Review of Discrete Math

Xiao Zhang

2020-12-15

Chapter 3 Algorithms

- What is an algorithms?(Difiniation)
- Some examples of algorithm
  - sort
  - search
- What is a good algorithm?(growth)
  - Big-O Notation
  - Big-O Estimates for Important Functions
  - Big-Omega and Big-Theta Notation

Definition

- **Definition**: An *algorithm* is a finite set of precise instructions for performing a computation or for solving a problem.
- Properties:
  - *Input*: An algorithm has input values from a specified set.
  - *Output*: From the input values, the algorithm produces the output values from a specified set. The output values are the solution.
  - *Correctness*: An algorithm should produce the correct output values for each set of input values.
  - *Finiteness*: An algorithm should produce the output after a finite number of steps for any input.
  - *Effectiveness*: It must be possible to perform each step of the algorithm correctly and in a finite amount of time.
  - *Generality*: The algorithm should work for all problems of the desired form.

Algorithm samples(search)

- Search

procedure binary search(x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
$i := 1$ {i is the left endpoint of interval}
$j := n$ {j is right endpoint of interval}
while $i < j$
  $m := \lfloor (i + j)/2 \rfloor$
  if $x > a_m$ then $i := m + 1$
  else $j := m$
if $x = a_i$ **then** *location* := i
else *location* := 0
return *location*{location is the subscript i of the term a_i equal to x, or 0 if x is not found}

Binary Search

procedure *max*($a_1, a_2, \ldots, a_n$: integers)
$max := a_1$
for $i := 2$ to n
  if $max < a_i$ then $max := a_i$
return *max*{*max* is the largest element}

Search Max

procedure *linear search*(x:integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
  $i := i + 1$
if $i \leq n$ **then** *location* := i
else *location* := 0
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

Linear Search

Algorithm samples(sorting)

- Sorting:To *sort* the elements of a list is to put them in increasing order (numerical order, alphabetic, and so on).

procedure *bubblesort*(a_1,…,a_n: real numbers with $n \geq 2$)

  for $i := 1$ to $n - 1$

    for $j := 1$ to $n - i$

      if $a_j > a_{j+1}$ **then** interchange a_j and a_{j+1}

{a_1,…, a_n is now in increasing order}

Bubble sort

procedure *insertion sort* (a_1,…,a_n: real numbers with $n \geq 2$)

  for $j := 2$ to n

    $i := 1$

    while $a_j > a_i$

      $i := i + 1$

    $m := a_j$

    for $k := 0$ to $j - i - 1$

      $a_{j-k} := a_{j-k-1}$

    $a_i := m$

{Now a_1,…,a_n is in increasing order}

Insertion sort

Algorithm samples(sorting)

- Greedy Algorithms:Greedy change-making algorithm for *n* cents. The algorithm works with any coin denominations $c_1, c_2, \ldots, c_r$.

procedure *change*(c_1, c_2, …, c_r: values of coins, where $c_1 > c_2 > \ldots > c_r$;
n: a positive integer)
for $i := 1$ to r
$d_i := 0$ [d_i counts the coins of denomination c_i]
while $n \geq c_i$
$d_i := d_i + 1$ [add a coin of denomination c_i]
$n = n - c_i$
[d_i counts the coins c_i]

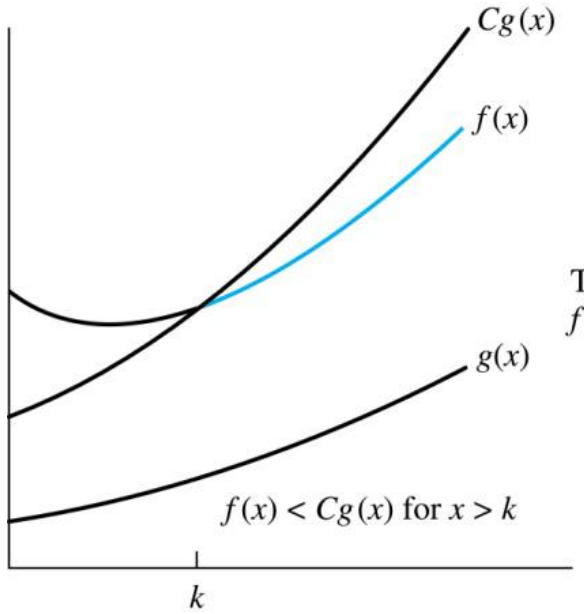
Big-O notation

- Big-O

  Let *f* and *g* be functions from the set of integers or the set of real numbers to the set of real numbers. We say that *f*(*x*) is *O*(*g*(*x*)) if there are constants *C* and *k* such that

$$|f(x)| \leq C|g(x)|$$

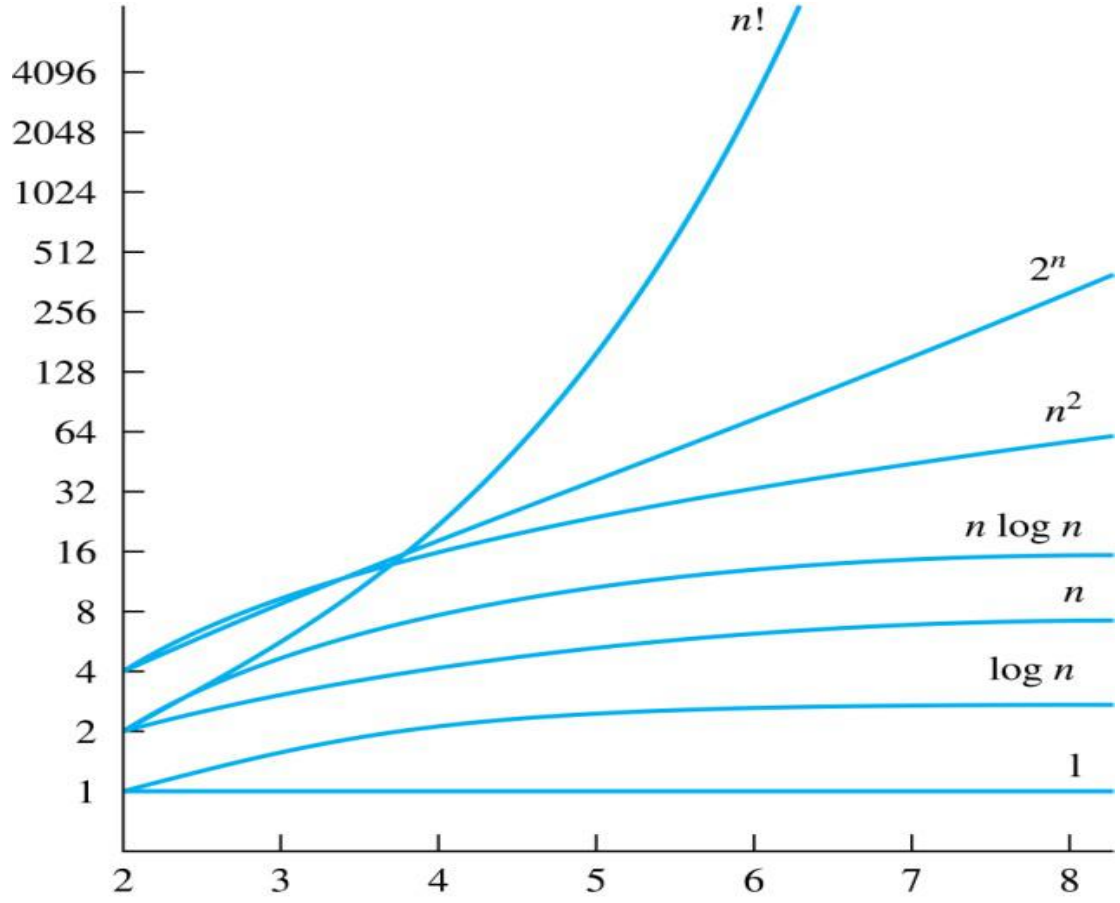
  whenever $x > k$.



