31 Number-Theoretic Algorithms

31.1 Elementary number-theoretic notions

31.1-1

> Prove that if $a > b > 0$ and $c = a + b$, then $c \mod a = b$.

$$\begin{aligned} c \mod a &= (a+b) \mod a \\ &= (a \mod a) + (b \mod a) \\ &= 0 + b \\ &= b. \end{aligned}$$

31.1-2

> Prove that there are infinitely many primes.

$$\begin{aligned} ((p_1 p_2 \cdots p_k) + 1) \mod p_i &= (p_1 p_2 \cdots p_k) \mod p_i + (1 \mod p_i) \\ &= 0 + 1 \\ &= 1. \end{aligned}$$

if $p_1 p_2 \cdots p_k$ is all prime numbers, then $(p_1 p_2 \cdots p_k) + 1$ is a new prime number.

31.1-3

> Prove that if $a \mid b$ and $b \mid c$, then $a \mid c$.

- If $a \mid b$, then $b = a \cdot k_1$.
- If $b \mid c$, then $c = b \cdot k_2 = a \cdot (k_1 \cdot k_2) = a \cdot k_3$, then $a \mid c$.

31.1-4

> Prove that if p is prime and $0 < k < p$, then $\gcd(k, p) = 1$.

- If $k \neq 1$, then $k \nmid p$.
- If $k = 1$, then the divisor is 1.

31.1-5

> Prove Corollary 31.5.
>
> For all positive integers n, a, and b, if $n \mid ab$ and $\gcd(a, n) = 1$, then $n \mid b$.

Assume for the purpose of contradiction that $n \mid ab$ and $\gcd(a, n) = 1$, but $n \nmid b$, so $\gcd(n, b) = 1$, for theorem 31.6, $\gcd(n, ab) = 1$ which is contradicting our assumption.

31.1-6

> Prove that if p is prime and $0 < k < p$, then $p \mid \binom{p}{k}$. Conclude that for all integers a and b and all primes p,
>
> $(a+b)^p \equiv a^p + b^p \pmod p$.

$$\begin{aligned} (a+b)^p &\equiv a^p + \binom{p}{1} a^{p-1} b^1 + \cdots + \binom{p}{p-1} a^1 b^{p-1} + b^p \pmod p \\ &\equiv a^p + 0 + \cdots + 0 + b^p \pmod p \\ &\equiv a^p + b^p \pmod p \end{aligned}$$

31.1-7

> Prove that if a and b are any positive integers such that $a \mid b$, then
>
> $$(x \mod b) \mod a = x \mod a$$
>
> for any x. Prove, under the same assumptions, that
>
> $x \equiv y \pmod b$ implies $x \equiv y \pmod a$
>
> for any integers x and y.

Suppose $x = kb + c$, we have

$$(x \mod b) \mod a = c \mod a,$$

and

$$x \mod a = (kb + c) \mod a = (kb \mod a) + (c \mod a) = c \mod a.$$

31.1-8

> For any integer $k > 0$, an integer n is a k***th power*** if there exists an integer a such that $a^k = n$. Furthermore, $n > 1$ is a ***nontrivial power*** if it is a kth power for some integer $k > 1$. Show how to determine whether a given β-bit integer n is a nontrivial power in time polynomial in β.

Because $2^\beta > n$, we only need to test values of k that satisfy $2 \le k < \beta$, therefore the testing procedure remains $O(\beta)$.

For any nontrivial power k, where $2 \le k < \beta$, do a binary search on a that costs

$$O(\log \sqrt[k]{n}) = O(\log \sqrt[k]{2^\beta}) = O(\frac{1}{2}\log 2^\beta) = O(\beta).$$

Thus, the total time complexity is

$$O(\beta) \times O(\beta) = O(\beta^2).$$

31.1-9

> Prove equations (31.6)–(31.10).

(Omit!)

31.1-10

> Show that the gcd operator is associative. That is, prove that for all integers a, b, and c,
>
> $\gcd(a, \gcd(b, c)) = \gcd(\gcd(a, b), c)$.

[The following proof is provided by my friend, Tony Xiao.]

Let $d = \gcd(a, b, c)$, $a = dp$, $b = dq$ and $c = dr$.

Claim $\gcd(a, \gcd(b, c)) = d$.

Let $e = \gcd(b, c)$, thus

$$b = es,$$
$$c = et.$$

Since $d \mid b$ and $d \mid c$, thus $d \mid e$.

