A Straight Line Program Computing the Integer Greatest Common Divisor

SIDI MOHAMED SEDJELMACI[1]

[1]**Laboratoire d'Informatique de Paris Nord** (LIPN CNRS UMR 7030), 99 Avenue Jean-Baptiste Clément, 93 430 Villetaneuse, France.

ABSTRACT

While **NC** algorithms have been discovered for the basic arithmetic operations, the parallel complexity of some fundamental problems as integer gcd is still open, since first being raised in a paper of Cook [2]. Many authors attempt to design fast parallel integer GCD algorithms. Chor and Goldreich [1] proposed $O(n/\log n)_\epsilon$ parallel time with $O(n^{1+\epsilon})$ number of processors, for any $\epsilon > 0$. Sorenson [5] and the author [4] also suggest other parallel algorithms with the same parallel performance. Since then, no major improvements have been made.

In this paper, we propose a straight line program computing the integer GCD. It has polynomial size, but the outputs are polynomials with exponential degree. This work is a first attempt to improve the parallel complexity of integer GCD, thanks to Valiant et al. [3] contraction method, and, as far as we know, it is the first straight line program for computing the integer GCD. Throuhough this paper, we represent the input integers as formal strings of bits.

The main idea

- Find a simple Integer GCD without divisions or branching.
- Apply the contraction method of Valiant et al. (Theorem 2)

The Integer GCD Algorithm

Input: $x,\ y \geq 1$, two odds integers of n bits ;
Output: $\gcd(x, y)$;

$\begin{pmatrix} u \\ v \end{pmatrix} \leftarrow \begin{pmatrix} x \\ y \end{pmatrix}$;

While ($u \neq v$)

$\begin{pmatrix} u \\ v \end{pmatrix} \leftarrow \begin{pmatrix} v \\ (u+v)/2^t \end{pmatrix}$; s.t.: $(u+v)/2^t$ is odd.

EndWhile
Return u.

An example

Let $(x, y) = (35, 19)$ we obtain in turn:

$$\begin{pmatrix} 35 \\ 19 \end{pmatrix} \to \begin{pmatrix} 19 \\ 27 \end{pmatrix} \to \begin{pmatrix} 27 \\ 23 \end{pmatrix} \to \begin{pmatrix} 23 \\ 25 \end{pmatrix} \to \begin{pmatrix} 25 \\ 3 \end{pmatrix} \to \begin{pmatrix} 3 \\ 7 \end{pmatrix}$$
$$\to \begin{pmatrix} 7 \\ 5 \end{pmatrix} \to \begin{pmatrix} 5 \\ 3 \end{pmatrix} \to \begin{pmatrix} 3 \\ 1 \end{pmatrix} \to \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

The Fixed Point Lemma

The following is a tool to avoid conditional loops:

Lemma: Let F be a discrete function defined on vectors (or a set of ordered list) of n integers. We assume that, for a given such vector X of integers, $F(X)$ is computed by the following while loop (the repeat-until case is similar) :

$X \leftarrow X_0$;
While Condition(X) **do**
  $X \leftarrow F(X)$;
EndWhile
Return X.

If the final value X^* is a fixed point of F, i.e.: $F(X^*) = X^*$, after no more than $N_{max} = n^{O(1)}$ iterations, then the computation of X^* can be done in a free conditional loop way (N is any integer such that $N \geq N_{max}$):

$X \leftarrow X_0$;
For $i = 1$ to $N \geq N_{max}$ **do**
  $X \leftarrow F(X)$;
EndFor
Return X.

Proof: If the while loop terminates with the value X^* after N_1 iterations with $N_1 \leq N_{max}$, so is in the for loop. The for loop continues and gives, in the next iteration $N_1 + 1$, the value $F(X^*) = X^*$, since X^* is a fixed point of F, and so on until iteration N, hence the result.

Theorem 1

Let $u, v \geq 1$ be two odd integers of n bits, $n \geq 1$, such that $|u - v| = r2^t > 1$, with $r \geq 1$ odd, and $t \geq 1$. Let (u_k, v_k) be the sequence of consecutive values of u and v, obtained in the GCD algorithm. Then

i) $\max\{u_{t+1}, v_{t+1}\} < (3/4)\ \max\{u, v\}$.

ii) The algorithm terminates after at most $n^2/\log_2(4/3)$ iterations and returns $gcd(u, v)$.

iii) The **While** $u \neq v$ condition of the GCD algorithm can be replaced by **For** $i = 1$ **to** $3n^2$.

