Lower Bounds for Conjunctive Query Evaluation

Stefan Mengel

CNRS, CRIL

25/06/2025

Motivation

Linear Time
- Boolean Queries
- Counting
- Direct Access

Beyond Linear Time
- Clique Problems
- Clique Embeddings

Motivation

Conjunctive Query Evaluation

Definition

Input: CQ q, database D
Output: $q(D)$

Question

How hard is conjunctive query evaluation? How hard are related questions?

Conjunctive Query Evaluation

Definition

Input: CQ q, database D
Output: $q(D)$

Question

How hard is conjunctive query evaluation? How hard are related questions?

Two (or three?) traditional ways of answering question:

- combined complexity
- data complexity
- parameterized complexity

discuss these and issues with them to motivate fine-grained complexity

Combined Complexity

- query and database are input

Theorem (Chandra, Merlin 1977, without proof!)

Boolean conjunctive query evaluation is NP-complete.

- problem: hardness even on constant size database
- know that many queries are easy, would like to understand them

Data Complexity

- only database input, query is fixed

Theorem

For every conjunctive query, evaluation is in AC^0 and thus in PTIME.

- very coarse
- $\|D\|^{|q|}$ runtime upper bound, but could be far easier
- does not differentiate hard and easy queries

Parameterized Complexity

- input size = size of database, query size parameter
- idea: determine influence of query size on complexity
- good complexity understanding for classes of queries
- still does not help for individual queries

Fine-Grained Complexity

- generally, tries to determine exact exponent for optimal runtime bounds
- hope for query evaluation: determine tight runtime bounds for individual queries
- particular use case: characterization of linear time queries

will survey some of this here, mostly simple arguments, but fine-grained complexity often far more complicated and technical

Which Complexity Assumptions?

"classical" approaches use different hardness assumptions

- combined complexity: SAT not in polynomial time
- parameterized complexity: ETH $\approx$ SAT takes time $2^{\Omega(n)}$

fine-grained complexity has many assumptions

- triangle finding, 3SUM, SETH, clique problems, matrix multiplication, ...

related in complicated ways, not surveyed here, see also Virginia's talk

General Assumptions

Query Restrictions

- consider only CQs
- self-join free!

Machine Model

RAM: random access, log-size registers, unit-cost, ...
(see e.g. [Grandjean, Jachiet 22])

Convention

n: size of active domain/number of vertices
m: size of database/number of tuples/number of edges

Linear Time

Linear Time:
Boolean Queries

Boolean Queries

Theorem (Yannakakis 81)

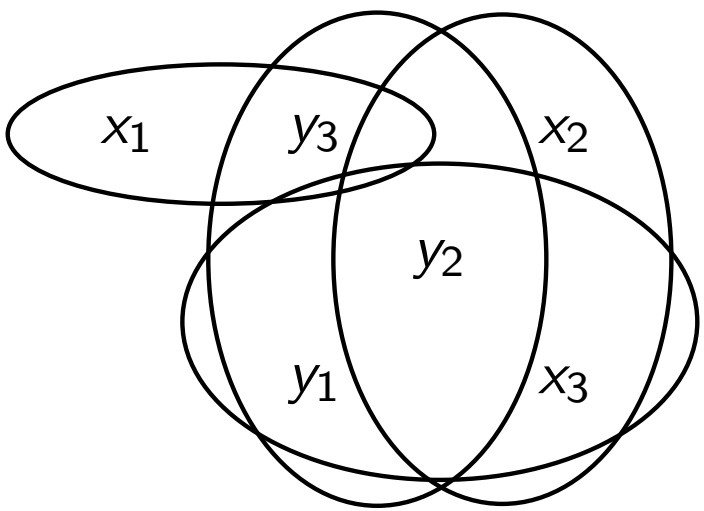
For every acyclic Boolean CQ, there is linear-time algorithm for query answering.

Reminder: Queries, Hypergraphs, Acyclicity

$$q(x_1, x_2, x_3) := \exists y_1 \exists y_2 \exists y_3 \, R_1(x_1, y_3) \wedge R_2(y_2, y_2, y_3) \\ \wedge R_3(y_1, y_2, x_3) \wedge R_4(x_2, x_3, y_2)$$

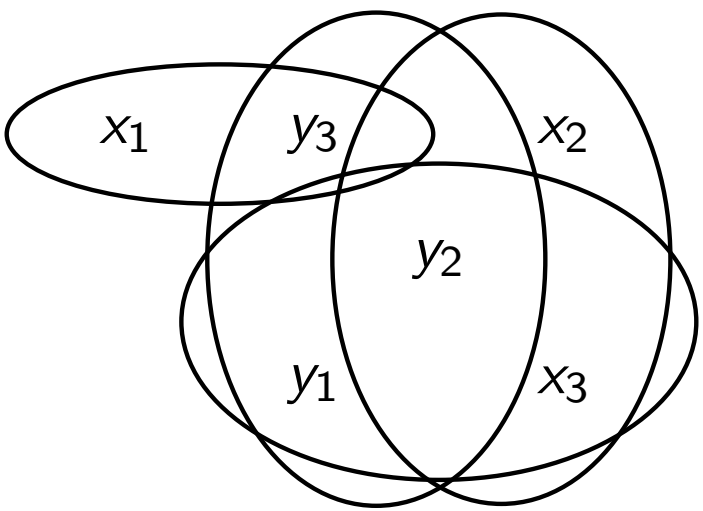
Reminder: Queries, Hypergraphs, Acyclicity

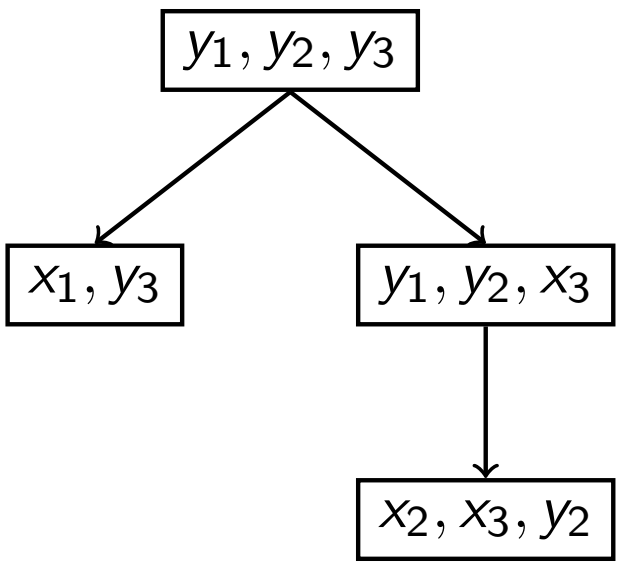
$$q(x_1, x_2, x_3) := \exists y_1 \exists y_2 \exists y_3 \, R_1(x_1, y_3) \wedge R_2(y_2, y_2, y_3) \wedge R_3(y_1, y_2, x_3) \wedge R_4(x_2, x_3, y_2)$$



Reminder: Queries, Hypergraphs, Acyclicity

$$q(x_1, x_2, x_3) := \exists y_1 \exists y_2 \exists y_3 \, R_1(x_1, y_3) \wedge R_2(y_2, y_2, y_3) \wedge R_3(y_1, y_2, x_3) \wedge R_4(x_2, x_3, y_2)$$





A Converse

Theorem (Brault-Baron 13)

No cyclic Boolean CQ has linear-time algorithm for query answering assuming some complexity hypotheses.

- how to prove this type of result?
- what are the hypotheses? and how credible are they?

The Graph Case

- assume first: all atoms arity 2
- acyclicity is graph acyclicity, so not having any cycles
- then cycle queries should be hard
- and even triangle query should be hard

The Graph Case

- assume first: all atoms arity 2
- acyclicity is graph acyclicity, so not having any cycles
- then cycle queries should be hard
- and even triangle query should be hard

Question

How hard is detecting triangles in graphs?

The Complexity of Detecting Triangles (I)

Theorem (Nešetřil, Poljak 1985?)

There is an algorithm that in time $\tilde{O}(n^\omega)$ decides if given graph G has a triangle.

- ω: matrix multiplication exponent, $2 \leq \omega < 2.371552$ [Vassilevska Williams, Xu, Xu, and Zhou 2024]

The Complexity of Detecting Triangles (I)

Theorem (Nešetřil, Poljak 1985?)

There is an algorithm that in time $\tilde{O}(n^\omega)$ decides if given graph G has a triangle.

- ω: matrix multiplication exponent, $2 \leq \omega < 2.371552$ [Vassilevska Williams, Xu, Xu, and Zhou 2024]

Proof (sketch).

- compute square A^2 of adjancency matrix
- non-zero entries correspond to pairs connected by 2-path
- intersect those pairs with edges

□

The Complexity of Detecting Triangles (II)

Theorem (Alon, Yuster, Zwick 1997)

There is an algorithm that in time $\tilde{O}(m^{\frac{2\omega}{\omega+1}})$ decides if given graph G has a triangle.

if $\omega = 2$, then $\tilde{O}(m^{\frac{4}{3}})$

The Complexity of Detecting Triangles (II)

Theorem (Alon, Yuster, Zwick 1997)

There is an algorithm that in time $\tilde{O}(m^{\frac{2\omega}{\omega+1}})$ decides if given graph G has a triangle.

if $\omega = 2$, then $\tilde{O}(m^{\frac{4}{3}})$

Proof (idea).

