All-pair Shortest path via fast matrix multiplication

Ran Duan

ALL-PAIR SHORTEST PATH

- Run the Dijkstra's algorithm from every vertex
  - Running time: $O(mn+n^2 \log n)$
- Floyd-Warshall algorithm
  - Running time: $O(n^3)$

```
FloydWarshall()
    For k=1 to n do
        For i=1 to n do
            For j=1 to n do
                d(i,j)=min{d(i,j), d(i,k)+d(k,j)}
```

ALL-PAIR SHORTEST PATH

- Run the Dijkstra's algorithm from every vertex
  - Running time: $O(mn+n^2 \log n)$
- Floyd-Warshall algorithm
  - Running time: $O(n^3)$

- There is no truly sub-cubic algorithm for real-weighted APSP
  - Major open problem in graph theory

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]

In this talk…

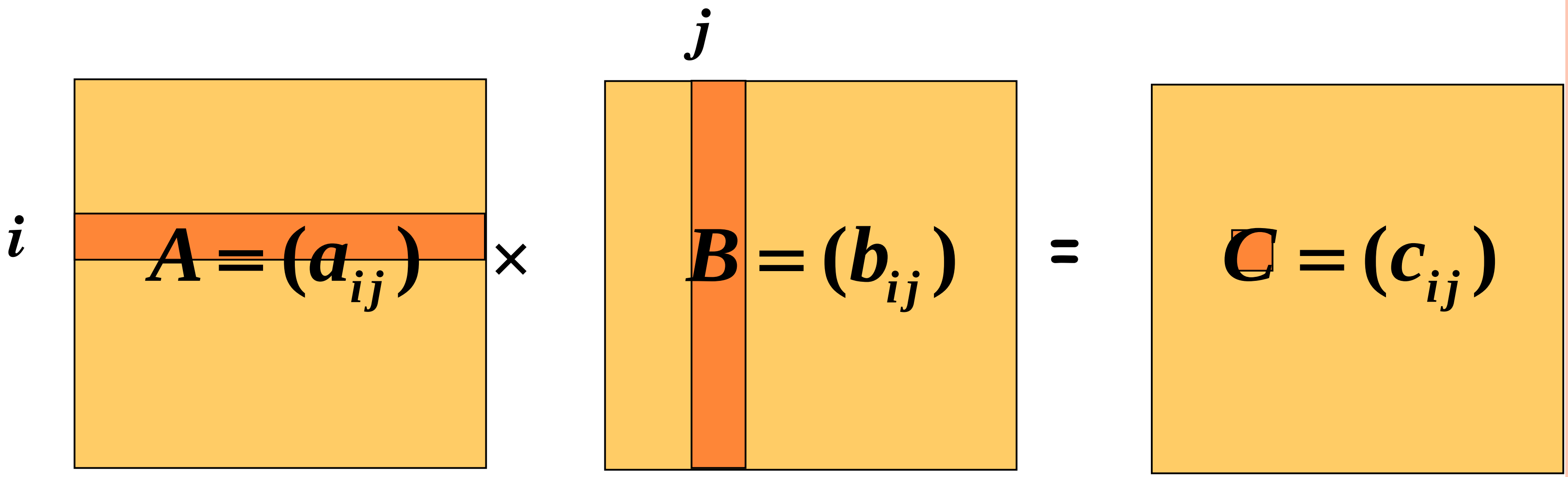
- We will use fast matrix multiplication algorithm to get $o(n^3)$ all-pair shortest path for small integer weights.

- The time for fast matrix multiplication is $O(n^\omega)$, $\omega=2.373$ at present
  - Improved by V. Williams this year from the well-known Coppersmith-Winograd bound of 2.376
  - We still use 2.376 bound in this talk.

Outline

- Algebraic matrix multiplication
- Transitive closure in $O(n^{\omega})$ time
- APSP in undirected unweighted graphs in $O(n^{\omega})$ time.
- APSP in directed graphs
  - Time: $O(M^{0.68}n^{2.58})$ for integer weighted [1..M] graphs
  - Min-plus product for matrices

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

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

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

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

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

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

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

Subtraction!

