Average Bit-Complexity of Euclidean Algorithms

Ali Akhavi, Brigitte Vallée
GREYC, Université de Caen, F-14032 Caen (France)
Ali.Akhavi@info.unicaen.fr, Brigitte.Vallee@info.unicaen.fr

Abstract

We obtain new results regarding the precise average bit-complexity of five algorithms of a broad Euclidean type. We develop a general framework for analysis of algorithms, where the average-case complexity of an algorithm is seen to be related to the analytic behaviour in the complex plane of the set of elementary transformations determined by the algorithms. The methods rely on properties of transfer operators suitably adapted from dynamical systems theory and provide a unifying framework for the analysis of an entire class of gcd-like algorithms.

Keywords: Average-case Analysis of algorithms, Bit-Complexity, Euclidean Algorithms, Dynamical Systems, Ruelle operators, Generating Functions, Dirichlet Series, Tauberian Theorems.

1 Introduction

Motivations. Euclid's algorithm was analysed first in the worst case in 1733 by de Lagny, then in the average-case around 1969 independently by Heilbronn [12] and Dixon [6], and finally in distribution by Hensley [13] who proved in 1994 that the Euclidean algorithm has Gaussian behaviour. The first methods used range from combinatorial (de Lagny, Heilbronn) to probabilistic (Dixon). In parallel, studies by Lévy, Khinchin, Kuzmin and Wirsing had established the metric theory of continued fractions by means of a specific density transformer. The more recent works rely for a good deal on *transfer operators*, a far-reaching generalization of density transformers, originally introduced by Ruelle [19, 20] in connection with the thermodynamic formalism and dynamical systems theory [1]. Examples are Mayer's studies on the continued fraction transformation [16], Hensley's work [13] and several papers of Vallée [22, 23, 24, 25].
All the previous analyses deal with the number of arithmetical operations performed during the execution of the algorithm. In this paper, we provide new analyses that study the precise average bit-complexity of a class of Euclidean algorithms.
We consider here five algorithms that are all classical variations of the Euclidean algorithm and are called *Classical* (G), *By-Excess* (L), *Centered* (K), *Subtractive* (T) and *Binary* (B). The complexity of these algorithms (in terms of the number of arithmetical operations to be performed) is now well–known: The two classical algorithms (G) and (K) have been analysed by Heilbronn, Dixon and Rieger [18]. The Subtractive algorithm (T) was studied by Knuth and Yao [27], and Vardi [26] analysed the By-Excess Algorithm (L) by comparing it to the Subtractive Algorithm. Brent [3] and Vallée [24] have analysed the Binary algorithm.

Methods. Our approach is a refinement of some methods that have been already used for instance in [4, 9, 24, 25]: it consists in viewing an algorithm of the broad gcd type as a dynamical system, where each iterative step is a linear fractional transformation (LFT) of the form $z \to (az+b)/(cz+d)$. A specific set of transformations is then associated to each algorithm. It already appears from previous treatments that the computational complexity of an algorithm is in fact dictated by the collective dynamics of its associated set of transformations. More precisely, two factors intervene: (i) the characteristics of the LFT's in the complex domain; (ii) their *contraction properties*, notably near fixed points.

A previous work [25] has described a classification of gcd-like algorithms in terms of the average number of arithmetical operations: some of them are "fast", that is, of logarithmic complexity $\Theta(\log N)$, while others are "slow", that is, of the "log-squared" type $\Theta(\log^2 N)$. It was established there that strong contraction properties of the elementary transformations that build up a gcd-like algorithm entail logarithmic cost, while the presence of an indifferent fixed-point leads to log-squared behaviour.

It is not a priori clear whether this previous classification between "fast" algorithms and "slow" algorithms extends to the average bit–complexity; The reason is that, even if fast algorithms perform less iterations, each iteration is often more complex than it is in the case of slow algorithms. In this paper, we prove that, in terms of the average-bit-complexity, fast algorithms are of "log-squared" type $\Theta(\log^2 N)$ while slow ones are of "log-cubed" type $\Theta(\log^3 N)$. Our approach also precisely determines the constants that are closely related to the entropy of the associated dynamical system: they are computable numbers but they are not always related to classical constants of analysis. Our method also opens access to characteristics of the distribution of bit-complexity costs, including information on moments: the fast algorithms appear to have concentration of distribution —the cost converges in probability to its mean— while the slow ones exhibit an extremely large dispersion of costs.

Technically, this paper relies on a description of relevant parameters by means of generating functions, a by now common tool in the average-case of algorithms [7, 8]. As is usual in number theory contexts, the generating functions are Dirichlet series. They are first proved to be algebraically related to specific operators that encapsulate all the important informations relative to the "dynamics" of the algorithm. Their analytical properties depend on spectral properties of the operators, most notably the existence of a "spectral gap" that separates the dominant eigenvalue from the remainder of the spectrum. This determines the singularities of the Dirichlet series of costs. The asymptotic extraction of coefficients is then achieved by means of Tauberian theorems, one of the many ways to derive the prime number theorem. Average bit–complexity estimates finally result. The main thread of the paper is thus adequately summarized by the chain:

$$\begin{array}{c}\text{Euclidean algorithm} \rightsquigarrow \text{Associated transformations} \rightsquigarrow \text{Transfer operator}\\ \rightsquigarrow \text{Dirichlet series of costs} \rightsquigarrow \text{Tauberian inversion} \rightsquigarrow \text{Average-case complexity.}\end{array}$$

This chain then leads to effective and simple criteria for distinguishing slow algorithms from fast ones and for establishing concentration of distribution, etc.

Results and plan of the paper. Section 3 is the central technical section of the paper. There, we develop the line of attack outlined earlier and introduce successively Dirichlet generating functions, transfer operators of the Ruelle type, and the basic elements of Tauberian theory that are adequate for our purposes. The main results of this section are summarized in Theorem 1 that describes the singularities of generating functions of bit–costs and in Theorem 2 that implies a general criterion for log-squared versus log-cubed behaviour, while providing a framework for higher moment analyses.

In Section 4, we return to our five favorite algorithms. The corresponding analyses are summarized in Theorems 3 and 4 where we list our main results that fall as natural consequences of the present framework. It results from the analysis (Theorem 3) that the algorithms of the Fast Class –the Classical Algorithm (G), the Centered Algorithm (K), and the Binary algorithm (B)– have their average bit-complexity of the form

$$B_N(\mathcal{H}) \sim A(\mathcal{H}) \log_2^2 N, \qquad \text{with } \mathcal{H} \in \{\mathcal{G}, \mathcal{K}, \mathcal{B}\},$$

when applied to random integers less than N. Each of the three constants $A(\mathcal{G}), A(\mathcal{K}), A(\mathcal{B})$ is a product of two constants: the first one is a constant à la Lévy and is effectively characterized as the inverse of the entropy of the associated dynamical system, while the second one is a constant à la Khinchin. The constants related to the two classical algorithms are explicit and easily obtained,

$$A(\mathcal{G}) = \frac{6 \log^2 2}{\pi^2} \; [2 + \frac{1}{\log 2} \log \prod_{k=0}^{\infty} (1 + \frac{1}{2^k})],$$

$$A(\mathcal{K}) = \frac{6\log\phi\log 2}{\pi^2}[3 + \frac{\log 2}{\log\phi} + \frac{1}{\log\phi}\log\prod_{k=3}^{\infty}\frac{(2^k-1)\phi^2+2\phi}{(2^k-1)\phi^2-2}].$$

