Discrete Mathematical Models

Lecture 10

Kane Townsend

Semester 2, 2024

Section B: Digital Information

Representing numbers (cont.)

Rational numbers

What is a rational number?

Recall that $\mathbb{Q}$ is the set of **rational numbers**. A rational number is a number that can be represented as the ratio of two integers.

EXAMPLE $\frac{2}{3}$ is a rational number.

Please note that every integer is a rational number as, for example $6 = \frac{6}{1}$.

What is a rational number?

Recall that $\mathbb{Q}$ is the set of **rational numbers**. A rational number is a number that can be represented as the ratio of two integers.

EXAMPLE $\frac{2}{3}$ is a rational number.

Please note that every integer is a rational number as, for example $6 = \frac{6}{1}$.

Are you happy with this definition?

What is a rational number?

Let $Q = \mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$.

What does an element of Q look like?

What is a rational number?

Let $Q = \mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$.

What does an element of Q look like?

The set Q may be partitioned so that any elements (n_1, d_1) and (n_2, d_2) of Q are in the same partition set if and only if $n_1 d_2 = n_2 d_1$.

So, for example, $\{(2,3), (-2,-3), (4,6), (-4,-6), (6,9), (-6,-9), \ldots\}$ is one of the sets in the partition

The sets in the partition may themselves be considered rational numbers. We usually write $\frac{2}{3}$ instead of $\{(2,3), (-2,-3), (4,6), (-4,-6), (6,9), (-6,-9), \ldots\}$.

Representing rationals in a computer

Our motivation will to represent a **non-zero** rational number q will be **scientific notation**. For any base b this is

$$\boxed{q = (-1)^s \times m \times b^n}$$

where

- $q \in \mathbb{Q}$, $q \neq 0$;
- $b \in \mathbb{Z}$, $b \geq 2$;
- $s \in \{0, 1\}$, (s is the **sign bit**)
- $m \in \mathbb{Q}$, $1 \leq m < b$, (m is the '**mantissa**') and
- $n \in \mathbb{Z}$ (n is the **exponent**).

Given q and b, the values of of s, m and n are uniquely determined by these conditions.

Example in base 10: $23.5 = (-1)^0 \times 2.35 \times 10^1$.

Example in base 10

$$23.5 = (-1)^0 \times 2.35 \times 10^1.$$

Example in base 10

$$23.5 = (-1)^0 \times 2.35 \times 10^1.$$

Some more examples:

- $-154 = (-1)^1 \times 1.54 \times 10^2.$
- $0.031 = (-1)^0 \times 3.1 \times 10^{-2}.$
- $\frac{2}{3} = (-1)^0 \times 6.\dot{6} \times 10^{-1}$

The number of digits of m is called the **precision**.

When storing a rational number efficiently in a computer we will not use decimal digits and we will have limitations on accuracy.

IEEE half-precision

The shortest IEEE standard for representing rational numbers is called *half-precision floating point*. It uses a 16-bit word partitioned into three parts.

To store a rational number x it is first represented as

$$(-1)^s \times m \times 2^n$$

- The sign bit s is stored as the left-most bit (bit 15).
- The mantissa $1 \leq m < 2$ ("significand" in IEEE parlance) is stored in the right-most 10 bits (bits 9 to 0). Only the fractional part of m is stored since m always has the form $1.\star\star\star\ldots$ in binary digits. Hence, we do not need need to store the 1.
- The exponent n is stored in the 5 bits in between (bits 14 to 10). It is stored with an "offset". In order to allow for both positive and negative exponents, but to avoid another sign bit, the value stored is $n + 15$.

IEEE half-precision (asides)

- In principle the five exponent bits can store exponents in the range $-15 \leq n \leq 16$, but 00000 and 11111 are reserved for special purposes, so in fact n is restricted to the range $-14 \leq n \leq 15$.
- With limits on precision and exponent size, some rational numbers can only be stored inaccurately, if at all. This was also true for integers, but with integers we can represent ALL of the integers close enough to 0, so it is easier to understand which integers we can and cannot represent.
- If you have a reason to need to represent rational numbers with an accuracy beyond the accuracy provided by some sort of standard set up, you can write dedicated software to represent numbers with greater precision. Requiring greater precision can lead to inefficiency.