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

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

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

7 multiplications
18 additions/subtractions

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

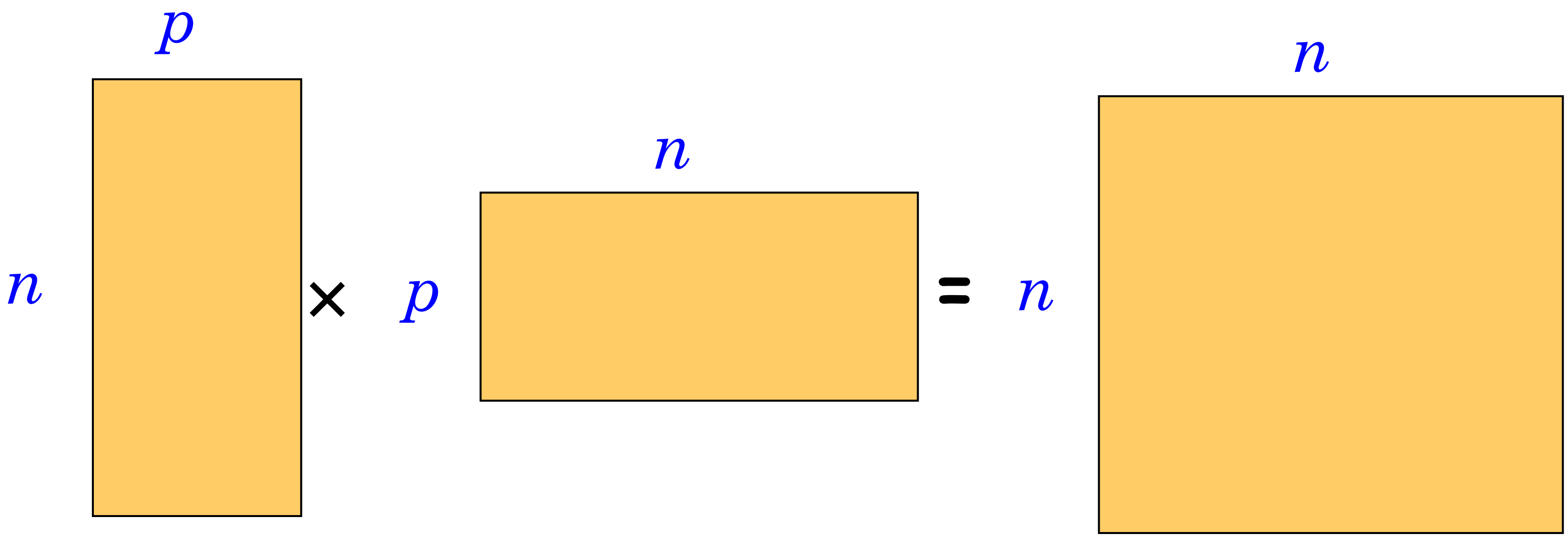
The $O(n^{2.81})$ bound of Strassen was improved by Pan, Bini-Capovani-Lotti-Romani, Schönhage and finally by Coppersmith and Winograd to $O(n^{2.376})$.

The algorithms are much more complicated…

We let $2 \leq \omega < 2.376$ be the exponent of matrix multiplication.

