MULTIPLE GCDs:
Probabilistic analysis of the plain algorithm

Valérie Berthé (1), Jean Creusefond (2),
Loïck Lhote (3), Brigitte Vallée (3)

(1) : LIAFA (CNRS and Université Paris Diderot), France
(2) : INSA de Rouen, France
(3) : GREYC (CNRS, Université de Caen and ENSICAEN), France

DYNALCO MiniSchool,
Buenos-Aires, February 2014
Computing GCDs of \( \ell \) inputs

For \( \ell = 2 \): the “classical” Euclid algorithm,
For \( \ell \geq 3 \), there are various strategies.
Computing GCDs of $\ell$ inputs

For $\ell = 2$: the “classical” Euclid algorithm,

For $\ell \geq 3$, there are various strategies.

The plain algorithm performs a sequence of computations on two entries;
On the input $(x_1, x_2, \ldots, x_\ell)$, it computes

- first: $y_2 := \gcd(x_1, x_2)$
- then, for $k \in [3..\ell]$: $y_k := \gcd(x_k, y_{k-1}) = \gcd(x_1, x_2, \ldots, x_k)$. 

Computing GCDs of ℓ inputs

For ℓ = 2: the “classical” Euclid algorithm, 
For ℓ ≥ 3, there are various strategies.

The plain algorithm performs a sequence of computations on two entries; 
On the input \((x_1, x_2, \ldots, x_\ell)\), it computes 
  – first: \(y_2 := \gcd(x_1, x_2)\) 
  – then, for \(k \in [3..\ell]\): \(y_k := \gcd(x_k, y_{k-1}) = \gcd(x_1, x_2, \ldots, x_k)\).

The “total” gcd \(y_\ell := \gcd(x_1, x_2, \ldots, x_\ell)\) is obtained after \(\ell - 1\) phases.
Each phase performs a call to the classical Euclid algorithm.
Computing GCDs of $\ell$ inputs

For $\ell = 2$: the “classical” Euclid algorithm,
For $\ell \geq 3$, there are various strategies.

The plain algorithm performs a sequence of computations on two entries; On the input $(x_1, x_2, \ldots, x_\ell)$, it computes

- first: $y_2 := \gcd(x_1, x_2)$
- then, for $k \in [3..\ell]$: $y_k := \gcd(x_k, y_{k-1}) = \gcd(x_1, x_2, \ldots, x_k)$.

The “total” gcd $y_\ell := \gcd(x_1, x_2, \ldots, x_\ell)$ is obtained after $\ell - 1$ phases. Each phase performs a call to the classical Euclid algorithm.

The same formal scheme

- for polynomials over a finite field: $\mathbb{F}_q[X]$
- for numbers : positive integers.
Computing GCDs of $\ell$ inputs

For $\ell = 2$: the “classical” Euclid algorithm,
For $\ell \geq 3$, there are various strategies.

The plain algorithm performs a sequence of computations on two entries;
On the input $(x_1, x_2, \ldots, x_\ell)$, it computes

- first: $y_2 := \gcd(x_1, x_2)$
- then, for $k \in [3..\ell]$: $y_k := \gcd(x_k, y_{k-1}) = \gcd(x_1, x_2, \ldots, x_k)$.

The “total” $\gcd y_\ell := \gcd(x_1, x_2, \ldots, x_\ell)$ is obtained after $\ell - 1$ phases.
Each phase performs a call to the classical Euclid algorithm.

The same formal scheme

- for polynomials over a finite field: $\mathbb{F}_q[X]$
- for numbers: positive integers.

A very natural scheme, proposed in Knuth’s book, but not yet analyzed.
In this talk, – we focus on the analysis of the polynomial case,
- we also explain the similarities between the two analyses.
Which behavior can be expected?

Knuth wrote: “In most cases, the length of the partial gcd decreases rapidly during the first few phases of the calculation. This will make the remainder of the computation quite fast”.

Our analysis exhibits a more precise phenomenon: a strong difference between the first phase and the subsequent phases. In most cases, “almost all the calculation” is done during the first phase. We prove the following facts about the number of divisions performed, measured with respect to the length of the input:

- during the first phase:
  - it is linear on average,
  - it asymptotically follows a beta law;
- during subsequent phases:
  - it is constant on average
  - it asymptotically follows a geometric law

The same phenomena occur for the size of the partial gcd.
Which behavior can be expected?

Knuth wrote: “In most cases, the length of the partial gcd decreases rapidly during the first few phases of the calculation. This will make the remainder of the computation quite fast”.

Our analysis exhibits a more precise phenomenon: A strong difference between the first phase and the subsequent phases. In most cases, “almost all the calculation” is done during the first phase.
Which behavior can be expected?

Knuth wrote: “In most cases, the length of the partial gcd decreases rapidly during the first few phases of the calculation. This will make the remainder of the computation quite fast”.

Our analysis exhibits a more precise phenomenon: A strong difference between the first phase and the subsequent phases. In most cases, “almost all the calculation” is done during the first phase.

We prove the following facts about the number of divisions performed, measured with respect to the length of the input:

- during the first phase:
  - it is linear on average,
  - it asymptotically follows a beta law;
Which behavior can be expected?

Knuth wrote: “In most cases, the length of the partial gcd decreases rapidly during the first few phases of the calculation. This will make the remainder of the computation quite fast”.

Our analysis exhibits a more precise phenomenon: A strong difference between the first phase and the subsequent phases. In most cases, “almost all the calculation” is done during the first phase.

We prove the following facts about the number of divisions performed, measured with respect to the length of the input:

– during the first phase:
  – it is linear on average,
  – it asymptotically follows a beta law;

– during subsequent phases:
  – it is constant on average
  – it asymptotically follows a geometric law
Which behavior can be expected?