The constant relative to the Binary Algorithm is expressed in terms of the invariant measure $\psi_2(t)dt$ and its distribution function $F_2(t)$ (that are not explicit in this case) as

$$A(\mathcal{B}) = \frac{2\log 2}{\pi^2\,\psi_2(1)}\;[1 + \sum_{m\quad \text{odd}\geq 1}\frac{1}{2^{\ell(m)}}\,F_2(\frac{1}{m})],$$

where $\ell(m)$ denotes the binary length of integer m. Exact computations (for the first two algorithms) or various batches of a few thousands simulations on numbers of order of 10^{100} (for the Binary Algorithm) suggest numerical values for the three constants

$$A(\mathcal{G}) \simeq 1.24237 \qquad A(\mathcal{K}) \simeq 1.12655 \qquad A(\mathcal{B}) \simeq 0.7.$$

These values prove the efficiency of fast Euclidean Algorithms, compared to naive multiplication of numbers whose average bit-complexity is $\log_2^2 N$ on integers less than N.
Theorem 4 proves that the algorithms of the Slow Class – the By-Excess Algorithm (L) and the Subtractive Algorithm (T)– have an average bit-complexity of the log–cubed type,

$$B_N(\mathcal{H}) \sim A(\mathcal{H})\;\log_2^3 N \qquad \text{with } \mathcal{H} \in \{\mathcal{L}, \mathcal{T}\},$$

and the constants satisfy

$$A(\mathcal{T}) = \frac{2\log^2 2}{\pi^2}, \qquad\qquad A(\mathcal{L}) = \frac{\log^2 2}{\pi^2}.$$

Finally, Theorem 5 provides probabilistic characterizations of the distribution of the costs: in particular, we prove that concentration of distribution holds in the case of the the three fast algorithms (G, K, B). On the contrary, slow algorithms have a great dispersion of costs.

2 Five variations of the Euclidean algorithm

We present here the five algorithms to be analysed; the first three use divisions, while the last two use only simpler operations, as subtractions and/or right shifts.

2.1. Euclidean Algorithms with divisions. There are two divisions between u and v ($v > u$), that produce a positive remainder r such that $0 \leq r < u$: the classical division (by-default) of the form $v = mu+r$, and the division by-excess, of the form $v = mu - r$. The centered division between u and v ($v > u$), of the form $v = mu + \varepsilon r$, with $\varepsilon = \pm 1$ produces a positive remainder r such that $0 \leq r < u/2$. There are three Euclidean algorithms associated to each type of division, respectively called the Classical Algorithm (G), the By-Excess Algorithm (L), and the Classical Centered Algorithm (K).
We denote by $\ell(x)$ the number of bits of the positive integer x. Then, the bit–cost of a division step, of the form $v = mu + \varepsilon r$ is equal to $\ell(u) \times \ell(m)$. It is followed by exchanges which involve numbers u and r, so that the total cost of a step is $\ell(u) \times \ell(m) + \ell(u) + \ell(r)$. In the case of the centered division, there is possibly a supplementary subtraction (in the case when $\varepsilon = -1$) in order to obtain a remainder in the interval $[0, (u/2)]$.

2.2. Euclidean Algorithms without divisions. On the other side, there are two algorithms where no divisions are performed, the Subtractive Algorithm (T) and the Binary Algorithm (B). The Subtractive Algorithm (T) uses only subtractions, since it replaces the classical division $v = mu+r$ by exactly a sequence of m subtractions of the form $v := v - u$. The Binary Algorithm (B) uses only subtractions and right shifts, since it performs a sequence of operations of the form $v := (v-u)/2^b$, where b is the dyadic valuation of $v-u$, denoted by $b := \text{Val}_2(v-u)$, and defined as the largest exponent b such that 2^b divides $v-u$.

These two algorithms have the same structure, since they have two nested loops: Each external loop corresponds to an exchange. Between two exchanges, there is a sequence of (internal) iterations that constitutes one external step.

Without-Division Euclidean Algorithm (u, v)

While $u \neq v$ **do**
  While $u < v$ **do**
    If Algo = (T) then $b = 0$;
    If Algo = (B) then $b := \text{Val}_2(v-u)$;
    $v := (v-u)/2^b$;
  Exchange u and v;
Output: u (or v).

Each internal step consists in subtractions and (possible) shifts and a sequence of internal steps can be written as

$$v = u + 2^{b_1} v_1, \qquad v_1 = u + 2^{b_2} v_2, \qquad v_2 = u + 2^{b_3} v_3, \qquad \ldots \qquad v_{\ell-1} = u + 2^{b_\ell} v_\ell, \tag{1}$$

Here v_ℓ is strictly less than u, and plays the rôle of a remainder r, so that the result of the sequence (1) is a decomposition of the form $v = mu + 2^s r$, with m odd, $m < 2^s$ and $r < u$ and constitutes an external step. The number of internal steps in (1) equals $b(m)$, where $b(x)$ denotes the number of ones in the binary expansion of x.
The cost of a subtractive step $v = u + (v-u)$ is equal to $\ell(v)$. The cost of a right shift $v := v/2^b$ is equal to $\ell(v)$. Then the bit–cost of a sequence of internal steps whose result is a decomposition $v = mu + dr$ equals $\ell(v) \times m$ for the Subtractive algorithm (T) and equals $\ell(v) \times b(m)$ for the Binary Algorithm (B). It is followed by an exchange, so that the total cost of an external step is $\ell(v) \times (m+2)$ for the Subtractive algorithm (T) and equals $\ell(v) \times (b(m)+2)$ for the Binary Algorithm (B).

2.3. The sets of linear fractional transformations. When given an input (u_1, u_0), $(u_1 \leq u_0)$, each of the five algorithms performs k (external) steps of the form

$$u_0 = m_1 u_1 + d_1 u_2, \qquad u_1 = m_2 u_2 + d_2 u_3, \qquad \ldots u_{k-1} = m_k u_k + d_k u_{k+1}$$

and decomposes the rational $x := (u_1/u_0)$ as $(u_1/u_0) = h_1 \circ h_2 \circ \ldots \circ h_k(a)$, where the h_i's are linear fractional transformations (LFT) of the form $h_i = h_{[m_i, d_i]}$ with $h_{[m,d]}(x) = 1/(m + dx)$ and $a := (u_{k+1}/u_k)$ is the last value of the rational.
The precise form of the possible LFT's depends on the algorithm; there may exists a special set of LFT's in the final step $(\mathcal{F})$. However, all the other steps use the same set of LFT's, that we call the generic set.
The value a equals 1 for the By-Excess Algorithm (L) and the Binary Algorithm (B), and equals 0 for the other three algorithms. For the centered algorithm (K), the rational x belongs to $\mathcal{I} = [0, 1/2]$. In the other four cases, the rational x belongs to $\mathcal{I} = [0, 1]$. Finally, the rational inputs of each algorithm belong to the basic interval $\mathcal{I} = [0, \rho]$ with $\rho = 1$ or $\rho = 1/2$. For the first four algorithms, the valid inputs are all the rationals of $\mathcal{I}$, while the valid inputs of the last algorithm are only the odd rationals of $\mathcal{I}$. The variable valid has two possible values { all, odd}, and finally, the type of the algorithm is defined as the triple $(\rho, \text{ valid}, a)$.

