Number theory is the branch of mathematics that studies properties of the integers. It is full of conjectures (propositions that have not been proven and have not been refuted) that are very simple to state, but extremely difficult to resolve. We'll stick to a lighter side of number theory that gives us a nice playground to exercise the concepts we just learned – logic, proofs, induction, invariants. We'll also see how these concepts help us figure out correctness and termination of algorithms.

1 Die hard, once and for all

You are standing next to a water source. You have two empty jugs: A 3 litre jug and a 5 litre jug with no marks on them. You must fill the larger jug with precisely 4 litres of water. Can you do it?

This scenario is straight out of the 1995 classic "Die Hard 3: With a Vengeance" featuring Bruce Willis and Samuel L. Jackson. Should they fail to complete this task within 5 minutes, a bomb goes off and New York City is obliterated. In the nick of time, Bruce and Samuel come up with a solution:

small jug	large jug	action
—	—	fill up large jug from source
—	5ℓ	top up small jug from large jug
3ℓ	2ℓ	empty small jug
—	2ℓ	pour large jug contents into small jug
2ℓ	—	fill up large jug from source
2ℓ	5ℓ	top up small jug from large jug
3ℓ	4ℓ	done!

The rumour is that in the sequels, Bruce and Samuel will be asked to measure — always 4 litres — with a 21 litre and a 26 litre jug; with an 899 litre and 1,147 litre jug; and finally, in the last movie of the series, "Die once and for all", with a 3 litre and 6 litre jug. What should they do? To help them out, let us first come up with a mathematical model of their problem.

The Die Hard state machine The water jug game can be naturally described by a state machine. The states are pairs of non-negative integers (x, y) describing the amount of water in the first and second jug, respectively. The initial state is $(0, 0)$. If a is the capacity of the first jug and b is the capacity of the second jug, the possible states are the pairs

$$(x, y) \text{ such that } 0 \leq x \leq a \text{ and } 0 \leq y \leq b.$$

The transitions specify the rules for pouring water. Let us specify them clearly — once and for all:

1. **Empty out a jug:** The emptied out jug now contains 0 litres, while the contents of the other jug stay the same. Symbolically,

$$(x, y) \to (0, y) \tag{T1}$$
$$(x, y) \to (x, 0) \tag{T2}$$

2. **Fill up a jug from the source:** The filled up jug now contains as much water as its capacity. The contents of the other jug stay the same. Symbolically,

$$(x,y) \to (a,y) \tag{T3}$$
$$(x,y) \to (x,b) \tag{T4}$$

3. **Transfer water between jugs:** Water can be poured from one jug into the other until one of them becomes full or empty. These transitions are a bit trickier to write down, but the effort will pay off later on.

   If water is poured from the first into the second jug then two things can happen: Either the total amount of water does not exceed the capacity b of the second jug, in which case the second one will contain $x+y$ litres:

$$(x,y) \to (0, x+y), \quad \text{if } x+y \le b, \tag{T5}$$

   or else, the second jug fills up to its capacity b, and $x+y-b$ litres of water remain in the first jug:

$$(x,y) \to (x+y-b, b), \quad \text{if } x+y > b. \tag{T6}$$

   We have analogous transitions for pouring water from the second jug into the first one:

$$(x,y) \to (x+y, 0), \quad \text{if } x+y \le a, \tag{T7}$$
$$(x,y) \to (a, x+y-a), \quad \text{if } x+y > a. \tag{T8}$$

We can now formulate the water jug game with a v litre target question mathematically: Is there a finite sequence of transitions that reaches a state (x,y) in which $x=v$ or $y=v$? To get some intuition about when this is possible, let us start by looking for some invariants.

Greatest common divisors and integer combinations

Let a, b be two integers. We say a *divides* b if $b = ak$ for some integer k. For example, 2 divides 4 and -6 but 2 does not divide 5; 4 does not divide 2; and every number divides zero.

The *greatest common divisor (GCD)* of two integers a, b is the largest integer k so that k divides a and k divides b. We write $\gcd(a,b)$ for the GCD of a and b. For example, $\gcd(2,6)=2$, $\gcd(4,6)=2$, $\gcd(3,5)=1$.

A number c is an *integer combination* of a and b if there exist integers s and t such that $c = s\cdot a + t\cdot b$. For example,

- 2 is an integer combination of 2 and 6 because $2 = 1\cdot 2 + 0\cdot 6$.
- 2 is an integer combination of 4 and 6 because $2 = (-1)\cdot 4 + 1\cdot 6$.
- 1 is an integer combination of 3 and 5 because $1 = 2\cdot 3 + (-1)\cdot 5$.

It looks like GCDs and integer combinations are closely related. This is not an accident.

