Analysis of de Rooij's Algorithm for the Greatest Common Divisor of n Numbers

Jeffrey Shallit
Department of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1
Canada
shallit@graceland.uwaterloo.ca
http://math.uwaterloo.ca/~shallit

This talk represents joint work with Charles Lam and Scott Vanstone.

An electronic copy of these slides can be found at
http://math.uwaterloo.ca/~shallit/talks.html

Greatest Common Divisor Algorithms

- Early ones, such as Euclid's (c. 300 B.C.E.) typically dealt with the case of two numbers.
- Analysis is often based on the number of **division steps** performed by the algorithm.
- Simon Jacob, a German reckoning master, apparently knew in the 16th century that the worst case of the Euclidean algorithm is when the inputs are consecutive Fibonacci numbers.
- In 1733, de Lagny discussed the "simplest" rational numbers having a continued fraction with a given length.
- In 1837, Léger stated without proof that the worst case of the Euclidean algorithm is when the inputs were (F_{n+2}, F_{n+1}).
- In 1841, Finck gave the first detailed, rigorous analysis of the Euclidean algorithm and concluded that the number of steps performed on input (u, v) with $u > v > 0$ is $\leq 2 \log_2 v + 1$.
- This was improved by Lamé in 1844 to $5\lfloor \log_{10} v \rfloor + 5$. Traditionally, Lamé has been given credit for the first analysis of an algorithm.

Analysis of Euclid's Algorithm

Lemma. If $u > v > 0$ causes Euclid's algorithm to perform n division steps, then $u \geq F_{n+2}$ and $v \geq F_{n+1}$, where F_n is the n'th Fibonacci number.

Proof. By induction on n.

Theorem. Let (u, v) with $u > v > 0$ be the lexicographically least pair causing Euclid's algorithm to perform n division steps. Then

$$(u, v) = (F_{n+1}, F_{n+1}).$$

Some GCD Algorithms are Difficult to Analyze

- For example, Purdy's algorithm:
- Assuming 2 does not divide both u and v:
- If u even, set $u \leftarrow u/2$.
- If v even, set $v \leftarrow v/2$.
- Otherwise set $(u, v) \leftarrow (\frac{u+v}{2}, \frac{u-v}{2})$
- Repeat these steps until one term is 0; the gcd is then the other term.
- No closed form known for the exact worst-case complexity

Another Difficult-to-Analyze GCD-type Algorithm

- Another GCD-type algorithm that has proved difficult to analyze:
- Let $a > b > 0$
- Set $b_0 = b$ and $b_{n+1} = a \bmod b_n$ until $b_r = 0$
- Define $P(a, b) = r$
- Then it is known that $P(a, b) = O(a^{1/3})$
- but probably $P(a, b) = O((\log a)^2)$
- I offer $ 25 for proof of this last conjecture.

de Rooij's Algorithm

$\mathrm{GCD}(L)$ $\{L$ is a list of $k \geq 2$ nonnegative integers $\}$
loop:
$u \leftarrow \text{Extract-Max}(L)$
$v \leftarrow \mathrm{Max}(L)$
if $v = 0$ then
return(u) and exit
else
$\mathrm{Insert}(L, u \bmod v)$
goto loop

- $\text{Extract-Max}(L)$ is a routine which removes and returns the largest element of L.
- If there is more than one largest element, any choice is permitted.
- $\mathrm{Max}(L)$ returns the largest element of L, but does not alter L.
- $\mathrm{Insert}(L, t)$ inserts t in list L; the particular position of the new element is unspecified.
- A heap can be used to perform these operations efficiently.

Naive Bit Complexity

For an integer $r \geq 0$ define

$$\lg r = \begin{cases} 1, & \text{if } r = 0; \\ 1 + \lfloor \log_2 r \rfloor, & \text{if } r > 0. \end{cases}$$

Thus $\lg r$ counts the number of bits required to represent r.