Knuth wrote: “In most cases, the length of the partial gcd decreases rapidly during the first few phases of the calculation. This will make the remainder of the computation quite fast”.

Our analysis exhibits a more precise phenomenon: A strong difference between the first phase and the subsequent phases. In most cases, “almost all the calculation” is done during the first phase.

We prove the following facts about the number of divisions performed, measured with respect to the length of the input:

- during the first phase:
  - it is linear on average,
  - it asymptotically follows a beta law;

- during subsequent phases:
  - it is constant on average
  - it asymptotically follows a geometric law

The same phenomena occur for the size of the partial gcd.
Probabilistic analysis of an algorithm and generating functions

– The set of the possible inputs for the algorithm is denoted by $\Omega$. 
Probabilistic analysis of an algorithm and generating functions

– The set of the possible inputs for the algorithm is denoted by $\Omega$.

– There is a size function $|| \cdot ||$ on $\Omega$.
  
  – The subset $\Omega_n := \{ \omega : ||\omega|| = n \}$ is finite
  
  – It is endowed with the uniform distribution $\mathbb{P}_n$.
Probabilistic analysis of an algorithm and generating functions

– The set of the possible inputs for the algorithm is denoted by $\Omega$.
– There is a size function $|| \cdot ||$ on $\Omega$.
  – The subset $\Omega_n := \{ \omega : ||\omega|| = n \}$ is finite
  – It is endowed with the uniform distribution $\mathbb{P}_n$.
– There is a cost function $L$ on $\Omega$,
  – study the probabilistic behavior of $L$ on each $\Omega_n$
    – estimate its mean, its variance, its distribution,
  – in an asymptotic way (for $n \to \infty$).
Probabilistic analysis of an algorithm and generating functions

– The set of the possible inputs for the algorithm is denoted by $\Omega$.

– There is a size function $|| \cdot ||$ on $\Omega$.
  – The subset $\Omega_n := \{ \omega : ||\omega|| = n \}$ is finite
  – It is endowed with the uniform distribution $\mathbb{P}_n$.

– There is a cost function $L$ on $\Omega$,
  – study the probabilistic behavior of $L$ on each $\Omega_n$
    – estimate its mean, its variance, its distribution,
  – in an asymptotic way (for $n \to \infty$).

\[ \mathbb{E}_n[L] \sim_{n \to \infty} a_n, \quad \mathbf{V}_n[L] \sim_{n \to \infty} b_n, \quad \mathbb{P}_n \left[ \frac{L - a_n}{\sqrt{b_n}} \in [x, x + dx] \right] \sim_{n \to \infty} f(x) dx \]
Probabilistic analysis of an algorithm and generating functions

– The set of the possible inputs for the algorithm is denoted by $\Omega$.

– There is a size function $\| \cdot \|$ on $\Omega$.
  – The subset $\Omega_n := \{ \omega : \| \omega \| = n \}$ is finite
  – It is endowed with the uniform distribution $\mathbb{P}_n$.

– There is a cost function $L$ on $\Omega$,
  – study the probabilistic behavior of $L$ on each $\Omega_n$
    – estimate its mean, its variance, its distribution,
    – in an asymptotic way (for $n \to \infty$).

$$E_n[L] \xrightarrow{n \to \infty} a_n, \quad \forall n[L] \xrightarrow{n \to \infty} b_n, \quad \mathbb{P}_n \left[ \frac{L - a_n}{\sqrt{b_n}} \in [x, x + dx] \right] \xrightarrow{n \to \infty} f(x)dx$$

– Generating functions $S(z) := \sum_{\omega \in \Omega} z^{\| \omega \|}, \quad S(z, u) := \sum_{\omega \in \Omega} z^{\| \omega \|} u^{L(\omega)},$
Probabilistic analysis of an algorithm and generating functions

– The set of the possible inputs for the algorithm is denoted by \( \Omega \).
– There is a size function \( \| \cdot \| \) on \( \Omega \).
  – The subset \( \Omega_n := \{ \omega : \| \omega \| = n \} \) is finite
  – It is endowed with the uniform distribution \( \mathbb{P}_n \).
– There is a cost function \( L \) on \( \Omega \),
  – study the probabilistic behavior of \( L \) on each \( \Omega_n \)
    – estimate its mean, its variance, its distribution,
    – in an asymptotic way (for \( n \to \infty \)).

\[
\mathbb{E}_n[L] \sim_{n \to \infty} a_n, \quad \mathbb{V}_n[L] \sim_{n \to \infty} b_n, \quad \mathbb{P}_n \left[ \frac{L - a_n}{\sqrt{b_n}} \in [x, x + dx] \right] \sim_{n \to \infty} f(x)dx
\]

– Generating functions \( S(z) := \sum_{\omega \in \Omega} z^{\| \omega \|}, \quad S(z, u) := \sum_{\omega \in \Omega} z^{\| \omega \|} u^L(\omega) \),

A main tool for studying distributions: \( \mathbb{P}_n[L > m] = \sum_{j > m} \frac{[z^n u^j]S(z, u)}{[z^n]S(z)} \)
Examples of limit laws

\( x \)-axis: possible values of the cost \( L(\omega) \)

\( y \)-axis: probability density \( x \mapsto f(x) \)

\[ f(x) \, dx := \mathbb{P}[\omega; \ L(\omega) \in [x, x + dx]] \]

Gaussian law

Beta law

Uniform law

Geometric law
Probabilistic analysis of the plain $\ell$–GCD algorithm on $\mathbb{F}_q[X]$.

