Important Concepts

In order for a problem to admit a greedy algorithm, it needs to satisfy two properties.

Optimal Substructure: an optimal solution of an instance of the problem contains within itself an optimal solution to a smaller subproblem (or subproblems).

Greedy-choice Property: There is always an optimal solution that makes a greedy choice.

Solutions

16-1: Coin Changing

16-1a.

Coin change using US currency

Input: n - a positive integer.

Output: minimum number of quarters, dimes, nickels, and pennies to make change for n. We assume that we have an infinite supply of coins of each denomination.

Consider the following natural greedy strategy:
Greedy strategy: To make change for n find a coin of maximum possible value $\leq n$, include it in your solution, continue recursively to solve the subproblem of making change for n minus the value of the coin selected.
If we implement the above strategy naively then the runtime would be $\Theta(n)$. Observe that the above strategy will keep including quarters until the value of the subproblem drops below 25. It will then proceed onto dimes, nickels, and pennies. Thus, the above greedy strategy can be implemented in $O(1)$ time if we use the division and the modulus operations and assume that all arithmetic operations take $O(1)$ time. In other words, the strategy becomes:
Greedy strategy adapted to run in $O(1)$ time for the US currency:

- use $a_1 = \lfloor n/25 \rfloor$ quarters, let $n_q = n \mod 25$ be the remaining number of cents;
- use $a_2 = \lfloor n_q \mod 10 \rfloor$ dimes, let $n_d = n_q \mod 10$ be the remaining number of cents;
- use $a_3 = \lfloor n_d/5 \rfloor$ nickels, let $n_k = n_d \mod 5$ be the remaining number of cents;
- use $a_4 = n_k$ pennies.

Claim 0.1. *Let n be the input. The greedy strategy above constructs a solution (a_1, a_2, a_3, a_4). Let $S_i = (a_1, \ldots, a_i)$. Then for all $i \in \{0, 1, 2, 3, 4\}$ we can extend S_i to an optimal solution using only denominations following the ith one.*

Proof. It is clear that we can extend $S_0 = \emptyset$ to an optimal solution using all other denominations. Normally we would prove the claim by induction on i, but we only need to consider finitely many values of i, so the rest of the proof is given by the following case analysis:

Step 1: Let (a'_1, a'_2, a'_3, a'_4) be an optimal solution given by the above base case of S_0. Note that due to the choice of our algorithm $a'_1 \leq a_1$. If $a_1 = a'_1$, we are done with this case, and can move on to the next case. Otherwise, we have $a'_1 < a_1$, i.e., a suggested optimal solution uses less quarters than our solution. Note than we have $a_1 > 0$, so $n \geq 25$. Consider the following subcases:

If $a'_3 >= 3$, we can always replace 3 dimes with a quarter and a nickel to obtain a more optimal solution $\rightarrow$ contradiction

If $a'_3 <= 2$, similar to **Step 2** below, we can break it down to prove that we can always replace some combinations of nickels, pennies and dimes by a quarter.

Thus, we must have that our partial solution $S_1 = (a_1)$ is extendable to some optimal solution (a_1, a'_2, a'_3, a'_4) using only dimes, nickels, and pennies.

Step 2: Let (a_1, a'_2, a'_3, a'_4) be an optimal solution guaranteed by the previous step. Again, we have $a'_2 \leq a_2$ by the way that our algorithm chooses a_2. If $a_2 = a'_2$, there is nothing to prove in this step, and we can move on to the next one. We need to show that $a'_2 < a_2$ is impossible. Let's argue by contradiction. Note that we can only have $a_2 \in \{0, 1, 2\}$. Assume that $a'_2 < a_2$. Then consider the following subcases:

When $a'_2 = 1$, $a'_1 \geq 5$, so we can replace 1 nickel and 5 pennies by 1 dime $\rightarrow$ contradicting the assumption of optimality.

When $a'_2 = 0$, $a'_1 = n \geq 10$, so we can replace 10 pennies with 1 dime $\rightarrow$ again contradicting the assumption of optimality.