In all the cases, when performing k (external) steps on the input (u_1, u_0), the bit-cost $C(u_1, u_0)$ of the algorithm is a sum of k terms, the i-th term representing the cost of the i-th (external) step and being a product of two factors; the first factor involves the binary length $\ell(u_j)$ of integer u_j (with j possibly equal

Alg., Type	*Division*	*Set $\mathcal{H}$ of LFT's*	*Final Set $\mathcal{F}$*	*Cost of the LFT.*
(G) (1, all, 0)	$v = mu + r$ $0 \leq r < u$	$\{\frac{1}{m+x}, m \geq 1\}$	$m \geq 2$	$\ell(m) + 2$
(L) (1, all, 1)	$v = mu - r$ $0 \leq r < u$	$\{\frac{1}{m-x}, m \geq 2\}$	$m \geq 3$	$\ell(m) + 2$
(K) ($\frac{1}{2}$, all, 0)	$v = mu + \varepsilon r$ $m \geq 2, \varepsilon = \pm 1, (m, \varepsilon) \neq (2, -1)$ $0 \leq r < \frac{u}{2}$	$\{\frac{1}{m+\varepsilon x}, m \geq 2, \varepsilon = \pm 1\}$	$\varepsilon = +1$	$\ell(m) + 2 + \frac{1-\varepsilon}{2}$
(T) (1, all, 0)	$v = mu + r$ $0 \leq r < u$	$\{\frac{1}{m+x}, m \geq 1\}$	$m \geq 2$	$m + 2$
(B) (1, odd, 1)	$v = mu + 2^s r$, m odd, $m < 2^r, r < u$	$\{\frac{1}{m+2^s x}, m \text{ odd}, \ m < 2^s\}$	$\mathcal{F} = \mathcal{H}$	$b(m) + 2$

Figure 1: The five Euclidean algorithms.

to $i-1, i$ or $i+1$ according to the precise algorithm), while the second one involves a cost relative to the i-th LFT to be performed, of the form $c(h_i)$. In the sequel, we will see that we can replace the length $\ell(u)$ of integer u by its logarithm $\log_2(u)$ in base 2 and always consider $\log_2(u_i)$ as the first factor to be studied. In contrast, we have to work with the exact cost due to the LFT. Finally, for all the five algorithms, the studied bit-cost $C(u_1, u_0)$ of the algorithm on the input (u_1, u_0) will be of the form

$$C(u_1, u_0) = \sum_{i=1}^{k} \log_2(u_i) \times c(h_i). \tag{2}$$

Here, the array of Figure 1 describes the precise form of the divisions, the generic set $\mathcal{H}$ of associated LFT's, the final set $\mathcal{F}$ and the cost of the LFT's.

3 Generating functions, dynamical operators and Tauberian Theorems

Here, we describe the general tools for analysing bit-complexities of algorithms of the Euclidean type. We first introduce the Dirichlet generating functions relative to the bit-cost of the algorithms, so that the average bit–complexity involves partial sums of coefficients of these Dirichlet series. Tauberian Theorems are a classical tool that transfers analytical behaviour of Dirichlet series near their singularities into asymptotic behaviour of their coefficients. Then, when viewing the algorithm as a dynamical system, we relate generating functions of bit–cost to the Ruelle operator associated to the algorithm, so that we can easily describe the singularities of generating functions.

3.1. Generating functions. The basic interval is $\mathcal{I} := [0, \rho]$, where ρ has two possible values $\rho = 1$ and $\rho = 1/2$. We consider the following sets

$$\widetilde{\Omega} := \{(u, v);\ u, v \text{ valid}, \ \frac{u}{v} \in \mathcal{I}\}, \quad \Omega := \{(u, v);\ u, v \text{ valid}, \gcd(u, v) = 1, \frac{u}{v} \in \mathcal{I}\},$$

$$\widetilde{\Omega}_N := \{(u, v) \in \widetilde{\Omega}, v \leq N\}, \qquad \Omega_N := \{(u, v) \in \Omega, v \leq N\},$$

for the possible inputs of an algorithm. We denote by $C(u, v)$ the bit-complexity cost of the algorithm on the input (u, v). We wish to study the average bit-complexity B_N of the algorithm on Ω_N and the average bit-complexity $\widetilde{B}_N$ of the algorithm on $\widetilde{\Omega}_N$, and evaluate the asymptotic behaviour (for $N \to \infty$) of these costs. These quantities satisfy

$$B_N = \frac{C_N}{|\Omega_N|}, \qquad \widetilde{B}_N = \frac{\widetilde{C}_N}{|\widetilde{\Omega}_N|}, \qquad \text{with} \qquad C_N := \sum_{(u,v)\in\Omega_N} C(u,v) \qquad \widetilde{C}_N := \sum_{(u,v)\in\widetilde{\Omega}_N} C(u,v). \tag{3}$$

The following Dirichlet generating functions of costs,

$$F(s) := \sum_{(u,v)\in\Omega} \frac{1}{v^s}, \qquad \widetilde{F}(s) = \sum_{(u,v)\in\widetilde{\Omega}} \frac{1}{v^s}, \qquad G(s) := \sum_{(u,v)\in\Omega} \frac{1}{v^s} C(u,v), \qquad \widetilde{G}(s) = \sum_{(u,v)\in\widetilde{\Omega}} \frac{1}{v^s} C(u,v), \tag{4}$$

are of the form

$$F(s) = \sum_{n\geq 1} \frac{a_n}{n^s}, \qquad \widetilde{F}(s) = \sum_{n\geq 1} \frac{\widetilde{a}_n}{n^s}, \qquad G(s) = \sum_{n\geq 1} \frac{c_n}{n^s}, \qquad \widetilde{G}(s) = \sum_{n\geq 1} \frac{\widetilde{c}_n}{n^s},$$

where a_n, $\widetilde{a}_n$ are the number of pairs (u,v) of Ω or $\widetilde{\Omega}$ with fixed $v = n$, and c_n, $\widetilde{c}_n$ are the cumulative costs on pairs (u,v) of Ω or $\widetilde{\Omega}$ with fixed $v = n$, so that the average costs in (3) to be studied

$$B_N = \frac{\sum_{n\leq N} c_n}{\sum_{n\leq N} a_n} \qquad \widetilde{B}_N = \frac{\sum_{n\leq N} \widetilde{c}_n}{\sum_{n\leq N} \widetilde{a}_n}$$

are exactly the quotient of partial sums of the coefficients of the Dirichlet series $F(s), \widetilde{F}(s), G(s), \widetilde{G}(s)$ defined in (4).

3.2. Tauberian Theorems. Thus, the asymptotic evaluation of B_N, $\widetilde{B}_N$ (for $N \to \infty$) is possible if we can apply the following Tauberian theorem [5], [21] to the Dirichlet series $F(s)$, $\widetilde{F}(s)$, $G(s)$, $\widetilde{G}(s)$.

Tauberian Theorem. [Delange] *Let $F(s)$ be a Dirichlet series with non negative coefficients such that $F(s)$ converges for $\Re(s) > \sigma > 0$. Assume that*
(i) $F(s)$ is analytic on $\Re(s) = \sigma, s \neq \sigma$, and
(ii) for some $\gamma \geq 0$, one has $F(s) = A(s)(s-\sigma)^{-\gamma-1} + C(s)$, where A, C are analytic at σ, with $A(\sigma) \neq 0$.
Then, as $N \to \infty$,

$$\sum_{n\leq N} a_n = \frac{A(\sigma)}{\sigma\Gamma(\gamma+1)}\; N^\sigma \log^\gamma N \quad [1+\varepsilon(N)], \quad \varepsilon(N) \to 0.$$

In the remainder of the paper, we show that the Tauberian Theorem applies to the Dirichlet series F, G defined in (4) with $\sigma = 2$. For $F(s)$, it applies with $\gamma = 0$. For $G(s)$, it applies with $\gamma = 2$ or $\gamma = 3$. For the slow algorithms, γ equals 3, and the average bit-complexity will be of order $\log^3 N$. For the fast algorithms, γ equals 2, and the average bit-complexity will be of order $\log^2 N$.
We first examine the case of functions $F(s), \widetilde{F}(s)$ that are closely linked to the Riemann series of valid numbers $\widehat{\zeta}$, defined as $\widehat{\zeta}(s) := \sum_{v \text{ valid}} v^{-s}$ via the equalities

$$\widetilde{F}(s) = \widehat{\zeta}(s)\, F(s), \qquad \widetilde{F}(s) = \rho\, \widehat{\zeta}(s-1) \quad \text{if valid= all}, \qquad \widetilde{F}(s) = \frac{\rho}{2}\, \widehat{\zeta}(s-1) \quad \text{if valid= odd}.$$

Since $\widehat{\zeta}$ is itself easily related to the classical Zeta function, it is then clear that the Tauberian Theorem applies to $F(s)$ and $\widetilde{F}(s)$, with $\sigma = 2$ and $\gamma = 0$. More precisely, at $s = 2$, one has:

$$F(s) \simeq \frac{2\rho}{\pi^2}\frac{1}{s-2} \quad \text{if valid= odd} \quad \text{and} \quad F(s) \simeq \frac{6\rho}{\pi^2}\frac{1}{s-2} \quad \text{if valid= all}. \tag{5}$$

In the following, we obtain alternative expressions for Dirichlet series $F(s), \widetilde{F}(s), G(s), \widetilde{G}(s)$ which involve suitable Ruelle operators.

