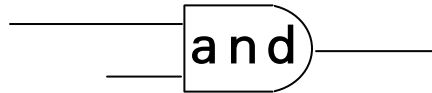


Mathematics for the Digital Age



Programming in Python

>>> Second Edition:
with Python 3

Maria Litvin

Phillips Academy, Andover, Massachusetts

Gary Litvin

Skylight Software, Inc.

Skylight Publishing
Andover, Massachusetts

Skylight Publishing
9 Bartlet Street, Suite 70
Andover, MA 01810

web: <http://www.skylit.com>
e-mail: sales@skylit.com
support@skylit.com

**Copyright © 2010 by Maria Litvin, Gary Litvin, and
Skylight Publishing**

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the authors and Skylight Publishing.

Library of Congress Control Number: 2009913596

ISBN 978-0-9824775-8-8 (soft cover)
ISBN 978-0-9824775-4-0 (hard cover)

The names of commercially available software and products mentioned in this book are used for identification purposes only and may be trademarks or registered trademarks owned by corporations and other commercial entities. Skylight Publishing and the authors have no affiliation with and disclaim any sponsorship or endorsement by any of these products' manufacturers or trademarks' owners.

1 2 3 4 5 6 7 8 9 10 15 14 13 12 11 10

Printed in the United States of America

17 Number Theory and Cryptology

17.1 Prologue

Number theory is a broadly defined branch of mathematics that deals with properties of numbers, especially integers. Many number theoretical concepts are easy to grasp: prime numbers, greatest common divisor, remainder, and so on. But some theorems are very difficult. You have probably heard of Fermat's Last Theorem, which states that the equation $a^n + b^n = c^n$ has no positive integer solutions for $n > 2$. This theorem was proposed by Pierre de Fermat in the 1630s. Fermat left a note in the margin of his copy of Diophantus's *Arithmetica* stating that he had a marvelous proof of this theorem, but the margin was too small to write it down. Three and a half centuries of futile attempts to prove the theorem led to many advances in number theory. Finally, Andrew Wiles of Princeton University presented a proof in 1993. Wiles' proof relied on earlier results that linked Fermat's Last Theorem to the properties of a certain class of elliptic curves. The proof took over seven years to complete and three lectures to present.*

In this chapter we will only scratch the surface. We will start with Euclid's algorithm for finding the greatest common divisor, which will lead us to the fundamental theorem of arithmetic. We will take that opportunity to present a mini-theory, with definitions, theorems, and proofs, to give you a taste of such things. We will then consider arithmetic operations on remainders and applications of number theory to cryptology, the science of ciphers.

17.2 Euclid's Algorithm

Given two integers, a and $d \neq 0$, we say that a is evenly divisible by d (or, simply, that a is divisible by d) if $a = qd$ for some integer q . We can also say that d divides a or that d is a *divisor* of a .

We will consider only positive divisors: $d > 0$.

* It took Wiles another year and some help from his former student Richard Taylor to fill a gap discovered in his proof.

The notation $d \mid a$ means d divides a .

If $d \mid a$ and $d \mid b$, then d is a *common divisor* of a and b . $\text{GCD}(a, b)$ stands for the greatest common divisor of a and b . (Sometimes, the greatest common divisor is called the *greatest common factor*, GCF.) Finding the GCD of two numbers is a common mathematical operation; it is used, for example, for reducing fractions.

You can find the $\text{GCD}(a, b)$ for positive a and b by simply trying every number d , starting from 2 and up to a or b , whichever is smaller, and testing whether d is a common divisor. Such a “brute-force” approach, however, gets pretty tedious for big numbers.

Over 2300 years ago, in Book VII of his *Elements*, Euclid described an efficient algorithm for finding the GCD. Euclid’s algorithm is based on the following key observation: if $a > b > 0$, then $\text{GCD}(a, b) = \text{GCD}(a - b, b)$. (We leave the proof of this fact to you — see Question 1.) This allows us to go from a and b to smaller numbers, $a - b$ and b . We repeat this operation several times, each time choosing the larger of a and b to reduce. Sooner or later we come to the situation where $a = b$. Then, of course, $\text{GCD}(a, b) = a = b$.

Example 1

Using Euclid’s algorithm, find $\text{GCD}(18, 30)$

Solution

$$\text{GCD}(18, 30) = \text{GCD}(18, 12) = \text{GCD}(6, 12) = \text{GCD}(6, 6) = 6$$

Example 2

Write a Python function that uses Euclid’s algorithm to find and return the greatest common divisor of two positive integers.

Solution

```
def gcd(a, b):
    while a != b:
        if a > b:
            a -= b
        else:
            b -= a
    return a          # or return b
```

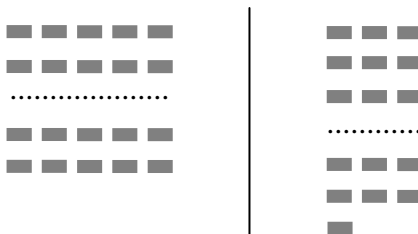


Two integers a and b are called *relatively prime* if they have no common divisors except 1, that is $\text{GCD}(a, b) = 1$.

The concept of relatively prime numbers and the idea of Euclid's algorithm help us analyze and solve equations of the type $ax + by = c$. Here a , b , and c are given integers, and a solution is a pair of integers x and y . This type of equation is called a *linear Diophantine equation in two variables*. In general, a polynomial equation in one or several variables is called a *Diophantine equation* if we are looking only for its integer solutions. Such equations are named after Diophantus, a Greek mathematician who lived in Alexandria in the 3rd century AD and studied equations with integer solutions. The first descriptions of linear Diophantine equations are found much earlier, in Indian texts that are 2800 years old.

Example 3

A classroom has several desks. When we arrange the desks in rows of 5, all the rows are complete, but when we arrange the desks in rows of 3, one desk remains:



How many desks are there in the classroom, if their number is between 15 and 30?

