J. Symbolic Computation (1990) **9**, 429–455

Analysis of Euclidean Algorithms for Polynomials over Finite Fields

KEJU MA

JOACHIM VON ZUR GATHEN

Department of Computer Science, University of Toronto
Toronto, Ontario M5S 1A4, Canada

This paper analyzes the Euclidean algorithm and some variants of it for computing the greatest common divisor of two univariate polynomials over a finite field. The minimum, maximum and average number of arithmetic operations both on polynomials and in the ground field are derived. We consider five different algorithms to compute $\gcd(A_1, A_2)$ where $A_1, A_2 \in \mathbb{Z}_2[x]$ have degrees $m \geq n \geq 0$. Compared with the classical Euclidean algorithm that needs on average $\frac{1}{2}n + 1$ polynomial divisions, two algorithms involving divisions need on average $\frac{1}{3}n + O(1)$ and $\frac{1}{4}n + O(1)$ polynomial divisions; two other algorithms use an average of $\frac{1}{2}m + \frac{1}{3}n + O(1)$ and $\frac{1}{4}m + \frac{2}{9}n + O(1)$ polynomial subtractions and no divisions.

1. Introduction

The polynomial gcd problem is to compute the greatest common divisor of any two nonzero polynomials over a unique factorization domain. Many sequential and parallel algorithms for this problem—all based on the Euclidean algorithm—are well known in the literature (e.g. Collins 1967; Brown 1971; Brown & Traub 1971; Aho *et al.* 1975, ch. 8; Knuth 1981, ch. 4; Borodin *et al.* 1982; Strassen 1983; von zur Gathen 1984).

The classical Euclidean algorithm for integers has been investigated in considerable depth. Lamé (1844) established an upper bound on the number of divisions. A much more complex analysis on the average number of divisions in the Euclidean algorithm was carried out by Heilbronn (1968) and Dixon (1970). Collins (1974) analyzed the computing time of the Euclidean algorithm on multiple-precision integers. We refer the reader to Knuth (1981, ch. 4) for an excellent review.

Stein (1967) discovered an interesting algorithm called the *binary Euclidean algorithm* for computing the gcd of two integers (Knuth 1981, p. 321). Unlike the classical Euclidean algorithm, this algorithm does not use divisions, but requires the arithmetic operations of subtraction and right shifting (division by 2), and hence is faster than the classical algorithm in each iteration. However, an exact analysis of this algorithm seems to be

This work was partly supported by National Science and Engineering Council of Canada, grant 3-650-126-40, and Fundación Andes, beca C-10246. Part of the second author's work was done during a visit to Pontificia Universidad Católica, Santiago, Chile, and as a Visiting Fellow at the Computer Science Laboratory, Australian National University, Canberra, Australia.

© 1990 Academic Press Limited

KEJU MA & JOACHIM VON ZUR GATHEN (1990). Analysis of Euclidean Algorithms for Polynomials over Finite Fields. *Journal of Symbolic Computation* **9**, 429–455. URL http://dx.doi.org/10.1016/S0747-7171(08)80021-1.
This document is provided as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, notwithstanding that these works are posted here electronically. It is understood that all persons copying any of these documents will adhere to the terms and constraints invoked by each copyright holder, and in particular use them only for noncommercial purposes. These works may not be posted elsewhere without the explicit written permission of the copyright holder. (Last update 2016/05/18-14 :22.)

very difficult to obtain. To date, there has been a discrete model and a continuous model which describe the average behavior of the algorithm under some reasonable assumptions, proposed by Knuth (1981, pp. 330-336) and Brent (1976) respectively.

In this paper we examine the classical Euclidean algorithm and some variants analogous to the binary Euclidean algorithm for computing the gcd of two polynomials over a finite field. The Euclidean algorithm for polynomials is essential to symbolic and algebraic computing (Collins 1967; Brown 1971; Knuth 1981, ch. 4). In Section 2, we determine precisely the minimum, maximum and average number of polynomial divisions and arithmetic operations in the ground field used in the Euclidean algorithm, depending on three parameters: the degrees of two input polynomials and the size of the ground field. We assume that "synthetic" polynomial division is used.

For the average-case analysis of algorithms, a difficulty is often to find an undisputed meaning of what the "average input" is. In our case, this is quite simple: up to a fixed degree, there is only a finite number of polynomials, and we consider each of them to be equally likely. All algorithms in Sections 3 to 6 require an "initialization" step, and we consider only inputs for which this step is irrelevant.

In Section 3 we analyze one variant of the Euclidean algorithm called the *indeterminate shift Euclidean algorithm* for an arbitrary finite field. The main feature of the new algorithm is to use indeterminate shifting (divisions by x) to speed up computing time; that is, one removes all factors x from the resulting polynomial remainder after each division. A careful analysis shows that the algorithm runs on average faster than the classical Euclidean algorithm. In Section 4 we consider the *linear factor shift Euclidean algorithm* which performs linear factor shifting so that all linear factors are removed from the resulting polynomial remainder after each division. The two new algorithms in Sections 5 and 6 are the polynomial analogs of the binary Euclidean algorithm for integers. The *subtractive indeterminate shift algorithm* in Section 5 uses only polynomial subtractions, but requires indeterminate shifting after each subtraction. The *subtractive linear factor shift algorithm* in Section 6 uses polynomial subtractions and performs linear factor shifting after each subtraction.

Exact average-case as well as best-case and worst-case analyses are performed for all algorithms in Sections 4 to 6 over the finite field $\mathbf{Z}_2$. In comparison with the classical Euclidean algorithm for computing polynomial gcd's over $\mathbf{Z}_2$, we show that the linear factor shift Euclidean algorithm runs twice as fast as the Euclidean algorithm in the sense of average number of polynomial divisions used, and the subtractive linear factor shift algorithm uses less polynomial subtractions than the Euclidean algorithm uses polynomial divisions.

Sections 2 through 6 all follow the same pattern. We first describe an algorithm, then state several theorems about its average cost, followed by a few remarks. After this, a lemma states the distribution of results of one basic step in the algorithm on random inputs, and a corollary expresses this as transition probability in a "lattice" (as explained in Section 3), and finally proofs of theorems. Out of space considerations, some theorems are left unproven; detailed proofs are in Ma (1987). At the end, we discuss the best and worst cases.

For many applications, such as inversion in a finite non-prime field, (rational, Hermite) interpolation, Padé approximation (see von zur Gathen 1986), and the decoder implementation of a variety of error-correcting codes (MacWilliams & Sloane 1977), one needs the "Extended Euclidean Algorithm" for (A_1, A_2), which also calculates polynomials S_i and

T_i for every remainder A_i with $S_i A_1 + T_i A_2 = A_i$. It is easy to "extend" our algorithms in this sense; we do not do this here.

The main purpose of this paper is to show that we can determine exactly the average-case behavior of nontrivial algorithms.

2. The Classical Euclidean Algorithm

Let F be a finite field with p elements (so that p is a power of a prime), and let $F[x]$ be the polynomial ring over F in the indeterminate x. For $0 \le n \le m$, let $P_m = \{A \in F[x] \mid \deg A = m\}$ and $P_{m,n} = P_m \times P_n$. Then $\#P_{m,n} = (p-1)^2 p^{m+n}$. Applying the polynomial Euclidean algorithm to an input $(A_1, A_2) \in P_{m,n}$, we obtain a unique polynomial quotient sequence $(Q_1, \ldots, Q_{t-1})$ and a unique polynomial remainder sequence $(A_3, \ldots, A_t)$ such that

$$\begin{aligned} A_1 &= Q_1 A_2 + A_3, \\ A_2 &= Q_2 A_3 + A_4, \\ \vdots\ &\ \vdots\ \ \vdots \\ A_{t-2} &= Q_{t-2} A_{t-1} + A_t, \\ A_{t-1} &= Q_{t-1} A_t, \end{aligned} \tag{1}$$

where $A_i \neq 0$, $\deg A_{i+1} < \deg A_i$ for $1 < i < t$. If $a \in F$ is the leading coefficient of A_t, then $\gcd(A_1, A_2) = A_t / a$ is the monic scalar multiple of A_t.

Let $n_i = \deg Q_i$ for $1 \le i < t$ and $n_t = \deg A_t$. Then $(n_1, \ldots, n_t)$ has the following properties: $n_1 = m - n$, $n_t \ge 0$, $n_i > 0$ $(1 < i < t)$, $\sum_{i=2}^{t} n_i = n$, $2 \le t \le n + 2 - n_t$. Each polynomial pair (A_1, A_2) has an associated extended quotient sequence $(Q_1, \ldots, Q_{t-1}, A_t)$, which is called by Knuth (1970) the *Euclidean representation* of (A_1, A_2), and $(n_1, \ldots, n_t)$ is the *Euclidean representation pattern* of (A_1, A_2).

The Euclidean algorithm establishes a bijection between the set of polynomial pairs (A_1, A_2) in $P_{m,n}$ whose Euclidean representation is $(n_1, \ldots, n_t)$ and the set of finite sequences of $t \ge 2$ polynomials $(Q_1, \ldots, Q_{t-1}, A_t)$. There are precisely $(p-1)^t p^m$ polynomial pairs (A_1, A_2) in $P_{m,n}$ whose Euclidean representation pattern is $(n_1, \ldots, n_t)$.

Let $0 \le n \le m$, $(u, v) \in P_{m,n}$, and $v \neq 0$. For any operation $\omega \in \{\div, -, \times\}$, let $d^{\omega}(u, v)$ denote the number of operations ω used in the "synthetic" polynomial division algorithm for the division of u by v (Knuth 1981, p. 402). Then

$$d^{\div}(u, v) = m - n + 1, \quad d^{-,\times}(u, v) = n(m - n + 1). \tag{2}$$

In particular, all inputs $(u, v) \in P_{m,n}$ uss the same number of arithmetic operations in F. For any operation $\omega \in \{\div, -, \times\}$, the computing time t^{ω} of the Euclidean algorithm is a function from $P_{m,n}$ to $\mathbb{N}$ such that for any input $(A_1, A_2) \in P_{m,n}$, $t^{\omega}(A_1, A_2)$ is the number of operations ω performed in the Euclidean algorithm where each polynomial division is carried out by the "synthetic" polynomial division algorithm. Furthermore, $t^{\text{div}}(A_1, A_2)$ denotes the number of polynomial divisions used.

THEOREM 2.1. *For $0 \le n \le m$ and $\omega \in \{\mathrm{div}, \div, -, \times\}$, the average number $t^{\omega}_{\mathrm{ave}}(m,n,p)$ of operations ω performed on (uniformly distributed) inputs from $P_{m,n}$ satisfies*

$$\begin{aligned} t^{\mathrm{div}}_{\mathrm{ave}}(m,n,p) &= (1-1/p)\,n+1, \\ t^{\div}_{\mathrm{ave}}(m,n,p) &= m+(1-1/p)\,n+1-(1-p^{-n})/(p-1), \\ t^{-,\times}_{\mathrm{ave}}(m,n,p) &= m\,n-\tbinom{n}{2}/p-n/(p-1)+p\,(1-p^{-n})/(p-1)^2. \end{aligned}$$

REMARK 2.2. Knuth (1981, Ex. 4.6.1—4) obtains a similar result on $t^{\omega}_{\mathrm{ave}}(m,n,p)$. The fastest known algorithms to compute the Euclidean representation are based on a divide-and-conquer technique (Lehmer 1938) and use $O(m \log^2 n \log\log n)$ arithmetic operations (Knuth 1970; Schönhage 1971; Moenck 1973). Strassen (1983) gives a detailed worst-case analysis and proves amazingly precise matching lower bounds.

AVERAGE CASE ANALYSIS OF THE EUCLIDEAN ALGORITHM

For $0 \le n \le m$, let $\tau : P_{m,n} \to 2^{\{0,1,\ldots,n-1\}}$ be the mapping such that $\tau(A_1,A_2) = \{\deg A_i \mid 3 \le i \le t\}$, where $(A_3,\ldots,A_t)$ is the resulting polynomial remainder sequence of (A_1,A_2) in the Euclidean algorithm. For any subset S of $\{0,1,\ldots,n-1\}$ of size $t-2$, say $S = \{s_3,\ldots,s_t\}$ with $s_3 > s_4 > \cdots > s_t$, we have $\#\tau^{-1}(S) = (p-1)^t p^m$. If $\tau(A_1,A_2) = S$, then (A_1,A_2) has the Euclidean representation pattern $(m-n, n-s_3, s_3-s_4, \ldots, s_{t-1}-s_t, s_t)$.

LEMMA 2.3. *For $1 \le n \le m$, let (A_1,A_2) be uniformly distributed in $P_{m,n}$ and $S = \tau(A_1,A_2)$. Then the n events "$k \in S$" $(0 \le k \le n-1)$ are independent, and each occurs with probability $1-1/p$.*

PROOF. Let $P^k_{m,n} = \{(A_1,A_2) \in P_{m,n} \mid k \in \tau(A_1,A_2)\}$. For any $S \subseteq \{0,\ldots,n-1\}$ with $k \notin S$, we have $\#\tau^{-1}(S \cup \{k\}) = (p-1)\,\#\tau^{-1}(S)$. Thus $\#P^k_{m,n} = (1-1/p)\,\#P_{m,n}$. □

Since $\mathbf{Pr}[0 \in S] = \mathbf{Pr}[\gcd(A_1,A_2) = 1]$, we obtain the following important proposition (see Knuth 1981, Ex. 4.6.1—5).

