Matrix Rigidity

Orthogonal Vectors, Hierarchy Theorems

Sasha Golovnev

October 14, 2020

Overview

- Recall that we want $M \in \mathbb{F}_2^{n \times n}$, $M \in \cancel{\mathsf{P}^{\mathsf{NP}}}$ E^{NP}

$$\mathcal{R}_M^{\mathbb{F}_2}(\underline{\varepsilon n}) \geq \Omega(\underline{n^{1+\delta}})\,.$$

Overview

- Recall that we want $M \in \mathbb{F}_2^{n\times n}$, $M \in \cancel{\mathsf{P}^{\mathsf{NP}}}$ E^{NP}

$$\mathcal{R}_M^{\mathbb{F}_2}(\varepsilon n) \geq \Omega(n^{1+\delta})\,.$$

- We'll prove that there is $M \in \mathbb{F}_2^{n\times n}$, $M \in \mathsf{P}^{\mathsf{NP}}$

$$\mathcal{R}_M^{\mathbb{F}_2}(2^{\log n/\log\log n}) \geq \Omega(n^2)\,.$$

$n^{\varepsilon} = 2^{\varepsilon \log n}$

Overview

- Recall that we want $M \in \mathbb{F}_2^{n \times n}$, $M \in \mathsf{P}^{\mathsf{NP}}$

$$\mathcal{R}_M^{\mathbb{F}_2}(\varepsilon n) \geq \Omega(n^{1+\delta}) .$$

- We'll prove that there is $M \in \mathbb{F}_2^{n \times n}$, $M \in \mathsf{P}^{\mathsf{NP}}$

$$\mathcal{R}_M^{\mathbb{F}_2}(2^{\log n / \log \log n}) \geq \Omega(n^2) .$$

- We'll use

Overview

- Recall that we want $M \in \mathbb{F}_2^{n \times n}$, $M \in \mathsf{P}^{\mathsf{NP}}$

$$\mathcal{R}_M^{\mathbb{F}_2}(\varepsilon n) \geq \Omega(n^{1+\delta}) .$$

- We'll prove that there is $M \in \mathbb{F}_2^{n \times n}$, $M \in \mathsf{P}^{\mathsf{NP}}$

$$\mathcal{R}_M^{\mathbb{F}_2}(2^{\log n / \log \log n}) \geq \Omega(n^2) .$$

- We'll use
  - Orthogonal Vectors – Fine-grained complexity

Overview

- Recall that we want $M \in \mathbb{F}_2^{n\times n}$, $M \in \mathsf{P}^{\mathsf{NP}}$
$$\mathcal{R}_M^{\mathbb{F}_2}(\varepsilon n) \geq \Omega(n^{1+\delta}) \,.$$
- We'll prove that there is $M \in \mathbb{F}_2^{n\times n}$, $M \in \mathsf{P}^{\mathsf{NP}}$
$$\mathcal{R}_M^{\mathbb{F}_2}(2^{\log n/\log\log n}) \geq \Omega(n^2) \,.$$
- We'll use
    - Orthogonal Vectors
    - Non-deterministic Hierarchy Theorem

Overview

- Recall that we want $M \in \mathbb{F}_2^{n \times n}$, $M \in \mathsf{P}^{\mathsf{NP}}$

$$\mathcal{R}_M^{\mathbb{F}_2}(\varepsilon n) \geq \Omega(n^{1+\delta}) \, .$$

- We'll prove that there is $M \in \mathbb{F}_2^{n \times n}$, $M \in \mathsf{P}^{\mathsf{NP}}$

$$\mathcal{R}_M^{\mathbb{F}_2}(2^{\log n / \log \log n}) \geq \Omega(n^2) \, .$$

- We'll use
    - Orthogonal Vectors
    - Non-deterministic Hierarchy Theorem
    - Rectangular PCPs

