All-Pairs shortest paths via fast matrix multiplication

Uri Zwick
Tel Aviv University

Otwarte wykłady dla doktorantów informatyki
Instytut Informatyki
Uniwersytetu Warszawskiego

Marzec 9-10, 2007

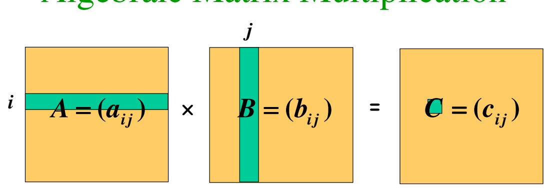
Outline

1. **Algebraic matrix multiplication**
   a. Strassen's algorithm
   b. Rectangular matrix multiplication
2. **Boolean matrix multiplication**
   a. Simple reduction to integer matrix multiplication
   b. Computing the transitive closure of a graph.
3. **Min-Plus matrix multiplication**
   a. Equivalence to the APSP problem
   b. Expensive reduction to algebraic products
   c. Fredman's trick

4. **APSP in undirected graphs**
   a. An $O(n^{2.38})$ algorithm for **unweighted** graphs (Seidel)
   b. An $O(Mn^{2.38})$ algorithm for **weighted** graphs (Shoshan-Zwick)
5. **APSP in directed graphs**
   1. An $O(M^{0.68}n^{2.58})$ algorithm (Zwick)
   2. An $O(Mn^{2.38})$ preprocessing / $O(n)$ query answering algorithm (Yuster-Zwick)
   3. An $O(n^{2.38}\log M)$ $(1+\varepsilon)$-approximation algorithm
6. **Summary and open problems**

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)**
$n^{2.38}$	**Coppersmith, Winograd (1990)**

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

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

$T(n) = O(n^{\log 8/\log 2}) = O(n^3)$

Strassen's 2×2 algorithm

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

$$\begin{aligned} C_{11} &= M_1 + M_4 - M_5 + M_7 \\ C_{12} &= M_3 + M_5 \\ C_{21} &= M_2 + M_4 \\ C_{22} &= M_1 - M_2 + M_3 + M_6 \end{aligned}$$

$M_1 = (A$ … Subtraction!
$M_2 = (A_{21} +$ … B_{11}

$$\begin{aligned} M_3 &= A_{11}(B_{12} - B_{22}) \\ M_4 &= A_{22}(B_{21} - B_{11}) \\ M_5 &= (A_{11} + A_{12})B_{22} \\ M_6 &= (A_{21} - A_{11})(B_{11} + B_{12}) \\ M_7 &= (A_{12} - A_{22})(B_{21} + B_{22}) \end{aligned}$$

7 multiplications
18 additions/subtractions

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.

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

$T(n) = O(n^{\log 7/\log 2}) = O(n^{2.81})$

Works over any ring!

Matrix multiplication algorithms

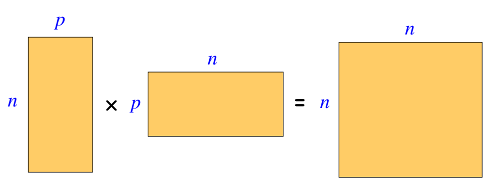
The $O(n^{2.81})$ bound of Strassen was improved by Pan, Bini-Capovani-Lotti-Romani, Schönhage and finally by Coppersmith and Winograd to $O(n^{2.38})$.

The algorithms are much more complicated…

We let $2 \le \omega < 2.38$ be the exponent of matrix multiplication.

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

Rectangular Matrix multiplication



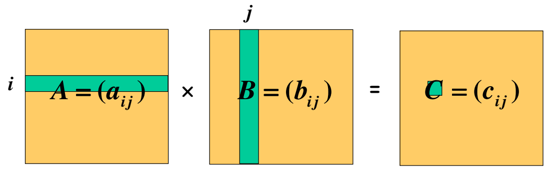
Naïve complexity: $n^2 p$

[Coppersmith '97]: $n^{1.85} p^{0.54} + n^{2+o(1)}$

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

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

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

$O(n^{2.38})$ algebraic operations

Boolean Product

$$C = A \cdot B$$
$$c_{ij} = \bigvee_k a_{ik} \wedge b_{kj}$$

?