- split vertices by degree Δ
- triangle with a low degree vertex easy to find in $\tilde{O}(m\Delta)$
- only $2m/\Delta$ heavy vertices; use algorithm from before for triangle with only heavy vertices $\tilde{O}((\frac{m}{\Delta})^{\omega})$
- choose Δ optimally as $m^{\frac{\omega-1}{\omega+1}}$

The Triangle Hypothesis

Hypothesis (Triangle Hypothesis)

No algorithm that in time $O(m)$ decides if given graph has triangle.

The Triangle Hypothesis

Hypothesis (Triangle Hypothesis)

No algorithm that in time $O(m)$ decides if given graph has triangle.

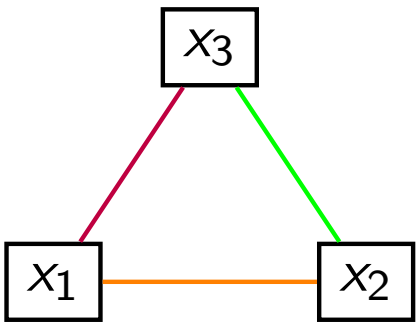
Lemma

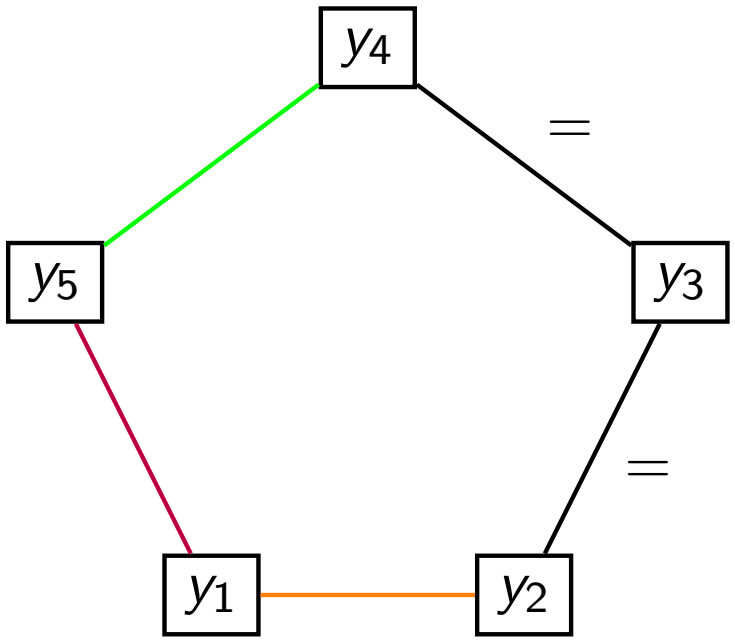
Assuming the Triangle Hypothesis, no cycle query

$$q_k^C := \exists x_1 \dots \exists x_k E_1(x_1, x_2) \wedge \dots \wedge E_{k-1}(x_{k-1}, x_k) \wedge E_k(x_k, x_1)$$

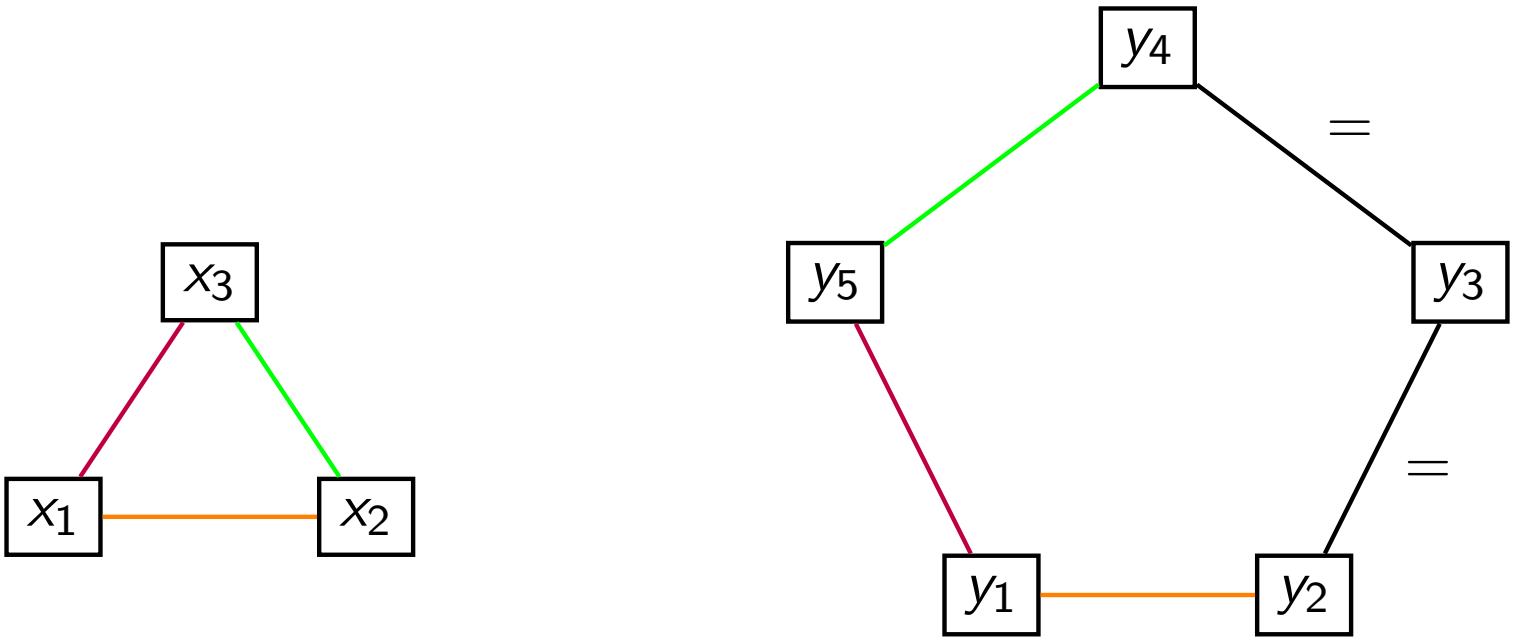
has a linear time algorithm

Proof by Picture and Corollary





Proof by Picture and Corollary



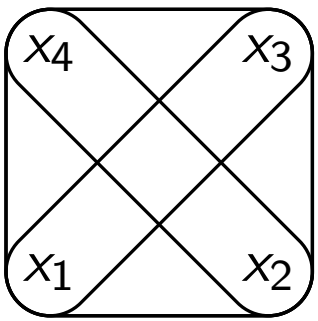
Corollary

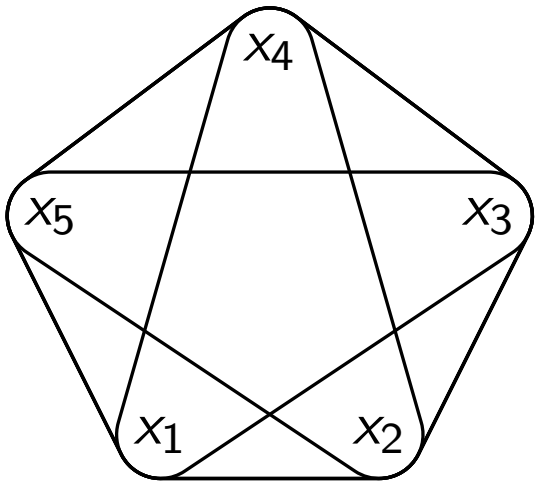
Assuming the Triangle Hypothesis, no cyclic graphlike Boolean CQ has linear-time algorithm.

Hypergraphs: Loomis-Whitney Joins

- hypergraph acyclicity more complicated

$$LW_k := \exists x_1 \dots \exists x_k \bigwedge_{X \subseteq \{x_1,\dots,x_k\} : |X| = k-1} R_X(X)$$

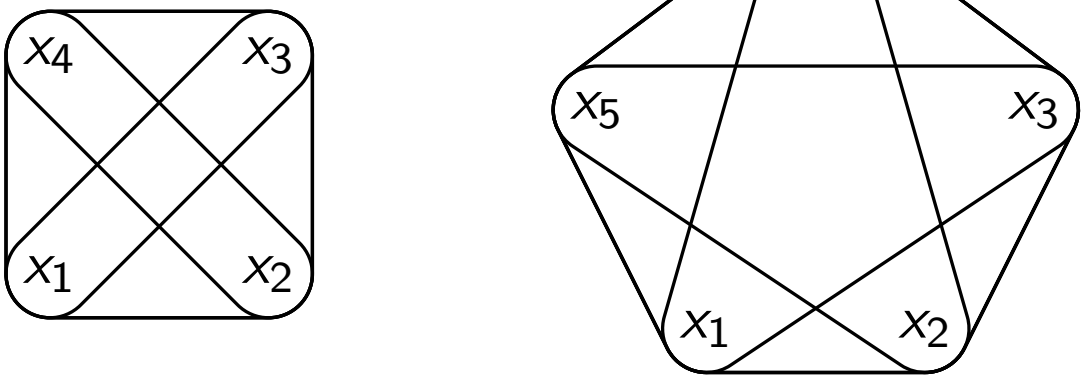




Hypergraphs: Loomis-Whitney Joins

- hypergraph acyclicity more complicated

$$LW_k := \exists x_1 \dots \exists x_k \bigwedge_{X \subseteq \{x_1,\dots,x_k\} : |X| = k-1} R_X(X)$$



- not clear how to embed triangle in useful way (every set of size 3 covered)
- but also no known linear-time algorithm

The Hyperclique Hypothesis

- h-uniform hypergraph: all edges have size h
- k-hyperclique in h-uniform hypergraph: vertex set C of size k such that every $S \subseteq C$ of size h is edge

The Hyperclique Hypothesis

- h-uniform hypergraph: all edges have size h
- k-hyperclique in h-uniform hypergraph: vertex set C of size k such that every $S \subseteq C$ of size h is edge

Hypothesis (Hyperclique Hypothesis)

For no $k > h > 2$ there is $\varepsilon > 0$ such that k-hyperclique in h-uniform hypergraphs can be decided in time $\tilde{O}(n^{k-\varepsilon})$.