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

Rectangular Matrix multiplication

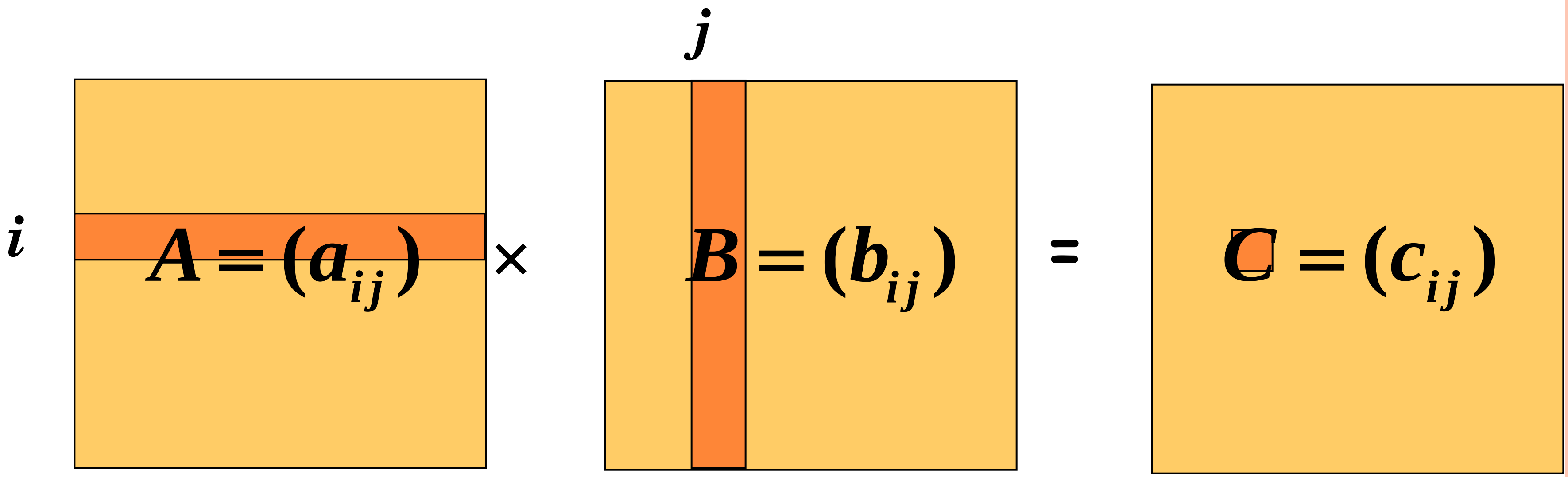


Naïve complexity: $n^2 p$

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

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

Boolean Matrix Multiplication



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

Can be also computed in O(n^{ω}) time.

Transitive Closure

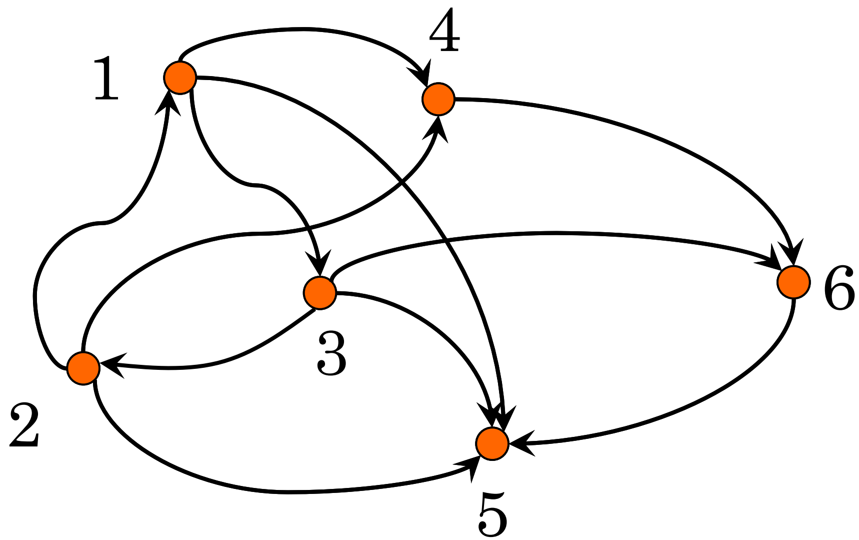
Let $G=(V,E)$ be a directed graph.

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

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

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

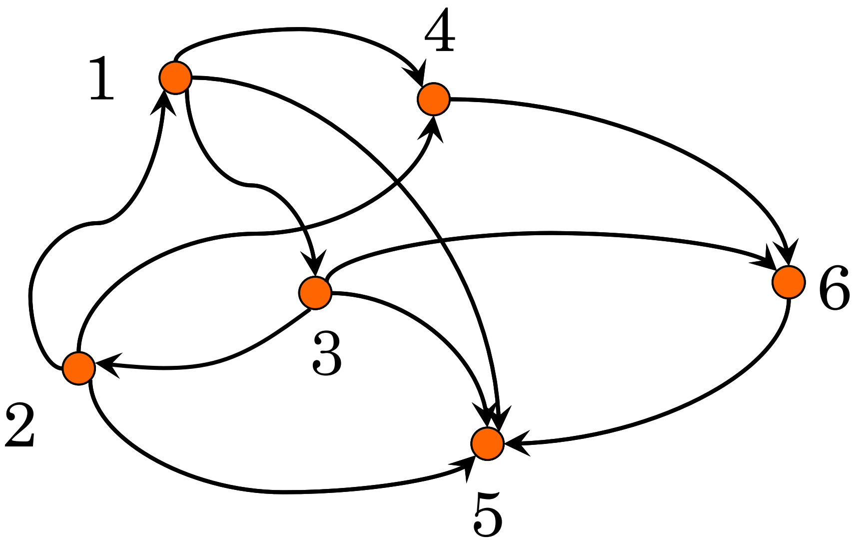
Adjacency matrix of a directed graph



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

If A is the adjacency matrix of a graph, then $(A^2)_{ij}=1$ iff there is a path (i, w, j) for a vertex w.