Lemma 1. *For all integers a and b, $\gcd(a,b)$ divides every integer combination of a and b.*

Proof. Let c be an integer combination of a and b. Then we can write

$$c = s\cdot a + t\cdot b \tag{1}$$

for some integers a and b. Since $\gcd(a,b)$ divides a, we can write $a = a' \cdot \gcd(a,b)$ for some integer a'. Since $\gcd(a,b)$ divides b, we can write $b = b' \cdot \gcd(a,b)$ for some integer b'. Substituting into (1) this gives

$$c = s \cdot a' \cdot \gcd(a,b) + t \cdot b' \cdot \gcd(a,b) = (sa' + tb') \cdot \gcd(a,b).$$

Therefore $\gcd(a,b)$ divides c. □

An invariant

We can now state an invariant about the die hard problem:

Lemma 2. *The amount of water in each jug is always an integer combination of a and b.*

Proof. We will prove that the invariant holds in the initial state and that it is preserved by the transitions. It then follows by induction that the invariant holds after an arbitrary number of steps.

Initial state: Initially, each jug has 0 liters of water and $0 = 0 \cdot a + 0 \cdot b$.

Transitions: Assume the invariant holds in state (x,y). namely both x and y are integer combinations of a and b. We show that this remains true after any possible transition:

T1: $(x,y) \to (0,y)$. Both 0 and y are integer combinations of a and b.

T2: $(x,y) \to (x,0)$. Both x and 0 are integer combinations of a and b.

T3: $(x,y) \to (a,y)$. Both a and y are integer combinations of a and b.

T4: $(x,y) \to (x,b)$. Both x and b are integer combinations of a and b.

T5: $(x,y) \to (0,x+y)$. Zero is an integer combination of a and b. As for $x+y$, we need to do a small calculation. If $x = s_1 a + t_1 b$ and $y = s_2 a + t_2 b$ for some integers s_1, t_1, s_2, t_2 then we can write

$$x + y = (s_1 + s_2)a + (t_1 + t_2)b$$

which is also an integer combination of a and b.

T6: $(x,y) \to (x+y-b, b)$. b is an integer combination of a and b. As for $x+y-b$, we can write

$$x + y - b = (s_1 + s_2)a + (t_1 + t_2 - 1)b$$

which is an integer combination of a and b.

T7, T8: These are identical to T5 and T6 with the roles of the jugs flipped.

In all cases, the amount of water in each jug remains an integer combination of a and b after the transition. □

Combining Lemma 14 and Lemma 1, we obtain a useful corollary:

Corollary 3. $\gcd(a,b)$ *always divides the amount of water in each jug.*

In particular, a 3 litre jug and a 6 litre jug can never measure 4 litres of water:

Corollary 4. *In "Die once and for all", Bruce dies.*

2 Euclid's algorithm

Euclid's algorithm is a procedure for calculating the GCD of two positive integers. It was invented by the Euclideans around 3,000 years ago and it still the fastest procedure for calculating GCDs. To explain Euclid's algorithm, we need to recall division with remainder.

Theorem 5. *Let n and d be integers with $d > 0$. There exists unique integers q and r such that*

$$n = q \cdot d + r \quad \textit{and} \quad 0 \le r < d.$$

For example, if $n = 13$ and $d = 3$, we can write $13 = 4 \cdot 3 + 1$. Moreover, $q = 4$ and $r = 1$ are unique assuming r is in the range $0 \le r < d$.

The numbers q and r can be calculated by the usual "division with remainder rule" that you learned in school.

Proof. First, we show existence: Let q be the largest integer such that $qd \le n$. That means $(q+1)d > n$. Then $0 \le n - qd < d$. Set $r = n - qd$.

Now we show uniqueness: Suppose we can write n in the desired form in two different ways:

$$n = qd + r, 0 \le r < d$$
$$n = q'd + r', 0 \le r' < d.$$

Then $qd + r = q'd + r'$, so $(q - q')d = r' - r$. Therefore $r' - r$ divides d. Since $0 \le r, r' < d$ we must have $-d < r' - r < d$. The only number in this range that divides d is zero, so $r' - r = 0$ and $q' - q = 0$. It follows that $q' = q$ and $r' = r$, so the two representations are the same. □

Euclid's algorithm is a *recursive* algorithm for calculating the GCD of positive integers.

```
Euclid's algorithm $E(n, d)$, where $n, d$ are integers such that $n > d \ge 0$.
    If $d = 0$, return $n$.
    Otherwise, write $n = qd + r$ where $0 \le r < d$. Return $E(d, r)$.
```

Let's apply Euclid's algorithm to calculate the GCD of 1147 and 899:

$$\begin{aligned}
E(1147, 899) &= E(899, 248) && \text{because } 1147 = 1 \cdot 899 + 248 \\
&= E(248, 155) && \text{because } 899 = 3 \cdot 248 + 155 \\
&= E(155, 93) && \text{because } 248 = 1 \cdot 155 + 93 \\
&= E(93, 62) && \text{because } 155 = 1 \cdot 93 + 62 \\
&= E(62, 31) && \text{because } 93 = 1 \cdot 62 + 31 \\
&= E(31, 0) && \text{because } 62 = 2 \cdot 31 + 0 \\
&= 31.
\end{aligned}$$

How can we be sure that Euclid's algorithm indeed outputs the GCD of n and d? How can we even be sure that Euclid's algorithm *terminates*? This is a theorem that we will have to prove:

Theorem 6. *For every pair of integers n, d such that $n > d \ge 0$, $E(n, d)$ terminates and outputs* $\gcd(n, d)$.

