

## Primitive Roots

This set of notes is a companion to chapter 4 in the book. As usual, I will skip many things and mention other things not in the book. In this chapter, the book makes extensive use of abstract algebra. I will try NOT to do this in my approach.

We say that a number  $A$  has **order**  $h$  modulo  $m$  if  $A^h \equiv 1 \pmod{m}$ , but  $A^x \not\equiv 1 \pmod{m}$  for any integer  $x$  with  $0 < x < h$ . For example, 2 has order 3 (mod 7) because  $2^3 \equiv 1 \pmod{7}$ , but  $2^1 \equiv 2, 2^2 \equiv 4 \pmod{7}$ . On the other hand, the order of 3 (mod 7) is 6 because  $3^1 \equiv 3, 3^2 \equiv 2, 3^3 \equiv 6, 3^4 \equiv 4, 3^5 \equiv 5, 3^6 \equiv 1 \pmod{7}$ . As a matter of convenience, these are often denoted  $\text{ord}_7(2) = 3, \text{ord}_7(3) = 6$ .

**Theorem.** If  $\text{ord}_m(a) = h$ , and  $a^n \equiv 1 \pmod{m}$ , then  $h \mid n$ .

**Proof.** Let  $n = hq + r$ , where  $0 \leq r < h$ . Then  $a^n = a^{hq} a^r = (a^h)^q a^r \equiv 1^q a^r \equiv a^r \pmod{m}$ . But  $a^n \equiv 1 \pmod{m}$ , so  $a^r \equiv 1 \pmod{m}$ . Since  $r < h$ , and  $h$  is the smallest positive integer such that  $a^h \equiv 1 \pmod{m}$ , it follows that  $r$  cannot be a positive integer. Hence,  $r = 0$ , meaning that  $n = hq$ . That is,  $n$  is divisible by  $h$ , as desired.

Not all numbers have orders. For example, 2 has no order modulo 10, since  $2^h$  is always even, so it can't be congruent to 1 (mod 10). The only numbers that can have orders are those numbers which are relatively prime to  $m$ . Since by Euler's theorem,  $a^{\varphi(m)} \equiv 1 \pmod{m}$  if  $a$  is relatively prime to  $m$ , it follows that every number relatively prime to  $m$  has an order. By the theorem we just proved, a number's order is a divisor of  $\varphi(m)$ . Note that this is consistent with our example above:  $\varphi(7) = 6$ , so the only possible orders (mod 7) are 1, 2, 3, or 6. In fact, each of these orders occurs:  $\text{ord}_7(1) = 1, \text{ord}_7(6) = 2, \text{ord}_7(2) = 3, \text{ord}_7(3) = 6$ .

If  $\text{ord}_m(a) = \varphi(m)$ ,  $a$  is called a **primitive root** modulo  $m$ . Thus, 3 is a primitive root (mod 7). Not all numbers have primitive roots. For example,  $\text{ord}_8(1) = 1, \text{ord}_8(3) = \text{ord}_8(5) = \text{ord}_8(7) = 2$ . Thus, no number has order  $4 = \varphi(8)$ , so 8 has no primitive roots. Gauss was the first to answer the question of which numbers have primitive roots. In fact, he proved the following:

**Theorem.** The only numbers  $n$  that have primitive roots are  $2$ ,  $4$ ,  $p^m$ , where  $p$  is an odd prime, and  $2p^m$ , where  $p$  is an odd prime.

I will try something less ambitious: I will show that all primes have primitive roots. First, we need a few facts.

**Lemma.** If  $\text{ord}_m(a) = h$ , then  $\text{ord}_m(a^n) = \frac{h}{\text{gcd}(h, n)}$ .

**Proof.** Let  $d = \text{gcd}(h, n)$ , and let  $n = dn'$ ,  $h = dh'$ . Then  $(a^n)^{h'} = a^{dn'h'} = (a^h)^{n'}$ . Since  $a^h \equiv 1 \pmod{m}$ , it follows that  $(a^n)^{h'} \equiv 1 \pmod{m}$ . We are not done yet—we must show that  $(a^n)^k \not\equiv 1 \pmod{m}$  for any  $k$  with  $0 < k < h'$ . So suppose that  $a^{kn} \equiv 1 \pmod{m}$ . Then  $h$  divides  $kn$ , so  $dh'$  divides  $kdn'$ . This means that  $h'$  divides  $kn'$ . But  $h'$  and  $n'$  are relatively prime, so  $h'$  divides  $k$ . Consequently,  $0 < k < h'$  is impossible.

