The Euclidean Algorithm

- The Euclidean Algorithm
  - Euclidean Domains
  - The Extended Euclidean Algorithm
  - Cost analysis for $\mathbb{Z}$ and $F[x]$
  - Uniqueness of the gcd. Normal Forms.

Modern Computer Algebra (2nd edition), by Joachim von zur Gathen and Jürgen Gerhard, Pages 43 – 66

The slides are not a substitute for reading the book!
Example: Euclidean algorithm for 126 and 35.

\[
126 = 3 \cdot 35 + 21 \\
35 = 1 \cdot 21 + 14 \\
21 = 1 \cdot 14 + 7 \\
14 = 2 \cdot 7
\]

The greatest common divisor of 126 and 35, \( \text{gcd}(126, 35) = 7 \)

Why does this algorithm work?
The Euclidean algorithm

- Given $a, b \in \mathbb{N}$, with $a = qb + r$ and $a \geq b$. 
- Assume $\text{gcd}(a, b) = m$, $a = x \cdot m$ and $b = y \cdot m$. 

\[
\begin{align*}
  a &= qb + r \\
  r &= a - qb \\
  r &= xm - q(ym) \\
  r &= m(x - qy)
\end{align*}
\]

- remainder $r$ is multiple of $m$
- $\text{gcd}(a, b) = \text{gcd}(b, r) = m$, can be used for recursive algorithm.
The Euclidean Algorithm

Algorithm (The Euclidean Algorithm - recursive)

**INPUT:** \( a, b \in \mathbb{N}, a \geq b \)

**OUTPUT:** \( \gcd(a, b) \)

1. if \( b == 0 \) then
2. \hspace{1em} return \( a \)
3. else
4. \hspace{1em} \( r \leftarrow a \mod b \)
5. \hspace{1em} return \( \gcd(b, r) \)
6. fi
Euclidean Domain

Definition

An integral domain $R$ with $d : R \rightarrow \mathbb{N} \cup \{\infty\}$ is a Euclidean domain, if for all $a, b \in R$ with $b \neq 0$, we can divide $a$ by $b$ with remainder, such that

$$a = qb + r \text{ with } q, r \in R \text{ and } d(r) < d(b)$$

- $q = a \text{ quo } b$ is the quotient.
- $r = a \text{ rem } b$ is the remainder (or $a \text{ mod } b$).
- $q$ and $r$ not necessarily unique.
- $d$ is called a Euclidean function on $R$.
  - $d(a) = |a|$ for $R = \mathbb{Z}$ and $d(a) = \deg(a)$ for $R = F[x]$. 
Euclidean Domain

Definition

Let $R$ be a ring and $a, b, c \in R$. Then $c$ is a greatest common divisor (gcd) of $a$ and $b$, if

1. $c | a$ and $c | b$.
2. if $d | a$ and $d | b$, then $d | c$ for all $d \in R$.

We call $c$ the least common multiple (lcm) of $a$ and $b$, if

1. $a | c$ and $b | c$.
2. if $a | d$ and $b | d$, then $c | d$ for all $d \in R$.

A unit $u \in R$ is any element with multiplicative inverse $v \in R$, s.t. $uv = 1$. Elements $a, b$ are associate $a \sim b$ if $a = ub$ for unit $u \in R$. 
Euclidean Domain

- 3 is a gcd(12, 15), 60 is a lcm(12, 15).
- gcd and lcm are not unique in general.
- In $\mathbb{Z}$, with units $−1$ and $1$, $−3$ and $3$ are all gcd(12, 15).

Unique ($R = \mathbb{Z}$), if defined as non-negative gcd and lcm.
- Integers $a, b$ are coprime, if their gcd is a unit.
- gcd and lcm always exist in a Euclidean Domain.
Euclidean Domain

Lemma

The gcd in \( \mathbb{Z} \) has the following properties, \( \forall a, b, c \in \mathbb{Z} \).

