Algebraic Graph Algorithms
Part I

Marek Cygan & Piotr Sankowski

University of Warsaw

Algorithmic Trends 19.02.2014

Outline - Part I & II

- Algebraic algorithms - idea
- Simple example - perfect matchings
- Shortest cycles in directed graphs
- Shortest paths in directed graphs
- Dynamic matrix algorithms
  - determinant and inverse
- Dynamic graph algorithms
  - transitive closure
- Static graph algorithms
  - matchings in graphs

Matrix Multiplication

$$C = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \times \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n,1} & b_{n,2} & \cdots & b_{n,n} \end{bmatrix}$$

Naive algorithm

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j} = a_{i,1} b_{1,j} + a_{i,2} b_{2,j} + \ldots + a_{i,n} b_{n,j}.$$

requires n operations to compute each element of C.
This gives $\sim n^3$ operations in total.

Strassen's Algorithm

$$\begin{bmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \times \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix}$$

$$Q_1 = (a_{1,1} + a_{2,2})(b_{1,1} + b_{2,2})$$

$$Q_2 = (a_{2,1} + a_{2,2})b_{1,1}$$

$$Q_3 = a_{1,1}(b_{1,2} - b_{2,2})$$

$$Q_4 = a_{2,2}(-b_{1,1} + b_{2,1})$$

$$Q_5 = (a_{1,1} + a_{1,2})b_{2,2}$$

$$Q_6 = (-a_{1,1} + a_{2,1})(b_{1,1} + b_{1,2})$$

$$Q_7 = (a_{1,2} - a_{2,2})(b_{2,1} + b_{2,2})$$

$$c_{1,1} = Q_1 + Q_4 - Q_5 + Q_7$$

$$c_{2,1} = Q_2 + Q_4$$

$$c_{1,2} = Q_3 + Q_5$$

$$c_{2,2} = Q_1 + Q_3 - Q_2 + Q_6$$

The matrix C can be computed with use of 7 multiplications instead 8!

Strassen Algorithm

After dividing the matrix in blocks we get

$$\begin{bmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{bmatrix} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \times \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}$$

we have to do 2×2 matrix multiplication on blocks.

Using this recursive multiplication, we need

$$\sim n^{\log_2 7} = n^{2.81},$$

operations to multiply $n \times n$ matrices.

Fast Matrix Multiplication

The matrix multiplication exponent is denoted by ω.

The $n \times n$ by $n \times n$ multiplication requires

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \times \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n,1} & b_{n,2} & \cdots & b_{n,n} \end{bmatrix}$$

$\sim n^{\omega}$ operations.

The best known bound is $\omega < 2.38$ —

Coppersmith, Winograd '90, Stathers '10, Williams '11.

Fast Matrix Multiplication

In $O(n^\omega)$ time for a $n \times n$ matrix we can:

- compute the determinant,
- compute the characteristic polynomial,
- compute the inverse matrix,
- solve the system of linear equations,
- compute the determinant of polynomial matrix,
- solve the system of linear equations over polynomials.

We will use these to solve graph problems.

Algebraic Algorithms

The determinant of the $n \times n$ matrix A is given as:

$$\det(A) = \sum_{p \in \Pi_n} \sigma(p) \prod_{i=0}^{n} a_{i,p_i}.$$

Is it possible to encode the graph problem in the matrix A in such a way that the element of the sum correspond to the solution of the problem?

By testing if the determinant is non-zero we will know if the problem has a solution.

Dynamic Problems

We want to solve given problem for a data structure that can be changed, e.g., we add and remove edges from the graph.

Can the algebraic methods be used in such a case?

YES

- if we can show dynamic algorithms for algebraic problems,
- if we can show appropriate reductions.

Example: Matchings

A *matching* in the graph $G = (V, E)$ is a subset of edges $M \subseteq E$ such, that no two edges in M share a common endpoint.

A *perfect* matching is a matching of size $|V|/2$.

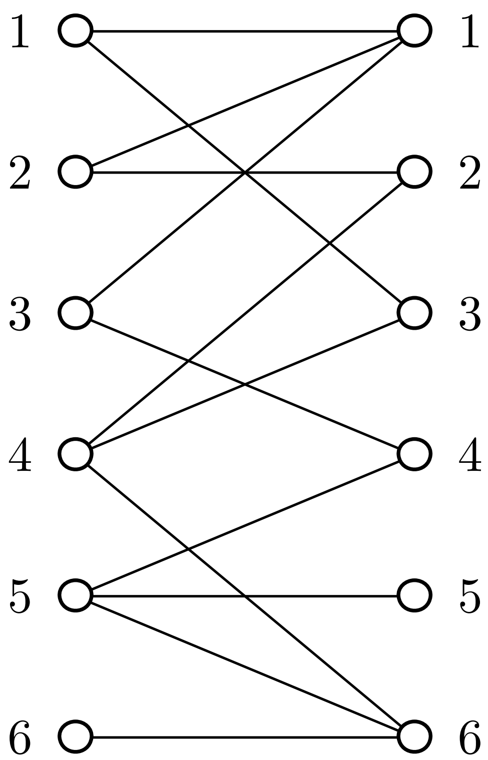
We want to:

- test if a graph contains a perfect matching,
- find any perfect matching in a graph,
- *find the maximum matching in the graph.*

Symbolic Adjacency Matrix

A symbolic adjacency matrix of the bipartite graph:

G



$$\Longrightarrow \quad \tilde{B}(G) = \begin{pmatrix} x_{11} & 0 & x_{13} & 0 & 0 & 0 \\ x_{21} & x_{22} & 0 & 0 & 0 & 0 \\ x_{31} & 0 & 0 & x_{34} & 0 & 0 \\ 0 & x_{42} & x_{43} & 0 & 0 & x_{46} \\ 0 & 0 & 0 & x_{54} & x_{55} & x_{56} \\ 0 & 0 & 0 & 0 & 0 & x_{66} \end{pmatrix}$$