$O(n^{2.38})$

or (∨) has no inverse!

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

operations on $O(\log n)$ bit words

Transitive Closure

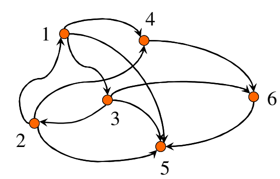
Let $G=(V,E)$ be a directed graph.

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.

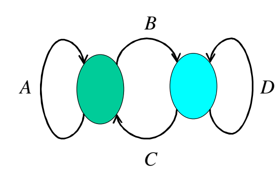
The transitive closure $G^*=(V,E^*)$ is the graph in which $(u,v)\in E^*$ iff there is a path from u to v.

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.

$$X = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$



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

$$TC(n) \le 2\, TC(n/2) + 6\, BMM(n/2) + O(n^2)$$

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

a) a triangle

b) a simple quadrangle

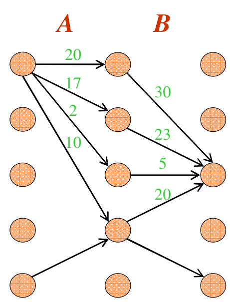
c) a simple cycle of length k.

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

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.

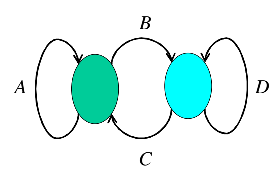
By induction, W^k gives the distances realized by paths that use at most k edges.

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

Thus: $\text{APSP}(n) \le \text{MPP}(n) \log n$

Actually: $\text{APSP}(n) = O(\text{MPP}(n))$

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



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

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

Algebraic Product	Min-Plus Product
$C = A \cdot B$	$C = A * B$
$c_{ij} = \sum_k a_{ik} b_{kj}$	$c_{ij} = \min_k \{a_{ik} + b_{kj}\}$
$O(n^{2.38})$	min operation has no inverse! ?

Using matrix multiplication to compute min-plus products

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

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

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

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

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 × M operations per polynomial product = Mn^{ω} operations per max-plus product

SHORTEST PATHS

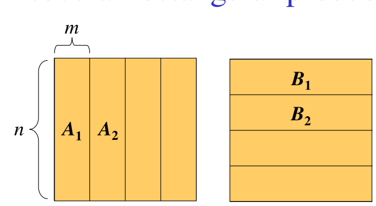
APSP – All-Pairs Shortest Paths

SSSP – Single-Source Shortest Paths

Fredman's trick

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

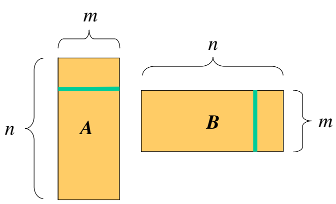
Breaking a square product into several rectangular products



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

MPP(n) ≤ (n/m) (MPP(n,m,n) + n^2)

Fredman's trick



$a_{ir}+b_{rj} \leq a_{is}+b_{sj}$

$\Updownarrow$

$a_{ir} - a_{is} \leq b_{sj} - b_{rj}$

Naïve calculation requires n^2m operations

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

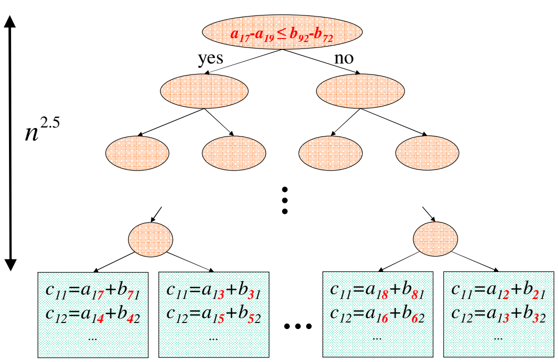
Fredman's trick (cont.)

