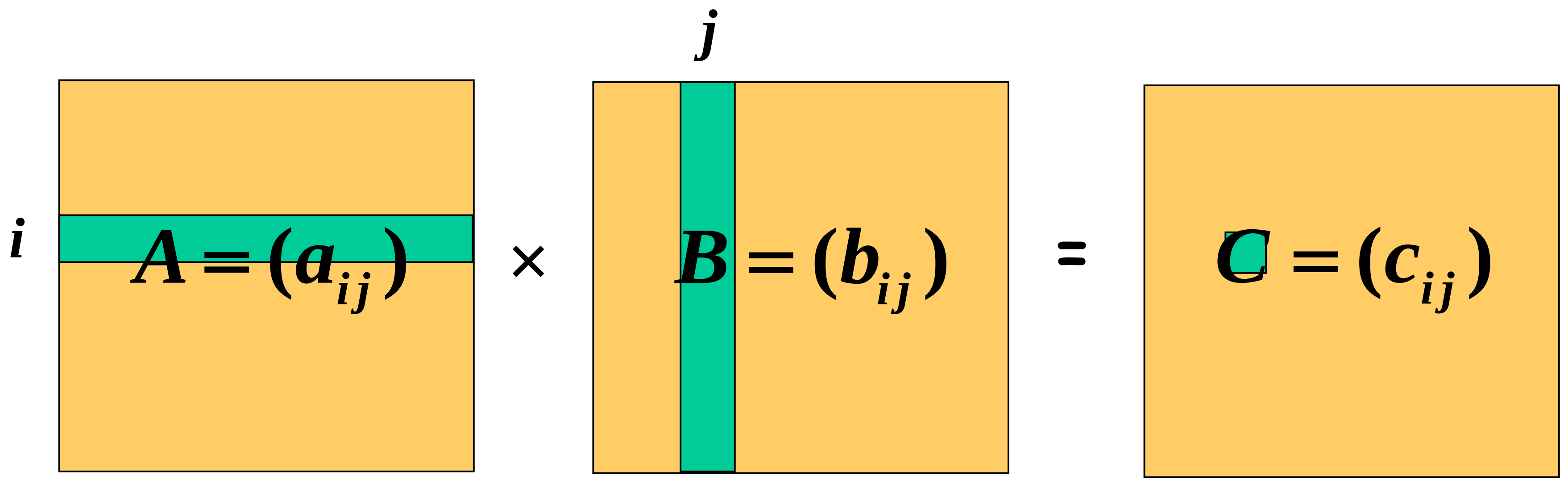
Matrix Multiplication and Graph Algorithms

Uri Zwick
Tel Aviv University

February 2015
Last updated: June 10, 2015

Short introduction to Fast matrix multiplication

Algebraic Matrix Multiplication



$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Can be computed naively in O(n^3) time.

Matrix multiplication algorithms

Complexity	Authors
n^3	—
$n^{2.81}$	**Strassen (1969)**
$\vdots$	
$n^{2.38}$	**Coppersmith-Winograd (1990)**

Conjecture/Open problem: $n^{2+o(1)}$???

Matrix multiplication algorithms - Recent developments

Complexity	Authors
$n^{2.376}$	**Coppersmith-Winograd (1990)**
$n^{2.374}$	**Stothers (2010)**
$n^{2.3729}$	**Williams (2011)**
$n^{2.37287}$	**Le Gall (2014)**

Conjecture/Open problem: $n^{2+o(1)}$???

Multiplying 2×2 matrices

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

$$\begin{aligned} C_{11} &= A_{11}B_{11} + A_{12}B_{21} \\ C_{12} &= A_{11}B_{12} + A_{12}B_{22} \\ C_{21} &= A_{21}B_{11} + A_{22}B_{21} \\ C_{22} &= A_{21}B_{12} + A_{22}B_{22} \end{aligned}$$

8 multiplications
4 additions

Works over any ring!

Multiplying *n*×*n* matrices

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

$$\begin{aligned} C_{11} &= A_{11}B_{11} + A_{12}B_{21} \\ C_{12} &= A_{11}B_{12} + A_{12}B_{22} \\ C_{21} &= A_{21}B_{11} + A_{22}B_{21} \\ C_{22} &= A_{21}B_{12} + A_{22}B_{22} \end{aligned}$$

8 multiplications

4 additions

$T(n) = 8\,T(n/2) + O(n^2)$

$T(n) = O(n^{\lg 8}) = O(n^3)$ ($\lg n = \log_2 n$)

“Master method” for recurrences

$$T(n) = a\,T\left(\frac{n}{b}\right) + f(n) \quad , \quad a \geq 1\,,\ b > 1$$

$$f(n) = O(n^{\log_b a - \varepsilon}) \quad \Rightarrow \quad T(n) = \Theta(n^{\log_b a})$$

$$f(n) = O(n^{\log_b a}) \quad \Rightarrow \quad T(n) = \Theta(n^{\log_b a} \log n)$$

$$\begin{array}{l} f(n) = O(n^{\log_b a + \varepsilon}) \\ af\left(\frac{n}{b}\right) \leq cn \ ,\ c < 1 \end{array} \quad \Rightarrow \quad T(n) = \Theta(f(n))$$

[CLRS 3rd Ed., p. 94]

Strassen’s 2×2 algorithm

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$
$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$
$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$
$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

$$C_{11} = M_1 + M_4 - M_5 + M_7$$
$$C_{12} = M_3 + M_5$$
$$C_{21} = M_2 + M_4$$
$$C_{22} = M_1 - M_2 + M_3 + M_6$$

Subtraction!

$$M_1 = (A$$
$$M_2 = (A_{21} + \quad B_{11}$$
$$M_3 = A_{11}(B_{12} - B_{22})$$
$$M_4 = A_{22}(B_{21} - B_{11})$$
$$M_5 = (A_{11} + A_{12})B_{22}$$
$$M_6 = (A_{21} - A_{11})(B_{11} + B_{12})$$
$$M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$$

7 multiplications
18 additions/subtractions

Works over any ring!
(Does not assume that multiplication is commutative)

Strassen's $n \times n$ algorithm

View each $n \times n$ matrix as a 2×2 matrix whose elements are $n/2 \times n/2$ matrices

Apply the 2×2 algorithm recursively

$$T(n) = 7\, T(n/2) + O(n^2)$$

$$T(n) = O(n^{\lg 7}) = O(n^{2.81})$$

Exercise: If n is a power of 2, the algorithm uses $n^{\lg 7}$ multiplications and $6(n^{\lg 7} - n^2)$ additions/subtractions

Winograd's 2×2 algorithm

$$\begin{array}{llll}
S_1 = A_{21} + A_{22} & T_1 = B_{21} - B_{11} & M_1 = A_{11} B_{11} & M_5 = S_1 T_1 \\
S_2 = S_1 - A_{11} & T_2 = B_{22} - T_1 & M_2 = A_{12} B_{21} & M_6 = S_2 T_2 \\
S_3 = A_{11} - A_{21} & T_3 = B_{22} - B_{12} & M_3 = S_4 B_{22} & M_7 = S_3 T_3 \\
S_4 = A_{12} - S_2 & T_4 = T_2 - B_{21} & M_4 = A_{22} T_4 &
\end{array}$$

$$\begin{array}{lll}
U_1 = M_1 + M_2 & U_5 = U_4 + M_3 & C_{11} = U_1 \\
U_2 = M_1 + M_6 & U_6 = U_3 - M_4 & C_{12} = U_5 \\
U_3 = U_2 + M_7 & U_7 = U_3 + M_5 & C_{21} = U_6 \\
U_4 = U_2 + M_5 & & C_{22} = U_7
\end{array}$$

Works over any ring!

7 multiplications
15 additions/subtractions

Exponent of matrix multiplication

Let ω be the "smallest" constant such that two $n \times n$ matrices can be multiplies in $O(n^\omega)$ time

$$2 \le \omega < 2.37287$$

(Many believe that $\omega = 2 + o(1)$)

Inverses / Determinants

The title of Strassen's 1969 paper is:
"Gaussian elimination is not optimal"

Other matrix operations that can
be performed in $O(n^{\omega})$ time:

- Computing inverses: $\boldsymbol{A}^{-1}$
- Computing determinants: $\det(\boldsymbol{A})$
- Solving systems of linear equations: $\boldsymbol{Ax} = \boldsymbol{b}$
- Computing LUP decomposition: $\boldsymbol{A} = \boldsymbol{LUP}$
- Computing characteristic polynomials: $\det(\boldsymbol{A} - \lambda \boldsymbol{I})$
- Computing $\mathit{rank}(A)$ and a corresponding submatrix

Block-wise Inversion

$$M^{-1} = \begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} + A^{-1}BS^{-1}CA^{-1} & -A^{-1}BS^{-1} \\ -S^{-1}CA^{-1} & S^{-1} \end{pmatrix}$$

$$\det(M) = \det(A) \cdot \det(S)$$

$$S = D - CA^{-1}B \qquad (\text{“Schur complement”})$$

Provided that *A* and *S* are invertible

$$I(n) = 2I(\frac{n}{2}) + O(n^{\omega}) \implies I(n) = O(n^{\omega})$$

If *M* is (square, real, symmetric) positive definite, ($M=N^TN$, *N* invertible), then *M* satisfies the conditions above

If *M* is a real invertible square matrix, $M^{-1}=(M^TM)^{-1}M^T$

Over other fields, use LUP factorization

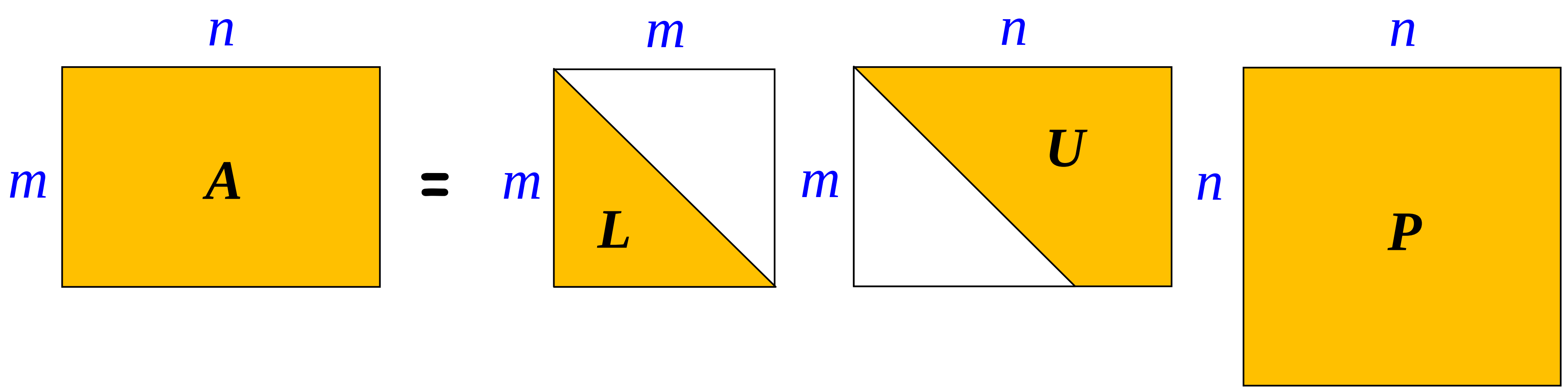
Positive Definite Matrices

A real symmetric $n \times n$ matrix A is said to be positive-definite (PD) iff $x^T A x > 0$ for every $x \neq 0$

Theorem: (Cholesky decomposition)
A is PD iff $A = B^T B$ where B invertible

Exercise: If M is PD then the matrices A and S encountered in the inversion algorithm are also PD

LUP decomposition



L is *unit lower triangular*

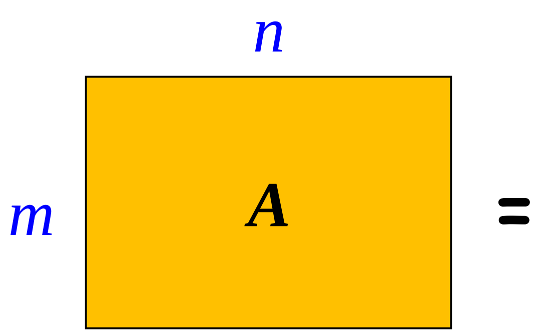
U is *upper triangular*

P is a *permutation* matrix

Can be computed in $O(n^{\omega})$ time

LUP decomposition (in pictures)

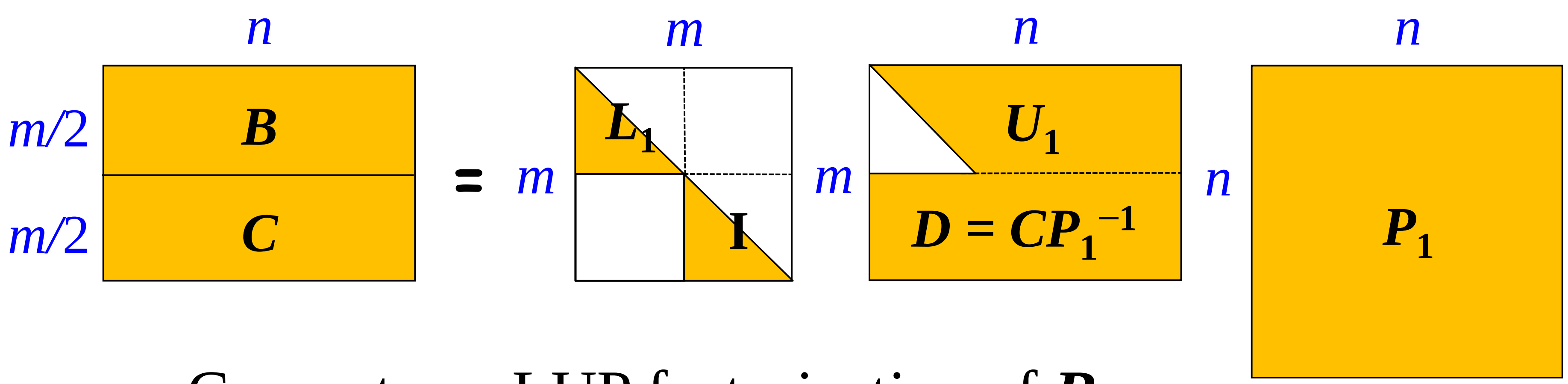
[Bunch-Hopcroft (1974)]



[AHU’74, Section 6.4 p. 234]

LUP decomposition (in pictures)

[Bunch-Hopcroft (1974)]

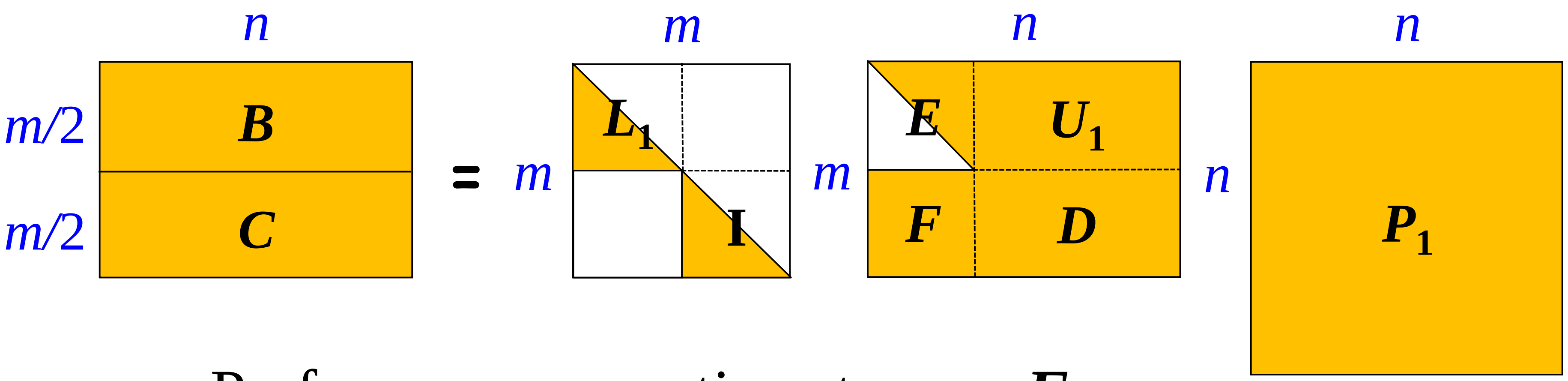


Compute an LUP factorization of $\boldsymbol{B}$

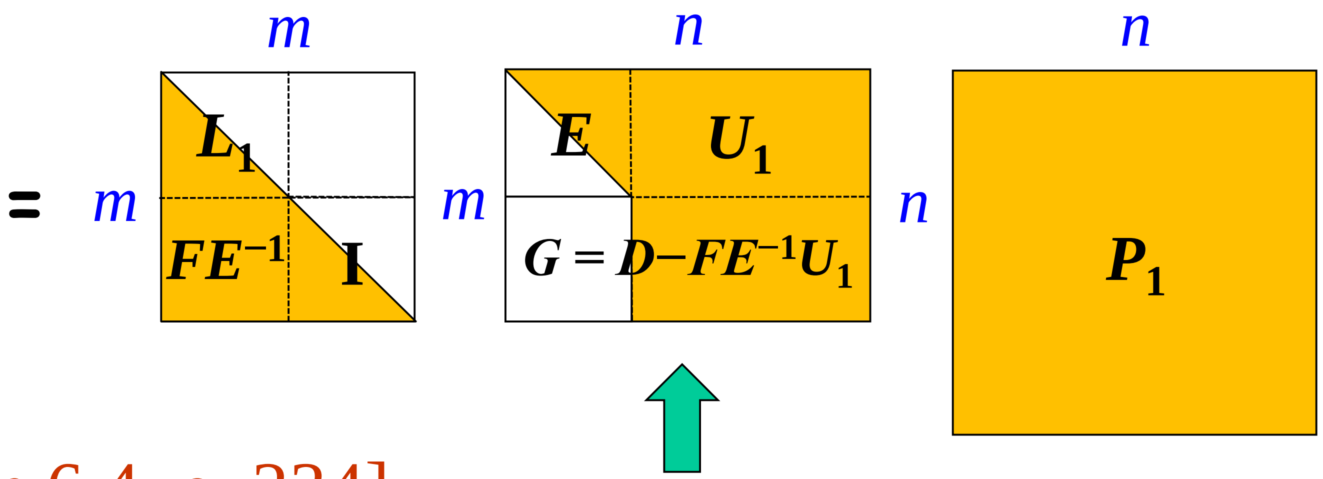
[AHU’74, Section 6.4 p. 234]

LUP decomposition (in pictures)

[Bunch-Hopcroft (1974)]



