ShanghaiTech University

CS101 Algorithms and Data Structures

Fall 2021

Homework 6

Due date: 23:59, November 7, 2021

1. Please write your solutions in English.
2. Submit your solutions to gradescope.com.
3. Set your FULL NAME to your Chinese name and your STUDENT ID correctly in Account Settings.
4. If you want to submit a handwritten version, scan it clearly. Camscanner is recommended.
5. When submitting, match your solutions to the according problem numbers correctly.
6. No late submission will be accepted.
7. Violations to any of the above may result in zero grade.

1: (12') Multiple Choices

Each question has one or more correct answer(s). Select all the correct answer(s). For each question, you get 0 point if you select one or more wrong answers, but you get 1 point if you select a non-empty subset of the correct answers.
Note that you should write you answers of section 1 in the table below.

Question 1	Question 2	Question 3	Question 4	Question 5	Question 6
C	B D	B D	A D	A	A C

Question 1. *Which of the followings are true?*

(A) For a min-heap, in-order traversal gives the elements in ascending order.

(B) For a min-heap, pre-order traversal gives the elements in ascending order.

(C) For a BST, in-order traversal gives the elements in ascending order.

(D) For a BST, pre-order traversal gives the elements in ascending order.

Question 2. *For a Binary Search Tree (BST), which of the followings are true?*

(A) If we erase the root node with two children, then it will be replaced by the maximum object in its right sub-tree.

(B) The cost for erasing the root node who has two children is $O(n)$.

(C) In a BST with N nodes, it always takes $O(\log N)$ to search for an specific element.

(D) For a BST, the newly inserted node will always be a leaf node.

Question 3. *Suppose we want to use Huffman Coding Algorithm to encode a piece of text made of characters. Which of the following statements are true?*

(A) Huffman Coding Algorithm will compress the text data with some information loss.

(B) The construction of binary Huffman Coding Tree using priority queue has time complexity $O(nlogn)$, where n is the size of the character set of the text.

(C) When inserting nodes into the priority queue, the higher the occurrence/frequency, the higher the priority in the queue.

(D) The Huffman codes obtained must satisfy prefix-property, that is, no code is a prefix of another code.

Question 4. *Which of the following statements are true for an AVL-tree?*

(A) Inserting an item can unbalance non-consecutive nodes on the path from the root to the inserted item before the restructuring.

(B) Inserting an item can cause at most one node imbalanced before the restructuring.

(C) Removing an item in leaf nodes can cause at most one node imbalanced before the restructuring.

(D) Only at most one node-restructuring has to be performed after inserting an item.

Question 5. *Consider an AVL tree whose height is h, which of the following are true?*

(A) This tree contains $\Omega(\alpha^h)$ *nodes, where* $\alpha = \frac{1+\sqrt{5}}{2}$.

(B) This tree contains $\Theta(2^h)$ *nodes.*

(C) This tree contains $O(h)$ *nodes in the worst case.*

(D) None of the above.

Question 6. *Which of the following is TRUE?*

(A) The cost of searching an AVL tree is $O(\log n)$ *but that of a binary search tree is* $O(n)$

(B) The cost of searching an AVL tree is $O(\log n)$ *but that of a complete binary tree is* $O(n \log n)$

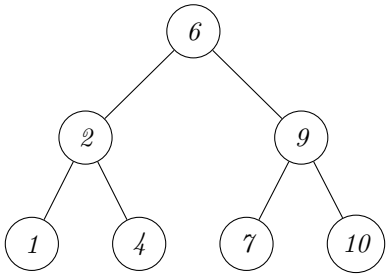
(C) The cost of searching a binary search tree with height h is $O(h)$ *but that of an AVL tree is* $O(\log n)$

(D) The cost of searching an AVL tree is $O(nlogn)$ *but that of a binary search tree is* $O(n)$

2: (3'+3'+2'+3') BST and AVL Tree

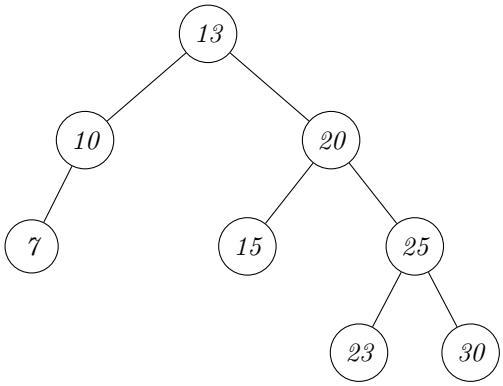
Question 7. *Draw a valid BST of minimum height containing the keys 1, 2, 4, 6, 7, 9, 10.*

Solution:

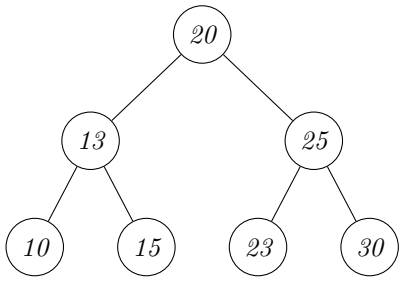


Question 8.

