

# All-Pairs shortest paths via fast matrix multiplication

Uri Zwick  
Tel Aviv University

Summer School on Shortest Paths (PATH05)  
DIKU, University of Copenhagen

## 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  $O(n^{2.38} \log M)$   $(1+\epsilon)$ -approximation algorithm

## 5. Summary and open problems

## ➔ 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

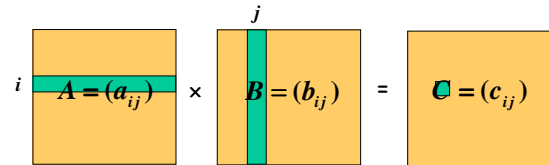
- a. An  $O(n^{2.38})$  algorithm for unweighted graphs (Seidel)
- b. An  $O(Mn^{2.38})$  algorithm for weighted graphs (Shoshan-Zwick)

## 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  $O(n^{2.38} \log M)$   $(1+\epsilon)$ -approximation algorithm

## 5. Summary and open problems

## Matrix multiplication



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

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

## Matrix multiplication

Complexity	Authors
$n^3$	(by definition)
$n^{2.81}$	Strassen (1969)
$n^{2.38}$	Coppersmith, Winograd (1990)

Conjecture/Open problem:  $n^{2+o(1)}$  ???

## Multiplying 2x2 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

## Strassen's 2x2 algorithm

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

7 multiplications  
18 additions/subtractions

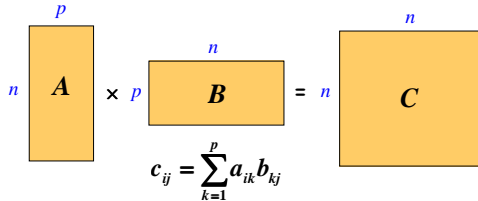
## Strassen's nxn algorithm

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.

$$\begin{aligned}
 T(n) &= 7 T(n/2) + O(n^2) \\
 T(n) &= O(n^{\log_2 7 / \log_2 2}) = O(n^{2.81})
 \end{aligned}$$

## Rectangular Matrix multiplication



Coppersmith (1997):

$$\text{Complexity} \leq n^{1.85} p^{0.54} + n^2 + o(1)$$

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

### 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

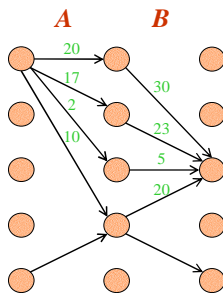
- a. An  $O(n^{2.38})$  algorithm for unweighted graphs (Seidel)
- b. An  $O(Mn^{2.38})$  algorithm for weighted graphs (Shoshan-Zwick)

### 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  $O(n^{2.38} \log M)$   $(1+\epsilon)$ -approximation algorithm

### 5. Summary and open problems

## 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

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

$$C = A * B$$

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

## Solving APSP by repeated squaring

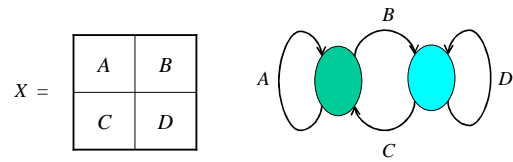
If  $W$  is an  $n$  by  $n$  matrix containing the edge weights of a graph. Then  $W^n$  is the distance matrix.

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:  $APSP(n) \leq MPP(n) \log n$

Actually:  $APSP(n) = O(MPP(n))$



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

$$APSP(n) \leq 2 APSP(n/2) + 6 MPP(n/2) + O(n^2)$$

## Algebraic Product

$$C = A \cdot B$$

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

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

## Min-Plus Product

$$C = A * B$$

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

The fast algebraic algorithm cannot be used, as the **min operation has no inverse**

### 1. Algebraic matrix multiplication

- Strassen's algorithm
- Rectangular matrix multiplication

### 2. Min-Plus matrix multiplication

- Equivalence to the APSP problem
- Expensive reduction to algebraic products
- Fredman's trick

### 3. APSP in undirected graphs

- An  $O(n^{2.38})$  algorithm for unweighted graphs (Seidel)
- An  $O(Mn^{2.38})$  algorithm for weighted graphs (Shoshan-Zwick)

### 4. APSP in directed graphs

- An  $O(M^{0.68}n^{2.58})$  algorithm (Zwick)
- An  $O(Mn^{2.38})$  preprocessing /  $O(n)$  query answering alg. (Yuster-Z)
- An  $O(n^{2.38} \log M)$   $(1+\epsilon)$ -approximation algorithm

### 5. Summary and open problems

## Using matrix multiplication to compute min-plus products

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

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

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

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

## Using matrix multiplication to compute min-plus products

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

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

$$n^{2.38} \text{ polynomial products} \times M \text{ operations per polynomial product} = Mn^{2.38} \text{ operations per max-plus product}$$

## Trying to implement the repeated squaring algorithm

$D \leftarrow W$   
**for**  $i \leftarrow 1$  **to**  $\log_2 n$       Consider an easy case:  
**do**  $D \leftarrow D * D$                       all weights are 1.

After the  $i$ -th iteration, the finite elements in  $D$  are in the range  $\{1, \dots, 2^i\}$ .

The cost of the min-plus product is  $2^i n^{2.38}$

The cost of the last product is  $n^{3.38}$  !!!

### 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

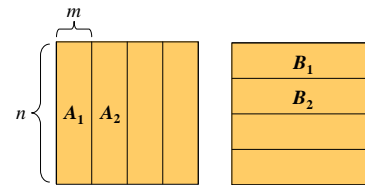
1. An  $O(M^{0.68}n^{2.58})$  algorithm (Zwick)
2. An  $O(Mn^{2.38})$  preprocessing /  $O(n)$  query answering alg. (Yuster-Z)
3. An  $O(n^{2.38} \log M)$   $(1+\epsilon)$ -approximation algorithm

### 5. Summary and open problems

## 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.

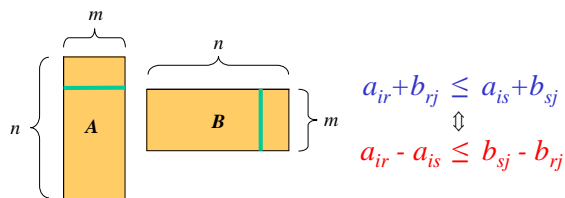
## Breaking a square product into several rectangular products



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

$$\text{MPP}(n) \leq (n/m) (\text{MPP}(n, m, n) + n^2)$$

## Fredman's trick



Naïve calculation requires  $n^2 m$  operations

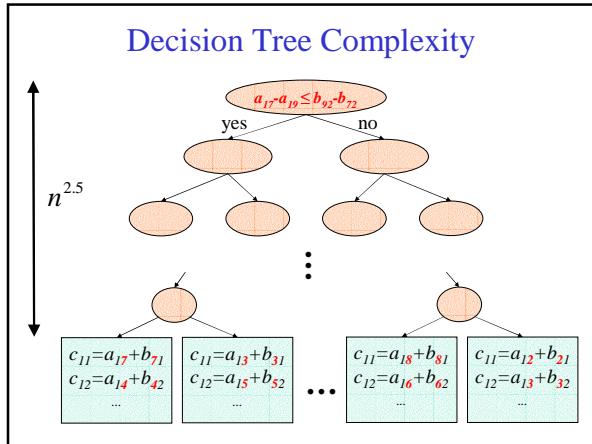
Fredman observed that the result can be **inferred** after performing only  $O(nm^2)$  operations

## Fredman's trick (cont.)

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

- **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.

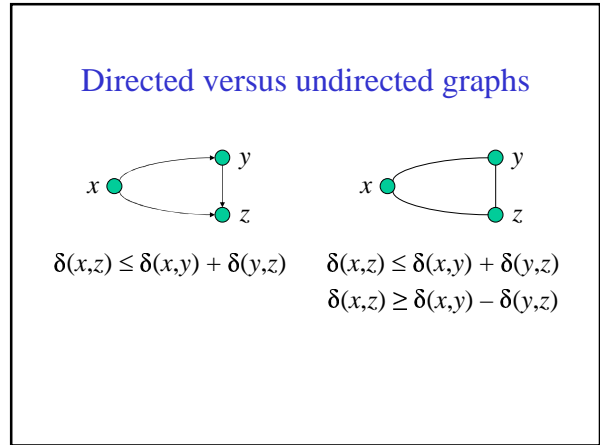
The ordering of the elements in the sorted list determines the result of the min-plus product !!!



### 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]

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  $O(n^{2.38} \log M)$   $(1+\epsilon)$ -approximation algorithm
5. **Summary and open problems**



### 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  $G^2$ .

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

**Lemma:** If  $\delta(u,v) = 2\delta^2(u,v) - 1$  then for every neighbor  $w$  of  $v$  we have  $\delta^2(u,w) \leq \delta^2(u,v)$  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  $G^2$

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

## 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 square.
4. Decide, using 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(A2)
  X ← CA, deg ← Ae-1
  dij ← 2cij - [xij < cij degj]
  return D
end
    
```

Complexity:  
 $O(n^{2.38} \log n)$

### 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  $O(n^{2.38} \log M)$   $(1+\epsilon)$ -approximation algorithm

### 5. Summary and open problems

## Sampled Repeated Squaring (Z '98)

```

D ← W
for i ← 1 to log3/2 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  $(9n \ln n)/s$

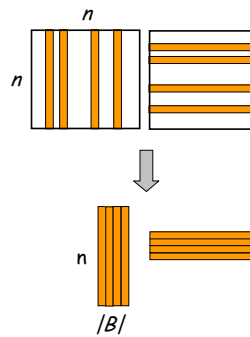
Select the columns

The is also a slightly more complicated algorithm of  $D$  whose indices are in  $B$

Select the rows

of  $D$  whose indices are in  $B$

## 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

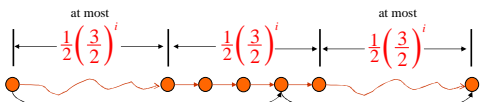
## Sampled Repeated Squaring - Correctness

```

D ← W
for i ← 1 to log3/2 n do
{
  s ← (3/2)i+1
  B ← rand(V, (9n ln n)/s)
  D ← min{ D, D[V,B]*D[B,V] }
}
    
```

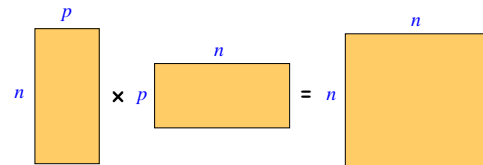
**Invariant:** After the  $i$ -th iteration, distances that are attained using at most  $(3/2)^i$  edges are correct.

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



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

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

## Complexity of APSP algorithm

The  $i$ -th iteration:

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

The elements are of absolute value at most  $Ms$

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

### 1. Algebraic matrix multiplication

- Strassen's algorithm
- Rectangular matrix multiplication

### 2. Min-Plus matrix multiplication

- Equivalence to the APSP problem
- Expensive reduction to algebraic products
- Fredman's trick

### 3. APSP in undirected graphs

- An  $O(n^{2.38})$  algorithm for unweighted graphs (Seidel)
- An  $O(Mn^{2.38})$  algorithm for weighted graphs (Shoshan-Zwick)

### 4. APSP in directed graphs

- An  $O(M^{0.68}n^{2.58})$  algorithm (Zwick)
- An  $O(Mn^{2.38})$  preprocessing /  $O(n)$  query answering alg. (Yuster-Z)
- An  $O(n^{2.38} \log M)$   $(1+\epsilon)$ -approximation algorithm

### 5. Summary and open problems

## The preprocessing algorithm (YZ '05)

```

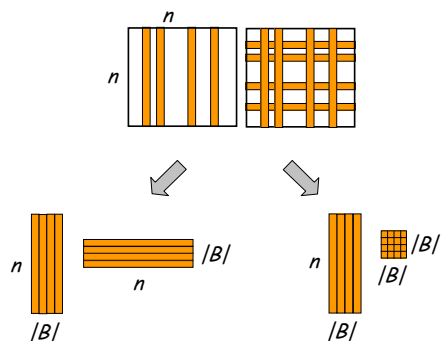
D ← W ; B ← V
for i ← 1 to log_{3/2} n do
{
  s ← (3/2)^{i+1}
  B ← rand(B, (9n ln n)/s)
  D[V, B] ← min{D[V, B], D[V, B]*D[B, B]}
  D[B, V] ← min{D[B, V], D[B, B]*D[B, V]}
}
    
```

## The APSP algorithm

```

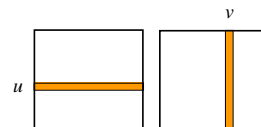
D ← W
for i ← 1 to log_{3/2} n do
{
  s ← (3/2)^{i+1}
  B ← rand(V, (9n ln n)/s)
  D ← min{ D, D[V, B]*D[B, V] }
}
    
```

## Twice Sampled Distance Products



## The query answering algorithm

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



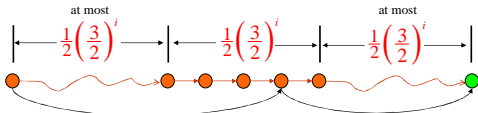
Query time:  $O(n)$

## The preprocessing algorithm: Correctness

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

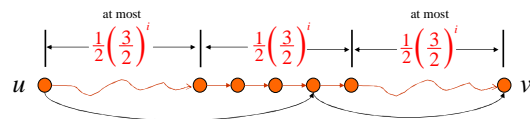
**Invariant:** After the  $i$ -th iteration, if  $u \in B_i$  or  $v \in B_i$ , and there is a shortest path from  $u$  to  $v$  that uses at most  $(3/2)^i$  edges, then  $D(u,v) = \delta(u,v)$ .

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  $O(n^{2.38})$  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  $O(n^{2.38} \log M)$   $(1+\epsilon)$ -approximation algorithm
5. **Summary and open problems**

## Approximate min-plus products

Obvious idea: scaling

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

**APX-MPP(A, B, M, R) :**  
 $A' \leftarrow \text{SCALE}(A, M, R)$   
 $B' \leftarrow \text{SCALE}(B, M, R)$   
 return  $\text{MPP}(A', B')$

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' ← ∞
for r ← log2R to log2M do
  A' ← SCALE(A, 2r, R)
  B' ← SCALE(B, 2r, 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  $O(n^{2.38})$  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  $O(n^{2.38} \log M)$   $(1+\epsilon)$ -approximation algorithm
- ➔ 5. **Summary and open problems**

## All-Pairs Shortest Paths in graphs with small integer weights

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

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  $\{-M,\dots,0,\dots,M\}$

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  $\{-M,\dots,0,\dots,M\}$

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  $\{0,1,\dots,M\}$   
 $(1+\epsilon)$ -approximate distances

Running time	Authors
$(n^{2.38} \log M)/\epsilon$	[Zwick '98]

## Open problems

- An  $O(n^{2.38})$  algorithm for the directed unweighted **APSP** problem?
- An  $O(n^{3-\epsilon})$  algorithm for the **APSP** problem with edge weights in  $\{1,2,\dots,n\}$ ?
- An  $O(n^{2.5-\epsilon})$  algorithm for the **SSSP** problem with edge weights in  $\{0,\pm 1, \pm 2, \dots, \pm n\}$ ?