LENGTH OF THE CONTINUED LOGARITHM ALGORITHM ON RATIONAL INPUTS

Jeffrey Shallit[1]
School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada
shallit@uwaterloo.ca

Received: 6/24/16, Revised: 6/16/20, Accepted: 8/19/20, Published: 8/31/20

Abstract

The continued logarithm algorithm was introduced by Gosper around 1978, and has been studied recently by several authors. In this note we show that the continued logarithm algorithm, on input of a rational number $p/q \geq 1$, terminates in at most $2\log_2 p + O(1)$ steps. Furthermore, this bound is tight, up to an additive constant.

1. Introduction

Let $\mathbb{Z}$ denote the integers, $\mathbb{N}$ denote the non-negative integers, and $\mathbb{N}_{>0}$ denote the positive integers.

The continued fraction algorithm, which expands every real number x in an expression of the form

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \ddots}}}, \tag{1}$$

with $a_0 \in \mathbb{Z}$ and $a_i \in \mathbb{N}_{>0}$ for $i \geq 1$, has been extensively studied, in part because of its relationship to the Euclidean algorithm. In particular, it is known that the expression (1) is essentially unique, and terminates with final term a_n if and only if x is a rational number. In this case, if $x = p/q$, the length of the expansion is at most $O(\log q)$, as has been known since the 1841 work of Finck [8]. Furthermore, examples achieving this bound are known.

Around 1978, Gosper [6] introduced an analogue of the continued fraction algorithm for real numbers, called the *continued logarithm algorithm*, which expands

[1]Supported by NSERC grant 2018-04118.

every real number $x \geq 1$ in an expression of the form

$$x = 2^{k_0}(1 + \cfrac{1}{2^{k_1}(1 + \cfrac{1}{2^{k_2}(1 + \cfrac{1}{2^{k_3}(1 + \ddots)})})}) \quad .$$

where $k_i \in \mathbb{N}$ for $i \geq 0$. This can be abbreviated as follows: $x = \langle k_0, k_1, k_2, k_3, \ldots \rangle$. More recently, this algorithm was studied by Brabec [3, 4, 5], Borwein, Calkin, Lindstrom, and Mattingly [1], and Borwein, Hare, and Lynch [2].

As in the case of the ordinary continued fraction algorithm, it is known that this expression is essentially unique (modulo the requirement that the last term must not be 0 if $x > 1$) and that the algorithm terminates if and only if x is rational. However, up to now, no worst-case estimate of the length of the expansion has been given. In this note, we provide such an estimate.

2. The Continued Logarithm Algorithm

Let $x_0 = x \geq 1$. The continued logarithm algorithm works as follows: at each stage, choose the integer $k \geq 0$ uniquely to satisfy the inequality $1 \leq x_i/2^k < 2$, and set $k_i = k$. If $x_i = 2^k$, the algorithm terminates with output $\langle k_0, k_1, \ldots, k_i \rangle$. Otherwise, the algorithm continues with $x_{i+1} = 1/(x_i/2^k - 1)$.

For example, if $x = 96/7$, we get

$$\begin{aligned}
x_0 &= x = 96/7 \\
k_0 &= 3 \\
x_1 &= 1/(x_0/2^3 - 1) = 7/5 \\
k_1 &= 0 \\
x_2 &= 1/(x_1/2^0 - 1) = 5/2 \\
k_2 &= 1 \\
x_3 &= 1/(x_2/2^1 - 1) = 4 \\
k_3 &= 2.
\end{aligned}$$

and hence

$$\begin{aligned}
\frac{96}{7} &= 2^3(1 + \cfrac{1}{2^0(1 + \cfrac{1}{2^1(1 + \cfrac{1}{2^2})})}) \\
&= \langle 3, 0, 1, 2 \rangle.
\end{aligned}$$

Similarly, $2^k = \langle k \rangle$ for $k \geq 0$. Since the continued logarithm expansion is (essentially) unique, we can regard an expression like $x = \langle k_0, k_1, \ldots, k_n \rangle$ as *either* an evaluation of a certain function of the variables $k_0, k_1, \ldots, k_n$, *or* as a statement about the output of the continued logarithm algorithm on an input x. We trust there will be no confusion on the proper interpretation in what follows.

There are two different natural measures of the complexity of the algorithm on rational inputs. The first is the number of steps $n+1$ in $x = \langle k_0, k_1, \ldots, k_n \rangle$, which we write as $L(x)$. The second is the *total* number of division steps $k_0 + k_1 + \cdots + k_n$, which we write as $T(x)$. In this note we get asymptotically tight bounds for L and T on rational numbers $p/q \geq 1$.

