CME 305: Discrete Mathematics and Algorithms
Instructor: Professor Aaron Sidford (sidford@stanford.edu)
February 1, 2018

Lecture 8 - Algebraic Methods for Matching[1]

In the last lecture we showed that given an algorithm that can determine whether or not a determinant of a matrix with variable entries over some field is non-zero we can determine whether or not a graph has a perfect matching. In this lecture we show how to use this fact to compute a perfect matching provided we can compute the determinant of an integer valued matrix with bounded entries and we show how to do this efficiently using fast matrix multiplication (FMM), a common algorithmic primitive equivalent to many natural problems in this area. In the next lecture we will discuss this equivalence further.

1 From Determinants to Matchings

In the the last lecture notes we proved the following lemma.

Lemma 1. *$G = (V, E)$ with vertices $V = \{1, ..., n\}$ has a perfect matching if and only if for variables x_{ij} for $ij \in [n]$ the Tutte matrix $\boldsymbol{T}(G)$ defined for all $i, j \in [n]$ by*[2]

$$\boldsymbol{T}(G)_{ij} = \begin{cases} 0 & \text{if } \{i,j\} \notin E \\ x_{ij} & \text{if } i < j \text{ and } \{v_i, v_j\} \in E \\ -x_{ij} & \text{if } i > j \text{ and } \{v_i, v_j\} \in E \end{cases}$$

satisfies $\det(T(G)) \neq 0$.

This lemma shows that we can check whether or not a graph has a perfect matching by checking whether or not the multivariate polynomial corresponding to the Tutte matrix is the 0 polynomial. So how do we test this?

1.1 The Schwatz Zippel Lemma

As we discussed last class, a natural idea is simply to compute the determinant of $T(G)[r_{11}, ..., r_{nn}]$, i.e. the determinant of $T(G)$ where x_{ij} is set to r_{ij}, for random values of r_{ij}. This corresponds to evaluating the multivariate polynomial at a random point. If $\det(T(G)) = 0$ then for all r_{ij} it must be $\det(T(G)[r_{11}, ..., r_{nn}]) = 0$. However, if not then we would hope there is some chance that $\det(T(G)[r_{11}, ..., r_{nn}]) \neq 0$, in which case we can conclude that G does have a perfect matching. To bound the probability that this happens, we use a classic result known as the Schwartz Zippel lemma, given as follows:

Lemma 2 (Schwartz Zippel Lemma)**.** *Let $f \in \mathbb{F}[x_1, ..., x_n]$ be a non-zero polynomial of degree d over field $\mathbb{F}$.*[3] *Let S be a finite subset of $\mathbb{F}$ and suppose $r_1, ..., r_n$ are chosen independently at random from S then*

$$\Pr[f(r_1, r_2, ..., r_n) = 0] \leq \frac{d}{|S|}.$$

[1] These lecture notes are a work in progress and there may be typos, awkward language, omitted proof details, etc. Moreover, content from lectures might be missing. These notes are intended to converge to a polished superset of the material covered in class so if you would like anything clarified, please do not hesitate to post to Piazza and ask.

[2] Note that the rule used to determine the sign of x_{ij} does not matter so long as it appears once with each sign.

[3] Throughout these notes, whenever we write $\mathbb{F}$ if you prefer you can set $\mathbb{F} = \mathbb{R}$ losing little in the intuition behind these notes.

Note that if $f \in \mathbb{F}[x]$ was a univariate polynomial of degree d then it has at most d roots, i.e. $f(x) = 0$ for at most d distinct values, and this lemma would trivially follow from the fact that the probability we pick one of these d values is at most d. This lemma says that for a multivariate polynomial, even though it may have many more roots, e.g. $(x_i - 1)(x_2 - 1)$ has degree 2 over the reals but an infinite number of roots, the probability of picking one when each variable is chosen at random is the same as in the one dimensional case.

We prove this lemma in several pieces. First, we formally show the standard fact that a univariate polynomial, i.e. $f \in \mathbb{F}[x]$, has at most $\deg(f)$ roots and then using this we prove the lemma by induction. We begin with the following lemma which essentially proves that long division can be performed on polynomials to reduce their degree, i.e. given a polynomial $f(x)$ that we wish to divide by $d(x)$ we can write $f(x) = q(x) \cdot d(x) + r(x)$ where $r(x)$ should be viewed as the remainder of dividing f by d and q is the amount of d that divided f.