The part of the graph of $f(x)$ that satisfies $f(x) < Cg(x)$ is shown in color.

Big-O problems

- Show f(x) is Big-O g(x)
- Big-O of polynomial functions
  - highest order
- Growth of different functions

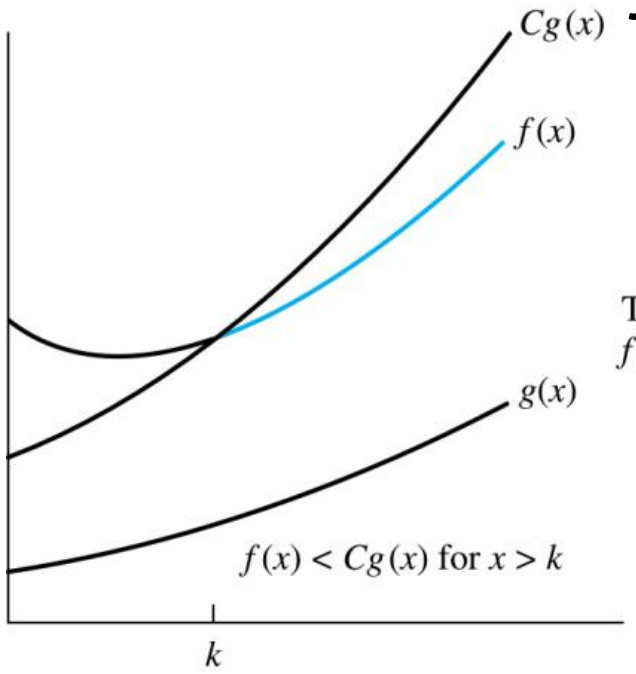


Big-Omega

Definition: Let *f* and *g* be functions from the set of integers or the set of real numbers to the set of real numbers. We say that

$$f(x) \text{ is } \Omega(g(x))$$

if there are constants *C* and *k* such that $|f(x)| \geq C|g(x)|$ when *x* > *k*.

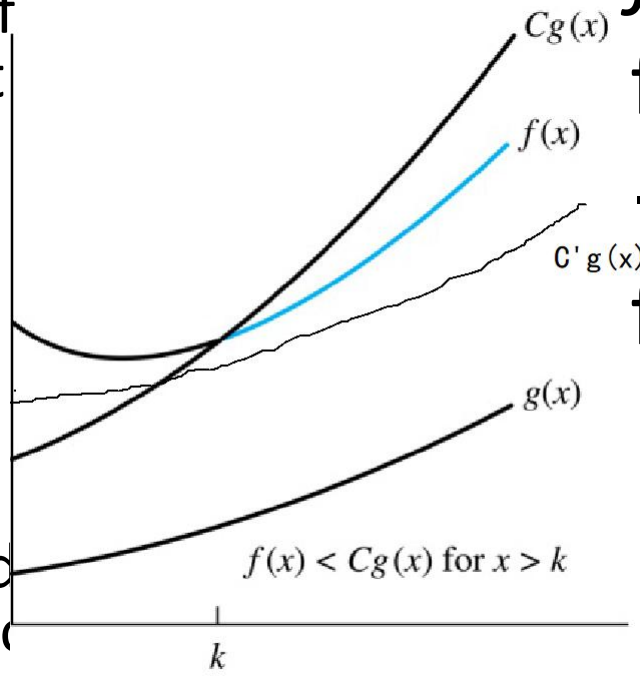


The part of the graph of $f(x)$ that satisfies $f(x) < Cg(x)$ is shown in color.

f(*x*) is *O*(*g*(*x*))
g(x) is Ω(f(x))

Big-Theta

- Let *f* and *g* be functions from the set of integers or the set of real numbers to t set of real numbers. The function
  if $f(x)$ is $\Theta(g(x))$
  $f(x)$ is $O(g(x))$ and
  $f(x)$ is $\Omega(g(x))$



f(x) is O(g(x))
f(x) is Ω(g(x))
-->
f(x) is Θ(g(x))

- We say that "f is big-Theta of *g*(*x*)" and also that "*f*(*x*) is of *order g*(*x*)" and also that "*f*(*x*) and *g*(*x*) are of
- the *same order*."

Complexity

- Time Complexity

Example: How many additions of integers and multiplications of integers are used by the matrix multiplication algorithm to multiply two n n matrices.

Solution: There are n^2 entries in the product. Finding each entry requires n multiplications and $n-1$ additions. Hence, n^3 multiplications and $n^2(n-1)$ additions are used.

Hence, the complexity of matrix multiplication is $O(n^3)$.

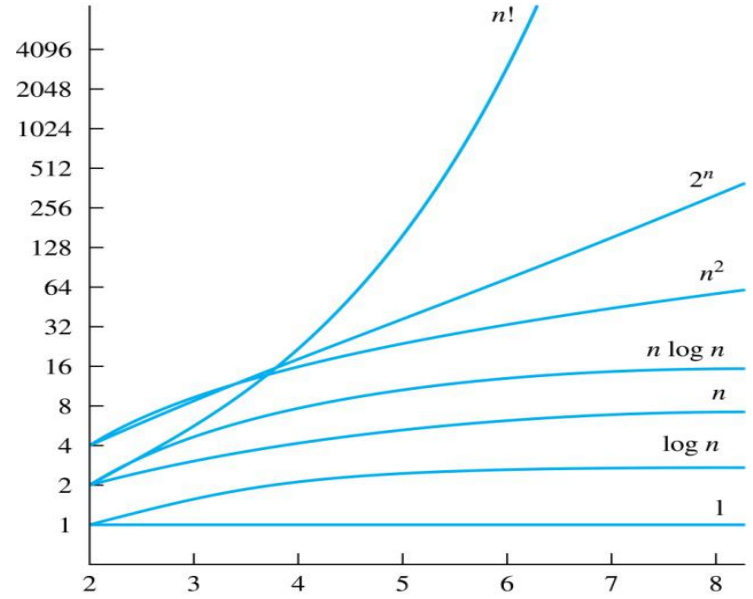
Worst-Case Complexity

- **Example**: Determine the time complexity of the linear search algorithm.

procedure *linear search*(x:integer,
$a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
  $i := i + 1$
if $i \leq n$ **then** *location* := i
else *location* := 0
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

TABLE 1 Commonly Used Terminology for the Complexity of Algorithms.

