COMP2119 Introduction to Data Structures and Algorithms
Assignment 4 - Trees and Sorting Algorithms

Due Date: Nov 29, 2024 7:00pm

Question 1 - Binary Search Tree [30%]

The *TreeNode* structure below represents one node in a binary search tree:

```
class TreeNode:
    Function Initialize(value):
        value = value
        left = None
        right = None
        parent = None
```

(a) [10%] Write pseudocode to implement a binary search tree. You only need to implement the insert method to support adding tree nodes into the tree.

(b) [20%] Write pseudocode to complete the tasks below by using the binary search tree you constructed in part (a):

(i) Design an algorithm to find the lowest common ancestor of two given nodes in the BST. The lowest common ancestor between two nodes a and b is the lowest node (starting from tree root) that has both a and b as descendants.

(ii) Design an algorithm to check if a BST is an AVL tree, i.e., for each node in a BST, the height of its left and right subtrees differ by no more than one.

Remarks:

- We assume the values of all tree nodes are distinct, i.e., there are no two nodes containing the same value.
- You may assume the BST always have more than one node.
- A node can be a descendant of itself.

Question 2 - AVL Tree [20%]

Starting from an empty tree, draw the resulting AVL tree after each of the following sequences of operations:

(1) [5%] Insert the values: 30, 20, 40, 10, 25
(2) [10%] Insert the value: 5
(3) [5%] Delete the value: 30

Remarks:
For tree node deletion, you should replace the deleted node value by it's immediate successor.

Question 3 - Sorting [25%]

(a) [10%] Consider the pseudocode of QuickSort below:

Algorithm 3.1 Pseudocode of QuickSort

```
1: function QuickSort(A, p, r)                    ▷ Conduct quicksort on array A[p...r]
2:     if (p < r) then
3:         q = Partition(A, p, r)
4:         QuickSort(A, p, q)
5:         QuickSort(A, q + 1, r)
6:     end if
7: end function
8:
9: function Partition(A, p, r)                    ▷ Partition the array segment A[p...r]
10:    v = pivot to be chosen in each sub-question
11:    i = p - 1
12:    j = r + 1
13:
14:    while (True) do
15:        do
16:            j = j - 1
17:        while (A[j] > v)
18:
19:        do
20:            i = i + 1
21:        while (A[i] < v)
22:
23:        if (i < j) then
24:            Swap(A[i], A[j])          ▷ Swap the position of A[i] and A[j] in O(1) time
25:        else
26:            return j
27:        end if
28:    end while
29: end function
```

Suppose we use the QuickSort function above to sort an array of size n (Assume n is large enough). Choose answer for each question below from $\{O(\log n)$, $O(n)$, $O(n \log n)$, $O(n^2)\}$. You do not need to prove your answer.

(i) What is the asymptotic running time of the Partition function when all elements in the array are equal?

(ii) What is the asymptotic running time of the QuickSort function when all elements in the array are equal, if the pivot-selection strategy picks the **leftmost** element in the range-to-be-sorted?

(iii) What is the asymptotic running time of the QuickSort function, if the array is already sorted and the pivot-selection strategy picks the **leftmost** element in the range-to-be-sorted?

(iv) What is the asymptotic running time of the QuickSort function, if the array is already sorted and the pivot-selection strategy picks the 2^{nd} **largest** element in the range-to-be-sorted?

(v) What is the asymptotic running time of the QuickSort function, if the array is already sorted and the pivot-selection strategy picks the **middle** element in the range-to-be-sorted?

(b) [15%] Given an array S with n integers, in which their values must be either {-1, 0, 1}.

(i) Write the pseudocode of Reorder(S) that sorts the array in ascending order in $O(n)$ time and $O(1)$ extra space.

Remarks:

- You can only use Swap($S[i]$, $S[j]$) operation to swap elements in the array.
- You cannot count the number of appearance of -1, 0, 1, and then assign them in the array one by one.

(ii) Verbally explain how your algorithm works, and analyze your algorithm's worst case time complexity.

Question 4 - Find Median from Data Stream [25%]

The median is the middle value in an ordered integer list. If the size of the list is even, the median is the mean of the two middle values. For example, for arr = [2,3,4], the median is 3; for arr = [2,3], the median is (2 + 3) / 2 = 2.5.

Suppose there is a *MedianFinder* class works as follows:

1. *MedianFinder*() initializes the *MedianFinder* object called *medianFinder* with two class functions *addNum* and *findMedian*.
2. *medianFinder.addNum*(int *num*) adds the integer *num* from the data stream to the data structure.
3. *medianFinder.findMedian*() returns the median of all elements so far.

Example

```
medianFinder = MedianFinder();
medianFinder.addNum(1); // arr = [1]
medianFinder.addNum(2); // arr = [1, 2]
medianFinder.findMedian(); // return 1.5 (i.e., (1 + 2) / 2)
medianFinder.addNum(3); // arr[1, 2, 3]
medianFinder.findMedian(); // return 2.0
```

Task

Design a *MedianFinder* class as described above in pseudocode. Suppose we will insert n numbers into *medianFinder*. Satisfying ***one*** of the following worst case time complexity requirements will earn the corresponding scores. Analyse how your implementation fulfill the task of finding the median and how the two functions of your implementation (*addNum* and *findMedian*) satisfy the time complexity.

1. *addNum* $O(n \log n)$ or $O(n^2)$; *findMedian* $O(1)$ [15%]
2. *addNum* $O(\log n)$; *findMedian* $O(n \log n)$ [15%]
3. *addNum* $O(\log n)$; *findMedian* $O(1)$ [25%] [Hint: Use two priority queues]

Submission

Please submit your assignment (one **PDF** file) to moodle by the deadline. Make sure the content is readable. Feel free to contact the TAs if you encounter any difficulty in this assignment. We are happy to help you!

Enquiries for Grading

If you have any issues regarding to the grading of the assignment, please contact the following graders for help:

Q1, 2: Ye Tian (Email: yetiansh@connect.hku.hk)
Q3: Jeff Siu (Email: jeffsiu1@hku.hk)
Q4: XiuXian Guan (Email: guanxx@connect.hku.hk)