Lemma 3 (Polynomial Division)**.** *Polynomials $f, d \in \mathbb{F}[x]$ with d non-zero has a unique representation as $f(x) = q(x) \cdot d(x) + r(x)$ where $q, r \in \mathbb{F}[x]$ with $\deg(r) < \deg(d)$.*

Proof. We prove by induction on $\deg(f)$. As our base case suppose $\deg(f) < \deg(d)$. If $q \neq 0$ then $\deg(q(x) \cdot d(x)) \geq \deg(d(x))$ and $\deg(q(x) \cdot d(x) + r(x)) \geq \deg(d(x)) > \deg(f)$. Consequently, in this case we must have $q(x) = 0$ and $r(x) = f(x)$ for $f(x) = q(x) \cdot d(x) + r(x)$.

Now, suppose that the claim holds whenever $\deg(f) < \deg(d) + k$ for $k \geq 0$ we show it holds for $k+1$. Without loss of generality we can write uniquely write $q(x) = \alpha_q x^{\deg(q)} + q_1(x)$ where $\alpha_q \in \mathbb{F}$ and $q_1(x) \in \mathbb{F}[x]$ with $\deg(q_1(x)) < \deg(q(x))$. Express $f(x) = \alpha_f x^{\deg(f)} + f_1(x)$, and $d(x) = \alpha_d x^{\deg(d)} + d_1(x)$ similarly. Note that in order for $f(x) = q(x) \cdot d(x) + r(x)$ it must be the case that $\deg(q) + \deg(d) = \deg(f)$. and $\alpha_q \alpha_d = \alpha_f$ and

$$f(x) - \alpha_q x^{\deg(q)} \cdot d(x) = q_1(x) d(x) + r(x)$$

However, $f(x) - \alpha_q x^{\deg(q)} \cdot d(x)$ has degree at least one less than $f(x)$ and thus the result holds by induction. □

We can simplify this lemma slightly in the case when $d(x) = x - a$; this is known as the *polynomial remainder theorem.*

Lemma 4 (Polynomial Remainder Theorem)**.** *Polynomial $f \in \mathbb{F}[x]$ for any $a \in \mathbb{F}$ has a unique representation as $f(x) = q(x) \cdot (x - a) + r$ where $r = f(a)$.*

Proof. By polynomial division we know that f has a unique representation as $f(x) = q(x) \cdot (x - a) + r(x)$ where $\deg(r(x)) < \deg(x - a) = 1$. Consequently $r(x)$ has degree 0 and $r(x) = r$ for some $r \in \mathbb{F}$. However, $f(a) = q(a) \cdot (a - a) + r = r$ implies $r = f(a)$ as desired. □

With this lemma established, formally we say that $d(x)$ divides $f(x)$ if when we try to divide f by d the remainder is 0.

Definition 5 (Divisibility)**.** We say polynomial $f \in \mathbb{F}[x]$ is divisible by $d \in \mathbb{F}[x]$ if and only if there exists $q \in \mathbb{F}[x]$ such that $f(x) = q(x)d(x)$ where equality is in terms of the representation.

With these lemmas and definitions established we see that a polynomial is divisible by $(x - a)$ if and only if $f(a) = 0$. Note that if $f(a) = 0$ and $f(b) = 0$ this also implies that $(x - a)(x - b)$ divides f. Repeating this we see that a polynomial of degree d has at most d roots, i.e. points at which it evaluates to 0, This means that if we want to know if a degree d polynomial is identically 0 it suffices to evaluate it at more than d random points.With this one dimensional version of the Schwarz Zippel Lemma established we prove the full version.

Proof of Lemma 2. We prove by strong induction on the degree n. If $n = 1$ then f is one dimensional and it has at most d roots. The probability that r_1 is exactly one of these roots is exactly $d/|S|$. Now suppose it holds for $n-1$ we show it holds for n. Note that we can write f uniquely as

$$f(x_1, ..., x_n) = \sum_{i=0}^{d} x_1^i f_i(x_2, ..., x_n)$$

where $f_i(x_2, ..., x_n) \in \mathbb{F}[x_2, ..., x_n]$ for all $i \in \{0, 1, ..., d\}$ is a degree at most $d-i$ polynomial. Now let $j \in \mathbb{Z}_{\geq 0}$ be the largest value such that f_j is not identically 0. By our inductive hypothesis (or trivially in the case $j = d$) with probability at most $\frac{d-j}{|S|}$ we have that $f_j(x_2..., x_n) = 0$. When this does not happens, we have that the resulting polynomial in x_1 is degree j. In this case, since x_1 is chosen independently from $x_2, .., x_n$ and since a degree j polynomial has at most j roots we have that $f(x_1, ..., x_n) = 0$ with probability $\frac{j}{|S|}$. By union bound, the probability that either of these events happen is at most $\frac{d}{|S|}$ as desired. □

1.2 Computing Perfect Matchings

With the Schwartz Zippel Lemma and the fact that $\det(T(G)) = 0$ if and only if G does not have a perfect matching we have everything we need to given an algorithm for computing a perfect matching in a general graph in polynomial time with high probability.