Orthogonal Vectors

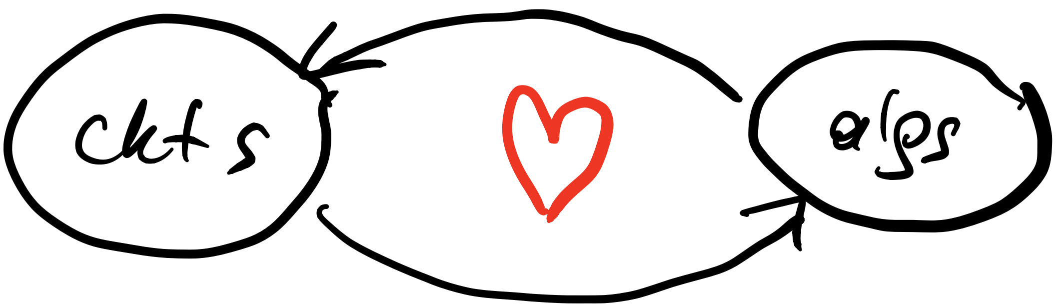
Orthogonal Vectors

$s = (0, 1, 0, 0) \qquad t = (1, 0, 0, 1)$

$\langle s, t \rangle = s_1 t_1 + s_2 t_2 + s_3 t_3 + s_4 t_4 = 0$

Definition

S, T are sets of $\underline{N}$ vectors from $\{0, 1\}^{\boxed{d}}$. Are there $s \in S$ and $t \in T$ such that $\langle s, t \rangle = \sum_{i=1}^{d} s_i \cdot t_i = 0$? (over $\mathbb{Z}$).



Complexity of OV

- Can be solved in $O(N^2 d)$

N go over $s \in S$
N go over $t \in T$
Time d to compute $\langle s, t \rangle$

$N^2 \cdot d$

Complexity of OV

- Can be solved in $O(N^2 d)$
- Can be solved in $O(N \cdot 2^d)$

Complexity of OV

- Can be solved in $O(N^2 d)$
- Can be solved in $O(N \cdot 2^d)$
- Conjecture: no algorithm can solve OV in time $N^{2-\varepsilon}\,\mathrm{poly}(d)$ for constant $\varepsilon > 0$

essentially optimal.

$N^{1.99}$

k-SAT

$(x_1 \vee \bar{x}_2 \vee \dots \vee x_k)$

n variables

m clauses of length k.

Want to check φ is satisfiable.

$$\boxed{2^n \cdot \mathrm{poly}(n, m)}$$

2^n brute force all possible assignments to n variables.

For k-SAT $\quad 2^{n\left(1 - \frac{1}{O(k)}\right)}$

$k \to \infty$ still 2^n alg.

SETH [IP01]

Strong Exponential Time Hypothesis :

No alg run. time $2^{0.99n}$ can solve SAT

SETH $\Rightarrow$ OV cannot be solved in $N^{1.99}$

Fine-grained complexity

SAT to OV

$$\overbrace{x_1 \quad \cdots \quad x_{n/2}}\ \Big|\ \overbrace{x_{n/2+1} \cdots x_n}$$

- Given a k-CNF ϕ, split its n input variables into two sets of size $n/2$

SAT to OV

- Given a k-CNF ϕ, split its n input variables into two sets of size $n/2$

$$\overbrace{\boxed{2^{n/2} \quad x_1 \ldots x_{n/2}}}^{S} \quad \overbrace{\boxed{2^{n/2} \quad x_{\frac{n}{2}+1} \ldots x_n}}^{T}$$

- For each assignment to the first group — a vector in S, for each assignment to the second — a vector in T

$2^{n/2}$ vectors in S

$2^{n/2}$ vectors in T

$$N = |S| = |T| = 2^{n/2}$$

SAT TO OV

- Given a k-CNF ϕ, split its n input variables into two sets of size $n/2$
- For each assignment to the first group — a vector in S, for each assignment to the second — a vector in T
- $N = 2^{n/2}$

SAT to OV

m – # of clauses

- For an assignment $x \in \{0,1\}^{n/2}$, add $s \in \{0,1\}^m$ to S:

$$s_i = 1 \text{ iff } x \text{ doesn't satisfy clause } C_i$$

$$\varphi = (x_1 \vee x_{n-1} \vee \bar{x}_3) \wedge (x_1 \vee x_n \vee \bar{x}_2) \wedge (x_3 \vee x_{n/2} \quad)$$

