Fine-Grained Complexity and Algorithm Design for Graph Reachability and Distance Problems

Karl Bringmann

"Fine-Grained Complexity, Logic, and Query Evaluation"
@ Simons Institute

September 28, 2023

UNIVERSITÄT DES SAARLANDES
max planck institut informatik
European Research Council
Established by the European Commission

Talk Outline

Hung's invitation:

- survey-ish talk about ***recursive query evaluation*** *from algorithms perspective*

- reachability problems (connected components, transitive closure, …)

- distance problems (shortest paths, diameter, …)

Many Problem Variants

Input: graph $G = (V, E)$

What type of graph?

undirected vs directed

weighted vs unweighted

encoding of weights, negative cycles?, …

Which parameters for measuring time?

$n =$ number of nodes

$m =$ number of edges

output size, range of weights, …

Reachability

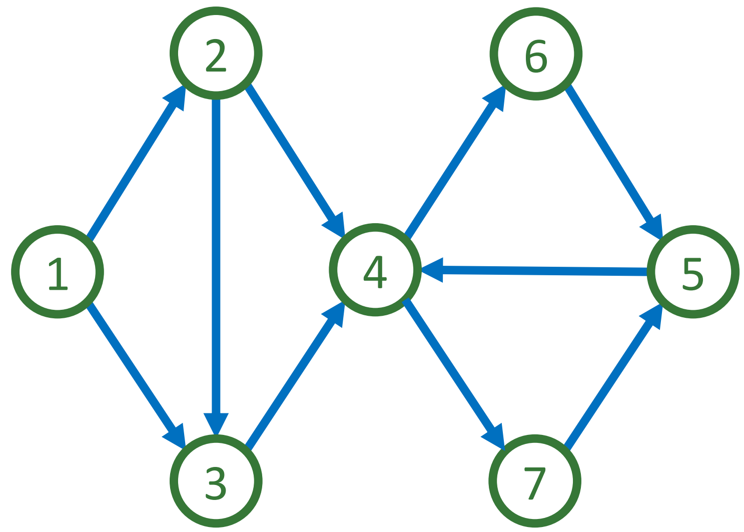
Single-Source Reachability

given a node s, compute all nodes that are reachable from s

Classic optimal algorithm:

Run depth-first-search from s

linear time $O(n + m)$



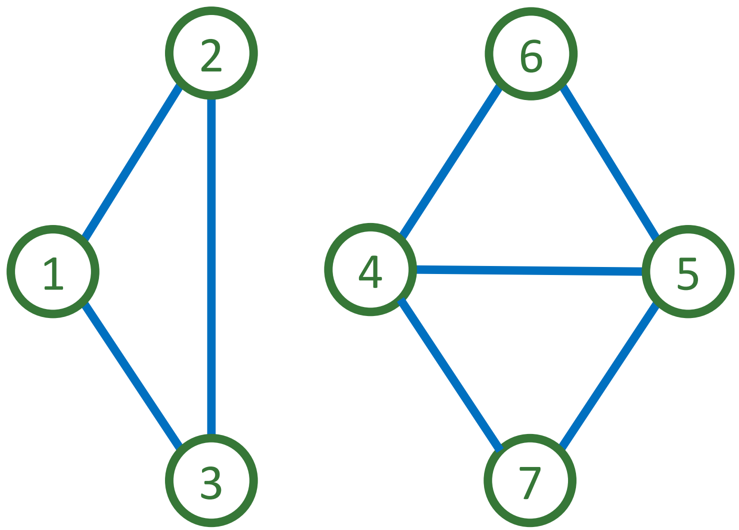
All-Pairs Reachability

compute for all nodes u, v whether u can reach v

Undirected graphs → connected components

Run depth-first-search from every unexplored node

linear time $O(n + m)$



All-Pairs Reachability

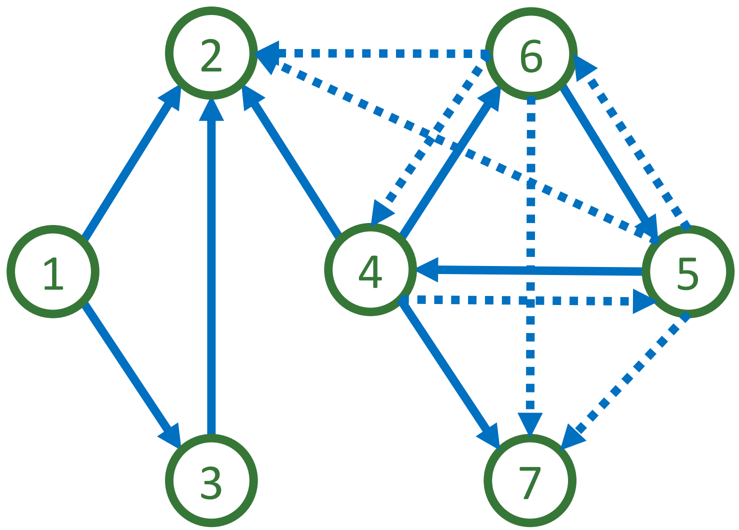
compute for all nodes u, v whether u can reach v

Directed graphs → transitive closure, parameter m:

Run single-source reachability from every node

time $O(nm) \leq O(m^2)$

optimal since output size can be up to $\Omega(m^2)$



All-Pairs Reachability

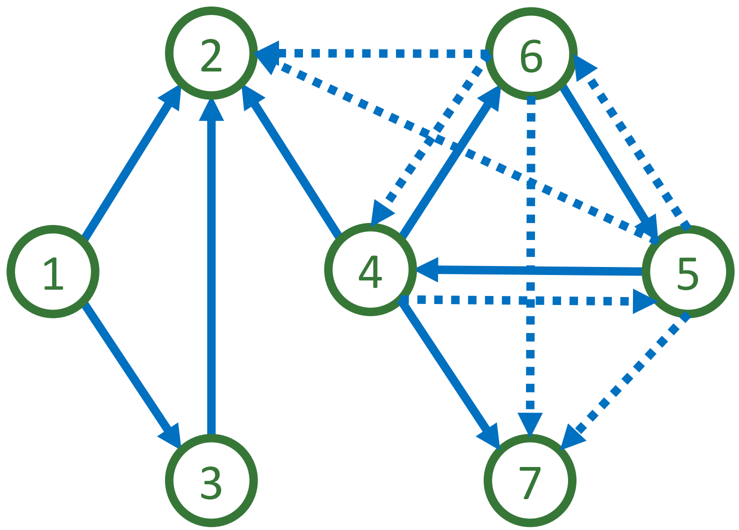
compute for all nodes u, v whether u can reach v

Directed graphs → transitive closure, parameter n:

Run single-source reachability from every node

time $O(nm) \leq O(n^3)$

equivalent to Boolean matrix multiplication



All-Pairs Reachability

compute for all nodes u, v whether u can reach v

Directed graphs → transitive closure, parameter n:

Transitive Closure

given directed n-node graph,
compute for all nodes u, v
whether u can reach v

→

Boolean Matrix Mult, BMM

given $n \times n$ matrices A, B,
compute matrix C with
$C[i,j] = \bigvee_k A[i,k] \wedge B[k,j]$

$A :=$ adjacency matrix plus selfloops
for $i = 1, \ldots, \log n$:
    $A :=$ Boolean matrix product $A * A$

→ compute transitive closure in time $\tilde{O}(n^{\omega})$

All-Pairs Reachability

compute for all nodes u, v whether u can reach v

Directed graphs → transitive closure, parameter n:

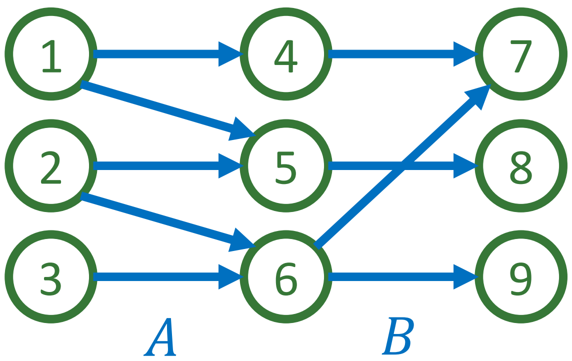
Transitive Closure

given directed n-node graph, compute for all nodes u, v whether u can reach v

←

Boolean Matrix Mult, BMM

given $n \times n$ matrices A, B, compute matrix C with $C[i,j] = \bigvee_k A[i,k] \wedge B[k,j]$



From the transitive closure of this graph we can read off the Boolean matrix product $A * B$

All-Pairs Reachability

compute for all nodes u, v whether u can reach v

Directed graphs → transitive closure, parameter n:

Transitive Closure		Boolean Matrix Mult, BMM
given directed n-node graph, compute for all nodes u, v whether u can reach v	$\equiv$	given $n \times n$ matrices A, B, compute matrix C with $C[i,j] = \bigvee_k A[i,k] \wedge B[k,j]$
$\tilde{O}(n^\omega)$		$\tilde{O}(n^\omega)$

All-Pairs Reachability

compute for all nodes u, v whether u can reach v

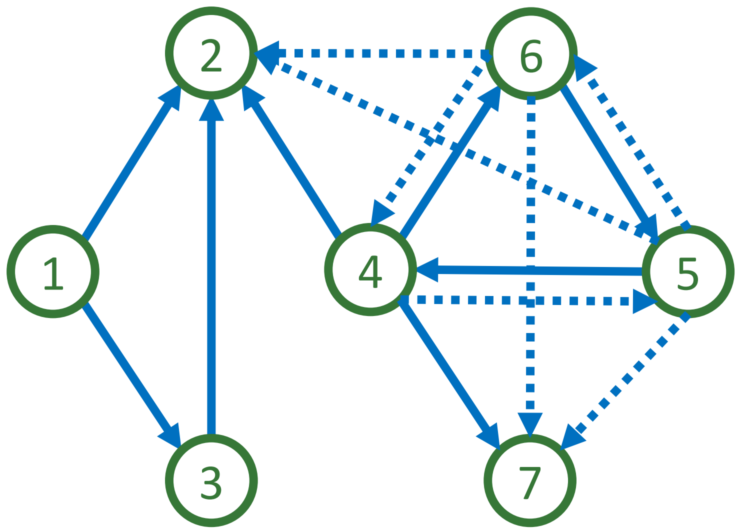
Directed graphs → transitive closure, parameter m:

optimal time $O(m^2)$, by output size bound

Directed graphs → transitive closure, parameter n:

optimal time $\tilde{O}(n^\omega)$, by equivalence with Boolean matrix product

parameter out = number of edges in transitive closure ?



