Exercises

1.1. Show that in any base $b \geq 2$, the sum of any three single-digit numbers is at most two digits long.

1.2. Show that any binary integer is at most four times as long as the corresponding decimal integer. For very large numbers, what is the ratio of these two lengths, approximately?

1.3. A d-ary tree is a rooted tree in which each node has at most d children. Show that any d-ary tree with n nodes must have a depth of $\Omega(\log n / \log d)$. Can you give a precise formula for the minimum depth it could possibly have?

1.4. Show that

$$\log(n!) \;\; = \;\; \Theta(n \log n).$$

(*Hint:* To show an upper bound, compare $n!$ with n^n. To show a lower bound, compare it with $(n/2)^{n/2}$.)

1.5. Unlike a decreasing geometric series, the sum of the *harmonic series* $1, 1/2, 1/3, 1/4, 1/5, \ldots$ diverges; that is,

$$\sum_{i=1}^{\infty} \frac{1}{i} = \infty.$$

It turns out that, for large n, the sum of the first n terms of this series can be well approximated as

$$\sum_{i=1}^{n} \frac{1}{i} \;\; \approx \;\; \ln n + \gamma,$$

where $\ln$ is natural logarithm (log base $e = 2.718\ldots$) and γ is a particular constant $0.57721\ldots$.

Show that

$$\sum_{i=1}^{n} \frac{1}{i} \;\; = \;\; \Theta(\log n).$$

(*Hint:* To show an upper bound, decrease each denominator to the next power of two. For a lower bound, increase each denominator to the next power of 2.)

1.6. Prove that the grade-school multiplication algorithm (page 24), when applied to binary numbers, always gives the right answer.

1.7. How long does the recursive multiplication algorithm (page 25) take to multiply an n-bit number by an m-bit number? Justify your answer.

1.8. Justify the correctness of the recursive division algorithm given in page 26, and show that it takes time $O(n^2)$ on n-bit inputs.

1.9. Starting from the definition of $x \equiv y \bmod N$ (namely, that N divides $x-y$), justify the substitution rule

$$x \equiv x' \bmod N, \;\; y \equiv y' \bmod N \;\; \Rightarrow \;\; x + y \equiv x' + y' \bmod N,$$

and also the corresponding rule for multiplication.

1.10. Show that if $a \equiv b \pmod N$ and if M divides N then $a \equiv b \pmod M$.

1.11. Is $4^{1536} - 9^{4824}$ divisible by 35?

1.12. What is $2^{2^{2006}} \pmod 3$?

1.13. Is the difference of $5^{30,000}$ and $6^{123,456}$ a multiple of 31?

1.14. Suppose you want to compute the nth Fibonacci number F_n, modulo an integer p. Can you find an efficient way to do this? (*Hint:* Recall Exercise 0.4.)

1.15. Determine necessary and sufficient conditions on x and c so that the following holds: for any a, b, if $ax \equiv bx \bmod c$, then $a \equiv b \bmod c$.

1.16. The algorithm for computing $a^b \bmod c$ by repeated squaring does not necessarily lead to the minimum number of multiplications. Give an example of $b > 10$ where the exponentiation can be performed using fewer multiplications, by some other method.

1.17. Consider the problem of computing x^y for given integers x and y: we want the *whole* answer, not modulo a third integer. We know two algorithms for doing this: the iterative algorithm which performs $y - 1$ multiplications by x; and the recursive algorithm based on the binary expansion of y.

Compare the time requirements of these two algorithms, assuming that the time to multiply an n-bit number by an m-bit number is $O(mn)$.

1.18. Compute $\gcd(210, 588)$ two different ways: by finding the factorization of each number, and by using Euclid's algorithm.

1.19. The *Fibonacci numbers* $F_0, F_1, \ldots$ are given by the recurrence $F_{n+1} = F_n + F_{n-1}$, $F_0 = 0$, $F_1 = 1$. Show that for any $n \geq 1$, $\gcd(F_{n+1}, F_n) = 1$.

