Rings and Integer Lattices in Computer Science

Notes from a series of lectures by

Manindra Agrawal
Indian Institute of Technology, Kanpur

The 2007 Barbados Workshop on Computational Complexity
February 25th – March 4th, 2007

Organizer:
Denis Thérien

Scribes:
Eric Allender
Victor Dalmau
László Egri
Faith Ellen
Valentine Kabanets
Antonina Kolokolova
Michal Koucký
Benoit Larose
Troy Lee
François Lemieux
Pierre McKenzie
Mark Mercer
Toni Pitassi
Mike Saks
Luc Segoufin
Pascal Tesson
Jacobo Torán
Sergey Yekhanin

Date:
July 31st, 2007

2007 Barbados Workshop on Computational Complexity	
RINGS AND INTEGER LATTICES IN COMPUTER SCIENCE	
Title: Rings	Lecture #: 1
Lecturer: Manindra Agrawal	Scribe: Faith Ellen and Toni Pitassi

1 Outline

- Finite Rings
  1. Definitions and the Complexity of Computing Ring Morphisms
  2. Applications of Finite Rings
     - polynomial factorization over finite fields
     - primality testing
     - identity testing
     - polynomial equivalence
     - graph isomorphism
- Integer Lattices
  1. Definitions and Computation on Short Vectors
  2. L^3 algorithm
  3. Hardness of Approximating the Shortest Vector Problem
  4. Factoring Polynomials over Q
  5. Integer Factoring
  6. Derandomizing Isolation Lemma
  7. Solving Modular Equation of Small Degree

2 Definitions

A *finite ring* $\mathcal{R} = (R, +, \cdot)$ consists of a finite set R and two operations: addition, $+$, and multiplication, $\cdot$. $(R, +)$ is an abelian group. $(R, \cdot)$ is a monoid. We use 1 to denote its multiplicative identity. In a *commutative ring*, multiplication is commutative. Unless otherwise specified, all rings are finite and commutative.

The notation $\mathcal{R} = \mathcal{R}_1 \otimes \mathcal{R}_2$ is used to denote that $\mathcal{R}$ is the direct sum of $\mathcal{R}_1$ and $\mathcal{R}_2$. Both $+$ and $\cdot$ are performed component-wise. For example, $\mathbb{Z}_6 = \mathbb{Z}_2 \otimes Z_3$. A commutative ring $\mathcal{R}$ is *local* (or *indecomposable*) if it cannot be written as a direct sum of two (smaller) rings.

Finite fields are finite local rings. Finite local rings are useful, because they behave like finite fields. Henrik Lenstra Jr. said, "One feels that any problem in mathematics can be hidden in a finite local ring." One example of a local finite ring that is not a field is $\mathbb{Z}_{p^2}$ for any prime p. Another example is $F[x_1, \ldots, x_n]/(x_1^{m_1}, \ldots, x_n^{m_n}, f_1, \ldots, f_k)$, where F is a finite field and $f_1, \ldots, f_k$ are polynomial functions of $x_1, \ldots, x_n$. Each x_i is nilpotent in this field. The unit of the field is $(x_1^0, \ldots, x_n^0)$.

A finite ring R can be decomposed into local rings, so that the direct sum of these local rings is isomorphic to R. This decomposition is unique, up to permutations.

For rings, we consider the following problems:

- RA: Given a finite ring, does it have a non-trivial automorphism?
- FRA: Given a finite ring, find a non-trivial automorphism.
- RI: Given two rings, are they isomorphic?
- FRI: Given two rings, find an isomorphism from one to the other?

3 Representations of a Finite Ring $(R, +, \cdot)$

1. **Table representation**: a list of all elements of R and the addition and multiplication tables for these elements.

2. **Basis representation**: a set of linearly independent generators $g_1, \ldots, g_k$ for $(R, +)$, the additive orders of the generators, (i.e. $d_i = \min\{d \in \mathbb{N}^+ \mid dg_i = 0\}$, and a multiplication table for the generators, where entry (i, j) of this table expresses the product of $g_i \cdot g_j$ as a linear combination of $g_1, \ldots, g_k$.

   Each element of the ring can be expressed in the form $a_1 g_1 + \cdots + a_k g_k$, where $a_i \in \{0, 1, \ldots, d_i - 1\}$ for $i = 1, \ldots, k$. Since the generators are linearly independent, each element of $(R, +)$ has a unique representation as a linear combination of the generators. Note that $+$ is performed component-wise: $(a_1 g_1 + \cdots + a_k g_k) + (b_1 g_1 + \cdots + b_k g_k) = c_1 g_1 + \cdots + c_k g_k$, where $c_i = a_i + b_i \mod d_i$ for $i = 1, \ldots, k$.

   The representation for $(R, +)$ as a set of linearly independent generators, together with their orders, is a decomposition of the abelian group $(R, +)$ as a direct sum of cyclic groups. Actually, the set of generators is not necessary for the representation of $(R, +)$. It suffices to have the multiset of the orders of the generators to completely specify $(R, +)$.

   There may be many different representations for the same abelian group $(R, +)$. For example, $\mathbb{Z}_6$ can be represented by the set $\{6\}$ or by the set $\{2, 3\}$. More generally, any composite order d can be replaced by the set of its maximal prime power divisors. The representation of $(R, +)$ as a multiset of prime power orders is unique. Unfortunately, knowledge of the factorization of the order of the group is needed, which may be computationally difficult to obtain.

3. **Polynomial representation**: a ring of polynomials over the integers modulo m for some integer $m > 1$, divided by the ideal generated by some collection of polynomials in this ring, i.e.

$$\mathbb{Z}_m[x_1, \ldots, x_n]/(f_1(x_1, \ldots, x_n), \ldots, f_k(x_1, \ldots, x_n)).$$

   Every finite ring can be represented in this form. This includes finite rings that are not local, such as $\mathbb{Z}_3[x]/(x^3 + x + 1)$. This representation can also be used for some infinite rings.

Examples:

1. One set of generators for the additive group of the ring $\mathbb{Z}_2[x, y]/(x^3, y^4, x^2 + y^3)$ consists of the elements $1, x, x^2, y, y^2, y^3, xy, xy^2, xy^3, x^2y, x^2y^2, x^2y^3$. Since $x^2 + y^3 = 0$, it follows that $y^3 = -x^2$, $xy^3 = -x^3 = -1$ and $x^2y^3 = -x^4 = -x$. Hence $1, x, x^2, y, y^2, xy, xy^2, x^2y, x^2y^2$ is a smaller list of generators.

2. The ring $R = \mathbb{Z}_2[x_1, \ldots, x_n]/(x_1^2, \ldots, x_n^2)$ has a basis of size 2^n, namely, all multilinear combinations of $x_1, \ldots, x_n$. When $n = 2$, this basis is $1, x_1, x_2, x_1x_2$. These generators are linearly independent. Thus R has 2^{2^n} elements, since the coefficient of each generator in a linear combination can be either 0 or 1. To represent an arbitrary element of R requires 2^n bits. Thus, the table representation of this ring uses $\Theta((2^{2^n})^2 2^n) = \Theta(2^{2^{n+1}+n})$ bits and the basis representation is exponentially more succinct.

4 The Complexity of the Ring Isomorphism Problem

The next three theorems together argue that the basis representation is the most interesting. Theorem 1 shows that using the most succinct representation, the polynomial representation, makes the isomorphism problem too hard. On the other hand, Theorem 2 shows that, using the table representation the ring isomorphism problem is fairly easy (in quasipolynomial time). Finally, Theorem 3 shows that, in the basis representation, the ring isomorphism problem has complexity similar to the graph isomorphism problem.

In this section, we will use the following notation. A ring R in basis representation form will be given by a vector $(d_1, d_2, \ldots, d_n)$, where each d_i is a prime power, and a matrix $(a_{ij}^k)_{1\le i,j,k\le n}$, where $0 \le a_{ij}^k \le d_k$ for all $1 \le i, j, k \le n$. This specifies a ring generated by n elements $e_1, \ldots e_n$ with each e_i having additive order d_i, and where the entries a_{ij} specify the result of multiplying e_i and e_j. That is, $e_i \cdot e_j = \sum_{k=1}^n a_{ij}^k e_k$.

Theorem 1. *The ring isomorphism problem for the polynomial representation is coNP-hard.*

Proof. Let $F = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ be a 3CNF formula over n variables, $x_1, \ldots, x_n$, where each C_i is a clause of size three. We convert each clause C_i of F to a polynomial as follows. Replace each positive variable x_i by the term $(1 - x_i)$; replace each negative variable $\overline{x}_i$ by the term x_i. The polynomial p_i corresponding to C_i is the product of these terms. For example, if $C_i = (x_1 \vee \overline{x}_2 \vee x_3)$, then the corresponding polynomial, p_i is $(1-x_1)x_2(1-x_3)$. Notice that for any assignment α, if α satisfies C_i, then $p_i(\alpha) = 0$, and if α falsifies C_i then $p_i(\alpha) = 1$.

Define I to be the ideal of $\mathbb{Z}_2[x_1, \ldots, x_n]$ generated by the following polynomials: p_j, for $j = 1, \ldots, m$, and $x_i^2 - x_i$, for $i = 1, \ldots, n$. We will show that F is unsatisfiable if and only if 1 is an element of I. The proof is a special case of Hilbert's Nullstellensatz, for the boolean setting. Note that the constant polynomial $1 \in I$ if and only if $\mathbb{Z}_2[x_1, \ldots, x_n]/I$ is isomorphic to the trivial ring consisting of the single element 0.

First suppose that F is satisfied by some assignment $\alpha \in \{0, 1\}^n$. Then $p_j(\alpha) = 0$ for $j = 1, \ldots, m$ and $\alpha_i^2 - \alpha_i = 0$ for $i = 1, \ldots, n$. Any element of the ideal I can be represented as $\sum_j P_j p_j + \sum_i Q_i(x_i^2 - x_i)$, where $P_j, Q_i \in \mathbb{Z}_2[x_1, \ldots, x_n]$ for $j = 1, \ldots, m$ and $i = 1, \ldots, n$. Under the assignment α, this polynomial has value 0, hence, it is not 1. Thus $1 \notin I$.

For the other direction, suppose that if F is unsatisfiable. We want to prove that there exist $P_j, Q_i \in \mathbb{Z}_2[x_1, \ldots, x_n]$ for $j = 1, \ldots, m$ and $i = 1, \ldots, n$ such that $\sum_j P_j p_j + \sum_i Q_i(x_i^2 - x_i) = 1$. Let ρ^k denote a boolean setting to the first k variables. Let $s^{i,b}$ denote the polynomial $x_i - b$. We will prove, by downward induction on k, for each partial assignment to the first k variables, ρ^k, that 1 is in the ideal generated by the following polynomials: (i) the polynomials p_j, $1 \le j \le m$; (ii) the polynomials $(x_i^2 - x_i)$, $1 \le i \le n$; and (iii) the polynomials $s^{i,b}$ where $1 \le i \le k$ and $b = \rho_i^k$ is the value of variable x_i as determined by ρ^k. Intuitively, the polynomials (iii) are extra constraints that say that the first k variables must be set according to ρ^k.

The base case is when $k = n$. Let ρ^n be some assignment to all n variables. Because F is unsatisfiable, there is some clause, C_i, and associated polynomial p_i such that $p_i(\rho^n) = 1$. Then 1 can be derived using p_i and the polynomials of type (iii). For example, suppose that ρ^n sets $C_i = (x_1 \vee \overline{x}_2 \vee x_3)$ to 0, i.e. $\rho^n = (0, 1, 0)$. Then $p_i + (x_2 - x_2x_3)s^{1,0} - s^{2,1} + (x_2)s^{3,0} = 1$, where $p_i = (1-x_1)x_2(1-x_3)$, $s^{1,0} = x_1$, $s^{2,1} = x_2 - 1$, and $s^{3,0} = x_3$.

For the induction step, assume the claim is true for $k + 1$ and we will prove that it is true for k. Let ρ^k be some assignment to the first k variables, and let $\rho^{k,0}$ be the assignment extending ρ^k by setting x_{k+1} to 0. Similarly define $\rho^{k,1}$. By the induction hypothesis, 1 can be derived from the original polynomials (i) and (ii) plus the polynomials s^{ℓ,b_ℓ}, $1 \le \ell \le k+1$, where $b_\ell = \rho_\ell^{k,1}$. Say,

$$1 = S_{k+1}^1 \cdot (x_{k+1} - 1) + \sum_{\ell=1}^{k} S_\ell^1 s^{\ell,b_\ell} + \sum_{j=1}^{m} P_j^1 p_j + \sum_{i=1}^{n} Q_i^1(x_i^2 - x_i). \tag{1}$$

Similarly, since $\rho_\ell^{k,0} = \rho_\ell^{k,1} = b_\ell$, for $\ell = 1, \ldots, k$,

$$1 = S_{k+1}^0 x_{k+1} + \sum_{\ell=1}^{k} S_\ell^0 s^{\ell,b_\ell} + \sum_{j=1}^{m} P_j^0 p_j + \sum_{i=1}^{n} Q_i^0(x_i^2 - x_i). \tag{2}$$

Then we can multiply both sides of (1) by $S^0_{k+1}x_{k+1}$, subtract $S^0_{k+1}x_{k+1}$ from both sides of (2), and add the results together to derive 1 from the desired polynomials:

$$\begin{aligned}
1 &= S^0_{k+1}x_{k+1} + 1 - S^0_{k+1}x_{k+1} \\
&= S^0_{k+1}x_{k+1}S^1_{k+1}(x_{k+1}-1) + S^0_{k+1}x_{k+1}\sum_{\ell=1}^{k} S^1_\ell s^{\ell,b_\ell} + S^0_{k+1}x_{k+1}\sum_{j=1}^{m} P^1_j p_j + S^0_{k+1}x_{k+1}\sum_{i=1}^{n} Q^1_i(x_i^2 - x_i) \\
&\quad + \sum_{\ell=1}^{k} S^0_\ell s^{\ell,b_\ell} + \sum_{j=1}^{m} P^0_j p_j + \sum_{i=1}^{n} Q^0_i(x_i^2 - x_i) \\
&= S^0_{k+1}S^1_{k+1}(x^2_{k+1} - x_{k+1}) + \sum_{\ell=1}^{k} S''_\ell s^{\ell,b_\ell} + \sum_{j=1}^{m} P''_j p_j + \sum_{i=1}^{n} Q''_i(x_i^2 - x_i),
\end{aligned}$$

where $S''_\ell = S^0_{k+1}x_{k+1}S^1_\ell + S^0_\ell$ for $\ell = 1, \ldots, k$, $P''_j = S^0_{k+1}x_{k+1}P^1_j + P^0_j$ for $j = 1, \ldots, m$, and $Q''_i = S^0_{k+1}x_{k+1}Q^1_i + Q^0_i$ for $i = 1, \ldots, n$. □

Theorem 2. *Ring isomorphism can be solved in time $n^{O(\log n)}$ in the table representation.*

Proof. Let R_1 and R_2 be the two rings in question, and assume that each has n elements and is given in table representation. Then the number of generators for the additive group of each ring is $O(\log n)$, because they have to be linearly independent. To check if the two rings are independent, first compute a set of $O(\log n)$ generators for the additive group of R_1. This takes time polynomial in n. Next, try all possible mappings of these generators to distinct elements of R_2. There are $n^{O(\log n)}$ such mappings. For each mapping, check whether it is an isomorphism by building the multiplication table for the image of the map and comparing it with the multiplication table for R_2. □

Theorem 3. *Ring isomorphism in the basis representation is in $NP \cap coAM$.*

Proof. We will first show that the ring isomorphism problem is in NP. Suppose R and R' are given in basis representation form. If the order of any of their generators are not prime powers, then, for each such generator g, we can guess a sequence of generators $g_1, \ldots, g_k$ with prime power orders to replace it. Checking that each of the generators g_i in the sequence is in the cyclic subgroup generated by g, checking that $g_1, \ldots, g_k$ are linearly independent, checking that each of their orders was guessed correctly, checking that their orders are relatively prime and their product is the order of g, and checking that g is in the subgroup generated by $g_1, \ldots, g_k$ can all be done in polynomial time.

So suppose that $f_1, \ldots, f_m$ is a sequence of generators for R with degrees $d_1, \ldots, d_m$, each of which is a prime power, and $h_1, \ldots, h_m$ is a sequence of generators for R' with degrees $d'_1, \ldots, d'_{m'}$, each of which is a prime power. Since the decomposition of a ring as a direct sum of local rings is unique up to permutations, it must be the case that $m = m'$ and $d'_1, \ldots, d'_m$ is a permutation of $d_1, \ldots, d_m$; otherwise R and R' are not isomorphic. This is easy to check in polynomial time.

For any prime p, let $R_p = \{r \in R \mid \text{the additive order of } r \text{ is a power of } p\}$ and let $R'_p = \{r \in R' \mid \text{the additive order of } r \text{ is a power of } p\}$. In other words, we will group together all d_i's that are powers of the same prime. Since d_i and d'_j are either relatively prime or powers of the same prime, it is easy to see that ϕ is an isomorphism from R to R' if and only if for all primes p, ϕ maps R_p to R'_p.

To check that R and R' are isomorphic, it suffices to guess an isomorphism ϕ from R to R' and check, for all primes p such that $R_p \neq \emptyset$, that ϕ is an isomorphism from R_p to R'_p. An isomorphism can be represented as a linear transformation of the generators of R. Guessing an isomorphism means guessing which element in R' each of the generators $f_1, \ldots, f_m$ map to. If $f_i \in R_p$, then it must map to an element of R'_p, which can be expressed as a linear combination $\sum\{c_{i,j}h_j \mid d'_j \text{ is a power of } p\}$ of the generators in $\{h_1, \ldots, h_m\} \cap R'_p$.

First, we want to check that ϕ is one-to-one. To do this it suffices to check that $det(C)$ is nonzero, where C is the $m \times m$ integer matrix describing the map ϕ. This matrix is block diagonal, so it suffices to compute the determinant or each block separately. Note that, in order to compute the determinant, it is necessary that we are in a field. This is the reason that we split up R and R' into R_p's and R'_p's.

Secondly, we check that for all $1 \leq i \leq m$, the additive order of each f_i is the same as that of $\phi(f_i)$. Lastly, we check that the mapping preserves multiplication. That is, we check that, for all $1 \leq i, j \leq m$,

$$\phi(f_i) \cdot \phi(f_j) = \sum_{k=1}^{m} a_{ij}^{k} \phi(f_k),$$

where $(a_{ij}^{k})_{1 \leq i,j,k \leq n}$ is the matrix that is given as input as part of the representation of R.

