

From Dynamic Matrix Inverse to Dynamic Shortest Distances

Piotr Sankowski
sank@mimuw.edu.pl

Institute of Informatics

Warsaw University

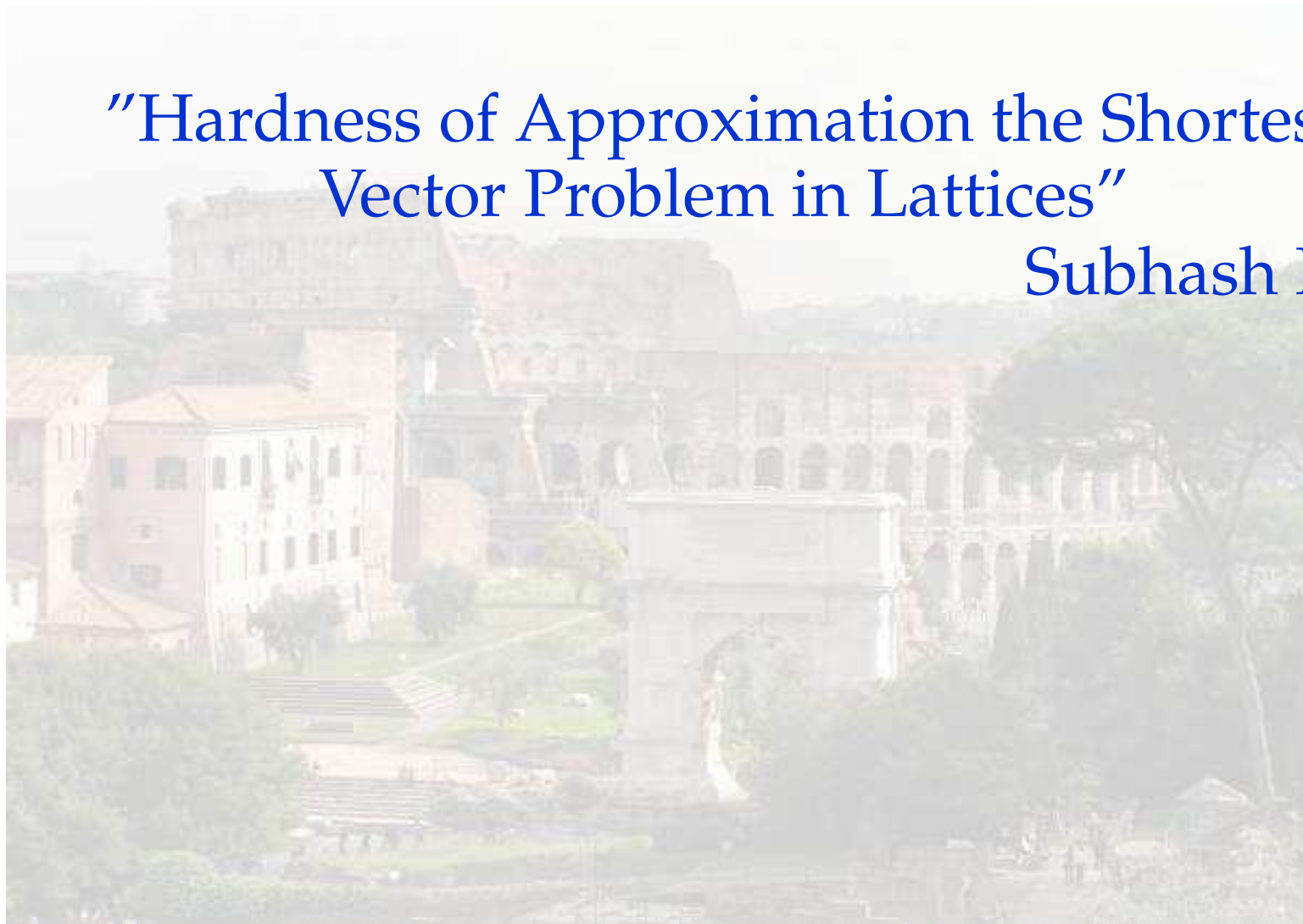
FOCS'04 Highlights



FOCS'04 Highlights

“Hardness of Approximation the Shortest
Vector Problem in Lattices”

Subhash Kh



FOCS'04 Highlights

“Hardness of Approximation the Shortest Vector Problem in Lattices”

Subhash Kh

Find the shortest vector in $\left\{ \sum_{i=1}^n a_i \vec{b}_i \mid a_i \in \mathcal{Z} \right\}$,
where $\vec{b}_i \in \mathcal{R}^m$.

FOCS'04 Highlights

“Hardness of Approximation the Shortest Vector Problem in Lattices”

Subhash Kh

Find the shortest vector in $\left\{ \sum_{i=1}^n a_i \vec{b}_i \mid a_i \in \mathcal{Z} \right\}$,
where $\vec{b}_i \in \mathcal{R}^m$.

Let $p > 1$, it is hard to approximate shortest vector in l_p norm within an arbitrarily constant factor.

FOCS'04 Highlights



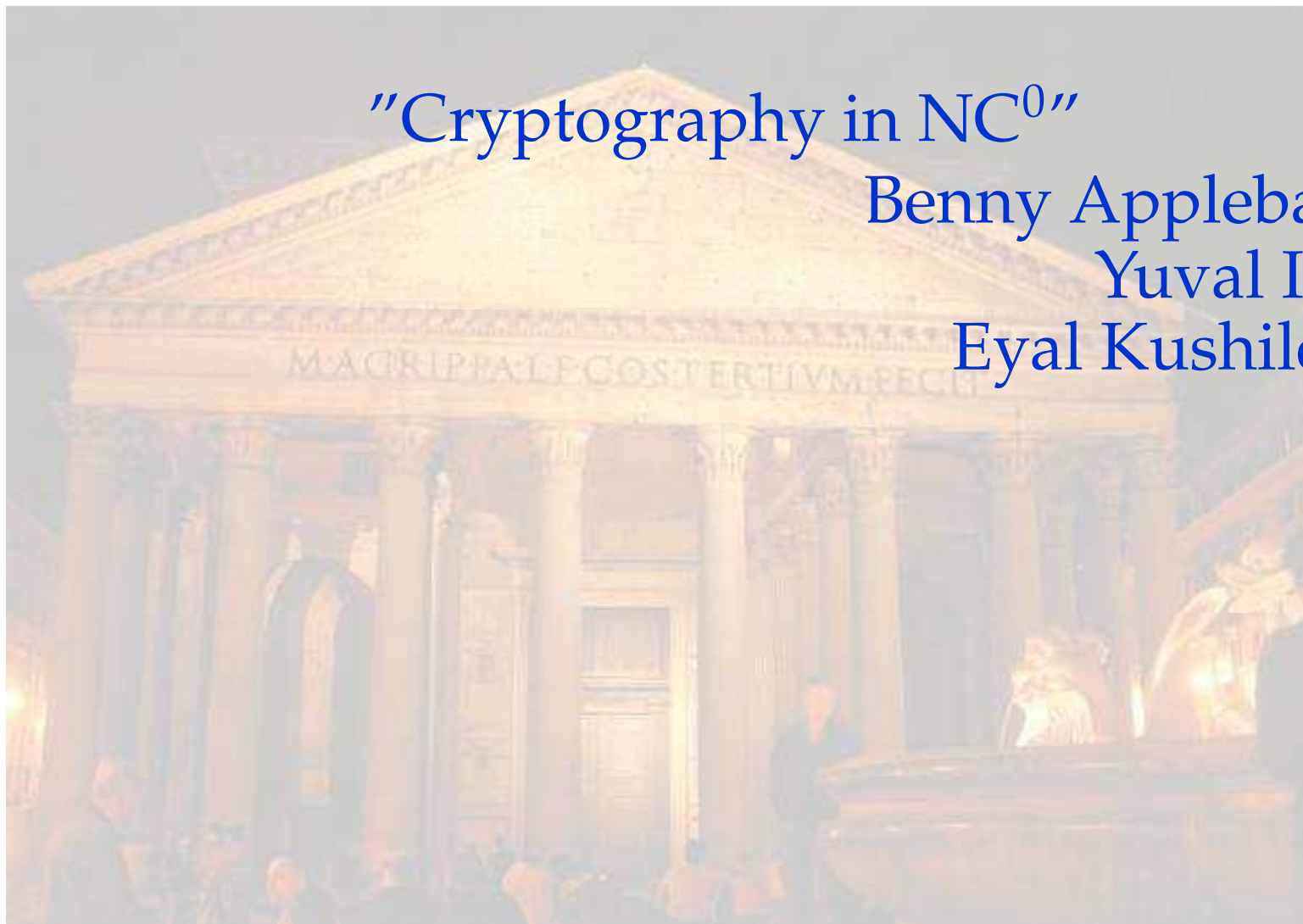
FOCS'04 Highlights

"Cryptography in NC⁰"

