Chapter 11 Fast Euclidean Algorithm

11.1 A fast Euclidean Algorithm for polynomials

Let $f = f_n x^n + f_{n-1}x^{n-1} + \cdots + f_0 \in F[x]$ with leading coefficient $f_n \neq 0$, and $k \in \mathbb{Z}$. We define the truncated polynomial

$$f \upharpoonright k = f_n x^k + f_{n-1}x^{k-1} + \cdots + f_{n-k}$$

Let $f, g, f^*, g^* \in F[x] \backslash \{0\}$ with $\deg f \geq \deg$ and $\deg f^* \geq \deg g^*$, and $k \in \mathbb{Z}$. Then (f, g) and (f^*, g^*) coincide up to k if

$$\begin{aligned} f \upharpoonright k &= f^* \upharpoonright k, \\ g \upharpoonright (k - (\deg f - \deg g)) &= g^* \upharpoonright (k - (\deg f^* - \deg g^*)) \end{aligned}$$

Lemma 1. *Let* $k \in \mathbb{Z}, (f, g)$ *and* $(f^*, g^*) \in (F[x] \backslash \{0\})^2$ *coincide up to* $2k$ *and* $k \geq \deg f - \deg g \geq 0$*, Define* $q, r, q^*, r^* \in F[x]$ *by division with remainder:*

$$\begin{aligned} f &= qg + r, \quad \deg r < \deg g, \\ f^* &= q^*g^* + r^*, \quad \deg r^* < \deg g^* \end{aligned}$$

Then $q = q^*$*, and either* (g, r) *and* (g^*, r^*) *coincide up to* $2(k - \deg q)$ *or* $r = 0$ *or* $k - \deg q < \deg g - \deg r$*.*

See example 11.2.

Consider the Euclidean Algorithm for two pairs r_0, r_1 and r_0^*, r_1^* of monic polynomials with $\deg r_0 > \deg r1$ and $\deg r_0^* > \deg r_1^*$:

$$
\begin{array}{rcccrcc}
r_0 & = & q_1 r_1 + \rho_2 r_2 & & r_0^* & = & q_1^* r_1^* + \rho_2^* r_2^*, \\
& \vdots & & & & \vdots & \\
r_{i-1} & = & q_i r_i + \rho_{i+1} r_{i+1} & & r_{i-1}^* & = & q_i^* r_i^* + \rho_{i+1}^* r_{i+1}^*, \\
& \vdots & & & & \vdots & \\
& \vdots & & & r_{l^*-1}^* & = & q_{l^*}^* r_{l^*}^* \\
r_{l-1} & = & q_l r_l & & & &
\end{array}
$$

Let $m_i = \deg q_i, \quad n_i = \deg r_i, \quad m_i^* = \deg q_i^*, n_i^* = \deg r_i^*,$

$$\eta(k) = \max\{0 \leq j \leq l : \sum_{1 \leq i \leq j} m_i \leq k\}$$

So that

$$n_0 - n_{\eta(k)} = \sum_{1 \le i \le \eta(k)} m_i \le k < \sum_{1 \le i \le \eta(k)+1} m_i = n_0 - n_{\eta(k)+1}$$

Divide-and-conquer algorithm based on dividing the sequence of the quotients into two parts such that the sum of the degrees in the first part is about the same as in the second part.

Lemma 2. *Let $k \in \mathbb{N}, h = \eta(k)$, and $h^* = \eta^*(k)$. If (r_0, r_1) and (r_0^*, r_1^*) coincide up to $2k$, then $h = h^*, q_i = q_i^*$ and $\rho_{i+1} = \rho_{i+1}^*$ for $1 \le i \le h$*

$$R_0 = \begin{pmatrix} s_0 & t_0 \\ s_1 & t_1 \end{pmatrix}, \quad R_i = \begin{pmatrix} s_i & t_i \\ s_{i+1} & t_{i+1} \end{pmatrix}, \quad R_i \cdot \begin{pmatrix} r_0 \\ r_1 \end{pmatrix} = \begin{pmatrix} r_i \\ r_{i+1} \end{pmatrix},$$

$$Q_i = \begin{pmatrix} 0 & 1 \\ \rho_{i+1}^{-1} & -q_i\rho_{i+1}^{-1} \end{pmatrix}, \; Q_i \begin{pmatrix} r_{i-1} & s_{i-1} & t_{i-1} \\ r_i & s_i & t_i \end{pmatrix} = \begin{pmatrix} r_i & s_i & t_i \\ r_{i+1} & s_{i+1} & t_{i+1} \end{pmatrix}$$

$$R_i = Q_i \cdots Q_1 R_0,$$

,

Algorithm 11.4 Fast Extended Euclidean Algorithm

Input: $r_0, r_1 \in F[x]$ normalized, $n_0 = \deg r_0 > n_1 = \deg r_1$, $k \in \mathbb{N}$ with $0 \leq k \leq n$.