To prove this theorem, we'll need a little lemma.

Lemma 7. *For all integers a, b, and t,* $\gcd(a,b) = \gcd(a+tb, b)$*.*

Proof of Theorem 11. We will prove the theorem by strong induction on d.

Base case: When $d = 0$, the algorithm returns $n = \gcd(n, 0)$ and terminates.

Inductive step: Now assume the algorithm terminates and outputs $\gcd(n, d')$ for all input pairs (n, d'), where $n > d'$ and $0 \le d' \le d$. Consider what the algorithm does on input $(n, d+1)$: It writes $n = q(d+1)+r$ with $0 \le r < d+1$ and returns $E(d+1, r)$. By our inductive assumption, $E(d+1, r)$ must terminate since r is between 0 and d and $E(d+1, r)$ outputs $\gcd(r, d+1)$. Therefore, $E(n, d+1)$ must terminate as well. Moreover, $E(n, d+1)$ outputs $\gcd(r, d+1) = \gcd(n - q(d+1), d+1)$. By Lemma 7, this number equals $\gcd(n, d+1)$. $\square$

Proof of Lemma 7. We will show that, for every integer k, k divides a and b if and only if k divides $a+tb$ and b. In particular, this means that the *largest* divisors of the two pairs of numbers – namely, their GCDs – must be equal.

First, assume k divides both a and b. Then we can write $a = a'k$ and $b = b'k$ for integers a', b'. We get that $a + tb = (a' + tb')k$, so k also divides $a' + tb'$.

For the other direction, assume k divides both $a + tb$ and b and write $a + tb = a'k$ and $b = b'k$ for integers a', b'. Then $a = a'k - tb'k = (a' - tb')k$, so k also divides a. $\square$

3 How to solve any water jug game

Corollary 3 is *sufficient* condition for Bruce to die: If $\gcd(a, b)$ is not 1, 2, or 4, then Bruce dies. This happens in the 3 litre + 6 litre scenario, as well as the 1147 litre + 899 litre one. What if we have a 21 litre and a 26 litre jug? The GCD of 21 and 26 is 1, so Corollary 3 does not rule out the possibility that Bruce may survive. But can he, actually, survive?

The key to survival is the following converse to Lemma 1.

Lemma 8. *Let a, b be any two nonnegative*[1] *integers. Then* $\gcd(a, b)$ *can be written as an integer combination of a and b.*

Corollary 9. *Let a, b be any two positive integers. Then there exist nonnegative integers s and t such that* $\gcd(a, b) = s \cdot a - t \cdot b$*.*

Proof of Corollary 9. By Lemma 8, there exist integers s and t such that $\gcd(a, b) = s \cdot a + t \cdot b$. Then for every integer k,

$$(s + kb) \cdot a + (t - ka) \cdot b = sa + tb = \gcd(a, b).$$

No matter what the values of s and t are, when k is sufficiently large, $s + kb$ is positive and $t - ka$ is negative. This gives us a representation of $\gcd(a, b)$ of the desired form. $\square$

For example, if $a = 21$ and $b = 26$, then $\gcd(21, 26) = 1$ and we can write

$$1 = 5 \cdot 21 - 4 \cdot 26.$$

[1]The assumption that the integers are nonnegative is not important; in fact, it is not necessary, but we will make it because it makes the arguments a bit simpler.

Multiplying both sides by 4, we get

$$4 = 20 \cdot 21 - 16 \cdot 26.$$

Now here is how Bruce can handle a 21 litre and a 26 litre jug: Keep filling up the 21 litre jug for a total of 20 times. Whenever it becomes full, pour all the water into the 26 litre jug. If that one becomes full, empty it out and continue pouring into it.

At the end, the 26 litre jug will contain exactly 4 litres. Here is why: We poured a total of $20 \cdot 21$ litres from the source, and some multiple of 26 litres out of the larger jug, so the amount of water left in the larger jug is of the form $20 \cdot 21 - t \cdot 26$. Let's see what these numbers look like:

t	$\cdots$	14	15	16	17	18	$\cdots$
$20 \cdot 21 - t \cdot 26$	$\cdots$	56	30	4	-22	-48	$\cdots$

So there *must* be 4 liters left in the 26 liter jug: This is the only number in the list which is within the capacity of the jug!

We can now specify the exact conditions under which the water jug game has a survivor.

