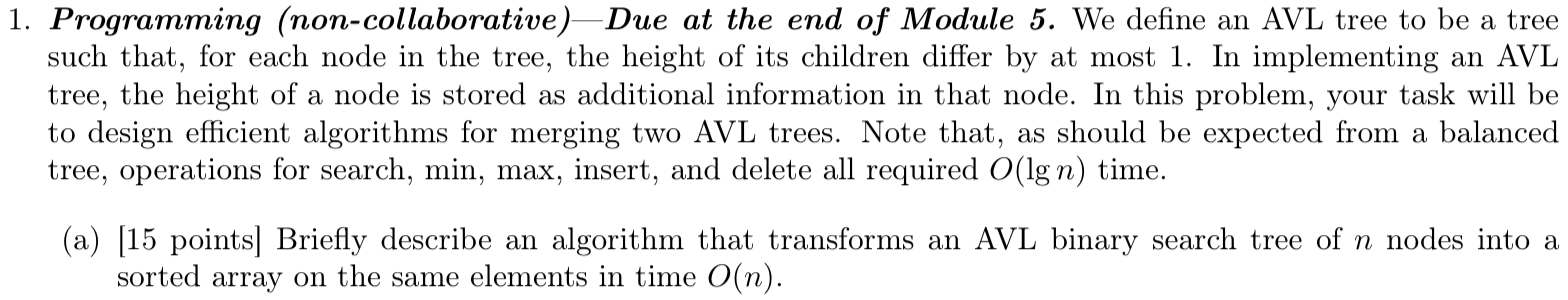
Homework #3

Student: Alice (Yuling) Xiong

Course: Foundations of Algorithms

Date: Oct 4, 2015

Cell: 312-646-9280



1. Starting from the root, we can do an in-order traversal recursively:

* Travel the left subtree if left subtree is not null
* handle and insert into your array
* Travel the right subtree if right subtree is not null

Since it needs to go through each value in the AVL tree, so it takes time linear in the size of AVL tree O(n). (see pseudo code below)

Pseudo code:

def **inorder**(node):

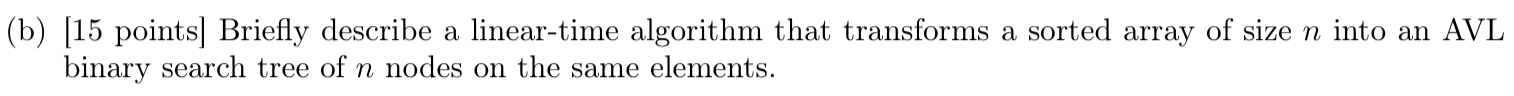
if node == null then return

if node.left is not null then inorder(node.left)

visit(node) # put the key of the node to the array

if node.right is not null then inorder(node.right)

Note: In my program, I created a python generator to output each node in the AVL tree and store them into array in order.



1. we can do it recursively:

* obtain the middle element in the array, and make it root
* recursively do the same thing for the left half and right half array.
  + obtain the middle element in the left half array, and make it left child of the root
  + obtain the middle element in the right half array, and make it right child of the root

since it takes constant time O(1) to obtain the middle element within the array, so the running time to construct an AVL tree from sorted array is O(n). (see pseudo code below)

Pseudo code:

def **fill\_AVL**(A, Astart, Aend):

if Aend < Astart:

return null

else:

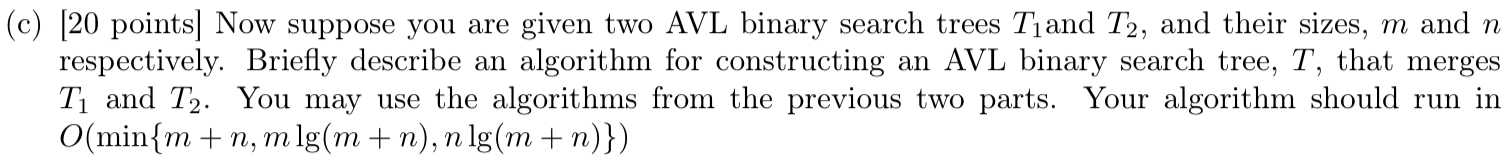
mid = (Aend + Astart) / 2

root.p = array[mid]

root.left = **fill\_AVL**(A, Astart, mid-1)

root.right = **fill\_AVL**(A, mid+1, Aend)

return root

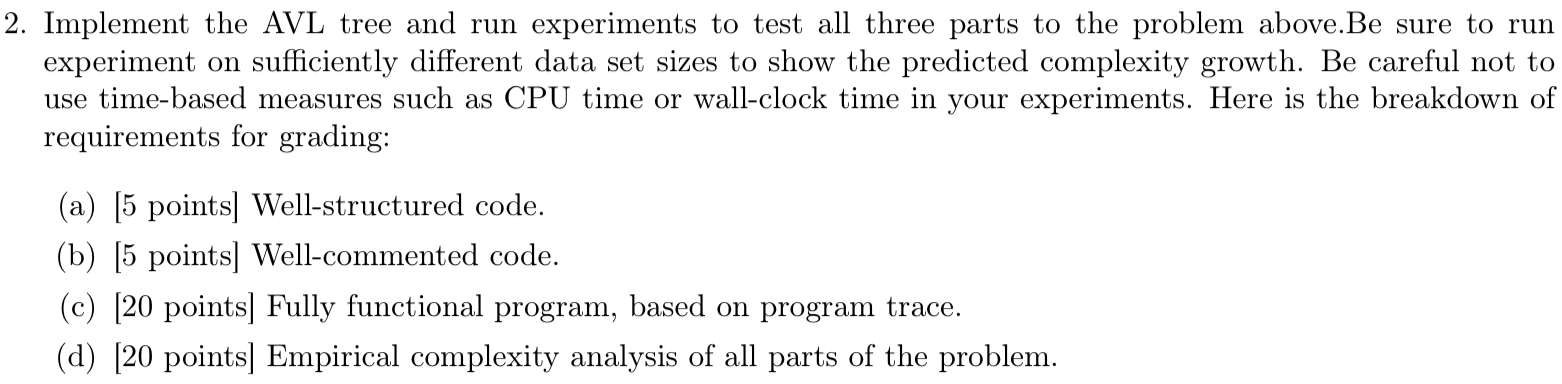


We can take all elements from the first(smaller) AVL tree one by one, then insert them into the second(larger) AVL tree. Let’s assume n>m, then insert an element into the larger AVL tree will take O(lgn) time, if we have m elements from smaller tree, then the time complexity is lgn+lg(n+1)+lg(n+2)+…+lg(n+m-1), then the time complexity will be O(m lg(m+n)) when n>m; the time complexity will be O(n lg(m+n)) when n<m. For the detail of the code, please refer to my program.

Pseudo code:

For element in smaller\_tree.inorder\_traversal():

Larger\_tree.insert(element)



See attached code and track log.

Trace log for problem (a) is AVL2Array.log

Trace log for problem (b) is Array2AVL.log

Trace log for problem (c) is MergeAVL.log