On the input $(x_1, x_2, \ldots, x_\ell)$,

- the algorithm computes the total gcd $y_\ell := \gcd(x_1, x_2, \ldots, x_\ell)$
- with $\ell - 1$ phases.
- The $k$-th phase computes the $k$–th gcd,
  $$y_k := \gcd(x_k, y_{k-1}) = \gcd(x_1, x_2, \ldots, x_k).$$
Probabilistic analysis of the plain $\ell$–GCD algorithm on $\mathbb{F}_q[X]$.

On the input $(x_1, x_2, \ldots, x_\ell)$,

- the algorithm computes the total gcd $y_\ell := \gcd(x_1, x_2, \ldots, x_\ell)$
- with $\ell - 1$ phases.
- The $k$-th phase computes the $k$–th gcd, 
  
  $y_k := \gcd(x_k, y_{k-1}) = \gcd(x_1, x_2, \ldots, x_k)$.

- each phase performs the classical Euclid algorithm
  via a sequence of Euclidean divisions
Probabilistic analysis of the plain $\ell$–GCD algorithm on $\mathbb{F}_q[X]$.

On the input $(x_1, x_2, \ldots, x_\ell)$,

– the algorithm computes the total gcd $y_\ell := \gcd(x_1, x_2, \ldots, x_\ell)$
– with $\ell - 1$ phases.
– The $k$-th phase computes the $k$–th gcd,
\[ y_k := \gcd(x_k, y_{k-1}) = \gcd(x_1, x_2, \ldots, x_k) . \]
– each phase performs the classical Euclid algorithm
  via a sequence of Euclidean divisions

The set of inputs is $\Omega = \{(x_1, \ldots, x_\ell); \ x_i \text{ monic} \in \mathbb{F}_q[X]\}$
The size of an input : $||(x_1, \ldots, x_\ell)|| = d(x_1) + \ldots + d(x_\ell)$
Probabilistic analysis of the plain $\ell$–GCD algorithm on $\mathbb{F}_q[X]$.

On the input $(x_1, x_2, \ldots, x_\ell)$,
- the algorithm computes the total gcd $y_\ell := \gcd(x_1, x_2, \ldots, x_\ell)$
- with $\ell - 1$ phases.
- The $k$-th phase computes the $k$–th gcd,
  \[ y_k := \gcd(x_k, y_{k-1}) = \gcd(x_1, x_2, \ldots, x_k) \]
- each phase performs the classical Euclid algorithm
  via a sequence of Euclidean divisions

The set of inputs is $\Omega = \{(x_1, \ldots, x_\ell); \ x_i \text{ monic } \in \mathbb{F}_q[X]\}$

The size of an input : $\|(x_1, \ldots, x_\ell)\| = d(x_1) + \ldots + d(x_\ell)$

Main parameters of interest
- the number $L_k$ of divisions during the $k$–th phase
  i.e. on the input $(x_k, y_{k-1})$
- the degree $D_k$ of the $k$–th gcd
  (at the beginning of the $k$-th phase).
The combinatorial bijection induced by the Euclid Algorithm \([\ell = 2]\)
The combinatorial bijection induced by the Euclid Algorithm \([\ell = 2]\)

\[
\text{Euclid}(a_1, a_2), \quad \text{[case } d(a_1) \geq d(a_2)\text{].}
\]

\[
\begin{align*}
a_1 &= m_1 a_2 + a_3 \quad &0 < d(a_3) < d(a_2) \\
a_2 &= m_2 a_3 + a_4 \quad &0 < d(a_4) < d(a_3) \\
\vdots &= \vdots + \\
a_{r-1} &= m_{r-1} a_r + a_{r+1} \quad &0 < d(a_{r+1}) < d(a_r) \\
a_r &= m_r a_{r+1} + 0
\end{align*}
\]

The last non zero remainder is the gcd \(y\). Here \(y = a_{r+1}\).
The combinatorial bijection induced by the Euclid Algorithm \([\ell = 2]\)

| Euclid\((a_1, a_2)\), \[\text{case } d(a_1) \geq d(a_2)\]. |
|---|---|
| \(a_1\)       | \(m_1 a_2 + a_3\) | \(0 < d(a_3) < d(a_2)\) |
| \(a_2\)       | \(m_2 a_3 + a_4\) | \(0 < d(a_4) < d(a_3)\) |
| \(\ldots\)   | \(\ldots\)         | \(\) |
| \(a_{r-1}\)  | \(m_{r-1} a_r + a_{r+1}\) | \(0 < d(a_{r+1}) < d(a_r)\) |
| \(a_r\)      | \(m_r a_{r+1} + 0\) | \(\) |

The last non-zero remainder is the gcd \(y\). Here \(y = a_{r+1}\).

The Euclid Algorithm is then extended to the case when \(d(a_1) < d(a_2)\), by letting \(\text{Euclid}(a_1, a_2) := \text{Euclid}(a_2, a_1)\)
The combinatorial bijection induced by the Euclid Algorithm \([\ell = 2]\)

\[
\begin{align*}
\text{Euclid}(a_1, a_2), \quad & \text{[case } d(a_1) \geq d(a_2)\text{].} \\
\end{align*}
\]

\[
\begin{align*}
a_1 &= m_1 a_2 + a_3 \quad & 0 < d(a_3) < d(a_2) \\
a_2 &= m_2 a_3 + a_4 \quad & 0 < d(a_4) < d(a_3) \\
& \quad \quad \ldots + \\
a_{r-1} &= m_{r-1} a_r + a_{r+1} \quad & 0 < d(a_{r+1}) < d(a_r) \\
a_r &= m_r a_{r+1} + 0
\end{align*}
\]

The last non-zero remainder is the gcd \(y\). Here \(y = a_{r+1}\).

