Approximation Algorithms

Lecture 1:
Introduction and Vertex Cover

Part I:
Organizational

Philipp Kindermann

Summer Semester 2020

Organizational

Lectures: Pre-recorded (as you see here)

Organizational

Lectures: Pre-recorded (as you see here)
Release date: Tuesday 10:00 (lecture time slot)

Organizational

Lectures: Pre-recorded (as you see here)
Release date: Tuesday 10:00 (lecture time slot)

Tutorials: One sheet per lecture

Organizational

Lectures:	Pre-recorded (as you see here)
	Release date: Tuesday 10:00 (lecture time slot)
Tutorials:	One sheet per lecture
	After one week: Video with sketch of solution

Organizational

Lectures: Pre-recorded (as you see here)
Release date: Tuesday 10:00 (lecture time slot)

Tutorials: One sheet per lecture
After one week: Video with sketch of solution
Can ask questions during tutorial time slot:
Thursday 14:00 - 16:00 (Zoom Meeting)

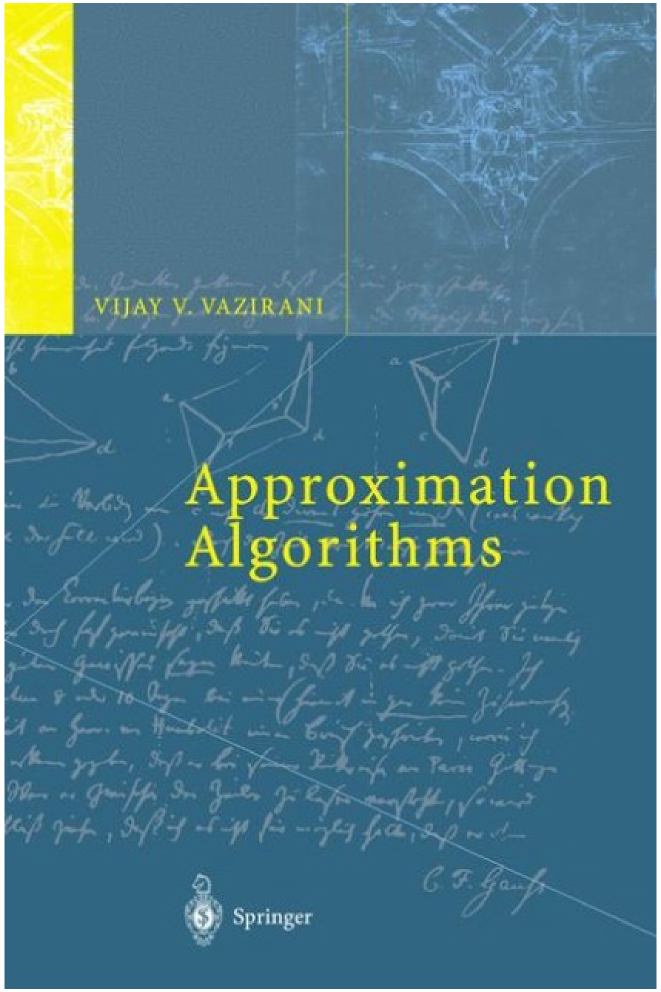
Organizational

Lectures: Pre-recorded (as you see here)
Release date: Tuesday 10:00 (lecture time slot)

Tutorials: One sheet per lecture
After one week: Video with sketch of solution
Can ask questions during tutorial time slot:
Thursday 14:00 - 16:00 (Zoom Meeting)

