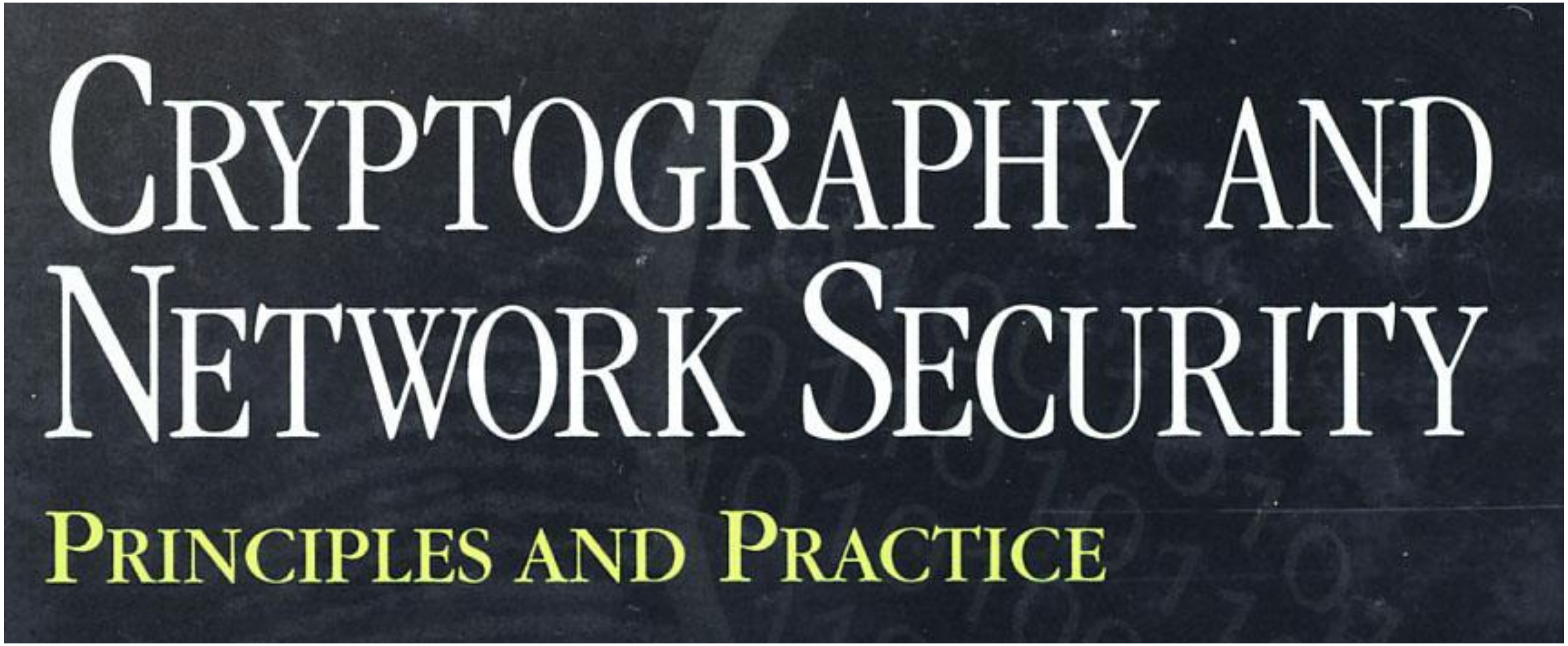
Fields and Polynomials

CS6025 Data Encoding

Yizong Cheng

2-10-15

Stallings’ Book



Error-Detecting Codes

ISBN-13: 978-0-13-609704-4
ISBN-10: 0-13-609704-9

EAN

9 780136 097044

90000 >

International Standard Book Number

- ISBN-13: 978-0-13-609704-4
  - 9 + 7x3 + 8 + 0x3 + 1 + 3x3 + 6 + 0x3 + 9 + 7x3 + 0 + 4x3 + 4
  - 9 + 21 + 8 + 0 + 1 + 9 + 6 + 0 + 9 + 21 + 0 + 12 + 4
  - = 100, a multiple of 10.
- The 13th digit is the “check digit” for error detection.
- Need “checksum” for data to detect error.

Divisibility: b|a if a = mb for some m

Divisibility

We say that a nonzero b **divides** a if $a = mb$ for some m, where a, b, and m are integers. That is, b divides a if there is no remainder on division. The notation $b|a$ is commonly used to mean b divides a. Also, if $b|a$, we say that b is a **divisor** of a.