PROPOSITION 2.4. *For $1 \le n \le m$, let (A_1,A_2) be uniformly distributed in $P_{m,n}$. Then A_1 and A_2 are relatively prime with probability $1-1/p$.*

PROPOSITION 2.5. *For $0 \le k \le n \le m$, let (A_1,A_2) be uniformly distributed in $P_{m,n}$. Then*

$$\mathbf{Pr}[\deg\gcd(A_1,A_2) = k] = \begin{cases} (1-1/p)\,p^{-k}, & \text{if } k < n, \\ p^{-k}, & \text{if } k = n. \end{cases}$$

PROOF. For $0 \le k < n$, it follows from Lemma 2.3 that

$$\mathbf{Pr}[\deg\gcd(A_1,A_2) = k] = \mathbf{Pr}[k \in S, k-1 \notin S, \ldots, 0 \notin S] = (1-1/p)\,p^{-k},$$
$$\mathbf{Pr}[\deg\gcd(A_1,A_2) = n] = \mathbf{Pr}[n-1 \notin S, \ldots, 0 \notin S] = p^{-n}. \quad \square$$

COROLLARY 2.6. *For $0 \le n \le m$, let (A_1,A_2) be uniformly distributed in $P_{m,n}$. Then the average degree of $\gcd(A_1,A_2)$ is $(1-p^{-n})/(p-1) < 1$.*

PROOF. $\mathbf{E}[\deg\gcd(A_1,A_2)] = \sum_{k=0}^{n} k \times \mathbf{Pr}[\deg\gcd(A_1,A_2)=k]$. $\square$

PROOF OF THEOREM 2.1. For $\omega \in \{\mathrm{div}, \div, -, \times\}$ and $(A_1,A_2) \in P_{m,n}$ with resulting remainder sequence $(A_3,\ldots,A_t)$ and $S = \tau(A_1,A_2) = \{\deg A_i \mid 3 \le i \le t\} \subseteq \{0,1,\ldots,n-1\}$, we have defined the following functions $t^\omega : P_{m,n} \to \mathbb{N}$:

$$t^\omega(A_1,A_2) = \sum_{1\le i<t} d^\omega(\deg A_i, \deg A_{i+1}) = d^\omega(m,n) + \sum_{2\le i<t} d^\omega(\deg A_i,\deg A_{i+1}),$$

where $d^{\mathrm{div}}(j,k)=1$, $d^{\div}(j,k) = j-k+1$, and $d^{-,\times}(j,k) = k(j-k+1)$ for $0 \le k \le j \le m$. Thus $t^{\mathrm{div}}(A_1,A_2)$ equals the number of polynomial divisions used in the Euclidean algorithm, $t^{\div}(A_1,A_2)$ equals the number of operations $\div$ used in the ground field, and $t^{-,\times}(A_1,A_2)$ equals the number of operations $-, \times$ used.

In order to derive $t^\omega_{\mathrm{ave}}(m,n,p)$, the average value of $t^\omega(A_1,A_2)$ with (A_1,A_2) uniformly distributed in $P_{m,n}$, we define the following random variables d_k^ω and e_k $(0 \le k \le n-1)$:

$$d_k^\omega = \begin{cases} d^\omega(\deg A_i, \deg A_{i+1}), & \text{if } \deg A_{i+1} = k \in S \text{ for some } i,\\ 0, & \text{if } k \notin S,\end{cases}$$
$$e_k = \begin{cases} \deg A_i, & \text{if } \deg A_{i+1} = k \in S \text{ for some } i,\\ 0, & \text{if } k \notin S.\end{cases}$$

By Lemma 2.3, the events "$k \in S$" $(0 \le k \le n-1)$ are independent and each occurs with probability $1-1/p$. Therefore, for $k+1 \le j \le n-1$,

$$\begin{aligned}\mathbf{Pr}[e_k = j] &= \mathbf{Pr}[j \in S, j-1 \notin S, j-2 \notin S, \ldots, k+1 \notin S \mid k \in S]\\ &= (1-1/p)\,p^{k+1-j},\\ \mathbf{Pr}[e_k = n] &= \mathbf{Pr}[n-1 \notin S, n-2 \notin S, \ldots, k+1 \notin S \mid k \in S] = p^{k+1-n}.\end{aligned}$$

These expressions depend only on $j-k$ and $n-k$, and thus for fixed k and $k+1 \le j \le n$, the probability that $e_k = j$ equals the probability that $\deg\gcd(u,v) = j-k-1$ with (u,v) uniformly distributed in $P_{m,n-k-1}$. By Corollary 2.6,

$$\sum_{k<j\le n} \mathbf{Pr}[e_k = j] \times (j-k-1) = (1-p^{k+1-n})/(p-1). \tag{3}$$

By definition, $\mathbf{E}[d_k^\omega] = \sum_{k<j\le n} \mathbf{Pr}[e_k = j] \times d^\omega(j,k)$, and hence

$$\begin{aligned} t^\omega_{\mathrm{ave}}(m,n,p) &= d^\omega(m,n) + \sum_{0\le k<n} \mathbf{Pr}[k \in S] \times \mathbf{E}[d_k^\omega]\\ &= d^\omega(m,n) + (1-1/p) \sum_{0\le k<n}\sum_{k<j\le n} \mathbf{Pr}[e_k = j] \times d^\omega(j,k).\end{aligned}$$

We distinguish the following three cases for $\omega \in \{\mathrm{div}, \div, -, \times\}$.

Case 1: $\omega = \mathrm{div}$. Then $\mathbf{E}[d_k^{\mathrm{div}}] = 1$, and $t^{\mathrm{div}}_{\mathrm{ave}}(m,n,p) = 1 + (1-1/p)\,n$.

Case 2: $\omega = \div$. Then

$$t^{\div}_{\mathrm{ave}}(m,n,p) = m-n+1 + (1-1/p) \sum_{0\le k<n}\sum_{k<j\le n} \mathbf{Pr}[e_k = j] \times (j-k+1)$$

$$\begin{aligned} &= m-n+1+(1-1/p) \sum_{0\le k<n} [2+\sum_{k<j\le n} \mathbf{Pr}[e_k=j]\times(j-k-1)] \\ &= m-n+1+(1-1/p) \sum_{0\le k<n} [2+(1-p^{k+1-n})/(p-1)], \text{ by (3)} \\ &= m+(1-1/p)\,n+1-(1-p^{-n})/(p-1). \end{aligned}$$

Case 3: $\omega = -, \times$. Then

$$\begin{aligned} t_{\text{ave}}^{-,\times}(m,n,p) &= n\,(m-n+1)+(1-1/p) \sum_{0\le k<n} k \sum_{k<j\le n} \mathbf{Pr}[e_k=j]\times(j-k+1) \\ &= n\,(m-n+1)+(1-1/p) \sum_{0\le k<n} k[2+(1-p^{k+1-n})/(p-1)], \text{ by (3)} \\ &= m\,n-\tbinom{n}{2}/p-n/(p-1)+p(1-p^{-n})/(p-1)^2. \quad \square \end{aligned}$$

BEST AND WORST CASE OF THE EUCLIDEAN ALGORITHM

THEOREM 2.7. *For $0\le n\le m$ and $\omega\in\{\text{div},\div,-,\times\}$, let $t_{\min}^{\omega}(m,n,p)$ and $t_{\max}^{\omega}(m,n,p)$ be the minimum and maximum number, respectively, of operations ω used on inputs from $P_{m,n}$. Then*

$$\begin{aligned} t_{\min}^{\text{div}}(m,n,p) &= 1, & t_{\max}^{\text{div}}(m,n,p) &= n+1, \\ t_{\min}^{\div}(m,n,p) &= m-n+1, & t_{\max}^{\div}(m,n,p) &= m+n+1, \\ t_{\min}^{-,\times}(m,n,p) &= n\,(m-n+1), & t_{\max}^{-,\times}(m,n,p) &= m\,n. \end{aligned}$$

PROOF. For $\omega\in\{\text{div},\div,-,\times\}$ and $(A_1,A_2)\in P_{m,n}$, recall the definition of $t^{\omega}(A_1,A_2)$ in the proof of Theorem 2.1. For $0\le k\le n-1$ and any $S\subseteq\{0,\ldots,n-1\}$ with $k\notin S$, it is easy to see that $t^{\omega}(A_1,A_2)<t^{\omega}(B_1,B_2)$, where $\tau(A_1,A_2)=S$ and $\tau(B_1,B_2)=S\cup\{k\}$. This reveals that $t^{\omega}(A_1,A_2)$ is minimal if $\tau(A_1,A_2)=\emptyset$ and maximal if $\tau(A_1,A_2)=\{0,\ldots,n-1\}$. The claims follow immediately from a simple calculation. $\square$

The minimum computing time occurs in the Euclidean algorithm for the input (A_1,A_2) where A_2 divides A_1. The maximum computing time occurs, e.g., for the input $(A_1,A_2)=(x^{m-n}f_n+f_{n-1},f_n)$, where $f_n=\sum_{i=0}^{\lfloor n/2\rfloor}\binom{n-i}{i}x^{n-2i}$ is the nth Fibonacci polynomial satisfying the recurrence relation $f_n=x\,f_{n-1}+f_{n-2}$ with $f_0=1$ and $f_1=x$. [If g_n is the nth Chebyshev polynomial of the second kind, then $f_n=(-\sqrt{-1})^n g_n(\sqrt{-1}\,x)$.] This is similar to the fact that the maximum computing time occurs in the integer Euclidean algorithm for two consecutive Fibonacci numbers $f_{n+1}(1)$ and $f_n(1)$.

Collins (1974) shows that the maximum and average computing times of the integer Euclidean algorithm are of the same order. This is also true for the polynomial Euclidean algorithm in the sense that for fixed m,n and $\omega\in\{\text{div},\div,-,\times\}$ (see Theorem 2.1),

$$\lim_{p\to\infty}\frac{t_{\text{ave}}^{\omega}(m,n,p)}{t_{\max}^{\omega}(m,n,p)}=1.$$

A similar scenario occurs when the "degree size" m and n are of the same order of magnitude as the "coefficient size" $\log p$, since then:

$$\lim_{n\to\infty}\frac{t_{\text{ave}}^{\omega}(m,n,p)}{t_{\max}^{\omega}(m,n,p)}=1.$$

3. The Indeterminate Shift Euclidean Algorithm

For $q, r \in F[x]$ and $i \in \mathbb{N}$ with $q \notin F$ and $r \neq 0$, let $q^i \parallel r$ denote that q^i divides r but q^{i+1} does not. Define $\tilde{r} = r/x^i$ for $x^i \parallel r$ if $r \neq 0$, and $\tilde{0} = 0$. For $0 \leq n \leq m$, also let $\tilde{P}_m = \{A \in P_m \mid x \nmid A\}$ and $\tilde{P}_{m,n} = \tilde{P}_m \times \tilde{P}_n$. The indeterminate shift Euclidean algorithm is formulated as follows.

Algorithm 3.1. For $1 \leq n \leq m$ and the input $(A_1, A_2) \in P_{m,n}$, this algorithm computes $\gcd(A_1, A_2)$.

```
1. [Initialization] For $x^{l_1} \parallel A_1$ and $x^{l_2} \parallel A_2$, let $u \leftarrow A_1/x^{l_1}$, $v \leftarrow A_2/x^{l_2}$, $l \leftarrow \min(l_1, l_2)$.
   If $\deg u < \deg v$ then swap $u$ and $v$.
2. [Division and shifting] Find the remainder such that $u = q\,v + r$ and compute $\tilde{r}$. If
   $\tilde{r} = 0$ then stop and return $\gcd(A_1, A_2) = x^l\,v$.
3. [Resetting] Set $u \leftarrow v$, $v \leftarrow \tilde{r}$, and go to step 2.
```

We shall prove the following results:

Theorem 3.2. *Let $1 \leq n \leq m$ and (u, v) be uniformly distributed in $\tilde{P}_{m,n}$. Then*

$$\mathbf{Pr}[\gcd(u, v) = 1] = \begin{cases} [1 - 1/(p+1)]\,(1 - p^{-2n}), & \text{if } m > n, \\ [1 - 1/(p+1)]\,[1 - 2\,p^{1-2n}/(p-1)], & \text{if } m = n. \end{cases}$$

Remark 3.3. For (u, v) uniformly distributed in $\tilde{P}_{m,n}$,

$$\lim_{n \to \infty} \mathbf{Pr}[\gcd(u, v) = 1] = 1 - 1/(p+1) > 1 - 1/p.$$

Here $1 - 1/p$ is the probability that $\gcd(u, v) = 1$ with (u, v) uniformly distributed in $P_{m,n}$. In fact, the probability that $\gcd(u, v) = 1$ with (u, v) uniformly distributed in $\tilde{P}_{m,n}$ is strictly greater than the same probability with (u, v) uniformly distributed in $P_{m,n}$ when $n \geq 3$.

Theorem 3.4. *Let $1 \leq n \leq m$. The average number $t_{\text{ave}}^{\text{div}}(m, n, p)$ of polynomial divisions used in Algorithm 3.1 on inputs from $\tilde{P}_{m,n}$ is $[1 - 2/(p+1)]\,n + 1 + \alpha(m, n, p)$, where $\alpha(m, n, p) < 1$ with $\lim_{p \to \infty} \alpha(m, n, p) = 0$ is given by*

$$\alpha(m, n, p) = \begin{cases} (p - p^{1-2n})/(p+1)^2, & \text{if } m > n, \\ (p^2 - 2\,p - 1 - 2\,p^{2-2n})/[(p-1)\,(p+1)^2], & \text{if } m = n. \end{cases}$$

Remark 3.5. By Theorem 2.1, the average number $e_{\text{ave}}^{\text{div}}(m, n, p)$ of polynomial divisions used in the classical Euclidean algorithm with a uniformly distributed input from $P_{m,n}$ is $(1 - 1/p)\,n + 1$. For $2 \leq n \leq m$, $t_{\text{ave}}^{\text{div}}(m, n, p)$ is strictly smaller than $e_{\text{ave}}^{\text{div}}(m, n, p)$. However, for fixed m, n and large p, both algorithms require $n + 1$ polynomial divisions for almost every input in the sense that

$$\lim_{p \to \infty} t_{\text{ave}}^{\text{div}}(m, n, p) = \lim_{p \to \infty} e_{\text{ave}}^{\text{div}}(m, n, p) = n + 1.$$

For a fair comparison of the two algorithms, we should consider the average cost of Algorithm 3.1 on inputs (A_1, A_2) from $P_{m,n}$, which by definition equals

$$\sum_{\substack{0\le l_1\le m\\ 0\le l_2\le n}} \mathbf{Pr}[x^{l_1} \parallel A_1]\, \mathbf{Pr}[x^{l_1} \parallel A_2]\, t_{\text{ave}}^{\text{div}}(m-l_1, n-l_2, p)$$

with $t_{\text{ave}}^{\text{div}}(m-l_1, n-l_2, p) = t_{\text{ave}}^{\text{div}}(n-l_2, m-l_1, p)$ if $m-l_1 < n-l_2$. However, a calculation shows that this quantity equals $[1-2/(p+1)]\,n + 1 + \beta(m,n,p)$, where $\beta(m,n,p) < \alpha(m,n,p) < 1$ with $\lim_{p\to\infty} \beta(m,n,p) = 0$. Therefore, the average computing time of Algorithm 3.1 with inputs from $P_{m,n}$ is dominated by the computation for those inputs from $\tilde{P}_{m,n}$.