3.3. Algebraic properties of Ruelle operators. The Ruelle operator $\mathbf{R}_{s,h}$ relative to a LFT h depends on a complex parameter s and is defined as

$$\mathbf{R}_{s,h}[f](x) := \frac{1}{D[h](x)^s}\ f \circ h(x), \tag{6}$$

where $D[h]$ denotes the denominator of the linear fractional transformation (LFT) h, defined for $h(x) = (ax+b)/(cx+d)$ with a, b, c, d coprime integers by $D[h](x) := |cx+d| = |\det h|^{1/2}\, |h'(x)|^{-1/2}$.
When given a cost function c relative to the LFT h, one defines another Ruelle operator relative to h,

$$\mathbf{R}^{[c]}_{s,h}[f](x) := \frac{c(h)}{D[h](x)^s}\ f \circ h(x). \tag{7}$$

Now, when given an algorithm and a set $\mathcal{H}$ of LFT's used by the algorithm, the Ruelle operators relative to $\mathcal{H}$ are defined by

$$\mathbf{H}_s := \sum_{h\in\mathcal{H}} \mathbf{R}_{s,h}, \qquad \mathbf{H}^{[c]}_s := \sum_{h\in\mathcal{H}} \mathbf{R}^{[c]}_{s,h}. \tag{8}$$

The multiplicative property of denominator D, i.e., $D[h\circ g](x) = D[h](g(x))\, D[g](x)$ is translated by a multiplicative property on Ruelle operators: When given two LFT's, h and g, the Ruelle operator $\mathbf{R}_{s,h\circ g}$ associated to the LFT $h\circ g$ is exactly the operator $\mathbf{R}_{s,g}\circ\mathbf{R}_{s,h}$. More generally, when given two sets of LFT's, $\mathcal{L}$ and $\mathcal{K}$ and their Ruelle operators $\mathbf{K}_s, \mathbf{L}_s$, the set $\mathcal{LK}$ is formed of all $h\circ g$ with $h\in\mathcal{L}$ and $g\in\mathcal{K}$, and the Ruelle operator relative to the set $\mathcal{LK}$ is exactly the operator $\mathbf{K}_s\circ\mathbf{L}_s$. In particular, the Ruelle operator relative to the set $\mathcal{H}^\star := \cup_{k\geq 0}\mathcal{H}^k$ is exactly $\sum_{k\geq 0}\mathbf{H}^k_s = (I-\mathbf{H}_s)^{-1}$. It is the quasi–inverse of the Ruelle operator $\mathbf{H}_s$ associated to the set $\mathcal{H}$.

3.4. Ruelle operators and generating functions. We show now how the Ruelle operators associated to the algorithms intervene in the evaluation of the generating functions of costs $G(s), \widetilde{G}(s)$.
We consider here a Euclidean Algorithm and its set of LFT's $\mathcal{H}$ together with its final set $\mathcal{F}$ defined in Fig.1. The Ruelle operators $\mathbf{H}_s, \mathbf{H}^{[c]}_s, \mathbf{F}_s, \mathbf{F}^{[c]}_s$ relative to $\mathcal{H}$ or $\mathcal{F}$ and defined in (6, 7, 8) will play a central rôle in the analysis.
Consider first an input (u_1, u_0) of Ω. (Then u_0 and u_1 are coprime). An execution of the algorithm on the input (u_1, u_0) performs k external steps of the form

$$u_0 = m_1 u_1 + d_1 u_2, \qquad u_1 = m_2 u_2 + d_2 u_3, \qquad \ldots u_{k-1} = m_k u_k + d_k u_{k+1}$$

and decomposes the rational (u_1/u_0) as $(u_1/u_0) = h_1\circ h_2\circ\ldots\circ h_k(a)$, where the h_i's are elements of $\mathcal{H}$ (for $i\leq k-1$) or elements of $\mathcal{F}$ (for $i=k$), and a is the last value of the rational. When given an index $i, 1\leq i\leq k$, we consider now three different parts of the LFT $h = h_1\circ h_2\circ\ldots\circ h_k$: the beginning part $b_i(h) := h_1\circ h_2\circ\ldots\circ h_{i-1}$, the ending part $e_i(h) := h_{i+1}\circ h_{i+2}\circ\ldots\circ h_k$, and finally the i-th component h_i. Then the sequence of the rationals (u_{i+1}/u_i) is defined from the relations

$$\frac{u_{i+1}}{u_i} = h_{i+1}\circ h_{i+2}\circ\ldots\circ h_k(a) = e_i(h)(a),$$

and, since u_i and u_{i+1} are coprime, the equality $D[e_i(h)](a) = u_i$ holds.
When given an operator $\mathbf{L}_s$ that depends on parameter s, the operator $\Delta\mathbf{L}_s$ is defined by

$$\Delta\mathbf{L}_s := \frac{-1}{\log 2}\frac{d}{ds}\mathbf{L}_s.$$

When applied to $\mathbf{R}_{s,h}$ defined in (6), it is well-suited to the problem since it produces at the numerator the logarithm $\log_2 D[h](x)$. We then introduce our main operator

$$\mathbf{S}_{s,h} := \sum_{i=1}^{k-1} \Delta\mathbf{R}_{s,e_i(h)}\ \circ\ \mathbf{R}^{[c]}_{s,h_i}\ \circ\ \mathbf{R}_{s,b_i(h)},$$

and we claim that, when applied to function $f=1$ and point $x=a$, this operator generates the cost $C(u_1,u_0)$ of the algorithm on input (u_1,u_0), defined in (2),

$$\mathbf{S}_{s,h}[1](a)=\sum_{i=1}^{k-1}\frac{1}{u_0^s}\,c(h_i)\times\log_2 u_i=\sum_{i=1}^{k}\frac{1}{u_0^s}\,c(h_i)\times\log_2 u_i=\frac{1}{u_0^s}C(u_1,u_0).$$

(we replace the upper index k by $k-1$ thanks to equality $u_k=1$ that holds since u_k is just the gcd of u_0 and u_1). Then, when (u_1,u_0) is a general element of Ω, the LFT h is a general element of set $\mathcal{H}^\star\mathcal{F}$, so that we obtain alternative expression of our main Dirichlet series F,G defined in (4)

$$F(s):=\sum_{(u_1,u_0)\in\Omega}\frac{1}{u_0^s}=\sum_{h\in\mathcal{H}^\star\mathcal{F}}\mathbf{R}_{s,h}[1](a)\qquad\text{and}\qquad G(s):=\sum_{(u_1,u_0)\in\Omega}\frac{1}{u_0^s}\,C(u_1,u_0)=\sum_{h\in\mathcal{H}^\star\mathcal{F}}\mathbf{S}_{s,h}[1](a).$$

