二进制入门

王立斌

School of Computer Science, South China Normal University

October 8, 2020

Table of contents

1. 导引
2. 认识二进制
3. 二进制与十进制之间的关系
4. 位置计数法
5. 加法
6. 乘法

开场白

世界上有 10 种人：懂二进制的与不懂二进制的。

学习目标

- 二进制的基本定义、规则、属性
- 二进制的若干操作：加法、减法、乘法
- 二进制的位置计数法、二级制与十进制之间的关系
- 进而推广到各种不同进制之间的关系
- 涉及到加法、乘法的算法

数字系统

- 自然数、有理数、实数、复数
- 十进制、二进制、十六进制、八进制

数字系统

- 自然数、有理数、实数、复数
- 十进制、二进制、十六进制、八进制

一个重要的事实.

计算系统中信息只以 0 和 1 两种数位进行表达，称为二进制数字系统。每一个数位称为一个*比特*。

二进制.

认识二进制

$$十进制数 : 0, 1, 2, \cdots , n$$

二进制.

认识二进制

$$十进制数：0, 1, 2, \cdots, n$$

$$二进制数：0, 1, 10, 11, 100, \cdots$$

二进制规则一

二进制规则一：$0+0=0$, $0+1=1$，$1+1=10$。即所谓“逢二进一”，与十进制的“逢十进一”相对应。

二进制加法实例.

二进制规则一

二进制规则一：$0+0=0$, $0+1=1$，$1+1=10$。（红色的 1 是“进位”）

二进制加法

任意给出两个二进制数（你可以不知道它们代表什么十进制数）1001 和 101，求 $1001+101$。

$$
\begin{array}{cccc}
1 & 0 & 0 & 1 \\
+0 & 1 & 0 & 1 \\
\hline
=1 & 1 & 1 & 0
\end{array}
$$

观察二进制数字

十进制与二进制数字的对比.

- 十进制:

$$0, 1, 2, 3, 4, 5, 6, 7, 8, \cdots$$

- 二进制:

$$0, 1, 10, 11, 100, 101, 110, 111, 1000$$

观察二进制数字

十进制与二进制数字的对比.

- 十进制:

$$0,1,2,3,4,5,6,7,8,\cdots$$

- 二进制:

$$0,1,10,11,100,101,110,111,1000$$

二进制规则二

二进制规则二：尾数为 0 是偶数；尾数为 1 是奇数。

观察二进制数字

十进制与二进制数字的对比.

- 十进制： $0, 1, 2, 3, 4, 5, 6, 7, 8, \cdots$
- 二进制： $0, 1, 10, 11, 100, 101, 110, 111, 1000, \cdots$

观察二进制数字

十进制与二进制数字的对比.

- 十进制： $0, 1, 2, 3, 4, 5, 6, 7, 8, \cdots$
- 二进制： $0, 1, 10, 11, 100, 101, 110, 111, 1000, \cdots$

二进制规则三

二进制规则三：任意一个二进制数 x，尾巴上 “加” 一个 0，等于乘 2。

观察二进制数字

十进制与二进制数字的对比.

- 十进制：$0, 1, 2, 3, 4, 5, 6, 7, 8, \cdots$
- 二进制：$0, 1, 10, 11, 100, 101, 110, 111, 1000, \cdots$

二进制规则三

二进制规则三：任意一个二进制数 x，尾巴上“加”一个 0，等于乘 2。

二进制的规律

二进制与十进制数之间的对应关系：0 等于 0, 10 等于 2，100 等于 4，1000 等于 8，难道 10000 会不等于 16 吗?

观察二进制数字

二进制的规律

二进制与十进制数之间的对应关系：0 等于 0, 10 等于 2，100 等于 4，1000 等于 8，确实，10000 等于 16！

能推广到一般情况吗？

观察二进制数字

二进制的规律

二进制与十进制数之间的对应关系：0 等于 0, 10 等于 2，100 等于 4，1000 等于 8，确实，10000 等于 16!

能推广到一般情况吗?