Proof:

First we observe that the transformation $(u, v) \leftarrow (v, (u+v)/2^t))$ preserves the GCD since u and v are both odds and $\gcd(v, (u+v)/2^t)) = \gcd(v, u+v) = \gcd(u, v)$.

i) The case $u = v$ is trivial. We assume that $u \neq v$ and $(u, v) = (v_0 + r2^t, v_0)$, with t even, the other cases $(u, v) = (u_0, u_0 + r2^t)$ and/or t odd are similar. We have

$$\begin{pmatrix} u_0 = v_0 + r2^t \\ v_0 \end{pmatrix} \to \begin{pmatrix} v_0 \\ v_0 + r2^{t-1} \end{pmatrix} \to \begin{pmatrix} v_0 + r2^{t-1} \\ v_0 + r2^{t-2} \end{pmatrix}$$
$$\cdots \to \begin{pmatrix} v_0 + r2^{t-2} + \ldots + 2r \\ 1/2^m\{v_0 + r2^{t-2} + \ldots r\} \end{pmatrix}.$$

After t iterations, the integer $2^m v_t = v_0 + r2^{t-2} + \ldots r$ is even. So $v_t < (1/2)u_0$ and $u_t < u_0$. Then, after $t+1$ iterations, we have $u_{t+1} = v_t < (1/2)u_0$ and $v_{t+1} \leq (1/2)(u_t + v_t) < (3/4)u_0$.

ii) Similarly, if $u_{t+1} = v_{t+1}$, the algorithm stops and returns the result: $u_{t+1} = \gcd(u, v)$. Otherwise $|u_{t+1} - v_{t+1}| = r_2 2^{t_2} > 1$, then we repeat the same process to the pair (u_{t+1}, v_{t+1}). Since $t_1 = t < n,\ t_2 < n, \ldots, t_p < n$, then after pn iterations we have $1 \leq \max\{u_{pn}, v_{pn}\} < (3/4)^p \max\{u, v\} < (3/4)^p 2^n$ and $p < \lfloor n/\log_2(4/3) \rfloor$.
Moreover, if $(u, v) = (ad, bd)$ with a, b, d odds and $\gcd(a, b) = 1$, then we have

$$\begin{pmatrix} ad \\ bd \end{pmatrix} \to \begin{pmatrix} bd \\ (\frac{a+b}{2^t})d \end{pmatrix} \to \cdots \to \begin{pmatrix} \gcd(a,b)\ .\ d \\ \gcd(a,b)\ .\ d \end{pmatrix} = \begin{pmatrix} d \\ d \end{pmatrix},$$

and the algorithm returns $\mathrm{GCD}(u, v)$.

iii) Let $F(u, v) = (v, (u+v)/2^t)$, such that $(u+v)/2^t)$ is odd. The output vector $X = (u, u)$ is a fixed point for F since $F(u, u) = (u, u)$. The Fixed Point Lemma applies.

Right shifts Without branching

We prove, in the following, how to compute rightshifts without division and branching. Let $A = (a_n, a_{n-1}, \cdots, a_1)$, and $a_{n+1} = 0)$:

Input: $A \geq 1$ represented by $A = (a_n, a_{n-1}, \cdots, a_1)$;
Output: An integer $A' \geq 1$ such that $A' = A/2^t$ is odd ;

For $k = 1$ **to** $n - 1$ **Do**
  $c = (1 - a_1)$;
  For i=1 **to** n **Do** $a_i = c\ .\ a_{i+1} + (1 - c)\ .\ a_i$
EndFor
Return A'=$(a_n, a_{n-1}, \cdots, a_1)$.

The `MakeOdd` procedure

A Staight Line Program Computing the Integer GCD

Consequently we obtain the following Straight Line Program computing the integer GCD:

Input: $x,\ y \geq 1$, two odds integers of n bits ;
Output: $\gcd(x, y)$;

$\begin{pmatrix} u \\ v \end{pmatrix} \leftarrow \begin{pmatrix} x \\ y \end{pmatrix}$;

For $i = 1$ **to** $3n^2$ **Do**

$\begin{pmatrix} u \\ v \end{pmatrix} \leftarrow \begin{pmatrix} v \\ \mathtt{MakeOdd}(u+v) \end{pmatrix}$;

EndFor
Return u.