Adjacency matrix of a directed graph



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

Similarly, 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 $(A\vee I)^{n-1}=\mathrm{A}^{\mathrm{n-1}}\vee\mathrm{A}^{\mathrm{n-2}}\vee\ldots\vee\mathrm{A}\vee\mathrm{I}$ is the adjacency matrix of G^*.
  - The matrix $(A\vee I)^{n-1}$ can be computed by $\log n$ squaring operations in $\mathrm{O}(n^{\omega}\log n)$ time.

- Thus, the transitive closure can also be computed in $\tilde{\mathrm{O}}(n^{\omega})$ time.

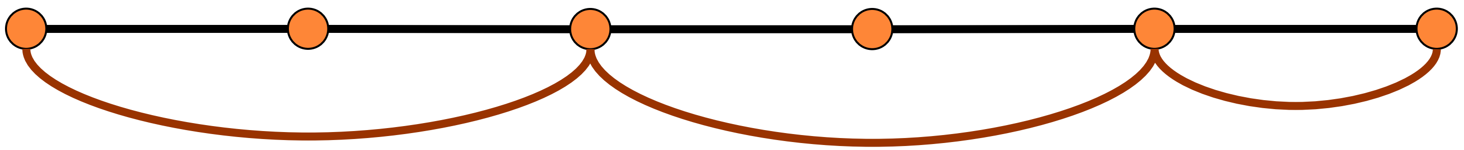
Undirected unweighted APSP

- An $O(n^{\omega})$ algorithm for undirected unweighted graphs (Seidel)

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 $u,v \in V$.

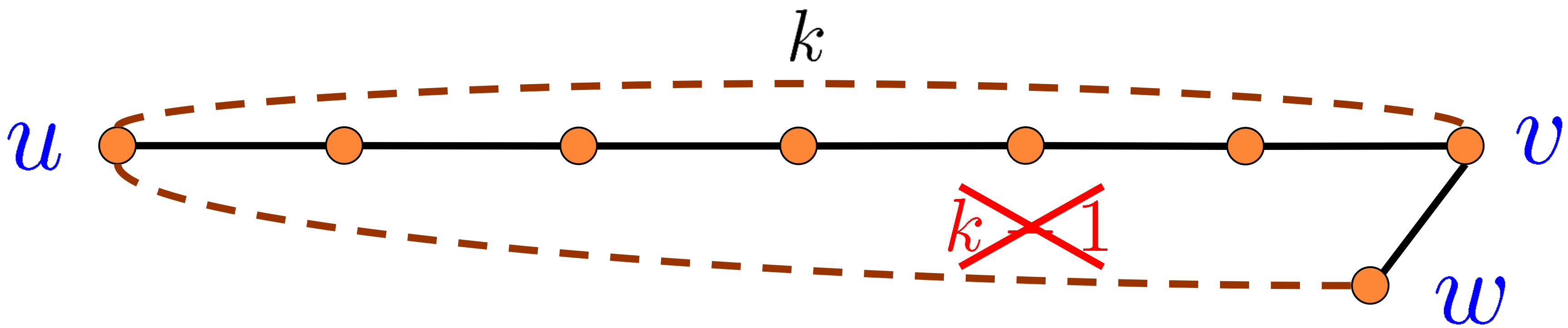
Thus: $\delta(u,v) = 2\delta^2(u,v)$ or
$\delta(u,v) = 2\delta^2(u,v) - 1$

Recursive procedure

- Suppose we have recursively computed the distance $\delta^2(u,v)$ for all pair u,v in G^2.
  - That is, we have the distance matrix C of G^2
- Then either $\delta(u,v) = 2\delta^2(u,v)$ or $\delta(u,v) = 2\delta^2(u,v) - 1$
  - We need to determine which one $\delta(u,v)$ is.

