

Proposal for Lab # 7

Karatsuba's polynomial multiplication algorithm

Algorithm: R is a commutative ring, with 1.

- Input: $f, g \in R[x]$ of degrees less than $n = 2^k$.
- Output: $fg \in R[x]$.
 1. If $n = 1$ then return $f.g \in R$.
 2. Let $f = F_1x^{n/2} + F_0$ and $g = G_1x^{n/2} + G_0$, with $F_0, F_1, G_0, G_1 \in R[x]$ of degrees less than $n/2$.
 3. Compute F_0G_0 , F_1G_1 , and $(F_0 + F_1)(G_0 + G_1)$ by a recursive call.
 4. return $F_1G_1x^n + ((F_0 + F_1)(G_0 + G_1) - F_0G_0 - F_1G_1)x^{n/2} + F_0G_0$.

1.

- a) Write a code **Karatsuba[P,Q,n]** with n being a power of two whose output is an array of $2n$ coordinates corresponding to the coefficients of PQ .
- b) Test the code on symbolic polynomials of degrees 2 and 3 (choose $n= 4$ in both cases).
- c) Test the code numerically on big random polynomials.
- d) Write a code with the naive product of two polynomials and compare the timing of both of them.
- e) Verify that if the size of the input polynomials is multiplied by 2, then the algebraic complexity of **Karatsuba** is multiplied by 3, while in the case of the naive algorithm the complexity is multiplied by a factor close to 4.

2. Let R be a ring (commutative, with 1) and $f, g \in R[x]$ of degrees less than n . You are to analyze two variants of Karatsuba's algorithm when n is not a power of two.

- a) The first variant is to call the initial algorithm with $2^{\lceil \log n \rceil}$ instead of n . Show that this takes at most $9 \cdot 3^{\lceil \log n \rceil} \leq 27 \cdot n^{\log 3}$ operations in R .

- b) Let $m = \lceil n/2 \rceil$, and modify the Karatsuba algorithm so that it divides f and g into blocks of degree less than m . If $T(n)$ denotes the cost of this algorithm, show that

$$\begin{aligned} T(1) &= 1 \\ T(n) &\leq 3T(\lceil n/2 \rceil) + 4n. \end{aligned}$$

- c) Show that $T(2^k) \leq 9 \cdot 3^k - 8 \cdot 2^k$ and $T(2^{k-1} + 1) \leq 6 \cdot 3^k - 4 \cdot 2^k - 2$ for all $k > 0$, and compare this running time with the algorithm proposed in a) for $n = 2^k$ and $n = 2^{k-1} + 1$.

Homework : Implement both algorithms given in 2a) and 2b) for $R = \mathbb{Z}$. Experiment with various values of n (say $2 \leq n \leq 50$). Use random polynomials with one-digit coefficients and also with n -digit coefficients.