Greedy Algorithms

CLRS Reading: Sections 16.1, 16.2, pages 414 -- 428

Making Change
**Make-Change (The Algorithm)**

Greedy-Make-Change(n)  
\[
\text{denominations} = \{25, 10, 5, 1\} \\
\text{coins} = \emptyset \\
\text{value} = 0 \\
\text{while not value} = n \\
\quad x = \text{largest } x \in \text{denominations} \text{ s.t. value} + x \leq n \\
\quad \text{coins} = \text{coins} \cup \{x\} \\
\quad \text{value} = \text{value} + x \\
\text{return coins}
\]

---

**Generic Greedy Algorithm**

Greedy(C)  
\[
\text{S} = \emptyset \\
\text{while not solution(S) and } C \neq \emptyset \\
\quad x = \text{greedy-choice}(C) \\
\quad C = C \setminus \{x\} \\
\quad \text{if feasible}(S \cup \{x\}) \\
\quad \quad S = S \cup \{x\} \\
\text{return } S
\]

---
Suppose We Knock Out the Nickle

Greedy-Make-Change(n)
  denominations = {25,10,1}
  coins = Ø
  value = 0
  while not value = n
    x = largest x in denominations s.t. value+x ≤ n
    coins = coins ∪ {x}
    value = value + x
  return coins

Pluses and Minuses to a Greedy Approach

• **Finding** a greedy procedure to a problem is generally easy.

• **Proving** the greedy procedure works is usually somewhat trickier.
**Activity-Selection Problem**

- Suppose we have a set $S = \{1, 2, 3, ..., n\}$ of $n$ activities to use a resource. Each activity $i$ has a start time $s_i$ and a finish time $f_i$.

**Order the Activities by Increasing Finish and Act Greedy**
Generic Greedy Algorithm

Greedy(C) {C is the set of choice candidate}
S = ∅ {S the solution set}
while not solution(S) and C ≠ ∅
  x = greedy-choice(C)
  C = C \ {x}
  if feasible(S ∪ {x})
    S = S ∪ {x}
return S

Greedy Code

Greedy-Activity-Selector(s,f)
s = s.length
A = {1}
j = 1
for i = 1 to n {Greedy choice}
  if s_i ≥ f_j {Feasibility test}
    A = A ∪ {i}
    j = i
return A
Proof of Correctness

**Theorem 16.1**
Algorithm Greedy-Activity-Selector produces solutions of maximum size for the activity-selection problem.

Elements of Greedy Strategy

*Greedy-choice property*
A globally optimal solution can be arrived at by making a locally optimal (greedy) choice.

*Optimal substructure*
An optimal solution to the problem contains within it optimal solutions to the subproblems.
• The *greedy-choice property* for coin changing depends upon denominations.

• It holds for denominations = \{25, 10, 5, 1\} while it fails for denominations = \{25, 10, 1\}.

• Fortunately, if the greedy-choice property fails for coin changing, it fails for a value near, but just exceeding the value of one of the coins.