二进制规则四

二进制规则四：任意给定一个二进制数 $b_i\underbrace{00\cdots0}_{i}$，$b_i=1$：

$$b_i\underbrace{00\cdots0}_{i}=2^i$$

二进制规则四

二进制规则四实例

任意给二进制数 $1\underbrace{00\cdots0}_{10}$,

$$1\underbrace{00\cdots0}_{10} = 2^{10} = 1024$$

观察二进制数字

以二进制规则四为基础可得二进制规则五

二进制规则五

二进制规则五：任意给定一个二进制数 $b_i\underbrace{00\cdots0}_{i}$,

$$b_i\underbrace{00\cdots0}_{i} - 1 = 2^i - 1 = \underbrace{11\cdots11}_{i}$$

二进制规则五

二进制规则五实例

任意给二进制数 $1\underbrace{00\cdots0}_{10}$，求 $1\underbrace{00\cdots0}_{10}-1$。

$$1\underbrace{00\cdots0}_{10}-1=2^{10}-1=\underbrace{11\cdots11}_{10}=1023$$

练习.

CSers 的计算方法.

请计算以下等式:

$$1 + 2 + 2^2 + 2^3 + \cdots + 2^{10}$$

观察二进制数字

尾巴加零等于乘二

继续规则三的分析。刚才我们都没有考虑奇数，会不会有其他问题，比如，考虑 7，

- 十进制数 7 的二进制是：111
- 二进制 111 后加一个 0：1110
- 二进制 1110 等于十进制的 14，因为.....

观察二进制数字

尾巴加零等于乘二

继续规则三的分析。刚才我们都没有考虑奇数，会不会有其他问题，比如，考虑 7，

- 十进制数 7 的二进制是：111
- 二进制 111 后加一个 0：1110
- 二进制 1110 等于十进制的 14，因为.....

问题！

似乎我们还不知道如何把二进制转换为十进制。

二进制转换为十进制

二进制转换为十进制

二进制 1110 等于十进制的哪一个数?

$$
\begin{array}{rrrr}
1 & 0 & 0 & 0 \\
+0 & 1 & 0 & 0 \\
+0 & 0 & 1 & 0 \\
\hline
=1 & 1 & 1 & 0
\end{array}
$$

即 1000 等于 8，100 等于 4，10 等于 2（规则四），将这几个十进制数加起来：$8+4+2=14$，大家的强项！所以二进制 1110 等于十进制的 14。

二进制与十进制之间的关系

二进制与十进制的转换规律

上述计算规则是否具有一般性？给定任意一个二进制数，记为 $b_n, b_{n-1}, \cdots b_0$，其中 $b_i \in \{0, 1\}$。

$$
\begin{array}{cccc}
b_n & 0 & \cdots & 0 \\
+0 & b_{n-1} & \cdots & 0 \\
+\ 0 & \cdots & 0 & \\
+0 & 0 & \cdots & b_0 \\
\hline
=b_n & b_{n-1} & \cdots & b_0
\end{array}
$$

然后，任意一个 $b_i 0 \cdots 0$（代表上式的某一行数值）等于什么？

二进制与十进制之间的关系

二进制与十进制的转换规律

上述计算规则是否具有一般性？给定任意一个二进制数，记为 $b_n, b_{n-1}, \cdots b_0$，其中 $b_i \in \{0, 1\}$。

$$
\begin{array}{cccc}
b_n & 0 & \cdots & 0 \\
+0 & b_{n-1} & \cdots & 0 \\
+\ 0 & \cdots & 0 & \\
+0 & 0 & \cdots & b_0 \\
\hline
=b_n & b_{n-1} & \cdots & b_0
\end{array}
$$

然后，任意一个 $b_i 0 \cdots 0$（代表上式的某一行数值）等于什么？任意一个 $b_i 0 \cdots 0 = b_i * 2^i$ 所以

$$b_n, b_{n-1}, \cdots b_0 = \sum_{i=0}^{n} b_i * 2^i$$

位置计数法（Position Notation）

二进制的位置计数法

给定任意一个二进制数，记为 $B = b_n, b_{n-1}, \cdots b_0$，其中 $b_i \in \{0, 1\}$。

$$B = \sum_{i=0}^{n} b_i * 2^i = b_n 2^n + b_{n-1} 2^{n-1} + \cdots + b_0 2^0$$

其中，2 就是二进制的 Base，每一个 b_i 就是一个比特。

位置计数法（Position Notation）

二进制的位置计数法

给定任意一个二进制数，记为 $B = b_n, b_{n-1}, \cdots b_0$，其中 $b_i \in \{0, 1\}$。

$$B = \sum_{i=0}^{n} b_i * 2^i = b_n 2^n + b_{n-1} 2^{n-1} + \cdots + b_0 2^0$$

其中，2 就是二进制的 Base，每一个 b_i 就是一个比特。

能否推广到任意进制?