Complexity	*Terminology*
$\Theta(1)$	Constant complexity
$\Theta(\log n)$	Logarithmic complexity
$\Theta(n)$	Linear complexity
$\Theta(n \log n)$	Linearithmic complexity
$\Theta(n^b)$	Polynomial complexity
$\Theta(b^n)$, where $b > 1$	Exponential complexity
$\Theta(n!)$	Factorial complexity



Growth of functions

- Suppose that f1(x) is O(g1(x)) and that f2(x) is O(g2(x)). Then (f1 + f2)(x) is O(max(|g1(x)|, |g2(x)|))
- Suppose that f1(x) is O(g1(x)) and f2(x) is O(g2(x)). Then (f1f2)(x) is O(g1(x)g2(x)).

Exercise

- § 3.1 2, 9,14,46,52
  §3.2 2,6,16,26,34,72
  §3.3 18,36

Chapter 4 Number Theory

- Divisibility and Modular Arithmetic
- Integer Representations and Algorithms
- Primes and Greatest Common Divisors
- Solving Congruences
- Applications of Congruences
- Cryptography

Divisibility

Definition: If a and b are integers with $a \neq 0$, then *a divides b* if there exists an integer c such that $b = ac$.

- When a divides b we say that a is a *factor* or *divisor* of b and that b is a multiple of a.
- The notation $a \mid b$ denotes that a divides b.
- If $a \mid b$, then b/a is an integer.
- If a does not divide b, we write $a \nmid b$.

Division Algorithm: If a is an integer and d a positive integer, then there are unique integers q and r, with $0 \leq r < d$, such that $a = dq + r$ (*proved in Section* 5.2).

- d is called the *divisor*.
- a is called the *dividend*.
- q is called the *quotient*.
- r is called the *remainder*.

Definitions of Functions **div** and **mod**

$q = a$ **div** d

$r = a$ **mod** d

Examples:

Theorem 1: Let a, b, and c be integers, where $a \neq 0$.

i. If $a \mid b$ and $a \mid c$, then $a \mid (b + c)$;

ii. If $a \mid b$, then $a \mid bc$ for all integers c;

iii. If $a \mid b$ and $b \mid c$, then $a \mid c$.

Proof: (i) Suppose $a \mid b$ and $a \mid c$, then it follows that there are integers s and t with $b = as$ and $c = at$. Hence,

$b + c = as + at = a(s + t)$. Hence, $a \mid (b + c)$

Notice:

a=dq+r

r is 0 or a positive number, it's unique

-101=d(-10)+r

Congruence Relation

Definition: If a and b are integers and m is a positive integer, then a is *congruent* to b *modulo* m if m divides $a - b$.

- The notation $a \equiv b \pmod{m}$ says that a is congruent to b modulo m.
- We say that $a \equiv b \pmod{m}$ is a *congruence* and that m is its *modulus*.
- Two integers are congruent mod m if and only if they have the same remainder when divided by m.
- If a is not congruent to b modulo m, we write $a \not\equiv b \pmod{m}$

Theorem 4: Let m be a positive integer. The integers a and b are congruent modulo m if and only if there is an integer k such that $a = b + km$.

Theorem 5: Let m be a positive integer. If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then

$a + c \equiv b + d \pmod{m}$ and $ac \equiv bd \pmod{m}$

Proof:

- Because $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, by Theorem 4 there are integers s and t with $b = a + sm$ and $d = c + tm$.
- Therefore,
  - $b + d = (a + sm) + (c + tm) = (a + c) + m(s + t)$ and
  - $b\,d = (a + sm)(c + tm) = ac + m(at + cs + stm)$.
- Hence, $a + c \equiv b + d \pmod{m}$ and $ac \equiv bd \pmod{m}$.

- Multiplying both sides of a valid congruence by an integer preserves validity.
  If $a \equiv b \pmod{m}$ holds then $c \cdot a \equiv c \cdot b \pmod{m}$, where c is any integer, holds by Theorem 5 with $d = c$.
- Adding an integer to both sides of a valid congruence preserves validity.
  If $a \equiv b \pmod{m}$ holds then $c + a \equiv c + b \pmod{m}$, where c is any integer, holds by Theorem 5 with $d = c$.
- Dividing a congruence by an integer DOES NOT always produce a valid congruence.

Corollary: Let m be a positive integer and let a and b be integers. Then

$(a + b) \bmod m = ((a \bmod m) + (b \bmod m)) \bmod m$

and $ab \bmod m = ((a \bmod m)(b \bmod m)) \bmod m$.

Arithmetic Modulo *m*

Definitions: Let $\mathbf{Z}_m$ be the set of nonnegative integers less than m: $\{0,1, \ldots, m-1\}$

- The operation $+_m$ is defined as $a +_m b = (a + b)$ **mod** m. This is *addition modulo m*.
- The operation $\cdot_m$ is defined as $a \cdot_m b = (a * b)$ **mod** m. This is *multiplication modulo m*.
- Using these operations is said to be doing *arithmetic modulo m*.

The operations $+_m$ and $\cdot_m$ satisfy many of the same properties as ordinary addition and multiplication.

- *Closure(闭包)*: If a and b belong to $\mathbf{Z}_m$, then $a +_m b$ and $a \cdot_m b$ belong to $\mathbf{Z}_m$.
- *Associativity(结合律)*: If a, b, and c belong to $\mathbf{Z}_m$, then $(a +_m b) +_m c = a +_m (b +_m c)$ and $(a \cdot_m b) \cdot_m c = a \cdot_m (b \cdot_m c)$.
- *Commutativity(交换律)*: If a and b belong to $\mathbf{Z}_m$, then $a +_m b = b +_m a$ and $a \cdot_m b = b \cdot_m a$.
- *Identity elements(幺元)*: The elements 0 and 1 are identity elements for addition and multiplication modulo m, respectively.
  - If a belongs to $\mathbf{Z}_m$, then $a +_m 0 = a$ and $a \cdot_m 1 = a$.
- *Additive inverses*（加法逆元）: If $a \neq 0$ belongs to $\mathbf{Z}_m$, then $m - a$ is the additive inverse of a modulo m and 0 is its own additive inverse.
  - $a +_m (m - a) = 0$ and $0 +_m 0 = 0$
- *Distributivity(分配律)*: If a, b, and c belong to $\mathbf{Z}_m$, then
  - $a \cdot_m (b +_m c) = (a \cdot_m b) +_m (a \cdot_m c)$ and
  - $(a +_m b) \cdot_m c = (a \cdot_m c) +_m (b \cdot_m c)$.

