Greatest Common Divisor

Jordi Cortadella
Department of Computer Science

Simplifying fractions

\[
\frac{114}{42} = \frac{57}{21} = \frac{19}{7}
\]

\[
\frac{114}{42} = \frac{19 \cdot \text{gcd}}{7 \cdot \text{gcd}}
\]

gcd(114, 42) = 6

gcd(19, 7) = 1

The largest square tile

What is the largest square tile that can exactly cover a \(w \times h\) rectangle?

90 tiles of side-length 4

What is the largest square tile that can exactly cover a \(w \times h\) rectangle?

90 tiles of side-length 4

40 tiles of side-length 6

Introduction to Programming
The largest square tile

What is the largest square tile that can exactly cover a $w \times h$ rectangle?

$w = 60$

$h = 24$

- 90 tiles of side-length 4
- 40 tiles of side-length 6
- 10 tiles of side-length 12

Euclid (300 B.C.)

Euclidean algorithm for gcd

Observation:

if $d$ divides 114 and 42, it also divides $114 - 42$

$$\frac{114}{d} - \frac{42}{d} = \frac{72}{d}$$

Therefore, all common divisors of 114 and 42 are also divisors of 72.

$$\gcd(114, 42) = \gcd(72, 42)$$
Euclidean algorithm for gcd

- Properties:
  \[ \gcd(a, a) = a; \quad \gcd(a, 0) = a \]
  If \( a > b \), then \( \gcd(a, b) = \gcd(a - b, b) \)

- Example

\[
\begin{array}{cccc}
\hline
 a & b \\
114 & 42 \\
72 & 42 \\
30 & 42 \\
30 & 12 \\
18 & 12 \\
6 & 12 \\
6 & 6 \\
\hline
\end{array}
\]

```c
int gcd(int a, int b) {
    while (a != b) {
        if (a > b) a = a - b;
        else b = b - a;
    }
    return a;
}
```

// Pre: a > 0, b > 0
// Returns the greatest common divisor of a and b

10000001 154
11 64935
Faster Euclidean algorithm for $\text{gcd}$

- Properties:
  \[ \text{gcd}(a, 0) = a \]
  \[\text{If } b > 0 \text{ then } \text{gcd}(a, b) = \text{gcd}(a \% b, b)\]

- Example

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,000,001</td>
<td>154</td>
</tr>
<tr>
<td>11</td>
<td>154</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
</tr>
</tbody>
</table>

```
// Pre:    a \geq 0, b \geq 0
// Returns the greatest common divisor of a and b
int gcd(int a, int b) {
    while (a != 0 and b != 0) {
        if (a > b) a = a\%b;
        else b = b\%a;
    }
    return a + b;
}
```

Termination: $a = 0$ or $b = 0$

Comparing algorithms for $\text{gcd}$

```
int gcd(int a, int b) {
    while (b > 0) {
        int r = a\%b;
        a = b;
        b = r;
    }
    return a;
}
```

```
int gcd(int a, int b) {
    while (a > 0 and b > 0) {
        int r = a\%b;
        if (a > b) a = a\%b;
        else b = b\%a;
    }
    return a + b;
}
```

```
int gcd(int a, int b) {
    while (b > 0) {
        int r = a\%b;
        a = b;
        b = r;
    }
    return a;
}
```

Every iteration:
- 3 comparisons
- 1 mod operation

Every iteration:
- 1 comparison
- 1 mod operation
Efficiency of the Euclidean algorithm

How many iterations will Euclid’s algorithm need to calculate \( \gcd(a, b) \) in the worst case (assume \( a > b \))?

- **Subtraction version:** \( a \) iterations (consider \( \gcd(1000000, 1) \))

- **Modulo version:** \( \leq 5 \cdot d(b) \) iterations, where \( d(b) \) is the number of digits of \( b \) (Gabriel Lamé, 1844)

### Binary Euclidean algorithm

Assume \( a \geq b \):

<table>
<thead>
<tr>
<th>( a )</th>
<th>( b )</th>
<th>( \gcd(a, b) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>132</td>
<td>84</td>
<td>( \times 2 )</td>
</tr>
<tr>
<td>66</td>
<td>42</td>
<td>( \times 2 )</td>
</tr>
<tr>
<td>33</td>
<td>21</td>
<td>( \times 3 )</td>
</tr>
<tr>
<td>6</td>
<td>21</td>
<td>( \times 2 )</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>( \times 3 )</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>( \times 3 )</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>( \times 3 )</td>
</tr>
</tbody>
</table>

// Pre: \( a \geq 0, b \geq 0 \)
// Returns the greatest common divisor of \( a \) and \( b \)
int gcd(int a, int b) {
    int r = 1; // Accumulates common powers of two
    while (a != 0 and b != 0) {
        if (a%2 == 0 and b%2 == 0) {
            a = a/2;
            b = b/2;
            r = r*2;
        } else if (a%2 == 0) a = a/2;
        else if (b%2 == 0) b = b/2;
        else if (a > b) a = (a - b)/2;
        else b = (b - a)/2;
    }
    return (a + b)*r;
}

Computers can perform \(*2\) and \(/2\) operations efficiently

\[(217)_{10} = (11011001)_2\]

\[(217 \cdot 2)_{10} = (434)_{10} = (110110010)_2\]

\[(217/2)_{10} = (108)_{10} = (1101100)_2\]

\[(217\%2)_{10} = 1 \text{ (least significant bit)}\]
// Pre:    a ≥ 0, b ≥ 0
// Returns the greatest common divisor of a and b
int gcd(int a, int b) {
    int r = 1; // Accumulates common powers of two
    while (a%2 == 0 and b%2 == 0) {
        a = a/2;
        b = b/2;
        r = r*2;
    }
    while (a != b) {
        if (a%2 == 0) a = a/2;
        else if (b%2 == 0) b = b/2;
        else if (a > b) a = (a - b)/2;
        else b = (b - a)/2;
    }
    return a*r;
}

Summary

• Euclid’s algorithm is very simple.
• It is widely used in some applications related to cryptography (e.g., electronic commerce).
• Euclid’s algorithm efficiently computes the gcd of very large numbers.
• Question: how would you compute the least common multiple of two numbers efficiently?