Symbolic Adjacency Matrix

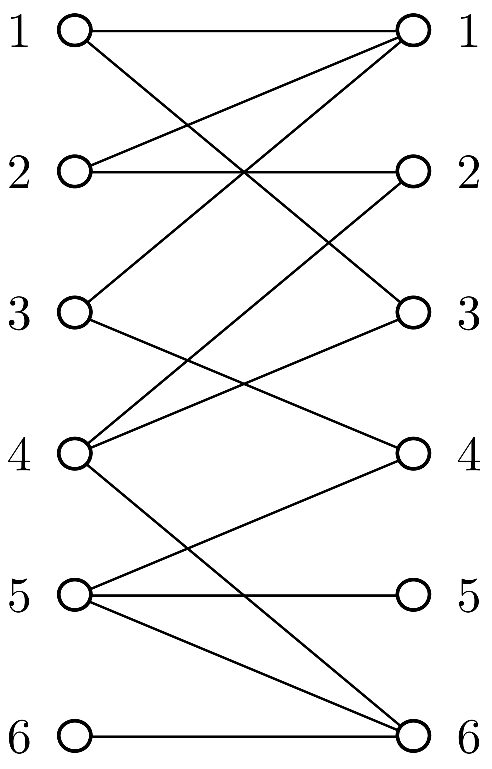
$$\det \begin{pmatrix} x_{11} & 0 & x_{13} & 0 & 0 & 0 \\ x_{21} & x_{22} & 0 & 0 & 0 & 0 \\ x_{31} & 0 & 0 & x_{34} & 0 & 0 \\ 0 & x_{42} & x_{43} & 0 & 0 & x_{46} \\ 0 & 0 & 0 & x_{54} & x_{55} & x_{56} \\ 0 & 0 & 0 & 0 & 0 & x_{66} \end{pmatrix} =$$

$$= -x_{13}x_{21}x_{34}x_{42}x_{55}x_{66} - x_{11}x_{22}x_{34}x_{43}x_{55}x_{66}.$$

Symbolic Adjacency Matrix

A symbolic adjacency matrix of the bipartite graph:

G



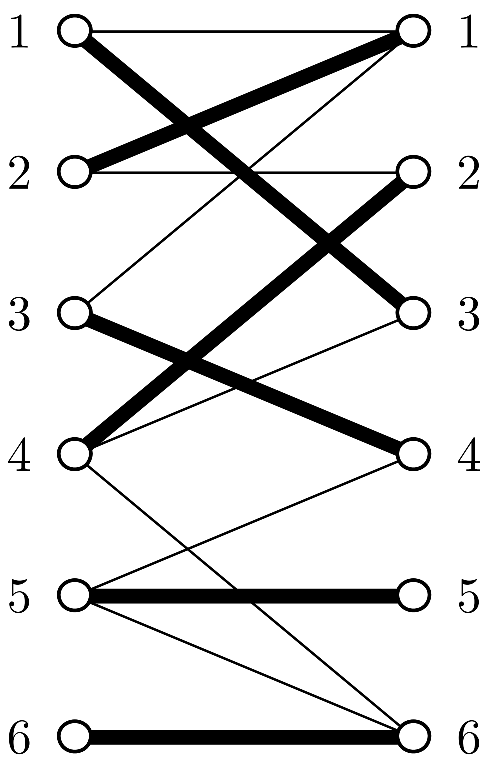
$$\Longrightarrow$$

$$\det(\tilde{B}(G)) =$$
$$-x_{13}x_{21}x_{34}x_{42}x_{55}x_{66}$$
$$-x_{11}x_{22}x_{34}x_{43}x_{55}x_{66}.$$

Symbolic Adjacency Matrix

A symbolic adjacency matrix of the bipartite graph:

G

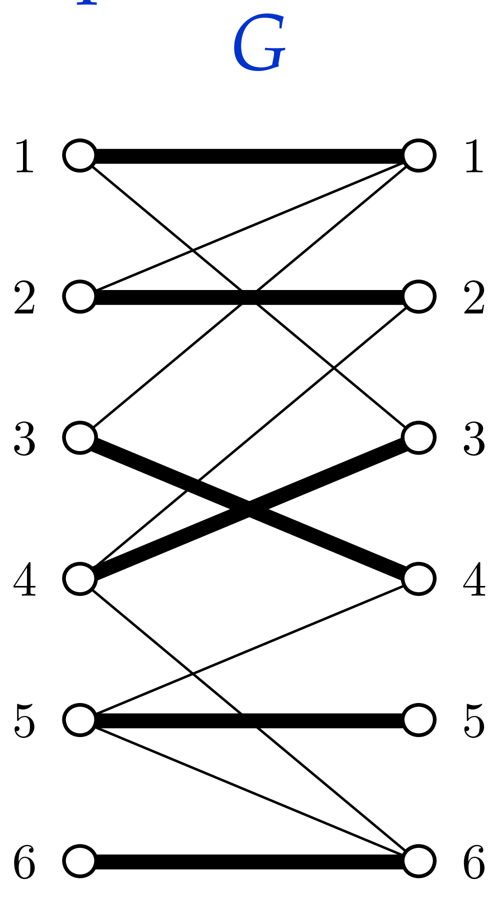


$\Longrightarrow$

$$\det(\tilde{B}(G)) =$$
$$\mathbf{-x_{13}x_{21}x_{34}x_{42}x_{55}x_{66}}$$
$$-x_{11}x_{22}x_{34}x_{43}x_{55}x_{66}.$$

Symbolic Adjacency Matrix

A symbolic adjacency matrix of the bipartite graph:



$$\Longrightarrow$$

$$\det(\tilde{B}(G)) =$$
$$-x_{13}x_{21}x_{34}x_{42}x_{55}x_{66}$$
$$\boldsymbol{-x_{11}x_{22}x_{34}x_{43}x_{55}x_{66}.}$$

Symbolic Adjacency Matrix

The determinant is given as:

$$\det(A) = \sum_{p \in \Pi_n} \sigma(p) \prod_{i=1}^{n} a_{i,p_i}.$$

p assigns different vertex p_i to each vertex i.

The elements of the sum correspond to perfect matchings in the graph.

The determinant is non-zero iff the graph has a perfect matching.

Lovász's Idea

The polynomial $\det(\tilde{B}(G))$ can have exponentially many terms. Can we efficiently test whether it is non-zero?

