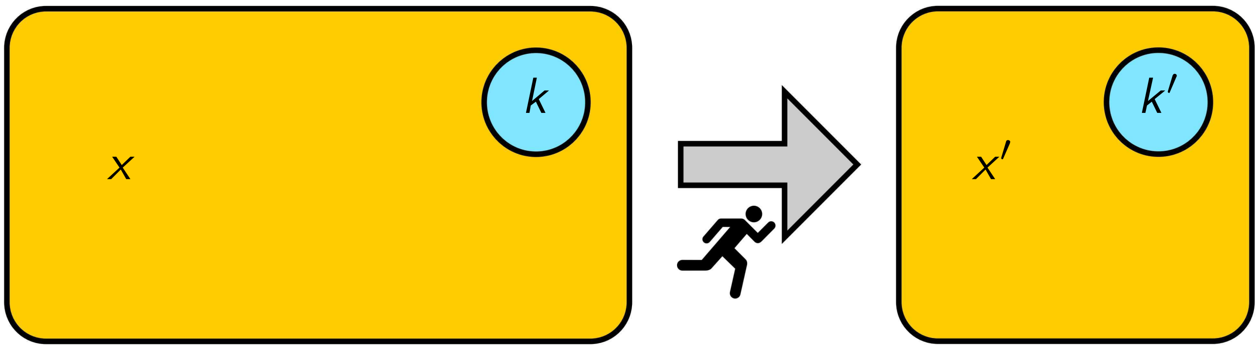
Kernelization

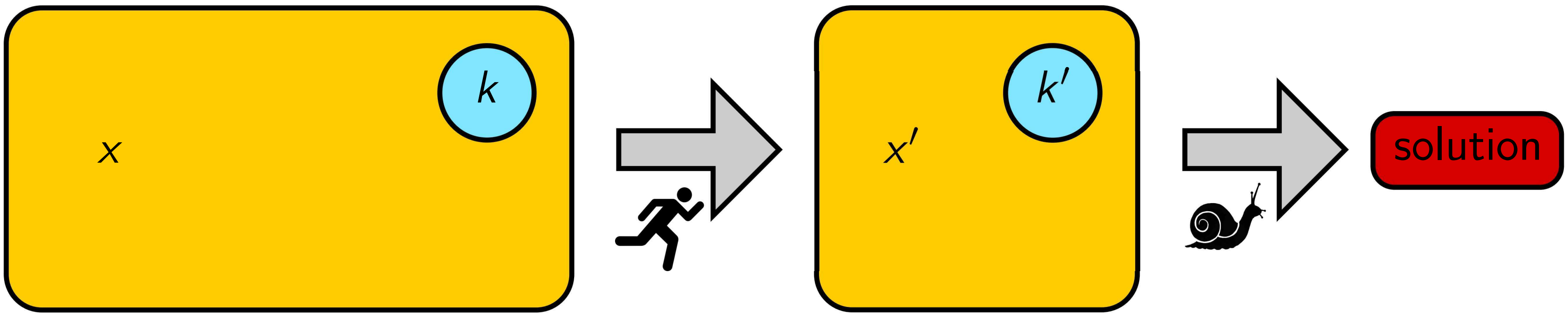
Data reductions—with a guarantee

- **Kernelization** is a method for parameterized preprocessing:
  - We want to efficiently reduce the size of the instance (x, k) to an equivalent instance with size bounded by $f(k)$.
- A basic way of obtaining FPT algorithms:
  - Reduce the size of the instance to $f(k)$ in polynomial time and then apply any brute force algorithm to the shrunk instance.
- Kernelization is also a rigorous mathematical analysis of efficient preprocessing.



Data reductions—with a guarantee

- **Kernelization** is a method for parameterized preprocessing:
  - We want to efficiently reduce the size of the instance (x, k) to an equivalent instance with size bounded by $f(k)$.
- A basic way of obtaining FPT algorithms:
  - Reduce the size of the instance to $f(k)$ in polynomial time and then apply any brute force algorithm to the shrunk instance.
- Kernelization is also a rigorous mathematical analysis of efficient preprocessing.



Vertex Cover

Input A graph G = (V,E) with n vertices, m edges, and k.

Vertex Cover

Input A graph $G = (V,E)$ with n vertices, m edges, and k.

Is there a subset of vertices S of size at most k that intersects all the edges? **Question**

VERTEX COVER

Input

A graph G = (V,E) with n vertices, m edges, and k.

Question

Is there a subset of vertices S of size at most k that intersects all the edges?

What if a vertex has more than k neighbors?

Vertex Cover

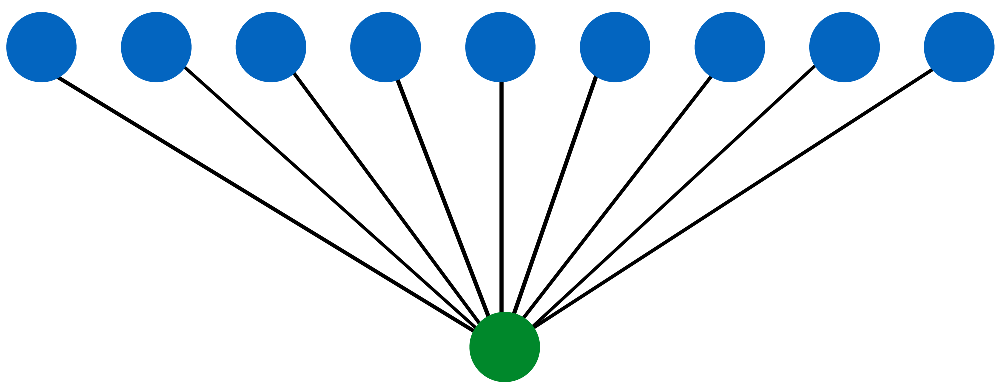
Input

A graph $G = (V,E)$ with n vertices, m edges, and k.

Is there a subset of vertices S of size at most k that intersects all the edges?

Question

What if a vertex has more than k neighbors?

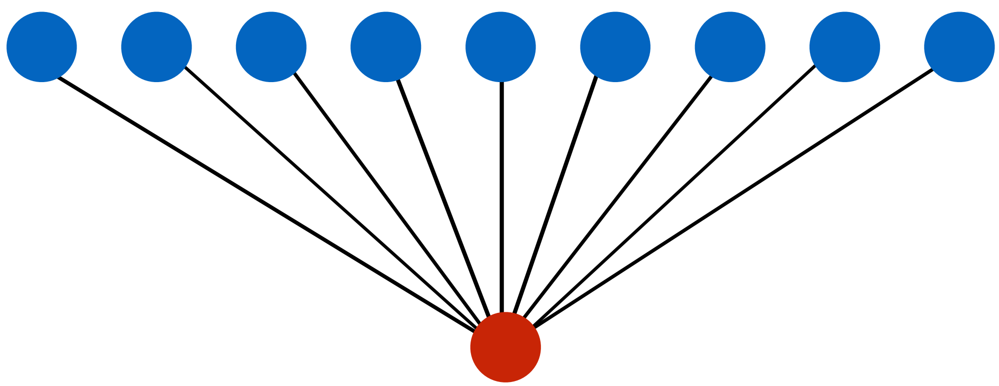


Vertex Cover

Input A graph G = (V,E) with n vertices, m edges, and k.

Is there a subset of vertices S of size at most k that intersects all the edges? Question

What if a vertex has more than k neighbors?

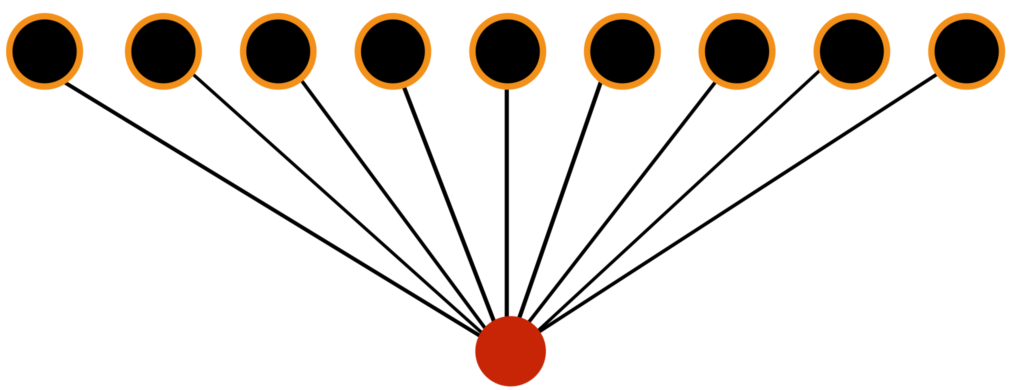


Vertex Cover

Input A graph G = (V,E) with n vertices, m edges, and k.

Is there a subset of vertices S of size at most k that intersects all the edges? Question

What if a vertex has more than k neighbors?



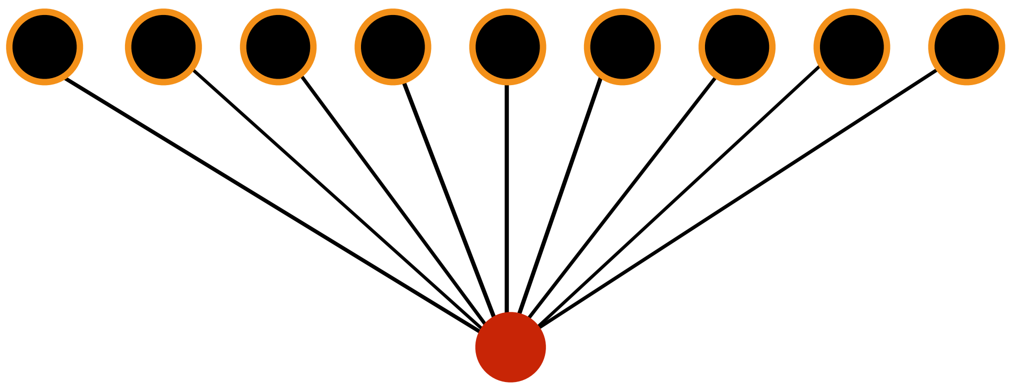
Vertex Cover

Input A graph G = (V,E) with n vertices, m edges, and k.

Is there a subset of vertices S of size at most k that intersects all the edges? **Question**

What if a vertex has more than k neighbors?

We cannot afford to leave v out of any vertex cover of size at most k.

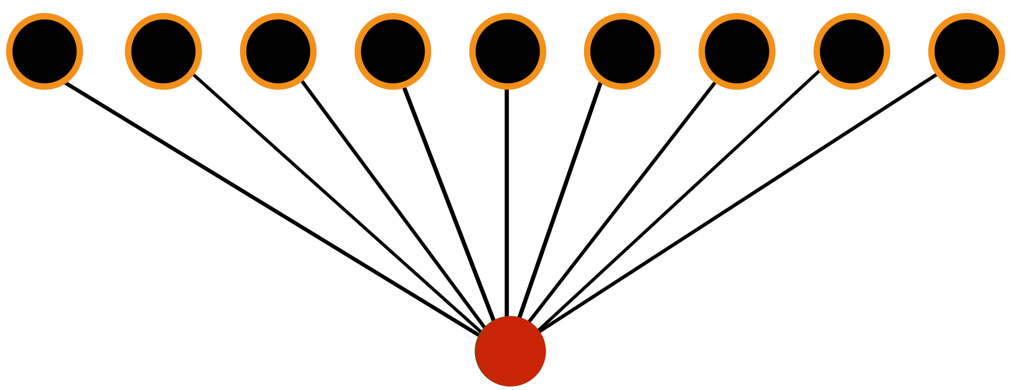


Vertex Cover

Input A graph G = (V,E) with n vertices, m edges, and k.

Is there a subset of vertices S of size at most k that intersects all the edges? Question

If a vertex has more than k neighbors,



Vertex Cover

Input

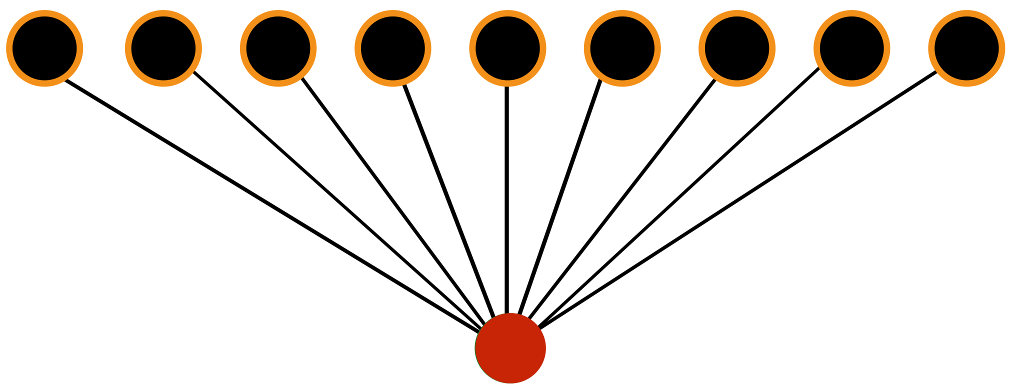
A graph G = (V,E) with n vertices, m edges, and k.

Question

Is there a subset of vertices S of size at most k that intersects all the edges?

If a vertex has more than k neighbors,

delete it from the graph and reduce the budget by one.

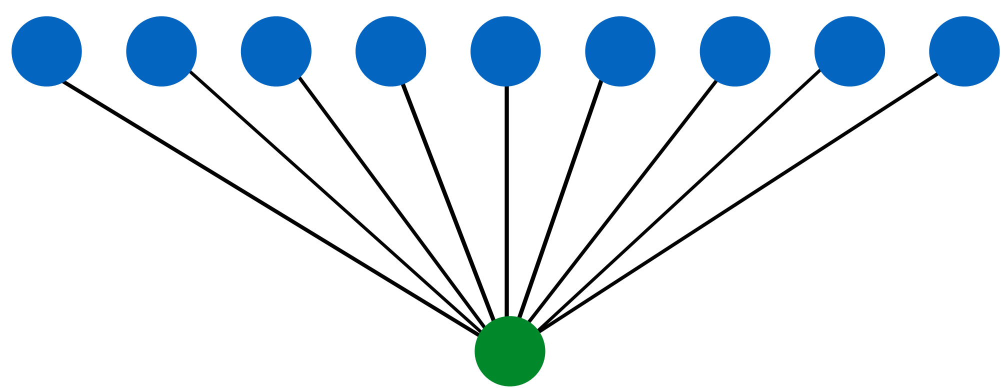


Vertex Cover

Input A graph G = (V,E) with n vertices, m edges, and k.

Is there a subset of vertices S of size at most k that intersects all the edges? Question

When we have nothing more to do…



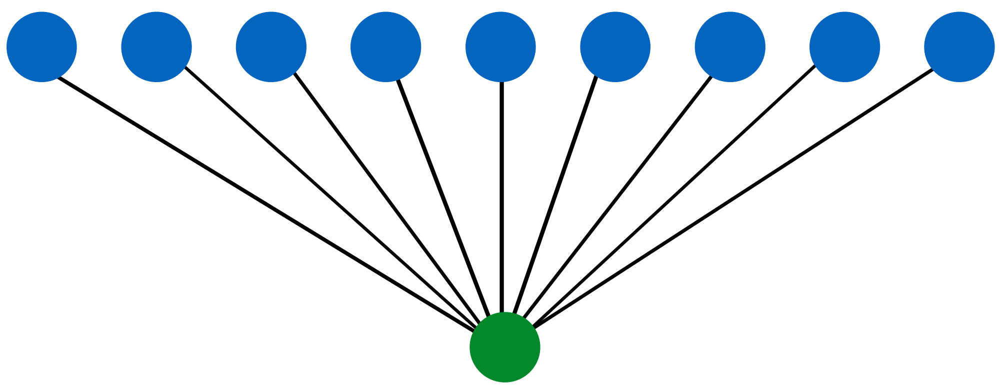
Vertex Cover

Input A graph $G = (V,E)$ with n vertices, m edges, and k.

Is there a subset of vertices S of size at most k that intersects all the edges? Question

When we have nothing more to do…

every vertex has degree at most k.



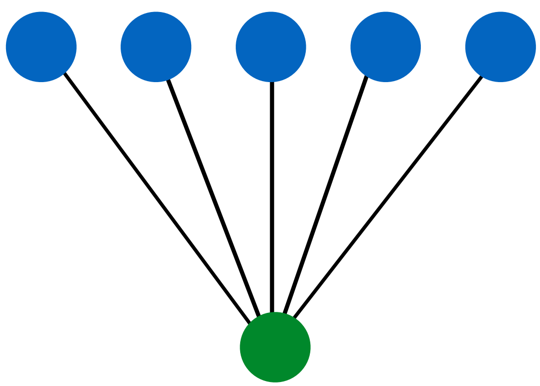
VERTEX COVER

Input A graph G = (V,E) with n vertices, m edges, and k.

Is there a subset of vertices S of size at most k that intersects all the edges? **Question**

When we have nothing more to do…

every vertex has degree at most k.

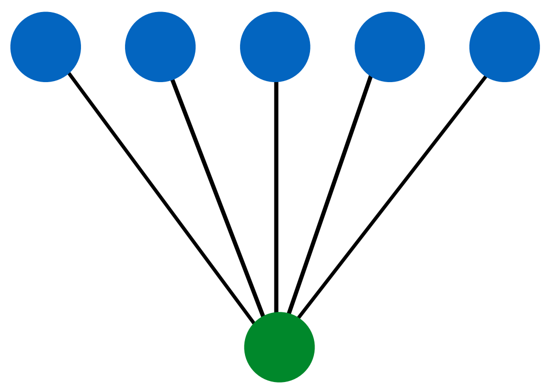


Vertex Cover

Input A graph G = (V,E) with n vertices, m edges, and k.

Is there a subset of vertices S of size at most k that intersects all the edges? Question

If the graph has more than k^2 edges,



Vertex Cover

Input

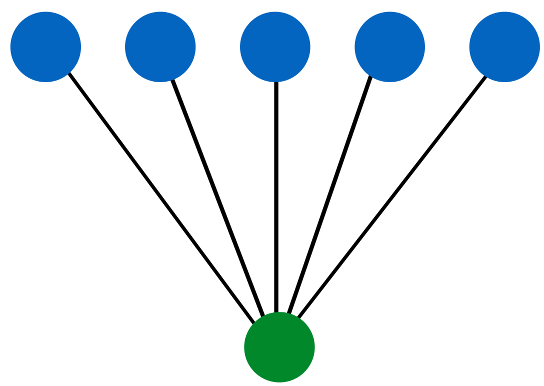
A graph G = (V,E) with n vertices, m edges, and k.

Is there a subset of vertices S of size at most k that intersects all the edges?

Question

If the graph has more than k^2 edges,

reject the instance.



Vertex Cover

Input

A graph G = (V,E) with n vertices, m edges, and k.

Is there a subset of vertices S of size at most k that intersects all the edges?

Question

Otherwise:

Vertex Cover

Input

A graph G = (V,E) with n vertices, m edges, and k.

Is there a subset of vertices S of size at most k that intersects all the edges?

Question

Otherwise:

the number of edges is at most k^2.

Vertex Cover

Input A graph G = (V,E) with n vertices, m edges, and k.

Is there a subset of vertices S of size at most k that intersects all the edges? Question

Otherwise:

the number of edges is at most k^2.

Vertices?

Vertex Cover

Input A graph $G = (V,E)$ with n vertices, m edges, and k.

Is there a subset of vertices S of size at most k that intersects all the edges? Question

Otherwise:

the number of edges is at most k^2.

Vertices?

k^2 edges can be involved in at most $2k^2$ vertices.
Throw away isolated vertices.

Vertex Cover

Input A graph $G = (V,E)$ with n vertices, m edges, and k.

Is there a subset of vertices S of size at most k that intersects all the edges? Question

This implies a kernel with at most $2k^2$ vertices and k^2 edges.

Vertex Cover

Input A graph G = (V,E) with n vertices, m edges, and k.

Is there a subset of vertices S of size at most k that intersects all the edges? **Question**

This implies a kernel with at most $2k^2$ vertices and k^2 edges.

😀

Vertex Cover

Kernelization: formal definition

- Let $P \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized probem and $f : \mathbb{N} \to \mathbb{N}$ a computable function.
- A **kernel** for P of size f is an algorithm that, given (x, k), takes time polynomial in $|x| + k$ and outputs an instance (x', k') such that
  - $(x, k) \in P \iff (x', k') \in P$
  - $|x'| \leq f(k)$, $k' \leq f(k)$.
- A **polynomial kernel** is a kernel whose function f is polynomial.

Kernelization: formal definition

- Let $P \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized probem and $f : \mathbb{N} \to \mathbb{N}$ a computable function.
- A **kernel** for P of size f is an algorithm that, given (x, k), takes time polynomial in $|x| + k$ and outputs an instance (x', k') such that
  - $(x, k) \in P \iff (x', k') \in P$
  - $|x'| \leq f(k)$, $k' \leq f(k)$.
- A **polynomial kernel** is a kernel whose function f is polynomial.

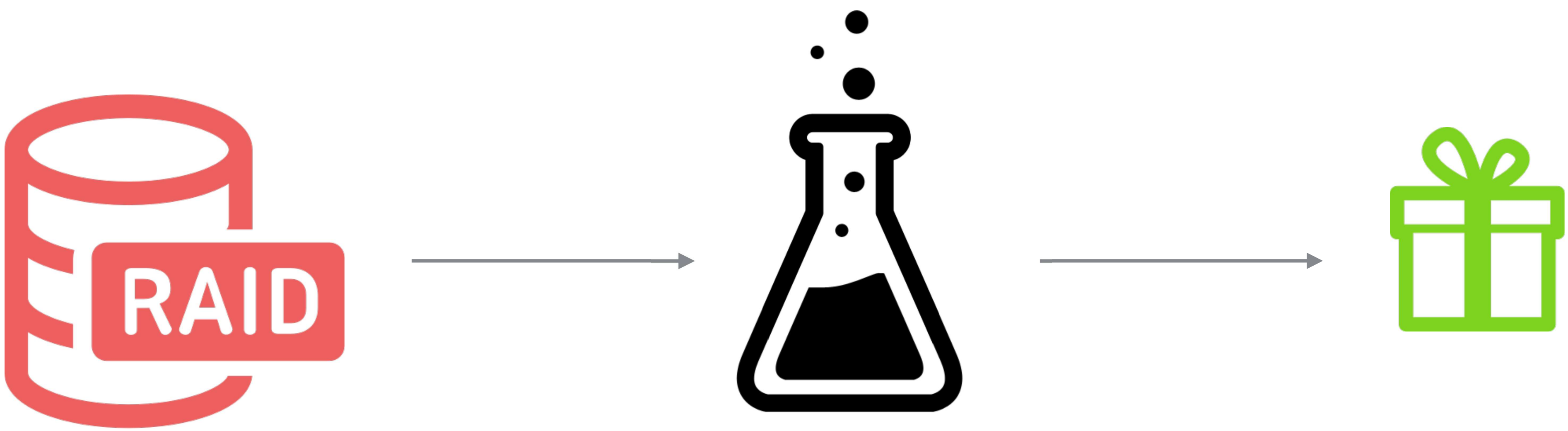
Which parameterized problems have kernels?

$$(x, k) \longrightarrow (x', k')$$

$|x|^{O(1)}$ time

SMALL

SANE



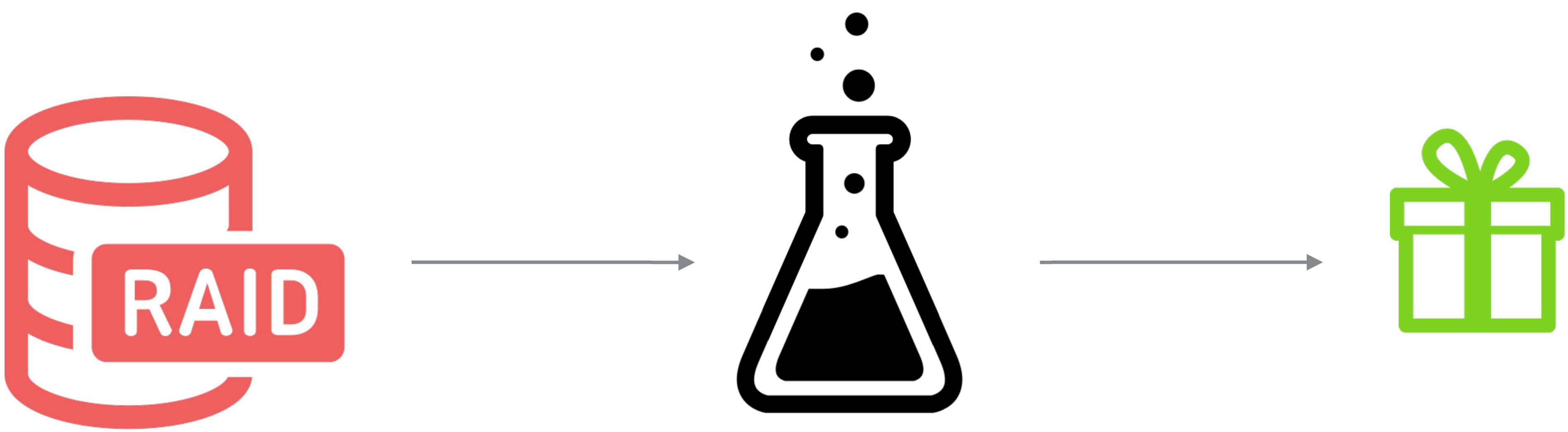
Preprocess

$$(x, k) \longrightarrow (x', k')$$

$|x|^{O(1)}$ time

$|x'| = f(k)$ and $k \leqslant k'$

SANE



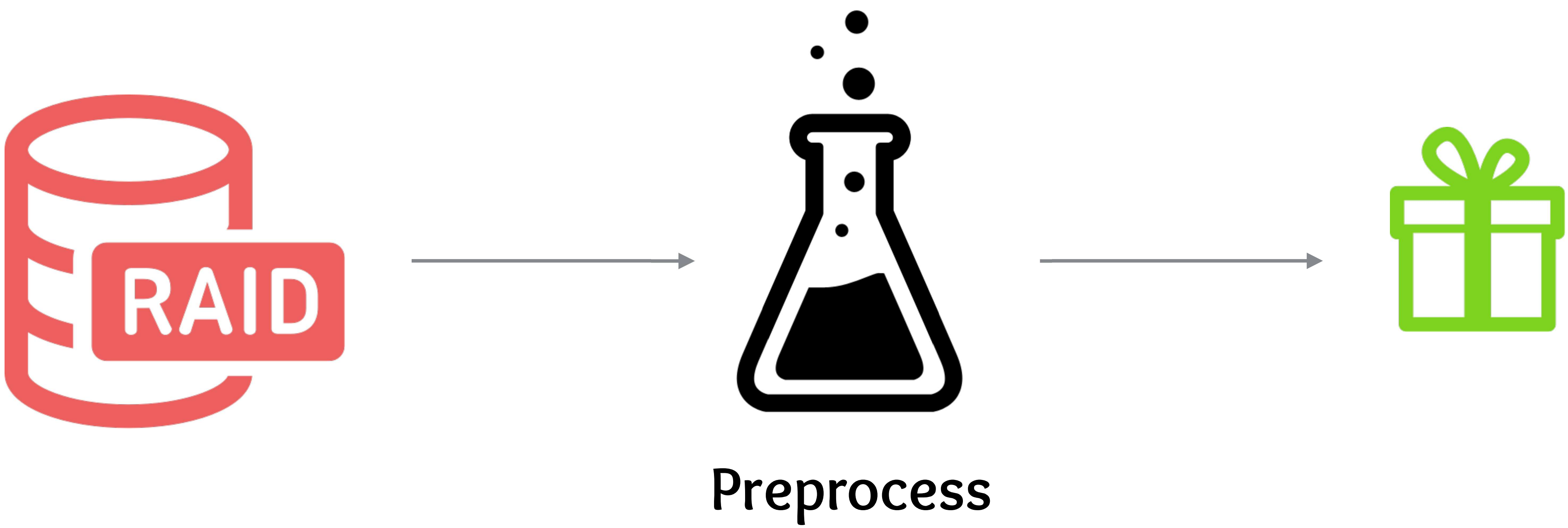
Preprocess

$$(x, k) \longrightarrow (x', k')$$

$|x|^{O(1)}$ **time**

$|x'| = f(k)$ **and** $k \leqslant k'$

$$(x, k) \equiv (x', k')$$



Max Sat

Input A CNF Formula over n variables with m clauses.