Theorem 10. *Assume $a \le b$. Bruce lives if and only if $\gcd(a, b)$ divides v and $0 \le v \le b$.*

Proof. Assume Bruce lives. By Corollary 3, $\gcd(a, b)$ must divide v. Clearly v must also be within the size of the larger bin.

Now assume $\gcd(a, b)$ divides v and $0 \le v \le b$. The cases $v = 0$ and $v = b$ are easy, so from here on we'll assume $0 < v < b$. By Corollary 9 we can write

$$\gcd(a, b) = sa - tb$$

for some nonnegative integers s, t. Since v divides $\gcd(a, b)$, we can write $v = k\gcd(a, b)$ for some k and

$$v = (ks)a - (kt)b.$$

To obtain v litres, we proceed like this: Keep filling jug A for a total of ks times. Whenever it becomes full, pour all the water into jug B. If that one becomes full, empty it out and continue pouring into it.

Suppose jug B was emptied a total of t' times. Then the amount of water left inside jug B in the end is

$$(ks)a - t'b = (ks)a - (kt)b + (kt - t')b = v + (kt - t')b$$

If $kt - t' < 0$, this number is negative; if $kt - t' > 0$, then it is larger than b. In either case, this is not a valid amount of water to be left in jug B. It must be that $kt - t' = 0$ and there are exactly v litres left in jug B. □

In fact, we do not need to fill jug A exactly ks times; we can interrupt the procedure as soon as we get 4 litres into jug B. I wrote a program that implements this strategy. You can play with it.

We are almost done – all that remains is to prove Lemma 8. This follows by analysing an extension of Euclid's algorithm, which we won't do here. The purpose of the extended Euclid's algorithm is to compute the numbers s and t from Lemma 8.

```
Extended Euclid's algorithm $X(n, d)$, where $n, d$ are integers such that $n > d \ge 0$.
    If $d = 0$, return $(1, 0)$.
    Otherwise,
        Write $n = qd + r$ where $0 \le r < d$.
        Calculate $(s, t) = X(d, r)$.
        Return $(t, s - q \cdot t)$.
```

Lemma 8 is now a consequence of the following Theorem:

Theorem 11. *For every pair of integers n, d such that $n > d \ge 0$, $X(n, d)$ terminates and outputs a pair of integers (s, t) such that $s \cdot n + t \cdot d = \gcd(n, d)$.*

Instead of proving this theorem, let us show on an example how s and t can be calculated by extending the reasoning we used for Euclid's algorithm. Let $n = 73$ and $d = 50$. Euclid's algorithm goes through the following steps:

$$\begin{aligned}
E(73,50) &= E(50,23) && \text{because } 73 = 1 \cdot 50 + 23 && \text{(E1)}\\
&= E(23,4) && \text{because } 50 = 2 \cdot 23 + 4 && \text{(E2)}\\
&= E(4,3) && \text{because } 23 = 5 \cdot 4 + 3 && \text{(E3)}\\
&= E(3,1) && \text{because } 4 = 1 \cdot 3 + 1 && \text{(E4)}\\
&= 1.
\end{aligned}$$

We want to express 1 as an integer combination of 73 and 50. We start with the equation (E4), which writes 1 as an integer combination of 4 and 3 and work backwards. We take (E4) and (E3) and cancel out the 3 which occurs in both. To do this, we scale the (E3) by -1 on both sides and add (E4) to it to get

$$-1 \cdot 23 + 4 = -5 \cdot 4 + 1.$$

We can rewrite this as

$$-1 \cdot 23 = -6 \cdot 4 + 1, \tag{2}$$

which is a representation of 1 as an integer combination of 23 and 4. We can now cancel out the 4 by combining (E2) and (2): After multiplying (E2) by 6 on both sides and subtracting (2) we get

$$6 \cdot 50 - 1 \cdot 23 = 12 \cdot 23 + 1,$$

from where

$$6 \cdot 50 = 13 \cdot 23 + 1, \tag{3}$$

which is a representation of 1 as an integer combination of 50 and 23. Finally, the 23 disappears if we multiply (E1) by -13 and subtract (3) to obtain

$$-13 \cdot 73 + 6 \cdot 50 = -13 \cdot 50 + 1$$

which gives us the desired integer combination

$$1 = -13 \cdot 73 + 19 \cdot 50.$$

4 Prime numbers

I am sure you already know quite a bit about prime numbers, so let me just state two useful facts. If you want to read the proofs, you can find them in the textbook.

Lemma 12. *If p is a prime and p divides $a \cdot b$, then p divides a or p divides b.*

Theorem 13 (Fundamental theorem of arithmetic)**.** *Every positive integer n can be written uniquely as a product of primes.*

For example, $15 = 3 \cdot 5$, $8 = 2 \cdot 2 \cdot 2$, $12 = 2 \cdot 2 \cdot 3$, and $17 = 17$.

