Journey Two:
Heirs of Pythagoras

How division with remainder led to discovery of many fundamental abstractions.

Heirs of Pythagoras

Lecture 1

Mathematics

- Science of numbers and space.

- Numbers and space are our innate abilities.

Pythagorean School

- Square, oblong, triangular numbers
- Discrete structure of space
- Arithmetization of geometry
- A catastrophe
  - the program cannot succeed

He discovered “the theory of irrationals...”

Proclus

Common Measure

- A segment *V* is a *measure* of a segment *A* iff *A* can be represented as a finite concatenation of copies of *V*.
- A segment *V* is a *common measure* of segments *A* and *B* iff it is a measure of both.
- A segment *V* is the greatest common measure of *A* and *B* if it is greater than any other common measure of *A* and *B*.

Properties of *gcm*

$$\begin{aligned}
& gcm(a,a) = a \\
& gcm(a,b) = gcm(a,a+b) \\
b < a \implies\ & gcm(a,b) = gcm(a-b,b) \\
& gcm(a,b) = gcm(b,a)
\end{aligned}$$

Line segment *gcm*

```
line_segment gcm(line_segment a,
                 line_segment b) {
    if (a == b) return a;
    if (b < a)  return gcm(a - b, b);
    if (a < b)  return gcm(a, b - a);
}
```

gcm(196in, 42in)

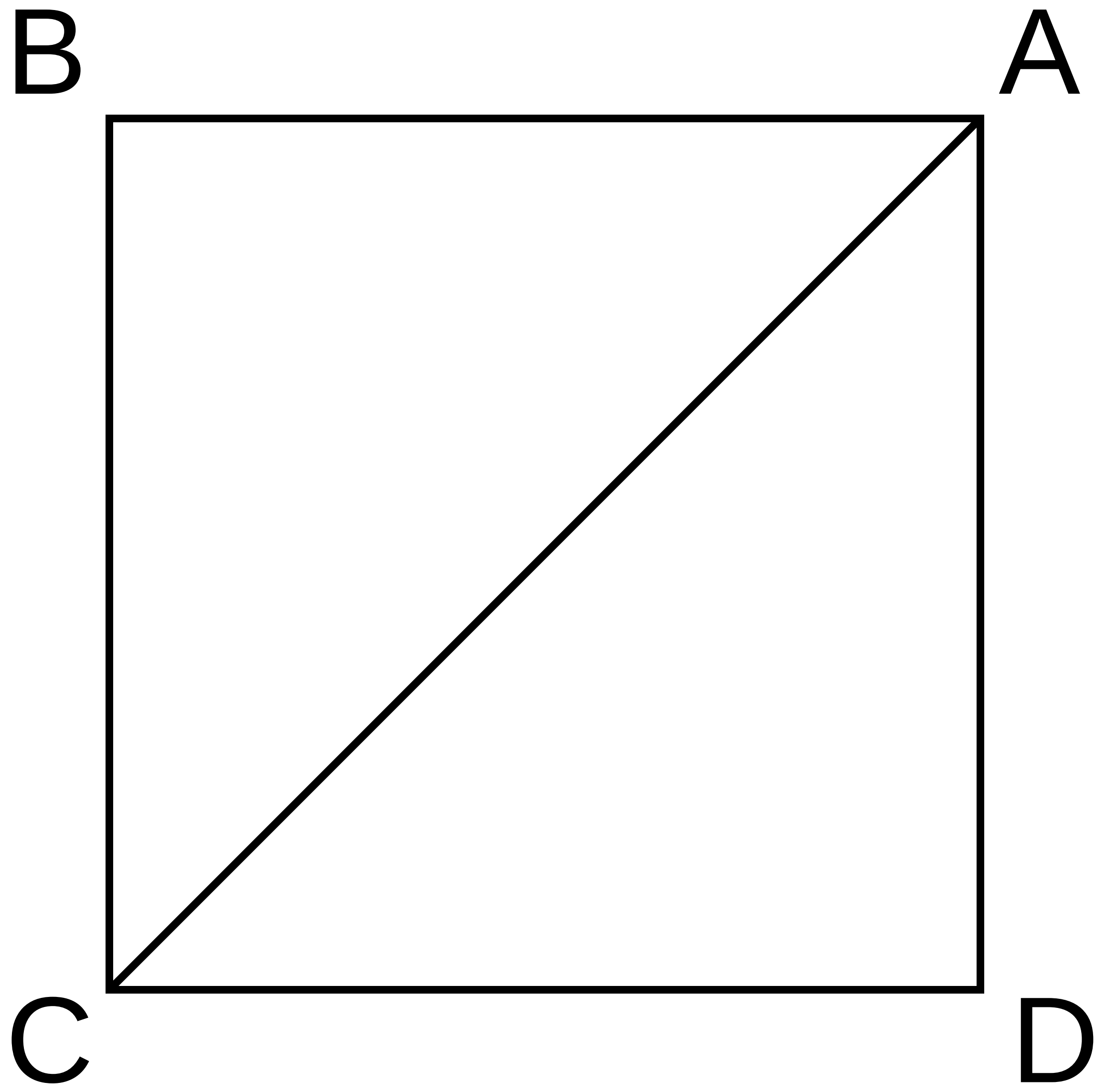
$$\begin{array}{rcll}
196 & > & 42 & \\
154 & > & 42 & \\
112 & > & 42 & \\
70 & > & 42 & \\
28 & < & 42 & \\
28 & > & 14 & \\
14 & = & 14 & \text{Done!}
\end{array}$$

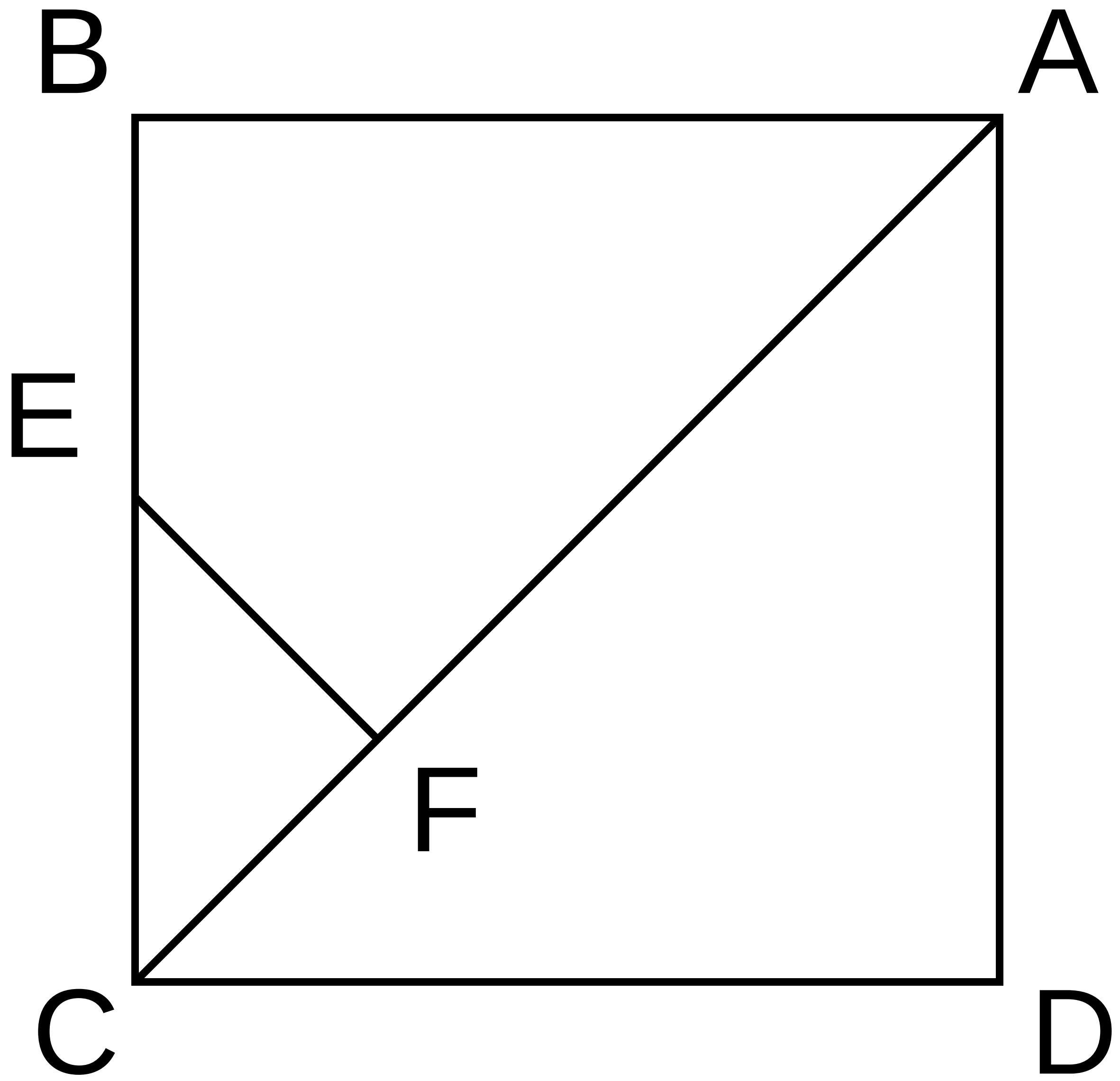
Well-ordering Principle

- Greeks found the principle that any set of natural numbers has a smallest element a very powerful proof technique.
  - It is equivalent to induction axiom.
- In order to prove that something does not exist, prove that if it exists, a smaller one also exists.

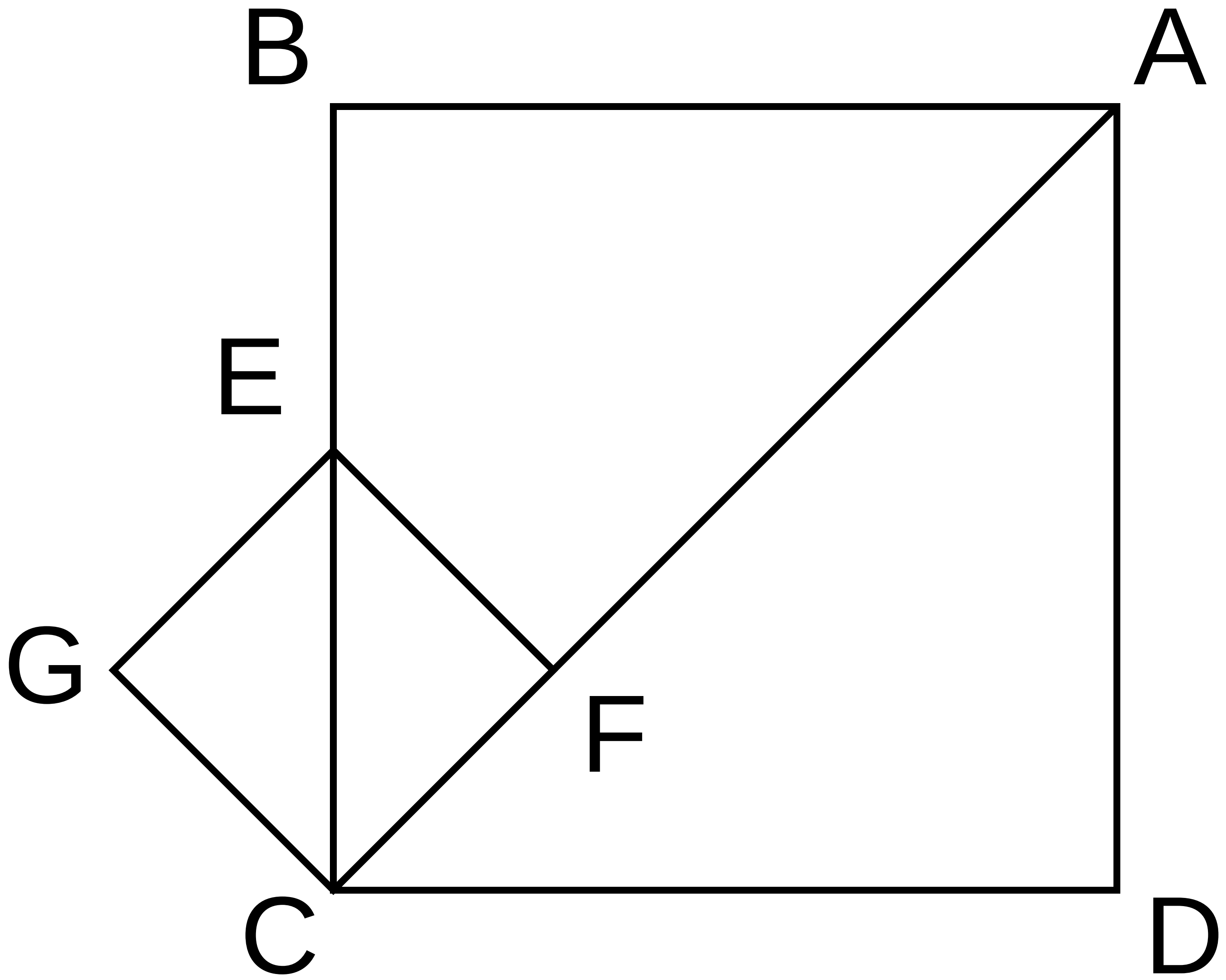
The Assumption

- Let us assume that there is a segment that can measure both the side and the diagonal of some square. Let us take the smallest such square for this segment.

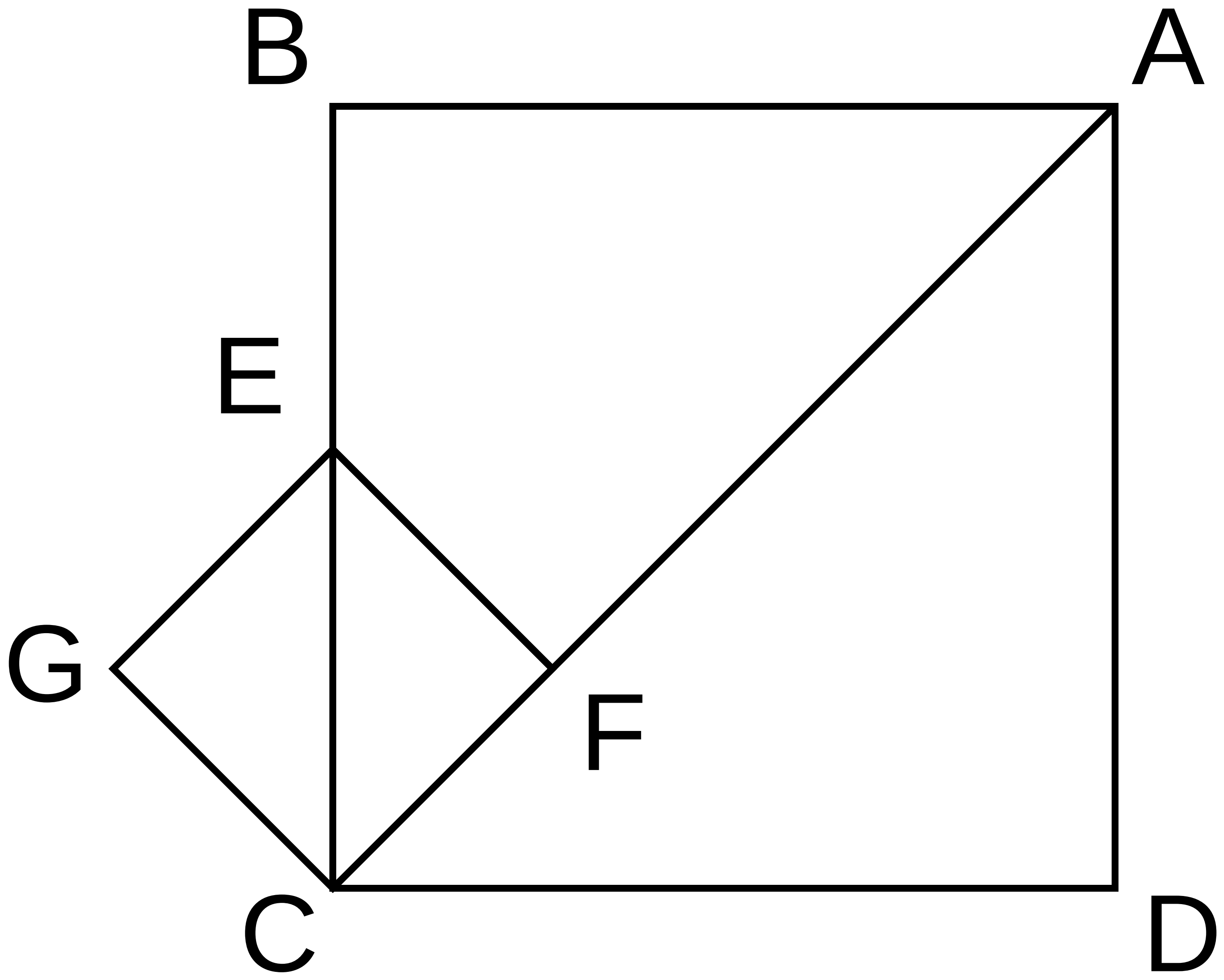

B
A
C
D



$$\overline{AB} = \overline{AF} \quad \wedge \quad \overline{AC} \perp \overline{EF}$$

