Graph Modification Problems

A Modern Perspective

Setting the stage: Definitions and Preliminary Observations

Setting the stage: Definitions and Preliminary Observations

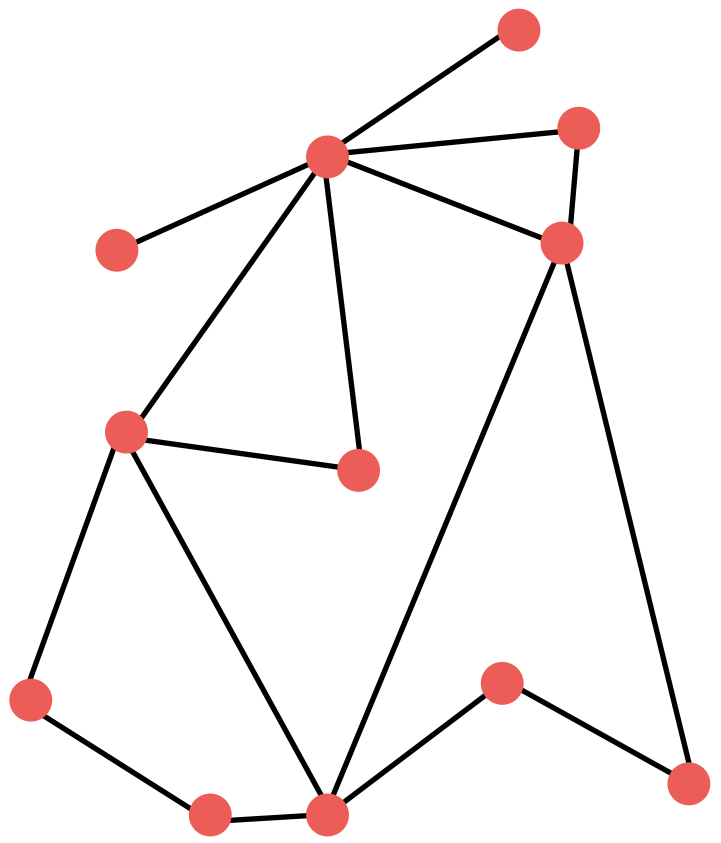
Brief excursions into specific examples

Setting the stage: Definitions and Preliminary Observations

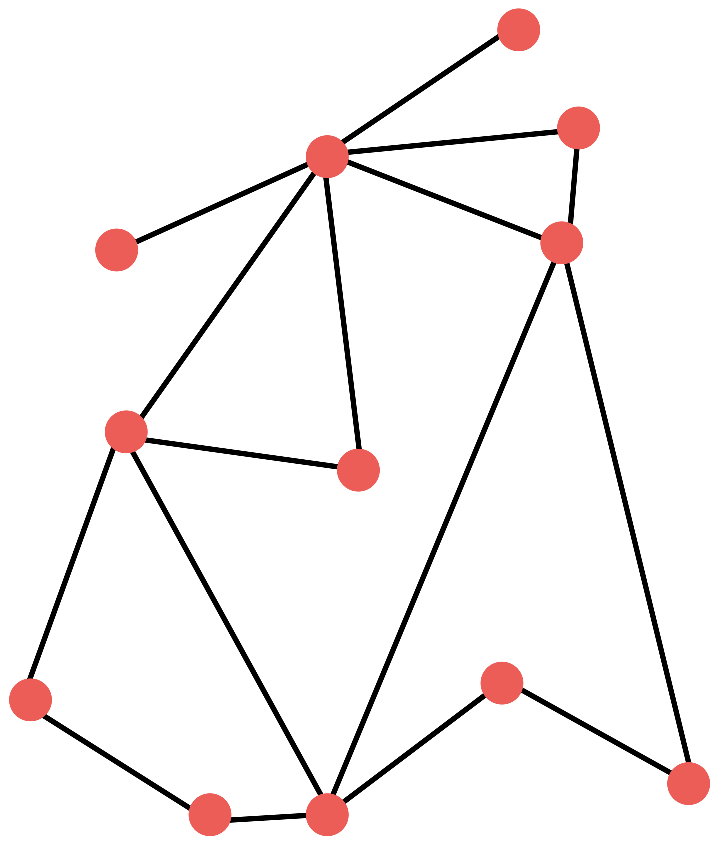
Brief excursions into specific examples

Current Trends & Future Directions

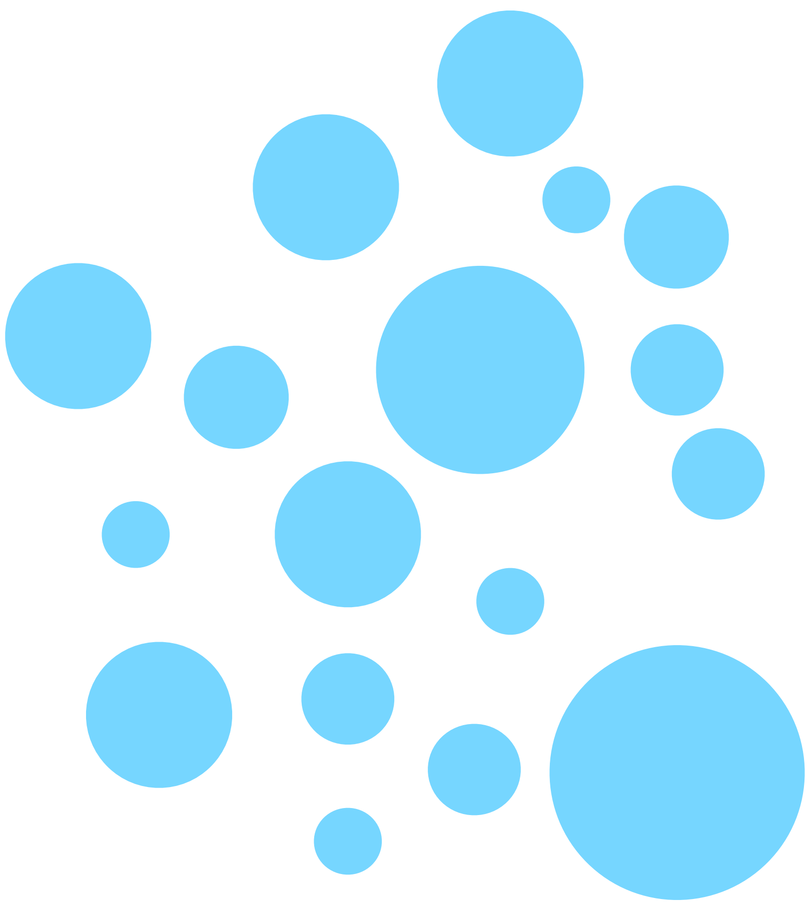
Ingredients of a typical Graph Modification Problem



Input Graph, G

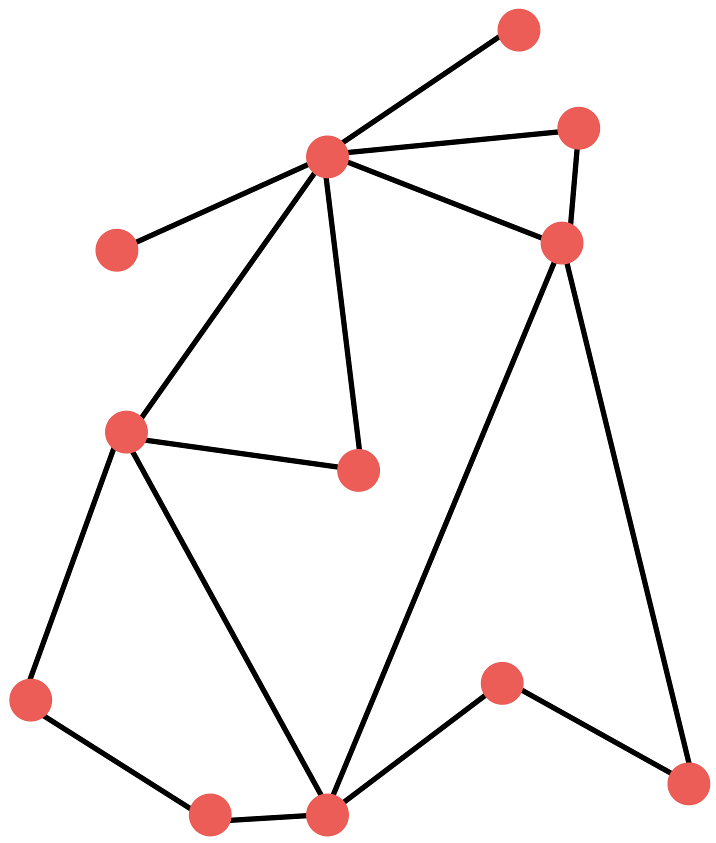


Input Graph, G

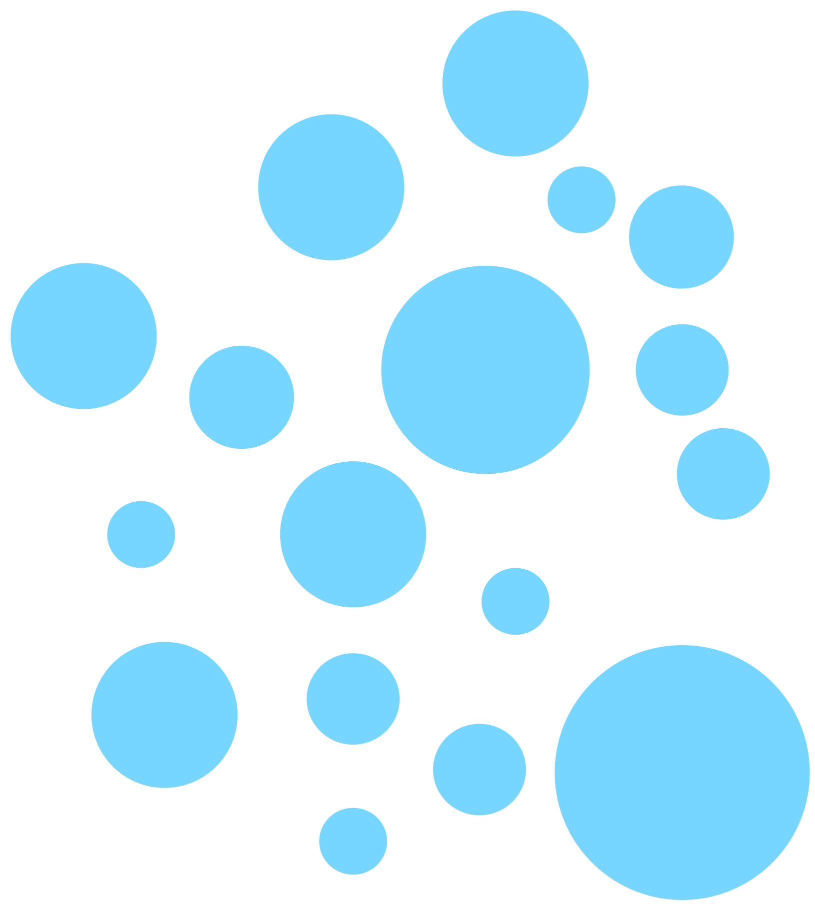


Graph Class or “property”, π

Goal: Transform G so that it belongs to **π** with "minimum damages".

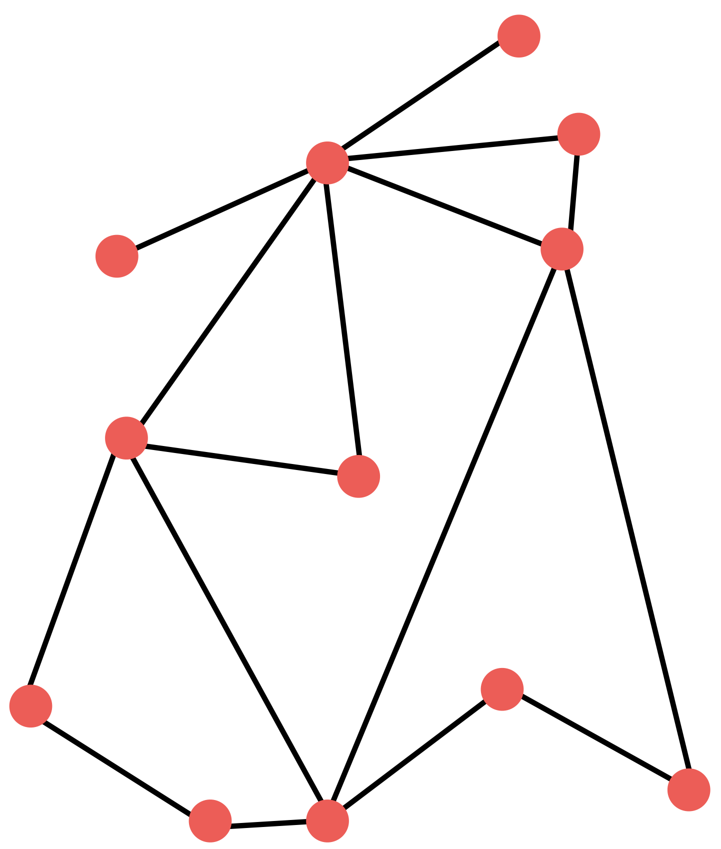


Input Graph, G

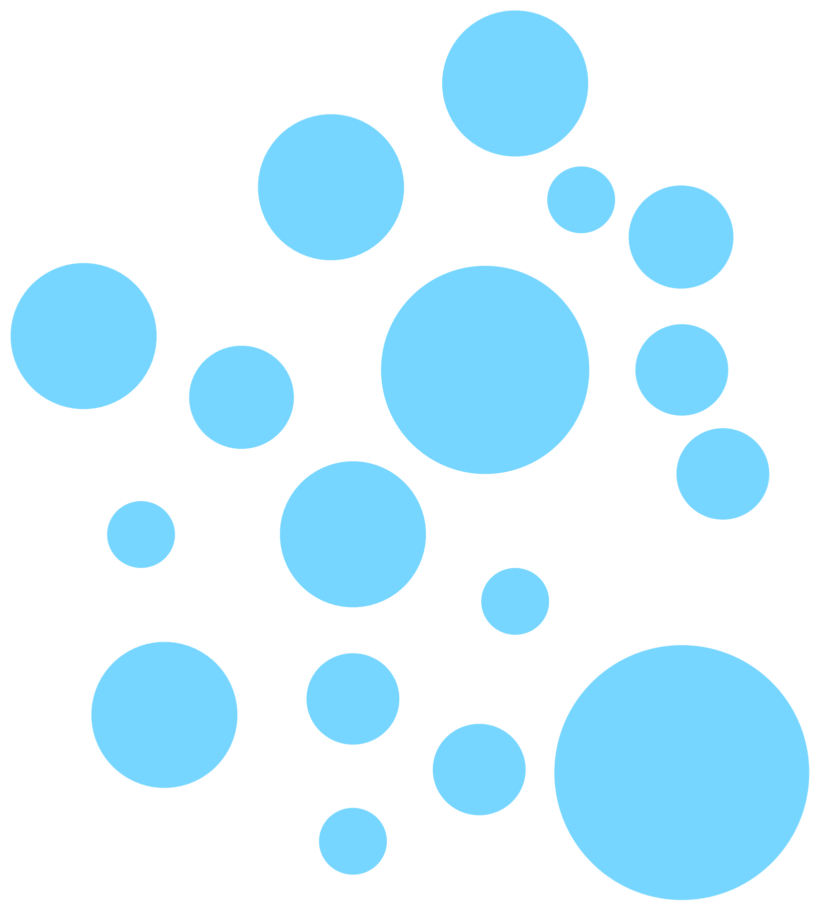


Graph Class or "property", π

Goal: Transform G so that it belongs to **π** with "minimum damages".



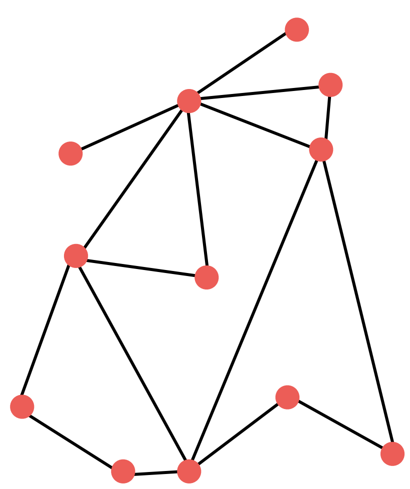
Input Graph, G



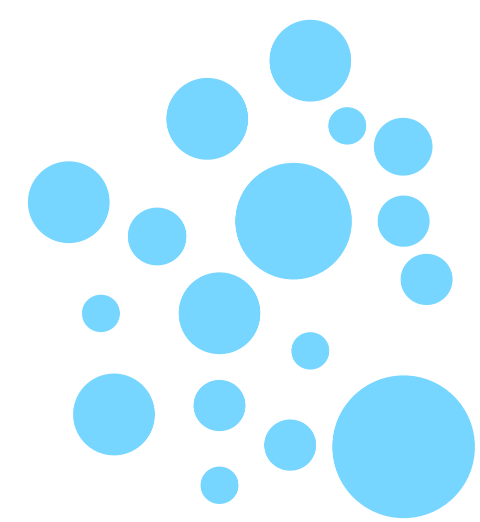
Graph Class or "property", π

Goal: Transform G so that it belongs to π with “minimum damages”.

Input Graph, G



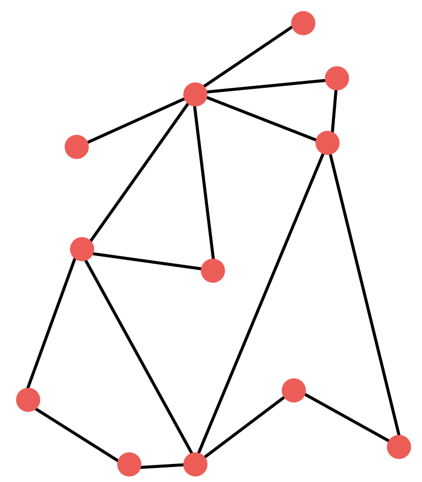
Graph Class or “property”, π



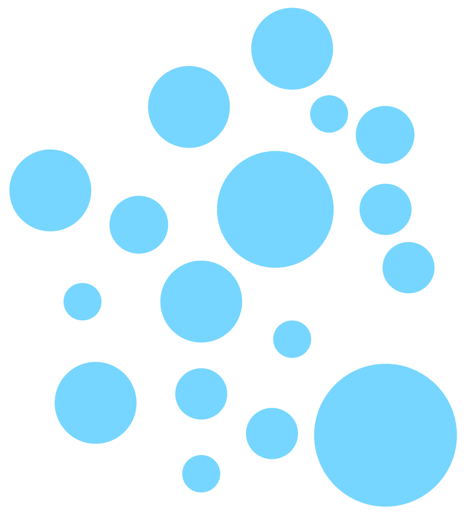
Goal: Transform G so that it belongs to π with "minimum damages".

vertex deletions

Input Graph, G



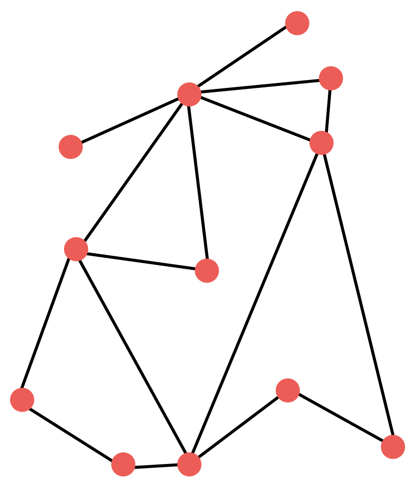
Graph Class or "property", π



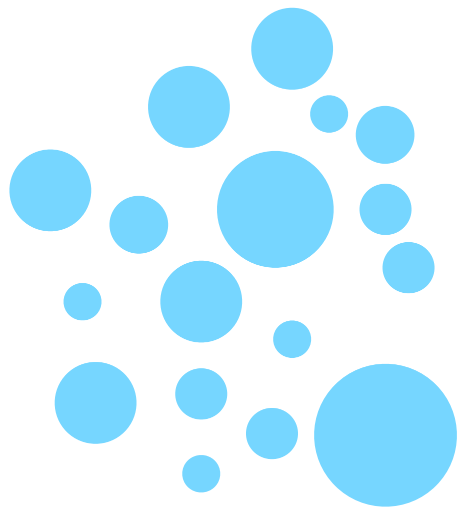
Goal: Transform G so that it belongs to **π** with "minimum damages".

vertex deletions   edge deletions

Input Graph, G



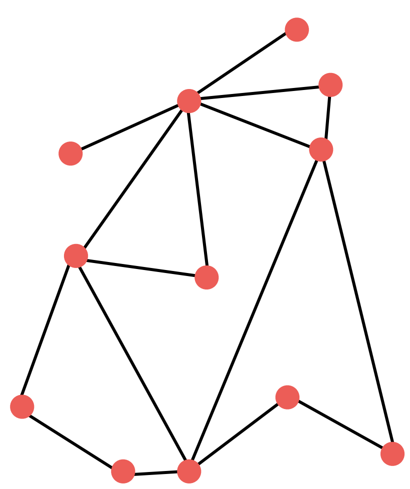
Graph Class or "property", π



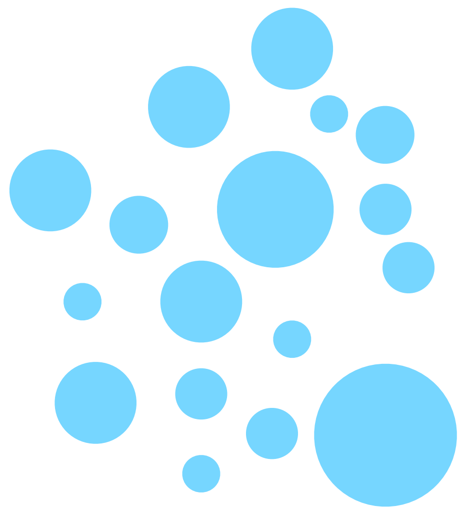
Goal: Transform G so that it belongs to π with "minimum damages".

vertex deletions edge deletions edge additions

Input Graph, G



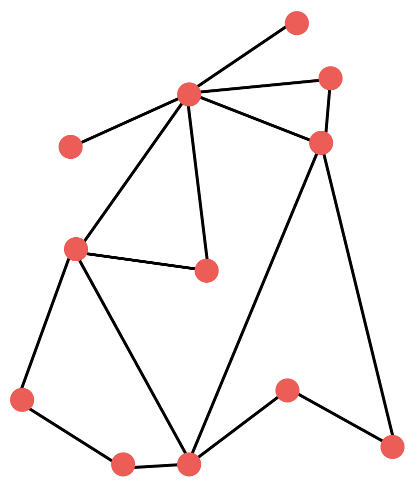
Graph Class or "property", π



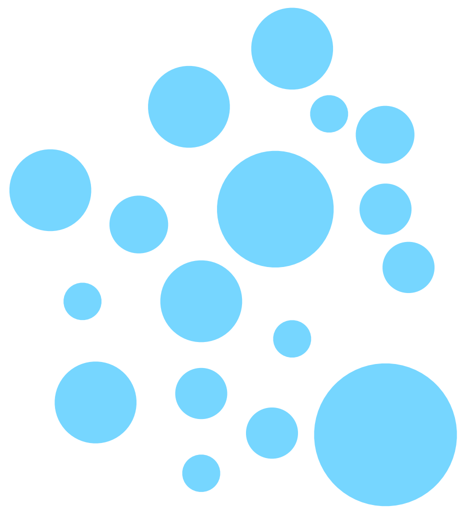
Goal: Transform G so that it belongs to π with "minimum damages".

vertex deletions edge deletions edge additions edge contractions

Input Graph, G



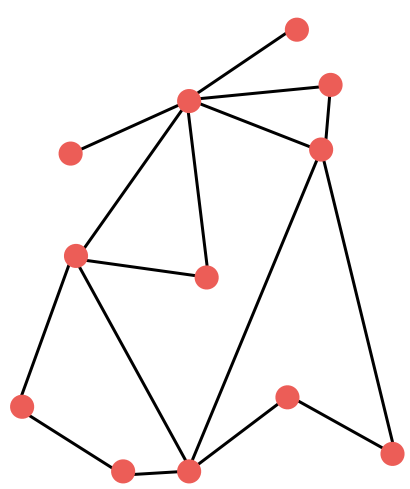
Graph Class or "property", π



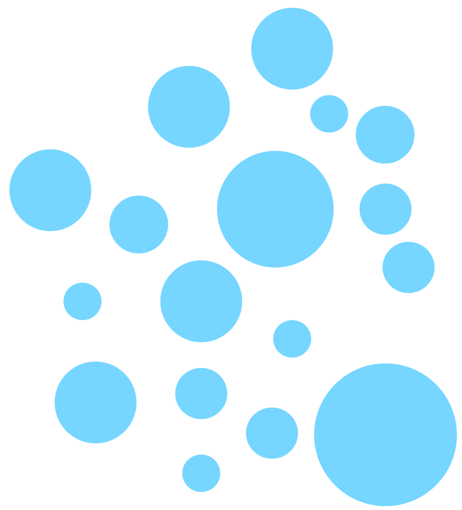
Goal: Transform G so that it belongs to **π** with "minimum damages".

vertex deletions edge deletions edge additions edge contractions edge editing

Input Graph, G



Graph Class or "property", π

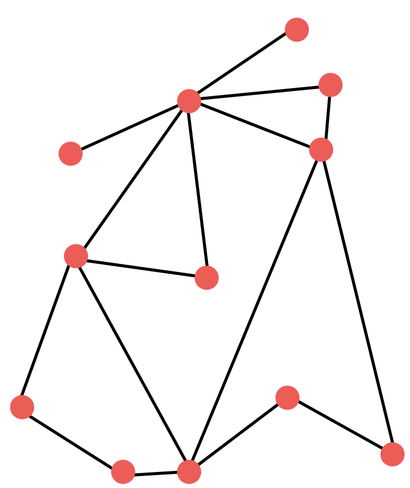


Goal: Transform G so that it belongs to π with "minimum damages".

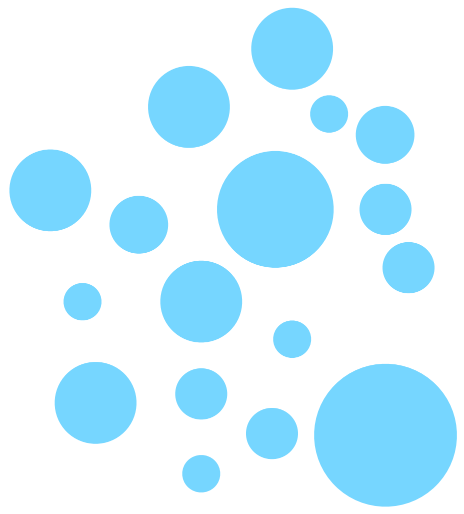
vertex deletions edge deletions edge additions edge contractions edge editing

VERTEX COVER

Input Graph, G



Edgeless Graphs

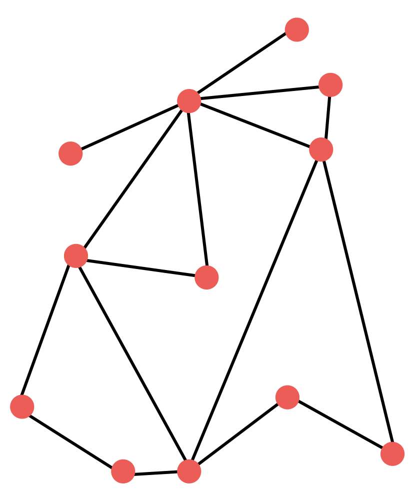


Goal: Transform G so that it belongs to π with "minimum damages".

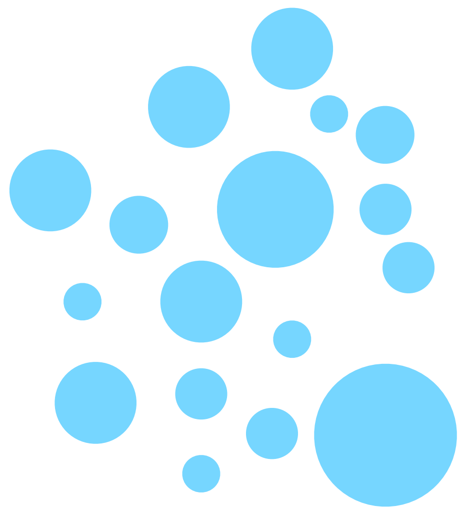
vertex deletions   edge deletions   edge additions   edge contractions   edge editing

Feedback Vertex Set

Input Graph, G



Acyclic Graphs

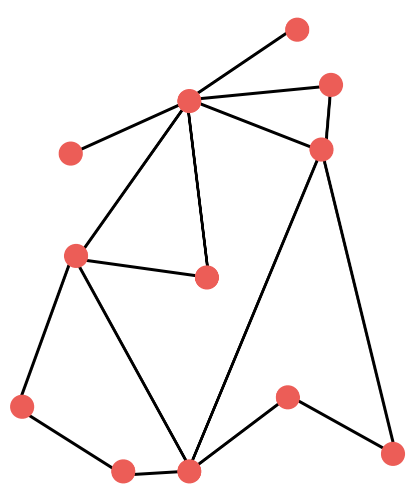


Goal: Transform G so that it belongs to π with "minimum damages".

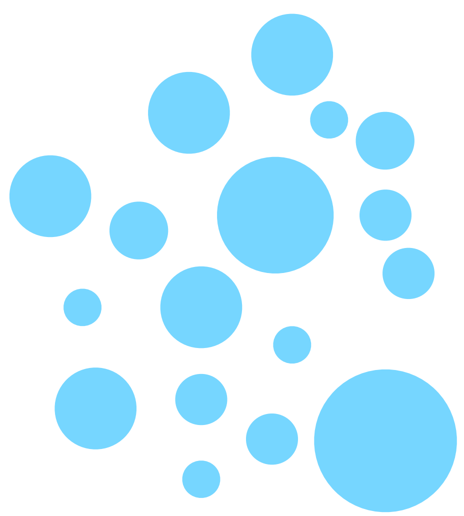
vertex deletions edge deletions **edge additions** edge contractions edge editing

MINIMUM FILL-IN

Input Graph, G



Chordal Graphs

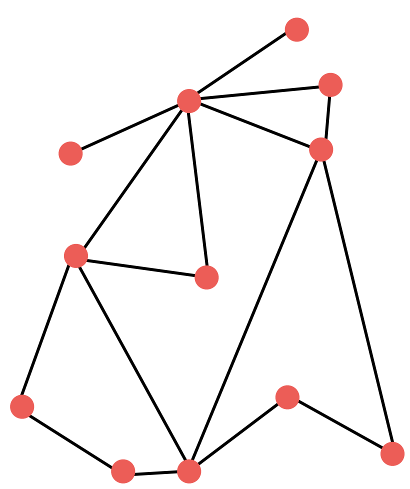


Goal: Transform G so that it belongs to π with "minimum damages".

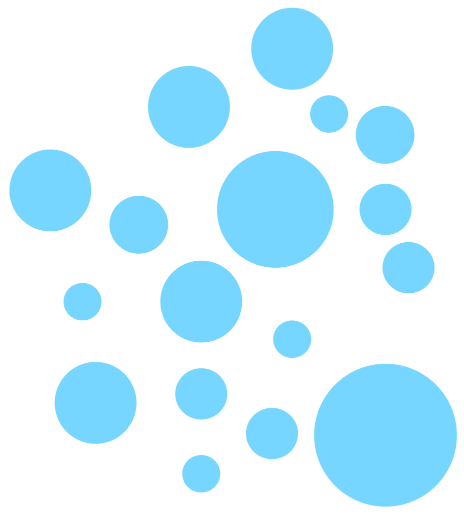
vertex deletions edge deletions edge additions edge contractions edge editing

Cluster Editing

Input Graph, G



Cluster Graphs



Goal: Transform G so that it belongs to π with “minimum damages”.

Goal: Transform G so that it belongs to **π** with "minimum damages".

Minimum **π**-Completion

Goal: Transform G so that it belongs to **π** with "minimum damages".

Minimum **π**-Completion

Minimum **π**-Supergraph

Goal: Transform G so that it belongs to **π** with "minimum damages".

Minimum **π**-Completion

Minimum **π**-Supergraph

Minimum **π**-Deletion

Goal: Transform G so that it belongs to **π** with "minimum damages".

Minimum **π**-Completion

Minimum **π**-Deletion

Minimum **π**-Supergraph

Maximum **π**-Spanning Subgraph

Goal: Transform G so that it belongs to **π** with “minimum damages”.

Minimum **π**-Completion

Minimum **π**-Deletion

Minimum **π**-Editing

Minimum **π**-Supergraph

Maximum **π**-Spanning Subgraph

Goal: Transform G so that it belongs to **π** with "minimum damages".

Minimum **π**-Completion

Minimum **π**-Deletion

Minimum **π**-Editing

Minimum **π**-Supergraph

Maximum **π**-Spanning Subgraph

Closest **π**-Graph

Goal: Transform G so that it belongs to π with "minimum damages".

Minimum π-Completion

Minimum π-Deletion

Minimum π-Editing

Minimum π-Vertex Deletion

Minimum π-Supergraph

Maximum π-Spanning Subgraph

Closest π-Graph

Goal: Transform G so that it belongs to **π** with "minimum damages".

Minimum **π**-Completion	Minimum **π**-Supergraph
Minimum **π**-Deletion	Maximum **π**-Spanning Subgraph
Minimum **π**-Editing	Closest **π**-Graph
Minimum **π**-Vertex Deletion	Maximum **π**-Induced Subgraph

Goal: Transform G so that it belongs to π with "minimum damages".

Minimum π-Completion

Minimum π-Deletion

Minimum π-Editing

Minimum π-Vertex Deletion

Minimum π-Supergraph

Maximum π-Spanning Subgraph

Closest π-Graph

Maximum π-Induced Subgraph

Completion to Minimum Max-Clique

Goal: Transform G so that it belongs to π with "minimum damages".

Minimum π-Completion

Minimum π-Supergraph

Minimum π-Deletion

Maximum π-Spanning Subgraph

Minimum π-Editing

Closest π-Graph

Minimum π-Vertex Deletion

Maximum π-Induced Subgraph

Completion to Minimum Max-Clique

Modification with Restrictions, eg, the Sandwich problem

Goal: Transform G so that it belongs to π with "minimum damages".