Integer Representations

- Base b Expansions
  - Decimal/Binary/Octal/Hexadecimal
- Binary operations
  - add/multiplcation/modula/Exponentiation

Theorem 1: Let b be a positive integer greater than 1. Then if n is a positive integer, it can be expressed uniquely in the form:

$$n = a_k b^k + a_{k-1} b^{k-1} + \ldots + a_1 b + a_0$$

where k is a nonnegative integer, $a_0, a_1, \ldots a_k$ are nonnegative integers less than b, and $a_k \neq 0$. The a_j, $j = 0, \ldots, k$ are called the base-b digits of the representation.

To construct the base b expansion of an integer n:

- Divide n by b to obtain a quotient and remainder.
  $n = bq_0 + a_0 \quad 0 \leq a_0 \leq b$
- The remainder, a_0, is the rightmost digit in the base b expansion of n. Next, divide q_0 by b.
  $q_0 = bq_1 + a_1 \quad 0 \leq a_1 \leq b$
- The remainder, a_1, is the second digit from the right in the base b expansion of n.
- Continue by successively dividing the quotients by b, obtaining the additional base b digits as the remainder. The process terminates when the quotient is 0.

TABLE 1 Hexadecimal, Octal, and Binary Representation of the Integers 0 through 15.

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Octal	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
Binary	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111

Initial 0s are not shown

Each octal digit corresponds to a block of 3 binary digits.
Each hexadecimal digit corresponds to a block of 4 binary digits.
So, conversion between binary, octal, and hexadecimal is easy.

Example:

convert 1001101010 to decimal/octal/hexadecimal

convert 985 to binary/ to $(?)_3$

Primes and Greatest Common Divisors

- Prime Numbers and their Properties
- Greatest Common Divisors and Least Common Multiples
- The Euclidian Algorithm
- gcds as Linear Combinations

Primes

Definition: A positive integer p greater than 1 is called *prime* if the only positive factors of p are 1 and p. A positive integer that is greater than 1 and is not prime is called *composite*.

Theorem: There are infinitely many primes. (Euclid)

Euclid （欧几里得）
(325 B.C.E. – 265 B.C.E.)

Proof: Assume finitely many primes: $p_1, p_2, \ldots, p_n$

- Let $q = p_1 p_2 \cdots p_n + 1$
- Either q is prime or by the fundamental theorem of arithmetic it is a product of primes.
  - But none of the primes p_j divides q since if $p_j \mid q$, then p_j divides $q - p_1 p_2 \cdots p_n = 1$.
  - Hence, there is a prime not on the list $p_1, p_2, \ldots, p_n$. It is either q, or if q is composite, it is a prime factor of q. This contradicts the assumption that $p_1, p_2, \ldots, p_n$ are all the primes.
- Consequently, there are infinitely many primes.

Theorem: Every positive integer greater than 1 can be written uniquely as a prime or as the product of two or more primes where the prime factors are written in order of nondecreasing size.

Prime Number Theorem: The ratio of the number of primes not exceeding x and $x/\ln x$ approaches 1 as x grows without bound. ($\ln x$ is the natural logarithm of x)

- The theorem tells us that the number of primes not exceeding x, can be approximated by $x/\ln x$.
- The odds that a randomly selected positive integer less than n is prime are approximately $(n/\ln n)/n = 1/\ln n$.

Greatest Common Divisor （GCD）

- **Definition**: The integers *a* and *b* are *relatively prime* if their greatest common divisor is 1.
- **Definition**: The least common multiple of the positive integers *a* and *b* is the smallest positive integer that is divisible by both *a* and *b*. It is denoted by lcm(*a*,*b*).

Suppose the prime factorizations of *a* and *b* are:

$$a = p_1^{a_1} p_2^{a_2} \cdots p_n^{a_n}, \qquad b = p_1^{b_1} p_2^{b_2} \cdots p_n^{b_n},$$

where each exponent is a nonnegative integer, and where all primes occurring in either prime factorization are included in both. Then:

$$\gcd(a,b) = p_1^{\min(a_1,b_1)} p_2^{\min(a_2,b_2)} \cdots p_n^{\min(a_n,b_n)}.$$

Lemma 1: Let $a = bq + r$, where a, b, q, and r are integers. Then $\gcd(a,b) = \gcd(b,r)$.

Proof:

- Suppose that *d* divides both *a* and *b*. Then *d* also divides $a - bq = r$ (by Theorem 1 of Section 4.1). Hence, any common divisor of *a* and *b* must also be a common divisor of *b* and *r*.
- Suppose that *d* divides both *b* and *r*. Then *d* also divides $bq + r = a$. Hence, any common divisor of *a* and *b* must also be a common divisor of *b* and *r*.
- Therefore, $\gcd(a,b) = \gcd(b,r)$. ◀

Very important!!!

The Euclidian algorithm(欧几里得算法，辗转相除法) is an efficient method for computing the greatest common divisor of two integers. It is based on the idea that gcd(*a*,*b*) is equal to gcd(*a*,*c*) when $a > b$ and *c* is the remainder when a is divided by *b*.

Example: Find gcd(91, 287):

- $287 = 91 \cdot 3 + 14$ Divide 287 by 91
- $91 = 14 \cdot 6 + 7$ Divide 91 by 14
- $14 = 7 \cdot 2 + 0$ Divide 14 by 7 (Stopping condition)

Euclid
(325 B.C.E. – 265 B.C.E.)

Exercise：
find gcd(12306,512)
find lcm(211,985)

gcds as Linear Combinations

Bézout's Theorem: If a and b are positive integers, then there exist integers s and t such that $\gcd(a,b) = sa + tb$.

(*proof in exercises of Section* 5.2)

Definition: If a and b are positive integers, then integers s and t such that $\gcd(a,b) = sa + tb$ are called *Bézout coefficients* of a and b. The equation $\gcd(a,b) = sa + tb$ is called *Bézout's identity.*

Dividing both sides of a valid congruence by an integer does not always produce a valid congruence (see Section 4.1).

