Datastructuren

October 12, 2009

1 Assignments

1. In this problem we compute the Internal Path Length (IPL) and average path length of a Drozdek-complete binary tree (a binary tree in which each node has two children, except the leaves and the leaves are all in the same level).

   a Consider the sum $\sum_{i=1}^{s} i \cdot x^i$. Find a closed form expression for this sum. Hint: There are many ways to find the expression for the sum. One way to do this is to proceed as follows. For $\sum_{i=1}^{s} i \cdot x^i$ we will write $x \cdot \sum_{i=1}^{s} i \cdot x^{i-1}$. In this form we see that each of the terms in the sum is a derivative (i.e., $i \cdot x^{i-1} = (x^i)'$). This together with the knowledge that $\sum_{i=1}^{N} y^i = \frac{y - y^{N+1}}{1-y}$ brings the closed form in sight.

   b Next consider a Drozdek-complete binary tree of height h. For each level we know how many nodes there are in that level and for each such a node the path length is $q - 1$ where q is the level under investigation. When taking the sum of all levels (from 1 to h) we obtain the IPL. The task is thus: come up with a formula for the IPL by carrying out the above suggested plan.

   c Compute the number of nodes in a Drozdek-complete binary tree of height h.

   d Compute the average path length for a Drozdek-complete binary tree of height h.

2. In this exercise we consider a structural transformation of a binary search tree (BST) which we call rotation. Let A be a BST and let x be a node of this tree. Let y be the left child of x. Denote by B, C the left and right subtree of the node y, and by D the right subtree of x. The tree $\tilde{A} = \text{rightRotation}(A, x)$ – a right rotation of A around x – is obtained by substituting in A the subtree with root x by a subtree with root y, the left subtree of which is B, and the right subtree has root x which has as left subtree C and as right subtree D.

   a Show that rightRotation(A,x) is a BST.

   b The inverse transformation of rightRotation(A,x) is a left rotation. Assuming that a node of the tree carries the fields $l, r, parent$ (left subtree, right subtree, parent respectively), write algorithms which correspond to a left rotation and a right rotation. The algorithms have as inputs A – a binary tree, and a a node in the tree.

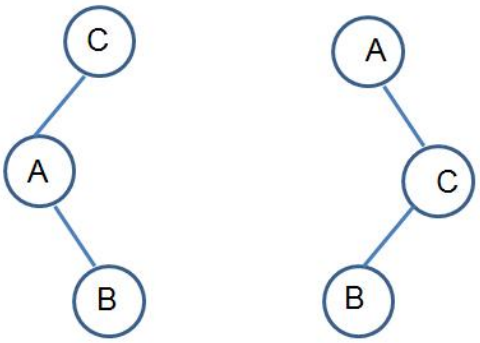


Figure 1: Two very simple Binary Search Trees

c Consider the two tiny binary search trees in Fig. 1. Show that you cannot balance either one by using any single rotation.

d Show that the height of the subtree which has been modified by rightRotation varies in the following way: it diminishes by 1 if and only if $h(B) > \max(h(D), h(C))$. It increases by 1 if and only if $h(D) > \max(h(B), h(C))$. In other cases the height does not change.

e Let v be a vertex (a node). Study the variation of height of $A(v)$ (the subtree rooted at v) depending on whether the height of $A_l(v)$ or $A_r(v)$ increases or diminishes by one unit. ($A_l(v)$ or $A_r(v)$ denote the left subtree respectively the right subtree of the vertex v.)

3. Consider the Binary Search Tree in Fig. 2. Show how the DSW algorithm turns this tree into a perfectly balanced BST.

4. Let A be BST, and x a node of A. The *balance factor* of x is defined as follows:

$$\text{balanceFactor}(x) = h(A_r(x)) - h(A_l(x))$$

. A BST tree A is called an *AVL tree*, if for each node x of A, we have:

$$\mid \text{balanceFactor}(x) \mid \leq 1$$

.

a Compute for the four BSTs in Fig. 3 the balance factors and determine which of them are AVL trees.

b Construct an AVL tree for which the leaves occur on more than 2 levels.