(1) Given an empty AVL tree, insert the sequence of integers $15, 20, 23, 10, 13, 7, 30, 25$ *from left to right into the AVL tree. Draw the final AVL tree.*
Solution:



(2) For the final AVL tree in the question (1), delete 7*. Draw the AVL tree after deletion.*
Solution:



(3) For an AVL tree, define D = the number of left children - the number of right children, for the root. Then what is the maximum of D for an AVL tree with height n ?

Solution:

The maximum of D should be the maximum number of the nodes in the left subtree minus the minimum number of the nodes in the right tree.

$$D_{max} = L_M - R_m$$

Therefore, the height of the left subtree shoud be $n-1$ and the height of the right substree shuold be $n-2$. The number of the nodes in the left substree is maximized when the left substree is a perfect tree. Then we have

$$L_M = 2^n - 1$$

The right subtree should also be an AVL tree. The minimum number of the nodes in the right substree is the minimum number of the nodes in an AVL tree with height $n-2$. Let the minumber number of the nodes in an AVL tree with given heght h be denoted as $F(h)$. Each AVL tree can be seperate into left subtree, right subtree and the root. Both the left substree and the right subtree also have minimal number of nodes. One of the height of the subtree should be $h-1$ and the other should be $h-2$. Therefore, we have:

$$\begin{cases} F(h) = 1 + F(h-1) + F(h-2) \quad (*) \\ F(0) = 1 \\ F(1) = 2 \end{cases}$$

The $(*)$ can also be represented as follows:

$$F(h) + 1 = [F(h-1) + 1] + [F(h-2) + 1]$$

Let $G(h) = F(h) + 1$

$$\begin{cases} G(h) = G(h-1) + G(h-2) \\ G(0) = 2 \\ G(1) = 3 \end{cases}$$

The chacracteristic funciton is

$$\begin{aligned} x^2 &= x + 1 \\ x_1 &= \frac{1+\sqrt{5}}{2} \\ x_2 &= \frac{1-\sqrt{5}}{2} \end{aligned}$$

Then we have

$$\begin{aligned} G(h) &= c_1(x_1)^h + c_2(x_2)^h \\ G(0) &= c_1 + c_2 = 2 \\ G(1) &= \frac{1+\sqrt{5}}{2}c_1 + \frac{1-\sqrt{5}}{2} = 3 \\ c_1 &= \frac{5+2\sqrt{5}}{5}, \\ c_2 &= \frac{5-2\sqrt{5}}{5} \end{aligned}$$

Therefore

$$G(h) = \frac{5+2\sqrt{5}}{5} \cdot \left(\frac{1+\sqrt{5}}{2}\right)^h + \frac{5-2\sqrt{5}}{5} \cdot \left(\frac{1-\sqrt{5}}{2}\right)^h$$
$$F(h) = \frac{5+2\sqrt{5}}{5} \cdot \left(\frac{1+\sqrt{5}}{2}\right)^h + \frac{5-2\sqrt{5}}{5} \cdot \left(\frac{1-\sqrt{5}}{2}\right)^h - 1$$

Then we can calculate R_m

$$R_m = F(n-2) = \frac{5+2\sqrt{5}}{5} \cdot \left(\frac{1+\sqrt{5}}{2}\right)^{(n-2)} + \frac{5-2\sqrt{5}}{5} \cdot \left(\frac{1-\sqrt{5}}{2}\right)^{(n-2)} - 1 = G(n-2) - 1$$

Hence,

$$\begin{aligned} D_{max} &= L_M - R_m \\ &= (2^n - 1) - (G(n-2) - 1) \\ &= 2^n - G(n-2) \\ &= 2^n - \frac{5+2\sqrt{5}}{5} \cdot \left(\frac{1+\sqrt{5}}{2}\right)^{(n-2)} - \frac{5-2\sqrt{5}}{5} \cdot \left(\frac{1-\sqrt{5}}{2}\right)^{(n-2)} \end{aligned}$$

3: (3'+3') Huffman Coding

After you compress a text file using Huffman Coding Algorithm, you accidentally spilled some ink on it and you found that one word becomes unrecognizable. Now, you need to recover that word given the following information:

Huffman-Encoded sequence of that word:

000101001110110100

Frequency table that stores the frequency of some characters:

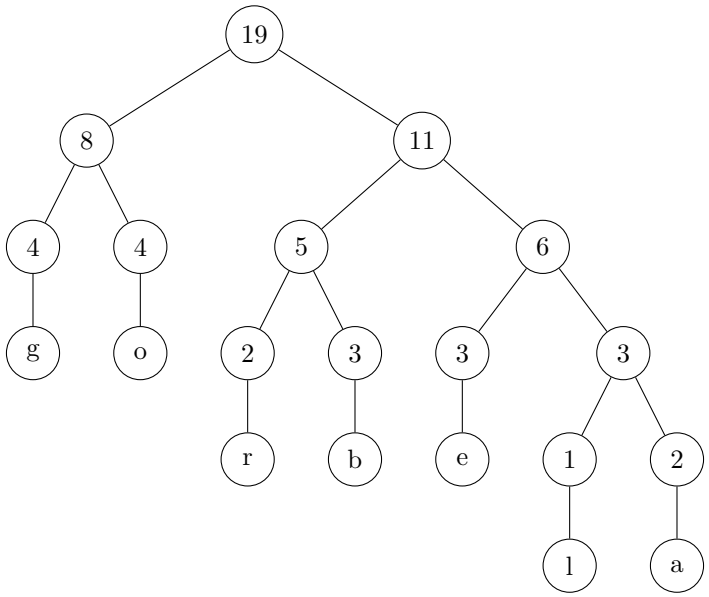
characters	a	b	e	g	l	o	r
frequency	2	3	3	4	1	4	2

