The Euclidean algorithm

Haining Fan, School of Software, Tsinghua University

fhn@tsinghua.edu.cn

September 11, 2016

Outline

1. Course Outline
   - Basic terminology and concepts
   - Symmetric-key ciphers
   - Public-key ciphers
2. The Euclidean algorithm
   - The division algorithm, GCD & the Euclidean algorithm
   - The balanced Euclidean algorithm
   - The extended Euclidean algorithm

Outline

1. Course Outline
   - Basic terminology and concepts
   - Symmetric-key ciphers
   - Public-key ciphers
2. The Euclidean algorithm
   - The division algorithm, GCD & the Euclidean algorithm
   - The balanced Euclidean algorithm
   - The extended Euclidean algorithm

References

1. D. Stinson, "Cryptography: Theory and Practice", 1996.

2. D. E. Knuth, "TAOCP", vol. 2, Chapter 4 - Arithmetic.

3. A. Menezes, P. Oorschot, and S. Vanstone, "Handbook of Applied Cryptography", 1996. (free download at http://www.cacr.math.uwaterloo.ca/hac)

Prerequisites: Linear Algebra, Elementary Number Theory.

References

1. D. Stinson, “Cryptography: Theory and Practice”, 1996.

2. D. E. Knuth, “TAOCP”, vol. 2, Chapter 4 - Arithmetic.

3. A. Menezes, P. Oorschot, and S. Vanstone,
“Handbook of Applied Cryptography”, 1996.
(free download at http://www.cacr.math.uwaterloo.ca/hac)

Prerequisites: Linear Algebra, Elementary Number Theory.

Course topics

finite fields, Chinese remainder theorem (CRT)

multiplication algorithms, fast Fourier transform,

primality test, discrete logarithms, factorization of integers,

public key ciphers (RSA and elliptic curve cryptosystem),

digital signatures, Advanced Encryption Standard (AES).

Course topics

finite fields, Chinese remainder theorem (CRT)

multiplication algorithms, fast Fourier transform,

primality test, discrete logarithms, factorization of integers,

public key ciphers (RSA and elliptic curve cryptosystem),

digital signatures, Advanced Encryption Standard (AES).

Aims of the course

After this course, you should

1. be aware of basic cryptographic concepts and methods.
2. be familiar with $GF(p^n)$s;

Intel introduced the PCLMULQDQ (Carry-less Multiplication) instruction in 2010.

Acknowledge

Some material is *copied* from various sources.

Aims of the course

After this course, you should

1. be aware of basic cryptographic concepts and methods.
2. be familiar with $GF(p^n)$s;

Intel introduced the PCLMULQDQ (Carry-less Multiplication) instruction in 2010.

Acknowledge

Some material is *copied* from various sources.

1. Course Outline
   - Basic terminology and concepts
   - Symmetric-key ciphers
   - Public-key ciphers

2. The Euclidean algorithm
   - The division algorithm, GCD & the Euclidean algorithm
   - The balanced Euclidean algorithm
   - The extended Euclidean algorithm

Cryptography is the study of mathematical techniques related to aspects of information security such as

1. Privacy (keeping secret data secret).
2. Data Integrity (preventing alteration).
3. (Message or Entity) Authentication (preventing frauds).
4. Non-repudiation (preventing denials of messages sent).

Cryptography is the study of mathematical techniques related to aspects of information security such as

1. Privacy (keeping secret data secret).
2. Data Integrity (preventing alteration).
3. (Message or Entity) Authentication (preventing frauds).
4. Non-repudiation (preventing denials of messages sent).

Cryptography is the study of mathematical techniques related to aspects of information security such as

1. Privacy (keeping secret data secret).
2. Data Integrity (preventing alteration).
3. (Message or Entity) Authentication (preventing frauds).
4. Non-repudiation (preventing denials of messages sent).

Cryptography is the study of mathematical techniques related to aspects of information security such as

1. Privacy (keeping secret data secret).
2. Data Integrity (preventing alteration).
3. (Message or Entity) Authentication (preventing frauds).
4. Non-repudiation (preventing denials of messages sent).

Modern Cryptography: a computational science

Security of a practical system must rely not on the impossibility but on the computational difficulty of breaking the system.

Rather than: "It is impossible to break the scheme".

We might be able to say: "No attack using $\leq 2^{160}$ time succeeds with probability $\geq 2^{-20}$".

Cryptography is not just mathematics; it needs to draw on computer science

1. Computational complexity theory;
2. Algorithm design.

Modern Cryptography: a computational science

Security of a practical system must rely not on the impossibility but on the computational difficulty of breaking the system.

Rather than: “It is impossible to break the scheme”.

We might be able to say: “No attack using $\leq 2^{160}$ time succeeds with probability $\geq 2^{-20}$”.

Cryptography is not just mathematics; it needs to draw on computer science

1. Computational complexity theory;
2. Algorithm design.

An encryption scheme

$m \in M$ is a plaintext of the message space M.
$c \in C$ is a ciphertext of the ciphertext space C.
$k \in K$ is a key of the key space K.

Encryption function E_e

$\forall e \in K$ uniquely determines a bijection $E_e : M \to C$.

Decryption function D_d

The corresponding $d \in K$ determines a bijection $D_d : C \to M$.

$$\text{A cipher} := E_e + D_d \quad \text{s.t.} \quad D_d(E_e(m)) = m.$$

An encryption scheme

$m \in M$ is a plaintext of the message space M.
$c \in C$ is a ciphertext of the ciphertext space C.
$k \in K$ is a key of the key space K.

Encryption function E_e

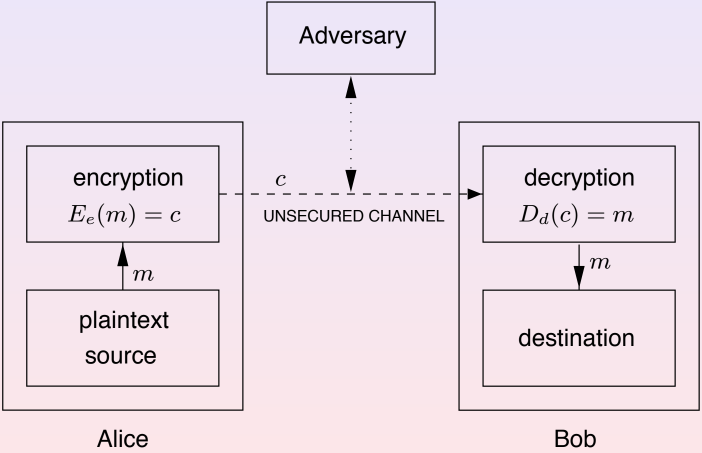
$\forall e \in K$ uniquely determines a bijection $E_e : M \to C$.

Decryption function D_d

The corresponding $d \in K$ determines a bijection $D_d : C \to M$.

$$\text{A cipher} := E_e + D_d \quad \text{s.t.} \quad D_d(E_e(m)) = m.$$

A simple model using encryption



1. Course Outline
   - Basic terminology and concepts
   - Symmetric-key ciphers
   - Public-key ciphers
2. The Euclidean algorithm
   - The division algorithm, GCD & the Euclidean algorithm
   - The balanced Euclidean algorithm
   - The extended Euclidean algorithm

Definitions

Definition (Symmetric-key & Asymmetric-key ciphers)

A cipher $E_e + D_d$ is said to be $\frac{symmetric-key}{asymmetric-key}$ if it is computationally $\frac{easy}{hard}$ to determine d knowing only e.

Example (Caesar Cipher)

$\forall x \in \mathbb{Z}_{26}$,
$E_3(x) := (x+3) \bmod 26; \quad D_{-3}(x) := (x+(-3)) \bmod 26;$

Notes:

1. $e = d$ in most practical symmetric-key ciphers,
2. Other terms: single-key, one-key, privatekey ...

The Kerckhoffs' principle (1883)

Definition (Shift Cipher)

$\forall x \in \mathbb{Z}_{26}$,

$E_k(x) := (x + k) \bmod 26; \quad D_k(x) := (x + (-k)) \bmod 26;$

Its secrecy is in the algorithm only, not the key.

In cryptography, Kerckhoffs' principle was stated by A. Kerckhoffs in the 19th century:

A cryptosystem should be secure even if everything about the system, except the key, is public knowledge.

The Kerckhoffs' principle (1883)

Definition (Shift Cipher)

$\forall x \in \mathbb{Z}_{26}$,
$E_k(x) := (x + k) \bmod 26; \quad D_k(x) := (x + (-k)) \bmod 26;$

Its secrecy is in the algorithm only, not the key.