Let $e = dm$, thus

$$b = (dm)s = dq,$$
$$c = (dm)t = dr.$$

Suppose $k = \gcd(p, m)$,

$$
\begin{aligned}
& k \mid p, k \mid m, \\
\Rightarrow & dk \mid dp, dk \mid dm, \\
\Rightarrow & dk \mid dp, dk \mid (dm)s, dk \mid (dm)t, \\
\Rightarrow & dk \mid a, dk \mid b, dk \mid c.
\end{aligned}
$$

Since $d = \gcd(a, b, c)$, thus $k = 1$.

$$
\begin{aligned}
\gcd(a, \gcd(b, c)) &= \gcd(a, e) \\
&= \gcd(dp, dm) \\
&= d \cdot \gcd(p, m) \\
&= d \cdot k \\
&= d.
\end{aligned}
$$

By the claim,

$$\gcd(a, \gcd(b, c)) = d = \gcd(\gcd(a, b), c).$$

31.1-11 ⋆

> Prove Theorem 31.8.

(Omit!)

31.1-12

> Give efficient algorithms for the operations of dividing a β-bit integer by a shorter integer and of taking the remainder of a β-bit integer when divided by a shorter integer. Your algorithms should run in time $\Theta(\beta^2)$.

Shift left until the two numbers have the same length, then repeatedly subtract with proper multiplier and shift right.

31.1-13

> Give an efficient algorithm to convert a given β-bit (binary) integer to a decimal representation. Argue that if multiplication or division of integers whose length is at most β takes time $M(\beta)$, then we can convert binary to decimal in time $\Theta(M(\beta) \lg \beta)$.

(Omit!)

31.2 Greatest common divisor

31.2-1

> Prove that equations (31.11) and (31.12) imply equation (31.13).

(Omit!)

31.2-2

> Compute the values (d, x, y) that the call $\text{EXTENDED-EUCLID}(899, 493)$ returns.

$(29, -6, 11)$.

31.2-3

> Prove that for all integers a, k, and n,
>
> $\gcd(a, n) = \gcd(a + kn, n)$.

- $\gcd(a, n) \mid \gcd(a + kn, n)$.

  Let $d = \gcd(a, n)$, then $d \mid a$ and $d \mid n$.

Since

$$(a + kn) \mod d = a \mod d + k \cdot (n \mod d) = 0$$

and $d \mid n$, we have

$$d \mid \gcd(a + kn, n)$$

and

$$\gcd(a, n) \mid \gcd(a + kn, n).$$

- $\gcd(a + kn, n) \mid \gcd(a, n)$.

  Suppose $d = \gcd(a + kn, n)$, we have $d \mid n$ and $d \mid (a + kn)$.

  Since

  $$(a + kn) \mod d = a \mod d + k \cdot (n \mod d) = a \mod d = 0,$$

  we have $d \mid a$.

  Since $d \mid a$ and $d \mid n$, we have

  $$d \mid \gcd(a, n)$$

  and

  $$\gcd(a + kn, n) \mid \gcd(a, n).$$

  Since

  $$\gcd(a, n) \mid \gcd(a + kn, n)$$

  and

  $$\gcd(a + kn, n) \mid \gcd(a, n),$$

  we have

  $$\gcd(a, n) = \gcd(a + kn, n).$$

31.2-4

> Rewrite EUCLID in an iterative form that uses only a constant amount of memory (that is, stores only a constant number of integer values).

```
EUCLID(a, b)
    while b ≠ 0
        t = a
        a = b
        b = t % b
    return a
```

31.2-5

> If $a > b \geq 0$, show that the call EUCLID(a, b) makes at most $1 + \log_\phi b$ recursive calls. Improve this bound to $1 + \log_\phi (b / \gcd(a, b))$.

$$b \geq F_{k+1} \approx \phi^{k+1} / \sqrt{5}$$

$$k + 1 < \log_\phi \sqrt{5} + \log_\phi b \approx 1.67 + \log_\phi b$$

$$k < 0.67 + \log_\phi b < 1 + \log_\phi b.$$

Since $d \cdot a \mod d \cdot b = d \cdot (a \mod b)$, $\gcd(d \cdot a, d \cdot b)$ has the same number of recursive calls with $\gcd(a, b)$, therefore we could let $b' = b/\gcd(a, b)$, the inequality $k < 1 + \log_\phi(b') = 1 + \log_\phi(b/\gcd(a, b))$. will holds.

31.2-6

> What does EXTENDED-EUCLID(F_{k+1}, F_k) return? Prove your answer correct.

- If k is odd, then $(1, -F_{k-2}, F_{k-1})$.
- If k is even, then $(1, F_{k-2}, -F_{k-1})$.

31.2-7

> Define the gcd function for more than two arguments by the recursive equation $\gcd(a_0, a_1, \cdots, a_n) = \gcd(a_0, \gcd(a_1, a_2, \cdots, a_n))$. Show that the gcd function returns the same answer independent of the order in which its arguments are specified. Also show how to find integers $x_0, x_1, \cdots, x_n$ such that $\gcd(a_0, a_1, \ldots, a_n) = a_0x_0 + a_1x_1 + \cdots + a_nx_n$. Show that the number of divisions performed by your algorithm is $O(n + \lg(\max\{a_0, a_1, \cdots, a_n\}))$.

