Updated 2/25/2020

T3: Efficient query evaluation
L15: Cyclic query evaluation

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp20)

https://northeastern-datalab.github.io/cs7240/sp20/

Version 2/25/2020

Outline: Acyclic conjunctive queries

- Acyclic conjunctive queries
  - The semijoin operator
  - Join trees & Yannakakis algorithm
  - Query hypergraphs & GYO reduction
  - A detailed Yannakakis example
  - Full semijoin reductions
- Cyclic conjunctive queries

Semijoin Reducer

$$Q(x,y,z) = R(x,y) \bowtie S(y,z) \bowtie T(z,w)$$

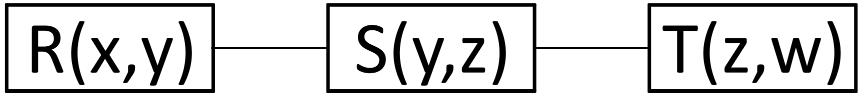
A full reducer is

?

Semijoin Reducer

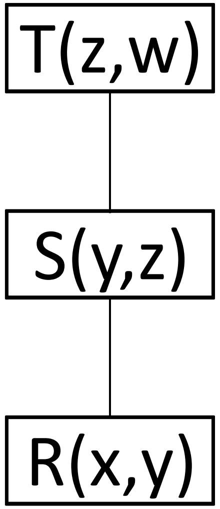
$$Q(x,y,z) = R(x,y) \bowtie S(y,z) \bowtie T(z,w)$$

A full reducer is



?

Join Tree for Q



Semijoin Reducer

$$Q(x,y,z) = R(x,y) \bowtie S(y,z) \bowtie T(z,w)$$

A full reducer is

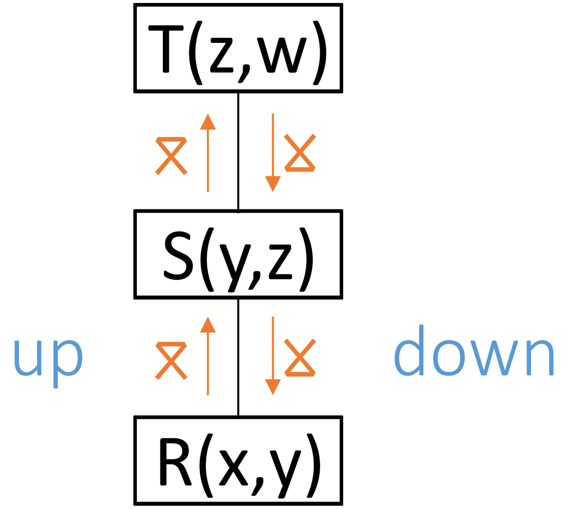
$$R(x,y) \underset{\ltimes}{\overset{\rtimes}{\rightleftarrows}} S(y,z) \underset{\ltimes}{\overset{\rtimes}{\rightleftarrows}} T(z,w)$$

$$\begin{aligned}
S_1(y,z) &= S(y,z) \ltimes R(x,y) \\
T_1(z,y) &= T(z,y) \ltimes S_1(y,z) \\
S_2(z,y) &= S_1(y,z) \ltimes T_1(z,y) \\
R_1(x,y) &= R(x,y) \ltimes S_2(y,z)
\end{aligned}$$

The rewritten query is

?

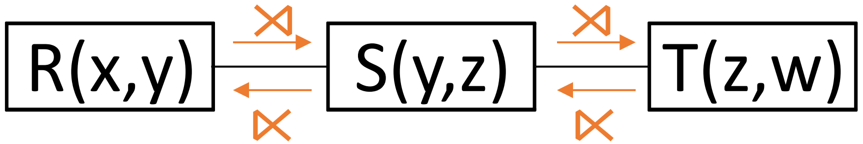
Join Tree for Q



Semijoin Reducer

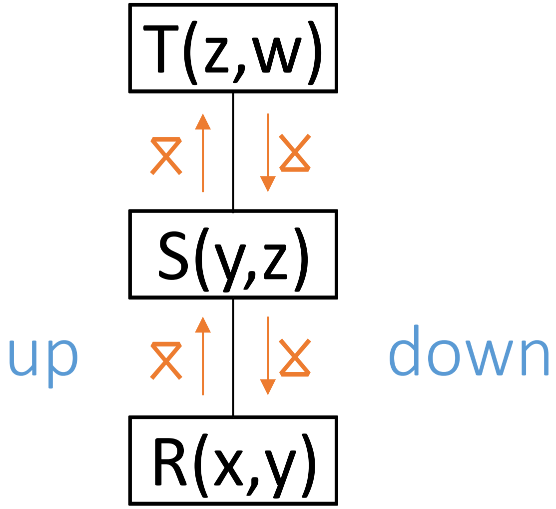
$$Q(x,y,z) = R(x,y) \bowtie S(y,z) \bowtie T(z,w)$$

A full reducer is



$$S_1(y,z) = S(y,z) \ltimes R(x,y)$$
$$T_1(z,y) = T(z,y) \ltimes S_1(y,z)$$
$$S_2(z,y) = S_1(y,z) \ltimes T_1(z,y)$$
$$R_1(x,y) = R(x,y) \ltimes S_2(y,z)$$

Join Tree for Q



The rewritten query is

$$Q(x,y,z) = R_1(x,y) \bowtie S_2(y,z) \bowtie T_1(z,w)$$

Semi-join reducers

GYO ear removal

- remove isolated nodes (variables)
- remove consumed or empty edges (atoms)

Q(x,y,z) :- R(x,y), S(y,z), T(x,z)

Join tree

?

Query hypergraph

?

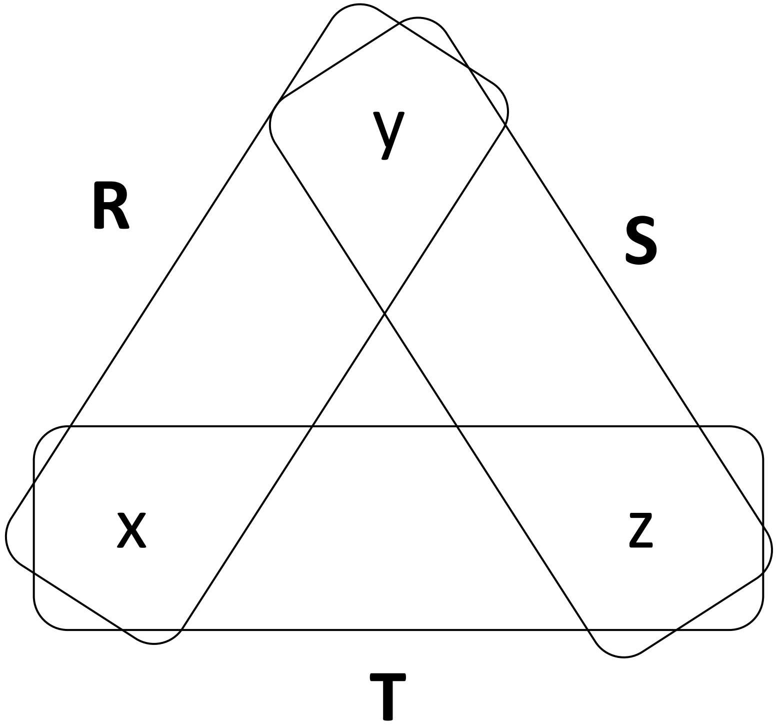
Semi-join reducers

GYO ear removal
- remove isolated nodes (variables)
- remove consumed or empty edges (atoms)

Q(x,y,z) :- R(x,y), S(y,z), T(x,z)

Join tree

?

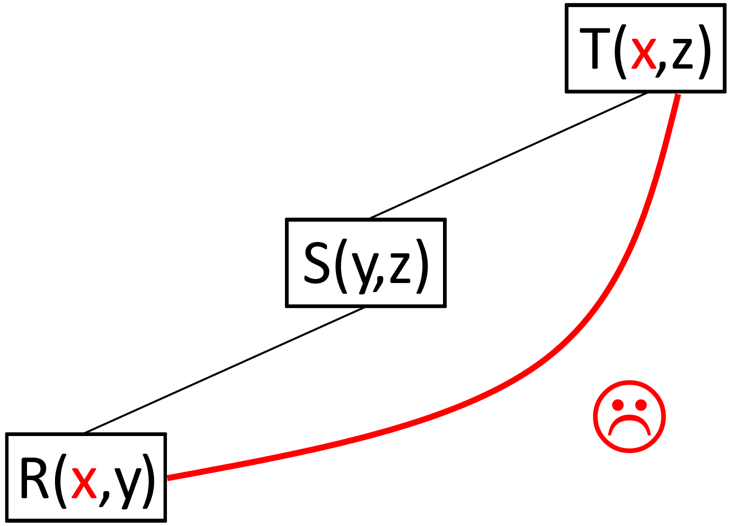


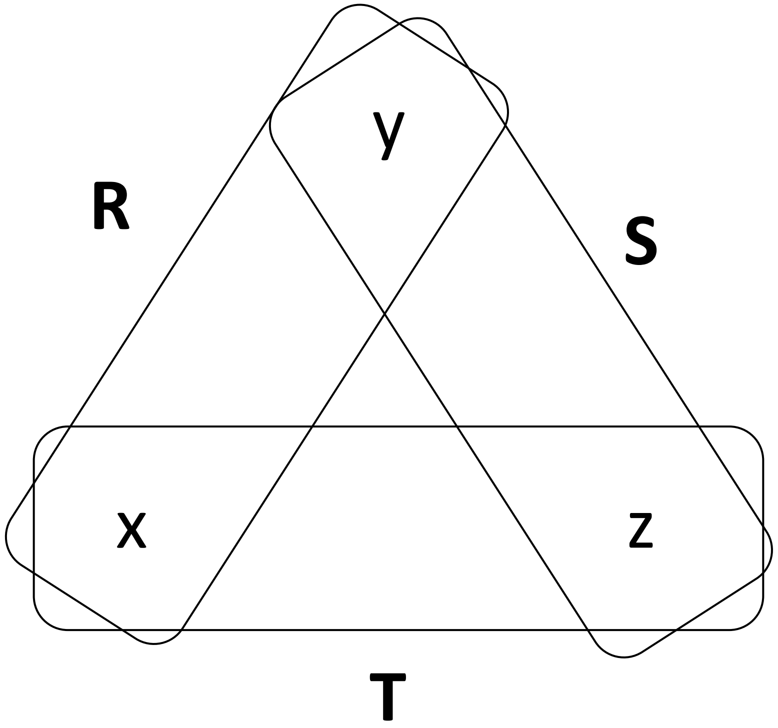
Semi-join reducers

GYO ear removal

- remove isolated nodes (variables)
- remove consumed or empty edges (atoms)

Q(x,y,z) :- R(x,y), S(y,z), T(x,z)





Semi-join reducers

GYO ear removal
- remove isolated nodes (variables)
- remove consumed or empty edges (atoms)

Q(x,y,z) :- R(x,y), S(y,z), T(x,z), W(x,y,z)

Join tree

?

Query hypergraph

?

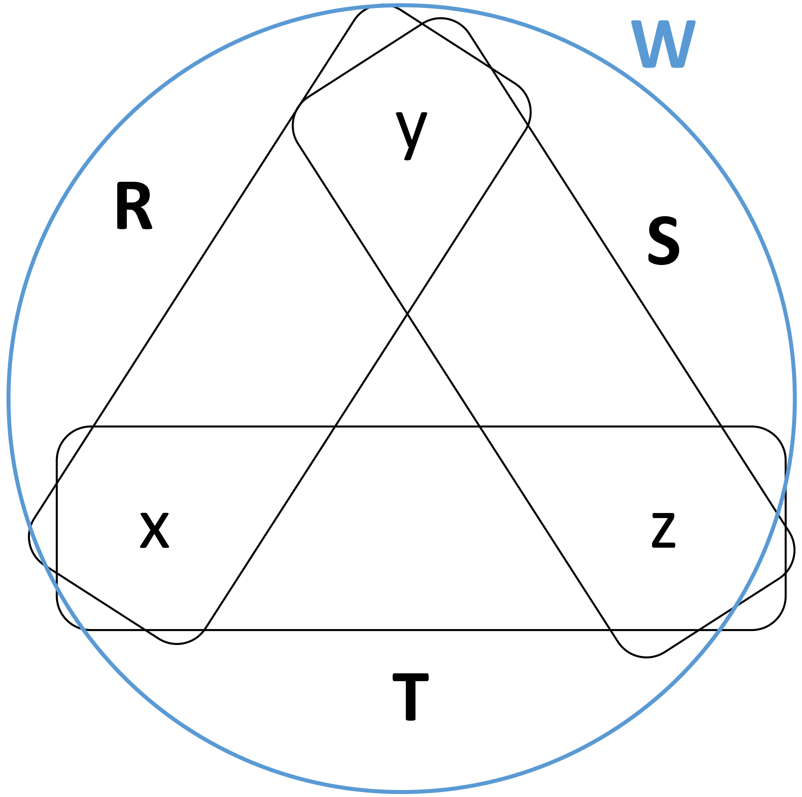
Semi-join reducers

GYO ear removal
- remove isolated nodes (variables)
- remove consumed or empty edges (atoms)

Q(x,y,z) :- R(x,y), S(y,z), T(x,z), W(x,y,z)

Join tree

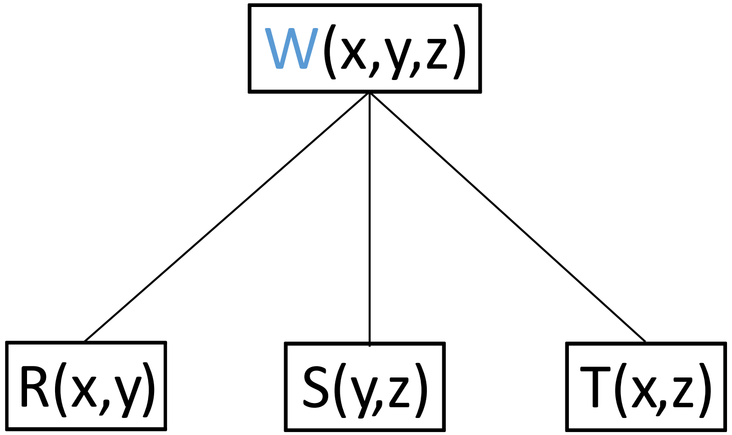
?