The Euclid Algorithm is then extended to the case when \(d(a_1) < d(a_2)\),
by letting \(\text{Euclid}(a_1, a_2) := \text{Euclid}(a_2, a_1)\)

The pair \((a_1, a_2)\) of monic polynomials is entirely determined by
– the sequence of quotients \((m_1, m_2, \ldots, m_r)\),
The combinatorial bijection induced by the Euclid Algorithm \([\ell = 2]\)

\[
\begin{align*}
\text{Euclid}(a_1, a_2), & \quad [\text{case } d(a_1) \geq d(a_2)]. \\
a_1 &= m_1 a_2 + a_3 \quad 0 < d(a_3) < d(a_2) \\
a_2 &= m_2 a_3 + a_4 \quad 0 < d(a_4) < d(a_3) \\
\ldots &= \ldots + \\
a_{r-1} &= m_{r-1} a_r + a_{r+1} \quad 0 < d(a_{r+1}) < d(a_r) \\
a_r &= m_r a_{r+1} + 0
\end{align*}
\]

The last non zero remainder is the gcd \(y\). Here \(y = a_{r+1}\).

The Euclid Algorithm is then extended to the case when \(d(a_1) < d(a_2)\), by letting \(\text{Euclid}(a_1, a_2) := \text{Euclid}(a_2, a_1)\)

The pair \((a_1, a_2)\) of monic polynomials is entirely determined by

– the sequence of quotients \((m_1, m_2, \ldots, m_r)\), where

– the first quotient \(m_1\) is monic, with

\[
d(m_1) \geq 0 \quad [d(a_1) \geq d(a_2)] \quad \text{or} \quad d(m_1) > 0 \quad [d(a_2) > d(a_2)]
\]
The combinatorial bijection induced by the Euclid Algorithm [\(\ell = 2\)]

\[
\begin{align*}
\text{Euclid}(a_1, a_2),  \quad & \text{[case } d(a_1) \geq d(a_2)\text{].} \\
a_1 & = m_1 a_2 + a_3 \quad 0 < d(a_3) < d(a_2) \\
a_2 & = m_2 a_3 + a_4 \quad 0 < d(a_4) < d(a_3) \\
\vdots & = \ldots + \\
a_{r-1} & = m_{r-1} a_r + a_{r+1} \quad 0 < d(a_{r+1}) < d(a_r) \\
a_r & = m_r a_{r+1} + 0 \quad a_r = m_r a_{r+1} + 0
\end{align*}
\]

The last non zero remainder is the gcd \(y\). Here \(y = a_{r+1}\).

The Euclid Algorithm is then extended to the case when \(d(a_1) < d(a_2)\), by letting \(\text{Euclid}(a_1, a_2) := \text{Euclid}(a_2, a_1)\).

The pair \((a_1, a_2)\) of monic polynomials is entirely determined by

– the sequence of quotients \((m_1, m_2, \ldots, m_r)\), where

  – the first quotient \(m_1\) is monic, with

  \[
  d(m_1) \geq 0 \quad [d(a_1) \geq d(a_2)] \text{ or } d(m_1) > 0 \quad [d(a_2) > d(a_2)]
  \]

  – any quotient \(m_i\) for \(i \in [2..r]\) is general with \(d(m_i) > 0\)
The combinatorial bijection induced by the Euclid Algorithm $[\ell = 2]$

<table>
<thead>
<tr>
<th>Euclid($a_1, a_2$), case $d(a_1) \geq d(a_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1 = m_1 a_2 + a_3$ \quad 0 &lt; d(a_3) &lt; d(a_2)</td>
</tr>
<tr>
<td>$a_2 = m_2 a_3 + a_4$ \quad 0 &lt; d(a_4) &lt; d(a_3)</td>
</tr>
<tr>
<td>... = ... +</td>
</tr>
<tr>
<td>$a_{r-1} = m_{r-1} a_r + a_{r+1}$ \quad 0 &lt; d(a_{r+1}) &lt; d(a_r)</td>
</tr>
<tr>
<td>$a_r = m_r a_{r+1} + 0$</td>
</tr>
</tbody>
</table>

The last non zero remainder is the gcd $y$. Here $y = a_{r+1}$.

The Euclid Algorithm is then extended to the case when $d(a_1) < d(a_2)$, by letting $\text{Euclid}(a_1, a_2) := \text{Euclid}(a_2, a_1)$

The pair $(a_1, a_2)$ of monic polynomials is entirely determined by

- the sequence of quotients $(m_1, m_2, \ldots, m_r)$, where
  - the first quotient $m_1$ is monic, with $d(m_1) \geq 0$ [\text{d}(a_1) \geq \text{d}(a_2)] or $d(m_1) > 0$ [\text{d}(a_2) > \text{d}(a_2)]
  - any quotient $m_i$ for $i \in [2..r]$ is general with $d(m_i) > 0$
- the monic gcd $y = a_{r+1}$. 

Generating functions relative to the Euclid algorithm \((\ell = 2)\).
Generating functions relative to the Euclid algorithm \((\ell = 2)\).

\(U(z)\) is the gen. function of the set \(U\) of monic polynomials
\n\(G(z)\) is the gen. function of the general non constant polynomials
Generating functions relative to the Euclid algorithm ($\ell = 2$).

$U(z)$ is the gen. function of the set $\mathcal{U}$ of monic polynomials

$G(z)$ is the gen. function of the general non constant polynomials

\[
U(z) = \sum_{a \in \mathcal{U}} z^{d(a)} = \frac{1}{1 - qz}, \quad G(z) = (q - 1) [U(z) - 1] = \frac{(q - 1)qz}{1 - qz}
\]
Generating functions relative to the Euclid algorithm ($\ell = 2$).