Benny Applebaum

Yuval Ishai

Eyal Kushilevich



FOCS'04 Highlights

"Cryptography in NC^0 "

Benny Applebaum

Yuval Ishai

Eyal Kushilevich

Every OWF (resp., PRG) computable in NC^1 ,
can be compiled into a corresponding OWF
(resp., PRG) in NC_4^0 .

FOCS'04 Highlights



FOCS'04 Highlights

“An Approximate Max-Steiner-Tree-Packing
Min-Steiner-Cut Theorem”

Lap Chi Lau



FOCS'04 Highlights

“An Approximate Max-Steiner-Tree-Packing
Min-Steiner-Cut Theorem”

Lap Chi Lau

Multigraph G and $S \subseteq V(G)$ to find largest
collection of edge-disjoint trees that each
connects S .

FOCS'04 Highlights

“An Approximate Max-Steiner-Tree-Packing
Min-Steiner-Cut Theorem”

Lap Chi Lau

Multigraph G and $S \subseteq V(G)$ to find largest
collection of edge-disjoint trees that each
connects S .

First polynomial time constant factor
approximation algorithm.

Problems

How to dynamically compute the inverse of a matrix?

How to dynamically compute the transitive closure of a graph?

How to dynamically compute the shortest distances in a graph?

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, \dots, x_n): set the input vector to (x_1, \dots, 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, \dots, x_n): set the input vector to (x_1, \dots, 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, \dots, x_n): set the input vector to (x_1, \dots, 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 $\text{adj}(A)$,

Dynamic Matrix Functions

We consider the following problems:

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

- **adjoint:** input A , output $\text{adj}(A)$,

where $\text{adj}(A)_{ij} = (-1)^{i+j} \det(A^{ji})$ 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 $\text{adj}(A)$,
- **inverse:** input A , output A^{-1} ,

Dynamic Matrix Functions

We consider the following problems:

- **determinant:** input A , output $\det(A)$,
- **adjoint:** input A , output $\text{adj}(A)$,
- **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 $\text{adj}(A)$,
- **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 a $n \times n^\epsilon$ matrix by an $n^\epsilon \times n$ matrix.

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

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 inverses 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 inverses 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 inverses 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 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 |
|------------------------------------|------------------------|----------------------|
| <i>Henzinger and King '95</i> | $\tilde{O}(nm^{0.58})$ | $\Theta(n / \log n)$ |
| <i>King and Sagert '99</i> | $O(n^{2.26})$ | $O(1)$ |
| <i>King '99</i> | $O(n^2 \log n)$ | $O(1)$ |
| <i>Demetrescu and Italiano '00</i> | $O(n^2)$ | $O(1)$ |
| <i>Roditty and Zwick '02</i> | $O(m\sqrt{n})$ | $O(\sqrt{n})$ |
| <i>Roditty and Zwick '04</i> | $O(m + n \log n)$ | $O(n)$ |
| <i>This work</i> | $O(n^{1.575})$ | $O(n^{0.575})$ |
| <i>This work</i> | $O(n^{1.495})$ | $O(n^{1.495})$ |

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.

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

Dynamic Transitive Closure

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

Dynamic Transitive Closure

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

- *There exists a path in G from i to j , iff $\text{adj}(I - A)$ 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 , then:*

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

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

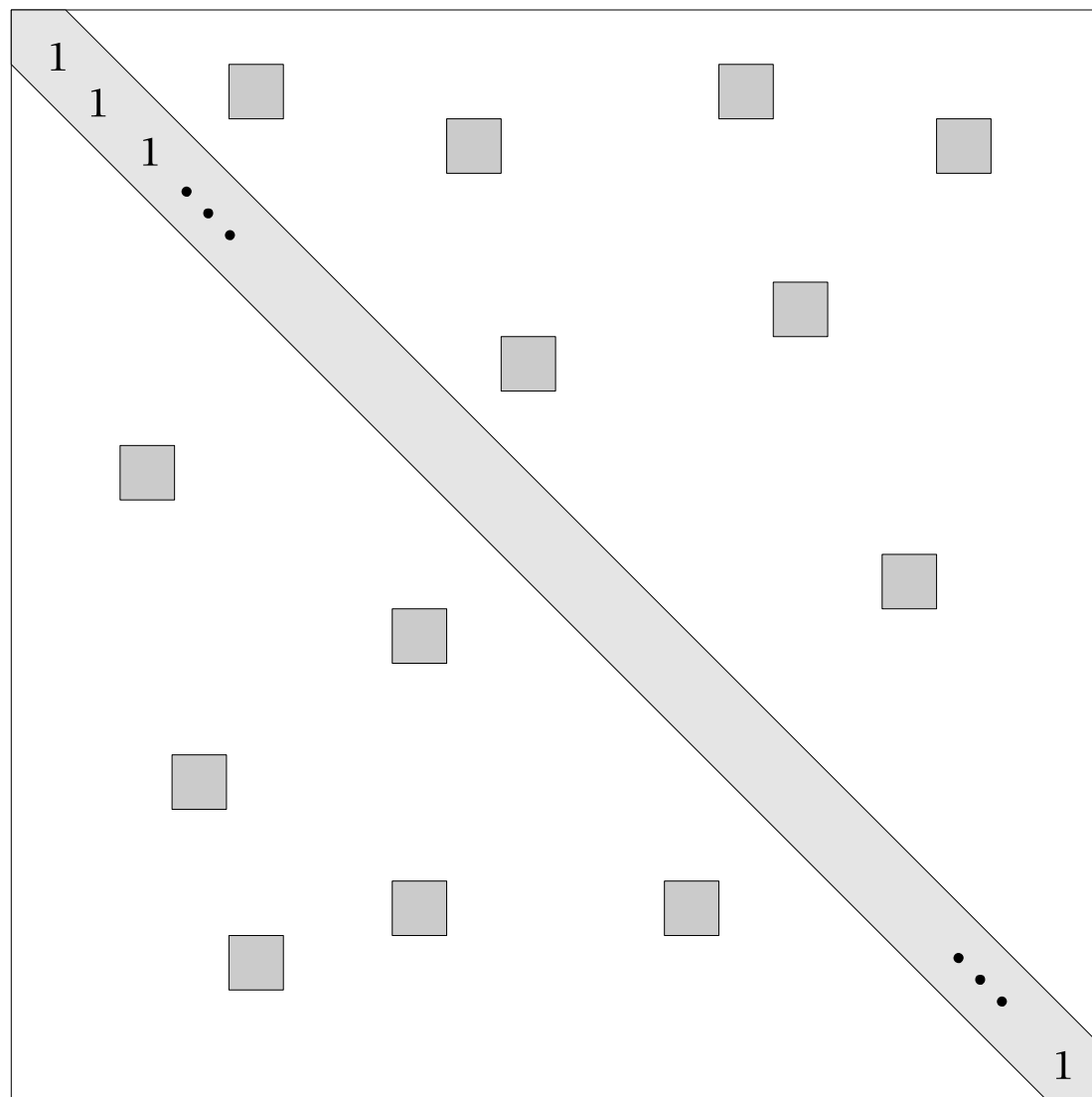
$$\det(X) =$$

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

The permutation p is a cycle cover of a graph.

