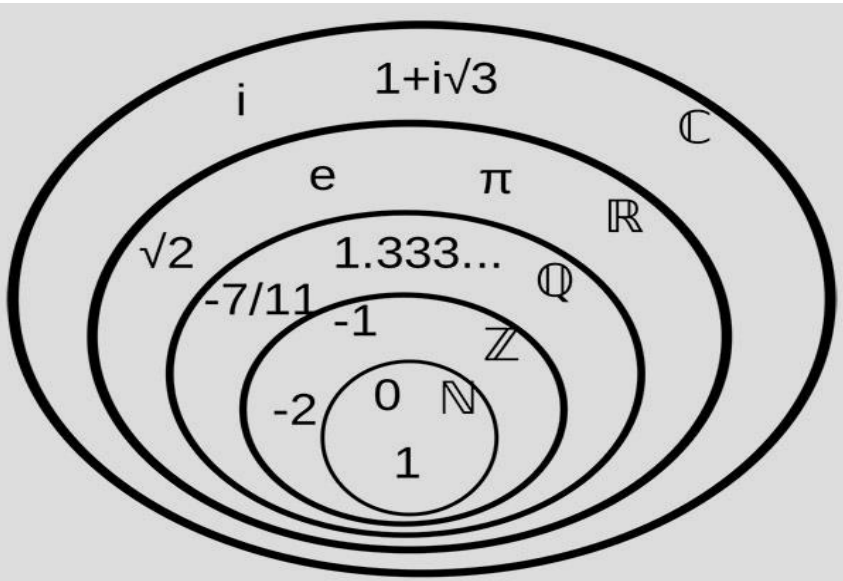

第1.6节 同余应用

Section 1.6: Applications of Congruences

知识要点

1. 散列函数
2. 伪随机数
3. 校验码

散列函数

- 定义: 一个**散列函数**h将内存地址$h(k)$分配给以k为键值的.
  - 最常用的散列函数是$h(k) = k \bmod m$, 其中m是可供使用的内存地址的数目.
  - 散列函数是满射的(也称作映上函数), 这样所有内存地址均可用.

- 例:令$h(k) = k \bmod 111$. 找出分配给社会保障号分别为064212848, 037149212, 107405723的客户记录的内存地址.
- 解:
  - h(064212848) = 064212848 **mod** 111 = 14,
  - h(037149212) = 037149212 **mod** 111 = 65,
  - h(107405723) = 107405723 **mod** 111 = 14

散列函数

- 解(续):
  - 在以上结果中, 注意因为散列函数不是一对一的, 所以有可能多个记录被分配到同一个内存地址, 例如上面的064212848和107405723, 这时叫做**冲突**.
  - 消解冲突的一个办法就是使用散列函数分配已被占用地址后面的第一个未占用的地址, 107405723的分配的内存地址则变为15.

伪随机数

- 在计算机中随机选择的数经常会被用到. 目前已经设计了许多方法来产生具有随机选择性质的数, 但因为由系统方法产生的数并不真正是随机的, 所以被称为**伪随机数**.

- 我们可以利用线性同余法来产生伪随机数.

- 我们选择4个整数:模数m, 倍数a, 增量c, 种子x_0, 满足 $2 \leq a < m$, $0 \leq c < m$, $0 \leq x_0 < m$. 通过连续地应用下面的递归函数, 来产生随机数序列$\{x_n\}$, 满足对于所有的n, $0 \leq x_n < m$:

$$x_{n+1} = (ax_n + c) \bmod m$$

伪随机数

- 例: 利用线性同余法来生成伪随机数序列, 其中模m=9, 倍数 a=7, 增量c=4, 种子x_0=3.
- 解: 连续应用递归定义的函数$x_{n+1} = (7x_n + 4)$ **mod** 9, 其中 x_0=3.
  - $x_1 = 7x_0 + 4$ **mod** $9 = 7\cdot3 + 4$ **mod** $9 = 25$ **mod** $9 = 7$,
  - $x_2 = 7x_1 + 4$ **mod** $9 = 7\cdot7 + 4$ **mod** $9 = 53$ **mod** $9 = 8$,
  - $x_3 = 7x_2 + 4$ **mod** $9 = 7\cdot8 + 4$ **mod** $9 = 60$ **mod** $9 = 6$,
  - $x_4 = 7x_3 + 4$ **mod** $9 = 7\cdot6 + 4$ **mod** $9 = 46$ **mod** $9 = 1$,
  - $x_5 = 7x_4 + 4$ **mod** $9 = 7\cdot1 + 4$ **mod** $9 = 11$ **mod** $9 = 2$,
  - $x_6 = 7x_5 + 4$ **mod** $9 = 7\cdot2 + 4$ **mod** $9 = 18$ **mod** $9 = 0$,
  - $x_7 = 7x_6 + 4$ **mod** $9 = 7\cdot0 + 4$ **mod** $9 = 4$ **mod** $9 = 4$,
  - $x_8 = 7x_7 + 4$ **mod** $9 = 7\cdot4 + 4$ **mod** $9 = 32$ **mod** $9 = 5$,

