max planck institut
informatik

# Complexity Theory of Polynomial-Time Problems

## Lecture 8: (Boolean) Matrix Multiplication

**Karl Bringmann**

# Recall: Boolean Matrix Multiplication

given $n \times n$ matrices $A, B$ with entries in $\{0,1\}$

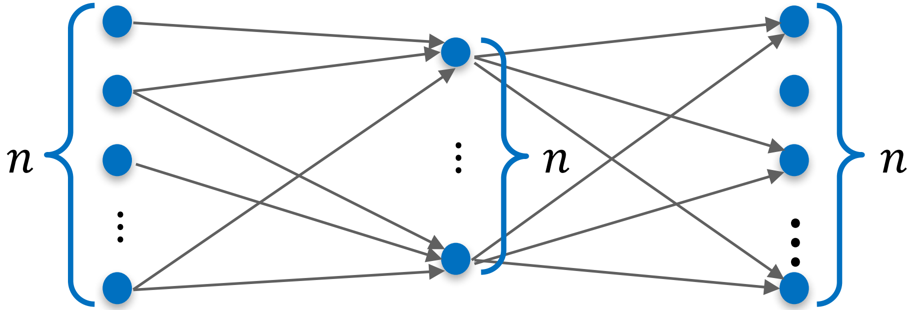
compute matrix $C$ with $C_{i,j} = \bigvee_{k=1}^{n} A_{i,k} \wedge B_{k,j}$

what we already know about BMM:

BMM is in time $O(n^3 / \log n)$ (four Russians)

BMM is equivalent to computing the **Transitive Closure** of a given graph

BMM can be reduced to APSP $\rightarrow O(n^3 / 2^{\sqrt{\log n}})$



# Exponent of Matrix Multiplication

define $\omega$ as the *infimum* over all $c$ such that MM has an $O(n^c)$ algorithm

note: MM is in time $O(n^{\omega+\varepsilon})$ for any $\varepsilon > 0$

we will be sloppy and write: MM is in time $O(n^\omega)$

note: MM is **not** in time $O(n^{\omega-\varepsilon})$ for any $\varepsilon > 0$ $\quad\omega \geq 2$

**Thm:** $\omega < 3$

this is very fast – in theory

all these algorithms have impractically large constant factors

(maybe except Strassen'69)

| | $\omega \leq$ … |
|---|---|
| **Strassen'69** | **2.81** |
| Pan'78 | 2.79 |
| Bini et al.'79 | 2.78 |
| Schönhage'80 | 2.52 |
| Romani'80 | 2.52 |
| Coppersmith,Winograd'81 | 2.50 |
| Strassen'86 | 2.48 |
| Coppersmith,Winograd'90 | 2.376 |
| Stothers'10 | 2.374 |
| Vassilevska-Williams'11 | 2.37288 |
| Le Gall'14 | **2.37287** |

# Boolean Matrix Multiplication

**Thm:** BMM is in time $O(n^\omega)$

given $n \times n$ matrices $A, B$ with entries in $\{0,1\}$

compute standard matrix product $C'$ with $C'_{i,j} = \sum_{i=1}^{n} A_{i,k} \cdot B_{k,j}$

define matrix $C$ with $C_{i,j} = [C'_{i,j} > 0]$

then $C$ is the Boolean matrix product of $A$ and $B$

**Hypothesis:** BMM is not in time $O(n^{\omega - \varepsilon})$

# Combinatorial Algorithms

fast matrix multiplication uses algebraic techniques which are impractical

"combinatorial algorithms": do not use algebraic techniques

not well defined!

| | |
|---|---|
| Arlazarov,Dinic,Kronrod, Faradzhev'70 (four russians) | $O(n^3/\log^2 n)$ |
| Bansal,Williams'09 | $O(n^3(\log\log n)^2/\log^{9/4} n)$ |
| Chan'15 | $O(n^3(\log\log n)^3/\log^3 n)$ |
| Yu'15 | $O(n^3 \text{ poly} \log\log n / \log^4 n)$ |

**Hypothesis:** BMM has no "combinatorial" algorithm in time $O(n^{3-\varepsilon})$

# In this lecture you learn that...

…(B)MM is useful for **designing theoretically fast algorithms**

- Exercise: k-Clique in $O(n^{\omega k/3})$
- Exercise: MaxCut in $O(2^{\omega n/3}\ \mathrm{poly}(n))$
- Node-Weighted Negative Triangle in $O(n^{\omega})$

…BMM is an **obstacle for practically fast / theoretically very fast algorithms**

no combinatorial $O(n^{3-\varepsilon})$ — not faster than $O(n^{2.373})$

- Transitive Closure has no $O(n^{\omega-\varepsilon})$ / combinatorial $O(n^{3-\varepsilon})$ algorithm
- Exercise: pattern matching with 2 patterns
- Sliding Window Hamming Distance
- context-free grammar problems

max planck institut informatik

# Outline

**1) Relations to Subcubic Equivalences**

2) Strassen’s Algorithm

3) Sliding Window Hamming Distance

4) Node-Weighted Negative Triangle

5) Context-Free Grammars

# Corollaries from Subcubic Equivalences

**BMM**

⇕

All-Pairs-Triangle

given an unweighted graph $G$

vertices $V = I \cup J \cup K$

$\forall i, j$: are they in a triangle with some $k$?

⇕

**Triangle**

given an unweighted graph $G$

does it contain a triangle?

