# Computer Algebra

Lecture Notes

based on a lecture by Prof. G. Kemper

Marvin Jahn, Leonhard Lampart
mail@marvin-jahn.de, leonhard@lampart.eu

February 17, 2022

# Contents

These are (unofficial) lecture notes for the lecture *Computer Algebra* held by Prof. G. Kemper at the Technical University Munich in the winter semester 2021/22.

# 1 Integer Arithmetic

Lec 1
2021-10-19

## 1.1 Addition of Integers

Any natural number $x \in \mathbb{N} \coloneqq \{0, 1, 2, \ldots\}$ can be written with respect to a fixed basis $B \in \mathbb{N}_{>1}$ as

$$x = \sum_{i=0}^{n-1} a_i \cdot B^i \text{ with } 0 \le a_i < B \text{ and } n \in \mathbb{N}.$$

We will mainly use *binary representation* $B = 2$, which plays a major role in computer science.

**Definition 1.1.** The **length** $l(x)$ of $x \in \mathbb{Z}$ w.r.t. basis $B \in \mathbb{N}_{>1}$ is defined to be

$$l(x) \coloneqq \begin{cases} \lfloor \log_B(|x|) \rfloor + 1 & \text{if } x \neq 0 \\ 1 & \text{if } x = 0 \end{cases}.$$

If $x \neq 0$, we may choose $n \in \mathbb{N}$ such that $a_{n-1} \neq 0$ and in that case, $l(x) = n$. The length of $x = \operatorname{sign}(x) \cdot |x|$ may be thought of as the amount of storage required to save $x$ on a computer with respect to the basis $B$. Technically, negative numbers require an additional bit for the minus sign, but we ignore this, since a single bit does not matter in the following arguments.

We start with the naive algorithm for addition like it is taught in elementary school.

**Algorithm 1.2 (Addition of natural numbers).**

Input: $x = \sum_{i=0}^{n-1} a_i B^i$, $y = \sum_{i=0}^{n-1} b_i B^i \in \mathbb{N}$.

Output: $z = \sum_{i=0}^{n} c_i B^i = x + y$.

```
(1) Set $\delta \coloneqq 0$.
(2) For $i = 0, \ldots, n-1$:
    (3) Let $c_i \coloneqq a_i + b_i + \delta$.
    (4) Set $\delta \coloneqq 0$.
    (5) If $c_i \ge B$: set $c_i \coloneqq c_i - B$ and $\delta \coloneqq 1$.
(6) Define $c_n \coloneqq \delta$.
```

**Definition 1.3** (informal)**.** A **bit-operation** is an operation that can be realized by one logic gate (*and*, *or*, *not*, *xor*) or by reading a bit from or writing a bit to memory.

For example, Algorithm 1.2 needs $10n + 2$ bit-operations.

**Definition 1.4.** Let $M$ be a set and $f, g : M \to \mathbb{R}_{>0}$ two functions. We write $f \in \mathcal{O}(g)$ if there exists $c \in \mathbb{R}$, such that $f(x) \le c \cdot g(x)$ for all $x \in M$.

Since we assumed that our functions map to the positive real numbers $\mathbb{R}_{>0}$, this is equivalent to the usual definition, where one requires $f(x) \le c \cdot g(x)$ to hold for all but finitely many $x \in M$.

**Theorem 1.5.** Let

$$f\colon \mathbb{N}_{>0} \to \mathbb{R},\ n \mapsto \max\bigl\{\text{number of bit-operations required for addition of } x, y \in \mathbb{N} \text{ with } l(x) \le n, l(y) \le n\bigr\}$$

and

$$\iota\colon \mathbb{N}_{>0} \to \mathbb{R},\ n \mapsto n$$

the canonical inclusion. Then $f \in \mathcal{O}(\iota)$; i.e. Algorithm 1.2 requires $\mathcal{O}(n)$ bit-operations. We say that it has *linear complexity.*

It is clear that this upper bound is actually optimal, since any algorithm for addition of two numbers needs to write the result, which already has linear complexity.

From now on, we will be less precise and omit such a $\max\{\ldots\}$ term, even though this will always be what we mean when we investigate the complexity of an algorithm.

Subtraction of two natural numbers $x = \sum_{i=0}^{n-1} a_i B^i$, $y = \sum_{i=0}^{n-1} b_i B^i$ with $x \ge y$ can be reduced to addition by the following trick: Define the *complement* $\bar{y}$ of $y$ as follows:

$$\bar{y} \coloneqq \sum_{i=0}^{n-1} (B - 1 - b_i) B^i = \sum_{i=0}^{n-1} (B-1) B^i - y = B^n - 1 - y$$

Therefore, $x - y = x + \bar{y} + 1 - B^n$ is obtained from $x + \bar{y}$ by starting Algorithm 1.2 with $\delta = 1$ and removing one from the coefficient of $B^n$.
In particular, subtraction of two natural numbers has linear complexity. This also means that addition and subtraction of whole numbers has linear complexity.

## 1.2 Multiplication of Integers

### 1.2.1 Grid Multiplication

From now on, we represent all our natural numbers in binary ($B = 2$). We start with the grid multiplication algorithm from elementary school.

**Algorithm 1.6 (Grid Multiplication).**

Input: $x = \sum_{i=0}^{n-1} a_i 2^i$, $y = \sum_{j=0}^{m-1} b_j 2^j \in \mathbb{N}$.

Output: $z = x \cdot y$.

```
(1) Let $z \coloneqq 0$.
(2) For $i = 0, \ldots, n-1$:
    If $a_i = 1$: $z \coloneqq z + \sum_{j=0}^{m-1} b_j 2^{i+j}$.
```

**Theorem 1.7.** Grid multiplication of numbers of length $n$ and $m$ (using Algorithm 1.6) requires $\mathcal{O}(n \cdot m)$ bit-operations (here the number of bit-operations is a function $\mathbb{N}_{>0} \times \mathbb{N}_{>0} \to \mathbb{R}$).
As a function of the total length of input $n + m$, it has *quadratic complexity* $\mathcal{O}((n+m)^2)$ since $n \cdot m \le \frac{(n+m)^2}{2}$.

### 1.2.2 Karatsuba-Multiplication

While we saw that the naive algorithm for addition (Algorithm 1.2) already has the best possible complexity, this is not the case for the naive algorithm for multiplication (Algorithm 1.6). An improvement is the *Karatsuba multiplication* algorithm.

We make the following observation: For two polynomials $ax+b, cx+d$ of degree 1, one can write their product as

$$(ax+b)(cx+d) = acx^2 + (ac+bd-(a-b)(c-d))x + bd, \tag{$*$}$$

which only uses three distinct multiplications instead of four. This leads to the following idea:

- Specialize $x = B$ for a very big basis $B \in \mathbb{N}$.
- Use recursive calls for the three multiplications.

**Algorithm 1.8** (**Karatsuba-Multiplication**).

Input: Natural numbers $x, y \in \mathbb{N}$.

Output: $z = x \cdot y$.

(1) Let $k \in \mathbb{N}$ minimal, such that $l(x), l(y) \leq 2^k$.

(2) If $k = 0$: Return $x \cdot y$ via a single *and* operation.

(3) Set $B := 2^{2^{k-1}}$ and write $x = x_0 + x_1 B$, $y = y_0 + y_1 B$ with $0 \leq x_i, y_i < B$.

(4) Compute $x_0 \cdot y_0, x_1 \cdot y_1$ and $(x_0 - x_1) \cdot (y_0 - y_1)$ by a recursive application of this algorithm.

(5) Return $z = x_0 y_0 + (x_0 y_0 + x_1 y_1 - (x_0 - x_1)(y_0 - y_1))B + x_1 y_1 B^2$.

*Proof (of correctness).* By definition of $k \in \mathbb{N}$, if $k = 0$, then $x$ and $y$ are only a single bit each, so multiplication is indeed a single *and* operation. If $k > 0$, then we have $2^{2^{k-1}} \leq x, y < 2^{2^k}$ and therefore $x$ and $y$ can be written as in step (3).
The numbers that are multiplied in step (4) are less than $B$, so $k$ decreases with each recursive application and the algorithm must terminate.
The correctness of the result then follows from $(*)$. □

Lec 2
2021-10-21

**Theorem 1.9.** For any two natural numbers $x, y \in \mathbb{N}$ with $l(x), l(y) \leq n$, Karatsuba-multiplication (Algorithm 1.8) requires $\mathcal{O}(n^{\log_2(3)})$ bit-operations.

*Proof.* Consider

$$\Theta(k) := \max\{\text{number of bit-operations required for } x \cdot y \colon x, y \in \mathbb{N} \text{ with } l(x), l(y) \leq 2^k\}.$$

If $k \geq 1$, then

$$\Theta(k) \leq 3\Theta(k-1) + C \cdot 2^k \tag{$*$}$$

for some constant $C > 0$, since there are three recursive calls and the additions, subtractions and the rewriting in step (3) require $\mathcal{O}(2^k)$ bit-operations.
*Claim:* $\Theta(k) \leq 3^k + 2C(3^k - 2^k)$ for all $k \in \mathbb{N}$.
We prove this by induction on $k$.

$k = 0$: This is true, because $\Theta(0) = 1$.

$k-1 \to k$: The claim follows from the calculation

$$\begin{aligned}\Theta(k) &\overset{(*)}{\leq} 3\Theta(k-1) + C \cdot 2^k \\ &\leq 3 \cdot (3^{k-1} + 2C(3^{k-1} - 2^{k-1})) + C \cdot 2^k \\ &= 3^k + 2C(3^k - 2^k).\end{aligned}$$

For given $n \in \mathbb{N}$, choose $k \in \mathbb{N}$ minimal such that $n \leq 2^k$. Then $2^{k-1} < n$ and it follows $k - 1 < \log_2(n)$. Using the claim, we conclude

$$\Theta(k) \leq (1+2C)3^k \leq 3(1+2C)3^{\log_2(n)} = 3(2C+1) \cdot 2^{\log_2(3)\cdot\log_2(n)} = 3(2C+1)n^{\log_2(3)}.$$

Since $n \leq 2^k$, $\Theta(k)$ is an upper bound for

$$\tau(n) \coloneqq \max\{\text{number of bit-operations required for } x \cdot y \colon x, y \in \mathbb{N} \text{ with } l(x), l(y) \leq n\}$$

and thus $\tau(n) \in \mathcal{O}(n^{\log_2(3)})$. □

Because $\log_2(3) \approx 1.59 < 2$, this shows that Karatsuba-multiplication (Algorithm 1.8) provides a substantial improvement over grid multiplication (Algorithm 1.6).

### 1.2.3 Discrete Fourier Transform

We investigate another way to calculate the product of two integers in an efficient way and start with some seemingly unrelated notions from analysis.

For a function $f \colon \mathbb{R} \to \mathbb{C}$, the **Fourier Transform** of $f$ (if it exists) is defined to be

$$\hat{f} \colon \mathbb{R} \to \mathbb{C}, \ \omega \mapsto \int_{-\infty}^{\infty} f(t) \cdot e^{i\omega t} dt.$$

The **convolution** $f * g$ of two functions $f, g \colon \mathbb{R} \to \mathbb{C}$ is the function

$$(f * g) \colon \mathbb{R} \to \mathbb{C}, \ x \mapsto \int_{-\infty}^{\infty} f(t) \cdot g(x-t) dt.$$

Under suitable conditions, the **convolution rule** holds true and states that

$$\widehat{f * g} = \hat{f} \cdot \hat{g}.$$

Polynomial multiplication $f \cdot g$ of two polynomials $f, g \in \mathbb{C}[x]$ can be seen as a special case of convolution with respect to the measure $\mu \coloneqq \sum_{i=0}^{\infty} \mathbb{1}_{\{i\}}$, where $\mathbb{1}_{\{i\}}$ is the *Dirac measure* with $\mathbb{1}_{\{i\}}(A) = 1$ if $i \in A$ and $\mathbb{1}_{\{i\}}(A) = 0$ otherwise. For this, we notice that a polynomial $f = \sum_{i=0}^{m} a_i x^i$ induces the function

$$\mathbb{R} \to \mathbb{C}, \ r \mapsto \begin{cases} a_r & \text{if } r \in \mathbb{N} \\ 0 & \text{otherwise} \end{cases}.$$

This gives an injective $\mathbb{C}$-linear map $\mathbb{C}[x] \hookrightarrow \mathbb{C}^{\mathbb{R}}$. Then for $f = \sum_{i=0}^{m} a_i x^i$, $g = \sum_{i=0}^{n} b_i x^i \in \mathbb{C}[x]$, we have $(f * g)(r) = 0$ if $r \notin \mathbb{N}$ and otherwise

$$(f * g)(r) = \int_{t \in \mathbb{N}} a_t b_{r-t} d\mu = \sum_{i=0}^{\infty} \int_{t \in \mathbb{N}} a_t b_{r-t} d\mathbb{1}_{\{i\}} = \sum_{i=0}^{\infty} a_i b_{r-i}.$$

This means that $f * g$ precisely corresponds to the polynomial $f \cdot g$.

The main idea of the discrete Fourier transform is to replace the integral by a sum and to change the exponential to a root of unity.

For a commutative ring $R$ (always with one), there is an $R$-module isomorphism

$$\bigoplus_{i=0}^{\infty} R \to R[x],$$

which restricts to an $R$-module isomorphism

$$R^n = \bigoplus_{i=0}^{n-1} R \to R[x]_{\leq n-1} \coloneqq \{f \in R[x] : \deg(f) \leq n-1\}.$$

for any $n \in \mathbb{N}_{>0}$. We can therefore identify $R^n$ with $R[x]_{\leq n-1}$.

**Definition 1.10.** Let $R$ be a commutative ring. An element $\mu \in R$ is called **$n$-th root of unity** if $\mu^n = 1$. It is called **primitive $n$-th root of unity** if additionally $\mu^k \neq 1$ for all $0 < k < n$.
The **discrete Fourier transform** with respect to $\mu$ is defined to be

$$\mathrm{DFT}_\mu \colon R^n \to R^n,\ (a_0, \dots, a_{n-1}) \mapsto (\hat{a}_0, \dots, \hat{a}_{n-1}),$$

where

$$\hat{a}_i \coloneqq \sum_{j=0}^{n-1} \mu^{i \cdot j} a_j.$$

Using our identification and $R[x] \twoheadrightarrow R[x]/(x^n - 1) \cong R[x]_{\leq n-1}$ (the isomorphism is induced by $R[x] \to R[x]_{\leq n-1}$, $\sum a_i x^i \mapsto \sum a_{i \bmod n} x^{i \bmod n}$) this amounts to the following for polynomials:

$$\mathrm{DFT}_\mu \colon R[x] \to R^n,\ f \mapsto (f(\mu^0), \dots, f(\mu^{n-1})).$$

The **convolution rule** holds true by the definition of polynomial multiplication:

$$\mathrm{DFT}_\mu(f \cdot g) = \mathrm{DFT}_\mu(f) \cdot \mathrm{DFT}_\mu(g).$$

The component-wise multiplication of two vectors with $n$ entries requires $\mathcal{O}(n)$ ring operations (not bit-operations!). However, $\mathrm{DFT}_\mu$ requires $\mathcal{O}(n^2)$ ring operations for polynomials of degree $< n$. Luckily, the *Fast Fourier transform* provides a faster way to compute the discrete Fourier transform of a polynomial.

**Algorithm 1.11** (**Fast Fourier Transformation**).

Input: $f \in R[x], \mu$ a $2^k$-th root of unity such that $\mu^{2^{k-1}} = -1$.

Output: $\mathrm{DFT}_\mu(f)$

(1) Write $f(x) = g(x^2) + x \cdot h(x^2)$ with $g, h \in R[x]$.

(2) If $k = 1$ (i.e. $\mu = -1$): return $(g(1) + h(1), g(1) - h(1))$.

(3) By a recursive call, compute $\hat{g} \coloneqq \mathrm{DFT}_{\mu^2}(g)$, $\hat{h} \coloneqq \mathrm{DFT}_{\mu^2}(h) \in R^{2^{k-1}}$.

(4) For $i = 0, \dots, 2^k - 1$: Set $\hat{f}_i := \hat{g}_i + \mu^i \hat{h}_i$, where $\hat{g}_i := \hat{g}_{i-2^{k-1}}$, $\hat{h}_i := \hat{h}_{i-2^{k-1}}$ and $\mu^i = -\mu^{i-2^{k-1}}$ for $i \geq 2^{k-1}$.

(5) Return $\hat{f} = (\hat{f}_0, \dots, \hat{f}_{2^k-1}) \in R^{2^k}$.

*Proof (of correctness).* If $k = 1$, then $\mu = -1$, so step (2) computes the "base case" correctly. By step (3), the $i$-th entries of $\hat{g}$ and $\hat{h}$ are $\hat{g}_i = g(\mu^{2i})$ and $\hat{h}_i = h(\mu^{2i})$, respectively. Therefore, $f(\mu^i) = \hat{g}_i + \mu^i \hat{h}_i = \hat{f}_i$ for all $i \in \{0, \dots, 2^{k-1} - 1\}$. For $i \in \{2^{k-1}, \dots, 2^k - 1\}$, we have

$$f(\mu^i) = f\left(-\mu^{i-2^{k-1}}\right) = g\left(\mu^{2(i-2^{k-1})}\right) - \mu^{i-2^{k-1}} h\left(\mu^{2(i-2^{k-1})}\right) = \hat{g}_{i-2^{k-1}} - \mu^{i-2^{k-1}} \hat{h}_{i-2^{k-1}},$$

so the algorithm returns the correct result. □

**Example 1.12.** Consider the primitive 4-th root of unity $\mu = e^{i\pi/2} = i \in \mathbb{C}$ and $f = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \in \mathbb{C}[x]$. Applying the Fast Fourier Transform (Algorithm 1.11) yields

$$g = a_0 + a_2 x, \quad h = a_1 + a_3 x, \quad \hat{g} = (a_0 + a_2, a_0 - a_2), \quad \hat{h} = (a_1 + a_3, a_1 - a_3)$$

and therefore

$$\hat{f} = (a_0 + a_2 + a_1 + a_3, a_0 - a_2 + i(a_1 - a_3), a_0 + a_2 - (a_1 + a_3), a_0 - a_2 - i(a_1 - a_3)).$$

**Theorem 1.13.** Let $n = 2^k$, $f \in R[x]$ a polynomial with $\deg f < n$ and $\mu \in R$ a primitive $n$-th root of unity with $\mu^{2^{k-1}} = -1$. Then Fast Fourier Transform (Algorithm 1.11) requires $\mathcal{O}(n \cdot \log_2(n))$ ring operations.

*Proof.* Let

$$\Theta(k) := \max\{\text{number of ring operations required for } f \in R[x], \deg f < 2^k\}.$$

For $k \geq 2$, we have

$$\Theta(k) \leq 2\Theta(k-1) + \underbrace{2^{k-1}}_{\text{powers of } \mu} + \underbrace{2^{k-1}}_{\text{products } \mu^i \hat{h}_i} + \underbrace{2^k}_{+/-} = 2\Theta(k-1) + 2^{k+1} \qquad (*)$$

*Claim*: $\Theta(k) \leq (2k-1) \cdot 2^k \ \forall \ k \geq 1$.
Proof by induction on $k$:

$k = 1$: This is true because $\Theta(1) = 2$.

$k - 1 \to k$: We calculate

$$\Theta(k) \overset{(*)}{\leq} 2\Theta(k-1) + 2^{k+1} \leq 2(2(k-1)-1) \cdot 2^{k-1} + 2^{k+1} \leq (2k-1) \cdot 2^k.$$

With $k = \log_2(n)$, it follows $\Theta(k) \leq (2 \cdot \log_2(n) - 1)n \in \mathcal{O}(n \cdot \log_2(n))$. □

We introduce the following ad-hoc terminology.

**Definition 1.14.** A primitive $n$-th root of unity is called **good primitive $n$-th root of unity** if for all $0 < i < n$, we have $\sum_{j=0}^{n-1} \mu^{ij} = 0$.

Equivalently, we may ask that $\mathrm{DFT}_\mu((1,\dots,1)) = (n,0,\dots,0)$.

Another equivalent characterization is to demand that $\sum_{j=0}^{n-1} \mu^{ij} = 0$ for all $i \in \mathbb{Z}$ with $n \nmid i$. This follows by writing $i$ as $i = k \cdot n + r$ with $k \in \mathbb{Z}$ and $r \in \{1,\dots,n-1\}$ and calculating Lec 3 2021-10-26

$$\sum_{j=0}^{n-1} \mu^{ij} = \sum_{j=0}^{n-1} \mu^{(kn+r)j} = \sum_{j=0}^{n-1} \mu^{knj} \cdot \mu^{rj} = \sum_{j=0}^{n-1} \mu^{rj} = 0.$$

**Example 1.15.** (1) $e^{2\pi i/n} \in \mathbb{C}$ is a good primitive $n$-th root of unity.

(2) $\mu = \bar{3} \in \mathbb{Z}/(8)$ is a primitive 2-nd root of unity, but it is not good, as $\bar{1} + \bar{3} = \bar{4} \neq \bar{0}$ shows.

The second example also demonstrates that our assumption in the previous algorithms that $\mu^{2^{k-1}} = -1$ for a primitive $2^k$-th root of unity $\mu$ is really necessary.

The following proposition shows that for good roots $\mu$, the reverse of the discrete Fourier transform $\mathrm{DFT}_\mu$ exists and is up to a scalar given by another discrete Fourier transform $\mathrm{DFT}_{\mu^{-1}}$.

**Proposition 1.16.** Let $\mu \in R$ be a good primitive $n$-th root of unity and let $a \in R^n$. Then

$$\mathrm{DFT}_{\mu^{-1}}(\mathrm{DFT}_\mu(a)) = n \cdot a,$$

where $n$ denotes the sum of $n$ ones in $R$.

*Proof.* The $i$-th component of $\mathrm{DFT}_{\mu^{-1}}(\mathrm{DFT}_\mu(a))$ is

$$\sum_{j=0}^{n-1} \mu^{-ij}\hat{a}_j = \sum_{j=0}^{n-1} \mu^{-ij} \sum_{k=0}^{n-1} \mu^{jk} a_k = \sum_{k=0}^{n-1} a_k \left( \sum_{j=0}^{n-1} \mu^{(k-i)j} \right) = n \cdot a_i. \qquad \square$$

The next proposition shows that in some common situations, there are "many" good $n$-th roots of unity.

**Proposition 1.17.**

(a) If $R$ is an integral domain, then all primitive $n$-th roots of unity are good.

(b) Let $R$ be a commutative ring with $\mathrm{char}(R) \neq 2$, $\mu \in R$ and $n \in \mathbb{N}_{>0}$. If $n = 2^k$ for $k \in \mathbb{N}_{>0}$ and $\mu^{n/2} = -1$ ("halfway property"), then $\mu$ is a good primitive $n$-th root of unity.

*Proof.* (a) Let $\mu \in R$ be a primitive $n$-th root of unity and $0 < i < n$. Since $0 = \mu^{in} - 1 = (\mu^i - 1)(\sum_{j=0}^{n-1} \mu^{ij})$ and $R$ is an integral domain, it follows $\sum_{j=0}^{n-1} \mu^{ij} = 0$.

(b) The halfway property directly implies $\mu^n = 1$ and $\mathrm{ord}(\mu) \mid 2^k$ shows that $\mathrm{ord}(\mu) = 2^l$ for some $l \in \mathbb{N}$. Because $\mu^{2^{k-1}} = -1$ and $-1 \neq 1$ as $\mathrm{char}(R) \neq 2$, it follows that $\mu$ is a primitive $n$-th root of unity.
To establish that $\mu$ is good, we write $i = r \cdot 2^{k-s} \in \{1,\dots,n-1\}$ with $r \in \mathbb{N}$ odd and $s \in \mathbb{N}_{>0}$. Since $\tilde{\mu} := \mu^i$ is a primitive $2^s$-th root of unity with the "halfway property" and $\sum_{j=0}^{n-1} \mu^{ij} = \sum_{j=0}^{n-1} \tilde{\mu}^j$, it is sufficient to check the case $i = 1$:

$$\sum_{j=0}^{n-1} \mu^j = \sum_{j=0}^{n/2-1} (\mu^j + \underbrace{\mu^{j+n/2}}_{=-\mu^j}) = 0. \qquad \square$$

For example, in $\mathbb{C}$ any primitive $n$-th root of unity $\mu$ is good. This makes sense intuitively, as the points $1, \mu, \dots, \mu^{n-1}$ form a regular polygon.

We can now use the discrete Fourier transform in order to calculate the product of two polynomials.

**Algorithm 1.18 (Polynomial multiplication with FFT).**
Let $R$ be a commutative ring in which $2 \in R$ is invertible.

Input: $f, g \in R[x]$, $\deg f + \deg g < 2^k =: n$, $\mu \in R$ such that $\mu^{n/2} = -1$.

Output: $h = f \cdot g$.

(1) Compute $\hat{f} := \mathrm{DFT}_\mu(f)$ and $\hat{g} := \mathrm{DFT}_\mu(g) \in R^n$ using a Fast Fourier transform (Algorithm 1.11).

(2) Calculate the component-wise product $\hat{h} = \hat{f} \cdot \hat{g}$.

(3) Compute $(h_0, \dots, h_{n-1}) := \frac{1}{n}\mathrm{DFT}_{\mu^{-1}}(\hat{h})$.

(4) Return $h = \sum_{i=0}^{n-1} h_i x^i$.

*Proof (of correctness).* That the algorithm computes the desired product is a direct consequence of Proposition 1.16 and Proposition 1.17. □

By Theorem 1.13, this algorithm requires $\mathcal{O}(n \cdot \log_2(n))$ ring operations.

We face the following problem: As $n$ gets larger, computations with a $n$-th root of unity will generally become harder. To solve this problem, we will work in a ring where we always have a particular "nice" good root of unity. Namely, we consider the ring $R := \mathbb{Z}/(m)$ with $m := 2^l + 1$ for some $l \in \mathbb{N}$. Thus by definition, $\bar{2}^l = \overline{-1}$, so $\bar{2} \in R$ is a good $2l$-th root of unity.

When we work in such a ring $\mathbb{Z}/(m)$ with $m := 2^l + 1$, we always represent our elements by their unique representative in $\{0, \dots, m-1\}$, which has a maximal length of $l+1$. It is important that our algorithms return another representative in $\{0, \dots, m-1\}$. A number $x \leq 2^{2l}$ can be reduced to a representative between 0 and $m-1$ with $\mathcal{O}(l)$ bit-operations:
Indeed, if $x = 2^{2l}$, then $\bar{x} = \overline{-1}^2 = \bar{1}$, so the result is 1. Otherwise, we consider the binary representation of $x$, take the first $l$ bits from the "right side" and interpret them as a new number $r$ and interpret the remaining bits as another number $y$. In other words, we write $x = 2^l \cdot y + r$ with $y, r < 2^l$. Because $\bar{2}^l = \overline{-1}$, $\bar{x}$ is the same as $\overline{r - y}$ and this subtraction requires $\mathcal{O}(l)$ bit-operations. Now $r - y$ might be negative, but it is certainly greater than $-2^l$, so if it is negative, we may simply add $2^l + 1$ to it, in order to obtain the desired representative between 0 and $m-1$.

**Proposition 1.19.** Let $R = \mathbb{Z}/(m)$ with $m = 2^l + 1$ for some $l \in \mathbb{N}$. Then addition in $R$ and multiplication by $\bar{2}^i \in R$ for $0 \leq i < 2l$ requires $\mathcal{O}(l)$ bit-operations.

*Proof.* For $x, y \in \{0, \dots, m-1\}$, the sum of representatives $x + y$ is computable in $\mathcal{O}(l)$ and if that sum exceeds $m$, subtracting $m$ also needs $\mathcal{O}(l)$ bit-operations. Together, this requires $\mathcal{O}(l)$ bit-operations.
We first consider multiplication by $2^i$ for $0 \leq i < l$, which amounts to shifting the binary representation $i$ places to the left. This requires $\mathcal{O}(l) + \mathcal{O}(g(i)) = \mathcal{O}(l)$ bit-operations, where $g$ denotes the cost of computing the starting address of the numbers.

The resulting number $x$ can be at most $2^{l-1} \cdot 2^l = 2^{2l-1}$, so the previous discussion shows that it can be computed with $\mathcal{O}(l)$ bit-operations.
Finally, if $i \geq l$, then $2^i = 2^l \cdot 2^{i-l} = -2^{i-l}$, so multiplying by $2^i$ amounts to multiplying by $2^{i-l}$ first and then negating the result. Since the first step requires $\mathcal{O}(l)$ bit-operations by the above and negating the representative is just another subtraction, the claim follows. □

**Proposition 1.20.** Let $k, r \in \mathbb{N}$, $r > 0$, $m := 2^{2^k \cdot r} + 1$, $R := \mathbb{Z}/(m)$ and $\mu = \bar{2}^r \in R$. Then $\mu$ is a good primitive $2^{k+1}$-th root of unity and $\bar{2} \in R$ is invertible.

*Proof.* By definition, $\mu^{2^k} = \overline{-1}$, so Proposition 1.17 shows the first part of the claim. For the second part, it suffices to notice that $m \nmid 2$ and in a finite ring, every element that is not a zero divisor is necessarily a unit. □

Using the above theory, we can now state another efficient algorithm that computes the product of two natural numbers.

**Algorithm 1.21** (**Variant of the Schönhage-Strassen algorithm**).

Input: $x, y \in \mathbb{N}$.

Output: $z = x \cdot y$.

(1) Choose $k \in \mathbb{N}$ minimal such that $l(x), l(y) \leq 2^{2k}$.

(2) If $k \leq 3$, compute $z = x \cdot y$ using grid multiplication (Algorithm 1.6).

(3) Set $B := 2^{2^k}$, write $x = \sum_{i=0}^{2^k-1} x_i B^i$ and $y = \sum_{i=0}^{2^k-1} y_i B^i$ with $x_i, y_i \in \mathbb{N}_{<B}$.

(4) Set $m := 2^{4 \cdot 2^k} + 1$, $R := \mathbb{Z}/(m)$ and $\mu = \bar{2}^4 = \overline{16}$.

(5) Compute

$$\hat{x} := \mathrm{DFT}_\mu\big(\bar{x}_0, \ldots, \bar{x}_{2^k-1}, \underbrace{0, \ldots, 0}_{2^k \text{ zeros}}\big), \quad \hat{y} := \mathrm{DFT}_\mu\big(\bar{y}_0, \ldots, \bar{y}_{2^k-1}, \underbrace{0, \ldots, 0}_{2^k \text{ zeros}}\big) \in R^{2^{k+1}}.$$

(6) Compute the component-wise product $\hat{z} := \hat{x} \cdot \hat{y} \in R^{2^{k+1}}$ by first taking the product of representatives in $\mathbb{N}_{<m}$ by a recursive call and then reducing the result modulo $m$ by subtracting high bits (just like in the proof of Proposition 1.19).

(7) Compute $(\bar{z}_0, \ldots, \bar{z}_{2^{k+1}-1}) := \frac{1}{2^{k+1}} \mathrm{DFT}_{\mu^{-1}}(\hat{z})$ with $z_i \in \mathbb{N}_{<m}$.

(8) Return $z := \sum_{i=0}^{2^{k+1}-1} z_i B^i$.

Lec 4
2021-10-28

*Proof (of correctness).* We first note that the rewrite in step (3) is possible, because

$$B^{2^k} = (2^{2^k})^{2^k} = 2^{2^k \cdot 2^k} = 2^{2^{2k}} > x, y.$$

Furthermore, by Proposition 1.20, $\mu$ is a good $2^{k+1}$-th root of unity, so by Proposition 1.16, we correctly compute the product $\bar{z} = \bar{x} \cdot \bar{y} \in R$.

To see that also $z = x \cdot y$ in $\mathbb{N}$, it is enough to notice that the $l$-th coefficient of $x \cdot y$ lies in $\{0, \ldots, m-1\}$:

$$\sum_{i+j=l} x_i y_j < 2^k \cdot B^2 = 2^k \cdot 2^{2 \cdot 2^k} = 2^{k+2^{k+1}} < m.$$

$\square$

**Theorem 1.22.** Algorithm 1.21 requires $\mathcal{O}(n \cdot \log_2(n)^4)$ bit-operations when applied to $x, y \in \mathbb{N}$ with $l(x), l(y) \leq n$.

*Proof.* Let $l := 4 \cdot 2^k$, so that $m = 2^l + 1$ and consider

$$\theta(k) := \max\{\text{number of bit-operations required for } x, y \in \mathbb{N} \colon l(x), l(y) \leq 2^{2k}\},$$

where we extend $\theta$ to $\mathbb{R}_{>0}$ by setting $\theta(x) := \theta(\lfloor x \rfloor)$ for $x \in \mathbb{R}_{>0}$.
We start by analyzing the bit-operation costs for the different steps.

(1) $\max\{l(x), l(y)\}$ can be calculated with $\mathcal{O}(n)$ bit-operations and $2^{2k} = 4^k$ can be obtained from $4^{k-1}$ by shifting the representing bits two places to the left. Therefore, we need at most $\mathcal{O}(n) + \lceil \log_4(n) \rceil \cdot \mathcal{O}(n) \leq \mathcal{O}(k \cdot 2^{2k})$ bit-operations.

(2) Since this step only runs for small numbers, it requires $\mathcal{O}(1)$ bit-operations.

(3) This step starts by dividing the binary representations of $x$ and $y$ into blocks of size $2^k$. At most we need to perform $\frac{n}{2^k} \leq 2^k$ such splits for $x$ and $y$, respectively. Additionally, we might need to move the location of the split numbers in memory. Because they have a length of at most $2^k$, this needs $\mathcal{O}(2^k)$ bit-operations. In total, this amounts to $\mathcal{O}(2^k \cdot 2^k) = \mathcal{O}(2^{2k})$ bit-operations.

(4) To save $m$ to memory, we need to set its first and $(4 \cdot 2^k + 1)$-st bit to 1, which is certainly in $\mathcal{O}(4 \cdot 2^k + 1) = \mathcal{O}(2^k)$. Computing $\mu$ needs $\mathcal{O}(1)$ bit-operations.

(5) By Theorem 1.13, the Fast Fourier Transform (Algorithm 1.11) requires $\mathcal{O}(2^{k+1}(k+1))$ ring operations, which are additions and multiplications by $\mu^i = \bar{2}^{4i}$ with $0 \leq i < 2^{k+1}$. Proposition 1.19 states that these ring operations correspond to $\mathcal{O}(2^k)$ bit-operations. In total, this amounts to $\mathcal{O}(k \cdot 2^{2k})$ bit-operations.