In cryptography, Kerckhoffs' principle was stated by A. Kerckhoffs in the 19th century:

A cryptosystem should be secure even if everything about the system, except the key, is public knowledge.

One-Time Pad (Mauborgne & Vernam, 1917)

Definition (One-Time Pad)

Given $m_1, m_2, \cdots, m_t$, where $m_i \in \{0, 1\}$,
$c_1, c_2, \cdots, c_t$ is obtained by $c_i := m_i \text{ XOR } k_i$,
where $k_i \in \{0, 1\}$ is randomly chosen and never used again.

Theorem (Shannon)

One-Time Pad is a perfect encryption scheme.

A cipher is perfect only if its key space is at least the size of its message space.

One-Time Pad (Mauborgne & Vernam, 1917)

Definition (One-Time Pad)

Given $m_1, m_2, \cdots, m_t$, where $m_i \in \{0, 1\}$,
$c_1, c_2, \cdots, c_t$ is obtained by $c_i := m_i$ XOR k_i,
where $k_i \in \{0, 1\}$ is randomly chosen and never used again.

Theorem (Shannon)

One-Time Pad is a perfect encryption scheme.

A cipher is perfect only if its key space is at least the size of its message space.

One-Time Pad (Mauborgne & Vernam, 1917)

The one-time pad can be shown to be theoretically unbreakable.

It is totally secure if used correctly.

However it is very hard to use correctly.

Two problems

1. the key distribution and storage.
2. synchronization.

One-Time Pad (Mauborgne & Vernam, 1917)

The one-time pad can be shown to be theoretically unbreakable.

It is totally secure if used correctly.

However it is very hard to use correctly.

Two problems

1. the key distribution and storage.
2. synchronization.

One-time pads have applications in today's world, primarily for ultra-secure low-bandwidth channels.

1. The hotline between the USA and the former Soviet Union was *rumored* to be encrypted with a one-time pad.
2. Many spy messages to agents were encrypted using one-time pads.

One-time pads have applications in today's world, primarily for ultra-secure low-bandwidth channels.

1. The hotline between the USA and the former Soviet Union was *rumored* to be encrypted with a one-time pad.
2. Many spy messages to agents were encrypted using one-time pads.

1. Course Outline
   - Basic terminology and concepts
   - Symmetric-key ciphers
   - Public-key ciphers

2. The Euclidean algorithm
   - The division algorithm, GCD & the Euclidean algorithm
   - The balanced Euclidean algorithm
   - The extended Euclidean algorithm

Public-Key Cryptography (1976(?)-present)

In most symmetric ciphers, A and B share the same secret key $K_{A,B}$. $K_{A,B}$ must be secretly generated and exchanged prior to using the unsecure channel.

> Merkle , "Secure communications over insecure channels",
> *Communications of the ACM*, 1978
>
> Diffie and Hellman, "New Directions in Cryptography",
> *IEEE Transactions on Information Theory*, 1976

Split the B's secret key K into two parts:

K_E to be used for encryption. K_E can be made public.
K_D to be used for decryption.

Definition

A public key cipher is a cipher where everyone knows the enciphering transformation and everyone's enciphering key, but no known polynomial time algorithm will get deciphering keys from those.

Public-key cipher is rarely used for message exchange since it is slower than symmetric key ciphers.

Main uses of public-key cryptography

1. Key distributions for a symmetric cryptosystem.

2. Digital signatures.

Definition

A public key cipher is a cipher where everyone knows the enciphering transformation and everyone's enciphering key, but no known polynomial time algorithm will get deciphering keys from those.

Public-key cipher is rarely used for message exchange since it is slower than symmetric key ciphers.

Main uses of public-key cryptography

1. Key distributions for a symmetric cryptosystem.

2. Digital signatures.

Some LARGE Numbers

seconds in a year $3 * 10^7 \approx 2^{25}$
operations of 1 GHz CPU per year $3 * 10^{16} \approx 2^{55}$
atoms in the sun $10^{57} \approx 2^{190}$
atoms in the galaxy $10^{67} \approx 2^{223}$
atoms in the universe $10^{77} \approx 2^{265}$

NIST has recommended 5 $GF(2^n)$s and 5 $GF(p)$s for the elliptic curve digital signature algorithm (ECDSA):

$GF(2^{163})$, $GF(2^{233})$, $GF(2^{283})$, $GF(2^{409})$ and $GF(2^{571})$.

$GF(2^{192}-2^{64}-1)$, $GF(2^{224}-2^{96}+1)$,
$GF(2^{256}-2^{224}+2^{192}+2^{96}-1)$, $GF(2^{384}-2^{128}-2^{96}+2^{32}-1)$
and $GF(2^{521}-1)$.

Some LARGE Numbers

\# seconds in a year $3 * 10^7 \approx 2^{25}$

\# operations of 1 GHz CPU per year $3 * 10^{16} \approx 2^{55}$

\# atoms in the sun $10^{57} \approx 2^{190}$

\# atoms in the galaxy $10^{67} \approx 2^{223}$

\# atoms in the universe $10^{77} \approx 2^{265}$

NIST has recommended 5 $GF(2^n)$s and 5 $GF(p)$s for the elliptic curve digital signature algorithm (ECDSA):

$GF(2^{163})$, $GF(2^{233})$, $GF(2^{283})$, $GF(2^{409})$ and $GF(2^{571})$.

$GF(2^{192} - 2^{64} - 1)$, $GF(2^{224} - 2^{96} + 1)$,
$GF(2^{256} - 2^{224} + 2^{192} + 2^{96} - 1)$, $GF(2^{384} - 2^{128} - 2^{96} + 2^{32} - 1)$
and $GF(2^{521} - 1)$.

Divisibility and primality

Definition

A positive integer other than 1 is said to be a prime if its only divisors are 1 and itself.

An integer other than 1 is called composite if it is not prime.

Theorem (The division algorithm)

$\forall 0 < m, a \in \mathbb{Z}$, there exist unique integers q, r with $0 \leq r < |m|, \quad s.t. \quad a = mq + r$.

Notations: q: the quotient of the division.
$r = (a \bmod m)$: the remainder (or residue) of the division.
If $r = 0$ then m is called a divisor of a. We write $m|a$.

Divisibility and primality

Definition

A positive integer other than 1 is said to be a prime if its only divisors are 1 and itself.

An integer other than 1 is called composite if it is not prime.

Theorem (The division algorithm)

$\forall 0 < m, a \in \mathbb{Z}$, *there exist unique integers* q, r *with*
$0 \le r < |m|, \quad s.t. \quad a = mq + r.$

Notations: q: the quotient of the division.
$r = (a \bmod m)$: the remainder (or residue) of the division.
If $r = 0$ then m is called a divisor of a. We write $m|a$.

Definition

$d \in \mathbb{Z}$ is a common divisor of $a, b \in \mathbb{Z}$ if $d|a$ and $d|b$.
d is called the greatest common divisor (GCD) of a and b if it is the largest among the common divisors of a and b.

Theorem (The Euclidean theorem, 300 B.C.)

If $a = bq + r$ then $\gcd(a, b) = \gcd(b, r)$.

Proof:

$$\gcd(a,b)|r = a - bq \;\Rightarrow\; \gcd(a,b)|\gcd(b,r).$$
$$\gcd(b,r)|a = bq + r \;\Rightarrow\; \gcd(b,r)|\gcd(a,b).$$

Note: It is not necessary for q and r chosen in the above theorem to be the quotient and remainder obtained by dividing b into a, i.e., $0 \le r < |b|$. The theorem holds for any integers q and r satisfying the equality $a = bq + r$.

Definition

$d \in \mathbb{Z}$ is a common divisor of $a, b \in \mathbb{Z}$ if $d|a$ and $d|b$.
d is called the greatest common divisor (GCD) of a and b if it is the largest among the common divisors of a and b.

Theorem (The Euclidean theorem, 300 B.C.)

If $a = bq + r$ *then* $\gcd(a, b) = \gcd(b, r)$.

Proof:
$$\gcd(a,b)|r = a - bq \;\Rightarrow\; \gcd(a,b)|\gcd(b,r).$$
$$\gcd(b,r)|a = bq + r \;\Rightarrow\; \gcd(b,r)|\gcd(a,b).$$

Note: It is not necessary for q and r chosen in the above theorem to be the quotient and remainder obtained by dividing b into a, i.e., $0 \leq r < |b|$. The theorem holds for any integers q and r satisfying the equality $a = bq + r$.

