Here is a dynamic-programming algorithm for MAKING CHANGE.

MAKING CHANGE .....

input: A pair $I = (D, T)$ where $D = \{d_1, d_2, \ldots, d_k\}$ is a set of *denominations*, and T is a target (all non-negative integers, with $1 \in D$).

output: A minimum-size sequence $C = (c_1, \ldots, c_n)$ of *coins* s.t. $c_i \in D$ for all i and $\sum_i c_i = T$.

A previous note described a greedy algorithm, and proved that it worked for denominations $D = \{1, 5, 10, 25\}$. However, that algorithm doesn't work for all denominations. For example, for $D = \{1, 3, 4\}$ it fails for target $T = 6$.

Next we give a dynamic-programming algorithm that works in all cases. Fix any input instance $(D = \{d_1, d_2, \ldots, d_k\}, T)$.

subproblems. For each target $t \in \{0, 1, \ldots, T\}$, define $M[t]$ to be the minimum number of coins needed to make change totaling t using denominations in D. The solution to the given instance (D, T) is $M[T]$.

recurrence relation. $M[0] = 0$, and, for $t \in \{1, \ldots, T\}$

$$M[t] = \min\{1 + M[t - d] : d \in D, t - d \geq 0\}.$$

run time. There of the $T+1$ subproblems takes time $O(|D|)$ to evaluate. Total time is $O(|T| \cdot |D|)$.

correctness. See the end of this note for a long-form proof that the recurrence is correct. We did not do the proof in class.

Reducing MAKING CHANGE to SHORTEST PATHS. We can think of the above algorithm as solving an underlying SHORTEST PATH problem. Given an instance (D, T) of MAKING CHANGE, consider the following instance $(G = (V, E), s, t)$ of SHORTEST PATH.

Take $V = \{0, 1, \ldots, T\}$. That is, there is a vertex for each subproblem $M[t]$. For each t, add an edge from $t - d$ to t for each $d \in D$ with $t - d \geq 0$. So $E = \{(t - d, t) : t \in V, d \in D, t - d \geq 0\}$. (There is an edge $(t - d, t)$ if subproblem $t - d$ occurs in the right-hand side of the recurrence for subproblem t.) Give each edge weight 1.

Then each path p in G from 0 to T corresponds to a way of making change totaling T, and vice versa. Specifically, if a path p from 0 to T goes through a sequence of vertices such as $0, c_1, c_1 + c_2, \ldots, c_1 + c_2 + \cdots + c_n = T$, then it corresponds to making change totaling T using coins in $c = (c_1, c_2, \ldots, c_n)$. The length of the path p is the number of coins in c. So, the minimum number of coins needed to make change totaling T equals the distance $\mathsf{dist}(0, T)$ from 0 to T in G, and another algorithm for MAKING CHANGE would work as follows:

```
makingChange(D, T) (return the min. num. of coins needed to make change for T with denominations in D)
1. construct the weighted DAG G described above for (D, T)
2. return ShortestPath(G, 0, T)
```

This algorithm is essentially equivalent to the previous dynamic-programming algorithm for MAKING CHANGE. Using this idea, how could you solve the following problem? Given (D, T), compute the *number of ways* to make change totaling T using coins with denominations in D.

(There is an ambiguity here: do we consider two ways of making change different if they differ only in the order of their coins? Is $c = (1, 1, 3)$ different than $c' = (3, 1, 1)$? First answer the question assuming that order matters (so these c and c' are different). Then solve it assuming order doesn't matter. Hint: restrict to sequences c such that, say $c_i \geq c_{i+1}$ for each i.)

A long-form proof that the recurrence relation for MAKING CHANGE is correct. Given an instance (D, T) of MAKING CHANGE, recall that, for $t \in \{0, 1, \ldots, T\}$, $M[t]$ is defined to be the minimum number of coins needed to make change totaling t, using denominations in D. Recall the recurrence relation: $M[0] = 0$, and, for $t \in \{1, \ldots, T\}$

$$M[t] = \min\{1 + M[t - d] : d \in D, t - d \geq 0\}.$$

Lemma 1. *The recurrence relation is correct.*

Proof (long form).

1. $M[0] = 0$ because the empty sequence gives total 0.
2.1. Consider any $t \in \{1, 2, \ldots, T\}$.
2.2. First we show that $M[t]$ is *at least* the right-hand side of the recurrence.
2.3. Let $c = (c_1, c_2, \ldots, c_n)$ be an optimal way of making change for t, so $|c| = M[t]$.
2.4. Let $c' = (c_1, c_2, \ldots, c_{n-1})$ (the first $n - 1$ coins in c).
2.5. There are $M[t] - 1$ coins in c', with total value $t - c_n$.
2.6. So it is possible to use $M[t] - 1$ coins to make change with value $t - c_n$.
2.7. $M[t - c_n]$ is the minimum number of coins needed to make change with value $t - c_n$.
2.8. So $M[t] - 1 \geq M[t - c_n]$.
2.9. That is, $M[t] \geq 1 + M[t - c_n]$.
2.10. Since $c_n \in D$ and $t - c_n \geq 0$, it follows that $M[t] \geq \min\{1 + M[t - d] : d \in D, t - d \geq 0\}$.
2.11. That is, $M[t]$ is at least the right-hand side of the recurrence.
2.12. Next we show that $M[t]$ is *at most* the right-hand side of the recurrence.
2.13.1. Consider any $d \in D$ with $t - d \geq 0$. We will show $M[t] \leq 1 + M[t - d]$.
2.13.2. Let $c' = (c'_1, c'_2, \ldots, c'_m)$ be an optimal way of making change for $t - d$.
2.13.3. So $|c'| = M[t - d]$, and the total value of coins in c' is $t - d$.
2.13.4. Let $c = (c'_1, c'_2, \ldots, c'_m, d)$ be obtained by adding a coin of denomination d to c'.
2.13.5. There are $1 + M[t - d]$ coins in c, with total value $t - d + d = t$.
2.13.6. So it is possible to use $M[t - d] + 1$ coins to make change with value t.
2.13.7. $M[t]$ is the minimum number of coins needed to make change with value t.
2.13.8. So $M[t] \leq 1 + M[t - d]$.
2.14. By Block 2.13, for each $d \in D$ with $t - d \geq 0$, we have $M[t] \leq 1 + M[t - d]$.
2.15. That is, $M[t] \leq \min\{1 + M[t - d] : d \in D, t - d \geq 0\}$.
2.16. That is, $M[t]$ is at most the right-hand side of the recurrence.
2.17. By this and Step 2.11, $M[t]$ equals the right-hand of the recurrence.
3. By Block 2, for each $t \in \{1, \ldots, T\}$, the recurrence holds. □