Algebraic Graph Algorithms
Part II - Transitive Closure

Piotr Sankowski

University of Warsaw

2nd PCC October 17-23, 2008

Outline

- Dynamic Algebraic Functions

Outline

- Dynamic Algebraic Functions
- Dynamic Matrix Inverse for Column Updates

Outline

- Dynamic Algebraic Functions
- Dynamic Matrix Inverse for Column Updates
- Dynamic Matrix Inverse for Element Updates

Outline

- Dynamic Algebraic Functions
- Dynamic Matrix Inverse for Column Updates
- Dynamic Matrix Inverse for Element Updates
- Dynamic Transitive Closure

Outline

- Dynamic Algebraic Functions
- Dynamic Matrix Inverse for Column Updates
- Dynamic Matrix Inverse for Element Updates
- Dynamic Transitive Closure
- Dynamic Matrix Inverse over Rings

Outline

- Dynamic Algebraic Functions
- Dynamic Matrix Inverse for Column Updates
- Dynamic Matrix Inverse for Element Updates
- Dynamic Transitive Closure
- Dynamic Matrix Inverse over Rings
- Dynamic Shortest Distances

Dynamic Algebraic Functions

Let $\mathcal{F} = (S, +, \cdot, 0, 1)$ be a field.

Let $f : S^n \rightarrow S^m$ be a function over $\mathcal{F}$.

Dynamic Algebraic Functions

Let $\mathcal{F} = (S, +, \cdot, 0, 1)$ be a field.

Let $f : S^n \rightarrow S^m$ be a function over $\mathcal{F}$.

A dynamic algebraic algorithm is able to process the following requests:

Dynamic Algebraic Functions

Let $\mathcal{F} = (S, +, \cdot, 0, 1)$ be a field.

Let $f : S^n \rightarrow S^m$ be a function over $\mathcal{F}$.

A dynamic algebraic algorithm is able to process the following requests:

- **initialize($x_1, \ldots, x_n$):** set the input vector to $(x_1, \ldots, x_n)$,

Dynamic Algebraic Functions

Let $\mathcal{F} = (S, +, \cdot, 0, 1)$ be a field.

Let $f : S^n \rightarrow S^m$ be a function over $\mathcal{F}$.

A dynamic algebraic algorithm is able to process the following requests:

- **initialize($x_1, \ldots, x_n$):** set the input vector to $(x_1, \ldots, x_n)$,
- **update(k,x'_k):** set the input k to x'_k,

Dynamic Algebraic Functions

Let $\mathcal{F} = (S, +, \cdot, 0, 1)$ be a field.

Let $f : S^n \rightarrow S^m$ be a function over $\mathcal{F}$.

A dynamic algebraic algorithm is able to process the following requests:

- **initialize($x_1, \ldots, x_n$):** set the input vector to $(x_1, \ldots, x_n)$,
- **update(k,x'_k):** set the input k to x'_k,
- **query(k):** return the value of the output k.

Dynamic Matrix Functions

We consider the following problems:

- **determinant:** input A, output $\det(A)$,

Dynamic Matrix Functions

We consider the following problems:

- **determinant:** input A, output $\det(A)$,
- **adjoint:** input A, output $\mathrm{adj}(A)$,

Dynamic Matrix Functions

We consider the following problems:

- **determinant:** input A, output $\det(A)$,
- **adjoint:** input A, output $\mathrm{adj}(A)$, where $\mathrm{adj}(A)_{ij} = (-1)^{i+j} \det(A^{ji})$ and A^{ji} is the $(n-1) \times (n-1)$ matrix obtained from A by deleting j'th row and i'th column,

Dynamic Matrix Functions

We consider the following problems:

- **determinant:** input A, output $\det(A)$,
- **adjoint:** input A, output $\mathrm{adj}(A)$,
- **inverse:** input A, output A^{-1},

Dynamic Matrix Functions

We consider the following problems:

- **determinant:** input A, output $\det(A)$,
- **adjoint:** input A, output $\mathrm{adj}(A)$,
- **inverse:** input A, output A^{-1},
- **linear system of equations:** input A and b, output $A^{-1}b$.

Dynamic Matrix Functions

We consider the following problems:

- **determinant:** input A, output $\det(A)$,
- **adjoint:** input A, output $\mathrm{adj}(A)$,
- **inverse:** input A, output A^{-1},
- **linear system of equations:** input A and b, output $A^{-1}b$.

Lower bounds — $\Omega(n)$ (Frandsen, Hansen and Miltersen STACS'99).

Fast Matrix Multiplication

Let $\omega(\epsilon)$ be the exponent of multiplication of an $n \times n$ matrix by an $n \times n^{\epsilon}$ matrix.

For $\epsilon = 1$ we get the square matrix multiplication exponent, known to be $\omega(1) < 2.376$ (Coppersmith, Winograd '90).

For $\epsilon < 0.294$ we have $\omega(\epsilon) = 2$.

Matrix inverse can be computed in the matrix multiplication time.

Dynamic Matrix Inverse

Let us assume that during the updates the matrix remains nonsingular.

We want to change the column i'th to v. The new matrix is given by:

$$A' = A + (v - (A)_i)e_i^T,$$

where $(A)_i$ is the i'th column of A, and e_i is the basis vector.

Dynamic Matrix Inverse

We will compute a matrix B such, that:

$$A' = A \cdot B.$$

Then matrix A'^{-1} can be computed with:

$$A'^{-1} = B^{-1} A^{-1}.$$

Dynamic Matrix Inverse

We substitute $B := I + be_i^T$ into $A' = A \cdot B$:

Dynamic Matrix Inverse

We substitute $B := I + be_i^T$ into $A' = A \cdot B$:

$$A + (v - (A)_i)e_i^T = A \cdot (I + be_i^T),$$

Dynamic Matrix Inverse

We substitute $B := I + be_i^T$ into $A' = A \cdot B$:

$$A + (v - (A)_i)e_i^T = A \cdot (I + be_i^T),$$

$$(v - (A)_i)e_i^T = A \cdot be_i^T.$$

Dynamic Matrix Inverse

We substitute $B := I + be_i^T$ into $A' = A \cdot B$:

$$A + (v - (A)_i)e_i^T = A \cdot (I + be_i^T),$$

$$(v - (A)_i)e_i^T = A \cdot be_i^T.$$

$$v - (A)_i = Ab,$$

Dynamic Matrix Inverse

We substitute $B := I + be_i^T$ into $A' = A \cdot B$:

$$A + (v - (A)_i)e_i^T = A \cdot (I + be_i^T),$$

$$(v - (A)_i)e_i^T = A \cdot be_i^T.$$

$$v - (A)_i = Ab,$$

$$b = A^{-1}(v - (A)_i).$$

Dynamic Matrix Inverse

The inverse of B:

$$B^{-1} = (I + be_i^T)^{-1} = \begin{bmatrix} 1 & \dots & 0 & -\frac{b_1}{1+b_i} & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ 0 & \dots & 1 & -\frac{b_{i-1}}{1+b_i} & 0 & \dots & 0 \\ 0 & \dots & 0 & (1+b_i)^{-1} & 0 & \dots & 0 \\ 0 & \dots & 0 & -\frac{b_{i+1}}{1+b_i} & 1 & \dots & 0 \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & -\frac{b_n}{1+b_i} & 0 & \dots & 1 \end{bmatrix}.$$

$A'^{-1} = B^{-1}A^{-1}$ can be computed in $O(n^2)$ arithmetic operations.