$x_{n/2+1}$ -- x_n

$x_1 \dots x_{n/2}$	
x_1	0
x_2	0
x_3	1
$\vdots$	
$x_{n/2}$	1

$$\xrightarrow{\varphi} (0 \vee x_{n-1} \vee 0) \wedge (0 \vee x_n \vee 1) \wedge (1 \vee 1 \quad)$$

$$= \boxed{0} \wedge \boxed{(1)} \wedge (1)$$

s_i: 1 0 $\boxed{0}$

SAT to OV

- For an assignment $x \in \{0,1\}^{n/2}$, add $s \in \{0,1\}^m$ to S:

$$s_i = 1 \text{ iff } x \text{ doesn't satisfy clause } C_i$$

- ϕ is SAT iff $\exists s \in S, t \in T$:

$$\forall i \in [m]: \; s_i \cdot t_i = 0$$

$s \in \{0,1\}^{n/2}$ $t \in \{0,1\}^{n/2}$

s.t. $\forall i \in [m]$ C_i is either satisfied by s OR by t

$s_i = 0$ OR $t_i = 0$

φ is SAT iF $\exists s \in S, t \in T$ $\langle s,t \rangle = \sum_{i=1}^{m} s_i t_i = 0$

SAT to OV

- For an assignment $x \in \{0,1\}^{n/2}$, add $s \in \{0,1\}^m$ to S:

$$s_i = 1 \text{ iff } x \text{ doesn't satisfy clause } C_i$$

- ϕ is SAT iff $\exists s \in S, t \in T$:

$$\forall i \in [m]: \quad x_i \cdot y_i = 0$$

SAT n vars $\Rightarrow$ OV $N = 2^{n/2}$

- An $N^{2-\varepsilon}$ algorithm for OV with $d = \omega(\log N)$ gives an algorithm for k-SAT with run time

$$N^{2-\varepsilon}$$

SAT to OV

- For an assignment $x \in \{0,1\}^{n/2}$, add $s \in \{0,1\}^m$ to S:

$$s_i = 1 \text{ iff } x \text{ doesn't satisfy clause } C_i$$

- ϕ is SAT iff $\exists s \in S, t \in T$:

$$\forall i \in [m]: \quad x_i \cdot y_i = 0$$

- An $N^{2-\varepsilon}$ algorithm for OV with $d = \omega(\log N)$ gives an algorithm for k-SAT with run time

$$N^{2-\varepsilon} = (2^{n/2})^{2-\varepsilon} = 2^{n-\varepsilon n/2} = 2^{0.99n}$$

break SETH

#OV

- Even harder problem #OV: Count the number of orthogonal pairs of vectors

$0 \leq t \leq N^2$

$t = \#$ of pairs (s, t)

$s \in S$, $t \in T$, $\langle s, t \rangle = 0$.

#OV

- Even harder problem #OV: Count the number of orthogonal pairs of vectors
- Still trivially solvable in $O(N^2 d)$

Brute force all pairs of vectors
in N^2, $\langle s,t \rangle$ in time $O(d)$

#OV

- Even harder problem #OV: Count the number of orthogonal pairs of vectors
- Still trivially solvable in $O(n^2 d)$
- We'll give a slightly faster algorithm: $n^{2-1/\log(d/\log n)}$

E.g., if $d = c \cdot \log n$, then

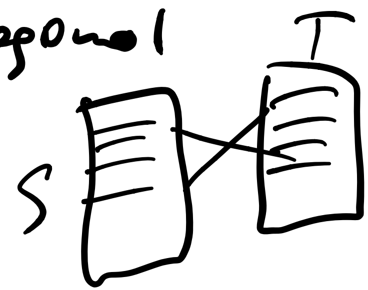
$n^{2-\frac{1}{\log c}}$ sub-quadratic

#OV

- Even harder problem #OV: Count the number of orthogonal pairs of vectors
- Still trivially solvable in $O(n^2 d)$
- We'll give a slightly faster algorithm: $n^{2-1/\log(d/\log n)}$
- We'll work with $\mathbb{F}_2$

s & t are orthogonal

$\langle s,t\rangle_2 = \sum s_i \cdot t_i = 0 \mod 2$



Algorithm for #OV

Theorem

There is a deterministic algorithm that solves #OV over $\mathbb{F}_2$ in time $O(n^{2-1/\log(d/\log n)})$ for any $d = o(n)$.