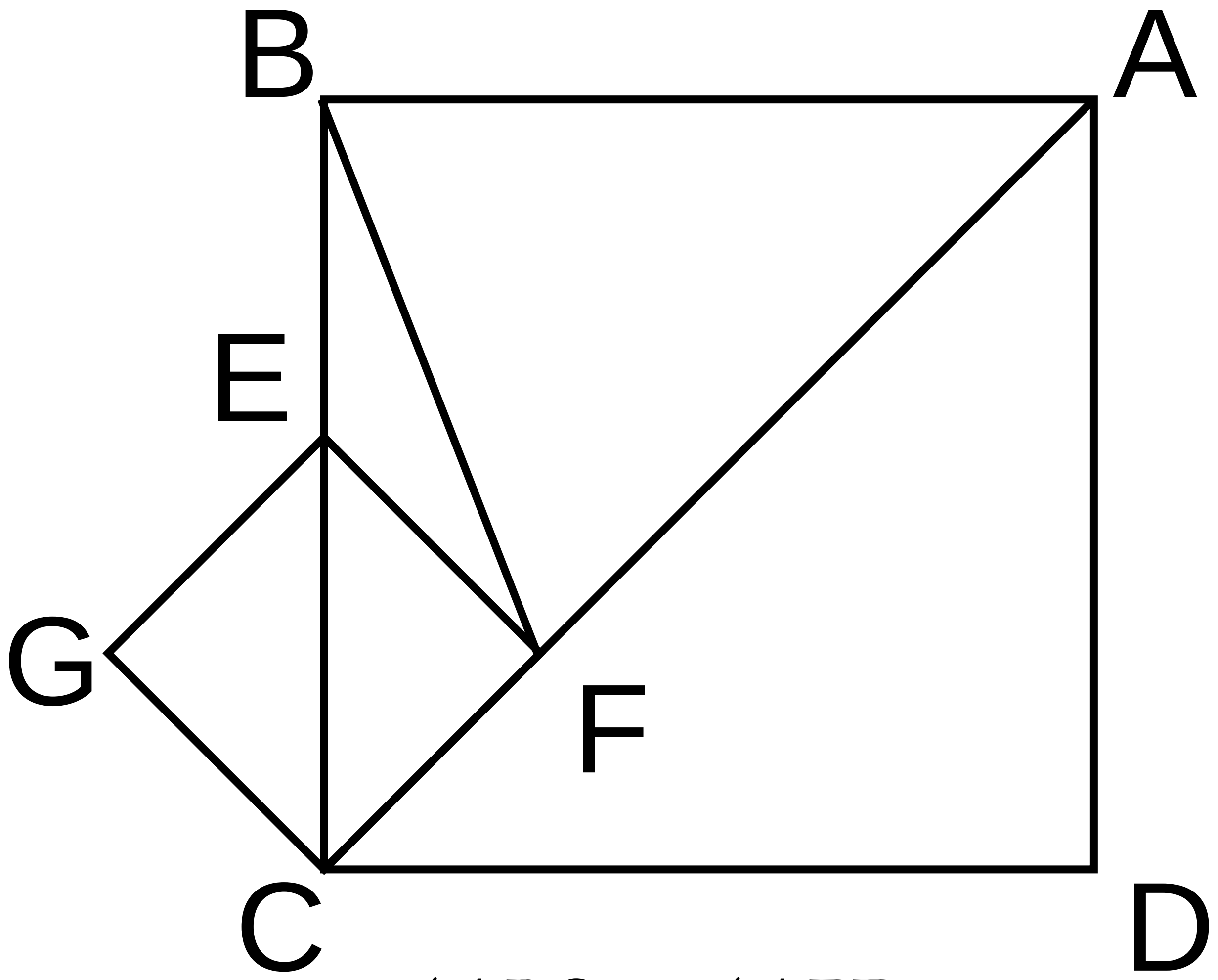


$$\overline{AB} = \overline{AF} \ \wedge \ \overline{AC} \perp \overline{EF}$$



$$\angle CEF = 180^\circ - \angle CFE - \angle ECF = 45^\circ$$

$$\angle CEF = \angle ECF \implies \overline{FC} = \overline{FE}$$

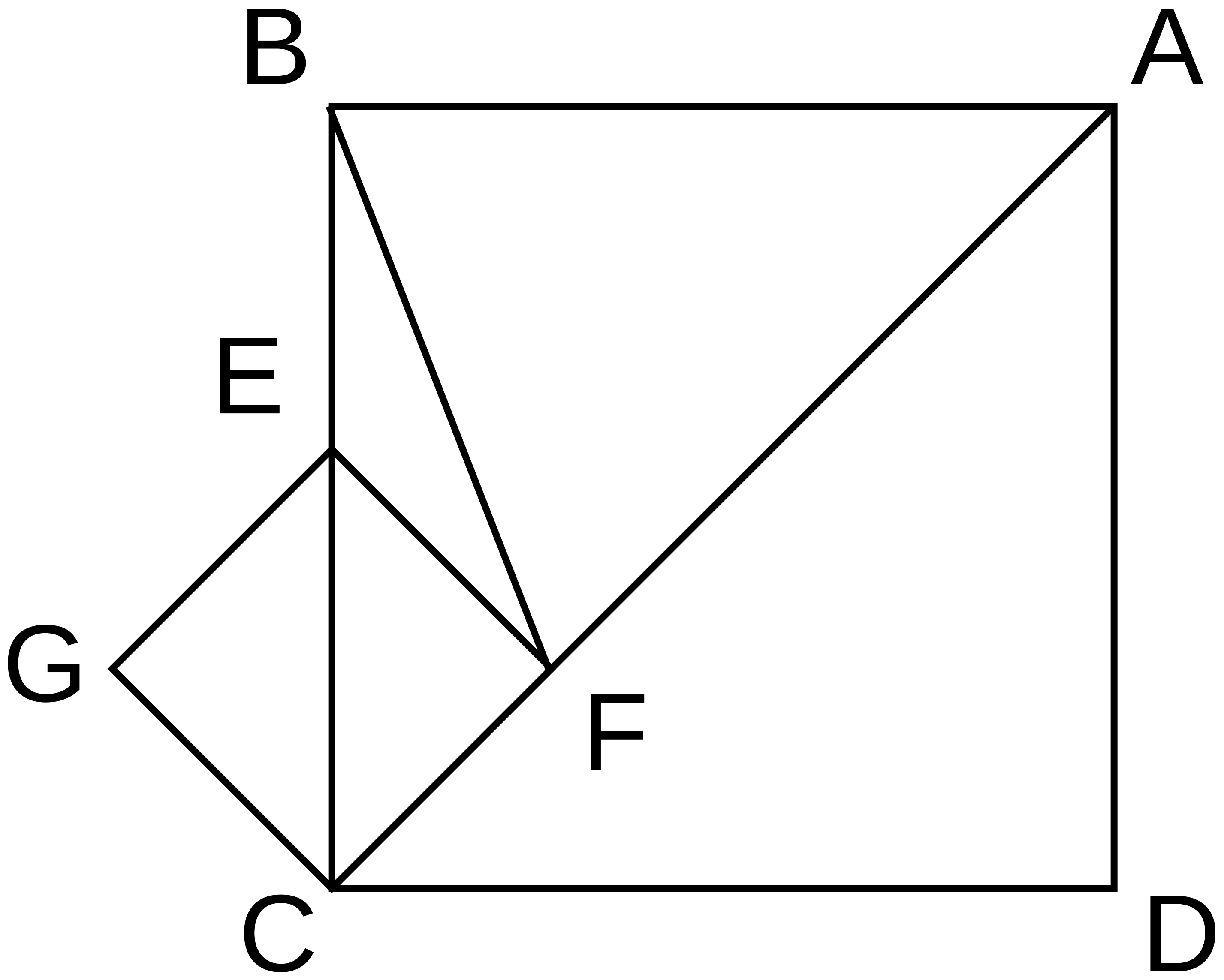


$$\angle ABC = \angle AFE$$

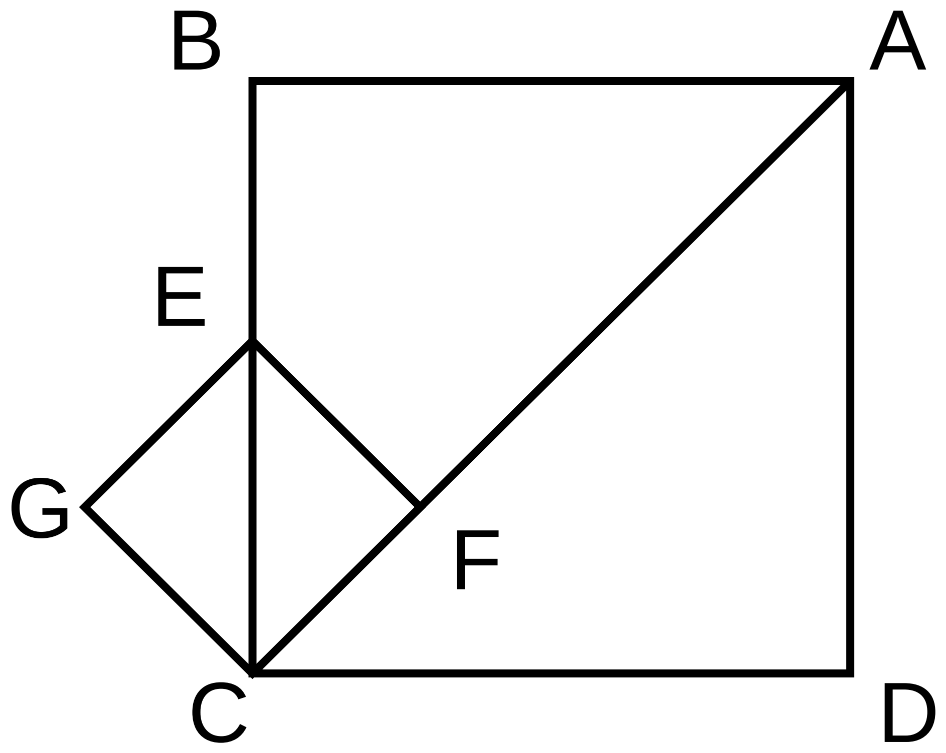
$$\angle ABF = \angle AFB$$

$$\angle ABC - \angle ABF = \angle AFE - \angle AFB$$

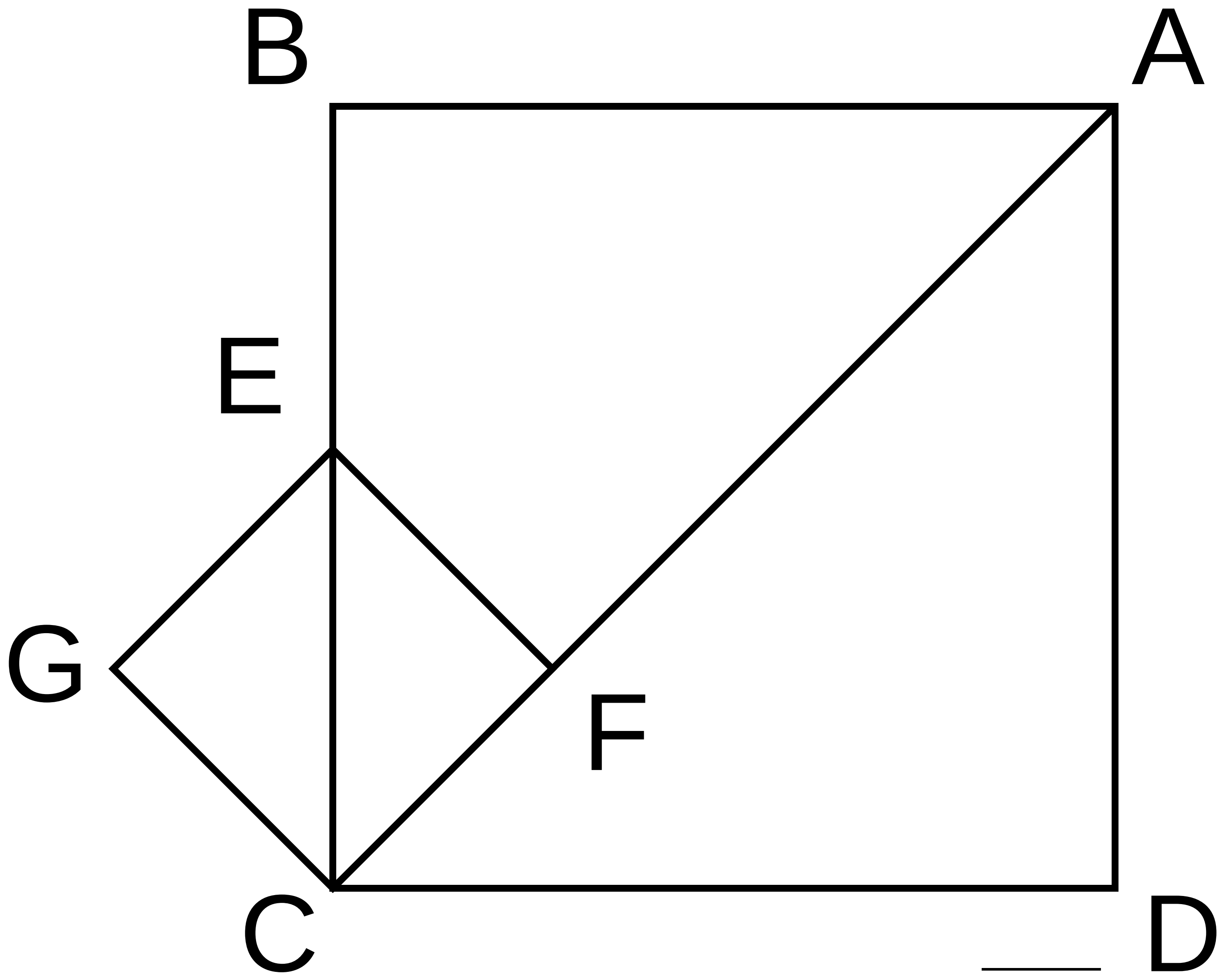
$$\angle EBF = \angle EFB \implies \overline{BE} = \overline{EF}$$



$$\overline{FC} = \overline{FE} = \overline{BE}$$



$$gcm(\overline{AC},\overline{AB}) = gcm(\overline{AB},\overline{CF}) = gcm(\overline{CE},\overline{CF})$$



$$\overline{EB} < \overline{EC} \implies \overline{EF} < \frac{\overline{AB}}{2}$$

Q.E.D.

We constructed a smaller square that could be measured by the segment. Contradiction!

Revolutionary Discovery

In modern terms,

Pythagoras discovered that $\sqrt{2}$ is irrational

Number-theoretic proof of irrationality of √2

Assume that $(m/n)^2 = 2$ and that m/n is irreducible.
$m^2 = 2n^2$.
m is even.
$m = 2u$.
$(2u)^2 = 2n^2$.
$4u^2 = 2n^2$.
$2u^2 = n^2$.
n is even.
Contradiction.

Dilemma

To ground geometry in arithmetic we need a unit of measurement. The unit must be small enough to measure all the segments involved in a given problem.

The proof we just have seen demonstrates that such a unit cannot exist.

The Perpetual Conflict

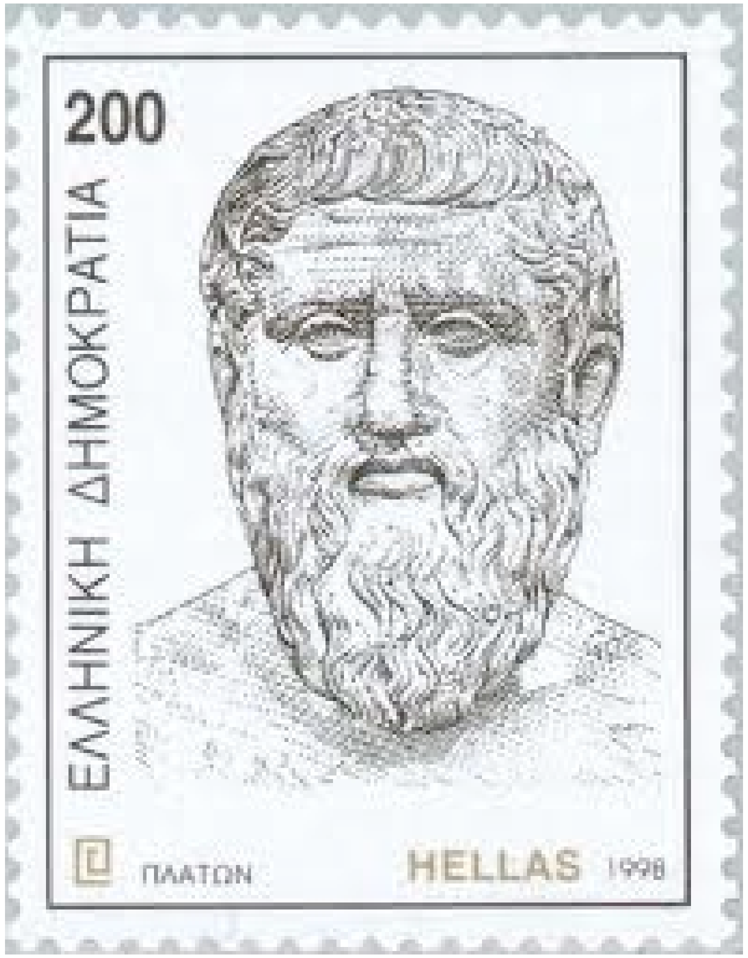
The tension between continuous and discrete remains central even today and, in my opinion, will remain with mathematics forever.