Minimum π-Completion

Minimum π-Supergraph

Minimum π-Deletion

Maximum π-Spanning Subgraph

Minimum π-Editing

Closest π-Graph

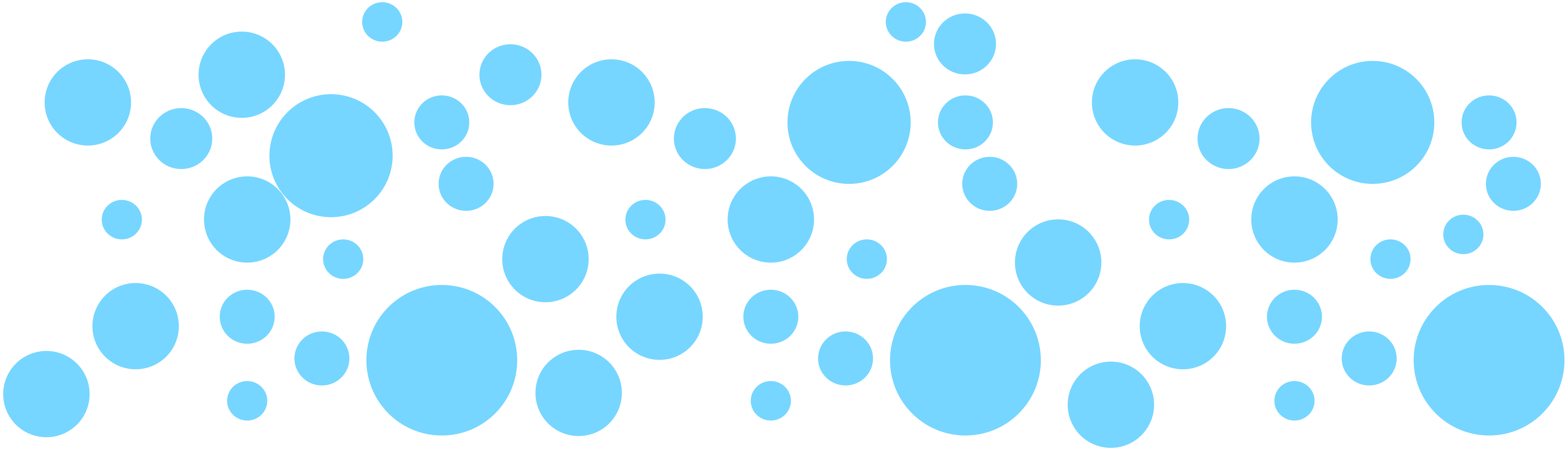
Minimum π-Vertex Deletion

Maximum π-Induced Subgraph

Completion to Minimum Max-Clique

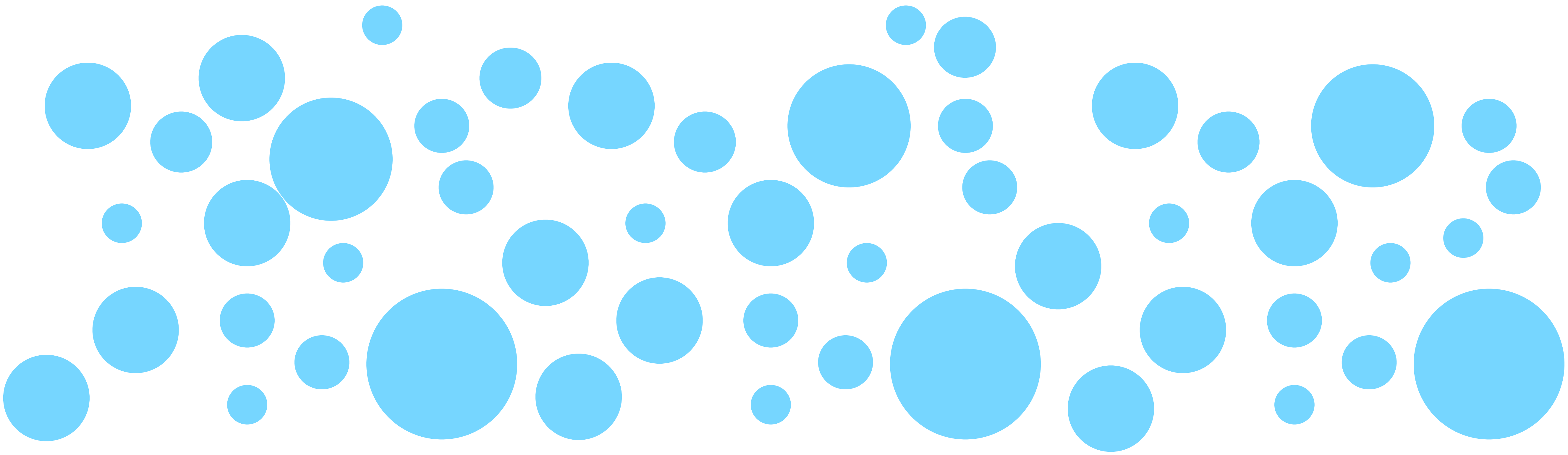
Modification with Restrictions, eg, the Sandwich problem

Restricted Classes of Input



Graph Class or “property”, π

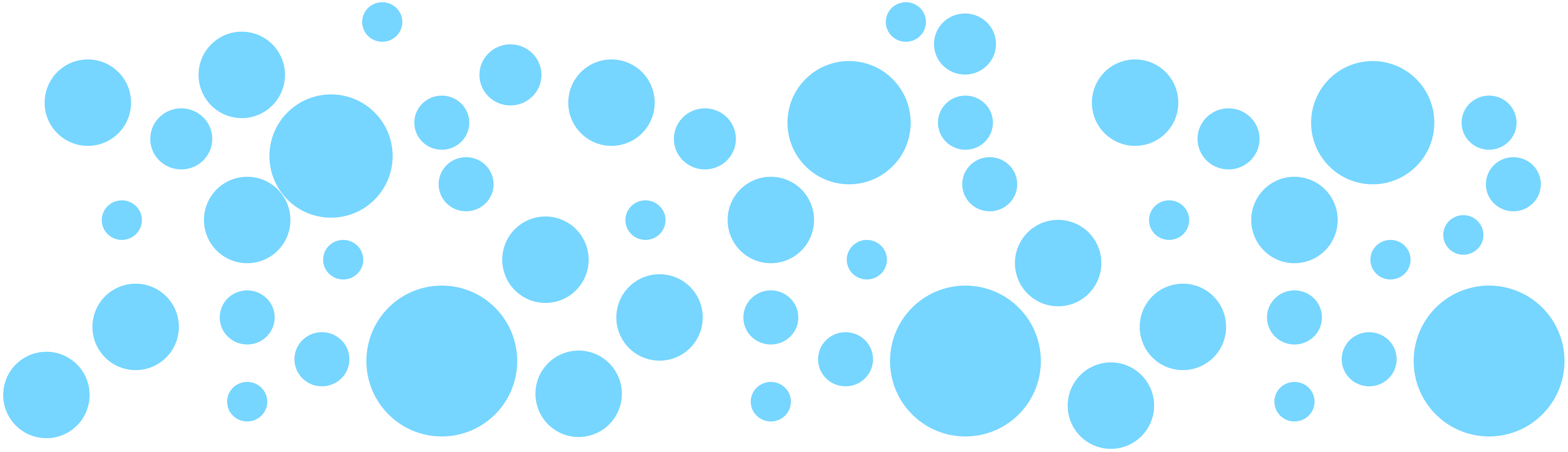
<u>Non-Trivial</u> **π** admits infinitely many graphs, and also excludes infinitely many graphs.



Graph Class or "property", π

Non-Trivial **π** admits infinitely many graphs, and also excludes infinitely many graphs.

Monotone If G belongs to **π**, then all **subgraphs** of G also belong to **π**.

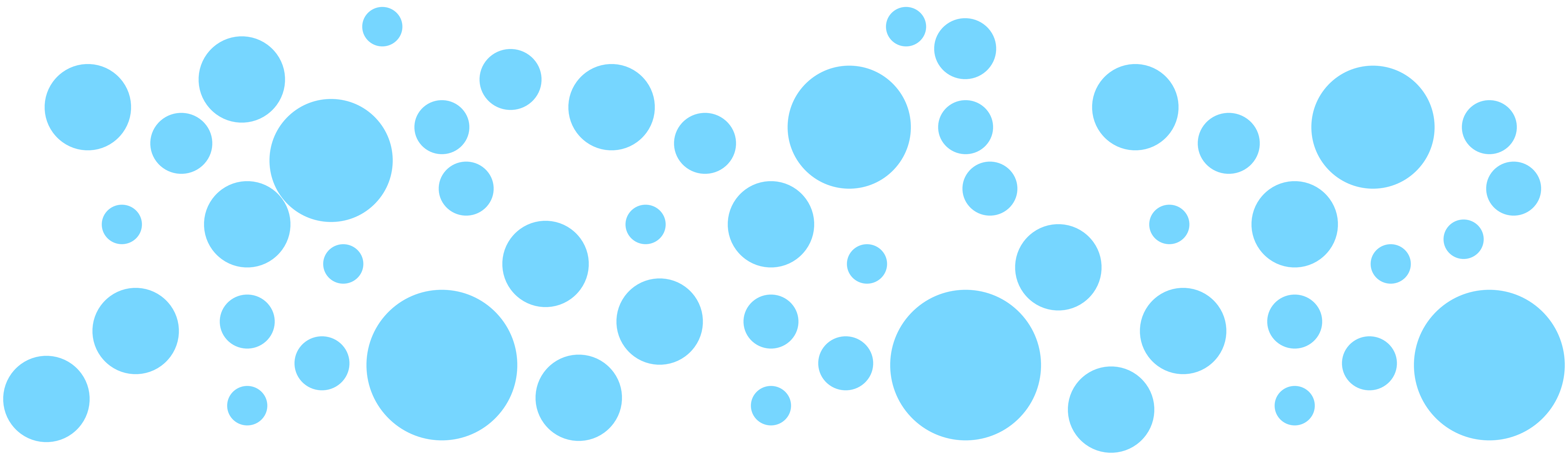


Graph Class or "property", π

Non-Trivial **π** admits infinitely many graphs, and also excludes infinitely many graphs.

Monotone If G belongs to **π**, then all **subgraphs** of G also belong to **π**.

Hereditary If G belongs to **π**, then all **induced subgraphs** of G also belong to **π**.



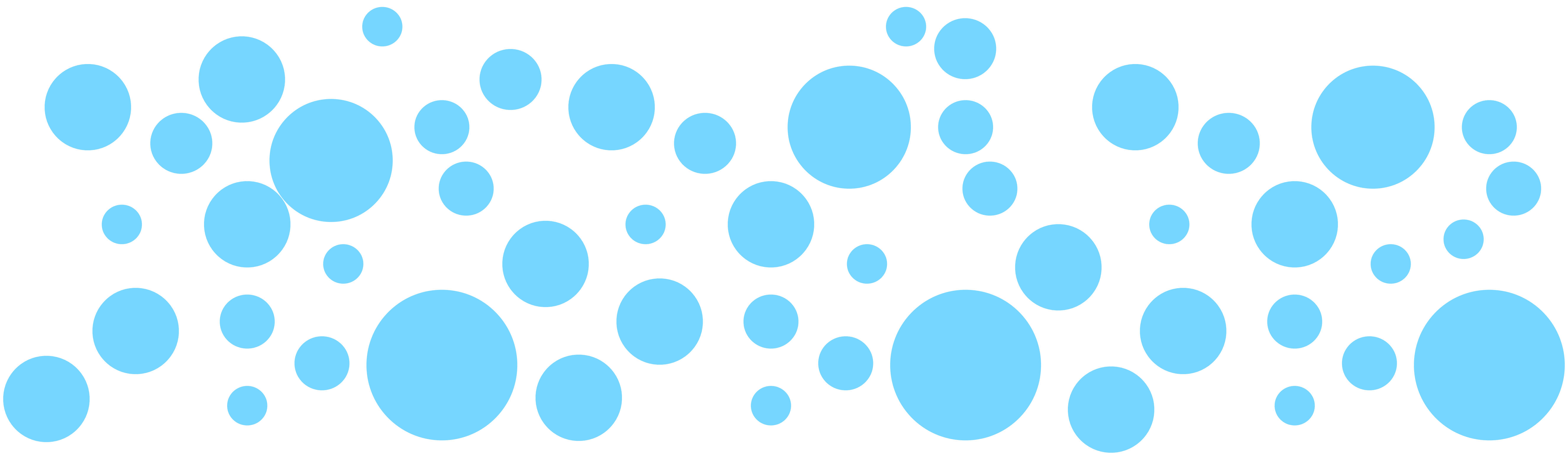
Graph Class or “property”, π

<u>Non-Trivial</u> **π** admits infinitely many graphs, and also excludes infinitely many graphs.

<u>Monotone</u> If G belongs to **π**, then all **subgraphs** of G also belong to **π**.

(Bipartite, Planar)

<u>Hereditary</u> If G belongs to **π**, then all **induced subgraphs** of G also belong to **π**.



Graph Class or “property”, π

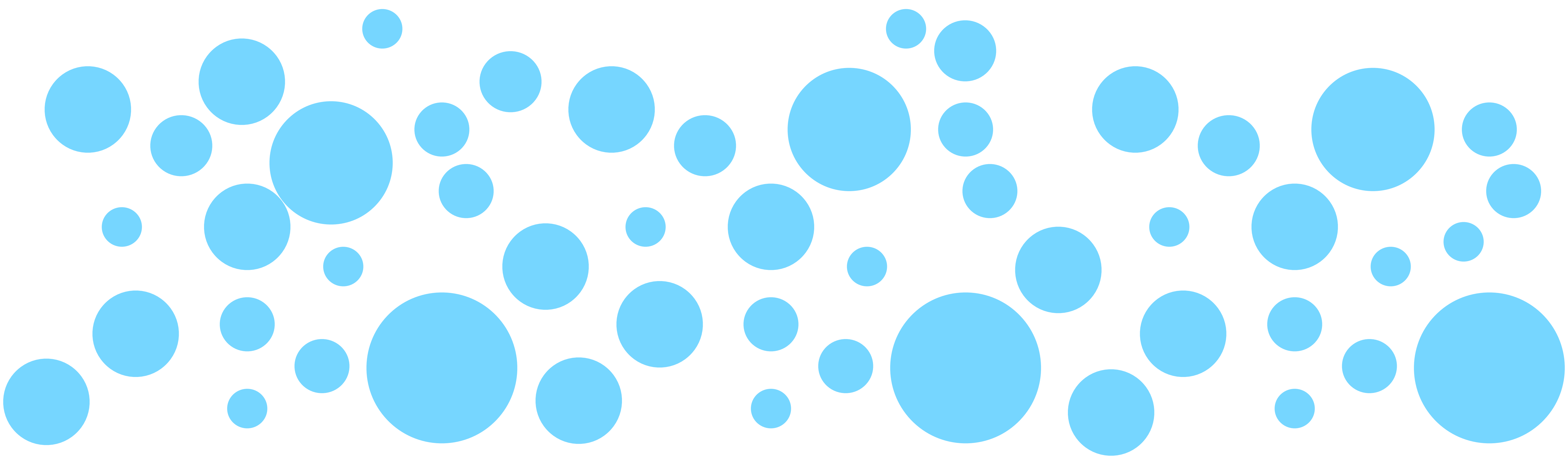
Non-Trivial π admits infinitely many graphs, and also excludes infinitely many graphs.

Monotone If G belongs to π, then all **subgraphs** of G also belong to π.

(Bipartite, Planar)

Hereditary If G belongs to π, then all **induced subgraphs** of G also belong to π.

(Complete graphs, Forests)



Graph Class or “property”, π

Non-Trivial π admits infinitely many graphs, and also excludes infinitely many graphs.

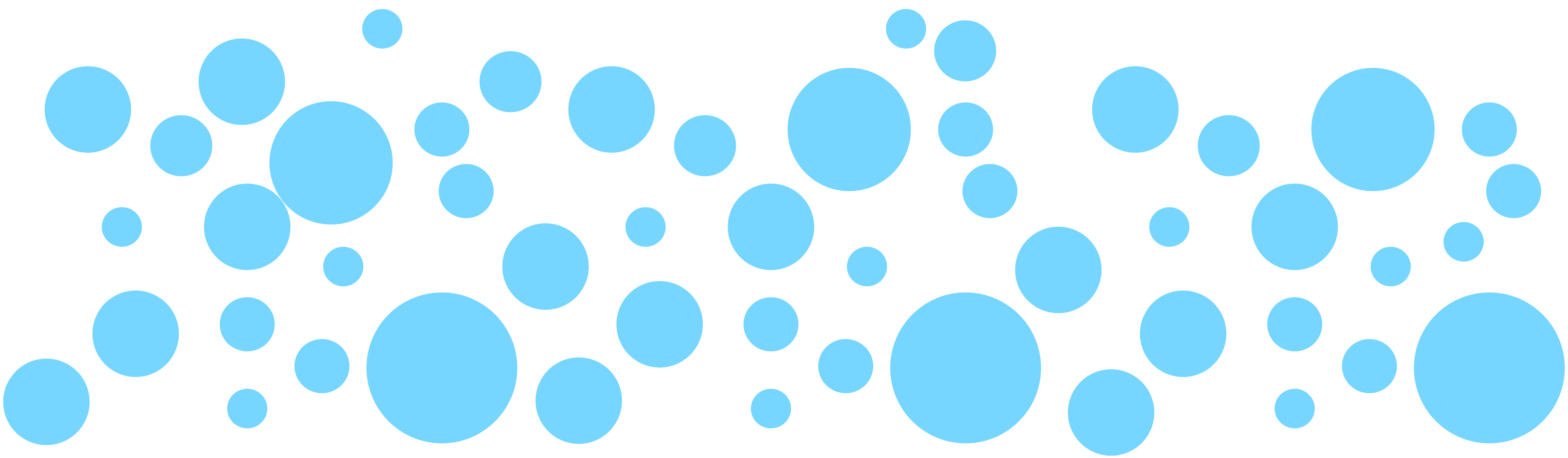
Monotone If G belongs to **π**, then all **subgraphs** of G also belong to **π**.

(Bipartite, Planar)

Hereditary If G belongs to **π**, then all **induced subgraphs** of G also belong to **π**.

(Complete graphs, Forests)

Connectivity, Biconnectivity, Trees, Stars, Eulerian, etc.



Graph Class or “property”, π

Lewis & Yannakakis [1980]

The vertex-deletion problem is **NP-complete** for non-trivial, hereditary properties.

Lewis & Yannakakis [1980]	The vertex-deletion problem is **NP-complete** for non-trivial, hereditary properties.
Yannakakis [1979]	The *connected* vertex-deletion problem is **NP-complete** for non-trivial properties that hold on connected induced subgraphs.

Lewis & Yannakakis [1980]

The vertex-deletion problem is **NP-complete** for non-trivial, hereditary properties.

(Edgeless, Acyclic, etc.)

Yannakakis [1979]

The *connected* vertex-deletion problem is **NP-complete** for non-trivial properties that hold on connected induced subgraphs.

Lewis & Yannakakis [1980]

The vertex-deletion problem is **NP-complete** for non-trivial, hereditary properties.

(Edgeless, Acyclic, etc.)

Yannakakis [1979]

The *connected* vertex-deletion problem is **NP-complete** for non-trivial properties that hold on connected induced subgraphs.

(Trees, Stars, etc.)

Edge-Deletion or Completion problems are sometimes easier than their vertex-deletion counterparts.

Edge-Deletion or Completion problems are sometimes easier than their vertex-deletion counterparts.

Add edges to make the input graph a cluster graph.

Edge-Deletion or Completion problems are sometimes easier than their vertex-deletion counterparts.

Add edges to make the input graph a cluster graph.

Remove edges to make the input graph edgeless.

Edge-Deletion or Completion problems are sometimes easier than their vertex-deletion counterparts.

Add edges to make the input graph a cluster graph.

Remove edges to make the input graph edgeless.

Remove edges to make the input graph acyclic.

Edge-Deletion or Completion problems are sometimes easier than their vertex-deletion counterparts.

Add edges to make the input graph a cluster graph.

Remove edges to make the input graph edgeless.

Remove edges to make the input graph acyclic.

Remove edges to make the input graph bipartite.

Edge-Deletion or Completion problems are sometimes easier than their vertex-deletion counterparts.

Add edges to make the input graph a cluster graph.

Remove edges to make the input graph edgeless.

Remove edges to make the input graph acyclic.

Edge-Deletion or Completion problems are sometimes easier than their vertex-deletion counterparts.

Add edges to make the input graph a cluster graph.

Remove edges to make the input graph edgeless.

Remove edges to make the input graph acyclic.

Alon, Shapira & Sudakov [2009]

Given a property π such that π holds for every bipartite graph, the Minimum π-Deletion problem is NP-hard.

Edge-Deletion or Completion problems are sometimes easier than their vertex-deletion counterparts.

Add edges to make the input graph a cluster graph.

Remove edges to make the input graph edgeless.

Remove edges to make the input graph acyclic.

Alon, Shapira & Sudakov [2009]

Given a property **π** such that **π** holds for every bipartite graph, the Minimum **π**-Deletion problem is NP-hard.

(Triangle-free)

Edge-Deletion or Completion problems are sometimes easier than their vertex-deletion counterparts.

Add edges to make the input graph a cluster graph.

Remove edges to make the input graph edgeless.

Remove edges to make the input graph acyclic.

<u>Alon, Shapira & Sudakov</u> [2009]

Given a property π such that π holds for every bipartite graph, the Minimum π-Deletion problem is NP-hard.

(Triangle-free)

<u>Colbourn & El-Mallah</u> [1988]

Making a graph P_k-free by deleting the minimum number of edges is NP-hard for every $k > 2$.

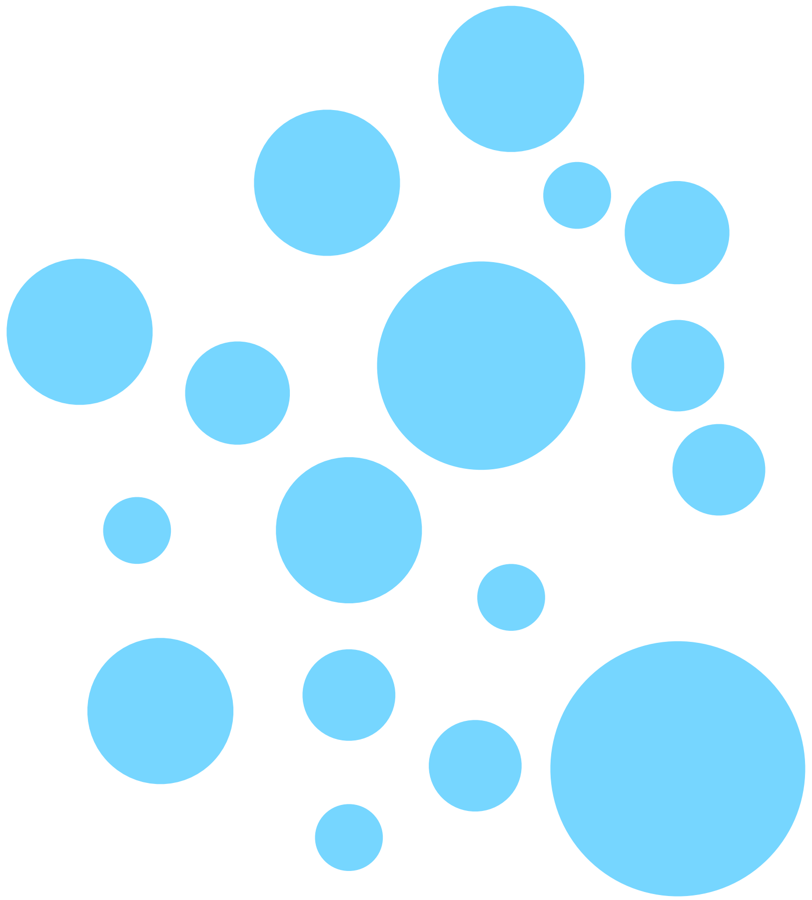
edge editing

edge contractions

vertex deletions

edge additions

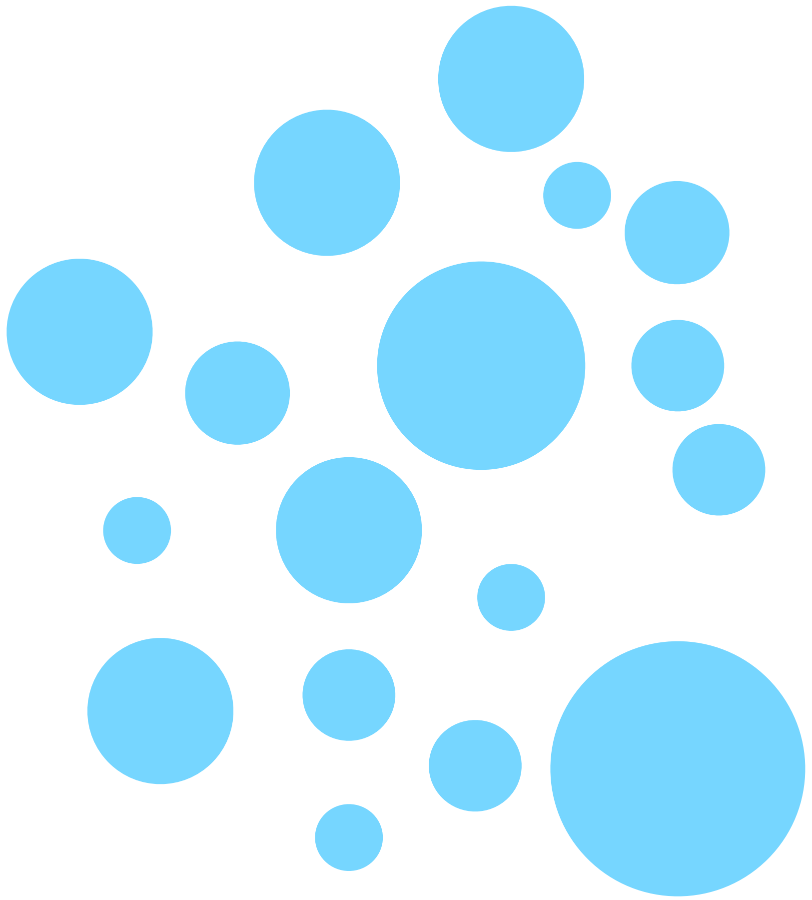
edge deletions



Graph Class or “property”, π

Perfect
Trivially Perfect
Cographs
Permutation Graphs
Bipartite
Trees
Weakly Chordal
Chordal
Strongly Chordal
Split
Interval
Proper Interval
Unit Interval
Forests
Caterpillars
Chain
Chordal Bipartite
Distance Hereditary
Comparability
Trapezoid
CoChordal
Circle
Planar

edge editing
edge contractions
vertex deletions
edge additions
edge deletions



Graph Class or “property”, π

Graph Modification Problems

A Modern Perspective

Graph Modification Problems

Why bother?

Special Graph Classes – correspond to some desirable property in real-world data.

Special Graph Classes – correspond to some desirable property in real-world data.

Planar Graphs

Interval Graphs

Cluster Graphs

k-Connected Graphs

Directed Acyclic Graphs

Chordal Graphs

Special Graph Classes – correspond to some desirable property in real-world data.

Planar Graphs	Graph Drawing
Interval Graphs	Physical Mapping of DNA
Cluster Graphs	Correlation Clustering
k-Connected Graphs	Reliability of Networks
Directed Acyclic Graphs	Deadlock Recovery, Operating Systems
Chordal Graphs	Gaussian Elimination over Sparse Matrices

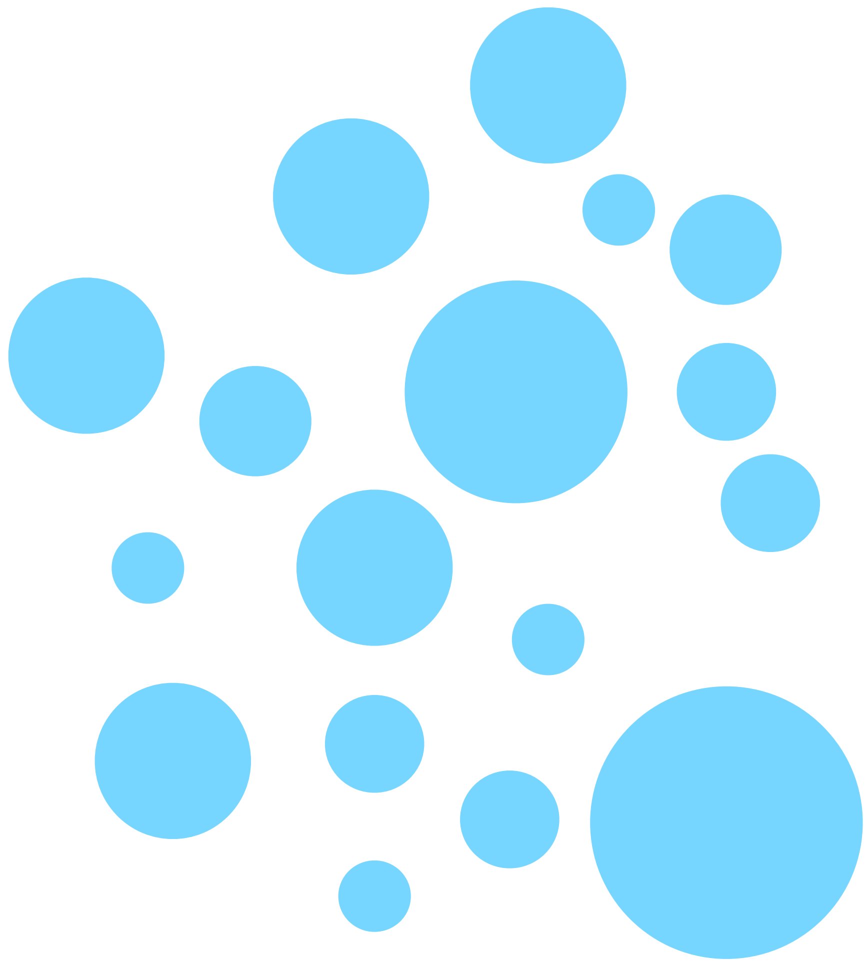
Special Graph Classes – correspond to some desirable property in real-world data.

We would like to "achieve" these properties with minimum cost, which naturally leads us to the graph modification framework.

Planar Graphs	Graph Drawing
Interval Graphs	Physical Mapping of DNA
Cluster Graphs	Correlation Clustering
k-Connected Graphs	Reliability of Networks
Directed Acyclic Graphs	Deadlock Recovery, Operating Systems
Chordal Graphs	Gaussian Elimination over Sparse Matrices

Graph Modification Problems

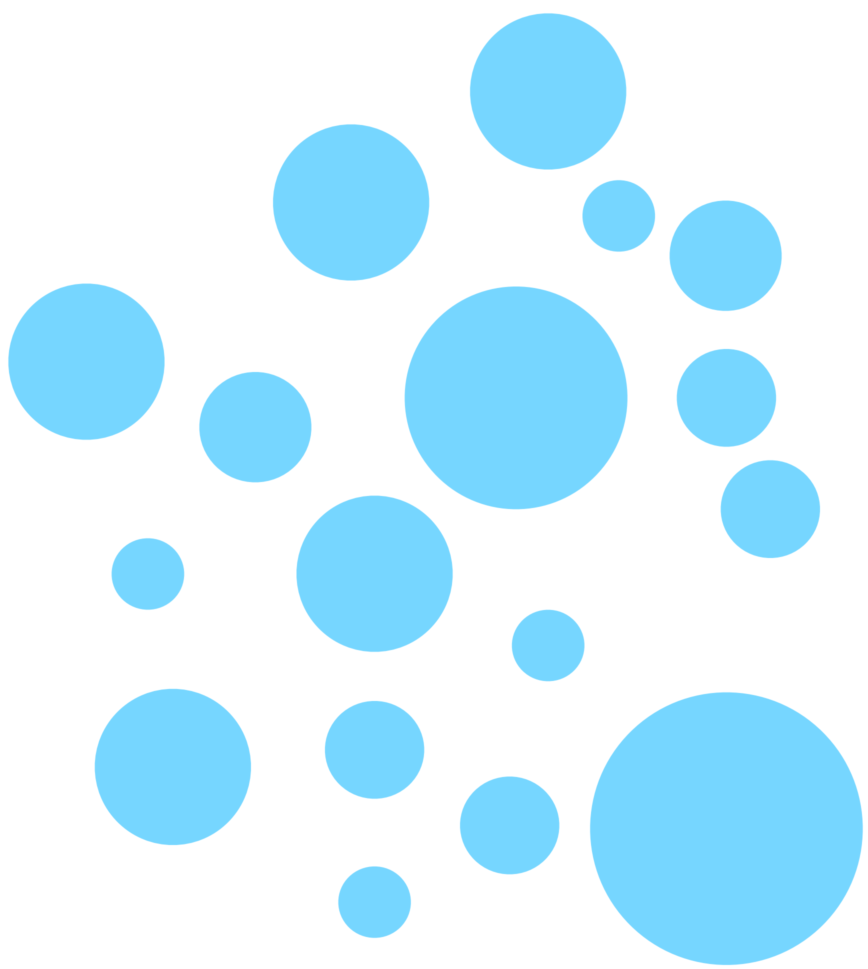
Algorithms



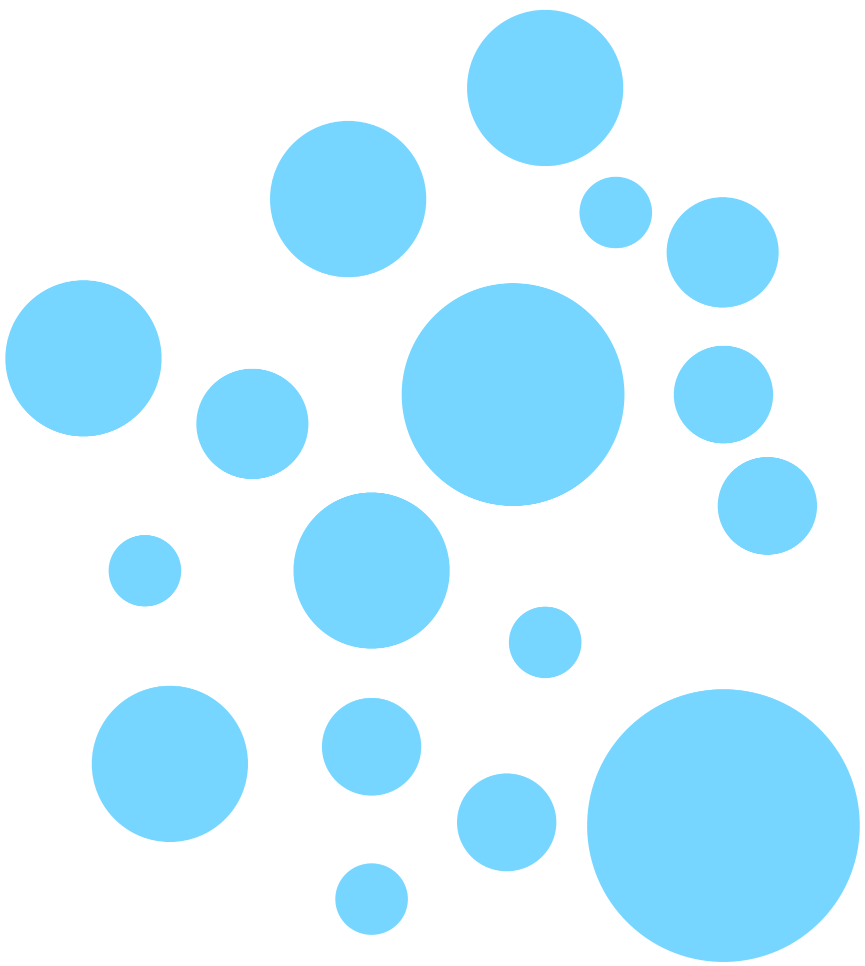
Graph Class or "property", π

Forbidden Subgraph Characterization

A graph G belongs to **π** if, and only if,
G does not contain any graph from Forb(**π**)
as an **induced subgraph**.



