Homework 2

Due Tuesday, September 15

Implementation Part

1. Write an algorthim called `slowPower(g,A,N)` which takes as input natural numbers $N, A \in \mathbb{N}$, and an element $g \in \mathbb{Z}/N\mathbb{Z}$, and returns $g^A \mod N$, that is, the reduction of g^A modulo N, in the following (slow) way

   (1) Set $g_0 = g$

   (2) For each i with $1 \leq i < A$, let $g_i = g_{i-1} * g \mod N$

   (3) return g_A

2. The fast powering algorithm requires computing the binary expansion of a number. In this exercise we will implement `getBinary(A)` which takes a positive integer A and returns the binary expansion of A, that is, a list `[A0,A1,...,Ar]` satisfying

   (i) Each $A_i \in \{0, 1\}$.

   (ii) $A = A_0 + A_1 * 2 + A_2 * 2^2 + \cdots A_r * 2^r$

   (iii) $A_r = 1$

   The algorithm is the following:

   (1) Set $x = A$, and $i = 0$

   (2) Divide x by 2 with remainder giving $x = 2q + r$ for $0 \leq r < 2$.

   (3) Set $A_i = r$ and $x = q$

   (4) If $x = 0$ return $[A_0, A_1, ..., A_r]$. Otherwise set $i = i + 1$ and return to step (ii).

   Think about why this algorithm works and how fast it should be while you're writing it. You will be proving the correctness and analyzing time complexity in the written part of this assignment.

3. With binary expansion in hand we can do fast powering.

   (a) Write an algorthim called `fastPower(g,A,N)` which takes as input natural numbers $N, A \in \mathbb{N}$, and an element $g \in \mathbb{Z}/N\mathbb{Z}$, and returns $g^A \mod N$, that is, the reduction of g^A modulo N. Follow the algorithm outlined in class:

   (1) Compute the binary expansion $[A_0, A_1, .., A_r]$ of A.

   (2) Compute $a_i \equiv g^{2^i} \mod N$ for $0 \leq i \leq r$ by successive squaring. (Notice that $a_{i+1} \equiv a_i^2 \mod N$).

   (3) Compute $g^A \equiv a_0^{A_0} \cdots a_r^{A_r} \mod N$.

   In step (3), rather than multiplying all the $a_i^{A_i}$ together and then reducing, which could produce a very large number that your computer may not want to handle, one should multiply by the next $a_i^{A_i}$ and then reduce $\mod N$ at each step. Notice this doesn't increase the number of multiplications.

As we showed in class, `fastPower` is much better than `slowPower`. Nevertheless, for large A it requires a remembering large amounts of data (including $\log_2(A)$ different powers of g), which is not ideal. This exercise writes a variant which avoids the need for storage. We will need the following definition: if r is a real number the *floor* of r, denoted $\lfloor r \rfloor$ is the largest integer less than or equal to than r (so for example $\lfloor \pi \rfloor = 3$).

(b) Write a function `fastPowerSmall(g,A,N)` taking the same inputs and implementing the following algorithm (from [HPS] Exercise 1.25).

```
(1) Set $a = g$ and $b = 1$.
(2) If $A \equiv 1 \mod 2$ set $b = b \cdot a \mod N$
(3) Set $a = a^2 \mod N$ and $A = \lfloor A/2 \rfloor$.
(4) If $A > 0$ go back to step 2. Otherwise return $b$.
```

Think about why this algorithm works and how fast it should be while you're writing it. You will be proving the correctness and analyzing time complexity in the written part of this assignment.

4. Test out `slowPower`,`fastPower`, and `fastPowerSmall` by using all 3 to compute the following.
   (a) $17^{183} \mod 256$
   (b) $11^{507} \mod 1237$
   (c) $2^{123456789} \mod 987654321$
   (d) The last 4 digits of 5 to the trillionth power.

