Based on K. H. Rosen: Discrete Mathematics and its Applications.

Lecture 15: Applications of modular congruences. Section 4.5

1 Application of modular congruences

Congruences have many applications to discrete mathematics, computer science, and many other disciplines.

1.1 Hashing

The central computer at an insurance company maintains records for each of its customers. How can memory locations be assigned so that customer records can be retrieved quickly? The solution to this problem is to use a suitably chosen **hashing function**. Records are identified using a **key**, which uniquely identifies each customer's records. A hashing function h assigns memory location $h(k)$ to the record that has k as its key. One of the most common is the function is

$$h(k) = k \bmod m,$$

where m is the number of available memory locations.

1.2 Pseudorandom numbers

Randomly chosen numbers are often needed for computer simulations. Different methods have been devised for generating numbers that have properties of randomly chosen numbers. Because numbers generated by systematic methods are not truly random, they are called pseudorandom numbers.
The most commonly used procedure for generating pseudorandom numbers is the **linear congruential method**. We choose four integers: the **modulus** m, **multiplier** a, **increment** c, and **seed** x_0 ,with $2 \leq a < m$, $0 \leq c < m$, and $0 \leq x_0 < m$. We generate a sequence of pseudorandom numbers $\{x_n\}$, with $0 \leq x_n < m$ for all n, by successively using the recursively defined function

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

Remark 1. To generate such numbers, we divide numbers generated with a linear congruential generator by the modulus: that is, we use the numbers x_n/m. Most computers do use linear congruential generators to generate pseudorandom numbers. Often, a linear congruential generator with increment $c = 0$ is used. Such a generator is called a **pure multiplicative generator**. For example, the pure multiplicative generator with modulus $m = 2^{31} - 1$ and multiplier $a = 7^5 = 16,807$ is widely used.

Remark 2. Pseudorandom numbers generated by linear congruential generators have long been used for many tasks. Unfortunately, it has been shown that sequences of pseudorandom numbers gen erated in this way do not share some important statistical properties that true random numbers have. Because of this, it is not advisable to use them for some tasks, such as large simulations.

1.3 Check digits

Congruences are used to check for errors in digit strings. A common technique for detecting errors in such strings is to add an extra digit at the end of the string. This final digit, or check digit, is calculated using a particular function. Then, to determine whether a digit string is correct, a check is made to see whether this final digit has the correct value. We begin with an application of this idea for checking the correctness of bit strings.

Example 3. Parity Check Bits Digital information is represented by bit string, split into blocks of a specified size. Before each block is stored or transmitted, an extra bit, called a parity check bit, can be appended to each block. The parity check bit x_{n+1} for the bit string $x_1, x_2, \ldots, x_n$ is defined by

$$x_{n+1} = x_1 + x_2 + \cdots + x_n \mod 2$$

It follows that x_{n+1} is 0 if there are an even number of 1 bits in the block of n bits and it is 1 if there are an odd number of 1 bits in the block of n bits. When we examine a string that includes a parity check bit, we know that there is an error in it if the parity check bit is wrong. However, when the parity check bit is correct, there still may be an error. A parity check can detect an odd number of errors in the previous bits, but not an even number of errors.