Next we sketch the proof that the ring isomorphism problem is in coAM. As before, the input is two rings R and R' in basis representation. Merlin's goal is to convince Arthur that R and R' are not isomorphic. Merlin first sends a sequence of linearly independent generators $f_1, \ldots, f_m$ and $h_1, \ldots, h_{m'}$ for R and R', respectively, and a sequence of their additive orders, each of which is a prime power. As above, Arthur checks that what Merlin sent is correct. If not, then Arthur rejects. Arthur also checks whether the additive groups $(R, +)$ and $(R', +)$ are isomorphic. If not, then Arthur knows that R and R' are not isomorphic rings.

Otherwise, let

$$\begin{aligned} S \quad = \quad & \{(B, \phi) \mid \exists \pi \in Aut(R, +), B \text{ is a multiplication table for the ring } \pi(R) \\ & \text{and } \phi \in Aut(R), \text{ described in terms of the basis } \{\pi(f_1), \ldots, \pi(f_m)\}\ \}. \end{aligned}$$

Note that $\{\pi(f_1), \ldots, \pi(f_m)\}$ is a basis for the additive group $(R, +)$, which is isomorphic to $(\pi(R), +)$, and, for all $1 \leq i, j \leq m$, $\pi(f_i) \cdot \pi(f_j) = \sum_{k=1}^{m} b_{ij}^{k} \pi(f_k)$. Similarly, let S' be the analogous set for R', obtained by replacing f_i by h_i, for $i = 1, \ldots, m$.

It can be shown that $\#S = \#Aut(R, +)$, which can be computed in polynomial time, since the orders of $f_1, \ldots, f_m$ are prime powers. Arthur can check whether $\#S = \#S'$ and, if not, he knows that R and R' are not isomorphic.

If R and R' are isomorphic, then $S = S'$, so $\#(S \cup S') = \#S$; otherwise, $S \cap S' = \emptyset$ and $\#(S \cup S') = \#S + \#S' = 2\#S$. Distinguishing between these two cases can be determined by an AM protocol, thus showing that ring isomorphism is in coAM.

The idea of the AM protocol is related to the Goldwasser Sipser protocol for graph non-isomorphism: Merlin sends Arthur a 1 to 1 function of the elements of $S \cup S'$ to a set of size $2\#S$. If R and R' are isomorphic, then half the points in the codomain have no preimage in $S \cup S'$. If R and R' are not isomorphic, then every point in the codomain has a preimage. Thus Arthur can select random points in the codomain and ask Merlin to send him their preimages in $S \cup S'$. If all of these are correct, then Arthur has good evidence that R and R' are not isomorphic. □

For example, consider the ring $R = \mathbb{Z}_2[x]/(x^2 + x + 1)$. This ring contains 4 elements: 0, 1, x, and $x + 1$. The set of elements $\{1, x\}$ forms a basis for $(R, +)$. Both have additive order 2. There are 6 automorphisms of $(R, +)$:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}.$$

The first of these is the identity. The second is the ring automorphism that maps 1 to 1 and x to $x + 1 = x^2$. The other four are not ring automorphisms, since they don't map 1 to 1.

Let $\pi_1, \ldots, \pi_6$ denote these automorphisms of $(R, +)$ and let $B_1, \ldots, B_6$ be the corresponding multiplication tables. Then $\pi_1(1) = \pi_2(1) = 1$, $\pi_3(1) = \pi_6(1) = x + 1$ and $\pi_4(1) = \pi_5(1) = x$. Also, $\pi_1(x) = \pi_3(x) = x$, $\pi_2(x) = \pi_4(x) = x + 1$ and $\pi_5(x) = \pi_6(x) = 1$.

Since $\pi_1(x) \cdot \pi_1(x) = x^2 = x + 1 = \pi_1(1) + \pi_1(x)$, $\pi_2(x) \cdot \pi_2(x) = (x + 1)^2 = x^2 + 1 = x = 1 + (x + 1) = \pi_2(1) + \pi_2(x)$, and $\pi_1(1) = 1 = \pi_2(1)$, it follows that B_1 and B_2 are both equal to

$$\begin{bmatrix} (1,0) & (0,1) \\ (0,1) & (1,1) \end{bmatrix}.$$

Similarly, $B_3 = B_4$ and $B_5 = B_6$. It follows that $\#S = 6$.

Finally, we prove that integer factoring randomly reduces to finding a ring isomorphism.

Theorem 4. *Integer factoring randomly reduces to FRI in expected polynomial time.*

Proof. Let n be an odd integer to be factored. Randomly pick $a \in Z_n - \{0\}$. If $gcd(a,n) \neq 1$, then it is a nontrivial factor of n. So, assume that $gcd(a,n) = 1$ and define $R_1 = Z_n[x]/(x^2-1)$, and $R_2 = Z_n[x]/(x^2-a^2)$. Note that $\{1,x\}$ is a basis for both $(R_1,+)$ and $(R_2,+)$. Suppose that ϕ is an isomorphism from R_1 to R_2. Then ϕ is determined by its action on 1. Let $\phi(x) = cx + d$. Because ϕ is an isomorphism, $c \neq 0 \bmod n$. If $gcd(c,n) \neq 1$, then it is a nontrivial factor of n. So, assume that $gcd(c,n) = 1$.

Since $x^2 - 1 = 0$ in R_1 and ϕ is a ring isomorphism, $0 = \phi(0) = \phi(x^2-1) = (\phi(x))^2 - 1$. Since $x^2 - a^2 = 0$ in R_2 and $\phi(x) = cx + d$, it follows that $(\phi(x))^2 = (cx+d)^2 = c^2x^2 + 2cdx + d^2 = 2cdx + (c^2a^2 + d^2)$. Therefore, $0 = 2cdx + (c^2a^2 + d^2 - 1)$, which implies that $2cd = 0 \bmod n$ and $c^2a^2 + d^2 = 1 \bmod n$. But $gcd(2,n) = 1 = gcd(c,n)$; therefore $d = 0 \bmod n$ and $c^2a^2 = 1 \bmod n$.

Since $gcd(a,n) = 1$, a has a multiplicative inverse in $\mathbb{Z}_n^*$. Call it b. It is easy to find using the standard algorithm for computing the gcd. Then both b and c are square roots of $b^2 \bmod n$.

Since a was chosen uniformly at random, b is equally likely to be any of the square roots of b^2. Since n is composite, it has at least four square roots. Thus, with probability at least 1/2, $b \neq c, -c$. In this case, neither $b-c$ nor $b+c$ is congruent to 0 modulo n, so $0 = b^2 - c^2 = (b-c)(b+c) \bmod n$, which gives a nontrivial factorization of n.

Hence, if we can find a ring isomorphism from R_1 to R_2 in polynomial time, we can find a nontrivial factorization of n and, hence, factor n in expected polynomial time. □

Finding an isomorphism between two graphs is equivalent to determining whether the two graphs are isomorphic. This follows from the self-reducibility of graph isomorphism. For rings, it is unknown whether finding an isomorphism (FRI) is harder than determining the existence of an isomorphism (RI). Determining the existence of a nontrivial automorphism is in P, so FRA is probably much harder than RA. Counting the number of isomorphisms from one ring to another is in $coNP \cap AM$.

Title: Some reductions and factorization of polynomials over finite fields — Lecture #: 2
Lecturer: Manindra Agrawal — Scribe: Benoit Larose

5 Introduction

Recall from the previous lecture the following notation:

- GI denotes the *Graph Isomorphism* problem: given two graphs G_1 and G_2, are they isomorphic ?
- RI denotes the *Ring Isomorphism* problem: given two finite rings R_1 and R_2, are they isomorphic ?
- RA denotes the *Ring Automorphism* problem: given a finite ring R, does it admit a non-trivial automorphism ?
- FRA denotes the problem of finding a non-trivial automorphism of a given ring R,
- IF denotes the problem *Integer Factoring*.

In the first part of this lecture, we shall see that (i) there is a many-one poly-time reduction of GI to RI, (ii) there exists a randomized poly-time reduction of IF to FRA, and (iii) RA is solvable in poly-time. In the second part we investigate the complexity of factorization of polynomials over finite fields.

6 Some reductions

Theorem 5. $GI \leq_m^p RI$.

Proof. Let $\mathbb{F}$ be any finite field not of characteristic 2; the proof shall make use of $\mathbb{F}$-algebras, which are local rings of characteristic p. Let G_1 and G_2 be two graphs on n vertices, which we may take to be $\{1, 2 \ldots, n\}$. We may suppose without loss of generality that these graphs are simple and undirected; in particular, we view the edges of each graph as unordered pairs of vertices. For convenience, we may also suppose that G_1 is *not* the disjoint union of a k-clique and an independent set of size $n-k$ (if it is, we can check if G_2 is also of this form, and we are done.)

Define two n-variable homogeneous polynomials of degree 2 as follows:

$$p_1(x_1, \ldots, x_n) = \sum_{\{i,j\} \in E(G_1)} x_i x_j$$

and

$$p_2(y_1, \ldots, y_n) = \sum_{\{i,j\} \in E(G_2)} y_i y_j .$$

Define a ring R_1 as follows:

$$R_1 = \mathbb{F}[x_1, \ldots, x_n] \,/ (x_i^2, x_i x_j x_k, p_1(x_1, \ldots, x_n)),$$

where $(x_i^2, x_i x_j x_k, p_1(x_1, \ldots, x_n))$ denotes the ideal of $\mathbb{F}[x_1, \ldots, x_n]$ generated by *all* x_i^2 (for all i), and *all* $x_i x_j x_k$ (for all i, j and k) and p_1. Similarly define the ring R_2 by

$$R_2 = \mathbb{F}[y_1, \ldots, y_n] \,/ (y_i^2, y_i y_j y_k, p_2(y_1, \ldots, y_n)) \ .$$

Claim 1. If $G_1 \simeq G_2$ then $R_1 \simeq R_2$.

This is clear: extend an isomorphism $\pi : G_1 \xrightarrow{\sim} G_2$ to an isomorphism $\phi : R_1 \xrightarrow{\sim} R_2$ by setting $\phi(x_i) = y_{\pi(i)}$ for all $1 \leq i \leq n$.

Claim 2. If $R_1 \simeq R_2$ then $G_1 \simeq G_2$.

Let $\phi : R_1 \xrightarrow{\sim} R_2$. Then ϕ satisfies the following properties: if

$$\phi(x_1) = c + \sum_i \alpha_i y_i + (\text{terms of degree} > 1)$$

then of course

$$0 = \phi(x_1^2) = c^2 + (\text{terms of degree} > 0)$$

and hence $c = 0$. Consequently

$$\phi(x_1) = \sum_i \alpha_i y_i + (\text{terms of degree} > 1);$$

then

$$0 = \phi(x_1^2) = 2 \sum_{i \neq j} \alpha_i \alpha_j y_i y_j = 2h(y_1, \ldots, y_n).$$

It follows that $h(y_1, \ldots, y_n) \in (y_i^2, y_i y_j y_k, p_2(y_1, \ldots, y_n))$. Since h has degree (at most) 2, and contains no y_i^2 there are only two cases:

- Case 1: the polynomial $p_2(y_1, \ldots, y_n)$ divides $h(y_1, \ldots, y_n)$. Since the polynomials are homogeneous of degree 2, it follows that

  $$\sum_{i \neq j} \alpha_i \alpha_j y_i y_j = \beta p_2(y_1, \ldots, y_n)$$

  for some $\beta \in \mathbb{F}$, and by a simple identity we deduce

  $$\beta p_2(y_1, \ldots, y_n) = \sum_{i \neq j} \alpha_i \alpha_j y_i y_j = \left(\sum_i \alpha_i y_i \right)^2 - \sum_i \alpha_i^2 y_i^2,$$

  which can only happen if the graph G_2 is the disjoint union of a clique and an independent set. Indeed, let $K = \{i : \alpha_i \neq 0\}$. Then for every $i \neq j$ in K, the term $\alpha_i \alpha_j y_i y_j$ is present in $h(y_1, \ldots, y_n)$ and hence in $p_2(y_1, \ldots, y_n)$ so $\{i, j\}$ is an edge of G_2; on the other hand every term $y_i y_j$ that appears has $i, j \in K$, so G_2 is the disjoint union of a clique on K and isolated vertices.

- Case 2: For every $i \neq j$, $\alpha_i \alpha_j = 0$. Then there are 2 possibilities:

  (a) $\alpha_i = 0$ for all i. Then this means that

  $$\phi(x_1) = \text{terms of degree} > 1.$$

  Since the argument of case 1 is also valid for $\phi(x_2)$, we get that

  $$\phi(x_1 x_2) = \phi(x_1)\phi(x_2) = (\text{terms of degree} > 1)(\text{terms of degree} \geq 1) = 0,$$

  i.e. $\phi(x_1 x_2) = 0$ in R_2. This means that $p_1(x_1, \ldots, x_n)$ divides $x_1 x_2$, and clearly it follows that $p_1(x_1, \ldots, x_n) = x_1 x_2$; this contradicts our hypothesis on G_1.

  (b) there exists precisely one α_i which is non-zero. We define a map π from G_1 to G_2 as follows: let $\pi(1) = i$ where α_i is the unique non-zero coefficient in the expansion of $\phi(x_1)$. Define $\pi(j)$ similarly for all $j > 2$ (notice that the arguments of case 1 and case 2 (a) are valid for any x_j).

  We must show that π is injective and that is it a graph homomorphism. First suppose that $\pi(i) = \pi(j)$, say $\phi(x_i) = \alpha_k y_k$ and $\phi(x_j) = \alpha'_k y_k$. Then

  $$\phi(x_i x_j) = \phi(x_i)\phi(x_j) = \alpha_k \alpha'_k y_k^2 = 0$$

and we obtain a contradiction as in case 2 (a). Hence π is a bijection.

Now we have that

$$\begin{aligned} 0 = \phi(p_1(x_1, \ldots, x_n)) &= \phi\left(\sum_{\{i,j\}\in E(G_1)} x_i x_j\right) \\ &= \sum_{\{i,j\}\in E(G_1)} \phi(x_i)\phi(x_j) \\ &= \sum_{\{i,j\}\in E(G_1)} \beta_{ij} y_{\pi(i)} y_{\pi(j)} \\ &= \beta p_2(y_1, \ldots, y_n) \end{aligned}$$

hence every $y_{\pi(i)}y_{\pi(j)}$ appears in $p_2(y_1, \ldots, y_n)$, which means that π does indeed preserve edges.

Notice finally that the number of basis elements of the ring $\mathbb{F}[x_1, \ldots, x_n] \big/ (x_i^2, x_i x_j x_k, p_1(x_1, \ldots, x_n))$ is $\binom{n}{2} - 1 = \mathcal{O}(n^2)$ and hence the size of the representation of our ring is polynomial in n. □

Theorem 6. *$IF \leq_r^p FRA$.*

Proof. Since the proof is slightly tricky we consider only the special case when $n = pq$ where p and q are distinct primes.

Pick a random degree 3 monic polynomial $f(x)$ over $\mathbb{Z}_n$. Let $R = \mathbb{Z}_n/(f(x))$. Then we have that

$$\mathbb{Z}_n/(f(x)) \simeq \mathbb{Z}_p/(f(x)) \otimes \mathbb{Z}_q/(f(x))$$

where the polynomial $f(x)$ is viewed with coefficients reduced modulo p or q respectively.

We want the polynomial f to satisfy the following properties: (i) f is irreducible over $\mathbb{Z}_p$ and (ii) f is reducible over $\mathbb{Z}_q$, and it is the product of a linear factor and an irreducible quadratic. It follows from the density of irreducible polynomials over $\mathbb{Z}$ that this will occur with probability $\approx 1/3 \times 1/3 = 1/9$.

Assume this occurs. Then we have that

$$R \simeq \mathbb{F}_{p^3} \otimes \mathbb{F}_{q^2} \otimes \mathbb{F}_q$$

where $\mathbb{F}_k$ denotes the field with k elements. Let ϕ be a non-trivial automorphism of the ring R; here we are using *FRA* as an oracle which returns the action of ϕ. Define

$$S = \{r \in R : \phi(r) = r\}.$$

This set can easily be computed: indeed, for a generic element $a_0 + a_1 x + a_2 x^2$ of R we have

$$\phi(a_0 + a_1 x + a_2 x^2) = a_0 + a_1 x + a_2 x^2 = a_0 + a_1 \phi(x) + a_2 \phi^2(x);$$

we may compute this, collect all terms of degree 0, 1, 2 and rewrite in the form $l_0 + l_1 x + l_2 x^2$. This yields a homogeneous system of linear equations over $\mathbb{Z}_n$.

Now notice that, since the automorphism ϕ must preserve the factors of the decomposition of R as a product of fields, the elements in each factor fixed by ϕ will constitute a subfield, hence:

- $|S| = pq^3 = nq^2$ if ϕ fixes all elements in $\mathbb{F}_{q^2}$ but not all elements in $\mathbb{F}_{p^3}$,
- $|S| = p^3 q^2 = n^2 q$ if ϕ fixes all elements in $\mathbb{F}_{p^3}$ but not all elements in $\mathbb{F}_{q^2}$,
- $|S| = pq^2 = nq$ if ϕ does not fix all elements in $\mathbb{F}_{p^3}$ nor all elements in $\mathbb{F}_{q^2}$.

In all three cases, dividing $|S|$ by n will yield a non-trivial factor of n. □

Remark. It should be remarked that $FRA \leq_r^p IF$ also holds so that $FRA \equiv_r^p IF$.

Theorem 7. $RA \in P$.

This result is a corollary of the following classification result. A ring is *rigid* if it admits no non-trivial automorphism.