**Theorem.** Suppose that  $f(x)$  is a polynomial of degree  $n$  and  $p$  is a prime which does not divide the coefficient of degree  $n$ . Then  $f(x) \equiv 0 \pmod{p}$  has at most  $n$  solutions  $x$  with  $0 \leq x \leq p - 1$ . (That is,  $f(x)$  has at most  $n$  solutions  $\pmod{p}$ .)

**Proof.** We use induction on the degree of  $f$ : if  $f(x)$  has degree 1, then  $f(x) = ax + b$ , and  $ax + b \equiv 0 \pmod{p}$ . We know that  $a$  is not divisible by  $p$ . Thus,  $a$  has a multiplicative inverse,  $c$  and  $x \equiv -bc \pmod{p}$  is the only solution. In general, if the statement is true for polynomials of degree  $n - 1$ , then we step up to degree  $n$  as follows: let  $f(x)$  have degree  $n$ . If it has no solutions  $\pmod{p}$ , that is fine. If  $f(a) \equiv 0 \pmod{p}$ , then dividing  $f(x)$  by  $x - a$ ,  $f(x) = (x - a)g(x) + r$ , for some polynomial  $g(x)$  and some integer  $r$ , and  $f(a) = r$ , so  $r \equiv 0 \pmod{p}$ . Thus,  $f(x) \equiv (x - a)g(x)$  for some polynomial  $g$  of degree  $n - 1$ . If  $b$  is a zero of  $f \pmod{p}$ , and  $b \neq a$ , then  $0 \equiv f(b) \equiv (b - a)g(b) \pmod{p}$ . This means that  $g(b) \equiv 0$ . Now  $g$  has at most  $n - 1$  solutions, so  $f$  has at most  $n - 1 + 1$  (the  $+1$  is for  $x = a$ ) for a total of  $n$ .

**Lemma.**  $\sum_{d \mid n} \varphi(d) = n$ .

What this says is that if you add up  $\varphi$  applied to all divisors of  $n$ , you get  $n$ . For example, if  $n = 20$ , then the possible values for  $d$  are  $1, 2, 4, 5, 10, 20$ . So

$$\sum_{d \mid 20} \varphi(d) = \varphi(1) + \varphi(2) + \varphi(4) + \varphi(5) + \varphi(10) + \varphi(20) = 1 + 1 + 2 + 4 + 4 + 8 = 20.$$

(Note that we need  $\varphi(1) = 1$  in order for this to work.)

**Proof.** Consider the set of fractions  $\left\{ \frac{1}{n}, \frac{2}{n}, \frac{3}{n}, \dots, \frac{n-1}{n}, \frac{n}{n} \right\}$ . There are a total of  $n$  fractions in this set. Put each fraction in lowest terms. For example, if  $n = 20$ , we have  $\left\{ \frac{1}{20}, \frac{2}{20}, \frac{3}{20}, \frac{4}{20}, \frac{5}{20}, \frac{6}{20}, \frac{7}{20}, \frac{8}{20}, \frac{9}{20}, \frac{10}{20}, \frac{11}{20}, \frac{12}{20}, \frac{13}{20}, \frac{14}{20}, \frac{15}{20}, \frac{16}{20}, \frac{17}{20}, \frac{18}{20}, \frac{19}{20}, \frac{20}{20} \right\}$   
 $= \left\{ \frac{1}{20}, \frac{1}{10}, \frac{3}{20}, \frac{1}{5}, \frac{1}{4}, \frac{3}{10}, \frac{7}{20}, \frac{2}{5}, \frac{9}{20}, \frac{1}{2}, \frac{11}{20}, \frac{3}{5}, \frac{13}{20}, \frac{7}{10}, \frac{3}{4}, \frac{4}{5}, \frac{17}{20}, \frac{9}{10}, \frac{19}{20}, \frac{1}{1} \right\}$ .

We could write this as a union:

$$\left\{ \frac{1}{20}, \frac{3}{20}, \frac{7}{20}, \frac{9}{20}, \frac{11}{20}, \frac{13}{20}, \frac{17}{20}, \frac{19}{20} \right\} \cup \left\{ \frac{1}{10}, \frac{3}{10}, \frac{7}{10}, \frac{9}{10} \right\} \cup \left\{ \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5} \right\} \cup \left\{ \frac{1}{4}, \frac{3}{4} \right\} \cup \left\{ \frac{1}{2} \right\} \cup \left\{ \frac{1}{1} \right\}.$$

In general, the denominators consist of the divisors of  $n$ , and for each divisor,  $d$ , of  $n$ , the set with denominator  $d$  will have numerators consisting of those things relatively prime to  $d$ . That is, each set corresponding to  $d$  has size  $\varphi(d)$ . Adding up the sizes of the sets gives  $n$ , so  $\sum_{d|n} \varphi(d) = n$ .

