Method of Complements

Number Theory Handout 28 Annotated

Modern computers internally represent numbers in binary, have limited precision (such as 32 bits, 64 bits, etc.), and can only add. How do they perform subtraction? One approach to performing the subtraction $x - y$ is to compute $x + (-y)$, but this requires being able to negate a number and being able represent negative numbers. One possibility[1] would be to have a single bit to denote the sign of the number (just as we write "+" or "−" before a number to indicate its sign). However, then a subtraction algorithm is still needed to add one positive and one negative number.

The *method of complements* was used with mechanical calculators that could only add, and was suggested to be used in electronic digital computers by John von Neumann. Many mechanical calculators were similar to electronic computers in that they had limited precision and could only add, but represented numbers in decimal instead of binary.

Let b be the base (either 10 or later 2) used in the device, and let n be the number of digits that can be stored. Additions are performed modulo b^n.

To perform the subtraction $x - y$ on a decimal device, we would like to form the negation of y modulo 10^n, which as a least residue would be $10^n - y$. This is known as the *ten's complement*. Unfortunately, performing the subtraction $10^n - y$ is not simple because of the need for carries.

The *nine's complement* of y instead computes $(10^n - 1) - y$. Note that $10^n - 1 = \underbrace{9999\cdots9}_{n \text{ digits}}$. The great advantage of computing the nine's complement of y is that no carries are needed when performing the subtraction $(10^n - 1) - y$. Each individual digit can be subtracted from 9. On many mechanical devices, the inputs were given secondary labels with these "complemented" digits.

The nine's complement of y can be used to perform the subtraction $x - y$ as follows:

$$x - y \equiv x + \big[(10^n - 1) - y\big] + 1 \pmod{10^n}.$$

Example 1. Consider a mechanical calculator with $n = 7$ digits. We wish to calculate $1234567 - 430967$. The nine's complement of 430967 is:

$$\begin{array}{rr} & 9999999 \\ - & 0430967 \\ \hline & 9569032 \end{array}$$

Adding, we obtain $1234567 + 9569032 \equiv 803599 \pmod{10^7}$. Adding 1, we obtain $803599 + 1 \equiv 803600 \pmod{10^7}$, which is the difference $1234567 - 430967$.

Discussion. This also gives a method for computing the ten's complement (ie, negation mod 10^n): compute the nine's complement and add one.

For a binary electronic digital computer with n bits, the *two's complement* is $2^n - y$, and the *one's complement* is $(2^n - 1) - y$. The one's complement is simple to compute since each individual bit of y is negated using NOT gates. The two's complement of y can be computed by calculating the one's complement and then adding 1.

[1] In fact, a sign bit is used when representing floating point numbers in a computer, but not when representing integers.

Example 2. For $n = 8$ bits, what is the two's complement of 52?

$$\begin{array}{rr} & 11111111 \\ - & 00110100 \\ \hline & 11001011 \\ + & 00000001 \\ \hline & 11001100 \end{array}$$

Note that $11001100_2 = 204 \equiv -52 \pmod{256}$.

Example 3. For $n = 8$ bits, what is the two's complement of 0?

$$\begin{array}{rr} & 11111111 \\ - & 00000000 \\ \hline & 11111111 \\ + & 00000001 \\ \hline & 00000000 \end{array}$$

So the two's complement of 0 is 0.

Example 4. For $n = 8$ bits, what is the two's complement of 1?

$$\begin{array}{rr} & 11111111 \\ - & 00000001 \\ \hline & 11111110 \\ + & 00000001 \\ \hline & 11111111 \end{array}$$

So the two's complement of 1 (aka, -1) has all bits set.

Signed representation of integers. Given an n-bit integer, we wish to interpret some as positive and some as negative. It is standard to interpret number as negative if the most significant bit is 1. What is the largest positive integer representable on an n-bit device? The smallest negative number?

Solution. The largest positive integer would be $0\underbrace{11\cdots1}_{n-1\text{ 1s}} = 2^{n-1}-1$. The smallest negative integer is $1\underbrace{00\cdots0}_{n-1\text{ 0s}}$, which is -2^{n-1}. The representatives of each residue class modulo 2^n are thus:

$$\underbrace{-2^{n-1}, -2^{n-1}+1, \ldots, -1}_{\text{signed}}, \overbrace{0, 1, \ldots, 2^{n-1}-1}^{\text{unsigned}}, 2^{n-1}, \ldots, 2^n - 1 \pmod{2^n}$$

For example, for $n = 8$, $2^n = 256$ and we have the following possible signed and unsigned 8-bit integers:

$$\underbrace{-128, -127, \ldots, -1}_{\text{signed}}, \overbrace{0, 1, \ldots, 127}^{\text{unsigned}}, 128, \ldots, 255 \pmod{256}$$