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

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



Reduction

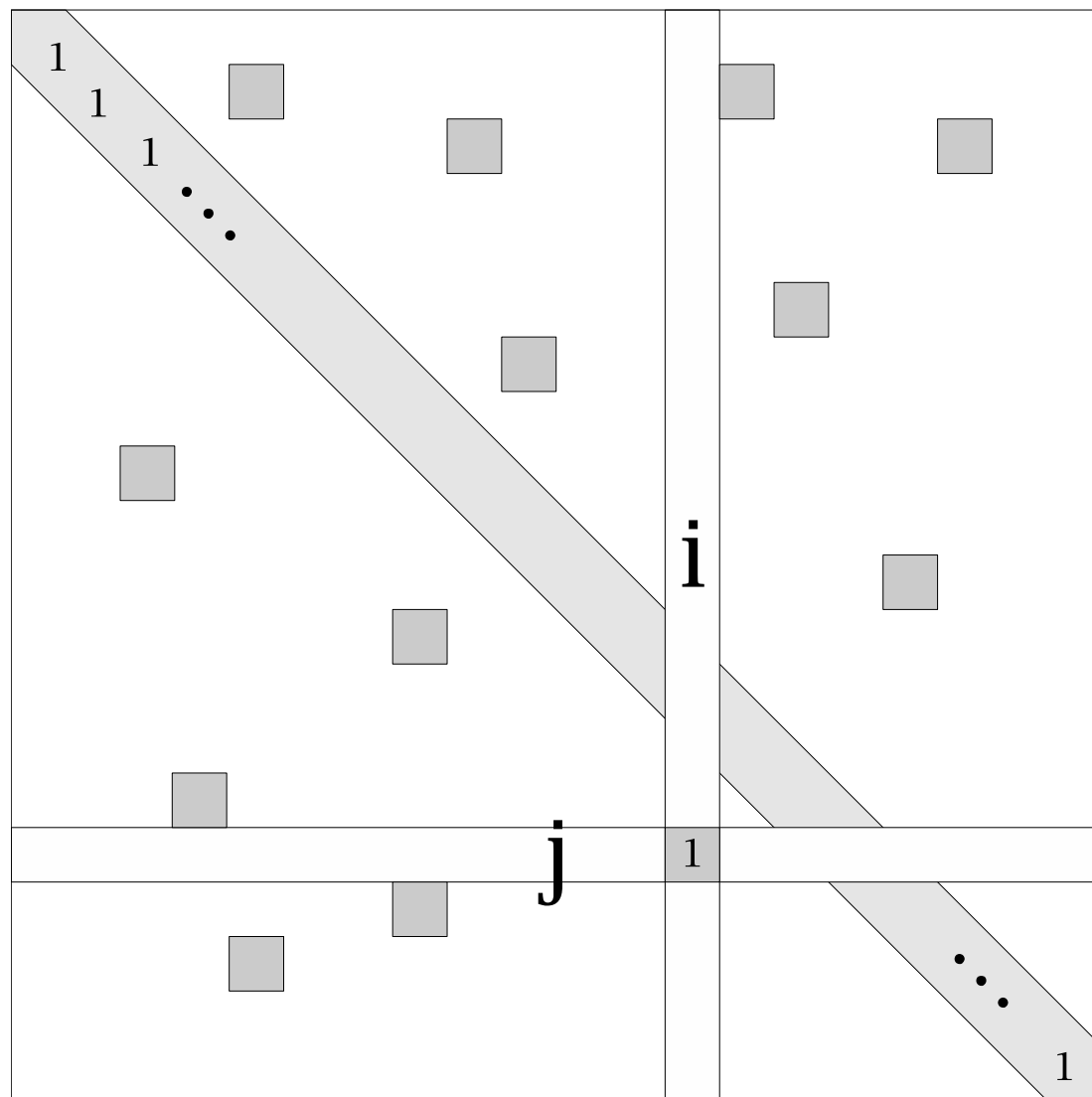
$$\det(X) =$$

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

The permutation p is a cycle cover of a graph.

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

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



Dynamic Transitive Closure

Theorem 5

Dynamic Matrix Inverse

Update $O(n^\alpha)$ operations

Query $O(n^\beta)$ operations

one may assume non-singularity \Downarrow

\Downarrow 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

$$A = A(u)|_{u=1},$$

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

$$\text{adj}(A) = \text{adj}(A(u))|_{u=1}.$$

Strassen's Idea

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

$$\begin{aligned}\frac{1}{1-z} &= 1 + z + z^2 + \dots = \\ &= (1 + z)(1 + z^2)(1 + z^4) \dots\end{aligned}$$

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

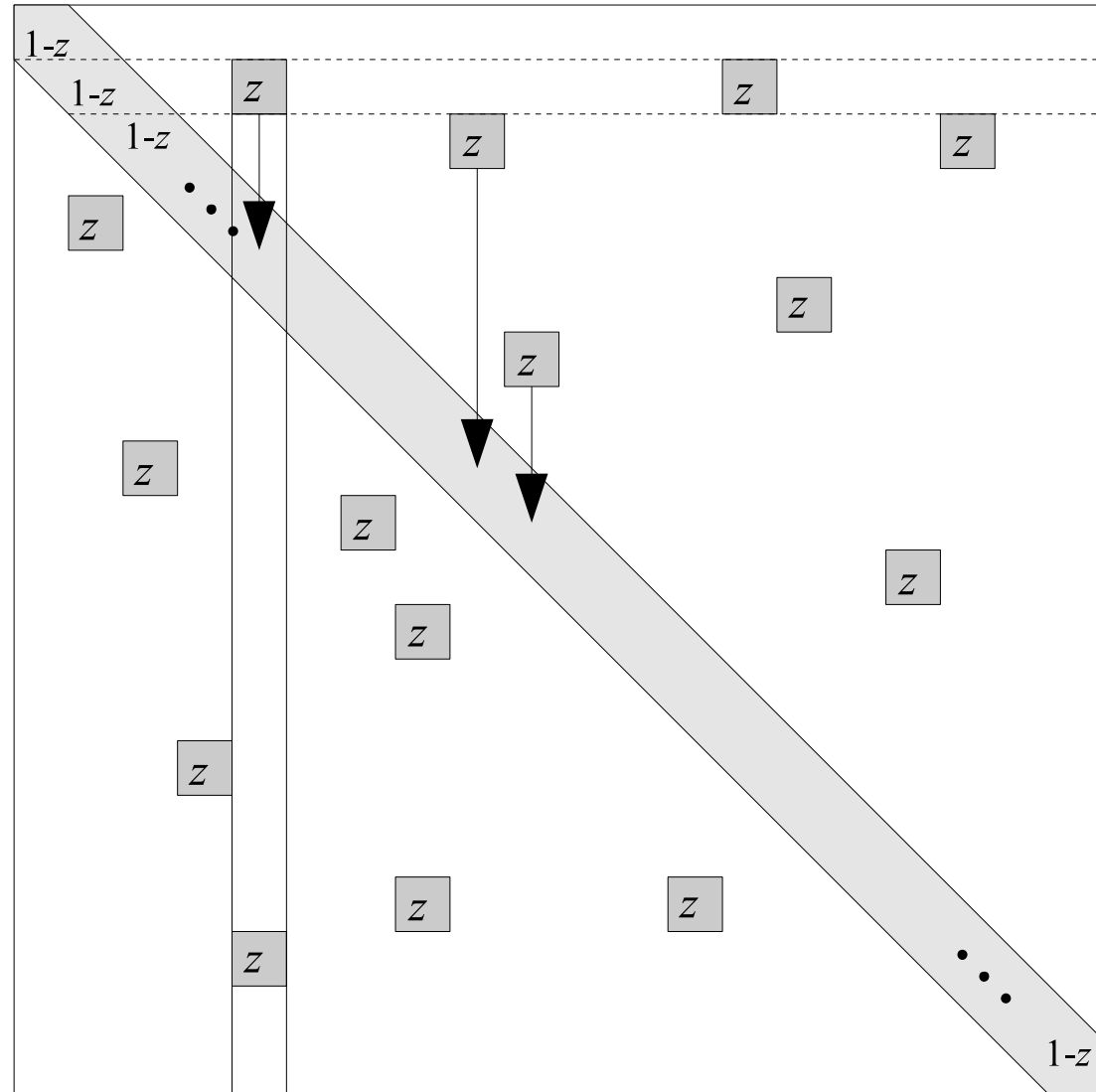
Strassen's Idea

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

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

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 us substitute $B = I + be_i^T$, then:

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

Dynamic Matrix Inverse over Rings

We compute matrix B such that

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

Let us substitute $B = I + be_i^T$, then:

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

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

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.