Solution

If D is the number of desks, then $D = 5x$ and $D = 3y + 1$, where x and y are some positive integers. Then $5x = 3y + 1$, which leads to the Diophantine equation $5x - 3y = 1$.

This equation is easy to solve. We know that D is divisible by 5. We also know that $D \geq 15$. We start counting by 5 from 15: 15, 20, ... — and soon get the answer: $D = 25$. Indeed, when 25 is divided by 3, the remainder is 1. Here $x = 5$ and $y = 8$. If we keep going — 30, 35, ... — we soon get another number with the same properties, $D = 40$, but it is out of the given range.

Another way to solve this puzzle is to start from 1 and count by 3 until we get a number evenly divisible by 5 and within the 15-30 range: 1, 4, 7, 10, 13, 16, 19, 22, 25.



If the equation $ax + by = c$ has a solution, then it has infinitely many solutions, because if an x, y pair is a solution, then the $x + b, y - a$ pair is also a solution, the $x + 2b, y - 2a$ pair is a solution, and so on.

Does an equation $ax + by = 1$, where a and b are integers, always have a solution? Does $6x - 4y = 1$, for example, have a solution? Of course not! The left side must be an even number, so it can't be 1. What about $14x + 21y = 1$? Here $a = 14, b = 21$, and they have a common divisor 7. Therefore, the left side is always divisible by 7, so it can't be equal to 1. In general, if a and b have a common divisor greater than 1, then the equation $ax + by = 1$ has no solutions. What if a and b have no common divisors other than 1?

Linear Diophantine Equation Theorem

An equation $ax + by = 1$, where a and b are integers, has an integer solution if and only if a and b are relatively prime.

Proof:

We have already shown that if a and b have a common divisor greater than 1, then the equation has no solutions. Now let's assume that a and b are relatively prime and show that a solution exists.

Note that 0 is not relatively prime with any number, so we must have $a \neq 0, b \neq 0$. Without loss of generality we can assume that a and b are positive: if they are not, we can adjust the sign of x and/or y accordingly. For example, $ax + by = 1$ can be rewritten as $ax + (-b)(-y) = 1$. Also, a and b can't be equal, unless $a = b = 1$.

The main idea of the proof is the same as in Euclid's algorithm. Suppose $a > b$. Then $ax + by = 1$ can be rewritten as $(a - b)x + b(x + y) = 1$ or $(a - b)x_2 + by_2 = 1$. The original equation has a solution if and only if the new equation, with the coefficients $(a - b)$ and b , has a solution. $(a - b)$ and b are still positive and relatively prime. If we repeat this procedure several times, always subtracting the smaller coefficient from the larger one, we get smaller and smaller coefficients, until

we come to $a = 1, b = 1$. Obviously the equation $x + y = 1$ has solutions (for example, $x = 1, y = 0$ or $x = 2, y = -1$). So the original equation must have a solution, too. Q.E.D.

Figure 17-1 illustrates the above proof for the equation $5x - 3y = 1$.

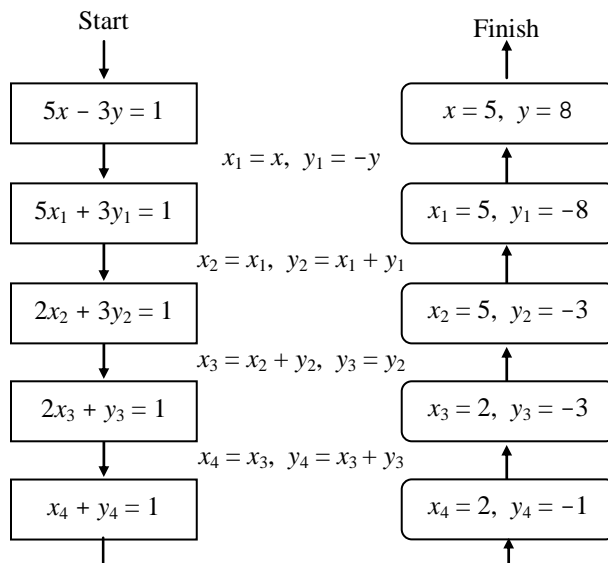


Figure 17-1. Finding a solution of a linear Diophantine equation by repeatedly reducing its coefficients

Exercises

1. Show that if $a > b > 0$, then $\text{GCD}(a, b) = \text{GCD}(a - b, b)$. ✓
2. Write a Python function `gcd(a, b)` that calculates the greatest common divisor of two positive integers a and b by using a “brute-force” approach: just check whether both a and b are divisible by d for all d from 2 to $\min(a, b)$. ⚡ Hint: it is more efficient to go from $\min(a, b)$ down to 1. ⚡
3. Rewrite the `gcd` function from Example 2 recursively, without loops.

- 4.▪ The gcd function in Example 2 can be made more efficient. If $a > b$, instead of subtracting b from a multiple times, we can replace a with the remainder of the division of a by b (using Python's modulo division operator). The same for $b > a$. Write and test this more efficient version of the gcd function. ⚡ Hint: make sure your code acts properly when a becomes divisible by b or b becomes divisible by a . ⚡ Compare the running times for the original and the modified code for $a = 18289894500228625200$, $b = 14814814692$.
5. Show that if a and b are relatively prime, then the equation $ax + by = c$ has an integer solution for any c . ✓
- 6.▪ Show that $ax + by = c$ has an integer solution if and only if c is divisible by $\text{GCD}(a, b)$.
7. Write and test a function that finds a solution of the equation $ax + by = 1$ for given a and b and returns it as a tuple. Use the approach of Example 3.
8. Modify the function from Question 7 to find the solution of $ax + by = 1$ with the smallest possible positive integer x . Write another version of this function that returns the solution with x in a given range.
9. Rewrite the function from Question 7 recursively. Use the approach outlined in the proof of the linear Diophantine equation theorem.
10. Show that if p and q are two different primes, any arithmetic sequence with the common difference p has a term that is divisible by q .
- 11.▪ Show that there is no prime that divides all Fibonacci numbers, starting from a certain place in the sequence.
12. Is it true that if $\text{GCD}(a, b) = 1$ or $\text{GCD}(b, c) = 1$ or $\text{GCD}(c, a) = 1$, then the greatest common factor of a, b and c is 1? If so, prove it; if not give a counterexample. Is the converse true? If so, prove it; if not give a counterexample.
- 13.♦ Formulate and prove the necessary and sufficient condition for the equation $ax + by + cz = 1$ to have an integer solution. ✓

