TargetSum.java

```
1       package 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 yet
59              for (int row[] : dp)
60  1               Arrays.fill(row, -1);
61
62              // Call the countPartitionsUtil function to calculate the number of ways
63  2           return countPartitionsUtil(n - 1, s2, arr, dp);
64          }
65      static int mod = (int) (Math.pow(10, 9) + 7);
66  
67      // Function to find the number of ways to achieve the target sum
68      static int findWays(int[] num, int tar) {
69          int n = num.length;
70  
71          // Create a 2D array to store results of subproblems
72  1       int[][] dp = new int[n][tar + 1];
73  
74          // Initialize the dp array for the first element of the array
75  1       if (num[0] == 0)
76              dp[0][0] = 2; // 2 cases - pick and not pick
77          else
78              dp[0][0] = 1; // 1 case - not pick
79  
80  3       if (num[0] != 0 && num[0] <= tar)
81              dp[0][num[0]] = 1; // 1 case - pick
82  
83          // Fill the dp array using dynamic programming
84  2       for (int ind = 1; ind < n; ind++) {
85  2           for (int target = 0; target <= tar; target++) {
86  1               int notTaken = dp[ind - 1][target];
87  
88                  int taken = 0;
89  2               if (num[ind] <= target)
90  2                   taken = dp[ind - 1][target - num[ind]];
91  
92  2               dp[ind][target] = (notTaken + taken) % mod;
93              }
94          }
95  
96  2       return dp[n - 1][tar];
97      }
98  
99      // Function to calculate the number of ways to achieve the target sum
100     static int targetSum1(int n, int target, int[] arr) {
101         int totSum = 0;
102 
103         // Calculate the total sum of elements in the array
104 2       for (int i = 0; i < n; i++) {
105 1           totSum += arr[i];
106         }
107 
108         // Checking for edge cases
109 6       if (totSum - target < 0 || (totSum - target) % 2 == 1)
110             return 0;
111 
112 3       return findWays(arr, (totSum - target) / 2);
113     }
114 
115 }
```

Mutations

```
9   1. negated conditional → KILLED
11  1. negated conditional → SURVIVED
    2. negated conditional → KILLED
12  1. replaced int return with 0 for DynamicProgramming/TargetSum::countPartitionsUtil →
    NO_COVERAGE
14  1. negated conditional → KILLED
    2. negated conditional → KILLED
15  1. replaced int return with 0 for DynamicProgramming/TargetSum::countPartitionsUtil →
    KILLED
20  1. negated conditional → KILLED
21  1. replaced int return with 0 for DynamicProgramming/TargetSum::countPartitionsUtil →
    KILLED
24  1. Replaced integer subtraction with addition → KILLED
30  1. negated conditional → KILLED
    2. changed conditional boundary → KILLED
31  1. Replaced integer subtraction with addition → KILLED
```

```
     2. Replaced integer subtraction with addition → KILLED
34   1. replaced int return with 0 for DynamicProgramming/TargetSum::countPartitionsUtil →
     KILLED
     2. Replaced integer addition with subtraction → KILLED
42   1. negated conditional → KILLED
     2. changed conditional boundary → KILLED
43   1. Replaced integer addition with subtraction → KILLED
47   1. changed conditional boundary → SURVIVED
     2. Replaced integer subtraction with addition → SURVIVED
     3. negated conditional → KILLED
49   1. negated conditional → KILLED
     2. Replaced integer modulus with multiplication → SURVIVED
     3. Replaced integer subtraction with addition → SURVIVED
53   1. Replaced integer subtraction with addition → SURVIVED
     2. Replaced integer division with multiplication → KILLED
56   1. Replaced integer addition with subtraction → KILLED
60   1. removed call to java/util/Arrays::fill → KILLED
63   1. Replaced integer subtraction with addition → KILLED
     2. replaced int return with 0 for DynamicProgramming/TargetSum::targetSum → KILLED
72   1. Replaced integer addition with subtraction → KILLED
75   1. negated conditional → KILLED
80   1. negated conditional → KILLED
     2. changed conditional boundary → KILLED
     3. negated conditional → KILLED
84   1. negated conditional → KILLED
     2. changed conditional boundary → KILLED
85   1. negated conditional → KILLED
     2. changed conditional boundary → KILLED
86   1. Replaced integer subtraction with addition → KILLED
89   1. negated conditional → KILLED
     2. changed conditional boundary → KILLED
90   1. Replaced integer subtraction with addition → KILLED
     2. Replaced integer subtraction with addition → KILLED
92   1. Replaced integer addition with subtraction → KILLED
     2. Replaced integer modulus with multiplication → KILLED
96   1. replaced int return with 0 for DynamicProgramming/TargetSum::findWays → KILLED
     2. Replaced integer subtraction with addition → KILLED
104  1. negated conditional → KILLED
     2. changed conditional boundary → KILLED
105  1. Replaced integer addition with subtraction → KILLED
109  1. Replaced integer subtraction with addition → SURVIVED
     2. Replaced integer subtraction with addition → SURVIVED
     3. negated conditional → KILLED
     4. negated conditional → KILLED
     5. Replaced integer modulus with multiplication → SURVIVED
     6. changed conditional boundary → SURVIVED
112  1. replaced int return with 0 for DynamicProgramming/TargetSum::targetSum1 → KILLED
     2. Replaced integer division with multiplication → KILLED
     3. Replaced integer subtraction with addition → SURVIVED
```

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

- DynamicProgramming.TragetSumTest.test1(DynamicProgramming.TragetSumTest) (0 ms)

Report generated by PIT 1.15.0