Session 33: Optimization Algorithms

- Greedy Algorithms
- Cashier’s Algorithm

Optimization Problems

Optimization problems minimize or maximize some parameter over all possible inputs

Examples

- Finding a route between two cities with the smallest total mileage.
- Determining how to encode messages using the fewest possible bits.
- Finding the fiber links between network nodes using the least amount of fiber.

Greedy Algorithms

Optimization problems can often be solved using a *greedy algorithm*, which makes the “best” choice at each step.

- Making the “best choice” at each step does not necessarily produce an optimal solution to the overall problem
  - but in many instances, it does.
- After specifying the greedy algorithm,
  - Either we prove that this approach always produces an optimal solution
  - or we find a counterexample to show that it does not.

Cashier’s Algorithm

Task: Find for an amount of any n cents the least total number of coins using the following coins: quarters (25 cents), dimes (10 cents), nickels (5 cents), and pennies (1 cent).

(Greedy) Idea: At each step choose the coin with the largest possible value that does not exceed the amount left.

Example

Change for $n = 67$ cents.

Cashier's Algorithm

procedure *change*(c_1, c_2, …, c_r: values of coins, where $c_1 > c_2 > \ldots > c_r$; n: a positive integer)

for $i := 1$ **to** r *{start with the largest coins}*

  $d_i := 0$ *{d_i counts the coins of denomination c_i}*

  while $n \geq c_i$

    $d_i := d_i + 1$ *{add a coin of denomination c_i}*

    $n = n - c_i$ *{remove the value of the coin}*

return d_1, d_2, …, d_r

Cashier's Algorithm

The algorithm works with any coin denominations $c_1, c_2, \ldots, c_r$

procedure *change*($c_1, c_2, \ldots, c_r$: values of coins, where $c_1 > c_2 > \ldots > c_r$; n: a positive integer)

for $i := 1$ **to** r *{start with the largest coins}*

  $d_i := 0$ *{d_i counts the coins of denomination c_i}*

  while $n \geq c_i$

    $d_i := d_i + 1$ *{add a coin of denomination c_i}*

    $n = n - c_i$ *{remove the value of the coin}*

return $d_1, d_2, \ldots, d_r$

For the example of U.S. currency, we have quarters, dimes, nickels and pennies, with $c_1 = 25$, $c_2 = 10$, $c_3 = 5$, and $c_4 = 1$.

Proving Optimality

Show that the change making algorithm for *U.S.* coins is optimal.

Lemma 1: If *n* is a positive integer, then *n* cents in change using quarters, dimes, nickels, and pennies, using the fewest coins possible

1. has at most 2 dimes, 1 nickel, 4 pennies
2. cannot have 2 dimes and 1 nickel
3. the total amount of change in dimes, nickels, and pennies cannot exceed 24 cents.

Proof of Lemma 1

Proof:

Property 1: By contradiction (not the fewest coins)

- If we had 3 dimes, we could replace them with a quarter and a nickel.
- If we had 2 nickels, we could replace them with 1 dime.
- If we had 5 pennies, we could replace them with a nickel.

Property 2: By contradiction (not the fewest coins)

- If we had 2 dimes and 1 nickel, we could replace them with a quarter.

Property 3: is a consequence

- The largest allowable combination, 2 dimes and 4 pennies, has a maximum value of 24 cents.

Proving Optimality

Theorem: The greedy change-making algorithm for U.S. coins produces change using the fewest coins possible.

Proof: By contradiction.

1. Assume there is a positive integer *n* such that change can be made with a fewer total number of coins than given by the algorithm.
2. Let *q'* be the number of quarters used in this optimal solution
   1. *q' ≤ q , since the greedy algorithm uses the maximum number of quarters possible*
   2. *q' < q is not possible* since then we would have more than 25 cents in dimes, nickels, and pennies, which contradicts Lemma 1.
3. Therefore *q' = q*
4. Similarly the greedy algorithm chose the maximum possible number of dimes
   1. *we cannot replace a dime by at most 1 nickel and 4 pennies of value 9.*
5. Similarly the greedy algorithm chose the maximum number of nickels, and therefore also the number of nickels and pennies is the same.

Cashier's Algorithm Discussion

Optimality depends on the denominations available.

If we allow only quarters (25 cents), dimes (10 cents), and pennies (1 cent), the algorithm no longer produces the minimum number of coins.

Counterexample: Consider the example of 31 cents.

- The optimal number of coins is 4, i.e., 3 dimes and 1 penny.
- Result of algorithm:

Summary

- Greedy Algorithms
- Cashier’s Algorithm
- Optimality Proof