Graph Class or "property", π



Graph Class or "property", π

Forbidden Subgraph Characterization

A graph G belongs to **π** if, and only if,
G does not contain any graph from Forb(**π**)
as an **induced subgraph**.

Forbidden Minor Characterization

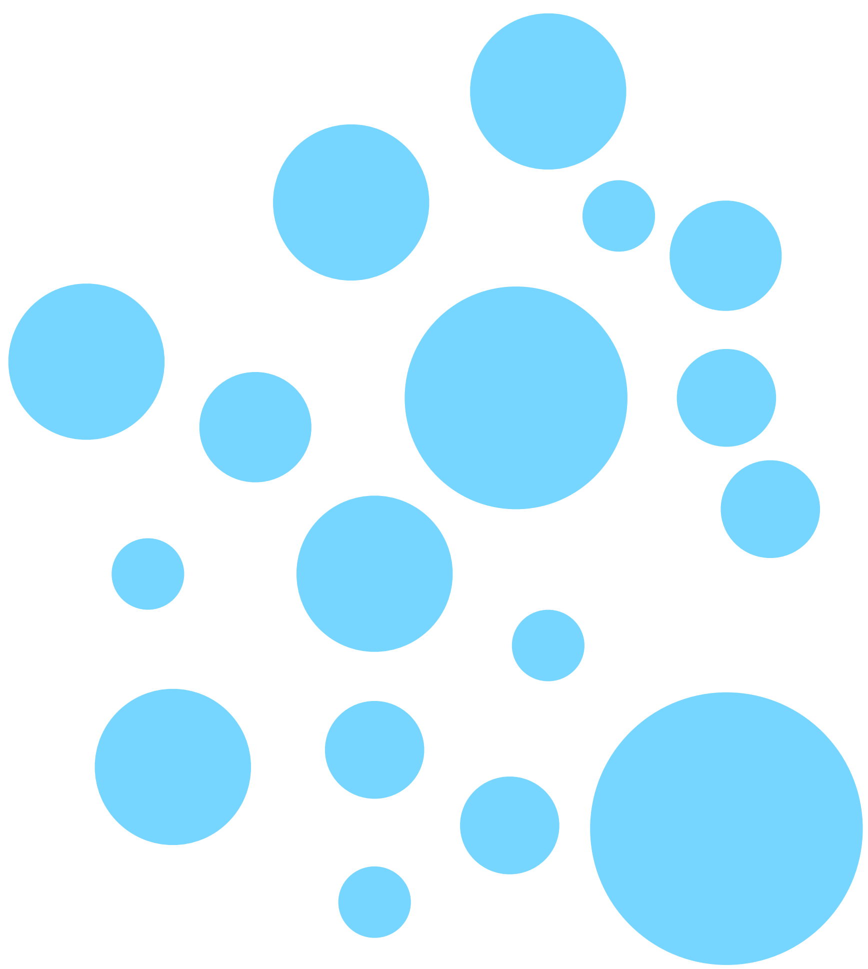
A graph G belongs to **π** if, and only if,
G does not contain any graph from Forb(**π**)
as a **minor**.

Forbidden Subgraph Characterization

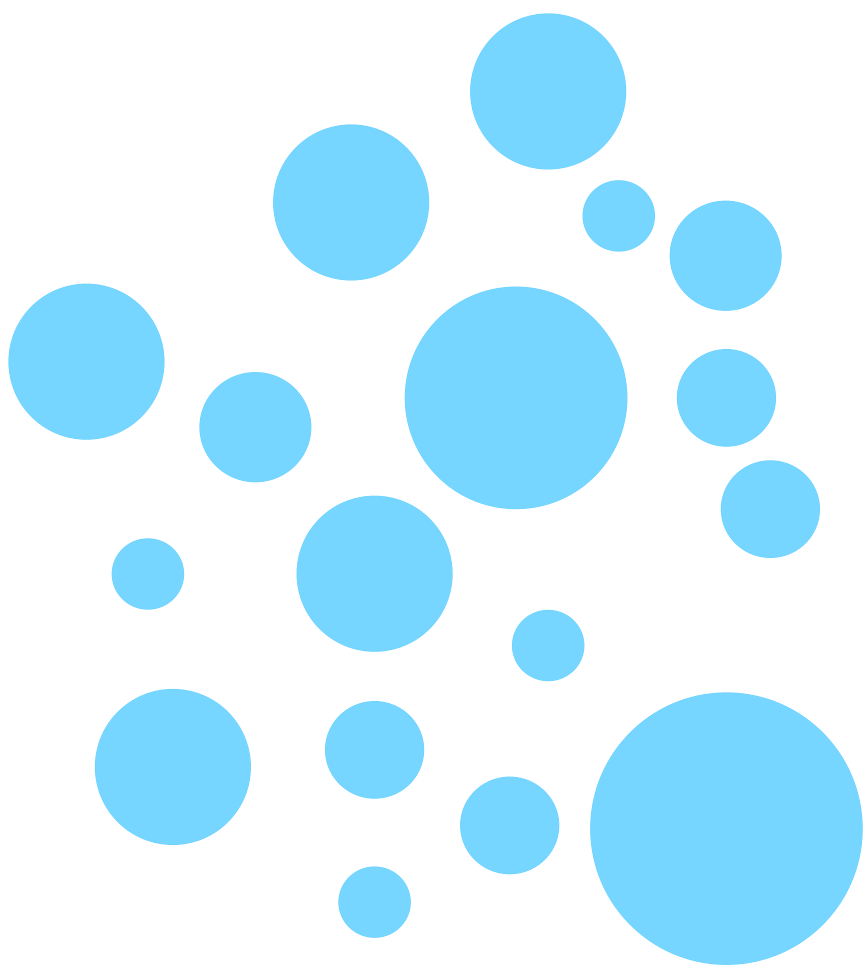
A graph admits a **forbidden subgraph characterization**
if, and only if,
it is **closed under taking induced subgraphs (hereditary)**,
that is, the property is preserved under vertex deletions.

Forbidden Minor Characterization

A graph G belongs to **π** if, and only if,
G does not contain any graph from Forb(**π**)
as a **minor**.



Graph Class or "property", π



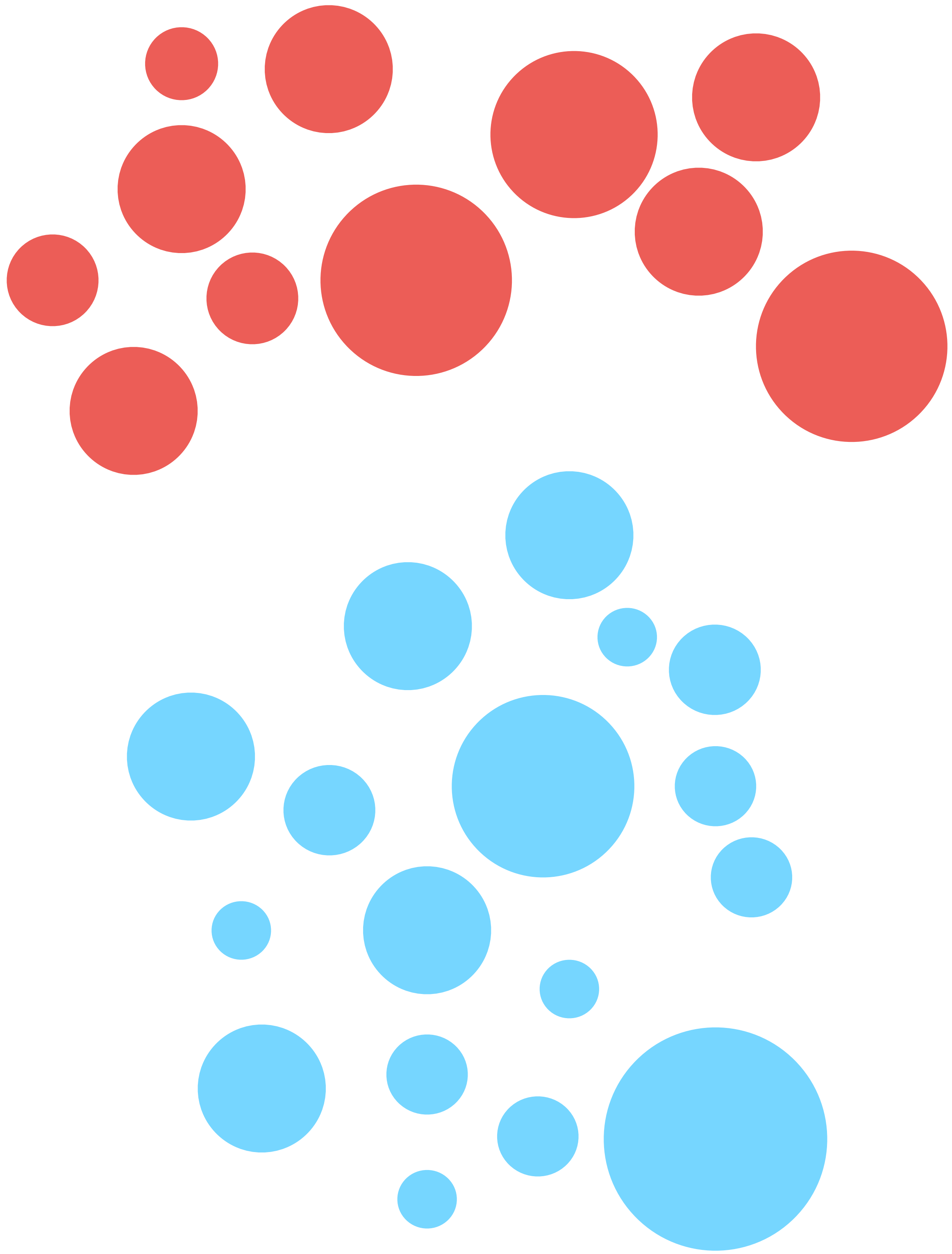
Graph Class or “property”, π

Forbidden Subgraph Characterization

A graph admits a **forbidden subgraph characterization**
if, and only if,
it is **closed under taking induced subgraphs (hereditary)**,
that is, the property is preserved under vertex deletions.

Forbidden Minor Characterization

A graph admits a **forbidden minor characterization**
if, and only if,
it is **closed under taking minors**,
that is, the property is preserved under vertex deletions,
edge deletions and edge contractions.



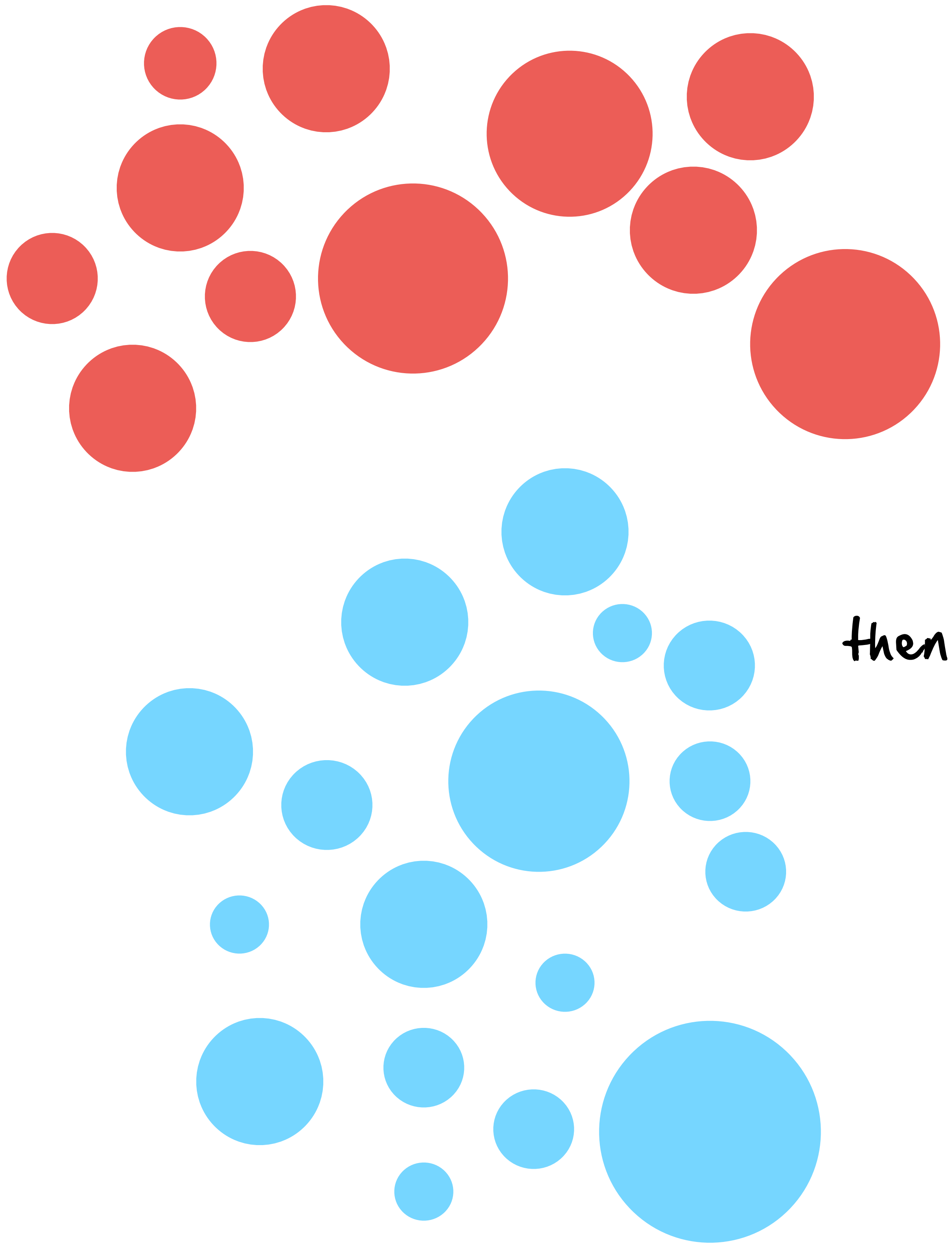
Graph Class or “property”, π

Forbidden Subgraph Characterization

A graph admits a **forbidden subgraph characterization**
if, and only if,
it is **closed under taking induced subgraphs (hereditary)**,
that is, the property is preserved under vertex deletions.

Forbidden Minor Characterization

A graph admits a **forbidden minor characterization**
if, and only if,
it is **closed under taking minors**,
that is, the property is preserved under vertex deletions,
edge deletions and edge contractions.



Graph Class or "property", π

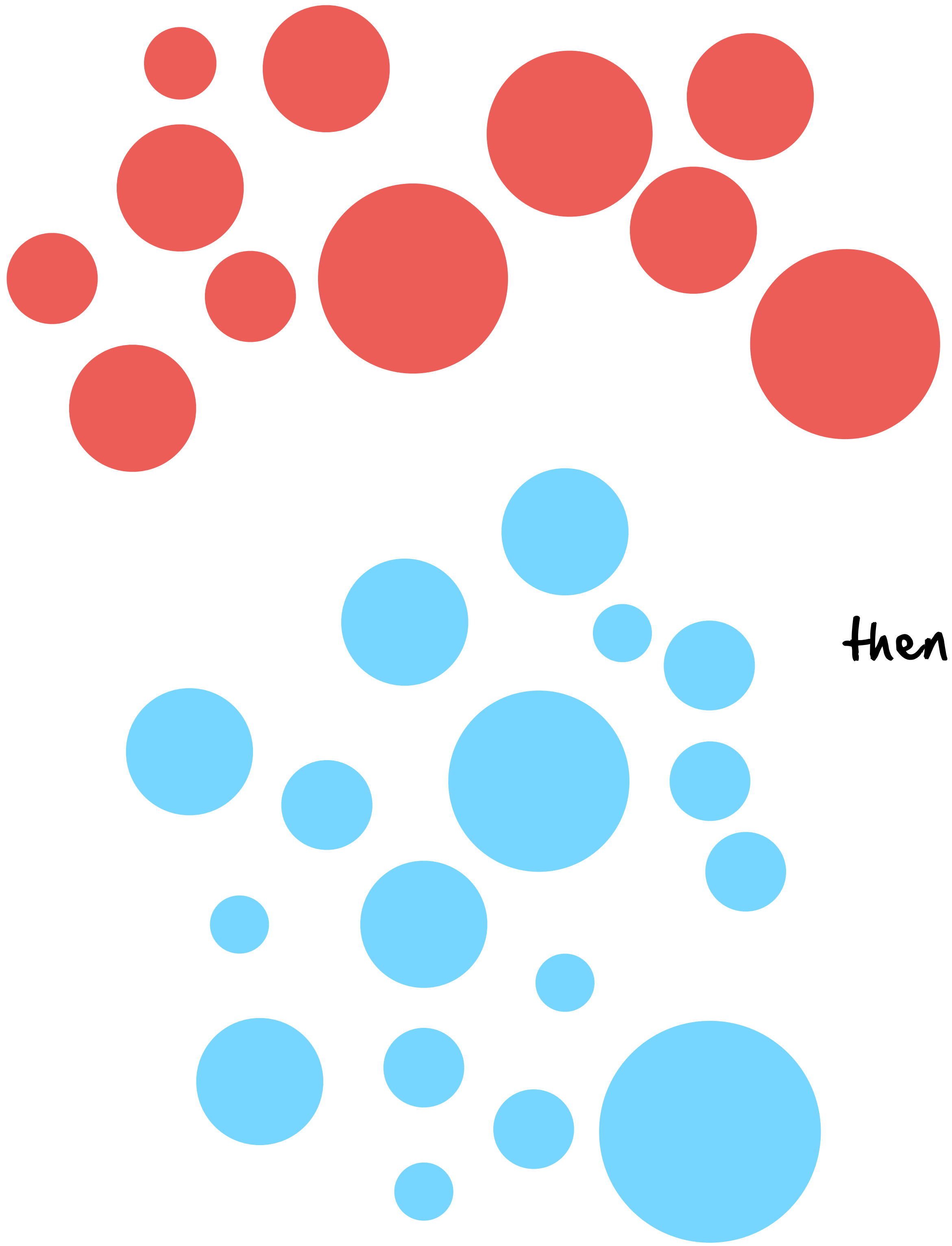
Forbidden Subgraph Characterization

A graph admits a **forbidden subgraph characterization**
if, and only if,
it is **closed under taking induced subgraphs (hereditary)**,
that is, the property is preserved under vertex deletions.

If there are finitely many forbidden graphs, then the corresponding graph modification problem is FPT.

Forbidden Minor Characterization

A graph admits a **forbidden minor characterization**
if, and only if,
it is **closed under taking minors**,
that is, the property is preserved under vertex deletions, edge deletions and edge contractions.



Graph Class or “property”, π

Forbidden Subgraph Characterization

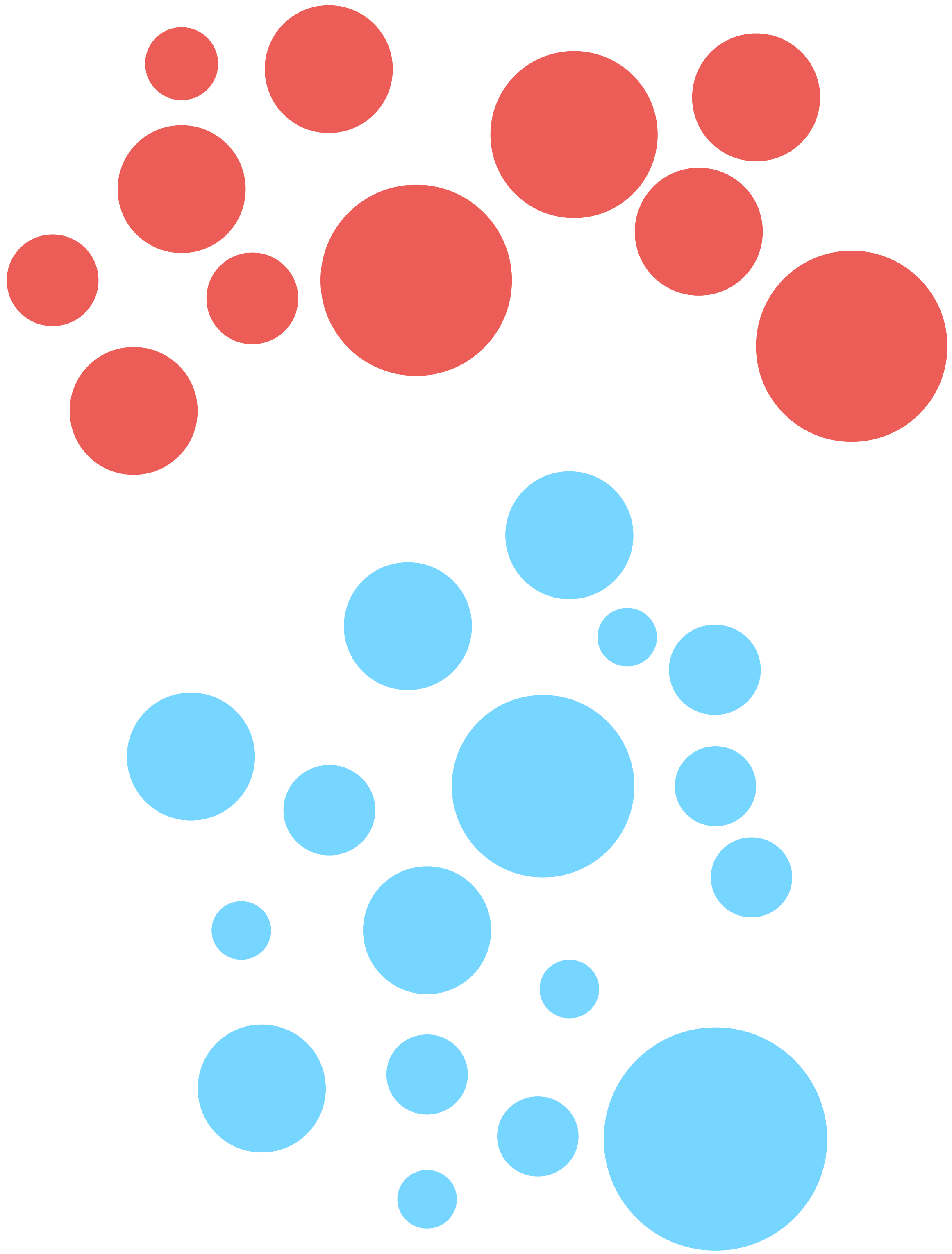
A graph admits a **forbidden subgraph characterization**
if, and only if,
it is **closed under taking induced subgraphs (hereditary)**,
that is, the property is preserved under vertex deletions.

If there are finitely many forbidden graphs,
then the corresponding graph modification problem is FPT.

Forbidden Minor Characterization

A graph admits a **forbidden minor characterization**
if, and only if,
it is **closed under taking minors**,
that is, the property is preserved under vertex deletions,
edge deletions and edge contractions.

If there are finitely many forbidden minors…?

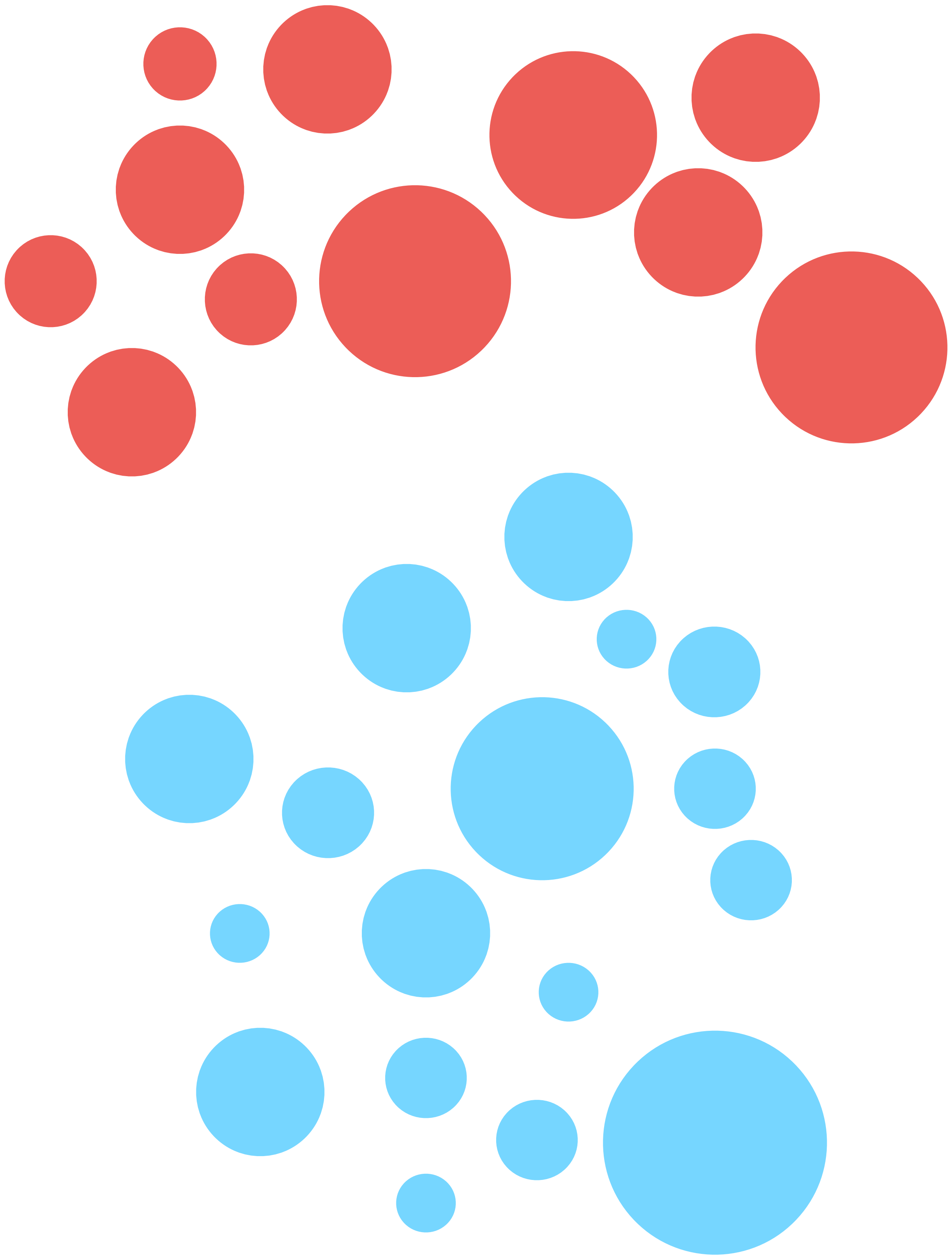


Graph Class or “property”, π

Forbidden Minor Characterization

A graph G belongs to **π** if, and only if,
G does not contain any graph from Forb(**π**)
as a **minor**.

A graph admits a **forbidden minor characterization**
if, and only if,
it is **closed under taking minors**.



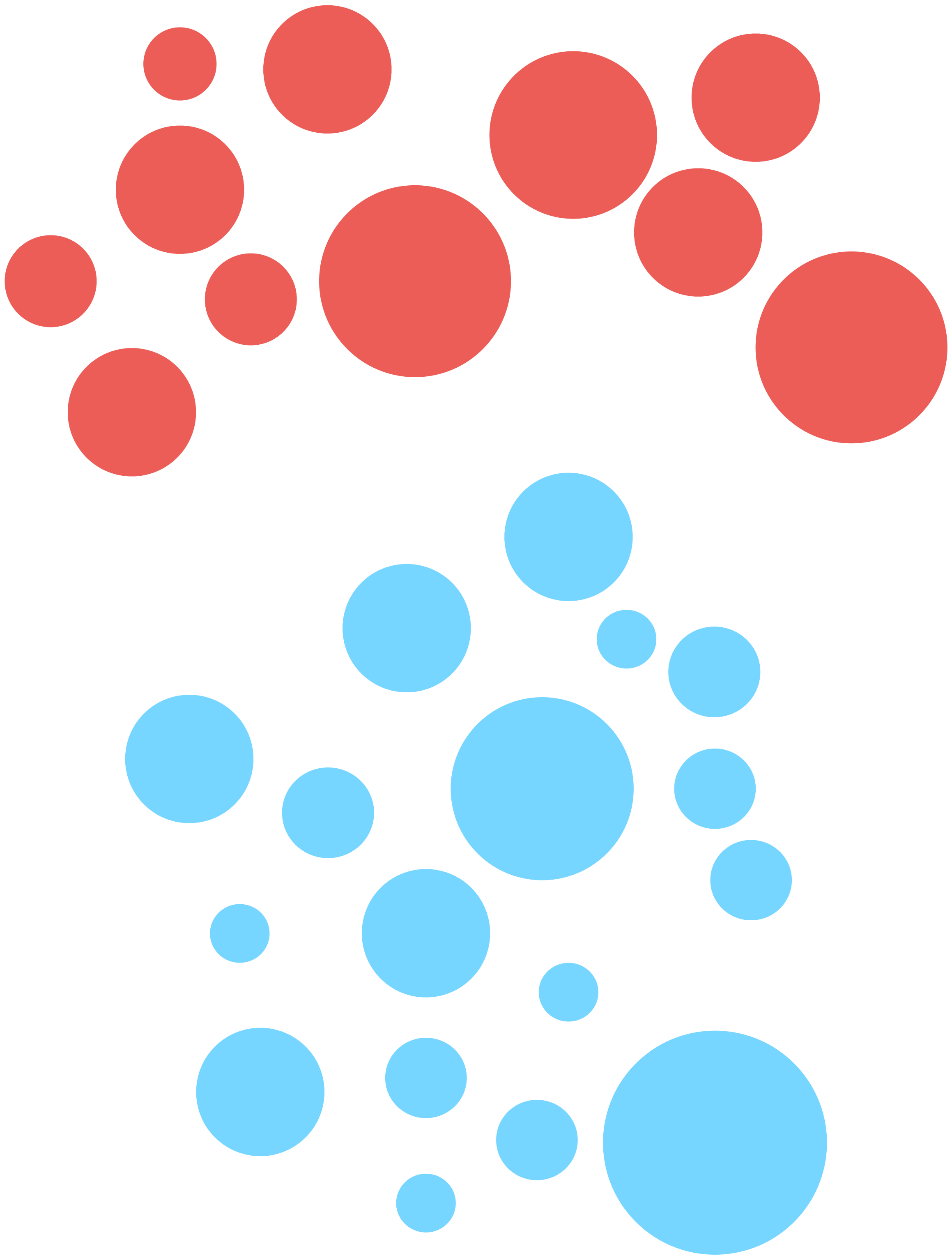
Graph Class or “property”, π

Forbidden Minor Characterization

A graph G belongs to **π** if, and only if,
G does not contain any graph from Forb(**π**)
as a **minor**.

A graph admits a **forbidden minor characterization**
if, and only if,
it is **closed under taking minors**.

In fact, there is always a finite forbidden set!
(Robertson-Seymour; Graph Minors Theorem)



Graph Class or "property", π

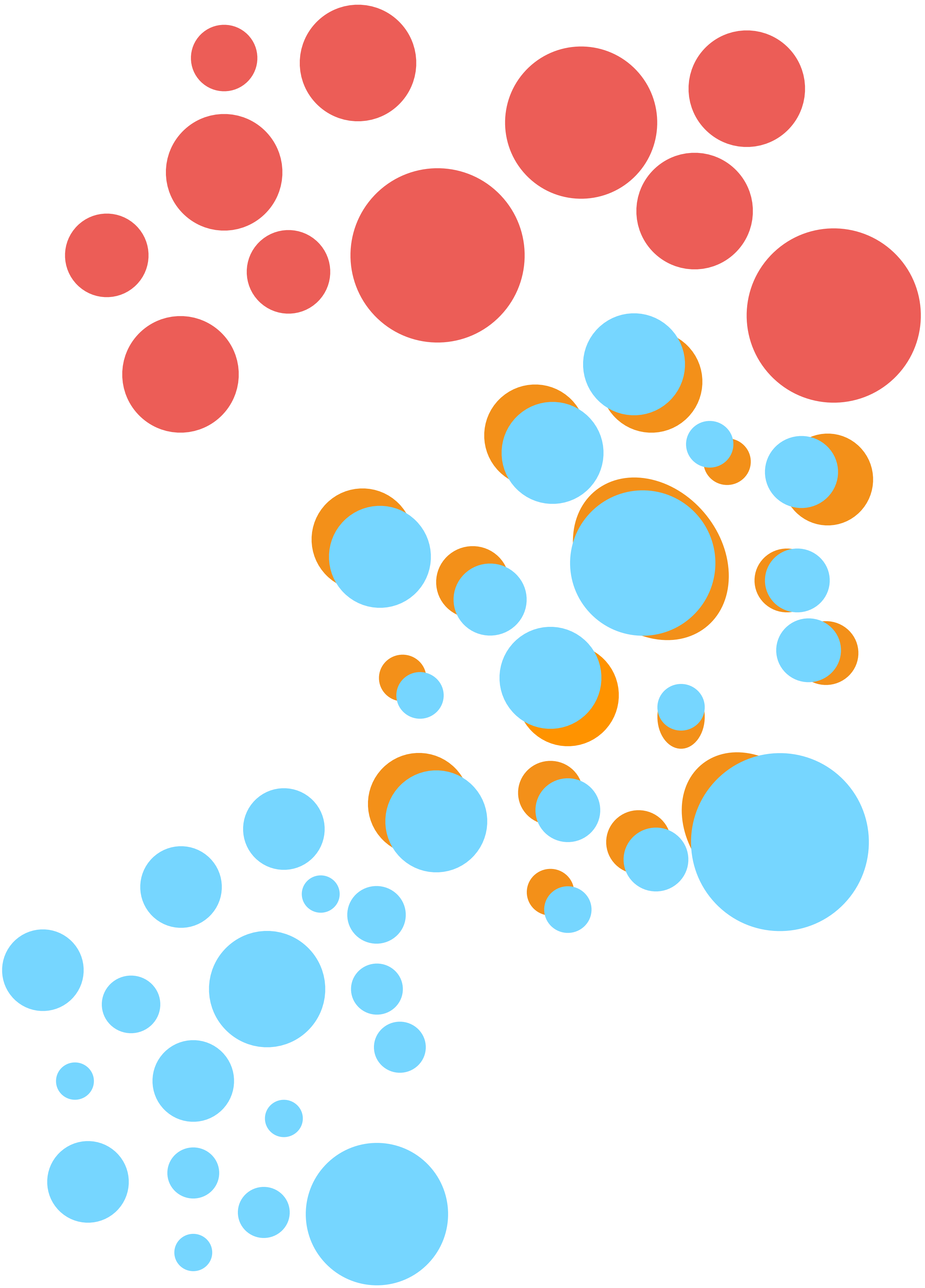
Forbidden Minor Characterization

A graph G belongs to **π** if, and only if,
G does not contain any graph from Forb(**π**)
as a **minor**.