17.3 The Fundamental Theorem of Arithmetic

In this section we present a mini-theory that leads to the fundamental theorem of arithmetic. We present it in the typical mathematical style with definitions and theorems. To make our theory complete, we restate here the definitions from Section 17.2.

Definition 1:

Let n be an integer. A positive integer d is called its *divisor* if there exists an integer q such that $n = qd$.

Definition 2:

An integer p is called a *prime* if $p > 1$ and p has no divisors other than 1 and p .

Definition 3:

Two integers m and n are called *relatively prime* if they have no common divisors greater than 1.

Theorem 1:

Any integer greater than 1 is either a prime or can be represented as a product of primes.

Proof:

This is a proof by mathematical induction.

Base case: The statement is true for $n = 2$, because 2 is a prime.

Induction hypothesis: Suppose that the statement is true for any integer greater than 1 and less than n . Now let's show that the statement is then true for n . Indeed, if n is a prime, the statement is true for n . If n is not a prime, then $n = qd$, where $1 < d < n$ and $1 < q < n$. By the induction hypothesis, d is a prime or a product of primes. The same for q . Combining the products for q and d into one product we conclude that n is a product of primes, too. By mathematical induction, the statement is true for any $n \geq 2$, Q.E.D.

A slightly different take on this proof relies on the fact that any non-empty set of positive integers has a smallest number. (This fact is equivalent to math induction.) Consider the set of all numbers greater than 1 for which the statement is not true. If this set is empty, our proof is complete. If not, let n be its smallest element. n can't be a prime, so $n = qd$, where $1 < d < n$ and $1 < q < n$. Since n is the smallest number for which the statement is false, the statement must be true for q and d . But if we factor both d and q into primes, and combine them into one product, we get n . So the statement must also be true for n . This is a contradiction, so no such n can exist.



A corollary is a mathematical fact that easily follows from a theorem.

Corollary:

There are infinitely many primes.

Proof:

Suppose the set of all primes were finite: p_1, p_2, \dots, p_n . Consider $n = p_1 \cdot p_2 \cdot \dots \cdot p_n + 1$. By Theorem 1, n must be a prime or have a prime divisor. That prime must be different from any of the p_1, p_2, \dots, p_n because none of the p_i is a divisor of n . This is a contradiction, so our assumption that the set of all primes is finite was false.

Theorem 2:

If integers a and b are relatively prime, then there exist integers x and y , such that $ax + by = 1$.

Proof:

This is a part of the linear Diophantine equation theorem from Section 17.2.

Theorem 3:

If p is a prime and $p \mid mn$, then $p \mid m$ or $p \mid n$.

This is Proposition 30 in Book VII of Euclid's *Elements*. It is also known as *Euclid's First Theorem*. The statement seems completely obvious, but its proof is not obvious at all. This is common in mathematics.

Proof:

This is a proof by contradiction. Suppose p divides neither m nor n . Then p and n are relatively prime (because p is a prime). By Theorem 2, there exist integers x and y such that $px + ny = 1$. Multiplying both sides by m we get $mpx + mny = m$. The left side is divisible by p , because $p \mid mn$. So m must be divisible by p , too. Therefore, our assumption that p divides neither m nor n was false. Q.E.D.

Corollary:

If p is a prime and $p \mid n_1 \cdot \dots \cdot n_k$, then $p \mid n_1$ or $p \mid n_2$ or ... or $p \mid n_k$.

Proof:

By Theorem 3, $p \mid n_1$ or $p \mid n_2 \cdot \dots \cdot n_k$. etc.



We are now ready to tackle the fundamental theorem of arithmetic.

Theorem 4 (the fundamental theorem of arithmetic):

Any integer greater than 1 can be represented as a product of primes, and such factorization is unique (if we disregard the order of factors).

Proof:

By Theorem 1, any integer greater than 1 is either a prime or a product of primes. Now we have to show that such factorization is unique. Suppose there exist numbers with two different factorizations. Let's take the smallest such number $n = p_1 \cdot \dots \cdot p_i = q_1 \cdot \dots \cdot q_j$. $q_1 \mid n$, so $q_1 \mid p_1 \cdot \dots \cdot p_i$. By Theorem 3, q_1 must be one of the primes p_1, \dots, p_i . Therefore, we can divide $p_1 \cdot \dots \cdot p_i$ and $q_1 \cdot \dots \cdot q_j$ by q_1 and get a smaller number n/q_1 with two different factorizations. This is a contradiction, so no such n can exist.

Exercises

1. Prove that for any positive integer k there are k consecutive positive integers, such that none of them is a prime. \Leftarrow Hint: start at $(k+1)! + 2$. \Rightarrow
2. 7, 13, 19 form an arithmetic sequence and they are all primes. Show that an infinite arithmetic sequence cannot contain only primes. \Leftarrow Hint: see Question 1. \Rightarrow
3. \blacksquare Write and test a program that prompts the user to enter an integer n greater than 1 and prints all its prime factors in ascending order. Each factor must be printed as many times as it appears in the factorization of n . The dialog with the program should look like this:

```
Enter an integer greater than 1: 90
90 = 2*3*3*5
```

or

```
Enter an integer greater than 1: 3
3 = 3
```

4. If p_1, p_2, \dots, p_n are the first n primes, is it always true that $p_1 \cdot p_2 \cdot \dots \cdot p_n + 1$ is a prime? If you believe it is true, prove it. If not, find a counterexample.
5. Write a program to find the smallest positive n such that $n^2 - n + 41$ is not a prime. (There is no polynomial, even in several variables, that produces only prime values.)

