The extended Euclidean Algorithm

The Euclidean algorithm to find \( d = \gcd(a, b) \), \( a > b \), computes a sequence of remainders \( \{r_j\} \) by

\[
a = q_1b + r_1
\]
\[
b = q_2r_1 + r_2
\]
\[\vdots\]
\[
r_{k-2} = q_{k-1}r_{k-1} + r_k
\]
\[
r_{k-1} = q_{k+1}r_k
\]

We can also write the last line as \( r_{k-1} = q_{k+1}r_k + r_{k+1} \) where \( r_{k+1} = 0 \). Then \( r_k = \gcd(a, b) \).

This works because of the following; since \( d = \gcd(a, b) \) then from the first step since \( d|a \) and \( d|b \), then \( d|r_1 \). From the second step since \( d|b \) and \( d|r_1 \), we have \( d|r_2 \) and by induction we have \( d|r_j \) for every \( j \). In particular, this is true for the last nonzero remainder \( r_k \) so that \( d \leq r_k \). From the last step in the algorithm, since each \( q_j \) is an integer, we must have \( r_k|r_{k-1} \) and from the second last step that \( r_k|r_{k-2} \). Repeating, we have \( r_k|r_j \) for \( 1 \leq j \leq k \) and, in particular, \( r_k|r_1 \) and \( r_k|r_2 \). From the second step we see this means \( r_k|b \) and then from the first that \( r_k|a \). Thus \( r_k \) is a common divisor of \( a, b \) and so \( r_k \leq d \). Combining with the previous, we must have \( d = r_k \).

The key step in the construction process for the extended Euclidean algorithm is to show that if \( r_j \) is a remainder obtained during the Euclidean algorithm process, then there exists integers \( x^* \) and \( y^* \) such that \( ax^* + by^* = r_j \). We expect \( x^* \) and \( y^* \) to depend on \( j \) but we will clear this up later.

The initial step of the algorithm gives \( r_1 = a - q_1b \) and so we can set \( x^* = 1 \) and \( y^* = -q_1 \). The second step is \( r_2 = b - q_2r_1 = (-q_2)a + (1 + q_1q_2)b \). In general, we have \( r_j = r_{j-2} - q_jr_{j-1} \) and we see that if we have written \( r_i \) in terms of linear combinations of \( a \) and \( b \) for \( i < j \) then we can do so also for \( i = j \). In particular, we can do this for the final step with \( r_k \).

We now have to get the indices straight. Back to the first step; we are going to write \( x^* = x_2 \) and \( y^* = y_2 \) so this gives \( x_2 = 1 \) and \( y_2 = -q_1 \). Now \( r_2 = b - q_2r_1 = (-q_2)a + (1 + q_1q_2)b \) and the next step would set \( x_3 = -q_2 = -q_2x_2 \) and \( y_3 = 1 + q_1q_2 = 1 - q_2y_2 \). We write \( r_j = ax_{j+1} + by_{j+1} \) and the general step gives

\[
r_j = r_{j-2} - q_jr_{j-1} \quad ax_{j+1} + by_{j+1} = ax_{j-1} + by_{j-1} - q_j(ax_j + by_j)
\]

Thus the sequences \( \{x_n\} \) and \( \{y_n\} \) must satisfy the recursion scheme \( z_j = -q_{j-1}z_{j-1} + z_{j-2} \).

How do we start these schemes? We need two starting values for each sequence, namely \( x_0, x_1 \) and \( y_0, y_1 \). The key observation is the requirement that \( x_2 \) satisfy \( x_2 = 1 \) regardless of the numbers \( a \) and \( b \). But \( x_2 = -q_1x_1 + x_0 \) and the only way to do this is to select \( x_0 = 1 \) and \( x_1 = 0 \). Now \( x_3 = -q_2x_2 + x_1 = -q_2 \) and this is what we expected. For the \( y \)'s we need \( y_2 = -q_1 \) from \( y_2 = -q_1y_1 + y_0 \) for any value of \( q_1 \). The only way to satisfy this is by taking \( y_0 = 0 \) and \( y_1 = 1 \). Thus the Extended Euclidean algorithm becomes:

