TUTORIAL 2

1 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 ω 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

$$Q(X) = P_l(\omega X) - P_h(\omega X).$$

2 Is squaring easier than multiplying?

Show that computing the square of a 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.

3 The “binary splitting” method computation of $n!$

We want to compute $n!$ and assume that n is a ”small” integer (i.e. it fits into one machine word). We denote with $M(k)$ the cost (in terms of elementary operations) of the multiplication of two k-bit numbers, and we assume $2M(k/2) \leqslant M(k)$ (we recall some typical values: $M(k) = O(k^2)$ with naive multiplication, $O(k^{\log(3)/\log(2)})$ with Karatsuba multiplication and $O(k \log k \log\log k)$ with the FFT-in finite ring variant of the Schönhage & Strassen algorithm). Use the fact that $\log n! \sim n \log n$.

1. What is the cost of multiplying $O(n)$-digit integer by a $O(1)$-digit integer by the naive algorithm. Argue that it is essentially optimal.

2. We first consider the simplest approach: $x_1 = 1,\ x_2 = 2x_1,\ x_3 = 3x_2,\ \ldots,\ x_n = nx_{n-1}$. Show that the cost of this approach is $O(n^2(\log n)^2)$.

3. We define

$$p(a, b) = (a + 1)(a + 2) \cdots (b - 1)b = \frac{b!}{a!}.$$

   Suggest a recursive method to compute $n!$ with cost $O(\log n M(n \log n))$. Conclude on the complexity of your method under different values of $M(k)$.

4 Logarithm and exponential

For polynomials $S, T \in \mathbb{K}[X]$ such that $S(0) = 0$ and $T(0) = 0$, we define

$$\exp_n(S(X)) = \sum_{k=0}^{n-1} \frac{S(X)^k}{k!} \bmod X^n$$

$$\log_n(1 + T(X)) = \sum_{k=1}^{n-1} (-1)^{k+1} \frac{T(X)^k}{k} \bmod X^n$$

1. Assume $A(0) = 0$, prove that $(A(X) + 1)^{-1} = \sum_{k=0}^{n-1} (-1)^k A(X)^k \bmod X^n$
2. Recall that $S(0) = 0$. Let $U_n(X) = S'(X)/(S(X) + 1) = \sum_{k=0}^{n-2} u_k X^k \bmod X^{n-1}$ (note that $S(X) + 1$ is invertible modulo X^n because $S(0) + 1 \neq 0$). Prove that

$$\log_n(1 + S(X)) = \sum_{k=1}^{n-1} u_{k-1} \frac{X^k}{k} \bmod X^n$$

3. Deduce a quasi-linear time (in the degree of $S(X)$) algorithm to compute $\log_n(S(X) + 1)$.
4. Prove that if $T(0) = 0$, then $\log_n(\exp_n(T(X)) = T(X)$ (remark that this is well defined because $\exp_n(T(0)) = 1$). (Hint: take the derivative).
5. Let $Y = \exp_N(T(X)) - 1 \mod X^N$. Using the above, we have that

$$f(Y) = \log_N(1 + Y) - T(X) = 0 \mod X^N.$$

   Using Hensel lifting, deduce an algorithm for computing $Y = \exp_N(T(X)) - 1 \mod X^N$ using $O(M(N))$ operations in K. (Hint: remember that as M is super-linear, we have that $M(N) + M(N/2) + \cdots + M(N/n^k) + \cdots \leq 2M(N)$).