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

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

To answer queries in constant number of operations we recompute the matrix $\text{adj}(A) = \text{adj}(A(u))|_{u=1}$ 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 $\text{adj}(A(u))_{ij} = \det(A(u))(A(u)^{-1})_{ij}$.

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

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

Shortest Distances

| | Update | Query |
|--|--|--|
| <i>Henzinger et al. '97</i> Planar Graphs | $O(n^{\frac{4}{3}} \log(nC))$ | $O(n^{\frac{4}{3}} \log(nC))$ |
| <i>Fakcharoemphol and Rao '02</i> Planar Graphs | $O(n^{\frac{4}{5}} \log^{\frac{13}{5}} n)$ | $O(n^{\frac{4}{5}} \log^{\frac{13}{5}} n)$ |
| <i>King '99</i> | $O(n^{2.5} \sqrt{C \log n})$ | $O(1)$ |
| <i>Demetrescu and Italiano '00</i> Real Weights | $O(n^{2.5} \sqrt{S \log^3 n})$ | $O(1)$ |
| <i>Demetrescu and Italiano '04</i> | $\tilde{O}(n^2)$ | $O(1)$ |
| <i>This work</i> | $O(n^{1.932})$ | $O(n^{1.288})$ |

Shortest Distances

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

Shortest Distances

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

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

Shortest Distances

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

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

Reduction

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

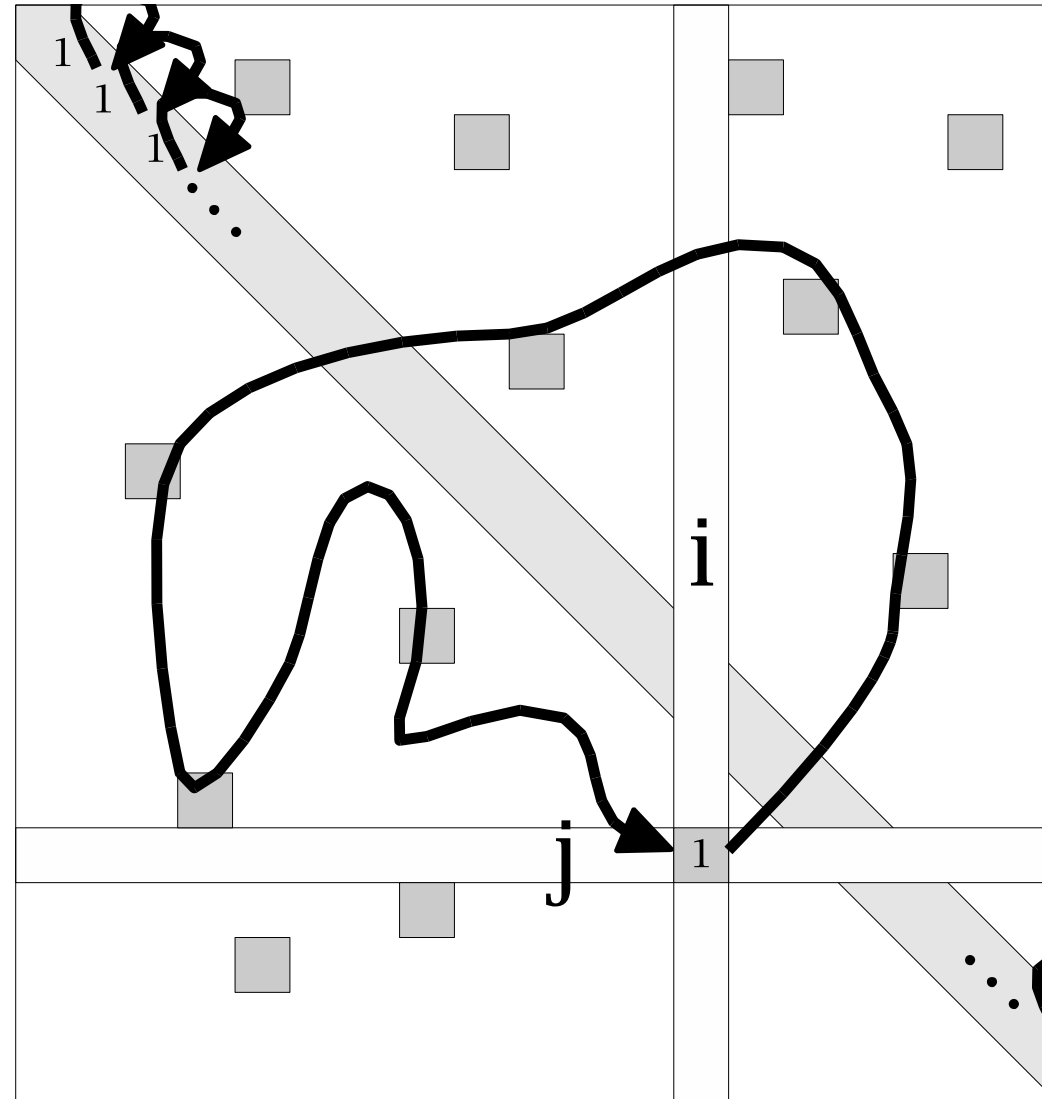
The permutation p is a cycle cover of a graph.

Reduction

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

The permutation p is a cycle cover of a graph.

$$\text{adj}(I - A)_{ij} = (-1)^{i+j} \det((I - A)^{ji})$$



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 $1, \dots, p - 1$.

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 $1, \dots, p - 1$.
- 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 $1, \dots, p - 1$.
- Now maintain dynamically the adjoint of the matrix $I - uB$ over polynomials of degree n ,
- answer queries by finding the smallest degree term in $\text{adj}(I - u\tilde{B})_{ij}$.

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 $1, \dots, p - 1$.
- Now maintain dynamically the adjoint of the matrix $I - uB$ over polynomials of degree n ,
- answer queries by finding the smallest degree term in $\text{adj}(I - u\tilde{B})_{ij}$.

In this way we get algorithm that supports updates in $\tilde{O}(n^{2.575})$ 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 an 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.

Dynamic Shortest Distances

Theorem 9 (Ullman and Yannakakis '90)

- *Let $H \subseteq V$ be a set of vertices chosen uniformly at random.*

Dynamic Shortest Distances

Theorem 9 (Ullman and Yannakakis '90)

- *Let $H \subseteq V$ be a set of vertices chosen uniformly at random.*
- *Let P_n be the probability that a given simple path has a sequence of more than $\frac{cn}{|H|} \log n$ vertices, none of which are from H ,*

Dynamic Shortest Distances

Theorem 9 (Ullman and Yannakakis '90)

- *Let $H \subseteq V$ be a set of vertices chosen uniformly at random.*
- *Let P_n be the probability that a given simple path has a sequence of more than $\frac{cn}{|H|} \log n$ vertices, none of which are from H ,*
- *For any $c > 0$, and for sufficiently large n , P_n is bounded by $2^{-\alpha c}$ for some positive α .*

Short Long Path Decomposition

The distances up to k are computed exactly —
matrix D .

