Quantifier Alternations and Graph Widths

Michael Lampis
LAMSADE

July 10th 2023 – GWP

What this talk is about

An interesting phenomenon:

- Adding quantifiers costs a level of exponentiation for treewidth.

Two points of view:

Concrete Problems

- SAT with quantifiers
- Σ_2^p, PH, ...

Meta-Theorems

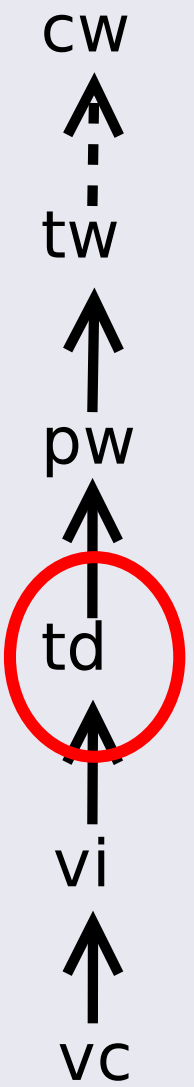
- Treewidth/Pathwidth (Courcelle)
- Vertex Cover, Vertex Integrity,...

Graph widths in this talk

- Tree-depth

$$\mathrm{td}(G) = \min_{S \subseteq V(G)} \left\{ |S| + \max_{S' \in \mathrm{cc}(G-S)} \mathrm{td}(S') \right\}$$

- Select small separator S so that all components have small **tree-depth**
- (Base case: K_1 has tree-depth 1)

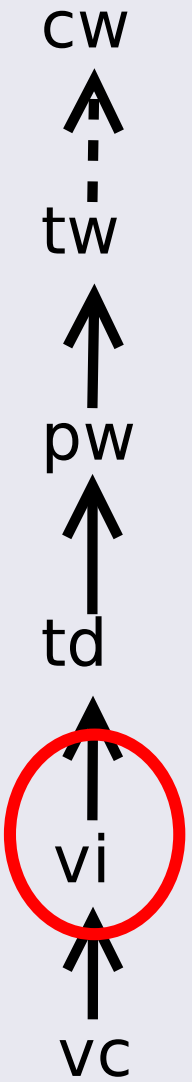


Graph widths in this talk

- Vertex Integrity

$$\mathrm{vi}(G) = \min_{S \subseteq V(G)} \left\{ |S| + \max_{S' \in \mathrm{cc}(G-S)} |S'| \right\}$$

- Select small separator S so that all components have small **size**

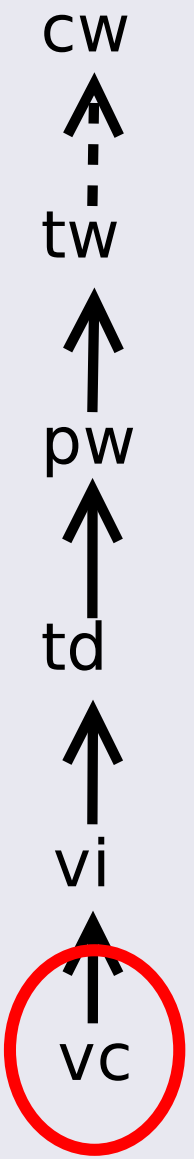


Graph widths in this talk

- Vertex Cover

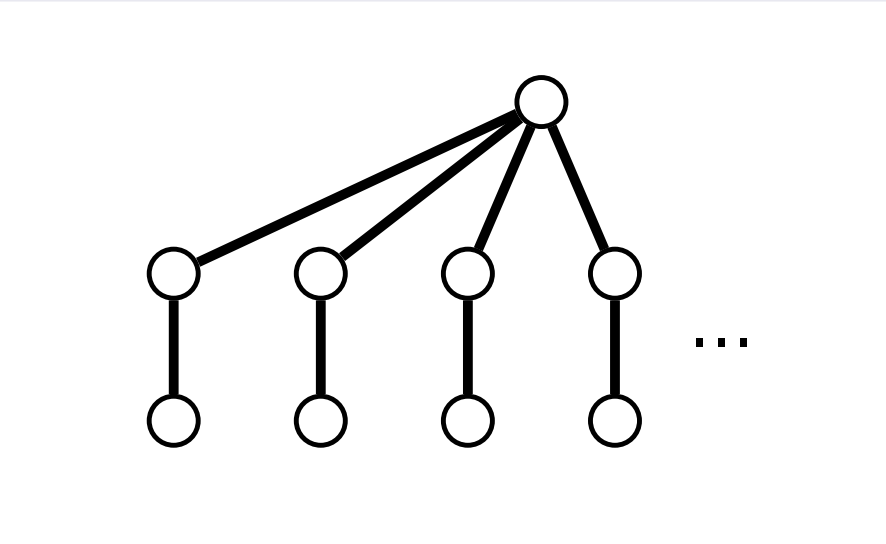
$$\mathrm{vc}(G) = \min_{S \subseteq V(G) \wedge G - S \text{ stable}} \{|S|\}$$

- Select small separator S so that all components **are singletons**.



Graph widths in this talk

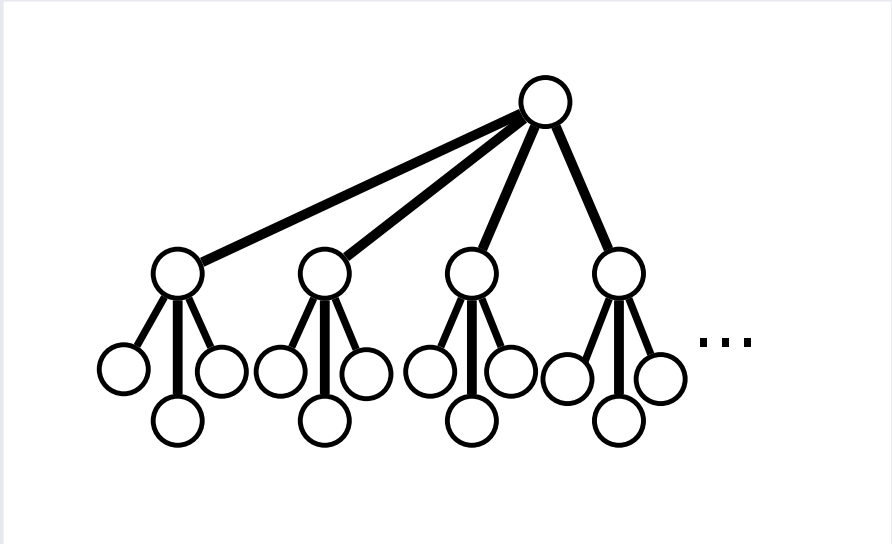
- Inclusions are strict!



- Small vertex integrity, large vertex cover

Graph widths in this talk

- Inclusions are strict!



- Large vertex integrity, small tree-depth

A Textbook Problem

Quantified SAT

$\exists\forall$-SAT definition:
Input: $\exists\mathbf{x}\forall\mathbf{y}\phi(x, y)$

- ϕ in DNF (why not CNF?)

Example:

$$(x_1 \wedge y_1) \vee (x_2 \wedge \neg y_1 \wedge y_2) \vee (\neg x_2 \wedge y_1 \wedge \neg y_2) \vee (\neg y_2)$$

Quantified SAT

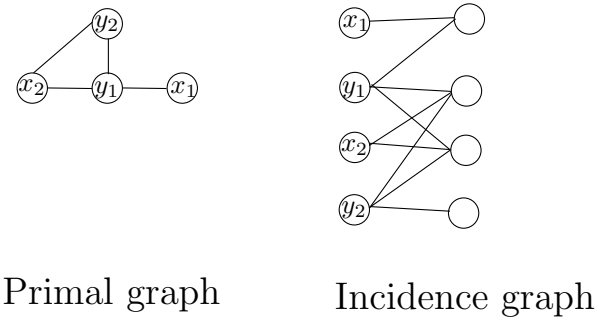
$\exists\forall$-SAT definition:
Input: $\exists \mathbf{x} \forall \mathbf{y} \phi(x, y)$

- ϕ in DNF (why not CNF?)

Example:

$$(x_1 \wedge y_1) \vee (x_2 \wedge \neg y_1 \wedge y_2) \vee (\neg x_2 \wedge y_1 \wedge \neg y_2) \vee (\neg y_2)$$

Graph structure:



Primal graph Incidence graph

(Note: for tw/pw incidence is more general than primal.)

Quantified SAT

$\exists\forall$-SAT definition:
Input: $\exists \mathbf{x} \forall \mathbf{y} \phi(x, y)$

- ϕ in DNF (why not CNF?)

Example:

$$(x_1 \wedge y_1) \vee (x_2 \wedge \neg y_1 \wedge y_2) \vee (\neg x_2 \wedge y_1 \wedge \neg y_2) \vee (\neg y_2)$$

Double-exponential $2^{2^{\mathrm{tw}}} n^{O(1)}$ algorithm

- Two assignments to $\exists$ variables are **equivalent** if:
  - They agree on variables of the bag (2^{tw} classes)
  - They "defeat" the same assignments of the universal player ($2^{2^{\mathrm{tw}}}$ classes)

Quantified SAT reduction

Strategy:

- Reduce 3-SAT on an n-variable formula ψ to $\exists\forall$-SAT on a formula ϕ with $\mathrm{tw}(\phi) = O(\log n)$.

Quantified SAT reduction

Strategy:

- Reduce 3-SAT on an n-variable formula ψ to $\exists\forall$-SAT on a formula ϕ with $\mathrm{tw}(\phi) = O(\log n)$.
- If we could solve $\exists\forall$-SAT in $2^{2^{o(\mathrm{tw})}}$ this would given $2^{o(n)}$ algorithm for 3-SAT.

Quantified SAT reduction

Strategy:

- Reduce 3-SAT on an n-variable formula ψ to $\exists\forall$-SAT on a formula ϕ with $\mathrm{tw}(\phi) = O(\log n)$.

Intuition:

- CNFSAT has an implied quantifier alternation:
  $\exists$ *assignment* $\forall$ *clause* satisfied.
- $\log m$ new universal variables will encode the m clauses in binary.

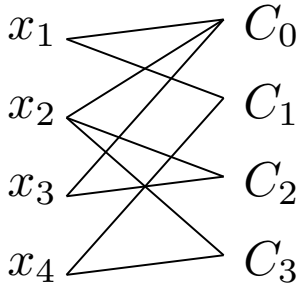
Quantified SAT reduction

Strategy:

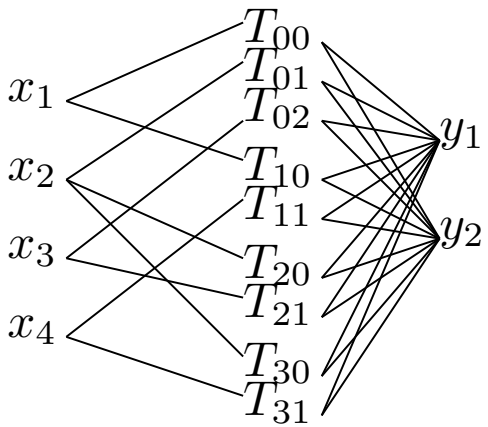
- Reduce 3-SAT on an n-variable formula ψ to $\exists\forall$-SAT on a formula ϕ with $\mathrm{tw}(\phi) = O(\log n)$.

Example:

$$\psi = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_4) \wedge (x_2 \vee x_3) \wedge (\neg x_2 \vee x_4)$$



Incidence graph of ψ



Incidence graph of ϕ

Quantified SAT reduction

Strategy:

- Reduce 3-SAT on an n-variable formula ψ to $\exists\forall$-SAT on a formula ϕ with $\mathrm{tw}(\phi) = O(\log n)$.

Example:

$$\psi = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_4) \wedge (x_2 \vee x_3) \wedge (\neg x_2 \vee x_4)$$

Gives the following DNF terms

C_0	C_1	C_2	C_3
$(x_1 \wedge \neg y_1 \wedge \neg y_2)$	$(x_1 \wedge \neg y_1 \wedge y_2)$	$(x_2 \wedge y_1 \wedge \neg y_2)$	$(\neg x_2 \wedge y_1 \wedge y_2)$
$(x_2 \wedge \neg y_1 \wedge \neg y_2)$	$(x_4 \wedge \neg y_1 \wedge y_2)$	$(x_3 \wedge y_1 \wedge \neg y_2)$	$(x_4 \wedge y_1 \wedge y_2)$
$(\neg x_3 \wedge \neg y_1 \wedge \neg y_2)$			

Quantified SAT reduction

Strategy:

- Reduce 3-SAT on an n-variable formula ψ to $\exists\forall$-SAT on a formula ϕ with $\mathrm{tw}(\phi) = O(\log n)$.

Example:

$$\psi = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_4) \wedge (x_2 \vee x_3) \wedge (\neg x_2 \vee x_4)$$

Gives the following DNF terms

C_0	C_1	C_2	C_3
$(x_1 \wedge \neg y_1 \wedge \neg y_2)$	$(x_1 \wedge \neg y_1 \wedge y_2)$	$(x_2 \wedge y_1 \wedge \neg y_2)$	$(\neg x_2 \wedge y_1 \wedge y_2)$
$(x_2 \wedge \neg y_1 \wedge \neg y_2)$	$(x_4 \wedge \neg y_1 \wedge y_2)$	$(x_3 \wedge y_1 \wedge \neg y_2)$	$(x_4 \wedge y_1 \wedge y_2)$
$(\neg x_3 \wedge \neg y_1 \wedge \neg y_2)$			

Key fact: No two existential variables appear together!
$\Rightarrow$ the $O(\log n)$ variables $\mathbf{y}$ form a vertex cover of the primal graph.

Quantified SAT reduction

Strategy:

- Reduce 3-SAT on an n-variable formula ψ to $\exists\forall$-SAT on a formula ϕ with $\mathrm{tw}(\phi) = O(\log n)$.

Construction:

- Start with a SAT formula $\exists\mathbf{x}\psi$, where $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$, $\psi = C_0 \wedge C_1 \ldots \wedge C_{m-1}$ and m is a power of 2.
- Construct $\exists\mathbf{x}\forall\mathbf{y}\phi$, where $\mathbf{y} = \{y_1, y_2, \ldots, y_{\log m}\}$ are fresh universal variables.
- For each clause C_i we construct $|C_i|$ terms T_{ij} in ϕ. Each T_{ij} has:
  - literal $l_j = (\neg)x_j$ of C_i and
  - a binary combination $\mathcal{B}(i, \mathbf{y})$ of positive and negative appearances of $\mathbf{y}$, unique for i.

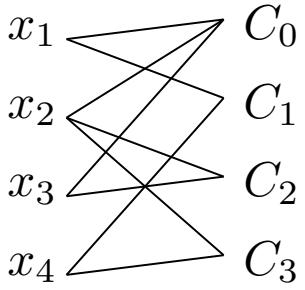
Quantified SAT reduction

Strategy:

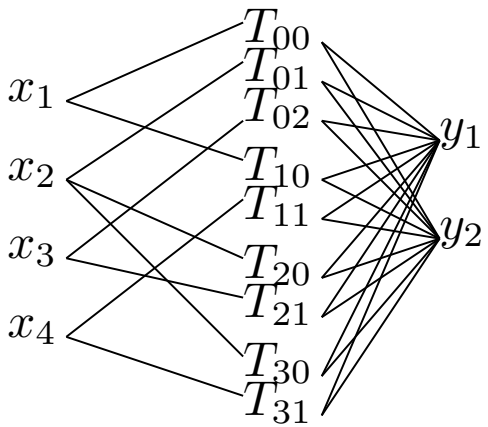
- Reduce 3-SAT on an n-variable formula ψ to $\exists\forall$-SAT on a formula ϕ with $\mathrm{tw}(\phi) = O(\log n)$.

Example:

$$\psi = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_4) \wedge (x_2 \vee x_3) \wedge (\neg x_2 \vee x_4)$$



Incidence graph of ψ



Incidence graph of ϕ

Known results about Quantified SAT

- SAT with k quantifiers complete for Σ_k^p
- Each extra quantifier costs **at most** one level of exponentiation
  - [Chen ECAI 2004]
- Each extra quantifier costs **at least** one level of exponentiation
  - [Pan and Vardi LICS 2006] – odd number of quantifiers
  - [L. and Mitsou IPEC 2017] – two quantifiers
  - [Fichte, Hecher, Pflander LICS 2020] – any number of quantifiers

Known results about Quantified SAT

- SAT with k quantifiers complete for Σ_k^p
- Each extra quantifier costs **at most** one level of exponentiation
  - [Chen ECAI 2004]
- Each extra quantifier costs **at least** one level of exponentiation
  - [Pan and Vardi LICS 2006] – odd number of quantifiers
  - [L. and Mitsou IPEC 2017] – two quantifiers
  - [Fichte, Hecher, Pflander LICS 2020] – any number of quantifiers

Extensions:

- Double-exponential lower bound extends to
  - Bounded term size
  - Bounded variable occurrences
  - $\exists_k\forall$-SAT, $\exists\forall_k$-SAT

Meta-Theorems

Meta-Theorems and Courcelle's Theorem

- Statements of the form:
  "Every problem in family $\mathcal{F}$ is *tractable*"
  - Family $\mathcal{F}$: often "expressible in FO/MSO or other logic"
  - Tractable: often "FPT parameterized by some parameter"

Meta-Theorems and Courcelle's Theorem

- Statements of the form:
  "Every problem in family $\mathcal{F}$ is *tractable*"
  - Family $\mathcal{F}$: often "expressible in FO/MSO or other logic"
  - Tractable: often "FPT parameterized by some parameter"

  Courcelle's famous meta-theorem:

  All problems expressible in MSO logic are FPT parameterized by treewidth.

Meta-Theorems and Courcelle's Theorem

- Statements of the form:
  "Every problem in family $\mathcal{F}$ is *tractable*"
  - Family $\mathcal{F}$: often "expressible in FO/MSO or other logic"
  - Tractable: often "FPT parameterized by some parameter"

  Courcelle's famous meta-theorem:

  > All problems expressible in MSO logic are FPT parameterized by treewidth.

- Notice that since this applies to treewidth, it applies to pathwidth, and tree-depth.

