A Polynomial-time Algorithm for the Change-Making Problem

David Pearson
Computer Science Department
Cornell University
Ithaca, New York 14853, USA
pearson@cs.cornell.edu

June 14, 1994

Abstract

The *change-making problem* is the problem of representing a given value with the fewest coins possible from a given set of coin denominations. To solve this problem for arbitrary coin systems is *NP*-hard [L]. We investigate the problem of determining whether the greedy algorithm always produces the optimal result for a given coin system. Chang and Gill [CG] show that this can be solved in time polynomial in the size of the largest coin and in the number of coins. Kozen and Zaks [KZ] give a more efficient algorithm, and pose as an open problem whether there is an algorithm to solve this problem which is polynomial in the size of the input.

In this paper, we will derive such an algorithm. We first obtain a characterization of the smallest counterexample (if there is one) for which the greedy algorithm is not optimal. We then derive a set of $O(n^2)$ possible values (where n is the number of coins) which must contain the smallest counterexample. Each can be tested with $O(n)$ arithmetic operations, giving us an $O(n^3)$ algorithm.

1 Introduction

In the *change-making problem,* we are given a finite set of coin denominations and an unlimited supply of coins in each denomination. We want to represent a given value with the fewest coins possible. The problem of determining the optimal representation in general is *NP*-hard [GJ, KZ]. However, in almost all coin systems in use in the world, the problem can be efficiently solved by the *greedy algorithm.* This algorithm repeatedly takes the largest coin whose value is no larger than the remaining amount. For general systems, the greedy algorithm is not always optimal. With the coin system 1,3,4, for example, the value 6 will be represented by the greedy algorithm as 4+1+1, but the optimal representation, 3+3, uses one coin fewer.

The English system in use before 1971 (when decimal currency was introduced) was an example of a "natural" coin system for which the greedy algorithm was not optimal. The coin denominations were (in units of a penny) $\frac{1}{2}$, 1, 3, 6, 12 (shilling), 24 (florin), 30 (half-crown), 60 (crown), and 240 (pound). 48 pence would be represented as 30+12+6 by the greedy algorithm, while the minimal representation is 24+24 (two florins). The picture was further complicated by an obsolete coin, the Guinea, with a value of 21 shillings.

In this paper, we will investigate the problem of determining, for a given coin system, whether the greedy algorithm always yields the optimal representation.

Definition 1 *A* coin system *is a finite set of positive integers, $c_1 > c_2 > \cdots > c_n$. We will assume that the smallest coin, c_n, has value 1, since otherwise not all values would be representable. Let C be the n-vector $(c_1, c_2, \ldots, c_{n-1}, c_n)$. A* representation *in C of a non-negative integer x is an n-vector V of non-negative integers such that $V \cdot C = x$, where $V \cdot C$ denotes the inner product of V with C, $\sum_{i=1}^{n} v_i c_i$. The* size $|V|$ *of a representation V is the number of coins in it, i.e. $V \cdot (1, 1, \ldots, 1)$.*

Note that we write the n-vectors with the higher coin denominations first. The leading terms of the representation vectors are then the most significant terms, as they are in positional number systems, or car odometers. The greedy algorithm has a particularly simple relationship with the lexicographic ordering on the representation vectors.

Definition 2 *For n-vectors $U = (u_1, u_2, \ldots, u_n)$ and $V = (v_1, v_2, \ldots, v_n)$, write $U < V$ if U lexicographically precedes V, i.e. if for some i: $1 \leq i \leq n$, $u_i < v_i$ and for all j: $1 \leq j < i$, $u_j = v_j$. The* greedy representation *of x, denoted by $G(x)$, is the lexicographically greatest of all the representations of x.*

Notice that if $x < y$, then $U = G(x) + (0, 0, \ldots, 0, y - x)$ is a representation for y. Clearly $G(x) < U$. By the definition of $G(y)$, we know that $U \leq G(y)$, so $G(x) < G(y)$. The greedy representation, in other words, has the nice property of preserving order.

For a given value x, there may be several representations with the same size, including the minimum size. We would like to choose a unique representative of all the minimum-size representations.

