Greatest Common Measure:
the Last 2500 Years

Alexander Stepanov

This lecture was originally prepared as the 1999 Arthur Schoffstall Lecture in Computer Science and Computer Engineering at the Rensselaer Polytechnic Institute

Abstract

The talk describes development of Euclid's algorithm from early Pythagoreans to modern times. This development shows gradual evolution of a notion of abstract (or generic) algorithm. While the term *generic programming* was introduced by Musser and Stepanov in 1988, the notion of algorithmic genericity goes back to remote centuries and is one of the main driving forces of mathematical progress.

Study of mathematics develops an architectural ability for organizing large body of knowledge, which is still lacking in Computer Science. Euclid's Elements (with a good modern guide such as Robin Hartshorne) is a wonderful textbook for software engineers.

The sense of history is very important for a scientist in order to be able to evaluate what is, and is not, important. To understand something we need to know its history.

Pythagoras (570BC - 475BC)

Plimpton 322

"He attached supreme importance to the study of arithmetic, which he advanced and took out of the region of commercial utility."
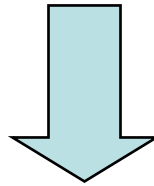
Aristoxenus

He maintained that "the principles of mathematics are principles of all existing things."
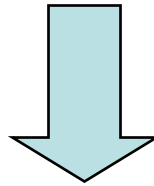
Aristotle

He discovered "the theory of irrationals and the construction of cosmic bodies."
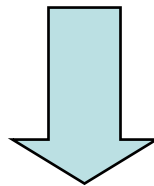
Proclus

ASTRONOMY

⬇

GEOMETRY

⬇

NUMBER THEORY

⬇

MUSIC

To reduce the world to numbers, one needs the absolute common measure, the smallest physically possible segment, the quantum of space.

IT DOES NOT EXIST!

However small a measure we pick there are segments that cannot be measured by it.

gcm(a, a) = a

gcm(a, b) = gcm(a, a + b)

a > b $\implies$ **gcm(a, b) = gcm(a - b, b)**

gcm(a, b) = gcm(b, a)

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

```
196     42
154     42
112     42
 70     42
 28     42
 28     14
 14     14 Done!

GCD:   14
```