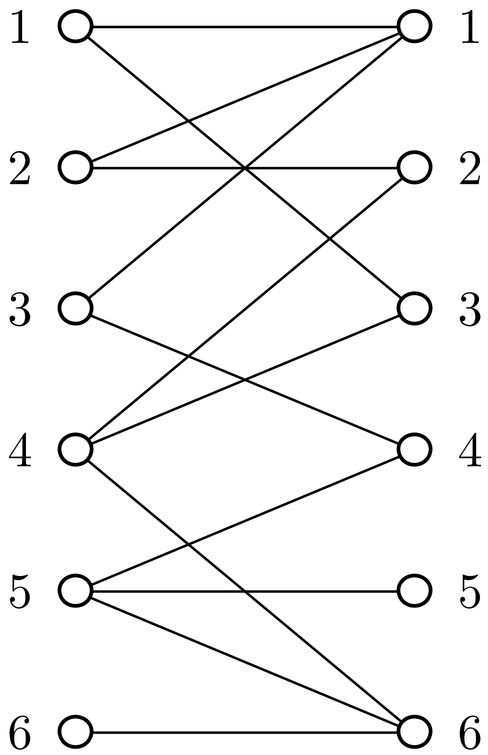
Substitute random numbers into variables in $\tilde{B}(G)$ and compute the determinant of the resulting matrix B — *random adjacency matrix*.

With high probability $\det B \neq 0$ iff $\det \tilde{B}(G) \neq 0$, because „polynomials do not have many zeros".

Random Adjacency Matrix

There is a perfect matchings.

G



$\Longrightarrow$

B

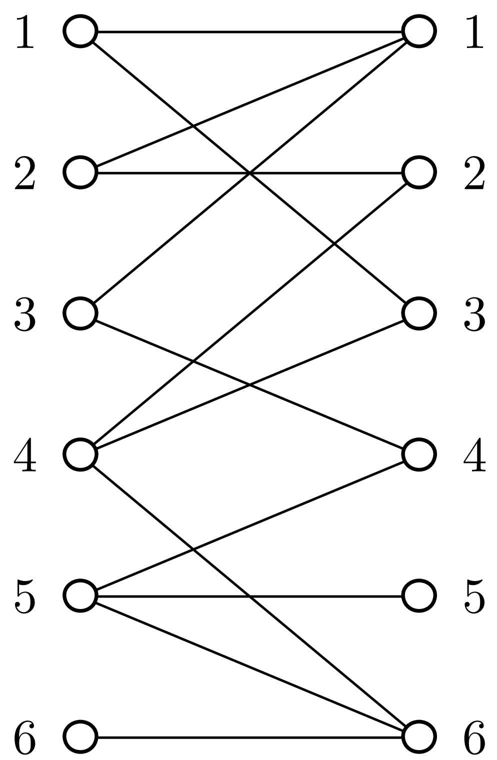
$$\begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\det(B) = 2$$

Random Adjacency Matrix

There is a perfect matchings.

G



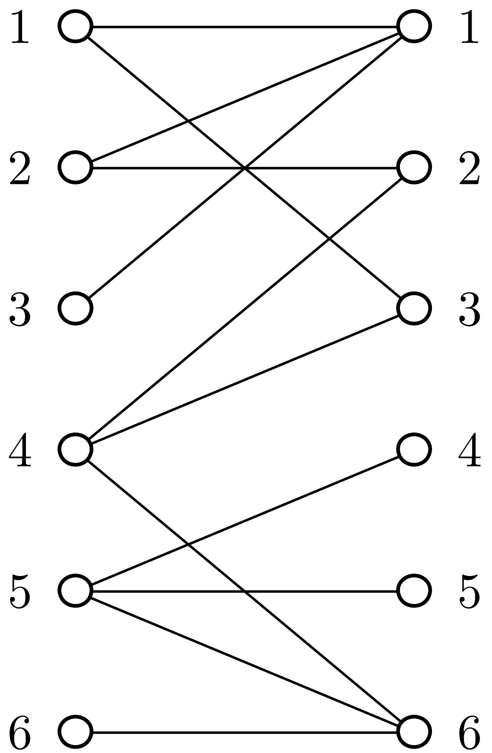
$$\Longrightarrow \quad B = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\det(B) = 0$$

Random Adjacency Matrix

There is no perfect matchings.

G



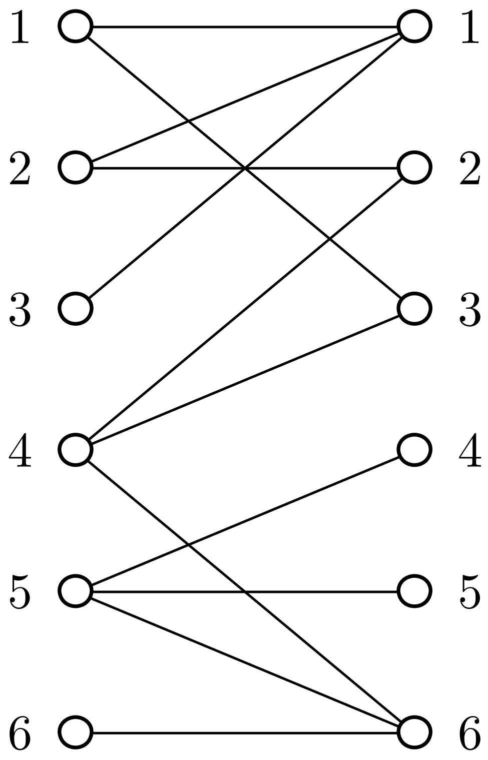
$$\Longrightarrow \quad B = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\det(B) = 0$$

Random Adjacency Matrix

There is no perfect matchings.

G



$\Longrightarrow$

B

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

$$\det(B) = 0$$

Zippel-Schwartz Lemma

Lemma 1 (Zippel, Schwartz) *Let $f(x_1, \ldots, x_k)$ be a degree n polynomial over the field F. Polynomial f has no more than $\frac{n}{|F|}|F|^k$ zeros.*

Let $F = \mathcal{Z}_p$ for some prime number $p = \Theta(n^{1+c})$, then the operations in Z_p can be performed in constant time.

The probability of a *false zero* – we get zero value for a non-zero polynomial – equals $O(\frac{1}{n^c})$.