十进制的位置计数法.

十进制的位置计数法

给定任意一个十进制数，记为 $D = d_n, d_{n-1}, \cdots d_0$，其中 $b_i \in \{0, 1, \cdots, 9\}$。

$$D = \sum_{i=0}^{n} d_i * 10^i = d_n 10^n + d_{n-1} 10^{n-1} + \cdots + d_0 10^0$$

其中，10 就是十进制的 Base，每一个 d_i 就是一个十进制数位。

十进制的位置计数法.

十进制的位置计数法

给定任意一个十进制数，记为 $D = d_n, d_{n-1}, \cdots d_0$，其中 $b_i \in \{0, 1, \cdots, 9\}$。

$$D = \sum_{i=0}^{n} d_i * 10^i = d_n 10^n + d_{n-1} 10^{n-1} + \cdots + d_0 10^0$$

其中，10 就是十进制的 Base，每一个 d_i 就是一个十进制数位。

十六进制的位置计数法你会了吗?

二进制的运算规则

回顾

二进制规则三

二进制规则三：任意一个二进制数 x，尾巴上“加”一个 0，等于乘 2。

在 C 语言中任意一个**int** a，a = a << 1代表 a 左移一个比特且最后比特补 0，即乘 2。请问a = a >> 1，即对 a 进行一次右移会不会是除 2?

二进制的运算规则

回顾

二进制规则三

二进制规则三：任意一个二进制数 x，尾巴上“加”一个 0，等于乘 2。

在 C 语言中任意一个**int** a，a = a << 1代表 a 左移一个比特且最后比特补 0，即乘 2。请问a = a >> 1，即对 a 进行一次右移会不会是除 2?

二进制规则六

二进制规则六：任意一个二进制数 x，如果右移一个比特（最高位补 0)，等于整除 2。

二进制除法

二进制规则六

二进制规则六：任意一个二进制数 x，如果右移一个比特（最高位补 0），等于整除 2。

十进制的位置计数法

15 的二进制比特是 1111，右移一比特得 111，这是 7，$15/2 = 7$。注意，是整除！

十进制数转换为二进制数.

十进制转换为二进制数.

给定任意一个十进制数 D，注意，此时 D 必然有一种二进制表达 B，

$$D = \sum_{i=0}^{n} b_i * 2^i = b_n 2^n + b_{n-1} 2^{n-1} + \cdots + b_0 2^0$$

第一步，我们可以知道这个 D 如果表达为二进制最后一个比特是什么？如果是偶数则为 0，否则为 1。我们怎么能知道这个数是奇数还是偶数呢？二进制规则二。

十进制数转换为二进制数.

十进制转换为二进制数.

给定任意一个十进制数 D，注意，此时 D 必然有一种二进制表达 B，

$$D = \sum_{i=0}^{n} b_i * 2^i = b_n 2^n + b_{n-1} 2^{n-1} + \cdots + b_0 2^0$$

第一步，我们可以知道这个 D 如果表达为二进制最后一个比特是什么？如果是偶数则为 0，否则为 1。我们怎么能知道这个数是奇数还是偶数呢？二进制规则二。
第二步，剩下的数位怎么办？扔掉最后一个比特，重复第一步！什么是扔掉最后一个比特，除 2 或者等价地，右移一个比特。

十进制数转换为二进制数的伪代码

Listing 1: 十进制数转换为二进制数的伪代码

```
function Dec2Bin(a)
# Input: a decimal number a;
# Output: the binary digits of a;
if a <= 1: # 终止条件!
    return a;
else:
    return Dec2Bin(a/2)||lsb(a) # ||是字符连接的意思.
# a/2 表示a右移了一个比特
# lsb(a)是a的最低位比特，其实就是a % 2，或者a & 1.
```

二进制运算

二进制运算分两大类

- 逻辑运算：与、或、非、异或、与非等
- 算术运算：加、减、乘、除

算术运算往往借助于逻辑运算。

二进制的逻辑运算

- 与 (&)： $0\&0=0$， $0\&1=0$, $1\&1=1$
- 或 (|)：
- 非 ($\sim$)：
- 异或 ($\wedge$)： $0\wedge 0=0$， $0\wedge 1=1$, $1\wedge 1=0$

诡异的加法.