An example

A rational number x is stored in half-precision floating point as the word 5555_{16}. Which number is being represented?

An example

A rational number x is stored in half-precision floating point as the word 5555_{16}. Which number is being represented?

$$5555_{16} = \underbrace{0101}_{5}\underbrace{0101}_{5}\underbrace{0101}_{5}\underbrace{0101}_{5} = \begin{array}{c|c|l} 0 & 10101 & 0101010101 \\ s & n+15 & \text{frac.part of } m \\ & = 21 & \end{array}$$

$$\longrightarrow (-1)^0 \times 1.0101010101 \times 2^{21-15}$$

$$\begin{aligned} &= 1.0101010101 \times 2^6 \\ &= 1010101.0101 \qquad \text{(moving the binary point 6 places right)} \\ &= 64+16+4+1+\frac{1}{4}+\frac{1}{16} = \boxed{85\tfrac{5}{16} = 85.3125_{10}}. \end{aligned}$$

Density

The largest number represented with half-precision floating point is:

$$0\ 11110\ 1111111111 = \begin{array}{r|c|l} 0 & 11110 & 1111111111 \\ s & n{+}15 & \text{frac.part of } m \\ & = 30 & \end{array}$$

$$\longrightarrow \ (-1)^0 \times 1.1111111111 \times 2^{30-15}$$

$$\begin{aligned} &= 1.1111111111 \times 2^{15} \\ &= 65504 \end{aligned}$$

Density (Cont.)

The 2nd largest number represented with half-precision floating point is:

$$0\ 11110\ 1111111110 = \begin{array}{c|c|l} 0 & 11110 & 1111111110 \\ s & n{+}15 & \text{frac.part of } m \\ & = 30 & \end{array}$$

$$\longrightarrow\ (-1)^0 \times 1.1111111110 \times 2^{30-15}$$

$$\begin{aligned} &= 1.1111111110 \times 2^{15} \\ &= 65472 \end{aligned}$$

You will notice that the difference is $2^5 = 32$ between the largest and second largest number you can store.

Density (Cont.)

The smallest and second smallest positive number:

$$0\ 00001\ 0000000000 = \begin{array}{r|c|l} 0 & 00001 & 0000000000 \\ s & n{+}15 & \text{frac.part of } m \\ & =1 & \end{array}$$

$$\longrightarrow\ (-1)^0 \times 1.0000000000 \times 2^{1-15}$$
$$= 1 \times 2^{-14}$$
$$= 0.00006103515625$$

Density (Cont.)

The smallest and second smallest positive number:

$$0\ 00001\ 0000000000 = \begin{array}{c|c|l} 0 & 00001 & 0000000000 \\ s & n{+}15 & \text{frac.part of } m \\ & =1 & \end{array}$$

$$\longrightarrow\ (-1)^0 \times 1.0000000000 \times 2^{1-15}$$

$$= \ 1 \times 2^{-14}$$

$$= \ 0.00006103515625$$

$$0\ 00001\ 0000000001 = \begin{array}{c|c|l} 0 & 00001 & 0000000001 \\ s & n{+}15 & \text{frac.part of } m \\ & =1 & \end{array}$$

$$\longrightarrow\ (-1)^0 \times 1.0000000001 \times 2^{1-15}$$

$$= \ (1 + \frac{1}{1024}) \times 2^{-14} =$$

$$= \ 0.000061094760894775390625$$

Difference is $2^{-24} = 1/16777216$

Modular arithmetic

A Theorem

$$\forall z \in \mathbb{Z}\ \forall d \in \mathbb{N}\ \exists! q \in \mathbb{Z}\ \exists! r \in \mathbb{Z}\ \ (z = qd + r) \wedge (0 \leq r < d)$$

A Theorem

$$\forall z \in \mathbb{Z}\, \forall d \in \mathbb{N}\, \exists! q \in \mathbb{Z}\, \exists! r \in \mathbb{Z}\ \ (z = qd + r) \wedge (0 \leq r < d)$$

Theorem: (The Quotient-Remainder Theorem). Given any integer z and any positive integer d, there is exactly one way to express z in the form $z = qd + r$, where q is an integer and $r \in \{0, 1, \ldots, d - 1\}$.

