TargetSum.java

```
1   package com.example.dynamicProgramming;
2
3   import java.util.Arrays;
4
5   public class TargetSum {
6
7       static int countPartitionsUtil(int ind, int target, int[] arr, int[][] dp) {
8           // Base case: If we have reached the first element
9   1       if (ind == 0) {
10              // Check if the target is 0 and the first element is also 0
11  2           if (target == 0 && arr[0] == 0)
12  1               return 2;
13              // Check if the target is equal to the first element or 0
14  2           if (target == 0 || target == arr[0])
15  1               return 1;
16              return 0;
17          }
18
19          // If the result for this subproblem has already been calculated, return it
20  1       if (dp[ind][target] != -1)
21  1           return dp[ind][target];
22
23          // Calculate the number of ways without taking the current element
24  1       int notTaken = countPartitionsUtil(ind - 1, target, arr, dp);
25
26          // Initialize the number of ways taking the current element as 0
27          int taken = 0;
28
29          // If the current element is less than or equal to the target, calculate 'taken'
30  2       if (arr[ind] <= target)
31  2           taken = countPartitionsUtil(ind - 1, target - arr[ind], arr, dp);
32
33          // Store the result in the dp array and return it
34  2       return dp[ind][target] = (notTaken + taken);
35      }
36
37      // Function to find the number of ways to achieve the target sum
38      static int targetSum(int n, int target, int[] arr) {
39          int totSum = 0;
40
41          // Calculate the total sum of elements in the array
42  2       for (int i = 0; i < arr.length; i++) {
43  1           totSum += arr[i];
44          }
45
46          // Checking for edge cases
47  3       if (totSum - target < 0)
48              return 0;
49  3       if ((totSum - target) % 2 == 1)
50              return 0;
51
52          // Calculate the second sum based on the total sum and the target
53  2       int s2 = (totSum - target) / 2;
54
55          // Create a 2D array to store results of subproblems
56  1       int dp[][] = new int[n][s2 + 1];
57
58          // Initialize the dp array with -1 to indicate that subproblems are not solved
59          // yet
60          for (int row[] : dp)
61  1           Arrays.fill(row, -1);
62
63          // Call the countPartitionsUtil function to calculate the number of ways
```

```
64 2        return countPartitionsUtil(n - 1, s2, arr, dp);
65      }
66
67      static int mod = (int) (Math.pow(10, 9) + 7);
68
69      // Function to find the number of ways to achieve the target sum
70      static int findWays(int[] num, int tar) {
71          int n = num.length;
72
73          // Create a 2D array to store results of subproblems
74 1        int[][] dp = new int[n][tar + 1];
75
76          // Initialize the dp array for the first element of the array
77 1        if (num[0] == 0)
78              dp[0][0] = 2; // 2 cases - pick and not pick
79          else
80              dp[0][0] = 1; // 1 case - not pick
81
82 3        if (num[0] != 0 && num[0] <= tar)
83              dp[0][num[0]] = 1; // 1 case - pick
84
85          // Fill the dp array using dynamic programming
86 2        for (int ind = 1; ind < n; ind++) {
87 2            for (int target = 0; target <= tar; target++) {
88 1                int notTaken = dp[ind - 1][target];
89
90                  int taken = 0;
91 2                if (num[ind] <= target)
92 2                    taken = dp[ind - 1][target - num[ind]];
93
94 2                dp[ind][target] = (notTaken + taken) % mod;
95              }
96          }
97
98 2        return dp[n - 1][tar];
99      }
100
101     // Function to calculate the number of ways to achieve the target sum
102     static int targetSum1(int n, int target, int[] arr) {
103         int totSum = 0;
104
105         // Calculate the total sum of elements in the array
106 2       for (int i = 0; i < n; i++) {
107 1           totSum += arr[i];
108         }
109
110         // Checking for edge cases
111 6       if (totSum - target < 0 || (totSum - target) % 2 == 1)
112             return 0;
113
114 3       return findWays(arr, (totSum - target) / 2);
115     }
116
117 }
```