When index i varies in $[1..k-1]$, beginning part $b_i(h)$ is a general element of $\mathcal{H}^\star$, ending part $e_i(h)$ is a general element of $\mathcal{H}^\star\mathcal{F}$, while the i-th component is a general element of $\mathcal{H}$, so that

$$\sum_{h\in\mathcal{H}^\star\mathcal{F}}\mathbf{S}_{s,h}=\Delta\left[\mathbf{F}_s\circ(I-\mathbf{H}_s)^{-1}\right]\;\circ\;\mathbf{H}_s^{[c]}\;\circ\;(I-\mathbf{H}_s)^{-1},\qquad\text{and}\qquad\sum_{h\in\mathcal{H}^\star\mathcal{F}}\mathbf{R}_{s,h}=\mathbf{F}_s\circ(I-\mathbf{H}_s)^{-1},$$

where $\mathbf{H}_s,\mathbf{F}_s$ are the Ruelle operators relative to the sets $\mathcal{H},\mathcal{F}$ used by the algorithm. We finally deduce an expression for the Dirichlet series $F(s),G(s)$,

$$F(s)=\mathbf{F}_s\circ(I-\mathbf{H}_s)^{-1}\;[1](a)\tag{9}$$

$$G(s)=\left[\Delta\mathbf{F}_s+\mathbf{F}_s\circ(I-\mathbf{H}_s)^{-1}\circ\Delta\mathbf{H}_s\right]\;\circ\;(I-\mathbf{H}_s)^{-1}\circ\mathbf{H}_s^{[c]}\circ(I-\mathbf{H}_s)^{-1}\;[1](a).\tag{10}$$

Consider now an input $(\widetilde{u}_1,\widetilde{u}_0)$ of $\widetilde{\Omega}$. Then there exists $\lambda\geq 1$ such that $(\widetilde{u}_1,\widetilde{u}_0)=\lambda(u_1,u_0)$ with $(u_1,u_0)\in\Omega$. The Euclidean algorithm deals now with the integers λu_i instead of the integers u_i and uses exactly the same LFT's as it uses for the input (u_1,u_0), so that the bit–cost $C(\widetilde{u}_1,\widetilde{u}_0)$ is easily related to $C(u_1,u_0)$,

$$C(\widetilde{u}_1,\widetilde{u}_0)=\sum_{i=1}^{k}\log_2(\lambda u_i)\times c(h_i)=C(u_1,u_0)+(\log_2\lambda)\sum_{i=1}^{k}c(h_i),$$

and the generating functions $\widetilde{F}(s),\widetilde{G}(s)$ are easily related to $F(s),G(s)$ via the Riemann series $\widehat{\zeta}(s)$ of valid numbers,

$$\widetilde{F}(s)=\widehat{\zeta}(s)F(s),\qquad\widetilde{G}(s)=\widehat{\zeta}(s)G(s)+\Delta\widehat{\zeta}(s)I(s)\qquad\text{with}\qquad I(s):=\sum_{(u_1,u_0)\in\Omega}\frac{1}{u_0^s}\sum_{i=1}^{k}c(h_i).\tag{11}$$

Then, the same principles as previously provide an alternative expression for the Dirichlet series $I(s)$

$$I(s)=\left[(I-\mathbf{H}_s)^{-1}\circ\;\mathbf{H}_s^{[c]}+\mathbf{F}_s^{[c]}\right]\;\circ\;(I-\mathbf{H}_s)^{-1}\;[1](a),\tag{12}$$

the final expression for $\widetilde{G}(s)$, and also an alternative expression for a third Dirichlet series,

$$J(s):=\sum_{(u_1,u_0)\in\Omega}\frac{1}{u_0^s}\,\log_2 u_0=\Delta[\mathbf{F}_s\circ(I-\mathbf{H}_s)^{-1}]\;[1](a)=[\Delta\mathbf{F}_s+\mathbf{F}_s\circ(I-\mathbf{H}_s)^{-1}\circ\Delta\mathbf{H}_s]\;\circ(I-\mathbf{H}_s)^{-1}\;[1](a).\tag{13}$$

3.5. Functional Analysis. Here, we consider a set $\mathcal{H}$ of LFTs and the following conditions on set $\mathcal{H}$ that will entail all the properties that we need for applying the Tauberian Theorem to the quasi–inverse $(I-\mathbf{H}_s)^{-1}$ of the Ruelle operator relative to $\mathcal{H}$.

Conditions $\mathcal{Q}(\mathcal{H})$. *There exist an open disk $\mathcal{V}$ whose closure contains the basic interval $\mathcal{I} := [0,\rho]$, and a real $\alpha < 2$ such that*

(C_1) Every LFT $h \in \mathcal{H}$ has an analytic continuation on $\mathcal{V}$, and h maps the closure $\bar{\mathcal{V}}$ of disk $\mathcal{V}$ inside $\mathcal{V}$. All the functions $|h'|$ have an analytic continuation on $\mathcal{V}$.
(C_2) There exists a convenient Banach functional space $\mathcal{F}(\mathcal{V})$ formed with analytic functions on $\mathcal{V}$ such that each Ruelle operator $\mathbf{R}_{s,h}$ (for $h \in \mathcal{H}$ and $s > \alpha$) acts on $\mathcal{F}(\mathcal{V})$ and is compact. Moreover, the series of norms $\sum_{h\in\mathcal{H}} ||\mathbf{R}_{s,h}||$ converges on the plane $\Re(s) > \alpha$.
(C_3) For $s = 2$, the operator $\mathbf{H}_2$ is a density transformer: for all $f \in \mathcal{F}(\mathcal{V})$ positive on $\mathcal{V}\cap\mathbf{R}$, one has

$$\int_0^\rho \mathbf{H}_2[f](t)dt = \int_0^\rho f(t)dt.$$

(C_4) For some integer A, the set $\mathcal{H}$ contains a subset $\mathcal{D} := \{h \mid h(x) = A/(c+x) \quad \text{with integers } c \to \infty\}$.

If conditions $\mathcal{Q}(\mathcal{H})$ hold, we can prove the following: Under conditions (C_1) and (C_2), the Ruelle operator $\mathbf{H}_s$ acts on $\mathcal{F}(\mathcal{V})$ for $\Re(s) > \alpha$ and is compact (even nuclear in the sense of Grothendieck [10, 11]). Furthermore, for real values of parameter s, it has positive properties that entail (via Theorems of Perron–Frobenius style due to Krasnoselsky [15]) the existence of dominant spectral objects: there exists a unique dominant eigenvalue $\lambda(s)$ positive, analytic for $s > \alpha$, a dominant eigenfunction denoted by ψ_s, and a dominant projector e_s. Under normalization condition $e_s[\psi_s] = 1$, these last two objects are unique too. Then, the compacity entails the existence of a spectral gap between the dominant eigenvalue and the remainder of the spectrum, that separates the operator $\mathbf{H}_s$ and the quasi-inverse $(I-\mathbf{H}_s)^{-1}$ in two parts : the "part" relative to the dominant eigenvalue, and the "part" relative to the remainder of the spectrum. Under conditions (C_3), the operator $\mathbf{H}_2$ is a density transformer, so that $\lambda(2) = 1$ and $e_2[f] = \int_0^\rho f(t)dt$. On the other hand, condition (C_4) implies that the operator $\mathbf{H}_s$ has no eigenvalue equal to 1 on the line $\Re(s) = 2, s \neq 2$. Finally, the quasi–inverse of the Ruelle operator which intervenes in the expression of generating functions $F(s)$ and $G(s)$ fulfills all the hypotheses of Tauberian Theorem:

Theorem 1. *Let $\mathcal{H}$ be a set of LFTs for which conditions $\mathcal{Q}(\mathcal{H})$ hold. Then, the quasi-inverse $(I-\mathbf{H}_s)^{-1}$ is analytic on the punctured plane $\{\Re(s) = 2, s \neq 2\}$ and has a pole of order 1 at $s = 2$. Near $s = 2$, one has, for any function f of $\mathcal{F}(\mathcal{V})$ positive on $\mathcal{V}\cap\mathbf{R}$, and any $x \in \mathcal{V}\cap\mathbf{R}$,*

$$(I-\mathbf{H}_s)^{-1}[f](x) \sim \frac{1}{(s-2)}\,(\frac{-1}{\lambda'(2)})\,\psi_2(x)\int_0^\rho f(x)dx, \tag{14}$$

where $\lambda(s)$ is the dominant eigenvalue of $\mathbf{H}_s$ and ψ_2 is the dominant eigenfunction of $\mathbf{H}_2$ defined by the normalization condition $\int_0^\rho \psi_2(x)dx = 1$.