Max Sat

Input A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses? **Question**

Max Sat

Input

A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses?

Question

What if k is at most m/2?

Max Sat

Input A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses? **Question**

What if k is at most m/2?

Max Sat

Input A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses? Question

What if k is at most m/2?

Max Sat

Input A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses? Question

What if k is at most m/2?

Say YES.

MAX SAT

Input A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses? Question

What if k is at most m/2?

Say YES.

k > m/2?

Max Sat

Input A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses? Question

What if k is at most m/2?

Say YES.

k > m/2?

The number of clauses is bounded by 2k.

Max Sat

Input

A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses?

Question

VARIABLES

Max Sat

Input A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses? Question

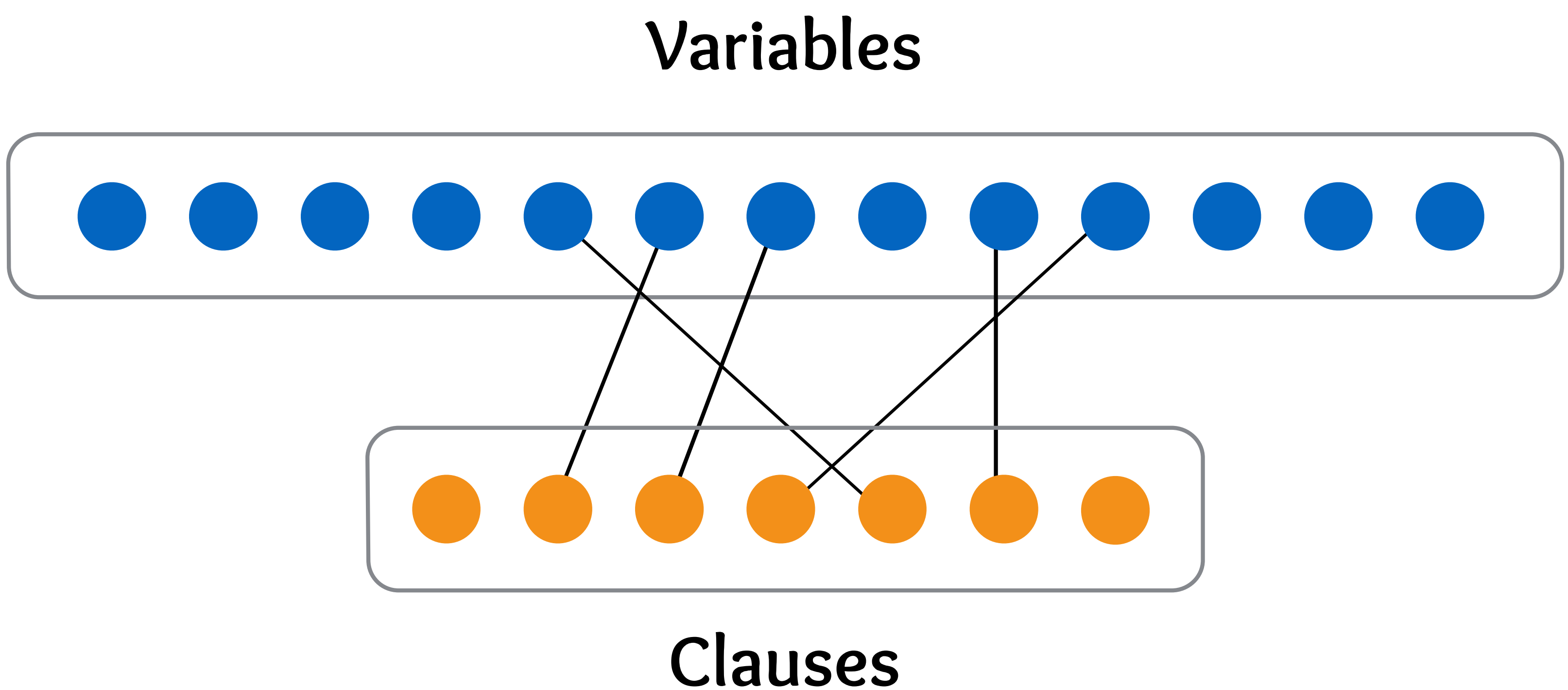
GOAL

We have at most k variables left.

Max Sat

Input A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses? **Question**



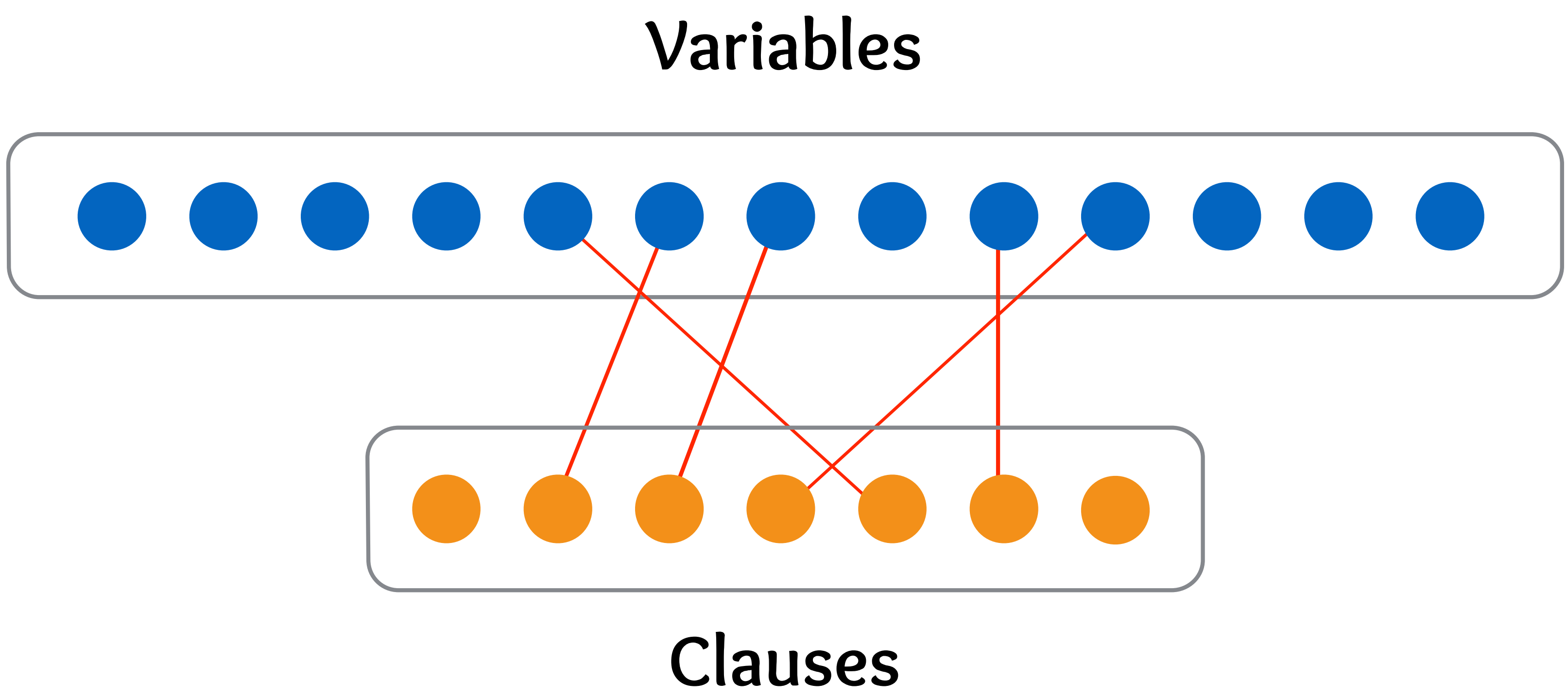
Max Sat

Input

A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses?

Question



Max Sat

Input A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses? **Question**

If we have at most k variables - nothing to do.

If we have at least k variables and we have a matching
from Variables — Clauses
then we can say YES.

Max Sat

Input A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses? Question

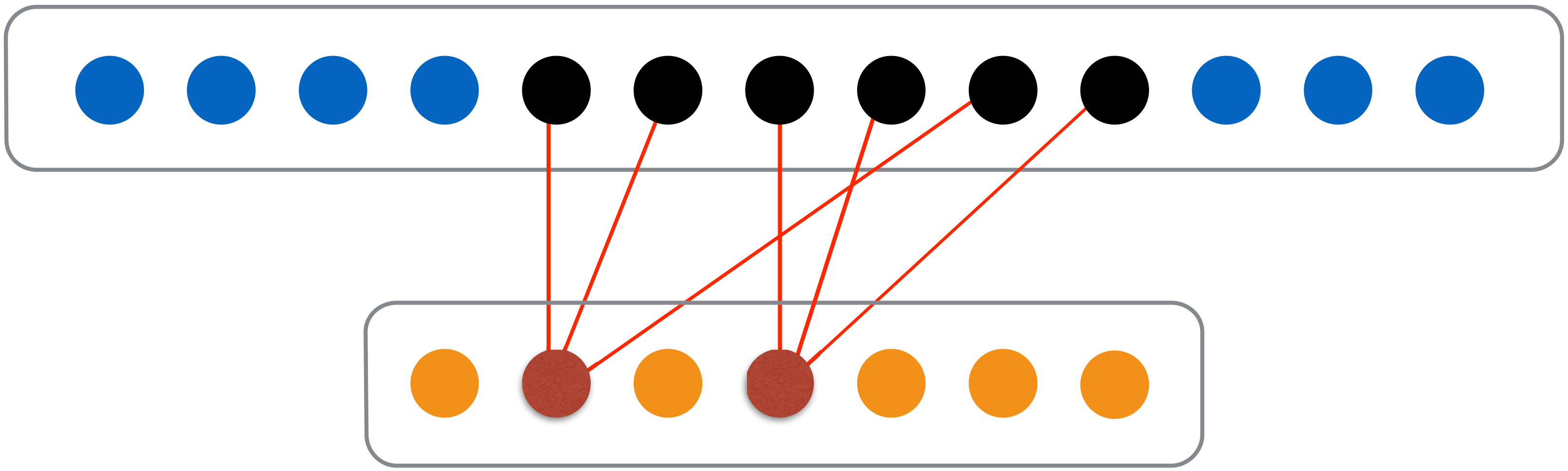
Else, we have at least k variables,
but no matching from Variables — Clauses.

Max Sat

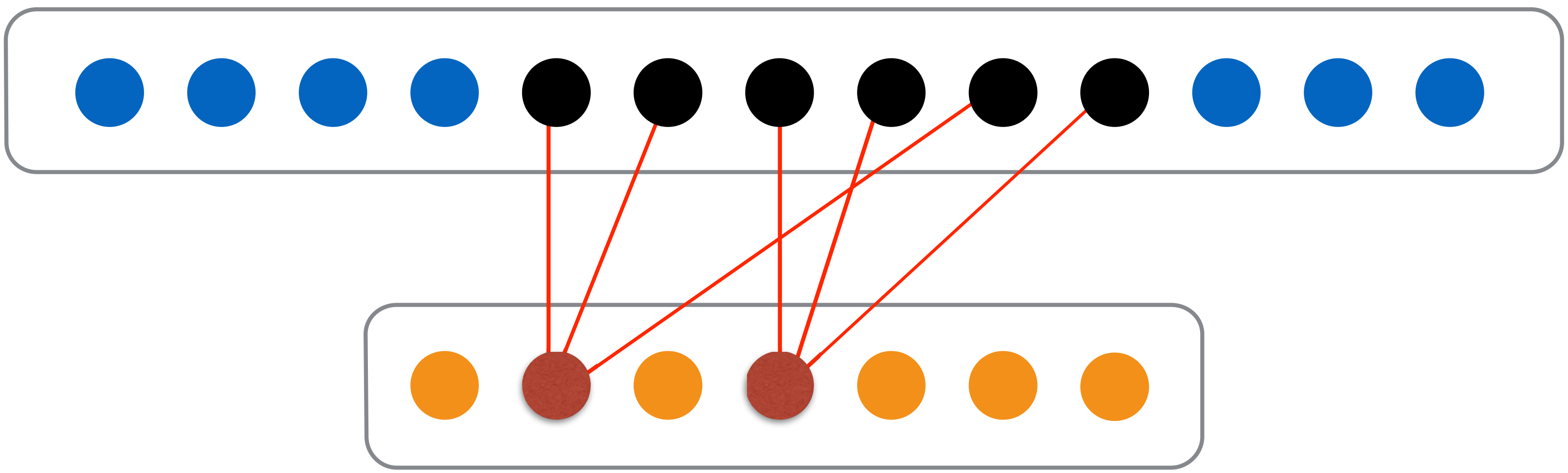
Hall's Theorem

If, in a bipartite graph with parts A and B,
there is no matching from A to B,
then there is a subset X of A
such that

$$|N(X)| < |X|$$

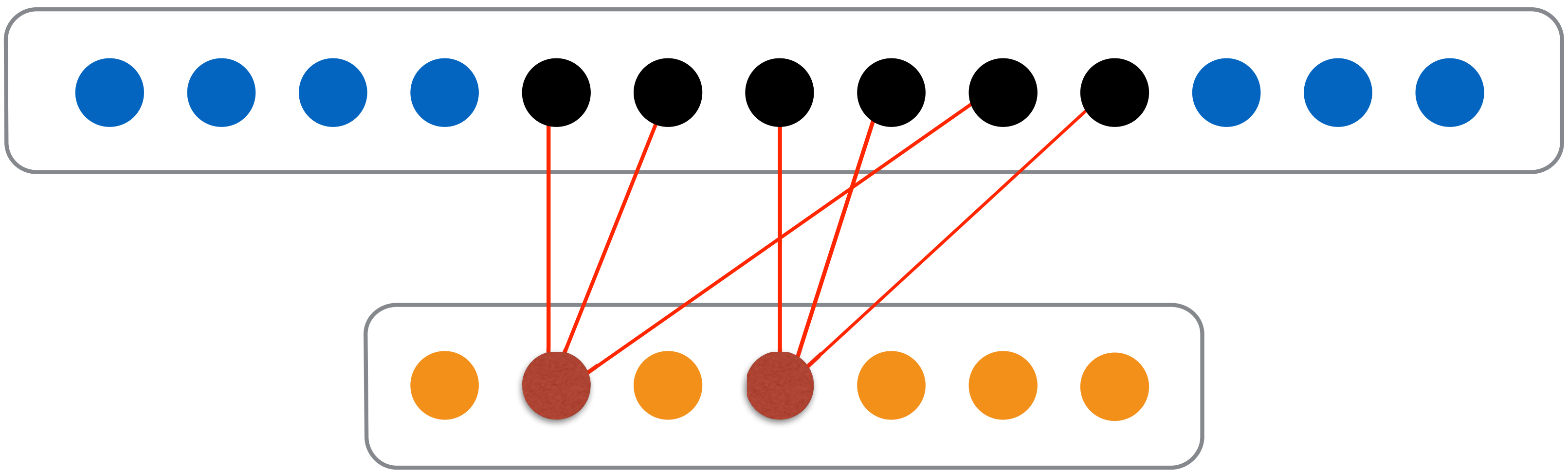


Hall's Theorem



Hall's Theorem

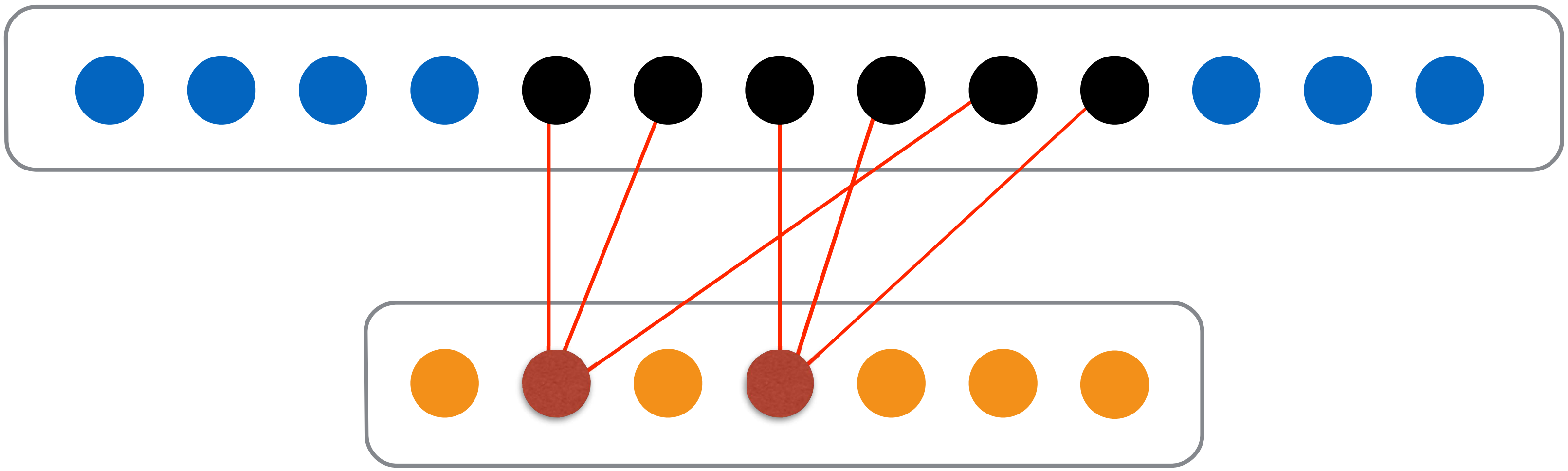
Such an "obstructing set" can be computed in polynomial time.



Hall's Theorem

[inclusion-minimal]

Such an "obstructing set" can be computed in polynomial time.



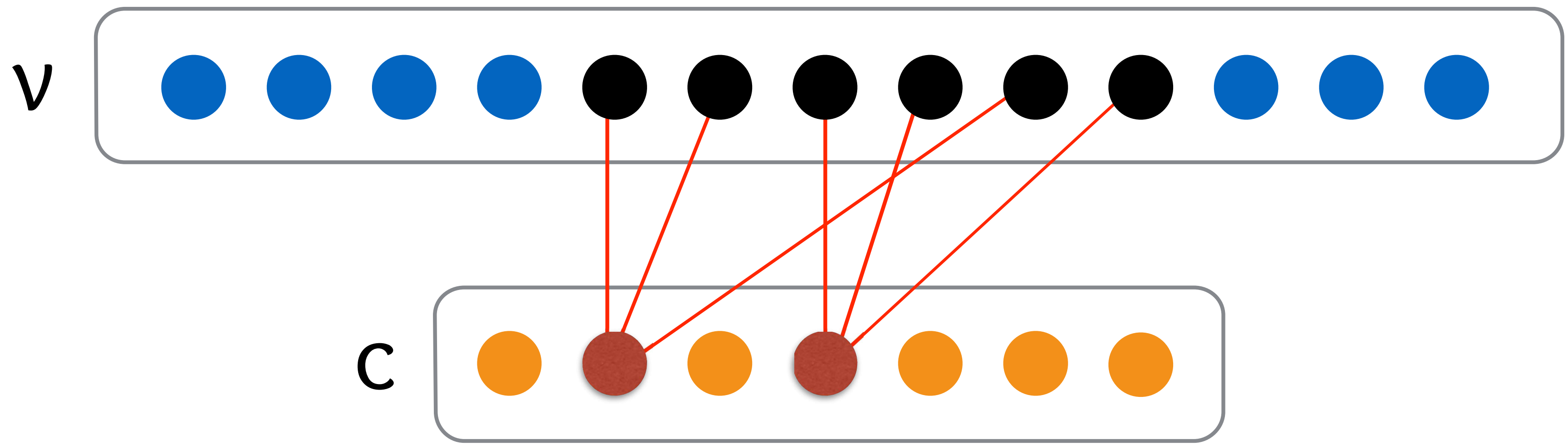
Hall's Theorem

Input A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses? Question

[inclusion-minimal]

Such an "obstructing set" can be computed in polynomial time.



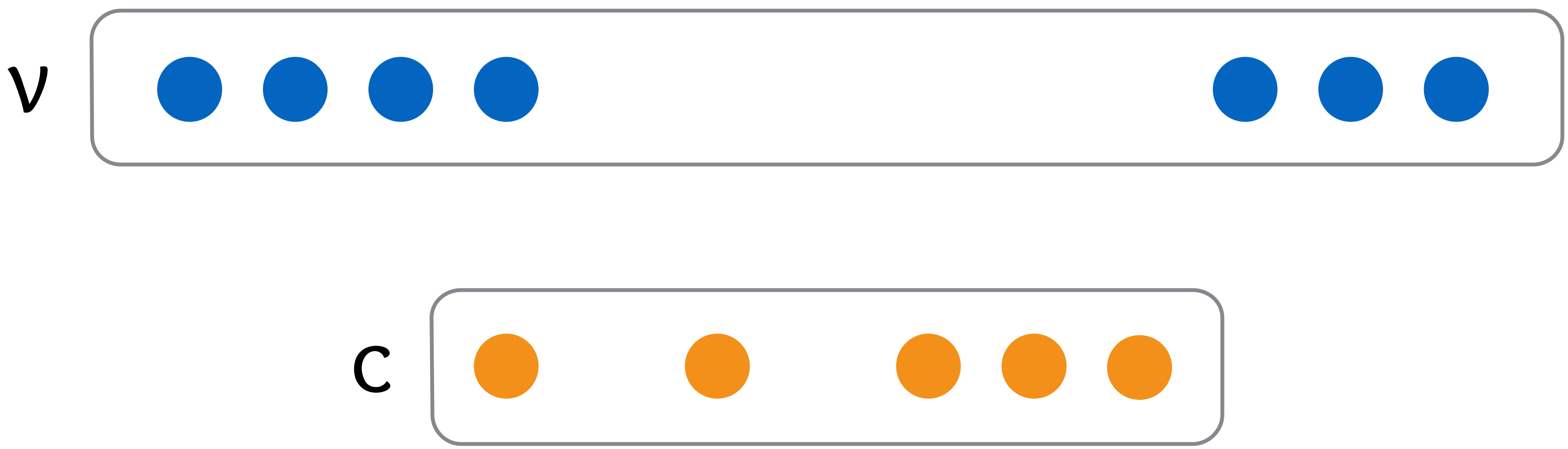
MAX SAT

Input A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses? **Question**

[inclusion-minimal]

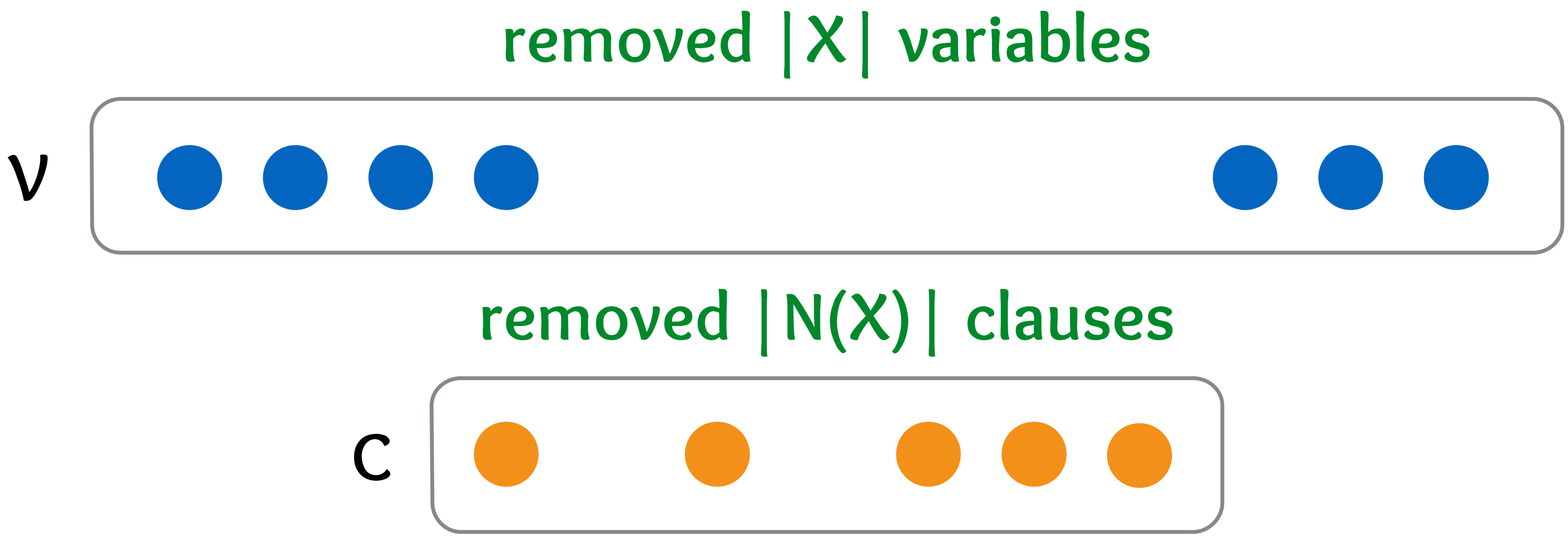
Such an "obstructing set" can be computed in polynomial time.