But dividing by an integer relatively prime to the modulus does produce a valid congruence:

Theorem 7: Let m be a positive integer and let a, b, and c be integers. If $ac \equiv bc \pmod{m}$ and $\gcd(c,m) = 1$, then $a \equiv b \pmod{m}$.

Proof: Since $ac \equiv bc \pmod{m}$, $m \mid ac - bc = c(a - b)$ by Lemma 2 and the fact that $\gcd(c,m) = 1$, it follows that $m \mid a - b$. Hence, $a \equiv b \pmod{m}$.

It's useful to find inverse of modulo

Solving Congruences

Definition: A congruence of the form

$$ax \equiv b \pmod{m},$$

where m is a positive integer, a and b are integers, and x is a variable, is called a *linear congruence*.

- The solutions to a linear congruence $ax \equiv b \pmod{m}$ are all integers x that satisfy the congruence.

Definition: An integer $\bar{a}$ such that $\bar{a}a \equiv 1 \pmod{m}$ is said to be an *inverse* of a modulo m.

The Euclidean algorithm and Bézout coefficients give us a systematic approaches to finding inverses.

Example: Find an inverse of 101 modulo 42620.

Solution: First use the Euclidian algorithm to show that gcd(101,42620) = 1.

$$42620 = 45 \cdot 101 + 75$$
$$101 = 1 \cdot 75 + 26$$
$$75 = 2 \cdot 26 + 23$$
$$26 = 1 \cdot 23 + 3$$
$$23 = 7 \cdot 3 + 2$$
$$3 = 1 \cdot 2 + 1$$
$$2 = 2 \cdot 1$$

Since the last nonzero remainder is 1, gcd(101,4260) = 1

Working Backwards:

$$1 = 3 - 1 \cdot 2$$
$$1 = 3 - 1 \cdot (23 - 7 \cdot 3) = -1 \cdot 23 + 8 \cdot 3$$
$$1 = -1 \cdot 23 + 8 \cdot (26 - 1 \cdot 23) = 8 \cdot 26 - 9 \cdot 23$$
$$1 = 8 \cdot 26 - 9 \cdot (75 - 2 \cdot 26) = 26 \cdot 26 - 9 \cdot 75$$
$$1 = 26 \cdot (101 - 1 \cdot 75) - 9 \cdot 75$$
$$= 26 \cdot 101 - 35 \cdot 75$$
$$1 = 26 \cdot 101 - 35 \cdot (42620 - 45 \cdot 101)$$
$$= -35 \cdot 42620 + 1601 \cdot 101$$

Bézout coefficients : -35 and 1601

1601 is an inverse of 101 modulo 42620

Theorem 1: If a and m are relatively prime integers and $m > 1$, then an inverse of a modulo m exists. Furthermore, this inverse is unique modulo m. (This means that there is a unique positive integer $\bar{a}$ less than m that is an inverse of a modulo m and every other inverse of a modulo m is congruent to $\bar{a}$ modulo m.)

Proof: Since $\gcd(a,m) = 1$, by Theorem 6 of Section 4.3, there are integers s and t such that $sa + tm = 1$.

- Hence, $sa + tm \equiv 1 \pmod{m}$.
- Since $tm \equiv 0 \pmod{m}$, it follows that $sa \equiv 1 \pmod{m}$
- Consequently, s is an inverse of a modulo m. ◀
- The uniqueness of the inverse is Exercise 7.

We can solve the congruence $ax \equiv b \pmod{m}$ by multiplying both sides by $\bar{a}$.

Chinese remender Theorem

Theorem 2: (*The Chinese Remainder Theorem*) Let $m_1, m_2, \ldots, m_n$ be pairwise relatively prime positive integers greater than one and $a_1, a_2, \ldots, a_n$ arbitrary integers. Then the system

$x \equiv a_1 \pmod{m_1}$
$x \equiv a_2 \pmod{m_2}$
.
.
.
$x \equiv a_n \pmod{m_n}$

has a unique solution modulo $m = m_1 m_2 \cdots m_n$.

(That is, there is a solution x with $0 \leq x < m$ and all other solutions are congruent modulo m to this solution.)

To construct a solution first let $M_k = m/m_k$ for $k = 1,2,\ldots,n$ and $m = m_1 m_2 \cdots m_n$.

Since $\gcd(m_k, M_k) = 1$, by Theorem 1, there is an integer y_k, an inverse of M_k modulo m_k, such that

$$M_k y_k \equiv 1 \pmod{m_k}.$$

Form the sum

$$x = a_1 M_1 y_1 + a_2 M_2 y_2 + \cdots + a_n M_n y_n.$$

Note that because $M_j \equiv 0 \pmod{m_k}$ whenever $j \neq k$, all terms except the kth term in this sum are congruent to 0 modulo m_k.

Because $M_k y_k \equiv 1 \pmod{m_k}$, we see that $x \equiv a_k M_k y_k \equiv a_k \pmod{m_k}$, for $k = 1,2,\ldots,n$.

Hence, x is a simultaneous solution to the n congruences.

$x \equiv a_1 \pmod{m_1}$
$x \equiv a_2 \pmod{m_2}$
.
.
.
$x \equiv a_n \pmod{m_n}$

◀

Example: Consider the 3 congruences from Sun-Tsu's problem:

$x \equiv 2 \pmod 3$, $x \equiv 3 \pmod 5$, $x \equiv 2 \pmod 7$.

- Let $m = 3 \cdot 5 \cdot 7 = 105$, $M_1 = m/3 = 35$, $M_2 = m/5 = 21$, $M_3 = m/7 = 15$.
- We see that
  - 2 is an inverse of $M_1 = 35$ modulo 3 since $35 \cdot 2 \equiv 2 \cdot 2 \equiv 1 \pmod 3$
  - 1 is an inverse of $M_2 = 21$ modulo 5 since $21 \equiv 1 \pmod 5$
  - 1 is an inverse of $M_3 = 15$ modulo 7 since $15 \equiv 1 \pmod 7$
- Hence,

$$x = a_1 M_1 y_1 + a_2 M_2 y_2 + a_3 M_3 y_3$$
$$= 2 \cdot 35 \cdot 2 + 3 \cdot 21 \cdot 1 + 2 \cdot 15 \cdot 1 = 233 \equiv 23 \pmod{105}$$

- We have shown that 23 is the smallest positive integer that is a simultaneous solution. Check it!

Exercise:

$x \equiv 2 \pmod 3$, $x \equiv 3 \pmod 7$, $x \equiv 3 \pmod{10}$

Fermat's Little Theorem

- **Theorem 3**: (*Fermat's Little Theo*rem) If p is prime and a is an integer not divisible by p, then $a^{p-1} \equiv 1 \pmod{p}$
- Furthermore, for every integer a we have $a^p \equiv a \pmod{p}$
- Exercise:
  - Find 7^{2020} **mod** 11.
  - Find 20^{2020} mod 13.