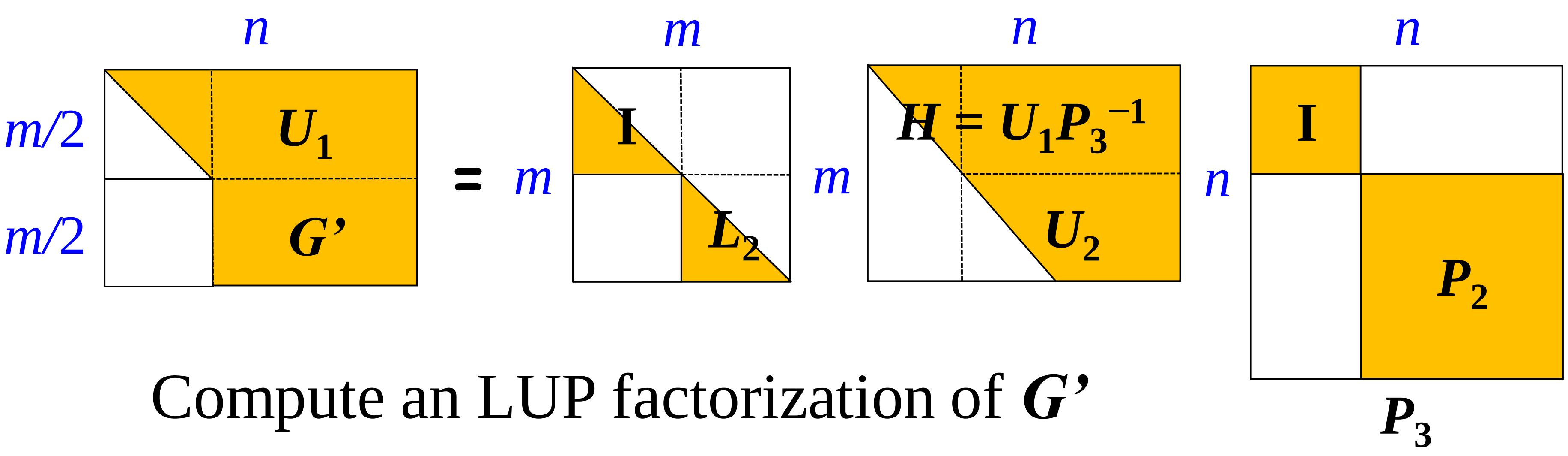
Perform row operations to zero $\boldsymbol{F}$



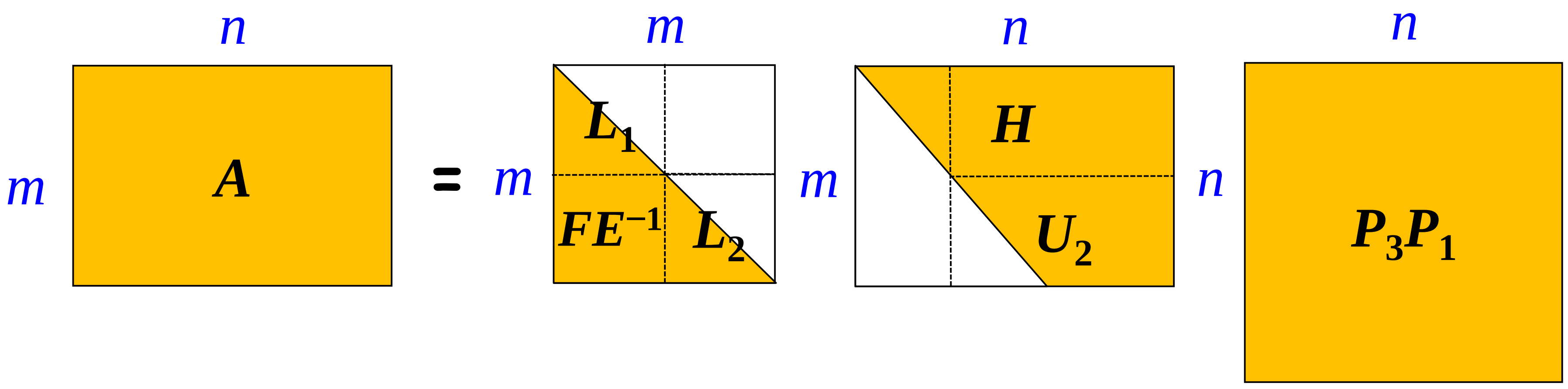
[AHU’74, Section 6.4 p. 234]

LUP decomposition (in pictures)

[Bunch-Hopcroft (1974)]



Compute an LUP factorization of G'



[AHU'74, Section 6.4 p. 234]

LUP decomposition (in pictures)
[Bunch-Hopcroft (1974)]

Where did we use the permutations?

In the base case *m*=1 !

Example: $[\,0\ 5\,] = [\,1\,][\,5\ 0\,]\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$

LUP decomposition - Complexity

[Bunch-Hopcroft (1974)]

$$L(m,n) \; = \; L\left(\frac{m}{2},n\right) + L\left(\frac{m}{2},n-\frac{m}{2}\right) + O\left(M\left(\frac{m}{2},\frac{m}{2},n\right)\right)$$

$$L(m,n) \; \leq \; 2\,L\left(\frac{m}{2},n\right) + O\left(\frac{n}{m}m^{\omega}\right)$$

$$L(m,n) = L(m)\,n$$

$$L(m) \; \leq \; 2\,L\left(\frac{m}{2}\right) + O(m^{\omega-1})$$

$$L(m) \; = \; \Theta(m^{\omega-1})$$

$$L(m,n) \; = \; O(m^{\omega-1}n)$$

$$L(n,n) \; = \; O(n^{\omega})$$

Inversion → Matrix Multiplication

$$\begin{pmatrix} I & A & 0 \\ 0 & I & B \\ 0 & 0 & I \end{pmatrix}^{-1} = \begin{pmatrix} I & -A & AB \\ 0 & I & -B \\ 0 & 0 & I \end{pmatrix}$$

Exercise: Show that matrix multiplication and matrix squaring are essentially equivalent.

Checking Matrix Multiplication

$$C = AB \quad ?$$

Matrix Multiplication
Determinants / Inverses

Combinatorial applications?

Transitive closure

Shortest Paths

Perfect/Maximum matchings

Dynamic transitive closure and shortest paths

k-vertex connectivity

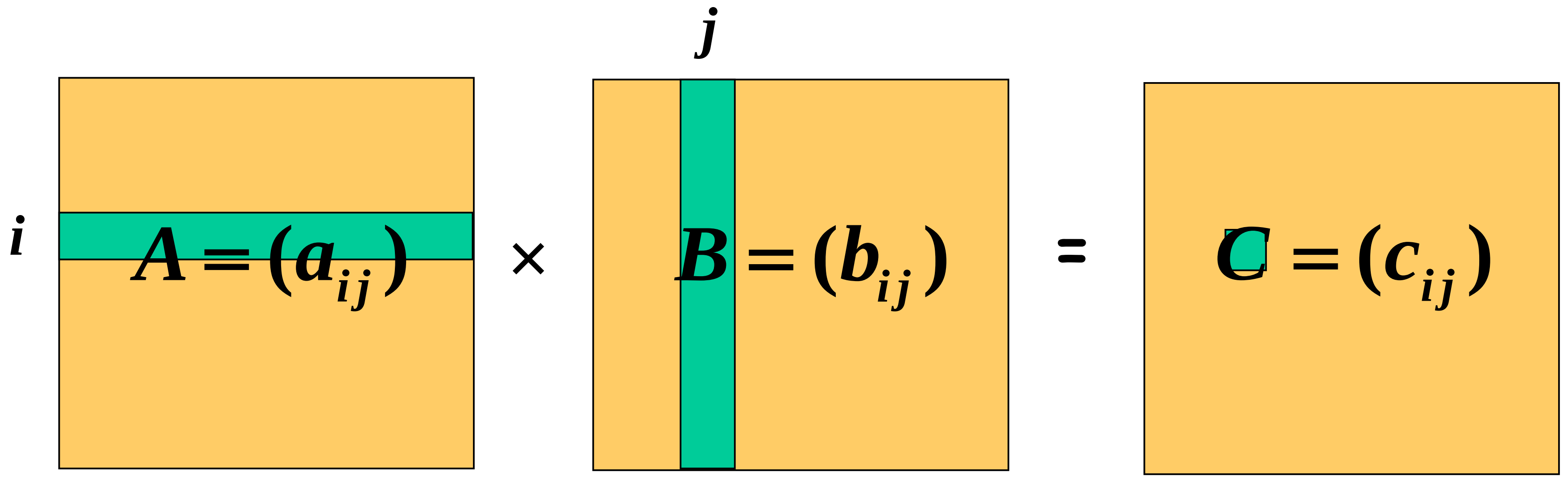
Counting spanning trees

BOOLEAN MATRIX MULTIPLICATION

AND

TRANSIVE CLOSURE

Boolean Matrix Multiplication



$$c_{ij} = \bigvee_{k=1}^{n} a_{ik} \wedge b_{kj}$$

Can be computed naively in O(n^3) time.

Algebraic Product

$$C = A\,B$$

$$c_{ij} = \sum_k a_{ik} b_{kj}$$

O(n^ω)

algebraic operations

Boolean Product

$$C = A \cdot B$$

$$c_{ij} = \bigvee_k a_{ik} \wedge b_{kj}$$

O(n^ω)?

Logical or ($\vee$) has no inverse!

But, we can work over the **integers!** (modulo n+1)

O(n^ω) operations on O(log n)-bit words

Witnesses for Boolean Matrix Multiplication

$$C = AB$$

$$c_{ij} = \bigvee_{k=1}^{n} a_{ik} \wedge b_{kj}$$

A matrix W is a matrix of **witnesses** iff

If $c_{ij} = 0$ then $w_{ij} = 0$

If $c_{ij} = 1$ then $w_{ij} = k$ where $a_{ik} = b_{kj} = 1$

Can we compute witnesses in $O(n^{\omega})$ time?

Transitive Closure

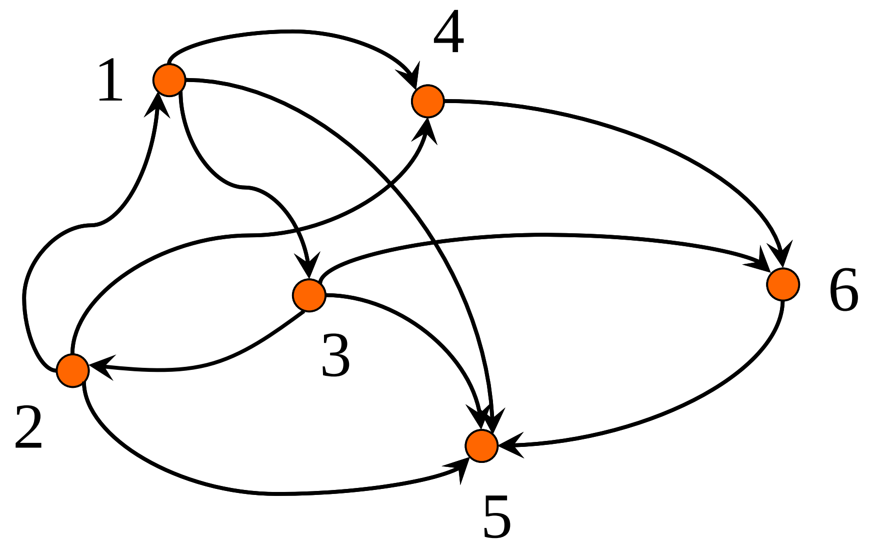
Let $G=(V,E)$ be a directed graph.

The transitive closure $G^*=(V,E^*)$ is the graph in which $(u,v)\in E^*$ iff there is a path from u to v.

Can be easily computed in $O(mn)$ time.

Can also be computed in $O(n^\omega)$ time.

Adjacency matrix of a directed graph



$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Exercise 0: If A is the adjacency matrix of a graph, then $(A^k)_{ij}=1$ iff there is a path of length k from i to j.

Transitive Closure using matrix multiplication

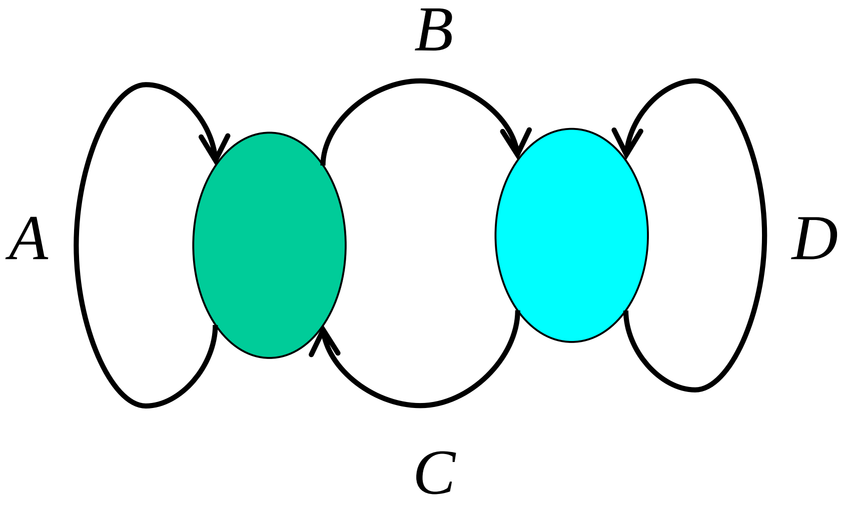
Let $G=(V,E)$ be a directed graph.

If A is the adjacency matrix of G, then $(A \vee I)^{n-1}$ is the adjacency matrix of G^*.

The matrix $(A \vee I)^{n-1}$ can be computed by $\log n$ squaring operations in $O(n^\omega \log n)$ time.

It can also be computed in $O(n^\omega)$ time.

$$X = \begin{array}{|c|c|}\hline A & B \\ \hline C & D \\ \hline\end{array}$$



$$X^* = \begin{array}{|c|c|}\hline E & F \\ \hline G & H \\ \hline\end{array} = \begin{array}{|c|c|}\hline & \\ \hline & \\ \hline\end{array}$$

$$\boldsymbol{TC(n) \le 2\ TC(n/2) + 6\ BMM(n/2) + O(n^2)}$$

Exercise 1: Give $O(n^\omega)$ algorithms for findning, in a directed graph,

a) a triangle
b) a simple quadrangle
c) a simple cycle of length k.

Hints:

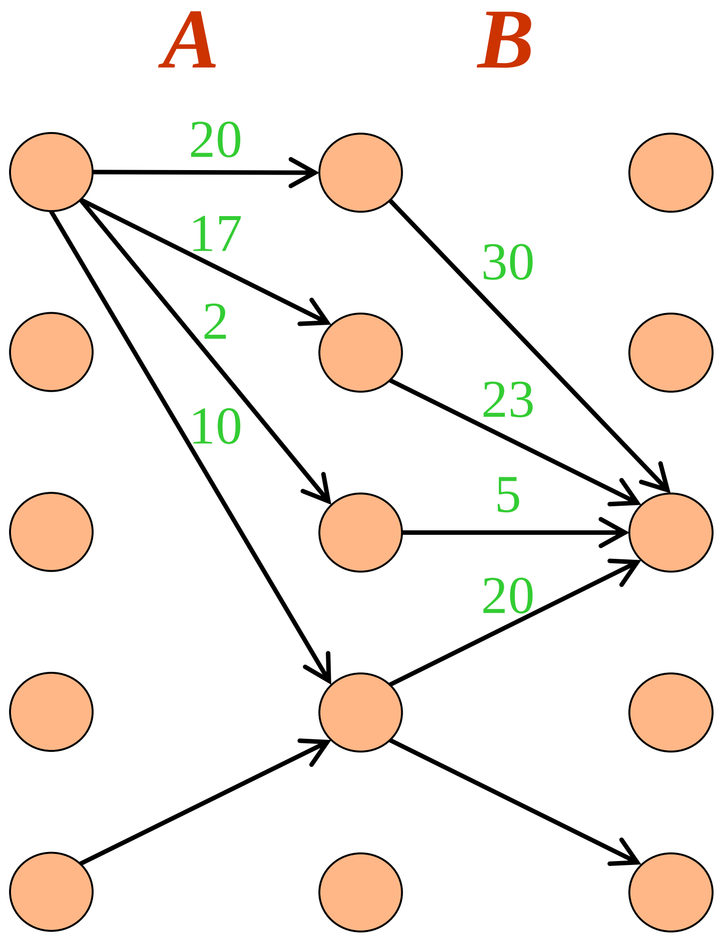
1. In an **acyclic** graph all paths are simple.
2. In c) running time may be **exponential** in k.
3. **Randomization** makes solution much easier.

MIN-PLUS MATRIX MULTIPLICATION

AND

ALL-PAIRS SHORTEST PATHS (APSP)

An interesting special case of the APSP problem



$$C = A * B$$

$$c_{ij} = \min_{k}\{a_{ik}+b_{kj}\}$$

Min-Plus product

Min-Plus Products

$$C = A * B$$

$$c_{ij} = \min_k \{a_{ik} + b_{kj}\}$$

$$\begin{pmatrix} -6 & -3 & -10 \\ 2 & 5 & -2 \\ -1 & -7 & -5 \end{pmatrix} = \begin{pmatrix} 1 & -3 & 7 \\ +\infty & 5 & +\infty \\ 8 & 2 & -5 \end{pmatrix} * \begin{pmatrix} 8 & +\infty & -4 \\ -3 & 0 & -7 \\ 5 & -2 & 1 \end{pmatrix}$$

Solving APSP by repeated squaring

If W is an n by n matrix containing the edge weights of a graph. Then W^n is the distance matrix.

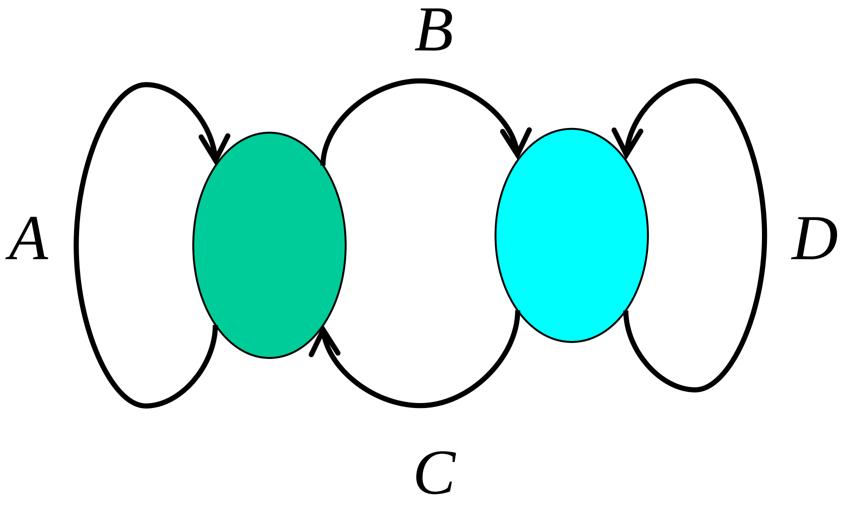
By induction, W^k gives the distances realized by paths that use at most k edges.

```
$D \leftarrow W$
for $i \leftarrow 1$ to $\lceil \log_2 n \rceil$
do $D \leftarrow D*D$
```

Thus: $APSP(n) \leq MPP(n) \log n$

Actually: $APSP(n) = O(MPP(n))$

$$X = \begin{array}{|c|c|} \hline A & B \\ \hline C & D \\ \hline \end{array}$$



$$X^* = \begin{array}{|c|c|} \hline E & F \\ \hline G & H \\ \hline \end{array} = \begin{array}{|c|c|} \hline (A \vee BD^*C)^* & EBD^* \\ \hline D^*CE & D^* \vee GBD^* \\ \hline \end{array}$$

$$\mathbf{APSP}(n) \le 2\ \mathbf{APSP}(n/2) + 6\ \mathbf{MPP}(n/2) + O(n^2)$$

Algebraic Product

$$C = A \cdot B$$

$$c_{ij} = \sum_k a_{ik} b_{kj}$$

$O(n^{\omega})$

Min-Plus Product

$$C = A * B$$

$$c_{ij} = \min_k \{a_{ik} + b_{kj}\}$$

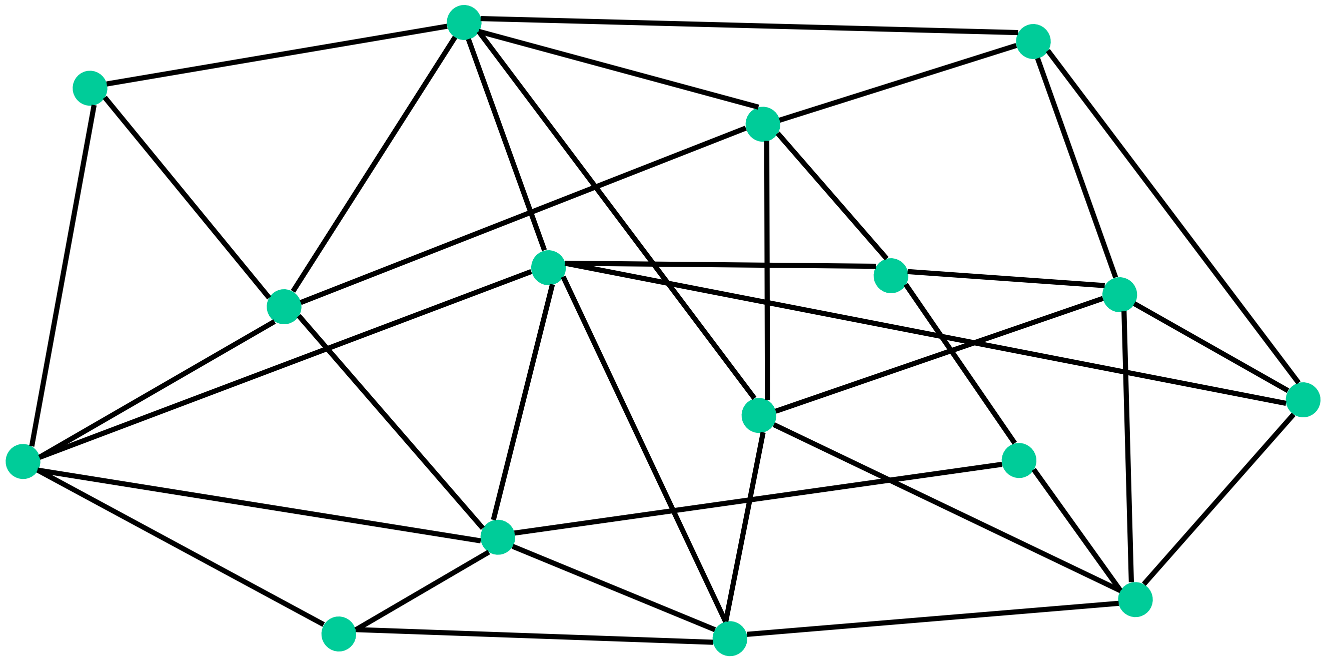
min operation has no inverse!