Definition

$d \in \mathbb{Z}$ is a common divisor of $a, b \in \mathbb{Z}$ if $d|a$ and $d|b$.
d is called the greatest common divisor (GCD) of a and b if it is the largest among the common divisors of a and b.

Theorem (The Euclidean theorem, 300 B.C.)

If $a = bq + r$ *then* $\gcd(a, b) = \gcd(b, r)$.

Proof:

$$\gcd(a,b)|r = a - bq \;\Rightarrow\; \gcd(a,b)|\gcd(b,r).$$
$$\gcd(b,r)|a = bq + r \;\Rightarrow\; \gcd(b,r)|\gcd(a,b).$$

Note: It is not necessary for q and r chosen in the above theorem to be the quotient and remainder obtained by dividing b into a, i.e., $0 \leq r < |b|$. The theorem holds for any integers q and r satisfying the equality $a = bq + r$.

Example: $\gcd(a = 24, b = 15)$

The Euclidean algorithm: $\gcd(a, b)$

If $(b = 0)$ then return a; $\quad \gcd(a, 0) = a$;
Return $\gcd(b,\ a \bmod b)$;

It generates a sequence r_i $(-1 \le i \le t + 1)$ ends with $r_{t+1} = 0$:

$$r_{k-1} = q_{k+1} r_k + r_{k+1}$$

$r_{-1} := a = 24$

$r_0 := b = 15 \qquad \gcd(r_{-1}, r_0) = \gcd(a, b)$

$r_1 := r_{-1} \bmod r_0 = 9$

$r_2 := r_0 \bmod r_1 = 6 \qquad \gcd(r_{i-1}, r_i) = \gcd(r_i, r_{i+1})$

$r_3 := r_1 \bmod r_2 = 3$

$r_4 := r_2 \bmod r_3 = 0 \qquad \gcd(r_t, r_{t+1} = 0) = \gcd(a, b) = r_3 = 3$

The descending sequence r_i

$$r_{-1} = a \qquad r_0 = b$$

$$\begin{aligned}
r_{-1} &= q_1 r_0 + r_1 \\
r_0 &= q_2 r_1 + r_2 \\
&\cdots \\
r_{i-3} &= q_{i-1} r_{i-2} + r_{i-1} \\
r_{i-2} &= q_i r_{i-1} + r_i \\
r_{i-1} &= q_{i+1} r_i + r_{i+1} \\
&\cdots \\
r_{t-2} &= q_t r_{t-1} + r_t \\
r_{t-1} &= q_{t+1} r_t + 0
\end{aligned}$$

$$d = \gcd(a, b) = \gcd(r_{i-1}, r_i) = \gcd(r_i, r_{i+1}) = \gcd(r_t, r_{t+1} = 0)$$

$$\begin{array}{ccccccccccccc}
r_{-1} & > & r_0 & > & r_1 & \cdots & r_{i-1} & > & r_i & \cdots & r_t = d & > & r_{t+1} = 0 \\
 & d & & d & & d & & d & & d & & d &
\end{array}$$

Euclid: If $a = bq + r$ then $\gcd(a, b) = \gcd(b, r)$.

A recursive Euclidean algorithm: $\gcd(a, b)$

INPUT: $a > 0$ and $b > 0$. **OUTPUT:** $\gcd(a, b)$.

If $(b = 0)$ then return a;
Return $\gcd(b,\ a \bmod b)$;

A nonrecursive Euclidean algorithm: $\gcd(a, b)$

```
While b ≠ 0 do        // (a, b) := (b, a mod b)
    t := b;
    b := a mod b;
    a := t;
Return a;
```

D. E. Knuth: "the oldest nontrivial algorithm that has survived to the present day."

The nonrecursive Euclidean algorithm: $\gcd(a, b)$

INPUT: $a > 0$ and $b > 0$. **OUTPUT:** $\gcd(a, b)$.

While $b \neq 0$ do // $(a,\ b) := (b,\ a \bmod b)$
- $t := b;$
- $b := a \bmod b;$
- $a := t;$

Return a;

Theorem (Dr. Finck - a French mathematician, 1841)

Suppose $0 < b$, $0 < a$ *and* $M = MAX(a, b)$. *The Euclidean algorithm will find* $\gcd(a, b)$ *after a cost of at most* $\lfloor 2\log_2 M \rfloor + 1$ *integer divisions*.

This is the first rigorous analysis of the Euclidean algorithm.

Another proof which makes use of the Fibonacci numbers can be found in "*Introduction to Algorithms*" by T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein.

An upper bound on # divisions in $\gcd(a, b)$

Lemma: If $0 < b \leq a$ then $(a \bmod b) \leq \frac{(a-1)}{2}$.

0 — — -b — — $\frac{a}{2}$ — — — — -a

0 — — — — -$\frac{a}{2}$ — -b — — -a

Proof.

Clearly $(a \bmod b) \leq b - 1$.
Further, $(a \bmod b) = a - \lfloor \frac{a}{b} \rfloor b \leq a - b$
Thus $(a \bmod b) \leq MIN(b - 1, a - b)$.
Now we distinguish two cases.

1. $b - 1 \leq a - b$, i.e., $b \leq \frac{(a+1)}{2}$, we have $(a \bmod b) \leq b - 1 \leq \frac{(a+1)}{2} - 1 = \frac{(a-1)}{2}$.

An upper bound on # divisions in $\gcd(a, b)$

Lemma: If $0 < b \le a$ then $(a \bmod b) \le \frac{(a-1)}{2}$.

0 — — -b — — $\frac{a}{2}$ — — — — -a 0 — — — — -$\frac{a}{2}$ — -b — — -a

Proof.

Clearly $(a \bmod b) \le b - 1$.
Further, $(a \bmod b) = a - \lfloor \frac{a}{b} \rfloor b \le a - b$
Thus $(a \bmod b) \le MIN(b-1, a-b)$.
Now we distinguish two cases.

1. $b - 1 \le a - b$, i.e., $b \le \frac{(a+1)}{2}$, we have $(a \bmod b) \le b - 1 \le \frac{(a+1)}{2} - 1 = \frac{(a-1)}{2}$.
2. $b - 1 > a - b$, i.e., $b > \frac{(a+1)}{2}$, we have $(a \bmod b) \le a - b < a - \frac{(a+1)}{2} = \frac{(a-1)}{2}$.

An upper bound on # divisions in $\gcd(a, b)$

Lemma: If $0 < b \leq a$ then $(a \bmod b) \leq \frac{(a-1)}{2}$.

$0 \text{---} b \text{---} \frac{a}{2} \text{-------} a$ $\qquad$ $0 \text{-------} \frac{a}{2} \text{---} b \text{---} a$

Proof.

Clearly $(a \bmod b) \leq b - 1$.
Further, $(a \bmod b) = a - \lfloor \frac{a}{b} \rfloor b \leq a - b$
Thus $(a \bmod b) \leq MIN(b - 1, a - b)$.
Now we distinguish two cases.

1. $b - 1 \leq a - b$, i.e., $b \leq \frac{(a+1)}{2}$, we have $(a \bmod b) \leq b - 1 \leq \frac{(a+1)}{2} - 1 = \frac{(a-1)}{2}$.
2. $b - 1 > a - b$, i.e., $b > \frac{(a+1)}{2}$, we have $(a \bmod b) \leq a - b < a - \frac{(a+1)}{2} = \frac{(a-1)}{2}$.

An upper bound on # divisions in $\gcd(a, b)$

Lemma: If $0 < b \le a$ then $(a \bmod b) \le \frac{(a-1)}{2}$.

0— —-b— —$\frac{a}{2}$— — — —-a $\qquad$ 0— — — — -$\frac{a}{2}$— -b— — -a

Proof.

Clearly $(a \bmod b) \le b - 1$.
Further, $(a \bmod b) = a - \lfloor \frac{a}{b} \rfloor b \le a - b$
Thus $(a \bmod b) \le MIN(b - 1, a - b)$.
Now we distinguish two cases.

1. $b - 1 \le a - b$, i.e., $b \le \frac{(a+1)}{2}$, we have $(a \bmod b) \le b - 1 \le \frac{(a+1)}{2} - 1 = \frac{(a-1)}{2}$.
2. $b - 1 > a - b$, i.e., $b > \frac{(a+1)}{2}$, we have $(a \bmod b) \le a - b < a - \frac{(a+1)}{2} = \frac{(a-1)}{2}$.

An upper bound on # divisions in $\gcd(a, b)$

Lemma: If $0 < b \le a$ then $(a \bmod b) \le \frac{(a-1)}{2}$.

Proof.