The tension has been the source of progress and revolutionary insights for many ages.

Athens in V century BC

- Marathon, Salamis, Themistocles, Aristides, Pericles, Aspasia, Aeschylus, Sophocles, Euripides, Aristophanes, Thucydides, Parthenon, Phidias…
- “…we are the school of Hellas…”

Plato (427BC - 347BC)



Significance of Plato

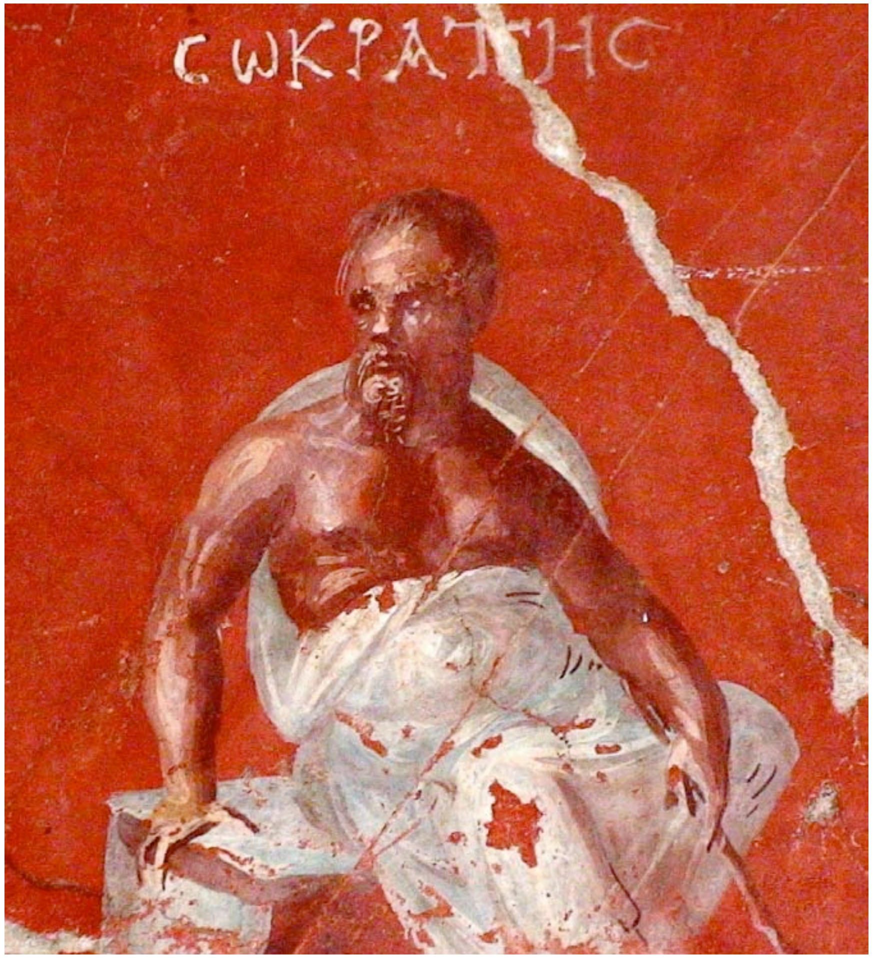
The safest general characterization of the European philosophical tradition is that it consists of a series of footnotes to Plato.

Alfred North Whitehead
Process and Reality

Ariston, the Poet

You gaze at stars, my Aster.
O, that I were heaven
To look back at you
With innumerable eyes.

Socrates 470BC – 399BC



The Gadfly

- I neither know nor think that I know.
- The life which is unexamined is not worth living.
- The hour of departure has arrived, and we go our ways - I to die, and you to live. Which is better God only knows.

Plato’s life after 399

- Megara, Cyrene, Heliopolis, Tarentum, Syracuse (399BC – 387BC)
  - sold into slavery by Dionysius I
- Starts the Academy near Athens (385BC)
- Syracuse (366BC)
  - “the palace was filled with sawdust, as they say, owing to the multitude of geometricians there.”
- Syracuse (361BC)
- Death (347BC)

Academy

- Μηδείς αγεωμέτρητος εισίτω
- Let no one ignorant of geometry enter
- 10 out of 15 years of study were fully dedicated to mathematics

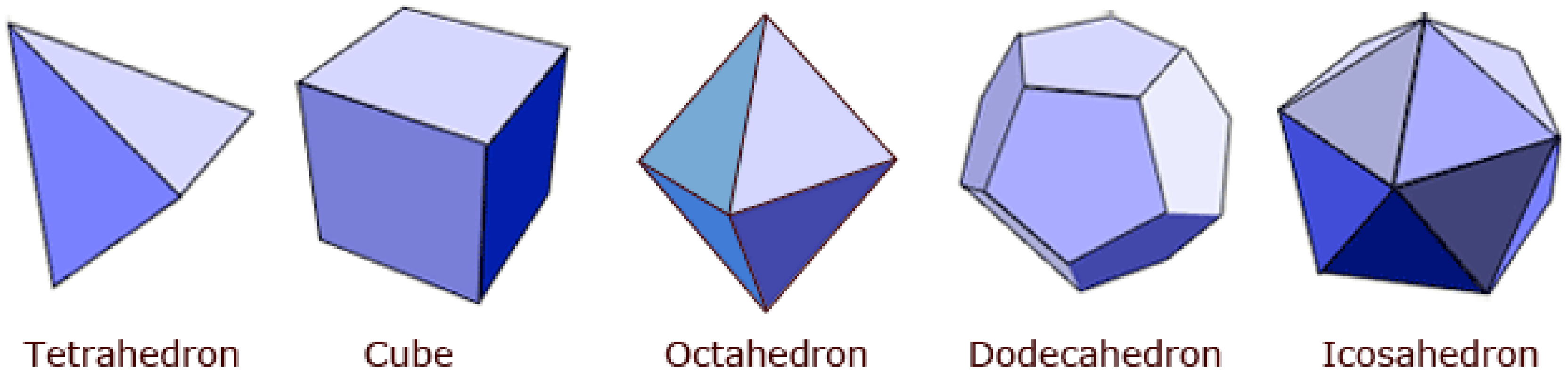
“The Century of Plato”

“At the center of scientific life stood the personality of Plato. He guided and inspired scientific work in his Academia and outside.”

B. L. van der Waerden

Science Awakening

Platonic Solids



Delian Problem

- Doubling the cube
- Several mechanical solutions by Plato’s students
- Fully solved in 1837 by Pierre Wantzel

Popular Reception

[Athenians] came to Plato’s lecture on the Good in the conviction that they would get some one or other of the things that the world calls good: riches, or health, or strength. But when they found that Plato’s reasonings were of mathematics their disenchantment was complete.

Aristoxenus

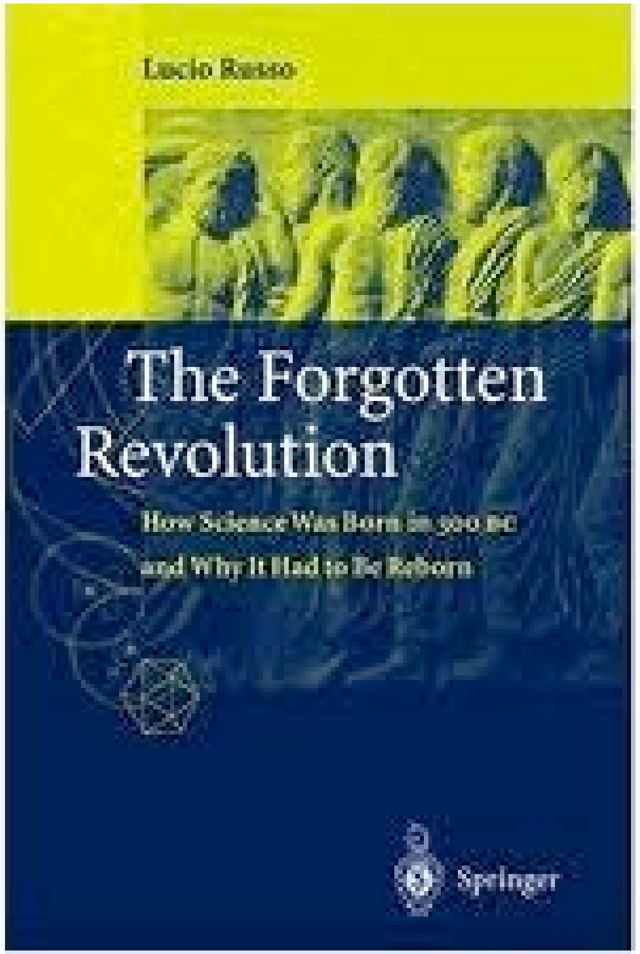
Alexandria

- Founded in 331BC
- Mouseion
  - over 1000 scholars
  - free room and board
- Lasting influence
  - canon
  - texts

Bibliotheca

- Patronage by Ptolemy (I, II, III, IV, V, VI, VII)
- 500000 scrolls
  - manuscript acquisitions
  - copying
  - translating
  - editing

Hellenistic Science


Lucio Russo

The Forgotten Revolution

How Science Was Born in 300 BC and Why It Had to Be Reborn

Springer

Euclid (325BC-265BC)

What do we gain from Euclid?

Some one who had begun to read geometry with Euclid, when he had learnt the first theorem, asked Euclid, “what shall I get by learning these things?” Euclid called his slave and said, “Give him three pence, since he must make gain out of what he learns.”

Strobaeus, *Florilegium*

Royal Road?

Ptolemy once asked Euclid whether there was any shorter way to a knowledge of geometry than by study of the *Elements*, whereupon Euclid answered that there was no royal road to geometry.

Proclus
Commentary on Elements

Poetic Insight

O blinding hour, O holy, terrible day,
When first the shaft into his vision shone
Of light anatomized! Euclid alone
Has looked on Beauty bare. Fortunate they
Who, though once only and then but far away,
Have heard her massive sandal set on stone.

Edna St Vincent Millay

Elements, Book X

Proposition 2. If, when the less of two unequal magnitudes is continually subtracted in turn from the greater that which is left never measures the one before it, then the two magnitudes are incommensurable.

gcm0

```
line_segment gcm0(line_segment a,
                  line_segment b) {
  while (a != b) {
    if (b < a) a = a - b;
    else       b = b - a;
  }
  return a;
}
```

a and *b* are incommensurable iff gcm0 does not terminate.

Problem 46

gcm0 is inefficient when one segment is much longer than the other. Come up with a more efficient implementation.

Problem 47

Prove that if a segment measures two other segments, then it measures their greatest common measure.

Heirs of Pythagoras

Lecture 2

gcm0 (repeat)

```
line_segment gcm0(line_segment a,
                  line_segment b) {
  while (a != b) {
    if (b < a) a = a - b;
    else       b = b - a;
  }
  return a;
}
```

gcm1

```
line_segment gcm1(line_segment a,
                  line_segment b) {
  while (a != b) {
    while (b < a) a = a – b;
    std::swap(a, b);

  }
  return a;
}
```

segment_remainder

```
line_segment
segment_remainder(line_segment a,
                  line_segment b) {
    while (b < a) a = a - b;
    return a;
}
```

Why does this terminate?

```
while (b < a) a = a - b;
```

Axiom of Archimedes:

$$\forall a, b \; \exists n : a \leq nb$$

(For Greeks segments are never 0.)

segment_gcm

```
line_segment
segment_gcm(line_segment a,
            line_segment b) {
  while (a != b) {
    a = segment_remainder(a, b);
    std::swap(a, b);
  }
  return a;
}
```

Recursive Remainder Lemma

If $r = \text{segment\_remainder}(a, 2b)$ then

$$\text{segment\_remainder}(a, b) = \begin{cases} r & \text{if } r \leq b \\ r - b & \text{if } r > b \end{cases}$$

fast_segment_remainder

```
line_segment
fast_segment_remainder(line_segment a,
                       line_segment b)
{
  if (a <= b) return a;
  if (a - b <= b) return a - b;
  a = fast_segment_remainder(a, b + b);
  if (a <= b) return a;
  return a - b;
}
```

fast_segment_gcm

```
line_segment
fast_segment_gcm(line_segment a,
                 line_segment b) {
while (a != b) {
    a = fast_segment_remainder(a, b);
    std::swap(a, b);
  }
  return a;
}
```

Discovery of 0

- Babylonian astronomers used 0 with base 60 positional notation.
  - The rest of their society used decimal non-positional notation.
- Greek astronomers used base 60 positional notation for their trigonometric tables.

Abacus

- Known throughout from ancient China to Rome
- Positional decimal representation
- Still, no written representation of 0 for 1000 years!

Decimal 0

- Indian mathematicians combined “natural” decimal integers with positional notation and 0
  - The notation spread through India to Persia from 6th to 9th century AD.
- Arab scholars adopted it and it was taught from Bagdad to Cairo to Cordoba.

Pisa

Leonardo Pisano (1170-1250)

Leonardo’s Journeys

- “In his boyhood” visits his father in Bugia, Algiers where he studies “Hindu digits”.
- Later, while on business, he studied further techniques in Egypt, Syria, Greece, Sicily, and Provence.
- Leonardo and Frederick II (Michael Scott, John of Palermo.)
- Given a salary by Pisa in recognition of his accomplishments.

Leonardo’s Books

- Liber Abaci (1st edition, 1203)
  - Book of Computations
- Practica Geometriae (1223)
- Flos (1225)
- Liber Quadratorum (1226)
- Liber Abaci (2nd edition, 1228)

Problem 64 (easy)

Prove that

$$\sqrt[3]{16} + \sqrt[3]{54} = \sqrt[3]{250}$$

Why was it difficult for the Greeks?

Problem 65

For any odd square number *x* find an even square number *y*, such that *x* + *y* is a square number.

Liber Quadratorum

Problem 66 (hard)

If x and y are both sums of two squares, then so is their product xy.

Liber Quadratorum

Zero segments

- We can view AA as a segment.
- That forces the re-adjustment in our procedures.

fast_segment_remainder1

```
line_segment
fast_segment_remainder1(line_segment a,
                        line_segment b)
{
  // precondition: b != 0
  if (a < b) return a;
  if (a - b < b) return a - b;
  a = fast_segment_remainder1(a, b + b);
  if (a < b) return a;
  return a - b;
}
```

Halving

- It is possible to divide a segment in two equal parts with ruler and compass.

- That allows us to eliminate the recursion from the fast remainder algorithm.

largest_doubling

```
line_segment
largest_doubling(line_segment a,
                 line_segment b) {
  // precondition: b ≠ 0
  while (b <= a – b) b = b + b;
  return b;
}
```

remainder

```
line_segment remainder(line_segment a,
                       line_segment b) {
  // precondition: b ≠ 0
  if (a < b) return a;
  line_segment c = largest_doubling(a, b);
  a = a – c;
  while (c != b) {
    c = half(c);
    if (c <= a) a = a – c;
  }
  return a;
}
```

Computing quotient

```
integer quotient(line_segment a, line_segment b) {
  // Precondition: b > 0
  if (a < b) return integer(0);
  line_segment c = largest_doubling(a, b);
  integer n(1);
  a = a – c;
  while (c != b) {
      c = half(c); n = n + n;
      if (c <= a) { a = a – c; n = n + 1; }
  }
  return n;
}
```

Ahmes Knew It

- A primitive version of quotient algorithm appears in the Rhind papyrus.
- It is an “algorithmic inverse” of Egyptian multiplication.
- It was known to the Greeks as Egyptian division.

Computing quotient and remainder

```
std::pair<integer, line_segment>
quotient_remainder(line_segment a, line_segment b) {
  // Precondition: b > 0
  if (a < b) return std::make_pair(integer(0), a);
  line_segment c = largest_doubling(a, b);
  integer n(1);
  a = a – c;
  while (c != b) {
      c = half(c); n = n + n;
      if (c <= a) { a = a – c; n = n + 1; }
  }
  return std::make_pair(n, a);
}
```

Computing both quotient and remainder is not more complex than just quotient.