(6) We need to perform $2^{k+1}$ multiplications of numbers with a length of at most $4 \cdot 2^k + 1$. By a previous discussion, reducing a number that is less than or equal to $2^{2l}$ modulo $m$ needs $\mathcal{O}(l) = \mathcal{O}(2^k)$ bit-operations. Thus in total, we perform $\mathcal{O}(2^{k+1} \cdot 2^k) = \mathcal{O}(2^{2k})$ bit-operations to reduce all the results.
We now consider the recursive calls that are used to compute the products. Writing $k'$ for the $k$ occurring in the recursive calls, we have $x_i, y_i < m$, so $l(x_i), l(y_i) \leq 4 \cdot 2^k = 2^{k+2}$ and thus $2k' \leq k+3$. Since we make $2^{k+1}$ recursive calls, this amounts to $2^{k+1} \cdot \theta(\frac{k+3}{2})$ bit-operations for the multiplications.
In total, step (6) requires $2^{k+1} \cdot \theta(\frac{k+3}{2}) + \mathcal{O}(2^{2k})$ bit-operations.

(7) This step also requires $\mathcal{O}(k \cdot 2^{2k})$ bit-operations and the argument is similar to that of step (5): By Theorem 1.13, computing $\mathrm{DFT}_{\mu^{-1}}$ requires $\mathcal{O}(2^{k+1}(k+1))$ ring operations, which are additions and multiplications by $\mu^{-i} = \bar{2}^{-4i}$ with $0 \leq i < 2^{k+1}$. Division by $\bar{2}^{k+1}$ is multiplication by $\bar{2}^{2^{k+3}-k-1}$ and by Proposition 1.19, this takes $\mathcal{O}(4 \cdot 2^k) = \mathcal{O}(2^k)$ bit-operations. Hence, step (7) requires $\mathcal{O}(2^{2k})$ bit-operations.

(8) Since

$$\frac{a}{b} < \frac{a+1}{b} \leq \frac{2b}{b+1}\frac{a+1}{b} = 2\frac{a+1}{b+1} \quad \forall\ a \in \mathbb{N},\ b \in \mathbb{N}_{>0},$$

we have for $j \leq 2^{k+1}$:

$$\sum_{i=0}^{j-1} z_i B^i \leq (m-1)\sum_{i=0}^{j-1} B^i = 2^{4\cdot 2^k}\frac{B^j - 1}{B-1} < 2 \cdot 2^{4\cdot 2^k}\frac{B^j}{B} = 2^{1+(4+j-1)2^k} = 2^{1+(j+3)2^k}.$$

Hence the length of that partial sum is at most $1+(j+3)2^k$ Because $z_j B^j$ starts at the $j{\cdot}2^k$-th bit and $l(z_j) \leq 4{\cdot}2^k$, if we add that $z_j B^j$ to the partial sum, we effectively add numbers of lengths $4 \cdot 2^k$ and $3 \cdot 2^k$, which requires $\mathcal{O}(2^k)$ bit-operations and thus $\mathcal{O}(2^{2k})$ in total.

We conclude that the following inequality holds for some constant $C \in \mathbb{R}$:

$$\theta(k) \leq 2^{k+1}\theta\left(\frac{k+3}{2}\right) + C \cdot k \cdot 2^{2k} \quad \forall\ k \in \mathbb{R},\ k \geq 4.$$

Consider

$$\Lambda(k) := \frac{\theta(k)}{2^{2k}} \quad \text{and} \quad \Omega(k) := \Lambda(k+3).$$

Then

$$\Lambda(k) \leq \frac{2^{k+1}\theta(\frac{k+3}{2})}{2^{2k}} + C \cdot k = \frac{16 \cdot \theta(\frac{k+3}{2})}{2^{k+3}} + C \cdot k = 16 \cdot \Lambda\left(\frac{k+3}{2}\right) + C \cdot k \quad \forall\ k \geq 4$$

and thus

$$\Omega(k) = \Lambda(k+3) \leq 16 \cdot \Lambda\left(\frac{k+6}{2}\right) + C \cdot (k+3) = 16 \cdot \Omega\left(\frac{k}{2}\right) + C \cdot (k+3) \quad \forall\ k \geq 1.$$

*Claim*: $\Lambda(k) \leq 16^i \Omega\left(\frac{k-3}{2^i}\right) + C \cdot (k-3) \cdot \sum_{j=0}^{i-1} 8^j + 3 \cdot C \cdot \sum_{j=0}^{i-1} 16^j$ for all $i \in \mathbb{N}$ with $2^{i-1} \leq k-3$, $k \geq 1$.
Proof by induction on $i$: If $i = 0$, then $\Lambda(k) \leq \Omega(k-3)$.
$i \to i+1$: We calculate

$$\begin{aligned}
\Lambda(k) &\leq 16^i \Omega\left(\frac{k-3}{2^i}\right) + C \cdot (k-3) \cdot \sum_{j=0}^{i-1} 8^j + 3 \cdot C \cdot \sum_{j=0}^{i-1} 16^j \\
&\leq 16^i \left(16 \cdot \Omega\left(\frac{k-3}{2^{i+1}}\right) + C \cdot \left(\frac{k-3}{2^i} + 3\right)\right) + C \cdot (k-3) \cdot \sum_{j=0}^{i-1} 8^j + 3 \cdot C \cdot \sum_{j=0}^{i-1} 16^j \\
&= 16^{i+1} \cdot \Omega\left(\frac{k-3}{2^{i+1}}\right) + C \cdot (k-3) \cdot \sum_{j=0}^{i} 8^j + 3 \cdot C \cdot \sum_{j=0}^{i} 16^j.
\end{aligned}$$

This finishes the induction proof. Now let $\nu \in \mathbb{N}$ be minimal, such that $2^\nu > k-3$. Then

$$D := \Omega(0) = \Omega\left(\left\lfloor \frac{k-3}{2^\nu} \right\rfloor\right) = \Omega\left(\frac{k-3}{2^\nu}\right).$$

Because $2^{\nu-1} \leq k-3$, the claim implies

$$\Lambda(k) \leq 16^{\nu} \cdot D + C \cdot \underbrace{(k-3)}_{<2^{\nu}} \frac{8^{\nu}-1}{7} + 3C\frac{16^{\nu}-1}{15} \in \mathcal{O}(16^{\nu}).$$

Since $\nu - 1 \leq \log_2(k-3)$, we obtain

$$\Lambda(k) \in \mathcal{O}(16^{\log_2(k-3)+1}) = \mathcal{O}(2^{4\cdot\log_2(k-3)}) = \mathcal{O}((k-3)^4),$$

so $\theta(k) = 2^{2k}\cdot\Lambda(k) \in \mathcal{O}(2^{2k}\cdot(k-3)^4)$. Finally, because $2^{2(k-1)} < n$, it follows $k-1 < \frac{\log_2(n)}{2}$ and we conclude $\theta(k) \in \mathcal{O}(n \cdot (\log_2(n))^4)$. □

We mention some modern algorithms that multiply two natural numbers of size $\leq n$: Lec 5 2021-11-02

- *Schönhage-Strassen* (1971): $\mathcal{O}(n \cdot \log_2(n) \cdot \log_2(\log_2(n)))$.
- *Fürer's algorithm* (2001): Asymptotically faster.
- *Murrey, von der Hoeven* (2021): $\mathcal{O}(n \cdot \log_2(n))$.

## 1.3 Division with Remainder and Greatest Common Divisors

We start with the straightforward algorithm and as before assume that our numbers are represented in binary.

**Algorithm 1.23** (**Division with remainder**).

Input: $b := \sum_{i=0}^{n-1} b_i 2^i$, $a := \sum_{i=0}^{n+m-1} a_i 2^i$ with $a_i, b_i \in \{0,1\}, b_{n-1} = 1$.

Output: $q, r \in \mathbb{N}$, such that $a = q\cdot b + r$, $0 \leq r < b$.

```
(1) Initialize $r := a$, $q := 0$.
(2) For $i = m, m-1, \ldots, 1, 0$:
    If $r \geq 2^i \cdot b$: $r := r - 2^i b$, $q := q + 2^i$
```

*Proof (of correctness).* It is clear that we have $a = q \cdot b + r$ after every iteration. Furthermore, we have

$$0 \leq r = a < 2^{n+m} = 2^{m+1} \cdot 2^{n-1} \leq 2^{m+1} \cdot b$$

after the initialization step, so by definition of step (2), we see that $0 \leq r < 2^j \cdot b$ is true after iteration step $i = j$. Therefore, the algorithm terminates when $0 \leq r < 2^0 b = b$. □

**Theorem 1.24.** Algorithm 1.23 requires $\mathcal{O}(n(m+1))$ bit-operations.

*Proof.* We iterate $(m+1)$-times and perform a multiplication by $2^i$ (bit-shift), a comparison and some additions, which are linear in the size $n+m$, so the claim follows. □

Integral domains that have a division with remainder are called **Euclidean rings**. Examples include $\mathbb{Z}$ (with grading given by the absolute value), the polynomial ring $\mathbb{K}[x]$ over a field $\mathbb{K}$ (with the degree as the grading) and the *Gaussian integers* $\mathbb{Z}[i] \subset \mathbb{C}$ (with the square of the absolute value as the grading). Note that the remainder $r$ is generally not unique, not even in $\mathbb{Z}$, as the following example with $a = 3$ and $b = 2$ shows:

$$1 \cdot 2 + 1 = 3 = 2 \cdot 2 - 1.$$

Another example is given by $a = x^2 \in \mathbb{K}[x]$ and $b = x + 1 \in \mathbb{K}[x]$:

$$x(x + 1) - x = x^2 = (x - 1)(x + 1) + 1$$

In the case of $\mathbb{Z}$, we can make the remainder $r$ unique by demanding that $0 \leq r < b$ as we did in the algorithm. Note that we may alternatively ask for $-\frac{b}{2} < r \leq \frac{b}{2}$ and Algorithm 1.23 can be easily modified to compute such a remainder $r$, because if we have $a = qb + r$ with $\frac{b}{2} < r < b$, then $r' := r - b$ satisfies $a = (q + 1)b + r'$ and $-\frac{b}{2} < r' < 0$. It is clear that Algorithm 1.23 extends to $a, b \in \mathbb{Z}$, $b \neq 0$ in a straightforward way.

In fact, division with remainder can be reduced to multiplication. More precisely: If two numbers of size $\leq n$ can be multiplied in $M(n)$ bit-operations, then division with remainder is possible in $\mathcal{O}(M(n))$ bit-operations.
We also mention that *Jebelean's Algorithm* (1997) with a runtime of $\mathcal{O}(n^{\log_2(3)})$ is of practical relevance.

The well-known Euclidean algorithm works in any Euclidean ring, but we just state it for $\mathbb{Z}$ and for simplicity restrict to the natural numbers. It is used to calculate the *greatest common divisor* $\gcd(a, b)$ of two elements $a, b$ in a Euclidean ring.
Note that the greatest common divisor is only unique up to multiplication with a unit. For example, in $\mathbb{Z}$, it is determined up to sign.

**Algorithm 1.25** (**Euclidean Algorithm**).

```
Input: $a, b \in \mathbb{N}$.
Output: $\gcd(a, b)$.
    (1) Initialize $r_0 := a$, $r_1 := b$.
    (2) For $i = 1, 2, 3, \ldots$:
        (3) If $r_i = 0$: Return $\gcd(a, b) = |r_{i-1}|$.
        (4) Compute division with remainder of $r_i$ by $r_{i-1}$ using (a slightly modified
            version of) Algorithm 1.23: $r_{i-1} = q_i r_i + r_{i+1}$ with $q_i \in \mathbb{Z}, r_{i+1} \in \mathbb{Z}$, $|r_{i+1}| \leq \frac{|r_i|}{2}$.
```

*Proof (of correctness).* Because the $r_i$ are always whole numbers and their absolute values are strictly decreasing, the algorithm must terminate. By step (4) we have for $x \in \mathbb{Z}$:

$$x \mid r_{i-1} \text{ and } x \mid r_i \iff x \mid r_{i+1} \text{ and } x \mid r_i.$$

Therefore, $\gcd(r_{i-1}, r_i) = \gcd(r_i, r_{i+1})$, so we conclude

$$\gcd(a, b) = \gcd(r_0, r_1) = \gcd(r_1, r_2) = \ldots = \gcd(r_k, 0) = |r_k|,$$

where $k$ is such that $r_{k+1} = 0$. □

**Theorem 1.26.** For two integers of lengths $n$ and $m$ respectively, Algorithm 1.25 requires $\mathcal{O}(n \cdot m)$ bit-operations.

*Proof.* Let $l(a) = n$, $l(b) = m$. If $a < b$, the first division with remainder will result in $r_2 = a$, thus we may assume that $a \geq b$.
Set $n_i = l(r_i)$; in particular, we have $n_0 = n$ and $n_1 = m$. For $i \geq 2$, we have $n_i \leq n_{i-1} - 1$, so by Theorem 1.24, the number of required bit-operations is bounded from above by

$$C \cdot \underbrace{\sum_{i=1}^{k} n_i \cdot (n_{i-1} - n_i + 1)}_{=:\sigma(n_0,\ldots,n_k)},$$

where $k$ denotes the number of divisions with remainder (i.e. $r_{k+1} = 0$) and $C$ is a constant.
We now consider the special case where we have $n_i = n_{i-1} - 1$ for all $i \geq 2$. In that case, it follows $n_i = n_1 - i + 1 = m - i + 1$ for all $i \geq 1$, implying $k = m$ and we see that the assertion holds for this special case:

$$\sigma(n_1, \ldots, n_k) = n_1(n_0 - n_1 + 1) + \sum_{i=2}^{m} (m - i + 1) \cdot 2 = m(n - m + 1) + m(m - 1) = m \cdot n.$$

*Claim*: The special case is the "worst case" of the algorithm; that is, it provides an upper bound for $\sigma(n_0, \ldots, n_k)$.
To prove this, let $n_0 \geq n_1 > n_2 > \cdots > n_k$ be arbitrary. We claim that we can obtain the special case from this sequence by successively inserting numbers into the "gaps". Indeed, if there is a $j \in \mathbb{N}$ with $n_{j-1} > n_j + 1$, we may insert a natural number $s$ with $n_j < s < n_{j-1}$ between $n_{j-1}$ and $n_j$ into the sequence. Because we have

$$\begin{aligned}
&\sigma(n_0, \ldots, n_{j-1}, s, n_j, \ldots, n_k) - \sigma(n_0, \ldots, n_k) \\
=&s(n_{j-1} - s + 1) + n_j(s - n_j + 1) - n_j(n_{j-1} - n_j + 1) \\
=&s + (n_{j-1} - s)s + n_j(s - n_{j-1}) = s + (n_{j-1} - s)(s - n_j) > 0,
\end{aligned}$$

this can only increase the number of required bit-operations. □

With its essentially quadratic complexity, computing greatest common divisors is a rather cheap operation.

Lec 6 2021-11-04

*Bezout's identity* states that for any two integers $a, b \in \mathbb{Z}$, there exist (not necessarily unique) $s, t \in \mathbb{Z}$, such that $as + bt = \gcd(a, b)$.
We now discuss a slight extension of Algorithm 1.25, which computes such $s, t$ in addition to the greatest common divisor.

**Algorithm 1.27** (**Extended Euclidean Algorithm**).

Input: $a, b \in \mathbb{N}$.

Output: $d = \gcd(a, b)$ and $s, t \in \mathbb{Z}$ such that $d = sa + tb$.

```
(1) Initialize $r_0 := a$, $r_1 := b$, $s_0 := 1$, $t_0 := 0$, $s_1 := 0$ and $t_1 := 1$.

(2) For $i = 1, 2, 3, \ldots$:
```

(3) If $r_i = 0$: Return $d \coloneqq |r_{i-1}|$, $s \coloneqq \operatorname{sign}(r_{i-1})s_{i-1}$ and $t \coloneqq \operatorname{sign}(r_{i-1})t_{i-1}$.

(4) Compute division with remainder of $r_{i-1}$ by $r_i$ using (a slightly modified version of) Algorithm 1.23: $r_{i-1} = q_i r_i + r_{i+1}$ with $q_i \in \mathbb{Z}, r_{i+1} \in \mathbb{Z}$, $|r_{i+1}| \leq \frac{|r_i|}{2}$.

(5) Set $s_{i+1} \coloneqq s_{i-1} - q_i s_i$ and $t_{i+1} \coloneqq t_{i-1} - q_i t_i$.

*Proof (of correctness).* The algorithm computes $\gcd(a, b)$ correctly, because this is true for Algorithm 1.25. Furthermore, it is easy to check that throughout the algorithm, we always have $r_i = s_i a + t_i b$. □

One can show that Algorithm 1.27 requires $\mathcal{O}(n \cdot m)$ bit-operations when applied to two integers of lengths $n$ and $m$, respectively.

As an application, the algorithm can be used to compute the inverse of $\bar{x} \in \mathbb{Z}/(m)$, if the representative $x \in \mathbb{Z}$ is coprime to $m$ (i.e. $\gcd(m, x) = 1$), because the algorithm then yields $s, t \in \mathbb{N}$, such that $1 = sx + tm$ and therefore $\overline{sx} = \overline{1} \in \mathbb{Z}/(m)$.

## 1.4 Primality Testing

We write $\mathbb{P} \subset \mathbb{N}$ for the set of prime numbers. Our goal is to determine whether a given $n \in \mathbb{N}$ is a prime number.

The most straightforward algorithm is to check whether $m \mid n$ for any $m \in \mathbb{N}$ with $2 \leq m^2 \leq n$. However, this requires $\mathcal{O}(\sqrt{n})$ divisions with remainder and because $n = \Theta(2^{l(n)})$, this is "almost exponential" in the length $l(n)$.

As a refinement, one can only check for the $m \in \mathbb{P}$ with $m^2 \leq n$. Since there are approximately $\frac{\sqrt{n}}{\log_2(\sqrt{n})}$ prime numbers that are smaller than $\sqrt{n}$, this does not improve the complexity by much.

We recall some results on arithmetic in $\mathbb{Z}/(m)$:

- For $p \in \mathbb{P}$, $\mathbb{F}_p = \mathbb{Z}/(p)$ is a field and its unit group $\mathbb{F}_p^\times$ is cyclic; i.e. there exists $u \in \mathbb{F}_p^\times$, such that $\mathbb{F}_p^\times = \langle u \rangle = \{u^i : i \in \mathbb{Z}\}$, $\operatorname{ord}(u) = p - 1$.
- By *Fermat's little theorem*, for any finite group $G$ and $u \in G$, we have $u^{|G|} = 1$. In particular, this means that $z^{p-1} \equiv 1 \mod p$ for $z \in \mathbb{Z}$ and $p \in \mathbb{P}$ with $p \nmid z$.
- Any $m \in \mathbb{N}_{>1}$ can be discomposed into its prime factorization: $m = \prod_{i=1}^r p_i^{e_i}$ with the $p_i$ pairwise distinct prime numbers and exponents $e_i \in \mathbb{N}_{>0}$. Then the *Chinese remainder theorem* states that
$$\mathbb{Z}/(m) \to \mathbb{Z}/(p_1^{e_1}) \oplus \cdots \oplus \mathbb{Z}/(p_r^{e_r}),\ x \mapsto (x \mod p_1^{e_1}, \ldots, x \mod p_r^{e_r})$$
is a ring homomorphism and thus $(\mathbb{Z}/(m))^\times \cong (\mathbb{Z}/(p_1^{e_1}))^\times \times \cdots \times (\mathbb{Z}/(p_r^{e_r}))^\times$ as groups.

The following theorem describes the unit group $(\mathbb{Z}/(p^e))^\times$.

**Theorem 1.28.** Let $p \in \mathbb{P} \setminus \{2\}$ and $e \in \mathbb{N}_{>0}$. Then $(\mathbb{Z}/(p^e))^\times$ is cyclic of order $(p-1)p^{e-1}$; i.e.
$$(\mathbb{Z}/(p^e))^\times \cong \mathbb{Z}/\big((p-1)p^{e-1}\big).$$

*Proof.* There are $p^e$ numbers between 1 and $p^e$ and $p^{e-1}$ of those are divisible by $p$, so $\operatorname{ord}\big((\mathbb{Z}/(p^e))^\times\big) = p^e - p^{e-1}$. We already know that the statement is true for $e = 1$, so let $e > 1$.
Since $\mathbb{F}_p^\times$ is cyclic, there exists $z \in \mathbb{Z}$, such that $z^i \equiv 1 \mod p$ for $i \in \mathbb{Z}$ if and only if $(p-1) \mid i$. Consider the element $a := z^{p^{e-1}} + p^e\mathbb{Z} \in (\mathbb{Z}/(p^e))^\times$.
*Claim*: $\operatorname{ord}(a) = p - 1$.
Using Fermat's little theorem, we first observe

$$a^{p-1} = z^{(p-1)p^{e-1}} + p^e\mathbb{Z} = 1 \in (\mathbb{Z}/(p^e))^\times,$$

so $\operatorname{ord}(a) \mid p - 1$. Now let $i \in \mathbb{N}_{>0}$ with $a^i = 1$ be given. Then we see that

$$z^{ip^{e-1}} \equiv 1 \mod p^e \implies z^{ip^{e-1}} \equiv 1 \mod p \implies (p-1) \mid i \cdot p^{e-1} \implies (p-1) \mid i,$$

implying $\operatorname{ord}(a) = p - 1$.
*Claim*: The element $b := 1 + p + p^e\mathbb{Z} \in (\mathbb{Z}/(e^p))^\times$ satisfies $\operatorname{ord}(b) = p^{e-1}$.
To prove this, we will first show that $(1+p)^{p^{k-1}} \equiv 1 + p^k \mod p^{k+1}$ by induction on $k$:

$k = 1$: For $k = 1$, we get the equality $1 + p = 1 + p$.

$k \to k+1$: By the inductive hypothesis, there exists $x \in \mathbb{Z}$, such that $(1+p)^{p^{k-1}} = 1 + p^k + x \cdot p^{k+1}$, so we calculate

$$\begin{aligned}
(1+p)^{p^k} &= \big(1 + p^k + x \cdot p^{k+1}\big)^p \\
&= \sum_{i=0}^{p} \binom{p}{i} \big(1+p^k\big)^{p-i} \big(xp^{k+1}\big)^i \\
&= \big(1+p^k\big)^p + \sum_{i=1}^{p} \binom{p}{i} \big(1+p^k\big)^{p-i} \big(xp^{k+1}\big)^i \\
&\equiv \big(1+p^k\big)^p \mod p^{k+2} \\
&= \sum_{i=0}^{p} \binom{p}{i} p^{ik} \equiv 1 + p^{k+1} \mod p^{k+2}.
\end{aligned}$$

Plugging in $k = e$ and $k = e - 1$ yields

$$\begin{aligned}
(1+p)^{p^{e-1}} &\equiv 1 + p^e \mod p^{e+1} \equiv 1 \mod p^e &&\implies \operatorname{ord}(b) \mid p^{e-1} \\
(1+p)^{p^{e-2}} &\equiv 1 + p^{e-1} \mod p^e \not\equiv 1 \mod p^e &&\implies \operatorname{ord}(b) \nmid p^{e-2},
\end{aligned}$$

implying $\operatorname{ord}(b) = p^{e-1}$.
*Claim*: We have $\operatorname{ord}(a \cdot b) = (p-1)p^{e-1}$; i.e. $a \cdot b$ generates $(\mathbb{Z}/(e^p))^\times$.
By Fermat's little theorem, we have $(a \cdot b)^{p^{e-1}(p-1)} = 1$, so $\operatorname{ord}(a \cdot b) \mid p^{e-1}(p-1)$. On the other hand, for $i \in \mathbb{N}_{>0}$ with $(ab)^i = 1$, we observe

$$1 = (ab)^{i(p-1)} = a^{i(p-1)} b^{i(p-1)} = 1 \cdot b^{i(p-1)}.$$

Because $\operatorname{ord}(b) = p^{e-1}$, this means that $p^{e-1} \mid i(p-1)$, so $p^{e-1} \mid i$. We conclude

$$1 = (ab)^{p^{e-1}i} = a^{p^{e-1}i} \implies (p-1) \mid p^{e-1}i \implies (p-1) \mid i \implies (p-1)p^{e-1} \mid i. \qquad \square$$

One can show that for $e \geq 3$, it holds $(\mathbb{Z}/(2^e))^\times \cong \mathbb{Z}/(2) \times \mathbb{Z}/(2^{e-2})$.

### 1.4.1 The Fermat Test

Lec 7
2021-11-09

By Fermat's little theorem, any prime number $n \in \mathbb{P}$ satisfies $a^{n-1} \equiv 1 \mod n$ for all $a \in \{1, \dots, n-1\}$. In fact, the other direction is also correct; any $n \in \mathbb{N}_{>1}$ with this property is a prime number, because if $a \in \{1, \dots, n-1\}$ with $a \mid n$ satisfies $a^{n-1} \equiv 1 \mod n$, then $a \mid 1$ and thus $a = 1$.
By checking the above condition for all numbers $a \in \{1, \dots, n-1\}$, we can thus determine whether a given number is prime or not. However, checking all such $a$ is very inefficient. Instead, we draw random numbers $a \in \{1, \dots, n-1\}$ and check if the condition is true for each of those. If one of the numbers fails the test, we know that $n$ is not prime; if all pass, we can at least suspect that $n$ is prime and by testing more and more numbers, we can be more confident in the obtained result. This procedure gives rise to the following randomized algorithm (*Monte Carlo Algorithm*).

**Algorithm 1.29 (Fermat Test).**

```
Input: $n \in \mathbb{N}_{>1}$ odd
Output: "$n \notin \mathbb{P}$" or "probably $n \in \mathbb{P}$"
    (1) Choose $a \in \{1, \dots, n-1\}$ at random.
    (2) Compute $b := a^{n-1} \mod n$.
    (3) Return $\begin{cases} \text{“probably } n \in \mathbb{P}\text{”} & b \equiv 1 \mod n \\ \text{“} n \notin \mathbb{P}\text{”} & \text{otherwise} \end{cases}$
```

Of course, when implementing such an algorithm on a computer, one would exclude 1 from the range of random numbers. To obtain a good complexity for Algorithm 1.29, we need to be able to quickly compute powers of a given element.

**Algorithm 1.30 (Fast exponential).**
Let $M$ be a monoid.

```
Input: $a \in M$, $e = \sum_{i=0}^{n-1} e_i 2^i \in \mathbb{N}$ with $e_i \in \{0, 1\}$
Output: $y = a^e \in M$
    (1) Initiate $y := 1 \in M$, $b := a$.
    (2) For $i = 0, \dots, n-1$:
        (3) If $e_i = 1$: $y := y \cdot b$
        (4) $b := b^2$.
```

**Example 1.31.** We have $a^{10} = a^{8+2} = \left((a^2)^2\right)^2 \cdot a^2$ for any element $a$ in a monoid.

Algorithm 1.30 requires $\mathcal{O}(n)$ operations in $M$. With the naive multiplication algorithm, we need $\mathcal{O}(l(n)^2)$ bit-operations for multiplying two numbers in $M = \mathbb{Z}/(n)$. Therefore, Algorithm 1.29 requires $\mathcal{O}(l(n)^3)$ bit-operations.

Algorithm 1.29 will never produce false negatives but there can be false positives and in fact quite a lot of those, as the following example shows.

**Example 1.32.** (a) Let $n := 561 = 3 \cdot 11 \cdot 17$ and $a \in \{1, \dots, n-1\}$ coprime to $n$ (i.e. $\gcd(a,n) = 1$). The number $n$ is chosen in such a way that $p-1$ divides $n-1$ for $p$ a prime factor of $n$, so by Fermat's little theorem, we have

$$\begin{aligned} a^{n-1} &= (a^2)^{280} \equiv 1 \mod 3 \\ a^{n-1} &= (a^{10})^{56} \equiv 1 \mod 11 \\ a^{n-1} &= (a^{16})^{35} \equiv 1 \mod 17, \end{aligned}$$

implying $a^{n-1} \equiv 1 \mod n$. Since this holds true for every coprime number $a \in \{1, \dots, n-1\}$, there are at least $(1 - \frac{1}{3} - \frac{1}{11} - \frac{1}{17}) \cdot 560 > 289$ - so more than half - false positives (the precise number of coprimes is 320).

(b) The number $2207 \cdot 6617 \cdot 15443$ turns out to have false positives in 99.9% of the cases.

The previous example shows that in bad cases, the Fermat test will probably return a wrong result. To capture which numbers constitute the bad cases, we introduce some terminology.

**Definition 1.33.** Let $n \in \mathbb{N}_{>1}$ odd and $a \in \{1, \dots, n-1\}$.

(a) $n$ is called a **pseudo prime to base** $a$ if $a^{n-1} \equiv 1 \mod n$.

(b) Otherwise, $a$ is called a **(Fermat) witness of compositeness** of $n$.

(c) If $n$ is composite and $a^{n-1} \equiv 1 \mod n$ for all coprime $a \in \{1, \dots, n-1\}$ then $n$ is called a **Carmichael number**.

By definition, a pseudo prime to base $a$ is just a natural number that passes the Fermat test w.r.t. $a$ and the problematic numbers from the previous example are the Carmichael numbers. One can show that there exist infinitely many Carmichael numbers.
Note that because a number $a \in \{1, \dots, n-1\}$ that is not coprime to $n$ is not invertible in $\mathbb{Z}/(n)$, it is always a witness of compositeness of $n$.

**Proposition 1.34.**

(a) Let $n \in \mathbb{N}$ be an odd composite number that is not a Carmichael number. Then more than $\frac{n-1}{2}$ numbers from $\{1, \dots, n-1\}$ are witnesses of compositeness of $n$.

(b) Every Carmichael number $n$ is square-free (i.e. $p^2 \nmid n$ for any $p \in \mathbb{P}$).

*Proof.* (a) Consider the group homomorphism

$$\Psi \colon G := (\mathbb{Z}/(n))^\times \to (\mathbb{Z}/(n))^\times, \ a \mapsto a^{n-1}$$

By hypothesis, the image $\mathrm{im}(\Psi)$ contains $\{1\}$ as a proper subset. From the homomorphism theorem it follows that

$$|\ker \Psi| = \frac{|G|}{|\mathrm{im}(\psi)|} \leq \frac{|G|}{2} < \frac{n-1}{2},$$

so at least $n - 1 - |\ker \Psi| > \frac{n-1}{2}$ numbers have witnesses.

(b) Suppose that $n$ is not square-free; i.e. $n = p^e \cdot r$ with $p \in \mathbb{P}$, $e \geq 2$, $r \in \mathbb{N}$, $p \nmid r$. By assumption, $p$ and $r$ are odd, so Theorem 1.28 yields the existence of $x \in \mathbb{Z}$ with $x^p \equiv 1 \mod p^e$ and $x \not\equiv 1 \mod p^e$. By the Chinese remainder theorem, there is $a \in \{1, \dots, n-1\}$, such that $a \equiv x \mod p^e$ and $a \equiv 1 \mod r$. Thus $a^p \equiv 1 \mod n$ and $a \not\equiv 1 \mod n$, so $a^n \equiv 1 \mod n$. We conclude $a^{n-1} \not\equiv 1 \mod n$, showing that $n$ is not Carmichael number. □

### 1.4.2 The Miller-Rabin Test

To solve the problem of Carmichael numbers, we introduce a refinement of Algorithm 1.29, which again is a Monte Carlo algorithm. The algorithm is based on the following proposition.

**Proposition 1.35.** Let $p \in \mathbb{P}$ be an odd prime and $a \in \{1, \dots, p-1\}$. Writing $p - 1 = m \cdot 2^k$ with $m \in \mathbb{N}$ odd, we have

$$a^m \equiv 1 \mod p \quad \text{or} \quad \exists\ i \in \{0, \dots, k-1\} : a^{m \cdot 2^i} \equiv -1 \mod p.$$

*Proof.* Assume $a^m \not\equiv 1 \mod p$ and let $i \in \mathbb{N}$ be maximal such that $a^{m \cdot 2^i} \not\equiv 1 \mod p$. By Fermat's little theorem, we have $i \in \{1, \dots, k-1\}$. This shows that $b \coloneqq a^{m \cdot 2^i} \in \mathbb{F}_p$ satisfies $b \neq 1$ and $b^2 = 1$, so $b$ is a zero of the polynomial $x^2 - 1 \in \mathbb{F}_p[x]$ and it follows $b = -1 \in \mathbb{F}_p$. □

**Algorithm 1.36** (**Miller-Rabin Test**).

```
Input: $n \in \mathbb{N}_{>1}$ odd
Output: "$n \notin \mathbb{P}$" or "probably $n \in \mathbb{P}$"
    (1) Write $n - 1 = m \cdot 2^k$ with $m \in \mathbb{N}$ odd.
    (2) Choose $a \in \{1, \dots, n-1\}$ at random.
    (3) Compute $b \coloneqq a^m \mod n$.
    (4) If $b \equiv 1 \mod n$ or $b \equiv -1 \mod n$: Return "probably $n \in \mathbb{P}$".
    (5) For $i = 1, \dots, k-1$:
        (6) $b \coloneqq b^2 \mod n$.
        (7) If $b \equiv -1 \mod n$: Return "probably $n \in \mathbb{P}$".
    (8) Return "$n \notin \mathbb{P}$".
```

**Definition 1.37.** Let $n \in \mathbb{N}_{>1}$ be an odd number and $a \in \{1, \dots, n-1\}$.

(a) $n$ is called a **strong pseudo prime to base** $a$ if Proposition 1.35 holds for $p = n$; i.e. if Algorithm 1.36 returns "probably $n \in \mathbb{P}$" when $a$ was the random number drawn.

(b) Otherwise, $a$ is called a **strong witness of compositeness of** $n$.

By definition, every strong pseudo prime is a pseudo prime and every witness is a strong witness. The hope that there are significantly less strong pseudo primes than pseudo primes is confirmed by the following example.

**Example 1.38.** Let $n \in \mathbb{N}_{>1}$ be an odd composite number.

(a) If $n < 2047$, then 2 is a strong witness for $n$.

(b) If $n < 1{,}373{,}653$, then 2 or 3 is a strong witness of compositeness of $n$.

Lec 8
2021-11-11

Because we need $\mathcal{O}(l(n))$ multiplications in $\mathbb{Z}/(n)$, Algorithm 1.36 requires $\mathcal{O}(l(n)^3)$ bit-operations in total, just like Algorithm 1.29. And just like that algorithm, it might produce false positives but never false negatives. The following theorem shows that there is no analogous concept to Carmichael numbers for the Miller-Rabin test, demonstrating that it constitutes a significant improvement over the Fermat test.

**Theorem 1.39.** If $n \in \mathbb{N}_{>1}$ is an odd composite number, then more than half of all numbers in $\{1, \ldots, n-1\}$ are strong witnesses.