(i) \( \gcd(a, b) = |a| \iff a | b \)

(ii) \( \gcd(a, a) = \gcd(a, 0) = |a| \) and \( \gcd(a, 1) = 1 \)

(iii) \( \gcd(a, b) = \gcd(b, a) \) (commutativity)

(iv) \( \gcd(a, \gcd(b, c)) = \gcd(\gcd(a, b), c) \) (associativity)

(v) \( \gcd(c \cdot a, c \cdot b) = |c| \cdot \gcd(a, b) \) (distributivity)

(vi) \( |a| = |b| \implies \gcd(a, c) = \gcd(b, c) \)
Euclidean Domain

Algorithm (Euclidean Algorithm - traditional)

**Input:** \( f, g \in R \) is a Euclidean Domain, \( d \) Euclidean function.

**Output:** \( \text{gcd}(f, g) \)

(1) \( r_0 \leftarrow f \)
(2) \( r_1 \leftarrow g \)
(3) \( i \leftarrow 1 \)
(4) while \( (r_i \neq 0) \) do
(5) \( r_{i+1} \leftarrow r_{i-1} \rem r_i \)
(6) \( i \leftarrow i + 1 \)
(7) od
(8) return \( r_{i-1} \)
The Extended Euclidean Algorithm

- Example: Euclidean algorithm for 126 and 35.

126 = 3 \cdot 35 + 21
35 = 1 \cdot 21 + 14
21 = 1 \cdot 14 + 7
14 = 2 \cdot 7

- We not only want the gcd but also its representation:

7 = 21 − 1 \cdot 14 = 21 − (35 − 1 \cdot 21)
= 2 \cdot (126 − 3 \cdot 35) − 35 = 2 \cdot 126 − 7 \cdot 35
The Extended Euclidean Algorithm

**Algorithm (Extended Euclidean algorithm)**

**Input:** \( f, g \in R \) is a Euclidean Domain, \( d \) Euclidean function.

**Output:** \( l \in \mathbb{N}, r_i, s_i, t_i \in R \) for \( 0 \leq i \leq l + 1 \) and \( q_i \in R \) for \( 1 \leq i \leq l \)

1. \( r_0 \leftarrow f, s_0 \leftarrow 1, t_0 \leftarrow 0 \)
2. \( r_1 \leftarrow g, s_1 \leftarrow 0, t_1 \leftarrow 1 \)
3. \( i \leftarrow 1 \)
4. while \( (r_i \neq 0) \) do
5. \( q_i \leftarrow r_{i-1} \) quo \( r_i \)
6. \( r_{i+1} \leftarrow r_{i-1} - q_i r_i \)
7. \( s_{i+1} \leftarrow s_{i-1} - q_i s_i \)
8. \( t_{i+1} \leftarrow t_{i-1} - q_i t_i \)
9. \( i \leftarrow i + 1 \)
10. \( l \leftarrow i - 1 \)
11. return \( l \in \mathbb{N}, r_i, s_i, t_i \in R \) for \( 0 \leq i \leq l + 1 \) and \( q_i \in R \) for \( 1 \leq i \leq l \)
Algorithm terminates because $d(r_i)$ strictly decreasing.

- Elements $r_i$ for $0 \leq i \leq l + 1$ are remainders.
- Elements $q_i$ for $0 \leq i \leq l + 1$ are quotients.
- Central property: $s_i f + t_i g = r_i$ for all $i$.
- In particular: gcd of $f$ and $g$ is $r_l = s_l f + t_l g$. 
The Extended Euclidean Algorithm

Example: \( f = 126 \) and \( g = 35 \) with \( R = \mathbb{Z} \).