FO and MSO logic reminder

FO logic:

- Two relations: $=$ and $\sim$ (equality, adjacency)
- (Quantified) Variables $x_1, x_2, \ldots$ represent vertices
- Standard boolean connectives $(\vee, \wedge, \neg, \rightarrow)$

Standard Example: 2-Dominating set

$$\exists x_1 \exists x_2 \forall x_3 \, (x_1 = x_3 \vee x_2 = x_3 \vee x_1 \sim x_3 \vee x_2 \sim x_3)$$

FO and MSO logic reminder

FO logic:

- Two relations: $=$ and $\sim$ (equality, adjacency)
- (Quantified) Variables $x_1, x_2, \ldots$ represent vertices
- Standard boolean connectives $(\vee, \wedge, \neg, \rightarrow)$

MSO logic: FO logic plus the following

- $\in$ relation
- (Quantified) **Set** Variables $X_1, X_2, \ldots$ represent sets of vertices

Standard Examples: 3-Coloring, Connectivity

$$\begin{aligned} \exists X_1 \exists X_2 \exists X_3 \quad \Big(\forall x_1 \quad & (x_1 \in X_1 \vee x_1 \in X_2 \vee x_1 \in X_3) \wedge \\ \forall x_2 \quad & (x_1 \sim x_2 \rightarrow (\neg(x_1 \in X_1 \wedge x_2 \in X_1)) \wedge \\ & (\neg(x_1 \in X_2 \wedge x_2 \in X_2)) \wedge \\ & (\neg(x_1 \in X_3 \wedge x_2 \in X_3))) \Big) \end{aligned}$$

FO and MSO logic reminder

FO logic:

- Two relations: $=$ and $\sim$ (equality, adjacency)
- (Quantified) Variables $x_1, x_2, \ldots$ represent vertices
- Standard boolean connectives $(\vee, \wedge, \neg, \rightarrow)$

MSO logic: FO logic plus the following

- $\in$ relation
- (Quantified) **Set** Variables $X_1, X_2, \ldots$ represent sets of vertices

Standard Examples: 3-Coloring, Connectivity
Brute-force Complexity:

- FO: n^q
- MSO: 2^{nq}

Question: For which classes, which f, can we solve FO in time $f(q)n^{O(1)}$?

A Closer Look

- Courcelle: If G has treewidth tw, we can check if it satisfies an MSO property ϕ in time

$$f(\mathrm{tw}, \phi) \cdot |G|$$

A Closer Look

- Courcelle: If G has treewidth tw, we can check if it satisfies an MSO property ϕ in time

$$f(\mathrm{tw}, \phi) \cdot |G|$$

- Problem: f is approximately $2^{2^{2^{\cdot^{\cdot^{\cdot^{2^{\mathrm{tw}}}}}}}}$, where the height of the tower is upper-bounded by the number of **quantifier alternations** in ϕ.

A Closer Look

- Courcelle: If G has treewidth tw, we can check if it satisfies an MSO property ϕ in time

$$f(\mathrm{tw}, \phi) \cdot |G|$$

- Problem: f is approximately $2^{2^{2^{\cdot^{\cdot^{\cdot^{2^{\mathrm{tw}}}}}}}}$, where the height of the tower is upper-bounded by the number of **quantifier alternations** in ϕ.
- **Serious Problem**: This tower of exponentials cannot be avoided[1] even for **FO logic on trees**!
  - "The complexity of first-order and monadic second-order logic **revisited**", [Frick and Grohe, APAL 2004].

[1] Assuming P≠NP or FPT≠W[1].

Treewidth – Pathwidth

Gentle definition of pathwidth k:

- We have k stacks. Initially each contains a vertex. They are arbitrarily connected.
- At each step we add a vertex to the top of a stack. It can be connected to vertices currently on top of a stack.

Gentle definition of pathwidth k:

- We have k stacks. Initially each contains a vertex. They are arbitrarily connected.
- At each step we add a vertex to the top of a stack. It can be connected to vertices currently on top of a stack.

Gentle definition of pathwidth k:

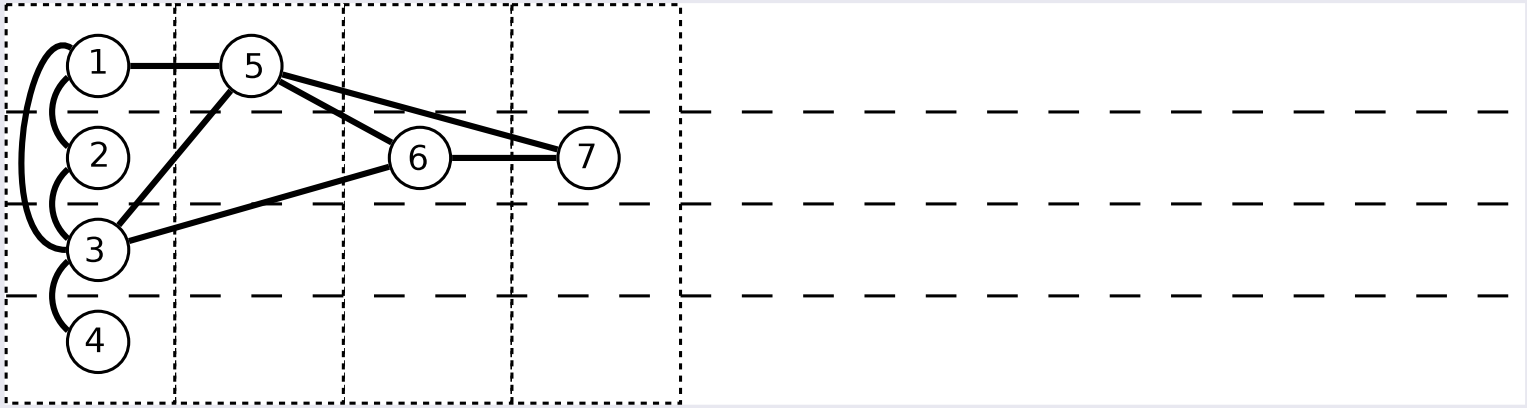
- We have k stacks. Initially each contains a vertex. They are arbitrarily connected.
- At each step we add a vertex to the top of a stack. It can be connected to vertices currently on top of a stack.

Gentle definition of pathwidth k:

- We have k stacks. Initially each contains a vertex. They are arbitrarily connected.
- At each step we add a vertex to the top of a stack. It can be connected to vertices currently on top of a stack.

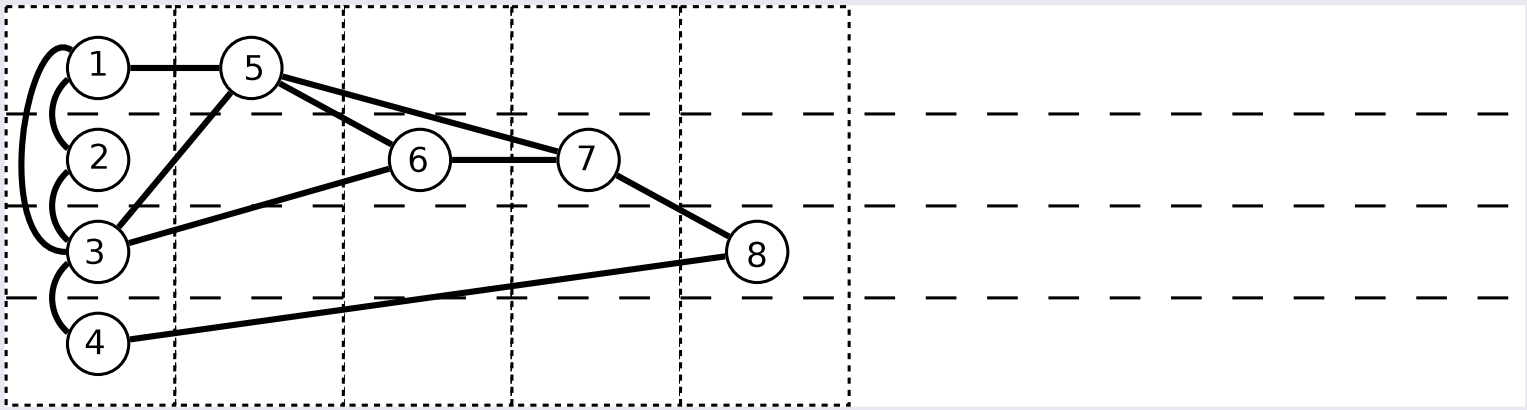
Gentle definition of pathwidth k:

- We have k stacks. Initially each contains a vertex. They are arbitrarily connected.
- At each step we add a vertex to the top of a stack. It can be connected to vertices currently on top of a stack.



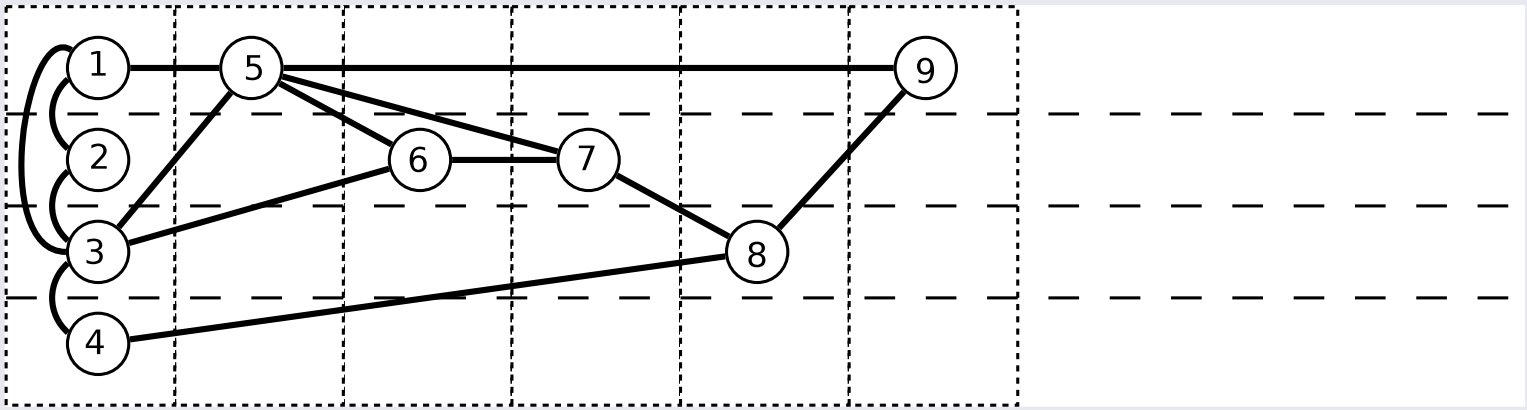
Gentle definition of pathwidth k:

- We have k stacks. Initially each contains a vertex. They are arbitrarily connected.
- At each step we add a vertex to the top of a stack. It can be connected to vertices currently on top of a stack.



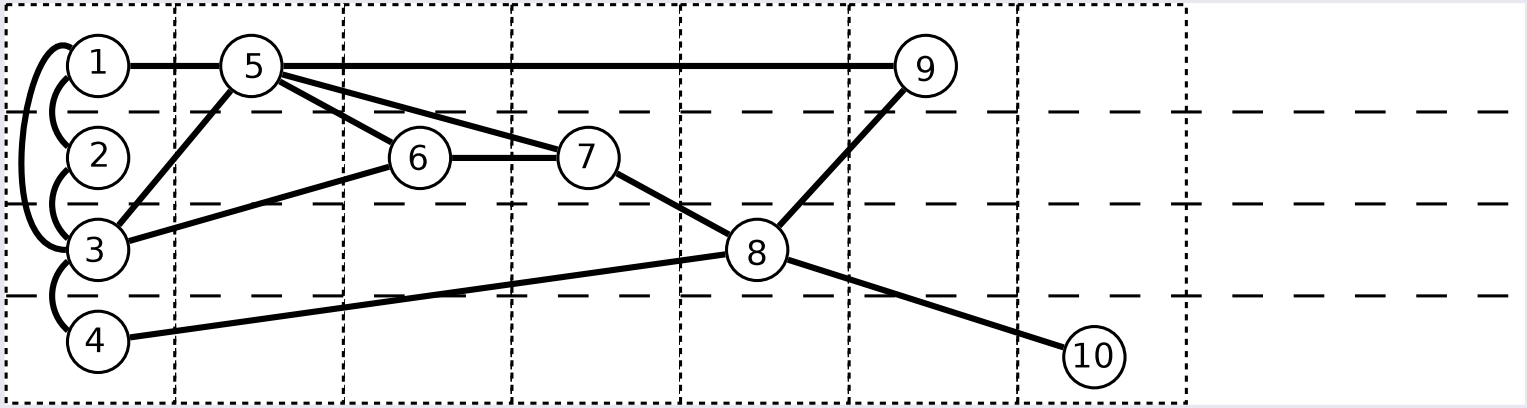
Gentle definition of pathwidth k:

- We have k stacks. Initially each contains a vertex. They are arbitrarily connected.
- At each step we add a vertex to the top of a stack. It can be connected to vertices currently on top of a stack.



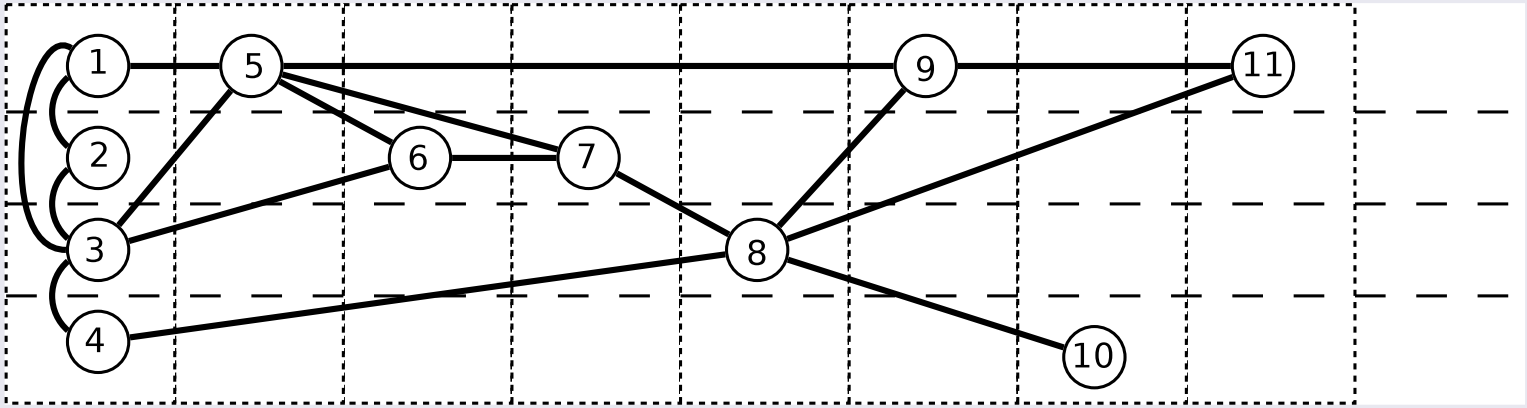
Gentle definition of pathwidth k:

- We have k stacks. Initially each contains a vertex. They are arbitrarily connected.
- At each step we add a vertex to the top of a stack. It can be connected to vertices currently on top of a stack.



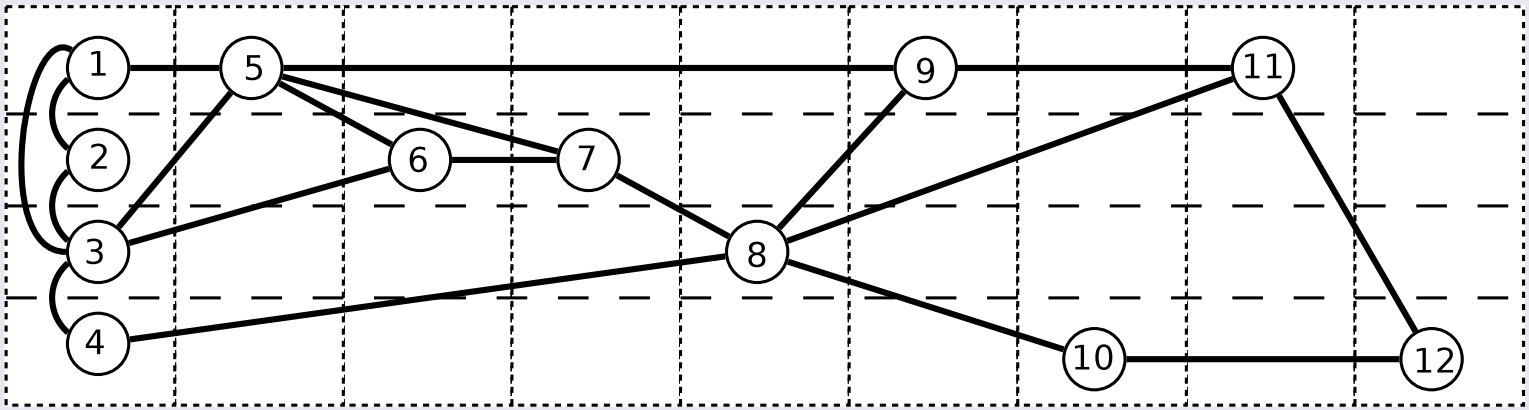
Gentle definition of pathwidth k:

- We have k stacks. Initially each contains a vertex. They are arbitrarily connected.
- At each step we add a vertex to the top of a stack. It can be connected to vertices currently on top of a stack.