Theorem 8. *(***Classification theorem for rigid rings***) Let R be a finite ring. Then R is rigid if and only if*

$$R = \bigotimes_{p \text{ prime}} R_p$$

where there exist $d_1 < \cdots < d_k$ such that

$$R_2 = \mathbb{Z}_{2^{d_1}} \otimes \cdots \otimes \mathbb{Z}_{2^{d_k}}$$

or

$$R_2 = \mathbb{Z}_{2^{d_0}}[x] \,/(x^2) \otimes \mathbb{Z}_{2^{d_1}} \otimes \cdots \otimes \mathbb{Z}_{2^{d_k}}$$

and for $p > 2$ there exist $e_1 < \cdots < e_l$ such that

$$R_p = \mathbb{Z}_{p^{e_1}} \otimes \cdots \otimes \mathbb{Z}_{p^{e_l}}.$$

Proof. (Overview) Clearly each ring in the list is rigid. For the converse: obviously the ring R is a direct sum of local rings; this decomposition is in fact unique. Hence it suffices to consider the case where R is a local ring. Then

$$R \simeq R_0[x_1, \ldots, x_n] \,/(x_1^{m_1}, \ldots, x_n^{m_n}, f_1(x_1, \ldots, x_n), \ldots, f_k(x_1, \ldots, x_n))$$

where

$$R_0 \simeq \mathbb{Z}_{p^m}[x] \,/(g(x))$$

for some polynomial g which is square-free and irreducible over $\mathbb{F}_p$.

Let M be the unique maximal ideal of R. Let t be the smallest positive integer such that $M^t = 0$. Define a map as follows: let $x_1 \mapsto x_1 + \alpha$ and $x_i \mapsto x_i$ for all $i > 1$, where α is a non-trivial element in M^{t-1}. Then this map induces a non-trivial automorphism of R. Hence if R is rigid it follows that $M = 0$ and hence $R = R_0$. Now if the degree of g is greater than 1, we may find a non-trivial automorphism of R_0 (unless the characteristic is 2.) It follows that the components of R have the desired form. □

7 Factorisation of polynomials over finite fields

Problem: given a finite field $\mathbb{F}$, and a polynomial $f(x) \in \mathbb{F}[x]$ of degree d, factor f over $\mathbb{F}$.

To simplify the discussion we shall let $\mathbb{F} = \mathbb{F}_p$ for p a prime. The time allowed for our computation is $poly(d \cdot \log p)$.

We may assume from the start that f is square-free, since otherwise we have that $\gcd(f(x), f'(x)) > 1$, where $f'(x)$ denotes the (formal) derivative of f. Hence by repeatedly computing this $\gcd$ we may reduce to the square-free case.

Next we shall show that we may concern ourselves only with the case where $f(x)$ is a product of irreducible factors *all of the same degree k*. Indeed, consider first the polynomial $x^p - x \in \mathbb{F}_p[x]$. Since $u^p = u$ for all $u \in \mathbb{F}_p$, this polynomial factors as

$$x^p - x = \prod_{\alpha \in \mathbb{F}_p} (x - \alpha)$$

and hence it follows that $\gcd(x^p - x, f(x)) > 1$ if and only $f(x)$ has an irreducible factor of degree 1. We may then use this $\gcd$ to factor out of $f(x)$ all irreducible factors of degree 1. Similarly, once we've factored out all irreducible factors of $f(x)$ of degree less than i, we have that $\gcd(x^{p^i} - x, f(x)) > 1$ if and only $f(x)$ has an irreducible factor of degree i (this follows from the fact that the elements of $\mathbb{F}_{p^i}$ are precisely the roots of the polynomial $x^{p^i} - x$.) Hence we may factor out all irreducible factors of degree i.

So now assume that there exists a $k > 1$ such that

$$f(x) = \prod_{i=1}^{m} g_i(x)$$

where every $g_i(x)$ is irreducible of degree k. We shall reduce the problem of factoring f to the case $k = 1$, i.e. to that of finding a root of a polynomial over $\mathbb{F}_p$.

Let $R = \mathbb{F}_p[x] / (f(x))$; then

$$R \simeq \bigotimes_{i=1}^{m} \mathbb{F}_{p^k},$$

and hence the map $\phi_1 : R \to R$ defined by $\phi_1(u) = u^p$ is an automorphism of R. Let

$$S = \{g(x) \in R : \phi_1(g(x)) = g(x)\}.$$

As we are considering polynomials with coefficients in a field, we may compute S in polynomial time, as it is in fact a linear subspace. Clearly there exist non-constant elements of R not in S (take for instance the element $(1, 0, 1, \dots) \in \bigotimes_{i=1}^{m} \mathbb{F}_{p^k}$), so let $e(x)$ be such an element. Here we view elements of R as polynomials (i.e. as representatives of elements of R.) We have that $e(x)$ is a constant modulo $g_i(x)$ for each i: indeed, viewed as an element of $\bigotimes_{i=1}^{m} \mathbb{F}_{p^k}$ it is of the form $(a_1, \dots, a_m)$ with $a_i \in \mathbb{F}_p$ since it is a fixed point of ϕ_1.

Consider the 2 variable polynomial $e(x) - y$. We are interested in the resultant of $e(x) - y$ and $f(x)$: we compute the determinant of $Res(e(x) - y, f(x))$, which will yield a polynomial $g(y)$ (simply view the entries of the resultant as elements in $\mathbb{F}_p[y]$). Then of course if γ is a root of $g(y)$ then the resultant will be zero, which implies that $e(x) - \gamma$ and $f(x)$ are *not* relatively prime, hence we may compute $\gcd(e(x) - \gamma, f(x))$ to obtain a factor of $f(x)$. Thus our problem reduces to finding a root of $g(x)$ over $\mathbb{F}_p$. In fact, we may use the same trick as earlier to remove all non-linear factors of $g(x)$, namely to compute

$$\gcd(g(y), y^p - y) = \widehat{g(y)}$$

and obviously both these polynomials have the same roots over $\mathbb{F}_p$; hence we may now consider the following situation:

Let

$$f(x) = \prod_{i=1}^{d} (x - \gamma_i)$$

where the γ_i are distinct elements of $\mathbb{F}_p$ (since $f(x)$ is square-free.) Our goal is to find a root of $f(x)$ over $\mathbb{F}_p$. This will require randomisation.

Choose a random $a \in \mathbb{F}_p$ and consider the polynomial

$$f(x^2 - a) = \prod_{i=1}^{d} (x^2 - a - \gamma_i).$$

Suppose the following holds:

- there exists an i such that $a - \gamma_i$ is a quadratic non-residue, and
- there exists a j such that $a - \gamma_j$ is a quadratic residue.

Note that this will occur with reasonable probability: indeed, notice that a is a root of the polynomial

$$(a - \gamma_1)^{\frac{p-1}{2}} - (a - \gamma_2)^{\frac{p-1}{2}}$$

with probability less than $1/2$ and hence the probability that $(a - \gamma_1)^{\frac{p-1}{2}} \neq (a - \gamma_2)^{\frac{p-1}{2}}$ is greater than $1/2$.

If this occurs then we may write

$$f(x^2 - a) = f_1(x^2 - a) f_2(x^2 - a)$$

where $f_1(x^2 - a)$ has only linear factors and $f_2(x^2 - a)$ has only degree 2 factors. As we did earlier, we can actually compute f_1 and f_2, using the gcd with the polynomials $x^p - x$ and $x^{p^2} - x$. Now in the polynomial $f_1(x^2 - a)$, every power of x is even, and by replacing each x^{2i} by x^i we obtain $f_1(x - a)$ from which we can easily compute $f_1(x)$, which is a factor of f since obviously $f(x) = f_1(x) f_2(x)$ by a simple substitution.

2007 Barbados Workshop on Computational Complexity

RINGS AND INTEGER LATTICES IN COMPUTER SCIENCE

Title: Primality testing and polynomial identity testing — Lecture #: 3
Lecturer: Manindra Agrawal — Scribe: François Lemieux and Pierre McKenzie

8 Introduction

In this lecture, we will first prove that primality testing is in P. This result is due to Agrawal, Kayal and Saxena ([1]). Here, we will stress the connection with ring automorphisms.

Recall Fermat's little theorem: If n is prime, then for all $a \in \mathbb{Z}_n$, $a^n \equiv a \mod n$. We can restate FLT as follows. Let $\Psi : \mathbb{Z}_n \to \mathbb{Z}_n$ map x to x^n.

Theorem 9 (Fermat)**.** *If n is prime then Ψ is an automorphism of $\mathbb{Z}_n$.*

It would be hard to argue that this restatement makes FLT more transparent. But what about the converse to FLT? This is where the above restatement will become helpful.

Fix a prime p such that $p|n$ (ie p divides n) and n is square-free. Then

$$\mathbb{Z}_n = F_p \otimes \cdots .$$

If Ψ is an automorphism of $\mathbb{Z}_n$ then it must fix F_p. In F_p, there is a unique automorphism, that is, $\Psi = \phi_1$, where $\phi_1(x) = x^p$ is called the *Frobenius map*. So we have

$$\begin{aligned}
&\forall a \in F_p,\ a^n = a^p \\
&\Rightarrow \forall a \in F_p \text{ except for } a = 0,\ a^{n-p} = 1 \\
&\Rightarrow n \equiv p \mod (p-1),
\end{aligned}$$

since the multiplicative group of F_p has $p - 1$ elements.

Now, instead of $\mathbb{Z}_n$, suppose that we looked at $R = \mathbb{Z}_n[y]/f(y)$ for some polynomial $f(y)$ chosen so that a large field sits once inside R. Suppose that

$$R = F_{p^k} \otimes \cdots .$$

It is known (see [21] pp.451-452) that the automorphisms of F_{p^k} are precisely the compositions of the Frobenius map $\phi_i(x) = x^{p^i}$ for $0 \leq i < k$. Then

$$\begin{aligned}
\Psi \text{ is an automorphism of } R &\Rightarrow \Psi \text{ is an automorphism of } F_{p^k} \\
&\Rightarrow \text{for some } i < k, \Psi = \phi_i \text{ in } F_{p^k} \\
&\Rightarrow \forall a \in F_{p^k}, a^n = a^{p^i} \\
&\Rightarrow \forall a \in F_{p^k} \text{ except for } a = 0,\ a^{n-p^i} = 1 \\
&\Rightarrow n \equiv p^i \mod (p^k - 1) \\
&\Rightarrow n = p^i \text{ provided that } n < p^k - 1.
\end{aligned}$$

Note that once we have ascertained that $n = p^i$ for some i, we can find p and i by using binary search.

9 Primality testing

There are two requirements to developing an efficient primality test for a number n along the lines outlined above:

1. Choose $f(y)$ such that $R = \mathbb{Z}_n[y]/f(y)$ contains a large enough field.
2. Efficiently test that Ψ defined by $\Psi(a) = a^n$ is an automorphism of R.

We begin with the second requirement. Suppose that

$$R = F \otimes \cdots$$

where char(F)=p. We need to test

a) $\Psi(a+b) = \Psi(a) + \Psi(b)$

b) $\Psi(ab) = \Psi(a) \cdot \Psi(b)$

c) Ψ is one-to-one and onto.

Note that **b)** holds since Ψ raises its argument to the nth power. Now if **a)** holds and $\Psi(a) = \Psi(b)$, then $\Psi(a-b) = 0$ and thus $(a-b)^n = 0$, which can be avoided by choosing R to have no nilpotent elements. Hence **c)** follows from **a)**.

So we now develop the linearity test required to ensure **a)**. We seek a test operating in time polynomial in $\log n$. Note that $|R| = n^{\deg f}$. Recall that F is our large field sitting inside R. We will do this in two steps:

Lemma 1 (Reduction Lemma 1)**.** *Let $S \subseteq F$ such that*

1. $\Psi(S) \subseteq S$

2. Ψ is linear in S

3. $|S| > n^{2\sqrt{t}}$

where t stands for the size of the set $G = \{\Psi^i \phi_1^j(y) \in F : i, j \geq 0\}$ (so $G = \{y^{n^i p^j} : i, j \geq 0\}$). Then $n = p^i$ for some i.

Proof. Since $|G| = t$, there exist α, β, γ and δ satisfying $0 \leq \alpha, \beta, \gamma, \delta \leq \sqrt{t}$, $(\alpha, \beta) \neq (\gamma, \delta)$, such that $\Psi^\alpha \phi_1^\beta(y) = \Psi^\gamma \phi_1^\delta(y)$. Consider $\Psi^\alpha \phi_1^\beta(a(y))$ for any polynomial $a(y) \in S$. Then

$$\begin{aligned} \Psi^\alpha \phi_1^\beta(a(y)) &= \phi_1^\beta \Psi^\alpha(a(y)) \\ &= \phi_1^\beta a(\Psi^\alpha(y)) \quad \text{by (2) and (1)} \\ &= a(\phi_1^\beta \Psi^\alpha(y)) \quad \text{by linearity of } \phi_1 \\ &= \phi_1^\delta \Psi^\gamma(a(y)). \end{aligned}$$

This implies in turn

$$\begin{aligned} &\forall a(y) \in S,\ ([a(y)]^{n^\alpha p^\beta} = [a(y)]^{n^\gamma p^\delta}) \\ &\Rightarrow \text{each } a(y) \in S \text{ is a root of the polynomial } x^{n^\alpha p^\beta} - x^{n^\gamma p^\delta} \text{ in } F \\ &\Rightarrow n^\alpha p^\beta = n^\gamma p^\delta \ \text{ (otherwise the poly of degree} \leq n^{2\sqrt{t}} \text{ has } |S| > n^{2\sqrt{t}} \text{ roots)} \\ &\Rightarrow n = p^i. \end{aligned}$$

∎

For our second step, let $T = \{\Psi^j(y) + i \in F \mid j \geq 1 \text{ and } 1 \leq i \leq 2\sqrt{t}\log n\}$ and let S be the multiplication closure of T.

Lemma 2 (Reduction Lemma 2)**.** *If Ψ is linear on T (i.e. $\Psi(a(y)) = a(\Psi(y))$) and $p > t > 4\log^2 n$ then*

1. $\Psi(S) \subseteq S$
2. *Ψ is linear on S*
3. $|S| > n^{2\sqrt{t}}$.

Proof. Let $a(y) \in S$. Then, we have $a(y) = b_1(y)\cdots b_l(y)$ where $b_i(y) \in T$.

$$\begin{array}{rcll} \Psi(a(y)) & = & \Psi(b_1(y))\cdots\Psi(b_l(y)) & (\Psi \text{ is an automorphism}) \\ & = & b_1(\Psi(y))\cdots b_l(\Psi(y)) & (\Psi \text{ is linear on } T) \\ & = & a(\Psi(y)). & \end{array}$$

Hence, Ψ is linear on S.

Now, if $b_1(y) = \Psi^m(y) + i$ then

$$\begin{array}{rcll} \Psi(b_1(y)) & = & \Psi^{m+1}(y) + \Psi(i) & \\ & = & \Psi^{m+1}(y) + i & (\text{since } \Psi(1) = 1). \end{array}$$

This implies that $\Psi(a(y)) \in S$.

To prove that $|S| > n^{2\sqrt{t}}$, define

$$Q = \{\prod_{j=1}^{t}(x + i_j) \mid 1 \le i_j \le 2\sqrt{t}\log n\}.$$

We have

$$|Q| = \binom{2\sqrt{t}\log n + t}{2\sqrt{t}\log n} \ge \binom{4\sqrt{t}\log n}{2\sqrt{t}\log n} > 2^{2\sqrt{t}\log n} = n^{2\sqrt{t}}.$$

We are going to show that S contains at least as many elements as Q. Observe first that if $q(x) \in Q$ then $q(y) \in S$. Suppose for a contradiction that $q(y) = g(y)$ for $q(x) \ne g(x)$ $(q(x), g(x) \in Q)$. Then

$$\begin{array}{rcl} q(\Psi^i\phi_1^j(y)) & = & \Psi^i\phi_1^j(q(y)) \\ & = & \Psi^i\phi_1^j(g(y)) \\ & = & g(\Psi^i\phi_1^j(y)). \end{array}$$

Hence, each element of G (defined in Reduction Lemma 1) is a root of the polynomial $q(x) - g(x)$ which has degree smaller than t. This contradicts the fact that $|G| = t$ and we must have $q(y) \ne g(y)$. This concludes the proof of the Size Reduction Lemma 2. $\square$

It remains to see how to choose $f(y)$, and thus R, in a way such that the following holds:

1. We can test efficiently that Ψ is linear on T.
2. $p > t > 4\log^2 n$.
3. R contains no idempotent.

Choose r such that the order of n modulo r is greater than $4\log^2 n$. Define the ring $R = Z_n[y]/(y^r - 1)$. We then have

$$4\log^2 n < t = |G| \le r$$

and

$$|T| \le 2r\sqrt{t}\log n \le 2r^{3/2}\log n.$$

The linearity of Ψ in T can thus be checked by testing all $|T|$ equations of the form $(y^j + i)^n = y^{jn} + i$ in R. The time required to do this depends on r. Note that we handle any $p \leq t$ by brute force, adding another dependence on r. To bound r, let

$$N = \prod_{i=1}^{4\log^2 n} (n^i - 1) \leq n^{16\log^4 n}.$$

Any prime r not dividing N will satisfy our requirement that the order of n at r exceeds $4\log^2 n$. As a rough estimate, N could have up to $\log N = 16\log^5 n$ distinct prime factors. Since there are approximately $k/\log k$ primes less than k, we are guaranteed to find a suitable $r \leq k$ when $k/\log k > 16\log^5 n$. Setting $k = \log^{5+\epsilon} n$ suffices.

Finally, we must show that R contains no idempotent. Assume the contrary and let $p(y) \in R$ be such that $p^k(y) \equiv 0 \pmod{y^r - 1}$ where $k \geq 2$. Hence, there exists a polynomial $q(y)$ such that $p(y)^k = q(y)(y^r - 1)$. It is known (e.g. see [21], Thm 17.44) that $t(y) = \frac{y^r - 1}{y - 1}$ is irreducible when r is prime. Since $k \neq 0$ then both $t(y)$ and $y - 1$ (and so $y^r - 1$) must be factors of $p(y)$.

The above bounds yield a primality test operating in time $O(\log^{11.5} n)$. Using bounds from analytic number theory, a finer analysis yields $O(\log^{7.5} n)$.

10 Polynomial identity testing

We will now consider another problem in BPP (actually co-RP), the *polynomial identity testing problem*, and discuss some consequences that would result from a proof that it also belongs to P. The problem is the following: Given a polynomial $P(x_1, \ldots, x_n)$ over a field F presented as an arithmetic circuit, test if the polynomial is identically 0.

Note that by "identically 0", we mean syntactically zero. For example, $x^2 - x$ is not identically zero, even over F_2.

