Problem

Submissions

Leaderboard

Discussions

Editorial

Given an amount and the denominations of coins available, determine how many ways change can be made for amount. There is a limitless supply of each coin type.

Example

$n = 3$

$c = [8, 3, 1, 2]$

There are 3 ways to make change for $n = 3$: $\{1, 1, 1\}$, $\{1, 2\}$, and $\{3\}$.

Function Description

Complete the getWays function in the editor below.

getWays has the following parameter(s):

- int n: the amount to make change for
- int c[m]: the available coin denominations

Returns

- int: the number of ways to make change

Input Format

The first line contains two space-separated integers n and m, where:

n is the amount to change

m is the number of coin types

The second line contains m space-separated integers that describe the values of each coin type.

Constraints

- $1 \leq c[i] \leq 50$
- $1 \leq n \leq 250$
- $1 \leq m \leq 50$
- Each $c[i]$ is guaranteed to be distinct.

Hints

Solve overlapping subproblems using Dynamic Programming (DP):

You can solve this problem recursively but will not pass all the test cases without optimizing to eliminate the overlapping subproblems. Think of a way to store and reference previously computed solutions to avoid solving the same subproblem multiple times. * Consider the degenerate cases:
- How many ways can you make change for 0 cents? - How many ways can you make change for > 0 cents if you have no coins? * If you're having trouble defining your solutions store, then think about it in terms of the base case $(n = 0)$. - The answer may be larger than a 32-bit integer.

```
# Complete the 'getWays' function below.
#
# The function is expected to return a LONG_INTEGER.
# The function accepts following parameters:
#  1. INTEGER n
#  2. LONG_INTEGER_ARRAY c
#
# DP Solution
# Create a DP array of solutions of ways to change coins until n
# Refer back to the previous solutions when counting current coin
def getWays(n, coins):
    dp = [0] * (n + 1)
    dp[0] = 1 # One way to return 0 coins

    for coin in sorted(coins):
        for idx in range(len(dp)):
            if coin <= idx: dp[idx] += dp[idx - coin]

    return dp[-1]




if __name__ == '__main__':…
```

Line: 17 Col: 14

Upload Code as File   Test against custom input   Run Code   Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

Test case 0

Test case 1

Test case 2

Test case 3

Compiler Message

```
Success
```

Input (stdin)   Download

```
4 3
1 2 3
```