TargetSum.java

```
   package com.example;

   import java.util.Arrays;

   public class TargetSum {

           static int countPartitionsUtil(int ind, int target, int[] arr, int[][] dp) {
           // Base case: If we have reached the first element
1          if (ind == 0) {
               // Check if the target is 0 and the first element is also 0
2              if (target == 0 && arr[0] == 0)
1                  return 2;
               // Check if the target is equal to the first element or 0
2              if (target == 0 || target == arr[0])
1                  return 1;
               return 0;
           }

           // If the result for this subproblem has already been calculated, return it
1          if (dp[ind][target] != -1)
1              return dp[ind][target];

           // Calculate the number of ways without taking the current element
1          int notTaken = countPartitionsUtil(ind - 1, target, arr, dp);

           // Initialize the number of ways taking the current element as 0
           int taken = 0;

           // If the current element is less than or equal to the target, calculate 'taken'
2          if (arr[ind] <= target)
2              taken = countPartitionsUtil(ind - 1, target - arr[ind], arr, dp);

           // Store the result in the dp array and return it
2          return dp[ind][target] = (notTaken + taken);
       }

       // Function to find the number of ways to achieve the target sum
       static int targetSum(int n, int target, int[] arr) {
           int totSum = 0;

           // Calculate the total sum of elements in the array
3          for (int i = 0; i < arr.length; i++) {
1              totSum += arr[i];
           }

           // Checking for edge cases
3          if (totSum - target < 0)
               return 0;
3          if ((totSum - target) % 2 == 1)
               return 0;

           // Calculate the second sum based on the total sum and the target
2          int s2 = (totSum - target) / 2;

           // Create a 2D array to store results of subproblems
1          int dp[][] = new int[n][s2 + 1];

           // Initialize the dp array with -1 to indicate that subproblems are not solved yet
           for (int row[] : dp)
1              Arrays.fill(row, -1);

           // Call the countPartitionsUtil function to calculate the number of ways
2          return countPartitionsUtil(n - 1, s2, arr, dp);
       }
       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  3       for (int ind = 1; ind < n; ind++) {
85  3           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 3       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 com/example/TargetSum::countPartitionsUtil → NO_COVERAGE
14   1. negated conditional → KILLED
     2. negated conditional → KILLED
15   1. replaced int return with 0 for com/example/TargetSum::countPartitionsUtil → KILLED
20   1. negated conditional → KILLED
21   1. replaced int return with 0 for com/example/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/TargetSum::countPartitionsUtil → KILLED
     1. changed conditional boundary → KILLED
42   2. Changed increment from 1 to -1 → KILLED
     3. negated conditional → KILLED
43   1. Replaced integer addition with subtraction → KILLED
     1. changed conditional boundary → KILLED
47   2. Replaced integer subtraction with addition → KILLED
     3. negated conditional → KILLED
     1. Replaced integer subtraction with addition → SURVIVED
49   2. Replaced integer modulus with multiplication → SURVIVED
     3. negated conditional → KILLED
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 com/example/TargetSum::targetSum → KILLED
72   1. Replaced integer addition with subtraction → KILLED
75   1. negated conditional → KILLED
     1. changed conditional boundary → KILLED
80   2. negated conditional → KILLED
     3. negated conditional → KILLED
     1. changed conditional boundary → KILLED
84   2. Changed increment from 1 to -1 → KILLED
     3. negated conditional → KILLED
     1. changed conditional boundary → KILLED
85   2. Changed increment from 1 to -1 → KILLED
     3. negated conditional → KILLED
86   1. Replaced integer subtraction with addition → KILLED
89   1. changed conditional boundary → KILLED
     2. negated conditional → 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 integer subtraction with addition → KILLED
     2. replaced int return with 0 for com/example/TargetSum::findWays → KILLED
     1. changed conditional boundary → KILLED
104  2. Changed increment from 1 to -1 → KILLED
     3. negated conditional → KILLED
105  1. Replaced integer addition with subtraction → KILLED
     1. changed conditional boundary → KILLED
     2. Replaced integer subtraction with addition → KILLED
109  3. Replaced integer subtraction with addition → SURVIVED
     4. Replaced integer modulus with multiplication → SURVIVED
     5. negated conditional → KILLED
     6. negated conditional → KILLED
     1. Replaced integer subtraction with addition → SURVIVED
112  2. Replaced integer division with multiplication → KILLED
     3. replaced int return with 0 for com/example/TargetSum::targetSum1 → KILLED
```

Active mutators

- BOOLEAN_FALSE_RETURN
- BOOLEAN_TRUE_RETURN
- CONDITIONALS_BOUNDARY_MUTATOR
- EMPTY_RETURN_VALUES
- INCREMENTS_MUTATOR
- INVERT_NEGS_MUTATOR
- MATH_MUTATOR
- NEGATE_CONDITIONALS_MUTATOR
- NULL_RETURN_VALUES
- PRIMITIVE_RETURN_VALS_MUTATOR
- VOID_METHOD_CALL_MUTATOR

Tests examined

- com.example.TargetSumTest.testSingleElementArray(com.example.TargetSumTest) (0 ms)
- com.example.TargetSumTest.test1(com.example.TargetSumTest) (1 ms)
- com.example.TargetSumTest.testLargeArray(com.example.TargetSumTest) (1 ms)
- com.example.TargetSumTest.testEmptyArray(com.example.TargetSumTest) (0 ms)

Report generated by PIT 1.5.0