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

(Greeks found the principle that any set of natural numbers has the smallest element a very powerful proof technique. Now we know that it is equivalent to mathematical induction.
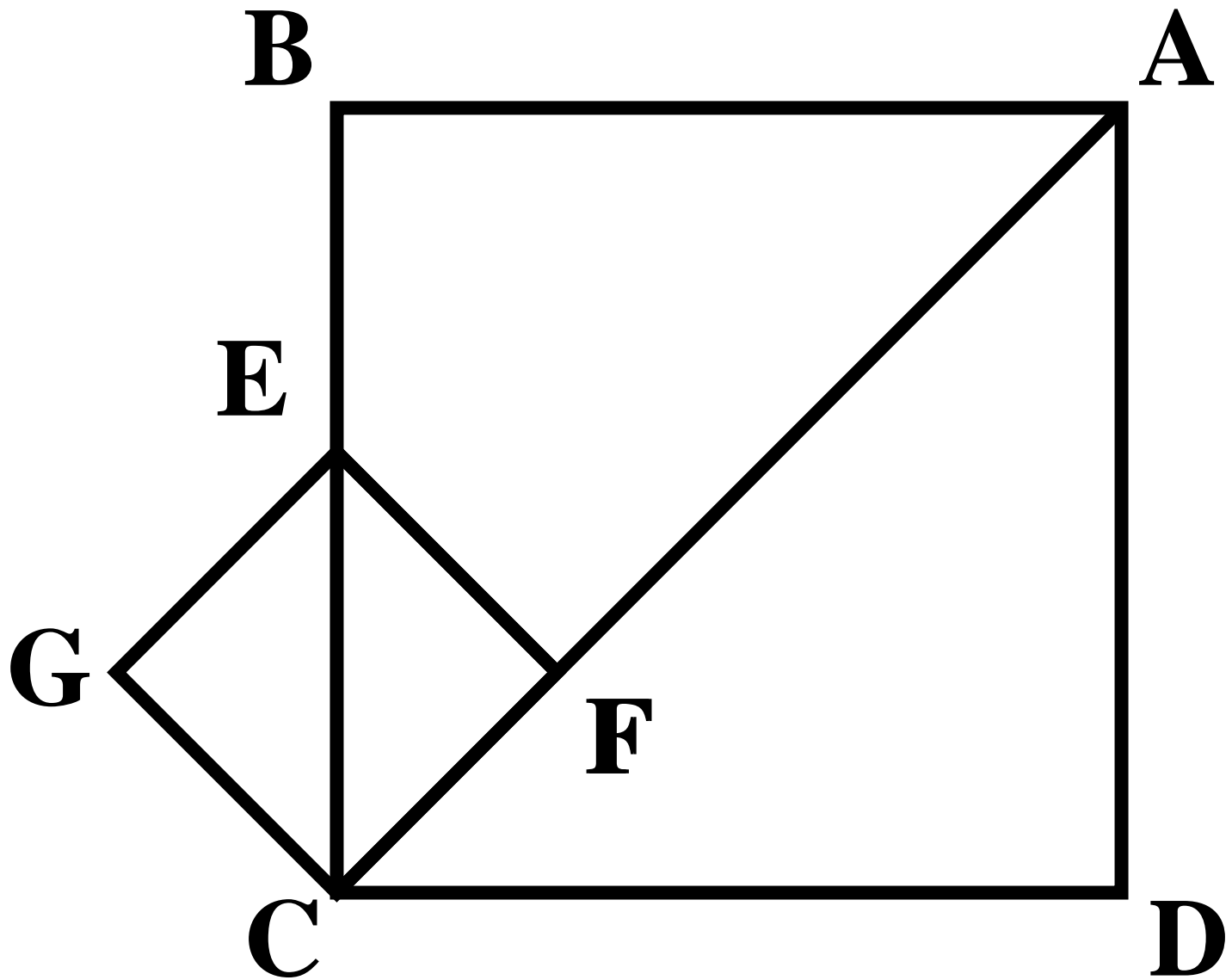
So, in order to prove that something does not exist, you prove that if it exists, a smaller one also exists.)
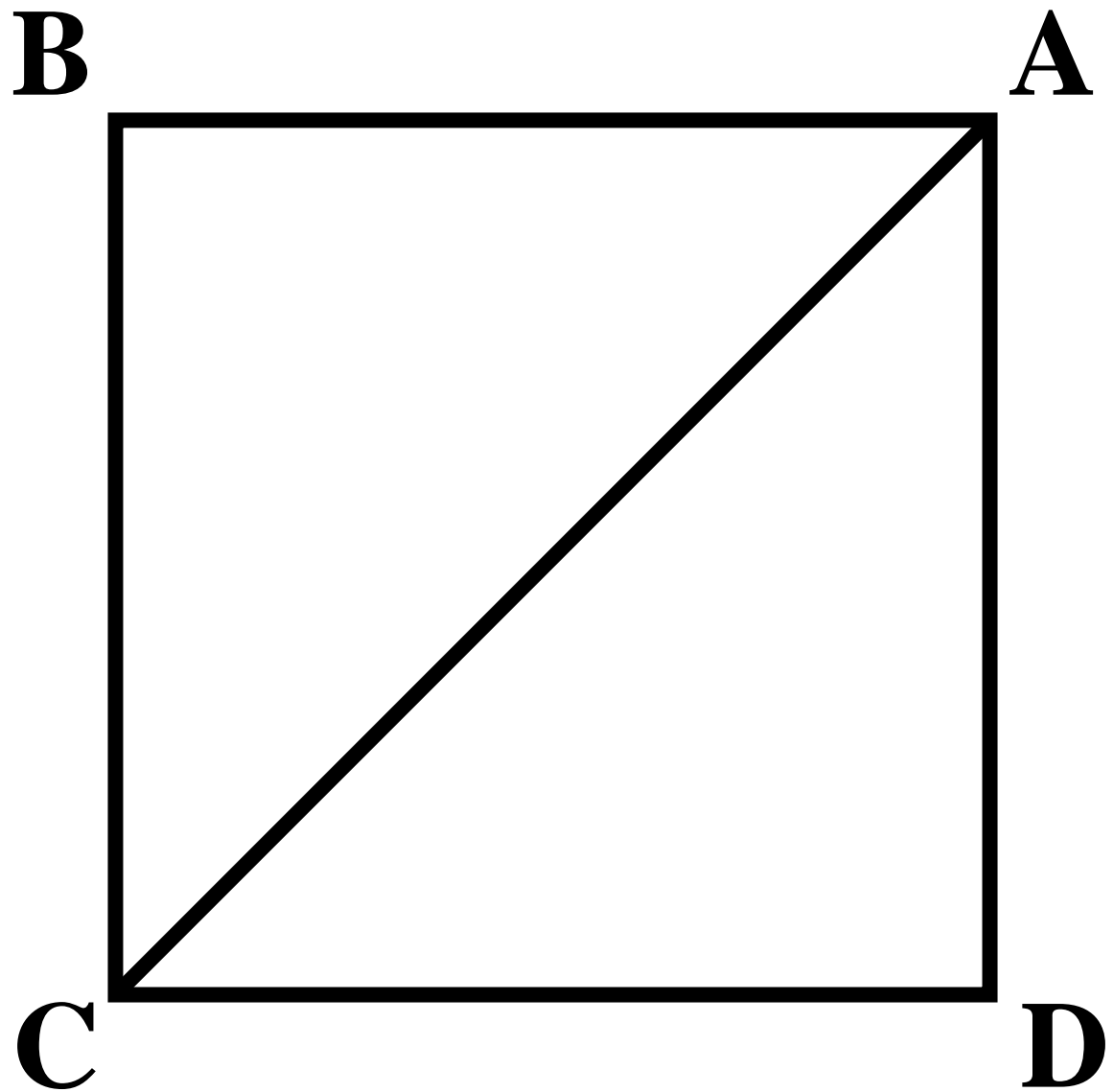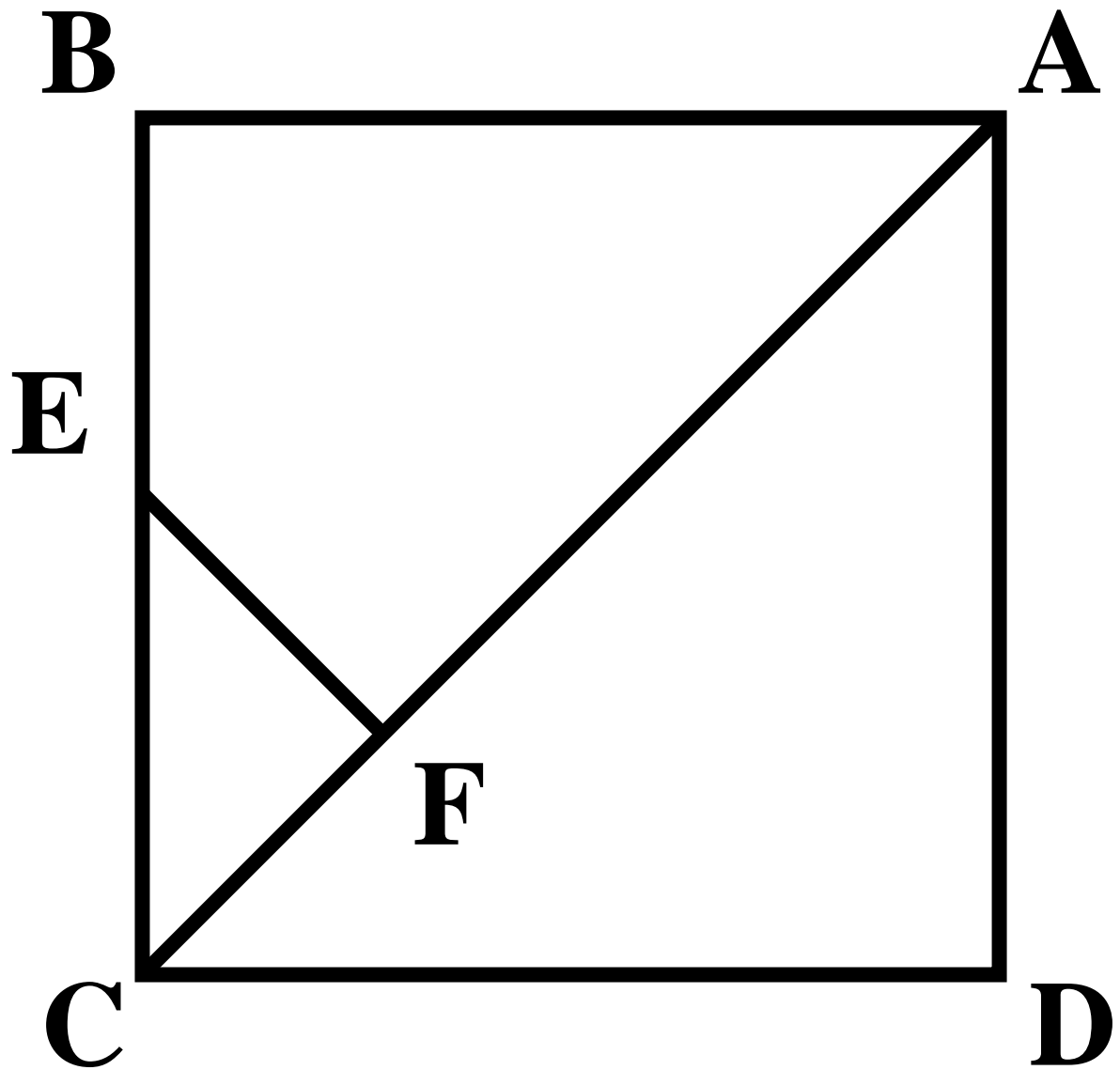
AB = AF   AC ⊥ EF

