# Section 8.3: Base-$b$ representation

## CS1231S Discrete Structures

Wong Tin Lok

National University of Singapore

2 October 2020

Which of the following is a multiple of 9?

- 800001000 ✓
- 800002000
- 800003000
- 800004000

Answer at `https://pollev.com/wtl/`.

# What we saw

## Definition 8.1.1

Let $n, d \in \mathbb{Z}$. Then $d$ is said to *divide* $n$ if

$$n = dk \quad \text{for some } k \in \mathbb{Z}.$$

We write $d \mid n$ for "$d$ divides $n$", and $d \nmid n$ for "$d$ does not divide $n$".

## Theorem 8.1.16 (Division Theorem) and Definition 8.1.17

For all $n \in \mathbb{Z}$ and $d \in \mathbb{Z}^+$, there exist unique $q, r \in \mathbb{Z}$ such that

$$n = dq + r \quad \text{and} \quad 0 \leqslant r < d.$$

Such $q$ and $r$ are denoted $n \,\underline{\mathrm{div}}\, d$ and $n \,\underline{\mathrm{mod}}\, d$ respectively.

## Definition 8.2.1

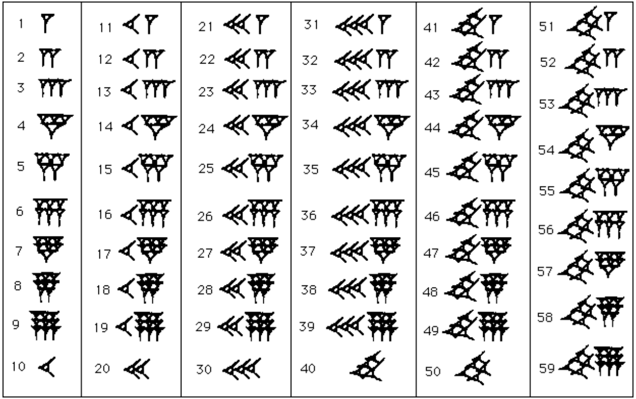
(1) A positive integer is *prime* if it has exactly two positive divisors.

(2) A positive integer is *composite* if it has (strictly) more than two positive divisors.

## Theorem 8.2.8 (Euclid)

There are infinitely many prime numbers.

# Base-$b$ representation — Why?




← digits used by ancient Babylonians in their base-60 system

Picture source: `https://www-history.mcs.st-andrews.ac.uk/HistTopics/Babylonian_numerals.html`

- To represent 1231 in a tallying system, one uses 1231 strokes.
- To represent 1231 in the base-10 system, one uses 4 symbols.

- Exponentially fewer symbols are needed: $4 = \lceil \log_{10}(\overbrace{1231}^{n} + 1) \rceil$.
- Finitely many symbols are enough to represent infinitely many numbers.

Our main focus: an algorithm to change bases

# Base-$b$ representation

Fix $b \in \mathbb{Z}_{\geqslant 2}$.

## Definition 8.3.1

The *base-$b$ representation* of a positive integer $n$ is

$$(a_\ell a_{\ell-1} \dots a_0)_b$$

where $\ell \in \mathbb{Z}_{\geqslant 0}$ and $a_0, a_1, \dots, a_\ell \in \{0, 1, \dots, b-1\}$ such that

$$n = a_\ell b^\ell + a_{\ell-1} b^{\ell-1} + \cdots + a_0 b^0 \quad \text{and} \quad a_\ell \neq 0. \tag{$*$}$$

The $a_0, a_1, \dots, a_\ell$ here are called *digits*.

## Convention 8.3.2

We identify a positive integer with its base-$b$ representation.

## Example 8.3.3

(1) $1231 = \underline{1} \times 10^3 + \underline{2} \times 10^2 + \underline{3} \times 10^1 + \underline{1} \times 10^0 = (1231)_{10}$.

(2) $182 = \underline{2} \times 3^4 + \underline{0} \times 3^3 + \underline{2} \times 3^2 + \underline{0} \times 3^1 + \underline{2} \times 3^0 = (20202)_3$.

## Theorem 8.3.13 (main theorem of this lecture)

For any $n \in \mathbb{Z}^+$, there exist unique $\ell \in \mathbb{Z}_{\geqslant 0}$ and $a_0, a_1, \dots, a_\ell \in \{0, 1, \dots, b-1\}$ such that $(*)$ holds.

