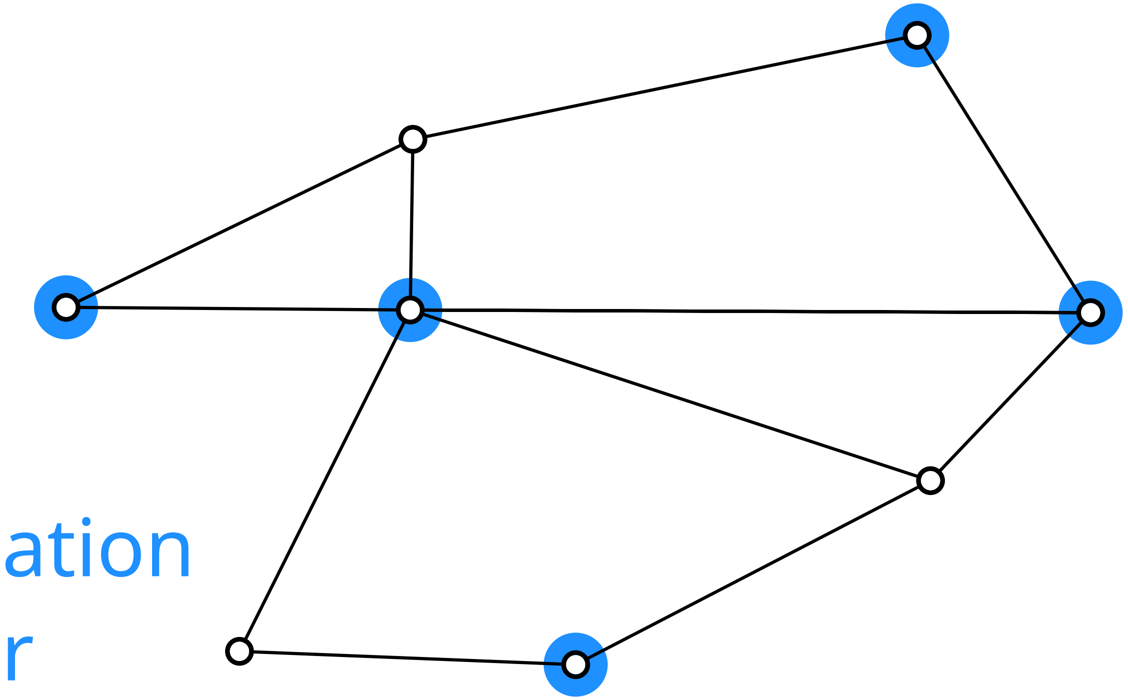
Approximation Algorithms

Introduction
NP-Optimization Problems and Approximation
Approximation Algorithm for Vertex Cover



"All exact science is dominated by the idea of approximation."

– Bertrand Russell (1872 – 1970)

Approximation Algorithms

- Many optimization problems are NP-hard!
  examples: traveling salesperson problem, maximum satisfiability

Approximation Algorithms

- Many optimization problems are NP-hard!
  examples: traveling salesperson problem, maximum satisfiability
- $\rightsquigarrow$ an optimal solution cannot be efficiently computed unless P=NP.

Approximation Algorithms

- Many optimization problems are NP-hard!
  examples: traveling salesperson problem, maximum satisfiability
- $\rightsquigarrow$ an optimal solution cannot be efficiently computed unless P=NP.
- However, good approximate solutions can often be found efficiently!

Approximation Algorithms

- Many optimization problems are NP-hard!
  examples: traveling salesperson problem, maximum satisfiability
- $\rightsquigarrow$ an optimal solution cannot be efficiently computed unless P=NP.
- However, good approximate solutions can often be found efficiently!
- Techniques for the design and analysis of approximation algorithms arise from studying specific optimization problems.

Approximation Algorithms

- Many optimization problems are NP-hard!
  examples: traveling salesperson problem, maximum satisfiability
- $\rightsquigarrow$ an optimal solution cannot be efficiently computed unless P=NP.
- However, good approximate solutions can often be found efficiently!
- Techniques for the design and analysis of approximation algorithms arise from studying specific optimization problems.

today:

Approximation Algorithms

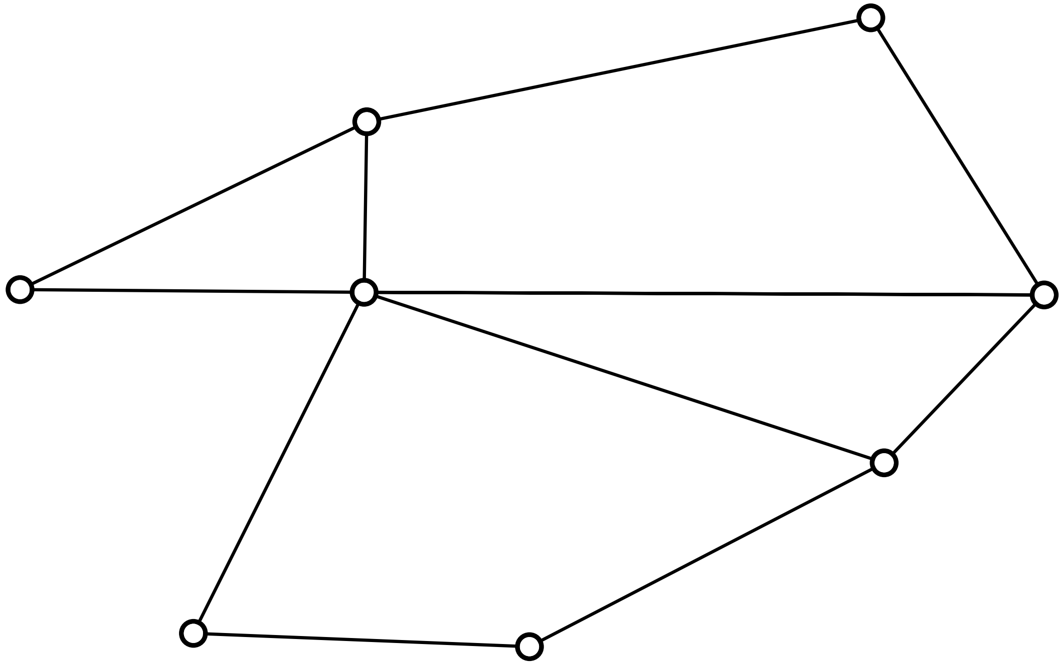
- Many optimization problems are NP-hard!
  examples: traveling salesperson problem, maximum satisfiability
- $\rightsquigarrow$ an optimal solution cannot be efficiently computed unless P=NP.
- However, good approximate solutions can often be found efficiently!
- Techniques for the design and analysis of approximation algorithms
  arise from studying specific optimization problems.

today:
technique: lower bounding optimal solution (key ingredient for approximation!)
optimization problem: vertex cover

VertexCover

Input: Graph $G = (V, E)$

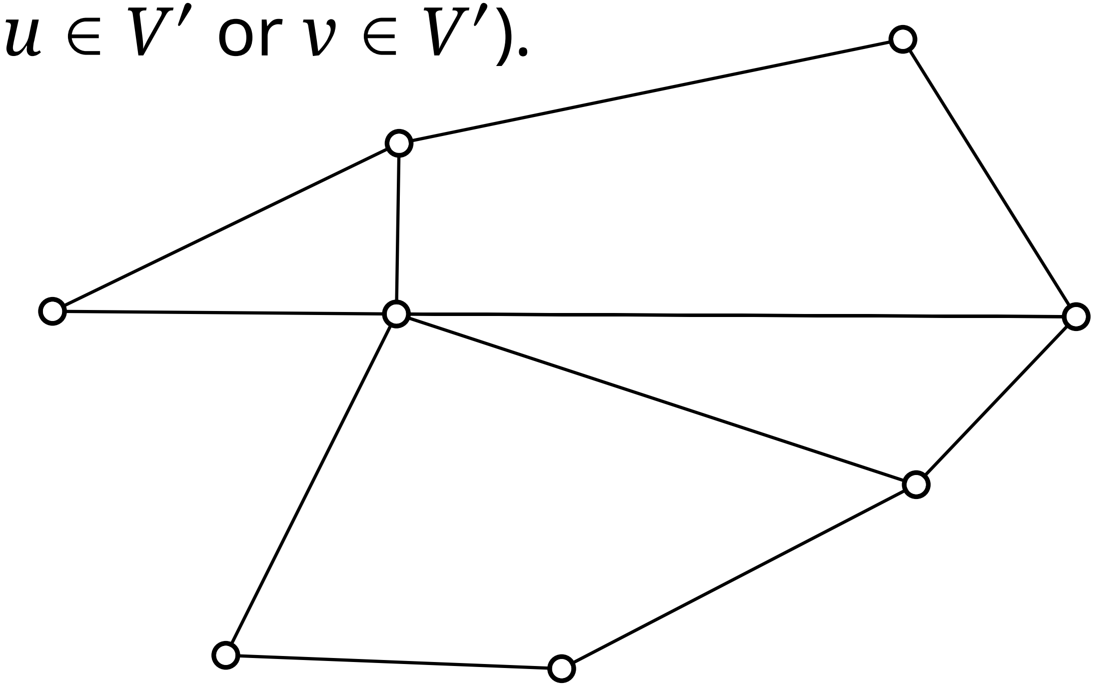
Output:



VertexCover

Input: Graph $G = (V, E)$

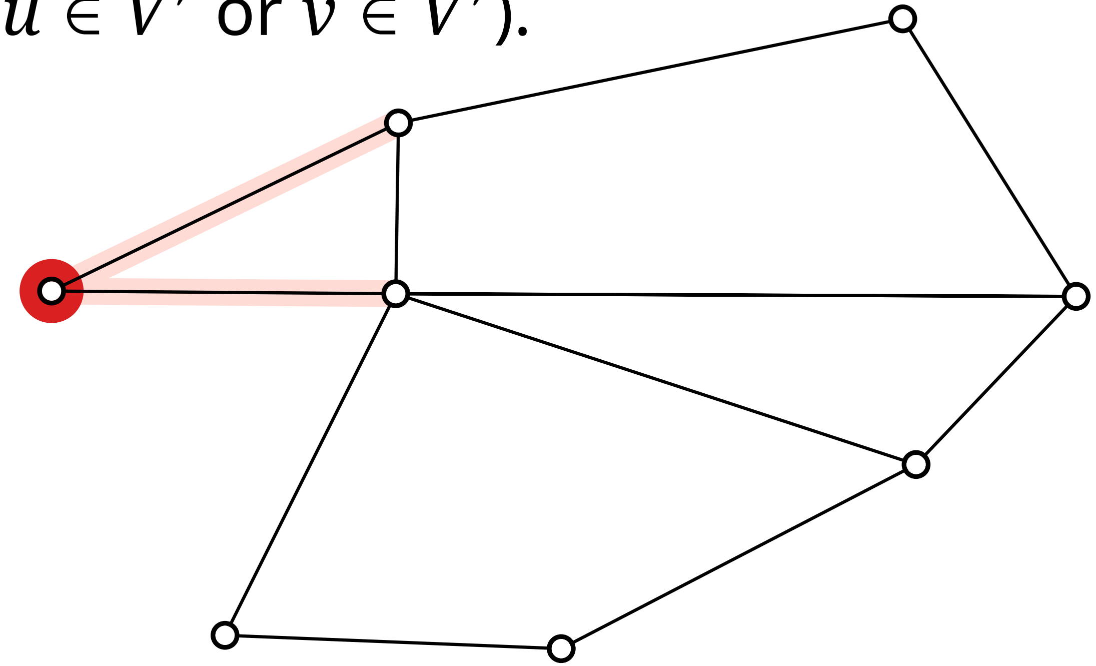
Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that
every edge is covered (i.e., for every $uv \in E$, it holds
that $u \in V'$ or $v \in V'$).



VERTEXCOVER

Input: Graph $G = (V, E)$

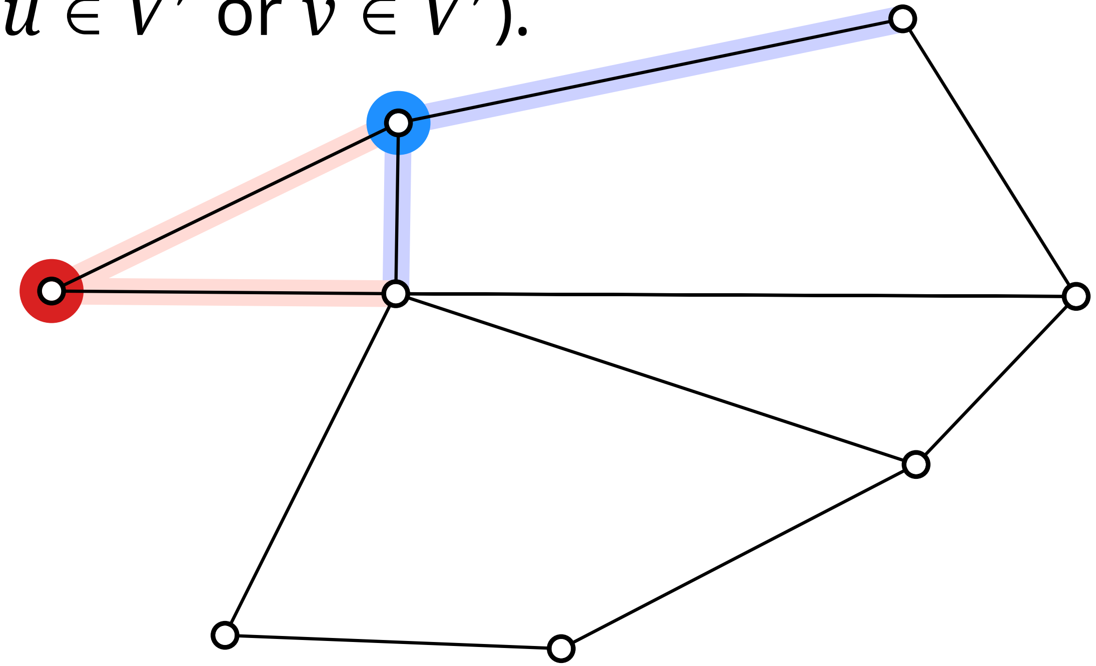
Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is covered (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).



VertexCover

Input: Graph $G = (V, E)$

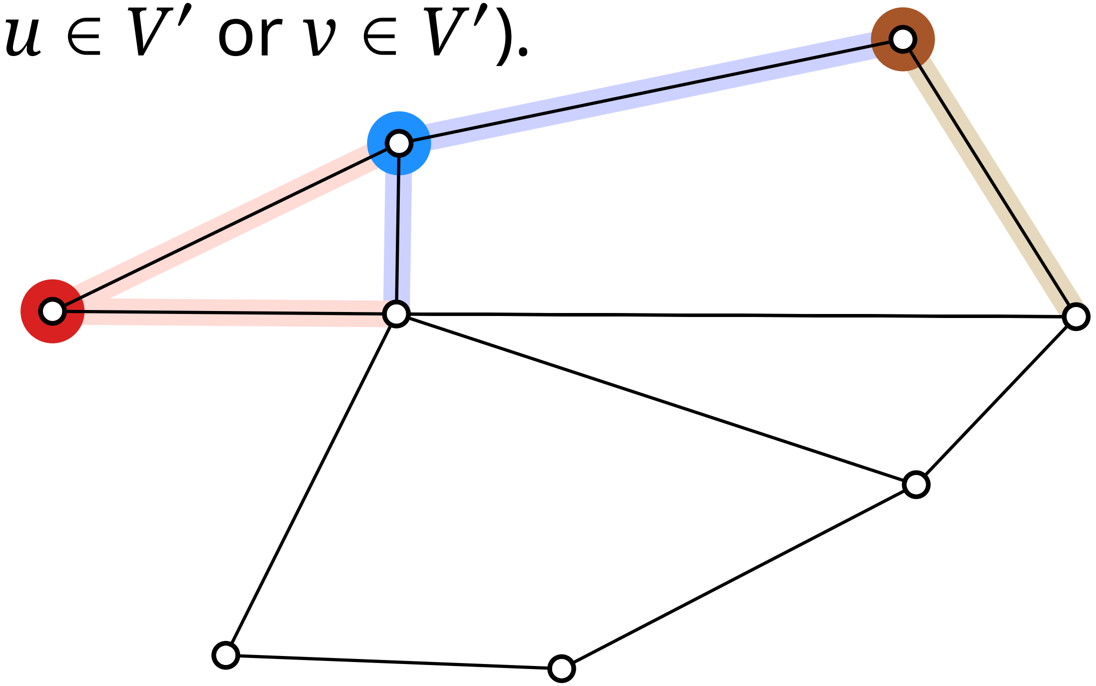
Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is covered (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).



VertexCover

Input: Graph $G = (V, E)$

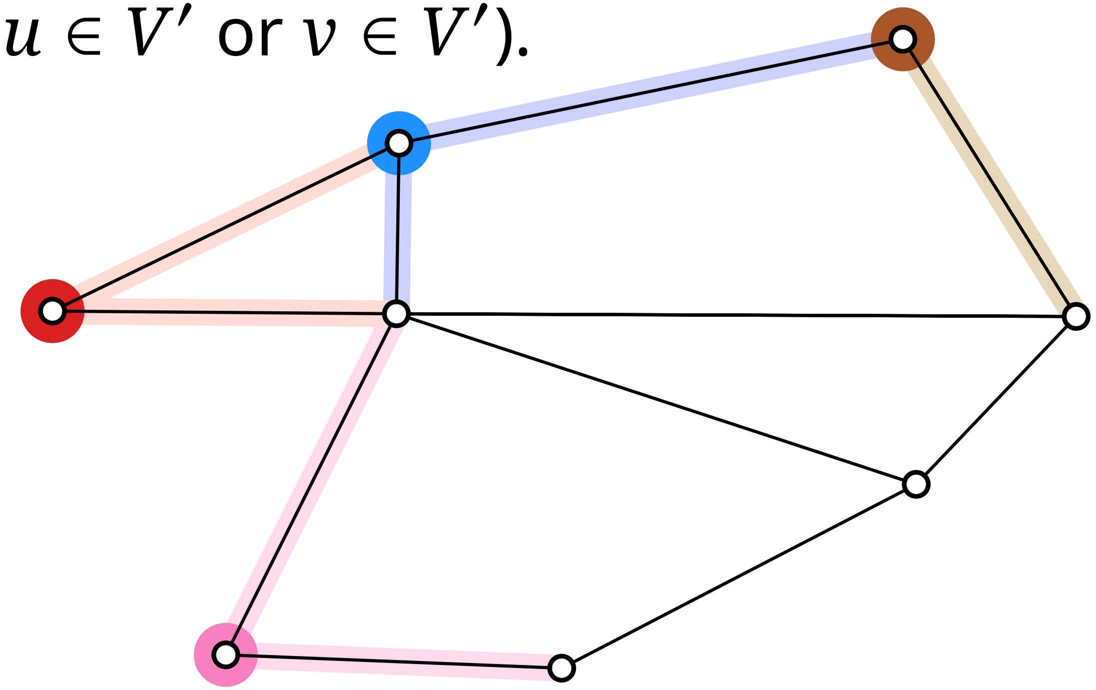
Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is covered (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).