Written Part

5. Let's begin by proving correcntess of the algorithms from the implementation part.
   (a) Prove the correctness of `getBinary` from Problem 2. Explicitly, prove that the list $[A_0, \cdots, A_r]$ it returns satisfies conditions (i)-(iii).
   (b) Analyze the time of complexity `getBinary`. What is the maximum amount of times it performs division with remainder? Justify your answer.
   (c) Prove that the correctness of `fastPowerSmall` from 3(b). Explicitly, prove that it correctly returns an eliment $b \in \{0, \cdots, N-1\}$ such that $g^A \equiv b \mod N$.
   (d) Analyze the time complexity of `fastPowerSmall`. What is the maximum amount of times it runs steps (2)-(4)? Justify your answer.
   (e) Write a couple of sentences informally discussing the space advanges of `fastPowerSmall` over `fastPower`.

6. Let $\{p_1, \cdots, p_r\}$ be a set of prime numbers and let $a = p_1 p_2 \cdots p_r + 1$. Show that a has a prime divisor which is not equal to any of the p_i. Use this fact to conclude that there are infinitely many prime numbers. (We should point out that this proof appears in Euclid's *Elements*, and is quite literally thousands of years old!)

7. Recall that in the $\mathbb{Z}$, you couldn't always divide by 2, but you could *sometimes* divide by 2. We can explain this phenomenon by saying $2x = b$ has a solution in $\mathbb{Z}$ precisely when b is even. Let's study this problem in the modular setting. Consider the congruence

$$ax = c \mod m.$$

(a) Prove that there is a solution if and only if $\gcd(a, m)$ divides c.

(b) In the case of $\mathbb{Z}$, the solution is unique. Here this need not be the case. Prove that if there is one solution to the congruence, there are precisely $\gcd(a, m)$ many solutions in $\mathbb{Z}/m\mathbb{Z}$.

(c) Check that this works by finding all the solutions to the following congruences:

i. $2x \equiv 4 \mod 6$

ii. $3x \equiv 4 \mod 9$

iii. $12x \equiv 15 \mod 21$

8. This homework assignment concludes with a study of square roots modulo p for some prime p (adapted from Exercises 1.36 and 1.39 in [HPS]). We will see that the behaviour is similar to that of $\mathbb{Z}$, where every number has either 2 square roots, or none at all.

(a) Let p be prime, and suppose $p \neq 2$, and $b \in \mathbb{Z}/p\mathbb{Z}$ a nonzero element. Show that b either has 2 square roots modulo p, or none. That is, show that the congruence

$$x^2 \equiv b \mod p$$

has either 2 or 0 solutions in $\mathbb{Z}/p\mathbb{Z}$. What happens if $p = 2$?

(b) Test out this works by finding all solutions to the following congruences:

i. $x^2 = 2 \mod 7$

ii. $x^2 = 3 \mod 11$

(c) Much like the integers, if a number has a square root modulo p, we call it a *perfect square* modulo p. Find all the perfect squares in $\mathbb{Z}/5\mathbb{Z}$ and $\mathbb{Z}/7\mathbb{Z}$.

Notice that in each case exactly half of the nonzero elements were perfect squares (suggesting that perfect squares are more common mod p than in $\mathbb{Z}$). Let's prove this in general.

(d) Let p be an odd prime and $g \in \mathbb{Z}/p\mathbb{Z}$ be a primitive root. Then any $a \in (\mathbb{Z}/p\mathbb{Z})^*$ is congruent to a power of g modulo p, say $a \equiv g^k \mod p$. Show that a is a perfect square if and only if k is even. Conclude that exactly half of the nonzero elements of $\mathbb{Z}/p\mathbb{Z}$ are perfect squares.

With these tools in hand we can prove the following variant of Fermat's little theorem.

(e) For each prime p such that $3 \leq p < 20$, compute $b^{(p-1)/2} \mod p$ for various nonzero values of $b \in \mathbb{Z}/p\mathbb{Z}$ (for example, $b = 2$). Feel free to use the algorithms you wrote in the implementation part. You should notice a pattern, make a conjecture as to the possible values of $b^{(p-1)/2} \mod p$ for $p \nmid b$, and prove it is correct. (**Hint:** Use Fermat's little theorem and part (a) above.)