**Theorem.** If  $p$  is a prime, then for each divisor  $d$  of  $p - 1$ , there are  $\varphi(d)$  numbers between 1 and  $p - 1$  that have order  $d \pmod{p}$ . In particular, there are  $\varphi(p-1)$  primitive elements.

**Proof.** Let  $g(d)$  be the number of things  $a$  such that  $\text{ord}_p(a) = d$  and  $1 \leq a \leq p - 1$ . Since  $a^{p-1} \equiv 1 \pmod{p}$  for every  $a$  with  $1 \leq a \leq p - 1$ , by the preceding stuff,  $g(d) = 0$  unless  $d$  is a divisor of  $p - 1$ . Consider the congruence  $x^d \equiv 1 \pmod{p}$ . If  $a$  is an element of order  $d$ , then  $a^d \equiv 1 \pmod{p}$ , so  $a$  is a solution to  $f(x) \equiv 0 \pmod{p}$  where  $f(x) = x^d - 1$ . We know that this polynomial congruence can have at most  $d$  solutions. In fact,  $a, a^2, a^3, \dots, a^d$  are all solutions to this congruence, and they are all different (if  $a^i \equiv a^j$  with  $i < j$ , then  $1 \equiv a^{j-i}$ , and  $\text{ord}_p(a)$  would be  $\leq j - i < d$ .) Since we have listed  $d$  solutions, this is the complete set. Not all of these elements has order  $d$ , however. From a lemma,  $\text{ord}_p(a^k) = \frac{d}{\gcd(d, k)}$ , so only those elements with

exponent relatively prime to  $d$  have order  $d$ . There are exactly  $\varphi(d)$  such exponents, so if there is an element of order  $d$  there are exactly  $\varphi(d)$  of them.

To summarize so far,  $g(d) = 0$ , if  $d$  is not a divisor of  $p - 1$ . If  $d$  is a divisor of  $p - 1$ , it is still conceivable that  $g(d)$  could be 0. But we have shown above that if  $g(d)$  is not 0, then  $g(d) = \varphi(d)$ . Since every number between 1 and  $p - 1$  has some order, and the only possible orders are divisors of  $p - 1$ , we have that  $\sum_{d \mid p-1} g(d) = p - 1$ .

We also have  $\sum_{d \mid p-1} \varphi(d) = p - 1$ . Since  $g(d) \leq \varphi(d)$  for every  $d$ , this can only happen if  $g(d) = \varphi(d)$ . This completes the proof.

As an application of these ideas, we now have a way to **prove** that a number is prime without trial division. Since  $\text{ord}_n(a) \leq \varphi(n) \leq n - 1$  and  $\varphi(n) = n - 1$  only when  $n$  is prime, if we can find an  $a$  with  $\text{ord}_n(a) = n - 1$ , then  $n$  is prime. That is, you can prove that  $p$  is prime by finding a primitive root of  $p$ .

For example, consider  $n = 97$ . We seek a number  $a$  such that  $\text{ord}_{97}(a) = 96$ . The first step is to find a number  $a$  with  $a^{96} \equiv 1 \pmod{97}$ . In fact,  $2^{96} \equiv 1 \pmod{97}$ . (If we had found that  $2^{96} \not\equiv 1 \pmod{97}$ , then we would have known that 97 was not prime!) Next, try to show that  $\text{ord}_{97}(2) = 96$ . We know that  $\text{ord}_{97}(2)$  must be a divisor of 96. This gives the following possibilities: 1, 2, 3, 4, 6, 8, 12, 16, 24, 32, 48, 96. This is a long list, but there is a trick: if  $\text{ord}_{97}(2)$  is not a divisor of 48 or 32, then 96 is the only possibility left. The reason for this is that every divisor of 96 is either a divisor of 48 or a divisor of 32 or it is 96. So we need only check  $2^{32} \pmod{97}$  and  $2^{48} \pmod{97}$ . We have  $2^{32} \equiv 35 \pmod{97}$ , but  $2^{48} \equiv 1 \pmod{97}$ . In fact, it turns out that  $\text{ord}_{97}(2) = 48$ . So we failed! This is common. It means that 2 is not a primitive root. Next, we try 3. We find, again, that  $3^{48} \equiv 1 \pmod{97}$ . However,  $5^{32} \equiv 35 \pmod{97}$ , and  $5^{48} \equiv 96 \pmod{97}$ . This forces  $\text{ord}_{97}(5) = 96$ . That is, 5 is a primitive root, and we have proved that 97 is prime.