Semi-join reducers

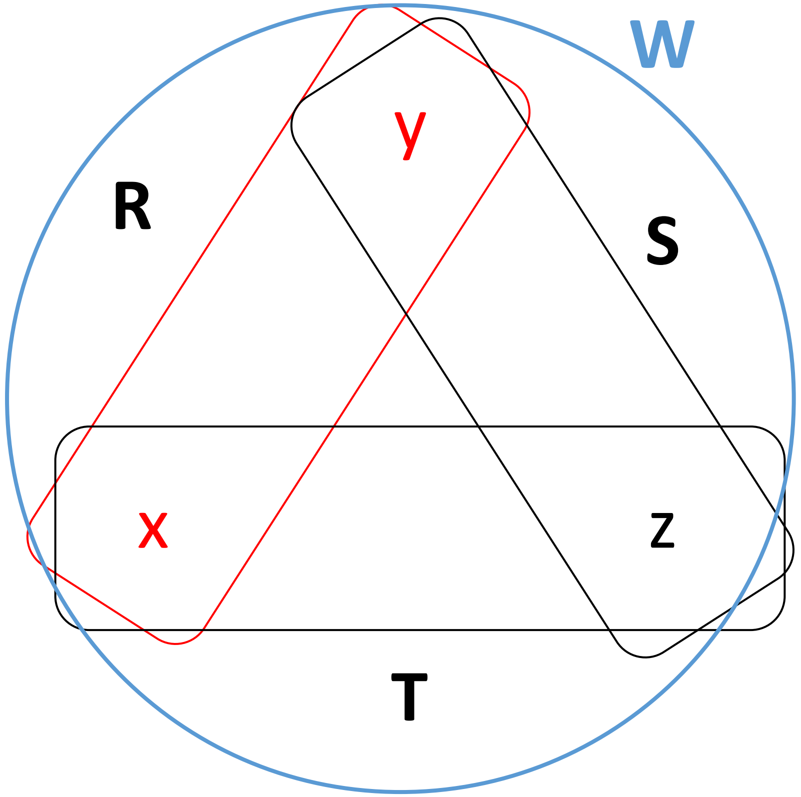
GYO ear removal
- remove isolated nodes (variables)
- remove consumed or empty edges (atoms)

Q(x,y,z) :- R(x,y), S(y,z), T(x,z), W(x,y,z)



Full reducer

?

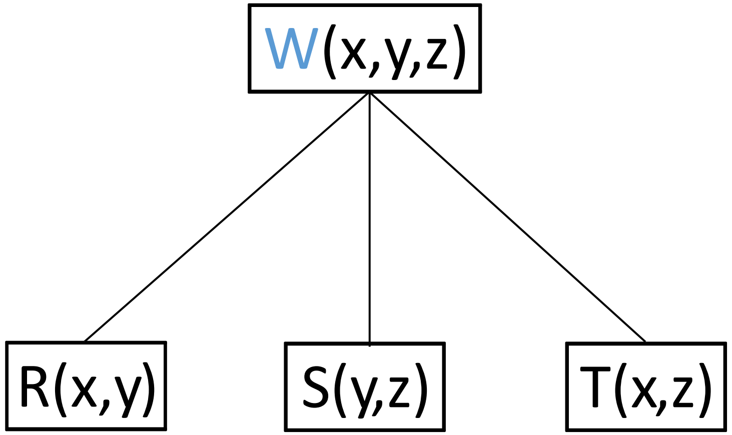


Semi-join reducers

GYO ear removal
- remove isolated nodes (variables)
- remove consumed or empty edges (atoms)

Q(x,y,z) :- R(x,y), S(y,z), T(x,z), W(x,y,z)

$$
\begin{aligned}
W_1(x,y,z) &= W(x,y,z) \ltimes R(x,y) \\
W_2(x,y,z) &= W_1(x,y,z) \ltimes S(y,z) \\
W_3(x,y,z) &= W_2(x,y,z) \ltimes T(x,z) \\
R_1(x,y) &= R(x,y) \ltimes W_3(x,y,z) \\
S_1(y,z) &= S(y,z) \ltimes W_3(x,y,z) \\
T_1(x,z) &= T(x,z) \ltimes W_3(x,y,z)
\end{aligned}
$$



$$Q(x,y,z) = R_1(x,y) \bowtie S_1(y,z) \bowtie T_1(x,z) \bowtie W_3(x,y,z)$$

?

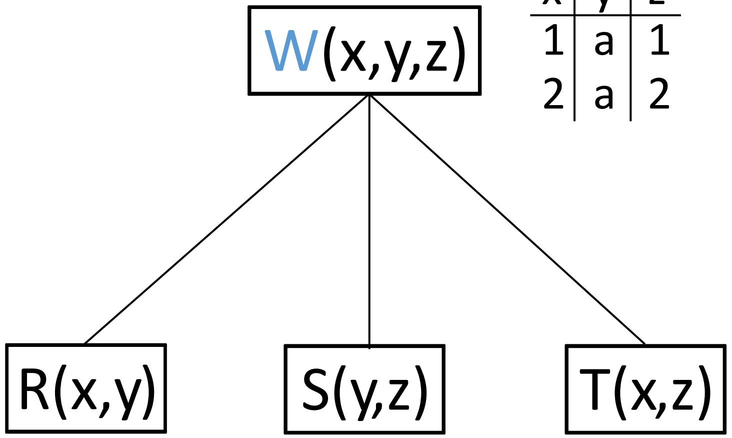
Semi-join reducers

GYO ear removal

- remove isolated nodes (variables)
- remove consumed or empty edges (atoms)

Q(x,y,z) :- R(x,y), S(y,z), T(x,z), W(x,y,z)

$$W_1(x,y,z) = W(x,y,z) \ltimes R(x,y)$$
$$W_2(x,y,z) = W_1(x,y,z) \ltimes S(y,z)$$
$$W_3(x,y,z) = W_2(x,y,z) \ltimes T(x,z)$$
$$R_1(x,y) = R(x,y) \ltimes W_3(x,y,z)$$
$$S_1(y,z) = S(y,z) \ltimes W_3(x,y,z)$$
$$T_1(x,z) = T(x,z) \ltimes W_3(x,y,z)$$



x	y	z
1	a	1
2	a	2

x	y
1	a
2	a

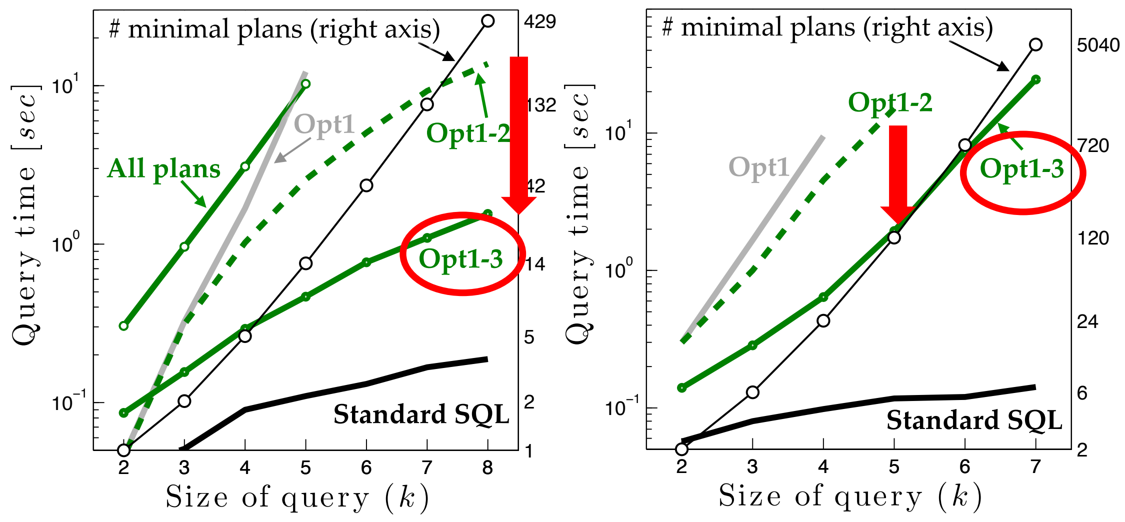
y	z
a	1
a	2

x	z
1	1
2	2

$$\cancel{Q(x,y,z) = R_1(x,y) \bowtie S_1(y,z) \bowtie T_1(x,z) \bowtie W_3(x,y,z)}$$

$$Q(x,y,z) = R_1(x,y) \bowtie W_3(x,y,z) \bowtie S_1(y,z) \bowtie T_1(x,z)$$

Semi-join reductions can be extremely powerful



(a) *k*-chain queries (b) *k*-star queries

Fig. 16 While the query complexity is exponential (number of minimal plans are shown on the right side), our optimizations can even evaluate a very large number of minimal plans (here shown up to 429 for a 8-chain query and 5040 (!) for a 7-star query).

6.3 Opt. 3: Deterministic semi-join reduction

The most expensive operations in probabilistic query plans are the group-bys for the probabilistic project operations. These are often applied early in the plans to tuples which are later pruned and do not contribute to the final query result. Our third optimization is to first apply a *full semi-join reduction on the input relations* before starting the probabilistic evaluation from these *reduced input relations*.

We like to draw here an important connection to [54], which introduces the idea of "lazy plans" and shows orders of magnitude performance improvements for safe plans by computing confidences not after each join and projection, but rather at the very end of the plan. We note that our semi-join reduction *serves the same purpose* with similar performance improvements and also apply for safe queries. The advantage of semi-join reductions, however, is that we do not require any modifications to the query engine.

Source: Gatterbauer, Suciu. Dissociation and propagation for approximate lifted inference with standard relational database management systems, VLDBJ 2017. https://arxiv.org/pdf/1310.6257

Outline: Cyclic conjunctive queries

- Acyclic conjunctive queries
- Cyclic conjunctive queries
  - 2SAT (a detour)
  - Tree decompositions
  - AGM bound (join processing of cyclic queries)
  - Duality in Linear programming (a quick primer)
  - Worst-case optimal joins
  - Hypertree & other decompositions
  - Optimal joins

cycles make everything more complicated ☹

Why cyclic queries (other than social networks)

```
Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink, cost)
```

2. Specify or choose a Query [Supported grammar]

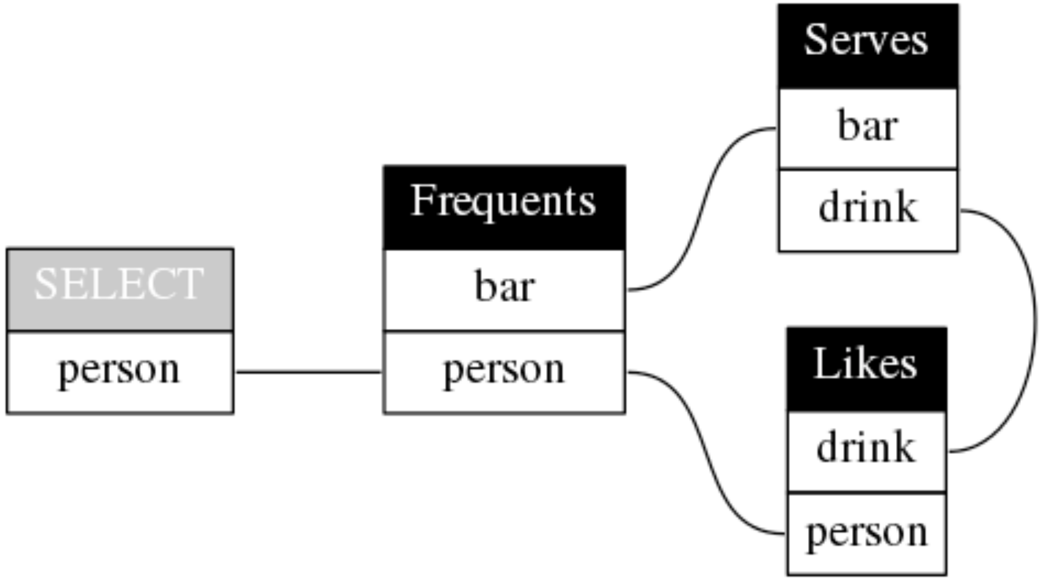
104 Bars: Persons who frequent some bar that serves some drink they like.

Why cyclic queries (other than social networks)

```
Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink, cost)
```

2. Specify or choose a Query [Supported grammar]

104 Bars: Persons who frequent some bar that serves some drink they like.



Source: http://queryviz.com/online/

Why cyclic queries (other than social networks)

```
Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink, cost)
```

2. Specify or choose a Query [Supported grammar]

104 Bars: Persons who frequent some bar that serves some drink they like.

```
SELECT  F1.person
FROM    Frequents F1
WHERE   exists
        (SELECT *
        FROM    Serves S2
        WHERE   S2.bar = F1.bar
        AND     exists
                (SELECT *
                FROM    Likes L3
                WHERE   L3.person = F1.person
                AND     S2.drink = L3.drink))
```

Source: http://queryviz.com/online/

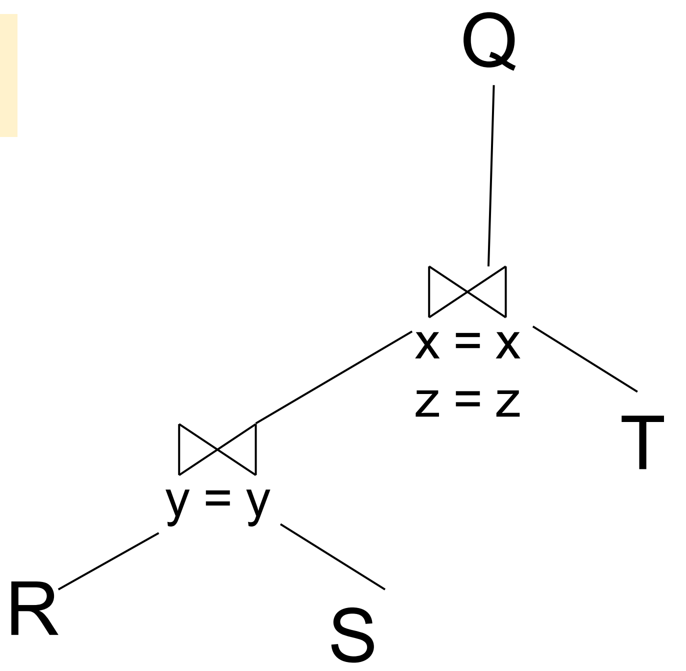
Joins in databases: one-at-a-time

Efficient multi-way join processing

Q(x,y,z) :- R(x,y), S(y,z), T(x,z)

