StefanPochmann ★ 46834 Last Edit: October 8, 2018 8:14 AM 5.2K VIEWS

16

For the follow-up question, this is O(k log n) and maybe O(k + log n), haven't thought it through yet. Probably by far the longest solution I've posted here, I even felt the need to include comments :-P

Imagine you didn't have a BST but a simple sorted array. How would you find the k values closest to the target value? You could do binary search to find the closest value and then move outwards with two index variables, collecting the k closest values. That's what I do here, except instead of simple array indexes I have two tree iterators in the form of root-to-node paths.

Finding the initial iterator=path to the closest value takes O(h) time, where h is the height of the tree. Increasing or decreasing these iterators=paths also takes O(h) time. So O(kh) overall. And maybe better, as moving an iterator=path will often be quick. I mean, it's just an inorder traversal, so I'd expect O(1) amortized moving time at least if we're traversing the entire tree.

```
def closestKValues(self, root, target, k):

    # Helper, takes a path and makes it the path to the next node
    def nextpath(path, kid1, kid2):
        if path:
            if kid2(path):
                path += kid2(path),
                while kid1(path):
                    path += kid1(path),
            else:
                kid = path.pop()
                while path and kid is kid2(path):
                    kid = path.pop()

    # These customize nextpath as forward or backward iterator
    kidleft = lambda path: path[-1].left
    kidright = lambda path: path[-1].right

    # Build path to closest node
    path = []
    while root:
        path += root,
        root = root.left if target < root.val else root.right
    dist = lambda node: abs(node.val - target)
    path = path[:path.index(min(path, key=dist))+1]

    # Get the path to the next larger node
    path2 = path[:]
    nextpath(path2, kidleft, kidright)

    # Collect the closest k values by moving the two paths outwards
    vals = []
    for _ in range(k):
        if not path2 or path and dist(path[-1]) < dist(path2[-1]):
            vals += path[-1].val,
            nextpath(path, kidright, kidleft)
        else:
            vals += path2[-1].val,
            nextpath(path2, kidleft, kidright)
    return vals
```

python fast

Comments: 8 Best | Most Votes | Newest to Oldest | Oldest to Newest

Type comment here... (Markdown is supported)

Post

leetcodeaccount ★ 2 July 27, 2017 12:12 PM

Same idea. Python3 (3.3 or above) version:

```
def closestKValues(self, root, target, k):
    def less_equal(root):
        if root:
            if root.val <= target:
```