The previous result shows that each occurrence of the quasi-inverse "brings" one pole at $s = 2$, so that the Dirichlet series $F(s), \widetilde{F}(s), G(s), \widetilde{G}(s)$ admit a pole at $s = 2$ of various order.
When applying Theorem 1 to $F(s)$, and comparing (14) with (5), we obtain an expression for the derivative $-\lambda'(2)$ in terms of ρ and $\mathbf{F}_2[\psi_2](a)$. The quantity $-2\lambda'(2)$ is quite important since it is equal to the entropy $h(\mathcal{H})$ of the dynamical system associated to the algorithm.

3.6. Fast algorithms versus slow algorithms. On the other side, Theorem 1 shows that Dirichlet series $G(s), \widetilde{G}(s)$ admit a pole at $s = 2$ of order at least three. This order is exactly three, if the Ruelle operator of costs $\mathbf{H}_s^{[c]}$ is regular at $s = 2$. However, the Ruelle operator of costs may have itself a pole at $s = 2$, so that the total order of pole at $s = 2$ becomes four. This implies a general criterion between "fast" algorithms with a log-squared behaviour and "slow" algorithms with a log-cubed behaviour

Theorem 2. *Let $\mathcal{H}$ be a set of LFTs for which conditions $\mathcal{Q}(\mathcal{H})$ hold.*
(a). Suppose that the Ruelle operator of costs $\mathbf{H}_s^{[c]}$ is regular at $s=2$. Then the average bit-complexity of the Euclidean Algorithm on the set of valid inputs of denominator less than N is of asymptotic logarithmic–squared order. It satisfies

$$\widetilde{B}_N(\mathcal{H}) \sim B_N(\mathcal{H}) \sim \ A(\mathcal{H}) \log_2^2 N \qquad \textit{with} \qquad A(\mathcal{H}) = \frac{\log 2}{h(\mathcal{H})}\, E_\infty[c].$$

Here $h(\mathcal{H})$ is the entropy of the dynamical system relative to the algorithm and $E_\infty[c]$ denotes the average value of cost c related to the LFT h when the interval $\mathcal{I}$ is endowed with the invariant measure associated to the dominant eigenfunction of $\mathbf{H}_2$.
(b). Suppose that the integral relative to Ruelle operator of costs $\int_0^\rho \mathbf{H}_s^{[c]}[\psi_2](t)dt$ has a pole of order 1 at $s=2$ with a residue equal to A. Then the average bit-complexity of the Euclidean Algorithm on the set of valid inputs of denominator less than N is of asymptotic logarithmic–cubed order. It satisfies

$$\widetilde{B}_N(\mathcal{H}) \sim B_N(\mathcal{H}) \sim \ A(\mathcal{H}) \log_2^3 N \qquad \textit{with} \qquad A(\mathcal{H}) = \frac{2\log^2 2}{3h(\mathcal{H})}\, A.$$

Proof. (a) Theorem 1 shows that the Dirichlet series $G(s)$ has a triple pole at $s=2$, and near $s=2$

$$F(s) \sim \frac{-1}{\lambda'(2)}\,\frac{\mathbf{F}_2[\psi_2](a)}{s-2}, \qquad\qquad G(s) \sim (\frac{-1}{\lambda'(2)})^3\, \frac{\mathbf{F}_2[\psi_2](a)}{(s-2)^3}\ \ (\int_0^\rho \Delta H_2[\psi_2](t)dt)\ \ (\int_0^\rho H_2^{[c]}[\psi_2](t)dt).$$

Both integrals are easily transformed. The first one

$$\int_0^\rho \Delta H_2[\psi_2](t)dt = \int_0^\rho |\log_2 t|\, \psi_2(t)dt$$

equals $-\lambda'(2)/\log 2$ and coincides with $h(\mathcal{H})/(2\log 2)$, where $h(\mathcal{H})$ is the entropy of the dynamical system. The second integral deals with the cost $c(h)$ of the LFT

$$\int_0^\rho H_2^{[c]}[\psi_2](t)dt = \sum_{h\in\mathcal{H}} c(h) \int_{h(\mathcal{I})} \psi_2(t)dt$$

and coincides with the average value of cost c when the interval $\mathcal{I}$ is endowed with the invariant measure $\psi_2(t)dt$. This average value is denoted by $E_\infty[c]$ and it is a constant of Khinchin's type.

(b) The Dirichlet series $G(s), \widetilde{G}(s)$ have now a pole of order 4 at $s=2$, and near $s=2$, one has

$$F(s) \sim \frac{-1}{\lambda'(2)}\,\frac{\mathbf{F}_2[\psi_2](a)}{s-2}, \qquad\qquad G(s) \sim (\frac{-1}{\lambda'(2)})^3\, \frac{\mathbf{F}_2[\psi_2](a)}{(s-2)^4}\ \ A\ \ (\int_0^\rho \Delta H_2[\psi_2](t)dt),$$

and, as previously, the integral equals $-\lambda'(2)/\log 2$ and coincides with $h(\mathcal{H})/(2\log 2)$, where $h(\mathcal{H})$ is the entropy of the dynamical system. ■

Remark. We can now prove why the approximation that we have made on the cost $C(u_1, u_0)$ is valid. When one replaces the binary length $\ell(u)$ by the logarithm $\log_2(u)$, or when one replaces $\log_2(u_i)$ by $\log_2(u_{i-1})$ or $\log_2(u_{i+1})$, one adds to Dirichlet series $G(s)$ some multiple of series $I(s)$ or $J(s)$; Equations (12) and (13) together with (14) show that each of these series has a pole at $s=2$ whose order is always strictly less than $G(s)$ so that the main order term of the average cost is not modified by our approximation.

4 Average-Bit Complexity of the Algorithms

We now come back to the analysis of the five algorithms, and we study successively the fast algorithms, then the slow ones. Then we obtain precise insights on the distribution of costs that refine the classification between fast algorithms and slow ones.

4.1. The fast algorithms. We study here three algorithms: the Classical Algorithm (G), the Centered Algorithm (K) and the Binary Algorithm (B). We begin by the first two, that constitute "easy cases".

The Classical Algorithm and the Centered Algorithm. We first consider the sets $\mathcal{G}, \mathcal{K}$ relative to the Classical Algorithm or the Centered Algorithm. There exists, in both cases, an open disk $\mathcal{V}$ whose diameter *strictly* contains the basic interval $\mathcal{I}$, such that each LFT h maps $\bar{\mathcal{V}}$ strictly inside itself. A convenient functional space is then the set $\mathcal{A}_\infty(\mathcal{V})$ formed with functions f that are analytic on $\mathcal{V}$ and continuous on $\bar{\mathcal{V}}$. Endowed with the sup-norm, this set is a Banach space and each Ruelle operator $\mathbf{R}_{s,h}$ acts on this set and is compact. The series of (C_2) is convergent for $\Re(s) > 1$. Moreover, at $s = 2$, the Ruelle operators are density transformers. The (normalized) invariant functions ψ_2 are respectively, in the classical case and in the centered case,

$$\frac{1}{\log 2}\frac{1}{1+x}, \qquad \frac{1}{\log\phi}[\frac{1}{\phi+x}+\frac{1}{\phi^2-x}],$$

and the related distribution functions $F_2(x)$ are

$$\frac{1}{\log 2}\log\frac{1}{1+x}, \qquad \frac{1}{\log\phi}\log\frac{\phi(\phi+x)}{\phi^2-x}.$$

Now, we can apply Theorem 2, version (a). Here, since invariant function ψ_2 is explicit, the same is true for the entropy that equals $\pi^2/(6\log 2)$ in the classical case and equals $\pi^2/(6\log\phi)$ in the centered case. Moreover, the average value $E_\infty[c]$ is also easily computed from explicit forms of cost c, of invariant function ψ_2 and distribution function F_2.