The positive divisors of 24 are 1, 2, 3, 4, 6, 8, 12, and 24.
$13|182$; $-5|30$; $17|289$; $-3|33$; $17|0$

Quotient and Remainder

The Division Algorithm

Given any positive integer n and any nonnegative integer a, if we divide a by n, we get an integer quotient q and an integer remainder r that obey the following relationship:

$$a = qn + r \qquad 0 \le r < n; q = \lfloor a/n \rfloor \tag{4.1}$$

where $\lfloor x \rfloor$ is the largest integer less than or equal to x. Equation (4.1) is referred to as the division algorithm.[1]

a = qn + r

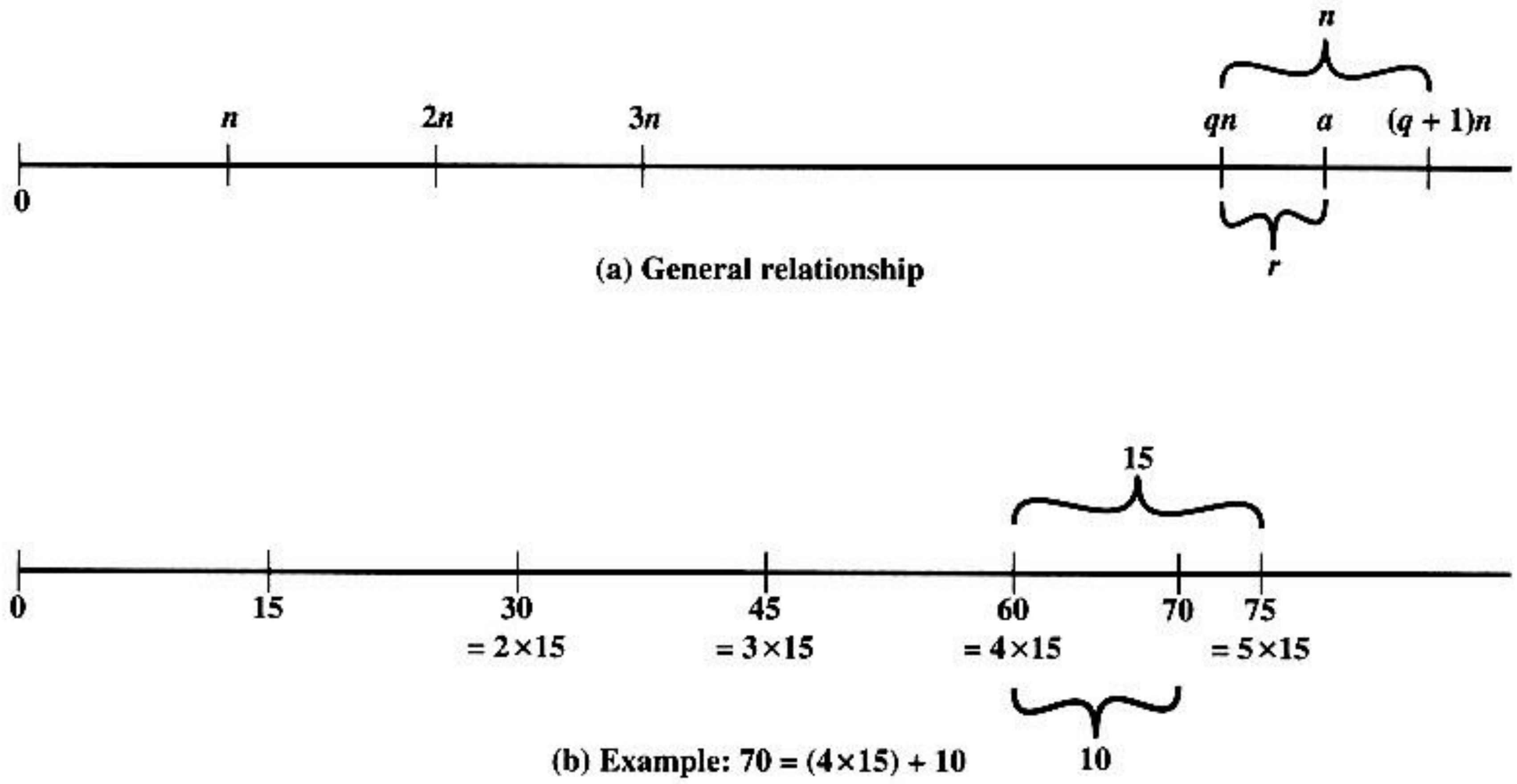


Figure 4.1 The Relationship $a = qn + r; 0 \leq r < n$

a mod n (different from a % n in Java)

The Modulus

If a is an integer and n is a positive integer, we define a mod n to be the remainder when a is divided by n. The integer n is called the **modulus**. Thus, for any integer a, we can rewrite Equation (4.1) as follows:

$$a = qn + r \qquad 0 \le r < n;\ q = \lfloor a/n \rfloor$$

$$a = \lfloor a/n \rfloor \times n + (a \bmod n)$$

$$\boxed{11 \bmod 7 = 4; \qquad -11 \bmod 7 = 3}$$

Congruence

Two integers a and b are said to be **congruent modulo n**, if $(a \bmod n) = (b \bmod n)$. This is written as $a \equiv b \pmod{n}$.[2]

$73 \equiv 4 \pmod{23}$;	$21 \equiv -9 \pmod{10}$

Note that if $a \equiv 0 \pmod{n}$, then $n|a$.