A graph admits a **forbidden minor characterization**
if, and only if,
it is **closed under taking minors**.

In fact, there is always a finite forbidden set!
(Robertson-Seymour; Graph Minors Theorem)

Suppose the class **π + (modifications)** is
closed under taking minors…



Graph Class or “property”, π + (modifications)

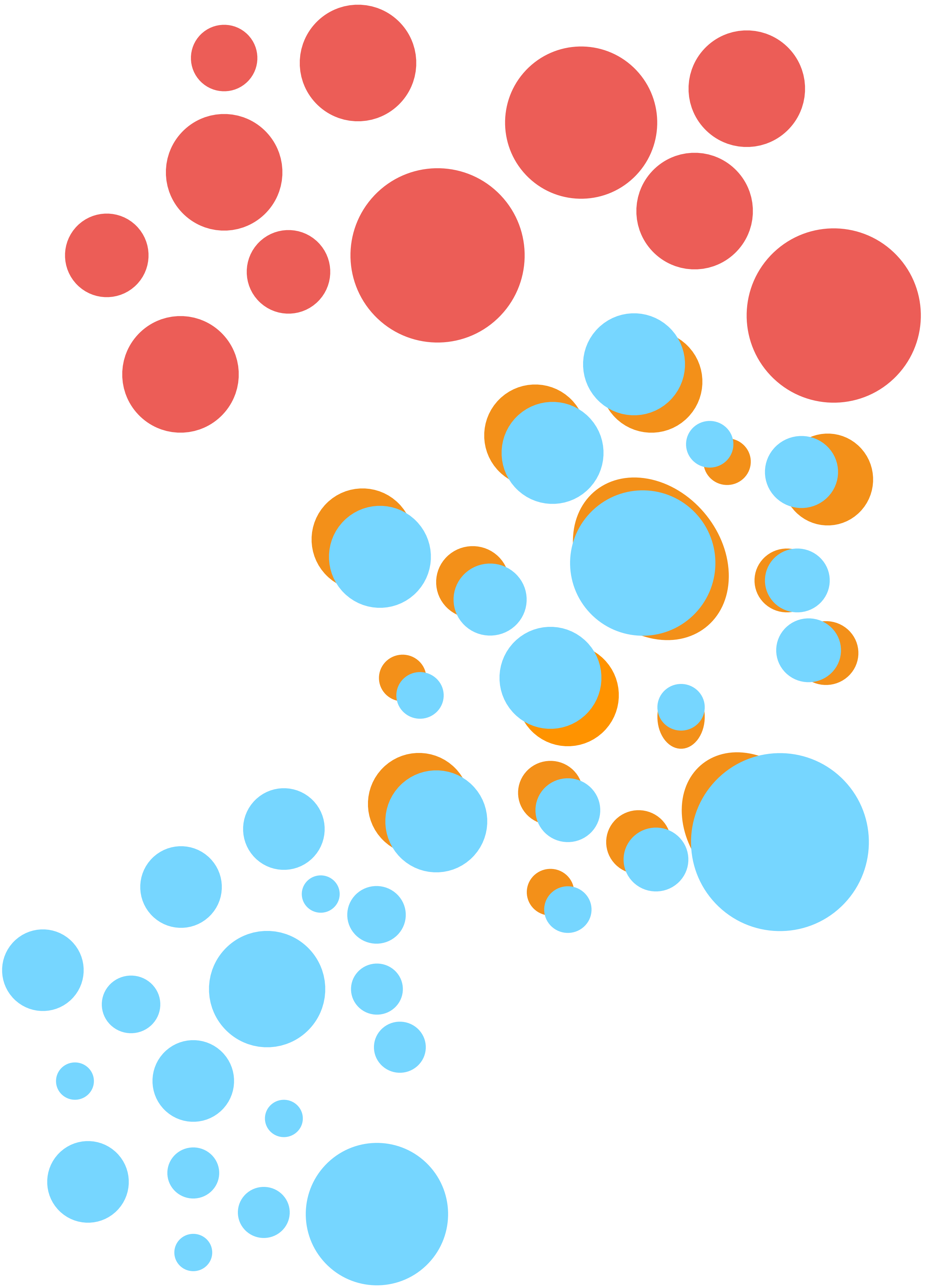
Forbidden Minor Characterization

A graph G belongs to **π** if, and only if,
G does not contain any graph from Forb(**π**)
as a **minor**.

A graph admits a **forbidden minor characterization**
if, and only if,
it is **closed under taking minors**.

In fact, there is always a finite forbidden set!
(Robertson-Seymour; Graph Minors Theorem)

Suppose the class **π + (modifications)** is
closed under taking minors…



Graph Class or "property", π + (modifications)

Forbidden Minor Characterization

A graph G belongs to **π** if, and only if,
G does not contain any graph from Forb(**π**)
as a **minor**.

A graph admits a **forbidden minor characterization**
if, and only if,
it is **closed under taking minors**.

In fact, there is always a finite forbidden set!
(Robertson-Seymour; Graph Minors Theorem)

Suppose the class **π + (modifications)** is
closed under taking minors…

Now: the graph modification problem has boiled down to a membership testing problem, which can be done in cubic time, given the finite forbidden set.

Forbidden Subgraph Characterization

A graph admits a **forbidden subgraph characterization**
if, and only if,
it is **closed under taking induced subgraphs (hereditary)**,
that is, the property is preserved under vertex deletions.

Forbidden Minor Characterization

A graph admits a **forbidden minor characterization**
if, and only if,
it is **closed under taking minors**,
that is, the property is preserved under vertex deletions,
edge deletions and edge contractions.

Forbidden Subgraph Characterization

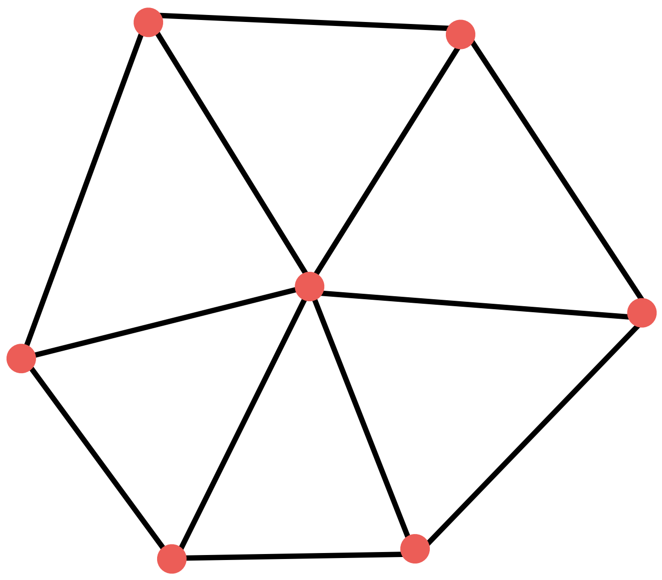
A graph admits a **forbidden subgraph characterization**
if, and only if,
it is **closed under taking induced subgraphs (hereditary)**,
that is, the property is preserved under vertex deletions.

Can the node deletion question for hereditary properties always be answered with the "graph minors hammer"?

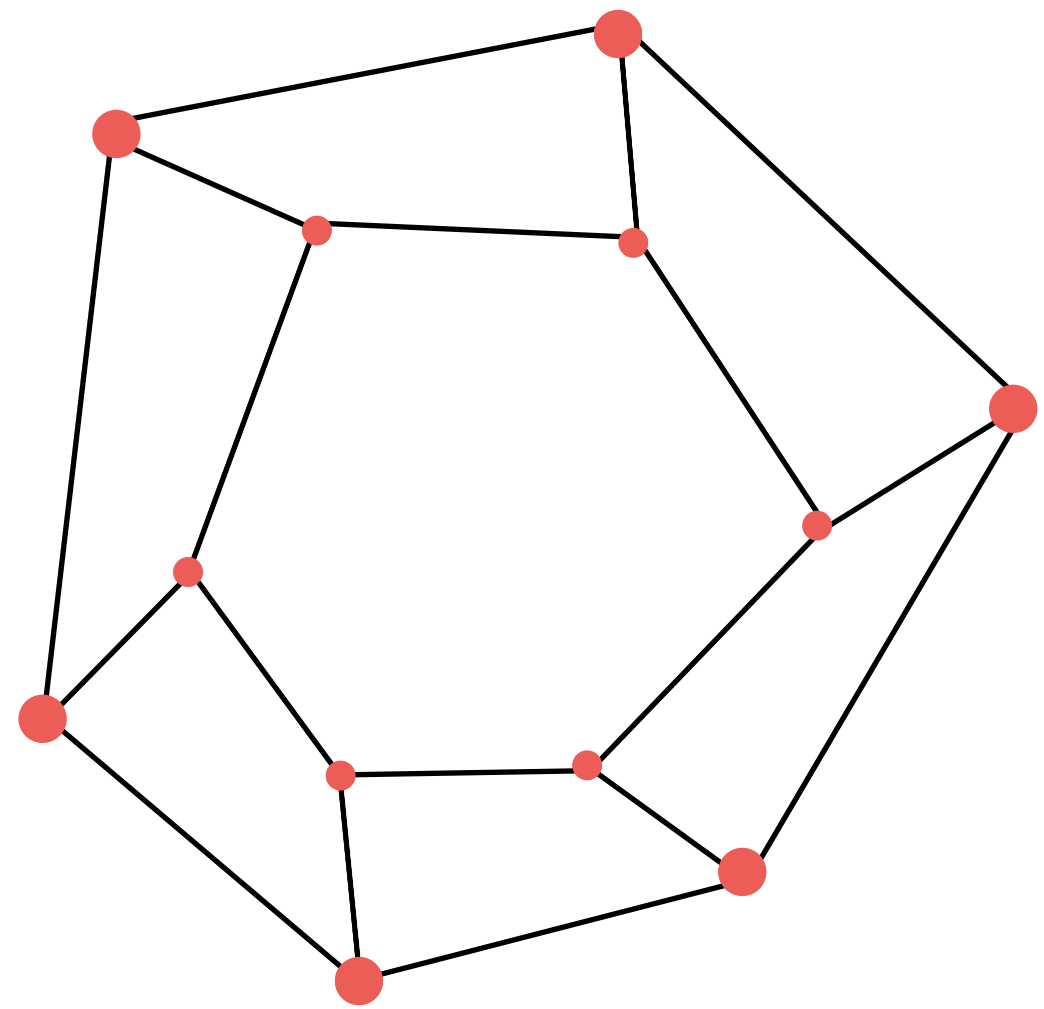
Forbidden Minor Characterization

A graph admits a **forbidden minor characterization**
if, and only if,
it is **closed under taking minors**,
that is, the property is preserved under vertex deletions, edge deletions and edge contractions.

Consider the property π of being wheel-free, that is, not having any wheel as an induced subgraph.

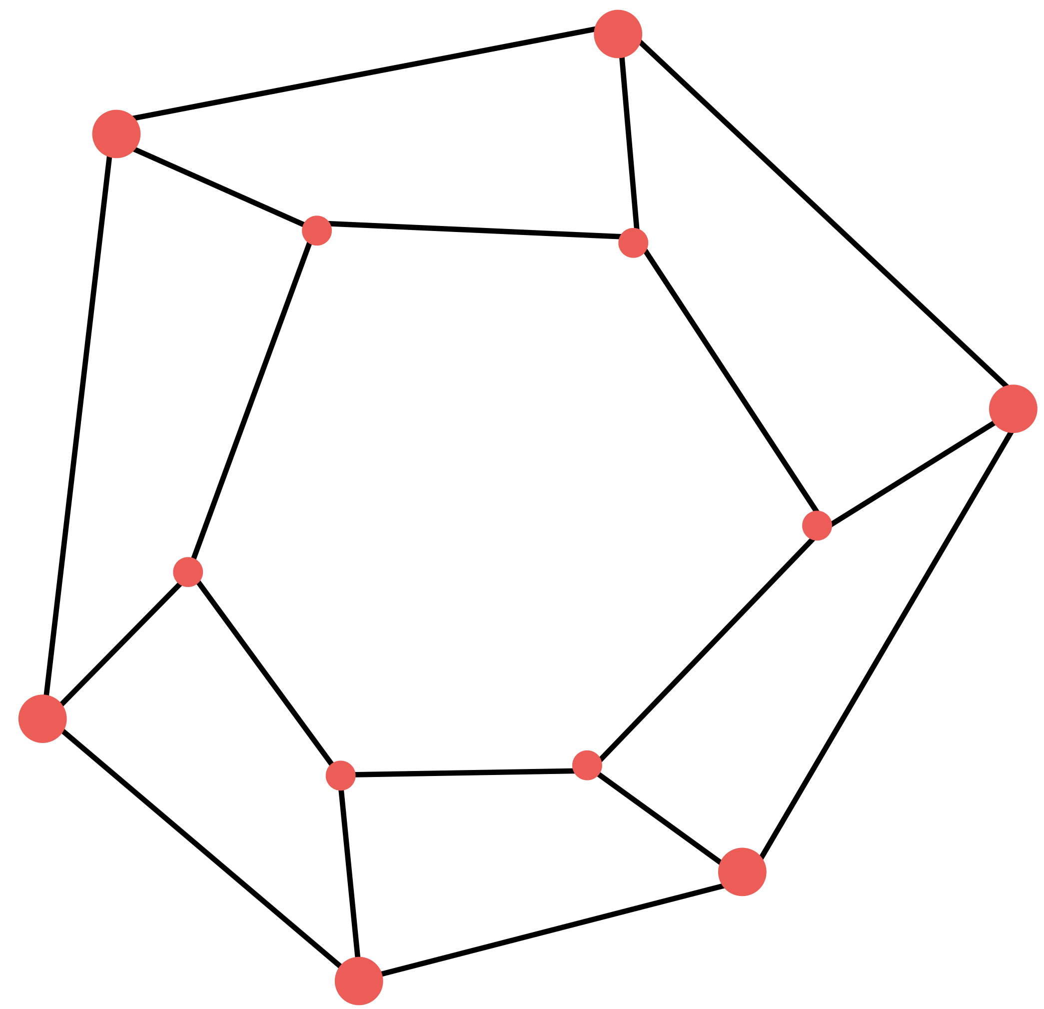


Consider the property **π** of being wheel-free, that is, not having any wheel as an induced subgraph.



The property is hereditary, but **not** minor-closed.

Consider the property **π** of being wheel-free, that is, not having any wheel as an induced subgraph.



Lokshtanov (2008)

Wheel-Free Deletion and Wheel-Free Vertex Deletion are **W[2]-hard.**

Finding large induced subgraphs with property π.

Khot and Raman, (2002)

Finding large induced subgraphs with property **π**.

Khot and Raman, (2002)

If **π** is a hereditary property that contains all independent sets and all cliques,
or if **π** excludes some independent sets and some cliques,
then the problem of finding an induced subgraph of size at least k, satisfying **π**,
is FPT.

Finding large induced subgraphs with property **π**.

Khot and Raman, (2002)

If **π** is a hereditary property that contains all independent sets and all cliques,
or if **π** excludes some independent sets and some cliques,
then the problem of finding an induced subgraph of size at least k, satisfying **π**,
is FPT.

If **π** is a hereditary property contains all independent sets but not all cliques
(or vice versa),
then the problem of finding an induced subgraph of size at least k, satisfying **π**,
is W[1]-hard.

Finding large induced subgraphs with property **π**.

Khot and Raman, (2002)

Ramsey Numbers!

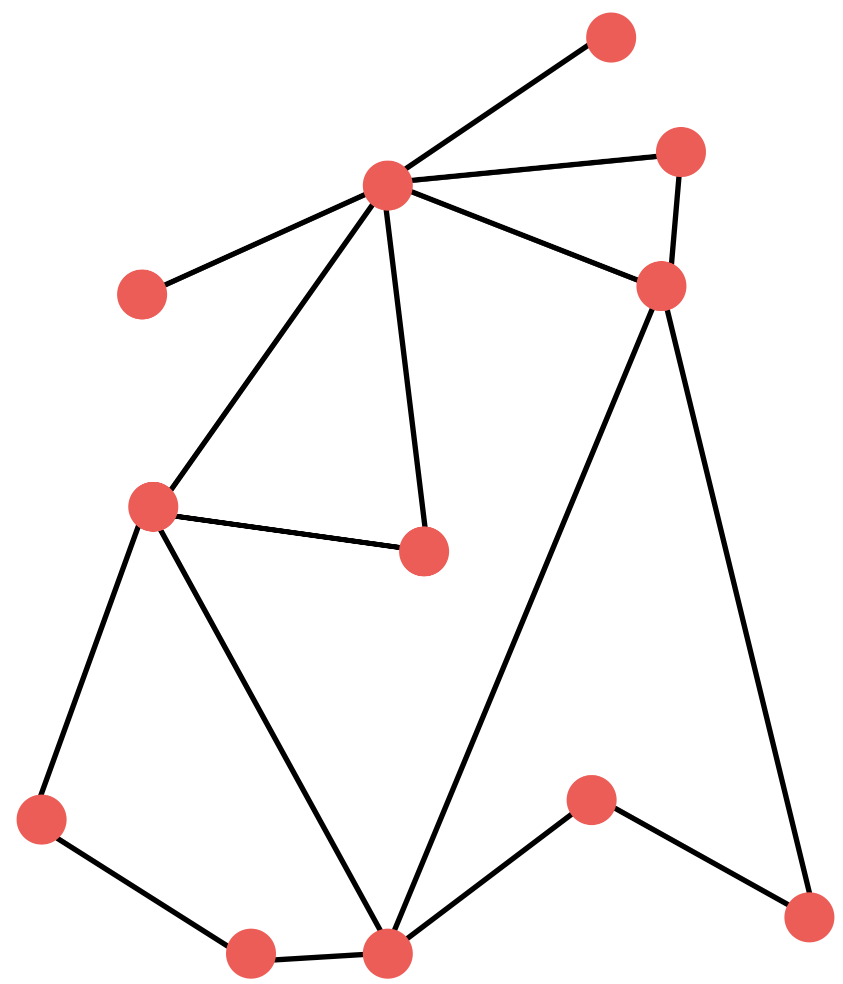
If **π** is a hereditary property that contains all independent sets and all cliques,
or if **π** excludes some independent sets and some cliques,
then the problem of finding an induced subgraph of size at least k, satisfying **π**,
is FPT.

Reduction from Independent Set

If **π** is a hereditary property contains all independent sets but not all cliques
(or vice versa),
then the problem of finding an induced subgraph of size at least k, satisfying **π**,
is W[1]-hard.

Vertex Cover

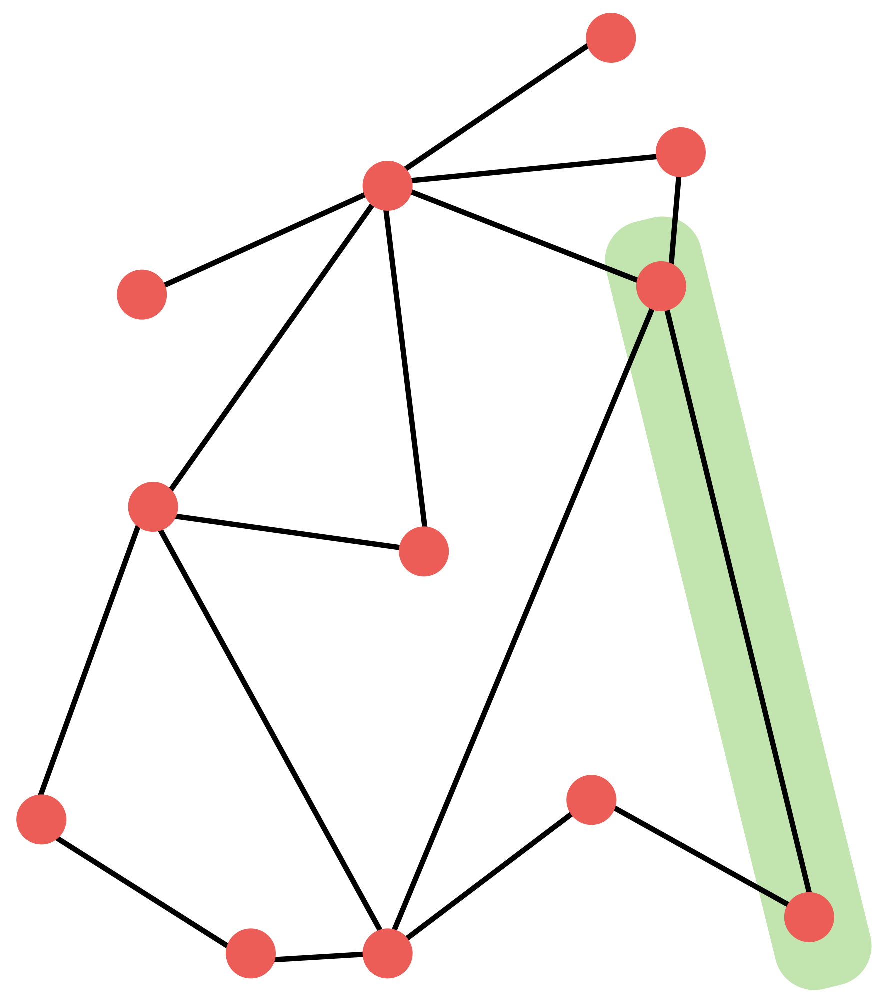
Can G be made edgeless by the removal of at most k vertices?



A simple randomized algorithm with an error probability of 2^{-k}.

Vertex Cover

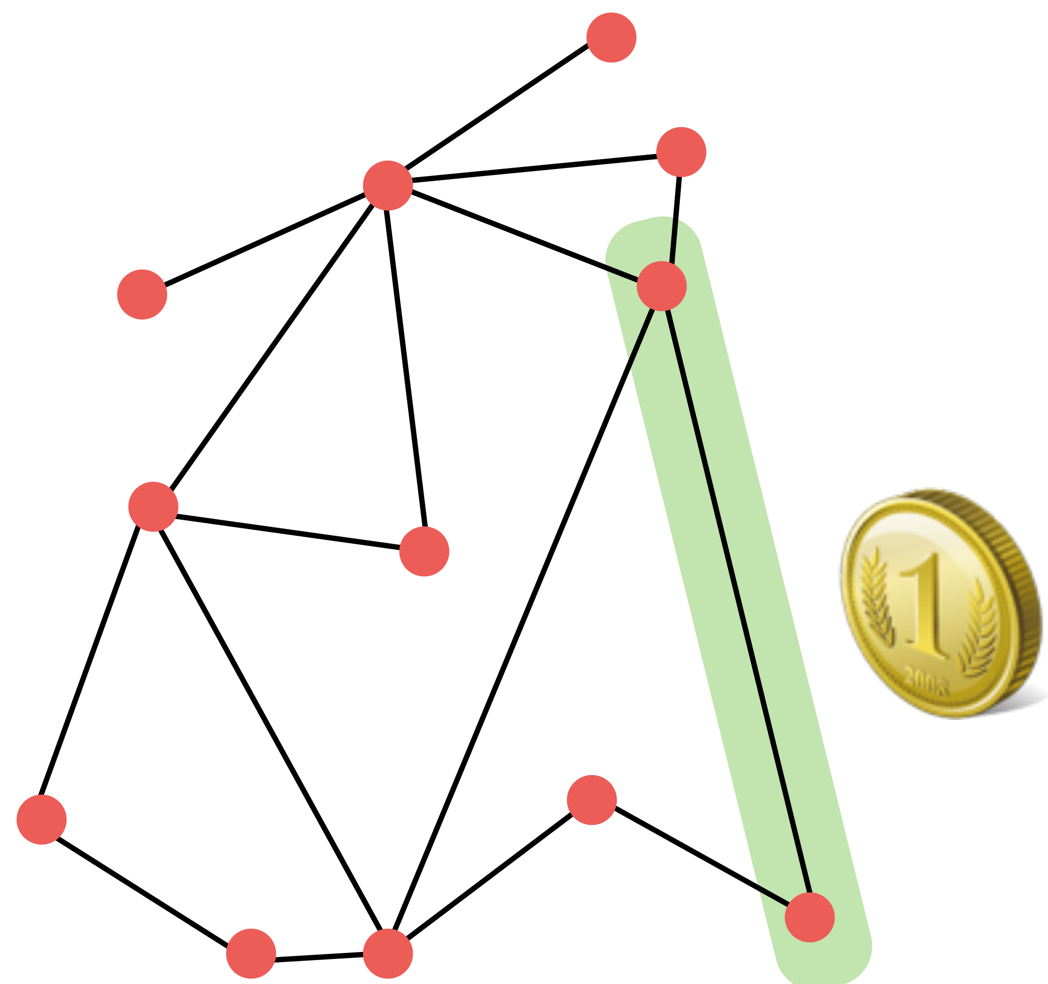
Can G be made edgeless by the removal of at most k vertices?



A simple randomized algorithm with an error probability of 2^{-k}.

Vertex Cover

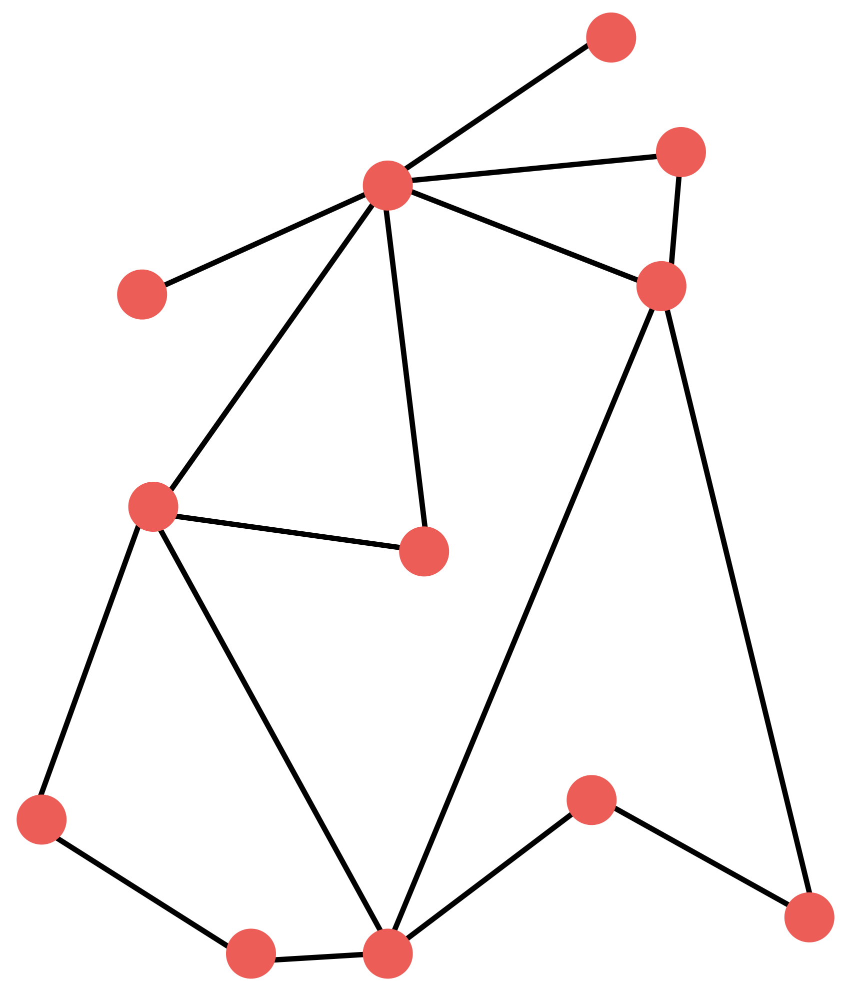
Can G be made edgeless by the removal of at most k vertices?



A simple randomized algorithm with an error probability of 2^{-k}.

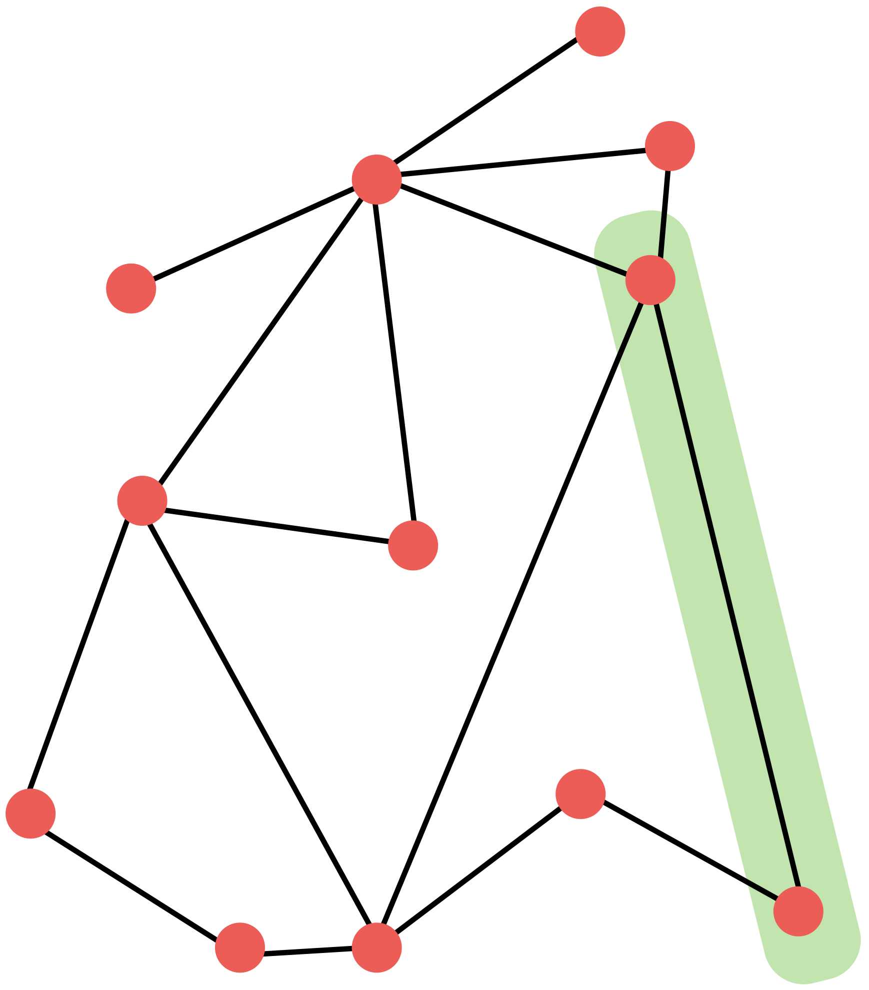
Feedback Vertex Set

Can G be made acyclic by the removal of at most k vertices?



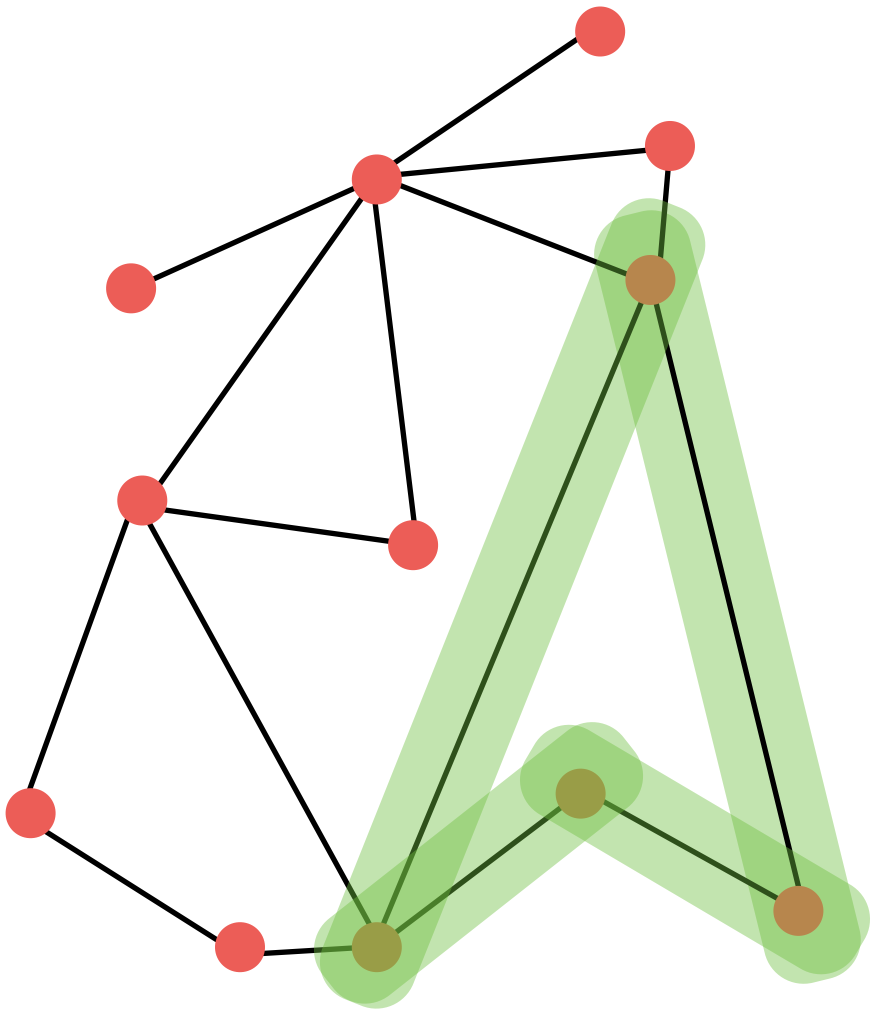
Feedback Vertex Set

Can G be made acyclic by the removal of at most k vertices?



