A BRIEF REFRESHER ON SOME MATH OFTEN USED IN COMPUTING
BY STEVE SUMMIT

There are a few mathematical concepts which figure prominently in programming. None of these involve higher math (or even algebra, and as we'll see, knowing algebra too well can make one particular programming idiom rather confusing). You don't have to understand these with deep mathematical sophistication, but keeping them in mind will help some things make more sense.

I. Integers vs. real numbers

An integer is a number without a fractional part, a number you could use to count things (although integers may also be negative). Mathematicians may distinguish between natural numbers and cardinal numbers, and linguists may distinguish between cardinal numbers and ordinal numbers, but these distinctions do not concern us here.

A real number is, for our purposes, simply a number with a fractional part. Since computers do not typically implement real numbers exactly, it is not necessary or even meaningful to distinguish between rational, irrational, and transcendental numbers.

II. Exponential Notation

Exponential or Scientific Notation is simply a method of writing a number as a base number times some power of ten. For example, we could write the number 2,000,000 as 2 x 10^6, the number 0.00023 as 2.3 x 10^-4, and the number 123.456 as 1.23456 x 10^2.

III. Binary Numbers

Our familiar decimal number system is based on powers of 10. The number 123 is actually 100 + 20 + 3 or 1 x 10^2 + 2 x 10^1 + 3 x 10^0.

The binary number system is based on powers of 2. The number 100101_{2} (that is, “100101 base two”) is 1 x 2^5 + 0 x 2^4 + 0 x 2^3 + 1 x 2^2 + 0 x 2^1 + 1 x 2^0 or 32 + 4 + 1 or 37.

We usually speak of the individual numerals in a decimal number as digits, while the “digits” of a binary number are usually called “bits.”

Besides decimal and binary, we also occasionally speak of octal (base 8) and hexadecimal (base 16) numbers. These work similarly: The number 45_{8} is 4 x 8^1 + 5 x 8^0 or 32 + 5 or 37. The number 25_{16} is 2 x 16^1 + 5 x 16^0 or 32 + 5 or 37. (So 37_{10}, 100101_{2}, 45_{8}, and 25_{16} are all the same number.)
IV. Boolean Algebra

Boolean algebra is a system of algebra (named after the mathematician who studied it, George Boole) based on only two numbers, 0 and 1, commonly thought of as “false” and “true.” Binary numbers and Boolean algebra are natural to use with modern digital computers, which deal with switches and electrical currents which are either on or off. (In fact, binary numbers and Boolean algebra aren’t just natural to use with modern digital computers, they are the fundamental basis of modern digital computers.)

There are four arithmetic operators in Boolean algebra: \textbf{NOT}, \textbf{AND}, \textbf{OR}, and \textbf{EXCLUSIVE OR}.

\textbf{NOT} takes one operand (that is, applies to a single value) and negates it: NOT 0 is 1, and NOT 1 is 0.

\textbf{AND} takes two operands, and yields a true value if both of its operands are true: 1 AND 1 is 1, but 0 AND 1 is 0, and 0 AND 0 is 0.

\textbf{OR} takes two operands, and yields a true value if either of its operands (or both) are true: 0 OR 0 is 0, but 0 OR 1 is 1, and 1 OR 1 is 1.

\textbf{EXCLUSIVE OR}, or XOR, takes two operands, and yields a true value if one of its operands, but not both, is true: 0 XOR 0 is 0, 0 XOR 1 is 1, and 1 XOR 1 is 0.

It is also possible to take strings of 0/1 values and apply Boolean operators to all of them in parallel; these are sometimes called “bitwise” operations. For example, the bitwise OR of 0011 and 0101 is 0111. (If it isn’t obvious, what happens here is that each bit in the answer is the result of applying the corresponding operation to the two corresponding bits in the input numbers.)