*Proof.* If $n$ is not be Carmichael number, then Proposition 1.34 shows that more than half of those numbers are witnesses and thus also strong witnesses.
Therefore, we may assume that $n$ is a Carmichael number; that is, for all coprime $a \in \{1, \ldots, n-1\}$ we have $a^{2^k \cdot m} \equiv 1 \mod n$, where $n-1 = 2^k \cdot m$ with $m \in \mathbb{N}$ odd. From Proposition 1.34 it follows that $n$ is square-free, so it can be written as $n = p \cdot r$ with $p \in \mathbb{P}$, $p \nmid r$ and $r \geq 3$.
Define $j \in \mathbb{N}$ minimal such that $a^{2^j \cdot m} \equiv 1 \mod p$ for all $a \in \{1, \ldots, n-1\}$ that are coprime to $n$. Clearly, $j \leq k$ and we observe that $j = 0$ is impossible, as it would imply $1 \equiv (n-1)^m \equiv (-1)^m \mod p$, which is not true since $m$ is odd.
Therefore, we have $j \in \{1, \ldots, k\}$ and we consider the subgroup

$$H := \left\{ x \in (\mathbb{Z}/(n))^\times : x^{2^{j-1} \cdot m} = \pm 1 \right\} \subset (\mathbb{Z}/(n))^\times =: G.$$

*Claim 1:* Any $a \in \{1, \ldots, n-1\}$ coprime to $n$ with $\overline{a} \in G \setminus H$ is a strong witness.
Since $\overline{a} \notin H$, we have $a^{2^i \cdot m} \not\equiv \pm 1 \mod n$ for $0 \leq i < j$. If $j \leq i \leq k-1$, then $a^{2^i \cdot m} \equiv 1 \mod p$; that is, $p \mid a^{2^i m} - 1$, so $p \nmid a^{2^i m} + 1$ and we conclude $a^{2^i \cdot m} \not\equiv -1 \mod n$. This shows that $a$ is a strong witness.
*Claim 2:* $H$ is a proper subgroup of $G$.
By minimality of $j$, there exists $x \in \{1, \ldots, n-1\}$, such that $x$ is coprime to $n$ and $x^{2^{j-1} \cdot m} \not\equiv 1 \mod p$. By the Chinese remainder theorem, there is $a \in \{1, \ldots, n-1\}$ with $a \equiv x \mod p$ and $a \equiv 1 \mod r$. Therefore $a^{2^{j-1} \cdot m} \not\equiv 1 \mod p$ but $a^{2^{j-1} \cdot m} \equiv 1 \mod r$, so $\overline{a}^{2^{j-1} \cdot m} \not\equiv \pm 1$ and $\overline{a} \notin H$.
Because every number that is not coprime to $n$ is a strong witness, Claim 1 and Claim 2 imply that the number of strong witnesses in $\{1, \ldots, n-1\}$ is at least

$$n - 1 - |H| \geq n - 1 - \frac{|G|}{2} > n - 1 - \frac{n-1}{2} = \frac{n-1}{2}. \qquad \square$$

In fact, by refining the previous argument, one can show that more than $\frac{3}{4}$ of all numbers are strong witnesses.
The theorem means that by iterating Algorithm 1.36, the probability of a false positive decreases exponentially (assuming the random number generation is independent).
It is also interesting to mention that one can define a deterministic variant of Algorithm 1.36 with running-time $\mathcal{O}(l(n)^5)$ if the generalized Riemann Hypothesis is true.

### 1.4.3 The AKS-test

We now study a deterministic primality test algorithm with polynomial running-time. It was proposed in 2004 by Agrawal, Kayal, Saxena in “*PRIMES is in P*”.

For a commutative ring $R$ whose characterstic $p := \operatorname{char}(R) \in \mathbb{P}$ is a prime number, the ring homomorphism

$$R \to R,\ a \mapsto a^p$$

is called *Frobenius homomorphism.*
That this is indeed additive follows from the calculation

$$(a+b)^p = \sum_{i=0}^{p} \binom{p}{i} a^i b^{p-i} = a^p + b^p \quad \forall\ a, b \in R.$$

**Proposition 1.40.** For any polynomial $f \in \mathbb{Z}[x]/(p)$, it holds $f^p = f(x^p)$.
In particular, for $p \in \mathbb{P}$, $a \in \mathbb{Z}$ and $r \in \mathbb{N}_{>0}$, we have

$$(x+a)^p = x^p + a \in \mathbb{Z}[x]/(p)$$

and

$$(x+a)^p = x^p + a \in \mathbb{Z}[x]/(x^r - 1, p). \tag{$*$}$$

*Proof.* Write $f = \sum_{i=0}^{n} a_i x^i \in \mathbb{F}_p[x] \cong \mathbb{Z}[x]/(p)$. Because the Frobenius homomorphism is additive, we have $(\sum_{i=0}^{n} a_i x^i)^p = \sum_{i=0}^{n} a_i^p x^{ip} \in \mathbb{Z}[x]/(p)$. Fermat's little theorem implies $a^p = a^{p-1} \cdot a = a \in \mathbb{F}_p$ for any $a \in \mathbb{F}_p$, and applying the injective ring homomorphism $\mathbb{F}_p \hookrightarrow \mathbb{F}_p[x] \cong \mathbb{Z}[x]/(p)$ to this equation yields $a^p = a \in \mathbb{Z}[x]/(p)$. This implies the first claim. The other equations follow by considering the polynomial $f = x + a$ and by using the homomorphism theorem

$$(\mathbb{Z}[x]/(p))/((x^r - 1)/(p)) \cong \mathbb{Z}[x]/(x^r - 1, p). \qquad \square$$

The main motivation for considering the second equation instead of the first one is that it makes computations more feasible. Indeed, to check if $n \in \mathbb{N}$ satisfies equation ($*$), we need $\mathcal{O}(l(n))$ multiplications in $\mathbb{Z}[x]/(x^r - 1, n)$ by Algorithm 1.30. Elements in this ring are represented by polynomials of degree smaller than $r$ and with coefficients in $\mathbb{N}_{<n}$. Therefore, multiplying two polynomials requires $\mathcal{O}(r^2)$ multiplications and additions in $\mathbb{Z}/(n)$, amounting to $\mathcal{O}(r^2 l(n))$ multiplications. If we use the naive multiplication algorithm (Algorithm 1.6), this requires $\mathcal{O}(r^2 l(n)^3)$ bit-operations.
After applying these operations, one has to reduce the resulting polynomial modulo $x^r - 1$. Denote the coefficients of that polynomial by $a_i$. Because $x^{r+k} \equiv x^k \mod (x^r - 1)$ for all $k \in \mathbb{N}$, one needs to add the "higher part" $a_{r+k}$ to the "lower part" $a_k$ for each $k \in \{0, \dots, r-1\}$, which requires $r$ operations in $\mathbb{Z}/(n)$, so $\mathcal{O}(rl(n)^2)$ bit-operations.
We conclude that the total number of bit-operations required is $\mathcal{O}(r^2 l(n)^3)$. Lec 9 2021-11-16

**Algorithm 1.41 (Test for perfect power).**

Input: $n \in \mathbb{N}_{>1}$.

Output: $m, e \in \mathbb{N}_{>1}$ with $n = m^e$ or "$n$ is not a perfect power".

```
(1) For $e = 2, \dots, \lfloor \log_2(n) \rfloor$:
    (2) Set $m_1 := 2$, $m_2 := n$.
    (3) While $m_1 \le m_2$:
        (4) Set $m := \lfloor \frac{m_1 + m_2}{2} \rfloor$.
        (5) If $m^e = n$: Return $m, e$.
        (6) If $m^e > n$: Set $m_2 := m - 1$.
            Else: Set $m_1 := m + 1$.
    (7) Return "$n$ is not a perfect power".
```

*Proof (of correctness).* The largest $e \in \mathbb{N}$ for which there can exist $m \in \mathbb{N}_{>1}$ with $n = m^e$ is $\lfloor \log_2(n) \rfloor$. Therefore, it is sufficient to search for $m = \sqrt[e]{n}$ in the interval $[2, n]$. By continuously bisecting this interval (i.e. performing a binary search for $\sqrt[e]{n}$), we can find $\sqrt[e]{n}$ or conclude that it is not a whole number. □

There is no point in using Algorithm 1.30 in this algorithm, because we need to compute all the powers $m^e$ throughout the algorithm anyway. Instead, we use the naive method $m^e = m \cdot m^{e-1}$ to obtain $m^e$. By stopping the calculation as soon as $m^{e'} > n$ for some $e' \leq e$, we always multiply numbers of length at most $l(n)$ and since we need at most $\log_2(l(n))$ multiplications, computing $m^e$ requires $\mathcal{O}(\log_2(l(n)) \cdot l(n)^2)$ bit-operations. Because both the *while* and *for* loop are run $\log_2(n) = \mathcal{O}(l(n))$ times, this amounts to $\mathcal{O}(l(n)^4 \cdot \log_2(l(n)))$ bit-operations for the whole algorithm.

Using this algorithm as a subroutine, we may now state the AKS-test and prove that it always gives the correct result.

**Algorithm 1.42 (AKS primality test).**

```
 Input: $n \in \mathbb{N}_{>1}$ of length $l := \lfloor \log_2(n) \rfloor + 1$.
Output: "$n \in \mathbb{P}$" or "$n \notin \mathbb{P}$".
    (1) Use Algorithm 1.41 to determine whether $n$ is a perfect power, and if so, return
        "$n \notin \mathbb{P}$".
    (2) Find $r \in \mathbb{N}_{>1}$ minimal, such that $r \mid n$ or $n^i \not\equiv 1 \mod r$ for all $i \in \{1, \ldots, l^2\}$.
    (3) If $r = n$: Return "$n \in \mathbb{P}$".
    (4) If $r \mid n$: Return "$n \notin \mathbb{P}$".
    (5) For $a = 1, 2, \ldots, \lfloor \sqrt{r} l \rfloor$:
        If $(x + a)^n \not\equiv x^n + a \mod (n, x^r - 1)$: Return "$n \notin \mathbb{P}$".
    (6) Return "$n \in \mathbb{P}$".
```

The proof that the algorithm always gives the correct result is rather involved. It requires the following two propositions.

**Proposition 1.43.** For $r \in \mathbb{N}_{>1}$ and $p \in \mathbb{P}$, let

$$I(r, p) := \{(m, f) \in \mathbb{N} \times \mathbb{F}_p[x] : f(x)^m \equiv f(x^m) \mod (x^r - 1)\} \subset \mathbb{N} \times \mathbb{F}_p[x].$$

Then the following statements hold true:

(a) If $(m, f), (m', f) \in I(r, p)$, then $(m \cdot m', f) \in I(r, p)$.

(b) If $(m, f), (m, g) \in I(r, p)$, then $(m, f \cdot g) \in I(r, p)$.

(c) If $(p \cdot m, f) \in I(r,p)$ and $p \nmid r$, then $(m,f) \in I(r,p)$.

*Proof.* (a) We have

$$f(x)^{mm'} = (f(x)^m)^{m'} \equiv f(x^m)^{m'} \mod (x^r - 1),$$

and by substituting $x^m$ for $x$ in the equation $f(x)^{m'} \equiv f(x^{m'}) \mod (x^r-1)$, it follows

$$f(x^m)^{m'} \equiv f\left(x^{mm'}\right) \mod (x^{rm}-1).$$

Since $(x^r)^m - 1 = \left(\sum_{i=0}^{m-1} x^{ri}\right) \cdot (x^r - 1)$ and thus $(x^r-1) \mid (x^{mr}-1)$, we conclude $f(x^m)^{m'} \equiv f\left(x^{mm'}\right) \mod (x^r-1)$.

(b) This follows from the calculation

$$(f \cdot g)(x)^m = f(x)^m g(x)^m \equiv f(x^m)g(x^m) = (f \cdot g)(x^m) \mod (x^r-1).$$

(c) By assumption and Proposition 1.40, we have

$$(f(x)^m)^p = f(x)^{mp} \equiv f(x^{mp}) = f(x^m)^p \mod (x^r-1),$$

so $x^r - 1 \mid (f(x)^m)^p - f(x^m)^p$ and as the Frobenius homomorphism is a ring homomorphism, this implies $x^r - 1 \mid (f(x)^m - f(x^m))^p$. Since $p \nmid r$, $x^r - 1$ is square-free and it follows that $(x^r-1) \mid f(x)^m - f(x^m)$; that is, $f(x)^m \equiv f(x^m) \mod (x^r-1)$. □

**Proposition 1.44.** For all $n \in \mathbb{N}_{>1}$, we have $\binom{2n+1}{n} > 2^{n+1}$.

*Proof.* We prove the assertion by induction. For $n=2$, we calculate $\binom{5}{2} = 10 > 8 = 2^3$ and the inductive step is

$$\begin{aligned}\binom{2n+3}{n+1} &= \binom{2n+2}{n+1} + \binom{2n+2}{n} \\ &= \binom{2n+1}{n+1} + \binom{2n+1}{n} + \binom{2n+1}{n} + \binom{2n+1}{n-1} \\ &> 2 \cdot \binom{2n+1}{n} > 2^{n+2}.\end{aligned}$$

□

**Proposition 1.45.** Let $n \in \mathbb{N}_{>1}$ be a natural number of length $l := \lfloor \log_2(n) \rfloor + 1$, which is not a perfect power (i.e. for all $m, e \in \mathbb{N}_{>1}$, we have $n \neq m^e$). If $r \in \mathbb{N}_{>1}$ is chosen minimal such that $n^i \not\equiv 1 \mod r$ for all $i \in \{1, \dots, l^2\}$ and if $s \nmid n$ for all $s \in \{2, \dots, r\}$ and for all $a \in \{1, \dots, \lfloor \sqrt{r} l \rfloor\}$, we have $(x+a)^n \equiv x^n + a \mod (n, x^r - 1)$, then $n$ is a prime number.

*Proof.* First note that if all prime divisors $p$ of $n$ satisfied $p \equiv 1 \mod r$, then we would have $n \equiv 1 \mod r$, contradicting the assumption. Therefore, there exists $p \in \mathbb{P}$ with $p \mid n$ and $p \not\equiv 1 \mod r$. Because all prime divisors of $n$ are larger than $r$, it follows $p > r$, $\gcd(n,r) = 1$ and $\gcd(p,r) = 1$.
Let $n = m \cdot p$ for some $m \in \mathbb{N}$ and observe that $\gcd(m,r) = 1$. Consider

$$I := I(r,p) = \{(m,f) \in \mathbb{N} \times \mathbb{F}_p[x] : f(x)^m \equiv f(x^m) \mod (x^r-1)\} \subset \mathbb{N} \times \mathbb{F}_p[x].$$

from Proposition 1.43 and

$$G \coloneqq \langle \overline{n}, \overline{p} \rangle = \langle \overline{m}, \overline{p} \rangle \subset (\mathbb{Z}/(r))^\times.$$

Notice that

$$g \coloneqq |G| \quad \text{satisfies} \quad l^2 < g < r. \tag{1}$$

Moreover, with $k \coloneqq \lfloor \sqrt{r} l \rfloor \in \mathbb{N}$, we have

$$(x + \overline{a})^n = x^n + \overline{a} \in \mathbb{Z}[x]/(n, x^r - 1) \cong (\mathbb{Z}[x]/(n))/(x^r - 1)$$

for all $a \in \{1, \ldots, k\}$, so applying the ring homomorphism $\mathbb{Z}[x]/(n) \twoheadrightarrow \mathbb{Z}[x]/(p) \cong \mathbb{F}_p[x]$ yields $(x + \overline{a})^n = x^n + \overline{a} \in \mathbb{F}_p[x]/(x^r - 1)$; i.e. $(n, x + \overline{a}) \in I$.

Proposition 1.43 shows that $(m, x + \overline{a}) \in I$ for all $a \in \{0, \ldots, k\}$ (the case $a = 0$ is clear) and since $(p, x + \overline{a}) \in I$ by Proposition 1.40, we conclude that $(p^s m^t, x + \overline{a}) \in I$ for arbitrary $a \in \{0, \ldots, k\}$ and $s, t \in \mathbb{N}$ (the case $s = t = 0$ is clear).
For $e = (e_0, \ldots, e_k) \in \mathbb{N}^{k+1}$, consider

$$f_e \coloneqq \prod_{a=0}^{k} (x + \overline{a})^{e_a} \in \mathbb{F}_p[x]$$

and notice that $(p^s m^t, f_e) \in I$ by Proposition 1.43; that is,

$$f_e(x)^{p^s m^t} = f_e(x^{p^s m^t}) \in \mathbb{F}_p[x]/(x^r - 1).$$

There is a finite field $\mathbb{F}_q \supseteq \mathbb{F}_p$ which contains a primitive $r$-th root of unity $\zeta \in \mathbb{F}_q$ and as $r \nmid p - 1$, it follows that $\zeta \notin \mathbb{F}_p$. Since the ring homomophism Lec 10 2021-11-18

$$\mathbb{F}_p[x] \hookrightarrow \mathbb{F}_q[x] \xrightarrow{\mathrm{eval}_\zeta} \mathbb{F}_q$$

descends to $\mathbb{F}_p[x]/(x^r - 1)$, we conclude that

$$f_e(\zeta)^{p^s m^t} = f_e\Big(\zeta^{p^s m^t}\Big) \in \mathbb{F}_q \quad \forall\ s, t \in \mathbb{N}. \tag{2}$$

Now consider the subgroup

$$H \coloneqq \langle \zeta + \overline{a} \mid a \in \{0, \ldots, k\} \rangle = \left\{ f_e(\zeta) \mid e \in \mathbb{N}^{k+1} \right\} \subset \mathbb{F}_q{}^\times,$$

the set

$$T \coloneqq \left\{ (e_0, \ldots, e_k) \in \mathbb{N}^{k+1} : \sum_{a=0}^{k} e_a < g \right\}$$

and the map

$$\Phi \colon T \to H,\ e \mapsto f_e(\zeta).$$

We claim that $\Phi$ is injective.
For $e, \hat{e} \in T$ with $f_e(\zeta) = f_{\hat{e}}(\zeta)$, we have

$$f_e\Big(\zeta^{p^s m^t}\Big) \overset{(2)}{=} f_e(\zeta)^{p^s m^t} = f_{\hat{e}}(\zeta)^{p^s m^t} = f_{\hat{e}}\Big(\zeta^{p^s m^t}\Big) \quad \forall\ s, t \in \mathbb{N}$$

and because $G = \langle \overline{m}, \overline{p} \rangle$, the polynomial $f_e - f_{\hat{e}} \in \mathbb{F}_p[x]$ has $g$ distinct roots (because $\zeta$ is primitive). But $\deg(f_e - f_{\hat{e}}) < g$, so $f_e = f_{\hat{e}}$. Because

$$k = \lfloor \sqrt{r} l \rfloor \leq \sqrt{r} l \overset{(1)}{<} \sqrt{r}^2 = r < p,$$

the equivalence classes of two distinct elements of $\{0, \dots, k\}$ are different, and we conclude $e = \hat{e}$ as otherwise $f_e$ and $f_{\hat{e}}$ would have at least one root with different multiplicities. Now let

$$M \coloneqq \Big\{(z_0, \dots, z_k) \in \{1, \dots, g+k\}^{k+1} : z_i < z_{i+1} \ \forall\, i \in \{0, \dots, k-1\}\Big\}.$$

Since any $(z_0, \dots, z_k) \in M$ (with $z_{-1} \coloneqq 0$) satisfies

$$\sum_{i=0}^{k} (z_i - z_{i-1} - 1) = z_k - z_{-1} - (k+1) = z_k - (k+1) < g,$$

we obtain an injective map

$$\psi \colon M \to T,\ (z_0, \dots, z_k) \mapsto (z_0 - z_{-1} - 1, \dots, z_k - z_{k-1} - 1).$$

Because $\phi$ and $\psi$ are injective, it follows

$$|H| \geq |T| \geq |M| = \binom{g+k}{k+1} \overset{(1)}{\geq} \binom{\lfloor l\sqrt{g} \rfloor + 1 + k}{k+1} = \binom{\lfloor l\sqrt{g} \rfloor + (k+1)}{\lfloor l\sqrt{g} \rfloor} \overset{(1)}{\geq} \binom{2\lfloor l\sqrt{g} \rfloor + 1}{\lfloor l\sqrt{g} \rfloor},$$

so Proposition 1.44 implies

$$|H| > 2^{\lfloor l\sqrt{g} \rfloor + 1} \geq 2^{l\sqrt{g}} > 2^{\sqrt{g}}. \tag{3}$$

As the final step, we assume that $n \notin \mathbb{P}$; i.e. $m \neq 1$ and derive a contradiction. Since $n$ is not a perfect power, there exists a prime number $p' \in \mathbb{P}$ with $p' \neq p$ and $p' \mid m$, so the map

$$\chi \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N},\ (s, t) \mapsto p^s m^t$$

is injective. Therefore, the set

$$A \coloneqq \big\{p^s m^t : s, t \in \{0, \dots, \lfloor \sqrt{g} \rfloor\}\big\} \subset \mathbb{N}$$

satisfies $|A| = \big(\lfloor \sqrt{g} \rfloor + 1\big)^2 > g$. By definition of $G$, this means that there must exist $u, u' \in A$, $u \neq u'$ with $u \equiv u' \mod r$. Furthermore, for any $h \in H$, there exists $e \in \mathbb{N}^{k+1}$, such that $h = f_e(\zeta)$ and the calculation

$$h^u = f_e(\zeta)^u \overset{(2)}{=} f_e(\zeta^u) = f_e(\zeta^{u'}) \overset{(2)}{=} f_e(\zeta)^{u'} = h^{u'}$$

shows that all elements of $H$ are roots of the polynomial $x^u - x^{u'} \in \mathbb{F}_q[x]$. However, we observe that

$$\deg\Big(x^u - x^{u'}\Big) = \max(u, u') \leq \max(A) = (pm)^{\lfloor \sqrt{g} \rfloor} = n^{\lfloor \sqrt{g} \rfloor} \leq n^{\sqrt{g}},$$

contradicting (3) and thus concluding the proof. □

That Algorithm 1.42 is indeed correct is now essentially just a corollary of the previous proposition.

*Proof (of correctness of Algorithm 1.42).* It is clear that the algorithm always terminates and that it yields the correct result whenever step (6) is not reached (Proposition 1.40). If the algorithm terminates in step (6), then we are precisely in the situation of Proposition 1.45, so the algorithm also works in that case. □

We now examine the number of bit-operations required for Algorithm 1.42 and start with the following lemma, which also constitutes an interesting result from number theory in its own right. Its proof uses some elementary analysis.

**Lemma 1.46.** Let $n \in \mathbb{N}_{\geq 1}$. Then $\lambda(n) \coloneqq \operatorname{lcm}(1, 2, \ldots, n) \geq 2^{n-2}$.

*Proof.* Any polynomial $f \coloneqq \sum_{i=0}^{n} a_i x^i \in \mathbb{Z}[x]$ satisfies

$$\int_0^1 f(x)dx = \sum_{i=0}^{n} \frac{a_i}{i+1} = \frac{k}{\lambda(n+1)}$$

for some $k \in \mathbb{Z}$.
For $m \in \mathbb{N}$, consider the polynomial $f = x^m(1-x)^m \in \mathbb{Z}[x]$ and notice that for $0 < x < 1$, we have $0 < f(x) \leq \frac{1}{4^m}$ and thus $0 < \int_0^1 f(x)dx \leq \frac{1}{4^m}$. Therefore, we have

$$0 < \frac{k}{\lambda(2m+1)} \leq \frac{1}{4^m} \quad \text{and thus} \quad \lambda(2m+1) \geq k \cdot 4^m \geq 4^m,$$

so we conclude

$$\lambda(n) \geq \lambda\left(2 \cdot \left\lfloor \frac{n-1}{2} \right\rfloor + 1\right) \geq 4^{\lfloor \frac{n-1}{2} \rfloor} \geq 4^{\frac{n-2}{2}} = 2^{n-2}. \qquad \square$$

The next lemma gives as an upper bound on $r \in \mathbb{N}$ appearing in Algorithm 1.42.

**Lemma 1.47.** Let $n \in \mathbb{N}_{>1}$ be of length $l \coloneqq \lfloor \log_2(n) \rfloor + 1$ and $r \in \mathbb{N}_{>1}$ minimal, such that $n^i \not\equiv 1 \mod r$ for all $i \in \{1, \ldots, l^2\}$. Then $r \leq l^5$.

*Proof.* Aiming for contradiction, assume that $r > l^5$. Then for all $k \in \{2, \ldots, l^5\}$, there exists some $i \in \{1, \ldots, l^5\}$ with $n^i \equiv 1 \mod k$, so $k \mid \prod_{i=1}^{l^2}(n^i - 1)$. It follows $\lambda(l^5) \mid \prod_{i=1}^{l^2}(n^i - 1)$ and with Lemma 1.46 and the equation $2^l \geq n$, we see that

$$2^{l^5-2} \leq \lambda(l^5) \leq \prod_{i=1}^{l^2}(n^i - 1) < \prod_{i=1}^{l^2} n^i = n^{l^2(l^2+1)/2} \leq 2^{l^3(l^2+1)/2}.$$

Hence $l^5 - 2 < \frac{l^3(l^2+1)}{2}$, so $l^5 - l^3 < 4$, contradicting $l \geq 2$. $\square$

We can now show that the number of bit-operations required by Algorithm 1.42 is bounded from above by a polynomial in the length of the input.

**Theorem 1.48.** Algorithm 1.42 requires $\mathcal{O}(l^{16.5})$ bit-operations when applied to $n \in \mathbb{N}_{>1}$ with length $l \coloneqq \lfloor \log_2(n) \rfloor + 1$.

*Proof.* Applying Algorithm 1.41 in step (1) requires $\mathcal{O}(l^4 \cdot \log_2(l))$ bit-operations.
In step (2), we need to check for each $r > 1$ whether $r \mid n$ ($\mathcal{O}(l^2)$ bit-operations) and compute $n^i \mod r$ for each $i \in \{1, \ldots, l^2\}$ ($\mathcal{O}(l^2 \cdot \log_2(r)^2)$ bit-operations). By Lemma 1.47, we only need to check $r \leq l^5$ and this also implies $l(r) \in \mathcal{O}(\log_2(l))$, resulting in a total of $\mathcal{O}(l^5 \cdot (\mathcal{O}(l^2) + \mathcal{O}(l^2 \cdot \log_2(l)^2))) = \mathcal{O}(l^5 \cdot l^2 \cdot \log_2(l)^2)$ for step (2).
Finally, step (5) requires $\mathcal{O}(\sqrt{r} l r^2 l^3)$ bit-operations and by Lemma 1.47, this amounts to $\mathcal{O}(l^{16.5})$ bit-operations. $\square$

One can define variants of the AKS algorithm that have a running-time of $\mathcal{O}(l^6 \log_2(l)^m)$ for some (quite complicated) $m \in \mathbb{N}$.

Lec 11
2021-11-23

We finish this section by proving a version of the *prime number theorem*. To that end, we introduce the following notation.

**Definition 1.49.** For $r \in \mathbb{R}$, we denote the number of prime number less than or equal to $r$ as

$$\pi \colon \mathbb{R} \to \mathbb{N},\ r \mapsto |\{p \in \mathbb{P} : p \leq r\}|.$$

This is called the **prime-counting function**.

It is now rather straightforward to give an upper and a lower bound for this function.

**Theorem 1.50.** Let $n \in \mathbb{N}_{>1}$. Then

$$\frac{n-2}{\log_2(n)} \leq \pi(n) \leq 10\frac{n}{\log_2(n)}$$

and

$$\frac{1}{2}\frac{n}{\log_2(n)} \leq \pi(n) \leq 10\frac{n}{\log_2(n)}.$$

*Proof.* Using Lemma 1.46, we calculate

$$2^{n-2} \leq \lambda(n) = \prod_{\substack{p \in \mathbb{P} \\ p \leq n}} p^{\lfloor \log_p(n) \rfloor} \leq \prod_{\substack{p \in \mathbb{P} \\ p \leq n}} p^{\log_p(n)} = n^{\pi(n)} = 2^{\log_2(n)\pi(n)},$$

so $n-2 \leq \log_2(n) \cdot \pi(n)$.
To show that $\pi(n) \leq 10\frac{n}{\log_2(n)}$, we use induction on $n$. The cases $n < 9$ can be easily checked explicitly, so let $n \geq 9$ and write $m = \lceil \frac{n}{2} \rceil$. Then we have

$$\prod_{\substack{p \in \mathbb{P} \\ m<p\leq 2m}} p \geq \prod_{\substack{p \in \mathbb{P} \\ m<p\leq 2m}} m = m^{\pi(2m)-\pi(m)} = 2^{\log_2(m)(\pi(2m)-\pi(m))}$$

and

$$2^{2m} = (1+1)^{2m} = \sum_{i=0}^{2m} \binom{2m}{i} \geq \binom{2m}{m}.$$

Because any $p \in \mathbb{P}$ with $m < p \leq 2m$ divides $\binom{2m}{m}$, it follows

$$\binom{2m}{m} \geq \prod_{\substack{p \in \mathbb{P} \\ m<p\leq 2m}} p,$$

so combining the inequalities yields

$$\log_2(m)(\pi(2m) - \pi(m)) \leq 2m. \tag{$*$}$$

With $(*)$ and the inductive hypothesis, it follows

$$\pi(n) \leq \pi(2m) \overset{(*)}{\leq} \pi(m) + \frac{2m}{\log_2(m)} \leq \frac{12m}{\log_2(m)} \leq \frac{12m}{\log_2(n)-1} = \frac{12m}{\log_2(n)\Big(1 - \frac{1}{\log_2(n)}\Big)}$$

and because $n \geq 9$, we have $\log_2(n) > 3$, so this is upper bounded by

$$\frac{12m}{\log_2(n)\left(1 - \frac{1}{\log_2(n)}\right)} \leq \frac{12m}{\log_2(n)\left(1 - \frac{1}{3}\right)} = \frac{18m}{\log_2(n)} \leq \frac{9(n+1)}{\log_2(n)} \leq \frac{10n}{\log_2(n)}.$$

The final inequality $\frac{1}{2}\frac{n}{\log_2(n)} \leq \pi(n)$ follows by explicitly checking it for $n < 4$ and by using the already proven inequality for $n \geq 4$:

$$\pi(n) \geq \frac{n-2}{\log_2(n)} = \frac{2n-4}{2\log_2(n)} \geq \frac{n}{2\log_2(n)}. \qquad \square$$

The previous theorem constitutes a version of the well-known *prime number theorem*, which we now state without a proof.

**Theorem 1.51 (Prime Number Theorem).**
The prime-counting function $\pi\colon \mathbb{R} \to \mathbb{N}$ satisfies

$$\lim_{x\to\infty} \frac{\pi(x)}{\frac{x}{\ln(x)}} = 1.$$

The theorem means that $\pi(x)$ is approximately equal to $f(x) := \frac{x}{\ln(x)}$ for larger $x \in \mathbb{R}$. The derivitive

$$f'(x) = \frac{\ln(x) - 1}{\ln(x)^2} \approx \frac{1}{\ln(x)}$$

can then be used to further interpret this result: For $a \in \mathbb{R}$, call the next larger prime $p \in \mathbb{P}$. Since $f(a) + (x-a)\cdot f'(a)$ provides an approximation to $f$ for $x \in \mathbb{R}$ close to $a$, we have

$$1 = \pi(p) - \pi(a) \approx f(p) - f(a) \approx (p-a)\cdot f'(a) \approx \frac{p-a}{\ln(a)},$$

so the expected distance between two primes in the order of magnitude of $x$ is $\ln(x)$; that is, their distance rises linearly in the length of the numbers. In other words, the probability of a number "close to" $x$ being prime is roughly $\frac{1}{\ln(x)}$, showing that large numbers are less likely to be prime than smaller ones.

# 2 Cryptography

In this section, we study encryption and decryption of messages, which is not only interesting it its own right but also provides an application of our results about prime numbers from the previous section.

We consider the following situation: $A$ wants to send a message to $B$ such that $E$ cannot understand its meaning. Therefore, $A$ encrypts the message $x$ via an *encryption function* $\phi : x \mapsto x'$ (which is assumed to be a bijection) and sends the encrypted message $x'$ to $B$, so that $B$ can decrypt the message $x'$ with the *decryption function* $\phi^{-1} : x' \mapsto x$ to obtain the original message $x$. This way, $E$ cannot comprehend the actual transmitted information.

There are two main types of cryptography systems.
In **symmetric-key cryptography**, $A$ and $B$ share a secret key used for encryption and decryption. The *Advanced Encryption Standard* (*AES*) is a present-time example of symmetric-key cryptography, approved by the US government in 2001.
The main advantages of this approach are that it is quick and provides good security. However, in many cases it is hard to exchange the secret key in such a way that it cannot be read by other parties.
In contrast, in **public-key cryptography** the encryption function $\phi : x \mapsto x'$ is made public by $B$, but the decryption function $\phi^{-1} : x' \mapsto x$ is kept secret. Of course, this only makes sense if it is extremely difficult to compute the inverse $\phi^{-1}$ from $\phi$. One advantage of this approach is that it circumvents the need to secretly share any encryption keys.

We will only investigate public-key cryptography in this section, as it heavily relies on prime numbers and their properties. This type of cryptography has many applications in practice, including sending data, exchanging keys for symmetric-key cryptography and user authentification (usually in the internet). In the latter example, a user can authenticate themselves by sending a message $x$ together with $\phi^{-1}(x)$, so anyone can verify that $\phi\big(\phi(x)^{-1}\big) = x$ to confirm the identity of $B$.

## 2.1 RSA Encryption

Lec 12
2021-11-25

*Euler's totient function* (also called *Euler's phi function*) counts the number of coprime numbers that are less than or equal to a given number $n \in \mathbb{N}$:

$$\varphi\colon \mathbb{N} \to \mathbb{N},\ n \mapsto |\{a \in \mathbb{N} : 1 \le a \le n,\ \gcd(n,a) = 1\}|.$$

By definition, $\varphi(n)$ is the order of the group of units $(\mathbb{Z}/(n))^\times$ for $n \in \mathbb{N}_{>0}$.
By the Chinese remainder theorem, if $m$ and $n$ are coprime, then $\varphi(m \cdot n) = \varphi(m) \cdot \varphi(n)$.
It is also easy to see that $\varphi(p^k) = p^{k-1}(p-1)$ for any prime number $p \in \mathbb{P}$ and $k \in \mathbb{N}_{>0}$.

Euler's totient function is used in the *Rivest-Shamir-Adleman algorithm* (*RSA*), which is one of the most prominent algorithms for public-key cryptography.

**Algorithm 2.1 (RSA).**
Suppose that $A$ wants to send a secret message to $B$.

(1) $B$ chooses two distinct, large prime numbers $p$ and $q$ (usually with more than 100 decimal digits) and computes their product $n := p \cdot q$.

(2) $B$ chooses $e \in \mathbb{N}$ and computes $f \in \mathbb{N}$ such that $e \cdot f \equiv 1 \mod \varphi(n)$, where $\varphi(n) = (p-1)(q-1)$ denotes Euler's totient function.