All-Pairs Reachability

compute for all nodes u, v whether u can reach v

Directed graphs → transitive closure, parameter out:

Transitive Closure

out = number of edges in transitive closure

$\tilde{O}(out^c)$

≡

Boolean Matrix Mult, BMM

in = number of nonzero entries in input matrices

out = number of nonzero entries in product matrix

$\tilde{O}\big((in + out)^c\big)$

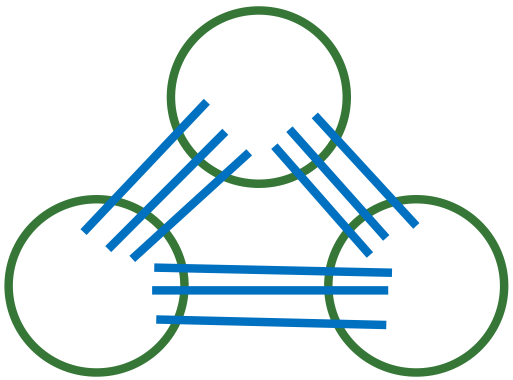
Fully-Sparse BMM

solve BMM in time $\tilde{O}((in + out)^c)$ *where* in / out *= # nonzeros in input / output*

	with current ω:	assuming $\omega = 2$:	
$c \geq \omega/2$	≥ 1.18	≥ 1	
$c \leq 1.5$	≤ 1.5	≤ 1.5	[van Gucht, Williams, Woodruff, Zhang ’15]
$c \leq \frac{2\omega}{\omega + 1}$	≤ 1.41	$\leq 4/3$	[Amossen, Pagh ‘09]
$c \leq 1 + \frac{\mu}{1 + \mu}$	≤ 1.3459	$\leq 4/3$	[Abboud, **B**, Fischer, Künnemann ‘23+]

where $\omega(\mu, 1, 1) = 2\mu + 1$

$0.5 \leq \mu \leq 0.5286$



Fully-Sparse BMM

solve BMM in time $\tilde{O}\left((in + out)^c\right)$ *where* in */* out *= # nonzeros in input / output*

	with current ω:	assuming $\omega = 2$:	
$c \geq \omega/2$	≥ 1.18	≥ 1	
$c \leq 1.5$	≤ 1.5	≤ 1.5	[van Gucht, Williams, Woodruff, Zhang '15]
$c \leq \frac{2\omega}{\omega+1}$	≤ 1.41	$\leq 4/3$	[Amossen, Pagh '09]
$c \leq 1 + \frac{\mu}{1+\mu}$	≤ 1.3459	$\leq 4/3$	[Abboud, **B**, Fischer, Künnemann '23+]

↓

deterministic algorithm for BMM

also works for integer matrix mult, but randomized

where $\omega(\mu, 1{,}1) = 2\mu + 1$

$0.5 \leq \mu \leq 0.5286$

Fully-Sparse BMM – Further Improvements?

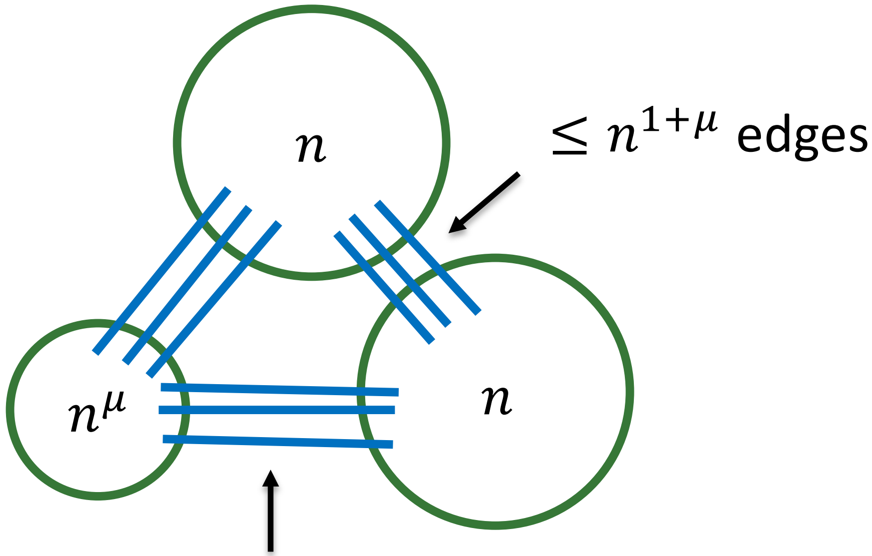
solve BMM in time $\tilde{O}\big((in + out)^c\big)$ *where* in */* out *= # nonzeros in input / output*