$AC \perp CG$ $EG \perp EF$
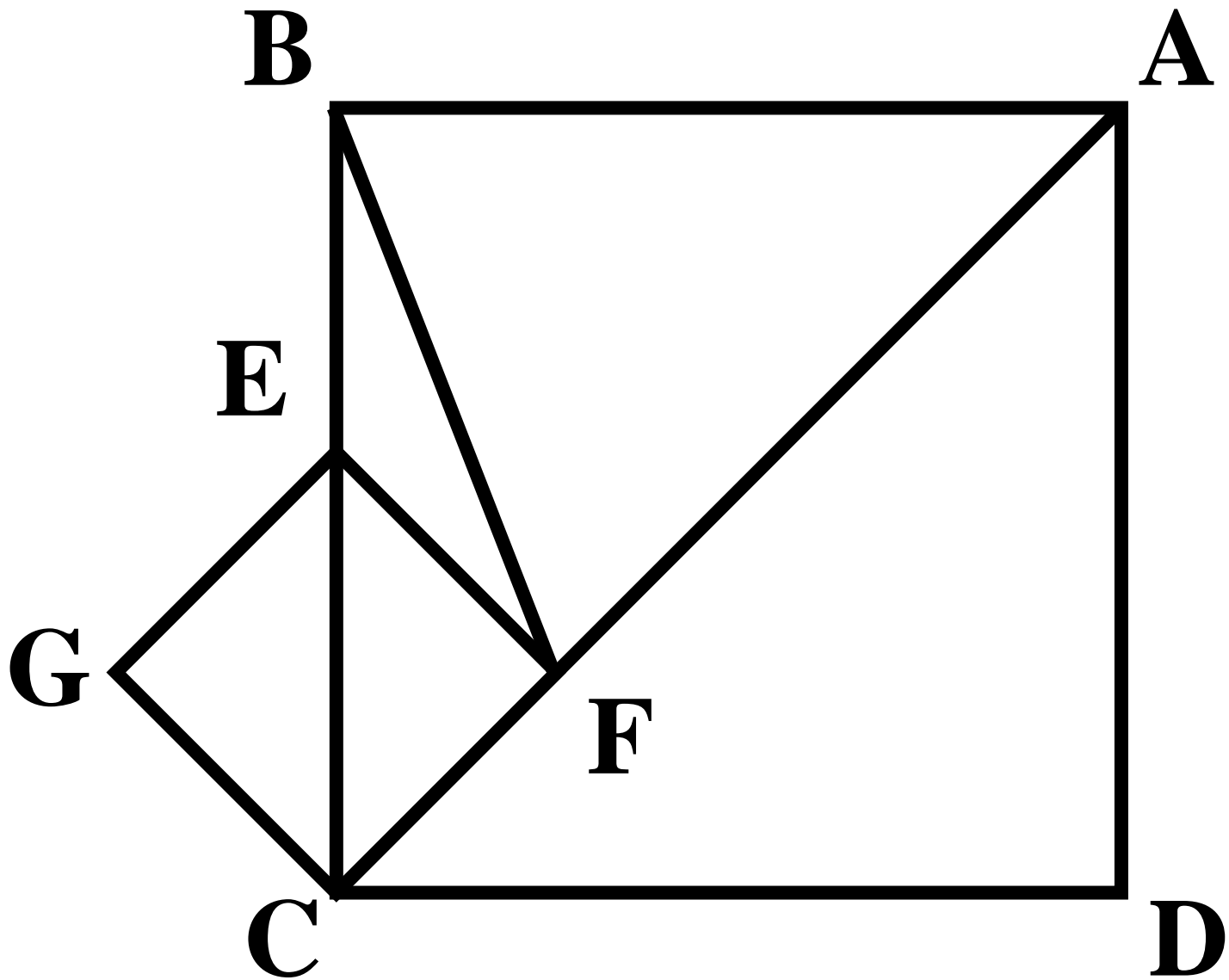
CF = EF = EB
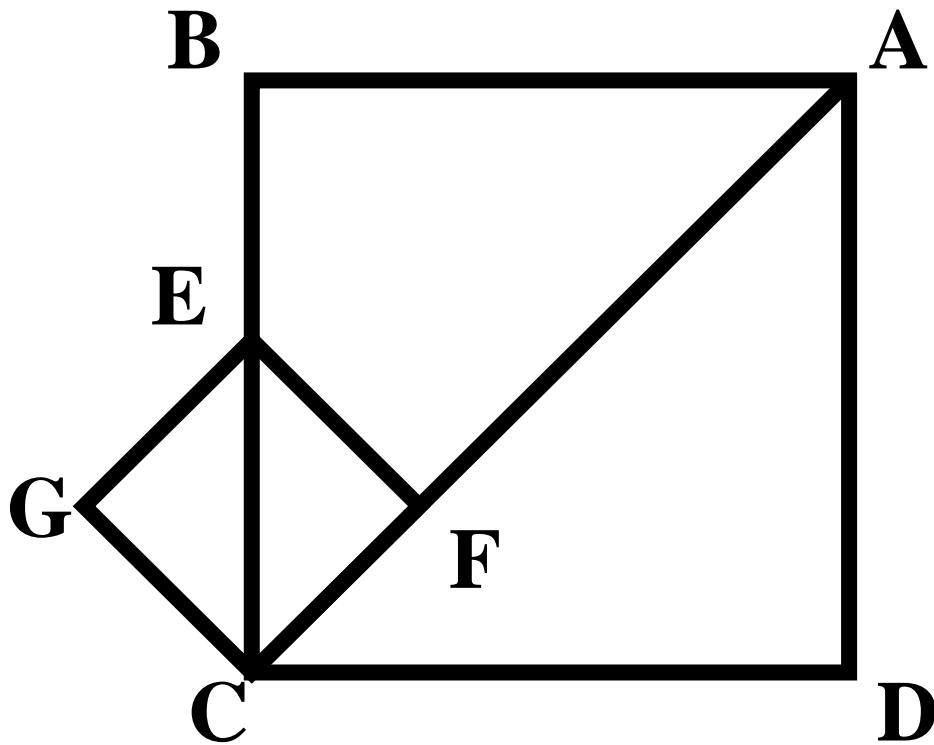
$$\text{gcm(AC,AB)} = \text{gcm(AB,CF)}$$