Suppose

$$\gcd(a_0, \gcd(a_1, a_2, \cdots, a_n)) = a_0 \cdot x + \gcd(a_1, a_2, \cdots, a_n) \cdot y$$

and

$$\gcd(a_1, \gcd(a_2, a_3, \cdots, a_n)) = a_1 \cdot x' + \gcd(a_2, a_3, \cdots, a_n) \cdot y',$$

then the coefficient of a_1 is $y \cdot x'$.

```
EXTENDED-EUCLID(a, b)
    if b == 0
        return (a, 1, 0)
    (d, x, y) = EXTENDED-EUCLID(b, a % b)
    return (d, y, x - (a / b) * y)
```

```
EXTENDED-EUCLID-MULTIPLE(a)
    if a.length == 1
        return (a[0], 1)
    g = a[a.length - 1]
    xs = [1] * a.length
    ys = [0] * a.length
    for i = a.length - 2 downto 0
        (g, xs[i], ys[i + 1]) = EXTENDED-EUCLID(a[i], g)
    m = 1
    for i = 1 to a.length
        m *= ys[i]
        xs[i] *= m
    return (g, xs)
```

31.2-8

> Define $\text{lcm}(a_1, a_2, \ldots, a_n)$ to be the ***least common multiple*** of the n integers $a_1, a_2, \ldots, a_n$, that is, the smallest nonnegative integer that is a multiple of each a_i. Show how to compute $\text{lcm}(a_1, a_2, \ldots, a_n)$ efficiently using the (two-argument) gcd operation as a subroutine.

```
GCD(a, b)
    if b == 0
        return a
    return GCD(b, a % b)
```

```
LCM(a, b)
    return a / GCD(a, b) * b
```

```
LCM-MULTIPLE(a)
    l = a[0]
    for i = 1 to a.length
        l = LCM(l, a[i])
    return l
```

31.2-9

> Prove that n_1, n_2, n_3, and n_4 are pairwise relatively prime if and only if
>
> $$\gcd(n_1 n_2, n_3 n_4) = \gcd(n_1 n_3, n_2 n_4) = 1.$$
>
> More generally, show that $n_1, n_2, \ldots, n_k$ are pairwise relatively prime if and only if a set of $\lceil \lg k \rceil$ pairs of numbers derived from the n_i are relatively prime.

Suppose $n_1 n_2 x + n_3 n_4 y = 1$, then $n_1(n_2 x) + n_3(n_4 y) = 1$, thus n_1 and n_3 are relatively prime, n_1 and n_4, n_2 and n_3, n_2 and n_4 are the all relatively prime. And since $\gcd(n_1 n_3, n_2 n_4) = 1$, all the pairs are relatively prime.

General: in the first round, divide the elements into two sets with equal number of elements, calculate the products of the two set separately, if the two products are relatively prime, then the element in one set is pairwise relatively prime with the element in the other set. In the next iterations, for each set, we divide the elements into two subsets, suppose we have subsets $\{(s_1, s_2), (s_3, s_4), \ldots\}$, then we calculate the products of $\{s_1, s_3, \ldots\}$ and $\{s_2, s_4, \ldots\}$, if the two products are relatively prime, then all the pairs of subset are pairwise relatively prime similar to the first round. In each iteration, the number of elements in a subset is half of the original set, thus there are $\lceil \lg k \rceil$ pairs of products.

To choose the subsets efficiently, in the kth iteration, we could divide the numbers based on the value of the index's kth bit.

31.3 Modular arithmetic

31.3-1

> Draw the group operation tables for the groups $(\mathbb{Z}_4, +_4)$ and $(\mathbb{Z}_5^*, \cdot_5)$. Show that these groups are isomorphic by exhibiting a one-to-one correspondence α between their elements such that $a + b \equiv c \pmod 4$ if and only if $\alpha(a) \cdot \alpha(b) \equiv \alpha(c) \pmod 5$.

- $(\mathbb{Z}_4, +_4)$: $\{0, 1, 2, 3\}$.
- $(\mathbb{Z}_5^*, \cdot_5)$: $\{1, 2, 3, 4\}$.

$\alpha(x) = 2^{x-1}$.

31.3-2

> List all subgroups of $\mathbb{Z}_9$ and of $\mathbb{Z}_{13}^*$.

- $\mathbb{Z}_9$:
  - $\langle 0 \rangle = \{0\}$,
  - $\langle 1 \rangle = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$,
  - $\langle 2 \rangle = \{0, 2, 4, 6, 8\}$.
- $\mathbb{Z}_{13}^*$:

- $\langle 1 \rangle = \{1\}$,
- $\langle 2 \rangle = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$.

31.3-3

> Prove Theorem 31.14.

A nonempty closed subset of a finite group is a subgroup.