Properties of Congruences

Congruences have the following properties:

1. $a \equiv b \pmod{n}$ if $n|(a - b)$.
2. $a \equiv b \pmod{n}$ implies $b \equiv a \pmod{n}$.
3. $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$ imply $a \equiv c \pmod{n}$.

Modular Arithmetic

Modular arithmetic exhibits the following properties

1. $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
2. $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
3. $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

Table 4.2 Arithmetic Modulo 8

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

(a) Addition modulo 8

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

(b) Multiplication modulo 8

w	$-w$	w^{-1}
0	0	—
1	7	1
2	6	—
3	5	3
4	4	—
5	3	5
6	2	—
7	1	7

(c) Additive and multiplicative inverses modulo 8

Residue Class Z_n

Define the set Z_n as the set of nonnegative integers less than n:

$$\mathrm{Z}_n = \{0, 1, \ldots, (n - 1)\}$$

This is referred to as the **set of residues**, or **residue classes** (mod n). To be more precise, each integer in Z_n represents a residue class. We can label the residue classes (mod n) as $[0], [1], [2], \ldots, [n - 1]$, where

$$[r] = \{a\text{: } a \text{ is an integer, } a \equiv r \pmod{n}\}$$

Laws of Modular Arithmetic

Table 4.3 Properties of Modular Arithmetic for Integers in Z_n

Property	Expression
Commutative Laws	$(w + x) \bmod n = (x + w) \bmod n$ $(w \times x) \bmod n = (x + w) \bmod n$
Associative Laws	$[(w + x) + y] \bmod n = [w + (x + y)] \bmod n$ $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$
Distributive Law	$[w \times (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$
Identities	$(0 + w) \bmod n = w \bmod n$ $(1 \times w) \bmod n = w \bmod n$
Additive Inverse ($-w$)	For each $w \in Z_n$, there exists a a z such that $w + z \equiv 0 \bmod n$

Group {G, •}

Groups

A **group** G, sometimes denoted by $\{G, \bullet\}$, is a set of elements with a binary operation denoted by $\bullet$ that associates to each ordered pair (a, b) of elements in G an element $(a \bullet b)$ in G, such that the following axioms are obeyed:[4]

(A1) Closure:	If a and b belong to G, then $a \bullet b$ is also in G.
(A2) Associative:	$a \bullet (b \bullet c) = (a \bullet b) \bullet c$ for all a, b, c in G.
(A3) Identity element:	There is an element e in G such that $a \bullet \mathrm{e} = \mathrm{e} \bullet a = a$ for all a in G.
(A4) Inverse element:	For each a in G, there is an element a' in G such that $a \bullet a' = a' \bullet a = \mathrm{e}$.

Abelian Group

If a group has a finite number of elements, it is referred to as a **finite group**, and the **order** of the group is equal to the number of elements in the group. Otherwise, the group is an **infinite group**.

A group is said to be **abelian** if it satisfies the following additional condition:

(A5) Commutative: $a \cdot b = b \cdot a$ for all a, b in G.

Cyclic Group and Generator

CYCLIC GROUP We define exponentiation within a group as a repeated application of the group operator, so that $a^3 = a \bullet a \bullet a$. Furthermore, we define $a^0 = e$ as the identity element, and $a^{-n} = (a')^n$, where a' is the inverse element of a within the group. A group G is **cyclic** if every element of G is a power a^k (k is an integer) of a fixed element $a \in G$. The element a is said to **generate** the group G or to be a **generator** of G. A cyclic group is always abelian and may be finite or infinite.

Ring {R, +, x}

Rings

A **ring** R, sometimes denoted by $\{R, +, \times\}$, is a set of elements with two binary operations, called *addition* and *multiplication*,[6] such that for all a, b, c in R the following axioms are obeyed.

(A1–A5) R is an abelian group with respect to addition; that is, R satisfies axioms A1 through A5. For the case of an additive group, we denote the identity element as 0 and the inverse of a as $-a$.

(M1) Closure under multiplication: If a and b belong to R, then ab is also in R.

(M2) Associativity of multiplication: $a(bc) = (ab)c$ for all a, b, c in R.

(M3) Distributive laws: $a(b + c) = ab + ac$ for all a, b, c in R.
$(a + b)c = ac + bc$ for all a, b, c in R.

Integral Domain

A ring is said to be **commutative** if it satisfies the following additional condition:

(M4) Commutativity of multiplication: $ab = ba$ for all a, b in R.

Next, we define an **integral domain**, which is a commutative ring that obeys the following axioms.

(M5) Multiplicative identity: There is an element 1 in R such that $a1 = 1a = a$ for all a in R.

(M6) No zero divisors: If a, b in R and $ab = 0$, then either $a = 0$ or $b = 0$.

Field {F, +, x}

Fields

A **field** F, sometimes denoted by $\{F, +, \times\}$, is a set of elements with two binary operations, called *addition* and *multiplication*, such that for all a, b, c in F the following axioms are obeyed.