Dynamic Matrix Inverse

Theorem 1 *The problem of dynamic matrix inverse, with non-singular column updates, can be solved with the following costs:*

- **initialization** $O(n^\omega)$ *arithmetic operations,*
- **update** $O(n^2)$ *arithmetic operations (worst-case),*
- **query** $O(1)$ *arithmetic operations (worst-case).*

Dynamic Matrix Inverse

Theorem 1 *The problem of dynamic matrix inverse, with non-singular column updates, can be solved with the following costs:*

- **initialization** $O(n^\omega)$ *arithmetic operations,*
- **update** $O(n^2)$ *arithmetic operations (worst-case),*
- **query** $O(1)$ *arithmetic operations (worst-case).*

After an update $O(n^2)$ entries of the inverse can change.

Matrix Inverse: Element Updates

Theorem 2 *The problem of dynamic matrix inverse, with non-singular element updates can be solved with the following costs:*

- **initialization** $O(n^{\omega})$ *arithmetic operations,*
- **update** $O(n^{1.575})$ *arithmetic operations (worst-case),*
- **query** $O(n^{0.575})$ *arithmetic operations (worst-case).*

Matrix Inverse: Element Updates

We maintain the matrix A in a lazy form $A = NM$.

At the beginning we set $M := A$ and $N := I$.

After each update we recompute only N^{-1}:

$$N^{-1} := N^{-1}B^{-1},$$

while M^{-1} is recomputed from time to time:

$$M^{-1} := M^{-1}N^{-1} \qquad N^{-1} := I.$$

Matrix Inverse: Element Updates

We have:

$$N^{-1} := N^{-1}B^{-1} = N^{-1}(B^{-1} - I) + N^{-1},$$

the matrix $B^{-1} - I$ has only one non-zero column.

After each update the number of columns in $N^{-1} - I$ increases by one — it's k after k updates.

Hence, the recomputation of N^{-1} takes $O(n^{1+k})$ time after k updates.

Matrix Inverse: Element Updates

We have:

$$M^{-1} := M^{-1}N^{-1} = M^{-1}(N^{-1} - I) + M^{-1}.$$

N^{-1} has k non-zero columns after $k = n^{\epsilon}$ updates.

We can multiply only non-zero columns using fast rectangular matrix multiplication in $O(n^{\omega(\epsilon)})$ time.

For an updates the amortized time is:

$$O(n^{1+\epsilon} + n^{\omega(\epsilon)-\epsilon}).$$

Matrix Inverse: Element Updates II

Theorem 3 *The problems of dynamic matrix inverse, with non-singular element updates can be solved with the following costs:*

- **initialization** $O(n^{\omega})$ *arithmetic operations,*
- **update** $O(n^{1.495})$ *arithmetic operations (worst-case),*
- **query** $O(n^{1.495})$ *arithmetic operations (worst-case).*

Dynamic Transitive Closure

	Update	**Query**
Henzinger and King '95	$\tilde{O}(nm^{0.58})$	$\Theta(n/\log n)$
King and Sagert '99	$O(n^{2.26})$	$O(1)$
King '99	$O(n^2 \log n)$	$O(1)$
Demetrescu and Italiano '00	$O(n^2)$	$O(1)$
Roditty and Zwick '02	$O(m\sqrt{n})$	$O(\sqrt{n})$
Roditty and Zwick '04	$O(m + n \log n)$	$O(n)$
Sankowski '04 (worst-case)	$O(n^2)$	$O(1)$
Sankowski '04 (worst-case)	$O(n^{1.575})$	$O(n^{0.575})$
Sankowski '04 (worst-case)	$O(n^{1.495})$	$O(n^{1.495})$

Counting Paths in DAG

Let A be a DAG adjacency matrix, then the number of paths from one vertex to another is given by a matrix:

$$A^* = \sum_{i=0}^{n-1} A^i \ .$$

Counting Paths in DAG

Let A be a DAG adjacency matrix, then the number of paths from one vertex to another is given by a matrix:

$$A^* = \sum_{i=0}^{n-1} A^i = (I - A)^{-1} ,$$

because $A^n = 0$.

Counting Paths in DAG

Let A be a DAG adjacency matrix, then the number of paths from one vertex to another is given by a matrix:

$$A^* = \sum_{i=0}^{n-1} A^i = (I - A)^{-1},$$

because $A^n = 0$.

In order to count paths in dynamic DAG we can maintain the inverse of the matrix $I - A$.

Dynamic Transitive Closure

Can this approach be used in general case?

Dynamic Transitive Closure

Can this approach be used in general case?

Yes, but instead of matrix inverse we have to use matrix adjoint.

$$A^{-1} = \frac{\mathrm{adj}(A)}{\det(A)}.$$

Dynamic Transitive Closure

Theorem 4 *Let A be the adjacency matrix of a graph G. Substitute distinct variables for non-zero elements of A, than:*

Dynamic Transitive Closure

Theorem 4 *Let A be the adjacency matrix of a graph G. Substitute distinct variables for non-zero elements of A, than:*

- *There exists a path in G from i to j, iff* $\mathrm{adj}(I - A)_{ij}$ *not equal to zero.*

Dynamic Transitive Closure

Theorem 4 *Let A be the adjacency matrix of a graph G. Substitute distinct variables for non-zero elements of A, than:*

- *There exists a path in G from i to j, iff* $\mathrm{adj}(I - A)_{ij}$ *not equal to zero.*
- $\mathrm{adj}(I - A)_{ij}$ *is non-zero over finite field* $\mathcal{Z}_p$.

Dynamic Transitive Closure

Theorem 4 *Let A be the adjacency matrix of a graph G. Substitute distinct variables for non-zero elements of A, than:*

- *There exists a path in G from i to j, iff* $\mathrm{adj}(I - A)_{ij}$ *not equal to zero.*
- $\mathrm{adj}(I - A)_{ij}$ *is non-zero over finite field* $\mathcal{Z}_p$.

For random substitution a non-zero polynomial has a non-zero value with high probability.

Reduction

$$\det(X) =$$

$$= \sum_{p \in \Pi_n} \operatorname{sgn}(p) \prod_{i=0}^{n} x_{i,p_i}$$

The permutation p is a cycle cover of a graph.

Reduction

$$\det(X) =$$

$$= \sum_{p \in \Pi_n} \operatorname{sgn}(p) \prod_{i=0}^{n} x_{i,p_i}$$

The permutation p is a cycle cover of a graph.