?

To be continued…

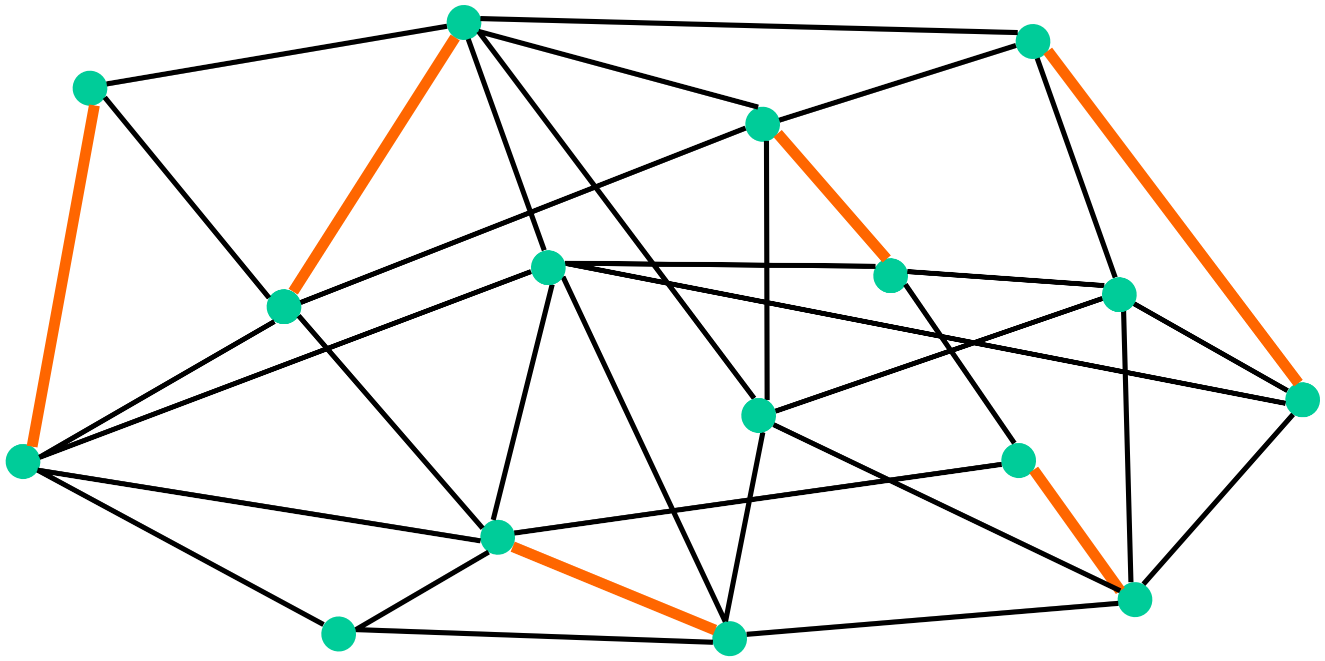
PERFECT MATCHINGS

Matchings



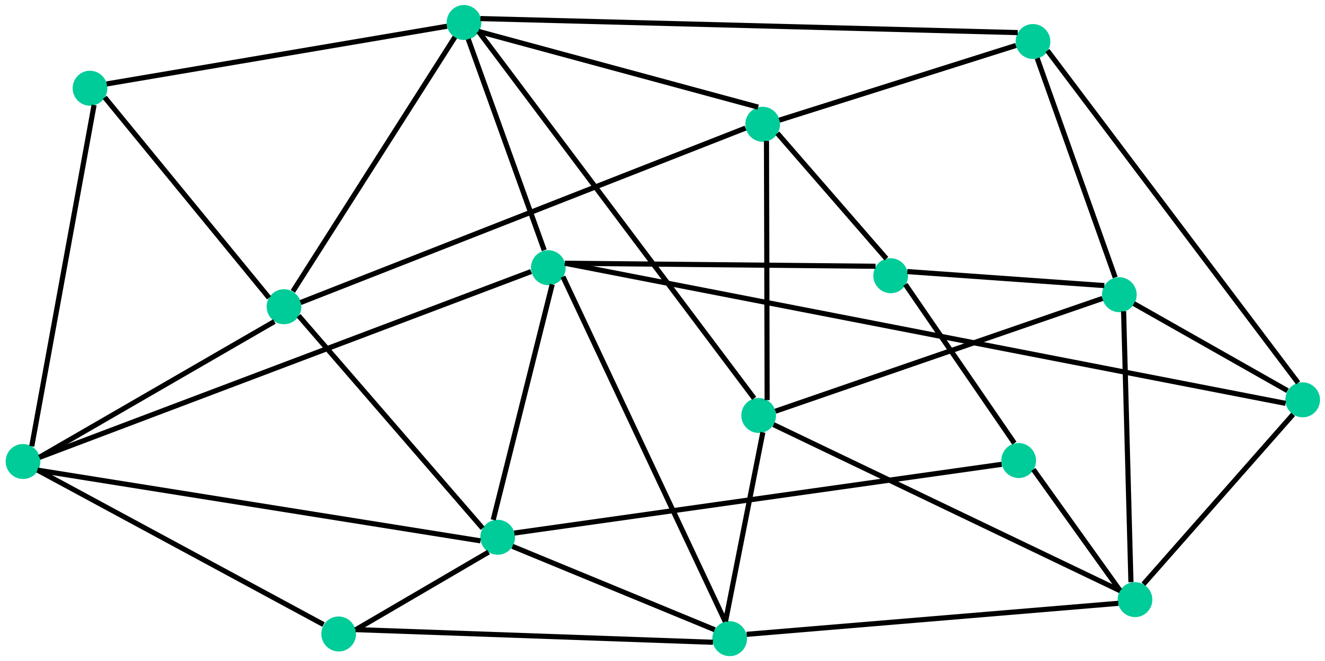
A matching is a subset of edges that do not touch one another.

Matchings



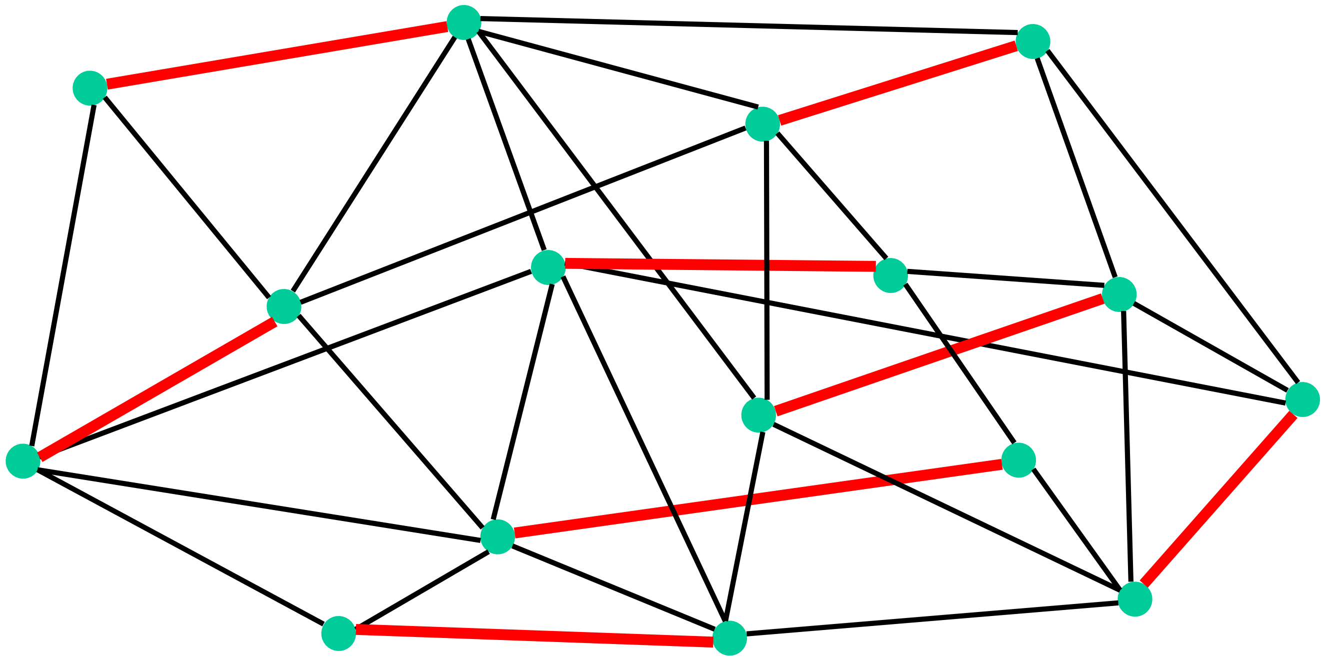
A matching is a subset of edges that do not touch one another.

Perfect Matchings



A matching is perfect if there are no unmatched vertices

Perfect Matchings

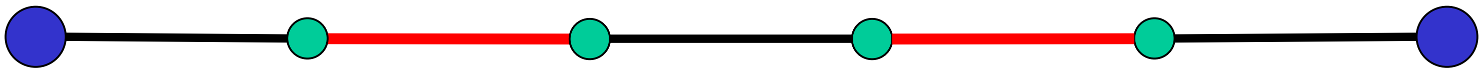


A matching is perfect if there are no unmatched vertices

Algorithms for finding perfect or maximum matchings

Combinatorial approach:

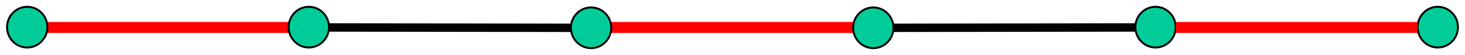
A matching *M* is a maximum matching iff it admits no augmenting paths



Algorithms for finding perfect or maximum matchings

Combinatorial approach:

A matching M is a maximum matching iff it admits no augmenting paths



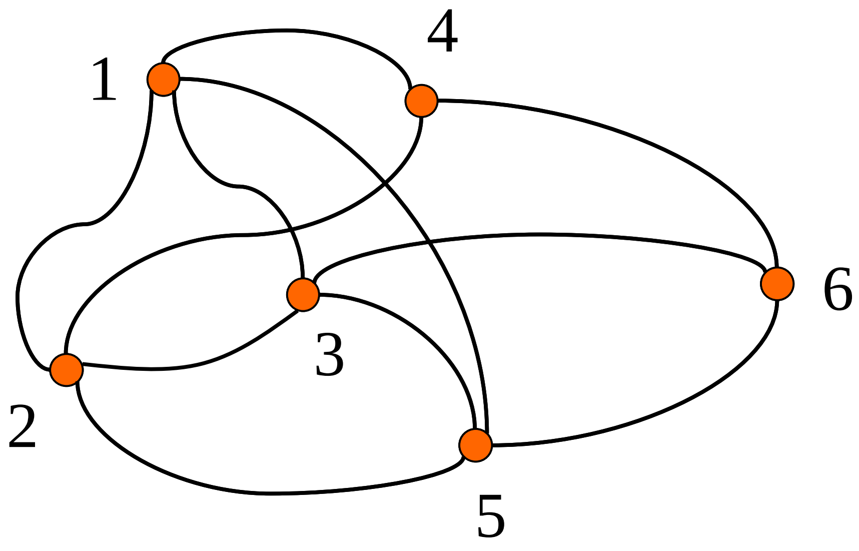
Combinatorial algorithms for finding perfect or maximum matchings

In bipartite graphs, augmenting paths, and hence maximum matchings, can be found quite easily using max flow techniques.

In non-bipartite the problem is much harder. (Edmonds' Blossom shrinking techniques)

Fastest running time (in both cases):
$O(mn^{1/2})$ [Hopcroft-Karp] [Micali-Vazirani]

Adjacency matrix
# of a undirected graph



$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

The adjacency matrix of an undirected graph is symmetric.

Matchings, Permanents, Determinants

$$\det(A) = \sum_{\pi \in S_n} sign(\pi) \prod_{i=1}^{n} a_{i\pi(i)}$$

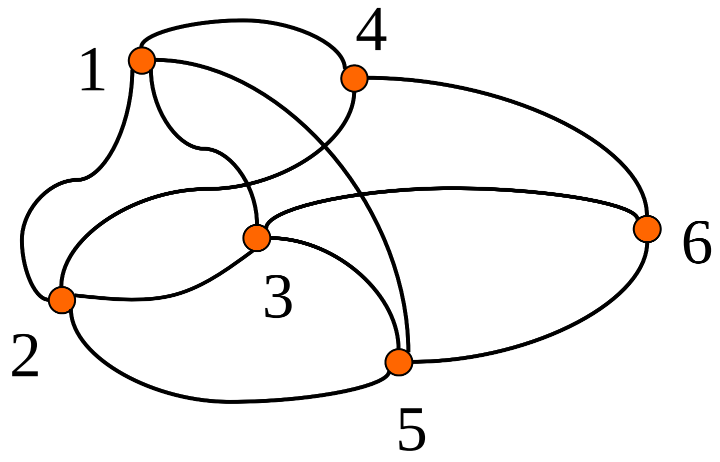
$$\text{per}(A) = \sum_{\pi \in S_n} \prod_{i=1}^{n} a_{i\pi(i)}$$

Exercise: Show that if *A* is the adjacency matrix of a bipartite graph *G*, then per(*A*) is the number of perfect matchings in *G*.

Unfortunately computing the permanent is **#P-complete**…

Tutte's matrix

(Skew-symmetric symbolic adjacency matrix)

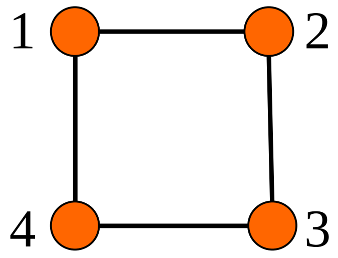


$$\begin{pmatrix} 0 & x_{12} & x_{13} & x_{14} & x_{15} & 0 \\ -x_{12} & 0 & x_{23} & x_{24} & x_{25} & 0 \\ -x_{13} & -x_{23} & 0 & 0 & x_{35} & x_{36} \\ -x_{14} & -x_{24} & 0 & 0 & 0 & x_{46} \\ -x_{15} & -x_{25} & -x_{35} & 0 & 0 & x_{56} \\ 0 & 0 & -x_{36} & -x_{46} & -x_{56} & 0 \end{pmatrix}$$

$$a_{ij} = \begin{cases} x_{ij} & \text{if } \{i,j\} \in E \text{ and } i < j, \\ -x_{ji} & \text{if } \{i,j\} \in E \text{ and } i > j, \\ 0 & \text{otherwise} \end{cases} \qquad A^T = -A$$

Tutte's theorem

Let ***G*=(*V*,*E*)** be a graph and let ***A*** be its Tutte matrix.
Then, ***G*** has a perfect matching iff det(***A***) $\not\equiv 0$.



