Draft.

COMPUTING LOGARITHM FLOORS IN ESSENTIALLY LINEAR TIME

DANIEL J. BERNSTEIN

ABSTRACT. This paper fills a gap in some incomplete algorithms stated in the literature, notably a recent algorithm for determining primality. What this paper presents are algorithms to compute $\lfloor(\log n)^2\rfloor$ and $\lfloor\sqrt{m}\lg n\rfloor$, given positive integers m and n. Here log is the natural logarithm, and lg is the base-2 logarithm. Baker's theorem on linear forms in logarithms implies that the algorithms take essentially linear time if $\lg m \in (\lg n)^{o(1)}$.

1. INTRODUCTION

As usual, log is the natural logarithm, and lg is the base-2 logarithm.

Section 2 of this paper presents an algorithm that, given a positive integer n, computes $\lfloor(\log n)^2\rfloor$ and $\lceil(\log n)^2\rceil$. The algorithm takes time at most $(\lg n)^{1+o(1)}$.

Section 3 presents an algorithm that, given positive integers m and n, computes $\lfloor\sqrt{m}\lg n\rfloor$ and $\lceil\sqrt{m}\lg n\rceil$. The algorithm takes time at most $(\lg n)^{1+o(1)}$ if $\lg m \in (\lg n)^{o(1)}$.

Previous authors have implicitly—and, I suspect, unintentionally—assumed the existence of polynomial-time algorithms for these two problems. See Section 4 for further discussion.

Proving computability. The usual way to compute $\lfloor\alpha\rfloor$ and $\lceil\alpha\rceil$ is to compute high-precision bounds on α. My paper [7] explains how to quickly compute high-precision bounds on logarithms. But this is not enough: what happens if α is an integer?

Answer: Lindemann's theorem implies that $(\log n)^2$ is not an integer unless it is an obvious integer, i.e., unless $n = 1$. The theorem states that an algebraic number outside $\{0, 1\}$ never has an algebraic logarithm; in particular, $\log n$ is not algebraic for $n > 1$. See [3, page 1].

Similarly, the Gelfond-Kuzmin theorem implies that $\sqrt{m}\lg n$ is not an integer unless it is an obvious integer. The theorem states that $(\log\alpha_1)/\log\alpha_2$ is never a quadratic irrational; here α_1, α_2 are algebraic numbers outside $\{0, 1\}$. This is a special case of the Gelfond-Schneider theorem, which states that $(\log\alpha_1)/\log\alpha_2$ is never algebraic unless it is rational. See [3, pages 1–2].

Permanent document ID: `97bbdc1ce6aff974c789eab21b9cfba1`.

Date: 2003.06.30.

2000 *Mathematics Subject Classification.* Primary 11Y16; Secondary 11J86.

The author was supported by the National Science Foundation under grant DMS–0140542, and by the Alfred P. Sloan Foundation.

Proving essentially-linear-time computability. Even if α is not an integer, what happens if α is extremely close to an integer? Bounds on α of increasingly high precision will eventually separate α from that integer, but what happens if the required precision is, say, $\exp\exp\exp b$, where b is the number of bits of input?

Answer: Baker's theorem implies that $b^{1+o(1)}$ bits of precision suffice. Explicit bounds appear in Sections 2 and 3. (Similar applications of transcendental number theory appear in [5] and [9].)

Beware that the bounds say nothing about real-world computations: they include extremely large constant factors. I have made no attempt to optimize those constant factors. The algorithms here are nevertheless reasonably fast in practice, because they start from low precision, using higher precision only if necessary.

2. Floor of logarithm squared

Here is an algorithm that, given a positive integer n, computes $\lfloor(\log n)^2\rfloor$ and $\lceil(\log n)^2\rceil$:

1. If $n = 1$: Print $0, 0$ and stop.
2. Compute a precise interval $[L, R]$ containing $\log n$, as explained in [7].
3. If $[L^2, R^2]$ does not contain an integer, print $\lfloor L^2\rfloor, \lceil R^2\rceil$ and stop.
4. Double the number of bits of precision. Go back to step 2.

This algorithm is parametrized by the starting precision. It is simplest to start with 1 bit of precision; it is fastest to start with slightly more than $2\lg\log n$ bits of precision.

The following theorem states that the algorithm terminates once the precision reaches approximately $3\cdot 2^{1000}(\lg n)(\lg\lg n)^2$ bits, if not sooner. Thus the total time for the algorithm is at most $(\lg n)^{1+o(1)}$.