Modulus-Amplifying Polynomials [BT94]

For every ℓ, the following univariate polynomial of degree $< 2\ell$

deg $\leq 2\ell - 1 < 2\ell$

$$\boxed{F_\ell(x)} = 1 - (1-x)^\ell \cdot \sum_{i=0}^{\ell-1} \binom{\ell+i-1}{i} x^i.$$

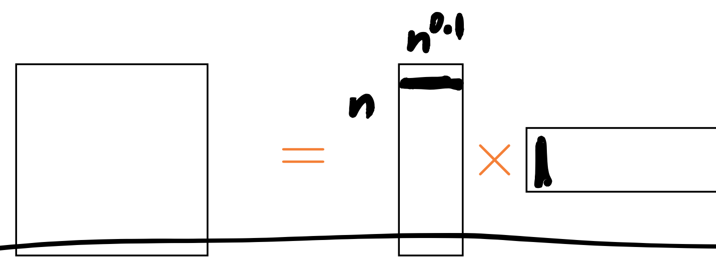
has the property that for every $x \in \mathbb{Z}$,

$$x = 0 \mod 2 \implies F_\ell(x) = 0 \mod 2^\ell.$$
$$x = 1 \mod 2 \implies F_\ell(x) = 1 \mod 2^\ell$$

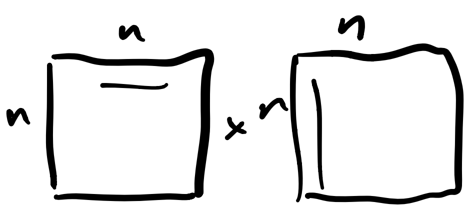
Rectangular Matrix Multiplication

Theorem (Cop82, Wil14)

There is a deterministic algorithm that multiplies two matrices $A \in \mathbb{F}_2^{n \times n^{0.172}}$ *and* $B \in \mathbb{F}_2^{n^{0.172} \times n}$ *in time* $O(n^2 \operatorname{poly}(\log n))$.



in time $n^2 \log^2 n$



$n^3 \to n^{\log_2 7} \to \underbrace{n^{2.373\ldots}}_{\omega}$

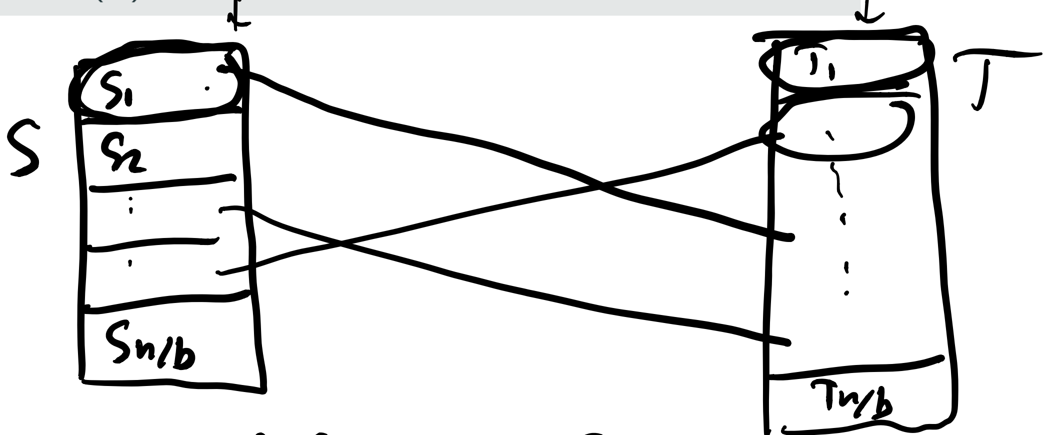
Algorithm for #OV

Theorem

There is a deterministic algorithm that solves $\#OV$ *over* $\mathbb{F}_2$ *in time* $O(n^{2-1/\log(d/\log n)})$ *for any* $d = o(n)$.