Short Long Path Decomposition

The distances up to k are computed exactly — matrix D .

For a chosen set of vertices H we compute distances between them using D .

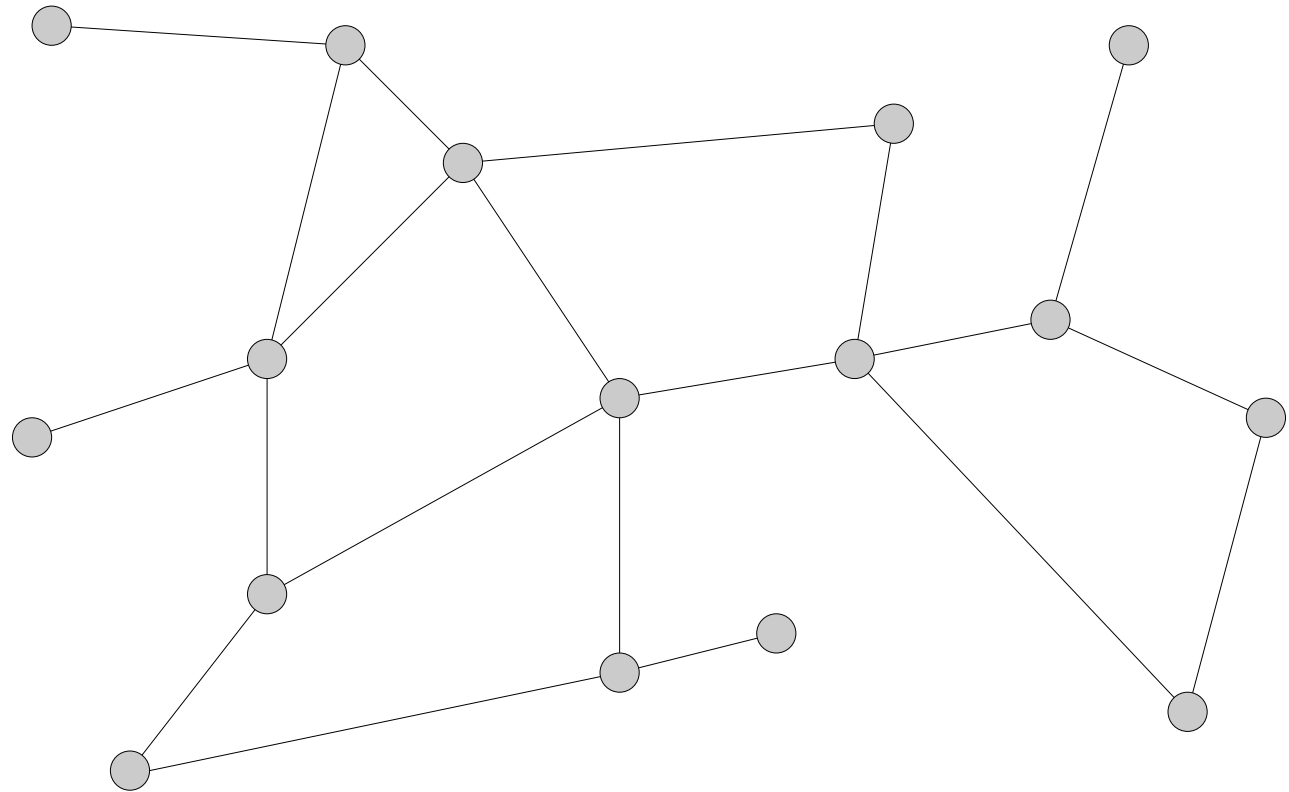
Short Long Path Decomposition

The distances up to k are computed exactly — matrix D .

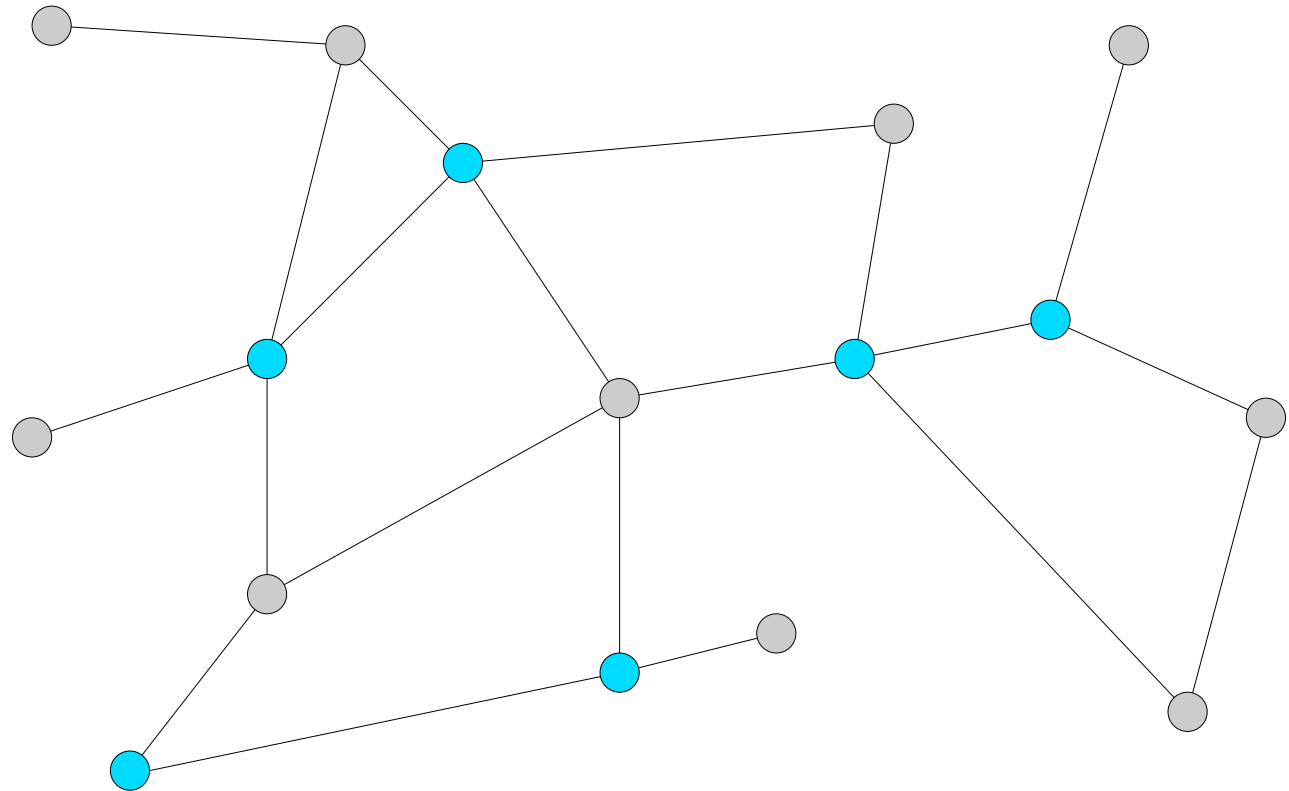
For a chosen set of vertices H we compute distances between them using D .

Distance between i and j is given by a distance from i to H and then from H to j .

