A Randomized Sublinear Time Parallel GCD Algorithm for the EREW PRAM*

Jonathan P. Sorenson
Computer Science and Software Engineering, Butler University
Indianapolis, IN 46208 USA
sorenson@butler.edu, http://www.butler.edu/~sorenson
http://digitalcommons.butler.edu/facsch_papers/81/ (for paper download in PDF)

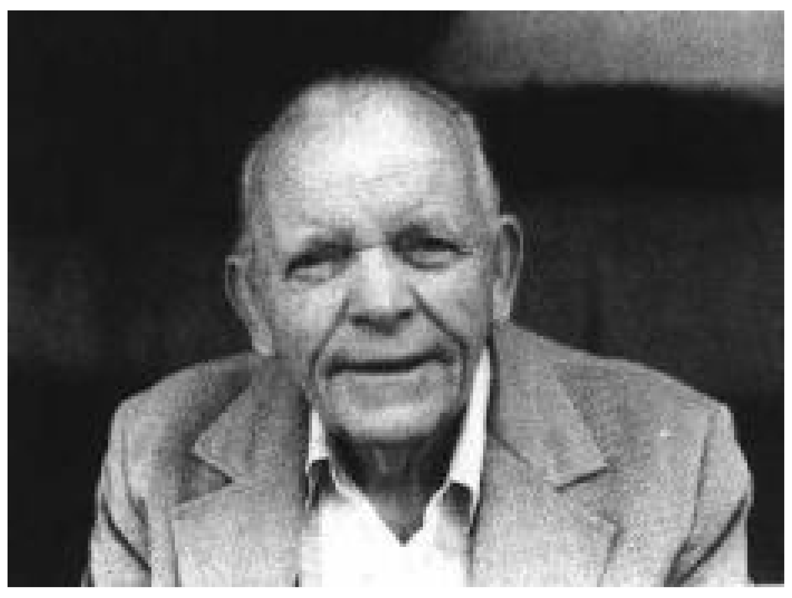
Overview

The Greatest Common Divisor of two integers x, y is the largest integer d such that $d \mid x$ and $d \mid y$. Most GCD algorithms are based, more or less, on Euclid's algorithm. Throughout, let $n := \log_2 x$, $m := \log_2 y$, with $n \geq m$.

Previous Work: Sequential Complexity

$O(nm)$	Euclid, about 300 BCE
$O(nm/\log n)$	Lehmer [9] (Jebelean [6], Sorenson [14])
$O(n(\log n)^2 \log\log n)$	Knuth-Schönhage [11]
	Stehlé and Zimmerman [17]
$O(n^2)$	Binary algorithm
	(Stein[18], Knuth[8], Brent[3], Vallée[19] & others)
$O(n^2/\log n)$	Jebelean [5], Weber [20], Sorenson [13, 15]

Euclid



D. H. Lehmer

Definitions and Background

Parallel Random Access Machine – Potentially infinite number of processors, potentially infinite shared memory with random access. Programs execute in lockstep.

- **EREW PRAM**: no concurrency of memory access allowed.
- **CREW PRAM**: concurrent reads allowed, but not writes.
- **CRCW PRAM**: concurrent reads and writes permitted.

The parallel complexity of the integer GCD problem is open. No $\mathcal{NC}$ algorithm is known, nor has it been shown to be $\mathcal{P}$-complete.

Previous Work: Parallel Complexity

CRCW PRAM Results

$O(n\log\log n/\log n)$	Kannan, Miller, Rudolph [7]
$O(n/\log n)$	Chor and Goldreich [4]
	(Sorenson [13], Sedjelmaci [12])
$O((\log n)^2)$	Adleman and Kompella [1]
randomized	$\exp[O(\sqrt{n}\log n)]$ processors

CREW PRAM Results

$O(n\log\log n/\log n)$ time by adapting [4] or [13]

EREW PRAM Results

$O(n)$ running time - Purdy's algorithm [10]

New Result

EREW PRAM: Compute $\gcd(x,y)$ with probability $1 - o(1)$ in $O(n\log\log n/\log n)$ time using $n^{6+\epsilon}$ processors. [16]

Reduction

Our inputs are integers x, y with $x \geq y > 0$.

- Choose a prime bound $B > 0$, and assume $p \mid x$ or $p \mid y$ implies $p > B$.
- Choose a at random, $1 \leq a \leq y - 1$.
- Compute $r := ax \bmod y$.
- Remove all prime divisors $\leq B$ from r producing s. Thus $P(r/s) \leq B$.

We use $(x, y) \to (y, s)$ for our reduction. We claim:

- $\gcd(x,y) = \gcd(y,s)$ with probability $1 - o(1)$.
  (This fails only if $\gcd(a,y) > 1$, or $\gcd(a,y) > B$, which is unlikely.)
- with probability at least $1/B$, we have

$$\log s \quad < \quad \log r - \frac{(\log B)^2}{2\log\log B}.$$

Algorithm Outline

Preprocessing

This takes $o(n)$ time in parallel:

- Choose $B := n^2$ (larger is ok)
- Remove and save common divisors of x, y that are $\leq B$.