Floyd-Knuth: no halving

```
line_segment remainder_fibonacci(
       line_segment a, line_segment b) {
  // Precondition: b > 0
  if (a < b) return a;
  line_segment c = b;
  do {
    line_segment tmp = c; c = b + c; b = tmp;
  } while (a >= c);
  do {
    if (a >= b) a = a - b;
    line_segment tmp = c – b; c = b; b = tmp;
  } while (b < c);
  return a;
}
```

Problem 76

Design quotient_fibonacci and quotient_remainder_fibonacci.

gcm_remainder

```
line_segment
gcm_remainder(line_segment a,
              line_segment b) {
  while (b != line_segment(0)) {
    a = remainder(a, b);
    std::swap(a, b);
  }
  return a;
}
```

Euclidean algorithm for integers

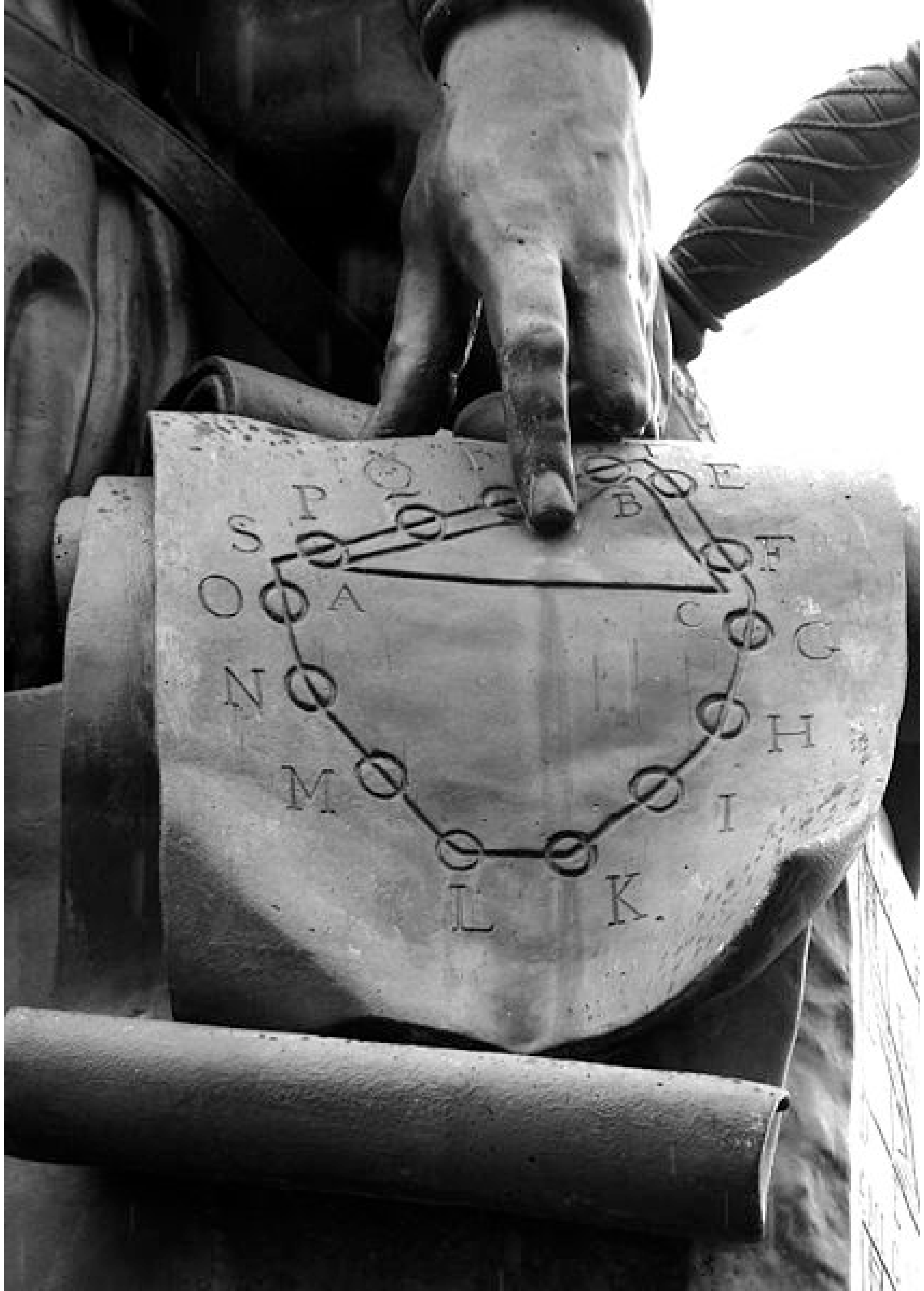
```
integer gcd(integer a, integer b) {
  while (b != integer(0)) {
    a = a % b;
    std::swap(a, b);
  }
  return a;
}
```

General Treatment

- Read *EoP*, chapter 5

Dutch Golden Age (1568–1700)

Simon Stevin (1548 – 1620)

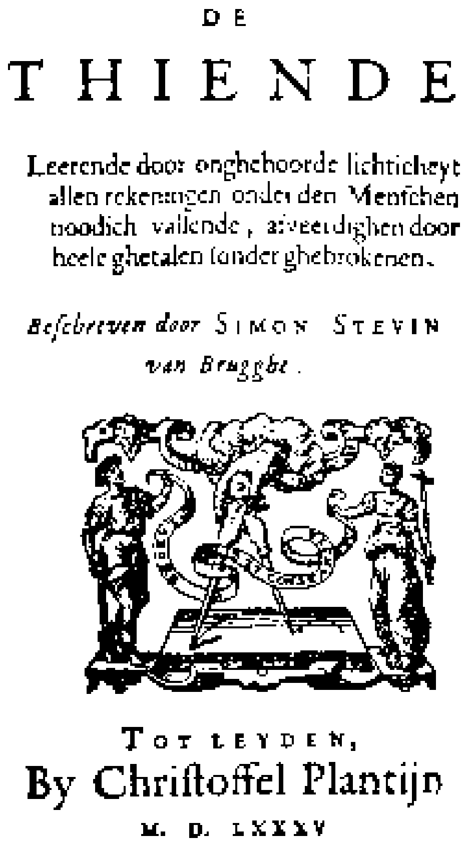


A
B
C
E
F
G
H
I
K
L
M
N
O
P
Q
S

Stevin’s Contributions

- Engineering
  - use of sluices for military purposes
  - wind-powered vehicle
- Physics
  - parallelogram of forces
    - representation of forces by vectors
  - hydrostatics
    - pressure is independent of shape
  - acceleration of falling bodies
- Music
  - $\sqrt[12]{2}$

The Tithe (1585)



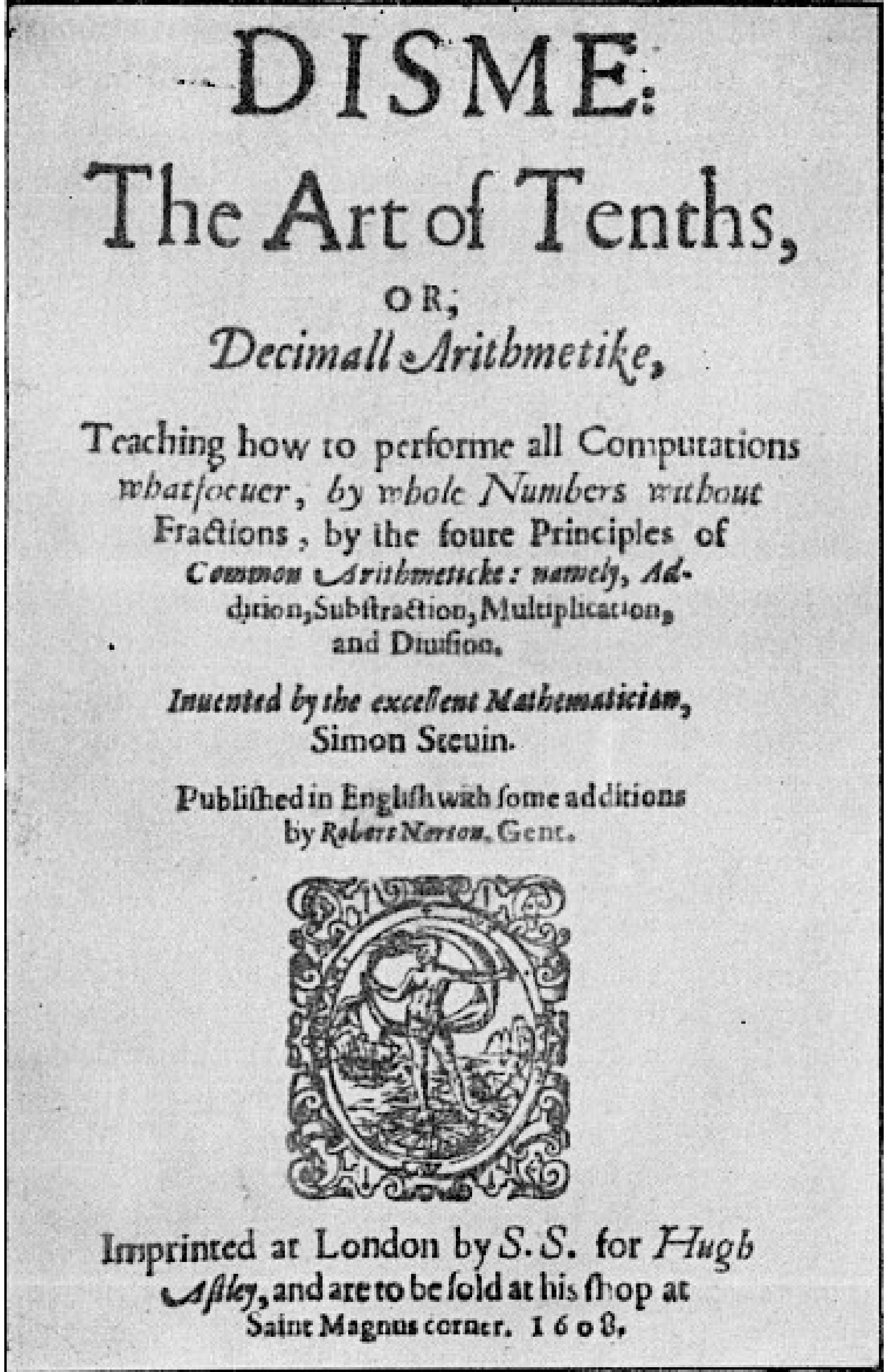
DE

THIENDE

Leerende door onghehoorde lichticheyt allen rekeningen onder den Menſchen noodich vallende, afveerdighen door heele ghetalen ſonder ghebrokenen.

Beſchreven door SIMON STEVIN *van Brugghe*.

TOT LEYDEN,
By Chriſtoffel Plantijn
M. D. LXXXV



DISME:

The Art of Tenths,

OR;

Decimall Arithmetike,

Teaching how to performe all Computations *whatſoeuer, by whole Numbers without* Fractions, by the foure Principles of *Common Arithmeticke: namely, Ad-*dition, Subſtraction, Multiplication, and Diuiſion.

Inuented by the excellent Mathematician, Simon Steuin.

Publiſhed in Engliſh with ſome additions by *Robert Norton*, Gent.

Imprinted at London by *S. S.* for *Hugh Aſley*, and are to be ſold at his ſhop at Saint Magnus corner. 1608.

Numbers measure everything

- *“Number* is that which expresseth the quantitie of each thing.”

  Simon Stevin

  Disme

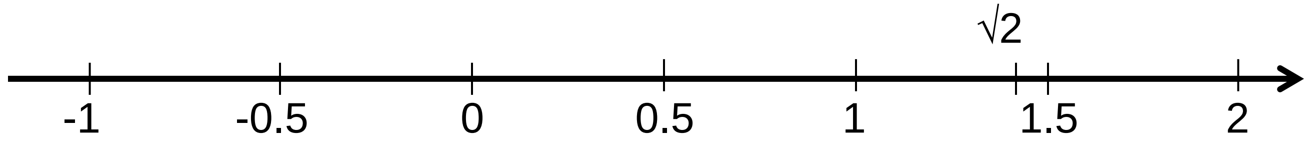
- “with one stroke, the classical restrictions of “numbers” to integers or to rational fractions was eliminated. His general notion of a real number was accepted by all later scientists.”

  B. L. van der Waerden

  History of Algebra

Stevin invented the *number line*

- negative
- irrational
- *inexplicable*…



Swept under the rug

- Replacement of finite with infinite representation
  - 1/7 vs. 0.142857142857142857142857…

Rene Descartes (1596 – 1650)

Cartesian Geometry

“Any problem in geometry can easily be reduced to such terms that a knowledge of lengths of certain strait lines is sufficient for its construction.”

Renatus Cartesius, *La Geometrie*

Polynomials (Stevin 1585)

$$4x^4 + 7x^3 - x^2 + 27x - 3$$

- Till Stevin it was an algorithm
  - take a number; raise it to 4th power; multiply it by 4 …
- Polynomial is a finite sequence of numbers:

  4, 7, -1, 27, -3

Intermediate Value Theorem

- *Appendice algébraique contenant règle générale de toutes équati*ons (1594)
- Stevin shows how the successive decimals of the root can be obtained.
- “in some cases the true value cannot be reached though one can obtain as many decimals of it as one may wish and come indefinitely near to it.”

Horner’s rule

$$4x^4 + 7x^3 - x^2 + 27x - 3 =$$
$$(((4x + 7)x - 1)x + 27)x - 3$$

Evaluating polynomials

```
template <InputIterator I, Semiring R>
R polynomial_value(I f, I l, R x) {
  if (f == l) return R(0);
  R sum(*f);
  while (++f != l) {
    sum = sum * x;
    sum = sum + *f;
  }
  return sum;
}
```

The value type of iterator

- The value type of the iterator (the type of the coefficients of the polynomial) does not have to be equal to the semiring R (the argument type of the polynomial and its result type).
  – e.g., polynomial with real coefficients can be evaluated with a matrix with real coefficients.
    - characteristic polynomial of a matrix can be applied to the matrix

Problem 96

What are the requirements on R and the value type of iterator? In other words, what are the requirements on coefficients of polynomials and on their values.

Operations on polynomials

- addition, subtraction
  - element by element

- multiplication:

$$
\begin{aligned}
c_0 &= a_0 b_0 \\
c_1 &= a_0 b_1 + a_1 b_0 \\
c_2 &= a_0 b_2 + a_1 b_1 + a_2 b_0 \\
&\vdots \\
c_k &= \sum_{k=i+j} a_i b_j
\end{aligned}
$$

Degree of polynomials

Degree of a polynomial is the index of the highest coefficient:

$$\begin{aligned} \deg(5) &= 0 \\ \deg(x+3) &= 1 \\ \deg(x^3+x-7) &= 3 \end{aligned}$$

Division with remainder

- Polynomial *a* is divisible by polynomial *b* with remainder if there are polynomials *q* and *r* such that

$$a = bq + r \quad \wedge \quad \deg(r) < \deg(b)$$

Problem 100

Prove that for any two polynomials *p(x)* and *q(x)*

1. $p(x) = q(x) \cdot (x - x_0) + r \implies p(x_0) = r$
2. $p(x_0) = 0 \implies p(x) = q(x) \cdot (x - x_0)$

Polynomial Division

$$
\begin{array}{rl}
 & 3x^2 + 2x - 2 \\
x-2 & \overline{)\,3x^3 - 4x^2 - 6x + 10} \\
 & \underline{3x^3 - 6x^2} \\
 & 2x^2 - 6x \\
 & \underline{2x^2 - 4x} \\
 & -2x + 10 \\
 & \underline{-2x + 4} \\
 & 6
\end{array}
$$

Stevin's *gcd* algorithm (1585)

```
polynomial<real> gcd(polynomial<real> m,
                     polynomial<real> n) {
   while (n != polynomial<real>(0)) {
     polynomial<real> t = remainder(m, n);
     m = n;
     n = t;
   }
   return m;
}
```

Proof of correctness

gcd is preserved at every step and the number of steps is finite:

1. $a = qb + r \implies \gcd(a, b) = \gcd(b, r)$
2. $\deg(r) < \deg(\mathrm{b})$

Problem 104 (Chrystal, *Algebra*)

Find *gcd* of:

1. $16x^4 - 56x^3 - 88x^2 + 278x + 105,$

   $16x^4 - 64x^3 - 44x^2 + 232x + 70$

2. $7x^4 + 6x^3 - 8x^2 - 6x + 1,$

   $11x^4 + 15x^3 - 2x^2 - 5x + 1$

3. $nx^{n+1} - (n+1)x^n + 1,$

   $x^n - nx + (n-1)$

Heirs of Pythagoras

Lecture 3

Die heilige deutsche Kunst

- Music
  - Bach, Händel, Haydn, Mozart, Beethoven, Schubert, Schumann, Brahms, Wagner, Mahler, Richard Strauss
- Literature
  - Goethe, Schiller
- Philosophy
  - Leibniz, Kant, Hegel, Schopenhauer, Marx, Nietzsche

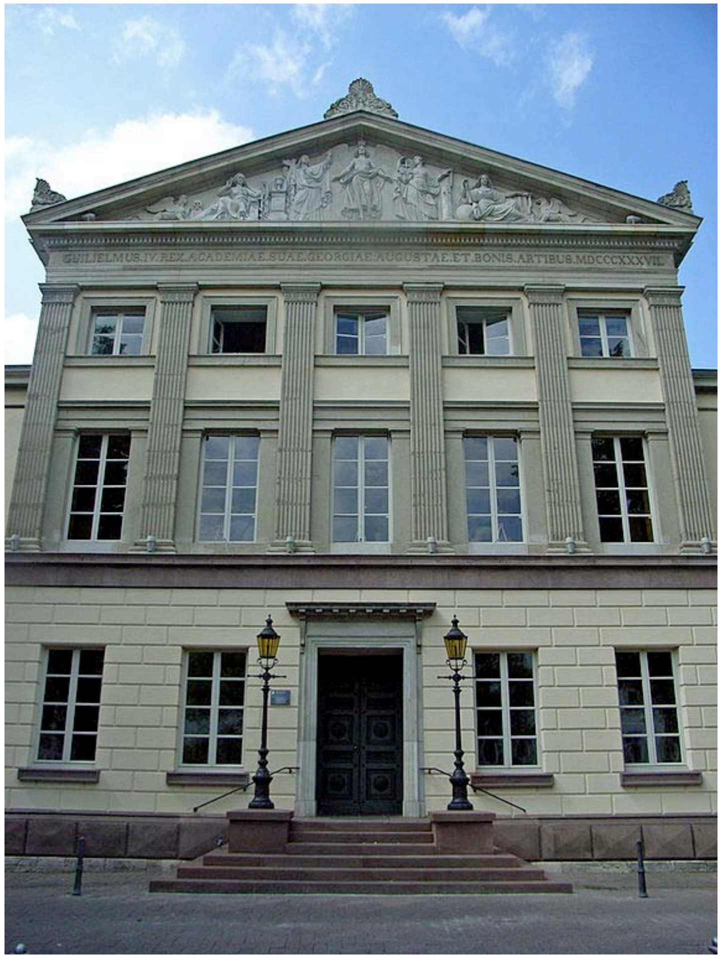
German Professor

- Commitment to the truth
- A scientist as a professional civil servant
- Collegial spirit
- Professor-student bond
- Continuity of research

Wir müssen wissen — wir werden wissen!