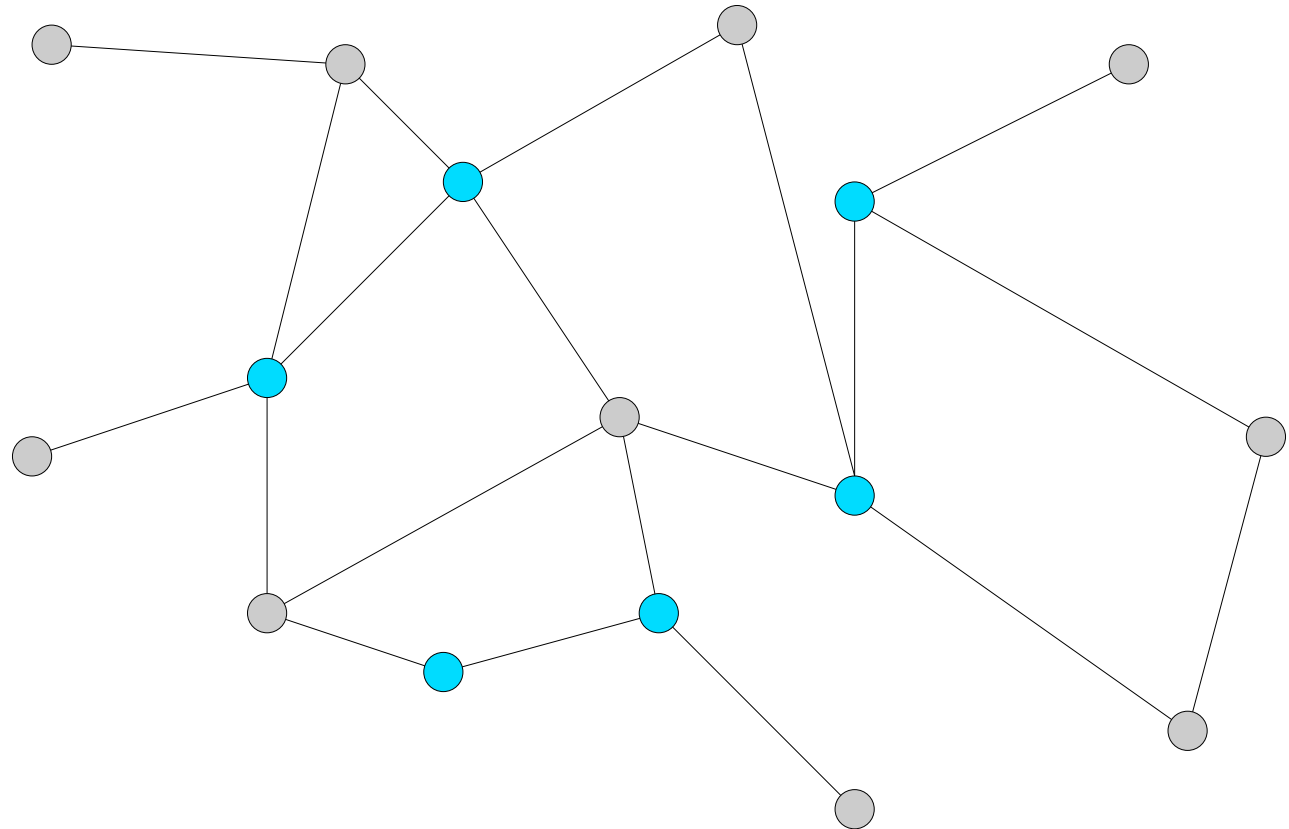
Short Long Path Decomposition



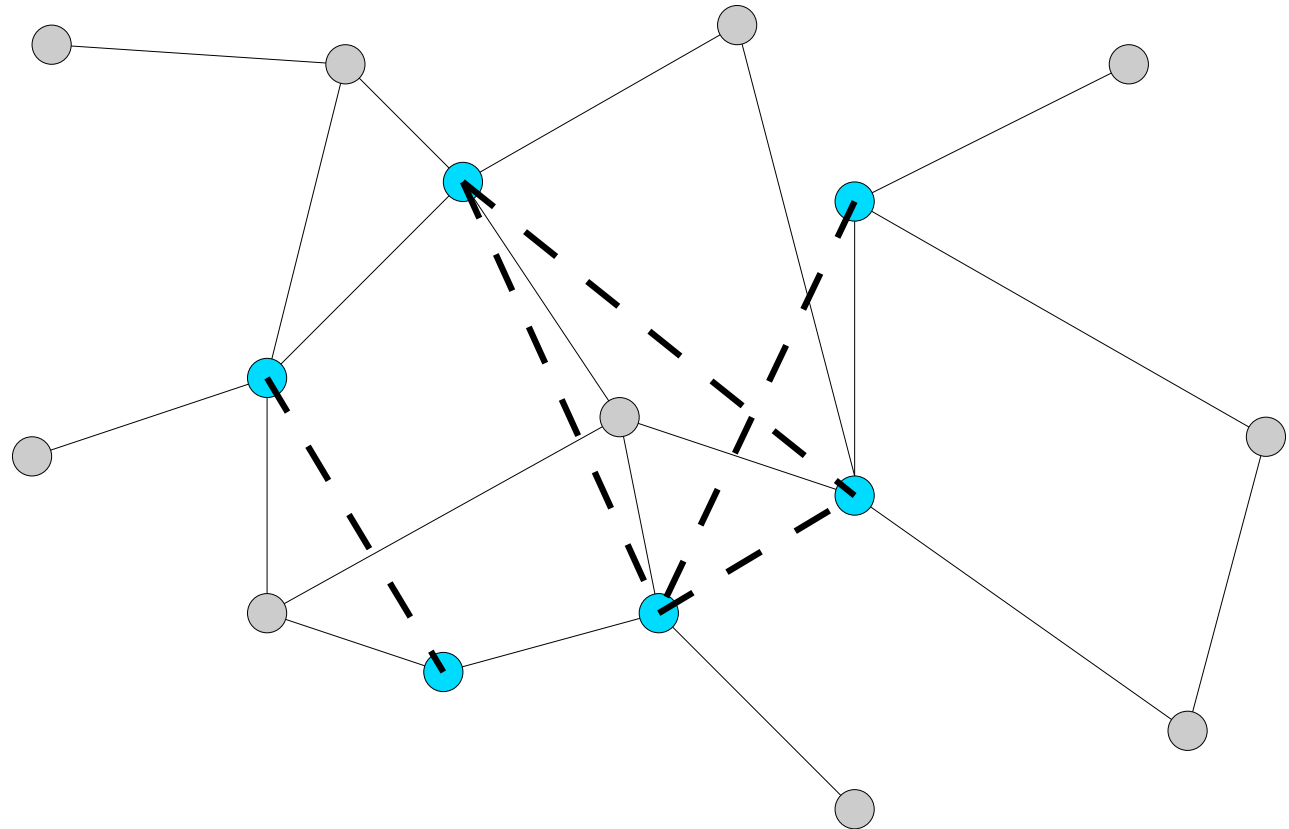
Short Long Path Decomposition



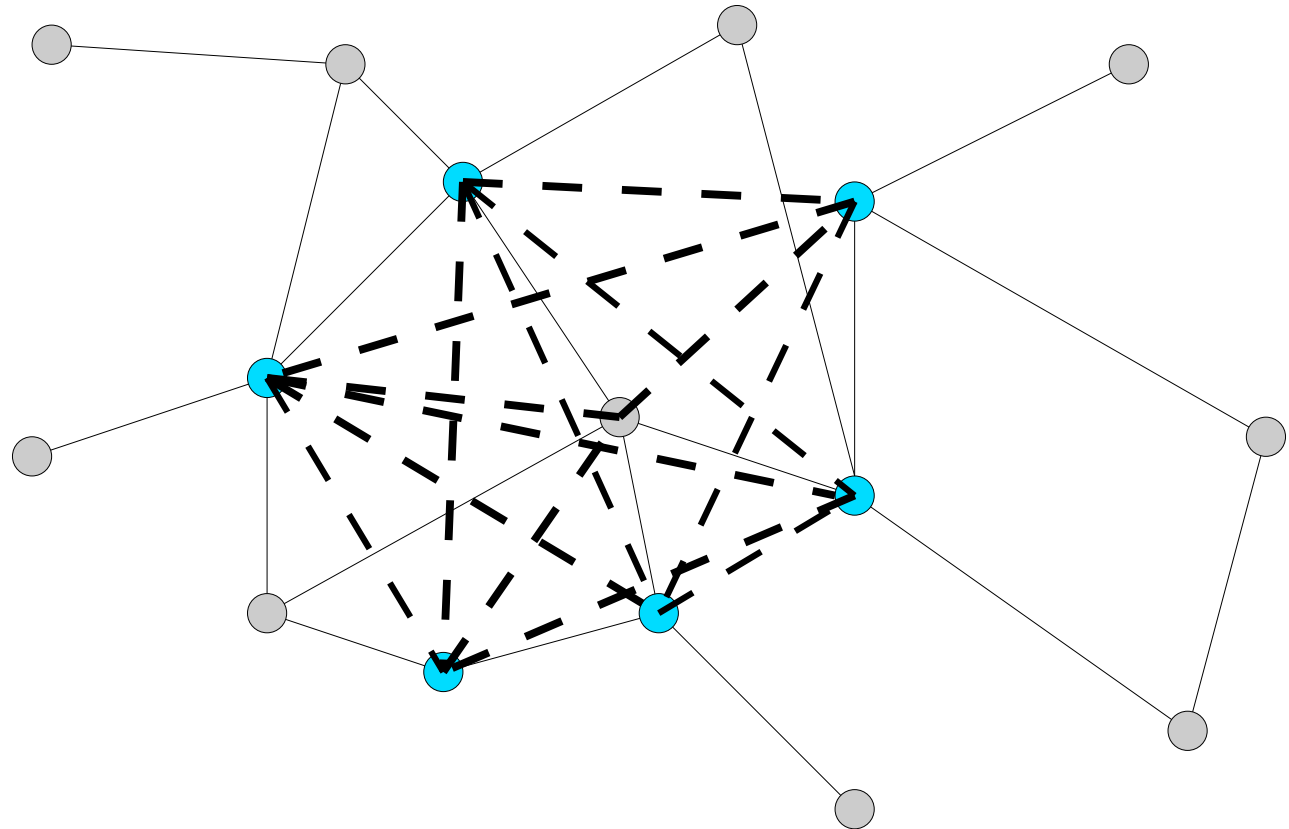
Short Long Path Decomposition



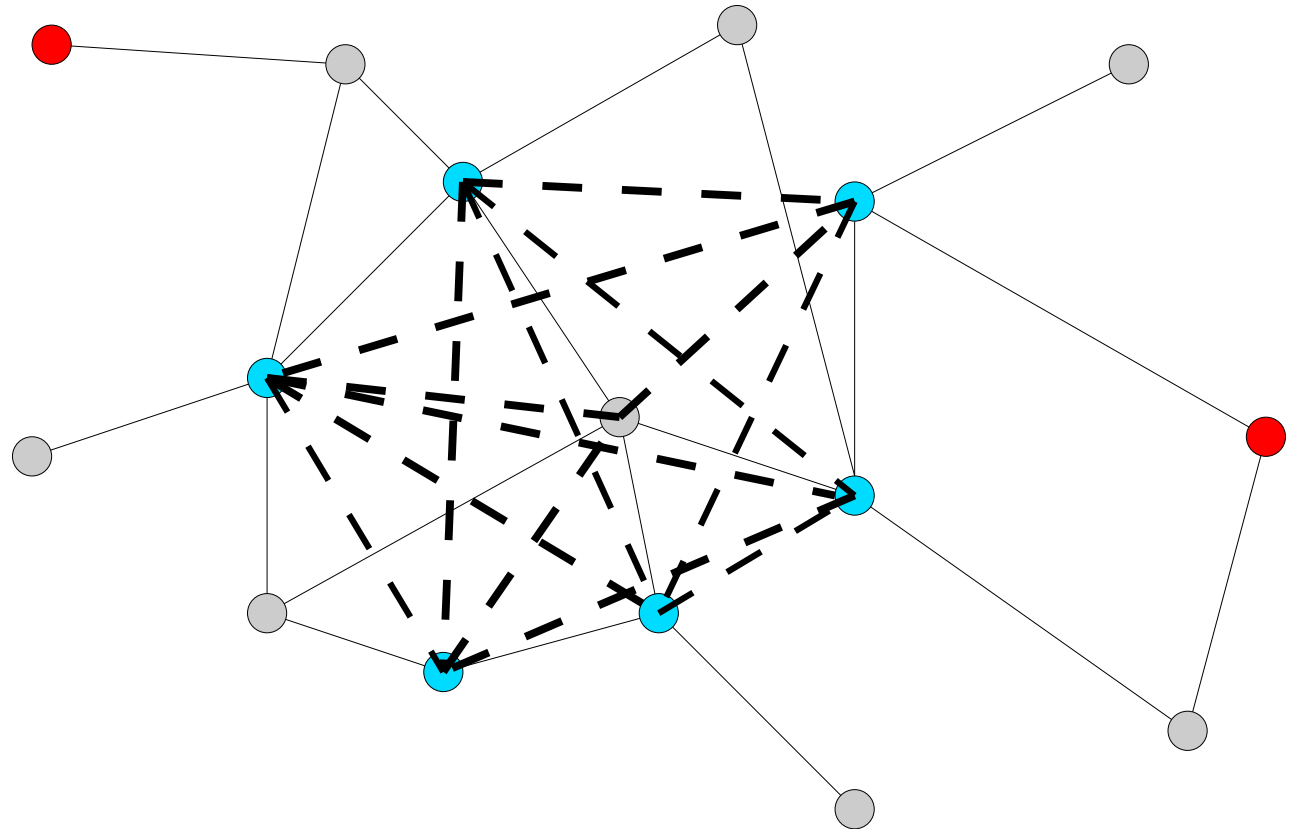
Short Long Path Decomposition



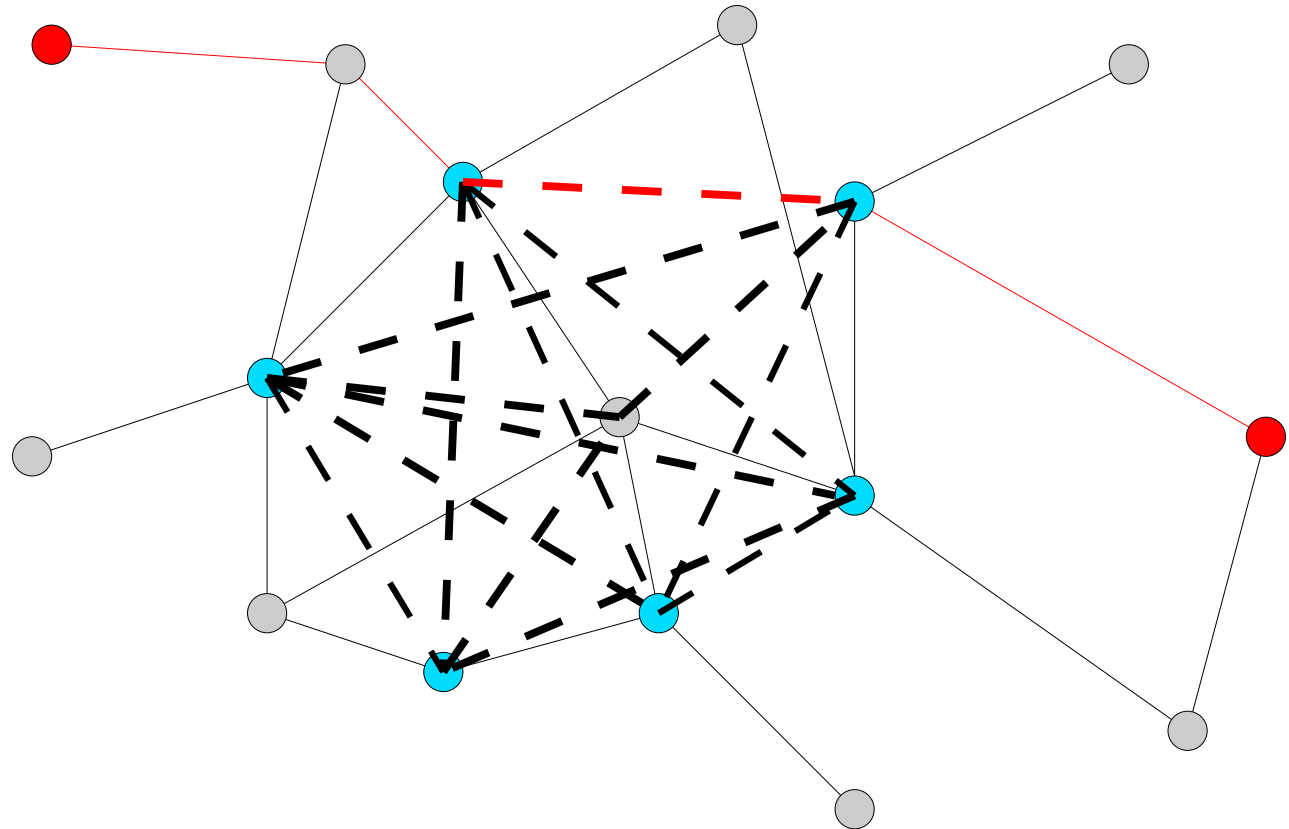
Short Long Path Decomposition



Short Long Path Decomposition



Short Long Path Decomposition



Short Long Path Decomposition - Updates

- A set $H \subseteq V$ of vertices uniformly at random with $|H| = \frac{cn}{n^\mu} \log n = O(n^{1-\mu})$, for any constant $c > 0$, where $c \log n \leq n^\mu \leq n$.

Short Long Path Decomposition - Updates

- A set $H \subseteq V$ of vertices uniformly at random with $|H| = \frac{cn}{n^\mu} \log n = O(n^{1-\mu})$, for any constant $c > 0$, where $c \log n \leq n^\mu \leq n$.
- An $n \times n$ matrix D such that D_{ij} is the shortest distance from i to j in G using at most n^μ edges.

Short Long Path Decomposition - Updates

- A set $H \subseteq V$ of vertices uniformly at random with $|H| = \frac{cn}{n^\mu} \log n = O(n^{1-\mu})$, for any constant $c > 0$, where $c \log n \leq n^\mu \leq n$.
- An $n \times n$ matrix D such that D_{ij} is the shortest distance from i to j in G using at most n^μ edges.
- An $|H| \times |H|$ matrix B obtained from D by choosing only columns and rows corresponding to H .

Short Long Path Decomposition - Updates

- A set $H \subseteq V$ of vertices uniformly at random with $|H| = \frac{cn}{n^\mu} \log n = O(n^{1-\mu})$, for any constant $c > 0$, where $c \log n \leq n^\mu \leq n$.
- An $n \times n$ matrix D such that D_{ij} is the shortest distance from i to j in G using at most n^μ edges.
- An $|H| \times |H|$ matrix B obtained from D by choosing only columns and rows corresponding to H .
- The Kleene closure B^* of matrix B , i.e., the distance matrix obtained from B .

Short Long Path Decomposition - Updates

