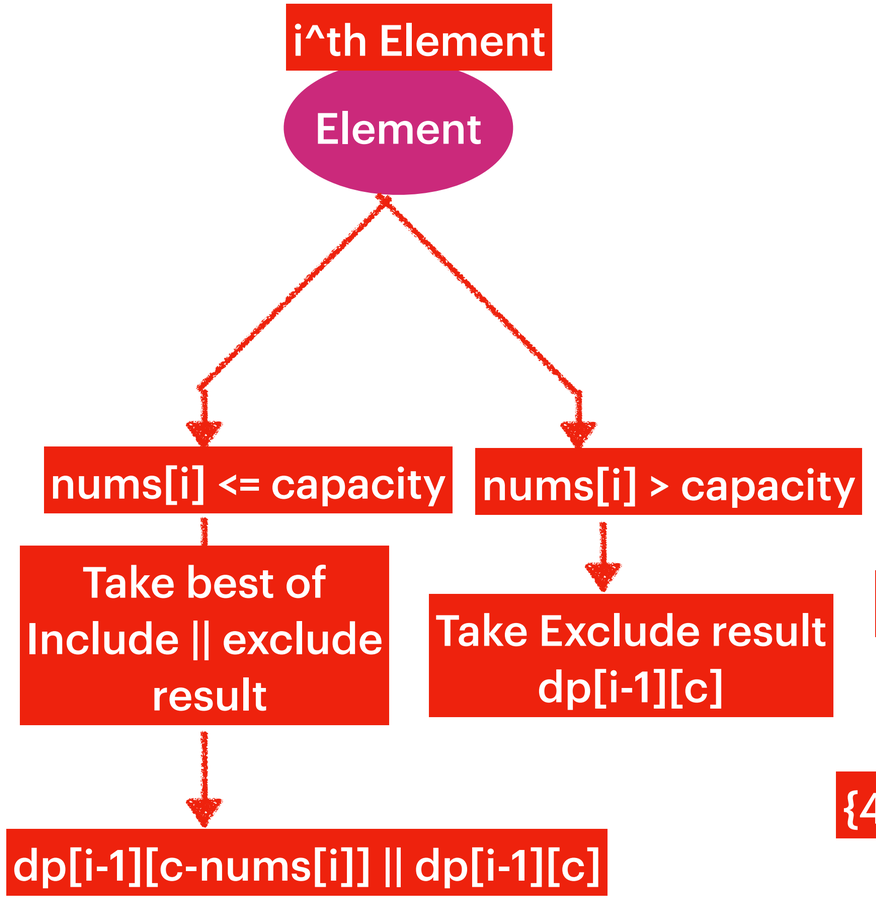
0	1	2
4	3	5

Total Sum = 12 , SubSetSum or Capacity = 6

Tabulation Approach : int[][] dp = new int[nums.length+1] [subSetSum+1]



{ }	0
{4}	1
{4,3}	2
{4,3,5}	3

SubSetSum						
0	1	2	3	4	5	6
True	False	False	False	False	False	False
True	False	False	False	True	False	False
True	False	False	True	False	False	False
True	False	False	True	False	True	False

[2,3]

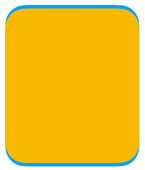
Jar —> 5ltr water

{2[0],3[1]}

dp[1][5] = dp[0][2] ; —-> dp[i-1][c-nums[i]]