(A1–M6) F is an integral domain; that is, F satisfies axioms A1 through A5 and M1 through M6.

(M7) Multiplicative inverse: For each a in F, except 0, there is an element a^{-1} in F such that $aa^{-1} = (a^{-1})a = 1$.

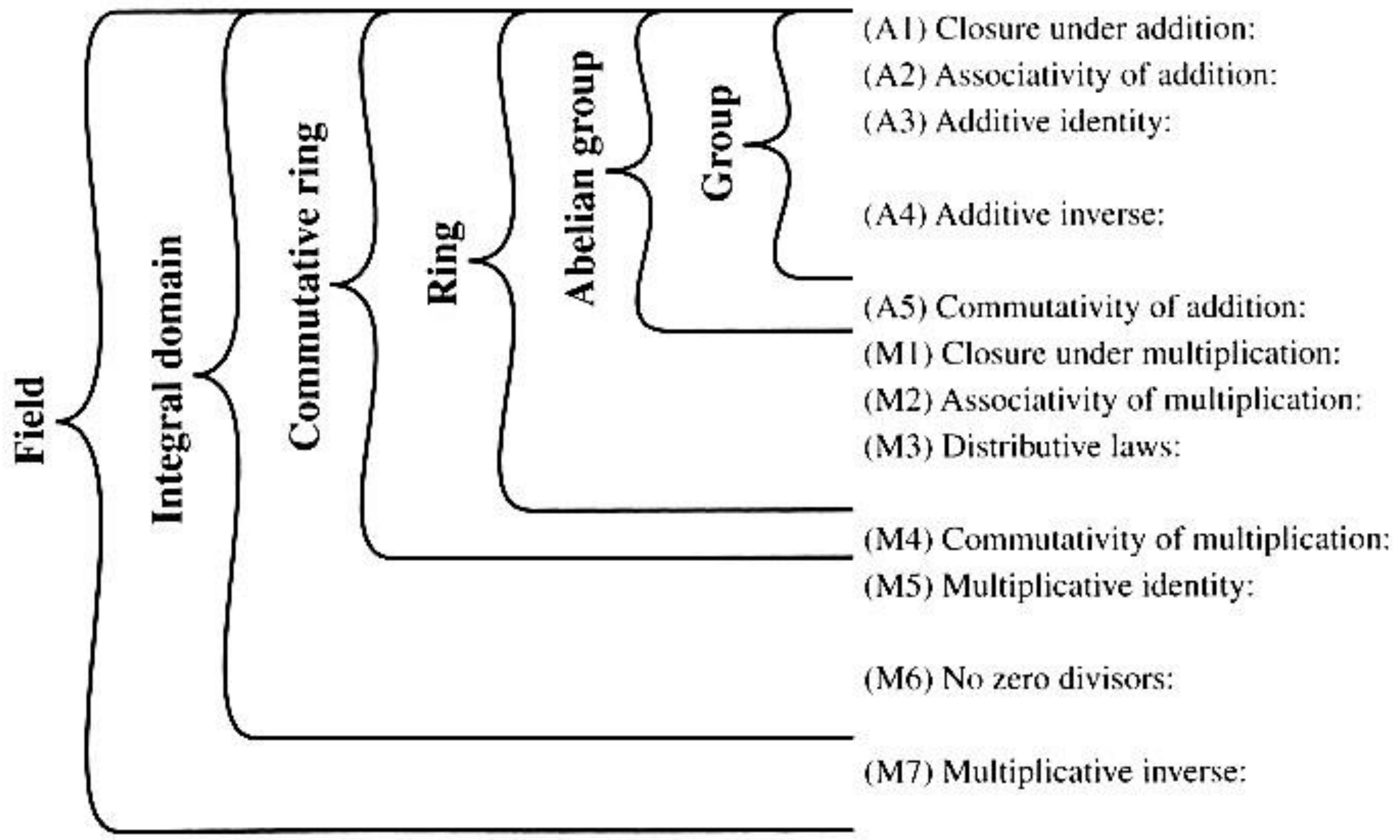


Figure 4.2 Groups, Ring, and Field

Galois Fields

The finite field of order p^n is generally written GF(p^n); GF stands for Galois field, in honor of the mathematician who first studied finite fields. Two special cases are of interest for our purposes. For $n = 1$, we have the finite field GF(p); this finite field has a different structure than that for finite fields with $n > 1$ and is studied in this section. In Section 4.7, we look at finite fields of the form GF(2^n).

{0,1} as a Field, Z_2 or GF(2)

The simplest finite field is GF(2). Its arithmetic operations are easily summarized:

+	0	1
0	0	1
1	1	0

Addition

×	0	1
0	0	0
1	0	1

Multiplication

w	$-w$	w^{-1}
0	0	—
1	1	1

Inverses

In this case, addition is equivalent to the exclusive-OR (XOR) operation, and multiplication is equivalent to the logical AND operation.