$a_{ir}+b_{rj} \leq a_{is}+b_{sj} \quad \Leftrightarrow \quad a_{ir} - a_{is} \leq b_{sj} - b_{rj}$

- **Generate** all the differences $a_{ir} - a_{is}$ and $b_{sj} - b_{rj}$.
- **Sort** them using $O(nm^2)$ comparisons. (Non-trivial!)
- **Merge** the two sorted lists using $O(nm^2)$ comparisons.

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

Decision Tree Complexity



All-Pairs Shortest Paths
in directed graphs with "real" edge weights

Running time	Authors
n^3	[Floyd '62] [Warshall '62]
$n^3 (\log\log n / \log n)^{1/3}$	[Fredman '76]
$n^3 (\log\log n / \log n)^{1/2}$	[Takaoka '92]
$n^3 / (\log n)^{1/2}$	[Dobosiewicz '90]
$n^3 (\log\log n / \log n)^{5/7}$	[Han '04]
$n^3 \log\log n / \log n$	[Takaoka '04]
$n^3 (\log\log n)^{1/2} / \log n$	[Zwick '04]
$n^3 / \log n$	[Chan '05]
$n^3 (\log\log n / \log n)^{5/4}$	[Han '06]
$n^3 (\log\log n)^3 / (\log n)^2$	[Chan '07]

UNWEIGHTED UNDIRECTED SHORTEST PATHS

4. **APSP in undirected graphs**
   a. An $O(n^{2.38})$ algorithm for **unweighted** graphs (Seidel)
   b. An $O(Mn^{2.38})$ algorithm for **weighted** graphs (Shoshan-Zwick)
5. **APSP in directed graphs**
   1. An $O(M^{0.68}n^{2.58})$ algorithm (Zwick)
   2. An $O(Mn^{2.38})$ preprocessing / $O(n)$ query answering algorithm (Yuster-Zwick)
   3. An $O(n^{2.38}\log M)$ $(1+\varepsilon)$-approximation algorithm
6. **Summary and open problems**

Directed versus undirected graphs

$\delta(x,z) \le \delta(x,y) + \delta(y,z)$

Triangle inequality

$\delta(x,z) \le \delta(x,y) + \delta(y,z)$

$\delta(x,y) \le \delta(x,z) + \delta(z,y)$

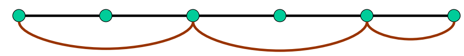
$\delta(x,z) \ge \delta(x,y) - \delta(y,z)$

Inverse triangle inequality

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.



$\delta(u,v)=5$ $\delta^2(u,v)=3$

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

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

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

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

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

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

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

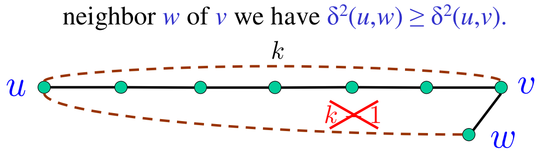
Lemma: If $\delta(u,v)=2\delta^2(u,v)-1$ then for every neighbor w of v we have $\delta^2(u,w) \le \delta^2(u,v)$ and for at least one neighbor $\delta^2(u,w) < \delta^2(u,v)$.

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

$$\sum_{(v,w)\in E} c_{u,w} = \sum_{w} c_{u,w} a_{w,v} = (CA)_{u,v} \quad : \quad \deg(v) c_{u,v}$$

Even distances

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



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

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

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 $\delta^2(u,w) < \delta^2(u,v)$.

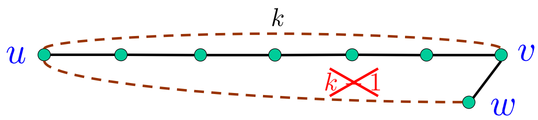
Exercise 2: Prove the lemma.

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

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

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

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

Seide

Assume that ***A*** has 1's on the diagonal.

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**.