Shortest Cycle Problem

We will study graphs $G = (V, E)$ with integer edge weights $w : E \to [-W, W]$ but without negative weight cylces.

In the *shortest cycle problem* we want to find the shortest cycle in a weighted graph G.

Directed and undirected problems are not equivalent.

When we bidirect an undirected graph new cycles appear, e.g., of length 2.

Shortest cycle problem

Complexity	Author
$O(nm + n^2 \log n)$ *dir.*	Johnson (1977)
$O(n^\omega)$ *nonnegative undir.*	Itai & Rodeh (1977)
$O(W^{0.681} n^{2.575})$ *dir.*	Zwick (2000)
$O(nm + n^2 \log\log n)$ *dir.*	Pettie (2004)
$O(n^3 \log^3 \log n / \log^2 n)$ *dir.*	Chan (2007)
$\tilde{O}(Wn^\omega)$ *dir. and nonnegative undir.*.	Roditty & Vassilevska-Williams (2011)
$\tilde{O}(Wn^\omega)$	Cygan, S., Gabow '12

Shortest Cycles: Idea

For directed graph $\overrightarrow{G} = (V, E)$ we define a symbolic $n \times n$ adjacency matrix $\tilde{A}(\overrightarrow{G})$ as

$$\tilde{A}(\overrightarrow{G})_{i,j} = \begin{cases} x_{i,j} & \text{if } (i,j) \in E, \\ 0 & \text{otherwise,} \end{cases}$$

where $x_{i,j}$ are unique variables.

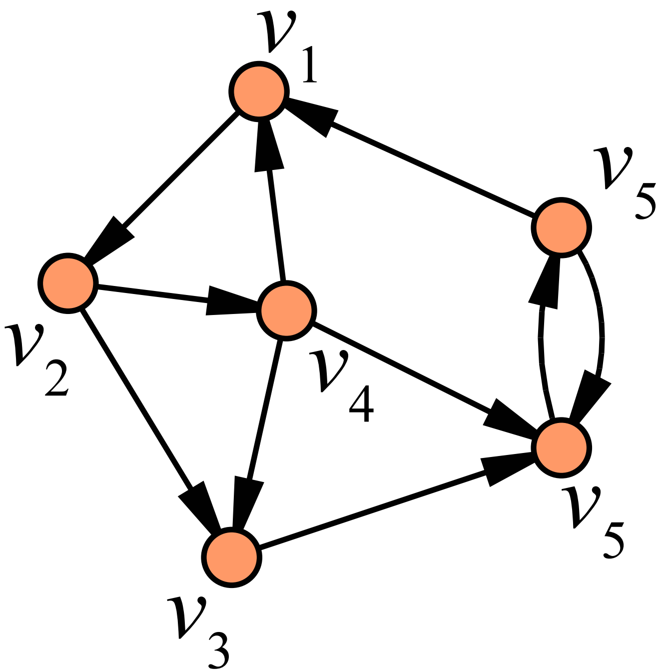
Theorem 2 *There exists a cycle in G if and only if*

$$\det\left(\tilde{A}(\overrightarrow{G}) + I\right) - 1 \neq 0.$$

Determinant

An example of the adjacency matrix:

$\overrightarrow{G}$

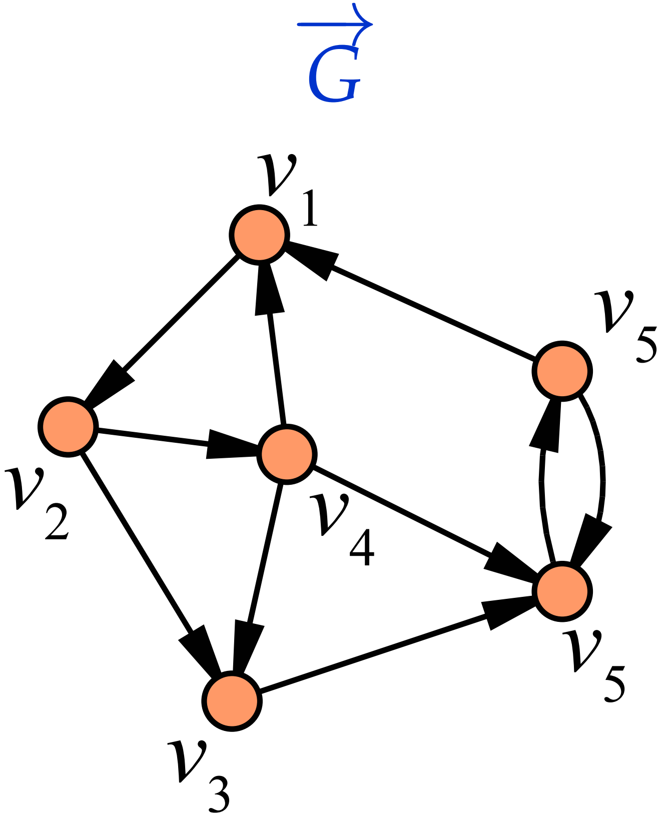


$$\Longrightarrow$$

$$\tilde{A}(\overrightarrow{G}) + I$$

$$\begin{pmatrix} 1 & x_{1,2} & 0 & 0 & 0 & 0 \\ 0 & 1 & x_{2,3} & x_{2,4} & 0 & 0 \\ 0 & 0 & 1 & 0 & x_{3,5} & 0 \\ x_{4,1} & 0 & x_{4,3} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & x_{5,6} \\ x_{6,1} & 0 & 0 & 0 & x_{6,5} & 1 \end{pmatrix}$$

Determinant



$\Longrightarrow$

$$\det(\tilde{A}(G)) = x_{1,2}x_{2,3}x_{3,5}x_{5,6}x_{6,1} +$$
$$-x_{1,2}x_{2,4}x_{4,1} - x_{1,2}x_{2,4}x_{4,5}x_{5,6}x_{6,1} +$$
$$+x_{1,2}x_{2,4}x_{4,3}x_{3,5}x_{5,6}x_{6,1} +$$
$$-x_{1,2}x_{2,4}x_{4,1}x_{5,6}x_{6,5} + x_{5,6}x_{6,5} + 1.$$

Terms of the determinant correspond to cycle packings in the graph.

