Journal of Symbolic Computation 74 (2016) 425–474

Contents lists available at ScienceDirect

Journal of Symbolic Computation

www.elsevier.com/locate/jsc

Probabilistic analyses of the plain multiple gcd algorithm ☆

Valérie Berthé [a], Loïck Lhote [b], Brigitte Vallée [c]

[a] LIAFA, UMR CNRS 7089, Université Paris Diderot, France
[b] GREYC, UMR CNRS 6072, ENSICAEN & Université de Caen Normandie, France
[c] GREYC, UMR CNRS 6072, Université de Caen Normandie, France

ARTICLE INFO

Article history:
Received 4 December 2014
Accepted 10 July 2015
Available online 22 August 2015

Keywords:
Gcd algorithms
Analysis of algorithms
Analytic combinatorics
Generating functions
Transfer operator
Dynamical analysis
Limit laws
Beta law
Perron Formula
Landau Theorem

ABSTRACT

Among multiple gcd algorithms on polynomials as on integers, one of the most natural ones performs a sequence of $\ell - 1$ phases (ℓ is the number of inputs), with each of them being the Euclid algorithm on two entries. We present here a complete probabilistic analysis of this algorithm, by providing both the average-case and the distributional analysis, and by handling in parallel the integer and the polynomial cases, for polynomials with coefficients in a finite field. The main parameters under consideration are the number of iterations in each phase and the evolution of the size of the current gcd along the execution. Three phenomena are clearly emphasized through this analysis: the fact that almost all the computations are performed during the first phase, the great difference between the probabilistic behavior of the first phase compared to subsequent phases, and, as can be expected, the great similarity between the integer and the polynomial cases.

© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

Among arithmetic operations, on polynomials as on integers, the computation of gcd's plays a prominent role. It can even be considered as the fifth main one, with an impressive range of applica-

☆ This work was supported by Agence Nationale de la Recherche through three projects: Dyna3S ANR-13-BS02-0003 – ANR BOOLE (ANR 2009 BLAN 0011) – ANR MAGNUM (ANR 2010 BLAN 0204).

E-mail addresses: berthe@liafa.univ-paris-diderot.fr (V. Berthé), loick.lhote@ensicaen.fr (L. Lhote), brigitte.vallee@unicaen.fr (B. Vallée).

http://dx.doi.org/10.1016/j.jsc.2015.08.007
0747-7171/© 2015 Elsevier Ltd. All rights reserved.

tions, described for instance in von zur Gathen and Gerhard (2003). Let us just quote as an example the fact that in many symbolic computation systems, a large proportion of the time is devoted to the computation of gcd's on numbers, or on polynomials, in order to keep fractions under an irreducible form. Being able to measure the efficiency of a gcd algorithm, and to perform its analysis, is thus crucial.

The plain algorithm. Even for two entries, there is a wide variety of gcd algorithms; see, e.g., von zur Gathen and Gerhard (2003) or Vallée (2006). In this case, the Euclid algorithm plays a central role. Observe that there are many possible variants, in particular in the integer case with fast gcd algorithms; see Brent (1976), Schönhage (1971), Stehlé and Zimmermann (2004). For the general case of ℓ entries ($\ell \geq 2$), one of the most natural and basic algorithms consists in performing a succession of $\ell - 1$ phases, with each of them being the Euclid algorithm on two entries, as described in the book (Knuth, 1998).

More precisely, inputs are here either nonnegative integers or univariate polynomials over a finite field $\mathbb{F}_q$. In order to compute the gcd of ℓ inputs $x_1, \ldots, x_\ell$ ($\ell \geq 2$), we consider a sequence of $\ell - 1$ phases, that is, of $\ell - 1$ gcd computations on two entries. Let $y_1 := x_1$, then, for $k \in [2..\ell]$, one successively computes $y_k := \gcd(x_k, y_{k-1}) = \gcd(x_1, x_2, \ldots, x_k)$. The total gcd is $y_\ell := \gcd(x_1, x_2, \ldots, x_\ell)$, and it is obtained after $\ell - 1$ phases. We call it the *plain ℓ-Euclid algorithm*.

This is a straightforward algorithm, which cannot be easily extended for computing Bézout coefficients. We are interested in performing its analysis, making more precise and proving the observations made in Knuth (1998): "In most cases, the length of the partial gcd decreases rapidly during the first few phases of the calculation, and this will make the remainder of the computation quite fast". There are indeed inputs for which $\ell - 1$ phases are required, but, as the probability for two uniformly chosen inputs to be coprime is asymptotically $6/\pi^2$ for integers and $2q/(q-1)$ for polynomials, this algorithm is expected to require in average less steps. We thus do not claim that this naive algorithm is efficient.[1] However, a first step in analysis of algorithms consists in understanding and precisely analyzing even the simplest algorithms; such an analysis is not as trivial as it may first appear and then provides a basis of comparison for other algorithms of the same class.

State of the art. To the best of our knowledge, the plain ℓ-Euclid algorithm has not been yet analyzed. Its analysis was proposed as an exercise in the second edition of the "Art of Computer Programming" (Knuth, 1998), and quoted as a difficult one (HM48). However, the exercise disappears in the third edition.... The situation contrasts with the case $\ell = 2$, where the classical Euclid algorithm and all its main variants running on integers or on polynomials are now precisely analyzed. See Berthé and Nakada (2000), Friesen and Hensley (1996), Knopfmacher and Knopfmacher, (1988), Ma and von zur Gathen (1990) for analyses on polynomials; see Heilbronn (1969), Dixon (1970) for the first analyses on integers and Hensley (1994), Vallée (2006), Baladi and Vallée (2005), Lhote and Vallée (2008) for more recent ones, involving distributional analyses. Here, in all these previous studies, the size of an input is defined as the *maximum* size of its components; and the size is the degree of the polynomial, or the logarithm of the integer. The same probabilistic behavior appears in both cases (polynomials and integers): with respect to the input size, the mean number of iterations is linear, and the arithmetic complexity is quadratic. Furthermore, the distribution of the number of iterations is asymptotically Gaussian.

There exists also a probabilistic algorithm proposed in von zur Gathen and Shparlinski (2006) for computing gcd's. It replaces a gcd computation on ℓ entries by a unique gcd on two random linear combinations of the initial input. The approach is different: we perform here a probabilistic analysis of a deterministic algorithm where the distribution of the inputs is chosen a priori, whereas the algorithm developed in von zur Gathen and Shparlinski (2006) is probabilistic, designed as requiring few steps, and handles the worst-case. In Section 10, we return to the comparison between the two strategies, and make more precise the comparison done in von zur Gathen and Shparlinski (2006, Section 3).

[1] Nevertheless, we will show that this algorithm is in fact not as "stupid" as it seems to be....

Download English Version:

https://daneshyari.com/en/article/402966

Download Persian Version:

https://daneshyari.com/article/402966

Daneshyari.com