$$\operatorname{adj}(I - A)_{ij} =$$

$$(-1)^{i+j} \det((I - A)^{ji})$$

Reduction

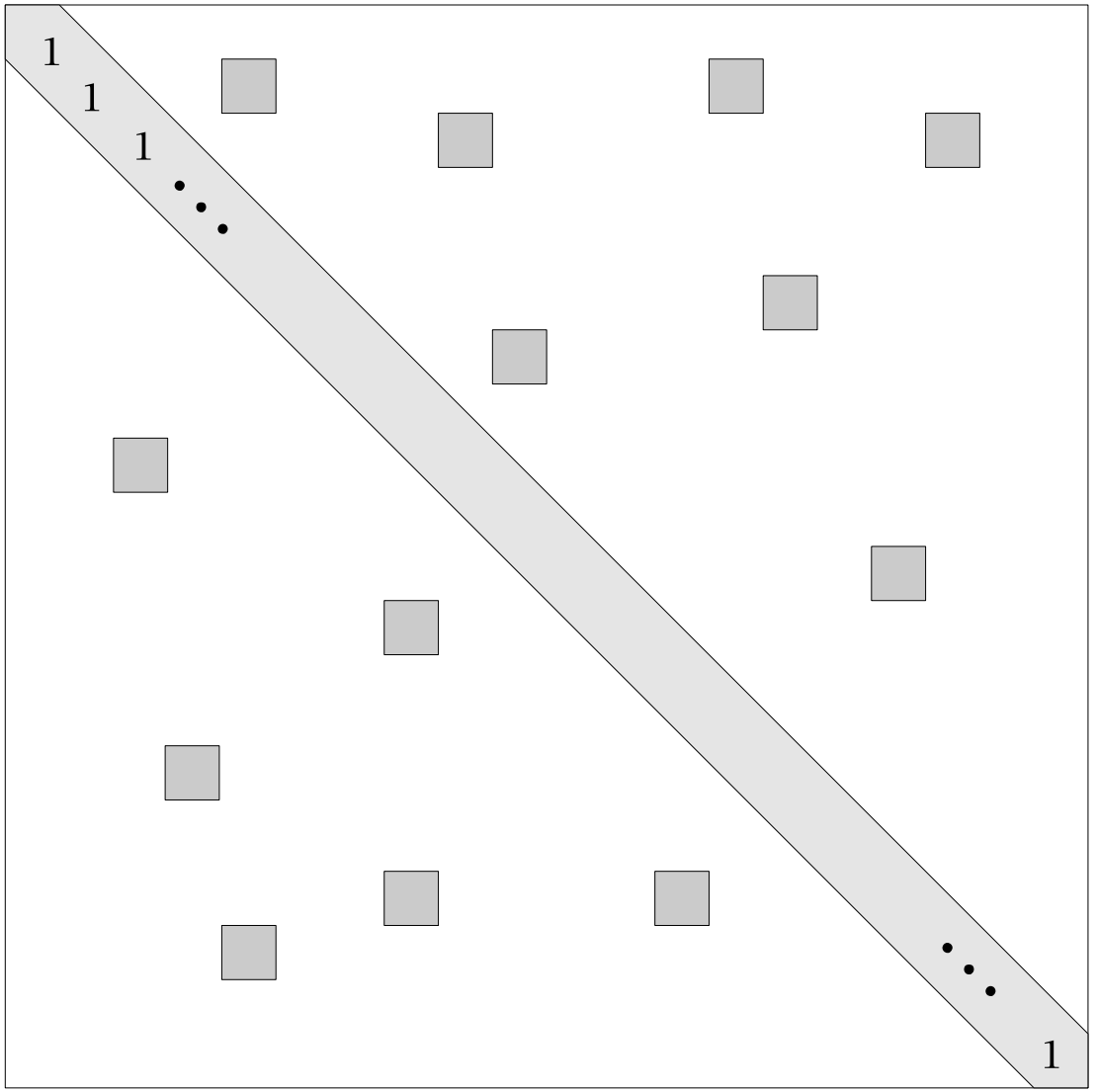
$$\det(X) =$$

$$= \sum_{p \in \Pi_n} \operatorname{sgn}(p) \prod_{i=0}^{n} x_{i,p_i}$$

The permutation p is a cycle cover of a graph.

$$\operatorname{adj}(I - A)_{ij} =$$

$$(-1)^{i+j} \det((I - A)^{ji})$$



Reduction

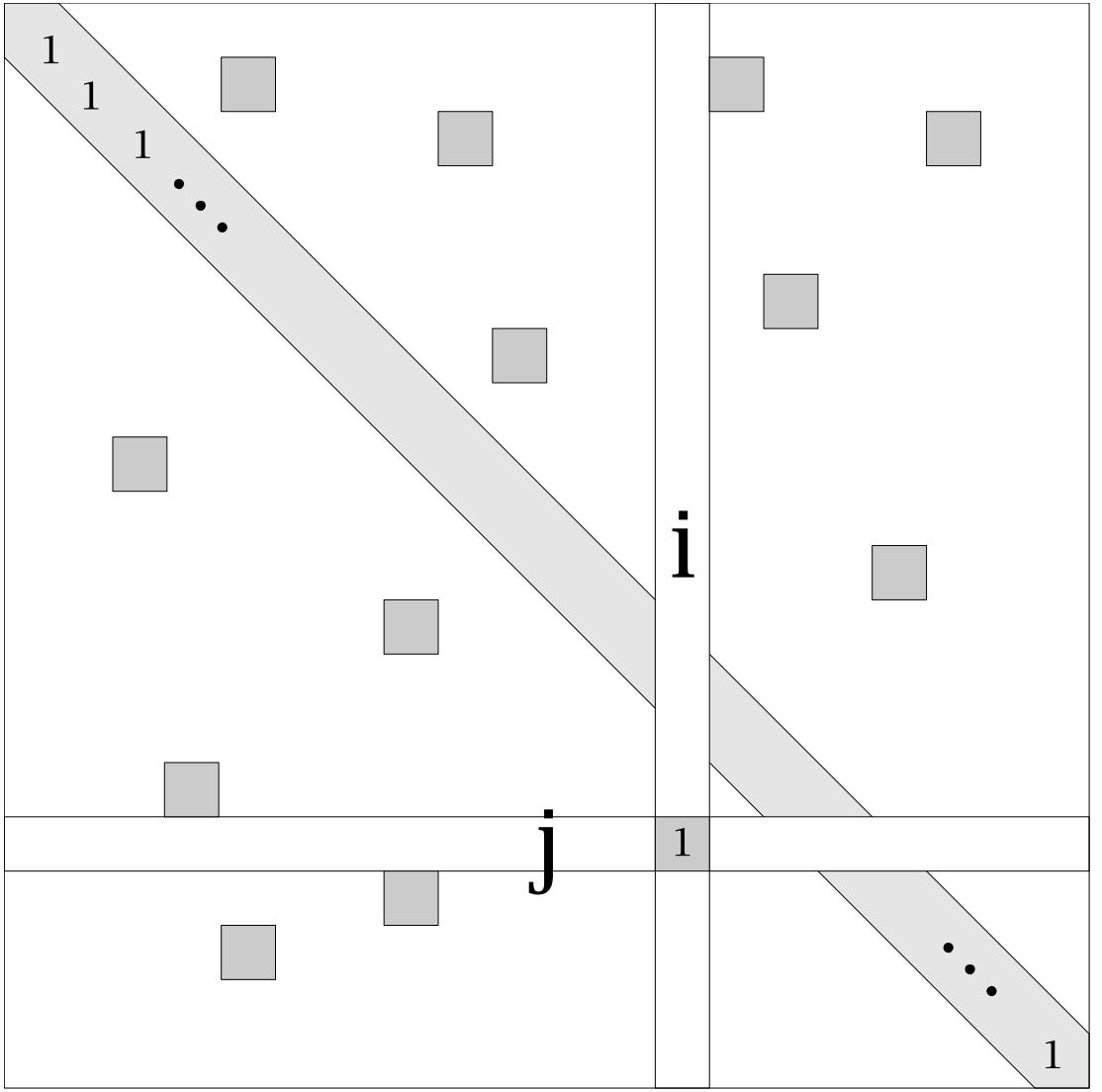
$$\det(X) =$$

$$= \sum_{p \in \Pi_n} \operatorname{sgn}(p) \prod_{i=0}^{n} x_{i,p_i}$$

The permutation p is a cycle cover of a graph.