Some Definitions

Let $\deg_y^*(p)$ be the term of p with the smallest degree in y:

$$\deg_y^*(y^{10} + 5y^4 + xy^3 + x^2) = 3.$$

Similarly, $\mathrm{term}_y^*(p)$ denotes term of degree $\deg_y^*(p)$.

Weights

For the directed graph $\overrightarrow{G} = (V, E)$ with weights $w : E \to [-W, W]$ we define the symbolic $n \times n$ adjacency matrix $\tilde{A}(\overrightarrow{G})$ as

$$\tilde{A}(\overrightarrow{G}, w)_{i,j} = \begin{cases} x_{i,j} y^{w(ij)} & \text{if } (i, j) \in E, \\ 0 & \text{otherwise,} \end{cases}$$

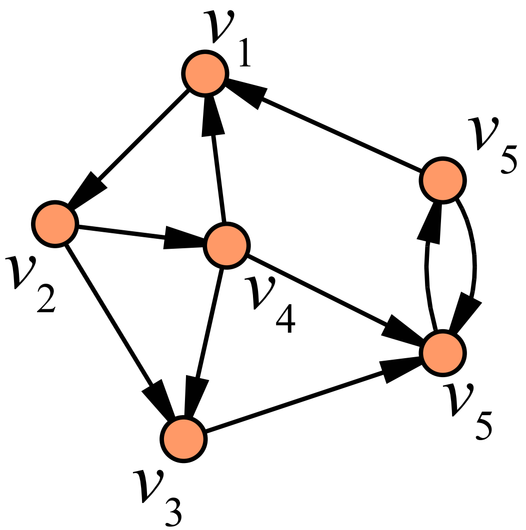
where $x_{i,j}$ are unique variables.

Theorem 3 *The weight of the shortest cycle in* $\overrightarrow{G}$ *is equal to* $\deg_y^* \left(\det \left(\tilde{A}(\overrightarrow{G}, w) + I \right) - 1 \right)$.

Determinant

An example of the adjacency matrix:

$\overrightarrow{G}$

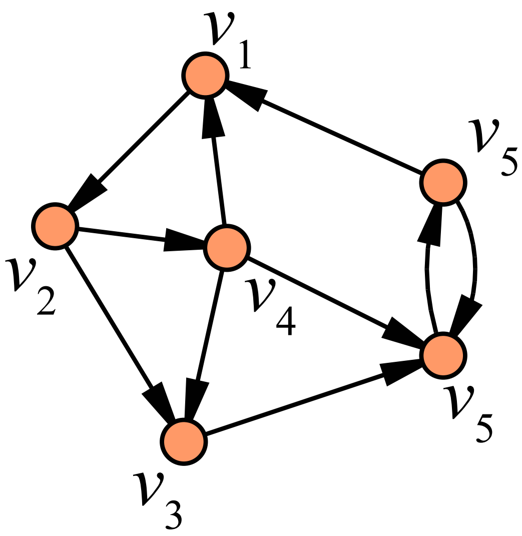


$$\tilde{A}(\overrightarrow{G}, w) + I$$

$$\Longrightarrow \begin{pmatrix} 1 & x_{1,2}y & 0 & 0 & 0 & 0 \\ 0 & 1 & x_{2,3}y & x_{2,4}y & 0 & 0 \\ 0 & 0 & 1 & 0 & x_{3,5}y & 0 \\ x_{4,1}y & 0 & x_{4,3}y & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & x_{5,6}y \\ x_{6,1}y & 0 & 0 & 0 & x_{6,5}y & 1 \end{pmatrix}$$

Determinant

$\overrightarrow{G}$



$\Longrightarrow$

$$\det(\tilde{A}(\overrightarrow{G}, w) + I) = x_{1,2}x_{2,3}x_{3,5}x_{5,6}x_{6,1}y^5 + \\ - x_{1,2}x_{2,4}x_{4,1}y^3 - x_{1,2}x_{2,4}x_{4,5}x_{5,6}x_{6,1}y^5 + \\ + x_{1,2}x_{2,4}x_{4,3}x_{3,5}x_{5,6}x_{6,1}y^6 + \\ - x_{1,2}x_{2,4}x_{4,1}x_{5,6}x_{6,5}y^5 + x_{5,6}x_{6,5}y^2 + 1.$$

We already know that the terms correspond to cycle packings.

Hence, the degree of y correspond to their weights.

Strojohann's Algorithm

Some of these problems can be solved for matrix polynomials as well.

Theorem 4 (Strojohann '03) *Let A be a matrix polynomial of degree W and size $n \times n$, let b be a vector polynomial of degree W and size n, then in $O(Wn^{\omega})$ time we can compute:*

- *determinant* $\det(A)$,
- *solve linear system of equations, i.e.,* $A^{-1}b$,

with high probability.

Some Problems

The matrix $\tilde{A}(\overrightarrow{G}, w) + I$ is a symbolic matrix — we cannot efficiently compute its determinant.

$\Rightarrow$ we can substitute random numbers for the variables.

The matrix $\tilde{A}(\overrightarrow{G}, w) + I$ is not a polynomial — we cannot apply Strojohann's theorem directly.

$\Rightarrow$ we can use

$$(\tilde{A}(\overrightarrow{G}, w) + I)y^W.$$

Algorithm for the Shortest Cycle

1: Substitute random numbers for variables in $\tilde{A}(G) + I$ to obtain A.
2: Compute $\delta = \det(Ay^W) - y^{nW}$ using Strojohann's theorem.
3: Return $\deg^*_y(\delta) - nW$.

Dynamic Functions

Let $f : \mathcal{R}^n \rightarrow \mathcal{R}^m$ be an n argument function returning m results.

A dynamic algorithm for f supports following operations:

- **initialization($x_1, \ldots, x_n$):** set the input vector to $(x_1, \ldots, x_n)$,
- **update(k,x'_k):** change the k-th input to x'_k,
- **query(k):** return the k-th result.

We will consider the problems of dynamically computing the determinant, the inverse matrix and the matrix rank.