(DeMillo-Lipton)-Schwartz-Zippel test [5, 20, 27]: *Randomly pick $a_1, \ldots a_n$ from F and evaluate $P(a_1, \ldots, a_n)$. If 0, output YES, otherwise, output NO.*

If $|F| > 2\deg P$ then the Schwartz-Zippel test succeeds with probability at least $1/2$. This test can be modified in many ways. Here are two examples.

Modification 1: *Choose $n + 1$ random polynomials in y of degree $2\log d$, where $d = \deg P$. Let these be $p_1(y), \ldots, p_n(y)$ and $q(y)$. Test whether the polynomial $P(p_1(y), \ldots, p_n(y))$ is 0 in the ring $F[y]/q(y)$.*

This test works over all fields F. Indeed, when q is irreducible (which happens with high probability), the p_i are random non-zero elements in $F[y]/q(y)$.

Modification 2: *Let $\hat{p}(y) = P(y, y^{d+1}, \ldots, y^{(d+1)^{n-1}})$. Test if $\hat{p}(y) = 0$ in the ring $F[y]/q(y)$ for a random $q(y)$ of degree $2n\log d$.*

This test works because most $q(y)$ will not divide $\hat{p}(y)$, as the least common multiple of all the polynomials of degree $2n\log d$ is much larger than d^n.

Question: Is there a polytime deterministic polynomial identity test?

A positive answer to this question would give an arithmetic circuit lower bound as we now show:

Theorem 10 ([9])**.** *Suppose that:*

a) *there is a polytime deterministic polynomial identity test over a field F, and*

b) *NEXP $\subseteq$ P/poly, and*

c) *the permanent has polynomial size arithmetic circuits over F.*

*This is a contradiction. In particular, assuming **a)**, either **b)** fails or **c)** fails.*

Proof. If NEXP $\subseteq$ P/poly then NEXP $=$ MA $\subseteq$ PH by a result of Impagliazzo and al. ([8]). Moreover, Valiant [23] has shown that the 0-1 Permanent is #P-complete and by a theorem of Toda [22] we have $PH \subseteq \mathrm{P}^{\#\mathrm{P}}$. Hence, if NEXP $\subseteq$ P/poly then computing the bits of the permanent of an integer matrix is NEXP-hard

Suppose now that **a)**, **b)** and **c)** hold. Then an NP algorithm computes the bits of the permanent of a matrix $M \in \mathbb{Z}^{n \times n}$. This is a contradiction since it implies that NEXP $\subseteq$ NP. Here is the algorithm:

1. Guess an arithmetic circuit $C_n(x_{1,1}, \ldots, x_{n,n})$.
2. Verify that C_n computes the $n \times n$ permanent.
3. Use C_n to compute the permanent of M.

Given our hypotheses, this algorithm runs in polytime, provided that the verification step 2 does. It remains to explain this step.

Suppose that C_n is correct. Then, by setting appropriate variables to 1 and 0, we can extract from C_n the arithmetic circuits $C_{n-1}, C_{n-2}, \ldots, C_1$ respectively computing the permanent of $(n-1) \times (n-1)$, $(n-2) \times (n-2), \ldots, 1 \times 1$ matrices. Step 2 consists in using the polynomial identity test to verify that the n equations with indeterminates

$$\begin{aligned} C_n(x_{1,1}, \ldots, x_{n,n}) = {} & x_{1,1} \cdot C_{n-1}(\overline{x} \text{ without row 1 and column } 1) + \\ & x_{1,2} \cdot C_{n-1}(\overline{x} \text{ without row 1 and column } 2) + \\ & \vdots \\ & x_{1,n} \cdot C_{n-1}(\overline{x} \text{ without row 1 and column } n) \end{aligned}$$

and

$$\begin{aligned} C_{n-1}(x_{1,1}, \ldots, x_{n-1,n-1}) = {} & x_{1,1} \cdot C_{n-2}(\overline{x} \text{ without row 1 and column } 1) + \\ & x_{1,2} \cdot C_{n-2}(\overline{x} \text{ without row 1 and column } 2) + \\ & \vdots \\ & x_{1,n-1} \cdot C_{n-2}(\overline{x} \text{ without row 1 and column } n-1) \end{aligned}$$

and so on, down to

$$C_2(x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}) = x_{1,1} \cdot C_1(x_{2,2}) \; + \; x_{1,2} \cdot C_1(x_{2,1})$$

and finally

$$C_1(x_{1,1}) = x_{1,1},$$

are identically satisfied. This is obviously the case if indeed C_n was correct. An induction shows that conversely, any sequence C_n, $C_{n-1}, \ldots C_2, C_1$ of polynomials that identically satisfy the above equations has the property that C_n computes the permanent. $\square$

11 Introduction

In this lecture we present a deterministic algorithm from Kayal and Saxena [10] for testing if an arithmetic circuit is identically zero (for a limited class of depth three circuits).

Then we introduce arithmetic branching programs, and we present a conjecture that would imply that the permanent cannot be computed by small arithmetic branching programs. Finally, we discuss some reasons why the conjecture is plausible.

12 Identity testing

Theorem 11. *There is a deterministic algorithm that tests if a given $\Sigma\Pi\Sigma$ (sums of products of sums) arithmetic circuit C over a field F with n inputs, top fan-in k, and multiplication fan-in $d \geq n$ is identically zero. The running time of the algorithm is polynomial in d^k.*

Proof. (sketch) The output of C can be looked at as

$$C = T_1 + T_2 + \ldots T_k$$

where $T_i = \Pi_{j=1}^{d} L_{i,j}$ with each $L_{i,j}$ being a linear form $L_{i,j} = \sum_{m=1}^{n_{i,j}} \alpha_{i,j,m} x_m + \beta_{i,j}$ with $\alpha_{i,j,m}$ and $\beta_{i,j}$ in F. W.l.o.g. (i.e., by introducing a new variable, if need be) we can consider the $\beta_{i,j}$ to be 0.

The basic idea in the algorithm for testing whether C is identically zero is to make a series of recursive calls for proving that other circuits are zero. In each call the top fan-in of the addition gate decreases while the underlying ring (initially F) becomes larger. The dimension of the base ring increases by a factor of at most d in each recursive call.

First we make an invertible linear transformation on the variables. Suppose for example that $T_1 = l_{1,1}^7 l_{1,2}^5 l_{1,3} l_{1,4}^2$ where the l_i, j's are linear forms in $F[x_1, \ldots, x_n]$. We make the linear transformation $T_1 = x_1^7 \hat{l}_{1,2}^5 \hat{l}_{1,3} \hat{l}_{1,4}^2$ and compute the polynomial modulo x_1^7. This will send T_1 to 0 and $C(\bmod x_1^7) = \hat{T}_2 + \cdots + \hat{T}_k$.

Initially we start with the field $F[x_1, \ldots, x_n]$. With the linear transformation and going mod x_1^7 we transform the field into $(F[x_1]/x_1^7)[x_2, \ldots, x_n]$. Observe that $F[x_1]/x_1^7$ is a local ring R_1. If we work with $l_{1,2}^5$ instead, we obtain a ring R_2, etc. If the transformed expressions turn out to be 0 in each of the rings R_1, R_2, R_3, R_4, then we can conclude that C evaluates to 0 over F, by Chinese Remaindering.

At this level $\hat{T}_2$ could for example look like $\hat{T}_2 = (2x_1 + 4x_2 + 7x_3)(8x_1 + 4x_2 + 7x_3)$. Again we apply an invertible linear transformation to transform $4x_2 + 7x_3$ in x_2. $C(\bmod (2x_1 + x_2)(8x_1 + x_2)) = T_3 + \cdots + T_k$. We obtain in this way a new ring $R_1' = R_1[x_2]/((2x_1 + x_2), (8x_1 + x_2))$.

The recursion tree built in this way, with nodes labeled by rings, has depth $k - 1$ and the fan-out of each node is at most d. At the leaf level there is just one term and it is possible to check deterministically if it is zero or not. $\square$

This method only works for $\Sigma\Pi\Sigma$ circuits of the described form.

13 Arithmetic Branching Programs

Arithmetic branching programs, introduced by Nisan [18], are a generalization of arithmetic formulae. (Some authors call them "algebraic branching programs".) They provide a model of computation that captures the complexity of computing determinants.

Formally, an arithmetic branching program is a directed acyclic graph with a single source node s and a single sink node t. Each edge is labeled either with an integer constant, or with a variable. Inductively, we associate a polynomial with each node in the graph, as follows. The source node s is associated with the constant polynomial 1. If a node v has edges into it from vertices $w_1, \ldots w_k$, and these edges are labeled by literals $l_1, \ldots l_k$ respectively, and if the vertices $w_1, \ldots w_k$ are associated with polynomials $p_1, \ldots, p_k$, then v is associated with the polynomial $\sum_i l_i p_i$. (Intuitively, the output node is associated with the polynomial obtained by taking all of the paths from s to t, multiplying the literals labeling the edges along these paths, and then summing these terms.)

Fact: A polynomial is computed by an arithmetic branching program of size s if and only if it can be expressed as the determinant of a matrix of size $s^{O(1)}$ with entries consisting of literals (constants and variables). (This equivalence is due to Vinay, Toda, and Damm. For references and a discussion of the history, see [14, 3].)

It follows by work of Valiant [24] that an arithmetic circuit of size s and degree d can be simulated by an arithmetic branching program of size $s^{O(\log d)}$. It is also known that any arithmetic circuit for the permanent can be simulated by an arithmetic circuit of polynomial degree (by splitting each gate into $n+1$ gates, each computing the terms of degree j, for $j \leq n$). Thus if one can prove that the permanent requires arithmetic branching programs of exponential size, it follows that the permanent requires arithmetic circuits of exponential size.

Conjecture A**:** Let A be an arithmetic branching program of size s computing the polynomial $p(x_1, \ldots, x_n)$. Then $p = 0$ if and only if for every r such that $1 \leq r \leq s^4$, $p(y, y^{s+1}, y^{(s+1)^2}, \ldots, y^{(s+1)^{n-1}})$ is zero mod $y^r - 1$.

Note that if Conjecture A is true, then there is a deterministic polynomial-time (even NC^2) algorithm to to solve identity testing for arithmetic branching programs. This would also imply a uniform NC^2 algorithm for computing perfect matchings, by [13].

Theorem 12. *If Conjecture A is true then computing the permanent requires exponential sized arithmetic branching programs.*

Proof. **Stage 1:** Constructing a hard-to-compute polynomial R. Define $Q(y) = \prod_{r=1}^{s^4}(y^r - 1)$.

Now define R to be a multilinear polynomial with m variables (where m is roughly $\log s$), where we require that

$$R(y, y^2, \ldots, y^{2^{m-1}}) = Q(y).$$

Because R is multilinear, we have

$$R(x_1, \ldots, x_n) = \sum_{S \subseteq [1 \ldots m]} c_S \prod_{i \in S} x_i.$$

Thus

$$R(y, y^2, \ldots, y^{2^{m-1}}) = \sum_{S \subseteq [1 \ldots m]} c_S y^{\sum_{i \in S} 2^{i-1}}.$$

To complete the definition of R, we pick $m = 8 \log s$, and pick the coefficients c_S to match the corresponding term in Q.

We claim that any arithmetic branching program computing R has size at least s (assuming Conjecture A).

To see that this claim is true, assume for the sake of a contradiction that C is an arithmetic beaching program for R having size at most s. By Conjecture A, R is identically zero if and only if the univariate polynomial $R(y, y^2, \ldots, y^{2^{m-1}})$ is identically zero mod $y^r - 1$ for $1 \leq r \leq s^4$. This happens only if $Q(y)$ is zero mod $y^r - 1$ for $1 \leq r \leq s^4$. However, by definition of Q, it really is zero mod $y^r - 1$ for $1 \leq r \leq s^4$. Thus this implies that R really is identically zero. But this contradicts the fact that R, by definition, is *not* identically zero. Thus the claim is true.

Stage 2: Computing R as a permanent.

$$R(x_1, \ldots, x_n) = \sum_{S \subseteq [1 \ldots m]} c_S \prod_{i \in S} x_i.$$

On input $x_1, \ldots, x_n$, we want to compute $R(x_1, \ldots, x_n)$ as a #P function. Since the permanent is hard for #P the results follows. R is a sum of small products and this can easily be expressed as a #P function. The only problem is to determine the coefficients c_S or to show that each of them can be computed as a #P function. We have

$$R(y, y^2, \ldots, y^{2^{m-1}}) = Q(y) = \sum_{S \subseteq [1 \ldots m]} c_S y^{\sum_{i \in S} 2^{i-1}} \text{ and } Q(y) = \prod_{r=1}^{s^4} (y^r - 1).$$

We consider first the easier case in which Q is an infinite product

$$Q_\infty(y) = \prod_{r=1}^{\infty} (1 - y^r).$$

Claim 1.

$$Q_\infty(y) = \sum_{j=0}^{\infty} \alpha_j y^j$$

where each of the coefficients α_j is either 1,-1 or 0.

Proof. (of the claim). Consider the coefficient of i^j in the infinite multiplication. Each term producing y^j is determined by a set of factors in the product in which the sum of the exponents of the y add exactly to j. This can be encoded by a finite sequence $l_1, \ldots, l_k$ with $\sum_i l_i = j$ and in which all the l_i's are different. The contribution of this term to the coefficient of y^j is either 1 or -1 depending on whether k is even or odd. We want to make a correspondence between sequences contributing 1 and sequences contributing -1 in order to cancel them. For this we represent each sequence $l_1 > l_2 > \cdots > l_k$ in triangle form as in Figure 1.

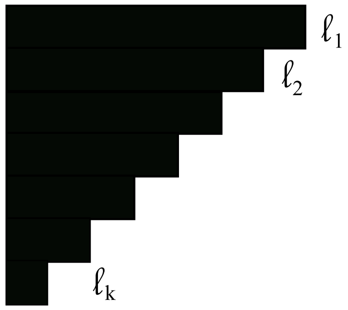


Figure 1

Consider the top diagonal defined starting by the largest element l_1 and the following elements $l_2, \ldots, l_d$, where l_d is the first element satisfying $l_d > l_{d+1} + 1$ (observe that d can be equal to 1). We compare d with the smallest element in the sequence l_k. If $d \geq l_k$ then we match the sequence $l_1, \ldots, l_d, \ldots, l_k$ with the sequence $l_1 + 1, \ldots, l_{l_k} + 1, l_{l_k+1} \ldots, l_d, \ldots, l_{k-1}$. Graphically this corresponds to taking the last row and placing it along the top diagonal. Observe that both sequences add up to j but the second has one less element. Therefore their contribution to the coefficient of y^j is in one case 1 and in the other -1, and they cancel each other. If $d < l_k$ then $l_1, \ldots, l_d, \ldots, l_k$ is matched to $l_1 - 1, \ldots, l_d - 1, l_{d+1} \ldots, l_k, l_{k+1}$ with $l_{k+1} = d$. This is the dual case. The only situation in which such a correspondence is not possible is when $d = k = l_k$. In this case the top diagonal cannot be placed as the bottom row in the figure since then the last row would be longer than the one before last. It is not hard to see that in this case $l_1 = 2d - 1$ and j has the form $j = \frac{3}{2}d^2 - \frac{1}{2}d$. This is called a pentagonal number. Summarizing, the coefficient of y^j is always 0 except when j is a pentagonal number $j = \frac{3}{2}d^2 - \frac{1}{2}d$. In this case α_j is 1 when d is even, and -1 otherwise. □

In the finite case it is harder to obtain the coefficients of $Q(y)$ since the product goes only up to s^4 and sequences with $l_1 = s^4$ and $d \geq l_k$ would be matched by the method with sequences having $l_1 = s^4 + 1$ and therefore not being considered in the product.

□

In the Barbados lecture, an approach was sketched, claiming to show that c_S is computable in #P. However, this approach turned out to have some bugs in it.

We give now a possible approach to prove A: Consider an arithmetic BP computing a polynomial over a field F. We force several normalizing conditions imposing structure on the BP. This can be done with every arithmetic branching program, implying at most a polynomial increase in its size. For the visualization of the branching program it helps to consider that every node in the BP represents a polynomial.

Normalizing conditions of the branching program:

i) it has one source and one sink. The source represents the polynomial 1.

ii) it is leveled in layers and the arrows go only from one layer to the next. The width of each layer is exactly w. (This means that we add extra "dummy" vertices in the layer containing the source node; these nodes are each labeled with the zero polynomial.) We can represent the BP with the edges pointing from left to right as in Figure 2. Each layer has exactly w nodes placed on different rows. (The dummy vertices are not pictured, but they should be viewed as being above the source node, so that the source node is on the bottom row.)

iii) it alternates linear and non-linear layers. In a linear layer all the edges are labeled with multiplication constants. The polynomial represented by the nodes direct after a linear layer are a linear combination of the polynomials represented by the nodes in the previous layer. The linear combination can be captured by a $w \times w$ matrix. We require this matrix to be invertible. A non-linear layer consists of w edges marked with the constant 1 pointing from each node to the one in the same row at the next layer, plus one more edge labeled with a variable. This edge must point to a row upwards. We also require that the edge marked with the variable points to a node computing the zero polynomial.

iv) the last layer is a linear one.

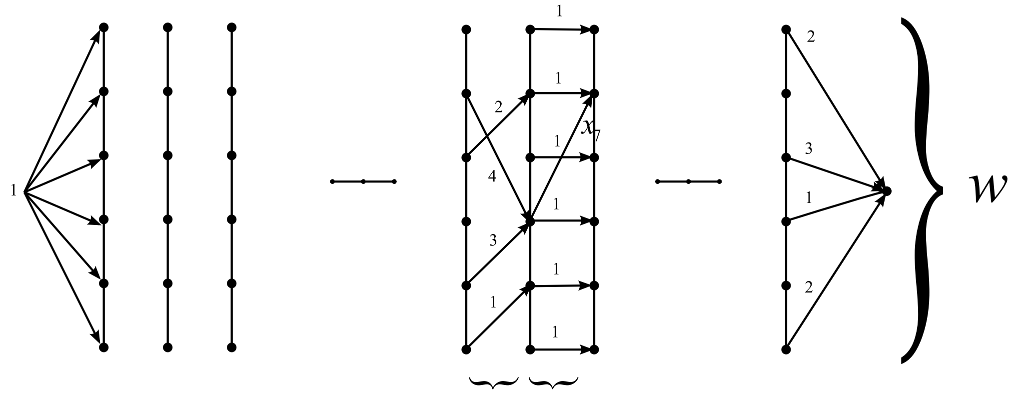


Figure 2

For each layer l let v_l^t be the vector of coefficients of monomial t in the polynomials computed by the nodes at this layer. Since there are exponentially-many terms t, this means that there are exponentially many vectors v_l^t associated with each layer l. $v_l^t \in F^w$. Suppose the BP has D layers. The polynomial computed by it is zero if and only if for all t $\rho_D^T \cdot v_D^t = 0$, where ρ_D is the vector of coefficients of the polynomials pointing to the sink.

