Solutions to CS 70 Challenge Problems: Modular Arithmetic and Polynomials

Other worksheets and solutions at https://alextseng.net/teaching/cs70/

Alex Tseng

---

1 Modular Arithmetic

(a) Let p be a prime number ($p > 2$). Recall that a perfect square y is such that there exists an integer x where $x^2 = y$. With a modulus, a perfect square y is such that there exists an integer x where $x^2 \equiv y \pmod{p}$. Then that makes x the square root of y. Prove that for any $a \in \{1, 2, ..., p-1\}$, a has either 0 or 2 distinct square roots (mod p).

Consider if we find a single square root r for a. That is, we find that $r^2 \equiv a \pmod{p}$. Then it must be that $-r$ is also a square root of a. Furthermore, $r \neq -r \pmod{p}$. This is because p must be odd (recall, p is a prime larger than 2). Therefore, we are guaranteed that if there exists a square root of a, then there must be at least 2 square roots.

Now we prove that there can only be 2 square roots. Let us assume for contradiction that in addition to $\pm r$, we find $\pm s$ are also square roots of a, where $|r| \neq |s| \pmod{p}$. Then $r^2 \equiv s^2 \equiv a \pmod{p}$, or $r^2 - s^2 \equiv 0 \pmod{p}$. Factoring, we get that $(r+s)(r-s) \equiv 0 \pmod{p}$. Then either $r + s \equiv 0 \pmod{p}$ or $r - s \equiv 0 \pmod{p}$. This implies that $|r| = |s|$—contradiction. Therefore, a can only have 0 or 2 square roots (mod p).

(b) Prove that there are exactly $\frac{p+1}{2}$ perfect squares (mod p).

We use the proposition from part (a). Note that if we square each possible number in mod p except 0 ($\{1, ..., p-1\}$), we get a perfect square each time. This yields all the possible perfect squares. Notice that we know for sure that all of these perfect squares have square roots, because we constructed them by ourselves. By part (a), we know that each one has two square roots. Within our range of $p - 1$ numbers that we squared, each perfect square has two distinct square roots, therefore they must overlap. That is, two of the numbers that we square must yield the same perfect square. This means that there can only be $\frac{p-1}{2}$ distinct perfect squares possible. We also notice that $0^2 \equiv 0 \pmod{p}$, so that gives one more perfect square. Thus, there are $\frac{p+1}{2}$ perfect square (mod p).

(c) *Challenge* Prove that there are *at least* $\frac{p}{3}$ perfect *cubes* (mod p).

Finding a cube root of a (mod p) is equivalently solving the equation $x^3 - a \equiv 0 \pmod{p}$. This cubic polynomial we know can have at most 3 roots. Therefore, each perfect cube can have at most 3 cube roots. Using the same logic as above in part (b), we realize that in our limited range of p values to cube, no cube can appear more than 3 times. Then there are at least $\frac{p}{3}$ cubes.

We could have also used this polynomial roots approach for part (b), but since we needed an exact number of perfect squares, we would have needed to show that any quadratic polynomial we came up with would have exactly 2 roots.

(d) *Challenge* Digit the cybird has sent a coded message with Motherboard access codes using RSA to Inez, Jackie, and Matt. He sends them each a copy of the same message. Inez, Jackie, and Matt each have their own set of public keys: $(N_I, e_I), (N_J, e_J), (N_M, e_M)$. It turns out that all three of them have selected the same encryption exponent $e_I = e_J = e_M = 3$. Additionally, Digit is quite terse in his message m, so that it is smaller than each of the public key moduli: $m < N_I, N_J, N_M$. Hacker has intercepted all three encrypted messages E_I, E_J, E_M, and wishes to decipher the access codes. Show how he can do this efficiently (without factoring the public keys).

If the secret message is m, than Hacker has intercepted m^3 for different 3 different moduli: N_I, N_J, and N_M. Thus, he can write the following congruences:

$m^3 \equiv E_I \pmod{N_I}$

$m^3 \equiv E_J \pmod{N_J}$

$m^3 \equiv E_M \pmod{N_M}$

This system of congruences can be solved easily using the Chinese Remainder Theorem. Let this solution

be x. That is, $x \equiv m^3 \pmod{N_I N_J N_M}$. Because $m < N_I, N_J, N_M$, it must be that $m^3 < N_I N_J N_M$. Thus, taking (mod $N_I N_J N_M$) of m^3 will not truncate this value at all. Then $x = m^3$ as a raw integer, not in any modulus. Taking cube roots of an integer is easy (without any modulus).

2 Polynomials

(a) Find a polynomial *of lowest degree* that satisfies the following congruences:
$P(0) \equiv 4 \pmod 7$
$P(2) \equiv 5 \pmod 7$
$P(4) \equiv 0 \pmod 7$
This is mechanical LaGrange Interpolation.
$\Delta_0 = (8)^{-1}(x-2)(x-4)$
$\Delta_2 = (-4)^{-1}x(x-4)$
$\Delta_4 = (8)^{-1}x(x-2)$ Don't let the 0 value scare you. If you look ahead, you won't actually need to calculate Δ_4 at all.
Putting it all together, we get $P(x) = 4\Delta_0 + 5\Delta_2 + 0\Delta_4$. Solving, you should get $P(x) = x^2 + 2x + 4$.

(b) Consider a function $d(n)$, which takes in a natural number n (in base 10), cubes all of the digits, and adds them up. For example, $d(24) = 2^3 + 4^3 = 8 + 64 = 72$. Prove that $d(n) = n \pmod 3$ for all $n \in \mathbb{N}$.
Given any n, let the decimal representation be $a_k a_{k-1} ... a_1 a_0$. That is, $n = a_k 10^k + a_{k-1}10^{k-1} + ... + a_1 10 + a_0$. Then $d(n) \equiv a_k^3 + a_{k-1}^3 + ... + a_1^3 + a_0^3 \pmod 3$. By Fermat's Little Theorem, since 3 is prime, anything cubed (mod 3) is itself. Then $d(n) \equiv a_k + a_{k-1} + ... + a_1 + a_0$. Now notice that $10 \equiv 1 \pmod 3$, so multiplying anything by 10 would be inconsequential. Similarly, multiply anything by any positive power of 10 would also be okay. Then we can write $d(n) \equiv a_k 10^k + a_{k-1}10^{k-1} + ... + a_1 10 + a_0 = n \pmod 3$.

(c) The engineering team of m people at Quince is about to release their newest model of qPhones. In order to guard against any overly-excited engineers from prematurely releasing the qPhone before the agreed-upon release date, the team decides lock up all of the qPhones in a safe with a secret code. Only the entire engineering team together should be able to open the safe. However, it has been suspected that one of the engineers is really a corporate spy, whose mission is to delay—or even cancel—the release of the qPhone. Devise a mechanism so that the engineering team can protect against premature-releases, while still being able to open the safe on time, even in the presence of a spy.
We use a secret sharing polynomial. This problem is an exact manifestation of general error correcting codes. Recall that in general, if we want n points to interpolate a polynomial of degree $n-1$, if k of those points are errors, then we need $2k$ additional points. In the context of this problem, we have m points, where one of them could be erroneous. This means we need 2 points set aside just for error-correcting purposes. Thus, we have $m-2$ points left, making our polynomial degree $m-3$.