Number Theory

Greatest Common Divisor and its applications.

Number Theory is a branch of mathematics that explores the properties of integers, most of the time only the natural numbers. Most problems in elementary Number Theory are easy to state and understand, because they're just extensions of grade school mathematics. At the same time, the solutions to these problems are not simple, usually requiring ingenious insights that are beautiful and fascinating. We will explore the problem of finding the greatest common divisor in this lecture.

Before we begin, there are a few definitions and notations that should be addressed.

Def: An integer a is said to be divisible by another integer b if we can write $a = bq$. (where q is known as the quotient of a and b) This is often denoted as $a|b$.

Def: Given any two integers a and b, there exists a unique pair of integers q and r such that $a = b \cdot q + r$ where $0 \le r < b$. This is known as the Division Algorithm. (it is actually a theorem, not an algorithm)

Def: Given two positive integers a and b, the greatest common divisor c is a number such that:

i) c is a common divisor of a and b. In other words, $c|a$ and $c|b$.

ii) for any other common divisors d of a and b, $d \le c$.

This is also denoted as gcd(a, b) = (a, b).

Consider the following methods of finding the gcd of a and b.

1) Brute force it! Consider just using a loop and checking if both integers are divisible by the loop counter. If it is then set it as the current maximum. The number of iterations this method would take is proportional to the minimum of a and b as no integer greater than this minimum will still be a factor of a and b. For large integers, this becomes infeasible.
2) Prime factorize both integers and take the intersection between them. However, factoring large integers is also infeasible.

As it turns out, there is a simple algorithm, called the Euclidean Algorithm that finds the gcd very quickly. Consider the division algorithm for a and b: (assuming $a > b$)

$$a = b \cdot q_1 + r_1 \qquad \text{for } 0 \le r_1 < b$$

Next, apply the division algorithm again, but this time using b and r_1:

$$b = r_1 \cdot q_2 + r_2 \qquad \text{for } 0 \le r_2 < r_1$$

And again with r and r_1:

$$\begin{aligned}
r_1 &= r_2 \cdot q_3 + r_3 && \text{for } 0 \le r_3 < r_{21} \\
r_2 &= r_3 \cdot q_4 + r_4 && \text{for } 0 \le r_4 < r_3 \\
&\vdots && \vdots \\
&\vdots && \vdots \\
r_{n-2} &= r_{n-1} \cdot q_n + r_n && \text{for } 0 \le r_n < r_{n-1} \\
r_{n-1} &= r_n \cdot q_{n+1} + 0
\end{aligned}$$

Notice that the remainders for each successive equation is decreasing, ie:

$$b > r_1 > r_2 > r_3 > r_4 > \ldots\ldots \geq 0$$

Since each remainder is a positive integer, this sequence cannot contain more than b terms and so there must exist a line where the remainder becomes 0.

Claim: $r_n = \gcd(a,b) = c$.

To prove this, we will first prove two lemmas.

Lemma 1:

If *a, b, m,* and *n* are integers, and if $c|a$ and $c|b$, then $c|(ma+nb)$.

Proof:

Since $c|a$ and $c|b$, there must exist integers e and f such that $a = ce$ and $b = cf$. Hence, $ma+nb = m(ce)+n(cf) = c(me+nf)$. Consequently, we see that $c|(ma+nb)$.

□

Lemma 2:

Let *a, b, and c* be integers. Then $(a + cb, b) = (a, b)$.

Proof:

Let *a, b, and c* be integers. We will show that the common divisors of a and b are exactly the same as the common divisors of $a + cb$ and b. This will show that $(a + cb, b) = (a, b)$. Let e be a common divisor of a and b. By Lemma 1, $e|(a + cb)$, so e is a common divisor of $a + cb$ and b. If f is a common divisor of $a + cb$ and b, then by Lemma 1, we see that f divides $(a + cb) - cb$, which is equal to a. So f is a common divisor of a and b. Hence, $(a+cb, b) = (a, b)$.

□

Now we are ready to proceed with the proof of the above claim.

Proof of Claim:

By Lemma 2, it follows that $(b \cdot q_1 + r_1, b) = (a, b) = (b, r_1) = (r_1, r_2) = (r_2, r_3) = \ldots\ldots\ldots =$ $(r_{n-2}, r_{n-1}) = (r_{n-1}, r_n) = (r_n, 0) = r_n$.

Enough with the proofs, here is the algorithm:

```
int gcd( int a, int b ) {
    if( b == 0 ) return a;
    else return gcd( b, a%b );
}
```

The algorithm follows directly from the above repeated division algorithm where the first column is the first argument and the second column (directly to the right of the equal sign) is the second argument to this function.

It turns out that the worst case input to the Euclidean Algorithm is two successive Fibonacci numbers. Given f_{n+1} and f_{n+2} successive Fibonacci numbers, then the Euclidean Algorithm takes exactly n divisions to show that $(f_{n+1}, f_{n+2}) = 1$.

Furthermore, Lamé's Theorem shows that the number of divisions needed to find the greatest common divisor of two positive integers using the Euclidean Algorithm does not exceed five times the number of decimal digits in the smaller of the two integers. It turns out that as a result, assuming that $a > b$, the $\gcd(a, b)$ can be found using $O((\log_2 a)^3)$ bit operations.

We can also solve the problem of finding the Least Common Multiple, denoted as $\text{lcm}(a, b)$, by using the following relation:

$$lcm(a,b) = \frac{a \cdot b}{\gcd(a,b)}$$

This is useful when we want to add two fractions with different denominators.

