Today we discussed algorithms for finding greatest common divisors of two numbers. We presented Euclid’s algorithm and analyzed its correctness and bit complexity.

1 Greatest Common Divisor

Definition 1. Suppose \( u, v \in \mathbb{Z} \), \( u, v \geq 1 \). Then the Greatest Common Divisor of \( u \) and \( v \), denoted \( \text{gcd}(u, v) \), is the maximum \( d \in \mathbb{Z} \) such that \( d \mid u \) and \( d \mid v \). We also define \( \text{gcd}(u, 0) = u \), and \( \text{gcd}(0, 0) = 0 \).

1.1 Finding the Greatest Common Divisor

The standard (and inefficient) algorithm for finding the greatest common divisor of integers \( u \) and \( v \) uses the prime factorizations of \( u \) and \( v \). Suppose \( u = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k} \) and \( v = p_1^{\beta_1} p_2^{\beta_2} \cdots p_k^{\beta_k} \) where \( p_1, \ldots, p_k \) are distinct primes that divide at least one of \( u, v \), and \( \alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_k \) are nonnegative integers. Then

\[
\text{gcd}(u, v) = \prod_{i=1}^{k} p_i^{\min\{\alpha_i, \beta_i\}}.
\]

For example, \( 12 = 2^2 \cdot 3^1 \cdot 5^0 \) and \( 30 = 2^1 \cdot 3^1 \cdot 5^1 \). Then \( \text{gcd}(12, 30) = 2^1 \cdot 3^1 \cdot 5^0 = 6 \).

Factoring is not an easy problem, however. We do not know of a factoring algorithm that runs in time that’s polynomial in the input length. Therefore, a method of finding greatest common divisors that relies on factoring is not suitable for large numbers. A method that doesn’t use prime factorization is necessary. We now describe Euclid’s algorithm for finding the greatest common divisor of two numbers.

1.2 Euclid’s Algorithm

Euclid’s algorithm for finding greatest common divisors is based on the following lemma:

Lemma 1. Suppose \( d, u, v \in \mathbb{Z} \). Then \( d \mid u \) and \( d \mid v \) if and only if \( d \mid v \) and \( d \mid u - qv \) where \( q \in \mathbb{Z} \).

Proof: (⇒) If \( d \mid u \) and \( d \mid v \), then clearly \( d \mid v \). Since \( d \mid u \), there exists \( n \in \mathbb{Z} \) such that \( u = nd \). Similarly, there is \( m \in \mathbb{Z} \) such that \( v = md \). Then \( u - qv = nd - qmd = (n - qm)d \), so \( d \mid u - qv \) as well.

(⇐) Again, it is trivial to see that if \( d \mid v \) and \( d \mid u - qv \), then \( d \mid v \). We can write \( v = md \) and \( u - qv = nd \) for some \( m, n \in \mathbb{Z} \). Then \( u = nd + qv = nd + qmd = (n + qm)d \), so \( d \mid u \). □

We want to find the greatest common divisor of integers \( u_0 \) and \( u_1 \), with \( u_0 \geq u_1 \geq 1 \). We describe how to find the greatest common divisor of \( u_0 \) and \( u_1 \) using Euclid’s algorithm. This proceeds in steps. In each step, we reduce the problem to finding the greatest common divisor of two integers that are smaller than in the previous step.
In step \( i \), we want to find the greatest common divisor of \( u_{i-1} \) and \( u_i \). We divide \( u_{i-1} \) by \( u_i \) and find the remainder, which we will call \( u_{i+1} \). Then we move to step \( i + 1 \) where we try to find the greatest common divisor of \( u_i \) and \( u_{i+1} \). If \( u_{i+1} = 0 \), we stop and say that \( u_i \) is the greatest common divisor of \( u_0 \) and \( u_1 \). Figure 1 shows the individual steps of the algorithm.

We use Euclid’s algorithm to find the greatest common divisor of 30 and 12. We know that \( 30 = 2\cdot 12 + 6 \), so \( gcd(30, 12) = gcd(12, 6) \). The problem thus reduces to finding the greatest common divisor of 12 and 6. Since \( 12 = 2\cdot 6 + 0 \), we are done, and can say that \( gcd(30, 12) = 6 \). The process is illustrated in Figure 1.

We argue that the algorithm terminates. If it terminates, it will return the greatest common divisor of two numbers because each step of the algorithm preserves the greatest common divisor by Lemma 1. Notice that \( u_{i+1} < u_i \) for all \( i \geq 1 \) by construction. Therefore, in each step, Euclid’s algorithm will have to find the greatest common divisor of two integers that are smaller than the integers in the previous step. Since \( u_0 \) and \( u_1 \) are finite and none of the \( u_i \) will ever be negative, the algorithm will eventually terminate and produce the right answer.