Suppose that $a \ge b$.
The algorithm generates a sequence $r_{-1}, r_0, r_1, \ldots$, where
$r_{-1} = a, \;\; r_0 = b, \;\; r_1 = r_{-1} \bmod r_0, \;\; r_2 = r_0 \bmod r_1$
and $r_{j+1} = r_{j-1} \bmod r_j \;\; (j \ge 1)$.

By the above lemma, $r_{j+1} \le \frac{r_{j-1}-1}{2} < \frac{r_{j-1}}{2}$.

Then, by induction on $j(\ge 0)$ it follows that either

$$r_{2j} < \frac{r_0}{2^j} \text{ or } r_{2j+1} < \frac{r_1}{2^j} \iff r_s < \frac{M}{2^{\lfloor s/2 \rfloor}}.$$

The algorithm has terminated if $r_s < 1$, i.e., $s > 2\log_2 M$.

If $a < b$ then $\gcd(a, b) = \gcd(b, a)$

An upper bound on # divisions in $\gcd(a, b)$

Lemma: If $0 < b \leq a$ then $(a \bmod b) \leq \frac{(a-1)}{2}$.

Proof.

Suppose that $a \geq b$.
The algorithm generates a sequence $r_{-1}, r_0, r_1, \ldots$, where
$r_{-1} = a, \quad r_0 = b, \quad r_1 = r_{-1} \bmod r_0, \quad r_2 = r_0 \bmod r_1$
and $r_{j+1} = r_{j-1} \bmod r_j \quad (j \geq 1)$.

By the above lemma, $r_{j+1} \leq \frac{r_{j-1}-1}{2} < \frac{r_{j-1}}{2}$.

Then, by induction on $j (\geq 0)$ it follows that either

$$r_{2j} < \frac{r_0}{2^j} \text{ or } r_{2j+1} < \frac{r_1}{2^j} \iff r_s < \frac{M}{2^{\lfloor s/2 \rfloor}}.$$

The algorithm has terminated if $r_s < 1$, i.e., $s > 2\log_2 M$.

If $a < b$ then $\gcd(a, b) = \gcd(b, a)$

An upper bound on # divisions in $\gcd(a, b)$

Lemma: If $0 < b \leq a$ then $(a \bmod b) \leq \frac{(a-1)}{2}$.

Proof.

Suppose that $a \geq b$.
The algorithm generates a sequence $r_{-1}, r_0, r_1, \ldots,$ where
$r_{-1} = a, \quad r_0 = b, \quad r_1 = r_{-1} \bmod r_0, \quad r_2 = r_0 \bmod r_1$
and $r_{j+1} = r_{j-1} \bmod r_j \quad (j \geq 1)$.

By the above lemma, $r_{j+1} \leq \frac{r_{j-1}-1}{2} < \frac{r_{j-1}}{2}$.

Then, by induction on $j(\geq 0)$ it follows that either

$$r_{2j} < \frac{r_0}{2^j} \text{ or } r_{2j+1} < \frac{r_1}{2^j} \iff r_s < \frac{M}{2^{\lfloor s/2 \rfloor}}.$$

The algorithm has terminated if $r_s < 1$, i.e., $s > 2\log_2 M$.

If $a < b$ then $\gcd(a, b) = \gcd(b, a)$

An upper bound on # divisions in $\gcd(a, b)$

Lemma: If $0 < b \le a$ then $(a \bmod b) \le \frac{(a-1)}{2}$.

Proof.

Suppose that $a \ge b$.
The algorithm generates a sequence $r_{-1}, r_0, r_1, \ldots$, where
$r_{-1} = a, \quad r_0 = b, \quad r_1 = r_{-1} \bmod r_0, \quad r_2 = r_0 \bmod r_1$
and $r_{j+1} = r_{j-1} \bmod r_j \quad (j \ge 1)$.

By the above lemma, $r_{j+1} \le \frac{r_{j-1} - 1}{2} < \frac{r_{j-1}}{2}$.

Then, by induction on $j (\ge 0)$ it follows that either

$$r_{2j} < \frac{r_0}{2^j} \text{ or } r_{2j+1} < \frac{r_1}{2^j} \iff r_s < \frac{M}{2^{\lfloor s/2 \rfloor}}.$$

The algorithm has terminated if $r_s < 1$, i.e., $s > 2 \log_2 M$.

If $a < b$ then $\gcd(a, b) \rightleftharpoons \gcd(b, a)$

An upper bound on # divisions in $\gcd(a, b)$

Lemma: If $0 < b \leq a$ then $(a \bmod b) \leq \frac{(a-1)}{2}$.

Proof.

Suppose that $a \geq b$.
The algorithm generates a sequence $r_{-1}, r_0, r_1, \ldots$, where
$r_{-1} = a, \quad r_0 = b, \quad r_1 = r_{-1} \bmod r_0, \quad r_2 = r_0 \bmod r_1$
and $r_{j+1} = r_{j-1} \bmod r_j \quad (j \geq 1)$.

By the above lemma, $r_{j+1} \leq \frac{r_{j-1}-1}{2} < \frac{r_{j-1}}{2}$.

Then, by induction on $j(\geq 0)$ it follows that either

$$r_{2j} < \frac{r_0}{2^j} \text{ or } r_{2j+1} < \frac{r_1}{2^j} \iff r_s < \frac{M}{2^{\lfloor s/2 \rfloor}}.$$

The algorithm has terminated if $r_s < 1$, i.e., $s > 2\log_2 M$.

If $a < b$ then $\gcd(a, b) \rightleftharpoons \gcd(b, a)$

The binary Euclidean algorithm

INPUT: two binary positive integers a and b.
OUTPUT: $\gcd(a, b)$.

$i := 0;$
while $(a \bmod 2 == b \bmod 2 == 0)$ do
  $(i, \ a, \ b) := (i+1, \ a/2, \ b/2);$
while $(a \bmod 2 == 0)$ do $a := a/2;$
while $(b \bmod 2 == 0)$ do $b := b/2;$
while $(a \neq b)$ do
  $(a, \ b) := (|a-b|, \ min(a, b));$
  repeat $a := a/2$ until $(a \bmod 2 \neq 0);$
return $a2^i;$

It is more efficient on average than the original Euclidean algorithm.

The 1st generalized division algorithm

Theorem (The division algorithm)

$\forall 0 < m, a \in \mathbb{Z}$, *there exist unique integers* q, r *with* $0 \leq r < |m|, \quad s.t. \quad a = mq + r.$

$$—\,—\frac{-m}{2}—\,—\,—\,—0—\,—\,—\,—\frac{m}{2}—\,—\,—\,—m—$$

Theorem (The 1st generalized division algorithm)

$\forall 0 < m, a, d \in \mathbb{Z}$, *there exist unique integers* Q, R *with* $d \leq R < |m| + d, \quad s.t. \quad a = mQ + R.$

Especially, if $d = -\left|\frac{m}{2}\right|$ then $-\left|\frac{m}{2}\right| \leq R < \left|\frac{m}{2}\right|$.

$$—\,—\frac{-m}{2}—\,—\,—\,—0—\,—\,—\,—\frac{m}{2}—\,—\,—\,—m—$$

The 1st generalized division algorithm

Theorem (The division algorithm)

$\forall 0 < m, a \in \mathbb{Z}$, there exist unique integers q, r with $0 \le r < |m|$, s.t. $a = mq + r$.

$$-- \frac{-m}{2} ---- 0 ---- \frac{m}{2} ---- m -$$

Theorem (The 1st generalized division algorithm)

$\forall 0 < m, a, d \in \mathbb{Z}$, there exist unique integers Q, R with $d \le R < |m| + d$, s.t. $a = mQ + R$.

Especially, if $d = -\left|\frac{m}{2}\right|$ then $-\left|\frac{m}{2}\right| \le R < \left|\frac{m}{2}\right|$.

$$-- \frac{-m}{2} ---- 0 ---- \frac{m}{2} ---- m -$$

Theorem (The 1st generalized Euclidean theorem)

If $a = bq + r$ *then*

$$\gcd(a,b) = \gcd(b,r) = \gcd(b,-r) = \gcd(b,b-r).$$

Example (Comparisons of the two methods to compute $\gcd(114,34) = 2$)

114=3*34+12	114=3*34+12
34=2*12 +10	34=3*12 -2
12=1*10+2	12=6*2
10=5*2	

Theorem (The 1st generalized Euclidean theorem)

If $a = bq + r$ *then*

$$\gcd(a,b) = \gcd(b,r) = \gcd(b,-r) = \gcd(b,b-r).$$

Example (Comparisons of the two methods to compute $\gcd(114,34) = 2$)

114=3*34+12	114=3*34+12
34=2*12 +10	34=3*12 -2
12=1*10+2	12=6*2
10=5*2	

The balanced Euclidean algorithm: $B\_gcd(a, b)$

INPUT: $a > 0$ and $b > 0$. **OUTPUT:** $\gcd(a, b)$.

