Prof. Friedrich Eisenbrand
Alfonso Cevallos

Discussions from: April 21, 2015

Computer Algebra

Spring 2015

Assignment Sheet 5

Note: These are just notes and not necessarily full solutions to each exercise. Please report any mistakes you may find.

Exercise 1

Remember that ω is a primitive n-th root of unity in a ring R if: a) $\omega^n = 1$; b) if $\mathbb{1}$ is the multiplicative identity element in R, then $n\mathbb{1} = \mathbb{1} + \cdots + \mathbb{1}$ (n times) is invertible in R; and c) $\omega^{n/p} - 1$ is not a zero divisor in R for any prime p dividing n.

1. The three properties that define primitive n-th roots of unity are easy to verify.
2. $\mathbb{Z}_8$ does not have a primitive square root of unity, because 2 is not invertible in $\mathbb{Z}_8$.
3. We can actually prove a more general statement: If ω is a primitive n-th root of unity, for $n = km$, then $\sigma = \omega^k$ is a primitive m-th root of unity. Proof: $\sigma^m = \omega^{km} = \omega^n = 1$; also σ is invertible, as $\sigma(\omega^{-1})^k = \omega^k(\omega^{-1})^k = 1$; and c) if p is a prime dividing m, then it also divides n, so $\sigma^{m/p} - 1 = (\omega^k)^{n/kp} - 1 = \omega^{n/p} - 1$ is not a zero divisor.

Exercise 2

Throughout this exercise we consider the ring $\mathbb{Z}_M$, with $M = 2^L + 1$, and a number $K = 2^k$ that divides L. Keep in mind that $2^L = -1$ in this ring.

1. The number $\omega = 2^{L/K}$ is a primitive $2K$-th root of unity because: a) $\omega^{2K} = (2^{L/K})^{2K} = (2^L)^2 = (-1)^2 = 1$; b) $2k$ is invertible, as $(2K)(2^{2L-k-1}) = 2^{k+1}2^{2L-k-1} = 2^{2L} = (-1)^2 = 1$; and c) $\omega^{2K/2} - 1 = (2^{L/K})^K - 1 = 2^L - 1 = -1 - 1 = -2$ is not a zero divisor.
2. We are to multiply two given numbers a and 2^j ($1 \leq j \leq L$), which are expressed as bit lists, and we must output $m = a \cdot 2^j$ reduced mod M (so $0 \leq m < M$). The challenge is to use neither standard multiplication nor division with remainder, because these operations are not linear. What we *can* use is additions, subtractions, and bit-shifting operations. We do the following:
a) Write a as $a_1 \cdot 2^{L-j} + a_0$, where $a_0 < 2^{L-j}$. We quickly find a_0 and a_1 by splitting the bit list of a into two sub-lists (as we did in Karatsuba algorithm).
b) Compute $m = a_0 \cdot 2^j - a_1$, and notice that $a \cdot 2^j = a_1 \cdot 2^L + a_0 \cdot 2^j \equiv a_0 \cdot 2^j - a_1 = m$ (because $2^L \equiv -1$), where $a_0 \cdot 2^j$ is quickly computed via bit-shifting, and $a_0 \cdot 2^j < 2^L$. Therefore, $|m| \leq \max(a_0 \cdot 2^j, a_1) < M$ (i.e. m is already "almost" reduced mod M).
c) If $m < 0$, let $m = m + M$. Output m.

3. Polynomials[1] $f, g \in \mathbb{Z}_M[X]$ are given in coefficient representation, and we want to compute their product $h = f \cdot g$, also in coeff. rep. All polynomials have degree $O(K)$, so they are stored as vector with $O(K)$ coordinates, and each entry is of size $O(L)$. The algorithm to obtain h consists of 3 parts:
   a) The evaluation part transforms f and g into their point-value representations $Y_f = DFT_\omega(f)$ and $Y_g = DFT_\omega(g)$. As seen in class, this algorithm takes $O(K \log K)$ basic operations. But these operations are just additions, subtractions and multiplications by powers of 2, all of complexity $O(L)$. Thus the bit-complexity of this step is $O(KL \log K)$.
   b) In point-value representation, we obtain $Y_h = Y_f \cdot Y_g$ simply by coordinate-wise multiplication. For each of the $O(K)$ coordinates, we perform a multiplication and a division with remainder (to reduce mod M) in time $O(M(L))$, so the complexity of this step is $O(KM(L))$.
   c) The interpolation part transforms Y_h into h, by computing $h = (DFT_\omega)^{-1}(Y_h) = (2K)^{-1} DFT_{\omega^{-1}} Y_h$, where $(2K)^{-1} = -2^{L-k-1}$ and $\omega^{-1} = -2^{L-L/K}$ (why?). This part, as part a), also has a complexity of $O(KL \log K)$.
   Therefore, the total complexity is $O(KL \log K + KM(L))$. Note: Considering that $L \geq K$, and $M(L) = \Omega(L \log L)$ for all currently known multiplication algorithms, we get $O(KL \log K + KM(L)) = O(KM(L))$.