**APSP**

⇕

**Min-Plus Product**

⇕

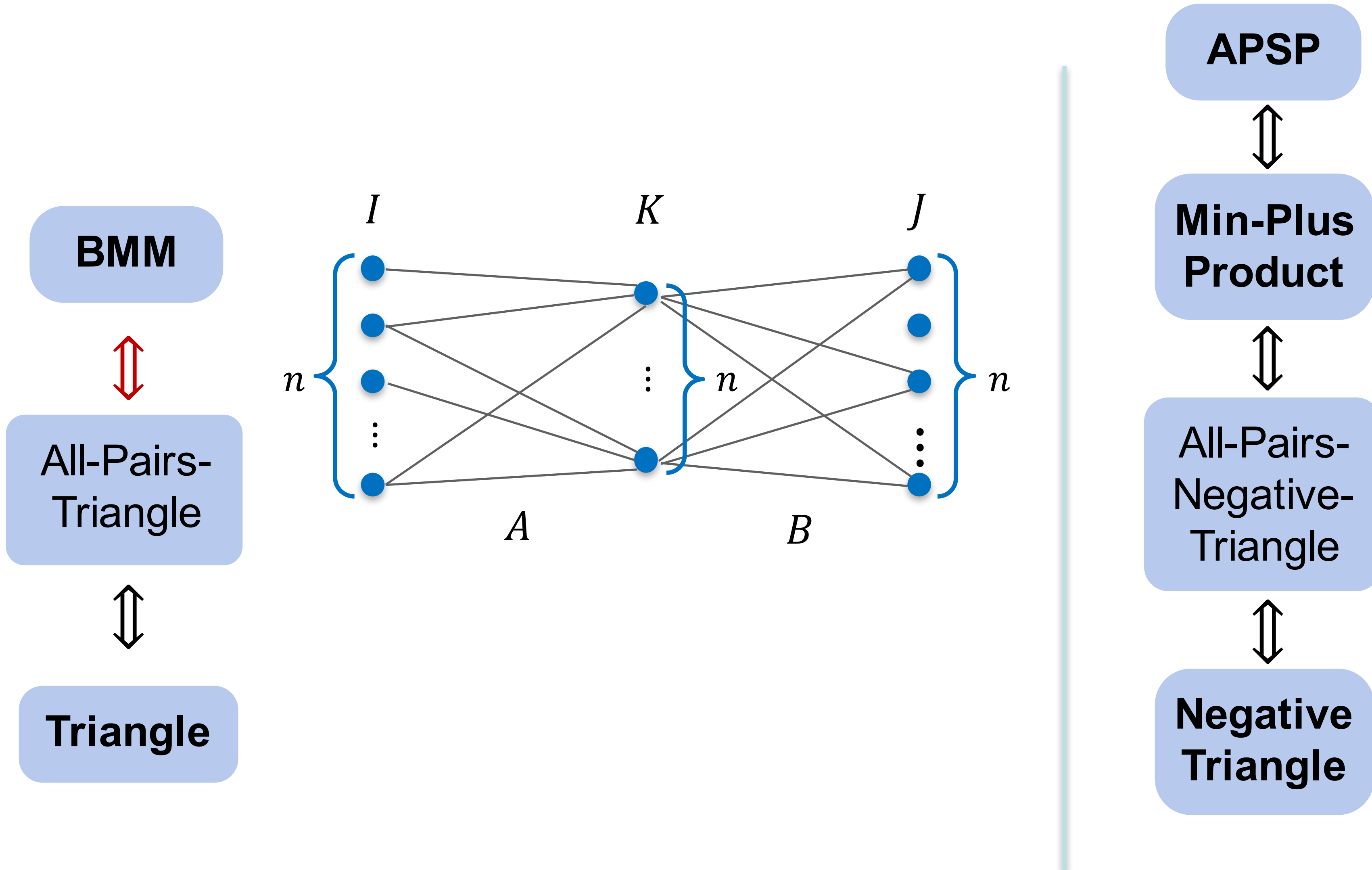
All-Pairs-Negative-Triangle

⇕

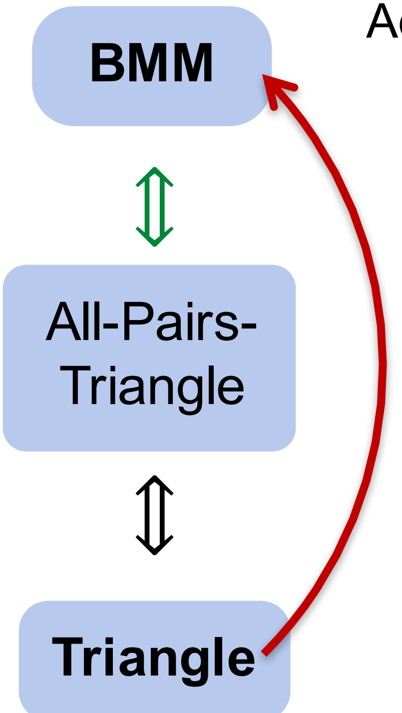
**Negative Triangle**

[Vassilevska-Williams,Williams'10]

# Corollaries from Subcubic Equivalences



# Corollaries from Subcubic Equivalences




Given an unweighted undirected graph $G$

Adjacency matrix $A$, entries in $\{0,1\}$

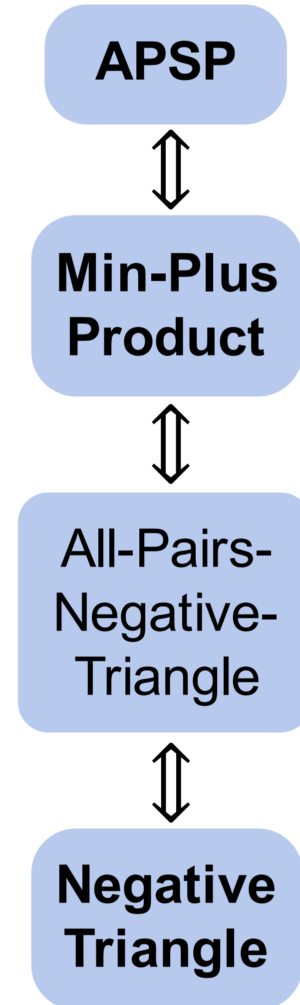
1. Compute Boolean Product $C \coloneqq A * A$:

$$C_{i,j} = \bigvee_k A_{i,k} \wedge A_{k,j}$$

2. Compute $\bigvee_{i,j} A_{i,j} \wedge C_{i,j}$

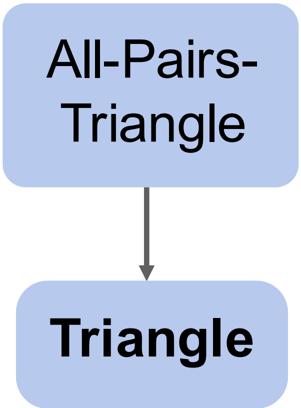
this equals $\bigvee_{i,j,k} A_{i,j} \wedge A_{i,k} \wedge A_{k,j}$

thus we solved triangle detection



# Corollaries from Subcubic Equivalences

# All-Pairs-Triangle to Triangle




**Triangle** Given graph $G$

Decide whether there are vertices $i, j, k$ such that $i, j, k$ form a triangle

**All-Pairs-Triangle** Given graph $G$ with vertex set $V = I \cup J \cup K$

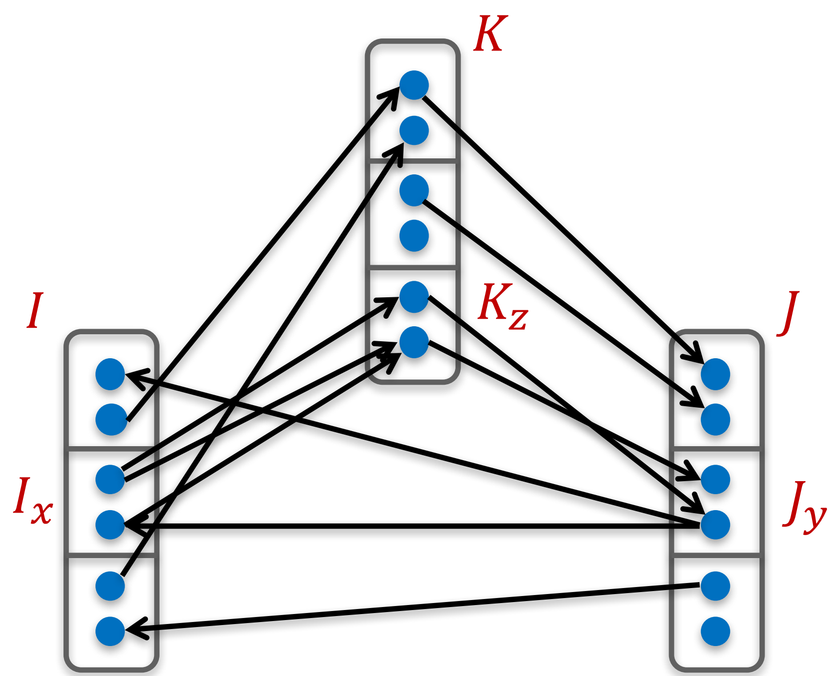
Decide for every $i \in I, j \in J$ whether there is a vertex $k \in K$ such that $i, j, k$ form a triangle