In the expression $z = qd + r$, q is called the **quotient** (when z is divided by d) and r is called the **remainder** (when z is divided by d).

Picturing q and r

Fix a choice of $z \in \mathbb{Z}$ and $d \in \mathbb{N}$.

Now picture a number line with the integers marked in some dramatic way.

Now: qd is the integer multiple of d that is closest to z but NOT to the right of z; and r is the distance between qd and z.

A picture will help...

‘mod’ and ‘div’

We define: $q = z \ \textbf{div} \ d; \quad r = z \ \textbf{mod} \ d.$

You may like to say that:

- $z \quad \text{div } d$ gives the **quotient** when z is divided by d;
- $z \quad \text{mod } d$ gives the **remainder** when z is divided by d.

Examples

Q: Evaluate the following expressions:

87 mod 13

-100 div 13

Examples

Q: Evaluate the following expressions:

87 mod 13

-100 div 13

A:
Since $87 = 6(13) + 9$, 87 mod 13 = 9.

Since $-100 = (-8)(13) + 4$, -100 div 13 = -8.

The division algorithm

The ‘primary school’ method of finding quotient and remainder is to use *repeated subtraction*. This only works for non-negative z.

Input: $z \in \mathbb{Z}_{\geq 0}$ and $d \in \mathbb{N}$.
Output: $q = z \text{ div } d$ and $r = z \bmod d$.

Method:

```
Set $r = z$, $q = 0$.
  Loop:   If $r < d$ stop.
          Replace $r$ by $r - d$.
          Replace $q$ by $q + 1$.
  Repeat loop
```

The division algorithm

The ‘primary school’ method of finding quotient and remainder is to use *repeated subtraction*. This only works for non-negative z.

Input: $z \in \mathbb{Z}_{\geq 0}$ and $d \in \mathbb{N}$.
Output: $q = z \text{ div } d$ and $r = z \bmod d$.

Method:

```
Set $r = z$, $q = 0$.
  Loop:    If $r < d$ stop.
           Replace $r$ by $r - d$.
           Replace $q$ by $q + 1$.
  Repeat loop
```

Q: Some small modifications to the algorithm allow it cope also with negative z. Could you make them?

Congruence modulo d

Let $d \in \mathbb{N}$. The **congruence modulo** d relation $R_d \subseteq \mathbb{Z} \times \mathbb{Z}$ is defined by

$$aR_d b \Leftrightarrow (\exists k \in \mathbb{Z} \;\; a = b + kd).$$

We have unusual notation for this relation. We write

$$a \equiv b \pmod{d}$$

to mean $aR_d b$

Congruence modulo d

Let $d \in \mathbb{N}$. The **congruence modulo** d relation $R_d \subseteq \mathbb{Z} \times \mathbb{Z}$ is defined by

$$aR_d b \Leftrightarrow (\exists k \in \mathbb{Z} \;\; a = b + kd).$$

We have unusual notation for this relation. We write

$$a \equiv b \pmod{d}$$

to mean $aR_d b$

Understanding the relation:

- Two integers are congruent modulo d IFF their difference is a multiple of d.
- Two integers are congruent modulo d IFF they leave the same remainder upon division by d.

Proof

Let $a, b \in \mathbb{Z}$ and let $d \in \mathbb{N}$.

Proof

Let $a, b \in \mathbb{Z}$ and let $d \in \mathbb{N}$. Suppose that $a \equiv b \pmod{d}$. Then there exists $k \in \mathbb{Z}$ such that $a = b + kd$. By the Quotient-Remainder Theorem, there exist unique integers q_1, r_1 such that $b = q_1 d + r_1$ and $0 \leq r_1 < d$. Then we have

$$a = b + kd = (q_1 d + r_1) + kd = (q_1 + k)d + r_1.$$

The uniqueness part of the Quotient-Remainder Theorem gives that r_1 is the remainder when a is divided by d. Thus a and b leave the same remainder upon division by d.