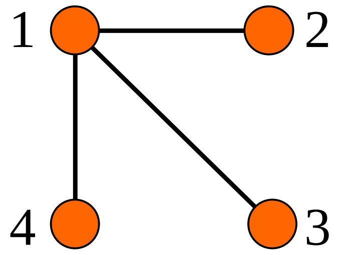
$$A = \begin{pmatrix} 0 & x_{12} & 0 & x_{14} \\ -x_{12} & 0 & x_{23} & 0 \\ 0 & -x_{23} & 0 & -x_{34} \\ -x_{14} & 0 & -x_{34} & 0 \end{pmatrix}$$

$$\begin{aligned} \det(A) &= x_{12}^2 x_{34}^2 + x_{14}^2 x_{23}^2 + 2x_{12}x_{23}x_{34}x_{41} \not\equiv 0 \\ &= (x_{12}x_{34} + x_{14}x_{23})^2 \end{aligned}$$

There are perfect matchings

Tutte's theorem

Let ***G*=(*V*,*E*)** be a graph and let ***A*** be its Tutte matrix.
Then, ***G*** has a perfect matching iff det(***A***) $\not\equiv 0$.



$$A = \begin{pmatrix} 0 & x_{12} & x_{13} & x_{14} \\ -x_{12} & 0 & 0 & 0 \\ -x_{13} & 0 & 0 & 0 \\ -x_{14} & 0 & 0 & 0 \end{pmatrix}$$

$$\det(A) \equiv 0$$

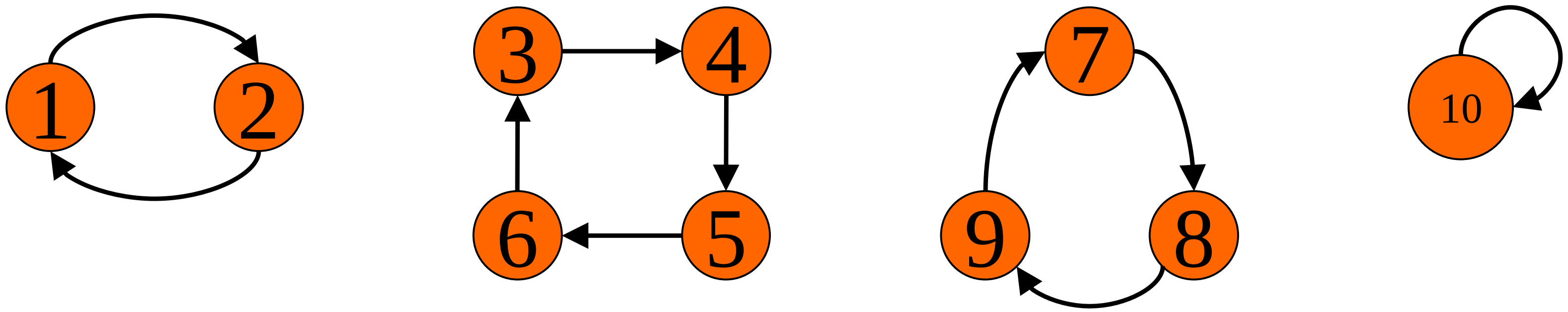
No perfect matchings

Proof of Tutte's theorem

$$\det(A) = \sum_{\pi \in S_n} sign(\pi) \prod_{i=1}^{n} a_{i,\pi(i)}$$

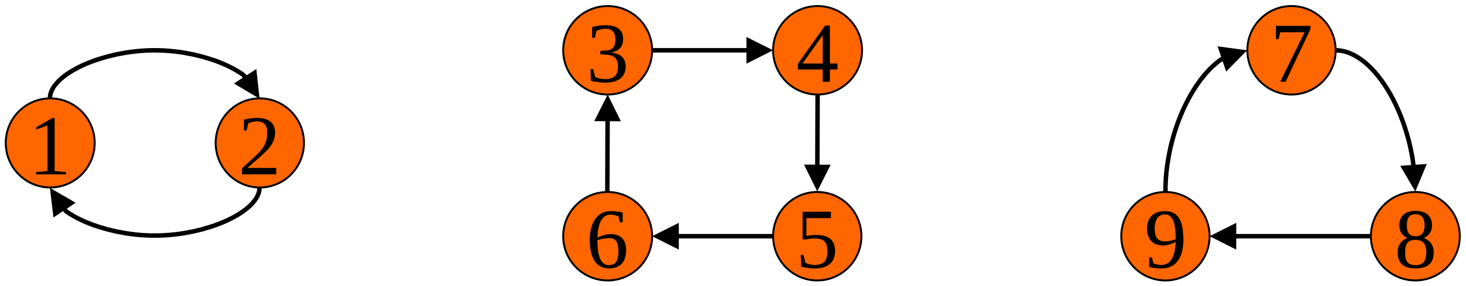
Every permutation $\pi \in S_n$ defines a cycle collection

$$\pi = (2\ 1\ 4\ 5\ 6\ 3\ 8\ 9\ 7\ 10)$$



Cycle covers

A permutation $\pi \in S_n$ for which $\{i,\pi(i)\} \in E$, for $1 \le i \le n$, defines a cycle cover of the graph.

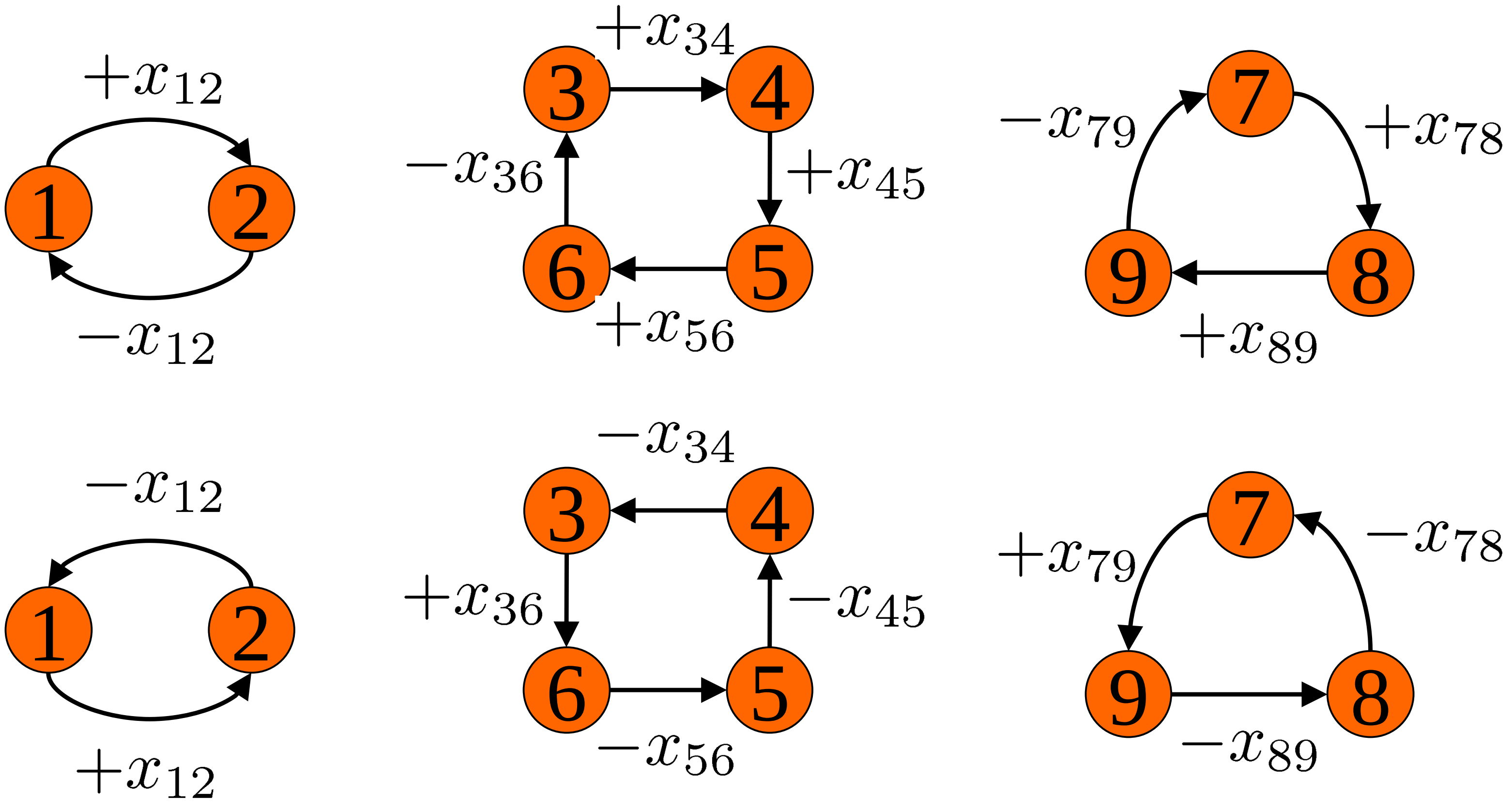


Exercise: If π' is obtained from π by reversing the direction of a cycle, then $sign(\pi')=sign(\pi)$.

$$\prod_{i=1}^{n} a_{i\pi'(i)} = \pm \prod_{i=1}^{n} a_{i\pi(i)}$$

Depending on the parity of the cycle!

Reversing Cycles



$$\prod_{i=1}^{n} a_{i\pi'(i)} = \pm \prod_{i=1}^{n} a_{i\pi(i)}$$

Depending on the parity of the cycle!

Proof of Tutte's theorem (cont.)

$$\det A = \sum_{\pi \in S_n} sign(\pi) \prod_{i=1}^{n} a_{i\pi(i)}$$

The permutations $\pi \in S_n$ that contain an **odd** cycle cancel each other!

We effectively sum only over **even cycle covers**.

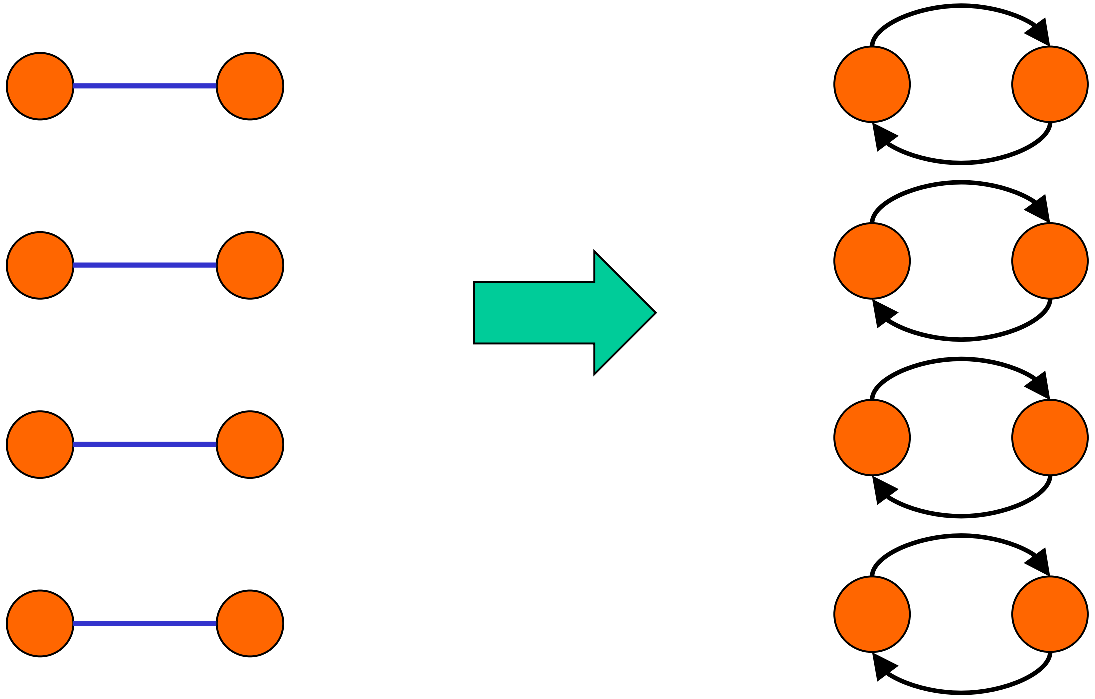
Different **even cycle covers** define different **monomials**, which do *not* cancel each other out.

A graph contains a perfect matching iff it contains an **even cycle cover**.

Proof of Tutte's theorem (cont.)

A graph contains a perfect matching iff it contains an **even cycle cover**.

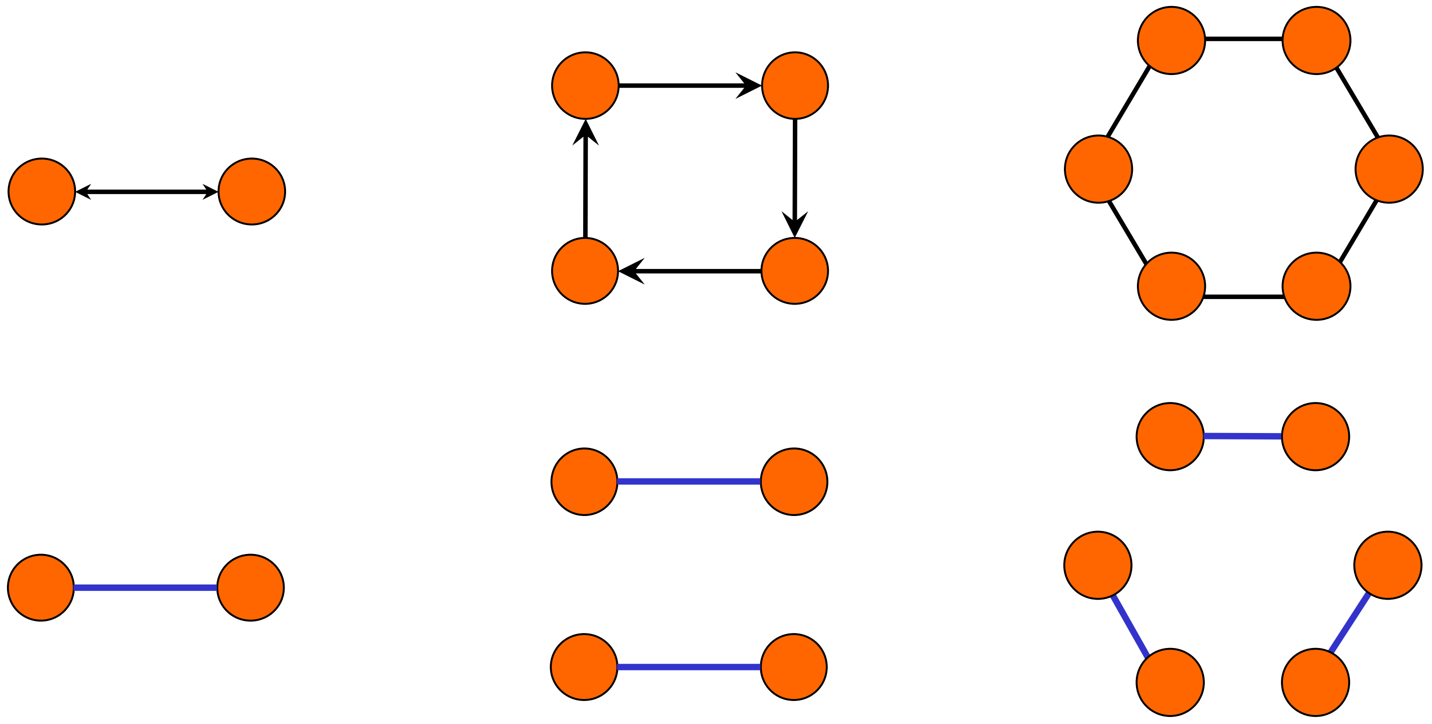
Perfect Matching → Even cycle cover



Proof of Tutte's theorem (cont.)

A graph contains a perfect matching iff it contains an **even cycle cover**.

Even cycle cover → Perfect matching



Pfaffians

$$\mathrm{pf}(A) \;=\; \sum_{M \in \mathcal{M}_n} sign(M) \prod_{(i,j)\in M} a_{i,j}$$

$$\mathcal{M}_n \;=\; \text{perfect matchings of } \{1, 2, \ldots, n\}$$

$$sign(\ \{(i_1, j_1), (i_2, j_2), \ldots, (i_{n/2}, j_{n/2})\ \}\ ) \;=\; sign\left(\begin{bmatrix} 1 & 2 & 3 & 4 & \cdots & n-1 & n \\ i_1 & j_1 & i_2 & j_2 & \cdots & i_{n/2} & j_{n/2} \end{bmatrix}\right)$$

(We may assume that $i_1 < j_1$, $i_2 < j_2$,$\ldots$)

Theorem [Muir (1882)]

If A is skew-symmetric, then

$$\det(A) = \mathrm{pf}(A)^2$$

An algorithm for perfect matchings?

- Construct the Tutte matrix $\boldsymbol{A}$.
- Compute $\det(\boldsymbol{A})$.
- If $\det(\boldsymbol{A}) \not\equiv 0$, say 'yes', otherwise 'no'.

Problem: $\det(\boldsymbol{A})$ is a symbolic expression that may be of exponential size!

Lovasz's solution: Replace each variable $\boldsymbol{x_{ij}}$ by a random element of $\boldsymbol{Z_p}$, where $\boldsymbol{p} = \Theta(\boldsymbol{n}^2)$ is a *prime* number

The Schwartz-Zippel lemma
[Schwartz (1980)] [Zippel (1979)]

Let $P(x_1,x_2,\ldots,x_n)$ be a polynomial of degree d over a field F. Let $S \subseteq F$. If $P(x_1,x_2,\ldots,x_n) \not\equiv 0$ and $a_1,a_2,\ldots,a_n$ are chosen independently and uniformly at random from S, then

$$\Pr[\,P(a_1,a_2,\ldots,a_n)=0\,] \;\le\; \frac{d}{|S|}$$

Proof by induction on n.

For $n=1$, follows from the fact that polynomial of degree d over a field has at most d roots

Proof of Schwartz-Zippel lemma

$$P(x_1, x_2, \ldots, x_n) = \sum_{i=0}^{d} P_i(x_2, \ldots, x_n)\, x_1^i$$

Let $k \leq d$ be the largest i such that $P_i(x_2, \ldots, x_n) \neq 0$

$$\Pr[\, P(a_1, a_2, \ldots, a_n) = 0\,]$$

$$\leq \Pr[\, P_k(a_2, \ldots, a_n) = 0\,] +$$

$$\Pr[\, P(a_1, a_2, \ldots, a_n) = 0 \mid P_k(a_2, \ldots, a_n) \neq 0\,]$$

$$\leq \frac{d-k}{|S|} + \frac{k}{|S|} = \frac{d}{|S|}$$

Lovasz's algorithm for existence of perfect matchings

- Construct the Tutte matrix $\boldsymbol{A}$.
- Replace each variable x_{ij} by a random element of Z_p, where $p \geq n^2$ is prime.
- Compute $\det(\boldsymbol{A})$.
- If $\det(\boldsymbol{A}) \neq 0$, say 'yes', otherwise 'no'.

If algorithm says 'yes', then
the graph contains a perfect matching.

If the graph contains a perfect matching, then
the probability that the algorithm says 'no',
is at most $n/p \leq 1/n$.

Exercise: In the proof of Tutte's theorem, we considered $\det(A)$ to be a polynomial over the integers. Is the theorem true when we consider $\det(A)$ as a polynomial over Z_p ?

Parallel algorithms

PRAM – Parallel Random Access Machine

NC - class of problems that can be solved in $O(\log^k n)$ time, for some fixed k, using a polynomial number of processors

NC^k - class of problems that can be solved using uniform bounded fan-in Boolean circuits of depth $O(\log^k n)$ and polynomial size

Parallel matching algorithms

Determinants can be computed
very quickly in parallel

$$DET \in NC^2$$

Perfect matchings can be detected
very quickly in parallel (using randomization)

$$PERFECT\text{-}MATCH \in RNC^2$$

Open problem:

??? $PERFECT\text{-}MATCH \in NC$???

Finding perfect matchings

Self Reducibility

Delete an edge and check
whether there is still a perfect matching

Needs $O(n^2)$ determinant computations

Running time $O(n^{\omega+2})$

Fairly slow…

Not parallelizable!

Finding perfect matchings

Rabin-Vazirani (1986): An edge $\{i,j\} \in E$ is contained in a perfect matching iff $(A^{-1})_{ij} \neq 0$.

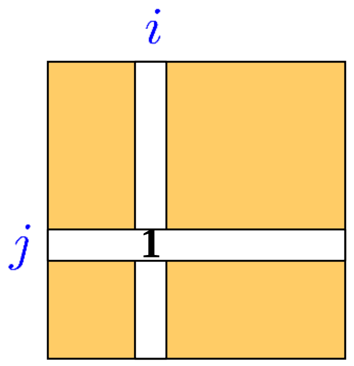
Leads immediately to an $O(n^{\omega+1})$ algorithm: Find an allowed edge $\{i,j\} \in E$, delete it and its vertices from the graph, and recompute A^{-1}.

Mucha-Sankowski (2004): Recomputing A^{-1} from scratch is very wasteful. Running time can be reduced to $O(n^{\omega})$!

Harvey (2006): A simpler $O(n^{\omega})$ algorithm.