Three plans

- (R ⋈ S) ⋈ T
- (S ⋈ T) ⋈ R
- (T ⋈ R) ⋈ S



Can we do better for cyclic queries? ☺

Outline: Cyclic conjunctive queries

- Acyclic conjunctive queries
- Cyclic conjunctive queries
  - 2SAT (a detour)
  - Tree decompositions
  - AGM bound (join processing of cyclic queries)
  - Duality in Linear programming (a quick primer)
  - Worst-case optimal joins
  - Hypertree & other decompositions
  - Optimal joins

2SAT

$\varphi = (x \vee y) \wedge (\neg y \vee z) \wedge (\neg x \vee \neg z) \wedge (z \vee y)$

- Instance: A 2-CNF formula φ
- Problem: To decide if φ is satisfiable

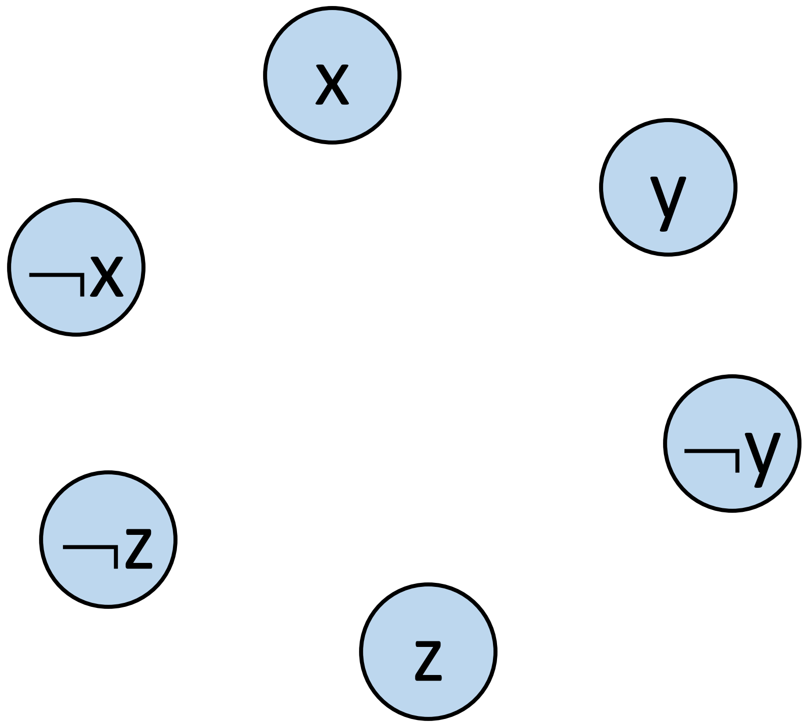
- Theorem: 2SAT is polynomial-time decidable.
  - Proof: We'll show how to solve this problem efficiently using path searches in graphs…

- Background: Given a graph $G=(V,E)$ and two vertices $s,t \in V$, finding if there is a path from s to t in G is polynomial-time decidable. Use some search algorithm (DFS/BFS).

2SAT: Graph Construction

$\varphi = (x \lor y) \land (\neg y \lor z) \land (\neg x \lor \neg z) \land (z \lor y)$

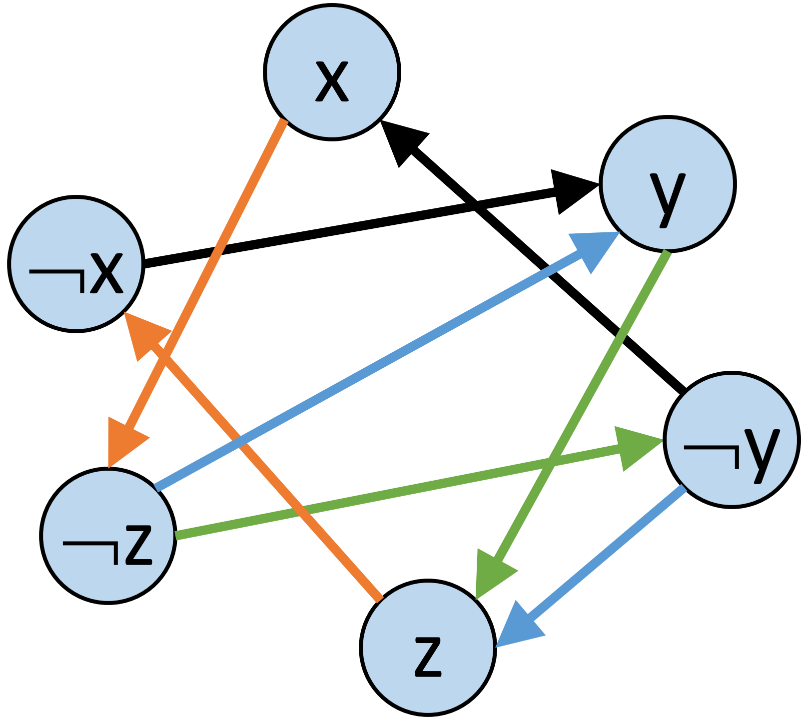
- Vertex for each variable and a negation of a variable



2SAT: Graph Construction

$\varphi = (x \vee y) \wedge (\neg y \vee z) \wedge (\neg x \vee \neg z) \wedge (z \vee y)$

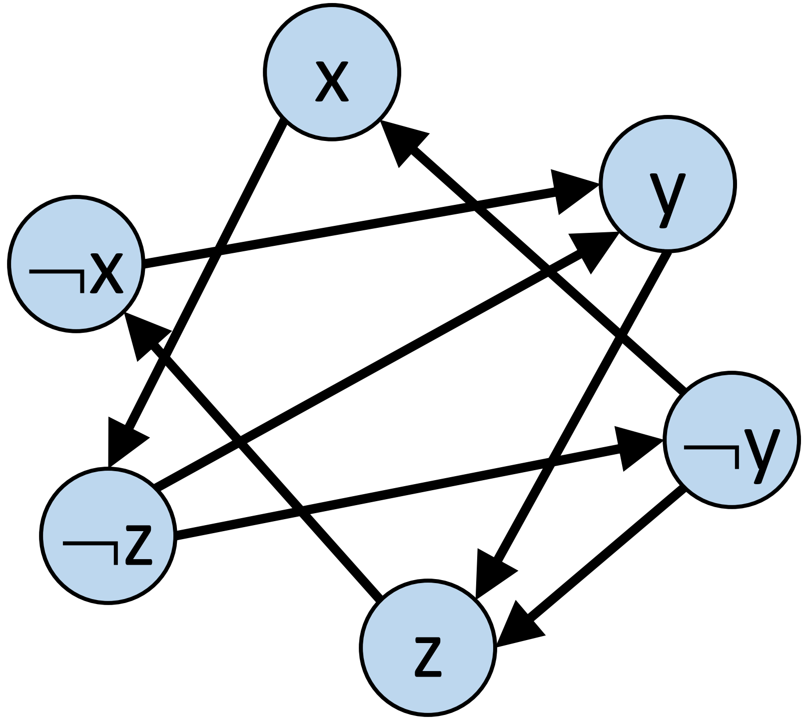
- Vertex for each variable and a negation of a variable
- Edge $(\neg x \rightarrow y)$ iff there exists a clause equivalent to $(x \vee y)$
  - Recall $(x \vee y)$ same as $(\neg x \Rightarrow y)$ and $(\neg y \Rightarrow x)$, thus also $(\neg y \rightarrow x)$



2SAT: Graph Construction

$\varphi = (x \vee y) \wedge (\neg y \vee z) \wedge (\neg x \vee \neg z) \wedge (z \vee y)$

- Claim: a 2-CNF formula φ is unsatisfiable iff there exists a variable x, such that:
  - there is a path from x to $\neg$x in the graph, and
  - there is a path from $\neg$x to x in the graph

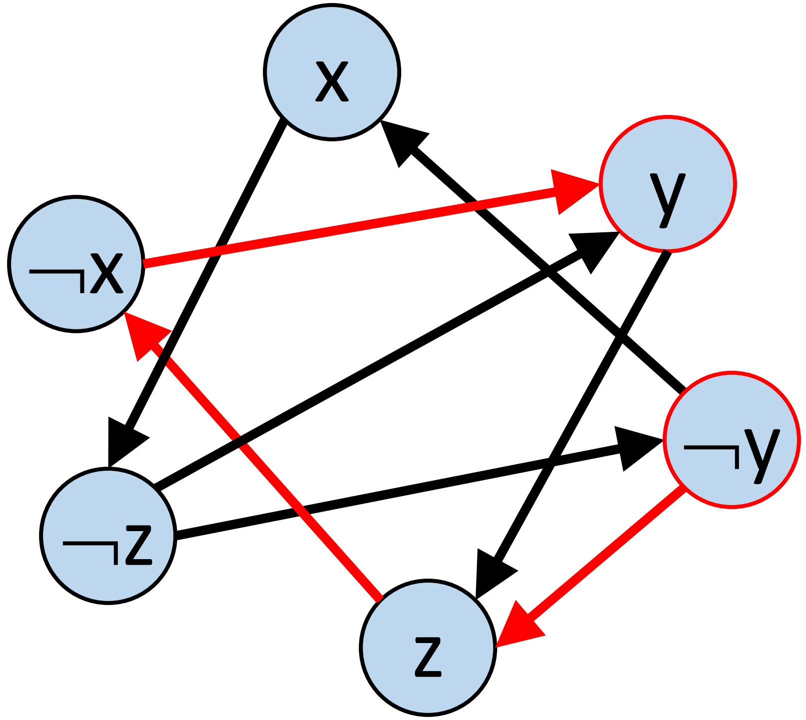


2SAT: Graph Construction

$\varphi = (x \lor y) \land (\neg y \lor z) \land (\neg x \lor \neg z) \land (z \lor y)$

- Claim: a 2-CNF formula φ is unsatisfiable iff there exists a variable x, such that:
  - there is a path from x to $\neg$x in the graph, and
  - there is a path from $\neg$x to x in the graph

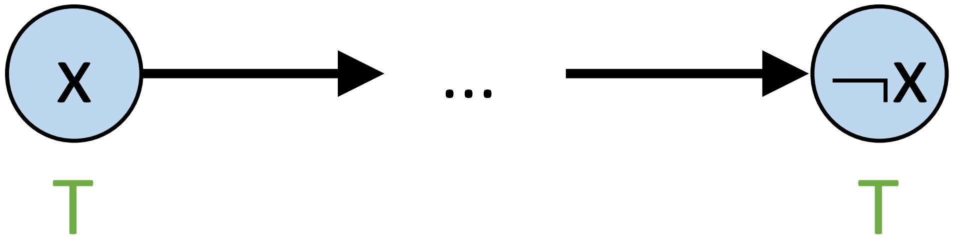
not enough,
needs both directions!



Correctness (1)

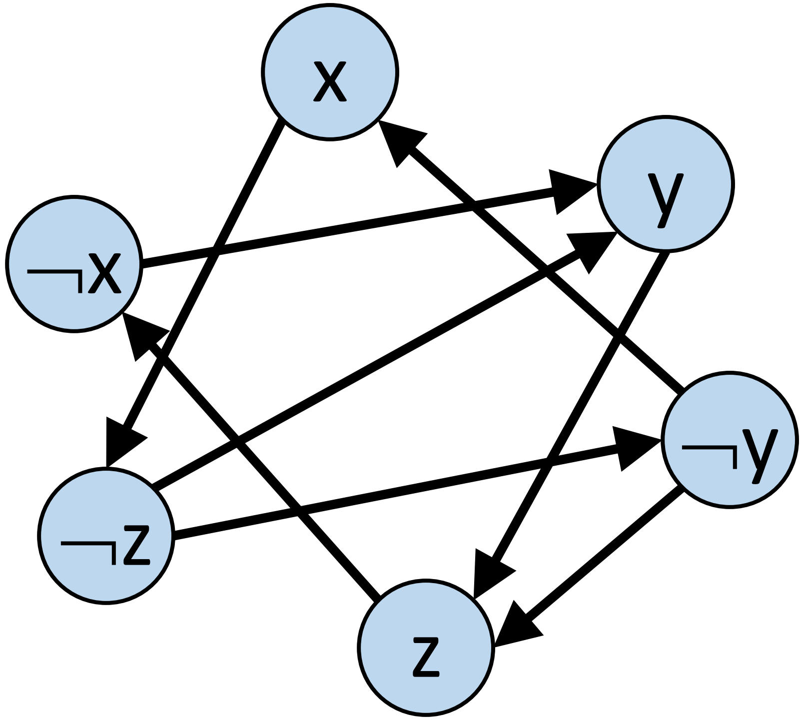
$\varphi = (x \lor y) \land (\neg y \lor z) \land (\neg x \lor \neg z) \land (z \lor y)$

- Suppose there are paths $x..\neg x$ and $\neg x..x$ for some variable x, but there's also a satisfying assignment ρ.
  - If $\rho(x)=T$:



  - Similarly for $\rho(x)=F$...

recall, needs to hold i both directions!