- Updating the matrix D takes — $O(n^\mu n^{1.575})$ time.

Short Long Path Decomposition - Updates

- Updating the matrix D takes — $O(n^\mu n^{1.575})$ time.
- Querying out the matrix B takes — $O(n^\mu n^{0.575} n^{2-2\mu})$ time.

Short Long Path Decomposition - Updates

- Updating the matrix D takes — $O(n^\mu n^{1.575})$ time.
- Querying out the matrix B takes — $O(n^\mu n^{0.575} n^{2-2\mu})$ time.
- Computing the matrix B^* takes — $O(n^{3-3\mu})$ time.

Short Long Path Decomposition - Updates

- Updating the matrix D takes — $O(n^\mu n^{1.575})$ time.
- Querying out the matrix B takes — $O(n^\mu n^{0.575} n^{2-2\mu})$ time.
- Computing the matrix B^* takes — $O(n^{3-3\mu})$ time.

For $\mu = 0.5$ we get $O(n^{2.075})$ update time.

Dynamic Shortest Distances

The the distance from i to j can be computed with:

$$\min \left\{ D_{ij}, \min_{p,q \in H} \left\{ D_{ip} + B_{pq}^* + D_{qj} \right\} \right\}.$$

Dynamic Shortest Distances

The the distance from i to j can be computed with:

$$\min \left\{ D_{ij}, \min_{p,q \in H} \left\{ D_{ip} + B_{pq}^* + D_{qj} \right\} \right\}.$$

- we have to query out row and column from M
— $O(n^\mu n^{0.575} n^{1-\mu})$ time,

Dynamic Shortest Distances

The the distance from i to j can be computed with:

$$\min \left\{ D_{ij}, \min_{p,q \in H} \left\{ D_{ip} + B_{pq}^* + D_{qj} \right\} \right\}.$$

- we have to query out row and column from L — $O(n^\mu n^{0.575} n^{1-\mu})$ time,
- to compute the minimum over p, q — $O(n^{2-2\mu})$ time.

Dynamic Shortest Distances

The the distance from i to j can be computed with:

$$\min \left\{ D_{ij}, \min_{p,q \in H} \left\{ D_{ip} + B_{pq}^* + D_{qj} \right\} \right\}.$$

- we have to query out row and column from M — $O(n^\mu n^{0.575} n^{1-\mu})$ time,
- to compute the minimum over p, q — $O(n^{2-2\mu})$ time.

For $\mu = 0.5$ we obtain $O(n^{1.575})$ query time.

Subquadratic Dynamic Distances