THEOREM 3.6. *Let $1 \le n \le m$. The average number of arithmetic operations used in Algorithm 3.1 on inputs from $\tilde{P}_{m,n}$ is*

$$\begin{aligned} t_{\text{ave}}^{-,\times}(m,n,p) &= m\,n - n^2/(p+1) - [(p^2+4\,p+1)/(p-1)\,(p+1)^2]\,n + \gamma(m,n,p),\\ t_{\text{ave}}^{\div}(m,n,p) &= m + [1-2/(p+1)]\,n + 1 + \epsilon(m,n,p), \end{aligned}$$

where $\gamma(m,n,p) < 4$ and $\epsilon(m,n,p) < 1$ are given by

$$\gamma(m,n,p) = \begin{cases} -2\,\frac{p^{1-n}}{(p-1)^2} + 2\,\frac{p^{2-2\,n}}{(p-1)^2\,(p+1)^3} + 2\,\frac{p\,(p^3+3\,p^2+2\,p+1)}{(p-1)^2\,(p+1)^3}, & \text{if } m > n,\\ -\frac{1}{p^2-1}\,n + \frac{p^{1-n}}{(p-1)^2}\,(n-2) + 4\,\frac{p^{3-2\,n}}{(p+1)^3\,(p-1)^3} + 2\,\frac{p\,(p^4+2\,p^3-2\,p^2-2\,p-1)}{(p+1)^3\,(p-1)^3}, & \end{cases}$$

$$\epsilon(m,n,p) = \begin{cases} 2\,\frac{p^{1-2n}}{(p+1)^2\,(p-1)} + 2\,\frac{p^2-p-1}{(p+1)^2\,(p-1)}, & \text{if } m > n,\\ -\frac{p^{1-n}}{(p-1)^2} + 4\,\frac{p^{2-2n}}{(p+1)^2\,(p-1)^2} + \frac{2\,p^2-3\,p-3}{(p+1)^2\,(p-1)}, & \text{if } m = n. \end{cases}$$

REMARK 3.7. For $\omega \in \{-,\times,\div\}$, let $e_{\text{ave}}^{\omega}(m,n,p)$ be the average number of operations ω used in the classical Euclidean algorithm with a uniformly distributed input from $P_{m,n}$. Then Theorems 2.1 and 3.6 reveal that for fixed m, n and large p, both algorithms need the same number of arithmetic operations in the ground field for almost every input, i.e., $m\,n$ for operations $-, \times$ and $m+n+1$ for operations $\div$, in the sense that

$$\begin{aligned} \lim_{p\to\infty} e_{\text{ave}}^{-,\times}(m,n,p) &= \lim_{p\to\infty} t_{\text{ave}}^{-,\times}(m,n,p) = m\,n,\\ \lim_{p\to\infty} e_{\text{ave}}^{\div}(m,n,p) &= \lim_{p\to\infty} t_{\text{ave}}^{\div}(m,n,p) = m+n+1. \end{aligned}$$

Consequently, Algorithms 3.1 does not attain a significant reduction in computing time unless p is quite small. For the finite field $F = \mathbf{Z}_2$ with only two elements, however, the average computing time of Algorithm 3.1 is significantly smaller than that of the classical Euclidean algorithm, as demonstrated in the following two facts (derived from Proposition 2.4, Theorems 2.1, 3.2, 3.4 and 3.6 by letting $p = 2$).

FACT 3.8. Let $F = \mathbf{Z}_2$, and the input (A_1, A_2) be uniformly distributed in $P_{m,n}$ with degrees $m \ge n \ge 1$ respectively. Then the probability that $\gcd(A_1, A_2) = 1$ is $\frac{1}{2}$; the average number of polynomial divisions used in the Euclidean algorithm is $\frac{1}{2}\,n + 1$; the average number of operations $-, \times$ used is $m\,n - \frac{1}{4}\,n^2 - \frac{3}{4}\,n + O(1)$; the average number of operations $\div$ used is $m + \frac{1}{2}\,n + O(1)$.

FACT 3.9. Let $F = \mathbb{Z}_2$, and the input (A_1, A_2) be uniformly distributed in $\tilde{P}_{m,n}$ with degrees $m \geq n \geq 1$ respectively. Then the probability that $\gcd(A_1, A_2) = 1$ is $\frac{2}{3} + O(4^{-n})$; the average number of polynomial divisions used in the indeterminate shift Euclidean algorithm is $\frac{1}{3}n + O(1)$; the average number of operations $-, \times$ used is $mn - \frac{1}{3}n^2 - \frac{13}{9}n + O(1)$ for $m > n$, and $mn - \frac{1}{3}n^2 - \frac{16}{9}n + O(1)$ for $m = n$; the average number of operations $\div$ used is $m + \frac{1}{3}n + O(1)$.

AVERAGE CASE ANALYSIS OF ALGORITHM 3.1

In order to analyze Algorithm 3.1, we use the *lattice model* suggested by Knuth (1981, pp. 330-336) for examining the binary Euclidean algorithm for integers. Assuming that (A_1, A_2) is uniformly distributed in $\tilde{P}_{m,n}$, we consider Algorithm 3.1 with input (A_1, A_2). To every pass through step 2, we associate with (u, v) the point $(\deg u, \deg v) \in \mathbb{N}^2$ and trace the passage of the algorithm through points of the *lattice* $\mathbb{N}^2$. Throughout the algorithm, we have $x \nmid u$ and $x \nmid v$. If (u, v) is uniformly distributed in $\tilde{P}_{i,j}$ and $\deg \tilde{r} = k$, then $(v, \tilde{r})$ is uniformly distributed in $\tilde{P}_{j,k}$. Thus an analysis of Algorithm 3.1 can be performed by solving the following recurrence relation:

$$S_{mn} = a + \sum_{0 \leq k < n} \mathbf{Pr}[\deg \tilde{r} = k]\, S_{nk}, \quad m \geq n \geq 1. \tag{4}$$

This recurrence relation is quite informative about the indeterminate shift Euclidean algorithm. It will follow from Corollary 3.12 that for $S_{n0} = 1$, S_{mn} is the average number of polynomial divisions used for computing $\gcd(A_1, A_2)$ if $a = 1$; S_{mn} is the probability that $\gcd(A_1, A_2) = 1$ if $a = 0$. Assuming the "synthetic" polynomial division algorithm and free indeterminate shifting, S_{mn} is the average number of operations $-, \times$ performed in Algorithm 3.1 if $a = n(m-n+1)$ and $S_{n0} = 0$; S_{mn} is the average number of operations $\div$ if $a = m - n + 1$ and $S_{n0} = n + 1$.

Let $R_n = \{A \in F[x] \mid \deg A < n\}$, $\tilde{R}_n = \{A \in R_n \mid A(0) \neq 0\}$, and $R_n^0 = R_n - \tilde{R}_n$. For $k \geq 0$, $v \in \tilde{P}_n$ and $r \in \tilde{R}_n$, also let $Q_k(v, r) = \{q \in P_k \mid q(0)\, v(0) + r(0) \neq 0\}$. Then

$$\#Q_k(v, r) = \begin{cases} p - 2, & \text{if } k = 0, \\ (p-1)^2 p^{k-1}, & \text{if } k \geq 1. \end{cases} \tag{5}$$

For $\omega \in \{+, \times\}$ and $F_1, F_2 \subseteq F[x]$, let $F_1 \omega F_2 = \{f_1 \omega f_2 \mid f_1 \in F_1, f_2 \in F_2\}$. The following lemma expresses in our notation the unique existence of quotients and remainders in polynomial divisions.

LEMMA 3.10. *Let $1 \leq n \leq m$ and $v \in \tilde{P}_n$. Then*

$$\tilde{P}_m = \left(\{v\} \times \tilde{P}_{m-n} + R_n^0\right) \bigcup \left(\bigcup_{r \in \tilde{R}_n} [\{v\} \times Q_{m-n}(v, r) + \{r\}] \right) \quad \textit{is a disjoint union.}$$

COROLLARY 3.11. *For $1 \leq n \leq m$ and uniformly distributed $(u, v) \in \tilde{P}_{m,n}$, let r be the remainder of u divided by v. Then for $f \in R_n$,*

$$\mathbf{Pr}[r = f] = \begin{cases} p^{-n}, & \textit{if } m > n \textit{ and } f \in R_n, \\ p^{1-n}/(p-1), & \textit{if } m = n \textit{ and } f \in R_n^0, \\ (p-2)\, p^{1-n}/(p-1)^2, & \textit{if } m = n \textit{ and } f \in \tilde{R}_n. \end{cases}$$

Proof. Applying Lemma 3.10, we have

$$\mathbf{Pr}[r=f]=\begin{cases} \#\tilde{P}_{m-n}/\#\tilde{P}_m, & \text{if } m>n \text{ and } f\in R_n^0,\\ \#Q_{m-n}(v,f)/\#\tilde{P}_m, & \text{if } m>n \text{ and } f\in \tilde{R}_n,\\ \#\tilde{P}_0/\#\tilde{P}_m, & \text{if } m=n \text{ and } f\in R_n^0,\\ \#Q_0(v,f)/\#\tilde{P}_m, & \text{if } m=n \text{ and } f\in \tilde{R}_n.\end{cases}$$

Since $\#\tilde{P}_m=(p-1)^2\,p^{m-1}$ and $\#\tilde{P}_0=p-1$, we complete the proof using (5). □

Corollary 3.12. *Let m,n,u,v,r be as above, and let $\tilde{r}=r/x^i$ where $x^i \parallel r$ if $r\neq 0$, and $\tilde{0}=0$. Then, for $1\le k\le n-1$,*

$$\mathbf{Pr}[\tilde{r}=0] = \begin{cases} p^{-n}, & \text{if } m>n,\\ p^{1-n}/(p-1), & \text{if } m=n,\end{cases}$$

$$\mathbf{Pr}[\deg\tilde{r}=0] = \begin{cases} (p-1)\,n\,p^{-n}, & \text{if } m>n,\\ [(p-1)\,n-1]\,p^{1-n}/(p-1), & \text{if } m=n,\end{cases}$$

$$\mathbf{Pr}[\deg\tilde{r}=k] = \begin{cases} (p-1)^2\,(n-k)\,p^{k-n-1}, & \text{if } m>n,\\ [(p-1)\,(n-k)-1]\,p^{k-n}, & \text{if } m=n.\end{cases}$$

Proof. By definition, $\mathbf{Pr}[\tilde{r}=0]=\mathbf{Pr}[r=0]$. Also, for $1\le k\le n-1$,

$$\mathbf{Pr}[\deg\tilde{r}=0]=\sum_{w\in F\setminus\{0\}}\sum_{0\le i<n}\mathbf{Pr}[r=w\,x^i],\ \mathbf{Pr}[\deg\tilde{r}=k]=\sum_{f\in\tilde{P}_k}\sum_{0\le i<n-k}\mathbf{Pr}[r=x^i f].$$

The claims follow from a straightforward calculation using Corollary 3.11. □

Proof of Theorems 3.2 and 3.4. Let $S_{n0}=1$ in (4). Applying Corollary 3.12, we obtain the following set of double recurrence relations:

$$S_{mn} = a+\frac{(p-1)\,n}{p^n}+\sum_{1\le k<n}\frac{(p-1)^2\,(n-k)}{p^{n+1-k}}\,S_{nk},\ m>n\ge 1, \tag{6}$$

$$S_{mn} = a+\frac{(p-1)\,n-1}{(p-1)\,p^{n-1}}+\sum_{1\le k<n}\frac{(p-1)\,(n-k)-1}{p^{n-k}}\,S_{nk},\ m=n\ge 1. \tag{7}$$

Note that $S_{mn}=S_{n+1,n}$ for $m>n$. Let $S_n=S_{n+1,n}$ and $S'_n=S_{nn}$. This substitution into (6) and (7) gives

$$S_n = a+\frac{(p-1)\,n}{p^n}+\frac{(p-1)^2}{p^{n+1}}\sum_{1\le k<n}(n-k)\,p^k\,S_k,\quad n\ge 1, \tag{8}$$

$$S'_n = a+\frac{(p-1)\,n-1}{(p-1)\,p^{n-1}}+\frac{1}{p^n}\sum_{1\le k<n}[(p-1)\,(n-k)-1]\,p^k\,S_k,\quad n\ge 1.$$

The above recurrences can be solved directly via generating functions; however, we prefer an alternative approach (see Knuth 1981, p. 332) which transform them first into linear recurrence relations through the following manipulation:

$$\begin{aligned} S_n - \frac{1}{p} S_{n-1} &= (1-\frac{1}{p})\, a + \frac{p-1}{p^n} + \frac{(p-1)^2}{p^{n+1}} \sum_{1\le k<n} p^k S_k, \quad n \ge 2, \\ S'_n - \frac{p}{p-1} S_n &= -\frac{1}{p-1}\, a - \frac{1}{(p-1)\, p^{n-1}} - \frac{1}{p^n} \sum_{1\le k<n} p^k S_k, \quad n \ge 1. \end{aligned}$$

It follows that for $n \ge 3$, $(S_n - \frac{1}{p} S_{n-1}) - \frac{1}{p}(S_{n-1} - \frac{1}{p} S_{n-2}) = (1-\frac{1}{p})^2 a + (1-\frac{1}{p})^2 S_{n-1}$. Note that for $n \ge 2$,

$$\frac{1}{p^n} \sum_{1\le k<n} p^k S_k = \frac{p}{(p-1)^2} \left[S_n - \frac{1}{p} S_{n-1} - (1-\frac{1}{p})\, a - \frac{p-1}{p^n} \right].$$

A straightforward calculation yields

$$\begin{aligned} S_n &= (1+\frac{1}{p^2})\, S_{n-1} - \frac{1}{p^2} S_{n-2} + (1-\frac{1}{p})^2 a, \quad n \ge 3, \\ S'_n &= \frac{(p-2)\, p}{(p-1)^2} S_n + \frac{1}{(p-1)^2} S_{n-1}, \quad n \ge 2. \end{aligned}$$

Solving these linear recurrence relations with the initial values $S_1 = a + 1 - 1/p$ and $S_2 = 2a + 1 - [(2a+1)p^2 - (a+1)p + 1]/p^3$ from (8), we obtain

$$\begin{aligned} S_n &= (1 - \tfrac{2}{p+1})\, a\, n + \tfrac{(a-1)p-1}{(p+1)^2} p^{1-2n} + \tfrac{p^2+(2a+1)p+a}{(p+1)^2}, \quad \text{if } n \ge 1, \\ S'_n &= (1 - \tfrac{2}{p+1})\, a\, n + 2\, \tfrac{(a-1)p-1}{(p+1)^2 (p-1)} p^{2-2n} + \tfrac{p^3+2ap^2-(2a+1)p-2a}{(p+1)^2 (p-1)}, \quad \text{if } n \ge 2. \end{aligned}$$

We observe that the above formula for S'_n also holds for $n = 1$. Let $a = 0$. Then the probability that $\gcd(A_1, A_2) = 1$ is S_n for $m > n$ and S'_n for $m = n$. This completes the proof of Theorem 3.2. The same argument with $a = 1$ proves Theorem 3.4. □

The proof of Theorem 3.6 is similar and can be found in Ma (1987, pp. 43-51).

