1 Multiplication of two polynomials

Give an algorithm to multiply a degree 1 polynomial by a degree 2 polynomial in at most 4 multiplications.

2 Remainder of a sparse polynomial

In this exercise we are interested in computing a remainder of a sparse polynomial $S$ after dividing by a polynomial $D$, where $S, D \in K[X]$. (Assume that operations in $K$ have unit cost.)

1. Give an example showing that assuming that $S$ is sparse does not lead to better bounds for the classical division algorithm.

2. What is the cost of an operation in $K[X]/(D(X))$?

3. Show that one can compute $X^N \mod D(X)$ in time $O((\deg D)^2 \log N)$. (Hint: use fast exponentiation.)

4. Assume that $S$ has $\omega$ nonzero terms. Show that you get an algorithm of complexity $O(\omega(\deg D)^2 \log \deg S)$ which beats the classical division for $\omega$ at most $\deg S - \deg D \deg D \log \deg S$.

3 Short product

We are given two polynomials $F$ and $G$ both of degree $< n$. We want to compute their short product, i.e., the value $FG \mod x^n$. We can either compute their full product $FG$ in time $O(n^{\ln 3/\ln 2})$ (using Karatsuba) and then discard large-degree coefficients, or we can be smarter and use the so-called Mulders’ trick to get the result faster.

1. Let $k$ be an integer such that $n/2 \leq k \leq n$ and let $M(n)$ denote the complexity of a full product and $S(n)$ the complexity of a short product. Show that a short product of two degree $n$ polynomials can be computed as a full product of two degree $k$ polynomials, and two short products of degree $n-k$ polynomials. In other words, show that

   $$S(n) = M(k) + 2S(n-k).$$

2. Assume that $M(n) = n^\alpha$ for some $\alpha > 1$ (so we leave out the constant factor). Further let $k = \beta n$ for some $\beta < 1$. The goal is to find the optimal value for $\beta$ that minimizes $S(n)$.

   1. $S(n) = \frac{\beta^n}{1-2(1-\beta)^2} M(n)$. You may want to use the fact that $\frac{S(\gamma n)}{S(n)} = \frac{M(\gamma n)}{M(n)}$ for $\gamma > 0$ and sufficiently large $n$.

   2. Find $\beta_{\min}$ as a function of $\alpha$ that minimizes the above expression.
4 Multiplication of bivariate polynomials

Fact: Let \( c_0, \ldots, c_d \) be \( d + 1 \) distinct elements of \( K \) and \( Q_0, \ldots, Q_d \in K[X] \). There is a unique polynomial \( P \in K[X,Y] \) of \( Y \)-degree at most \( d \) satisfying \( P(X, c_i) = Q_i \) for every \( i = 0, \ldots, d \).

Let us assume that we can efficiently find such \( P \). Again, assume that operations in \( K \) have unit cost.

1. What is the cost of a naive multiplication of two bivariate polynomials \( A \) and \( B \) of \( X \)-degree at most \( D_1 \) and \( Y \)-degree at most \( D_2 \)?

2. Give an algorithm that computes \( A(X, c) \) for a given \( c \in K \), with \( A \) of \( X \)-degree at most \( D_1 \) and \( Y \)-degree at most \( D_2 \). What is its cost?

3. Assuming that \( |K| \geq 2D_2 + 1 \) and using the fact above, describe an algorithm for multiplying bivariate polynomials (which would, assuming that we have a fast algorithm for multiplication of polynomials of one variable, beat the naive multiplication).

5 Alternative FFT algorithm

Let \( P \) be a polynomial of degree at most \( 2^k - 1 \), and write \( P = P_h X^{2^k - 1} + P_l \). Let \( \omega \) be a primitive \( 2^k \)-th root of 1.

1. Prove that \( P(\omega^{2i}) = P_h(\omega^{2i}) + P_l(\omega^{2i}) \) and \( P(\omega^{2i+1}) = -P_h(\omega^{2i+1}) + P_l(\omega^{2i+1}) \)

2. Deduce an alternative FFT algorithm. You will need to introduce the polynomial

\[
Q(X) = P_l(\omega X) - P_h(\omega X).
\]

6 Is squaring easier than multiplying?

Show that computing the square of an \( n \)-digit number is not (asymptotically) easier than multiplying two \( n \)-digit numbers. We assume we work in a ring where we can divide by 2.

7 Refined Karatsuba

In class, we’ve seen that Karatsuba algorithm allows to multiply two polynomials of degree \( n \) in time \( O(n^{3/2} \ln n) \). In this exercise we look at a more refined complexity bound and, in particular, improve the \( O(n) \)-factor. Assume, \( n \) is divisible by 2.

1. First, recall Karatsuba identity, where we let \( \deg(F_0) = \deg(G_0) = \lfloor n/2 \rfloor \) and \( k := \deg(F_1) = \deg(G_1) \leq n/2 \).

\[
(F_0 + x^{n/2} F_1)(G_0 + x^{n/2} G_1) = F_0 G_0 + x^{n/2} ((F_0 + F_1)(G_0 + G_1) - F_0 G_0 - F_1 G_1) + x^n F_1 G_1. \tag{1}
\]

Argue that this identity leads to the bound \( M(n) \leq 3M(n/2) + 4n + \Theta(1) \).

2. Consider a quadratic polynomial \( H = h_0 + h_1 x + h_2 x^2 \). Recall that this polynomial can be reconstructed from \( H(0) = h_0, H(1) = h_0 + h_1 + h_2, \) and \( H(\infty) = h_2 \) as \( H = (1-x)H(0) + xH(1) + x(x-1)H(\infty) \). Now assume \( H \) is the result of the product \( (F_0 + x F_1)(G_0 + x G_1) \). Show how to obtain the refined Karatsuba identity

\[
(F_0 + x^{n/2} F_1)(G_0 + x^{n/2} G_1) = (1 - x^{n/2})(F_0 G_0 - x^{n/2} F_1 G_1) + x^{n/2}(F_0 + F_1)(G_0 + G_1). \tag{2}
\]

Estimate the number of multiplications and additions you’ll need to perform using this identity.