$$\operatorname{adj}(I - A)_{ij} =$$

$$(-1)^{i+j} \det((I - A)^{ji})$$



Reduction

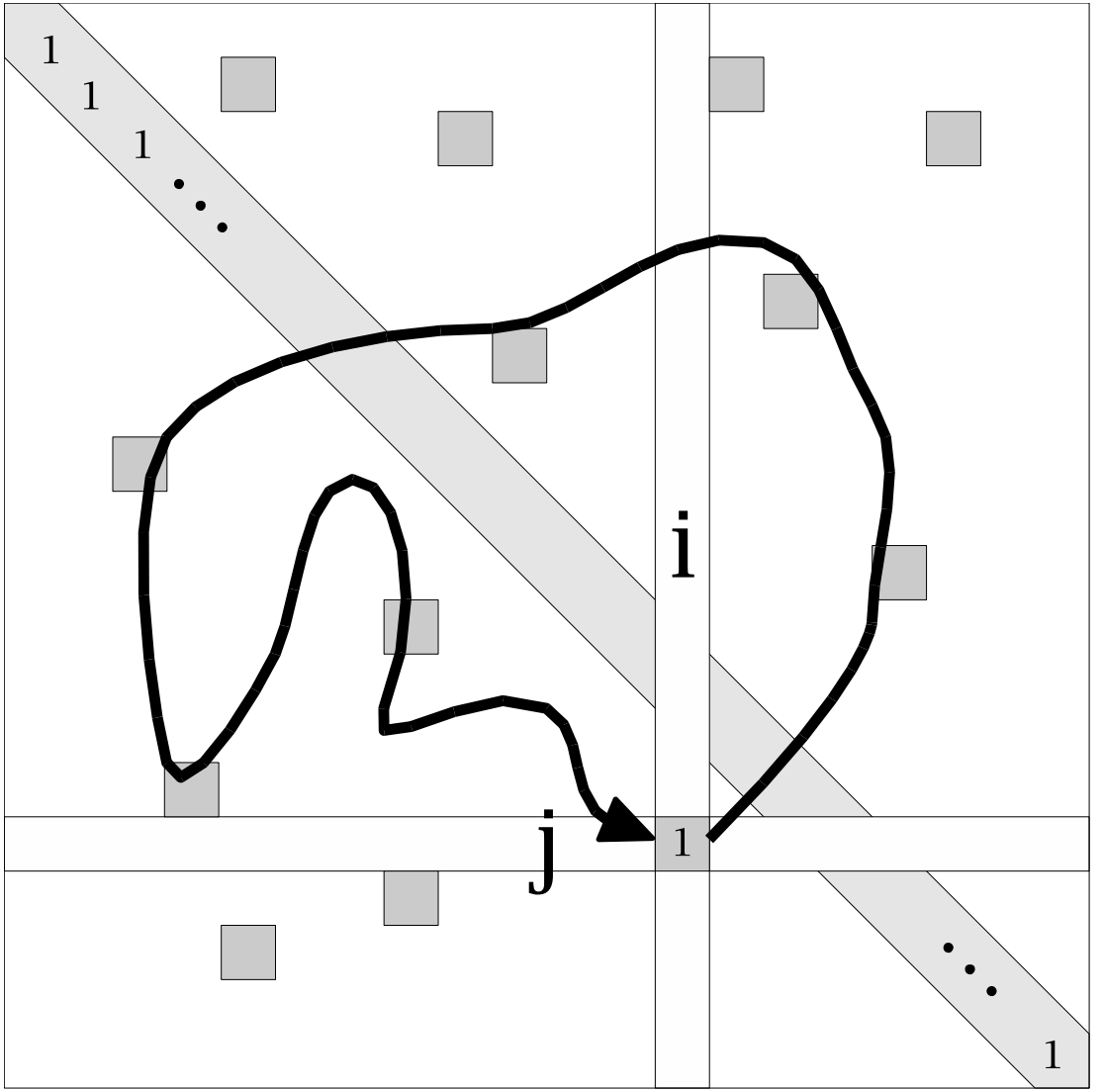
$$\det(X) =$$

$$= \sum_{p \in \Pi_n} \operatorname{sgn}(p) \prod_{i=0}^{n} x_{i,p_i}$$

The permutation p is a cycle cover of a graph.

$$\operatorname{adj}(I - A)_{ij} =$$

$$(-1)^{i+j} \det((I - A)^{ji})$$



Reduction

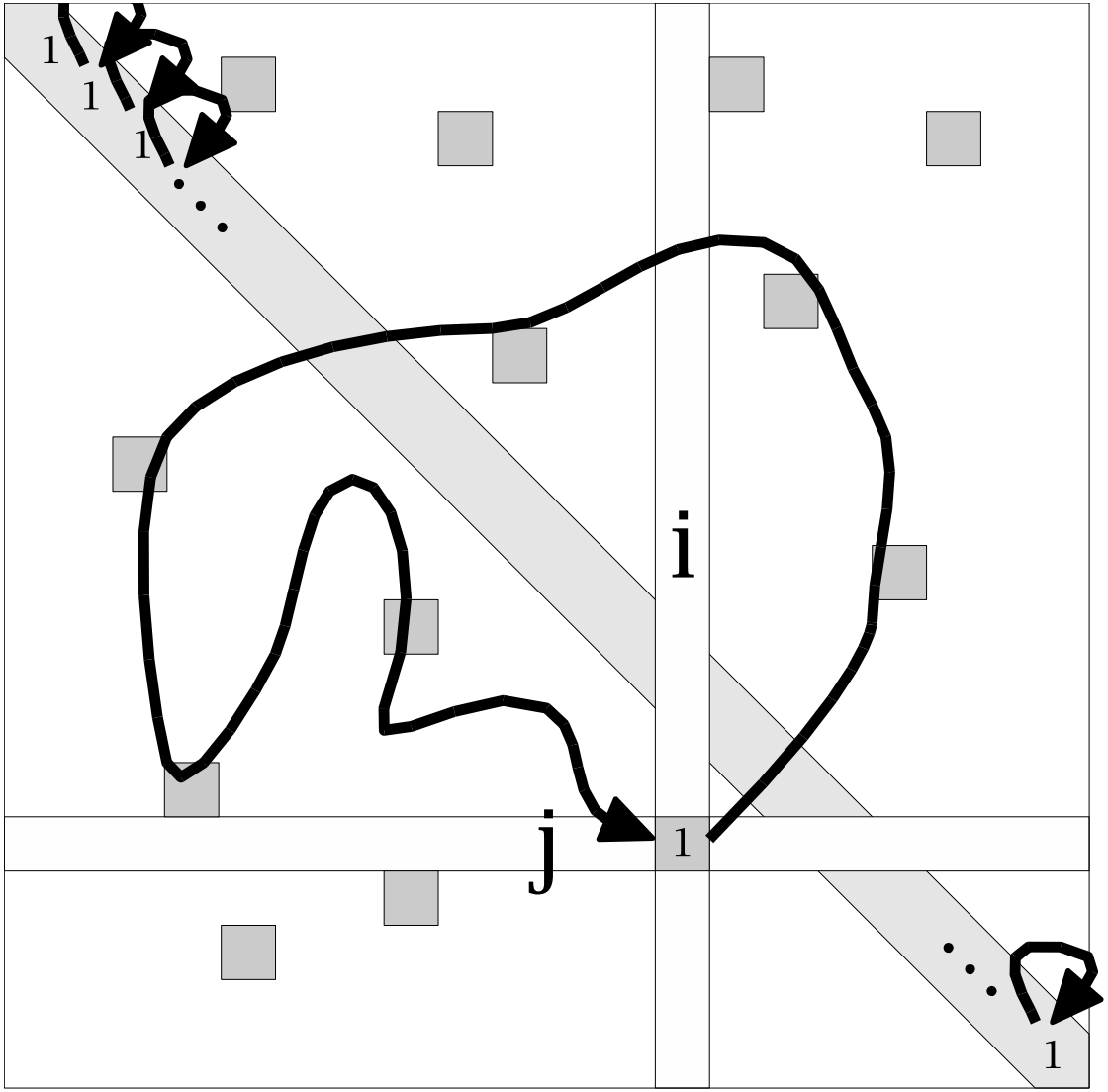
$$\det(X) =$$

$$= \sum_{p \in \Pi_n} \operatorname{sgn}(p) \prod_{i=0}^{n} x_{i,p_i}$$

The permutation p is a cycle cover of a graph.

$$\operatorname{adj}(I - A)_{ij} =$$

$$(-1)^{i+j} \det((I - A)^{ji})$$



Dynamic Transitive Closure

Theorem 5

Dynamic Matrix Inverse

Update $O(n^{\alpha})$ operations

Query $O(n^{\beta})$ operations

one may assume non-singularity

Dynamic Transitive Closure

Update $O(n^{\alpha})$ time

Query $O(n^{\beta})$ time

randomized with one-sided error

Matrix Determinant over Rings

Gaussian elimination cannot be performed without divisions.