- breaking hypothesis would give surprising algorithms for Max-k-SAT, so rather believable
- cliques in graphs are exception (and indeed have better algorithms)
- graph clique algorithms do not generalize [Lincoln, Vassilevska Williams, Williams 2018]

Complexity of Loomis-Whitney Joins – I

Theorem

Assuming Hyperclique Hypothesis, there is no $k > 3$ and $\varepsilon > 0$ with algorithm for LW_k algorithm with runtime $\tilde{O}(m^{1+\frac{1}{k-1}-\varepsilon})$.

- $\tilde{O}(m^{1+\frac{1}{k-1}})$ algorithms exist (worst-case optimal join)
- rules out linear time algorithm for Loomis-Whitney joins

Complexity of Loomis-Whitney Joins – II

Theorem

Assuming Hyperclique Hypothesis, there is no $k > 3$ and $\varepsilon > 0$ with algorithm for LW_k algorithm with runtime $\tilde{O}(m^{1+\frac{1}{k-1}-\varepsilon})$.

Proof (sketch).

- use LW_k to solve k-clique in $(k-1)$-uniform hypergraph H
- database D: all relations contain for every $e \in E(H)$ all permutations; size n^{k-1}
- LW_k true on D iff H has k-clique
- assume $\tilde{O}(m^{1+\frac{1}{k-1}-\varepsilon})$ algorithm for LW_k, then runtime on D

$$\tilde{O}((n^{k-1})^{1+\frac{1}{k-1}-\varepsilon}) = \tilde{O}(n^{k-(k-1)\varepsilon})$$

which breaks the Hyperclique Hypothesis

□

Back to Brault-Baron

Lemma (Brault-Baron 2013)

Every cyclic CQ contains as a subquery

- *a cycle query, or*
- *a Loomis-Whitney query*

- several related earlier results [Bagan 2009], [Beeri, Fagin, Maier, Yannakakis 1983] but not quite the same

Back to Brault-Baron

Lemma (Brault-Baron 2013)

Every cyclic CQ contains as a subquery

- *a cycle query, or*
- *a Loomis-Whitney query*

- several related earlier results [Bagan 2009], [Beeri, Fagin, Maier, Yannakakis 1983] but not quite the same

Theorem (Brault-Baron 2013)

Assuming Triangle and Hyperclique Hypotheses, no cyclic Boolean CQ has linear-time algorithm.

Linear Time:
Counting

Counting Number of Answers

- for acyclic join queries Yannakakis-variant in linear time (so pretty uninteresting here)

Counting Number of Answers

- for acyclic join queries Yannakakis-variant in linear time (so pretty uninteresting here)

Theorem (Pichler, Skritek 2013)

Counting answers to acyclic CQs with projection #P-hard (in combined complexity)

- so something interesting happens with projection
- series of papers for understanding projection in combined and parameterized complexity (see survey for references)

Counting Number of Answers

- for acyclic join queries Yannakakis-variant in linear time (so pretty uninteresting here)

Theorem (Pichler, Skritek 2013)

Counting answers to acyclic CQs with projection #P-hard (in combined complexity)

- so something interesting happens with projection
- series of papers for understanding projection in combined and parameterized complexity (see survey for references)

Question

What is impact of projection in fine-grained complexity?

Detour: SETH and Dominating Set

Hypothesis (Strong Exponential Time Hypothesis (SETH))

For every $\varepsilon > 0$ there is k such that k-SAT cannot be solved in time $\tilde{O}(2^{n(1-\varepsilon)})$

Detour: SETH and Dominating Set

Hypothesis (Strong Exponential Time Hypothesis (SETH))

For every $\varepsilon > 0$ there is k such that k-SAT cannot be solved in time $\tilde{O}(2^{n(1-\varepsilon)})$

Definition (k-DS)

Input:	graph G
Question:	is there dominating set of size k in G?

(S dominating set if every vertex in G is in S or neighbor of S)

Detour: SETH and Dominating Set

Hypothesis (Strong Exponential Time Hypothesis (SETH))

For every $\varepsilon > 0$ there is k such that k-SAT cannot be solved in time $\tilde{O}(2^{n(1-\varepsilon)})$

Definition (k-DS)

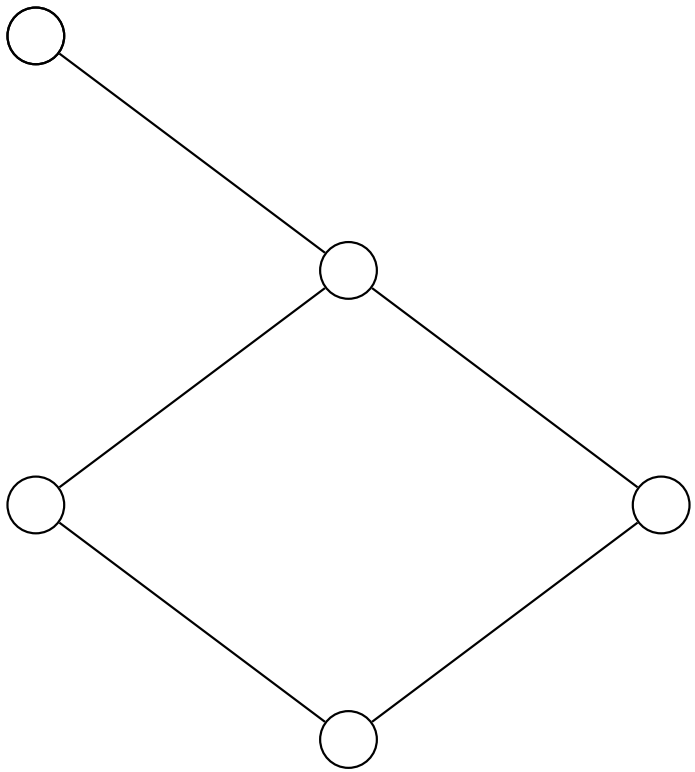
Input:	graph G
Question:	is there dominating set of size k in G?

(S dominating set if every vertex in G is in S or neighbor of S)

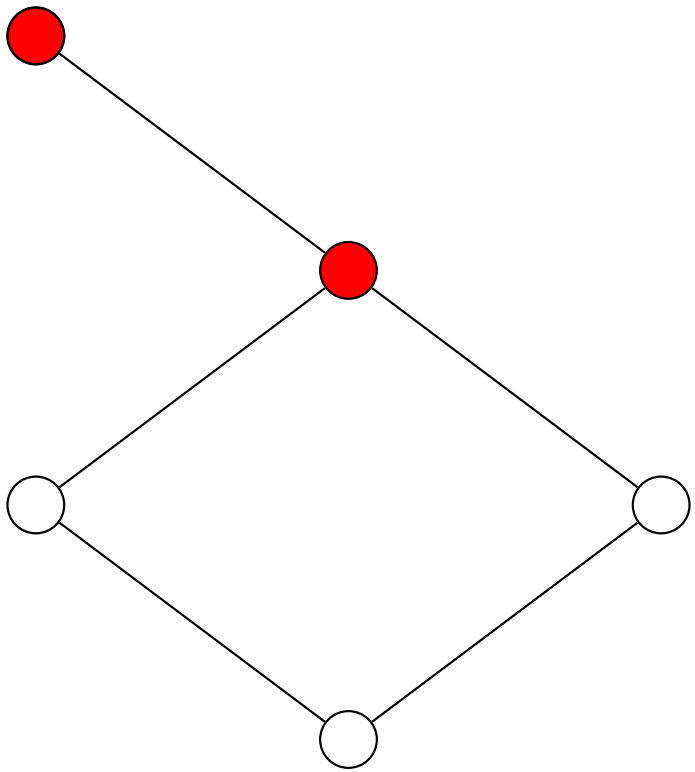
Theorem (Pătrașcu, Williams 2010)

Assuming SETH, for no $k \geq 3$ and no $\varepsilon > 0$, there is algorithm for k-DS with runtime $\tilde{O}(n^{k-\varepsilon})$.