## Special bases

Convention 8.3.5. use A, B, C, D, E, F for 10, 11, 12, 13, 14, 15 respectively.

### Definition 8.3.4

(1) Base-10 representations are called *decimal representations*.

(2) Base-2 representations are called *binary representations*.

(3) Base-8 representations are called *octal representations*.

(4) Base-16 representations are called *hexadecimal representations*.

(5) Base-60 representations are called *sexagesimal representations*.

### Example 8.3.6

(1) $(1231)_{10}$ is the decimal representation of 1231.

(2) $(1000011)_2$ is the binary representation of 67 because

$$1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 67.$$

(3) $(117)_8$ is the octal representation of 79 because $1 \times 8^2 + 1 \times 8^1 + 7 \times 8^0 = 79$.

(4) $(4D)_{16}$ is the hexadecimal representation of 77 because $4 \times 16^1 + 13 \times 16^0 = 77$.

# Algorithm for finding base-$b$ representation

**Algorithm 8.3.8** ($b \in \mathbb{Z}_{\geqslant 2}$ fixed)

1. **input** $n \in \mathbb{Z}^+$
2. $q := n$
3. $\ell := 0$
4. **while** $q \neq 0$ **do**
5. $\quad a_\ell := q \,\underline{\text{mod}}\, b$ collect remainders
6. $\quad q := q \,\underline{\text{div}}\, b$ remove by base
7. $\quad \ell := \ell + 1$
8. **end do**
9. **output** $(a_{\ell-1} a_{\ell-2} \ldots a_1 a_0)_b$

**Definition 8.1.17.** $q \,\underline{\text{div}}\, b$ and $q \,\underline{\text{mod}}\, b$ denote respectively the quotient and the remainder when $q$ is divided by $b$.

**Example 8.3.9.** $(b, n) = (8, 1511)$

$$
\begin{array}{r|l}
8 & 1511 \\ \hline
8 & 188 \quad — \; 7 \quad \to a_0 \\ \hline
8 & 23 \quad — \; 4 \quad \to a_1 \\ \hline
8 & 2 \quad — \; 7 \quad \to a_2 \\ \hline
 & 0 \quad — \; 2 \quad \to a_3
\end{array}
\uparrow
$$

So $1511 = (2747)_8$.

**Example 8.3.10.** $(b, n) = (16, 1511)$

$$
\begin{array}{r|l}
16 & 1511 \\ \hline
16 & 94 \quad — \; 7 \quad \to a_0 \\ \hline
16 & 5 \quad — \; 14 = \text{E} \quad \to a_1 \\ \hline
 & 0 \quad — \; 5 \quad \to a_2
\end{array}
\uparrow
$$

So $1511 = (5\text{E}7)_{16}$.

# Why does this algorithm stop?

Algorithm 8.3.8 ($b \in \mathbb{Z}_{\geqslant 2}$ fixed)

**input** $n \in \mathbb{Z}^+$<br>
$q := n$<br>
$\ell := 0$<br>
**while** $q \neq 0$ **do**<br>
    $a_\ell := q \,\underline{\text{mod}}\, b$<br>
    $q := q \,\underline{\text{div}}\, b$<br>
    $\ell := \ell + 1$<br>
**end do**<br>
**output** $(a_{\ell-1} a_{\ell-2} \dots a_1 a_0)_b$

Definition 8.1.17. $q \,\underline{\text{div}}\, b$ and $q \,\underline{\text{mod}}\, b$ denote respectively the quotient and the remainder when $q$ is divided by $b$.

- Let $q_i$ be the value of the variable $q$ when the stopping condition $q \neq 0$ of the **while** loop is checked the $(i+1)$th time.
- Then, as $b \geqslant 2$,

strictly smaller and decreasing but larger than 0

$$\begin{aligned} n = q_0 &> q_0 \,\underline{\text{div}}\, b = q_1 \\ &> q_1 \,\underline{\text{div}}\, b = q_2 \\ &> q_2 \,\underline{\text{div}}\, b = q_3 \\ &\ \vdots \end{aligned}$$

- Since each $q_i \geqslant 0$, the **while** loop can be executed at most $n$ times.
- In particular, this algorithm stops.

