Cryptography & Encryption:6G7Z1011

Keith Yates

February 1, 2019

Cryptography & Encryption:6G7Z1011 : Euclid, and the Fast Powering Algorithm

Keith Yates

February 1, 2019

Speed of Computation

We have already coded algorithms of relevance to cryptography. For example:

1. determining if a number is or is not prime;
2. determining the greatest common denominator of two integers.

Both are of importance in cryptography. Given the integers that are used in real world cryptography can be hundreds of digits long there is an interest in its speed.

Estimates

When we talk of the 'time' taken to of an algorithm, we will measure it in bit operations, not real time. If all the computers became twice as fast over night we would not say the algorithms are running twice as fast, because what the algorithm did in terms of bit operations has not changed. Let us return to a problem we have meet before, and see if we can find a faster algorithm.

Euclidean algorithm

Let a, b be positive integers with $b \leq a$, there is a decomposition

$$a = qb + r \quad \text{with} \quad 0 \leq r < b. \tag{0.1}$$

That is b fits 'into' a q times and there is a remainder r that is always less than b.

1. Suppose d divided into both a and b, then eqn. 0.3 implies d must divide into r, which we write $d \mid r$
2. Suppose e divided into both r and b, then eqn. 0.3 implies it must divide into a.

Thus

$$\gcd(a, b) = \gcd(b, r). \tag{0.2}$$

Euclid's algorithm

Euclid realised that as

$$a = qb + r \quad \text{with} \quad 0 \leq r < b. \tag{0.3}$$

and

$$\gcd(a, b) = \gcd(b, r) \tag{0.4}$$

both held, then (as $r < b$) it would be easier to evaluate the right-hand side of eqn. 0.4, and being clever he also realised he could apply the same trick to (b, r) and write $b = rq' + r'$ with $0 \leq r' < r$ and

$$\gcd(b, r) = \gcd(r', r) \tag{0.5}$$

An example will clarify all this.

Proof

We have

$$a = bq + r \quad \text{where} \quad 0 \leq r < b. \tag{0.6}$$

1. Let α divides a and b then

$$\frac{r}{\alpha} = \frac{a - bq}{\alpha} \in \mathbb{N} \tag{0.7}$$

   so α divides r.

2. Let α divides b and r then

$$\frac{a}{\alpha} = \frac{bq + r}{\alpha} \in \mathbb{N} \tag{0.8}$$

Euclid

We have shown that if

$$a = bq + r \quad \text{where} \quad 0 \leq r < b. \tag{0.9}$$

then

$$\gcd(a, b) = \gcd(b, r). \tag{0.10}$$

This is clever: we have moved the problem to the same problem but with smaller numbers, we have $r < b < a$. And we can apply the same procedure to $\gcd(b, r)$ —note as the remainder $0 \leq r$; this process must stop after a finite number of steps.

Example

That was rather abstract, have a go apply (pen and paper calculation) to work out

$$\gcd(2024, 748) \tag{0.11}$$

Working out

$$
\begin{aligned}
2024 &= 748 \times 2 + 528 \\
748 &= 528 \times 1 + 220 \\
528 &= 220 \times 2 + 88 \\
220 &= 88 \times 2 + 44 \\
88 &= 44 \times 2 + 0
\end{aligned}
\tag{0.12}
$$

We have

$$
\begin{aligned}
&\gcd(2024, 748) = \gcd(748, 528) = \gcd(528, 220) = \\
&\gcd(220, 88) = \gcd(88, 44)
\end{aligned}
\tag{0.13}
$$

The answer is 44; not we reached this in 5 iterations of the algorithm a large saving over 748 'checks' in the naive approach.

Coding Euclid's algorithm

The following is Euclid's algorithm in words; you should code it in Java in the lab session. Input a and b we seek $\gcd(a, b)$

1. Let $r_0 = a$, $r_1 = b$.
2. Set $i = 1$.
3. Divide r_{i-1} by r_i to get quotient q_1 and remainder r_{i+1}, that is

$$r_{i-1} = r_i q_i + r_{i+1} \quad \text{with} \quad 0 \leq r_{i+1} < r_i \tag{0.14}$$

4. If $r_{i+1} = 0$ then $r_i = \gcd(a, b)$ and quit algorithm
5. Set $i = i + 1$ and go to 3.

Note as r_i decreases each time it must reach 0 after finitely many steps - so the algorithm does stop!

How Fast is Euclid's algorithm?

Euclid's algorithm for gcd is our first 'proper' algorithm. To complete its study we need to determine how fast it runs. Recall this requires us to create a graph plotting n the size of the smallest number against 'time' how long the algorithm takes to complete.

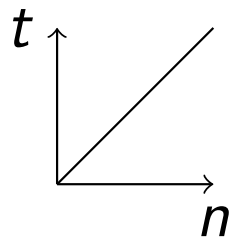

Figure: If we plot n (size of the smallest number argument to our gcd algorithm) against t time taken to run then for our naive gcd algorithm it will scale linearly; double the size of the number, then we double the checks to be performed. How does Euclid perform?

