Recitation - 07

CS2040S Recitation Team

March 2022

Problem[1]

Given an integer n, return the number of structurally unique BST's (binary search trees) which has exactly n nodes of unique values from 1 to n. Try to write a program and comment on the running time. Figure 1 shows an example with $n = 3$.

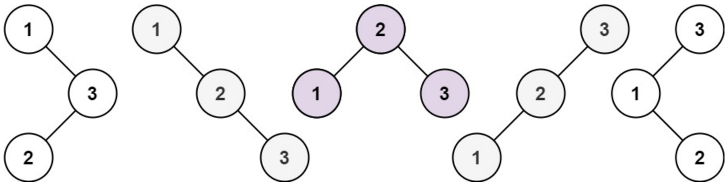
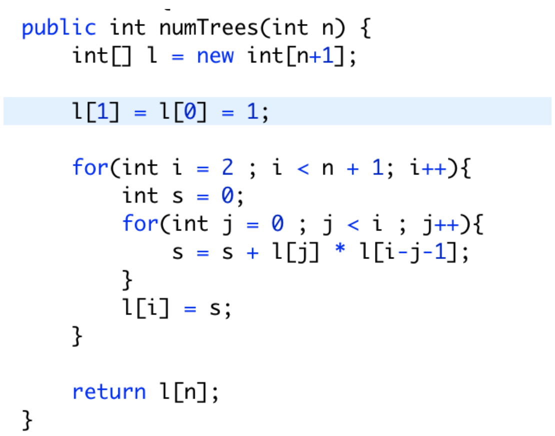


Figure 1: Example : $n = 3$, there are 5 different trees possible

If we count Binary Trees(BTs) instead of BSTs, how many different BTs are there?

[1] Problem Credits : https://leetcode.com/problems/unique-binary-search-trees/

Solution



```
public int numTrees(int n) {
    int[] l = new int[n+1];

    l[1] = l[0] = 1;

    for(int i = 2 ; i < n + 1; i++){
        int s = 0;
        for(int j = 0 ; j < i ; j++){
            s = s + l[j] * l[i-j-1];
        }
        l[i] = s;
    }

    return l[n];
}
```

Figure 2: *numTrees(3)* $= 5$

The given source code runs in $O(n^2)$ time.

Catalan number gives the answer directly. $C_n = \frac{(2n)!}{(n+1)! \times n!}$. Still this is not O(1) solution. This is still $O(n)$. However better than $O(n^2)$.

If we are interested in BTs, then there will be $C_n \times n!$ trees.