Note 8.3.12. We implicitly used the Well-Ordering Principle here to deduce that, since

$$\{q_0, q_1, q_2, \dots\} \subseteq \mathbb{Z}_{\geqslant 0}$$

is nonempty, it must have a smallest element.

therefore this proves that every number >= 2, has a base b representation

# Why is this algorithm correct?

Algorithm 8.3.8 ($b \in \mathbb{Z}_{\geqslant 2}$ fixed)

**input** $n \in \mathbb{Z}^+$
$q := n$
$\ell := 0$
**while** $q \neq 0$ **do**
  $a_\ell := q \,\underline{\mathrm{mod}}\, b$
  $q := q \,\underline{\mathrm{div}}\, b$
  $\ell := \ell + 1$
**end do**
**output** $(a_{\ell-1} a_{\ell-2} \ldots a_1 a_0)_b$

Definition 8.1.17. $q \,\underline{\mathrm{div}}\, b$ and $q \,\underline{\mathrm{mod}}\, b$ denote respectively the quotient and the remainder when $q$ is divided by $b$.

- Let $q_i$ be the value of the variable $q$ when the stopping condition $q \neq 0$ of the **while** loop is checked the $(i+1)$th time.
- Suppose the stopping condition of the **while** loop is checked $\ell + 1$ times in total, where $\ell \in \mathbb{Z}_{\geqslant 0}$.
- Then $q_{\ell-1} > 0$ and $q_\ell = 0$ by the stopping condition of the **while** loop.
- As $q_i = bq_{i+1} + a_i$ for each $i \in \{0, 1, \ldots, \ell - 1\}$,

sub in

$$\begin{aligned} n = q_0 = \underline{bq_1 + a_0} &= b(\underline{bq_2 + a_1}) + a_0 \\ &= b^2 q_2 + a_1 b + a_0 = b^2(bq_3 + a_2) + a_1 b + a_0 \\ &= b^3 q_3 + a_2 b^2 + a_1 b + a_0 \\ &\vdots \\ &= \cancel{b^\ell q_\ell} + a_{\ell-1} b^{\ell-1} + \cdots + a_1 b + a_0 \\ &= a_{\ell-1} b^{\ell-1} + \cdots + a_1 b + a_0 \qquad \text{as } q_\ell = 0. \end{aligned}$$

because 'ql' == 0

- Also $a_{\ell-1} = bq_\ell + a_{\ell-1} = q_{\ell-1} > 0$.

non-zero because executed the loop again to 'ql'

- So $n = (a_{\ell-1} a_{\ell-2} \ldots a_1 a_0)_b$.

# Uniqueness of base-$b$ representation — a proof by Strong MI

$\boxed{b \in \mathbb{Z}_{\geqslant 2}}$

2.1. For each $n \in \mathbb{Z}^+$, let $P(n)$ be the proposition "$n$ has at most one base-$b$ representation".

2.2. (Base step)

$0\ (a_1 \ldots a_0) = (a_0)_b = c_b$ ∴ unique

2.2.1. Let $c \in \{1, 2, \ldots, b-1\}$. Suppose $c = a_\ell b^\ell + a_{\ell-1} b^{\ell-1} + \cdots + \boxed{a_0 b^0}$ and $a_\ell \neq 0$, where $\ell \in \mathbb{Z}_{\geqslant 0}$ and $a_0, a_1, \ldots, a_\ell \in \{0, 1, \ldots, b-1\}$.

base-b representation of c

2.2.2. If we have $i \in \{1, 2, \ldots, \ell\}$ such that $a_i \geqslant 1$, then

since i >= 1

$$b-1 \geqslant c = a_\ell b^\ell + a_{\ell-1} b^{\ell-1} + \cdots + a_0 b^0 \geqslant a_i b^i \geqslant 1 \cdot b^1 = b,$$

which contradicts the choice of $c$.

leading digit must be 0

2.2.3. This means $a_1 = a_2 = \cdots = a_\ell = 0$, and so $\ell = 0$.

cause must have a value at least

2.2.4. Thus $c = a_0 \underbrace{b^0}_{=1} = a_0$.

2.2.5. Hence all base-$b$ representations of $c$ must be the same as $(c)_b$.

2.2.6. So $P(c)$ is true.

2.3. (Induction step)

Let $k \in \mathbb{Z}_{\geqslant b-1}$ such that $P(1), P(2), \ldots, P(k)$ are true. ...... So $P(k+1)$ is true.

