The Fewest Coins - Alphastar CC41B Knapsack Practice, problem 1 (from USACO 2006?)

Saturday, October 7, 2023 7:18 PM

The Fewest Coins
Farmer John has gone to town to buy some farm supplies. Being a very efficient man, he always pays for his goods in such a way that the smallest number of coins changes hands, i.e., the number of coins he uses to pay plus the number of coins he receives in change is minimized. Help him to determine what this minimum number is.
FJ wants to buy T (1 <= T <= 10,000) cents of supplies. The currency system has N (1 <= N <= 100) different coins, with values V1, V2, ..., VN (1 <= Vi <= 120). Farmer John is carrying C1 coins of value V1, C2 coins of value V2, ..., and CN coins of value VN (0 <= Ci <= 10,000). The shopkeeper has an unlimited supply of all the coins, and always makes change in the most efficient manner (although Farmer John must be sure to pay in a way that makes it possible to make the correct change).
PROBLEM NAME: fewcoins
INPUT FORMAT:
- Line 1: Two space-separated integers: N and T.
- Line 2: N space-separated integers, respectively V1, V2, ..., VN coins (V1...VN).
- Line 3: N space-separated integers, respectively C1, C2, ..., CN

SAMPLE INPUT:
3 70
5 25 50
5 2 1
INPUT DETAILS:
Farmer John has 5x5 cent coins, 2x25 cent coins and 1x50 cent coin. His bill is 70 cents.
OUTPUT FORMAT:
- Line 1: A line containing a single integer, the minimum number of coins involved in a payment and change-making. If it is impossible for Farmer John to pay and receive exact change, output -1.

SAMPLE OUTPUT:
3
OUTPUT DETAILS:
Farmer John pays 75 cents using a 50 cents and a 25 cents coin, and receives a 5 cents coin in change, for a total of 3 coins used in the transaction.

From <https://lms.alphastar.academy/mod/quiz/attempt.php?attempt=462764&cmid=85510>

I'm writing these notes after finishing the problem.

The largest problem I ran into was how to figure the change out, until I realized that you have to make the amount you're paying and the amount of change. Then, it was just a matter of figuring out how to represent everything.

I settled on a 2D array, dp[x][y]. X represents the current amt of money, the array inside holds in the 0th pos the # of coins the most optimal solution to make it has, and each subsequent index has the amount of the k-1th coin used.

Then, I loop through each coin value (for int k = 0; k < n; k++), and then inside it, each money (for int x = 0; x <= t + maxcoinvalue; x++).
For each state: k, x:
We check if a solution exists to make x money.
If it does, we check the state to see if we can add another one of the current coin.

If so, we check if such a solution is better than the one already for val[k] + x.
If this is true, then we replace dp[val[k] + x] with the better state (dp[x] + 1 more of coin k)
If val[k] + x is larger than t, we add the optimal solution for change (val[k] + x - t) to the state. Then we compare the #coins to the current best solution at dp[t], if it's better then we replace it.

Once we're done, we output dp[t][0].