Assignment Solutions | Binary Search Tree 2 | Week 18

1. Given a BST, count subtrees in it whose nodes lie within a given range.

Solution:

```
class Solution {
public:
    int ans = 0;
    bool helper(TreeNode* root, int low, int high) {
        if (root == NULL) return true;
        bool l = helper(root->left, low, high);
        bool r = helper(root->right, low, high);
        if (l && r && low <= root->val && root->val <= high) {
            ans++; return true;
        }
        return false;
    }
    int getCount(TreeNode* root, int low, int high) {
        int count = 0;
        helper(root, low, high);
        return count;
    }
};
```

2. Given a BST and two keys in it. Find the distance between two nodes with given two keys. It may be assumed that both keys exist in BST.

Solution:

created with craft

```
class Solution {
public:
  int distanceFromRoot(Treenode* root, int x) {
    if (root->val == x) return 0;
    else if (root->val > x) return 1 + distanceFromRoot(root->left, x);
    else return 1 + distanceFromRoot(root->right, x);
  }
  int distance(Treenode* root, int a, int b) {
    if (!root) return 0;
    if (root->val > a && root->val > b) return distanceBetween2(root->left, a, b);
    if (root->val < a && root->val < b) return distanceBetween2(root->right, a, b);

    // found the LCA
    if (root->val >= a && root->val <= b) return distanceFromRoot(root, a) +
distanceFromRoot(root, b);
  }
};
```

©

created with craft