Adjoint and Cramer's rule

$$(\mathrm{adj}(A))_{ij} \;=\; (-1)^{i+j}\,\det(A^{j,i}) \;=\; \det$$



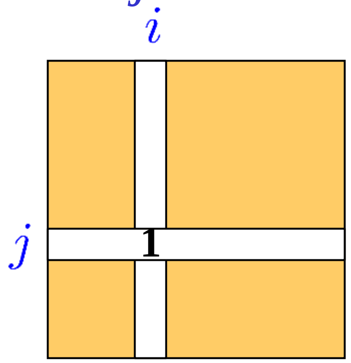
A with the *j*-th row
and *i*-th column deleted

Cramer's rule: $$A^{-1} \;=\; \frac{\mathrm{adj}(A)}{\det(A)}$$

Finding perfect matchings

Rabin-Vazirani (1986): An edge $\{i,j\} \in E$ is contained in a perfect matching iff $(A^{-1})_{ij} \neq 0$.

$$(\mathrm{adj}(A))_{ij} = (-1)^{i+j} \det(A^{j,i}) = \det$$



Leads immediately to an $O(n^{\omega+1})$ algorithm:
Find an allowed edge $\{i,j\} \in E$, delete it and its vertices from the graph, and recompute A^{-1}.

Still not parallelizable

Finding unique minimum weight perfect matchings

[Mulmuley-Vazirani-Vazirani (1987)]

Suppose that edge $\{i,j\} \in E$ has integer weight w_{ij}

Suppose that there is a unique minimum weight perfect matching M of total weight W

Replace x_{ij} by $2^{w_{ij}}$

Then, $2^{2W} \mid \det(A)$ but $2^{2W+1} \nmid \det(A)$

Furthermore, $\{i,j\} \in M$ iff $\frac{2^{w_{ij}} \det(A^{ij})}{2^{2W}}$ is odd

Exercise: Prove the last two claims

Isolating lemma

[Mulmuley-Vazirani-Vazirani (1987)]

Suppose that G has a perfect matching

Assign each edge $\{i,j\}\in E$
a random integer weight $w_{ij}\in[1,2m]$

Lemma: With probability of at least ½, the minimum weight perfect matching of G is unique

Lemma holds for general collections of sets, not just perfect matchings

Proof of Isolating lemma

[Mulmuley-Vazirani-Vazirani (1987)]

An edge $\{i,j\}$ is ambivalent if there is a minimum weight perfect matching that contains it and another that does not

If minimum not unique, at least one edge is ambivalent

Assign weights to all edges except $\{i,j\}$

Let a_{ij} be the largest weight for which $\{i,j\}$ participates in some minimum weight perfect matchings

If $w_{ij}<a_{ij}$, then $\{i,j\}$ participates in all minimum weight perfect matchings

$\{i,j\}$ can be ambivalent only if $w_{ij}=a_{ij}$

The probability that $\{i,j\}$ is ambivalent is at most $1/(2m)$!

Finding perfect matchings

[Mulmuley-Vazirani-Vazirani (1987)]

Choose random weights in $[1,2m]$

Compute determinant and adjoint

Read of a perfect matching (w.h.p.)

Is using $2m$-bit integers **cheating**?

Not if we are willing to pay for it!

Complexity is $O(mn^{\omega}) \leq O(n^{\omega+2})$

Finding perfect matchings in RNC^2

Improves an RNC^3 algorithm by

[Karp-Upfal-Wigderson (1986)]

Multiplying two N-bit numbers

"School method"

$$N^2$$

[Schönhage-Strassen (1971)]

$$N \log N \log \log N$$

[Fürer (2007)]

[De-Kurur-Saha-Saptharishi (2008)]

$$N \log N \, 2^{O(\log^* N)}$$

For our purposes… $\tilde{O}(N)$

Karatsuba's Integer Multiplication [Karatsuba and Ofman (1962)]

$$x = x_1 2^{n/2} + x_0 \qquad u = (x_1 + x_0)(y_1 + y_0)$$

$$y = y_1 2^{n/2} + y_0 \qquad v = x_1 y_1$$

$$w = x_0 y_0$$

$$xy = v\,2^n + (u - v - w)2^{n/2} + w$$

$$T(n) = 3T(n/2+1) + O(n)$$

$$T(n) = \Theta(n^{\lg 3}) = O(n^{1.59})$$

Finding perfect matchings

The story not over yet…

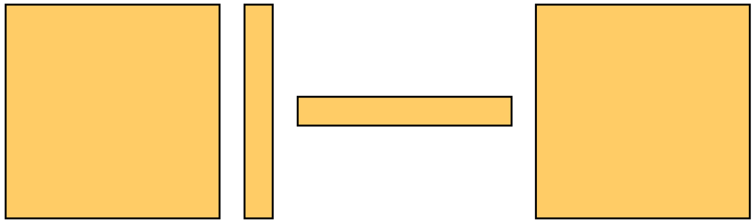
[Mucha-Sankowski (2004)]
Recomputing A^{-1} from scratch is wasteful.
Running time can be reduced to $O(n^{\omega})$!

[Harvey (2006)]
A simpler $O(n^{\omega})$ algorithm.

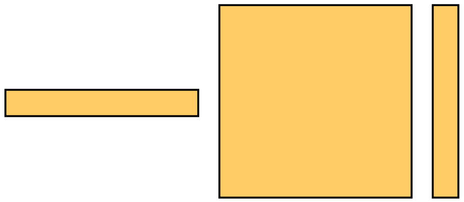
Sherman-Morrison formula

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1} u}$$

$A^{-1}uv^T A^{-1}$:



$v^T A^{-1} u$:



Inverse of a rank one update
is a rank one update of the inverse

Inverse can be updated in O(n^2) time

Finding perfect matchings

A simple O(n^3)-time algorithm

[Mucha-Sankowski (2004)]

Let A be a random Tutte matrix

Compute A^{-1}

Repeat $n/2$ times:

Find an edge $\{i,j\}$ that appears in a perfect matching (i.e., $A_{i,j} \neq 0$ *and* $(A^{-1})_{i,j} \neq 0$)

Zero all entries in the i-th and j-th rows and columns of A, and let $A_{i,j}=1$, $A_{j,i}=-1$

Update A^{-1}

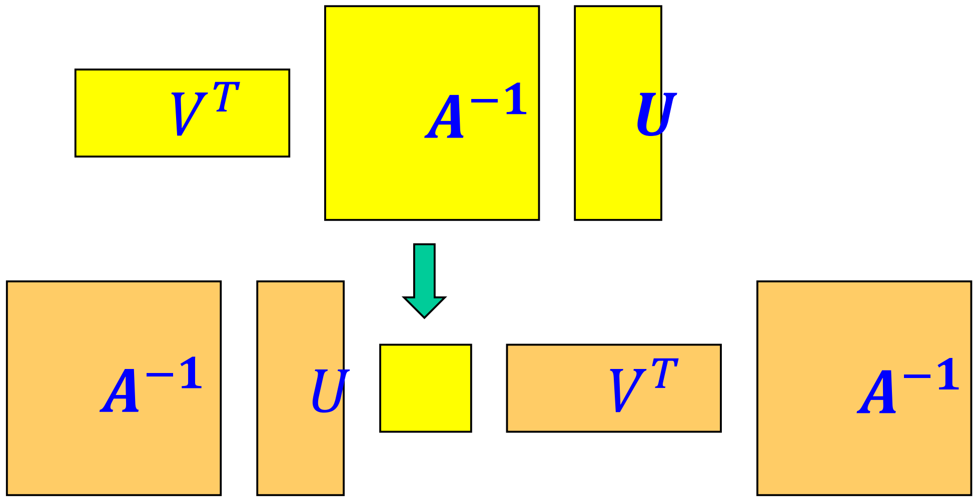
Exercise: Is it enough that the random Tutte matrix A, chosen at the beginning of the algorithm, is invertible?

What is the success probability of the algorithm if the elements of A are chosen from Z_p

Sherman-Morrison-Woodbury formula

$$(A + UV^T)^{-1} =$$

$$A^{-1} - A^{-1}U\,(I + V^T A^{-1} U)^{-1}\,V^T A^{-1}$$



Inverse of a rank k update
is a rank k update of the inverse

Can be computed in O($M(n,k,n)$) time

A Corollary [Harvey (2009)]

Let A be an invertible matrix and let $S \subseteq [n]$.
Let $\tilde{A}$ be a matrix that differs from A only in $S \times S$.
Let $\Delta = \tilde{A}_{S,S} - A_{S,S}$.

Then, $\tilde{A}$ is invertible iff $\det(I + \Delta(A^{-1})_{S,S}) \neq 0$

If $\tilde{A}$ is invertible then

$$\tilde{A}^{-1} = A^{-1} - (A^{-1})_{\star,S}(I+\Delta(A^{-1})_{S,S})^{-1}\Delta(A^{-1})_{S,\star}$$

In particular,

$$(\tilde{A}^{-1})_{S,S} = (A^{-1})_{S,S} - (A^{-1})_{S,S}(I + \Delta(A^{-1})_{S,S})^{-1}\Delta(A^{-1})_{S,S}$$

Harvey's algorithm [Harvey (2009)]

Go over the edges one by one and *delete* an edge if there is still a perfect matching after its deletion

Check the edges for *deletion* in a clever order!

Concentrate on small portion of the matrix and update only this portion after each deletion

Instead of *selecting* edges,
as done by Rabin-Vazirani,
we *delete* edges

Can we delete edge $\{i,j\}$?

Set $a_{i,j}$ and $a_{j,i}$ to 0

Check whether the matrix is still invertible

We are only changing $A_{S,S}$, *where* $S=\{i,j\}$

New matrix is invertible iff

$$\det(I + \Delta(A^{-1})_{S,S}) \neq 0$$

$$\det\left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & a_{i,j} \\ -a_{i,j} & 0 \end{pmatrix}\begin{pmatrix} 0 & b_{i,j} \\ -b_{i,j} & 0 \end{pmatrix}\right)$$

$$= \det\begin{pmatrix} 1 + a_{i,j}b_{i,j} & 0 \\ 0 & 1 + a_{i,j}b_{i,j} \end{pmatrix} = (1 + a_{i,j}b_{i,j})^2$$

$\{i,j\}$ can be deleted iff $a_{i,j}\, b_{i,j} \neq -1 \pmod{p}$

Harvey’s algorithm [Harvey (2009)]

```
Find-Perfect-Matching($G$=($V$=[$n$],$E$)):
Let $A$ be a the Tutte matrix of $G$
Assign random values to the variables of $A$
If $A$ is singular, return ‘no’
Compute $B = A^{-1}$
Delete-In($V$)
Return the set of remaining edges
```

Harvey's algorithm [Harvey (2009)]

If $S \subseteq V$, **Delete-In**(S) deletes all possible edges connecting two vertices in S

If $S, T \subseteq V$, **Delete-Between**(S,T) deletes all possible edges connecting S and T

We assume $|S| = |T| = 2^k$

Before calling
Delete-In(S) and **Delete-Between**(S,T)
keep copies of
$A[S,S]$, $B[S,S]$, $A[S{\cup}T, S{\cup}T]$, $B[S{\cup}T, S{\cup}T]$

```
Delete-In($S$):
If $|S| = 1$ then return
Divide $S$ in half: $S = S_1 \cup S_2$
For $i \in \{1,2\}$
    Delete-In($S_i$)
    Update $B[S,S]$
Delete-Between($S_1$,$S_2$)
```

Invariant: When entering and exiting, A is up to date, and $B[S,S]=(A^{-1})[S,S]$

Same **Invariant** with $B[S∪T, S∪T]$

```
Delete-Between($S$,$T$):
If $|S| = 1$ then
    Let $s \in S$ and $t \in T$
    If $A_{s,t} = 0$ and $A_{s,t}\, B_{s,t} \neq -1$ then
        // Edge $\{s,t\}$ can be deleted
        Set $A_{s,t} = A_{t,s} = 0$
        Update $B[S∪T, S∪T]$  // (Not really necessary!)
Else
    Divide in half: $S = S_1 \cup S_2$ and $T = T_1 \cup T_2$
    For $i \in \{1, 2\}$ and for $j \in \{1, 2\}$
        Delete-Between($S_i$,$T_j$)
        Update $B[S∪T, S∪T]$
```

Maximum matchings

Theorem: [Lovasz (1979)]

Let A be the symbolic Tutte matrix of G. Then $\text{rank}(A)$ is twice the size of the maximum matching in G.

If $|S|=\text{rank}(A)$ and $A[S,*]$ is of full rank, then $G[S]$ has a perfect matching, which is a maximum matching of G.

Corollary: Maximum matchings can be found in $O(n^{\omega})$ time

"Exact matchings" [MVV (1987)]

Let G be a graph. Some of the edges are red.
The rest are black. Let k be an integer.
Is there a perfect matching in G
with exactly k red edges?

Exercise*: Give a *randomized* polynomial time algorithm for the exact matching problem

No *deterministic* polynomial time algorithm is known for the exact matching problem!

MIN-PLUS MATRIX MULTIPLICATION

AND

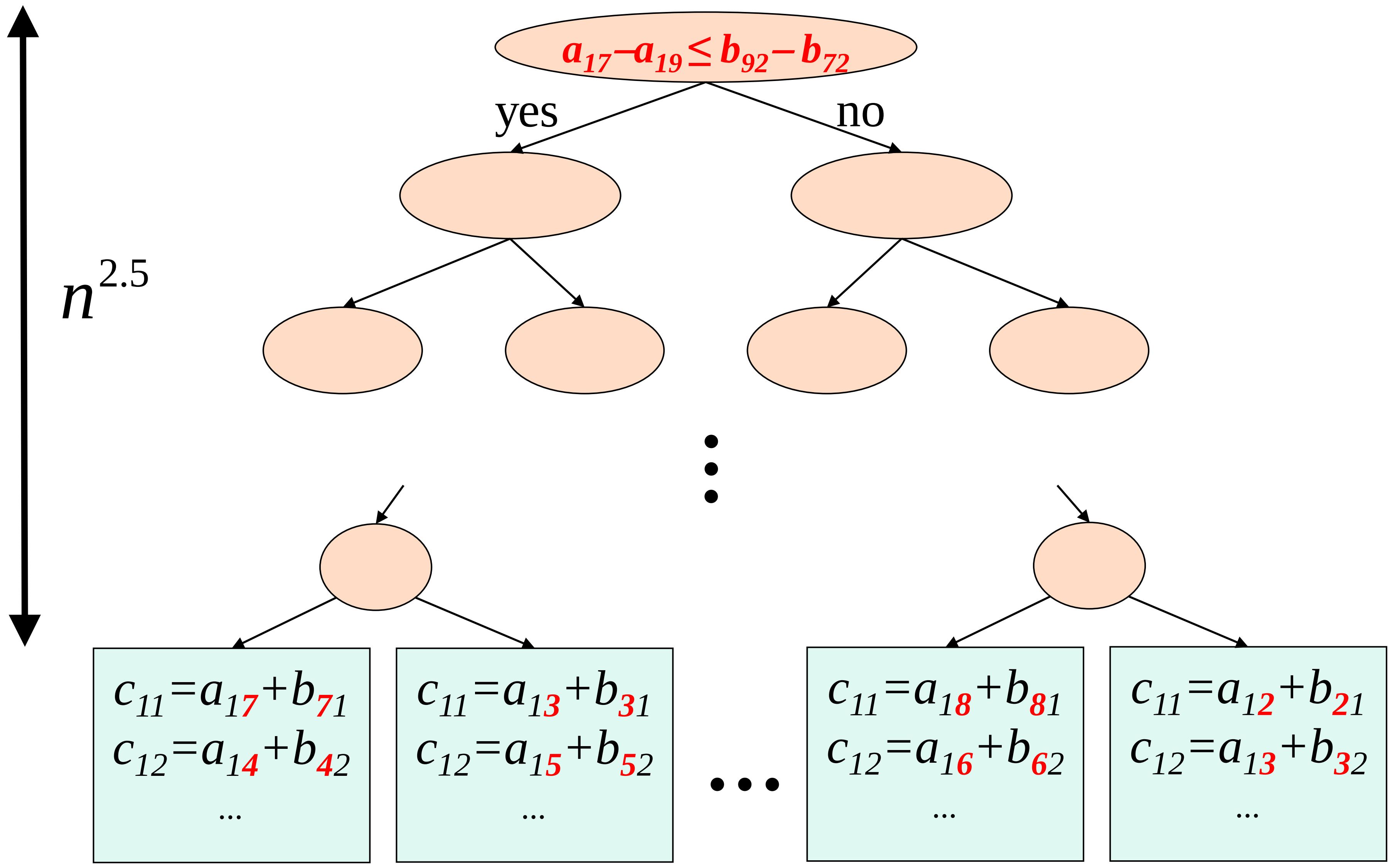
ALL-PAIRS SHORTEST PATHS (APSP)

Fredman's trick [Fredman (1976)]

The **min-plus** product of two $n \times n$ matrices can be **deduced** after only $O(n^{2.5})$ additions and comparisons.

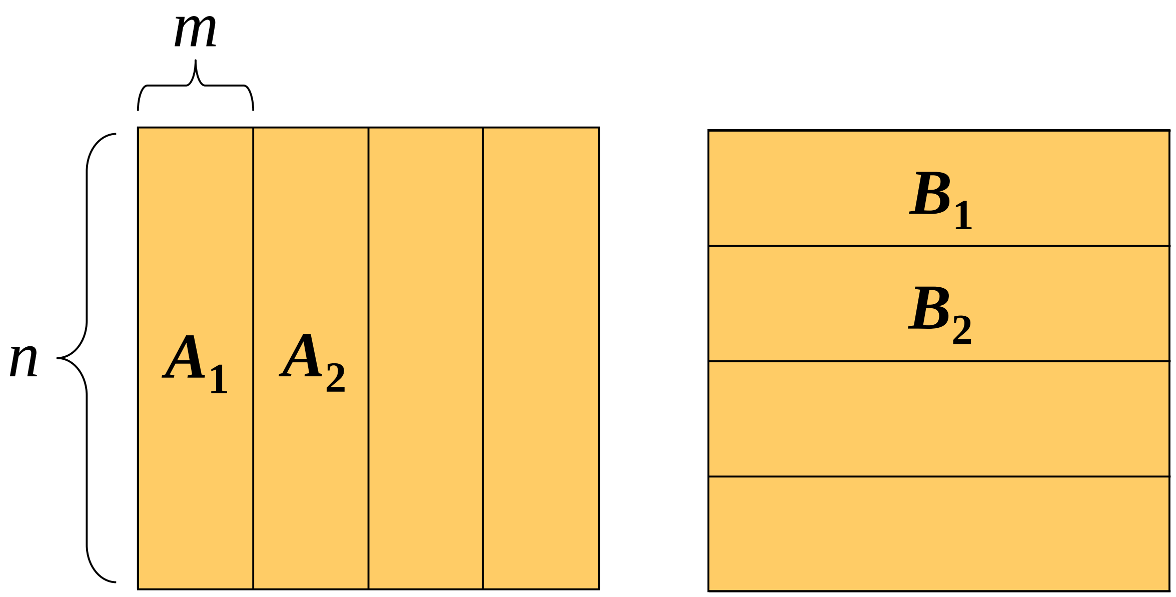
It is not known how to implement the algorithm in $O(n^{2.5})$ time.

