Information Processing Letters 115 (2015) 280–284

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

An enumeration algorithm for all integers nonrepresentable by some positive integers

Shunichi Matsubara

Department of Integrated Information Technology, Aoyama Gakuin University, 5-10-1, Fuchinobe, Chuo-ku, Sagamihara, Kanagawa, 252-5258, Japan

ARTICLE INFO

Article history:
Received 27 June 2014
Received in revised form 25 September 2014
Accepted 26 September 2014
Available online 11 October 2014
Communicated by R. Uehara

Keywords:
Analysis of algorithms
Frobenius number
Enumeration algorithm
Amortized polynomial-time delay
Amortized analysis

ABSTRACT

In this paper, we propose an algorithm for enumerating all integers nonrepresentable by a given set of positive integers. We say that a positive integer n is *nonrepresentable* by positive integers $a_0, a_1, \cdots, a_{d-1}$ if there exist no nonnegative integers $x_0, x_1, \cdots, x_{d-1}$ such that $\sum_{i=0}^{d-1} x_i a_i = n$. In this paper, we prove that the new algorithm runs in $O(t^2 s)$ time, where t and s denote the input and output sizes, respectively; i.e. we prove that the algorithm can enumerate all the integers nonrepresentable by a given set of positive integers in amortized polynomial-time delay.

1. Introduction

Research on nonrepresentable integers has been motivated by both theoretical and practical considerations. In addition, the dimension d is regarded from two different perspectives. In this work, we present a theoretical study of enumeration for a fixed d; i.e. we estimate the computational complexity of the enumeration regarding d as a constant.

Let $a_0, a_1, \cdots, a_{d-1}$ be relatively prime positive integers that satisfy the inequality $a_0 < a_1 < \cdots < a_{d-1}$. We say that a positive integer n is *representable* by $a_0, a_1, \cdots, a_{d-1}$ if n can be written as a linear combination of nonnegative integers; i.e. there exists a set of nonnegative integers $x_0, x_1, \cdots, x_{d-1}$ such that $\sum_{i=0}^{d-1} x_i a_i = n$. Conversely, we say that n is *nonrepresentable* by $a_0, a_1, \cdots, a_{d-1}$ if n is not representable by $a_0, a_1, \cdots, a_{d-1}$. The set of all nonrepresentable integers is denoted by $NR(a_0, a_1, \cdots, a_{d-1})$. We call the greatest nonrepresentable integer the *Frobenius number*, denoted $NR_{max}(a_0, a_1, \cdots, a_{d-1})$. For example, $NR(4, 7, 9) = \{1, 2, 3, 5, 6, 10\}$ and $NR_{max}(4, 7, 9) = 10$. When enumerating all integers nonrepresentable by a_0, $a_1, \cdots, a_{d-1}$, we call d and the set of $a_0, a_1, \cdots, a_{d-1}$ the *dimension* and the *input* of the enumeration, respectively.

The number $|NR(a_0, a_1, \cdots, a_{d-1})|$ has been actively researched [1], where the vertical bars on both sides denote the cardinality of the set. Sylvester proved that $|NR(a, b)| = (1/2)(a-1)(b-1)$ for two positive integers a and b [2], which is among the best known results in nonrepresentable integer research. In a computational study, Krawczyk and Paz found the upper and lower bounds for $|NR(a_0, a_1, \cdots, a_{d-1})|$ and they proved that these bounds can be computed in polynomial time [3]. Although many such bounds have been identified [1], Krawczyk and Paz's bounds are adopted in our newly proposed algorithm. Besides the result by Krawczyk and Paz, a lot of bounds have been found [1]. However, no formula such as $|NR(a, b)| = (1/2)(a-1)(b-1)$ is known for dimensions higher than 2.

E-mail address: matsubara@it.aoyama.ac.jp.

http://dx.doi.org/10.1016/j.ipl.2014.09.028
0020-0190/© 2014 Elsevier B.V. All rights reserved.

In this paper, we propose an algorithm for enumerating all integers nonrepresentable by a given set of positive integers. The algorithm runs in $O(t^2 s)$ time, where t and s are the input and output sizes, respectively; i.e. the algorithm enumerates all nonrepresentable integers in amortized polynomial-time delay. The proposed algorithm adopts a simple approach; it sequentially checks all nonnegative integers from 0 to Krawczyk and Paz's upper bound of nonrepresentable integers. For each nonnegative integer in the span, the algorithm spends $O(t^2)$ time.

Although the enumeration of all integers nonrepresentable by a given set of positive integers has been little investigated, computational methods for finding the Frobenius number have been extensively published. The problem of finding the Frobenius number is known as the *Frobenius coin-exchange problem*. Ramírez Alfonsín proved that the Frobenius coin-exchange problem is NP-hard if the dimension d is a input parameter [4]. However, Kannan found that, if d is a constant, the problem can be computed in polynomial time [5]. Kannan's algorithm relies on the concepts and results from the geometry of numbers. An alternative polynomial-time algorithm for fixed d was developed by Barvinok and Woods [6]. Thus, if we consider d as a constant, then we may be able to develop an efficient algorithm for enumerating nonrepresentable integers by extending an algorithm that computes the Frobenius number. However, in Section 4, we point out that extending Frobenius number computations to enumerating nonrepresentable integers may not lead to efficient algorithms.