If $(b = 0)$ then return a;
$r := a \bmod b$;
If $(2r > b)$ then $r := b - r$; $r \leq \frac{(b-1)}{2} < \frac{b}{2}$
Return $B\_gcd(b, r)$;

The Euclidean algorithm: $\gcd(a, b)$

If $(b = 0)$ then return a;
$r := a \bmod b$; $r \leq \frac{(a-1)}{2} < \frac{a}{2}$
Return $\gcd(b, r)$;

$B\_gcd(a, b)$: — —**0**— — — — $\frac{b}{2}$ — — — — -b— — — —a—

$\gcd(a, b)$: — —**0**— — — — $\frac{b}{2}$— — — —b— — — —a—

An upper bound on # divisions in $B\_gcd(a, b)$

Theorem

Suppose $0 < b, 0 < a$ *and* $M = MAX(a, b)$. *The balanced Euclidean algorithm will find* $\gcd(a, b)$ *after a cost of at most* $\lfloor \log_2 M \rfloor + 1$ *integer divisions*.

Recall that this number is $\lfloor 2 \log_2 M \rfloor + 1$ in $\gcd(a, b)$.

This theorem does *not* mean that $B\_gcd(a, b)$ is faster than $\gcd(a, b)$ since it provides only the upper bound.

In fact, the number of integer divisions in $B\_gcd(a, b)$ is not greater than that in $\gcd(a, b)$.

An upper bound on # divisions in $B\_gcd(a, b)$

Theorem

Suppose $0 < b, 0 < a$ *and* $M = MAX(a, b)$. *The balanced Euclidean algorithm will find* $\gcd(a, b)$ *after a cost of at most* $\lfloor \log_2 M \rfloor + 1$ *integer divisions*.

Recall that this number is $\lfloor 2 \log_2 M \rfloor + 1$ in $\gcd(a, b)$.

This theorem does *not* mean that $B\_gcd(a, b)$ is faster than $\gcd(a, b)$ since it provides only the upper bound.

In fact, the number of integer divisions in $B\_gcd(a, b)$ is not greater than that in $\gcd(a, b)$.

An upper bound on # divisions in $B\_gcd(a, b)$

Theorem

Suppose $0 < b, 0 < a$ *and* $M = MAX(a, b)$. *The balanced Euclidean algorithm will find* $\gcd(a, b)$ *after a cost of at most* $\lfloor \log_2 M \rfloor + 1$ *integer divisions*.

Recall that this number is $\lfloor 2 \log_2 M \rfloor + 1$ in $\gcd(a, b)$.

This theorem does *not* mean that $B\_gcd(a, b)$ is faster than $\gcd(a, b)$ since it provides only the upper bound.

In fact, the number of integer divisions in $B\_gcd(a, b)$ is not greater than that in $\gcd(a, b)$.

An upper bound on # divisions in $B\_gcd(a, b)$

Theorem

Suppose $0 < b, 0 < a$ *and* $M = MAX(a, b)$. *The balanced Euclidean algorithm will find* $\gcd(a, b)$ *after a cost of at most* $\lfloor \log_2 M \rfloor + 1$ *integer divisions*.

Proof.

Suppose that $a \geq b$.

The algorithm generates a sequence $r_{-1}, r_0, r_1, r_2, \ldots$, where $r_{-1} = a, r_0 = b$ and $r_{j+1} \leq \frac{r_{j-1}}{2} \quad (j \geq 0)$.

An upper bound on # divisions in $B\_gcd(a, b)$

Theorem

Suppose $0 < b$, $0 < a$ *and* $M = MAX(a, b)$. *The balanced Euclidean algorithm will find* $\gcd(a, b)$ *after a cost of at most* $\lfloor \log_2 M \rfloor + 1$ *integer divisions.*

Proof.

Suppose that $a \geq b$.

The algorithm generates a sequence $r_{-1}, r_0, r_1, r_2, \ldots$, where $r_{-1} = a, r_0 = b$ and $r_{j+1} \leq \frac{r_j - 1}{2} \quad (j \geq 0)$.

Similarly, we obtain $r_s < \frac{M}{2^s}$. $\quad$ ($r_{j+1} \leq \frac{r_{j-1}-1}{2}$ in $\gcd(a, b)$).

The algorithm has terminated if $r_s < 1$, i.e., $s > \log_2 M$.

If $a < b$ then $\gcd(a, b) \rightleftharpoons \gcd(b, a)$

An upper bound on # divisions in $B\_gcd(a,b)$

Theorem

Suppose $0 < b$, $0 < a$ *and* $M = MAX(a,b)$. *The balanced Euclidean algorithm will find* $\gcd(a,b)$ *after a cost of at most* $\lfloor \log_2 M \rfloor + 1$ *integer divisions.*

Proof.

Suppose that $a \geq b$.

The algorithm generates a sequence $r_{-1}, r_0, r_1, r_2, \ldots$, where $r_{-1} = a, r_0 = b$ and $r_{j+1} \leq \frac{r_j - 1}{2} \quad (j \geq 0)$.

Similarly, we obtain $r_s < \frac{M}{2^s}$. $\quad$ ($r_{j+1} \leq \frac{r_{j-1}-1}{2}$ in $\gcd(a,b)$).

The algorithm has terminated if $r_s < 1$, i.e., $s > \log_2 M$.

If $a < b$ then $\gcd(a,b) \rightleftharpoons \gcd(b,a)$

An upper bound on # divisions in $B\_gcd(a, b)$

Theorem

Suppose $0 < b$, $0 < a$ *and* $M = MAX(a, b)$. *The balanced Euclidean algorithm will find* $\gcd(a, b)$ *after a cost of at most* $\lfloor \log_2 M \rfloor + 1$ *integer divisions*.

Proof.

Suppose that $a \geq b$.

The algorithm generates a sequence $r_{-1}, r_0, r_1, r_2, \ldots$, where $r_{-1} = a, r_0 = b$ and $r_{j+1} \leq \frac{r_j - 1}{2} \quad (j \geq 0)$.

Similarly, we obtain $r_s < \frac{M}{2^s}$. $\quad$ ($r_{j+1} \leq \frac{r_{j-1} - 1}{2}$ in $\gcd(a, b)$).

The algorithm has terminated if $r_s < 1$, i.e., $s > \log_2 M$.

If $a < b$ then $\gcd(a, b) \rightleftharpoons \gcd(b, a)$

An upper bound on # divisions in $B\_gcd(a, b)$

Theorem

Suppose $0 < b$, $0 < a$ *and* $M = MAX(a, b)$. *The balanced Euclidean algorithm will find* $\gcd(a, b)$ *after a cost of at most* $\lfloor \log_2 M \rfloor + 1$ *integer divisions*.

Proof.

Suppose that $a \geq b$.

The algorithm generates a sequence $r_{-1}, r_0, r_1, r_2, \ldots$, where $r_{-1} = a, r_0 = b$ and $r_{j+1} \leq \frac{r_j - 1}{2} \quad (j \geq 0)$.

Similarly, we obtain $r_s < \frac{M}{2^s}$. $\quad$ ($r_{j+1} \leq \frac{r_{j-1} - 1}{2}$ in $\gcd(a, b)$).

The algorithm has terminated if $r_s < 1$, i.e., $s > \log_2 M$.

If $a < b$ then $\gcd(a, b) \rightleftharpoons \gcd(b, a)$

An upper bound on # divisions in $B\_gcd(a, b)$

Theorem

Suppose $0 < b, 0 < a$ *and* $M = MAX(a, b)$. *The balanced Euclidean algorithm will find* $\gcd(a, b)$ *after a cost of at most* $\lfloor \log_2 M \rfloor + 1$ *integer divisions*.

Proof.

Suppose that $a \geq b$.

The algorithm generates a sequence $r_{-1}, r_0, r_1, r_2, \ldots$, where $r_{-1} = a, r_0 = b$ and $r_{j+1} \leq \frac{r_j - 1}{2} \quad (j \geq 0)$.

Similarly, we obtain $r_s < \frac{M}{2^s}$. $\quad$ ($r_{j+1} \leq \frac{r_{j-1} - 1}{2}$ in $\gcd(a, b)$).

The algorithm has terminated if $r_s < 1$, i.e., $s > \log_2 M$.