(3) $B$ shares the *public key* $(n, e)$ and keeps the *private key* $f$ secret.

(4) $A$ represents the message as an element $x \in \mathbb{Z}/(n)$. Of course if $n$ is too small, $A$ can just split the information into multiple messages.

(5) $A$ computes $y := x^e$ and sends $y$ to $B$.

(6) $B$ computes $y^f$, which is the original message $x \in \mathbb{Z}/(n)$

*Proof (of correctness).* We have to check that $x = y^f$. By definition, there exists $a \in \mathbb{N}$, such that

$$e \cdot f = a \cdot (p-1)(q-1) + 1.$$

If $p \nmid x$, then we have $x^{ef-1} \equiv 1 \mod p$ by Fermat's little theorem, implying $x^{ef} \equiv x \mod p$ and we observe that this equation also holds if $p \mid x$. Since the same holds true for $q$, we conclude $x^{ef} \equiv x \mod n$ and $y^f = x^{ef} = x$. □

Note that the steps (1)-(3) are "initialization steps"; i.e. they only need to be performed a single time and don't need to be rerun if $A$ wants to send more messages to $B$. Since we will generally assume that our messages are given as elements of $\mathbb{Z}/(n)$, the only steps that needs to be performed for each sent message are the steps (5) and (6).

When investigating the complexity of this algorithm, we see that the initialization steps are the most expensive part of the algorithm.

(1) We find a prime number of length $l$ by a random search, where we skip any even number encountered. For each attempt, we perform up to $m$ Miller-Rabin tests (Algorithm 1.36) so that a number that passes all those tests is composite with a probability of less than $2^{-m}$ (Theorem 1.39). Since the Miller-Rabin test has complexity of $\mathcal{O}(l^3)$ and because by the prime number theorem, we need about $\mathcal{O}(l)$ attempts, it can be expected that this step requires a total of $\mathcal{O}(ml^4)$ bit-operations.

(2) Calculating $\varphi(n)$ requires $\mathcal{O}(l^2)$ bit-operations. We can then use the Extended Euclidean Algorithm (Algorithm 1.27) to compute $\gcd(e, \varphi(n))$. If the result is 1, then $f$ has been found: $\mathcal{O}(l^2)$.

(5) Using fast exponentiation (Algorithm 1.30), step (5) requires $\mathcal{O}(l(e)l^2) = \mathcal{O}(l^3)$ bit-operations.

(6) Similarly, step (6) needs $\mathcal{O}(l(f)l^2) = \mathcal{O}(l^3)$ bit-operations.

Note that in the unlikely case that we are unlucky and obtain a composite number in step (1), there will be unexpected errors in the encryption or decryption process, so this will almost certainly be detected throughout multiple applications of the algorithm.

Next we investigate the security of this algorithm. As mentioned, any algorithm implementing public-key cryptography has to ensure that it is computationally very difficult to obtain the private key from the public one. In the setting of this algorithm, any person that knows $p$ and $q$ can compute the private key $f$, so the security hinges on the

assumption that factorization is computationally extremly difficult.
Currently, this assumption seems to be true. However, there are some algorithms relying on the use of quantum computers that can compute factorizations in polynomial time, most notably *Shor's algorithm*.

While it is impossible to prove that Algorithm 2.1 is secure, we can at least prove that it is secure if and only if factorization is difficult (i.e. knowing $f$ is equivalent to knowing $p$ and $q$). Indeed, the following algorithm shows that one can compute $p$ and $q$ from $f$ in polynomial time. We note that for $m \coloneqq ef - 1$, we have $\varphi(n) \mid m$ and $m \leq n^2$ (because $e, f \leq \varphi(n) \leq n$) in the above algorithm.

**Algorithm 2.2** (**Compute a proper divisor of certain numbers**).

Input: $n \in \mathbb{N}_{>1}$ odd, composite and square-free; $m \in \mathbb{N}$ with $\varphi(n) \mid m$ and $m \leq n^2$.

Output: $d \in \mathbb{N}$ with $1 < d < n$ and $d \mid n$.

```
(1) Repeat:
    (2) Choose $a \in \{2, \dots, n-2\}$ randomly.
    (3) Set $k \coloneqq m$ and $d \coloneqq \gcd(a, n)$.
    (4) If $d \neq 1$: return $d$.
    (5) While $d \neq 1$ and $k \in \mathbb{N}$ even:
        (6) Compute $d \coloneqq \gcd\big(a^k - 1, n\big)$.
        (7) If $1 < d < n$: return $d$.
        (8) Set $k \coloneqq \frac{k}{2}$.
```

This algorithm is a *Las-Vegas algorithm*, meaning that if it terminates, then it yields the desired result. We will now prove this claim.

**Theorem 2.3.** If Algorithm 2.2 terminates, then it computes a factor of $n$. The expected number of iterations (of the the outer loop) required is less than or equal to 2.

*Proof.* Let $n = \prod_{i=1}^{r} p_i$ be a decomposition of $n$ into distinct primes $p_i$. Then $\varphi(n) = \prod_{i=1}^{r}(p_i - 1)$, so at the beginning of the algorithm, all $p_i - 1$ divide $k$. After some iterations of the algorithm, it will either terminate in step (4) or (7) or it will happen for the first time that $p_j - 1 \nmid k$ for some $j \in \{1, \dots, r\}$. In the first case, it is clear that a proper divisor of $n$ is returned, so we only need to consider the second case. We partition the index set $\{1, \dots, r\}$ into two subsets $\{1, \dots, r\} = I \coprod J$, where

$$I \coloneqq \{i \in \{1, \dots, r\} : p_i - 1 \mid k\}, \quad J \coloneqq \{i \in \{1, \dots, r\} : p_i - 1 \nmid k\}$$

and notice that $j \in J$. Let $a \in \mathbb{N}_{>0}$ be coprime to $n$. Then by Fermat's little theorem, we have $a^k \equiv 1 \mod p_i$ for all $i \in I$.
We now consider the case $i \in J$. Then $p_i - 1 \mid 2k$ and $p_i - 1 \nmid k$, so $k \equiv \frac{p_i - 1}{2} \mod p_i - 1$, implying $a^k \equiv \pm 1 \mod p_i$. Furthermore, with $k = l(p_i - 1) + r$ with $l \in \mathbb{Z}$ and $r \in \{1, \dots, p_i - 2\}$, we observe that the two polynomials $f = x^k - 1 \in \mathbb{F}_{p_i}[x]$ and $g = x^r - 1 \in \mathbb{F}_{p_i}[x]$ induce the same polynomial function $\mathbb{F}_{p_i} \to \mathbb{F}_{p_i}$ and therefore at least

one $a \in \mathbb{Z}/(p_i)^\times$ must satisfy $a^k = -1$, because otherwise $g$ would have $p_i - 1 > r = \deg(g)$ roots. In particular, there is some element $b \in \mathbb{Z}/(p_j)^\times$ with $b^k = -1$.
Consider the group homomorphism

$$\phi\colon \mathbb{Z}/(n)^\times \to \mathbb{Z}/(p_1)^\times \times \cdots \times \mathbb{Z}/(p_r)^\times,\ a \mapsto \bigl(a^k \mod p_1, \ldots, a^k \mod p_r\bigr),$$

which is just the composition of the homomorphism of the Chinese remainder theorem with the exponentiation map $(-)^k$:

$$\mathbb{Z}/(n)^\times \longrightarrow \mathbb{Z}/(p_1)^\times \times \cdots \times \mathbb{Z}/(p_r)^\times \xrightarrow{(-)^k} \mathbb{Z}/(p_1)^\times \times \cdots \times \mathbb{Z}/(p_r)^\times.$$

By the above, the image $\operatorname{im}(\phi)$ is contained in the subgroup $\{\pm 1\} \times \cdots \times \{\pm 1\}$, so by composing with the canonical projection $\{\pm 1\} \times \cdots \times \{\pm 1\} \twoheadrightarrow \{\pm 1\} \times \cdots \times \{\pm 1\}/\langle(-1, \ldots, -1)\rangle$, we obtain the group homomorphism

$$\tilde{\phi}\colon \mathbb{Z}/(n)^\times \to \{\pm 1\} \times \cdots \times \{\pm 1\}/\langle(-1, \ldots, -1)\rangle,\ a \mapsto \overline{(a^k \mod p_1, \ldots, a^k \mod p_r)}.$$

By definition, we have

$$\{c \in \mathbb{Z}/(n)^\times : \phi(c) \notin \{(1, \ldots, 1), (-1, \ldots, -1)\}\} = \Bigl\{c \in \mathbb{Z}/(n)^\times : c \notin \ker(\tilde{\phi})\Bigr\}$$

and this set is nonempty, because by the Chinese remainder theorem, there exists $c \in \mathbb{Z}/(n)^\times$, such that $c \equiv 1 \mod p_i$ for $i \neq j$ and $c \equiv b \mod p_j$.
Therefore, the homomorphism theorem implies that

$$2 \leq \Bigl|\operatorname{im}\Bigl(\tilde{\phi}\Bigr)\Bigr| = \frac{\bigl|\mathbb{Z}/(n)^\times\bigr|}{\Bigl|\ker(\tilde{\phi})\Bigr|},$$

so at least half the elements $a \in \mathbb{Z}/(n)^\times$ satisfy $\phi(a) \notin \{(1, \ldots, 1), (-1, \ldots, -1)\}$. In other words, for each such $a$, there exist $v, w \in \{1, \ldots, r\}$, such that $a^k \equiv 1 \mod p_v$ and $a^k \equiv -1 \mod p_w$, implying $p_v \mid a^k - 1$ and $p_w \nmid a^k - 1$. We conclude that the algorithm terminates for at least half the $a \in \{1, \ldots, n-1\}$ and because $k \in \mathbb{N}$ is even, it suffices to check $a \in \{2, \ldots, n-2\}$. Finally, the expected number of $a \in \{2, \ldots, n-2\}$ that need to be tested is thus upper bounded by

$$\sum_{i=1}^{\infty} i\left(\frac{1}{2}\right)^i = \sum_{i=1}^{\infty}\sum_{j=i}^{\infty}\left(\frac{1}{2}\right)^j = \sum_{i=1}^{\infty}\left(2 - \frac{\left(\frac{1}{2}\right)^i - 1}{\frac{1}{2} - 1}\right) = \sum_{i=1}^{\infty}\left(\frac{1}{2}\right)^{i-1} = 2.$$

□

One can show that the expected number of bit-operations of Algorithm 2.2 is in $\mathcal{O}(l(n)^4)$.

The *RSA problem* is the question whether it is possible to decrypt some (or even all) messages in the RSA algorithm without knowing the private key.

## 2.2 Diffie-Hellman Key Exchange

Lec 13
2021-12-02

The *Diffie-Hellman key exchange* is a simple algorithm that can be used to exchange keys over a channel that is visible to other parties.

**Algorithm 2.4 (Diffie-Hellman key exchange).**
Suppose that $A$ and $B$ want to exchange a key via an insecure channel.

(1) $A$ and $B$ agree on a large prime number $p \in \mathbb{P}$ and on some $g \in (\mathbb{Z}/(p))^\times$.

(2) $A$ chooses $a \in \mathbb{N}$ randomly and sends $u := g^a$ to $B$.

(3) $B$ chooses $b \in \mathbb{N}$ randomly and sends $v := g^b$ to $A$.

(4) $A$ computes $v^a = g^{ab}$ and $B$ computes $u^b = g^{ab}$. They share the secret key $g^{ab}$.

**Example 2.5.** We demonstrate the procedure with an example.

(a) Let $p = 17$ and $g = \overline{3} \in \mathbb{Z}/(17)^\times$.

(b) $A$ chooses $a = 7$, $\overline{3}^a = \overline{11}$.

(c) $B$ chooses $b = 4$, $\overline{3}^b = \overline{13}$.

(d) The key is $\overline{11}^4 = \overline{13}^7 = \overline{4}$.

For a given monoid $M$ and elements $m \in M$, $t \in \langle m \rangle$, the *discrete logarithm problem* $(DLP)$ is the task to compute some $a \in \mathbb{N}$ with $t = m^a$.
In the above algorithm, it is sufficient to compute $a$ or $b$ in order to determine the secret key $g^{ab}$, so if the discrete logarithm problem can be solved efficiently, then the algorithm offers very little security. Note that this is only a sufficient condition. Indeed, the *Diffie-Hellman problem* asks whether it is also necessary to solve DLP in order to obtain the secret key in the Diffie-Hellman key exchange.
It is possible to use *elliptic curve cryptography* in order to obtain a group that can be used for potentially difficult DLPs.

# 3 Factorization

In this section, we want to factorize a given natural number $n \in \mathbb{N}$. Of course, if we have an algorithm that determines a proper divisor $d \in \mathbb{N}$ (i.e. $d \mid n$ and $1 < d < n$) or tells us that the number is prime, then we can factorize any natural number by iterating the algorithm.

We start with a naive algorithm to determine all prime numbers that are less than or equal to a given threshold.

**Algorithm 3.1** (**Sieve of Eratosthenes**).

```
 Input: $n \in \mathbb{N}_{>0}$.
Output: A list $P$ of all prime number $p \leq n$, ordered by size.
   (1) Form a list $L$ of all natural numbers $m \in \mathbb{N}$ with $1 < m \leq n$ and let $P$ be an
       empty list.
   (2) While not all numbers in $L$ are marked:
       (3) Let $p$ be smallest unmarked number in $L$.
       (4) Add $p$ to the end of $P$.
       (5) Mark all numbers in $L$ which are divisible by $p$.
```

Given a composite number $n \in \mathbb{N}$, we can obtain a proper divisor of $n$ by running Algorithm 3.1 with input $\sqrt{n}$ and then performing trial divisions. This naive algorithm requires $\mathcal{O}(\sqrt{n} \cdot \log_2(n))$ bit-operations.

## 3.1 Pollard's rho Method

Let $M$ be a set and $f : M \to M$ a function. Then there is a left monoid action of the monoid $(\mathbb{N}, +)$ on $M$, where $n \in \mathbb{N}$ acts on $M$ by applying $f$ $n$ times; i.e. $n \cdot m = f^n(m)$. For $x \in M$, the orbit $\mathbb{N} \cdot x = \{n \cdot x : n \in \mathbb{N}\}$ is just a sequence $(x_n)_{n\in\mathbb{N}}$.
This observation is used in *Pollard's rho method*, which is another approach to factorizing numbers. The key idea is to consider a "chaotic" function $f \colon \mathbb{Z}/(n) \to \mathbb{Z}/(n)$, to choose a "starting value" $x_0 \in \mathbb{Z}/(n)$ and to consider the orbit $x_i \coloneqq f(x_{i-1}) = f^i(x_0)$.
Because $\mathbb{Z}/(n)$ is a finite set, there exist indices $i, j \in \mathbb{N}$ with $i < j$ and $x_i = x_j$. In particular, there are indices $i, j \in \mathbb{N}$ with $i < j$, such that $x_i \equiv x_j \mod p$ for some (unknown) prime divisor $p$ of $n$. From $i$ onwards, the sequence $(x_k \mod p)_{k\in\mathbb{N}}$ becomes periodic. Since the start of the sequence might be non-periodic, it can be visualized as the Greek letter $\rho$, hence the name.
The hope is now that $x_i \not\equiv x_j \mod n$. In that case, $d \coloneqq \gcd(x_i - x_j, n)$ satisfies $d < n$ and $p \mid d$, so it is a proper devisor of $n$.
In order to turn this idea into an algorithm, we need to determine $i, j \in \mathbb{N}$ with the above property. Checking all pairs $(i, j) \in \mathbb{N} \times \mathbb{N}$ would be very inefficient, but luckily the following proposition shows that it suffices to check pairs of the form $(i, 2i)$.

**Proposition 3.2.** Let $M$ be a set, $f \colon M \to M$ a function and $x_0 \in M$. Set $x_i \coloneqq f^i(x_0)$ and suppose that there exist $t, l \in \mathbb{N}$, $l > 0$ with $x_{t+l} = x_t$. Then there is $j \in \mathbb{N}$, such that $x_{2j} = x_j$ and $t \leq j < t + l$.

*Proof.* By assumption, we have $f^l(x_t) = x_t$, so any $a \in \mathbb{N}$ satisfies $f^{al}(x_t) = x_t$. Let $j := a \cdot l$ for some $a \in \mathbb{N}$. If $j \geq l$, then

$$x_{2j} = x_{al+j} = x_{(j-t)+t+al} = f^{j-t}\big(f^{al}(x_t)\big) = f^{j-t}(x_t) = x_j.$$

Therefore, if $t = 0$, then we may choose $j = l$ and otherwise we can use $j = t + (-t \mod l)$, where $-t \mod l$ is understood to be the unique representative of $-t$ in $\{0, \ldots, l-1\}$. □

Lec 14 2021-12-07

We can now state the algorithm that is based on the above observations with $f(x) := x^2 + 1$. It is a *Las-Vegas algorithm.*

**Algorithm 3.3 (Pollard's rho algorithm).**

Input: $n \in \mathbb{N}_{>1}$ composite.

Output: A proper divisor $d \in \mathbb{N}$ of $n$.

```
(1) Repeat:
    (2) Choose $x \in \{0, \ldots, n-1\}$ randomly.
    (3) Set $y := x$, $d := 1$.
    (4) While $d \neq n$:
        (5) Set $x := x^2 + 1 \mod n$ and $y := (y^2+1)^2 + 1 \mod n$.
        (6) Compute $d := \gcd(n, x - y)$.
        (7) If $1 < d < n$: return $d$.
```

It is clear that if the algorithm terminates, then it indeed returns a proper divisor of $n$. In order to estimate the expected running-time of the algorithm, we require the following lemma.

**Lemma 3.4.** When uniformly and independently drawing random numbers from $\{1, \ldots, n\}$, the expected number of draws until some number has appeared twice is less than $\sqrt{\frac{\pi n}{2}} + 2$.

*Proof.* Let $s \in \mathbb{N}$ be the number of draws until a match occurs. Note that

$$e^{-x} \geq 1 - x \quad \forall\ x \in \mathbb{R}_{\geq 0}, \tag{$*$}$$

because $f(x) := e^{-x} - (1-x)$ satisfies $f(0) = 0$ and $f'(x) \geq 0$ for all $x \in \mathbb{R}_{\geq 0}$. If $k \geq n+1$, then $\mathbb{P}(s > k) = 0$. Otherwise, we see that

$$\mathbb{P}(s > k) = \prod_{i=1}^{k}\left(1 - \frac{i-1}{n}\right) \overset{(*)}{\leq} \prod_{i=1}^{k} e^{-\frac{i-1}{n}} = e^{-\frac{(k-1)k}{2n}} \leq e^{-\frac{(k-1)^2}{2n}}.$$

Because $\mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0},\ x \mapsto e^{-\frac{(x-1)^2}{2n}}$ is monotonically decreasing and $\int_{-\infty}^{\infty} e^{-x^2} = \sqrt{\pi}$, the expected value of $s$ is

$$\sum_{k=0}^{\infty} \mathbb{P}(s > k) < 2 + \sum_{k=2}^{\infty} e^{-\frac{(k-1)^2}{2n}} \leq 2 + \int_1^{\infty} e^{-\frac{(x-1)^2}{2n}} dx = 2 + \int_0^{\infty} e^{-\frac{x^2}{2n}} dx$$
$$= 2 + \sqrt{2n} \int_0^{\infty} e^{-x^2} dx = 2 + \sqrt{2n} \cdot \frac{\sqrt{\pi}}{2} = 2 + \sqrt{\frac{n\pi}{2}},$$

where we first substituted $x - 1 \rightsquigarrow x$ and then $\frac{x}{\sqrt{2n}} \rightsquigarrow x$. □

This lemma has an interesting application in the *birthday problem*, which constitutes the following example.

**Example 3.5.** Suppose that people arrive at a party one after the other. When can we expect that there are two guests that share the same birthday (ignoring leap years and assuming that all dates are equally likely and independent from each other)? By Lemma 3.4, this is likely to be true after $\sqrt{\frac{365\pi}{2}} + 2 \approx 26$ people have arrived.

**Theorem 3.6.** Let $p \in \mathbb{P}$ be the smallest prime divisor of $n$. If the distribution of $f^i(x)$ with $f(x) := x^2 + 1 \mod p$ is (sufficiently) uniform and independent, then the expected number of bit-operations required for the inner part of Algorithm 3.3 (i.e. excluding the outer loop) is $\mathcal{O}(\sqrt[4]{n} \cdot \log_2(n)^2)$.

*Proof.* Denote by $x_i$ the value of $x$ in the $i$-th iteration and by $y_i$ the value of $y$ in the $i$-th iteration and note that $y_i = f^i(x_i) = f^{2i}(x_0) = x_{2i}$ for all $i \in \mathbb{N}$, because this is true in the beginning and

$$y_{i+1} = f^2(y_i) = f^2\big(f^i(x_i)\big) = f^{i+1}(x_{i+1}).$$

By minimaltiy, $p \leq \sqrt{n}$, so the expected number of iterations until $x_i = x_j \mod p$ is $\mathcal{O}(\sqrt[4]{n})$ by Lemma 3.4 and by Proposition 3.2 the same is true until we have $x_{2j} = x_j \mod p$. Because each iteration requires $\mathcal{O}(\log_2(n)^2)$ bit-operations, the assertion follows. □

## 3.2 Pollard's $p - 1$ Algorithm

Another approach to factoring numbers is based on the following idea. Let $p \mid n$ be a prime divisor. If $a \in \mathbb{Z}$ satisfies $p \nmid a$, then $a^{p-1} \equiv 1 \mod p$ by Fermat's little theorem. Therefore, if we can find $m \in \mathbb{N}$, such that $p - 1 \mid m$, then $a^m \equiv 1 \mod p$, implying $p \mid \gcd(n, a^m - 1)$.
Of course, since $p$ is unknown, there is no obvious way to obtain such a $m$, but one can try to multiply prime powers dividing $p - 1$.

**Definition 3.7.** Let $p \in \mathbb{P}$ and $B \in \mathbb{N}$. If all prime powers dividing $p - 1$ are less than or equal to $B$, then $p - 1$ is $B$-**powersmooth**.

If $p - 1$ is $B$-powersmooth, then

$$m := \prod_{\substack{q \in \mathbb{P} \\ q \leq B}} q^{\lfloor \log_q(B) \rfloor}$$

is a multiple of $p - 1$.
Therefore, we can choose some upper bound $B \in \mathbb{N}$, choose $m$ as above and hope that $p - 1$ is $B$-powersmooth. This gives rise to the following algorithm, which is another Las-Vegas algorithm.

**Algorithm 3.8** (**Pollard's $p - 1$ algorithm**).

Input: $n \in \mathbb{N}_{>1}$ composite.

Output: A proper divisor $d \in \mathbb{N}$ of $n$.

```
(1) Repeat:
    (2) Choose a smoothness bound $B \in \mathbb{N}$.
    (3) Using Algorithm 3.1, obtain a list $L$ of all primes $\leq B$.
    (4) Choose $a \in \{2, \dots, n-2\}$ randomly.
    (5) For $q \in L$:
        (6) Compute $k := q^{\lfloor \log_q(B) \rfloor}$
        (7) Set $a := a^k \mod n$ and $d := \gcd(n, a-1)$.
        (8) If $1 < d < n$: return $d$.
```

For RSA, this means that the prime numbers $p$ and $q$ used to generate the key should not be chosen such that $p-1$ or $q-1$ become $B$-powersmooth for a "moderate" $B$, because otherwise it may be easy to obtain the private key using Pollard's $p-1$ algorithm.

## 3.3 The Quadratic Sieve

Lec 15 2021-12-09

The **quadratic sieve** is from a mathematical standpoint the state of the art algorithm for factorization.

The main idea is to observe that writing a number $n \in \mathbb{N}$ as a product of two natural numbers is equivalent to writing it as a difference of two square numbers. Indeed, if $n = x^2 - y^2$ with $x, y \in \mathbb{N}$, then $n = (x+y)(x-y)$ is a factorization of $n$. Conversely, if $n = a \cdot b$ is odd with $a, b \in \mathbb{N}$, then we can write $n = \left(\frac{a+b}{2}\right)^2 - \left(\frac{a-b}{2}\right)^2$ with $\frac{a+b}{2}, \frac{a-b}{2} \in \mathbb{Z}$.

Our goal is thus to find $x, y \in \mathbb{Z}$, such that $n \mid x^2 - y^2$ but $n \nmid x+y$ and $n \nmid x-y$, because then $\gcd(n, x+y)$ and $\gcd(n, x-y)$ will be a proper divisors. In other words, we want to determine $x, y \in \mathbb{Z}$, such that $x^2 \equiv y^2 \mod n$ and $x \not\equiv \pm y \mod n$. Let $n = \prod_{j=1}^r p_j^{e_j}$ be a decomposition into prime factors and assume that $n$ odd. Furthermore, assume that $p_j \nmid x$ for all $j \in \{1, \dots, r\}$. By the Chinese remainder theorem, we have

$$x^2 \equiv y^2 \mod n \iff \left( \frac{y + p_j^{e_j}\mathbb{Z}}{x + p_j^{e_j}\mathbb{Z}} \right)^2 = 1 \in \left(\mathbb{Z}/\left(p_j^{e_j}\right)\right)^\times \quad \forall j \in \{1, \dots, r\}.$$

**Lemma 3.9.** Let $G = \langle t \rangle$ be a cyclic group of even order $n$. Then there are exactly two square roots of unity, namely 1 and $t^{\frac{n}{2}}$.

*Proof.* Let $x \in G$ with $x^2 = 1$ and $x = t^k$ for $k \in \{0, \dots, n-1\}$. It follows $t^{2k} = 1$, so $n \mid 2k$, which implies $\frac{n}{2} \mid k$. We conclude that $k = 0$ or $k = \frac{n}{2}$, which shows the assertion. □

Since $\left(\mathbb{Z}/\left(p_j^{e_j}\right)\right)^\times$ is cyclic of even order, Lemma 3.9 implies that there are exactly two square roots of unity in this group, so there exist exactly $2^r$ equivalence classes of $y$ in $(\mathbb{Z}/(n))^\times$ with $x^2 \equiv y^2 \mod n$. Because $x \equiv \pm y \mod n$ happens for exactly two equivalence classes, the likelihood of the "bad case" $x \equiv \pm y \mod n$ is $2^{1-r}$. We can explicitly check the case $r = 1$ with Algorithm 1.41.

**Example 3.10.** We demonstrate how one could try to determine $x, y, k \in \mathbb{Z}$, such that $x^2 = kn + y^2$ for given $n \in \mathbb{N}$.

(a) Let $n = 91$. We might try to choose $x$ slightly bigger than $\sqrt{kn}$, so that $x^2 - kn$ is small, so it is "more likely" to be a perfect square. Setting $k = 1$, we have $\sqrt{kn} \approx 9.5$, inspiring the choice $x = 10$ and because $x^2 \equiv 3^2 \mod n$, we let $y = 3$ and obtain $n = 10^2 - 3^2 = (10 + 3)(10 - 3) = 13 \cdot 7$.

(b) Let $n = 4633$ and choose $k = 3$. Because $\sqrt{3n} \approx 117.9$, we choose $x = 118$. Then $118 \equiv 5^2 \mod n$, so let $y = 5$. Because $\gcd(118 - 5, n) = 113$ and $\gcd(118 + 5, n) = 41$, we conclude $4633 = 41 \cdot 113$.

While the approach highlighted in the example works decently for "single usage", we require a more systematic search for $x$ and $y$ for practical application, which we investigate now.

We start by choosing a "suitable" smoothness bound $B$. With Algorithm 3.1, we can produce a list of all prime numbers $\leq B$ and we label them consecutively starting with $p_2$; that is, we set $p_2 = 2, p_3 = 3, p_4 = 5$, $p_5 = 7$,... (up to $p_r$). With $p_1 := -1$, the $p_i$ form a *factor basis* (w.r.t. $B$); that is, we can write every $B$-powersmooth number as a product of the $p_i$. Then we may determine a $y$ with the desired properties using the following procedure:

(a) Find "enough" $a_1, \ldots, a_m \in \mathbb{Z}$ (called *$B$-numbers*), such that $(a_i^2 \mod n) = \prod_{j=1}^r p_j^{e_{i,j}}$ for some $e_{i,j} \in \mathbb{N}$.

(b) If the vectors $(\overline{e}_{i,1}, \ldots, \overline{e}_{i,r}) \in \mathbb{F}_2^r$, $1 \leq i \leq m$ are linearly dependent, then there are $\mu_1, \ldots, \mu_m \in \{0, 1\} \subset \mathbb{Z}$ with some $\mu_i \neq 0$, such that $\sum_{i=1}^m \mu_i e_{i,j} = 2k_j$ for some $k_j \in \mathbb{N}$. Setting

$$x := \prod_{i=1}^m a_i^{\mu_i} \mod n, \qquad y := \prod_{j=1}^r p_j^{k_j} \mod n,$$

we get the desired equality:

$$x^2 = \prod_{i=1}^m a_i^{2\mu_i} \equiv \prod_{i=1}^m \prod_{j=1}^r p_j^{\mu_i e_{ij}} = \prod_{j=1}^r p_j^{\sum_{i=1}^m \mu_i e_{i,j}} = \prod_{j=1}^r p_j^{2k_j} = y^2 \mod n.$$

To obtain $x \not\equiv \pm y \mod n$, we should choose the $a_i$ not too small.

Therefore, the final step is to find a good way to determine $B$-numbers. We try numbers of the form $t + \lfloor\sqrt{n}\rfloor$ with $s \in \mathbb{R}$ and $t \in [-s, s] \cap \mathbb{Z} =: I$ (called *sieve interval*). Writing $(a \mod n)$ for the unique integer $x \in \mathbb{Z}$ with $-\frac{n}{2} < x \leq \frac{n}{2}$ and $x \equiv a \mod n$ for $a \in \mathbb{Z}$, we observe that for small enough $s$, we have

$$\left(\left(t + \lfloor\sqrt{n}\rfloor\right)^2 \mod n\right) = \left(t + \lfloor\sqrt{n}\rfloor\right)^2 - n =: f(t),$$

so

$$p_j^{e_j} \mid f(t) \iff \left(t + \lfloor\sqrt{n}\rfloor\right)^2 \equiv n \mod p_j^{e_j}. \qquad (*)$$

Note that $(*)$ can only hold if $n$ is a square number $\mod p_j^{e_j}$, so in particular $n$ must be a square number $\mod p_j$. Therefore, we may delete all prime numbers from our factor base that do not have this property. Furthermore, if $(*)$ is satisfied for some $t$, then it also holds true for all elements of $t + p_j^{e_j}\mathbb{Z}$. We thus obtain the following "sieving procedure": After having found $f(t)$ with $p_j^{e_j} \mid f(t)$, mark all elements of $t + p_j^{e_j}\mathbb{Z} \cap I$ and iterate.

The above observations give rise to the following Las-Vegas algorithm.

**Algorithm 3.11 (Quadratic Sieve).**

```
Input: $n \in \mathbb{N}_{>1}$ odd composite number.
Output: A proper divisor $d \in \mathbb{N}$ of $n$.
  (1) If $n = m^e$ for some $m, e \in \mathbb{N}_{>1}$ (use Algorithm 1.41): return $m$.
  (2) Choose a “smoothness bound” $B \in \mathbb{N}$ and a “sieve bound” $s \in \mathbb{N}$.
  (3) Let $p_1 \coloneqq -1$, $p_2, \dots, p_r$ be a factor basis for $B$ as above, excluding those $p_i$ for
      which $n$ is not a square $\mod p_i$.
  (4) For $t = -s, -s+1, \dots, s-1, s$: compute $f_t \coloneqq \left|(t + \lfloor\sqrt{n}\rfloor)^2 - n\right| \in \mathbb{N}$.
  (5) For $t = -s, \dots, s$: set $e_t = (0, \dots, 0) \in \mathbb{N}^r$.
  (6) For $t = -s, \dots, 0$: set $e_{t,1} \coloneqq 1$.
  (7) For $j = 2, \dots r$:
      (8) For $e = 1, \dots, \lfloor \log_{p_j}(B) \rfloor$:
          (9) Find all solution $(t_1 \mod p_j^e), \dots, (t_m \mod p_j^e)$ of $(t + \lfloor\sqrt{n}\rfloor)^2 \equiv n$
              $\mod p_j^e$.
          (10) For $t \in (t_i + p_j^e\mathbb{Z}) \cap \mathbb{Z}$ for some $i \in \{1, \dots, m\}$: set $f_t \coloneqq \frac{f_t}{p_j}$ and $e_{t,j} \coloneqq$
               $e_{t,j} + 1$.
  (11) Let $t_1, \dots, t_m \in I$ be those $t \in \{-s, \dots, s\}$, such that $f_t = 1$.
  (12) Set $a_i \coloneqq t_i + \lfloor\sqrt{n}\rfloor$ for $i \in \{1, \dots, m\}$.
  (13) If the vectors $\{(e_{t_i} \mod 2) \in \mathbb{F}_2^r : 1 \le i \le m\}$ are linearly independent, restart.
       Otherwise, compute $\mu_1, \dots, \mu_m \in \{0,1\}$ not all zero and $k_j \in \mathbb{N}$, such that
       $$\sum_{i=1}^{m} \mu_i e_{t_i,j} = 2 \cdot k_j \qquad \forall\ 1 \le j \le r.$$
  (14) Set
       $$x \coloneqq \prod_{i=1}^{m} a_i^{\mu_i} \mod n, \qquad y \coloneqq \prod_{i=1}^{r} p_j^{k_j} \mod n.$$
  (15) If $\gcd(n, x-y)$ or $\gcd(n, x+y)$ is a proper divisor of $n$, return it. Otherwise,
       restart.
```

Lec 16
2021-12-14

While the algorithm is justified by the considerations before, we still provide some additional comments.

(6) The $t \in \{-s, \dots, 0\}$ are precisely those $t \in \{-s, \dots, s\}$ for which the term in the absolute value in the definition of $f_t$ is negative. Therefore, the factor basis element $-1$ is needed to represent those numbers.

(9) We will see that $m \in \{0, 2, 4\}$ and usually $m = 2$.

(11) For those $t_i$, the $a_i$ as defined in step (12) are $B$-numbers and the $e_{t,i}$ are the correct exponent vectors for the factorization of $(a_i^2 \mod n)$.

To demonstrate the algorithm "in action", we calculate an extensive example.

**Example 3.12.** Let $n = 20437$ and choose $B = 10$, $s = 3$. A corresponding factor basis is $p_1 = -1, p_2 = 2, p_3 = 3, p_4 = 7$, where 5 was excluded, because $n \equiv 2 \mod 5$ is not a square. Because $\lfloor\sqrt{n}\rfloor = 142$, we need to solve $(t + 142)^2 \equiv n \mod p_j^e$ (step (9) in the algorithm).