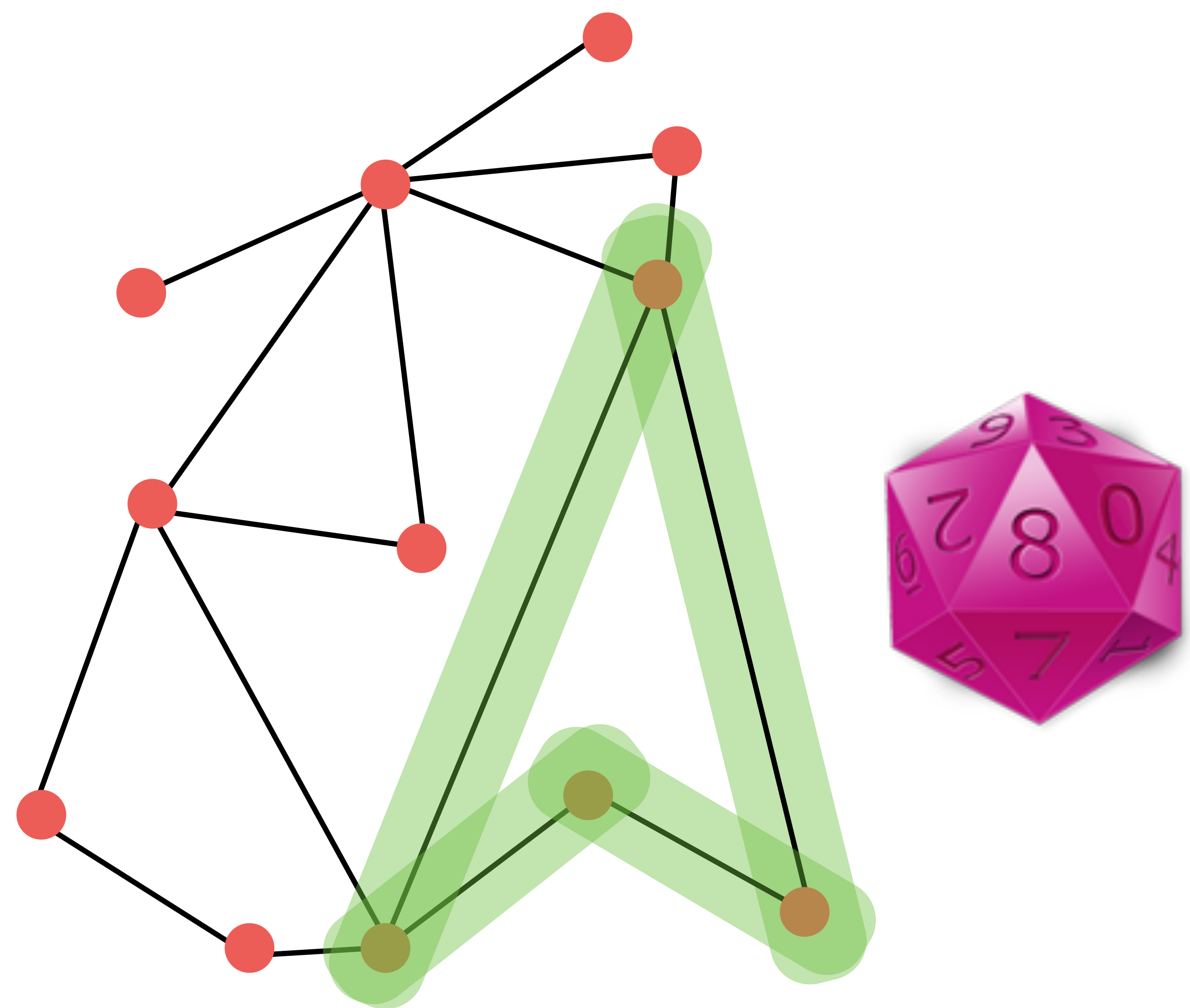
Feedback Vertex Set

Can G be made acyclic by the removal of at most k vertices?



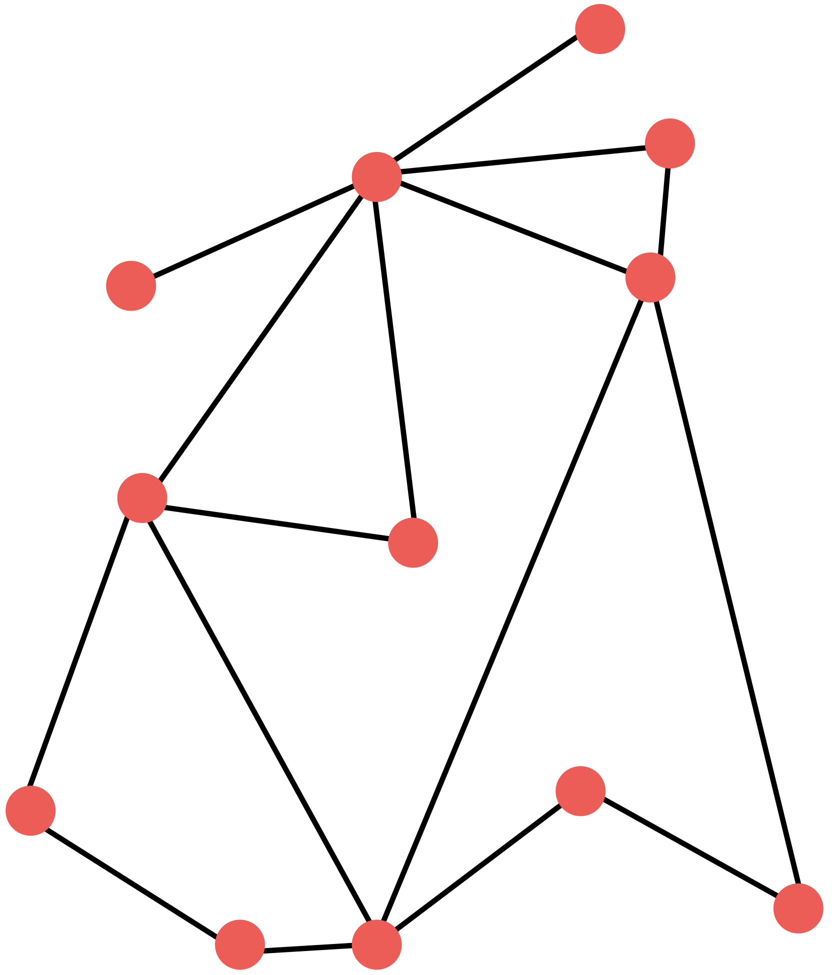
Feedback Vertex Set

Can G be made acyclic by the removal of at most k vertices?



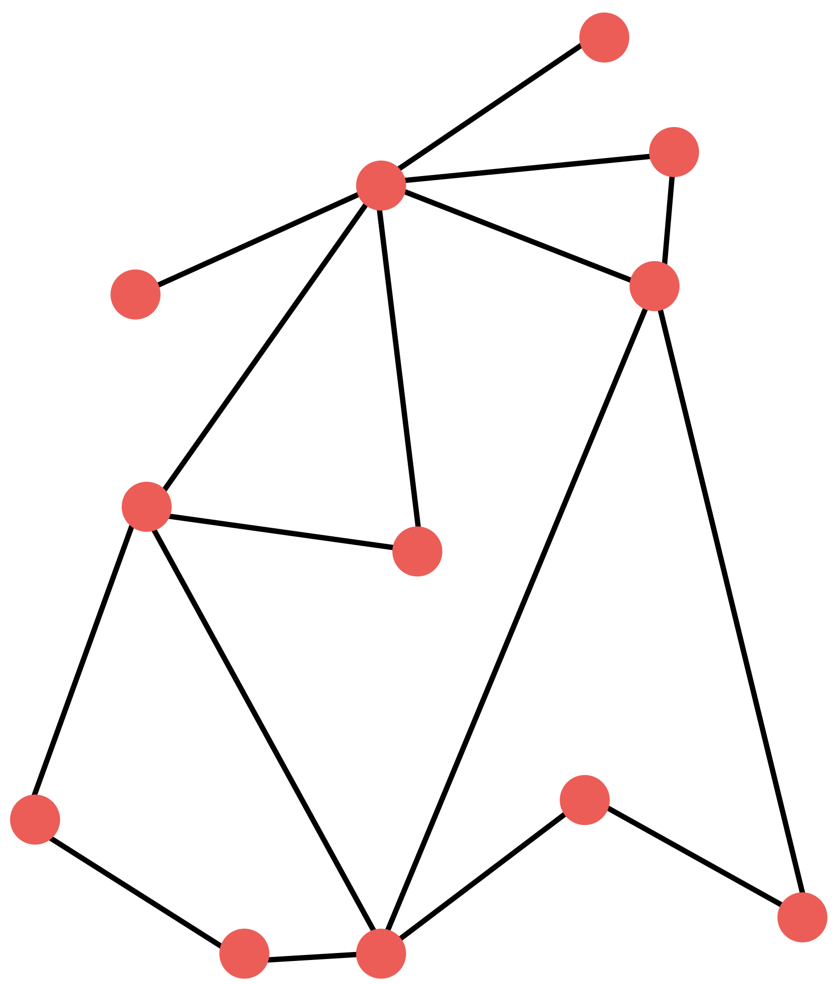
Feedback Vertex Set

Can G be made acyclic by the removal of at most k vertices?



Feedback Vertex Set

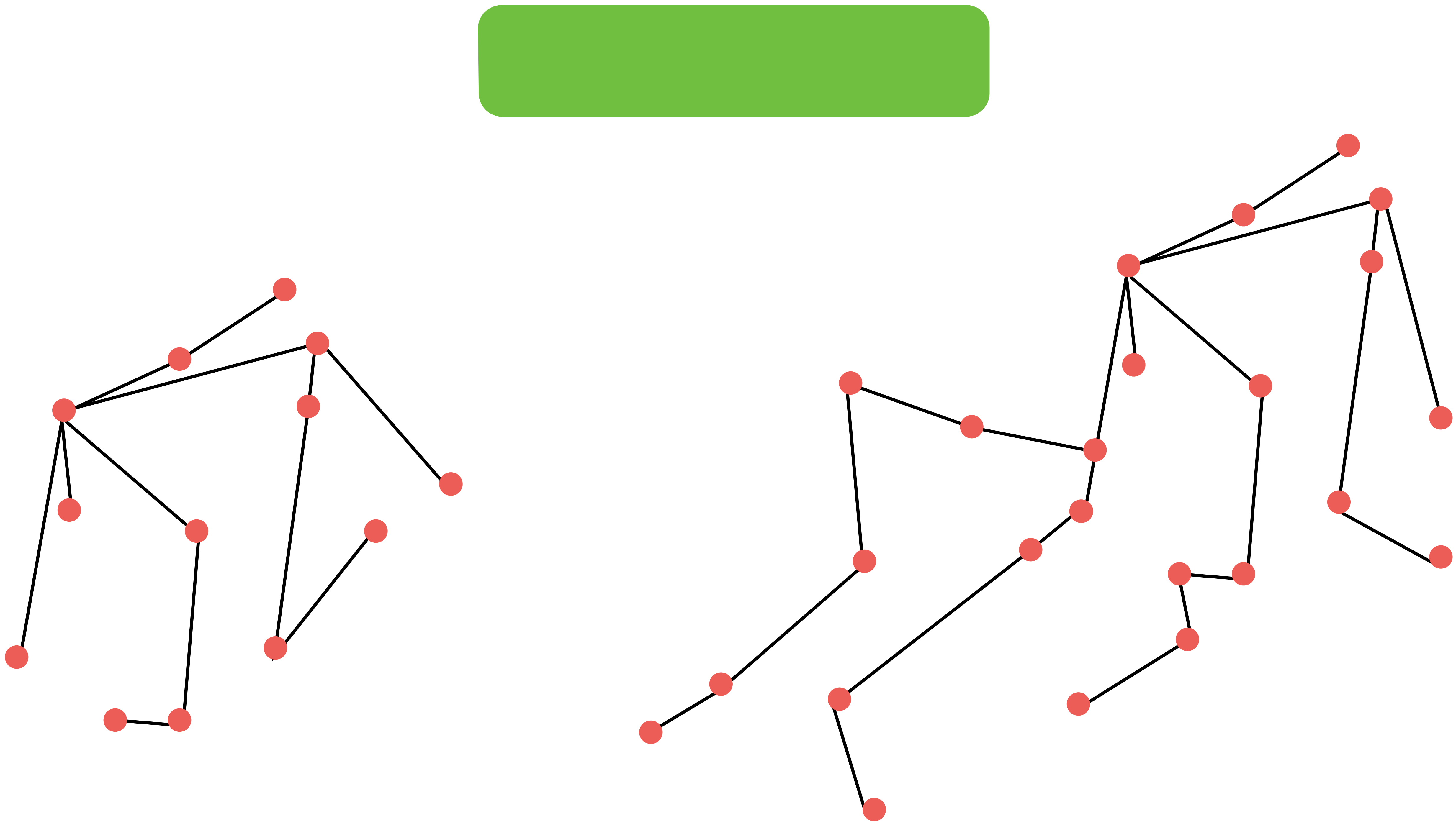
Can G be made acyclic by the removal of at most k vertices?



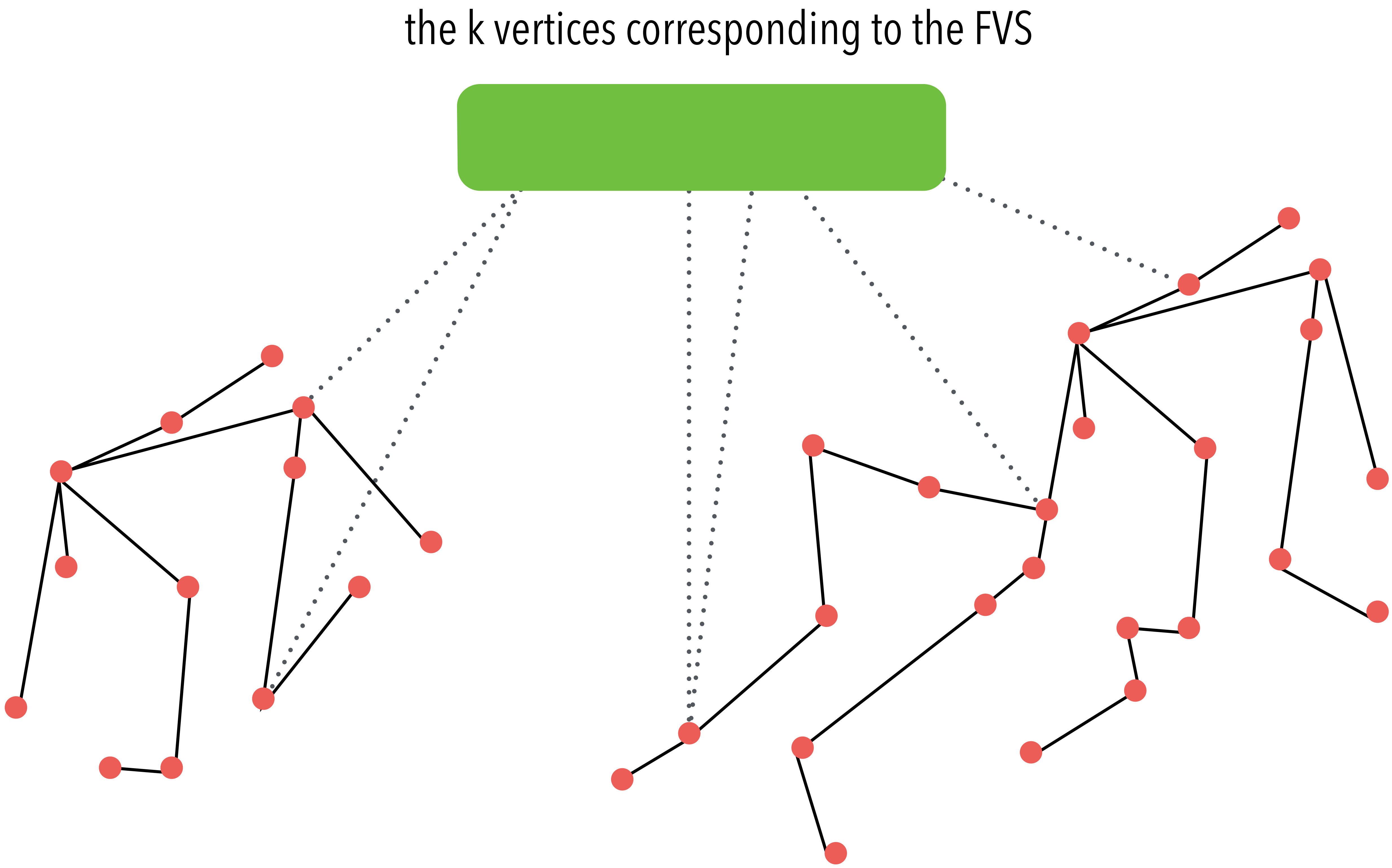
Let's stare at the *structure* of a YES-instance.

#whisper: Let us also restrict ourselves to graphs of constant maximum degree, say five.

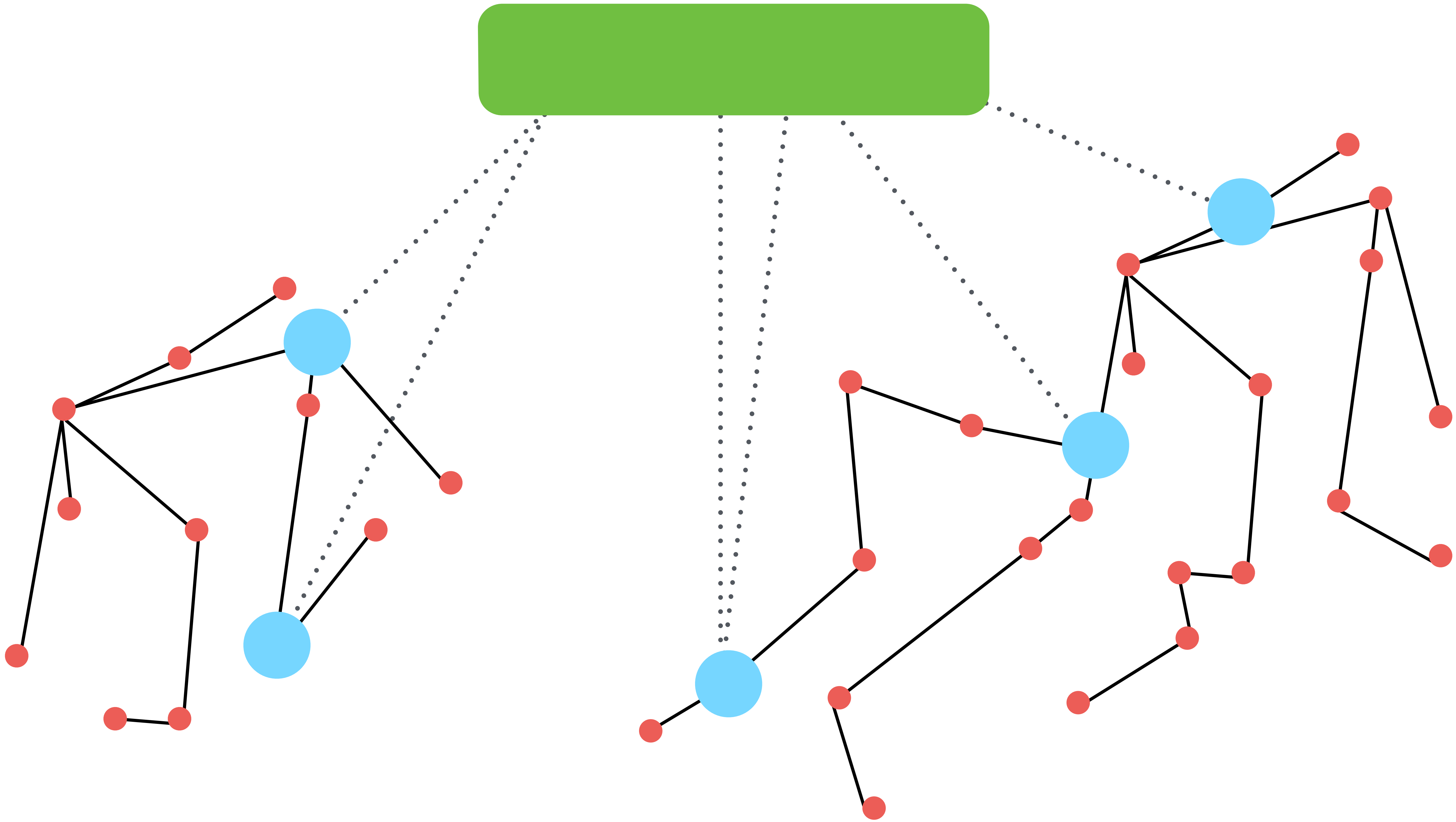
the k vertices corresponding to the FVS

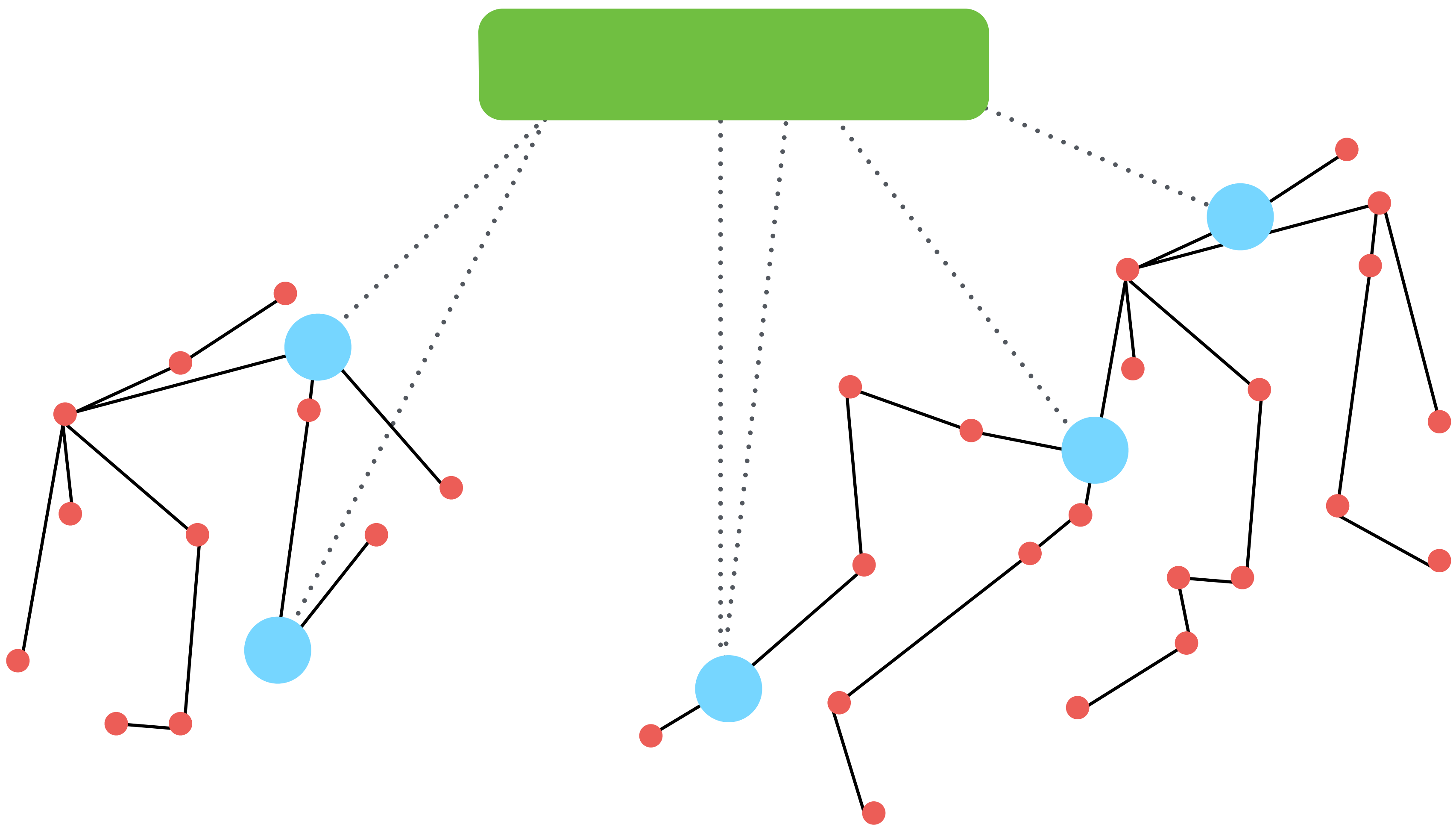


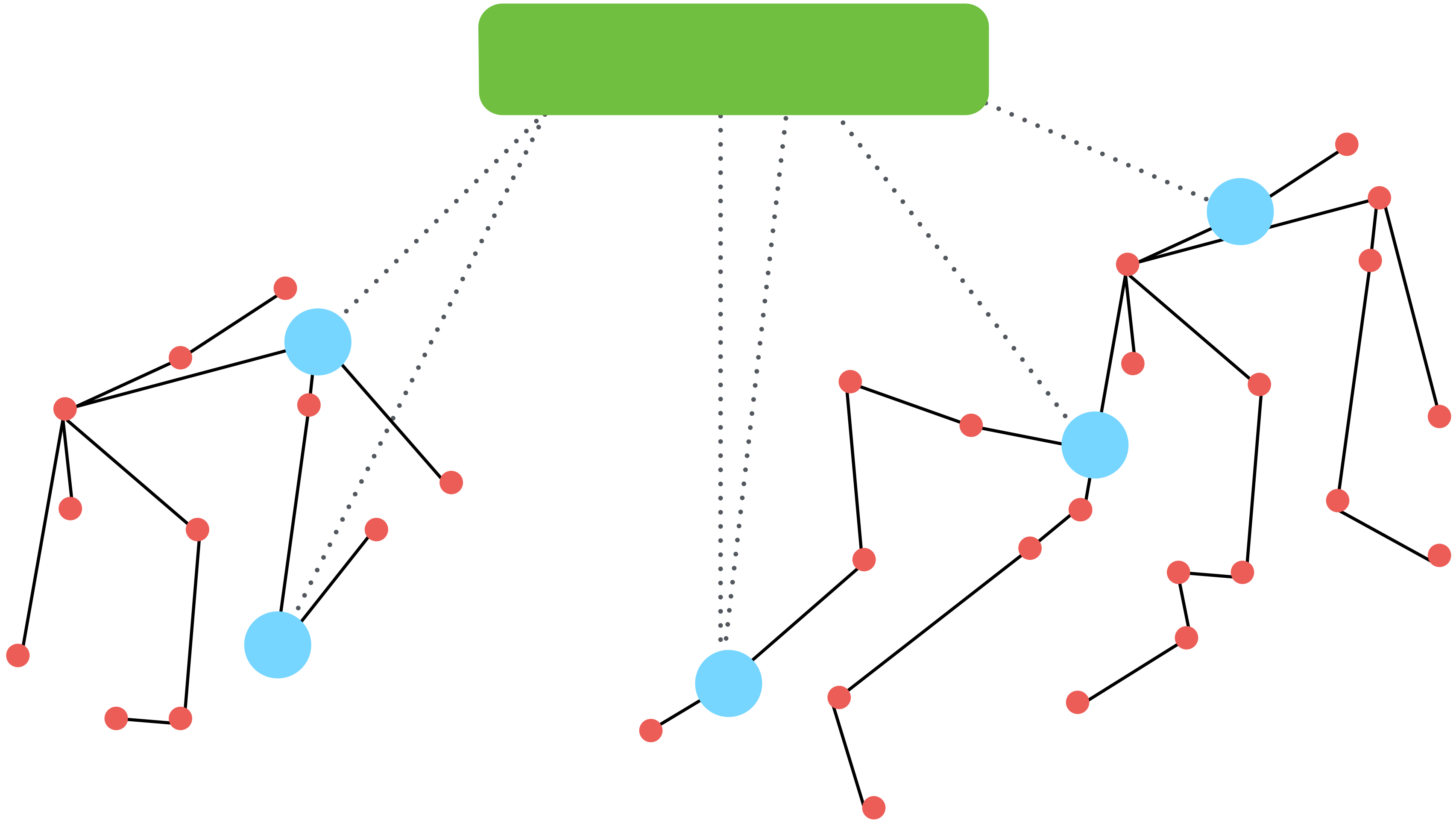
(n-k) vertices and at most (n-k-1) edges "outside".



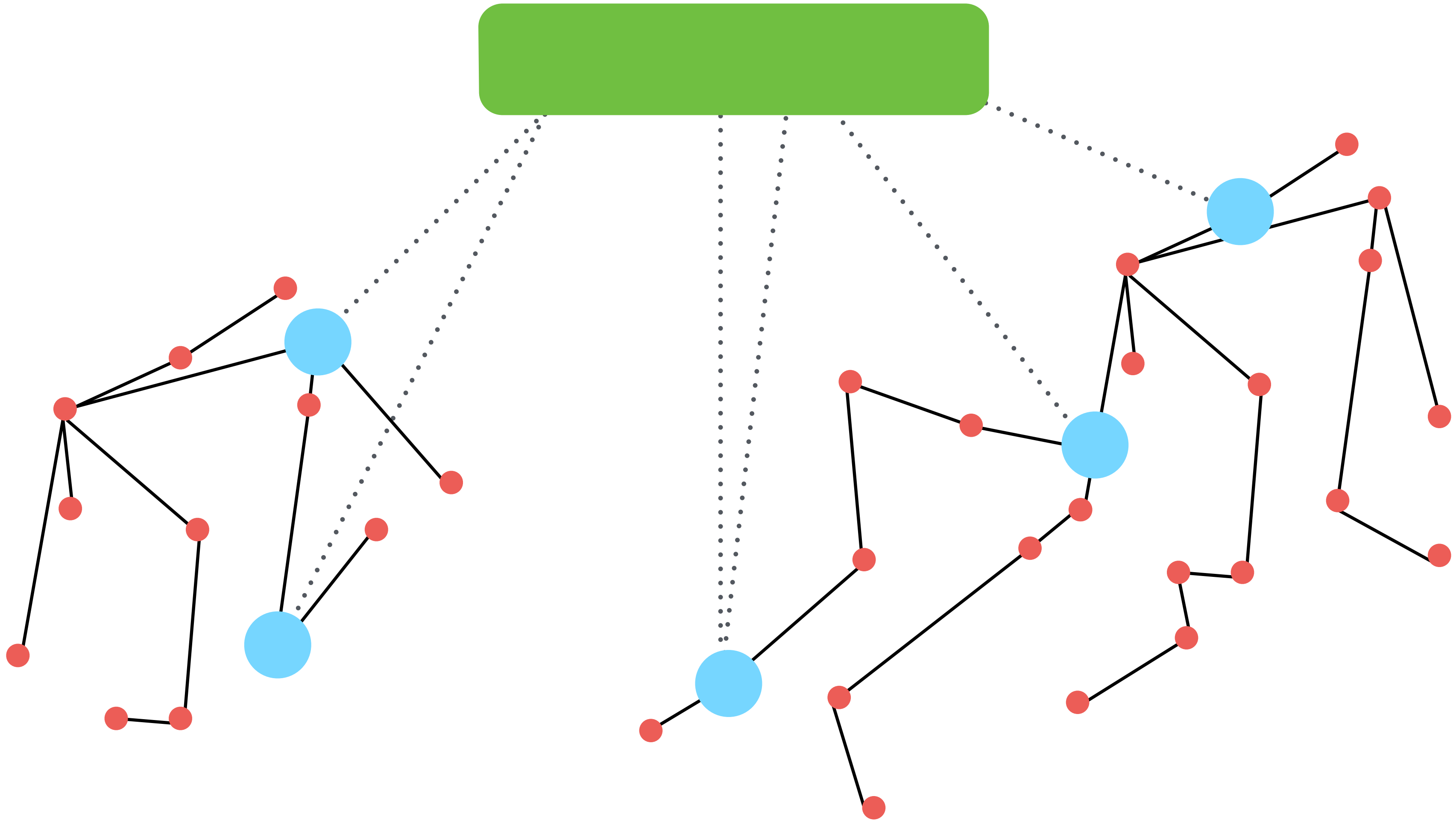
(n-k) vertices and at most (n-k-1) edges "outside".



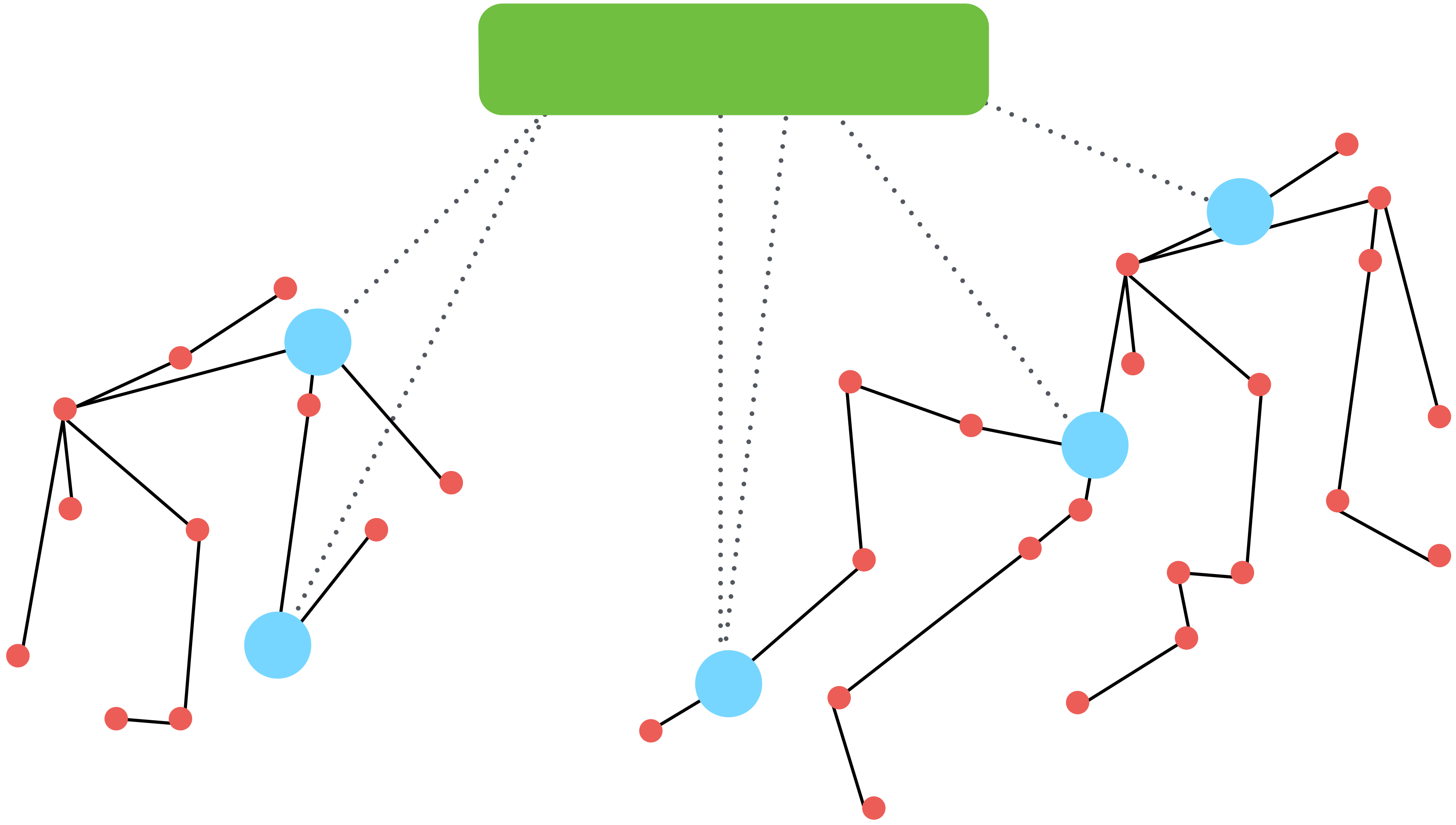




t “points of contact” between S and G\S…

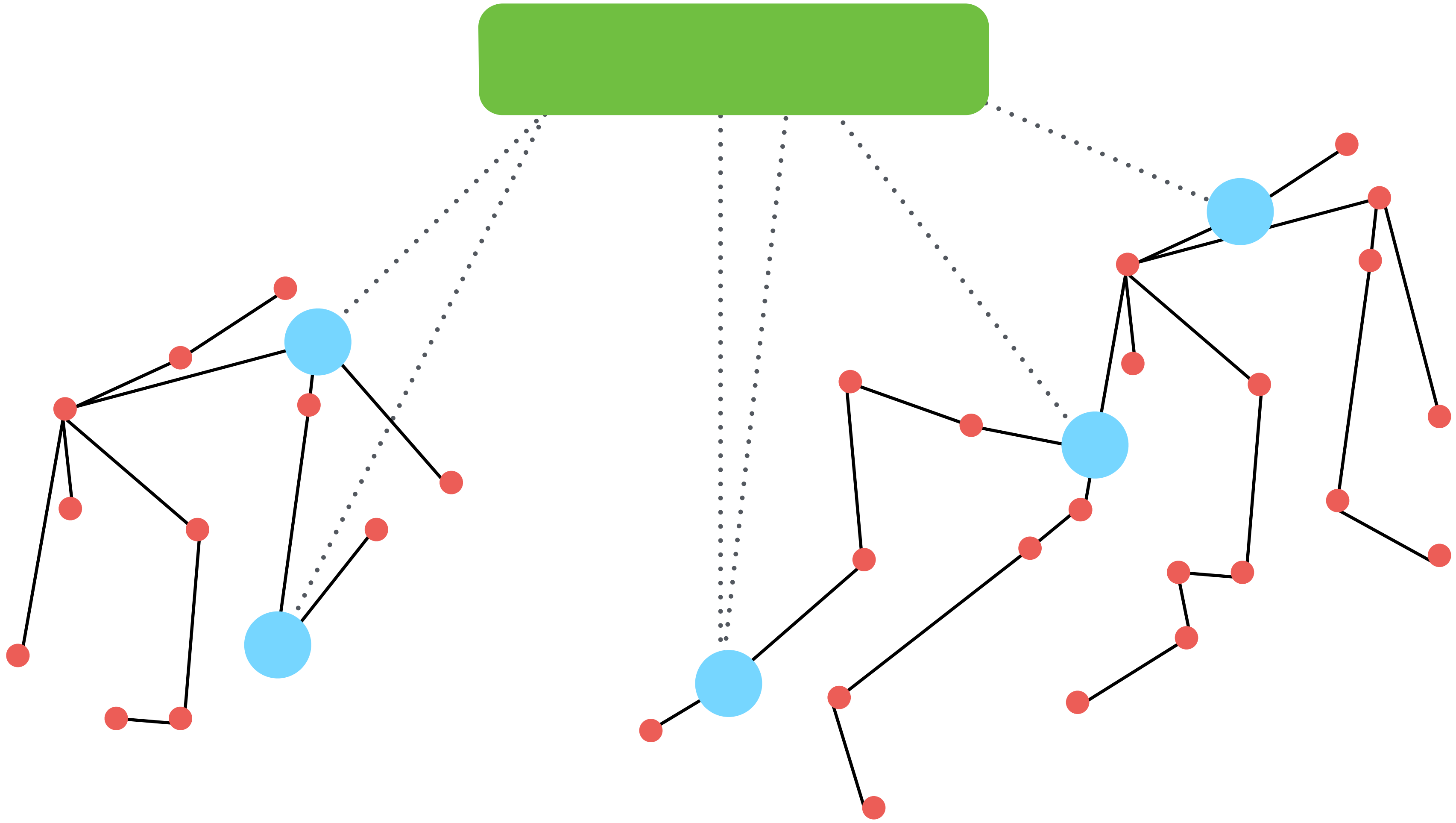


t “points of contact” between S and G\S…   at least t edges going across the two sets.



t “points of contact” between S and G\S…   at least t edges going across the two sets.

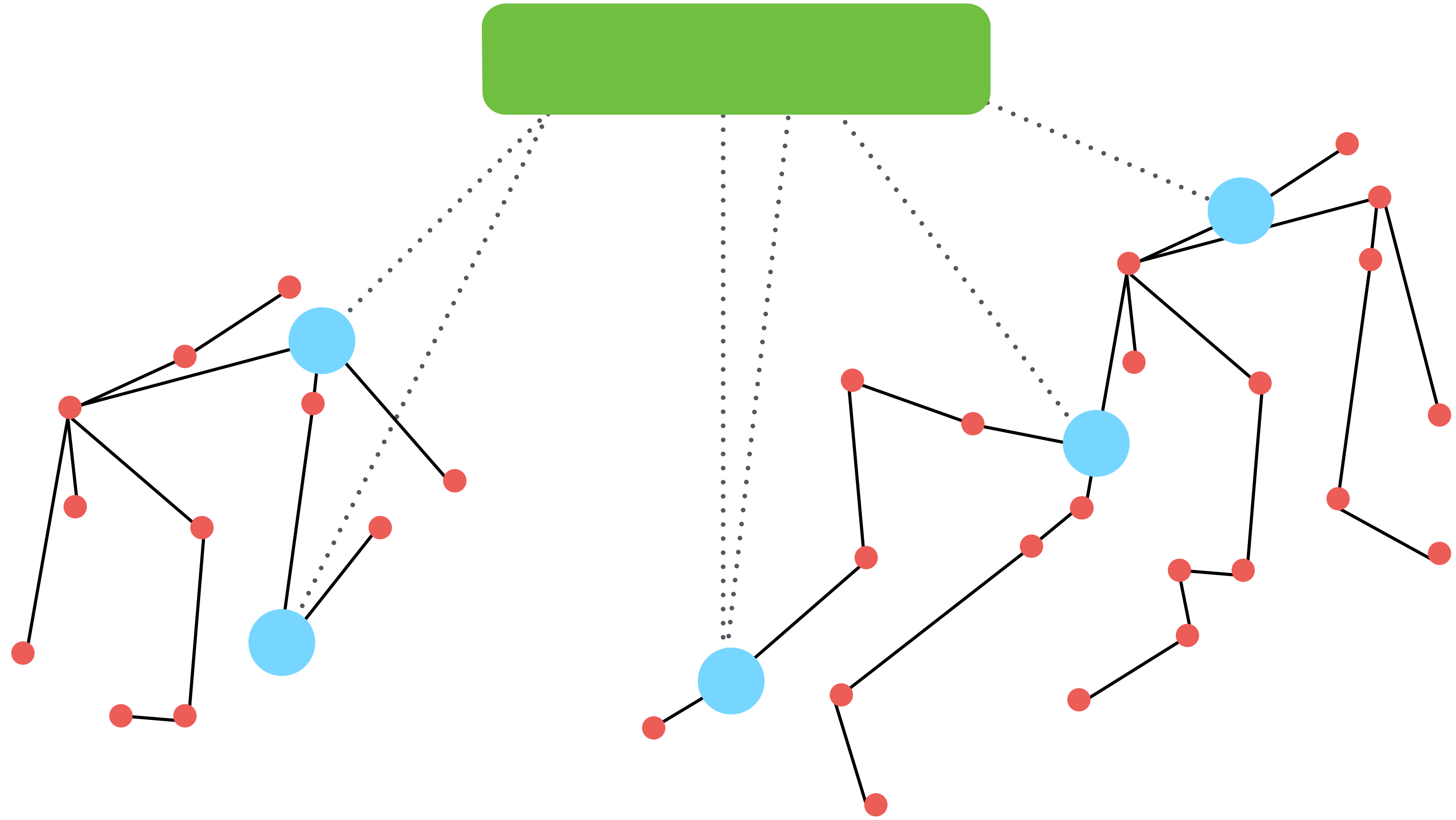
Delete the points of contact in G\S to get at most 5t “pieces”.



t “points of contact” between S and G\S… at least t edges going across the two sets.

Delete the points of contact in G\S to get at most 5t “pieces”.

Suppose each piece has constant size, say c.

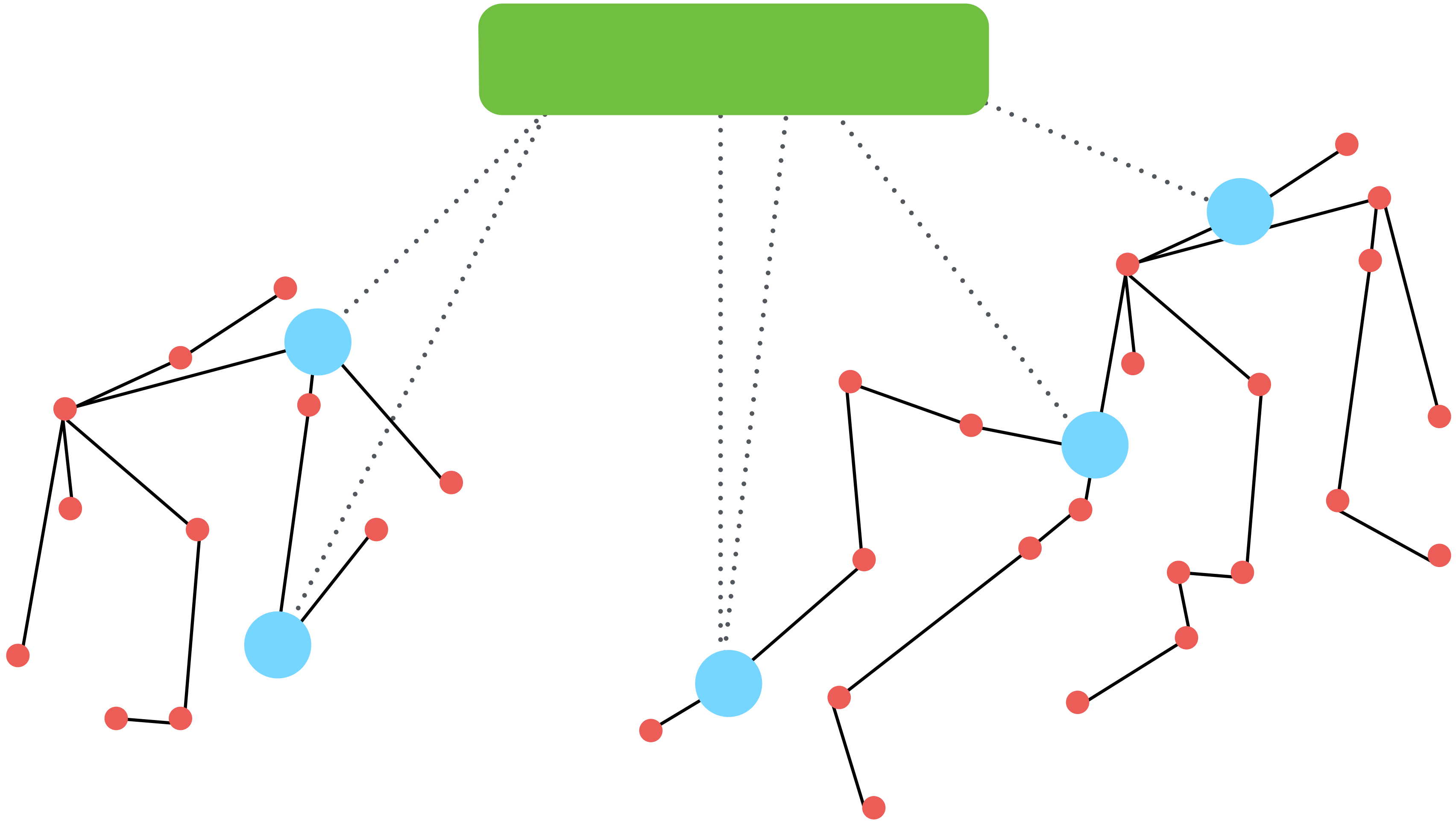


t “points of contact” between S and G\S…   at least t edges going across the two sets.

Delete the points of contact in G\S to get at most 5t “pieces”.

Suppose each piece has constant size, say c.

The total number of edges in G\S is at most 5ct.



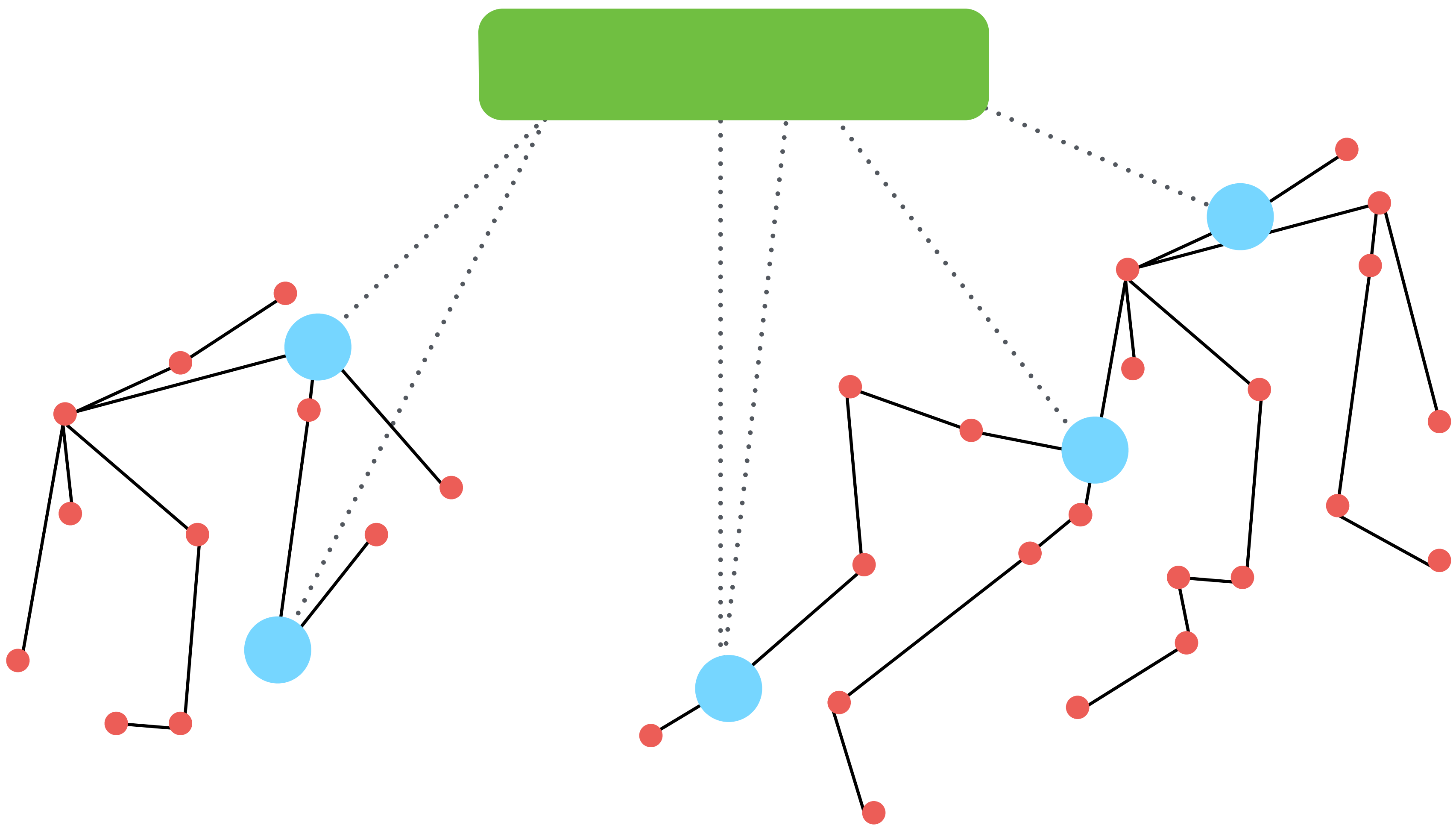
t “points of contact” between S and G\S… at least t edges going across the two sets.

Delete the points of contact in G\S to get at most 5t “pieces”.

Suppose each piece has constant size, say c.

The total number of edges in G\S is at most 5ct.

The number of **good edges is at least t,** and the number of **bad edges is at most 5ct.**



t "points of contact" between S and G\S… at least t edges going across the two sets.

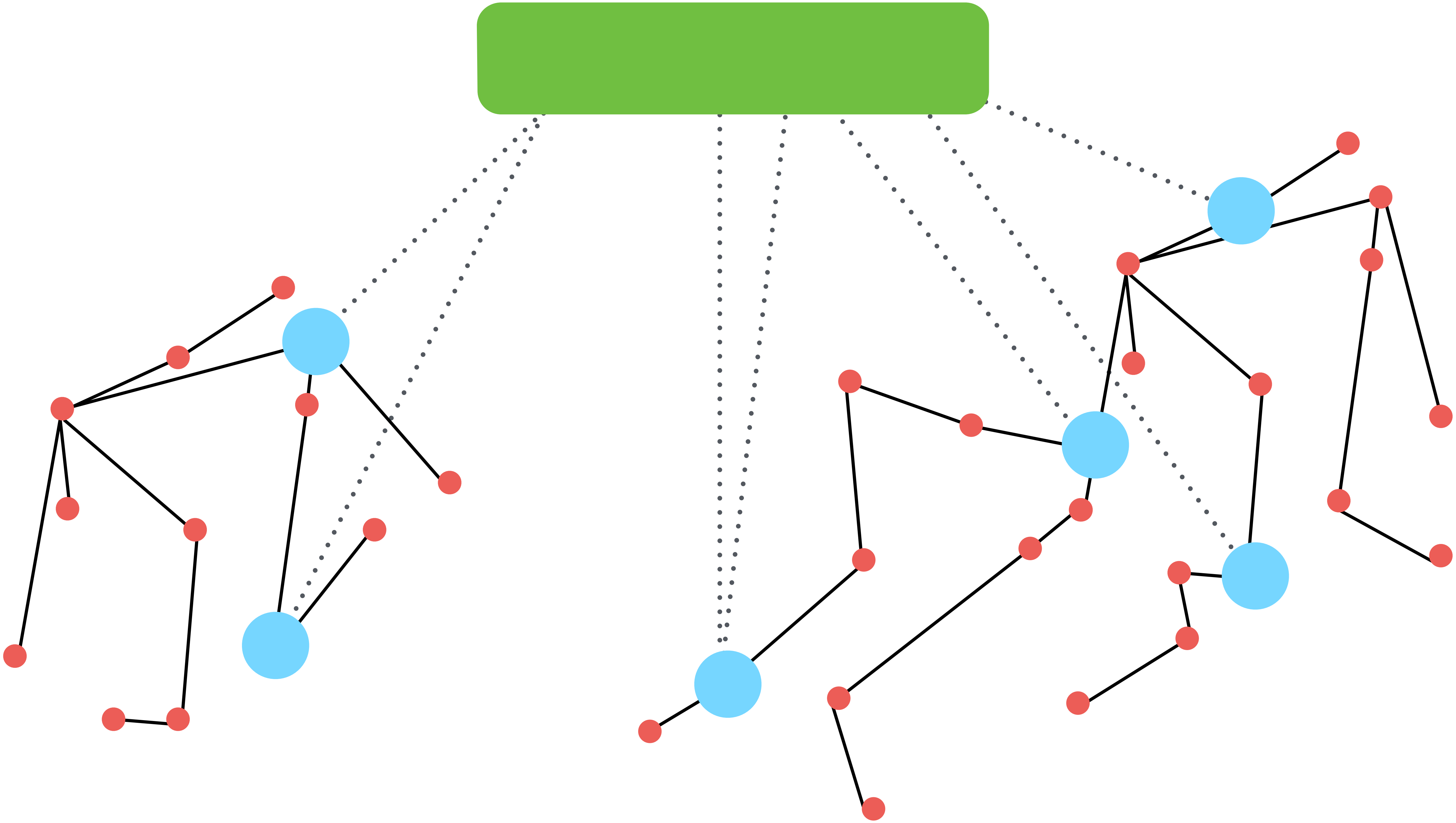
Delete the points of contact in G\S to get at most 5t "pieces".

Suppose each piece has constant size, say c.

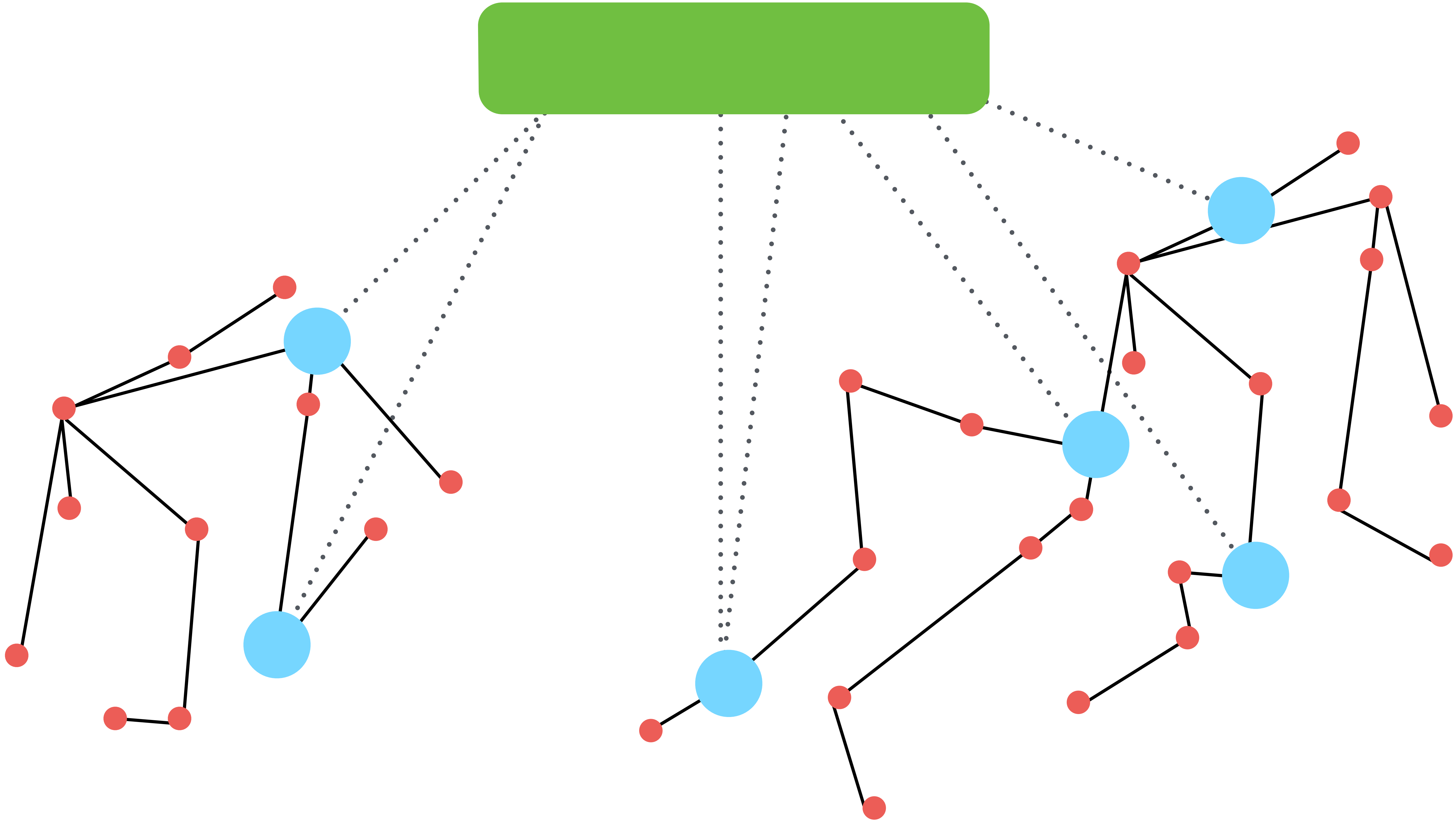
The total number of edges in G\S is at most 5ct.

The number of **good edges is at least t,** and the number of **bad edges is at most 5ct.**

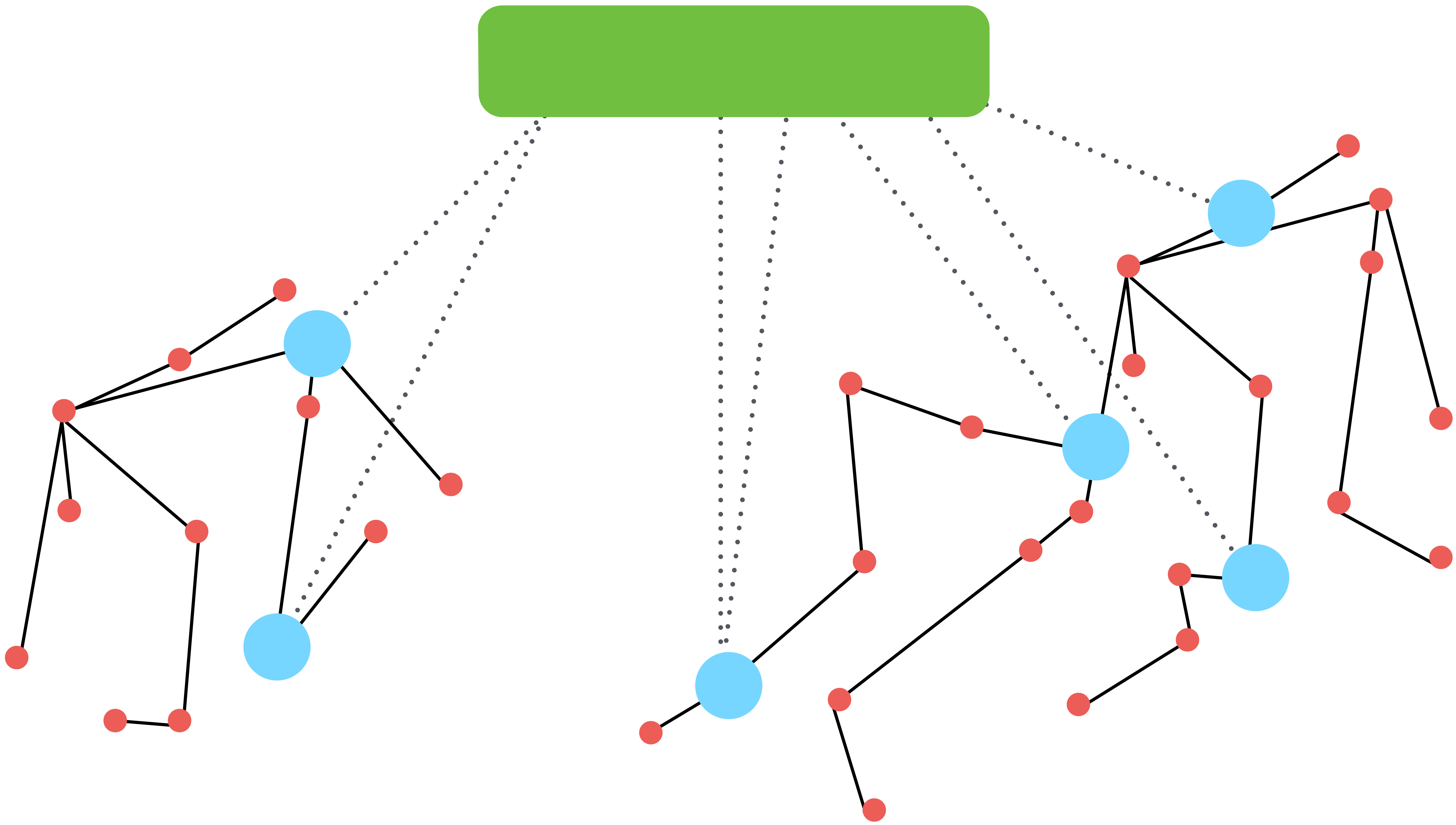
In this scenario, a randomly chosen edge will be **good with constant probability!**



Bad scenario: **one of the pieces is large.**

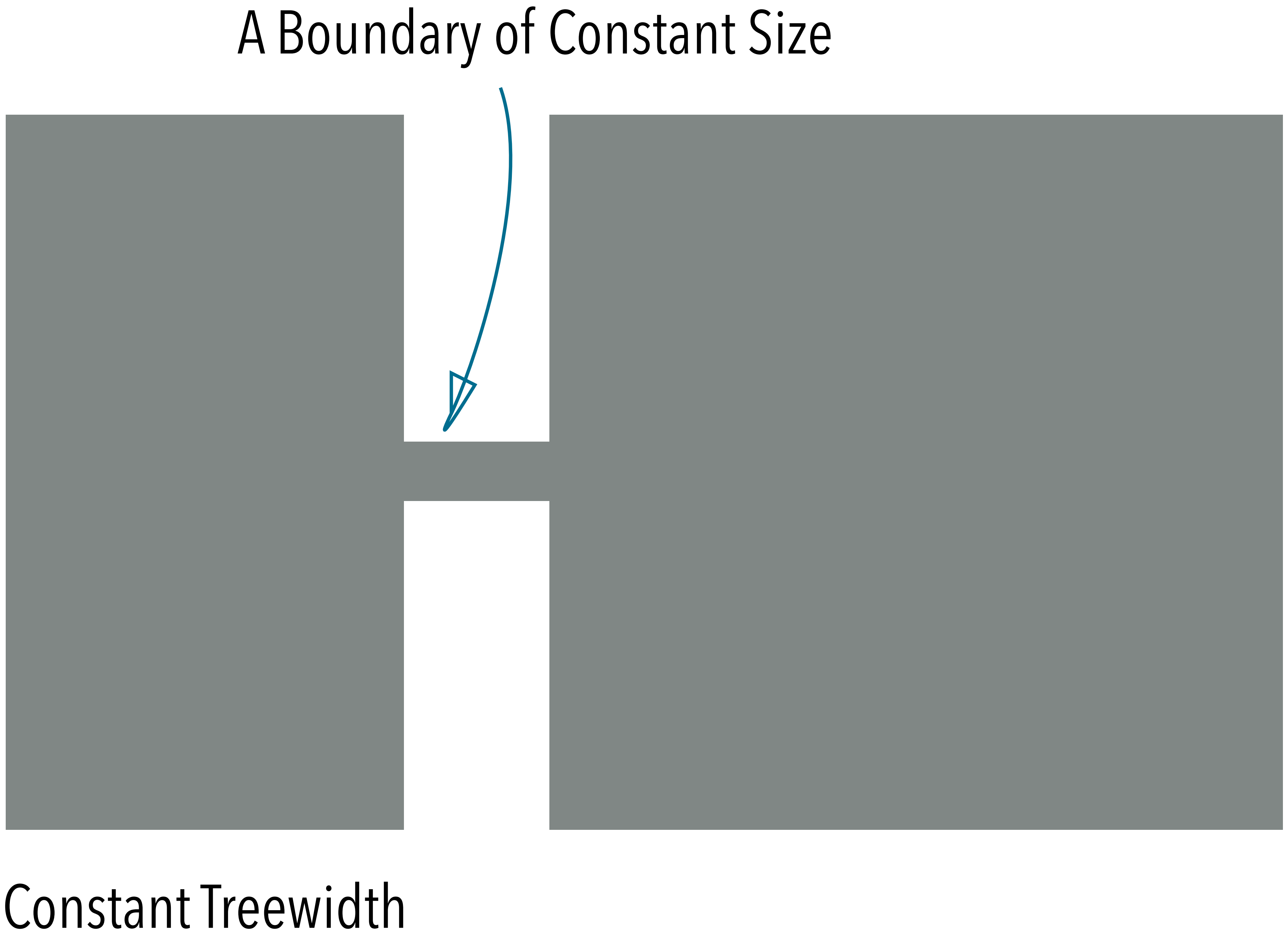


These pieces have **simple structure** and **bounded interaction** with the outer world.



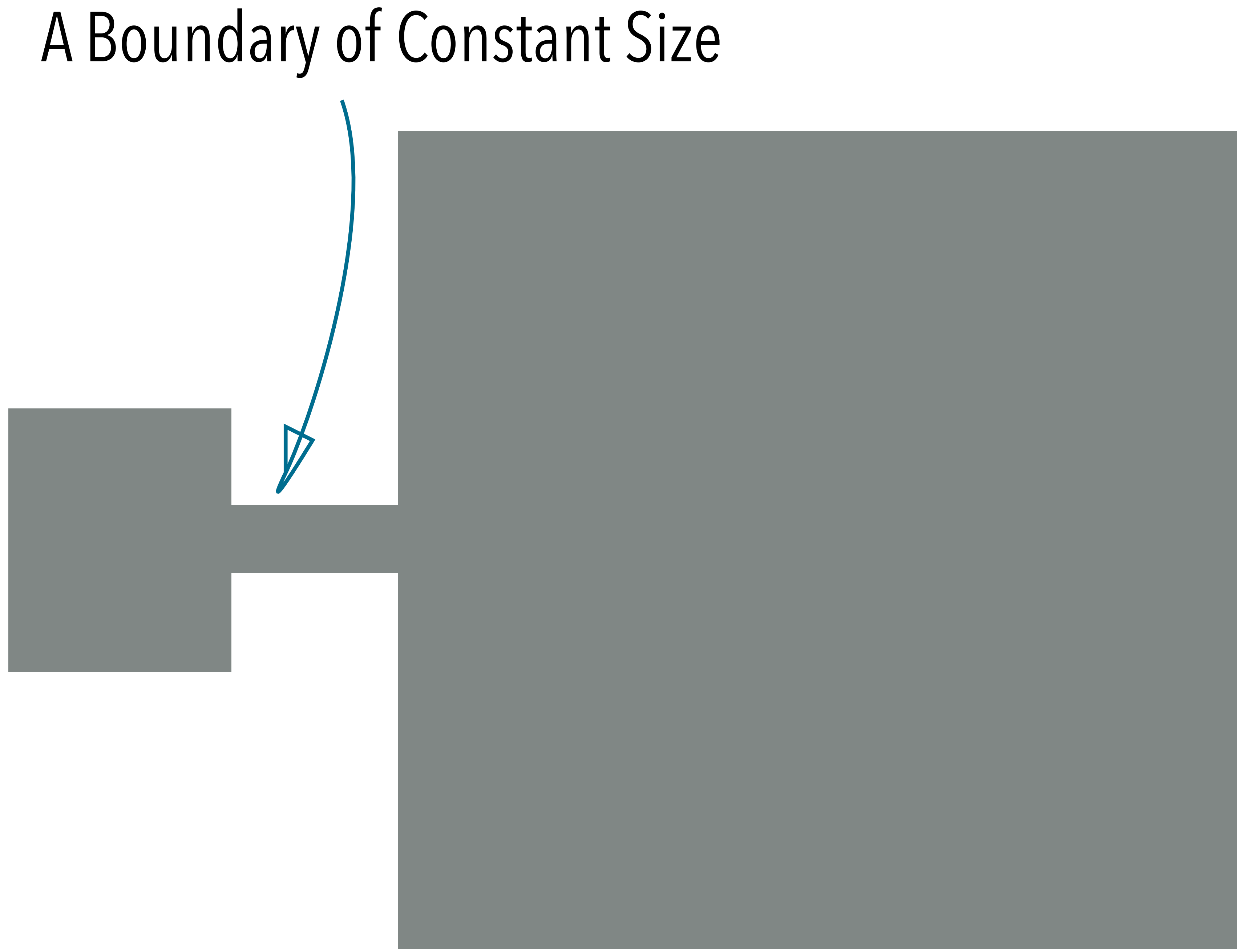
These pieces have **simple structure** and **bounded interaction** with the outer world.

