Determinant (Gauss).

Greatest Common Divisor (Euclid).

are algebraic.

Remarkable Algorithms (to me most appealing).

- Name: Informing
  - Text by AI "Kuzmich. "Algorithm"

1. History: "Algebra" (AI 1998) come from a

Course Work

- Matrix Algebra
- Graph Isomorphism
- Primality Testing
- Factorization of Polynomials

Algebra & Algorithms
\( P = NP \iff \text{Belief: Perm}(m) \text{ requires } \exp(n) \text{ time to compute} \)

- No sign.
- The permanent
- \( \frac{1}{m} \sum_{i=1}^{m} \text{perm}(\text{cyclic}(1,m,i)) \)
- \( \text{perm}(M) = \frac{1}{m} \text{det}(T) \)
- But can be computed in polynomial time.

Definition involving \( \left( \frac{e}{n} \right) \) binomials.

- Field F matrix over \( m \times n \)
- \( M \in F^{m \times n} \)
- \( \text{det}(M) = \prod_{i=1}^{m} \text{sign}(s) \)
- \( s \subseteq \{1, \ldots, n\} \)
- \( \text{sign}(s) = -1 \) if \( \text{odd} \)
- \( \text{sign}(s) = 1 \) if \( \text{even} \)

Formal definition:

Aside: Determinant

2
Given $f, g \in \mathbb{F}[x]\langle m \rangle$, let $u$ be a root of unity. Let $v = u^n$.

Essence of Idea (over $\mathbb{F}$)

"Lever any field (even better: $\mathbb{C}$, in log-log ($\mathbb{R}$) field)"

Even better: $\mathbb{C}$ (log-log) field

But addition can do better - Karatsuba $O(n^{\log_2 3})$

2. Multiplication: Nivens

3. Addition: Takes $\Theta(n^2)$ time for addition

Polynomials + Algorithms
(Field specific).

Even polynomial is non-trivial.

Factored: \(\mathcal{O}(1.5)\) roughly.

6. \(\mathcal{O}(n \log n)\)

Factorization:

7. \(\mathcal{O}(n \log n)\)

\(f(x) = g(x) \cdot r(x) + r(x)\)

For all \(r \in \mathcal{R}, \deg g \geq \deg r\) all.

Division with remainder: Given \(f, g\) compute.

8. \(\mathcal{O}(n \log n)\) for some \(c 

\text{Compute: } f(a), \ldots, f(n) \text{...} \begin{cases} 
  \text{for some } c 

\text{Multinomial Evaluation: Given } a_0, a_1, \ldots, a_l \text{...} 

\text{compute } c_0, c_1, \ldots, c_n \end{cases}

9. Interpolation: Given \(x_0, \ldots, x_n\): Compute \(c_0, c_1, \ldots, c_n\)
\[ \text{Key Idea:} \quad \text{Factorization over finite fields} \]

\[ x^2 - x = \prod \left( x - \alpha \right) \text{ for } \alpha \in \mathbb{F}_p \]
to get $K$ for $\mathbb{F}[z]$

Any $K$ for $\mathbb{F}[z]$ can get fractionation

Given fractionation $K \subset \mathbb{F} \{ \mathbb{F}(x) \}$

Field

Field

One key notion:

- function fields
- multivariate
- over rational

Based on findings all kinds of polynomials
\[ N \text{ is prime } \iff (x+4)^2 \equiv x+4 \pmod{N} \]

Key idea:

- Via Algebra II
- Deterministic algorithm for primality

2003: Agrawal, Kayal, Saxena

But no "proof" of primality

- randomized test (n) think algorithm to

\[ \begin{array}{c}
\text{Yes } [\text{Rabin, Miller, Solovay-Strassen}]\\
N \text{ is prime}
\end{array} \]

Given \( 0 \leq N < 2^r \), compute determinant of

Primality Testing
Key ingredient in algorithms: $O(\sum_{i=1}^{N}x_i)$

Accept if all tests accept:

```
end end
```

Verify $\forall (X,N)\in\operatorname{dom}N, x_i - 1$ for $v = 1 \ldots \lfloor \log_2(n) \rfloor$

```
for a = 1 \ldots \lfloor \log_2(n) \rfloor
do
```

First algorithm:

```
1. $\log^3 \approx 3$
```

1. Pick $\tilde{a}(X) = x_i - 1$ for $i = 1$ to $\lfloor \log_2(n) \rfloor$

2. Randomized. No proofs of probability.

```
(X+a)_N = x + a \pmod{N, \tilde{a}(X)}
```

At random. Verify

```
2. Vern: Pick $\tilde{a}(X)$ of degree $\approx \log(n)$
```

```
= \exp(n) \text{ time}
```

But how to choose density? Taker $N$ odd...
Key Idea: Solve "String Compression"

[Not fully but almost!]

\[ \text{Buba: 2012} \mid O(n \log n) \text{ time decomp.} \]

But still, aims: \( C \rightarrow \) in no polynomial algorithms known.

"Not believed to be "NP-hard"

"Not known to be in "NP"

\[
\forall \phi \in \{0,1\}^*, \exists \psi \in \{0,1\}^* \rightarrow (\phi,\psi) \in L \]

\[ M \equiv \phi \leq 1 \rightarrow \exists H \ni \phi \text{ homomorphic?} \]

\[ M = M \times M \]

\[ E \equiv \forall \psi \in \{0,1\}^* \]

\[ \text{Given: } E = (\forall, E) \]

Graph Isomorphism
Weil-Parity

\[ \text{Membership in permutation groups} \]

- New Algorithms in Group Theory
- Efficient Group Theory

First Part: String Homomorphism in Time \( O(n^{1.5}) \).

Every Group Homomorphism \( \approx \) String Homomorphism.

\[ \text{Given } A \text{ and } B, \text{ define } C \leq G \]