Table 4.5 Arithmetic in GF(7)

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

(a) Addition modulo 7

×	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

(b) Multiplication modulo 7

w	$-w$	w^{-1}
0	0	—
1	6	1
2	5	4
3	4	5
4	3	2
5	2	3
6	1	6

(c) Additive and multiplicative inverses modulo 7

Table 4.6 Arithmetic in GF(2^3)

		000	001	010	011	100	101	110	111
	+	0	1	2	3	4	5	6	7
000	0	0	1	2	3	4	5	6	7
001	1	1	0	3	2	5	4	7	6
010	2	2	3	0	1	6	7	4	5
011	3	3	2	1	0	7	6	5	4
100	4	4	5	6	7	0	1	2	3
101	5	5	4	7	6	1	0	3	2
110	6	6	7	4	5	2	3	0	1
111	7	7	6	5	4	3	2	1	0

(a) Addition

		000	001	010	011	100	101	110	111
	×	0	1	2	3	4	5	6	7
000	0	0	0	0	0	0	0	0	0
001	1	0	1	2	3	4	5	6	7
010	2	0	2	4	6	3	1	7	5
011	3	0	3	6	5	7	4	1	2
100	4	0	4	3	7	6	2	5	1
101	5	0	5	1	4	2	7	3	6
110	6	0	6	7	1	5	3	2	4
111	7	0	7	5	2	1	6	4	3

(b) Multiplication

w	$-w$	w^{-1}
0	0	—
1	1	1
2	2	5
3	3	6
4	4	7
5	5	2
6	6	3
7	7	4

(c) Additive and multiplicative inverses

Polynomial over a Field

A **polynomial** of degree n (integer $n \geq 0$) is an expression of the form

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = \sum_{i=0}^{n} a_i x^i$$

where the a_i are elements of some designated set of numbers S, called the **coefficient set**, and $a_n \neq 0$. We say that such polynomials are defined over the coefficient set S.

A zero-degree polynomial is called a **constant polynomial** and is simply an element of the set of coefficients. An nth-degree polynomial is said to be a **monic polynomial** if $a_n = 1$.

In the context of abstract algebra, we are usually not interested in evaluating a polynomial for a particular value of x [e.g., $f(7)$]. To emphasize this point, the variable x is sometimes referred to as the **indeterminate**.

Polynomial Addition and Multiplication

Addition and subtraction are performed by adding or subtracting corresponding coefficients. Thus, if

$$f(x) = \sum_{i=0}^{n} a_i x^i; \qquad g(x) = \sum_{i=0}^{m} b_i x^i; \qquad n \geq m$$

then addition is defined as

$$f(x) + g(x) = \sum_{i=0}^{m} (a_i + b_i) x^i + \sum_{i=m+1}^{n} a_i x^i$$

and multiplication is defined as

$$f(x) \times g(x) = \sum_{i=0}^{n+m} c_i x^i$$

where

$$c_k = a_0 b_k + a_1 b_{k-1} + \cdots + a_{k-1} b_1 + a_k b_0$$

Polynomial Ring

Let us now consider polynomials in which the coefficients are elements of some field F; we refer to this as a polynomial over the field F. In that case, it is easy to show that the set of such polynomials is a ring, referred to as a **polynomial ring**. That is, if we consider each distinct polynomial to be an element of the set, then that set is a ring.[8]

Polynomial Division

However, as we demonstrate presently, even if the coefficient set is a field, polynomial division is not necessarily exact. In general, division will produce a quotient and a remainder. We can restate the division algorithm of Equation (4.1) for polynomials over a field as follows. Given polynomials $f(x)$ of degree n and $g(x)$ of degree (m), $(n \geq m)$, if we divide $f(x)$ by $g(x)$, we get a quotient $q(x)$ and a remainder $r(x)$ that obey the relationship

$$f(x) = q(x)g(x) + r(x) \tag{4.10}$$

with polynomial degrees:

Degree $f(x) = n$
Degree $g(x) = m$
Degree $q(x) = n - m$
Degree $r(x) \leq m - 1$

Data as Polynomials over GF(2)

- Data is viewed as bit strings.
- Each bit is a coefficient of a polynomial over Z_2.
  - 0 is the polynomial 0, 1 is the polynomial 1
  - 10 is the polynomial x, 11 is $x + 1$
  - 100 is x^2, 101 is $x^2 + 1$, 110 is $x^2 + x$,
  - 111 is $x^2 + x + 1$, 1000 is x^3, …

Cyclic Redundancy Check (CRC)

- Data represented as polynomials over Z_2.
- 101001 → $x^5+x^3+1 = M(x)$
- A checksum of fixed length n is appended.
- $C(x) = x^nM(x) + R(x)$, where $R(x)$ is a polynomial of degree $< n$.
- A generator polynomial $G(x)$ of degree n is used to compute $R(x)$ so that $G(x)$ divides $C(x) = x^nM(x) + R(x)$.