Matrix Determinant over Rings

Gaussian elimination cannot be performed without divisions.

The determinant of a matrix can be computed without divisions in $\tilde{O}(n^{\omega+1})$ ring operations *(Strassen '73)*.

Strassen's Idea

The idea is to work in $\mathcal{R}[[u]]$ — the ring of formal power series over $\mathcal{R}$.

Strassen's Idea

The idea is to work in $\mathcal{R}[[u]]$ — the ring of formal power series over $\mathcal{R}$.

Let A be the matrix over the ring R, we define

$$A(u) = I + u(A - I)$$

Strassen's Idea

The idea is to work in $\mathcal{R}[[u]]$ — the ring of formal power series over $\mathcal{R}$.

Let A be the matrix over the ring R, we define

$$A(u) = I + u(A - I)$$

We have

$$A = A(u)|_{u=1},$$
$$\det(A) = \det(A(u))|_{u=1},$$
$$\mathrm{adj}(A) = \mathrm{adj}(A(u))|_{u=1}.$$

Strassen's Idea

The elements of the form $1 - z$, where $z \in u\mathcal{R}[[u]]$ are invertible:

$$\frac{1}{1-z} = 1 + z + z^2 + \ldots =$$

$$= (1+z)(1+z^2)(1+z^4)\ldots.$$

It is possible to compute this quantity without inverting elements of $\mathcal{R}$.

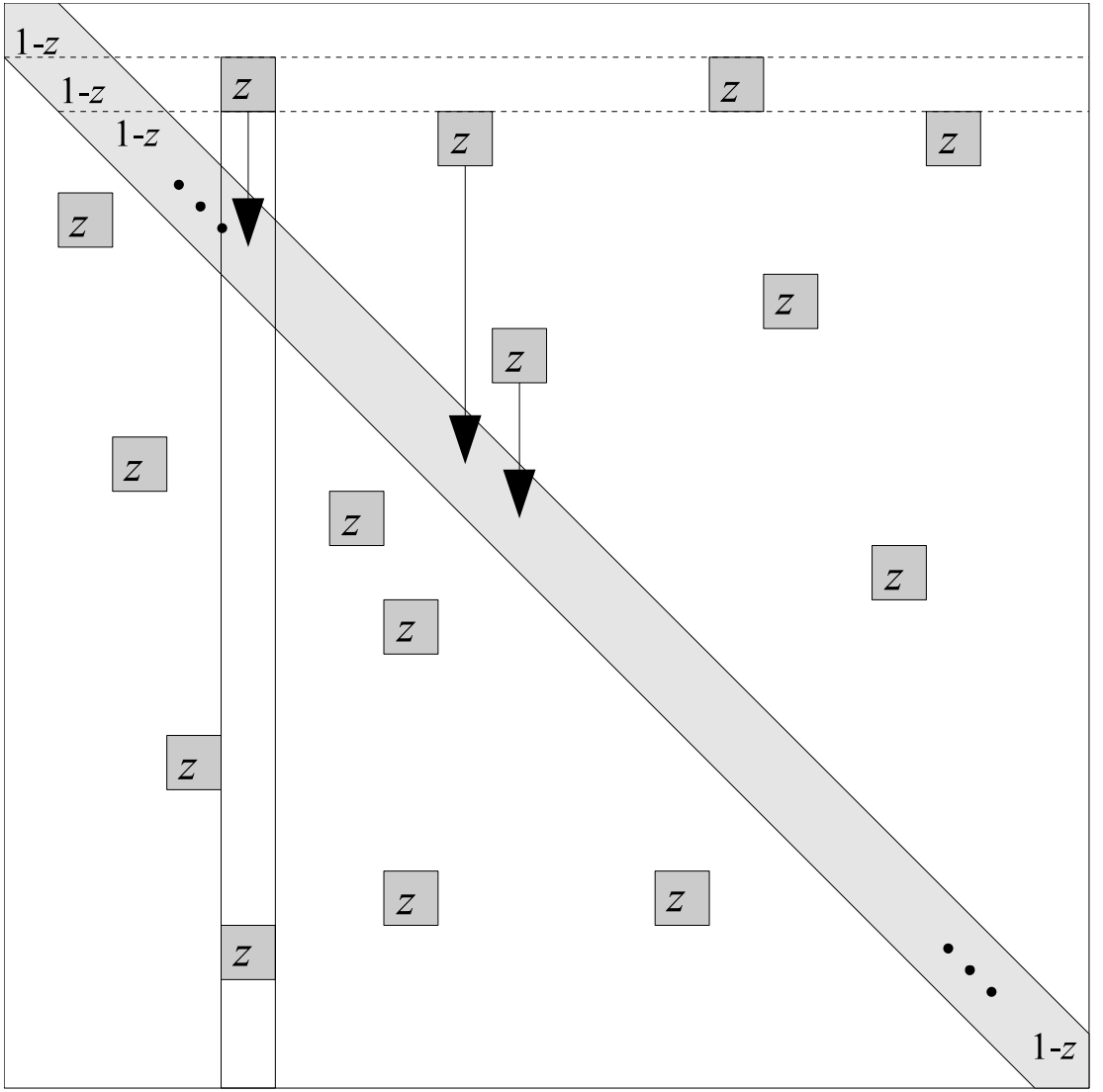
Strassen's Idea

Determinant of

$$A(u) = I + u(A - I)$$

can be computed with Gaussian Elimination.

The elements on the diagonal are of the form $1 - z$.



Strassen's Idea

The result of the evaluation $\det(A(u))$ is a polynomial of degree n in u, so the computations can be carried modulo u^{n+1}.

Strassen's Idea

The result of the evaluation $\det(A(u))$ is a polynomial of degree n in u, so the computations can be carried modulo u^{n+1}.

The elements of $R[[u]]$ can be multiplied with use of $O(n \log(n) \log(\log(n)))$ operations *(Strassen '71)*.

Strassen's Idea

The result of the evaluation $\det(A(u))$ is a polynomial of degree n in u, so the computations can be carried modulo u^{n+1}.

The elements of $R[[u]]$ can be multiplied with use of $O(n \log(n) \log(\log(n)))$ operations *(Strassen '71)*.

We obtain a complexity of $\tilde{O}(n^{\omega+1})$ for computing the determinant without divisions.

Dynamic Matrix Inverse over Rings

The matrix $A(u)$ is non-singular for all A

$$A(u)^{-1} = \frac{1}{I + u(A - i)} = \sum_{i=0}^{\infty} (-u(A - I))^i .$$

Dynamic Matrix Inverse over Rings

The matrix $A(u)$ is non-singular for all A

$$A(u)^{-1} = \frac{1}{I + u(A - i)} = \sum_{i=0}^{\infty} (-u(A - I))^i .$$

We can dynamically maintain the inverse of $A(u)$.