Best and worst case of Algorithm 3.1

Let $1 \le n \le m$ and $(A_1, A_2) \in \tilde{P}_{m,n}$. For $F \ne \mathbb{Z}_2$, the indeterminate shift Euclidean algorithm has the same minimum and maximum computing times as the classical Euclidean algorithm (see Theorem 2.7). The minimum computing time occurs for the input (A_1, A_2) where A_2 divides A_1. The maximum computing time occurs, e.g., for the following input (A_1, A_2): $A_1 = a\, x^{m-n} f'_n + f'_{n-1}$ and $A_2 = f'_n$, where $a \in F\backslash\{0, -1\}$ and

$$f'_n = \sum_{0\le i\le n} \binom{n-\lfloor (i+1)/2 \rfloor}{\lfloor i/2 \rfloor} x^{n-i} = x\, f'_{n-1} + f'_{n-2}$$

with the initial values $f'_0 = 1$ and $f'_1 = x + 1$. It is easy to see that $f'_n = f_n + f_{n-1}$, where f_n is the n^{th} Fibonacci polynomial defined in Remark 2.2.

The situation with $F = \mathbb{Z}_2$ (i.e., $p = 2$) is slightly different. Clearly, both algorithms still have the same minimum computing time for all $1 \le n \le m$. However, the two algorithms have the same maximum computing time only for $m > n$, which occurs, for example, when the input (A_1, A_2) in Algorithm 3.1 is $A_1 = x^{m-n} f'_n + f'_{n-1}$ and $A_2 = f'_n$.

In the case $m = n$, Algorithm 3.1 uses at most n polynomial divisions rather than $n+1$ in the Euclidean algorithm. As an example, we can construct the following worst-case input for $m = n \geq 3$: $A_1 = x^2 f'_{n-2} + f'_{n-3} + x f'_{n-2}$ and $A_2 = x^2 f'_{n-2} + f'_{n-3}$. This gives

$$\begin{aligned} t^{\div}_{\max}(n,n,p) &= 1 + t^{\div}_{\max}(n, n-2, p) = 1 + n + (n-2) + 1 = 2n, \\ t^{-,\times}_{\max}(n,n,p) &= n + t^{-,\times}_{\max}(n, n-2, p) = n + n(n-2) = n^2 - n, \end{aligned}$$

which are just slightly better than $e^{\div}_{\max}(n,n,p) = 2n+1$ and $e^{-,\times}_{\max}(n,n,p) = n^2$ in the classical Euclidean algorithm.

4. The Linear Factor Shift Euclidean Algorithm

The indeterminate shift Euclidean algorithm in Section 3 can be extended to perform linear factor shifting, namely, to remove all linear factors from the resulting polynomial remainder after each division. We have not been able to give a complete and exact analysis for this new algorithm over an arbitrary finite field, but we do so for $F = \mathbb{Z}_2$. Polynomial arithmetic over $\mathbb{Z}_2$ has a strong analogy to binary arithmetic. In this case, subtractions and additions in the ground field are the same operations, and each $-, \times$ operation amounts to just one bit operation. Also, the computation of a polynomial remainder is equivalent to a sequence of conditional polynomial subtractions. Therefore, there is no need to count arithmetic operations like $\div, \times$ for our Euclidean algorithms over $\mathbb{Z}_2$.

The following proposition gives an estimate on linear factors in a uniformly distributed polynomial over any finite field F with $p \geq 2$ elements (Knuth 1981, Ex. 4.6.2—1; Ma 1987, pp. 53-55).

PROPOSITION 4.1. *Let $n \geq 2$ and f be a uniformly distributed polynomial from P_n. Then the probability that f has a linear factor is $1 - \sum_{i=0}^{n} (-1)^i \binom{p}{i} p^{-i} > 1 - e^{-1} > 1/2$; the average number of linear factors in f is $p(1-p^{-n})/(p-1) > 1$.*

In particular, for $F = \mathbb{Z}_2$ the probability that f has a linear factor is $3/4$, and the average number of linear factors in f is $2(1-2^{-n})$.

Let $f = x^{m_k} + x^{m_{k-1}} + \cdots + x^{m_1} \in \mathbb{Z}_2[x]$, $n = m_k > m_{k-1} > \cdots > m_1 \geq 0$. If k is odd then f has no factor $x+1$. Otherwise, the following holds:

$$\begin{aligned} \frac{f}{x+1} = {} & x^{m_k - 1} + x^{m_k - 2} + \cdots + x^{m_{k-1}} \\ & + x^{m_{k-2} - 1} + x^{m_{k-2} - 2} + \cdots + x^{m_{k-3}} \\ & \vdots \\ & + x^{m_2 - 1} + x^{m_2 - 2} + \cdots + x^{m_1}. \end{aligned}$$

This suggests that we can remove the factor $x+1$ from a polynomial over $\mathbb{Z}_2$ through some shifting and copying operations, and the cost of doing this is $O(n)$ bit operations. For the convenience of analysis, we will assume that this removal is free of charge. The skeptical reader might argue that the savings reported for the new algorithms in Sections 4 and 6 are due to this admittedly optimistic assumption. However, one can imagine fine-tuned fast subroutines (or maybe even hardware) to perform these shifts very quickly; they would not be of (asymptotic) help in the standard Euclidean algorithm. It should be pointed out that for an arbitrary finite field with $p > 2$ elements, the cost of removing linear factors cannot be ignored.

The linear factor shift Euclidean algorithm in the following is formulated only for $\mathbb{Z}_2$, but can be easily extended for any finite field.

For $2 \leq n \leq m$, let $P^{\emptyset}_{m,n} = P^{\emptyset}_{m} \times P^{\emptyset}_{n}$ where $P^{\emptyset}_{m}$ is the set of linear factor free polynomials of degree m. For $r \in \mathbb{Z}_2[x]$ and $r \neq 0$, let $\bar{r} = r/[x^{k_1}(x+1)^{k_2}]$ where $x^{k_1} \parallel r$ and $(x+1)^{k_2} \parallel r$, and $\bar{0} = 0$.

ALGORITHM 4.2. Given the input $(A_1, A_2) \in P_{m,n}$ $(m \geq n \geq 2)$, this algorithm outputs $\gcd(A_1, A_2)$.

1. [Initialization] For $x^{k_1} \parallel A_1$, $x^{k_2} \parallel A_2$, $(x+1)^{l_1} \parallel A_1$, and $(x+1)^{l_2} \parallel A_2$, let $u \leftarrow A_1/[x^{k_1}(x+1)^{l_1}]$, $v \leftarrow A_2/[x^{k_2}(x+1)^{l_2}]$, $k \leftarrow \min(k_1, k_2)$, and $l \leftarrow \min(l_1, l_2)$. If $\deg v > \deg u$ then swap u and v.

2. [Division and shifting] Find the remainder $r \in R_n$ such that $u = q\,v + r$ and compute $\bar{r}$. If $\bar{r} = 0$, then stop and return $\gcd(A_1, A_2) = x^k (x+1)^l v$.

3. [Resetting] Set $u \leftarrow v$, $v \leftarrow \bar{r}$. Go to step 2.

THEOREM 4.3. *Let $2 \leq n \leq m$ and (A_1, A_2) be uniformly distributed in $P^{\emptyset}_{m,n}$. Then the probability that* $\gcd(A_1, A_2) = 1$ *equals* $\frac{8}{9} + \kappa(m,n)$, *where* $\kappa(m,n) \leq \frac{1}{9}$ *with* $\lim_{n\to\infty} \kappa(m,n) = 0$ *is given by*

$$\kappa(m,n) = \begin{cases} \frac{1}{9}(3n-11)\,4^{1-n}, & \text{if } m > n+1, \\ \frac{1}{9}(3n-5)\,4^{2-n}, & \text{if } m = n+1, \\ -\frac{2}{9}(3n-2)\,4^{2-n}, & \text{if } m = n. \end{cases}$$

THEOREM 4.4. *Let $2 \leq n \leq m$, and $\delta_n = \frac{1}{4}$ if $n = 2$, and $\delta_n = 0$ otherwise. Then the average number $t^{\mathrm{div}}_{\mathrm{ave}}(m,n)$ of polynomial divisions used in Algorithm 4.2 on inputs from $P^{\emptyset}_{m,n}$ is $\frac{1}{4}n + \lambda(m,n)$, where $\lambda(m,n) < 2$ is given by*

$$\lambda(m,n) = \begin{cases} \frac{43}{36} - \frac{1}{9}(3n-17)\,4^{1-n} - \delta_n, & \text{if } m > n+1, \\ \frac{43}{36} - \frac{1}{9}(3n-11)\,4^{2-n} - \delta_n, & \text{if } m = n+1, \\ \frac{25}{36} + \frac{2}{9}(3n-8)\,4^{2-n} + \delta_n, & \text{if } m = n. \end{cases}$$

THEOREM 4.5. *Let $2 \leq n \leq m$, and δ_n as above. Then the average number $t^{-}_{\mathrm{ave}}(m,n)$ of operations $-$ used in Algorithm 4.2 on inputs from $P^{\emptyset}_{m,n}$ is $m\,n - \frac{3}{8}n^2 - \frac{49}{24}n + \mu(m,n)$, where $\mu(m,n) < 5$ is given by*

$$\mu(m,n) = \begin{cases} \frac{73}{18} - \frac{2}{9}4^{2-n} - \delta_n, & \text{if } m > n+1, \\ \frac{85}{18} - (n-2)\,2^{3-n} - \frac{2}{9}4^{3-n} - \delta_n, & \text{if } m = n+1, \\ \frac{5}{9} - \frac{1}{2}n + (3n-5)\,2^{3-n} + \frac{1}{9}4^{4-n} + \delta_n, & \text{if } m = n. \end{cases}$$

REMARK 4.6. Theorem 4.4 indicates that the linear factor shift Euclidean algorithm runs on average twice as fast as the classical Euclidean algorithm in the sense of number of polynomial divisions used (divisions by x and $x+1$ are not counted). In comparison with the Euclidean algorithm and the indeterminate shift Euclidean algorithm (see Facts 3.8 and 3.9), the linear factor shift Euclidean algorithm uses on average the fewest arithmetic operations in $\mathbb{Z}_2$, while the Euclidean algorithm uses the most arithmetic operations.

Our results are stated for inputs from $P^{\emptyset}_{m,n}$, for which the initialization step is trivial. It is then easy to obtain the corresponding results for average inputs from $P_{m,n}$ (see Remark 3.5).

AVERAGE CASE ANALYSIS OF ALGORITHM 4.2

As in Section 3, we can analyze the linear factor shift Euclidean algorithm by solving the following recurrence relation:

$$S_{mn} = a + \sum_{0\le k<n} \mathbf{Pr}[\deg \bar{r} = k]\, S_{nk}, \quad m \ge n \ge 2. \tag{9}$$

Assuming that (A_1, A_2) is uniformly distributed in $P^{\emptyset}_{m,n}$, Corollary 4.9 below yields that for $S_{n0} = 1$, S_{mn} is the average number of polynomial divisions used in Algorithm 4.2 for computing $\gcd(A_1, A_2)$ if $a = 1$; S_{mn} is the probability that $\gcd(A_1, A_2) = 1$ if $a = 0$. If the "synthetic" polynomial division algorithm is used, then S_{mn} is the average number of operations $-$ used in Algorithm 4.2 for $a = n\,(m-n+1)$ and $S_{n0} = 0$.

For $n \ge 2$ and $z \in 2^{\mathbb{Z}_2}$, let $\{P^{\emptyset}_n, P^0_n, P^1_n, P^{01}_n\}$ and $\{R^{\emptyset}_n, R^0_n, R^1_n, R^{01}_n\}$ be the following partitions of P_n and R_n:

$$\begin{aligned} P^z_n &= \{f \in P_n \mid \forall a \in \mathbb{Z}_2\ f(a) = 0 \iff a \in z\}, \\ R^z_n &= \{f \in R_n \mid \forall a \in \mathbb{Z}_2\ f(a) = 0 \iff a \in z\}. \end{aligned}$$

It is easy to see that each of the eight sets P^z_n and R^z_n contains exactly 2^{n-2} polynomials. The following lemma expresses in our notation the unique existence of quotients and remainders of polynomial divisions.

