Before you start:

Homework Files

You can download the starter files for coding as well as this *tex* file (you only need to modify *homework2.tex*) on canvas and do your homework with latex (recommended). Or you can scan your handwriting, convert to pdf file, and upload it to canvas before the due date. If you choose to write down your answers by hand, you can directly download the pdf file on canvas which provides more blank space for solution box.

Submission Form

For homework 2, you need to upload a pdf file in the following format:

- VE281_HW2_[Your Student ID]_[Your name].pdf

Please strictly follow the format given above!!! Everyone who does not obey the format will get **2 points** deduction!!!
Notes: No extra folders (extracting this tar should only give you two files), no space in your name (use underscore(_) instead), no brackets. One example for name of pdf:
VE281_HW2_518370910000_Run_Peng.pdf

Estimated time used for this homework: **3-4 hours.**

0 Student Info (1 point)

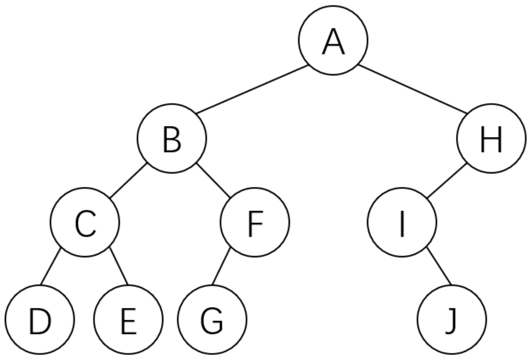
Your name and student id:

Solution: Lan Wang, 519370910084

1 Tree Traversal (26 points)

1.1 Given A Tree (16 points)

Given a binary tree below, please write out the following traversals:



(a) Pre-order depth-first traversal. (4 points)

Solution: A B C D E F G H I J

(b) Post-order depth-first traversal. (4 points)

Solution: D E C G F B J I H A

(c) In-order depth-first traversal. (4 points)

Solution: D C E B G F A I J H

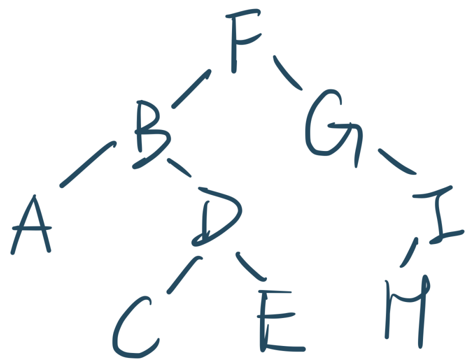
(d) Level-order traversal. (4 points)

Solution: A B H C F I D E G J

1.2 Draw The Tree (10 points)

a) Now we have a specific binary tree, but we only know some of its traversals. Its pre-order traversal is: **FBADCEGIH**, and its in-order traversal is: **ABCDEFGHI**. Then please **draw out the binary tree** and show its **post-order traversal**. (4 points)

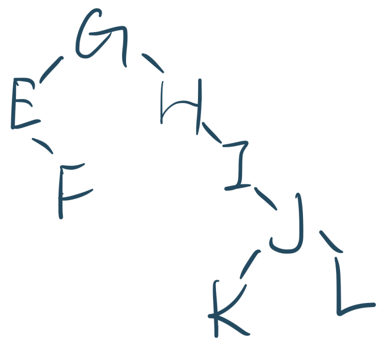
Solution:



post-order traversal: A C E D B H I G F

b) Now we have a specific binary tree, but we only know some of its traversals. Its post-order traversal is: **FEKLJIHG**, and its in-order traversal is: **EFGHIKJL**. Then please **draw out the binary tree** and show its **pre-order traversal**. (4 points)

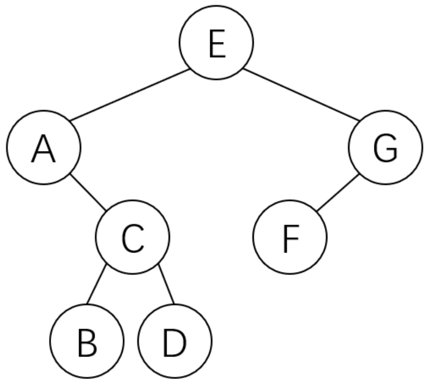
Solution:



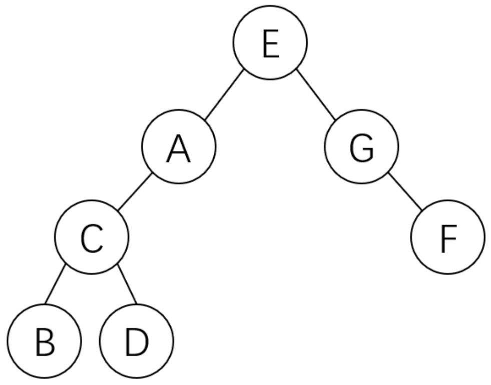
pre-order traversal: G E F H I J K L

c) Now students are required to draw out the binary tree according to the given traversals: Its pre-order traversal is: **EACBDGF**, and its post-order traversal is: **BDCAFGE**. Then please **draw out the binary tree** and show its **pre-order traversal**.

