**CS681** **Computational Number Theory**

# Lecture 5 and 6: Chinese Remaindering

*Instructor: Piyush P Kurur* *Scribe: Ramprasad Saptharishi*

## Overview

In the next two lectures, we shall see a very important techinique used in computer science. The technique is called *Chinese Remaindering*. This comes in extremely handy in various arithmetic problems. We shall be looking at the problem of evaluating the determinant as a motivation.

## 1 Motivation for CRT: The Determinant

We are given an integer square matrix $A$ and we are to find the determinant of the matrix. Before we talk about solving the problem, we need to understand the input as such. How big is the input?

The size is not just $n^2$ since the entries of the matrix also need to be represented. Hence the input size also depends on the size of the entries in the matrix. Let us call $\lambda = \max_{i,j} |a_{ij}|$. Then each entry in the matrix requires atmost $\log \lambda$ bits and there are $n^2$ entries. Thus, the input size is $n^2 \log \lambda$. We are looking for an algorithm that runs in time polynomial in the input size.

The naive approach is to do Gaussian Elimination, or the elementary row-operation method done in high-school: pick the first non-zero element in the first row, divide that row by the number (making it 1), and use this row to clear all other entries in that row.

This however has two problems:

1. Involves division and hence manipulating rational numbers.
2. Numbers in the matrix can become huge during gaussian elimination.

Firstly we need to understand why the first point is really a problem. We are given a matrix with just integer entries. We shall see now that that such a matrix will have an integer determinant. Thus, it may not be efficient to have rational number manipulation.

## 1.1 Integer Matrices have Integer Determinants

The group of permutations over $n$ indices is denoted by $S_n$. Given any permutation in $S_n$, we can talk about the sign of the permutation. The definition is based on the fact that every permutation can be written as a product of cycles of length 2.

**Lemma 1.** *Every permutation $\sigma$ can be written as a product of disjoint cycles.*

*Proof.* Start with the index 1. Look at the image of 1 which is $\sigma(1)$ and its image $\sigma(\sigma(1))$ etc. Eventually some $\sigma^i(1) = 1$ since the number of elements is finite. Hence this corresponds to the cycle $\left(1\ \sigma(1)\ \sigma^2(1)\ \cdots\ \sigma^i(1)\right)$. Now look at the next smallest index that has not been covered in this cycle and do the same. Our permutation $\sigma$ is the product of these cycles and they are clearly disjoint. □

**Lemma 2.** *Every permutation $\sigma$ can be written as a product of cycles of length 2.*

*Proof.* By the previous lemma, it is enough to show that every cycle can be written as a product of 2-cycles. And this is very easy to see. Consider any cycle of the form $(a_1\ a_2\ \cdots\ a_k)$. Easy to check that this is equal to the product $(a_1\ a_2)(a_1\ a_3)\cdots(a_1\ a_k)$. □

Now, if a permutation $\sigma$ can be represented as a product of $m$ 2-cycles, then we define the sign of $\sigma$ to be $(-1)^m$. An immediate question is whether this is well defined. That is, suppose a permutation can be represented as a product of 7 such 2-cycles and also as a product of 24 2-cycles in a different way, won't it give two conflicting values for the sign of the permutation. The answer is that such a thing won't happen. It is not too hard to check but we leave this to the interested reader.

(Hint: Every permutation of $n$ indices can be thought of as a tuple $(x_1, \cdots, x_n)$ where the $i$-th index corresponds to the image of $i$ under the permutation. Now look at the sign of the expression

$$\prod_{i>j}(x_i - x_j)$$

The sign of this expression is an equivalent definition of the sign of the permutation.

Now can you see why a permutation cannot be expressed as a product of $s$ transpositions and $t$ transpositions where $s$ and $t$ are of different parity?)

This is another formula to evaluate the determinant.

$$detA = \sum_{\sigma \in S_n} sign(\sigma) \prod_{i=1}^{n} a_{i\sigma(i)}$$

As an example, any $2 \times 2$ matrix has its determinant as $a_{11}a_{22} - a_{12}a_{21}$ where the first term corresponds to the permutation $(1)(2)$ and the second to the permutation $(1\ 2)$.

It is clear from the above formula that a determinant of an integer matrix has to be an integer.

### 1.2 First Attempt: An Euclidian Approach

Since we are assured that the answer is going to be an integer, it doesn't make much of sense to use rational numbers during our computation. And besides, the denominators can grow really huge through successive row operations.

