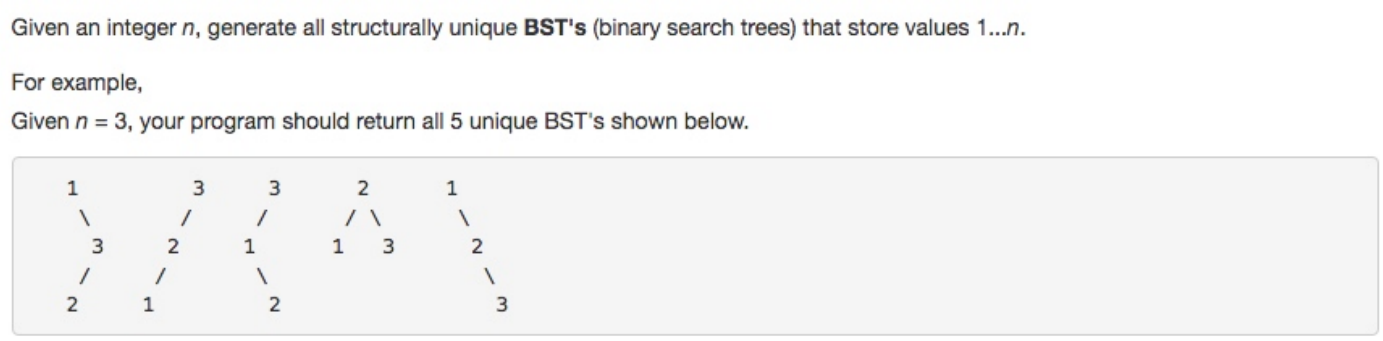
096. Unique Binary Search Trees

096. Unique Binary Search Trees

- **Dynamic Programming** + tree

Description

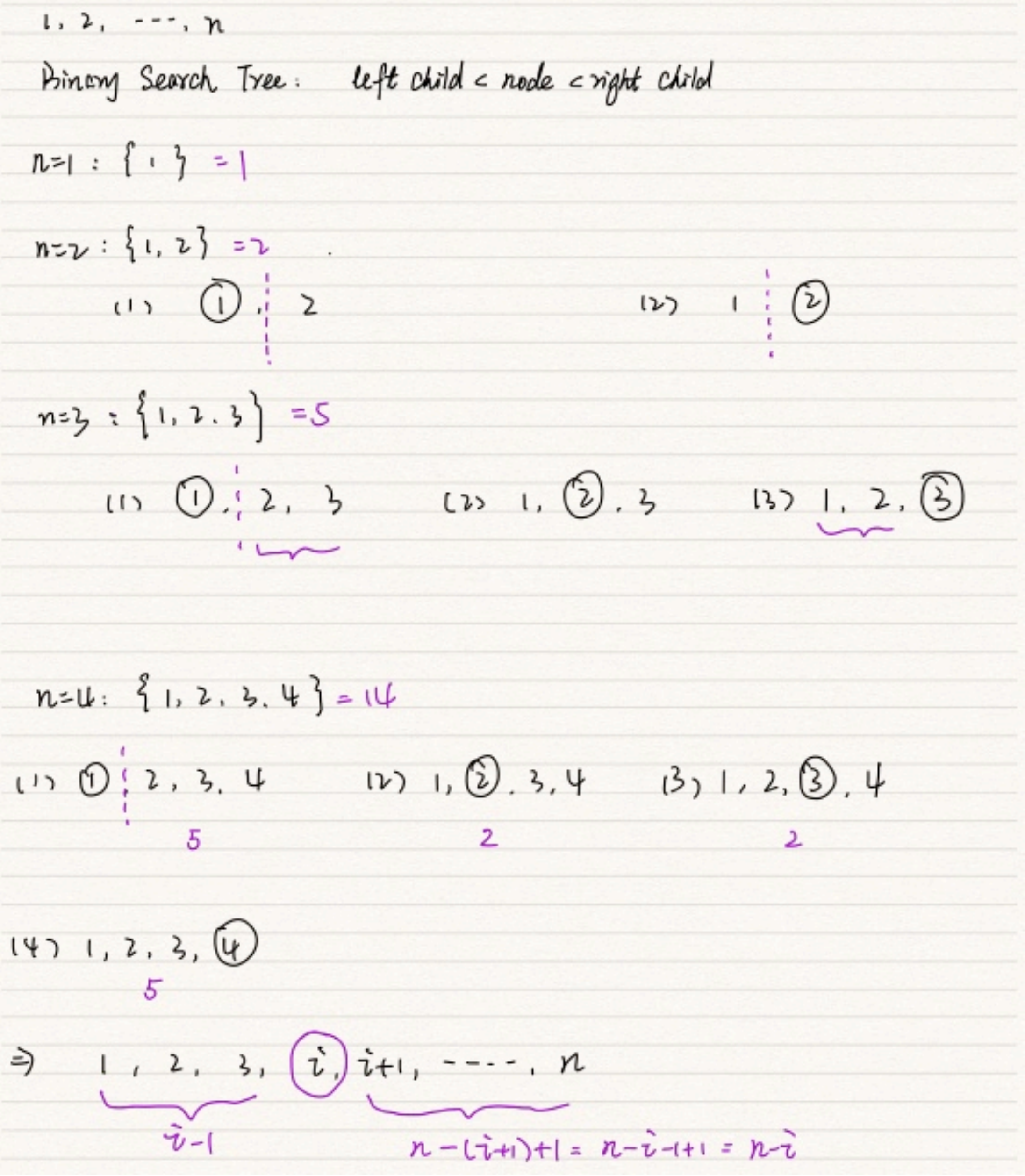


Given an integer *n*, generate all structurally unique **BST's** (binary search trees) that store values 1...*n*.

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
```

1. Thought line



1, 2, ---, n

Binary Search Tree: left child < node < right child

n=1 : {1} =1

n=2 : {1, 2} =2

(1) ①, 2   (2) 1 ②

n=3 : {1, 2, 3} =5

(1) ①, 2, 3   (2) 1, ②, 3   (3) 1, 2, ③

n=4: {1, 2, 3, 4} = 14

(1) ① 2, 3, 4   (2) 1, ②, 3, 4   (3) 1, 2, ③, 4

5   2   2

(4) 1, 2, 3, ④

5

$$\Rightarrow \underbrace{1,\ 2,\ 3,}_{i-1}\ \textcircled{i},\ \underbrace{i+1,\ ----,\ n}_{n-(i+1)+1\,=\,n-i-1+1\,=\,n-i}$$

2. Dynamic Programming + tree

```
class Solution {
public:
    int numTrees(int n) {
        vector<int> uniqueBST(n+1,1);

        for (int i=2; i<=n; ++i){
            int uniqueBSTofCurrentNode = 0;
            for (int node = 1; node<=i; ++node){
                int leftNodeNum = node-1, rightNodeNum = i-node;
                uniqueBSTofCurrentNode +=uniqueBST[leftNodeNum]*uniqueBST[rightNodeNum];
            }
            uniqueBST[i] = uniqueBSTofCurrentNode;
        }
        return uniqueBST[n];
    }
};
```