Assignment 1

September 23, 2009

1. Give the steps of Karatsuba's algorithm with the input polynomials $1 - x - 2x^2 + 3x^3$ and $1 - x - 2x^2 - x^3$.

   For all such computational questions, you are free to do the computations by hand, or to implement the algorithm and run it. If you implement in a language like C or java, you can use `float`*s or* `int`*s as coefficients.*

2. Give the steps of the Fourier Transform for $n = 4$, with the input polynomial $P = 1-x-2x^2-3x^3$. Then, perform an *inverse* Fourier Transform to recover the polynomial P.

3. Using the master theorem, give a big-O estimate for the function S under the following assumptions:

   - $S(n) = 3S(n/2) + n^{1.2}$
   - $S(n) = 4S(n/2) + n^{1.75}$
   - $S(n) = 3S(n/2) + n^{1.5}$

4. As a warm-up, recall the (easy) proof that the number of multiplications in Karatsuba's algorithm is $O(n^{\log_2(3)})$. Then, this problem asks you to write the proof in the general case, *without using the master theorem*. To make things precise, we start from the following recursion:

   - $T(1) = 1$
   - $T(n) = 3T(n/2) + \ell n$, for n a power of 2, where ℓ is a constant.

   Prove that

   $$T(2) = 3 + 2\ell, \quad T(4) = 9 + 10\ell, \quad T(8) = 27 + 38\ell$$

   and more generally, for $n \geq 2$ (by induction)

   $$T(2^n) = 3^n + 2(3^{n-1} + 2 \cdot 3^{n-2} + \cdots + 2^{n-2} \cdot 3 + 2^{n-1})\ell.$$

   Deduce that $T(2^n) = O(3^n)$.

5. Prove the uniqueness of the quotient and remainder in Euclidean division.

6. Prove the correctness of the addition and multiplication rules in Euclidean division.

7. Prove that you can compute the *modular multiplication*

$$c_0 + c_1 x = (a_0 + a_1 x)(b_0 + b_1 x) \text{ rem } (x^2 + 2)$$

using **3** multiplications (don't count the multiplication by a constant as a "real" multiplication).

Hint: Use the trick of Karatsuba's algorithm.

8. You are to study an alternative to Karatsuba or Toom to multiply polynomials. Let $f = f_0 + f_1 x + f_2 x^2$ and $g = g_0 + g_1 x + g_2 x^2$, and let $h = h_0 + h_1 x + h_2 x^2 + h_3 x^3 + h_4 x^4$ be their product. For this size of inputs, the algorithm does the following:

   (a) compute $F_0 = f(0), F_1 = f(1), F_{-1} = f(-1), F_{x^2+2} = f \text{ rem } (x^2 + 2)$.

   (b) compute $G_0 = g(0), G_1 = g(1), G_{-1} = g(-1), G_{x^2+2} = g \text{ rem } (x^2 + 2)$.

   (c) compute $H_0 = F_0 G_0, H_1 = F_1 G_1, H_{-1} = F_{-1} G_{-1}, H_{x^2+2} = F_{x^2+2} G_{x^2+2} \text{ rem } (x^2 + 2)$.

   (d) recover h.

   First, prove that

$$H_0 = h(0),\ H_1 = h(1),\ H_{-1} = h(-1),\ H_{x^2+2} = h \text{ rem } (x^2 + 2).$$

   Then, show how to recover h from $H_0,\ H_1,\ H_{-1},\ H_{x^2+2}$. Finally, count how many multiplications you use. *Hint: use the previous problem.*

   Without giving all details, explain how you could use this trick recursively, and indicate what complexity you would expect.

9. How much time did you spend on the assignment?