VertexCover

Input: Graph $G = (V, E)$

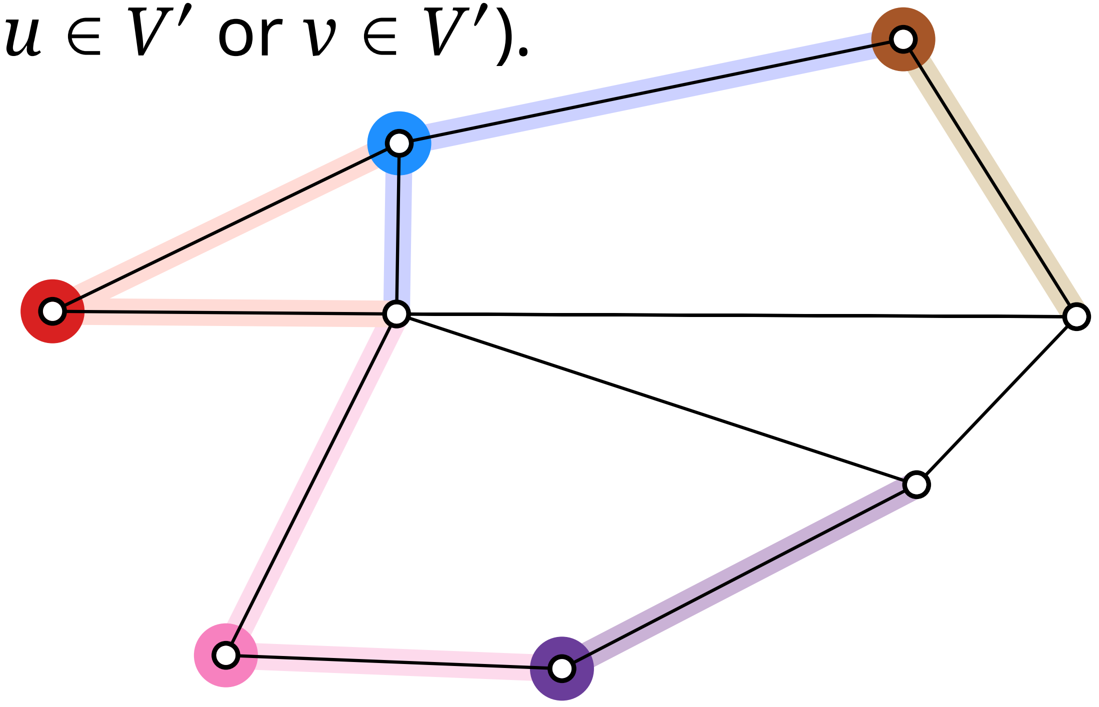
Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is covered (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).



VertexCover

Input: Graph $G = (V, E)$

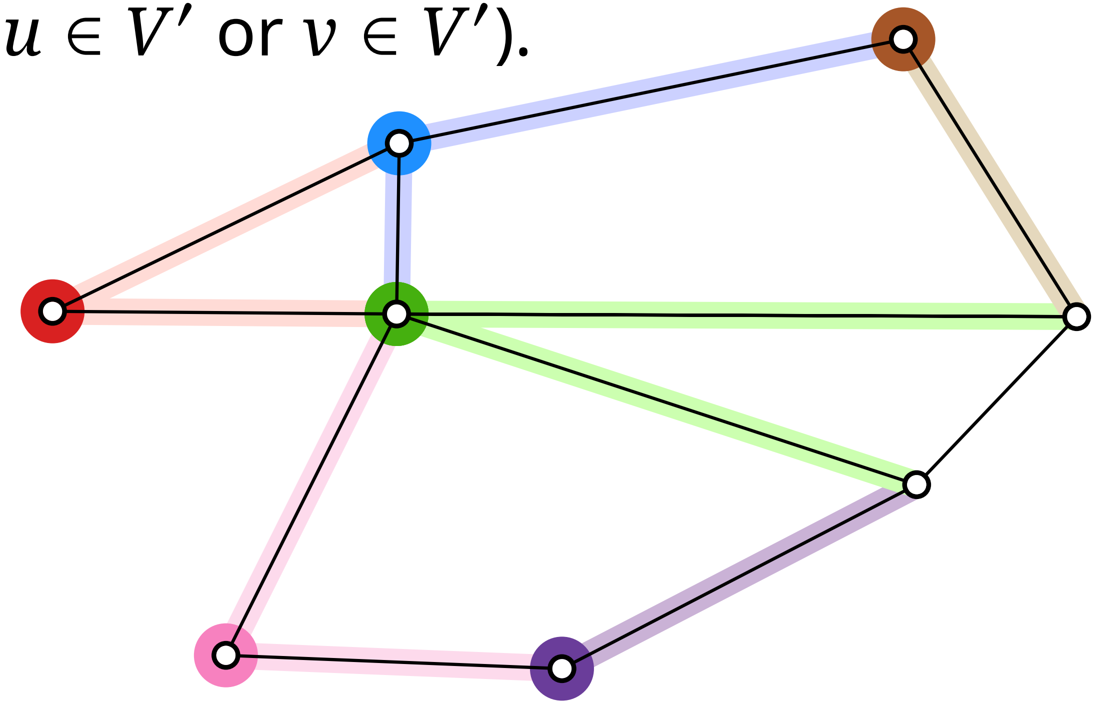
Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that
every edge is covered (i.e., for every $uv \in E$, it holds
that $u \in V'$ or $v \in V'$).



VertexCover

Input: Graph $G = (V, E)$

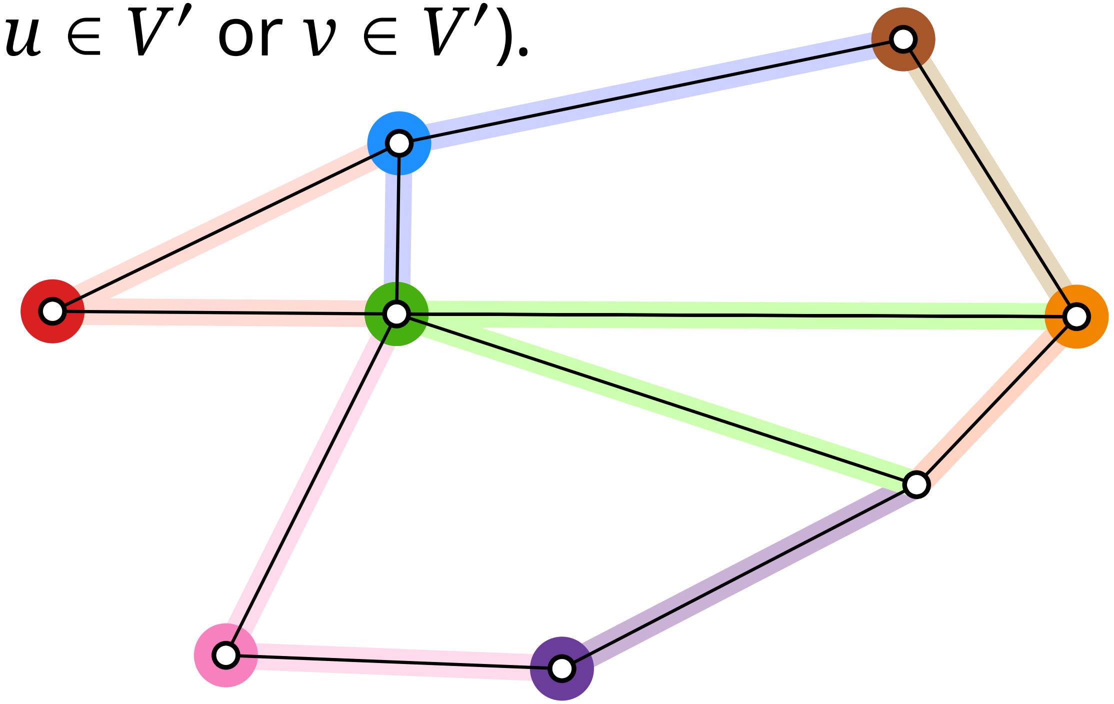
Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is covered (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).



VERTEXCOVER

Input: Graph $G = (V, E)$

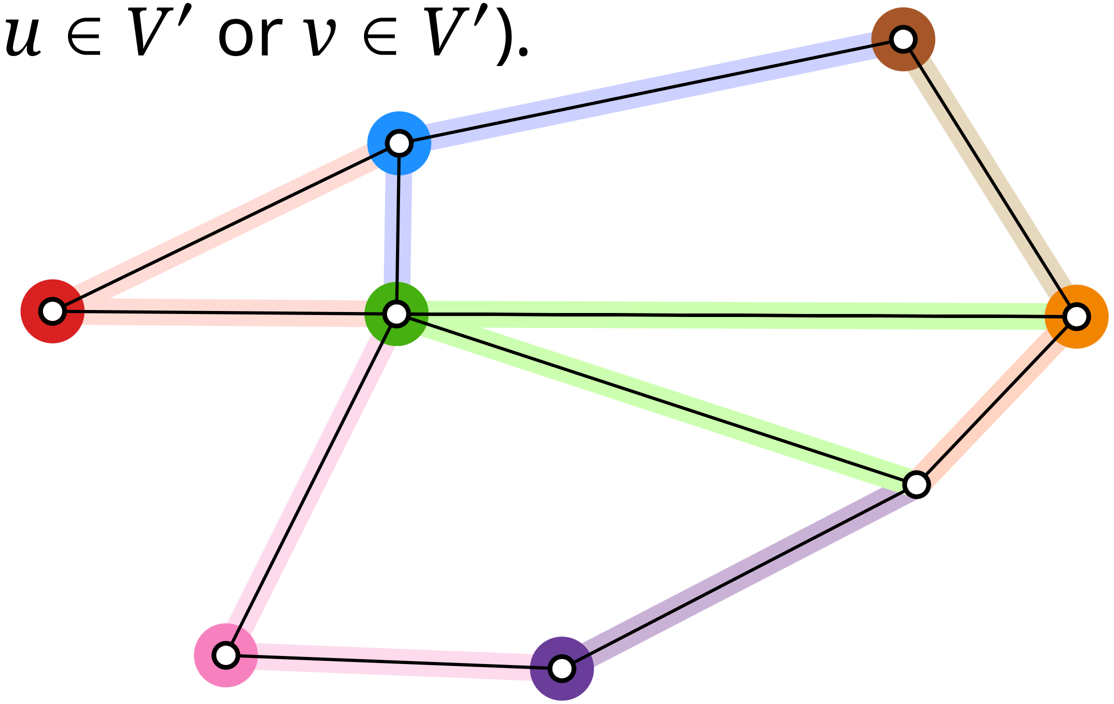
Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is covered (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).



VertexCover

Input: Graph $G = (V, E)$

Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is covered (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).

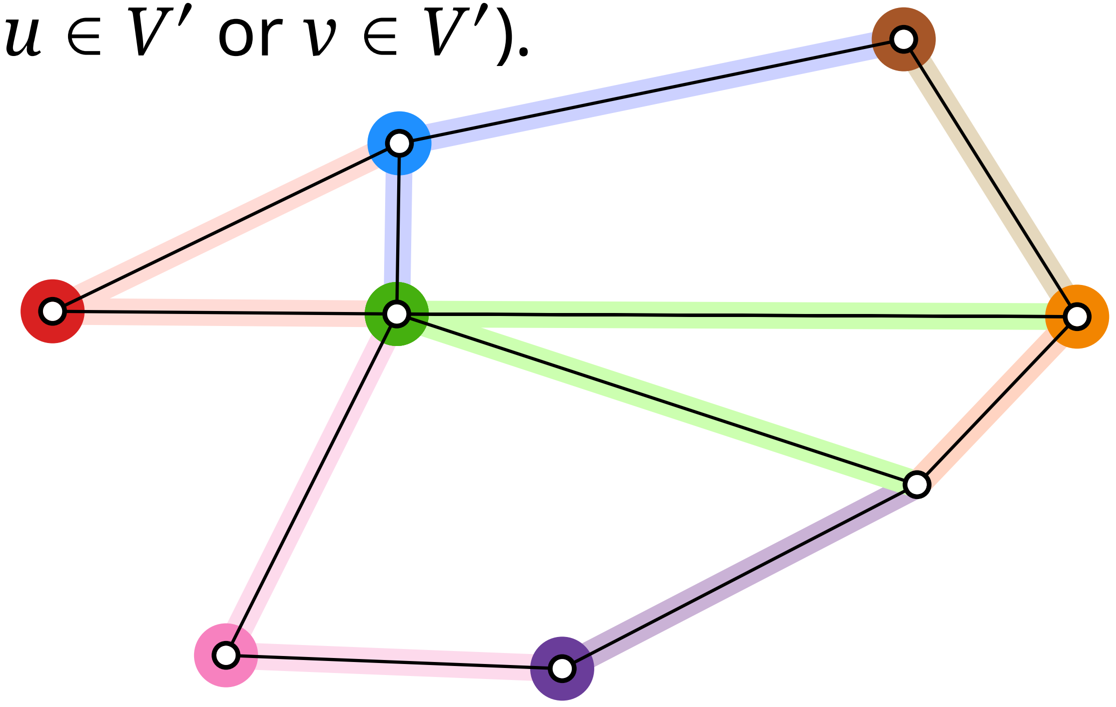


this is a vertex cover

VertexCover

Input: Graph $G = (V, E)$

Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is covered (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).

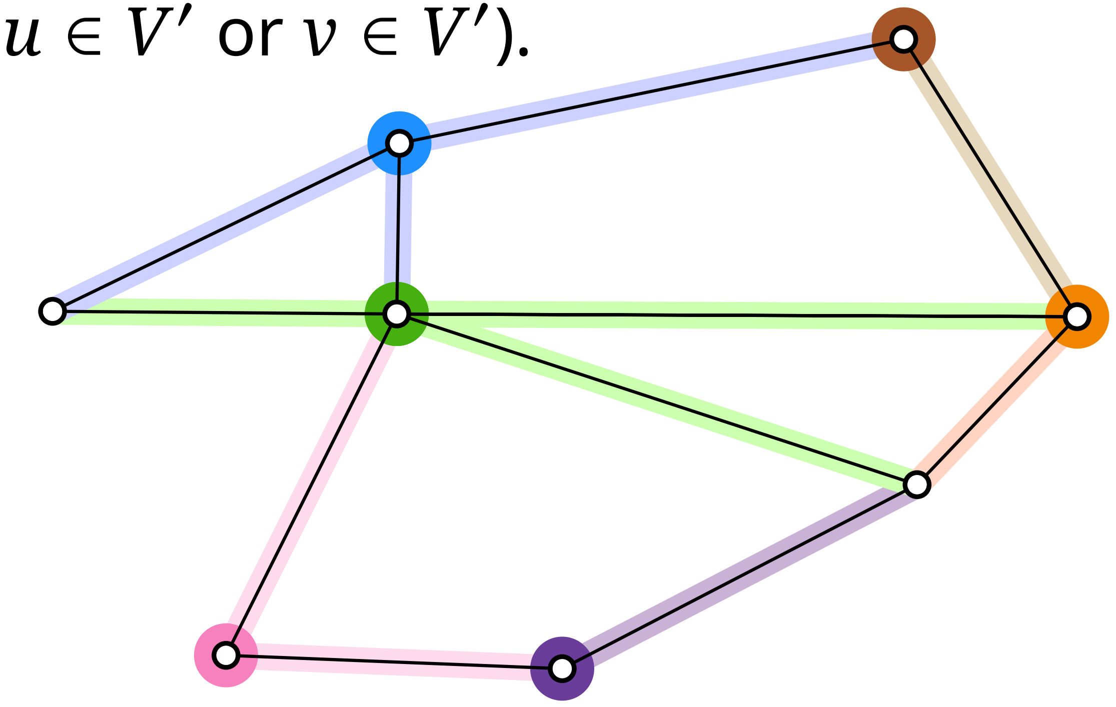


this is a vertex cover Q: is this optimal ?

VertexCover

Input: Graph $G = (V, E)$

Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is covered (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).

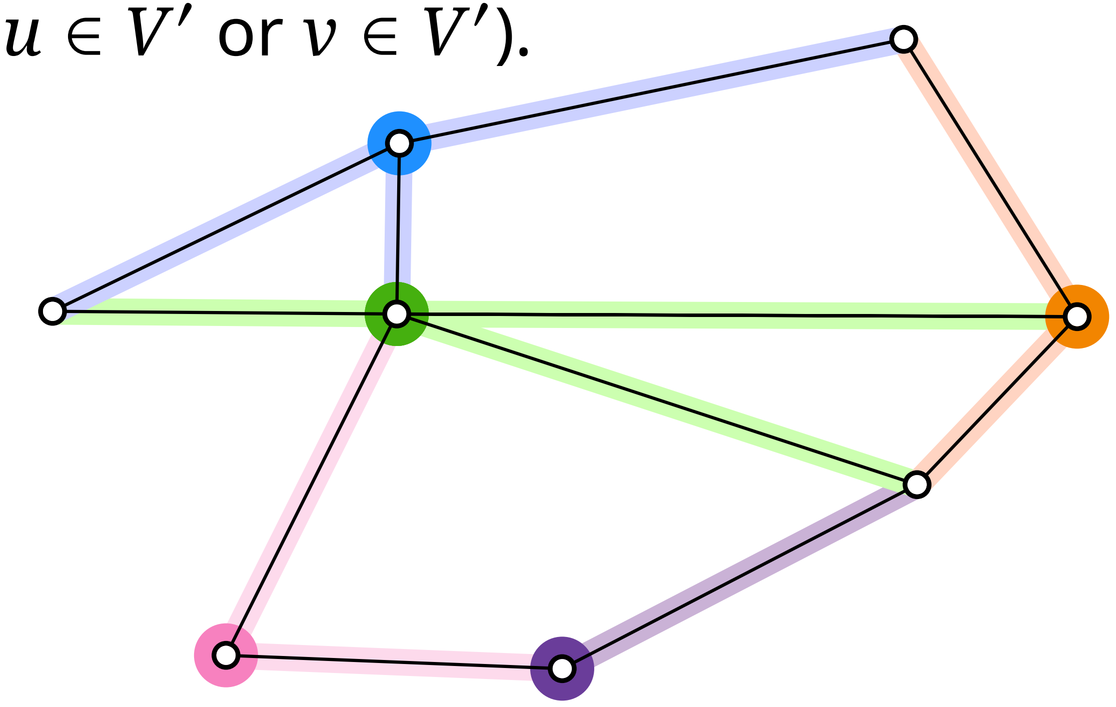


this is a vertex cover Q: is this optimal ?

VertexCover

Input: Graph $G = (V, E)$

Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is covered (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).

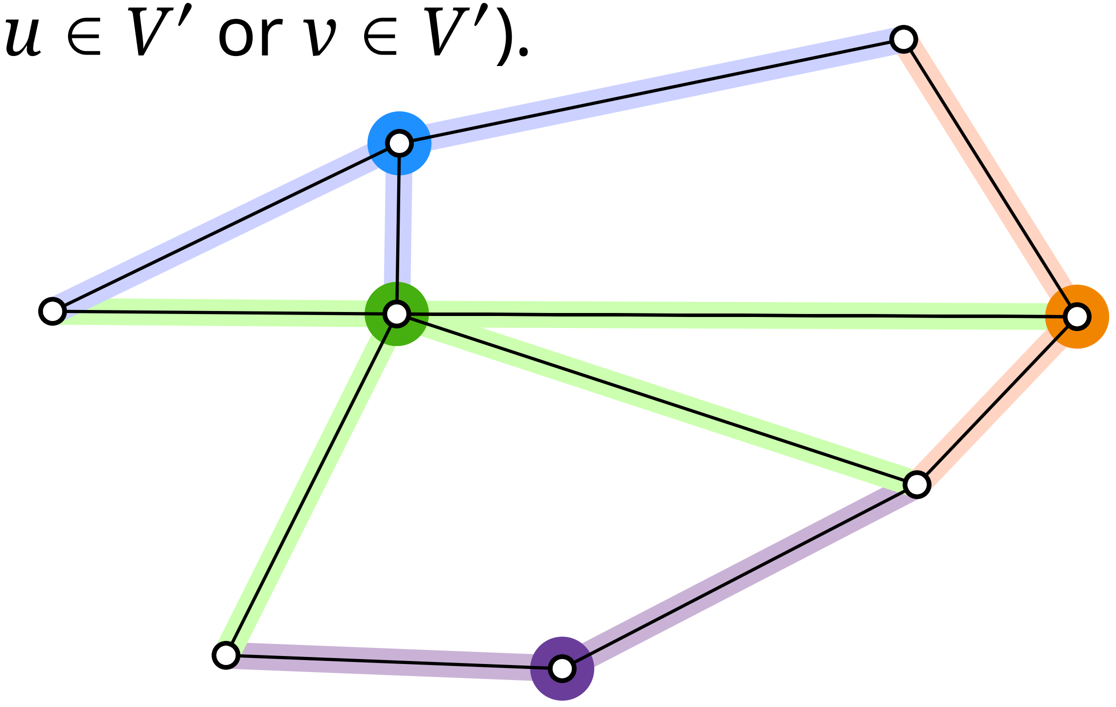


this is a vertex cover Q: is this optimal ?

