

# Grokking the Coding Interview: Patterns for Coding Questions





Pattern: Cyclic Sort

Intervals



Pattern: Tree Breadth

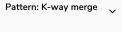
















# Solution Review: Problem Challenge 2

```
We'll cover the following

Target Sum (hard)

Example 1:
Example 2:
Solution
Code
Time and Space complexity
Space Optimized Solution
```

# Target Sum (hard)

You are given a set of positive numbers and a target sum 'S'. Each number should be assigned either a '+' or  $\cdot$ ' sign. We need to find the total ways to assign symbols to make the sum of the numbers equal to the target 'S'.

Example 1:

```
Input: {1, 1, 2, 3}, S=1
Output: 3
Explanation: The given set has '3' ways to make a sum of '1': {+1-1-2+3} & {-1+1-2+3} & {+1+1+2-3}
```

Example 2:

```
Input: {1, 2, 7, 1}, S=9
Output: 2
Explanation: The given set has '2' ways to make a sum of '9': {+1+2+7-1} & {-1+2+7+1}
```

#### Solution

This problem follows the **0/1 Knapsack pattern** and can be converted into Count of Subset Sum. Let's dig into this.

We are asked to find two subsets of the given numbers whose difference is equal to the given target 'S'. Take the first example above. As we saw, one solution is {+1-1-2+3}. So, the two subsets we are asked to find are {1, 3} & {1, 2} because,

```
(1 + 3) - (1 + 2) = 1
```

Now, let's say 'Sum(s1)' denotes the total sum of set 's1', and 'Sum(s2)' denotes the total sum of set 's2'. So the required equation is:

```
Sum(s1) - Sum(s2) = S
```

This equation can be reduced to the subset sum problem. Let's assume that 'Sum(num)' denotes the total sum of all the numbers, therefore:

```
Sum(s1) + Sum(s2) = Sum(num)
```

Let's add the above two equations:

```
=> Sum(s1) - Sum(s2) + Sum(s1) + Sum(s2) = S + Sum(num)

=> 2 * Sum(s1) = S + Sum(num)

=> Sum(s1) = (S + Sum(num)) / 2
```

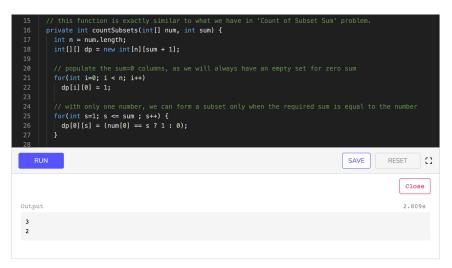
Which means that one of the set 's1' has a sum equal to (5 + Sum(num)) / 2. This essentially converts our problem to: "Find the count of subsets of the given numbers whose sum is equal to (5 + Sum(num)) / 2"

### Code

Let's take the dynamic programming code of Count of Subset Sum and extend it to solve this problem:







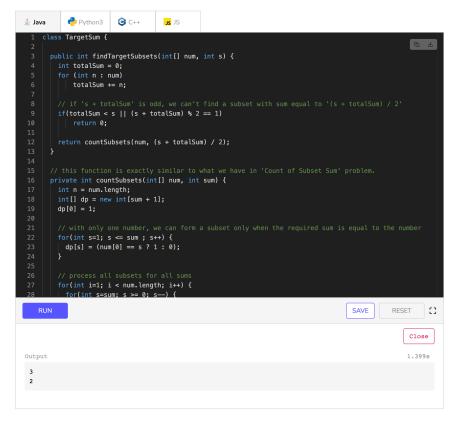
Time and Space complexity

The above solution has time and space complexity of O(N\*S), where 'N' represents total numbers and 'S' is the desired sum.

We can further improve the solution to use only  $\mathcal{O}(S)$  space.

## Space Optimized Solution

Here is the code for the space-optimized solution, using only a single array:



Interviewing soon? We've partnered with Hired so that companies apply to you instead of you  $\quad \times$  applying to them. See how  $\odot$ 

```
      ← Back
      Next →

      Problem Challenge 2
      Introduction

      ✓ Mark as Completed

      ① Report an Issue ② Ask a Question
```