CSE 311: Foundations of Computing (Autumn, 2015)

Homework 5 Out: Friday, 30-Oct. Due: Friday, 13-Nov, **11: 59 PM** on **Gradescope**

1. Modular exponentiation (8 points)

Compute $9^{85} \bmod 100$ by hand using the modular exponentiation algorithm, showing your intermediate results. How many multiplications does the algorithm use for this computation?

2. Extended Euclidean algorithm (12 points)

Use the extended Euclidean algorithm to find $\gcd(1053, 130000)$, then express it as a linear combination of 1053 and 130000.

3. Modular Congruences (16 points)

(a) Prove that there are no solutions to the modular equation $9x + 10 \equiv 6y - 1 \pmod{15}$

(b) Find an x such that $150\ x \equiv 19 \pmod{1999}$. Show your work.

4. Divisibility by 12 (15 points)

Prove that for any positive integers a, b, c, d,

$$(a-b)(a-c)(a-d)(b-c)(b-d)(c-d)$$

is divisible by 12.

5. Sum of Odd Numbers (10 points)

Use mathematical induction to show that for any positive integer n,

$$1 + 3 + 5 + \cdots + 2n - 1 = n^2$$

6. Taylor Series (15 points)

Use induction to prove that for any real number $-1 \leq x \leq 1$ and any integer $n \geq 1$,

$$1 + x \leq \sum_{i=0}^{n} \frac{x^i}{i!},$$

Recall that $i! = i \cdot (i-1) \cdot (i-2) \ldots 2 \cdot 1$. Also, we assume that $0! = 1$. We remark that the RHS of the above inequality converges to e^x as $n \to \infty$. So, your proof implies that for any $-1 \leq x \leq 1$, $1 + x \leq e^x$ (which is a very useful fact).

7. Binomial Expansion (15 points)

For integers $0 \leq k \leq n$ let $\binom{n}{k} = \frac{n!}{k!(n-k)!}$. For example, if $k = 0$, then $\binom{n}{k} = \frac{n!}{0!n!} = 1$.

Recall that $0! = 1$.

a) Show that for any $1 \leq k \leq n$,

$$\binom{n-1}{k-1} + \binom{n-1}{k} = \binom{n}{k}.$$

b) Use part (a) and induction to prove that for any integer $n \geq 1$ and any real number x,

$$(1+x)^n = \sum_{k=0}^{n} x^k \binom{n}{k}$$

8. Extra credit: Magical Pebbles (20 points)

Consider an infinite sequence of positions $1,2,3,\ldots$ and suppose we have a pebble at position 1 and another pebble at position 2. In each step, we choose one of the pebbles and move it according to the following rule: Say we decide to move the pebble at position i, if the other pebble is not at any of the positions $i+1,\ldots,2i$, then it goes to $2i$, otherwise it goes to $2i+1$.

For example, in the first step, if we move the pebble at position 1, it will go to 3 and if we move the pebble at position 2 it will go to 4. Note that, no matter how we move the pebbles, they will never be at the same position.

Use induction to prove that, for any given positive integer n, it is possible to move one of the pebbles to position n. For example, if $n=7$, first we move the pebble at position 1 to 3. Then, we move the pebble at position 2 to 5. Finally, we move the pebble at position 3 to 7.

9. Extra credit: Rabbits (20 points)

Suppose there are n boxes $B_1, B_2, \ldots, B_n$ in a row. There is a rabbit in one of the boxes that we want to find. In each time step, we open one of the boxes. If the rabbit was in that box we win; otherwise, we close the box and the rabbit jumps from its box to a neighboring box. For example, if the rabbit is in B_n it will be in B_{n-1} in the next step. Give a sequence of boxes to open and prove that by the end of the sequence we find the rabbit no matter where it started. Hint: First, solve the problem in the special case where the rabbit starts in an odd numbered box.

8. Extra credit: RSA and modular exponentiation

We know that we can reduce the base of exponent modulo n: $a^k \equiv (a \bmod n)^k \pmod n$.

But the same is not true of the exponent itself! We cannot write $a^k \equiv a^{k \bmod n} \pmod n$. This is easily seen to be false in general. Consider, for instance, $2^{10} \bmod 3 = 1$, but $2^{10 \bmod 3} \bmod 3 = 2^1 \bmod 3 = 2$.

The correct law for the exponent is more subtle. Define

$$\varphi(n) = |\{\, m \in \mathbb{N} : 1 \le m \le n-1, \gcd(m,n) = 1 \,\}|$$

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

Theorem: For all integers $n \ge 1$ and $a > 0$ with $\gcd(a,n) = 1$, it holds that $a^{\varphi(n)} \equiv 1 \pmod n$.

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

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

(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 $b \ge 0$, we have

$$a^b \equiv a^{b \bmod \varphi(n)} \pmod n.$$

This theorem is the fundamental fact underlying the RSA cryptosystem.

Let e be such that $\gcd\big(e, \varphi(n)\big) = 1$. Recall that in this case (by Bezout's theorem), we can find a multiplicative inverse to d to e modulo $\varphi(n)$, i.e. such that $ed \equiv 1 \big(\bmod \varphi(n)\big)$.

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

Now we'll see how RSA works.

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

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 $\varphi(n) = (p-1)(q-1)$. Also choose an integer e such that $\gcd\big(e, (p-1)(q-1)\big) = 1$, 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 $c = m^e \pmod{n}$ 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.)

(a) 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.