1 Applications of the Extended Euclidean Algorithm (EEA)

1.1 Computing the inverse

1. Let $n$ be an integer, and $0 \leq a < n$ be such that $\gcd(a, n) = 1$. Give an algorithm that computes $a^{-1} \mod n$ in time $O(M(\log n) \log \log n)$.

2. Let $P \in K[X]$ be a polynomial of degree $d$ with coefficients in a field $K$ and $Q \in K[X]$ be a polynomial of degree less than $d$, such that $\gcd(P, Q) = 1$. Prove that $Q$ is invertible modulo $P$ and give an algorithm to compute its inverse using $O(M(d) \log d)$ operations in $K$.

1.2 Diophantine equation

The aim of this exercise is to describe the set of all integer solutions $(u, v)$ of the equation

$$au + bv = t \quad (1)$$

1. Show that if $(u, v) = (s_1, s_2)$ is a solution of (1), the general solution is of the form $(u, v) = (s_1 + s'_1, s_2 + s'_2)$ for $(s'_1, s'_2)$ satisfying $as'_1 + bs'_2 = 0$.

2. Find all solutions of $au + bv = 0$ for $a, b$ coprime.

3. Find a solution of (1) for $a, b$ coprime.

4. Observe that $t$ must be divisible by $\gcd(a, b)$.

5. Using the previous questions, give the general solution of (1).

2 Rational function reconstruction

Let $K$ be a field, $m \in K[X]$ of degree $n > 0$, and $f \in K[X]$ such that $\deg f < n$. For a fixed $k \in \{1, \ldots, n\}$, we want to find a pair of polynomials $(r, t) \in K[X]^2$, satisfying

$$r = t \cdot f \mod m, \quad \deg r < k, \quad \deg t \leq n - k \quad \text{and} \quad t \neq 0 \quad (2)$$

1. Consider $A(X) = \sum_{t=0}^{N-1} a_t X^t \in K[X]$ a polynomial. Show that if $A(X) = P(X)/Q(X)$ mod $X^N$, where $P, Q \in K[X]$, $Q(0) = 1$ and $\deg P < \deg Q$, then the coefficients of $A$, starting from $a_{\deg Q}$ can be computed as a linear recurrent sequence of previous $\deg Q$ coefficients of $A$. What can you say in the converse setting when the coefficients of $A$ satisfy a linear recurrence relation?
2. Inside [2], consider the case when \( m = x^n \). Describe a linear algebra-based method for finding some \( t \) and \( r \). (Suggestion: do not use the previous question).

3. Show that, if \((r_1, t_1)\) and \((r_2, t_2)\) are two pairs of polynomials that satisfy [2], then we have \( r_1 t_2 = r_2 t_1 \).

We will use the Extended Euclidean Algorithm to solve problem [2].

4. Let \( r_j, u_j, v_j \in F[X] \) be the quantities computed during the \( j \)-th pass of the Extended Euclidean Algorithm for the pair \((m, f)\), where \( j \) is minimal such that \( \deg r_j < k \). Show that \((r_j, v_j)\) satisfy (2). What can you say about the complexity of this method?

5. Application. Given \( 2n \) consecutive terms of a recursive sequence of order \( n \), give the recurrence. (Hint: this is where you use question 1). Illustrate your method on the Fibonacci sequence.

3 Fast polynomial gcd

Let \( a \) and \( b \) be polynomials in \( K[x] \), \( \deg(a) = n \) and \( \deg(b) = n - 1 \). The goal of this exercise is to develop an algorithm that computes \( \gcd(a, b) \) in time \( O(M(n) \log^{2} n) \). Let \((r_i)_i \in K[x] \) be the sequence of remainders produced by Extended Euclidean Algorithm (EEA), and \((q_i)_i \) - the sequence of quotients, i.e.,

\[
r_{i-1} = q_i r_i + r_{i+1}, \quad \text{with} \quad r_0 = a, r_1 = b, r_N = \gcd(a, b).
\]

We shall assume that \( \deg(r_i) = \deg(r_{i-1}) - 1 \) for all \( i \). This is merely to simplify notations, the idea works in general.