<table>
<thead>
<tr>
<th>( i )</th>
<th>( q_i )</th>
<th>( r_i )</th>
<th>( s_i )</th>
<th>( t_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>126</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>35</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>21</td>
<td>1</td>
<td>-3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>14</td>
<td>-1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>7</td>
<td>2</td>
<td>-7</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>0</td>
<td>-5</td>
<td>18</td>
</tr>
</tbody>
</table>

\[ \text{gcd}(126, 35) = 7 = 2 \cdot 126 + (-7) \cdot 35 \text{ (row 4)} \]
The Extended Euclidean Algorithm

Example: \( f = 18x^3 + 42x^2 + 30x - 6 \) and \( g = -12x^2 + 10x - 2 \) with \( R = \mathbb{Q}[x] \).

<table>
<thead>
<tr>
<th>( i )</th>
<th>( q_i )</th>
<th>( r_i )</th>
<th>( s_i )</th>
<th>( t_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>( 18x^3 - 42x^2 + 30x - 6 )</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>(-\frac{3}{2}x + \frac{9}{4})</td>
<td>(-12x^2 + 10x - 2)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>(-\frac{8}{3}x + \frac{4}{3})</td>
<td>(\frac{9}{2}x - \frac{3}{2})</td>
<td>1</td>
<td>(\frac{3}{2}x - \frac{9}{4})</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>(\frac{8}{3}x - \frac{4}{3})</td>
<td>4(x^2 - 8x + 4)</td>
<td></td>
</tr>
</tbody>
</table>

From row 2 we get:

\[
\frac{9}{2}x - \frac{3}{2} = 1 \cdot (18x^3 - 42x^2 + 30x - 6) + \left(\frac{3}{2}x - \frac{9}{4}\right) (-12x^2 + 20x - 2).
\]
Global view on extended gcd using matrices:

\[ R_0 = \begin{pmatrix} s_0 & t_0 \\ s_1 & t_1 \end{pmatrix} \quad Q_i = \begin{pmatrix} 0 & 1 \\ 1 & -q_i \end{pmatrix} \text{ for } 1 \leq i \leq l \]

in \( R^{2 \times 2} \), and \( R_i = Q_i \cdots Q_1 R_0 \) for \( 0 \leq i \leq l \).
Lemma (invariants of extended gcd)

For $0 \leq i \leq l$, with $r_{l+1} = 0$ (convention), we have

(i) $R_i \begin{pmatrix} f \\ g \end{pmatrix} = \begin{pmatrix} r_i \\ r_{i+1} \end{pmatrix}$,

(ii) $R_i = \begin{pmatrix} s_i & t_i \\ s_{i+1} & t_{i+1} \end{pmatrix}$,

(iii) $\gcd(f, g) \sim \gcd(r_i, r_{i+1}) \sim r_l$,

(iv) $s_i f + t_i g = r_i$ \hspace{1cm} (also holds for $i = l + 1$),

(v) $s_i t_{i+1} - t_i s_{i+1} = (-1)^i$,

(vi) $\gcd(r_i, t_i) \sim \gcd(f, t_i)$,

(vii) $f = (-1)^i(t_{i+1}r_i - t_i r_{i+1})$, \hspace{0.5cm} $g = (-1)^{i+1}(s_{i+1}r_i - s_i r_{i+1})$. 

Proof.

Induction over $i$ for (i) and (ii). $i = 0$ is clear from step 1 in the algorithm. We may assume for $i \geq 1$. Then

$$Q_i \begin{pmatrix} r_{i-1} \\ r_i \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -q_i \end{pmatrix} \begin{pmatrix} r_{i-1} \\ r_i \end{pmatrix} - \begin{pmatrix} r_i \\ r_{i-1} - q_i r_i \end{pmatrix} = \begin{pmatrix} r_i \\ r_{i+1} \end{pmatrix}$$

and (i) follows from $R_i = Q_i R_{i-1}$ and the induction hypothesis. Similarly, (ii) follows from

$$Q_i \begin{pmatrix} s_{i-1} & t_{i-1} \\ s_i & t_i \end{pmatrix} = \begin{pmatrix} s_i & t_i \\ s_{i+1} & t_{i+1} \end{pmatrix}$$

and the induction hypothesis.
The Extended Euclidean Algorithm

Proof.