LEMMA 4.7. *Let $2 \le n \le m$ and $v \in P^{\emptyset}_n$. Then $P^{\emptyset}_m = (\{v\} \times P^{\emptyset}_{m-n} + R^{01}_n) \cup (\{v\} \times P^0_{m-n} + R^1_n) \cup (\{v\} \times P^1_{m-n} + R^0_n) \cup (\{v\} \times P^{01}_{m-n} + R^{\emptyset}_n)$ is a disjoint union.*

COROLLARY 4.8. *For $2 \le n \le m$ and uniformly distributed $(u, v) \in P^{\emptyset}_{m,n}$, let r be the remainder of u divided by v. Then r is uniformly distributed in R_n if $m > n+1$; r is uniformly distributed in $R^0_n \cup R^1_n$ if $m = n+1$; r is uniformly distributed in R^{01}_n if $m = n$.*

COROLLARY 4.9. *Let m, n, u, v, r be as above, and let $\bar{r} = r/[x^{k_1}(x+1)^{k_2}]$, where $x^{k_1} \parallel r$ and $(x+1)^{k_2} \parallel r$ if $r \ne 0$, and $\bar{0} = 0$. Then $\mathbf{Pr}[\deg \bar{r} = 1] = 0$, and for $2 \le k \le n-1$,*

$$\mathbf{Pr}[\bar{r} = 0] = \begin{cases} 2^{-n}, & \text{if } m > n+1, \\ 0, & \text{if } m = n+1, \\ 2^{2-n}, & \text{if } m = n, \end{cases}$$

$$\Pr[\deg \bar{r} = 0] = \begin{cases} n\,(n+1)\,2^{-n-1}, & \textit{if } m > n+1, \\ (n-1)\,2^{2-n}, & \textit{if } m = n+1, \\ (n-2)\,(n-1)\,2^{1-n}, & \textit{if } m = n, \end{cases}$$

$$\Pr[\deg \bar{r} = k] = \begin{cases} (n-k)\,(n-k+1)\,2^{k-n-3}, & \textit{if } m > n+1, \\ (n-k-1)\,2^{k-n}, & \textit{if } m = n+1, \\ (n-k-2)\,(n-k-1)\,2^{k-n-1}, & \textit{if } m = n. \end{cases}$$

PROOF. By definition, $\Pr[\deg \bar{r} = 1] = 0$, and $\Pr[\bar{r} = 0] = \Pr[r = 0]$. Also,

$$\Pr[\deg \bar{r} = 0] = \sum_{0 \le i < n} \sum_{k_1 + k_2 = i} \Pr[r = x^{k_1}\,(x+1)^{k_2}],$$
$$\Pr[\deg \bar{r} = k] = \sum_{f \in P_k^{\emptyset}} \sum_{0 \le i < n-k} \sum_{k_1 + k_2 = i} \Pr[r = x^{k_1}\,(x+1)^{k_2}\, f].$$

A simple calculation using Corollary 4.8 completes the proof. □

PROOF OF THEOREMS 4.3 AND 4.4. Let $S_{n0} = 1$ in (9). Applying Corollary 4.9, we obtain the following set of triple recurrence relations for $n \ge 2$:

$$S_{mn} = a + \frac{n\,(n+1)}{2^{n+1}} + \sum_{2 \le k < n} \frac{(n-k)\,(n-k+1)}{2^{n-k+3}}\, S_{nk}, \quad \text{if } m > n+1,$$
$$S_{mn} = a + \frac{n-1}{2^{n-2}} + \sum_{2 \le k < n} \frac{(n-k-1)}{2^{n-k}}\, S_{nk}, \quad \text{if } m = n+1,$$
$$S_{mn} = a + \frac{(n-2)\,(n-1)}{2^{n-1}} + \sum_{2 \le k < n} \frac{(n-k-2)\,(n-k-1)}{2^{n-k+1}}\, S_{nk}, \quad \text{if } m = n.$$

Note that $S_{mn} = S_{n+2,n}$ for $m \ge n+2$. Let $S_n = S_{n+2,n}$, $S'_n = S_{n+1,n}$ and $S''_n = S_{nn}$. This substitution into the above recurrence relations gives

$$S_n = a + \frac{n\,(n+1)}{2^{n+1}} + \frac{1}{2^{n+3}} \sum_{2 \le k \le n-2} (n-k)\,(n-k+1)\,2^k\, S_k + \frac{1}{8}\, S'_{n-1}, \qquad (10)$$
$$S'_n = a + \frac{(n-1)}{2^{n-2}} + \frac{1}{2^n} \sum_{2 \le k \le n-2} (n-k-1)\,2^k\, S_k, \qquad (11)$$
$$S''_n = a + \frac{(n-2)\,(n-1)}{2^{n-1}} + \frac{1}{2^{n+1}} \sum_{2 \le k \le n-2} (n-k-2)\,(n-k-1)\,2^k\, S_k.$$

By a manipulation similar to the one used in the proof of Theorems 3.2 and 3.4, we can transform the above recurrence relations into the following:

$$S_n - \tfrac{3}{2}\, S_{n-1} + \tfrac{9}{16}\, S_{n-2} - \tfrac{1}{16}\, S_{n-3} = \tfrac{9}{64}\, a, \quad \text{if } n \ge 6,$$
$$S'_n - S'_{n-1} + \tfrac{1}{4}\, S'_{n-2} = \tfrac{1}{4}\, a + \tfrac{1}{4}\, S_{n-2}, \quad \text{if } n \ge 5,$$
$$S''_n = S_{n-2} + \tfrac{1}{8}\, S_{n-3} - \tfrac{1}{8}\, S'_{n-3}, \quad \text{if } n \ge 5.$$

Solving these linear recurrence relations with the initial values $S_3 = \frac{9}{8}a + \frac{7}{8}$, $S_4 = \frac{21}{16}a + \frac{57}{64}$, $S_5 = \frac{199}{128}a + \frac{57}{64}$, $S_3' = a + 1$ and $S_4' = \frac{5}{4}a + \frac{15}{16}$ from (10) and (11), we obtain

$$\begin{aligned} S_n &= \tfrac{1}{4}an + \tfrac{1}{3}(-2a+1)n4^{1-n} + \tfrac{1}{9}(28a-11)4^{1-n} + \tfrac{11}{36}a + \tfrac{8}{9}, && \text{if } n \geq 3,\\ S_n' &= \tfrac{1}{4}an + \tfrac{1}{3}(-2a+1)n4^{2-n} + \tfrac{1}{9}(16a-5)4^{2-n} + \tfrac{11}{36}a + \tfrac{8}{9}, && \text{if } n \geq 3,\\ S_n'' &= \tfrac{1}{4}an + \tfrac{2}{3}(2a-1)n4^{2-n} + \tfrac{1}{9}(-5a+1)4^{3-n} - \tfrac{7}{36}a + \tfrac{8}{9}, && \text{if } n \geq 5. \end{aligned}$$

We observe that the above formula for S_n'' also holds for $n = 3, 4$. Let $a = 0$. Then the probability is S_n for $m > n+1$, S_n' for $m = n+1$, and S_n'' for $m = n$. Since $S_2 = \frac{3}{4}$, $S_2' = 1$ and $S_2'' = 0$, this completes the proof of Theorem 4.3. The same argument with $a = 1$ proves Theorem 4.4. □

The proof of Theorem 4.5 is similar and is omitted here (see Ma 1987, pp. 67-72).

Best and worst case of Algorithm 4.2

For $a, b \in \mathbb{Z}$ with $b \geq 1$ we denote by $a \operatorname{rem} b$ the remainder of a on division by b, i.e., $c = a \operatorname{rem} b$ if and only if $0 \leq c < b$ and $c \equiv a \bmod b$.

Theorem 4.10. *For $2 \leq n \leq m$ and $\omega \in \{\text{div}, -\}$, let $t^{\omega}_{\min}(m,n)$ and $t^{\omega}_{\max}(m,n)$ be the minimum and maximum number of operations ω performed on inputs from $P^{\emptyset}_{m,n}$. Then we have*

$$\begin{aligned} t^{\text{div}}_{\min}(m,n) &= \begin{cases} 1, & \text{if } m \neq n+1,\\ 2, & \text{if } m = n+1,\end{cases}\\ t^{-}_{\min}(m,n) &= n(m-n+1), \quad \text{if } m \geq n,\\ t^{\text{div}}_{\max}(m,n) &= \begin{cases} \lceil (2n+1)/3 \rceil, & \text{if } m > n+1,\\ \lceil 2n/3 \rceil, & \text{if } m = n+1,\\ \lceil (2n-2)/3 \rceil, & \text{if } m = n,\end{cases}\\ t^{-}_{\max}(m,n) &= \begin{cases} mn - \lceil (n^2+n)/6 \rceil - (-n \operatorname{rem} 3), & \text{if } m > n+1,\\ \lfloor (5n^2+3n-2)/6 \rfloor - (-n-1 \operatorname{rem} 3), & \text{if } m = n+1,\\ \lfloor (5n^2-7n)/6 \rfloor - 1 - (-n \operatorname{rem} 3), & \text{if } m = n \geq 5,\\ n, & \text{if } 2 \leq m = n \leq 4.\end{cases} \end{aligned}$$

Lemma 4.11. *For $2 \leq n \leq m$ and $\omega \in \{\text{div}, -\}$, $t^{\omega}_{\max}$ is monotonically increasing in both arguments:*

$$t^{\omega}_{\max}(m+1,n) \geq t^{\omega}_{\max}(m,n) \quad \text{if } m \geq n+1, \tag{12}$$
$$t^{\omega}_{\max}(m,n+1) \geq t^{\omega}_{\max}(m,n) \quad \text{if } m \geq n+2. \tag{13}$$

Proof. Let $(A_1, A_2) \in P^{\emptyset}_{m,n}$ with cost $t^{\omega}_{\max}(m,n)$, and write $A_1 = Q_1 A_2 + A_3$ with $\deg A_3 < \deg A_2$. To prove (12), let

$$B_1 = A_1 + x^{m-n}(x+1)A_2 = (Q_1 + x^{m-n}(x+1))A_2 + A_3.$$

Then $(B_1, A_2) \in P^{\emptyset}_{m+1,n}$ and the computation of Algorithm 4.2 after the first step is the same for (A_1, A_2) and (B_1, A_2). It follows that

$$t^{\omega}_{\max}(m+1,n) \geq t^{\omega}(B_1, A_2) \geq t^{\omega}(A_1, A_2) = t^{\omega}_{\max}(m,n).$$

As for (13), it is easy to check for $n = 2, 3$, and we now assume $n \geq 4$. Then $A_3 \neq 0$. We write $A_2 = Q_2\bar{A}_3 + A_4$ with $\deg A_4 < \deg \bar{A}_3$. We have to consider three cases. First assume that $n_3 = \deg \bar{A}_3 = n - 1$, and let

$$B_2 = x\,(x+1)\,\bar{A}_3 + \bar{A}_4, \quad B_1 = x^{m-n-1}B_2 + (x+1)\bar{A}_3.$$

If $\bar{A}_4 = 0$, we use $B_2 = (x^2 + x + 1)\bar{A}_3$. Then $(B_1, B_2) \in P^{\emptyset}_{m,n+1}$ and the algorithm proceeds through $(B_1, B_2) \to (B_2, \bar{A}_3) \to (\bar{A}_3, \bar{A}_4)$. For $\omega \in \{\text{div}, -\}$, recall the definition of d^{ω} in (2). We have

$$\begin{aligned} t^{\omega}_{\max}(m, n+1) &\geq t^{\omega}(B_1, B_2) \\ &= d^{\omega}(m, n+1) + d^{\omega}(n+1, n-1) + t^{\omega}(\bar{A}_3, \bar{A}_4) \\ &\geq d^{\omega}(m, n) + d^{\omega}(n, n-1) + t^{\omega}(\bar{A}_3, \bar{A}_4) \\ &= t^{\omega}(A_1, A_2) = t^{\omega}_{\max}(m, n), \end{aligned}$$

with strict inequality for $\omega = -$. So now we may assume that $n_3 \leq n - 2$. We first deal with the case that $m \geq n + 3$. Let

$$B_2 = x\,A_2 + (x+1)\,\bar{A}_3, \quad B_1 = x^{m-n-2}(x+1)B_2 + A_2.$$

Then $(B_1, B_2) \in P^{\emptyset}_{m,n+1}$, and $(B_1, B_2) \to (B_2, A_2) \to (A_2, \bar{A}_3)$. It follows that $t^{\omega}(B_1, B_2) > t^{\omega}(A_1, A_2)$.

We now come to the last case, where $n_3 \leq n - 2$ and $m = n + 2$. Take any $(C_2, \bar{C}_3) \in P^{\emptyset}_{n,n-2}$ with cost $t^{\omega}_{\max}(n, n-2)$, and write $C_2 = P_2\,\bar{C}_3 + C_4$, with $\deg C_4 < \deg \bar{C}_3$. Let

$$B_2 = x^2\,(x+1)\,\bar{C}_3 + \bar{C}_4, \quad B_1 = xB_2 + (x+1)\,\bar{C}_3.$$

If $\bar{C}_4 = 0$, we use $B_2 = (x^3 + x + 1)\bar{C}_3$. Then $(B_1, B_2) \in P^{\emptyset}_{n+2,n+1}$, and $(B_1, B_2) \to (B_2, \bar{C}_3) \to (\bar{C}_3, \bar{C}_4)$. We have $t^{\omega}_{\max}(n, n-2) \geq t^{\omega}_{\max}(n, n_3) \geq t^{\omega}(A_2, \bar{A}_3)$; this is clear if $n_3 = n - 2$, and follows from the previous cases if $n_3 \leq n - 3$ (provided that $n_3 \geq 2$, otherwise it is trivial). It follows that

$$\begin{aligned} t^{\omega}_{\max}(n+2, n+1) &\geq t^{\omega}(B_1, B_2) \\ &= d^{\omega}(n+2, n+1) + d^{\omega}(n+1, n-2) + t^{\omega}(\bar{C}_3, \bar{C}_4) \\ &= d^{\omega}(n+2, n) + d^{\omega}(n, n-2) + t^{\omega}(\bar{C}_3, \bar{C}_4) \\ &= d^{\omega}(n+2, n) + t^{\omega}_{\max}(n, n-2) \\ &\geq d^{\omega}(n+2, n) + t^{\omega}(A_2, \bar{A}_3) \\ &= t^{\omega}(A_1, A_2) = t^{\omega}_{\max}(n+2, n). \quad \square \end{aligned}$$

PROOF OF THEOREM 4.10. By Corollary 4.8, we have $t^{\text{div}}_{\min}(m, n) = 1$ for $m \neq n+1$, since there exists $(A_1, A_2) \in P^{\emptyset}_{m,n}$ such that A_2 divides A_1. For $m = n + 1$, we have $t^{\text{div}}_{\min} = 2$, since there exists $(A_1, A_2) \in P^{\emptyset}_{m,n}$ such that $A_1 = (x+1)\,A_2 + x$. The minimum number $t^{-}_{\min}(m, n)$ of operations $-$ performed is clearly $n\,(m - n + 1)$ for all $m \geq n \geq 2$.