6. Show that any prime greater than 2 can be uniquely represented as $a^2 - b^2$, where a and b are positive integers. ✓
7. ■ Write a program to find the biggest prime among the first 100 Fibonacci numbers. (Mathematicians don't yet know whether there are infinitely many primes among Fibonacci numbers.)
8. ■ *Goldbach's conjecture* states that every even integer greater than 2 can be represented as the sum of two primes. For example, $12 = 5 + 7$. It is not known whether this conjecture is true or false. Write a program to verify Goldbach's conjecture for all even numbers from 4 to 100.
9. ■ Show that the number of different divisors of n is even, unless n is a perfect square. The number of different divisors of a perfect square is odd.
10. ■ Suppose $n = p_1^{j_1} \cdots p_k^{j_k}$, where p_1, \dots, p_k are different primes. Express the total number of divisors of n in terms of j_1, \dots, j_k .
11. ♦ Suppose $n = p^h \cdot q^{j_2}$, where p and q are two different primes. Show that the number of positive integers that are less than n and are relatively prime with n is $n \left(1 - \frac{1}{p}\right) \left(1 - \frac{1}{q}\right)$. Extend the result for three primes; for any number of primes.

17.4 Arithmetic of Remainders

Let n and d be integers and $d > 0$. We can divide n by d with a remainder. This means that we can find integers q and r , such that $n = qd + r$, where $0 \leq r < d$. Here q is the quotient and r is the remainder.

Two integers m and n are said to be congruent modulo d , if they give the same remainder when divided by d . This is written as $m \equiv n \pmod{d}$.

$n \bmod d$ denotes the remainder when n is divided by d .

For example, $12 \equiv 27 \pmod{5}$: both 12 and 27 give the same remainder when divided by 5: $12 \bmod 5 = 27 \bmod 5 = 2$.

If $m \equiv n \pmod{d}$, then $d \mid (m - n)$, that is, $m - n$ is divisible by d .

Congruence modulo d is an *equivalence relation* on integers (see Section 15.3). Indeed, all three required criteria for an equivalence relation are satisfied:

1. Reflexivity: $n \equiv n \pmod{d}$
2. Symmetry: If $m \equiv n \pmod{d}$, then $n \equiv m \pmod{d}$
3. Transitivity: If $k \equiv m \pmod{d}$ and $m \equiv n \pmod{d}$, then $k \equiv n \pmod{d}$

Therefore, for a given d , all integers fall into non-overlapping equivalence classes with respect to congruence modulo d . Two integers from the same class give the same remainder when divided by d , and two integers from different classes give different remainders. The number of congruence classes is d .

Example 1

List the congruence classes modulo 3.

Solution

There are three classes:

$$\{\dots -12, -9, -6, -3, 0, 3, 6, 9, 12, \dots\} \equiv 0 \pmod{3}$$

$$\{\dots -11, -8, -5, -2, 1, 4, 7, 10, 13, \dots\} \equiv 1 \pmod{3}$$

$$\{\dots -10, -7, -4, -1, 2, 5, 8, 11, 14, \dots\} \equiv 2 \pmod{3}$$



If we take any integer k and add it to all numbers in a congruence class modulo d , all of them will shift into another congruence class. For example, -4 , 5 , and 14 all belong to the “2” congruence class modulo 3. If we add 7 to each of them, we get 3, 12, and 21, all of which belong to the “0” congruence class. This is so, because $(n+k) \pmod{d}$ and $n \pmod{d} + k \pmod{d}$ are congruent modulo d :

$$[(n+k) \pmod{d}] \equiv [n \pmod{d} + k \pmod{d}] \pmod{d}$$

The same is true for multiplication:

$$[(n \cdot k) \pmod{d}] \equiv [n \pmod{d} \cdot k \pmod{d}] \pmod{d}$$

All of this means that we can add and multiply two congruence classes, or more precisely, any two representatives of these congruence classes. The result falls into the same class, regardless of which representatives we chose.

From now on, we will represent each congruence class modulo d simply by all possible remainders when n is divided by d : $0, 1, 2, \dots, d-1$. Let's use the symbol \oplus_d to denote addition modulo d . The sum of two numbers modulo d simply “wraps around” at d . It is as if the number line was wrapped around a circle with the numbers $0, 1, 2, \dots, d-1$ evenly spaced on it (Figure 17-2). We are all familiar with such an arrangement on a clock, which adds hours modulo 12 (or modulo 24 on a European digital clock) and adds minutes modulo 60. The only difference is that in math we usually go counterclockwise.

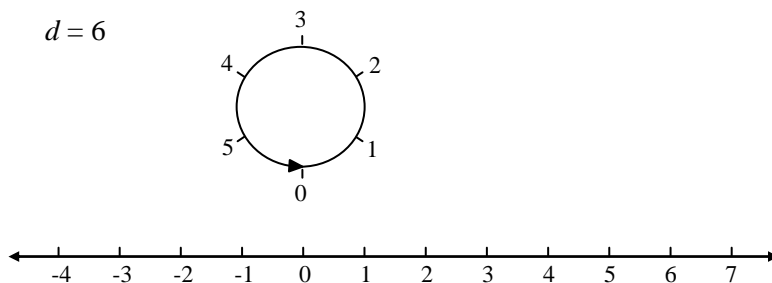


Figure 17-2. Addition and multiplication modulo d can be viewed as performed on a “number circle”

Example 2

$2 \oplus_3 2 = 4 \bmod 3 = 1$, is in the same congruence class as $5 \oplus_3 11 = 16 \bmod 3 = 1$.