3. The Bound on L

Consider performing the continued logarithm algorithm CL on a rational input $\frac{p}{q} \geq 1$, getting back $\frac{p}{q} = \langle k_0, k_1, \ldots, k_n \rangle$. We can associate a rational number $\frac{p}{q}$ with the pair (p, q). While this association is not unique (for example, 2 can be represented either by $(2, 1)$ or $(4, 2)$), it does not create problems in what follows. By consolidating the division steps, we can express the continued logarithm algorithm on rational numbers as a function of two integers that takes its value on finite lists, as follows:

$$\mathrm{CL}(p, q) = \begin{cases} k, & \text{if } p = 2^k q \text{ for some } k \geq 0; \\ k, \mathrm{CL}(2^k q, p - 2^k q), & \text{if } 1 < \frac{p}{2^k q} < 2. \end{cases}$$

Here the comma denotes concatenation. For example, $\mathrm{CL}(96, 7) = (3, 0, 1, 2)$.

The idea of our bound on L is to consider how the measure $f(p, q) = p^2 + q^2$ changes as the algorithm proceeds. In our interpretation of the algorithm on pairs (p, q), it replaces (p, q) with (p', q'), where $p' = 2^k q$ and $q' = p - 2^k q$ for $1 \leq \frac{p}{2^k q} < 2$, and terminates when $q = 0$. First we show that $f(p, q)$ strictly decreases in each step of the algorithm.

Lemma 1. *If the continued logarithm algorithm takes (p, q) to (p', q'), then $f(p', q') < f(p, q)$.*

Proof. From the inequality $\frac{p}{q \cdot 2^k} \geq 1$ we get

$$\frac{p}{q} \geq 2^k > 2^k - \frac{1}{2^{k+1}}.$$

Multiplying by $2^{k+1} q^2$, we get $2^{k+1} pq > (2^{2k+1} - 1) q^2$. Adding p^2 to both sides and

rearranging gives

$$\begin{aligned} p^2+q^2 &> p^2-2^{k+1}pq+2^{2k+1}q^2 \\ &= (2^kq)^2+(p-2^kq)^2 \\ &= (p')^2+(q')^2, \end{aligned}$$

as desired. □

Next we show how f decreases as the algorithm proceeds. We use the notation $(p,q) \to^k (p',q')$ to denote that one step of the algorithm replaces p/q with p'/q', where $p' = 2^kq$ and $q' = p - 2^kq$.

Lemma 2. *If $(p,q) \to^0 (p',q')$ then $f(p',q') \le f(p,q)/2$.*

Proof. The condition $k = 0$ implies $p' = q$ and $q' = p - q$. Then $1 \le \frac{p}{q} < 2$. If $\frac{p}{q} = 1$ then the algorithm terminates, so assume $\frac{p}{q} > 1$. Write $\frac{p}{q} = c$. If $1 < c \le 2$, then $(c-1)(c-3) < 0$. Multiplying by q^2 gives $q^2(c^2-4c+3) < 0$. Hence, using the fact that $p = cq$, we get $p^2 - 4pq + 3q^2 < 0$. Dividing by 2 and rearranging, we get $f(p',q') = q^2 + (p-q)^2 < \frac{1}{2}(p^2+q^2) < \frac{1}{2}f(p,q)$. □

Lemma 3. *If two steps of the continued logarithm algorithm are $(p,q) \to^k (p',q') \to^0 (p'',q'')$ with $k \ge 1$ then $f(p'',q'') < \frac{1}{4}f(p,q)$.*

Proof. The first step implies that $1 < \frac{p}{2^kq} < 2$, and $p' = 2^kq$, $q' = p - 2^kq$. The second step implies that $1 < \frac{2^kq}{p-2^kq} < 2$ and $p'' = p - 2^kq$, $q'' = 2^{k+1}q - p$.

Define $c = \frac{p}{2^kq}$ and observe that the inequalities of the previous paragraph imply $3/2 \le c \le 2$. Consider the polynomial $h(c) = 7c^2 - 24c + 20$. Since the roots of this polynomial are $10/7$ and 2, we clearly see that $h(c) \le 0$ for $3/2 \le c \le 2$. Hence $q^2(7c^2-24c+20)2^{2k} < q^2$ for $q \ge 1$ and $k \ge 1$. Substituting $c = \frac{p}{2^kq}$ and simplifying gives $7p^2 - 24 \cdot 2^kpq + 20 \cdot 2^{2k}q^2 < q^2$. Adding p^2 to both sides and then dividing by 4 gives $2p^2 - 6 \cdot 2^kpq + 5 \cdot 2^{2k}q^2 < \frac{1}{4}(p^2+q^2)$. But the left side of this inequality is $(p-2^kq)^2 + (q2^{k+1}-p)^2$. Thus we have proved $(p'')^2 + (q'')^2 < \frac{1}{4}(p^2+q^2)$. □

Lemma 4. *If two steps of the continued logarithm algorithm are $(p,q) \to^k (p',q') \to^{k'} (p'',q'')$ with $k,k' \ge 1$, then p'' and q'' are both divisible by 2, and $f(p''/2, q''/2) < \frac{1}{4}f(p,q)$.*