Lemma 6. *If for all n we can compute whether the determinant of an $n \times n$ integer valued matrix with polynomially bounded entries is 0 in $O(T_{\det}(n))$ time, for $T_{\det}(n) = \Omega(n^2)$, then we can compute perfect matchings w.h.p. in n-node graphs in $O(n^2 \mathcal{T}_{\det}(n))$ time.*

Proof. First we show that we can determine whether or not a n-node graph G has a perfect matching with high probability in $O(\mathcal{T}_{\det}(n))$ time. To do this we pick $r_{ij} \in \{1, ..., n^c\}$ independently at random for all $i, j \in [n]$ for a constant value of c we will pick later to ensure that the algorithm succeeds with high probability. We then compute $T(G)[r_{11}, ..., r_{nn}]$, this clearly takes $O(n^2)$ time, since we can write down each entry of $T(G)[r_{11}, ..., r_{nn}]$ in $O(1)$. Note that $T(G)[r_{11}, ..., r_{nn}]$ is a integer valued matrix with polynomially bounded values as c is a constant and thus we can compute $\det(T(G)[r_{11}, ..., r_{nn}])$ in $O(T_{\det}(n))$. If $\det(T(G)[r_{11}, ..., r_m]) = 0$ we output that G does not have a perfect matching and if $\det(T(G)[r_{11}, ..., r_m]) \neq 0$ then we say it does. Note that if G does not have a perfect matching we will always output the correct answer as in this case, $\det(T(G)[r_{11}, ..., r_m]) = 0$ as the multivariate polynomial $\det(T(G)) = 0$. However, if G does have a perfect matching then $\det(T(G)) \neq 0$ and since $\det(T(G)) \leq n$ by Schwartz Zippel we have that the probability $\det(T(G)[r_{11}, ..., r_m]) = 0$ is at most $\frac{n}{n^c} = n^{-(c-1)}$. Consequently, in $O(T_{\det}(n))$ we output the correct result with probability at least $1 - \frac{1}{n^{c-1}}$ and since we can pick c to be whatever constant we want this means the algorithm succeeds with high probability in n.

Now, given the ability to check whether or not a graph has a perfect matching in $O(T_{\det}(n))$ w.h.p we can achieve the desired running time. We simply iterate through the edges one at a time, checking if the graph without that edge contains a perfect matching. If the graph does, we remove the edge, if it does not we keep the edge. Note that every time we remove an edge the remaining graph still has a perfect matching and whenever we keep an edge it must be the case that it is in *every* perfect matching. Consequently, we only need to consider each edge once giving a running time of $O(n^2 T_{\det}(n))$ as the graph has at most n^2 edges.[4] □

[4] Note that we do get a better running time when the number of edges m is significantly less than n^2, i.e. we obtain a running time of $O(m \cdot T_{\det}(n))$. However, even faster algorithms are known as we will discuss later.

2 From Matrix Multiplication to Determinants

So how fast can we compute the determinant of a integer $n \times n$ matrix? This is a wide open research question that has been extensively studied for decades. However, as we will see the problem of computing the determinant of a matrix is in many cases equivalent to the problem of solving a conceptually simpler problem, multiplying 2 square matrices. In this section we introduce this problem of fast matrix multiplication and prove one side of this reduction, i.e. that given algorithms for fast matrix multiplication, we can compute the determinant.

2.1 Matrix Multiplication

To check whether or not a determinant is 0 we will show that it suffices to be able to multiple two $n \times n$ matrices and that by obtaining faster algorithms for this problem we can generically improve the running time for computing the determinant. This matrix multiplication is incredibly well studied and as we will discuss equivalent to many problems in linear algebra and combinatorial optimization. Formally we introduce the fast matrix multiplication problem is as follows.

Definition 7 (Fast Matrix Multiplication (FMM))**.** The *fast matrix multiplication problem* is as follows, given two square matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ compute $\mathbf{C} = \mathbf{AB}$. The best known value of ω for which this can be done in $O(n^\omega)$ is known as the *matrix multiplication constant.*.