1.20. Find the inverse of: $20 \bmod 79,\ 3 \bmod 62,\ 21 \bmod 91,\ 5 \bmod 23$.

1.21. How many integers modulo 11^3 have inverses? (Note: $11^3 = 1331$.)

1.22. Prove or disprove: If a has an inverse modulo b, then b has an inverse modulo a.

1.23. Show that if a has a multiplicative inverse modulo N, then this inverse is unique (modulo N).

1.24. If p is prime, how many elements of $\{0, 1, \ldots, p^n - 1\}$ have an inverse modulo p^n?

1.25. Calculate $2^{125} \bmod 127$ using any method you choose. (*Hint:* 127 is prime.)

1.26. What is the least significant decimal digit of $17^{17^{17}}$? (*Hint:* For distinct primes p, q, and any $a \not\equiv 0 \pmod{pq}$, we proved the formula $a^{(p-1)(q-1)} \equiv 1 \pmod{pq}$ in Section 1.4.2.)

1.27. Consider an RSA key set with $p = 17, q = 23, N = 391$, and $e = 3$ (as in Figure 1.9). What value of d should be used for the secret key? What is the encryption of the message $M = 41$?

1.28. In an RSA cryptosystem, $p = 7$ and $q = 11$ (as in Figure 1.9). Find appropriate exponents d and e.

1.29. Let $[m]$ denote the set $\{0, 1, \ldots, m - 1\}$. For each of the following families of hash functions, say whether or not it is universal, and determine how many random bits are needed to choose a function from the family.

(a) $H = \{h_{a_1,a_2} : a_1, a_2 \in [m]\}$, where m is a fixed prime and

$$h_{a_1,a_2}(x_1, x_2) = a_1x_1 + a_2x_2 \bmod m.$$

Notice that each of these functions has signature $h_{a_1,a_2} : [m]^2 \to [m]$, that is, it maps a pair of integers in $[m]$ to a single integer in $[m]$.

(b) H is as before, except that now $m = 2^k$ is some fixed power of 2.

(c) H is the set of all functions $f : [m] \to [m - 1]$.

1.30. The grade-school algorithm for multiplying two n-bit binary numbers x and y consists of adding together n copies of x, each appropriately left-shifted. Each copy, when shifted, is at most $2n$ bits long.

In this problem, we will examine a scheme for adding n binary numbers, each m bits long, using a *circuit* or a *parallel architecture*. The main parameter of interest in this question is therefore the depth of the circuit or the longest path from the input to the output of the circuit. This determines the total time taken for computing the function.

To add two m-bit binary numbers naively, we must wait for the carry bit from position $i-1$ before we can figure out the ith bit of the answer. This leads to a circuit of depth $O(m)$. However carry lookahead circuits (see wikipedia.com if you want to know more about this) can add in $O(\log m)$ depth.

(a) Assuming you have carry lookahead circuits for addition, show how to add n numbers each m bits long using a circuit of depth $O((\log n)(\log m))$.

(b) When adding *three* m-bit binary numbers $x+y+z$, there is a trick we can use to parallelize the process. Instead of carrying out the addition completely, we can re-express the result as the sum of just *two* binary numbers $r+s$, such that the ith bits of r and s can be computed independently of the other bits. Show how this can be done. (*Hint:* One of the numbers represents carry bits.)

(c) Show how to use the trick from the previous part to design a circuit of depth $O(\log n)$ for multiplying two n-bit numbers.

1.31. Consider the problem of computing $N! = 1 \cdot 2 \cdot 3 \cdots N$.

(a) If N is an n-bit number, how many bits long is $N!$, approximately (in $\Theta(\cdot)$ form)?

(b) Give an algorithm to compute $N!$ and analyze its running time.

1.32. A positive integer N is a *power* if it is of the form q^k, where q, k are positive integers and $k > 1$.

(a) Give an efficient algorithm that takes as input a number N and determines whether it is a square, that is, whether it can be written as q^2 for some positive integer q. What is the running time of your algorithm?

(b) Show that if $N = q^k$ (with N, q, and k all positive integers), then either $k \leq \log N$ or $N = 1$.

(c) Give an efficient algorithm for determining whether a positive integer N is a power. Analyze its running time.

1.33. Give an efficient algorithm to compute the *least common multiple* of two n-bit numbers x and y, that is, the smallest number divisible by both x and y. What is the running time of your algorithm as a function of n?