Dynamic Matrix Inverse

We want to change the column i'th to v. The new matrix is given by:

$$A' = A + (v - (A)_i)e_i^T,$$

and

$$A'(u) = I + u\left(A + (v - (A)_i)e_i^T - I\right).$$

Dynamic Matrix Inverse over Rings

We compute matrix B such that

$$A'(u) = A(u) \cdot B.$$

Dynamic Matrix Inverse over Rings

We compute matrix B such that

$$A'(u) = A(u) \cdot B.$$

Let as substitute $B = I + be_i^T$, than:

$$I + u\left(A + (v - (A)_i)e_i^T - I\right) = A(u) \cdot (I + be_i^T),$$

Dynamic Matrix Inverse over Rings

We compute matrix B such that

$$A'(u) = A(u) \cdot B.$$

Let as substitute $B = I + be_i^T$, than:

$$I + u\left(A + (v - (A)_i)e_i^T - I\right) = A(u) \cdot (I + be_i^T),$$

$$A(u) + u(v - (A)_i)e_i^T = A(u) \cdot (I + be_i^T),$$

Dynamic Matrix Inverse over Rings

$$A(u) + u(v - (A)_i)e_i^T = A(u) \cdot (I + be_i^T),$$

Dynamic Matrix Inverse over Rings

$$A(u) + u(v - (A)_i)e_i^T = A(u) \cdot (I + be_i^T),$$

$$u(v - (A)_i)e_i^T = A(u) \cdot be_i^T.$$

Dynamic Matrix Inverse over Rings

$$A(u) + u(v - (A)_i)e_i^T = A(u) \cdot (I + be_i^T),$$

$$u(v - (A)_i)e_i^T = A(u) \cdot be_i^T.$$

$$u(v - (A)_i) = A(u) \cdot b.$$

Dynamic Matrix Inverse over Rings

$$A(u) + u(v - (A)_i)e_i^T = A(u) \cdot (I + be_i^T),$$

$$u(v - (A)_i)e_i^T = A(u) \cdot be_i^T.$$

$$u(v - (A)_i) = A(u) \cdot b.$$

$$b = A(u)^{-1}u(v - (A)_i),$$

Dynamic Matrix Inverse over Rings

$$b = A(u)^{-1}u(v - (A)_i),$$

and:

$$B = I + uA(u)^{-1}(v - (A)_i).$$

Dynamic Matrix Inverse over Rings

$$b = A(u)^{-1}u(v - (A)_i),$$

and:

$$B = I + uA(u)^{-1}(v - (A)_i).$$

We can show that the matrix B is invertible.

$$\begin{aligned} B^{-1} &= \sum_{k=0}^{\infty} \left(-ube_i^T\right)^k = \\ &= \sum_{k=0}^{\infty} \left(-uA(u)^{-1}(v - (A)_i)e_i^T\right)^k. \end{aligned}$$

Dynamic Matrix Inverse over Rings

The vector b can be computed in $O(n^2)$ operations in $\mathcal{R}[[u]]$.

The matrix B^{-1} in $O(n^2)$ operations.

The matrix B^{-1} has only $O(n)$ non-zero entries, so the multiplication $A'^{-1} = B^{-1}A^{-1}$ can be done with $O(n^2)$.

Dynamic Matrix Inverse over Rings

Theorem 6 *The problems of dynamic determinant and matrix adjoint over commutative ring R with row and column updates can be solved with the following costs:*

- **initialization** $\tilde{O}(n^{\omega+1})$ *ring operations,*
- **update** $\tilde{O}(n^3)$ *ring operations operations (worst-case),*
- **query** $O(1)$ *ring operations (worst-case).*

Dynamic Matrix Inverse over Rings

Notice, that $\det(A'(u)) = (1 + b_i)\det(A(u))$ and the determinant of $A(u)$ can be also updated.

Dynamic Matrix Inverse over Rings

Notice, that $\det(A'(u)) = (1 + b_i)\det(A(u))$ and the determinant of $A(u)$ can be also updated.

We have $\mathrm{adj}(A(u))_{ij} = \det(A(u))(A(u)^{-1})_{ij}$.

To answer queries in constant number of operations we recompute the matrix $\mathrm{adj}(A) = \mathrm{adj}(A(u))|_{u=1}$ after every update.

Dynamic Matrix Inverse over Rings

Notice, that $\det(A'(u)) = (1 + b_i)\det(A(u))$ and the determinant of $A(u)$ can be also updated.

We have $\mathrm{adj}(A(u))_{ij} = \det(A(u))(A(u)^{-1})_{ij}$.

To answer queries in constant number of operations we recompute the matrix $\mathrm{adj}(A) = \mathrm{adj}(A(u))|_{u=1}$ after every update.

This requires $\tilde{O}(n^3)$ ring operations.

Shortest Distances

	Update	**Query**
Henzinger et al. '97 Planar Grpahs	$O(n^{\frac{4}{3}} \log(nC))$	$O(n^{\frac{4}{3}} \log(nC))$
Fakcharoemphol and Rao '02 Planar Grpahs	$O(n^{\frac{4}{5}} \log^{\frac{13}{5}} n)$	$O(n^{\frac{4}{5}} \log^{\frac{13}{5}} n)$
King '99	$O(n^{2.5}\sqrt{C \log n})$	$O(1)$
Demetrescu and Italiano '00 Real Weights	$O(n^{2.5}\sqrt{S \log^3 n})$	$O(1)$
Demetrescu and Italiano '04	$\tilde{O}(n^2)$	$O(1)$
Sankowski '05	$O(n^{1.932})$	$O(n^{1.288})$

Shortest Distances

Theorem 7 *Let A be the adjacency matrix of a graph G. Substitute distinct variables for non-zero elements of A, then than:*

Shortest Distances

Theorem 7 *Let A be the adjacency matrix of a graph G. Substitute distinct variables for non-zero elements of A, then than:*

- *The length of the shortest path in G from i to j is equal to the degree of the smallest degree term in u in* $\mathrm{adj}(I - Au)_{ij}$.

Shortest Distances

Theorem 7 *Let A be the adjacency matrix of a graph G. Substitute distinct variables for non-zero elements of A, then than:*

- *The length of the shortest path in G from i to j is equal to the degree of the smallest degree term in u in* $\mathrm{adj}(I - Au)_{ij}$.
- *Moreover all non-zero terms in* $\mathrm{adj}(I - uA)_{ij}$ *are also non-zero over finite field $\mathcal{Z}_p$.*

Reduction

$$\det(X) =$$

$$= \sum_{p \in \Pi_n} \operatorname{sgn}(p) \prod_{i=0}^{n} x_{i,p_i}$$

The permutation p is a cycle cover of a graph.