1. Re-write the EEA algorithm as a sequence of \( 2 \times 2 \) matrix-vector multiplications of the form 

\[
M_i \cdot \begin{bmatrix} r_{i-1} \\ r_i \end{bmatrix}.
\]

Give an explicit form of \( M_i \)'s.

2. We will first design a divide-and-conquer algorithm that gives the last term in the remainder sequence whose degree is more than \( \deg(a)/2 \), i.e, \( r_{[\deg(a)/2]} \).

The algorithm relies on the idea that the quotient of two polynomials of degrees \( d_1 \) resp. \( d_2 \) depends only on the leading \( \min\{d_1 - d_2 + 1, d_2\} \) terms of the divisor and the leading \( d_1 - d_2 + 1 \) terms of the dividend. More formally, consider two polynomials

\[
a(x) = a_1(x)x^k + a_2(x) \\
b(x) = b_1(x)x^k + b_2(x),
\]

where \( \deg(a_2) < k \) and \( \deg(b_2) < k \). Let

\[
a(x) = q(x)b(x) + r(x) \\
a_1(x) = q_1(x)b_1(x) + r_1(x),
\]

where \( \deg(r) < \deg(b) \) and \( \deg(r_1) < \deg(b_1) \). Show that if \( \deg(b_1) \geq 1/2 \deg(a_1(x)) \), which implies that \( k \leq 2 \deg(b(x)) - \deg(a(x)) = n - 2 \), then

1. \( q(x) = q_1(x) \)
2. \( r(x) \) and \( r_1(x)x^k \) agree in all terms of degree \( k+1 \) or higher.

3. Using the notation

\[
M_{i,j}^{a,b} = \begin{cases}
1, & i = j; \\
\begin{bmatrix} 0 & 1 \\ 1 & -q_i \\ \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 1 & -q_{j-1} \\ \end{bmatrix} \cdot \ldots \cdot \begin{bmatrix} 0 & 1 \\ 1 & -q_{i+1} \\ \end{bmatrix} & i < j,
\end{cases}
\]

and the previous question, argue that

\[
M_{a,b}^{0,\lceil(n+k)/2\rceil} = M_{a_1,b_1}^{0,\lceil(n-k)/2\rceil}.
\]

4. Consider the following \texttt{Hgcd} ("Half-GCD") algorithm that takes two polynomials \( a, b \in K[x] \) and returns a matrix \( M_{0,\lceil n/2 \rceil}^{a,b} \) which yields the remainder \( r_{\lceil n/2 \rceil} \).

\[
\text{function Hgcd}(a, b)
\]

\[
\text{If } n = 0 \text{ then return } \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
\]

\[
\text{Else } m = \lceil n/2 \rceil \\
\text{end if}
\]

\[
f \leftarrow a \text{ quo } x^m; \\
g \leftarrow b \text{ quo } x^m \\
M \leftarrow \text{Hgcd}(f, g) \\
\begin{bmatrix} a' \\ b' \end{bmatrix} \leftarrow M \begin{bmatrix} a \\ b \end{bmatrix} \\
c' \leftarrow a' \mod b' \\
M' \leftarrow \begin{bmatrix} 0 & 1 \\ 1 & -(a' \text{ quo } b') \end{bmatrix} \\
b'' \leftarrow b' \text{ quo } x^{\lfloor m/2 \rfloor} \\
c'' \leftarrow c' \text{ quo } x^{\lfloor m/2 \rfloor} \\
M'' \leftarrow \text{Hgcd}(b'', c'') \\
\text{Return } M'' M' M
\]

end function

Using question 2, show its correctness. Argue that the complexity of this algorithm is \( O(M(n) \log^2 n) \).

5. Describe a recursive fast polynomial GCD algorithm of complexity \( O(M(n) \log^2 n) \) that uses \texttt{Hgcd} as a subroutine.