Thm 3.2 (Schönhage, using ideas of Lehmer, Knuth)
Let $n \geq k \geq 1$. For polynomials $f, g \in K(x)$ of degree $\leq n$, you can find a matrix $M \in GL_2^{\pm 1}(K(x))$ whose entries have

$\uparrow$ $\{M \in GL_2 : \det(M) = \pm 1\}$

degree $\leq k$ and such that $M\begin{pmatrix} f \\ g \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix}$ for some pol. $a, b \in K(x)$ with $\deg(b) \leq n-k-1$,
in time $O(n + \mu(k)\log k)$ (for large k) on an $O(\log n)$-bit RAM.

Cor 3.3 (Fast extended Euclidean algorithm)
You can find the gcd of polynomials $f, g \in K(x)$ of degree $\leq n$, in time $O(\mu(n)\log n)$ (for large n).
(and pol. c, d s.t. $a = cf + dg$)

Pf of Cor Apply the Thm with $k = n$.
Then, $M\begin{pmatrix} f \\ g \end{pmatrix} = \begin{pmatrix} a \\ 0 \end{pmatrix}$, so $\gcd(f, g) = a$.

computable in $O(\mu(n))$

If $M = \begin{pmatrix} c & d \\ * & * \end{pmatrix}$, then $a = cf + dg$. □

Cor 3.4 If $\gcd(f, g) = 1$, you can find the inverse of g mod f in time $O(\mu(n)\log n)$. ...

Pf of Cor $dg \equiv a \mod f$.

$\uparrow$ constant pol. $\neq 0$

□

Alg for Thm 3.2 (Strassen)

W.l.o.g. $k \le n$. (Otherwise, replace k by n.)

If $n = k = 0$, (so f, g are constant polynomials) it's easy: Take $M = \begin{pmatrix} 0 & 1 \\ 1 & -f/g \end{pmatrix}$.

If $n > 2k$, we can, according to Lemma 3.1, replace n by $2k$, f by $\lfloor f/x^{n-2k} \rfloor$, g by $\lfloor g/x^{n-2k} \rfloor$.

Assume $1 \le k \le n \le 2k$. Let $k' = \lfloor \frac{k}{2} \rfloor$.

1) Recursively apply the alg. to find M_1 s.t.

$M_1 \begin{pmatrix} f \\ g \end{pmatrix} = \begin{pmatrix} c \\ d \end{pmatrix}$ with $\deg(d) \le n - k' - 1$.

(computable in $\mathcal{O}(k)$)

Time $\le C \cdot \mu(\frac{k}{2}) \log_2(\frac{k}{2}) \le \frac{C}{2} \mu(k)(\log_2(k) - 1)$

2) If $d = 0$, we're done. Otherwise:

Let $M_2 = \begin{pmatrix} 0 & 1 \\ 1 & -\lfloor c/d \rfloor \end{pmatrix}$. $\Rightarrow M_2 M_1 \begin{pmatrix} f \\ g \end{pmatrix} = M_2 \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} d \\ c \bmod d \end{pmatrix}$

Both have $\deg \le n - k' - 1$.

Time $\mathcal{O}(\mu(k))$

3) Recursively apply the alg. to find M_3 s.t.

$M_3 M_2 M_1 \begin{pmatrix} f \\ g \end{pmatrix} = M_3 \begin{pmatrix} d \\ c \bmod d \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix}$ with $\deg(b) \le (n - k' - 1) - k' - 1$

$\le n - k - 1$.

Time $\le \frac{C^*}{2} \cdot \mu(k)(\log_2(k) - 1)$

Total: $C \mu(k)(\log_2 k - 1) + \mathcal{O}(\mu(k)) \le C \mu(k) \log_2 k$ for suff. large C. $\square$

Rmk The algorithm also computes the quotient polynomials q_i that arise in the Euclidean alg. s.t.

$$M = \begin{pmatrix} 0 & 1 \\ 1 & q_r \end{pmatrix} \cdots \begin{pmatrix} 0 & 1 \\ 1 & q_1 \end{pmatrix}.$$

$$\frac{f}{g} = q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 \ddots \frac{}{q_r}}}$$

Thm 3.5 (Knuth–Schönhage)

For binary integers x, y with $\leq n$ bits, we can compute $a = \gcd(x,y)$, and c, d s.t. $a = cx + dy$ in time $O(n \log n)$ on an $O(\log n)$-bit RAM.

Rk 1 Similar to polynomials, replacing $\deg(f)$ by $\log|f|$.

It is more complicated:

Polynomials satisfy the nice ineq. $\deg(f+g) \leq \max(\deg(f), \deg(g))$, (nonarch. triangle ineq.)

whereas, we only have $\log|f+g| \leq \log|f| + \log|g| + \log 2$. (arch. triangle ineq.)

$\log|x+y|$ can be slightly larger than $\max(\log|x|, \log|y|)$

→ Lemma 3.1 fails "slightly".

But you can carefully deal with it! "□"

Rk 2 (Stehlé–Zimmerman: A Binary Recursive GCD algorithm)

Idea: Instead of the usual division with remainder ($x = qy + r$ with $q, r \in \mathbb{Z}$, $0 \leq r < |y|$)

use generalised binary division

($x = qy + r$ with $r \in \mathbb{Z}$, $v_2(r) > v_2(y)$, $q \in \mathbb{Q}$, $|q| < 1$, denominator of q is a power of 2)

where $v_2(x) < v_2(y)$

$v_2(r)$: nr. of times r is divisible by 2 (∞ if $r = 0$)

The Euclidean algorithm still terminates with this division.

after $O(n)$ steps

There is something analogous to Lemma 3.1 that can be used to speed up the algorithm very similar to Thm 3.2. "□"