Even distances

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



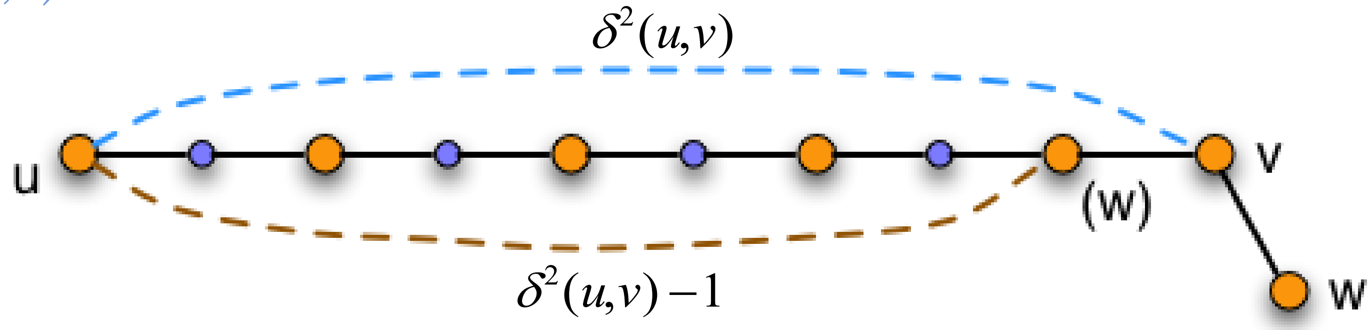
Let A be the adjacency matrix of the G.
Let C be the distance matrix of G^2

$$\sum_{(v,w)\in E} \delta^2(u,w) \geq \deg(v)\cdot \delta^2(u,v)$$

$$\sum_{w\in V} \delta^2(u,w)\cdot A_{w,v} = (C\cdot A)_{u,v} \geq \deg(v)\cdot \delta^2(u,v)$$

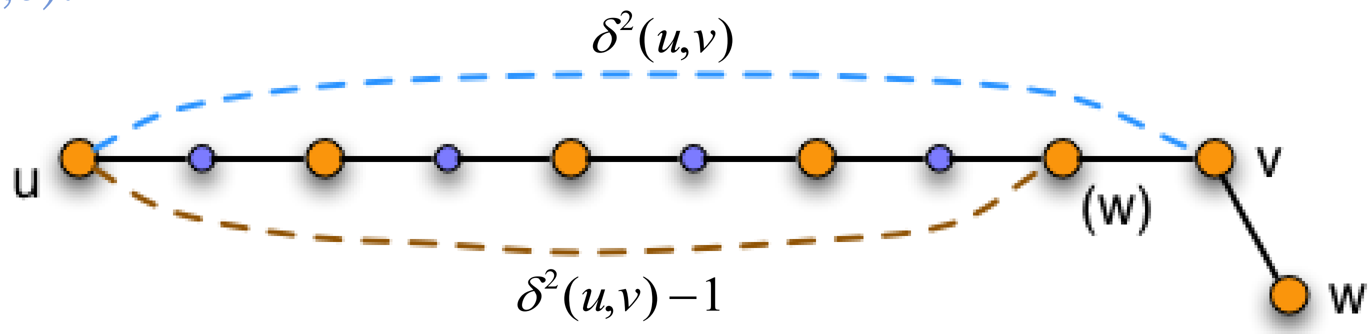
ODD DISTANCES

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



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



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

$$\sum_{(v,w)\in E} \delta^2(u,w) < \deg(v)\cdot \delta^2(u,v)$$

$$\sum_{w\in V} \delta^2(u,w)\cdot A_{w,v} = (C\cdot A)_{u,v} < \deg(v)\cdot \delta^2(u,v)$$

Recursive procedure

- Suppose we have recursively computed the distance $\delta^2(u,v)$ for all pairs u,v in G^2.
  - That is, we have the distance matrix C of G^2

- Then either $\delta(u,v) = 2\delta^2(u,v)$ or $\delta(u,v) = 2\delta^2(u,v) - 1$
  - Thus, we can judge which one $\delta(u,v)$ is for all pairs u,v by computing the matrix product C•A

Seidel's algori…

Assume that A has 1's on the diagonal.

1. If A is an all one matrix, then all distances are 1.

Seidel's algorithm

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.

Boolean matrix multiplicaion

Seidel's algorithm

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**.

Integer matrix multiplicaion

Seidel's algorithm

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**.

