Euclid’s Algorithm

Alexander S. Kulikov

Steklov Mathematical Institute at St. Petersburg, Russian Academy of Sciences
and
University of California, San Diego

Outline

Greatest Common Divisor

Euclid’s Algorithm

Extended Euclid’s Algorithm

Greatest Common Divisor

Definition

The greatest common divisor, $\gcd(a, b)$, of integers a and b (not both equal to zero) is the largest integer that divides both a and b

Greatest Common Divisor

Definition

The greatest common divisor, $\gcd(a, b)$, of integers a and b (not both equal to zero) is the largest integer that divides both a and b

Examples

$\gcd(24, 16) = 8$, $\gcd(9, 17) = 1$, $\gcd(239, 0) = 239$

Greatest Common Divisor

Definition

The greatest common divisor, $\gcd(a, b)$, of integers a and b (not both equal to zero) is the largest integer that divides both a and b

Examples

$\gcd(24, 16) = 8$, $\gcd(9, 17) = 1$, $\gcd(239, 0) = 239$

Convention

We assume that a and b are non-negative

First Application: Computing Inverses

Given integers a and n, how to find an integer k such that $ak \equiv 1 \mod n$? Basic primitive in modern crypto protocols, used billions times per day

pexels.com

Second Application: Computing Fractions

$$\frac{31}{177} + \frac{29}{59}$$

Second Application: Computing Fractions

$$\frac{31}{177} + \frac{29}{59} = \frac{31 \times 59 + 177 \times 29}{177 \times 59}$$

Second Application: Computing Fractions

$$\frac{31}{177} + \frac{29}{59} = \frac{31 \times 59 + 177 \times 29}{177 \times 59} = \frac{6962}{10443}$$

Second Application: Computing Fractions

$$\frac{31}{177} + \frac{29}{59} = \frac{31 \times 59 + 177 \times 29}{177 \times 59} = \frac{6962}{10443} = \frac{2}{3}$$

Second Application: Computing Fractions

$$\frac{31}{177} + \frac{29}{59} = \frac{31 \times 59 + 177 \times 29}{177 \times 59} = \frac{6962}{10443} = \frac{2}{3}$$

```
from fractions import Fraction
print(Fraction(31, 177) + Fraction(29, 59))
```

Second Application: Computing Fractions

$$\frac{31}{177} + \frac{29}{59} = \frac{31 \times 59 + 177 \times 29}{177 \times 59} = \frac{6962}{10443} = \frac{2}{3}$$

```
from fractions import Fraction
print(Fraction(31, 177) + Fraction(29, 59))
```

```
2/3
```

Naive Algorithm

To find the greatest common divisor, simply try all numbers and select the largest one

Naive Algorithm: Code

```
def gcd(a, b):
  assert a >= 0 and b >= 0 and a + b > 0

  if a == 0 or b == 0:
    return max(a, b)

  for d in range(min(a, b), 0, -1):
    if a % d == 0 and b % d == 0:
      return d

  return 1
```

Naive Algorithm: Code

```
def gcd(a, b):
  assert a >= 0 and b >= 0 and a + b > 0

  if a == 0 or b == 0:
    return max(a, b)

  for d in range(min(a, b), 0, -1):
    if a % d == 0 and b % d == 0:
      return d

  return 1
```

```
print(gcd(24, 16))
```

```
8
```

Naive Algorithm: Analysis

- If $\gcd(a, b) = 1$, the algorithm will perform $\min\{a, b\}$ divisions

Naive Algorithm: Analysis

- If $\gcd(a, b) = 1$, the algorithm will perform $\min\{a, b\}$ divisions
- Modern laptops perform roughly one billion (10^9) operations per second

Naive Algorithm: Analysis

- If $\gcd(a, b) = 1$, the algorithm will perform $\min\{a, b\}$ divisions
- Modern laptops perform roughly one billion (10^9) operations per second
- On your laptop, the call

  ```
  print(gcd(790933790547, 1849639579327))
  ```

  will take more than one minute

Supercomputer

If *a* and *b* consist of hundreds of digits (typical case for crypto protocols), even on a supercomputer with quadrillion operations per second this algorithm will run for more than thousand years

Next Parts

Euclid's algorithm:
efficient algorithm for computing the greatest common divisor of two integers

Outline

Euclid's Lemma

Euclid's Lemma

d divides a and b, if and only if d divides $a - b$ and b.

Euclid's Lemma

Euclid's Lemma

d divides a and b, if and only if d divides $a - b$ and b.

Proof

$$\Rightarrow \text{ if } a = dp \text{ and } b = dq\text{, then}$$
$$a - b = d(p - q)$$

Euclid's Lemma

Euclid's Lemma

d divides a and b, if and only if d divides $a - b$ and b.