BMM has algorithm with exponent $c < 1 + \frac{\mu}{1+\mu}$

$\Leftrightarrow$

AllEdgesTriangle$(n^\mu, n, n; n^{1+\mu})$ can be solved in time $O(n^{1+2\mu-\varepsilon})$ for $\varepsilon > 0$

[Abboud, **B**, Fischer, Künnemann '23+]



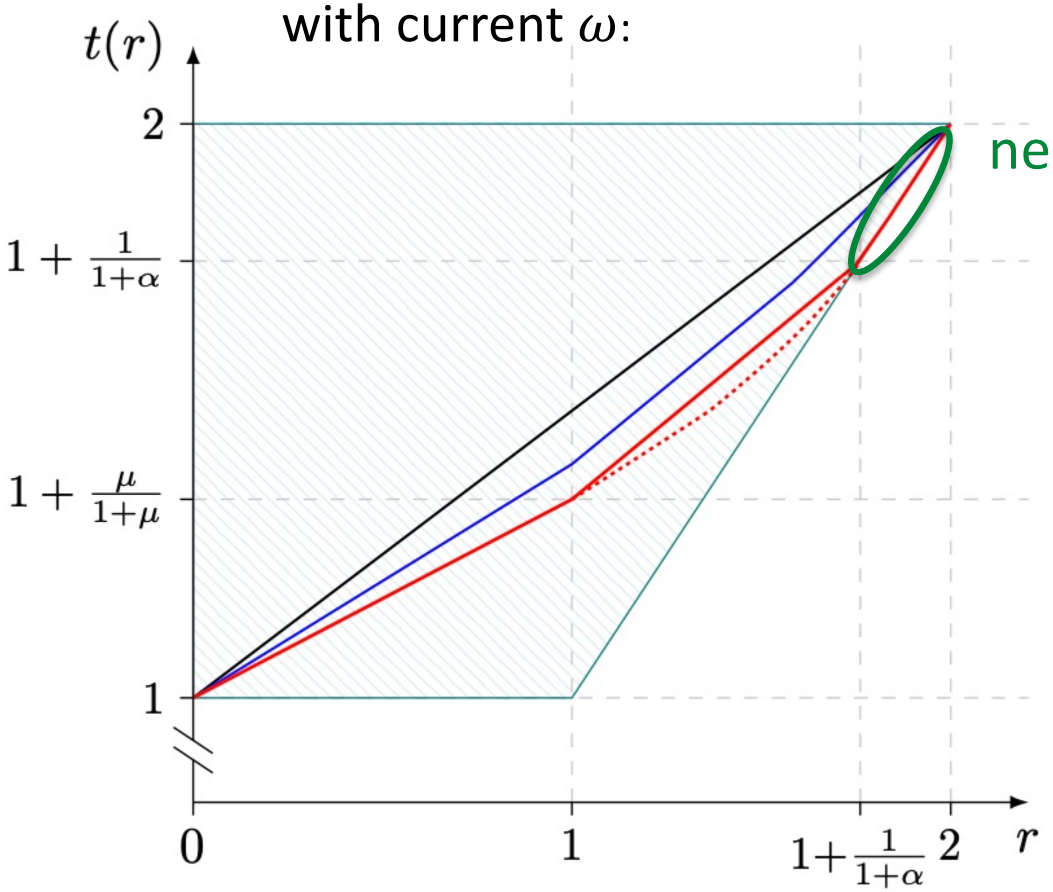
$0.5 \leq \mu \leq 0.5286$

Fully-Sparse BMM – General Tradeoff

our bound $\tilde{O}\left((in + out)^{1.3459}\right)$ is optimized for $out \approx in$

general setting: $out \approx in^r$ for some $r \in [0,2]$

with current ω:



near-linear time $\tilde{O}(out)$ if $out \geq in^{1.762}$

[van Gucht, Williams, Woodruff, Zhang '15] $\tilde{O}(in\sqrt{out})$

[Amossen, Pagh '09]

[Abboud, **B**, Fischer, Künnemann '23+]

