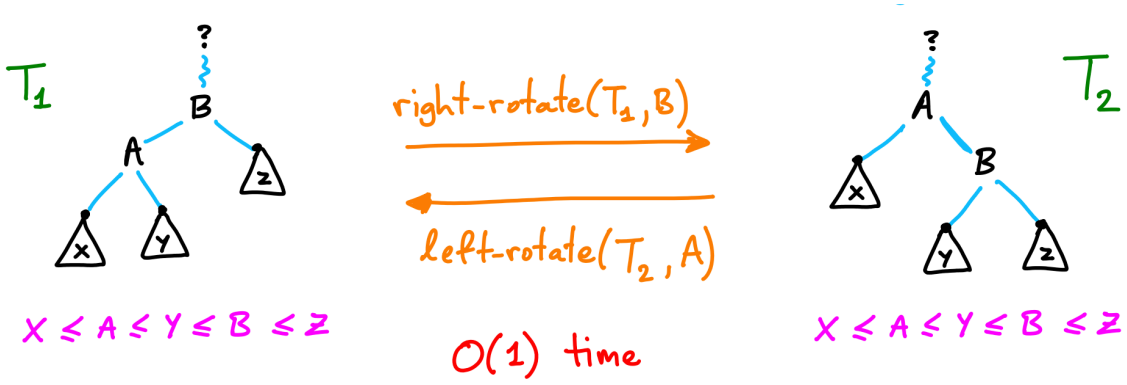
ALGORITHMS, FALL 2019, HOMEWORK 6

– This assignment is worth 1 unit.
– Due on Saturday, November 2, at noon.
– Please do not include the problem statement in your submission. Just include your solution.

1. The *rotation* operation is defined via the figure below. It is discussed in the unit on Red-Black trees, but nothing about Red-Black trees needs to be understood in order to define rotations. It is an operation on BSTs in general. Rotations transform BSTs locally by reassigning a few parent-child pointers. For this assignment it suffices to understand the changes depicted in the figure. A and B are nodes, X,Y,Z are subtrees, and the "?" signifies that B (and then A) might have a parent.



Let P and Q be two given binary search trees storing the same n values.
Note: it is enough to think of just the shapes of P and Q. The values won't matter.

(a) Invent an algorithm that can transform any given P into any given Q, only using rotations. Your algorithm should use $O(n)$ rotations in the worst case. It does not need to have the best possible hidden constant. You should justify correctness, in terms of the result and number of rotations used.

(b) Show that any algorithm that uses rotations for such a transformation must use $\Omega(n)$ rotations in the worst case. To show this, give an example of two trees that require a linear number of rotations, no matter what rotation-based algorithm is used. Explain why the bound holds.