We can generalize this idea:

**Theorem.** Let  $p$  be a prime, and suppose  $p \nmid a$ . If for every prime divisor  $q$  of  $p - 1$ ,  $a^{(p-1)/q} \not\equiv 1 \pmod{p}$ , then  $a$  is a primitive root module  $p$ .

**Proof.** If  $a$  is not a primitive root (mod  $p$ ), then the order of  $a$  is a proper divisor of  $p - 1$ . Thus, if  $a$  has order  $d$ , then  $dk = p - 1$  for some integer  $k > 1$ . If  $q$  is a prime divisor of  $k$ , then  $d$  is a divisor of  $\frac{p-1}{q}$ , so  $a^{(p-1)/q} \equiv 1 \pmod{p}$ , contradicting the hypothesis of the theorem. Thus, the order of  $a$  cannot be a proper divisor of  $p - 1$  meaning that the order must equal  $p - 1$ .

**Corollary.** If  $n$  is a positive integer, and  $a$  has the property that  $a^{n-1} \equiv 1 \pmod{n}$ , but for each divisor  $q$  of  $n - 1$ ,  $a^{(n-1)/q} \not\equiv 1 \pmod{n}$ , then  $n$  is prime.

**Proof.** As before, these conditions force the order of  $a$  to be  $n - 1$ . Since the order of  $a$  can't be any larger than  $\varphi(n)$ ,  $\varphi(n) = n - 1$ , so  $n$  must be prime.

How does one go about finding a primitive root? If 2 is not a primitive root, then neither is  $2^2$  or  $2^3$ ,  $\dots$ . It is possible for 6 to be a primitive root even though neither 2 nor 3 is, however. If you believe the Extended Riemann Hypothesis, then every prime  $p$  has a primitive root  $a < 2(\ln(p))^2$ .

There is one other issue, however, in proving that a number  $p$  is prime: our method of proving that some number  $a$  is a primitive root of  $p$  requires us to produce a factorization of  $p - 1$ . If  $p$  is a very large number, factoring  $p - 1$  can be very hard. For example, consider the number  $2^{329} + 39$ . Maple reports that this number is probably prime. Suppose we would like to prove that it is prime. By the above, we would need to factor  $p - 1 = 2^{329} + 38$ . Using Maple's `ifactor` routine, we get an answer like:

```
> ifactor(p - 1, easy);
(2) (5)2 (7)*(11) _c92 (5101)
```

Here, I used the modifier "easy" so that Maple would not do too much work. It reports back what it found "easily" (using trial division up to some point, followed by a number of steps of the  $p - 1$  method) and says that the unfactored part has 92 digits. Moreover, the `_c92` indicates that it know this 92-digit number is composit. Factoring a general 92 digit number is beyond Maple's ability with any algorithm.

### The discrete logarithm problem.

Primitive roots have applications to cryptography: Suppose that  $g$  is a primitive root (mod  $p$ ). Then every number between 1 and  $p - 1$  is a power of  $g$ . For example, 5 is a primitive root (mod 23), and the powers of 5 (mod 23) are:

x	1	2	3	4	5	6	7	8	9	10	11
$5^x$	5	2	10	4	20	8	17	16	11	9	22
x	12	13	14	15	16	17	18	19	20	21	22
$5^x$	18	21	13	19	3	15	6	7	12	14	1

From this chart, we see that  $5^9 \equiv 11 \pmod{23}$ , etc. Also, if someone asks for a solution to  $5^x \equiv 15 \pmod{23}$ , then we can look at the chart and see that  $x = 17$  works. The general solution to this congruence would be  $x \equiv 17 \pmod{22}$ : Since  $5^{22} \equiv 1$ , multiples of 22 in the exponent do not affect things.

This second question, in general form: given a primitive root  $g \pmod{p}$ , and a number  $a$  with  $1 \leq a \leq p - 1$ , find  $x$  so that  $g^x \equiv a \pmod{p}$  is called the discrete logarithm problem. The reason for this is that the action is taking place in the exponent. If we were talking about real numbers, and wanted to solve  $g^x = a$ , we would simply take a logarithm. By analogy, we think of this congruence question as a discrete version of a logarithm.