We first prove that the stated values are upper bounds on $t^{\omega}_{\max}$. Intuitively, for $m \geq n + 2$ Algorithm 4.2 has the worst behavior if for every pass through step 2, the associated "lattice point" $(\deg u, \deg v)$ changes according to the following pattern:

$$(m, n) \to (n, n-1) \to (n-1, n-3) \to (n-3, n-4) \to (n-4, n-6) \to \cdots.$$

More formally, we have the following inequalities for $n \geq 5$ by Corollary 4.8 and (13):

$$\begin{aligned} t^{\text{div}}_{\max}(n+2,n) &\leq 1 + t^{\text{div}}_{\max}(n,n-1) \leq 2 + t^{\text{div}}_{\max}(n-1,n-3), \\ t^{-}_{\max}(n+2,n) &\leq 3\,n + t^{-}_{\max}(n,n-1) \leq 5\,n - 2 + t^{-}_{\max}(n-1,n-3). \end{aligned}$$

Note that $t^{\text{div}}_{\max}(4,2) \leq 2$, $t^{\text{div}}_{\max}(5,3) \leq 3$, $t^{\text{div}}_{\max}(6,4) \leq 3$, and $t^{-}_{\max}(4,2) \leq 6$, $t^{-}_{\max}(5,3) \leq 13$, $t^{-}_{\max}(6,4) \leq 18$. We obtain, for $n \geq 2$,

$$t^{\text{div}}_{\max}(n+2,n) \leq \lceil (2\,n+1)/3 \rceil, \quad t^{-}_{\max}(n+2,n) \leq \lfloor (5\,n^2 + 11\,n)/6 \rfloor - (-n \operatorname{rem} 3). \quad (14)$$

By Corollary 4.8 and (13), the following holds for $m \geq n+2$:

$$\begin{aligned} &t^{\text{div}}_{\max}(m,n) \leq 1 + t^{\text{div}}_{\max}(n,n-1), && t^{-}_{\max}(m,n) \leq n\,(m-n+1) + t^{-}_{\max}(n,n-1), \\ &t^{\text{div}}_{\max}(n+1,n) \leq 1 + t^{\text{div}}_{\max}(n,n-2), && t^{-}_{\max}(n+1,n) \leq 2\,n + t^{-}_{\max}(n,n-2), \\ &t^{\text{div}}_{\max}(n,n) \leq 1 + t^{\text{div}}_{\max}(n,n-3), && t^{-}_{\max}(n,n) \leq n + t^{-}_{\max}(n,n-3). \end{aligned}$$

Note that $t^{-}_{\max}(n,n) = n$ for $2 \leq n \leq 4$, $t^{\omega}_{\max}(m,1) = 0$ since $P^{\emptyset}_1 = \emptyset$, and $t^{\text{div}}_{\max}(m,0) = 1$, $t^{-}_{\max}(m,0) = 0$. The claimed upper bounds on $t^{\omega}_{\max}$ follow from a straightforward calculation using (14).

To prove that these are also lower bounds, we exhibit the following $(A_1, A_2) \in P^{\emptyset}_{m,n}$, for which the bounds are attained. This property is checked by verifying that one application of steps 2 and 3 of Algorithm 4.2 transforms any such pair into another pair on the list, and that all of the given inequalities are actually equalities in this case.

- $A_1 = x\,(x+1)^{m-n-1}\,t_n + s_{n-1}$ and $A_2 = t_n$ for $m \geq n+2 \geq 4$,
- $A_1 = t_{n+1}$ and $A_2 = s_n$ for $m = n+1 \geq 3$,
- $A_1 = x\,(x+1)^2\,t_{n-3} + x\,(x+1)\,t_{n-3} + s_{n-4}$ and $A_2 = x\,(x+1)^2\,t_{n-3} + s_{n-4}$ for $m = n \geq 5$.

Here $s_n, t_n \in P^{\emptyset}_n$ for $n \geq 4$ are defined by $s_n = x\,(x+1)\,t_{n-2} + s_{n-3}$ and $t_n = (x+1)\,s_{n-1} + x\,t_{n-3}$ with the initial values $s_1 = t_1 = 1$, $s_2 = t_2 = x^2 + x + 1$, and $s_3 = t_3 = x^3 + x + 1$. □

5. The Subtractive Indeterminate Shift Algorithm

Analogous to the binary shift Euclidean algorithm for computing integer gcd's, the indeterminate shift algorithm computes polynomial gcd's using only polynomial subtractions and divisions by x. The algorithm works for any finite field, but we formulate and analyze it only for $\mathbb{Z}_2$.

ALGORITHM 5.1. Given the input $(A_1, A_2) \in P_{m,n}$ $(m \geq n \geq 1)$, this algorithm computes $\gcd(A_1, A_2)$.

1. [Initialization] For $x^{k_1} \parallel A_1$ and $x^{k_2} \parallel A_2$, let $u \leftarrow A_1/x^{k_1}$, $v \leftarrow A_2/x^{k_2}$, and $k \leftarrow \min(k_1, k_2)$. If $\deg v > \deg u$, then swap u and v. Stop and return $\gcd(A_1, A_2) = x^k$ if $\deg v = 0$.

2. [Subtraction and shifting] Let $r = u - v$ and compute $\tilde{r}$, as defined in Section 3. If $\tilde{r} = 0$ or $\tilde{r} = 1$, then stop and return $\gcd(A_1, A_2) = x^k v$ or $\gcd(A_1, A_2) = x^k$, respectively.
3. [Resetting] If $\deg \tilde{r} \geq \deg v$, then $u \leftarrow \tilde{r}$; otherwise, $u \leftarrow v$, $v \leftarrow \tilde{r}$. Go to step 2.

Theorem 5.2. *Let $1 \leq n \leq m$. The average number $t^{\text{sub}}_{\text{ave}}(m, n)$ of polynomial subtractions used in Algorithm* 5.1 *on inputs from $\tilde{P}_{m,n}$ is $\frac{1}{2} m + \frac{1}{3} n + \nu(m, n)$, where $\nu(m, n) < 1$ is given by*

$$\nu(m,n) = \begin{cases} -\frac{2}{9} 4^{-n} + 2^{1-n} - \frac{7}{9}, & \textit{if } m > n, \\ -\frac{2}{9} 4^{1-n} - (n-4)\, 2^{-n} - \frac{10}{9}, & \textit{if } m = n. \end{cases}$$

We assume that a subtraction $u - v$ needs exactly $1 + \deg v$ operations $-$ in $\mathbb{Z}_2$, if $\deg u \geq \deg v$.

Theorem 5.3. *Let $1 \leq n \leq m$. The average number $t^{-}_{\text{ave}}(m, n)$ of operations $-$ used in Algorithm* 5.1 *on inputs from $\tilde{P}_{m,n}$ is $\frac{1}{2} m n - \frac{1}{12} n^2 + \frac{1}{2} m - \frac{1}{36} n + \pi(m, n)$, where $\pi(m, n) < 2$ is given by*

$$\pi(m,n) = \begin{cases} \frac{31}{27} - \frac{1}{27} 4^{1-n}, & \textit{if } m > n, \\ \frac{16}{27} - \frac{1}{3} n - \frac{1}{27} 4^{2-n} + 2^{1-n}, & \textit{if } m = n. \end{cases}$$

Average case analysis of algorithm 5.1

Assuming that (A_1, A_2) is uniformly distributed in $\tilde{P}_{m,n}$, we can analyze this algorithm by solving the following recurrence relation:

$$S_{mn} = a + \sum_{0 \leq k < m} \mathbf{Pr}[\deg \tilde{r} = k] S_{nk}, \quad S_{mn} = S_{nm}, \quad m \geq n \geq 1. \tag{15}$$

Corollary 5.5 below implies that for $a = 1$ and $S_{n0} = 0$, S_{mn} is the average number $t^{\text{sub}}_{\text{ave}}(m, n)$ of polynomial subtractions used in Algorithm 5.1 for computing $\gcd(A_1, A_2)$. S_{mn} is the average number $t^{-}_{\text{ave}}(m, n)$ of operations $-$ used in Algorithm 5.1 for $a = n + 1$ and $S_{n0} = 0$.

Lemma 5.4. *For $1 \leq n \leq m$, let (u, v) be uniformly distributed in $\tilde{P}_{m,n}$ and $r = u - v$. Then r is uniformly distributed in $\{x\} \times P_{m-1}$ if $m > n$; r is uniformly distributed in $\{x\} \times R_{n-1}$ if $m = n$.*

Corollary 5.5. *Let m, n, u, v, r be as above, and let $\tilde{r} = r / x^i$ where $x^i \parallel r$ if $r \neq 0$, and $\tilde{0} = 0$. Then, for $1 \leq k \leq m - 1$,*

$$\mathbf{Pr}[\tilde{r} = 0] = \begin{cases} 0, & \textit{if } m > n, \\ 2^{1-n}, & \textit{if } m = n, \end{cases}$$

$$\mathbf{Pr}[\deg \tilde{r} = 0] = \begin{cases} 2^{1-m}, & \textit{if } m > n, \\ (n-1)\, 2^{1-n}, & \textit{if } m = n, \end{cases}$$

$$\mathbf{Pr}[\deg \tilde{r} = k] = \begin{cases} 2^{k-m}, & \textit{if } m > n, \\ (n-k-1)\, 2^{k-n}, & \textit{if } m = n. \end{cases}$$

PROOF. We distinguish two cases.
Case 1: $m > n$. Since r is uniformly distributed in $\{x\} \times P_{m-1}$, $\mathbf{Pr}[\tilde{r} = 0] = 0$, and

$$\mathbf{Pr}[\deg \tilde{r} = 0] = \mathbf{Pr}[r = x^m] = 2^{1-m}, \ \mathbf{Pr}[\deg \tilde{r} = k] = \sum_{f \in \tilde{P}_k} \mathbf{Pr}[r = x^{m-k} f] = 2^{k-m}.$$

Case 2: $m = n$. Since r is uniformly distributed in $\{x\} \times R_{n-1}$, $\mathbf{Pr}[\tilde{r} = 0] = 2^{1-n}$,

$$\mathbf{Pr}[\deg \tilde{r} = 0] = \sum_{0 \le i \le n-2} \mathbf{Pr}[r = x^{i+1}] = (n-1)\, 2^{1-n},$$

$$\mathbf{Pr}[\deg \tilde{r} = k] = \sum_{f \in \tilde{P}_k} \sum_{0 \le i \le n-k-2} \mathbf{Pr}[r = x^{i+1} f] = (n-k-1)\, 2^{k-n}. \qquad \square$$

PROOF OF THEOREM 5.2. Let $a = 1$ and $S_{n0} = 0$ in (15). Applying Corollary 5.5, we obtain the following set of double recurrence relations:

$$S_{mn} = 1 + \sum_{1 \le k < m} 2^{k-m} S_{nk}, \quad \text{if } m > n \ge 1,$$

$$S_{mn} = 1 + \sum_{1 \le k < n} (n-k-1)\, 2^{k-n} S_{nk}, \quad \text{if } m = n \ge 1.$$

Note that $S_{mn} - \frac{1}{2} S_{m-1,n} = \frac{1}{2} + \frac{1}{2} S_{n,m-1}$ for $m > n+1$; that is, $S_{mn} = S_{m-1,n} + \frac{1}{2}$. By induction on $m - n$, we have

$$S_{mn} = S_{n+1,n} + \tfrac{1}{2}(m - n - 1), \quad \text{if } m \ge n+1. \tag{16}$$

Let $S_n = S_{n+1,n}$ and $S'_n = S_{nn}$. This substitution into the above recurrence relations gives

$$S_n = 1 + \frac{1}{2^{n+1}} \sum_{1 \le k < n} 2^k \left[S_k + \frac{1}{2}(n-k-1)\right] + \frac{1}{2} S'_n, \quad \text{if } n \ge 1, \tag{17}$$

$$S'_n = 1 + \frac{1}{2^n} \sum_{1 \le k < n} (n-k-1)\, 2^k \left[S_k + \frac{1}{2}(n-k-1)\right], \quad \text{if } n \ge 1. \tag{18}$$

As in the proof of Theorems 3.2 and 3.4, we can use a similar transformation to obtain the following linear recurrence relations from (17) and (18):

$$S_n - \tfrac{5}{4} S_{n-1} + \tfrac{1}{4} S_{n-2} = \tfrac{5}{8} - 2^{-n}, \quad \text{if } n \ge 3,$$

$$S'_n - S'_{n-1} + \tfrac{1}{4} S'_{n-2} = \tfrac{1}{4} S_{n-2} + \tfrac{5}{8} - 2^{-(n-1)}, \quad \text{if } n \ge 3.$$

Solving these linear recurrence relations with the initial values $S_1 = \frac{3}{2}$, $S_2 = \frac{15}{8}$ and $S'_1 = S'_2 = 1$ from (17) and (18), we obtain

$$S_n = \tfrac{5}{6} n + 2^{1-n} - \tfrac{2}{9} 4^{-n} - \tfrac{5}{18}, \quad \text{if } n \ge 1,$$
$$S'_n = \tfrac{5}{6} n - (n-4)\, 2^{-n} - \tfrac{2}{9} 4^{1-n} - \tfrac{10}{9}, \quad \text{if } n \ge 1.$$

The claim then follows from (16). $\square$
The proof of Theorem 5.3 is similar and can be found in Ma (1987, pp. 75-80).