Theorem 2.1. *Let n be an integer with $n \ge 8$. Define $j = \lceil\lg n\rceil$ and $k = 3\cdot 2^{1000} j\lceil\lg j\rceil^2$. Let L and R be real numbers such that $L \le \log n \le R$ and $|R - L| \le 2^{-k}$. Then $\lfloor(\log n)^2\rfloor < L^2 \le R^2 < \lceil(\log n)^2\rceil$.*

This is a typical application of Baker's theorem. Here is the general statement of Baker's theorem from [3, Theorem 1]: Assume that $\beta_0, \beta_1, \ldots, \beta_\ell, \alpha_1, \ldots, \alpha_\ell$ are elements of a number field of degree at most d; that each β_i has height at most $B \ge 4$, where "height" means "maximum absolute value of coefficients in the minimal polynomial over $\mathbf{Z}$"; that α_i has height at most $A_i \ge 4$; that $\Lambda \ne 0$, where $\Lambda = \beta_0 + \beta_1\log\alpha_1 + \cdots + \beta_\ell\log\alpha_\ell$; and that $\Omega = (\log A_1)\cdots(\log A_\ell)$. Then $|\Lambda| > (B\Omega)^{-(16\ell d)^{200\ell}\Omega\log\Omega}$. Baker actually states this bound with $\log(\Omega/\log A_\ell)$ in place of $\log\Omega$, but that improvement is only for $\ell \ge 2$.

Proof. I will show that $f < L^2$ if $f = \lfloor(\log n)^2\rfloor$, and that $f > R^2$ if $f = \lceil(\log n)^2\rceil$. Note that either choice of f satisfies $4 \le f \le j^2$ since $4 \le (\log n)^2 \le j^2$.

By Lindemann's theorem, $\log n \ne \sqrt{f}$. Apply Baker's theorem with $\ell = 1$, $\beta_0 = \sqrt{f}$, $\beta_1 = -1$, $\alpha_1 = n$, $d = 2$, $\Lambda = \sqrt{f} - \log n \ne 0$, $B = f \ge 4$, $A_1 = n \ge 4$, $\Omega = \log n < j$, $B\Omega < fj \le j^3$, $(16\ell d)^{200\ell} = 2^{1000}$, $\Omega\log\Omega < j\lg j$, and $(16\ell d)^{200\ell}\Omega\log\Omega\lg B\Omega < 2^{1000} j\lg j\lg j^3 \le k$ to see that $|\sqrt{f} - \log n| > 2^{-k}$.

In particular, if $f = \lfloor(\log n)^2\rfloor$, then $\sqrt{f} < \log n$, so $\sqrt{f} < \log n - 2^{-k} \le R - 2^{-k} \le L$; i.e., $f < L^2$ as claimed. Similarly, if $f = \lceil(\log n)^2\rceil$, then $\sqrt{f} > \log n$, so $\sqrt{f} > \log n + 2^{-k} \ge L + 2^{-k} \ge R$; i.e., $f > R^2$ as claimed. □

3. Floor of square root times logarithm

Here is an algorithm that, given positive integers m and n, computes $\lfloor\sqrt{m}\lg n\rfloor$ and $\lceil\sqrt{m}\lg n\rceil$:

1. If $n = 1$: Print $0, 0$ and stop.
2. If n is a power of 2 and m is a square: Print $\sqrt{m}\lg n, \sqrt{m}\lg n$ and stop.
3. Compute a precise interval $[L, R]$ containing $\sqrt{m}\lg n$, as explained in [7].
4. If $[L, R]$ does not contain an integer, print $\lfloor L\rfloor, \lceil R\rceil$ and stop.
5. Double the number of bits of precision. Go back to step 3.

For theoretical purposes, it is simplest to start with 1 bit of precision, as in Section 2. See [5] for square-testing algorithms.

The following theorem states that the algorithm terminates once the precision reaches approximately $2^{2401}\lg n \lg\lg n \lg(m\lg n)$ bits, if not sooner. In particular, the required precision is at most $(\lg n)^{1+o(1)}$ if $\lg m \in (\lg n)^{o(1)}$.

Theorem 3.1. *Let m and n be positive integers. Assume that $n \ge 2$, and that n is not a power of 2 if m is a square. Define $j = \lceil\lg 2n\rceil$ and $k = 2^{2401} j \lceil\lg j\rceil \lceil\lg 2mj\rceil$. Let L and R be real numbers such that $L \le \sqrt{m}\lg n \le R$ and $|R - L| \le 2^{-k}$. Then $\lfloor\sqrt{m}\lg n\rfloor < L \le R < \lceil\sqrt{m}\lg n\rceil$.*

Proof. I will show that $f < L$ if $f = \lfloor\sqrt{m}\lg n\rfloor$, and that $f > R$ if $f = \lceil\sqrt{m}\lg n\rceil$. Note that either choice of f satisfies $1 \le f \le mj$ since $1 \le \sqrt{m}\lg n \le mj$.

If m is a square then n is not a power of 2 so $n^{\sqrt{m}} \ne 2^f$. If m is not a square then, by the Gelfond-Kuzmin theorem, the quadratic irrational $f/\sqrt{m}$ does not equal $(\log n)/\log 2$. Either way, $\sqrt{m}\log n - f\log 2 \ne 0$.