Dynamic Matrix Functions

We are given the matrix:

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} \qquad \det(A) = -2$$

Dynamic Matrix Functions

We are given the matrix:

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} \qquad \det(A) = -2$$

After the change:

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} \qquad \det(A) = -4$$

Dynamic Matrix Functions

We are given the matrix:

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} \qquad \det(A) = -2$$

After the change:

$$A = \begin{bmatrix} 2 & 1 & 2 \\ 2 & 2 & 2 \\ 1 & 2 & 2 \end{bmatrix} \qquad \det(A) = 2$$

Dynamic Matrix Inverse

Theorem 5 (Sherman and Morrison '49)
The problem of dynamically computing:

- *the determinant,*
- *the inverse matrix,*

for non-singular column updates can be solved with the following costs:

- **initialization:** $O(n^\omega)$ *time,*
- **update:** $O(n^2)$ *time,*
- **query:** $O(1)$ *time.*

Dynamic Matrix Functions

We are given the matrix:

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} \qquad \det(A) = -2$$

Dynamic Matrix Functions

We are given the matrix:

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} \qquad \det(A) = -2$$

After the change:

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 2 & 2 \\ 0 & 2 & 2 \end{bmatrix} \qquad \det(A) = 0$$

Dynamic Matrix Functions

We are given the matrix:

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} \qquad \det(A) = -2$$

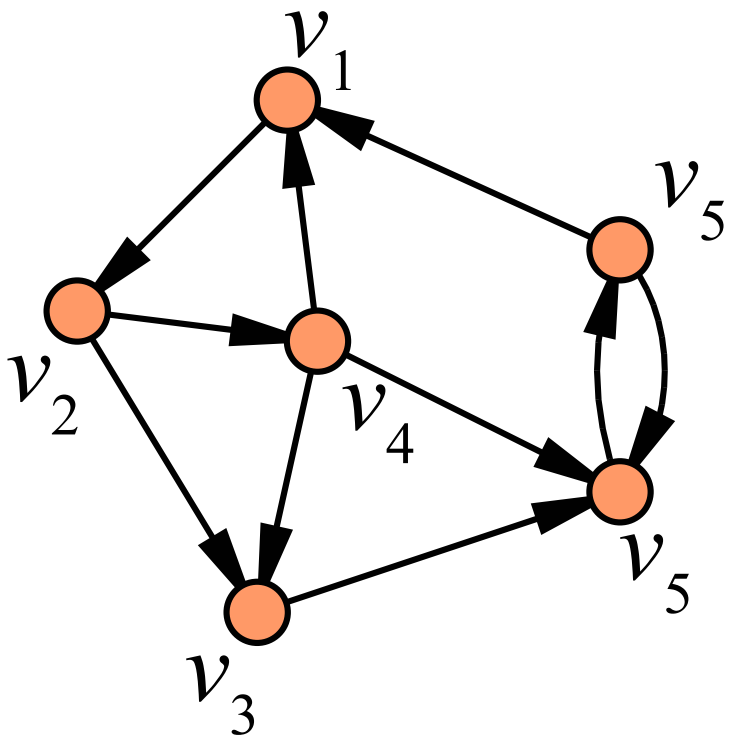
After the change:

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 2 & 2 \\ 0 & 2 & 2 \end{bmatrix} \qquad \det(A) = 0$$

Algorithm returns FAILURE.

Dynamic Transitive Closure

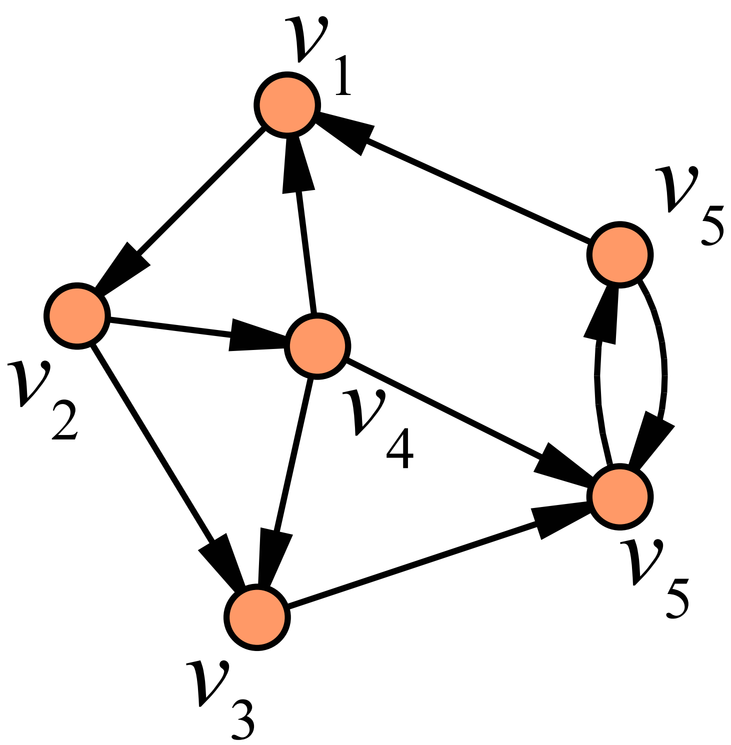
For a given graph:



Is there a path from v_1 to v_4?

Dynamic Transitive Closure

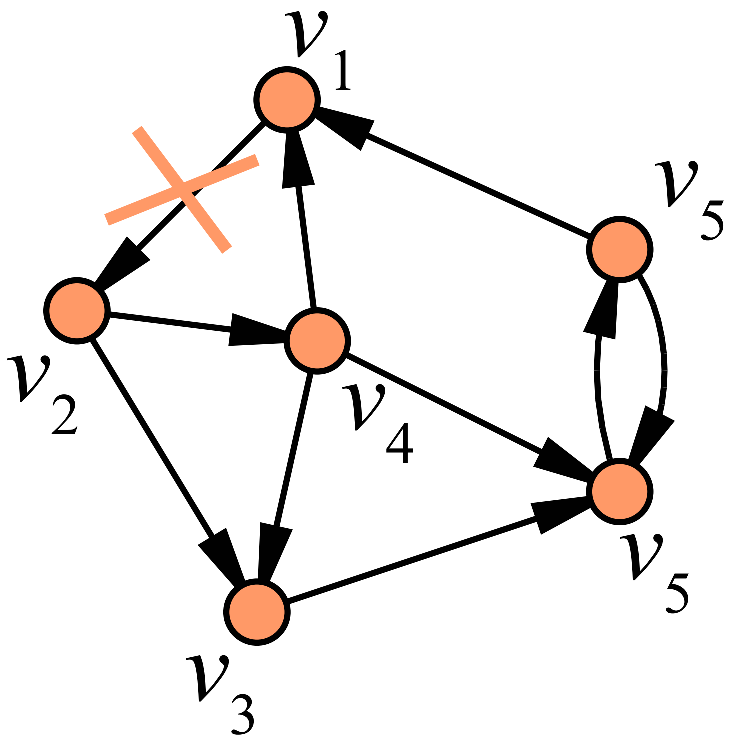
For a given graph:



