**Change Problem**

*Find the minimum number of coins needed to make change.*

**Input:** An integer *money* and an integer array *Coins* = (*coin*1, ..., *coin*d).

**Output:** The minimum number of coins with denominations *Coins* that changes *money*

Imagine that you bought a textbook for $69.24, which you paid for with $70 in cash. You are due 76 cents in change, and the cashier now must make a decision whether to give you a fistful of 76 1-cent coins or just four coins (25 + 25 + 25 + 1 = 76). Making change in this example is easy, but it casts light on a more general problem — how can a cashier make change using the fewest number of coins? Different currencies have different possible coin values, or **denominations**. In the US, the coin denominations are (1, 5, 10, 25, 50 100); in the Roman Republic before Julius Caesar’s assassination, they were (1, 4, 5, 10, 24, 30, 40, 120).

**Input Format.** The first line of the input contains the positive integer *money*. The second line contains a comma-delimited list of positive integers *Coins*.

**Output Format.** The minimum number of coins with denominations *Coins* that changes *money*.

**Constraints.** *money* ≤ 20,000; |*Coins*| ≤ 7, 1 ≤ |*coini*| ≤ 100 for all *i*.

**SAMPLE DATASET:**

Input:

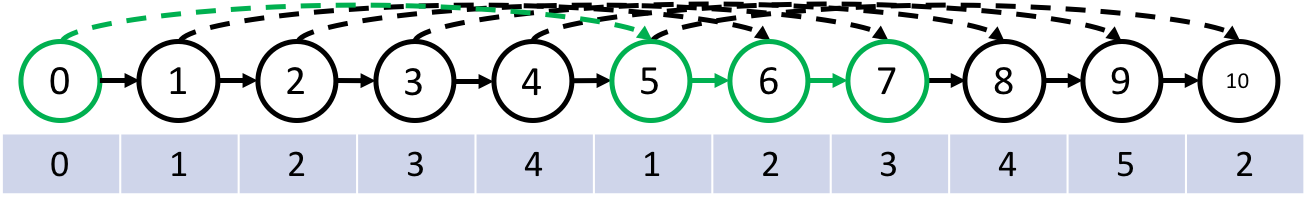
7

1,5

Output:

3

The smallest number of coins to change 7 is 3 (5+1+1).



**TEST DATASET 1:**

Input:

10

1,2,3,4,5,10

Output:

1

This dataset makes sure that your code is correctly considering the last coin denomination in the *Coins* array. If your solution has an off-by-one indexing mistake while iterating over the *Coins* array it could be possible that the last *coin* is not considered. In this case code run on this dataset will output 2 instead of the correct answer 1.

**TEST DATASET 2:**

Input:

10

10,5,4,3,2,1

Output:

1

This dataset makes sure that your code is correctly considering the first coin denomination in the *Coins* array. If your solution has an off-by-one indexing mistake while iterating over the *Coins* array it could be possible that the first *coin* is not considered. In this case code run on this dataset will output 2 instead of the correct answer 1.

**TEST DATASET 3:**

Input:

11

1,5

Output:

3

This dataset checks if your code correctly returns the final value in the array in the dynamic programming approach to solving this problem. It is possible that an off-by-one indexing error (could be related to confusing 0/1 indexing) results in your code outputting the minimum number of coins needed to make change for *money -1* instead of *money*. In this case your code will output 2 instead of the correct value of 3.

**TEST DATASET 4:**

Input:

12

9,6,1

Output:

2

This test dataset checks to make sure you are not using a greedy algorithm to solve this problem. While a greedy algorithm in which the largest valued *coin* is used on each iteration may work in some cases, it will fail on this dataset. A greedy approach would start by using the *9* *coin* which would only allow for the use of 3 of the 1 *coin* to get to 12, resulting in an output of 4. Using 2 of the 6 *coin* will result in exact change with only 2 coins.