The Binary Algorithm. The set $\mathcal{B}$ of LFT's relative to the Binary algorithm is more difficult to deal with. First, it is not possible to find an open disk whose diameter contains the basic interval $\mathcal{I} := [0,\rho]$ and on which all the LFT's are analytic. The reason is that the sequence of pôles of LFT's is of the form $x = -m/2^s$ and has an accumulation point at $x = 0$. We choose for $\mathcal{V}$ an open disk of diameter $]0,\alpha[$ with $1 < \alpha < 2$, and a convenient functional space is then the Hardy space of order two relative to $\mathcal{V}$. It is denoted by $\mathcal{H}^2(\mathcal{V})$ and is formed with all functions f analytic inside $\mathcal{V}$ and such that $|f|^2$ is integrable along the frontier of $\mathcal{V}$. Each Ruelle operator $\mathbf{R}_{s,h}$ acts on this set and is compact. The series of (C_2) is convergent for $\Re(s) > (3/2)$. Now, we can apply Theorem 2, version (a) as previously. However, the dynamical systems to which the algorithm is associated is now more complex, since it is a random dynamical system. The reason is that the pseudo-division is related to dyadic valuation, so that the binary continued fraction expansion is only defined for rationals numbers. However, one can define random binary continued fraction for real numbers when choosing at random the dyadic valuation k of a real number, according to the law $\Pr[k = d] = 2^{-d}$ (for $d \geq 1$), that extends the natural law on even integers. In this manner, we choose the LFT of determinant 2^k with probability 2^{-k}, and, for LFT's of determinant 2^k, the quantity

$$\frac{1}{D[h](x)^2}f\circ h(x)dx = \frac{|h'(x)|}{2^k}f\circ h(x)dx$$

represents exactly a random change of variables. Then, the Ruelle operator relative to the Binary Algorithm can be viewed as the transfer operator relative to this random dynamical system and for $s = 2$ it is a (random) density transformer.
Furthermore, even if the invariant eigenfunction ψ_2, the distribution function F_2 are no more explicit, the entropy and the average value $E_\infty[c]$ can be easily computed as functions of F_2 and ψ_2.

Theorem 3. *The average bit-complexities of the Classical Algorithm (G), the Centered Algorithm (K) and the Binary Algorithm on the set of valid inputs of denominator less than N are of asymptotic logarithmic–squared order. They all satisfy*

$$\widetilde{B}_N(\mathcal{H}) \sim B_N(\mathcal{H}) \sim A(\mathcal{H})\log_2^2 N \qquad \textit{with} \qquad A(\mathcal{H}) = \frac{\log 2}{h(\mathcal{H})}E_\infty[c] \qquad \textit{for} \qquad \mathcal{H}\in\{\mathcal{G},\mathcal{K},\mathcal{B}\}.$$

Here $h(\mathcal{H})$ is the entropy of the dynamical system relative to the algorithm and $E_\infty[c]$ denotes the average value of cost c related to the LFT h when the interval $\mathcal{I}$ is endowed with the invariant measure. More precisely, in the classical case (G), the cost $c(h)$ equals $\ell(m)+2$ where $\ell(m)$ is the number of bits of digit m, and

$$A(\mathcal{G}) = \frac{6\log^2 2}{\pi^2}\ [2 + \frac{1}{\log 2}\log\prod_{k=0}^{\infty}(1+\frac{1}{2^k})];$$

in the centered case (K), the cost $c(h)$ equals $\ell(m)+2+(1-\varepsilon)/2$ where $\ell(m)$ is the number of bits of digit m, and $\varepsilon = \pm 1$ the sign used, so that

$$A(\mathcal{K}) = \frac{6\log\phi\log 2}{\pi^2}[3 + \frac{\log 2}{\log\phi} + \frac{1}{\log\phi}\log\prod_{k=3}^{\infty}\frac{(2^k-1)\phi^2+2\phi}{(2^k-1)\phi^2-2}].$$

In the binary case (B), the cost $c(h)$ equals $b(m)+2$ where $b(m)$ is the number of ones in the binary expansion of digit m, and

$$A(\mathcal{B}) = \frac{2\log 2}{\pi^2\,\psi_2(1)}\ [1 + \sum_{m\ \ odd \geq 1}\frac{1}{2^{\ell(m)}}\,F_2(\frac{1}{m})],$$

where $\psi_2(t)dt$ is the invariant measure, F_2 its distribution function, and, as previously, $\ell(m)$ is the number of bits of integer m.

Remark. In these three cases, the average number of iterations $P_N(\mathcal{H})$ is well–known so that the average bit–complexity $B_N(\mathcal{H})$ can be directly related to $P_N(\mathcal{H})$,

$$\widetilde{B}_N(\mathcal{H}) \sim B_N(\mathcal{H}) \sim P_N(\mathcal{H}) \times E_\infty[c] \times \frac{1}{2}\log_2 N, \qquad \text{with} \qquad P_N(\mathcal{H}) \sim \frac{2}{h(\mathcal{H})}\log N$$

and the mean-value of $\log_2(u_i)$ is $(1/2)\log_2 N$.

4.2. The slow algorithms. We study now the Subtractive Algorithm (T) and the By-Excess Algorithm (L). In these cases, the analysis requires a special twist that takes its inspiration from the study of intermittency phenomena in physical systems that was introduced by Bowen [2] and is nicely exposed in a paper of Prellberg and Slawny [17].

The Subtractive Algorithm. Here, Theorem 2, version (b) directly applies since the integral relative to the Ruelle operator of costs is a divergent series at $s=2$ of the form

$$\int_0^\rho \mathbf{H}_s^{[c]}[\psi_2](t)dt = \sum_{m\geq 1}\frac{1}{\log 2}\int_0^\rho \frac{m}{(m+t)^{s-1}}\frac{1}{(m+1+t)}dt,$$

with residue A equal to $1/\log 2$.

The By-Excess Algorithm. In this case, the set of LFTs $\mathcal{L}$ does not fulfill conditions $\mathcal{Q}(\mathcal{L})$ since it contains one "bad" LFT $p : x \to 1/(2-x)$ which possesses an indifferent point at $x=1$, i.e., a fixed point where the absolute value of the derivative equals 1. In this case, it is not possible to find an open disk $\mathcal{V}$ that contains the basic interval $\mathcal{I}$ and such that the LFT p maps $\bar{\mathcal{V}}$ inside $\mathcal{V}$. However, the remainder of the set, i.e., the set $\mathcal{L}\setminus\{p\}$ is well-behaved, so that we adapt the *method of inducing* that originates from dynamical systems theory. The main idea is to consider a sequence of p to be followed by a good LFT q that belong to $\mathcal{L}\setminus\{p\}$. This is always possible since final set $\mathcal{F}$ does not contain p. In that way, instead of set $\mathcal{L}$, we use set $\mathcal{M} := p^\star[\mathcal{L}\setminus\{p\}]$ that now fulfills conditions $\mathcal{Q}(\mathcal{M})$. An iteration that uses an element $p^k q$ of $\mathcal{M}$ with $k \geq 0$ and $q \neq p$