伪随机数

- 解(续):
  - $x_9 = 7x_8 + 4 \bmod 9 = 7 \cdot 5 + 4 \bmod 9 = 39 \bmod 9 = 3$.
  - 由于$x_9 = x_0$，而且每一项都只依赖于其前面的一项，所以产生序列3,7,8,6,1,2,0,4,5,3,7,8,6,1,2,0,4,5,3,...
  - 这个序列中包含9个不同的数，然后重复.
- 以上伪随机数重复太快，因此大部分计算机生成伪随机数时使用增量$c = 0$. 这种的生成器称为**纯倍式生成器**. 例如$2^{31} - 1$为模，7^5为倍数的纯倍式生成器就被广泛使用，它在重复之前会产生的$2^{31} - 1$个数.

校验码

- 同余可用于检查数字串中的错误.
- **奇偶校验位**: 数字信息一般用位串表示, 并划分为指定大小的块. 每个块在存储之前，为末尾额外添加一位，称为奇偶校验位. 位串$x_1x_2x_3...x_n$的奇偶校验位x_{n+1}定义为$x_{n+1}=x_1+x_2+x_3+...+x_n$ **mod** 2.
- 由此得出如果在这n位中有偶数个1位, 则x_{n+1}=0；如果有奇数个1位, 则x_{n+1}=1.
- 在检查阶段, 如果奇偶校验位错了那我们就知道位串中有错误. 但它也有局限性, 可以检测到前面n位中奇数个错误, 不能检查到偶数个错误.

校验码

- 例:接受到位串01100101和11010110, 其中每个位串都以一个奇偶校验位结束, 这些位串是正确的吗?
- 解:首先将这些位串假定为正确的. 我们来检测它的奇偶校验位.
  - 第一串(01100101)的奇偶校验位应该为0+1+1+0+0+1+0 **mod** 2=1, 而传输过来的奇偶校验位也是1, 所以奇偶校验位是正确的.
  - 第二串(11010110)的奇偶校验位应该为1+1+0+1+0+1+1 **mod** 2=1, 而传输过来的奇偶校验位是0, 所以奇偶校验位不是正确的.
  - 因此我们可以得出结论, 第一串可能是正确的(说可能是因为它有可能包含偶数个错误, 奇偶校验位检测不出来), 但第二串一定是错误的.

校验码:UPC

- 零售产品通常由其**通用产品代码标识**(Universal Product Code,UPC).
- UPC最常用的形式是12位十进制数字$x_1x_2x_3...x_{11}x_{12}$, 第一位数字标识产品种类(x_1), 接着五位标识制造商($x_2x_3x_4x_5x_6$), 再五位标识特定产品($x_7x_8x_9x_{10}x_{11}$), 最后一位是校验码(x_{12}). 校验码由同余式决定:

$$3x_1+x_2+3x_3+x_4+3x_5+x_6+3x_7+x_8+3x_9+x_{10}+3x_{11}+x_{12}\equiv 0(mod\ 10)$$

校验码:UPC

- 例:假设UPC的前11位为79357343104, 求最后一位校验码是多少?
- 解:将校验码x_{12}代入上面的同余式3·7 + 9 + 3·3 + 5 + 3·7 + 3 + 3·4 + 3 + 3·1 + 0 + 3·4 + x_{12} ≡ 0 (mod 10) , 简化后得98 + x_{12} ≡ 0 (mod 10). 由此x_{12} ≡ 2 (mod 10) , 校验位为2.

- 例:041331021641是否是一个合法的UPC?
- 解:将这些数字代入同余式, 得3·0 + 4 + 3·1 + 3 + 3·3 + 1 + 3·0 + 2 + 3·1 + 6 + 3·4 + 1 ≡ 0 (mod 10) , 但是0 + 4 + 3 + 3 + 9 + 1 + 0+ 2 + 3 + 6 + 12 + 1 = 44 ≡ 4 ≠ 0 (mod 10), 所以它不是一个合法的UPC.

校验码:ISBN

- 所有图书都由一个**国际标准书号**(International Standard Book Number, ISBN-10)标识.
- 一个由出版商指定的10位数代码$x_1x_2x_3...x_{10}$ (备注, 最新已经有ISBN-13, 这儿不讲这种更大量标识的国际标准书号), 它包含不同分组来表示语言、出版商、出版公司赋予图书的编号、最后一位校验码(可以是数字也可以是字母X, 其中字母X表示整数10). 其中校验码需要满足:

$$x_{10} = \sum_{i=1}^{9} ix_i \ (\mathbf{mod}\ 11)$$

或者等价于$\sum_{i=1}^{10} ix_i \equiv 0\ (mod\ 11)$

校验码:ISBN

- 例:某书的ISBN-10的前9位为 007288008. 求校验码是多少?
- 解:将前9位代入同余式可得x_{10} = 1·0 + 2·0 + 3·7 + 4·2 + 5·8 + 6·8 + 7· 0 + 8·0 + 9·8 ($\boldsymbol{mod}$ 11). 故校验码x_{10}= 2.