1.34. On page 38, we claimed that since about a $1/n$ fraction of n-bit numbers are prime, on average it is sufficient to draw $O(n)$ random n-bit numbers before hitting a prime. We now justify this rigorously.

Suppose a particular coin has a probability p of coming up heads. How many times must you toss it, on average, before it comes up heads? (*Hint:* Method 1: start by showing that the correct expression is $\sum_{i=1}^{\infty} i(1-p)^{i-1}p$. Method 2: if E is the average number of coin tosses, show that $E = 1 + (1-p)E$.)

1.35. Wilson's theorem says that a number N is prime if and only if

$$(N-1)! \equiv -1 \pmod{N}.$$

(a) If p is prime, then we know every number $1 \leq x < p$ is invertible modulo p. Which of these numbers are their own inverse?

(b) By pairing up multiplicative inverses, show that $(p-1)! \equiv -1 \pmod{p}$ for prime p.

(c) Show that if N is *not* prime, then $(N-1)! \not\equiv -1 \pmod{N}$. (***Hint:*** Consider $d = \gcd(N, (N-1)!)$.)

(d) Unlike Fermat's Little theorem, Wilson's theorem is an if-and-only-if condition for primality. Why can't we immediately base a primality test on this rule?

1.36. *Square roots.* In this problem, we'll see that it is easy to compute square roots modulo a prime p with $p \equiv 3 \pmod{4}$.

(a) Suppose $p \equiv 3 \pmod{4}$. Show that $(p+1)/4$ is an integer.

(b) We say x is a *square root* of a modulo p if $a \equiv x^2 \pmod{p}$. Show that if $p \equiv 3 \pmod{4}$ and if a has a square root modulo p, then $a^{(p+1)/4}$ is such a square root.

1.37. *The Chinese remainder theorem.*

(a) Make a table with three columns. The first column is all numbers from 0 to 14. The second is the residues of these numbers modulo 3; the third column is the residues modulo 5. What do you observe?

(b) Prove that if p and q are distinct primes, then for every pair (j, k) with $0 \leq j < p$ and $0 \leq k < q$, there is a unique integer $0 \leq i < pq$ such that $i \equiv j \bmod p$ and $i \equiv k \bmod q$. (***Hint:*** Prove that no two different i's in this range can have the same (j, k), and then count.)

(c) In this one-to-one correspondence between integers and pairs, it is easy to go from i to (j, k). Prove that the following formula takes you the other way:

$$i = \{j \cdot q \cdot (q^{-1} \bmod p) + k \cdot p \cdot (p^{-1} \bmod q)\} \bmod pq.$$

(d) Can you generalize parts (b) and (c) to more than two primes?

1.38. To see if a number, say 562437487, is divisible by 3, you just add up the digits of its decimal representation, and see if the result is divisible by 3. ($5+6+2+4+3+7+4+8+7 = 46$, so it is not divisible by 3.)

To see if the same number is divisible by 11, you can do this: subdivide the number into pairs of digits, from the right-hand end ($87, 74, 43, 62, 5$), add these numbers, and see if the sum is divisible by 11 (if it's too big, repeat).

How about 37? To see if the number is divisible by 37, subdivide it into triples from the end ($487, 437, 562$) add these up, and see if the sum is divisible by 37.

This is true for any prime p other than 2 and 5. That is, for any prime $p \neq 2, 5$, there is an integer r such that in order to see if p divides a decimal number n, we break n into r-tuples of decimal digits (starting from the right-hand end), add up these r-tuples, and check if the sum is divisible by p.

(a) What is the smallest such r for $p = 13$? For $p = 17$?

(b) Show that r is a divisor of $p - 1$.

1.39. Give a polynomial-time algorithm for computing $a^{b^c} \bmod p$, given a, b, c, and prime p.

1.40. Show that if x is a nontrivial square root of 1 modulo N, that is, if $x^2 \equiv 1 \bmod N$ but $x \not\equiv \pm 1 \bmod N$, then N must be composite. (For instance, $4^2 \equiv 1 \bmod 15$ but $4 \not\equiv \pm 1 \bmod 15$; thus 4 is a nontrivial square root of 1 modulo 15.)