Definition: A primitive root modulo a prime p is an integer r in $\mathbf{Z}_p$ such that every nonzero element of $\mathbf{Z}_p$ is a power of r.

Example: Since every element of $\mathbf{Z}_{11}$ is a power of 2, 2 is a primitive root of 11.

Powers of 2 modulo 11: $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, $2^4 = 5$, $2^5 = 10$, $2^6 = 9$, $2^7 = 7$, $2^8 = 3$, $2^{10} = 1$.

Example: Since not all elements of $\mathbf{Z}_{11}$ are powers of 3, 3 is not a primitive root of 11.

Powers of 3 modulo 11: $3^1 = 3$, $3^2 = 9$, $3^3 = 5$, $3^4 = 4$, $3^5 = 1$, and the pattern repeats for higher powers.

Important Fact: There is a primitive root modulo p for every prime number p.

Applications of Congruences

- Hashing Functions

Definition: A *hashing function* h assigns memory location $h(k)$ to the record that has k as its key.
- A common hashing function is $h(k) = k$ **mod** m, where m is the number of memory locations.
- Because this hashing function is onto, all memory locations are possible.

- Pseudorandom Numbers

Pseudorandom numbers are not truly random since they are generated by systematic methods.

The *linear congruential method* is one commonly used procedure for generating pseudorandom numbers.

Four integers are needed: the *modulus* m, the *multiplier* a, the *increment* c, and *seed* x_0, with $2 \le a < m, 0 \le c < m, 0 \le x_0 < m$.

We generate a sequence of pseudorandom numbers $\{x_n\}$, with $0 \le x_n < m$ for all n, by successively using the recursively defined function

$$x_{n+1} = (ax_n + c) \bmod m.$$

- Check Digits

A common method of detecting errors in strings of digits is to add an extra digit at the end, which is evaluated using a function. If the final digit is not correct, then the string is assumed not to be correct.

Understand the algorithm of the check digits and calcluate carefully.

Example: Retail products are identified by their *Universal Product Codes (UPCs)*. Usually these have 12 decimal digits, the last one being the check digit. The check digit is determined by the congruence:

$$3x_1 + x_2 + 3x_3 + x_4 + 3x_5 + x_6 + 3x_7 + x_8 + 3x_9 + x_{10} + 3x_{11} + x_{12} \equiv 0 \pmod{10}.$$

a. Suppose that the first 11 digits of the UPC are 79357343104. What is the check digit?

b. Is 041331021641 a valid UPC?

Solution:

a. $3\cdot7 + 9 + 3\cdot3 + 5 + 3\cdot7 + 3 + 3\cdot4 + 3 + 3\cdot1 + 0 + 3\cdot4 + x_{12} \equiv 0 \pmod{10}$

$21 + 9 + 9 + 5 + 21 + 3 + 12 + 3 + 3 + 0 + 12 + x_{12} \equiv 0 \pmod{10}$

$98 + x_{12} \equiv 0 \pmod{10}$

$x_{12} \equiv 0 \pmod{10}$ So, the check digit is 2.

b. $3\cdot0 + 4 + 3\cdot1 + 3 + 3\cdot3 + 1 + 3\cdot0 + 2 + 3\cdot1 + 6 + 3\cdot4 + 1 \equiv 0 \pmod{10}$

$0 + 4 + 3 + 3 + 9 + 1 + 0 + 2 + 3 + 6 + 12 + 1 = 44 \equiv 4 \not\equiv \pmod{10}$

Hence, 041331021641 is not a valid UPC.

Cryptography

- Classical Cryptography
  - Caesar Cipher $f(p) = (p + 3) \bmod 26$
  - Shift Cipher $f(p) = (p + k) \bmod 26$
  - Affine Ciphers $f(p) = (ap + b) \bmod 26$
  - Block Ciphers

    A simple type of block cipher is called the *transposition cipher*. The key is a permutation σ of the set $\{1,2,\ldots,m\}$, where m is an integer, that is a one-to-one function from $\{1,2,\ldots,m\}$ to itself.

    To encrypt a message, split the letters into blocks of size m, adding additional letters to fill out the final block. We encrypt $p_1,p_2,\ldots,p_m$ as $c_1,c_2,\ldots,c_m = p_{\sigma(1)},p_{\sigma(2)},\ldots,p_{\sigma(m)}$.

    To decrypt the $c_1,c_2,\ldots,c_m$ transpose the letters using the inverse permutation σ^{-1}.

- Cryptosystems

The process of recovering plaintext from ciphertext without knowledge both of the encryption method and the key is known as *cryptanalysis* or *breaking codes*.

An important tool for cryptanalyzing ciphertext produced with a affine ciphers is the relative frequencies of letters. The nine most common letters in the English texts are E 13%, T 9%, A 8%, O 8%, I 7%, N 7%, S 7%, H 6%, and R 6%.

To analyze ciphertext:

- Find the frequency of the letters in the ciphertext.
- Hypothesize that the most frequent letter is produced by encrypting E.
- If the value of the shift from E to the most frequent letter is k, shift the ciphertext by $-k$ and see if it makes sense.
- If not, try T as a hypothesis and continue.

Definition: A *cryptosystem* is a five-tuple $(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$, where

- $\mathcal{P}$ is the set of plainntext strings,
- $\mathcal{C}$ is the set of ciphertext strings,
- $\mathcal{K}$ is the *keyspace* (set of all possible keys),
- $\mathcal{E}$ is the set of encription functions, and
- $\mathcal{D}$ is the set of decryption functions.

The encryption function in $\mathcal{E}$ corresponding to the key k is denoted by E_k and the description function in $\mathcal{D}$ that decrypts cipher text encrypted using E_k is denoted by D_k. Therefore:

$$D_k(E_k(p)) = p, \text{ for all plaintext strings } p.$$

Cryptography 2

- RSA

All classical ciphers, including shift and affine ciphers, are *private key cryptosystems*. Knowing the encryption key allows one to quickly determine the decryption key.

All parties who wish to communicate using a private key cryptosystem must share the key and keep it a secret.

In public key cryptosystems, first invented in the 1970s, knowing how to encrypt a message does not help one to decrypt the message. Therefore, everyone can have a publicly known encryption key. The only key that needs to be kept secret is the decryption key.

To encrypt a message using RSA using a key (n,e) :

i. Translate the plaintext message M into sequences of two digit integers representing the letters. Use 00 for A, 01 for B, etc.
ii. Concatenate the two digit integers into strings of digits.
iii. Divide this string into equally sized blocks of $2N$ digits where $2N$ is the largest even number 2525...25 with $2N$ digits that does not exceed n.
iv. The plaintext message M is now a sequence of integers $m_1, m_2, \ldots, m_k$.
v. Each block (an integer) is encrypted using the function $C = M^e \bmod n$.