$U(z)$ is the gen. function of the set $\mathcal{U}$ of monic polynomials

$G(z)$ is the gen. function of the general non constant polynomials

\[ U(z) = \sum_{a \in \mathcal{U}} z^{d(a)} = \frac{1}{1 - qz}, \quad G(z) = (q - 1) [U(z) - 1] = \frac{(q - 1)qz}{1 - qz} \]

$\Omega = \mathcal{U}^2 \approx \{\text{first quotient}\} \times \{\text{sequence of quotients}\} \times \{\text{GCD}\}$

$(a_1, a_2) \approx m_1 \times (m_2, \ldots, m_r) \times y$
Generating functions relative to the Euclid algorithm \((\ell = 2)\).

\(U(z)\) is the gen. function of the set \(\mathcal{U}\) of monic polynomials

\(G(z)\) is the gen. function of the general non constant polynomials

\[
U(z) = \sum_{a \in \mathcal{U}} z^{d(a)} = \frac{1}{1 - qz}, \quad G(z) = (q - 1) [U(z) - 1] = \frac{(q - 1)qz}{1 - qz}
\]

\[
\Omega = \mathcal{U}^2 \approx \{\text{first quotient}\} \times \{\text{sequence of quotients}\} \times \{\text{GCD}\}
\]

\( (a_1, a_2) \approx m_1 \times (m_2, \ldots, m_r) \times y \)

\[
U(z_1)U(z_2) = U(z_1) + [U(z_2) - 1] \cdot \frac{1}{1 - G(z_1z_2)} \cdot U(z_1z_2)
\]
Generating functions relative to the Euclid algorithm \((\ell = 2)\).

\(U(z)\) is the gen. function of the set \(U\) of monic polynomials

\(G(z)\) is the gen. function of the general non constant polynomials

\[
U(z) = \sum_{a \in U} z^{d(a)} = \frac{1}{1 - qz}, \quad G(z) = (q - 1) [U(z) - 1] = \frac{(q - 1)qz}{1 - qz}
\]

\[
\Omega = U^2 \approx \{\text{first quotient}\} \times \{\text{sequence of quotients}\} \times \{\text{GCD}\}
\]

\((a_1, a_2) \approx m_1 \times (m_2, \ldots, m_r) \times y
\]

\[
U(z_1) U(z_2) = U(z_1) + [U(z_2) - 1] \cdot \frac{1}{1 - G(z_1z_2)} \cdot U(z_1z_2)
\]

Proof: there are two cases : (I) \(d(a_1) \geq d(a_2)\) or (II) \(d(a_2) > d(a_1)\)

\[
\begin{aligned}
(I) & \quad d(a_1) = d(m_1) + d(m_2) + \ldots + d(m_r) + d(y) \\
& \quad z_1^{d(a_1)} z_2^{d(a_2)} = z_1^{d(m_1)} \cdot (z_1z_2)^{d(m_2)+\ldots+d(m_r)} \cdot (z_1z_2)^{d(y)}
\end{aligned}
\]
Generating functions relative to the Euclid algorithm ($\ell = 2$).

$U(z)$ is the gen. function of the set $\mathcal{U}$ of monic polynomials

$G(z)$ is the gen. function of the general non constant polynomials

$$U(z) = \sum_{a \in \mathcal{U}} z^{d(a)} = \frac{1}{1 - qz}, \quad G(z) = (q - 1) [U(z) - 1] = \frac{(q - 1)qz}{1 - qz}$$

$$\Omega = \mathcal{U}^2 \approx \{\text{first quotient}\} \times \{\text{sequence of quotients}\} \times \{\text{GCD}\}$$

$(a_1, a_2) \approx m_1 \times (m_2, \ldots, m_r) \times y$

$$U(z_1) U(z_2) = U(z_1) + [U(z_2) - 1] \cdot \frac{1}{1 - G(z_1z_2)} \cdot U(z_1z_2)$$

Proof: there are two cases: (I) $d(a_1) \geq d(a_2)$ or (II) $d(a_2) > d(a_1)$

<table>
<thead>
<tr>
<th></th>
<th>$d(a_1)$</th>
<th>$d(a_2)$</th>
<th>$z_1^{d(a_1)} z_2^{d(a_2)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(I)</td>
<td>$d(m_1)$</td>
<td>$d(m_2) + \ldots + d(m_r) + d(y)$</td>
<td>$z_1^{d(m_1)} \cdot (z_1z_2)^{d(m_2)+\ldots+d(m_r)} \cdot (z_1z_2)^{d(y)}$</td>
</tr>
<tr>
<td>(II)</td>
<td>$d(m_1) + d(m_2) + \ldots + d(m_r) + d(y)$</td>
<td>$z_2^{d(m_1)} \cdot (z_1z_2)^{d(m_2)+\ldots+d(m_r)} \cdot (z_1z_2)^{d(y)}$</td>
<td></td>
</tr>
</tbody>
</table>
Generating function relative to the \( \ell \)-Euclid Algorithm

We have shown that the Euclid algorithm \((\ell = 2)\) translates as a product

\[
U(z_1)U(z_2) = T(z_1, z_2)U(z_1z_2), \quad \text{with} \quad T(z, t) = \frac{U(z) + U(t) - 1}{1 - G(zt)}
\]
Generating function relative to the $\ell$-Euclid Algorithm

We have shown that the Euclid algorithm ($\ell = 2$) translates as a product

$$U(z_1) U(z_2) = T(z_1, z_2) U(z_1 z_2), \quad \text{with} \quad T(z, t) = \frac{U(z) + U(t) - 1}{1 - G(zt)}$$