$$\text{gcm(CE,CF)} = \text{gcm(AC,AB)}$$
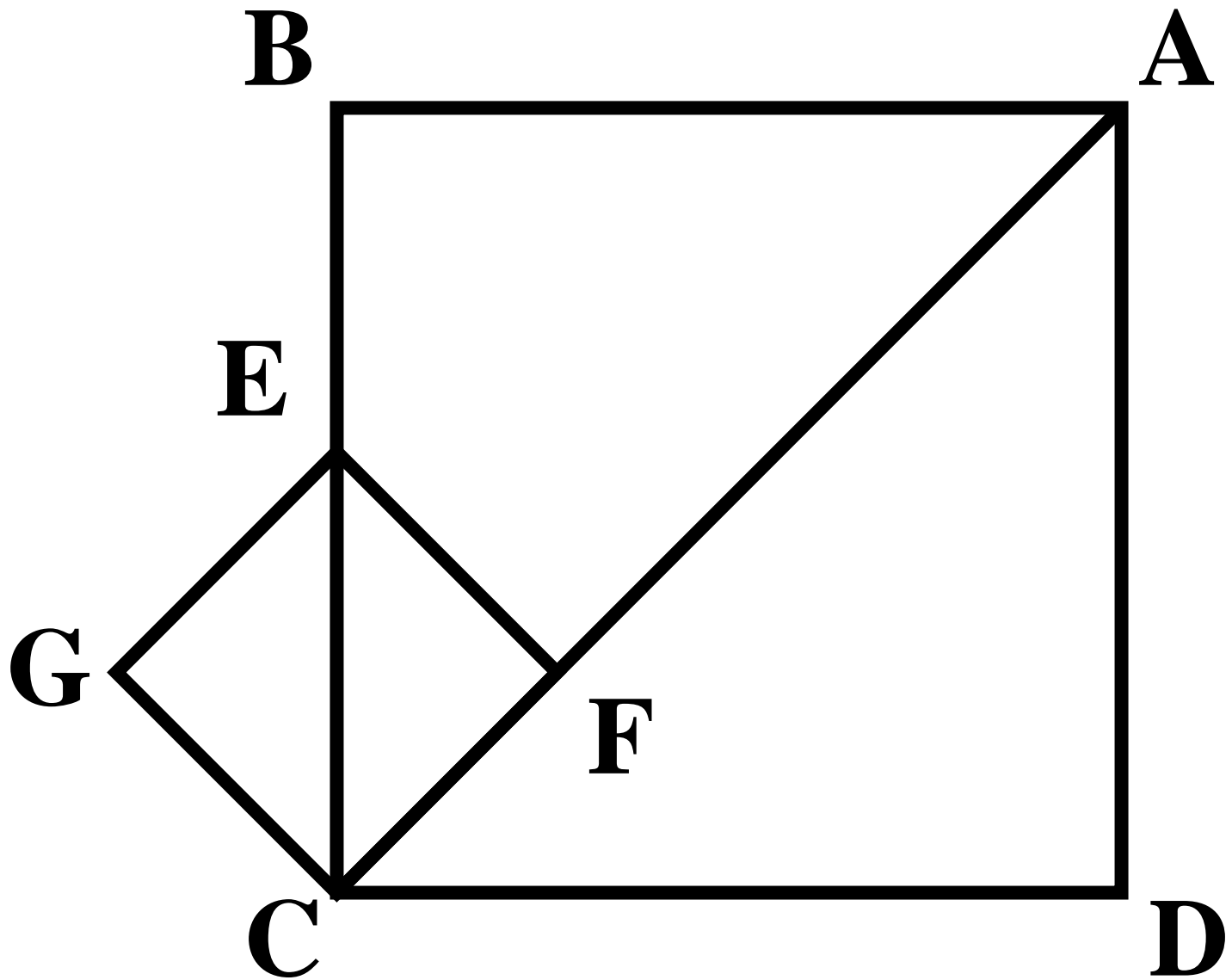
$$EC > EB \longrightarrow EB < AB/2$$

