5.1

[i, h, j, l, k, m]

5.2

Step 1: $$Root$$ = the last node in Postorder

Step 2: Find $$Root\_{index}$$ = index of $$Root$$ in Inorder

Step 3: recurse on ($$Inorder[0:Root\_{index} - 1]$$, $$Postorder[0:Root\_{index} - 1]$$) as the left subtree of $$Root$$; recurse on ($$Inorder[Root\_{index}+1:], Postorder[Root\_{index}

+1:]$$) as the right subtree of $$Root$$;

Runtime Analyze:

Upper bound: for every node, step 1 takes O(1), step 2 takes O(n); so the total algorithm takes O(n^2)

Lower bound (an example):

Consider a tree with n nodes in the way that every non-leaf node has a right child and no left child. For the k-th node, step 2 takes n+1-k searches, so in total it takes $$n + (n-1) + (n-2)+ ... + 1 = (n^2 - n)/2$$ operations.

5.3

An improvement from the algorithm shown above:

Step 2: Use a hash table where key = node value, value = index in $$Inorder$$

Step 3: Instead of slicing, use start and end index in each recursion

In each recursion:

Step 1 and 3 natually takes O(1); Step 2 takes O(1) to search in hash map, so add a node to tree takes O(1), building the total tree takes O(n). After building the tree, an in order traversal takes O(n), the total algorithm takes O(n)

5.4

import sys

*# DO NOT REMOVE THIS LINE*

sys.setrecursionlimit(10000)

*def* \_constructor(*inorder*,*postorder*,*preorder*,*inorder\_map*,*instart*,*inend*,*poststart*,*postend*):

*global* preorder\_pos

    if inend - instart < 0:

        return

    root = postorder[postend]

    preorder[preorder\_pos] = root

    preorder\_pos+=1

    root\_index = inorder\_map[root]

    \_constructor(inorder,postorder,preorder,inorder\_map,instart,root\_index - 1,poststart,poststart + (root\_index - 1 - instart))

    \_constructor(inorder,postorder,preorder,inorder\_map,root\_index + 1, inend, postend - inend + root\_index, postend - 1)

    return preorder

*def* construct\_preorder\_traversal(*inorder*, *postorder*):

    """

    Args:

        inorder (list): the inorder traversal of the tree

        postorder (list): the postorder traversal of the tree

    Output:

        list: the preorder traversal of the tree.

    """

*global* preorder\_pos

    preorder\_pos = 0

    inorder\_map = {}

    for i in range(len(inorder)):

        inorder\_map[inorder[i]] = i

    return \_constructor(inorder,postorder,['a']\*len(inorder),inorder\_map,0,len(inorder)-1,0,len(inorder)-1)