For $1 \leq l \leq D$, let V_l be the space spanned by the vectors v_l^t. We have dim $V_l \leq w$. We look now at the univariate version of the polynomial. Given any term t, we now substitute $x_1, \ldots, x_n$ by $y, y^{s+1}, \ldots, y^{2^{n-1}}$. This gives us the same set of vectors, but now they are labeled by univariate powers of y. Now, we add all of these vectors that are in the same equivalence class modulo $(y^r - 1)$ for $1 \leq r \leq s^4$. Let U_l be the space spanned by the vectors at layer l after the reduction mod $(y^r - 1)$. Note that there probably are fewer vectors in U_l than in V_l, since there are now only r terms, each representing an equivalence class of the original vectors t. Clearly, $U_l \subseteq V_l$. It would suffice to prove the theorem, if we could show that for all l, there is some r such that dim $U_l =$ dim V_l.

Let us consider how V_l evolves when l goes from 1 to D. Initially $V_1 = (0,0,\ldots,0,1)^T$ since, in the first layer, the source node is labeled 1, and all of the dummy vertices are labeled 0.

In a linear layer, $v_{l+1}^t = Av_l^t$ for all t. (A is the matrix representing the linear transformation at the layer). Therefore $V_{l+1} = AV_l$ and the dimension of the space does not change.

In a non-linear layer with an edge pointing from row k to row $m, (m < k)$, marked with variable x_i (that is then substituted by y^α for some α, $v_{l+1} = v_l^t + (e_k^T \cdot v_l^{t-\alpha})e_m$ where $e_k^T = (0,0,\ldots,1,\ldots,0)$ (a 1 at the k-th position).

We have the following observations:

i) dim $V_l \leq$ dim V_{l+1}.

ii) dim $V_l \geq$ dim $V_{l+1} - 1$.

iii) If dim $V_l =$ dim V_{l+1} then $V_{l+1} = BV_l$ for some invertible B.

iv) If dim $V_{l+1} =$ dim $V_l + 1$ then $e_m \in V_{l+1}$ implies $V_{l+1} = V_l + e_m$.

We look at how the $U_l's$ evolve:

i) $U_1 = V_1$.

ii) If $U_l = V_l$ and $l \to l+1$ is a linear layer then $U_{l+1} = V_{l+1}$.

iii) If $U_l = V_l$ and $l \to l+1$ is a non-linear layer then

- dim $U_{l+1} \geq$ dim U_l and
- $e_m \in U_{l+1}$ if and only if $U_{l+1} = V_{l+1}$.

The idea in order to prove the conjecture is to show that assuming $U_l = V_l$, for most r's $e_m \in U_{l+1}$. One would have to prove that the property holds for at least one r for all layers.

14 Polynomial equivalence

A polynomial is of *cubic form* if it is homogeneous and of degree 3 over n variables. The problem of polynomial equivalence is that given two cubic forms $p(\overline{x})$, $q(\overline{x})$, is there an invertible matrix A such that $p(\overline{x}) = q(\overline{x}A)$. In the quadratic case, this can be done in PTime, in the cubic case this is open.

This lecture has been canceled by lack of time. The plan of this lecture was to show that:

Theorem 13. *Testing equivalence of cubic form polynomials is the same as testing isomorphism of F-algebras.*

The proof of this theorem can be found in Nitin's thesis.

Remark: A F*-algebra* is a special case of rings where every element has the same characteristic. The proof given in the first lecture actually showed that **Graph Isomorphism** reduces to testing isomorphism of F-algebras.

15 Graph Isomorphism

This lecture will present some tentative ideas for a proof that **Graph Isomorphism** is in PTime. We will actually only consider the problem of **Graph Automorphism**: i.e. finding whether a given graph has a non-trivial automorphism. The method proposed works in two steps. In the first step, a method for coloring graphs is presented. This method is essentially the 2-dimensional Weisfeiler-Lehman algorithm [26] (abbreviated as 2-dim WL in the sequel). The colors induced by the 2-dim WL algorithm are preserved by any automorphism of the graph. The 2-dim WL algorithm works in PTime but fails to solve the graph automorphism problem: i.e. there exists rigid graphs such that their rigidity cannot be confirmed by the 2-dim WL algorithm. In the second step of the proposed method, an algebraic structure is exhibited which will hopefully help to refine the coloring.

Let $G = (V, E)$ be a loop-free graph with $|V| = n$. We view G as a complete colored graph with three colors, one for vertices, one for edges, and one for non-edges. More precisely, G is presented as a triple of $n \times n$ matrices $(I, E, \bar{E})$, each with entries in $\{0, 1\}$, such that I is the identity matrix, E is the matrix such that $E_{i,j} = 1$ iff there is an edge from vertex i to vertex j in G and $\bar{E}_{i,j} = 1$ iff there is no edge from vertex i to vertex j in G.

We view $I, E, \bar{E}$ as colors for G. Our goal is to refine those colors until the colors give enough information to solve the graph automorphism problem.

With this presentation, we can view any automorphism P of G as an $n \times n$ matrix. Any such automorphism will commute with the colors I, E, $\bar{E}$.

15.1 Refinement algorithm: 2-dim WL

A set of colors $\{C_1, \ldots, C_m\}$ is said to be *well-behaved* for G if

1. Each C_i is an $n \times n$ $0/1$ matrix (the entries are in $\{0, 1\}$).
2. The colors partition the graph: $\Sigma_{i=1}^{m} C_i = J$ where J is the $n \times n$ matrix with all entries set to 1.
3. The colors are preserved by all automorphisms of G, i.e. for any automorphism P of G, and all $1 \leq i \leq m$, we have $PC_i = C_iP$.

It is clear that the presentation of G as $(I, E, \bar{E})$ is always a well-behaved coloring for G. The goal of the first step is to refine this initial coloring in polynomial time. Assume that we are given a well-behaved coloring $\{C_1, \ldots, C_m\}$ for G. If for any i, j we have that $C_iC_j = \sum_{k=1}^{m} kC_{\alpha_k}$ for some set $\{\alpha_k\}$ of color indices, then the algorithm stops and output the current set of colors.

In the other case, there exists i, j such that $C_iC_j = \sum_{k=1}^{m} kD_k$, where each D_k is a $0/1$ matrix.

Claim 2. *For all automorphisms P of G and all k, we have $PD_k = D_kP$.*

Proof. By well-behavedness we have $PC_iC_j = C_iC_jP$ therefore $\sum kPD_k = \sum kD_kP$. This implies the result by identifying on both sides the entries whose values are k for each k. □

Add all D_k to the set of colors. This yields a new set of colors $\{C'_1, \ldots, C'_l\}$. Note that from the observations above, this set of colors satisfy conditions 1 and 3 of well-behavedness . If $\sum_i C'_i = J$ then we have a new set of well-behaved colors and we restart the refining process as above. Otherwise, take some pair C'_i and C'_j which share some ones. Let $D'_1 = C'_i \backslash C'_j$, $D'_2 = C'_j \backslash C'_i$, and $D'_3 = C'_i \cap C'_j$. We then remove C'_i and C'_j from the set of colors and replace them by D'_1, D'_2, D'_3.

Claim 3. *For all automorphisms P of G and all $k \in \{1, 2, 3\}$ we have $PD'_k = D'_kP$.*

Proof. Let P be an automorphism of G. Consider an edge (i, j) which is in D'_3. By definition it is in C'_i and in C'_j. As P is preserved by C'_i and C'_j, the image of (i, j) by P is in C'_i and in C'_j and therefore in D'_3. A similar argument shows that if the image of an edge (i, j) is in D'_3 then (i, j) is in D'_3. Therefore, P preserves D'_3. As P preserves C'_i and D'_3, it must preserve D'_1. Similarly, it preserves D'_2. □

We repeat this last step until condition 2 of well-behavedness holds.

This completes the presentation of the first step. Note that this process must converge in PTime because the set of colors $\{C_1, \ldots, C_k\}$ can have at most n^2 elements (when each edge and vertex have their own color).

Remarks: The set coloring obtained by this algorithm is independent of the choices made during the intermediary stages. This is because there is a graph characterization for the colors obtained by the algorithm [4]. Observe that it is possible to give a canonical name to all the colors created by this algorithm. We assume such a canonical naming and we fix an arbitrary order on the names. Given a colored graph G, the multiset of colors of G is the array of color names followed by the cardinalities of each color, ordered by their name. Suppose that we run this process for two graphs G_1 and G_2 in parallel, mimicking the operations on both sides. If at any point we generate different colors on one side compared to the other, or we find that there exists a color which has different cardinalities in G_1 and G_2, then we can safely conclude that the two graphs are not isomorphic. The opposite is not true in general (see again [4] for a thorough discussion of the conditions under which this algorithm will and will not succeed).

15.2 Towards graph automorphism

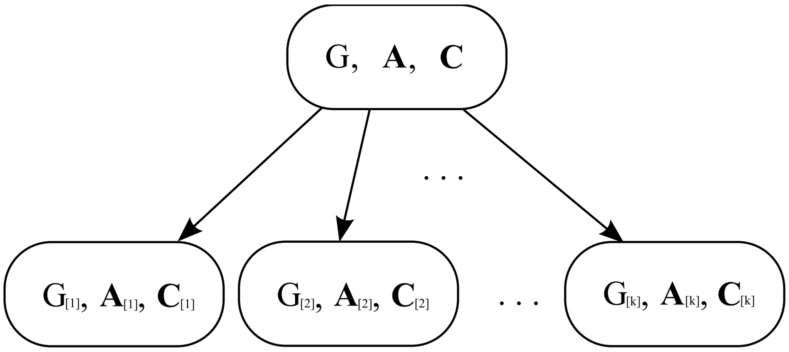
We now present an algebraic structure which hopefully will help us to further refine the coloring. Suppose that we apply the refinement algorithm to some initial set of colors. Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be the result of this procedure, and let $\mathbb{C} = \mathbb{F}[C_1, \ldots, C_m]$ for some finite field $\mathbb{F}$. The set $\mathbb{C}$ forms a ring, since by construction $C_iC_j = \Sigma_l C_{\alpha_l}$ for all i, j. By the same argument, if $\mathcal{C}$ is any set of well-behaved colors such that for each i, j it holds that $C_iC_j = \sum_\ell C_{\alpha_\ell}$ then we can form a ring $\mathbb{C}$ which we call a *ring of colors* in the sequel.

Remark: A graph G may have several rings of colors. Among them there is one which is the most refined which we call the *max-ring*. Let $\{D_1, \ldots, D_l\}$ be the set of all orbits of $Aut(G)$, the set of automorphisms of G. That is (x, y) and (u, v) have the same color iff there is an automorphism P of G mapping (x, y) to (u, v). Then $\mathbb{F}[D_1, \ldots, D_l]$ is the *max-ring*. It is indeed a ring since the D_is are closed under product.

Ideally, we would like to have a situation where a simple set of colors such as E, $\overline{E}$, and I would refine itself down to the max-ring. However there are graphs which are rigid but for which the color set $\{E, \overline{E}, I\}$ does not proceed past the initial coloring. The rest of the lecture presents ideas that lets us refine the coloring in ways that the 2-dim WL procedure does not cover.

We associate to every ring of colors $\mathbb{C}$ another ring of matrices $\mathbb{A}(\mathbb{C})$ which we call the *ring of commutators*. For a color ring $\mathbb{C}$, we define $\mathbb{A}(\mathbb{C}) = \{A | \forall C \in \mathbb{C}, AC = CA\}$. Observe that $\mathbb{A}$ is indeed a ring under matrix operations, and its basis can be computed easily from $\mathbb{C}$ by solving a system of linear equations. If all automorphisms of a graph G commute with all of the colors, then these automorphisms are present in $\mathbb{A}$.

Now, starting from a triple $(G, \mathbb{C}, \mathbb{A})$ we construct a tree as follows. Consider $C \in \mathcal{C}$ which is a vertex color such that $k = |C| > 1$ (if no such C exists then G is rigid and we are done). Let $v_1, \ldots, v_k$ be the k vertices of G of color C. For each i, let $\mathbb{C}_{[i]}$ be the ring of colors obtained by applying the 2-dim WL algorithm to the set $\mathcal{C}'$ of colors, where $\mathcal{C}'$ is defined from $\mathcal{C}$ by replacing C by C_1 and C_2 where C_1 contains only node v_i and C_2 contains all the remaining nodes of C. Set $\mathbb{A}_{[i]} = \mathbb{A}(\mathbb{C}_{[i]})$.



It is easy to see that all automorphisms of G that fix v_i are present in $\mathbb{A}_{[i]}$, but we have "lost" the automorphisms that do not fix any of the nodes which have the color C. We need to recover these automorphisms.

Assume first that there is an $i > 1$ such that the multiset of colors of $\mathbb{C}_{[i]}$ is different that those of $\mathbb{C}_{[1]}$. Then the set of automorphisms fixing v_1 is different from the set of automorphisms fixing v_i, and thus there cannot be an automorphism mapping 1 to i. Accordingly, we can refine the color set $\mathcal{C}$ as follows. Let C_1 be the set of all v_i such that $\mathbb{C}_{[i]}$ and $\mathbb{C}_{[1]}$ have the same multiset of colors, and let C_2 be the remaining nodes of C. Let $\mathcal{C}'$ be the color set formed by replacing C by C_1 and C_2 in $\mathcal{C}$, and let $\mathbb{C}'$ be the result of the 2-dim WL algorithm when started on $\mathcal{C}'$. We restart from the beginning after replacing $\mathbb{C}$ by $\mathbb{C}'$ and $\mathbb{A}$ by $\mathbb{A}(\mathbb{C}')$. By doing this we have added at least one new color preserved by all automorphisms of G, so we have made progress towards the max-ring.

We can eventually assume that for all $j \in \{1, \ldots, k\}$, $\mathbb{C}_{[1]}$ and $\mathbb{C}_{[j]}$ have the same multiset of colors. Therefore there is an invertible matrix A_{1j} such that $A_{1j}\mathbb{C}_{[1]} = \mathbb{C}_{[j]}A_{1j}$. Note that A_{1j} can be computed easily by solving a set of linear equations.

Claim 4. *For all $j \in \{1, \ldots, k\}$, $A_{1j}\mathbb{A}_{[1]} = \mathbb{A}_{[j]}A_{1j}$.*

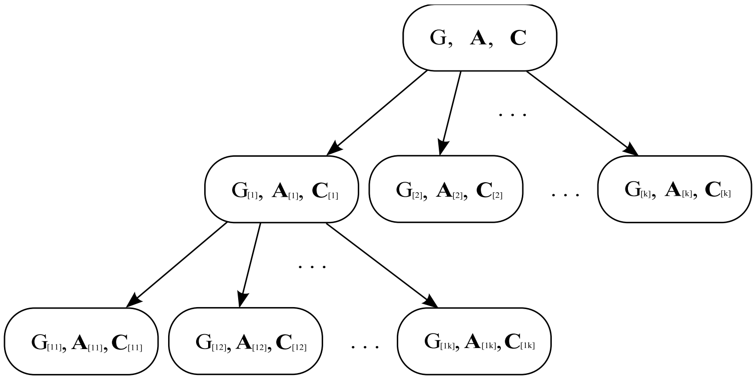
Proof. Let $A \in \mathbb{A}_{[1]}$ and set $B = A_{1j}AA_{1j}^{-1}$. Let $C \in \mathbb{C}_{[j]}$ and C' be the corresponding color in $\mathbb{C}_{[1]}$ (i.e. $CA_{1j} = C'$). We have $CB = CA_{1j}AA_{1j}^{-1} = A_{1j}C'AA_{1j}^{-1} = A_{1j}AC'A_{1j}^{-1} = A_{1j}AA_{1j}^{-1}C$. Therefore $B \in \mathbb{A}_{[j]}$. The other direction is similar. $\square$

Claim 5. *For all $j \in \{1, \ldots, k\}$, all automorphisms of G sending v_1 to v_j are in $A_{1j}\mathbb{A}_{[1]}$.*

Proof. Let P be an automorphism of G sending v_1 to v_j. Consider $P' = A_{1j}^{-1}P$. It is immediate to see that P' sends v_1 to v_1 and that P' preserves all other colors of $\mathbb{C}_{[1]}$. Therefore P' is in $\mathbb{A}_{[1]}$. $\square$

Consider $\mathbb{A}' = \mathbb{A}_{[1]} + A_{12}\mathbb{A}_{[1]} + \cdots + A_{1k}\mathbb{A}_{[1]}$. Clearly $\mathbb{A}' \subseteq \mathbb{A}$ since $\mathbb{A}'$ contains only matrices preserving colors which refine $\mathbb{C}$. If $\mathbb{A}' \subset \mathbb{A}$, we may be able to improve our situation. We may find that the ring of colors $\mathbb{C}'$ associated with $\mathbb{A}'$ is a refinement of $\mathbb{C}$, since $\mathbb{A}'$ contains all of the automorphisms of G. From now on we assume that $\mathbb{A} = \mathbb{A}_{[1]} + A_{12}\mathbb{A}_{[1]} + \cdots + A_{1k}\mathbb{A}_{[1]}$.

With all the claims above we can now replace $\mathbb{A}_{[j]}$ by $A_{1j}\mathbb{A}_{[1]}$ without losing any information. We can then continue this procedure recursively with $(G, \mathbb{C}_{[j]}, A_{1j}\mathbb{A}_{[1]})$, producing an exponential sized tree:



This tree contains all the automorphisms of G and therefore it would be useful to parse it efficiently. The idea is to use the extra algebraic structure of the tree in order to prune it or avoid parsing the whole tree.

Suppose that there are no linear dependencies amongst the subspaces $\mathbb{A}_{[1]}, A_{12}\mathbb{A}_{[1]}, \ldots, A_{1k}\mathbb{A}_{[1]}$. Then we know that $\dim A_{[1]} = \frac{1}{k}\dim \mathbb{A}$ and this is good because the dimension decreases sufficiently so that a brute force search of the tree would be polynomial.