Correctness (2)

$\varphi = (x \vee y) \wedge (\neg y \vee z) \wedge (\neg x \vee \neg z) \wedge (z \vee y)$

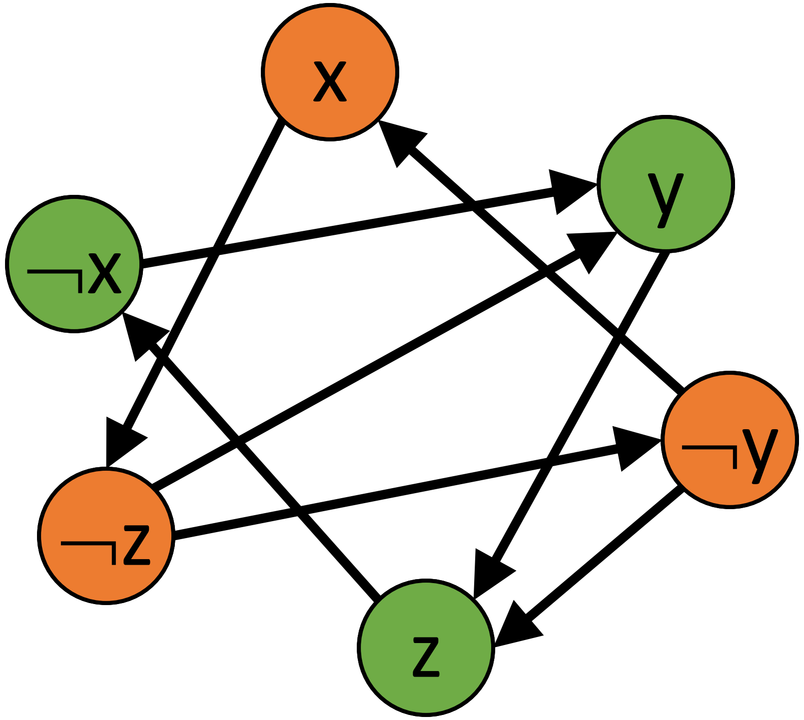
- Suppose there are no such paths.
- Construct an assignment as follows:

1. pick an unassigned literal α, with no path from α to $\neg\alpha$, and assign it T

2. assign T to all reachable vertices

3. assign F to their negations

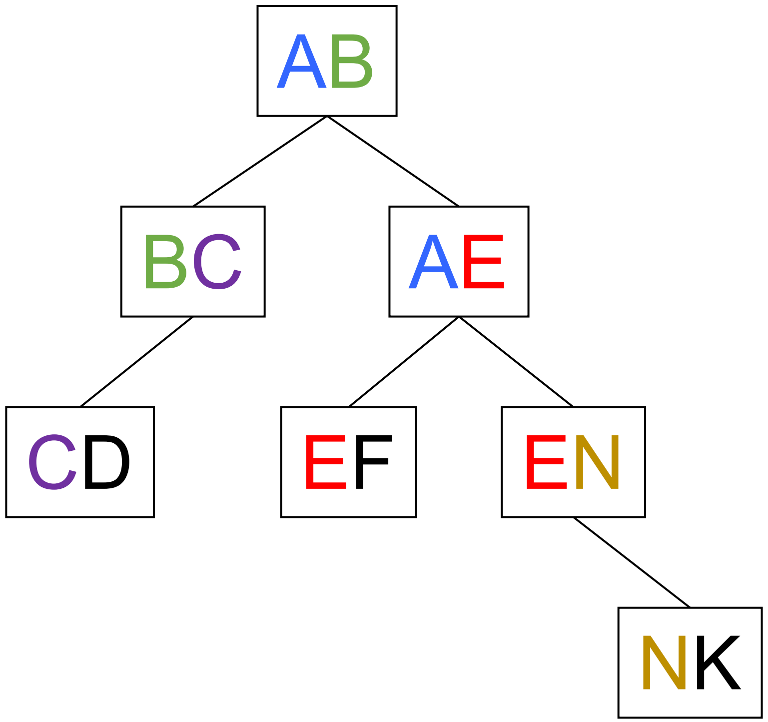
4. Repeat until all vertices are assigned



Outline: Cyclic conjunctive queries

- Acyclic conjunctive queries
- Cyclic conjunctive queries
  - 2SAT (a detour)
  - Tree decompositions
  - AGM bound (join processing of cyclic queries)
  - Duality in Linear programming (a quick primer)
  - Worst-case optimal joins
  - Hypertree & other decompositions
  - Optimal joins

Definition of an *attribute-connected tree* (also *running intersection property* or *coherence*)



A tree is attribute-connected if the sub-tree induced by each attribute is connected

Tree decomposition

A tree decomposition of graph $G(N, E)$ is a tree $T(V, F)$ and a subset $N_v \subseteq N$ assigned to each vertex $v \in V$ s.t.:

(1) Node coverage: Every vertex of G is assigned least one vertex in T

(2) Edge coverage: For every edge e of G, there is a vertex in T that contains both ends of e

(3) Coherence: The tree is "attribute-connected"

The width of a tree decomposition is the size of its largest set minus one

Tree decomposition ... of a tree

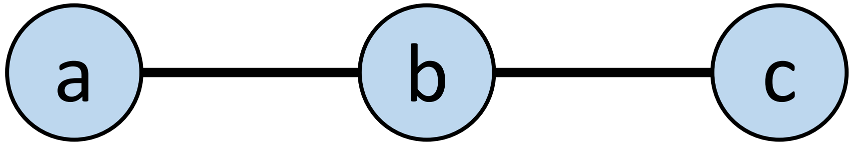
A tree decomposition of graph $G(N, E)$ is a tree $T(V, F)$ and a subset $N_v \subseteq N$ assigned to each vertex $v \in V$ s.t.:

(1) Node coverage: Every vertex of G is assigned least one vertex in T

(2) Edge coverage: For every edge e of G, there is a vertex in T that contains both ends of e

(3) Coherence: The tree is "attribute-connected"

The width of a tree decomposition is the size of its largest set minus one



tree decomposition

?

Tree decomposition ... of a tree

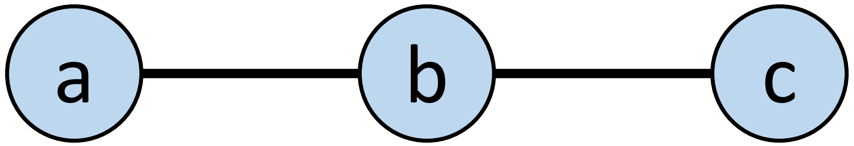
A tree decomposition of graph $G(N, E)$ is a tree $T(V, F)$ and a subset $N_v \subseteq N$ assigned to each vertex $v \in V$ s.t.:

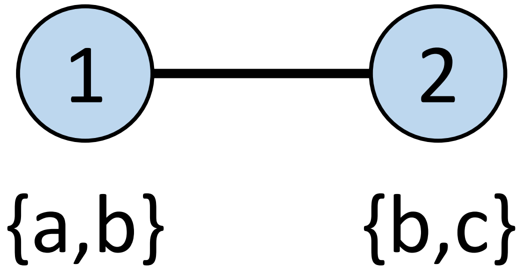
(1) Node coverage: Every vertex of G is assigned least one vertex in T

(2) Edge coverage: For every edge e of G, there is a vertex in T that contains both ends of e

(3) Coherence: The tree is "attribute-connected"

The width of a tree decomposition is the size of its largest set minus one





That's why treewidth defined as max cardinality - 1

Tree decomposition example

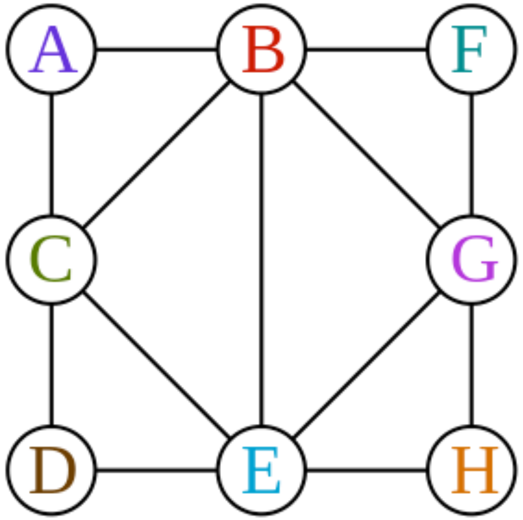
A tree decomposition of graph $\mathrm{G(N, E)}$ is a tree $\mathrm{T(V, F)}$ and a subset $\mathrm{N_v \subseteq N}$ assigned to each vertex $\mathrm{v \in V}$ s.t.:

(1) Node coverage: Every vertex of G is assigned least one vertex in T

(2) Edge coverage: For every edge e of G, there is a vertex in T that contains both ends of e

(3) Coherence: The tree is "attribute-connected"

The width of a tree decomposition is the size of its largest set minus one



tree decomposition

?

Source: https://en.wikipedia.org/wiki/Tree_decomposition

Tree decomposition example

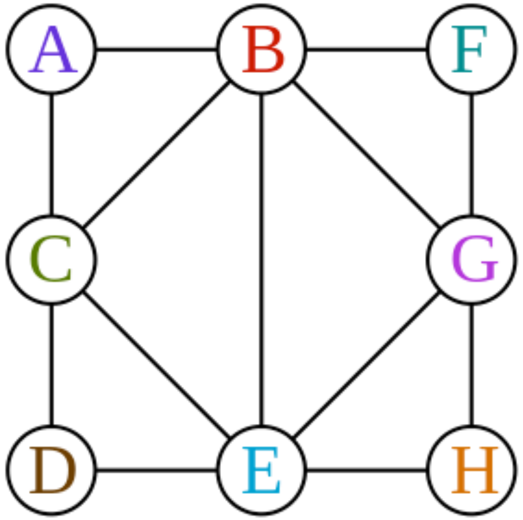
A tree decomposition of graph $G(N, E)$ is a tree $T(V, F)$ and a subset $N_v \subseteq N$ assigned to each vertex $v \in V$ s.t.:

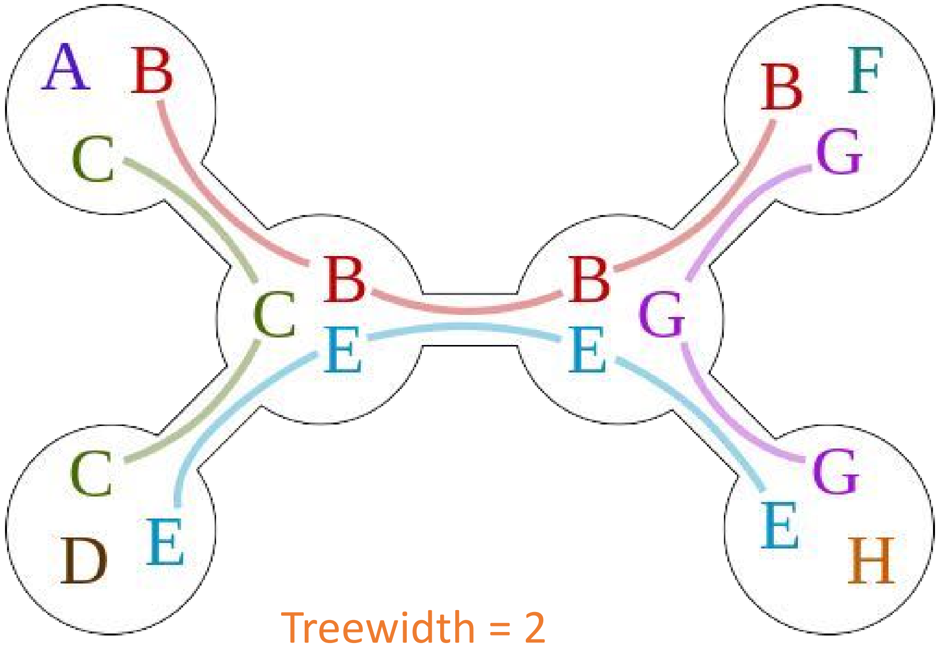
(1) Node coverage: Every vertex of G is assigned least one vertex in T

(2) Edge coverage: For every edge e of G, there is a vertex in T that contains both ends of e

(3) Coherence: The tree is "attribute-connected"

The width of a tree decomposition is the size of its largest set minus one





Treewidth = 2

Notice running intersection property

Source: https://en.wikipedia.org/wiki/Tree_decomposition

Tree decomposition example

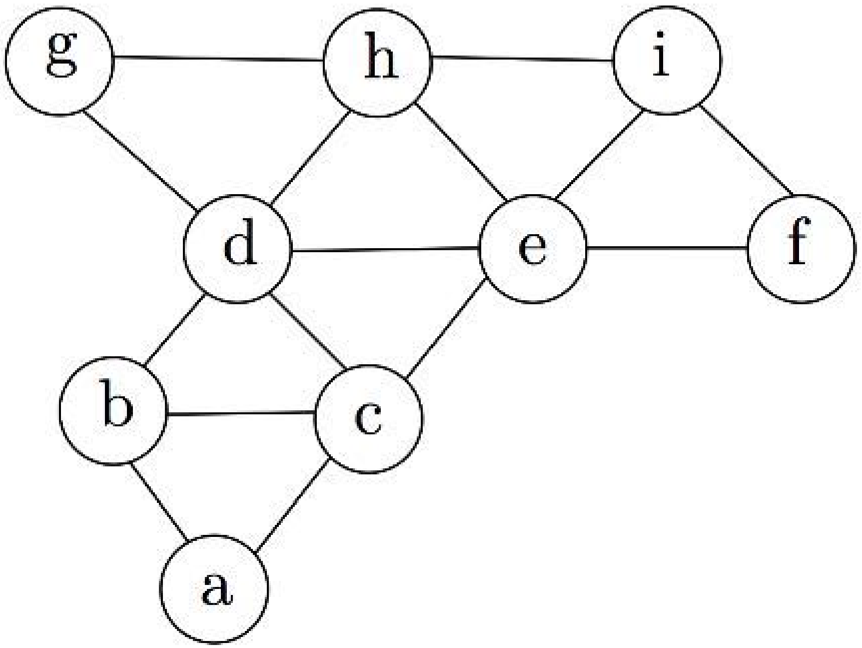
A tree decomposition of graph $G(N, E)$ is a tree $T(V, F)$ and a subset $N_v \subseteq N$ assigned to each vertex $v \in V$ s.t.:

(1) Node coverage: Every vertex of G is assigned least one vertex in T

(2) Edge coverage: For every edge e of G, there is a vertex in T that contains both ends of e

(3) Coherence: The tree is "attribute-connected"

The width of a tree decomposition is the size of its largest set minus one



tree decomposition

?

Source: https://www.mi.fu-berlin.de/en/inf/groups/abi/teaching/lectures/lectures_past/WS0910/V____Discrete_Mathematics_for_Bioinformatics__P1/material/scripts/treedecomposition1.pdf

Tree decomposition example

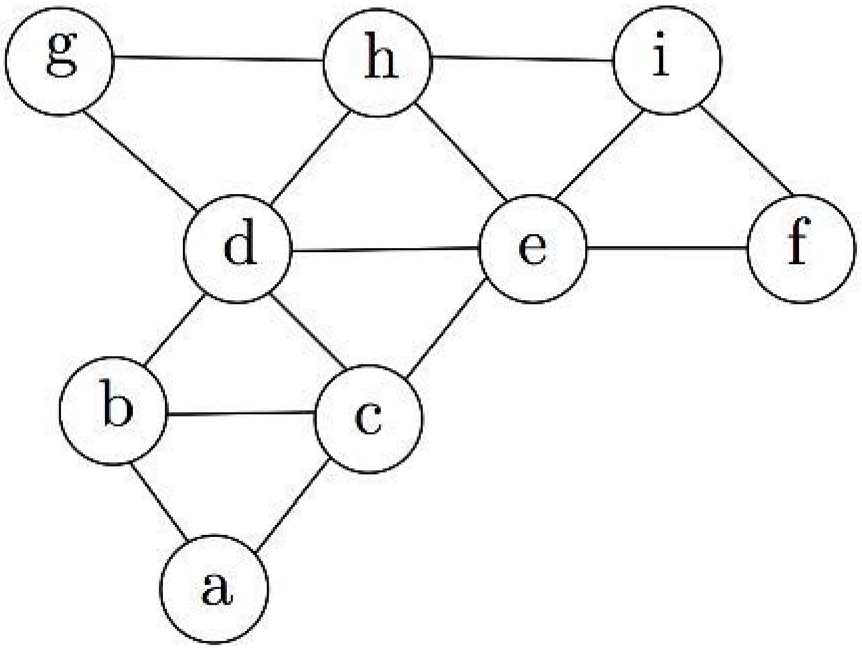
A tree decomposition of graph $G(N, E)$ is a tree $T(V, F)$ and a subset $N_v \subseteq N$ assigned to each vertex $v \in V$ s.t.:

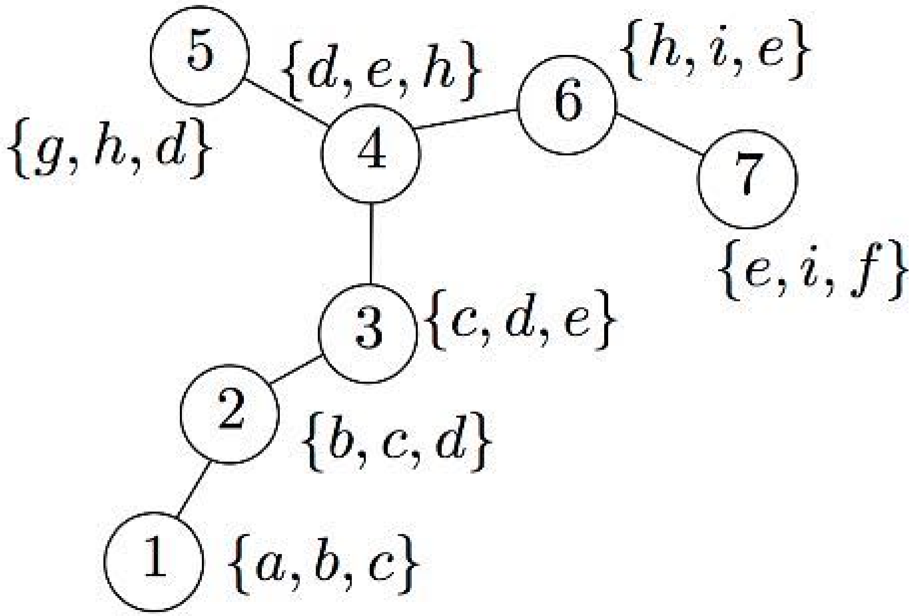
(1) Node coverage: Every vertex of G is assigned least one vertex in T

(2) Edge coverage: For every edge e of G, there is a vertex in T that contains both ends of e

(3) Coherence: The tree is "attribute-connected"

The width of a tree decomposition is the size of its largest set minus one





Source: https://www.mi.fu-berlin.de/en/inf/groups/abi/teaching/lectures/lectures_past/WS0910/V____Discrete_Mathematics_for_Bioinformatics__P1/material/scripts/treedecomposition1.pdf

Tree decomposition of a cycle

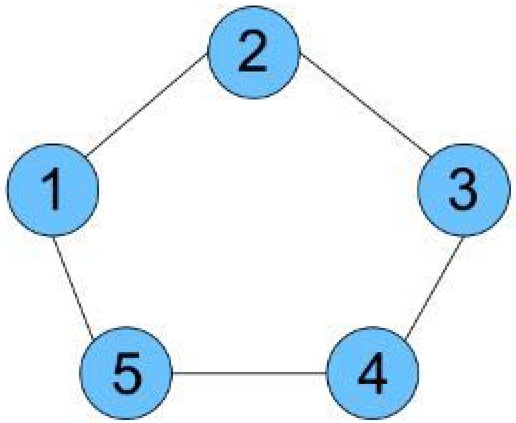
A tree decomposition of graph $\mathrm{G}(\mathrm{N}, \mathrm{E})$ is a tree $\mathrm{T}(\mathrm{V}, \mathrm{F})$ and a subset $\mathrm{N}_\mathrm{v} \subseteq \mathrm{N}$ assigned to each vertex $\mathrm{v} \in \mathrm{V}$ s.t.:

(1) Node coverage: Every vertex of G is assigned least one vertex in T

(2) Edge coverage: For every edge e of G, there is a vertex in T that contains both ends of e

(3) Coherence: The tree is "attribute-connected"

The width of a tree decomposition is the size of its largest set minus one



tree decomposition

?

Tree decomposition of a cycle

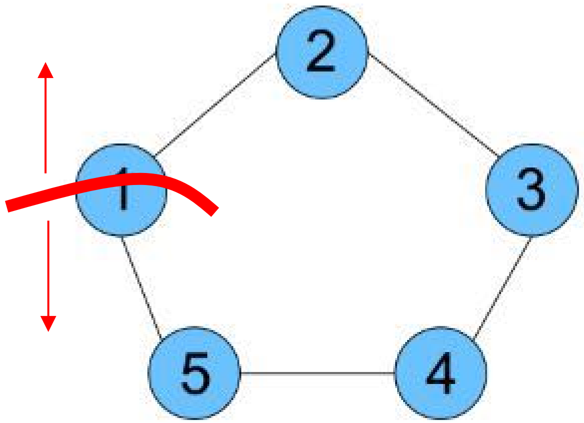
A tree decomposition of graph $G(N, E)$ is a tree $T(V, F)$ and a subset $N_v \subseteq N$ assigned to each vertex $v \in V$ s.t.:

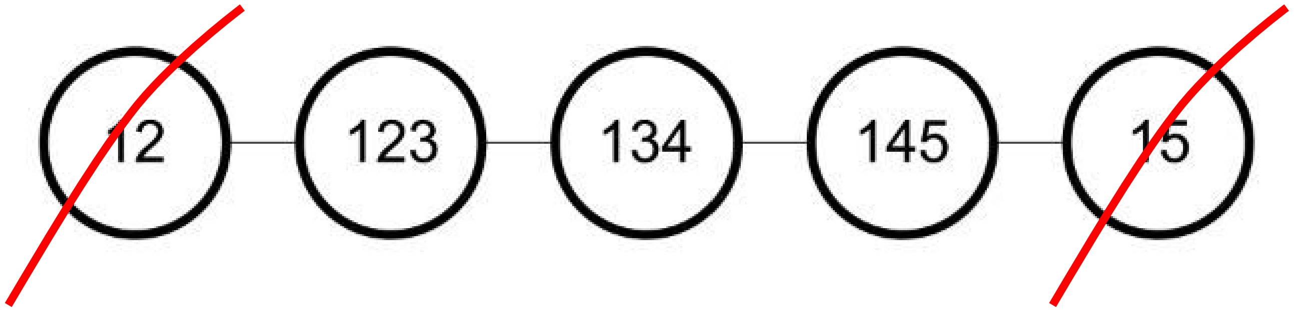
(1) Node coverage: Every vertex of G is assigned least one vertex in T

(2) Edge coverage: For every edge e of G, there is a vertex in T that contains both ends of e

(3) Coherence: The tree is "attribute-connected"

The width of a tree decomposition is the size of its largest set minus one





Tree decomposition of a triangle

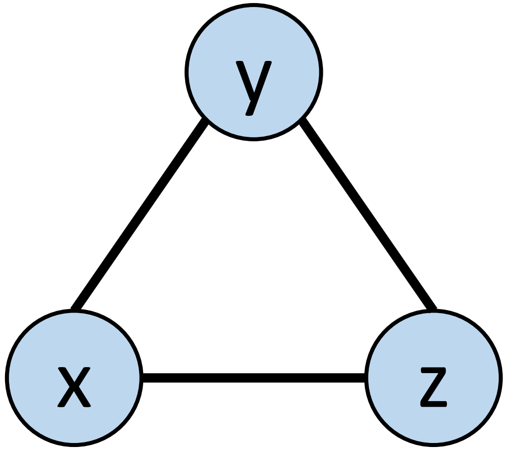
A tree decomposition of graph $\mathrm{G(N, E)}$ is a tree $\mathrm{T(V, F)}$ and a subset $\mathrm{N_v \subseteq N}$ assigned to each vertex $\mathrm{v \in V}$ s.t.:

(1) Node coverage: Every vertex of G is assigned least one vertex in T

(2) Edge coverage: For every edge e of G, there is a vertex in T that contains both ends of e

(3) Coherence: The tree is "attribute-connected"

The width of a tree decomposition is the size of its largest set minus one



tree decomposition

?

Tree decomposition of a triangle

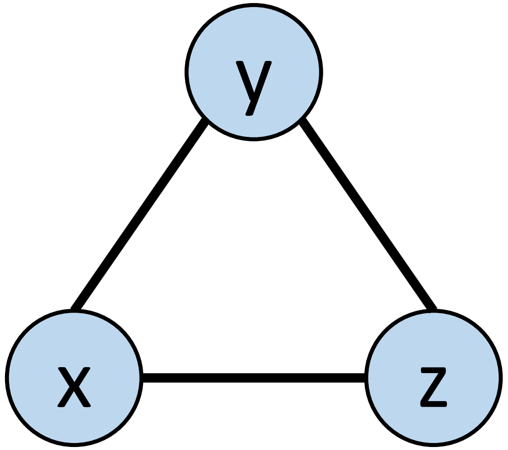
A tree decomposition of graph $G(N, E)$ is a tree $T(V, F)$ and a subset $N_v \subseteq N$ assigned to each vertex $v \in V$ s.t.:

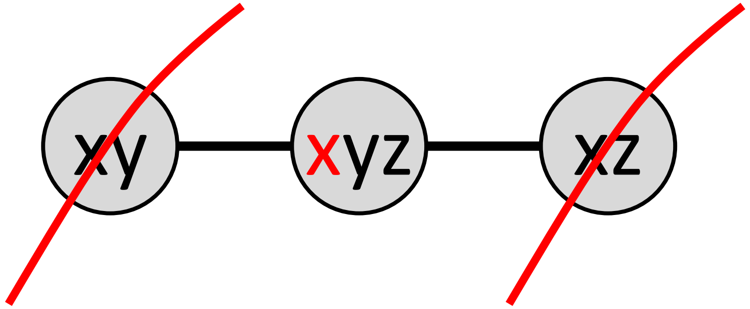
(1) Node coverage: Every vertex of G is assigned least one vertex in T

(2) Edge coverage: For every edge e of G, there is a vertex in T that contains both ends of e

(3) Coherence: The tree is "attribute-connected"

The width of a tree decomposition is the size of its largest set minus one





Tree decomposition of a longer tree

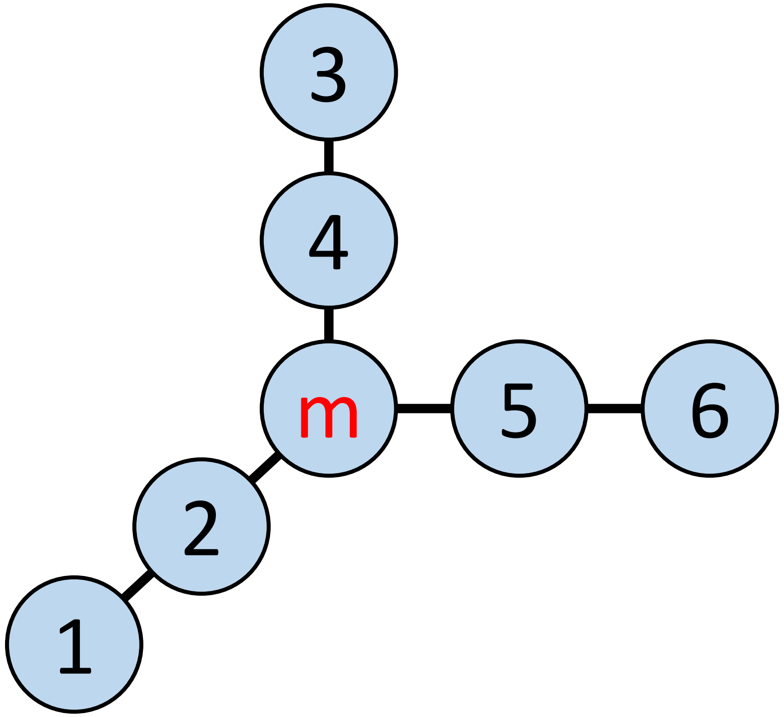
A tree decomposition of graph $\mathrm{G(N,E)}$ is a tree $\mathrm{T(V,F)}$ and a subset $\mathrm{N_v} \subseteq \mathrm{N}$ assigned to each vertex $\mathrm{v} \in \mathrm{V}$ s.t.:

(1) Node coverage: Every vertex of G is assigned least one vertex in T

(2) Edge coverage: For every edge e of G, there is a vertex in T that contains both ends of e

(3) Coherence: The tree is "attribute-connected"

The width of a tree decomposition is the size of its largest set minus one



tree decomposition

?