We constructed a smaller square that could be measured by this measure. Contradiction!

The side and the diagonal of the square produced by the proof result from two iterations of the algorithm:

d, s => s, d - s => 2s - d, d - s

And the original ratio is repeated:

$$d/s = (2s - d)/(d - s)$$

Or, as we would say it now, Pythagoras discovered that √2 is irrational

Plato (427BC - 347BC)

ΑΓΕΩΜΕΤΡΗΤΟΣ ΜΗΔΕΙΣ ΕΙΣΙΤΩ

LET NO ONE WHO DOES NOT KNOW GEOMETRY ENTER

They 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

Plato's Algorithm

```
void sqrt_of_2(int count) {
  int side = 1;
  int diagonal = 1;

  for(int n = 0; n < count; ++n) {
    int tmp = side + diagonal;
    diagonal = tmp + side;
    side = tmp;
  }

  display(diagonal, side, count);
}
```

Rational diameter of:

1 is 1   (1)
2 is 3   (1.5)
5 is 7   (1.4)
12 is 17   (1.41667)
29 is 41   (1.41379)
70 is 99   (1.41429)
169 is 239   (1.4142)
408 is 577   (1.41422)
985 is 1393   (1.41421)
2378 is 3363   (1.41421)

Euclid (325BC-265BC)

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 threepence, since he must make gain out of what he learns."

Strobaeus, *Florilegium*

Euclid guaranteed termination by changing the input types:

```
unsigned int gcd(unsigned int a,
                      unsigned int b) {
   assert(a > 0 && b > 0);
   // should wait for Arabs
   // and Leonardo Pisano
   if (a == b)    return a;
   if (a > b)    return gcd(a-b, b);
/* if (b > a) */ return gcd(a, b-a);
}
```

Elements, Book X

Proposition 1. Two unequal magnitudes being set out, if from the greater there is subtracted a magnitude greater than its half, and from that which is left a magnitude greater than its half, and if this process is repeated continually, then there will be left some magnitude less than the lesser magnitude set out.

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.

Elements, Book X

Proposition 3. To find the greatest common measure of two given commensurable magnitudes.

Corollary. If a magnitude measures two magnitudes, then it also measures their greatest common measure.

Euclid guaranteed termination by changing the input types:

```
unsigned int gcd(unsigned int a,
                 unsigned int b) {
   assert(a > 0 && b > 0);
   // should wait for Arabs
   // and Leonardo Pisano
   if (a == b)    return a;
   if (a > b)     return gcd(a-b, b);
/* if (b > a) */ return gcd(a, b-a);
}
```

Why **a-b**, not **a%b**?

Lincoln and Euclid

"He studied and nearly mastered the six books of Euclid since he was a member of Congress. He regrets his want of education, and does what he can to supply the want."

--Lincoln's Autobiography of 1860

New Math against Euclid

In 1959, at a conference on the teaching of mathematics in Réalmont, France, Jean Dieudonné rose and hurled the slogans "Down with Euclid!" and "Death to triangles!"

I.M.Yaglom, *Elementary Geometry Then and Now*

The Years of Decline: 212BC - 1202AD

In summo apud illos honore geometria fuit, itaque nihil mathematicis inlustrius; at nos metiendi ratiocinandique utilitate huius artis terminavimus modum.

Among them [the Greeks] geometry was held in highest honor; nothing was more glorious than mathematics. But we [the Romans] have limited the usefulness of this art to measuring and calculating.

Cicero, *Tusculan Disputations*

Omar Khayyam (1048-1123)

Sqrt(2) = 1 + 1/(2 + 1/(2 + 1/(2 + ...

Continued fractions generated by quotients represent a ratio of any two segments.