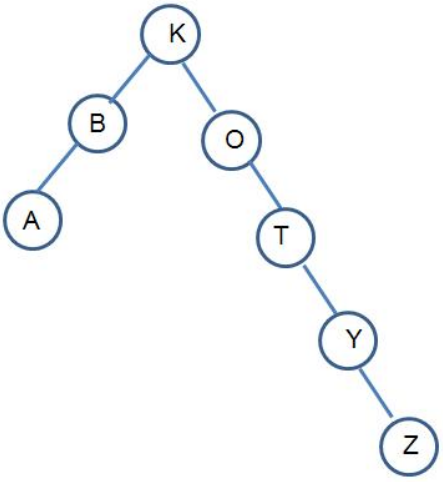


Figure 2: an unbalanced Binary Search Tree

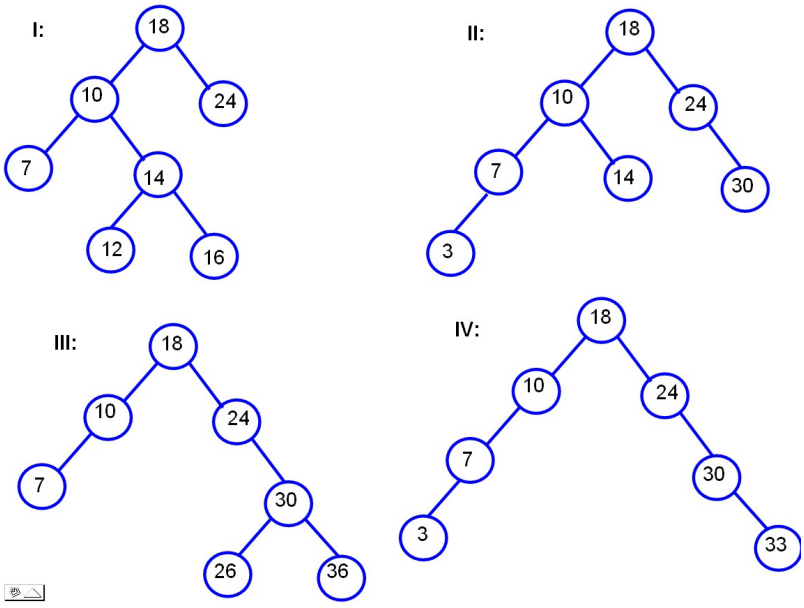


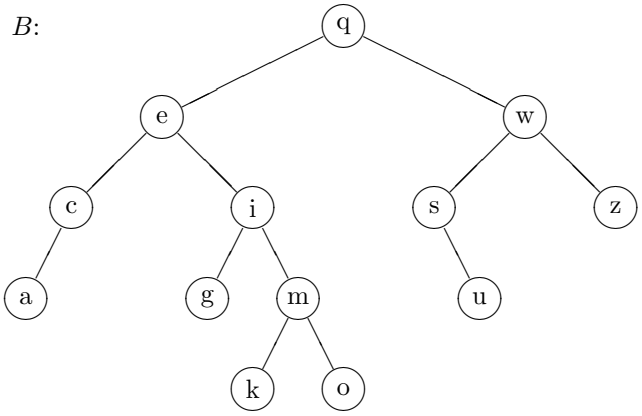
Figure 3: Four BSTs

c Design an algorithm which will update the balance factors, right after an insertion into an AVL tree without having done the balancing rotation(s) yet, in each of the nodes on the path from the inserted node to the first unbalanced node – or root whichever comes first. (Make explicit which information fields the nodes carry.)

d Wat is het minimale en wat het maximale aantal knopen in een AVL-boom met hoogte $h \geq 0$ ($h = 0$ correspondeert met een lege boom, $h = 1$ met een enkele knoop)?

e Beschouw de onderstaande AVL-boom B:

B:



Laat zien welke AVL-boom uit B ontstaat als:

i. aan B een knoop p wordt toegevoegd,

ii. aan B (de originele boom dus) een knoop l wordt toegevoegd,

iii. uit B (de originele boom dus) de knoop z wordt verwijderd,

iv. aan B (de originele boom dus) eerst de knopen d en b worden toegevoegd (in die volgorde), waarna knoop z wordt verwijderd.

Het moet steeds een AVL-boom blijven! (gebruik zonodig rotaties)

f Welke AVL-boom ontstaat wanneer de volgende getallen in de aangegeven volgorde in de boom worden opgeslagen? Het moet steeds een binaire AVL-boom blijven:

79, 51, 32, 59, 65, 71, 37, 94, 86, 15, 23, 44.

g Bereken de interne padlengte voor deze boom. Wat is het verwachte aantal sleutelvergelijkingen nodig voor een succesvolle zoekpoging, ervan uitgaande dat elk van de aanwezige getallen een even grote kans heeft om gezocht te worden?

h Verwijder het getal 65 uit de boom en geef de resulterende AVL-boom.

5. Consider AVL trees. In the following we address the question of how much labor needs to be don in order to restore the AVL property after insertions.

a Give examples of AVL trees where the insertion of a key does not destroy the AVL-property.

b Catalogue systematically the cases of insertion into an AVL tree after which the AVL property needs to be restored.

c Suppose that after the insertion into an AVL tree, the tree is not balanced any more. Show that the repair operation (rotation) has to occur in exactly one place of the tree. The place where the repair rotation needs to take place is the first node on the path from the inserted leaf towards the root where the balance factor has become either -2 or +2.