Now clearly $\omega \geq 2$ as just naively writing down C takes $\Omega(n^2)$ time. Furthermore, clearly $\omega \leq 3$ as computing computing $[\mathbf{AB}]_{ij} = \sum_{k \in [n]} \mathbf{A}_{ik}\mathbf{B}_{kj}$ for any $i, j \in [n]$ can be done in $O(n)$ and doing this for all n^2 values of $i, j \in [n]$ therefore takes $O(n^3)$. Trying to obtain values of ω near 2 has been a central open problem for decades and the current best known bound on the matrix multiplication exponent is as follows.

Theorem 8 (FMM State of the Art)**.** $\omega < 2.373$.

Now proving this theorem is far outside the scope of this class. However, to give a taste of how such results might be shown we will prove a classic result of Strassen that $\omega \leq \log_2(7) < 3$. As motivation for this algorithm, suppose we wanted to compute the product of 2 $n \times n$ matrices $\mathbf{M}_1$ and $\mathbf{M}_2$ recursively, by some sort of divide and conquer approach where we solve a number of smaller subproblems to compute the product. To do this we could write $\mathbf{M}_1$ and $\mathbf{M}_2$ in block form

$$\mathbf{M}_1 = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \text{ and } \mathbf{M}_2 = \begin{bmatrix} \mathbf{E} & \mathbf{F} \\ \mathbf{G} & \mathbf{H} \end{bmatrix}$$

where if n is divisible by 2 then $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F}, \mathbf{G}, \mathbf{H} \in \mathbb{R}^{\frac{n}{2} \times \frac{n}{2}}$ (if not we could simply recurse on non-square matrices or pad $\mathbf{M}_1$ and $\mathbf{M}_2$ with 0's to make their dimensions a power of 2). Now with this we see that

...

$$\mathbf{M}_1\mathbf{M}_2 = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{E} & \mathbf{F} \\ \mathbf{G} & \mathbf{H} \end{bmatrix} = \begin{bmatrix} \mathbf{AE} + \mathbf{BG} & \mathbf{AF} + \mathbf{BH} \\ \mathbf{CE} + \mathbf{DG} & \mathbf{CF} + \mathbf{DH} \end{bmatrix}.$$

This suggests a natural recursive algorithm, we could compute the 8 matrix products $\mathbf{AE}, \mathbf{BG}, \mathbf{CE}, \mathbf{DG}, \mathbf{AF}, \mathbf{BH}, \mathbf{CF}, \mathbf{DH}$ of $n/2$ by $n/2$ matrices recursively and then sum the results in $O(n^2)$ time to obtain the result. If the runtime of this procedure is $T(n)$ then by the master theorem we see that

$$T(n) = O(n^2) + 8 \cdot T(n/2) = O(n^{\log_2 8}) = O(n^3).$$

This unfortunately did not give us an improvement, it just gave us another n^3 time algorithm. To do better we need a smarter recursion and that is what Strassen showed in the following classic result.

Theorem 9 (Strassen)**.** $\omega \leq \log_2 7$.

Proof. As before let us write $\mathbf{M}_1, \mathbf{M}_2 \in \mathbb{R}^{n \times n}$ as

$$\mathbf{M}_1 = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \text{ and } \mathbf{M}_2 = \begin{bmatrix} \mathbf{E} & \mathbf{F} \\ \mathbf{G} & \mathbf{H} \end{bmatrix}$$

for even n where $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F}, \mathbf{G}, \mathbf{H} \in \mathbb{R}^{\frac{n}{2} \times \frac{n}{2}}$. Now, directly checking yields that we can write

$$\mathbf{M}_1\mathbf{M}_2 = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{E} & \mathbf{F} \\ \mathbf{G} & \mathbf{H} \end{bmatrix} = \begin{bmatrix} \mathbf{S}_1 + \mathbf{S}_2 - \mathbf{S}_4 + \mathbf{S}_6 & \mathbf{S}_4 + \mathbf{S}_5 \\ \mathbf{S}_6 + \mathbf{S}_7 & \mathbf{S}_2 - \mathbf{S}_3 + \mathbf{S}_5 - \mathbf{S}_7 \end{bmatrix}$$

