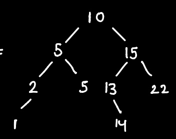
Find Closest Value in BST

Input:

tree =



target = 12

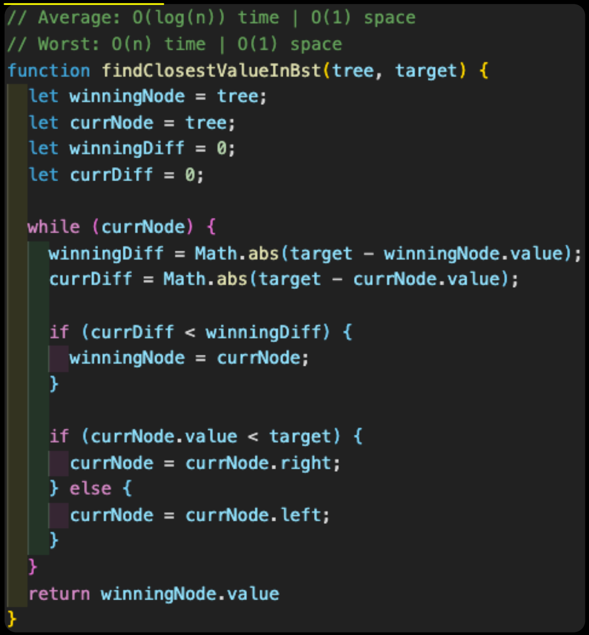
Output: 13

Input: Binary Search Tree
Target value

Output: return closest value to the target value contained in the BST

Assume: There will only be one closest value

Iterative Solution:



```
// Average: O(log(n)) time | O(1) space
// Worst: O(n) time | O(1) space
function findClosestValueInBst(tree, target) {
  let winningNode = tree;
  let currNode = tree;
  let winningDiff = 0;
  let currDiff = 0;

  while (currNode) {
    winningDiff = Math.abs(target - winningNode.value);
    currDiff = Math.abs(target - currNode.value);

    if (currDiff < winningDiff) {
      winningNode = currNode;
    }

    if (currNode.value < target) {
      currNode = currNode.right;
    } else {
      currNode = currNode.left;
    }
  }
  return winningNode.value
}
```

Declare a winning Node and current Node value.

currNode ⟶ node that we are currently at

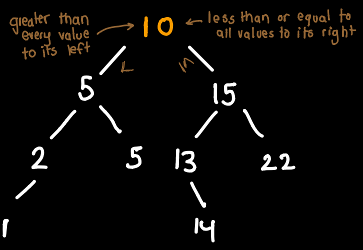
winningNode ⟶ node that is closest to our target

We set the curr / winning node to be the root of the tree

We then calculate the difference between the target and the winning Node / currNode. If the currNode difference is smaller than the winning Node difference, the currNode becomes the winning Node

We then traverse the tree and do this at each node

Once we reach a null node, we exit the while loop and return the winningNode


greater than every value to its left
less than or equal to all values to its right

O(logn) time on avg since we get rid of half the BST at each iteration

O(n) time at worst since the tree could be one branch only

O(1) space since no more space gets used as input grows