In general, if  $p$  is large, solving  $g^x \equiv a \pmod{p}$  is thought to be a hard problem, just as the problem of factoring a large number is thought to be a hard problem. It should be pointed out that just because people think the problem is hard does not make it hard. You could become famous by finding an easy way to solve the general problem  $g^x \equiv a \pmod{p}$  for  $x$ .

When ever a problem is considered hard, people try to use it to generate cryptosystems. To see why, we introduce some cryptographic terminology. (I talked about most of this stuff in class already, but I'll give you a formal write up here.) A function  $f(x)$  from a set  $S$  to a set  $T$  is called a one-way function if

- 1) it is one-to-one (different values for  $x$  give different values for  $f$ ),

- 2)  $f(x)$  is “easy” to calculate,
- 3) for most values of  $a$ , it is “hard” to solve  $f(x) = a$  for  $x$ .

Related to a one-way function is a trapdoor function. A trapdoor function is a function  $f(x)$  for which it is hard to solve  $f(x) = a$  without extra knowledge about  $f$ .

An example of a trapdoor function is the one used in the RSA cryptosystem: given a large number  $n$ , known not to be prime, and an integer  $e$  relatively prime to  $\varphi(n)$ ,  $f(x) \equiv x^e \pmod{n}$ . In this case, if you have the additional knowledge that  $n = pq$  for two primes  $p$  and  $q$  (that is, if you know how  $n$  factors), then you can solve the congruence  $ed \equiv 1 \pmod{\varphi(n)}$ , and the solution to  $f(x) = a$  is simply  $a^d \pmod{n}$ . Trapdoor functions are required for public key cryptosystems—you want the entire world to know how to send you a coded message, but you want to be the only one able to decode the message.

The difference between a trapdoor function and a one-way function is that there is no (known) easy way to solve  $f(x) = a$  for  $x$  with a one-way function. That is, being “in the know” will not help speed up the calculation. One-way functions have a very common use in computer systems: when a system administrator stores passwords for the users on a network, it would be nice if you didn’t have to worry about someone hacking into the password file and uncovering all the passwords. One way to do this is to not store the passwords, but to store instead, a one-way function applied to the passwords. That is, convert a password,  $w$ , to a number, and then store  $f(w)$ . When someone logs on and gives their password, then what you do is compare  $f(\text{password})$  to  $f(w)$ . If it matches, you figure the person knows the password. Since the file contains  $f(w)$  instead of  $w$ , the only way for someone to get the password from  $f(w)$  is to solve the problem  $f(x) = a$ , where  $a$  is what is recorded in the password file. Both UNIX and VMS operating systems use this idea.

Where as the function  $f(x) = x^e \pmod{n}$  is a trapdoor function, the function  $f(x) = g^x \pmod{p}$  is thought to be an honest one-way function. Here are some cryptosystems that have been built out of this function.



3136526413619143163891401357990519975209295258784426004049708368537465242.

Your key is now

$K = 3136526413619143163891401357990519975209295258784426004049708368537465242^{12345}$   
 $(\text{mod } p) =$

8927797676014428152574530503316966596961070897823134511524228318154909024,  
 which your friend calculates via

$4303677725596168664586776887051985961310865957112980711082389439927639988^{11223}$   
 $(\text{mod } p).$

(I did verify that they are the same using Maple!). You now send  $M + K (\text{mod } p) =$   
 3616907665965514041571639986108042486358870108772045431104167802043011886,

which your friend translates by subtracting  $K (\text{mod } p).$

### The ElGamal cryptosystem

Again, we fix a large prime  $p$  and a primitive element  $g$ . Pick a random number  $a$  with  $0 < a < p - 1$ , and calculate  $K = g^a (\text{mod } p)$ . To send a message,  $M$  to someone, choose a random integer  $k$  and send the pair  $(g^k, Mg^{ak})$ . If your friend knows  $a$ , then they can recover the message as follows:  $g^{ak} = (g^k)^a$ , and given  $g^{ak}$ , you can find its multiplicative inverse,  $x (\text{mod } p)$ , and get  $M$  by  $M \equiv M g^{ak} x (\text{mod } p)$ . This system requires that you and your friend both know about  $a$ , but the rest of the world does not.