Reduction

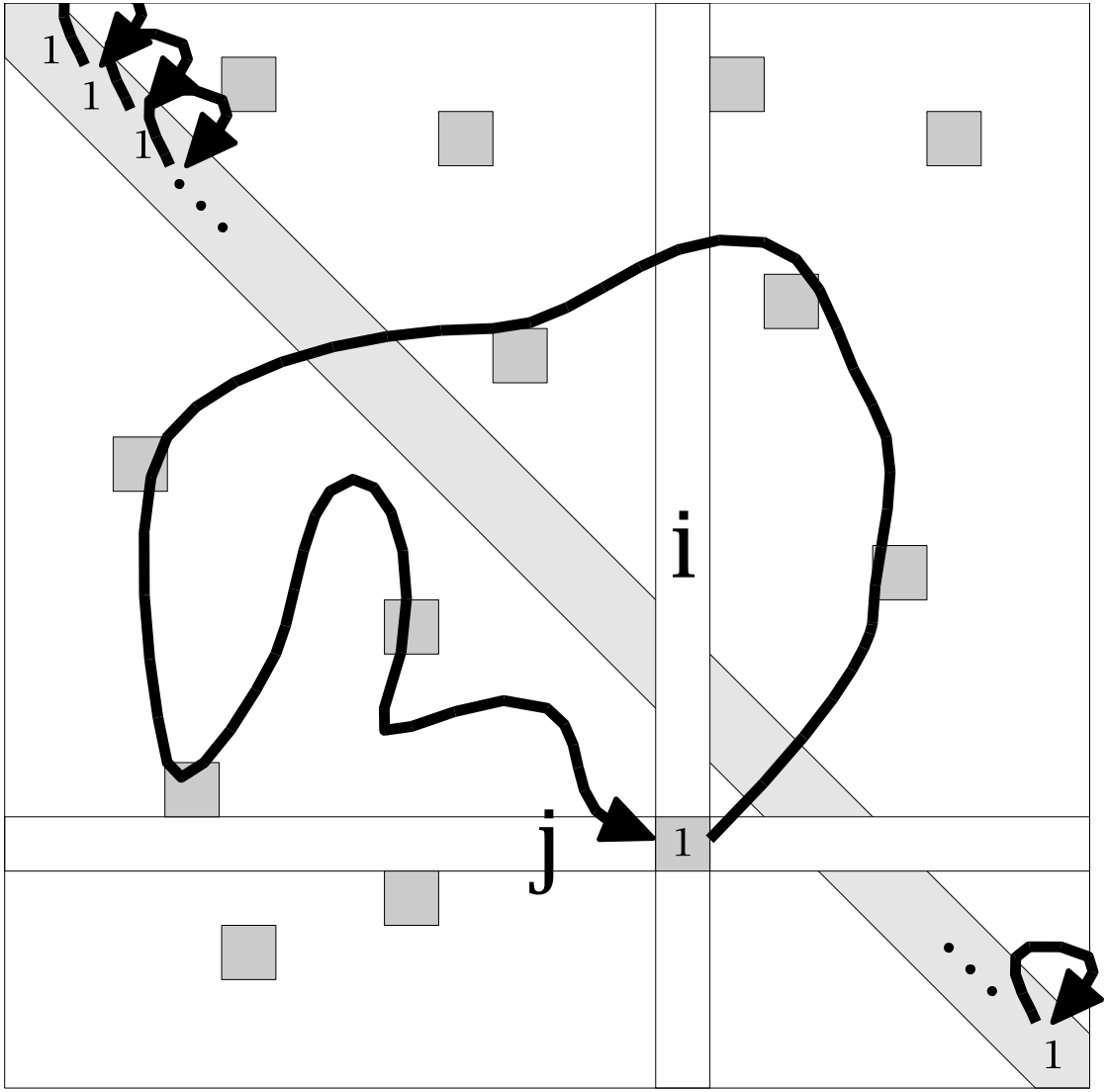
$$\det(X) =$$

$$= \sum_{p \in \Pi_n} \operatorname{sgn}(p) \prod_{i=0}^{n} x_{i,p_i}$$

The permutation p is a cycle cover of a graph.

$$\operatorname{adj}(I - A)_{ij} =$$

$$(-1)^{i+j} \det((I - A)^{ji})$$



Reduction

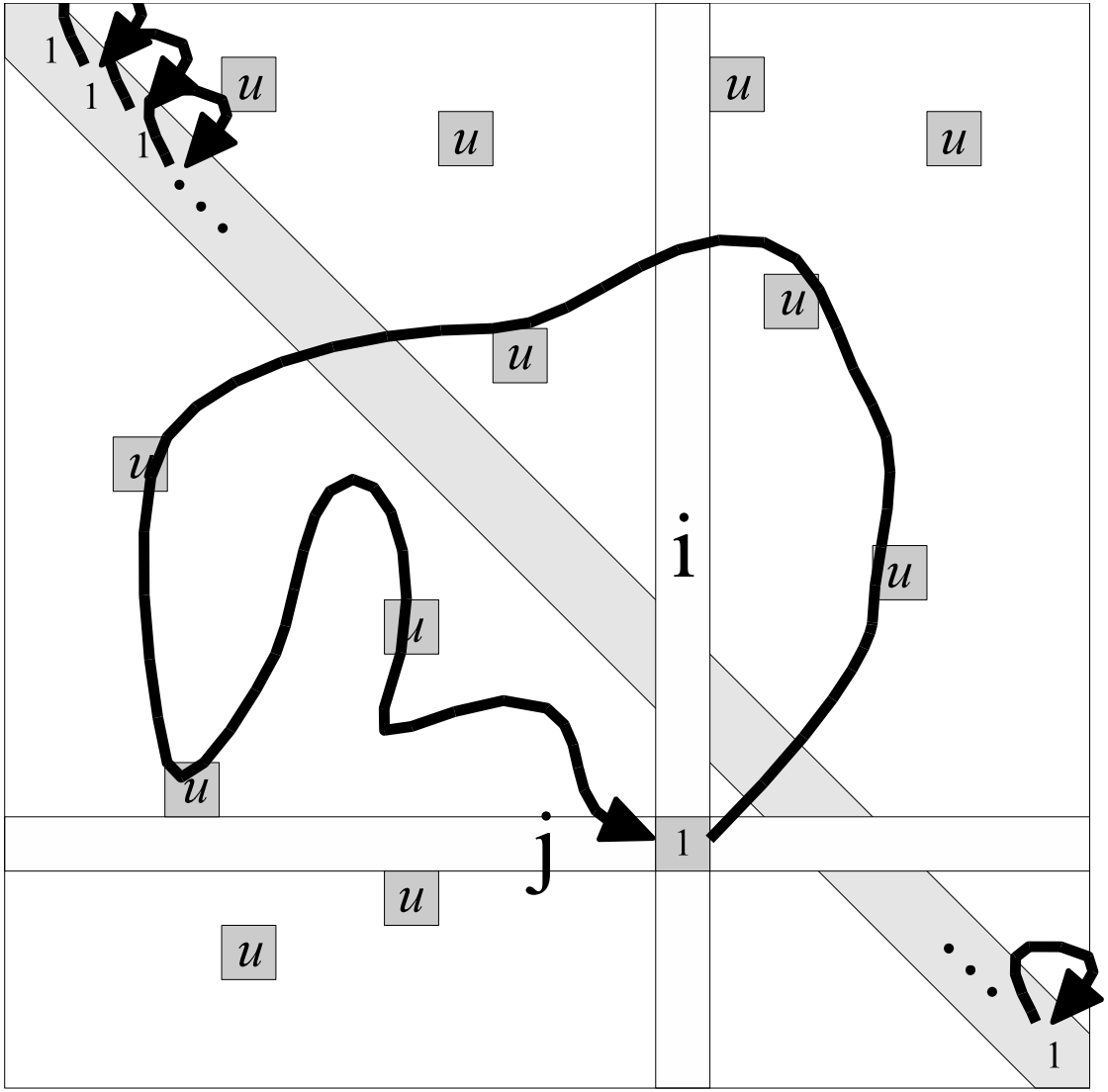
$$\det(X) =$$

$$= \sum_{p \in \Pi_n} \operatorname{sgn}(p) \prod_{i=0}^{n} x_{i,p_i}$$

The permutation p is a cycle cover of a graph.

$$\operatorname{adj}(I - uA)_{ij} =$$

$$(-1)^{i+j} \det((I - uA)^{ji})$$



Dynamic Shortest Distances

- Generate random adjacency matrix $\tilde{A}$ from the adjacency matrix A of G by substituting each nonzero entry in A with a random number in the range $1, \ldots, p-1$.

Dynamic Shortest Distances

- Generate random adjacency matrix $\tilde{A}$ from the adjacency matrix A of G by substituting each nonzero entry in A with a random number in the range $1, \ldots, p-1$.
- Now maintain dynamically the adjoint of the matrix $I - uB$ over polynomials of degree n,

Dynamic Shortest Distances

- Generate random adjacency matrix $\tilde{A}$ from the adjacency matrix A of G by substituting each nonzero entry in A with a random number in the range $1, \ldots, p-1$.
- Now maintain dynamically the adjoint of the matrix $I - uB$ over polynomials of degree n,
- Answer queries by finding the smallest degree term in $\mathrm{adj}(I - u\tilde{B})_{ij}$.