Algorithm APD(A)
if A=J then
  return J–I
else
  C←APD(A^2)
  X←CA , deg←Ae–1
  d_{ij}←2c_{ij}– [x_{ij}<c_{ij}deg$_j$]
  return D
end

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

All-Pairs Shortest Paths
in graphs with small integer weights

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

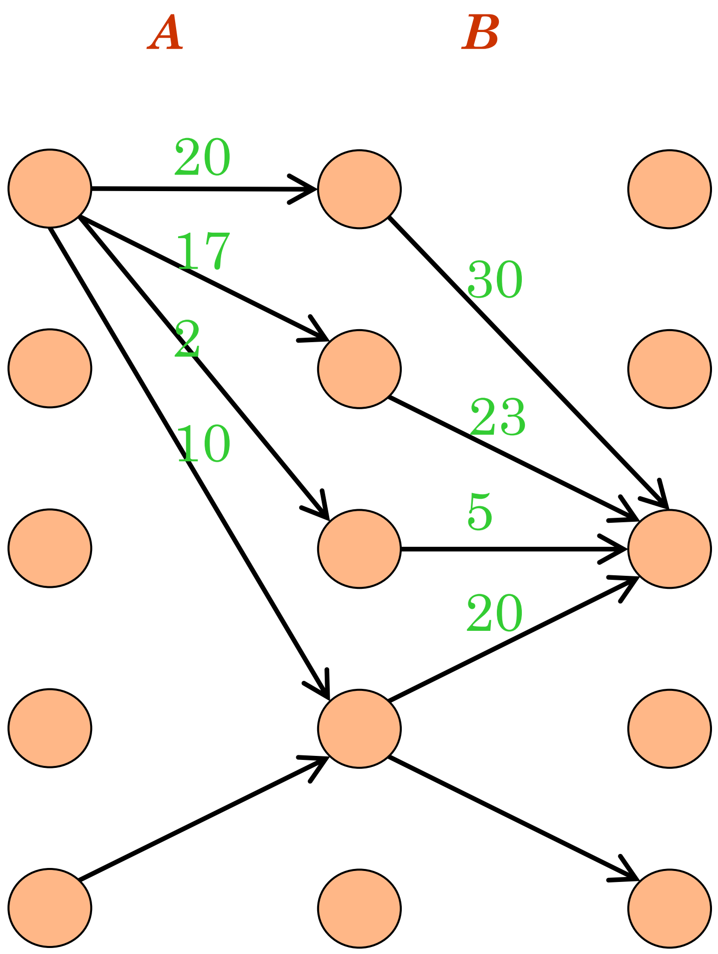
Running time	**Authors**
Mn^{ω}	[Shoshan-Zwick '99]

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

DIRECTED UNWEIGHTED APSP

- We will first talk about 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.

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

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

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

Algebraic Product

$$C = A \cdot B$$

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

$O(n^{2.38})$

Min-Plus Product

$$C = A * B$$

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

min operation has no inverse!

Algebraic Product

$$C = A \cdot B$$

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

$O(n^{2.38})$

Min-Plus Product

$$C = A * B$$

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

There is still no $O(n^{3-\varepsilon})$ algorithm for real weighted min-plus product

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 $n^{\omega+1}$!!!

A simple observation

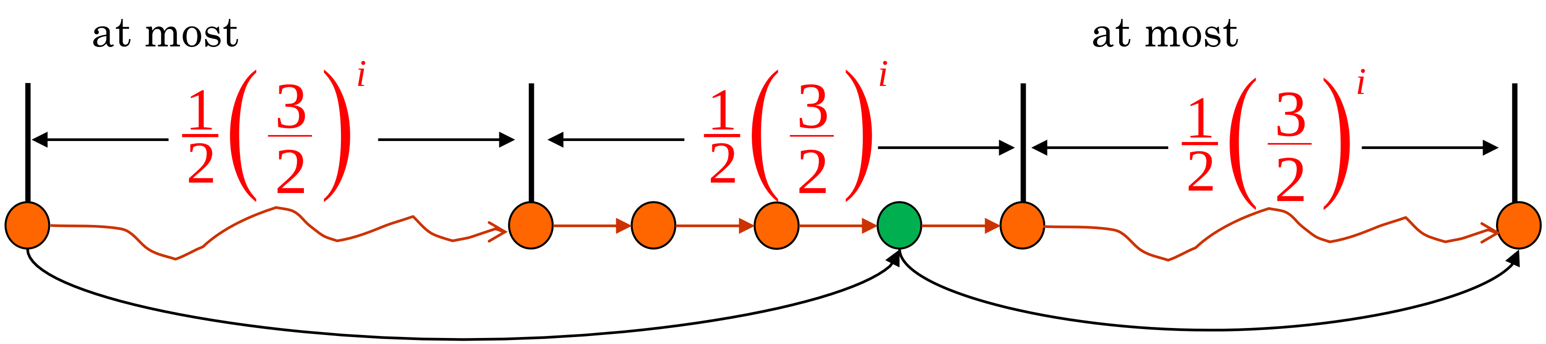
- If we randomly choose a subset S of n/k vertices
- Then any path of length k will contain a vertex in S with high-probability

