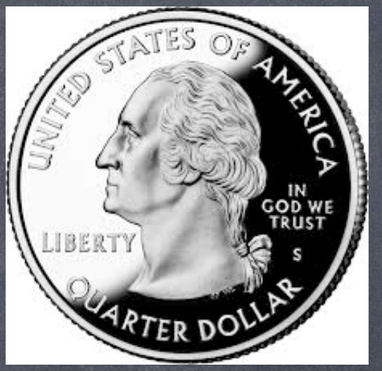
Giving Change

An Example of a Greedy Algorithm

Andreas Klappenecker

Giving Change

- Suppose that you buy an item that costs $1.52
- The store allows cash transactions only, so you give $2.00. You expect $0.48 of change.
- How does the cashier select from the coin values { $0.01, $0.05, $0.10, and $0.25 } the correct amount of change?



Coin Changing Algorithm

Suppose we have n types of coins with values

v[1] > v[2] > … > v[n] > 0

Given an amount C >=0, the following algorithm tries to give change for C:

```
m[1] := 0; m[2]:=0; … ; m[n] := 0; // multiplicities of coins
for i = 1 to n do
   while C >= v[i] do
     C := C - v[i]; m[i]++;
   od;
```

The Algorithm in Action

Suppose that v[1] = 5; v[2] =2; v[3] = 1;

Let C = 14.

Then the algorithm calculates

m[1] = 2 and m[2] = 2

This gives the correct amount of change:

C = v[1]*m[1]+v[2]*m[2] = 5*2+2*2 = 14

Correctness

Suppose we have n types of coins with values

v[1] > v[2] > … > v[n] > 0

Input: A positive integer C

```
m[1] := 0; m[2]:=0; … ; m[n] := 0; // multiplicities of coins
for i = 1 to n do
    while C >= v[i] do
      C := C - v[i]; m[i]++;
    od;
```

Does the algorithm return the correct amount of change?

Optimality

Suppose we have n types of coins with values

v[1] > v[2] > … > v[n] = 1

Input: A positive integer C

```
m[1] := 0; m[2]:=0; … ; m[n] := 0; // multiplicities of coins
for i = 1 to n do
   while C >= v[i] do
      C := C - v[i]; m[i]++;
   od;
```

Does the algorithm return the least number of coins?

Optimality Example

v[1] = 5; v[2] = 2; v[3] = 1;

If C<2, then the algorithm gives a single coin, which is optimal.

If C<5, then the algorithm gives at most 2 coins:

C = 4 = 2*2 // 2 coins

C = 3 = 2+1 // 2 coins

C = 2 = 2 // 1 coins

In each case this is optimal.

If C >= 5, then the algorithm uses the most coins of value 5 and then gives an optimal change for the remaining value < 5. One cannot improve upon this solution. Let us see why.

Optimality Example

Any optimal solution to give change for C with

C = v[1]*m[1] + v[2]*m[2] + v[3] * m[3]

with v[1] = 5; v[2] = 2; v[3] = 1 must satisfy

a) m[3] <= 1 // if m[3] >= 2, replace two 1's with 2.

b) 2*m[2] + m[1] < 5 // otherwise use a 5

c) If C >= 5*k > 0, then m[1] >= k // otherwise solution violates b)

Thus, any optimal solution greedily chooses maximal number of 5s! The remaining value in the optimal value has to choose maximal number of 2s, so any optimal solution is the greedy solution!

Another Example

Let v[1]=4; v[2]=3; v[3]=1;

The algorithm yields 3 coins of change for

C = 6 namely 6=4+1+1

Is this optimal?

Another Example

Any optimal solution to give change for C with

C = v[1]*m[1] + v[2]*m[2] + v[3] * m[3]

with v[1] = 4; v[2] = 3; v[3] = 1 must satisfy

a) m[3] <= 2 // if m[3] > 2, replace three 1's with a 3.

b) We cannot have both m[3]>0 and m[2]>0, for otherwise we could use a 4 to reduce the coin count.

c) m[2]<4 // otherwise use 4 to reduce number of coins

Greedy will fail in general. One can still find an optimal solution efficiently, but the solution is not unique. Why? 9=3+3+3=4+4+1

How Likely is it Optimal?

Let N be a positive integer.

Let us choose integer a and b uniformly at random subject to N>b>a>1. Then the greedy coin-changing algorithm with

v[1]=b; v[2]=a; v[3]=1

gives always optimal change with probability

$8/3\ N^{-1/2} + O(1/N)$

as Thane Plambeck has shown [AMM 96(4), April 1989, pg 357].

Greedy Algorithms

Greedy-Choice Property

The greedy choice property is that a globally optimal solution can be arrived at by making a locally optimal (=greedy) choice.

Optimal Substructure

A problem exhibits optimal substructure if and only if an optimal solution to the problem contains within it optimal solutions to subproblems.

Greedy Algorithms

The development of a greedy algorithm can be separated into the following steps:

a) Cast the optimization problem as one in which we make a locally optimal choice and are left with one subproblem to solve.

b) Prove that there is always an optimal solution to the original problem that makes the greedy choice, so that the greedy choice is always safe.

c) Demonstrate that, having made the greedy choice, what remains is a subproblem with the property that if we combine an optimal solution to the subproblem with the greedy choice that we have made, we arrive at an optimal solution to the original problem.