$p_2 = 2$: $n \equiv 5 \mod 8$ is not a square, but $n \equiv 1 \mod 4$ (and also $\mod 2$) is one with square root $\pm 1 \mod 4$. If $t$ is even, then $f(t)$ is odd and otherwise $4 \mid f(t)$, so $e_{t,2} = 0$ if $t$ is even and $e_{t,2} = 2$ otherwise.

$p_3 = 3$: We have $n \equiv 1 \mod 3$ and $\lfloor\sqrt{n}\rfloor \equiv 1 \mod 3$, so

$$3 \mid f(t) \iff t + 1 \equiv \pm 1 \mod 3 \iff (t \equiv 0 \mod 3) \vee (t \equiv 1 \mod 3).$$

Similarly, $n \equiv 7 \equiv (\pm 4)^2 \mod 9$ and $\lfloor\sqrt{n}\rfloor \equiv 7 \mod 9$, so

$$9 \mid f(t) \iff t+7 \equiv \pm 4 \mod 9 \iff (t \equiv -2 \mod 9) \vee (t \equiv -3 \mod 9).$$

$p_4 = 7$: We calculate $n \equiv 4 \equiv (\pm 2)^2 \mod 7$ and $\lfloor\sqrt{n}\rfloor \equiv 2 \mod 7$, so

$$7 \mid f(t) \iff t + 2 \equiv \pm 2 \mod 7 \iff (t \equiv 0 \mod 7) \vee (t \equiv 3 \mod 7).$$

We can now perform the "sieving" steps:

| t | -3 | -2 | -1 | 0 | 1 | 2 | 3 |
|---|---|---|---|---|---|---|---|
| $f_t$ | 1116 | 837 | 556 | 273 | 12 | 299 | 588 |
| 1st component of $e_t$ | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 2nd component of $e_t$ | 2 | 0 | 2 | 0 | 2 | 0 | 2 |
| new $f_t$ (i.e. $f_t$ divided by powers of 2) | 279 | 837 | 139 | 273 | 3 | 299 | 147 |
| 3rd component of $e_t$ | 2 | 2 | 0 | 1 | 1 | 0 | 1 |
| new $f_t$ | 31 | 93 | 139 | 91 | 1 | 299 | 49 |
| 4th component of $e_t$ | 0 | 0 | 0 | 1 | 0 | 0 | 2 |
| new $f_t$ | 31 | 93 | 139 | 13 | 1 | 299 | 1 |

Therefore, we obtain two $B$-numbers. For $t = 1$, the corresponding exponent is $e_1 = (0, 2, 1, 0)$ and for $t = 3$, it is $e_3 = (0, 2, 1, 2)$. These two vectors are linearly dependent mod 2, as

$$e_1 + e_3 = 2 \cdot (0, 2, 1, 1) = 2 \cdot (k_1, k_2, k_3, k_4),$$

so $\mu_1 = 1$ and $\mu_3 = 1$. It follows $x = (1 + 142)(3 + 142) \equiv 298 \mod n$ and $y = p_2^2 p_3 p_4 = 2^2 \cdot 3 \cdot 7 = 84$. The divisors returned by the algorithm are thus $\gcd(n, x + y) = 191$ and $\gcd(n, x - y) = 107$ and since $n = 107 \cdot 191$, this is even the complete factorization.

Step (9) of the algorithm requires computing square roots in $\mathbb{Z}/(p^e)$; i.e. solving $x^2 \equiv n \mod p^e$ for $x \in \mathbb{Z}/(p^e)$. We discuss a systematic way to compute these solutions, called *Hensel lifting*.

$p \neq 2$: Find the two solutions of $x^2 = n \in \mathbb{F}_p$ using brute force. Then we may inductively "lift" such a solution $\mod p$ to a solution $\mod p^e$ as follows: For $k \geq 1$, assume we already have $x \in \mathbb{Z}$ with $x^2 \equiv n \mod p^k$, so there is $r \in \mathbb{Z}$ with $x^2 - n = r \cdot p^e$. Then for $y \in \mathbb{Z}$, we have

$$\left(x + p^k y\right)^2 - n = x^2 + 2xyp^k + p^{2k}y^2 - n \equiv (2xy + r)p^k \mod p^{k+1},$$

so by solving the equation $2xy \equiv -r \mod p$ for $y$, we obtain a solution for $x^2 \equiv n \mod p^{k+1}$.

$p = 2$: Find $x \in \mathbb{Z}$, such that $x^2 \equiv n \mod 8$ using brute force. Because $n$ is odd, this has either four or no solutions. If we have $x \in \mathbb{Z}$ with $x^2 \equiv n \mod 2^k$ and $k \geq 3$, then there is $r \in \mathbb{Z}$, such that $x^2 - n = r \cdot 2^k$. Thus for $y \in \mathbb{Z}$, we have

$$\left(x + 2^{k-1}y\right)^2 - n = x^2 + 2^k xy + y^2 2^{2k-2} - n \equiv (r + xy)2^k \mod 2^{k+1},$$

so by choosing $y \equiv r \mod 2$ ($x$ is necessarily odd), we may inductively construct the solution.

It can be shown that by choosing $B$ and $s$ close to $\exp\left(\sqrt{\frac{\ln(n)\cdot\ln(\ln(n))}{2}}\right)$, then the expected running time of Algorithm 3.11 is $\mathcal{O}(\exp\left(\sqrt{\ln(n) \cdot \ln(\ln(n))}\right))$.

# 4 A Computational View towards Linear Algebra

Lec 17
2021-12-16

## 4.1 Complexity of Operations from Linear Algebra

In this section, investigate the complexity of common operations from linear algebra like solving systems of linear equations, inverting matrices, determining the rank of a matrix, calculating its determinant and performing matrix multiplication. Throughout this section, $\mathbb{K}$ denotes a field. Instead of counting in bit-operations, we count the cost in field operations.

**Proposition 4.1.**

(a) Solving a homogeneous $m \times n$ linear equation by *Gaussian elimination* requires $\mathcal{O}(\max\{m, n\}^3)$ field operations.
If the system is inhomogeneous, we need $\mathcal{O}(\max\{m, n+1\}^3)$ field operations.

(b) For $A \in \mathrm{GL}_n(\mathbb{K})$, computing the inverse $A^{-1}$ by the "usual" method with Gaussian elimination requires $\mathcal{O}(n^3)$ field operations.

(c) For $A \in \mathbb{K}^{n \times n}$, computing $\det(A)$ requires $\mathcal{O}(n^3)$ field operations.

(d) For $A \in \mathbb{K}^{m \times n}$ and $B \in \mathbb{K}^{n \times l}$, computing their matrix product $A \cdot B$ in the obvious way takes $\mathcal{O}(m \cdot n \cdot l)$ field operations.

*Proof.*

(a) Let $r \coloneqq \operatorname{rank}(A)$. Using Gaussian elimination, we transform the matrix into strict row echelon form. Therefore, for $1 \leq k \leq r$, we need to divide the $k$-th row by its first nonzero entry (this amounts to $\mathcal{O}(n)$ field operations) and subtract it from all rows except itself (requiring $\mathcal{O}(m \cdot n)$ field operations). Thus, for each $1 \leq k \leq r$, we need $\mathcal{O}(n + m \cdot n)$ field operations, which amounts to $\mathcal{O}(\max(n, m)^3)$ field operations in total. Determining a basis for the solution space is subsumed under this expression.
Solving an inhomogeneous system essentially amounts to solving the homogeneous $m \times (n+1)$ system obtained by interpreting the inhomogeneous "solution vector" as the last column of the matrix.

(b) Computing the inverse can be reduced to solving a linear system of size $n \times 2n$, so the assertion follows from (a).

(c) To compute $\det(A)$, we transform $A$ into strict row echelon form and multiply the entries on the main diagonal, so the claim again follows from (a).

(d) Each of the $m \cdot l$ entries of the resulting matrix is obtained by calculating the dot product of a row of $A$ with a column of $B$. □

Because of this result, we say that these basic computations from linear algebra have cubic complexity (in $\max\{m, n\}$). Note that this is somewhat inprecise, because the input length is $\mathcal{O}(\max\{m, n\}^2)$.

## 4.2 Strassen Multiplication

It turns out that the naive way to multiply two matrices is not the most efficient algorithm for this task. One improvement is *Strassen multiplication*. The basic idea behind the algorithm is that it can be useful to split a matrix into submatrices ("block matrices"). Indeed, for two matrices $A, B \in \mathbb{K}^{2n\times 2n}$, we can write

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} \quad \text{with} \quad A_{i,j}, B_{i,j} \in \mathbb{K}^{n\times n}.$$

Then

$$A \cdot B = C = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix} \quad \text{with} \quad C_{i,j} = A_{i,1}B_{1,j} + A_{i,2}B_{2,j} \in \mathbb{K}^{n\times n}.$$

Similar to Karatsuba multiplication for integers (see Algorithm 1.8), we can reduce the eight multiplications needed for the above calculation by one using the following trick. Defining seven (temporary) matrices, $M_1, \ldots, M_7$

$$\begin{aligned} M_1 &= (A_{1,2} - A_{2,2}) \cdot (B_{2,1} + B_{2,2}), & M_4 &= (A_{1,1} + A_{1,2}) \cdot B_{2,2}, \\ M_2 &= (A_{1,1} + A_{2,2}) \cdot (B_{1,1} + B_{2,2}), & M_5 &= (A_{2,1} + A_{2,2}) \cdot B_{1,1}, \\ M_3 &= (A_{1,1} - A_{2,1}) \cdot (B_{1,1} + B_{1,2}), & M_6 &= A_{1,1} \cdot (B_{1,2} - B_{2,2}), \\ & & M_7 &= A_{2,2} \cdot (B_{2,1} - B_{1,1}). \end{aligned}$$

explicit caluclations show that one can obtain $C_{i,j}$ by

$$\begin{aligned} C_{1,1} &= M_1 + M_2 - M_4 + M_7, & C_{1,2} &= M_4 + M_6, \\ C_{2,1} &= M_5 + M_7, & C_{2,2} &= M_2 - M_3 - M_5 + M_6. \end{aligned}$$

This caluclation requires seven multiplications and eighteen additions of $n \times n$ matrices. Since addition has quadratic runtime and multiplication cubic, this is asymptotically superior to eight multiplications. This observation translates to the following algorithm.

**Algorithm 4.2.**

Input: $A \in \mathbb{K}^{m\times n}$, $B \in \mathbb{K}^{n\times l}$.

Output: $A \cdot B \in \mathbb{K}^{m\times l}$.

(1) Find $k \in \mathbb{N}$ minimal, such that $m, n, l \leq 2^k$.

(2) If $k = 0$: return $A \cdot B$ (multiplication in $\mathbb{K}$).

(3) Add zeros to $A$ and $B$ in order to turn them into $\mathbb{K}^{2^k\times 2^k}$ matrices; that is, let

$$A' := \begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{K}^{2^k\times 2^k} \quad \text{and} \quad B' := \begin{pmatrix} B & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{K}^{2^k\times 2^k}.$$

(4) Write $A' = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix}$ and $B' = \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix}$ with $A_{i,j}, B_{i,j} \in \mathbb{K}^{2^{k-1}\times 2^{k-1}}$.

(5) Recursively call this algorithm to compute $M_1, \ldots, M_7$ and $C_{1,1}, C_{1,2}, C_{2,1}, C_{2,2}$ as defined above.

(6) Write $\begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix} = \begin{pmatrix} D_{1,1} & D_{1,2} \\ D_{2,1} & D_{2,2} \end{pmatrix}$ with $D_{1,1} \in \mathbb{K}^{m \times l}$ and return $D_{1,1}$.

**Theorem 4.3.** For matrices $A \in \mathbb{K}^{m\times n}$, $B \in \mathbb{K}^{n\times l}$ Algorithm 4.2 requires

$$\mathcal{O}(\max\{m,n,l\}^{\log_2(7)})$$

field operations in order to compute $A \cdot B$.

*Proof.* Define $k$ as in step (1) and consider

$$\Theta(k) = \max\big\{\text{number of field operations required for } A \in \mathbb{K}^{m\times n},\ B \in \mathbb{K}^{n\times l}\big\}.$$

The only step in the algorithm requiring field operations is step (5), so from the above, we obtain

$$\Theta(k) = 7\Theta(k-1) + 18\big(2^{k-1}\big)^2.$$

*Claim:* $\Theta(k) = 7^{k+1} - 6 \cdot 4^k$.
We prove this by induction on $k$.
The base case $k = 0$ holds true, since in that case the involved matrices are just numbers, implying $\Theta(0) = 1$.
The inductive step follows from the calculation

$$\begin{aligned} \Theta(k) &= 7\Theta(k-1) + 18\big(2^{k-1}\big)^2 \\ &= 7\big(7^k - 6 \cdot 4^{k-1}\big) + 18 \cdot 4^{k-1} \\ &= 7^{k+1} + 4^{k-1}(18 - 42) \\ &= 7^{k+1} - 24 \cdot 4^{k-1} \\ &= 7^{k+1} - 6 \cdot 4^k. \end{aligned}$$

By definition of $k$, we have $2^{k-1} < r$ for $r := \max\{m,n,l\}$, so the assertion follows from the claim:

$$\Theta(k) < 7^{k+1} < 7^{\log_2(r)+1+1} = 49 \cdot 7^{\log_2(r)} = 49 \cdot 2^{\log_2(7)\log_2(r)} = 49 \cdot r^{\log_2(7)}. \qquad \square$$

Because $\log_2(7) \approx 2.807$, this is indeed an improvement of the naive algorithm.

## 4.3 Common Operations as Matrix Multiplication

Lec 18
2021-12-21

The goal of this section is to reduce common operations like determining the rank or determinant of a matrix to matrix multiplication. In this way, better algorithms for matrix multiplication also allow us to compute these other operations more efficiently.

Let $M\colon \mathbb{N}_{>0} \to \mathbb{R}_{>0}$ be a monotonic function such that two matrices in $\mathbb{K}^{n\times n}$ can be multiplied in at most $M(n)$ field operations. We assume that there exists $\epsilon > 0$ satisfying

$$2^{2+\epsilon} \cdot M(n) \le M(2n) \le 8 \cdot M(n) \qquad \forall\, n \in \mathbb{N}_{>0}, \tag{$*$}$$

which is true for all currently known multiplication algorithms.
Intuitively, this means that we assume matrix multiplication to be more than quadratic in $n$ and at most cubic.

In particular, $(*)$ and the monotonicity imply that for all $n \in \mathbb{N}$ with $n \geq \left(\frac{2^{2+\epsilon}}{M(1)}\right)^{\frac{1}{\epsilon}}$, we have

$$\begin{aligned} n^2 = \frac{n^2}{M(1)} \cdot M(1) &\leq \frac{n^2}{M(1) \cdot 2^{(2+\epsilon)\lfloor \log_2(n) \rfloor}} \cdot M(2^{\lfloor \log_2(n) \rfloor}) \\ &\leq \frac{2^{2+\epsilon}}{M(1) \cdot n^{\epsilon}} \cdot M(2^{\lfloor \log_2(n) \rfloor}) \leq M(2^{\lfloor \log_2(n) \rfloor}) \leq M(n). \qquad (**) \end{aligned}$$

By adjusting the value of $M$ for all smaller $n$, we may assume that $(**)$ holds for all $n \in \mathbb{N}$. This is also intuitive in the sense that any multiplication algorithm would certainly need to read the input and output its result, which is already quadratic in $n$.
We also note that by adding zeros to a non-quadratic matrix, the product of any two matrices $A \in \mathbb{K}^{m \times n}$, $B \in \mathbb{K}^{n \times l}$ can be computed with at most $M(\max\{m, n, l\})$ field operations.

The first step towards our goal is to observe that inverting a triangular matrix can be computed by multiplying some suitable matrices.

**Proposition 4.4.** The inverse $A^{-1}$ of an invertible triangular matrix $A \in \mathrm{GL}_n(\mathbb{K})$ can be computed in $\mathcal{O}(M(n))$ field operations.

*Proof.* Because the operations of transposing and inverting a matrix commute, we may assume that $A$ is an upper triangular matrix. Let $k \in \mathbb{N}$ be minimal, such that $n \leq 2^k$ and form

$$B := \begin{pmatrix} A & 0 \\ 0 & I_{2^k - n} \end{pmatrix} = \begin{pmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{pmatrix} \in \mathbb{K}^{2^k \times 2^k} \quad \text{with } B_{ij} \in \mathbb{K}^{2^{k-1} \times 2^{k-1}}.$$

It is easy to verify that

$$B^{-1} = \begin{pmatrix} {B_{11}}^{-1} & -{B_{11}}^{-1} B_{12} {B_{22}}^{-1} \\ 0 & {B_{22}}^{-1} \end{pmatrix}$$

and we can recursively calculate ${B_{11}}^{-1} \in \mathbb{K}^{2^{k-1} \times 2^{k-1}}$ and ${B_{22}}^{-1} \in \mathbb{K}^{2^{k-1} \times 2^{k-1}}$.
To calculate the complexity, let

$$\Theta(k) = \max\big\{\text{number of field operations required for } A \in \mathbb{K}^{n \times n}\big\}$$

and note that

$$\Theta(k) \leq 2 \cdot \Theta(k-1) + 2 \cdot M\big(2^{k-1}\big).$$

*Claim:* $\Theta(k) \leq 2^k + M\big(2^k\big)$.
We prove this by induction on $k$. The base case $k = 0$ is clear, so assume that the claim holds for $k-1$ and calculate

$$\begin{aligned} \Theta(k) &\leq 2 \cdot \Theta(k-1) + 2 \cdot M\big(2^{k-1}\big) \\ &\leq 2 \cdot \big(2^{k-1} + M\big(2^{k-1}\big)\big) + 2 \cdot M\big(2^{k-1}\big) \\ &\overset{(*)}{<} 2^k + M\big(2^k\big). \end{aligned}$$

Because $2^{k-1} < n$, the assertion follows from the claim, $(*)$ and $(**)$:

$$\Theta(k) \leq 2^k + M\big(2^k\big) \overset{(*)}{<} 2n + M(2n) \overset{(*)}{\leq} 2n + 8M(n) \in \mathcal{O}(M(n)).$$

□

We can now state the central algorithm of this section, which allows us to transform a given matrix into simpler ones.

**Algorithm 4.5.**

Input: $A \in \mathbb{K}^{m\times n}$

Output: $r = \operatorname{rank}(A)$ and matrices $L$, $Q$, $P$, $U$, such that

$$L \cdot Q \cdot A \cdot P = \begin{array}{cc} \begin{pmatrix} U \\ 0 \end{pmatrix} & \begin{array}{c} r \\ m-r \end{array} \\ n & \end{array} \in \mathbb{K}^{m\times n}$$

and

(a) $L \in \mathbb{K}^{m\times m}$ is a lower triangular matrix with ones on the diagonal;
(b) $Q \in \mathbb{K}^{m\times m}$ is a permutation matrix;
(c) $P \in \mathbb{K}^{n\times n}$ is a permutation matrix;
(d) $U \in \mathbb{K}^{r\times n}$ is an upper triangular matrix with non-zero diagonal entries.

Additionally, if $\operatorname{rank}(A) = m$, then $Q = I_m$.

(1) If $A = \begin{pmatrix} 0 & \cdots & 0 \end{pmatrix} \in \mathbb{K}^{1\times n}$: return $L = (1) \in \mathbb{K}^{1\times 1}$, $Q = (1) \in \mathbb{K}^{1\times 1}$, $P = I_n \in \mathbb{K}^{n\times n}$, $U \in \mathbb{K}^{0\times n}$.

(2) If $A = \begin{pmatrix} a_1 & \cdots & a_n \end{pmatrix} \in \mathbb{K}^{1\times n}$: Find $i \in \{1, \ldots, n\}$ minimal such that $a_i \neq 0$ and let $P$ denote the matrix in $\mathbb{K}^{n\times n}$ corresponding to the transposition $\begin{pmatrix} 1 & i \end{pmatrix}$. Return $P$, $L = (1) \in \mathbb{K}^{1\times 1}$, $Q = (1) \in \mathbb{K}^{1\times 1}$, $U = A \cdot P$.

(3) Set $m_1 := \lfloor \frac{m}{2} \rfloor$, $m_2 := \lceil \frac{m}{2} \rceil$ and write

$$A = \begin{array}{cc} \begin{pmatrix} B \\ C \end{pmatrix} & \begin{array}{c} m_1 \\ m_2 \end{array} \\ n & \end{array} .$$

(4) By recursively applying this algorithm to $B \in \mathbb{K}^{m_1\times n}$, compute $L_1, Q_1, P_1$ with

$$L_1 \cdot Q_1 \cdot B \cdot P_1 = \begin{array}{cc} \begin{pmatrix} U_1 \\ 0 \end{pmatrix} & \begin{array}{c} r_1 \\ m_1 - r_1 \end{array} \\ n & \end{array} .$$

(5) Write

$$L_1 := \begin{array}{cc} \begin{pmatrix} L_t & 0 \\ L_l & L_r \end{pmatrix} & \begin{array}{c} r_1 \\ m_1 - r_1 \end{array} \\ \begin{array}{cc} r_1 & m_1 - r_1 \end{array} & \end{array}, \qquad Q_1 := \begin{array}{cc} \begin{pmatrix} Q_t \\ Q_b \end{pmatrix} & \begin{array}{c} r_1 \\ m_1 - r_1 \end{array} \\ m_1 & \end{array}, \qquad U_1 := \begin{array}{cc} \begin{pmatrix} E & U_1' \end{pmatrix} & r_1 \\ \begin{array}{cc} r_1 & m - r_1 \end{array} & \end{array}$$

and compute

$$D := CP_1 = \begin{pmatrix} F & D' \end{pmatrix}, \qquad G := D' - FE^{-1}U_1' \in \mathbb{K}^{m_2\times(n-r_1)}.$$

(6) By recursively applying this algorithm to $G \in \mathbb{K}^{m_2 \times (n - r'_1)}$, calculate $L_2$, $Q_2$, $P_2$, such that

$$L_2 \cdot Q_2 \cdot G \cdot P_2 = \begin{array}{cc} \begin{pmatrix} U_2 \\ 0 \end{pmatrix} & \begin{array}{c} r_2 \\ m_2 - r_2 \end{array} \\ n - r_1 & \end{array} .$$

(7) Return

$$L = \begin{array}{cccc} \left( \begin{array}{ccc} L_t & 0 & 0 \\ -L_2 Q_2 F E^{-1} L_t & L_2 & 0 \\ L_l & 0 & L_r \end{array} \right) & \begin{array}{c} r_1 \\ m_2 \\ m_1 - r_1 \end{array} \\ \begin{array}{ccc} r_1 & m_2 & m_1 - r_1 \end{array} & \end{array} , \qquad Q = \begin{array}{cc} \left( \begin{array}{cc} Q_t & 0 \\ 0 & Q_2 \\ Q_b & 0 \end{array} \right) & \begin{array}{c} r_1 \\ m_2 \\ m_1 - r_1 \end{array} \\ \begin{array}{cc} m_1 & m_2 \end{array} & \end{array} ,$$

$$P = P_1 \cdot \begin{array}{cc} \begin{pmatrix} I_{r_1} & 0 \\ 0 & P_2 \end{pmatrix} & \begin{array}{c} r_1 \\ n - r_1 \end{array} \\ \begin{array}{cc} r_1 & n - r_1 \end{array} & \end{array} , \qquad U = \begin{array}{cc} \begin{pmatrix} E & U'_1 P_2 \\ 0 & U_2 \end{pmatrix} & \begin{array}{c} r! \\ r_2 \end{array} \\ \begin{array}{cc} r_1 & n - r_1 \end{array} & \end{array} .$$

*Proof (of correctness).* We prove the correctness by induction on $m$. The base case $m = 1$ (corresponding to the steps (1) and (2) of the algorithm) is straightforward to check. Thus we may assume $m > 1$ and that all recursive calls are correct. Then we have

$$\begin{aligned} L \cdot Q \cdot A \cdot P &= \begin{pmatrix} L_t Q_t & 0 \\ -L_2 Q_2 F E^{-1} L_t Q_t & L_2 Q_2 \\ L_l Q_t + L_r Q_b & 0 \end{pmatrix} \cdot \begin{pmatrix} B P_1 \\ D \end{pmatrix} \cdot \begin{pmatrix} I_{r_1} & 0 \\ 0 & P_2 \end{pmatrix} \\ &\overset{(4)}{=} \begin{array}{cc} \begin{pmatrix} U_1 \\ L_2 Q_2 (D - F E^{-1} U_1) \\ 0 \end{pmatrix} & \begin{array}{c} r_1 \\ m_2 \\ m_1 - r_1 \end{array} \\ n & \end{array} \cdot \begin{pmatrix} I_{r_1} & 0 \\ 0 & P_2 \end{pmatrix} \\ &= \begin{pmatrix} E & U'_1 P_2 \\ 0 & L_2 Q_2 G P_2 \\ 0 & 0 \end{pmatrix} \overset{(6)}{=} \begin{array}{cc} \begin{pmatrix} U \\ 0 \end{pmatrix} & \begin{array}{c} r_1 + r_2 \\ m_1 + m_2 \end{array} \\ n & \end{array} . \end{aligned}$$

Additionally, if $r = m$, then $r_1 = m_1$ and $r_2 = m_2$ by the form of $U$. Thus by the inductive hypothesis, we have $Q_1 = I_{m_1}$ and $Q_2 = I_{m_2}$. Therefore, $Q_t = Q_1$ and $Q_b \in \mathbb{K}^{0 \times r_1}$, so $Q = I_m$ by definition of $Q$.
Finally, that $r = \operatorname{rank}(A)$ is part of Theorem 4.7. □

Lec 19
2021-12-23

**Theorem 4.6.** Algorithm 4.5 requires $\mathcal{O}(\left(\frac{n}{m} + 1\right) M(m))$ field operations for $A \in \mathbb{K}^{m \times n}$.

*Proof.* Fix $n_0 \in \mathbb{N}_{>0}$ and consider for $k \in \mathbb{N}$

$$\Theta(k) = \max\left\{\text{number of field operations required for } A \in \mathbb{K}^{m \times n} \text{ with } m \leq 2^k,\ n \leq n_0\right\}.$$

For each $k$, we may choose $A \in \mathbb{K}^{m\times n}$ such that it achieves the maximum $\Theta(k)$ (i.e. it is a "worst case" input of the algorithm). By definition, $m_1, m_2 \le 2^{k-1}$. The only steps in the algorithm that involve field operations are the steps (4)-(7), so we investigate their complexity.

(4) Because $m_1 \le 2^{k-1}$, the recursive call requires at most $\Theta(k-1)$ field operations.

(5) By Proposition 4.4, inverting $E$ requires $\mathcal{O}(M(2^{k-1}))$ field operations. Furthermore, computing the products $C \cdot P_1$ and $F \cdot E^{-1}$ needs at most $M(2^{k-1})$ field operations each. By splitting $U_1'$ into $\lceil \frac{n-r_1}{2^{k-1}} \rceil$ blocks of size $r_1 \times 2^{k-1}$ (the last block might be smaller), we can compute the product $(FE^{-1}) \cdot U_1'$ in at most $\lceil \frac{n-r_1}{2^{k-1}} \rceil \cdot M(2^{k-1}) \le \big(2^{1-k}n + 1\big) \cdot M(2^{k-1})$ field operations.
Calculating the difference of two $m_2 \times (n - r_1)$ matrices as in the definition of $G$ amounts to

$$m_2(n-r_1) \le 2^{k-1}n \le 2^{1-k}n \cdot M(2^{k-1})$$

field operations, because $2^{2k-2} \le M(2^{k-1})$ by $(**)$.

(6) Since $m_2 \le 2^{k-1}$, the recursive call needs at most $\Theta(k-1)$ field operations.

(7) Note that $F \cdot E^{-1}$ has already been computed in step (5) and that any product where one of the two matrices is a permutation matrix does not demand any field operations. The remaining two products of matrices of size $\le 2^{k-1}$ amount to at most $2 \cdot M(2^{k-1})$ field operations and multiplying the resulting $m_2 \times r_1$ matrix by $-1$ needs $m_2 \cdot r_1 \le (2^{k-1})^2 \overset{(**)}{\le} M(2^{k-1})$ field operations.

Therefore, there exists a constant $C \in \mathbb{R}_{>0}$, such that for all $k \in \mathbb{N}_{>0}$, the total cost is upper bounded by

$$\Theta(k) \le 2\Theta(k-1) + \Big(\big(2^{1-k}n + 1\big) + 2^{1-k}n + C\Big) \cdot M(2^{k-1}).$$

According to $(*)$, we have $M(2^{k-1}) < \frac{1}{4}M(2^k)$, so by replacing $C$ with $C+1$, we obtain

$$\Theta(k) \le 2\Theta(k-1) + \big(2^{-k}n + C\big) \cdot M(2^k).$$

*Claim:* $\Theta(k) \le \Big(2^{-k}n\frac{1-2^{-k\epsilon}}{1-2^{-\epsilon}} + 2C\big(1-2^{-k}\big)\Big) \cdot M(2^k)$.
We prove this by induction on $k$. The base case $k=1$ is correct as $\Theta(0) = 0$ by step (1) and (2) of the algorithm.
Assuming the assertion holds true for $k-1$, we use the inductive hypothesis and $(*)$ in the form $M(2^{k-1}) \le 2^{-2-\epsilon} \cdot M(2^k)$ to calculate:

$$\begin{aligned}
\Theta(k) &\le 2\Theta(k-1) + \big(2^{-k}n + C\big) \cdot M(2^k) \\
&\le \left(2^{-k+1}n\frac{1-2^{-k\epsilon+\epsilon}}{1-2^{-\epsilon}} + 2C\big(1-2^{-k+1}\big)\right) \cdot M(2^{k-1}) + \big(2^{-k}n + C\big) \cdot M(2^k) \\
&\le \left(2^{-k-1-\epsilon}n\frac{1-2^{-k\epsilon+\epsilon}}{1-2^{-\epsilon}} + 2^{-1-\epsilon}C\big(1-2^{-k+1}\big) + 2^{-k}n + C\right) \cdot M(2^k) \\
&\le \left(2^{-k}n\left(2^{-1-\epsilon}\frac{1-2^{-k\epsilon+\epsilon}}{1-2^{-\epsilon}} + 1\right) + 2C\left(2^{-2-\epsilon}\big(1-2^{-k+1}\big) + \frac{1}{2}\right)\right) \cdot M(2^k).
\end{aligned}$$

Now the claim follows from the following three estimations:

$$2^{-1-\epsilon}\frac{1-2^{-k\epsilon+\epsilon}}{1-2^{-\epsilon}}+1 = 2^{-1-\epsilon}\cdot\frac{2-2^{-k\epsilon+\epsilon}-2^{-\epsilon}}{1-2^{-\epsilon}} = \frac{1}{2}\cdot\frac{2^{1-\epsilon}-2^{-k\epsilon}-2^{-2\epsilon}}{1-2^{-\epsilon}},$$

$$2^{1-\epsilon}-2^{-2\epsilon} = -\left(2^{-\epsilon}\right)^2+2\cdot 2^{-\epsilon}-1+1 = -\left(2^{-\epsilon}-1\right)^2+1\le 1,$$

$$2^{-2-\epsilon}\left(1-2^{-k+1}\right)+\frac{1}{2} = 2^{-(1+\epsilon)}\left(\frac{1}{2}-2^{-k}\right)+\frac{1}{2} < \frac{1}{2}-2^{-k}+\frac{1}{2} = 1-2^{-k}.$$

Having established the claim, we can now finish the proof. To that end, let $B\in\mathbb{K}^{m\times n}$ be an arbitrary matrix and choose $k\in\mathbb{N}$ minimal such that $m\le 2^k$. Then by the claim the maximal number of field operations for $B$ is upper bounded by

$$\begin{aligned}\Theta(k) &\le \left(2^{-k}n\frac{1}{1-2^{-\epsilon}}+2C\right)M(2^k)\\ &\le \left(\frac{n}{m}\frac{1}{1-2^{-\epsilon}}+2C\right)M(2m)\\ &\overset{(*)}{\le} 8\max\left\{\frac{1}{1-2^{-\epsilon}},2C\right\}\left(\frac{n}{m}+1\right)M(m)\in\mathcal{O}\left(\left(\frac{n}{m}+1\right)M(m)\right).\end{aligned}$$

□

We note that a collection of matrices as in Algorithm 4.5 gives rise to the decomposition

$$A = Q^{-1}\cdot L^{-1}\cdot\begin{pmatrix}U\\0\end{pmatrix}\cdot P^{-1}$$

of $A$. If $Q=I_m$ (e.g. if $\operatorname{rank}(A)=m$), then $A=L^{-1}\cdot\begin{pmatrix}U\\0\end{pmatrix}\cdot P^{-1}$ is called a *LUP-decomposition*. If in addition we have $P=I_n$, then $A=L^{-1}\begin{pmatrix}U\\0\end{pmatrix}$ constitutes a *LU-decomposition*.

We can now show that a decomposition similar to that in the algorithm (but slightly more general) indeed allows us to determine many important properties of the original matrix.

**Theorem 4.7.** For $A\in\mathbb{K}^{m\times n}$ let $L,Q\in\mathrm{GL}_m(\mathbb{K})$, $P\in\mathrm{GL}_n(\mathbb{K})$, $E\in\mathrm{GL}_r(\mathbb{K})$ and $U\in\mathbb{K}^{r\times(n-r)}$ with

$$L\cdot Q\cdot A\cdot P = \begin{pmatrix}E & U\\ 0 & 0\end{pmatrix}$$

Then the following statements are true:

(a) We have $\operatorname{rank}(A)=r$.

(b) The columns of $P\cdot\begin{pmatrix}E^{-1}U\\-I_{n-r}\end{pmatrix}$ are a basis of $\ker(A)$.

(c) Writing

$$L=\begin{pmatrix}L_1\\L_2\end{pmatrix},$$

a linear system $Ax=b\in\mathbb{K}^m$ is solvable if and only if $L_2\cdot Q\cdot b=0$.
In that case, $x=P\cdot\begin{pmatrix}E^{-1}L_1\\0\end{pmatrix}\cdot Q\cdot b$ is a solution.

(d) If $A \in \mathrm{GL}_n(\mathbb{K})$ and $L, Q \in \mathrm{SL}_n(\mathbb{K})$, then $\det(A) = \det(E) \cdot \det(P)^{-1}$ and $A^{-1} = P \cdot E^{-1} \cdot L \cdot Q$.

*Proof.*

(a) Because $L, Q$ and $P$ are invertible, $\operatorname{rank}(A) = \operatorname{rank}(U) = r$.

(b) Since
$$L \cdot Q \cdot A \cdot P \cdot \begin{pmatrix} E^{-1}U \\ -I_{n-r} \end{pmatrix} = \begin{pmatrix} E & U \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} E^{-1}U \\ -I_{n-r} \end{pmatrix} = 0$$
and $L$ and $Q$ are invertible, all the columns are in $\ker(A)$. Furthermore, they form a linearly independent subset with $n - r$ elements and because $\dim(\ker(A)) = n - r$ by (a) and the dimension theorem, they even constitute a basis of $\ker(A)$.

