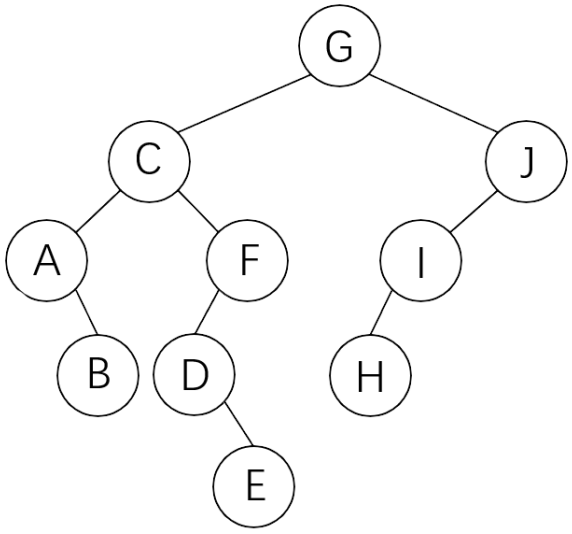
1 Tree Traversal (27 points)

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: GCABFDEJIH

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

Solution: BAEDFCHIJG

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

Solution: ABCDEFGHIJ

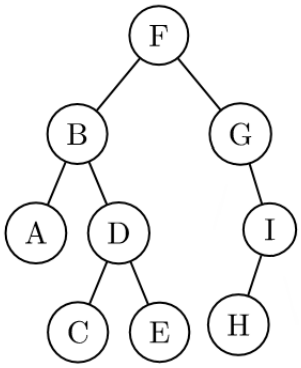
(d) Level-order traversal. (4 points)

Solution: GCJAFIBDHE

1.2 Draw The Tree (11 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)

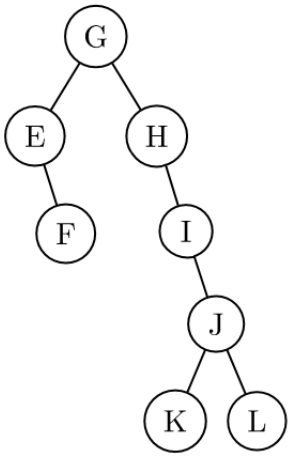
Solution:



post-order traversal: **ACEDBHIGF**

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)

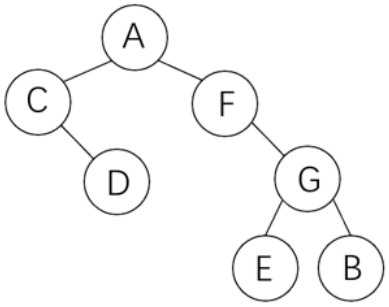
Solution:



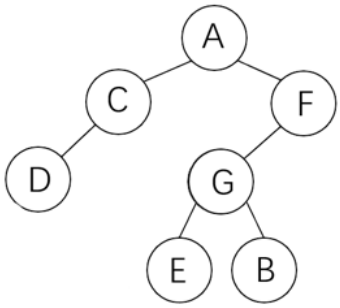
pre-order travesal: **GEFHIJKL**

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

William and Coned both provide their plot of the unknown trees.



Coned's answer



William's answer

Their answers are totally different, but seem to be both correct. Can you explain why such a case happens? (3 points) 如果节点只有一个子节点，我们不知道该子节点是左孩子还是右孩子

Solution:
It is because that only knowing the pre-order and post-order cannot guarantee us with the definite structure of a binary tree. Pre-order traversal has an order of node-left-right and post-order traversal has an order of left-right-node. If the node only has one child, we have no information about whether it is its left child or right child. Therefore, there are many possible structures only given the pre-order traversal and the post-order traversal of the binary tree.

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. No deduction if the solution is right without explanation. 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. Another node should be used to replace the root after deleting the root.

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

Solution: True.

g) $[1, 13, 17, 23, 14, 16, 20, 35]$ is a representation of a min-heap.

Solution: False. 16 is the left child of 17, while $16 < 17$.

h) $[83, 47, 9, 22, 15, 7, 1, 20]$ is a representation of a **max-heap**.