Then, for any $\ell \geq 2$, the $\ell$–Euclid algorithm translates as the product

$$U(z_1) \cdot \ldots \cdot U(z_\ell) = U(t_\ell) \prod_{k=1}^{\ell-1} T(z_{k+1}, t_k) \quad [t_k := z_1 \cdot z_2 \cdot \ldots \cdot z_k, \,]$$
Generating function relative to the $\ell$-Euclid Algorithm

We have shown that the Euclid algorithm ($\ell = 2$) translates as a product

$$U(z_1)U(z_2) = T(z_1, z_2)U(z_1z_2), \quad \text{with} \quad T(z, t) = \frac{U(z) + U(t) - 1}{1 - G(zt)}$$

Then, for any $\ell \geq 2$, the $\ell$–Euclid algorithm translates as the product

$$U(z_1) \cdot \ldots \cdot U(z_\ell) = U(t_\ell) \prod_{k=1}^{\ell-1} T(z_{k+1}, t_k) \quad \left[ t_k := z_1 \cdot z_2 \cdot \ldots \cdot z_k, \right]$$

Now, with $z = z_1 = \ldots = z_\ell$,

the (plain) generating function $S(z)$ of $U^\ell$ has an alternative expression

$$S(z) = U(z)^\ell = U(z^\ell) \prod_{k=1}^{\ell-1} T(z, z^k)$$

which is an exact translation of the $\ell$-Euclid algorithm.

$T$ is the “phase generating function”.
Generating functions relative to the $\ell$-Euclid Algorithm

We start with: 

$$S(z) = U(z)^\ell = U(z^\ell) \prod_{k=1}^{\ell-1} T(z, z^k)$$
We start with:

\[ S(z) = U(z)^\ell = U(z^\ell) \prod_{k=1}^{\ell-1} T(z, z^k) \]

For studying the **distribution** of the two parameters:
- \( L_k \) (number of steps in the \( k \)-th phase)
- \( D_k \) (degree of the gcd at the beginning of the \( k \)-th phase)

we use **bivariate** generating functions, with an extra variable \( u \)
Generating functions relative to the $\ell$-Euclid Algorithm

We start with:

$$S(z) = U(z)^\ell = U(z^\ell) \prod_{k=1}^{\ell-1} T(z, z^k)$$

For studying the distribution of the two parameters:

- $L_k$ (number of steps in the $k$-th phase)
- $D_k$ (degree of the gcd at the beginning of the $k$-th phase)

we use bivariate generating functions, with an extra variable $u$

$$L_k(z, u) = U(z)^\ell \cdot \frac{T(z, z^k, u)}{T(z, z^k)}$$
$$D_k(z, u) = U(z)^\ell \cdot \frac{U(z^k, u)}{U(z^k)}$$

with

$$T(z, t, u) = u \frac{U(z) + U(t) - 1}{1 - uG(zt)}$$
$$U(t, u) = \frac{1}{1 - qu t}.$$
Generating functions relative to the $\ell$-Euclid Algorithm

We start with:

$$S(z) = U(z)^\ell = U(z^\ell) \prod_{k=1}^{\ell-1} T(z, z^k)$$

For studying the distribution of the two parameters:
- $L_k$ (number of steps in the $k$-th phase)
- $D_k$ (degree of the gcd at the beginning of the $k$-th phase)

we use bivariate generating functions, with an extra variable $u$

$$L_k(z, u) = U(z)^\ell \cdot \frac{T(z, z^k, u)}{T(z, z^k)}, \quad D_k(z, u) = U(z)^\ell \cdot \frac{U(z^k, u)}{U(z^k)},$$

with

$$T(z, t, u) = u \frac{U(z) + U(t) - 1}{1 - u G(zt)}, \quad U(t, u) = \frac{1}{1 - qut}.$$

For the expectations, the cumulative generating functions are useful:

$$\hat{L}_k(z) := \left. \frac{\partial}{\partial u} L_k(z, u) \right|_{u=1} = U(z)^\ell \left( \frac{1 - qz^{k+1}}{1 - q^2 z^{k+1}} \right),$$

$$\hat{D}_k(z) := \left. \frac{\partial}{\partial u} D_k(z, u) \right|_{u=1} = U(z)^\ell \left( \frac{qz^k}{1 - qz^k} \right).$$
Towards the distributional analysis of $L_k$ and $D_k$.

\[
\mathbb{P}_n[L_k > m] = \sum_{j > m} \frac{[u^j z^n]L_k(z, u)}{[z^n]S(z)} = \frac{[z^n] \sum_{j > m}[u^j]L_k(z, u)}{[z^n]S(z)} = \frac{[z^n] \hat{L}^m_k(z)}{[z^n]S(z)}
\]

\[
\mathbb{P}_n[D_k > m] = \sum_{j > m} \frac{[u^j z^n]D_k(z, u)}{[z^n]S(z)} = \frac{[z^n] \sum_{j > m}[u^j]D_k(z, u)}{[z^n]S(z)} = \frac{[z^n] \hat{D}^m_k(z)}{[z^n]S(z)}
\]
Towards the distributional analysis of $L_k$ and $D_k$.

\[
\mathbb{P}_n[L_k > m] = \sum_{j > m} \frac{[u^j z^n]L_k(z, u)}{[z^n]S(z)} = \frac{[z^n]\sum_{j > m}[u^j]L_k(z, u)}{[z^n]S(z)} = \frac{[z^n]\hat{L}_k^m(z)}{[z^n]S(z)}
\]

\[
\mathbb{P}_n[D_k > m] = \sum_{j > m} \frac{[u^j z^n]D_k(z, u)}{[z^n]S(z)} = \frac{[z^n]\sum_{j > m}[u^j]D_k(z, u)}{[z^n]S(z)} = \frac{[z^n]\hat{D}_k^m(z)}{[z^n]S(z)}
\]