Example: Dominating Sets

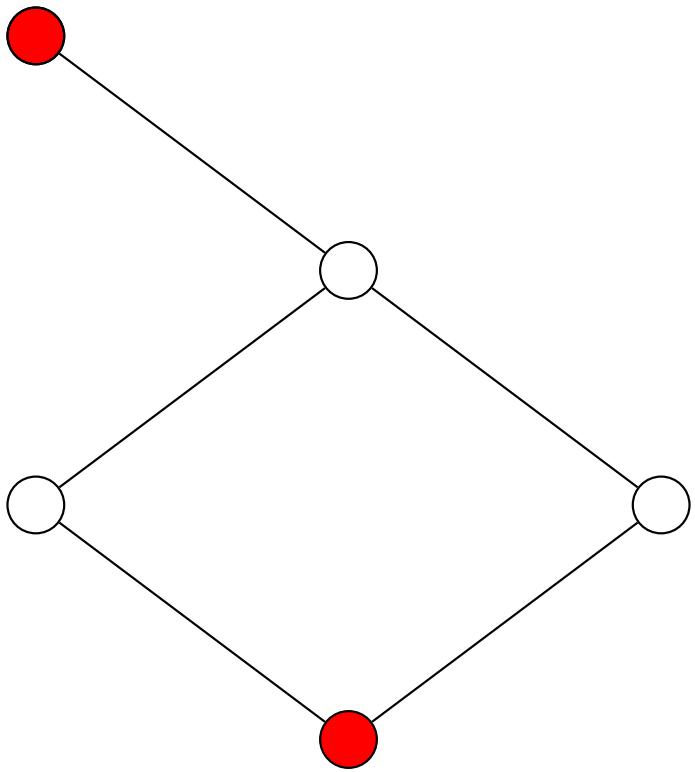


Example: Dominating Sets



no dominating set

Example: Dominating Sets



dominating set

Detour: SETH and Dominating Set

Hypothesis (Strong Exponential Time Hypothesis (SETH))

For every $\varepsilon > 0$ there is k such that k-SAT cannot be solved in time $\tilde{O}(2^{n(1-\varepsilon)})$

Definition (k-DS)

Input:	graph G
Question:	is there dominating set of size k in G?

(S dominating set if every vertex in G is in S or neighbor of S)

Detour: SETH and Dominating Set

Hypothesis (Strong Exponential Time Hypothesis (SETH))

For every $\varepsilon > 0$ there is k such that k-SAT cannot be solved in time $\tilde{O}(2^{n(1-\varepsilon)})$

Definition (k-DS)

Input:	graph G
Question:	is there dominating set of size k in G?

(S dominating set if every vertex in G is in S or neighbor of S)

Theorem (Pătrașcu, Williams 2010)

Assuming SETH, for no $k \geq 3$ and no $\varepsilon > 0$, there is algorithm for k-DS with runtime $\tilde{O}(n^{k-\varepsilon})$.

2-Stars and a Reduction From Dominating Set

Theorem

Assuming SETH, no linear time counting for

$$q_2^\star(x_1, x_2) := \exists z\, E_1(x_1, z) \wedge E_2(x_2, z).$$

2-Stars and a Reduction From Dominating Set

Theorem

Assuming SETH, no linear time counting for

$$q_2^\star(x_1, x_2) := \exists z\, E_1(x_1, z) \wedge E_2(x_2, z).$$

Proof.

- reduce from 4-DS, so let $G = (V, E)$ be input
- database has relations of size $O(n^3)$

$$E_i^D := \{((v_1, v_2), u) \mid v_1 u \notin E, v_2 u \notin E, v_1 \neq u, v_2 \neq u\}$$

- choice of x_1, x_2 in $q_2^\star$ is choice of ≤ 4 vertices in G
- x_1, x_2 not a dominating set iff x_1, x_2 model of $q_2^\star$, so number of 4-DS in G is $n^4 - |q_2^\star(D)|$
- so 4-DS solved with one call to $q_2^\star$; if linear time, then $\tilde{O}(n^3)$ algorithm for 4-DS

2-Stars and a Reduction From Dominating Set

Theorem

Assuming SETH, no linear time counting for

$$q_2^\star(x_1, x_2) := \exists z\, E_1(x_1, z) \wedge E_2(x_2, z).$$

- can be improved to excluding $\tilde{O}(m^{2-\varepsilon})$ by increasing k
- can be lifted to bigger stars
- idea to reduce from k-DS from [Dell, Roth, Wellnitz 2019], but adapted for linear time case

Generalization: Bad Path

Definition (Bad Path)

Bad path in query $q(X)$: path $v_1, \ldots, v_k$ in hypergraph of q with

- v_1, v_k output variables,
- other variables no output variables
- v_1, v_k in no common edge

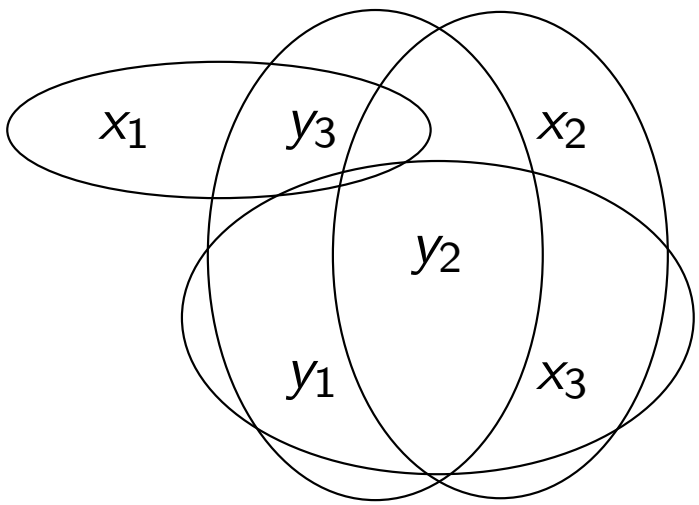
Example

- $$q_2^\star(x_1, x_2) := \exists z\, E_1(x_1, z) \wedge E_2(x_2, z)$$

  has bad path x_1, z, x_2

Examples

$$q(x_1, x_2, x_3) := \exists y_1 \exists y_2 \exists y_3 \, R_1(x_1, y_3) \wedge R_2(y_2, y_2, y_3) \wedge R_3(y_1, y_2, x_3) \wedge R_4(x_2, x_3, y_2)$$



bad path x_1, y_3, y_2, x_2

Hardness by Bad Paths

Definition (Bad Path)

Bad path in query $q(X)$: path $v_1, \ldots, v_k$ in hypergraph of q with

- v_1, v_k output variables,
- other variables no output variables
- v_1, v_k in no common edge

Theorem

Assuming SETH, if $q(X)$ contains a bad path, then answers of q cannot be counted in linear time.

Proof (idea).

embed $q_2^\star$ into bad path □

Free-Connex Acyclic Queries

Definition

Query $q(X)$ called free-connex acyclic if

- it is acyclic, and
- it has no bad path

- originally from context of enumeration [Bagan, Durand, Grandjean 2007]
- several equivalent definitions
- can be generalized into width measure ("star size") [Durand, M 2014]

Counting for Free-Connex Acyclic Queries

Theorem (Brault-Baron 2013)

For every free-connex acyclic query $q(X)$, answers can be counted in linear time.

Theorem

Assume the Triangle and Hyperclique Hypotheses and SETH. Then answer counting for $q(X)$ can be done in linear time iff $q(X)$ is free-connex acyclic.

Counting for Free-Connex Acyclic Queries

Theorem (Brault-Baron 2013)

For every free-connex acyclic query $q(X)$, answers can be counted in linear time.

Theorem

Assume the Triangle and Hyperclique Hypotheses and SETH. Then answer counting for $q(X)$ can be done in linear time iff $q(X)$ is free-connex acyclic.

Proof.

- algorithm above
- if not acyclic, then hardness from decision
- if not free-connex, then counting hardness through bad path

