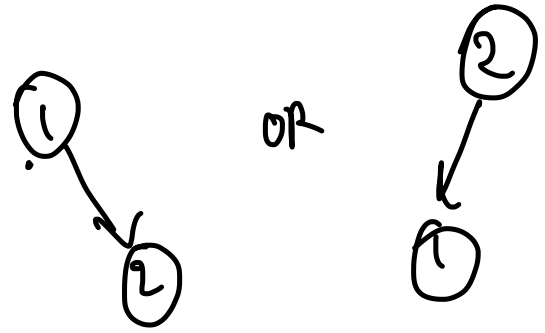
96 Unique Binary Search Trees

eg $n = 1$

only 1 way of arranging

$n = 2$

$(1, 2)$



$n = 3$

$(1, 2, 3) =$

Case1
① as root, then arranging ② ③

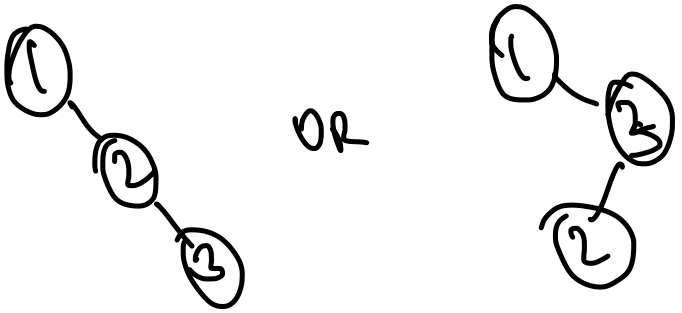
Case2
② as root, then arranging ① ③

Case3
③ as root, then arranging ① ②

Case1

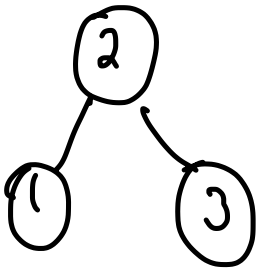
① as ②, ③ are greater than ①

both will be on right side



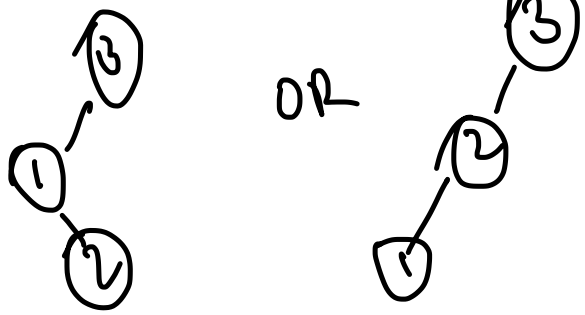
Case 2

(2) root (1) will be on left subtree
(3) will be on right subtree



Case 3

(3) root (1), (2) on left subtree



Total ways = 5

Dynamic Programming approach

For given n we have nodes

1, 2, 3, ... n

we have to consider case of each of them being root node

1, 2, 3, (x), ..., n

Suppose this case

dp[n] = For all x [dp[x] * dp[n-x]]

∴ dp[x] = ways of arranging 'x' nodes

Subproblem

dp[0] = 1

dp[1] = 1

```
for x in range(2, n):
    for y in range(1, x):
        dp[x] += dp[y] * dp[y-x]

return dp[n]
```