Proof

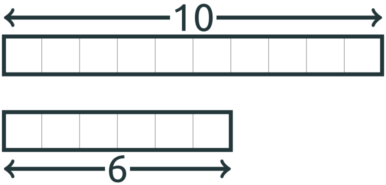
$\Rightarrow$ if $a = dp$ and $b = dq$, then
$a - b = d(p - q)$

$\Leftarrow$ if $a - b = dp$ and $b = dq$, then
$a = d(p + q)$

$\square$

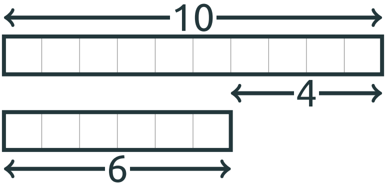
Euclid's Lemma: Pictorially

2 divides 10 and 6, hence 2 divides $4 = 10 - 6$



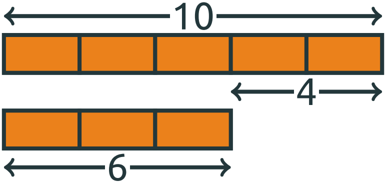
Euclid’s Lemma: Pictorially

2 divides 10 and 6, hence 2 divides $4 = 10 - 6$



Euclid's Lemma: Pictorially

2 divides 10 and 6, hence 2 divides $4 = 10 - 6$



Algorithm

```
def gcd(a, b):
  assert a >= 0 and b >= 0 and a + b > 0

  while a > 0 and b > 0:
    if a >= b:
      a = a - b
    else:
      b = b - a

  return max(a, b)
```

Analysis

- On some inputs, works much faster than the previous algorithm

Analysis

- On some inputs, works much faster than the previous algorithm
- Still, there are inputs where this code is too slow:

```
gcd(790933790548, 7)
```

Analysis

- On some inputs, works much faster than the previous algorithm
- Still, there are inputs where this code is too slow:

```
gcd(790933790548, 7)
```

- Reason: the code will subtract 7 billions of times!

Analysis

- On some inputs, works much faster than the previous algorithm
- Still, there are inputs where this code is too slow:

```
gcd(790933790548, 7)
```

- Reason: the code will subtract 7 billions of times!
- Idea: what is left is the reminder modulo 7

Euclid’s Algorithm

```
def gcd(a, b):
  assert a >= 0 and b >= 0 and a + b > 0

  while a > 0 and b > 0:
    if a >= b:
      a = a % b
    else:
      b = b % a

  return max(a, b)
```

Analysis

- Already quite fast: if *a* and *b* are 100 digits long, the number of iterations of the `while` loop is at most 660

Analysis

- Already quite fast: if a and b are 100 digits long, the number of iterations of the `while` loop is at most 660
- Each iteration is a division

Code with Logging

```
def gcd(a, b):
  assert a >= 0 and b >= 0 and a + b > 0

  while a > 0 and b > 0:
    print("gcd({},␣{})=".format(a, b))
    if a >= b:
      a = a % b
    else:
      b = b % a

  print("gcd({},␣{})=".format(a, b))
  return max(a, b)

print(gcd(790933790547, 1849639579327))
```

Code with Logging: Output

```
gcd(790933790547, 1849639579327)=
gcd(790933790547, 267771998233)=
gcd(255389794081, 267771998233)=
gcd(255389794081, 12382204152)=
gcd(7745711041, 12382204152)=
gcd(7745711041, 4636493111)=
gcd(3109217930, 4636493111)=
gcd(3109217930, 1527275181)=
gcd(54667568, 1527275181)=
gcd(54667568, 51250845)=
gcd(3416723, 51250845)=
gcd(3416723, 0)=
3416723
```

Analysis

- The numbers are getting shorter and shorter
- A more quantitative statement: at each iteration of the `while` loop the larger number drops by at least a factor of 2

Analysis

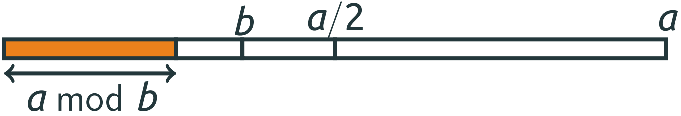
- The numbers are getting shorter and shorter
- A more quantitative statement: at each iteration of the `while` loop the larger number drops by at least a factor of 2

Lemma

Let $a \geq b > 0$. Then $(a \bmod b) < a/2$.

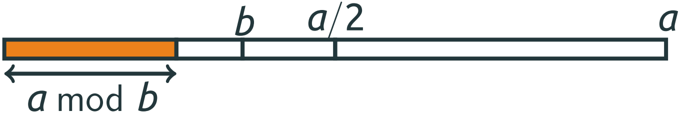
Proof

- If $b \leq a/2$, then $(a \bmod b) < b \leq a/2$.

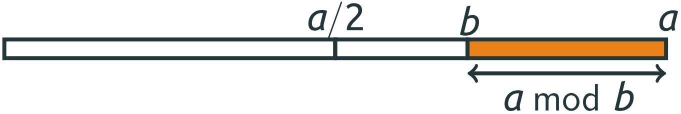


Proof

- If $b \leq a/2$, then $(a \bmod b) < b \leq a/2$.