How fast is Euclid's algorithm?

Consider integers a, $b \in \mathbb{N}$ with $b \leq a$ then the algorithm loops at most

$$2\ln_2(b) + 1 \tag{0.15}$$

times $\ln_2$ is log to the base 2. This is some saving, consider our earlier example $\gcd(2024, 748)$ Then

$$2\ln_2(748) + 1 \approx 21. \tag{0.16}$$

Theoretical Analysis of Euclid's algorithm

Recall Euclid's algorithm generates a sequence of remainder

$$0 = r_n < r_{n-1} < .. < r_3 < r_2 < r_1. \tag{0.17}$$

At each stage the sequence decreases, and terminates at zero, the key issue is how fast does it decrease? We claim

$$r_{i+2} < \frac{r_i}{2}. \tag{0.18}$$

So after 5 iterations the remainder has got smaller by a factor of $\frac{1}{2^5}$.

Speed of Euclid's Algorithm

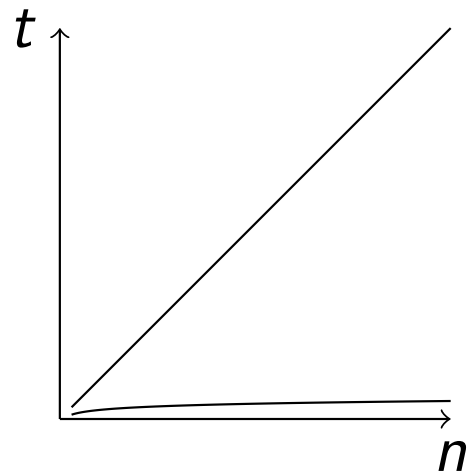

Figure: How does Euclid perform?The bottom line is $t = \log_2(n)$, the straight line $t = n$.

Proof

We need to prove $r_{i+2} < \frac{r_i}{2}$ holds for all i. There are two cases:

1. $r_{i+1} \leq \frac{r_i}{2}$: Then $r_{i+2} < r_{i+1} < \frac{r_i}{2}$.
2. $r_{i+1} > \frac{r_i}{2}$: r_{i+1} is so large that we have $r_i = 1.r_{i+1} + r_{i+2}$. Thus

$$r_{i+2} = r_i - r_{i+1} = r_i - \frac{r_i}{2} = \frac{r_i}{2} \tag{0.19}$$

We have shown $r_{i+2} < \frac{r_i}{2}$. As such

$$r_{2k+1} < \frac{r_{2k-1}}{2} < \frac{r_{2k-3}}{2^2} < \frac{r_{2k-5}}{2^3} < \ldots \frac{r_1}{2^k} = \frac{b}{2^k} \tag{0.20}$$

Thus if $b \leq 2^k$ then $r_{2k+1} < 1$ so it must be zero; and

$$\text{iterations required } \leq 2k = 2(k-1)+2 < 2\ln_2(b)+2. \tag{0.21}$$

End of the Story?

Given the importance of Euclid's algorithm it has been studied extensively and there are improvements (they are non-examinable but if you are interested they are discussed in [**?**]); the best to date is

$$t = 0.85 \ln_2(n) + 0.14 \tag{0.22}$$

The fast powering algorithm

In encryption algorithms we often need to compute x where

$$x = a^n \mod m \tag{0.23}$$

where a, n and m may be very large numbers. A naive approach would be

$$\begin{aligned} g_1 &= a \mod m \\ g_2 &= ag_1 \mod m \\ g_3 &= ag_2 \mod m \\ g_4 &= ag_3 \mod m \\ &. \end{aligned} \tag{0.24}$$

then $g_n = a^n \mod n$. Is it feasible?

How much time have you got?

Consider $x = a^n \mod m$, let a, m be large and let $n = 2^{1000}$. Consider a computer chip with a clock speed of 2^{20} (faster than any computer in the world) and let it process 2^{20} iterations in a second.

$$\text{time to complete all iterations} = \frac{2^{1000}}{2^{20}} > \text{ life time of universe.} \tag{0.25}$$

example

We — obviously — need a way of evaluating powers that is quicker than the above estimate. By way of an example let us compute

$$x = 3^{218} \mod 1000. \tag{0.26}$$

Any thoughts on how to do this?