MAX SAT

Input A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses? Question

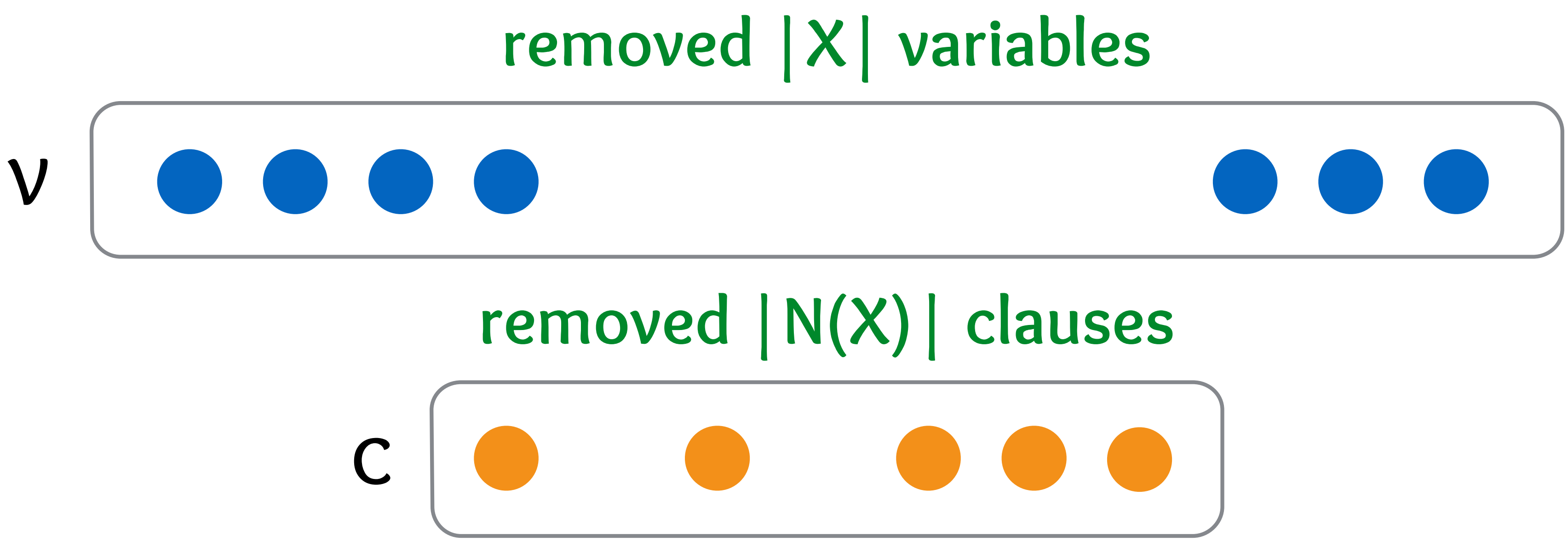


Max Sat

Input A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses? **Question**

Ask now if $k - |N(X)|$ clauses can be satisfied.



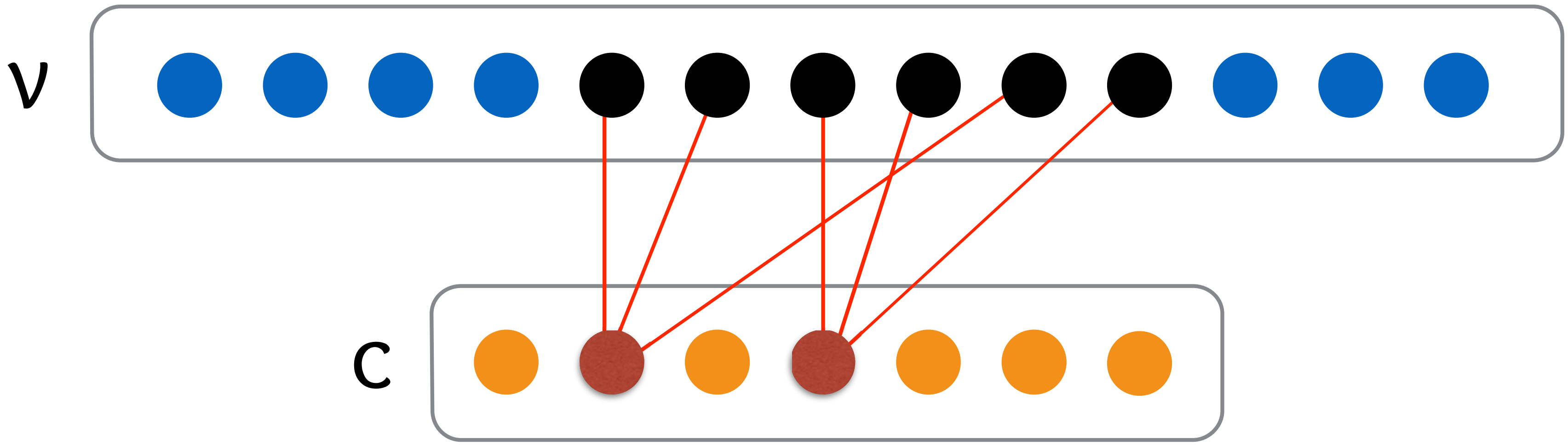
MAX SAT

Input

A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses?

Question

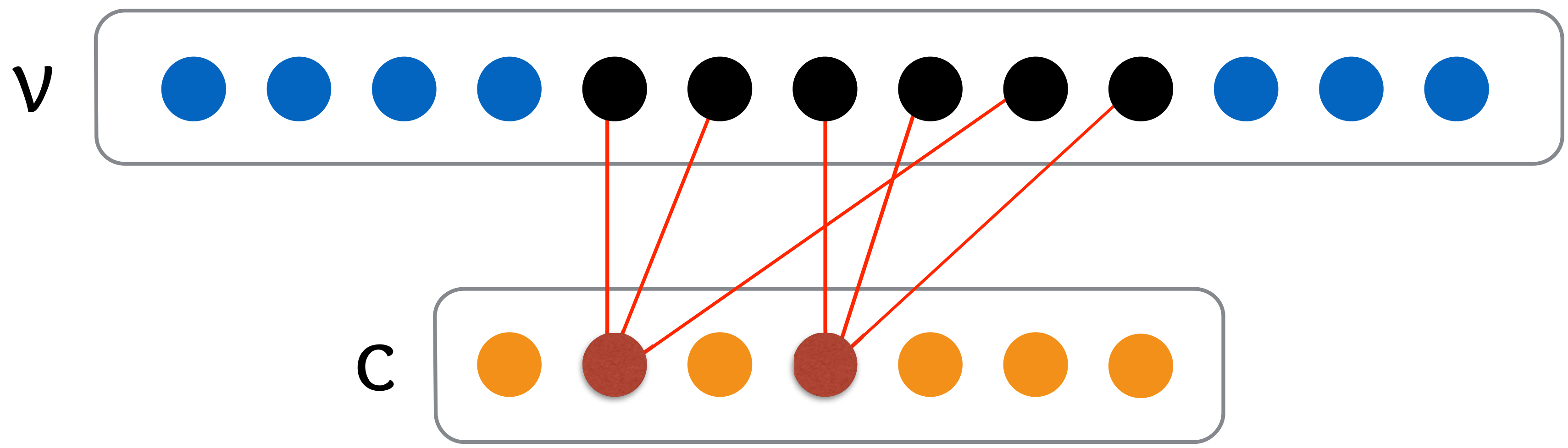


Max Sat

Input — A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses? — Question

We removed an inclusion-minimal violating set.

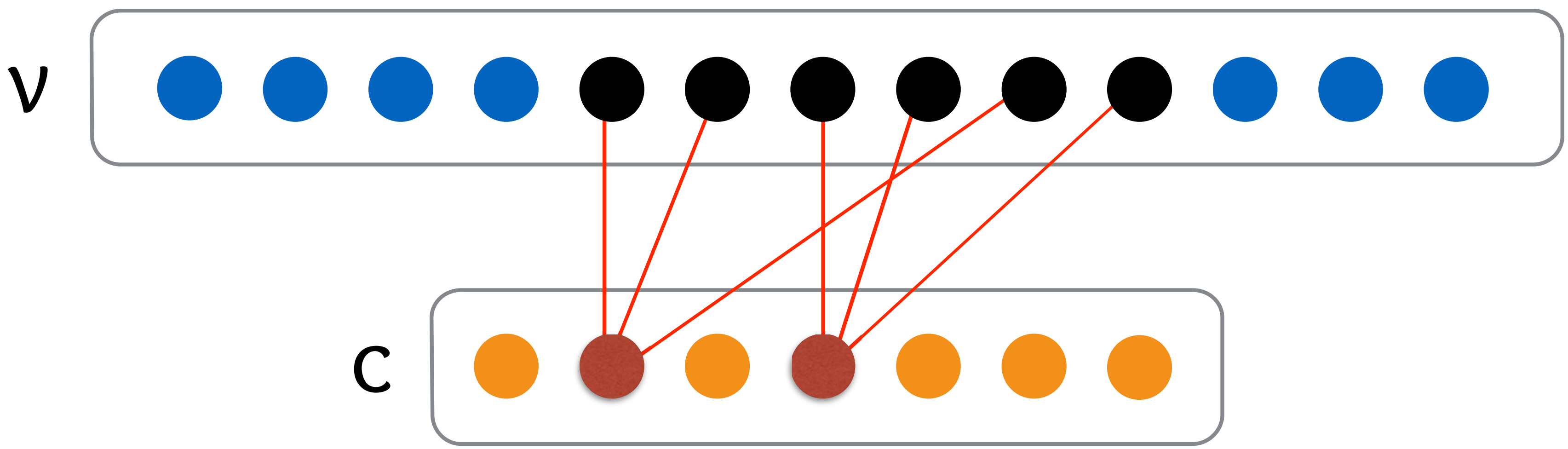


Max Sat

Input A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses? Question

Get rid of one vertex...

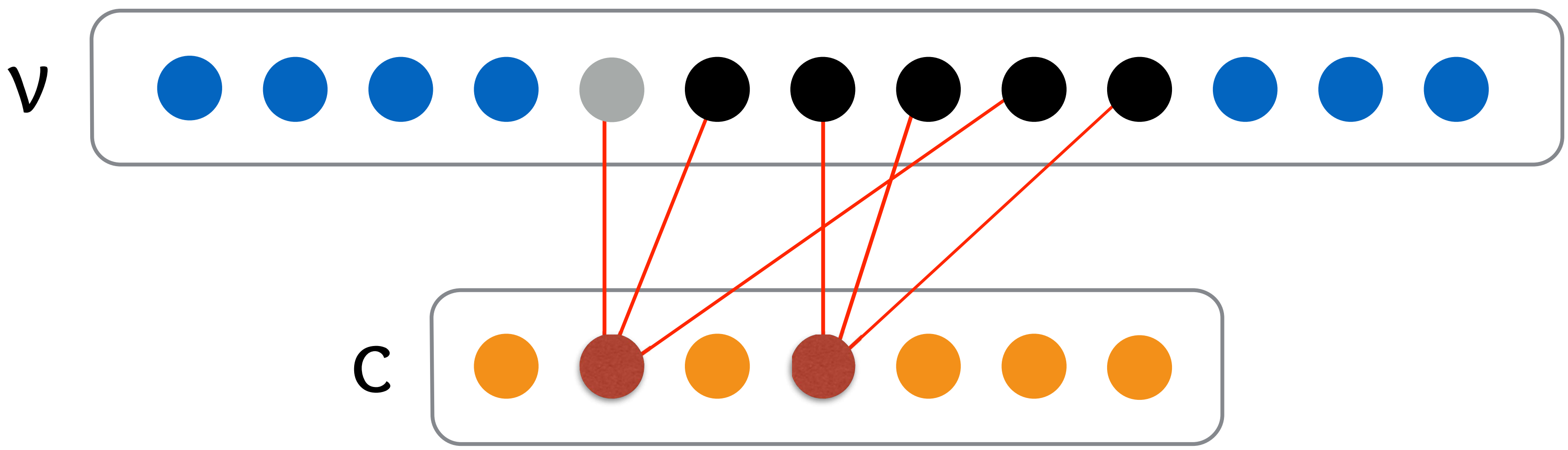


MAX SAT

Input A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses? Question

Get rid of one vertex...



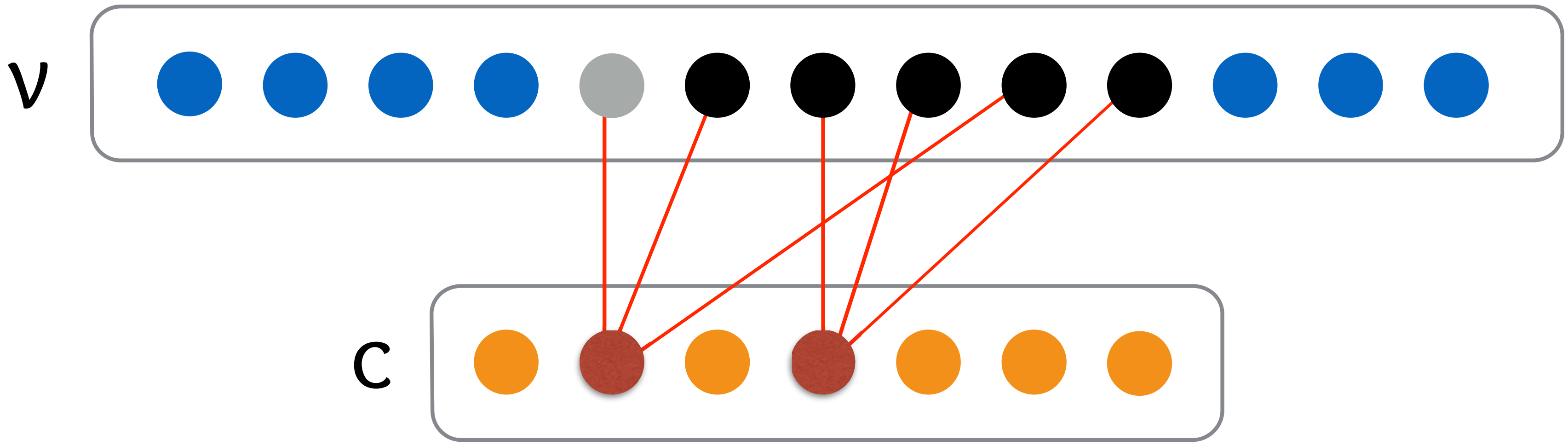
MAX SAT

Input A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses? Question

Get rid of one vertex...

The rest of it can be matched!



MAX SAT

Input

A CNF Formula over n variables with m clauses.

Is there an assignment satisfying at least k clauses?

Question

This implies a kernel with at most k variables and 2k clauses.

MAX SAT

Feedback Vertex Set

Problem Definition

FEEDBACK VERTEX SET **Parameter:** k
Input: An undirected graph G and a positive integer k.
Question: Does there exists a subset X of size at most k such that $\mathsf{G} - \mathsf{X}$ is acyclic?

X is called feedback-vertex set (fvs) of G.

Problem Definition

FEEDBACK VERTEX SET **Parameter:** k
Input: An undirected graph G and a positive integer k.
Question: Does there exists a subset X of size at most k such that $G - X$ is acyclic?

X is called feedback-vertex set (fvs) of G.
Goal is to obtain a polynomial kernel for FEEDBACK VERTEX SET.

What reduction rules we already know?

Reduction.FVS

If there is a loop at a vertex v*, delete* v *from the graph and decrease* k *by one.*

What reduction rules we already know?

Multiplicity of a multiple edge does not influence the set of feasible solutions to the instance (G, k).

Reduction.FVS

If there is an edge of multiplicity larger than 2, reduce its multiplicity to 2.

What reduction rules we already know?

Any vertex of degree at most 1 does not participate in any cycle in G, so it can be deleted.

Reduction.FVS

If there is a vertex v *of degree at most* 1*, delete* v*.*

What reduction rules we already know?

Concerning vertices of degree 2, observe that, instead of including into the solution any such vertex, we may as well include one of its neighbors.

Reduction.FVS

If there is a vertex v of degree 2, delete v and connect its two neighbors by a new edge.

What do we achieve after all these?

After exhaustively applying these four reduction rules, the resulting graph G

(P1) contains no loops,

(P2) has only single and double edges, and

(P3) has minimum vertex degree at least 3.

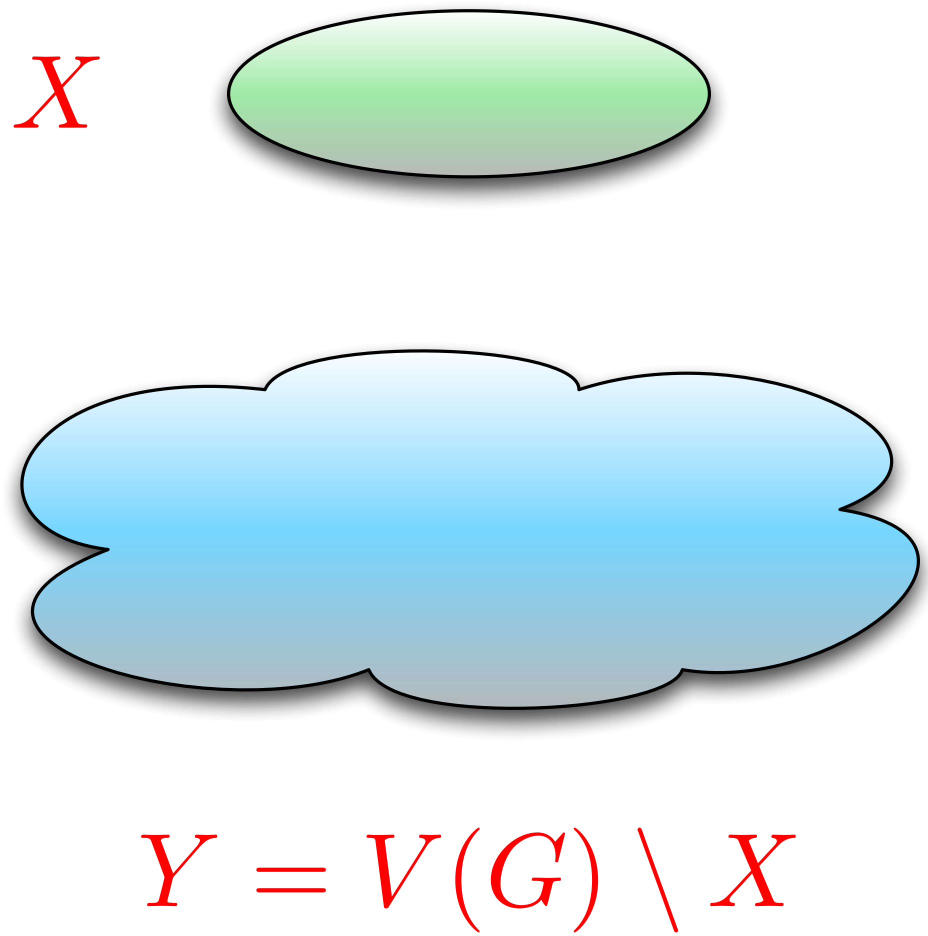
Stopping rule.

A rule that stops the algorithm if we already exceeded our budget.

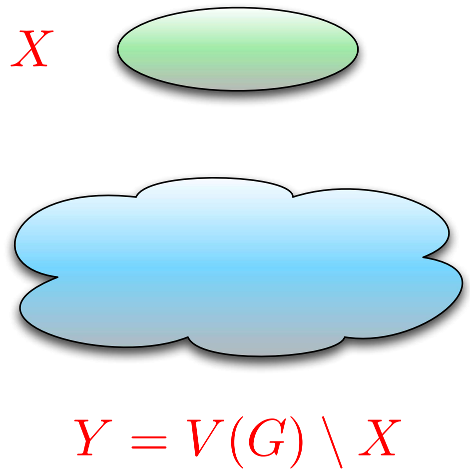
Reduction.FVS

If $k < 0$*, terminate the algorithm and conclude that* (G, k) *is a no-instance.*

A picture :)

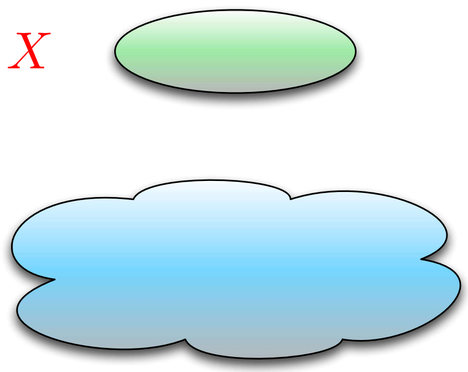


Maximum degree is $\mathbf{d}$.



Maximum degree is **d**.

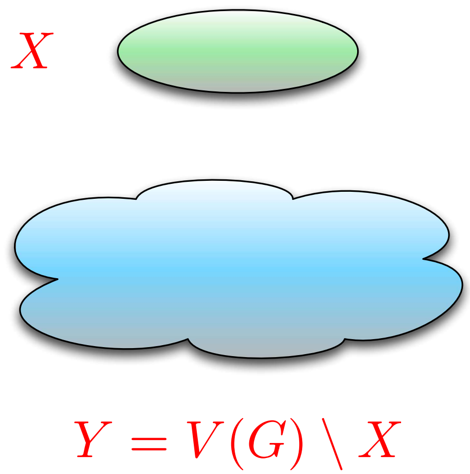
X



$$Y = V(G) \setminus X$$

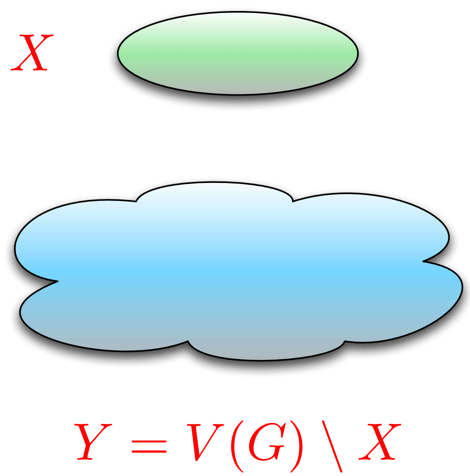
$$\sum_{v \in V(G)} \text{degree}(v) = 2|E(G)|$$

Maximum degree is **d**.



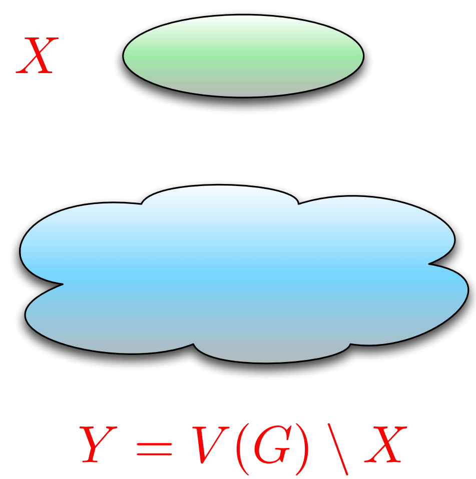
$$3|V(G)| \leqslant \sum_{v \in V(G)} \text{degree}(v) = 2|E(G)|$$

Maximum degree is **d**.



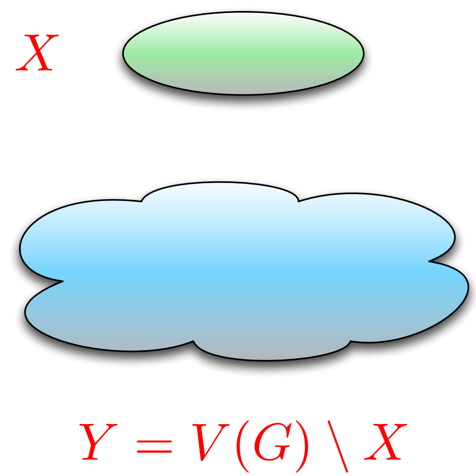
$$1.5|V(G)| \leqslant |E(G)|$$

Maximum degree is **d**.