Tree decomposition of a longer tree

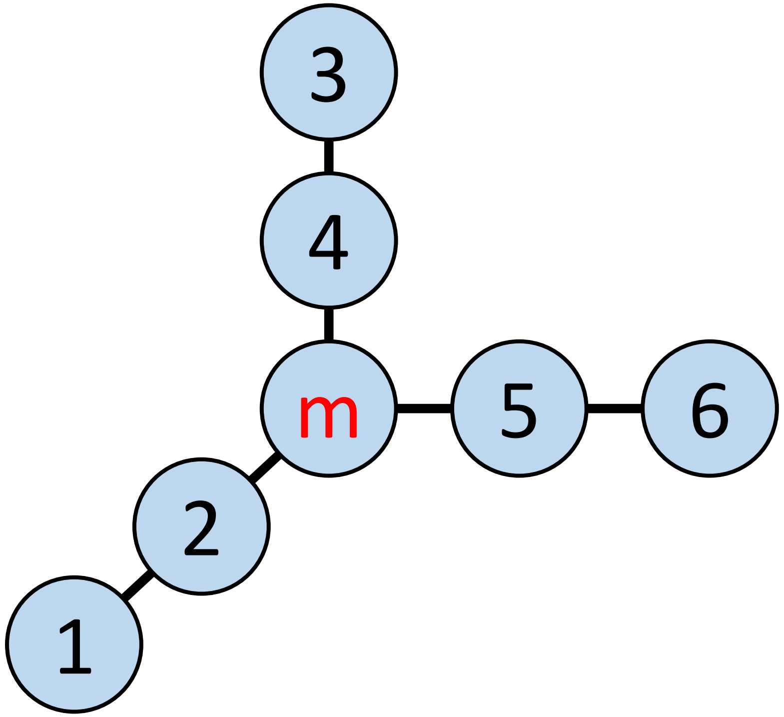
A tree decomposition of graph $\mathrm{G(N, E)}$ is a tree $\mathrm{T(V, F)}$ and a subset $\mathrm{N_v \subseteq N}$ assigned to each vertex $\mathrm{v \in V}$ s.t.:

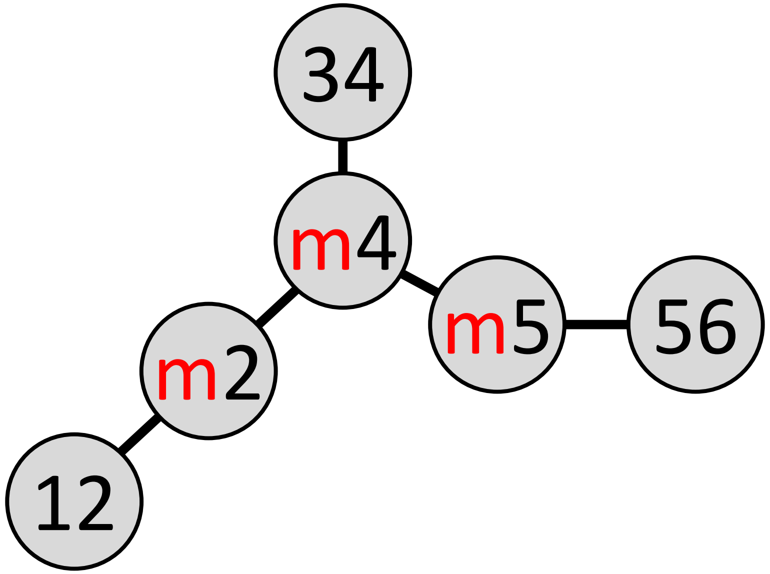
(1) Node coverage: Every vertex of G is assigned least one vertex in T

(2) Edge coverage: For every edge e of G, there is a vertex in T that contains both ends of e

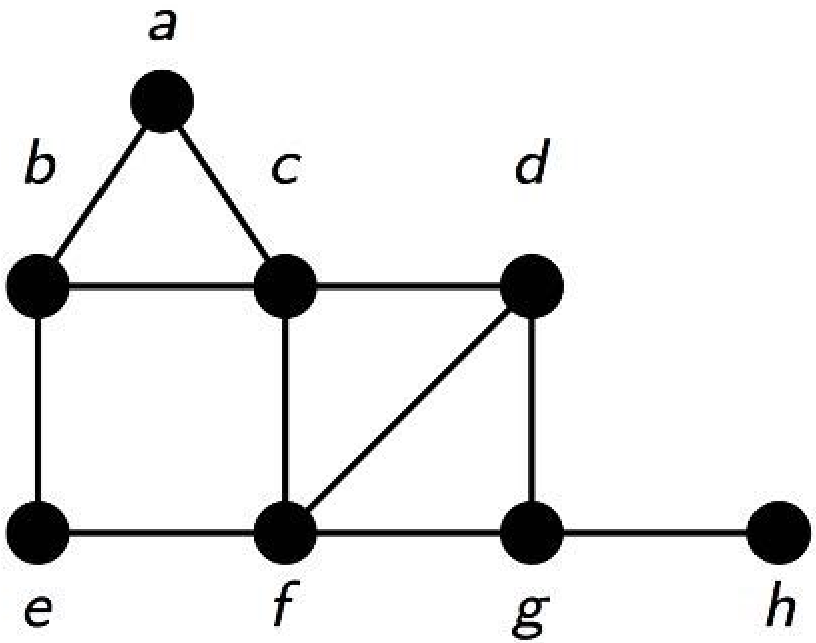
(3) Coherence: The tree is "attribute-connected"

The width of a tree decomposition is the size of its largest set minus one



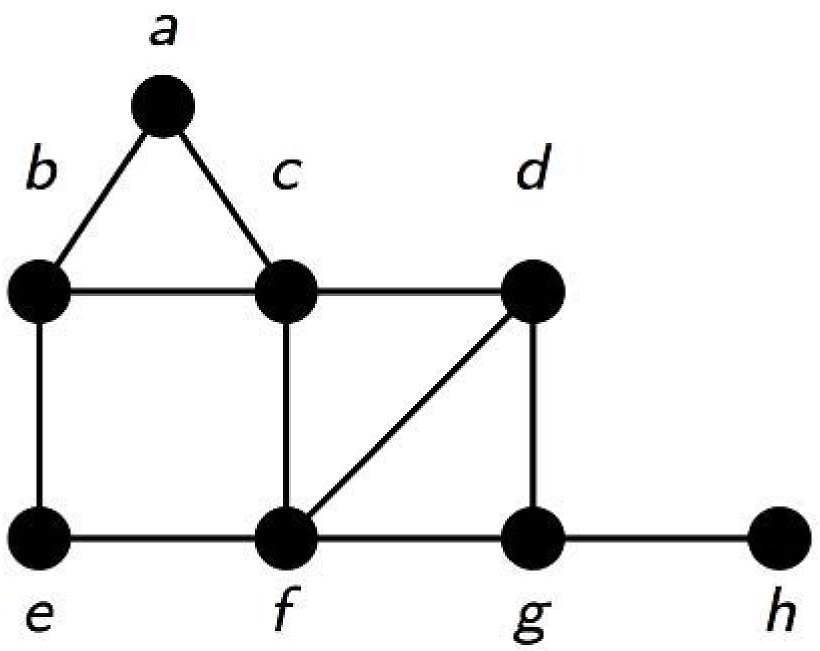


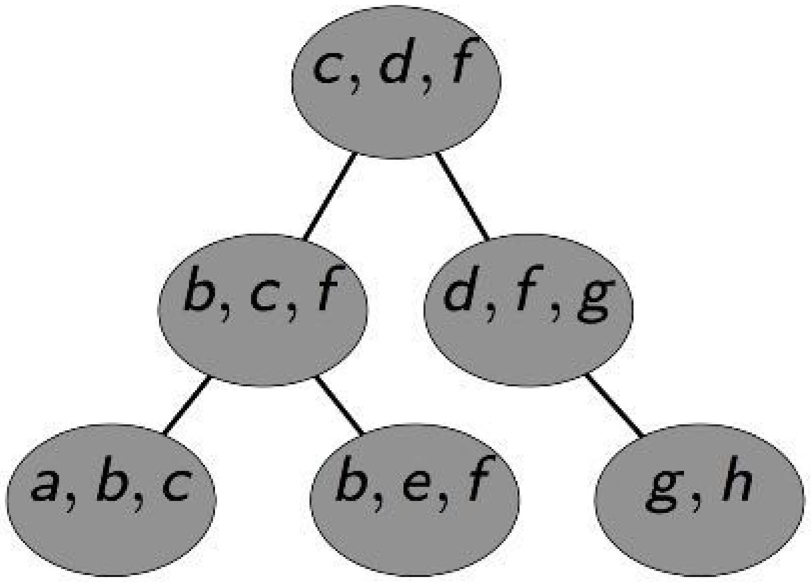
Tree decomposition



Source: 2017 - Marx - Graphs, hypergraphs, and the complexity of conjunctive database queries

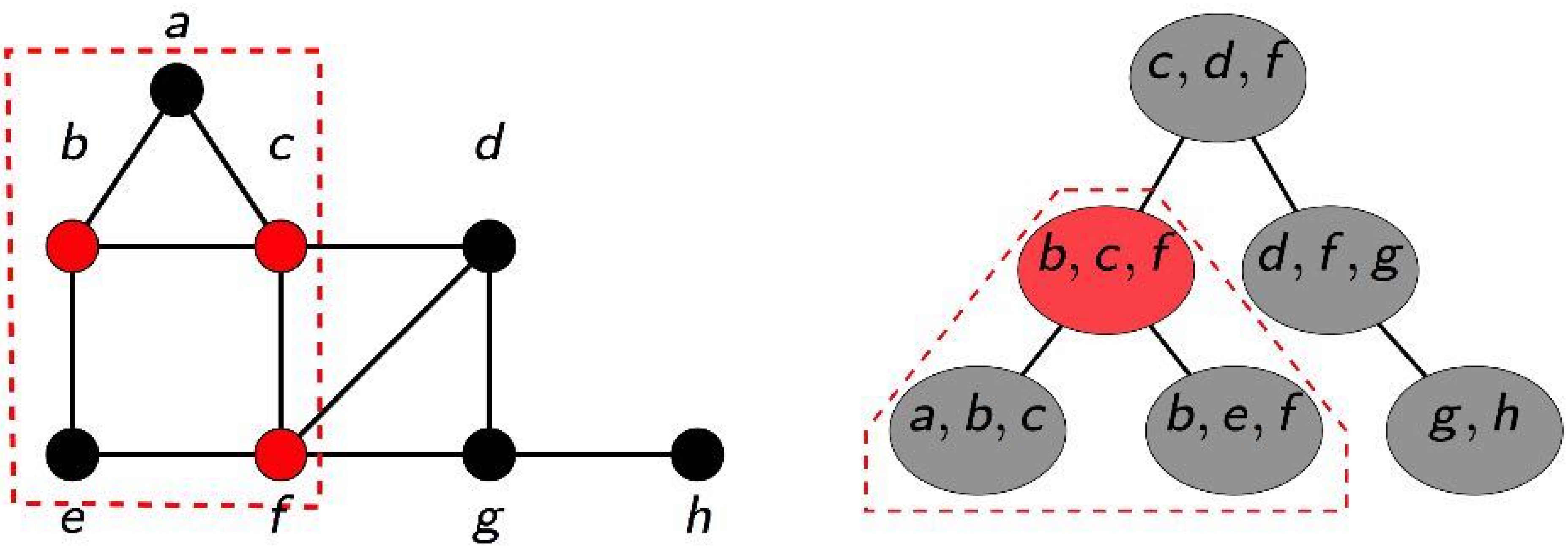
Tree decomposition





Source: 2017 - Marx - Graphs, hypergraphs, and the complexity of conjunctive database queries

Tree decomposition



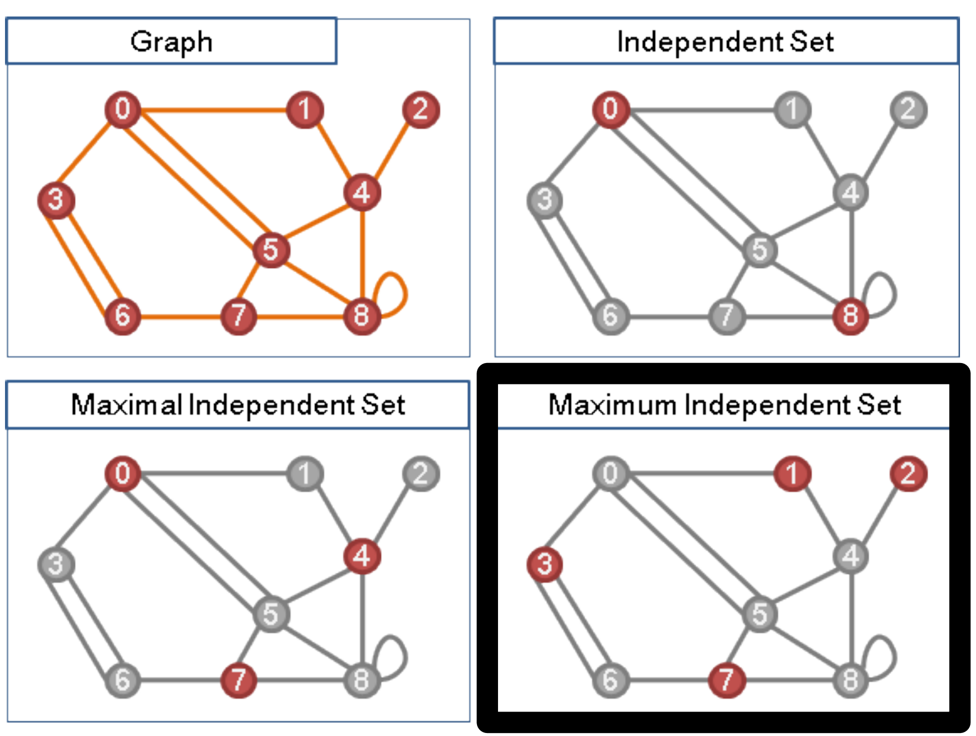
A subtree communicates with the outside world only via the root of the subtree.