But the problem of division can be sorted out using a Euclid's Algorithm sort of approach. Consider the first row of the matrix. Suppose each element is a multiple of the least entry, then we are in good shape. There would be no need to divide at all. How do we make sure that this happens? Somehow get the gcd of the numbers as one of the entries!

Pick up the least element in the row, say $a_{11}$. Now every other $a_{i1} = qa_{11} + r$. Now subtract $q$ times the first row from the $i$-th row. This essentially reduces every entry to the remainder when divided by $a_{11}$. Now continue this procedure by picking up the least element until you get the gcd of the numbers and then use it to kill every other entry in the row.

But, this still causes numbers to blow up. While we do operations to work on the first column, the other entries can grow to become too large.

### 1.3 Second Attempt: The Big Primes Method

One clever trick is to do all computations modulo a prime large enough. Since we know that the determinant is equal to $\sum sign(\sigma) \prod a_{i\sigma(i)}$, this value is atmost $n!\lambda^n$ since there are $n!$ terms and each term can be atmost $M = \lambda^n$. Now choose a prime $P$ larger than this bound $M$.

Now division reduces to multiplying by the inverse modulo $P$ and this can be done efficiently (this is the reason we want our $P$ to be a prime. We can't choose any arbitrary number since inverses may not exist). Now the

gaussian elimination works and numbers will be bounded by $P$. Gaussian elimination modulo this prime $P$ will give us a final answer $D$ that is in the range $[0, P-1]$. But this could still mean that there are two choices for the integer determinant. The determinant could either be $D$ or $D-P$. So to get around this small catch, we choose $P$ to be a prime larger than $2M$. In this way, if the value we get is less than $P/2$, we know it is the determinant. Else, it will be $D-P$.

Thus we can solve the determinant problem by doing all computations modulo a large prime. But how do we get a large prime? How do we find a prime larger than the bound $M$ quickly?

By a theorem on the density of primes, a random number between $m$ and $2m$ is a prime with reasonably good probability. Thus we can just pick a random number, test if it is prime, if not pick again. We will hit a prime soon enough.

This however introduces randomness in our algorithm. We would like to have a deterministic polynomial time algorithm. This is where Chinese Remaindering comes in.

## 2 Chinese Remainder Theorem: Over Integers

**Theorem 3.** *Let $N = a_1 a_2 \cdots a_k$ such that each pair $a_i, a_j$ are coprime. Then we have the following isomorphism between the two rings.*

$$\mathbb{Z}/(N\mathbb{Z}) \cong \mathbb{Z}/(a_1\mathbb{Z}) \times \mathbb{Z}/(a_2\mathbb{Z}) \times \cdots \mathbb{Z}/(a_k\mathbb{Z})$$

*And more so, the isomorphism and the inverse map are computable easily.*

*Proof.* First we look at the following homomorphism

$$\begin{array}{rcl} \phi : \mathbb{Z} & \longrightarrow & \mathbb{Z}/(a_1\mathbb{Z}) \times \mathbb{Z}/(a_2\mathbb{Z}) \times \cdots \mathbb{Z}/(a_k\mathbb{Z}) \\ x & \mapsto & (x \bmod a_1, x \bmod a_2, \cdots, x \bmod a_k) \end{array}$$

It is easy to check that this is indeed a homomorphism of rings. What is the kernel of the map? We are looking at the inverse image of $(0, 0, \cdots, 0)$. This just means that any $x$ in the kernel must be $0 \bmod a_i$ for each $i$. And since the $a_i$'s are coprime, this inturn means that $x$ must be divisible by $N$. Thus the kernel of this map is $(N\mathbb{Z})$.

Hence, by the first isomorphism theorem, we have that the induced quotient map is an injective homomorphism:

$$\hat{\phi} : \mathbb{Z}/(N\mathbb{Z}) \hookrightarrow \mathbb{Z}/(a_1\mathbb{Z}) \times \mathbb{Z}/(a_2\mathbb{Z}) \times \cdots \mathbb{Z}/(a_k\mathbb{Z})$$

It's just left to show that the map is not only injective but also surjective; that would establish that it is indeed a homomorphism. Since the rings in the picture are finite rings, we can use a cardinality argument here. The cardinality of the ring on the left is $N$ and so is the cardinality of the ring on the right $N$ (since it is equal to $a_1 a_2 \cdots a_n$). Hence, since the map is injective between two sets of the same finite cardinality, it has to be an isomorphism.