Example 3

$7 \oplus_{12} 10 = 17 \bmod 12 = 5$



The same principle is used for multiplication. We will use the \otimes_d symbol to denote multiplication modulo d .

Example 4

$$2 \otimes_3 2 = 4 \pmod 3 = 1$$

$$3 \otimes_5 4 = 12 \pmod 5 = 2$$

$$7 \otimes_{12} 10 = 70 \pmod{12} = 10$$



Figure 17-3 shows modulo 5 addition and multiplication tables.

| \oplus_5 | 0 | 1 | 2 | 3 | 4 |
|------------|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 |
| 1 | 1 | 2 | 3 | 4 | 0 |
| 2 | 2 | 3 | 4 | 0 | 1 |
| 3 | 3 | 4 | 0 | 1 | 2 |
| 4 | 4 | 0 | 1 | 2 | 3 |

| \otimes_5 | 0 | 1 | 2 | 3 | 4 |
|-------------|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 | 3 | 4 |
| 2 | 0 | 2 | 4 | 1 | 3 |
| 3 | 0 | 3 | 1 | 4 | 2 |
| 4 | 0 | 4 | 3 | 2 | 1 |

Figure 17-3. Modulo 5 addition and multiplication tables



Modulo d arithmetic is similar in many ways to arithmetic on integers:

- It obeys the associative and commutative laws for addition and subtraction;
- It obeys the distributive law;
- There is a 0;
- There is a 1.
- Each element x has an additive inverse element y , such that $x \oplus_d y = 0$ (namely, $y = d - x$).

But there is a peculiarity, too: if d is not a prime, then you can find x and y such that $x \neq 0, y \neq 0$, but $x \otimes_d y = 0$. For example, $2 \otimes_6 3 = 0$. Such x and y are called *divisors of zero*.

⤵ If p is a prime, then multiplication modulo p has no divisors of zero. Moreover, for each $x \neq 0$ there is a y such that $x \otimes_p y = 1$. For example, take a look at the multiplication table in Figure 17-3: $2 \otimes_5 3 = 1$. Every row of the table has a 1. We can say that in modulo p arithmetic, $y = \frac{1}{x}$. We can do division! So the arithmetic

of remainders modulo p , for a prime p , is similar to the arithmetic of rational numbers.

Let's prove this fact: that in modulo p arithmetic any $x \neq 0$ has a reciprocal $y = \frac{1}{x}$.

But first we need to review a couple of properties of exponents:

- $x^0 = 1$
- $x^m \cdot x^n = x^{m+n}$

Theorem 1:

Let p be a prime. For any $0 < x < p$, there exist a $0 < y < p$, such that $xy \equiv 1 \pmod{p}$.

Proof:

Consider the geometric sequence $1, x, x^2, x^3, \dots$. Since there are only p different remainders modulo p , sooner or later two terms in this sequence will have the same remainder. Suppose $x^n \equiv x^m \pmod{p}$ where $n > m$. Then $x^n - x^m \equiv 0 \pmod{p}$. $x^n - x^m = x^m(x^{n-m} - 1)$ and p does not divide x^m (see Theorem 3 in Section 17.3). Therefore $x^{n-m} - 1 \equiv 0 \pmod{p} \Rightarrow x^{n-m} \equiv 1 \pmod{p}$. We can take $y = x^{n-m-1}$.

↑ Q.E.D.



If p is a prime and we take any $0 < a < p$ and look at the remainders of $1, a, a^2, a^3, \dots$ modulo p , we can see that they form a periodic sequence (Figure 17-4). The period of the sequence always divides $p - 1$, and $a^{p-1} \equiv 1 \pmod{p}$.

| mod 7 | 1 | a | a^2 | a^3 | a^4 | a^5 | a^6 | a^7 |
|---------|---|-----|-------|-------|-------|-------|-------|-------|
| $a = 2$ | 1 | 2 | 4 | 1 | 2 | 4 | 1 | 2 |
| $a = 3$ | 1 | 3 | 2 | 6 | 4 | 5 | 1 | 3 |
| $a = 4$ | 1 | 4 | 2 | 1 | 4 | 2 | 1 | 4 |

Figure 17-4. Remainders modulo p for a geometric sequence form a periodic sequence (here $p = 7$)

Theorem 2 (Fermat's Little Theorem):

For any positive integer a and a prime p , $a^p \equiv a \pmod{p}$.

Proof:

If $p \mid a$, then $a^p \equiv a \equiv 0 \pmod{p}$. If not, as we saw above $a^{p-1} \equiv 1 \pmod{p} \Rightarrow a^p \equiv a \pmod{p}$.

Exercises

1. Explore what Python's `n % d` operator returns when $n < 0$ and/or $d < 0$.
2. Show that addition and multiplication modulo 2 corresponds to the XOR and AND operations, respectively, where FALSE is 0 and TRUE is 1.
3. Write and test a function `elapsedTime` that returns the time difference in minutes from `(hour1, min1)` to `(hour2, min2)`. ✓
4. ■ Suppose the days of the week are represented by numbers: Sunday — 0, Monday — 1, and so on. Write and test a function that takes the day of the week for January 1 and returns the date of Thanksgiving (fourth Thursday in November) for that year (assuming this is not a leap year).
5. Write by hand the addition and multiplication tables modulo 6. ✓
6. Show that there are two numbers in any set of 101 integers whose difference ends with two zeroes.
7. Write and test a program that prompts the user to enter an integer d greater than 1 and prints out the addition and multiplication tables modulo d .
8. Show that if p is a prime and $p \mid a^2$ then $p^2 \mid a^2$.
9. ■ Find the smallest positive n , such that $n \equiv 2 \pmod{3}$, $n \equiv 3 \pmod{4}$, $n \equiv 4 \pmod{5}$, and so on, $n \equiv 11 \pmod{12}$.
10. ■ Calculate $3^{22222} \pmod{23}$ using pencil and paper only.