VertexCover

Input: Graph $G = (V, E)$

Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is covered (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).

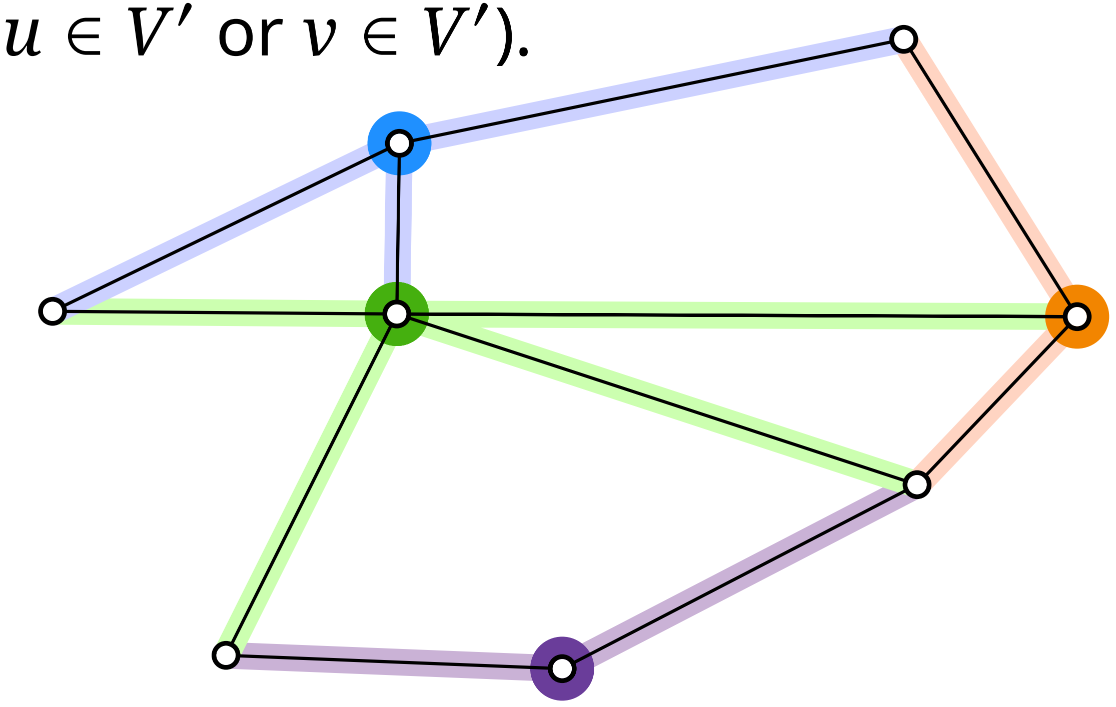


this is a vertex cover Q: is this optimal ?

VertexCover

Input: Graph $G = (V, E)$

Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is covered (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).

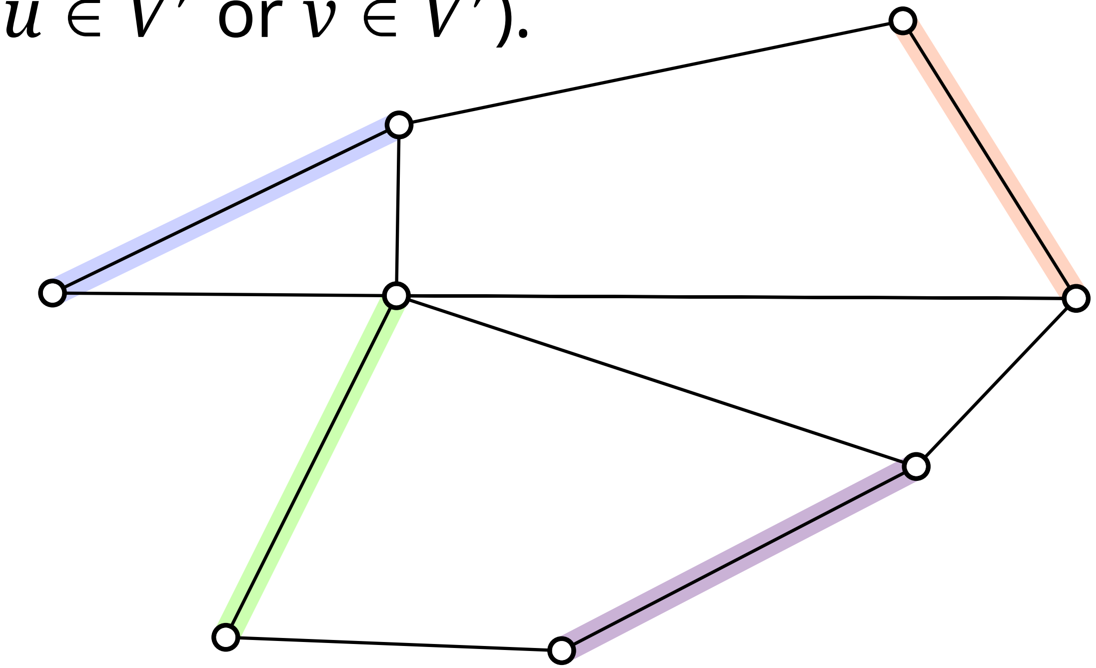


this is a vertex cover Q: can you argue that it is optimal ?

VertexCover

Input: Graph $G = (V, E)$

Output: a minimum vertex cover, that is, a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is covered (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).

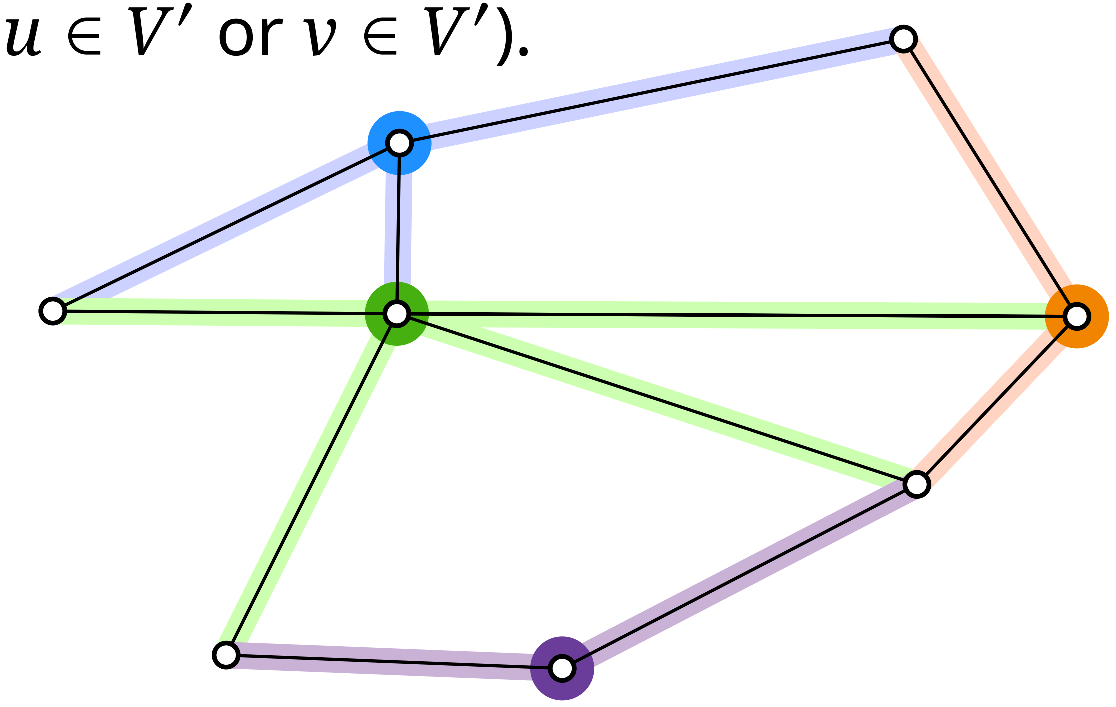


this is a vertex cover 1 vertex per colored edge needed

VertexCover

Input: Graph $G = (V, E)$

Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is covered (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).

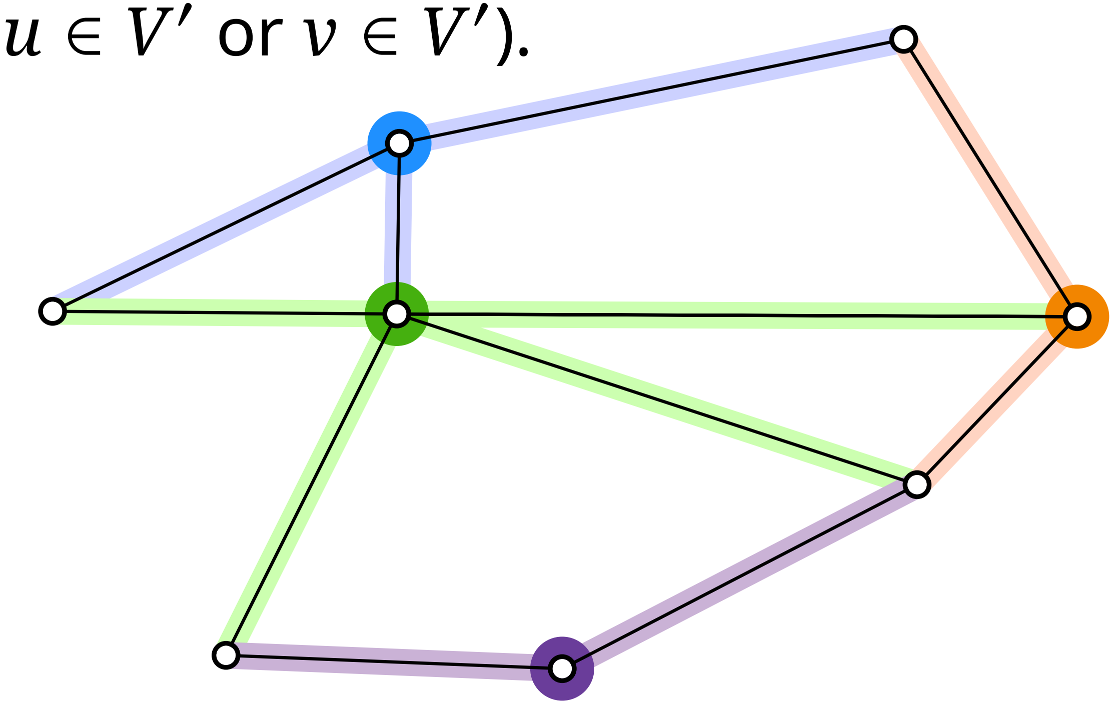


Optimum (OPT $= 4$) – but in general NP-hard to find :-(

VertexCover

Input: Graph $G = (V, E)$

Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is covered (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).



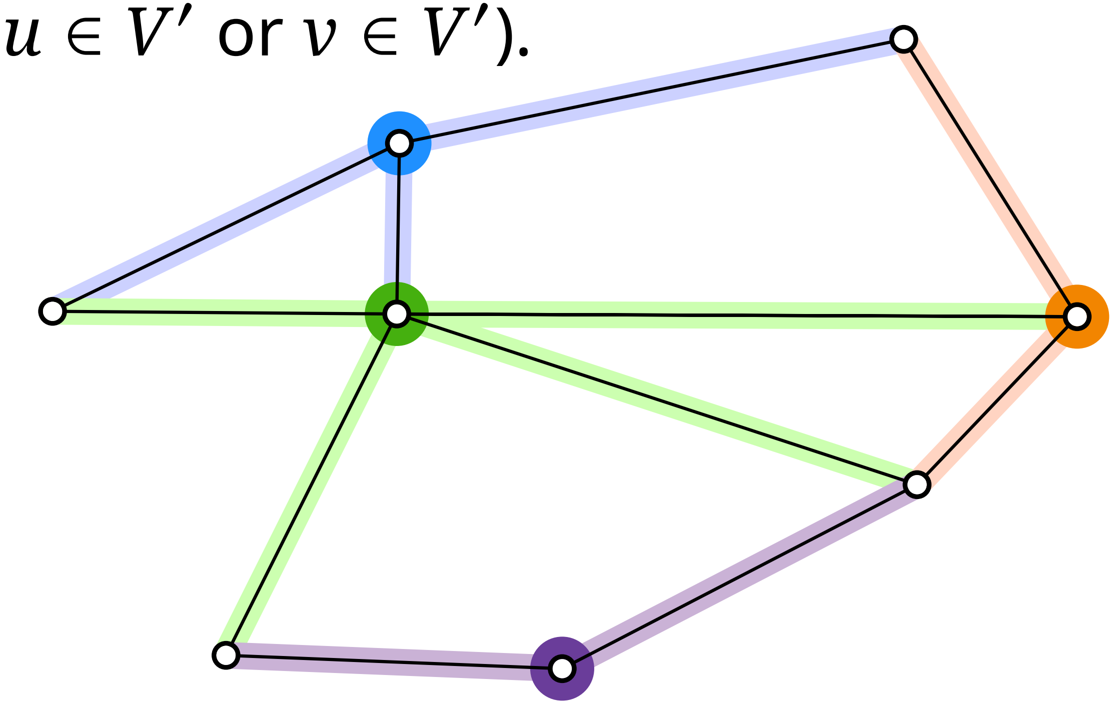
Exact Algorithm: ???

Optimum ($\mathrm{OPT} = 4$) – but in general NP-hard to find :-(

VertexCover

Input: Graph $G = (V, E)$

Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that
every edge is covered (i.e., for every $uv \in E$, it holds
that $u \in V'$ or $v \in V'$).



Exact Algorithm:

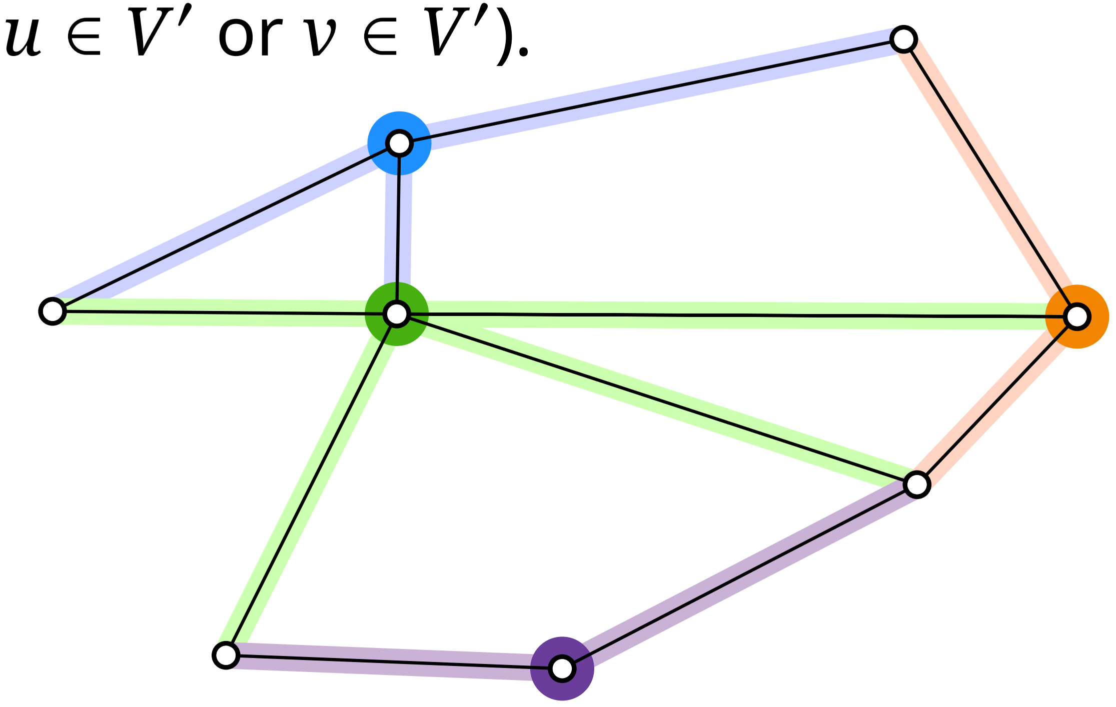
test all possible subsets
in $O(2^n m)$ time

Optimum (OPT $= 4$) – but in general NP-hard to find :-(

VertexCover

Input: Graph $G = (V, E)$

Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is covered (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).



Exact Algorithm:

test all possible subsets in $O(2^n m)$ time

$\longrightarrow$ ILP-formulation

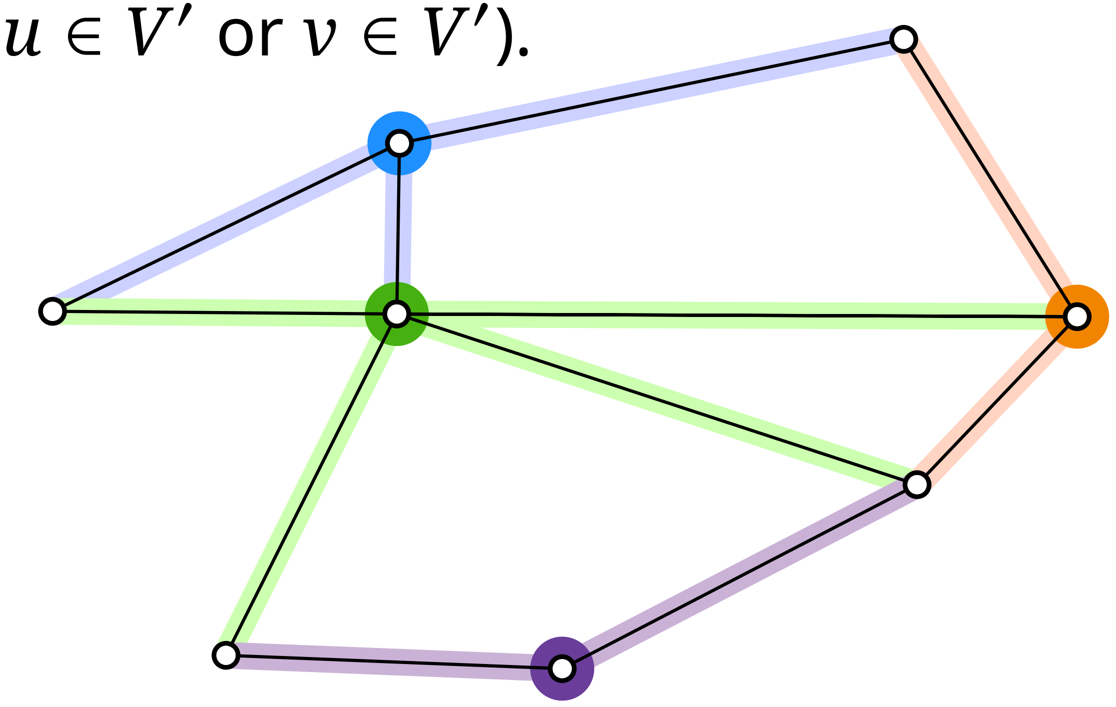
$\longrightarrow$ parameterized in k = size of the vertex cover

Optimum (OPT $= 4$) – but in general NP-hard to find :-(

VertexCover

Input: Graph $G = (V, E)$

Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that
every edge is covered (i.e., for every $uv \in E$, it holds
that $u \in V'$ or $v \in V'$).