The generating functions “of the numerator” are

\[
\hat{L}_k^m(z) = \frac{1}{(1 - qz)^\ell} \cdot G(z^{k+1})^m, \quad \hat{D}_k^m(z) = \frac{1}{(1 - qz)^\ell} \cdot (qz^k)^m,
\]

both of type

\[
\frac{1}{(1 - qz)^\ell} \cdot A_k(z)^m,
\]
Towards the distributional analysis of $L_k$ and $D_k$.

$$
P_n[L_k > m] = \sum_{j > m} \frac{[u^j z^n]L_k(z, u)}{[z^n]S(z)} = \frac{[z^n] \sum_{j > m} [u^j]L_k(z, u)}{[z^n]S(z)} = \frac{[z^n] \hat{L}_k^m(z)}{[z^n]S(z)}
$$

$$
P_n[D_k > m] = \sum_{j > m} \frac{[u^j z^n]D_k(z, u)}{[z^n]S(z)} = \frac{[z^n] \sum_{j > m} [u^j]D_k(z, u)}{[z^n]S(z)} = \frac{[z^n] \hat{D}_k^m(z)}{[z^n]S(z)}
$$

The generating functions “of the numerator” are

$$
\hat{L}_k^m(z) = \frac{1}{(1 - qz)^\ell} \cdot G(z^{k+1})^m, \quad \hat{D}_k^m(z) = \frac{1}{(1 - qz)^\ell} \cdot (qz^k)^m,
$$

both of type $$
\frac{1}{(1 - qz)^\ell} \cdot A_k(z)^m,
$$

The asymptotics depends on the value $a := A_k(1/q)$ at the pole $z = 1/q$

- For the first phase $k = 1$, one has $a = 1$
- For the subsequent phases $k \geq 2$, one has $a < 1$
A general result.

Consider the function \[ F^m(z) = \frac{1}{(1 - z)^\ell} A(z)^m, \]

where (i) \( A(z) \) is analytic on the disk \( |z| \leq \rho \) with \( \rho > 1 \),
(ii) \( a := A(1) \neq 0, \quad b := A'(1) > 0, \)
(iii) for \( |z| \) close enough to 1, \( |A(z)| \leq A(|z|) \).

Then, for any \( c \in [0, a/b] \), when \( m/n \to c \), one has:

\[
[z^n] F^m(z) = \frac{n^{\ell-1}}{\ell - 1)!} a^m \left( 1 - \frac{b}{a}c \right)^{\ell-1} \left[ 1 + O \left( \frac{1}{n} \right) \right].
\]

Application to the present situation.

For the first phase: \( a = 1 \implies \) A “beta” behavior \((1, \ell - 1)\)
For the subsequent phases: \( a < 1 \implies \) A geometric behavior of ratio \( a \)
Main result for the number of divisions $L_k$ – First phase ($k = 1$)

The number of divisions $L_1$ performed by the $\ell$-Euclid algorithm during the first phase has a mean value of linear order

$$E_n[L_1] = \frac{q - 1}{2q} \frac{n}{\ell} + \frac{3q + 1}{4q} + O\left(\frac{1}{n}\right).$$

$$\frac{2q}{q - 1} = \text{entropy}$$

It follows an asymptotic beta law of parameter $(1, \ell - 1)$ and its distribution satisfies when $n \to \infty$, and $m/n \to c$ with $c \in ]0, (q - 1)/(2q)[$

$$P[L_1 > m] = \left(1 - \frac{2q}{q - 1}c\right)^{\ell - 1} \left[1 + O\left(\frac{1}{n}\right)\right].$$
Main result for the number of divisions $L_k$ – Subsequent phases (case $k \geq 2$)

For $k \geq 2$, the number of divisions performed by the $\ell$-Euclid algorithm during the $k$-th phase
- has a mean value of constant order
- follows an asymptotic geometric law, with ratio $p_k := \frac{q-1}{q^k-1}$

$$
\mathbb{P}_n[L_k \geq m] = \left( \frac{q-1}{q^k-1} \right)^m \left[ 1 + O \left( \frac{m}{n} \right) \right] \text{ for } m = o(n),
$$

$$
\mathbb{E}_n[L_k] = \left[ 1 + \frac{q-1}{q^k-q} \right] \left[ 1 + O \left( \frac{1}{n} \right) \right],
$$
Main result for the number of divisions $L_k$ – Subsequent phases (case $k \geq 2$)

For $k \geq 2$, the number of divisions performed by the $\ell$-Euclid algorithm during the $k$-th phase
– has a mean value of constant order
– follows an asymptotic geometric law, with ratio $p_k := \frac{q - 1}{q^k - 1}$

$$\mathbb{P}_n[L_k \geq m] = \left(\frac{q - 1}{q^k - 1}\right)^m \left[1 + O\left(\frac{m}{n}\right)\right] \quad \text{for } m = o(n),$$

$$\mathbb{E}_n[L_k] = \left[1 + \frac{q - 1}{q^k - q}\right] \left[1 + O\left(\frac{1}{n}\right)\right],$$
And now, for integers?.... there are carries!

The Euclid algorithm ($\ell = 2$) on polynomials translates as a product of power generating functions

$$U(z) U(t) = U(zt) \cdot \frac{1}{1 - G(zt)} \cdot [U(z) + U(t) - 1].$$
And now, for integers?.... there are carries!

