Gabe McKay
Terisa Le
Reeve Masilamani

Homework 2

ECS 122A
October 24, 2025

Problem 1

Part 2

$dp[i] \doteq$ The Number of coins needed to reach 0 cents from i cents by assigning coins and decreasing i. If it is impossible, then we set it to some arbitrary value, ∞.

Recurrence:
if $i - d_j > 0$ and $dp[i - d_j] \neq \infty$:
  $dp[i] = min(dp[i], dp[i - d_j] + 1)$
d_j represents the j-th coin.

Initialization: $dp[0] = 0$

for i = 1 to n:
  smallestValue $\leftarrow \infty$
  for j = 1 to c:
    if $i - d_j > 0$ and $dp[i - d_j] \neq \infty$:
      smallestValue = min(smallestValue, $dp[i - d_j] + 1$)
  dp[i] = smallestValue
return dp[n]

Runtime: We have two loops giving n loops of O(c) work as the calls to dp are constant time. Meaning we have a total of O(cn) work.

Proof: Once we arrive at some $dp[i]$ we have filled $dp[1..i-1]$ as we have looped through all previous i values from a bottom up initialization.

The recurrence itself tries to use all coins at the current number of cents, i. This represents all possible moves, and checks to see the minimum number of coins used at each call of the recurrence that make up a total of i cents.

It takes this the minimum between all moves and that value goes into dp[i]. This means that the move that uses the least number of coins to go from i cents to 0 cents will go into dp[i].

The minimum number of coins that add up n cents becomes the returned value of $dp[n][c]$ which is the overall solution.