dp[0][2] = true

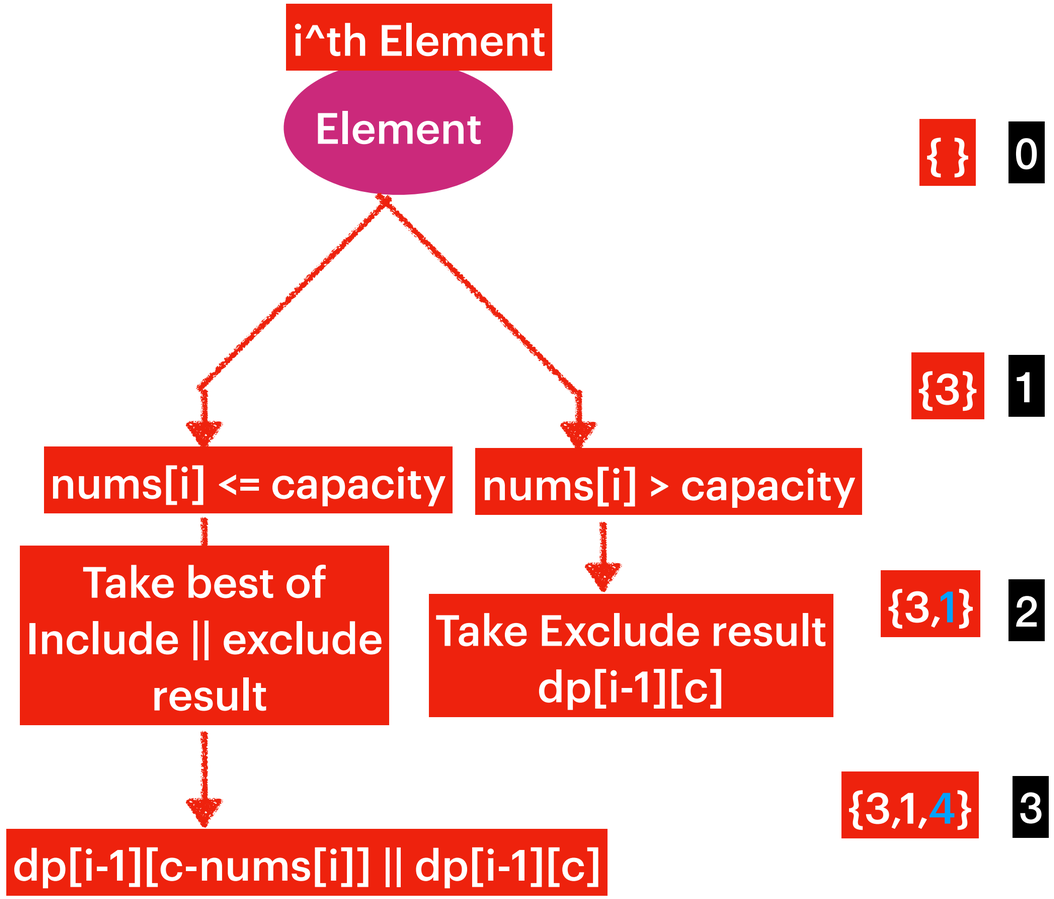
[2]



[2,3]

Total Sum = 8 , SubSetSum or Capacity = 4

Tabulation Approach : int[][] dp = new int[nums.length+1] [subSetSum+1]