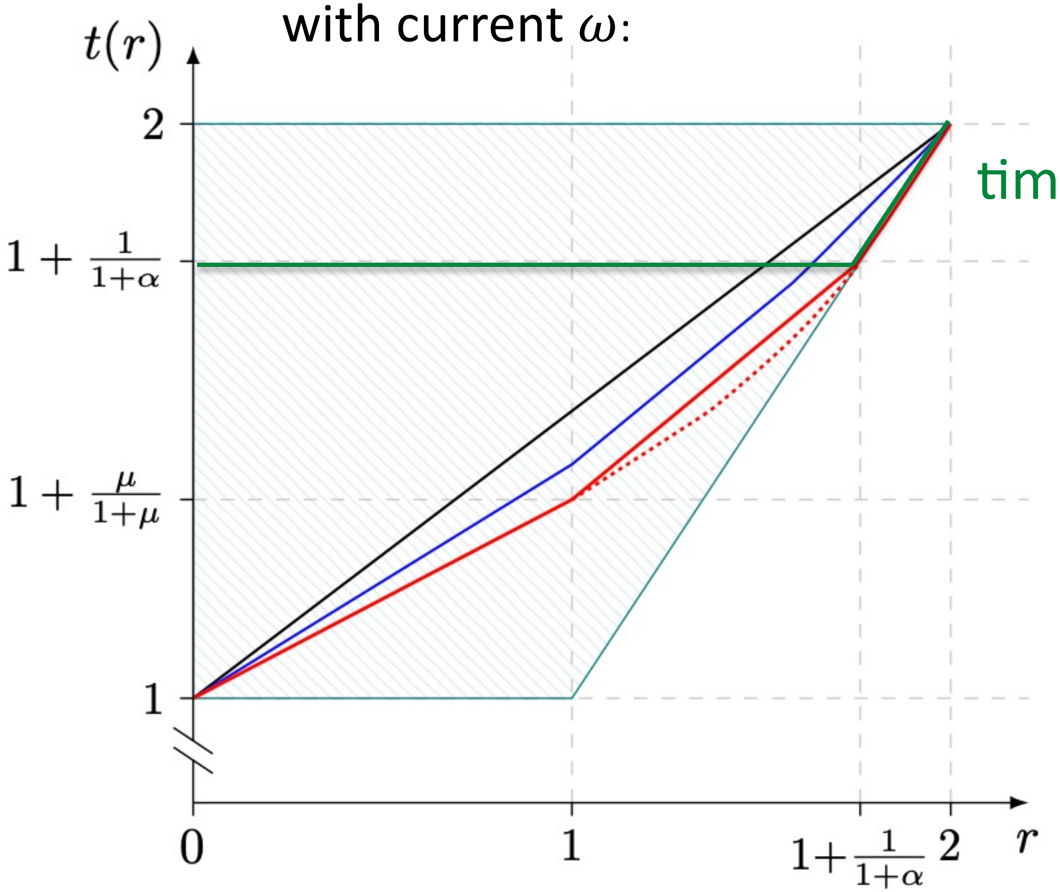
$\tilde{O}(in \cdot out^{0.3459} + in^{0.8002} out^{0.5457} + out)$

Fully-Sparse BMM – General Tradeoff

our bound $\tilde{O}\left((in + out)^{1.3459}\right)$ is optimized for $out \approx in$

general setting: $out \approx in^r$ for some $r \in [0,2]$

with current ω:



time $\tilde{O}(in^{1.762} + out)$

[van Gucht, Williams, Woodruff, Zhang '15]
$\tilde{O}(in\sqrt{out})$

[Amossen, Pagh '09]

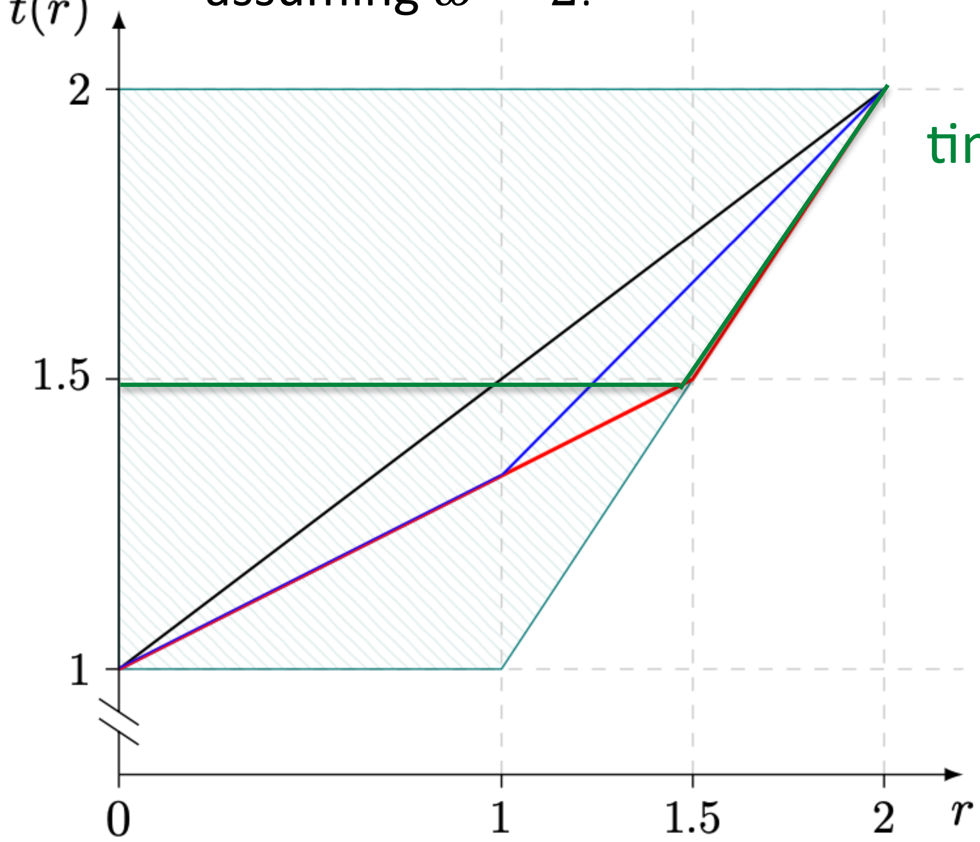
[Abboud, **B**, Fischer, Künnemann '23+]
$\tilde{O}(in \cdot out^{0.3459} + in^{0.8002} out^{0.5457} + out)$