where

- $\mathbf{S}_1 = (\mathbf{B} - \mathbf{D}) \cdot (\mathbf{G} + \mathbf{H})$
- $\mathbf{S}_2 = (\mathbf{A} + \mathbf{D}) \cdot (\mathbf{E} + \mathbf{H})$
- $\mathbf{S}_3 = (\mathbf{A} - \mathbf{C}) \cdot (\mathbf{E} + \mathbf{F})$
- $\mathbf{S}_4 = \mathbf{H} \cdot (\mathbf{A} + \mathbf{B})$
- $\mathbf{S}_5 = \mathbf{A} \cdot (\mathbf{F} - \mathbf{H})$
- $\mathbf{S}_6 = \mathbf{D} \cdot (\mathbf{G} - \mathbf{E})$
- $\mathbf{S}_7 = \mathbf{E} \cdot (\mathbf{C} + \mathbf{D})$

Now, this seems quite complicated. But carefully checking this we see that we are only doing 7 matrix multiplications of $n/2 \times n/2$ matrices at the cost of more $O(n^2)$ additions and subtractions. Consequently, if we run this procedure recusively its running time $T(n)$ for $n \times n$ matrices by the master theorem can be shown to be

$$T(n) = O(n^2) + 7 \cdot T(n/2) = O(n^{\log_2 7}).$$

□

Faster results can be achieved by finding larger matrices that are computable using even less multiplications, possibly at the cost of more additions. Even faster results are achieved by finding even more structured matrices that can be multiplied faster in which matrix multiplication can be embedded. This is still an active area of research with people proposing new algorithms and approaches and people proving lower bounds against these approaches. It is open if $\omega = 2$ but so we can leverage this body of research without writing the best value of ω, as it can change, we just write our running times in terms of ω.

2.2 From Matrix Multiplication to Inversion

Now that we have introduced this new tool of FMM we show how to use it to invert an abitrary matrix and through this, check if the determinant is non-zero. For simplicity, in this section we assume $\mathbb{F} = \mathbb{R}$ though there are ways of modifying the techniques in this section to apply more broadly.

Ultimately we want to show that we can compute the inverse of $\mathbf{A}$ in $O(n^\omega)$, however we do this in several steps. First as a warmup, we show how to prove this theorem in the case where $\mathbf{A}$ is *symmetric positive definition (SPD)* as we define below.

Definition 10 (Symmetric Positive Definition)**.** We call a matrix $\mathbf{A}$ *symmetric positive definition (SPD)* if it is *symmetric*, i.e. $\mathbf{A} = \mathbf{A}^\top$ or $\mathbf{A}_{ij} = \mathbf{A}_{ji}$ for all i, j, and positive definite (PD), i.e. $x^\top \mathbf{A} x > 0$ for all $x \neq 0$.

We will discuss more later, this means that $\mathbf{A}$ has an orthonormal basis of eigenvectors and all eigenvalues are positive. One of the nice things about working with SPD matrices is that we can show that they are always invertible.

Lemma 11. *If $\mathbf{A}$ is SPD then it is invertible.*

Proof. Proceed by contradiction and suppose that $\mathbf{A}$ is SPD and not invertible. Then there exists $x \neq 0$ with $\mathbf{A}x = 0$ and therefore $x^\top \mathbf{A} x = 0$ contradicting that $x^\top \mathbf{A} x > 0$. □

Another nice property of SPD matrices is that their submatrices are invertible. using this we can give the following block 2×2 formula for the inverse of a SPD matrix.

Lemma 12. *If $\mathbf{A}$ is SPD and $\mathbf{A} = \begin{bmatrix} \mathbf{M} & \mathbf{B}^\top \\ \mathbf{B} & \mathbf{C} \end{bmatrix}$ for $\mathbf{M} \in \mathbb{R}^{a \times a}$, $\mathbf{B}^{b \times a}$, and $\mathbf{C} \in \mathbb{R}^{b \times b}$ then $\mathbf{M}$ and the matrix $\mathbf{S} = \mathbf{C} - \mathbf{B}\mathbf{M}^{-1}\mathbf{B}^\top$, known as the Shur complement, are SPD and*

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{M}^{-1} + \mathbf{M}^{-1}\mathbf{B}^\top \mathbf{S}^{-1}\mathbf{B}\mathbf{M}^{-1} & -\mathbf{M}^{-1}\mathbf{B}^\top \mathbf{S}^{-1} \\ -\mathbf{S}^{-1}\mathbf{B}\mathbf{M}^{-1} & \mathbf{S}^{-1} \end{bmatrix}.$$