We must know — we will know!

University of Göttingen (1734)


GULIELMUS.IV.REX.ACADEMIAE.SUAE.GEORGIAE.AUGUSTAE.ET.BONIS.ARTIBUS.MDCCCXXXVII

- Gauss, Riemann, Dirichlet, Dedekind, Klein, Hilbert, Minkowski, Courant, Noether
- Max Born, Heisenberg, Oppenheimer

Carl Friedrich Gauss

CARL FRIEDRICH GAUSS
1777
1855
DDR
20

Regular heptodecagon

Princeps mathematicorum

- Fundamental Theorem of Algebra
- Number Theory
- (Non-Euclidean geometry)
- Normal distribution and method of least squares
- Celestial Mechanics
- Differential Geometry
- Electromagnetism

Disquisitiones Arithmeticae (1801)

- Fundamental theorem of arithmetic
- Modular arithmetic
- Quadratic reciprocity law
- Theory of integral quadratic forms
- Cyclotomic equations and construction of regular polygons

Gauss gcd “algorithm”

Given many numbers *A*, *B*, *C*, etc. the *greatest common divisor* is found as follows. Let all the numbers be resolved into their prime factors, and from these extract the ones which are common to *A*, *B*, *C*, etc…

Moreover, we know from elementary consideration how to solve these problems when the resolution of the numbers *A*, *B*, *C*, etc. into factors is not given.

Disquisitiones Arithmeticae, art. 18

Complex Numbers

- Imaginary numbers (*xi* where $i^2 = -1$) were used since XVI century, but never acquired legitimacy.
- In 1831 Gauss introduced *complex numbers* $z = a + bi$ as points (a,b) on Cartesian plane.
- In the same paper he introduced complex integers (*Gaussian integers*) which became a major tool in number theory.

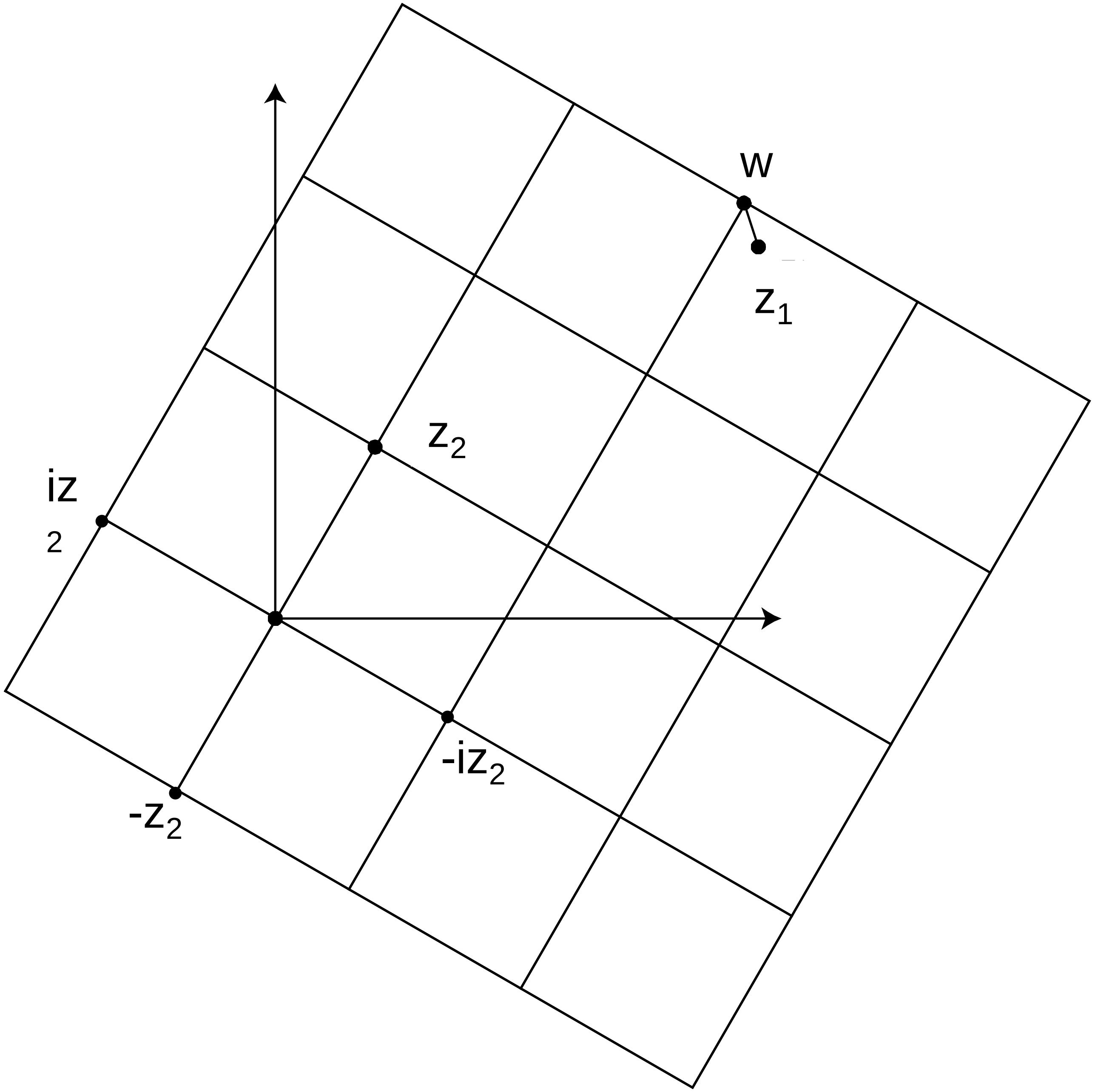
complex number:	$z = x + yi$
complex conjugate:	$\bar{z} = x - yi$
real part:	$\text{Re}(z) = \frac{1}{2}(z + \bar{z}) = x$
imaginary part:	$\text{Im}(z) = \frac{1}{2i}(z - \bar{z}) = y$
norm:	$z\bar{z} = \|z\| = x^2 + y^2$
absolute value:	$\lvert z \rvert = \sqrt{\|z\|} = \sqrt{x^2 + y^2}$
argument:	$\arg(z) = \arccos(\frac{x}{\lvert z \rvert})$
addition:	$z_1 + z_2 = (x_1 + x_2) + (y_1 + y_2)i$
subtraction:	$z_1 - z_2 = (x_1 - x_2) + (y_1 - y_2)i$
multiplication:	$z_1 z_2 = (x_1 x_2 - y_1 y_2) + (x_2 y_1 + x_1 y_2)i$
reciprocal:	$\frac{1}{z} = \frac{\bar{z}}{\|z\|} = \frac{x}{x^2+y^2} - \frac{y}{x^2+y^2}i$

Gaussian Integers

- Complex numbers of the form $a + bi$ where a and b are integers.

Gaussian remainder z_1 % z_2

- Construct a grid on complex plane generated by z_2, iz_2, $-iz_2$ and $–z_2$
- Find a square in the grid containing z_1
- Find a vertex w of the square closest to z_1
- $z_1 – w$ is the remainder


w
z_1
z_2
iz_2
$-iz_2$
$-z_2$

Gaussian integer *gcd*

```
complex<integer> gcd(complex<integer> m,
                     complex<integer> n) {
   while (real(n) != 0 || imag(n) != 0) {
     complex<integer> t = m % n;
     m = n;
     n = t;
   }
   return m;
}
```

Use of Gaussian Integers

The introduction of Gaussian integers lead to extraordinary simplification of several difficult proofs in number theory.

Johann Peter Gustav Lejeune Dirichlet (1805-1859)

Dirichlet Life

- Proof of Last Fermat Theorem for $n = 5$
- Slept with *Disquisitionae Arithmeticae* under his pillow
- In 1855 succeeded Gauss in Gottingen

Primes in arithmetic progressions

If $gcd(a, b) = 1$ then there are infinitely many primes of the form $an + b$

Vorlesungen über Zahlentheorie (Lectures on Number Theory)

…the whole structure of number theory rests on a single foundation, namely the algorithm for finding the greatest common divisor of two numbers.

All the subsequent theorems … are still only simple consequences of the result of this initial investigation…

…any analogous theory, for which there is a similar algorithm for the greatest common divisor, must also have consequences analogous to those in our theory. In fact, such theories exist.

Lectures on Number Theory

If one considers all numbers of the form:

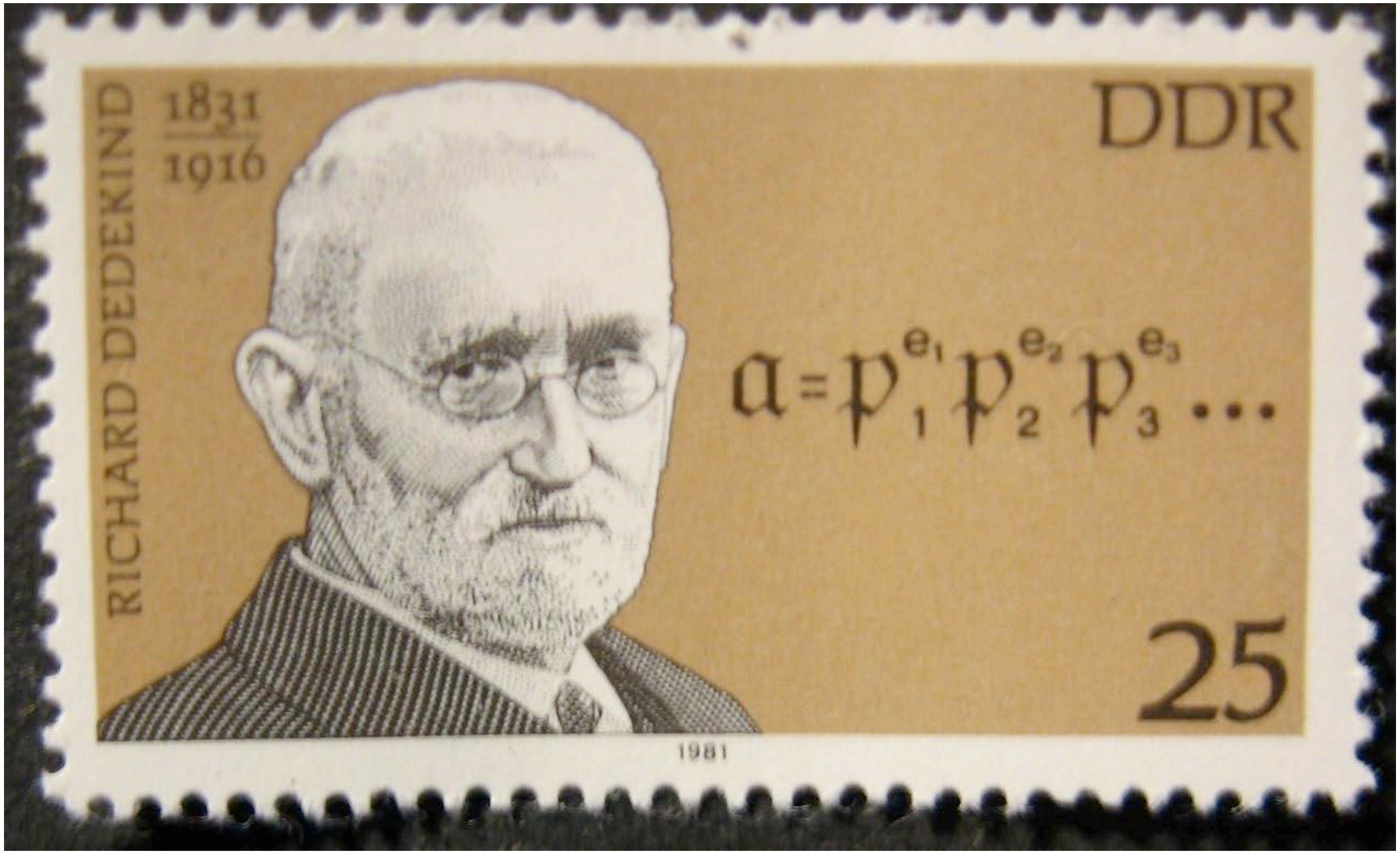
$$t + n\sqrt{-a}$$

…it is only for certain values of a, e.g. $a = 1$, that the greatest common divisor of two numbers could be found by an algorithm like the one for … integers…

However, it is otherwise when $a = 5$:

$$21 = 3 \cdot 7 = (1 + 2\sqrt{-5}) \cdot (1 - 2\sqrt{-5})$$

Richard Dedekind (1831 – 1916)


RICHARD DEDEKIND
1831
1916
DDR
$\mathfrak{a} = \mathfrak{p}_1^{e_1} \mathfrak{p}_2^{e_2} \mathfrak{p}_3^{e_3} \ldots$
25
1981

Dedekind’s Life

- Last student of Gauss
- Collaborator of Dirichlet
- TU Braunschweig
- Ring Theory
- Algebraic Integers
- Foundations of real numbers
  - also integers and rationals

Algebraic integers

Algebraic integers are linear integral combinations of roots of a *monic* polynomial with integer coefficients:

$x^2 + 1$ generates Gaussian integers $ai + b$

$x^3 - 1$ generates *Eisenstein integers* $a + b\frac{-1+i\sqrt{3}}{2}$

$x^2 + 5$ generates integers $\mathbb{Z}[\sqrt{-5}]$

The Promise of Algebraic Integers

- If Euclidean algorithm can be used with certain kind of algebraic integers (cyclotomic integers) that would lead to the proof of the Last Fermat Theorem.
- Unfortunately, it is not so. Algebraic integers generated by the polynomial

$$x^{23} - 1 = 0$$

do not work with the Euclidean algorithm.