Algebraic Decision Trees



$a_{17}-a_{19} \leq b_{92}-b_{72}$
yes
no
$n^{2.5}$
$c_{11}=a_{17}+b_{71}$ $c_{12}=a_{14}+b_{42}$ …
$c_{11}=a_{13}+b_{31}$ $c_{12}=a_{15}+b_{52}$ …
…
$c_{11}=a_{18}+b_{81}$ $c_{12}=a_{16}+b_{62}$ …
$c_{11}=a_{12}+b_{21}$ $c_{12}=a_{13}+b_{32}$ …

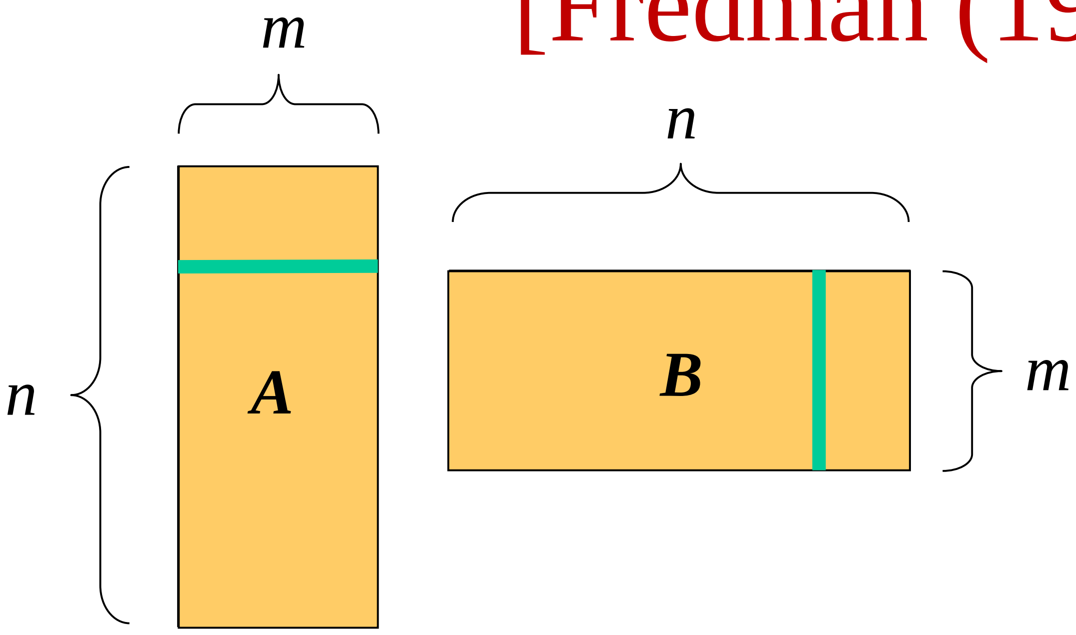
Breaking a square product into several rectangular products



$$A * B = \min_i A_i * B_i$$

MPP(n) ≤ (n/m) (MPP(n,m,n) + n^2)

Fredman's trick [Fredman (1976)]



$$a_{i,r}+b_{r,j} \le a_{i,s}+b_{s,j}$$
$$\Updownarrow$$
$$a_{i,r}-a_{i,s} \le b_{s,j}-b_{r,j}$$

Naïve calculation requires n^2m operations

Fredman observed that the result can be inferred after performing only $O(nm^2)$ operations

Fredman's trick (cont.)

$$a_{i,r}+b_{r,j} \le a_{i,s}+b_{s,j} \quad \Leftrightarrow \quad a_{i,r}-a_{i,s} \le b_{s,j}-b_{r,j}$$

- **Sort** all the differences $a_{i,r}-a_{i,s}$ and $b_{s,j}-b_{r,j}$
- Trivially using $O(m^2 n \log n)$ comparisons
- (Actually enough to sort separately for every r,s)
- Non-Trivially using $O(m^2 n)$ comparisons

The ordering of the elements in the sorted list determines the result of the min-plus product !!!

Sorting differences

$$a_{i,r}+b_{r,j} \le a_{i,s}+b_{s,j} \;\Leftrightarrow\; a_{i,r}-a_{i,s} \le b_{s,j}-b_{r,j}$$

Sort all $a_{i,r} - a_{i,s}$ and all $b_{s,j} - b_{r,j}$ and the merge

Number of orderings of the m^2n differences $a_{i,r} - a_{i,s}$ is at most the number of regions in $\mathbb{R}^{mn}$ defined by the $(m^2n)^2$ hyperplanes $a_{i,r} - a_{i,s} = a_{i',r'} - a_{i',s'}$

Lemma: Number of regions in $\mathbb{R}^d$ defined by N hyperplanes is at most $\binom{N}{0} + \binom{N}{1} + \cdots + \binom{N}{d}$

Theorem: [Fredman (1976)] If a sequence of n items is known to be in one of Γ different orderings, then it can be sorted using at most $\log_2 \Gamma + 2n$ comparisons

All-Pairs Shortest Paths

in directed graphs with “real” edge weights

Running time	**Authors**
n^3	[Floyd (1962)] [Warshall (1962)]
$\dfrac{n^3}{\left(\frac{\log n}{\log\log n}\right)^{1/3}}$	[Fredman (1976)]
$\vdots$	$\vdots$
$\dfrac{n^3}{2^{\Omega\left(\left(\frac{\log n}{\log\log n}\right)^{1/2}\right)}}$	[Williams (2014)]

Sub-cubic equivalences

in graphs with integer edge weights in $[-M, M]$

[Williams-Williams (2010)]

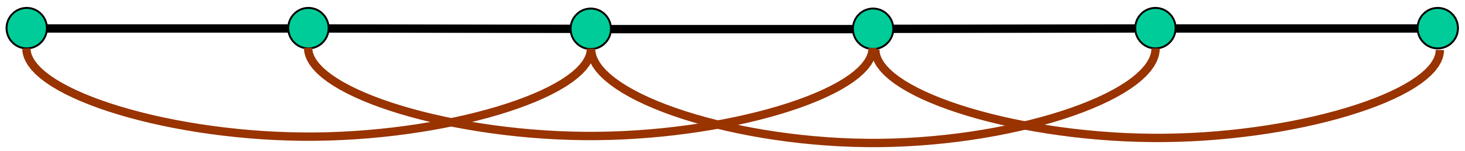
If one of the following problems has
an $O(n^{3-\varepsilon}\text{poly}(\log M))$ algorithm, $\varepsilon > 0$,
then all have! (Not necessarily with the same ε.)

- Computing a min-plus product
- APSP in weighted directed graphs
- APSP in weighted undirected graphs
- Finding a negative triangle
- Finding a minimum weight cycle (non-negative edge weights)
- Verifying a min-plus product
- Finding replacement paths

UNWEIGHTED
UNDIRECTED
SHORTEST PATHS

Distances in G and its square G^2

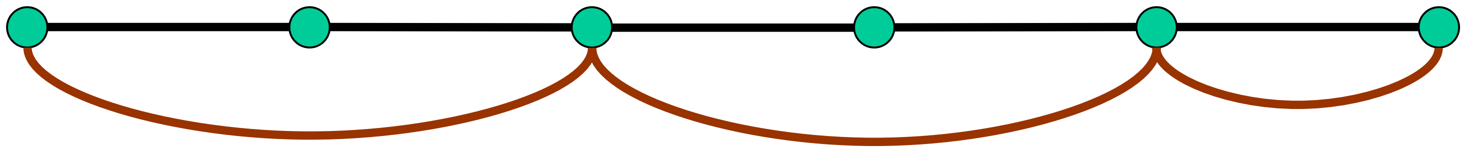
Let $G=(V,E)$. Then $G^2=(V,E^2)$, where $(u,v)\in E^2$ if and only if $(u,v)\in E$ or there exists $w\in V$ such that $(u,w),(w,v)\in E$



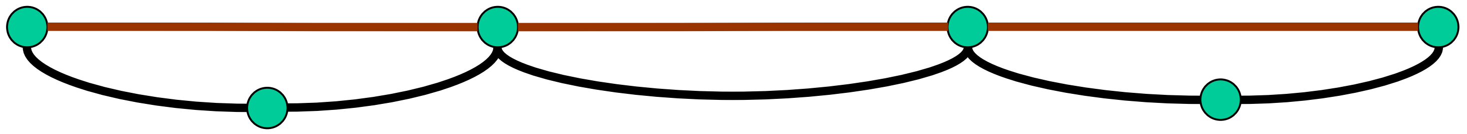
Let $\delta(u,v)$ be the distance from u to v in G.
Let $\delta^2(u,v)$ be the distance from u to v in G^2.

Distances in *G* and its square G^2 (cont.)

Lemma: $\delta^2(u,v) = \lceil \delta(u,v)/2 \rceil$, for every $u,v \in V$.



$$\delta^2(u,v) \le \lceil \delta(u,v)/2 \rceil$$



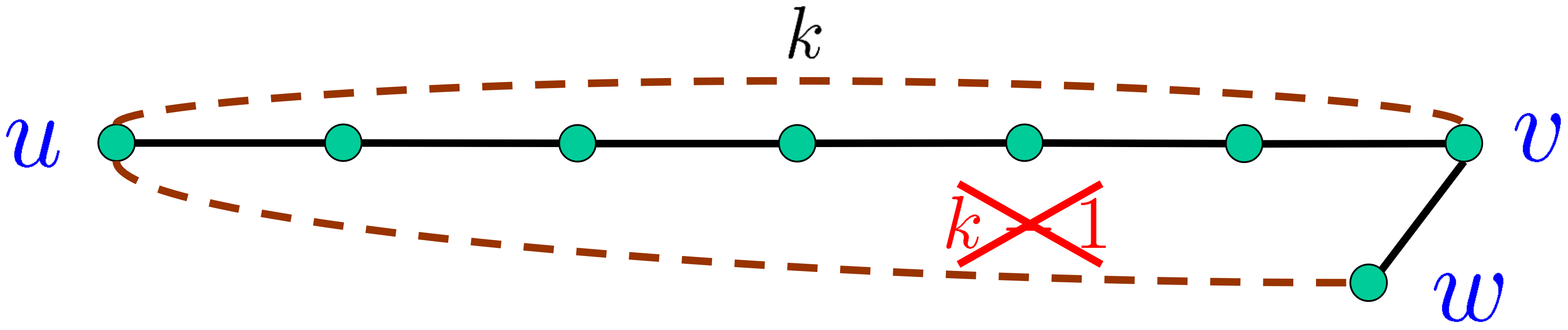
$$\delta(u,v) \le 2\delta^2(u,v)$$

Thus: $\delta(u,v) = 2\delta^2(u,v)$ or

$\delta(u,v) = 2\delta^2(u,v) - 1$

Even distances

Lemma: If $\delta(u,v) = 2\delta^2(u,v)$ then for every neighbor w of v we have $\delta^2(u,w) \geq \delta^2(u,v)$.



Let A be the adjacency matrix of the G.
Let C be the distance matrix of G^2

$$\sum_{(v,w)\in E} c_{uw} = \sum_{w\in V} c_{uw} a_{wv} = (CA)_{uv} \geq \deg(v) c_{uv}$$

Odd distances

Lemma: If $\delta(u,v) = 2\delta^2(u,v)-1$ then for every neighbor w of v we have $\delta^2(u,w) \le \delta^2(u,v)$ and for at least one neighbor $\delta^2(u,w) < \delta^2(u,v)$.

Exercise: Prove the lemma.

Let A be the adjacency matrix of the G.
Let C be the distance matrix of G^2

$$\sum_{(v,w)\in E} c_{uw} = \sum_{w \in V} c_{uw} a_{wv} = (CA)_{uv} < \deg(v) c_{uv}$$

Seidel's algor 95)]

1. If A is an all one matrix, then all distances are 1.
2. Compute A^2, the adjacency matrix of the squared graph.
3. Find, recursively, the distances in the squared graph.
4. Decide, using one integer matrix multiplication, for every two vertices u,v, whether their distance is **twice** the distance in the square, or **twice minus 1**.

Assume that A has 1's on the diagonal.

Boolean matrix multiplicaion

```
A)

else
   $C \leftarrow \mathrm{APD}(A^2)$
   $X \leftarrow CA$ , $\deg \leftarrow Ae$
   $\deg_j]$
```

Integer matrix multiplicaion

Complexity:
$O(n^{\omega} \log n)$

Exercise$^+$: Obtain a version of Seidel's algorithm that uses only **Boolean** matrix multiplications.

Hint: Look at distances also modulo 3.

Distances vs. Shortest Paths

We described an algorithm for computing all **distances**.

How do we get a representation of the **shortest paths**?

We need **witnesses** for the Boolean matrix multiplication.

Witnesses for Boolean Matrix Multiplication

$$C = AB$$

$$c_{ij} = \bigvee_{k=1}^{n} a_{ik} \wedge b_{kj}$$

A matrix *W* is a matrix of **witnesses** iff

$$\text{If } c_{ij} = 0 \text{ then } w_{ij} = 0$$

$$\text{If } c_{ij} = 1 \text{ then } w_{ij} = k \text{ where } a_{ik} = b_{kj} = 1$$

Can be computed naively in $O(n^3)$ time.

Can also be computed in $O(n^\omega \log n)$ time.

Exercise n+1:

a) Obtain a deterministic $O(n^\omega)$-time algorithm for finding **unique** witnesses.

b) Let $1 \le d \le n$ be an integer. Obtain a *randomized* $O(n^\omega)$-time algorithm for finding witnesses for all positions that have between d and $2d$ witnesses.

c) Obtain an $O(n^\omega \log n)$-time *randomized* algorithm for finding all witnesses.

Hint: In b) use **sampling**.

All-Pairs Shortest Paths

in graphs with small integer weights

Undirected graphs.

Edge weights in $\{0,1,\ldots M\}$

Running time	**Authors**
Mn^{ω}	[Shoshan-Zwick ’99]

Improves results of

[Alon-Galil-Margalit ’91] [Seidel ’95]

DIRECTED
SHORTEST PATHS

Exercise:

Obtain an $O(n^\omega \log n)$-time algorithm for computing the **diameter** of an unweighted directed graph.

Exercise:

For every $\varepsilon > 0$, give an $O(n^\omega \log n)$-time algorithm for computing $(1+\varepsilon)$-approximations of all distances in an unweighted directed graph.

Using matrix multiplication to compute min-plus products

$$\begin{pmatrix} c_{11} & c_{12} & \\ c_{21} & c_{22} & \\ & & \ddots \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \\ a_{21} & a_{22} & \\ & & \ddots \end{pmatrix} * \begin{pmatrix} b_{11} & b_{12} & \\ b_{21} & b_{22} & \\ & & \ddots \end{pmatrix}$$

$$c_{ij} = \min_k \{a_{ik} + b_{kj}\}$$

$$\begin{pmatrix} c'_{11} & c'_{12} & \\ c'_{21} & c'_{22} & \\ & & \ddots \end{pmatrix} = \begin{pmatrix} x^{a_{11}} & x^{a_{12}} & \\ x^{a_{21}} & x^{a_{22}} & \\ & & \ddots \end{pmatrix} \times \begin{pmatrix} x^{b_{11}} & x^{b_{12}} & \\ x^{b_{21}} & x^{b_{22}} & \\ & & \ddots \end{pmatrix}$$

$$c'_{ij} = \sum_k x^{a_{ik}+b_{kj}} \qquad c_{ij} = \mathit{first}(c'_{ij})$$

Using matrix multiplication to compute min-plus products

Assume: $0 \le a_{ij}, b_{ij} \le M$

$$\begin{pmatrix} c'_{11} & c'_{12} & \\ c'_{21} & c'_{22} & \\ & & \ddots \end{pmatrix} = \begin{pmatrix} x^{a_{11}} & x^{a_{12}} & \\ x^{a_{21}} & x^{a_{22}} & \\ & & \ddots \end{pmatrix} * \begin{pmatrix} x^{b_{11}} & x^{b_{12}} & \\ x^{b_{21}} & x^{b_{22}} & \\ & & \ddots \end{pmatrix}$$

n^{ω} polynomial products $\times$ M operations per polynomial product $=$ Mn^{ω} operations per min-plus product

Trying to implement the repeated squaring algorithm

```
$D \leftarrow W$
for $i \leftarrow 1$ to $\log_2 n$
    $D \leftarrow D*D$
```

Consider an easy case: all weights are 1

After the i-th iteration, the finite elements in D are in the range $\{1,\ldots,2^i\}$.

The cost of the min-plus product is $2^i n^\omega$

The cost of the last product is $n^{\omega+1}$!!!

Sampled Repeated Squaring [Z (1998)]

```
$D \leftarrow W$
for $i \leftarrow 1$ to $\log_{3/2} n$ do
{
    $s \leftarrow (3/2)^{i+1}$
    $B \leftarrow \text{rand}(V, (9n \ln n)/s)$
    $D \leftarrow \min\{\, D \,, D[V,B] * D[B,V] \,\}$
}
```

Choose a subset of V of size $\approx n/s$

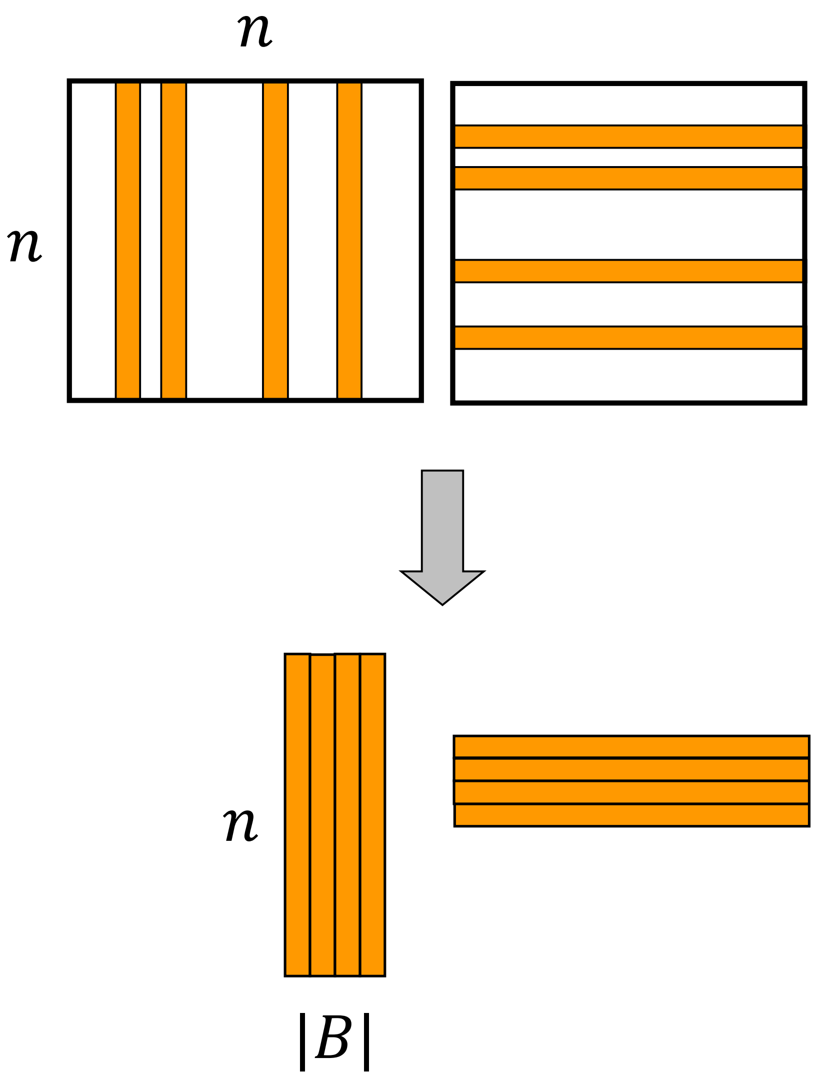
Select the **columns** of D whose indices are in B

Select the **rows** of D whose indices are in B

With high probability, all distances are correct!

