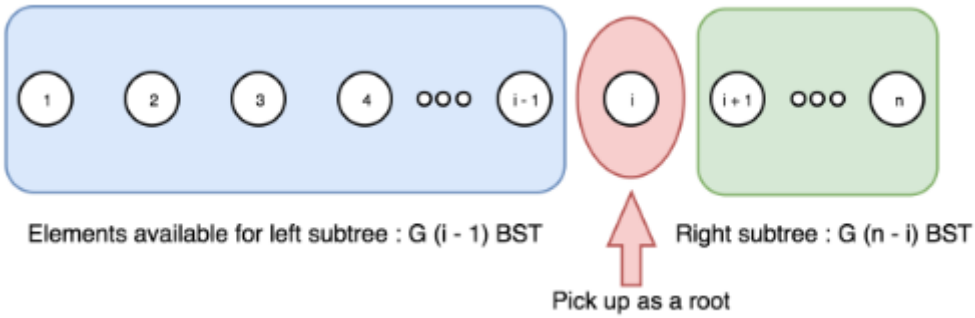
LeetCode95 不同的二叉搜索树

> *给定一个整数 n，生成所有由 1 ... n 为节点所组成的**二叉搜索树**。*

- 主要思路是**递归构造**，选取i 为 `根节点`，左节点为G[i-1]，右节点为G[n-i]
- G为卡特兰数

$$G(i) = \sum_{n=1}^{N} G(i-1) \times G(n-1)$$

```
// 参考代码
vector<int> dp(n + 1);
dp[0] = 1; dp[1] = 1;
for (int i = 2; i <= n; i++) {
        for (int j = 1; j <= i; j++) {
                dp[i] += dp[j - 1] * dp[i - j];
        }
}
```



批注 2020-01-24 133525

由于这道题需要我们将`二叉搜索树`构造出来，不仅仅是计算出来数量，因此我们需要构造的角度思考

1. 我们的函数结构应该是如下的，`start`，`end` 用来表示起始位置，如果`start > end`，则需要返回空树，即此时我们无法继续构造，这是函数的**出口**

```
vector<TreeNode*> makeTrees(int start, int end) {
        vector<TreeNode*> alltrees;
        if (start > end) {
                alltrees.push_back(NULL);
                return alltrees;
```

```
        }
        else{
            // do something
            return alltrees;
        }
}
```

2. 在第二个分支`do something`，我们需要完成三件事情
    - 建立根节点
    - 递归构造左子树
    - 递归构造右子树

a. 建立根节点，注意我们需要对`[start,end]`中每一个节点都作为根节点建立一次

```
for (int i = start; i <= end; i++) {

}
```

b. 构造左子树，右子树

```
vector<TreeNode*>left = makeTrees(start, i - 1);
vector<TreeNode*>right = makeTrees(i + 1, end);
```

c. 由于`left``right`返回的是列表，这里`left`中的每一个元素都可以作为左子树，因此我们还需要两重循环，将左右子树拼接，根节点选择`i`

```
for (auto LeftIter = left.begin(); LeftIter != left.end(); LeftIter++) {
        for (auto RightIter = right.begin(); RightIter != right.end(); RightIter++) {
                TreeNode* a = new TreeNode(i);
                a->left = *LeftIter;
                a->right = *RightIter;
                alltrees.push_back(a);
        }
}
```

3. 最后调用时我们只需要`return makeTrees(1, n);`，但是如果`n==0`，我们返回空列表