Apply Baker's theorem with $\ell = 2$, $\beta_0 = 0$, $\beta_1 = \sqrt{m}$, $\beta_2 = -f$, $\alpha_1 = n$, $\alpha_2 = 2$, $d = 2$, $\Lambda = \sqrt{m}\log n - f\log 2 \ne 0$, $B = 4mf \ge 4$, $A_1 = 2n \ge 4$, $A_2 = 4$, $\Omega = (\log 2n)\log 4 < j$, $B\Omega < 4mfj \le (2mj)^2$, $(16\ell d)^{200\ell} = 2^{2400}$, $\Omega\log\Omega < j\lg j$, and $(16\ell d)^{200\ell}\Omega\log\Omega\lg B\Omega < 2^{2400} j\lg j\lg((2mj)^2) \le k$ to see that $|\sqrt{m}\lg n - f| > |\sqrt{m}\log n - f\log 2| > 2^{-k}$.

In particular, if $f = \lfloor\sqrt{m}\lg n\rfloor$, then $f < R - 2^{-k} \le L$. Similarly, if $f = \lceil\sqrt{m}\lg n\rceil$, then $f > L + 2^{-k} \ge R$. □

4. Applications

I wrote this paper to retroactively justify the claim that two algorithms in the literature take polynomial time:

- Bach and Shallit in [2, page 268] state an algorithm that performs an inner loop "for $a \leftarrow 2$ to $\lfloor(\log n)^2\rfloor$." They claim that the time for the algorithm is "clearly" dominated by the time for the inner loop, which in turn is $o((\lg n)^6)$. However, they neglect to prove that $\lfloor(\log n)^2\rfloor$ is computable from n in time $o((\lg n)^6)$.
- Agrawal, Kayal, and Saxena in [1, page 3] state an algorithm (repeated on the front cover of the May 2003 Notices of the AMS) that performs an inner loop for each integer a from 1 through "$\lfloor 2\sqrt{\phi(r)}\log n\rfloor$," where "log" means lg. They claim that this algorithm takes polynomial time. However, they neglect to prove that $\lfloor\sqrt{4\phi(r)}\lg n\rfloor$ is computable from $\phi(r)$ and n in polynomial time.

One could, in both cases, use a wider range of integers a; perhaps the authors actually meant $\lfloor(31487/65536)\lceil\lg n\rceil^2\rfloor$ and $2\lceil\sqrt{\phi(r)}\rceil\lceil\lg n\rceil$, both of which are easily computable without the techniques in this paper. That is, however, not what they wrote.

In my own presentations of the Agrawal-Kayal-Saxena idea, such as [6, Theorem 2.1], I used a smaller range of integers a, terminated by an easily computable binomial-coefficient condition, which is also the condition that naturally arises in the Agrawal-Kayal-Saxena proof. I am happy to sacrifice short formulas in favor of simple, fast programs and straightforward proofs. I realize, however, that many authors take the opposite view. This paper provides subroutines for those authors to use.

References

[1] Manindra Agrawal, Neeraj Kayal, Nitin Saxena, *PRIMES is in P (revised)* (2003).

[2] Eric Bach, Jeffrey Shallit, *Algorithmic number theory, volume 1: efficient algorithms*, MIT Press, Cambridge, Massachusetts, 1996. ISBN 0–262–02405–5. Available from `http://www.math.uwaterloo.ca/~shallit/ant.html`.

[3] Alan Baker, *The theory of linear forms in logarithms*, in [4] (1977), 1–27. MR 58:16543.

[4] Alan Baker, David W. Masser (editors), *Transcendence theory: advances and applications: proceedings of a conference held at the University of Cambridge, Cambridge, January–February, 1976*, Academic Press, London, 1977. ISBN 0–12–074350–7. MR 56:15573.

[5] Daniel J. Bernstein, *Detecting perfect powers in essentially linear time*, Mathematics of Computation **67** (1998), 1253–1283. ISSN 0025–5718. MR 98j:11121. Available from `http://cr.yp.to/papers.html`.

[6] Daniel J. Bernstein, *Proving primality in essentially quartic random time.*

[7] Daniel J. Bernstein, *Computing logarithm intervals in essentially linear time with the AGM iteration.*

[8] Krzysztof Diks, Wojciech Ritter (editors), *Mathematical foundations of computer science 2002: 27th international symposium, MFCS 2002, Warsaw, Poland, 26–30.08.2002: proceedings*, Lecture Notes in Computer Science, 2420, Springer, Berlin, 2002.

[9] Mika Hirvensalo, Juhani Karhumäki, *Computing partial information out of intractable one—the first digit of* 2^n *at base* 3 *as an example*, in [8] (2002), 319–327.

Department of Mathematics, Statistics, and Computer Science (M/C 249), The University of Illinois at Chicago, Chicago, IL 60607–7045

E-mail address: `djb@cr.yp.to`