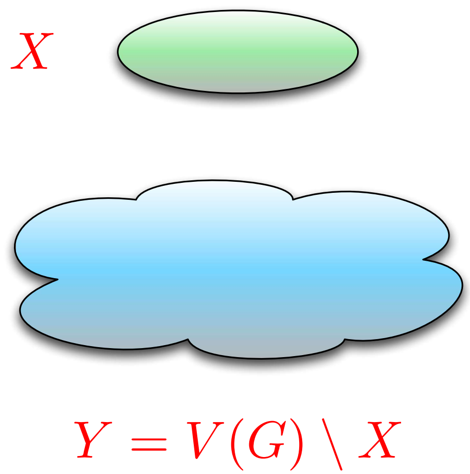
$$|E(G)| \leqslant d|X| + (|V(G)| - |X| - 1)$$

Maximum degree is $\mathbf{d}$.



$$1.5|V(G)| \leqslant |E(G)| \leqslant d|X| + (|V(G)| - |X|)$$

Maximum degree is **d**.



$$1.5|V(G)| \leqslant |E(G)| \leqslant d|X| + (|V(G)| - |X|)$$

$$\implies |V(G)| \leqslant 2(d-1)|X| \leqslant 2(d-1)k.$$

Summarizing:

Lemma

If a graph G *has minimum degree at least* 3*, maximum degree at most* d*, and feedback vertex set of size at most* k*, then it has less than* $2(d-1)k$ *vertices and less than* $2(d-1)dk$ *edges.*

Summarizing: (possible to prove)

Lemma

If a graph G *has minimum degree at least* 3*, maximum degree at most* d*, and feedback vertex set of size at most* k*, then it has less than* $(d+1)k$ *vertices and less than* $2dk$ *edges.*

A new rule

Reduction.FVS

If $|\mathsf{V}(\mathsf{G})| \geqslant (\mathsf{d}+1)\mathsf{k}$ *or* $|\mathsf{E}(\mathsf{G})| \geqslant 2\mathsf{d}\mathsf{k}$, *where* d *is the maximum degree of* G, *then terminate the algorithm and return that* (G, k) *is a no-instance.*

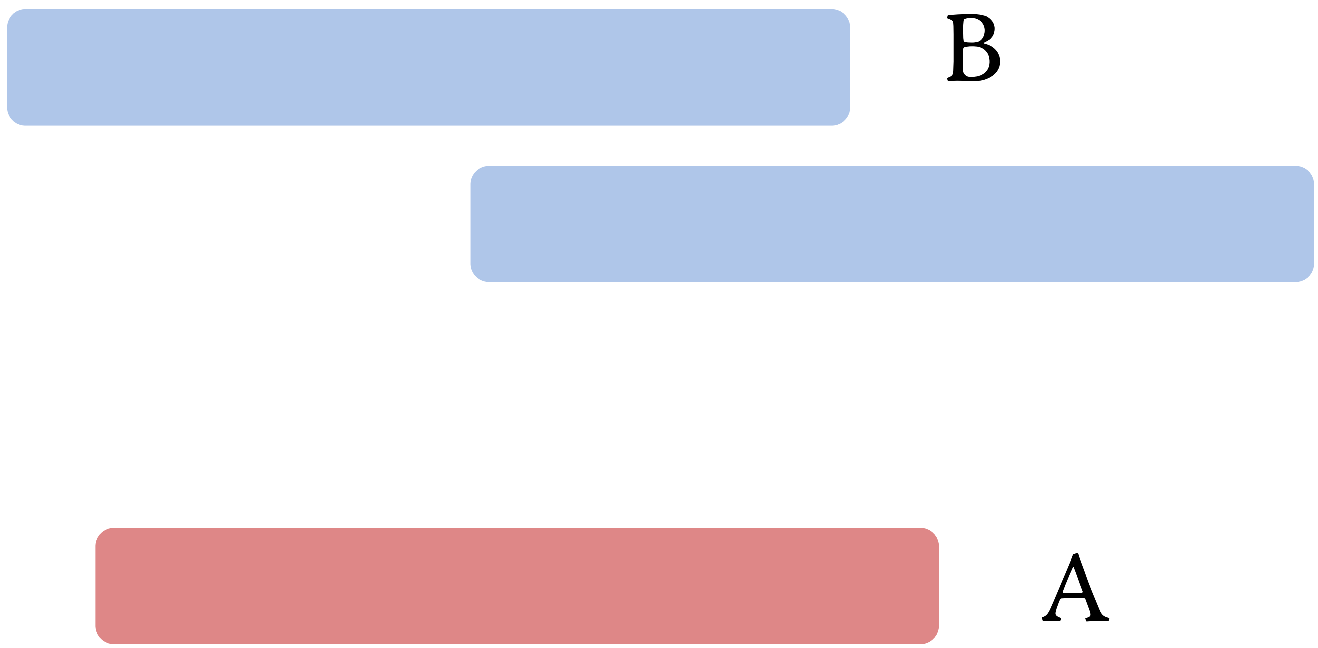
So what do we need to get the polynomial kernel?

So what do we need to get the polynomial kernel?

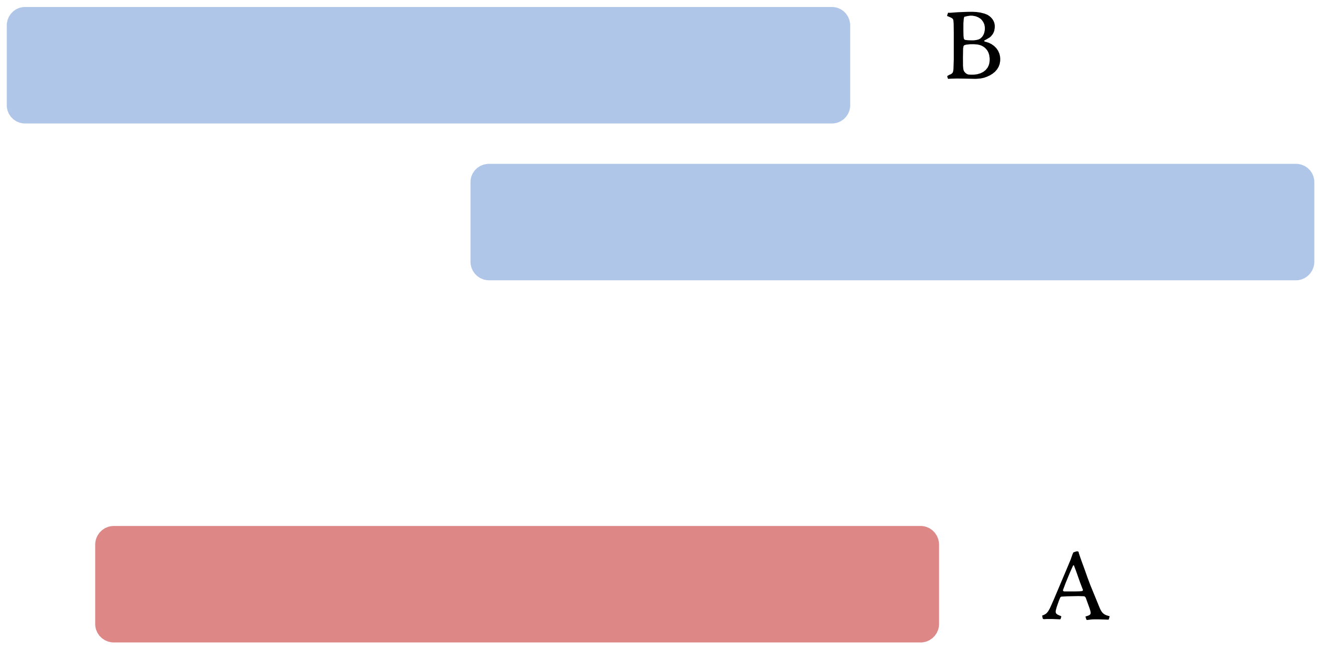
Bound the maximum degree of the graph by a polynomial in k.

Part 2: Recap

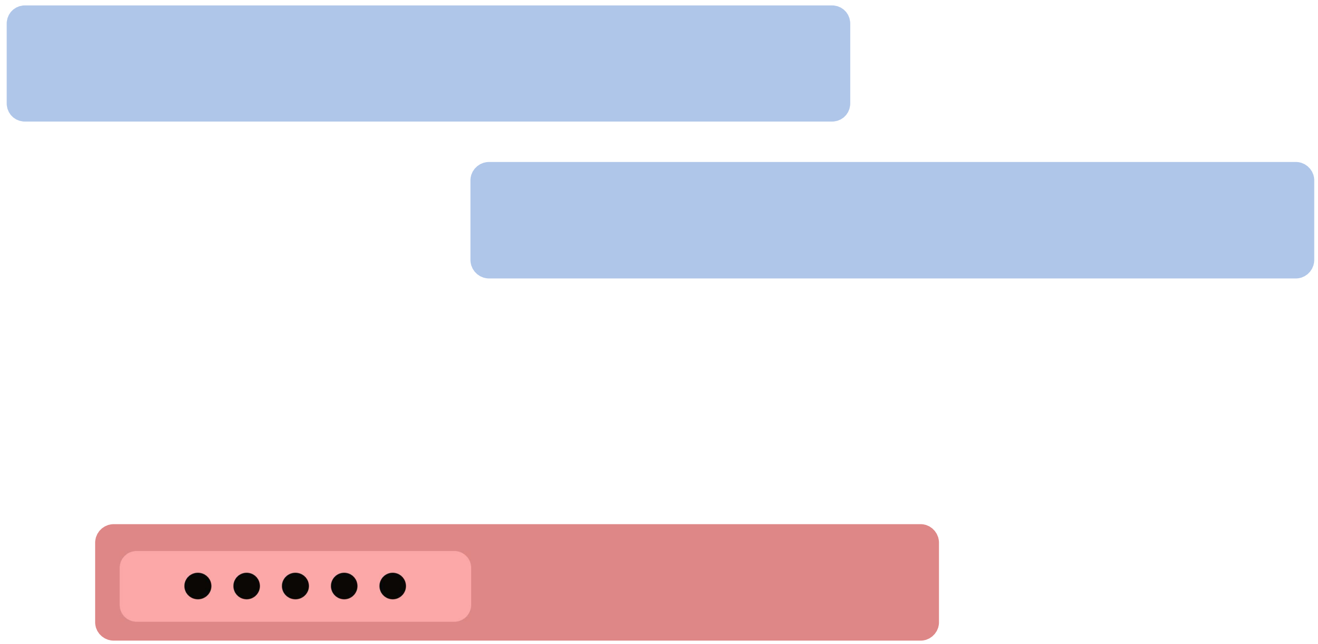
A Tale of **2** Matchings



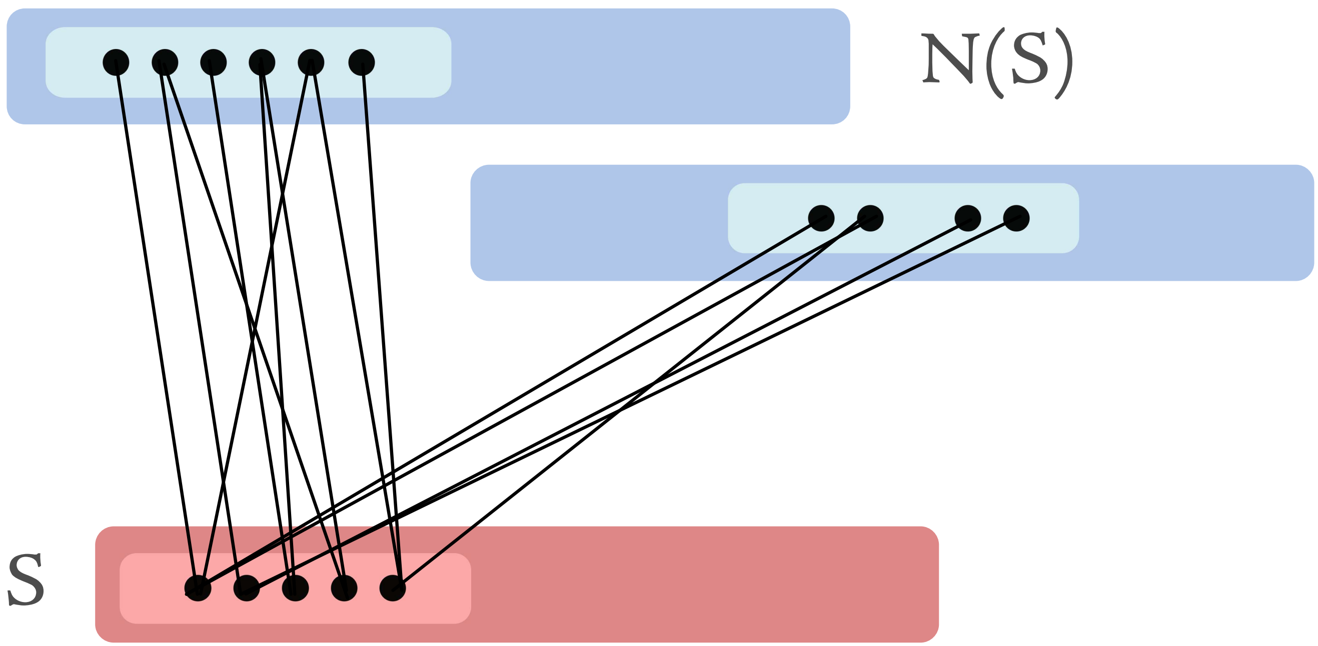
Consider a bipartite graph one of whose parts (say B) is at least twics as big as the other (call this A).



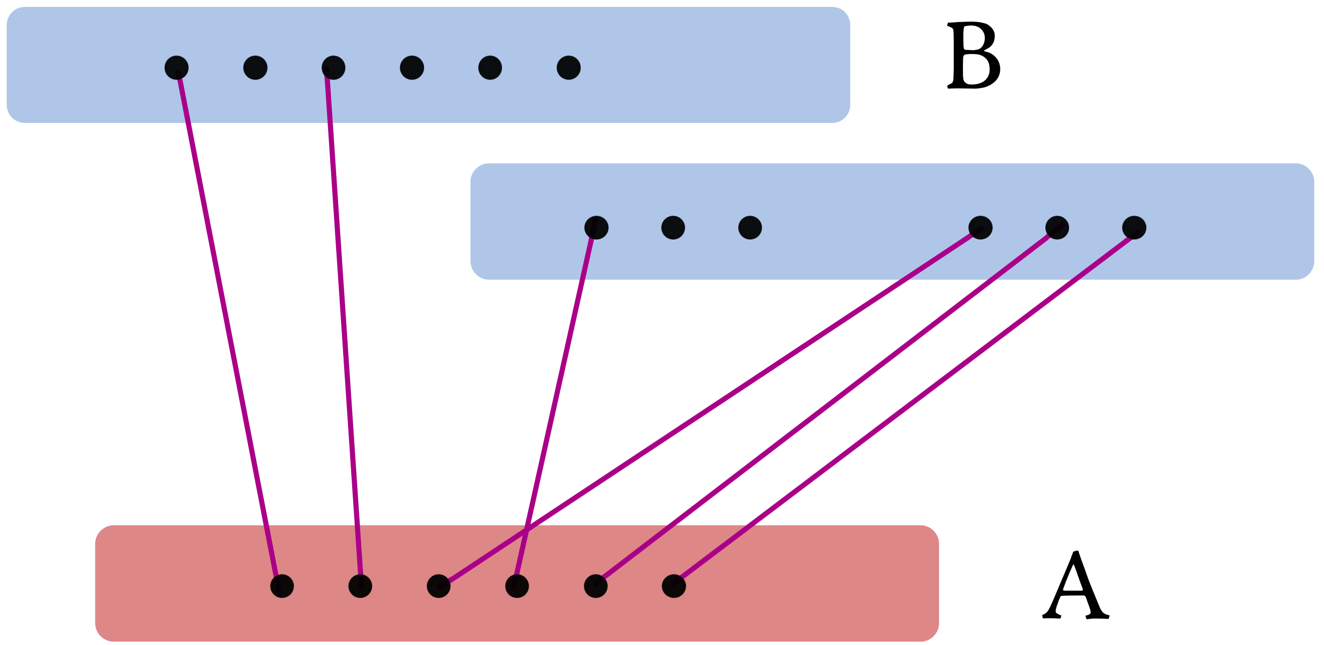
Assume that there are no isolated vertices in B.



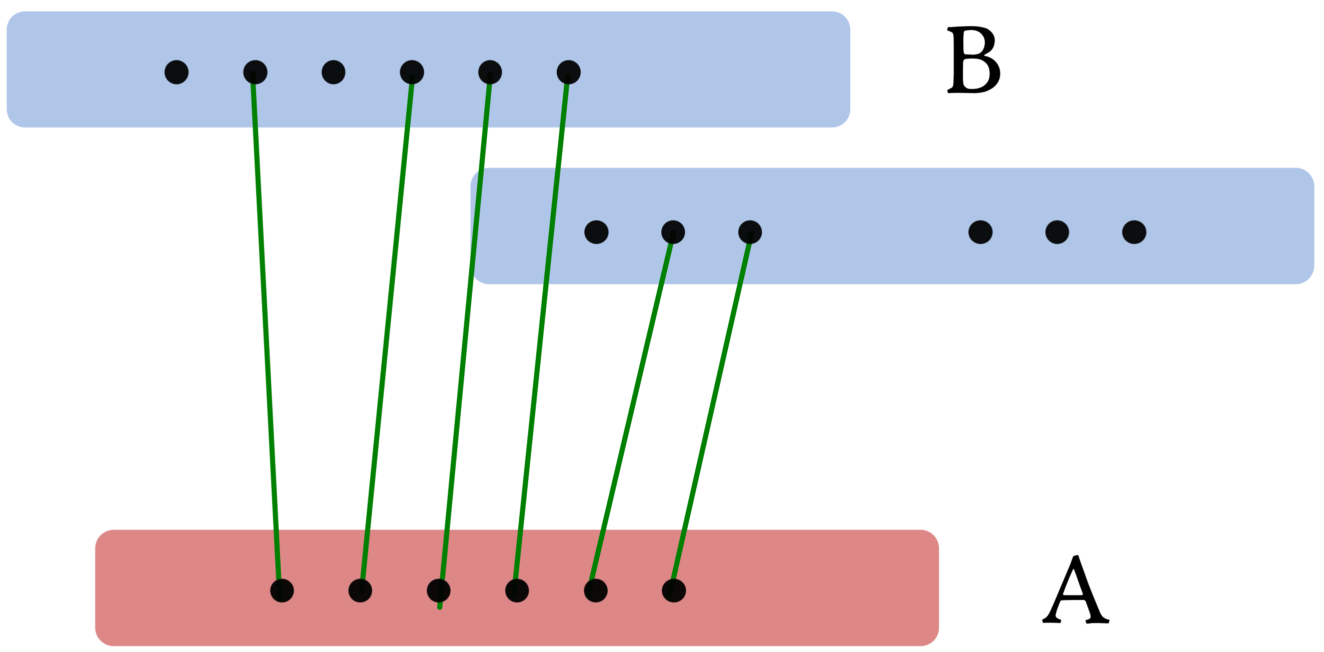
Suppose, further, that for every subset S in A,



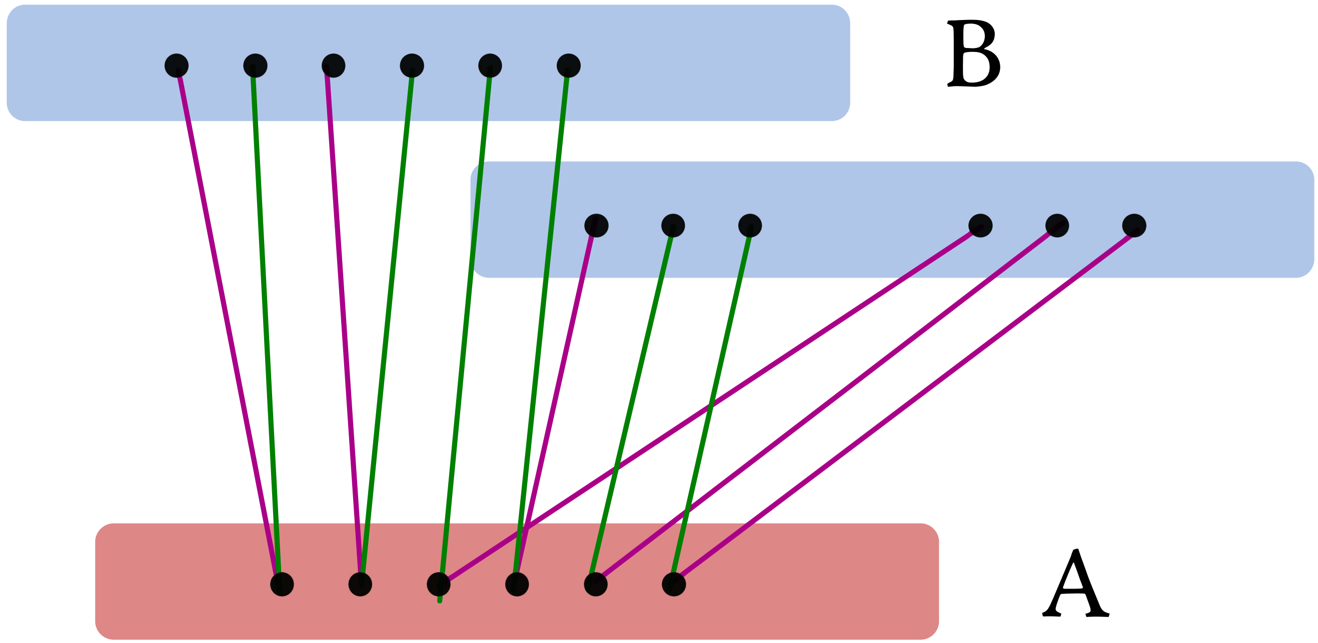
Suppose, further, that for every subset S in A, $\mathsf{N}(S)$ is at least twice as large as $|S|$.



Then there exist two matchings saturating A,



Then there exist two matchings saturating A,



Then there exist two matchings saturating A, and disjoint in B.

Claim:

If $|\mathrm{B}| \geqslant 2|A|$, then there exists a subset X of A such that:

there exists 2 matchings saturating the subset X that are vertex-disjoint in B.

Claim:

If $|\mathrm{B}| \geqslant 2|A|$, then there exists a subset X of A such that:

there exists 2 matchings saturating the subset X that are vertex-disjoint in B.

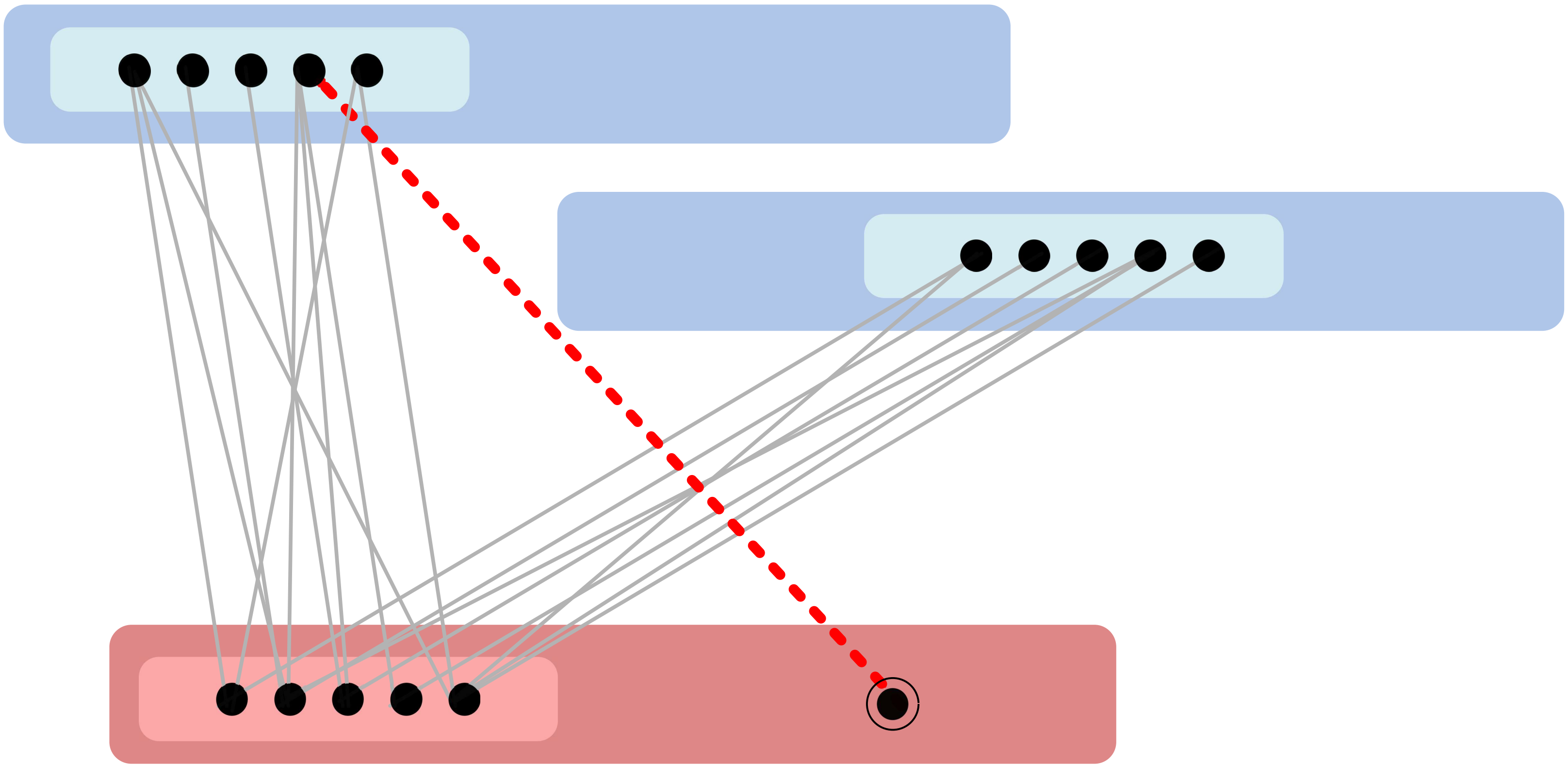
Claim:

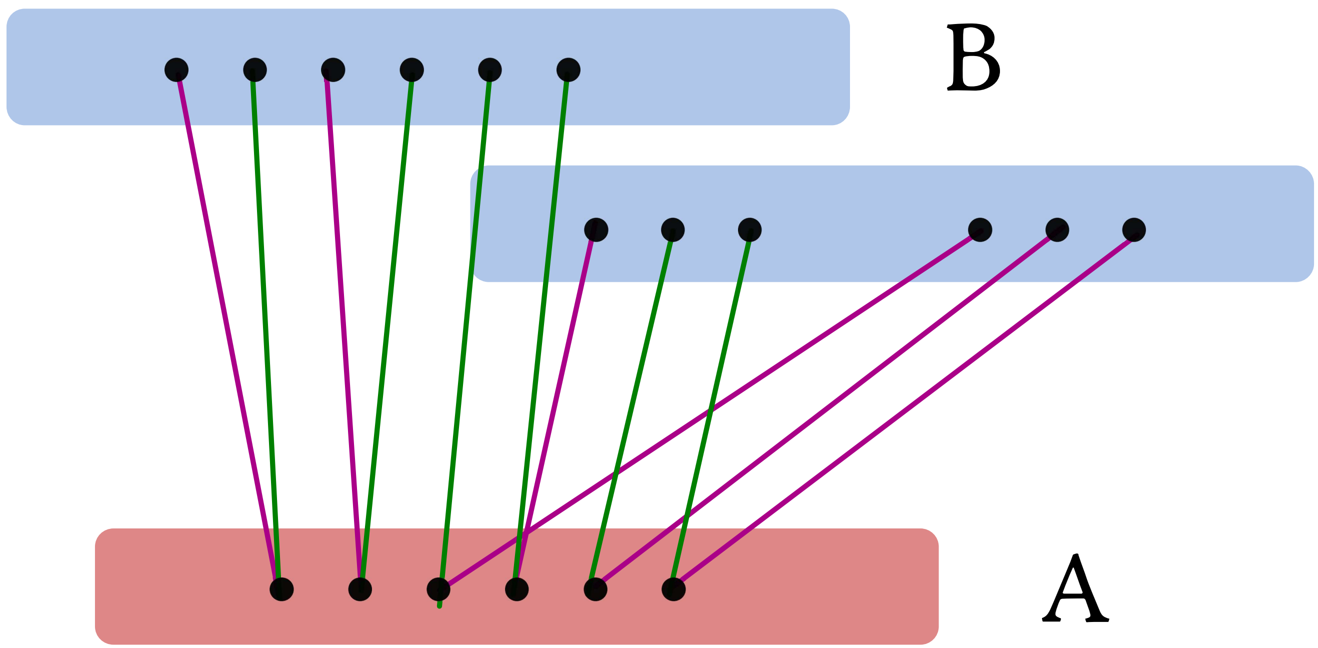
If $|B| \geqslant 2|A|$, then there exists a subset X of A such that:

there exists 2 matchings saturating the subset X that are vertex-disjoint in B,

provided B does not have any isolated vertices.