Exact Algorithm:

test all possible subsets
in $O(2^n m)$ time

$\longrightarrow$ ILP-formulation

$\longrightarrow$ parameterized in k =
size of the vertex cover

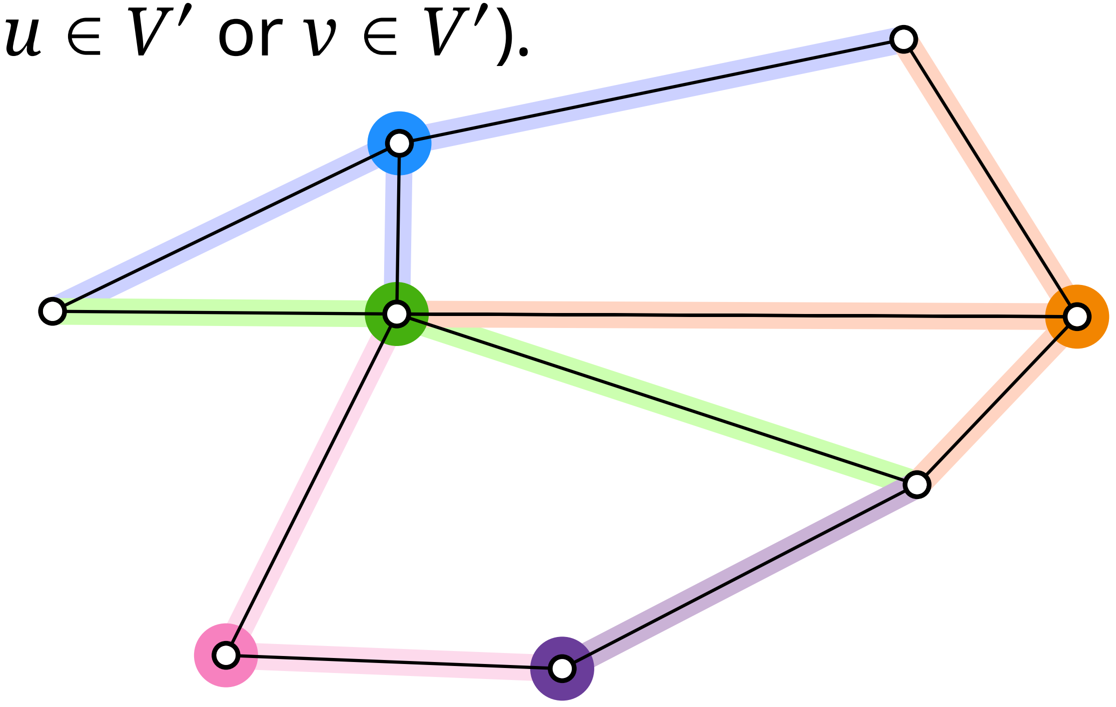
or approximate!

Optimum ($\mathrm{OPT} = 4$) – but in general NP-hard to find :-(

VertexCover

Input: Graph $G = (V, E)$

Output: a minimum vertex cover, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is covered (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).



Exact Algorithm:

test all possible subsets in $O(2^n m)$ time

$\longrightarrow$ ILP-formulation

$\longrightarrow$ parameterized in k = size of the vertex cover

or approximate!

"good" ($5/4$-) approximate solution

Key Concepts: NP-Optimization Problems and Approximation Algorithms

NP-Optimization Problem

An **NP-optimization problem Π** is given by:

NP-Optimization Problem

An **NP-optimization problem** Π is given by:

- A set D_Π of **instances**.
  We denote the size of an instance $I \in D_\Pi$ by $|I|$.

NP-Optimization Problem

An **NP-optimization problem** Π is given by:

- A set D_Π of **instances**.
  We denote the size of an instance $I \in D_\Pi$ by $|I|$.
- For each instance $I \in D_\Pi$, a set $S_\Pi(I) \neq \emptyset$ of **feasible solutions** for I such that:

NP-Optimization Problem

An **NP-optimization problem** Π is given by:

- A set D_Π of **instances**.
  We denote the size of an instance $I \in D_\Pi$ by $|I|$.
- For each instance $I \in D_\Pi$, a set $S_\Pi(I) \neq \emptyset$ of **feasible solutions** for I such that:
  - for each solution $s \in S_\Pi(I)$,
    its size $|s|$ is polynomially bounded in $|I|$, and

NP-Optimization Problem

An **NP-optimization problem** Π is given by:

- A set D_Π of **instances**.
  We denote the size of an instance $I \in D_\Pi$ by $|I|$.
- For each instance $I \in D_\Pi$, a set $S_\Pi(I) \neq \emptyset$ of **feasible solutions** for I such that:
  - for each solution $s \in S_\Pi(I)$,
    its size $|s|$ is polynomially bounded in $|I|$, and
  - there is a polynomial-time algorithm that decides,
    for each pair (s, I), whether $s \in S_\Pi(I)$.

NP-Optimization Problem

An **NP-optimization problem** Π is given by:

- A set D_Π of **instances**.
  We denote the size of an instance $I \in D_\Pi$ by $|I|$.
- For each instance $I \in D_\Pi$, a set $S_\Pi(I) \neq \emptyset$ of **feasible solutions** for I such that:
  - for each solution $s \in S_\Pi(I)$,
    its size $|s|$ is polynomially bounded in $|I|$, and
  - there is a polynomial-time algorithm that decides,
    for each pair (s, I), whether $s \in S_\Pi(I)$.
- A polynomial time computable **objective function** obj_Π
  which assigns a positive objective value $\mathrm{obj}_\Pi(I, s) \geq 0$ to
  any given pair (s, I) with $s \in S_\Pi(I)$.

NP-Optimization Problem

An **NP-optimization problem** Π is given by:

- A set D_Π of **instances**.
  We denote the size of an instance $I \in D_\Pi$ by $|I|$.
- For each instance $I \in D_\Pi$, a set $S_\Pi(I) \neq \emptyset$ of **feasible solutions** for I such that:
  - for each solution $s \in S_\Pi(I)$,
    its size $|s|$ is polynomially bounded in $|I|$, and
  - there is a polynomial-time algorithm that decides,
    for each pair (s, I), whether $s \in S_\Pi(I)$.
- A polynomial time computable **objective function** obj_Π
  which assigns a positive objective value $\mathrm{obj}_\Pi(I, s) \geq 0$ to
  any given pair (s, I) with $s \in S_\Pi(I)$.

- Π is either a minimization or maximization problem.

VERTEXCOVER: NP-Optimization Problem

Task: Fill in the gaps for the problem Π = VERTEX COVER.

D_Π = ???

For $I \in D_\Pi$: ??? $|I|$ = ???

$S_\Pi(I)$ = ???

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?

- For a given pair (s, I), how can we efficiently decide whether $s \in S_\Pi(I)$?

$\text{obj}_\Pi(I, s)$ = ???

Π is a m....imization problem.

VertexCover: NP-Optimization Problem

Task: Fill in the gaps for the problem $\Pi = $ Vertex Cover.

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: ??? $|I| =$???

$S_\Pi(I) =$???

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?

- For a given pair (s, I), how can we efficiently decide whether $s \in S_\Pi(I)$?

$\text{obj}_\Pi(I, s) =$???

Π is a m....imization problem.

VertexCover: NP-Optimization Problem

Task: Fill in the gaps for the problem Π = Vertex Cover.

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: $|I| =$???

$G=(V,E)$ $S_\Pi(I) =$???

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?
- For a given pair (s, I), how can we efficiently decide whether $s \in S_\Pi(I)$?

$\text{obj}_\Pi(I, s) =$???

Π is a m....imization problem.

VertexCover: NP-Optimization Problem

Task: Fill in the gaps for the problem Π = Vertex Cover.

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: $|I| = |V|+|E|$

$G=(V,E)$

$S_\Pi(I) =$???

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?
- For a given pair (s, I), how can we efficiently decide whether $s \in S_\Pi(I)$?

$\text{obj}_\Pi(I, s) =$???

Π is a m....imization problem.

VERTEXCOVER: NP-Optimization Problem

Task: Fill in the gaps for the problem Π = VERTEX COVER.

D_Π = Set of all graphs

For $I \in D_\Pi$: $|I|$ = $|V|+|E|$

$G=(V,E)$ $S_\Pi(I)$ = Set of all vertex covers of G

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?
- For a given pair (s, I), how can we efficiently decide whether $s \in S_\Pi(I)$?

$\text{obj}_\Pi(I, s)$ = ???

Π is a m....imization problem.

$\textsc{VertexCover}$: NP-Optimization Problem

Task: Fill in the gaps for the problem $\Pi = \textsc{Vertex Cover}$.

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: $|I| =$ $|V|+|E|$

$G=(V,E)$ $S_\Pi(I) =$ Set of all vertex covers of G

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?

  $s \subseteq V \Rightarrow |s| \leq |V| \leq |I|$

- For a given pair (s, I), how can we efficiently decide whether $s \in S_\Pi(I)$?

$\text{obj}_\Pi(I, s) =$???

Π is a m....imization problem.

VERTEXCOVER: NP-Optimization Problem

Task: Fill in the gaps for the problem Π = VERTEX COVER.

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: $|I| =$ $|V|+|E|$

$G=(V,E)$ $S_\Pi(I) =$ Set of all vertex covers of G

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?

  $s \subseteq V \Rightarrow |s| \leq |V| \leq |I|$

- For a given pair (s, I), how can we efficiently decide whether $s \in S_\Pi(I)$? Test whether all edges are covered.

$\text{obj}_\Pi(I, s) =$???

Π is a m....imization problem.

VertexCover: NP-Optimization Problem

Task: Fill in the gaps for the problem Π = Vertex Cover.

D_Π = Set of all graphs

For $I \in D_\Pi$: $|I|$ = $|V|+|E|$

$G=(V,E)$ $S_\Pi(I)$ = Set of all vertex covers of G

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?

  $s \subseteq V \Rightarrow |s| \leq |V| \leq |I|$

- For a given pair (s, I), how can we efficiently decide whether $s \in S_\Pi(I)$? Test whether all edges are covered.

$\text{obj}_\Pi(I, s)$ = $|s|$

Π is a m....imization problem.

VertexCover: NP-Optimization Problem

Task: Fill in the gaps for the problem Π = Vertex Cover.

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: $\quad |I| =$ $|V|+|E|$

$G=(V,E)$ $\quad S_\Pi(I) =$ Set of all vertex covers of G

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?

  $s \subseteq V \Rightarrow |s| \leq |V| \leq |I|$

- For a given pair (s, I), how can we efficiently decide whether $s \in S_\Pi(I)$? Test whether all edges are covered.

$\text{obj}_\Pi(I, s) =$ $|s|$

Π is a minimization problem.

Optimum and Optimal Objective Value

Let Π be a minimization problem and $I \in D_\Pi$ an instance of Π.

Optimum and Optimal Objective Value

Let Π be a minimization problem and $I \in D_\Pi$ an instance of Π.

A feasible solution $s^* \in S_\Pi(I)$ is optimal if

$\mathrm{obj}_\Pi(I, s^*)$ is minimal among the objective values

attained by the feasible solutions of I.

Optimum and Optimal Objective Value

maximization problem

Let Π be a minimization problem and $I \in D_\Pi$ an instance of Π.

A feasible solution $s^* \in S_\Pi(I)$ is optimal if

maximal

$\mathrm{obj}_\Pi(I, s^*)$ is minimal among the objective values

attained by the feasible solutions of I.

Optimum and Optimal Objective Value

maximization problem

Let Π be a minimization problem and $I \in D_\Pi$ an instance of Π.

A feasible solution $s^* \in S_\Pi(I)$ is optimal if

maximal

$\mathrm{obj}_\Pi(I, s^*)$ is minimal among the objective values

attained by the feasible solutions of I.

The optimal value $\mathrm{obj}_\Pi(I, s^*)$ of the objective function is denoted by $\mathrm{OPT}_\Pi(I)$ or simply by OPT in context.

Approximation Algorithms

Let Π be a minimization problem and $\alpha \in \mathbb{Q}^+$.

Approximation Algorithms

Let Π be a minimization problem and $\alpha \in \mathbb{Q}^{+}$.

A factor-α approximation algorithm for Π is an efficient algorithm that provides, for **any** instance $I \in D_{\Pi}$, a feasible solution $s \in S_{\Pi}(I)$ such that

Approximation Algorithms

Let Π be a minimization problem and $\alpha \in \mathbb{Q}^+$.

A factor-α approximation algorithm for Π is an efficient algorithm that provides, for **any** instance $I \in D_\Pi$, a feasible solution $s \in S_\Pi(I)$ such that

$$\frac{\mathrm{obj}_\Pi(I, s)}{\mathrm{OPT}_\Pi(I)}$$

Approximation Algorithms

Let Π be a minimization problem and $\alpha \in \mathbb{Q}^+$.

A factor-α approximation algorithm for Π is an efficient algorithm that provides, for **any** instance $I \in D_\Pi$, a feasible solution $s \in S_\Pi(I)$ such that

$$\frac{\mathrm{obj}_\Pi(I, s)}{\mathrm{OPT}_\Pi(I)} \leq \alpha.$$

Approximation Algorithms

$\alpha\colon \mathbb{N} \to \mathbb{Q}$

Let Π be a minimization problem and $\cancel{\alpha \in \mathbb{Q}^+}$.

A factor-α approximation algorithm for Π is an efficient algorithm that provides, for **any** instance $I \in D_\Pi$, a feasible solution $s \in S_\Pi(I)$ such that

$$\frac{\mathrm{obj}_\Pi(I, s)}{\mathrm{OPT}_\Pi(I)} \leq \cancel{\alpha}. \quad \alpha(|I|)$$

Approximation Algorithms

maximization problem $\alpha\colon \mathbb{N} \to \mathbb{Q}$

Let Π be a minimization problem and $\cancel{\alpha \in \mathbb{Q}^+}$.

A factor-α approximation algorithm for Π is an efficient algorithm that provides, for **any** instance $I \in D_\Pi$, a feasible solution $s \in S_\Pi(I)$ such that

$$\frac{\mathrm{obj}_\Pi(I, s)}{\mathrm{OPT}_\Pi(I)} \overset{\geq}{\leq} \cancel{\alpha}. \quad \alpha(|I|)$$

Approximation Algorithm for VertexCover

Approximation Algorithm for VERTEXCOVER

Ideas?

Approximation Algorithm for VertexCover

Ideas?

- Edge-Greedy

Approximation Algorithm for VERTEXCOVER

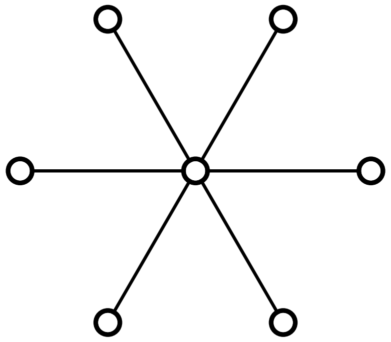
Ideas?

- Edge-Greedy
- Vertex-Greedy

Approximation Algorithm for VERTEXCOVER

Ideas?

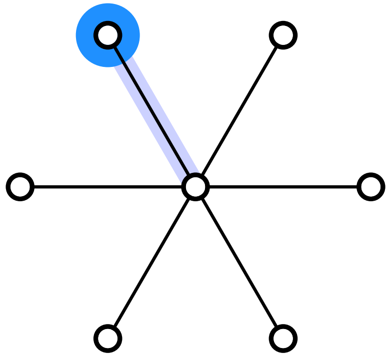
- Edge-Greedy
- Vertex-Greedy



Approximation Algorithm for VERTEXCOVER

Ideas?

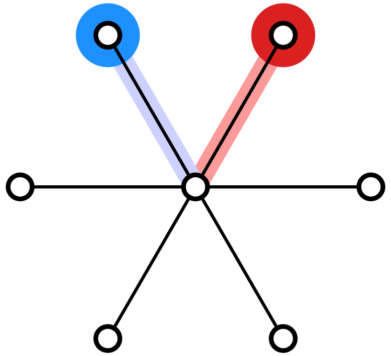
- Edge-Greedy
- Vertex-Greedy



Approximation Algorithm for VERTEXCOVER

Ideas?

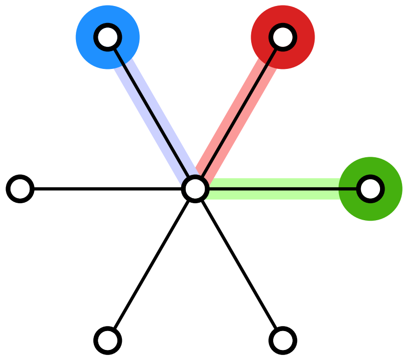
- Edge-Greedy
- Vertex-Greedy



Approximation Algorithm for VERTEXCOVER

Ideas?

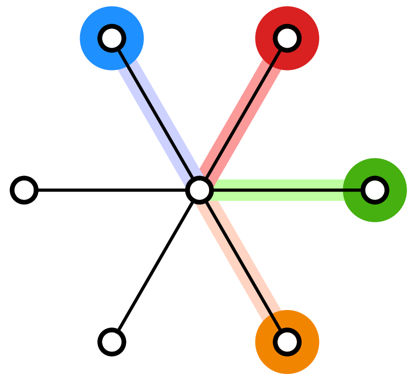
- Edge-Greedy
- Vertex-Greedy



Approximation Algorithm for VERTEXCOVER

Ideas?

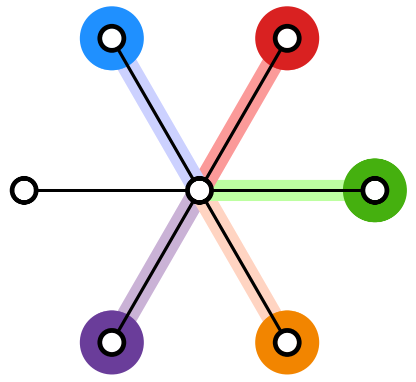
- Edge-Greedy
- Vertex-Greedy



Approximation Algorithm for VERTEXCOVER

Ideas?

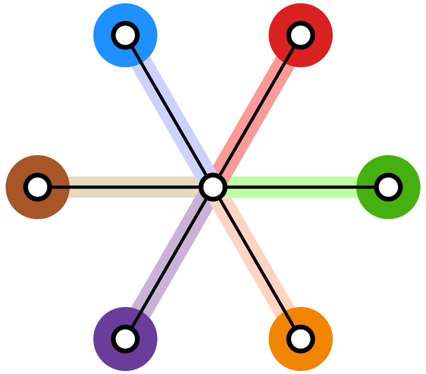
- Edge-Greedy
- Vertex-Greedy



Approximation Algorithm for VertexCover

Ideas?

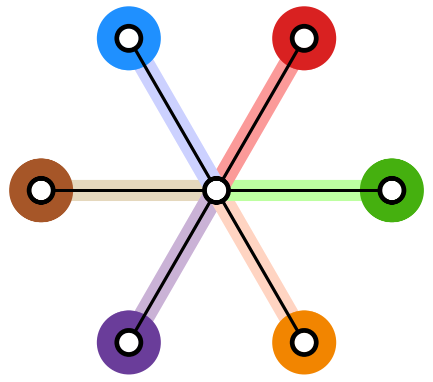
- Edge-Greedy
- Vertex-Greedy



Approximation Algorithm for VERTEXCOVER

Ideas?