Such pieces are called **“protrusions”.**

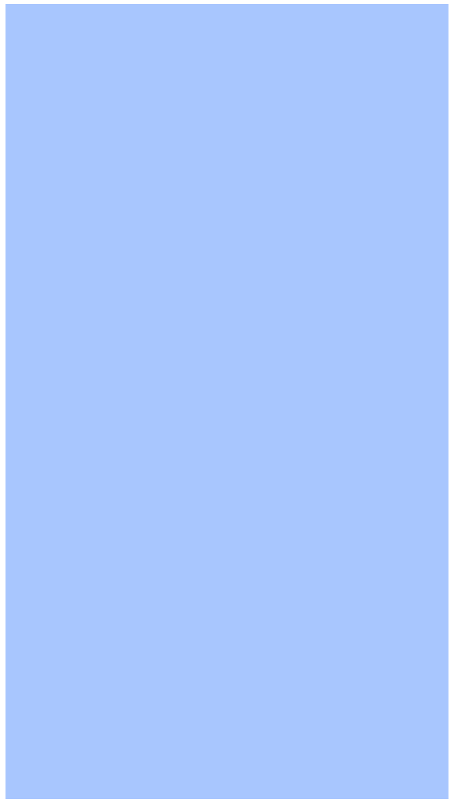

A Boundary of Constant Size
Constant Treewidth

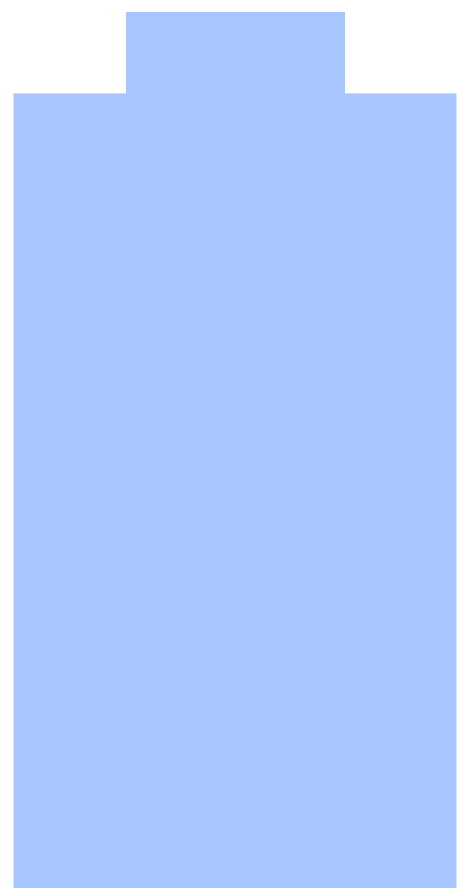
A Boundary of Constant Size

Constant Treewidth

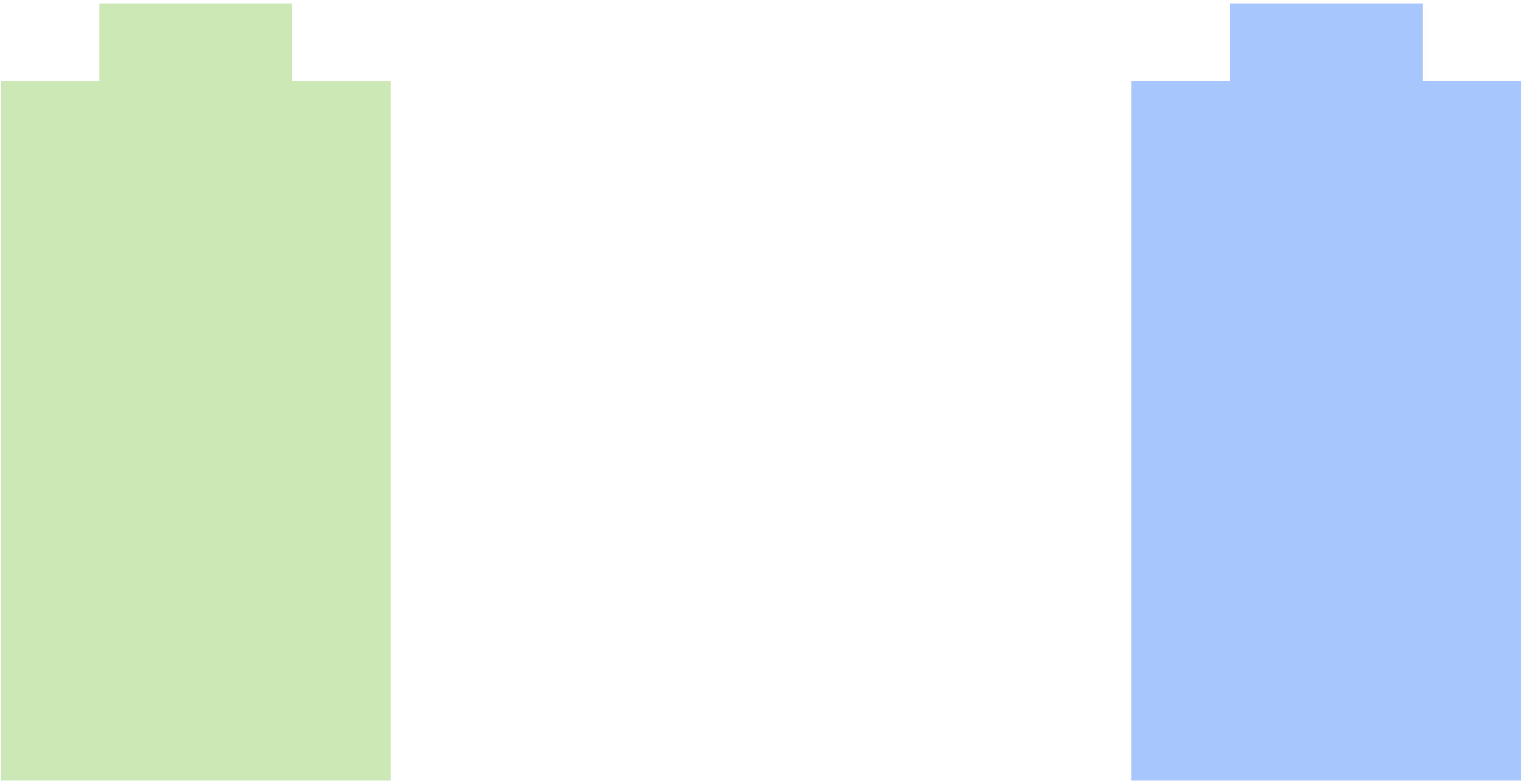


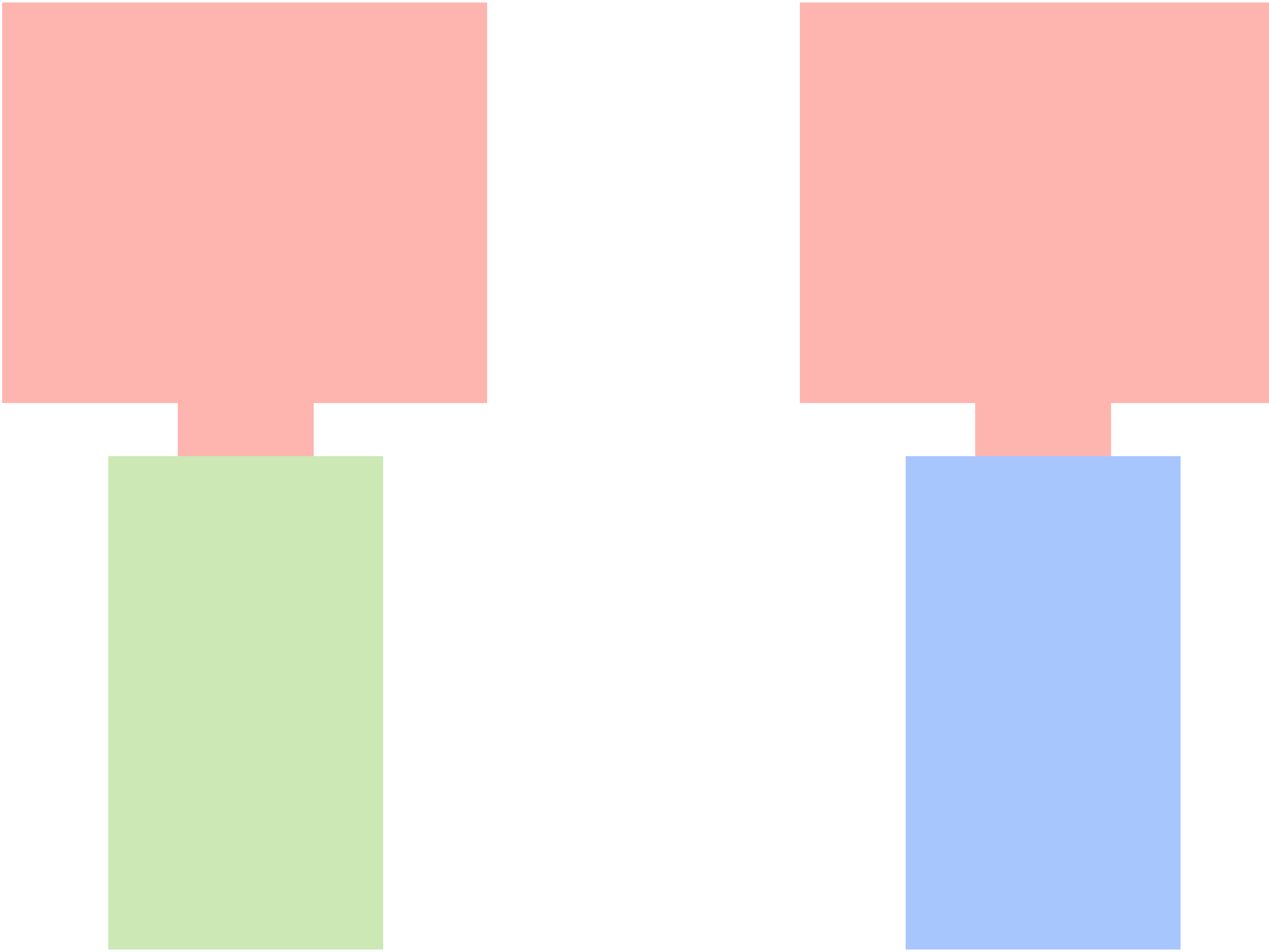
Constant Treewidth



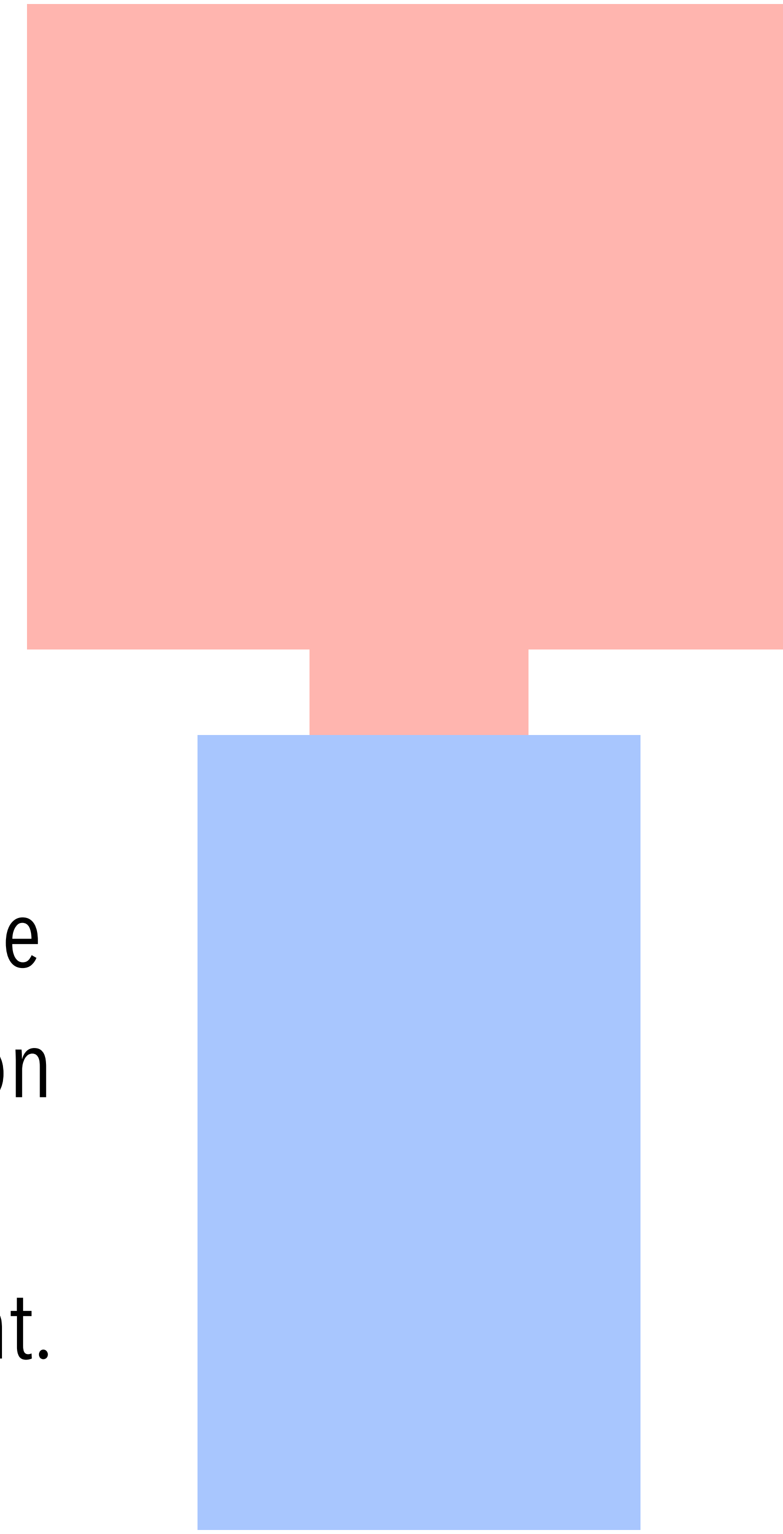


The space of t-boundaried graphs
can be broken up into **equivalence classes**
based on how they "behave" with
the "other side" of the boundary.





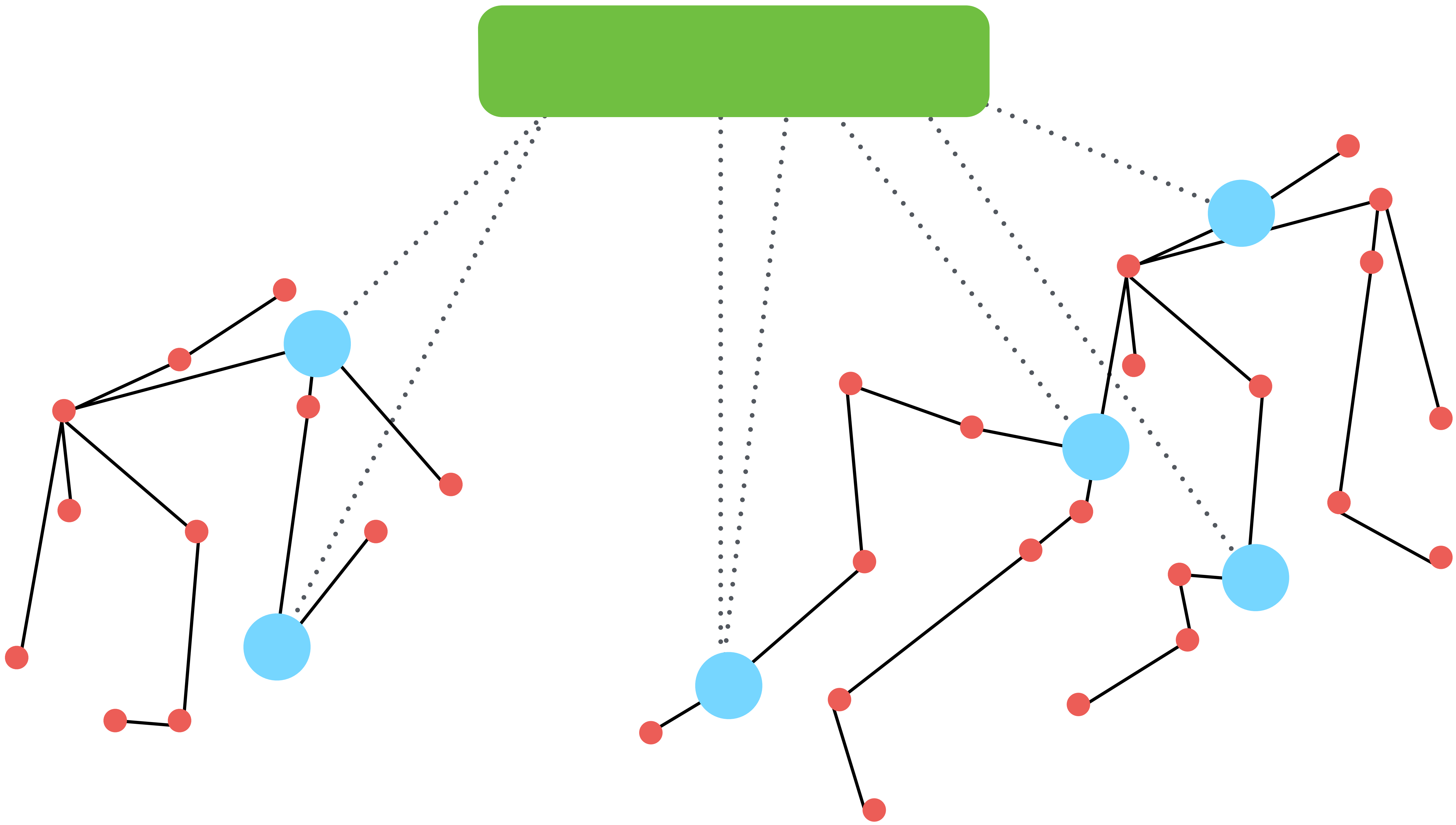
The value of the
optimal solution
is the same
up to a constant.

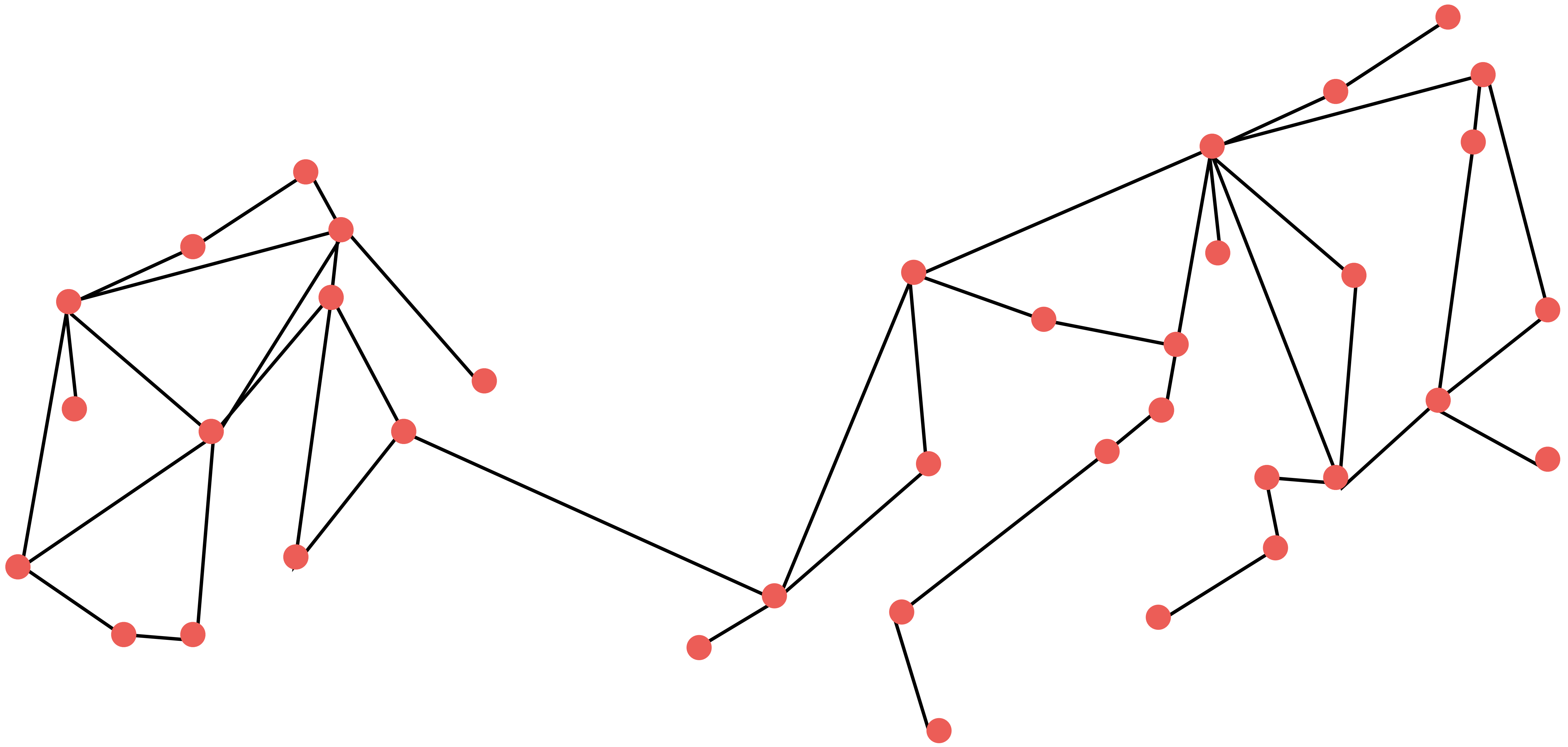


The space of t-boundaried graphs
can be broken up into **equivalence classes**
based on how they "behave" with
the "other side" of the boundary.

The space of t-boundaried graphs
can be broken up into **equivalence classes**
based on how they “behave” with
the “other side” of the boundary.

For some problems,
the number of equivalence classes is **finite**,
allowing us to replace protrusions in graphs.





Finding protrusions? What do we replace them with?

The $\mathcal{F}$-Deletion Problem

The $\mathcal{F}$-Deletion Problem

Can G be made H-minor free, for all H in $\mathcal{F}$,
by the removal of at most k vertices?

The $\mathscr{F}$-Deletion Problem

Can G be made H-minor free, for all H in $\mathscr{F}$,

by the removal of at most k vertices?

Suppose G is a **YES-instance**, and further, suppose $\mathscr{F}$ contains **at least one planar graph**.

The $\mathscr{F}$-Deletion Problem

Can G be made H-minor free, for all H in $\mathscr{F}$,
by the removal of at most k vertices?

Suppose G is a **YES-instance**, and further, suppose $\mathscr{F}$ contains **at least one planar graph**.

Then G\S cannot have large grids.

The $\mathscr{F}$-Deletion Problem

Can G be made H-minor free, for all H in $\mathscr{F}$,

by the removal of at most k vertices?

Suppose G is a **YES-instance**, and further, suppose $\mathscr{F}$ contains **at least one planar graph**.

Then G\S cannot have large grids.

Thus the treewidth of G\S is **bounded by a constant** that depends only on $\mathscr{F}$.

The $\mathcal{F}$-Deletion Problem

Can G be made H-minor free, for all H in $\mathcal{F}$,
by the removal of at most k vertices?

Suppose G is a **YES-instance**, and further, suppose $\mathcal{F}$ contains **at least one planar graph**.

Then G\S cannot have large grids.

Thus the treewidth of G\S is **bounded by a constant** that depends only on $\mathcal{F}$.

The Planar $\mathcal{F}$-Deletion Problem

The Planar $\mathcal{F}$-Deletion Problem

The Planar $\mathcal{F}$-Deletion Problem

Ensure that protrusions are removed.

The Planar $\mathcal{F}$-Deletion Problem

Ensure that protrusions are removed.

Pick an edge uniformly at random.

The Planar $\mathcal{F}$-Deletion Problem

Ensure that protrusions are removed.

Pick an edge uniformly at random.

Include both endpoints.

The Planar $\mathcal{F}$-Deletion Problem

Ensure that protrusions are removed.

Pick an edge uniformly at random.

Include both endpoints.

Pick an endpoint u.a.r.

The Planar $\mathscr{F}$-Deletion Problem

Ensure that protrusions are removed.

Pick an edge uniformly at random.

Include both endpoints.

Pick an endpoint u.a.r.

Repeat till G is $\mathscr{F}$-free.

The Planar $\mathcal{F}$-Deletion Problem

Ensure that protrusions are removed.

Pick an edge uniformly at random.

Include both endpoints.

(Approximation algorithm.)

Pick an endpoint u.a.r.

Repeat till G is $\mathcal{F}$-free.

The Planar $\mathcal{F}$-Deletion Problem

Ensure that protrusions are removed.

Pick an edge uniformly at random.

Include both endpoints.

(Approximation algorithm.)

Pick an endpoint u.a.r.

(Randomized algorithm.)

Repeat till G is $\mathcal{F}$-free.

The Planar $\mathscr{F}$-Deletion Problem accounts for several specific problems…

The Planar $\mathcal{F}$-Deletion Problem accounts for several specific problems…

$\mathcal{F}$	
K_2	Vertex Cover
K_3	Feedback Vertex Set
((k+1)-by-(k+1)) Grids	Treewidth k
$K_{2,3}$, K_4	Outerplanar
K_4	Series-Parallel
K_3,T_2	Pathwidth-One

Thanks to **Graph Minors**, we have a $f(k)n^2$ algorithm for the $\mathscr{F}$-Deletion problem.

Thanks to **Graph Minors**, we have a $f(k)n^2$ algorithm for the $\mathscr{F}$-Deletion problem.

For **Planar** $\mathscr{F}$-Deletion, the starting point was a double-exponential algorithm.

[Bodlaender, 1997]

Thanks to **Graph Minors**, we have a $f(k)n^2$ algorithm for the $\mathscr{F}$-Deletion problem.

For **Planar** $\mathscr{F}$-Deletion, the starting point was a double-exponential algorithm.
[Bodlaender, 1997]

Many single-exponential algorithms are known for special cases of $\mathscr{F}$.

Thanks to **Graph Minors**, we have a $f(k)n^2$ algorithm for the $\mathscr{F}$-Deletion problem.

For **Planar** $\mathscr{F}$-Deletion, the starting point was a double-exponential algorithm.
[Bodlaender, 1997]

Many single-exponential algorithms are known for special cases of $\mathscr{F}$.

Cao, Chen & Liu
[2010]

Feedback Vertex Set has a $O^*(3.83^k)$ algorithm.

Thanks to **Graph Minors**, we have a $f(k)n^2$ algorithm for the $\mathscr{F}$-Deletion problem.

For **Planar** $\mathscr{F}$-Deletion, the starting point was a double-exponential algorithm.
[Bodlaender, 1997]

Many single-exponential algorithms are known for special cases of $\mathscr{F}$.

<u>Cao, Chen & Liu</u> [2010] — **Feedback Vertex Set** has a $O^*(3.83^k)$ algorithm.

<u>Kim, Paul & Philip</u> [2012] — **$\{K_4\}$-Deletion** has a $O^*(2^{O(k)})$ algorithm.

Thanks to **Graph Minors**, we have a $f(k)n^2$ algorithm for the $\mathscr{F}$-Deletion problem.

For **Planar** $\mathscr{F}$-Deletion, the starting point was a double-exponential algorithm. [Bodlaender, 1997]

Many single-exponential algorithms are known for special cases of $\mathscr{F}$.

Cao, Chen & Liu [2010]	**Feedback Vertex Set** has a $O^*(3.83^k)$ algorithm.
Kim, Paul & Philip [2012]	**$\{K_4\}$-Deletion** has a $O^*(2^{O(k)})$ algorithm.
Cygan, Pilipczuk, Pilipczuk & Wojtaszczyk [2010]	**Pathwidth-1-Deletion** has a $O^*(4.65^k)$ algorithm.

Fomin, Lokshtanov, M. & Saurabh [2013]

Planar $\mathscr{F}$-Deletion admits a randomized **$(2^{O(k)}n)$** algorithm when every graph in $\mathscr{F}$ is connected.

Fomin, Lokshtanov, M. & Saurabh [2013]

Planar $\mathscr{F}$-Deletion admits a randomized **($2^{O(k)}n$)** algorithm when every graph in $\mathscr{F}$ is connected.

Best possible under ETH.

Fomin, Lokshtanov, M. & Saurabh [2013]

Planar $\mathscr{F}$-Deletion admits a randomized **($2^{O(k)}n$)** algorithm when every graph in $\mathscr{F}$ is connected.

Best possible under ETH.

Planar $\mathscr{F}$-Deletion admits a deterministic **($2^{O(k)}n \log^2 n$)** algorithm when every graph in $\mathscr{F}$ is connected.

Fomin, Lokshtanov, M. & Saurabh [2013]

Planar $\mathscr{F}$-Deletion admits a randomized **($2^{O(k)}n$)** algorithm when every graph in $\mathscr{F}$ is connected.

Best possible under ETH.

Planar $\mathscr{F}$-Deletion admits a deterministic **($2^{O(k)}n \log^2 n$)** algorithm when every graph in $\mathscr{F}$ is connected.

Planar $\mathscr{F}$-Deletion admits an O(nm) randomized algorithm that leads us to a **constant-factor approximation**.

Can we get a deterministic constant-factor approximation algorithm?

Fomin, Lokshtanov, M. & Saurabh [2013]

Planar $\mathscr{F}$-Deletion admits a randomized **$(2^{O(k)}n)$** algorithm when every graph in $\mathscr{F}$ is connected.

Best possible under ETH.

Planar $\mathscr{F}$-Deletion admits a deterministic **$(2^{O(k)}n \log^2 n)$** algorithm when every graph in $\mathscr{F}$ is connected.

Planar $\mathscr{F}$-Deletion admits an O(nm) randomized algorithm that leads us to a **constant-factor approximation**.

Can we get a deterministic constant-factor approximation algorithm?

Fomin, Lokshtanov, M., Philip & Saurabh [2013]

Planar $\mathscr{F}$-Deletion admits an deterministic algorithm that leads us to a **$O(\log^{3/2}(OPT))$** approximation.

Fomin, Lokshtanov, M. & Saurabh [2013]

Planar $\mathscr{F}$-Deletion admits a randomized $\mathbf{(2^{O(k)}n)}$ algorithm when every graph in $\mathscr{F}$ is connected.

Best possible under ETH.

Planar $\mathscr{F}$-Deletion admits a deterministic $\mathbf{(2^{O(k)}n \log^2 n)}$ algorithm when every graph in $\mathscr{F}$ is connected.

Planar $\mathscr{F}$-Deletion admits an O(nm) randomized algorithm that leads us to a **constant-factor approximation**.

Can we get a deterministic constant-factor approximation algorithm?

Fomin, Lokshtanov, M., Philip & Saurabh [2013]

Planar $\mathscr{F}$-Deletion admits an deterministic algorithm that leads us to a $\mathbf{O(\log^{3/2}(OPT))}$ approximation.

Kim, Langer, Paul, Reidl, Rossmanith, Sau, & Sikdar [2013]

Planar $\mathscr{F}$-Deletion admits a deterministic $\mathbf{(2^{O(k)}n^2)}$ algorithm.

Fomin, Lokshtanov, M. & Saurabh [2013]

Planar $\mathscr{F}$-Deletion admits a randomized $(2^{O(k)}n)$ algorithm when every graph in $\mathscr{F}$ is connected.

Best possible under ETH.

Fomin, Lokshtanov, M., Ramanujan& Saurabh [2015]

Planar $\mathscr{F}$-Deletion admits a deterministic $(2^{O(k)}n)$ algorithm when every graph in $\mathscr{F}$ is connected.

Planar $\mathscr{F}$-Deletion admits an O(nm) randomized algorithm that leads us to a **constant-factor approximation**.

Can we get a deterministic constant-factor approximation algorithm?

Fomin, Lokshtanov, M., Philip & Saurabh [2013]

Planar $\mathscr{F}$-Deletion admits an deterministic algorithm that leads us to a $O(\log^{3/2}(OPT))$ approximation.

Kim, Langer, Paul, Reidl, Rossmanith, Sau, & Sikdar [2013]

Planar $\mathscr{F}$-Deletion admits a deterministic $(2^{O(k)}n^2)$ algorithm.

Many **polynomial kernels** are also known for special cases of $\mathscr{F}$.

Many **polynomial kernels** are also known for special cases of $\mathscr{F}$.

Thomasse [2009]

Feedback Vertex Set has a $O(k^2)$ instance kernel.

Many **polynomial kernels** are also known for special cases of $\mathscr{F}$.

Thomasse [2009]	**Feedback Vertex Set** has a $O(k^2)$ instance kernel.
(Various)	**Vertex Cover** has a $O(k)$ vertex kernel.

Many **polynomial kernels** are also known for special cases of $\mathscr{F}$.

Thomasse [2009]

Feedback Vertex Set has a $O(k^2)$ instance kernel.

(Various)

Vertex Cover has a O(k) vertex kernel.

Cygan, Pilipczuk, Pilipczuk & Wojtaszczyk [2010]

Pathwidth-1-Deletion has a polynomial kernel

Many **polynomial kernels** are also known for special cases of $\mathscr{F}$.

Thomasse [2009]	**Feedback Vertex Set** has a $O(k^2)$ instance kernel.
(Various)	**Vertex Cover** has a O(k) vertex kernel.
Cygan, Pilipczuk, Pilipczuk & Wojtaszczyk [2010]	**Pathwidth-1-Deletion** has a polynomial kernel
Fomin, Lokshtanov, M. & Saurabh [2013]	**Planar $\mathscr{F}$-Deletion** admits a polynomial kernel.

