Extra Slides

Coin Changing

Greed is good. Greed is right. Greed works. Greed clarifies, cuts through, and captures the essence of the evolutionary spirit.
- Gordon Gecko (Michael Douglas)

Coin Changing

Goal. Given currency denominations: 1, 5, 10, 25, 100, devise a method to pay amount to customer using fewest number of coins.

Ex: 34¢.

Cashier’s algorithm. At each iteration, add coin of the largest value that does not take us past the amount to be paid.

Ex: $2.89.

Cashier’s algorithm. At each iteration, add coin of the largest value that does not take us past the amount to be paid.

Sort coins denominations by value: $c_1 < c_2 < \ldots < c_n$.

\begin{verbatim}
coins selected
S ← ∅
while \(x \neq 0\) do
    let \(k\) be largest integer such that \(c_k \leq x\)
    if \((k = 0)\) then
        return "no solution found"
    x ← x - \(c_k\)
    S ← S \cup (k)
return S
\end{verbatim}

Q. Is cashier’s algorithm optimal?
**Coin-Changing: Analysis of Greedy Algorithm**

**Theorem.** Greed is optimal for U.S. coinage: 1, 5, 10, 25, 100.

**Proof.** (by induction on $x$)
- Consider optimal way to change $c_i \leq x < c_{i+1}$: greedy takes coin $k$.
- We claim that any optimal solution must also take coin $k$.
  - if not, it needs enough coins of type $c_i$, ..., $c_{k-1}$ to add up to $x$
  - table below indicates no optimal solution can do this
- Problem reduces to coin-changing $x$ - $c_i$ cents, which, by induction, is optimally solved by greedy algorithm.

<table>
<thead>
<tr>
<th>$k$</th>
<th>$c_k$</th>
<th>All optimal solutions must satisfy</th>
<th>Max value of coins</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>$p \leq 4$</td>
<td>1, 2, ..., $k-1$</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>$n \leq 1$</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>$n + d \leq 2$</td>
<td>4 + 5 = 9</td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td>$q \leq 3$</td>
<td>20 + 4 = 24</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>no limit</td>
<td>75 + 24 = 99</td>
</tr>
</tbody>
</table>

**Observation.** Greedy algorithm is sub-optimal for US postal denominations: 1, 10, 21, 34, 70, 100, 350, 1225, 1500.

**Counterexample.** 140¢.
- Greedy: 100, 34, 1, 1, 1, 1, 1.
- Optimal: 70, 70.

**Selecting Breakpoints**

**Selecting breakpoints.**
- Road trip from Princeton to Palo Alto along fixed route.
- Refueling stations at certain points along the way.
- Fuel capacity = C.
- Goal: makes as few refueling stops as possible.

**Greedy algorithm.** Go as far as you can before refueling.
Selecting Breakpoints: Greedy Algorithm

Truck driver's algorithm.

Sort breakpoints so that: \( b_0 < b_1 < b_2 < \ldots < b_n = L \)

\[
\begin{align*}
S & \leftarrow \{ b_0 \} \quad \text{--- breakpoints selected} \\
x & \leftarrow 0 \quad \text{--- current location} \\
\text{while } (x \neq b_n) & \quad \text{--- continue while } x \neq b_n \\
& \quad \text{let } p \text{ be largest integer such that } b_p \leq x + C \\
& \quad \text{if } (b_p = x) \\
& \quad \quad \text{return } "\text{no solution}" \\
& \quad \quad x \leftarrow b_p \\
& \quad S \leftarrow S \cup \{ p \} \\
\text{return } S
\end{align*}
\]

Implementation. \( O(n \log n) \)
- Use binary search to select each breakpoint \( p \).

Selecting Breakpoints: Correctness

**Theorem.** Greedy algorithm is optimal.

**Pf.** (by contradiction)
- Assume greedy is not optimal, and let’s see what happens.
- Let \( 0 = g_0 < g_1 < \ldots < g_r = L \) denote set of breakpoints chosen by greedy.
- Let \( 0 = f_0 < f_1 < \ldots < f_r = L \) denote set of breakpoints in an optimal solution with \( f_i = g_i, f_1 = g_1, \ldots, f_r = g_r \) for largest possible value of \( r \).
- Note: \( g_{i+1} > f_i \) by greedy choice of algorithm.

Greedy: 

<table>
<thead>
<tr>
<th>( g_0 )</th>
<th>( g_1 )</th>
<th>( g_2 )</th>
<th>( g_3 )</th>
<th>( g_{i+1} )</th>
</tr>
</thead>
</table>

OPT: 

<table>
<thead>
<tr>
<th>( f_0 )</th>
<th>( f_1 )</th>
<th>( f_2 )</th>
<th>( f_3 )</th>
<th>( f_{i+1} )</th>
</tr>
</thead>
</table>

Edsger W. Dijkstra

The question of whether computers can think is like the question of whether submarines can swim.

Do only what only you can do.

In their capacity as a tool, computers will be but a ripple on the surface of our culture. In their capacity as intellectual challenge, they are without precedent in the cultural history of mankind.

The use of COBOL cripes the mind; its teaching should, therefore, be regarded as a criminal offence.

APL is a mistake, carried through to perfection. It is the language of the future for the programming techniques of the past; it creates a new generation of coding bums.