NMAI 076 - Algebra 2 - spring semester 2025

Homework 3

Deadline 15.05.2025, 15:40

Exercise 1. (10 points) Let $f \in \mathbb{Z}_{17}[x]$ be a polynomial of degree < 4 with $DFT_4(f) = (0, 1, 0, 1)$. Determine f using the FFT algorithm.

Exercise 2. (10 points) Let $M = 2^{u2^{k-1}} + 1$, for some $u, k \geq 1$.

1. Show that $\omega = 2^u$ is a 2^k-th primitive root of 1 in $\mathbb{Z}_M$.
2. Show that 2^k is invertible modulo M.
3. Conclude that the FFT algorithm (for computing DFT and IDFT) also works for polynomials from $\mathbb{Z}_M[x]$.

Exercise 3. (10 points) The product $c = a \cdot b$ of two numbers given in binary, i.e. $a = \sum_{i=0}^{l} a_i 2^i$ and $b = \sum_{i=0}^{l} b_i 2^i$ for $a_i, b_i \in \{0, 1\}$ can be computed as follows: take the polynomials $A(x) = \sum a_i x^i$ and $B(x) = \sum b_i x^i$ over $\mathbb{Z}[x]$ and compute their product $C(x) = \sum_{i=0}^{2l} c_i x^i$ using fast polynomial multiplication in a ring $\mathbb{Z}_p[x]$, for suitable $p > 2(l + 1)$. Then clearly $c = C(2)$.

1. "If we compute C and return the coefficient $(c_0, c_1, \ldots, c_{2l})$ we get a binary expansion of c in time $O(l \log(l))$." Why is this statement wrong?
2. How can we modify the above to obtain a working algorithm? Give an upper bound to its running time in l and p. For this, observe that we can add or multiply two numbers $a, b \in \mathbb{Z}_p$ (that are given by their binary expansion) in time $O(\log(p)^2)$.

Remark Combining the observations in Exercise 2 and 3 we can derive an algorithm for multiplying two integers that has complexity $O(l \log(l) \log\log(l))$. This algorithm is called the *Schönhage-Strassen-Algorithm*, and was the fastest known until 2007.

The main idea of the algorithm is to represent $a = \sum_{i=0}^{l} a_i 2^{ui}$ and $b = \sum_{i=0}^{l} b_i 2^{ui}$, with $a_i, b_i \in \{0, 1, \ldots, 2^u - 1\}$ (i.e. we subdivide a and b in blocks of u bits, for some u. We then work in a polynomial ring $\mathbb{Z}_M[x]$ with $M = 2^{u2^{k-1}} + 1$, for suitable k. We can compute the product $a \cdot b$ by FFT of polynomials in $\mathbb{Z}_M[x]$, as described in Exercise 3. But now, instead of performing all resulting 'smaller' multiplication over $\mathbb{Z}_M$ by long multiplication as in 3.2, we recursively call our routine to compute them. The running time results from choosing the right values for u and k and the nice properties of $\mathbb{Z}_M$ with respect to binary representations of numbers. In particular, multiplication by powers of the unit root $\omega = 2^k$ in binary can be done by bit-shifts; computing the binary expansion of x mod M can also be done very quickly.

The provably best asymptotic running time of $O(l \log(l))$ was achieved in 2021 by Harvey and van der Hoeven. Their algorithm is however far from being useful in practice, in contrast to Schönhage-Strassen, which also performs well for small l.