(c) We first assume that the linear system is solvable; i.e. there exists $x \in \mathbb{K}^n$, such that $Ax = b$. Then
$$\begin{pmatrix} E & U \\ 0 & 0 \end{pmatrix} \cdot P^{-1} \cdot x = L \cdot Q \cdot A \cdot x = L \cdot Q \cdot b = \begin{pmatrix} L_1 Q b \\ L_2 Q b \end{pmatrix}$$
and because the last $m - r$ entries of the left column vector are necessarily zero, the same must be true for the right one; that is, $L_2 Q b = 0$.
On the other hand, if $L_2 \cdot Q \cdot b = 0$, then $x$ as defined is a solution:
$$\begin{aligned} Ax &= A \cdot P \cdot \begin{pmatrix} E^{-1}L_1 \\ 0 \end{pmatrix} \cdot Q \cdot b = Q^{-1} L^{-1} \begin{pmatrix} E & U \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} E^{-1}L_1 \\ 0 \end{pmatrix} \cdot Q \cdot b \\ &= Q^{-1} \cdot L^{-1} \cdot \begin{pmatrix} L_1 \\ 0 \end{pmatrix} \cdot Q \cdot b = Q^{-1} \cdot L^{-1} \cdot \begin{pmatrix} L_1 \\ L_2 \end{pmatrix} \cdot Q \cdot b = b. \end{aligned}$$

(d) If $A$ is invertible, then $r = n$ by (a) and thus $L \cdot Q \cdot A \cdot P = E$, implying $A = Q^{-1} \cdot L^{-1} \cdot E \cdot P^{-1}$ Hence, the claim follows from the fact that the determinant is multiplicative. □

With the previous two theorems and the corresponding algorithm we have thus completed our objective to reduce most operations of linear algebra to matrix multiplication and we have demonstrated that such decompositions can be computed in $\mathcal{O}(\left(\frac{n}{m} + 1\right) M(m))$ for a matrix $A \in \mathbb{K}^{m \times n}$.

# 5 Algebraic Systems of Equations and Gröbner Bases

Lec 20
2022-01-11

## 5.1 Affine Varieties

Throughout this section, $\mathbb{K}$ denotes a field.
Given polynomials $f_1, \dots, f_m \in \mathbb{K}[x_1, \dots, x_n]$, how can we find their common roots; i.e. $\zeta = (\zeta_1, \dots, \zeta_n) \in \mathbb{K}^n$, such that $f_i(\zeta) = 0$ for all $i \in \{1, \dots, m\}$?
In algebraic geometry, the set of common roots

$$\mathcal{V}(f_1, \dots, f_m) \coloneqq \{\xi \in \mathbb{K}^n : f_i(\xi) = 0 \; \forall \, i \in \{1, \dots, m\}\}$$

is called an *affine variety*. This is just the solution set of a system of polynomial equations. If $I = (f_1, \dots, f_m)$ is the ideal generate by $m$, then $\mathcal{V}(f_1, \dots, f_m) = \mathcal{V}(I)$. In that case, $f_1, \dots, f_m$ is called an *ideal basis* of $I$. However, note that despite the name, the size of an ideal basis is not unique.
In general, it is not easy to compute an affine variety $\mathcal{V}(f_1, \dots, f_m)$ or to even determine if it is nonempty. However, Gröbner Bases will give us a systematic way to solve such polynomial equations. Therefore, they constitute an essential tool in disciplines like commutative algebra and algebraic geometry and their role can even be compared to that of Gaussian elimination in linear algebra.

**Example 5.1.**

(a) Let $f_1 = x_1x_3x_4^2 - 2x_2x_4^2 + x_1x_3 - 2x_2$, $f_2 = x_1x_3x_4 - 2x_2x_4 - 1$ and $f_3 = x_1x_4^2 + x_1 + 2 \in \mathbb{K}[x_1, \dots, x_4]$. Then $(-x_1x_4) \cdot f_1 + (x_1x_4^2 + x_1) \cdot f_2 + f_3 = 2$, so $I \coloneqq (f_1, f_2, f_3)$ satisfies $\mathcal{V}(I) = \emptyset$. Since $2 \in I$, $\{1\}$ is an alternative ideal basis for $I$.

(b) Let $f_1 = x^3 + x^2y + xy + y^2$, $f_2 = x^2y^2 + x^2 + y^3 + y$, $f_3 = x^3 + xy \in \mathbb{K}[x, y]$ and consider $I = (f_1, f_2, f_3)$. Because $x^2 + y$ divides all three polynomials, $|\mathcal{V}(f_1, f_2, f_3)| = \infty$ for $\mathbb{K} = \mathbb{R}$.

## 5.2 The Univariable Polynomial Ring

We first consider the easy case where $n = 1$; that is, we want to compute the affine variety $\mathcal{V}(f_1, \dots, f_m)$ of polynomials $f_1, \dots, f_m \in \mathbb{K}[x]$.
Because $\mathbb{K}[x]$ is a Euclidean ring, we can use the (extended) Euclidean algorithm (see Algorithm 1.27), in order to compute $h_1, h_2 \in K[x]$, such that $\gcd(f_1, f_2) = h_1f_1 + h_2f_2$. By iterating this, we obtain $h_1, \dots, h_m \in K[x]$ with $g \coloneqq \gcd(f_1, \dots, f_m) = \sum_{i=1}^m h_if_i$. Therefore, if $f_i(\xi) = 0$ for all $i \in \{1, \dots, m\}$, then $g(\xi) = 0$ and the reverse is also true, as $g$ divides all $f_i$.
It follows that $\mathcal{V}(f_1, \dots, f_m) = \mathcal{V}(g)$, which then can be computed numerically.

## 5.3 Resultant method

Before we actually turn to Gröbner bases, we want to sketch another method for the same task, called the *resultant method*. For $f, g \in \mathbb{K}[x] \setminus \{0\}$, we have

$$\gcd(f, g) \neq 1 \iff \operatorname{res}(f, g) = 0.$$

We now assume that $\mathbb{K}$ is algebraically closed. Then

$$\gcd(f, g) \neq 1 \iff f, g \text{ have at least one common root.}$$

This observation can be used to recursively obtain a common roots of $f_1, f_2 \in \mathbb{K}[x_1, \dots, x_n]$. Indeed, if $\zeta = (\zeta_1, \dots, \zeta_{n-1}) \in \mathbb{K}^{n-1}$, then there exists $\zeta_n \in \mathbb{K}$, such that $f_1(\zeta_1, \dots, \zeta_n) = f_2(\zeta_1, \dots, \zeta_n) = 0$ if and only if

$$\operatorname{res}_{x_n}(f_1(\zeta_1, \dots, \zeta_{n-1}, x_n), f_2(\zeta_1, \dots, \zeta_{n-1}, x_n)) = 0.$$

If we furthermore assume that $\deg_{x_n}(f_i) = \deg_{x_n}(f_i(\zeta))$ for $i \in \{1, 2\}$, then first plugging $(\xi_1, \dots, \xi_{n-1})$ into the $f_i$ and then computing the resultant w.r.t. $x_n$ is the same as first computing the resultant of $f_1, f_2 \in (\mathbb{K}[x_1, \dots, x_{n-1}])[x_n]$ and then plugging in $(\xi_1, \dots, \xi_{n-1})$:

$$\operatorname{res}_{x_n}(f_1(\zeta_1, \dots, \zeta_{n-1}, x_n), f_2(\zeta_1, \dots, \zeta_{n-1}, x_n)) = \operatorname{res}_{x_n}(f_1, f_2)(\xi_1, \dots, \xi_{n-1}).$$

Thus it suffices to determine solutions of $\operatorname{res}_{x_n}(f_1, f_2) \in \mathbb{K}[x_1, \dots, x_{n-1}]$.
Iterating this, it suffices to find the roots of two polynomials in the univariable polynomial ring.
This procedure cannot be easily generalized to $m$ polynomials, so in that case, it needs to be applied $m-1$ times. This is not only computationally expensive but also theoretically unsatisfying. We will see that Gröbner bases provide a much more sophisticated framework for computing common roots of polynomials.

## 5.4 Hilbert's Nullstellensatz

As another preliminary step, we recall *Hilbert's Nullstellensatz* and the corresponding terminology.
For a commutative ring $R$ and an ideal $I \subset R$, the *radical ideal* of $I$ is the ideal

$$\sqrt{I} := \left\{ r \in R : \exists\, k \in \mathbb{N} : a^k \in I \right\} \subset R$$

and $I$ is called *radical* if $\sqrt{I} = I$.
For a subset $S \subset \mathbb{K}^n$, the *vanishing ideal* of $S$ is the ideal

$$\mathcal{I}(S) := \{ f \in K[x_1, \dots, x_n] : f(s) = 0 \; \forall\, s \in S \} \subset K[x_1, \dots, x_n].$$

**Theorem 5.2** (**Hilbert's Nullstellensatz**)**.** Let $\mathbb{K}$ be an algebraically closed field and $I \subset \mathbb{K}[x_1, \dots, x_n]$ an ideal. Then

$$\mathcal{I}(\mathcal{V}(I)) = \sqrt{I}$$

and in particular

$$\mathcal{V}(I) = \emptyset \iff 1 \in I.$$

The theorem gives us rise to an inclusion-reversing bijective correspondence

$$\begin{aligned} \{S \subset \mathbb{K}^n \text{ affine variety}\} &\longleftrightarrow \{I \subset K[x_1, \dots, x_n] \text{ radical ideal}\} \\ S &\longmapsto \mathcal{I}(S) \\ \mathcal{V}(I) &\longleftarrow\!\shortmid\; I. \end{aligned}$$

## 5.5 Monomial Orderings

Lec 21
2022-01-13

One property that makes the polynomial ring in a single variable $K[x]$ easier to work with than a polynomial ring in multiple variables is that there is an obivous total order on the monomials: We may say that a monomial $x^n$ is smaller than $x^m$ if and only if $n < m$.
It is therefore a natural idea to try to define a "suitable" order on the multivariant polynomial ring $K[x_1, \dots, x_n]$. However, to keep the theory as general as possible, one considers a class of "suitable" orders instead of just a single one. They are called *monomial orderings*.

**Definition 5.3.**

(a) A **monomial** is a polynomial of the form $x_1^{e_1} \cdots x_n^{e_n}$. We denote the set of monomials by $M$ and write $\mathrm{M}(f)$ for the set of monomials occuring in $f \in \mathbb{K}[x_1, \dots, x_n]$. A **term** is a polynomial of the form $c \cdot t$ for $t \in M$, $c \in \mathbb{K} \setminus \{0\}$.

(b) A **monomial ordering** $\leq$ is a relation on $M$, such that

   (i) $\leq$ is a total order; i.e. it is a partial order and all monomials are comparable.

   (ii) $1 \in M$ is the smallest element w.r.t. $\leq$; i.e. $1 \leq t$ for all $t \in M$.

   (iii) $\leq$ is "compatible" with multiplication; i.e. for all $t_1, t_2, s \in M$ with $t_1 \leq t_2$, we have $s \cdot t_1 \leq s \cdot t_2$.

(c) For a given monomial order $\leq$ and a polynomial $f \in \mathbb{K}[x_1, \dots, x_n] \setminus \{0\}$, the **leading monomial** $\mathrm{LM}(f) := \max\{\mathrm{M}(f)\}$ of $f$ is the largest monomial occuring in $f$. The corresponding coefficient is the **leading coefficient** $\mathrm{LC}(f)$ and the corresponding term is the **leading term** $\mathrm{LT}(f)$. Furthermore, we set $\mathrm{LM}(0) = \mathrm{LC}(0) = \mathrm{LT}(0) = 0$.

In particular, if $s \mid t$, then $1 \leq \frac{t}{s}$ implies $s \leq t$, so any monomial ordering refines the partial order given by division.
Because of the monoid isomorphism

$$(\mathbb{N}^n, +) \to (M, \cdot),\ (k_1, \dots, k_n) \mapsto x_1^{k_1} \dots x_n^{k_n},$$

we may identify $M$ with $\mathbb{N}^n$. With this identification, the partial order on $M$ defined by division corresponds to the product order $\leq$ on $\mathbb{N}^n$ given by

$$(k_1, \dots, k_n) \leq (l_1, \dots, l_n) \ :\Longleftrightarrow\ k_1 \leq l_1 \wedge \dots \wedge k_n \leq l_n.$$

We now give some examples of important monomial orderings in the polynomial ring $\mathbb{K}[x_1, \dots, x_n]$.

**Example 5.4.** Let $t = x_1^{e_1} \cdot x_n^{e_n}$ and $t' = x_1^{e_1'} \cdot x_n^{e_n'}$ be two monomials.

(a) The *lexicographic ordering* $\leq_{\text{lex}}$ is defined to be

$$t \leq_{\text{lex}} t' \ :\Longleftrightarrow\ t = t' \ \vee\ e_i < e_i' \text{ for the smallest } i \in \{1, \dots, n\} \text{ with } e_i \neq e_i'.$$

This monomial ordering is inspired by the typical order of words in a dictionary, where words are compared by their first letter first and if they are equal by the second one and so on.

(b) The *graded lexicographic ordering* $\leq_{\text{glex}}$ is similar, but also takes the degree of the monomials into account:

$$t \leq_{\text{glex}} t' \;:\Longleftrightarrow\; t = t' \;\vee\; \deg(t) < \deg(t') \;\vee\; (\deg(t) = \deg(t') \wedge t <_{\text{lex}} t').$$

(c) The *graded reverse lexicographic ordering* $\leq_{\text{grevlex}}$ is a variation of the previous order:

$$t \leq_{\text{grevlex}} t' \;:\Longleftrightarrow\; t = t' \;\vee\; \deg(t) < \deg(t') \;\vee \\ (\deg(t) = \deg(t') \wedge e_i > e_i' \text{ for the largest } i \in \{1, \dots, n\} \text{ with } e_i \neq e_i').$$

For example, in $\mathbb{K}[x_1, x_2, x_3]$, we have

$$x_2^2 <_{\text{lex}} x_1x_3, \qquad x_2^2 <_{\text{glex}} x_1x_3 \quad \text{and} \quad x_2^2 >_{\text{grevlex}} x_1x_3.$$

For the remainder of this section, we fix an arbitrary monomial order $\leq$.

**Proposition 5.5.** For two polynomials $f, g \in \mathbb{K}[x_1, \dots, x_n]$, we have:

(a) $\mathrm{LT}(f \cdot g) = \mathrm{LT}(f) \cdot \mathrm{LT}(g)$,

(b) $\mathrm{LM}(f + g) \leq \max\{\mathrm{LM}(f), \mathrm{LM}(g)\}$.

*Proof.*

(a) Write $\mathrm{LT}(f) = c \cdot s$ and $\mathrm{LT}(g) = d \cdot t$ for $c, d \in \mathbb{K}$ and $s, t \in M$. For $s' \in \mathrm{M}(f)$ and $t' \in \mathrm{M}(g)$, we have $s' \leq s$ and $t' \leq t$, so $s' \cdot t' \leq s \cdot t' \leq s \cdot t$ with equality if and only if $s = s'$ and $t = t'$. Therefore, $\mathrm{LT}(f \cdot g) = c \cdot d \cdot s \cdot t = \mathrm{LT}(f) \cdot \mathrm{LT}(g)$.

(b) Because every monomial occuring in $f + g$ must occur in $f$ or in $g$, the statement is clear. □

The first statement of the previous proposition essentially states that when taking the product of two polynomials, their leading terms cannot cancel.

**Lemma 5.6 (Dickson's Lemma).** Let $S \subset M$ be a set of monomials. Then there is a finite subset $B \subset S$, such that for all $s \in S$, there exists $t \in B$ with $t \mid s$.
Such a subset $B$ is called a *basis* of $S$.

*Proof.* We identify $M$ with $\mathbb{N}^n$ and proceed by induction on $n$. The base case $n = 1$ is easy: If $S = \emptyset$ we may use $B = \emptyset$ and otherwise we can use the fact that the natural numbers are well ordered and set $B = \{\min(S)\}$.
We now perform the induction step. For $k \in \mathbb{N}$, consider

$$S_k \coloneqq \big\{(e_2, \dots, e_n) \in \mathbb{N}^{n-1} : (k, e_2, \dots, e_n) \in S\big\} \subset \mathbb{N}^{n-1}.$$

By the inductive hypothesis, there exist finite bases $B_k \subset S_k$ and furthermore the set $\bigcup_{k \in \mathbb{N}} B_k$ has some finite basis $C$. Since $|C| < \infty$, there exists $r \in \mathbb{N}$ with $C \subset \bigcup_{k=0}^{r} B_k$. We claim that

$$B \coloneqq \{(e_1, \dots, e_n) \in \mathbb{N}^n : e_1 \in \{0, \dots, r\}, (e_2, \dots, e_n) \in B_{e_1}\} \subset S$$

is a basis of $S$. By definition, $B$ is finite. It is left to prove that for any $e = (e_1, \dots, e_n) \in S$, there exists some element in $B$ which is smaller or equal to it.

Because $(e_2, \dots, e_n) \in S_{e_1}$, there is $(d_2, \dots, d_n) \in B_{e_1}$ such that $(d_2, \dots, d_n) \le (e_2, \dots, e_n)$. If $e_1 \le r$, then $(e_1, d_2, \dots, d_n) \in B$, so $(e_1, d_2, \dots, d_n) \le e$. Otherwise, we use that $(d_2, \dots, d_n) \in \bigcup_{k \in \mathbb{N}} B_k$ to conclude that there must exist $c = (c_2, \dots, c_n) \in C$ with $c \le d$. In particular, there is $k \in \{0, \dots, r\}$ such that $c \in B_k$, showing $(k, c_2, \dots, c_n) \in B$ with $(k, c_2, \dots, c_n) \le (r, d_2, \dots, d_n) \le e$. □

In essence, Dickson's lemma is really a statement about the product order on $\mathbb{N}^n$. For our purposes of monomial orderings, it yields a useful corollary.

**Corollary 5.7.** Every monomial ordering is a well ordering (i.e. a total order such that every non-empty subset has a least element).

*Proof.* For a non-empty subset $S \subset M$ there exists a finite basis $B \subset S$ by Lemma 5.6 and because any monomial ordering is total, this set must have a least element, which is then a least element of all of $M$. □

## 5.6 Gröbner Bases

As before, we fix an arbitrary monomial ordering $\le$. In particular, any uniqueness claims in this section only apply once the monomial ordering is chosen.

**Definition 5.8.**

(a) For a set $S \subset \mathbb{K}[x_1, \dots, x_n]$, the **leading ideal** of $S$ is defined to be

$$\begin{aligned} \mathrm{L}(S) &\coloneqq (\mathrm{LM}(f) : f \in S) \\ &= \left\{ \sum_{i=1}^{r} g_i \,\mathrm{LM}(f_i) : g_i \in \mathbb{K}[x_1, \dots, x_n], f_i \in S, r \in \mathbb{N} \right\} \\ &= \{ g \in \mathbb{K}[x_1, \dots, x_n] : \forall\, t \in \mathrm{M}(g)\ \exists\, f \in S : \mathrm{LM}(f) \mid t \}. \end{aligned}$$

(b) For an ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$, a finite subset $G \subset I$ is called a **Gröbner basis** of $I$ (w.r.t. the chosen monomial ordering) if $\mathrm{L}(G) = \mathrm{L}(I)$.

We first observe that that the name “basis” is justified as any Gröbner basis of an ideal indeed generates it.

**Proposition 5.9.** Let $G$ ba a Gröbner basis of the ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$. Then the ideal $(G)$ generated by $G$ is equal to $I$.

*Proof.* It is clear that $(G) \subset I$. For the other inclusion, assume for contradiction that there exists $f \in I \setminus (G)$. By Corollary 5.7, we may additionally suppose that $f$ is minimal with this property.
Since $\mathrm{LM}(f) \in \mathrm{L}(G)$, there exists $g \in G$ with $\mathrm{LM}(g) \mid \mathrm{LM}(f)$. Thus the polynomial $\tilde{f} \coloneqq f - \frac{\mathrm{LT}(f)}{\mathrm{LT}(g)} \cdot g$ satisfies $\mathrm{LM}\left(\tilde{f}\right) < \mathrm{LM}(f)$ and $\tilde{f} \in (G)$ if and only if $f \in (G)$. Therefore, the existence of $\tilde{f}$ contradicts the minimality of $f$. □

**Example 5.10.** The ideal $I = \mathbb{K}[x]$ has many bases, e.g. $\{1\}$ or $\{x, x+1\}$. The first set is a Gröbner basis but the second is not, because $\mathrm{L}(\{x, x+1\}) = (x)$ but $\mathrm{L}(I) = \mathrm{L}(M) = \mathbb{K}[x]$.

**Definition 5.11.** An ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$ is called **monomial**, if it can be generated by monomials.

In particular, any monomial ideal generated by finitely many monomials $m_1, \dots, m_r$ has the Gröbner basis $\{m_1, \dots, m_r\}$. It is also clear that any principal ideal is a Gröbner basis.

Lec 22
2022-01-18

**Theorem 5.12.** Every ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$ has a Gröbner basis.
In particular, any ideal in $\mathbb{K}[x_1, \dots, x_n]$ is finitely generated, so $\mathbb{K}[x_1, \dots, x_n]$ is Noetherian.

*Proof.* Let $S \coloneqq \{\mathrm{LM}(f) \mid f \in I \setminus \{0\}\} \subset M$. By Lemma 5.6, there exist $f_1, \dots, f_r \in I$, such that $\{\mathrm{LM}(f_1), \dots, \mathrm{LM}(f_r)\}$ is a basis of $S$, so $G \coloneqq \{f_1, \dots, f_r\} \subset I$ is a Gröbner basis of $I$. □

This result is generalized by *Hilbert's basis theorem*, which states that every polynomial ring over a Noetherian ring is Noetherian.

Since we now know that any ideal has a Gröbner basis, the next natural question is how to compute them. To that end, we introduce the notion of *normal forms.*

**Definition 5.13.** Let $S = \{g_1, \dots, g_r\} \subset \mathbb{K}[x_1, \dots, x_n]$ be a finite set of polynomials and $f \in \mathbb{K}[x_1, \dots, x_n]$.

(a) $f$ is **in normal form** w.r.t. $S$ if none of the monomials $t \in \mathrm{M}(f)$ occuring in $f$ are divisible by any of the $\mathrm{LM}(g_i)$ for $i \in \{1, \dots, r\}$.

(b) A polynomial $f^* \in \mathbb{K}[x_1, \dots, x_n]$ is called a **normal form** of $f$ if

  (i) $f^*$ is in normal form;

  (ii) there exist $h_i \in \mathbb{K}[x_1, \dots, x_n]$, $i \in \{1, \dots, r\}$ with $\mathrm{LM}(h_i g_i) \le \mathrm{LM}(f)$ such that
$$f^* - f = \sum_{i=1}^{n} h_i g_i.$$

In particular, any normal form $f^*$ of $f$ w.r.t. $S$ is congruent to $f$ modulo $(S)$.

**Example 5.14.** Consider the set $S \coloneqq \{x, x+1\} \subset \mathbb{K}[x]$.

(a) $f = 1$ is in normal form w.r.t. $S$, even though $f \equiv 0 \mod (S)$.

(b) $f = x$ has normal forms 0 and $-1$. In particular, this shows that normal forms need not be unique.

We can give a straightforward algorithm to compute a normal form of a given polynomial.

**Algorithm 5.15.**

Input: $S \coloneqq \{g_1, \dots, g_r\} \subset \mathbb{K}[x_1, \dots, x_n]$, $f \in \mathbb{K}[x_1, \dots, x_n]$.

Output: A normal form $f^*$ of $f$ and $h_1, \dots, h_r$ as in Definition 5.13.

```
(1) Set $f^* = f$, $h_i = 0$ for $i \in \{1, \dots, r\}$.
```

(2) Repeat:

(3) Form $M \coloneqq \{(t,i) : t \in \mathrm{M}(f^*), i \in \{1,\dots,r\}, \mathrm{LM}(g_i) \mid t\}$.

(4) If $M = \emptyset$: return $f^*, h_1, \dots, h_r$.

(5) Choose $(t,i) \in M$ such that $t$ is maximal. If there are multiple second entries with that same first entry, use any method to pick the second entry (e.g. by always choosing the second entry minimal).

(6) Let $c \in \mathbb{K}$ denote the coefficient of $t$ in $f^*$ and set

$$f^* \coloneqq f^* - \frac{ct}{\mathrm{LT}(g_i)} g_i, \qquad h_i \coloneqq h_i - \frac{ct}{\mathrm{LT}(g_i)}.$$

*Proof (of correctness).* At the beginning of the algorithm, we have $f^* - f = \sum_{i=1}^n h_i g_i$ and $\mathrm{LM}(h_i g_i) \le \mathrm{LM}(f)$ and this holds true throughout the algorithm.
The algorithm terminates when $M$ is empty and in that case $f^*$ is indeed a normal form of $f$. Because $t = \mathrm{LT}\left(\frac{ct}{\mathrm{LT}(g_i)} g_i\right)$, the leading term of the new $f^*$ in step (6) is strictly smaller than that of the old one, so the sequence of $t$'s is strictly decreasing. Since any monomial ordering is a well ordering (Corollary 5.7), this implies that the algorithm terminates after finitely many steps. □

Our next result states that normal forms w.r.t. Gröbner bases are unique and that two Gröbner bases of the same ideal give rise to the same normal forms.

**Theorem 5.16.** Let $G$ be a Gröbner basis of an ideal $I \subset \mathbb{K}[x_1,\dots,x_n]$.

(a) Every polynomial $f \in \mathbb{K}[x_1,\dots,x_n]$ has exactly one normal form w.r.t. $G$. Thus we obtain a map

$$\mathrm{NF}_G \colon \mathbb{K}[x_1,\dots,x_n] \to \mathbb{K}[x_1,\dots,x_n],\ f \mapsto f^*.$$

(b) $\mathrm{NF}_G$ is $\mathbb{K}$-linear and $\ker(\mathrm{NF}_G) = I$.

(c) If $\widetilde{G}$ is another Gröbner basis of $I$ (w.r.t. the same monomial ordering), then $\mathrm{NF}_G = \mathrm{NF}_{\widetilde{G}}$.

*Proof.* We prove (a) and (c) together. Let $f \in \mathbb{K}[x_1,\dots,x_n]$ and suppose that $f^*$ is a normal form of $f$ w.r.t. G and that $\tilde{f}$ is a normal form of $f$ w.r.t. to $\widetilde{G}$. Since $f^* - \tilde{f} \in I$, there exist $g \in G$ and $\tilde{g} \in \widetilde{G}$ such that

$$\mathrm{LM}(g) \mid \mathrm{LM}\left(f^* - \tilde{f}\right) \quad \text{and} \quad \mathrm{LM}(\tilde{g}) \mid \mathrm{LM}\left(f^* - \tilde{f}\right).$$

If $f^* - \tilde{f} \neq 0$, then $\mathrm{LM}\left(f^* - \tilde{f}\right) \in \mathrm{M}\left(f^* - \tilde{f}\right)$ and $\mathrm{LM}\left(f^* - \tilde{f}\right) \in \mathrm{M}(f^*) \cup \mathrm{M}\left(\tilde{f}\right)$, which contradicts the fact that $f^*$ and $\tilde{f}$ are in normal form. This shows (a) and (c) and it remains to prove (b).
To that end, let $f, g \in \mathbb{K}[x_1,\dots,x_n]$ and $c \in \mathbb{K}$. We observe that

$$h \coloneqq \mathrm{NF}_G(f + cg) - \mathrm{NF}_G(f) - c\,\mathrm{NF}_G(g)$$

satisfies $h \equiv f + cg - f - cg = 0 \mod I$, so $h \in I$, which means that there exists $r \in G$ with $\mathrm{LM}(r) \mid \mathrm{LM}(h)$. But $h$ is in normal form w.r.t. to $G$, so $h = 0$ and $\mathrm{NF}_G$ is $\mathbb{K}$-linear.
Finally, any $f \in I$ satisfies $\mathrm{NF}_G(f) = 0$ and conversely if $f \in \ker(\mathrm{NF}_G)$, then $f \equiv 0 \mod I$, so $f \in I$.

□

We can now discuss some first applications of Gröbner bases.

**Application 5.17.** Let $I \subset \mathbb{K}[x_1, \ldots, x_n]$ be an ideal and $G \subset I$ a Gröbner basis of $I$.

(1) We have $1 \in I$ if and only if $G$ contains a non-zero constant, so we can decide whether an ideal is proper or not.
If $\mathbb{K}$ is algebraically closed, then by Hilbert's Nullstellensatz (Theorem 5.2) this is equivalent to $\mathcal{V}(I) = \emptyset$.

(2) More generally, we can determine if an arbitrary polynomial $f \in \mathbb{K}[x_1, \ldots, x_n]$ is contained in $I$ by computing its normal form $\mathrm{NF}_G(f)$ and checking if $\mathrm{NF}_G(f) = 0$.

(3) We can "honestly" compute in the factor ring $\mathbb{K}[x_1, \ldots, x_n]/I$, because $\mathrm{NF}_G$ induces an embedding (injective $\mathbb{K}$-linear map) $\mathbb{K}[x_1, \ldots, x_n]/I \hookrightarrow \mathbb{K}[x_1, \ldots, x_n]$.

## 5.7 Buchberger's Algorithm

**Definition 5.18.** Let $f, g \in \mathbb{K}[x_1, \ldots, x_n] \setminus \{0\}$ and set $t := \gcd(\mathrm{LM}(f), \mathrm{LM}(g))$. Then the *S-polynomial* of $f$ and $g$ is defined to be

$$S(f, g) := \frac{\mathrm{LT}(g)}{t} f - \frac{\mathrm{LT}(f)}{t} g.$$

The main motivation behind this construction is that the leading terms of the two summands in the definition are precisely $\mathrm{LC}(f) \cdot \mathrm{LC}(g) \cdot \mathrm{lcm}(\mathrm{LM}(f), \mathrm{LM}(g))$ (because $\mathrm{lcm}(h, h') \cdot \gcd(h, h') = h \cdot h'$ for all $h, h'$ in a factorial ring), so they cancel. Lec 23 2022-01-20

**Example 5.19.** Consider the polynomials $f = x^2 + y^2$, $g = xy \in \mathbb{K}[x, y]$ equipped with the lexicographic ordering $x > y$. We calculate

$$S(f, g) = \frac{xy}{x} \cdot f - \frac{x^2}{x} \cdot g = y^3.$$

*Buchberger's criterion* gives an equivalent characterization of a finite set being a Gröbner basis. It gives rise to *Buchberger's algorithm*, which brings Gröbner bases into the realm of computability.

**Theorem 5.20 (Buchberger's Criterion).**
For a finite set $G \subset \mathbb{K}[x_1, \ldots, x_n] \setminus \{0\}$, the following statements are equivalent:

(a) $G$ is a Gröbner basis of the ideal $(G)$ generated by it.

(b) For all $f, g \in G$, $S(f, g)$ has normal form 0 w.r.t. $G$.