We have to query out a large submatrix B of the matrix D .

Subquadratic Dynamic Distances

We have to query out a large submatrix B of the matrix D .

We can use fast matrix multiplication to do this
— $O(n^{\omega(1-\mu, \epsilon, 1-\mu)})$ arithmetic operations.

Subquadratic Dynamic Distances

We have to query out a large submatrix B of the matrix D .

We can use fast matrix multiplication to do this
— $O(n^{\omega(1-\mu, \epsilon, 1-\mu)})$ arithmetic operations.

$$O(n^\mu n^{1.575} + n^\mu n^{\omega(1-\mu, 0.575, 1-\mu)} + n^{3-3\mu})$$

Subquadratic Dynamic Distances

We have to query out a large submatrix B of the matrix D .

We can use fast matrix multiplication to do this — $O(n^{\omega(1-\mu, \epsilon, 1-\mu)})$ arithmetic operations.

$$O(n^\mu n^{1.575} + n^\mu n^{\omega(1-\mu, 0.575, 1-\mu)} + n^{3-3\mu})$$

For $\mu = \frac{3-1.575}{4} = 0.357$ we obtain $O(n^{1.932})$ update time.

Subquadratic Dynamic Distances

Because $1 - \mu > 0.575$, we have

$$\begin{aligned} O(n^\mu n^{\omega(1-\mu, 0.575, 1-\mu)}) &= O(n^\mu n^{(1-\mu)\omega}) = \\ &= O(n^{1.885}) = O(n^{1.932}) \end{aligned}$$

Subquadratic Dynamic Distances

Because $1 - \mu > 0.575$, we have

$$\begin{aligned} O(n^\mu n^{\omega(1-\mu, 0.575, 1-\mu)}) &= O(n^\mu n^{(1-\mu)\omega}) = \\ &= O(n^{1.885}) = O(n^{1.932}) \end{aligned}$$

The query cost is unchanged — $O(n^{1.575})$ time.

Subquadratic Dynamic Distances

Theorem 10 *There exists an randomized algorithm for the dynamic shortest distances problem in unweighted graph $G = (V, E)$ supporting edge updates in $O(n^{1.932})$ and queries in $O(n^{1.288})$ time.*

Other Applications

The dynamic matrix inverse algorithm can be used also for:

- counting spanning trees in graphs,
- testing for allowed edges,
- computing shortest paths lengths.

Summary

- Dynamic algorithms for computing:
- determinant,
 - matrix inverse,
 - adjoint,
 - solution to a linear system of equation,
 - transitive closure.