If $a < b$ then $\gcd(a, b) \rightleftharpoons \gcd(b, a)$

Theorem (Bezout's identity)

$\forall a, b \in \mathbb{Z}, \exists u, v \in \mathbb{Z}\ s.t.\ \gcd(a, b) = ua + vb.$

Proof: Define $S = \{au + bv | u, v \in \mathbb{Z}\}$. We will prove:

$$ax + by = n \text{ is solvable in } \mathbb{Z} \Leftrightarrow n \in S \Leftrightarrow \gcd(a, b) | n.$$

The 1st $\Leftrightarrow$ and the 2nd $\Rightarrow$ are easy. Now we prove the 2nd $\Leftarrow$.

1. If $x, y \in S$ then $x \pm y \in S$.

$\because \exists u, v, s, t \in \mathbb{Z}$ s.t. $x = au + bv$ and $y = as + bt$
$\therefore x \pm y = a(u \pm s) + b(v \pm t) \in S.$

2. If $x \in S$ then $cx \in S$ for $\forall c \in \mathbb{Z}$.

$x = au + bv \in S \quad \Rightarrow \quad cx = a(cu) + b(cv) \in S.$

Theorem (Bezout's identity)

$\forall a, b \in \mathbb{Z}, \exists u, v \in \mathbb{Z}\ s.t.\ \gcd(a, b) = ua + vb.$

Proof: Define $S = \{au + bv | u, v \in \mathbb{Z}\}$. We will prove:

$$ax + by = n \text{ is solvable in } \mathbb{Z} \Leftrightarrow n \in S \Leftrightarrow \gcd(a, b) | n.$$

The 1st $\Leftrightarrow$ and the 2nd $\Rightarrow$ are easy. Now we prove the 2nd $\Leftarrow$.

1. If $x, y \in S$ then $x \pm y \in S$.

$\because \exists u, v, s, t \in \mathbb{Z}$ s.t. $x = au + bv$ and $y = as + bt$
$\therefore x \pm y = a(u \pm s) + b(v \pm t) \in S$.

2. If $x \in S$ then $cx \in S$ for $\forall c \in \mathbb{Z}$.

$x = au + bv \in S \quad \Rightarrow \quad cx = a(cu) + b(cv) \in S.$

Theorem (Bezout's identity)

$\forall a, b \in \mathbb{Z}, \exists u, v \in \mathbb{Z}\ s.t.\ \gcd(a, b) = ua + vb.$

Proof: Define $S = \{au + bv | u, v \in \mathbb{Z}\}$. We will prove:

$$ax + by = n \text{ is solvable in } \mathbb{Z} \Leftrightarrow n \in S \Leftrightarrow \gcd(a, b) | n.$$

The 1st $\Leftrightarrow$ and the 2nd $\Rightarrow$ are easy. Now we prove the 2nd $\Leftarrow$.

1. If $x, y \in S$ then $x \pm y \in S$.

$\because \exists u, v, s, t \in \mathbb{Z}$ s.t. $x = au + bv$ and $y = as + bt$
$\therefore x \pm y = a(u \pm s) + b(v \pm t) \in S.$

2. If $x \in S$ then $cx \in S$ for $\forall c \in \mathbb{Z}$.

$x = au + bv \in S \quad \Rightarrow \quad cx = a(cu) + b(cv) \in S.$

3. Let d =Min$(S \cap \mathbb{Z}^+)$. Prove S is the set of all multiples of d.

$\because a, b \in S \quad \therefore \pm a, \pm b \in S \quad \therefore \phi \neq S \cap \mathbb{Z}^+$.
Let d =Min$(S \cap \mathbb{Z}^+)$, $\quad \therefore$ All multiples of d are in S by (2).

On the other hand, $\forall x \in S$, we have $x = qd + r$, where
$0 \leq r < d$. $\because dq \in S \quad \therefore r = x - dq \in S$. $\quad \therefore r = 0$.

4. $d = \gcd(a, b)$.

Let $D = \gcd(a, b)$ and $d = au + bv$. $\because D|a,\ D|b, \quad \therefore D|d$.
$\because a, b \in S \quad \therefore d|a$ and $d|b$ by (3).
$\therefore d$ is a common divisor of a and b. $\therefore d \leq D$. $\quad \therefore\ d = D$.

Define $S = \{au + bv | u, v \in \mathbb{Z}\}$. We have proved:

$$ax + by = n \text{ is solvable in } \mathbb{Z} \Leftrightarrow n \in S \Leftrightarrow \gcd(a, b)|n.$$

3. Let d =Min$(S \cap \mathbb{Z}^+)$. Prove S is the set of all multiples of d.

$\because a, b \in S \quad \therefore \pm a, \pm b \in S \quad \therefore \phi \neq S \cap \mathbb{Z}^+$.
Let d =Min$(S \cap \mathbb{Z}^+)$, $\quad \therefore$ All multiples of d are in S by (2).

On the other hand, $\forall x \in S$, we have $x = qd + r$, where $0 \leq r < d$. $\because dq \in S \quad \therefore r = x - dq \in S. \quad \therefore r = 0$.

4. $d = \gcd(a, b)$.

Let $D = \gcd(a, b)$ and $d = au + bv$. $\because D|a,\ D|b, \quad \therefore D|d$.
$\because a, b \in S \quad \therefore d|a$ and $d|b$ by (3).
$\therefore d$ is a common divisor of a and b. $\therefore d \leq D. \quad \therefore\ d = D$.

Define $S = \{au + bv | u, v \in \mathbb{Z}\}$. We have proved:

$$ax + by = n \text{ is solvable in } \mathbb{Z} \Leftrightarrow n \in S \Leftrightarrow \gcd(a, b)|n.$$

The extended Euclidean algorithm

Problem

Find d, u and $v \in \mathbb{Z}$, s.t. $d = gcd(a,b) = ua + vb$.
Integers u and v are called Bezout coefficients.

Notes:

Bezout coefficients u and v are not unique:

$$\gcd(a,b) = ua + vb = ua + ba - ab + vb = (u+b)a + (v-a)b$$

Coefficients u and v are useful for the computation of modular multiplicative inverses in $\mathbb{Z}_n$, i.e.,

If $\quad 1 = d = gcd(a,b) = ua + vb \quad$ then $\quad 1 \equiv vb \pmod{a}$.
So, v is the multiplicative inverse of b modulo a.

The extended Euclidean algorithm

Problem

Find d, u and $v \in \mathbb{Z}$, s.t. $d = gcd(a, b) = ua + vb$.
Integers u and v are called Bezout coefficients.

Notes:

Bezout coefficients u and v are not unique:
$\gcd(a, b) = ua + vb = ua + ba - ab + vb = (u + b)a + (v - a)b$

Coefficients u and v are useful for the computation of modular multiplicative inverses in $\mathbb{Z}_n$, i.e.,

If $\quad 1 = d = gcd(a, b) = ua + vb \quad$ then $\quad 1 \equiv vb \pmod{a}$.
So, v is the multiplicative inverse of b modulo a.

Example: $\gcd(a = 252, b = 198)$

$252 = 1 \cdot 198 + 54$			$18 = -198 + 4 \cdot (252 - 1 \times 198)$ $= 4 \times 252 - 5 \cdot 198$
$198 = 3 \cdot 54 + 36$	$\downarrow$		$18 = 1 \cdot 54 - 1 \cdot (198 - 3 \cdot 54)$ $= -1 \cdot 198 + 4 \cdot 54$
$54 = 1 \cdot 36 + 18$		$\uparrow$	$18 = 0 \cdot 36 + 1 \cdot (54 - 1 \cdot 36)$ $= 1 \cdot 54 - 1 \cdot 36$
$36 = 2 \cdot 18 + (0)$			$18 = 1 \cdot 18 + 0 \cdot (36 - 2 \cdot 18)$ $= 0 \cdot 36 + 1 \cdot 18$
call$(18, 0)$ and return $18 = 1 \cdot 18 + 0 \cdot (0)$;			

$$\therefore 18 = \gcd(252, 198) = 4 \cdot 252 - 5 \cdot 198.$$

Bezout coefficients are not unique:

If " return $18 = 1 \cdot 18 + 1 \cdot (0)$ " then $18 = -7 \cdot 252 + 9 \cdot 198$.

The following recursive program is based on this idea.