- Edge-Greedy
- Vertex-Greedy

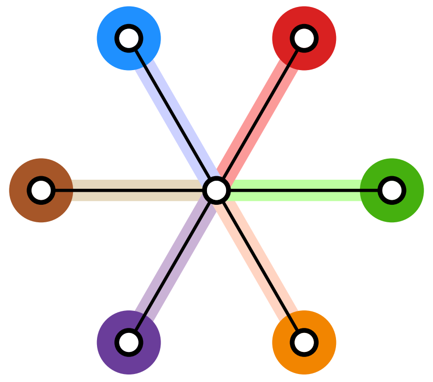


Quality?

Approximation Algorithm for VERTEXCOVER

Ideas?

- Edge-Greedy
- Vertex-Greedy



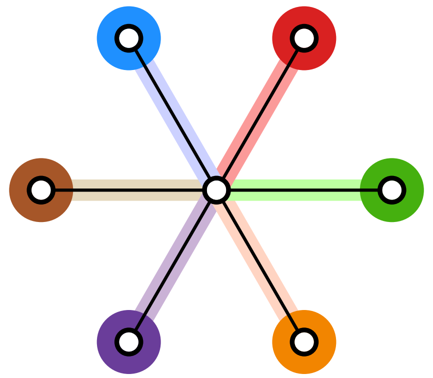
Quality?

Problem: How can we estimate $\mathrm{obj}_{\Pi}(I, s)/\mathrm{OPT}$, when it is hard to compute OPT?

Approximation Algorithm for VertexCover

Ideas?

- Edge-Greedy
- Vertex-Greedy



Quality?

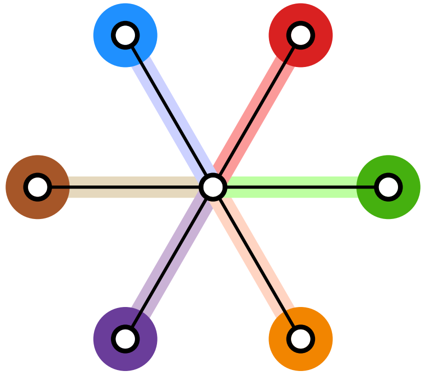
Problem: How can we estimate $\mathrm{obj}_\Pi(I, s)/\mathrm{OPT}$, when it is hard to compute OPT?

Idea: Find a "good" lower bound $L \leq \mathrm{OPT}$ for OPT and compare it to our approximate solution.

Approximation Algorithm for VERTEXCOVER

Ideas?

- Edge-Greedy
- Vertex-Greedy



Quality?

Problem: How can we estimate $\mathrm{obj}_\Pi(I, s)/\mathrm{OPT}$, when it is hard to compute OPT?

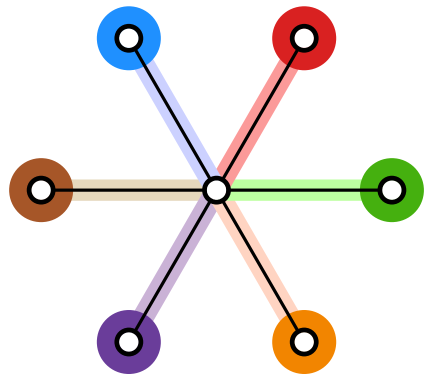
Idea: Find a "good" lower bound $L \leq \mathrm{OPT}$ for OPT and compare it to our approximate solution.

$$\frac{\mathrm{obj}_\Pi(I, s)}{\mathrm{OPT}} \leq \frac{\mathrm{obj}_\Pi(I, s)}{L}$$

Approximation Algorithm for VertexCover

Ideas?

- Edge-Greedy
- Vertex-Greedy



Quality?

Problem: How can we estimate $\text{obj}_\Pi(I, s)/\text{OPT}$, when it is hard to compute OPT?

Idea: Find a “good” lower bound $L \leq \text{OPT}$ for OPT and compare it to our approximate solution.

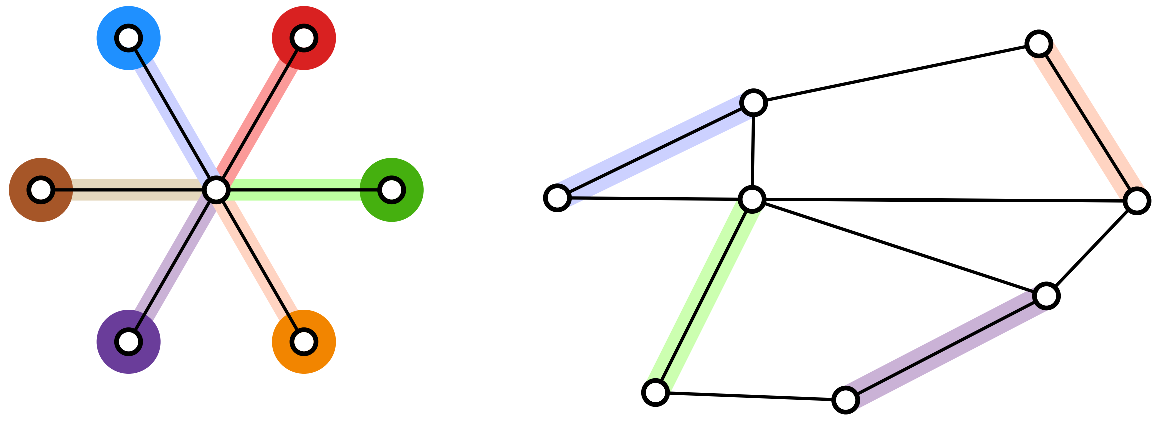
$$\frac{\text{obj}_\Pi(I, s)}{\text{OPT}} \leq \frac{\text{obj}_\Pi(I, s)}{L}$$

Q: how can we lower bound the size of a vertex cover?

Approximation Algorithm for VertexCover

Ideas?

- Edge-Greedy
- Vertex-Greedy



Quality?

Problem: How can we estimate $\mathrm{obj}_\Pi(I, s)/\mathrm{OPT}$, when it is hard to compute OPT?

Idea: Find a "good" lower bound $L \leq \mathrm{OPT}$ for OPT and compare it to our approximate solution.

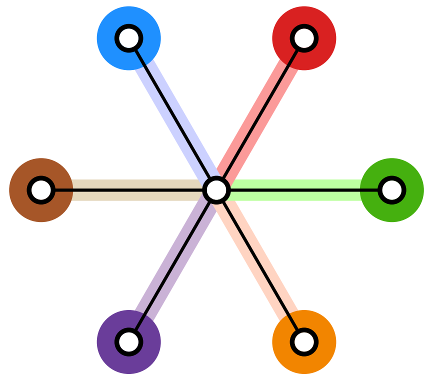
$$\frac{\mathrm{obj}_\Pi(I, s)}{\mathrm{OPT}} \leq \frac{\mathrm{obj}_\Pi(I, s)}{L}$$

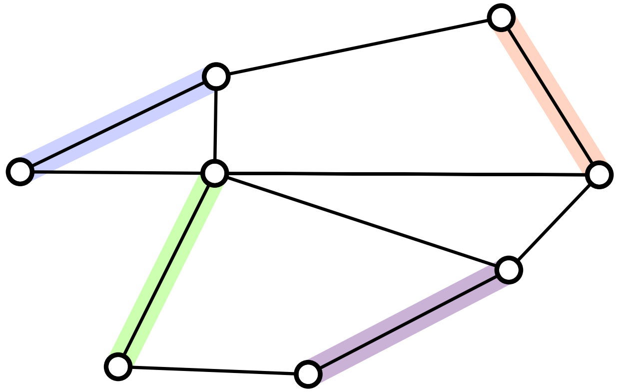
Q: how can we lower bound the size of a vertex cover?

Approximation Algorithm for VertexCover

Ideas?

- Edge-Greedy
- Vertex-Greedy





How did we argue that $OPT = 4$ for this instance?

Quality?

Problem: How can we estimate $\mathrm{obj}_\Pi(I, s)/\mathrm{OPT}$, when it is hard to compute OPT?

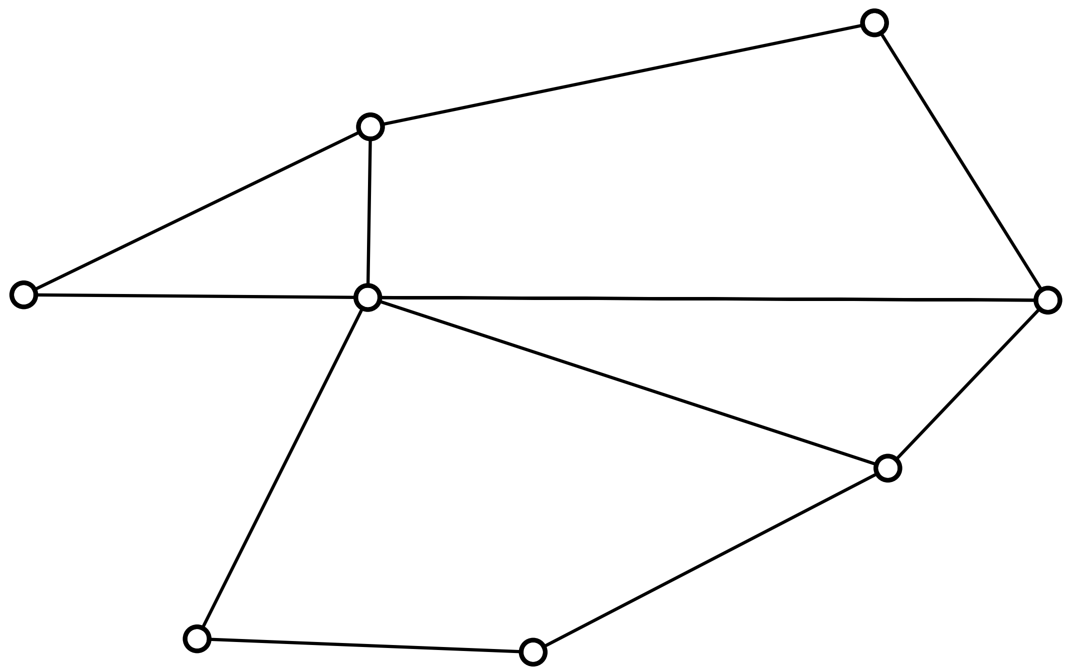
Idea: Find a “good” lower bound $L \leq \mathrm{OPT}$ for OPT and compare it to our approximate solution.

$$\frac{\mathrm{obj}_\Pi(I, s)}{\mathrm{OPT}} \leq \frac{\mathrm{obj}_\Pi(I, s)}{L}$$

Q: how can we lower bound the size of a vertex cover?

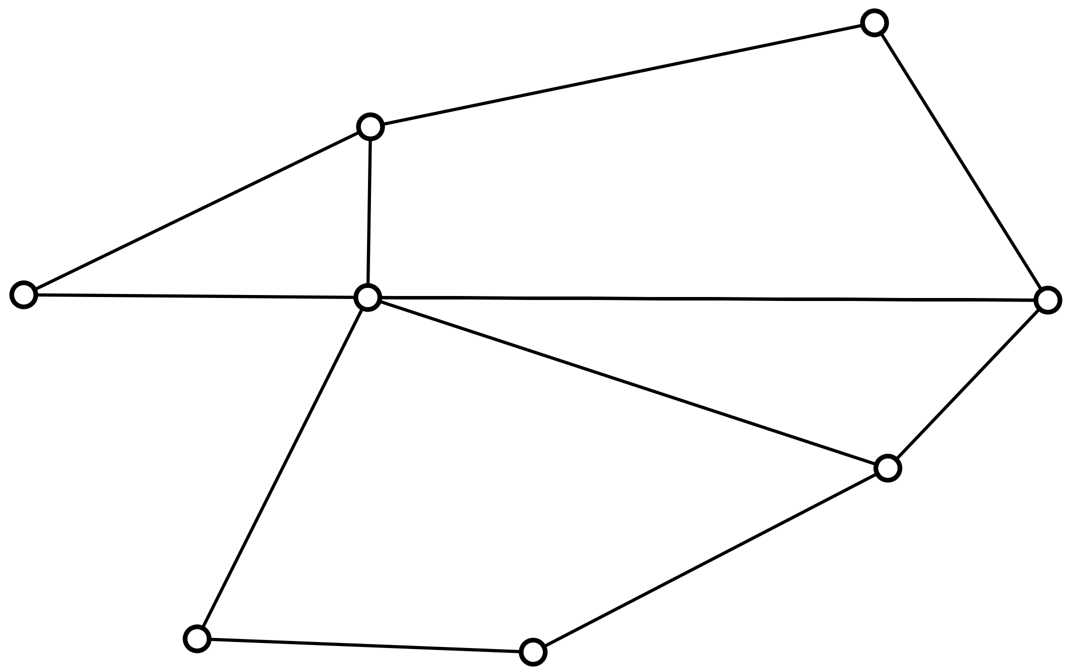
need at least one vertex for each edge in a vertex-disjoint set of edges

Lower Bound by Matchings



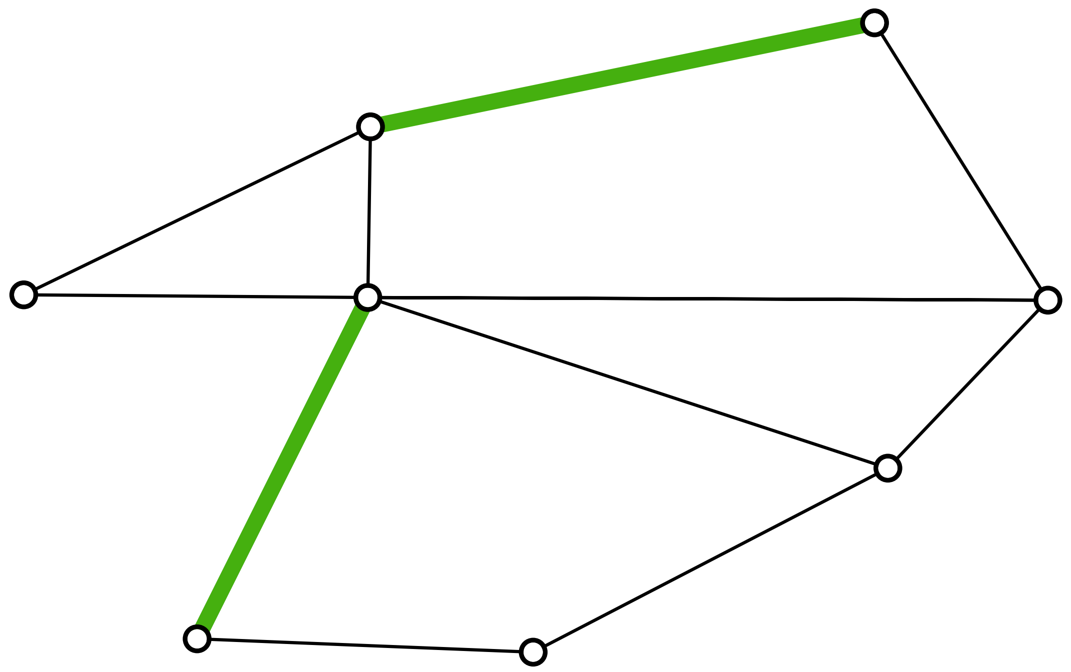
Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a matching if no two edges of M are adjacent (i.e., share an end vertex).



Lower Bound by Matchings

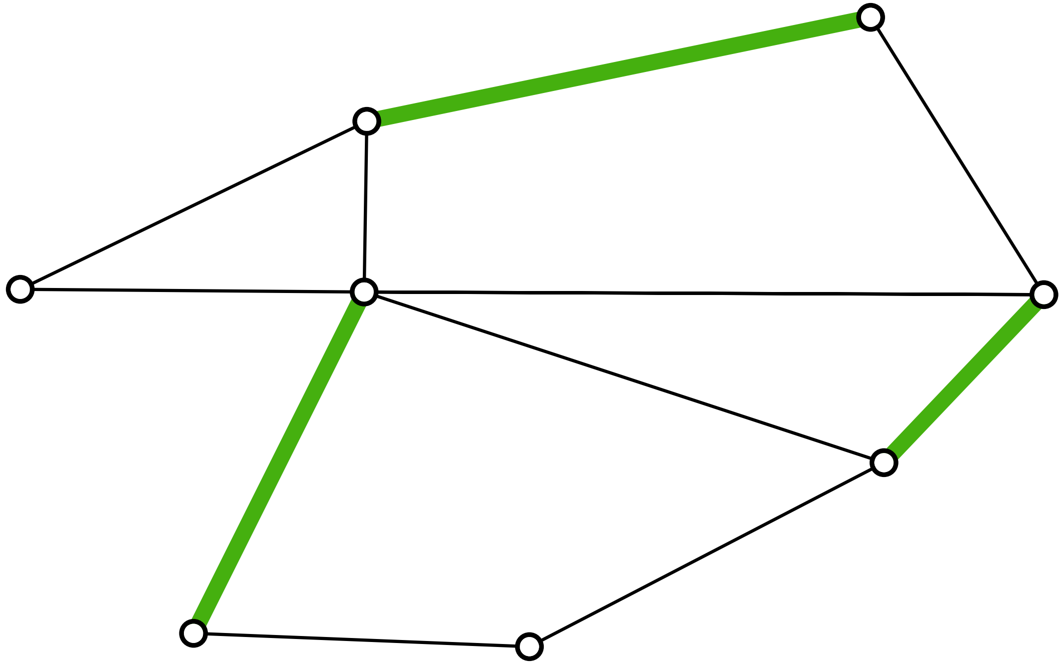
An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a matching if no two edges of M are adjacent (i.e., share an end vertex).



Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a matching if no two edges of M are adjacent (i.e., share an end vertex).

M is maximal if there is no matching M' with $M' \supsetneq M$.

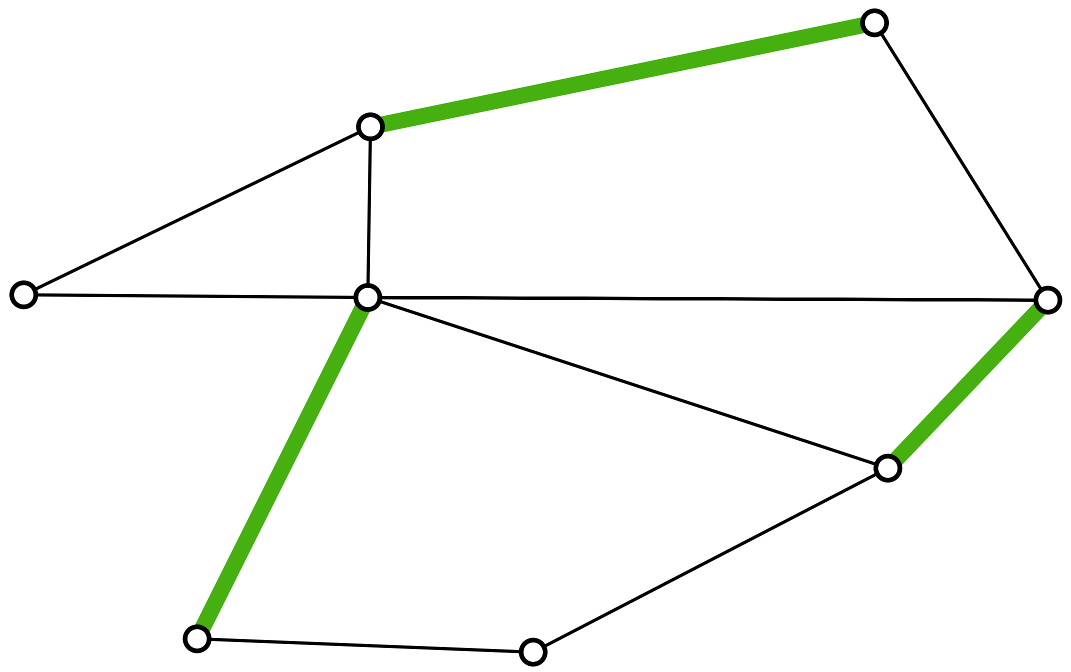


Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a matching if no two edges of M are adjacent (i.e., share an end vertex).