Cyclic Codes

- $M(x)x^n$ represents the bit string left shifted by n positions.
- Let $R(x)$ be the remainder of $M(x)x^n$ divided by $G(x)$.
- The codeword for $M(x)$ is
- $C(x) = M(x)x^n + R(x)$.
- No remainder for $C(x)$ divided by $G(x)$
- as error checking.

Error Polynomials

- Error can be represented as a polynomial $E(x)$ added to $C(x)$.
- Error escapes detection only when $G(x)$ divides $E(x)$.
- If $G(x)$ has two non-zero coefficients, then it won't divide $E(x)=x^j$, or all single-bit errors get detected.

Some CRC Uses

- The header error check (HEC) field in ATM cell header uses CRC-8, or
- $G(x) = x^8+x^2+x+1$, or 100000111.
- Ethernet has CRC-32, 100000100110000010001110110110111.
- USB data packets use CRC-CCITT, $G(x)=x^{16}+x^{15}+x^2+1$.
- USB handshaking packets are single bytes with complementary halves as error detection.
- USB token packets have two more bytes that contains 11 bits for address and 5 bits for CRC checksum with $G(x) = x^5+x^2+1$.

Properties of Cyclic Codes

- If M1(x) and M2(x) are information polynomials with R1(x) and R2(x) as the remainders divided by G(x), then R1(x)+R2(x) is the remainder of M1(x)+M2(x) divided by G(x).
- If M1(x)x^n+R1(x) and M2(x)x^n+R2(x) are codewords, then so is their sum.
- Cyclic codes are linear codes.

Properties of Cyclic Codes

- All codewords are divisible by G(x).
- In particular, G(x) is the non-zero codeword with minimum degree .
- Shifts of G(x), or $x^i G(x)$, i=0,…, k-1, are codewords and form a basis for the space of all codewords.

Computing the Checksum

- Append zeros to the message: $x^n M(x)$
- Divide this by $G(x)$ and the remainder is used as $R(x)$.
- Proof: $x^n M(x) = Q(x)G(x) + R(x)$
- Thus, $C(x) = x^n M(x) + R(x) = Q(x)G(x)$.
- Codewords are multiples of $G(x)$.
- $G(x)$ is hence called the generator.

CRC Encoding

- m=01101, g=1011,
- c=01101001
- 01101000
- _1011__
- 1100
- __1011_
- 1110
- __1011
- 1010
- 1011
- 001

Reflected CRC Encoding

- m=01101, g=1011,
- c=01101 001
- 01101000
- _1011__
- 1100
- __1011_
- 1110
- ___1011
- 1010
- _____1011
- 001

- m = 10110, g = 1101
- 110 shift(0)
- 011 shift(1)
- 101 XOR
- 110
- 011 shift(1)
- 001 XOR
- 110
- 111 shift(1)
- 011 XOR
- 110
- 101 shift(1)
- 010 XOR
- 110
- 100
- c = 100 10110

Error Detection of CRC

- Received $C'(x) = C(x) + E(x)$ where $E(x)$ is the error polynomial.
- Divide $C'(x)$ by $G(x)$ and the remainder is called the syndrome.
- If syndrome is not 0, an error is detected.
- An error is undetected if and only if $E(x)$ is a multiple of $G(x)$.

Single-Bit Error Detection

- A single-bit error can be represented by $E(x) = x^k$.
- A bit flipped at position k.
- It will be detected if $G(x)$ contains more than one non-zero coefficient.
- Only x^j with $j <= k$ will divide x^k.

Double-Bit Error Detection

- A double-bit error is $E(x) = x^k + x^j$.
- $E(x) = x^j(x^{k-j} + 1)$.
- To have this error undetected, $G(x)$ has to divide $x^{k-j} + 1$.
- Make sure this does not happen for k-j up to the data frame size.
- Then all double-bit errors are detected.

Odd Number of Bit Errors

- If x+1 is a factor of G(x), then any error involving an odd number of bits will be detected.
- To have E(x) be a multiple of G(x), it will be a multiple of x+1, or a sum of shifted double bits.
- The number of bits in any multiple of x+1 is even.

Burst Errors

- If $G(x)$ is of degree n, then all its non-zero multiples will have degrees larger than or equal to n.
- Any burst error of up to n bits will be detected when the degree of $E(x)$ is smaller than n.

Standard CRC Generators

- ATM: x^8+x^2+x+1
- CRC-16: $x^{16}+x^{15}+x^2+1$
- CRC-CCITT: $x^{16}+x^{12}+x^5+1$
- CD-ROM: $(x^{16}+x^{15}+x^2+1)(x^{16}+x^2+x+1)$
- CRC-32: $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

Example: CRC-16

- $x^{16} + x^{15} + x^2 + 1 = (x + 1)(x^{15} + x + 1)$
- $x^{15} + x + 1$ is primitive, or the smallest N such that this polynomial divides $x^N + 1$ is $N = 2^{15} – 1$.
- CRC-16 detects all 1-bit errors, all double-bit errors when they are within 32767 bits of each other, and all odd number of bit errors.