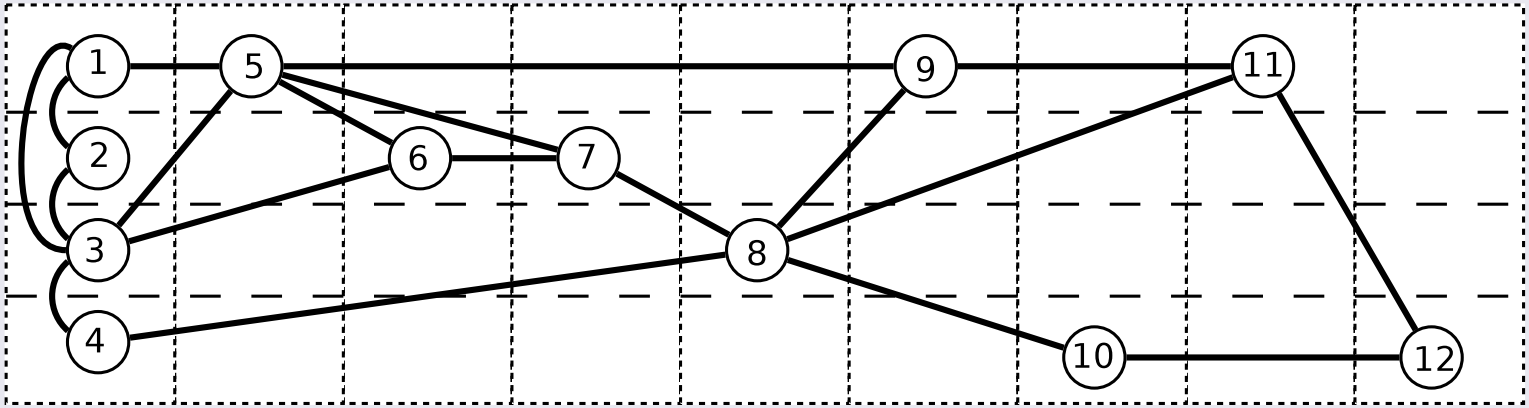


Gentle definition of pathwidth k:

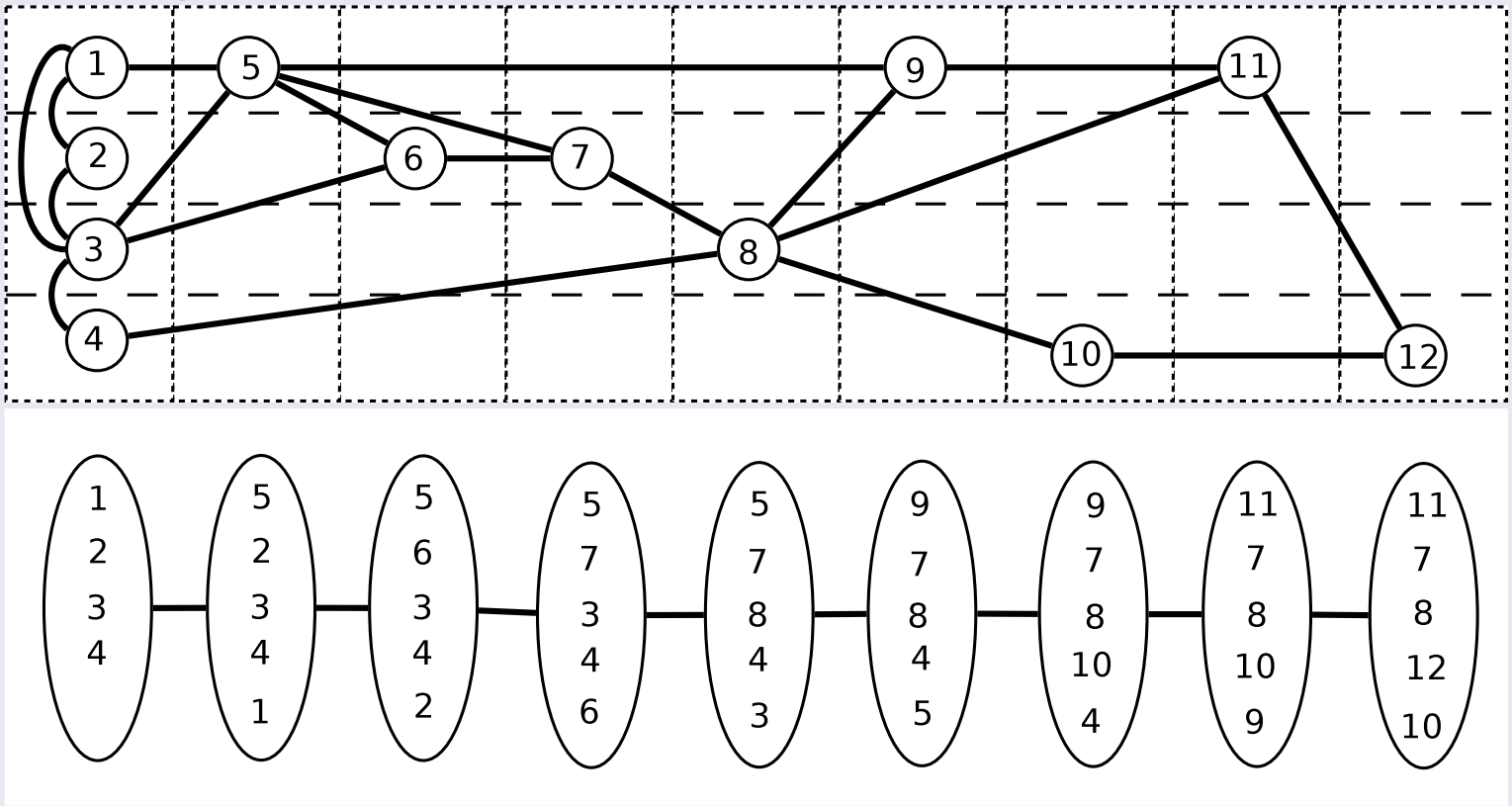
- We have k stacks. Initially each contains a vertex. They are arbitrarily connected.
- At each step we add a vertex to the top of a stack. It can be connected to vertices currently on top of a stack.



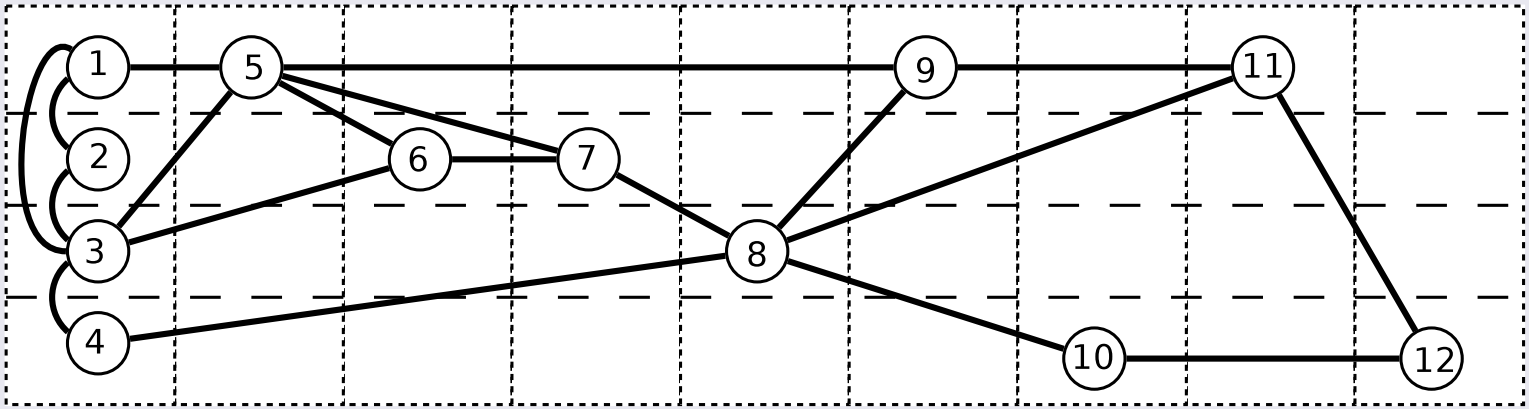
Note that this is equivalent to the standard definition of path decompositions.



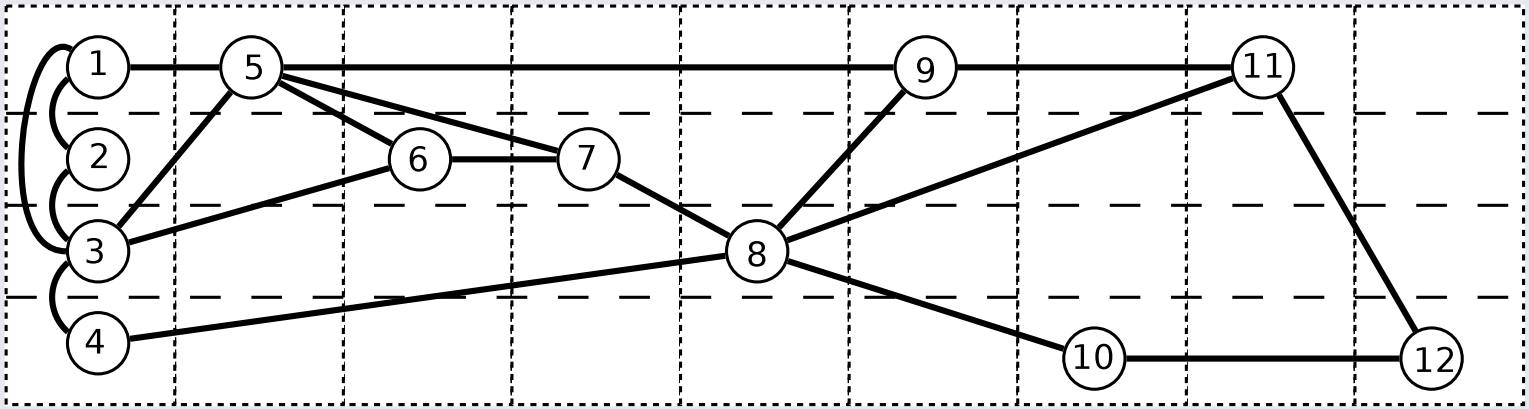
Note that this is equivalent to the standard definition of path decompositions.



Courcelle's Theorem for Pathwidth:

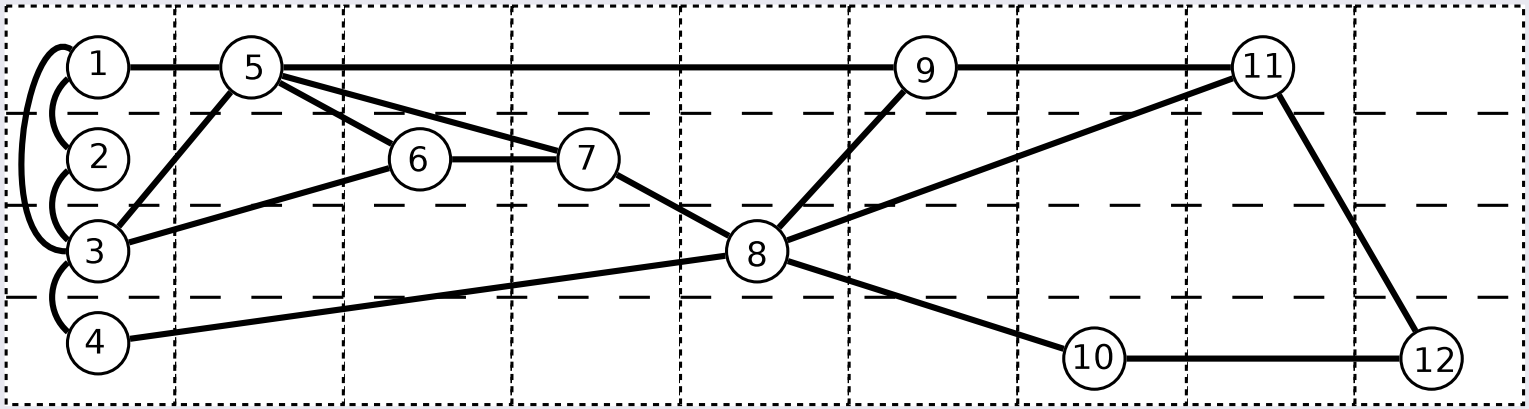


Courcelle's Theorem for Pathwidth:



Can be expressed as a **string** over an alphabet of size $p \cdot 2^p$.

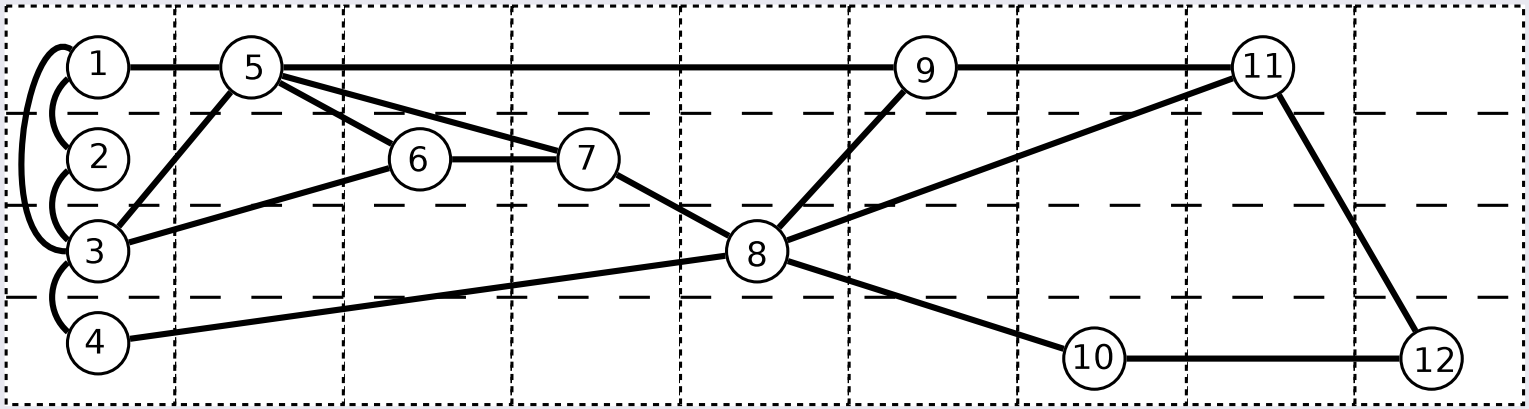
Courcelle's Theorem for Pathwidth:



Can be expressed as a **string** over an alphabet of size $p \cdot 2^p$.

Vertex	5	6	7	8	9
Character	(1, 1010)	(2, 1010)	(2, 1100)	(3, 0101)	(1, 1010)

Courcelle's Theorem for Pathwidth:



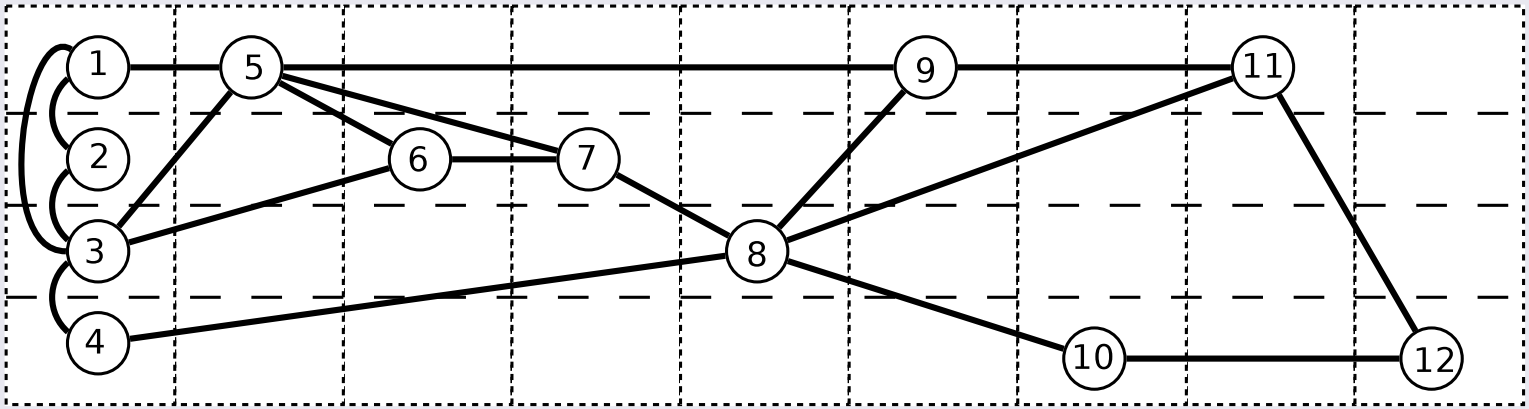
Can be expressed as a **string** over an alphabet of size $p \cdot 2^p$.

Vertex	5	6	7	8	9
Character	(1, 1010)	(2, 1010)	(2, 1100)	(3, 0101)	(1, 1010)

MSO logic on **Strings**:

- $\exists x$ means there exists a character x...
- Vocabulary: $x \preceq y$ (x is to the left of y), unary predicates for Σ.

Courcelle's Theorem for Pathwidth:



Can be expressed as a **string** over an alphabet of size $p \cdot 2^p$.

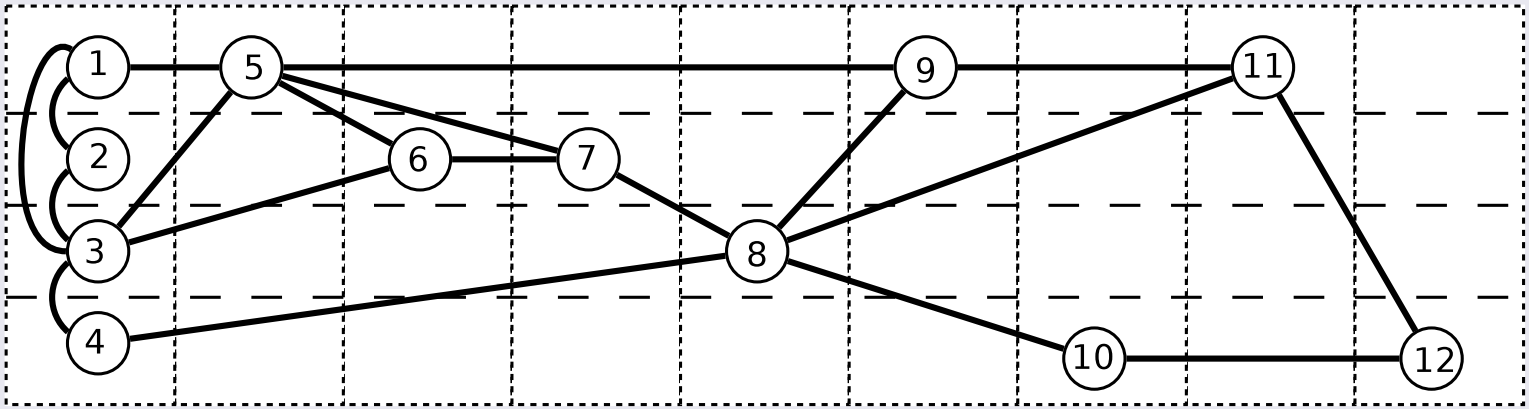
Vertex	5	6	7	8	9
Character	(1, 1010)	(2, 1010)	(2, 1100)	(3, 0101)	(1, 1010)

Idea: Adjacency in G can be expressed in MSO logic in the string!