In the case where there is a linear dependency amongst the subspaces $\mathbb{A}_{[1]}, A_{12}\mathbb{A}_{[1]}, \ldots, A_{1k}\mathbb{A}_{[1]}$, we set $I = \{A \in \mathbb{A}_{[1]} \mid A \in A_{12}\mathbb{A}_{[1]} + \cdots + A_{1k}\mathbb{A}_{[1]}\}$.

Claim 6. *I is an ideal of $\mathbb{A}_{[1]}$.*

Proof. It is clear that I is closed under addition, and right multiplication with $\mathbb{A}_{[1]}$. It remains to show that I is closed under left multiplication with $\mathbb{A}_{[1]}$.

Let $A \in \mathbb{A}_{[1]}$ and $\hat{A} \in I$. We want to show that $A\hat{A} \in I$. From $\hat{A} \in I$ we have $\hat{A} \in \mathbb{A}_{[1]}$ and therefore $A\hat{A} \in \mathbb{A}_{[1]}$. It remains to show that $A\hat{A} \in A_{12}\mathbb{A}_{[1]} + \cdots + A_{1k}\mathbb{A}_{[k]}$.

From $\hat{A} \in I$ we have $\hat{A} = \sum_{j=2}^{k} \alpha_j A_{1j} A_j$, with $A_j \in \mathbb{A}_{[1]}$.

Therefore $A\hat{A} = \sum_j \alpha_j A A_{1j} A_j$.

If we repeat the construction above to $(G, \mathbb{C}_{[1]}, \mathbb{A}_{[1]})$, we get:

$$\mathbb{A}_{[1]} \quad = \quad \bigoplus_{\ell} A_{1,2\ell}\mathbb{A}_{[1,2]} = \cdots = \bigoplus_{\ell} A_{1,k\ell}\mathbb{A}_{[1,k]} \tag{3}$$

where the sum goes from $\ell = 2$ to $\ell = k$, where $\mathbb{A}_{[1,i]}$ is the set of linear transformations fixing v_1 and v_i and $A_{1,ij}$ sends $\mathbb{C}_{[1,i]}$ to $\mathbb{C}_{[1,j]}$ (it is the identity in the case $i = 1$).

For all j, this implies that $A = \sum_{\ell=2}^{\ell=k} B_{1,j\ell} B_{1,j}$, where $B_{i,j} \in \mathbb{A}_{[1,j]}$ and $B_{1,j\ell} \in A_{1,j\ell}$.

Therefore $AA_{1j} = \sum_{\ell} B_{1,j\ell} B_{1,j} A_{1,j}$.

Notice now that each element of the sum maps 1 to ℓ and preserves colors. Therefore it can be written as $A_{1\ell} B_1^{(j\ell)}$ for some $B_1^{(j\ell)} \in \mathbb{A}_{[1]}$.

Thus $AA_{1j} = \sum_{\ell} A_{1\ell} B_1$ and $A\hat{A} = \sum_j \alpha_j (\sum_{\ell} A_{1\ell} B_1^{(j\ell)} A_j)$.

This implies that $A\hat{A} = \sum_{\ell} A_{1\ell} (\sum_j \alpha_j B_1^{(j\ell)} A_j)$.

Notice now that $(\sum_j \alpha_j B_1^{(j\ell)} A_j) \in \mathbb{A}_{[1]}$ because both $B_1^{(j\ell)}$ and A_j are in $\mathbb{A}_{[1]}$. This implies $A\hat{A} \in A_{12}\mathbb{A}_{[1]} + \cdots + A_{1k}\mathbb{A}_{[1]}$ as desired. ∎

For $j > 1$, set $I_{1j} = A_{1j}\mathbb{A}_{[1]} \cap (\mathbb{A}_{[1]} + \sum_{\ell \neq j} A_{1\ell}\mathbb{A}_{[1]})$.

Claim 7. *For all $j > 1$, $I_{1j} = A_{1j}I$.*

Proof. Consider $B \in A_{1j}\mathbb{A}_{[1]} \cap (\mathbb{A}_{[1]} + \sum_{\ell \neq j} A_{1\ell}\mathbb{A}_{[1]})$. Then, B can be written as:

$$B = A_{1j}B_j = B_1 + \sum_{\ell \neq j} A_{1\ell}B_\ell,$$

with each $B_i \in \mathbb{A}_{[1]}$. Multiplying by $(A_{1j})^{-1}$, we get:

$$B_j = (A_{1j})^{-1}B_1 + \sum_{\ell \neq j} (A_{1j})^{-1}A_{1\ell}B_\ell.$$

$(A_{1j})^{-1}$ sends j to 1 and can be written as $A_{j1,i2} + A_{j1,i3} + \cdots + A_{j1,ik}$ for $i \neq j$ and matrices $A_{j1,im}$ which send j to 1 and j to m. Using this, for all $\ell \neq j$ we can write $(A_{1j})^{-1}A_{1\ell}B_\ell$ as:

$$(A_{j1,\ell 2} + A_{j1,\ell 3} + \cdots + A_{j1,\ell k})A_{1\ell}B_\ell = (A'_{12} + A'_{13} + \cdots + A'_{1k})B_\ell.$$

Therefore, $B_j \in I$ implying $B \in A_{1j}I$. A similar argument shows that any B in $A_{1j}I$ can be written as above. $\square$

Finally consider $\hat{I} = I + A_{12}I + \cdots + A_{1k}I$.

Claim 8. *$\hat{I}$ is an ideal of $\mathbb{A}$.*

The proof was omitted. We now want to use this ideal $\hat{I}$ to prune the tree defined earlier.

16 Integer Lattices

Definition 1. *$L \subseteq \mathbb{R}^n$ is an* integer lattice *if there exist linearly independent vectors $v_1, v_2, \ldots, v_m \in \mathbb{R}^n$ such that $L = \{\sum_{i=1}^m a_i v_i | a_i \in \mathbb{Z}\}$. The vectors $v_1, \ldots, v_m$ form a basis of L.*

Example 1. Lattice consisting of unit vectors, $v_i = e_i$. It can be viewed as a parallelepiped, repeated infinitely many times.

Lattices are interesting because of many applications. Consider a shortest vector in the lattice. Clearly $\vec{0}$ is the shortest vector. The problem is to find a shortest *non-zero* vector. There is a well-defined shortest vector[1].

Let's define some parameters associated with the lattice. We will assume that $L \subseteq \mathbb{R}^n$ is an n-dimensional lattice. Define $d(L) = |det[v_1 \ldots v_n]|$, where $v_1 \ldots v_n$ are the basis vectors of L.

Observation 1. *The value $d(L)$ is independent of basis.*

Proof. Let B_1 and B_2 be different bases of L. Then $B_1 = AB_2$ and A has integer entries. Also, $B_2 = A'B_1$, where A' has integer entries. So $\det(B_1) = \det(A)\det(B_2)$ and $\det(B_2) = \det(A')\det(B_1)$. Since $\det(A)$ and $\det(A')$ are integers, it follows that both of them are equal to 1. ☐

Definition 2. *The number $d(L)$ is called the* volume *of L (since it corresponds to the volume of the parallelepiped).*

Theorem 14 (Minkowski's theorem[16])**.** *For any lattice L, the length $\lambda(L)$ of its shortest vector is at most $\sqrt{n} \cdot (d(L))^{1/n}$.*

Problem 1. *Computational problem: find a shortest vector efficiently.*

The bottleneck here is the following.

Theorem 15 (Ajtai[2])**.** *Finding a shortest vector in a lattice is NP-hard.*

On the other hand, we have the following algorithm due to Lenstra, Lenstra, and Lovasz; this algorithm is usually called L^3 algorithm.

Theorem 16 (Lenstra-Lenstra-Lovasz[12])**.** *There is a polynomial-time algorithm to find a vector in a lattice of length at most $2^{\frac{n-1}{2}} \cdot \lambda(L)$.*

Improvement to Ajtai's theorem: getting within the $\sqrt{2}$ factor is NP-hard.

Before proving either of these theorems, let's see why lattices are useful. In the next section, we'll use lattices to break RSA with certain parameters.

[1] Here, Michal asked what about a lattice with π and 1? Then the shortest vector $\to 0$. Decided that linear independence fixes this problem

17 Breaking RSA public key

Recall the definition of the RSA public key cryptosystem. We are given $n = pq$, where p, q are large primes, and a public key e (small exponent, e.g., $e = 3$, so that encryption is fast). A message m is encoded into $c = m^e (\mod n)$.

Assume that we know the first part of the message m, e.g., $m = a \cdot 2^k + x$, where x is unknown, and a and k are known. So we know $\log n - k$ bits of the message, and don't know the last k bits. Here

$$c = (a \cdot 2^k + x)^3 (\mod n).$$

We can compute this as the following degree 3 polynomial: $x^3 + \alpha x^2 + \beta x + \gamma (\mod n)$, where α, β, γ are known. Then, $x^3 + \alpha x^2 + \beta x + (\gamma - c) = 0 (\mod n)$. We want to know x such that $x < 2^k$.

Note that here we don't have a fields, so can't find roots even for cubic, or quadratic polynomials. (If we could find roots of quadratic polynomials, then we could factor by finding square roots as in the previous lectures.)

Let

$$p(x) = x^3 + \alpha x^2 + \beta x + (\gamma - c).$$

Define a lattice L_6 using the following six polynomials:

$$q_0 = n, q_1 = nx, q_2 = nx^2, q_3 = p(x), q_4 = xp(x), q_5 = x^2 p(x).$$

$$L_6 = \begin{vmatrix} n & 0 & 0 & 0 & 0 & 0 \\ 0 & n & 0 & 0 & 0 & 0 \\ 0 & 0 & n & 0 & 0 & 0 \\ \gamma - c & \beta & \alpha & 1 & 0 & 0 \\ 0 & \gamma - c & \beta & \alpha & 1 & 0 \\ 0 & 0 & \gamma - c & \beta & \alpha & 1 \end{vmatrix}$$

A shortest vector in L_6 is of length at most $\sqrt{6}(d(L_6))^{1/6} = \sqrt{6n}$, since $\lambda(L_6) = n^3$. Running the L^3 algorithm on the lattice L_6 yields a vector u of length at most $\sqrt{32 \cdot 6n} = \sqrt{192n}$.

Consider the polynomial $u(x)$ with elements of u as coefficients.

$$u(x) = \sum_{i=0}^{5} c_i q_i(x) = c_3\, p(x) + c_4\, x\, p(x) + c_5\, x^2\, p(x) (\mod n) = (c_3 + c_4 x + c_5 x^2) p(x) (\mod n)$$

If $p(\delta) = 0 (\mod n)$, then $u(\delta) = 0 (\mod n)$, i.e., all roots of p are roots of u. We want to find all small roots of $u(x)$, i.e., all small roots δ of $u(x)$ such that $\delta < 2^k$.

Let $u(x) = \sum_{i=0}^{5} d_i x^i$. By length of u, we know that $abs(d_i) \le \sqrt{192n}$. So, $abs(u(\xi)) = abs(\sum_{i=0}^{5} d_i \delta_i) \le \sum_{i=0}^{n} \sqrt{192n} \cdot 2^{ik} < \sqrt{192n} \cdot 2^{6k}$ for $\xi < 2^k$.

Suppose that k is such that $\sqrt{192n} \cdot 2^{6k} < n$. That is, $k \sim \frac{1}{12} \log n$. If $u(\delta) = 0 (\mod n)$, for $\delta < 2^k$, then $u(\delta) = 0$. So we can factor $u(x)$ over Q and get a small root of $p(x) (\mod n)$.

Using these arguments, originally due to Håstad [7], we can bring k to about $1/2$ of the message.

18 NP-hardness of the Shortest Vector Problem

Here we will prove that finding a shortest vector is NP-hard, a result originally due to Ajtai [2]. In fact, we will show a somewhat stronger result that even approximating the length of a shortest vector to within a factor $\sqrt{2}$ is NP-hard. This was shown by Micciancio [15].

Recall the definition. **Shortest Vector Problem (SVP)**: given L and k, is shortest vector in L of length $\le k$?

Theorem 17 ([2, 15])**.** *SVP is NP-hard.*

Proof. The proof is by reduction from the Subset Sum (SS) problem. Recall that the language Subset Sum is $SS = \{(a_1, \ldots, a_n, b) \mid \exists S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} a_i = b\}$. Starting with an instance $(a_1, \ldots, a_n, b)$ of SS, define a lattice L as follows.

$$L = \begin{vmatrix} 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & 0 \\ N_{a_1} & N_{a_2} & \ldots & N_{a_n} & N_b \end{vmatrix}$$

where N is a large number (to be chosen). Name the columns $b_1 \ldots b_{n+1}$.

If there is a solution to this SS instance, then there is a vector in L of length $\leq \sqrt{n}$. If there is no solution to the SS instance, then for all integer $\alpha_1, \ldots, \alpha_n$ and $\beta > 0$, we have

$$|\sum \alpha_i b_i - \beta b_{n+1}| > N.$$

(A related Closest Vector Problem is NP-hard by the argument above.)

Let $L = [b_1 \ldots b_{n+1}]$. We'll add something to the top portion of this lattice L so that either the last column is used (and hence the result is large), or the added part on top makes the result large. On the other hand, we must ensure that if there exists a solution, then the added part at the top doesn't interfere too much.

Consider a lattice $M_{m \times (m+1)} \subseteq \mathbb{R}^{m+1}$ and a vector $c \in \mathbb{R}^{m+1}$ with the following properties:

1. $\lambda(M) > \sqrt{2}r$.
2. There are many vectors $z \in \{0,1\}^m$ such that $|Mz - c| \leq r$.
3. Let $C = \{z \in \mathbb{Z}^m \mid |Mz - c| \leq r\}$. Then there is a linear map $T : \mathbb{Z}^m \to \mathbb{Z}^n$ such that $\{0,1\}^n \subseteq TC$.

Assuming the existence of such M and c, let's complete the proof. Define a lattice $\hat{L}$, of dimension $m + n + 1$ by $m + 2$, as follows:

$$\hat{L} = \begin{vmatrix} M & c \\ L_0 T & b_{n+1} \end{vmatrix} \qquad \text{where } L_0 \text{ is } [b_1 \ldots b_n].$$

Suppose the SS instance has no solution. Consider vector $\hat{L}\bar{\alpha}$, where

$$\bar{\alpha} = \begin{vmatrix} \alpha_1 \\ \vdots \\ \alpha_m \\ \alpha_{m+1} \end{vmatrix} = \begin{vmatrix} \hat{\alpha} \\ \alpha_{m+1} \end{vmatrix}$$

Then $|\hat{L}\bar{\alpha}|^2 = |M\hat{\alpha} + \alpha_{m+1}c|^2 + |L_0 T\hat{\alpha} + \alpha_{m+1}b_{n+1}|^2 \geq \min\{2r^2, N^2\}$.

On the other hand, suppose SS has a solution

$$\hat{\beta} = \begin{vmatrix} \beta_1 \\ \vdots \\ \beta_n \end{vmatrix}$$

Let $\hat{w} \in \mathbb{Z}^m$ be such that $T\hat{w} = \hat{\beta}$. Now let

$$\bar{w} = \begin{vmatrix} \hat{w} \\ -1 \end{vmatrix}$$

Then $|\hat{L}\bar{w}|^2 = |M\hat{w} - c|^2 + |L_0\hat{\beta} - b_{n+1}|^2 \leq r^2 + n$. The multiplicative gap between Yes and No instances is thus $\sqrt{2}$, as required.

The only missing part is the construction of M. We'll present it in the next lecture. $\square$

19 The LLL Algorithm

This description of the LLL algorithm is based on Manindra's lecture and the notes at `www.cs.tau.ac.il/~odedr/goto.php?name=ln_lll&link=teaching/lattices_fall_2004/ln/lll.pdf`.

19.1 Definitions and Lemmas

We describe an approximation algorithm to the Shortest Vector Problem (SVP). This algorithm was developed in 1982 by A. K. Lenstra H. W. Lenstra Jr. and L. Lovász and it is usually called the LLL algorithm. It gives a $(\frac{2}{\sqrt{3}})^n$ approximation ratio, where n is the dimension of the lattice. For simplicity, we describe the LLL algorithm for full-rank lattices but it is easy to remove this restriction. Furthermore, this description only applies to the l_2 norm but extensions to other norms are known.

First we remind the reader the Gram-Schmidt orthogonalization. Let $b_1, b_2, \ldots, b_n$ be a set of n linearly independent vectors. We define their Gram-Schmidt orthogonalization as the sequence of vectors $b_1^*, b_2^*, \ldots, b_n^*$ where

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*, \text{ where } \mu_{i,j} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}.$$

In words, b_i^* is the component of b_i orthogonal to $b_1^*, \ldots, b_{i-1}^*$. The Gram-Schmidt orthogonalization has many nice properties, one of which is the following.

Lemma 3. *Let $B = \{b_1, \ldots, b_n\}$ be a rank-n lattice basis, and let B^* be its Gram-Schmidt orthogonalization. Then*

$$\lambda(\mathcal{L}(B)) \geq \min_{i=1,\ldots,n} \|b_i^*\| > 0$$

Notation: B is a square matrix and $\mathcal{L}(B)$ denotes the lattice whose basis vectors are the columns of B. $\lambda(\mathcal{L}(B))$ denotes the shortest vector in $\mathcal{L}(B)$.