Output: $h = \eta(k) \in \mathbb{N}$, R_h

1. if $r_1 = 0$ or $k < n_0 - n_1$ then return 0 and $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

2. $d \longleftarrow \lfloor k/2 \rfloor$

3. call the algorithm recursively with input $r_0 \upharpoonright 2d, r_1 \upharpoonright (2d - (n_0 - n_1))$ and d, returning $j - 1 = \eta(d)$ and $R = Q_{j-1} \cdots Q_1$

4. $\begin{pmatrix} r_{j-1} \\ r_j \end{pmatrix} \longleftarrow R \begin{pmatrix} r_0 \\ r_1 \end{pmatrix}$, $\begin{pmatrix} n_{j-1} \\ n_j \end{pmatrix} \longleftarrow \begin{pmatrix} \deg r_{j-1} \\ \deg r_j \end{pmatrix}$

5. if $r_j = 0$ or $k < n_0 - n_j$ then return $j - 1$ and R

6. $q_j \longleftarrow r_{j-1}$ quo $r_j, \rho_{j=1} \longleftarrow$ lc$(r_{j-1}$ rem $r_j)$,
$r_{j+1} \longleftarrow$ normal$(r_{j-1}$ rem $r_j), n_{j+1} \longleftarrow \deg r_{j+1}$

7. $d^* \longleftarrow k - (n_0 - n_j)$

8. call the algorithm recursively with input $r_j \restriction 2d^*, r_{j+1} \restriction (2d^* - (n_j - n_{j+1}))$ and d^*, returning $h - j = \eta(d^*)$ and $S = Q_h \ldots Q_{j+1}$

9. $Q_j \longleftarrow \begin{pmatrix} 0 & 1 \\ \rho_{j+1}^{-1} & -q_j \rho_{j+1}^{-1} \end{pmatrix}$ return h and SQ_jR

Continued of example 11.2

Theorem 1. *Algorithm 11.4 works correctly and uses $O(M(k)\log k)$ operations in F if $n_0 = n \leq 2k$.*

Corollary 1. *For polynomials $f, g \in F[x]$ of degree at most n, each of the following can be computed with $O(M(n)\log n)$ operations.*

1. $r_l = \gcd(f, g)$

2. $s, t \in F[x]$ *with* $sf + tg = r_l$

11.2 Subresultants via Euclid's Algorithm

We write $S_k(f,g)$ for the k-th submatrix of $\mathrm{Syl(f,g)}$, and $\sigma_k(f,g) = \det S_k(f,g)$.

Lemma 3. *Let $f, g, r \in F[x]$ be monic polynomials with degrees n, m, d respectively, $n \geq m, \rho r = f$ rem g with a nonzero $\rho \in F$, and $0 \leq k \leq d$. Then*

$$\sigma_k(f,g) = (-1)^{(n-k)(m-k)} \rho^{m-k} \sigma_k(g,r)$$

Theorem 2. *Fundamental Theorem on Subresultants* *Let $f = \rho_0 r_0, g = \rho_1 r_1 \in F[x]$ of degree $n \geq m$ respectively, with nonzero $\rho_0, \rho_1 \in F$ and monic r_0, r_1, n_i and ρ_i the degrees and "leading coefficients" from the Euclidean Algorithm 3.6 for $0 \leq i \leq l$.*

1. *For $0 \leq k \leq m$, the k-th subresultant of (f, g) is*

$$\sigma_k = \det S_k = \begin{cases} (-1)^{\tau_i} \rho_0^{m-n_i} \prod_{1 \leq j \leq i} \rho_j^{n_{j-1}-n_i} & k = n_i \text{ for some } i \leq l \\ 0 & \text{otherwise} \end{cases}$$

where $\tau_i = \sum_{1 \leq j < i} (n_{j-1} - n_i)(n_j - n_i)$.

2. *The subresultants satisfy for $1 \leq i < l$ the recursive formulas*

$$\sigma_m = \rho_1^{n-m}, \sigma_{n_{i+1}} = (-1)^{(n_i - n_{i+1})(n - n_{i+1} + i + 1)} (\rho_0 \cdots \rho_{i+1})^{n_i - n_{i+1}} \sigma_{n_i}$$

Corollary 2. *Let $f, g \in F[x]$ have degrees $n = n_0 \geq n_1$, and $0 \leq k \leq n_1$. Then all subresultants σ_j for $n_l \geq j \geq (n-k)$ of (f, g) can be calculated with at most $(24M(k) + O(k)) \log k$ operations in F.*

Corollary 3. *Let $f, g \in F[x]$ have degrees $n = n_0 \geq n_1$, let $n_2, \ldots, n_l$ be the degrees of the remainders in the Euclidean Algorithm for (f, g), and $\rho_0, \ldots, \rho_l$ the leading coefficients. If $\deg \gcd(f, g) \leq 1$, then res$(f, g) = 0$. Otherwise,*

$$\textit{res}(f, g) = (-)^{\tau} \rho_0^{n_1} \prod_{1 \leq j \leq l} \rho_j^{n_{j-1}}$$

where $\tau = \sum_{1 \leq j < l} n_{j-1} n_j$. This resultant can be calculated using no more than $(24M(n) + O(n)) \log n$ operations in F.

See example for application in gcd computation.