$x \sim y$ iff

- $x \preceq y$.
- $\nexists z$ s.t. $x \preceq z \preceq y$ and z is on same stack as x.
- Check (p bits of) symbol of y and stack number of x.

Courcelle's Theorem for Pathwidth:



Can be expressed as a **string** over an alphabet of size $p \cdot 2^p$.

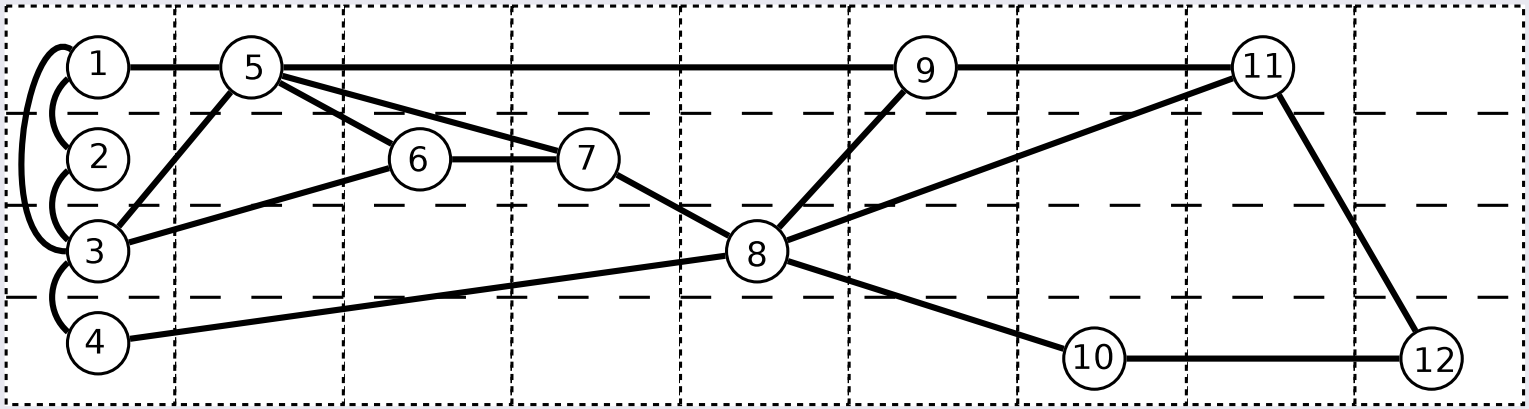
Vertex	5	6	7	8	9
Character	(1, 1010)	(2, 1010)	(2, 1100)	(3, 0101)	(1, 1010)

Idea: Translate MSO question on graph to MSO question on string.

Theorem: MSO logic on Strings $\equiv$ **regular** languages [Büchi 1960].

Consequence: Linear-time algorithm for MSO logic on bounded pathwidth graphs.

Courcelle's Theorem for Pathwidth:



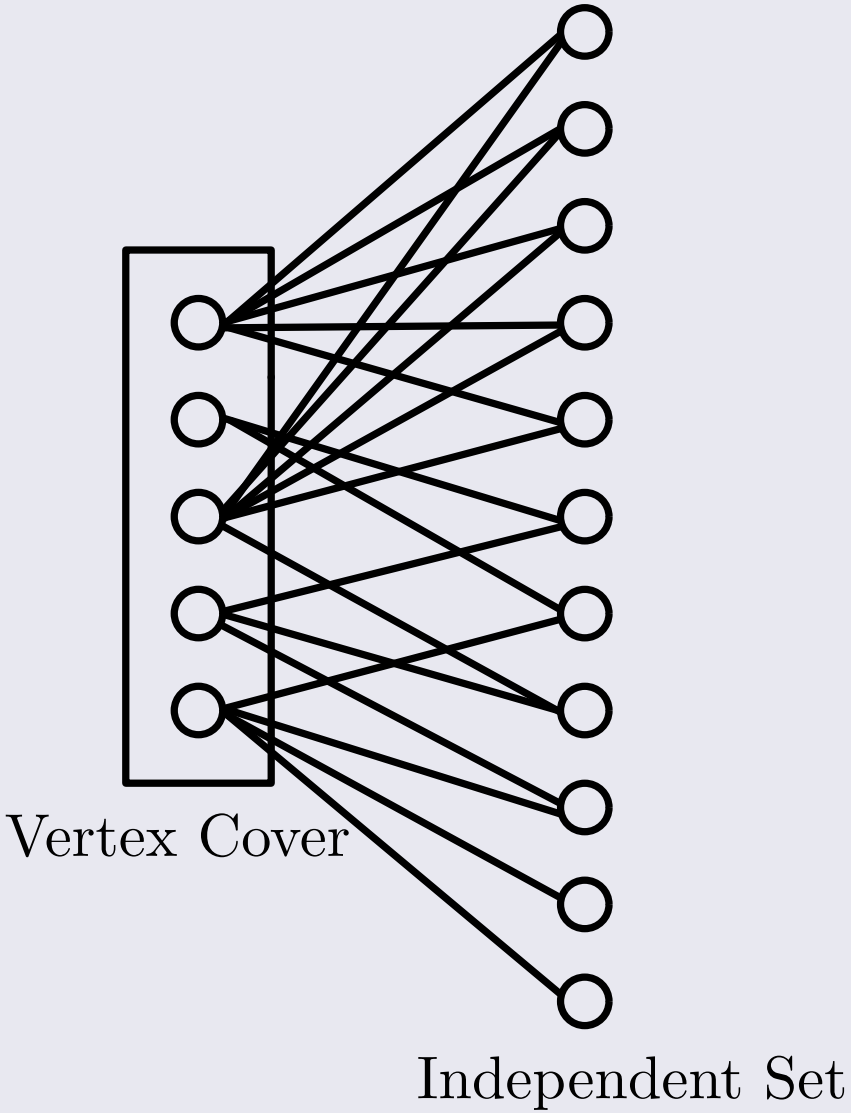
Can be expressed as a **string** over an alphabet of size $p \cdot 2^p$.

Vertex	5	6	7	8	9
Character	(1, 1010)	(2, 1010)	(2, 1100)	(3, 0101)	(1, 1010)

Intuition:

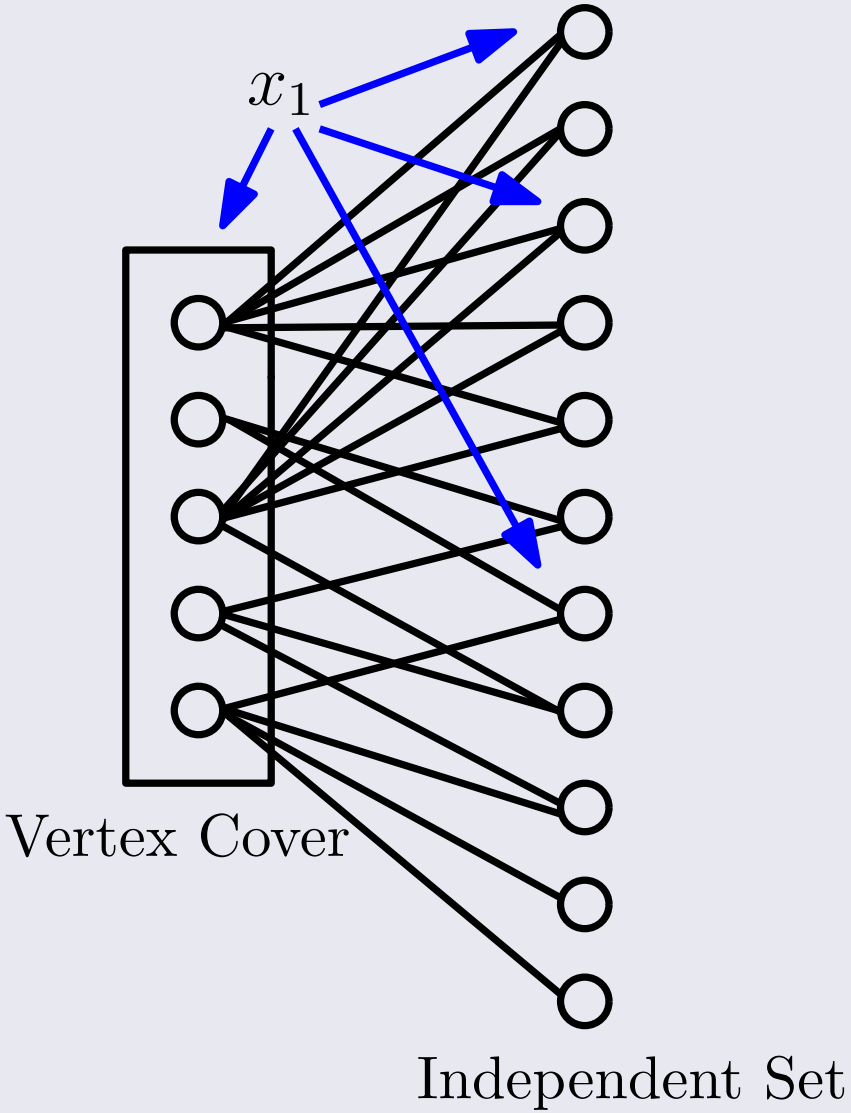
- Quantifier Alternations force us to make the automaton deterministic.
- **Consequence**: each alternation gives a level of exponentiation.

Vertex Cover Meta-Theorem



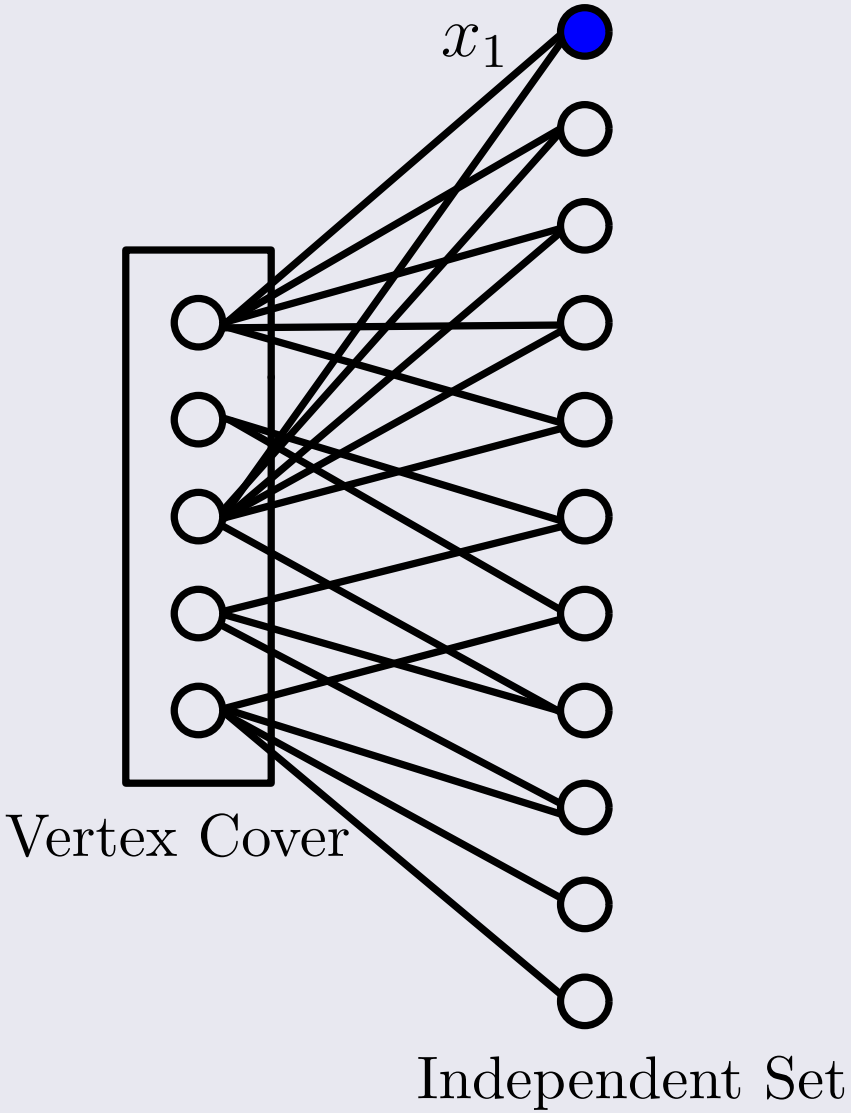
- Given a graph with vertex cover $\mathrm{vc} = 5$
- we want to check an FO property ϕ with $q = 3$ variables.

Vertex Cover Meta-Theorem



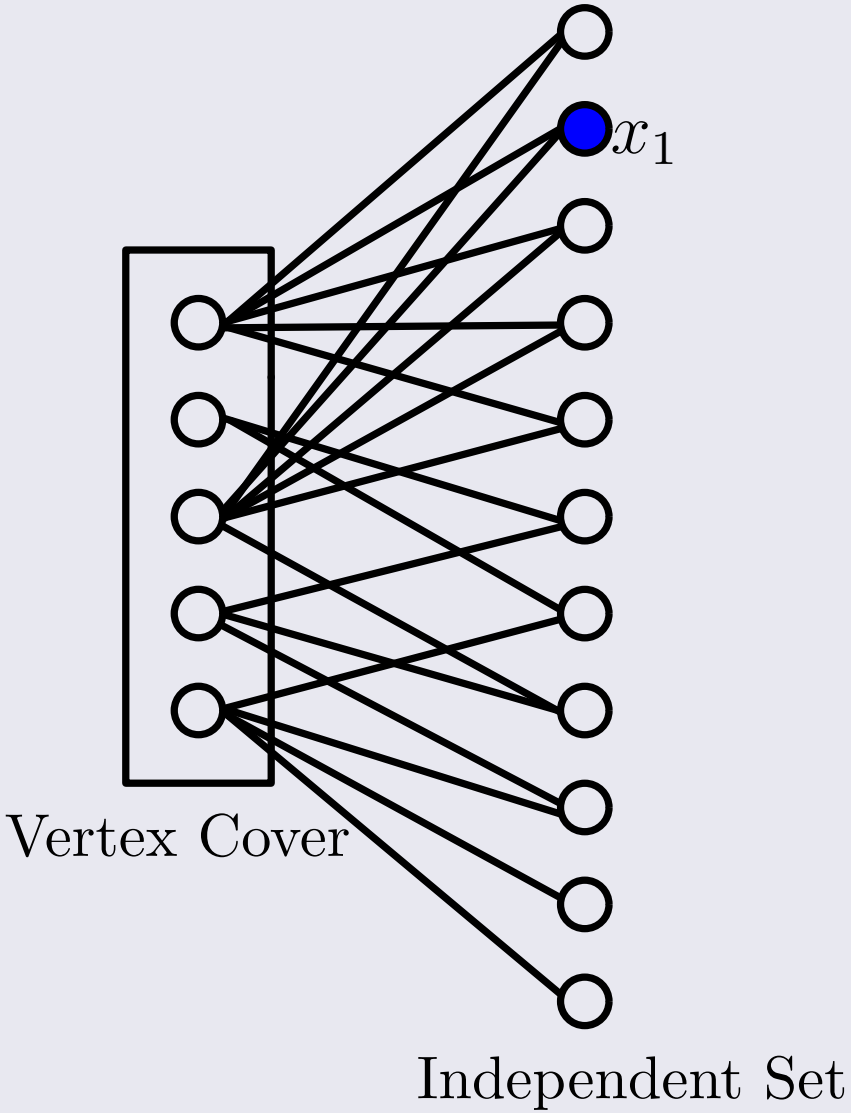
- Sentence has form $\exists x_1 \psi(x_1)$
- We must "place" x_1 somewhere in the graph
- If we try all cases we get n^q running time.

Vertex Cover Meta-Theorem



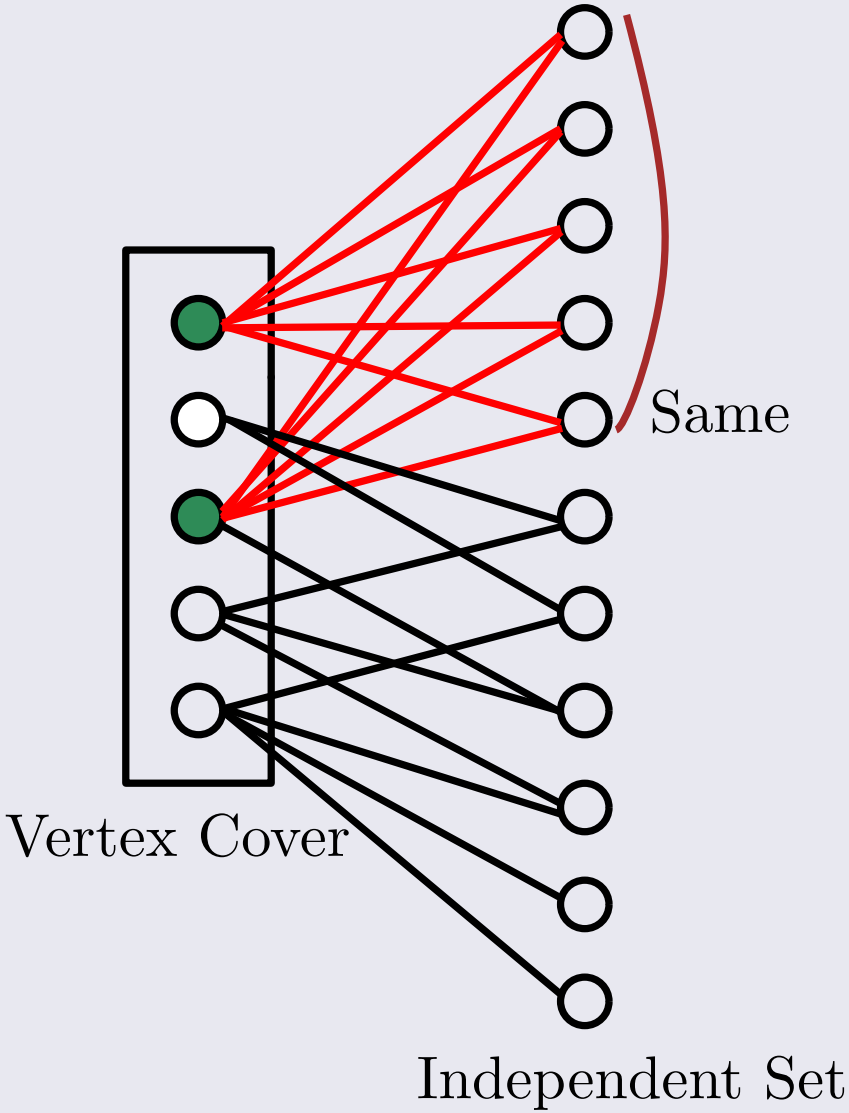
- Sentence has form $\exists x_1 \psi(x_1)$
- We must "place" x_1 somewhere in the graph
- If we try all cases we get n^q running time.