2.4. Hence $\forall n \in \mathbb{Z}^+\ P(n)$ is true by Strong MI.

# Uniqueness of base-$b$ representation — a proof by Strong MI

$b \in \mathbb{Z}_{\geqslant 2}$

2.1. For each $n \in \mathbb{Z}^+$, let $P(n)$ be the proposition "$n$ has at most one base-$b$ representation".

2.3. (Induction step) 2.3.1. Let $k \in \mathbb{Z}_{\geqslant b-1}$ such that $P(1), P(2), \ldots, P(k)$ are true.

2.3.2. Let $\ell, m \in \mathbb{Z}_{\geqslant 0}$ and $a_0, a_1, \ldots, a_\ell, d_0, d_1, \ldots, d_m \in \{0, 1, \ldots, b-1\}$ such that

$$a_\ell b^\ell + a_{\ell-1} b^{\ell-1} + \cdots + a_0 b^0 = k+1 = d_m b^m + d_{m-1} b^{m-1} + \cdots + d_0 b^0 \qquad (*)$$

and $a_\ell > 0$ and $d_m > 0$.

2.3.3. The quotients one gets when these are divided by $b$ are equal too, i.e.,

$$a_\ell b^{\ell-1} + a_{\ell-1} b^{\ell-2} + \cdots + a_1 b^0 = (k+1)\,\underline{\text{div}}\,b = d_m b^{m-1} + d_{m-1} b^{m-2} + \cdots + d_1 b^0. \qquad (\dagger)$$

2.3.4. Note that $1 \leqslant (k+1)\,\underline{\text{div}}\,b \leqslant (k+k)\,\underline{\text{div}}\,2 = k$ because $k+1 \geqslant b \geqslant 2$.

2.3.5. So $P((k+1)\,\underline{\text{div}}\,b)$ is true by the induction hypothesis, i.e., $(k+1)\,\underline{\text{div}}\,b$ has at most one base-$b$ representation.

2.3.6. This implies $\ell = m$ and $a_i = d_i$ for all $i \in \{1, 2, \ldots, \ell\}$ in view of $(\dagger)$.

2.3.7. Substituting these back into $(*)$ gives

$$\begin{aligned} a_\ell b^\ell + a_{\ell-1} b^{\ell-1} + \cdots + a_1 b^1 + a_0 b^0 &= d_m b^m + d_{m-1} b^{m-1} + \cdots + d_1 b^1 + d_0 b^0 \\ &= a_\ell b^\ell + a_{\ell-1} b^{\ell-1} + \cdots + a_1 b^1 + d_0 b^0. \end{aligned}$$

2.3.8. Thus $a_0 = a_0 b^0 = d_0 b^0 = d_0$.

2.3.9. So $P(k+1)$ is true.

2.4. Hence $\forall n \in \mathbb{Z}^+ \;\; P(n)$ is true by Strong MI.

# Summary

## What we saw

- base-$b$ representation
- an algorithm for finding it, together with a proof that it always stops and gives the correct result
- uniqueness of base-$b$ representation

> A mathematical understanding of this concept of correctness is useful beyond the field of program verification. It provides a way of thinking that can improve all aspects of writing programs and building systems.
>
> Leslie Lamport 2018

## Theorem 8.3.13 (main theorem of this lecture)

For any $b \in \mathbb{Z}_{\geqslant 2}$ and any $n \in \mathbb{Z}^+$, there exist unique $\ell \in \mathbb{Z}_{\geqslant 0}$ and $a_0, a_1, \ldots, a_\ell \in \{0, 1, \ldots, b-1\}$ such that

$$n = a_\ell b^\ell + a_{\ell-1} b^{\ell-1} + \cdots + a_0 b^0 \quad \text{and} \quad a_\ell \neq 0.$$

## Next

- greatest common divisor
- the Euclidean Algorithm
- a proof of the Fundamental Theorem of Arithmetic