An (almost) abstract algebra

- Dedekind work on algebraic integers contained all the fundamental concepts of modern abstract algebra.
- Es steht alles schon bei Dedekind!
  - It is all already in Dedekind – *Emmy Noether*

Emmy Noether (1882 -1935)

Life of Noether

- Gottingen
  - Invariant Theory
  - Noether’s theorem
  - Algebra
- Moscow
- Bryn Mawr
  - not Princeton "men's university, where nothing female is admitted"

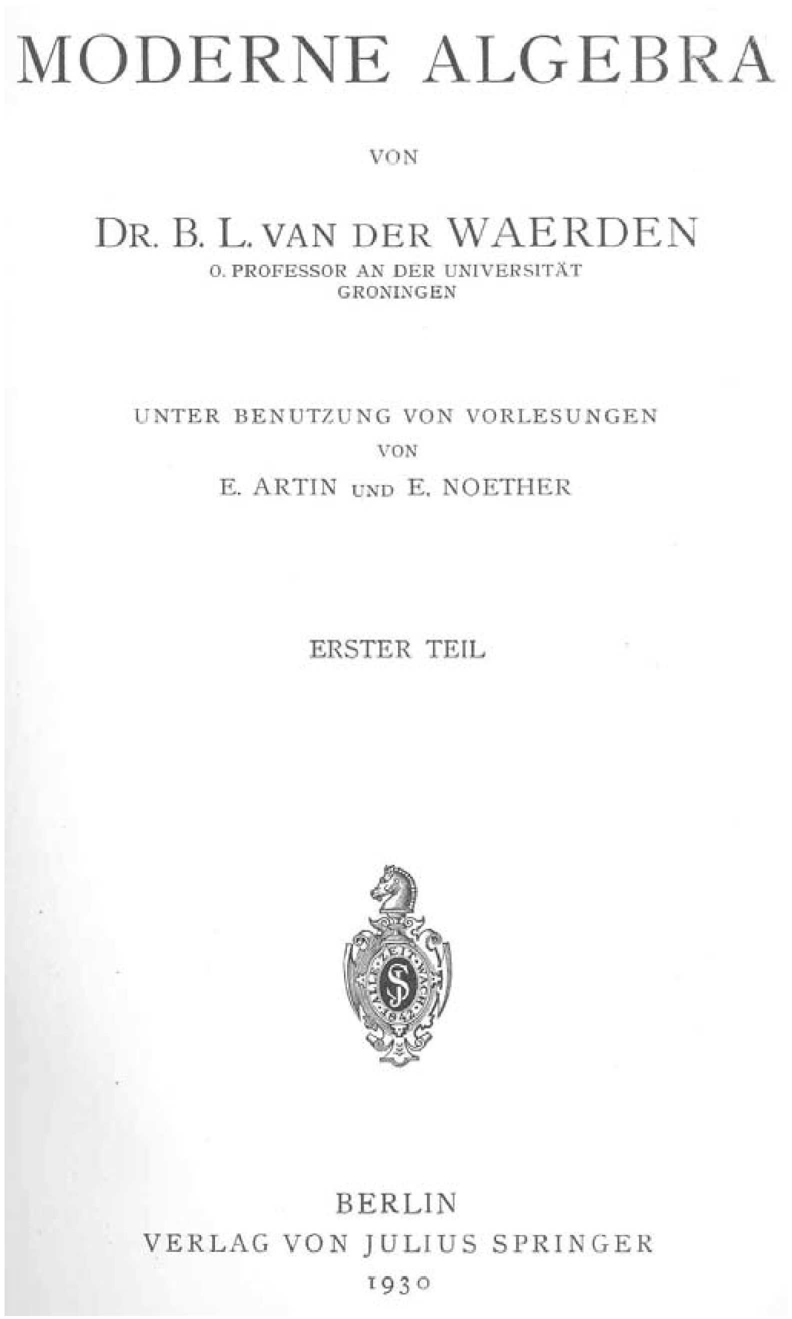
It was she who taught us to think in terms of simple and general algebraic *concepts* – homomorphic mappings, groups and rings with operators, ideals…

P.S. Alexandrov

For Emmy Noether, relationships among numbers, functions, and operations became transparent, amenable to generalisation, and productive only after they have been dissociated from any particular objects and have been reduced to general conceptual relationships…

B.L. van der Waerden

Bartel Leendert van der Waerden
(1903 - 1996)



MODERNE ALGEBRA

VON

DR. B. L. VAN DER WAERDEN
O. PROFESSOR AN DER UNIVERSITÄT
GRONINGEN

UNTER BENUTZUNG VON VORLESUNGEN
VON
E. ARTIN UND E. NOETHER

ERSTER TEIL

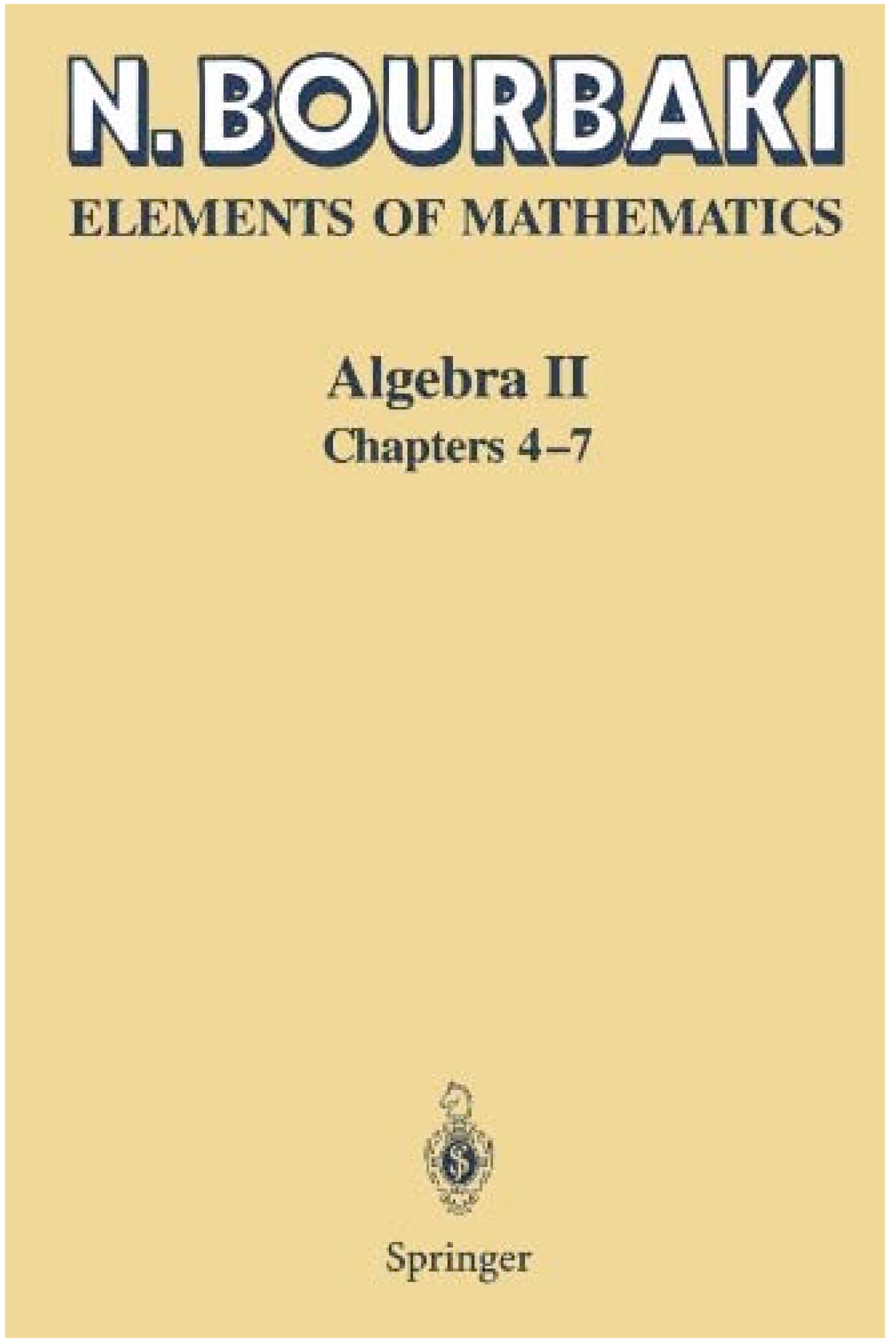
BERLIN
VERLAG VON JULIUS SPRINGER
1930

Noether gcd algorithm

```
template <EuclideanDomain R>
R gcd(R m, R n) {
   while (n != R(0)) {
     R t = m % n;
     m = n;
     n = t;
   }
   return m;
}
```

Nicolas Bourbaki

$\emptyset$



N. BOURBAKI

ELEMENTS OF MATHEMATICS

Algebra II

Chapters 4–7

Springer

Euclid and Göttingen

- Gauss: Non-Euclidean geometry
- Dirichlet: Infinity of primes in arithmetic progression
- Riemann: Non-Euclidean geometry
- Dedekind: Real numbers
- Klein: Erlanger program
- Hilbert: Foundations of Geometry, Mechanization of mathematics
- Minkowski: Space-time

Group

operations: $x \circ y, \quad x^{-1}$

constant: e (identity)

axioms:

$$x \circ (y \circ z) = (x \circ y) \circ z$$

$$x \circ e = e \circ x = x$$

$$x \circ x^{-1} = x^{-1} \circ x = e$$

Notation for group operation

- Mathematicians use circle or star or dot formally.
  - They often replace *x*y* with *xy*.
- They use + for Abelian (commutative) groups.

All groups are *transformation groups*

- Every element *a* of the group *G* defines a transformation of *G* onto itself:

  $$x \longrightarrow ax$$

- This transformation is one-to-one because of invertibility:

  $$y \longrightarrow a^{-1}x$$

Group transformations are one-to-one

For any finite set *S* of elements of group *G* and any element *a* of *G*, a set of elements *aS* has the same number of elements as *S*:

Proof:

$S = \{s_1, \ldots, s_n\}$ then $aS = \{as_1, \ldots, as_n\}$.
Suppose $as_i = as_j$; then $a^{-1}(as_i) = a^{-1}(as_j)$.
By associativity, $(a^{-1}a)s_i = (a^{-1}a)s_j$;
by cancellation, $es_i = es_j$ and by identity, $s_i = s_j$.

Examples of groups

- Additive group of integers
  - there is no multiplicative group of integers
- Multiplicative group of remainders modulo 7
- Permutation group of 52 cards
- Multiplicative group of invertible matrices with real coefficients
- Group of rotations of the plane

Kinds of group

- Abelian groups
- Finite groups
- Finite cyclic group

$$ab = e \implies b = a^{-1}$$

$$ab = e$$

$$a^{-1}(ab) = a^{-1}e$$

$$(a^{-1}a)b = a^{-1}$$

$$eb = a^{-1}$$

$$b = a^{-1}$$

$$(ab)^{-1} = b^{-1}a^{-1}$$

$$(ab)(b^{-1}a^{-1}) = a(bb^{-1})a^{-1} = aa^{-1} = e$$

$$(x^{-1})^n = (x^n)^{-1}$$

By induction :

Case 1 : $(x^{-1})^1 = x^{-1} = (x^1)^{-1}$

Case n : Assume that $(x^{-1})^{n-1} = (x^{n-1})^{-1}$ then

$x^n(x^{-1})^n = (x^{n-1}x)(x^{-1}(x^{-1})^{n-1}) = e$ therefore

$(x^n)^{-1} = (x^{-1})^n$

Problem 148 (very easy)

Prove that any group has at least one element.

Order of a group

If a group has *n* > 0 elements, *n* is is called the group's *order*.

If n has infinitely many elements, it is of *infinite* order.

Order of an element

An element a has an order $n > 0$ if $a^n = e$ and for any $0 < k < n$, $a^k \neq e$.

If such n does not exist, a has an infinite order.

Problem 151 (very easy)

- What is the order of *e*?

- Prove that *e* is the only element of such order.

Every element of a finite group has finite order

If *n* is an order of the group, then for any element *a*, $\{a, a^2, a^3, \ldots, a^{n+1}\}$ has at least one repetition a^i and a^j. Let us assume that $i < j$ and a^i is the first repeated element. Then

$$a^j = a^i$$

$$a^j a^{-i} = a^i a^{-i} = e$$

$$a^{j-i} = e$$

And $j - i > 0$ is the order of *a*.

Problem 153

Prove that if *a* is an element of order *n*, then

$$a^{-1} = a^{n-1}$$

Subgroups

- A subgroup is a group that is a subset of another group.
- Examples
  - the additive group of even numbers are a subgroup of the additive group of integers.

Multiplication modulo 7 with inverses

$$
\begin{array}{lll}
1 \times i: & \mathbf{1},2,3,4,5,6 & 1 \\
2 \times i: & 2,4,6,\mathbf{1},3,5 & 4 \\
3 \times i: & 3,6,2,5,\mathbf{1},4 & 5 \\
4 \times i: & 4,\mathbf{1},5,2,6,3 & 2 \\
5 \times i: & 5,3,\mathbf{1},6,4,2 & 3 \\
6 \times i: & 6,5,4,3,2,\mathbf{1} & 6
\end{array}
$$

Subgroups of $\mathbb{Z}_7$

$\{1\}$ is a multiplicative subgroup of $\mathbb{Z}_7$

$\{1, 6\}$ is a multiplicative subgroup of $\mathbb{Z}_7$

$\{1, 2, 4\}$ is a multiplicative subgroup of $\mathbb{Z}_7$

$\{1, 2, 3, 4, 5, 6\}$ is a multiplicative subgroup of $\mathbb{Z}_7$

Problem 157 (easy)

- Find orders of every element of multiplicative group of remainders mod 7.

- Find orders of every element of multiplicative group of remainders mod 11.

Cyclic group

A finite group is called *cyclic* if it has an element *a* such that for any element *b* there is an integer *n* where

$$b = a^n$$

Cyclic group of an element

Powers of a given element in a finite group form a subgroup.

- Powers contain e.
- Powers contain inverse.

Example of a cyclic group

Additive group of remainders modulo *n*.

Problem 161

Prove that any subgroup of a cyclic group is cyclic.

Problem 162 (easy)

Prove that a cyclic group is Abelian.

Cosets

If G is a group, $H \subset G$ its subgroup,
then for any $a \in G$ the *(left) coset* of a by H is a set

$$aH = \{g \in G \mid \exists h \in H \ : \ g = ah\}$$

Examples of cosets

Consider additive group of integers Z and its subgroup, integers divisible by 4, $4Z$. It has 4 distinct cosets

- 4*n*
- 4*n* + 1
- 4*n* + 2
- 4*n* + 3

Size of cosets

The number of elements in a coset aH is the same as the number of elements in the subgroup H.

Proof:

Follows from one-to-one property of the transformation ax.

Complete coverage by cosets

Every element *a* of group *G* belongs to some coset of subgroup *H*.

Proof:

$$a = ae \implies a \in aH$$

Cosets are either disjoint or identical

If two cosets *aH* and *bH* in a group *G* have a common element *c*, then *aH* = *bH*.

Proof:

$$c = ah_a \ \wedge \ c = bh_b$$

$$ah_a = bh_b$$

$$a = bh_b h_a^{-1}$$

$$\forall x \in H, ax = bh_b h_a^{-1} x \in bH$$

$$aH \subseteq bH$$

Similarly, $bH \subseteq aH$, and so $bH = aH$

Lagrange's Theorem

The order of a subgroup *H* of a finite group *G* divides the order of the group.

Proof:

1. The group *G* is covered by cosets of *H*.
2. Different cosets are disjoint.
3. They are of the same size *n*.

The order of *G* is *nm* where *m* is the number of distinct cosets.

Corollary of Lagrange’s theorem

The order of any element in the finite group divides the order of the group.

Proof:

An order of an element is equal to the order of the cyclic group of its powers.

Second Corollary

G is a group of order n. If a is an element of G then $a^n = e$.

Proof:

If a has an order m, than m divides n,

and $n = qm$.

$a^m = e$, therefore $(a^m)^q = e$ and $a^n = e$.

New Proof of Little Fermat Theorem

Let us take a multiplicative group of remainders modulo *p*. Since *p - 1* is the order of the group it follows immediately from the second corollary of Lagrange’s Theorem that:

$$a^{p-1} = 1 \mod p$$

New Proof of Euler Theorem

Let us take a multiplicative group of coprime remainders modulo *n*. Since *φ(n)* is by definition the order of the group it follows immediately from the second corollary of Lagrange’s Theorem that:

$$\gcd(a,n) = 1 \implies a^{\phi(n)} = 1 \mod n$$

Problem 173 (very easy)

What are subgroups of a group of order 101?

Problem 174

Prove that every group of prime order is cyclic.

Heirs of Pythagoras

Lecture 4

Rings

operations: $x + y, -x, xy$
constant: $0_R, 1_R$
axioms:

$$x + (y + z) = (x + y) + z$$

$$x + 0 = 0 + x = x \qquad x + -x = -x + x = 0$$

$$x + y = y + x$$

$$x(yz) = (xy)z$$

$$1 \neq 0 \qquad 1x = x1 = x \qquad 0x = x0 = 0$$

$$x(y + z) = xy + xz \qquad (y + z)x = yx + zx$$

Examples of Rings

- Integers
- $n \times n$ matrices with real coefficients
- Gaussian integers
- Univariate polynomials with integer coefficients

Commutative Rings

A ring is commutative if $xy = yx$.

- non-commutative rings usually come from linear algebra
- rings of algebraic integers and polynomial rings commute
- Two “branches” of abstract algebra
  - Commutative algebra
  - Non-commutative algebra

We are dealing with commutative algebra.

Invertible Elements

An element *x* of a ring is called *invertible* if there is an element *x*$^{-1}$ such that

$$xx^{-1} = x^{-1}x = 1$$

An invertible element of a ring is called a *unit* of this ring.

Problem 180 (very easy)

What are units in the ring $\mathbb{Z}$ of integers?
What are units in the ring $\mathbb{Z}[\sqrt{-1}]$ of Gaussian integers?

A product of units

A product of units is a unit.

Proof:

If a is a unit and b is a unit then so is ab:

$$ab(b^{-1}a^{-1}) = 1$$

Multiplicative group of units

- Units are closed under multiplication.
- 1 is a unit.
- Inverse of a unit is a unit.

Integral Domain

An element *x* of a ring is called a *zero divisor* if

1. $x \neq 0$
2. $\exists y \neq 0, \;\; xy = 0$

A commutative ring is an *integral domain* if it has no zero divisors.

Examples of Integral Domain

- Integers
- Gaussian integers
- Polynomials over integers
- Rational functions over integers

$$\frac{x^2+1}{x^3-2}$$

Problem 185 (very easy)

Prove that a zero divisor is not a unit.

Euclidean domain (Euclidean ring)

E is a Euclidean domain if:

- E is an integral domain
- E has operations *quot* and *rem*
- E has a non-negative norm:

$$\begin{aligned}
&\|x\| : E \to \mathbb{N} \\
&\text{satisfying :} \\
&\quad \|a\| = 0 \iff a = 0 \\
&\quad b \neq 0 \implies \|ab\| \geq \|a\| \\
&\quad b \neq 0 \implies a = \mathrm{quot}(a,b) \cdot b + \mathrm{rem}(a,b) \\
&\quad \|\mathrm{rem}(a,b)\| < \|b\|
\end{aligned}$$

Euclidean Domain gcd

```
template <EuclideanDomain R>
R gcd(R m, R n) {
   while (n != R(0)) {
     R t = m % n;  // ||t|| < ||n||
     m = n;
     n = t;
   }
   return m;
}
```

Fields

An integral domain where every non-zero element is invertible is called a *field*.

