CS-E4500 Advanced Course in Algorithms (5 cr)

Problem Set 2 *Spring 2019*

1. Multiplication with the discrete Fourier transform. Let us multiply $f = 1 + x + x^2 \in \mathbb{Z}_{13}[x]$ and $g = 2 + 12x^3 \in \mathbb{Z}_{13}[x]$ using the discrete Fourier transform.

   (a) Compute $\mathrm{DFT}_\omega(f)$ and $\mathrm{DFT}_\omega(g)$ in $\mathbb{Z}_{13}^6$ utilizing the fact that $\omega = 4$ is a primitive root of unity of order 6 in $\mathbb{Z}_{13}$.

   (b) Compute the pointwise product $\mathrm{DFT}_\omega(f) \cdot \mathrm{DFT}_\omega(g) \in \mathbb{Z}_{13}^6$.

   (c) Compute the inverse $\frac{1}{6}\,\mathrm{DFT}_{\omega^{-1}}(\mathrm{DFT}_\omega(f) \cdot \mathrm{DFT}_\omega(g)) \in \mathbb{Z}_{13}^6$.

   Hints: To ease your computations, we have $4^{-1} = 10 \in \mathbb{Z}_{13}$ and $6^{-1} = 11 \in \mathbb{Z}_{13}$. Check that your result for (c) agrees with the sequence of coefficients of $fg \in \mathbb{Z}_{13}[x]$.

2. The convolution identity. Let $\omega \in R$ be a primitive root of unity of order n in a ring R. Show that for all $f, g \in R[x]/\langle x^n - 1\rangle$ we have $\mathrm{DFT}_\omega(fg) = \mathrm{DFT}_\omega(f) \cdot \mathrm{DFT}_\omega(g)$.

   Hints: Recall that we may view f and g as elements of $R[x]$ of degree at most $n-1$, in which case we obtain $fg \in R[x]/\langle x^n - 1\rangle$ by multiplying f and g in $R[x]$ and then substituting $x^n = 1$ until the result has degree at most $n-1$. Recalling that $\omega^n = 1$, show that the vectors on the left-hand side and the right-hand side of the identity agree in each position.

3. The fast Fourier transform. Let $\omega \in R$ be a primitive root of unity of order $n = 2^k$ in a ring R with $k \in \mathbb{Z}_{\geq 0}$. Present detailed pseudocode for an algorithm that given

   $$f = (\varphi_0, \varphi_1, \ldots, \varphi_{n-1}) \in R^n$$

   as input computes the discrete Fourier transform

   $$\mathrm{DFT}_\omega(f) = (\hat{\varphi}_0, \hat{\varphi}_1, \ldots, \hat{\varphi}_{n-1}) \in R^n$$

   in $O(n \log_2 n)$ arithmetic operations in R. Carefully analyse the number of arithmetic operations in R that your algorithm uses.

   Hints: For example, you may want to rely on a recursive design, or alternatively use a factorization of a composite-order DFT as developed in the lecture slides. When working recursively, remember to set up base cases for the recursion. Also, you may want to precompute powers of ω into a look-up table so that they are immediately available. If you want to test your design, you can make use of the fact that $\omega = 19$ is a primitive root of unity of order 32 in $\mathbb{Z}_{97}$. Compare the output of your algorithm with a reference output obtained by multiplying the input with an appropriate Vandermonde matrix. To analyse your algorithm, set up a recurrence and solve it using tools from CS-E3190.

4. Fast integer multiplication by reduction to polynomial multiplication. Let $\alpha, \beta \in \mathbb{Z}_{\geq 1}$ with $\lfloor \log_2 \alpha \rfloor + 1 \leq m$ and $\lfloor \log_2 \beta \rfloor + 1 \leq m$ be given as input. Furthermore, let us assume that α and β are represented in binary as sequences of 64-bit words. That is, we have $\alpha = \sum_{i=0}^{\lfloor m/64 \rfloor} \alpha_i \cdot 2^{64i}$ and $\beta = \sum_{i=0}^{\lfloor m/64 \rfloor} \beta_i \cdot 2^{64i}$ with $\alpha_i, \beta_i \in \mathbb{Z}$ and $0 \leq \alpha_i, \beta_i \leq 2^{64} - 1$. Design an algorithm that computes the product $\gamma = \alpha\beta$ represented as a sequence of 64-bit words using $\tilde{O}(m)$ operations in $\mathbb{Z}_{2^{128}}$.

Hints: You may want to apply the Schönhage–Strassen algorithm from the lecture slides. View α and β as polynomials $a = \sum_{i=0}^{\lfloor m/64 \rfloor} \alpha_i y^i$ and $b = \sum_{i=0}^{\lfloor m/64 \rfloor} \beta_i y^i$ in a polynomial ring $S[y]$ for a carefully chosen coefficient ring S. Maybe you want to try $S = \mathbb{Z}_u$ for some $u \in \mathbb{Z}_{\geq 2}$. Suppose you have access to the polynomial product $c = ab$. How do you recover from c the sequence of words that represents γ? Be careful with carries in addition. How does the size of S depend on m? Observe also that 2 must be a unit in S if you want to apply Schönhage–Strassen, so this somewhat limits your choice for u. Carefully justify that the number of operations in $\mathbb{Z}_{2^{128}}$ used by your algorithm is $O(m(\log m)^d)$ for some constant d independent of m. You may use classical arithmetic algorithms for arithmetic in S, but note that each arithmetic operation in S may consume multiple operations in $\mathbb{Z}_{2^{128}}$ and these need to be accounted for in your analysis.

Deadline and submission instructions. This problem set is due no later than Sunday 27 January 2019, 20:00 (8pm), Finnish time. Please submit your solutions as a single PDF file via e-mail to the lecturer (`petteri.kaski(atsymbol)aalto.fi`). Please use the precise title

```
CS-E4500 Problem Set 2: [your-student-number]
```

with "`[your-student-number]`" replaced by your student number. For example, assuming that my student number is 123456, I would carefully title my e-mail

```
CS-E4500 Problem Set 2: 123456
```

and attach to the e-mail a single PDF file containing my solutions. Please note that the submissions are automatically processed and archived, implying that failure to follow these precise instructions may result in your submission not being graded.