- Closure: the subset is closed.
- Identity: suppose $a \in S'$, then $a^{(k)} \in S'$. Since the subset is finite, there must be a period such that $a^{(m)} = a^{(n)}$, hence $a^{(m)}a^{(-n)} = a^{(m-n)} = 1$, therefore the subset must contain the identity.
- Associativity: inherit from the origin group.
- Inverses: suppose $a^{(k)} = 1$, the inverse of element a is $a^{(k-1)}$ since $aa^{(k-1)} = a^{(k)} = 1$.

31.3-4

> Show that if p is prime and e is a positive integer, then
>
> $\phi(p^e) = p^{e-1}(p-1)$.

$\phi(p^e) = p^e \cdot \left(1 - \frac{1}{p}\right) = p^{e-1}(p-1)$.

31.3-5

> Show that for any integer $n > 1$ and for any $a \in \mathbb{Z}_n^*$, the function $f_a : \mathbb{Z}_n^* \to \mathbb{Z}_n^*$ defined by $f_a(x) = ax \mod n$ is a permutation of $\mathbb{Z}_n^*$.

To prove it is a permutation, we need to prove that

- for each element $x \in \mathbb{Z}_n^*$, $f_a(x) \in \mathbb{Z}_n^*$,
- the numbers generated by f_a are distinct.

Since $a \in \mathbb{Z}_n^*$ and $x \in \mathbb{Z}_n^*$, then $f_a(x) = ax \mod n \in \mathbb{Z}_n^*$ by the closure property.

Suppose there are two distinct numbers $x \in \mathbb{Z}_n^*$ and $y \in \mathbb{Z}_n^*$ that $f_a(x) = f_a(y)$,

$$\begin{aligned} f_a(x) &= f_a(y) \\ ax \mod n &= ay \mod n \\ (a \mod n)(x \mod n) &= (a \mod n)(y \mod n) \\ (x \mod n) &= y \mod n \\ x &\equiv y \mod n, \end{aligned}$$

which contradicts the assumption, therefore the numbers generated by f_a are distinct.

31.4 Solving modular linear equations

31.4-1

> Find all solutions to the equation $35x \equiv 10 \pmod{50}$.

$\{6, 16, 26, 36, 46\}$.

31.4-2

> Prove that the equation $ax \equiv ay \pmod n$ implies $x \equiv y \pmod n$ whenever $\gcd(a, n) = 1$. Show that the condition $\gcd(a, n) = 1$ is necessary by supplying a counterexample with $\gcd(a, n) > 1$.

$$d = \gcd(ax, n) = \gcd(x, n)$$

Since $ax \cdot x' + n \cdot y' = d$,

we have

$$x \cdot (ax') + n \cdot y' = d.$$

$$\begin{aligned} x_0 &= x'(ay/d), \\ x_0' &= (ax')(y/d) = x'(ay/d) = x_0. \end{aligned}$$

31.4-3

> Consider the following change to line 3 of the procedure MODULAR-LINEAR-EQUATION-SOLVER:
>
> ```
> 3  x0 = x'(b / d) mod (n / d)
> ```
>
> Will this work? Explain why or why not.

Assume that $x_0 \geq n/d$, then the largest solution is $x_0 + (d-1) \cdot (n/d) \geq d \cdot n/d \geq n$, which is impossible, therefore $x_0 < n/d$.

31.4-4 ⋆

> Let p be prime and $f(x) \equiv f_0 + f_1 x + \cdots + f_t x^t \pmod{p}$ be a polynomial of degree t, with coefficients f_i drawn from $\mathbb{Z}_p$. We say that $a \in \mathbb{Z}_p$ is a ***zero*** of f if $f(a) \equiv 0 \pmod{p}$. Prove that if a is a zero of f, then $f(x) \equiv (x-a)g(x) \pmod{p}$ for some polynomial $g(x)$ of degree $t-1$. Prove by induction on t that if p is prime, then a polynomial $f(x)$ of degree t can have at most t distinct zeros modulo p.

(Omit!)

31.5 The Chinese remainder theorem

31.5-1

> Find all solutions to the equations $x \equiv 4 \pmod{5}$ and $x \equiv 5 \pmod{11}$.

$$\begin{aligned} m_1 &= 11, m_2 = 5. \\ m_1^{-1} &= 1, m_2^{-1} = 9. \\ c_1 &= 11, c_2 = 45. \\ a &= (c_1 \cdot a_1 + c_2 \cdot a_2) \mod (n_1 \cdot n_2) \\ &= (11 \cdot 4 + 45 \cdot 5) \mod 55 = 49. \end{aligned}$$

31.5-2

> Find all integers x that leave remainders $1, 2, 3$ when divided by $9, 8, 7$ respectively.

$10 + 504i, i \in \mathbb{Z}$.