Leonardo Pisano (1170-1250)

It took over 1500 years to move to:

```
unsigned int gcd(unsigned int m,
                 unsigned int n) {
    while (n != 0) {
        unsigned int t = m % n;
        m = n;
        n = t;
    }
    return m;
}
```

```
196    42    196 = 42*4 + 28

 42    28     42 = 28*1 + 14

 28    14     28 = 14*2 + 0

 14     0     Done!


GCD:   14
```

Simon Stevin (1548 - 1620)
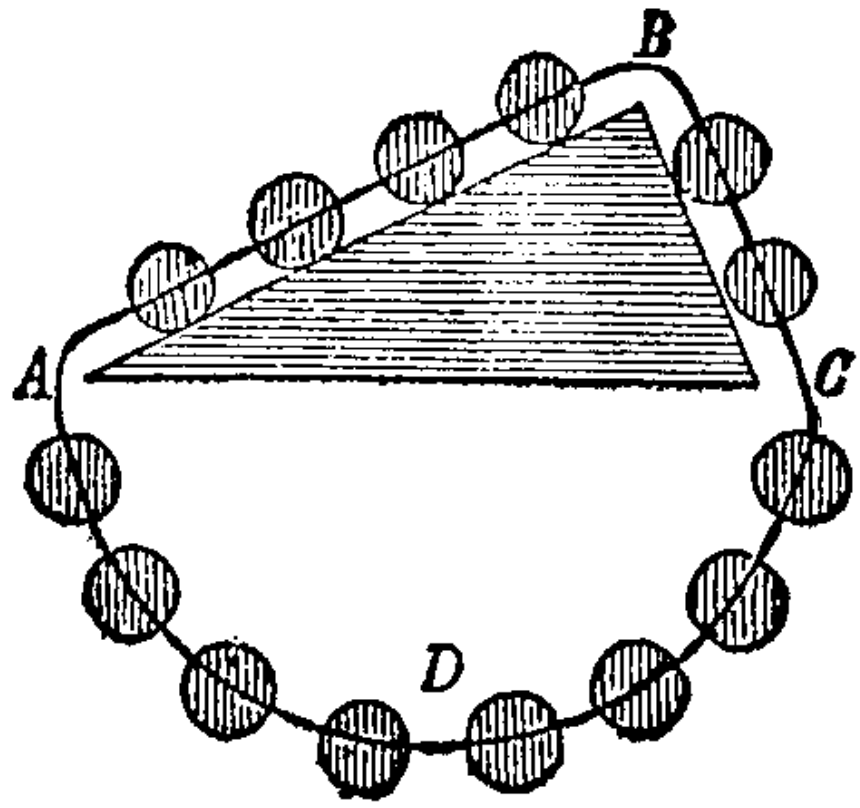
Simon Stevin:

```
int gcd(int m, int n) {
    while (n != 0) {
        int t = m % n;
        m = n;
        n = t;
    }
    return m;
}
```

Simon Stevin:

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

$$
\begin{array}{r}
3x^2+2x-2 \\
x-2 \overline{\smash{\big)}\ 3x^3-4x^2-6x+10} \\
\underline{3x^3-6x^2} \\
2x^2-6x \\
\underline{2x^2-4x} \\
-2x+10 \\
\underline{-2x+4} \\
6
\end{array}
$$

Carl Friedrich Gauss
(1777 - 1855)

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

Gauss, *Disqisitiones Arithmeticae*, art. 18

Carl Gauss:

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

Finding a Gaussian remainder

To find a remainder of `z1` divided by `z2`

1. Construct a square grid on the complex plane generated by `z2` together with `i*z2, -i*z2` and `-z2`.

2. Find a square in the grid containing `z1`.

3. Find a vertex w of the square closest to `z1`.

4. The remainder is `z1 - w`.
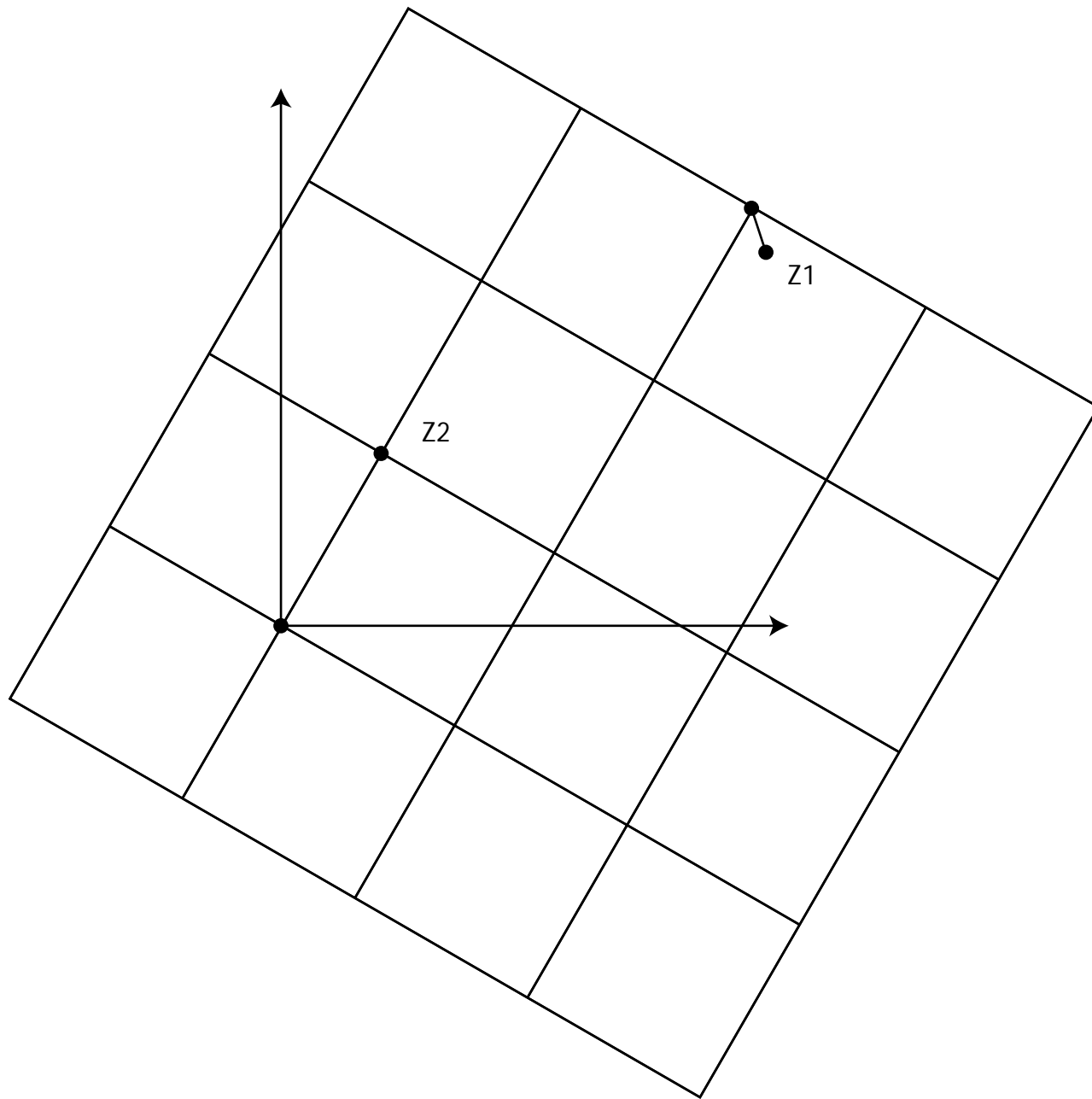
Lejeune Dirichlet
(1805 - 1857)

"It is now clear that 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, so one is entitled to make the following claim: 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."

Lejeune Dirichlet,
Lectures on Number Theory

"If one considers all numbers of the form
$$t + n\sqrt{-a}$$
where `a` is a particular positive integer and `t` and `n` are arbitrary integers … 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`. … For example, `21 = 3*7 = (1+2√-5)*(1-2√-5)…`"

Lejeune Dirichlet,

Lectures on Number Theory

Richard Dedekind
(1831 - 1916)

Algebraic integers

Algebraic integers are roots of monic polynomials with integer coefficients.

Es steht alles schon bei Dedekind!

Emmy Noether

(It is all already in Dedekind.)

All = {rings, fields, ideals, modules,…}

Euclid and Göttingen

- Gauss
  - Regular polygons
  - Non-Euclidean geometry
    - Lobachevsky and Bolyai
    - The equality of two volumes of two tetrahedra
- Dirichlet
  - Infinity of primes in arithmetic progression
- Riemann
  - Non-Euclidian geometry
- Dedekind
  - Edoxian theory reborn
- Klein
  - Erlanger program
- Hilbert
  - Foundations of Geometry
  - One must be able to say at all times -- instead of points, straight lines, and planes -- tables, beer mugs, and chairs
  - Mechanization of mathematics

Emmy Noether (1882 -1935)

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)

Dedekind, Noether, van der Waerden:

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

Euclidean domain

- A commutative ring (`+`, `-`, `*`)
- Function `norm`: `Ring -> Unsigned`
  - `norm(a*b) >= norm(a)`
  - For any `a`, `b`, where `b != 0`, there exist `q`, `r`, such that `a == q*b + r` and `r == 0 || norm(r) < norm(b)`

Donald Knuth
(1938 -)

Knuth's objection: $\mathbf{gcd(1, -1) = -1}$

```
template <class
            EuclideanRingElement>
EuclideanRingElement
gcd(EuclideanRingElement m,
    EuclideanRingElement n) {
  while (n != 0) {
    EuclideanRingElement t = m % n;
    m = n;
    n = t;
  }
  if (m < 0) m = -m;
  return m;
}
```

Depends on the definition!

Greatest common divisor is a common divisor that is divisible by any other common divisor.

What is Euclidian Domain?

- What are operations and their requirements?

- What are intended models?

- What are related algorithms?

Extended Euclid

```cpp
template <class EuclideanDomain>
triple<EuclideanDomain, EuclideanDomain, EuclideanDomain>
extended_euclid(EuclideanDomain u, EuclideanDomain v) {
    EuclideanDomain u0 = 1;
    EuclideanDomain v0 = 0;
    EuclideanDomain u1 = u;
    EuclideanDomain v1 = v;
    while (v1 != 0) {
        EuclideanDomain q = u1/v1;
        u0 -= v0 * q;
        swap(u0, v0);
        u1 -= v1 * q;
        swap(u1, v1);
    }
    return make_triple(u0, (u1 - u * u0) / v, u1);
}
```

Josef Stein (1961):

```
gcd(n, 0) = gcd(0, n) = n
gcd(n, n) = n
gcd(2n, 2m) = 2gcd(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(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    Done!
GCD: 7*2 = 14
```

```
template <class BinaryInteger>
BinaryInteger gcd(BinaryInteger m,
                  BinaryInteger n) {

  make_non_negative(m);
  make_non_negative(n);

  if (is_zero(m)) return n;
  if (is_zero(n)) return m;

  int d = 0;

  while (is_even(m) && is_even(n)) {
    half_non_negative(m);
    half_non_negative(n);
    ++d;
  }
```

```
while (is_even(m)) half_non_negative(m);

while (is_even(n)) half_non_negative(n);

while (true)
  if (m < n) {
    n = n - m;
    do {
      half_non_negative(n);
    } while (is_even(n));
  } else if (n < m) {
    m = m - n;
    do {
      half_non_negative(m);
    } while (is_even(m));
  } else
    return left_shift(m, d);
}
```

Stein for polynomials

Use x as 2!