Crucially: it turns out that the endpoints of the matchings in B (the larger set) do not have neighbors outside X.




B
A

q-Expansion Lemma

Let $q \geqslant 1$ be a positive integer and G be a bipartite graph with vertex bipartition (A, B) such that

(i) $|B| \geqslant q|A|$, and
(ii) there are no isolated vertices in B.

Then there exist nonempty vertex sets $X \subseteq A$ and $Y \subseteq B$ such that

- there is a q-expansion of X into Y, and
- no vertex in Y has a neighbor outside X, that is, $N(Y) \subseteq X$.

Furthermore, the sets X and Y can be found in time polynomial in the size of G.

q-Expansion Lemma

Let $q \geqslant 1$ be a positive integer and G be a bipartite graph with vertex bipartition (A, B) such that

(i) $|B| \geqslant q|A|$, and
(ii) there are no isolated vertices in B.

Then there exist nonempty vertex sets $X \subseteq A$ and $Y \subseteq B$ such that

- there is a q-expansion of X into Y, and
- no vertex in Y has a neighbor outside X, that is, $N(Y) \subseteq X$.

Furthermore, the sets X and Y can be found in time polynomial in the size of G.

We will use this lemma with $q = 2$.

Part 3

2-Expansions and FVS

- For VERTEX COVER – if a vertex has degree $k+1$ then we must have it in the solution.

- For Vertex Cover – if a vertex has degree $k + 1$ then we must have it in the solution.

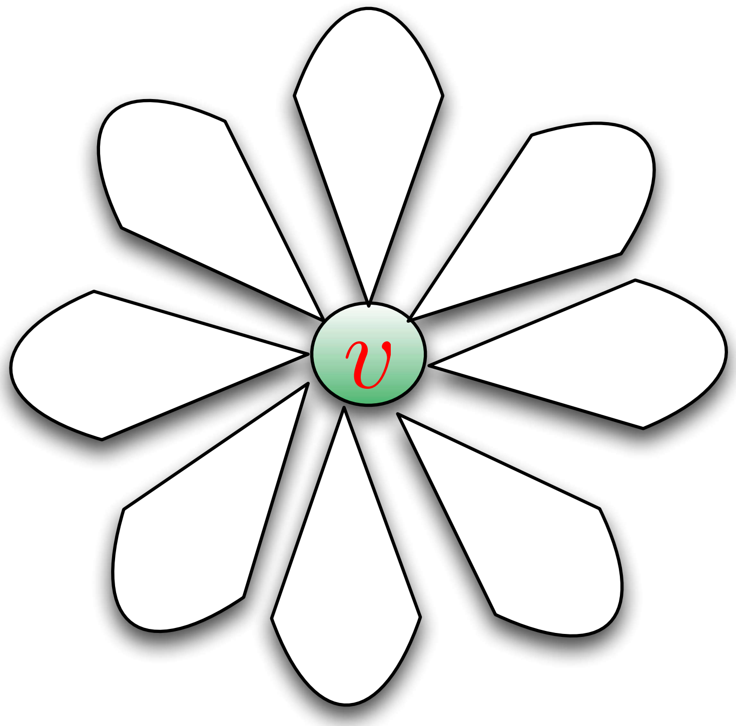
What would be the analogous rule for Feedback vertex Set.

- For VERTEX COVER – if a vertex has degree $k+1$ then we must have it in the solution.

What would be the analogous rule for FEEDBACK VERTEX SET.

For VERTEX COVER – wanted to hit edges and for FEEDBACK VERTEX SET – want to hit cycles..

Flower



$k+1-$ vertex disjoint cycles passing through it

Flower Rule.

Reduction.FVS

If there is a $k+1$-*flower passing through a vertex* v *then* $(G \setminus \{v\}, k-1)$.

Given a high-degree vertex v, finding a small feedback vertex set that does not contain v.

Given a high-degree vertex v, finding a small feedback vertex set that does not contain v.

A subset whose removal makes the graph acyclic.

Given a high-degree vertex v, finding a small feedback vertex set that does not contain v.

A polynomial function of k.

Given a high-degree vertex v, finding a small feedback vertex set that does not contain v.

Find an approximate feedback vertex set T.

Given a high-degree vertex v, finding a small feedback vertex set that does not contain v.

If T does not contain v, we are done.

Given a high-degree vertex v, finding a small feedback vertex set that does not contain v.

Else: $v \in \mathsf{T}$. Delete $\mathsf{T} \setminus v$ from G.

Given a high-degree vertex v, finding a small feedback vertex set that does not contain v.

The only remaining cycles pass through v.

Given a high-degree vertex v, finding a small feedback vertex set that does not contain v.

Find an optimal cut set for paths from $\mathsf{N}(v)$ to $\mathsf{N}(v)$.

Given a high-degree vertex v, finding a small feedback vertex set that does not contain v.

When is this cut set small enough?

Given a high-degree vertex v, finding a small feedback vertex set that does not contain v.

When is this cut set small enough?

When the largest collection of vertex disjoint paths from $\mathsf{N}(v)$ to $\mathsf{N}(v)$ is small.

Given a high-degree vertex v, finding a small feedback vertex set that does not contain v.

When is this cut set small enough?

When the largest collection of vertex disjoint paths from $\mathsf{N}(v)$ to $\mathsf{N}(v)$ is *not* small...

Given a high-degree vertex v, finding a small feedback vertex set that does not contain v.

When is this cut set small enough?

When the largest collection of vertex disjoint paths from $\mathsf{N}(v)$ to $\mathsf{N}(v)$ is *not* small... we get a reduction rule.

Given a high-degree vertex v, finding a small feedback vertex set that does not contain v.

When is this cut set small enough?

More than k vertex-disjoint paths from $N(v)$ to $N(v)$
$\rightarrow$ v belongs to *any* feedback vertex set
($k+1$-flower) of size at most k.

Given a high-degree vertex v, finding a small feedback vertex set that does not contain v.

So either v "forced", or we have feedback vertex set of suitable size.
Notice that we need to arrive at either situation in "polynomial time".

Approximate fvs

- There is a factor 2 approximation algorithm for Feedback Vertex Set. So use this to get T. If $|\mathsf{T}| > 2\mathsf{k}$ return no-instance. Else, we have the desired T.
-

Approximate fvs

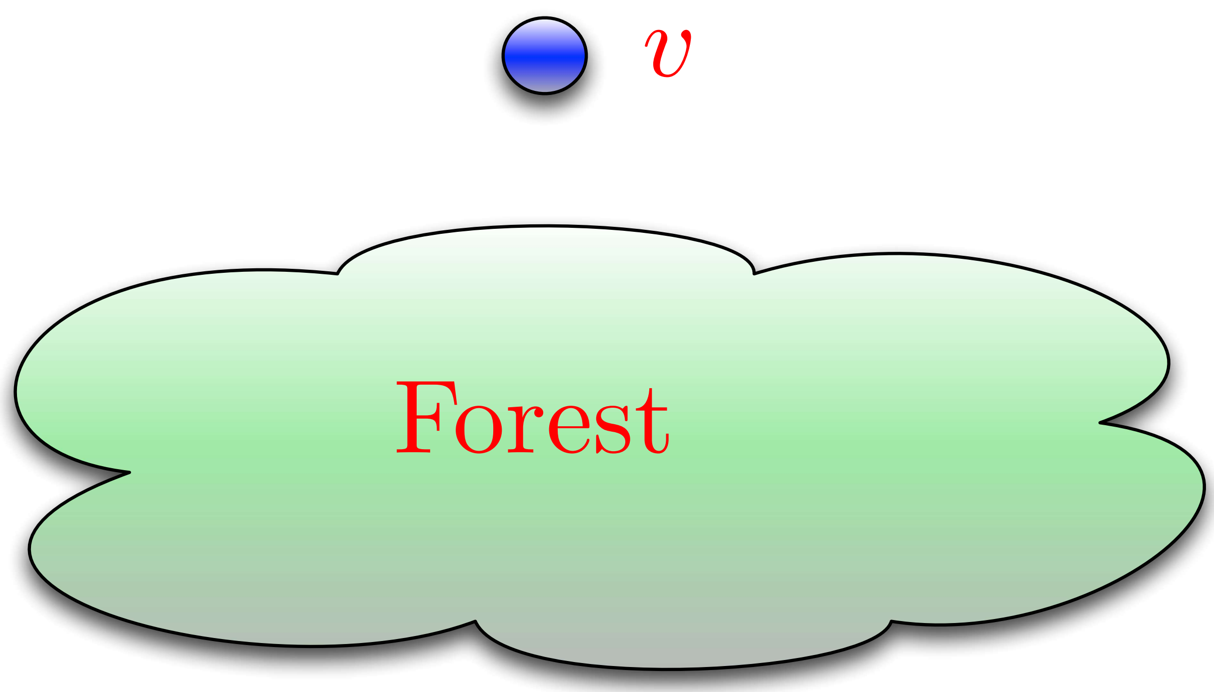
- There is a factor 2 approximation algorithm for FEEDBACK VERTEX SET. So use this to get T. If $|T| > 2k$ return no-instance. Else, we have the desired T.
- We have seen if G has minimum degree 3, then any fvs of size at most k contains one among the first $3k$ vertices of highest degree. Use this to get T of size $3k^2$ or return no-instance.

fvs without v when $v \in \mathsf{T}$.

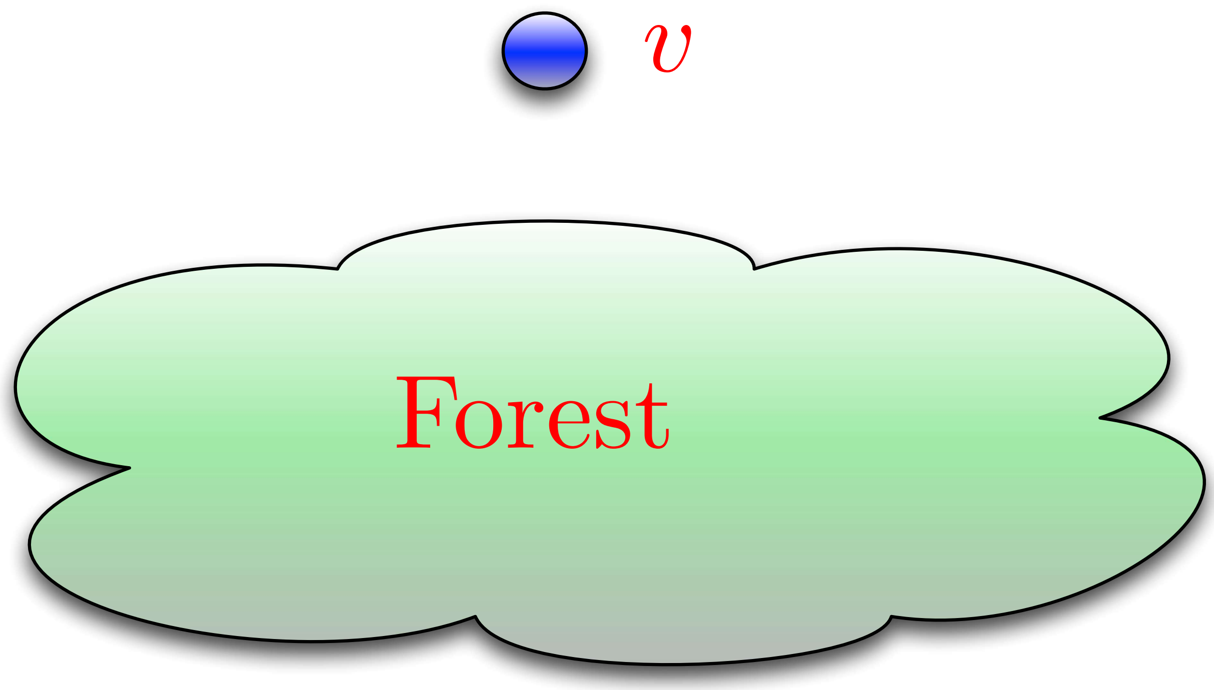
- $\mathsf{Z}_v = \mathsf{T} \setminus \{v\} + W$(something more).

fvs without v when $v \in \mathsf{T}$.

- $\mathsf{Z}_v = \mathsf{T} \setminus \{v\} + W$(something more).



fvs without v when $v \in T$.



W will be a fvs for Forest $+\, v$.

- Check whether there is a $k + 1$-flower containing v in Forest $+ v$ (if yes then we have reduction rule). (How to find?)
-
-

- Check whether there is a $k+1$-flower containing v in Forest $+ v$ (if yes then we have reduction rule). (How to find?)
- Else, we can show that there is fvs for Forest $+ v$ of size at most $2k$.
-

Book – Gallai Theorem

Theorem (Gallai)

Given a simple graph G*, a set* $\mathsf{T} \subseteq \mathsf{V}(\mathsf{G})$ *and an integer* s*, one can in polynomial time find either*

1. *a family of* $\mathsf{s}+1$ *pairwise vertex-disjoint* T*-paths, or*
2. *a set* B *of at most* $2\mathsf{s}$ *vertices, such that in* $\mathsf{G} \setminus \mathsf{B}$ *no connected component contains more than one vertex of* T*.*

What did we show.

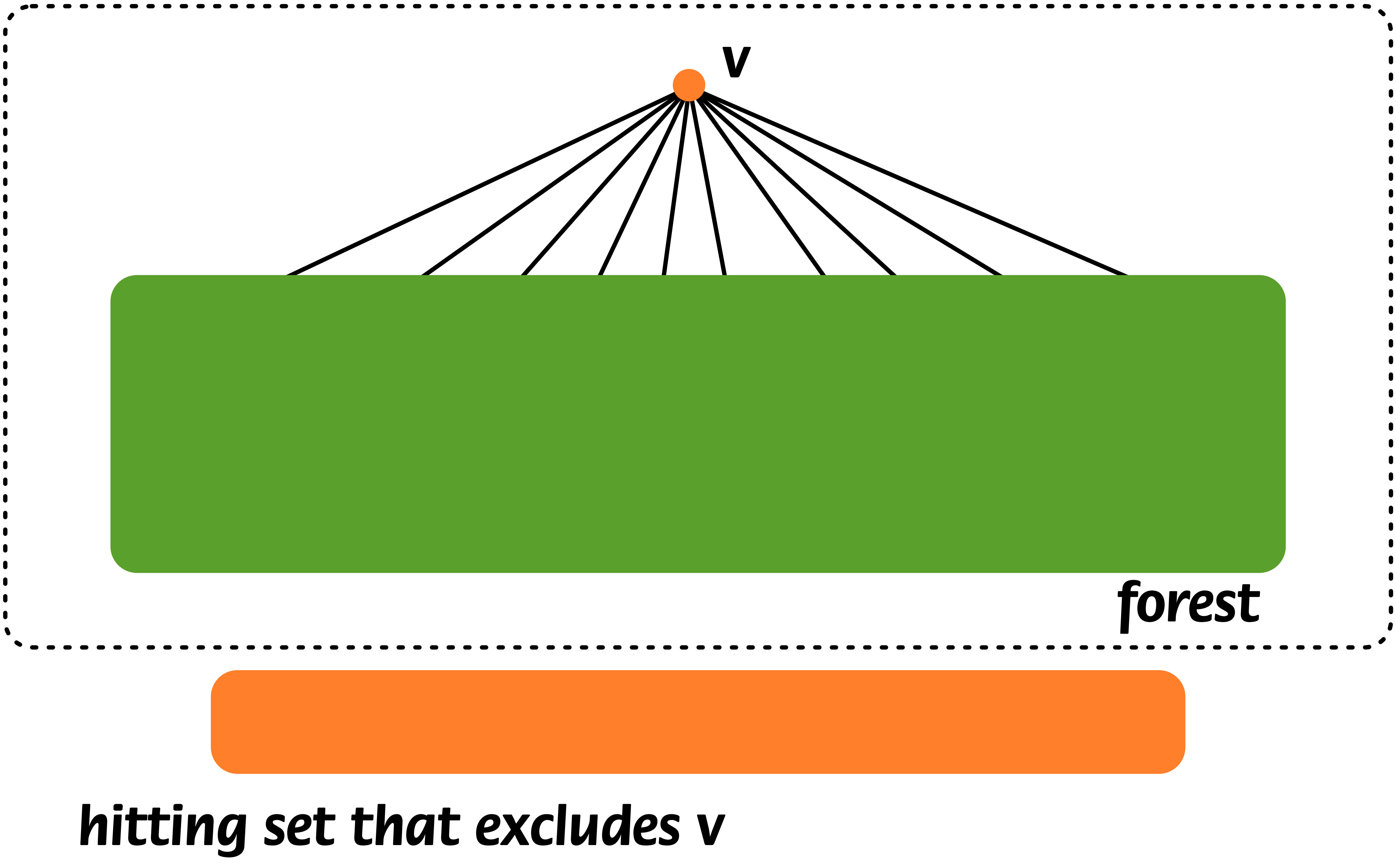
- For every vertex v either there is a $k+1$-flower passing through v or there is a Z_v of size at most $4k$ that does not include v and is a fvs of G.
-
-

What did we show.

- For every vertex v either there is a $k+1$-flower passing through v or there is a Z_v of size at most $4k$ that does not include v and is a fvs of G.
- In the first case we apply Flower Rule.
-

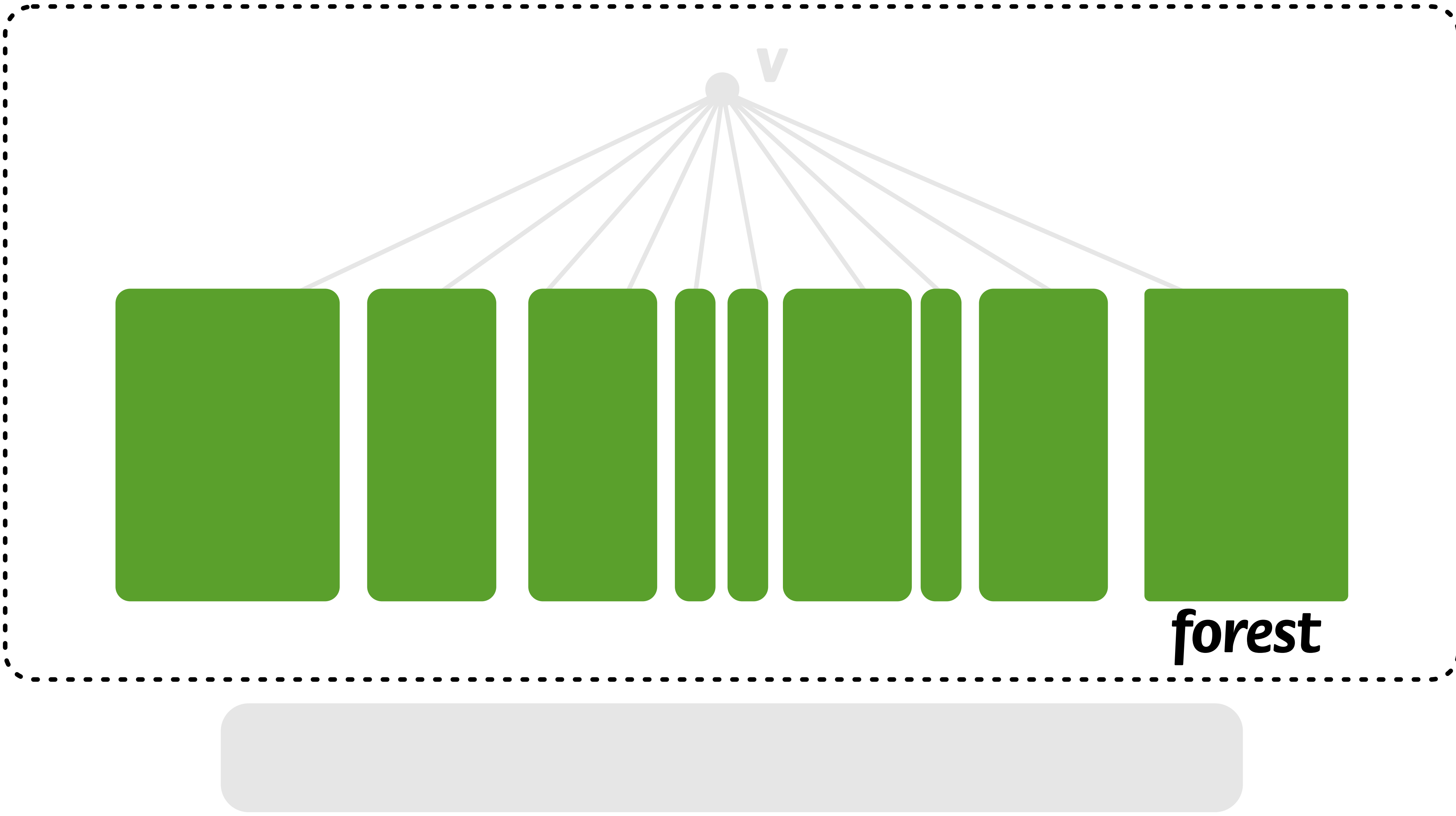
What did we show.

- For every vertex v either there is a $k+1$-flower passing through v or there is a Z_v of size at most $4k$ that does not include v and is a fvs of G.
- In the first case we apply Flower Rule.
- Assume that the first case does not happen, so we have Z_v of size at most $4k$ for every vertex $v \in V(G)$.


v
forest
hitting set that excludes v

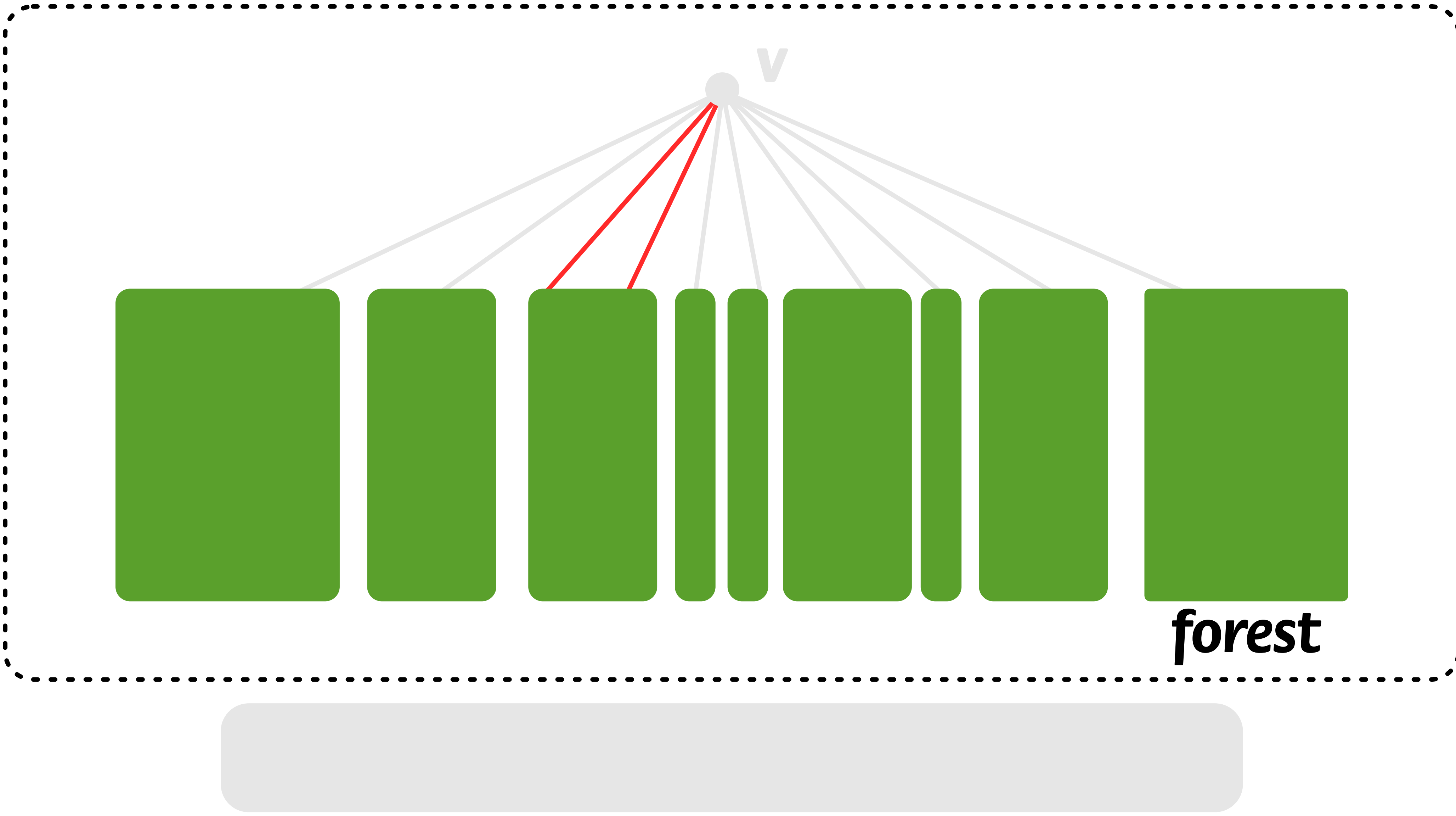
Focussing on the green Part

Consider the connected components of $V(G) \setminus (Z_v \cup \{v\})$.