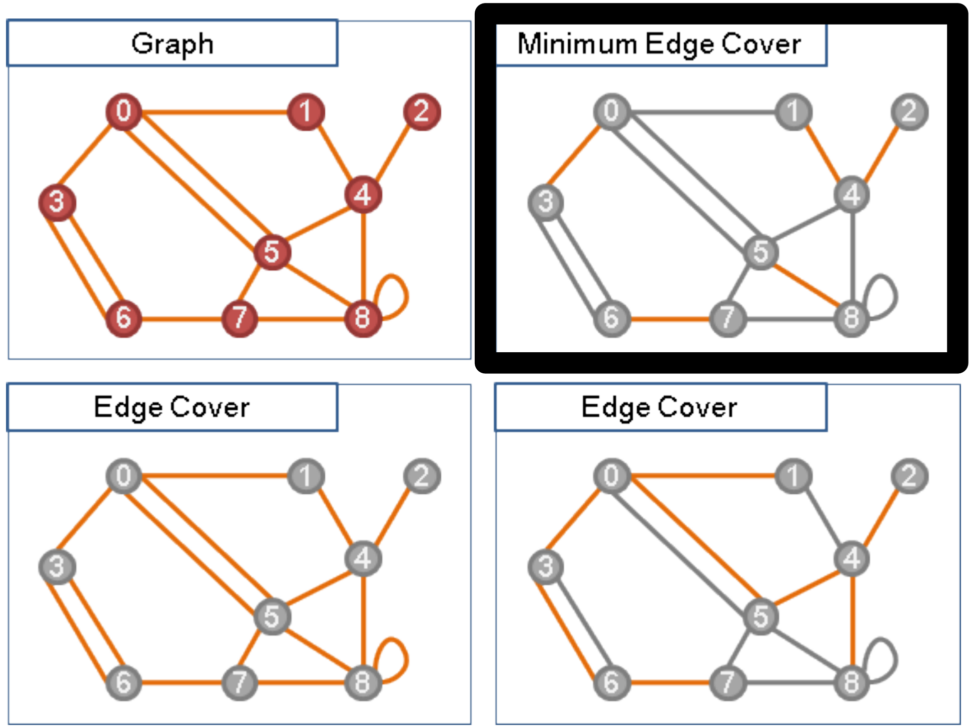
Source: 2017 - Marx - Graphs, hypergraphs, and the complexity of conjunctive database queries

Outline: Cyclic conjunctive queries

- Acyclic conjunctive queries
- Cyclic conjunctive queries
  - 2SAT (a detour)
  - Tree decompositions
  - AGM bound (join processing of cyclic queries)
  - Duality in Linear programming (a quick primer)
  - Worst-case optimal joins
  - Hypertree & other decompositions
  - Optimal joins

Background: MAX independent (vertex) set ≤ MIN edge cover





- Assume graph G is connected. Thus, every vertex has at least one edge (unless just one vertex)
- Suppose S is an independent set and E is an edge cover.
- Then for each vertex $v \in S$ there exists at least one edge $e \in E$ incident with v.
- By definition of independent set no two $u, v \in S$, have a common edge in E.
- Therefore $|S| \leq |E|$

Source of figures: http://www.csie.ntnu.edu.tw/~u91029/Domination.html

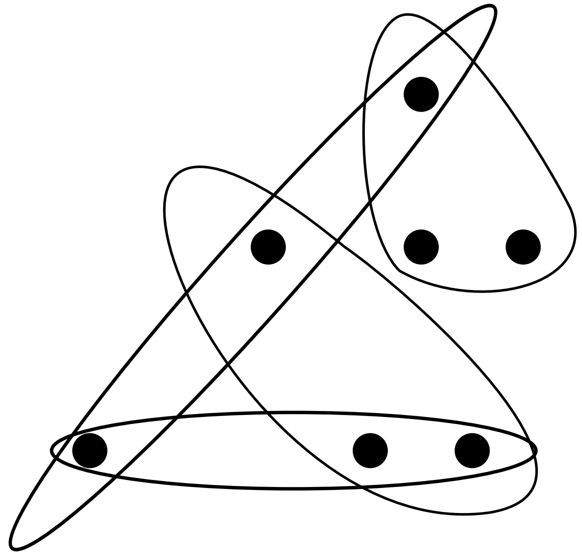
What do we know about bounding the size of the answer?

(...and enumerating all solutions)

Source: Daniel Marx. Graphs, hypergraphs, and the complexity of conjunctive database queries. ICDT 2017: http://edbticdt2017.unive.it/marx-icdt2017-talk.pdf

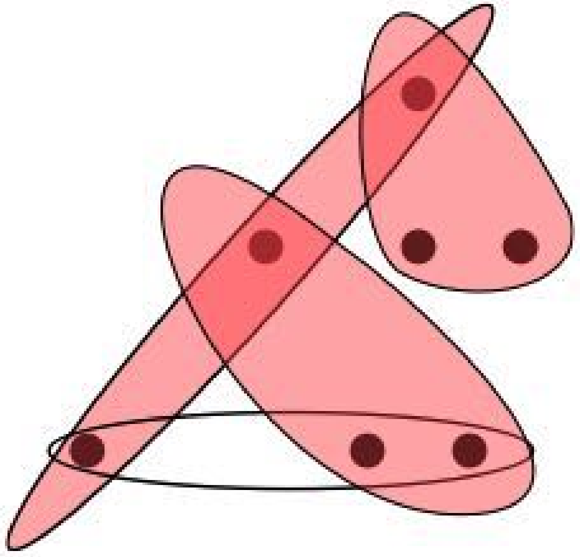
Upper bound

Observation: If the hypergraph has edge cover number ρ and every relation has size at most N, then there are at most N^{ρ} tuples in the answer.



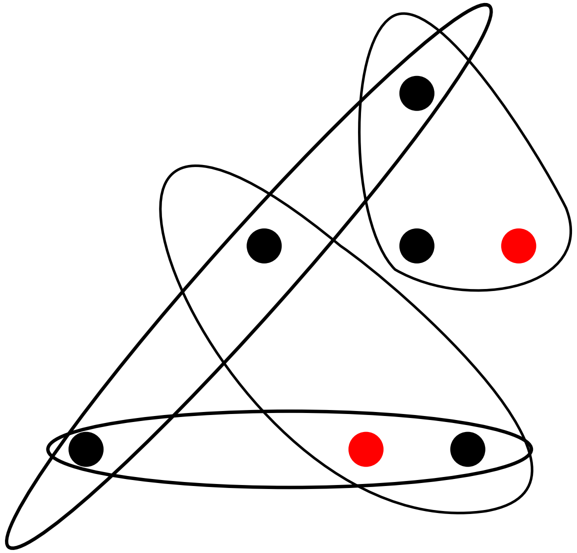
Upper bound

Observation: If the hypergraph has edge cover number ρ and every relation has size at most N, then there are at most N^{ρ} tuples in the answer.



Lower bound

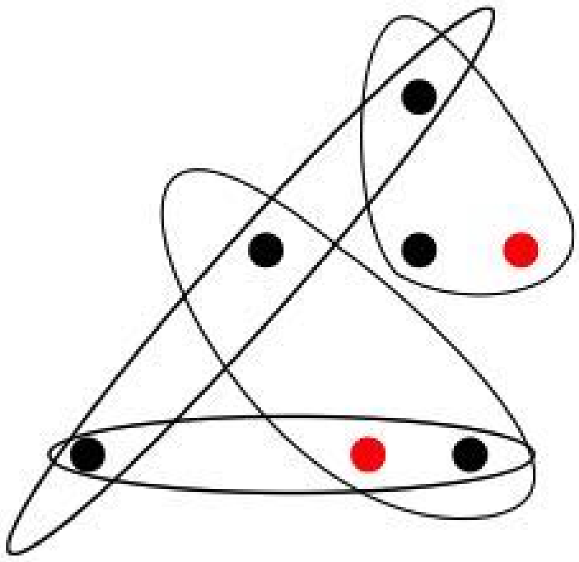
Observation: If the hypergraph has independence number α, then one can construct an instance where every relation has size N at the answer has size N^α.



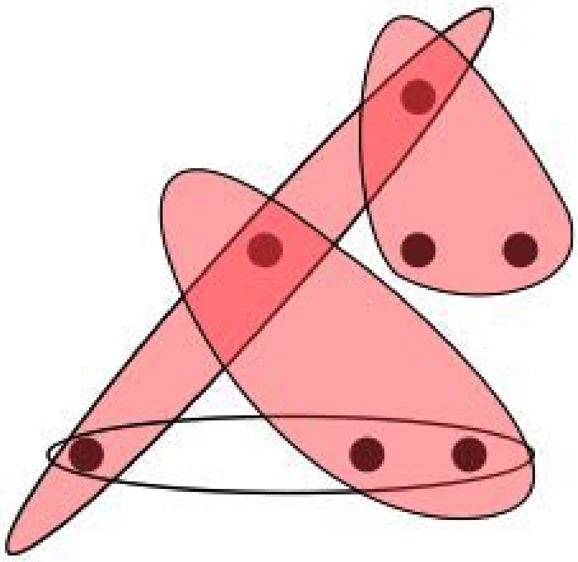
Definition of the relations:

- If variable A is in the independent set, then it can take any value in $[N]$.
- Otherwise it is forced to 1.

Source: Daniel Marx. Graphs, hypergraphs, and the complexity of conjunctive database queries. ICDT 2017: http://edbticdt2017.unive.it/marx-icdt2017-talk.pdf



N^2

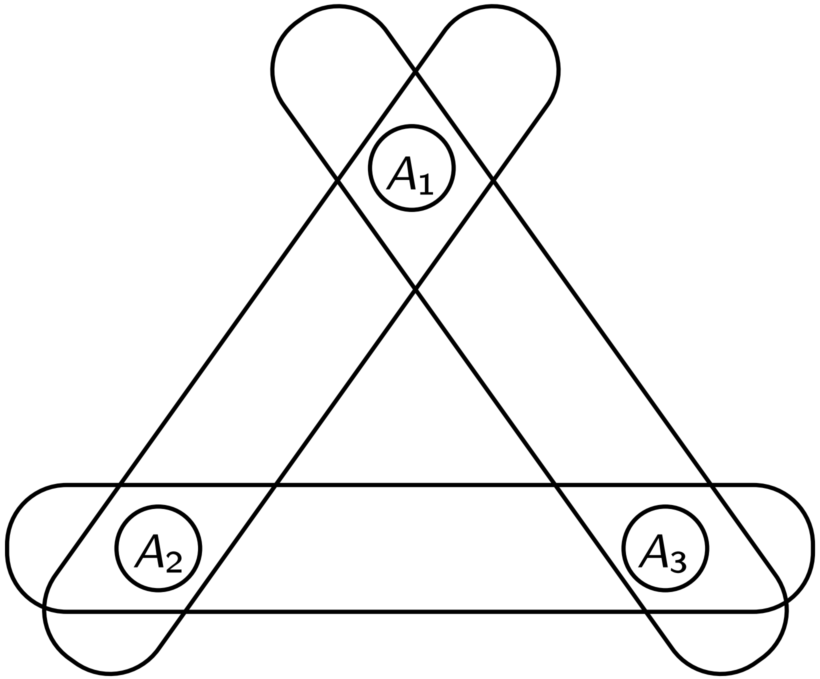


N^3

Which is tight: the upper bound or the lower bound?

Source: Daniel Marx. Graphs, hypergraphs, and the complexity of conjunctive database queries. ICDT 2017: http://edbticdt2017.unive.it/marx-icdt2017-talk.pdf

Example: triangles



Upper bound

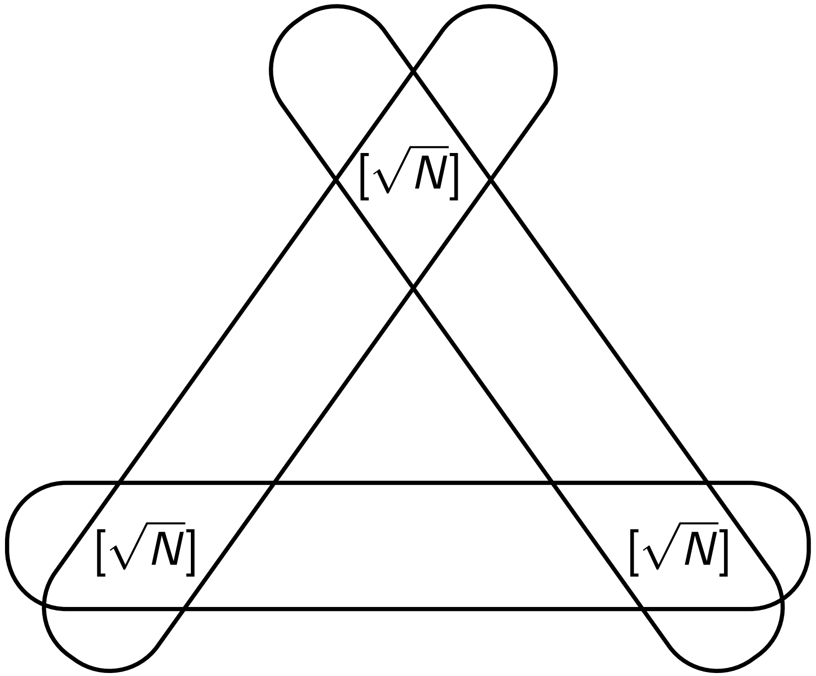
Two kind of values for A_1:

- Light: can be extended to at most $\sqrt{N}$ ways to A_2.
  $\Rightarrow \leq N \cdot \sqrt{N}$ answers with light A_1
- Heavy: can be extended to at least $\sqrt{N}$ ways to A_2.
  $\Rightarrow \leq \sqrt{N}$ heavy values $\Rightarrow \leq \sqrt{N} \cdot N$ answers with heavy A_1

$\Rightarrow$ At most $2 \cdot N^{3/2}$ answers.

Source: Daniel Marx. Graphs, hypergraphs, and the complexity of conjunctive database queries. ICDT 2017: http://edbticdt2017.unive.it/marx-icdt2017-talk.pdf

Example: triangles



Lower bound

Allow every variable to be any value from $[\sqrt{N}] \Rightarrow N^{3/2}$ answers.

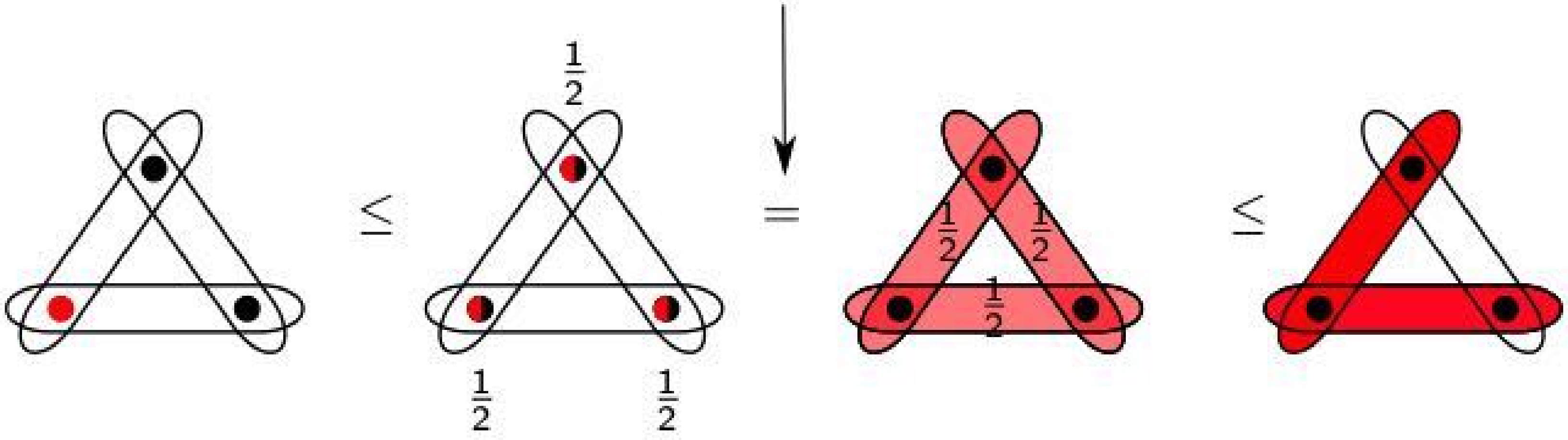
The correct bound $N^{3/2}$ is between
$N^{\alpha} = N^1$ and $N^{\rho} = N^2$.