This however will now help when the rings are infinite. For example $\mathbb{R}$ happily sits injectively inside $\mathbb{C}$ but they are clearly not isomorphic. We shall soon be getting to a general setting when such a cardinality argument won't work. Thus we need a more algebraic proof.

Here we shall use a small lemma.

**Lemma 4.** *We can easily compute elements $x_i$ such that $x_i = 1 \bmod a_i$ and $x_i = 0 \bmod a_j$ for all $i \neq j$. In other words, the image of $x_i$ is the tuple that has $1$ on the $i$-th coordinate and $0$ everywhere else.*

*Pf:* Since all the $a_i$'s are pairwise coprime, $a_i$ is coprime to $\bar{a}_i = \prod_{j \neq i} a_j$. Thus, by euclid's lemma, there exists elements $x$ and $y$ such that

$$x a_i + y \bar{a}_i = 1$$

Going modulo $a_i$, we get $y\bar{a}_i = 1 \bmod a_i$. And since $y\bar{a}_i$ is divisible by each other $a_j$, it is $0 \bmod a_j$. Thus this number $y\bar{a}_i$ is our required $x_i$ and hence can be computed easily by the extended euclid's algorithm. □

Now that we have these $x_i$'s, computing the inverse map is very simple. Given a tuple $(z_1, z_2, \cdots, z_k)$, the inverse image is just $\sum_{i=1}^{k} z_i x_i$.

Thus the map $\hat{\phi}$ is indeed an isomorphism and its image and inverse images can be computed easily. □

## 2.1 Solving Determinant through CRT

We are going to pick up small primes $p_1, p_2, \cdots, p_m$ such that $\prod p_i = N > 2M$ and then use chinese remaindering. How many primes do we need to pick? Since each prime is greater than or equal to two, the product of $m$ distinct primes is clearly greater than $2^m$. Thus in order to go larger than

$2M$, we just need to pick $\log 2M$ primes, which is $O(n \log n + n \log \lambda)$ and is clearly polynomial in the input size.

How do we go about picking them? Just keep testing numbers from 2 onwards, check if it is prime, and do this until we have enough primes. How long do we have to go before we get enough primes?

The prime number theorem tells us that the number of primes less than $n$ is $O\left(\frac{n}{\log n}\right)$. With a little bit of calculations, it is easy to see that we would have found our $m$ primes if we go just up till $m^2 \log^2 m$. And since $m$ is polynomial in the input size, so is $m^2 \log^2 m$.

So we just need to look at all numbers up till $m^2 \log^2 m$ where $m = \log(2n!\lambda^n)$, and pick up all the primes. Note all these primes are extremely small, even their magnitude smaller than $m$ which is about $\log M$. In the big prime method, we were picking a prime that required $\log M$ bits to even represent it in binary; its magnitude was about $2^{\log M} = M$. These primes are logarithmically smaller. Hence, to check if the numbers are really primes, you can use even the extremely inefficient exponential time sieve method or something. It also makes sense to store these small primes in the library, precompute them and keep it.

Now that we have these primes, we compute the $x_i$'s as indicated in the lemma using the extended euclid's algorithm. Now we compute the determinant of the matrix $A$ modulo each of these primes $p_i$ using gaussian elimination. Let us say we get our value of the determinant mod $p_i$ as $d_i$. Once you have done this calculation for each $p_i$, we get the tuple $(d_1, d_2, \cdots, d_m)$. Now using the $x_i$'s, find the inverse map to get the value of the determinant $D$ mod $N$. If this value is less than $N/2$, return it. Else, return $D - N$.

## 3 Chinese Remainder Theorem for Arbitrary Rings

In order to state the theorem for arbitrary rings, we need analogues of divisibility and coprimeness in terms of rings. This can be done using ideals. We say $m \mid n$, or $m$ divides $n$, if every multiple of $n$ is also a multiple of $m$. This interms of ideals translates to $m\mathbb{Z}$ containing the ideal $n\mathbb{Z}$.

As for coprimes, we know that two numbers $a, b$ are coprime if there exist $x, y$ such that $xa + yb = 1$. For this, we need a notion of an ideal sum.

Given ideals $\mathfrak{a}, \mathfrak{b}$ of a ring $R$, we define the sum-ideal as

$$\mathfrak{a} + \mathfrak{b} = \{a + b \ : \ a \in \mathfrak{a}, b \in \mathfrak{b}\}$$

This is also the ideal generated by the union of the two ideals.

