Gabe McKay
Terisa Le
Reeve Masilamani

ECS 122A
October 23, 2025

Homework 2

Problem 1

Part 1

$dp[i][j] \doteq$ A boolean that represents whether it is possible to find coins with indexes less than j that add to equal to i cents where i is the current number of cents unassigned and j is the current coin index.
i ranges from $0..n$ and j ranges from $1..c$.

Recurrence:
$dp[i][j] = dp[i][j-1]$ or $dp[i-d_j][j-1]$ where d_j is the denomination of the j-th coin.

Initialization:
$dp[0][0] = true$, since if you have a combination of coins that get the sum to 0 cents you have found a solution.

```
for $i = 1$ to n:
  for $j = 1$ to c:
    b ← false
    if $i >= d_j$:
      $b = dp[i - d_j][j - 1]$
    $dp[i][j] = dp[i][j - 1]$ or $b$

return $dp[n][c]$
```

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 to fill the dp table, then a constant call to return the answer.

Proof: By the time you get to $dp[i][j]$ we have filled all previous $dp[1..i-1][1..j-1]$ as we used 2 loops to fill the table bottom up.

From here we can analyze the recurrence relation. The recurrence looks at two cases. One where you use the d_j coin and one where you don't. In the case where you use the d_j coin you subtract the value of d_j from your current amount of cents, i, as represented by $dp[i-d_j][j-1]$. The other case looks at skipping that coin and not changing the current i change, as represented by $dp[i][j-1]$.

If either case returns true then its possible to get to 0 cents with the remaining coins, otherwise it isn't.

The overall solution will be $dp[n][c]$.

Gabe McKay
Terisa Le
Reeve Masilamani

Homework 2

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 ← $\infty$
    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.