The is also a slightly more complicated deterministic algorithm

Sampled Distance Products (Z '98)



In the i-th iteration, the set B is of size $\approx n/s$, where $s = (3/2)^{i+1}$

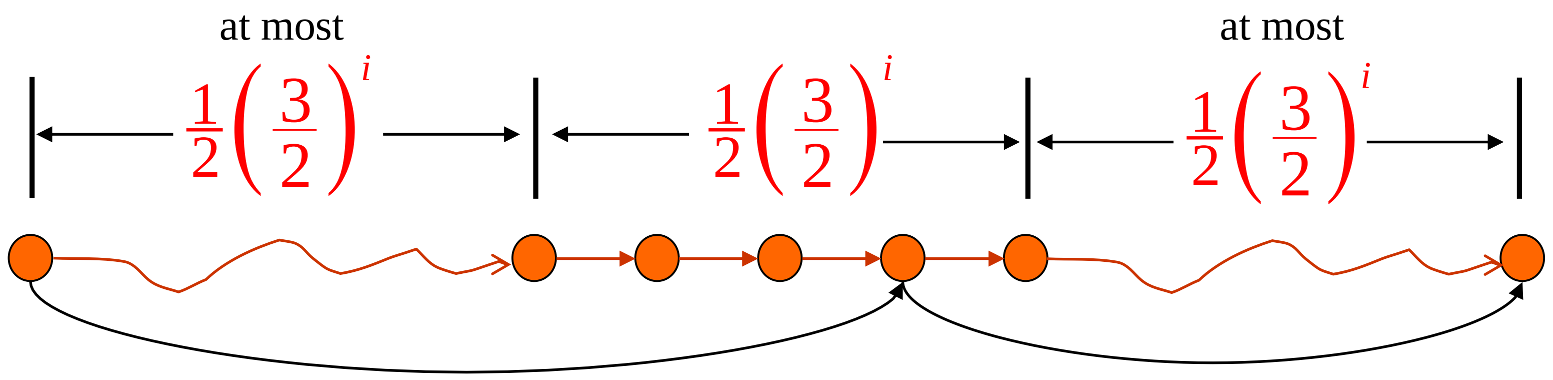
The matrices get smaller and smaller but the elements get larger and larger

Sampled Repeated Squaring - Correctness

```
$D \leftarrow W$
for $i \leftarrow 1$ to $\log_{3/2} n$ do
{
        $s \leftarrow (3/2)^{i+1}$
        $B \leftarrow \text{rand}(V,(9n \ln n)/s)$
        $D \leftarrow \min\{ D , D[V,B]*D[B,V] \}$
}
```

Invariant: After the i-th iteration, distances that are attained using at most $(3/2)^i$ edges are correct.

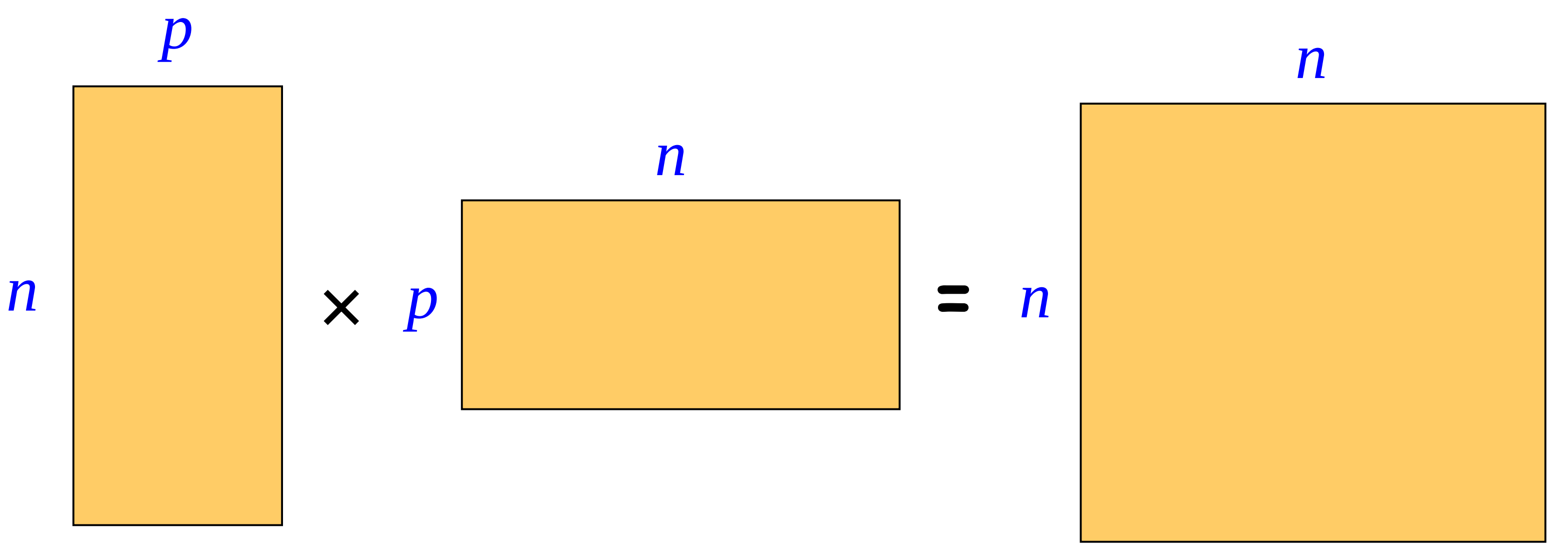
Consider a shortest path that uses at most $(3/2)^{i+1}$ edges


at most
$\frac{1}{2}\left(\frac{3}{2}\right)^i$
$\frac{1}{2}\left(\frac{3}{2}\right)^i$
at most
$\frac{1}{2}\left(\frac{3}{2}\right)^i$

Let $s = (3/2)^{i+1}$

Failure probability : $\left(1-\frac{9\ln n}{s}\right)^{s/3} < n^{-3}$

Rectangular Matrix multiplication



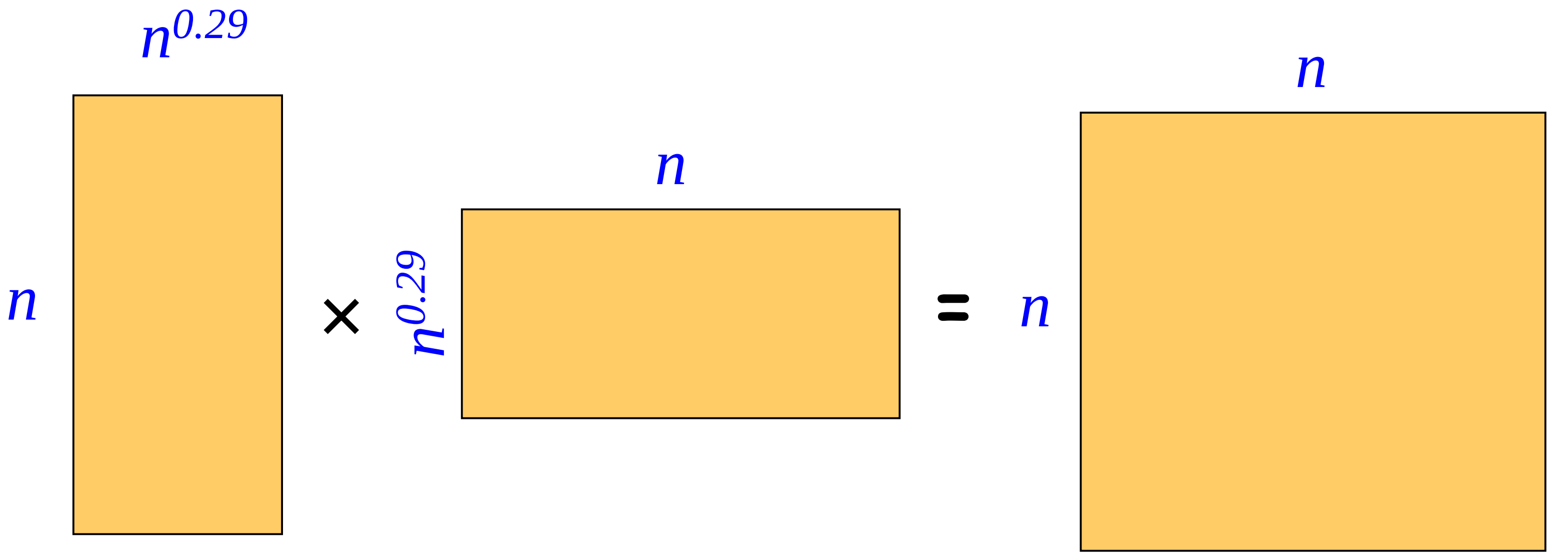
Naïve complexity: n^2p

[Coppersmith (1997)] [Huang-Pan (1998)]

$$n^{1.85}p^{0.54}+n^{2+o(1)}$$

For $p \le n^{0.29}$, complexity $= n^{2+o(1)}$!!!

Rectangular Matrix multiplication



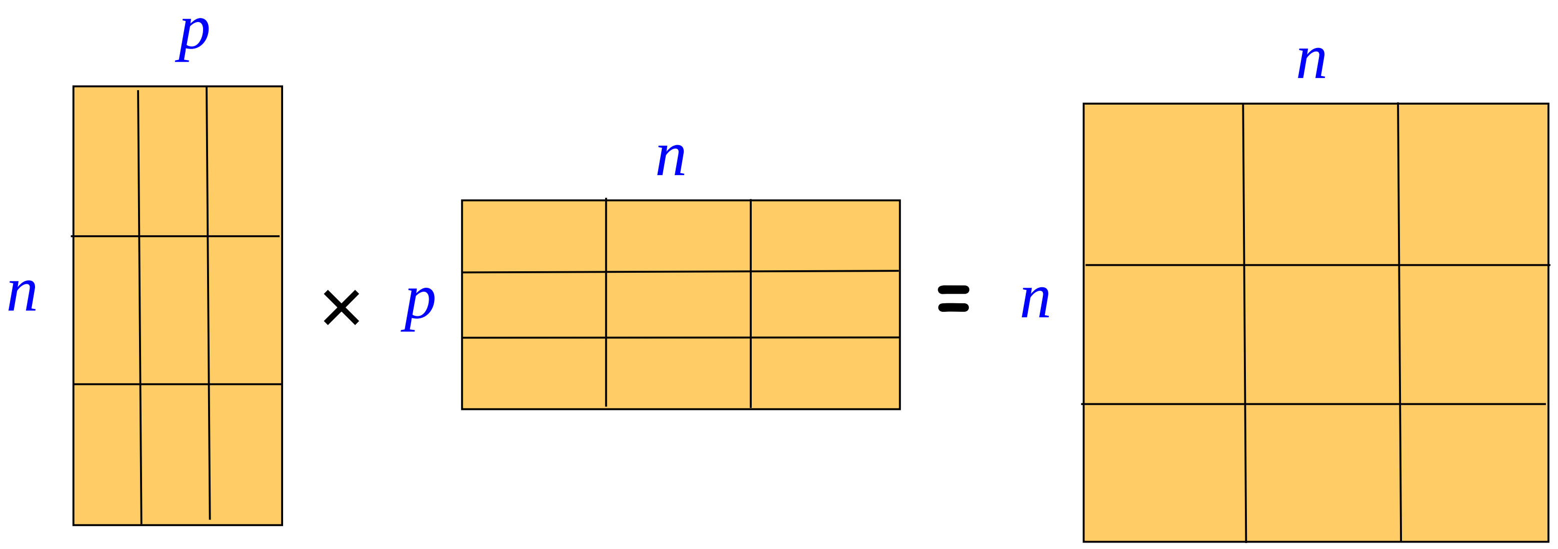
[Coppersmith (1997)]

$n \times n^{0.29}$ by $n^{0.29} \times n$

$n^{2+o(1)}$ operations!

$\alpha = 0.29\ldots$

Rectangular Matrix multiplication



[Huang-Pan (1998)]

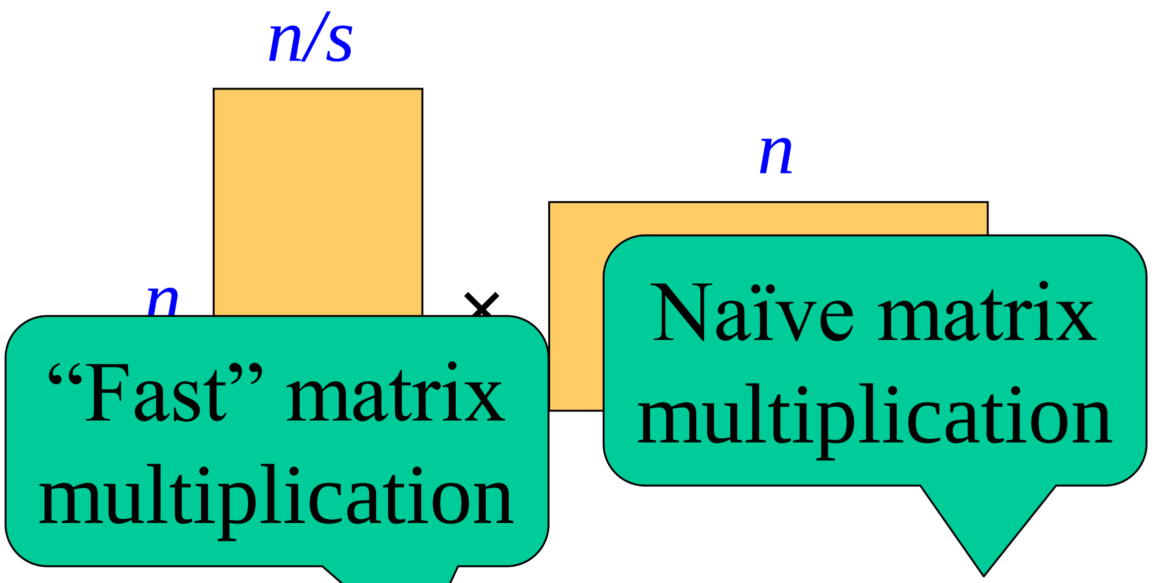
Break into $q \times q^{\alpha}$ and $q^{\alpha} \times q$ sub-matrices

$$q = \left(\frac{n}{p}\right)^{\frac{1}{1-\alpha}} \qquad \left(\frac{n}{q}\right)^{\omega} \cdot q^2 \quad = \quad n^{\omega - \frac{\omega-2}{1-\alpha}} \cdot p^{\frac{\omega-2}{1-\alpha}}$$

$$\approx \quad n^{1.85} p^{0.54}$$

Complexity of APSP algorithm

The i-th iteration:



$s = (3/2)^{i+1}$

The elements are of absolute value at most Ms

$$\min\{\, Ms \cdot n^{1.85} \left(\frac{n}{s}\right)^{0.54}, \frac{n^3}{s} \} \;\le\; M^{0.68} n^{2.58}$$

Complexity of APSP algorithm

Exercise:

The claim that the elements in the matrix in the i-th iteration are of absolute value at most Ms, where $s = (3/2)^{i+1}$, is not true. Explain why and how it can be fixed.

Open problem:

Can APSP in unweighted directed graphs be solved in $O(n^{\omega})$ time?

[Yuster-Z (2005)]

A directed graphs can be processed in $O(n^{\omega})$ time so that any distance query can be answered in $O(n)$ time.

Corollary:

SSSP in directed graphs in $O(n^{\omega})$ time.

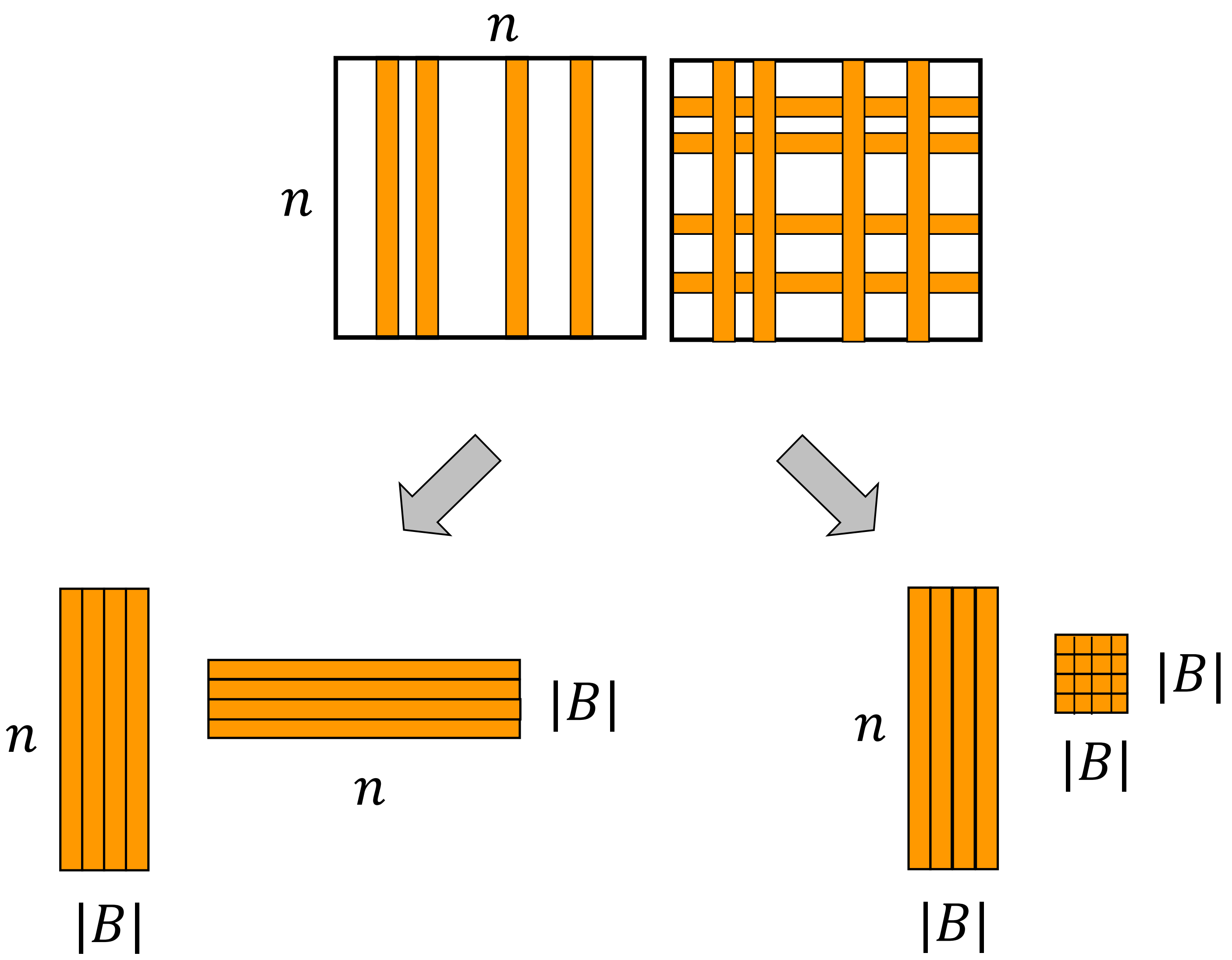
Also obtained, using a different technique, by [Sankowski (2005)]

The preprocessing algorithm [YZ (2005)]