For (iii), For $i \in \{0, \ldots, l\}$ we conclude from (i):

\[
\begin{pmatrix}
  r_i \\
  0
\end{pmatrix} = Q_l \cdots Q_{i+1} R_i \begin{pmatrix} f \\ g \end{pmatrix} = Q_l \cdots Q_{i+1} \begin{pmatrix} r_i \\ r_{i+1} \end{pmatrix}.
\]

We see that $r_l$ is linear combination of $r_i$ and $r_{i+1}$ (compare both sides). Therefore a common divisor of $r_i$ and $r_{i+1}$ will also divide $r_l$. On the other hand, $\det Q_i = -1$ and therefore $Q_i$ invertible over $R$:

\[
Q_i^{-1} = \begin{pmatrix} q_i & 1 \\ 1 & 0 \end{pmatrix}
\]

and hence

\[
\begin{pmatrix} r_i \\ r_{i+1} \end{pmatrix} = Q_{i+1}^{-1} \cdots Q_l^{-1} \begin{pmatrix} r_i \\ r_{i+1} \end{pmatrix}
\]

$r_l | r_i$ and $r_l | r_{i+1}$ and $r_l \sim \gcd(r_i, r_{i+1})$. (iii) follows from $i = 0$. 

[QED]
The Extended Euclidean Algorithm

Proof.

(iv) follows directly from (i) and (ii).

(v) follows from taking the determinants:

\[ s_i t_{i+1} - t_i s_{i+1} = \det \begin{pmatrix} s_i & t_i \\ s_{i+1} & t_{i+1} \end{pmatrix} = \det R_i \]

\[ = \det Q_i \cdots Q_1 \det \begin{pmatrix} s_0 & t_0 \\ s_1 & t_1 \end{pmatrix} = (-1)^i. \]

Corollary

Any two elements \( f, g \) of a Euclidean Domain \( R \) have \( \gcd \ h \in R \). \( h \) is expressible as linear combination \( h = sf + tg \), with \( s, t \in R \).
Cost analysis for $\mathbb{Z}$ and $F[x]$

- Cost of traditional Extended Euclidean Algorithm, $f, g \in R$ with $n = d(f) \geq d(g) = m \geq 0$.
- Number of division steps $l$ bounded by $l \leq d(g) + 1$.
- For $R = F[x]$ ($F$ is a field), as usual $d(a) = \deg(a)$.
- Let $n_i = \deg r_i$ for $0 \leq i \leq n + 1$ and $r_{l+1} = 0$.
- Then $n_0 = n \geq n_1 = m > n_2 > \cdots > n_l$ and degree for quotient $\deg q_i = n_{i-1} - n_i$ for $1 \leq i \leq l$.
- The degree of polynomials $r_i$ is decreasing.
We can divide with remainder \( r_{i-1} \) with \( \deg r_{i-1} = n_{i-1} \) by \( r_i \) with \( \deg r_i = n_i \) and \( n_{i-1} \geq n_i \) using at most

\[
(2n_i + 1)(n_{i-1} - n_i + 1)
\]

additions and multiplications plus one inversion in \( F \) (part 2).

Cost for computing only \( r_i \) and \( q_i \), including \( \gcd(f, g) \)

\[
\sum_{1 \leq i \leq l} (2n_i + 1)(n_{i-1} - n_i + 1)
\]

additions and multiplications and \( l \leq m + 1 \) inversions in \( F \).
Cost analysis for \( \mathbb{Z} \) and \( F[x] \)

- Evaluate expression for the normal case, where degree drops by exactly 1.