Roihn and Conedx both provide their plot of the unknown trees.



Roihn's Answer



Conedx's Answer

Their answers are totally different, but seem to be both correct. Can you explain why such a case happens? (2 points)

Solution:
When doing pre-order traversal, we visit the node, then left subtree, then right subtree. So when there is only one child and we just write, for example, **XY**, we cannot make sure that **X** is the left or right child of **Y**. Similarly, for **YX** in post-order, we are stil not sure. In this case, we can only know that **C** and **F** are children of **A** and **G** respectively, but cannot know whether they are left or right children.

2 True or False (20 points)

Please judge whether the following statements are **TRUE** or **FALSE**. You can briefly state your reason if you would like to get partial points. Each question takes 2 points.

a) A complete tree is a tree where every node has either 0 or 2 children.

Solution: False.

b) A complete tree is a tree where every level except the last is necessarily filled, and in the last level, nodes are filled in from left to right.

Solution: True.

c) Heapsort has worst-case $\Theta(n \log n)$ time complexity.

Solution: True.

d) Percolate-up has an average-case time complexity of $\Theta(\log n)$.

Solution: True.

e) Dequeue is done by simply removing the root.

Solution: False.

f) Enqueue in a min-heap is done by inserting the element at the end, and then calling percolate-up.

Solution: True.

g) $[2, 13, 8, 16, 13, 10, 40, 25, 17]$ is a representation of a min-heap.

Solution: True.

h) $[3, 5, 6, 7, 12, 15, 14, 9, 10, 11]$ is a representation of a min-heap.

Solution: False.

i) Proceeding from the bottom of the heap to the top, while repeatedly calling percolateDown() can initialize a min-heap.

Solution: True.

j) Proceeding from the top of the heap to the bottom, while repeatedly calling percolateUp() can not initialize a min-heap.

Solution: False.

3 Hea———p! (23 points)

Consider a min-heap represented by the following array:

$$\{63, 74, 67, 85, 91, 94, 72, 88\}$$

Perform the following operations using the algorithms for binary heaps discussed in lecture. Ensure that the heap property is restored at the end of every individual operation.
For the following operations, please briefly describe what and how you use the given functions: **percolateUp()** and **percolateDown()**, and show the result of the heap after each operation in either tree form or array form.

a) Push the value of 60 into this min-heap. (4 points)

Solution:

Add 60 to the end:
{63, 74, 67, 85, 91, 94, 72, 88, 60}

To maintain the property, call **percolateUp()**:
{63, 74, 67, 60, 91, 94, 72, 88, 85}
{63, 60, 67, 74, 91, 94, 72, 88, 85}
{60, 63, 67, 74, 91, 94, 72, 88, 85}

b) Push the value of 79 into this min-heap. (4 points)

Solution:

Add 60 to the end:
{60, 63, 67, 74, 91, 94, 72, 88, 85, 79}

To maintain the property, call **percolateUp()**:
{60, 63, 67, 74, 79, 94, 72, 88, 85, 91}

c) Update element 85 to have a value of 58 (Suppose you have the access to each element). (5 points)

Solution:

85 -> 58:
{60, 63, 67, 74, 79, 94, 72, 88, 58, 91}

To maintain the property, call **percolateUp()**:
{60, 63, 67, 58, 79, 94, 72, 88, 74, 91}
{60, 58, 67, 63, 79, 94, 72, 88, 74, 91}
{58, 60, 67, 63, 79, 94, 72, 88, 74, 91}

d) Remove the min element from the heap. (5 points)

Solution:

Copy the root (58) and return. Cut the last leaf and copy it to the root:
{91, 60, 67, 63, 79, 94, 72, 88, 74}

To maintain the property, call **percolateDown()**:
{60, 91, 67, 63, 79, 94, 72, 88, 74}
{60, 63, 67, 91, 79, 94, 72, 88, 74}
{60, 63, 67, 74, 79, 94, 72, 88, 91}

e) Update element 67 to have a value of 96 (Suppose you have the access to each element) (5 points)

Solution:

67 -> 96:
{91, 60, 96, 63, 79, 94, 72, 88, 74}

To maintain the property, call **percolateDown()**:
{91, 60, 72, 63, 79, 94, 96, 88, 74}

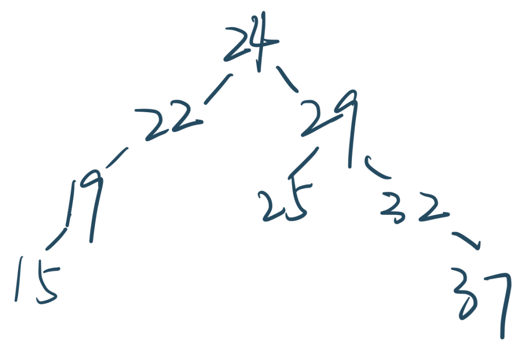
4 Binary Search Tree (30 points)

4.1 Simple simulation (14 points)

Perform the following operations to construct a binary search tree. Show the result of the BST after each operation in either tree form or array form.