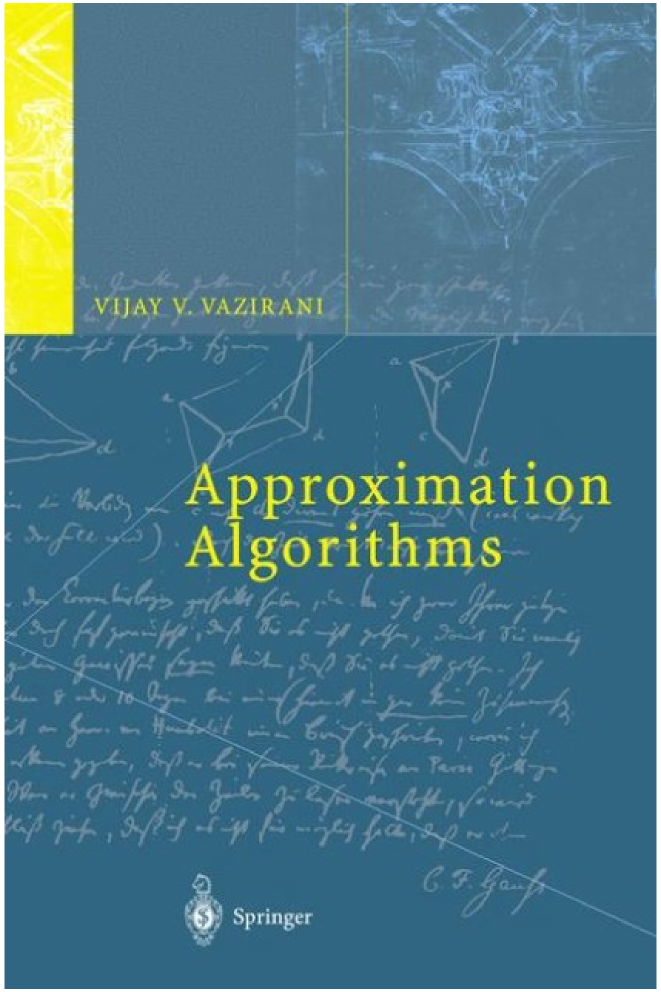
Questions/Tasks during the lecture

Textbooks

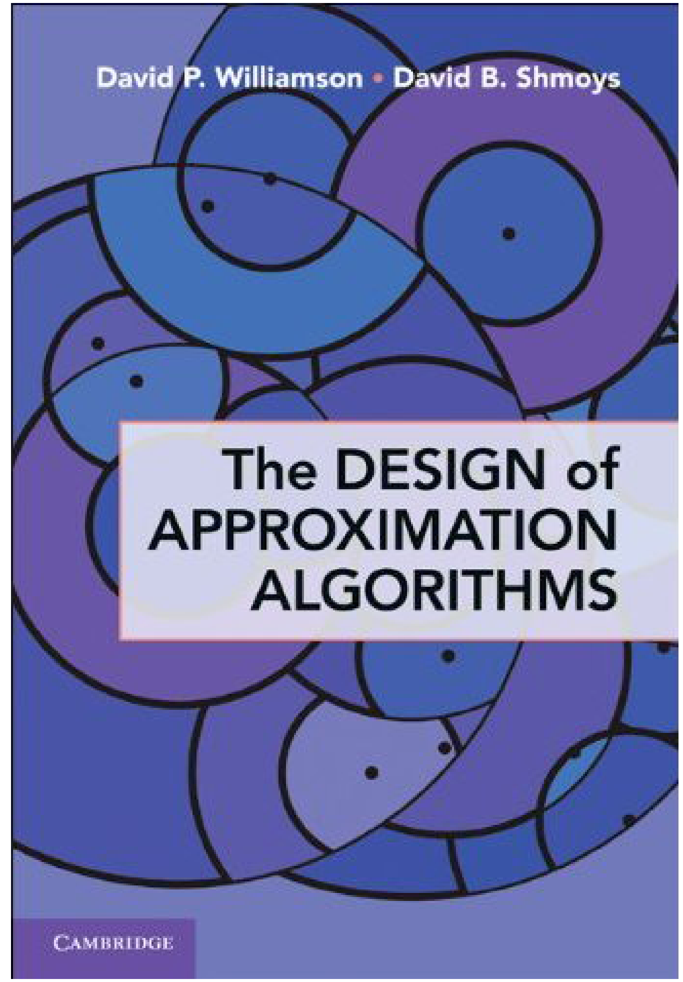


Vijay V. Vazirani:
Approximation
Algorithms
Springer-Verlag, 2003.

Textbooks



Vijay V. Vazirani:
Approximation
Algorithms
Springer-Verlag, 2003.

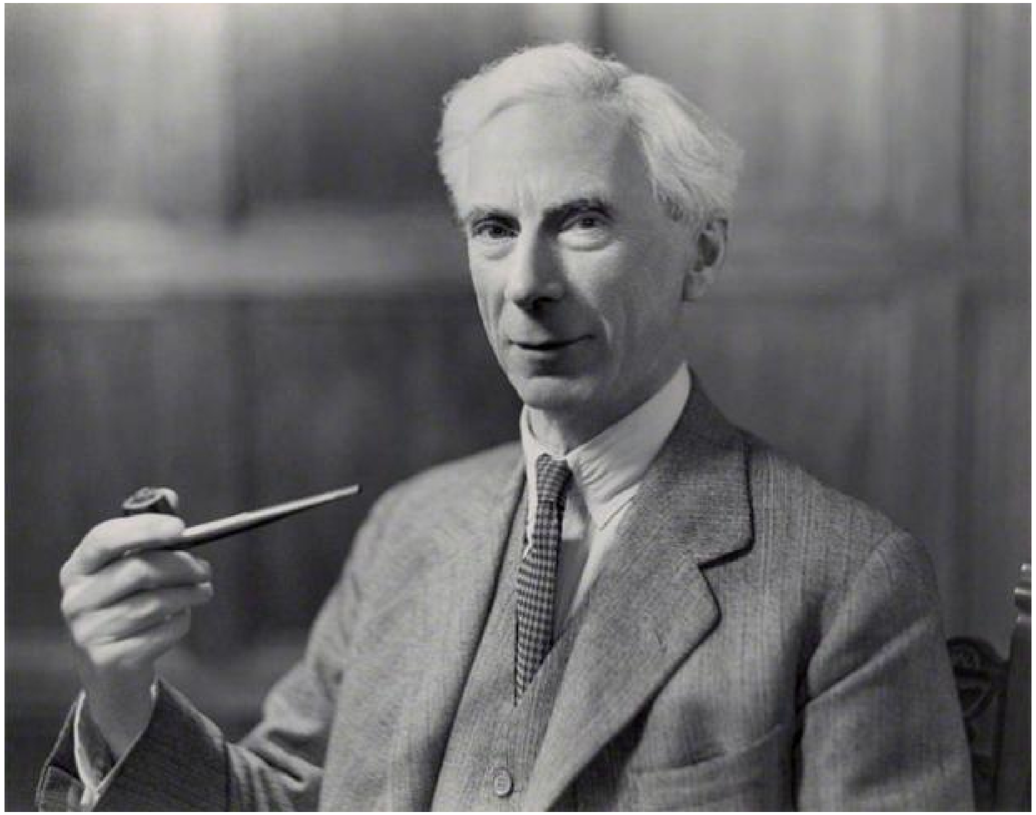


D. P. Williamson & D. B. Shmoys:
The Design of Approximation Algorithms
Cambridge-Verlag, 2011.
http://www.designofapproxalgs.com/

Approximation Algorithms

„All exact science is dominated by the idea of approximation.“

– Bertrand Russell
(1872 – 1970)



Approximation Algorithms

- Many optimization problems are NP-hard (e.g. the traveling salesman problem)

Approximation Algorithms

- Many optimization problems are NP-hard (e.g. the traveling salesman problem)
- $\leadsto$ an optimal solution cannot be efficiently computed unless P=NP.

Approximation Algorithms

- Many optimization problems are NP-hard (e.g. the traveling salesman problem)
- $\rightsquigarrow$ 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 (e.g. the traveling salesman problem)
- $\rightsquigarrow$ 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.

Overview

Combinatorial Algorithms

- Introduction (Vertex Cover)
- Set Cover via Greedy
- Shortest Superstring via reduction to SC
- Steiner Tree via MST
- Multiway Cut via Greedy
- *k*-Center via param. Pruning
- Min-Deg-Spanning-Tree & local search
- Knapsack via DP & Scaling
- Euclidean TSP via Quadtrees

Overview

Combinatorial Algorithms

- Introduction (Vertex Cover)
- Set Cover via Greedy
- Shortest Superstring via reduction to SC
- Steiner Tree via MST
- Multiway Cut via Greedy
- *k*-Center via param. Pruning
- Min-Deg-Spanning-Tree & local search
- Knapsack via DP & Scaling
- Euclidean TSP via Quadtrees

LP-based Algorithms

- introduction to LP-Duality
- Set Cover via LP Rounding
- Set Cover via Primal-Dual Schema
- Maximum Satisfiability
- Scheduling und Extrem point solutions
- Steiner Forest via Primal-Dual

Approximation Algorithms

Lecture 1:
Introduction and Vertex Cover

Part II:
Vertex Cover (card.)

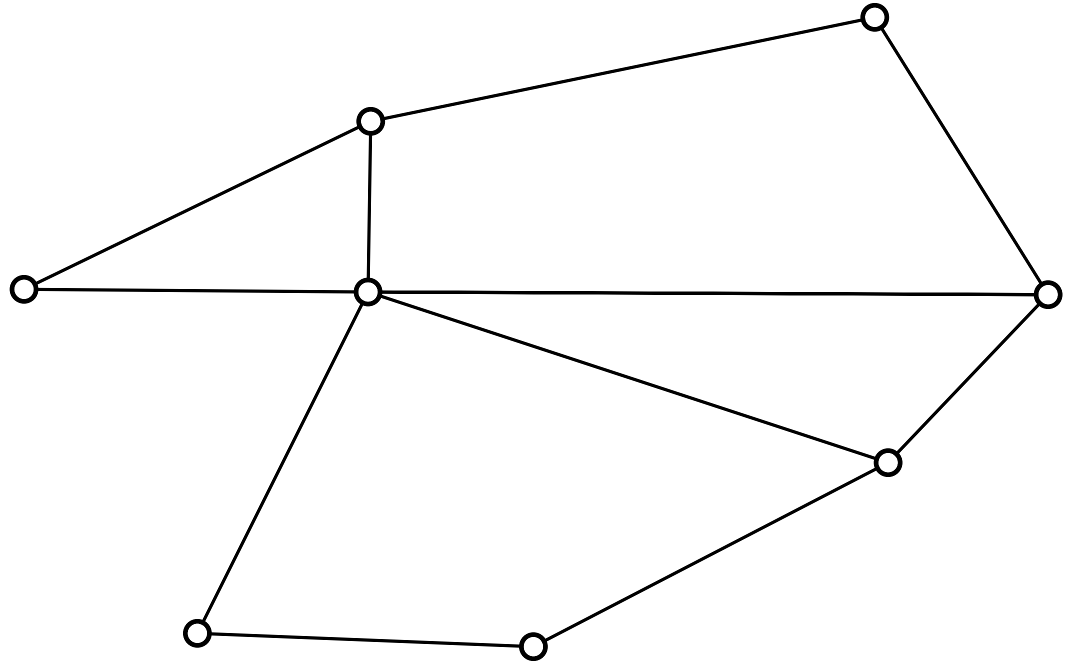
Philipp Kindermann

Summer Semester 2020

VertexCover (card.)

