

CSCI 162/284 Cryptography: Euclidean Algorithm for the GCD

The euclidean algorithm for the gcd of two elements is thought to be the oldest existing algorithm. It is recursive. We describe the main idea behind it as a theorem with proof, then the algorithm, and then an argument that the algorithm is correct.

1 Main Idea

Theorem: $\gcd(m, a) = \gcd(a, m \text{ rem } a)$

Proof: Let g be any common factor of m and a , and $r = m \text{ rem } a$, then $m = r + qa, q \in \mathbb{Z}$.

$$\begin{aligned} g|m, g|a \\ \Rightarrow m = q_1g, a = q_2g \text{ for some } q_1, q_2 \in \mathbb{Z} \\ \Rightarrow r = g(q_1 - qq_2) \\ \Rightarrow g|r, g|a \end{aligned}$$

Further, let h be any common factor of a and r , then:

$$\begin{aligned} h|a, h|r \\ \Rightarrow a = q_3h, r = q_4h \text{ for some } q_3, q_4 \in \mathbb{Z} \\ \Rightarrow m = h(q_4 + qq_3) \\ \Rightarrow h|r, h|m \end{aligned}$$

Thus common factors of a and r are exactly the common factors of m and a . Hence the greatest common factor of a and r is the greatest common factor of a and m .

Note here that any number is considered a factor of 0, so we do not consider the case $r = 0$ separately.

2 Algorithm

The euclidean algorithm is as follows:

Algorithm_gcd(m, a) / m > a */*

```

(X, Y) := (m, a) /* Initialize */
while (Y ≠ 0) (X, Y) := (Y, X rem Y) /* Recursion */
return(X)

```

Example Use the euclidean algorithm to determine $\gcd(575, 299)$.

$$\begin{aligned} (X, Y) &= (575, 299) \\ (X, Y) &= (299, 276) \\ (X, Y) &= (276, 23) \\ (X, Y) &= (23, 0) \\ &\text{return}(23) \end{aligned}$$

Example Use the euclidean algorithm to determine $\text{gcd}(4316, 1245)$.

$$\begin{aligned}(X, Y) &= (4316, 1245) \\(X, Y) &= (1245, 581) \\(X, Y) &= (581, 83) \\(X, Y) &= (83, 0) \\ &\text{return}(83)\end{aligned}$$

Example Use the euclidean algorithm to determine $\text{gcd}(2652, 8855)$.

$$\begin{aligned}(X, Y) &= (8855, 2652) \\(X, Y) &= (2652, 899) \\(X, Y) &= (899, 854) \\(X, Y) &= (854, 45) \\(X, Y) &= (45, 44) \\(X, Y) &= (44, 1) \\(X, Y) &= (1, 0) \\ &\text{return}(1)\end{aligned}$$

3 Correctness of Algorithm

In each recursion, $\text{gcd}(X, Y)$ stays the same while X and Y change. Further, at each step, we decrease both X and Y , and, because both are remainders, neither is ever negative. Hence the algorithm will end some time, in fact, in at most a steps. Finally, at the last but one recursion, because $X \text{ rem } Y$ is zero, X is a multiple of Y and hence $\text{gcd}(X, Y) = Y$. At the last recursion, $(X, Y) := (Y, 0)$ and the returned value X is the correct gcd (it is the value of Y from the previous recursion).

A more formal proof is as follows.

Theorem: $\text{Algorithm_gcd}(m, a)$ returns $\text{gcd}(m, a)$.

Proof: We denote the value of (X, Y) in the i^{th} iteration as (X_i, Y_i) . Then, $(X_i, Y_i) := (Y_{i-1}, X_{i-1} \text{ rem } Y_{i-1})$, and $(X_0, Y_0) = (m, a)$. Let the total number of iterations be N . That is, $Y_N = 0$, and $\text{Algorithm_gcd}(m, a)$ returns X_N .

Then, from the theorem proved earlier, if N is finite,

$$\text{gcd}(m, a) = \text{gcd}(X_0, Y_0) = \text{gcd}(X_1, Y_1) = \dots = \text{gcd}(X_{N-1}, Y_{N-1})$$

Further, $Y_{N-1} | X_{N-1}$, hence $\text{gcd}(X_{N-1}, Y_{N-1}) = Y_{N-1} = X_N$. Hence $\text{gcd}(m, a) = X_N$, which is what $\text{Algorithm_gcd}(m, a)$ returns.

Further, $0 < Y_i < Y_{i-1}$ unless $Y_i = 0$. Hence Y_i is strictly decreasing and non-negative. As the algorithm stops when $Y_i = 0$, and Y_i decreases by at least 1 at each step, $N \leq a = Y_0$. Hence N is finite.