Division Algorithm, Euclidean Algorithm

Robert Y. Lewis

CS 0220 2024

March 1, 2024

Overview

1 Quotients and Remainders (8.1.2)

2 The Greatest Common Divisor (8.2)

3 Euclid's Algorithm (8.2.1)

Division theorem

Theorem: Let n and $d > 0$ be integers. There exists a unique pair of integers q and r, such that $n = q \cdot d + r$ and $0 \leq r < d$.

$q = \mathrm{qcnt}(n, d)$ is the quotient, $r = \mathrm{rem}(n, d)$ is the remainder. I'd call them "integer division" and "mod".

Examples:

- $\mathrm{qcnt}(2716, 10) = 271$. Since $2716 = 271 \cdot 10 + 6$
- $\mathrm{rem}(2716, 10) = 6$. Same reason.
- $\mathrm{rem}(-11, 7) = 3$. Since $-11 = -2 \cdot 7 + 3$

Definitions

Definition: c is a *common divisor* of a and b if $c|a$ and $c|b$.

Example: 2 is a common divisor of 24 and 54.

Definition: c is the *greatest common divisor* (GCD) of a and b if c is a common divisor of a and b and no other common divisor is larger.

Example: 6 is the greatest common divisor of 24 and 54.

What's the greatest common divisor of 0 and 0?

A not-so-good algorithm for GCD

```
def gcd($a$,$b$):
    biggest = 0
    $n = \max(a, b)$
    for $i \in [0, n]$:
        if $i|a$ AND $i|b$ AND $i >$ biggest: biggest = $i$
    return(biggest)
```

Is it correct? Sure, it checks every value and returns the biggest. Except at (0, 0)...

Is there at least one common divisor? Well, 1 should always work.

Running time? *n*, which might be quite big.

Another not-so-good algorithm for GCD

```
def gcd($a$,$b$):
    $l_1 =$ prime factors of $a$
    $l_2 =$ prime factors of $b$
    $k = 1$
    while $l_1$ and $l_2$ have a number $x$ in common:
        remove $x$ from $l_1$ and $l_2$
        $k = k * x$
    return($k$)
```

Is it correct? Yes, the GCD can be constructed by collecting all the common prime factors.

Running time? The bottleneck is factoring, which can take up to time n, which might be quite big.

Wait, do we have to go up?

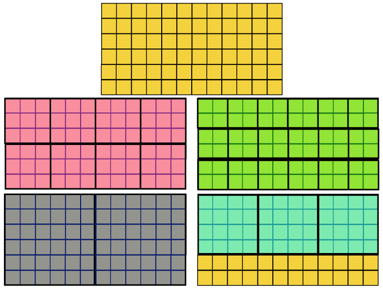
```
def gcd($a$,$b$):
    $n = \min(a, b)$
    for $i \in n$ down to 1:
        if $i|a$ and $i|b$: return($i$)
```

Is it correct? Sure, it tries the biggest possible value first and goes down until it works. Must be biggest, must work, must terminate (1).

Running time? Still could be n, too big.

Visual insights

Consider an $a \times b$ rectangle.



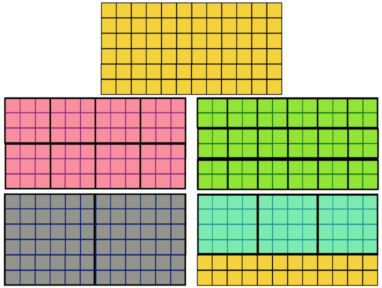
We can tile the rectangle with $c \times c$ squares if and only if c is a common divisor of a and b.

So, $\gcd(a, b)$ is the size of the *biggest* square that covers the $a \times b$ rectangle.

Special case

Let a be the smaller of a and b, without loss of generality. That just means we can make this assumption safely. Why? Because we can just rename the two values—there are no other constraints.

How can we check if $a = \gcd(a, b)$? If $\text{rem}(b, a) = 0$! Examples:



1, 3, 2, 6, are common divisors of 6 and 12. 4 is not.

Algorithm?

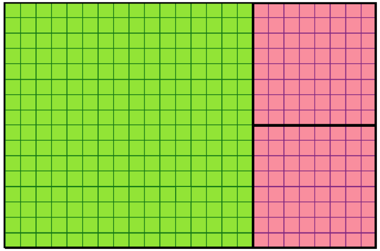
```
def gcd($a$,$b$):
    if $a|b$: return $a$
```

Is it correct? Yes, if it returns something. But, might not, which is bad.

