LeetCode Unique Binary Search Trees II: Bottom Up Approach

Overview

LeetCode Unique Binary Search Trees II: Enumerate all the unique trees with a recursive bottom up tree construction by trying each number as the root.

LeetCode Unique Binary Search Trees II

Given *n*, generate all structurally unique **BST's** (binary search trees) that store values 1…*n*.

For example,
Given *n* = 3, your program should return all 5 unique BST’s shown below.

```
1         3     3      2       1
   \      /    /      / \      \
    3     2     1      1   3      2
   /    /       \                 \
  2     1         2                 3
```

Solution: Bottom Up Approach

Use a bottom up approach to build all the possible trees, again, the basic idea is the same, for each of the 1 to n to the root, we recursively build all the possible left and right subtrees, we THEN make one node of this root for all the possible combinations of the left and right subtree, and return these set of all the possible trees build on 1 to n. We keep recursively doing this until all the possible trees are enumerated. The following compact code implements this idea and passed the Unique Binary Search Trees II from the LeetCode OJ:

```
class Solution {
    public:
        vector<TreeNode *> ret;
```

```
vector<TreeNode *> dfs(int s, int e) {
    if (s > e) {
        vector<TreeNode *> ret(1, NULL);
        return ret;
    }

    vector<TreeNode *> ret;
    for (int r = s; r <= e; ++r) {
        vector<TreeNode *> left  = dfs(s, r-1);
        vector<TreeNode *> right = dfs(r+1, e);

        for (int i = 0, len = left.size(); i < len; ++i) {
            for (int j = 0, len2 = right.size();  j < len2; ++j) {
                ret.push_back(new TreeNode(r));
                ret.back()->left = left[i];
                ret.back()->right = right[j];
            }
        }
    }
    return ret;
}
```

```
    vector<TreeNode *> generateTrees(int n) {
        return dfs(1, n);
    }
};
```

Note if there is no such a tree, we need to push a NULL node into the results, i.e., the correct results for genertateTrees(0) would be a vector of a single NULL element rather than an empty vector.

Summary

LeetCode Unique Binary Search Trees II: Enumerate all the unique trees with a recursive bottom up tree construction by trying each number as the root.

Written on May 23, 2013

« leetcode: Restore IP Addresses

LeetCode Gray Code: Backtracking vs Iterative Way »