MATH 895 Assignment 1, Summer 2017

Instructor: Michael Monagan

Please hand in the assignment by 9:30am Thursday May 18th.
Late Penalty -20% off for up to 24 hours 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 4.5–4.9 of Geddes, Czapor and Labahn and/or sections 8.2,8.3,9.1 of von zur Gathen and Gerhard.

Question 1 An iterative FFT.

The file FFT1.c contains my recursive radix 2 FFT code. Convert it to an iterative algorithm. You will need to accomplish the bit-reverse permutation separately. Check that your algorithm is correct by checking that

$$FFT^{-1}(FFT(A,\omega),\omega^{-1}) = nA$$

for a problem of your choosing modulo $p = 97$.

Question 2 Analysis of the FFT.

Let K be a field and $\omega = i$ be a primitive 4'th root of unity in K. Let $a = a_0 + a_1 x + a_2 x^2 + a_3 x^3$ and $A = [a_0, a_1, a_2, a_3] \in K^4$. The FFT computes $F = [a(1), a(\omega), a(\omega^2), a(\omega^3)]^T$. This polynomial evaluation can be expressed as an affine transformation. Let V_4 be the 4×4 Vandermonde matrix

$$V = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$

Then the FFT computes $V_4 A^T$, that is, $F = V_4 A^T$. For both FFT algorithms (FFT1 and FFT2) factor the matrix V_4 into a product of three matrices so that $V_4 = UVW$ where one of the matrices will be a permutation matrix. The two factorizations explain how the two algorithms both compute $V_4 A = F$. Check that the two permutation matrices are inverses of each other.

Question 3 Fast Division

Consider dividing a by b in $F[x]$ where $\deg a = d$, $\deg b = m$ with $d \geq m > 0$. Program the Newton iteration (Algorithm 4.6) recursively in Maple for $F = \mathbb{Z}_p$ to compute f^{-1} as a power series to $O(x^n)$ where $n = d - m + 1$.

To make the Newton iteration efficient when n is not a power of 2, compute $y = f^{-1}$ recursively to order $O(x^{\lceil n/2 \rceil})$. To truncate a polynomial b modulo x^n you could use the Maple command `rem(b,x^n,x)`. Use `convert(taylor(b,x,n),polynom)` instead which is more efficient in Maple.

Test your algorithm on the following problem in $\mathbb{Z}_p[x]$.

```
> p := 101;
> a := randpoly(x,degree=100,dense) mod p;
> b := randpoly(x,degree=50,dense) mod p;
> Quo(a,b,x) mod p;
```

Question 4 Complexity of Fast Division

Let $f \in F[x]$ and let $D(n)$ be the number of multiplications in F for computing f^{-1} as power series to order $O(x^n)$ using the Newton iteration. Let $M(n)$ be the number of multiplications in F that your favorite multiplication algorithm takes to multiply two polynomials of degree $n-1$ in $F[x]$. For $n = 2^k$ explain why

$$D(n) = D(n/2) + M(n) + M(n/2) + cn$$

for some constant $c > 0$. Now, using $D(1) = d$ for some constants $d > 0$, solve this recurrence relation, show that

$$D(n) < 3M(n) + 2cn + d.$$

Use the fact that $2M(n/2) < M(n)$, i.e., $M(n) > O(n)$.
Thus conclude that the cost of the Newton iteration is roughly 3 multiplications.