Example: $\gcd(a = 252, b = 198)$

$252 = 1 \cdot 198 + 54$	$\downarrow$	$\uparrow$	$18 = -198 + 4 \cdot (252 - 1 \times 198)$ $= 4 \times 252 - 5 \cdot 198$
$198 = 3 \cdot 54 + 36$			$18 = 1 \cdot 54 - 1 \cdot (198 - 3 \cdot 54)$ $= -1 \cdot 198 + 4 \cdot 54$
$54 = 1 \cdot 36 + 18$			$18 = 0 \cdot 36 + 1 \cdot (54 - 1 \cdot 36)$ $= 1 \cdot 54 - 1 \cdot 36$
$36 = 2 \cdot 18 + (0)$			$18 = 1 \cdot 18 + 0 \cdot (36 - 2 \cdot 18)$ $= 0 \cdot 36 + 1 \cdot 18$
call$(18, 0)$ and return $18 = 1 \cdot 18 + 0 \cdot (0)$;			

$$\therefore 18 = \gcd(252, 198) = 4 \cdot 252 - 5 \cdot 198.$$

Bezout coefficients are not unique:

If " return $18 = 1 \cdot 18 + 1 \cdot (0)$ " then $18 = -7 \cdot 252 + 9 \cdot 198$.

The following recursive program is based on this idea.

A recursive extended Euclidean algorithm $E\_gcd(a, b)$

INPUT: two positive integers $a > b$.
OUTPUT: $(d,\ u,\ v)$ that satisfies $d = \gcd(a, b) = ua + vb$.
LOCAL: q, r, u, v, u', v'; d, d' can be global.

If $b = 0$ return $(a,\ 1,\ 0)$; $\because a = \gcd(a, 0) = 1 \cdot a + 0 \cdot 0$
Let $a := qb + r$;
$(d',\ u',\ v') := E\_gcd(b, r)$;
return $(d,\ u,\ v) := (d',\ v',\ u' - qv')$;

Proof: 1. $d = \gcd(a, b) = \gcd(b, r) = d'$.

2. After the recursive step, we get u' and v' s.t. $d' = bu' + rv'$.
We want to compute u and v in $d = ua + vb$ at the next step:
$\because d = d' = bu' + rv' = bu' + (a - qb)v' = v'a + (u' - qv')b$,
$\therefore$ Choosing $u = v'$ and $v = u' - qv'$ satisfies $d = ua + vb$.

We present a nonrecursive program in the following.

A recursive extended Euclidean algorithm $E\_gcd(a, b)$

INPUT: two positive integers $a > b$.
OUTPUT: $(d, \; u, \; v)$ that satisfies $d = \gcd(a, b) = ua + vb$.
LOCAL: q, r, u, v, u', v'; $\quad$ d, d' can be global.

If $b = 0$ return $(a, \; 1, \; 0)$; $\because a = \gcd(a, 0) = 1 \cdot a + 0 \cdot 0$
Let $a := qb + r$;
$(d', \; u', \; v') := E\_gcd(b, r)$;
return $(d, \; u, \; v) := (d', \; v', \; u' - qv')$;

Proof: 1. $d = \gcd(a, b) = \gcd(b, r) = d'$.

2. After the recursive step, we get u' and v' s.t. $d' = bu' + rv'$.
We want to compute u and v in $d = ua + vb$ at the next step:
$\because d = d' = bu' + rv' = bu' + (a - qb)v' = v'a + (u' - qv')b$,
$\therefore$ Choosing $u = v'$ and $v = u' - qv'$ satisfies $d = ua + vb$.

We present a nonrecursive program in the following.

A recursive extended Euclidean algorithm $E\_gcd(a, b)$

INPUT: two positive integers $a > b$.
OUTPUT: $(d, \ u, \ v)$ that satisfies $d = \gcd(a, b) = ua + vb$.
LOCAL: q, r, u, v, u', v'; $\quad d, d'$ can be global.

If $b = 0$ return $(a, \ 1, \ 0)$; $\because a = \gcd(a, 0) = 1 \cdot a + 0 \cdot 0$
Let $a := qb + r$;
$(d', \ u', \ v') := E\_gcd(b, r)$;
return $(d, \ u, \ v) := (d', \ v', \ u' - qv')$;

Proof: 1. $d = \gcd(a, b) = \gcd(b, r) = d'$.

2. After the recursive step, we get u' and v' s.t. $d' = bu' + rv'$.
We want to compute u and v in $d = ua + vb$ at the next step:
$\because d = d' = bu' + rv' = bu' + (a - qb)v' = v'a + (u' - qv')b$,
$\therefore$ Choosing $u = v'$ and $v = u' - qv'$ satisfies $d = ua + vb$.

We present a nonrecursive program in the following.

$$
\begin{array}{lclclcl}
r_{-1} = a & & r_0 = b & & r_{-1} & = & 1 \cdot a + 0 \cdot b \\
 & & & & r_0 & = & 0 \cdot a + 1 \cdot b \\
r_{-1} & = & q_1 r_0 + r_1 & & r_1 & = & r_{-1} - q_1 r_0 = 1a - q_1 b \\
r_0 & = & q_2 r_1 + r_2 & & r_2 & = & r_0 - q_2 r_1 = -q_2 a + (1 + q_1 q_2) b \\
 & \cdots & & & & \cdots & \\
r_{i-3} & = & q_{i-1} r_{i-2} + r_{i-1} & & r_{i-1} & = & u_{i-1} a + v_{i-1} b \\
r_{i-2} & = & q_i r_{i-1} + r_i & & r_i & = & u_i a + v_i b \\
r_{i-1} & = & q_{i+1} r_i + r_{i+1} & & r_{i+1} & = & r_{i-1} - q_{i+1} r_i = u_{i+1} a + v_{i+1} b \\
 & \cdots & & & & \cdots & \\
r_{t-2} & = & q_t r_{t-1} + r_t & & r_t & = & u_t a + v_t b = \gcd(a, b) \\
r_{t-1} & = & q_{t+1} r_t + 0 & & 0 & = & u_{t+1} a + v_{t+1} b
\end{array}
$$

$\because$ Both r_{i-1} and r_i are linear combinations of a and b,
$\therefore r_{i+1} = r_{i-1} - q_{i+1} r_i \in S$ is also the linear combination of a and b.

$$
\begin{aligned}
r_{i+1} &= u_{i+1} a + v_{i+1} b = r_{i-1} - q_{i+1} r_i = [u_{i-1} a + v_{i-1} b] - q_{i+1} [u_i a + v_i b] \\
&= (u_{i-1} - q_{i+1} u_i) a + (v_{i-1} - q_{i+1} v_i) b.
\end{aligned}
$$

$$
\begin{array}{llllll}
r_{-1} = a & & r_0 = b & r_{-1} & = & 1 \cdot a + 0 \cdot b \\
 & & & r_0 & = & 0 \cdot a + 1 \cdot b \\
r_{-1} & = & q_1 r_0 + r_1 & r_1 & = & r_{-1} - q_1 r_0 = 1a - q_1 b \\
r_0 & = & q_2 r_1 + r_2 & r_2 & = & r_0 - q_2 r_1 = -q_2 a + (1 + q_1 q_2) b \\
 & \cdots & & & \cdots & \\
r_{i-3} & = & q_{i-1} r_{i-2} + r_{i-1} & r_{i-1} & = & u_{i-1} a + v_{i-1} b \\
r_{i-2} & = & q_i r_{i-1} + r_i & r_i & = & u_i a + v_i b \\
r_{i-1} & = & q_{i+1} r_i + r_{i+1} & r_{i+1} & = & r_{i-1} - q_{i+1} r_i = u_{i+1} a + v_{i+1} b \\
 & \cdots & & & \cdots & \\
r_{t-2} & = & q_t r_{t-1} + r_t & r_t & = & u_t a + v_t b = \gcd(a, b) \\
r_{t-1} & = & q_{t+1} r_t + 0 & 0 & = & u_{t+1} a + v_{t+1} b
\end{array}
$$

$$
\begin{aligned}
r_{i+1} &= u_{i+1} a + v_{i+1} b = r_{i-1} - q_{i+1} r_i = [u_{i-1} a + v_{i-1} b] - q_{i+1} [u_i a + v_i b] \\
&= (u_{i-1} - q_{i+1} u_i) a + (v_{i-1} - q_{i+1} v_i) b.
\end{aligned}
$$

$$
\begin{array}{llll}
u_{-1} = 1, & u_0 = 0, & u_{i+1} = u_{i-1} - q_{i+1} u_i, & 0 \le i \le t, \\
v_{-1} = 0, & v_0 = 1, & v_{i+1} = v_{i-1} - q_{i+1} v_i, & 0 \le i \le t.
\end{array}
$$

$$\begin{array}{llll}
r_{-1} = a, & r_0 = b, & r_{i+1} = r_{i-1} - q_{i+1} r_i, & 0 \le i \le t \\
u_{-1} = 1, & u_0 = 0, & u_{i+1} = u_{i-1} - q_{i+1} u_i, & 0 \le i \le t, \\
v_{-1} = 0, & v_0 = 1, & v_{i+1} = v_{i-1} - q_{i+1} v_i, & 0 \le i \le t. \\
 & & r_t = u_t a + v_t b, & r_t = \gcd(a, b), \\
 & \textit{Stop if} & r_{t+1} = u_{t+1} a + v_{t+1} b = 0. &
\end{array}$$

A nonrecursive extended Euclidean algorithm $E\_gcd(A, B)$

OUTPUT: (d, u, v) that satisfies $d = gcd(A, B) = uA + vB$.

