MATH 895 Assignment 1, Summer 2009

Instructor: Michael Monagan

Please hand in the assignment by 3:30pm May 27th.
Late Penalty -20% off for up to one day late. Zero after that.
For Maple problems, please submit a printout of a Maple worksheet containing Maple code and the execution of examples.

References: Sections 4.5–4.8 of Geddes, Czapor and Labahn and sections 8.2–8.3, 11.1 of von zur Gathen and Gerhard.

Primitive n'th roots of unity.

(a) Let F be a field, n be even and ω be a primitive n'th root of unity in F.
Show that (i) $\omega^i = -\omega^{i+n/2}$ and (ii) ω^2 is a primitive $n/2$'th root of unity.

(b) The ring $\mathbb{Z}_5$ does not have a primitive 8'th root of unity. Consider the ring $R = \mathbb{Z}_5[y]/(y^4+1)$. In class we saw that $[y] \in R$ is a primitive 8'th root of unity. Find a smallest subring S of R which has a primitive 8th root of unity and identify one primitive 8th root of unity in S. Hint: y^4+1 is not irreducible in $\mathbb{Z}_5[y]$.

Question 2 Implementing the FFT.

Implement the FFT, the forward transform, in Maple. See algorithm 4.4 in the Geddes text. Program it to take as input a Maple list of integers, e.g., $[a_0, a_1, ..., a_{m-1}, 0, ..., 0]$ for $a(x)$, and to output a list of integers. To make your implementation efficient optimize it for the case $n = 2$.

Check that your implementation is correct by computing the Fourier transform of the following polynomial $f(x)$ using the prime $p = 7 \times 2^{20} + 1$, then applying the inverse FFT to get back to $f(x)$. You will need a primitive $n = 64$'th root of unity.

```
> p := 7*2^20+1;
> f := Randpoly(50,x) mod p;
> a := [seq(coeff(f,x,i),i=0..50), 0$13];
```

Time your implementation on inputs of suitable degree d and check that the complexity of your implementation is $O(d \log d)$ and NOT $O(d^2)$.

Question 3 Integer multiplication using the FFT.

Design and implement an algorithm which uses the FFT to multiply two large integers a and b using three machine primes p_1, p_2, p_3 and then applying the Chinese remainder theorem. Do this using the base $B = 2^{30}$ and the primes $p_1 = 2^{24} \times 10 + 1$, $p_2 = 2^{24} \times 28 + 1$ and $p_3 = 2^{24} \times 45 + 1$. Test your algorithm on multiplying $a \times b$ where

```
> r := rand(2^(10^6)):
> a := r():
> b := r():
```

Do not focus on the efficiency of splitting up a large integer into blocks. Just use the following Maple command to write an integer a base B.

```
> Blocks := convert(a,base,B);
```

Question 4 Shönhage Strassen integer multiplication.

Implement the Schönhage-Strassen integer multiplication algorithm in Maple. It uses a large integer modulus of the form $p = 2^{2^r} + 1$. To divide by p just use Maple so that you can reuse your FFT code from question 2 here. Also, use Maple for doing the multiplications in $C := [A_i \times B_i \bmod p, \text{ for } i = 1..n]$, i.e. don't make these multiplications recursive.

Test your algorithm on the example in question 3.

Question 5 The fast Euclidean algorithm.

Implement the fast HGCD procedure that on input of two integers $a \geq b > 0$ of length n digits outputs a matrix A such that $[c, d]^T := A[a, b]^T$ satisfies $\gcd(a, b) = \gcd(c, d)$ and the length of d is about half the length of a. Now write a procedure GCD that repeatedly calls HGCD to compute the gcd of a and b. Just try to get this to work. Use our own examples. Don't worry too much about efficiency.

To obtain the leading half of the digits of a (and b) use

```
> n := ilog2(a);
> m := iquo(n+1,2);
> a1 := iquo(a,2^m);
> b1 := iquo(b,2^m);
```