At this step, we have that the solution produced by the algorithm has to agree with some optimal in the first two choices, i.e., there is an optimal solution of the form (a_1, a_2, a'_3, a'_4).

Step 3: Let (a_1, a_2, a'_3, a'_4) be an optimal solution obtained from **Step 2**. By the way our algorithm chooses a_3 we have $a'_3 \leq a_3$. If $a'_3 = a_3$ we are done. Otherwise we have $0 = a'_3 < a_3 = 1$. If $a'_3 = 0$, $a'_1 \geq 5$ so that we can replace 5 pennies with 1 nickel to get a better solution $\rightarrow$ contradicting optimality.

Note that once a_1, a_2, a_3 are fixed, a_4 becomes determined. We have shown that there is an optimal solution that agrees with the solution produced by our algorithm on a_1, a_2, a_3 therefore it has to agree with our solution on a_4, as well. This shows that the solution found by the algorithm is, in fact, optimal. □

16-1b.

Coin change using denominations that are powers of a fixed constant

Input: $c > 1, k \geq 1, n \geq 1$ - integers.

Output: minimum number of coins needed to make change for n. The denominations of coins are allowed to be $c^0, c^1, \ldots, c^k$. We assume that we have an infinite supply of coins of each denomination.

Consider the same greedy strategy as the one presented in the previous part:
Greedy strategy: To make change for n find a coin of maximum possible value $\leq n$, include it in your solution, continue recursively to solve the subproblem of making change for n minus the value of the coin selected.
The naive implementation of the above strategy requires runs in time $\Theta(kn)$, which is not even polynomial in the size of the input. As before, the implementation can be improved to run in time $\Theta(k)$ (left as an exercise for the reader) using the division and the modulus operations, assuming that all the arithmetic operations run in constant time.

Claim 0.2. *Let a_i denote the number of coins of denomination c^i (where $i \in \{0, 1, \ldots, k\}$) used by the greedy algorithm. We claim that there is a unique optimal solution that has to agree with the greedy solution.*

Proof. Let $(a'_0, a'_1, \ldots, a'_k)$ denote an optimal solution. First note that $a'_i < c$ for all $i < k$. This is simply because if we had $a'_i \geq c$ for some $i < k$, we could replace c coins of denomination c^i by a single coin of denomination c^{i+1}, improving upon an optimal - contradiction.

Next, we prove that $a'_i = a_i$ for all i. Assume for contradiction that this is not the case. Let j be the largest possible index such that $a'_j \neq a_j$. Note that by the way that our algorithm picks values a_i we have $a'_j < a_j$. Let $N = \sum_{i=0}^{j} a_i c^i = \sum_{i=0}^{j} a'_i c^i$ (the equality holds because the two solutions agree on $\{a_{j+1}, a_{j+2}, \ldots, a_k\}$ by assumption). Thus

$$\begin{aligned} N = \sum_{i=0}^{j-1} a'_i c^i + a'_j c^j &\leq \sum_{i=0}^{j-1} (c-1)c^i + a'_j c^j \\ &= (c-1)\frac{c^j - 1}{c-1} + a'_j c^j \\ &= c^j - 1 + a'_j c^j \\ &= c^j(a'_j + 1) - 1 \\ &\leq a_j c^j - 1 \\ &\leq \sum_{i=0}^{j} a_j c^j - 1 \\ &= N - 1. \end{aligned}$$

Contradiction. □

Note that the above proof technique is not the standard proof technique for greedy algorithms. The standard proof technique uses the loop invariant "partial solution can always be extended to some optimal solution." The above proof is shorter and simpler, and this is possible because the optimal solution is unique in this problem, which is usually not the case for optimization problems. Thus, the above proof technique belongs to the category of "ad-hoc" proofs.

16-1c. Suppose that we have only coins of denominations $1, 10, 25$ and we need to make change for 30. Then the greedy solution would use one quarter and five pennies - 6 coins overall. A non-greedy solution that uses 3 dimes is strictly better.