Before you start:

Homework Files

You can download the starter files for coding as well as this *tex* file (you only need to modify *homework3.tex*) on canvas and do your homework with latex. 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 3, there are only one part of submission, which is a pdf file as your solution named as VE281_HW3_[Your Student ID]_[Your name].pdf uploaded to canvas.
Estimated time used for this homework: **3-4 hours.**

Great credits to 2020FA VE281 TA Group and enormous thanks to 2021SU VE281 TA Roihn!!!

0 Student Info (0 point)

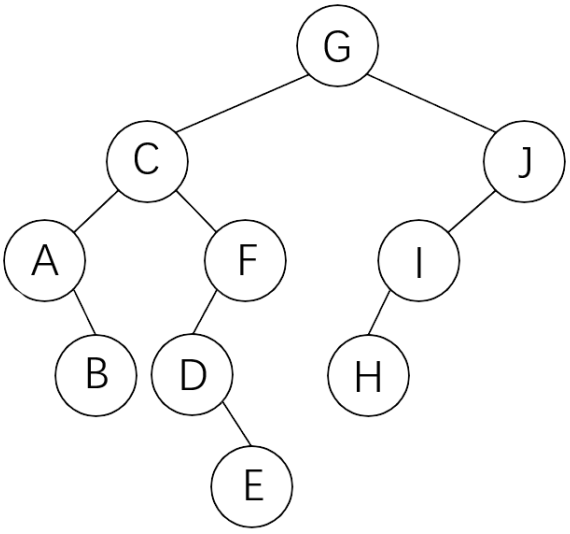
Your name and student id:

Solution: 陶思郡 Sijun Tao 519021910906

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

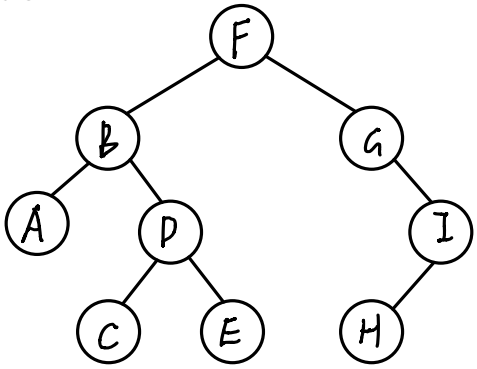
(d) Level-order traversal. (4 points)

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)

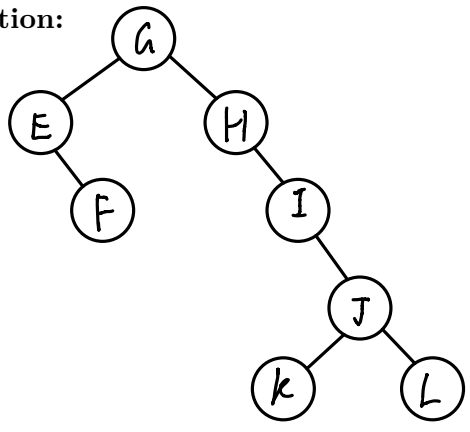
Solution:



post-order: 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)

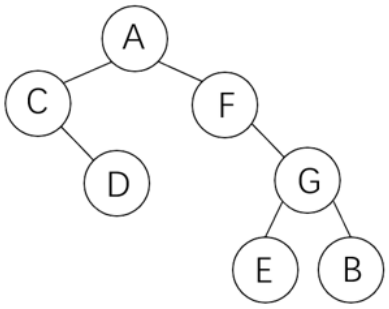
Solution:



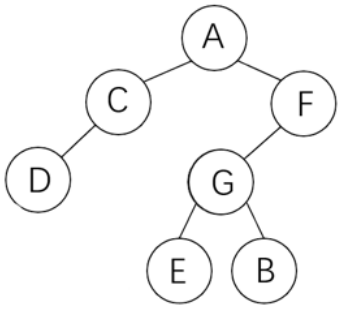
pre-order: GEFHIJKL

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)

Solution:

The position of the previous node of the root node in post-order traversal is adjacent to that of the root node in pre-order traversal. As a result, we cannot decide which is the left subtree and which is the right subtree.

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. It is a proper tree. However, proper tree is not necessarily a complete tree.

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. Swap the rightmost leaf node with the root node and percolate down it.

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. 17 and 16 don't satisfy the relationship.

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)

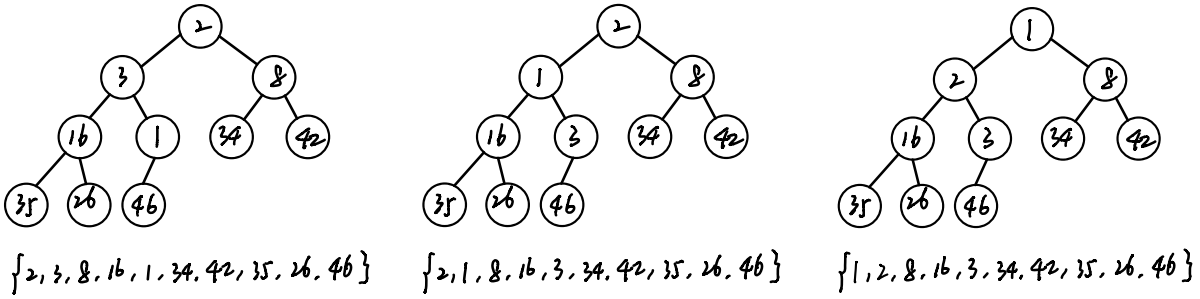
Consider a min-heap represented by the following array:

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

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: number of percolations : 3.