Question 9. *Please construct the binary Huffman Coding Tree according to the given frequency table and draw the final tree below.*

Note: The initial priority queue is given as below. When popping nodes out of the priority queue, the nodes with the same frequency follows "First In First Out".

l	a	r	b	e	g	o
1	2	2	3	3	4	4

Solution:



Question 10. *Now you can "decompress" the encoded sequence and recover the original word you lost. Please write the original word below.*

Solution:

googler

4: (9')Only-child

We define that the node is only-child if its parent node only have one children(Note: The root does not qualify as an only child). And we define a function for any binary tree T : $OC(T)$ = the number of only-child node.

Question 11. *(3')Prove the conclusion that for any nonempty AVL tree T with n nodes, OC(T)*$\leq \frac{1}{2}n$.

Solution:

Claim: In an AVL tree, the only-child can only appear on the leaves.

Proof: If an internal nodes I is only-child, then his parent P will be unbalanced. Since I is only-child, then P only has one child. Therefore, P has a substree with height 0. I is internal node, it has at least one substree with height $h(h > 0)$. Therefore, the substree of P containing I has height $h + 1 > 1$, and the difference in height of the two subtree of P is bigger than 1. P is unbalanced. Therefore, the only-child can only be leaf nodes.

Let L be the number of leaf nodes, and N be the be the number of the tree nodes.

Claim: If a binary tree is not a full binary tree, $L \leq \frac{N}{2}$.

Proof: If the tree is a full binary tree, then $N = 2L - 1$, $L \leq \frac{N+1}{2}$. If the tree is not a full binary tree, we add $t(t > 0)$ leave nodes to make it full and apply the relationship between leaves and nodes in full binary tree. $L' = L + t, N' = 2L' - 1, N = N' - t$.

$$\begin{aligned} N &= N' - t \\ &= 2L' - 1 - t \\ &= 2L + 2t - t - 1 \\ &= 2L + t - 1 \geq 2L \\ L &\leq \frac{N}{2} \end{aligned}$$

Therefore, in a binary tree which is not a full binary tree, $L \leq \frac{N}{2}$.

Next, we begin to prove that $OC(T) \leq \frac{1}{2}n$.

We know that only-child node can only be leaf nodes in an AVL tree, and an only-child node can't have sibling node. If T is not a full binary tree, we remove all the leaf nodes that is not an only-child node from T. The tree after removal is denoted as T'. T and T' have the same number of the only-child nodes and T' is not a full binary tree. The leaf nodes of T' equal to all the only-child nodes in T'. The number of leaves in T' is L', then we have

$$OC(T) = OC(T') = L' \leq \frac{1}{2}n$$

If T is a full binary tree, $OC(T) = 0 \leq \frac{1}{2}n$.

Hence, for a nonempty AVL tree T with n nodes, $OC(T) \leq \frac{1}{2}n$

Question 12. *(3') For any binary tree T with n nodes, Is it true that if* $OC(T) \leq \frac{1}{2}n$ *then height*$(T) = O(\log n)$*? If true, prove it. If not, give a counterexample.*

Solution:

It is not true. We can construct the following tree T. T is composed by two parts: a perfect binary tree T_1 with $\frac{n}{2}$ nodes and a tree T_2 whose nodes are all only-child nodes excpte the root nodes with $\frac{n}{2}$ nodes (T_2 has a linked-list-like structure). T_2 is linked to the bottom of T_1. Then T has n nodes and $\frac{n}{2}$ only-child nodes, satisfying $OC(T) \leq \frac{1}{2}n$. T_1 has height $\log \frac{n}{2}$ and T_2 has height $\frac{n}{2}$. Therefore, T has height $\frac{n}{2} + \log \frac{n}{2}$. $height(T) = O(\frac{n}{2} + \log \frac{n}{2}) = O(n) \neq O(\log n)$. Hence, the statement is not true.

Question 13. *(3') For any binary tree T, Is it true that if there are n_0 only-children and they are all leaves, then $height(T) = O(\log n_0)$? If true, prove it. If not, give a counterexample.*

Solution:

It is not true. Let T be a complete tree with height $h(h > 1)$ and $n = 2^{h+1} - 2$ nodes. T has only one only-child. $n_0 = 1$. $height(T) = O(h) = O(\log n) \neq O(\log n_0) = O(1)$. Therefore, the statement is not true.