Theorem 2 (Valiant-Skyum-Berkowitz-Rackoff [3])

Any sequential program computing a polynomial of degree $< d$ with C steps can be converted to a parallel program with parallel time $O(\ (\log d)\ (\log C + \log d)\ )$ using $O(\ (Cd)^\beta\ )$ processors, for an appropriate $\beta \geq 1$.

Theorem 3

The previous sraight line program has $O(n^4)$ size but the output polynomials have exponential degrees.

Proof:

It is obvious that the size is $O(n^4)$. Moreover, let us denote by $x_n, x_{n-1}, \ldots, x_1$ and $y_n, y_{n-1}, \ldots, y_1$, the bits of respectively integers x and y. Let $g_n, g_{n-1}, \ldots, g_1$ be the bits of $g = \gcd(x, y)$. Then each bit g_k of g, for $k = 1, 2, \ldots, n$, is a formal multivariate polynomial of the input bit variables x_i and y_j, i.e.:

$$\forall k = 1, 2, \ldots, n, \quad g_k \in \mathbb{Z}[x_n, x_{n-1}, \ldots, x_1;\ y_n, y_{n-1}, \ldots, y_1].$$

When a program contains n multiplications (equivalant AND) gates in sequence, the degree of the formal expression computed by it is 2^n in general. In particular the formal expression corresponding to the previous straight line program has exponential degree.

Another version:

Let $x_n, x_{n-1}, \ldots, x_1$ and $y_n, y_{n-1}, \ldots, y_1$, the bits of respectively two odd integers x and y. The right-shifting number t such that $(x+y)/2^t$ is odd can be computed straightforward by the function $t = \mathtt{ShiftNumber}(x, y)$ defined by:

$$t = 1 + \sum_{i=2}^{n} \Pi_{j=1}^{i}\ (x_j - y_j)^2,$$

and the `MakeOdd` procedure can be replaced by $\mathtt{RightShif}(\mathtt{A}, \mathtt{t})$ with the following specification:
Input: $A \geq 1$ represented by $A = (a_n, a_{n-1}, \cdots, a_1)$ and $0 \leq t \leq n-1$
Output: $A' = A/2^t$ represented by $A' = (0, \cdots, 0, a_n, \cdots, a_{t+2}, a_{t+1})$;

Then an alternative version of the integer GCD is:

Input: $x,\ y \geq 1$, two odds integers of n bits ;
Output: $\gcd(x, y)$;

$\begin{pmatrix} u \\ v \end{pmatrix} \leftarrow \begin{pmatrix} x \\ y \end{pmatrix}$;

For $i = 1$ **to** $3n^2$ **Do**
  $t = \mathtt{ShiftNumber}(u, v)$;
  $\begin{pmatrix} u \\ v \end{pmatrix} \leftarrow \begin{pmatrix} v \\ \mathtt{RightShift}(\mathtt{u+v}, \mathtt{t}) \end{pmatrix}$;

EndFor
Return u.

Conclusion

- This work is a first attempt to improve the parallel complexity of integer GCD, thanks to Valiant et al. [3] contraction method.
- Although the Valiant et al. [3] contraction method does not apply, because of the exponential degree of the output computed polynomials, our algorithm is, as far as we know, the first straight line program for computing the integer GCD.
- There are different ways to solve this issue:
  - Try to simplify the expression of our algorithm so that it gives rise to polynomials of small degrees.
  - Find other straight line programs computing the integer GCD.

References

[1] B. Chor and O. Goldreich, An improved parallel algorithm for integer GCD, Algorithmica, 5 (1990).

[2] S. Cook, A Taxonomy of Problems with Fast Parallel Algorithms, Information and Control, 64 (1985) 2–22.

[3] L.G. Valiant, S. Skyum, S. Berkowitz and C. Rackoff, Fast parallel computation of polynomials using few processors, SIAM J. Computing, 12-4, (1983), 641–644.

[4] S.M. Sedjelmaci, On A Parallel Lehmer-Euclid GCD Algorithm in Proc. of the International Symposium on Symbolic and Algebraic Computation (ISSAC'2001) 2001, 303-308.

[5] J. Sorenson, Two Fast GCD Algorithms, J. of Algorithms, (1994) 110–144.