Now that we have seen how we could find the greatest common divisor of two numbers. Let's extend this idea further to find integers x and y such that $ax + by = (a, b)$. This is otherwise known as Bezout's Identity. We first show that (a, b) is the smallest integer that can be written in the form $ax + by$.

Proof:
Consider the set of all positive integers of the form $ax + by$. Let $k = au + bv$ be the smallest element of this set. (this must exist because of the Well-Ordering Principle) Let $\gcd(a, b) = c$. The goal of the proof is to show that $k = c$.
First we will show that $k|a$ and $k|b$.
If $a = 0$ then we $k|a$. If it is non-zero, then by the Division algorithm, there are integers q and r such that $a = kq + r$, where $0 \le r < k$. So:

$$\begin{aligned} r &= a - kq \\ &= a - (au + bv)q \\ &= a(1 - uq) + b(-vq) \end{aligned}$$

But since k is the smallest element of the set, and $r < k$, it must imply that $r = 0$. Thus $a = kq$ and so $k|a$.
The same argument will show that $k|b$.
Since c is the greatest common divisor of a and b, it follows that $c|a$ and $c|b$. In other words, $a = cr$ and $b = cs$, for integers r and s. Substituting this into our equation for k, we get:

$$\begin{aligned} k &= au + bv \\ &= (cr)u + (cs)v \\ &= c(ru + sv) \end{aligned}$$

Because k and c are both positive, $ru + sv$ must also be positive. Therefore we have that $k \ge c$. But c is the greatest common divisor so it must be the case that $k = c$. In other words, c is the smallest positive integer that can be written in the form $ax + by$.

□

To solve linear equations such as $ax + by = e$ for some integer e, it must be the case that $c|e$ and in this case, there are infinitely many solutions. If not, then there are no solutions. For now, let's look at how we can modify our code to find solutions to $ax + by = c$.

Consider again the division algorithm of a and b, where $b > a$.

$$a = b \cdot q_1 + r_1 \qquad \text{where } r_1 = a - b \cdot q_1$$

Then we can equivalently write the second equation as:

$$\begin{aligned} r_2 &= b - r_1 \cdot q_2 \\ &= b - (a - b \cdot q_1) \cdot q_2 \\ &= a(-q_2) + b(1 + q_1 q_2) \end{aligned}$$

The third equation as:

$$\begin{aligned} r_3 &= r_1 - r_2 q_3 \\ &= (a - bq_1) - (a(-q_2) + b(1 + q_1 q_2))q_3 \\ &= a(1 + q_2 q_3) + b(-q_1 - q_3 - q_1 q_2 q_3) \end{aligned}$$

Continuing in this fashion, we will eventually get to r_n as a linear combination of a and b, which is what we're after. Here is an implementation of the Extended Euclidean Algorithm.

```
// struct to hold our solution
struct Triple {
    int g, x, y;
    Triple( int d, int w, int e ) : g(d), x(w), y(e) {}
};

// extended gcd, returns <g, x, y> such that (A,B) = g and Ax + By = g
Triple egcd( int A, int B ) {
    if( B == 0 ) return Triple( A, 1, 0 );
    Triple tr = egcd( B, A%B );
    return Triple( tr.g, tr.y, tr.x – A/B * tr.y );
}
```

The code looks very similar to that of our original code, except we do a little more bookkeeping by using a struct. The first line is just as before, the base case of the recursion. If B is zero then the greatest common divisor is the first argument, and so A = A(1). Line two is the recursion, which solves the equation:

$$B \cdot tr.x + (A\%B) \cdot tr.y = tr.g \qquad (1)$$

but what we're after is

$$A \cdot tr2.x + B \cdot tr2.y = tr2.g = tr.g \qquad (2)$$

To find the relationship between $tr.x$ and $tr2.x$ and $tr.y$ and $tr2.y$, consider the division algorithm for A:

$$A = qB + r = \left\lfloor A/B \right\rfloor B + A\%B$$

Substituting this into (1), we get:

$$\begin{aligned} B \cdot tr.x + (A - \left\lfloor A/B \right\rfloor B) \cdot tr.y &= tr.g \\ B \cdot tr.x + A \cdot tr.y - \left\lfloor A/B \right\rfloor B \cdot tr.y &= tr.g \\ A \cdot tr.y + B \cdot (tr.x - \left\lfloor A/B \right\rfloor \, tr.y) &= tr.g \end{aligned}$$

Therefore, $tr2.x = tr.y$ and $tr2.y = tr.x - \lfloor A/B \rfloor \, tr.y$, which is exactly what the implementation does.

To solve the general equation of $ax + by = e$, we first need to check if c divides e. In other words, we need to check if $e = cq$. If it is, then we can run the above algorithm to find x and y, such that:

$$ax + by = c$$

Then we multiply our solution by q, to get:

$$q(ax + by) = cq = e$$

$$a(xq) + b(yq) = e$$

In general, if there exist one solution to this equation, then there must exist an infinite amount of solutions. This is generated by:

$$x' = x + t(b/c) \text{ and } y' = y - t(a/c)$$

To see that $ax'+by' = e$, substitute the general solution into our first equation:

$$a(x + t(b/c)) + b(y - t(a/c))$$

$$= ax + t(ab/c) + by - t(ab/c)$$

$$= ax + by$$

$$= e$$

References:
- Frank and Igor's CS490 Notes.
- Cormen, Thomas H., et al. Introduction to Algorithms.
- Rosen, Kenneth. Elementary Number Theory and its applications.
- http://en.wikipedia.org/wiki/Euclidean_algorithm
- http://en.wikipedia.org/wiki/Extended_Euclidean_algorithm