Theorem

There is a deterministic algorithm that solves #OV over $\mathbb{F}_2$ in time $O(n^{2-1/\log(d/\log n)})$ for any $d = o(n)$.



$S = S_1 \sqcup S_2 \cdots \sqcup S_{n/b}$

$T = T_1 \sqcup T_2 \cdots \sqcup T_{n/b}$

$|S_i| = |T_j| = b$

It suffices to solve

$\#OV(S_i, T_j) \quad \forall i,j \in [n/b]$

$$\#OV(S,T) = \sum_{i,j \in [n/b]} \#OV(S_i, T_j)$$

$$S = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} \qquad T = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{bmatrix}$$

$s_i, t_j \in \mathbb{F}_2^d$

$$\#OV(S,T) = \#(s,t) \text{ s.t. } s \in S, t \in T$$
$$\langle s, t \rangle_2 = 0.$$

$$S_1 = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_b \end{bmatrix} \qquad T_1 = \begin{bmatrix} t_1 \\ \vdots \\ t_b \end{bmatrix}$$

$$S_2 = \begin{bmatrix} s_{b+1} \\ \vdots \\ s_{2b} \end{bmatrix}$$

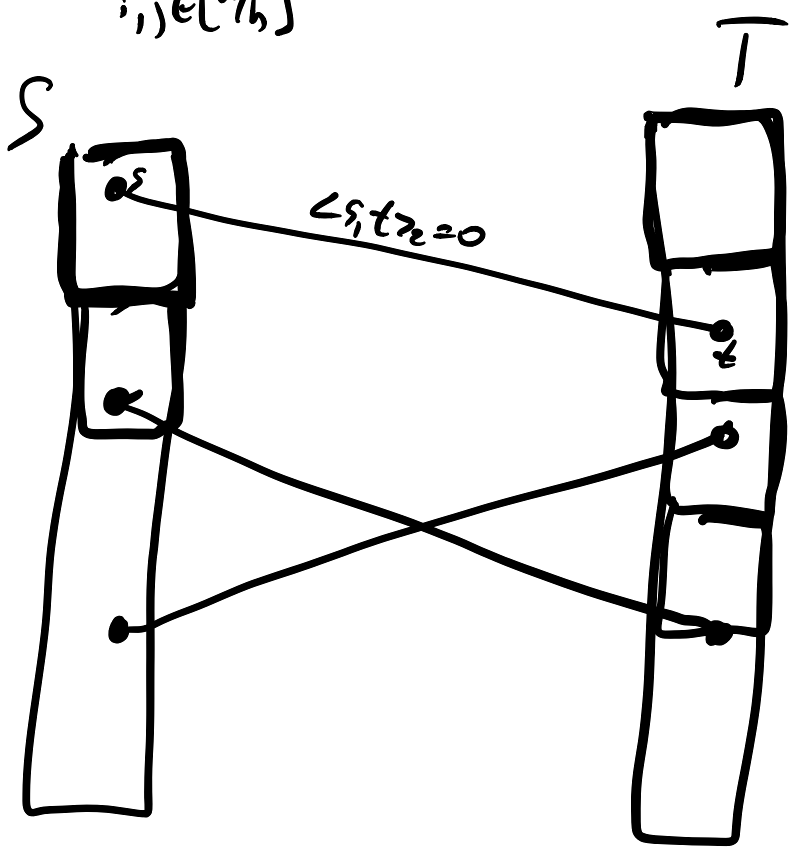
$\vdots$

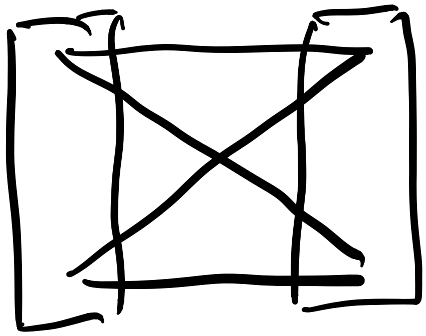
$$S_{n/b} = \begin{bmatrix} s_{n-b+1} \\ \vdots \\ s_n \end{bmatrix} \qquad T_{n/b} = \begin{bmatrix} t_{n-b+1} \\ \vdots \\ t_n \end{bmatrix}$$

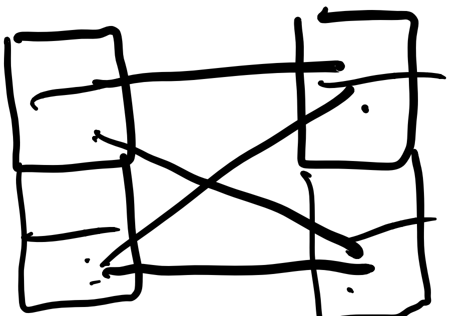
$$\boxed{\forall i,j \in [n/b] \quad \#OV(S_i, T_j)}$$

$$\#OV(S,T) = \left[\begin{matrix}\langle s,t\rangle_2 = 0 \\ s\in S,\ t\in T\end{matrix}\right] \iff \left[\begin{matrix}\langle s,t\rangle_2 = 0 \\ s\in S_i,\ t\in T_j\end{matrix}\right]$$

$$= \sum_{i,j\in[n/b]} \#OV(S_i, T_j)$$







I need to solve

$\#OV(\underline{S_i}, \underline{T_j})$ for

$\left(\frac{n}{b}\right)^2$ pairs of sets of size b.