Best and worst case of Algorithm 5.1

THEOREM 5.6. *For $1 \le n \le m$ and $\omega \in \{\mathrm{sub}, -\}$, let $t^{\omega}_{\min}(m,n)$ and $t^{\omega}_{\max}(m,n)$ be the minimum and maximum number of operations ω used in Algorithm 5.1 on inputs from $\tilde{P}_{m,n}$. Then, except that $t^{\mathrm{sub}}_{\max}(2,2)=1$ and $t^{-}_{\max}(2,2)=3$, we have*

$$t^{\mathrm{sub}}_{\min}(m,n)=1, \qquad t^{\mathrm{sub}}_{\max}(m,n)=m+\lceil n/2\rceil-1,$$

$$t^{-}_{\min}(m,n)=n+1, \quad t^{-}_{\max}(m,n)=m\,n-\lfloor n^2/4\rfloor+m-1+n \operatorname{rem} 2.$$

PROOF. Using Lemma 5.4, we have $t^{\mathrm{sub}}_{\min}(m,n)=1$ and $t^{-}_{\min}(m,n)=n+1$ for all $1\le n\le m$, since there exists $(A_1,A_2)\in\tilde{P}_{m,n}$ such that $A_1-A_2=x^m$ for $m>n$, and $A_1-A_2=0$ for $m=n$.

The estimate of the maximum cost follows the lines of the proof of Theorem 4.10; we omit some details. It is easy to see that $t^{\mathrm{sub}}_{\max}(2,2)=1$ and $t^{-}_{\max}(2,2)=3$. For the other cases, we first prove the claimed upper bound on $t^{\omega}_{\max}$. Intuitively, the algorithm has the worst behavior if the "lattice point" $(\deg u, \deg v)$ associated at each pass through step 2 changes in the following pattern:

$$\begin{aligned}
&(m,1)\to(m-1,1)\cdots\to(1,1), \text{ for } m\ge n=1,\\
&(m,2)\to(m-1,2)\cdots\to(3,2)\to(2,1)\to(1,1), \text{ for } m>n=2,\\
&(3,3)\to(3,1)\to(2,1)\to(1,1), \text{ for } m=n=3,\\
&(4,4)\to(4,2)\to(3,2)\to(2,1)\to(1,1), \text{ for } m=n=4,\\
&(n,n)\to(n,n-2)\to(n-1,n-2)\to(n-2,n-2)\to\cdots, \text{ for } m=n\ge 5,\\
&(m,n)\to(m-1,n)\to\cdots\to(n,n), \text{ for } m>n\ge 3.
\end{aligned}$$

Formally, one proves monotonicity of $t^{\omega}_{\max}$ in the sense that

$$\begin{aligned}
t^{\omega}_{\max}(m+1,n) &\ge t^{\omega}_{\max}(m,n) \quad \text{if } m\ge n\ge 1,\\
t^{\omega}_{\max}(m,n+1) &\ge t^{\omega}_{\max}(m,n) \quad \text{if } m\ge n+1\ge 2
\end{aligned}$$

except that $t^{\omega}_{\max}(2,2)<t^{\omega}_{\max}(2,1)$, and the following inequalities for $n\ge 5$:

$$t^{\mathrm{sub}}_{\max}(n,n)\le 3+t^{\mathrm{sub}}_{\max}(n-2,n-2), \quad t^{-}_{\max}(n,n)\le 3n-1+t^{-}_{\max}(n-2,n-2).$$

Note that $t^{\mathrm{sub}}_{\max}(3,3)\le 4$, $t^{\mathrm{sub}}_{\max}(4,4)\le 5$, and $t^{-}_{\max}(3,3)\le 10$, $t^{-}_{\max}(4,4)\le 15$. We obtain the following for $n\ge 3$:

$$t^{\mathrm{sub}}_{\max}(n,n)\le n+\lceil n/2\rceil-1, \quad t^{-}_{\max}(n,n)\le\lceil 3n^2/4\rceil+n-1+n \operatorname{rem} 2.$$

Since

$$\begin{aligned}
&t^{\mathrm{sub}}_{\max}(m,1)\le m \text{ and } t^{-}_{\max}(m,1)\le 2m, && \text{for } m\ge 1,\\
&t^{\mathrm{sub}}_{\max}(m,2)\le m \text{ and } t^{-}_{\max}(m,2)\le 3(m-2)+4, && \text{for } m\ge 3,\\
&t^{\mathrm{sub}}_{\max}(m,n)\le m-n+t^{\mathrm{sub}}_{\max}(n,n), && \text{for } m\ge n\ge 3,\\
&t^{-}_{\max}(m,n)\le (m-n)(n+1)+t^{-}_{\max}(n,n), && \text{for } m\ge n\ge 3,
\end{aligned}$$

the claimed upper bounds follow immediately.

For the corresponding lower bounds, we construct explicit examples. Let $h_n, g_n\in\tilde{P}_n$ for $n\ge 1$ be defined as $h_1=g_1=x+1$, $h_2=x^2+x+1$, $g_2=x^2+1$, $h_3=x^3+x^2+x+1$, $g_3=x^3+1$, $h_4=x^4+x^3+x^2+1$, $g_4=x^4+x^2+x+1$, and $g_n=x^2h_{n-2}+(x+1)g_{n-2}$, $h_n=g_n+x\,g_{n-2}$ for $n\ge 5$. Then the stated bounds are attained for the following input $(A_1,A_2)\in\tilde{P}_{m,n}$ for $m\ge n\ge 1$:

- $A_1 = x^{m-3}\, g_3 + g_2 \sum_{0\le i<m-3} x^i$ and $A_2 = g_2$, for $m > n = 2$,
- $A_1 = x^{m-n}\, h_n + g_n \sum_{0\le i<m-n} x^i$ and $A_2 = g_n$, otherwise. □

6. The Subtractive Linear Factor Shift Algorithm

The subtractive indeterminate shift algorithm in Section 5 can be easily extended to do linear factor shifting, namely, to remove all linear factors from the resulting polynomial after each subtraction. The subtractive linear factor shift algorithm works for any finite field, but we formulate and analyze it only for $\mathbf{Z}_2$.

ALGORITHM 6.1. Given the input $(A_1, A_2) \in P_{m,n}$ $(m \ge n \ge 2)$, this algorithm outputs $\gcd(A_1, A_2)$.

1. [Initialization] For $x^{k_1} \parallel A_1$, $x^{k_2} \parallel A_2$, $(x+1)^{l_1} \parallel A_1$, and $(x+1)^{l_2} \parallel A_2$, let $u \leftarrow A_1/[x^{k_1}(x+1)^{l_1}]$, $v \leftarrow A_2/[x^{k_2}(x+1)^{l_2}]$, $k \leftarrow \min(k_1,k_2)$, and $l \leftarrow \min(l_1,l_2)$. If $\deg v > \deg u$, then swap u and v. Stop and return $\gcd(A_1,A_2) = x^k(x+1)^l$ if $\deg v = 0$.
2. [Subtraction and shifting] Let $r = u - v$ and compute $\bar r$, as defined in Section 4. If $\bar r = 0$ or $\bar r = 1$, then stop and return $\gcd(A_1,A_2) = x^k(x+1)^l v$ or $\gcd(A_1,A_2) = x^k(x+1)^l$, respectively.
3. [Resetting] If $\deg \bar r \ge \deg v$, then $u \leftarrow \bar r$; otherwise, $u \leftarrow v$, $v \leftarrow \bar r$. Go to step 2.

THEOREM 6.2. *Let $2 \le n \le m$. The average number $t^{\mathrm{sub}}_{\mathrm{ave}}(m,n)$ of polynomial subtractions used in Algorithm 6.1 on inputs from $P^{\emptyset}_{m,n}$ is $\frac14 m + \frac29 n + \rho(m,n)$, where $\rho(m,n) < 1$ is given by*

$$\rho(m,n) = \begin{cases} \frac{1}{27}(15n-47)\,4^{1-n} + 3\cdot 2^{-n} - \frac{85}{108}, & \text{if } m > n+1,\\ \frac{1}{27}(15n-17)\,4^{2-n} + (n^2-5n+6)\,2^{-n-1} - \frac{73}{108}, & \text{if } m = n+1,\\ -\frac{1}{54}(15n-2)\,4^{3-n} + (-3n^2+13n+12)\,2^{-n-1} - \frac{121}{108}, & \text{if } m = n. \end{cases}$$

REMARK 6.3. If $m-n$ is constant and n is large, then Algorithm 6.1 needs an average of $\frac{17}{36}n + O(1)$ polynomial subtractions to compute $\gcd(A_1,A_2)$. This shows that the subtractive linear factor shift algorithm uses on average less polynomial subtractions than the classical Euclidean algorithm uses polynomial divisions.

THEOREM 6.4. *Let $2 \le n \le m$. The average number $t^{-}_{\mathrm{ave}}(m,n)$ of operations $-$ used in Algorithm 6.1 on inputs from $P^{\emptyset}_{m,n}$ is $\frac14 mn - \frac{1}{72}n^2 + \frac14 m + \frac{55}{216}n + \sigma(m,n)$, where $\sigma(m,n) < 1$ is given by*

$$\sigma(m,n) = \begin{cases} \frac{5}{54} - \frac19 n + \frac{5}{27}(5n-17)\,4^{1-n} + 3\cdot 2^{-n}, & \text{if } m > n+1,\\ \frac{13}{54} + \frac{5}{27}(5n-7)\,4^{2-n} + (n^2-9n+10)\,2^{-n-1}, & \text{if } m = n+1,\\ -\frac{19}{54} - \frac49 n - \frac{5}{54}(5n-2)\,4^{3-n} + (-3n^2+25n+4)\,2^{-n-1}, & \text{if } m = n. \end{cases}$$

Average case analysis of algorithm 6.1

Lemma 6.5. *For $2 \le n \le m$, let (u, v) be uniformly distributed in $P^{\emptyset}_{m,n}$ and $r = u - v$. Then r is uniformly distributed in $\{x\,(x+1)\} \times P_{m-2}$ when $m > n$; r is uniformly distributed in $\{x\,(x+1)\} \times R_{n-2}$ when $m = n$.*

Corollary 6.6. *Let m, n, u, v, r be as above, and let $\bar{r} = r/[x^i\,(x+1)^j]$, where $x^i \parallel r$ and $(x+1)^j \parallel r$ if $r \ne 0$, and $\bar{0} = 0$. Then $\mathbf{Pr}[\deg \bar{r} = 1] = 0$, and for $2 \le k \le m-1$,*

$$\begin{aligned}
\mathbf{Pr}[\bar{r} = 0] &= \begin{cases} 0, & \text{if } m > n, \\ 2^{2-n}, & \text{if } m = n, \end{cases} \\
\mathbf{Pr}[\deg \bar{r} = 0] &= \begin{cases} (m-1)\,2^{2-m}, & \text{if } m > n, \\ (n-2)\,(n-1)\,2^{1-n}, & \text{if } m = n, \end{cases} \\
\mathbf{Pr}[\deg \bar{r} = k] &= \begin{cases} (m-k-1)\,2^{k-m}, & \text{if } m > n, \\ (n-k-2)\,(n-k-1)\,2^{k-n-1}, & \text{if } m = n. \end{cases}
\end{aligned}$$

Proof. By definition, $\mathbf{Pr}[\deg \bar{r} = 1] = 0$. We distinguish two cases.
Case 1: $m > n$. Since r is uniformly distributed in $\{x\,(x+1)\} \times P_{m-2}$, $\mathbf{Pr}[\bar{r} = 0] = 0$,

$$\begin{aligned}
\mathbf{Pr}[\deg \bar{r} = 0] &= \sum_{k_1+k_2=m-2} \mathbf{Pr}[r = x^{k_1+1}(x+1)^{k_2+1}] = (m-1)\,2^{2-m}, \\
\mathbf{Pr}[\deg \bar{r} = k] &= \sum_{f \in P^{\emptyset}_k} \sum_{k_1+k_2=m-2-k} \mathbf{Pr}[r = x^{k_1+1}\,(x+1)^{k_2+1}\,f] = (m-k-1)\,2^{k-m}.
\end{aligned}$$

Case 2: $m = n$. Since r is uniformly distributed in $\{x\,(x+1)\} \times R_{n-2} = R^{01}_n$, the same probabilities have been derived in Corollary 4.9. □

Proof of Theorem 6.2. Let $2 \le n \le m$ and (A_1, A_2) be uniformly distributed in $P^{\emptyset}_{m,n}$. Corollary 6.6 reveals that the average number $t^{\text{sub}}_{\text{ave}}(m, n)$ of polynomial subtractions used in Algorithm 6.1 is the solution S_{mn} to the following recurrence relation (with $S_{n0} = 0$ and $a = 1$):

$$S_{mn} = a + \sum_{0 \le k < m} \mathbf{Pr}[\deg \bar{r} = k]\, S_{nk}, \quad S_{mn} = S_{nm}, \quad m \ge n \ge 2.$$

Applying Corollary 6.6, we obtain the following

$$S_{mn} = 1 + \sum_{2 \le k \le m-2} \frac{(m-k-1)}{2^{m-k}}\, S_{nk}, \quad \text{if } m > n \ge 2, \tag{19}$$

$$S_{mn} = 1 + \sum_{2 \le k \le n-2} \frac{(n-k-2)\,(n-k-1)}{2^{n+1-k}}\, S_{nk}, \quad \text{if } m = n \ge 2. \tag{20}$$

Note that $S_{mn} - \frac{1}{2}\, S_{m-1,n} = \frac{1}{2} + \frac{1}{2^m} \sum_{k=2}^{m-2} S_{nk}\, 2^k$ for $m > n+1$. Therefore,

$$S_{mn} - S_{m-1,n} + \frac{1}{4}\, S_{m-2,n} = \frac{1}{4} + \frac{1}{4}\, S_{n,m-2}$$

for $m > n+2$; that is, $S_{mn} = S_{m-1,n} + \frac{1}{4}$. An easy induction on $m - n$ yields

$$S_{mn} = S_{n+2,n} + \tfrac{1}{4}\,(m-n-2), \quad \text{if } m \ge n+2. \tag{21}$$

Let $S_n = S_{n+2,n}$, $S'_n = S_{n+1,n}$ and $S''_n = S_{nn}$. This substitution into (19) and (20) gives

$$S_n = 1 + \frac{1}{2^{n+2}} \sum_{2 \le k \le n-2} (n-k+1)\, 2^k \left[S_k + \frac{1}{4}(n-k-2)\right] + \frac{1}{4} S'_{n-1} + \frac{1}{4} S''_n, \quad (22)$$

$$S'_n = 1 + \frac{1}{2^{n+1}} \sum_{2 \le k \le n-2} (n-k)\, 2^k \left[S_k + \frac{1}{4}(n-k-2)\right] + \frac{1}{4} S'_{n-1}, \quad (23)$$

$$S''_n = 1 + \frac{1}{2^{n+1}} \sum_{2 \le k \le n-2} (n-k-2)(n-k-1)\, 2^k \left[S_k + \frac{1}{4}(n-k-2)\right]. \quad (24)$$