- \( n_i = m - i + 1 \) for \( 2 \leq i \leq l = m + 1 \) (worst case). Therefore \( n_1 = m \) and \( n_{i-1} - n_i + 1 = 2 \), for \( i \geq 2 \). We get

\[
(2m + 1)(n - m + 1) + 2 \sum_{2 \leq i \leq m+1} (2(m - i + 1) + 1) = (2m + 1)(n - m + 1) + 2(m^2 - m) + 2m = 2nm + n + m + 1.
\]

- Bound also holds for any case (details see book).
Cost analysis for $\mathbb{Z}$ and $F[x]$

- Determine cost for computing $s_i, t_i$.

**Lemma**

\[(i)\] \[\deg s_i = \sum_{2 \leq j < i} \deg q_j = n_1 - n_{i-1} \text{ for } 2 \leq i \leq l + 1\]

\[(ii)\] \[\deg t_i = \sum_{1 \leq j < i} \deg q_j = n_0 - n_{i-1} \text{ for } 1 \leq i \leq l + 1\]

- Show \((i)\) and $\deg s_{i-1} < \deg s_i$ for $2 \leq i \leq l + 1$ by induction.
Proof.

\( i = 2 \): We find \( s_2 = s_0 - q_1 s_1 = 1 - q_1 \cdot 0 = 1 \) and 
\[ \deg s_1 = -\infty < 0 = \deg s_2. \]

\( i \geq 2 \): By induction hypothesis:
\[ \deg s_{i-1} < \deg s_i < n_{i-1} - n_i + \deg s_i = \deg(q_i s_i) \]
which implies
\[ \deg s_{i+1} = \deg(s_{i-1} - q_i s_i) = \deg q_i + \deg s_i > \deg s_i, \]
and
\[ \deg s_{i+1} = \deg q_i + \deg s_i = \sum_{2 \leq j < i} \deg q_j + \deg q_i = \sum_{2 \leq j < i+1} \deg q_j \]
The traditional Extended Euclidean Algorithm for polynomials $f, g \in F[x]$ with $\deg f = n \geq \deg g = m$ can be performed with

- at most $m + 1$ inversions and $2nm + O(n)$ additions and multiplications in $F$, if only the quotient $q_i$ and the remainders $r_i$ are needed.
- at most $m + 1$ inversions and $6nm + O(n)$ additions and multiplications in $F$ for computing all results.

Later we will introduce (much) faster algorithms for gcd.
For $R = \mathbb{Z}$ we use $d(a) = |a|$.

We assume $f = r_0 \geq g = r_1 > r_2 > \cdots > r_l \geq 0$, s.t. $q_i \geq 1$ for all $i$.

Length $\lambda(a) = \lfloor \frac{\log(a)}{64} \rfloor + 1$.

Bound for division steps (from polynomial case)

$$l \leq d(g) + 1 = g + 1 = (2^{64})^{(\log g)/64} + 1 \leq 2^{64} \lambda(g)$$

for the pair $(f, g) \in \mathbb{N}^2$ is exponential in the input size $\lambda(f) + \lambda(g)$.
Cost analysis for $\mathbb{Z}$ and $F[x]$

For $1 \leq i \leq l$ we have

$$r_{i-1} = q_ir_i + r_{i+1} \geq r_i + r_{i+1} > 2r_{i+1}$$

Thus

$$\prod_{2 \leq i < l} r_{i-1} > 2^{l-2} \prod_{2 \leq i < l} r_{i+1}$$

if $l \geq 2$, and $r_{l-1} \geq 2$ implies that

$$2^{l-2} < \frac{r_1r_2}{r_{l-1}r_l} < \frac{r_1^2}{2},$$

$$l \leq \lfloor 2 \log r_1 \rfloor + 1 = \left\lfloor 128 \frac{\log g}{64} \right\rfloor \leq 128 \left( \left\lfloor \frac{\log g}{64} \right\rfloor + 1 \right)$$

$$= 128 \lambda(g).$$
Cost analysis for $\mathbb{Z}$ and $F[x]$

- Using Fibonacci numbers we can improve the bound to

$$l \approx \frac{12(\ln 2)^2}{\pi^2} \log g \approx 0.584 \log g$$

**Lemma**

$$|s_i| \leq \frac{g}{r_{i-1}} \text{ and } |t_i| \leq \frac{f}{r_{i-1}} \text{ for } 1 \leq i \leq l + 1.$$  