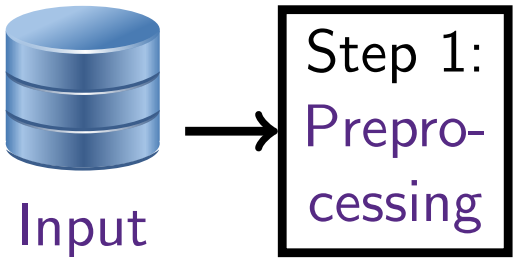
□

Linear Time:
Direct Access

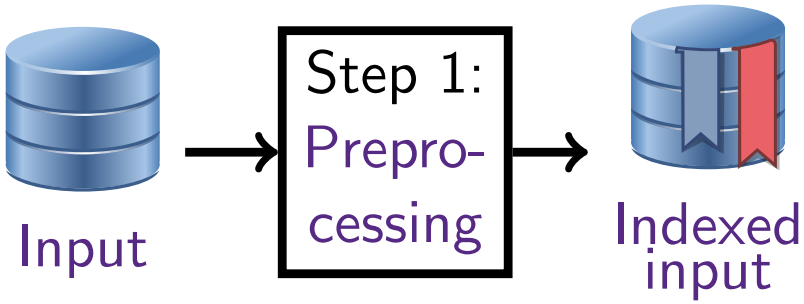
Direct Access Algorithm

Input

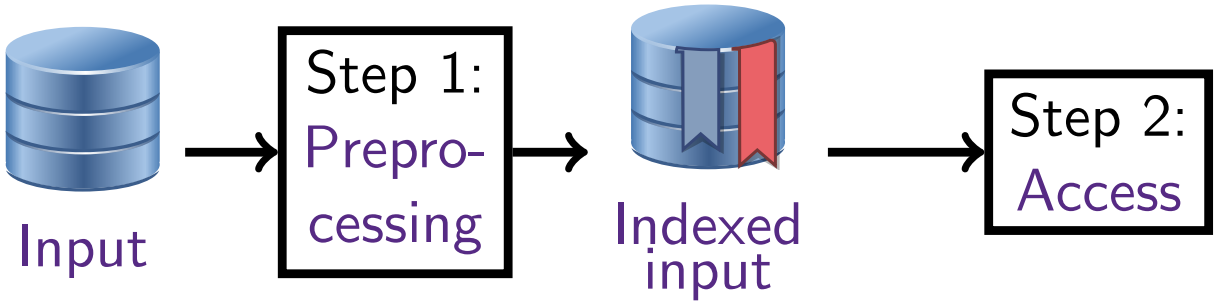
Direct Access Algorithm



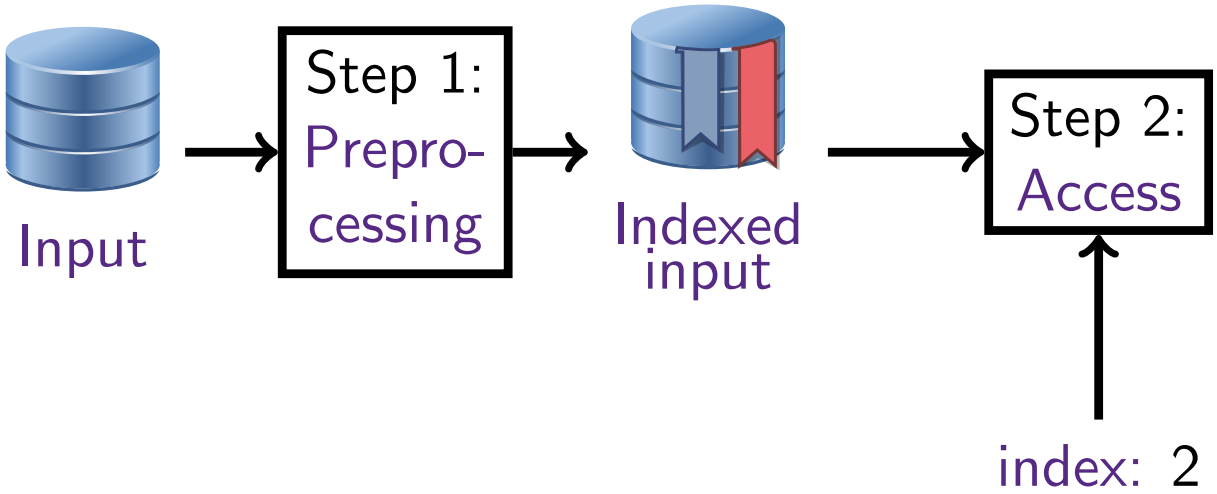
Direct Access Algorithm



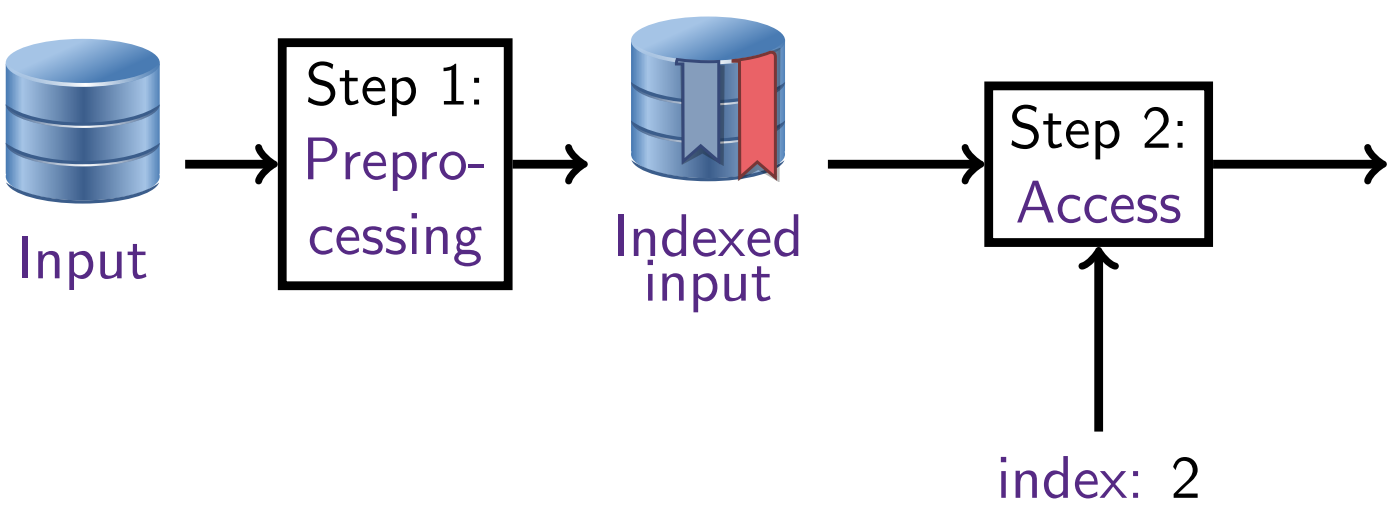
Direct Access Algorithm



Direct Access Algorithm



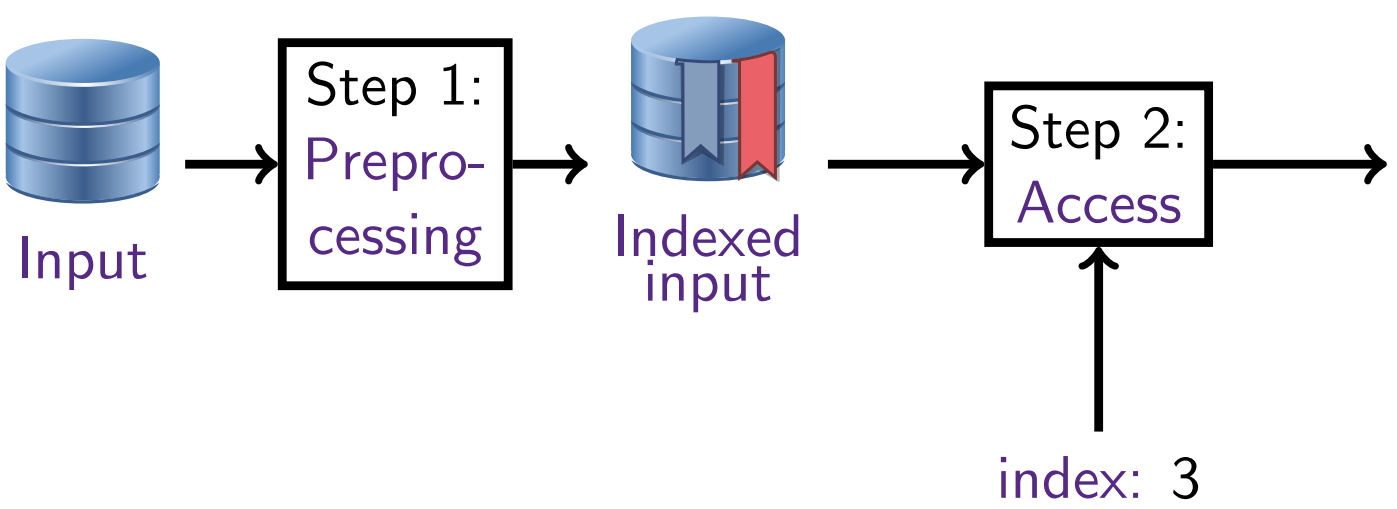
Direct Access Algorithm



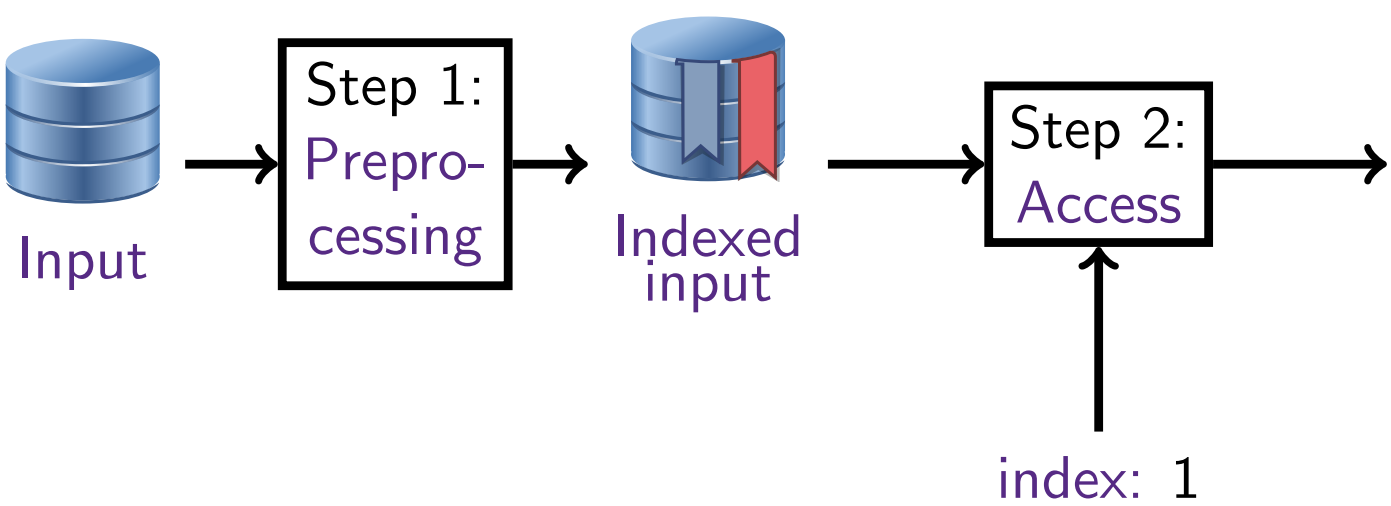
index	A	B	C
2	a'	b	c

Results

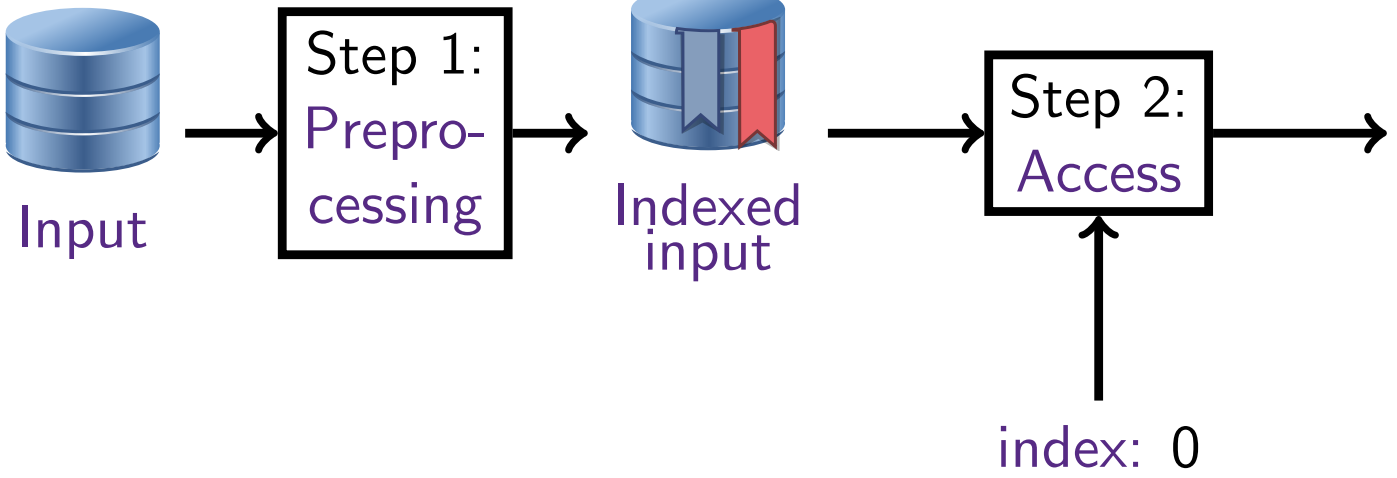
Direct Access Algorithm



Direct Access Algorithm



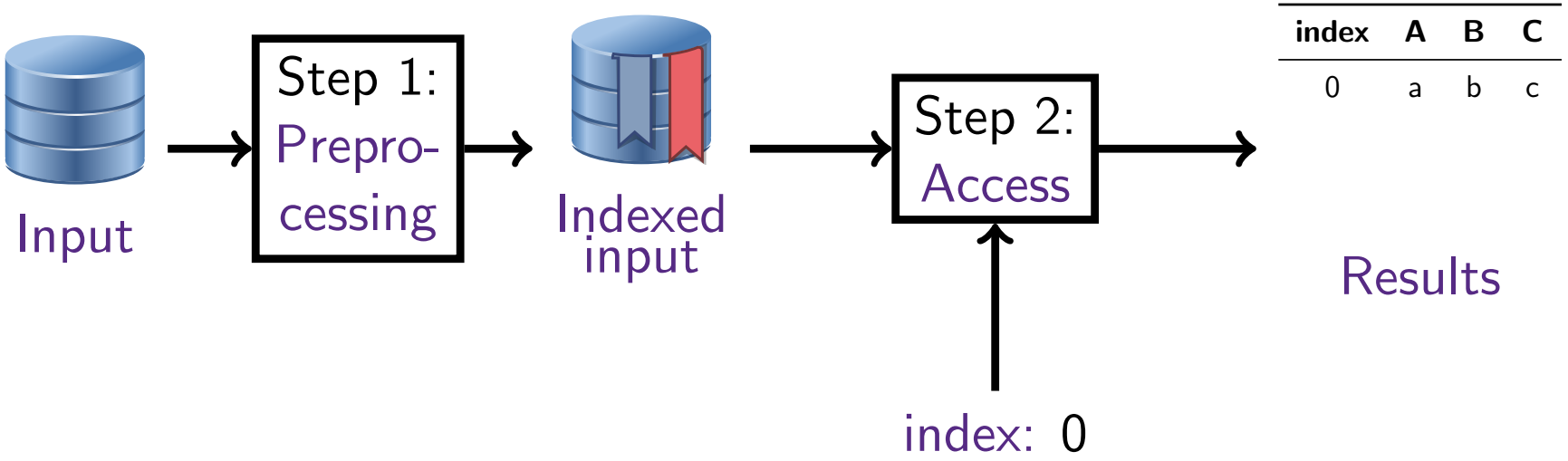
Direct Access Algorithm



index	A	B	C
0	a	b	c

Results

Direct Access Algorithm

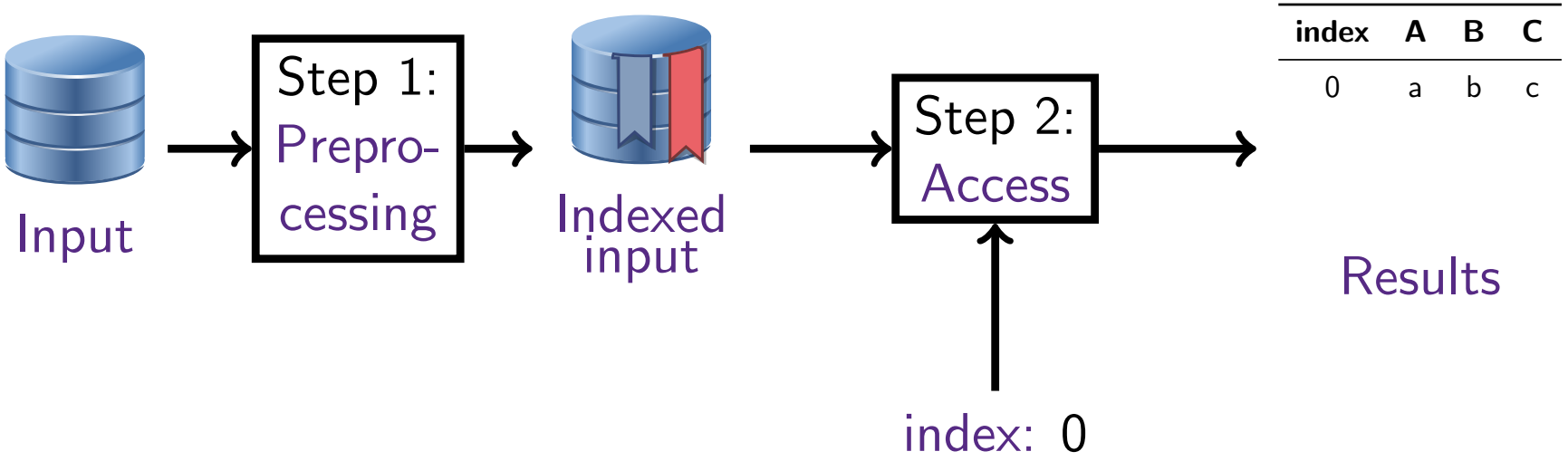


- model introduced by [Bagan, Durand, Grandjean, Olive 2008] in context of enumeration
- want polylogarithmic access time

Question

Which preprocessing time necessary for which query?

Direct Access Algorithm



- model introduced by [Bagan, Durand, Grandjean, Olive 2008] in context of enumeration
- want polylogarithmic access time

Question

Which preprocessing time necessary for which query?

Local Restriction

Only join queries in this part!

Known Result

Theorem (Brault-Baron 2013)

For acyclic queries there is direct access algorithm with

- *linear preprocessing,*
- *logarithmic query time*

Question

Getting anwer at index j, but in which order?

Known Result

Theorem (Brault-Baron 2013)

For acyclic queries there is direct access algorithm with

- *linear preprocessing,*
- *logarithmic query time*

Question

Getting anwer at index j, but in which order?