Bounding the Number of Steps

Theorem. Let $a_1 \geq a_2 \geq \cdots \geq a_k$. The number of division steps performed by de Rooij's algorithm GCD on input $(a_1, a_2, \ldots, a_n)$ is bounded by

$$k + (\lg a_2) + \sum_{2 \leq i \leq k} \lg a_i = O(\sum_{2 \leq i \leq k} \lg a_i).$$

Proof. First, do one division step. If a_1 and a_2 are the two largest elements, the algorithm replaces a_1 with $a_1 \bmod a_2 < a_2$.

Now let u, v be the two largest elements of the new list with $u \geq v$. If $v = 0$, then the algorithm performs no division steps. Otherwise write $u = qv + r$ with $0 \leq r < v$. If $q \geq 2$, then $r < v \leq u/2$. If $q = 1$, then $v > u/2$, and $r = u - v < u/2$. Hence $r < u/2$ in both cases. Provided $(u, v) \neq (1, 1)$, this means that $\lg r < \lg u$, and the algorithm continues with r in place of u. Thus, exclusive of the number of division steps where $(u, v) = (1, 1)$, after the first step the algorithm uses at most $(\lg a_2) + \Sigma_{2 \leq i \leq k} \lg a_i$ division steps. When the pair $(u, v) = (1, 1)$ is encountered, it is replaced by $(0, 1)$, and this can occur at most $k - 1$ times before the algorithm halts. The total number of division steps is therefore $\leq 1 + (k - 1) + (\lg a_2) + \Sigma_{2 \leq i \leq k} \lg a_i$. ∎

Bit Complexity of de Rooij's Algorithm

Given positive integers u, v, we can write $u = qv + r$ with $0 \leq r < v$ using $O((\lg q)(\lg v))$ bit operations.

Then the bit complexity for the division steps in de Rooij's algorithm is

$$O((\lg a_2)(\sum_{1 \leq i \leq k} \lg a_i))$$

on input $a_1 \geq a_2 \geq \cdots \geq a_k$.

More Detailed Worst-Case Analysis

- We'd like to find the "smallest" input that causes the algorithm to perform n division steps
- Not immediately clear what "smallest" means when there are two or more inputs
- For example, both $(17, 10, 5, 4)$ and $(14, 10, 7, 5)$ cause the algorithm to perform 9 division steps, and the sum of both 4-tuples is 36
- Also, $(89, 55, 0, 0)$ causes the algorithm to perform 9 division steps
- Lexicographically least k-tuple is a reasonable interpretation of "smallest"

Definition of Lexicographically Least

Given two k-tuples

$$A = (a_1, a_2, \ldots, a_k)$$

and

$$B = (b_1, b_2, \ldots, b_k)$$

with

$$a_1 \geq a_2 \geq \cdots \geq a_k$$

and

$$b_1 \geq b_2 \geq \cdots \geq b_k,$$

we say A is lexicographically less than B (and write $A < B$) if there exists an index i, $1 \leq i \leq k$ such that $a_j = b_j$ for $1 \leq j < i$ and $a_i < b_i$.

More Detailed Worst-Case Analysis

Definition. Let k, n be integers with $k \geq 2$ and $n \geq 0$. Define

$$A_k(n) = \begin{cases} 0, & \text{if } n = 0; \\ 1, & \text{if } 1 \leq n < k; \\ A_k(n-1) + A_k(n-k), & \text{if } n \geq k. \end{cases}$$

Here are the first few terms for the first few sequences:

$k \backslash n$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	0	1	1	2	3	5	8	13	21	34	55	89	144	233	377
3	0	1	1	1	2	3	4	6	9	13	19	28	41	60	88
4	0	1	1	1	1	2	3	4	5	7	10	14	19	26	36
5	0	1	1	1	1	1	2	3	4	5	6	8	11	15	20
6	0	1	1	1	1	1	1	2	3	4	5	6	7	9	12

Note that the case $k = 2$ corresponds to the ordinary Fibonacci sequence.

Theorem. Let $n \geq k \geq 2$. On input

$$A = (A_k(n+2), A_k(n+1), \ldots, A_k(n+3-k))$$

the algorithm GCD performs n division steps, and furthermore A is the lexicographically least k-tuple (with terms in decreasing order) with this property.