Proof. In that case the first step replaces (p,q) by $(p',q') = (q \cdot 2^k, p - q \cdot 2^k)$, and the second step replaces this latter pair with

$$(p'',q'') = ((p - q \cdot 2^k)2^{k'}, q(2^{k+k'} + 2^k) - p \cdot 2^{k'}).$$

Now both elements of this latter pair are divisible by $2^{\min(k,k')} \ge 2$. Since these correspond to numerator and denominator, we can divide both elements of the pair

by 2 and obtain an equivalent pair of integers $(p''/2, q''/2)$. Note that $\mathrm{CL}(p'', q'') = \mathrm{CL}(p''/2, q''/2)$. By Lemma 3 we have $f(p''/2, q''/2) = \frac{1}{4} f(p'', q'') < \frac{1}{4} f(p', q') < \frac{1}{4} f(p, q)$, as desired. □

Theorem 1. *On input $p/q \geq 1$ the continued logarithm algorithm uses at most $2 \log_2 p + 2$ steps.*

Proof. Consider the continued logarithm expansion and process it from left to right as follows: if a term is 0, use Lemma 2. If a term is $k \geq 1$, group it with the term that follows and use either Lemma 3 or Lemma 4. By doing so we group all terms except possibly the last. Lemma 2 shows that a single step reduces f by a factor of 2. Lemmas 3 and 4 show that two steps reduce f by a factor of 4. Thus the total number of steps on input (p, q) is at most $\log_2(p^2 + q^2) + 1$, where the $+1$ term takes into account the last term that might be ungrouped.

So the algorithm uses at most $\log_2(p^2 + q^2) + 1$ steps. Since $p \geq q$, we have

$$\log_2(p^2 + q^2) + 1 \leq \log_2(2p^2) + 1 \leq \log_2(p^2) + 2 \leq (2 \log_2 p) + 2.$$

□

A nearly matching lower bound of $2 \log_2 p + O(1)$ is achievable, as the following class of examples shows.

Theorem 2. *For $n \geq 1$, on input $2^n - 1$ the continued logarithm algorithm takes $2n - 2$ steps.*

Proof. We have $2^n - 1 = \langle n-1, 0, n-2, 0, \ldots, 2, 0, 1, 1 \rangle$, as can be easily proved by induction. □

4. The Bound on T

Theorem 3. *Let $\frac{p}{q} \geq 1$. Then $T(\frac{p}{q}) < (\log_2 p)(2 \log_2 p + 2)$.*

Proof. As the continued logarithm algorithm proceeds, the numerators strictly decrease, so each k is bounded by $\log_2 p$. The number of steps is bounded by Theorem 1. □

Theorem 4. *For $n \geq 1$ we have $T(2^n - 1) = n(n-1)/2 + 1$.*

Proof. Follows immediately from the expansion $2^n - 1 = \langle n-1, 0, n-2, 0, \ldots, 2, 0, 1, 1 \rangle$ given in the proof of Theorem 3. □

5. Remarks

In a previous version of this paper, we asked, "What is the average case behavior of the number of steps of the continued logarithm algorithm on rational numbers p/q, with $q < p < 2q$, as $q \to \infty$?" This has recently been answered by Rotondo, Vallée, and Viola [7].

Another interesting question is whether the sequence $(L(n))_{n\geq 1}$ a k-regular sequence for some $k \geq 2$. The available numerical evidence suggests it is not.

Acknowledgment. I thank Robbert Fokkink and Wieb Bosma for their helpful comments.

References

[1] Jonathan M. Borwein, Neil J. Calkin, Scott B. Lindstrom, and Andrew Mattingly, Continued logarithms and associated continued fractions, *Experimental Math.* **26** (2017), 412–429.

[2] Jonathan M. Borwein, Kevin G. Hare, and Jason G. Lynch, Generalized continued logarithms and related continued fractions, *J. Integer Seq.* **20** (2017), Article 17.5.7.

[3] Tomáš Brabec, Hardware implementation of continued logarithm arithmetic, In *Scientific Computing, Computer Arithmetic and Validated Numerics, 2006. SCAN 2006*, IEEE, 2006, pp. 1–9.

[4] Tomáš Brabec, On progress of investigations in continued logarithms, Preprint. Available at http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.93.4552&rep=rep1&type=pdf.

[5] Tomáš Brabec, Speculatively redundant continued logarithm representation, *IEEE Trans. Computers* **59** (2010), 1441–1454.

[6] Bill Gosper. Continued fraction arithmetic, Unpublished manuscript, c. 1978. Available at http://perl.plover.com/classes/cftalk/INFO/gosper.txt or http://www.tweedledum.com/rwg/cfup.htm.

[7] P. Rotondo, B. Vallée, and A. Viola, Analysis of the continued logarithm algorithm, In M. A. Bender et al., eds., *LATIN 2018*, LNCS 10807, pp. 849–863, 2018.

[8] J. Shallit, Origins of the analysis of the Euclidean algorithm, *Historia Math.* **21** (1994), 401–419.