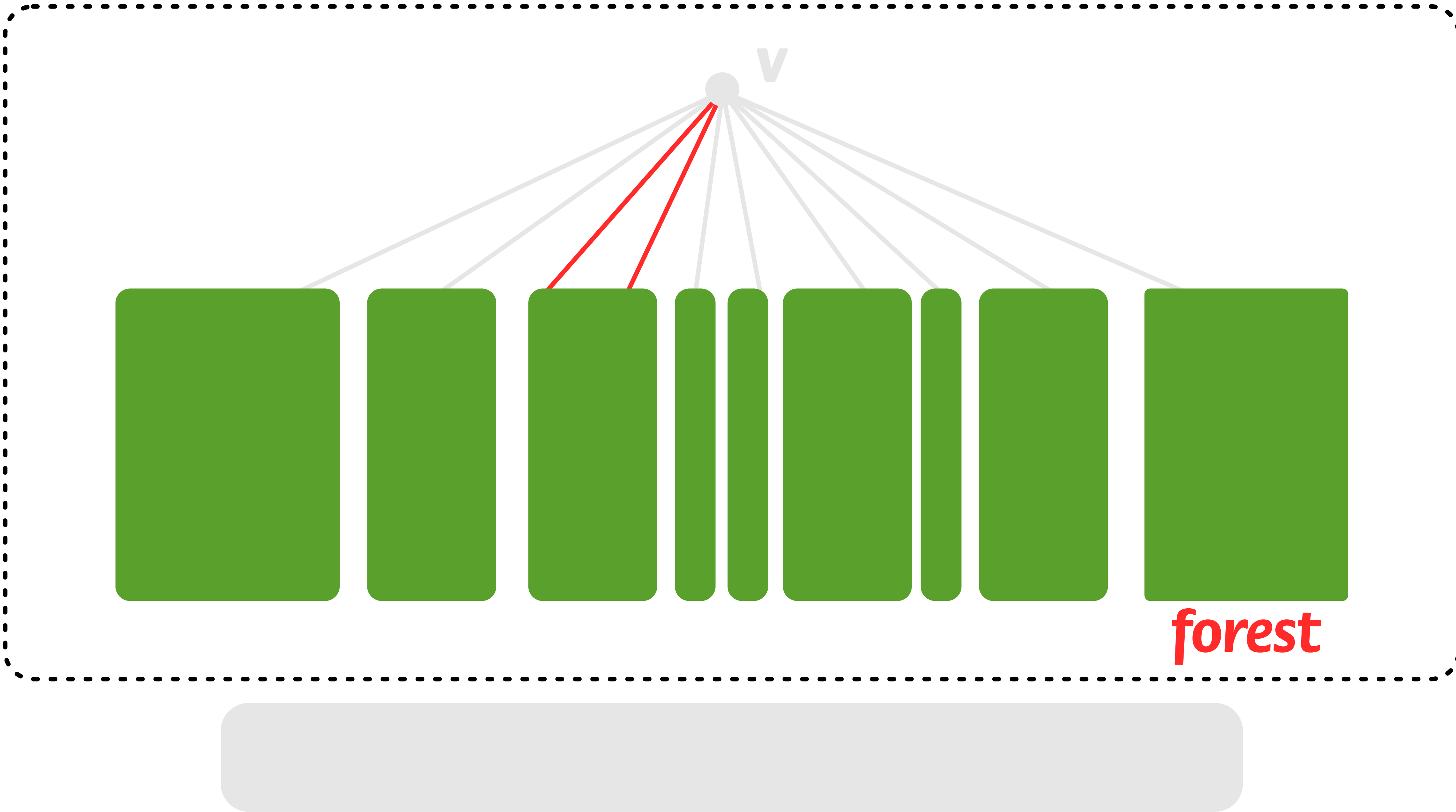


hitting set that excludes v

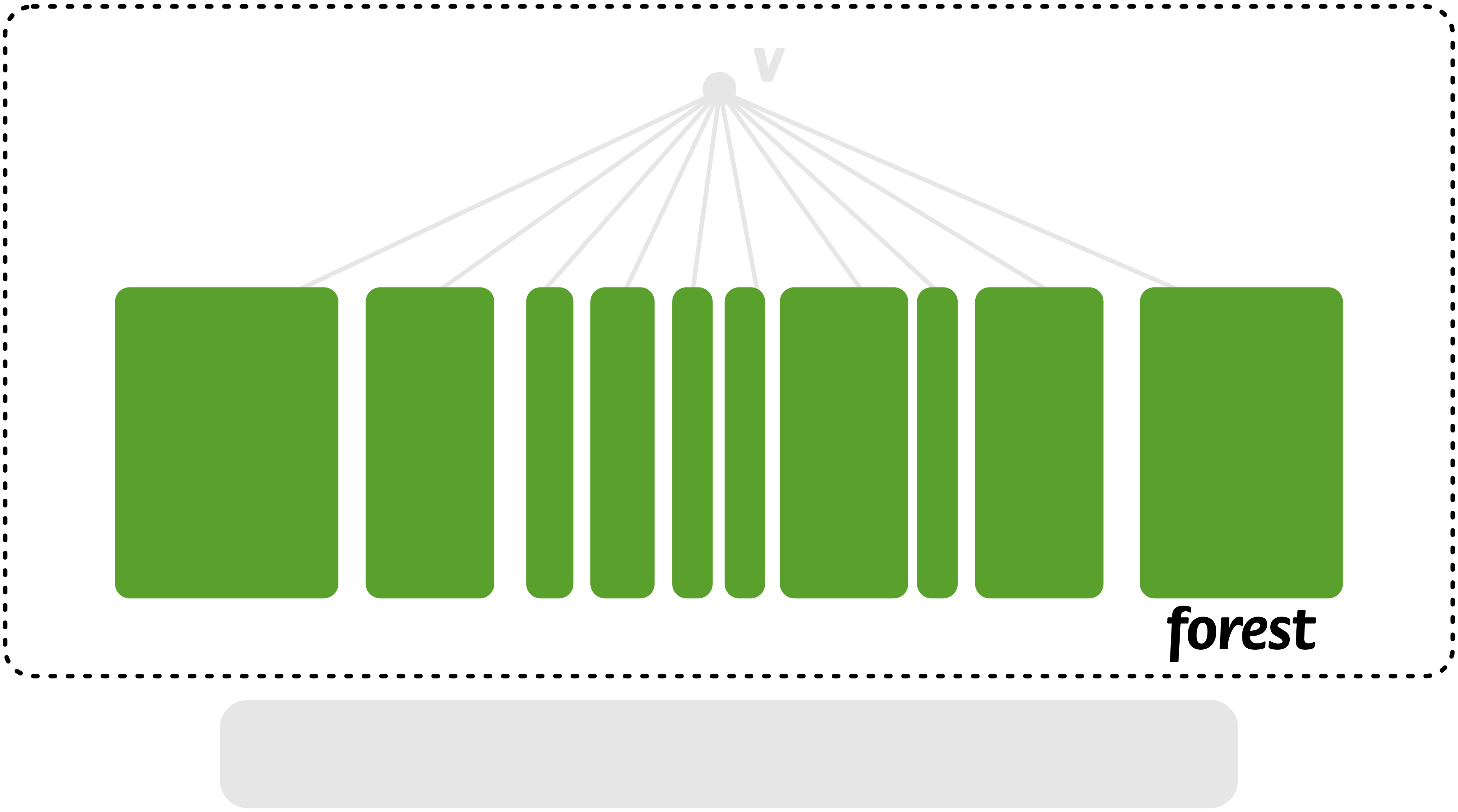
Could v have two neighbor in a connected components of $V(G) \setminus (Z_v \cup \{v\})$?



hitting set that excludes v

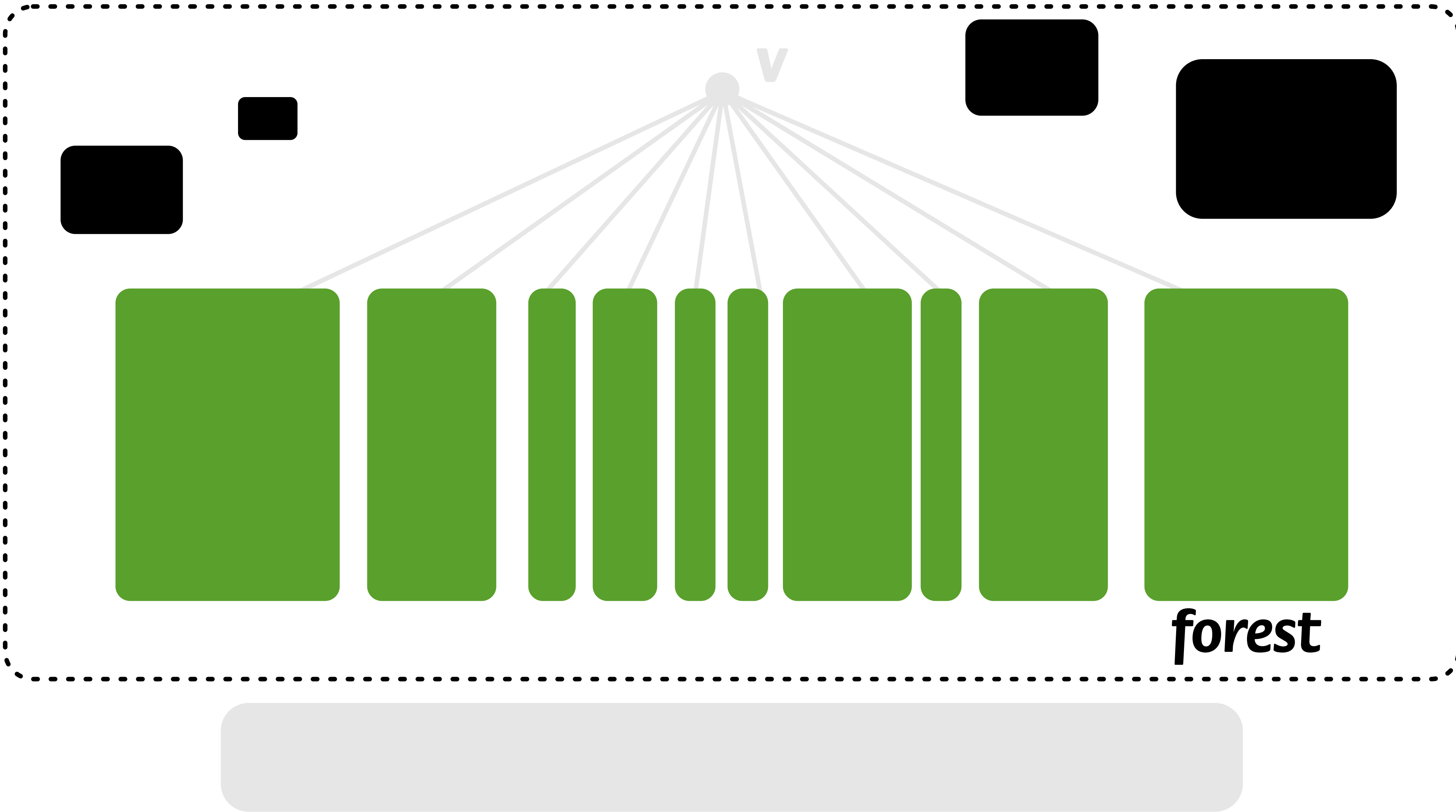


hitting set that excludes v



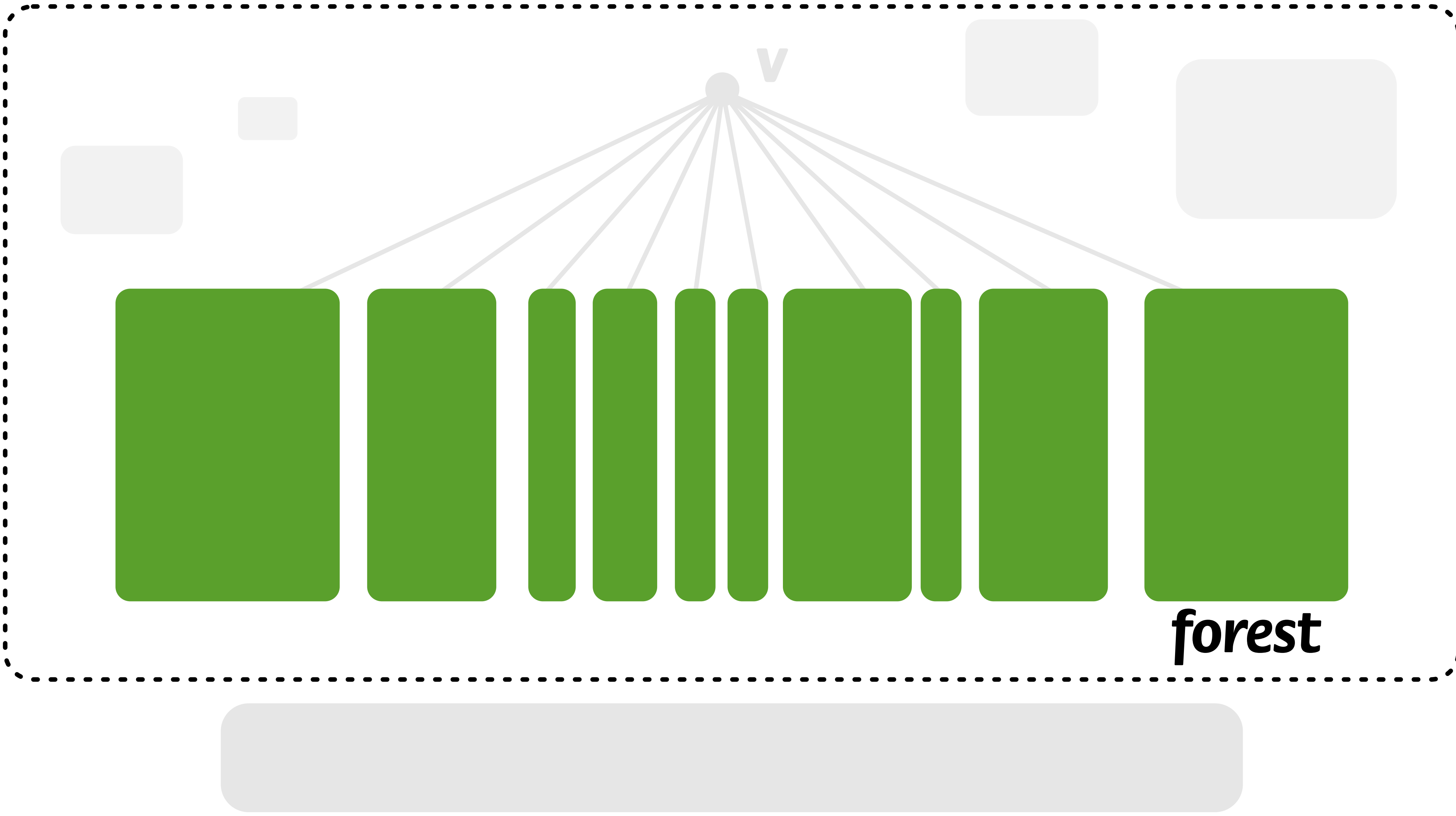
hitting set that excludes v

There could be components in $V(G) \setminus (Z_v \cup \{v\})$ that do not see any neighbor of v. Important, for us is that any component contains at most one neighbor of v and we will focus on them.



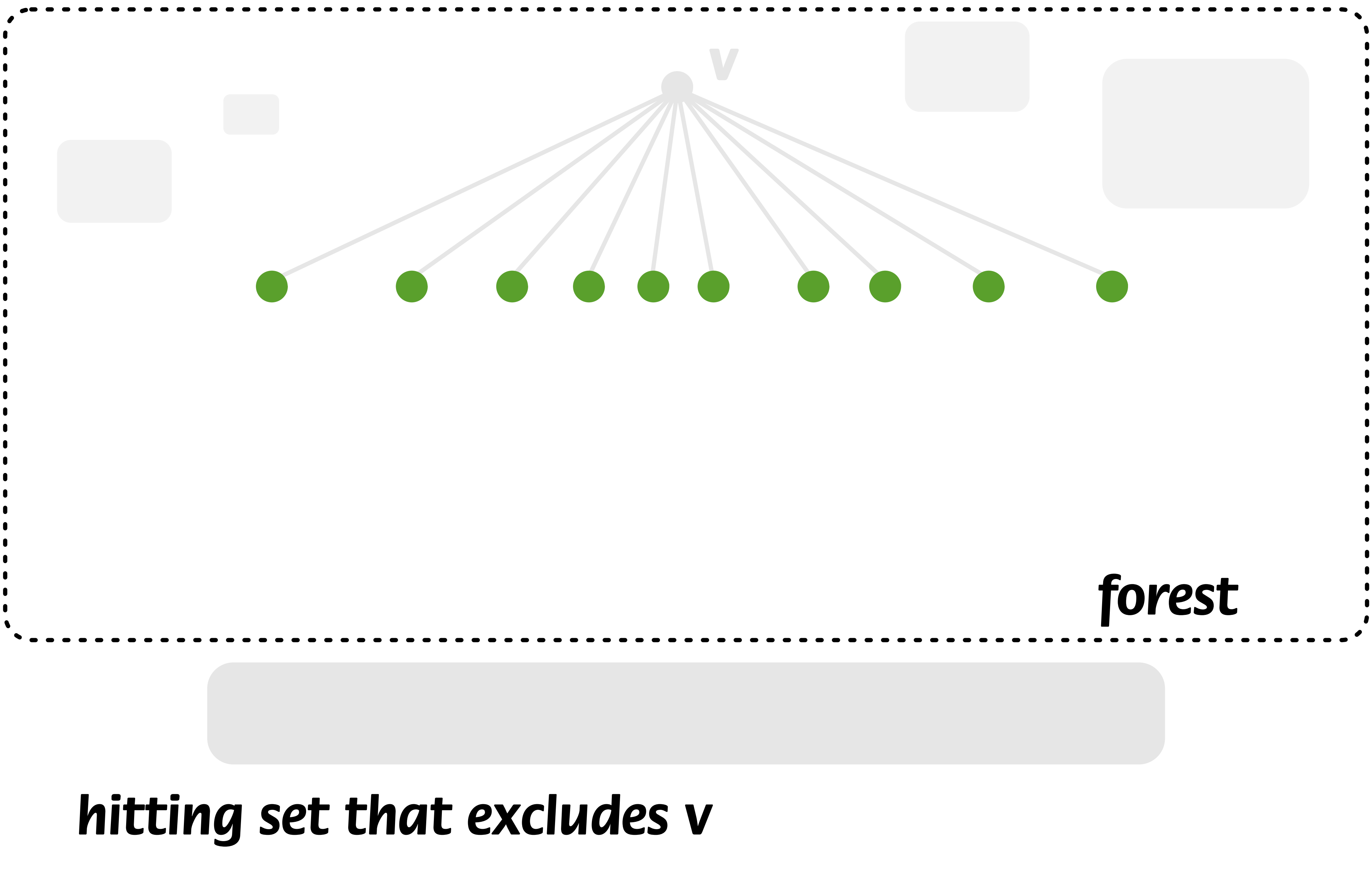
hitting set that excludes v

To bound the degree of v or to delete an edge incident to v we only focus on those components that contain some (exactly one) neighbor of v.

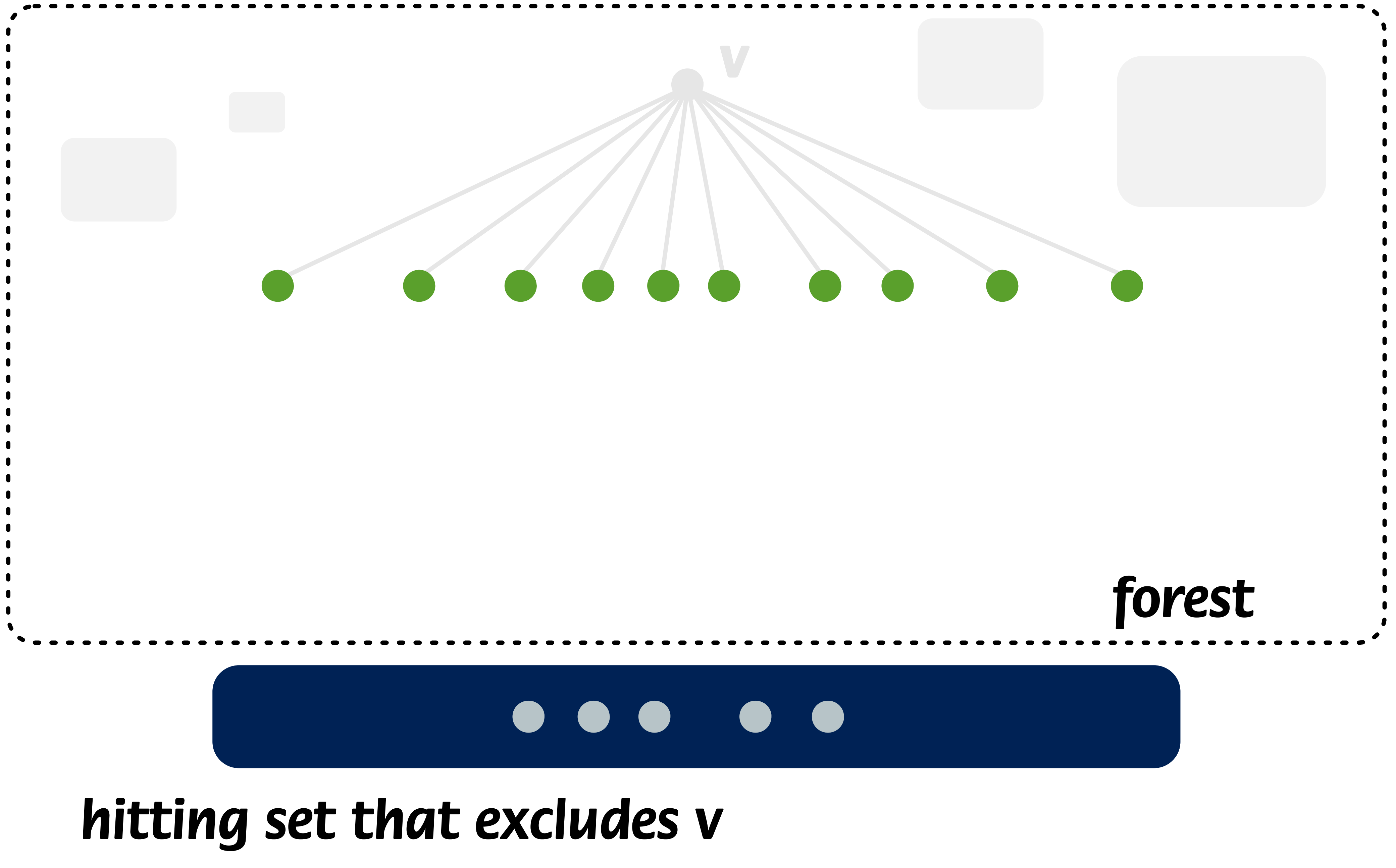


hitting set that excludes v

To apply 2-expansion lemma we need a bipartite graph. In one part (say B) we will have a vertex for every component in $V(G) \setminus (Z_v \cup \{v\})$ that contains a neighbor of v.


v
forest
hitting set that excludes v

To apply 2-expansion lemma we need a bipartite graph. In one part (say B) we will have a vertex for every component in $V(G) \setminus (Z_v \cup \{v\})$ that contains a neighbor of v. The other part A will be Z_v.