Main Loop

While $xy \neq 0$ do the following:

- Perform $2B\log n$ reductions in parallel
- Save the smallest s value found
- $x := y$; $y := s$;

Notes:

- $(1 - 1/B)^{2B\log n} = O(1/n^2)$.
- One loop iteration takes $O(\log n)$ time – division by y is the bottleneck (Beame, Cooke, Hoover [2]).
- Total number of iterations is $O\left(\frac{n\log\log B}{(\log B)^2}\right)$.

Postprocessing

This takes $o(n)$ time in parallel:

- Restore saved common divisors $\leq B$, and combine those with $x + y$ to compute the answer.
- Verify the answer divides the original values of x, y. If not, report failure.

Technical Theorem

Define

$$W(x) := \frac{c \cdot (\log B(x))^2}{\log\log B(x)}, \quad c > 0,$$
$$F(x) := \#\{n \leq x \,:\, n = my,\ P(m) \leq B(x),\ \log m \geq W(x)\}.$$

That is, $F(x)$ counts integers $n \leq x$ where n has a $B(x)$-smooth divisor that is $\geq \exp W(x)$.

Theorem. *Let $\epsilon > 0$. For sufficiently large x we have*

$$F(x) \quad \geq \quad x \cdot B(x)^{-c(1+\epsilon)}.$$

Proof: Exercise for the reader. □

References

[1] L. M. Adleman and K. Kompella. Using smoothness to achieve parallelism. In *20th Annual ACM Symposium on Theory of Computing*, pages 528–538, 1988.

[2] P. W. Beame, S. A. Cook, and H. J. Hoover. Log depth circuits for division and related problems. *SIAM Journal on Computing*, 15:994–1003, 1986.

[3] R. P. Brent. Analysis of the binary Euclidean algorithm. In J. F. Traub, editor, *Algorithms and Complexity*, pages 321–355. Academic Press, 1976.

[4] B. Chor and O. Goldreich. An improved parallel algorithm for integer GCD. *Algorithmica*, 5:1–10, 1990.

[5] Tudor Jebelean. A generalization of the binary GCD algorithm. In M. Bronstein, editor, *1993 ACM International Symposium on Symbolic and Algebraic Computation*, pages 111–116, Kiev, Ukraine, 1993. ACM Press.

[6] Tudor Jebelean. A double-digit Lehmer-Euclid algorithm for finding the GCD of long integers. *J. Symbolic Comput.*, 19(1-3):145–157, 1995. MR 96h:11128.

[7] R. Kannan, G. Miller, and L. Rudolph. Sublinear parallel algorithm for computing the greatest common divisor of two integers. *SIAM Journal on Computing*, 16(1):7–16, 1987.

[8] D. E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms*, volume 2. Addison-Wesley, Reading, Mass., 3rd edition, 1998.

[9] D. H. Lehmer. Euclid's algorithm for large numbers. *American Mathematical Monthly*, 45:227–233, 1938.

[10] G. B. Purdy. A carry-free algorithm for finding the greatest common divisor of two integers. *Computers & Mathematics with Applications*, 9(2):311–316, 1983.

[11] A. Schönhage. Schnelle Berechnung von Kettenbruchentwicklungen. *Acta Informatica*, 1:139–144, 1971.

[12] Sidi Mohamed Sedjelmaci. A parallel extended gcd algorithm. *J. Discrete Algorithms*, 6(3):526–538, 2008.

[13] Jonathan P. Sorenson. Two fast GCD algorithms. *Journal of Algorithms*, 16:110–144, 1994.

[14] Jonathan P. Sorenson. An analysis of Lehmer's Euclidean GCD algorithm. In A. H. M. Levelt, editor, *1995 ACM International Symposium on Symbolic and Algebraic Computation*, pages 254–258, Montreal, Canada, July 1995. ACM Press.

[15] Jonathan P. Sorenson. An analysis of the generalized binary gcd algorithm. In Alf van der Poorten and Andreas Stein, editors, *High Primes and Misdemeanors: Lectures in Honour of the 60th Birthday of Hugh Cowie Williams*, pages 327–340, Banff, Alberta, Canada, 2004.

[16] Jonathan P. Sorenson. A randomized sublinear time parallel GCD algorithm for the EREW PRAM. *Information Processing Letters*, 110(5):198 – 201, 2010. Preliminary version available from arXiv.org.

[17] Damien Stehlé and Paul Zimmermann. A binary recursive GCD algorithm. In Duncan Buell, editor, *Sixth International Algorithmic Number Theory Symposium*, pages 411–425, Burlington, Vermont, USA, June 2004. Springer. LNCS 3076.

[18] J. Stein. Computational problems associated with Racah algebra. *Journal of Computational Physics*, 1:397–405, 1967.

[19] Brigitte Vallée. The complete analysis of the binary Euclidean algorithm. In *Algorithmic number theory (Portland, OR, 1998)*, volume 1423 of *Lecture Notes in Comput. Sci.*, pages 77–94. Springer, Berlin, 1998. MR 2000k:11143a.

[20] Kenneth Weber. The accelerated integer GCD algorithm. *ACM Trans. Math. Software*, 21(1):111–122, 1995. MR 96h:68084.

* Supported in part by a grant from the the Holcomb Awards Committee. Appeared as [16].