31.5-3

> Argue that, under the definitions of Theorem 31.27, if $\gcd(a, n) = 1$, then
>
> $$(a^{-1} \mod n) \leftrightarrow ((a_1^{-1} \mod n_1), (a_2^{-1} \mod n_2), \ldots, (a_k^{-1} \mod n_k)).$$

$$\gcd(a, n) = 1 \rightarrow \gcd(a, n_i) = 1.$$

31.5-4

> Under the definitions of Theorem 31.27, prove that for any polynomial f, the number of roots of the equation $f(x) \equiv 0 \pmod{n}$ equals the product of the number of roots of each of the equations

> $$f(x) \equiv 0 \pmod{n_1}, f(x) \equiv 0 \pmod{n_2}, \ldots, f(x) \equiv 0 \pmod{n_k}.$$

Based on 31.28–31.30.

31.6 Powers of an element

31.6-1

> Draw a table showing the order of every element in $\mathbb{Z}_{11}^*$. Pick the smallest primitive root g and compute a table giving $\text{ind}_{11,g}(x)$ for all $x \in \mathbb{Z}_{11}^*$.

$g = 2$, $\{1, 2, 4, 8, 5, 10, 9, 7, 3, 6\}$.

31.6-2

> Give a modular exponentiation algorithm that examines the bits of b from right to left instead of left to right.

```
MODULAR-EXPONENTIATION(a, b, n)
    i = 0
    d = 1
    while (1 << i) ≤ b
        if (b & (1 << i)) > 0
            d = (d * a) % n
        a = (a * a) % n
        i = i + 1
    return d
```

31.6-3

> Assuming that you know $\phi(n)$, explain how to compute $a^{-1} \pmod{n}$ for any $a \in \mathbb{Z}_n^*$ using the procedure MODULAR-EXPONENTIATION.

$$\begin{aligned} a^{\phi(n)} &\equiv 1 \pmod{n}, \\ a \cdot a^{\phi(n)-1} &\equiv 1 \pmod{n}, \\ a^{-1} &\equiv a^{\phi(n)-1} \pmod{n}. \end{aligned}$$

31.7 The RSA public-key cryptosystem

31.7-1

> Consider an RSA key set with $p = 11$, $q = 29$, $n = 319$, and $e = 3$. What value of d should be used in the secret key? What is the encryption of the message $M = 100$?

$\phi(n) = (p-1) \cdot (q-1) = 280$.

$d = e^{-1} \pmod{\phi(n)} = 187$.

$P(M) = M^e \pmod{n} = 254$.

$S(C) = C^d \pmod{n} = 254^{187} \pmod{n} = 100$.

31.7-2

> Prove that if Alice's public exponent e is 3 and an adversary obtains Alice's secret exponent d, where $0 < d < \phi(n)$, then the adversary can factor Alice's modulus n in time polynomial in the number of bits in n. (Although you are not asked to prove it, you may be interested to know that this result remains true even if the condition $e = 3$ is removed. See Miller [255].)

$$ed \equiv 1 \mod \phi(n)$$

$$ed - 1 = 3d - 1 = k\phi(n)$$

If $p, q < n/4$, then

$$\phi(n) = n - (p + q) + 1 > n - n/2 + 1 = n/2 + 1 > n/2.$$

$kn/2 < 3d - 1 < 3d < 3n$, then $k < 6$, then we can solve $3d - 1 = n - p - n/p + 1$.

31.7-3 ⋆

> Prove that RSA is multiplicative in the sense that
>
> $P_A(M_1)P_A(M_2) \equiv P_A(M_1, M_2) \pmod{n}$.
>
> Use this fact to prove that if an adversary had a procedure that could efficiently decrypt 1 percent of messages from $\mathbb{Z}_n$ encrypted with P_A, then he could employ a probabilistic algorithm to decrypt every message encrypted with P_A with high probability.

Multiplicative: e is relatively prime to n.

Decrypt: In each iteration randomly choose a prime number m that m is relatively prime to n, if we can decrypt $m \cdot M$, then we can return $m^{-1} M$ since $m^{-1} = m^{n-2}$.

31.8 Primality testing

31.8-1

> Prove that if an odd integer $n > 1$ is not a prime or a prime power, then there exists a nontrivial square root of 1 modulo n.

(Omit!)

31.8-2 ⋆

> It is possible to strengthen Euler's theorem slightly to the form
>
> $a^{\lambda(n)} \equiv 1 \pmod{n}$ for all $a \in \mathbb{Z}_n^*$,
>
> where $n = p_1^{e_1} \cdots p_r^{e_r}$ and $\lambda(n)$ is defined by
>
> $$\lambda(n) = \operatorname{lcm}(\phi(p_1^{e_1}), \ldots, \phi(p_r^{e_r})). \tag{31.42}$$
>
> Prove that $\lambda(n) \mid \phi(n)$. A composite number n is a Carmichael number if $\lambda(n) \mid n - 1$. The smallest Carmichael number is $561 = 3 \cdot 11 \cdot 17$; here, $\lambda(n) = \operatorname{lcm}(2, 10, 16) = 80$, which divides 560. Prove that Carmichael numbers must be both "square-free" (not divisible by the square of any prime) and the product of at least three primes. (For this reason, they are not very common.)