v
forest
hitting set that excludes v

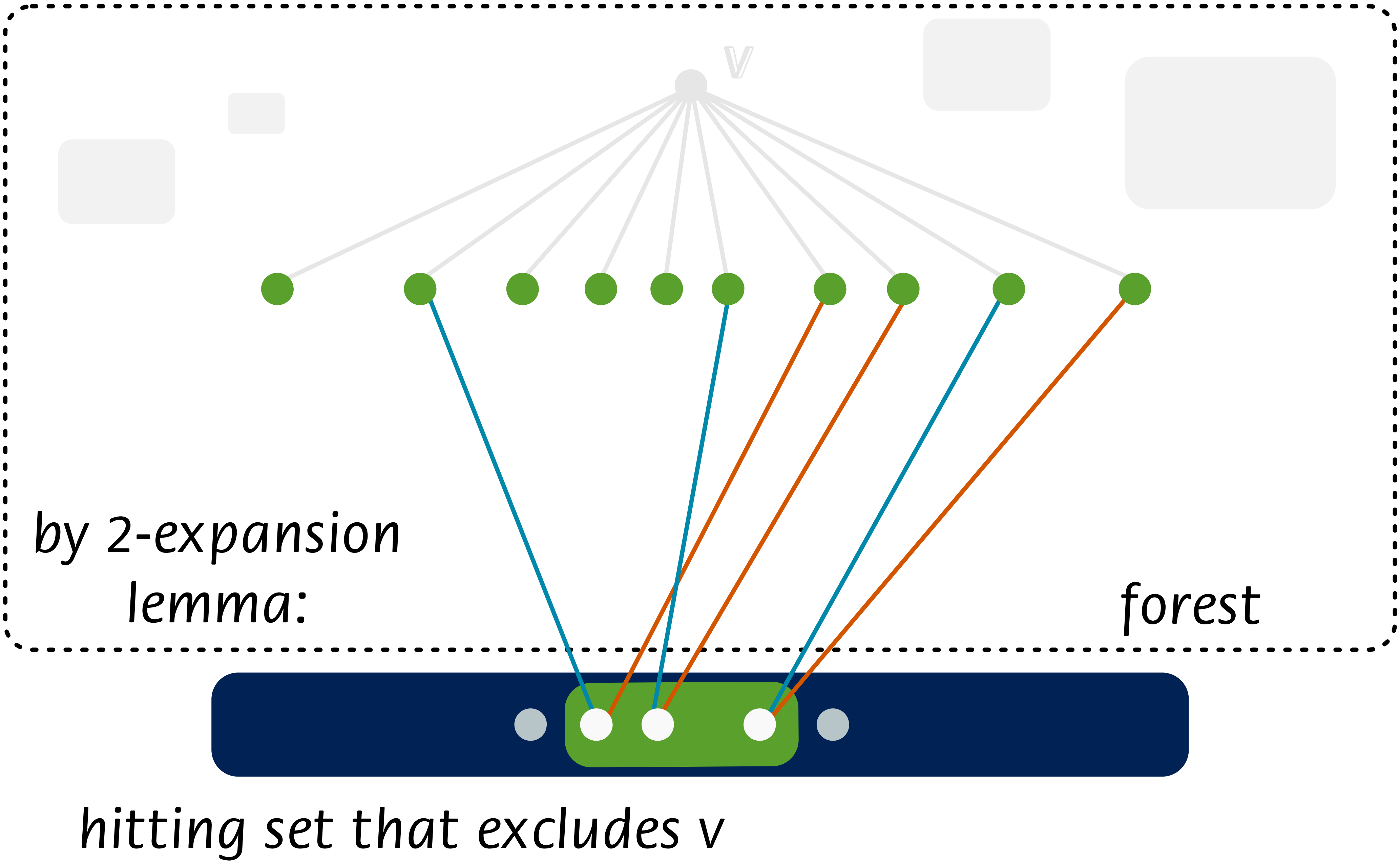
- So we have A and B. We put an edge between a vertex x in A and a vertex w in B, if x is adjacent to some vertex in the component represented by the vertex w. Essentially, we have obtained this bipartite graph by contracting the components.
-

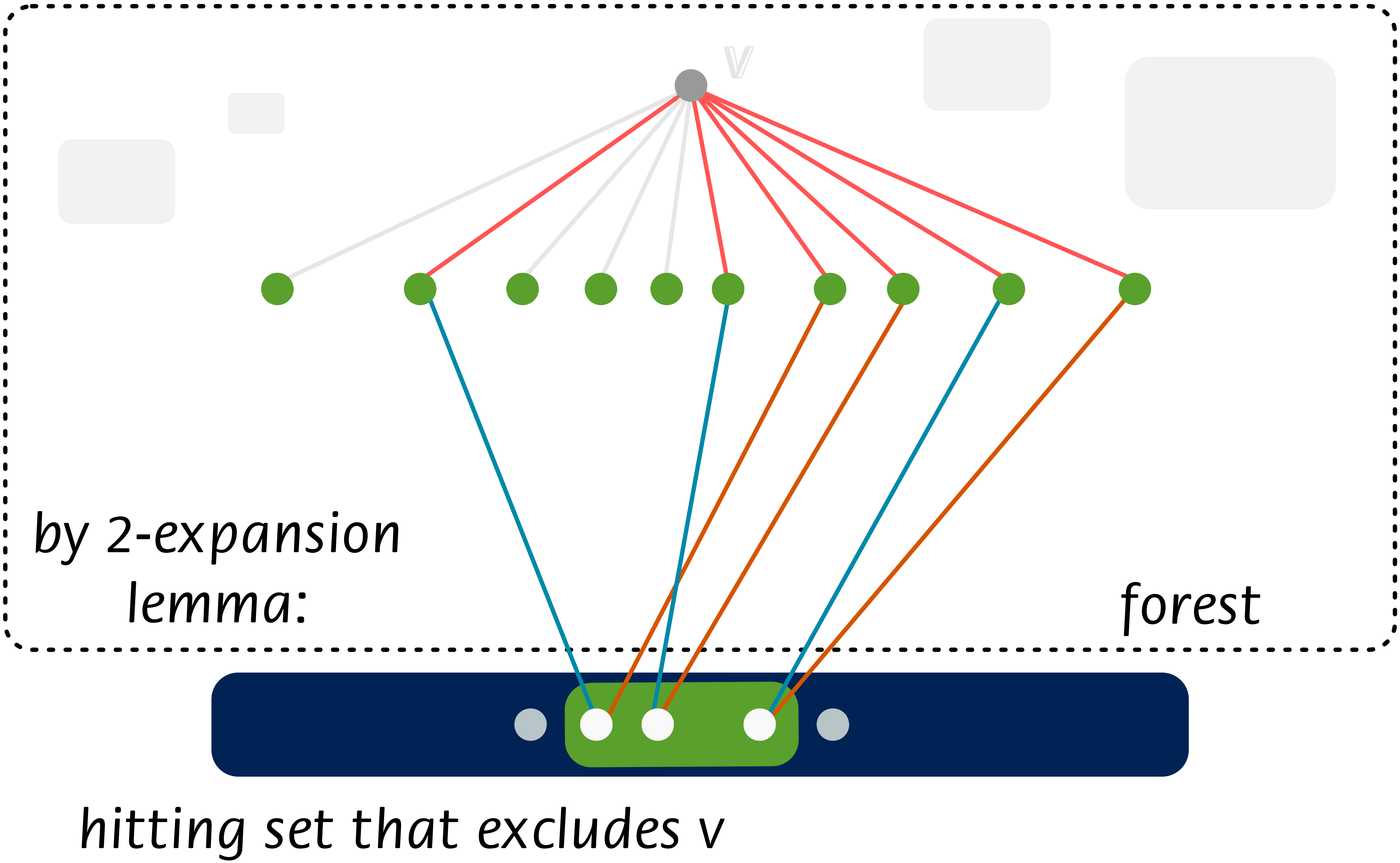
- So we have A and B. We put an edge between a vertex x in A and a vertex w in B, if x is adjacent to some vertex in the component represented by the vertex w. Essentially, we have obtained this bipartite graph by contracting the components.
- If $|B| < 2|A| \leqslant 8k$ then v already has its degree bounded by $8k$. So assume that

$$|B| > 2|A|$$

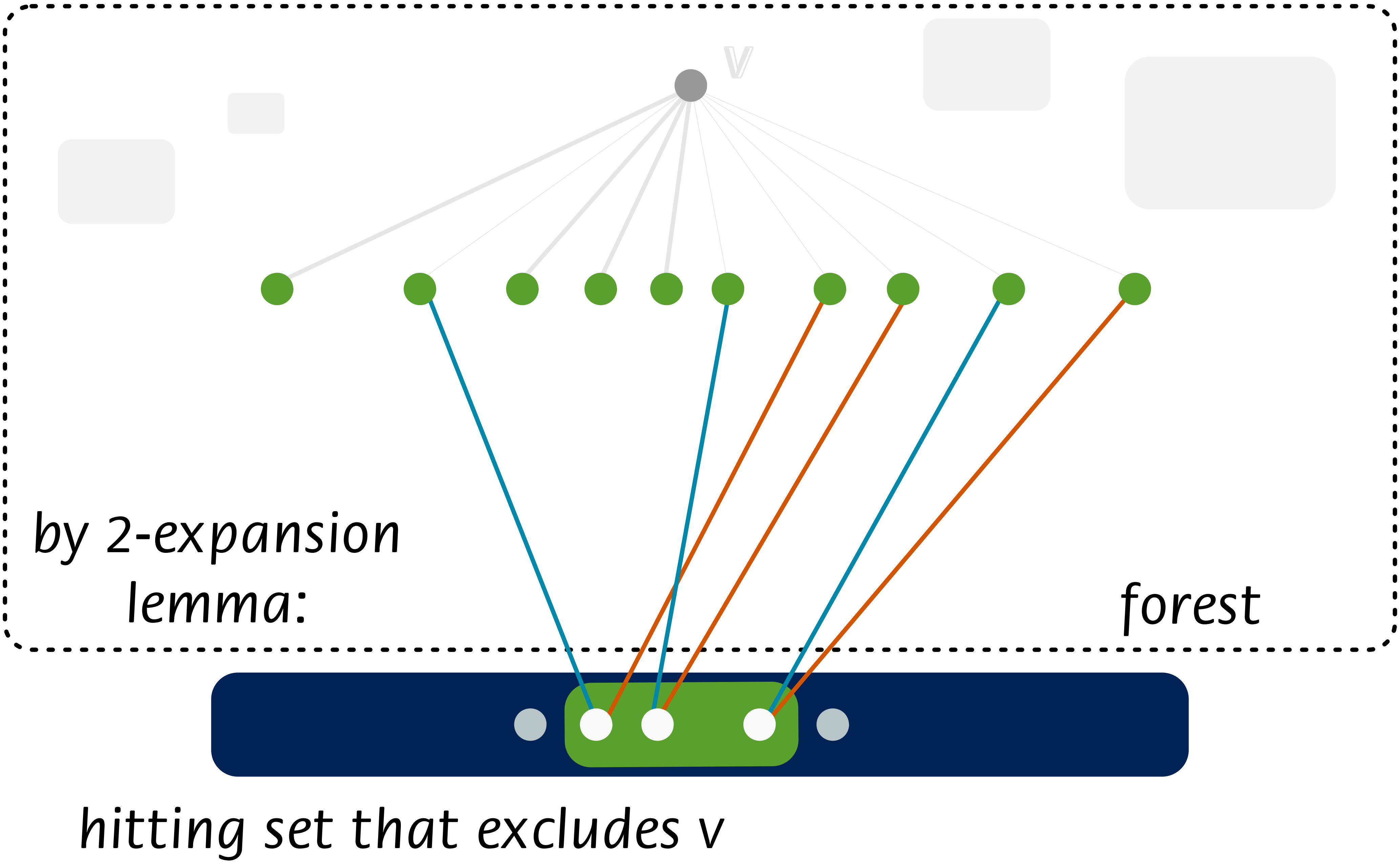
Now by expansion lemma (applied with $q = 2$) we have that there exist nonempty vertex sets $X \subseteq A$ and $Y \subseteq B$ such that

- there is a 2-expansion of X into Y, and
- no vertex in Y has a neighbor outside X, that is, $N(Y) \subseteq X$.

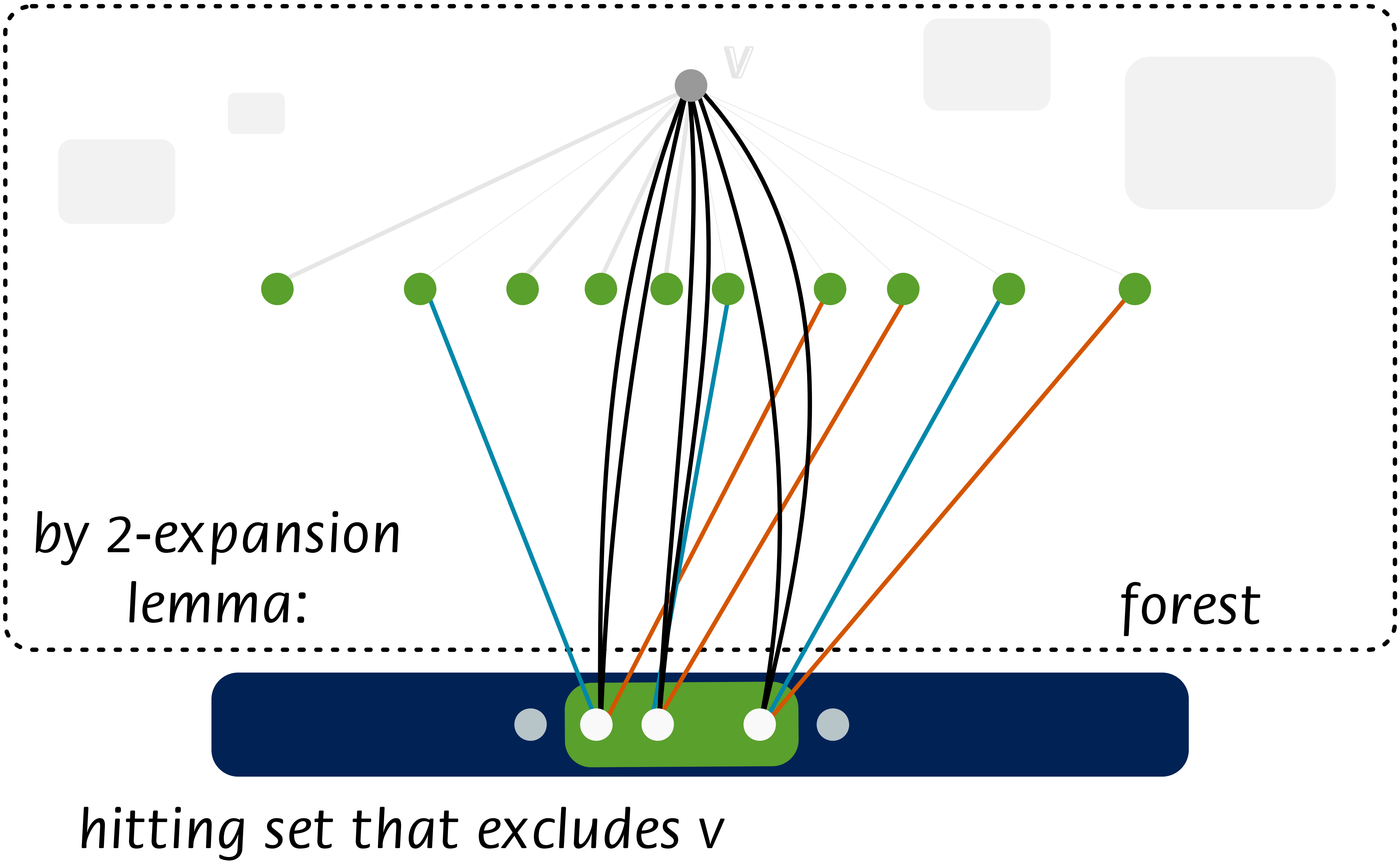

v
by 2-expansion lemma:
forest
hitting set that excludes v


v
by 2-expansion
lemma:
forest
hitting set that excludes v

So the reduction rule is:


v
by 2-expansion lemma:
forest
hitting set that excludes v

... and add the following edges if already not present.

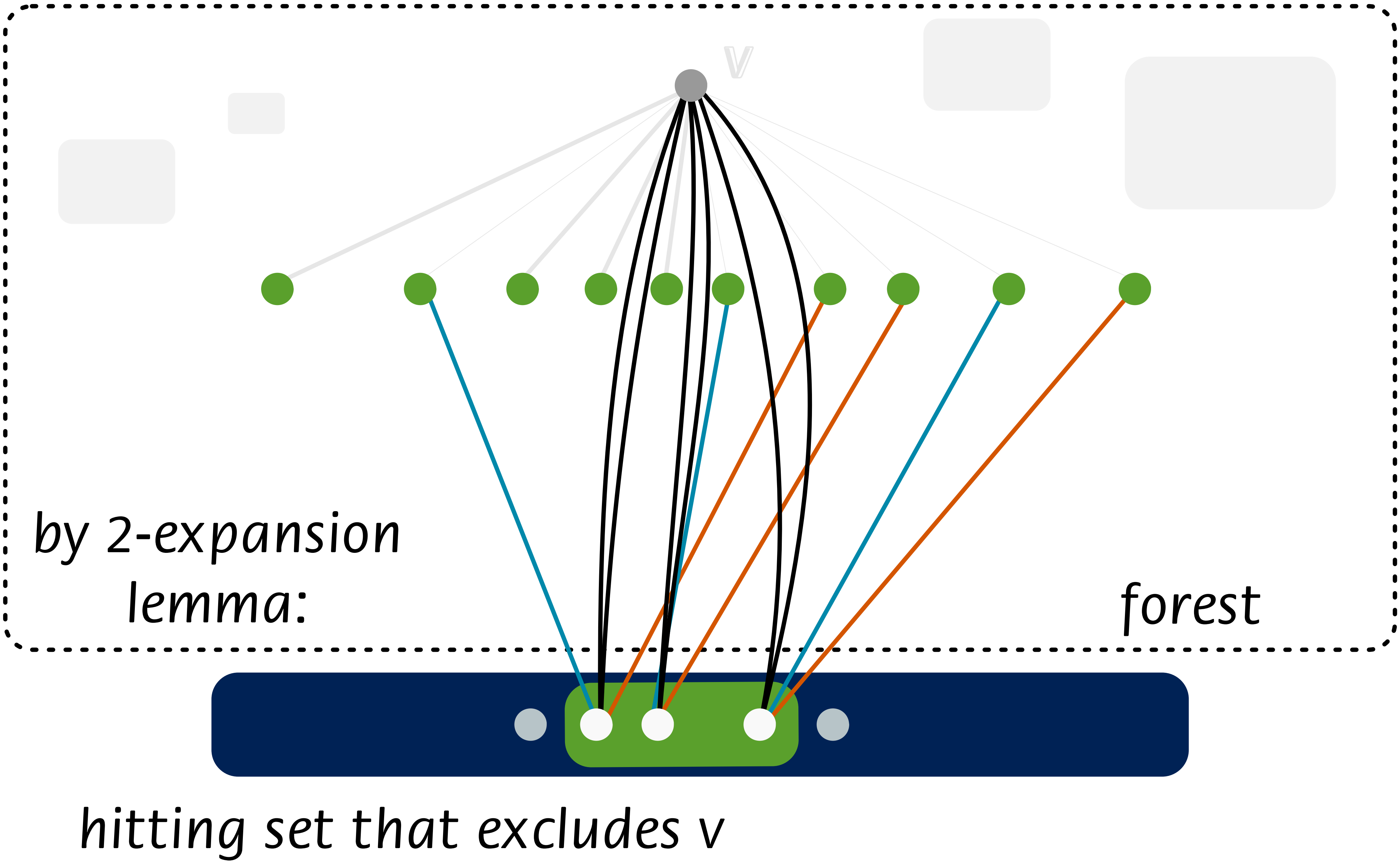

v
by 2-expansion
lemma:
forest
hitting set that excludes v

Let us argue correctness!

The Forward Direction

The Forward Direction

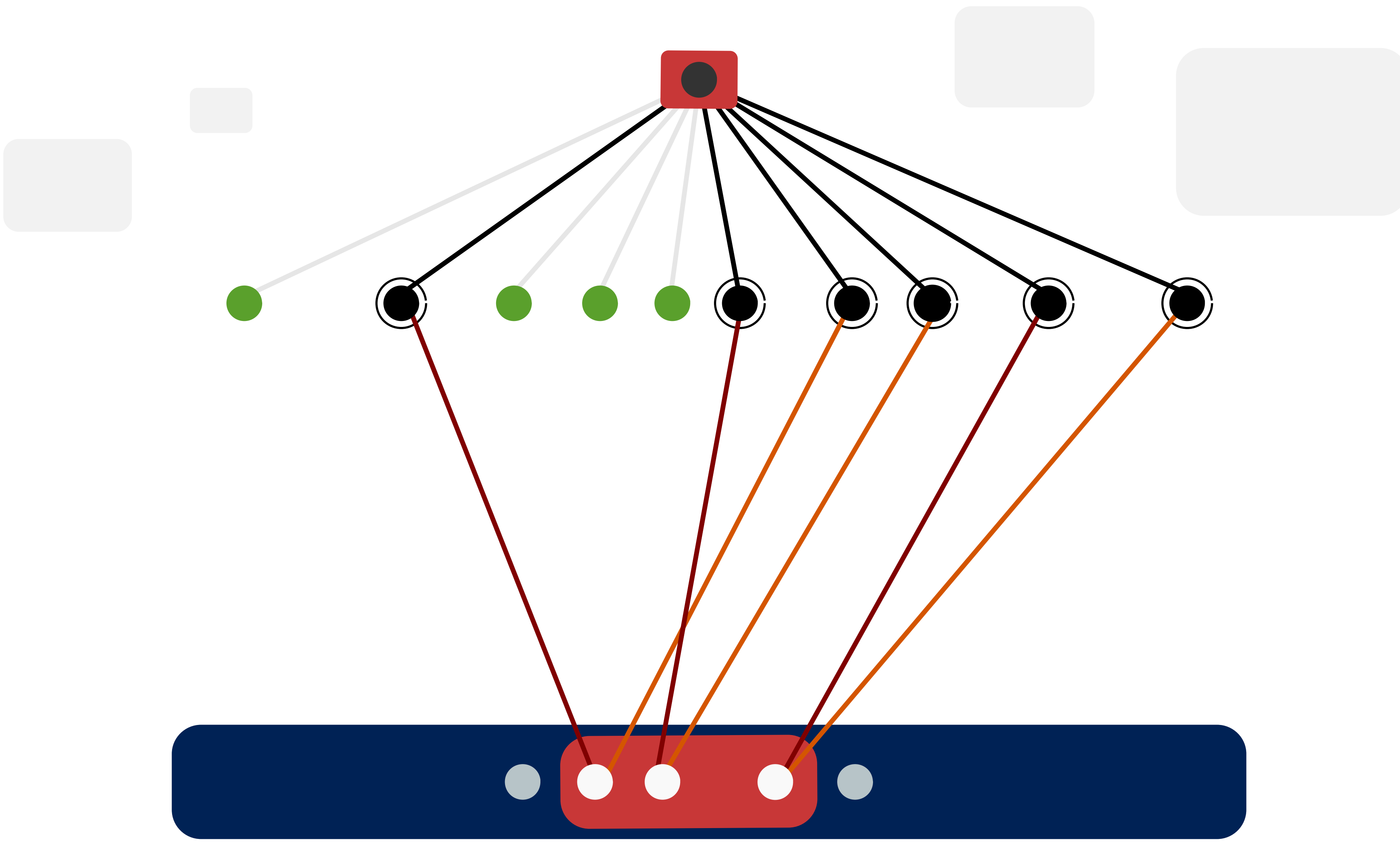
$$\text{FVS} \leqslant \text{k in } \mathsf{G} \Rightarrow \text{FVS} \leqslant \text{k in } \mathsf{H}$$