Proof. Note that for all $x \in \mathbb{R}^a$ with $x \neq 0$ we have

$$0 < \begin{pmatrix} x \\ 0 \end{pmatrix}^\top \begin{bmatrix} \mathbf{M} & \mathbf{B}^\top \\ \mathbf{B} & \mathbf{C} \end{bmatrix} \begin{pmatrix} x \\ 0 \end{pmatrix} = x^\top \mathbf{M} x\,.$$

Consequently, $\mathbf{M}$ is SPD and invertible and for all $y \in \mathbb{R}^b$ with $y \neq 0$ we have

$$\begin{aligned} 0 &< \begin{pmatrix} -\mathbf{M}^{-1}\mathbf{B}^\top y \\ y \end{pmatrix}^\top \begin{bmatrix} \mathbf{M} & \mathbf{B}^\top \\ \mathbf{B} & \mathbf{C} \end{bmatrix} \begin{pmatrix} -\mathbf{M}^{-1}\mathbf{B}^\top y \\ y \end{pmatrix} \\ &= y^\top \mathbf{B}\mathbf{M}^{-1}\mathbf{B}^\top y + 2y^\top \mathbf{B}\mathbf{M}^{-1}\mathbf{B}^\top y + y^\top \mathbf{C} y = y^\top \mathbf{S} y \end{aligned}$$

and therefore $\mathbf{S}$ is SPD. Consequently, $\mathbf{S}$ is invertible as well. It can then be checked directly that $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$. □

A similar formula holds for arbitrary matrices, when $\mathbf{M}$ is invertible. With care this can be used to give $O(n^\omega)$ time algorithms for inverting an arbitrary matrices over arbitrary fields directly. .

Now using this lemma we can invert arbitrary PD matrices in $O(n^\omega)$.

Lemma 13. *Can compute inverse of a symmetric PD $n \times n$ matrix $\mathbf{A}$ in $O(n^\omega)$.*

Proof. We simply apply the previous lemma recursively. Note again, we can either assume n is a power of 2 so all matrices are square (by padding the original matrix with an identity block) or simply show that everything works when the matrices are nearly square. In either case we get that if we can compute an inverse in $O(T(n))$ time then since we can simply invert $\mathbf{M}$ in $T(n/2)$ and then form $\mathbf{S}$ in $O(n^\omega)$ and invert it in $T(n/2)$ and then form the rest of the matrix in $O(n^\omega)$ we have by the master theorem and the fact that $\omega > 2$

$$T(n) = O(n^\omega) + 2 \cdot T(n/2) = O(n^\omega)\,.$$

□

Using this we can compute inverse of invertible $\mathbf{A}$

Lemma 14. *We can compute inverse of invertible* $\mathbf{A}$ *in* $O(n^\omega)$

Proof. If $\mathbf{A}$ is invertible then $\mathbf{A}x \neq 0$ for all $x \neq 0$ and therefore $x^\top \mathbf{A}^\top \mathbf{A}x > 0$ and $\mathbf{A}^\top \mathbf{A}$ is SPD. Now since $\mathbf{A}$ and $\mathbf{A}^\top$ are $n \times n$ matrices we can compute $\mathbf{A}^\top \mathbf{A}$ in $O(n^\omega)$. Furthermore, since $\mathbf{A}^\top \mathbf{A}$ is SPD we can compute its inverse in $O(n^\omega)$. Thus we can compute $(\mathbf{A}^\top \mathbf{A})^{-1}\mathbf{A}^\top$ in $O(n^\omega)$. Since $(\mathbf{A}^\top \mathbf{A})^{-1}\mathbf{A}^\top = \mathbf{A}^{-1}$ the result follows. □

Using this we can check if the determinant of $\mathbf{A}$ is nonzero in $O(n^\omega)$.

Theorem 15. *We can compute if* $\det(\mathbf{A}) \neq 0$ *for* $\mathbf{A} \in \mathbb{R}^{n \times n}$ *in* $O(n^\omega)$.

Proof. We can try to compute $\mathbf{A}^{-1}$ in $O(n^\omega)$ and then check if $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$ in $O(n^\omega)$. This succeeds if and only if $\det(\mathbf{A}) \neq 0$. □

Putting everything in these notes together we see that we can compute perfect matchings in a general graph if $O(n^{\omega+2})$. This can be improved to $O(n^\omega)$ but it is outside the scope of this class. If you would like references for it let me know.

Note that many of these reductions can be shown to work in both directions and we investigate this further giving more applications of FMM next lecture.