Proof. Let $x \in \mathbb{Z}^n$ be an arbitrary non-zero integer vector. We will show that $\|Bx\| \geq \min_{i=1,\ldots,n} \|b_i^*\|$. Let $j \in \{1, \ldots, n\}$ be the largest such that $x_j \neq 0$. Then

$$|\langle Bx, b_j^* \rangle| = |\langle \sum_{i=1}^{j} x_i b_i, b_j^* \rangle| \tag{4}$$
$$= |x_j| \langle b_j^*, b_j^* \rangle \tag{5}$$
$$= |x_j| \|b_j^*\|^2 \tag{6}$$

where the second equality follows from the fact that for all $i < j$, $\langle b_i, b_j^* \rangle = 0$ and that $\langle b_j, b_j^* \rangle = \langle b_j^*, b_j^* \rangle$. By the Cauchy-Schwarz inequality we have that $|\langle Bx, b_j^* \rangle| \leq \|Bx\| \|b_j^*\|$. It follows that

$$\|Bx\| \geq |x_j| \|b_j^*\| \geq \|b_j^*\| \geq \min_{i=1,\ldots,n} \|b_i^*\|.$$

□

Definition 3. *Consider a basis $B = \{b_1, \ldots, b_n\} \in \mathbb{R}^n$ and its Gram-Schmidt orthogonalization. Then B is called δ-**reduced** if the following conditions hold:*

1. *For all i, $1 \le i \le n$ such that $j < i$, $\|\mu_{i,j}\| \le \frac{1}{2}$;*
2. *For all i, $1 \le i \le n$, $\delta\|b_i^*\|^2 \le \|\mu_{i+1,i}b_i^* + b_{i+1}^*\|^2$.*

It is always possible to transform a basis to a reduced basis and this is what the LLL algorithm does. The algorithm works with any δ, $\frac{1}{4} < \delta < 1$ but it is helpful to consider the case when $\delta = \frac{3}{4}$. In fact, Manindra considered only this case. When he defined a reduced basis his second condition was: "For all i, $1 \le i \le n$, $\|b_i^*\|^2 \le 2\|b_{i+1}^*\|^2$". To see the equivalence note that in Definition 3 the second condition can be written as

$$\delta\|b_i^*\|^2 \le \|\mu_{i+1,i}b_i^* + b_{i+1}^*\|^2 = \mu_{i+1,i}^2\|b_i^*\|^2 + \|b_{i+1}^*\|^2,$$

where the second equality follows since b_i^* and b_{i+1}^* are orthogonal. It follows that

$$\|b_{i+1}^*\|^2 \ge (\delta - \mu_{i+1,i}^2)\|b_i^*\|^2 \ge (\delta - \frac{1}{4})\|b_i^*\|^2,$$

and when $\delta = \frac{3}{4}$ we have that

$$2\|b_{i+1}^*\|^2 \ge \|b_i^*\|^2.$$

Intuitively, the second condition says that "b_{i+1}^* is not much shorter than b_i^*".

It is also helpful to consider the orthonormal basis obtained by normalizing the Gram-Schmidt vectors $b_1^*, \ldots, b_n^*$. In this basis, B can be expressed as

$$\begin{pmatrix} \|b_1^*\| & * & \cdots & * \\ 0 & \|b_2^*\| & \cdots & * \\ \vdots & & \ddots & \vdots \\ & & & * \\ 0 & \cdots & & \|b_n^*\| \end{pmatrix}$$

where column i shows the coordinates of b_i in this orthonormal basis. The first condition in Definition 3 says that the absolute value any off-diagonal element in the upper triangle is at most half the value written in the diagonal element on the same row. So the above matrix becomes

$$\begin{pmatrix} \|b_1^*\| & \le \frac{1}{2}\|b_1^*\| & \cdots & \le \|b_1^*\| \\ 0 & \|b_2^*\| & \cdots & \le \|b_2^*\| \\ \vdots & & \ddots & \vdots \\ & & & \le \frac{1}{2}\|b_{n-1}^*\| \\ 0 & \cdots & & \|b_n^*\| \end{pmatrix}$$

Here $\frac{1}{2}\|b_j^*\|$ indicates that the absolute value of this coordinate is at most $\frac{1}{2}\|b_j^*\|$. The second condition can be thought of as a restriction on the entries of a 2×2 submatrix of the above matrix where the upper left entry is (i, i):

$$\begin{pmatrix} \|b_i^*\| & \mu_{i+1,i}\|b_i^*\| \\ & \\ 0 & \|b_{i+1}^*\| \end{pmatrix}.$$

The following Lemma states that the first vector of a δ-reduced basis is relatively short. This provides us with one way of giving an approximation to SVP because we can just produce a δ-reduced basis and return b_1 as our answer. When $\delta = \frac{3}{4}$, this will give as a $2^{\frac{n-1}{2}}$ approximation.

Lemma 4. *If a basis $B = \{b_1, \ldots, b_n\}$ is δ-reduced then $\|b_1\| \le \left(\frac{2}{\sqrt{4\delta - 1}}\right)\lambda(\mathcal{L}(B))$. Note that for $\delta = \frac{3}{4}$ this gives $\|b_1\| \le 2^{\frac{n-1}{2}}\lambda(\mathcal{L}(B))$.*

Proof. Because B is δ-reduced,

$$\|b_n^*\|^2 \geq \left(\delta - \frac{1}{4}\right) \|b_{n-1}^*\|^2 \geq \ldots \geq \left(\delta - \frac{1}{4}\right)^{n-1} \|b_1^*\|^2 = \left(\delta - \frac{1}{4}\right)^{n-1} \|b_1\|^2$$

where the last equality holds because by definition, $b_1^* = b_1$. It follows that for any i,

$$\left(\delta - \frac{1}{4}\right)^{\frac{-(n-1)}{2}} \|b_i^*\| \geq \left(\delta - \frac{1}{4}\right)^{\frac{-(i-1)}{2}} \|b_i^*\| \geq \|b_1^*\| = \|b_1\|.$$

Therefore we have that

$$\|b_1\| \leq \left(\delta - \frac{1}{4}\right)^{\frac{-(n-1)}{2}} \min_{i=1,\ldots,n} \|b_i^*\| \leq \left(\delta - \frac{1}{4}\right)^{\frac{-(n-1)}{2}} \lambda(\mathcal{L}(B)),$$

where the last inequality follows from Lemma 3. □

Now our task is to generate a δ-reduced basis from our input basis and return b_1 as our approximation to the shortest vector. In the next section we describe the LLL algorithm and prove it's correctness.

19.2 The Algorithm and Its Correctness

Input: Lattice basis $B = \{b_1, \ldots, b_n\}$ where $b_i \in \mathbb{Z}^n$
Output: A δ-reduced basis for $\mathcal{L}(B)$

Start: compute the Gram-Schmidt orthogonalization $b_1^*, \ldots, b_n^*$
Reduction Step:
for $i = 2$ to n **do**
  for $j = i - 1$ down to 1 **do**
    $b_i \leftarrow b_i - c_{i,j} b_j$ where $c_{i,j} = \left\lceil \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \right\rfloor$
  end for
end for
Swap Step:
if $\exists i$ s.t. $\delta \|b_i^*\|^2 > \|\mu_{i+1,i} b_i^* + b_{i+1}^*\|$ **then**
  $b_i \leftrightarrow b_{i+1}$
  Go to Start
end if
Output $b_1, \ldots, b_n$

Notation: $\lceil x \rfloor$ denotes the nearest integer to x. Let us make an important observation. During the Reduction Step, the algorithm performs only column operations of the form $b_i \leftarrow b_i + a b_j$ for $i > j$ and $a \in \mathbb{Z}$ on the vectors $b_1, \ldots, b_n$. It is easy to see that this does not change the Gram-Schmidt basis during the Reduction Step.

Lemma 5 (Correctness)**.** *If the LLL algorithm ever terminates then its output is a δ-reduced basis for the lattice spanned by the input basis $B = \{b_1, \ldots, b_n\}$.*

Proof. We show that the output of the LLL algorithm is a basis for $\mathcal{L}(B)$ that satisfies both conditions of Definition 3. It is easy to see that the second property is enforced by the check during the Swap Step. To see that the output of the algorithm is indeed a basis for $\mathcal{L}(B)$ note that the algorithm performs only column operations of the form $b_i \leftarrow b_i + a b_j$ for $i \neq j$, and $a \in \mathbb{Z}$.

It remains to show that after the reduction step, $b_1, \ldots, b_n$ satisfy $\|\mu_{i,j}\| \leq \frac{1}{2}$, for all $i > j$. Consider some $i > j$ and consider the jth iteration of the inner loop in the ith iteration of the outer loop. Then we can write $|\mu_{i,j}|$ as follows,

$$\begin{aligned}
|\mu_{i,j}| &= \left| \frac{\langle b_i - c_{i,j} \cdot b_j, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \right| \\
&= \left| \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} - \left\lceil \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \right\rfloor \cdot \frac{\langle b_j, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \right| \\
&\leq \frac{1}{2}.
\end{aligned}$$

Here, the first equality follows from the definition of $\mu_{i,j}$, the definition of b_i in line 5 of the algorithm, and the above-mentioned observation that the Gram-Schmidt basis does not change throughout the Reduction Step. The last inequality follows from the fact that $\langle b_j, b_j^* \rangle = \langle b_j^*, b_j^* \rangle$. $\square$

It can be shown that the LLL algorithm runs in polynomial time. Manindra started explaining the beginning of the proof but due to lack of time he did not get far. The argument is described in detail in the notes that can be found at the URL at the beginning of this section.

Title: Factoring polynomials over rationals | Lecture #: 8
Lecturer: Manindra Agrawal | Scribe: Michal Koucký, Mike Saks

20 Factoring polynomials over rationals

We have already seen how to factor polynomials over small finite fields. In this lecture we will show how to factor a polynomial $f \in Q[x]$ over rationals. We will make three simplifying assumptions about f:

1. f has integral coefficients,
2. the leading coefficient of f is 1, and
3. f is square-free.

We justify these assumptions. It is easy to make coefficients of f integral by multiplying f by the least common multiple of all the coeff's denominators. This does not affect factorization of f in a substantial way. So the first property can be satisfied. If the leading coefficient a of f is not one then we can multiply f by $a^{(\deg f)-1}$ and substitute $x = y/a$ for x. A factorization of the new polynomial in variable y provides a factorization of f by substituting $y = ax$ into it. So we can assume the second property to hold. Last, if f is not square-free then there is a common nontrivial factor between f and the (formal) derivative f' of f. Such a factor can easily be found by Euclid's algorithm. So from now on we assume that f has the required properties. That allows us to use the following consequence of Gauss's lemma:

Proposition 1. *Let $f = gh$ be polynomials from $Q[x]$. If the leading coefficients of f, g and h are one and f has only integral coefficients then g and h have also only integral coefficients.*

Hence, we essentially need to factor f over integers instead of rationals. The factoring algorithm has four steps:

1. Find a small prime p such that a) f is square-free over the field F_p and b) none of the coefficients of f is divisible by p.
2. Factor f as $g \cdot h$ over F_p, with g irreducible.
3. *Lift* these factors to modulo p^ℓ, i.e., solve $f = \hat{g} \cdot \hat{h} \mod p^\ell$, where $\hat{g}$ is irreducible modulo p^ℓ. The integer ℓ will be chosen later.
4. Setup a lattice L using $\hat{g}$. A short vector in L leads to a factor of f over Z.

The first step can be done by brute force provided that such a small prime exists. Indeed, it exists. Consider a linear map $Res_{f,g}(a, b) = af + bg$, where a, b are polynomials of degrees $\deg a < \deg g$ and $\deg b < \deg f$. The mapping $Res_{f,g}$ can be described by a square matrix $Res(f, g)$ of dimension $\deg f + \deg g - 1$ with entries being either zero or coefficients of f and g. Clearly, $Res(f, g)$ is singular over a ring R iff f and g share a common nontrivial factor over R. The determinant of $Res(f, g)$ is of size at most $2^{\mathrm{poly}(n(\deg f + \deg g))}$. (Here n stands for the upper bound on the bit-size of coefficients of f.) In order for f and its derivative f' to have a common factor over F_p the determinant $Res(f, f')$ must be zero over F_p. But then p divides the determinant of $Res(f, g)$ over Z. There cannot be many such primes p.

We have already seen an algorithm for factoring polynomials over F_p. Thus we will use it. The third step is done via *Hensel Lifting*.

20.1 Hensel Lifting

Suppose we know factorization of $f(x) = g(x) \cdot h(x)$ modulo some integer m. We want to find a factorization of $f(x)$ modulo m^2. We use *Hensel Lifting*. Assume that we have polynomials $f = g \cdot h \mod m$, with the leading coefficients of f and g co-prime with m, g irreducible over m and such that for some polynomials s and t, $sg+th = 1 \mod m$. Hensel lifting computes polynomials g', and h' such that $f = g' \cdot h' \mod m^2$. Set $e = f - gh$. If $e = 0 \mod m^2$ then we do not need to lift g and h otherwise define $g' = g+te \mod m^2$ and $h' = h+se \mod m^2$. Hence

$$f - g'h' = f - gh - gse - hte - ste^2 = f - gh - (sg+th)e - ste^2 = (1 - sg - th)e - ste^2$$

Since $e = 0 \mod m$ and $(1 - sg - th) = 0 \mod m$, $f - g'h'$ is divisible by m^2, i.e., $f - g'h' = 0 \mod m^2$.

We can also find s' and t' such that $s'g' + t'h' = 1 \mod m^2$: Define $d = sg' + th' - 1 \mod m^2$ and set $s' = s(1-d)$ and $t' = t(1-d)$. Clearly, $s'g' + t'h' = (sg' + th')(1-d) = 1 - d^2 \mod m^2$. Since $g' = g \mod m$ and $h' = h \mod m$, $sg' + th' - 1 = sg + th - 1 = 0 \mod m$. Hence, $d = 0 \mod m$. Thus $s'g' + t'h' = 1 \mod m^2$.

This particular way of lifting has one drawback, however. The degrees of g' and h' can be larger than that of g and h. To overcome this problem we modify the process as follows: Set $e = f - gh$ and let r be a polynomial of degree $\deg r < \deg g$ such that $te = qg + r \mod m^2$. Define $g' = g + r \mod m^2$ and $h' = h + se + qh \mod m^2$. Then

$$\begin{aligned} f - g'h' &= f - (g+r)(h+se+qh) \\ &= f - (g+te-qg)(h+se+qh) = f - gh - (sg+th)e - ste^2 + (sg-th)qe + ghq^2 \\ &= (1 - sg - th)e - ste^2 + (sg-th)qe + ghq^2 \mod m^2. \end{aligned}$$

As before, $e = 0 \mod m$ and $(1 - sg - th) = 0 \mod m$. Furthermore, $q = 0 \mod m$ since the leading coefficient of g is co-prime with m, $te = 0 \mod m$ and $\deg r < \deg g$. (Consider how the usual algorithm for dividing two polynomials works.) $q = 0 \mod m$ and $te = 0 \mod m$ imply $r = 0 \mod m$. Hence $g' = g \mod m$ and $h' = h \mod m$. Thus s' and t' defined previously also give $s'g' + t'h' = 1 \mod m^2$. The leading coefficients of g and g' are the same, i.e., co-prime with m^2, and the degrees of g and g' are the same. It is straightforward to verify that if g is irreducible modulo m then also g' is irreducible modulo m^2. ($g' = a'b' \mod m^2$ where a and b are nontrivial polynomials along with $g = g' \mod m$ imply $g = (a \mod m) \cdot (b \mod m) \mod m$, where $a \mod m$ and $b \mod m$ are also nontrivial polynomials.) Also notice, since none of the non-zero coefficients of h' is divisible by m^2, the leading coefficients of g' and f are co-prime with m and $f = g'h' \mod m^2$, $\deg h' = \deg f - \deg g'$.

We can repeat this procedure multiple times to lift the factorization $f = gh \mod p$ to $f = \hat{g} \cdot \hat{h} \mod p^{2^k}$, for arbitrary $k > 0$.

We will need the following claim.

Lemma 6 (Uniqueness of Hensel Lifting)**.** *Assume $g^*h^* = gh \mod m^\ell$, where $g^* = g \mod m$, the leading coefficients of g and g^* are the same and are co-prime with m, the degrees of g and g^* are the same and for some polynomials s, t, $1 = sg + th \mod m$. Then $g^* = g \mod m^\ell$ and $h^* = h \mod m^\ell$.*

Proof. Let i be the largest integer such that m^i divides both $g^* - g$ and $h^* - h$. We will prove by contradiction that $i \geq \ell$. Assume $i < \ell$. Let $g^* - g = m^i u$ and $h^* - h = m^i v$. By the conditions of the lemma

$$0 = g^*h^* - gh = g^*(h^* - h) + (g^* - g)h \mod m^\ell.$$

Hence,

$$\begin{aligned} 0 &= p^i g^* v + p^i uh \mod m^\ell \\ 0 &= g^* v + uh \mod m^{\ell - i} \\ 0 &= gv + uh \mod m. \end{aligned}$$

Since $1 - gs = th \mod m$,

$$\begin{aligned} 0 &= tgv + u(1 - sg) \mod m \\ 0 &= (tv - us)g + u \mod m. \end{aligned}$$

Since the leading coefficient of g is co-prime with m and $\deg u < \deg g$ (as the leading coefficients of g and g^* are the same) it must be that $(tv - us) = 0 \mod m$. Thus $u = 0 \mod m$. Consequently, $0 = gv + uh = gv \mod m$. Again, as the leading coefficient of g is co-prime with m, $v = 0 \mod m$. That contradicts the maximality of i. □

20.2 Setting up the lattice

In this section we describe the last step of the algorithm. Assume that we have $f = \hat{g} \cdot \hat{h} \mod p^l$, where $\hat{g}$ is irreducible modulo p^ℓ with the leading coefficient being one. (Note, we may request such a polynomial g in Step 2 and its properties will remain unchanged during Hensel Lifting in Step 3 of the algorithm.) Let $d_1 = \deg \hat{g} < d = \deg f$. Define a d-dimensional lattice $L \subseteq R^d$ using polynomials $p^\ell, p^\ell x, \ldots, p^\ell x^{d_1 - 1}$, $\hat{g}(x), x\hat{g}(x), \ldots, x^{d-d_1-1}\hat{g}(x)$. L is spanned by the d-dimensional vectors of coefficients of these polynomials. Let u be a vector in L. Then

$$\begin{aligned} u(x) &= \sum_{i=0}^{d_1-1} \alpha_i x^i p^\ell + \sum_{i=d_1}^{d} \alpha_i x^{i-d_1} \hat{g}(x) \\ &= g'(x) \cdot \hat{g}(x) \mod p^\ell. \end{aligned}$$

We claim that L contains a factor f_1 of f over $Z[x]$. Let $g \in F_p[x]$ be the irreducible factor of f obtained in the second step of the algorithm. As f factorizes uniquely over the field F_p, there is some factor $f_1 \in Z[x]$ of f over Z such that g is a factor of f_1. By the uniqueness of Hensel Lifting, $\hat{g}$ obtained in the third step of the algorithm divides f_1 also over Z_{p^ℓ}. (Lifting g as a factor of f or f_1 must give the same $\hat{g}$ modulo p^ℓ.)