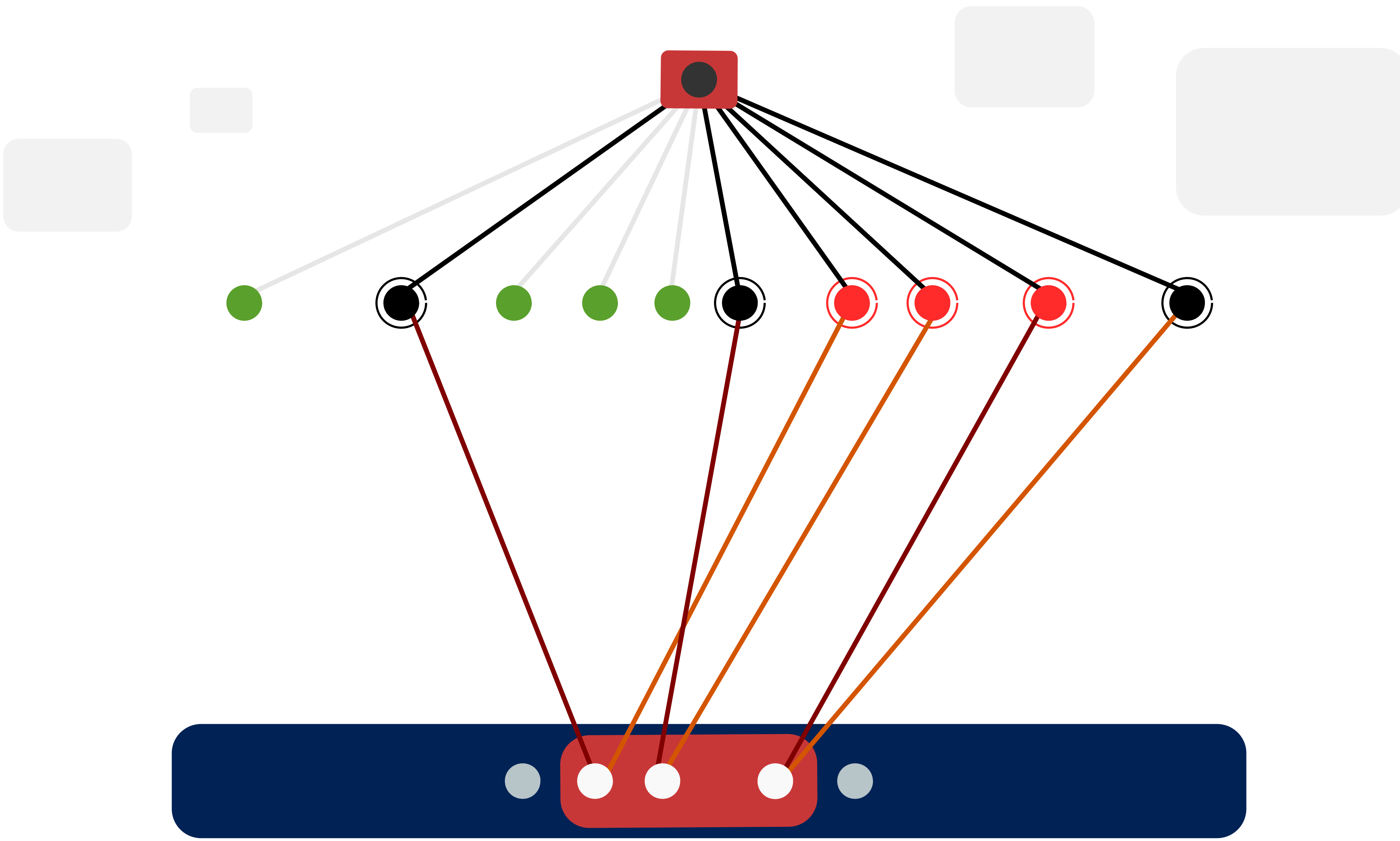

v
by 2-expansion
lemma:
forest
hitting set that excludes v

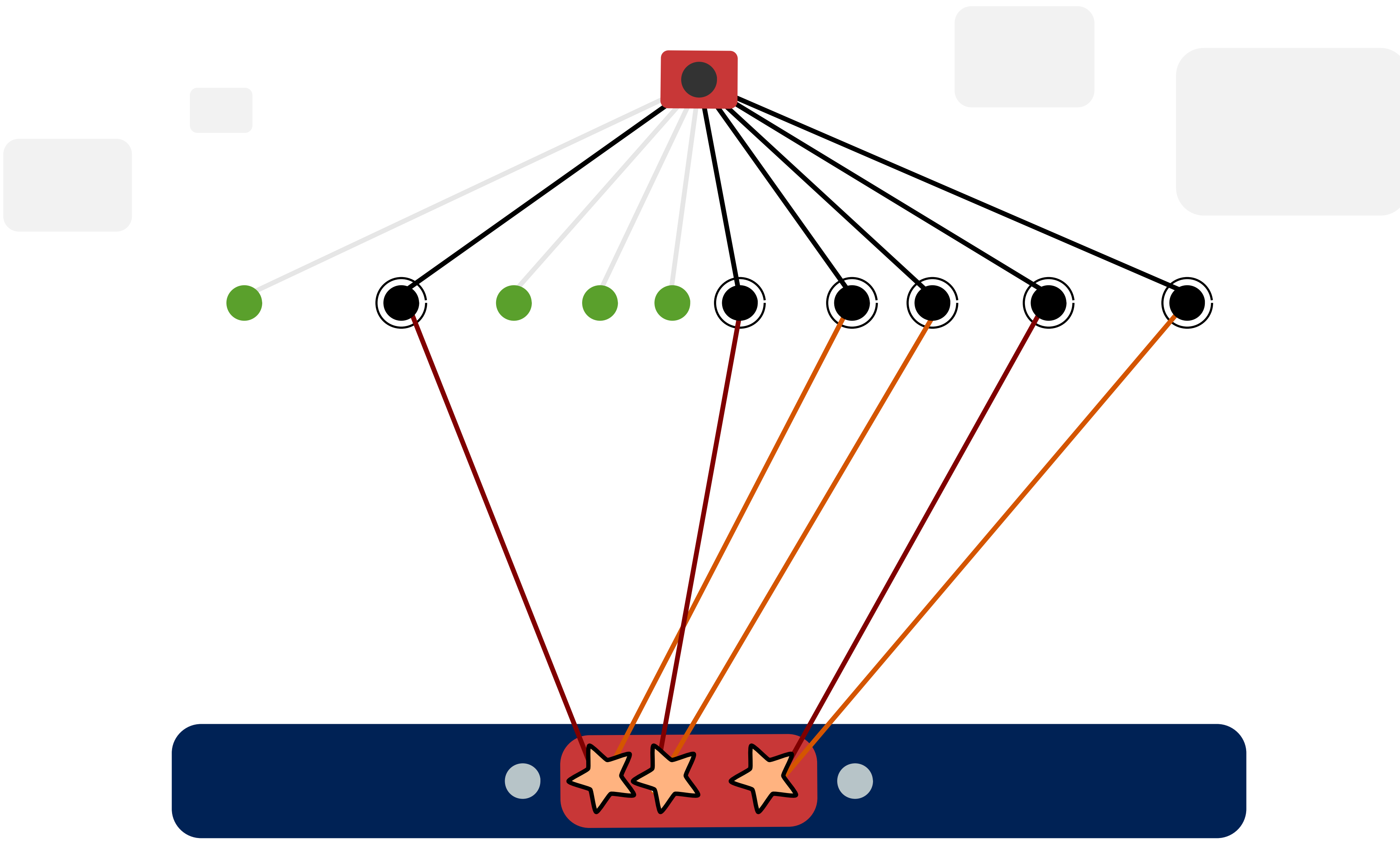
If G has a FVS that either contains v or all of X,
we are in good shape.

Consider now a FVS that:

- Does not contain v,
- and omits at least one vertex of X.







Notice that this does not lead to a larger FVS:

Notice that this does not lead to a larger FVS:

For every vertex v in X that a FVS of G leaves out,

Notice that this does not lead to a larger FVS:

For every vertex v in X that a FVS of G leaves out,

it must pick a vertex u that kills no more than all of X.

The Reverse Direction

The Reverse Direction

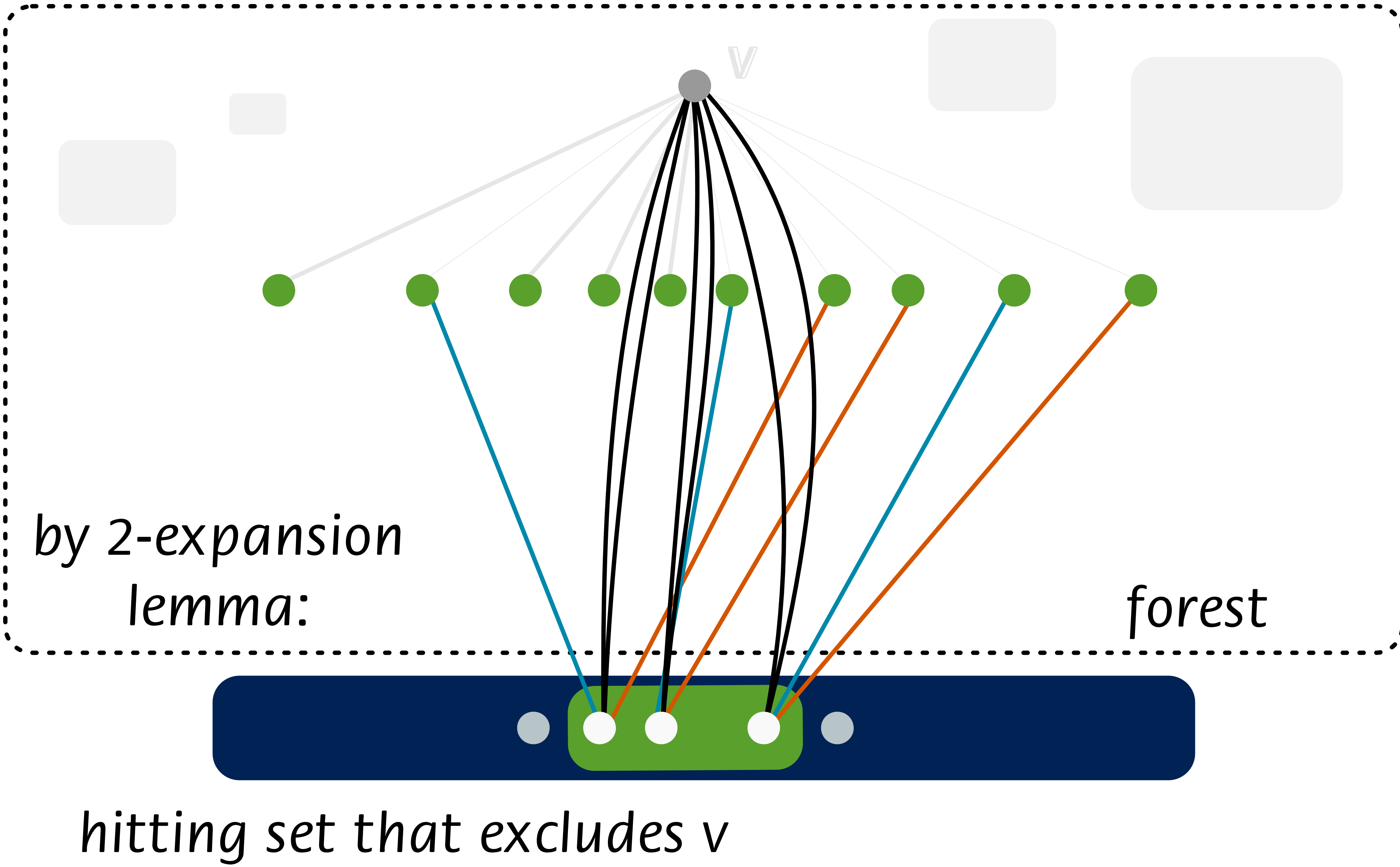
$$\text{FVS} \leqslant \mathsf{k} \text{ in } \mathsf{G} \Leftarrow \text{FVS} \leqslant \mathsf{k} \text{ in } \mathsf{H}$$

The Reverse Direction

The Reverse Direction

$$\text{FVS} \leqslant k \text{ in } \mathsf{G} \Leftarrow \text{FVS} \leqslant k \text{ in } \mathsf{H}$$

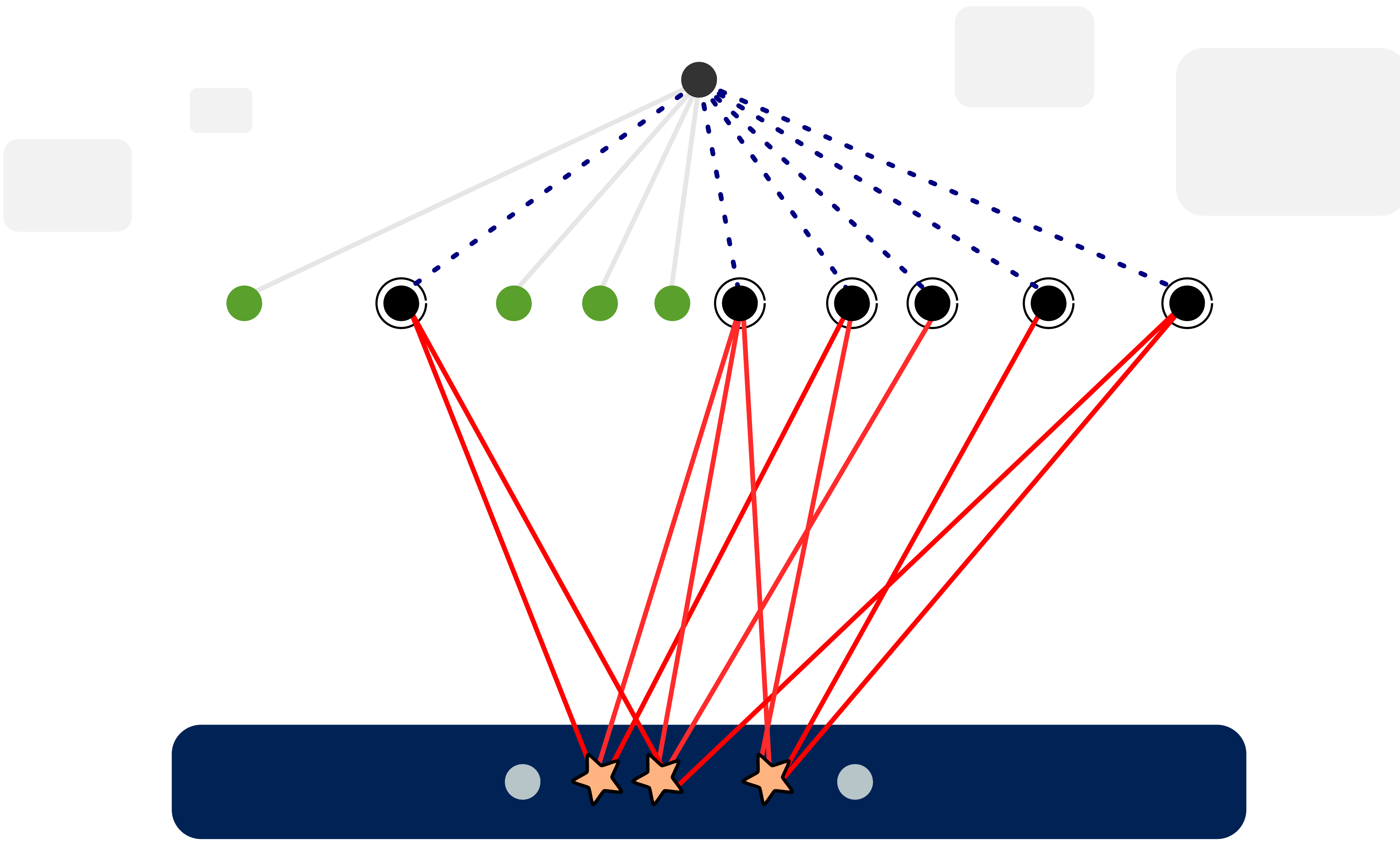
If FVS in H contains ν then the same works for G also as $\mathsf{G} \setminus \{\nu\}$ is isomorphic to $\mathsf{H} \setminus \{\nu\}$. So assume that FVS in H does not contain ν.


v
by 2-expansion
lemma:
forest
hitting set that excludes v

Let W be a FVS of H, the Only Danger for W to be a FVS of G:

Cycles that:

- pass through v,
- non-neighbors of v in H (neighbors in G, however)
- and do not pass through X.

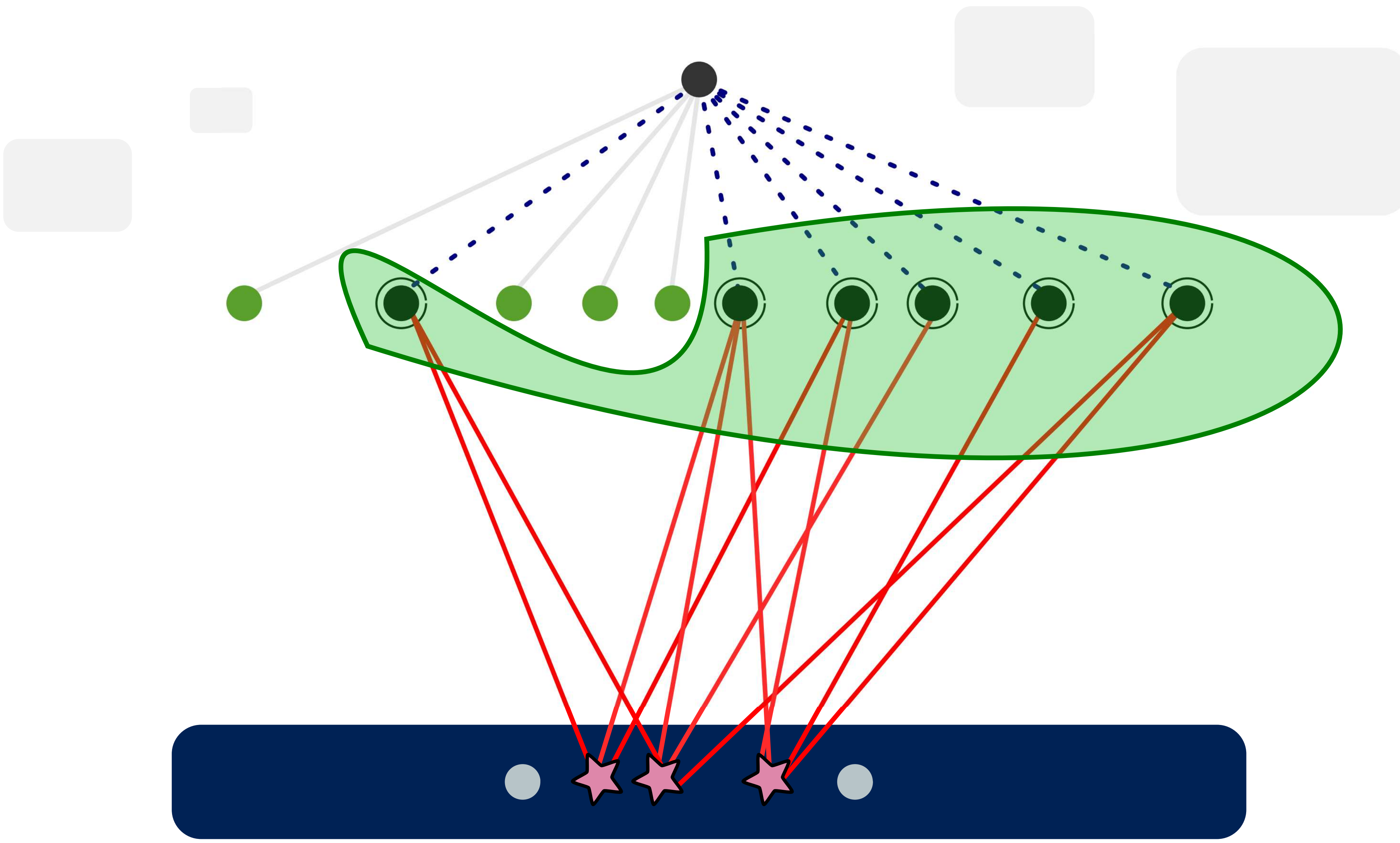


Let W be a FVS of H, the Only Danger for W to be a FVS of G:

Cycles that:

- pass through v,
- non-neighbors of v in H (neighbors in G, however)
- and do not pass through X.

However recall that $\mathsf{N}(\mathsf{Y}) \subseteq \mathsf{X}$.



Wrapping Up

- A priori it is not obvious that previous Reduction Rule actually makes some simplification of the graph, since it *substitutes some set of edges with some other set of double edges!*
-

Wrapping Up

- A priori it is not obvious that previous Reduction Rule actually makes some simplification of the graph, since it *substitutes some set of edges with some other set of double edges!*
- We need to formally prove that the reduction rules cannot be applied infinitely, or superpolynomially many times.

Final Result

Theorem

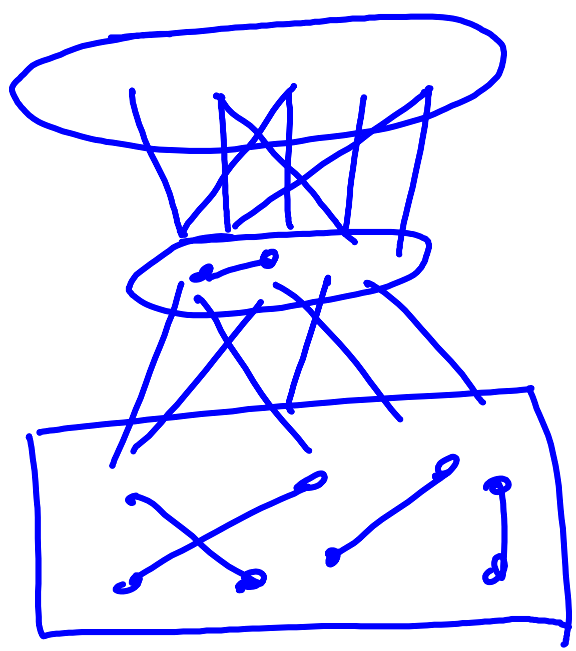
FEEDBACK VERTEX SET *admits a kernel with at most* $O(k^2)$ *vertices and* $O(k^2)$ *edges.*

Crown Decomposition

Crown Decomposition

A partition of the vertex set of a graph into 3 parts (crown)C, (head)H and (the rest) R, such that:

- C is non-empty and an independent set, with edges to vertices of H alone.
- The bipartite graph between C and H in G contains a matching of size $|H|$.



Crown Decomposition

Lemma

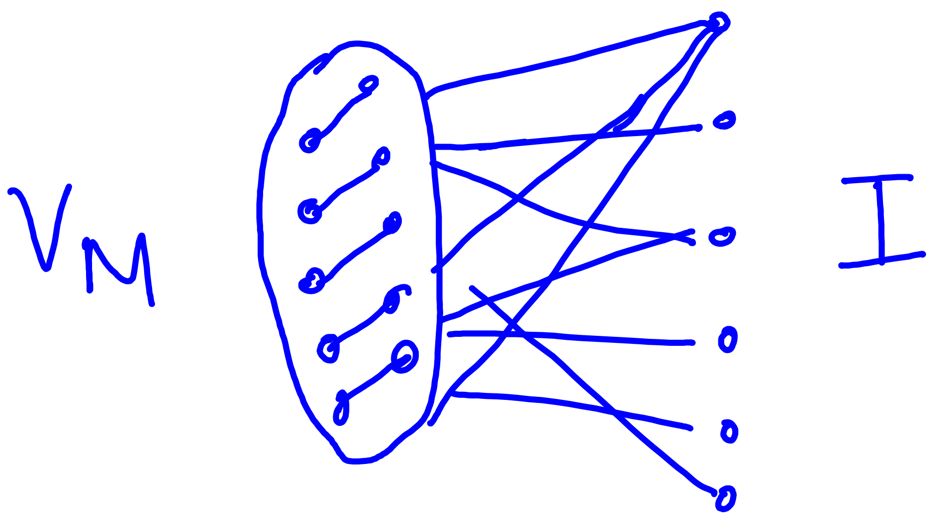
Let G be a graph on at least $3k+1$ vertices. Then in polynomial time, either we can find a matching of size $k+1$ or a Crown Decomposition of G.

Crown Decomposition

Lemma

Let G be a graph on at least $3k+1$ vertices. Then in polynomial time, either we can find a matching of size $k+1$ or a Crown Decomposition of G.

- Find a greedy matching M of G, if $|M| \geq k+1$ we are done
- Else V_M be the endpoints of M and $I = V(G) \setminus V_M$
- Consider the bipartite graph G' between V_M and I, compute a minimum vertex cover X of G'



Crown Decomposition

Lemma

Let G be a graph on at least $3k+1$ vertices. Then in polynomial time, either we can find a matching of size $k+1$ or a Crown Decomposition of G.

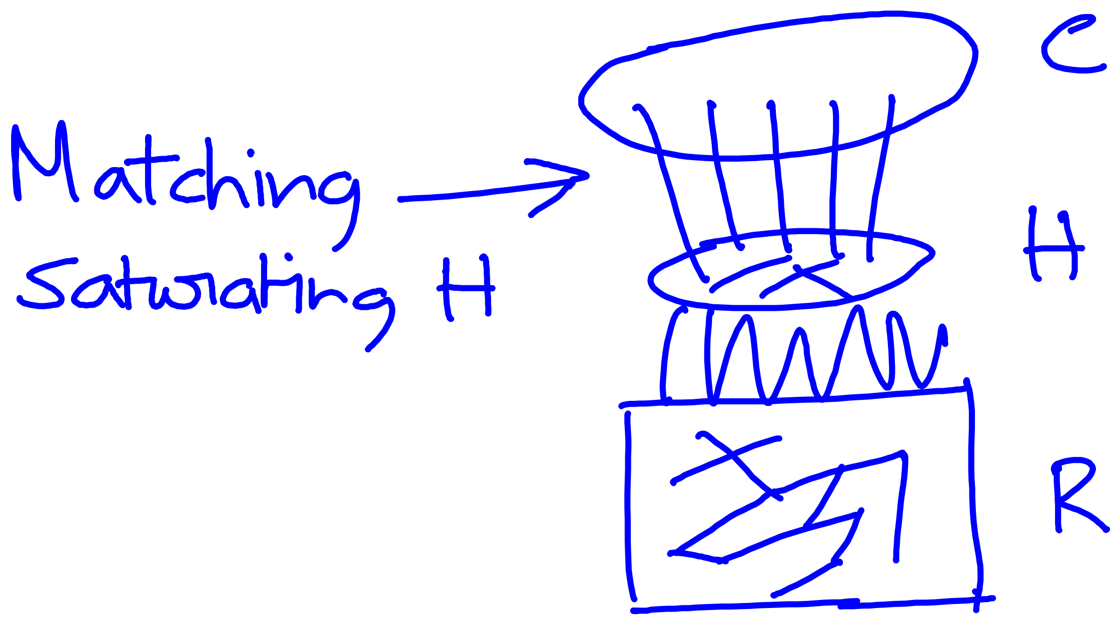
- Find a greedy matching M of G, if $|M| \geq k+1$ we are done
- Else V_M be the endpoints of M and $I = V(G) \setminus V_M$
- Consider the bipartite graph G' between V_M and I, compute a minimum vertex cover X of G'
- If $X \cap V_M = \emptyset$, then $|I| \leq k$, and hence $|V(G)| \leq 3k$
- Else, M' be a maximum matching in G', and M^* is subset of edges with exactly one endpoint in X.
- Crown Decomposition:

$$C = V(M^*) \cap I,\ H = V(M^*) \cap X, R$$

Crown Decomposition

Vertex Cover kernel on $3k$ vertices.

- Remove all isolated vertices in G
- Find a Crown Decomposition (C, H, R) or a $k+1$ matching
- In the former case, the reduced instance is ~~$(G - C, k - |C|)$~~ $(G - C \cup H, k - |H|)$
- In the latter case, a trivial no instance



A surprising equivalence

Theorem

A parameterized problem is FPT if and only if it is decidable and has a kernel (of arbitrary size).

A surprising equivalence

Theorem

A parameterized problem is FPT if and only if it is decidable and has a kernel (of arbitrary size).

Proof:

- If the problem has a kernel:
  Reducing the size of the instance to $f(k)$ in poly time + brute force
  $\Rightarrow$ problem is FPT.

A surprising equivalence

Theorem

A parameterized problem is FPT if and only if it is decidable and has a kernel (of arbitrary size).

Proof:

- If the problem has a kernel:
  Reducing the size of the instance to $f(k)$ in poly time + brute force
  $\Rightarrow$ problem is FPT.
- If the problem can be solved in time $f(k)|x|^{O(1)}$:
  - If $|x| \leq f(k)$, then we already have a kernel of size $f(k)$.
  - If $|x| \geq f(k)$, then we can solve the problem in time $f(k)|x|^{O(1)} \leq |x| \cdot |x|^{O(1)}$ (polynomial in $|x|$) and then output a trivial yes- or no-instance.

A surprising equivalence

Theorem

A parameterized problem is FPT if and only if it is decidable and has a kernel (of arbitrary size).

Proof:

- If the problem has a kernel:
  Reducing the size of the instance to $f(k)$ in poly time + brute force
  $\Rightarrow$ problem is FPT.
- If the problem can be solved in time $f(k)|x|^{O(1)}$:
  - If $|x| \leq f(k)$, then we already have a kernel of size $f(k)$.
  - If $|x| \geq f(k)$, then we can solve the problem in time $f(k)|x|^{O(1)} \leq |x| \cdot |x|^{O(1)}$ (polynomial in $|x|$) and then output a trivial yes- or no-instance.
- The existence of kernels is not a separate question...
- ...but the existence of **polynomial kernels** is a deep and nontrivial topic!