5 Modular arithmetic

Let's fix an integer p, which we will call the *modulus*. Theorem 5 says that for every integer n there exists a unique integer between 0 and $p-1$ such that $n = qp + r$ for some q. We call r the *remainder* of n *modulo* p — in short n modulo p — and we write

$$r = n \bmod p.$$

We can do arithmetic using the numbers $0, 1, \dots, p-1$: additions, subtractions, multiplications, divisions. The arithmetic remains valid as long as we take the remainders of all expressions modulo p. Taking remainders all the time is tedious so instead we'll take advantage of the concept of congruence: Two integers m and n are *congruent* modulo p if p divides $m - n$. We write $m \equiv n \pmod{p}$ for "m and n are congruent modulo p." Congruences behave nicely with respect to additions and multiplications:

$$\text{If} \quad x \equiv x' \pmod{p} \quad \text{and} \quad y \equiv y' \pmod{p}, \qquad \begin{array}{l} \text{then} \quad x + y \equiv x' + y' \pmod{p} \\ \text{and} \quad x \cdot y \equiv x' \cdot y' \pmod{p}. \end{array}$$

Addition and subtraction Addition and subtraction modulo p is fairly easy: For example, if I want to calculate $3 + 8$ modulo 9 first I add 3 and 8 as integers to get 11 then I take the remainder of 11 modulo 9 to get 2:

$$(3 + 8) \bmod 9 = 11 \bmod 9 = 2.$$

The *additive inverse* of a modulo p is the number $(-a) \bmod p$. Say the modulus is 5. Then the additive inverse of 0 is 0, the additive inverse of 1 is 4, the additive inverse of 2 is 3, and vice versa. Subtraction can then be done by replacing each negative number with its additive inverse.

$$(3 - 7) \bmod 8 = (3 + 1) \bmod 8 = 4 \bmod 8 = 4.$$

Multiplication and division From here on we will assume that the modulus p is a prime number. (If it isn't, some of the following statements won't be true.) Just like addition, to multiply two numbers modulo p, first we multiply them as integers and then we take the remainder modulo p, for example

$$3 \cdot 4 \bmod 7 = 12 \bmod 7 = 5.$$

What about division? First, let's show how to calculate *multiplicative inverses*, namely ratios of the form $1/x$. This is more commonly written as x^{-1}. Division by zero is forbidden, but otherwise we have a nice lemma:

Lemma 14. *For every x between 1 and $p-1$ there exists a unique y between 1 and $p-1$ such that $xy \equiv 1 \pmod{p}$.*

Proof. First we show existence of y. Take any x between 1 and $p-1$. Since p is prime, $\gcd(x, p) = 1$. By Lemma 8 there exist integers s, t for which $s \cdot x + t \cdot p = 1$. Taking both sides modulo p, we get that $s \cdot x \equiv 1 \pmod{p}$. Take $y = s \bmod p$. Then $y \cdot x \equiv s \cdot x \pmod{p}$, so $y \cdot x \equiv 1 \pmod{p}$.

Now we show uniqueness. We just saw that for every x between 1 and $p-1$ there is *at least one* y in the same range such that $xy \equiv 1 \pmod{p}$. Now we show there is at most one such y. Suppose $xy \equiv 1 \pmod{p}$ and $xy' \equiv 1 \pmod{p}$. Then $x(y - y') \equiv 0 \pmod{p}$, so p divides $x(y - y')$. By Lemma 12, p divides x or p divides $y - y'$. Since $x < p$, p must divide $y - y'$. But $y - y'$ is a number between $-(p-2)$ and $p-2$, so this is only possible if $y = y'$, namely $y - y' = 0$. □

The proof of Lemma 14 not only tells us that x^{-1} exists, but also how to calculate it: First, use the Extended Euclid's algorithm to find s and t such that $s \cdot x + t \cdot p = 1$. Then output $s \bmod p$. For example, to get the multiplicative inverse of 11 modulo 17, I run $X(11, 17)$ to get $(-3, 2)$, namely $(-3) \cdot 11 + 2 \cdot 17 = 1$. Therefore

$$11^{-1} \bmod 17 = -3 \bmod 17 = 14.$$

To divide two numbers, we first take the multiplicative inverse of the denominator, then multiply by the numerator, for instance to divide 3 by 11 modulo 17 we calculate

$$3 \cdot 11^{-1} \bmod 17 = 3 \cdot 14 \bmod 17 = 42 \bmod 17 = 8.$$

6 Encryption*

Encryption is a type of mechanism that allows Alice and Bob to exchange private messages in a public environment, like a cellular network, or the internet.

Here is a simple protocol for doing this called the one-time pad. Say Alice wants to send Bob the message "`Charlie is a spy`". This message is represented as a large number, say by replacing each character with its ASCII representation:

$$m = \texttt{067104097114108105101032105115032097032115112121}$$

Alice and Bob have agreed ahead of time on a number p, which is at least as large as the message m to be encoded. In this example, $p = 10^{50}$ would suffice and exchanged in secret a random number k between 0 and $p - 1$ called the *secret key*.