Stein for polynomials over a field

```
gcd(p,0) = gcd(0,p) = p
gcd(p, p) = p
gcd(x*p1,x*p2) = x*gcd(p1, p2)
gcd(x*p1,x*p2+c) = gcd(p1, x*p2+c)
gcd(x*p1+c,x*p2) = gcd(x*p1+c,p2)
if degree(p1) >= degree(p2)
 gcd(x*p1+c1,x*p2+c2) =
  gcd(p1-(c1/c2)*p2, x*p2+c2)
if degree(p1) < degree(p2)
 gcd(p1,p2) = gcd(p2,p1)
```

x^3-3x-2 $\qquad$ x^2-4

$x^3-.5x^2-3x$ $\qquad$ x^2-4

$x^2-.5x-3$ $\qquad$ x^2-4

x^2-2x $\qquad$ x^2-4

$x-2$ $\qquad$ x^2-4

x^2-4 $\qquad$ $x-2$

x^2-2x $\qquad$ $x-2$

$x-2$ $\qquad$ $x-2$ Done!

GCD: $x-2$

Weilert algorithm for Gaussian Integers

Use 1+i as 2!

Division by 1+i

$a+bi/1+i = (a+bi)(1-i)/(1+i)(1-i) =$

$(a+bi)(1-i)/2 = ((a+b) - (a-b)i)/2$

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+i*z_2 and z_1-i*z_2 are also divisible by $1+i$.

And,

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

Damgård and Frandsen Algorithm

- Stein algorithm works for Eisenstein integers Z[ζ], i.e. the integers extended with ζ ($\frac{-1+\sqrt{-3}}{2}$), a complex primitive third root of unity.

- We use 1 - ζ as our 2.

What is Stein domain?

Few definitions

- **is_unit(u)** iff there is a **v** such that **v*u == 1**

- **are_associates(m, n)** iff **is_unit(u) && u*n == m**

- **is_smallest_prime(p)** iff for any **q != 0**,
    norm(q) < norm(p) => is_unit(q)

Few lemmas

- **`is_unit(u) => norm(a) == norm(u*a)`**

- if a Euclidean ring is not a field it has a smallest prime

- **`is_smallest_prime(p) => is_unit(q%p) || q%p == 0`**

Conjecture

Every Euclidean domain is a Stein domain

```
template <typename EuclideanElement>
EuclideanElement binary_gcd(EuclideanElement m, EuclideanElement n)
{
    EuclideanElement p = smallest_prime<EuclideanElement>();

    if (is_zero(m)) return n;
    if (is_zero(n)) return m;

    EuclideanElement r = unit<EuclideanElement>();

    while (n%p == 0 && m%p == 0) {
                r *= p;
                n /= p;
                m /= p;
    }

    while (n%p == 0) n /= p;
    while (m%p == 0) m /= p;

    while (!are_associates(m, n)) {
        m -= ((m%p)/(n%p))*n;    // does not always work ☹
        while (m%p == 0) m /= p;
        swap(m, n);
    }

    return n*r;
}
```

Non-Euclidean Stein domains

In 2004 Agarwal and Frandsen demonstrated that there is a non-Euclidean ring (ring of integers in $\mathbb{Q}(\sqrt{-19})$) that is a Stein domain

Conclusions

- Computer Science is Mathematics
- The Math which is important for CS is what should be our high school math
  - Algebra
  - Geometry
- Finally, know your heroes
  - Euclid and Euler, not Jobs and Gates

Suggestions for study

- Geometry teaches you an architectural sense:

  - Euclid, *Elements*, translated by Sir Thomas L. Heath, Dover , 1956 (3 volumes) - Amazon

- Algebra teaches you transformational techniques:

  - George Chrystal, *Textbook of Algebra*, AMS Chelsea Publishing, 1964 (2 volumes) - AMS

Extended bibliography

This is the list of books and papers from which the material for the talk was gathered.

David Fowler, *The Mathematics Of Plato's Academy,* Oxford, 1999

John Stillwell, *Mathematics and Its History*, Springer-Verlag, 1989

Sir Thomas Heath, *History of Greek Mathematics*, Dover, 1981 (2 volumes)

Euclid, *Elements,* translated by Sir Thomas L. Heath, Dover , 1956 (3 volumes)

B. L. van der Waerden, *Geometry and Algebra in Ancient Civilizations*, Springer-Verlag, 1983

Robin Hartshorne, *Geometry: Euclid and Beyond*, Springer-Verlag, 2000

Lucio Russo, *The Forgotten Revolution*, Springer-Verlag, 2004

Laurence E. Siegler, *Fibonacci's Liber Abaci*, Springer-Verlag, 2002

Nicolas Bourbaki, *Elements of the History of Mathematics,* Springer-Verlag, 1999

Carl Friedrich Gauss, *Disquisitiones Arithmaticae,* Yale, 1965

John Stillwell, *Elements of Number Theory*, Springer-Verlag, 2002

Peter Gustav Lejeune Dirichlet, *Lectures on Number Theory,* AMS, 1999

Richard Dedekind, *Theory of Algebraic Integers*, Cambridge, 1996

B. L. van der Waerden, *Algebra*, Springer-Verlag, 1994

Donald Knuth, *Art of Computer Programming, vol. 2, Seminumerical Algorithms*, Addison-Wesley, 1998

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

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

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

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