Hence, any factor f_1 of f over $Z[x]$ that is divisible by $\hat{g}$ modulo p^ℓ is in L. (Since the leading coefficient of $\hat{g}$ is one and f_1 is divisible by $\hat{g}$ modulo p^ℓ , there are polynomials $q, r \in Z[x]$ such that $f_1 = \hat{g}q + r$, $\deg r < \deg \hat{g}$ and $r = 0 \mod p^\ell$.) This provides a bound on $\lambda(L)$ using Mignotte bound:

Mignotte bound: For polynomials $g, h, f \in Z[x]$ such that $f = gh$,

$$||g||_2^2 \leq 2^{2 \deg g} ||f||_2^2,$$

where $|| \cdot ||_2^2$ is the square of L_2 norm of a polynomial, i.e., the sum of the squares of all the coefficients of that polynomial.

Hence, $\lambda(L) \leq \sqrt{d} 2^{dn}$. The LLL algorithm can find a vector u of size $2^{(d-1)/2}\lambda(L) \leq \sqrt{d}2^{d(n+1)}$ in L. We claim that u and f have a common nontrivial factor over Z provided that $\ell > 10n^2d^2$:

For any vector $u \in L$, f and u have a nontrivial factor modulo p^ℓ. Hence $Res(f, u)$ is singular modulo p^ℓ. If all the coefficients of u are between $-K$ and K, for $K = \sqrt{d}2^{d(n+1)}$ then $det|Res(f, u)| \leq (2d!)(K2^n)^2 d \leq 2^{4d^2 n} < p^\ell$. As $Res(f, u)$ is singular modulo p^ℓ, and $det|Res(f, u)| < p^\ell$, $det|Res(f, u)| = 0$ over Z and f and u have a common nontrivial factor in $Z[x]$. That finishes the algorithm.

21 Isolation Lemma and Integer Lattices

The following very important (isolation) lemma is due to Mulmuley, Vazirani and Vazirani [17]. In subsection 21.1 we prove the lemma and briefly describe some of its most notable applications. In subsection 21.2 we discuss whether the isolation lemma can be derandomized. We suggest an approach towards a partial derandomization that is based on integer lattices. Throughout the section we use the standard notation $[s] = \{1, \dots, s\}$.

21.1 Isolation lemma

Lemma 7. *Let S be a universe of size n and F be a family of subsets of S. Let $\nu : S \to [2n]$ be a random assignment of integer weights to elements of S. Then,*

$$\Pr[\textit{There exists a unique minimum weight set in } F] \geq 1/2,$$

where the probability is over the choice of ν.

Proof. Let E be the event that there exist two distinct minimum weight subsets f_1 and f_2 in F. For $i \in [n]$, let E_i be the event that there exist two distinct minimum weight subsets f_1 and f_2 in F such that $i \in f_1$ and $i \notin f_2$. Clearly,

$$E \subseteq \cup_{i \in [n]} E_i.$$

Now we prove that for every $i \in [n]$, $\Pr(E_i) \leq 1/2n$. Let i be fixed. Let $\hat{\nu} : S \setminus \{i\} \to [2n]$ be arbitrary. Note that there exists at most a single assignment $\nu : S \to [2n]$ such that:

- for all $j \in S \setminus \{i\} : \nu(j) = \hat{\nu}(j)$;
- there exist two minimum weight subsets $f_1 \neq f_2 \in F$ such that $i \in f_1$ and $i \notin f_2$.

The above observation yields $\Pr(E_i) \leq 1/2n$ and $\Pr(E) \leq 1/2$. $\square$

We now briefly review some applications of the isolation lemma:

1. Historically the first application of the lemma was to show that perfect matching in general graphs can be computed in RNC [17]. We sketch the high-level idea of the proof for the case of bipartite graphs.

   Let $G = (V, E)$ be a bipartite graph with n edges. Let E be our universe. Consider the family of all perfect matchings in G. Pick a random weight assignment $\nu : E \to [2n]$. Lemma 7 implies that with probability at least a half there is a unique perfect matching of minimal weight in G. Let $M \subseteq E$ be such a matching and w be its weight. Let A be the incidence matrix of G, where a location corresponding to an edge e contains the integer $2^{\nu(e)}$. We use the notation B_e to denote the matrix that results from removing the row and the column corresponding to edge e from matrix A. One can argue [17] that the largest power of two dividing $\det A$ is w, and edge e belongs to M if and only if $2^{\nu(e)} \det B_e / 2^w$ is odd. This yields an RNC algorithm to compute perfect matchings.

2. Isolation lemma can be used to reduce SAT to $\mathrm{UNIQUE}-\mathrm{SAT}$ [25, 17].

   The high-level idea of such a reduction is the following. Given a formula $\phi(x_1,\ldots,x_n)$ we treat the set of variables $\{x_1,\ldots,x_n\}$ as our universe U. Every satisfying assignment to ϕ yields a subset $\{x_{i_1},\ldots,x_{i_k}\}\subseteq U$ of variables that are set to 1. We take all such subsets to be our family F. Below is our randomized reduction:

   - Pick a random assignment $\nu:\{x_1,\ldots,x_n\}\to[2n]$ and a random integer $r\in[2n^2]$.
   - Express the following assertion via a boolean formula ϕ' : *There exists* $x\in\{0,1\}^n$ *s.t.* $\phi(x)=1$ *and* $\nu(x)=r$.

   Lemma 7 implies if ϕ is satisfiable than with non-negligible probability ϕ' has a unique satisfying assignment, and if ϕ is not satisfiable than ϕ' is always not satisfiable.

3. In a relatively recent work [19] have used isolation lemma to argue that complexity classes $\mathrm{NL/poly}$ and $\mathrm{UL/poly}$ coincide. Recall that $\mathrm{NL/poly}$ stands for non-deterministic non-uniform log-space and $\mathrm{UL/poly}$ stands for unambiguous non-uniform log-space. The proof of [19] uses isolation lemma to pick a single path (out of exponentially many possible paths) from source to sink in a certain directed graph.

21.2 Derandomizing isolation lemma

It can be shown that isolation lemma can not be fully derandomized (with polynomial weights), i.e., for all constants c_1,c_2 and for all of assignments $\{\nu_i\}_{i\in[n^{c_1}]}$, with range $[n^{c_2}]$, there exists a family F of subsets of $[n]$ such that for every $j\in[n^{c_1}]$ there are two minimal weight subsets of F with respect to ν_j.

This however leaves open the possibility to derandomize many particular uses of the lemma. Specifically, it *may* be possible to replace the random choice of the assignment ν in all three examples from section 21.1 by a deterministic procedure. One reason that makes us believe in possibility of such derandomization is that in all those cases the lemma is used against very special collections of sets. In each case there are at most $2^{n^{O(1)}}$ possible collections of subsets of a universe of size $[n]$ that can arise. (Note that in general there are 2^{2^n} such collections.)

It seems highly desirable to develop some insight into the structure of collections of subsets (arising in specific application of the lemma), since this may be a step towards derandomization. In what follows we show how one can use integer lattices to get some insight of this nature. We deal with the first example from section 21.1 (i.e., perfect matchings in graphs).

Given a simple undirected graph $G=(V,E)$ with $2n$ nodes we treat perfect matchings in G as $\{0,1\}$-vectors in R^{n^2} of Hamming weight n. Let $v_1,\ldots,v_m$ be all such vectors. Consider a new family of $(m-1)$ vectors $\{(v_2-v_1),\ldots,(v_m-v_1)\}$, and let $u_1,\ldots,u_k\in\{(v_2-v_1),\ldots,(v_m-v_1)\}$ be the basis (over integers) of the space spanned by vectors $\{(v_2-v_1),\ldots,(v_m-v_1)\}$. Define L to be the lattice with basis $\{u_1,\ldots,u_k\}$. Define affine lattice $\tilde{L}$ to be the shift of L be the vector v_1, i.e.,

$$\tilde{L}=\{u+v_1\mid u\in L\}.$$

Now we are ready to present our insight into the structure of perfect matchings in graphs.

Lemma 8. *Suppose* $C=\{\tilde{u}\in\tilde{L}\mid\tilde{u}$ *is the closest to the origin*$\}$; *then* C *is precisely the collection of matching vectors.*

Proof. Note that for every $\tilde{u}\in\tilde{L}$ the L_1 norm of $\tilde{u}$ is n. Thus the length of $\tilde{u}$ is at least $\sqrt{n}$. It remains to notice that vectors of minimal length $\sqrt{n}$ necessarily contain exactly n ones and n^2-n zeros and correspond to matchings in G. ∎

22 Factoring and Integer Lattices

This part of the lecture will necessarily be more philosophical. The basic idea will be to review current factorization approaches and see how they can be formulated as finding an automorphism of a finite ring. Formulated in this way, we see how this approach might be broadened to bring in some new ideas—for example, looking at the problem in a larger ring, or employing integer lattices. The factorization of polynomials with rational coefficients uses lattices, and we expect integer factorization to be harder, so lets try lattices here too! Manindra then details an instantiation of these ideas to show what such an approach might look like.

22.1 Classical approach

The most important classical factoring algorithms are elliptic curve factoring, the quadratic sieve, and the number field sieve. The fastest classical algorithm is the general number field sieve, with running time

$$O(\exp((\frac{64}{9}\log n)^{1/3}(\log\log n)^{2/3}))$$

A good overview of these techniques is given by Arjen Lenstra's survey [11].

We won't go into the details of these approaches, except to say that both the quadratic and number field sieves aim to find a nontrivial solution to the congruence of squares. That is, if n is the number we wish to factor, then in this approach we look for a, b such that

$$a^2 \equiv b^2 \mod n. \tag{7}$$

This means $n \mid a^2 - b^2$ and so $n \mid (a+b)(a-b)$. A nontrivial solution means $a + b \not\equiv 0 \mod n$ and $a - b \not\equiv 0 \mod n$. In this case, the factors of n will be split between $a+b$ and $a-b$, and we can find a factor of n by computing $\gcd(n, a+b), \gcd(n, a-b)$.

We now see how this approach can be formulated as an automorphism problem. Let $R = \mathbb{Z}_n[x]/(x^2-1)$. This ring has two trivial automorphisms, the identity and the mapping $x \mapsto -x$.

Claim 9. *Let n be odd. Then finding a solution to (7) is equivalent to finding a nontrivial automorphism of R.*

Proof. First, let a, b be such that $a^2 \equiv b^2 \mod n$ and $a \not\equiv \pm b \mod n$. A basis for the ring is $1, x$, thus we just specify the action of the automorphism on the basis elements. It is straightforward to verify that $\phi(1) = 1$ and $\phi(x) = ax/b$ gives a nontrivial automorphism of R.

For the other direction, let ϕ be a nontrivial automorphism of R, and say that $\phi(x) = ax + b$. We first claim that a and n are co-prime. Let $d = (a, n)$ and consider

$$\phi(\frac{n}{d}x) = \frac{n}{d}ax + \frac{n}{d}b = \frac{n}{d}b \mod n.$$

We also have, however,

$$\phi(\frac{n}{d}b) = \frac{n}{d}b.$$

As ϕ is one-to-one, we must have $nx/d \equiv= nb/d \mod n$, which means $d = 1$.

Now consider

$$0 = \phi(x^2-1) = \phi(x)^2 - 1 = (ax+b)^2 - 1 = 2abx + a^2 + b^2 - 1.$$

This implies that $2ab \equiv 0 \mod n$ and $a^2 + b^2 = 1 \mod n$. As $(a, n) = 1$ and n is odd, the first equation implies $b \equiv 0 \mod n$. Thus the second equation gives $a^2 \equiv 1 \mod n$. As ϕ is not a trivial automorphism, $a \neq \pm 1$ and we have found a nontrivial solution to the congruence of squares. □

Thus we see that the congruence of squares approach to factorization is equivalent to finding a nontrivial automorphism of the ring $\mathbb{Z}_n[x]/(x^2-1)$.

22.2 Larger rings and lattices

In light of this equivalence, Manindra suggests trying some tricks that have been successful for these types of problems—namely looking at larger rings and trying to use lattice algorithms.

What if we look at a larger ring? For example, consider the ring $\mathbb{Z}_n[x]/(x^r-1)$. One should think of a small r (logarithmic, poly-log) here, as used in the primality testing algorithm.

Let n be the number we wish to factor, and suppose $n = pq$ is the product of two primes. We consider polynomials of the form

$$P_a = (a+x)^n - a^n - x^n \pmod{n, x^r - 1}.$$

Expanding the term $(a+x)^n$, we find

$$P_a(x) = \sum_{j=1}^{n-1} \binom{n}{j} x^j a^{n-j} \pmod{n, x^r - 1},$$

where the first and last terms have canceled.

Now we use the fact that n is a product of two primes. Notice that if $j \not| p$ and $j \not| q$, then $\binom{n}{j} \equiv 0 \mod n$. Thus we can write $P_a(x)$ as

$$P_a(x) = \sum_{j=1}^{p-1} \binom{n}{jq} x^{jq} a^{n-jq} + \sum_{j=1}^{q-1} \binom{n}{jp} x^{jp} a^{n-jp} \pmod{n, x^r - 1}.$$

Further, notice that $\binom{n}{jq} \equiv \binom{p}{j} \mod n$. Thus

$$P_a(x) = \sum_{j=1}^{p-1} \binom{p}{j} x^{jq} a^{n-jq} + \sum_{j=1}^{q-1} \binom{q}{j} x^{jp} a^{n-jp} \pmod{n, x^r - 1}.$$

Now we look at the reduction modulo $x^r - 1$. This will cause some of the binomial coefficients to be compressed and added together.

$$P_a(x) \mod n, x^r - 1 \equiv \sum_{i=0}^{r-1} c_{i,a} x^i.$$

Take the polynomials $P_a(x)$ for k many values of a. We associate a lattice to these polynomials of dimension $r + k$.

$$L = \left[\begin{array}{cccc|cccc}
n & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
0 & n & \ddots & 0 & 0 & 0 & \ddots & 0 \\
0 & 0 & \ddots & 0 & 0 & 0 & \ddots & 0 \\
0 & 0 & \dots & n & 0 & 0 & \dots & 0 \\
\hline
\cdots & P_{a_1}(x) & \cdots & & 1 & 0 & \dots & 0 \\
\cdots & P_{a_2}(x) & \cdots & & 0 & 1 & \dots & 0 \\
\cdots & \vdots & \cdots & & 0 & 0 & \ddots & 0 \\
\cdots & P_{a_k}(x) & \cdots & & 0 & 0 & \dots & 1
\end{array}\right]$$

Here the top left block is the r-by-r matrix nI, and the bottom right block is the k-by-k identity matrix. As this matrix is lower triangular, its determinant is simply n^r. Thus by the Minkowski bound, there is a vector in the lattice of length at most $\sqrt{r+k} n^{r/r+k}$. The Lenstra, Lenstra, Lovász algorithm will then return a vector of length at most $2^{(r+k)/2} n^{r/r+k}$. The hope is that some coefficients of this short vector will have nontrivial gcd with n. The intuition for this hope is that the first r rows should not contribute to the shortest vector, and a natural way to make a short vector is to multiply each P_{a_i} by p. The downside to this is that this will create p's on the bottom right diagonal, which can boost the length of this vector when p is large, say for $p \approx \sqrt{n}$. Manindra says that he has done some experiments with this approach and it has thus far not worked too well. If you want to try it out yourself, Manindra highly recommends Shoup's number theory library `www.shoup.net`, which contains an implementation of the LLL algorithm.

References

[1] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. *Annals of Mathematics*, **160**:781–793, 2004.

[2] M. Ajtai. The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract). In *ACM Symposium on Theory of Computing*, pages 10–19, 1998.

[3] E. Allender and M. Ogihara. Relationships among PL, #L, and the determinant. *RAIRO - Theoretical Information and Applications*, 30:1–21, 1996.

[4] J.-Y. Cai, M. Fürer and N. Immerman. An optimal lower bound on the number of variables for graph identifications. *Combinatorica*, **12**(4), pp. 389–410, 1992.

[5] R.A. DeMillo and R. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978.

[6] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra.* Cambridge University Press, Cambridge, 1999.

[7] J. Hastad. Solving simultaneous modular equations of low degree. *SIAM J. Comput.*, 17(2):336–341, 1988.

[8] R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

[9] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1–2):1–46, 2004. (preliminary version in STOC'03).

[10] N. Kayal and N. Saxena. Polynomial identity testing for depth 3 circuits. In *Proc. 21st IEEE Conference on Computational Complexity*, pages 9–17, 2006.

[11] A. Lenstra. Integer factorization. *Designs, Codes and Cryptography*, 19:101–128, 2000.

[12] A.K. Lenstra, H.W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[13] L. Lovász. On the structure of factorizable graphs. *Acta Mathematica Academiae Scientiarum Hungaricae*, 23:179–195, 1972.

[14] V. Mahajan and V. Vinay. A combinatorial algorithm for the determinant. *Chicago Journal on Theoretical Computer Science*, 1997. Article 5.

[15] D. Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. In *IEEE Symposium on Foundations of Computer Science*, pages 92–98, 1998.

[16] H. Minkowski. *Geometrie der Zahlen.* BG Teubner, 1910.

[17] K. Mulmuley, U. Vazirani, and V. Vazirani. Matching is as easy as matrix inversion. *Proc. 19th ACM Symposium on Theory of Computing,* pp. 345-354, 1987.

[18] N. Nisan. Lower bounds for non-commutative computation. In *Proc. 23rd ACM Symposium on Theory of Computing*, pages 410–418, 1991.

[19] K. Reinhardt and E. Allender. Making nondeterminism unambigious. *SICOMP*, 29:1118–1131, 2000.

[20] J.T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, October 1980.

[21] V. Shoup. *A computational introduction to number theory and algebra.* Cambridge University Press, 2005.

[22] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, October 1991.

[23] L.G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8, 1979.

[24] L.G. Valiant. Completeness classes in algebra. In *Proc. 11th ACM Symposium on Theory of Computing*, pages 249–261, 1979.

[25] L.G. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85-93, 1986.

[26] B. Weisfeiler and A.A. Lehman. Reduction of a graph to a canonical form and an algebra which appears in the process. *Nauchno-Tekhn. Inform. Sb. VINITI* Ser. 2, 1968, no. 9, 12-16. (Russian)

[27] R. Zippel. Probabilistic algorithms for sparse polynominals. In Edward W. Ng, editor, *Proceedings of the International Symposium on Symbolic and Algebraic Manipulation (EUROSAM '79)*, volume 72 of *LNCS*, pages 216–226, Marseille, France, June 1979. Springer.