Example: Encrypt the message STOP using the RSA cryptosystem with key(2537,13).

- $2537 = 43 \cdot 59$,
- $p = 43$ and $q = 59$ are primes and $\gcd(e,(p-1)(q-1)) = \gcd(13, 42 \cdot 58) = 1$.

Solution: Translate the letters in STOP to their numerical equivalents 18 19 14 15.

- Divide into blocks of four digits (because $2525 < 2537 < 252525$) to obtain 1819 1415.
- Encrypt each block using the mapping $C = M^{13} \bmod 2537$.
- Since $1819^{13} \bmod 2537 = 2081$ and $1415^{13} \bmod 2537 = 2182$, the encrypted

To decrypt a RSA ciphertext message, the decryption key d, an inverse of e modulo $(p-1)(q-1)$ is needed. The inverse exists since $\gcd(e,(p-1)(q-1)) = \gcd(13, 42 \cdot 58) = 1$.

With the decryption key d, we can decrypt each block with the computation $M = C^d \bmod p \cdot q$. (*see text for full derivation*)

RSA works as a public key system since the only known method of finding d is based on a factorization of n into primes. There is currently no known feasible method for factoring large numbers into primes.

Example: The message 0981 0461 is received. What is the decrypted message if it was encrypted using the RSA cipher from the previous example.

Solution: The message was encrypted with $n = 43 \cdot 59$ and exponent 13. An inverse of 13 modulo $42 \cdot 58 = 2436$ (*exercise 2 in Section 4.4*) is $d = 937$.

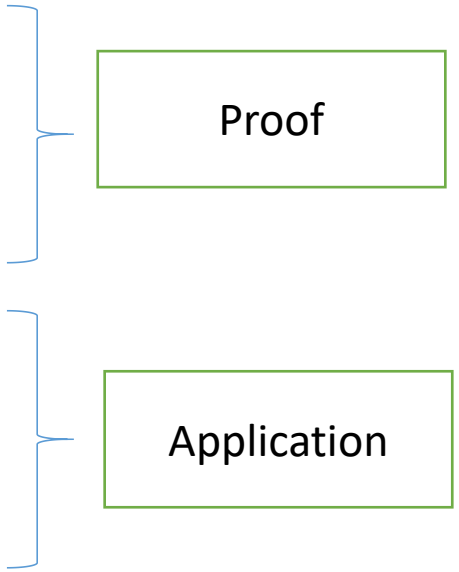
- To decrypt a block C, $M = C^{937} \bmod 2537$.
- Since $0981^{937} \bmod 2537 = 0704$ and $0461^{937} \bmod 2537 = 1115$, the decrypted message is 0704 1115. Translating back to English letters, the message is HELP.

Exercise

- §4.1 5,9(b,d,f,h)
  §4.2 26
  §4.3 16, 24(a,c,f), 30
- §4.4 12 19 22
- §4.5 16 18
- §4.6 6 8 24

Chapter 5： Induction and recursion

- Mathematical Induction
- Strong Induction
- Well-Ordering
- Recursive Definitions
- Structural Induction
- Recursive Algorithms



Mathematical Induction

Principle of Mathematical Induction: To prove that $P(n)$ is true for all positive integers n, we complete these steps:

- *Basis Step*: Show that $P(1)$ is true.
- *Inductive Step*: Show that $P(k) \rightarrow P(k+1)$ is true for all positive integers k.

To complete the inductive step, assuming the *inductive hypothesis* that $P(k)$ holds for an arbitrary integer k, show that must $P(k+1)$ be true.

Mathematical induction can be expressed as the rule of inference

$$(P(1) \wedge \forall k\,(P(k) \rightarrow P(k+1))) \rightarrow \forall n\,P(n),$$

Example: Use mathematical induction to prove that $n^3 - n$ is divisible by 3, for every positive integer n.

Solution: Let $P(n)$ be the proposition that $n^3 - n$ is divisible by 3.

- BASIS STEP: $P(1)$ is true since $1^3 - 1 = 0$, which is divisible by 3.
- INDUCTIVE STEP: Assume $P(k)$ holds, i.e., $k^3 - k$ is divisible by 3, for an arbitrary positive integer k. To show that $P(k+1)$ follows:

$$(k+1)^3 - (k+1) = (k^3 + 3k^2 + 3k + 1) - (k+1)$$
$$= (k^3 - k) + 3(k^2 + k)$$

By the inductive hypothesis, the first term $(k^3 - k)$ is divisible by 3 and the second term is divisible by 3 since it is an integer multiplied by 3. So by part (i) of Theorem 1 in Section 4.1 , $(k+1)^3 - (k+1)$ is divisible by 3.

Therefore, $n^3 - n$ is divisible by 3, for every integer positive integer n.

Template for Proofs by Mathematical Induction

1. Express the statement that is to be proved in the form "for all $n \geq b$, $P(n)$" for a fixed integer b.
2. Write out the words "Basis Step." Then show that $P(b)$ is true, taking care that the correct value of b is used. This completes the first part of the proof.
3. Write out the words "Inductive Step."
4. State, and clearly identify, the inductive hypothesis, in the form "assume that $P(k)$ is true for an arbitrary fixed integer $k \geq b$."
5. State what needs to be proved under the assumption that the inductive hypothesis is true. That is, write out what $P(k+1)$ says.
6. Prove the statement $P(k+1)$ making use the assumption $P(k)$. Be sure that your proof is valid for all integers k with $k \geq b$, taking care that the proof works for small values of k, including $k = b$.
7. Clearly identify the conclusion of the inductive step, such as by saying "this completes the inductive step."
8. After completing the basis step and the inductive step, state the conclusion, namely that by mathematical induction, $P(n)$ is true for all integers n with $n \geq b$.

Strong Induction and

Strong Induction: To prove that $P(n)$ is true for all positive integers n, where $P(n)$ is a propositional function, complete two steps:

- *Basis Step*: Verify that the proposition $P(1)$ is true.
- *Inductive Step*: Show the conditional statement $[P(1) \wedge P(2) \wedge \cdots \wedge P(k)] \rightarrow P(k+1)$ holds for all positive integers k.

We can always use strong induction instead of mathematical induction. But there is no reason to use it if it is simpler to use mathematical induction. (*See page* 335 *of text*.)

In fact, the principles of mathematical induction, strong induction, and the well-ordering property are all equivalent. (*Exercises* 41-43)

Sometimes it is clear how to proceed using one of the three methods, but not the other two.

Example: Show that if n is an integer greater than 1, then n can be written as the product of primes.