11. ■ Show that for any positive n , $6 \mid n(n+1)(2n+1)$
12. ■ Show that if p is a prime greater than or equal to 5, then $p^2 - 1$ is evenly divisible by 24.
13. ■ Write and test a function that takes a positive integer a and a prime p and returns the smallest $k > 1$ such that $a^k - a \equiv 0 \pmod{p}$.
14. Show that if p and q are two different primes, and $0 \leq a < p$ and $0 \leq b < q$, you can find x such that $x \equiv a \pmod{p}$ and $x \equiv b \pmod{q}$. (This is a simple special case of the *Chinese remainder theorem*.) ≡ Hint: see Question 10 in Section 17.2. ≻
15. Wilson's theorem states that if p is a prime, $(p-1)! + 1$ is evenly divisible by p .
- (a) Show that Wilson's theorem is true only for primes.
- (b) ♦ Prove Wilson's theorem. ≡ Hint: recall that every x has a reciprocal $y = \frac{1}{x}$ modulo p , and that $x = y$ only for $x = 1$ and $x = p-1$. ≻ ✓
- (c) ■ Write and test a program that prints out the first 100 primes using Wilson's theorem.
16. ♦ Fermat's little theorem can be used to test a number p for primeness: p is a prime if and only if $a^{p-1} \equiv 1 \pmod{p}$ for all positive a that are not divisible by p . We can choose several such values of a at random and test the condition; if it works for all of them, then the probability of p being a prime is very high. Write a function that takes a positive integer p and checks whether it is likely to be a prime by using up to 100 random values for a ($1 < a < p$). Using this function, find the first prime over 1,000,000. ≡ Hint: recall that the `randint(a, b)` function from the `random` module returns a random integer from a to b (inclusive). ≻

17.5 Number Theory in Cryptology

Cryptology is a science of making and analyzing ciphers. In a simple case, if Alice and Bob want to send each other encrypted messages, all they have to do is agree on which encryption method they are going to use. (“Alice” and “Bob” are often used in cryptology literature to explain secure communications between two people or organizations. There is a third character, eavesdropper “Eve,” who monitors all exchanges between Alice and Bob. Eve will try to guess which method Alice and Bob are using.)

Example 1

The simplest substitution cipher: each letter is replaced with the next letter in the alphabet; ‘Z’ is replaced by ‘A’. “Got an A” is encrypted as “Hpu bo B”.



This type of cipher is easy to break once many people start using the same method. In a more advanced cipher, the encryption method is widely known, but Alice and Bob share a secret *key*, which modifies the encryption scheme.

Example 2

In a simple substitution cipher, a secret key tells which letter should replace ‘A’, ‘B’, ..., ‘Z’. For example:

| | |
|--------------------|----------------------------|
| Letter to encrypt: | ABCDEFGHIJKLMNOPQRSTUVWXYZ |
| Key: | QEKUOYMBJXRCDZNVTFASHWPLI |

“On time” is encrypted as “Nz ajdo.” This cipher is easy to break by comparing the frequencies of occurrence of different letters in the plain text and encoded text and by guessing about the common short words (articles, prepositions, etc.).



In a cipher that uses a secret key, Alice and Bob must somehow share the key. They can meet or send a sealed envelope by snail mail. This might work if *A* and *B* are just two people. But what if *A* is Amazon.com and *B* is any customer who wants to place a book order in a secure manner? Imagine 100,000,000 Internet customers and 100,000 secure servers at e-commerce sites. You would need to distribute

10,000,000,000,000 different keys to allow each to communicate with each! To address this problem, companies use several very clever schemes that emerged in the 1970s. We will consider two of them: *Diffie-Hellman Key Exchange* and, in the following section, *RSA public/private key encryption*.

The Diffie-Hellman Scheme

The Diffie-Hellman Key Exchange algorithm was invented by Whitfield Diffie and Martin Hellman. The key idea (pun intended!) of the D-H scheme is to let each member of the community own “one half” of a key. All these half-keys are public, either published in a directory or available upon request from the owner. Any two halves can be put together to make a unique key. However, the person combining the two halves must know the secret code for at least one of the halves to be able to join them together. Alice has a secret code to her half and Bob has a secret code to his half, so either Alice or Bob can put together their halves to make a complete key; the resulting key will be the same. But Eve cannot put together Alice’s and Bob’s half-keys (Figure 17-5).

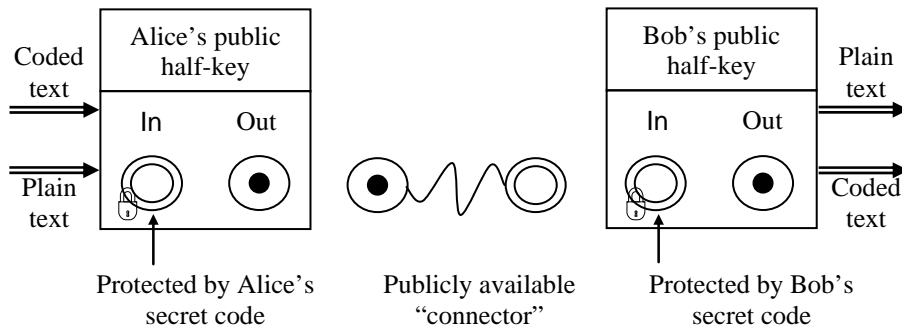


Figure 17-5. A metaphor for the D-H scheme: Alice’s and Bob’s half-keys are publicly available, but you need to know either Alice’s or Bob’s secret code to combine them into a complete key suitable for encoding and decoding messages.

In the numerical model of this scheme, Alice’s public half-key is $K_A = r^a \text{ mod } p$, Bob’s public half-key is $K_B = r^b \text{ mod } p$, and the combined key is $K = r^{ab} \text{ mod } p$, where p and r are in the public domain and are used by everyone. p is a large prime: it is chosen to have about 200 digits! $1 < r < p$.