### The Massey–Omura cryptosystem

As before, we start by having two people agree on a large prime  $p$  with primitive root  $g$ . Next, you pick a random number  $a$  such that  $\text{gcd}(a, p - 1) = 1$ , and solve  $ab \equiv 1 (\text{mod } p - 1)$  for  $b$ . Your friend picks a random number  $c$  relatively prime to  $p - 1$  and solves  $cd \equiv 1 (\text{mod } p - 1)$  for  $d$ . Given a message,  $M$ , send your friend  $M^a (\text{mod } p)$ . Your friend cannot translate this message. Your friend calculates  $(M^a)^c \equiv M^{ac} (\text{mod } p)$ , and sends this back to you. You take this new message and calculate  $(M^{ac})^b \equiv M^{abc} \equiv (M^a)^c \equiv M^c (\text{mod } p)$ . You send this back, and finally, your friend calculates  $(M^c)^d \equiv M^{cd} \equiv M (\text{mod } p)$ , to get the message.

This technique works because  $M^{p-1} \equiv 1 \pmod{p}$ , so  $M^{ab} \equiv M$  and  $M^{cd} \equiv M$ . At no time can anyone reading the transmissions crack the code, but the end result is that your friend gets the message without knowing  $a$  or  $b$ .

### Big step, Little step

Here is one method for solving  $a \equiv g^x \pmod{p}$  for  $x$ . Let  $m = \lfloor \sqrt{p-1} \rfloor$ . Then  $x = qm + r$ , where  $0 \leq q \leq m$ , and  $0 \leq r < m$ . Thus,  $g^x = (g^m)^q g^r$ , and we can find  $x$  if we can find  $q$  and  $r$ . We can do this as follows: Compute  $g^m, g^{2m}, \dots, g^{m^2}$ . This is called the big step. Now if we calculated  $g, g^2, g^3, \dots, g^{m-1}$ , and looked at all products,  $a$  would have to appear on the list. Unfortunately, this takes roughly  $p$  steps, so it is no better than calculating  $g, g^2, g^3, \dots, g^{p-1}$ . However, there is a trick: Calculate  $g^{-1} \pmod{p}$ . That is, calculate  $y$  such that  $gy \equiv 1 \pmod{p}$ . This can be done with the Euclidean algorithm. Next, calculate  $ay, ay^2, ay^3$ , and compare them against the list of powers of  $g^m$ . Once you get a match, say  $ay^r = (g^m)^q$ , then we have found  $q$  and  $r$ , and thus,  $x$ .

For an example, let's find  $x$  so that  $2^x \equiv 25 \pmod{37}$ . Here,  $\sqrt{36} = 6$ , so  $m = 6$ . we have  $2^6 \equiv 27, 2^{12} \equiv 26, 2^{18} \equiv 36, 2^{24} \equiv 10, 2^{30} \equiv 11, 2^{36} \equiv 1 \pmod{37}$ . In this case,  $2^{-1} \equiv 19$  (we can use a trick:  $2^{-1} \equiv \frac{p+1}{2} \pmod{p}$ ). In general, this trick only works for  $g = 2$ , unfortunately.) Now,  $25$  is not on the "big" list,  $25 \cdot 19 \equiv 31$ , which is not on the list,  $31 \cdot 19 \equiv 34, 34 \cdot 19 \equiv 17, 17 \cdot 19 \equiv 27$ , which is on the list. Thus,  $25 \cdot 2^{-4} \equiv (2^6)^1$ , which says that  $25 \equiv 2^{6 \cdot 1 + 4}$ , so  $x = 9$ . This algorithm takes at most  $2m$  steps, with an average of  $m$  steps needed. Thus, we can solve  $a \equiv g^x \pmod{p}$  for  $x$  in roughly  $\sqrt{p}$  steps. There are other, more elaborate schemes for solving for  $x$ , but none of them is particularly easy.

To give credence to the idea that the discrete logarithm problem is hard, here are some extra credit problems, each worth 3 points. Each is of the same flavor: I give you a prime  $p$ , a primitive root  $g$ , and  $g^x$  for some  $x$ , and you get the extra credit if you can figure out what  $x$  is.

1.  $p = 83$ ,  $g = 2$ ,  $g^x \equiv 27 \pmod{83}$  (Ok, this should be doable! To get credit, use the big step–little step method.)
2.  $p = 10^{10} + 19$ ,  $g = 2$ ,  $g^x = 111$
3.  $p = 10^{15} + 37$ ,  $g = 2$ ,  $g^x = 274897353777826$
4.  $p = 10^{20} + 39$ ,  $g = 3$ ,  $g^x = 58818535957436841843$