M is maximal if there is no matching M' with $M' \supsetneq M$.

$\mathrm{OPT} \geq |M|$



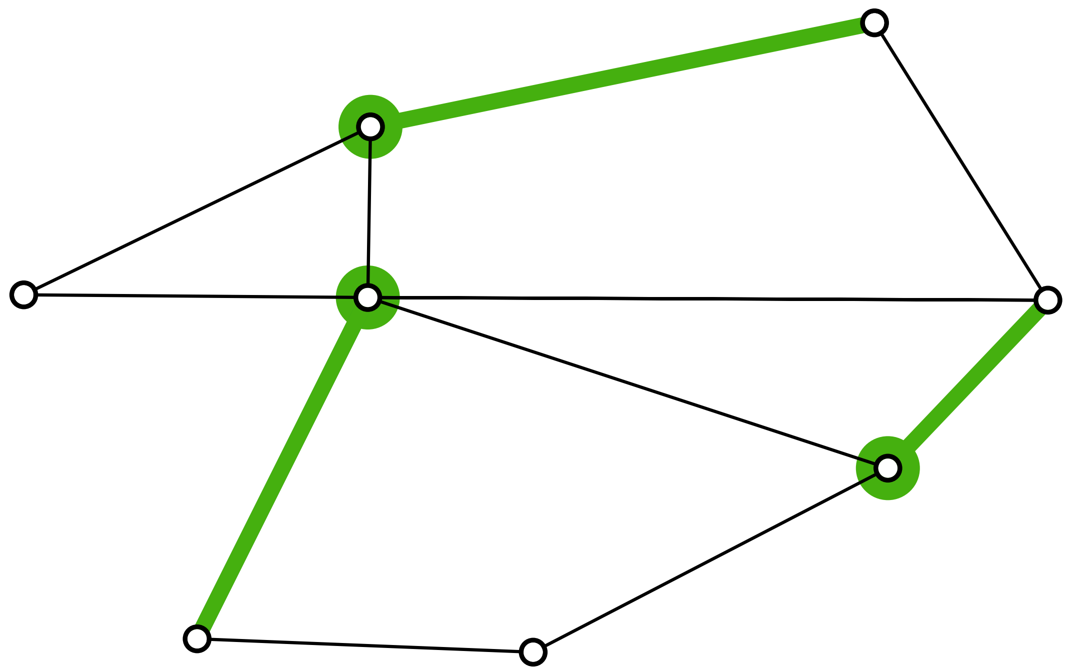
Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a matching if no two edges of M are adjacent (i.e., share an end vertex).

M is maximal if there is no matching M' with $M' \supsetneq M$.

$\mathrm{OPT} \geq |M|$

Vertex cover of M



Lower Bound by Matchings

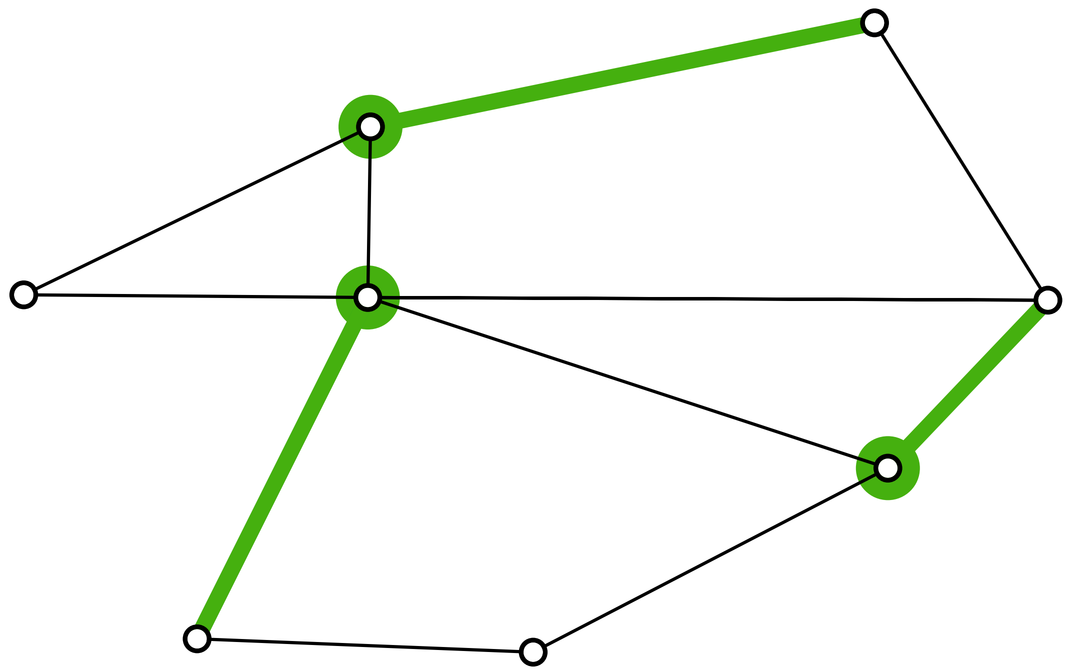
An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a matching if no two edges of M are adjacent (i.e., share an end vertex).

M is maximal if there is no matching M' with $M' \supsetneq M$.

$\mathrm{OPT} \geq |M|$

$\mathrm{OPT} = |M| \quad ?$

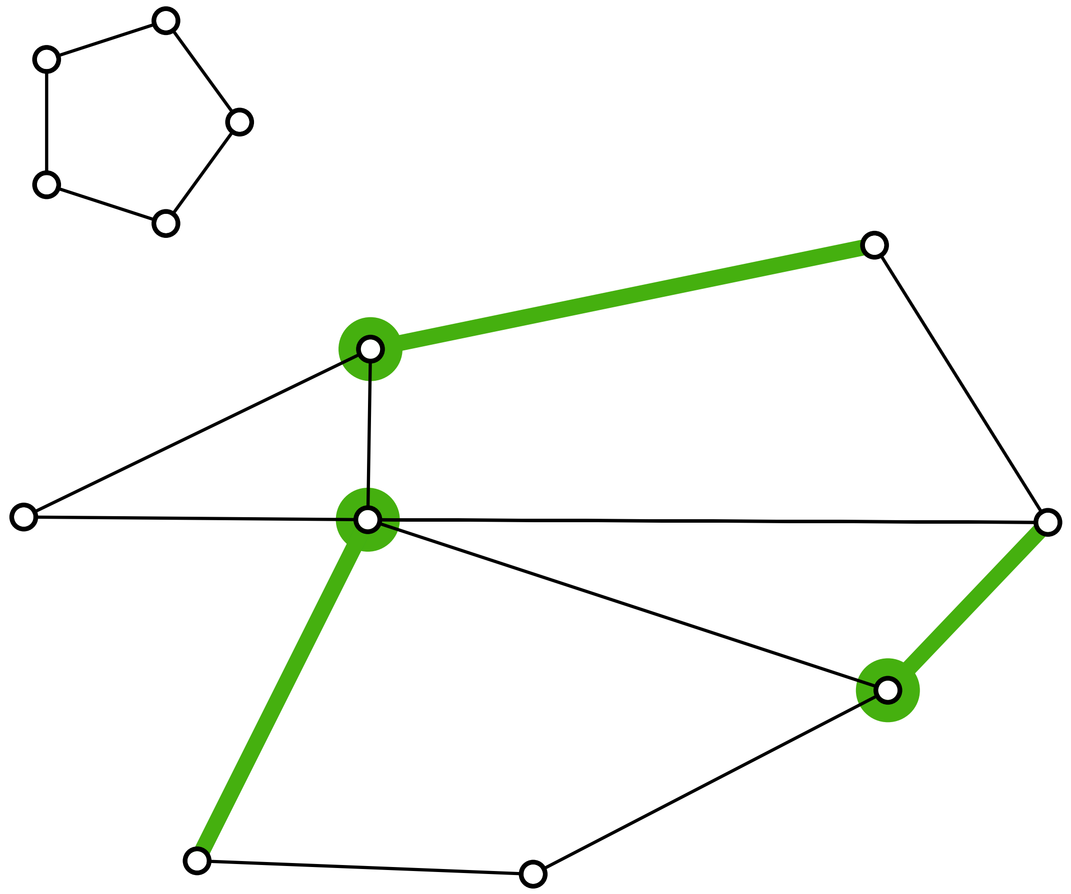
Vertex cover of M



Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a matching if no two edges of M are adjacent (i.e., share an end vertex).

M is maximal if there is no matching M' with $M' \supsetneq M$.



$\mathrm{OPT} \geq |M|$

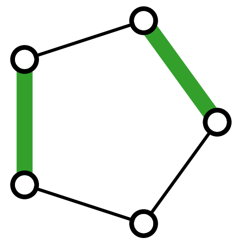
$\mathrm{OPT} = |M|$?

Vertex cover of M

Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a matching if no two edges of M are adjacent (i.e., share an end vertex).

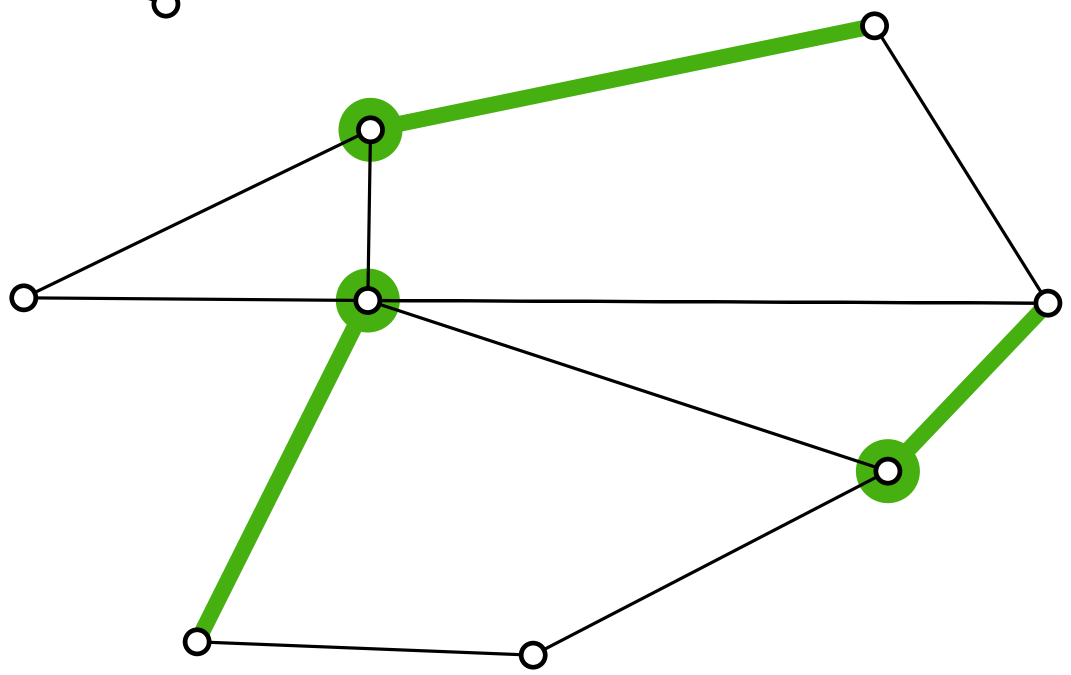
M is maximal if there is no matching M' with $M' \supsetneq M$.



$\mathrm{OPT} \geq |M|$

$\mathrm{OPT} = |M|$?

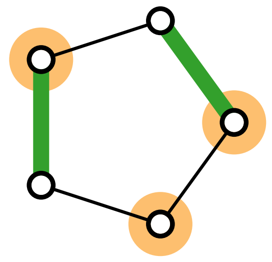
Vertex cover of M



Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a matching if no two edges of M are adjacent (i.e., share an end vertex).

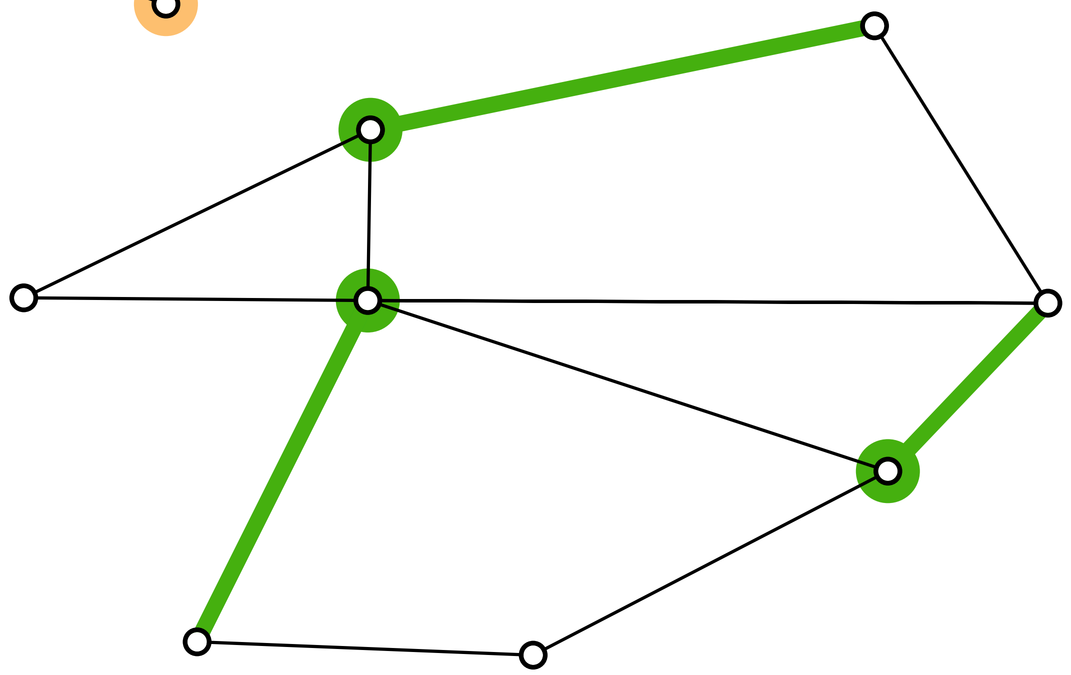
M is maximal if there is no matching M' with $M' \supsetneq M$.



$\mathrm{OPT} \geq |M|$

$\mathrm{OPT} = |M|$?

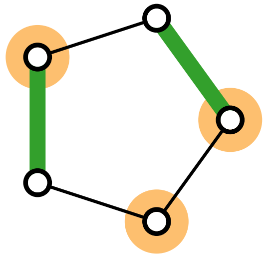
Vertex cover of M



Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a matching if no two edges of M are adjacent (i.e., share an end vertex).

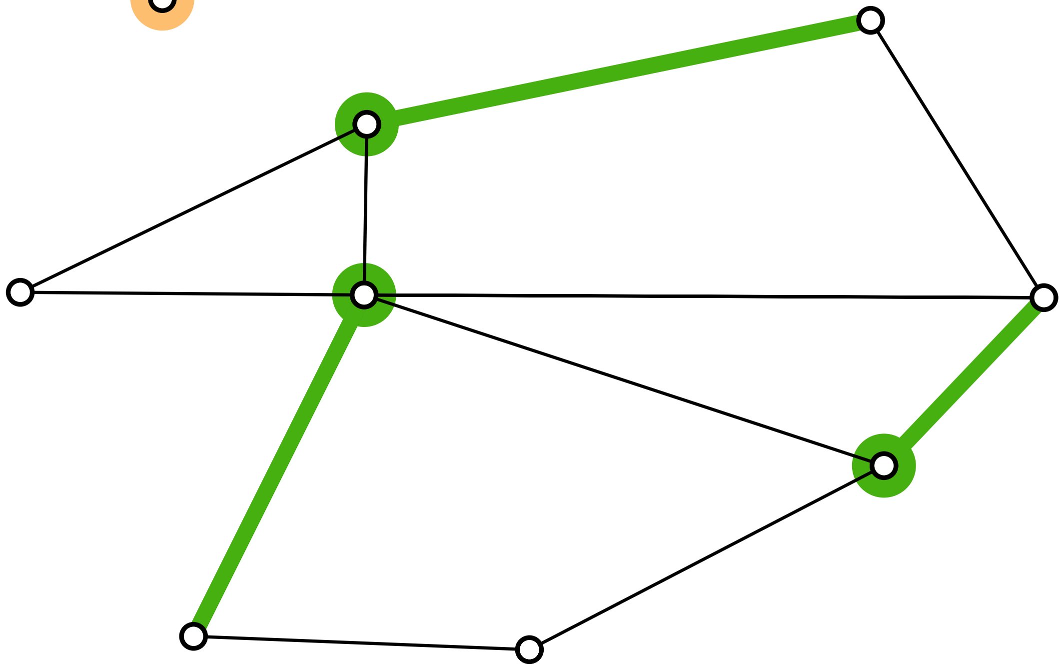
M is maximal if there is no matching M' with $M' \supsetneq M$.



$\mathrm{OPT} \geq |M|$

$\cancel{\mathrm{OPT} = |M| \quad ?}$

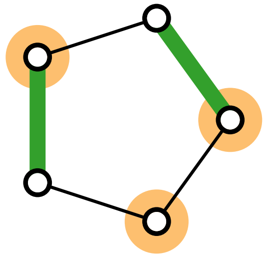
Vertex cover of M



Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a matching if no two edges of M are adjacent (i.e., share an end vertex).

M is maximal if there is no matching M' with $M' \supsetneq M$.

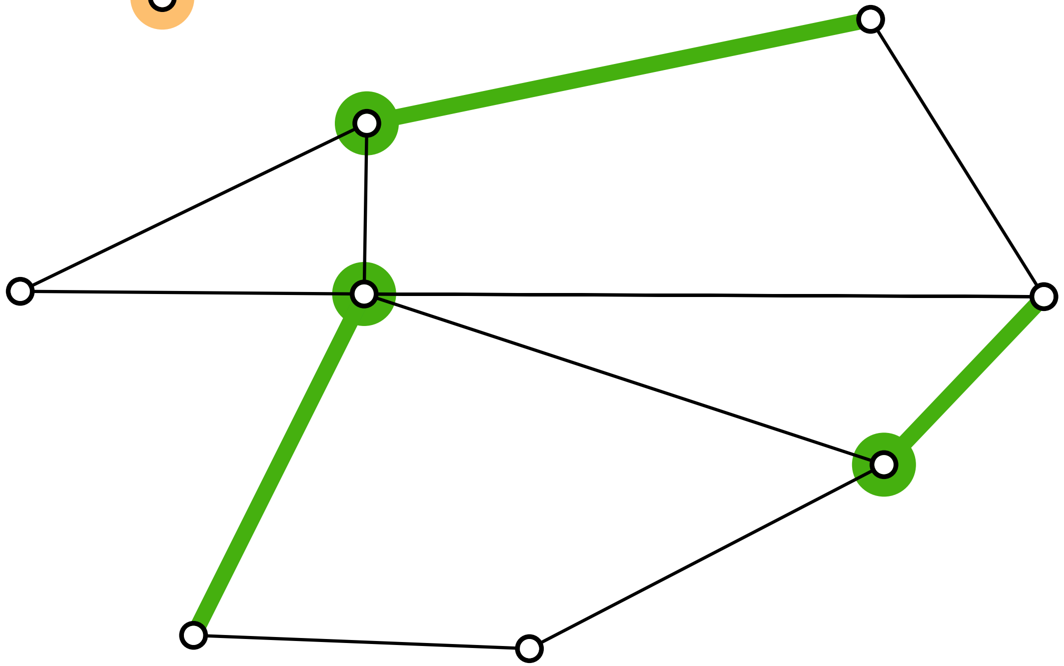


Can you come up with a vertex cover based on M?