Solution: Let $P(n)$ be the proposition that n can be written as a product of primes.

- BASIS STEP: $P(2)$ is true since 2 itself is prime.
- INDUCTIVE STEP: The inductive hypothesis is $P(j)$ is true for all integers j with $2 \leq j \leq k$. To show that $P(k+1)$ must be true under this assumption, two cases need to be considered:
  - If $k+1$ is prime, then $P(k+1)$ is true.
  - Otherwise, $k+1$ is composite and can be written as the product of two positive integers a and b with $2 \leq a \leq b < k+1$. By the inductive hypothesis a and b can be written as the product of primes and therefore $k+1$ can also be written as the product of those primes.

Hence, it has been shown that every integer greater than 1 can be written as the product of primes. ◀

Well-Ordering

Well-ordering property: Every nonempty set of nonnegative integers has a least element.

The well-ordering property is one of the axioms of the positive integers listed in Appendix 1.

The well-ordering property can be used directly in proofs, as the next example illustrates.

The well-ordering property can be generalized.

- **Definition:** A set is *well ordered if every subset has a least element*.
  - **N** is well ordered under ≤.
  - The set of finite strings over an alphabet using lexicographic ordering is well ordered.

Example: Use the well-ordering property to prove the division algorithm, which states that if a is an integer and d is a positive integer, then there are unique integers q and r with $0 \le r < d$, such that $a = dq + r$.

Solution: Let S be the set of nonnegative integers of the form $a - dq$, where q is an integer. The set is nonempty since $-dq$ can be made as large as needed.

- By the well-ordering property, S has a least element $r = a - dq_0$. The integer r is nonnegative. It also must be the case that $r < d$. If it were not, then there would be a smaller nonnegative element in S, namely, $a - d(q_0 + 1) = a - dq_0 - d = r - d > 0$.
- Therefore, there are integers q and r with $0 \le r < d$.

(*uniqueness of q and r is Exercise* 37)

Recursive Definitions and Recusive difinition

Definition: A *recursive* or *inductive definition* of a function consists of two steps.

- BASIS STEP: Specify the value of the function at zero.
- RECURSIVE STEP: Give a rule for finding its value at an integer from its values at smaller integers.

A function $f(n)$ is the same as a sequence a_0, a_1, … , where a_i, where $f(i) = a_i$. This was done using recurrence relations in Section 2.4.

Recursive definitions of sets have two parts:

- The *basis step* specifies an initial collection of elements.
- The *recursive step* gives the rules for forming new elements in the set from those already known to be in the set.

Sometimes the recursive definition has an *exclusion rule*, which specifies that the set contains nothing other than those elements specified in the basis step and generated by applications of the rules in the recursive step.

We will always assume that the exclusion rule holds, even if it is not explicitly mentioned.

We will later develop a form of induction, called *structural induction*, to prove results about recursively defined sets.

Example: Suppose f is defined by:

$f(0) = 3$,

$f(n + 1) = 2f(n) + 3$

Find $f(1), f(2), f(3), f(4)$

Solution:

- $f(1) = 2f(0) + 3 = 2 \cdot 3 + 3 = 9$
- $f(2) = 2f(1) + 3 = 2 \cdot 9 + 3 = 21$
- $f(3) = 2f(2) + 3 = 2 \cdot 21 + 3 = 45$
- $f(4) = 2f(3) + 3 = 2 \cdot 45 + 3 = 93$

Example : Subset of Integers S:

BASIS STEP: $3 \in S$.

RECURSIVE STEP: If $x \in S$ and $y \in S$, then $x + y$ is in S.

- Initially 3 is in S, then $3 + 3 = 6$, then $3 + 6 = 9$, etc.

Binary tree、Rooted tree

Example: Give a recursive definition of the set of balanced parentheses(圆括号) P.

Solution:

BASIS STEP: $() \in P$

RECURSIVE STEP: If $w \in P$, then $()w \in P$, $(w) \in P$ and $w() \in P$.

Show that (() ()) is in P.

Why is))(() not in P?

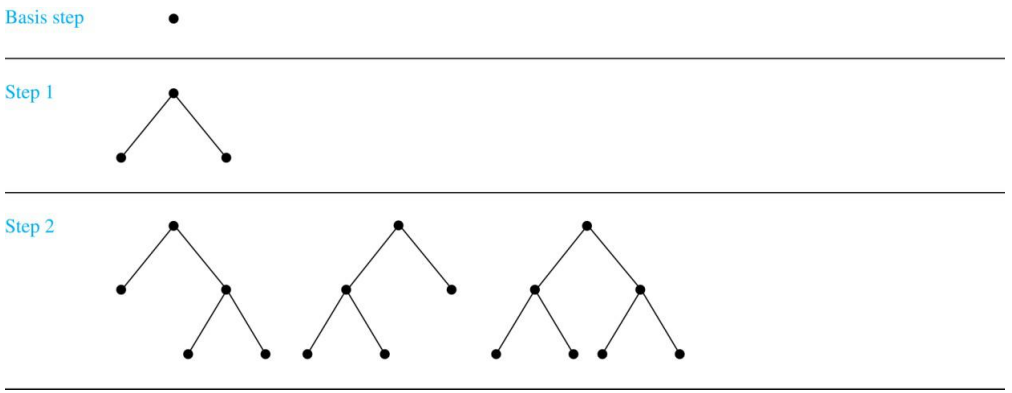
Structural Induction

Definition: To prove a property of the elements of a recursively defined set, we use *structural induction*.

BASIS STEP: Show that the result holds for all elements specified in the basis step of the recursive definition.

RECURSIVE STEP: Show that if the statement is true for each of the elements used to construct new elements in the recursive step of the definition, the result holds for these new elements.

The validity of structural induction can be shown to follow from the principle of mathematical induction.



Definition: The *height h(T)* of a full binary tree T is defined recursively as follows:

- BASIS STEP: The height of a full binary tree T consisting of only a root r is $h(T) = 0$.
- RECURSIVE STEP: If T_1 and T_2 are full binary trees, then the full binary tree $T = T_1 \cdot T_2$ has height $h(T) = 1 + \max(h(T_1), h(T_2))$.

The number of vertices $n(T)$ of a full binary tree T satisfies the following recursive formula:

- **BASIS STEP**: The number of vertices of a full binary tree T consisting of only a root r is $n(T) = 1$.
- **RECURSIVE STEP**: If T_1 and T_2 are full binary trees, then the full binary tree $T = T_1 \cdot T_2$ has the number of vertices $n(T) = 1 + n(T_1) + n(T_2)$.

Exercise

- §5.1 6,10,22
- §5.2 2,8,14,28
- §5.3 4,10,18
- §5.4 2,16,24,28