Definition 3 *The* minimal *representation of x, denoted by $M(x)$, is the lexicographically greatest of all the representations of x of minimum size. We call the coin system C* canonical *if $G(x) = M(x)$ for all x.*

It is not immediately clear whether there is a finite process to show that a coin system is canonical, since there are infinitely many values x to test. However, Chang and Gill [CG] show that if no counterexample exists for which $G(x) \neq M(x)$ in a certain finite set, then no counterexample exists. The size of this set is polynomial in the value of the largest coin c_1, in the worst case $O({c_1}^3)$. Kozen and Zaks [KZ] show that the smallest counterexample w (if there is one) must lie in the range $c_{n-2} \leq w < c_1 + c_2$. This gives a set of candidates whose size is linear in c_1, and they give an $O(nc_1)$ algorithm for testing whether the system is canonical. This time bound is still potentially exponential in the size of the input, however.

They give another algorithm which takes time polynomial in $\log(c_1)$ for any fixed number of coins, but exponential in the number of coins. They leave it as an open problem whether an algorithm exists which is polynomial in the size of the input.

It is *NP*-hard to decide, for an arbitrary coin system, whether the greedy algorithm is optimal for a specific value x [KZ]. Answering the same question for all x seems intuitively harder, but is actually easier. The smallest counterexample is much easier to test than an arbitrary one, and it has a particular structure which will make it easy to find.

In this paper, we obtain a set of possible counterexamples which is polynomial in the number of coins in the system and independent of the sizes of the coins. This will lead to a true polynomial-time algorithm.

2 A Polynomial-time algorithm

In this section, we will characterize the smallest counterexample, if there is one, and derive a small ($O(n^2)$) set of candidates that must include the smallest. As a preliminary, we first need to establish some facts about minimal and greedy representations.

For two vectors U and V, we will say that $U \subseteq V$ if each element u_i of U is less than or equal to the corresponding element v_i of V. Equivalently, $V = U + D$ for some vector D of non-negative integers. If V is a representation of x, then U is a representation of the number $x - D \cdot C \leq x$. It is a surprising fact that if V is a greedy or minimal representation, then U is also.

Lemma 1 *Call U* greedy *if $U = G(U \cdot C)$, and* minimal *if $U = M(U \cdot C)$. Then*

(a) if $U \subseteq V$ and V is greedy, then U is greedy

(b) if $U \subseteq V$ and V is minimal, then U is minimal

Proof. For (a), note that vector addition preserves lexicographic order:

$$A \leq B \Longleftrightarrow A + D \leq B + D$$

Let U' be any representation of $U \cdot C$. Then

$$\begin{array}{rcll} U' \cdot C & = & U \cdot C & \\ (V - U + U') \cdot C & = & V \cdot C & \text{by linearity} \\ V - U + U' & \leq & V & \text{since } V \text{ greedy (and } V - U \text{ non-negative)} \\ U' & \leq & U & \text{since addition is order-preserving} \end{array}$$

Thus U is greedy, since it is the lexicographically greatest representation of $U \cdot C$.

For (b), define $A \sqsubseteq B$ if $|A| > |B|$ or ($|A| = |B|$ and $A \leq B$). Then the minimal representation is the *greatest* under $\sqsubseteq$. Addition is again order-preserving with respect to $\sqsubseteq$. The above proof is valid if we substitute $\sqsubseteq$ for $\leq$. □

Now suppose that C is not canonical. Let w be the smallest counterexample, i.e. the smallest element such that $G(w) \neq M(w)$. We will characterize w, and show that it must be one of $O(n^2)$ candidates. The fact that w is the smallest counterexample makes it easy to compute $M(w)$, giving us a polynomial-time test.

One important fact about w is that the sets of nonzero entries in $G(w)$ and $M(w)$ are disjoint. If this were not true, if both were nonzero in position k, for example, then we could subtract 1 from the kth position in both $G(w)$ and $M(w)$ to get two new vectors, which would be the greedy and minimal representations of $w - c_k$, by Lemma 1, contradicting the minimality of w.

