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

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.**

0 Student Info

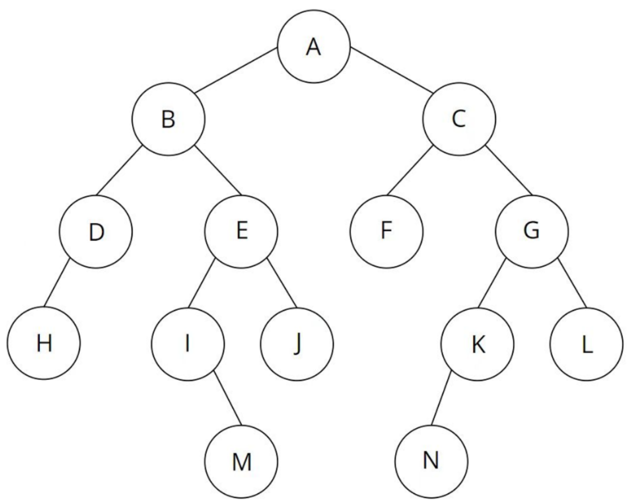
Your name and student id:

Solution:

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:

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

Solution:

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

Solution:

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

Solution:

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: **GCABFEDIHKJ**, and its in-order traversal is: **ABCDEFGHIJK**. Then please **draw out the binary tree** and show its **post-order traversal**. (4 points)

Solution:

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

Solution:

c) After finishing the previous two questions, our smart Mr. Blue Tiger decides to publish paper title **New discovery!! Any pair of 2 distinctive DFS sequences has one and only one corresponding binary tree.** (distinctive here means following different order of DFS) However, our strong TAA Chengyu does not seem to agree with Mr. Blue Tiger. He decides to publish another paper to explain why Mr. Blue Tiger's statement is wrong. Could you briefly explain the reasons? A more detailed statement should be made. (Hint: This problem is related to different combinations of distinctive DFS sequences, i.e. which two out of three orders are picked as known sequences)

Solution:

2 Heap (23 points)

Consider a min-heap represented by the following array:

$$\{18, 25, 32, 77, 86, 35, 93, 80\}$$

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 20 into this min-heap. (4 points)

Solution:

b) Push the value of 11 into this min-heap. (4 points)

Solution:

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

Solution:

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

Solution:

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

Solution:

3 BST Basics (26 points)

3.1 Simple Simulation (10 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.

i) Insert 24, 290, 22, 25, 19, 32, 15, 37 (2 points)

Solution:

ii) Delete 22 (3 points)

Solution:

iii) Delete 29 (3 points)

Solution:

iv) Insert 22 (2 points)

Solution:

3.2 Basic Questions (16 points)

Please finish the multiple choice questions below, no explanation is needed.

i) The following numbers are inserted into an empty binary search tree in the given order: 10, 1, 3, 5, 15, 12, 16. What is the **height** of the binary search tree? (4 points)

A. 2

B. 3

C. 4

D. 5

Solution:

ii) Suppose we want to delete a node with its left and right child as non-empty in a binary search tree, we may need to find the largest element in its left subtree (inorder predecessor) or the smallest element in its right subtree (inorder successor). Which of the following is **true** about the inorder successor? (4 points)

A. Inorder successor is always a leaf node.

B. Inorder success is always either a left node or a node with empty left child.

C. Inorder successor may be an ancestor of the node.

D. Inorder successor is always either a leaf node or a node with empty right child.

Solution:

iii) A binary search tree is used to locate the number 43. Which one of the following probe sequence is **not possible**? (4 points)

A. 61, 52, 14, 17, 40, 43

B. 10, 65, 31, 48, 37, 43

C. 81, 61, 52, 14, 41, 43

D. 17, 77, 27, 66, 18, 43

Solution:

iv) Consider the following statements:

I. The smallest element in a max-heap is always at a leaf node.

II. The second largest element in a max-heap is always a child of the root node.

III. A max-heap can be constructed from a binary search tree in $\Theta(n)$ time.

IV. A binary search tree can be constructed from a max-heap in $\Theta(n)$ time.

Which of the above statements are **true**? (4 points)

A. I, II and III

B. I, II and IV

C. I, III and IV

D. I, II, III, and IV

Solution:

4 Binary Search Tree Analysis (12 points)

4.1 BST Better than List? (4 points)

After learning binary search tree, Ssy thinks that BST can perform much better than list. As he is still trying to finish project2, he immediately thinks of an idea to combine hash table with BST. In separate chaining strategy, he uses binary search tree instead of list inside each bucket. Do you think there are any **advantages or disadvantages** of this strategy? Briefly explain your idea.

Solution:

4.2 Simple Application (8 points)

Suppose now you're going to implement an algorithm which accepts a root node of a binary search tree and two values. These two values are known to present in the BST. The algorithm will print the value of a node which is the least common ancestor of these 2 elements. Please finish the algorithm below.

Note: A node X is said to be the common ancestor of node A and B means both A and B are in the subtree (either left or right) of X. A least common ancestor is a common ancestor such that all other common ancectors are its ancestors.

```
void Find_LCA (Node *root, int a, int b) {
    while (root != null) {
        if( /*write your code here*/ ) {
            root = root.left;
        }
        else if( /*write your code here*/ ) {
            root = root.right;
        }
        else {
            break;
        }
    }
    std::cout << root.value << std::endl;
}
```

5 BST Interesting Questions (13 points)

5.1 Perfect Balance (8 points)

Propose an algorithm which inserts a set of keys into an initially empty binary search tree such that the tree produced is equivalent to binary search. This means the sequence of compares done in find() is the **same** as the sequence of compares used by binary search for the same key. Also analyze time complexity of this algorithm.
Hint: In binary search, we first compare current key with $\frac{n}{2}$th key in the array, then compare current key with $\frac{n}{4}$th key or $\frac{3n}{4}$th key in the array, etc.

Solution:

5.2 BST with Duplicate keys (5 points)

The binary search tree we introduced in the class does not support duplicate keys. By some modifications, we can make BST support duplicate keys. A simple approach is to change the rule of BST: the key smaller or equal to the current key goes to the left subtree, the key greater than the current key goes to the right subtree. However, a better solution is to add an additional field to each node called *count*, which is the number of current key in the binary search tree. Briefly introduce how to implement **insert** and **remove** in this kind of BST and explain why this approach is better than the first one.

Solution: