**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**

---

**Algebraic Matrix Multiplication**

Given two matrices $A$ and $B$, we can compute their product $C$ as:

$$C = AB$$

where

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

This can be computed naively in $O(n^3)$ time.

**Matrix multiplication algorithms**

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n^3$</td>
<td>—</td>
</tr>
<tr>
<td>$n^{2.81}$</td>
<td>Strassen (1969)</td>
</tr>
<tr>
<td>$n^{2.38}$</td>
<td>Coppersmith, Winograd (1990)</td>
</tr>
</tbody>
</table>

Conjecture/Open problem: $n^{2+\omega(1)}$ ???
Multiplying $2 \times 2$ matrices

\[
\begin{pmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{pmatrix} =
\begin{pmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{pmatrix}
\begin{pmatrix}
U_{11} & U_{12} \\
B_{21} & B_{22}
\end{pmatrix}
\]

\[
C_{11} = A_{11}B_{11} + A_{12}B_{21}
\]
\[
C_{12} = A_{11}B_{12} + A_{12}B_{22} \quad 8 \text{ multiplications}
\]
\[
C_{21} = A_{21}B_{11} + A_{22}B_{21} \quad 4 \text{ additions}
\]
\[
C_{22} = A_{21}B_{12} + A_{22}B_{22}
\]

\[
T(n) = 8 T(n/2) + O(n^2)
\]
\[
T(n) = O(n \log^8(n) \log^2) = O(n^3)
\]

Strassen’s $2 \times 2$ algorithm

\[
\begin{align*}
C_{11} &= A_{11}B_{11} + A_{12}B_{21} \\
C_{12} &= A_{11}B_{12} + A_{12}B_{22} \\
C_{21} &= A_{21}B_{11} + A_{22}B_{21} \quad 8 \text{ multiplications} \\
C_{22} &= A_{21}B_{12} + A_{22}B_{22} \\
C_{11} &= M_1 + M_4 - M_5 + M_7 \\
C_{12} &= M_2 + M_6 \\
C_{21} &= M_3 + M_5 \\
C_{22} &= M_1 - M_2 + M_6 + M_8 \quad 7 \text{ multiplications}
\end{align*}
\]

\[
\begin{align*}
M_1 &= (A_{11} + A_{12})(B_{12} - B_{22}) \\
M_2 &= (A_{21} + A_{22})B_2 \\
M_3 &= (A_{11} + A_{22})B_{21} \\
M_4 &= A_{11}(B_{12} - B_{22}) \\
M_5 &= A_{12}B_{21} - B_{11} \\
M_6 &= A_{21}B_{22} - A_{11}(B_{21} + B_{22}) \\
M_7 &= (A_{21} - A_{22})(B_{12} + B_{22}) \\
M_8 &= (A_{11} - A_{12})(B_{21} + B_{22})
\end{align*}
\]

\[
\begin{align*}
\text{18 additions/subtractions}
\end{align*}
\]

Strassen’s $n \times n$ algorithm

View each $n \times n$ matrix as a $2 \times 2$ matrix whose elements are $n/2 \times n/2$ matrices.

Apply the $2 \times 2$ algorithm recursively.

\[
T(n) = 7 T(n/2) + O(n^2)
\]
\[
T(n) = O(n \log^7(n) \log^2) = O(n^{2.81})
\]

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 $2 \leq \omega < 2.38$ be the exponent of matrix multiplication.

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

Rectangular Matrix multiplication

\[
\begin{align*}
\text{Naïve complexity: } & \quad n^2p \\
\text{[Coppersmith ‘97]: } & \quad n^{1.85}p^{0.54} + n^{2+\omega(1)}
\end{align*}
\]

For $p \leq n^{0.29}$, complexity $= n^{2+\omega(1)}$
### Boolean Matrix Multiplication

![Matrix Multiplication Diagram](image)

\[ A = (a_{ij}) \times B = (b_{ij}) = C = (c_{ij}) \]

\[ c_{ij} = \bigvee_{k=1}^{n} a_{ik} \land b_{kj} \]

Can be computed naively in \( O(n^3) \) time.

### Algebraic Product

\[ C = A \cdot B \]

\[ c_{ij} = \sum_{k} a_{ik} b_{kj} \]

**Operations**: \( O(n^{2.38}) \)

### Boolean Product

\[ C = A \cdot B \]

\[ c_{ij} = \bigvee_{k} a_{ik} \land b_{kj} \]

**Operations**: \( O(n^{2.38}) \)

### 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 \((A^k)_{ij} = 1 \) is the adjacency matrix of \( G^* \).

The matrix \((A^k)_{ij} \) 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.

### TC(n) bound

\[ TC(n) \leq 2 \cdot TC(n/2) + 6 \cdot BMM(n/2) + O(n^2) \]
**Exercise 1:** Give $O(n^w)$ algorithms for finding, 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**

**Min-Plus Products**

$$C = A \ast B$$

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

$$\begin{pmatrix}
-6 & -3 & -10 \\
2 & 5 & -2 \\
-1 & -7 & -5
\end{pmatrix} = \begin{pmatrix}
1 & -3 & 7 \\
+\infty & 5 & +\infty \\
8 & 2 & -5
\end{pmatrix} \ast \begin{pmatrix}
8 & +\infty & -4 \\
-3 & 0 & -7 \\
5 & -2 & 1
\end{pmatrix}$$

---

**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, $W$ gives the distances realized by paths that use at most $k$ edges.

```markdown
D \leftarrow W \\
for i \leftarrow 1 to \lceil \log_2 n \rceil \\
do D \leftarrow D^\ast D
```

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

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

---

**$X$**

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**$X'$**

<table>
<thead>
<tr>
<th></th>
<th>$E$</th>
<th>$F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$H$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$X' = \begin{pmatrix}
(A\ast BD \ast C)^* & EBD^* \\
D^* CE & D^\ast GBD^*
\end{pmatrix}$

$\text{APSP}(n) \leq 2 \cdot \text{APSP}(n/2) + 6 \cdot \text{MPP}(n/2) + O(n^2)$
Using matrix multiplication to compute min-plus products

Assume: \(0 \leq a_{ij}, b_{ij} \leq M\)

\[
\begin{pmatrix}
 c_{11}' & c_{12}' \\
 c_{21}' & c_{22}' \\
 \vdots & \vdots
\end{pmatrix}
= \begin{pmatrix}
 x_{11} & x_{12} & \cdots \\
 x_{21} & x_{22} & \cdots \\
 \vdots & \vdots & \ddots
\end{pmatrix}
\times
\begin{pmatrix}
 x_{11}^a & x_{12}^a & \cdots \\
 x_{21}^b & x_{22}^b & \cdots \\
 \vdots & \vdots & \ddots
\end{pmatrix}
\]

\(n^6\) polynomial products \(\times\) \(M\) operations per polynomial product = \(Mn^6\) 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

\[
A \ast B = \min_i A_i \ast B_i
\]

\(\text{MPP}(n) \leq (n/m) \times (\text{MPP}(n,m,n) + n^2)\)
Fredman’s trick

\[ a_{ir} + b_{rj} \leq a_{is} + b_{sj} \]
\[ a_{ir} - a_{is} \leq b_{sj} - b_{rj} \]

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.)

\[ a_{ir} + b_{rj} \leq a_{is} + b_{sj} \iff a_{ir} - a_{is} \leq b_{sj} - b_{rj} \]

• 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

<table>
<thead>
<tr>
<th>Running time</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n^3 )</td>
<td>[Floyd ’62] [Warshall ’62]</td>
</tr>
<tr>
<td>( n^3 (\log \log n / \log n)^{1/2} )</td>
<td>[Fredman ’76]</td>
</tr>
<tr>
<td>( n^3 (\log \log n / \log n)^{1/2} )</td>
<td>[Takaoka ’92]</td>
</tr>
<tr>
<td>( n^3 (\log \log n / \log n)^{1/2} )</td>
<td>[Dobosiewicz ’90]</td>
</tr>
<tr>
<td>( n^3 (\log \log n / \log n)^{1/2} )</td>
<td>[Han ’04]</td>
</tr>
<tr>
<td>( n^3 (\log \log n / \log n)^{1/2} )</td>
<td>[Chan ’05]</td>
</tr>
<tr>
<td>( n^3 (\log \log n / \log n)^{1/2} )</td>
<td>[Takaoka ’04]</td>
</tr>
<tr>
<td>( n^3 (\log \log n / \log n)^{1/2} )</td>
<td>[Zwick ’04]</td>
</tr>
<tr>
<td>( n^3 (\log \log n / \log n)^{1/2} )</td>
<td>[Han ’06]</td>
</tr>
<tr>
<td>( n^3 (\log \log n / \log n)^{1/2} )</td>
<td>[Chan ’07]</td>
</tr>
</tbody>
</table>

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 \( O(M^{1.68}n^{2.58}) \) 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

\[
\delta(x, z) \leq \delta(x, y) + \delta(y, z)
\]

**Triangle inequality**

\[
\delta(x, z) \leq \delta(x, y) + \delta(y, z)
\]

In order to have

\[
\delta(x, z) \leq \delta(x, y) + \delta(y, z)
\]

be the distance from \(x\) to \(z\) through \(y\), we need \(\delta(y, z) \leq \delta(z, y)

**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 \(G^2\).

\[
\delta(u,v) = 5 \quad \delta^2(u,v) = 3
\]

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

Lemma: \(\delta^2(u,v) \leq \left\lceil \frac{\delta(u,v)}{2} \right\rceil\)

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

Let \(A\) be the adjacency matrix of the \(G\).

Let \(C\) be the distance matrix of \(G^2\)

\[
\sum_{(u,v)\in E} \delta_u(v) = \sum_{w\in V} \delta_u(w) = (CA)_{u,v} = \deg(v)\delta_u(v)
\]

Even distances

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

Let \(A\) be the adjacency matrix of the \(G\).

Let \(C\) be the distance matrix of \(G^2\)

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

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

**Exercise 2:** Prove the lemma.

Let \(A\) be the adjacency matrix of the \(G\).

Let \(C\) be the distance matrix of \(G^2\)

\[
\sum_{(u,v)\in E} c_{uw} = \sum_{w\in V} c_{uw}a_{wv} = (CA)_{u,v} < \deg(v)c_{uw}
\]
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 \( G^2 \)

\[
\sum_{(u,v) \in E} c_{uw} = \sum_{u \in V} c_{uw} \cdot \alpha_{uw} = (C'A)_{uv} \geq \deg(v) c_{uw}
\]

Exercise 3: (*) Obtain a version of Seidel’s algorithm that uses only Boolean matrix multiplications.

Hint: Look at distances also modulo 3.

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.

Complexity: \( O(n^{\omega} \log n) \)

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

\[
C = AB
\]

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 \( \{0, 1, \ldots, M\} \)

<table>
<thead>
<tr>
<th>Running time</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Mn^{\Theta} )</td>
<td>[Shoshan-Zwick ‘99]</td>
</tr>
</tbody>
</table>

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

### Exercise 5:
Obtain an \( O(n^{\Theta} \log n) \) time algorithm for computing the *diameter* of an unweighted directed graph.

Using matrix multiplication to compute min-plus products

\[
\begin{pmatrix}
    c_{11} & c_{12} \\
    c_{21} & c_{22} \\
    \vdots & \vdots
\end{pmatrix}
= 
\begin{pmatrix}
    a_{11} & a_{12} \\
    a_{21} & a_{22} \\
    \vdots & \vdots
\end{pmatrix}
\times 
\begin{pmatrix}
    b_{11} & b_{12} \\
    b_{21} & b_{22} \\
    \vdots & \vdots
\end{pmatrix}
\]

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

\[
\begin{pmatrix}
    c'_{11} & c'_{12} \\
    c'_{21} & c'_{22} \\
    \vdots & \vdots
\end{pmatrix}
= 
\begin{pmatrix}
    x_{1}^{a_{11}} & x_{1}^{a_{12}} \\
    x_{2}^{a_{21}} & x_{2}^{a_{22}} \\
    \vdots & \vdots
\end{pmatrix}
\times 
\begin{pmatrix}
    x_{1}^{b_{11}} & x_{1}^{b_{12}} \\
    x_{2}^{b_{21}} & x_{2}^{b_{22}} \\
    \vdots & \vdots
\end{pmatrix}
\]

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

Using matrix multiplication to compute min-plus products

Assume: \( 0 \leq a_{ij}, b_{ij} \leq M \)

\[
\begin{pmatrix}
    c_{11} & c_{12} \\
    c_{21} & c_{22} \\
    \vdots & \vdots
\end{pmatrix}
= 
\begin{pmatrix}
    x_{1}^{a_{11}} & x_{1}^{a_{12}} \\
    x_{2}^{a_{21}} & x_{2}^{a_{22}} \\
    \vdots & \vdots
\end{pmatrix}
\times 
\begin{pmatrix}
    x_{1}^{b_{11}} & x_{1}^{b_{12}} \\
    x_{2}^{b_{21}} & x_{2}^{b_{22}} \\
    \vdots & \vdots
\end{pmatrix}
\]

\( M \) operations per max-plus product

\( n^{\Theta} \) polynomial products \times \( M \) operations per polynomial product = \( Mn^{\Theta} \) operations per max-plus product

Trying to implement the repeated squaring algorithm

\[
D \leftarrow W
\]
for \( i \leftarrow 1 \) to \( \log_2 n \)
do \( D \leftarrow D \times D \)

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 \( 2^i n^{\Theta} \)

The cost of the last product is \( n^{\Theta+1} \) !!!
Sampled Repeated Squaring (Z ’98)

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

The elements are of absolute value at most $M s^{0.54} n^{1.85}$

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 = $n^{2 + o(1)}$ !!!

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

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 $s = (3/2)^i$

Failure probability: $1 - \frac{9 \ln n}{s} < n^{-3}$

Complexity of APSP algorithm

The $i$-th iteration:

The elements are of absolute value at most $Ms$

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

Open problem:

Can APSP in directed graphs be solved in $O(n^w)$ time?

Related result: [Yuster-Zwick’04]

A directed graph can be processed in $O(n^w)$ time so that any distance query can be answered in $O(n)$ time.

Corollary:

SSSP in directed graphs in $O(n^w)$ time.

The corollary obtained using a different technique by Sankowski (2004)
The preprocessing algorithm (YZ '05)

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

The APSP algorithm

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

Twice Sampled Distance Products

The query answering algorithm

\[ \delta(u,v) \leftarrow D([u],V) \cdot D[V,[v]] \]

Query time: \( O(n) \)

The preprocessing algorithm: Correctness

Let \( B_i \) be the \( i \)-th sample. \( B_i \supseteq B_{i+1} \supseteq \cdots \)

**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

\[ \frac{1}{2} \left( \frac{3}{2} \right)^i \]

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

\[ \frac{1}{2} \left( \frac{3}{2} \right)^i \leq \delta(u,v) \leq \frac{1}{2} \left( \frac{3}{2} \right)^{i+1} \]
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 $O(n^{2.38})$ algorithm 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 $O(n^{2.38} \log M)(1+\varepsilon)$-approximation algorithm
5. Summary and open problems

Approximate min-plus products

Obvious idea: scaling

SCALE($A$, $M$, $R$):

\[
\begin{cases}
    a'_{ij} = \left\lfloor \frac{a_{ij}}{M} \right\rfloor, & \text{if } 0 \leq a_{ij} \leq M \\
    \infty, & \text{otherwise}
\end{cases}
\]

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

\[
A' \leftarrow \text{SCALE}(A, 2^r, R)
\]

\[
B' \leftarrow \text{SCALE}(B, 2^r, R)
\]

\[
C' \leftarrow \min\{C', \text{MPP}(A', B')\}
\]

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

Addaptive Scaling

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

\[
C' \leftarrow Rn^{2.38} \log M
\]

Complexity is $Rn^{2.38} \log M$

Stretch at most 1+4/R

All-Pairs Shortest Paths
in graphs with small integer weights

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

<table>
<thead>
<tr>
<th>Running time</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Mn^{2.38}$</td>
<td>Shoshan-Zwick ’99</td>
</tr>
</tbody>
</table>

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

All-Pairs Shortest Paths
in graphs with small integer weights

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

<table>
<thead>
<tr>
<th>Running time</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M^{0.68} n^{2.58}$</td>
<td>Zwick ’98</td>
</tr>
</tbody>
</table>

Implements results of
[Alon-Galil-Margalit ’91][Takaoka ’98]
Answering distance queries

Directed graphs. Edge weights in \([-M, \ldots, 0, \ldots, M]\)

<table>
<thead>
<tr>
<th>Preprocessing time</th>
<th>Query time</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Mn^{2.38})</td>
<td>(n)</td>
<td>[Yuster-Zwick '05]</td>
</tr>
</tbody>
</table>

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. \(mn^{0.5}log M\)

Open problems

- An \(O(n^{2.38})\) algorithm for the directed unweighted APSP problem?
- An \(O(n^{3.5})\) algorithm for the APSP problem with edge weights in \(\{1, 2, \ldots, n\}\)?
- An \(O(n^{2.5})\) algorithm for the SSSP problem with edge weights in \(\{0, \pm 1, \pm 2, \ldots, \pm n\}\)?

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

<table>
<thead>
<tr>
<th>Running time</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>((n^{2.38})log M/\varepsilon)</td>
<td>[Zwick '98]</td>
</tr>
</tbody>
</table>