1 apply to right weighted tree 1 2 3

On first go apply append recurred left tree branch and then cons entry 1 and recur on right branch

Left branch is null so it returns null to append, right branch is not null so it calls again

On 2nd go apply appends recurred left tree branch and then cons entry 2 and recurs on the right branch

Left branch is null so it returns null to append, right branch is not null so it calls again

On 2nd go apply appends recurred left tree branch and then cons entry 3 and recurs on the right branch

Left branch is null so it returns null to append, right branch is null so it returns append

Append should flatten all this out to 1 2 3

O(2n) or O(n) in this case

Balanced tree 1 2 3

On first go apply append recurred left tree branch and then cons entry 2 and recur on right branch

Left branch is not null so append on recurred left branch and then cons entry 1 and recur on the right branch

Left branch is null so return null to append

Right branch is null so return null to append

Append null onto 1 and null

Return 1 to top level

Right branch is not null so append on recurred left branch and then cons entry 3 and recur on the right branch

Left branch so null is returned to append

Right branch is null so null is returned to append

Append null onto 3 and null

Return 3 to top level

Append and flatten all to 1 2 3

This appears to be O(n^2)

Result-list starts null

Not null so first time calls copy to list on left branch and cons entry on to copy to list of right branch with a null result-list as the result-list

Second time is also not null so calls copy to list on left branch and cons entry on copy to list of right branch with “cons entry prime on to copy to list of right branch prime with a null result” as result-list