Numerical Evidence

- Numerical evidence suggests that the lexicographically least k-tuple consists of consecutive terms of a linear recurrence, e.g., for $k = 4$
- For example

  [insert table]

Form of Theorem

Note: we cannot get a lemma such as

"If $(a_1, a_2, \ldots, a_k)$ with $a_1 \geq a_2 \geq \cdots \geq a_k$ is any k-tuple causing the algorithm to perform n division steps then

$$\begin{aligned} a_1 &\geq A_k(n+2) \\ &\vdots \\ a_k &\geq A_k(n+3-k)" \end{aligned}$$

because we can always cause the algorithm to perform arbitrarily many steps with an input of the form $(a_1, a_2, 0, 0, \ldots, 0)$.

Sketch of Proof When $k = 4$

The key step is to prove the following

Lemma.

If (w, x, y, z) causes the algorithm to perform $n \geq 4$ division steps, then we have

(1) $w \geq A_4(n+2)$

(2) $x \geq A_4(n+1)$

(3) $w + z \geq A_4(n+3)$

(4) $w + y + z \geq A_4(n+4)$.

Once this result is proved, let

$$A = (A_4(n+2),\ A_4(n+1),\ A_4(n),\ A_4(n-1)).$$

Suppose $w \geq x \geq y \geq z$ and $B = (w, x, y, z)$ causes the algorithm to perform n division steps and $B \leq A$.

Then by (1) we know $w \geq A_4(n+2)$. But since $B \leq A$, $w \leq A_4(n+2)$. Hence $w = A_4(n+2)$.

Similarly, by (2), $x = A_4(n+1)$.

Now by (3) $w + z \geq A_4(n+3)$. But we know $w = A_4(n+2)$. Hence $z \geq A_4(n+3) - A_4(n+2) = A_4(n-1)$. This means that after one step of the algorithm on input (w, x, y, z) the new list is (in decreasing order) $(x, y, z, w \bmod x)$.

Sketch of Proof When $k = 4$ (continued)

(1) $w \geq A_4(n+2)$

(2) $x \geq A_4(n+1)$

(3) $w + z \geq A_4(n+3)$

(4) $w + y + z \geq A_4(n+4)$.

(continued)

We know this is the order because

$$\begin{aligned} w \bmod x &= A_4(n+2) \bmod A_4(n+1) \\ &= A_4(n-2) < A_4(n-1) \leq z. \end{aligned}$$

Now $y \geq A_4(n)$ by (3), and so $y = A_4(n)$.

Similarly, after one more step of the algorithm we get $(y, z, w \bmod x, x \bmod y)$. Hence $z \geq A_4(n-1)$ by (3), and so $z = A_4(n-1)$. It follows that $B = A$, so A is lexicographically least. ∎

The General Case

- must prove that every k-tuple causing the algorithm to perform n steps satisfies
- $a_1 \geq A_k(n+2)$
- $a_2 \geq A_k(n+1)$
- $a_1 + \Sigma_{k-i \leq j \leq k}\, a_j \geq A_k(n+i+3)$ for $0 \leq i \leq k-3$
- Proof is by induction on $n+k$.

For Further Reading

1. E. Bach and J. Shallit. *Algorithmic Number Theory.* The MIT Press, 1996.

2. P. de Rooij. Efficient exponentiation using precomputation and vector addition chains. In A. de Santis, editor, *Advances in Cryptology—EUROCRYPT '94 Proceedings*, Vol. 950 of *Lecture Notes in Computer Science*, pages 389–399. Springer-Verlag, 1995.

3. P. J. E. Finck. *Traité Élémentaire d'Arithmétique à l'Usage des Candidats aux Écoles Spéciales.* Derivaux, Strasbourg, 1841.

4. G. Lamé. Note sur la limite du nombre des divisions dans la recherche du plus grand commun diviseur entre deux nombres entiers. *C. R. Acad. Sci. Paris* **19** (1844), 867–870.

5. J. Shallit, Origins of the analysis of the Euclidean algorithm, *Historia Mathematica* **21** (1994), 401–419.

An electronic copy of these slides can be found at
`http://math.uwaterloo.ca/~shallit/talks.html`