Vertex Cover Meta-Theorem



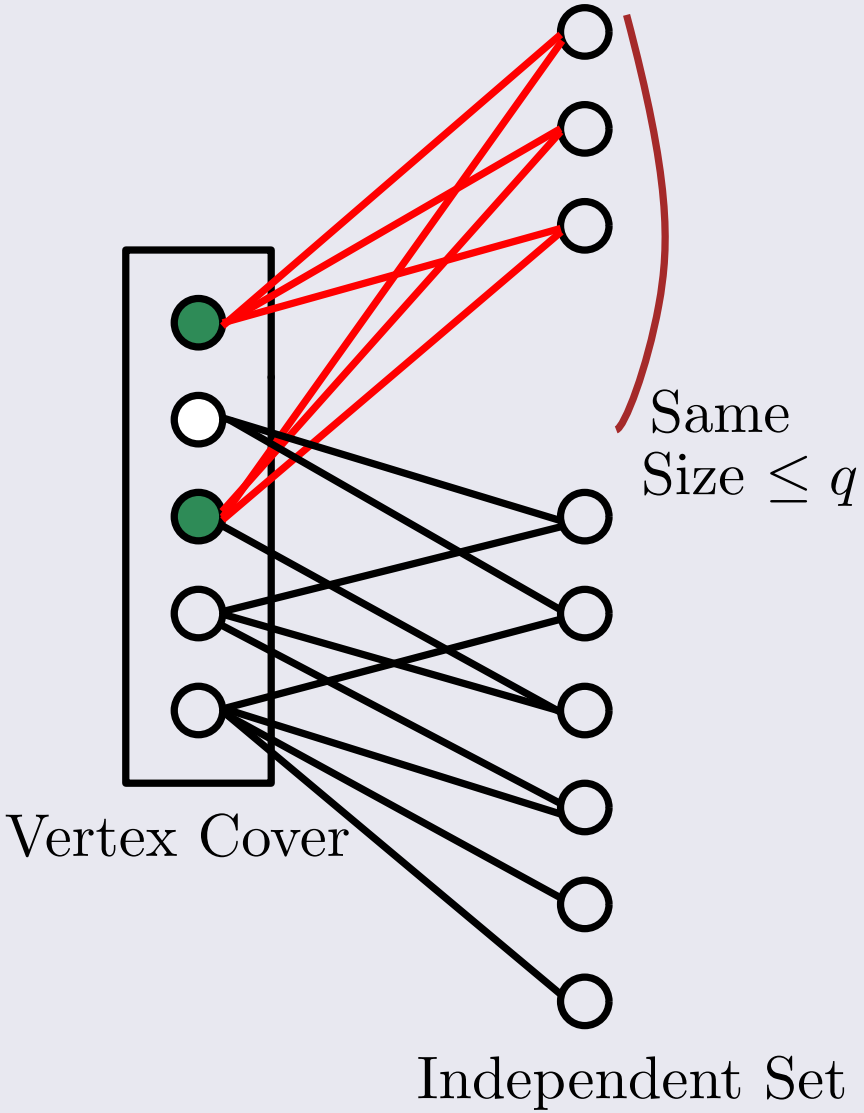
- Sentence has form $\exists x_1 \psi(x_1)$
- We must "place" x_1 somewhere in the graph
- If we try all cases we get n^q running time.

Vertex Cover Meta-Theorem



- We observe that some vertices of the independent set have the same neighbors.
- These vertices should be equivalent.

Vertex Cover Meta-Theorem



- We observe that some vertices of the independent set have the same neighbors.
- These vertices should be equivalent.
- Key idea: if a group has $> q$ vertices, we can simply remove one!

Vertex Cover and FO logic

Summary of previous argument:

- Partition graph into $2^{\mathrm{vc}} + \mathrm{vc}$ sets of equivalent vertices.
- If a set has $> q$ vertices, delete one, repeat.
- If not, $|V(G)| \leq q2^{O(\mathrm{vc})}$.
- Trivial algorithm now runs in $2^{O(\mathrm{vc} \cdot q)} q^q$.

Key idea:

FO logic with q quantifiers can distinguish sets of size at most q.

Vertex Cover and FO logic

Summary of previous argument:

- Partition graph into $2^{\mathrm{vc}} + \mathrm{vc}$ sets of equivalent vertices.
- If a set has $> q$ vertices, delete one, repeat.
- If not, $|V(G)| \leq q2^{O(\mathrm{vc})}$.
- Trivial algorithm now runs in $2^{O(\mathrm{vc} \cdot q)} q^q$.

Key idea:

FO logic with q quantifiers can distinguish sets of size at most q.

What about MSO?

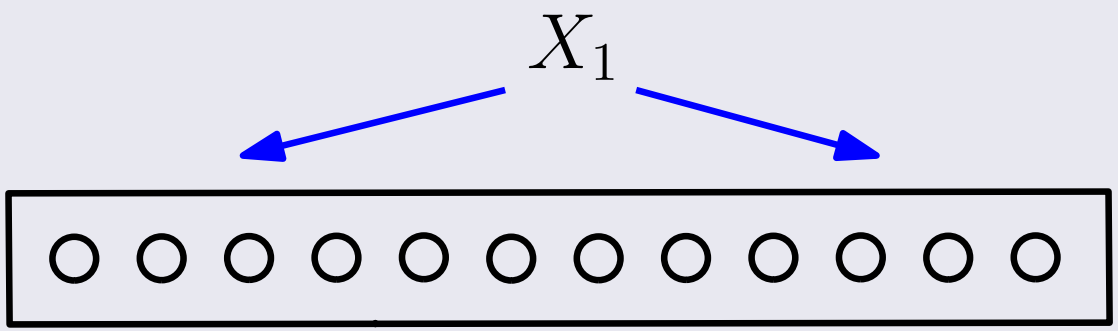
MSO and Vertex Cover

Key idea:

MSO logic with q quantifiers can distinguish sets of size at most 2^q.

Proof by induction:

- Want to prove, if set has size $> 2^q$, can delete one vertex.
- Suppose OK for up to $q - 1$ quantifiers.
- Want to check if $\exists X_1 \psi(X_1)$, where ψ has $q - 1$ quantifiers.



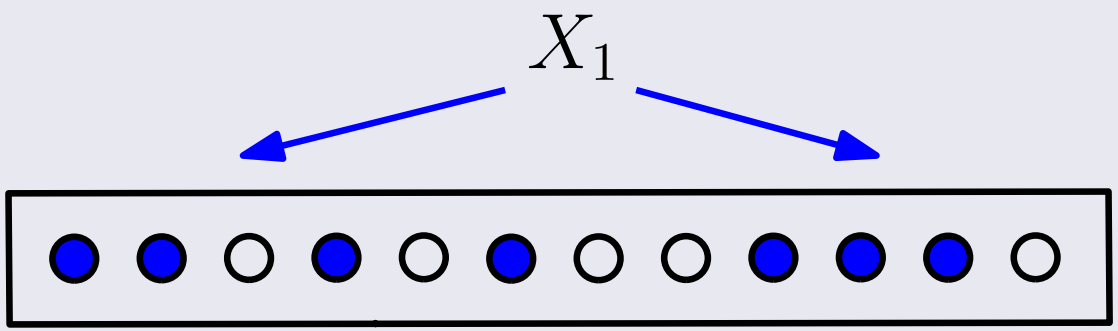
MSO and Vertex Cover

Key idea:

MSO logic with q quantifiers can distinguish sets of size at most 2^q.

Proof by induction:

- Want to prove, if set has size $> 2^q$, can delete one vertex.
- Suppose OK for up to $q - 1$ quantifiers.
- Want to check if $\exists X_1 \psi(X_1)$, where ψ has $q - 1$ quantifiers.



- For any choice of X_1 a set of 2^{q-1} identical vertices remains.
- Apply inductive hypothesis.

MSO and Vertex Cover

Key idea:

MSO logic with q quantifiers can distinguish sets of size at most 2^q.

- Graph has 2^{vc} sets of equivalent vertices.
- While one has size $> 2^q$, delete a vertex.
- Otherwise, $|V(G)| \leq 2^{\mathrm{vc}+q}$.
- Brute force:

$$2^{nq} \leq 2^{2^{\mathrm{vc}+q}q} = 2^{2^{O(\mathrm{vc}+q)}}$$

Back to Quantified SAT

QBF parameterized by vertex cover

QBF: $\exists x_1 \forall x_2 \exists x_3 \ldots Q x_n \phi(x_1, \ldots, x_n)$

Thm: QBF can be solved in time $2^{3^{\mathrm{vc}}} n^{O(1)}$ [L. and Mitsou IPEC 2017]

QBF parameterized by vertex cover

QBF: $\exists x_1 \forall x_2 \exists x_3 \ldots Q x_n \phi(x_1, \ldots, x_n)$

Thm: QBF can be solved in time $2^{3^{\mathrm{vc}}} n^{O(1)}$ [L. and Mitsou IPEC 2017]

(Referring to **primal** vertex cover. Incidence vc is easy...)

QBF parameterized by vertex cover

QBF: $\exists x_1 \forall x_2 \exists x_3 \ldots Q x_n \phi(x_1, \ldots, x_n)$

Thm: QBF can be solved in time $2^{3^{\mathrm{vc}}} n^{O(1)}$ [L. and Mitsou IPEC 2017]

Algorithm:

- If x_1 only appears positive (or negative) easy to set.
- If a clause C_1 is contained in a clause C_2, remove C_2.
- Otherwise, branch on both values of x_1.

QBF parameterized by vertex cover

QBF: $\exists x_1 \forall x_2 \exists x_3 \ldots Q x_n \phi(x_1, \ldots, x_n)$

Thm: QBF can be solved in time $2^{3^{\text{vc}}} n^{O(1)}$ [L. and Mitsou IPEC 2017]

Algorithm:

- If x_1 only appears positive (or negative) easy to set.
- If a clause C_1 is contained in a clause C_2, remove C_2.
- Otherwise, branch on both values of x_1.

Proof of running time:

- If x_1 part of vertex cover, great!
- If not, we have a clause $(x_1 \vee C_1)$ and a clause $(\neg x_1 \vee C_2)$
- $\rightarrow$ new instances have a new clause C_1 or C_2 **contained** in the vertex cover.
- Cannot construct more than 3^{vc} such clauses!

QBF parameterized by vertex cover

QBF: $\exists x_1 \forall x_2 \exists x_3 \dots Q x_n \phi(x_1, \dots, x_n)$

Thm: QBF can be solved in time $2^{3^{\mathrm{vc}}} n^{O(1)}$ [L. and Mitsou IPEC 2017]

Algorithm:

- If x_1 only appears positive (or negative) easy to set.
- If a clause C_1 is contained in a clause C_2, remove C_2.
- Otherwise, branch on both values of x_1.

Success!

More Meta-Theorems

Meta-Theorems to SAT algorithms?

Meta-theorem for vertex cover $\rightarrow$ QBF/vc worked well!

Other elementary meta-theorems to try

- Vertex Integrity [L. and Mitsou, ISAAC 2021]
- Tree-depth [Gajarsky and Hlineny, MFCS 2012, LMCS 2015]
- Pathwidth

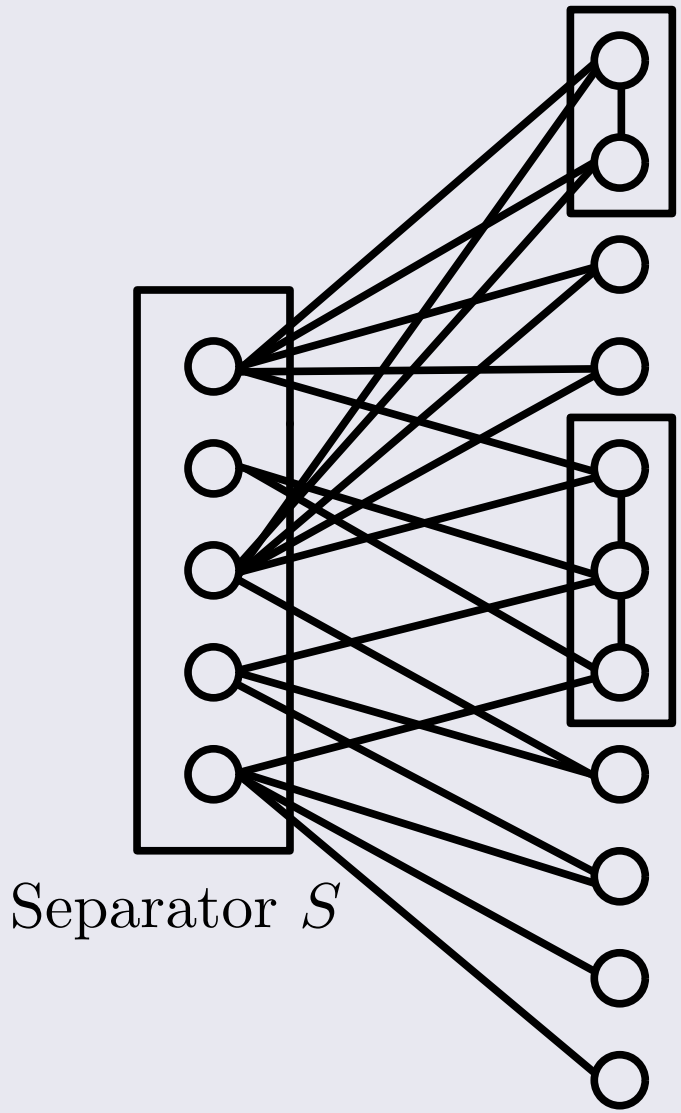
Meta-Theorems to SAT algorithms?

Meta-theorem for vertex cover $\rightarrow$ QBF/vc worked well!

Other elementary meta-theorems to try

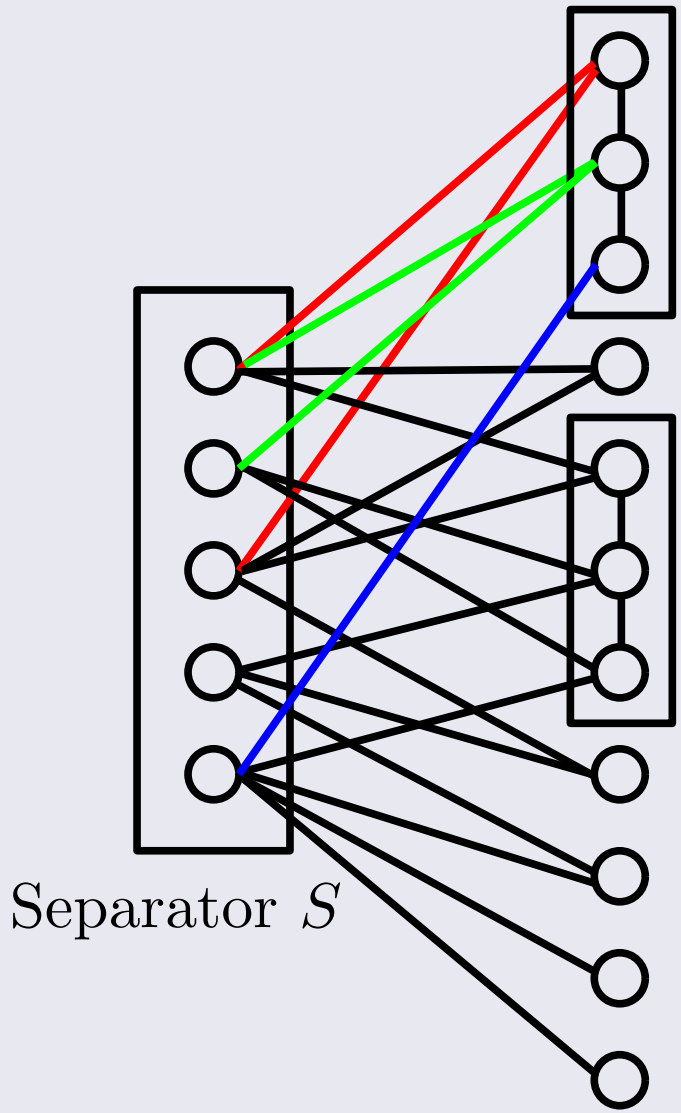
- Vertex Integrity [L. and Mitsou, ISAAC 2021]
- Tree-depth [Gajarsky and Hlineny, MFCS 2012, LMCS 2015]
- Pathwidth (this ICALP!! please come to my talk!!)

What is different now?