*Proof.* If $G$ is a Gröbner basis of $(G)$, then $S(f,g) \in (G)$ for any $f, g \in G$, so $\mathrm{NF}_G(S(f,g)) = 0$ by Theorem 5.16.
For the other direction, write $G = \{g_1, \dots, g_r\} \subset \mathbb{K}[x_1, \dots, x_n] \setminus \{0\}$ and assume for contradiction that $G$ is not a Gröbner basis of $(G)$. This means that there exists $f \in (G)$ with $\mathrm{LM}(f) \notin L(G)$. Writing $f = \sum_{i=1}^r g_i h_i$ with $h_i \in \mathbb{K}[x_1, \dots, x_n] \setminus \{0\}$, we consider the set

$$\left\{ \max\{\mathrm{LM}(g_i h_i) : i \in \{1, \dots, r\}\} : h_i \in \mathbb{K}[x_1, \dots, x_n] \setminus \{0\}, \sum_{i=1}^r g_i h_i = f \right\}.$$

Because any monomial ordering is a well ordering (Corollary 5.7), we may assume that our $h_i$ correspond to the smallest element $t := \max\{\mathrm{LM}(g_i h_i) : i \in \{1, \dots, r\}\}$ of this set. Furthermore, we have $\mathrm{LM}(f) \in \mathrm{M}(g_i h_i)$ for some $i \in \{1, \dots, r\}$ and because $\mathrm{LM}(f) \notin L(G)$, this implies $\mathrm{LM}(f) < \mathrm{LM}(g_i h_i)$, so in particular $\mathrm{LM}(f) < t$, showing that the coefficient of $t$ in $\sum_{i=1}^r g_i h_i$ is zero. Setting

$$c_i := \begin{cases} \mathrm{LC}(h_i) & \mathrm{LM}(g_i h_i) = t \\ 0 & \text{otherwise} \end{cases} \in \mathbb{K},$$

this means that

$$\sum_{i=1}^r c_i \,\mathrm{LC}(g_i) = 0. \tag{$*$}$$

By reordering the $g_i$ we may assume that $c_1 \neq 0$. Now let $l \in \{2, \dots, r\}$ and with $t_l = \mathrm{lcm}(\mathrm{LM}(g_l), \mathrm{LM}(g_1))$, we have

$$S(g_l, g_1) = \frac{\mathrm{LC}(g_1) t_l}{\mathrm{LM}(g_l)} g_l - \frac{\mathrm{LC}(g_l) t_l}{\mathrm{LM}(g_1)} g_1.$$

Because the leading coefficients cancel, $\mathrm{LM}(S(g_l, g_1)) < t_l$. By assumption, this S-polynomial has normal form 0; i.e. there exist $h_{l,j} \in \mathbb{K}[x_1, \dots, x_n]$, such that

$$S(g_l, g_1) = \sum_{j=1}^r h_{l,j} g_j \quad \text{and} \quad \mathrm{LM}(h_{l,j} g_j) \leq \mathrm{LM}(S(g_l, g_1)) < t_l \ \ \forall\ j \in \{1, \dots, r\}.$$

Suppose $c_l \neq 0$. Then we have $\mathrm{LM}(h_l)\,\mathrm{LM}(g_l) = t = \mathrm{LM}(h_1)\,\mathrm{LM}(g_1)$ and thus $t_l \mid t$. Therefore,

$$s_l := \frac{t}{t_l} S(g_l, g_1) = \sum_{j=1}^r \frac{t}{t_l} h_{l,j} g_j \quad \text{satisfies} \quad \mathrm{LM}\left(\frac{t}{t_l} h_{l,j} g_j\right) < t \tag{$**$}$$

and can also be written as

$$s_l = \mathrm{LC}(g_1)\,\mathrm{LM}(h_l) g_l - \mathrm{LC}(g_l)\,\mathrm{LM}(h_1) g_1.$$

We obtain

$$\begin{aligned} \sum_{l=2}^r c_l s_l &\overset{(*)}{=} \sum_{l=2}^r c_l (\mathrm{LC}(g_1)\,\mathrm{LM}(h_l) g_l - \mathrm{LC}(g_l)\,\mathrm{LM}(h_1) g_1) \\ &\qquad + \left(c_1\,\mathrm{LC}(g_1) + \sum_{l=2}^r c_l\,\mathrm{LC}(g_l)\right) \mathrm{LM}(h_1) g_1 \\ &= \sum_{l=1}^r c_l\,\mathrm{LC}(g_1)\,\mathrm{LM}(h_l) g_l = \mathrm{LC}(g_1) \cdot \sum_{l=1}^r c_l\,\mathrm{LM}(h_l) g_l. \end{aligned}$$

Because $\mathrm{LC}(g_1) \neq 0$, this equation together with $(**)$ yields the existence of $\tilde{h}_j \in \mathbb{K}[x_1, \dots, x_n] \setminus \{0\}$, such that

$$g \coloneqq \sum_{l=1}^{r} c_l \, \mathrm{LM}(h_l) g_l = \sum_{j=1}^{r} \tilde{h}_j g_j \quad \text{and} \quad \mathrm{LM}\Big(\tilde{h}_j g_j\Big) < t \;\; \forall \; j \in \{1, \dots, r\}.$$

Finally, we have

$$f = f - g + g = \sum_{j=1}^{r} \Big(h_j - c_j \, \mathrm{LM}(h_j) + \tilde{h}_j\Big) g_j$$

and for all $j \in \{1, \dots, r\}$:

$$\mathrm{LM}\Big(\Big(h_j - c_j \, \mathrm{LM}(h_j) + \tilde{h}_j\Big) g_j\Big) = \begin{cases} \mathrm{LM}\Big((h_j - \mathrm{LT}(h_j) + \tilde{h}_j) g_j\Big) & \text{if } c_j \neq 0 \\ \mathrm{LM}\Big((h_j + \tilde{h}_j) g_j\Big) & \text{if } c_j = 0 \end{cases}.$$

However, this leading monomial is less than $t$: In the first case, this is due to the leading monomials of $h_j g_j$ canceling; in the second case, all $h_j g_j$ are less than $t$ anyway. This contradicts the minimality of $t$ and thus completes the proof. □

**Algorithm 5.21 (Buchberger's Algorithm).**

```
  Input: $S \subset \mathbb{K}[x_1, \dots, x_n]$ finite.
Output: A Gröbner basis $G$ of the ideal $(S)$ generated by $S$.
    (1) Set $G \coloneqq S \setminus \{0\}$.
    (2) For $g, h \in G$:
        (3) Compute the S-polynomial $s \coloneqq S(g, h)$ and a normal form $s^*$ of $s$ w.r.t. $G$.
        (4) If $s^* \neq 0$: Set $G = G \cup \{s^*\}$.
    (5) Return $G$.
```

*Proof (of correctness).* Because all $s^*$ lie in $(S)$, it follows directly from Theorem 5.20 that the algorithm yields a Gröbner basis of $(G) = (S)$ if it terminates. In order to see that the algorithm always terminates after finitely many steps, assume for contradiction that this is not the case. Let $G_l$ denote $G$ after the $l$-th step and consider

$$S_l \coloneqq \{\mathrm{LM}(g) : g \in G_l\}, \qquad S = \bigcup_{l \in \mathbb{N}} S_l.$$

By Lemma 5.6, there exists a finite basis $B$ of $S$ and because the $S_r$ are increasing, this basis must be contained in some $S_r$. But by assumption $s^* \neq 0$ and because $s^*$ is in normal form, we have $\mathrm{LM}(s^*) \notin L(G_r)$. But this means that no $g \in G_r$ satisfies $\mathrm{LM}(g) \mid \mathrm{LM}(s^*)$ even though $\mathrm{LM}(s^*) \in S_{r+1} \subset S$, which contradicts the fact that $B$ is a basis.
With a little more theorey, the argument becomes much easier: Because $\mathrm{L}(G)$ is strictly increasing in each iteration, the algorithm must terminate as $\mathbb{K}[x_1, \dots, x_n]$ is Noetherian. □

This presentation of the algorithm is highly unoptimized and only serves to demonstrate the basic idea. Indeed, many optimizations are possible. For example, there is no need to check pairs $(g, h)$ for which $(h, g)$ was already considered or which were already checked in a previous iteration. A somewhat less obvious optimization is to ignore pairs whose leading monomials are coprime, because the corresponding S-polynomial will always have 0 as a normal form.

Lec 24
2022-01-25

**Example 5.22.** Like Example 5.19, let $f = x^2 + y^2$, $g = xy \in \mathbb{K}[x, y]$ and $S = \{f, g\}$ with the lexicographic ordering $x > y$. Because $h := S(f, g) = y^3$ is in normal form w.r.t. $S$, we set $G = \{f, g, h\}$. The calculation

$$\begin{aligned} &S(f,h) = y^3 f - x^2 h = y^5 \implies \mathrm{NF}_G(S(f,h)) = 0 \\ &S(g,h) = 0 \quad \text{(because the S-polynomial of two monomials is always 0)} \end{aligned}$$

shows that $G$ is a Gröbner basis of $(S)$.

We now introduce *reduced* Gröbner bases, which are essentially Gröbner bases that contain no redundant elements.

**Definition 5.23.** A Gröbner basis $G$ is called **reduced**, if every $g \in G$ is in normal form w.r.t. $G \setminus \{g\}$ and all elements of $G$ are monic ($\mathrm{LC}(g) = 1$ for all $g \in G$).

Computing a reduced Gröbner basis from an unreduced one is straightforward: Iteratively replace all elements $g \in G$ by a normal form of $g$ w.r.t. $G \setminus \{g\}$. Then remove all zero polynomials and normalize the remaining elements.
Moreover, unlike ordinary Gröbner bases, reduced Gröbner bases are unique.

**Theorem 5.24.** Every ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$ has a unique reduced Gröbner basis.

*Proof.* We already saw that any Gröbner basis gives rise to a reduced Gröbner basis, so we only have to prove uniqueness. To that end, let $G, G'$ be reduced Gröbner bases of $I$ and take $g \in G$. Because $G'$ is a Gröbner basis, there exists $g' \in G'$, such that $\mathrm{LM}(g') \mid \mathrm{LM}(g)$ and since $G$ is also a Gröbner basis, there is $g'' \in G$ with $\mathrm{LM}(g'') \mid \mathrm{LM}(g')$. Since $g$ is in normal form w.r.t. $G \setminus \{g\}$, we have $g = g''$ and $\mathrm{LM}(g) = \mathrm{LM}(g')$.
*Claim:* $g = g'$.
Aiming for contradiction, assume that $g - g' \in I \setminus \{0\}$. Then $t := \mathrm{LM}(g - g') \in L(G) \cap L(G')$ satisfies $t \in \mathrm{M}(g) \cup \mathrm{M}(g')$ and since both $g$ and $g'$ have leading coefficient 1, it follows $t < \mathrm{LM}(g)$. If $t \in \mathrm{M}(g)$, there exists $\widetilde{g} \in G$ with $\mathrm{LM}(\widetilde{g}) \mid t$ since $t \in L(G)$. But $\widetilde{g} \neq g$, so this would mean that $g$ is not in normal form w.r.t. $G \setminus \{g\}$, contradicting our assumption. The analogous argument applies if $t \in \mathrm{M}(g')$, so the claim follows.
The claim implies that $G \subset G'$ and the other inclusion follows by symmetry. □

We look at the special case of linear equations; i.e. a matrix $A \in \mathbb{K}^{m \times n}$. Choosing a monomial ordering with $x_1 > x_2 > \cdots > x_n$, we observe that a row echolon form obtained by Gaussian elimination is a Gröbner basis of the original system of linear equations, because the leading monomials are coprime and thus their S-polynomials have normal form 0. Furthermore, a reduced row echolon form precisely corresponds to a reduced Gröbner basis. In particular, this shows that the reduced row echolon form of a matrix over a field is unique.

Analyzing the complexity of Buchberger's algorithm (Algorithm 5.21) is extremely hard. Indeed, no upper bound for its running time is known.
One result is that for a given finite set $S \subset \mathbb{K}[x_1, \dots, x_n]$, an upper bound for the degree of the elements in a resulting Gröbner basis $G$ is

$$2 \cdot \left(\frac{d^2}{2} + d\right)^{2^{n-1}} \qquad \text{with} \qquad d = \max\{\deg(f) : f \in S\},$$

which is "doubly exponential" in $n$. Regardless, Buchberger's algorithm often works in practice in reasonable time. There also exist many variants, for example one can keep track of how the polynomials in $G$ arise as $\mathbb{K}[x_1, \dots, x_n]$-linear combination of the input polynomials.

# 6 Applications of Gröbner Bases

In this section, we highlight some applications of Gröbner bases, which should serve to underline their usefulness.

## 6.1 Elimination Ideals

**Definition 6.1.**

(a) Let $I \subset \mathbb{K}[x_1, \dots, x_n]$ and let $l \in \{1, \dots, n\}$. Then the intersection

$$I_l \coloneqq \mathbb{K}[x_1, \dots, x_l] \cap I \subset \mathbb{K}[x_1, \dots, x_l]$$

is an ideal and is called the $l$-th **elimination ideal**.

(b) A monomial ordering $\leq$ is called an $l$-**elimination ordering** if

$$x_i^k < x_j \qquad \forall\ 1 \leq i \leq l < j \leq n,\ k \in \mathbb{N}.$$

The concept of $l$-elimination orderings simply describes those monomial orderings for which all monomials in $\mathbb{K}[x_1, \dots, x_l] \subset \mathbb{K}[x_1, \dots, x_n]$ are less than $x_j$ for all $j > l$.
It is also common to define elimination ideals for arbitrary subsets $S \subset \{x_1, \dots, x_n\}$ instead of just for subsets of the form $\{x_1, \dots, x_l\}$. Because applying a permutation to the indeterminates yields a $\mathbb{K}$-algebra automorphism $\mathbb{K}[x_1, \dots, x_n] \to \mathbb{K}[x_1, \dots, x_n]$, these definitions are equivalent.
A similar construction works for $l$-elimination orderings: Any $S$-elimination ordering $\leq$ gives rise to a $|S|$-elimination ordering (as in our definition) $\leq'$ by first applying a permutation on the monomials which maps $S$ to the first $|S|$ indeterminants and then comparing the resulting monomials using $\leq$.
Therefore, we will also use $S$-elimination ideals and orderings when needed.
Note that any monomial ordering on $\mathbb{K}[x_1, \dots, x_n]$ trivially is a $n$-elimination ordering.

**Example 6.2.**

(a) The lexicographic ordering $\leq_{\text{lex}}$ with $x_1 < x_2 < \dots < x_n$ is an $l$-elimination ordering for all $l$.

(b) The graded lexicographic ordering $\leq_{\text{glex}}$ and its reversed version $\leq_{\text{grevlex}}$ are no $l$-elimination orderings, except in the trivial case $l = n$.

(c) Let $\leq$ be any monomial ordering on $\mathbb{K}[x_1, \dots, x_n]$. We construct a new monomial ordering $\leq'$ by defining for $s = x_1^{e_1} \cdot \dots \cdot x_n^{e_n}$ and $t = x_1^{d_1} \cdot \dots \cdot x_n^{d_n}$:

$$s \leq' t \;:\Longleftrightarrow\; \sum_{i=l+1}^{n} d_i < \sum_{i=l+1}^{n} e_i \;\vee\; \left( \sum_{i=l+1}^{n} d_i = \sum_{i=l+1}^{n} e_i \;\wedge\; s \leq t \right)$$

This is an $l$-elimination ordering.

The followig theorem states that Gröbner bases w.r.t. $l$-elimination ideals restrict to Gröbner bases of the elimination ideal, which is very convenient for computations.

**Theorem 6.3.** Let $G$ be a Gröbner basis of $I \subset \mathbb{K}[x_1, \dots, x_n]$ w.r.t. an $l$-elimination ordering. Then

$$G_l \coloneqq G \cap \mathbb{K}[x_1, \dots, x_l]$$

is a Gröbner basis of the elimination ideal $I_l$ w.r.t. the restricted monomial ordering.

*Proof.* It is clear that $G_l \subset I_l$. For the other inclusion, let $f \in I_l \setminus \{0\}$ and $t \coloneqq \mathrm{LM}(f)$. As $G$ is a Gröbner basis, there exists $g \in G$, such that $\mathrm{LM}(g) \mid \mathrm{LM}(f)$ and since $f \in \mathbb{K}[x_1, \dots, x_l]$, we have that $\mathrm{M}(f) \subset \mathbb{K}[x_1, \dots, x_l]$ and thus $\mathrm{LM}(g) \in \mathbb{K}[x_1, \dots, x_l]$. But because we have an $l$-elimination ordering, this means that all monomials of $g$ must lie in $\mathbb{K}[x_1, \dots, x_l]$, which implies $g \in \mathbb{K}[x_1, \dots, x_l]$ and shows the claim. □

**Example 6.4.** Like in Example 5.22 (but with $x$ and $y$ interchanged), let $f = x^2 + y^2$, $g = xy$, $h = x^3 \in \mathbb{K}[x, y]$ and set $S \coloneqq \{f, g\}$, $G \coloneqq \{f, g, h\}$. Then $G$ is a Gröbner basis of $(S)$ w.r.t. the lexicographic ordering $y > x$, so the previous theorem shows that the elimination ideal $I_x = I \cap \mathbb{K}[x]$ has Gröbner basis $G_x = \{x^3\}$.

Lec 25
2022-01-27

Our next goal is to give a geometric interpretation for the elimination ideal. To that end, we define a topology on $K^n$.

**Definition 6.5.** The **Zariski topology** on $K^n$ has the affine varieties as the closed sets; i.e.

$$X \subset \mathbb{K}^n \text{ is closed } :\Longleftrightarrow \exists\, S \subset \mathbb{K}[x_1, \dots, x_n] : \mathcal{V}(S) = X.$$

It is equivalent to demand that $S$ in the definition above is also an ideal or even a radical ideal. Because $\emptyset = \mathcal{V}(\mathbb{K}[x_1, \dots, x_n])$ and $\mathbb{K}^n = \mathcal{V}(\{0\})$, this is indeed a topology as a consequence of the following lemma.

**Lemma 6.6.**

(a) For two ideals $I, J \subset \mathbb{K}[x_1, \dots, x_n]$, we have $\mathcal{V}(I) \cup \mathcal{V}(J) = \mathcal{V}(I \cap J)$.

(b) For an arbitrary collection $(J_i)_{i \in I}$ of ideals (with index set $I$), we have $\bigcap_{i \in I} \mathcal{V}(J_i) = \mathcal{V}\big(\bigcup_{i \in I} J_i\big)$.

*Proof.* (a) It is clear that $\mathcal{V}(I) \cup \mathcal{V}(J) \subset \mathcal{V}(I \cap J)$. To see the other inclusion, let $v \in \mathcal{V}(I \cap J)$ and assume that $v \notin \mathcal{V}(I)$, so there exists $f \in I$, such that $f(v) \neq 0$. For any $g \in J$, we have $f \cdot g \in I \cap J$, so $0 = (f \cdot g)(v) = f(v) \cdot g(v)$. and it follows that $g(v) = 0$. This shows $v \in \mathcal{V}(J)$.

(b) Because $\mathcal{V}(-)$ is inclusion-reversing, this is clear. □

This defines a very coarse ("few open and closed sets") topology. For example, in the case that $\mathbb{K} = \mathbb{C}$, we observe that anything that is closed w.r.t. the Zariski topology is also closed w.r.t. the usual Euclidean topology on $\mathbb{C}^n$: Indeed, an affine variety is just the intersection of the zero sets $f^{-1}(0)$ of polynomials $f \in \mathbb{C}[x_1, \dots, x_n]$, which are continuous w.r.t. the Euclidean topology on $\mathbb{C}^n$, so any affine variety is also closed w.r.t. the Euclidean topology.
Note that all singletons and finite sets are closed w.r.t. the Zariski topology. On $\mathbb{K}^1$, the closed sets are precisely the finite sets.
For an arbitrary subset $X \subset \mathbb{K}^n$, its (topological) closure is by definition

$$\overline{X} = \bigcap_{\substack{Y \subset \mathbb{K}^n \text{ closed} \\ X \subset Y}} Y$$

Any closed $Y \subset \mathbb{K}^n$ is of the form $Y = \mathcal{V}(I)$ for some ideal $I \subset \mathbb{K}^n$. Since

$$X \subset Y \iff X \subset \mathcal{V}(I) \iff I \subset \mathcal{I}(X) \implies \mathcal{V}(\mathcal{I}(X)) \subset Y$$

and $X \subset \mathcal{V}(\mathcal{I}(X))$ is closed, it follows that

$$\overline{X} = \mathcal{V}(\mathcal{I}(X)).$$

The following theorem gives some geometric meaning to the elimination ideal.

**Theorem 6.7.** For $l \in \{1, \ldots, n\}$ consider the coordinate projection

$$\pi_l \colon \mathbb{K}^n \to \mathbb{K}^l, \ (a_1, \ldots, a_n) \mapsto (a_1, \ldots, a_l).$$

Assume that $\mathbb{K}$ is algebraically closed and let $I \subset \mathbb{K}[x_1, \ldots, x_n]$ be an ideal. Then

$$\mathcal{V}(I_l) = \overline{\pi_l(\mathcal{V}(I))}.$$

*Proof.* Let $(a_1, \ldots, a_l) \in \pi_l(\mathcal{V}(I))$, so there exist $a_{l+1}, \ldots, a_n \in \mathbb{K}$, such that $(a_1, \ldots, a_n) \in \mathcal{V}(I)$. Because any $f \in I_l = I \cap \mathbb{K}[x_1, \ldots, x_l]$ satisfies $f(a_1, \ldots, a_l) = f(a_1, \ldots, a_n) = 0$ we have $(a_1, \ldots, a_l) \subset \mathcal{V}(I_l)$ and thus $\pi_l(\mathcal{V}(I)) \subset \mathcal{V}(I_l)$. Since $\mathcal{V}(I_l)$ is closed, this shows that $\overline{\pi_l(\mathcal{V}(I))} \subset \mathcal{V}(I_l)$.
For the other inclusion, we first note that the statement

$$\mathcal{V}(I_l) \subset \overline{\pi_l(\mathcal{V}(I))} = \mathcal{V}(\mathcal{I}(\pi_l(\mathcal{V}(I)))),$$

is equivalent to

$$\mathcal{I}(\pi_l(\mathcal{V}(I))) = \sqrt{\mathcal{I}(\pi_l(\mathcal{V}(I)))} \subset \sqrt{I_l}.$$

by Hilbert's Nullstellensatz (Theorem 5.2). Now let $f \in \mathcal{I}(\pi_l(\mathcal{V}(I)))$, which in particular means that $f \in \mathbb{K}[x_1, \ldots, x_l]$. For $(a_1, \ldots, a_n) \in \mathcal{V}(I)$, we have

$$0 = f(\pi_l(a_1, \ldots, a_n)) = f(a_1, \ldots, a_l) = f(a_1, \ldots, a_n).$$

Hilbert's Nullstellensatz implies that $f \in \mathcal{I}(\mathcal{V}(I)) = \sqrt{I}$, so there exists $k \in \mathbb{N}$ such that $f^k \in I$. Finally, $f \in \mathbb{K}[x_1, \ldots, x_l]$ shows that $f^k \in \mathbb{K}[x_1, \ldots, x_l]$, so $f \in \sqrt{I_l}$. □

**Example 6.8.** (a) We first give a counterexample showing that the assumption that $\mathbb{K}$ is algebraically closed is really needed for the theorem to be true.
For this, let $I = (x^2 + y^2 + 1) \subset \mathbb{R}[x, y]$. Then $\mathcal{V}(I) = \emptyset$ and $\pi_1(\mathcal{V}(I)) = \emptyset$, but $I_1 = (0)$ ($I$ is a Gröbner basis), showing that $\mathcal{V}(I_1) = \mathbb{R} \neq \emptyset = \overline{\pi_1(\mathcal{V}(I))}$.

(b) We now give an example demonstrating that the closure operation in the theorem cannot be omitted. Consider $I = (xy - 1)$, so $\mathcal{V}(I)$ is essentially an hyperbola. Then $I_1 = (0)$ and $\mathcal{V}(I_1) = \mathbb{K}$ but $\pi_1(\mathcal{V}(I)) = \mathbb{K} \setminus \{0\}$.
By considering this example over a finite field, we also otain another counterexample similar to that in (a).

(c) Let $\mathbb{K}$ be algebraically closed and consider $I = ((x + 1)(y - 2), 4x^2 - 4xy + y^2)$ with Gröbner basis $G = \left\{x^2 - 2x + \frac{1}{4}y^2 + y - 2, (x + 1)(y - 2), (y - 2)(y + 2)^2\right\}$ w.r.t. the lexicographic ordering $x > y$. Eliminating $x$, we obtain $I_1 = \left((y - 2)(y + 2)^2\right)$, so the Theorem implies that $\overline{\pi_1(\mathcal{V}(I))} = \mathcal{V}(I_1) = \{\pm 2\}$. Because finite sets are always closed, it follows that $\pi_1(\mathcal{V}(I)) = \{\pm 2\}$. Substituting this into $G$ yields

$$y = +2: \quad x^2 - 2x + 1 = 0 \implies x = 1,$$
$$y = -2: \quad x^2 - 2x - 3 = 0 \implies x = -1.$$

In total, we conclude that $\mathcal{V}(I) = \{(1, 2), (-1, -2)\}$.

The method in (c) can be generalized, which results in the following algorithm. It allows us to solve systems of algebraic equations under the assumption that the field is algebraically closed and the solution set is finite. Indeed, if the solution set is infinite, it is not even clear how to represent it. This stands in contrast to the case of linear equations, for which we could simply return a basis.

**Algorithm 6.9 (Solving Systems of Algebraic Equations).**
Let $\mathbb{K}$ be algebraically closed.

```
 Input: $f_1, \dots, f_m \in \mathbb{K}[x_1, \dots, x_n]$.
Output: $\mathcal{V}((f_1, \dots, f_m))$ if that set is finite and "$\infty$" otherwise.
    (1) Compute a (reduced) Gröbner basis $G$ of $(f_1, \dots, f_m)$ w.r.t. the lexicographic
        ordering $x_1 < \dots < x_n$.
    (2) Set $M := \{()\} \subset \mathbb{K}^0$.
    (3) For $l = 1, \dots, n$:
        (4) Set $S := \emptyset$.
        (5) For $(a_1, \dots, a_{l-1}) \in M$:
            (6) Define $g := \gcd(\{f(a_1, \dots, a_{l-1}, x_l) : f \in \mathbb{K}[x_1, \dots, x_l] \cap G\})$.
            (7) If $g = 0$: return "$\infty$".
            (8) Set $S := S \cup \{(a_1, \dots, a_l) : a_l \in \mathbb{K}, g(a_l) = 0\}$.
        (9) Set $M := S$.
   (10) Return $M$.
```

It should be noted that step (8) requires numerical methods in order to solve the polynomial equation in a single variable.

Lec 26 2022-02-01

But elimination ideals are not only useful for computing affine varieties. They can also be used to compute the intersection of two ideals, which we now briefly highlight.

**Proposition 6.10.** Let $I, J \subset \mathbb{K}[x_1, \dots, x_n]$ be two ideals and consider the ideal

$$L := (y \cdot I, (1 - y) \cdot J) \subset \mathbb{K}[x_1, \dots, x_n, y],$$

where $y$ is another indeterminate. Then

$$I \cap J = \mathbb{K}[x_1, \dots, x_n] \cap L.$$

*Proof.* For $f \in I \cap J$, we have $f \in \mathbb{K}[x_1, \dots, x_n]$ and $f = y \cdot f + (1-y) \cdot f \in L$, so $I \cap J \subset \mathbb{K}[x_1, \dots, x_n] \cap L$.
For the other inclusion, let $f \in \mathbb{K}[x_1, \dots, x_n] \cap L$, so we can write

$$f = \sum_{i=1}^{r} y \cdot h_i f_i + \sum_{i=r+1}^{r+s} (1-y) \cdot h_i g_i \quad \text{for some} \quad h_i \in \mathbb{K}[x_1, \dots, x_n, y], f_i \in I, g_i \in J.$$

By setting $y = 0$ first and then setting $y = 1$, the claim follows:

$$f = \sum_{i=r+1}^{r+s} h_i(y=0) \cdot g_i \in J, \qquad f = \sum_{i=1}^{r} h_i(y=1) \cdot f_i \in I.$$

□

## 6.2 Dimensions

For the solution spaces of linear equations, we have a useful notion of dimension. We now introduce the analogous concept for affine varieties. We will then see that Gröbner bases allow us to compute those dimensions.

**Definition 6.11.** Polynomials $f_1, \dots, f_m \in \mathbb{K}[x_1, \dots, x_n]$ are called **algebraically independent**, if they obey no algebraic relation; i.e. for all polynomials $F \in \mathbb{K}[y_1, \dots, y_m] \setminus \{0\}$, we have $F(f_1, \dots, f_m) \neq 0$.
For an ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$, the $f_i$ are called **algebraically independent modulo** $I$, if their equivalence classes are algebraically independent in $\mathbb{K}[x_1, \dots, x_n]/I$.

**Example 6.12.** Let $x_{i_j} \in \mathbb{K}[x_1, \dots, x_n]$ be distinct indeterminants. They are algebraically independent module an ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$ if and only if $I \cap \mathbb{K}[x_{i_1}, \dots, x_{i_m}] = \{0\}$. Note that this is an elimination ideal.

**Definition 6.13.** The **(Krull) dimension** of an ideal $I \subsetneq \mathbb{K}[x_1, \dots, x_n]$ is defined to be

$$\dim(I) \coloneqq \sup\{k \in \mathbb{N} : \exists\, f_1, \dots, f_k \in \mathbb{K}[x_1, \dots, x_n] \text{ algebraically independent mod } I\}$$

Furthermore, we set $\dim(\mathbb{K}[x_1, \dots, x_n]) \coloneqq -1$.
If $\mathbb{K}$ is algebraically closed, we define the **(Krull) dimension** of an affine variety $X = \mathcal{V}(I)$ to be $\dim(X) \coloneqq \dim(I)$.

An equivalent definition of the Krull dimension is

$$\dim(I) = \operatorname{trdeg}_{\mathbb{K}}\{\mathbb{K}[x_1, \dots, x_n]/I\}.$$

The last part of the definition is well-defined, because two ideals $I, J \subset \mathbb{K}[x_1, \dots, x_n]$ with $\mathcal{V}(I) = \mathcal{V}(J)$ satisfy $\sqrt{I} = \sqrt{J}$ by Hilbert's Nullstellensatz (Theorem 5.2), and clearly $\dim(I) = \dim(\sqrt{I})$ for any ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$.
To distinguish the notations, we denote the (ordinary) dimension of a $\mathbb{K}$-vector space $V$ by $\dim_{\mathbb{K}}(V)$.

It can be shown that this definition for affine varities agrees with the intuitive notion of dimension: "points" are zero dimensional; "lines" are one dimensional, etc.
In order to develop some form of geometric interpretation, we introduce maps that

"preserve" the structure of an affine variety. For affine varieties $X = \mathcal{V}(I) \subset \mathbb{K}^n$, $Y = \mathcal{V}(J) \subset \mathbb{K}^m$ a *morphism of affine varieties* is a map of the form

$$f\colon X \to Y,\ x \mapsto (f_1(x), \dots, f_m(x))$$

for some polynomials $f_1, \dots, f_m \in \mathbb{K}[x_1, \dots, x_n]$.
Together with the affine varieties as objects, this defines the *category of affine varieties.*

**Lemma 6.14.** Let $\mathbb{K}$ be algebraically closed, $X = \mathcal{V}(I) \subset \mathbb{K}^n$ an affine variety induced by the ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$ and $f_1, \dots, f_m \in \mathbb{K}[x_1, \dots, x_n]$ polynomials. Consider the morphism of varieties $f$ induced by the $f_i$:

$$f\colon X \to \mathbb{K}^m,\ x \mapsto (f_1(x), \dots, f_m(x)).$$

Then $f_1, \dots, f_m$ are algebraically independent mod $I$ if and only if $\operatorname{im}(f) \subset \mathbb{K}^n$ is dense. In particular, $k := \dim(X)$ is the largest integer such that there exists a morphism $X \to \mathbb{K}^k$ with dense image.

*Proof.* We may view the $f_i$ and the elements of $I$ as polynomials in $\mathbb{K}[y_1, \dots, y_m, x_1, \dots, x_n]$. With the projection $\pi_y\colon \mathbb{K}^{m+n} \twoheadrightarrow \mathbb{K}^m$ onto the first $m$ coordinates, we have

$$\operatorname{im}(f) = \pi_y(\mathcal{V}((f_1 - y_1, \dots, f_m - y_m) \cup I)),$$

so Theorem 6.7 shows that the ideal

$$J := (I \cup \{f_1 - y_1, \dots, f_m - y_m\})$$

satisfies $\overline{\operatorname{im}(f)} = \mathcal{V}(J \cap \mathbb{K}[y_1, \dots, y_m])$. Therefore, $\operatorname{im}(f) \subset \mathbb{K}^n$ is dense if and only if $J \cap \mathbb{K}[y_1, \dots, y_m] = (0)$.
It is left to show that this condition in turn is equivalent to the algebraic independence mod $I$ of the $f_1, \dots, f_m$. To that end, let $F \in \mathbb{K}[y_1, \dots, y_m]$ be a polynomial. Because we have

$$F(y_1, \dots, y_m) \equiv F(f_1, \dots, f_m) \mod J.$$

and $F(f_1, \dots, f_m) \in \mathbb{K}[x_1, \dots, x_n]$, it follows that

$$F \in J \iff F(f_1, \dots, f_m) \in \mathbb{K}[x_1, \dots, x_n] \cap J.$$

*Claim:* $I = \mathbb{K}[x_1, \dots, x_n] \cap J$.
We have $I \subset \mathbb{K}[x_1, \dots, x_n] \cap J$ by definition of $J$. For the other inclusion, write $g \in \mathbb{K}[x_1, \dots, x_n] \cap J$ in the form

$$g = \sum_{i=1}^{r} h_i g_i + \sum_{i=1}^{m} l_i (f_i - y_i) \quad \text{with} \quad h_i, l_i \in \mathbb{K}[x_1, \dots, x_n, y_1, \dots, y_m],\ g_i \in I.$$

Evaluating $y_i = f_i$ for all $i \in \{1, \dots, m\}$ yields

$$g = \sum_{i=1}^{r} h_i(y_1 = f_1, \dots, y_m = f_m) g_i \in I,$$

so the claim follows.
We conclude that

$$F \in J \cap \mathbb{K}[y_1, \dots, y_m] \iff F(f_1, \dots, f_m) \in I$$

and in particular

$$J \cap \mathbb{K}[y_1, \dots, y_m] = (0) \iff f_1, \dots, f_m \text{ are algebraically independent mod } I,$$

which establishes the claim. □

Our next goal is to derive an easier characterization of the dimension of an ideal. This requires three lemmas.

**Lemma 6.15.** A non-empty set $M$ of ideals in $\mathbb{K}[x_1, \dots, x_n]$ has a maximal element.

*Proof.* If this were not the case, then there exists a strictly ascending chain

$$I_0 \subsetneq I_1 \subsetneq I_2 \subsetneq \dots \quad \text{with } I_i \in M.$$

Then $I \coloneqq \bigcup_{i=1}^{\infty} I_i$ is an ideal, so by *Hilbert's basis theorem* (Theorem 5.2) there exist finitely many generators $f_1, \dots, f_r \in \mathbb{K}[x_1, \dots, x_n]$ of $I$. But this means that there exists some $i \in \mathbb{N}_{>0}$, such that $f_j \in I_i$ for all $j \in \{1, \dots, r\}$, implying $I_{i+1} \subset I = (f_1, \dots, f_r) \subset I_i \subset I_{i+1}$, which contradicts our assumption.
If one is familiar with the concept of Noetherian rings, this is immediate: In a Noetherian ring, any strictly ascending chain of ideals must be finite. □

**Lemma 6.16.** Let $I \subsetneq \mathbb{K}[x_1, \dots, x_n]$ be a proper radical ideal. Then $I$ is can be written as the intersection of finitely many prime ideals.

*Proof.* Aiming for contradiction, assume that this is not the case. Then Lemma 6.15 yields an ideal $I$ that is maximal with this property. In particular, $I$ cannot be prime, so there exist $a, b \in \mathbb{K}[x_1, \dots, x_n] \setminus I$ with $a \cdot b \in I$. Consider

$$I_1 \coloneqq \sqrt{I + (a)}, \qquad I_2 \coloneqq \sqrt{I + (b)}.$$

*Claim:* $I = I_1 \cap I_2$.
It is clear that $I \subset I_1 \cap I_2$. For the other inclusion, let $f \in I_1 \cap I_2$. This means that there exists $m \in \mathbb{N}$, such that $f^m \in I + (a)$ and $f^m \in I + (b)$. Therefore, $f^{2m} \in (I + (a))(I + (b)) \subset I$ and because $I$ is radical this implies $f \in I$ and establishes the claim.
Since $I \subset I_1$ and $I \subset I_2$, it follows from the maximality of $I$ that both $I_1$ and $I_2$ are finite intersections of prime ideals, so the same holds true for $I$, which contradicts our assumption. □

Lec 27
2022-02-03

**Lemma 6.17.** Let $L = \mathbb{K}(\alpha_1, \dots, \alpha_n)$ be a finite field extension. Then $\operatorname{trdeg}_{\mathbb{K}}(L)$ is the size of a maximal algebraically independent subset of $\{\alpha_1, \dots, \alpha_n\}$.

We can now state and prove the theorem that gives an easier characterization of the dimension of an ideal.