In modulo p arithmetic, it is virtually impossible to calculate K from K_A and K_B for a very large p , unless you know a or b . It is also virtually impossible to find a from K_A or b from K_B . But if you know a or b , combining K_A and K_B into one key K is easy. Recall the property of exponents: $(r^a)^b = r^{ab}$ (see Question 4). Likewise, $(r^b)^a = r^{ab}$. So $K = (K_B)^a = (K_A)^b$.

Here are the steps Alice takes to send a coded message to Bob:

1. Alice knows the standard p and r from the public domain.
2. Alice requests K_B from Bob.
3. Alice makes the encryption key $K = (K_B)^a \pmod p = r^{ab} \pmod p$ using her secret number a .
4. Alice encodes the plain text message M into the encoded message C using K as the key for encryption: $C = \text{encode}(M, K)$. (It is assumed that everyone is using the same encryption method — only the keys are different.)
5. Alice sends C to Bob.

Bob takes similar steps to decode the message received from Alice:

1. Bob knows the standard p and r from the public domain.
2. Bob requests K_A from Alice.
3. Bob makes the encryption key $K = (K_A)^b \pmod p = r^{ab} \pmod p$ using his secret number b .
4. Bob decodes C using K : $M = \text{decode}(C, K)$.

In regular arithmetic, if you know r and r^a , it is very easy to calculate a . For example, if $r = 10$, and $r^a = 1000$, you know right away that $a = 3$.

If $r^a = x$, then a is called the *logarithm of x to the base r* . It is written as $a = \log_r x$.

In regular arithmetic, $\log_r x$ increases in a predictable way when x increases, which makes it easy to compute. Not so in modulo p arithmetic. When a changes from 1 to $p-1$, $r^a \pmod p$ jumps around in the range from 1 to $p-1$.

$a = \log_r x \pmod p$ is called the *discrete logarithm*.

Example 3

Find $\log_6 8 \pmod{11}$, that is, find a such that $6^a \equiv 8 \pmod{11}$.

Solution

Make a table of powers of 6 modulo 11:

| | | | | | | | | | | | |
|-----------------|---|---|---|---|---|----|---|---|---|---|----|
| a | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $6^a \pmod{11}$ | 1 | 6 | 3 | 7 | 9 | 10 | 5 | 8 | 4 | 2 | 1 |

An entry in the bottom row is the previous entry times 6 (mod 11). For example $6 \cdot 6 = 36 \equiv 3 \pmod{11}$; $3 \cdot 6 = 18 \equiv 7 \pmod{11}$. Look in the bottom row for the power of 6 that is equal to 8 and mark the corresponding a — here it is $a=7$. Therefore, $\log_6 8 \pmod{11} = 7$.



Can't we make and search a similar table for any r and p ? Not if p has 200 digits! We can always find r such that all the powers of r modulo p are different. Then the table of powers of r will have about 10^{200} columns! This is more than the number of atoms in the universe... squared! If we combine all the computers in the world (including all play stations*) and make them search for a discrete logarithm, it will take more time than the age of the universe... cubed!

* In the fall of 2007, The Guinness Book of Records recognized folding@home (FAH) as the world's most powerful distributed computing network. FAH has signed up more than 670,000 PS3 play stations to analyze, during their idle hours, the shapes of proteins and their effect on various diseases. The network is estimated to perform 10^{15} floating-point operations per second.

The discrete logarithm is virtually impossible to compute for very large p .

On the other hand, calculating r^a is relatively quick, even for a very large a . For example, you need to perform only 10 multiplications to calculate r^{1024} (see Question 6).



The RSA Algorithm

The name of the RSA algorithm is formed by the first letters Ron Rivest, Adi Shamir, and Leonard Adleman at MIT, who invented the algorithm in 1977, independently of previous work.** The RSA algorithm is now widely used for secure communications and e-commerce on the Internet.

The main idea of the RSA scheme is simple. Suppose Alice wants to send a secret message to Bob. Alice asks Bob to send her an open padlock to which only Bob has a key. (Bob, like every other member of the community, has an unlimited supply of such locks and sends them out for free on request.) When Alice receives the lock, she puts her message in a box, puts the lock on it, clicks it locked, and sends the box to Bob.

In the numerical implementation, Bob sends to Alice, over an open channel, two numbers, E and N . These numbers together serve as the “lock” (but they are called Bob’s public encryption key). Alice’s message is stored in a computer as a sequence of 0s and 1s, and the RSA algorithm treats these bits as binary digits of a positive integer M . Alice “puts a lock” on M , that is, creates a coded message $C = M^E \pmod N$ and sends C back to Bob. Bob uses his secret private key D to decode C and get back M : $M = C^D \pmod N$.

For this to work, we must have $C^D \equiv (M^E)^D \equiv M^{ED} \equiv M \pmod N$, for any positive integer M . How is this possible?

** British mathematician Clifford Cocks, who worked for a UK intelligence agency, described a similar algorithm in 1973 in a top-secret internal paper. His discovery remained unknown until 1997.

Recall Fermat's little theorem (Section 17.4), which states that if p is a prime, then $x^p \equiv x \pmod{p}$, for any positive integer x . This is our starting point. If x is relatively prime with p , then $x^{p-1} \equiv 1 \pmod{p}$, which means $p \mid (x^{p-1} - 1)$.

The RSA scheme relies on a more general theorem. In RSA, N is not a prime but a product of two different primes: $N = p \cdot q$. p and q are chosen to be very large primes, at least 100 digits each, so N has at least 200 digits (over 640 binary digits). It is easy to calculate $N = p \cdot q$ but it is virtually impossible to factor N if p and q are kept secret — it would take forever.