$\mathrm{OPT} \geq |M|$

$\cancel{\mathrm{OPT} = |M| \quad ?}$

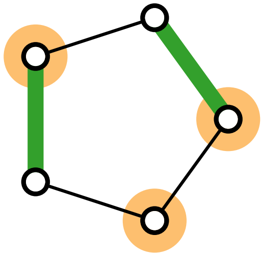
Vertex cover of M



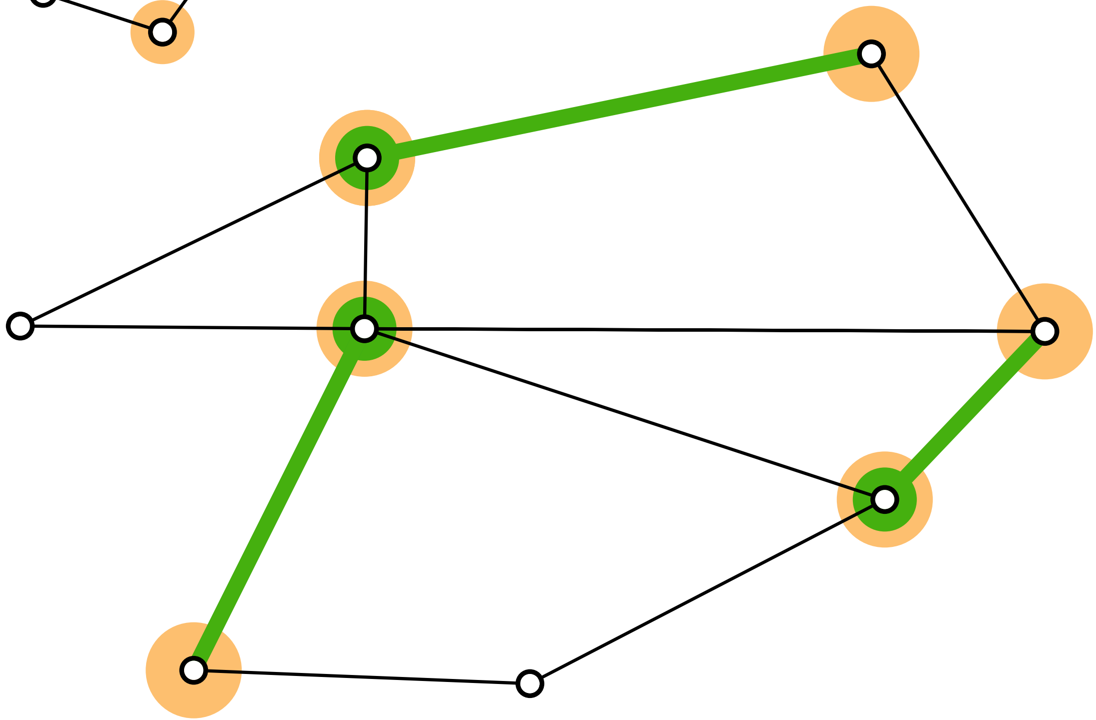
Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a matching if no two edges of M are adjacent (i.e., share an end vertex).

M is maximal if there is no matching M' with $M' \supsetneq M$.



Can you come up with a vertex cover based on M?



$\mathrm{OPT} \geq |M|$

~~$\mathrm{OPT} = |M|$?~~

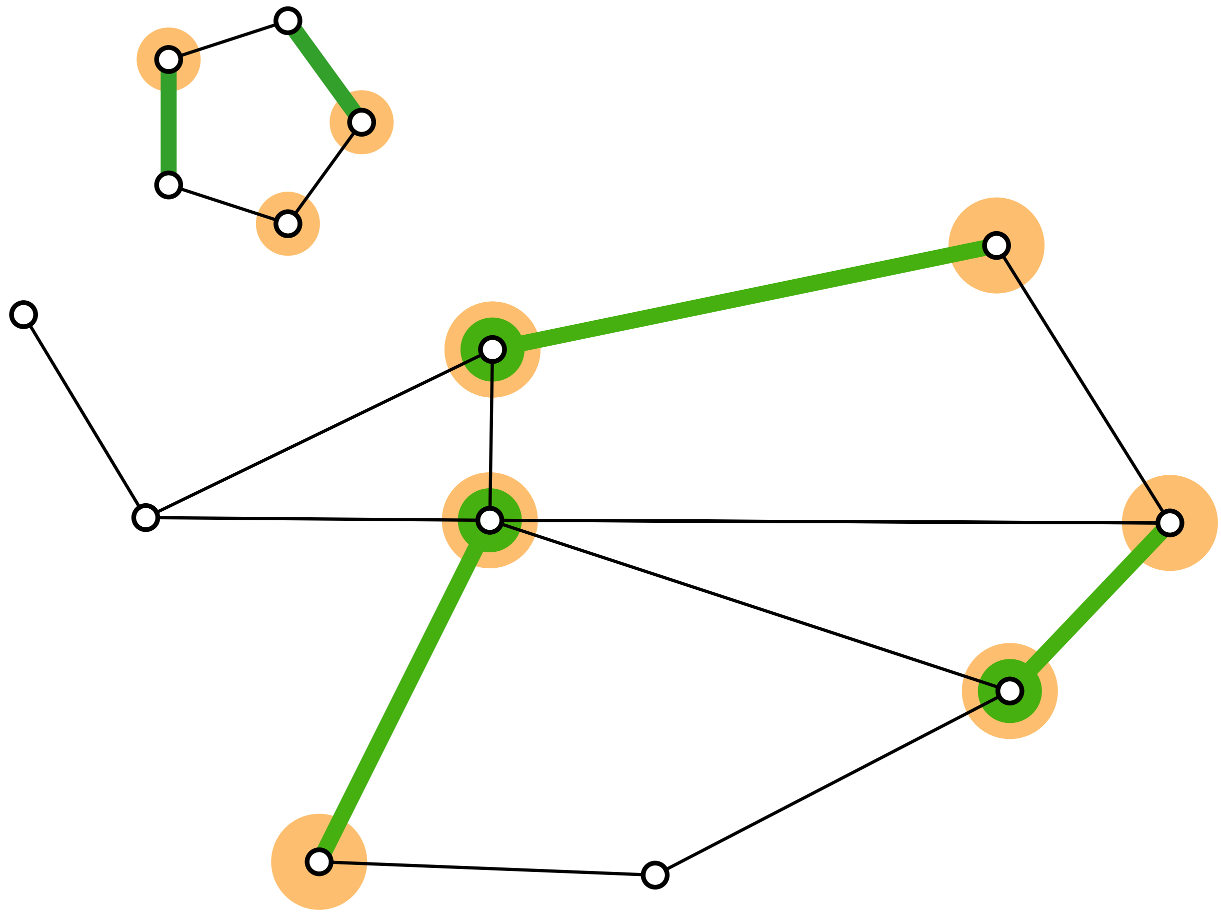
Vertex cover of M

Vertex cover of E

Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a matching if no two edges of M are adjacent (i.e., share an end vertex).

M is maximal if there is no matching M' with $M' \supsetneq M$.



$\mathrm{OPT} \geq |M|$

~~$\mathrm{OPT} = |M|$?~~

Vertex cover of M

Vertex cover of E

Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a matching if no two edges of M are adjacent (i.e., share an end vertex).

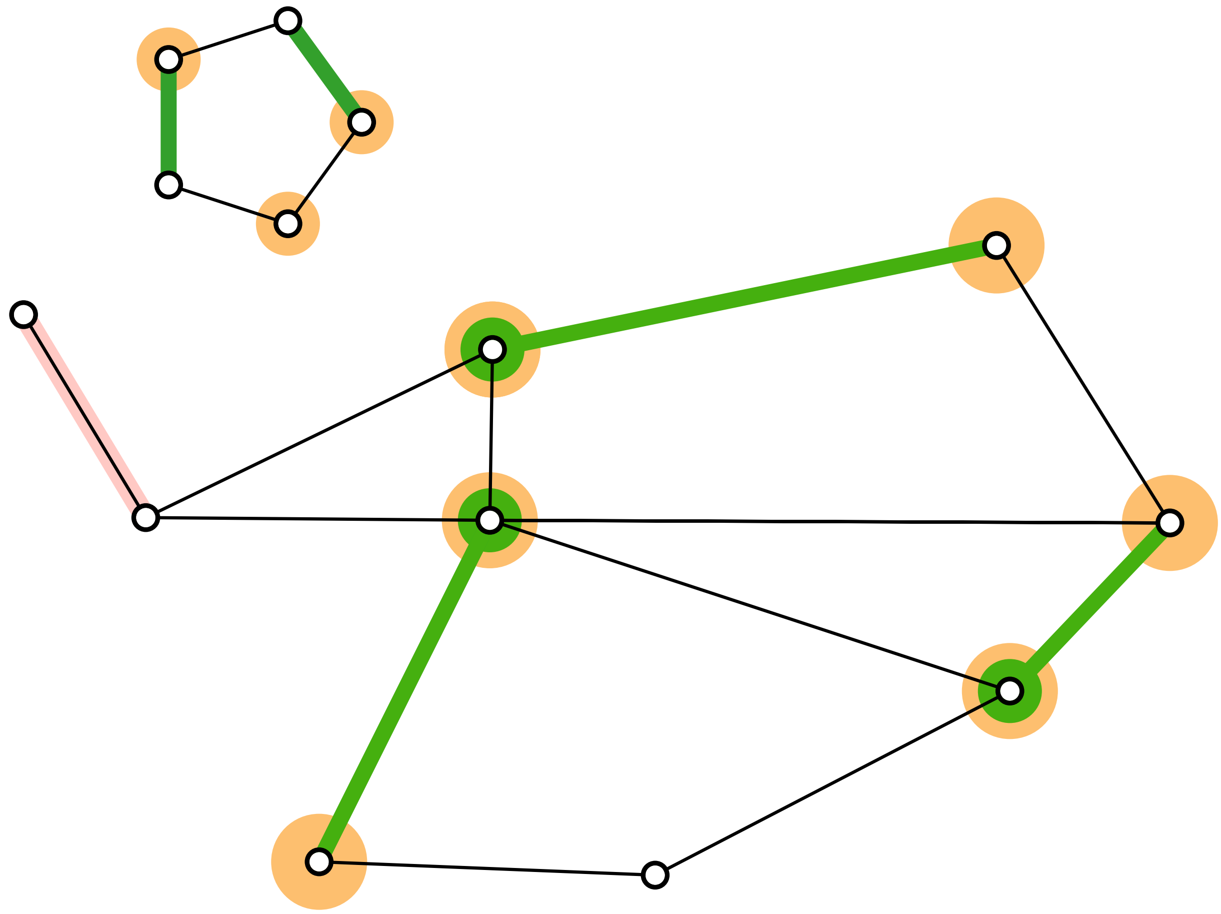
M is maximal if there is no matching M' with $M' \supsetneq M$.

$\mathrm{OPT} \geq |M|$

~~$\mathrm{OPT} = |M|$?~~

Vertex cover of M

Vertex cover of E



Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a matching if no two edges of M are adjacent (i.e., share an end vertex).

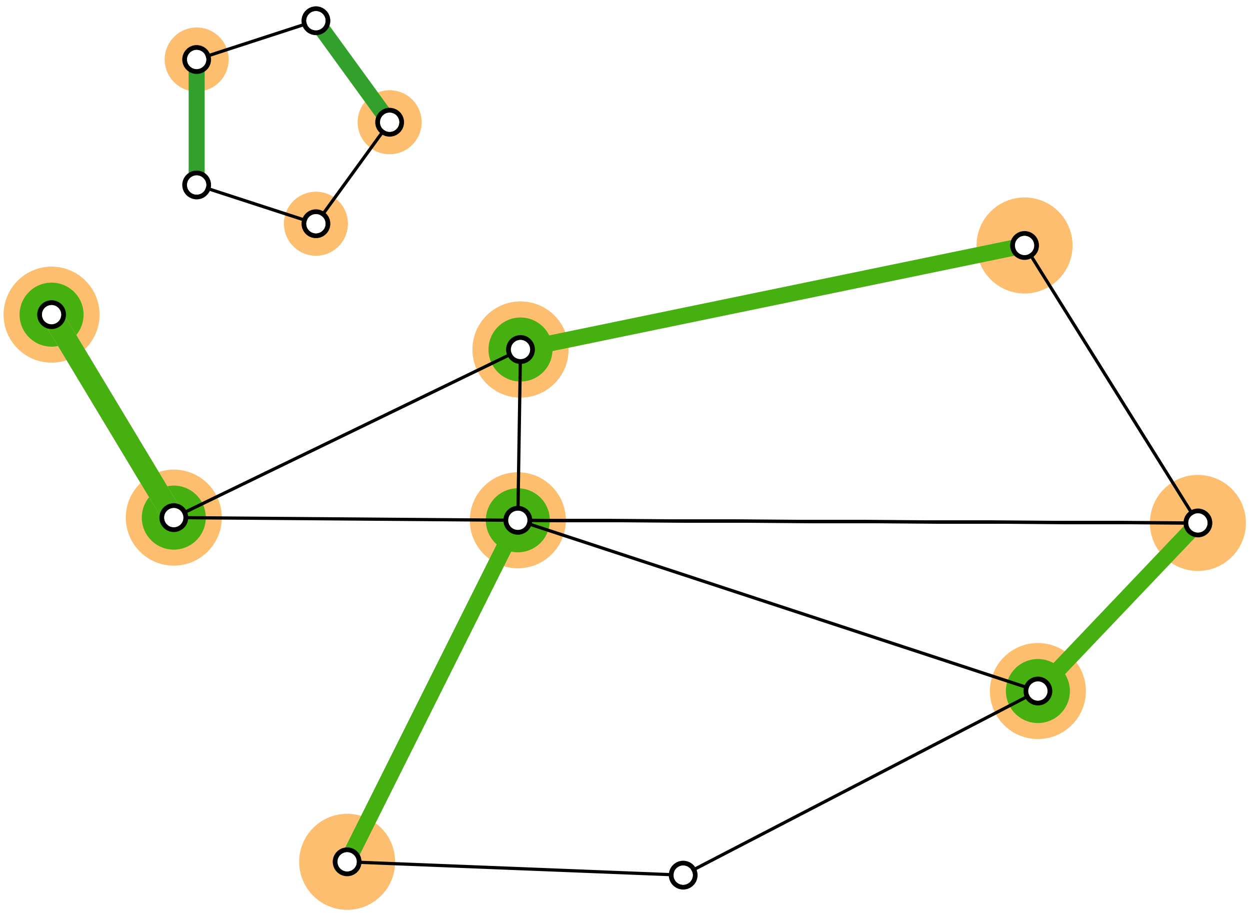
M is maximal if there is no matching M' with $M' \supsetneq M$.

$\mathrm{OPT} \geq |M|$

~~$\mathrm{OPT} = |M|$?~~

Vertex cover of M

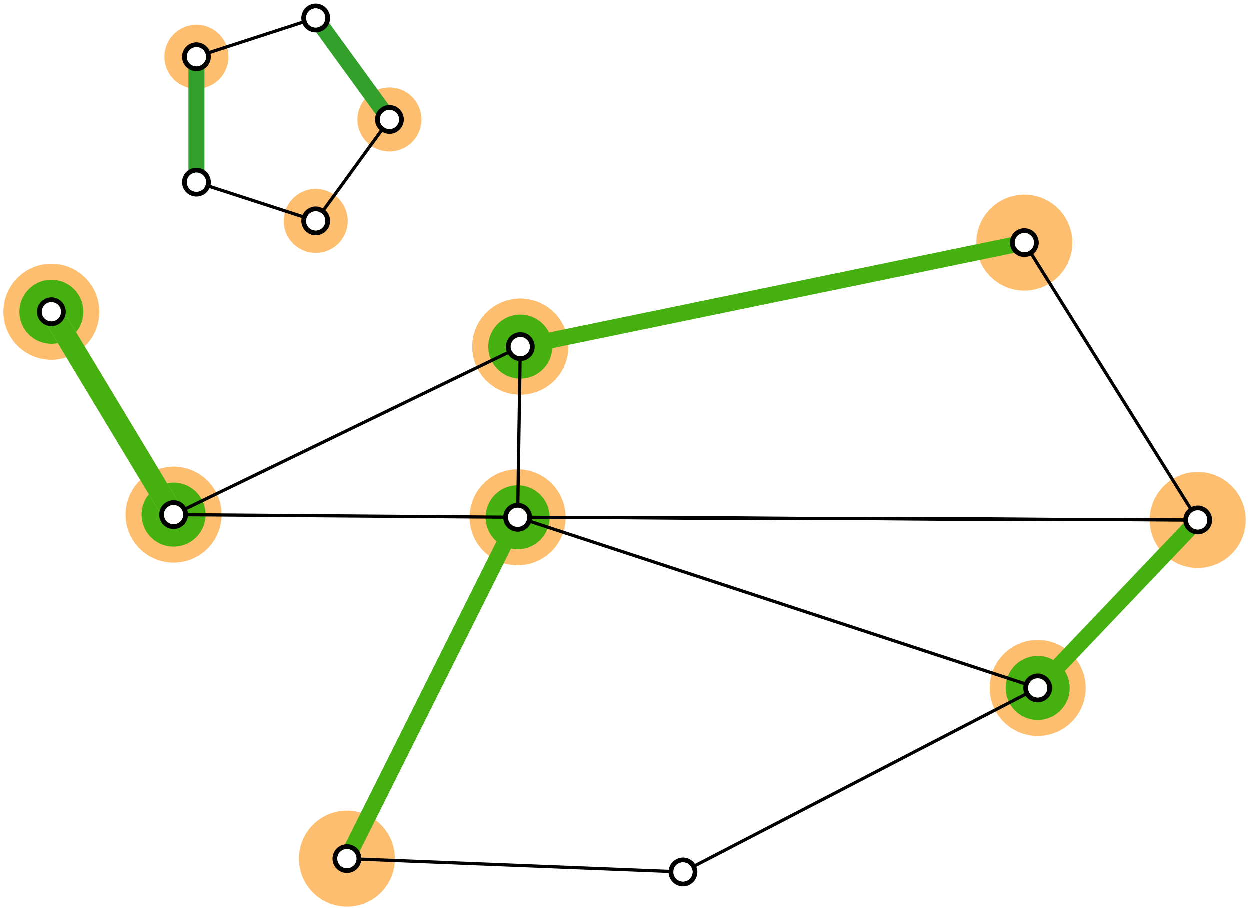
Vertex cover of E



Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a matching if no two edges of M are adjacent (i.e., share an end vertex).

M is maximal if there is no matching M' with $M' \supsetneq M$.