$(a,\ b) := (A,\ B)$; $(u,\ e) := (1,\ 0)$; $(v,\ f) := (0,\ 1)$;
While $b \neq 0$ do
  Let $a := qb + r$;
  $(a,\ b) := (b,\ a - qb = r)$;
  $(u,\ e) := (e,\ u - qe)$;
  $(v,\ f) := (f,\ v - qf)$;
return$(a,\ u,\ v)$.

A Matrix Interpretation of $E\_gcd(a, b)$

Theorem (The Euclidean theorem, 300 B.C.)

If $a = bq + r$ *then* $\gcd(a, b) = \gcd(b, r)$.

The Euclidean algorithm: $\gcd(a, b)$

INPUT: $a > 0$ and $b > 0$. **OUTPUT:** $\gcd(a, b)$.

If $(b = 0)$ then return a;

Let $a = bq + r$; Return $\gcd(b,\ r = a - bq)$;

$(a,\ b)$ is replaced by $(b,\ a \bmod b)$ in each iteration.

Schönhage formulated this as a matrix multiplication in 1971:

$$(b, a \bmod b) = (a, b) \begin{pmatrix} 0 & 1 \\ 1 & -q \end{pmatrix}$$

A Matrix Interpretation of $E\_gcd(a, b)$

Theorem (The Euclidean theorem, 300 B.C.)

If $a = bq + r$ *then* $\gcd(a, b) = \gcd(b, r)$.

The Euclidean algorithm: $\gcd(a, b)$

INPUT: $a > 0$ and $b > 0$. **OUTPUT:** $\gcd(a, b)$.

If $(b = 0)$ then return a;

Let $a = bq + r$; Return $\gcd(b,\ r = a - bq)$;

$(a,\ b)$ is replaced by $(b,\ a \bmod b)$ in each iteration.

Schönhage formulated this as a matrix multiplication in 1971:

$$(b, a \bmod b) = (a, b) \begin{pmatrix} 0 & 1 \\ 1 & -q \end{pmatrix}$$

Example: $3 = E\_gcd(a = 15, b = 6)$

Step 1: $15 = 2 \times 6 + 3$

$$(6,3) = (15,6)\begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix}.$$

Step 2: $6 = 2 \times 3 + 0$

$$(3,0) = (6,3)\begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix}.$$

$$\therefore (3 = \gcd(a,b), 0) = (a,b)\begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix}$$

A Matrix Interpretation of $E\_gcd(a,b)$

If the algorithm terminates after $t+1$ iterations, i.e.,

$$r_{-1} = a = q_1 b + r_1 \qquad r_0 = b = q_2 r_1 + r_2 \quad \cdots$$
$$r_{t-2} = q_t r_{t-1} + r_t \quad \text{and} \quad r_{t-1} = q_{t+1} r_t + 0,$$

then we have

$$(\gcd(a,b), 0) = (a,b)\left[\begin{pmatrix} 0 & 1 \\ 1 & -q_1 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & -q_2 \end{pmatrix}\cdots\begin{pmatrix} 0 & 1 \\ 1 & -q_{t+1} \end{pmatrix}\right]$$

Let $\begin{pmatrix} u & e \\ v & f \end{pmatrix} := [\cdots]$. Then we have

$$(\gcd(a,b), 0) = (a,b)\begin{pmatrix} u & e \\ v & f \end{pmatrix}$$

$\therefore \exists u, v \in \mathbb{Z}$ s.t. $\gcd(a,b) = ua + vb$.

A Matrix Interpretation of $E\_gcd(a, b)$

If the algorithm terminates after $t+1$ iterations, i.e.,

$$r_{-1} = a = q_1 b + r_1 \qquad r_0 = b = q_2 r_1 + r_2 \quad \cdots$$
$$r_{t-2} = q_t r_{t-1} + r_t \quad \text{and} \quad r_{t-1} = q_{t+1} r_t + 0,$$

then we have

$$(\gcd(a,b), 0) = (a,b)\left[\begin{pmatrix} 0 & 1 \\ 1 & -q_1 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & -q_2 \end{pmatrix}\cdots\begin{pmatrix} 0 & 1 \\ 1 & -q_{t+1} \end{pmatrix}\right]$$

Let $\begin{pmatrix} u & e \\ v & f \end{pmatrix} := [\cdots]$. Then we have

$$(\gcd(a,b), 0) = (a,b)\begin{pmatrix} u & e \\ v & f \end{pmatrix}$$

$\therefore \exists u, v \in \mathbb{Z}$ s.t. $\gcd(a,b) = ua + vb$.

Example: $E\_gcd(a = 15, b = 6)$

$$15 = 2 \times 6 + 3; \qquad 6 = 2 \times 3 + 0.$$

$$(\gcd(a,b), 0) = (a,b) \begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \xrightarrow{\times \begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix}} \begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix} \xrightarrow{\times \begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix}} \begin{pmatrix} 1 & -2 \\ -2 & 5 \end{pmatrix}$$

$$\therefore (3,\ 0) = (15,\ 6) \begin{pmatrix} 1 & -2 \\ -2 & 5 \end{pmatrix} \text{ and}$$

$$3 = \gcd(15,\ 6) = 15 \times 1 + 6 \times (-2).$$

A matrix version of $E\_gcd(A, B)$

INPUT: two positive integers $A > B > 0$.
OUTPUT: (d, u, v) that satisfies $d = gcd(A, B) = uA + vB$.

$$\begin{pmatrix} u & e \\ v & f \end{pmatrix} := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix};$$

$(a, \ b) := (A, \ B)$;

While $b \neq 0$ do

Let $a := qb + r$;

$$\begin{pmatrix} u & e \\ v & f \end{pmatrix} := \begin{pmatrix} u & e \\ v & f \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -q \end{pmatrix};$$

$(a, \ b) := (b, \ r)$;

return$(a = Au + Bv, \ u, \ v)$.

A matrix version of $E\_gcd(A, B)$

INPUT: two positive integers $A > B > 0$.
OUTPUT: (d, u, v) that satisfies $d = gcd(A, B) = uA + vB$.

$$\begin{pmatrix} u & e \\ v & f \\ a & b \end{pmatrix} := \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ A & B \end{pmatrix};$$

While $b \neq 0$ do

Let $a := qb + r$;

$$\begin{pmatrix} u & e \\ v & f \\ a & b \end{pmatrix} := \begin{pmatrix} u & e \\ v & f \\ a & b \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -q \end{pmatrix};$$

return$(a,\ \ u,\ \ v)$.

Fibonacci number

Definition

$F_0 = 0, \quad F_1 = 1, \quad F_n = F_{n-1} + F_{n-2}.$

$$F_n = \left[\left(\tfrac{1+\sqrt{5}}{2}\right)^n - \left(\tfrac{1-\sqrt{5}}{2}\right)^n \right] \div \sqrt{5}$$

$$(F_n, F_{n-1}) = (F_{n-1}, F_{n-2}) \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = (F_1, F_0) \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-1}$$

Therefore we can compute F_n using $\mathcal{O}(\log_2 n)$ operations.

Can we improve the Euclidean algorithm further?

Theorem (The Euclidean theorem, 300 B.C.)

If $a = bq + r$ *then* $\gcd(a, b) = \gcd(b, r)$.

INPUT: $a > b > 0$. **OUTPUT:** $\gcd(a, b)$.

While $b \neq 0$ do // $(a,\ b) := (b,\ a \bmod b)$
 $t := b$;
 $b := a \bmod b$;
 $a := t$;
Return a;

Exercises

1. Are there $s, t \in \mathbb{Z}$ such that $24s + 14t = 1$?
2. Let a and b be two n-bit positive integers. Explain that the average bit complexity of the Euclidean algorithm is $\mathcal{O}(n^2)$.