Answer

lexicographical order, attribute order depends on shape of query

Example: Lexicographic Orders

$$q(x_1, x_2, x_3) = R_1(x_1, x_2) \wedge R_2(x_2, x_3)$$

R_1:

x_1	x_2
a	a
a	c
b	b

R_2:

x_2	x_3
a	c
b	b
c	a

variable order: $x_1 \succ x_2 \succ x_3$

Example: Lexicographic Orders

$$q(x_1, x_2, x_3) = R_1(x_1, x_2) \wedge R_2(x_2, x_3)$$

R_1:

x_1	x_2
a	a
a	c
b	b

R_2:

x_2	x_3
a	c
b	b
c	a

variable order: $x_1 \succ x_2 \succ x_3$

x_1	x_2	x_3
a	a	c
a	c	a
b	b	b

Example: Lexicographic Orders

$$q(x_1, x_2, x_3) = R_1(x_1, x_2) \wedge R_2(x_2, x_3)$$

R_1:

x_1	x_2
a	a
a	c
b	b

R_2:

x_2	x_3
a	c
b	b
c	a

variable order: $x_2 \succ x_3 \succ x_1$

x_2	x_3	x_1
a	c	a
b	b	b
c	a	a

Example: Lexicographic Orders

$$q(x_1, x_2, x_3) = R_1(x_1, x_2) \wedge R_2(x_2, x_3)$$

R_1:

x_1	x_2
a	a
a	c
b	b

R_2:

x_2	x_3
a	c
b	b
c	a

variable order: $x_3 \succ x_1 \succ x_2$

x_3	x_1	x_2
a	a	c
b	b	b
c	a	a

Prescribing the Order

Question

Getting anwer at index j, but in which order?

Answer

lexicographical order, attribute order depends on shape of query

Prescribing the Order

Question

Getting anwer at index j, but in which order?

Answer

lexicographical order, attribute order depends on shape of query

Question

Can we choose attribute order without degrading runtime?

Linear Preprocessing

Theorem (essentially Carmeli, Tziavelis, Gatterbauer, Kimelfeld, Riedewald 2021)

Let q be acyclic, self-join free query.
Assuming the Triangle Hypothesis, direct access with variable order π and linear preprocessing possible if and only if q and π have no disruptive trio .

Linear Preprocessing

Theorem (essentially Carmeli, Tziavelis, Gatterbauer, Kimelfeld, Riedewald 2021)

Let q be acyclic, self-join free query.
Assuming the Triangle Hypothesis, direct access with variable order π and linear preprocessing possible if and only if q and π have no disruptive trio .

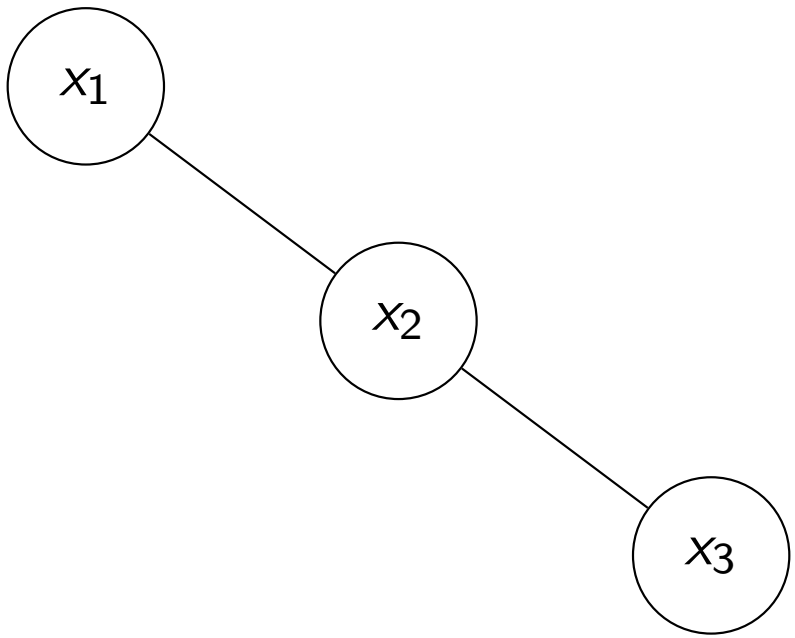
disruptive trio:

- variables x, y, z with $x \succ z$, $y \succ z$
- there is atom with variables x, z
- there is atom with variables y, z
- there is no atom with variables x, y

Example: Disruptive Trios

$$q(x_1, x_2, x_3) = R_1(x_1, x_2) \wedge R_2(x_2, x_3)$$

order: $x_1 \succ x_2 \succ x_3$

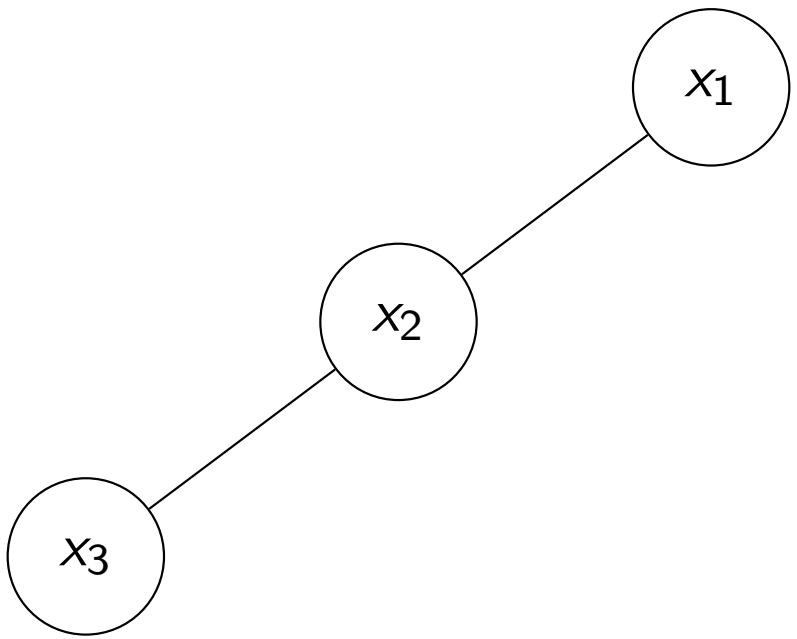


no disruptive trio

Example: Disruptive Trios

$$q(x_1, x_2, x_3) = R_1(x_1, x_2) \wedge R_2(x_2, x_3)$$

order: $x_3 \succ x_2 \succ x_1$

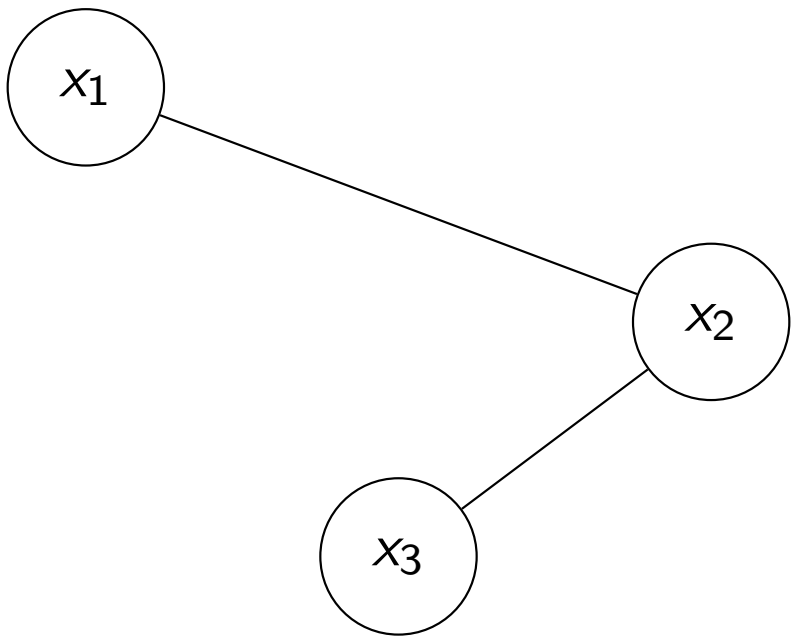


no disruptive trio

Example: Disruptive Trios

$$q(x_1, x_2, x_3) = R_1(x_1, x_2) \wedge R_2(x_2, x_3)$$

order: $x_1 \succ x_3 \succ x_2$



disruptive trio

Why Disruptive Trios Make Things Hard

- allow simulating direct access for

$$q(x_1, x_2, z) = R_1(x_1, z) \wedge R_2(x_2, z)$$

with order $x_1 \succ x_2 \succ z$

Why Disruptive Trios Make Things Hard

- allow simulating direct access for

$$q(x_1, x_2, z) = R_1(x_1, z) \wedge R_2(x_2, z)$$

  with order $x_1 \succ x_2 \succ z$
- binary search allows enumeration and testing for

$$q(x_1, x_2) = \exists z R_1(x_1, z) \wedge R_2(x_2, z)$$

  which is hard under Triangle Hypothesis

From Triangles to Disruptive Trios

Lemma

Assuming the Triangle Hypothesis, testing for

$$q(x_1, x_2) = \exists z R_1(x_1, z) \wedge R_2(x_2, z)$$

not with linear preprocessing and polylogarithmic testing time.

From Triangles to Disruptive Trios

Lemma

Assuming the Triangle Hypothesis, testing for

$$q(x_1, x_2) = \exists z R_1(x_1, z) \wedge R_2(x_2, z)$$

not with linear preprocessing and polylogarithmic testing time.

Proof.

Algorithm for triangle finding in $G = (V, E)$:

- set $R_1 = R_2 = E$ and preprocess
- for every edge $uv \in E$, check if $(u, v) \in q(D)$; if so, found a triangle
- overall runtime:

$$t_{preproc} + |E| t_{test}$$

General Queries

Theorem (essentially Carmeli, Tziavelis, Gatterbauer, Kimelfeld, Riedewald 2021)

Let q be acyclic, self-join free query.
Assuming the Triangle Hypothesis, direct access with variable order π and linear preprocessing possible if and only if q and π have no disruptive trio .

