

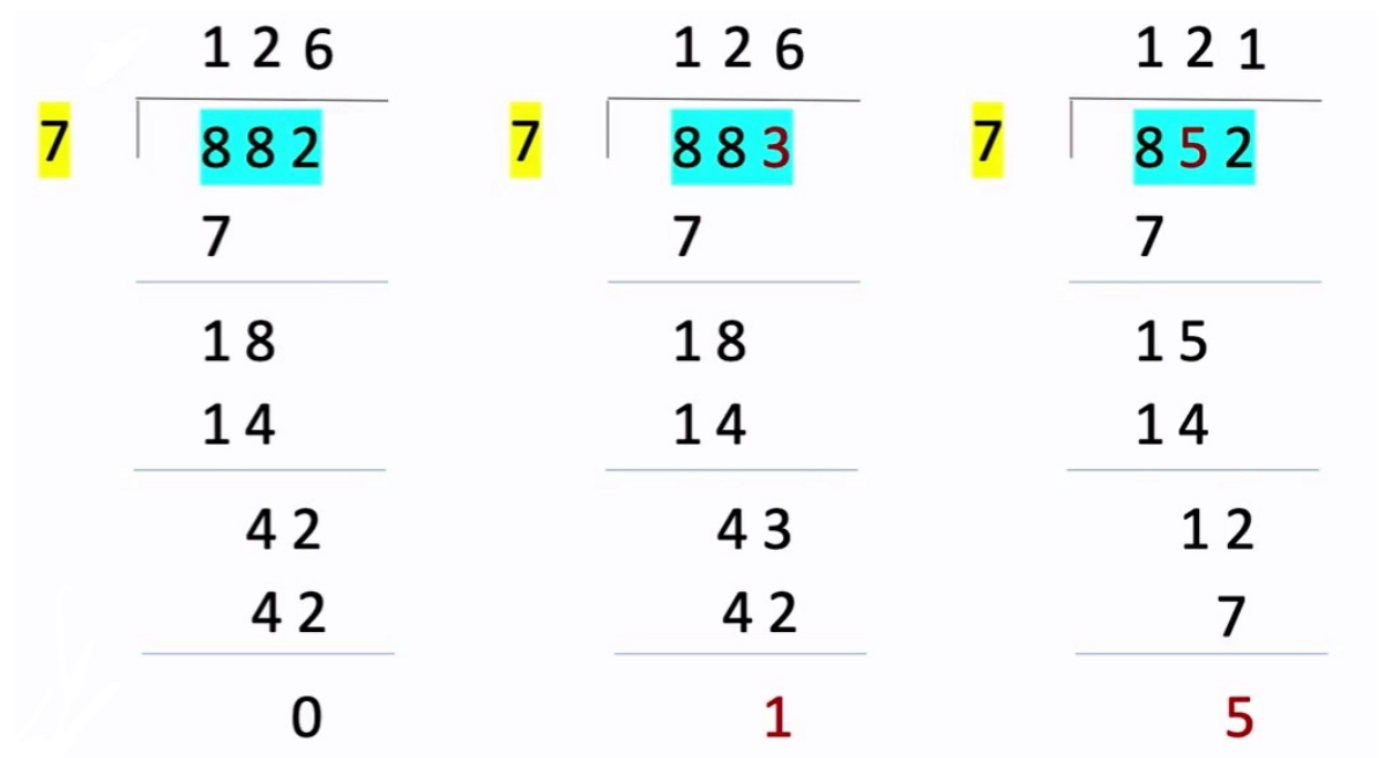
检错编码——循环冗余校验码（CRC 码）

日期: 2024 年 10 月 20 日

知识总览

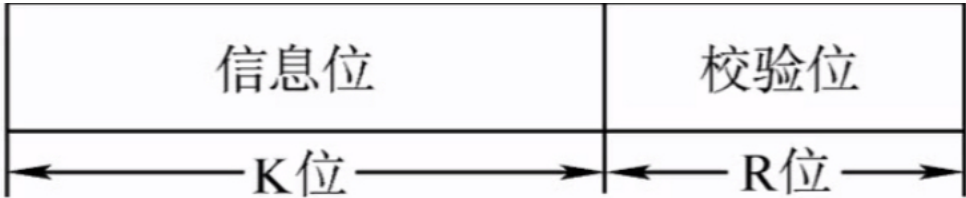
- 循环冗余校验码（CRC 码）
 - CRC 码的基本思想
 - 如何构造
 - 如何检错纠错
- 循环冗余校验：Cyclic Redundancy Check（CRC）

循环冗余校验码的基本思想



- 以上校验的思想就是检查余数是否发生错误
- 循环冗余校验码的思想
 - 数据发送、接收方约定一个“除数”（二进制除数）
 - K 个信息位 + R 个校验位作为“被除数”，添加校验位后需保证除法的余数为 0