Running time? One step!

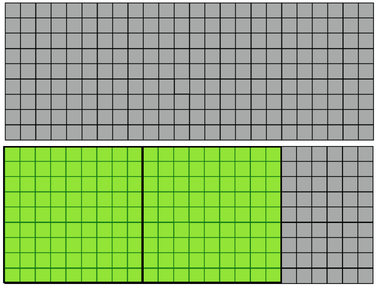
Another case: Difference

Consider gcd(16, 24). Our rule doesn't work because 16 doesn't divide 24 evenly. But $24 - 16 = 8$ does. Does that help?



Since $24 - 16$ divides 16 evenly, it must also divide 24 evenly. Note the 16×16 square. The 8×8 square fits on one side of it. But, it's a square, so it fits on the other side, too. It's a common divisor. Could there be a larger one? No. It would have to divide 16 evenly (green), so it would cover green. So, it would have to cover pink, too. Pink square is the biggest such square.

Sneaking up on an idea

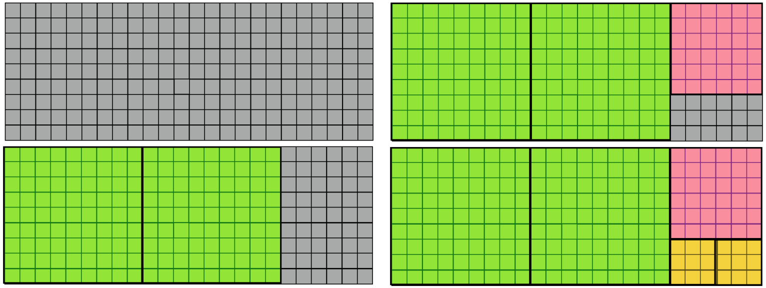


Make as many big $a \times a$ squares as possible. The biggest square that covers the leftover rectangle is necessarily the biggest square the covers the original rectangle.

Why? It has to evenly divide both sides.

So, solve the smaller problem and we solve the bigger problem.

Sneaking up on an idea (2)



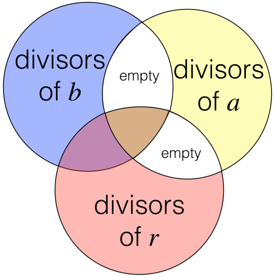
$\gcd(9, 24)$
$= \gcd(6, 9)$
$= \gcd(3, 6)$
$= 3$, by the original special case.

Remainder lemma

Lemma: When $a \le b$, $\text{gcd}(a, b) = \text{gcd}(\text{rem}(b, a), a)$.

Proof: Write $b = q \cdot a + r$ where $r = \text{rem}(b, a)$. Why? Division theorem. So, b is a linear combination of a and r, which implies that any divisor of a and r is a divisor of b. Why? Integer linear combination property.

Similarly, r is a linear combination of a and b, specifically $r = 1 \cdot b - q \cdot a$. Thus, any divisor of a and b is a divisor of r. Why? Integer linear combination property, again.



So, a and b have the same common divisors as a and r. They must also then have the same *greatest* common divisor. QED.

Euclid's algorithm

```
def gcd($a$,$b$):
    if $a = 0$: return($b$)
    else: return(gcd(rem($b, a$), $a$))
```

Is it correct? Yes, by the remainder lemma.

Running time? Notice that $\text{rem}(b, a) \leq b/2$. That's because $\text{rem}(b, a) \leq a$, so if $a \leq b/2$, it holds. Otherwise, $\text{rem}(b, a) = b - a \leq b/2$. Therefore, the "$a$" parameter is at least halved every 2 iterations. So, in $1 + 2 \log a$ iterations, terminates.

Example

$$
\begin{aligned}
&\gcd(1687, 2791)\\
&= \gcd(1104, 1687)\\
&= \gcd(583, 1104)\\
&= \gcd(521, 583)\\
&= \gcd(62, 521)\\
&= \gcd(25, 62)\\
&= \gcd(12, 25)\\
&= \gcd(1, 12)\\
&= \gcd(0, 1)\\
&= 1
\end{aligned}
$$