Exercise 3

We are given polynomials $f(x) = \sum_{i=0}^{n} a_i x^i$ and $g(x) = \sum_{j=0}^{n} b_j x^j$ in $\mathbb{Z}[x]$, with $|a_i|, |b_j| \leq B$ for all i, j, and want to compute $h(x) = f(x) * g(x)$. We simply consider f, g, h as polynomials in $\mathbb{Z}_M[X]$, for conveniently chosen values of M and K, and perform the process detailed in exercise 2.

Choosing K: We know that the degree of h is at most $2n$. We pick $K = 2^k$ as the smallest power of 2 such that $2K > 2n$, so $K \leq 2n = O(n)$.

Choosing $M = 2^L + 1$: There are two conditions on L. First, L must be a multiple of K. Next, if $h(x) = \sum_{k=0}^{2n} c_k x^k$, we have $c_k = \sum_{i+j=k} a_i b_j$; so for any k, $|c_k| \leq |\sum_{i+j=k} a_i b_j| \leq (n+1)B^2$. So each coefficient c_k takes one of $2(n+1)B^2 + 1$ possible values. Selecting $M \geq 2(n+1)B^2 + 1$ ensures that h can be mapped from $\mathbb{Z}_M[x]$ back to $\mathbb{Z}[x]$ unambiguously, so $L \geq \log_2(2(n+1)B^2)$. We can pick an L satisifying both conditions, such that $L = O(\max(n, \log n + size(B))) = O(n + size(B))$.

Choosing ω: We know that $\omega = 2^{L/K}$ will be a $2K$-th root of unity.

Reconstruction: Once we find $h(x) = \sum_{k=0}^{2n} c_k x^k$ in $\mathbb{Z}_M[x]$, we make sure that each coefficient c_k is in the range $[-\frac{M-1}{2}, \frac{M-1}{2}]$. This is the correct mapping back to $\mathbb{Z}[x]$.

Complexity: From exercise 2, the total complexity is $O(KM(L)) = O(nM(n + size(B)))$.

[1] Please have a look at the chart on page 2 of the scanned notes for this explanation. http://disopt.epfl.ch/files/content/sites/disopt/files/shared/cal15/Lecture08.pdf

Exercise 4
For convenience, when we reduce modulo 17, we opt to keep all numbers between -8 and 8. For $f(x) = 5x^3+3x^2-4x+3$ and $g(x) = 2x^3-5x^2+7x-2$ in $\mathbb{Z}_{17}[x]$, the standard polynomial multiplication (mod 17) gives $h(x) = fx)g(x) = -7x^6-2x^5-5x^4+3x^3+2x^2-5x-6$. We are working in the framework of exercise 2, with $L = K = 4$, so from part 2.2 it follows directly that $\omega = 2^{L/K} = 2$ is a primitive 8-th root of unity. Its inverse is $w^{-1} = -8$. The Vandermonde matrices are:

$$V_2 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & -1 & -2 & -4 & -8 \\ 1 & 4 & -1 & -4 & 1 & 4 & -1 & -4 \\ 1 & 8 & -4 & 2 & -1 & -8 & 4 & -2 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -2 & 4 & -8 & -1 & 2 & -4 & 8 \\ 1 & -4 & -1 & 4 & 1 & -4 & -1 & 4 \\ 1 & -8 & -4 & -2 & -1 & 8 & 4 & 2 \end{pmatrix}, \; V_{-8} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -8 & -4 & -2 & -1 & 8 & 4 & 2 \\ 1 & -4 & -1 & 4 & 1 & -4 & -1 & 4 \\ 1 & -2 & 4 & -8 & -1 & 2 & -4 & 8 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 8 & -4 & 2 & -1 & -8 & 4 & -2 \\ 1 & 4 & -1 & -4 & 1 & 4 & -1 & -4 \\ 1 & 2 & 4 & 8 & -1 & -2 & -4 & -8 \end{pmatrix}$$

And $V_2 V_{-8} = 8I_8$. Seen as vectors, we have $f = (3,-4,3,5,0,0,0,0)$ and $g = (-2,7,-5,2,0,0,0,0)$. In the evaluation part, we get $Y_f = DFT_2(f) = V_2 \cdot f = (7,-4,-2,3,5,0,2,-4)$ and $Y_g = DFT_{-8}(g) = V_{-8} \cdot g = (2,8,6,-7,1,-1,0,-8)$; and by coordinate-wise multiplication we obtain the vector $Y_h = (-3,2,5,-4,5,0,0,-2)$.

Finally, $DFT_{-8}(Y_h) = V_{-8} \cdot Y_h = (3,-6,-1,7,-6,1,-5,0)$; and multiplying each entry of this last vector by $8^{-1} = -2$ yields $h = (-6,-5,2,3,-5,-2,-7,0)$, which is what we computed at the beginning.

Exercise 5
See the code.