CSEN102
Introduction to Computer Science

Lecture 7:
Representing Information I

Prof. Dr. Slim Abdennadher
Dr. Aysha Alsafty, slim.abdennadher@guc.edu.eg,
aysha.alsafty@guc.edu.eg

German University Cairo, Department of Media Engineering and Technology

02.12.2017 - 07.12.2017

1 Synopsis

1.1 Synopsis

Synopsis

What you have learned so far:

- The idea of *algorithms*
  - The idea of the meta-solution, history, aspects, definition
- *Notation* and *principal components* of algorithms
  - pseudo-code
  - sequential, conditional, and iterative control-flow
- *Aspects* and *efficiency* of algorithms
  - Analysis of time and space efficiency, depending on the input size
  - Order of magnitude: constant, linear, cubic, ... ($O(1)$, $O(n)$, $O(n^2)$, ...)

In theory, you are now able to write an *algorithm* for *any* given *computable function*.

2 Number systems

2.1 Notation through history

And now to something completely different…

Positional numbering systems: decimal
1, 10, 100, … are all *powers of ten*!

The meaning of a decimal number:
$9 \times 10^0 + 3 \times 10^1 + 1 \times 10^2 + 3 \times 10^3 + 2 \times 10^4 + 6 \times 10^5 + 5 \times 10^6 = 5,623,139$

- This is why it is called decimal
- The position determines the power of 10, with which the *digit at the position has to be multiplied*!
- The first position is with $10^0 = 1$, the second with $10^1 = 10$, and so on…

Finding the perfect base

Question
How do I *choose* a *good base*?

- Which one is best? *10*? *20*? *60*?

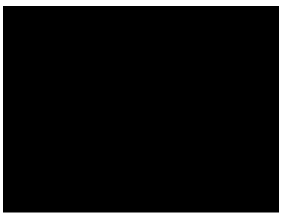
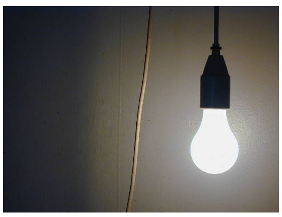
- Base 10 allows to counting with your *fingers*
- Base 20 allows to counting with your *fingers and toes*
- Base 60 is a clever *mathematical choice* because it has *many factors* (1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 60)
- Another clever *mathematical choice* would be a *prime number* as base such as 7 or 11 (opposite idea).

2.2 Computer numbering system

The right numbering system

Question
What base should we chose for a *Computer*?

Computers run with electricity
There are *two* distinct states of most electrical devices (including the transistor which is the basis of computers):


off

on

$\Rightarrow$ So we should chose a *base two* system!

3 Binary numbers

3.1 Binary position system

Introduction to binary numbers

For the *binary* system we use the powers of *2*:

$$2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8, 2^4 = 16, \ldots$$

Example 1.

$$5,623,139_{10} = 10101011100110101100011_2$$

Observations

- We need *two* different symbols (on–off, true–false, 1–0, high–low, *etc.*)
- To write the decimal number 5,623,139 in binary, we need *23 digits*.
- A modern computer usually uses *64 digits*, which allows for roughly *18.4 thousand trillion numbers*.

Terms and names

Some terms:

Bit : The *single binary digit* (0 or 1), the smallest unit of information.

Byte : *Eight bit*, which means up to *256* different numbers in positional binary.

Word : The base number of digits used by a computer, usually *eight byte* or *64 bit*.

3.2 Using binary

Familiar bases

Observation

It is *tedious* to switch between binary and decimal!

It is much *easier* with these bases:

- *Octal*, or base 8
- *Hexadecimal*, or base 16 (do not mix up with “hexagesimal”!)

Octal, hexadecimal, and binary

Imagine a numbering system with base *8* (Octal)

- Numbers: 0, 1, 2, 3, 4, 5, 6, 7

Example translation:

$$101\underbrace{101}_{5}001\underbrace{001}_{1}011\underbrace{011}_{3}110\underbrace{110}_{6}{}_2 = 5136_8$$

Imagine a numbering system with base *16* (Hexadecimal)

- Numbers: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Example translation:

$$1010\underbrace{1010}_{\mathrm{A}}0101\underbrace{0101}_{5}1110\underbrace{1110}_{\mathrm{E}}{}_2 = \mathrm{A5E}_{16}$$

3.3 First summary

Summary

- We are used to a *base 10* positional system
- *Other bases* (20, 60) were used through history
- Generally, a base n system encodes numbers as follows:

$$(x_i x_{i-1} \dots x_1 x_0)_n = x_i \times n^i + x_{i-1} \times n^{i-1} + \dots + x_1 \times n^1 + x_0 \times n^0$$

- We can *convert* any positional system into any other positional system
  1. Write down the digits
  2. Multiply each digit by its positional value (the respective power of the base)
  3. Add the products
- Some conversions are very *convenient* (binary–octal, binary–hexadecimal, ...)
- Binary is *ideal for computers*

3.4 Conversions

Two important conversions: Binary to decimal

Problem:
Convert a binary number into decimal

1. Write down the binary number
2. Write down the positional weight (the factor)
3. Multiply each digit by its weight
4. Do the sum

Example 2.

Number:	1	0	1	1	0	0	0	1
Positional weight:	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Factor:	128	64	32	16	8	4	2	1

$128 + 0 + 32 + 16 + 0 + 0 + 0 + 1 = 177_{10}$

Two important conversions: Decimal to binary

Problem:
Convert a decimal number into binary

Solution by *successive division*:

1. Divide by the base (*i. e.*, 2) and write down the remainder
2. Repeat division until the quotient equals 0
3. Read the binary number by reading the remainders (bottom-up)

	Division	Quotient	Remainder	
	43/2	21	*1*	
	21/2	10	*1*	
Convert *43* into binary	10/2	5	*0*	$43_{10} = 101011_2$
	5/2	2	*1*	
	2/2	1	*0*	
	1/2	0	*1*	

Conversion in general

Both algorithms work for any base!

Example 3 (Convert (base 60) to decimal:).

Number:		
Positional weight:	60^1	60^0
Factor:	60	1

$1500 +$ $46 = 1546_{10}$

Example 4 (Convert 1546 (base 10) to hexadecimal).

Division	Quotient	Remainder
1546/16	96	10
96/16	6	0
6/16	0	6

$1546_{10} = 60\text{A}_{16}$

Conversion in general

Hence you can now *convert any base to any other base*

Problem

Convert N_1 with base n to N_2 with base m

Solution:

1. Convert N_1 of base n to a decimal number N
2. Convert N to the number N_2 base m

Note:

The conversions also work directly as a transition from base n to base m (without the indirection over decimal).

It is just *unusual*.