循环冗余校验码



• 例：设生成多项式为 $G(x) = x^3 + x^2 + 1$ ，信息码为 101001，求对应的 CRC 码

1. 确定 K、R 以及生成多项式对应的二进制码

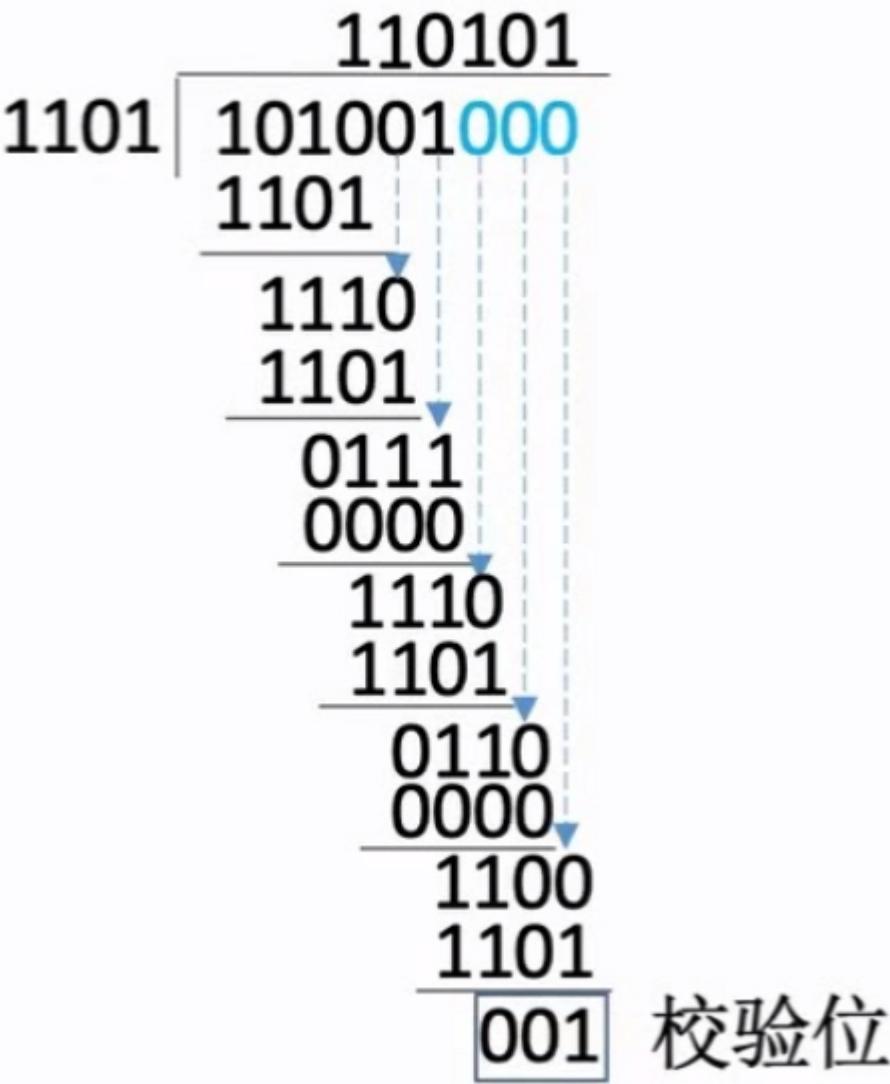
- K = 信息码的长度 = 6, R = 生成多项式的最高次幂 = 3 \rightarrow 校验码位数 $N = K + R = 9$
- 生成多项式 $G(x) = 1 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$ ，对应二进制码**1101**

2. 移位

- 将信息码左移 R 位，低位补 0，即得到 101001**000**

3. 相除

- 对移位后的信息码，用生成多项式进行**模 2 除法**，产生余数.过程如图



▪ “模 2 除”“模 2 减”，模 2 减就是按位异或

- 校验位：**001**
- 得出对应的 CRC 码：**101001 001**



4. 检错和纠错

- 发送：101001001, 记为 $C_9C_8C_7C_6C_5C_4C_3C_2C_1$
- 接收：101001001, 用 1101 进行模 2 除→余数为 000, 代表没有出错
- 接收：101001011, 用 1101 进行模 2 除→余数为 010, 代表 C_2 出错

接受	余数	出错位
101001 01 0	001	1
101001 01 1	010	2
101001 1 01	100	3
10100 0 001	101	4
1010 1 1 001	111	5
101 1 01 001	011	6
10 0 001 001	110	7
1 1 1001 001	001	8
0 01001 001	010	9

- 出错位 2 和出错位 9 对应的余数相同，这是信息码过长导致的

• 再看一个例子

信息位: 0100
生成多项式: $G(x)=x^3+x^2+1$ (1101)  0100 000 对 1101 模二除, 余数为 011  CRC码: 0100 011

接受	余数	出错位
0100 01 0	001	1
0100 01 1	010	2
0100 1 01	100	3
010 1 001	101	4
01 1 0 001	111	5
0 000 001	011	6
1 100 001	110	7

- **重要结论**：对于确定的生成多项式 $G(x)$ ，出错位与余数是相对应的；只要数据位数没有超过余数所能表示的范围，那么余数和出错位是**一一对应的**

- 对于 K 个信息位， R 个校验位，若生成多项式选择得当，且 $2^R \geq K + R + 1$ （000 对应正确），则 CRC 码**可以纠正 1 错位**
 - 但在实际计算机网络的应用中，CRC 校验码一般只用于检错，不用于纠错，因为信息码很长校验码很短
- **理论而言**，可以证明循环冗余验证码的检错能力有以下特点：
 - 可检测出所有奇数个错误
 - 可检测出所有双比特的错误
 - 可检测出所有小于等于校验位长度（ R ）的连续错误
- **CRC 码名字中带“循环”的原因**（不重要）
 - 数据循环移位（即将数据从一端移至另一端），生成的 CRC 码将保持相同的形式