**Proof.**

Idea: Analog bound for length of $s_i$ and $t_i$ as in the polynomial case.
Cost analysis for \( \mathbb{Z} \) and \( F[x] \)

**Theorem**

*The traditional Extended Euclidean Algorithm for positive integers \( f, g \) with \( \lambda(f) = n \geq \lambda(g) = m \) can be performed with \( O(nm) \) word operations.*

Probability for two random numbers below \( N \) to be coprime,

\[
\frac{C_N}{N^2} \approx 0.6079271016 + O \left( \frac{\log N}{N} \right).
\]

<table>
<thead>
<tr>
<th>( N )</th>
<th>( \frac{C_N}{N^2} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.63</td>
</tr>
<tr>
<td>100</td>
<td>0.6087</td>
</tr>
<tr>
<td>1000</td>
<td>0.608383</td>
</tr>
<tr>
<td>10000</td>
<td>0.60794971</td>
</tr>
<tr>
<td>100000</td>
<td>0.6079301507</td>
</tr>
</tbody>
</table>
(Non-) Uniqueness of the gcd

- non-uniqueness of gcd problematic for implementations.
- In \( \mathbb{Q} \) every \( a \neq 0 \) is a unit.
- Therefore \( ua \sim a \) in \( R = \mathbb{Q}[x] \) for all \( u \in \mathbb{Q} \) and \( a \in R \).
- Which element to choose for \( \gcd(f, g) \in \mathbb{Q}[x] \)?
- Which multiple of \( a \) to choose?
Example: Euclidean algorithm for 126 and 35.

\[
\begin{align*}
126 &= 3 \cdot 35 + 21 \\
35 &= 1 \cdot 21 + 14 \\
21 &= 1 \cdot 14 + 7 \\
14 &= 2 \cdot 7
\end{align*}
\]

In \( \mathbb{Z} \) the greatest common divisor of 126 and 35, 
\( \gcd(126, 35) = 7 \)

In \( \mathbb{Q} \) every \( u \cdot 7 \) is a common divisor of 126 and 35, with nonzero \( u \in \mathbb{Q} \).

8 is a common divisor: \( 126 = 8 \cdot \frac{18 \cdot 7}{8} \) and \( 35 = 8 \cdot \frac{5 \cdot 7}{8} \).
For $R = \mathbb{Q}[x]$ the monic polynomial (leading coefficient is 1) is a good choice.

- $\text{lc}(a) \in \mathbb{Q} \setminus \{0\}$ leading coefficient of $a \in \mathbb{Q}[x]$.
- We use $\text{normal}(a) = \frac{a}{\text{lc}(a)}$ as the normal form of $a$.
- Select normal form for Euclidean domain $R$, s.t. $\text{normal}(a) \sim a$ for all $a \in R$.
- We call unit $u \in R$ with $a = u \cdot \text{normal}(a)$ the leading unit $\text{lu}(a)$ of $a$. 
(Non-) Uniqueness of the gcd

Two properties are required:

- Two elements of $R$ have the same normal form, iff they are associate. With $a, b, u \in R$ and $u$ is a unit:
  \[
  \text{normal}(a) = \text{normal}(b) \iff a \sim b \iff a = u \cdot b
  \]

- The normal form of a product is equal to the products of the normal forms.
  \[
  \text{normal}(a \cdot b) = \text{normal}(a) \cdot \text{normal}(b)
  \]

We say that $a$ in normal form, s.t. $1u(a) = 1$, is normalized.
Two main applications are integers and univariate polynomials over a field.