⍰

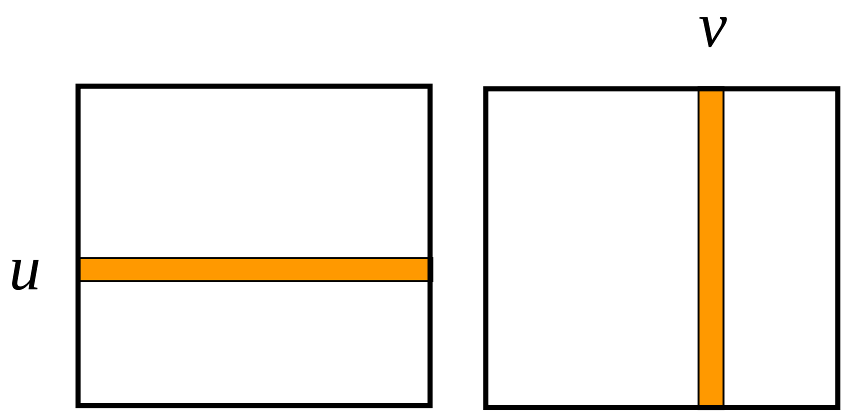
```
$D \quad W$ ; $B \leftarrow V$
for $i \leftarrow 1$ to $\log_{3/2} n$ do
{
    $s \leftarrow (3/2)^{i+1}$
    $B \leftarrow \mathrm{rand}(B, (9n \ln n)/s)$
    $D[V,B] \leftarrow \min\{ D[V,B] , D[V,B] * D[B,B] \}$
    $D[B,V] \leftarrow \min\{ D[B,V] , D[B,B] * D[B,V] \}$
}
```

Twice Sampled Distance Products



The query answering algorithm

$$\boxed{\boldsymbol{\delta(u,v)} \leftarrow \boldsymbol{D[\{u\},V] * D[V,\{v\}]}}$$



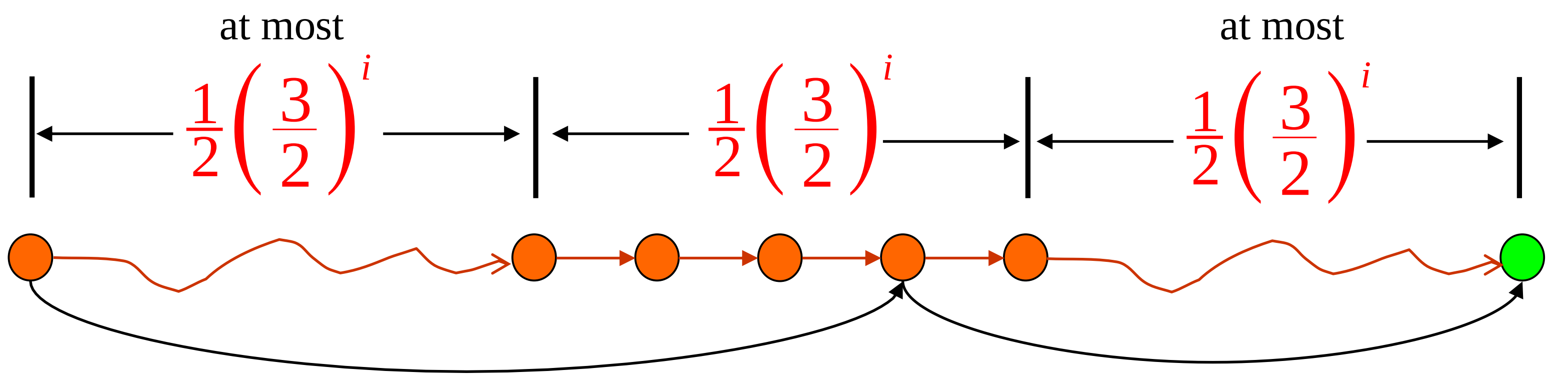
Query time: $O(n)$

The preprocessing algorithm: Correctness

Let B_i be the i-th sample. $B_1 \supseteq B_2 \supseteq B_3$ …

Invariant: After the i-th iteration, if $u \in Bi$ or $v \in Bi$ and there is a shortest path from u to v that uses at most $(3/2)^i$ edges, then $D(u,v) = \delta(u,v)$.

Consider a shortest path that uses at most $(3/2)^{i+1}$ edges


at most
$\frac{1}{2}\left(\frac{3}{2}\right)^i$
$\frac{1}{2}\left(\frac{3}{2}\right)^i$
at most
$\frac{1}{2}\left(\frac{3}{2}\right)^i$

Answering distance queries

Directed graphs. Edge weights in $\{-M,\ldots,0,\ldots M\}$

Preprocessing time	**Query time**	**Authors**
$Mn^{2.38}$	n	[Yuster-Zwick (2005)]

In particular, any $Mn^{1.38}$ distances can be computed in $Mn^{2.38}$ time.

For dense enough graphs with small enough edge weights, this improves on Goldberg's SSSP algorithm. $Mn^{2.38}$ vs. $mn^{0.5}\log M$

Approximate All-Pairs Shortest Paths

in graphs with non-negative integer weights

Directed graphs.

Edge weights in $\{0,1,\ldots,M\}$

$(1+\varepsilon)$-approximate distances

Running time	**Authors**
$(n^{2.38}\log M)/\varepsilon$	[Z (1998)]

Open problems

An $O(n^{\omega})$ algorithm for the directed unweighted APSP problem?

An $O(n^{3-\varepsilon})$ algorithm for the APSP problem with edge weights in $\{1,2,\ldots,n\}$?

An $O(n^{2.5-\varepsilon})$ algorithm for the SSSP problem with edge weights in $\{-1,0,1,2,\ldots, n\}$?

DYNAMIC
TRANSITIVE CLOSURE

Dynamic transitive closure

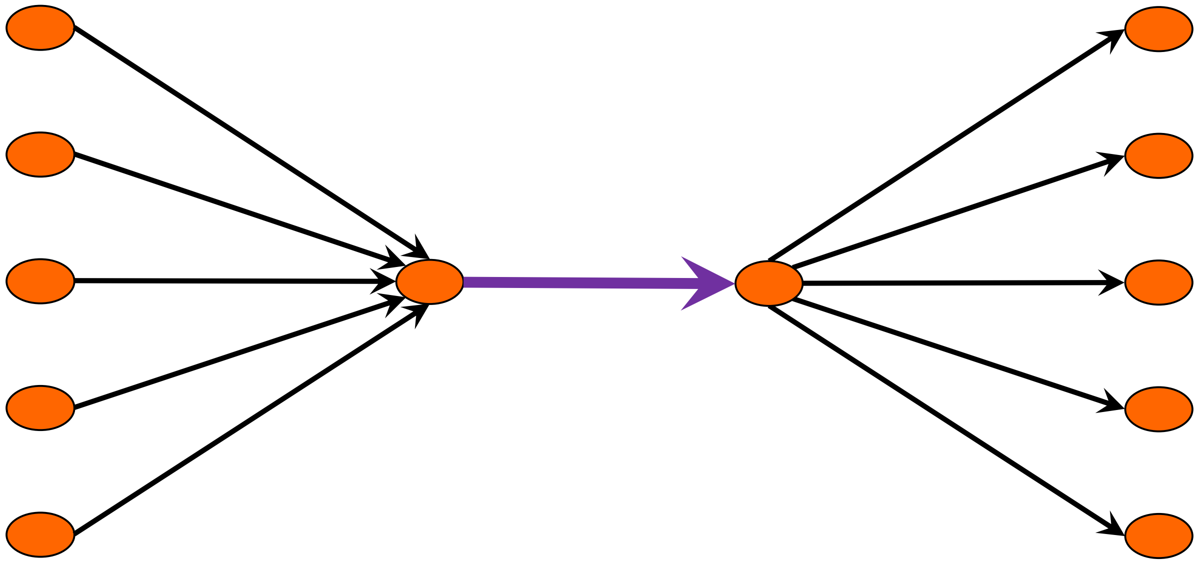
- **Edge-Update**(*e*) – add/remove an edge *e*
- **Vertex-Update**(*v*) – add/remove edges touching *v*.
- **Query**(*u*,*v*) – is there are directed path from *u* to *v*?

[Sankowski '04]

Edge-Update			
Vertex-Update			
Query			

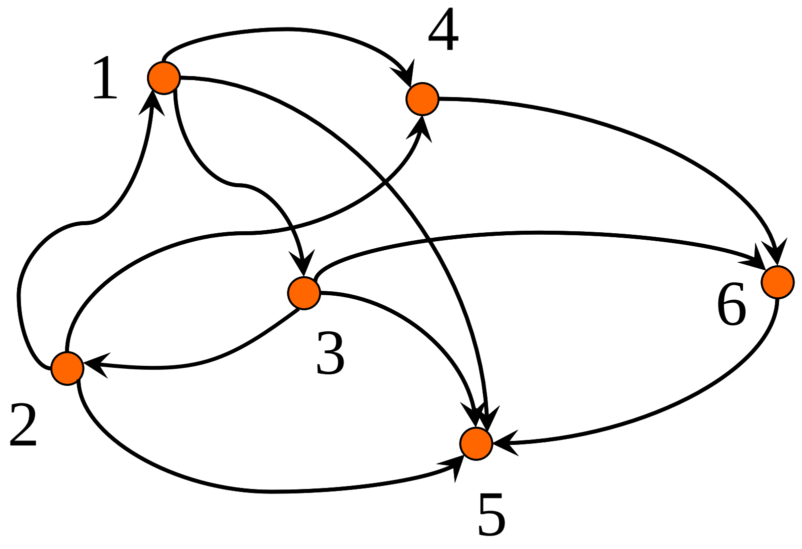
(improving [Demetrescu-Italiano '00], [Roditty '03])

Inserting/Deleting and edge



May change $\Omega(n^2)$ entries of the transitive closure matrix

Symbolic Adjacency matrix



$$\begin{pmatrix} 1 & 0 & x_{13} & x_{14} & x_{15} & 0 \\ x_{21} & 1 & 0 & x_{24} & x_{25} & 0 \\ 0 & x_{32} & 1 & 0 & x_{35} & x_{36} \\ 0 & 0 & 0 & 1 & 0 & x_{46} \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & x_{56} & 1 \end{pmatrix}$$

$$\det(A) \neq 0$$

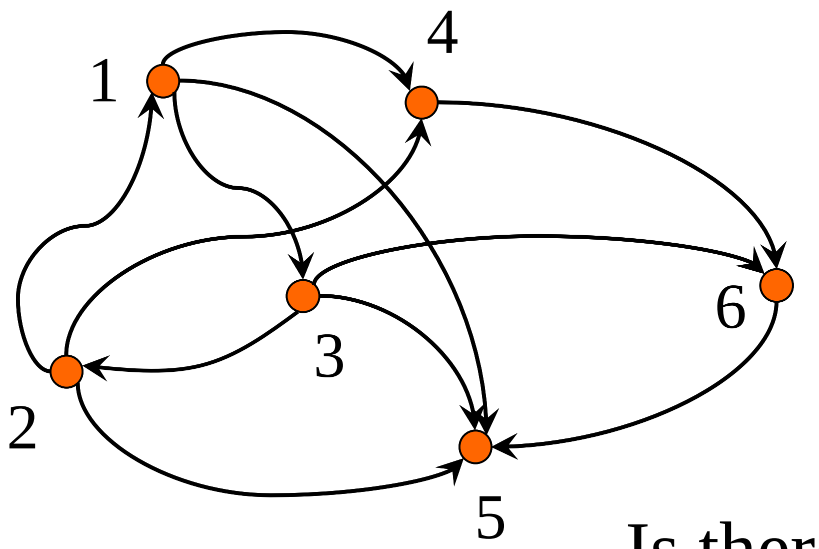
Reachability via adjoint [Sankowski ’04]

Let A be the symbolic adjacency matrix of G.
(With 1s on the diagonal.)

There is a directed path from i to j in G iff

$$(\mathrm{adj}(A))_{ij} \neq 0$$

Reachability via adjoint (example)



$$\begin{pmatrix} 1 & 0 & x_{13} & x_{14} & x_{15} & 0 \\ x_{21} & 1 & 0 & x_{24} & x_{25} & 0 \\ 0 & x_{32} & 1 & 0 & x_{35} & x_{36} \\ 0 & 0 & 0 & 1 & 0 & x_{46} \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & x_{65} & 1 \end{pmatrix}$$

Is there a path from 1 to 5?

$$\det \begin{pmatrix} 0 & 0 & x_{13} & x_{14} & x_{15} & 0 \\ 0 & 1 & 0 & x_{24} & x_{25} & 0 \\ 0 & x_{32} & 1 & 0 & x_{35} & x_{36} \\ 0 & 0 & 0 & 1 & 0 & x_{46} \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_{65} & 1 \end{pmatrix} = \begin{matrix} -x_{15} \\ -x_{13}x_{32}x_{25} \\ +x_{13}x_{35} \\ -x_{13}x_{36}x_{56} \\ -x_{14}x_{46}x_{65} \\ -x_{13}x_{32}x_{24}x_{46}x_{65} \end{matrix}$$

Dynamic transitive closure

- **Edge-Update**(e) – add/remove an edge e
- **Vertex-Update**(v) – add/remove edges touching v.
- **Query**(u,v) – is there are directed path from u to v?

$$\Downarrow$$

Dynamic matrix inverse

- **Entry-Update**(i,j,x) – Add x to A_{ij}
- **Row-Update**(i,v) – Add v to the i-th row of A
- **Column-Update**(j,u) – Add u to the j-th column of A
- **Query**(i,j) – return $(A^{-1})_{ij}$

$O(n^2)$ update / $O(1)$ query algorithm [Sankowski ’04]

Let $p \approx n^3$ be a prime number

Assign random values $a_{ij} \in F_p$ to the variables x_{ij}

Maintain A^{-1} over F_p

Edge-Update → Entry-Update

Vertex-Update → Row-Update + Column-Update

Perform updates using the Sherman-Morrison formula

Small error probability
(by the Schwartz-Zippel lemma)

Lazy updates

Consider single entry updates

$$A_k = A_{k-1} + a_k u_k v_k$$

$$a_k = \pm a_{i_k, j_k} \qquad u_k = e_{i_k} \qquad v_k = e_{j_k}^T$$

$$A_k^{-1} = A_{k-1}^{-1} + \alpha_k u'_k v'_k$$

$$\alpha_k = 1 + a_k v_k A_{k-1}^{-1} u_k = 1 + a_k (A_{k-1}^{-1})_{j_k, i_k}$$

$$u'_k = A_{k-1}^{-1} u_k = (A_{k-1}^{-1})_{*, i_k}$$

$$v'_k = v_k A_{k-1}^{-1} = (A_{k-1}^{-1})_{j_k, *}$$

$$A_k^{-1} = A_0^{-1} + \sum_{i=1}^{k} \alpha_i u'_i v'_i$$

Lazy updates (cont.)

$$A_k^{-1} = A_0^{-1} + \sum_{i=1}^{k} \alpha_i u_i' v_i'$$

Do not maintain A_k^{-1} explicitly!

Maintain $\alpha_i, u_i', v_i',\ i = 1, 2, \ldots, k$

Querying $(A_k^{-1})_{r,c}$ – $O(k)$ time

Computing α_k, u_k', v_k' – $O(nk)$ time

Queries and updates get more and more expensive!

Lazy updates (cont.)

$$A_k^{-1} = A_0^{-1} + \sum_{i=1}^{k} \alpha_i u_i' v_i'$$

Query time – $O(k)$

Update time – $O(nk)$

Compute A_k^{-1} explicitly after each K updates

Time required – $O(M(n,K,n))$ time

Amortized update time – $O(nK + M(n,K,n)/K)$

Update time minimized when $K \approx n^{0.575}$

Can be made **worst-case**

Even Lazier updates

$$A_k^{-1} = A_0^{-1} + \sum_{i=1}^{k} \alpha_i u_i' v_i'$$

After ℓ updates in positions

$$(r_1, c_1), (r_2, c_2), \ldots, (r_\ell, c_\ell)$$

maintain:

$$\alpha_i, (u_i')_{c_j}, (v_i')_{r_j}, \text{ for } 1 \leq i, j \leq \ell$$

Query time – $O(k^2)$

Update time – $O(k^2)$

After K, explicitly update A_k^{-1}

Dynamic transitive closure

- **Edge-Update**(e) – add/remove an edge e
- **Vertex-Update**(v) – add/remove edges touching v.
- **Query**(u,v) – is there are directed path from u to v?

[Sankowski '04]

Edge-Update	n^2	$n^{1.575}$	$n^{1.495}$
Vertex-Update	n^2	–	–
Query	1	$n^{0.575}$	$n^{1.495}$

(improving [Demetrescu-Italiano '00], [Roditty '03])

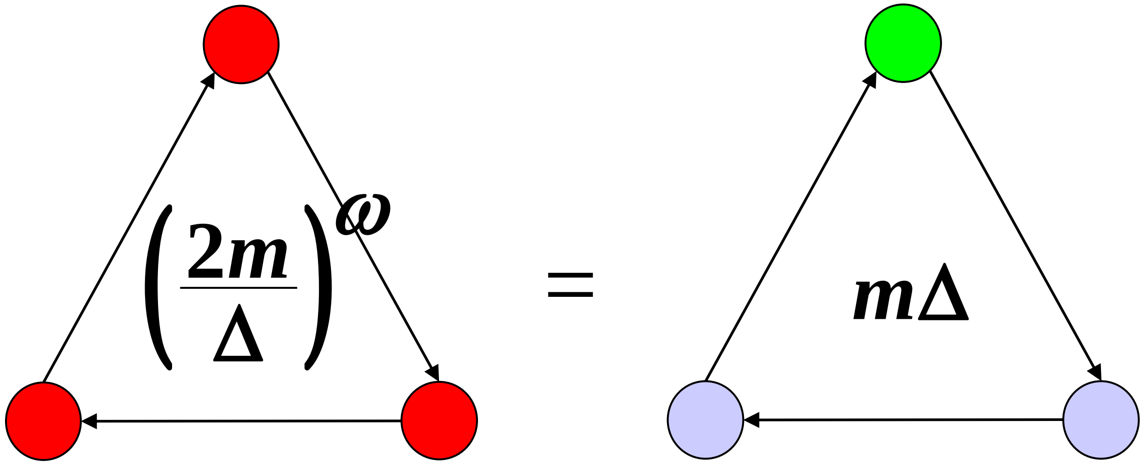
Finding triangles in $O(m^{2\omega/(\omega+1)})$ time
[Alon-Yuster-Z (1997)]

Let Δ be a parameter. $\Delta = m^{(\omega-1)/(\omega+1)}$

High degree vertices: vertices of degree $\geq \Delta$.

Low degree vertices: vertices of degree $< \Delta$.

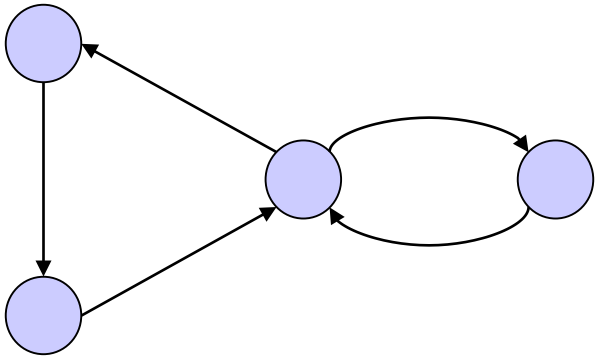
There are at most $2m/\Delta$ high degree vertices



$$\left(\frac{2m}{\Delta}\right)^{\omega} = m\Delta$$

Finding longer simple cycles

A graph G contains a C_k iff $\mathrm{Tr}(A^k) \neq 0$?



We want simple cycles!

Color coding [AYZ ’95]

Assign each vertex v a random number $c(v)$ from $\{0,1,\ldots,k-1\}$.

Remove all edges (u,v) for which $c(v)\neq c(u)+1 \pmod k$.

All cycles of length k in the graph are now simple.

If a graph contains a C_k then with a probability of at least k^{-k} it still contains a C_k after this process.

An improved version works with probability $2^{-O(k)}$.

Can be derandomized at a logarithmic cost.