Fast Powering Algorithm $x = 3^{218} \mod 1000$

First express 218 as the sum of powers of two

$$218 = 2 + 2^3 + 2^4 + 2^6 + 2^7 \tag{0.27}$$

Then

$$3^{218} = 3^{2+2^3+2^4+2^6+2^7} = 3^2.3^{2^3}.3^{2^4}.3^{2^6}.3^{2^7}. \tag{0.28}$$

We have

i	0	1	2	3	4	5	6	7
$3^{2^i} \mod 1000$	3	9	81	561	721	841	281	961

$$
\begin{aligned}
3^{2^6} &= 3^{2^5}.3^{2^5} \\
&= 3^{2^4}3^{2^4}3^{2^4}3^{2^4} \\
&= (43036.1000 + 721).(42046.1000 + 721) \\
&(43036.1000 + 721).(42046.1000 + 721) \mod 1000 \\
&= (721.721).(721.721) \\
&= 841.841 \mod 1000 \\
&= 281 \mod 1000
\end{aligned}
\tag{0.29}
$$

A bit more ...

We deduce

$$\begin{aligned} 3^{218} &= 3^2.3^{2^3}.3^{2^4}.3^{2^6}.3^{2^7} \\ &= 9.(6.1000+561).(1000.43046+721) \\ &(281).961 \mod 1000 \\ &= 480 \mod 1000. \end{aligned} \tag{0.30}$$

$$\begin{aligned} 3^{218} &= 3^2.3^{2^3}.3^{2^4}.3^{2^6}.3^{2^7} \\ &= 9.561.721.281.961 \mod 1000 \\ &= 480 \mod 1000. \end{aligned} \tag{0.31}$$

All done in two slides!

Extended Euclidean Algorithm

We return now to solution of equations. Let $a, b \in \mathbb{N}$ with $\gcd(a, b)$. Then there exists integers m and n such that

$$\gcd(a, b) = am + bn. \tag{0.32}$$

To see this consider the set

$$S = \{\alpha a + \beta b \mid \alpha, \beta \in \mathbb{Z}\}. \tag{0.33}$$

This set S has a smallest positive element, we claim it is $\gcd(a, b)$.

Proof of Claim

Let d denote the smallest number greater than or equal to zero in the set $S = \{\alpha a + \beta b \mid \alpha, \beta \in \mathbb{Z}\}$; say $d = \alpha_1 a + \beta_1 b$. We have two tasks:

1. Proving d divides a and b;
2. Any number dividing a and b is smaller than or equal to d (so $d = \gcd(a, b)$).

Proof

1. Using Euclid we have $a = dq + r$ where $0 \leq r < d$ then

$$r = a - dq = a - (\alpha_1 a + \beta_1 b)q = a(1 - \alpha_1 q) - (\beta_1 q)b \in S \tag{0.34}$$

but $0 \leq r < d$ and d is the smallest element in S; thus $r = 0$ and $d \mid a$. Similar reasoning gives $d \mid b$.

2. Finally let γ be any number dividing both a and b then as $d = \alpha_1 a + \beta_1 b$ we deduce $\gamma \mid d$; thus d is the greatest common denominator.

important property of the primes

Let p be prime and a, $b \in \mathbb{N}$ such that $p \mid ab$; then $p \mid a$ or $p \mid b$.
To prove this suppose $p \nmid a$ we show $p \mid b$. We have $\gcd(p, a) = 1$ so we have u and v such that

$$up + va = 1 \quad \text{thus} \quad upb + vab = b; \tag{0.35}$$

so as $p \mid b$ we deduce $p \mid ab$.
Note this result does not hold when p is not prime for example 15 divides $210 = 6 \times 35$ but 15 does not divide into 6 nor into 35.

Important Result

Recall, if $a, b \in \mathbb{N}$ with a greatest common denominator of $\gcd(a, b)$ then there exists integers m and n such that

$$\gcd(a, b) = am + bn. \tag{0.36}$$

This implies something of interest to us.

Existence of an inverse

Let a denote an integer then there is a solution to

$$ab = 1 \mod m \tag{0.37}$$

if and only if $\gcd(a, m) = 1$.

Proof.

- $\Rightarrow$: Suppose $\gcd(a, m) = 1$ then we have integers u and v such that $au + mv = 1$, and we deduce

$$au - 1 = mv, \quad \text{thus} \quad au = 1 \mod m. \tag{0.38}$$

- $\Leftarrow$: Suppose $ab = 1 \mod m$ then

$$ab - \alpha m = 1 \tag{0.39}$$

and any number d that divides both a and m must divide 1, which means $d = 1$.

Returning to our toy examples $\mathbb{Z}(5)$ and $\mathbb{Z}(6)$, then

1. $\mathbb{Z}(5)$: as $\gcd(a, 5) = 1$ for all $1 \leq a < 5$ (5 is prime) then every non-zero element of $\mathbb{Z}(5)$ has an inverse.
2. $\mathbb{Z}(6)$ $\gcd(3, 6) = 3 \neq 1$ so (for example) 3 has no inverse. Explicitly:
   a) $3.1 = 3 \neq 1$,
   b) $3.2 = 0 \neq 1$,
   c) $3.3 = 3 \neq 1$,
   d) $3.4 = 0 \neq 1$,
   e) $3.5 = 3 \neq 1$.

Introducing S_n

If I have time I will introduce S_n, the permutation group on n objects.