General Queries

Theorem (essentially Carmeli, Tziavelis, Gatterbauer, Kimelfeld, Riedewald 2021)

Let q be acyclic, self-join free query.
Assuming the Triangle Hypothesis, direct access with variable order π and linear preprocessing possible if and only if q and π have no disruptive trio .

- proof: simple embedding of query from last slide
- generalizations [Bringmann, Carmeli, M 2025]
  - getting rid of self-join assumption
  - getting rid of acyclicity
  - determine optimal runtime for all join queries and variable orders

Beyond Linear Time

Problems When Going Beyond Linear Time

General approach we have seen

1. use right hardness assumption to show lower bound for one query
2. embed query in all other hard ones

Problems When Going Beyond Linear Time

General approach we have seen

1. use right hardness assumption to show lower bound for one query
2. embed query in all other hard ones

Problem

Missing both ingredients to generally go beyond linear time:

- right hardness assumptions unclear
- no general embedding results

Sketch approaches to both problems

Beyond Linear Time:
Clique Problems

Complexity of Clique

- Clique is starting point for reductions in many areas
- would be very useful if k-Clique required time $\Omega(n^k)$ (false!!!)

Complexity of Clique

- Clique is starting point for reductions in many areas
- would be very useful if k-Clique required time $\Omega(n^k)$ (false!!!)

Theorem (Nešetřil, Poljak 1985)

Let k be divisible by 3. Then k-Clique on graphs with n vertices can be solved in time $\tilde{O}(n^{\omega k/3})$

Complexity of Clique

- Clique is starting point for reductions in many areas
- would be very useful if k-Clique required time $\Omega(n^k)$ (false!!!)

Theorem (Nešetřil, Poljak 1985)

Let k be divisible by 3. Then k-Clique on graphs with n vertices can be solved in time $\tilde{O}(n^{\omega k/3})$

Proof (sketch).

given graph $G = (V, E)$ construct new graph $G' = (V', E')$ with

- V' contains all cliques of size $k/3$ in G as vertices
- $uv \in E' \Leftrightarrow u \cup v$ induces $2k/3$-clique in G

constructed in time $n^{2k/3}$
triangles in G' are k-cliques in G, so use fast triangle algorithm based on matrix multiplication □

Combinatorial k-Clique Algorithms

Hypothesis (Combinatorial k-Clique Hypothesis)

There is no combinatorial algorithm for k-Clique with runtime $\tilde{O}(n^{k-\varepsilon})$.

Combinatorial Algorithms?

"I know it when I see it"

Combinatorial Algorithms?

No formal definition of “combinatorial algorithm”, take criteria from [Abboud, Fischer, Shechter 2024]:

Combinatorial Algorithms?

No formal definition of "combinatorial algorithm", take criteria from [Abboud, Fischer, Shechter 2024]:

- practical efficiency: no large hidden factors
- elegance: combinatorially interpretable intermediate results
- generalizable: e.g. weighted instances, hypergraphs, generation of solutions

Combinatorial Algorithms?

No formal definition of "combinatorial algorithm", take criteria from [Abboud, Fischer, Shechter 2024]:

- practical efficiency: no large hidden factors
- elegance: combinatorially interpretable intermediate results
- generalizable: e.g. weighted instances, hypergraphs, generation of solutions

arguably some algorithms in database theory fail some of these criteria (e.g. PANDA, Courcelle-style algorithms, some counting algorithms, ...)

Combinatorial k-Clique Algorithms

Hypothesis (Combinatorial k-Clique Hypothesis)

There is no combinatorial algorithm for k-Clique with runtime $\tilde{O}(n^{k-\varepsilon})$.

- not formally defined notion
- useful: allows excluding fast combinatorial algorithms for many other problems
- but what do we actually show?

Combinatorial k-Clique Algorithms

Hypothesis (Combinatorial k-Clique Hypothesis)

There is no combinatorial algorithm for k-Clique with runtime $\tilde{O}(n^{k-\varepsilon})$.

- not formally defined notion
- useful: allows excluding fast combinatorial algorithms for many other problems
- but what do we actually show?

Personal Opinion

- avoid hypothesis if possible
- be very clear about naming! not just "k-Clique Hypothesis"!
- discuss short-comings of inferred lower bounds

Weighted Clique

Definition (Min-Weight k-Clique Problem)

Input: graph G with edge weights $w : E \to \mathbb{Z}$
Output: minimal sum of edge-weights for k-clique in G

Hypothesis (Min-Weight k-Clique Hypothesis)

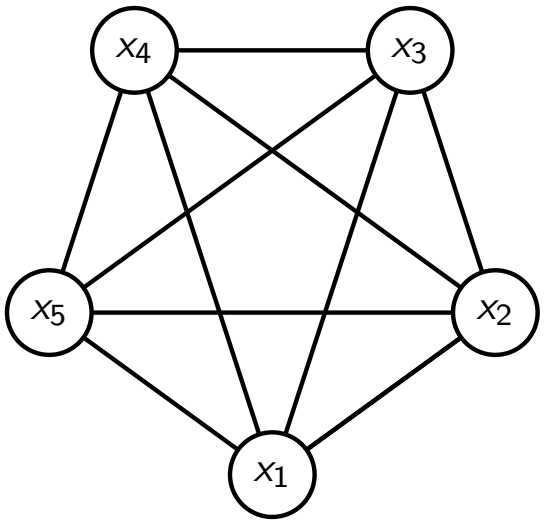
There is no k and $\varepsilon > 0$ such that min-width k-clique on can be solved in time $\tilde{O}(n^{k-\varepsilon})$

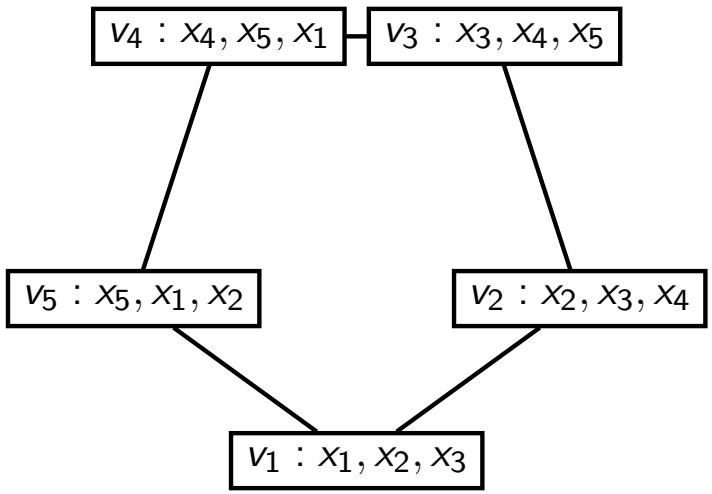
- increasingly used in fine-grained literature (often for triangles)
- matrix multiplication does not seem to help
- aggregation over $(\min, +)$-semiring

Beyond Linear Time:
Clique Embeddings

Clique Embeddings [Fan, Koutris, Zhao 2023]

- idea: solve clique problems by embedding into other queries





- properties:
  - every x_i gets mapped somewhere
  - bags containing x_i connected
  - all pairs x_i, x_j touch on cycle

Lower Bound by Clique Embedding

Lemma

Assuming Min-k-Clique Hypothesis, no algorithm with runtime $\tilde{O}(m^{\frac{5}{4}-\varepsilon})$ *for any* $\varepsilon > 0$ *for aggregation on* 5*-cycle query over* $(\min, +)$*-semiring.*

Lower Bound by Clique Embedding

Lemma

Assuming Min-k-Clique Hypothesis, no algorithm with runtime $\tilde{O}(m^{\frac{5}{4}-\varepsilon})$ *for any* $\varepsilon > 0$ *for aggregation on* 5*-cycle query over* $(\min, +)$*-semiring.*

Proof idea.

- input graph G with n vertices and edge weights
- choose database D such that query result is 5-cliques of G, size n^4
- aggregation over $(\min, +)$ gives minimal weight of clique in G
- runtime $\tilde{O}(m^{\frac{5}{4}-\varepsilon}) = \tilde{O}(n^{4\cdot(\frac{5}{4}-\varepsilon)}) = \tilde{O}(n^{5-4\varepsilon})$ breaks Min-Weight-k-Clique Hypothesis

□

More on Clique Embeddings

- can be developed into a general framework [Fan, Koutris, Zhao 2023]
- gives some tight bounds for aggregation and combinatorial algorithms
- also lower bounds for submodular width
- unfortunately, generally not tight 

Conclusion

Conclusion

- understand linear time queries pretty well for decision, counting but also enumeration, direct access
- self-joins add some subtlety
- a lot to do for superlinear case, only direct access for lexicographic orders well understood
- more results and lots of references in survey on arxiv
  `https://arxiv.org/abs/2506.17702`
  read also enumeration tutorial [Berkholz, Gerhardt, Schweikardt 2020]

Conclusion

- understand linear time queries pretty well for decision, counting but also enumeration, direct access
- self-joins add some subtlety
- a lot to do for superlinear case, only direct access for lexicographic orders well understood
- more results and lots of references in survey on arxiv
  `https://arxiv.org/abs/2506.17702`
  read also enumeration tutorial [Berkholz, Gerhardt, Schweikardt 2020]

Thank you for your attention!