1. Unique Binary Search Trees II

Medium

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

**Example:**

Input: 3  
Output:  
[  
 [1,null,3,2],  
 [3,2,null,1],  
 [3,1,null,null,2],  
 [2,1,3],  
 [1,null,2,null,3]  
]  
Explanation:  
The above output corresponds to the 5 unique BST's shown below:  
  
 1 3 3 2 1  
 \ / / / \ \  
 3 2 1 1 3 2  
 / / \ \  
 2 1 2 3

**Solution**

递归建树。对于n个节点，从1 ~ n依次作为根节点，然后递归建立左子树和右子树

/\*\*  
 \* Definition for a binary tree node.  
 \* struct TreeNode {  
 \* int val;  
 \* TreeNode \*left;  
 \* TreeNode \*right;  
 \* TreeNode(int x) : val(x), left(NULL), right(NULL) {}  
 \* };  
 \*/  
typedef TreeNode\* pnode;  
class Solution {  
public:  
 vector<TreeNode\*> generateTrees(int n) {  
 vector<pnode>ans;  
 if(n > 0)ans = creat(1, n);  
 return ans;  
 }  
 vector<pnode> creat(int start, int end){  
 if(start > end)return vector<pnode>{NULL};  
 vector<pnode>ans;  
 for(int i = start; i <= end; ++i){  
 vector<pnode> left = creat(start, i - 1);  
 vector<pnode> right = creat(i + 1, end);  
 for(pnode l : left){  
 for(pnode r : right){  
 pnode cur = new TreeNode(i);  
 cur->left = l;  
 cur->right = r;  
 ans.push\_back(cur);  
 }  
 }  
 }  
 return ans;  
 }  
};