A simple observation

- If we randomly choose a subset S of n/k vertices
- Then any path of length k will contain a vertex in S with high-probability

- So we just need to compute a rectangular matrix multiplication when computing large distances

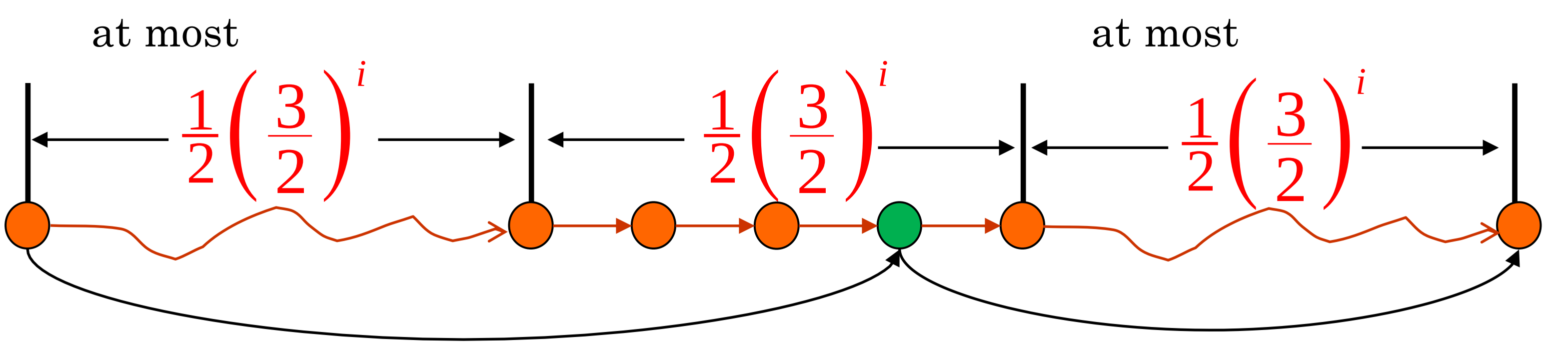
- If we randomly choose a *bridging* set B of vertices,
- Consider a shortest path that uses at most $(3/2)^{i+1}$ edges, we wish that there is a vertex of B in the middle range
- Then the path is composed of two subpaths of length $\leq (3/2)^{i}$.



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

Failure probability: $$\left(1 - \frac{|B|}{n}\right)^{s/3}$$

- Let $|B| = 9n \ln n / s$



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

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

Sampled Repeated Squaring (Z '98)