1. Prove that $\lambda(n) \mid \phi(n)$.

   We have

   $$\begin{aligned} n &= p_1^{e_1} \cdots p_r^{e_r} \\ \phi(n) &= \phi(p_1^{e_1}) \times \cdots \times \phi(p_r^{e_r}). \end{aligned}$$

   Thus,

   $$\begin{aligned} &\lambda(n) \mid \phi(n) \\ \Rightarrow\ &\operatorname{lcm}(\phi(p_1^{e_1}, \ldots, \phi(p_r^{e_r})) \mid (\phi(p_1^{e_1}) \times \cdots \times \phi(p_r^{e_r})) \end{aligned}$$

2. Prove that Carmichael numbers must be "square-free" (not divisible by the square of any prime).

Assume $n = p^{\alpha}m$ is a Carmichael number, where $\alpha \geq 2$ and $p \nmid m$.

By the Chinese Remainder Theorem, the system of congruences

$$\begin{aligned} x &\equiv 1 + p \pmod{p}^{\alpha}, \\ x &\equiv 1 \pmod{m} \end{aligned}$$

has a solution a. Note that $\gcd(a, n) = 1$.

Since n is a Carmichael number, we have $a^{n-1} \equiv 1 \pmod{n}$. In particular, $a^{n-1} \equiv 1 \pmod{p^{\alpha}}$; therefore, $a^n \equiv a \pmod{p^2}$.

So, $(1+p)^n \equiv 1 + p \pmod{p^2}$. Expand $(1+p)^n$ modulo p^2 using the binomial theorem. We have

$$(1+p)^n \equiv 1 \pmod{p^2},$$

since the first two terms of the expansion are 1 and np, and the rest of the terms are divisible by p^2.

Thus, $1 \equiv 1 + p \pmod{p^2}$. This is impossible.

Stack Exchange Reference

3. Prove that Carmichael numbers must be the product of at least three primes.

Assume that $n = pq$ is a Carmichael number, where p and q are distant primes and $p < q$.

Then we have

$$\begin{aligned} & q \equiv 1 \pmod{q-1} \\ \Rightarrow & n \equiv pq \equiv p \pmod{q-1} \\ \Rightarrow & n - 1 \equiv p - 1 \pmod{q-1}, \end{aligned}$$

Here, $0 < p - 1 < q - 1$, so $n - 1$ is not divisible by $q - 1$.

Stack Exchange Reference

31.8-3

> Prove that if x is a nontrivial square root of 1, modulo n, then $\gcd(x-1, n)$ and $\gcd(x+1, n)$ are both nontrivial divisors of n.

$$\begin{aligned} x^2 &\equiv 1 \pmod{n}, \\ x^2 - 1 &\equiv 0 \pmod{n}, \\ (x+1)(x-1) &\equiv 0 \pmod{n}. \end{aligned}$$

$n \mid (x+1)(x-1)$, suppose $\gcd(x-1, n) = 1$, then $n \mid (x+1)$, then $x \equiv -1 \pmod{n}$ which is trivial, it contradicts the fact that x is nontrivial, therefore $\gcd(x-1, n) \neq 1$, $\gcd(x+1, n) \neq 1$.

31.9 Integer factorization

31.9-1

> Referring to the execution history shown in Figure 31.7(a), when does \text{POLLARDRHO} print the factor 73 of 1387?

$x = 84$, $y = 814$.

31.9-2

> Suppose that we are given a function $f : \mathbb{Z}_n \to \mathbb{Z}_n$ and an initial value $x_0 \in \mathbb{Z}_n$. Define $x_i = f(x_{i-1})$ for $i = 1, 2, \ldots$. Let t and $u > 0$ be the smallest values such that $x_{t+i} = x_{t+u+i}$ for $i = 0, 1, \ldots$. In the terminology of

> Pollard's rho algorithm, t is the length of the tail and u is the length of the cycle of the rho. Give an efficient algorithm to determine t and u exactly, and analyze its running time.

(Omit!)

31.9-3

> How many steps would you expect POLLARD-RHO to require to discover a factor of the form p^e, where p is prime and $e > 1$?

$\Theta(\sqrt{p})$.

31.9-4 ⋆

> One disadvantage of POLLARD-RHO as written is that it requires one gcd computation for each step of the recurrence. Instead, we could batch the gcd computations by accumulating the product of several x_i values in a row and then using this product instead of x_i in the gcd computation. Describe carefully how you would implement this idea, why it works, and what batch size you would pick as the most effective when working on a β-bit number n.