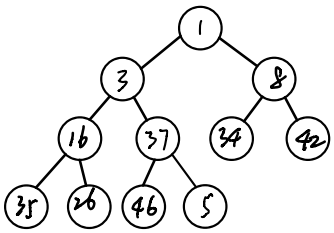


b) After a), push the value of 5 into this min-heap. (4 points)

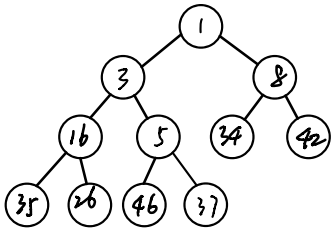
Solution: number of percolations : 0

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

Solution: number of percolations : 2.



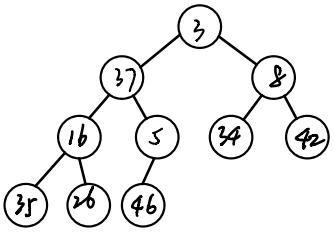
{1, 3, 8, 16, 37, 34, 42, 35, 26, 46, 5}



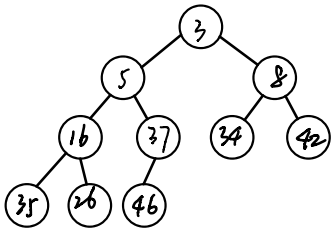
{1, 3, 8, 16, 5, 34, 42, 35, 26, 46, 37}

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

Solution: number of percolations : 2.



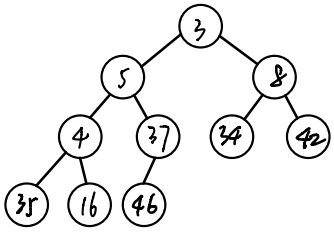
{3, 37, 8, 16, 5, 34, 42, 35, 26, 46}



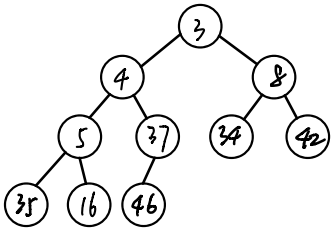
{3, 5, 8, 16, 37, 34, 42, 35, 26, 46}

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

Solution: number of percolations : 2.



{3, 5, 8, 4, 37, 34, 42, 35, 16, 46}



{3, 4, 8, 5, 37, 34, 42, 35, 16, 46}

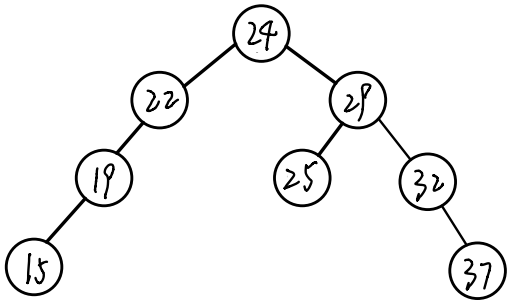
4 Binary Search Tree (30 points)

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.

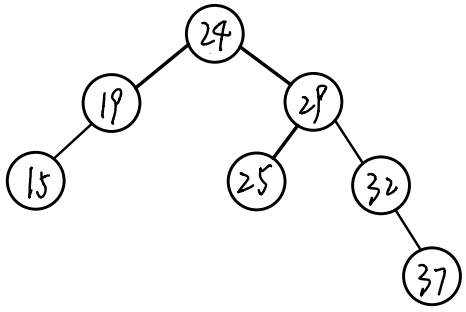
a) Insert 24, 29, 22, 25, 19, 32, 15, 37 (3 points)

Solution:



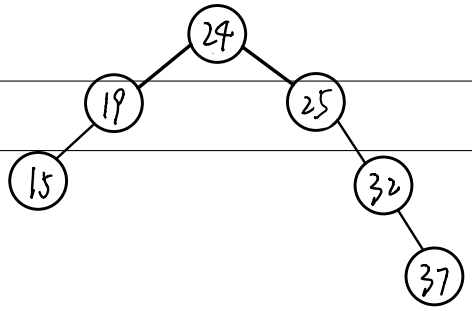
b) Delete 22 (3 points)

Solution:



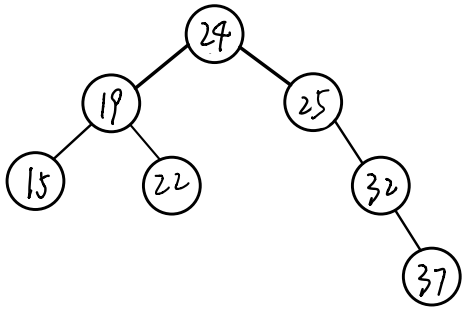
c) Delete 29 (3 points)

Solution:



d) Insert 22 (3 points)

Solution:



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

Solution: in-order : 15, 19, 22, 24, 25, 32, 37
in-order predecessor of node 25 : 24
in-order successor of node 32 : 37

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 briefly. Assume that for each node, a variable *leftsize* is maintained as introduced in the lecture. (8 points)

Solution:
① Deterministic selection algorithm has a time complexity of $O(n)$ and a space usage of $O(n)$

② BST rank search has a time complexity of $O(\log n)$ and a space usage of $O(\log n)$

$\Rightarrow$ BST rank search is faster and has fewer space usage.

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 average 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: Yes.

the average time complexity of building up a binary search tree is

$$n\,O(\log n) = O(n\log n)$$

Thus, the total average time complexity of BST rank search is

$$O(n\log n) + O(\log n) = O(n\log n)$$

$\Rightarrow$ Considering building up a BST tree, the time complexity is $O(n\log n)$, which is not better than $O(n)$.

Reference

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