

Computer Science Stack Exchange is a question and answer site for students, researchers and practitioners of computer science. Join them; it only takes a minute:

[Sign up](#)

Here's how it works:



Anybody can ask a question



Anybody can answer



The best answers are voted up and rise to the top

Rearranging linear tree with right rotates [closed]

I ran into a fun interview question, yesterday. Can anyone help me?

Suppose a binary tree with six nodes is given, such that each node has only a left child. With how many "right rotate" operations (without any left rotates), we can convert this tree to a tree in which each node has only a right child?

[algorithms](#)

[data-structures](#)

[trees](#)

[binary-trees](#)

edited Mar 16 '15 at 12:49



[David Richerby](#)

50.5k 9 74 146

asked Mar 16 '15 at 12:37



[Homan](#)

11 3

closed as unclear what you're asking by [D.W.](#), [Juho](#), [Luke Mathieson](#), [Nicholas Mancuso](#), [J.-E. Pin](#) Mar 21 '15 at 23:44

Please clarify your specific problem or add additional details to highlight exactly what you need. As it's currently written, it's hard to tell exactly what you're asking. See the [How to Ask](#) page for help clarifying this question.

If this question can be reworded to fit the rules in the [help center](#), please [edit the question](#).

2 What did you try? Where did you get stuck? – [David Richerby](#) Mar 16 '15 at 12:49

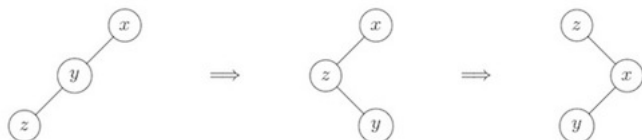
2 Answers

Just so we're on the same page, here's what a right rotation about node x will do:



Where T , U , V are the subtrees of nodes x and y . It's easy to see that if you start with a left-child chain consisting of nodes x and y as above (with T , U , V empty), a right rotation about x will yield the right-child chain in the right figure.


Now do the same thing, starting with a left-child chain of three nodes, x , y , z , reading from the top. If you do a right rotation about y and then another about x , you'll have the following transformations:



and now if you recursively transform the x , y left-child branch (since we know how to do that), you'll wind up with a right-child tree, z , y , x , reading from the top. It shouldn't be hard to see that this process will work for left-child chains of arbitrary length n , and will require $n(n - 1)$ rotations, so no more than that number of right rotations will change a left-child tree to a right-child tree.

It's worth noting that, although the question didn't require it, this process will maintain the binary search tree property, giving the nodes in the reverse order of the original.

edited Mar 17 '15 at 1:18



answered Mar 16 '15 at 18:1

Rick Decker

12k 3 26 43

I don't get your second-to-last paragraph. With one more rotation around y , we have moved z to the root, and the rest of the chain *remains a left child of z* , just as it was in the beginning. A linear number of rotations is sufficient by that scheme. – Raphael ♦ Mar 17 '15 at 6:56

Also, in the last paragraph you say, "giving the nodes in the reverse order of the original". Why? These rotations maintain the in-order sequence. (Which is why we use them in AVL trees, after all.) Or do you mean the "read from top to bottom" traversal? – Raphael ♦ Mar 17 '15 at 6:58

i think @Raphael, is correct. – Homan Mar 17 '15 at 12:33

why you didnt reflect any comment ? – Homan Mar 19 '15 at 8:22

If the chain has n nodes, $n - 1$ right rotations around the root will do the trick.



answered Mar 17 '15 at 6:57

Raphael ♦

53k 20 128 284