1.41. *Quadratic residues.* Fix a positive integer N. We say that a is a *quadratic residue* modulo N if there exists x such that $a \equiv x^2 \bmod N$.

(a) Let N be an odd prime and a be a non-zero quadratic residue modulo N. Show that there are exactly two values in $\{0, 1, \ldots, N-1\}$ satisfying $x^2 \equiv a \bmod N$.

(b) Show that if N is an odd prime, there are exactly $(N+1)/2$ quadratic residues in $\{0, 1, \ldots, N-1\}$.

(c) Give an example of positive integers a and N such that $x^2 \equiv a \bmod N$ has more than two solutions in $\{0, 1, \ldots, N-1\}$.

1.42. Suppose that instead of using a composite $N = pq$ in the RSA cryptosystem (Figure 1.9), we simply use a prime modulus p. As in RSA, we would have an encryption exponent e, and the encryption of a message $m \bmod p$ would be $m^e \bmod p$. Prove that this new cryptosystem is not secure, by giving an efficient algorithm to decrypt: that is, an algorithm that given p, e, and $m^e \bmod p$ as input, computes $m \bmod p$. Justify the correctness and analyze the running time of your decryption algorithm.

1.43. In the RSA cryptosystem, Alice's public key (N, e) is available to everyone. Suppose that her private key d is compromised and becomes known to Eve. Show that if $e = 3$ (a common choice) then Eve can efficiently factor N.

1.44. Alice and her three friends are all users of the RSA cryptosystem. Her friends have public keys $(N_i, e_i = 3)$, $i = 1, 2, 3$, where as always, $N_i = p_i q_i$ for randomly chosen n-bit primes p_i, q_i. Show that if Alice sends the same n-bit message M (encrypted using RSA) to each of her friends, then anyone who intercepts all three encrypted messages will be able to efficiently recover M.

(*Hint:* It helps to have solved problem 1.37 first.)

1.45. *RSA and digital signatures.* Recall that in the RSA public-key cryptosystem, each user has a public key $P = (N, e)$ and a secret key d. In a *digital signature scheme*, there are two algorithms, `sign` and `verify`. The `sign` procedure takes a message and a secret key, then outputs a signature σ. The `verify` procedure takes a public key (N, e), a signature σ, and a message M, then returns "true" if σ could have been created by `sign` (when called with message M and the secret key corresponding to the public key (N, e)); "false" otherwise.

(a) Why would we want digital signatures?

(b) An RSA signature consists of $\texttt{sign}(M, d) = M^d \pmod N$, where d is a secret key and N is part of the public key. Show that anyone who knows the public key (N, e) can perform $\texttt{verify}((N, e), M^d, M)$, i.e., they can check that a signature really was created by the private key. Give an implementation and prove its correctness.

(c) Generate your own RSA modulus $N = pq$, public key e, and private key d (you don't need to use a computer). Pick p and q so you have a 4-digit modulus and work by hand. Now sign your name using the private exponent of this RSA modulus. To do this you will need to specify some one-to-one mapping from strings to integers in $[0, N-1]$. Specify any mapping you like. Give the mapping from your name to numbers $m_1, m_2, \ldots m_k$, then sign the first number by giving the value $m_1^d \pmod N$, and finally show that $(m_1^d)^e = m_1 \pmod N$.

(d) Alice wants to write a message that looks like it was digitally signed by Bob. She notices that Bob's public RSA key is $(17, 391)$. To what exponent should she raise her message?

1.46. *Digital signatures, continued.* Consider the signature scheme of Exercise 1.45.

(a) Signing involves decryption, and is therefore risky. Show that if Bob agrees to sign anything he is asked to, Eve can take advantage of this and decrypt any message sent by Alice to Bob.

(b) Suppose that Bob is more careful, and refuses to sign messages if their signatures look suspiciously like text. (We assume that a randomly chosen message—that is, a random number in the range $\{1, \ldots, N-1\}$—is very unlikely to look like text.) Describe a way in which Eve can nevertheless still decrypt messages from Alice to Bob, by getting Bob to sign messages whose signatures look random.