Solution: True.

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 Min Heap (23 points)

Consider a min-heap represented by the following array:

$$\{2, 3, 8, 16, 46, 34, 42, 35, 26\}$$

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, tell how many percolations(either up or down) are needed and show the result of the heap after each **percolation** in either tree form or array form.

a) Push the value of 1 into this min-heap. (4 points)

> **Solution:** {1,2,8,16,3,34,42,35,26,46}
> 3 swaps, 1 percolation.

b) After a), push the value of 5 into this min-heap. (4 points)

> **Solution:** {1,2,8,16,3,34,42,35,26,46,5}
> 0 swap, 1 percolation.

c) After b), update element 2 to have a value of 37 (Suppose you have the access to each element). (5 points)

> **Solution:** {1,3,8,16,5,34,42,35,26,46,37}
> 2 swaps, 1 percolation.

d) After c), remove the min element from the heap. (5 points)

> **Solution:** {3,5,8,16,37,34,42,35,26,46}
> 2 swaps, 1 percolation.

e) After d), update element 26 to have a value of 4 (Suppose you have the access to each element) (5 points)

> **Solution:** {3,4,8,5,37,34,42,35,16}
> 2 swaps, 1 percolation.

Notes: Here actually when I am designing this problem, I intend to let you write out the swap numbers for each operation but due to my mistake, the problem is not described as I want. But writing out **1 percolation** is ok for this problem. It is also a good thing to let you be clear that we will only need 1 percolation for each operation :)
Here writing either the right swap amount or the right percolation amount will not lead to any deducti

4 Binary Search Tree (30 points)

4.1 Simple simulation (16 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:** {24,22,29,19, ,25,32,15, , , , , , ,37}

b) Delete 22 (3 points)

> **Solution:** {24,19,29,15, ,25,32, , , , , , , ,37}

c) Delete 29 (3 points)

> **Solution:** {24,19,25,15, , ,32, , , , , , , ,37} or {24,19,32,15, ,25,37}

d) Insert 22 (3 points)

> **Solution:** {24,19,25,15,22, ,32, , , , , , , ,37} or {24,19,32,15,22,25,37}

e) What is the in-order predecessor of node 25? What is the in-order successor of node 32? (4 points)

> **Solution:** 24 is the in-order predecessor while the in-order successor is 32 no matter which tree you come up with after d).

4.2 Better than linear selection? (14 points)

After learning the application of the BST, Coned thinks that with BST, rank search can be done either in a faster way or with less space compared with the linear selection algorithms introduced in the lecture.

a) Compare BST rank search with deterministic selection algorithm in terms of time complexity and space usage. Assume that for each node, a variable *leftsize* is maintained as introduced in the lecture. (8 points)

> **Solution:** In terms of space complexity, since we implement the BST with a variable $leftSize$ for each node (as well as the pointers), we will have an extra $O(n)$ space usage (excluding the data themselves), which is in accordance with deterministic linear selection.
>
> In terms of time complexity, since the average time complexity for rank search in BST is just $log(n)$ while the average time complexity for deterministic search is $O(n)$, BST is more efficient. Notice that for the worst case, rank search in BST also has a time complexity of $O(n)$.

b) William, our master of algorithm, has a different idea again. He states that for the linear selection algorithm, the input can be arbitrary array while for BST rank search, you have to first build up a binary search tree, which will also take some time. Therefore, the time complexity of BST rank search is not better than $O(n)$. Is he right? (6 points)
Assume that we are working on a fixed array.

Solution: William is wrong again :) It is right that we will need a $O(nlogn)$ time for initialization, but according to the amortized analysis, such time cost will be distributed to each operation. It will lead to a time complexity of $O(\frac{nlogn}{m})$, where m is how many times the rank search is carried out. Since we are working on a fixed array, n is actually a constant and with m becomes very large, it is equal to $O(1)$, which means the initialization time cost is $O(1)$.

Reference

Assignment 3, VE281, FA2020, UMJI-SJTU.
Homework 2, VE281, SU2021, UMJI-SJTU.