If Alice wants to send Bob the secret message m, she *encrypts* m as $c = m + k \bmod p$ and posts the *ciphertext* c in public. To recover Alice's message, Bob calculates $c - k \bmod p$.

I will now argue that the content of Alice's message remains secret with respect to any observer Eve who sees the ciphertext but does not know the secret key. This requires a bit of elementary probability. Let's assume that all the numbers in the range $0, 1, \dots, p - 1$ were equally likely to be chosen as the secret key k. Then, regardless of what the message m is, the ciphertext $m + k \bmod n$ observed by Eve is also equally likely to take any of the values $0, 1, \dots, p - 1$. (For example, if $m = 1$ and $p = 4$, then k is equally likely to take any of the values 0, 1, 2, and 3, so $m + k \bmod p$ is equally likely to evaluate to 1, 2, 3, and 0.) Therefore the value observed by Eve is completely independent of the message sent by Alice, and Eve obtains no information about the message.

This type of encryption works as long as Alice and Bob have previously met to exchange the secret key k (and kept it secure in the meantime). Such an assumption is unreasonable if, say, Alice is a customer in Bob's online store and she wants to send him her credit card number in secret. Alice and Bob could be living on opposite sides of the world and might have never talked to one another before. How can they agree on a secret key in such circumstance?

Before we describe one proposed solution to this problem we need to discuss a bit more modular arithmetic, specifically the operation of powering and its inverse, the *discrete logarithm*

Powering One way to calculate $b^n \bmod p$ is by performing repeated modular multiplication n times:

$$b^n \bmod p = \underbrace{((b \cdot b \bmod p) \cdot b \bmod p) \cdots b \bmod p}_{n \text{ times}}.$$

This works well in practice when the exponent n is not too large, but is very slow when n is on the order of, say, 10^{50}. A better way to power is to take advantage of the following recursive formula for modular exponentiation:

$$b^n \bmod p = \begin{cases} (b^2 \bmod p)^{n/2} \bmod p, & \text{if } n \text{ is even} \\ b \cdot (b^2 \bmod p)^{(n-1)/2} \bmod p, & \text{if } n \text{ is odd} \end{cases}$$

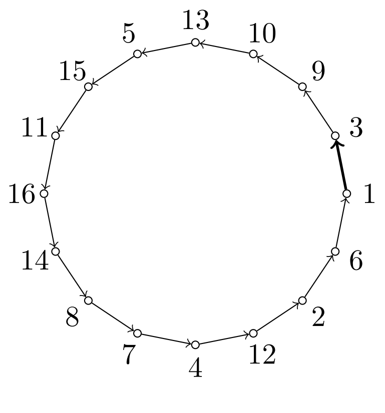
which is valid for all $n \geq 1$. This reduces the task of calculating the n-th power of a number to the task of calculating the $n/2$-th or $(n-1)/2$-th power of some other number (depending on the parity of n), and at most three modular multiplication (one to calculate $b^2 \bmod p$ and possibly another one by b). If we implement this calculation by a computer program, the exponent n will be at worst halved in every step of the recursion, so the total number of recursive calls will not exceed $\log_2 n$. It is therefore possible to calculate $b^n \bmod p$ using at most $3 \log_2 n$ modular multiplications. If n is on the order of 10^{50} then the number of multiplications is about 500, which is a huge improvement over the 10^{50} required by the naive powering procedure.

Discrete logarithm When we work with real numbers, the operation inverse to powering is taking logarithms. By analogy, we call an integer x a *discrete logarithm* of y in base b modulo p if $b^x \bmod p = y$. Unlike ordinary logarithms, discrete logarithms do not behave well at all: Some numbers may have multiple logarithms, and some may have none at all. In these notes we will focus on the situation when p is a prime number, which is already complicated enough.

Let's take the modulus $p = 17$ as an example. Then 10 has no discrete logarithm in base 9. On the other hand, 13 has (at least) two base 9 logarithms as both 9^2 and 9^{10} equal 13 modulo 17. The key to understanding the discrete logarithm is the following theorem which we won't prove:

Theorem 15. *For every prime number p there exists a number g between 1 and $p-1$ such that the sequence $g^0, g^1, \ldots, g^{p-2}$ covers every nonzero number modulo p exactly once.*

Such a number g is called a *primitive root* modulo p. For example, $g = 3$ is a primitive root modulo 17. In the following diagram, arrows indicate multiplication by 3. You can see that the sequence $1, 3, 3^2, \ldots, 3^{15}$ covers every nonzero number modulo 17 exactly once.