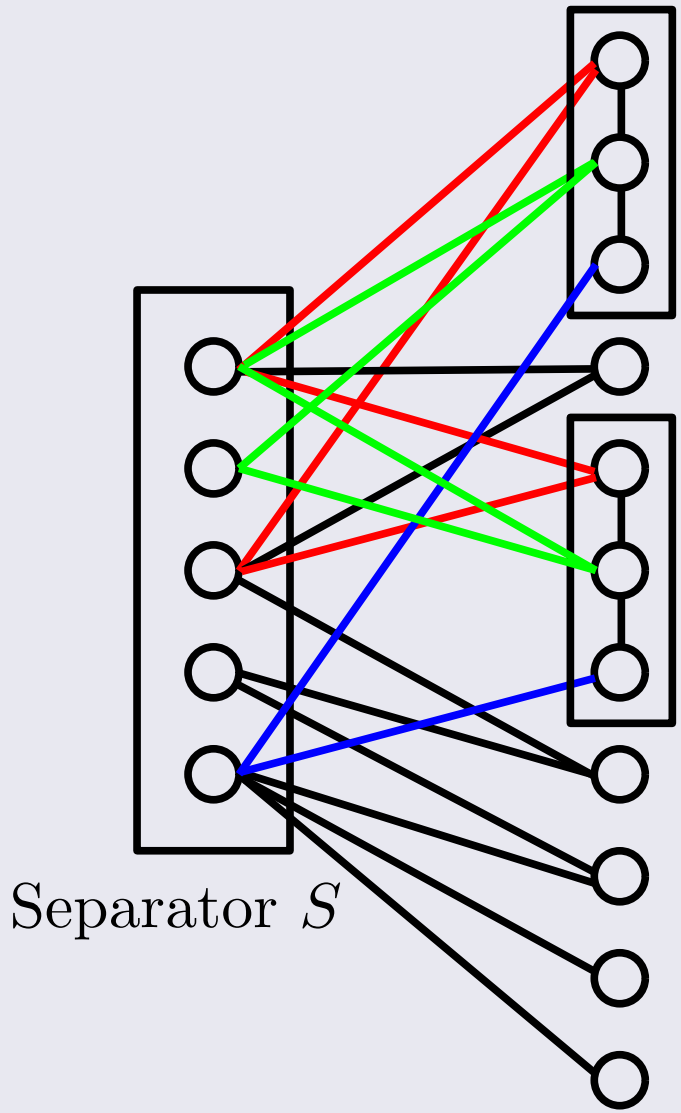
- Main idea: some components of $G - S$ are the same.
  - The same internally.
  - The same with respect to S.
- More precisely:
  - Two components C_1, C_2 of $G - S$ are "the same" if there exists an automorphism of G that maps C_1 to C_2.

What is different now?



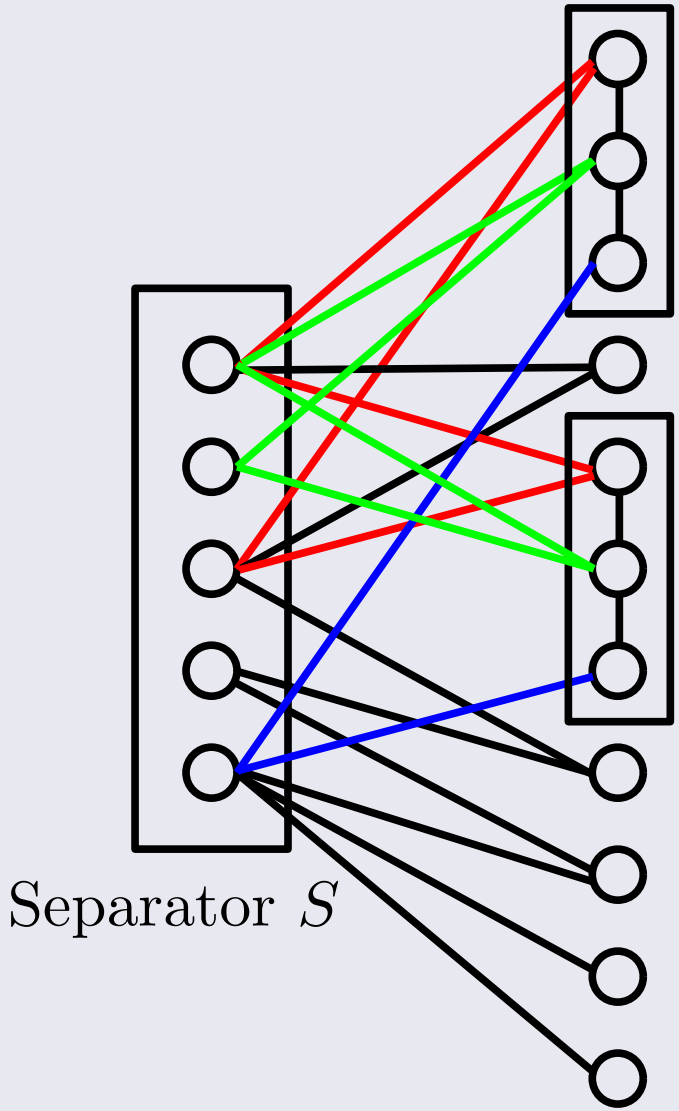
- Main idea: some components of $G - S$ are the same.
  - The same internally.
  - The same with respect to S.
- More precisely:
  - Two components C_1, C_2 of $G - S$ are "the same" if there exists an automorphism of G that maps C_1 to C_2.

What is different now?



- Main idea: some components of $G - S$ are the same.
  - The same internally.
  - The same with respect to S.
- More precisely:
  - Two components C_1, C_2 of $G - S$ are "the same" if there exists an automorphism of G that maps C_1 to C_2.

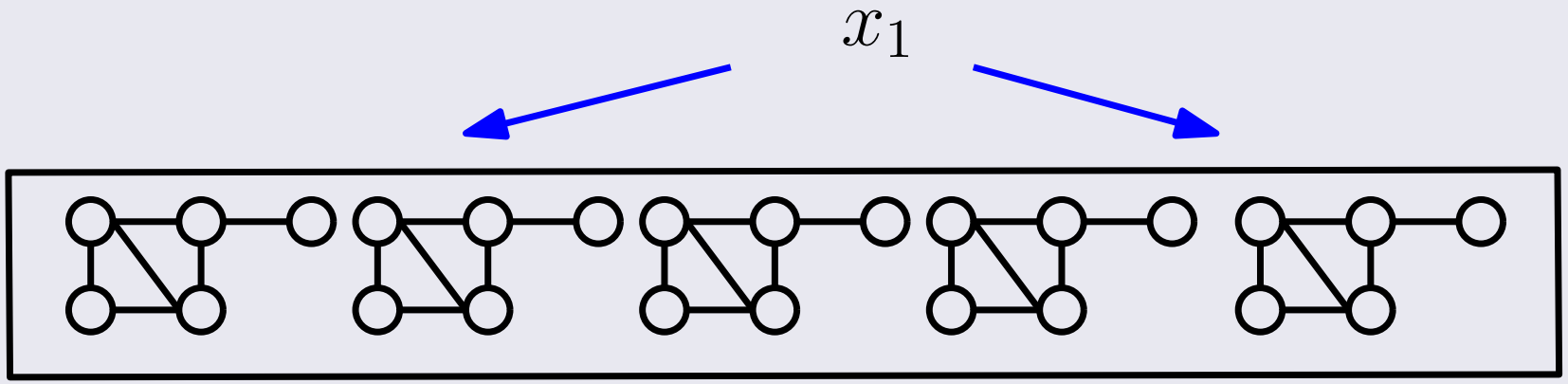
How many types of components?



- Equivalent components of $G - S$ are
  - The same internally.
  - The same with respect to S.
- How many choices?
- Recall, components of $G - S$ have size $\leq \mathrm{vi}$
  - At most 2^{vi^2} different internal structures.
  - At most 2^{vi^2} different connections to S.
- All in all, $2^{O(\mathrm{vi}^2)}$ possible types.

Counting Power – FO

How many identical components can we distinguish with q FO quantifiers?



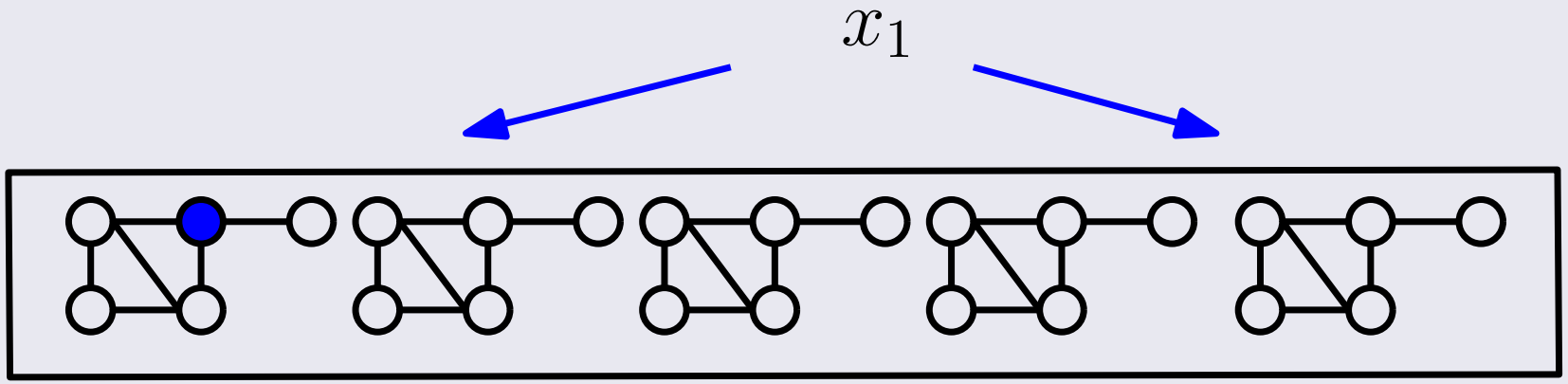
Claim: if we have $> q$ components, we can delete one.

Induction:

- Suppose true for $q - 1$ quantifiers.
- We have a formula $\exists x_1 \psi(x_1)$, where ψ has $q - 1$ quantifiers.
- Mapping it to any component is the same.
- We have $> q - 1$ identical components left.
- By induction, we can delete one.

Counting Power – FO

How many identical components can we distinguish with q FO quantifiers?



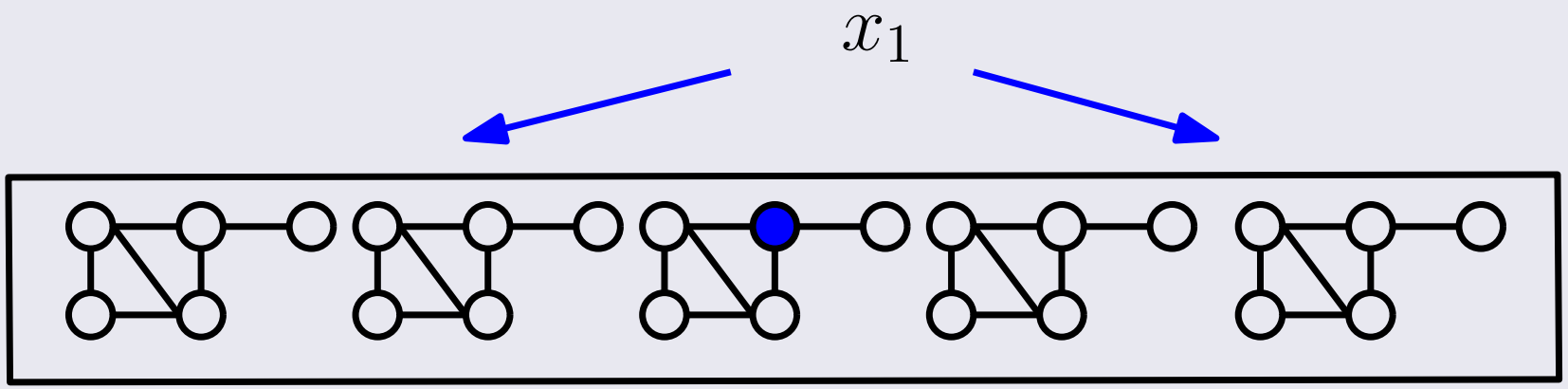
Claim: if we have $> q$ components, we can delete one.

Induction:

- Suppose true for $q - 1$ quantifiers.
- We have a formula $\exists x_1 \psi(x_1)$, where ψ has $q - 1$ quantifiers.
- Mapping it to any component is the same.
- We have $> q - 1$ identical components left.
- By induction, we can delete one.

Counting Power – FO

How many identical components can we distinguish with q FO quantifiers?



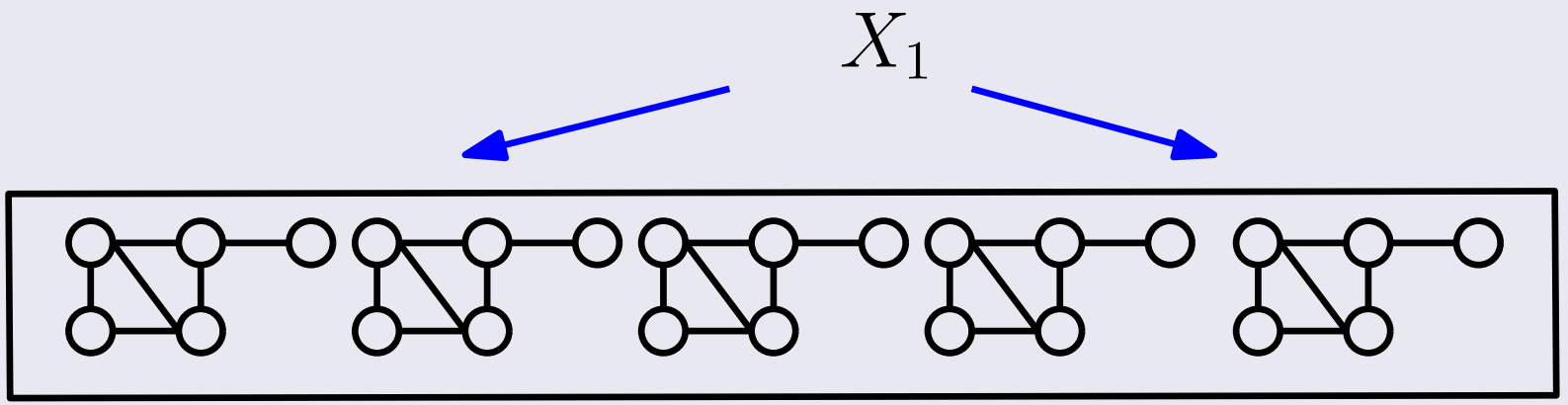
Claim: if we have $> q$ components, we can delete one.

Induction:

- Suppose true for $q - 1$ quantifiers.
- We have a formula $\exists x_1 \psi(x_1)$, where ψ has $q - 1$ quantifiers.
- Mapping it to any component is the same.
- We have $> q - 1$ identical components left.
- By induction, we can delete one.

Counting Power – MSO

How many components can we distinguish with q MSO quantifiers?



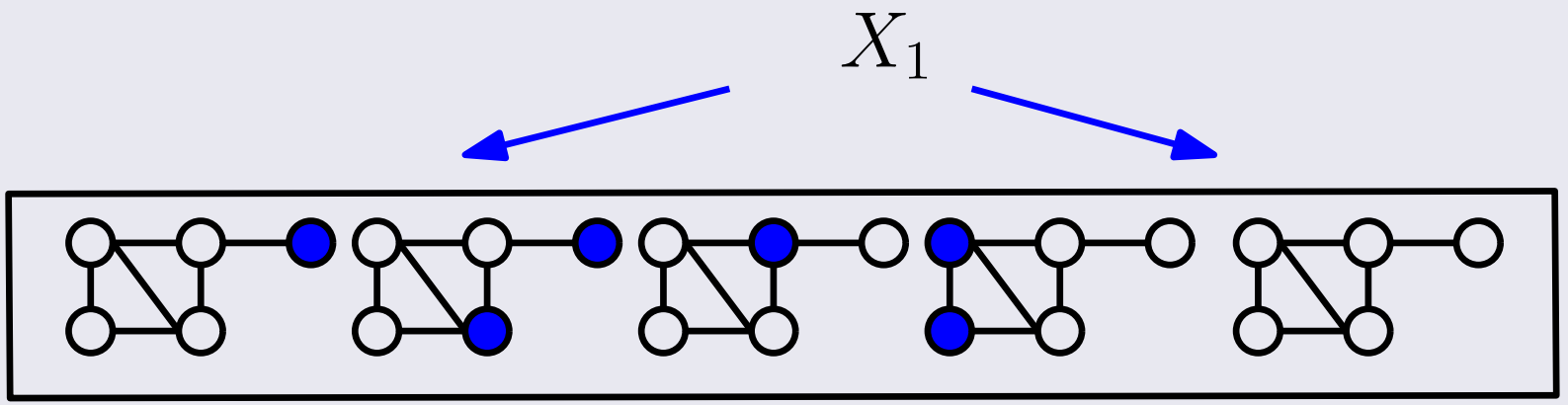
Claim: if we have $>$?? components, we can delete one.

Problem:

- When we select a set X_1 this may distinguish many components.
- Intuitively: if X_1 interacts with two previously identical components in different ways, these components are not identical any more!
- What to do?

Counting Power – MSO

How many components can we distinguish with q MSO quantifiers?



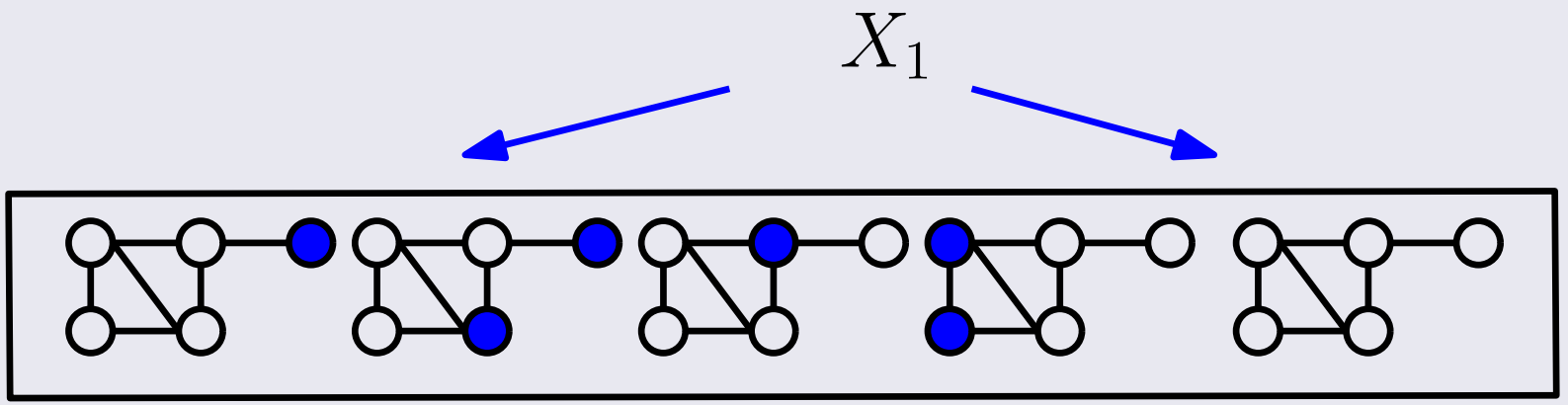
Claim: if we have $>$?? components, we can delete one.

Problem:

- When we select a set X_1 this may distinguish many components.
- Intuitively: if X_1 interacts with two previously identical components in different ways, these components are not identical any more!
- What to do?

Counting Power – MSO (cont'd)

How many components can we distinguish with q MSO quantifiers?



Claim: if we have $> 2^{\mathrm{vi} \cdot q}$ components, we can delete one.

Solution:

- Our components have size $\leq \mathrm{vi}$.
- There are at most 2^{vi} intersections of X_1 with each component.
- If we have $> 2^{\mathrm{vi} \cdot q}$ identical components initially...
- ...by PHP one intersection type appears $> 2^{\mathrm{vi} \cdot q} / 2^{\mathrm{vi}} = 2^{\mathrm{vi}(q-1)}$ times.
- These components are identical, use inductive hypothesis!

Putting things together

- There are at most 2^{vi^2} types of components.
- Maximum number of same components in reduced graph is
  - q for FO logic.
  - $2^{\mathrm{vi} \cdot q}$ for MSO logic.

Putting things together

- There are at most 2^{vi^2} types of components.
- Maximum number of same components in reduced graph is
  - q for FO logic.
  - $2^{\mathrm{vi} \cdot q}$ for MSO logic.
- For FO logic
  - Reduced graph has size $q2^{\mathrm{vi}^2}$.
  - Trivial algorithm runs in $2^{q \cdot \mathrm{vi}^2} q^q$.

Putting things together

- There are at most 2^{vi^2} types of components.
- Maximum number of same components in reduced graph is
  - q for FO logic.
  - $2^{\mathrm{vi}\cdot q}$ for MSO logic.
- For FO logic
  - Reduced graph has size $q2^{\mathrm{vi}^2}$.
  - Trivial algorithm runs in $2^{q\cdot \mathrm{vi}^2}q^q$.
- For MSO logic
  - Reduced graph has size $2^{\mathrm{vi}^2+\mathrm{vi}\cdot q}$.
  - Trivial algorithm runs in $2^{2^{vi^2}+\mathrm{vi}\cdot q}$.
- Are these meta-theorems optimal?

Putting things together

- There are at most 2^{vi^2} types of components.
- Maximum number of same components in reduced graph is
  - q for FO logic.
  - $2^{\mathrm{vi}\cdot q}$ for MSO logic.
- For FO logic
  - Reduced graph has size $q2^{\mathrm{vi}^2}$.
  - Trivial algorithm runs in $2^{q\cdot \mathrm{vi}^2} q^q$.
- For MSO logic
  - Reduced graph has size $2^{\mathrm{vi}^2+\mathrm{vi}\cdot q}$.
  - Trivial algorithm runs in $2^{2^{vi^2}+\mathrm{vi}\cdot q}$.
- Are these meta-theorems optimal?

Yes!! (under ETH) – details skipped

Tree-depth Meta-theorem

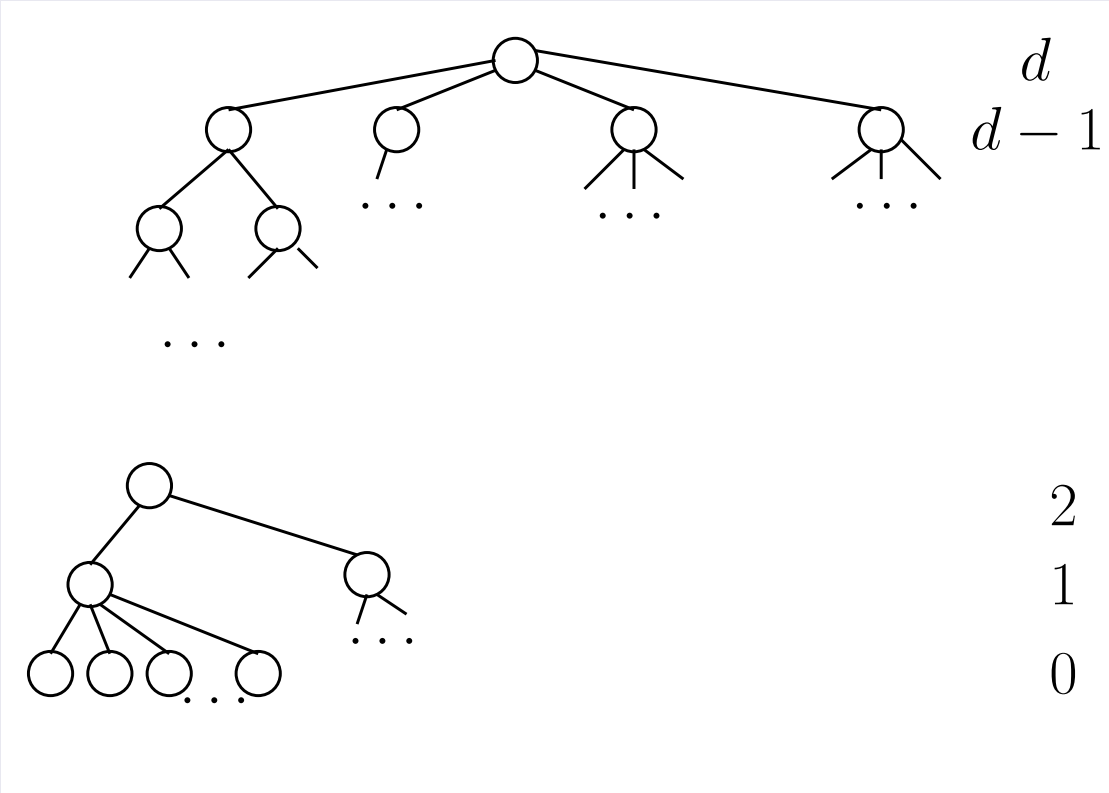
Any fool can come up with an exponential-time algorithm...

Tree-depth Meta-theorem

Any fool can come up with an exponential-time algorithm. . .
but to come up with a tower of exponentials, **you have to really know what you're doing!**

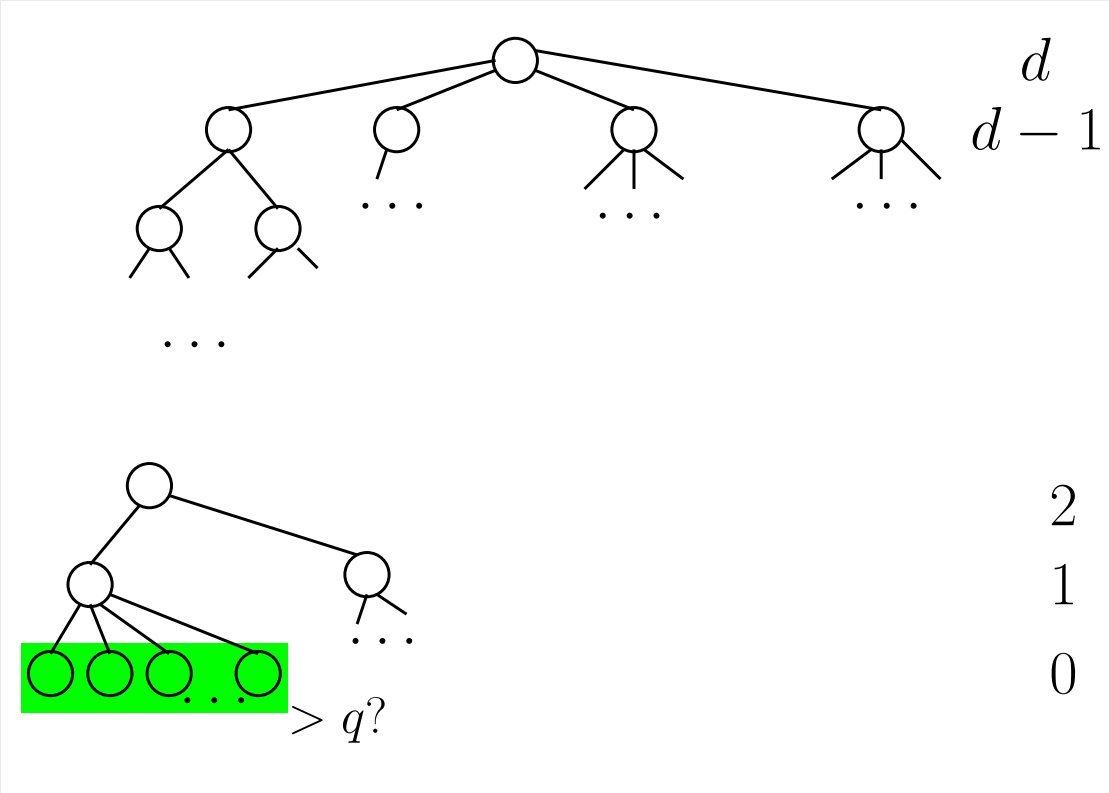
(Daniel Marx)

Tree-depth Meta-theorem



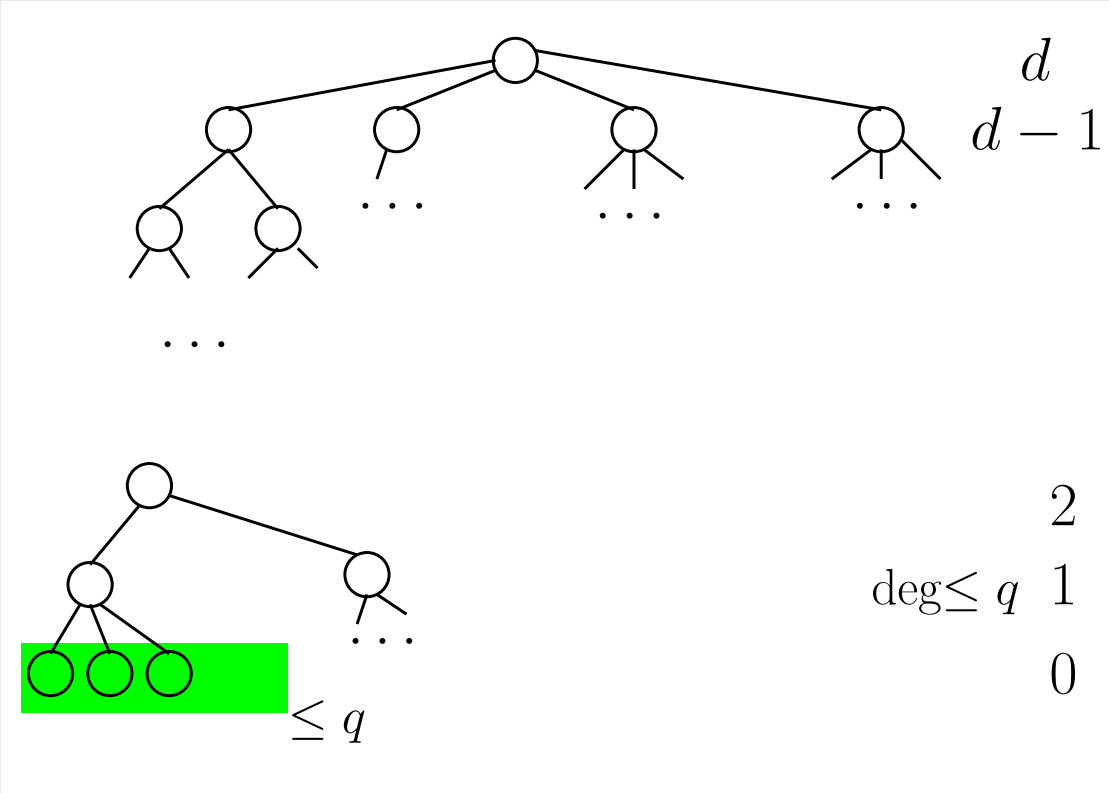
We have a rooted tree with d layers (d fixed)

Tree-depth Meta-theorem



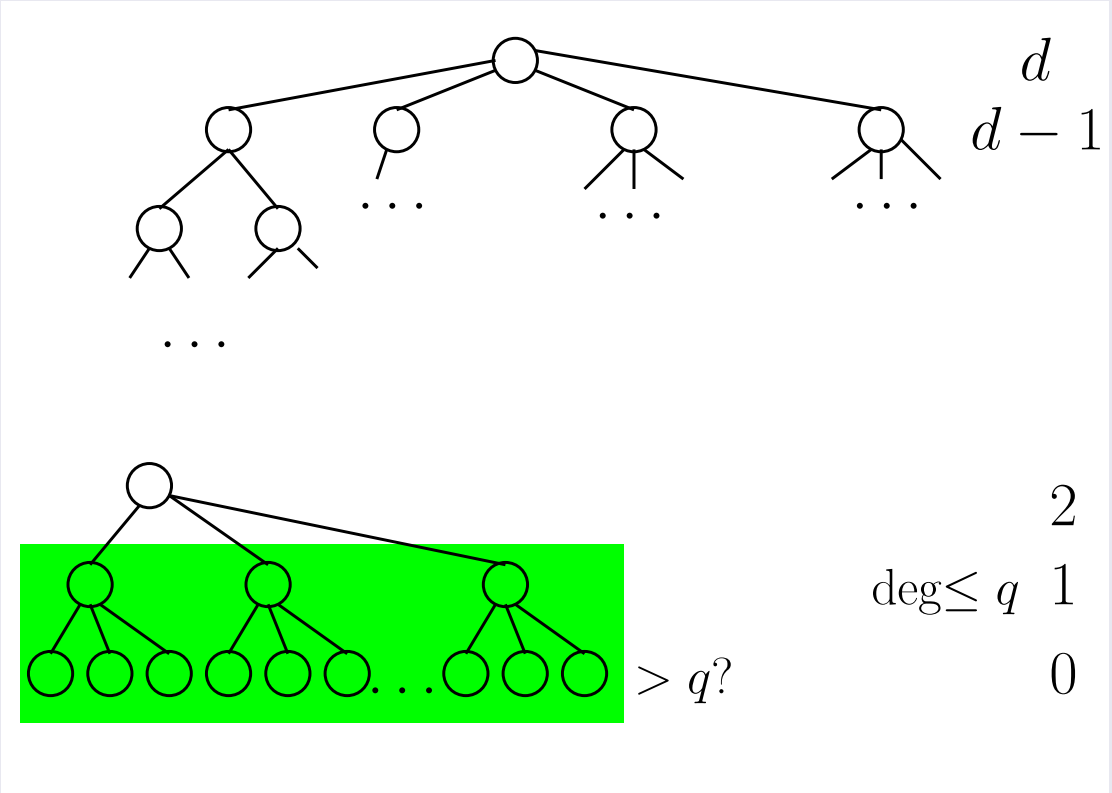
Apply the previous argument to the bottom layer (leaves)

Tree-depth Meta-theorem



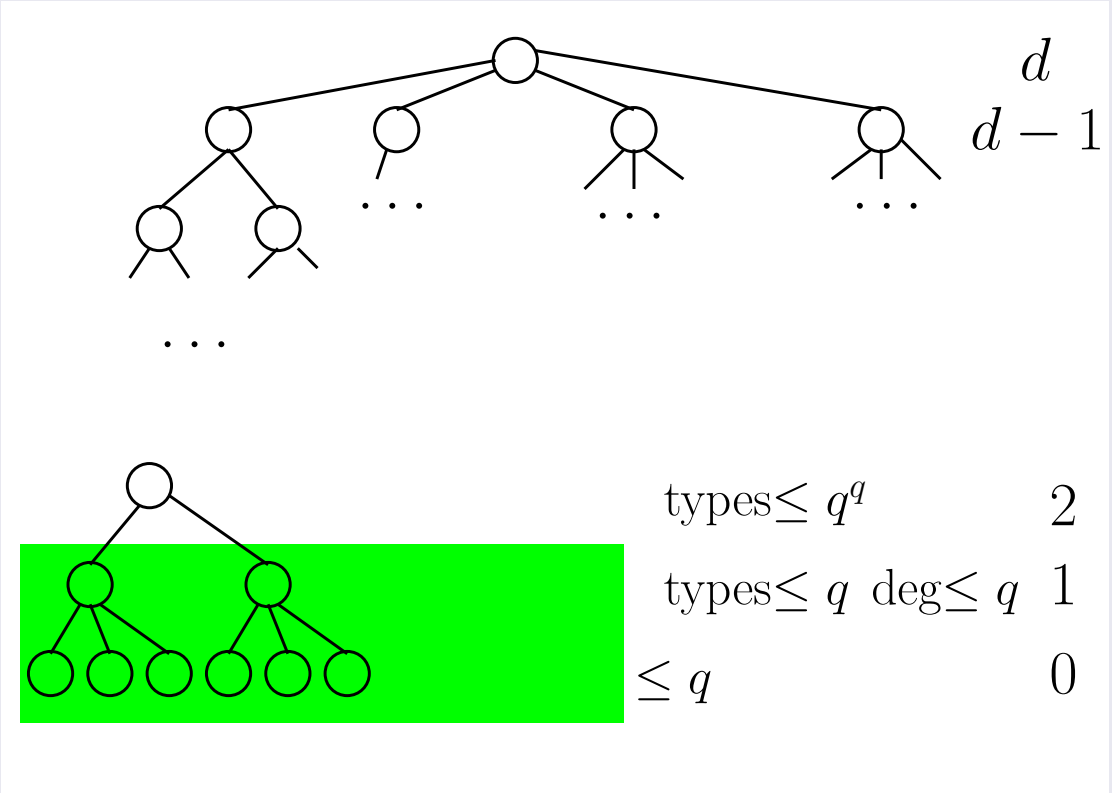
Apply the previous argument to the bottom layer (leaves)