Implementation

- Precompute CRC for all 256 8-bits messages and store them in a table.
- Proceed one byte a time to complete.
- Example: RFC 1952, GZIP File Format
- Hardware Implementation with shift registers.

Serial Communication

- Burst errors happens to consecutive bits in transmission of bits in serial communication.
- Data polynomials should represent the serial bit stream.
- Ethernet, USB, serial port sends the least significant bit of a byte first.
- Bytes are reflected and then concatenated.
- Hardware implementation uses this reflection.
- So are many software implementations.
  - java.util.zip.CRC32

Precomputing long[] crc_table

```
void makeTable(){
 for (int n = 0; n < 256; n++) {
     long c = (long) n;
     for (int k = 0; k < 8; k++) {
           if ((c & 1) != 0) {
              c = 0xedb88320L ^ (c >> 1);
           } else {
              c = c >> 1;
           }
     }
     crc_table[n] = c;
 }
}
```

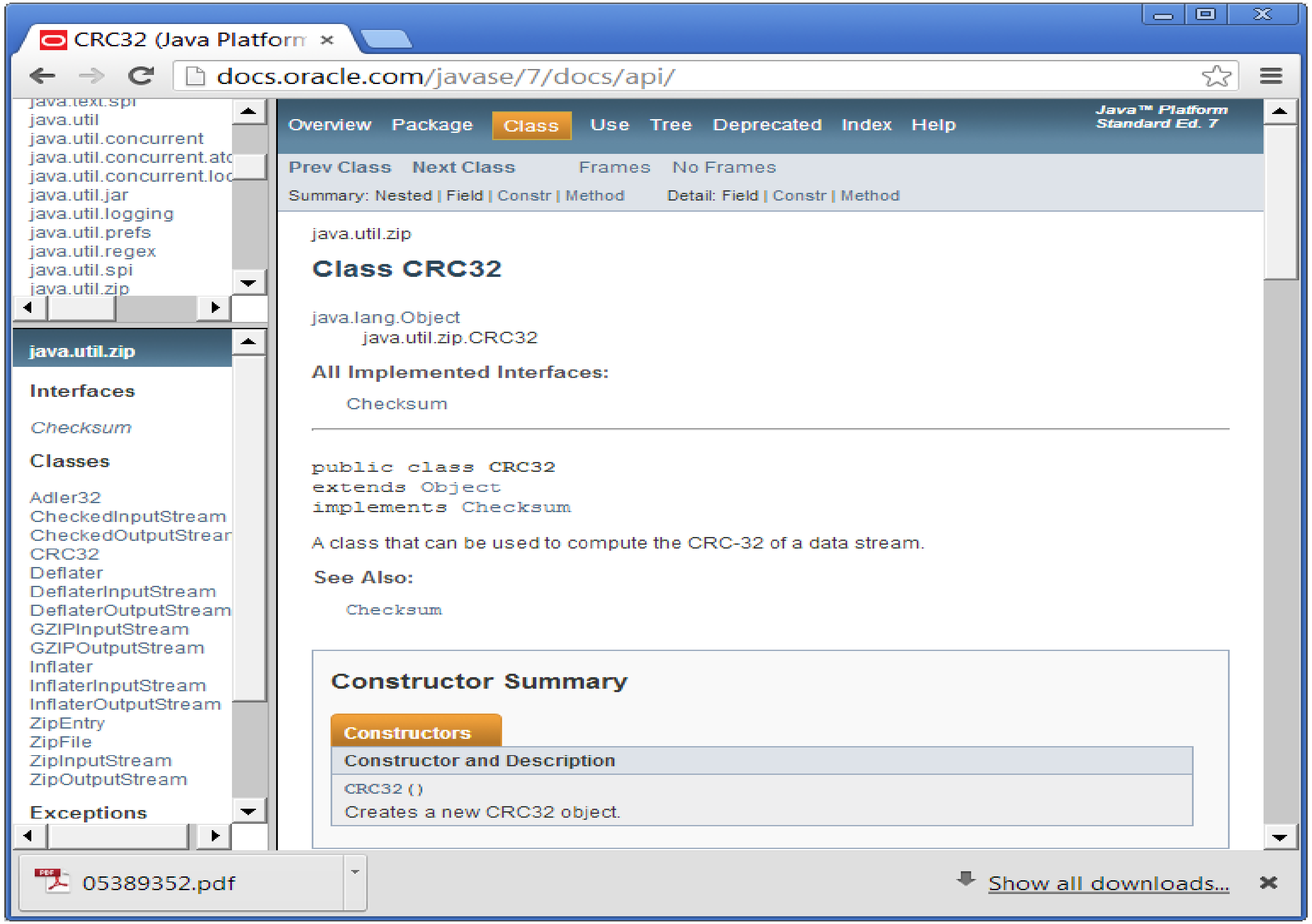
Reflected G(x) for CRC32

- CRC-32: $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10} +x^8+x^7+x^5+x^4+x^2+x+1$
- 1 0000 0100 1100 0001 0001 1101 1011 0111.
- Without the x^{32} term, it is 0x04c11db7
- Reflected into
- 1110 1101 1011 1000 1000 0011 0010 0000 1
- Without the x^{32} term, it is 0xedb88320L.