This paper is organized as follows. Section 2 presents essential concepts and notations of nonrepresentable integers. In addition, we also explain the concepts of bit complexity and polynomial-time delay. In Section 3, developing a new algorithm for enumerating all integers nonrepresentable by a given set of positive integers and proving its validity are presented. Moreover, we prove that the algorithm runs in amortized polynomial-time delay. A potential alternative approach, extending the algorithms for computing the Frobenius number to algorithms for enumerating nonrepresentable integers, is described in Section 4. The paper concludes with ideas for future work in Section 5.

2. Preliminary

When discussing integers nonrepresentable by positive integers $a_0, a_1, \cdots, a_{d-1}$, we assume that $a_0, a_1, \cdots, a_{d-1}$ are relatively prime and satisfy $a_0 < a_1 < \cdots < a_{d-1}$. In addition, we assume that $d \geq 2$. These assumptions are consistent with those of existing researches on the Frobenius number [1,7,5,3]. Where no ambiguity arises, $NR_{max}(a_0, a_1, \cdots, a_{d-1})$ and $NR(a_0, a_1, \cdots, a_{d-1})$ are often simply denoted by NR_{max} and NR, respectively.

The base of the logarithm is set to 2. Suppose that an algorithm outputs integers $o_0, o_1, \cdots, o_{n-1}$ from given integers $i_0, i_1, \cdots, i_{m-1}$. The *input size* is then defined as $\sum_{\alpha=0}^{m-1}(\lfloor \log(i_\alpha) \rfloor + 1)$. In other words, the input size is the number of bits required to write the integers. Similarly, the *output size* is defined as $\sum_{\alpha=0}^{n-1}(\lfloor \log(o_\alpha) \rfloor + 1)$.

We adopt a *bit operation* as a primitive operation. A bit operation is a unit of time rather than an elementary arithmetic operation. If a problem P can be solved in n bit operations, then n denotes the *bit complexity* of P. When finding nonrepresentable integers, computational costs are conventionally estimated by the bit complexity. Bit complexity is a theoretical measure; in practical research, the time unit is an elementary arithmetic operation.

In this paper, we focus on enumeration algorithms rather than single-object computation algorithms. In this case, the parameters of the computational complexity generally include both the input and output sizes, since the number of the outputs to enumerate can exponentially increase. For this purpose, we use the term *polynomial-time delay*. We say that an enumeration algorithm runs in *amortized polynomial-time delay* if the algorithm runtime is of the order of a function that is polynomial in the input size and linear in the output size and in *worst-case polynomial-time delay* if the following three parts can be computed in polynomial time in the input size. (1) Obtaining the first output object after starting the algorithm; (2) obtaining the next output object after outputting any output; (3) terminating the algorithm after obtaining the last output object. An enumeration algorithm runs in *polynomial-space* if the spatial requirements of the algorithm are of the order of a polynomial function in the input size. The algorithm notations adopted in this paper follow from pseudocode conventions in [8].

3. A new enumeration algorithm for nonrepresentable integers

3.1. Definitions

We propose a new algorithm called Enumerate, which enumerates all integers nonrepresentable by a set of positive integers. This algorithm uses an upper bound B of nonrepresentable integers for a given set of positive integers, which was reported by Krawczyk and Paz [3]. Enumerate checks the nonrepresentabilities of all integers from 0 to B and prints each nonrepresentable case. The representability of an integer is decided from a binary array T. For each i with $0 \leq i \leq B$, $T[i]$ determines whether i is nonrepresentable $(T[i] = 0)$ or representable $(T[i] = 1)$.

First, Enumerate computes the upper bound B and initializes each element of T to 0 except for $T[0]$, which is set to 1 because 0 is representable for any input. Next, Enumerate executes the main processes, sequentially stepping through each integer i from 0 to B in ascending order. The main processes determine the representability of each integer $i + a_j$, where $0 \leq j \leq d-1$. When Enumerate processes for $T[i]$, the algorithm rewrites $T[i + a_j]$ for each $i + a_j$, where $0 \leq j \leq d-1$. If $T[i] = 0$, then the algorithm does nothing. If $T[i] = 1$, then it overwrites $T[i + a_j]$ with 1 for each j $(0 \leq j \leq d-1)$. Note that the value of $T[i]$ has already been decided at the start of processing $T[i]$.

3.2. Formal proofs

Formal proofs are developed in this subsection. The following notation is adopted. The positive integers for which we prove statements are denoted $a_0, a_1, \cdots, a_{d-1}$. The symbols NR_{max} and NR denote $NR_{max}(a_0, a_1, \cdots, a_{d-1})$