Daily Coding Problem #125

Problem

This problem was asked by Google.

Given the root of a binary search tree, and a target `K`, return two nodes in the tree whose sum equals `K`.

For example, given the following tree and `K` of 20

```
    10
   /   \
 5      15
       /  \
     11    15
```

Return the nodes `5` and `15`.

Solution

This question is similar to the two-sum problem with a list. We can actually reduce this problem into that one by turning the tree into a list. To save some space, we'll use generators, which are like list-like but are generated on-the-fly.

```
def two_sum(root, K):
    seen = {} # Map of val to node

    for node in iter_tree(root):
        if K - node.val in seen:
            return (node, seen[K - node.val])
        seen[node.val] = node

    return None

def iter_tree(root):
    if root:
        for node in iter_tree(root.left):
            yield node

        yield root

        for node in iter_tree(root.right):
            yield node
```

Another solution is to simply to iterate over each node and do a binary tree search for `K - node.val`. This takes O(N log N) time since for each node, we do a search which takes log N. However, it will only take O(log N) space because the call stack gets log N deep.

```
def two_sum(root, K):
    for node_one in iter_tree(root):
        node_two = search(root, K - node_one.val)

        if node_two:
            return (node_one, node_two)

    return None
```

```
def search(node, val):
    if not node:
        return None

    if node.val == val:
        return node
    elif node.val < val:
        return search(node.right, val)
    else:
        return search(node.left, val)
```