Applying a similar manœuvre as in the proof of Theorems 3.2 and 3.4, we can transform the above recurrence relations into the

$$S_n - \tfrac{7}{4} S_{n-1} + \tfrac{15}{16} S_{n-2} - \tfrac{13}{63} S_{n-3} + \tfrac{1}{64} S_{n-4} = \tfrac{51}{256} - 3 \cdot 2^{-n-3}, \qquad \text{if } n \ge 6,$$

$$S'_n - \tfrac{5}{4} S'_{n-1} + \tfrac{1}{2} S'_{n-2} - \tfrac{1}{16} S'_{n-3} = \tfrac{1}{4} S_{n-2} - \tfrac{1}{16} S_{n-3} + \tfrac{21}{64} - 2^{-n}, \quad \text{if } n \ge 5,$$

$$S''_n - \tfrac{3}{2} S''_{n-1} + \tfrac{3}{4} S''_{n-2} - \tfrac{1}{8} S''_{n-3} = \tfrac{1}{8} S_{n-3} + \tfrac{1}{4} - 3 \cdot 2^{-n}, \qquad \text{if } n \ge 5.$$

Solving these linear recurrence relations with the initial values $S_2 = \frac{5}{4}$, $S_3 = \frac{3}{2}$, $S_4 = \frac{115}{64}$, $S_5 = \frac{139}{64}$, $S'_2 = 1$, $S'_3 = \frac{5}{4}$, $S'_4 = \frac{13}{8}$ and $S''_2 = S''_3 = S''_4 = 1$ from (22), (23) and (24), we obtain

$$\begin{aligned} S_n &= \tfrac{17}{36} n + \tfrac{1}{27}(15n - 47)\, 4^{1-n} + 3 \cdot 2^{-n} - \tfrac{31}{108}, \quad \text{if } n \ge 2, \\ S'_n &= \tfrac{17}{36} n + \tfrac{1}{27}(15n - 17)\, 4^{2-n} + (n^2 - 5n + 6)\, 2^{-n-1} - \tfrac{23}{54}, \quad \text{if } n \ge 2, \\ S''_n &= \tfrac{17}{36} n - \tfrac{1}{54}(15n - 2)\, 4^{3-n} + (-3n^2 + 13n + 12)\, 2^{-n-1} - \tfrac{121}{108}, \quad \text{if } n \ge 2. \end{aligned}$$

The claim follows from (21). □

The proof of Theorem 6.4 is similar and can be found in Ma (1987, pp. 82-90).

Best and worst case of Algorithm 6.1

Theorem 6.7. *For $2 \le n \le m$ and $\omega \in \{\mathrm{sub}, -\}$, let $t^{\omega}_{\min}(m,n)$ and $t^{\omega}_{\max}(m,n)$ be the minimum and maximum number of operations ω used in Algorithm 6.1 on inputs from $P^{\emptyset}_{m,n}$. Let $d_{m,n} = (m-n) \operatorname{rem} 2$. Then we have*

$$\begin{aligned} t^{\mathrm{sub}}_{\min}(m,n) &= 1, \\ t^{-}_{\min}(m,n) &= n+1, \\ t^{\mathrm{sub}}_{\max}(m,n) &= \begin{cases} (m-n)/2 + 1, & \text{if } m = n = 4, \text{ or } d_{m,n} = 0 \text{ and } 2 \le n \le 3, \\ \lceil (m+n)/2 \rceil - 2, & \text{otherwise,} \end{cases} \\ t^{-}_{\max}(m,n) &= \begin{cases} (m-n+2)(n+1)/2, & \text{if } m = n = 4, \text{ or } d_{m,n} = 0 \text{ and } 2 \le n \le 3, \\ (m + d_{m,n})(n+1)/2 - 3, & \text{otherwise.} \end{cases} \end{aligned}$$

Proof. By Lemma 6.5, we have $t^{\mathrm{sub}}_{\min}(m,n) = 1$ and $t^{-}_{\min}(m,n) = n+1$ for all $2 \le n \le m$, since there exists $(A_1, A_2) \in P^{\emptyset}_{m,n}$ such that $A_1 - A_2 = x^{m-1}(x+1)$ for $m > n$, and $A_1 - A_2 = 0$ for $m = n$.

As usual, we first prove an upper bound on $t^{\omega}_{\max}$, following the lines of the proof of Theorem 4.10. Intuitively, the algorithm has the worst behavior if the "lattice point" $(\deg u, \deg v)$ associated at each pass through step 2 changes in the following pattern:

$(n+1,n) \to (n,n-1) \to (n-1,n-2) \to \cdots \to (3,2)$, if $m = n+1 \geq 3$,

$(n,n) \to (n,n-3) \to (n-2,n-3) \to \cdots \to (3,2)$, if $m = n \geq 5$,

$(m,4) \to (m-2,4) \to (m-4,4) \to \cdots \to (5,4)$, if $m > n = 4$ and $d_{m,4} = 1$,

$(m,4) \to (m-2,4) \to \cdots \to (6,4) \to (4,3) \to (3,2)$, if $m > n = 4$ and $d_{m,4} = 0$,

$(m,n) \to (m-2,n) \to (m-4,n) \to \cdots \to (n+d_{m,n},n)$, if $m \geq n+2$ and $n \neq 4$.

Formally, one proves monotonicity of $t^\omega_{\max}$ in the sense that

$$\begin{aligned} t^\omega_{\max}(m+1,n) &\geq t^\omega_{\max}(m,n) \quad \text{if } m \geq n \geq 2, \\ t^\omega_{\max}(m,n+1) &\geq t^\omega_{\max}(m,n) \quad \text{if } m \geq n+1 \geq 3, \end{aligned}$$

except that $t^\omega_{\max}(4,4) < t^\omega_{\max}(4,3)$. Defining $d^{\text{sub}}(n) = 1$ and $d^-(n) = n+1$, we have the following inequalities:

$$\begin{aligned} t^\omega_{\max}(n+1,n) &\leq d^\omega(n) + t^\omega_{\max}(n,n-1), \text{ if } n+1 \geq 4, & (25)\\ t^{\text{sub}}_{\max}(n,n) &\leq 2 + t^{\text{sub}}_{\max}(n-2,n-3), \text{ if } n \geq 5, & (26)\\ t^-_{\max}(n,n) &\leq 2n-1+t^-_{\max}(n-2,n-3), \text{ if } n \geq 5, & (27)\\ t^\omega_{\max}(m,n) &\leq \lfloor (m-n)/2 \rfloor d^\omega(n) + t^\omega_{\max}(n+d_{m,n},n), \text{ if } m \geq n+2, n \neq 4, & (28)\\ t^\omega_{\max}(m,n) &\leq \lfloor (m-4)/2 \rfloor d^\omega(4) + t^\omega_{\max}(5,4), \text{ if } m > n = 4 \text{ and } d_{m,4} = 1, & (29)\\ t^\omega_{\max}(m,n) &\leq (m-6)d^\omega(4)/2 + t^\omega_{\max}(6,4), \text{ if } m > n = 4 \text{ and } d_{m,4} = 0, & (30) \end{aligned}$$

Since $t^{\text{sub}}_{\max}(3,2) \leq 1$ and $t^-_{\max}(3,2) \leq 3$, it follows from (25) to (27) that $t^{\text{sub}}_{\max}(m,n) \leq m-2$ and $t^-_{\max}(m,n) \leq (m+3)(m-2)/2$ for $2 \leq n \leq m \leq n+1$, with the exception that $t^\omega_{\max}(n,n) = d^\omega(n)$ for $2 \leq n \leq 4$. Plugging this into (28) and (29) and noting that $t^{\text{sub}}_{\max}(6,4) \leq 3$ and $t^-_{\max}(6,4) \leq 12$ in (30), we obtain the upper bounds as claimed.

For the lower bounds, we again construct explicit examples. Let $q = x^2 + x$, and $u_n \in P^\emptyset_n$ for $n \geq 4$ be defined by $u_n = u_{n-1} + q\,u_{n-2}$ with $u_2 = x^2+x+1$ and $u_3 = x^3+x^2+1$. Furthermore, let $v_n = q\,u_{n-2} + q\,u_{n-3} + u_{n-3}$ and $w_n = q\,u_{n-2} + u_{n-3}$ for $n \geq 5$, $v_2 = w_2 = u_2$, $v_3 = u_3$, $w_3 = x^3+x+1$, $v_4 = x^4+x^3+1$, and $w_4 = x^4+x+1$. Then, for $n \geq 2$, $u_n, v_n, w_n \in P^\emptyset_n$, and the stated bounds are attained for the following input $(A_1, A_2) \in P^\emptyset_{m,n}$:

- $A_1 = u_{n+1}$ and $A_2 = u_n$, for $m = n+1$;
- $A_1 = v_n$ and $A_2 = w_n$, for $m = n$;
- $A_1 = q^{(m-n-1)/2}\,u_{n+1} + u_n \sum_{0 \leq i \leq (m-n-3)/2} q^i$ and $A_2 = u_n$, for $m \geq n+2$ and $d_{m,n} = 1$;
- $A_1 = q^{(m-n)/2}\,v_n + w_n \sum_{0 \leq i \leq (m-n-2)/2} q^i$ and $A_2 = w_n$, for $m \geq n+2$, $n \neq 4$ and $d_{m,n} = 0$,
- $A_1 = x\,q^{(m-4)/2}\,u_3 + u_4 \sum_{0 \leq i \leq (m-6)/2} q^i$ and $A_2 = u_4$, for $m \geq n+2$, $n = 4$ and $d_{m,n} = 0$. □

7. Conclusion

We have given exact average-case analyses for several variants of Euclid's algorithm for polynomials over a finite field. It would be nice to extend the results of Sections 4 to 6 from $\mathbb{Z}_2$ to an arbitrary finite field. More importantly, it would be desirable to have analogous results for various algorithms for factoring polynomials, say in $\mathbb{Z}_2[x]$, to determine the optimal sequence of the usual operations of extracting square roots (resp. pth roots), dividing out $\gcd(A, A')$, and the various factoring algorithms for squarefree polynomials (e.g., Berlekamp 1970; Rabin 1980; Ben-Or 1981; Cantor & Zassenhaus 1981).

References

Aho, A. V., Hopcroft, J. E., Ullman, J. D. (1975). *The design and analysis of computer algorithms.* Reading, MA: Addison-Wesley.

Ben-Or, M. (1981). Probabilistic algorithms in finite fields. *Proc. 22nd IEEE Symp. on the Foundations of Computer Science*, Nashville, TN, pp. 394-398.

Berlekamp, E. R. (1970). Factoring polynomials over large finite fields. *Math. Comp.* **24**, 713-735.

Borodin, A., von zur Gathen, J., Hopcroft, J. E. (1982). Fast parallel matrix and GCD computations. *Information and Control* **52**, 241-256.

Brent, R. P. (1976). Analysis of the binary Euclidean algorithm. In: *Algorithms and Complexity*, ed. J. F. Traub, pp. 321-355. New York: Academic Press.

Brown, W. S. (1971). On Euclid's algorithm and the computation of polynomial greatest common divisors. *J. Assoc. Comput. Mach.* **18**, 478-504.

Brown, W. S., Traub, J. F. (1971). On Euclid's algorithm and the theory of subresultants. *J. Assoc. Comput. Mach.* **18**, 505-514.

Cantor, D. G., Zassenhaus, H. (1981). On algorithms for factoring polynomials over finite fields. *Math. Comp.* **36**, 587-592.

Collins, G. E. (1967). Subresultants and reduced polynomial remainder sequences. *J. Assoc. Comput. Mach.* **14**, 128-142.

Collins, G. E. (1974). The computing time of the Euclidean algorithm. *SIAM J. Comput.* **3**, 1-10.

Dixon, J. D. (1970). The number of steps in the Euclidean algorithm. *J. Number Theory* **2**, 414-422.

von zur Gathen, J. (1984). Parallel algorithms for algebraic problems. *SIAM J. Comput.* **13**, 802-824.

von zur Gathen, J. (1986). Representations and parallel computations for rational functions. *SIAM J. Comput.* **15**, 432-452.

Heilbronn, H. (1968). On the average length of a class of continued fractions. In: *Abhandlungen aus Zahlentheorie und Analysis*, ed. P. Turán, pp. 87-96. Berlin: VEB Deutscher Verlag.

Knuth, D. E. (1970). The analysis of algorithms. *Proc. Internat. Congress Math.*, Nice, vol. 3, pp. 269-274, Paris: Gauthier-Villars.

Knuth, D. E. (1981). *The Art of Computer Programming,* vol. 2, *Semi-numerical Algorithms.* Reading, MA: Addison-Wesley.

Lamé, G. (1844). 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**, 867-870.

Lehmer, D. H. (1938). Euclid's algorithm for large numbers. *Amer. Math. Monthly* **45**, 227-233.

Ma, K. (1987). Analysis of polynomial GCD computations over finite fields. *Tech. Rep.* **195**, Dept. of Computer Science, University of Toronto, 93 pp.

MacWilliams, F. J., Sloane, N. J. A. (1977). *The theory of error-correcting codes.* Amsterdam: North-Holland.

Moenck, R. (1973). Fast computation of GCD's. *Proc. Fifth ACM Symp. on Theory of Computing,* New York, NY, pp. 142-151.

Rabin, M. O. (1980). Probabilistic algorithms in finite fields. *SIAM J. Comput.* **9**, 273-280.

Schönhage, A. (1971). Schnelle Berechnung von Kettenbruchentwicklungen. *Acta Inform.* **1**, 139-144.

Stein, J. (1967). Computational problems associated with Racah Algebra. *J. Comp. Phys.* **1**, 397-405.

Strassen, V. (1983). The computational complexity of continued fractions. *SIAM J. Comput.* **12**, 1-27.