**Algorithm 1** DETERMINANT: USING CRT

Let $M = n!\lambda^n$ and $m > 2\log M$.
Enumerate the first $m^2 \log^2 m$ numbers and check for primes. Pick the first $m$ primes.
Let $N = p_1 p_2 \cdots p_m$.
**for** $i = 1$ to $m$ **do**
    Evaluate, using gaussian elimination, $det(A) \bmod p_i$.
    Let $d_i = det(A) \bmod p_i$.
    Evaluate, using extended euclid's algorithm, the $x_i$ as in the lemma.
**end for**
Let $D = \sum_{i=1}^{m} x_i d_i$.
**if** $D < N/2$ **then**
    **return** $D$.
**else**
    **return** $D - N$.
**end if**

And now we can say that two ideals $\mathfrak{a}$ and $\mathfrak{b}$ are coprime if the ideal $\mathfrak{a} + \mathfrak{b} = R$.

Using these definitions, we have the Chinese Remainder Theorem for arbitrary rings. We state and prove the theorem for two ideals, but the general case is similar.

**Theorem 5.** *Let $R$ be any commutative ring with identity. Let $\mathfrak{a}$ and $\mathfrak{b}$ be two ideals of $R$ that are coprime to each other. Then we have the following ring isomorphism:*

$$R/(\mathfrak{a} \cap \mathfrak{b}) \cong R/\mathfrak{a} \times R/\mathfrak{b}$$

*Proof.* The proof is almost exactly the same as the proof of CRT over integers. Consider the following homomorphism.

$$\begin{aligned} \phi : R &\longrightarrow R/\mathfrak{a} \times R/\mathfrak{b} \\ x &\mapsto (x \bmod \mathfrak{a}, x \bmod \mathfrak{b}) \end{aligned}$$

Here, mod corresponds to the coset that contains $x$. This again is clearly a homomorphism. And the kernel is the set of all elements that ar $0 \bmod \mathfrak{a}$ and $0 \bmod \mathfrak{b}$ which means that the element is present in both $\mathfrak{a}$ and $\mathfrak{b}$. Thus the kernel of the homomorphism is $\mathfrak{a} \cap \mathfrak{b}$.

Thus all that's left to do is to show that the quotient map

$$\hat{\phi} : R/(\mathfrak{a} \cap \mathfrak{b}) \hookrightarrow R/\mathfrak{a} \times R/\mathfrak{b}$$

is also surjective. To show that, we construct the inverse map.

We need to find the inverse image of any given tuple $(x, y)$. Since we know that the two ideals are coprime, $\mathfrak{a} + \mathfrak{b} = R$ and in particular contains the identity element $1$.

Hence there exists two elements $a \in \mathfrak{a}, b \in \mathfrak{b}$ such that $a + b = 1$. Just as in the integer case, if we go modulo $\mathfrak{b}$ we see that $a$ is an element that is $0 \bmod \mathfrak{a}$ but $1 \bmod \mathfrak{b}$ and similarly the element $b$. Now consider the element $xb + ya$. This is easily seen to be $x \bmod \mathfrak{a}$ and $y \bmod \mathfrak{b}$. Hence $xb + ya$ is the inverse image of $(x, y)$.

Thus the map is also surjective and hence is indeed an isomorphism. □

Thus any ring that lets us compute the elements $a, b$ such that $a + b = 1$ will allow CRT to go through. One such example is the ring of polynomials over integers. In this ring, the irreducible polynomial will act as the primes and we will be able to compute the determinant of a matrix with polynomial entries as well.

Another important property is that the units go to the units. That is, if you take any element $x$ in the ring $R/(\mathfrak{a} + \mathfrak{b})$ whose inverse exists, corresponding tuple $(a, b)$ also have the property that $a$ and $b$ are invertible. The proof of this is pretty easy.

Suppose an invertible element $x$ was mapped to $(a, b)$. Then, since the inverse exists, look at the image of the inverse of $x$. Lets call it $(a', b')$. By definition, $1 = xx^{-1} \mapsto (aa', bb') = (1, 1)$. And hence $a'$ must be the inverse of $a$ and $b'$ the inverse of $b$. And thus if $x$ is invertible, so is $a, b$ and viceversa.

Chinese Remainder Theorem is a really powerful tool used in numerous occasions in computer science. We shall see more in the days to come.