Modulo Arithmetic

Introduction

Modular arithmetic is at the heart of many mathematical techniques and procedures arising in Computer Science and Computer Engineering. Many books introduce this topic as ‘clock’ arithmetic, so, in keeping with tradition, I will do the same.

Consider a typical 12 hour clock. The only possible hour values are 1 thru 12. We frequently have to figure out what time it will be at some arbitrary number of hours from now. For instance, suppose it is now 8:00 and I want to know what time it will be 18 hours from now. Most of us just count on our fingers, resetting the count to zero each time we hit 12, as in the following counting sequence:

Add these numbers to 8    1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
to get            8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
reset at each 12     8 9 10 11 12 1 2 3 4 5 6 7 8 9 10 11 12 1 2

We thus figure that 18 hours from now it will be 2:00. Notice that we had to ‘reset’ our counter twice during this operation. Now observe that the actual count (without ‘resets’) was 26, and we needed to reset our count twice. That is, we ‘threw away’ 12 every time our count exceeded 12. How many times did we remove 12? Twice. It is important to realize at this point that this is the same amount as the quotient we would get if we divided 26 by 12.

\[
\frac{26}{12} = 2 \text{ with a remainder of } 2
\]

Further note that the remainder is the answer we originally wanted: 2. (The fact that this is the same as the quotient is pure coincidence). So let’s take another example. If the current time is 11:00, what time will it be in 32 hours? Simply add 32 to 11, divide the result by 12 and the remainder will be your answer:

\[
\frac{11 + 32}{12} = \frac{43}{12} = 3 \text{ with a remainder of } 7 \quad [M1]
\]

Therefore, 32 hours from now (11:00) is 7:00.

We have ignored the problem of determining whether the time we end up with is am or pm. To solve this problem it is better to use a military clock of 24 hours. Then, assuming 11:00 is 11:00am this is 1100 in military time and the time 32 hours from now would be

\[
\frac{1100+3200}{2400} = 1 \text{ with a remainder of } 1900, \quad [M2]
\]
so the new time is 1900 in military time, which is 7 in the evening.

Finding the remainder after division is what we call a **modulo operation**, where the divisor is called the **modulus**. Below we introduce two new arithmetic operators which are used to denote the operations of finding the remainder after division by some modulus, as well as finding the quotient.

**The Modulo Operators**

**MOD**  This operator provides the remainder of a number after division. Under normal division dividing a number $N$ by a dividend, $D$ produces a quotient, $Q$, and a remainder $R$ as follows:

\[
\frac{N}{D} = Q + \frac{R}{D} \tag{[M3]}
\]

For example, $13/4 = 3 + 1/4$. The result of the MOD operator is to produce $R$ from [M3]. That is:

\[
N \mod D = R = N - DQ \tag{[M4]}
\]

Some examples: $13 \mod 4 = 1$ (see the example above); $24 \mod 9 = 6$; $12 \mod 2 = 0$.

Equation [M1] now becomes $4300 \mod 1200 = 700$ and [M2] is $4300 \mod 2400 = 1900$.

Note that **the MOD operator always produces a result between 0 and $D-1$**; Notice that strictly speaking our clock example above is not really modulo arithmetic since the values on our clock range from 1 through 12 instead of 0 through 11. This is easily rectified by replacing 12 by $0^1$.

The modulo operator is not confined to positive values. **Any multiple of the modulus can be added to, or subtracted from, the original number without changing the remainder.** Looking at [M4], it should be clear that, since $Q$ is an integral number of $D$s we can add or subtract an arbitrary number of $D$s to/from $N$ without affecting $R$. Start with [M3]:

\[
\frac{N}{D} = Q + \frac{R}{D}.
\]

Add $KD$ to $N$ (an arbitrary multiple of $D$)

\[
1^1\text{In fact, in military time this is explicit: Midnight is known as 0000 hours.}
\]
\[
\frac{(N + KD)}{D} = \frac{N}{D} + K \\
= Q + \frac{R}{D} + K \\
= (Q + K) + \frac{R}{D} \\
= Q' + \frac{R}{D}
\]

where the Q’ is just some new integer (= Q + K). So, what time was it 13 hours ago if it is now 2:00? First, apply the MOD operator

\[
(2 - 13) \mod 12 = (-11) \mod 12
\]

This isn’t very helpful, but we know we can add any multiple of 12 to the left hand argument and the result will still be correct. Let’s add just one multiple of 12:

\[
(-11 + 12) \mod 12 = 1 \mod 12 = 1
\]

So 13 hours ago it was 1:00

**DIV** This operator is called integer division, and produces the quotient (Q in expression [M3]) of a normal division while throwing away the remainder:

\[
N \div D = Q \quad \text{[M5]}
\]

13 \div 4 = 3; 24 \div 9 = 2; D \div D = 1. Unlike R, which must be between 0 and D-1, there is no limit on the range of Q.