Dynamic Shortest Distances

- Generate random adjacency matrix $\tilde{A}$ from the adjacency matrix A of G by substituting each nonzero entry in A with a random number in the range $1, \ldots, p-1$.
- Now maintain dynamically the adjoint of the matrix $I - uB$ over polynomials of degree n,
- Answer queries by finding the smallest degree term in $\mathrm{adj}(I - u\tilde{B})_{ij}$.

In this way we get algorithm that supports updates in $O(n^{2.575})$ time and queries in $O(n^{0.575})$ time.

Dynamic Shortest Distances

Theorem 8 *There exist an algorithm for dynamically computing shortest distances up to k in unweighted graph supporting updates in $\tilde{O}(kn^{1.575})$ time and queries in $O(kn^{0.575})$ time.*

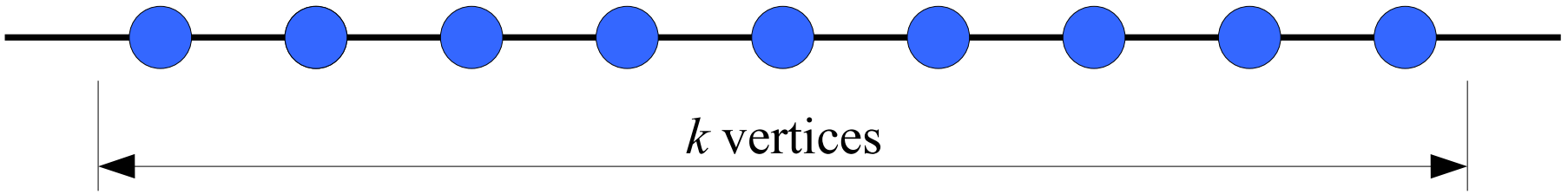
The algorithm is randomized and with small probability may return wrong higher values.

Path Decomposition

Theorem 9 (Ullman and Yannakakis '90) *Given a graph $G = (V, E)$, let $H \subseteq V$ be a set of vertices chosen uniformly at random. Then with high probability in a sequence of $k = \tilde{O}(\frac{n}{|H|})$ vertices on a given simple path there is at least one vertex from H.*

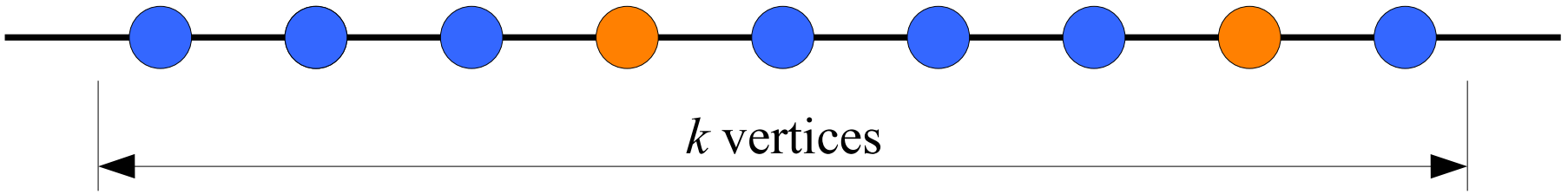
Path Decomposition

Theorem 9 (Ullman and Yannakakis '90) *Given a graph $G = (V, E)$, let $H \subseteq V$ be a set of vertices chosen uniformly at random. Then with high probability in a sequence of $k = \tilde{O}(\frac{n}{|H|})$ vertices on a given simple path there is at least one vertex from H.*



Path Decomposition

Theorem 9 (Ullman and Yannakakis '90) *Given a graph $G = (V, E)$, let $H \subseteq V$ be a set of vertices chosen uniformly at random. Then with high probability in a sequence of $k = \tilde{O}(\frac{n}{|H|})$ vertices on a given simple path there is at least one vertex from H.*



Path Decomposition

To compute distance from u to v we

Path Decomposition

To compute distance from u to v we

- generate a set $H \subset V$ of vertices uniformly at random, with $|H| = O(n^{1-\mu})$,

Path Decomposition

To compute distance from u to v we

- generate a set $H \subset V$ of vertices uniformly at random, with $|H| = O(n^{1-\mu})$,
- compute the distances in G restricted to the paths that use at most n^{μ} edges,

Path Decomposition

To compute distance from u to v we

- generate a set $H \subset V$ of vertices uniformly at random, with $|H| = O(n^{1-\mu})$,
- compute the distances in G restricted to the paths that use at most n^{μ} edges,
- compute the distances in G only between the vertices in H,

Path Decomposition

To compute distance from u to v we

- generate a set $H \subset V$ of vertices uniformly at random, with $|H| = O(n^{1-\mu})$,
- compute the distances in G restricted to the paths that use at most n^{μ} edges,
- compute the distances in G only between the vertices in H,
- compute the minimum sum of the distances
  - from u to $u' \in H$,
  - from v' to u',
  - from v to $v' \in H$.

Path Decomposition

This might work faster because

- we consider only paths that use small number of edges — $O(n^{\mu})$,
- the distances between the vertices in the subset H of vertices can be computed in time $O(n^{3-3\mu})$ with use of the short paths.

Dynamic Path Decomposition

In dynamic algorithm we maintain the following

Dynamic Path Decomposition

In dynamic algorithm we maintain the following

- a set $H \subset V$ of vertices generated uniformly at random,

Dynamic Path Decomposition

In dynamic algorithm we maintain the following

- a set $H \subset V$ of vertices generated uniformly at random,
  - has to be regenerated from time to time,

Dynamic Path Decomposition

In dynamic algorithm we maintain the following

- a set $H \subset V$ of vertices generated uniformly at random,
  - has to be regenerated from time to time,
- a matrix D of the distances in G restricted to the paths that use at most n^{μ} edges,

Dynamic Path Decomposition

In dynamic algorithm we maintain the following

- a set $H \subset V$ of vertices generated uniformly at random,
  - has to be regenerated from time to time,
- a matrix D of the distances in G restricted to the paths that use at most n^{μ} edges,
  - we will show latter how to compute it.

Dynamic Path Decomposition

In dynamic algorithm we maintain the following

- a set $H \subset V$ of vertices generated uniformly at random,
  - has to be regenerated from time to time,
- a matrix D of the distances in G restricted to the paths that use at most n^{μ} edges,
  - we will show latter how to compute it.
- compute the distances restricted to H.

Dynamic Path Decomposition

In dynamic algorithm we maintain the following

- a set $H \subset V$ of vertices generated uniformly at random,
  - has to be regenerated from time to time,
- a matrix D of the distances in G restricted to the paths that use at most n^{μ} edges,
  - we will show latter how to compute it.
- compute the distances restricted to H.
  - we recompute it using the matrix D.

Dynamic Path Decomposition

In order to answer a query on the distance from u to v we compute the minimum sum of the distances

- from u to $u' \in H$,
- from v' to u',
- from v to $v' \in H$.

Dynamic Path Decomposition

In order to answer a query on the distance from u to v we compute the minimum sum of the distances

- from u to $u' \in H$,
- from v' to u',
- from v to $v' \in H$.

To compute the matrix D we use algebraic approach.

Dynamic Shortest Distances

Theorem 10 *There exist an algorithm for dynamically computing shortest distances in unweighted graph supporting updates in $O(n^{1.95})$ time and queries in $O(n^{1.3})$ time.*

The algorithm is randomized and with small probability may return wrong higher values.