We'll choose $b = n^{\Theta\left(\frac{1}{\log(d/\log n)}\right)}$

II. Each $\#OV(\underline{S_i}, \underline{T_j})$ can be solved in (amortized) time polylog(n)

Conclude: All $\#OV(S_i, T_j)$ in time $\left(\frac{n}{b}\right)^2 \cdot \mathrm{polylog}(n) = n^{2-\Theta\left(\frac{1}{\log(d/\log n)}\right)}$

$\Rightarrow$ Alg for $\#OV(S,T)$ with running time $n^{2-\Theta\left(\frac{1}{\log(d/\log n)}\right)}$

$X = S_i, \quad Y = T_j$

$\#OV(X,Y)$

$|X| = |Y| = b = 2^{\ell/4}$

def. ℓ.

$$\boxed{P(X,Y)} = \boxed{\sum_{\substack{x \in X \\ y \in Y}} 1 - F_\ell(\langle x,y \rangle)} \quad \text{over } \mathbb{Z}.$$

F_ℓ - Mod Ampl.

$\deg(F_\ell) < 2\ell$ s.t.

$$F_\ell(z) \bmod 2^\ell = z \bmod 2.$$

$$\boxed{1 - F_\ell(\langle x,y \rangle)} \underset{\bmod 2^\ell}{=} \begin{cases} 1, & \text{if } \langle x,y \rangle = 0 \bmod 2 \\ 0, & \text{if } \langle x,y \rangle = 1 \bmod 2 \end{cases}$$

$\langle x, y\rangle = 0 \bmod 2 \iff$
x & y are orthogonal over $\mathbb{F}_2$

$$\boxed{P(X, Y)} = \sum_{\substack{x \in X \\ y \in Y}} \left(1 - F_\ell(\langle x, y\rangle)\right)$$

$=$ ~~# orthogonal pairs from (X, Y)~~ mod 2^ℓ

$$\boxed{1 - F_\ell(\langle x, y\rangle)} = \quad \deg(F_\ell) < 2\ell$$

$$= 1 - F_\ell\left(\boxed{x_1 y_1} + \boxed{x_2 y_2} + \dots + x_d y_d\right)$$

$$= \sum_{i=1}^{M} c_i \cdot (x_1 y_1) \cdot (x_3 y_3) \cdot (x_d y_d) \dots$$

$$= \boxed{\sum_{i=1}^{M} c_i \cdot \prod_{j \in S_i} x_j \cdot \prod_{j \in S_i} y_j} \quad \dots \quad \checkmark$$

$S_i \subseteq [d] \qquad |S_i| < 2\ell.$

$$\boxed{M \le \binom{d}{\le 2\ell}}$$

Last step.