整数加法

```
//输入: 两个整数a和b
//输出: a与b的和
int add(int a, int b) {

    if (b == 0) return a;
    return add(a ^ b, (b & a) << 1);

}
```

理解加法.

要理解这个加法算法只需要地正确回答出以下三个问题:

1. $a \wedge b$ 得到的是什么?
2. $(b\&a) << 1$ 得到的是什么?
3. 该算法为什么会终止?

Naïve Multiplication.

```
# Input: two integers a and b, where a >= b.
# Output: the product of a and b.
def naive_multiply(a, b):
    if b == 0:
        return 0
    return a + naive_multiply(a, b - 1)
```

问题!

能不能做得更高效?

Rule of Multiplication.

Rule of Multiplication.

$$a \cdot b = \begin{cases} 2(a \cdot \lfloor b/2 \rfloor) & \text{if } b \text{ is even;} \\ a + 2(a \cdot \lfloor b/2 \rfloor) & \text{if } b \text{ is odd.} \end{cases} \tag{1}$$

Simple multiplication.

```
# Input: two integers a and b.
# Output: the product of a and b.
def multiply(a, b):
  if b == 0:
    return 0;
  if is_even(b): # the last bit of b is 0;
      return 2*multiply(a, b/2);
  else: # b is odd
      return 2*multiply(a, b/2) + a;
```

两种乘法的区别.

问题

朴素乘法与简单乘法的区别在哪里？如何可以一眼看出简单乘法更高效?

两种乘法的区别.

问题

朴素乘法与简单乘法的区别在哪里？如何可以一眼看出简单乘法更高效?

答案

朴素乘法的迭代次数是 b 的数值大小；简单乘法的迭代次数是 b 的比特长度。考虑一下 $2^{10} = 1024$，1000 以内数值的比特长度是 10。

Simple multiplication.

Intuition.

For two n-bits integers a and b, view b as a binary number in position notation(位置记数法).

$$
\begin{aligned}
a \cdot b &= a \cdot (b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \cdots + b_1 2 + b_0) \\
&= b_{n-1}(a \cdot 2^{n-1}) + b_{n-2}(a \cdot 2^{n-2}) + \cdots + b_1(a \cdot 2) + b_0 a
\end{aligned}
$$

Observations: 1. Every term in the last equation has $2^i a$, which means we must continue to do $a* = 2$
2. The bit b_i in every term plays as a flag, when $b_i == 1$, we must have $res+ = a$;

Simple multiplication.

Example.

To compute 3×10.

Simple multiplication.

Example.

To compute 3×10.
10's binary string is 1010.

Simple multiplication.

Example.

To compute 3×10.
10's binary string is 1010.
Repeatly multiply 3 by 2^i and get: 3, 6, 12, 24.

Simple multiplication.

Example.

To compute 3×10.
10's binary string is 1010.
Repeatly multiply 3 by 2^i and get: 3, 6, 12, 24.
Choose the right terms to add, they are 6 and 24.

Simple multiplication.

Example.

To compute 3×10.
10's binary string is 1010.
Repeatly multiply 3 by 2^i and get: 3, 6, 12, 24.
Choose the right terms to add, they are 6 and 24.
Return the result which is 30.

Happy ending.

请用 C 语言完成以下编程作业。

- 请用 C 语言实现is_even函数，输入一个整数，判定其是否偶数，如果是返回 1，否则返回 0。
- 请用 C 语言实现这样一个函数 (prt_binary)，输入一个整数，在屏幕输出打印这个整数的二进制比特。
- 请用 C 语言实现这样一个函数 (binary_reverse)，输入一个正整数，将这个整数二进制比特倒序排列，并返回相应的整数。
- 请用 C 语言编程实现课件中的加法（add），输入两个正整数 a 和 b，返回 $a+b$，但是不能使用 $+$。
- 请用 C 语言编程分别用递归法和迭代法实现课件中提及的简单乘法（mul），输入两个正整数 a 和 b，返回 $a*b$，但是不能使用 $*$。