Mutations

9	1. negated conditional → KILLED
11	1. negated conditional → SURVIVED 2. negated conditional → KILLED
12	1. replaced int return with 0 for com/example/dynamicProgramming/TargetSum::countPartitionsUtil → NO_COVERAGE
14	1. negated conditional → KILLED 2. negated conditional → KILLED
15	1. replaced int return with 0 for com/example/dynamicProgramming/TargetSum::countPartitionsUtil → KILLED
20	1. negated conditional → KILLED

21	1. replaced int return with 0 for com/example/dynamicProgramming/TargetSum::countPartitionsUtil → KILLED
24	1. Replaced integer subtraction with addition → KILLED
30	1. changed conditional boundary → KILLED 2. negated conditional → KILLED
31	1. Replaced integer subtraction with addition → KILLED 2. Replaced integer subtraction with addition → KILLED
34	1. Replaced integer addition with subtraction → KILLED 2. replaced int return with 0 for com/example/dynamicProgramming/TargetSum::countPartitionsUtil → KILLED
42	1. negated conditional → KILLED 2. changed conditional boundary → KILLED
43	1. Replaced integer addition with subtraction → KILLED
47	1. Replaced integer subtraction with addition → SURVIVED 2. changed conditional boundary → SURVIVED 3. negated conditional → KILLED
49	1. negated conditional → KILLED 2. Replaced integer subtraction with addition → SURVIVED 3. Replaced integer modulus with multiplication → SURVIVED
53	1. Replaced integer division with multiplication → KILLED 2. Replaced integer subtraction with addition → SURVIVED
56	1. Replaced integer addition with subtraction → KILLED
61	1. removed call to java/util/Arrays::fill → KILLED
64	1. replaced int return with 0 for com/example/dynamicProgramming/TargetSum::targetSum → KILLED 2. Replaced integer subtraction with addition → KILLED
74	1. Replaced integer addition with subtraction → KILLED
77	1. negated conditional → KILLED
82	1. negated conditional → KILLED 2. negated conditional → KILLED 3. changed conditional boundary → KILLED
86	1. negated conditional → KILLED 2. changed conditional boundary → KILLED
87	1. changed conditional boundary → KILLED 2. negated conditional → KILLED
88	1. Replaced integer subtraction with addition → KILLED
91	1. changed conditional boundary → KILLED 2. negated conditional → KILLED
92	1. Replaced integer subtraction with addition → KILLED 2. Replaced integer subtraction with addition → KILLED
94	1. Replaced integer modulus with multiplication → KILLED 2. Replaced integer addition with subtraction → KILLED
98	1. Replaced integer subtraction with addition → KILLED 2. replaced int return with 0 for com/example/dynamicProgramming/TargetSum::findWays → KILLED
106	1. changed conditional boundary → KILLED 2. negated conditional → KILLED
107	1. Replaced integer addition with subtraction → KILLED
111	1. negated conditional → KILLED 2. Replaced integer subtraction with addition → SURVIVED 3. negated conditional → KILLED 4. changed conditional boundary → SURVIVED 5. Replaced integer modulus with multiplication → SURVIVED 6. Replaced integer subtraction with addition → SURVIVED
114	1. Replaced integer subtraction with addition → SURVIVED 2. Replaced integer division with multiplication → KILLED 3. replaced int return with 0 for com/example/dynamicProgramming/TargetSum::targetSum1 → KILLED

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NEGATE_CONDITIONALS
- NULL_RETURNS
- PRIMITIVE_RETURNS
- TRUE_RETURNS
- VOID_METHOD_CALLS

Tests examined

- com.example.dynamicProgramming.TargetSumTest.test1(com.example.dynamicProgramming.TargetSumTest) (0 ms)

Report generated by PIT 1.15.0