Let $M(w) = (m_1, m_2, \ldots, m_n)$, and let i and j denote the first and last nonzero entries in $M(w)$ respectively, i.e. $m_i \neq 0$ and $m_j \neq 0$, and $m_k = 0$ for $1 \leq k < i$ and $j < k \leq n$. Note that $M(w) < G(w)$ by the definition of G, but since they do not share any nonzero entries, $G(w)$ must have a zero in position i and be nonzero in some earlier position. This means in particular that $i > 1$.

We can now characterize $M(w)$ in a precise way. The following theorem shows a close similarity between $M(w)$ and $G(c_{i-1} - 1)$.

Theorem 1 *$M(w)$ agrees with $G(c_{i-1} - 1)$ in entries 1 through $j - 1$, and is one greater in entry j. The remaining entries $m_{j+1}, \ldots, m_n$ are all zero.*

Proof. Recall that $G(w)$ must have a zero in position i and a nonzero entry in some earlier position. This means that $w \geq c_{i-1}$. On the other hand, subtracting one from the jth entry in $M(w)$ results in a valid minimal representation for $w - c_j$ (by Lemma 1) which must also be the greedy representation, since there are no smaller counterexamples. This representation has no coin larger than c_i, so $w - c_j < c_{i-1}$. Thus we get the following bounds:

$$w - c_j < c_{i-1} \leq w \tag{1}$$

Let $V = (v_1, v_2, \ldots, v_n) = G(c_{i-1} - 1)$. Since $c_{i-1} - 1 \geq c_i$, $v_i \neq 0$. This means we can reduce both v_i and m_i by one to get new valid greedy representations $G(c_{i-1} - 1 - c_i)$ and $G(w - c_i)$. We know from (1) that $c_{i-1} - 1 - c_i < w - c_i$, so $G(c_{i-1} - 1 - c_i) < G(w - c_i)$. Adding back the one in position i won't change their lexicographic ordering, so

$$V < M(w) \tag{2}$$

On the other hand, if we decrease m_j by one, we will get a valid greedy representation $G(w - c_j)$. Since $w - c_j \leq c_{i-1} - 1$ by (1), we have $G(w - c_j) \leq V$. Combining this with (2), $G(w - c_j) \leq V < M(w)$. Notice that $G(w - c_j)$ and $M(w)$ differ only in entry j, so if V differed from them in any position before j, it could not lie between them.

V and $M(w)$ therefore agree in positions $1, 2, \ldots, j - 1$, but $V < M(w)$, so we must have $v_k < m_k$ for some $k \geq j$. But since $m_{j+1}, m_{j+2}, \ldots, m_n$ are all zero, this could only be true for entry j: $v_j < m_j$. But $G(w - c_j) \leq V$, so $m_j - 1 \leq v_j$. The only possible value for m_j is thus $v_j + 1$. $\square$

In a fixed coin system, if we know i and j, Theorem 1 allows us to determine w exactly. There are only $O(n^2)$ possible values of i and j to test, so we have obtained our polynomial-sized sample set.

Notice that Theorem 1 gives us both the candidate w and the corresponding $M(w)$ (which need be valid *only* if w is actually the smallest counterexample). To test each candidate in the set requires generating this potential $M(w)$ and the true $G(w)$, then checking whether $|M(w)| < |G(w)|$. This test can be performed in $O(n)$ time, giving us an $O(n^3)$ algorithm overall.

3 Acknowledgements

I am very indebted to Dexter Kozen for posing this problem, and for sharing some of the clarity of his mathematical thought. The support of a Fannie and John Hertz Foundation Graduate Fellowship, and the National Science Foundation under grant CCR-9317320, is gratefully acknowledged.

References

[CG] Chang, S. K., A. Gill, "Algorithmic solution of the change-making problem," *J. Assoc. Comput. Mach.* 17:1 (January 1970), pp. 113-122.

[GJ] Garey, M. R., D. S. Johnson, *Computers and Intractability: a Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[KZ] Kozen, D., S. Zaks, "Optimal bounds for the change-making problem," *Theor. Comput. Sci.* 123 (1994), pp. 377-388.

[L] Lueker, G. S., "Two NP-complete problems in nonnegative integer programming," Report No. 178, Computer Science Laboratory, Princeton University, 1975.