Yu's Coding Garden

My blog for LeetCode Questions and Answers...

Wednesday, May 22, 2013

leetcode Question115: Unique Binary Search Trees II

Unique Binary Search Trees II

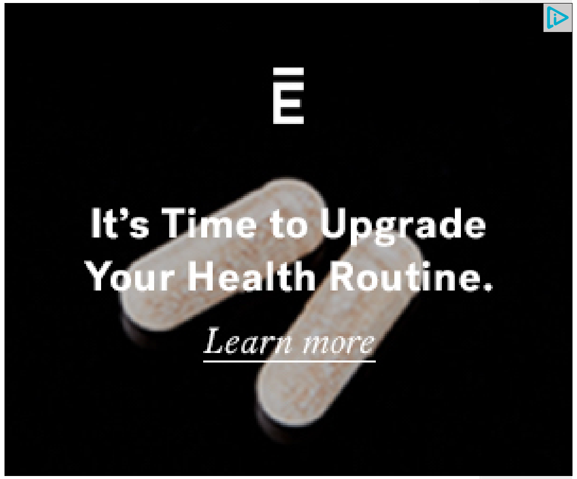
Given *n*, how many structurally unique **BST's** (binary search trees) that store values 1...*n*?

For example,
Given *n* = 3, there are a total of 5 unique BST's.

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

Analysis:

The basic idea is still using the DFS scheme. It is a little hard to think the structure of the argument list in the function. It is clear that for each tree/subtree, we will set the root as the start position to the end position, and recursively construct the left subtree with the left part and the right subtree with the right part.
So first we can have



```
void dfs (int st, int ed     ){
  if (st>ed) {   // generate a null node }
  else{
    for (int i=st;i<=ed;i++){

      dfs(st,i-1,    );      //generate left subtree
      dfs(i+1,ed,    );  // generate right subtree
    }
  }

}
```

Next step is to think about how to store all the possible solutions.
This is important ! Think about the root node, all the possible solutions of the tree are from the combinations of all the possible solutions of its left subtree, and its right subtree. One step further, if we have a root node and a left node, for the left node, still the subtrees below it are the combinations of the possible solutions of its left and right subtree, until the leaf node.

In other words, we store all the possible solutions for each node, instead of storing the only tree. So, we can have
void dfs(int st, int ed, vector<TreeNode *> res){}
in this function, recursively generate the left tree and right tree, then construct the current node, and push it to the current solution vector.
Details see the code.

Code:

```
/**
 * Definition for binary tree
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    void dfs(int st,int ed,vector<TreeNode *> &res){
        if (st>ed){
            res.push_back(NULL);
        }else{
            for (int i=st;i<=ed;i++){
                vector<TreeNode *> lefts;
                dfs(st,i-1,lefts);
                vector<TreeNode *> rights;
                dfs(i+1,ed,rights);

                for (int li = 0; li<lefts.size();li++) {
                    for (int ri =0; ri<rights.size();ri++){
                        TreeNode* node = new TreeNode(i);
```

Who am I?

I have just got my Ph.D. in Computer Science. Now I am a research engineer at a startup company.

My research focuses on deep learning and computer vision.

My homepage: (1)
My Google Scholar Page: here

New!!!
I am starting to add *Python* Code for the problems!!!

Useful link: C++ reference

Basics:

Common Sorting Algorithms

Hash Table Basics

Tree Traversal (Recursive & Non-Recursive)

Search the leetcode solutions here:

Search

```
                        node->left=lefts[li];
                        node->right=rights[ri];
                        res.push_back(node);
                    }
                }
            }
        }
    }

    vector<TreeNode *> generateTrees(int n) {
        // Start typing your C/C++ solution below
        // DO NOT write int main() function
        vector<TreeNode*> res;
        dfs(1,n,res);
        return res;
    }
};
```

Posted by Yu Zhu at 1:01 PM +3 Recommend this on Google

Labels: leetcode

2 comments:

Peiyuan Zhou November 3, 2014 at 6:38 PM

Great idea!

Reply

yang jin June 21, 2015 at 7:59 AM

impressive!

Reply

Enter your comment...

Comment as: Select profile...

I started to post **C++ Solutions for the famous book:**
Cracking The Coding Interview (5th Edition)
Please Look at my gitHub:
here
(Regularly update)

Note:
All the code provided for the ***leetcode questions*** *have been tested*
and have ***passed both small and large*** *tests of the online judge*
system from leetcode.

Pages

Home

Labels

answer (1) array (19) backtracking (1) BFS (13) binary search (10) binary tree (5) bit manipulations (8) BST (2) bubble sort (1) bucket sort (1) c++ (1) collision (1) common prefix (1) compare (1) design (1) DFS (16) dict (2) divide and conquer (2) DP (10) dynamic programming (1) graph (2)

[Newer Post](#) [Home](#) [Older Post](#)

Subscribe to: Post Comments (Atom)

hash (5) hash function (1) hash table (6) heap (2) heap sort (3) Index (2) inorder (2) insert sort. (2) iteration (3) iterative (2) iteratively (2) iterator (1) KMP (1) leetcode (202) linked list (8) list (2) loop (7) map (5) mask (1) math (6) merge sort (2) minimum substring (1) non-recursive (1) O(n) (1) object oriented design (1) online judge (1) oop (1) permutations (1) pointers (9) postorder (1) preorder (2) prime number (1) python (1) questions (1) queue (2) quick sort (1) radix sort (1) Re-view (3) recursive (7) selection sort (1) solutions (1) sort (2) sorting algorithm (2) stack (5) starter (4) string (10) stringstream (2) topology (1) trapping rain water (1) traverse (1) Tree (29) Tree Traversal (8) Trie (1) two pointers (1) twosum (1) XOR (1)

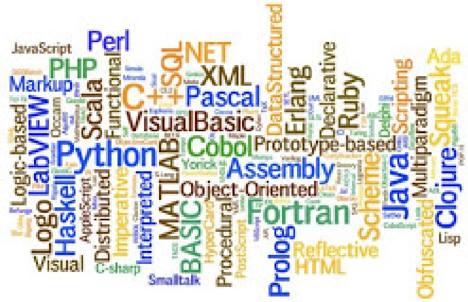
Blog Archive

- ► 2017 (1)
- ► 2016 (25)
- ► 2015 (44)
- ► 2014 (15)
- ▼ 2013 (126)
  - ► Dec 2013 (14)
  - ► Oct 2013 (3)
  - ► Sep 2013 (2)
  - ► Aug 2013 (6)
  - ► Jul 2013 (1)

- leetcode Question 86: Reverse Nodes in k-Group
- leetcode Question 108: Swap Nodes in Pairs
- leetcode Question 109: Symmetric Tree
- leetcode Question 110: Text Justification
- leetcode Question 111: Trapping Rain Water

- ► Apr 2013 (32)
- ► Mar 2013 (6)
- ► Feb 2013 (9)
- ► Jan 2013 (35)

► 2012 (29)



Loading data...
+
−