For $R = \mathbb{Z}$, we define $\text{lu}(a) = \text{sign}(a)$, if $a \neq 0$ and $\text{normal}(a) = |a|$ (integers are normalized if nonnegative).

For $R = F[x]$, we define $\text{lu}(a) = \text{lc}(a)$ (convention $\text{lu}(0) = 1$) and $\text{normal}(a) = \frac{a}{\text{lc}(a)}$ (monic polynomials are normalized).

gcd then is the unique normalized associate of all greatest common divisors of $a$ and $b$.

Similarly, lcm is normalized associate of all least common multiples of $a$ and $b$. 
Algorithm (Extended Euclidean algorithm)

**Input:** $f, g \in R$ is a Euclidean Domain with normal form.

**Output:** $l \in \mathbb{N}, p_i, r_i, s_i, t_i \in R$ for $0 \leq i \leq l + 1$ and $q_i \in R$ for $1 \leq i \leq l$

1. $p_0 \leftarrow \text{lu}(f)$, $r_0 \leftarrow \text{normal}(f)$, $s_0 \leftarrow p_0^{-1}$, $t_0 \leftarrow 0$
2. $p_1 \leftarrow \text{lu}(g)$, $r_1 \leftarrow \text{normal}(g)$, $s_1 \leftarrow 0$, $t_1 \leftarrow p_1^{-1}$
3. $i \leftarrow 1$
4. while ($r_i \neq 0$) do
5.   $q_i \leftarrow r_{i-1} \text{ quo } r_i$
6.   $p_{i+1} \leftarrow \text{lu}(r_{i-1} - q_i r_i)$
7.   $r_{i+1} \leftarrow (r_{i-1} - q_i r_i) / p_{i+1}$
8.   $s_{i+1} \leftarrow (s_{i-1} - q_i s_i) / p_{i+1}$
9.   $t_{i+1} \leftarrow (t_{i-1} - q_i t_i) / p_{i+1}$
10. $i \leftarrow i + 1$
11. $l \leftarrow i - 1$
12. return $l \in \mathbb{N}, p_i, r_i, s_i, t_i \in R$ for $0 \leq i \leq l + 1$ and $q_i \in R$ for $1 \leq i \leq l$
(Non-) Uniqueness of the gcd

- $r_i$ are the remainders for $1 \leq i \leq l + 1$.
- $q_i$ are the quotients, for $1 \leq i \leq l$.
- Elements $s_l, t_l$, that satisfy $s_l f + t_l g = \gcd(f, g)$, are called Bézout coefficients of $f$ and $g$.
- Matrices $Q_i$ now become:

$$Q_i = \begin{pmatrix} 0 & 1 \\ p_{i+1}^{-1} & -q_ip_{i+1}^{-1} \end{pmatrix} \text{ for } 1 \leq i \leq l$$

and $R_i = Q_i \cdots Q_1 R_0$ for $0 \leq i \leq l$. 
(Non-) Uniqueness of the gcd

Example:

\[ f = 18x^3 + 42x^2 + 30x - 6, \quad \text{normal}(f) = x^3 + \frac{7}{3}x^2 + \frac{5}{3}x - \frac{1}{3} \]

and \( g = -12x^2 + 10x - 2, \quad \text{normal}(g) = x^2 - \frac{5}{6}x + \frac{1}{6} \) in \( \mathbb{Q}[x] \).

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
i & q_i & p_i & r_i & s_i & t_i \\
\hline
0 & 18 & & & \frac{1}{18} & 0 \\
1 & x - \frac{3}{2} & -12 & x^3 - \frac{7}{3}x^2 + \frac{5}{3}x - \frac{1}{3} & & - \frac{1}{12} \\
2 & x - \frac{1}{2} & 1 & x^2 - \frac{5}{6}x + \frac{1}{6} & \frac{2}{9} & \frac{1}{3}x - \frac{1}{2} \\
3 & & & x - \frac{1}{3} & & - \frac{1}{3}x^2 + \frac{2}{3}x - \frac{1}{3} \\
\hline
\end{array}
\]