The Euclid algorithm ($\ell = 2$) on polynomials translates as a product of power generating functions

$$U(z)U(t) = U(zt) \cdot \frac{1}{1 - G(zt)} \cdot [U(z) + U(t) - 1].$$

The Euclid algorithm ($\ell = 2$) on integers translates as a product of Dirichlet generating functions

$$2\zeta(s)\zeta(t) = \zeta(s + t) \cdot [(I - G_{s+t})^{-1} \circ (G_s + G_t)[1](0)].$$
And now, for integers?.... there are carries!

The Euclid algorithm ($\ell = 2$) on polynomials

translates as a product of power generating functions

$$U(z)U(t) = U(zt) \cdot \frac{1}{1 - G(zt)} \cdot [U(z) + U(t) - 1].$$

The Euclid algorithm ($\ell = 2$) on integers

translates as a product of Dirichlet generating functions

$$2\zeta(s)\zeta(t) = \zeta(s + t) \cdot \left[(I - G_{s+t})^{-1} \circ (G_s + G_t)[1](0)\right]$$

which involves – the Riemann Dirichlet series $\zeta(s) = \sum_{n\geq1} n^{-s}$

– the transfer operator $G_s$ of the dynamical system.
And now, for integers?... there are carries!

The Euclid algorithm ($\ell = 2$) on polynomials translates as a product of power generating functions

$$U(z)U(t) = U(zt) \cdot \frac{1}{1 - G(zt)} \cdot [U(z) + U(t) - 1].$$

The Euclid algorithm ($\ell = 2$) on integers translates as a product of Dirichlet generating functions

$$2\zeta(s)\zeta(t) = \zeta(s + t) \cdot [(I - G_{s+t})^{-1} \circ (G_s + G_t)[1](0)]$$

which involves – the Riemann Dirichlet series $\zeta(s) = \sum_{n \geq 1} n^{-s}$

– the transfer operator $G_s$ of the dynamical system.

This is a functional operator which depends on a complex parameter $s$,

$$G_s[f](t) = \sum_{m \geq 1} \left( \frac{1}{m + t} \right)^s f \left( \frac{1}{m + t} \right)$$

We use the underlying dynamical system, and perform a “dynamical” analysis. It is more involved than the previous one, but provides the same type of results.
### Similarities and differences between the two analyses

<table>
<thead>
<tr>
<th></th>
<th>Polynomials</th>
<th>Integers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GF</strong></td>
<td><strong>Power GF</strong></td>
<td><strong>Dirichlet GF and operators</strong></td>
</tr>
<tr>
<td><strong>Basic tool</strong></td>
<td>( G(z) = \sum_m z^{d(m)} )</td>
<td>( G_s<a href="t">f</a> = \sum_{m \geq 1} \left( \frac{1}{m + t} \right)^s f \left( \frac{1}{m + t} \right) )</td>
</tr>
<tr>
<td><strong>Phase GF</strong></td>
<td>( \frac{U(z^k) + U(z) - 1}{1 - G(z^{k+1})} )</td>
<td>( (I - G_{(k+1)s})^{-1} \circ [G_{ks} + G_s] )</td>
</tr>
<tr>
<td><strong>Singularities</strong></td>
<td>( z \text{ s.t. } G(z^{k+1}) = 1 )</td>
<td>( s \text{ s.t. } \lambda((k + 1)s) = 1 )</td>
</tr>
<tr>
<td><strong>Extraction</strong></td>
<td>Cauchy Formula Disks</td>
<td>Perron Formula Vertical lines</td>
</tr>
<tr>
<td><strong>Contours</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Similarities and differences between the two analyses

<table>
<thead>
<tr>
<th></th>
<th>Polynomials</th>
<th>Integers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GF</strong></td>
<td><strong>Power GF</strong></td>
<td><strong>Dirichlet GF and operators</strong></td>
</tr>
<tr>
<td><strong>Basic tool</strong></td>
<td>$G(z) = \sum_m z^{d(m)}$</td>
<td>$G_s<a href="t">f</a> = \sum_{m \geq 1} \left(\frac{1}{m + t}\right)^s f\left(\frac{1}{m + t}\right)$</td>
</tr>
<tr>
<td><strong>Phase GF</strong></td>
<td>$\frac{U(z^k) + U(z) - 1}{1 - G(z^{k+1})}$</td>
<td>$(I - G_{(k+1)s})^{-1} \circ [G_{ks} + G_s]$</td>
</tr>
<tr>
<td><strong>Singularities</strong></td>
<td>$z$ s.t. $G(z^{k+1}) = 1$</td>
<td>$s$ s.t. $\lambda((k + 1)s) = 1$</td>
</tr>
<tr>
<td><strong>Extraction</strong></td>
<td>Cauchy Formula</td>
<td>Perron Formula</td>
</tr>
<tr>
<td><strong>Contours</strong></td>
<td>Disks</td>
<td>Vertical lines</td>
</tr>
</tbody>
</table>

$\lambda(s)$ is the **dominant eigenvalue** of $G_s$

$\lambda(2) = 1$ ; $\lambda'(2)$ closely related to the entropy
Results in the integer case.

We prove the following facts about the number of divisions performed
Results in the integer case.

We prove the following facts about the number of divisions performed

- during the first phase:
  - it is linear on average,
  - it asymptotically follows a beta law;
Results in the integer case.

We prove the following facts about the number of divisions performed
– during the first phase:
  – it is linear on average,
  – it asymptotically follows a beta law;
– during subsequent phase:
  – it is constant on average
  – it asymptotically follows a geometric law

The same phenomena occur for the size of the partial gcd.
Thank you!