1. Compute the sequences \( r_j \) and \( q_j \), for \( j = 1, \ldots, k \) where \( k \) is that index yielding \( r_k = d = \gcd(a, b) \). This is the line in the Euclidean table immediately above the final one where the remainder is zero.
2. Set \( x_0 = 1, x_1 = 0, y_0 = 0, y_1 = 1 \) where \( x_j, y_j \) satisfy the recursion relation

\[
z_j = -q_{j-1}z_{j-1} + z_{j-2}.
\]

(\*)

3. Then, \( ax_{k+1} + by_{k+1} = r_k = d \).

Note: This assumes the first step was \( a = q_1b + r_1 \), that is we assumed that \( a > b \). The initial value of \( y \), \( y_0 = 0 \) corresponds to the lower of the values \( b \) since \( y \) is coupled with \( b \).
We know that the scheme must converge since each $q_i \geq 1$ and so the sequence $\{r_j\}$ is strictly decreasing and hence must terminate since the final value $d = r_k \geq 1$. An important question is in how many steps does this occur?

Based only on the decreasing of the remainders we get the very crude estimate that it terminates in at most $b$ steps, which, if the best available would render the algorithm useless for anything but relatively small numbers. We actually can do much better. In fact, it is easy to see that after any two iterations we must have cut the remainder down by at least one half. We now show this claim.

If $r_{i+1} \leq \frac{1}{2} r_i$ then since $\{r_j\}$ is decreasing, $r_{i+2} < r_{i+1} \leq \frac{1}{2} r_i$ and we have proved the claim. If instead, if $r_{i+1} > \frac{1}{2} r_i$ then from $r_i = q_i r_{i+1} + r_{i+2}$ and the fact that all quantities are nonnegative it follows that $r_i \geq 1, r_{i+1} + r_{i+2}$ or $r_{i+2} \leq r_i - r_{i+1} < r_i - \frac{1}{2} r_i = \frac{1}{2} r_i$, which again shows the claim. Using this repeatedly we obtain

\[ r_{2j+1} < \frac{1}{2} r_{2j-1} < \frac{1}{4} r_{2j-3} < \ldots < \frac{1}{2^j} r_1 < \frac{1}{2^j} b. \]

Now the final step is at $r_k$ with $r_{k+1} = 0$ and hence $k + 1 \leq 2j + 1$ or $k \leq 2j$, so that the algorithm terminates in at most $2j$ steps. If we choose the smallest $j$ with $2^j \geq b > 2^{j-1}$, that is with $j \geq \log_2(b) > j - 1$, then the number of iterations is at most $2j < 2(j - 1) + 1 < 2 \log_2(b) + 1$. We have thus proven

**Theorem 1.** The Euclidean algorithm terminates in at most $2 \log_2(b) + 1$ steps.

**Remark:** Can the worst case state of affairs actually occur and what does it lead to? At each stage we have a recursion scheme of the form $r_i = q_i r_{i+1} + r_{i+2}$ and it is clear that the difference $r_i - r_{i+2}$ will be least (that is the slowest decay) when $q_i$ is least, that is $q = 1$. Then in total we will have the slowest decay (and hence the most number of iterations when every $q_j = 1$. Note we already used this $q_i = 1$ for all $i$ as a lower bound in our proof above. In this case we then have the recursion scheme

\[ r_i = r_{i+1} + r_{i+2} \quad \text{for all } i = 0, 1, \ldots, t \]

**Example:** Compute $d = \gcd(1197, 833)$ and hence numbers $x$ and $y$ such that $ax + by = d$.

\[ 1197 = 1.833 + 364 \]
\[ 833 = 2.364 + 105 \]
\[ 364 = 3.105 + 49 \]
\[ 105 = 2.49 + 7 \]
\[ 49 = 7.7 + 0 \]

This shows that $d = 7$, $k = 4$, and we have $q_1 = 1, q_2 = 2, q_3 = 3, q_4 = 2, q_5 = 7$. Now we compute $x_j$ and $y_j$ from equation (*), with the final values being $x_5$ and $y_5$:

\[ x_0 = 1, \quad x_1 = 0 \]
\[ x_2 = -1.x_1 + x_0 = 1 \]
\[ x_3 = -2.x_2 + x_1 = -2 \]
\[ x_4 = -3.x_3 + x_2 = 7 \]
\[ x_5 = -2.x_4 + x_3 = -16 \]

\[ y_0 = 0, \quad y_1 = 1 \]
\[ y_2 = -1.y_1 + y_0 = -1 \]
\[ y_3 = -2.y_2 + y_1 = 3 \]
\[ y_4 = -3.y_3 + y_2 = -10 \]
\[ y_5 = -2.y_4 + y_3 = 23 \]