(Omit!)

Problem 31-1 Binary gcd algorithm

> Most computers can perform the operations of subtraction, testing the parity (odd or even) of a binary integer, and halving more quickly than computing remainders. This problem investigates the ***binary gcd algorithm***, which avoids the remainder computations used in Euclid's algorithm.
>
> **a.** Prove that if a and b are both even, then $\gcd(a, b) = 2 \cdot \gcd(a/2, b/2)$.
>
> **b.** Prove that if a is odd and b is even, then $\gcd(a, b) = \gcd(a, b/2)$.
>
> **c.** Prove that if a and b are both odd, then $\gcd(a, b) = \gcd((a - b)/2, b)$.
>
> **d.** Design an efficient binary gcd algorithm for input integers a and b, where $a \geq b$, that runs in $O(\lg a)$ time. Assume that each subtraction, parity test, and halving takes unit time.

(Omit!)

d.

```
BINARY-GCD(a, b)
    if a < b
        return BINARY-GCD(b, a)
    if b == 0
        return a
    if (a & 1 == 1) and (b & 1 == 1)
        return BINARY-GCD((a - b) >> 1, b)
    if (a & 1 == 0) and (b & 1 == 0)
        return BINARY-GCD(a >> 1, b >> 1) << 1
    if a & 1 == 1
        return BINARY-GCD(a, b >> 1)
    return BINARY-GCD(a >> 1, b)
```

Problem 31-2 Analysis of bit operations in Euclid's algorithm

> **a.** Consider the ordinary "paper and pencil" algorithm for long division: dividing a by b, which yields a quotient q and remainder r. Show that this method requires $O((1+\lg q)\lg b)$ bit operations.
>
> **b.** Define $\mu(a,b) = (1+\lg a)(1+\lg b)$. Show that the number of bit operations performed by EUCLID in reducing the problem of computing $\gcd(a,b)$ to that of computing $\gcd(b, a \mod b)$ is at most $c(\mu(a,b) - \mu(b, a \mod b))$ for some sufficiently large constant $c > 0$.
>
> **c.** Show that $\text{EUCLID}(a,b)$ requires $O(\mu(a,b))$ bit operations in general and $O(\beta^2)$ bit operations when applied to two β-bit inputs.

a.

- Number of comparisons and subtractions: $\lceil \lg a \rceil - \lceil \lg b \rceil + 1 = \lceil \lg q \rceil$.
- Length of subtraction: $\lceil \lg b \rceil$.
- Total: $O((1+\lg q)\lg b)$.

b.

$$\begin{aligned}
& \mu(a,b) - \mu(b, a \mod b) \\
= \;& \mu(a,b) - \mu(b,r) \\
= \;& (1+\lg a)(1+\lg b) - (1+\lg b)(1+\lg r) \\
= \;& (1+\lg b)(\lg a - \lg r) && (\lg r \le \lg b) \\
\ge \;& (1+\lg b)(\lg a - \lg b) \\
= \;& (1+\lg b)(\lg q + 1) \\
\ge \;& (1+\lg q)\lg b
\end{aligned}$$

c. $\mu(a,b) = (1+\lg a)(1+\lg b) \approx \beta^2$

Problem 31-3 Three algorithms for Fibonacci numbers

> This problem compares the efficiency of three methods for computing the nth Fibonacci number F_n, given n. Assume that the cost of adding, subtracting, or multiplying two numbers is $O(1)$, independent of the size of the numbers.
>
> **a.** Show that the running time of the straightforward recursive method for computing F_n based on recurrence (3.22) is exponential in n. (See, for example, the FIB procedure on page 775.)
>
> **b.** Show how to compute F_n in $O(n)$ time using memoization.
>
> **c.** Show how to compute F_n in $O(\lg n)$ time using only integer addition and multiplication. (Hint: Consider the matrix
>
> $$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$
>
> and its powers.)
>
> **d.** Assume now that adding two β-bit numbers takes $\Theta(\beta)$ time and that multiplying two β-bit numbers takes $\Theta(\beta^2)$ time. What is the running time of these three methods under this more reasonable cost measure for the elementary arithmetic operations?

a. In order to solve $\text{FIB}(n)$, we need to compute $\text{FIB}(n-1)$ and $\text{FIB}(n-1)$. Therefore we have the recurrence

$$T(n) = T(n-1) + T(n-2) + \Theta(1).$$

We can get the upper bound of Fibonacci as $O(2^n)$, but this is not the tight upper bound.

The Fibonacci recurrence is defined as

$$F(n) = F(n-1) + F(n-2).$$

The characteristic equation for this function will be

$$\begin{aligned} x^2 &= x + 1 \\ x^2 - x - 1 &= 0. \end{aligned}$$

We can get the roots by quadratic formula: $x = \frac{1 \pm \sqrt{5}}{2}$.