### 1.3 Bit Complexity of Euclid’s Algorithm

Suppose Euclid’s algorithm terminates in step \( n \). We are interested in an upper bound on \( n \) (we ignore the trivial cases when \( u_0 \) or \( u_1 \) is zero). Notice that \( q_i \geq 1 \) for all \( i \in \{1, \ldots, n\} \), so

\[
\begin{align*}
u_0 &\geq u_1 + u_2, \\
u_1 &\geq u_2 + u_3, \\
u_2 &\geq u_3 + u_4, \\
\vdots \\
u_{n-2} &\geq u_{n-3} + u_{n-2}, \\
\end{align*}
\]

and

\[
\begin{align*}
u_{n-1} &\geq u_n, \\
u_n &\geq 1.
\end{align*}
\]

We also know that \( u_n \geq 1 \), so

\[
\begin{align*}
u_{n-1} &\geq u_n \geq 1, \\
u_{n-2} &\geq u_{n-1} + u_n \geq 1 + 1 = 2, \\
u_{n-3} &\geq u_{n-2} + u_{n-1} \geq 2 + 1 = 3, \\
\vdots \\
u_0 &\geq u_1 + u_2 \geq F_n \text{ where } F_n \text{ is the } n\text{-th Fibonacci number}
\end{align*}
\]

We know that \( F_n \) can be approximated by \( c \cdot \left( \frac{1 + \sqrt{5}}{2} \right)^n \), so

\[
\log u_0 \geq \log \left[ c \cdot \left( \frac{1 + \sqrt{5}}{2} \right)^n \right] = \log c + n \log \left( \frac{1 + \sqrt{5}}{2} \right).
\]
This implies that the number of steps Euclid’s algorithm will have to perform, \( n \), is in \( O(\log u_0) \). This result is known as Lamé’s Theorem.

Since division in each step of the algorithm will take \( O((\log u_0)^2) \) bit operations using the standard arithmetic algorithm, we get that Euclid’s algorithm takes \( O((\log u_0)^3) \) bit operations as a corollary of Lamé’s Theorem. The analysis can be improved, however.

**Theorem 1** (Collins). The bit complexity of Euclid’s algorithm is \( O((\log u_0)(\log u_1)) \).

**Proof:** Step \( i \) of Euclid’s algorithm costs \( O((\log q_i)(\log u_i)) \) bit operations because that’s the complexity of division using the standard arithmetic algorithm and because division is the most expensive operation in each step. Therefore, the bit complexity of Euclid’s algorithm is

\[
O\left(\sum_{i=1}^{n} (\log q_i)(\log u_i)\right).
\]

We manipulate the expression inside the summation. Recall that \( \log x = \lceil \log_2 x \rceil + 1 \).

\[
\sum_{i=1}^{n} (\log q_i)(\log u_i) \leq (\log u_1) \sum_{i=1}^{n} (\log q_i) = (\log u_1) \sum_{i=1}^{n} (\lceil \log_2 q_i \rceil + 1) \leq (\log u_1) \sum_{i=1}^{n} \log_2 q_i + n \cdot (\log u_1)
\]

Since \( u_{i+1} \geq q_i u_i \), we have

\[
u_0 \geq q_1 u_1 \geq q_1 q_2 u_2 \geq \cdots \geq \prod_{i=1}^{n} q_i, \quad \text{so} \quad \sum_{i=1}^{n} \log_2 q_i \leq \log_2 u_0 \leq \log u_0.
\]

By Lamé’s Theorem, we also have \( n = O(\log u_0) \). Therefore,

\[
(\log u_1) \sum_{i=1}^{n} \log_2 q_i + n \cdot (\log u_1) = O((\log u_1)(\log u_0)) + O((\log u_0)(\log u_1)) = O((\log u_0)(\log u_1)),
\]

which is what we wanted. \( \square \)

### 1.4 Closing Remarks

Arnold Schönhage proved that the greatest common divisor of \( u_0 \) and \( u_1 \) could be found using \( O((\log u_0)^{1+\epsilon}) \) bit operations.

It is an open problem whether it is possible to compute the greatest common divisor of two \( l \)-bit numbers using a circuit with \( O(l^k) \) gates of fan-in 2 and depth \( O((\log l)^{k'}) \). The class of problems with such circuits is known as \( \text{NC} \). We know that addition, subtraction, multiplication and division are in \( \text{NC} \). The gate number versus circuit depth tradeoffs for circuits computing the greatest common divisor known today are not good enough, and any improvement in this direction would be a major result.

### 2 Next Time

In the next lecture we will discuss how to solve the simplest diophantine equation \( ax + by = c \) in integers, and how to find multiplicative inverses mod \( n \).