Fully-Sparse BMM – General Tradeoff

our bound $\tilde{O}\left((in + out)^{1.3459}\right)$ is optimized for $out \approx in$

general setting: $out \approx in^r$ for some $r \in [0,2]$

assuming $\omega = 2$:



time $\tilde{O}(in^{1.5} + out)$

[van Gucht, Williams, Woodruff, Zhang '15]
$\tilde{O}(in\sqrt{out})$

[Amossen, Pagh '09]

[Abboud, **B**, Fischer, Künnemann '23+]
$\tilde{O}(in \cdot out^{1/3} + out)$

Fully-Sparse BMM – Algorithm Overview

A is $x \times y$-matrix, B is $y \times z$-matrix

1. Output Densification:

use hashing / sparse recovery to reduce outer dimensions to $x \cdot z = O(out)$

2. High-degree/low-degree:

split y's into degree higher than Δ or lower than Δ

low degree: enumerate all 2-paths in time $O(in \cdot \Delta)$

high degree: matrix multiplication in time $\text{MM}(x, y_H, z)$

$$\leq \text{MM}\left(x, \frac{in}{\Delta}, \frac{out}{x}\right) \qquad \Delta \leq x \leq \frac{out}{\Delta}$$

$$\leq \text{MM}\left(\Delta, \frac{in}{\Delta}, \frac{out}{\Delta}\right)$$

use bounds on MM to bound both terms and balance their sum

All-Pairs Reachability

compute for all nodes u, v whether u can reach v

Directed graphs → transitive closure, parameter out:

Transitive Closure

out = number of edges in transitive closure

$\tilde{O}(out^{1.3459})$

≡

Boolean Matrix Mult, BMM

in = number of nonzero entries in input matrices

out = number of nonzero entries in product matrix

$\tilde{O}\left((in + out)^{1.3459}\right)$

[Abboud, **B**, Fischer, Künnemann '23+]

Q: What is the optimal exponent?

Distances

Weight Encoding

each edge e has a weight/length $w(e)$

RAM model: each edge weight fits into a machine cell

arithmetic operations on two machine cells in time $O(1)$

1. integer weights in $\{-W, \dots, W\}$

1.1. near-constant weights: W factors in running time are okay

1.2. polynomial weights: $W \leq n^{O(1)}$, $\log W$ factors hidden by $\tilde{O}$

1.3. mildly superpolynomial weights: $\log W$ factors are okay

1.4. strongly polynomial algorithms: running time independent of W

2. real weights

2.1. RealRAM: arithmetic operations on reals in constant time

2.2. floating point approximation, e.g. $O(\log(n/\varepsilon))$-bit mantissa and exponent

Single-Source Shortest Paths

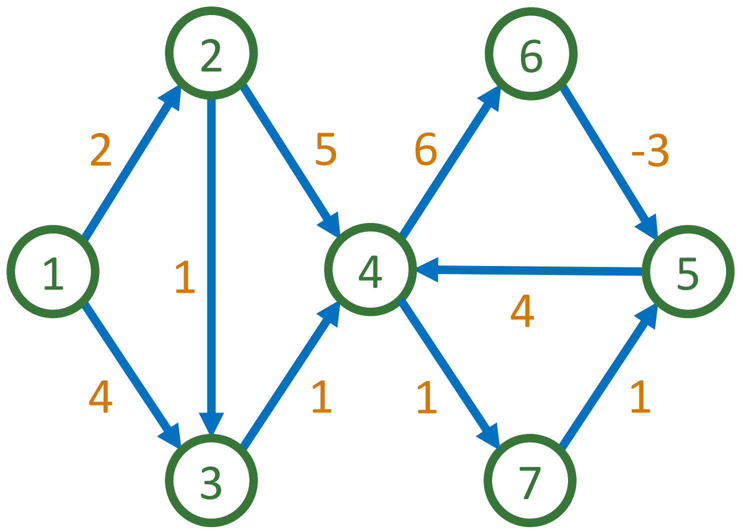
given a node s, compute distances from s to all other nodes

nonnegative edge weights:

Dijkstra's algorithm: $\tilde{O}(m) = O(m + n \log n)$

general edge weights:

Bellman-Ford algorithm: $O(mn)$ [Ford' 56, Bellman '58]



Single-Source Shortest Paths

given a node s, compute distances from s to all other nodes

nonnegative edge weights:

Dijkstra's algorithm: $\tilde{O}(m) = O(m + n \log n)$

general edge weights:

Q: Can the $\log W$ factor be removed?