Notice also that $3^{16} = 1 \pmod 17$, so calculating $3^x \pmod 17$ for an arbitrary x reduces to calculating $3^{x \pmod 16} \pmod 17$ for every x. This is not an accident:

Corollary 16. *For every prime p and every nonzero a modulo p, $a^{p-1} = 1 \bmod p$.*

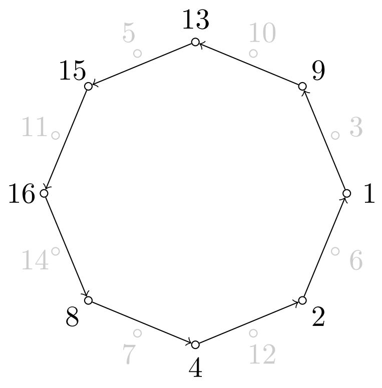
Proof. First assume a is a primitive root g. Since the sequence $1, g, g^2, \ldots, g^{p-2}$ covers every nonzero number modulo p, g^{p-1} must equal g^i for some i between 0 and $p-2$. Dividing both sides by g^i

we obtain that $g^{p-1-i} = 1$. The only possibility is that $i = 0$ and so $g^i = 1$; otherwise, $p - 1 - i$ indexes some entry in the sequence other than the first one, so g^{p-1-i} cannot equal 1.

If a is not a primitive root, then it must be a power of a primitive root, namely $a = g^j$ for some j. But then $a^{p-1} = (g^j)^{p-1} = (g^{p-1})^j = 1^j = 1 \bmod p$, as desired. □

Corollary 16 tells us in particular that a discrete logarithm can always be represented as a nonzero number modulo $p-1$. Theorem 15 tells us that if the base is a primitive root g then every nonzero number modulo $p-1$ has a unique discrete logarithm in base g. In the above diagram, the discrete logarithm of x base 3 (modulo 17) is the number of steps it takes to reach x starting at 1.

If the base b is not a primitive root, then the powers of b will not cover all the nonzero numbers modulo p, so not every number will have a discrete logarithm in base b. For example, if $b = 9$ then the numbers that have discrete logarithms modulo 17 are the ones reachable from 1 in the following diagram (where the arrows indicate multiplication by 9):



Notice also that $9^8 = 1 \bmod 17$, so the value of the base 9 discrete logarithm (if it exists) reduces to a unique value between 0 and 7, and not between 0 and 15 as was the case in base 3. This is no accident as $9 = 3^2$ so the powers of 9 are the even powers of 3 modulo 17.

Calculating discrete logarithms Let us now assume that the base is a primitive root g modulo p and see how we could go about calculating the discrete logarithm of some input y. One way to do so is to look for the first occurrence of y in the sequence $1, g, g^2, \ldots, g^{p-2}$ modulo p and output the index of this occurrence. This works fine when p is of moderate size, but would be unimaginably slow when p is, say, on the order of 10^{50}. Is there a better way?

One possible strategy would be to begin by looking for some partial information about the discrete logarithm of y — for example, is it even or is it odd? The following theorem tells us that this information can be extracted by calculating a suitable power of g:

Theorem 17. *Assume $p > 2$ is a prime number, g is a primitive root modulo p, and y is a nonzero number modulo p. Then the discrete logarithm of y in base g modulo p is even if and only if $y^{(p-1)/2} = 1 \bmod p$.*

Proof. We prove the theorem by cases depending on the parity of the discrete logarithm of y. If it is even, then $y = g^{2n}$ for some k, and $y^{(p-1)/2} = (g^{p-1})^n = 1 \bmod p$ by Corollary 16. If it is odd, then $y = g^{2n+1}$ for some n, and $y^{(p-1)/2} = (g^{p-1})^n \cdot g^{(p-1)/2} = g^{(p-1)/2} \bmod p$ by Corollary 16. By Theorem 15, $g^{(p-1)/2}$ cannot be equal to 1 modulo p. □

Let's illustrate this on our example $p = 17, g = 3$. Theorem 17 says that the discrete logarithm of y in base g is even if and only if $y^8 = 1$. The numbers that satisfy this condition are exactly those that are at an even distance from 1 in the diagram: Moving forward by the same even amount 8 times always brings us back to 1, but if the amount is odd we must end up in a different place.

Once we know if the discrete logarithm of y in base g is even, one natural idea is to try recursion and find out the parity of the higher-order bits of the discrete logarithm n. To develop some intuition about how this strategy might work, let's fix $p = 17, g = 3$, and look for the discrete logarithm of 14. We first calculate $14^8 \bmod 17 = 16$, so the discrete logarithm of 14 must be even. In other words, we now know that $n = 2n' + 1$ and so

$$14 = 3^{2n'+1} \quad \text{for some } n' \text{ between 0 and 7.}$$

We can rewrite this equation as

$$14 \cdot 3^{-1} = (3^2)^{n'} \quad \text{for some } n' \text{ between 0 and 7.}$$

We have now reduced our problem of finding the discrete logarithm of 14 in base 3 to the problem of finding the discrete logarithm of $14 \cdot 3^{-1} \equiv 16$ in base $3^2 = 9$! The number 9 is, unfortunately, no longer a primitive root modulo 17 so we cannot apply Theorem 17 again. We can, however, apply the following *generalization*, which you should be able to prove on your own by applying the same reasoning as in the proof of Theorem 17

