Target Sum

We'll cover the following

Given a set of positive numbers (non zero) and a target sum ‘S’. Each number should be assigned either a ‘+’ or ‘-’ sign. We need to find out total ways to assign symbols to make the sum of numbers equal to 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}
```

Try it yourself

Try solving this question here:

Python3

```
def find_target_subsets(num, s):
  # TODO: Write your code here
  return -1


def main():
  print("Total ways: " + str(find_target_subsets([1, 1, 2, 3], 1)))
  print("Total ways: " + str(find_target_subsets([1, 2, 7, 1], 9)))


main()
```

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 fin {
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
```

This essentially converts our problem to: "Find count of subsets of the given numbers whose sum is equal to",

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

Code

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

Python3

```
def find_target_subsets(num, s):
  if any(i < 1 for i in num):
    return -1 #invalid input, the problem expects only positive numbers
  totalSum = sum(num)

  # if 's + totalSum' is odd, we can't find a subset with sum equal to '(s +
  if totalSum < s or (s + totalSum) % 2 == 1:
    return 0

  return count_subsets(num, int((s + totalSum) / 2))


# this function is exactly similar to what we have in 'Count of Subset Sum'
def count_subsets(num, s):
  n = len(num)
  dp = [[0 for x in range(s+1)] for y in range(n)]

  # populate the sum = 0 columns, as we will always have an empty set for ze
  for i in range(0, n):
    dp[i][0] = 1

  # with only one number, we can form a subset only when the required sum is
  # equal to the number
  for s in range(1, s+1):
    dp[0][s] = 1 if num[0] == s else 0
```

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 $O(S)$ space.

Space Optimized Solution

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

Python3

```
def find_target_subsets(num, s):
  if any(i < 1 for i in num):
    return -1 #invalid input, the problem expects only positive numbers

  totalSum = sum(num)

  # if 's + totalSum' is odd, we can't find a subset with sum equal to '(s +
  if totalSum < s or (s + totalSum) % 2 == 1:
    return 0

  return count_subsets(num, int((s + totalSum) / 2))


# this function is exactly similar to what we have in 'Count of Subset Sum'
def count_subsets(num, sum):
  n = len(num)
  dp = [0 for x in range(sum+1)]
  dp[0] = 1

  # with only one number, we can form a subset only when the required sum is
  for s in range(1, sum+1):
    dp[s] = 1 if num[0] == s else 0

  # process all subsets for all sums
  for i in range(1, n):
    for s in range(sum, -1, -1):
      if s >= num[i]:
```

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

← Back
Count of Subset Sum

Next →
Unbounded Knapsack

Mark as Completed

Report an Issue