Thus $(-16).1197 + (23).833 = 7$
where \( r_t \) is the final step (so \( r_t = d = \gcd(a, b) \)) and \( r_{t+1} = 0 \). This \( \{r_i\} \) is a decreasing sequence with initial values \( r_0 = b \) and \( r_{-1} = a \). We can reverse the order, taking initial two values for the two term recursion of \( \tilde{r}_0 = d \) and \( \tilde{r}_{-1} = 0 \) and use the scheme

\[
\tilde{r}_{i+2} = \tilde{r}_{i+1} + \tilde{r}_i \quad \text{for all } i = -1, 1, \ldots, t
\]

This is the famous Fibonacci sequence and has a well worked out behaviour which we can now utilize. Let’s suppose for the moment \( d = 1 \) (so that the initial values are \( \{0, 1\} \)). This just makes the argument more straightforward but isn’t essential and in fact is also a worst case since the larger \( d \) is the least number of steps we require all else being the same.

We know that the terminal value is \( \tilde{r}_N = b \) and thus need a formula for the \( N^{th} \) Fibonacci number \( F_N \). This comes from the following lemma in which \( [x] \) is the nearest integer to \( x \).

**Lemma.** For the Fibonacci sequence \( F_{k+2} = F_{k+1} + F_k \), for any \( n \), after 5 steps of the recursion scheme we must have \( F_{n+5} \geq 10F_n \).

**Proof:** First observe that \( F_{k+2} = F_{k+1} + F_k = 2F_k + F_{k-1} \). Then for \( n \) sufficiently large

\[
F_{n+5} = F_{n+4} + F_{n+3} = (F_{n+3} + F_{n+2}) + F_{n+3} \\
= 2F_{n+3} + F_{n+2} = 2(F_{n+2} + F_{n+1}) + F_{n+2} \\
= 3F_{n+2} + 2F_{n+1} = 3(F_{n+1} + F_n) + 2F_{n+1} \\
= 5F_{n+1} + 3F_n = 5(F_n + F_{n-1}) + 3F_n \\
= 8F_n + 5F_{n-1} = 8(F_{n-1} + F_{n-2}) + 5F_{n-1} \\
= 13F_{n-1} + 8F_{n-2} = 13(F_{n-2} + F_{n-3}) + 8F_{n-2} \\
= 21F_{n-2} + 13F_{n-3} \\
\geq 10(2F_{n-2} + F_{n-3}) = 10F_n
\]

Where in the last step we used \( F_{k+2} = 2F_k + F_{k-1} \) again but now with \( k = n - 2 \).

This shows that after 5 steps we have actually decreased the current value of the remainder by a factor of 10 - substantially better than the above estimate and means we have dropped at least a decimal place after each 5 steps. Since \( 5 \log_{10}(2) = 1.505 \) this gives the improved result

**Theorem 2.** The Euclidean algorithm terminates in at most \( 5 \log_{10}(b) + 1 \) or \( 1.505 \log(b) + 1 \) steps.

**Remark:** Note that even this is not quite sharp and in consequence the result can be further improved. The easy thing to do is replace the final value of 10 by 21/2 in the penultimate step and thus we find that after 5 steps we reduce the value by a factor of 10.5. Then 1.505 can be reduced to \( 5/\log_2(10.5) \approx 1.474 \). Again, we have a little room for improvement and (quite a bit) more work lets us reduce this value down to \( \approx 1.45 \). (see Knuth\(^1\)).

**Remark:** These estimates are all worst-case; we are guaranteed to take no more than the above. However, another question is what is the expected number of iterations for a random choice of \( a \) and \( b \)? Again this is widely researched and the question can be extended from asking the expected average to “what is the distribution of the number of iterations.” Again, see Knuth. Here is one estimate: the average number expected for random \( a \) and \( b \) with \( a > b \) is \( k \approx 0.85 \log_2(b) + 2 \). With decimal notation this translates to just under 3 steps for each decimal place.