$$u_0 = 2u_1 - u_2, \qquad u_1 = 2u_2 - u_3, \qquad \ldots u_{k-1} = 2u_k - u_{k+1}, \qquad u_k = mu_{k+1} - u_{k+2} \qquad \text{with } m \neq 2$$

deals with integers u_i (for $1 \le i \le k$) that are well-approximated by a linear function $u_0/(i+2) \le u_i \le u_0$, so that the global cost of of the iteration is of the form

$$\sum_{i=1}^{k} \log_2(u_i) + \ell(m)\log_2(u_{k+1}) \approx k\log_2(u_0).$$

Finally, one can apply Theorem 2, version (b), to the set $\mathcal{M}$, and the integral relative to the Ruelle operator of costs $\int_0^\rho \mathbf{M}_s^{[c]}[\psi_2](t)dt$ is a divergent series at $s=2$ with residue A equals $1/(2\log 2)$. Remark the equality between the quasi-inverses of Ruelle operators $\mathbf{L}_s, \mathbf{P}_s, \mathbf{M}_s$ relative to sets $\mathcal{L}, \{p\}, \mathcal{M}$,

$$(I-\mathbf{L}_s)^{-1} = (I-\mathbf{P}_s)^{-1}\ (I-\mathbf{M}_s)^{-1}$$

that provides an explicit expression for the entropy of the (induced) dynamical system.
In fact, we have used implicitly this method of inducing for the Subtractive algorithm too. For each (internal) step, the algorithm can use two LFTs, $p(x) := x/(1+x), q(x) := 1/(1+x)$, depending if the subtraction is followed or not by an exchange. The LFT q is good, but the LFT p is bad, with an indifferent point at $x=0$. Then the induced set $\mathcal{M}$ is just the set $\mathcal{G}$.

Theorem 4. *The average bit–complexities of the By-Excess Algorithm (L) and the Subtractive Algorithm (T) on the set of valid inputs of denominator less than N are of asymptotic log–cubed order. They all satisfy*

$$\widetilde{B}_N(\mathcal{H}) \sim B_N(\mathcal{H}) \sim\ A(\mathcal{H})\log^3 N \qquad \textit{with} \qquad A(\mathcal{T}) = \frac{2\log^2 2}{\pi^2}, \qquad A(\mathcal{L}) = \frac{\log^2 2}{\pi^2}.$$

Remark. In these two cases, the average number of iterations $P_N(\mathcal{H})$ is well–known so that the average bit–complexity $B_N(\mathcal{H})$ can be directly related to $P_N(\mathcal{H})$,

$$\widetilde{B}_N(\mathcal{H}) \sim B_N(\mathcal{H}) \sim P_N(\mathcal{H})\,\frac{1}{3}\log_2 N, \qquad \text{with} \qquad P_N(\mathcal{H}) \sim \frac{1}{h(\mathcal{H})\log 2}\log^2 N$$

and the mean-value of $\log_2(u_i)$ is $(1/3)\log_2 N$.

4.3. Higher moments. When adapting our methods, we can easily evaluate higher moments of costs.

Theorem 5. (i) *For any integer ℓ and any of the three fast algorithms, the moment of order ℓ of the cost function is asymptotic to the ℓth power of the mean. In particular the standard deviation is $o(\log N)$. Consequently the random variable expressing the cost satisfies the concentration of distribution property.*
(ii) *For any integer $\ell \ge 2$ and any of the two slow algorithms, the ℓth moment of the cost function is of order $N^{\ell-1}$. In particular the standard deviation is $\Theta(\sqrt{N})$.*

References

[1] Bedford, T., Keane, M., and Series, C., Eds. *Ergodic Theory, Symbolic Dynamics and Hyperbolic Spaces*, Oxford University Press, 1991.

[2] Bowen, R. Invariant measures for Markov maps of the interval, *Commun. Math. Phys.* 69 (1979) 1–17.

[3] Brent, R.P. Analysis of the binary Euclidean algorithm, Algorithms and Complexity, New directions and recent results, ed. by J.F. Traub, Academic Press 1976, pp 321–355

[4] Daudé, H., Flajolet, P., and Vallée, B. An average-case analysis of the Gaussian algorithm for lattice reduction, *Combinatorics, Probability and Computing* (1997) 6 397–433

[5] Delange, H. Généralisation du Théorème d'Ikehara, *Ann. Sc. ENS, (1954) 71, pp 213–242.*

[6] Dixon, J. D. The number of steps in the Euclidean algorithm, *Journal of Number Theory 2* (1970), 414–422.

[7] Flajolet, P. Analytic analysis of algorithms, In Proceedings of the 19th International Colloquium "Automata, Languages and Programming", Vienna, July 1992, W. Kuich, editor, Lecture Notes in Computer Science 623, pp 186–210

[8] Flajolet, P. and Sedgewick, R. Analytic Combinatorics, Book in preparation (1999), see also INRIA Research Reports 1888, 2026, 2376, 2956.

[9] Flajolet, P., and Vallée, B. Continued fraction Algorithms, Functional operators and Structure constants, *Theoretical Computer Science* 194 (1998), 1–34.

[10] Grothendieck, A. Produits tensoriels topologiques et espaces nucléaires, *Mem. Am. Math. Soc.* 16 (1955)

[11] Grothendieck, A. La théorie de Fredholm, *Bull. Soc. Math. France* 84 pp 319-384.

[12] Heilbronn, H. On the average length of a class of continued fractions, Number Theory and Analysis, ed. by P. Turan, New-York, Plenum, 1969, pp 87-96.

[13] Hensley, D. The number of steps in the Euclidean algorithm, *Journal of Number Theory* 49, 2 (1994), 142–182.

[14] Knuth, D.E. The art of Computer programming, Volume 2, 3rd edition, Addison Wesley, Reading, Massachussets, 1998.

[15] Krasnoselskii, M. *Positive solutions of operator equations*, P. Noordhoff, Groningen, 1964.

[16] Mayer, D. H. Continued fractions and related transformations, In *Ergodic Theory, Symbolic Dynamics and Hyperbolic Spaces*, T. Bedford, M. Keane, and C. Series, Eds. Oxford University Press, 1991, pp. 175–222.

[17] Prellberg, T. and Slawny, J. Maps of intervals with Indifferent fixed points: Thermodynamic formalism and Phase transitions. *Journal of Statistical Physics* 66 (1992) 503-514

[18] Rieger, G. J. Uber die mittlere Schrittazahl bei Divisionalgorithmen, *Math. Nachr.* (1978) pp 157-180

[19] Ruelle, D. *Thermodynamic formalism,* Addison Wesley (1978)

[20] Ruelle, D. *Dynamical Zeta Functions for Piecewise Monotone Maps of the Interval*, vol. 4 of *CRM Monograph Series*, American Mathematical Society, Providence, 1994.

[21] Tenenbaum, G. *Introduction à la théorie analytique des nombres*, vol. 13. Institut Élie Cartan, Nancy, France, 1990.

[22] Vallée, B. Opérateurs de Ruelle-Mayer généralisés et analyse des algorithmes d'Euclide et de Gauss, *Acta Arithmetica* 81.2 (1997) 101–144.

[23] Vallée, B. Fractions continues à contraintes périodiques, *Journal of Number Theory* 72 (1998) pp 183–235.

[24] Vallée, B. Dynamics of the Binary Euclidean Algorithm: Functional Analysis and Operators., *Algorithmica* (1998) vol 22 (4) pp 660–685.

[25] Vallée, B. A unifying Framework for the analysis of a class of Euclidean Algorithms., To appear in the proceedings of LATIN'2000, LNCS

[26] Vardi, I. Continued fractions, Preprint, chapter of a book in preparation.

[27] Yao, A.C., and Knuth, D.E. Analysis of the subtractive algorithm for greatest common divisors. *Proc. Nat. Acad. Sc. USA* 72 (1975) pp 4720-4722.