SubSetSum

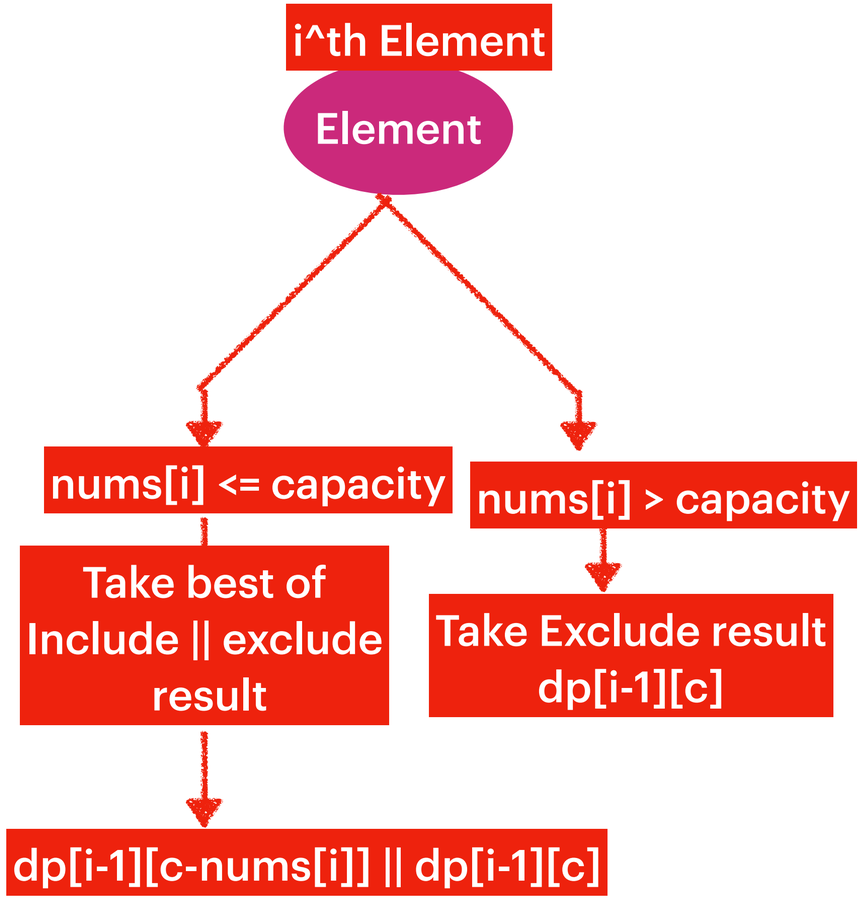
		0	1	2	3	4
{ }	0	True	False	False	False	False
{3}	1	True	False	False	True	False
{3,1}	2	True	True	False	True	True
{3,1,4}	3	True	True	False	True	True

0	1	2
3	1	4

Total Sum = 8 , SubSetSum or Capacity = 4

Solving with one dimensional array :

Tabulation Approach : int[] dp = new int [subSetSum+1]



SubSetSum

		0	1	2	3	4
{ }	i:0	True	False	False	False	False
{3}	i:1	True	False	False	True	False
{3,1}	i:2	True	True	False	True	True
{3,1,4}	i:3	True	True	False	True	True

494. Target Sum

Medium 6691 256 Add to List Share

You are given an integer array `nums` and an integer `target`.

You want to build an **expression** out of nums by adding one of the symbols `'+'` and `'-'` before each integer in nums and then concatenate all the integers.

- For example, if `nums = [2, 1]`, you can add a `'+'` before `2` and a `'-'` before `1` and concatenate them to build the expression `"+2-1"`.

Return the number of different **expressions** that you can build, which evaluates to `target`.

Example 1:

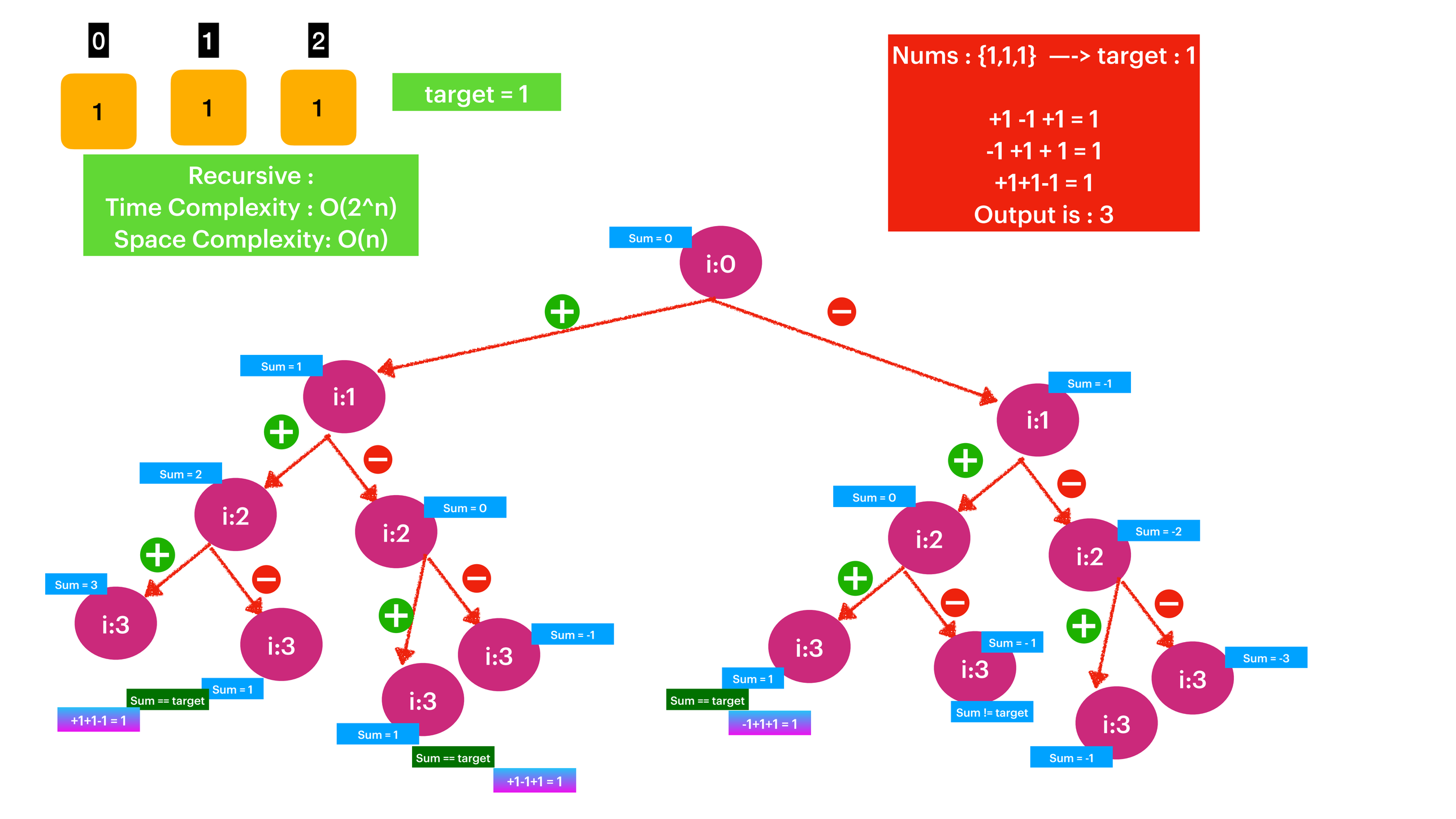
```
Input: nums = [1,1,1,1,1], target = 3
Output: 5
Explanation: There are 5 ways to assign symbols to make the
sum of nums be target 3.
-1 + 1 + 1 + 1 + 1 = 3
+1 - 1 + 1 + 1 + 1 = 3
+1 + 1 - 1 + 1 + 1 = 3
+1 + 1 + 1 - 1 + 1 = 3
+1 + 1 + 1 + 1 - 1 = 3
```

Example 2:

```
Input: nums = [1], target = 1
Output: 1
```

Constraints:

- `1 <= nums.length <= 20`
- `0 <= nums[i] <= 1000`
- `0 <= sum(nums[i]) <= 1000`
- `-1000 <= target <= 1000`


0
1
2
1
1
1
target = 1
Nums : {1,1,1} —-> target : 1
+1 -1 +1 = 1
-1 +1 + 1 = 1
+1+1-1 = 1
Output is : 3
Recursive :
Time Complexity : O(2^n)
Space Complexity: O(n)
Sum = 0
i:0
Sum = 1
i:1
Sum = -1
i:1
Sum = 2
i:2
Sum = 0
i:2
Sum = 0
i:2
Sum = -2
i:2
Sum = 3
i:3
i:3
Sum = -1
i:3
i:3
Sum = 1
Sum == target
+1+1-1 = 1
Sum = 1
Sum == target
+1-1+1 = 1
Sum = 1
Sum == target
-1+1+1 = 1
Sum = - 1
Sum != target
Sum = -3
i:3
Sum = -1