Boolean matrix multiplicaion

A)

else
$C \leftarrow \mathrm{APD}(A^2)$
$X \leftarrow CA$, deg$\leftarrow A$e–1
deg$_j$]

Integer matrix multiplicaion

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

Exercise 3: (*) 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

If $c_{ij} = 0$ then $w_{ij} = 0$
If $c_{ij} = 1$ then $w_{ij} = k$ 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 4:

a) Obtain a deterministic $O(n^\omega)$-time algorithm for finding **unique** witnesses.
b) Let $1 \leq d \leq n$ 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 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 5:
Obtain an $O(n^{\omega}\log n)$ time algorithm for computing the **diameter** of 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 × M operations per polynomial product = Mn^{ω} operations per max-plus product

Trying to implement the repeated squaring algorithm

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

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 '98)

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

Choose a subset of V of size $(9n \ln n)/s$

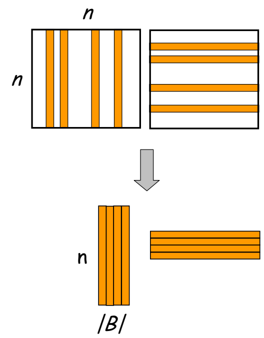
Select the **columns** of D whose indices are in B

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 $n \ln n / s$, where $s = (3/2)^{i+1}$

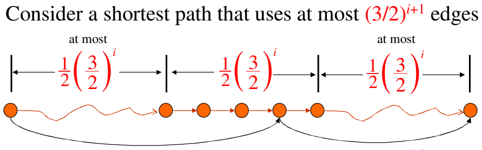
The matrices get smaller and smaller but the elements get larger and larger

Sampled Repeated Squaring - Correctness

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

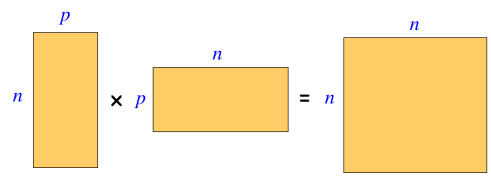
Invariant: After the i-th iteration, distances that are attained using at most $(3/2)^i$ edges are correct.

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



Let $s = (3/2)^{i+1}$ Failure probability : $\left(1-\frac{9\ln n}{s}\right)^{s/3} < n^{-3}$

Rectangular Matrix multiplication



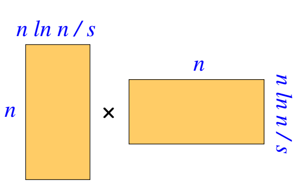
Naïve complexity: n^2p

[Coppersmith '97]: $n^{1.85}p^{0.54}+n^{2+o(1)}$

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

Complexity of APSP algorithm

The i-th iteration:



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

The elements are of absolute value at most Ms

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

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

Related result: [Yuster-Zwick'04]
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.

The corollary obtained using a different technique by Sankowski (2004)

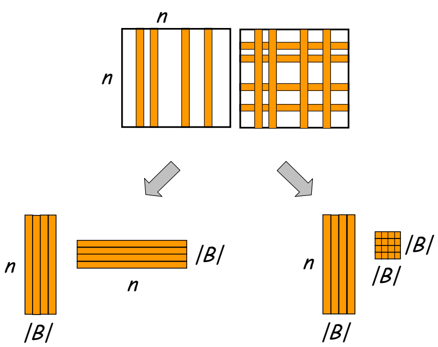
The preprocessing algorithm (YZ ’05)

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

The APSP algorithm

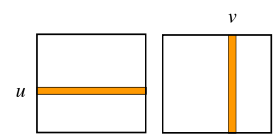
```
D ← W
for i ← 1 to log_{3/2} n do
{
    s ← (3/2)^{i+1}
    B ← rand(V,(9n ln n)/s)
    D ← min{ D , D[V,B]*D[B,V] }
}
```

Twice Sampled Distance Products



The query answering algorithm

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



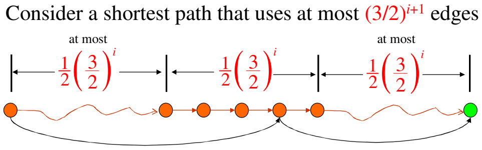
Query time: $O(n)$

The preprocessing algorithm: Correctness

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

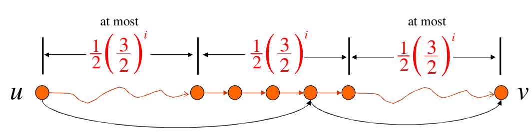
Invariant: After the i-th iteration, if $u \in B_i$ or $v \in B_i$ and there is a shortest path from u to v that uses at most $(3/2)^i$ edges, then $D(u,v)=\delta(u,v)$.

