CS2040S: Data Structures and Algorithms

Tutorial Problems for Week 9: BST and Other Ordered Maps

For: 11 Oct 2021, Tutorial 7

Problem 1. True or False?

Decide whether each of the following statements is true or false. Provide a short justification for your choice.

Problem 1.a. (AY19/20 Sem 1 Final Exam) Given any AVL tree of height 4, deleting any vertex in the tree will not result in more than 1 rebalancing operation (not rotation but rebalancing operations!).

Problem 1.b. The minimum number of vertices in an AVL tree of height 5 is 21.

Problem 1.c. In a tree, if for every vertex x that is not a leaf, $x.left.key < x.key$ if x has a left child and $x.key < x.right.key$ if x has a right child, the tree is a BST.

Problem 2. In-Order Traversal

The in-order traversal visits nodes in a BST in sorted order. The following implementation of in-order traversal that prints the nodes in a balanced BST in sorted order has been proposed.

Algorithm 1 In-Order Traversal

```
1: procedure InOrderTraversal(T)                    ▷ bBST T is supplied as input
2:     currentNodeValue ← T.findMin()
3:     while currentNodeValue ≠ −1 do
4:         output currentNodeValue
5:         currentNodeValue ← T.successor(currentNodeValue)
6:     end while                                    ▷ successor returns -1 if there is no successor
7: end procedure
```

Answer the following questions assuming that T is a balanced BST.

Problem 2.a. What is the running time of Algorithm 1?

Problem 2.b. Propose modifications to the *successor* function such that Algorithm 1 runs in $O(n)$ time.

Problem 3. Rank and Select

A node x has rank k in a BST if there are $k-1$ nodes that are smaller than x in the BST. The *rank* operation finds the rank of a node in a BST.

Problem 3.a. Describe an algorithm that finds the rank of a given node in the BST in $O(h)$ time, where h is the height of the BST.

The *select* operation returns the node with rank k in the BST.

Problem 3.b. Describe an algorithm that finds the node with rank k in the BST in $O(h)$ time, where h is the height of the BST.

Problem 4. Transmission System

There are n servers in a line, numbered from 1 to n. A server can either be enabled or disabled, only enabled servers will send data. Server i has been configured in a way such that it can only send data directly to server $i+1$, $1 \leq i < n$. For $i < j$, server i can send data indirectly to server j, as long as both servers are enabled, and all servers between them (if any) are all enabled as well. Initially, all servers are enabled.

You need to support q of the following three types of operations:

- $Enable(i)$: Enable the ith server. If the ith server is already enabled, nothing happens.
- $Disable(i)$: Disable the ith server. If the ith server is already disabled, nothing happens.
- $Send(i, j)$: Return $true$ if it is possible to send data from server i to j, $false$ otherwise.

Describe the most efficient algorithm you can think of for each of the three types of operations. What is the running time of your algorithm for each of the three types of operations?

Problem 5. Lowest Common Ancestor (CS2040S AY19/20 Sem 1, Quiz 2)

The Lowest Common Ancestor (LCA) of two nodes a and b in a BST is the node furthest from the root that is an ancestor of both a and b. For example, given the Binary Tree shown in Fig 1, the LCA for 2 and 6 is 5. The LCA of 12 and 2 is 10. Note that a node is an ancestor of itself. That means the LCA of 5 and 7 is 5. Similarly, the LCA of 18 and 14 is 18.

Consider a **Balanced Binary Search Tree** (bBST) T containing n nodes with *unique* keys, where **nodes do**

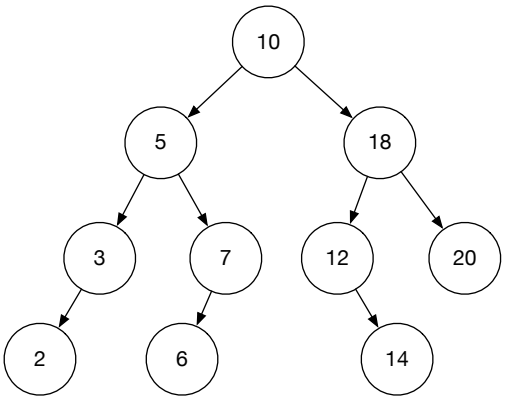

Figure 1: An example BST.

***not* have parent pointers** (i.e. given a node, you *cannot* find its parent in $O(1)$ time).

Describe the most efficient algorithm you can think of to find the LCA of two given nodes in the bBST. What is the running time of your algorithm?

Problem 6. Does this prefix exist

Given a set S of N strings which are all of length L, answer the following query:

- $GotPrefix(k)$: return true if string k is a prefix of some string in S, return false otherwise.

Give an algorithm to do so in $O(k.length)$ time after spending at most $O(N * L)$ time preprocessing S.