Arithmetic Algorithms 2014: HW2

Taylor Lee

March 3, 2014

Problem 1

Part A:

First, since a has an inverse $\mod p$, we know that it has an inverse $\mod p^k$, and we set out to find it with newton iteration. Hence, if we let $f(x) = a - \frac{1}{x} \equiv 0 \pmod{p^k}$, then

$$x_{i+1} = x_i - \frac{f_i(x_i)}{f_i'(x_i)} \implies x = x - \frac{a - \frac{1}{x}}{\frac{1}{x^2}}$$

which in turn gives us:

$$x' = x - \left(ax^2 - x\right) = 2x - ax^2$$

Now if we multiply both sides by a, we see a multiple of p^{2k} emerge:

$$ax' = 1 - 1 + 2xa - a^2x^2 = 1 - (1 - ax)^2 \equiv 0 \pmod{p^{2k}}.$$

This is since $ax - 1 \equiv 0 \pmod{p^k} \implies (1 - ax)^2 \equiv 0 \pmod{p^{2k}}$. Thus we are left with:

$$ax' \equiv 1 \pmod{p^{2k}}.$$

Part B:

Here, since a again has an inverse $\pmod p$, and hence higher powers of p, we can start our lift again with Newton Iteration. Thus, $a - \frac{1}{x^2} = 0 \pmod{p^k}$ gives us:

$$x_{i+1} = x_i - \frac{f_i(x_i)}{f_i(x_i)} \implies x' = x - \frac{a - \frac{1}{x^2}}{\frac{2}{x^3}}$$

which in turn gives us:

$$x' = x - \left(\frac{ax^3 - x}{2}\right) = \left(\frac{x}{2}\right)\left(3 - ax^2\right) \pmod{p^{2k}}$$

We can slightly modify previous right hand side to get:

$$x' = x + \frac{x}{2}\left(1 - ax^2\right) \pmod{p^{2k}}$$

And so after we square both sides and multiply through by a we get:

$$ax'^2 = ax^2 + ax^2\left(1 - ax^2\right) + \left(1 - ax^2\right)^2 \pmod{p^{2k}}.$$

Now we can see that the term on the far right is equivalent to 0 (mod p^{2k}). If $1 - ax^2$ is congruent to 0 (mod p^{2k}), then our work is finished, and we are left with a 1 on the right hand side of our equality:

$$ax^2 \equiv 1 \pmod{p^{2k}}.$$

If this is not the case, then $1 - ax^2 \equiv p^k \pmod{p^{2k}}$, and our main equality changes to:

$$ax^2 = 1 - p^k + (1 - p^k)p^k = 1 - p^k + p^k - p^{2k} = 1 \pmod{p^{2k}}$$

Hence, part b is proved.

Problem 2

Part A:

$L_t(a)$ is clearly linear, since whenever a polynomial is squared, all possible pairs of terms are multiplied together and added to the new sum, and the only time there are not two copies of a these new terms in the new sum is when a term is paired with itself: hence for any $a \in R/[t]$, sending it through the map will only double the degrees of the terms of a, before finally subtracting at from this new term, $a^2 \pmod{\mathbb{Z}_2}$. In other words, if $a, b \in R/[t]$, then:

$$L_t(a+b) = (a+b)^2 - t(a+b) \pmod{f} = a^2 + b^2 - ta - tb \pmod{f}.$$

And hence $L_t(a)$ is $\mathbb{Z}_2$ linear.

Part B:

First, suppose that $a \in \ker(L_x)$, then $a^2 \equiv aX \pmod{f}$, and hence this congruency must hold for all $f_1, f_2, \dots f_i$ irreducible factors of f. Since we have

$$a^2 \equiv ax \pmod{f_i}$$

for all f_i irreducible, we can make an observation regarding the form of the a in our congruences. Namely, since f_i is irreducible, it forms a field when its quotient is taken from the polynomial ring $\mathbb{Z}_2[x]$. Hence, either a must have an inverse in $\mathbb{Z}[x]/(f)$, which would imply $a = X \pmod{f_i}$, or a must be zero $\pmod{f_i}$, which concludes our proof from left to right.

Now if a is congruent to $0, X \pmod{f_i}$, for some f_i, we can use the Chinese Remainder Theorem to create a solution to this system of congruences which will be unique $\pmod{f}$. To do this, we create a sum $\pmod{f}$ which is a chain of terms of the form $e_i \bar{f}_i X$, where $\bar{f}_i$ is the product of all the other irreducible f_j which divide f, e_i is the inverse of $f_i \pmod{f}$. This term will create a sort of dimple at f_i, being congruent to $X \pmod{f_i}$, and zero everywhere else. Now, if this term is only congruent to $X \pmod{f_i}$ and zero everything else, then it will be congruent to $X \pmod{f}$, and hence will be in the kernel of L_X, since $X^2 - X^2 = 0 \pmod{f}$. Since L_X is $\mathbb{Z}_2$ linear, the sum all of the individual dimples that are added together to mimic the behavior of a single polynomial over the smaller congruences will be mapped as if they were sent through the function individually and added together in the image. Since each one on its own is sent to zero, their sum in the image will be zero, which means their sum, if taken before the function, is in the kernel.