Theorem 18. *Assume $p > 2$ is a prime number, b is a nonzero number modulo p, and k is the smallest positive integer such that $b^k = 1 \pmod p$. Assume that k is even and y has a discrete logarithm in base b modulo p. The discrete logarithm is even if and only if $y^{k/2} = 1 \bmod p$.*

Theorem 18 tells us that the base 9 discrete logarithm of 16 is even if and only if $16^4 = 1$ modulo 17. This is indeed the case, so we can conclude that n' is even and so we can write $n' = 2n''$, which tells us that

$$16 = 9^{2n''} \quad \text{for some } n'' \text{ between 0 and 3.}$$

So we are left with finding the discrete logarithm n'' of 16 in base $9^2 = 13 \bmod 16$. Continuing in the same manner, we find that the parity of n'' is determined by the value $16^2 \bmod 17$, which again equals 1. Therefore n'' itself is even, and so we must have $16 = 13^{2n'''}$ for $n''' = n''/2$, which is either zero or one. Now $13^2 \pmod 1)7 = 16$, so it follows by inspection that $n''' = 1$, from where $n'' = 2$, $n' = 4$, and $n = 9$.

In general, this strategy for finding discrete logarithm is much faster than brute-force search through the list of powers as the range of possible discrete logarithms is halved at each iteration. But were we lucky with our example here, or can it be used in a general context? One seemingly innocuous assumption that we made in Theorem 18 is that the number k, which represents the *period* of the sequence $1, b, b^2, \ldots$ must be even. When the modulus p is of the form $2^t + 1$ (as in our example) then the period of the base will remain even in all iterations because it has initial value 2^t and it is halved in every iteration. Prime numbers of this form are called *Fermat primes*, and indeed discrete logarithms modulo a Fermat prime can be calculating reasonably efficiently using the algorithm that we informally described.

Efficient calculation of discrete logarithms can, more generally, be extended to work modulo any prime p provided that the number $p - 1$ does not contain any "large" prime factors.[2] Many prime numbers p have these property, but there are also many that don't. How large can a prime factor of $p - 1$ be when p is prime? Assuming p is greater than 2, $p - 1$ must be an even number, so

[2]More precisely, the running time of the discrete logarithm procedure has a linear dependence on the size of the largest prime factor of $p - 1$.

its largest prime factor cannot exceed $(p-1)/2$. A prime number p for which $(p-1)/2$ is also a prime number are called a *safe prime*. It is not known if the number of safe primes is infinite, but very large examples are known. Wikipedia says that "Finding a 500-digit safe prime, like $2 \cdot (1,416,461,893 + 10^{500}) + 1$, is now quite practical."

Unlike in the case of Fermat primes, it is not known how to calculate discrete logarithms modulo a safe prime in an efficient manner. For a number like the Wikipedia safe prime, the best known algorithms may require 10^{160} or so steps.

Key exchange We are now ready to describe a famous proposal by which Alice and Bob can exchange a key that looks secret to any observer using only public communication channels like the internet. First, Alice and Bob agree (in public) on a safe prime p and choose a primitive roof g modulo p. For example, they could take p to equal the Wikipedia safe prime and g to equal 2 (I checked that 2 is a primitive root on my computer). They then exchange the following public messages:

A: Choose a random nonzero number x modulo p (in private). Send the value $a = g^x$ to Bob.

B: Choose a random nonzero number y modulo p (in private). Send the value $b = g^y$ to Alice.

A: Upon receiving b from Bob, calculate b^x. This is Alice's secret key.

B: Upon receiving a from Alice, calculate a^y. This is Bob's secret key.

At the end of the protocol, Alice and Bob, they will both agree on the same secret key g^{xy}.

Now consider a potential observer Eve that has seen Alice's and Bob's messages. This Eve has seen the values g^x and g^y. Can she use these to gain any knowledge about the secret key g^{xy}? One way for Eve to do this is to calculate the discrete logarithms x and y, in which case she can easily compute the secret key. But calculating discrete logarithms modulo safe primes is very time-consuming.

It is, in fact, not known how to calculate g^{xy} efficiently given only knowledge of the values g^x and g^y. This gives Alice and Bob some confidence that their secret key is indeed secret, even though all their messages were exchanged in public.

The actual protocols used for key exchange and secure communication used on the internet are a bit more complex than the one I described here. One of the aspects in which the protocol I described may be deficient is that even though Eve can conceivably not completely recover Alice's and Bob's secret key, she might still be able to recover some partial information about it that may allow her to intercept future messages exchanged between them. If you want to know more about this fascinating subject, you can look at the book *Introduction to Modern Cryptography* by Jonathan Katz and Yehuda Lindell.

References

This lecture is based on Chapters 6 and 9 of the text *Mathematics for Computer Science* by E. Lehman, T. Leighton, and A. Meyer.