Many **polynomial kernels** are also known for special cases of $\mathscr{F}$.

Thomasse [2009]	**Feedback Vertex Set** has a $O(k^2)$ instance kernel.
(Various)	**Vertex Cover** has a O(k) vertex kernel.
Cygan, Pilipczuk, Pilipczuk & Wojtaszczyk [2010]	**Pathwidth-1-Deletion** has a polynomial kernel
Fomin, Lokshtanov, M. & Saurabh [2013]	**Planar $\mathscr{F}$-Deletion** admits a polynomial kernel.

Best possible under standard assumptions.

The $\mathcal{F}$-Deletion Problem

The $\mathscr{F}$-Deletion Problem

Very little is known beyond the graph minors result.

The $\mathscr{F}$-Deletion Problem

Very little is known beyond the graph minors result.

The most fundamental $\mathscr{F}$-deletion problem that is not accounted for by Planar $\mathscr{F}$-Deletion is **Planarization**.

The $\mathscr{F}$-Deletion Problem

Very little is known beyond the graph minors result.

The most fundamental $\mathscr{F}$-deletion problem that is not accounted for by Planar $\mathscr{F}$-Deletion is **Planarization**.

Marx and Schlotter [2012]

Planarization admits an algorithm with running time $O(2^{g(k)}n^2)$ where $g(k) = k^{O(k3)}$

The $\mathscr{F}$-Deletion Problem

Very little is known beyond the graph minors result.

The most fundamental $\mathscr{F}$-deletion problem that is not accounted for by Planar $\mathscr{F}$-Deletion is **Planarization**.

<u>Marx and Schlotter</u> [2012]	Planarization admits an algorithm with running time $O(2^{g(k)}n^2)$ where $g(k) = k^{O(k3)}$
<u>Kawarabayashi</u> [2009]	Planarization admits an algorithm with running time $O(f(k)n)$ where $f(k)$ is not explicitly specified.

The $\mathscr{F}$-Deletion Problem

Very little is known beyond the graph minors result.

The most fundamental $\mathscr{F}$-deletion problem that is not accounted for by Planar $\mathscr{F}$-Deletion is **Planarization**.

Marx and Schlotter [2012]	Planarization admits an algorithm with running time $O(2^{g(k)}n^2)$ where $g(k) = k^{O(k3)}$
Kawarabayashi [2009]	Planarization admits an algorithm with running time $O(f(k)n)$ where $f(k)$ is not explicitly specified.
Jansen, Lokshtanov & Saurabh [2014]	Planarization admits an algorithm with running time $2^{O(k \log k)}n$, achieving the best combined dependence on k and n.

The $\mathscr{F}$-Deletion Problem

Very little is known beyond the graph minors result.

The most fundamental $\mathscr{F}$-deletion problem that is not accounted for by Planar $\mathscr{F}$-Deletion is **Planarization**.

Marx and Schlotter [2012]	Planarization admits an algorithm with running time $O(2^{g(k)}n^2)$ where $g(k) = k^{O(k3)}$
Kawarabayashi [2009]	Planarization admits an algorithm with running time O(f(k)n) where f(k) is not explicitly specified.
Jansen, Lokshtanov & Saurabh [2014]	Planarization admits an algorithm with running time $2^{O(k \log k)}n$, achieving the best combined dependence on k and n.

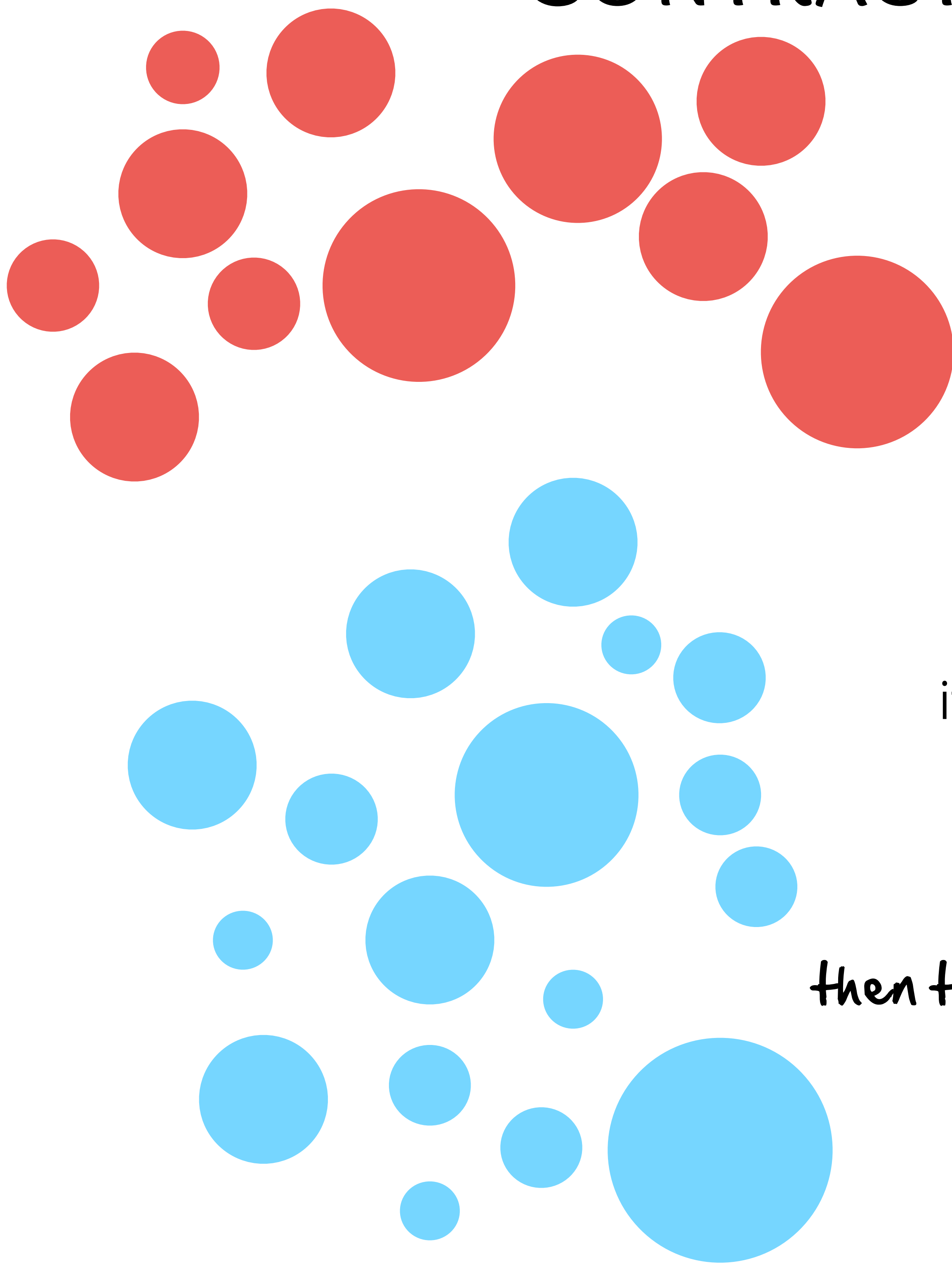
The question of kernels is **completely open.**

Graph Modification Problems

When the operation induces hardness

Contraction Problems

Contraction Problems



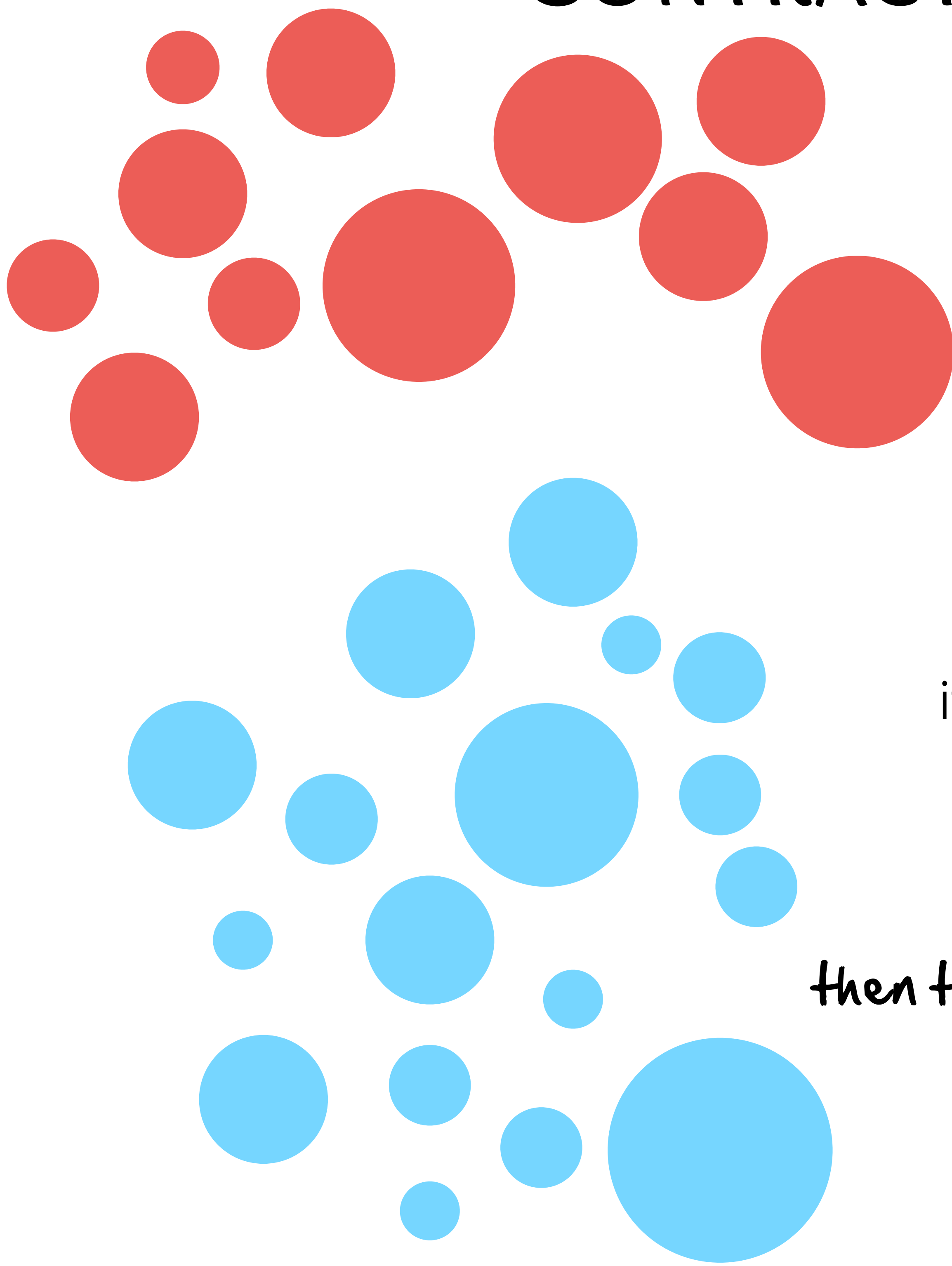
Graph Class or "property", π

Forbidden Subgraph Characterization

A graph admits a **forbidden subgraph characterization**
if, and only if,
it is **closed under taking induced subgraphs (hereditary)**,
that is, the property is preserved under vertex deletions.

If there are finitely many forbidden graphs,
then the corresponding graph modification problem is FPT.

Contraction Problems



Graph Class or "property", π

<u>Forbidden Subgraph Characterization</u>

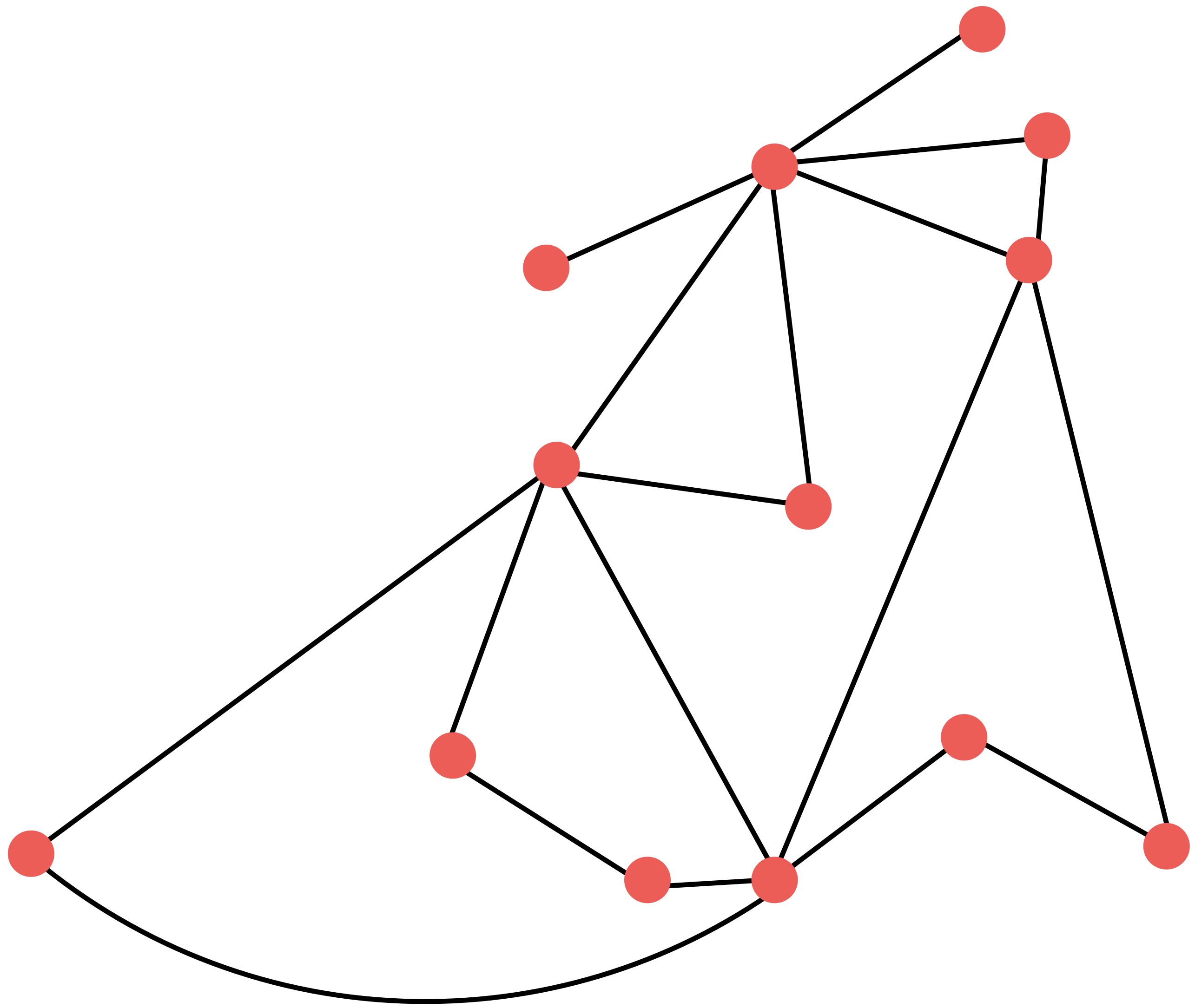
A graph admits a **forbidden subgraph characterization**
if, and only if,
it is **closed under taking induced subgraphs (hereditary)**,
that is, the property is preserved under vertex deletions.

*If there are finitely many forbidden graphs,
then the corresponding graph modification problem is FPT.*

Not true any more!

Contraction To C_4-free graphs

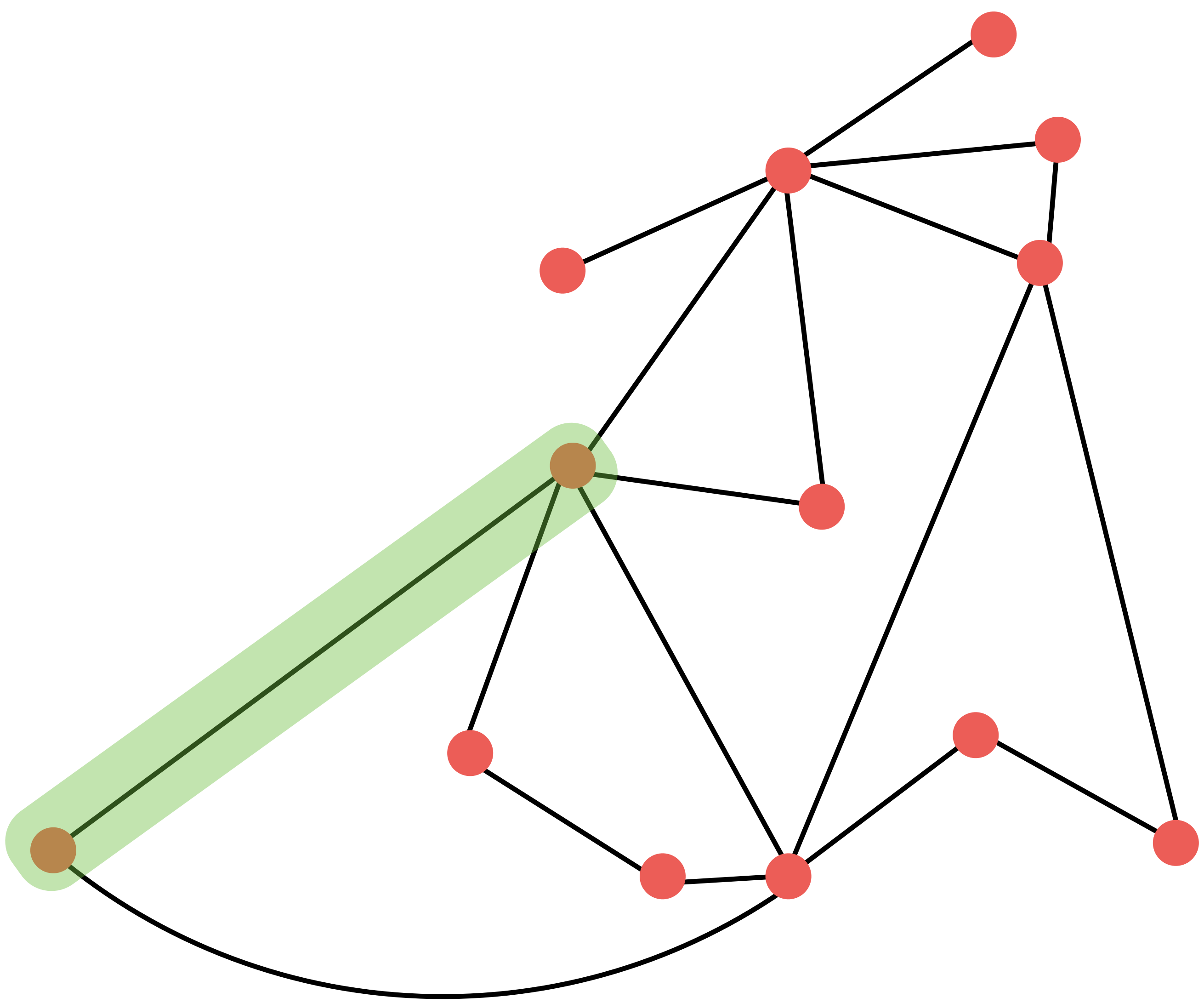
Can G be made C_4-free by the contraction of at most k edges?



W[2]-hard by a simple reduction from Hitting Set.

Contraction To C_4-free graphs

Can G be made C_4-free by the contraction of at most k edges?



W[2]-hard by a simple reduction from Hitting Set.

Lokshtanov, M. & Saurabh [2013]

C_t-free Contraction is **W[2]-hard** if t ⩾ 4 and **FPT** if t⩽ 3.

P_t-free Contraction is **W[2]-hard** if t ⩾ 5 and **FPT** if t ⩽ 4.

Lokshtanov, M. & Saurabh [2013]

C_t-free Contraction is **W[2]-hard** if $t \geq 4$ and **FPT** if $t \leq 3$.

P_t-free Contraction is **W[2]-hard** if $t \geq 5$ and **FPT** if $t \leq 4$.

Cai and Guo [2013]

Chordal Contraction is **W[2]-hard** while Clique contraction is **FPT** but is **unlikely to admit a polynomial kernel**.

Lokshtanov, M. & Saurabh [2013]

C_t-free Contraction is **W[2]-hard** if $t \geqslant 4$ and **FPT** if $t \leqslant 3$.

P_t-free Contraction is **W[2]-hard** if $t \geqslant 5$ and **FPT** if $t \leqslant 4$.

Cai and Guo [2013]

Chordal Contraction is **W[2]-hard** while Clique contraction is **FPT** but is **unlikely to admit a polynomial kernel**.

Characeterizations are open.

Lokshtanov, M. & Saurabh [2013]

C_t-free Contraction is **W[2]-hard** if t ⩾ 4 and **FPT** if t⩽ 3.
P_t-free Contraction is **W[2]-hard** if t ⩾ 5 and **FPT** if t ⩽ 4.

Cai and Guo [2013]

Chordal Contraction is **W[2]-hard** while Clique contraction is **FPT** but is **unlikely to admit a polynomial kernel**.

Characeterizations are open.

Heggernes, van't Hof, Lokshtanov & Paul [2011]

Contracting to Bipartite graphs is **fixed-parameter tractable** with a double-exponential running time.

Lokshtanov, M. & Saurabh [2013]

C_t-free Contraction is **W[2]-hard** if $t \geq 4$ and **FPT** if $t \leq 3$.

P_t-free Contraction is **W[2]-hard** if $t \geq 5$ and **FPT** if $t \leq 4$.

Cai and Guo [2013]

Chordal Contraction is **W[2]-hard** while Clique contraction is **FPT** but is **unlikely to admit a polynomial kernel**.

Characeterizations are open.

Heggernes, van't Hof, Lokshtanov & Paul [2011]

Contracting to Bipartite graphs is **fixed-parameter tractabl**e with a double-exponential running time.

Polynomial kernels are open.

Lokshtanov, M. & Saurabh [2013]

C_t-free Contraction is **W[2]-hard** if t ⩾ 4 and **FPT** if t⩽ 3.
P_t-free Contraction is **W[2]-hard** if t ⩾ 5 and **FPT** if t ⩽ 4.

Cai and Guo [2013]

Chordal Contraction is **W[2]-hard** while Clique contraction is **FPT** but is **unlikely to admit a polynomial kernel**.

Characeterizations are open.

Heggernes, van't Hof, Lokshtanov & Paul [2011]

Contracting to Bipartite graphs is **fixed-parameter tractabl**e with a double-exponential running time.

Polynomial kernels are open.

Heggernes, van't Hof, Lévêque, Lokshtanov & Paul [2011]

Contracting to paths is **FPT** and has a **linear-vertex kernel**; while contracting to trees is **FPT** but is **unlikely to admit a polynomial kernel**.

Graph Modification Problems

Completion

Chordal Completion

Chordal Completion

Yannakakis [1981]

The minimum fill-in problem is **NP-complete** (was left open in the first edition of Garey and Johnson).

Chordal Completion

Yannakakis [1981]

The minimum fill-in problem is **NP-complete** (was left open in the first edition of Garey and Johnson).

Kaplan, Shamir & Tarjan [1999]

Chordal Completion is **fixed-parameter tractable (FPT)** with a running time of $O(k^6 16^k + k^2 mn)$.

Chordal Completion

Yannakakis [1981]

The minimum fill-in problem is **NP-complete** (was left open in the first edition of Garey and Johnson).

Kaplan, Shamir & Tarjan [1999]

Chordal Completion is **fixed-parameter tractable (FPT)** with a running time of $O(k^6 16^k + k^2 mn)$.

Bodlaender, Heggernes, & Villanger [2011]

Chordal Completion is **fixed-parameter tractable (FPT)** with a running time of $O(2.36^k + k^2 mn)$.

Chordal Completion

Yannakakis [1981]	The minimum fill-in problem is **NP-complete** (was left open in the first edition of Garey and Johnson).
Kaplan, Shamir & Tarjan [1999]	Chordal Completion is **fixed-parameter tractable (FPT)** with a running time of $O(k^6 16^k + k^2 mn)$.
Bodlaender, Heggernes, & Villanger [2011]	Chordal Completion is **fixed-parameter tractable (FPT)** with a running time of $O(2.36^k + k^2 mn)$.
Fomin & Villanger [2013]	Chordal Completion admits a **sub exponential** parameterized algorithm with a running time of $O(2^{\sqrt{k} \log k} + k^2 mn)$.

Interval Completion

Interval Completion

Kaplan, Shamir & Tarjan
[1999]

Chordal Completion, Strongly Chordal Completion, and Proper Interval Completion are fixed-parameter tractable (FPT).

Interval Completion

Kaplan, Shamir & Tarjan
[1999]

Chordal Completion, Strongly Chordal Completion, and Proper Interval Completion are fixed-parameter tractable (FPT).

Heggernes, Paul, Telle & Villanger
[2007]

Interval Completion is **fixed-parameter tractable** (FPT), with a running time of $O(k^{2k}n^3m)$.

Interval Completion

Kaplan, Shamir & Tarjan [1999]

Chordal Completion, Strongly Chordal Completion, and Proper Interval Completion are fixed-parameter tractable (FPT).

Heggernes, Paul, Telle & Villanger [2007]

Interval Completion is **fixed-parameter tractable** (FPT), with a running time of $O(k^{2k}n^3m)$.

Cao [2007]

Interval Completion has a **single-exponential** parameterized algorithm, with a running time of $O(6^k(n+m))$.

Interval Completion

Kaplan, Shamir & Tarjan [1999]	Chordal Completion, Strongly Chordal Completion, and Proper Interval Completion are fixed-parameter tractable (FPT).
Heggernes, Paul, Telle & Villanger [2007]	Interval Completion is **fixed-parameter tractable** (FPT), with a running time of $O(k^{2k}n^3m)$.
Cao [2007]	Interval Completion has a **single-exponential** parameterized algorithm, with a running time of $O(6^k(n+m))$.
Bliznets, Fomin, Pilipczuk & Pilipczuk [2014]	Interval Completion admits a **subexponential** parameterized algorithm with a running time of $k^{O(\sqrt{k})}n^{O(1)}$.

Graph Modification Problems

A Summary

Edge Deletions

Kernels on Planar Graphs & the Meta-Kernel project

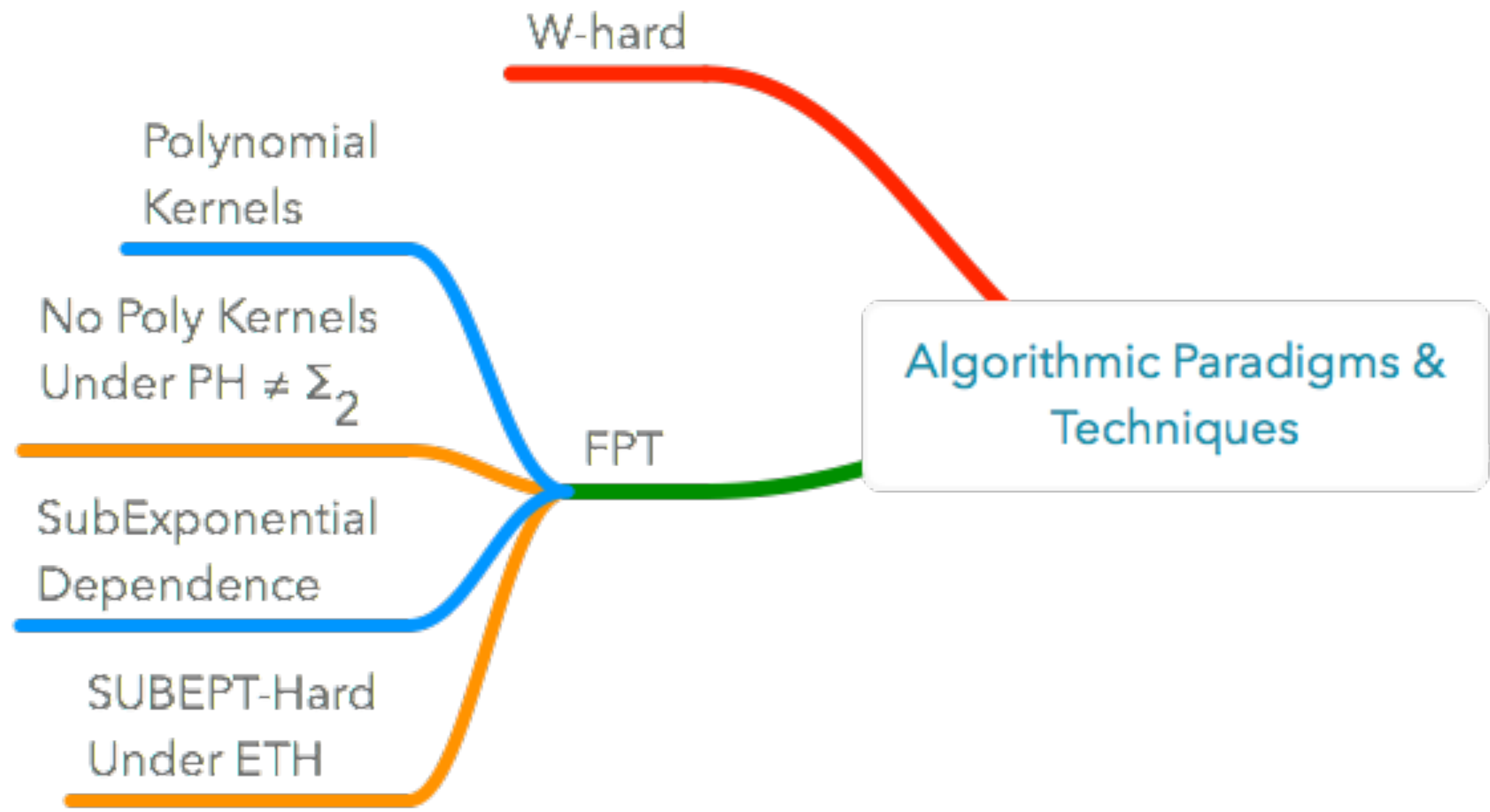
Special Graph Classes, eg, Feedback Arc Set on Tournaments

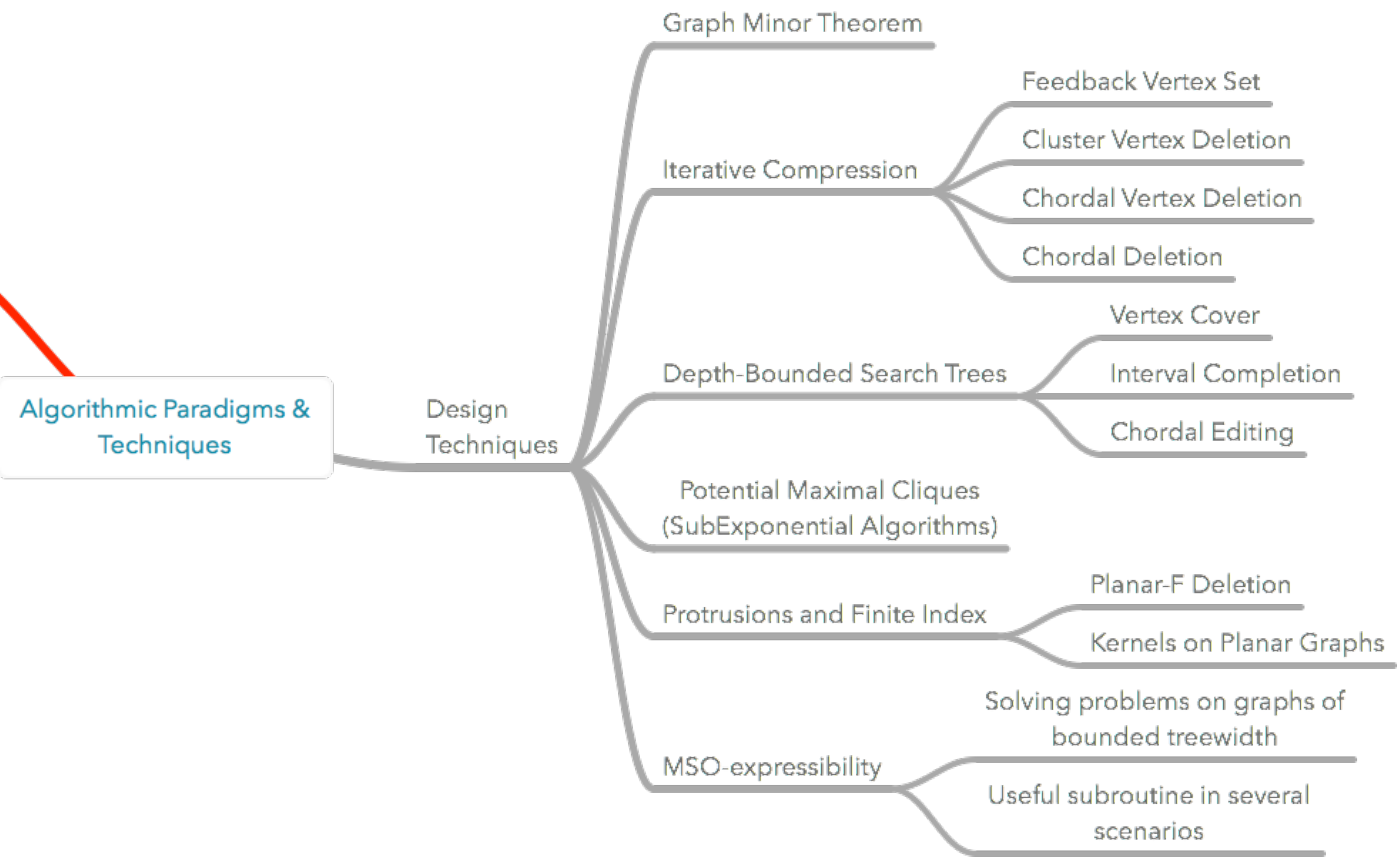
Connectivity Augmentation

The descriptive complexity of graph modification

Structural parameters and other objective functions

Backdoor Sets!


Algorithmic Paradigms & Techniques
W-hard
FPT
Polynomial Kernels
No Poly Kernels Under PH ≠ Σ2
SubExponential Dependence
SUBEPT-Hard Under ETH


Algorithmic Paradigms & Techniques
Design Techniques
Graph Minor Theorem
Iterative Compression
Feedback Vertex Set
Cluster Vertex Deletion
Chordal Vertex Deletion
Chordal Deletion
Depth-Bounded Search Trees
Vertex Cover
Interval Completion
Chordal Editing
Potential Maximal Cliques (SubExponential Algorithms)
Protrusions and Finite Index
Planar-F Deletion
Kernels on Planar Graphs
MSO-expressibility
Solving problems on graphs of bounded treewidth
Useful subroutine in several scenarios