The math needed to get $M^{ED} \equiv M \pmod{N}$ is a little long but elegant:

1. We limit M to $M < p$ and $M < q$. Then M is relatively prime to p and to q . So is any power of M .
2. Applying Fermat's little theorem to $x = M^{y(q-1)}$ and p , where y is any positive integer, we get $p \mid \left((M^{y(q-1)})^{p-1} - 1 \right) \Rightarrow p \mid (M^{y(q-1)(p-1)} - 1)$. Similarly, $q \mid (M^{y(p-1)(q-1)} - 1)$.
3. Since both p and q are divisors of $M^{y(p-1)(q-1)} - 1$, their product N is also a divisor: $N \mid (M^{y(p-1)(q-1)} - 1)$. Therefore $M^{y(p-1)(q-1)} \equiv 1 \pmod{N}$. Multiplying both sides by M we get $M^{y(p-1)(q-1)+1} \equiv M \pmod{N}$.
4. All of the above are well-known results in number theory. Now all we need is to represent $y(p-1)(q-1)+1$ as a product ED . Bob chooses such E that it is relatively prime with $p-1$ and $q-1$. Then E is relatively prime with $(p-1)(q-1)$. Solving the Diophantine equation $Ex - (p-1)(q-1)y = 1$ (see Section 17.2), we find x and y and set $D = x$. Then $y(p-1)(q-1)+1 = ED$.
5. We have found D and E , such that $M^{ED} \equiv M \pmod{N}$ for any M relatively prime with N (in particular, for any M such that $M < p$ and $M < q$).

Eve knows only $N = pq$ and E , but not p and q . She can't factor N or compute $(p-1)(q-1)$, so she can't compute D .

The RSA scheme is a little slow, so, in practice, it is used only to send a secret key for a different cipher. Once both Alice and Bob know that secret key, they can communicate in that cipher.

Exercises

1. Write and test two functions, `encode(text, key)` and `decode(code, key)`, that implement a substitution cipher with a key. `text`, `code`, and `key` are strings, and each of the functions returns a string.
2. Write and test a function `letterCounts(text)` that calculates how many times each letter of the alphabet occurs in `text` and returns a list of these 26 counts.
3. Find on the Internet a description of the Vigenere cipher. Write and test `encode(text, key)` and `decode(code, key)` that use that cipher.
4. Show that for positive integers a and b , $(r^a)^b = r^{ab}$.
5. Write and test an efficient iterative function (no recursion) that performs only d multiplications to calculate $r^{(2^d)}$. Do not use `**`, `pow`, or any other Python built-in functions.
- 6.▪ Calculate 3^{2013} without using the `**` operator, the `pow` function, or any other Python built-in function, in such a way that multiplication is performed fewer than 60 times total. ⚡ Hint: write an efficient iterative function to calculate r^a . Recall that $a = d_k \cdot 2^k + \dots + d_1 \cdot 2 + d_0$, where d_k, \dots, d_0 are binary digits of a , and see Question 5. ⚡
7. Write and test a function that calculates and returns $r^a \bmod p$ in such a way that no intermediate result ever exceeds rp .
- 8.♦ $p = 170141183460469231731687303715884105727$ is a prime.
 $a = 618970019642690137449562111$.
 $r = 5$.
Calculate $r^a \bmod p$. ⚡ Hint: write an efficient function for $r^a \bmod p$; see Questions 6 and 7. ⚡

9. ■ Write and test a recursive function that uses a divide-and-conquer algorithm to calculate $r^a \bmod p$ efficiently.
10. ■ $p = 31, r = 3$.
- Write and test a function `generatePublicKeys(secretCodes)` that takes a list of secret codes for several users and generates and returns a list of public half-keys for them in the D-H scheme. For example, `generatePublicKeys([3, 7, 4])` should return `[27, 17, 19]`. ≡ Hint: use your function from Question 9 and a list comprehension. ≻
 - Write a function `encode(msg, a, kB)` that encodes a message from Alice to Bob according to the D-H scheme using her secret code `a` and his public half-key `kB`. Assume that `msg` is a number, and simply use `msg ^ k` (bit-wise xor with the key) to encode it. The function should return a number that corresponds to the encoded message.
 - Write a function `decode(code, b, kA)` that decodes a message from Alice to Bob according to the D-H scheme using his secret code `b` and her public half-key `kA`. The function should return the decoded message.
 - Test the `encode` and `decode` functions. For example, `encode(9, 3, 17)` should return 6, and `decode(6, 7, 27)` should return 9.
11. Show that if $x \equiv a \pmod{p}$ and $x \equiv a \pmod{q}$, then $x \equiv a \pmod{pq}$.
12. In the RSA algorithm, given $p = 13, q = 17$, and $E = 5$, find D .
13. In describing RSA, we used a metaphor of Bob sending an open lock to Alice and Alice sending Bob a secret message in a box locked by that lock. Suppose only locked boxes are allowed in the mail. Can Alice still send a secret message to Bob? ≡ Hint: two locks can fit on a box. ≻

17.6 Review

Terms and notation introduced in this chapter:

| | | |
|--|---------------------------------|-----------------------|
| <i>Number theory</i> | <i>Divisors of 0</i> | $p \mid a$ |
| <i>Remainder</i> | <i>Fermat's little theorem</i> | $\text{GCD}(a, b)$ |
| <i>Divisor</i> | <i>Substitution cipher</i> | $a \equiv b \pmod{d}$ |
| <i>GCD</i> | <i>Diffie-Hellman scheme</i> | $\log_r x$ |
| <i>Euclid's algorithm</i> | <i>Properties of exponents:</i> | |
| <i>Relatively prime numbers</i> | $r^a \cdot r^b = r^{a+b}$ | |
| <i>Diophantine equation</i> | $(r^a)^b = r^{ab}$ | |
| <i>The fundamental theorem of arithmetic</i> | <i>Discrete logarithm</i> | |
| <i>Corollary</i> | <i>The RSA algorithm</i> | |
| <i>Congruence modulo d</i> | | |