Bellman-Ford algorithm: O … '58]

scaling-based algorithms: $O(m\sqrt{n} \log W)$

[Gabow '83, Gabow, Tarjan '89, Goldberg '95]

recent breakthrough: $\tilde{O}(m \log W) = O(m \log^8 n \log W)$

[Bernstein, Nanongkai, Wulff-Nilsen FOCS'22 best paper]

further improvements: $O\big((m + n \log\log n) \log^2 n \log(nW)\big)$

[**B**, Cassis, Fischer FOCS'23]

All-Pairs Shortest Paths

compute all pairwise distances in a graph

negative edge weights can be removed in time $O(nm)$ [Johnson'77]

parameter m:

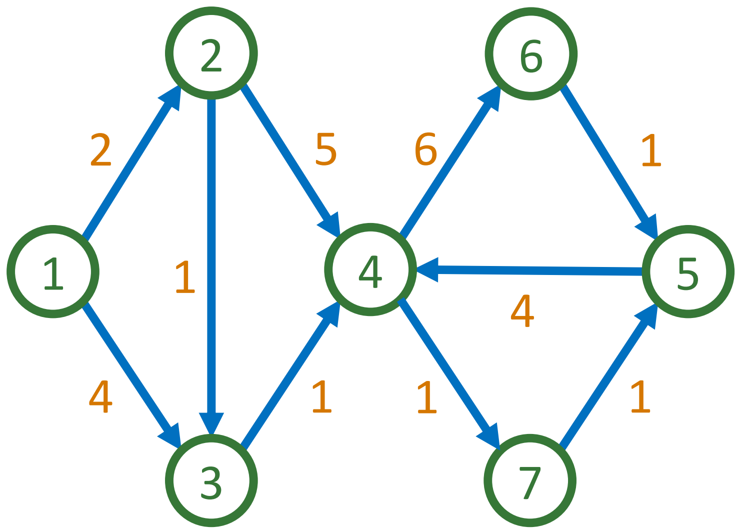
Run single-source shortest paths from every node

time $\tilde{O}(nm) \leq \tilde{O}(m^2)$, optimal by output size

parameter n:

time $\tilde{O}(nm) \leq \tilde{O}(n^3)$

equivalent to MinPlusProduct



All-Pairs Shortest Paths

compute all pairwise distances in a graph

All-Pairs Shortest Paths

given a directed graph,
compute for all nodes u, v
the distance from u to v

$\tilde{O}(n^3)$

→

MinPlusProduct

given $n \times n$ matrices A, B,
compute matrix C with

$$C[i,j] = \min_k A[i,k] + B[k,j]$$

$\tilde{O}(n^3)$

$A \coloneqq$ weighted adjacency matrix plus 0-weight selfloops

for $i = 1, \dots, \log n$:

   $A \coloneqq$ MinPlus matrix product $A * A$

All-Pairs Shortest Paths

compute all pairwise distances in a graph

All-Pairs Shortest Paths

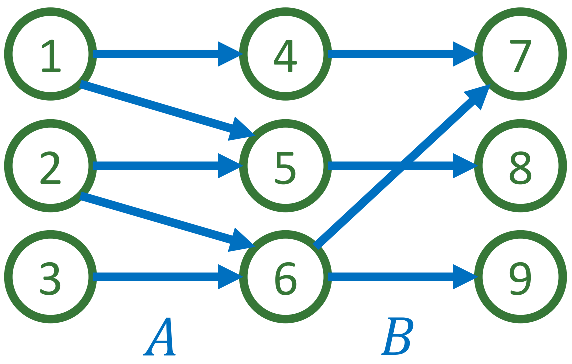
given a directed graph,
compute for all nodes u, v
the distance from u to v

$\tilde{O}(n^3)$

MinPlusProduct

given $n{\times}n$ matrices A, B,
compute matrix C with
$C[i,j] = \min_k A[i,k] + B[k,j]$

$\tilde{O}(n^3)$



From the pairwise distances in this graph we can read off the MinPlus matrix product $A * B$

All-Pairs Shortest Paths

compute all pairwise distances in a graph

All-Pairs Shortest Paths

given a directed graph,
compute for all nodes u, v
the distance from u to v

$\tilde{O}(n^3)$

$\equiv$

MinPlusProduct

given $n \times n$ matrices A, B,
compute matrix C with
$C[i,j] = \min_k A[i,k] + B[k,j]$

$\tilde{O}(n^3)$

[Vassilevska Williams, Williams '10] $\equiv_3$

Negative Triangle

given an edge-weighted graph,
are there nodes x, y, z with
$w(x,y) + w(y,z) + w(z,x) < 0$?

APSP Hypothesis:

These problems cannot
be solved in time $O(n^{3-\delta})$

All-Pairs Shortest Paths

compute all pairwise distances in a graph

parameter m:

Run single-source shortest paths from every node

time $\tilde{O}(nm) \le \tilde{O}(m^2)$, optimal by output size

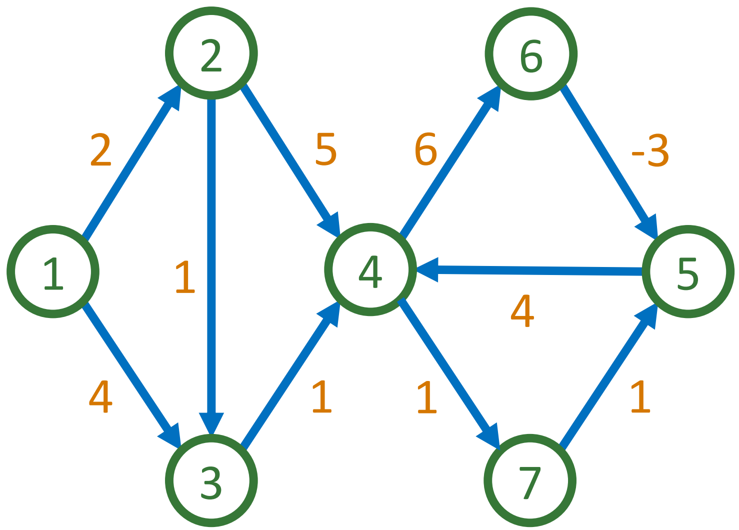
parameter n:

time $\tilde{O}(nm) \le \tilde{O}(n^3)$

equivalent to MinPlusProduct

optimality is the APSP hypothesis

$n^3/2^{\Omega(\sqrt{\log n})}$ [Williams '14]

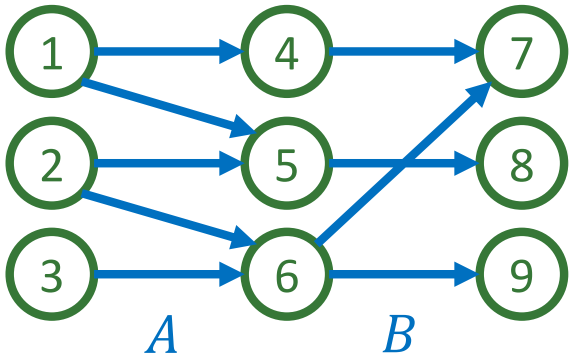


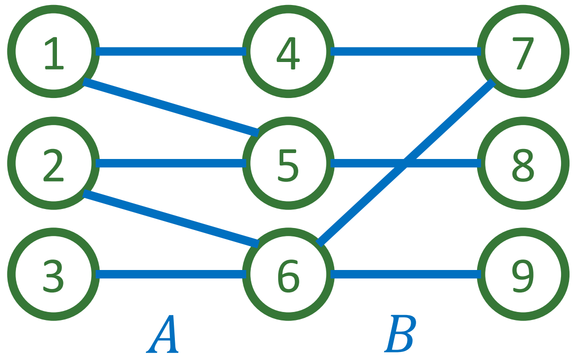
Approximate All-Pairs Shortest Paths

compute α-approximation of all pairwise distances in a graph

directed graph OR undirected graph and $\alpha < 2$:

time $\Omega(n^\omega)$, since at least as hard as BMM





Approximate All-Pairs Shortest Paths

compute α-approximation of all pairwise distances in a graph

directed graph OR undirected graph and $\alpha < 2$:

time $\Omega(n^\omega)$, since at least as hard as BMM

$(1+\varepsilon)$-approximation: time $\tilde{O}\left(\frac{n^\omega}{\varepsilon} \log W\right)$ [Zwick '02]

$(1+\varepsilon)$-approximate All-Pairs Shortest Paths	$\equiv$	**$(1+\varepsilon)$-approximate MinPlusProduct**

Is $\log W$ factor necessary?

Approximate All-Pairs Shortest Paths

compute α-approximation of all pairwise distances in a graph

directed graph OR undirected graph and $\boldsymbol{\alpha < 2}$:

time $\Omega(n^\omega)$, since at least as hard as BMM

$(\mathbf{1}+\boldsymbol{\varepsilon})$-approximation: time $\tilde{O}\left(\frac{n^\omega}{\varepsilon}\log W\right)$ [Zwick '02]

.. in undirected graphs: time $\tilde{O}\left(\frac{n^\omega}{\varepsilon}\right)$ [**B**, Künnemann, Wegrzycki STOC'19]

.. in directed graphs: time $\tilde{O}\left(\frac{n^{(3+\omega)/2}}{\varepsilon}\right)$ [**B**, Künnemann, Wegrzycki STOC'19]

equivalent to exact MinMaxProduct,
for which best known time is $\tilde{O}\left(n^{(3+\omega)/2}\right)$

Approximate All-Pairs Shortest Paths

compute α-approximation of all pairwise distances in a graph

directed graph OR undirected graph and $\alpha < 2$:

time $\Omega(n^\omega)$, since at least as hard as BMM

$O(1)$-approximation in undirected graphs:

preprocess given graph in time $O(mn^{1/k})$, $k = O(1)$ in [Thorup, Zwick '05]

then query(u, v) returns a $(2k-1)$-approximation of dist(u, v)

in query time $O(1)$

Under 3SUM, in the same preprocessing time and $n^{o(1)}$ query time we cannot compute a $< k$-approximation $\rightarrow$ ***hardness of approximation in P***

[Abboud, **B**, Khoury, Zamir STOC'22] [Abboud, **B**, Fischer STOC'23] [Jin, Xu STOC'23]

Conclusion

Graph reachability and distance problems:

single-source: mostly in near-linear time

all-pairs: mostly equivalent (up to logfactors) to an appropriate matrix product

Many, many more directions:

centrality measures: diameter, radius, eccentricities, girth, …

additive approximation, small weights, …

dynamic graphs, failing edges (replacement paths), spanners, …

… a huge, active research area

Thank you!