Is there a path from v_1 to v_4? YES

Dynamic Transitive Closure

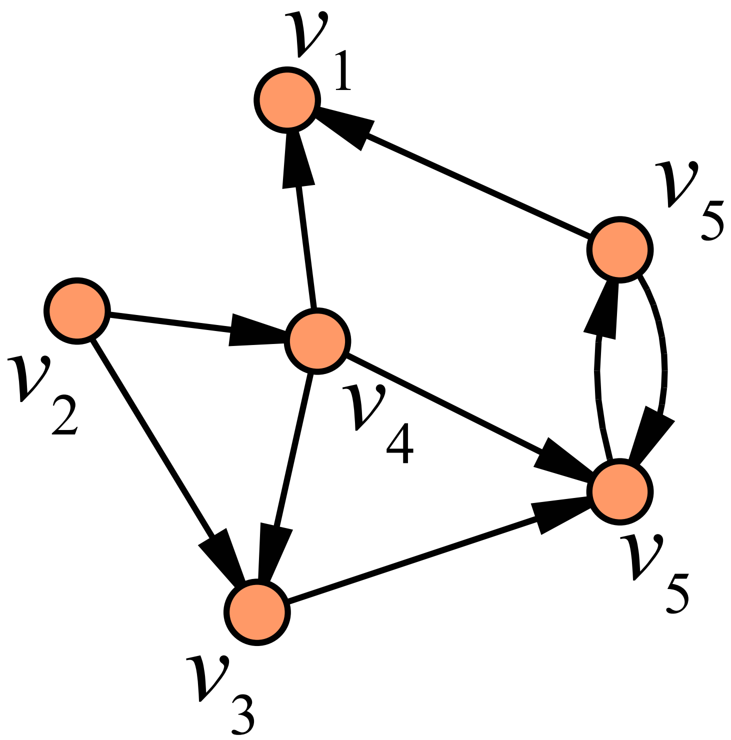
For a given graph:



Is there a path from v_1 to v_4?

Dynamic Transitive Closure

For a given graph:



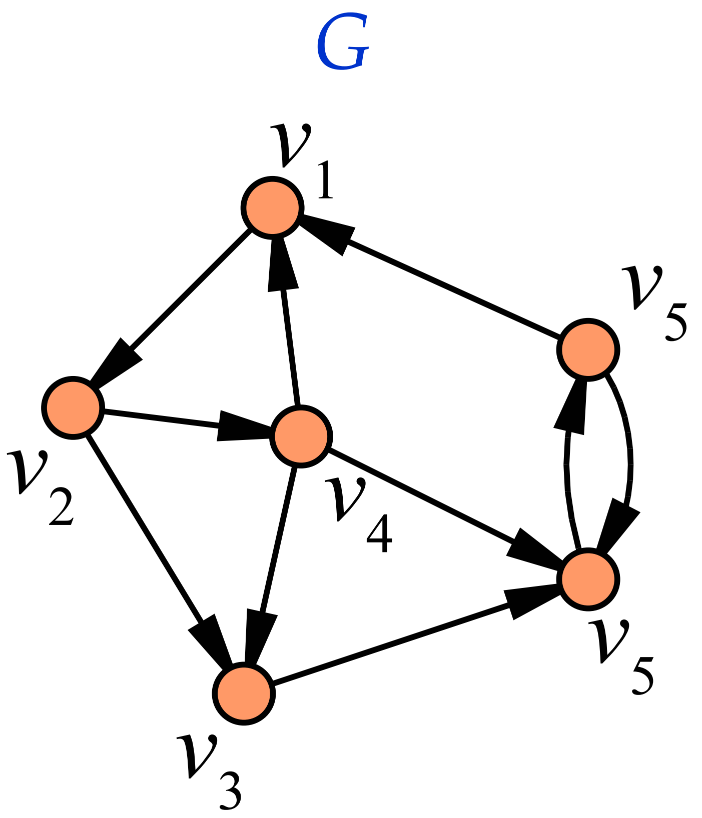
Is there a path from v_1 to v_4?

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	$O(m + n \log n)$	$O(n)$
S. '04 (worst-case but randomized)	$O(n^2)$	$O(1)$

Symbolic Adjacency Matrix

Symbolic adjacency matrix of the graph:



$$\Longrightarrow \quad \tilde{A}(\overrightarrow{G}) + I$$

$$\begin{pmatrix} 1 & x_{1,2} & 0 & 0 & 0 & 0 \\ 0 & 1 & x_{2,3} & x_{2,4} & 0 & 0 \\ 0 & 0 & 1 & 0 & x_{3,5} & 0 \\ x_{4,1} & 0 & x_{4,3} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & x_{5,6} \\ x_{6,1} & 0 & 0 & 0 & x_{6,5} & 1 \end{pmatrix}$$