We know the solution of a linear recursive function is given as

$$\begin{aligned} F(n) &= \alpha_1^n + \alpha_2^n \\ &= \left(\frac{1 + \sqrt{5}}{2} \right)^n + \left(\frac{1 - \sqrt{5}}{2} \right)^n, \end{aligned}$$

where α_1 and α_2 are the roots of the characteristic equation.

Since both $T(n)$ and $F(n)$ are representing the same thing, they are asymptotically the same.

Hence,

$$\begin{aligned} T(n) &= \left(\frac{1 + \sqrt{5}}{2} \right)^n + \left(\frac{1 - \sqrt{5}}{2} \right)^n \\ &= \left(\frac{1 + \sqrt{5}}{2} \right)^n \\ &\approx O(1.618)^n. \end{aligned}$$

b. This is same as 15.1-5.

c. Assume that all integer multiplications and additions can be done in $O(1)$. First, we want to show that

$$ \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^k

$$\begin{pmatrix} F_{k-1} & F_k \\ F_k & F_{k+1} \end{pmatrix}$$

. $$

By induction,

$$\begin{aligned} \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{k+1} &= \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^k \\ &= \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} F_{k-1} & F_k \\ F_k & F_{k+1} \end{pmatrix}^k \\ &= \begin{pmatrix} F_k & F_{k+1} \\ F_{k-1} + F_k & F_k + F_{k+1} \end{pmatrix} \\ &= \begin{pmatrix} F_k & F_{k+1} \\ F_{k+1} & F_{k+2} \end{pmatrix}. \end{aligned}$$

We show that we can compute the given matrix to the power $n - 1$ in time $O(\lg n)$, and the bottom right entry is F_n.

We should note that by 8 multiplications and 4 additions, we can multiply any two 2 by 2 matrices, that means matrix multiplications can be done in constant time. Thus we only need to bound the number of those in our algorithm.

It takes $O(\lg n)$ to run the algorithm $\text{MATRIX-POW}(A, n - 1)$ becasue we half the value of n in each step, and within each step, we perform a constant amount of calculation.

The recurrence is

$$T(n) = T(n/2) + \Theta(1).$$

```
MATRIX-POW(A, n)
    if n % 2 == 1
        return A * MATRIX-POW(A^2, (n - 1) / 2)
    return MATRIX-POW(A^2, n / 2)
```

d. First, we should note that $\beta = O(\log n)$.

- For part (a),

  We naively add a β-bit number which is growing exponentially each time, so the recurrence becomes

  $$\begin{aligned} T(n) &= T(n-1) + T(n-2) + \Theta(\beta) \\ &= T(n-1) + T(n-2) + \Theta(\log n), \end{aligned}$$

  which has the same solution $T(n) = O\left(\frac{1+\sqrt{5}}{2}\right)^n$ since $\Theta(\log n)$ can be absorbed by exponential time.

- For part (b),

  The recurrence of the memoized verstion becomes

  $$M(n) = M(n-1) + \Theta(\beta).$$

  This has a solution of $\sum_{i=2}^{n} \beta = \Theta(n\beta) = \Theta(2^{\beta} \cdot \beta)$ or $\Theta(n \log n)$.

- For part (c),

  We perform a constant number of both additions and multiplications. The recurrence becomes

  $$P(n) = P(n/2) + \Theta(\beta^2),$$

  which has a solution of $\Theta(\log n \cdot \beta^2) = \Theta(\beta^3)$ or $\Theta(\log^3 n)$.

Problem 31-4 Quadratic residues

> Let p be an odd prime. A number $a \in Z_p^*$ is a ***quadratic residue*** if the equation $x^2 = a \pmod{p}$ has a solution for the unknown x.
>
> **a.** Show that there are exactly $(p-1)/2$ quadratic residues, modulo p.
>
> **b.** If p is prime, we define the ***Legendre symbol*** $\left(\frac{a}{p}\right)$, for $a \in \mathbb{Z}_p^*$, to be 1 if a is a quadratic residue modulo p and -1 otherwise. Prove that if $a \in \mathbb{Z}_p^*$, then
>
> $$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod{p}.$$
>
> Give an efficient algorithm that determines whether a given number a is a quadratic residue modulo p. Analyze the efficiency of your algorithm.
>
> **c.** Prove that if p is a prime of the form $4k+3$ and a is a quadratic residue in $\mathbb{Z}_b^*$, then $a^{k+1} \mod p$ is a square root of a, modulo p. How much time is required to find the square root of a quadratic residue a modulo p?
>
> **d.** Describe an efficient randomized algorithm for finding a nonquadratic residue, modulo an arbitrary prime p, that is, a member of $\mathbb{Z}_p^*$ that is not a quadratic residue. How many arithmetic operations does your algorithm require on average?

(Omit!)