Examples of fields

Rational numbers $\mathbb{Q}$
Real numbers $\mathbb{R}$
Prime remainder fields $\mathbb{Z}_p$
Complex numbers $\mathbb{C}$

Kinds of fields

- Finite fields
- Algebraically closed fields
- Non-commutative field of quaternions

$$\mathbb{R}[i,j,k]$$

$$i^2 = j^2 = k^2 = -1$$

$$ij = k,\ jk = i,\ ki = j$$

$$ji = -k,\ kj = -i,\ ik = -j$$

Josef (“Yossi”) Stein (1961)

“I was in the beginning of my PhD thesis, which included using the Racah Algebra. Using "Racah Algebra" meant doing calculations with numbers of the form a/b*sqrt(c), where, a, b, c were integers. I wrote a program for the only available computer in Israel at that time - The WEIZAC at the Weizmann institute. Addition time was 57 microseconds, division took about 900 microseconds. Shift took less than addition. …I had the right conditions for finding that algorithm. ***Fast GCD meant survival***.”

Delayed publication

“Yes, 1961 is the correct year…The reason why I did not publish the algorithm earlier was, that when I told my advisor (G. Racah himself) about my idea, he said: "Yesss” (in Hebrew) in his Italian accent. Somehow this did not sound like an encouragement for publication. In 1966, when I was looking for a post-doc I met another great physicist: John Blatt. He encouraged me to publish it.”

Josef Stein*, Computational problems associated with Racah algebra,* J. Comput. Phys., (1967) 1, 397-405

Stein’s Mathematics

$$\gcd(n, 0) = \gcd(0, n) = n$$

$$\gcd(n, n) = n$$
$$\gcd(2n, 2m) = 2 \cdot \gcd(n, m)$$
$$\gcd(2n, 2m + 1) = \gcd(n, 2m + 1)$$
$$\gcd(2n + 1, 2m) = \gcd(2n + 1, m)$$
$$\gcd(2n + 1, 2(n + k) + 1) = \gcd(2n + 1, k)$$
$$\gcd(2(n + k) + 1, 2n + 1) = \gcd(2n + 1, k)$$

196	42	
98	21	2
49	21	2
28	21	2
14	21	2
7	21	2
14	7	2
7	7	2

GCD: $7 \times 2 = 14$

Stein's algorithm (prologue)

```
template <BinaryInteger I>
I stein_gcd(I m, I n) {
  if (m < I(0)) m = -m;
  if (n < I(0)) n = -n;
  if (m == I(0)) return n;
  if (n == I(0)) return m;

  //      $m > 0 \wedge n > 0$
```

Stein’s algorithm (factoring out power of 2)

```
int d_m = 0;
while (even(m)) { m >>= 1; ++d_m;}

int d_n = 0;
while (even(n)) { n >>= 1; ++d_n;}

//
```

$$\operatorname{odd}(m) \wedge \operatorname{odd}(n)$$

Stein’s algorithm (the main loop)

```
while (m != n) {
  if (n > m) swap(n, m);
  m -= n;
  do m >>= 1; while (even(m));
}

//     $m = n$
```

Stein’s algorithm (epilogue)

```
  return m << min(d_m, d_n);
}
```

Is it just a hack?

- Is Stein’s algorithm only interesting because of slow hardware?

- Is it only useful for binary integers?

Every useful algorithm is based on some fundamental mathematical truth.

- The discoverer of the algorithm might not see this truth.
- There might be a long time between the first discovery of the algorithm and its understanding.
- Every useful program is a worthy object of study.

Generalization of Euclid’s gcd

- Integers
  - Greeks (V BC)
- Polynomials
  - Stevin (ca. 1600)
- Gaussian Integers
  - Gauss (ca. 1830)
- Algebraic Integers
  - Dirichlet, Dedekind (ca. 1860)
- Generic Version
  - Noether, van der Waerden (ca. 1930)

Stein for polynomials $\mathbb{R}[x]$ (1969)

Use *x* as 2 !

- $x^2 + x$ is “even”
- $x^2 + x + 1$ is “odd”
- $x^2 + x$ “shifts to” $x + 1$

Stein for polynomials over a field

$$\gcd(p, 0) = \gcd(0, p) = p$$
$$\gcd(p, p) = p$$

$$\gcd(xp, xq) = x \cdot \gcd(p, q)$$
$$\gcd(xp, xq + c) = \gcd(p, xq + c)$$
$$\gcd(xp + c, xq) = \gcd(xp + c, q)$$
$$\deg(p) \geq \deg(q) \implies \gcd(xp + c, xq + d) = \gcd(p - \frac{c}{d}q, xq + d)$$
$$\deg(p) < \deg(q) \implies \gcd(xp + c, xq + d) = \gcd(xp + c, q - \frac{d}{c}q)$$

n	m	operation
$x^3 - 3x - 2$	$x^2 - 4$	$n - (0.5x^2 - 2)$
$x^3 - 0.5x^2 - 3x$	$x^2 - 4$	shift(n)
$x^2 - 0.5x - 3$	$x^2 - 4$	$n - (0.75x^2 - 3)$
$0.25x^2 - 0.5x$	$x^2 - 4$	normalize(n)
$x^2 - 2\mathrm{x}$	$x^2 - 4$	shift(n)
$x - 2$	$x^2 - 4$	$m - (2x - 4)$
$x - 2$	$x^2 - 2x$	shift(m)
$x - 2$	$x - 2$	GCD: $x - 2$

Andre Weilert algorithm for Gaussian Integers (2000)

Use 1+*i* as 2 !

Andre Weilert, *(1+i)-ary GCD Computation in Z*[*i*] *as an Analogue of the Binary GCD Algorithm*, J. Symbolic Computation (2000) 30, 605-617

Division by 1+*i*

$$\frac{a+bi}{1+i} = \frac{(a+bi)(1-i)}{(1+i)(1-i)} = \frac{(a+b)-(a-b)i}{2}$$

A Gaussian integer *a*+*bi* is divisible by 1+*i* if and only if *a*=*b* (mod 2)

Remainder Cancellation

If two Gaussian integers z_1 and z_2 are not divisible by $1+i$ then z_1+z_2 is divisible by $1+i$. Then z_1-z_2, z_1+iz_2 and z_1-iz_2 are also divisible by $1+i$.

And,

$$\min(|z_1+z_2|, |z_1-z_2|, |z_1+iz_2|, |z_1-iz_2|) < \max(|z_1|, |z_2|)$$

Further extension of Stein (2003)

Ivan Bjerre Damgård and Gudmund Skovbjerg Frandsen, *Efficient algorithms for GCD and cubic residuosity in the ring of Eisenstein integers,* Proceedings of the 14th International Symposium on Fundamentals of Computation Theory, Lecture Notes in Computer Science 2751, Springer-Verlag (2003), 109-117

Damgård and Frandsen Algorithm

- Stein algorithm works for Eisenstein integers $\mathbb{Z}[\zeta]$, i.e. the integers extended with a complex primitive cubic root of 1

$$\zeta = \frac{-1+\sqrt{-3}}{2}$$

- In Stein, we use $1-\zeta$ instead of 2.

Stein works where Euclid does not!

In 2004 Agarwal and Frandsen demonstrated that there is a ring that is not a Euclidean domain (ring of integers in $\mathbb{Q}(\sqrt{-19})$) where Stein algorithm works.

Saurabh Agarwal, Gudmund Skovbjerg Frandsen: *Binary GCD Like Algorithms for Some Complex Quadratic Rings*. ANTS 2004: 57-71

What is *Stein domain*?

Fundamental operations on Stein

- is_unit(u)

  there is a *v* such that *vu = 1*
- are_associates(m,n)

  m = *un* where *u* is a unit
- is_smallest_prime(p)

$$q \neq 0 \ \wedge \ \|q\| < \|p\| \implies q \text{ is a unit}$$

Three lemmas for Stein

- If a Euclidean ring is not a field it has a smallest prime.
- u is a unit $\implies \|a\| = \|ua\|$
- p is a smallest prime $\implies \text{rem}(q, p)$ is either a unit or 0

Conjecture

Every Euclidean domain is a Stein domain

Generalized Stein's algorithm (prologue)

```
template <SteinDomain R>
R stein_gcd(R m, R n) {

  if (m == R(0)) return n;
  if (n == R(0)) return m;
```

Generalized Stein's algorithm (factoring out power of smallest prime)

```
int d_m = 0;
while (divisible_by_smallest_prime(m)) {
   m = divide_by_smallest_prime(m);
   ++d_m;
}
int d_n = 0;
while (divisible_by_smallest_prime(n)) {
   n = divide_by_smallest_prime(n);
   ++d_n;
}
```

Generalized Stein's algorithm (the main loop)

```
while (!is_associate(m, n)) {
  if (norm(n) > norm(m)) swap(n, m);
  m = reduce_associate_remainders(n, m);
  do {
    m = divide_by_smallest_prime(m);
  } while (divisible_by_smallest_prime(m));
}
```

Generalized Stein’s algorithm (epilogue)

```
    R p = smallest_prime<R>();
   return m * power(p, min(d_m, d_n));
}
```

Lessons of Stein

- Even a classical problem studied by great mathematicians may have a new solution.
- Performance constraints are good for creativity.
- Behind every optimization there is solid mathematics.

Project

Compare the performance of the Stein and Euclid algorithms on random integers from ranges $[0, 2^{16})$, $[0, 2^{32})$, and $[0, 2^{64})$.

Heirs of Pythagoras

Lecture 5

Bézout's identity

$$\forall a, b \; \exists x, y \; : \; xa + yb = \gcd(a, b)$$

Claude Gaspard Bachet de Méziriac (1581 – 1638)

Bachet’s work

- Diophantus, *Arithmetic* (1621)
  - Bachet’s Diophantus immortalized by Fermat

- *Problèmes Plaisants* (1612, 1624)
  - first book on recreational mathematics
    - the modern equivalent is *Mathematical Recreations and Essays* by W. W. Rouse Ball

Ideals

An *ideal I* is a non-empty subset of a ring *R* such that

$$1.\ \forall x, y \in I \ : \ x + y \in I$$
$$2.\ \forall x \in I\ \forall a \in R \ : \ ax \in I$$

Examples of Ideals

- Even numbers
- Univariate polynomials with root 5
- Polynomials with x and y with free coefficient 0

  $x^2 + 3y^2 + xy + x$

Subrings that are not ideals

1. $\mathbb{Z}$ is a subring of $\mathbb{Z}[\sqrt{-1}]$
2. $\mathbb{Z}[x]$ is a subring of $\mathbb{Z}[x, y]$

Problem 228

1. Prove that an ideal *I* is closed under subtraction:

$$\forall x, y \in I \ : \ x + y \in I$$

2. Prove that *I* contains 0.

Linear combination ideal

In a ring, for any two elements *a* and *b*, the set of all elements $\{xa + yb\}$ forms an ideal.

Proof:

closed under addition:

$(x_1a + y_1b) + (x_2a + y_2b) = (x_1 + x_2)a + (y_1 + y_2)b$

closed under multiplication by an arbitrary element:

$z(xa + yb) = (zx)a + (zy)b$

Problem 230

Prove that all the elements of a linear combination ideal are divisible by any of the common divisors of *a* and *b*.

Ideals in Euclidean Domains

Any ideal in an Euclidean domain is closed under remainder operation and under Euclidean gcd.

Proof:

1. Closed under remainder:

   rem(a, b) = a – quot(a, b)$\cdot b$

2. Closed under gcd :

   Immediately follows from 1.

Principal Ideals

An ideal I of the ring R is called a *principal ideal* if there is an element $a \in R$ called the *principal element* such that

$$x \in I \iff \exists y \in R \;:\; x = ay$$

Examples of Principal Ideals

- Even numbers

- Polynomials with root 5

An ideal that is not principal

- Polynomials with x and y with free coefficient 0

  $x^2 + 3y^2 + xy + x$

  Couldn't be generated by x – would not contain y; and vice versa.

Problem 235

Prove that any element in a principal ideal is divisible by the principal element.

Principal Ideal Domain

An integral domain is called a *principal ideal domain* (*PID*) if every ideal in it is a principal ideal.

$ED \implies PID$

Every Euclidean domain is a principal ideal domain.

Proof:
Any ideal I contains an element m with a minimal positive norm. Then $\forall a \in I \ : \ a = qm$.
Otherwise, $a = qm + r$ where $0 < \|r\| < \|m\|$.
Contradiction. Therefore m is the principal element of I and I is a principal ideal.

Bachet theorem

A linear combination ideal $I = \{xa + yb\}$ of a Euclidean domain contains $\gcd(a, b)$.

Proof:

$$a \in I \ \wedge \ b \in I \implies \gcd(a,b) \in I$$

Invertibility lemma

$$\forall a, n \in \mathbb{Z} \ : \ \gcd(a, n) = 1 \implies \\ \exists b \in \mathbb{Z}_n \ : \ ab = 1 \mod n$$

Proof:

By Bachet theorem,

$$\exists b, d \in \mathbb{Z} \ : \ ba + dn = 1$$

Therefore, $ab = 1 - dn$ and

$$ab = 1 \mod n$$

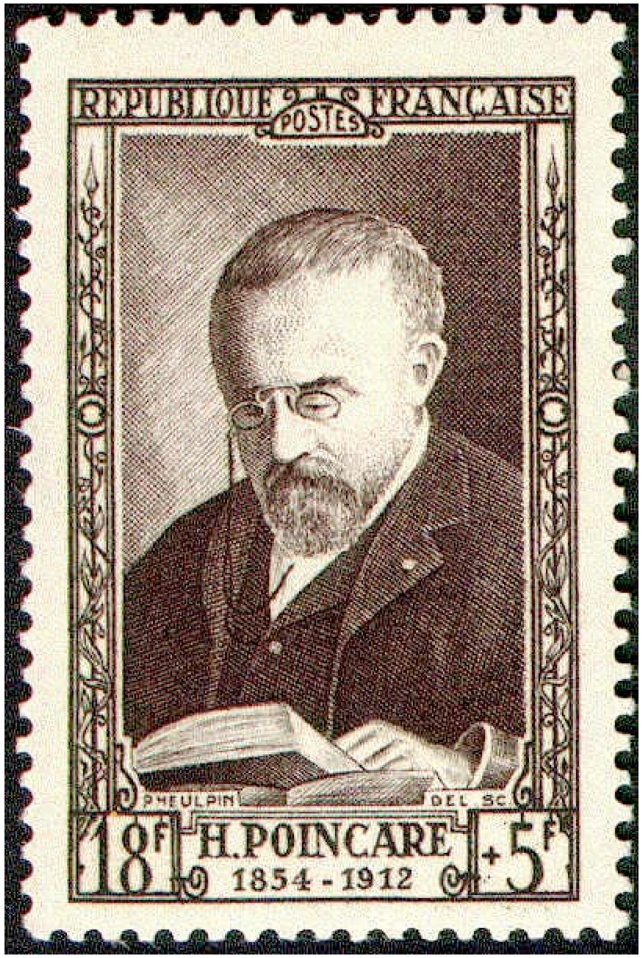
Problem 240

Using Bachet’s theorem prove that if p is prime, then any $0 < a < p$ has a multiplicative inverse modulo p.

Non-constructive proofs

- We know that inverses exist, but have no idea how to find them.
- This situation is unsatisfactory.
- The philosophy of mathematics that rejects non-constructive proofs is called *constructivism*.
  - *Intuitionism* is historically most important variety of constructivism.

Henri Poincaré



Poincaré's Contributions

- Almost every branch of mathematics
  - originating several (algebraic topology…)
- Special relativity theory
- Criticism of set theory and the formalist (Hilbert) agenda

A major tragedy of XX century science was the rejection of Poincare's legacy by Bourbaki and Bourbaki's epigones.

Poincaré’s Books

Poincaré was a brilliant writer. In 1909 he was elected a member of *Académie française*. His books on philosophy of science are very important

- *Science and Hypothesis*
- *Value of Science*
- *Science and Method*

Poincaré on Science

“Science has had marvelous applications, but a science that would only have applications in mind would not be science anymore, it would be only cookery.”

Bachet’s algorithm

There is a constructive way to find x and y such that

$$xa + yb = \gcd(a, b)$$

Trace of Euclidean Algorithm

$$a = b \cdot q_1 + r_1$$

$$b = r_1 \cdot q_2 + r_2$$

$$\cdots$$

$$r_{n-2} = r_{n-1} \cdot q_n + r_n$$

$$r_{n-1} = r_n \cdot q_{n+1}$$

Remainder trace

$$r_1 = a - b \cdot q_1$$

$$r_2 = b - r_1 \cdot q_2$$

$$\cdots$$

$$r_{i+2} = r_i - r_{i+1} \cdot q_{i+2}$$

$$\cdots$$

First steps of Bachet’s algorithm

$$\begin{aligned} a &= 1 \cdot a + 0 \cdot b \\ b &= 0 \cdot a + 1 \cdot b \\ r_1 &= 1 \cdot a - q_1 \cdot b \\ r_2 &= b - (a - b \cdot q_1) \cdot q_2 \\ &= -q_2 \cdot a + (1 + q_1 \cdot q_2) \cdot b \end{aligned}$$

Iterative recurrence for Bachet

Assume that

$$\begin{aligned} r_i &= x_i \cdot a + y_i \cdot b \\ r_{i+1} &= x_{i+1} \cdot a + y_{i+1} \cdot b \end{aligned}$$

Then

$$\begin{aligned} r_{i+2} &= r_i - r_{i+1} \cdot q_{i+2} \\ &= x_i \cdot a + y_i \cdot b - (x_{i+1} \cdot a + y_{i+1} \cdot b) \cdot q_{i+2} \\ &= (x_i - x_{i+1} \cdot q_{i+2}) \cdot a + (y_i - y_{i+1} \cdot q_{i+2}) \cdot b \end{aligned}$$

In other words

$$\begin{aligned} x_{i+2} &= x_i - x_{i+1} \cdot q_{i+2} \\ y_{i+2} &= y_i - y_{i+1} \cdot q_{i+2} \end{aligned}$$

We do not need y!