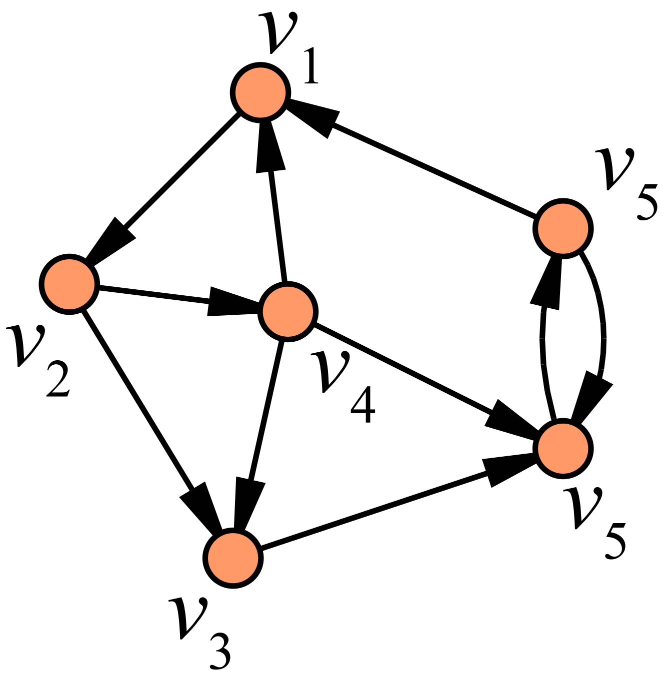
Symbolic Adjacency Matrix

Let us compute $\text{adj}(A)_{1,3} = \det(A^{3,1})$.

$$A \qquad\qquad\qquad\qquad A^{3,1}$$

$$\left(\begin{array}{c|ccccc} 1 & x_{1,2} & 0 & 0 & 0 & 0 \\ 0 & 1 & x_{2,3} & x_{2,4} & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & x_{3,5} & 0 \\ \hline x_{4,1} & 0 & x_{4,3} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & x_{5,6} \\ x_{6,1} & 0 & 0 & 0 & x_{6,5} & 1 \end{array}\right) \Longrightarrow \left(\begin{array}{c|ccccc} 0 & x_{1,2} & 0 & 0 & 0 & 0 \\ 0 & 1 & x_{2,3} & x_{2,4} & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & x_{4,3} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & x_{5,6} \\ 0 & 0 & 0 & 0 & x_{6,5} & 1 \end{array}\right)$$

Symbolic Adjacency Matrix



$$\det \begin{bmatrix} 0 & x_{1,2} & 0 & 0 & 0 & 0 \\ 0 & 1 & x_{2,3} & x_{2,4} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & x_{4,3} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & x_{5,6} \\ 0 & 0 & 0 & 0 & x_{6,5} & 1 \end{bmatrix} =$$

$$= x_{1,2}x_{2,3} - x_{1,2}x_{2,4}x_{4,3} +$$

$$- x_{1,2}x_{2,3}x_{5,6}x_{6,5} + x_{1,2}x_{2,4}x_{4,3}x_{5,6}x_{6,5}.$$

The monomials of the determinant correspond to paths from v_1 to v_3 in G.

Dynamic Transitive Closure

Theorem 6 (S. '04) *Let $\tilde{A}(\overrightarrow{G})$ be a symbolic adjacency matrix of $\overrightarrow{G}$, substitute random numbers into variables in obtaining the matrix A:*

- *there is a path from i to j in $\overrightarrow{G}$ iff $(\tilde{A}(\overrightarrow{G}) + I)^{-1}_{ij}$ is non-zero (with high probability).*

This allows us to compute the transitive closure by inverting the matrix once — can be easily used in the dynamic case.

Transitive Closure

Theorem 7 (S. '04)

Dynamic matrix inverse

Update in $O(n^\alpha)$ time

Query in $O(n^\beta)$ time

can assume nonsingularity $\Downarrow$

$\Downarrow$ Dynamic transitive closure

Update in $O(n^\alpha)$ time

Query in $O(n^\beta)$ time

randomized with one sided error

Algorithm for Transitive Closure

- Generate random adjacency matrix A from the adjacency matrix $\tilde{A}(\overrightarrow{G}) + I$ by substituting $x_{i,j}$ with a random numbers from Z_p.
- compute the adjoint of the matrix A

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

- with high probability $\mathrm{adj}(A)_{i,j} \neq 0$ iff there is a path from i to j in $\overrightarrow{G}$.

The algorithm works in $O(n^{\omega})$ time.

Single Source Shortest Paths

For a weighted directed graph $G = (V, E)$, where $w : E \to \{-W, \ldots, 0, \ldots, W\}$ is the edge weight function, we denote by $\mathrm{dist}_G(i, j)$ the distance from i to j.

For given source s we want to find distances from s to all other nodes in G, or detect negative length cycle.

Single Source Shortest Paths

Complexity	Author
$O(n^4)$	Shimbel (1955)
$O(n^2 mW)$	Ford (1956)
$\boldsymbol{O(nm)}$	Bellman (1958), Moore (1959)
$O(n^{\frac{3}{4}} m \log W)$	Gabow (1983)
$O(\sqrt{n} m \log(nW))$	Gabow and Tarjan (1989)
$\boldsymbol{O(\sqrt{n} m \log(W))}$	Goldberg (1993)
$\boldsymbol{O(n^{2.38} W)}$	S. '05 and Yuster and Zwick '05

The Idea — Weighted Case

Theorem 8

$$\mathrm{dist}_G(i,j) = \deg_u^* \left(\mathrm{adj}\left(\tilde{A}(\overrightarrow{G},w) + I\right)_{i,j} \right).$$

Corollary 9 *Let G be a directed weighted graph without negative length cycles then*

$$\mathrm{dist}_G(i,j) = \deg_y^* \left(\mathrm{adj}\left((\tilde{A}(\overrightarrow{G},w) + I)y^W\right)_{i,j} \right) - (n-1)W.$$

Algorithm

- Generate random adjacency matrix A from the adjacency matrix $(\tilde{A}(G) + I)y^W$ by substituting $x_{i,j}$ with a random numbers from Z_p.
- Compute $\det(A^T)$ and $\left(A^T\right)^{-1} e_i$ with Storjohann's Algorithm,
- With high probability

$$\mathrm{dist}_G(i,j) = \deg_y^* \left(\left(\det(A^T) \left(A^T\right)^{-1} e_i \right)_j \right),$$

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

Conclusions

The algebraic techniques can be used to construct the asymptotically fastest algorithms for:

- dynamic transitive closure,
- dynamic distances in graphs,
- dynamic vertex connectivity,
- dynamic maximum matchings,
- maximum matchings in graphs,
- maximum weighted matchings in graphs.