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



The query answering algorithm: Correctness

Suppose that the shortest path from u to v uses between $(3/2)^i$ and $(3/2)^{i+1}$ edges



1. **Algebraic matrix multiplication**
   a. Strassen's algorithm
   b. Rectangular matrix multiplication
2. **Min-Plus matrix multiplication**
   a. Equivalence to the APSP problem
   b. Expensive reduction to algebraic products
   c. Fredman's trick
3. **APSP in undirected graphs**
   a. An $O(n^{2.38})$ anlgorithm for unweighted graphs (Seidel)
   b. An $O(Mn^{2.38})$ algorithm for weighted graphs (Shoshan-Zwick)
4. **APSP in directed graphs**
   1. An $O(M^{0.68}n^{2.58})$ algorithm (Zwick)
   2. An $O(Mn^{2.38})$ preprocessing / $O(n)$ query answering alg. (Yuster-Z)
   3. An $O(n^{2.38}\log M)$ (1+ε)-approximation algorithm
5. **Summary and open problems**

Approximate min-plus products

Obvious idea: scaling

SCALE(A,M,R): $a'_{ij} \leftarrow \begin{cases} \lceil R a_{ij} / M \rceil , & \text{if } 0 \le a_{ij} \le M \\ +\infty , & \text{otherwise} \end{cases}$

APX-MPP(A,B,M,R) :
A'←SCALE(A,M,R)
B'←SCALE(B,M,R)
return MPP(A',B')

Complexity is $Rn^{2.38}$, instead of $Mn^{2.38}$, but small values can be greatly distorted.

Addaptive Scaling

APX-MPP(A,B,M,R) :

C'←∞
for r←$\log_2 R$ to $\log_2 M$ do
  A'←SCALE(A,2^r,R)
  B'←SCALE(B,2^r,R)
  C'←min{C',MPP(A',B')}
end

Complexity is $Rn^{2.38} \log M$
Stretch at most $1+4/R$

1. **Algebraic matrix multiplication**
   a. Strassen's algorithm
   b. Rectangular matrix multiplication
2. **Min-Plus matrix multiplication**
   a. Equivalence to the APSP problem
   b. Expensive reduction to algebraic products
   c. Fredman's trick
3. **APSP in undirected graphs**
   a. An $O(n^{2.38})$ anlgorithm for unweighted graphs (Seidel)
   b. An $O(Mn^{2.38})$ algorithm for weighted graphs (Shoshan-Zwick)
4. **APSP in directed graphs**
   1. An $O(M^{0.68}n^{2.58})$ algorithm (Zwick)
   2. An $O(Mn^{2.38})$ preprocessing / $O(n)$ query answering alg. (Yuster-Z)
   3. An $O(n^{2.38}\log M)$ (1+ε)-approximation algorithm
5. **Summary and open problems**

All-Pairs Shortest Paths
in graphs with small integer weights

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

Running time	Authors
$Mn^{2.38}$	[Shoshan-Zwick '99]

Improves results of
[Alon-Galil-Margalit '91] [Seidel '95]

All-Pairs Shortest Paths
in graphs with small integer weights

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

Running time	Authors
$M^{0.68} n^{2.58}$	[Zwick '98]

Improves results of
[Alon-Galil-Margalit '91] [Takaoka '98]

Answering distance queries

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

Preprocessing time	Query time	Authors
$Mn^{2.38}$	n	[Yuster-Zwick '05]

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
$(n^{2.38} \log M)/\varepsilon$	[Zwick '98]

Open problems

- An $O(n^{2.38})$ 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 $\{0,\pm1, \pm2,\ldots, \pm n\}$?