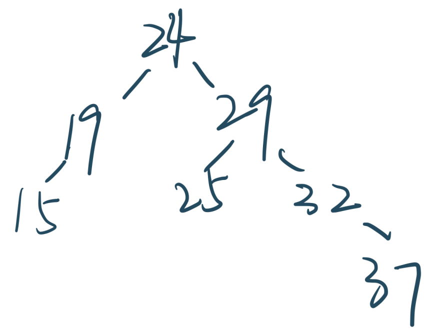
a) Insert 24, 29, 22, 25, 19, 32, 15, 37 (3 points)

Solution:



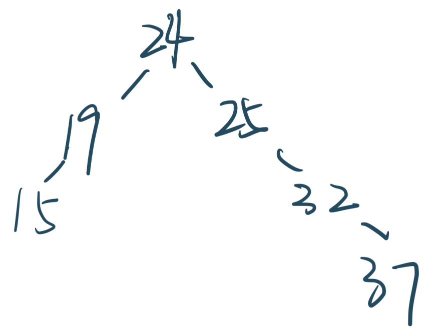
b) Delete 22 (3 points)

Solution:



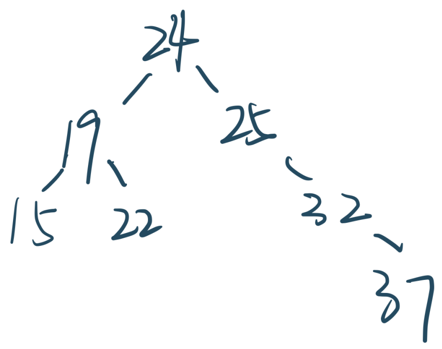
c) Delete 29 (3 points)

Solution:



d) Insert 22 (3 points)

Solution:



e) What is the inorder predessor of node 25? What is the inorder sucessor of node 32? (2 points)

Solution:

inorder predessor of node 25: 24
inorder sucessor of node 32: 37

4.2 Let's BST! (16 points)

a) Suppose that in a binary search tree, when a node with two children is deleted, it is replaced by its inorder successor. Given a pointer to the node to be deleted, what is the time complexity of finding the inorder successor in the average case, if the tree contains n nodes? Briefly explain your answer. (4 points)

Solution:

$O(\log n)$
Assume that the node we are going to delete is a subtree containing m elements (including itself). Then it takes average $O(\log m)$ to find its inorder successor.
To calculate the average time complexity regarding n, consider the average number M of elements that a subtree of a n-element tree can contain.
Consider a perfect BST, with n elements and height h. We can find that:

$$M = \frac{2^h}{2^h - 1}(h + \frac{1}{2^h} - 1) = O(h) = O(\log n)$$

Then, the average time complexity in this case is: $O(\log M) = O(\log n)$.
Among all the possible cases, the more unbalanced, the less likely that it will occur. So roughly speaking, the average case should be balanced, nearly a perfect BST. So the average time complexity among all cases should be $O(\log n)$.

b) Suppose that you want to insert 12 distinct elements into a binary search tree. How many worst-case trees are possible for these 12 elements? (6 points)

Solution:

In the worst case, the tree becomes a linked list (each node has only one child or none). So the root should be the smallest or the largest element, or it will have two children. And then, each following node should be the largest or smallest element among the remained elements, or there will be two children. For example, for {1, 2, 5}, if we insert 4, indicating that 5 is not the smallest or largest element in the remained elements, then 2 will have two children. Then we have:

$$N = 2^{11}$$

possible worst-case trees. Each node has two choices, except for the last one.

c) Consider a tree that satisfies the following conditions:

1. The tree is a binary search tree of integers.
2. The number of elements in the root node's left and right subtrees are the same.
3. There are no duplicate values in the tree.
4. The first element of an inorder traversal of the tree is 11.
5. The last element of an inorder traversal of the tree is 24.
6. The last element of a postorder traversal of the tree is 16.

What is the largest possible integer you can attain by summing up all the values in a tree that satisfies the above constraints? (6 points)

Solution:

According to 4, 5, 6, we can know that the smallest element is 11, the largest is 24, and the root is 16.
Since it's a BST, and no duplicate values, all the elements in the root's left subtree should be smaller than 16, and all the elements in the root's right subtree should be bigger than 16. According to 1, we just consider integers. There are 5 integers in $[11, 16)$ and 8 integers in $(16, 24]$. According to 2, and to make the sum be the largest, we take the largest 5 elements in both side.
The sum is:

$$11 + 12 + 13 + 14 + 15 + 16 + 20 + 21 + 22 + 23 + 24 = 191$$

Reference

Assignment 3, VE281, FA2020, UMJI-SJTU.