- If $b > a/2$, then $(a \bmod b) = a - b < a/2$.



$\square$

Final Analysis

- Hence, at each iteration, either a or b is dropped by at least a factor of 2

Final Analysis

- Hence, at each iteration, either a or b is dropped by at least a factor of 2
- Thus, the total number of iterations is at most $\log_2 a + \log_2 b$

Final Analysis

- Hence, at each iteration, either a or b is dropped by at least a factor of 2
- Thus, the total number of iterations is at most $\log_2 a + \log_2 b$
- If a consists of less than 5 000 decimal digits (i.e., $a < 10^{5\,000}$), then $\log_2 a < 16\,610$

Compact Code

```python
def gcd(a, b):
  assert a >= b and b >= 0 and a + b > 0
  return gcd(b, a % b) if b > 0 else a
```

Outline

Greatest Common Divisor

Euclid's Algorithm

Extended Euclid's Algorithm

Certificate

- Somebody computed the greatest common divisor of a and b and wants to convince you that it is equal to d

Certificate

- Somebody computed the greatest common divisor of a and b and wants to convince you that it is equal to d
- You can check that d divides both a and b, but this only shows that d is a common divisor of a and b, but does not guarantees that is the greatest one

Certificate

- Somebody computed the greatest common divisor of a and b and wants to convince you that it is equal to d
- You can check that d divides both a and b, but this only shows that d is a common divisor of a and b, but does not guarantees that is the greatest one
- It turns out that it is enough to represent d as $ax + by$ (for integers x and y)!

Test

Lemma

If d divides a and b and $d = ax + by$ for integers x and y, then $d = \gcd(a, b)$.

Test

Lemma

If d divides a and b and $d = ax + by$ for integers x and y, then $d = \gcd(a, b)$.

Proof

- d is a common divisor of a and b, hence $d \leq \gcd(a, b)$

Test

Lemma

If d divides a and b and $d = ax + by$ for integers x and y, then $d = \gcd(a, b)$.

Proof

- d is a common divisor of a and b, hence $d \leq \gcd(a, b)$
- $\gcd(a, b)$ divides both a and b, hence it also divides $d = ax + by$, and hence $\gcd(a, b) \leq d$ $\square$

Examples

- $\gcd(10, 6) = 2 = 10 \cdot (-1) + 6 \cdot 2$

Examples

- $\gcd(10, 6) = 2 = 10 \cdot (-1) + 6 \cdot 2$
- $\gcd(7, 5) = 1 = 7 \cdot (-2) + 5 \cdot 3$

Examples

- $\gcd(10, 6) = 2 = 10 \cdot (-1) + 6 \cdot 2$
- $\gcd(7, 5) = 1 = 7 \cdot (-2) + 5 \cdot 3$
- $\gcd(391, 299) = 23 = 391 \cdot (-3) + 299 \cdot 4$

Examples

- $\gcd(10, 6) = 2 = 10 \cdot (-1) + 6 \cdot 2$
- $\gcd(7, 5) = 1 = 7 \cdot (-2) + 5 \cdot 3$
- $\gcd(391, 299) = 23 = 391 \cdot (-3) + 299 \cdot 4$
- $\gcd(239, 201) = 1 = 239 \cdot (-37) + 201 \cdot 44$

Extending Euclid's Algorithm

- Recall that Euclid's algorithm uses the fact that, for $a \geq b$, $\gcd(a, b) = \gcd(b, a \bmod b)$

Extending Euclid's Algorithm

- Recall that Euclid's algorithm uses the fact that, for $a \geq b$, $\gcd(a, b) = \gcd(b, a \bmod b)$
- Assume that $d = \gcd(b, a \bmod b)$ and that $d = bp + (a \bmod b)q$

Extending Euclid's Algorithm

- Recall that Euclid's algorithm uses the fact that, for $a \geq b$, $\gcd(a, b) = \gcd(b, a \bmod b)$
- Assume that $d = \gcd(b, a \bmod b)$ and that $d = bp + (a \bmod b)q$
- Then

$$
\begin{aligned}
d &= bp + (a \bmod b)q \\
&= bp + \left(a - \left\lfloor \frac{a}{b} \right\rfloor b\right)q \\
&= aq + b\left(p - \left\lfloor \frac{a}{b} \right\rfloor q\right)
\end{aligned}
$$

Extended Euclid's Algorithm

```
# returns gcd(a,b), x, y: gcd(a,b)=ax+by
def extended_gcd(a, b):
  assert a >= b and b >= 0 and a + b > 0

  if b == 0:
    d, x, y = a, 1, 0
  else:
    (d, p, q) = extended_gcd(b, a % b)
    x = q
    y = p - q * (a // b)

  assert a % d == 0 and b % d == 0
  assert d == a * x + b * y
  return (d, x, y)
```