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

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

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

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

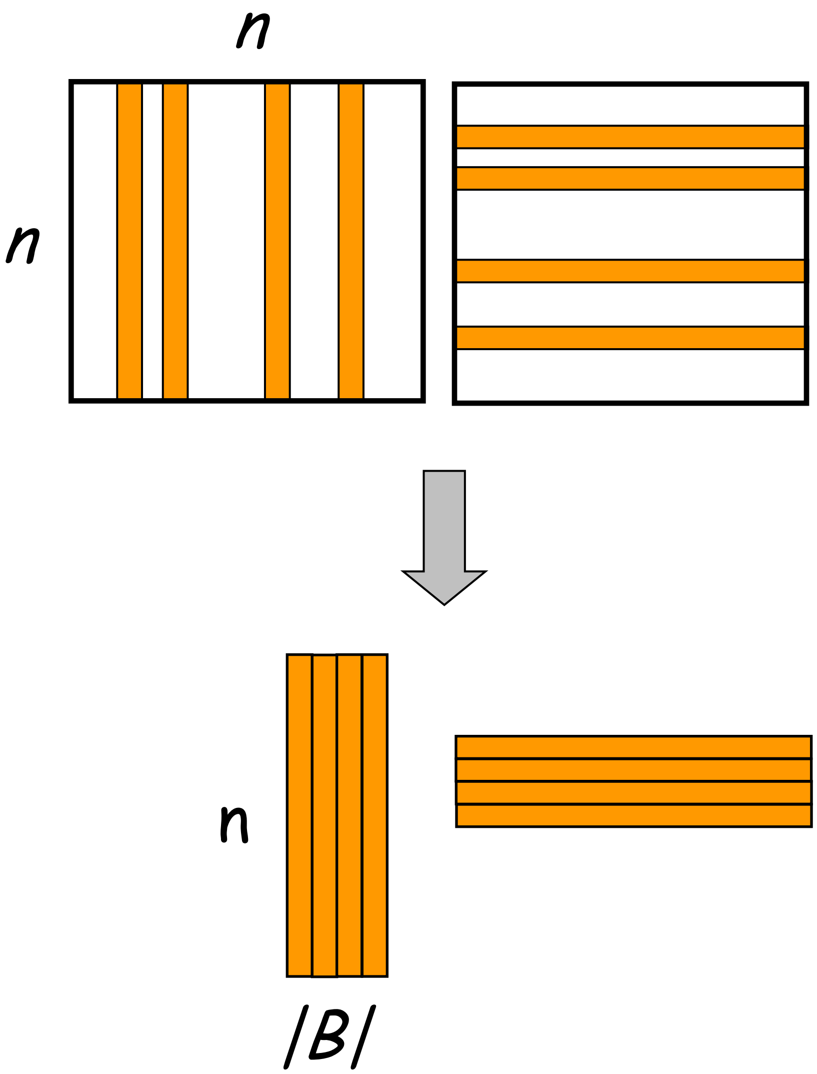
Sampled Repeated Squaring (Z '98)

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

With high probability, all distances are correct!

The is also a slightly more complicated deterministic algorithm

Sampled Distance Products (Z '98)



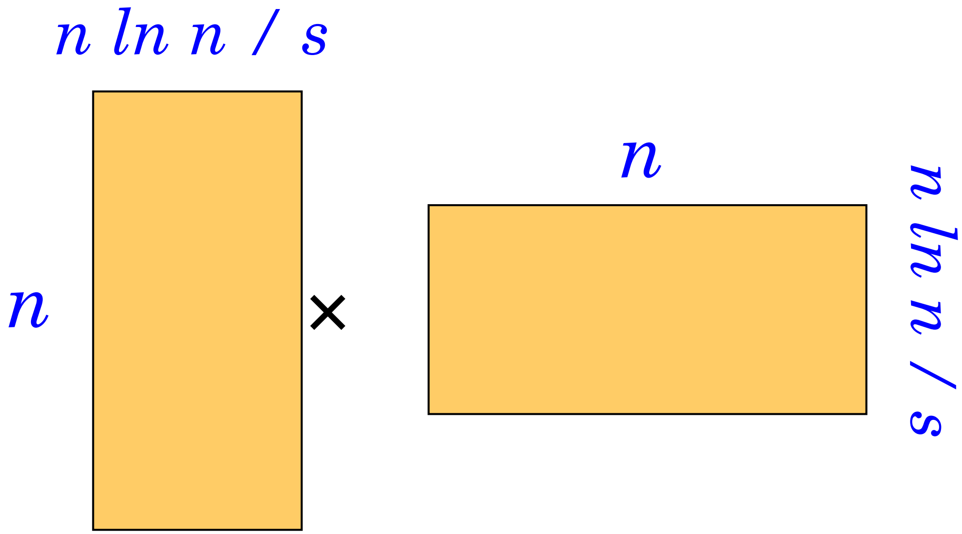
In the i-th iteration, the set B is of size $n \ln n \,/\, s$, where $s = (3/2)^{i+1}$

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

Complexity of APSP algorithm

The i-th iteration:

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



The elements are of absolute value at most Ms

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

Summary

All-Pairs Shortest Paths with integer edge weights in {1,2,…,M}

Problem	Running time	Authors
Transitive closure	$O(n^{\omega})=O(n^{2.38})$	*trivial*
Undirected unweighted APSP	$O(n^{\omega})=O(n^{2.38})$	Seidel ’95
Undirected APSP	$O(Mn^{2.38})$	Shoshan-Zwick ’99
Directed APSP	$O(M^{0.68}n^{2.58})$	Zwick ’98
(1+ε)-Approximate APSP	$O(n^{2.38} \log M)/\varepsilon$	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 $\{0,\pm 1, \pm 2,\ldots, \pm n\}$?

Thank you!