From row 2 we get:

\[
\gcd(f, g) = x - \frac{1}{3} = \frac{2}{9} \cdot (18x^3 - 42x^2 + 30x - 6) + \left(\frac{1}{3}x - \frac{1}{2}\right) (-12x^2 + 20x - 2).
\]
(Non-) Uniqueness of the gcd

Lemma (invariants for EEA - updated)

For $0 \leq i \leq l$, with $r_{l+1} = 0$ (convention), we have

(iii) $\gcd(f, g) = \gcd(r_i, r_{i+1}) = r_l,$

(v) $s_it_{i+1} - t_is_{i+1} = (-1)^i(p_0 \cdots p_{i+1})^{-1},$

(vi) $\gcd(r_i, t_i) = \gcd(f, t_i),$

(vii) $f = (-1)^i p_0 \cdots p_{i+1}(t_{i+1}r_i - t_ir_{i+1}),$

$g = (-1)^{i+1} p_0 \cdots p_{i+1}(s_{i+1}r_i - s_ir_{i+1}).$
Proof.

Use proof for earlier invariants lemma with following changes:

\[
Q_i \begin{pmatrix} r_{i-1} \\ r_i \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ p_{i+1}^{-1} & -q_ip_{i+1}^{-1} \end{pmatrix} \begin{pmatrix} r_{i-1} \\ r_i \end{pmatrix} = \begin{pmatrix} r_i \\ (r_{i-1} - q_ir_i)p_{i+1}^{-1} \end{pmatrix} = \begin{pmatrix} r_i \\ r_{i+1} \end{pmatrix},
\]

\[
Q_i^{-1} = \begin{pmatrix} q_i & p_{i+1} \\ 1 & 0 \end{pmatrix},
\]
(Non-) Uniqueness of the gcd

Proof.

\[
\det Q_i \cdots \det Q_1 \cdot \det \begin{pmatrix} s_0 & t_0 \\ s_1 & t_1 \end{pmatrix} = (-1)^i (p_0 \cdots p_{i+1})^{-1},
\]

\[
\begin{pmatrix} r_0 \\ r_1 \end{pmatrix} = R_i^{-1} \begin{pmatrix} r_i \\ r_{i+1} \end{pmatrix} = (-1)^i (p_0 \cdots p_{i+1}) \begin{pmatrix} t_{i+1} & -t_i \\ -s_{i+1} & s_i \end{pmatrix} \begin{pmatrix} r_i \\ r_{i+1} \end{pmatrix}.
\]

The statements (iii) and (vi) follow from the fact that all involved elements are normalized.
(Non-) Uniqueness of the gcd

Degree formulas (cost analysis) hold for updated EEA, if $R = F[x]$ for a field $F$, $\deg f \geq \deg g$, and $n_i = \deg r_i$ for all $i$.

**Theorem**

*For the monic normal form $\text{normal}(h) = h/\text{lc}(h)$ on $F[x]$, the Extended Euclidean Algorithm for polynomials $f, g \in F[x]$ with $\deg f = n \geq \deg g = m$ can be performed with*

- at most $m + 2$ inversions and $2nm + O(n)$ additions and multiplications in $F$, if only the quotient $q_i$, the remainders $r_i$ and the coefficients $p_i$ are needed.

- at most $m + 2$ inversions and $6nm + O(n)$ additions and multiplications in $F$ for computing all results.
Normalization is often not compatible across domains.

For computer algebra software other rings besides \( \mathbb{Z} \) or \( \mathbb{F}[x] \) might be relevant.

\( R \) is a domain with normal form \( \text{normal}_R \), then for polynomial ring \( R[x] \) we set

\[
\text{normal}_R[f] = \frac{\text{normal}_R[\text{lc}(f)]}{\text{lc}(f)} \cdot f.
\]

with \( \text{lc}(f) \) the leading coefficient of \( f \).