Source: Daniel Marx. Graphs, hypergraphs, and the complexity of conjunctive database queries. ICDT 2017: http://edbticdt2017.unive.it/marx-icdt2017-talk.pdf

Fractional values

- α: independence number
- α^*: fractional independence number
  (max. weight of vertices s.t. each edge contains weight ≤ 1)
- ρ^*: fractional edge cover number
  (min. weight of edges s.t. each vertex receives weight ≥ 1)
- ρ: edge cover number

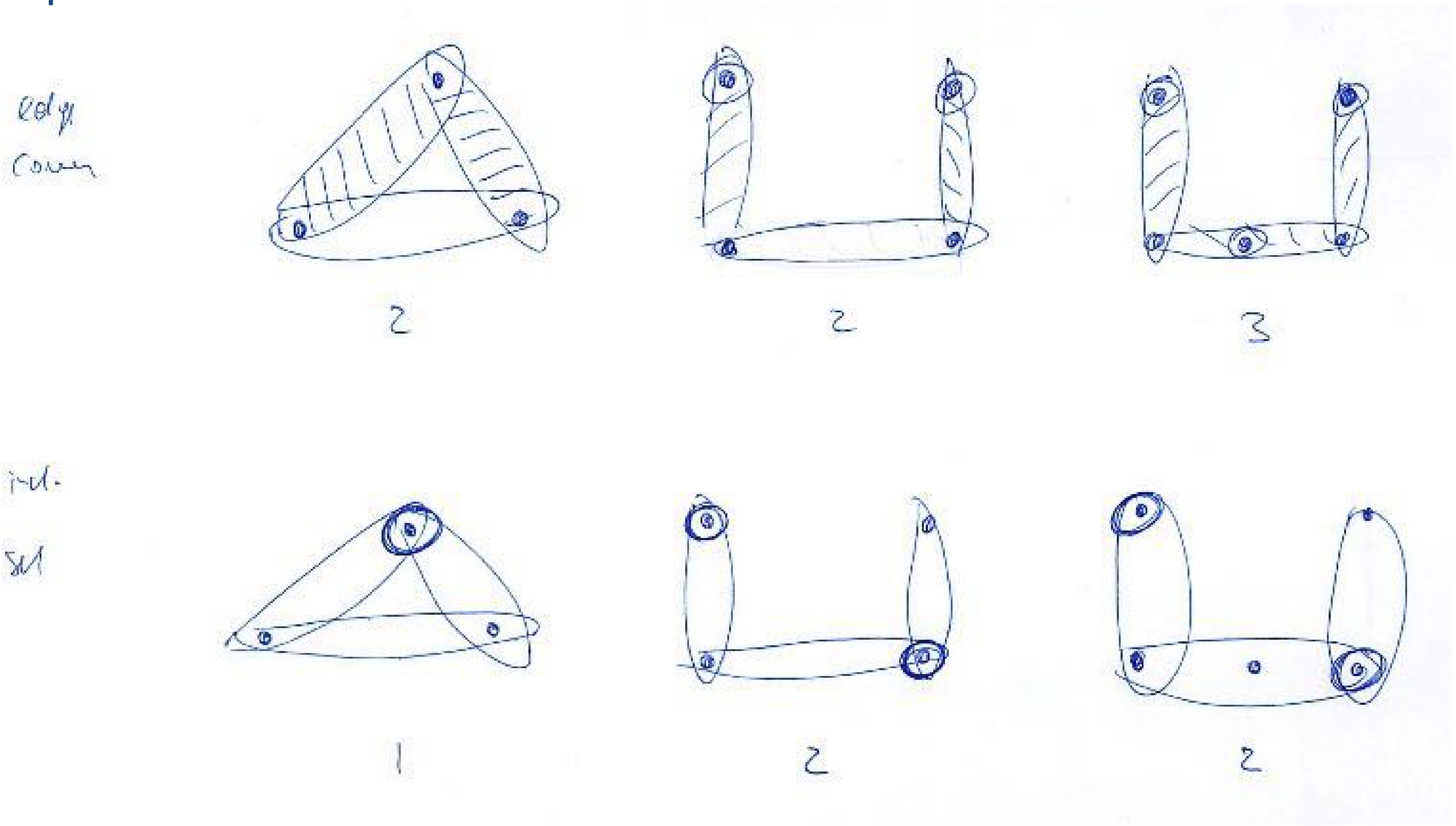
LP duality!



$\alpha = 1$ $\alpha^* = 3/2$ $\rho^* = 3/2$ $\rho = 2$

Source: Daniel Marx. Graphs, hypergraphs, and the complexity of conjunctive database queries. ICDT 2017: http://edbticdt2017.unive.it/marx-icdt2017-talk.pdf

Examples



Outline: Cyclic conjunctive queries

- Acyclic conjunctive queries
- Cyclic conjunctive queries
  - 2SAT (a detour)
  - Tree decompositions
  - AGM bound (join processing of cyclic queries)
  - Duality in Linear programming (a quick primer)
  - Worst-case optimal joins
  - Hypertree & other decompositions
  - Optimal joins

Dual Optimization Problem

- A maximization problem **M** and a minimization problem **N**, defined on the same instances (such as graphs) such that:
  1. for every candidate solution *M* to **M** and every candidate solution *N* to **N**, the value of *M* is less than or equal to the value of *N*
  2. obtaining candidate solutions *M* and *N* that have the same value proves that *M* and *N* are optimal solutions for that instance.

A quick primer on Duality in Linear Programming

$$\begin{aligned} \max \quad x_1 + 6x_2 & \\ x_1 &\leq 200 \\ x_2 &\leq 300 \\ x_1 + x_2 &\leq 400 \\ x_1, x_2 &\geq 0 \end{aligned}$$

optimum solution to be $(x_1, x_2) = (100, 300)$

objective value 1900

Example taken from: Dasgupta, Papadimitriou, Vazirani. Algorithms. 2006. http://algorithmics.lsi.upc.edu/docs/Dasgupta-Papadimitriou-Vazirani.pdf

A quick primer on Duality in Linear Programming

$$\max \; x_1 + 6x_2$$

$$x_1 \le 200 \quad \text{x 1}$$

$$x_2 \le 300 \quad \text{x 6}$$

$$x_1 + x_2 \le 400 \quad \text{x 0}$$

$$x_1, x_2 \ge 0$$

$$x_1 + 6x_2 \le 2000$$

upper bound!

optimum solution to be $(x_1, x_2) = (100, 300)$

objective value 1900

Example taken from: Dasgupta, Papadimitriou, Vazirani. Algorithms. 2006. http://algorithmics.lsi.upc.edu/docs/Dasgupta-Papadimitriou-Vazirani.pdf

A quick primer on Duality in Linear Programming

$$\max \; x_1 + 6x_2$$

$$x_1 \leq 200 \qquad \text{x 0}$$

$$x_2 \leq 300 \qquad \text{x 5}$$

$$x_1 + x_2 \leq 400 \qquad \text{x 1}$$

$$x_1, x_2 \geq 0$$

$$x_1 + 6x_2 \; \leq \; 1900$$

(0,5,1)... certificate of optimiality

optimum solution to be $(x_1, x_2) = (100, 300)$

objective value 1900

Example taken from: Dasgupta, Papadimitriou, Vazirani. Algorithms. 2006. http://algorithmics.lsi.upc.edu/docs/Dasgupta-Papadimitriou-Vazirani.pdf

A quick primer on Duality in Linear Programming

non-negative!

$$\max \; x_1 + 6x_2$$
$$x_1 \leq 200$$
$$x_2 \leq 300$$
$$x_1 + x_2 \leq 400$$
$$x_1, x_2 \geq 0$$

	Multiplier	Inequality
x 0	y_1	$x_1 \leq 200$
x 5	y_2	$x_2 \leq 300$
x 1	y_3	$x_1 + x_2 \leq 400$

$$x_1 + 6x_2 \leq 1900$$

(0,5,1)... certificate of optimiality

Example taken from: Dasgupta, Papadimitriou, Vazirani. Algorithms. 2006. http://algorithmics.lsi.upc.edu/docs/Dasgupta-Papadimitriou-Vazirani.pdf

A quick primer on Duality in Linear Programming

non-negative!

$$\max \; x_1 + 6x_2$$
$$x_1 \le 200$$
$$x_2 \le 300$$
$$x_1 + x_2 \le 400$$
$$x_1, x_2 \ge 0$$

Multiplier	Inequality
y_1	$x_1 \le 200$
y_2	$x_2 \le 300$
y_3	$x_1 + x_2 \le 400$

$$(y_1 + y_3)x_1 + (y_2 + y_3)x_2 \le \boxed{200y_1 + 300y_2 + 400y_3}$$

right side upper bound for $(x_1 + 6x_2)$ if

$$y_1 + y_3 \ge 1$$
$$y_2 + y_3 \ge 6$$

Example taken from: Dasgupta, Papadimitriou, Vazirani. Algorithms. 2006. http://algorithmics.lsi.upc.edu/docs/Dasgupta-Papadimitriou-Vazirani.pdf

A quick primer on Duality in Linear Programming

$$\textbf{Primal}: (x_1, x_2) = (100, 300); \quad \textbf{Dual}: (y_1, y_2, y_3) = (0, 5, 1)$$

$$\begin{aligned} \max\ x_1 + 6x_2 & \\ x_1 &\leq 200 \\ x_2 &\leq 300 \\ x_1 + x_2 &\leq 400 \\ x_1, x_2 &\geq 0 \end{aligned}$$

$$\boxed{\begin{aligned} \min\ 200y_1 + 300y_2 + 400y_3 & \\ y_1 + y_3 &\geq 1 \\ y_2 + y_3 &\geq 6 \\ y_1, y_2, y_3 &\geq 0 \end{aligned}}$$

$$(y_1 + y_3)x_1 + (y_2 + y_3)x_2 \quad \leq \boxed{200y_1 + 300y_2 + 400y_3}$$

right side upper bound for $(x_1 + 6x_2)$ if

$y_1 + y_3 \geq 1$

$y_2 + y_3 \geq 6$

Example taken from: Dasgupta, Papadimitriou, Vazirani. Algorithms. 2006. http://algorithmics.lsi.upc.edu/docs/Dasgupta-Papadimitriou-Vazirani.pdf

A quick primer on Duality in Linear Programming

Figure 7.10 A generic primal LP in matrix-vector form, and its dual.

Primal LP:

$$\begin{aligned} \max\ & \mathbf{c}^T\mathbf{x} \\ \mathbf{A}\mathbf{x} &\le \mathbf{b} \\ \mathbf{x} &\ge 0 \end{aligned}$$

Dual LP:

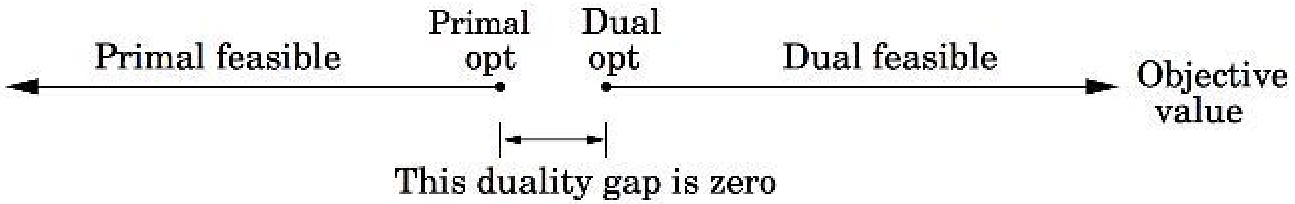
$$\begin{aligned} \min\ & \mathbf{y}^T\mathbf{b} \\ \mathbf{y}^T\mathbf{A} &\ge \mathbf{c}^T \\ \mathbf{y} &\ge 0 \end{aligned}$$

Primal LP:

$$\begin{aligned} \max\ & c_1x_1 + \cdots + c_nx_n \\ a_{i1}x_1 + \cdots + a_{in}x_n &\le b_i \quad \text{for } i \in I \\ a_{i1}x_1 + \cdots + a_{in}x_n &= b_i \quad \text{for } i \in E \\ x_j &\ge 0 \quad \text{for } j \in N \end{aligned}$$

Dual LP:

$$\begin{aligned} \min\ & b_1y_1 + \cdots + b_my_m \\ a_{1j}y_1 + \cdots + a_{mj}y_m &\ge c_j \quad \text{for } j \in N \\ a_{1j}y_1 + \cdots + a_{mj}y_m &= c_j \quad \text{for } j \notin N \\ y_i &\ge 0 \quad \text{for } i \in I \end{aligned}$$



Pointers to related work

- "*AGM bound*": Atserias, Grohe, Marx. *Size bounds and query plans for relational joins*. SIAM J. Comput. 2013. https://doi.org/10.1137/110859440 (also FOCS 2008)
- "*Worst-Case Optimal (WCO) joins*": Ngo, Porat, Re, Rudra. *Worst-case optimal join algorithms*. JACM 2018. https://doi.org/10.1145/3180143 (also PODS 2012)
- "*FAQ paper*": Khamis, Ngo, Rudra. *FAQ: Questions Asked Frequently*. PODS 2016. https://doi.org/10.1145/2902251.2902280 (see also SIGMOD record 2017).
- Khamis, Ngo, Suciu. *What do Shannon-type inequalities, submodular width, and disjunctive Datalog have to do with one another?* PODS 2017. https://doi.org/10.1145/3034786.3056105
- Robertson, Seymour. *Graph minors. II. Algorithmic aspects of tree-width*. Journal of Algorithms. 1986. https://doi.org/10.1016/0196-6774(86)90023-4