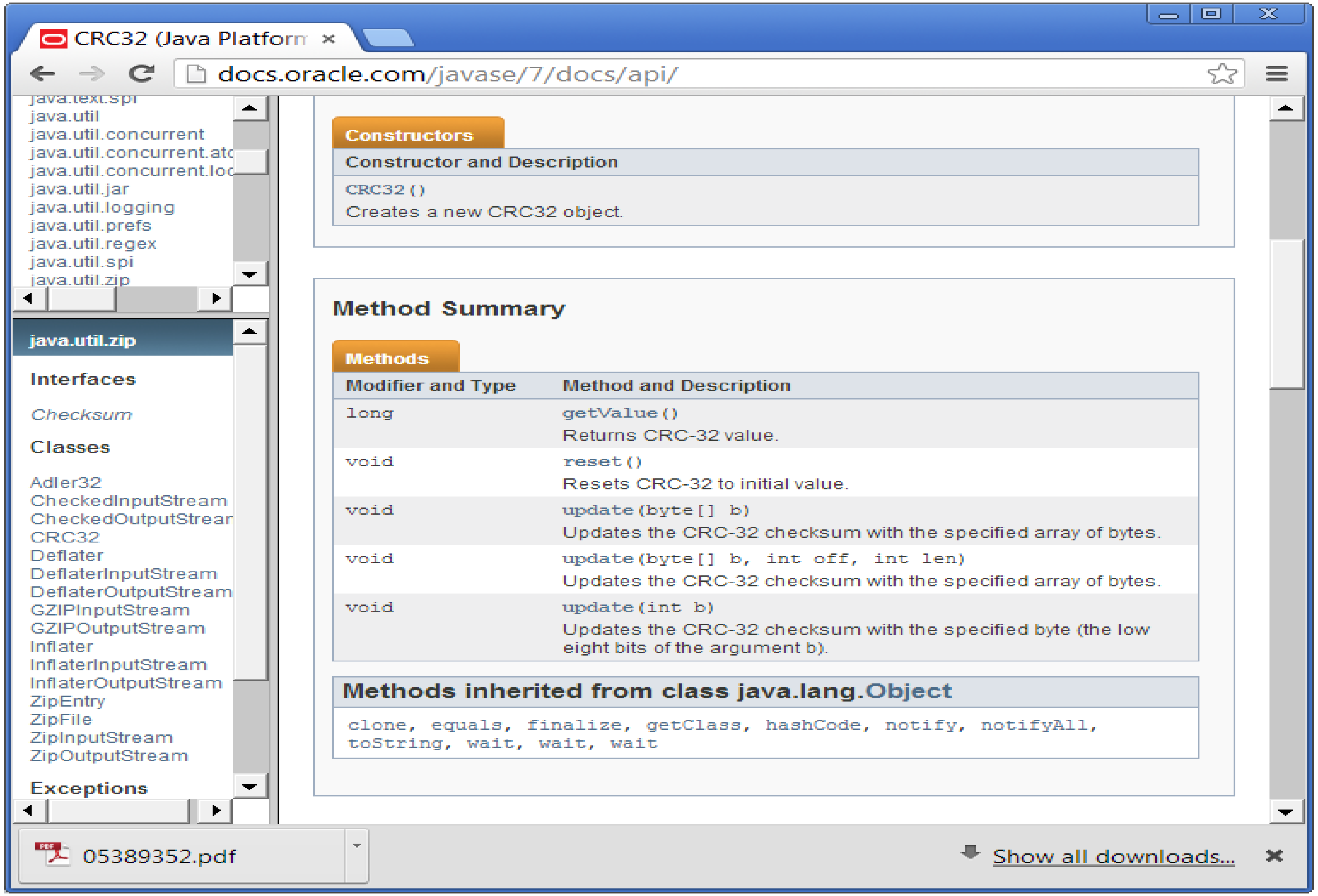
CRC for Data in byte[] buf

```
long checksum(byte[] buf){
   long c = 0xffffffffL;  // initial value
   int len = buf.length;
   for (int n = 0; n < len; n++)
    c = crc_table[(int)(c ^ buf[n]) & 0xff] ^ (c >> 8);
    return c ^ 0xffffffffL;  // final mask value
 }
```

java.util.zip.CRC32



CRC32.getValue(), CRC32.update()



Homework 9: due 2-16-15

- Complete the test() function in H9.java to print out the CRC32 checksum of a random data sequence of 256 bytes, using first the java.util.zip.CRC32 class and then using the given Java implementation checksum() that uses a pre-computed table.