- 例:084930149X是否是合法的ISBN-10?
- 解:将10位一起代入第二个同余式可得1·0 + 2·8 + 3·4 + 4·9 + 5·3 + 6·0 + 7· 1 + 8·4 + 9·9 + 10·10 =0 + 16 + 12 + 36 + 15 + 0 + 7 + 32 + 81 + 100 = 299 ≡ 2 ≠ 0 (mod 11), 故它不是一个合法的 ISBN-10.

本章知识点回顾

- **a|b**:存在整数c使得$b = ac$
- **a和b模m同余**:$m|a - b$
- **模算术**:以一个整数m（m大于2）为模数所做的计算
- **素数**:大于1且恰只有1和它自身两个正因子的整数
- **合数**:大于1又不是素数的整数
- **gcd(a,b)**:能整除a和b的最大整数
- **互素整数**:满足$\gcd(a, b) = 1$的整数a和b
- **lcm(a,b)**:能被a和b整除的最小正整数
- $\boldsymbol{n = (a_k a_{k-1 \ldots} a_1 a_0)_b}$:$n$的$b$进制表示

本章知识点回顾

- **二进制表示**:整数以2为基数的表示
- **八进制表示**:整数以8为基数的表示
- **十六进制表示**:整数以16为基数的表示
- **贝祖系数**:$sa + tb = \gcd(a, b)$成立的整数s和t
- **a模m的逆**:$\bar{a}a \equiv 1 (mod\ m)$成立的整数$\bar{a}$
- **线性同余方程**:$ax \equiv b (mod\ m)$,其中x为整数变量
- **以b为基数的伪素数**:$b^{n-1} \equiv 1 (mod\ n)$成立的合数$n$
- **卡米切尔数**:合数n使得对所有满足$\gcd(b, n) = 1$的正整数b, n是以b为基数的伪素数

本章知识点回顾

- **素数p的原根**:Z_p中的整数r使得每个不能被p整除的整数模p同余r的一个幂次
- **以r为底a模p的离散对数**:满足0≤ e ≤ $p-1$,$r^e \equiv a\ (mod\ p)$的整数e
- **移位密码**:将明文p加密成$(p+k)$ **mod** m的密码，其中k为整数
- **仿射密码**:将明文p加密成$(ap+k)$ **mod** m的密码，其中a和k是整数,且$\gcd(a,26)=1$
- **字符密码**:逐个字符地加密的密码
- **分组密码**:按等长字符分组加密的密码

本章知识点回顾

- **密码系统**:一个五元组(p, c, k, e, d), p是明文消息的集合, c是密文消息的集合, k是密钥的集合, e是加密函数的集合, d是解密函数的集合
- **私钥加密**:加密和解密密钥都需要保密的加密
- **公钥加密**:加密密钥公开, 解密密钥保密的加密
- **RSA**:密码系统,加密密钥$k = (n, e)$, 其中$n = pq$,其中p和q是大素数, e是正整数. 解密密钥d是e模$(p-1)(q-1)$的逆. $E_k(p) = p^e \bmod n$, $D_k(c) = c^d \bmod n$.

本章总结与展望：数论的无限魅力

華中科技大學
计算机科学与技术学院
School of Computer Science &Technology，HUST

数论核心概念

整除性与模运算

整除性是数论的基础，模运算提供了处理周期性问题的强大工具，广泛应用于计算机科学和密码学。

素数与欧几里得算法

素数是数论的基石，欧几里得算法提供了计算最大公约数的高效方法，这些概念构成了现代密码学的基础。

同余方程与定理

费马小定理、欧拉定理和中国剩余定理为解决同余方程提供了强大工具，在密码学和计算机科学中有着广泛应用。

密码学应用

RSA等公钥密码系统将数论的抽象概念转化为保护数字世界安全的实用工具，展示了纯数学与现实应用的完美结合。

未来展望

量子计算挑战

量子计算的发展对传统密码构成挑战，推动了后量子密码学的研究，基于格密码学和超奇异同源的新算法正在成为研究热点

区块链与去中心化系统

数论将继续为区块链技术提供理论基础，零知识证明等高级密码学原语将推动去中心化系统的发展和应用场景拓展

跨学科融合

数论与人工智能、生物信息学等领域的交叉研究将产生新的突破，数学模型将在更多领域发挥关键作用

数论的魅力不仅在于其优雅的理论体系，更在于它与现实世界的紧密联系.