Tree-depth Meta-theorem



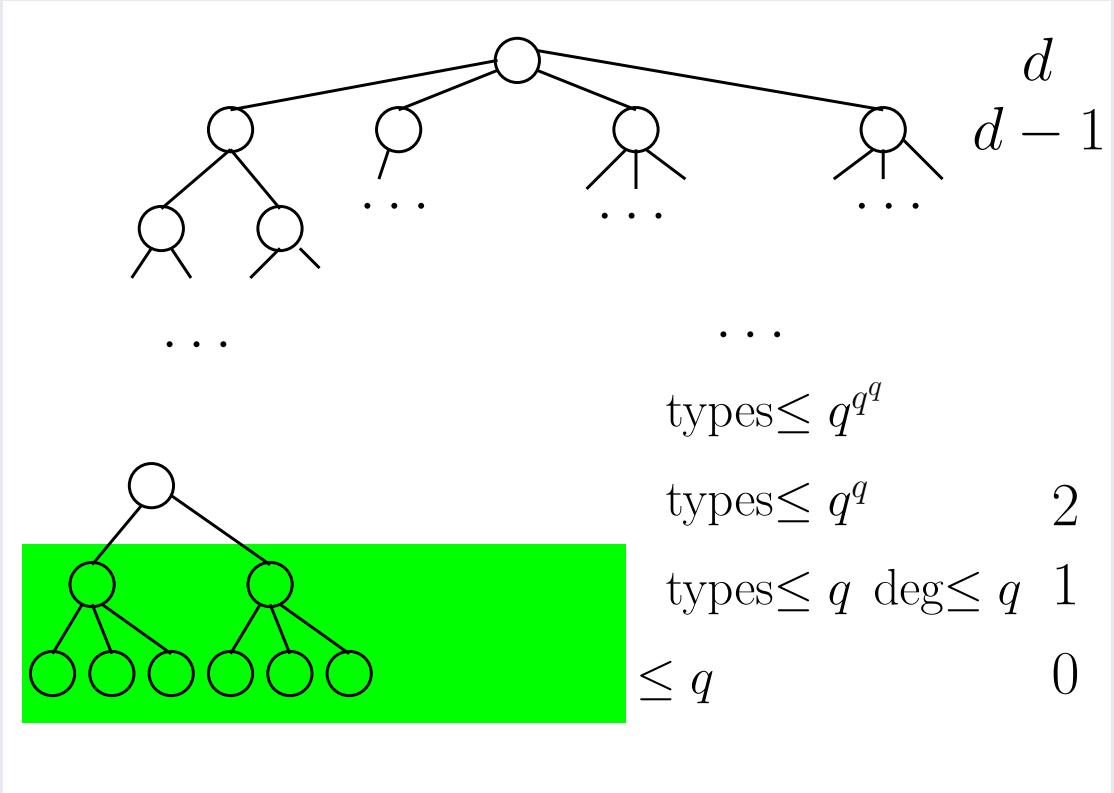
Key intuition: same argument can be applied to level 2, deleting identical sub-trees.

Tree-depth Meta-theorem



Key intuition: same argument can be applied to level 2, deleting identical sub-trees.

Tree-depth Meta-theorem



There are q^q different "types" of vertices at level 2. Applying the same argument to level 3, there are q^{q^q} types of vertices of level 3. ...

In the end graph has bounded size![2]

[2] bounded by a tower of exponentials of height d.

SAT again?

Meta-Theorems to QBF-SAT?

Intuition: elementary meta-theorem should give FPT algorithm for QBF?

Elementary dependence meta-theorems

FO logic & pathwidth	
MSO logic & tree-depth	
MSO logic & vertex integrity	

Meta-Theorems to QBF-SAT?

Intuition: elementary meta-theorem should give FPT algorithm for QBF?

Elementary dependence meta-theorems

FO logic & pathwidth	No! [Atserias and Oliva JCSS 2014]
MSO logic & tree-depth	
MSO logic & vertex integrity	

Meta-Theorems to QBF-SAT?

Intuition: elementary meta-theorem should give FPT algorithm for QBF?

Elementary dependence meta-theorems

FO logic & pathwidth	No! [Atserias and Oliva JCSS 2014]
MSO logic & tree-depth	???
MSO logic & vertex integrity	

Meta-Theorems to QBF-SAT?

Intuition: elementary meta-theorem should give FPT algorithm for QBF?

Elementary dependence meta-theorems

FO logic & pathwidth	No! [Atserias and Oliva JCSS 2014]
MSO logic & tree-depth	???
MSO logic & vertex integrity	???

Meta-Theorems to QBF-SAT?

Intuition: elementary meta-theorem should give FPT algorithm for QBF?

Elementary dependence meta-theorems

FO logic & pathwidth	No! [Atserias and Oliva JCSS 2014]
MSO logic & tree-depth	???
MSO logic & vertex integrity	???

- Complexity of QBF for these parameters is **OPEN!**
- Intuitive difficulty: graph does not capture order of quantification of variables.

Typical Hard Problems

Where is this useful?

- Typical example problems are complete for Σ_2^p or higher levels of PH.

Where is this useful?

- Typical example problems are complete for Σ_2^p or higher levels of PH.
  - This is not a rule! (cf. Esther's talk)
  - This is not even true for $\exists\forall$-SAT instances we saw!

Where is this useful?

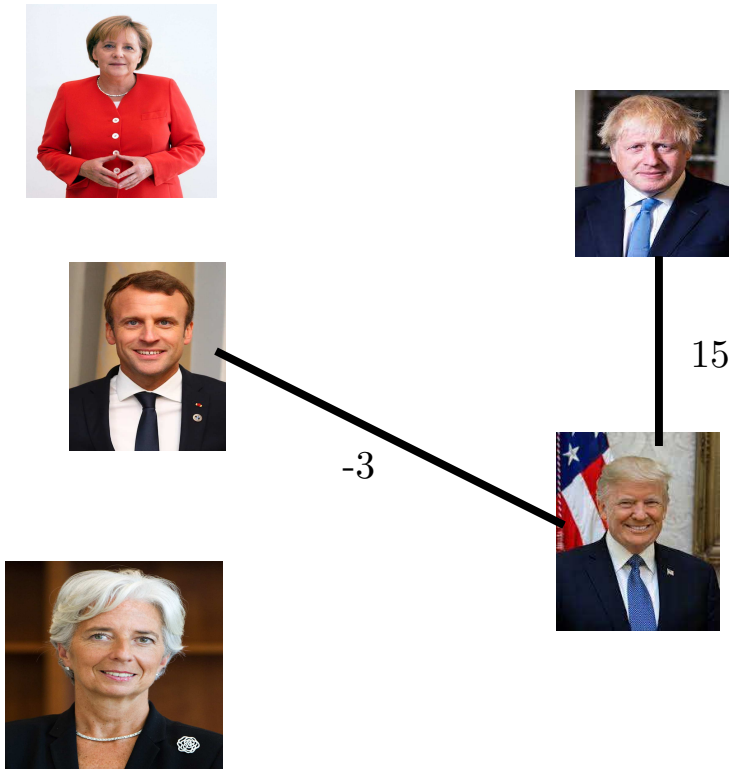
- Typical example problems are complete for Σ_2^p or higher levels of PH.
- Applications:
  - Reduce $\exists\forall$-SAT to your problem to get double-exponential lower bound.
  - Reduce your problem to $\exists\forall$-SAT to get double-exponential upper bound. [L., Mengel, Mitsou, SAT 2018]

  Examples:

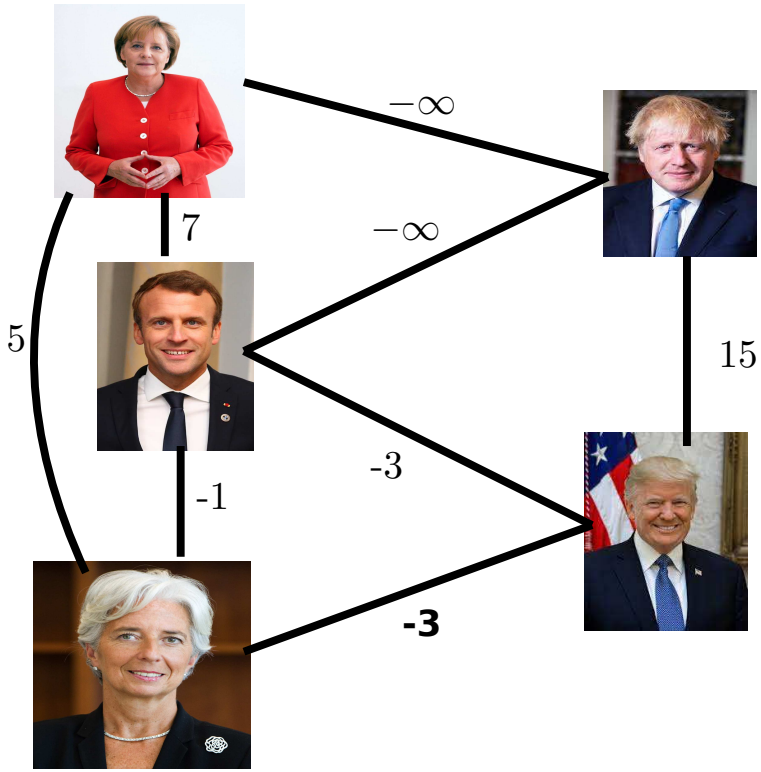
  - k-Choosability (easier proof than [Marx, Mitsou, ICALP 2016])
  - Stability in Hedonic games (ongoing work with Tesshu Hanaka and Noleen Köhler)

Hedonic games

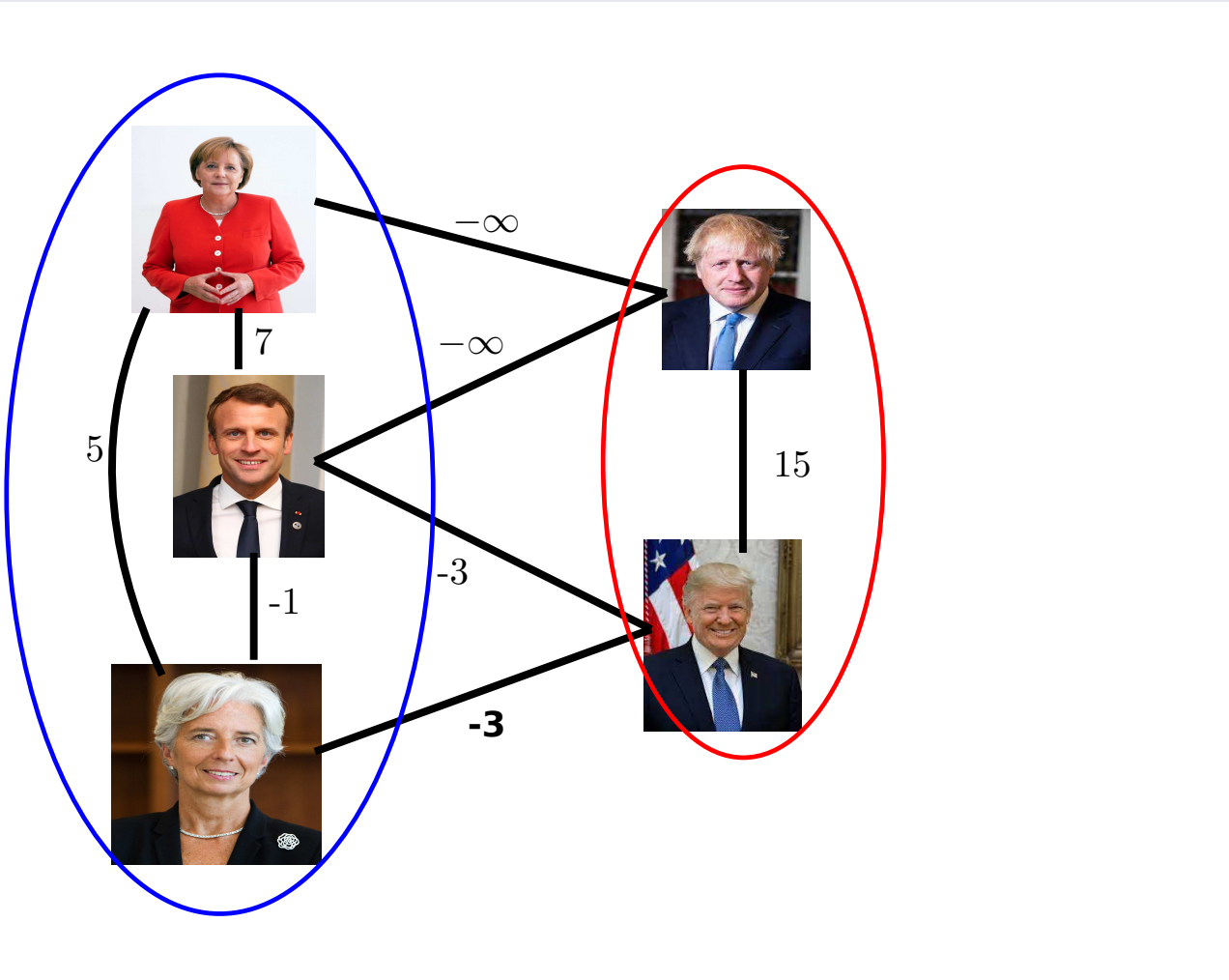
Hedonic games



Hedonic games



Hedonic games



Hedonic games

Question: Does a Nash-stable partition exist?

- "Correct" complexity is $(\Delta\mathrm{tw})^{O(\Delta\mathrm{tw})}$ [Hanaka, L. ESA 2022]

Hedonic games

Question: Does a Nash-stable partition exist?

- "Correct" complexity is $(\Delta\mathrm{tw})^{O(\Delta\mathrm{tw})}$ [Hanaka, L. ESA 2022]

Question: Does a Core-stable partition exist?

- Partition that resists any coalition of diverging agents.
- Σ_2^p-complete for constant Δ
- Σ_2^p-complete for constant vc

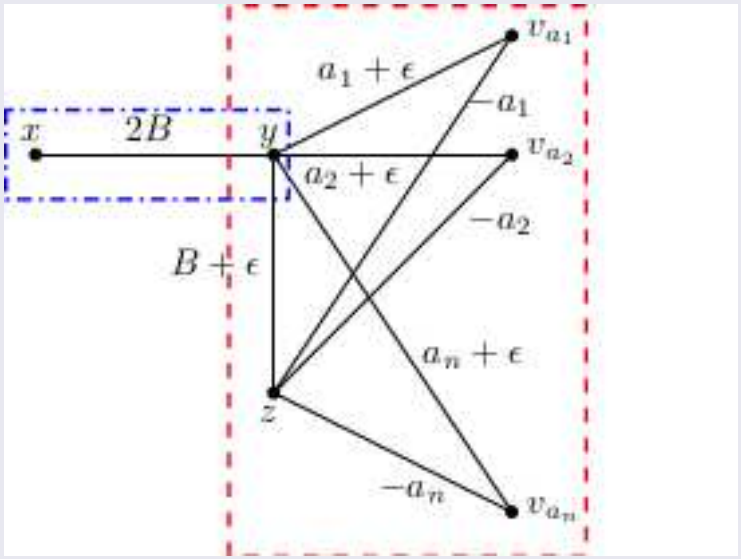
Hedonic games

Question: Does a Nash-stable partition exist?

- "Correct" complexity is $(\Delta\mathrm{tw})^{O(\Delta\mathrm{tw})}$ [Hanaka, L. ESA 2022]

Question: Does a Core-stable partition exist?

- Partition that resists any coalition of diverging agents.
- Σ_2^p-complete for constant Δ
- Σ_2^p-complete for constant vc



Hedonic games

Question: Does a Nash-stable partition exist?

- "Correct" complexity is $(\Delta\mathrm{tw})^{O(\Delta\mathrm{tw})}$ [Hanaka, L. ESA 2022]

Question: Does a Core-stable partition exist?

- Partition that resists any coalition of diverging agents.
- Σ_2^p-complete for constant Δ
- Σ_2^p-complete for constant vc

- Correct complexity is **double-exponential** in $\mathrm{tw} + \Delta$
- Upper bound: reduce to $\exists\forall$-SAT
- Lower bound: run existing Σ_2-completeness proof from $\exists\forall$-SAT instance with bounded Δ and $\mathrm{tw} = O(\log n)$.

Conclusions

Conclusions

- Quantifier alternations **might** give extra levels of exponentiation
- Even poly-time computable quantifier alternations can do this! (cf. Esther's talk)
- Σ_2^p-complete problems are more likely candidates for this
- Reductions from $\exists\forall$-SAT may give easy double-exponential lower bounds!

Conclusions

- Quantifier alternations **might** give extra levels of exponentiation
- Even poly-time computable quantifier alternations can do this! (cf. Esther's talk)
- Σ_2^p-complete problems are more likely candidates for this
- Reductions from $\exists\forall$-SAT may give easy double-exponential lower bounds!

Open questions:

- Complexity of QBF parameterized by tree-depth/vertex integrity?
- QBF parameterized by vertex cover has 2^{vc^k} running time for k-CNF. Optimal?

Conclusions

- Quantifier alternations **might** give extra levels of exponentiation
- Even poly-time computable quantifier alternations can do this! (cf. Esther's talk)
- Σ_2^p-complete problems are more likely candidates for this
- Reductions from $\exists\forall$-SAT may give easy double-exponential lower bounds!

Open questions:

- Complexity of QBF parameterized by tree-depth/vertex integrity?
- QBF parameterized by vertex cover has 2^{vc^k} running time for k-CNF. Optimal?

Thank you!