Proof

Let $a, b \in \mathbb{Z}$ and let $d \in \mathbb{N}$. Suppose that $a \equiv b \pmod{d}$. Then there exists $k \in \mathbb{Z}$ such that $a = b + kd$. By the Quotient-Remainder Theorem, there exist unique integers q_1, r_1 such that $b = q_1 d + r_1$ and $0 \leq r_1 < d$. Then we have

$$a = b + kd = (q_1 d + r_1) + kd = (q_1 + k)d + r_1.$$

The uniqueness part of the Quotient-Remainder Theorem gives that r_1 is the remainder when a is divided by d. Thus a and b leave the same remainder upon division by d. Now suppose that a and b leave the same remainder upon division by d. Then there exist $q_1, q_2, r \in \mathbb{Z}$ such that $a = q_1 d + r$ and $b = q_2 d + r$ and $0 \leq r < d$. Now $r = b - q_2 d$, so

$$a = q_1 d + r = q_1 d + b - q_2 d = b + (q_1 - q_2)d.$$

Hence, by definition, $a \equiv b \pmod{d}$. $\square$

Example

Example: $-17 \equiv 15 \pmod{8}$ since $(-17) - 15 = -32 = (-4)8$.

≡ partitions the integers

For any $d \in \mathbb{N}$ and any $a \in \mathbb{Z}$ the **congruence class $[\mathbf{a}]_\mathbf{d}$** (or ‘equivalence class’) of a modulo d is defined by

$$[a]_d = \{m \in \mathbb{Z} \mid m \equiv a \pmod{d}\}.$$

Lemma: R_d induces the partition $\{[0]_d, [1]_d, \ldots, [d-1]_d\}$ on $\mathbb{Z}$

Example: $\mathcal{P} = \{[0]_5, [1]_5, [2]_5, [3]_5, [4]_5\}$ is a partition of $\mathbb{Z}$

Q: Can you see how the Q-R theorem can be used to prove the lemma?

Modular arithmetic

For any $a_1, a_2, b_1, b_2 \in \mathbb{Z}$ and $d \in \mathbb{N}$:

$$\boxed{\begin{array}{l} a_1 \equiv a_2 \pmod{d} \\ b_1 \equiv b_2 \pmod{d} \end{array}} \implies \boxed{\begin{array}{l} a_1 \pm b_1 \equiv a_2 \pm b_2 \pmod{d} \\ a_1 \times b_1 \equiv a_2 \times b_2 \pmod{d} \end{array}}$$

Example:

Since

$$27 \equiv -1 \pmod{7} \text{ and } 36 \equiv 1 \pmod{7}$$

we have

$$27{+}36 \equiv -1{+}1 \equiv 0 \pmod{7}$$

and

$$27{\times}36 \equiv -1{\times}1 \equiv -1 \pmod{7}$$

A key idea

When computing “modulo d”, you may at any time replace a number by something to which it is equivalent. In this way, you may simplify computations so that you never have to work with large integers.

A key idea

When computing "modulo *d*", you may at any time replace a number by something to which it is equivalent. In this way, you may simplify computations so that you never have to work with large integers.

Examples:

$$379 - 803 \equiv 1 - 5 \equiv -4 \equiv 3 \pmod 7$$

and

$$379 \times 803 \equiv 1 \times 5 \equiv 5 \pmod 7$$

and

$$803^5 \equiv 5^5 \equiv 25 \times 25 \times 5 \equiv \; 4 \times 4 \times 5 \equiv 80 \equiv 3 \pmod 7$$

An important problem

The following problem is called the **discrete logarithm problem**: Given $d \in \mathbb{N}$ and $A, Q \in \{1, \dots, d-1\}$, find $x \in \{1, \dots, d-1\}$ such that $A^x \equiv Q \pmod{d}$.

A naive solution to the problem: Compute $A^1 \mod d, A^2 \mod d, \dots$ until one of your computations produces Q.

A randomised naive solution to the problem: Repeatedly, select a number t from $\{1, 2, \dots, d-1\}$ at random and compute $A^t \mod d$. Stop when one of your computations produces Q.

Your privacy on the internet often relies on the following fact: For certain choices of d, A and Q, the naive solution to the discrete logarithm problem will take a very long time and the randomised naive solution to the discrete logarithm problem will almost certainly take a very long time.