$\Phi_1(X)$ – vector of length M.

$$\Phi_1(X) = \Big(\sum_{x \in X} c_1 \cdot \prod_{j \in S_1} x_j,$$

$$\sum_{x \in X} c_2 \cdot \prod_{j \in S_2} x_j,$$

$$\ldots$$

$$\sum_{x \in X} c_M \cdot \prod_{j \in S_M} x_j \Big)$$

✓

$\Phi_2(Y)$ – vector of length M

✓

$$\Phi_2(Y) = \sum_{y \in Y} \prod_{j \in S_1} y_j,$$

$$\sum_{y \in Y} \prod_{j \in S_2} y_j,$$

$$\ldots$$

$$\langle \Phi_1(X), \Phi_2(Y) \rangle = \sum_{i=1}^{M} \sum_{\substack{x \in X \\ y \in Y}} c_i \prod_{j \in S_i} x_j \prod y_j$$

$$= \sum_{\substack{x \in X \\ y \in Y}} 1 - F_e(x,y) =$$

$$= P(X,Y) =$$

$$\boxed{= \#\ \text{orthogonal pairs } (X,Y).}$$

$$
\begin{array}{cc}
 & M \\
\frac{n}{b} & \begin{bmatrix} \varphi_1(S_1) \\ \varphi_1(S_2) \\ \vdots \\ \varphi_1(S_{n/b}) \end{bmatrix}
\end{array}
\quad
\begin{array}{cc}
 & \frac{n}{b} \\
M & \begin{bmatrix} \varphi_2(T_1) & \varphi_2(T_2) & \cdots & \varphi_2(T_{n/b}) \end{bmatrix}
\end{array}
=
$$

We want $\langle \varphi_1(S_i), \varphi_2(T_j) \rangle$

$\Rightarrow$ solve all $\#OV(S_i, T_j)$

$\Rightarrow$ solve $\#OV(S, T)$

$\varphi_1(S_1)\cdot\varphi_2(T_1) \quad \varphi_1(S_1)\varphi_2(T_2)$

$$\boxed{\varphi_1(S_i)\cdot\varphi_2(T_j)}$$

computes $\varphi_1(S_i)\cdot\varphi_2(T_j)$

$\forall i,j.$

To multiply 2 matrices

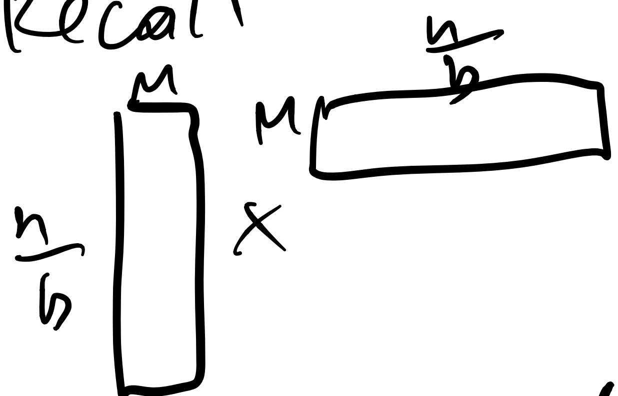
$\frac{n}{b}\times M \; ; \; M\times\frac{n}{b}$

We choose b so

$$\frac{n}{b} < n$$

$$M = n^{0.1}$$

Recall



can be multiplied $\left(\frac{n}{b}\right)^2 \cdot \text{polylog}\, n$

$$\approx \frac{n^2}{b^2} < n^2$$ - what we wanted.