	T	F	T	T	F

(a) (1') In a BST with n nodes, we can always find its maximum element in $O(\log(n))$.

Solution: A BST may be un-balanced. Its height may be $\Theta(n)$.

(b) (1') The bottleneck of heap sort is building heap, which has to take $\Theta(n \log n)$ to finish.

Solution: Floyd's Method runs in $\Theta(n)$ of time.

(c) (1') For the same character set, if their frequencies differ, then the Huffman coding trees generated by them will always be different.

Solution: They could be the same. For example, two characters of frequency 10 and 20, and two characters of frequency 100 and 200, will have the same Huffman Coding Trees.

(d) (1') In a BST, if a_i represents the i-th element visited during the in-order traversal, then $a_i \leq a_j$ for all $i < j$.

Solution: The in-order traversal of a BST is non-descending.

(e) (1') In a binary max-heap, there is at least one internal node that is the smallest element in its subtree.

Solution: An internal node must have at least one children, and it must be greater than or equal to them.

(f) (1') If the post-order traversal of a complete binary tree is descending, then the binary tree is a min-heap.

Solution: From the condition we can prove that each node is smaller than its children.

(g) (1') We could use a priority queue to finish Huffman Coding.

Solution: We need a data structure that supports insertion and poping the minima. Priority queue supports these operations.

(h) (1') There is no BST with more than 1 nodes such that its post-order traversal is ascending.

Solution: Suppose the root with value 2 has only a left child with value 1. Then the post-order traversal is $(1, 2)$.

2. (5 points) Fill in the Blanks

We have a complete binary min-heap with n nodes. Let h be the height of the heap. Assume the n elements are **distinct**. Fill the blanks below. If multiple answers apply to a blank, list them all.

(a) (2') Insertion operation has a best case runtime of $\Theta(1)$ and a worst case runtime of $\Theta(\log n)$. Write in terms of n and use Θ symbol.

Solution: This depends on percolation. The number of percolation is at least 0 and at most $h = O(\log n)$.

(b) (1') The sum of degrees of all the nodes is $n - 1$.

Solution: Actually this applies to any tree. Each edge contributes 1 to the sum-of-degree, and every 1 in the sum-of-degree comes from an edge. Hence, the sum-of-degree is equal to the number of edges, which is $n - 1$.

(c) (2') If the heap is a perfect tree, then an element on the bottom level (i.e. depth=h) must be smaller than 0 elements and greater than h elements. Write a number in each blank.

Solution: This element could be the maxima, thus it could be smaller than no one. And it must be greater than its ancestors except itself, where the number of such nodes is h.

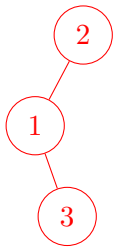
3. (8 points) isBST

(a) (2') If binary tree T satisfies that its in-order traversal is ascending, can we say T is a BST for sure? Answer Yes/No: Yes

Solution: This is one of the definitions of BST.

(b) (2') If binary tree T satisfies that for each node x, we have $x.left.value < x.value < x.right.value$ (if x doesn't have left/right child, omit the corresponding comparison), can we say T is a BST for sure? Answer Yes/No: No

Solution: No. Here is a counter-example. To make it a true definition of BST, we need to make sure that "the entire left subtree of x is smaller than x and the entire right subtree of x is greater than x".



(c) (4') Here is a function that checks whether a given binary tree is a BST. Please complete the function below. Use `x->value` to access the integer value of node `x`. Suppose all the values are in the range (`INT_MIN`, `INT_MAX`).

```
bool isBST(Node *root){
    return isBST_helper(root, INT_MIN, INT_MAX);
}
bool isBST_helper(Node* x, int min, int max){
    if(x == nullptr)
        return true;
    if(______________________||______________________)
        return false;
    return isBST_helper(x->left, ___________________, ___________________) &&
           isBST_helper(x->right, ___________________, ___________________)
}
```

Solution:

```
x->value < min || x->value > max
```

`min, x->value`

`x->value, max`

The `min` and `max` parameters stand for a possible range of the values in the subtree of x If there is any node outside of this range, there must be a contradiction. When recurring to x's children, we need to update the range to make sure that "the entire left subtree of x is smaller than x and the entire right subtree of x is greater than x".