**Theorem 6.18.** For an ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$, we have

$$\dim(I) = \max\{k \in \mathbb{N} : \exists\, i_1 < \dots < i_k, \{x_{i_1}, \dots, x_{i_k}\} \text{ algebraically independent mod } I\}.$$

*Proof.* Because each $x_{i_j}$ is a polynomial, it is clear that the right side is less or equal to the left one.
For the other inequality, let $f_1, \dots, f_r \in \mathbb{K}[x_1, \dots, x_n]$ be algebraically independent mod $I$. We need to show that there exist $i_1, \dots, i_r \in \mathbb{N}$, $1 \leq i_1 < \dots < i_r \leq n$ such that $\{x_{i_1}, \dots, x_{i_r}\}$ is algebraically independent mod $I$. If $\sqrt{I} = \mathbb{K}[x_1, \dots, x_n]$, then $I = \mathbb{K}[x_1, \dots, x_n]$ and the claim is clear, so we may assume that $\sqrt{I} \subsetneq \mathbb{K}[x_1, \dots, x_n]$. By Lemma 6.16, we may write

$$\sqrt{I} = P_1 \cap \ldots \cap P_s \quad \text{with } P_i \in \operatorname{Spec}(\mathbb{K}[x_1, \dots, x_n]).$$

Aiming for contradiction, assume that for every $i \in \{1, \dots, s\}$, $\{f_j : j \in \{1, \dots, r\}\}$ is algebraically dependent mod $P_i$; that is, there exist polynomials $F_i \in \mathbb{K}[y_1, \dots, y_r] \setminus \{0\}$ with $F_i(f_1, \dots, f_r) \in P_i$. This implies

$$\prod_{i=1}^{s} F_i(f_1, \dots, f_r) \in \bigcap_{i=1}^{s} P_i = \sqrt{I},$$

so there exists $m \in \mathbb{N}$ such that

$$G = \prod_{i=1}^{s} F_i^m \neq 0$$

satisfies $G(f_1, \dots, f_r) \in I$, contradicting our assumption that the $f_1, \dots, f_r$ are algebraically independent mod $I$.
Thus, there exists some $i \in \{1, \dots, s\}$, such that the $f_1, \dots, f_r$ are algebraically independent mod $P_i$, so their equivalence classes in $L := \operatorname{Quot}(\mathbb{K}[x_1, \dots, x_n]/P_i)$ are algebraically independent as well. Therefore, $\operatorname{trdeg}_{\mathbb{K}}(L) \geq r$ and since the $x_i$ generate $L$ as a field extension over $\mathbb{K}$, Lemma 6.17 guarantees the existence of algebraically independent elements $x_{i_1}, \dots, x_{i_r} \in L$. In particular, they are algebraically independent mod $P_i$ and thus mod $I$. □

In particular, the dimension of any ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$ is at most $n$.

**Example 6.19.** The ideal $I = (xy, xz) \subset \mathbb{K}[x, y, z]$ (which is a Gröbner basis) induces the affine variety

$$\mathcal{V}(I) = \big\{(a, b, c) \in \mathbb{K}^3 : a = 0 \text{ or } b = c = 0\big\}.$$

Because the algebraically independent mod $I$ subsets of $\{x, y, z\}$ are $\emptyset, \{x\}, \{y\}, \{z\}, \{y, z\}$, it follows $\dim(I) = 2$.

As a consequence, we get the following characterization of zero-dimensional proper ideals $I \subsetneq \mathbb{K}[x_1, \dots, x_n]$:

$$\begin{aligned} \dim(I) = 0 &\iff \text{Every singleton } \{x_i\} \text{ for } i \in \{1, \dots, n\} \text{ is algebraic mod } I. \\ &\iff \dim_{\mathbb{K}}(\mathbb{K}[x_1, \dots, x_n]/I) < \infty \end{aligned}$$

Now we can sketch our first algorithm to compute the dimension of an ideal: For any subset $S \subset \{x_1 \dots, x_n\}$, we compute a Gröbner basis $G$ w.r.t. some $S$-elimination ordering. By Theorem 6.3, $S$ is algebraically independent if and only if every element of $G$ contains some variable that is not in $S$. Then the size of the largest such $S$ is the dimension of the given ideal.

It is clear that this algorithm is highly inefficient because of the large amount of Gröbner bases that have to be computed.
In order to find a superior algorithm, we take a detour to the *Hilbert series.*

We have a canonical $\mathbb{K}$-algebra homomorphism

$$\mathbb{K}[x_1,\ldots,x_n] \to \mathbb{K}^{\mathbb{K}^n} = \{f\colon \mathbb{K}^n \to \mathbb{K}\},\ f \mapsto (x \mapsto f(x))$$

that assigns to each polynomial its polynomial function. For $X \subset \mathbb{K}^n$ an affine variety and $I = \mathcal{I}(X)$, we may compose this map with the $\mathbb{K}$-algebra homomorphism $\mathbb{K}^{\mathbb{K}^n} \to \mathbb{K}^X$ induced by the inclusion $X \hookrightarrow \mathbb{K}^n$ and the resulting map has kernel equal to $I$, so writing $A = \mathbb{K}[x_1,\ldots,x_n]/I$, we obtain an injective $\mathbb{K}$-algebra homomorphism

$$\phi\colon A \to \mathbb{K}^X = \{f\colon X \to \mathbb{K}\},\ f \mapsto (x \mapsto f(x)).$$

By definition, the image $\operatorname{im}(\phi)$ (called *ring of regular functions*) consists precisely of those functions that can be written as a polynomial expression in $x$ and $A$ is isomorphic to that subalgebra.
The main idea is to study the affine variety $X$ by considering its regular functions or equivalently the $\mathbb{K}$-algebra $A = \mathbb{K}[x_1,\ldots,x_n]/\mathcal{I}(X)$.

We saw that if $\dim(X) \neq 0$, then $\dim_{\mathbb{K}}(A) = \infty$. To better quantify the "size" of $A$, we consider a filtration $(A_d)_{d\in\mathbb{N}}$ as follows.

**Definition 6.20.** Denote the $\mathbb{K}$-subvector space of polynomials of degree $\leq d$ by $\mathbb{K}[x_1,\ldots,x_n]_{\leq d}$, where $\deg(f) := \max\{\deg(t) : t \in M(f)\}$.
Let $I \subset \mathbb{K}[x_1,\ldots,x_n]$ be an ideal and $A = \mathbb{K}[x_1,\ldots,x_n]/I$. For every $d \in \mathbb{N}$ the canonical projection $\pi\colon \mathbb{K}[x_1,\ldots,x_n] \twoheadrightarrow A$ induces a subspace

$$A_d := \pi(\mathbb{K}[x_1,\ldots,x_n]_{\leq d}) = \{f + I : \deg(f) \leq d\}.$$

The **Hilbert function** of $I$ is defined to be

$$h_I : \mathbb{N} \to \mathbb{N},\ d \mapsto \dim_{\mathbb{K}}(A_d)$$

and the **Hilbert series** of $I$ is the formal power series

$$H_I(t) = \sum_{d=0}^{\infty} h_I(d)t^d \in \mathbb{Z}[\![t]\!].$$

**Example 6.21.**

(a) For $I = (x_1,\ldots,x_n) \subset \mathbb{K}[x_1,\ldots,x_n]$, we have $A = \mathbb{K}[x_1,\ldots,x_n]/I \cong \mathbb{K}$, so $h_I(d) = 1$ for all $d \in \mathbb{N}$ and $H_I(t) = \frac{1}{1-t}$ by the geometric series.

(b) For $I = (x - y^2)$, a basis of $A_d$ (as a $\mathbb{K}$-vector space) is given by the equivalence classes of
$$\{1, x, \ldots, x^d, y, xy, \ldots, x^{d-1}y\},$$
so $h_I(d) = 2d + 1$ and $H_I(t) = \sum_{d=0}^{\infty}(2d+1)t^d = \frac{1+t}{(1-t)^2}$, as it is straightforward to confirm that $\left(\sum_{d=0}^{\infty}(2d+1)t^d\right) \cdot (1-t)^2 = 1 + t$.

(c) For $I = (0) \subset \mathbb{K}[x_1, \ldots, x_n]$ the zero ideal, we vary the number of variables $n$ in the polynomial ring and write $H_n$ for $H_{(0)}$ and $h_n$ for $h_{(0)}$. Using (a), it follows $H_0(t) = \frac{1}{1-t}$. For $n \geq 1$, we have an isomorphism of vector spaces

$$\mathbb{K}[x_1, \ldots, x_n]_{\leq d} \cong \bigoplus_{i+j=d} \mathbb{K}[x_1, \ldots, x_{n-1}]_{\leq i} \cdot x_n^j,$$

which implies

$$h_n(d) = \dim_{\mathbb{K}}(\mathbb{K}[x_1, \ldots, x_n]_{\leq d}) = \sum_{i=0}^{d} \dim_{\mathbb{K}}(\mathbb{K}[x_1, \ldots, x_{n-1}]_{\leq i}) = \sum_{i=0}^{d} h_{n-1}(i).$$

so by definition of the Cauchy product and induction, we conclude

$$H_n(t) = \sum_{d=0}^{\infty} h_n(d)t^d = \left(\sum_{d=0}^{\infty} t^d\right) \cdot \left(\sum_{d=0}^{\infty} h_{n-1}(d)t^d\right) = \frac{1}{1-t} H_{n-1}(t) = \frac{1}{(1-t)^{n+1}}.$$

But this formal power series can also be written as

$$H_n(t) = \sum_{d=0}^{\infty} \binom{-n-1}{d} (-t)^d = \sum_{d=0}^{\infty} \binom{n+d}{d} t^d = \sum_{d=0}^{\infty} \binom{n+d}{n} t^d,$$

thus we deduce $h_n(d) = \binom{n+d}{n}$.

We now start to draw the connection to Gröbner bases.

**Definition 6.22.** A monomial ordering $\leq$ is called a **total degree ordering** if

$$\deg(t) < \deg(t') \implies t < t' \qquad \text{for all } t, t' \in M.$$

Intuitively, a total degree ordering is just a monomial ordering which compares by degree "first" and only if they are equal resorts to other criteria. Examples include the graded lexicographic ordering and its reversed version.

Our interest in total degree orderings is motivated by the following theorem.

**Theorem 6.23.** Let $\leq$ be a total degree ordering. Then for any ideal $I \subset \mathbb{K}[x_1, \ldots, x_n]$, its Hilbert function and series is equal to that of its leading ideal $\mathrm{L}(I)$:

$$h_I(t) = h_{\mathrm{L}(I)}(t), \qquad H_I(t) = H_{\mathrm{L}(I)}(t).$$

*Proof.* Write $A = \mathbb{K}[x_1, \ldots, x_n]/I$ and let $G$ be a Gröbner basis of $I$. By Theorem 5.16, the normal form $\mathrm{NF}_G$ induces an injective $\mathbb{K}$-linear map

$$\phi\colon A \to \mathbb{K}[x_1, \ldots, x_n],\ f + I \mapsto \mathrm{NF}_G(f).$$

For $d \in \mathbb{N}$, we consider its restriction

$$\phi_d \coloneqq \phi|_{A_d}\colon A_d \to \mathbb{K}[x_1, \ldots, x_n],\ f + I \mapsto \mathrm{NF}_G(f)$$

and denote by $V_d$ the $\mathbb{K}$-subspace spanned by all monomials $t \in M \subset \mathbb{K}[x_1, \ldots, x_n]$, such that $\deg(t) \leq d$ and $t \notin \mathrm{L}(I)$.

*Claim:* $V_d = \operatorname{im}(\phi_d)$.
Because all elements of $V_d$ are already in normal form w.r.t. $G$, it directly follows that $V_d \subset \operatorname{im}(\phi_d)$. For the other inclusion, let $f + I \in A_d$. By definition of the normal form, there exist $h_i \in \mathbb{K}[x_1, \dots, x_n], g_i \in G$, such that

$$\mathrm{NF}_G(f) = f + \sum_{i=1}^{r} h_i g_i \quad \text{and} \quad \mathrm{LM}(h_i g_i) \leq \mathrm{LM}(f).$$

Since $\leq$ is a total degree ordering, we must have $\deg(h_i g_i) \leq \deg(f) \leq d$. Therefore, we have $\deg(\mathrm{NF}_G(f)) \leq d$ and because none of the monomials of $\mathrm{NF}_G(f)$ lie in $\mathrm{L}(G) = \mathrm{L}(I)$, this shows that $\mathrm{NF}_G(f) \in V_d$ and establishes the claim.
The claim implies that $h_I(d) = \dim_{\mathbb{K}}(V_d)$. But $V_d$ only depends on the leading ideal, so for any two ideals $I, J \subset \mathbb{K}[x_1, \dots, x_n]$ with $\mathrm{L}(I) = \mathrm{L}(I)$, we have $h_I = h_J$. Because $\mathrm{L}(\mathrm{L}(I)) = \mathrm{L}(I)$, the assertion follows. □

Therefore, if we work with a total degree ordering, then it suffices to be able to compute the Hilbert function for monomial ideals. Lec 28 2022-02-08

Our next goal is to derive an explicit formula for the Hilbert series.
To that end, we first derive an equivalent characterization of the Hilbert series using some basic facts about graded modules $M = \bigoplus_{i=0}^{\infty} M_i$ over a graded commutative ring $R = \bigoplus_{i=0}^{\infty} R_i$, which we now recall.
For one, the $M_i$ turn out to be not only abelian groups but even $R_0$-modules. A submodule $N \subset M$ inherits the grading and thus becomes a graded $R$-module if and only if it is *homogeneous*, which means that it can be generated by homogeneous elements. Equivalently, the homogeneous parts of any element in the submodule also lie in the submodule. Furthermore, if $N$ is homogeneous, then the quotient module $M/N$ has the following graded $R$-module structure:

$$M/N = \bigoplus_{i=0}^{\infty} M_i/N_i, \quad N_i = N \cap M_i.$$

We now apply this to the graded $\mathbb{K}[x_1, \dots, x_n]$-module $\mathbb{K}[x_1, \dots, x_n]$:

$$\mathbb{K}[x_1, \dots, x_n] = \bigoplus_{i=0}^{\infty} \mathbb{K}[x_1, \dots, x_n]_i, \quad \mathbb{K}[x_1, \dots, x_n]_i = \{f \in \mathbb{K}[x_1, \dots, x_n] : \deg(f) = i\}.$$

Then the homogeneous modules (i.e. ideals) are precisely the monomial ideals. Let $I$ be such an ideal and write $I_{\leq i} := \bigoplus_{j=0}^{i} I_j$.
The canonical projection $\mathbb{K}[x_1, \dots, x_n] \twoheadrightarrow \mathbb{K}[x_1, \dots, x_n]/I$ is not only a module homomorphism but also maps the $i$-th homogeneous component to the $i$-th homogeneous components; that is, it is an isomorphism of graded $\mathbb{K}[x_1, \dots, x_n]$-modules (and in particular of $\mathbb{K}$-vector spaces). Therefore, the restriction $\mathbb{K}[x_1, \dots, x_n]_{\leq d} \twoheadrightarrow (\mathbb{K}[x_1, \dots, x_n]/I)_{\leq d}$ is well-defined and surjective, so we obtain a $\mathbb{K}$-linear isomorphism

$$\mathbb{K}[x_1, \dots, x_n]_{\leq d}/I_{\leq d} \cong (\mathbb{K}[x_1, \dots, x_n]/I)_{\leq d}.$$

This gives another characterization of the Hilbert function $h_I(d) = \dim_{\mathbb{K}}((\mathbb{K}[x_1, \dots, x_n]/I)_{\leq d})$, which we capture in a lemma.

**Lemma 6.24.** For a monomial ideal $I \subset \mathbb{K}[x_1, \ldots, x_n]$, we have

$$h_I(d) = h_{(0)}(d) - \dim_{\mathbb{K}}(I_{\leq d})$$

and

$$H_I(t) = H_{(0)}(t) - \sum_{d=0}^{\infty} \dim_{\mathbb{K}}(I_{\leq d}) t^d.$$

Applying this to the case that $I = (m)$ is generated by a single monomial $m \in \mathbb{K}[x_1, \ldots, x_n]$ and using that for $d \geq \deg(m)$, multiplication by $m$ provides a $\mathbb{K}$-linear isomorphism $\mathbb{K}[x_1, \ldots, x_n]_{\leq d-\deg(m)} \cong I_{\leq d}$, we obtain

$$H_{(m)}(t) = H_{(0)}(t) - t^{\deg(m)} H_{(0)}(t) = H_{(0)}(t) \cdot (1 - t^{\deg(m)}). \tag{$*$}$$

Let $I = (m_1, \ldots, m_l) \subset \mathbb{K}[x_1, \ldots, x_n]$ be an ideal generated by monomials $m_i$ and consider $J \coloneqq (m_1, \ldots, m_{l-1})$. The projection

$$J \to I/(m_l), \; g \mapsto g + (m_l),$$

constitutes a surjective $\mathbb{K}[x_1, \ldots, x_n]$-module homomorphism with kernel $(m_l) \cap J$, so

$$J/((m_l) \cap J) \cong I/(m_l)$$

as $\mathbb{K}[x_1, \ldots, x_n]$-modules (and in particular as $\mathbb{K}$-vector spaces). Because it is also graded, we obtain a $\mathbb{K}$-linear isomorphism

$$(J_{\leq i})/((m_l) \cap J)_{\leq i} \cong I_{\leq i}/(m_l)_{\leq i},$$

which with Lemma 6.24 translates to

$$H_J(t) + H_{(m_l)}(t) = H_I(t) + H_{(m_l) \cap J}(t). \tag{$**$}$$

With this observation, we can prove an explicit formula for the Hilbert series.

**Theorem 6.25.** Let $I = (m_1, \ldots, m_l) \subset \mathbb{K}[x_1, \ldots, x_n]$ be an ideal generated by monomials $m_i$. Then the Hilbert series of $I$ is

$$H_I(t) = \frac{1}{(1-t)^{n+1}} \sum_{S \subset \{1, \ldots, l\}} (-1)^{|S|} t^{\deg(\operatorname{lcm}\{m_i : i \in S\})}.$$

*Proof.* We prove this by induction on $l$. For $l = 0$, we have $I = (0)$ and $H_{(0)}(t) = \frac{1}{(1-t)^{n+1}}$ by Example 6.21. The case $l = 1$ follows directly from $(*)$.
For $l > 1$, we use $(**)$, $(*)$, the inductive hypothesis and the fact that

$$(m_l) \cap (m_1, \ldots, m_{l-1}) = (\operatorname{lcm}(m_1, m_l), \ldots, \operatorname{lcm}(m_{l-1}, m_l))$$

to conclude

$$\begin{aligned}(1-t)^{n+1} \cdot H_I(t) = &\sum_{S \subset \{1, \ldots, l-1\}} (-1)^{|S|} t^{\deg(\operatorname{lcm}\{m_i : i \in S\})} + \left(1 - t^{\deg(m_l)}\right) \\ &- \sum_{S \subset \{1, \ldots, l-1\}} (-1)^{|S|} t^{\deg(\operatorname{lcm}\{\operatorname{lcm}(m_i, m_l) : i \in S\})}\end{aligned}$$

Now the assertion follows from the observation

$$\operatorname{lcm}\{\operatorname{lcm}(m_i, m_l) : i \in S\} = \begin{cases} \operatorname{lcm}\{m_i : i \in S \cup \{l\}\} & S \neq \emptyset \\ 1 & S = \emptyset \end{cases}$$

and the decomposition

$$\{A : A \subset \{1, \dots, l\}\} = \{A : A \subset \{1, \dots, l-1\}\} \coprod \{A \cup \{l\} : A \subset \{1, \dots, l-1\}\}.$$

□

**Corollary 6.26 (Hilbert-Sierre theorem).**
The Hilbert function of any ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$ is a rational function of the form

$$H_I(t) = \frac{a_0 + a_1 t + \cdots + a_k t^k}{(1-t)^{n+1}} \qquad \text{for } a_i \in \mathbb{Z}.$$

Furthermore, the *Hilbert polynomial*, defined to be

$$p_I(d) := \sum_{i=0}^{k} a_i \binom{d-i+n}{n} \in \mathbb{Q}[d],$$

coincides with the Hilbert function $h_I(d)$ for all $d \geq k$.

*Proof.* The first claim is an immediate consequence of Theorem 6.23 and Theorem 6.25. Using Example 6.21, we see that for $i \in \{0, \dots, k\}$, we have

$$\frac{a_i t^i}{(1-t)^{n+1}} = a_i t^i \sum_{d=0}^{\infty} \binom{n+d}{n} t^d = \sum_{d=i}^{\infty} a_i \binom{n+d-i}{n} t^d.$$

For $d \geq i$, the coefficient of $t^d$ of this summand precisely corresponds to one of the summands in the definition of $p_I(d)$, so for $d \geq k$, we conclude $p_I(d) = h_I(d)$. □

We previously saw that for an ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$:

$$\dim(I) = 0 \iff 0 \neq \dim_{\mathbb{K}}(\mathbb{K}[x_1, \dots, x_n]/I) < \infty.$$

But this is equivalent to the Hilbert function $h_I(d)$ becoming a nonzero constant for large enough $d \in \mathbb{N}$, which by the previous theorem is equivalent to $\deg(p_I) = 0$. Therefore, in the case that $\dim(I) = 0$, we have $\dim(I) = \deg(p_I)$.
Our next goal is to generalize this result and show that it is true for any ideal that its dimension is precisely the degree of its Hilbert polynomial. In order to see this, we consider the case of finitely generated $\mathbb{K}$-algebras.

By the homomorphism theorem, any finitely generated $\mathbb{K}$-algebra $A$ is isomorphic to $\mathbb{K}[x_1, \dots, x_n]/I$ for some $n \in \mathbb{N}$ and ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$. Indeed, a $\mathbb{K}$-algebra epimorphism $\phi\colon \mathbb{K}[x_1, \dots, x_n] \twoheadrightarrow A$ precisely corresponds to a choice of generators $A = \mathbb{K}[a_1, \dots, a_n]$: Given $\phi$, we have $A = \mathbb{K}[\phi(x_1), \dots, \phi(x_n)]$ and on the other hand, generators $a_1, \dots, a_n$ induce the epimorphism $\mathbb{K}[x_1, \dots, x_n] \twoheadrightarrow A, x_i \mapsto a_i$.
Because there are many possible choices of generators (even the number of generators can vary), it is not clear how to generalize the Hilbert function to finitely generated $\mathbb{K}$-algebras.
However, it turns out that the degree of the Hilbert polynomial does not depend on the chosen isomorphism.

**Lemma 6.27.** Let $I \subset \mathbb{K}[x_1, \dots, x_n]$, $J \subset \mathbb{K}[y_1, \dots, y_m]$ be ideals, such that we have a $\mathbb{K}$-algebra isomorphism

$$\mathbb{K}[x_1, \dots, x_n]/I \cong \mathbb{K}[y_1, \dots, y_m]/J.$$

Then the degrees of the corresponding Hilbert polynomials agree: $\deg(p_I) = \deg(p_J)$.

*Proof.* Writing $A := \mathbb{K}[x_1, \dots, x_n]/I$ and $B = \mathbb{K}[y_1, \dots, y_m]/J$, let $\phi$ denote the isomorphism $A \cong B$. Then there exist $g_1, \dots, g_m \in \mathbb{K}[x_1, \dots, x_n]$, such that $\phi(g_i + I) = y_i + J$. With $l := \max\{\deg(g_i) : i \in \{1, \dots, m\}\}$, we have

$$B_{\leq d} \subset \phi(A_{\leq dl}) \implies h_J(d) \leq h_I(dl) \implies \deg(p_J) \leq \deg(p_I).$$

and the other inequality follows by symmetry. □

Lec 29
2022-02-10

**Theorem 6.28 (Noether normalization).** Let $A$ be a non-zero, finitely generated $\mathbb{K}$-algebra. Then there are $c_1, \dots, c_d \in A$ algebraically independent, such that $A$ is finitely generated as a module over $C := \mathbb{K}[c_1, \dots, c_d]$ ($d = 0$ means $C = \mathbb{K}$); i.e. $A = \sum_{i=1}^m C a_i$ for some $a_i \in A$.
Furthermore, for $A = \mathbb{K}[x_1, \dots, x_n]/I$, we have $d = \dim(I)$.

*Proof.* Let $A = \mathbb{K}[x_1, \dots, x_n]/I$ for some proper ideal $I \subsetneq \mathbb{K}[x_1, \dots, x_n]$. We use induction on $n$. The case $I = (0)$ is straightforward, as we can just choose $c_i = x_i$ and $d = n$. This includes the base case $n = 0$. For $n > 0$ and $I \neq (0)$, let $f = \sum_{t \in \mathrm{M}(f)} a_t t \in I \setminus \{0\}$, $a_t \in \mathbb{K}$ and set $m := \deg(f) + 1$. Then the map

$$S\colon \mathrm{M}(f) \to \mathbb{N},\ x_1^{e_1} \cdots x_n^{e_n} \mapsto \sum_{i=1}^n e_i \cdot m^{i-1}$$

(which essentially corresponds to a representation of the monomials of $f$ w.r.t. the basis $m$) is injective. For $i \in \{2, \dots, n\}$, we set $y_i := x_i - x_1^{m^{i-1}}$ and calculate

$$\begin{aligned}
f &= f(x_1, \dots, x_n) \\
&= f\Big(x_1, y_2 + x_1^m, \dots, y_n + x_1^{m^{n-1}}\Big) \\
&= \sum_{t = x_1^{e_1} \cdots x_n^{e_n} \in \mathrm{M}(f)} a_t x_1^{e_1} \prod_{i=2}^n \Big(y_i + x_1^{m^{i-1}}\Big)^{e_i} \\
&= \sum_{t \in \mathrm{M}(f)} a_t \Big(x_1^{S(t)} + g_t(x_1, y_2, \dots, y_n)\Big),
\end{aligned}$$

where $a_t \in \mathbb{K}$ and $g_t \in \mathbb{K}[x_1, \dots, x_n]$ satisfies $\deg_{x_1}(g_t) < S(t)$. Because $S$ is injective, there exists exactly one monomial $t \in \mathrm{M}(f)$, such that $k := S(t)$ is maximal, so $f$ is of the form

$$f = a_t x_1^k + h(x_1, y_2, \dots, y_n)$$

for some $h \in \mathbb{K}[x_1, \dots, x_n]$ with $\deg_{x_1}(h) < k$. Since $f \in I$, it follows

$$x_1^k + {a_t}^{-1} h(x_1, y_2, \dots, y_n) \in I \qquad (*)$$

and $B := \mathbb{K}[y_2 + I, \dots, y_n + I] \subset A$ satisfies

$$A = \sum_{i=0}^{k-1} B(x_1 + I)^i.$$

By the inductive hypothesis, there exist algebraically independent elements $c_1, \dots, c_d \subset B$ and such that $B = \sum_{j=1}^{l} \mathbb{K}[c_1, \dots, c_d]b_j$ for some $b_1, \dots, b_l \in B$. Therefore, the first part of the claim follows:

$$A = \sum_{i=0}^{k-1} \sum_{j=1}^{l} \mathbb{K}[c_1, \dots, c_d]b_j(x_1 + I)^i.$$

It is left to show that $d = \dim(I)$. By the inductive hypothesis, we have $d = \operatorname{trdeg}_{\mathbb{K}}(B)$ and because $B \subset A$, it follows $d \leq \operatorname{trdeg}_{\mathbb{K}}(A)$. For the other inequality, let $f_1, \dots, f_r \in \mathbb{K}[x_1, \dots, x_n]$ be algebraically independent mod $I$. Because $\dim(I) = \dim(\sqrt{I})$, we may assume that $I$ is a radical ideal. By the same argument as in the proof of Theorem 6.18, there exists a prime ideal $P \subset \mathbb{K}[x_1, \dots, x_n]$ containing $I$, such that $f_1, \dots, f_r$ are algebraically independent mod $P$. Therefore, the field $L := \operatorname{Quot}(\mathbb{K}[x_1, \dots, x_n]/P)$ satisfies $\operatorname{trdeg}_{\mathbb{K}}(L) \geq r$ and by $(*)$, $L$ is algebraic over

$$L' := \operatorname{Quot}(\mathbb{K}[y_2 + P, \dots, y_n + P]) \subset L,$$

so $\operatorname{trdeg}_{\mathbb{K}}(L') \geq r$. The canonical projections $\mathbb{K}[y_2, \dots, y_n] \twoheadrightarrow B$ and the inclusion $I \subset P$ induce $\mathbb{K}$-algebra homomorphisms

$$B \cong \mathbb{K}[y_2, \dots, y_n]/(\mathbb{K}[y_2, \dots, y_n] \cap I) \twoheadrightarrow \mathbb{K}[y_2, \dots, y_n]/(\mathbb{K}[y_2, \dots, y_n] \cap P).$$

and similarly, we have $\mathbb{K}[y_2, \dots, y_n]/(\mathbb{K}[y_1, \dots, y_n] \cap P) \cong \mathbb{K}[y_2 + P \dots, y_n + P]$. We conclude

$$d = \operatorname{trdeg}_{\mathbb{K}}(B) \geq \operatorname{trdeg}_{\mathbb{K}}(\mathbb{K}[y_2 + P, \dots, y_n + P]) = \operatorname{trdeg}_{\mathbb{K}}(L') \geq r,$$

and thus $\operatorname{trdeg}_{\mathbb{K}}(A) \leq d$. □

Noether normalization can be interpreted geometrically. Indeed, let $X \subset \mathbb{K}^n$ be an affine variety and $A = \mathbb{K}[x_1, \dots, x_n]/\mathcal{I}(X)$ the ring of regular functions on $X$. Then choosing $c_1, \dots, c_d \in A$ corresponds to choosing a morphism of varities

$$f\colon X \to \mathbb{K}^d,\ v \mapsto (c_1(v), \dots, c_d(v)).$$

It can be shown that if the $c_i$ are chosen according to the theorem, then $f$ is surjective and has finite fibers $f^{-1}(\{y\})$ for all $y \in \mathbb{K}^d$.

**Example 6.29.** Consider the hyperbola $X = \mathcal{V}(xy - 1) \subset \mathbb{C}^2$. Then projecting to the $x$ or $y$ axis is not surjective, as 0 is not in the image. One can show that $c = x - y$ (which corresponds to projecting onto the “diagonal”) constitutes a Noether normalization.

**Theorem 6.30.** For any ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$, we have $\dim(I) = \deg(p_I)$.

*Proof.* The case $I = \mathbb{K}[x_1, \dots, x_n]$ is covered in Example 6.21. Thus we assume $I \subsetneq \mathbb{K}[x_1, \dots, x_n]$ and set $A := \mathbb{K}[x_1, \dots, x_n]/I$. By Noether normalization (Theorem 6.28), there exist algebraically independent elements $c_1, \dots, c_m$, such that $A = \sum_{i=1}^{l} \mathbb{K}[c_1, \dots, c_m]b_i$

with $m = \dim(I)$ for some $b_1, \dots, b_l \in A$. We may assume that $b_1 = 1$ by "shifting" the $b_i$ and setting $b_1 = 1$. The $\mathbb{K}$-algebra homomorphism

$$\phi \colon \mathbb{K}[y_1, \dots, y_d, z_1, \dots, z_l] \to A,\ y_i \mapsto c_i,\ z_i \mapsto b_i$$

is surjective, so with $J = \ker(\phi)$ and $B := \mathbb{K}[y_1, \dots, y_m, z_1, \dots, z_l]/J$, we have $A \cong B$. By Lemma 6.27, it suffices to show that $\deg(p_J) = m$. For $d \in \mathbb{N}$, we write

$$B_{\leq d} = \{f + J : f \in \mathbb{K}[y_1, \dots, y_m, z_1, \dots, z_l], \deg(f) \leq d\}$$

and

$$C_{\leq d} = \{f + J : f \in \mathbb{K}[y_1, \dots, y_m], \deg(f) \leq d\}.$$

Using $C_{\leq d} \subset B_{\leq d}$ for all $d \in \mathbb{N}$, the algebraic independence of the $c_i$ and Example 6.21, it follows

$$h_J(d) = \dim(B_{\leq d}) \geq \dim(C_{\leq d}) = \dim(\mathbb{K}[y_1, \dots, y_m]) = \binom{m+d}{m}.$$

This shows that $\deg(p_J) \geq m$.
For the other inequality, we first observe that for $1 \leq i \leq j \leq l$, we can write

$$b_i \cdot b_j = \sum_{k=1}^{l} a_{i,j,k} b_k \quad \text{for some } a_{i,j,k} \in \mathbb{K}[c_1, \dots, c_m].$$

With $e := \max_{i,j,k} a_{i,j,k}$, this implies $b_i \cdot b_j \in \sum_{k=1}^{l} C_{\leq e} b_k$. Iterating this, it follows that the product of $s$ ($s \in \mathbb{N}_{>0}$) of the $b_i$ lies in $\sum_{k=1}^{l} C_{\leq (s-1)e} b_k$. Therefore, for all $d \geq 0$, we have

$$B_{\leq d} \subset C_{\leq d} \cdot b_1 + \sum_{s=1}^{d} \sum_{k=1}^{l} C_{\leq d-s} C_{(s-1)e} b_k \subset \sum_{k=1}^{l} C_{\leq de} b_k =: V_d.$$

Thus the assertion follows with the algebraic independence of the $c_i$ and Example 6.21:

$$h_J(d) = \dim(B_{\leq d}) \leq \dim(V_d) \leq l \cdot \dim(C_{\leq de}) = \dim(\mathbb{K}[y_1, \dots, y_m]) = l \cdot \binom{m+de}{m}.$$

□

**Corollary 6.31.** For any ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$ and total degree ordering $\leq$, we have

$$\dim(I) = \dim(\mathrm{L}(I)).$$

*Proof.* This is a direct consequence of Theorem 6.23, Corollary 6.26 and Theorem 6.30. □

In fact, it can be shown that the previous corollary holds true for any monomial ordering.

Our interest in the Hilbert function and series was mainly motivated by this result. Now that we have established it, we can determine the dimension of an ideal in a more efficient way and without computing the Hilbert function or series:

Let $I \subset \mathbb{K}[x_1, \dots, x_n]$ be an ideal with leading ideal $L(I) = (m_1, \dots, m_k)$.
By Theorem 6.18 and Corollary 6.31, we have $\dim(I) = n - |S|$ for any subset $S \subset$

$\{x_1, \ldots, x_n\}$ of minimal size such that $\{x_1, \ldots, x_n\} \setminus S$ is algebraically independent mod $(m_1, \ldots, m_k)$. This is equivalent to the elimination ideal $(m_1, \ldots, m_k) \cap \mathbb{K}[\{x_1, \ldots, x_n\} \setminus S]$ being zero, which means that its reduced Gröbner basis is empty. Using Theorem 6.3 and the fact that $(m_1, \ldots, m_k)$ is a Gröbner basis of $L(I)$ w.r.t. any monomial ordering, the above is equivalent to $\{m_1, \ldots, m_k\} \cap \mathbb{K}[\{x_1, \ldots, x_n\} \setminus S] = \emptyset$.
This observation gives rise to our final algorithm, which only requires computing a single Gröbner basis.

**Algorithm 6.32.**

```
Input: $I \subset \mathbb{K}[x_1, \ldots, x_n]$ ideal.
Output: Krull dimension $\dim(I)$
    (1) Compute a Gröbner basis $G$ of $I$. Let $m_1, \ldots, m_l$ be the leading monomials of
        the elements of $G$.
    (2) If one of the $m_i$ is constant: return $-1$.
    (3) Find a minimal subset $S \subset \{x_1, \ldots, x_n\}$ such that every $m_i$ contains some
        variable of $S$.
    (4) Return $n - |S|$.
```