$\mathrm{OPT} \geq |M|$

$\cancel{\mathrm{OPT} = |M| \quad ?}$

Vertex cover of M

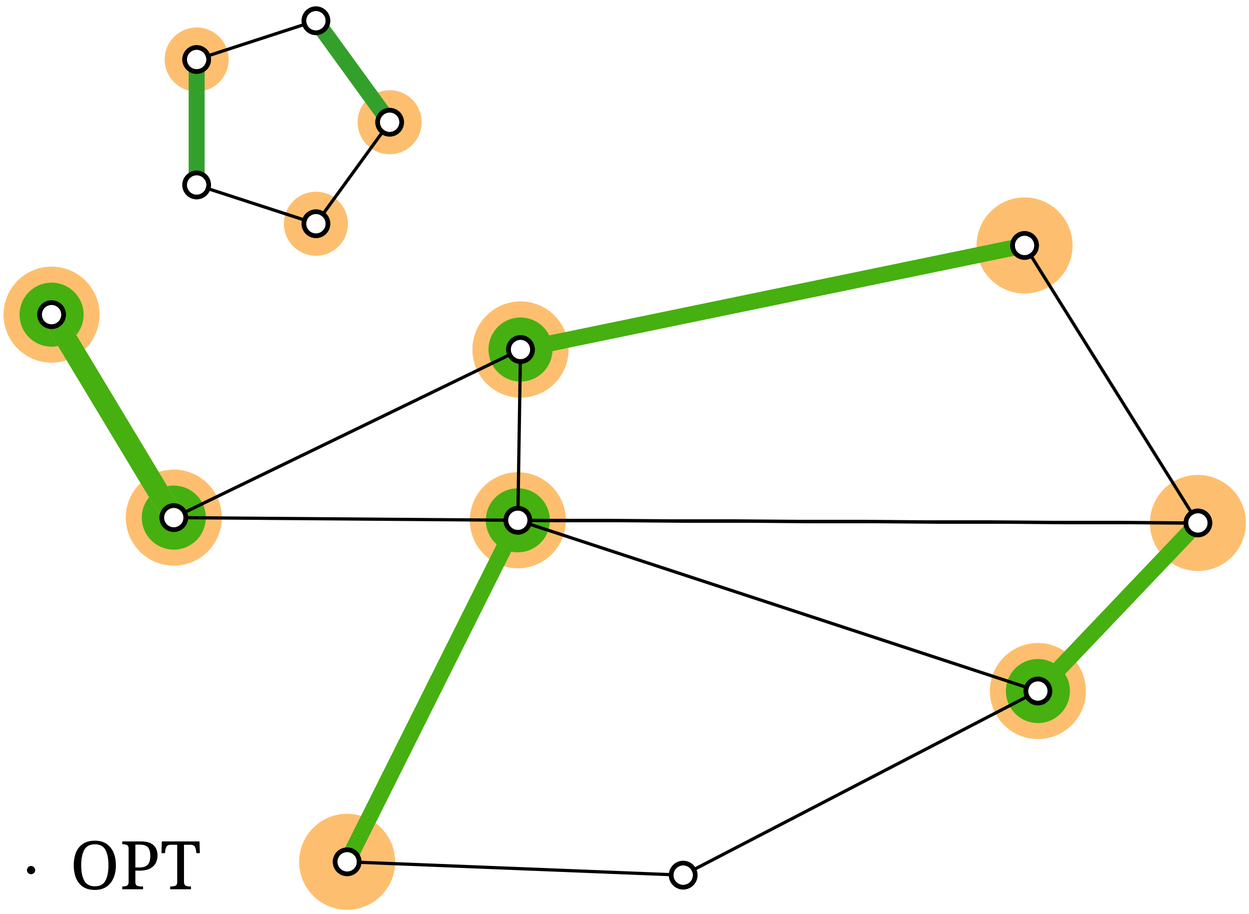
Vertex cover of E

$\mathrm{ALG} = 2 \cdot |M| \leq$

Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a matching if no two edges of M are adjacent (i.e., share an end vertex).

M is maximal if there is no matching M' with $M' \supsetneq M$.



$\mathrm{OPT} \geq |M|$

$\cancel{\mathrm{OPT} = |M| \quad ?}$

Vertex cover of M

Vertex cover of E

$\text{ALG} = 2 \cdot |M| \leq \; 2 \; \cdot \; \mathrm{OPT}$

Approximation Algorithm for VERTEXCOVER

Algorithm VertexCover(G)

$M \leftarrow \emptyset$

Approximation Algorithm for VertexCover

Algorithm VertexCover(G)

$M \leftarrow \emptyset$

foreach $e \in E(G)$ **do**

Approximation Algorithm for VERTEXCOVER

Algorithm VertexCover(G)

$M \leftarrow \emptyset$

foreach $e \in E(G)$ **do**

  if e is not adjacent to any edge in M **then**

Approximation Algorithm for VertexCover

Algorithm VertexCover(G)

$M \leftarrow \emptyset$

foreach $e \in E(G)$ **do**

  if e is not adjacent to any edge in M **then**

    $M \leftarrow M \cup \{e\}$

Approximation Algorithm for VertexCover

Algorithm VertexCover(G)

$M \leftarrow \emptyset$

foreach $e \in E(G)$ **do**

  if e is not adjacent to any edge in M **then**

    $M \leftarrow M \cup \{e\}$

return $\{ u, v \mid uv \in M \}$

Approximation Algorithm for VertexCover

Algorithm VertexCover(G)

$M \leftarrow \emptyset$
foreach $e \in E(G)$ **do**
  if e is not adjacent to any edge in M **then**
    $M \leftarrow M \cup \{e\}$
return $\{ u, v \mid uv \in M \}$

Theorem. The above algorithm is a factor-2 approximation algorithm for VertexCover.

Approximation Algorithm for VERTEXCOVER

Algorithm VertexCover(G)

$M \leftarrow \emptyset$
foreach $e \in E(G)$ **do**
  if e is not adjacent to any edge in M **then**
    $M \leftarrow M \cup \{e\}$
return $\{\, u, v \mid uv \in M \,\}$

Theorem. The above algorithm is a factor-2 approximation algorithm for VERTEXCOVER.

Questions:
- better analysis of this algorithm?
- better algorithm based on this lower bound?
- better lower bound for designing an approx Alg?

Approximation Alg. for Vertex Cover

Questions:
- better analysis of this algorithm?
- better algorithm based on this lower bound?
- better lower bound for designing an approx Alg?

Approximation Alg. for Vertex Cover

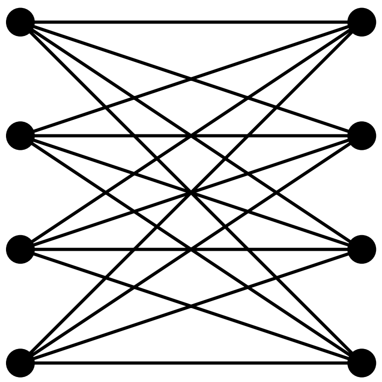
Questions:
- better analysis of this algorithm?
- better algorithm based on this lower bound?
- better lower bound for designing an approx Alg?

tight example: graph with $ALG = 2\ \mathrm{OPT}$?

Approximation Alg. for Vertex Cover

Questions:
- better analysis of this algorithm?
- better algorithm based on this lower bound?
- better lower bound for designing an approx Alg?

No! On the $K_{n,n}$ the algorithm will pick all $2n$ vertices

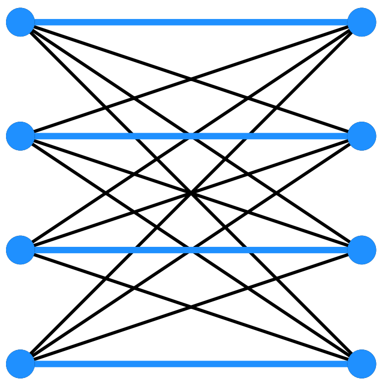


Approximation Alg. for Vertex Cover

Questions:
- better analysis of this algorithm?
- better algorithm based on this lower bound?
- better lower bound for designing an approx Alg?

No! On the $K_{n,n}$ the algorithm will pick all $2n$ vertices

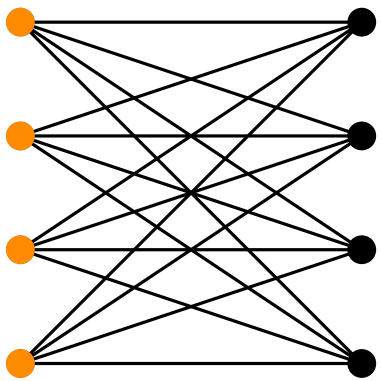
because any maximal matching has size n



Approximation Alg. for Vertex Cover

Questions:
- better analysis of this algorithm?
- better algorithm based on this lower bound?
- better lower bound for designing an approx Alg?

No! On the $K_{n,n}$ the algorithm will pick all $2n$ vertices



but an optimal vertex cover has size n

Approximation Alg. for Vertex Cover

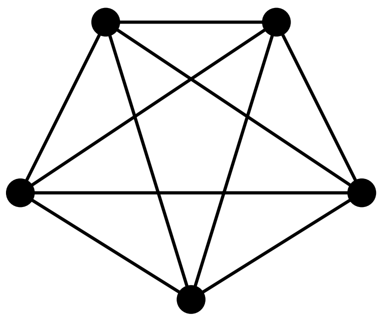
Questions:
- better analysis of this algorithm?
- better algorithm based on this lower bound?
- better lower bound for designing an approx Alg?

tight example: graph with $|M| = \mathrm{OPT}/2$?

Approximation Alg. for Vertex Cover

Questions:
- better analysis of this algorithm?
- better algorithm based on this lower bound?
- better lower bound for designing an approx Alg?

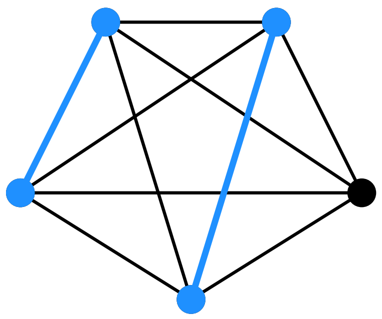
No! On the K_n, where n odd, the size . . .



Approximation Alg. for Vertex Cover

Questions:
- better analysis of this algorithm?
- better algorithm based on this lower bound?
- better lower bound for designing an approx Alg?

No! On the K_n, where n odd, the size . . .

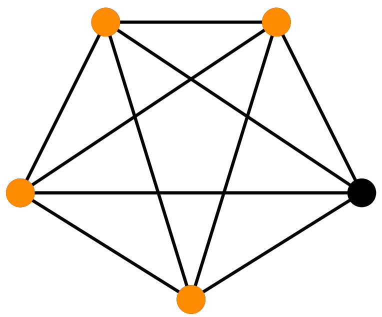


of a maximal matching is $(n-1)/2$

Approximation Alg. for Vertex Cover

Questions:
- better analysis of this algorithm?
- better algorithm based on this lower bound?
- better lower bound for designing an approx Alg?

No! On the K_n, where n odd, the size . . .



of a maximal matching is $(n-1)/2$

of a minimal vertex cover is $n-1$

Approximation Alg. for Vertex Cover

Questions:

- better analysis of this algorithm?
- better algorithm based on this lower bound?
- better lower bound for designing an approx Alg?

Still open!

Approximation Alg. for Vertex Cover

Questions:
- better analysis of this algorithm?
- better algorithm based on this lower bound?
- better lower bound for designing an approx Alg?

Still open!

The best known approximation factor for VertexCover is

$$2 - \Theta(1/\sqrt{\log n}).$$

Approximation Alg. for Vertex Cover

Questions:
- better analysis of this algorithm?
- better algorithm based on this lower bound?
- better lower bound for designing an approx Alg?

Still open!

The best known approximation factor for VertexCover is

$$2 - \Theta(1/\sqrt{\log n}).$$

If P $\neq$ NP, VertexCover cannot be approximated within a factor of 1.3606.

Approximation Alg. for Vertex Cover

Questions:
- better analysis of this algorithm?
- better algorithm based on this lower bound?
- better lower bound for designing an approx Alg?

Still open!

The best known approximation factor for VertexCover is

$$2 - \Theta(1/\sqrt{\log n}).$$

If P ≠ NP, VertexCover cannot be approximated within a factor of 1.3606.

VertexCover cannot be approximated within a factor of $2 - \Theta(1)$
– if the Unique Games Conjecture holds.

assuming that a certain problem
is NP-hard to approximate

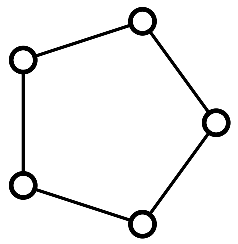
Matching and Vertex Cover

we used the lower bound *max |matching| ≤ min |vertex cover|*

Matching and Vertex Cover

we used the lower bound $max\ |matching| \leq min\ |vertex\ cover|$

and we saw graphs where $max\ |matching| < min\ |vertex\ cover|$

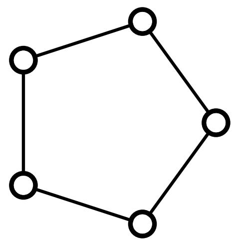


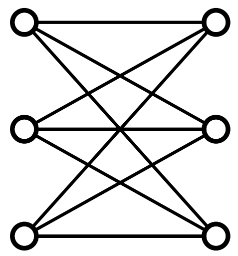
Matching and Vertex Cover

we used the lower bound $max\ |matching| \leq min\ |vertex\ cover|$

and we saw graphs where $max\ |matching| < min\ |vertex\ cover|$

as well as graphs where $max\ |matching| = min\ |vertex\ cover|$





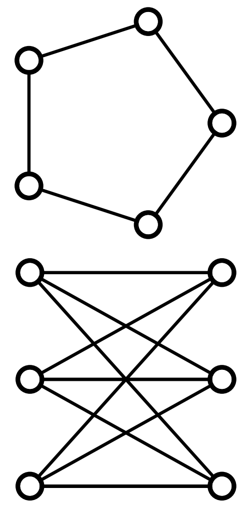
Matching and Vertex Cover

we used the lower bound $\mathit{max}\ |\mathit{matching}| \leq \mathit{min}\ |\mathit{vertex\ cover}|$

and we saw graphs where $\mathit{max}\ |\mathit{matching}| < \mathit{min}\ |\mathit{vertex\ cover}|$

as well as graphs where $\mathit{max}\ |\mathit{matching}| = \mathit{min}\ |\mathit{vertex\ cover}|$

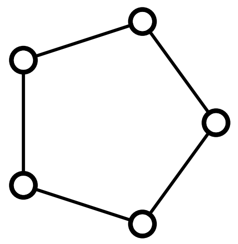
in fact, this holds for all bipartite graphs!



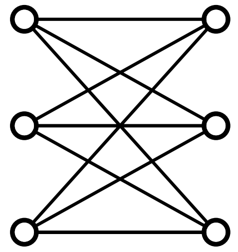
Matching and Vertex Cover

we used the lower bound $\max|\text{matching}| \leq \min|\text{vertex cover}|$

and we saw graphs where $\max|\text{matching}| < \min|\text{vertex cover}|$



as well as graphs where $\max|\text{matching}| = \min|\text{vertex cover}|$



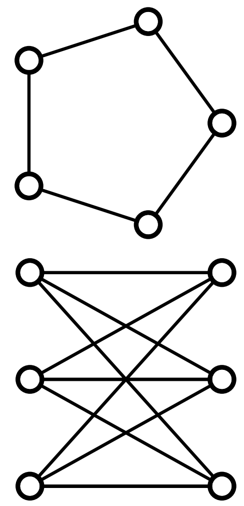
Theorem [König-Egerváry, 1931]
In a bipartite graph, the size of a maximum matching equals the size of a minimum vertex cover.

Matching and Vertex Cover

we used the lower bound $\max|\text{matching}| \leq \min|\text{vertex cover}|$

and we saw graphs where $\max|\text{matching}| < \min|\text{vertex cover}|$

as well as graphs where $\max|\text{matching}| = \min|\text{vertex cover}|$



Theorem [König-Egerváry, 1931]
In a bipartite graph, the size of a maximum matching equals the size of a minimum vertex cover.

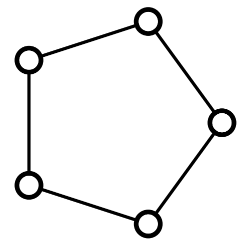
Hence, (only) for bipartite graphs, these problems are equal!

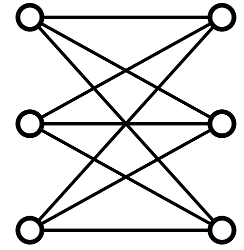
Matching and Vertex Cover

we used the lower bound $\mathit{max}\ |\mathit{matching}| \leq \mathit{min}\ |\mathit{vertex\ cover}|$

and we saw graphs where $\mathit{max}\ |\mathit{matching}| < \mathit{min}\ |\mathit{vertex\ cover}|$

as well as graphs where $\mathit{max}\ |\mathit{matching}| = \mathit{min}\ |\mathit{vertex\ cover}|$

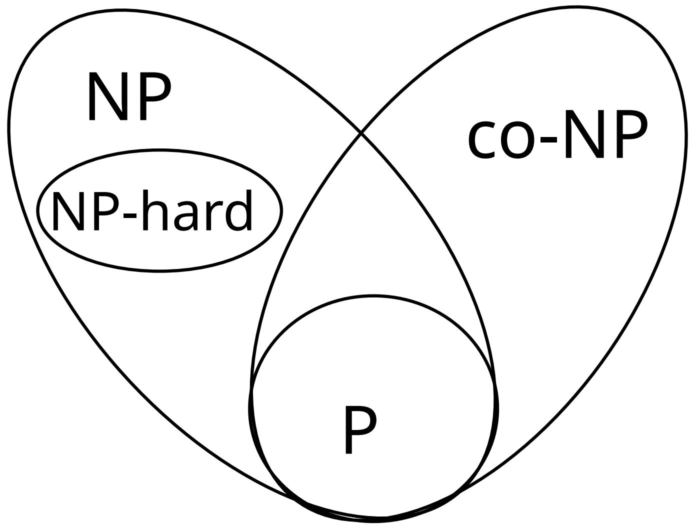




Theorem [König-Egerváry, 1931]
In a bipartite graph, the size of a maximum matching equals the size of a minimum vertex cover.

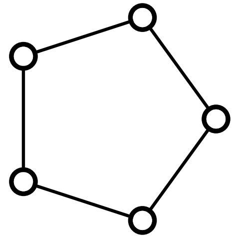
A little complexity theory

Q: Where are vertex cover and matching in general?



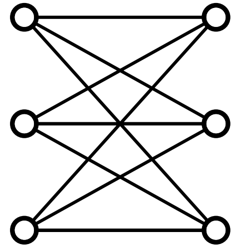
Matching and Vertex Cover

we used the lower bound $max\ |matching| \leq min\ |vertex\ cover|$



and we saw graphs where $max\ |matching| < min\ |vertex\ cover|$

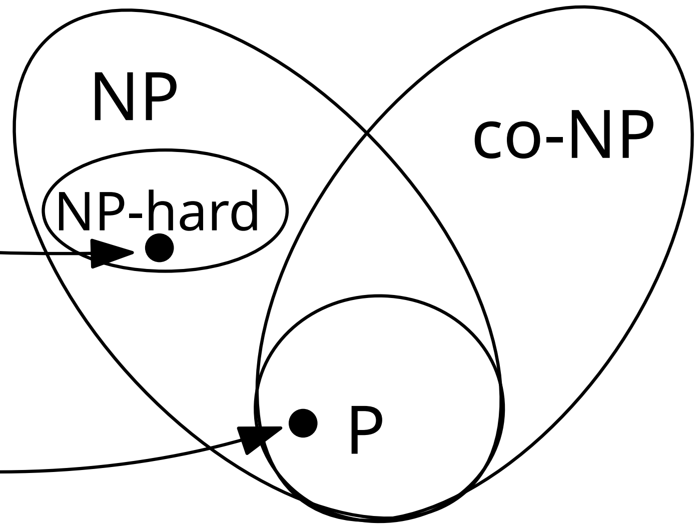
as well as graphs where $max\ |matching| = min\ |vertex\ cover|$



Theorem [König-Egerváry, 1931]
In a bipartite graph, the size of a maximum matching equals the size of a minimum vertex cover.

A little complexity theory

Q: Where are vertex cover
and matching in general?



Summary

Theorem. Minimum-cardinality VERTEXCOVER can be approximated within a factor-2 by greedily computing a maximal matching M, and taking as vertex cover the endpoints of the edges in M.

- key ingredient: lower bound $|M|$ on OPT
- giving tight examples provides crucial insight into the functioning and the algorithm and can provide ideas what to improve

Acknowledgements

Slides for approximation algorithms are mostly due to colleagues in Würzburg,
in particular Joachim Spoerhase and Alexander Wolff. **Thanks!**