CSE 311: Foundations of Computing I

Homework 5 (due Nov 2nd at 6:00 PM)

Directions: *Write up carefully argued solutions to the following problems. The first task is to be complete and correct. The more subtle task is to keep it simple and succinct. Your solution should be clear enough that it should explain to someone who does not already understand the answer why it works. You may use any results proven in lecture without proof. Anything else must be argued rigorously. Unless otherwise specified, all answers are expected to be given in closed form.*

1. Mod Madness (15 points)

Prove that for any integers a and b and any positive integers c and m,

$$ca \equiv cb \pmod{cm} \text{ if and only if } a \equiv b \pmod{m}.$$

2. GCDs are easier than factoring (10 points)

(a) [1 Point] Compute $\gcd(0, 12^{73})$.

(b) [3 Points] Compute $\gcd(139, 69)$ using Euclid's Algorithm.

(c) [6 Points] Compute $\gcd(91, 434)$ using Euclid's Algorithm. Show your intermediate results.

3. Solveit (20 points)

(a) [5 Points] Compute the multiplicative inverse of 122 modulo 17 using the Extended Euclidean Algorithm. Show your intermediate results.

(b) [5 Points] Find all solutions x with $0 \leq x < 43$ to the following equation:

$$67x \equiv 3 \pmod{43}$$

Show your intermediate results.

(c) [5 Points] Prove that there are no solutions to the following equation:

$$51x \equiv 2 \pmod{141}$$

(d) [5 Points] Find all solutions to:

$$10x \equiv 70 \pmod{135}$$

using the property that you are supposed to prove in Problem 1 above.

4. Modular$^{\text{Exponentiation}^{\text{Question}}}$ (10 points)

Compute $3^{70} \bmod 100$ using the efficient modular exponentation algorithm. Show your intermediate results. How many multiplications does the algorithm use for this computation?

5. Palindromes (15 points)

We say an integer is *palindromic* if the digits read the same when written forward or backward. Prove that every palindromic integer with an even number of digits is divisible by 11.

Hint 1: $10 \equiv -1 (\text{mod } 11)$.
Hint 2: Use the base-10 representation of the number as a summation.

6. An Equality (15 points)

Prove that for every positive integer n, it holds that

$$\sum_{k=1}^{n} k2^k = (n-1)2^{n+1} + 2.$$

7. An Inequality (15 points)

Prove that for all $n \in \mathbb{N}$ and $x \in \mathbb{R}$, $x > -1$: $(1+x)^n \geq 1 + nx$.

8. Extra credit: RSA and modular exponentiation (-NoValue- points)

We know that we can reduce the base of exponent modulo $n : a^k \equiv (a \text{ mod } n)^k (\text{mod } n)$. But the same is not true of the exponent itself! We cannot write $a^k \equiv a^{k \text{ mod } n} (\text{mod } n)$. This is easily seen to be false in general. Consider, for instance, $2^{10} \text{ mod } 3 = 1$, but $2^{10 \text{ mod } 3} \text{ mod } 3 = 2^1 \text{ mod } 3 = 2$.

The correct law for the exponent is more subtle. Define

$$\phi(n) = |\{m \in \mathbb{N} : 1 \leq m \leq n-1, \text{gcd}(m,n) = 1\}|.$$

For instance, if p is a prime, you should check that $\phi(p) = p - 1$. In this problem, you will prove the following.

Theorem: For all integers $n \geq 1$ and $a > 0$ with $\text{gcd}(a,n) = 1$, it holds that $a^{\phi(n)} \equiv 1 (\text{mod } n)$.

Your proof should proceed as follows. Let $R = \{m : 1 \leq m \leq n-1, \text{gcd}(m,n) = 1\}$.

a) Define the set $aR = \{ax \text{ mod } n : x \in R\}$. Prove that $aR = R$ for every integer $a > 0$ with $\text{gcd}(a,n) = 1$.

b) Consider the product of all the elements in R modulo n and the product of all the elements in aR modulo n. By comparing those two expressions, prove carefully that the theorem is true.

c) Using the theorem, prove that under the assumptions, for any $b \geq 0$, we have

$$a^b \equiv a^{b \text{ mod } \phi(n)} (\text{mod } n).$$

This theorem is the fundamental fact underlying the RSA cryptosystem. Let e be such that gcd$(e, \phi(n)) = 1$. Recall that in this case (by Bezout's theorem), we can find a multiplicative inverse, d, to e modulo $\phi(n)$, i.e. such that $ed \equiv 1(\text{mod } \phi(n))$.

e) Suppose you know d. Let $m \in \{0, 1, \ldots, n-1\}$ be a message. Given the value $c = m^e$ mod n, state an efficient algorithm for recovering the initial message m, and argue why it's correct.

Now we'll see how RSA works.

f) Prove that if gcd$(a, b) = 1$ then $\phi(ab) = \phi(a)\phi(b)$ for all positive integers a, b. This shows that if p and q are prime then $\phi(pq) = (p-1)(q-1)$.

Choose two large distinct prime numbers p and q. We'll see in a future homework extra credit how you can generate large primes efficiently. Compute the product $n = pq$ and the value $\phi(n) = (p-1)(q-1)$. Also choose an integer e such that gcd$(e, (p-1)(q-1)) = 1$, and find its multiplicative inverse d modulo $(p-1)(q-1)$. Your **public key** is the pair (n, e) and your private key is the pair (n, d). You release (n, e) to the public and keep (n, d) secret. The idea is now that someone can take a message m and encrypt it to $c = m^e(\text{mod } n)$ using the modular exponentiation algorithm and your public key. You, knowing d, can use part (c) to decrypt the message. This is basically how SSL works: To send your password to your bank, you take their public key and use it to encrypt your password. Then the bank can decrypt your password (to see if you can login), but no eavesdropper sniffing your wifi can do the same. The benefit of this public key cryptosystem is that you don't need a secure channel in order to communicate securely with your bank. (Classical private key cryptosystems would require that you meet ahead of time and exchange a secret key before being able to communicate securely.)

g) Prove that if an attacker knew the factorization $n = pq$ then they would have an efficient algorithm to decrypt the message (and thereby break the cryptosystem).

Presently, the most efficient known way to break RSA is to factor n. Using any known factoring algorithm, this takes an extremely long time (if the primes p, q are chosen large enough initially), and that's what makes RSA secure in practice. But it is not known (i) whether factoring is actually a difficult computational problem, and (ii) whether breaking RSA requires factoring—there could be an easier way to recover the plaintext message.