Split $I, J, K$ into $n/s$ parts of size $s$:

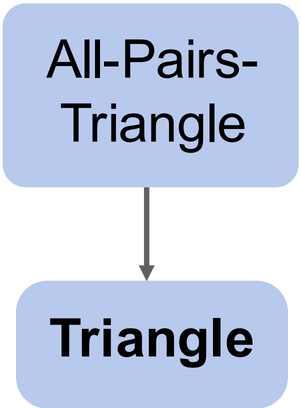
$I_1, \dots, I_{n/s}, J_1, \dots, J_{n/s}, K_1, \dots, K_{n/s}$

For each of the $(n/s)^3$ triples $(I_x, J_y, K_z)$:

consider graph $G[I_x \cup J_y \cup K_z]$



# All-Pairs-Triangle to Triangle




Initialize $C$ as $n \times n$ all-zeroes matrix

For each of the $(n/s)^3$ triples of parts $(I_x, J_y, K_z)$:

While $G[I_x \cup J_y \cup K_z]$ contains a triangle:

Find a triangle $(i,j,k)$ in $G[I_x \cup J_y \cup K_z]$

Set $C[i,j] \coloneqq 1$

Delete edge $(i,j)$

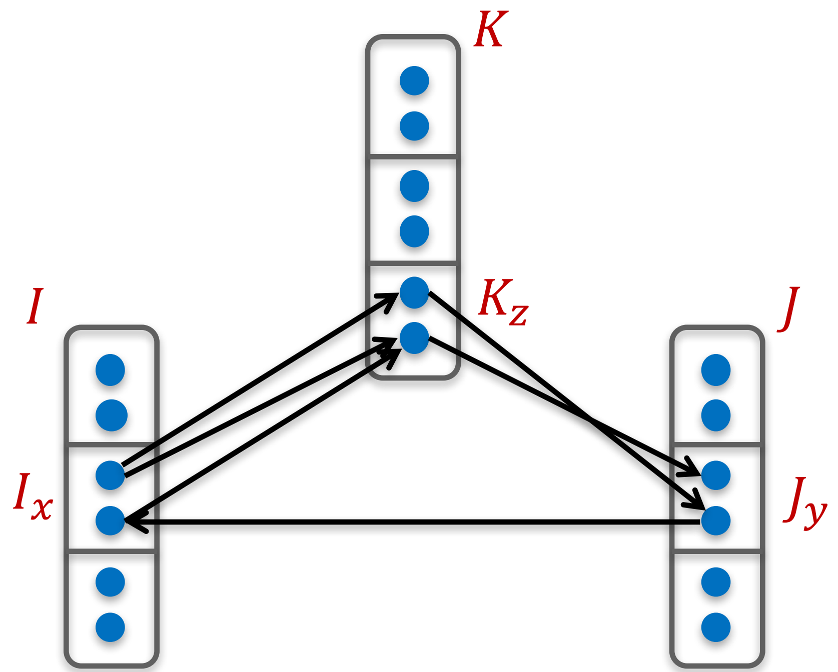
$(i,j)$ is in no more triangles

✔ guaranteed termination:

can delete $\leq n^2$ edges

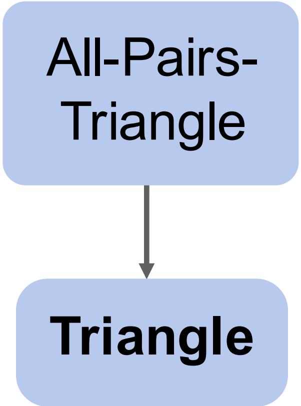
✔ correctness:

if $(i,j)$ is in a triangle,
we will find one




max planck institut
informatik

# All-Pairs-Triangle to Triangle




Find a triangle $(i, j, k)$ in $G[I_x \cup J_y \cup K_z]$

*How to **find** a triangle*
*if we can only **decide** whether one exists?*

Partition $I_x$ into $I_x^{(1)}, I_x^{(2)}$, $J_y$ into $J_y^{(1)}, J_y^{(2)}$, $K_z$ into $K_z^{(1)}, K_z^{(2)}$

Since $G[I_x \cup J_y \cup K_z]$ contains a triangle,

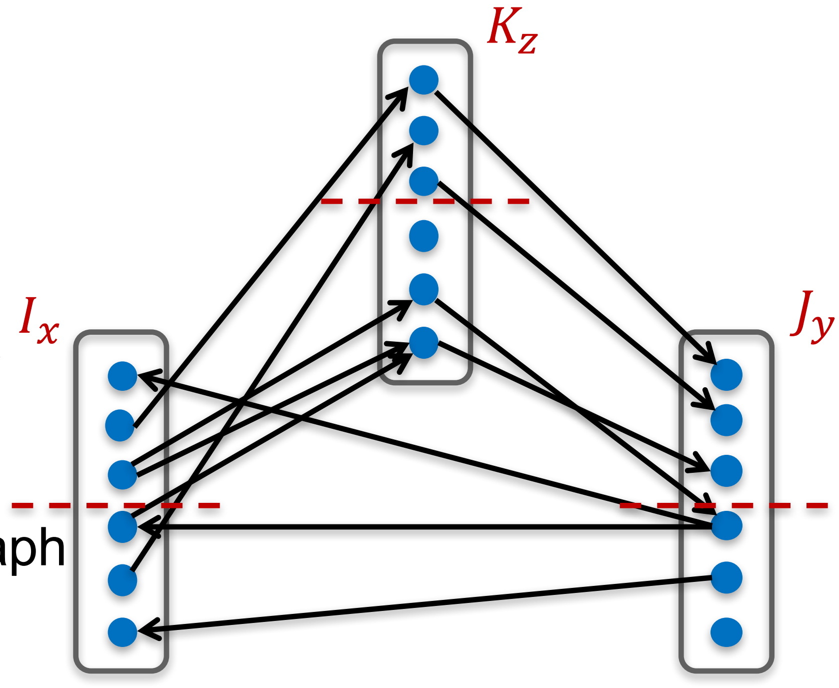
at least one of the $2^3$ subgraphs

$$G[I_x^{(a)} \cup J_y^{(b)} \cup K_z^{(c)}]$$

contains a triangle

Decide for each such subgraph whether it contains a triangle

Recursively find a triangle in one subgraph



# All-Pairs-Triangle to Triangle

All-Pairs-Triangle → **Triangle**

Find a triangle $(i, j, k)$ in $G[I_x \cup J_y \cup K_z]$

*How to **find** a triangle*
*if we can only **decide** whether one exists?*

Partition $I_x$ into $I_x{}^{(1)}, I_x{}^{(2)}$, $J_y$ into $J_y{}^{(1)}, J_y{}^{(2)}$, $K_z$ into $K_z{}^{(1)}, K_z{}^{(2)}$

Since $G[I_x \cup J_y \cup K_z]$ contains a triangle,
at least one of the $2^3$ subgraphs

$$G[I_x{}^{(a)} \cup J_y{}^{(b)} \cup K_z{}^{(c)}]$$

contains a triangle

Decide for each such subgraph whether it contains a triangle

Recursively find a triangle in one subgraph

Running Time:

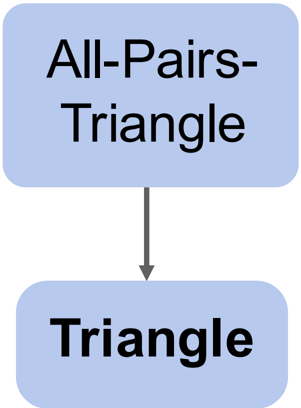
$$T_{\text{FindTriangle}}(n) \leq$$

$$2^3 \cdot T_{\text{DecideTriangle}}(n)$$

$$+ T_{\text{FindTriangle}}(n/2)$$

$$= O(T_{\text{DecideTriangle}}(n))$$

# All-Pairs-Triangle to Triangle




Initialize $C$ as $n \times n$ all-zeroes matrix

For each of the $(n/s)^3$ triples of parts $(I_x, J_y, K_z)$:

- While $G[I_x \cup J_y \cup K_z]$ contains a triangle:
  - Find a triangle $(i, j, k)$ in $G[I_x \cup J_y \cup K_z]$
  - Set $C[i, j] \coloneqq 1$
  - Delete edge $(i, j)$

$(*)$

**Running Time:**

$$(*) = O(T_{\text{FindTriangle}}(s)) = O(T_{\text{DecideTriangle}}(s))$$

Total time: $\big((\#\text{triples}) + (\#\text{triangles found})\big) \cdot (*)$

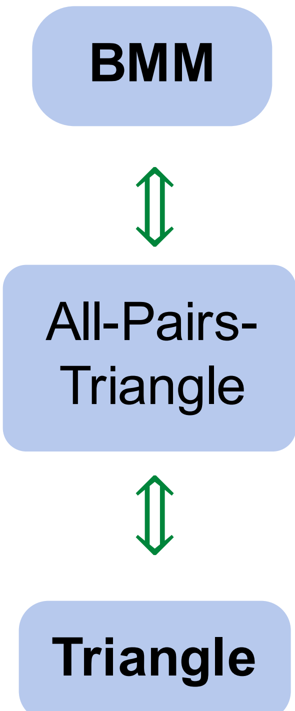
$$\leq \big((n/s)^3 + n^2\big) \cdot T_{\text{DecideTriangle}}(s)$$

Set $s = n^{1/3}$ and assume $T_{\text{DecideTriangle}}(n) = O(n^{3-\varepsilon})$

Total time: $O\big(n^2 \cdot n^{1-\varepsilon/3}\big) = O(n^{3-\varepsilon/3})$

max planck institut
informatik

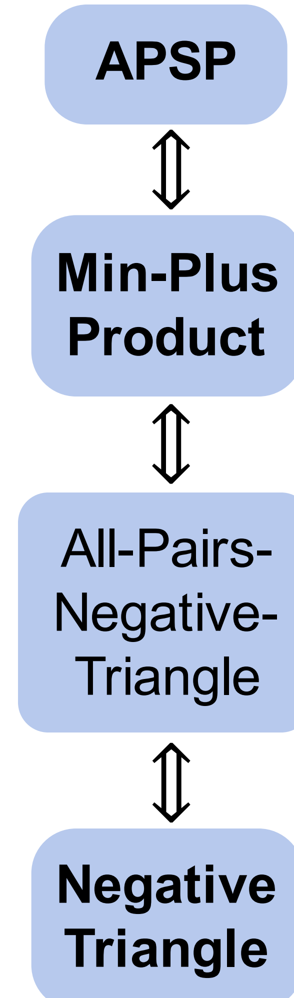
# Corollaries from Subcubic Equivalences




If BMM has (combinatorial) $O(n^{3-\varepsilon})$ algorithm
then Triangle has (combinatorial) $O(n^{3-\varepsilon})$ algorithm

If Triangle has (combinatorial) $O(n^{3-\varepsilon})$ algorithm
then BMM has (combinatorial) $O\left(n^{3-\varepsilon/3}\right)$ algorithm

→ **subcubic equivalent**,
but this mainly makes sense
for **combinatorial** algorithms



# Outline

1) Relations to Subcubic Equivalences

**2) Strassen's Algorithm**

3) Sliding Window Hamming Distance

4) Node-Weighted Negative Triangle

5) Context-Free Grammars

# Strassen‘s Algorithm

shows $\omega \leq 2.81$

$$\overbrace{\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix}}^{A} \cdot \overbrace{\begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix}}^{B} = \overbrace{\begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}}^{C}$$

$$C_{1,1} = A_{1,1} \cdot B_{1,1} + A_{1,2} \cdot B_{2,1}$$

$$C_{1,2} = A_{1,1} \cdot B_{1,2} + A_{1,2} \cdot B_{2,2}$$

$$C_{2,1} = A_{2,1} \cdot B_{1,1} + A_{2,2} \cdot B_{2,1}$$

$$C_{2,2} = A_{2,1} \cdot B_{1,2} + A_{2,2} \cdot B_{2,2}$$

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

$$T(n) \leq O(n^3)$$

# Strassen's Algorithm

shows $\omega \leq 2.81$

$$\overset{A}{\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix}} \cdot \overset{B}{\begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix}} = \overset{C}{\begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}}$$

$$M_1 = (A_{1,1} + A_{2,2}) \cdot (B_{1,1} + B_{2,2})$$

$$M_2 = (A_{2,1} + A_{2,2}) \cdot B_{1,1}$$

$$M_3 = A_{1,1} \cdot (B_{1,2} - B_{2,2})$$

$$M_4 = A_{2,2} \cdot (B_{2,1} - B_{1,1})$$

$$M_5 = (A_{1,1} + A_{1,2}) \cdot B_{2,2}$$

$$M_6 = (A_{2,1} - A_{1,1}) \cdot (B_{1,1} + B_{1,2})$$

$$M_7 = (A_{1,2} - A_{2,2}) \cdot (B_{2,1} + B_{2,2})$$

$$C_{1,1} = M_1 + M_4 - M_5 + M_7$$

$$C_{1,2} = M_3 + M_5$$

$$C_{2,1} = M_2 + M_4$$

$$C_{2,2} = M_1 - M_2 + M_3 + M_6$$

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

$$T(n) \leq O\left(n^{\log_2 7}\right) = O(n^{2.8074})$$

# Faster Matrix Multiplication

tensor = 3-dimensional matrix

**matrix multiplication tensor**:

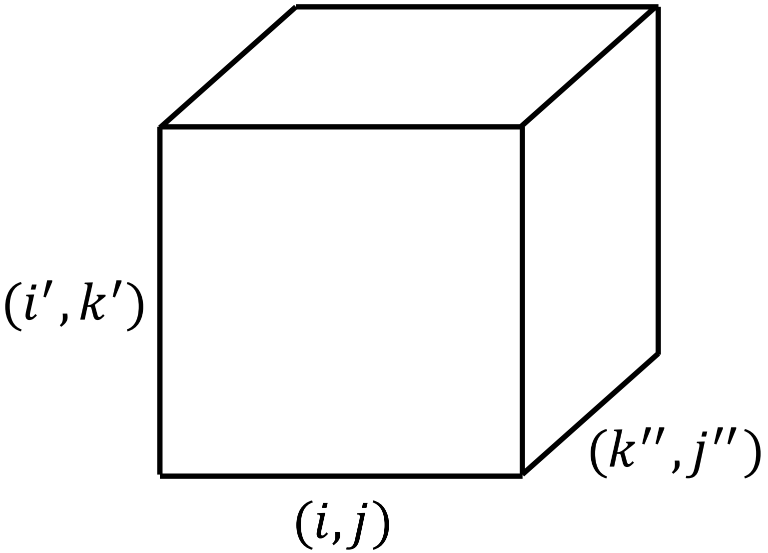
$n^2$ rows/columns/...

entries in $\{0,1\}$

entry $T_{(i,j),(i',k'),(k'',j'')} = 1$

iff $i = i'$ and $j = j''$ and $k' = k''$

i.e. $A_{i',k'} \cdot B_{k'',j''}$ appears in $C_{i,j}$




matrix of rank 1: outer product of two vectors

matrix of rank $r$: sum of $r$ rank-1-matrices

matrix rank is in P

tensor of rank 1: outer product of three vectors

tensor of rank $r$: sum of $r$ rank-1-tensors

tensor rank is not known to be in P

# Faster Matrix Multiplication

tensor = 3-dimensional matrix

**matrix multiplication tensor**:

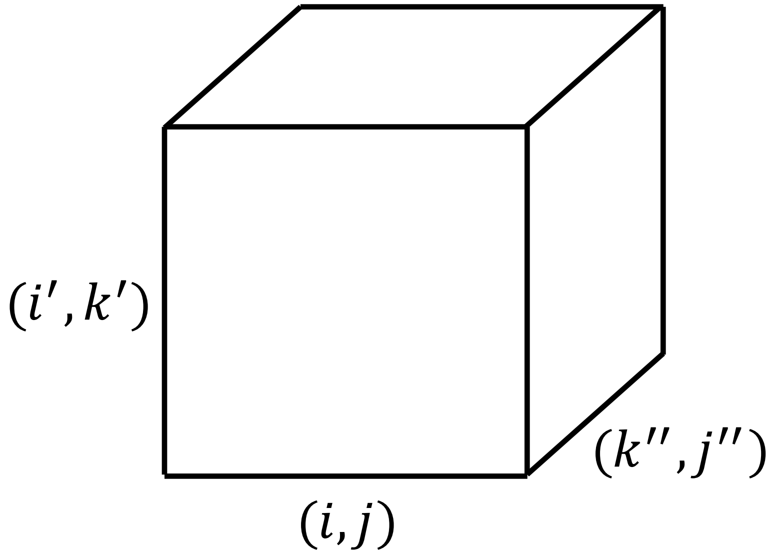
$n^2$ rows/columns/...

entries in $\{0,1\}$

entry $T_{(i,j),(i',k'),(k'',j'')} = 1$

iff $i = i'$ and $j = j''$ and $k' = k''$

i.e. $A_{i',k'} \cdot B_{k'',j''}$ appears in $C_{i,j}$




Strassen: rank of MM-tensor for $n = 2$ is at most 7

any bound on rank of MM-tensor can be transformed into a MM-algorithm

thus search for faster MM-algorithms is a mathematical question

this is **complete**: one can find $\omega$ by analyzing tensor rank!

# Outline

1) Relations to Subcubic Equivalences

2) Strassen's Algorithm

**3) Sliding Window Hamming Distance**

4) Node-Weighted Negative Triangle

5) Context-Free Grammars

# Sliding Window Hamming Distance

given two strings: text $T$ of length $n$ and pattern $P$ of length $m < n$

compute for each $i$ the Hamming distance of $P$ and $T[i..i+m-1]$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **a** | **b** | **c** | **b** | **b** | **c** | **a** | **a** | |
| b | b | c | a | | | | | 2 |
| | b | b | c | a | | | | 3 |
| | | b | b | c | a | | | 3 |
| | | | b | b | c | a | | 0 |
| | | | | b | b | c | a | 2 |

best known algorithm:

$O(n\sqrt{m}\,\mathrm{polylog}\, n)$

$\leq O(n^{1.5001})$

[Indyk,Porat,Clifford'09]

**Thm:** Sliding Window Hamming Distance has no $O(n^{\omega/2-\varepsilon})$ algorithm or combinatorial $O(n^{1.5-\varepsilon})$ algorithm unless the BMM-Hypothesis fails

$\approx O(n^{1.18})$

Open Problem: get rid of „combinatorial“ or design improved algorithm using MM

max planck institut informatik

# Sliding Window Hamming Distance

given two strings: text $T$ of length $n$ and pattern $P$ of length $m < n$

compute for each $i$ the Hamming distance of $P$ and $T[i..i+m-1]$

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \cdot B = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} = ?$$

$A$ ↓

| | | |
|---|---|---|
| 1 | x | x |
| 1 | 2 | 3 |
| x | 2 | 3 |

$B$ ↓

| | | |
|---|---|---|
| 1 | 1 | y |
| 2 | y | 2 |
| y | 3 | 3 |

alphabet: {1,2,...,$n$,x,y,$}

pattern = concat rows:

1 x x 1 2 3 x 2 3

text = concat columns + padding:

$ $ $ $ $ $ 1 2 y $ 1 y 3 $ y 2 3 $ $ $ $ $ $

1 x x 1 2 3 x 2 3

# Sliding Window Hamming Distance

| $ | $ | $ | $ | $ | $ | 1 | 2 | y | $ | 1 | y | 3 | $ | y | 2 | 3 | $ | $ | $ | $ | $ | $ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | 1 | x | x | 1 | 2 | 3 | x | 2 | 3 | | | | | | | | |
| | | | | | | | | | | 1 | x | x | 1 | 2 | 3 | x | 2 | 3 | | | | |
| | | | | | | | | | | | | | | 1 | x | x | 1 | 2 | 3 | x | 2 | 3 |
| | | | 1 | x | x | 1 | 2 | 3 | x | 2 | 3 | | | | | | | | | | | |
| | | | | | | | 1 | x | x | 1 | 2 | 3 | x | 2 | 3 | | | | | | | |
| | | | | | | | | | | | 1 | x | x | 1 | 2 | 3 | x | 2 | 3 | | | |
| 1 | x | x | 1 | 2 | 3 | x | 2 | 3 | | | | | | | | | | | | | | |
| | | | | 1 | x | x | 1 | 2 | 3 | x | 2 | 3 | | | | | | | | | | |
| | | | | | | | | 1 | x | x | 1 | 2 | 3 | x | 2 | 3 | | | | | | |

put a 1 if there is at least one match

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

# Sliding Window Hamming Distance

given Boolean $n \times n$-matrices $A, B$

we construct text+pattern of length $O(n^2)$ (in time $O(n^2)$)

thus, an $O(n^{\omega/2-\varepsilon})$ algorithm for Sliding Window Hamming Distance
would yield an $O(n^{\omega-2\varepsilon})$ algorithm for BMM, contradicting BMM-Hypothesis

and an $O(n^{1.5-\varepsilon})$ combinatorial algorithm for Sliding Window Hamming Dist.
would yield an $O(n^{3-2\varepsilon})$ combinatorial algorithm for BMM

**Thm:** Sliding Window Hamming Distance has no $O(n^{\omega/2-\varepsilon})$ algorithm or combinatorial $O(n^{1.5-\varepsilon})$ algorithm unless the BMM-Hypothesis fails

# Outline

1) Relations to Subcubic Equivalences

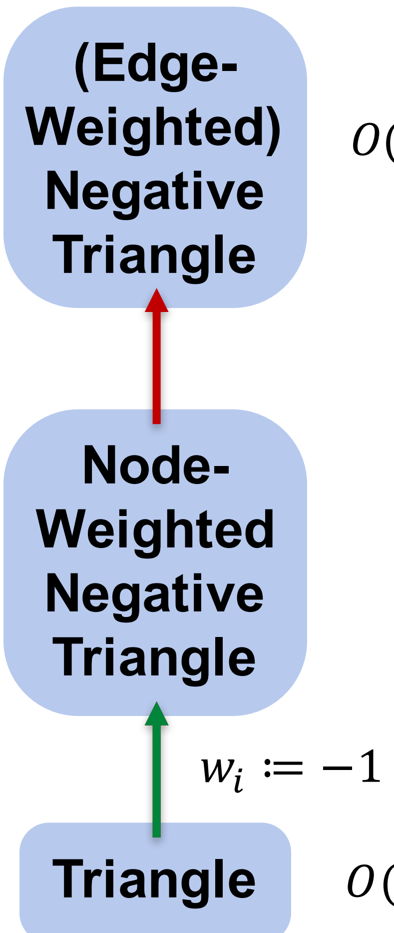
2) Strassen's Algorithm

3) Sliding Window Hamming Distance

**4) Node-Weighted Negative Triangle**

5) Context-Free Grammars

# Node-Weighted Negative Triangle




**(Edge-Weighted) Negative Triangle** — $O(n^3)$

given a directed graph with weights $w_{i,j}$ on **edges**,
is there a triangle $i, j, k$: $w_{j,i} + w_{i,k} + w_{k,j} < 0$ ?

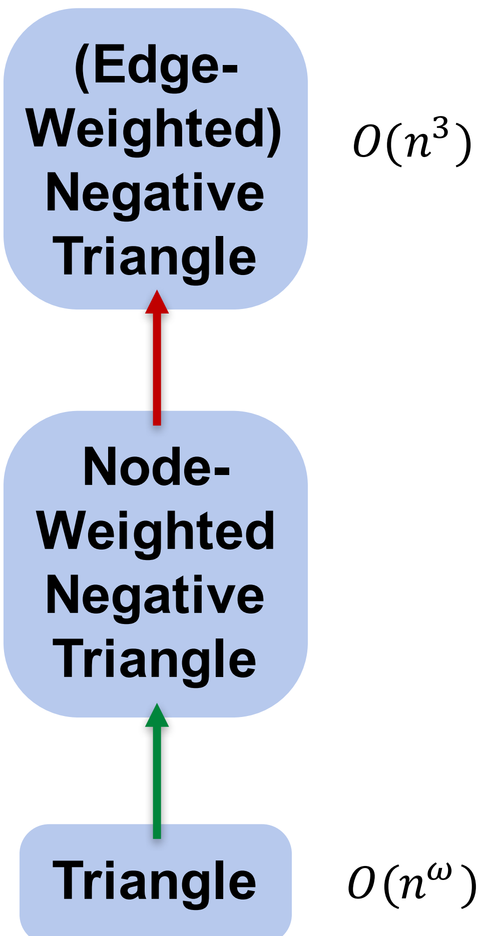
**Node-Weighted Negative Triangle**

given a directed graph with weights $w_i$ on **nodes**,
is there a triangle $i, j, k$: $w_i + w_j + w_k < 0$ ?

**Triangle** — $O(n^\omega)$

given an **unweighted** undirected graph,
is there a triangle?

# Node-Weighted Negative Triangle




find appropriate **edge** weights that simulate the given **node** weights:

set $w_{i,j} := (w_i + w_j)/2$

then for a triangle $i, j, k$:

$$w_{j,i} + w_{i,k} + w_{k,j} = w_i + w_j + w_k$$

# Node-Weighted Negative Triangle

**(Edge-Weighted) Negative Triangle** $O(n^3)$

↑

**Node-Weighted Negative Triangle** $O(n^\omega)$ [Czumaj,Lingas'07]

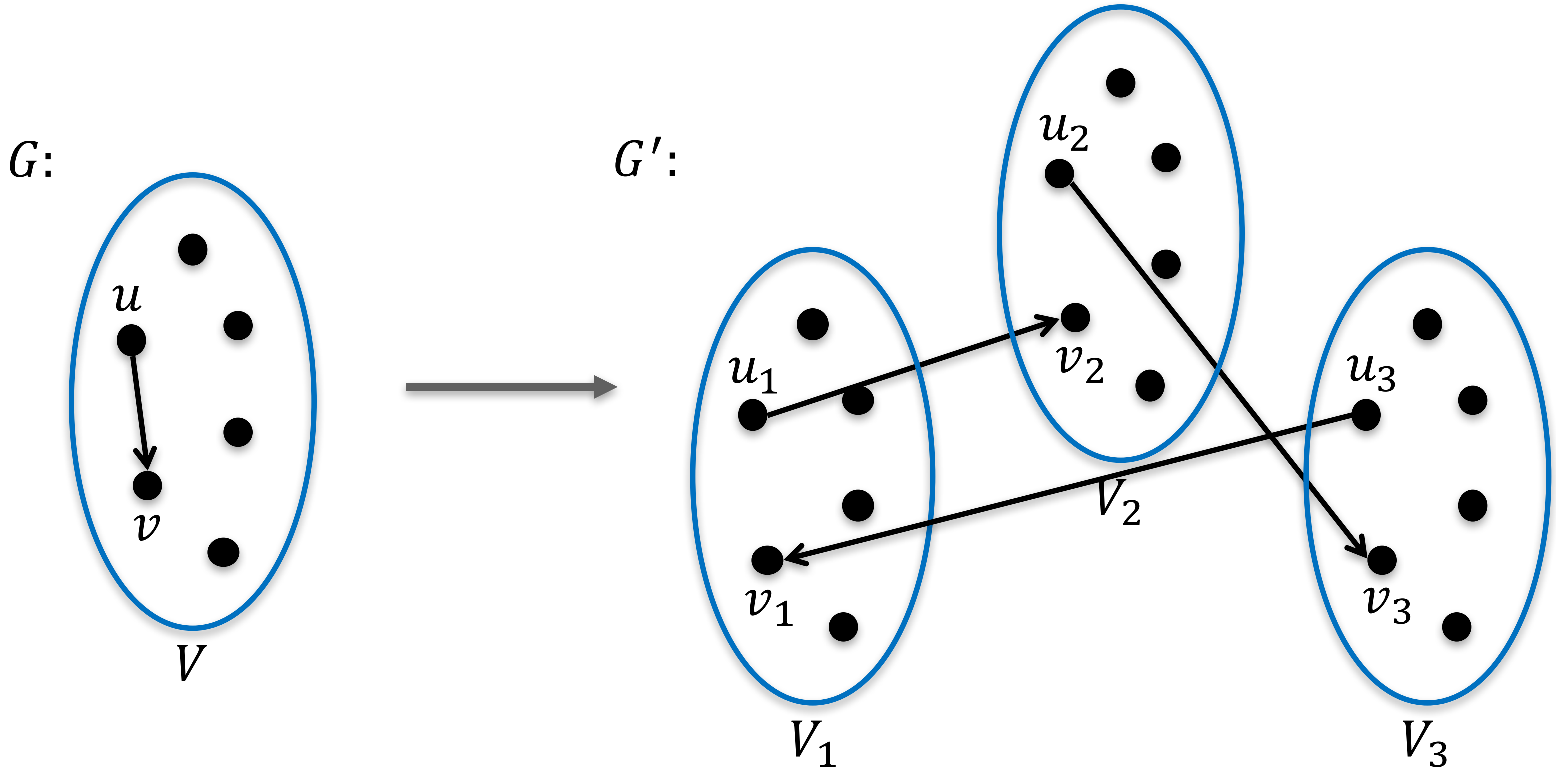
actually for **Node-Weighted Minimum Weight Triangle**

↑

**Triangle** $O(n^\omega)$

# Node-Weighted Minimum Weight Triangle

we can assume that the graph is **tripartite**:




triangle $i, j, k$ $\Leftrightarrow$ triangle $i_1, j_2, k_3$

# Node-Weighted Minimum Weight Triangle

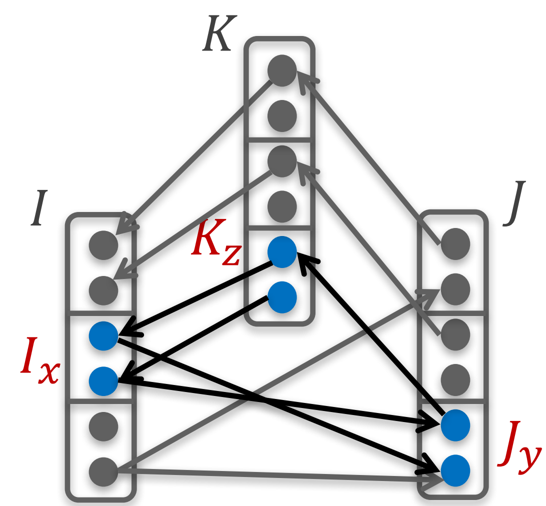
given graph $G = (V, E)$ with $V = I \cup J \cup K$ and node weights $w_v$,
compute minimum weight $q$ s.t.
there are $i \in I, j \in J, k \in K$ with $(i,j),(j,k),(k,i) \in E$ and $w_i + w_j + w_k = q$

- assume that $I, J, K$ are sorted by weight
- parameter $p$ (=sufficiently large constant)
- assume $n \coloneqq |I| = |J| = |K| = p^\ell$ for some $\ell \in \mathbb{N}$
  (add isolated dummy vertices)

# Node-Weighted Minimum Weight Triangle

given graph $G = (V, E)$ with $V = I \cup J \cup K$ and node weights $w_v$,
compute minimum weight $q$ s.t.
there are $i \in I, j \in J, k \in K$ with $(i,j),(j,k),(k,i) \in E$ and $w_i + w_j + w_k = q$

- assume that $I, J, K$ are sorted by weight
- parameter $p$ (=sufficiently large constant)
- assume $n := |I| = |J| = |K| = p^\ell$ for some $\ell \in \mathbb{N}$
  (add isolated dummy vertices)




ALG(G): 0) if $n = O(1)$ then solve in constant time

1) split $I = I_1 \cup \dots \cup I_p$, $J = J_1 \cup \dots \cup J_p$, $K = K_1 \cup \dots \cup K_p$
   (in sorted order: $\max w(I_x) \le \min w(I_{x+1})$ and so on)

2) $R := \{(x,y,z) \in \{1,\dots,p\}^3 \mid G[I_x \cup J_y \cup K_z] \text{ contains a triangle}\}$

**3) for each $(x,y,z) \in R$** s.t. there is no $(x',y',z') \in R$ with $x' < x,\ y' < y,\ z' < z$

   run ALG($G[I_x \cup J_y \cup K_z]$)

# Node-Weighted Minimum Weight Triangle

3) for each $(x, y, z) \in R$ s.t. there is no $(x', y', z') \in R$ with $x' < x,\ y' < y,\ z' < z$

run ALG($G[I_x \cup J_y \cup K_z]$)

Correctness:

if there is no triangle in $G[I_x \cup J_y \cup K_z]$ then we can ignore it

if $(x, y, z) \in R$ is dominated by $(x', y', z') \in R$:

let $i, j, k$ be a triangle in $G[I_x \cup J_y \cup K_z]$, and $i', j', k'$ a triangle in $G[I_{x'} \cup J_{y'} \cup K_{z'}]$

then $w_i + w_j + w_k \quad \geq \min w(I_x) + \min w(J_y) + \min w(K_z)$

VI VI VI

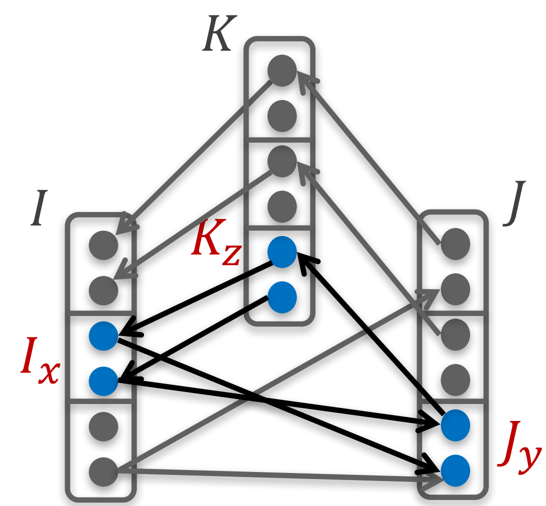
and $w_{i'} + w_{j'} + w_{k'} \leq \max w(I_{x'}) + \max w(J_{y'}) + \max w(K_{z'})$

so we can safely ignore $G[I_x \cup J_y \cup K_z]$

# Node-Weighted Minimum Weight Triangle

given graph $G = (V, E)$ with $V = I \cup J \cup K$ and node weights $w_v$,
compute minimum weight $q$ s.t.
there are $i \in I, j \in J, k \in K$ with $(i,j), (j,k), (k,i) \in E$ and $w_i + w_j + w_k = q$

- assume that $I, J, K$ are sorted by weight
- parameter $p$ (=sufficiently large constant)
- assume $n := |I| = |J| = |K| = p^\ell$ for some $\ell \in \mathbb{N}$
  (add isolated dummy vertices)




ALG(G): 0) if $n = O(1)$ then solve in constant time

1) split $I = I_1 \cup \dots \cup I_p$, $J = J_1 \cup \dots \cup J_p$, $K = K_1 \cup \dots \cup K_p$
   (in sorted order: $\max I_x \leq \min I_{x+1}$ aso.)

2) $R := \{(x,y,z) \in \{1,\dots,p\}^3 \mid G[I_x \cup J_y \cup K_z] \text{ contains a triangle}\}$ $O(p^3 n^\omega)$

3) for each $(x,y,z) \in R$ s.t. there is no $(x',y',z') \in R$ with $x' < x,\ y' < y,\ z' < z$
   run ALG($G[I_x \cup J_y \cup K_z]$) size $n/p$

how many iterations?

max planck institut informatik

# Node-Weighted Minimum Weight Triangle

3) for each $(x, y, z) \in R$ s.t. there is no $(x', y', z') \in R$ with $x' < x,\ y' < y,\ z' < z$

How many iterations?

define $\Delta_{r,s} \coloneqq \{(x, y, z) \in \{1, \dots, p\}^3 \mid x - y = r \text{ and } x - z = s\}$

$$= \{(1, 1 - r, 1 - s), (2, 2 - r, 2 - s), \dots, (p, p - r, p - s)\} \cap \{1, \dots, p\}^3$$

for $r, s \in \{-p, \dots, p\}$

the sets $\Delta_{r,s}$ cover $\{1, \dots, p\}^3$

line 3) applies to at most one $(x, y, z)$ in $\Delta_{r,s}$ for any $r, s$!

hence, there are at most $(2p)^2 = 4p^2$ recursive calls to ALG

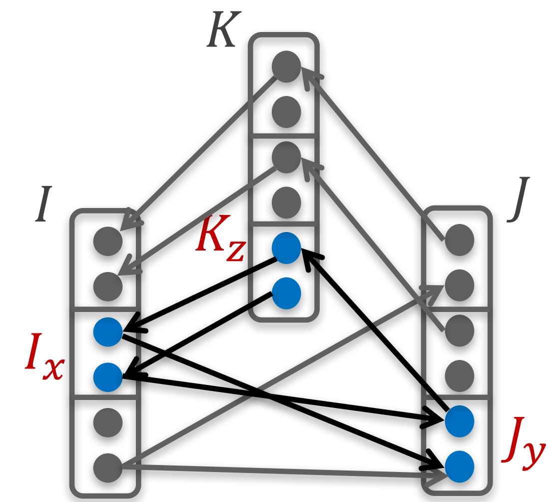


recursion: $T(n) \le 4p^2 \cdot T(n/p) + O(p^3 n^\omega)$

# Node-Weighted Minimum Weight Triangle

given graph $G = (V, E)$ with $V = I \cup J \cup K$ and node weights $w_v$,
compute minimum weight $q$ s.t.
there are $i \in I, j \in J, k \in K$ with $(i,j), (j,k), (k,i) \in E$ and $w_i + w_j + w_k = q$

- assume that $I, J, K$ are sorted by weight
- parameter $p$ (=sufficiently large constant)
- assume $n := |I| = |J| = |K| = p^\ell$ for some $\ell \in \mathbb{N}$
  (add isolated dummy vertices)




ALG(G): 0) if $n = O(1)$ then solve in constant time

1) split $I = I_1 \cup \cdots \cup I_p$, $J = J_1 \cup \cdots \cup J_p$, $K = K_1 \cup \cdots \cup K_p$
(in sorted order: $\max I_x \leq \min I_{x+1}$ aso.)

2) $R := \{(x,y,z) \in \{1, \ldots, p\}^3 \mid G[I_x \cup J_y \cup K_z] \text{ contains a triangle}\}$ $O(p^3 n^\omega)$

3) for each $(x,y,z) \in R$ s.t. there is no $(x',y',z') \in R$ with $x' < x$, $y' < y$, $z' < z$
run ALG($G[I_x \cup J_y \cup K_z]$) size $n/p$

recursion: $T(n) \leq 4p^2 \cdot T(n/p) + O(p^3 n^\omega)$

# Node-Weighted Minimum Weight Triangle

recursion: $T(n) \leq 4p^2 \cdot T(n/p) + O(p^3 n^\omega)$

$T(n) \leq 4p^2 \cdot T(n/p) + \alpha\, p^3 n^\omega$ for some constant $\alpha$

want to show: $T(n) \leq 2\alpha\, p^3 n^\omega$

plug in: $T(n) \leq 4p^2 \cdot 2\alpha\, p^3 (n/p)^\omega + \alpha\, p^3 n^\omega$

$= \alpha\, p^3 n^\omega (1 + 8p^{2-\omega})$

assume $\omega > 2$: $\leq 2\alpha\, p^3 n^\omega$ for a sufficiently large constant $p$

so we have: $T(n) \leq O(n^\omega)$

(if $\omega = 2$: show that $T(n) \leq 2\alpha\, p^3 n^{\omega+\varepsilon}$)

in total: $T(n) \leq O(n^\omega + n^{2+\varepsilon})$ for any $\varepsilon > 0$

# In this lecture you learned that...

...(B)MM is useful for **designing theoretically fast algorithms**

- Exercise: k-Clique in $O(n^{\omega k/3})$
- Exercise: MaxCut in $O(2^{\omega n/3}\ \mathrm{poly}(n))$
- Node-Weighted Negative Triangle in $O(n^{\omega})$

...BMM is an **obstacle for practically fast / theoretically very fast algorithms**

no combinatorial $O(n^{3-\varepsilon})$   not faster than $O(n^{2.373})$

- Transitive Closure has no $O(n^{\omega-\varepsilon})$ / combinatorial $O(n^{3-\varepsilon})$ algorithm
- Exercise: pattern matching with 2 patterns
- Sliding Window Hamming Distance
- context-free grammar problems