Expression [M3] could be rewritten as \(N/D = N \div D + (N \mod D)/D\).
Practice Problems:

Find the result of the following operations

1. 23 mod 7
2. -14 mod 3
3. 3 div 4
4. n mod n
5. n mod 1
6. 0 mod 4
7. 10 mod 10
8. 50 div 6
9. (n-1) mod n
10. (2n+1) mod 2
11. 1 mod 6
12. -5 mod 5
13. 8 div 8
14. 2n mod 2
15. kn div n

16. If today’s date is March 23, write the modulo arithmetic expression which would be used to calculate what the date will be 17 days from now.

17. Number the days of the week 0 (Sun) through 6 (Sat). If today is Tuesday,
   a. write the modulo arithmetic expression for the number of the day of the week 18 days from now;
   b. write the modulo arithmetic expression for the day of the week 9 days ago.

Congruence

It is often important to know whether or not two numbers have the same remainder when divided by the same modulus. When this is the case the two numbers are said to be congruent to each other. The symbol \( \equiv \) is used to signify this as follows:

\[ A \equiv B \mod m \]

means ‘A is congruent to B modulo m’. This is just a shorthand way of writing

\[ A \mod m = B \mod m. \]

We now state the laws of modulo arithmetic.

1. Addition, Subtraction, and Multiplication are valid under modulo arithmetic. If

\[ A \equiv C \mod m \text{ and } B \equiv D \mod m \]

then

\[ A + B \equiv C + D \mod m, \]
\[ A - B \equiv C - D \mod m, \]
\[ A \times B \equiv C \times D \mod m. \]
(A ± B) ≡ (C ± D) modulo m or A ± B mod m = C ± D mod m
AB ≡ CD modulo m or AB mod m = CD mod m

It follows from the above that

\[ A^n = C^n \text{ modulo } m \text{ and } (A \text{ mod } m)^n \text{ mod } m = A^n \text{ mod } m \]

Example 1: 13 = 16 mod 3 and 8 = 26 mod 3 then (13 + 8) = (16 + 26) mod 3 = 21 = 42 mod 3. Note that 21 mod 3 = 42 mod 3 = 0.

Example 2: Since 2 = 5 mod 3, \(2^3 = 5^3\) mod 3. \(8 = 125\) mod 3.

Example 3: \((6 \mod 4)^2\) mod 4 = \(6^2\) mod 4

2. Division is valid under modulo arithmetic if the divisor is relatively prime to the modulus. If

\[ AC = BD \text{ mod } m, \quad A = B \text{ mod } m, \quad \text{and } A \text{ is relatively prime to } m, \]

then

\[ C = D \text{ mod } M \text{ (i.e. we can divide out common factors).} \]

Example: Suppose 15 = 120 mod 7 and 3 = 10 mod 7 then
\[ 15/3 = 120/10 \text{ mod } 7 \text{ which is } 5 = 12 \text{ mod } 7 \]

3. A mod x = B mod x if and only if Ay mod xy = By mod xy. That is, we can multiply a modulus by some value if we multiply all congruencies by the same value.

4. If

\[ A \text{ mod } x = B \text{ mod } x, \quad A \text{ mod } z = B \text{ mod } z, \quad \text{and } x \text{ is relatively prime to } z \]

then

\[ A \text{ mod } xz = B \text{ mod } xz \]

**Application Notes**

Within a computer, the hardware which actually performs arithmetic stores operands
and results in hardware storage devices called *registers*. The size of a register (the number of digits it can hold, or store) is fixed and finite. Thus, the size of all operands and results are limited to what can be represented in these registers. These sizes also depend, as we will see, on how numbers are actually represented.

Suppose, for example, that such a register can hold only 4 decimal digits\(^2\). Then the only numbers (assuming positive integers only) that may be stored must be in the range 0 through 9999. Any arithmetic operation that produces a result greater than 9999 will 'overflow' the register leaving a result which is in the range of 0-9999. In effect all arithmetic operations are done using modulo 10000 arithmetic.

Example; Add 3245 to 8756 and place the result in a four-digit register. The result remaining in the four-digit register will be

\[(3245 + 8756) \mod 10000 = 12001 \mod 10000 = 2001.\]

If it is desired to represent negative numbers there are various ways to do so. It turns out that, depending on how we choose to represent negative numbers, having overflows as just demonstrated can be a good thing, and is used to simplify how computer hardware performs subtraction.

*Answers to Selected Practice Problems*

1. 2
2. 1
3. 0
4. 0
5. 0

16. If today's date is March 23, write the modulo arithmetic expression which would be used to calculate what the date will be 17 days from now.

\[(23 + 17) \mod 31 = 40 \mod 31\]

\(^2\)In most present day computers such registers actually hold binary, not decimal, digits.