If $b \neq 0$ we can compute x and gcd and get y using the formula

$$y = \mathrm{quot}(\gcd(a, b) - ax, b)$$

Problem 252

What are x and y if $b = 0$?

Extended gcd

```
template <EuclideanDomain R>
pair<R, R> extended_gcd(R a, R b) {
  R x0(1);
  R x1(0);
  while (b != R(0)) {
    // compute new r and x
    pair<R, R> p = quotient_remainder(a, b);
    R x2 = x1 – p.first * x0;
    // shift r and x
    x0 = x1; x1 = x2;
    a = b; b = p.second;
  }
  return make_pair(x0, a);
}
```

Project 254

Develop a version of extended gcd based on Stein’s algorithm.

Applications of gcd

- Cryptography
- Rational arithmetic
- Symbolic integration
- std::rotate

Permutations

Permutation is a function from a sequence of *n* objects onto itself.

Notation: $$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 1 & 3 \end{pmatrix}$$

Shorthand: $\begin{pmatrix} 2 & 4 & 1 & 3 \end{pmatrix}$

Example: (2 4 1 3): {*a*, *b*, *c*, *d*} = {*c*, *a*, *d*, *b*}

Symmetric Groups

- A set of all permutations on *n* elements constitutes a group called the symmetric group S_n.
  - group operation: composition
    - function composition is associative
  - identity element: identity permutation
  - inverse: inverse permutation

Problem 258

What is the order of S_n?

Transpositions

A *transposition* (*i j*) is a permutation that exchanges the *i*th and *j*th elements (*i* ≠ *j*) leaving the rest in place.

(2 3): {*a*, *b*, *c*, *d*} = {*a*, *c*, *b*, *d*}

(In C++, we call it swap.)

Transposition Lemma

Any permutation is a product of transpositions.

Proof:

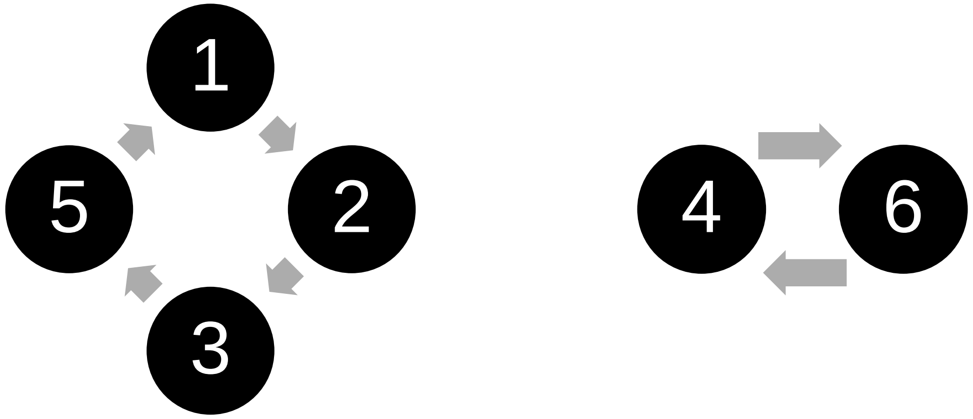
One transposition can put one element into its final destination. Therefore, $n - 1$ transpositions will put all elements into their final destinations.

Problem 261

Prove that if $n > 2$, S_n is not Abelian.

Cycles in the permutations

Every permutation can be decomposed into cycles. For example, (2 3 5 6 1 4) contains two cycles:



It is written (2 3 5 6 1 4) = (1 2 3 5)(4 6)

Cycles are disjoint

If you are at a position in a cycle, you can get to all other positions in that cycle. Therefore, if two cycles share a position, they share all the positions.

Trivial cycle

A cycle containing one element is called a *trivial cycle*.

Problem 265

How many non-trivial cycles could a permutation of *n* elements contain?

Number of assignments

The number of assignments needed to perform a permutation in place is $n - u + v$, where n is the number of elements, u is the number of trivial cycles and v is the number of non-trivial cycles.

Proof:

Every non-trivial cycle of length k requires $k + 1$ assignments. The number of elements in non-trivial cycles is equal $n - u$ and v is added for all non-trivial cycles.

Problem 267

Design an in-place reverse algorithm for forward iterators; that is, it should work for singly-linked lists without modifying the links.

An algorithm is *in-place* if for an input of length n it uses $O(p(\log n))$ additional space where p is a polynomial. (Such algorithms are also called *polylog space* algorithms.)

Heirs of Pythagoras

Lecture 6

Lessons from Mathematics

- Abstracting from specific types to generalized theories

- Breaking the reasoning discourse into a sequence of small self-contained lemmas

Swapping ranges

```
while (condition) swap(*f0++, *f1++);
```

Ranges

	semi-open	closed
Bounded: two iterators	[*i*, *j*)	[*i*, *j*]
Counted: iterator and integer	[*i*, *n*)	[*i*, *n*]

Swapping an explicit range with an implicit range

```
template <ForwardIterator I0,
          ForwardIterator I1>
I1 swap_ranges(I0 f0, I0 l0, I1 f1) {
  while (f0 != l0) swap(*f0++, *f1++);
  return f1;
}
```

The Law of Useful Return

A procedure should return all the potentially useful information it computed.

- this does not imply doing unneeded extra computations.
- this does not imply returning useless information.

The Law of Separating Types

Do not assume that two types are the same when they may be different.

```
template <ForwardIterator I0,
          ForwardIterator I1>
I1 swap_ranges(I0 f0, I0 l0, I1 f1);
```

not

```
template <ForwardIterator I>
I swap_ranges(I f0, I l0, I f1);
```

Swapping explicit ranges

```
template <ForwardIterator I0,
          ForwardIterator I1>
pair<I0, I1> swap_ranges(I0 f0, I0 l0,
                         I1 f1, I1 l1) {
  while (f0 != l0 && f1 != l1) {
    swap(*f0++, *f1++);
  }
  return pair<I0, I1>(f0, f1);
}
```

The Law of Completeness

When designing an interface, consider all the related procedures.

Swapping counted ranges

```
template <ForwardIterator I0,
          ForwardIterator I1,
          Integer N>
pair<I0, I1> swap_ranges_n(I0 f0,
                           I1 f1,
                           N n) {
  while (n != N(0)) {
    swap(*f0++, *f1++);
    --n;
  }
  return pair<I0, I1>(f0, f1);
}
```

Problem 278

Why don’t we provide

```
pair<I0, I1> swap_ranges_n(I0 f0,
                           I1 f1,
                           N0 n0,
                           N1 n1)
```

Indexing

While mathematical texts index sequences from 1, the computer science convention of indexing from 0 is more natural.

- For a sequence with n elements the indices are in the range [0, n) and any iteration is bounded by the length.
- rotating *n* elements to the right by *k* transforms an index *i* to the index *i* + *k* mod *n*.

Rotation

A permutation of n elements by k where $k \geq 0$:

(k mod n, k + 1 mod n, …, k + n – 2 mod n, k + n – 1 mod n)

is called an *n by k* rotation.

(We index permutations from 0.)

rotate

rotate is the most important algorithmic primitive of which you never heard.

We are going to see its uses throughout the next Journey.

Designing Interfaces

- We can design a useful interface only after we figure out its multiple future uses.

- The design is a multi-pass activity.

Interface to rotate

Experience shows that it is convenient to define rotation with three iterators: *f*, *m* and *l* where [*f*, *m*) and [*m*, *l*) are valid ranges. Rotation then interchanges ranges [*f*, *m*) and [*m*, *l*).

An example of rotate

```
0 1 2 3 4 5 6
f  m       l
```

Produces:

```
2 3 4 5 6 0 1
```

Problem 285

Prove that if we do rotate(f, m, l) then it performs distance(*f*, *l*) by distance(*m, l*) rotation.

Gries-Mills algorithm

```
template <ForwardIterator I>
void gries_mills_rotate(I f, I m, I l) {
  // u = distance(f, m) && v = distance(m, l)
  if (f == m || m == l) return; // u == 0 || v == 0
  pair<I, I> p = swap_ranges(f, m, m, l);
  while(p.first != m || p.second != l) {
    if (p.first == m) {         // u < v
      f = m; m = p.second;      // v = v - u
    } else {                    // v < u
      f = p.first;              // u = u - v
    }
    p = swap_ranges(f, m, m, l);
  }
  return;                       // u == v
}
```

Problem 287

If you inline swap_ranges you see that the algorithm does unnecessary iterator comparisons. Re-write the algorithm so that no unnecessary iterator comparisons are done.

The number of swaps during the last swap_range

The number of swaps during the last swap_range is gcd(n, k) where

n = distance(f, l)

k = distance(m, l)

Number of cycles in the rotate

- Every swap_range moves elements along their cycles.
- The number of cycles is gcd(n, k)

- For a formal proof see EoP pages 178-179

Complexity of Gries-Mills

- During the call to all the swap_range except the last one, every swap puts one element into the final destination.
- During the last swap_range every swap puts two elements into the final destination.
- The total number of swaps n – gcd(n, k)
- The total number of assignments 3(n – gcd(n, k))

Trace of Gries-Mills

```
0 1 2 3 4 5 6
f  m       l
2 3 0 1 4 5 6
  f   m    l
2 3 4 5 0 1 6
     f  m  l
2 3 4 5 6 1 0
      f m  l
2 3 4 5 6 0 1
       f m
         l
```

The return value of rotate

- Many applications benefit if rotate returns a new middle: a position where the first element moved.
- Observe, that
  rotate(f, rotate(f, m, l), l)
  is an identity permutation.
- The task is to find a way to return the desired value without doing any extra work.

An auxiliary rotate

```
template <ForwardIterator I>
I rotate_unguarded(I f, I m, I l, I m1) {
  // assert(f != m && m != l)
  pair<I, I> p = swap_ranges(f, m, m, l);
  while (p.first != m || p.second != l) {
    f = p.first;
    if (m == f) m = p.second;
    p = swap_ranges(f, m, m, l);
  }
  return m1;
}
```

Final rotate for forward iterators

```
template <ForwardIterator I>
I rotate(I f, I m, I l, forward_iterator_tag) {
  if (f == m) return l;
  if (m == l) return f;
  pair<I, I> p = swap_ranges(f, m, m, l);
  while (p.first != m || p.second != l) {
    if (p.second == l)
      return rotate_unguarded(p.first, m, l, p.first);
    f = m;
    m = p.second;
    p = swap_ranges(f, m, m, l);
  }
  return m;
}
```

In search of faster algorithm

- We know that we can do a permutation without trivial cycles with $n + c$ assignments where n is the size of permutation and c is the number of cycles.
- Such an algorithm requires stronger requirements on the iterators.
- And we need to go through cycles in reverse order!

cycle_from

```
template <ForwardIterator I,
         Transformation F>
void cycle_from(I i, F from) {
  typedef typename iterator_traits<I>::value_type V;
  V tmp = *i;
  I start = i;
  for (I j = from(i); j != start; j = from(j)) {
    *i = *j;
    i = j;
  }
  *i = tmp;
}
```

transformation for rotate

```
template <RandomAccessIterator I>
struct rotate_transform {
  typedef typename iterator_traits<I>::difference_type N;
  N plus;
  N minus;
  I m1;

  rotate_transform(I f, I m, I l) :
    plus(m – f), minus(m – l), m1(f + (l – m)){}

  I operator()(I i) const {
    return i + ((i < m1) ?  plus : minus);
  }
};
```

Modified Fletcher-Silver algorithm

```
template <RandomAccessIterator I>
I rotate(I f, I m, I l,
         random_access_iterator_tag) {
  if (f == m) return l;
  if (m == l) return f;
  typedef iterator_traits<I>::difference_type N;
  N d = gcd(m - f, l - m);
  rotate_transform<I> rotator(f, m, l);
  while (d-- > 0) cycle_from(f + d, rotator);
  return rotator.m1;
}
```

3-reverse rotate

```
template <BidirectionalIterator I>
void three_reverse_rotate(I f, I m, I l) {
    reverse(f, m);
    reverse(m, l);
    reverse(f, l);
}
```

Problem 300

- How many assignments does 3-reverse rotate perform?

reverse_until

```
template <BidirectionalIterator I>
pair<I, I> reverse_until(I f, I m, I l) {
  while (f != m && m != l) swap(*f++, *--l);
  return pair<I, I>(f, l);
}
```

Bidirectional rotate

```
template <BidirectionalIterator I>
I rotate(I f, I m, I l,
        bidirectional_iterator_tag) {
    reverse(f, m);
    reverse(m, l);
    pair<I, I> p = reverse_until(f, m, l);
    reverse(p.first, p.second);
    if (m == p.first) return p.second;
    return p.first;
}
```

Iterator category dispatch

```
template <ForwardIterator I>
inline
I rotate(I f, I m, I l) {
  typename iterator_traits<I>::iterator_category c;
  return rotate(f, m, l, c);
}
```

Bidirectional reverse

```
template <BidirectionalIterator I>
void reverse(I f, I l,
             bidirectional_iterator_tag) {
  while (f != l && f != --l) swap(*f++, *l);
}
```

Returning from reverse

It might appear that according to the law of useful return we should return

```
pair<I, I>(f, l)
```

However, there is no evidence that it is useful; therefore the law does not apply.

reverse_n

```
template <BidirectionalIterator I,
          Integer N>
void reverse_n(I f, I l, N n) {
  n >>= 1;
  while (n-- > N(0)) {
    swap(*f++, *--l);
  }
}
```

Problem 307

Unroll the loop of reverse_n 4 times.

(Read about *Duff’s device*.)

Random Access reverse

```
template <RandomAccessIterator I>
void reverse(I f, I l,
             random_access_iterator_tag) {
  reverse_n(f, l, l - f);
}
```

Recursive reverse

```
template <ForwardIterator I,
          BinaryInteger N>
I reverse_recursive(I f, N n) {
  if (n == 0) return f;
  if (n == 1) return ++f;
  N h = n >> 1;
  I m = reverse_recursive(f, h);
  advance(m, n & 1);
  I l = reverse_recursive(m, h);
  swap_ranges_n(f, m, h);
  return l;
}
```

Forward Iterator reverse

```
template <ForwardIterator I>
void reverse(I f, I l,
             forward_iterator_tag) {
  reverse_recursive(f, distance(f, l));
}
```

Generic reverse

```
template <ForwardIterator I>
inline
void reverse(I f, I l) {
  typename iterator_traits<I>::iterator_category c;
  reverse(f, l, c);
}
```

Complexity of computation

- Time complexity
  - Hartmanis and Stearns
  - time hierarchy theorem
- Space complexity
  - Lewis, Stearns and Hartmanis

Space complexity in concrete algorithmics

- In-place
  - An algorithm is *in-place* if for an input of length n it uses $O(p(\log n))$ additional space where p is a polynomial. (Such algorithms are also called *polylog space* algorithms.)
- Not in-place
  - usually means that you can create a copy of your data

reverse with buffer

```
template <ForwardIterator I,
          Integer n,
          BidirectionalIterator B>
I reverse_n_with_buffer(I f, N n, B buffer) {
  return reverse_copy(buffer,
                      copy_n(f, n, buffer),
                      f);
}
```

reverse_copy

```
template <BidirectionalIterator I,
          OutputIterator O>
O reverse_copy(I f, I l, O r) {
  while (f ! = l) *r++ = *--l;
  return r;
}
```

Memory-adaptive algorithms

- In practice, the dichotomy of in-place and not in-place algorithms is not very useful.
- While the assumption of unlimited memory is not realistic, neither is the assumption of only polylog extra memory.
- Usually 25%, 10%, 5% or at least 1% of extra memory is available.
- Algorithms need to adapt to however much is available.

Adaptive reverse

```
template <ForwardIterator I, Integer n,
          BidirectionalIterator B>
I reverse_n_adaptive(I f, N n,
                     B b, N b_n) {
  if (n == N(0)) return f;
  if (n == N(1)) return ++f;
  if (n <= b_n) return reverse_n_with_buffer(f, n, b);
  N h = n >> 1;
  I m = reverse_n_adaptive(f, h, b, b_n);
  advance(m, n & 1);
  I l = reverse_n_adaptive(m, h, b, b_n);
  swap_ranges_n(f, m, h);
  return l;
}
```

The sad story of get_temporary_buffer

- get_temporary_buffer takes a size and should return the largest available buffer not greater than the size that fits into physical memory.
- There is a bogus implementation – using malloc till it returns something.
- No vendor provides the correct implementation.

Words of wisdom

Whose heirs are we?