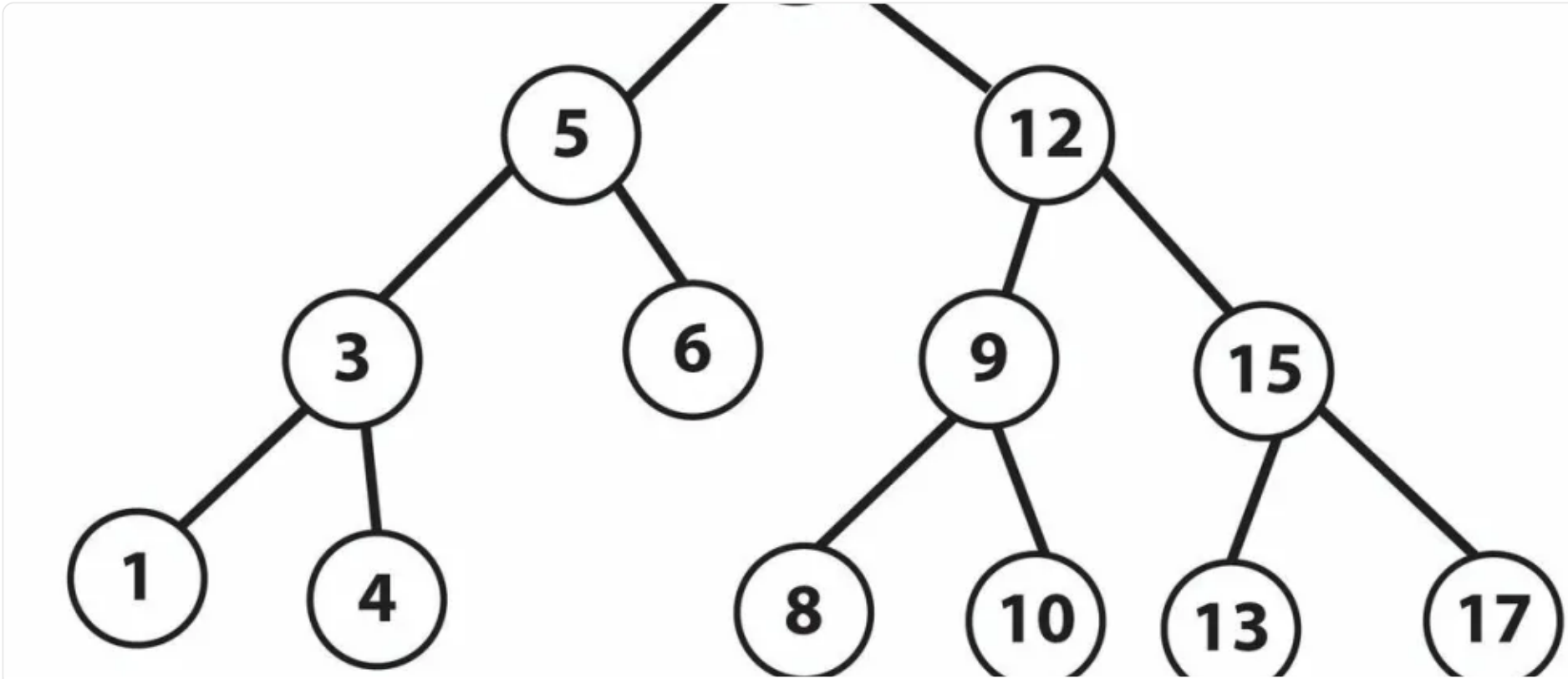

Youdiowei Eteimorde
Posted on ٢٧ نوفمبر ٢٠٢١

Find closest value in Binary Search Tree

#algorithms #tutorial #beginners #programming

Binary Search Tree is a sorted **binary tree** meaning given a root node all the values to the left are less than

it and all the values to the right are greater it. This property enables Binary Search to be performed on the tree.

Let exploit this capability, Given a **target value** let's find a number that is the closest to it in the tree. First we create our BST node which has a value and two pointers. One to the right and one to the left.

```
# A BST node
class BST:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None
```

Now we create our tree by attaching our nodes together

```
#             7                   <-- Root Node
#       5           12            <-- Depth One
#    3     6     9        15      <-- Depth Two
#  1   4       8   10  13    17   <-- Depth Three


# Create the tree
tree = BST(7)
# depth one
tree.right = BST(5)
tree.left = BST(12)
# Right node --> depth two
tree.right.right = BST(15)
tree.right.left = BST(9)
```

```
# Right node --> depth three (right node)
tree.right.right.right = BST(17)

tree.right.right.left = BST(13)

# Right node --> depth three (left node)
tree.right.left.right = BST(10)
tree.right.right.left = BST(8)

# Left node --> depth two
tree.left.right = BST(6)
tree.left.left = BST(3)

# Left node --> depth three (left node)
tree.left.left.right = BST(4)
tree.right.right.left = BST(1)
```

The Psuedo Code below is the solution to the problem.

Pseudo Code:

1. Assign the closest value to the root node.
2. Check if |target - closest| > |target - tree's value|.
    - If yes assign the tree's value as the closest value
    - If no the continue using the closest value

3. Check if the target's value > current tree's value
    - If yes Go to the right sub tree.
    - If no move on.

4. Check if the target's value < current tree's value

- If yes Go to the left sub tree.

- If no move on.

5. Check if there is are no sub trees left
   - If yes return current closest value
   - If no go back to step 2

The Pseudo Code above can be implemented recursively or iteratively

```
# Recursive solution
closest = tree.value # Initial closest value is the root's value
def findClosestValueInBstRecursive(tree, target, closest):
    if tree is None:
        return closest
    if abs(target - closest) > abs(target - tree.value):
        closest = tree.value
    if target < tree.value:
        return findClosestValueInBstRecursive(tree.left, target, closest)
    elif target > tree.value:
        return findClosestValueInBstRecursive(tree.right, target, closest)
    else:
        return closest

# Iterative solution
def findClosestValueInBstIterative(tree, target, closest):
    currentNode = tree
    while currentNode is not None:
        if abs(target - closest) > abs(target - currentNode.value):
            closest = currentNode.value
        if target < currentNode.value:
```

```
            currentNode = currentNode.left
        elif target > currentNode.value:
            currentNode = currentNode.right
        else:
            break
    return closest
```

Now time to test both solutions given the value of **14** find its closest value within the tree defined earlier.

```
tar = 14
findClosestValueInBstRecursive(tree, tar, closest)
# Output: 15

tar = 14
findClosestValueInBstIterative(tree, tar, closest)
#Output: 15
```

Both solutions arrived at the same answer **15** which just happens to be the closest value in the tree.

Discussion (0)

Code of Conduct • Report abuse

Youdiowei Eteimorde

Just another dev..

LOCATION
Port Harcourt, Nigeria

JOINED
٢٥ يوليو ٢٠٢١

More from Youdiowei Eteimorde

The two Sum problem

#algorithms #datastructure #programming #computerscience