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

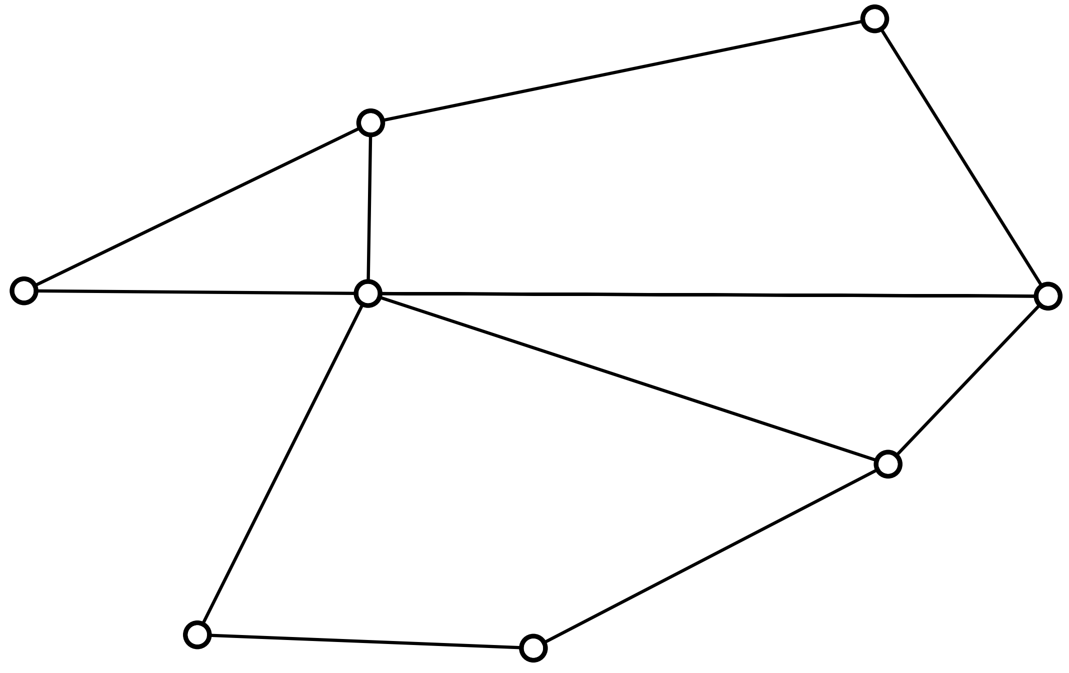
Out:



VertexCover (card.)

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

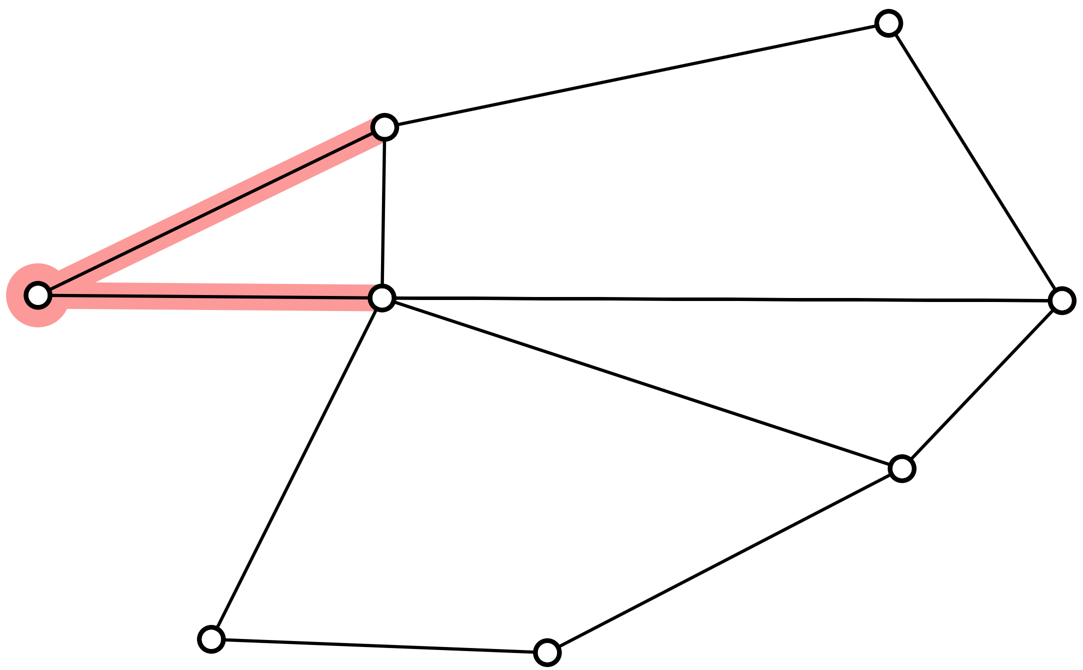
Out: a minimum **vertex cover**: a minimum vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, either $u \in V'$ or $v \in V'$).



VertexCover (card.)

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

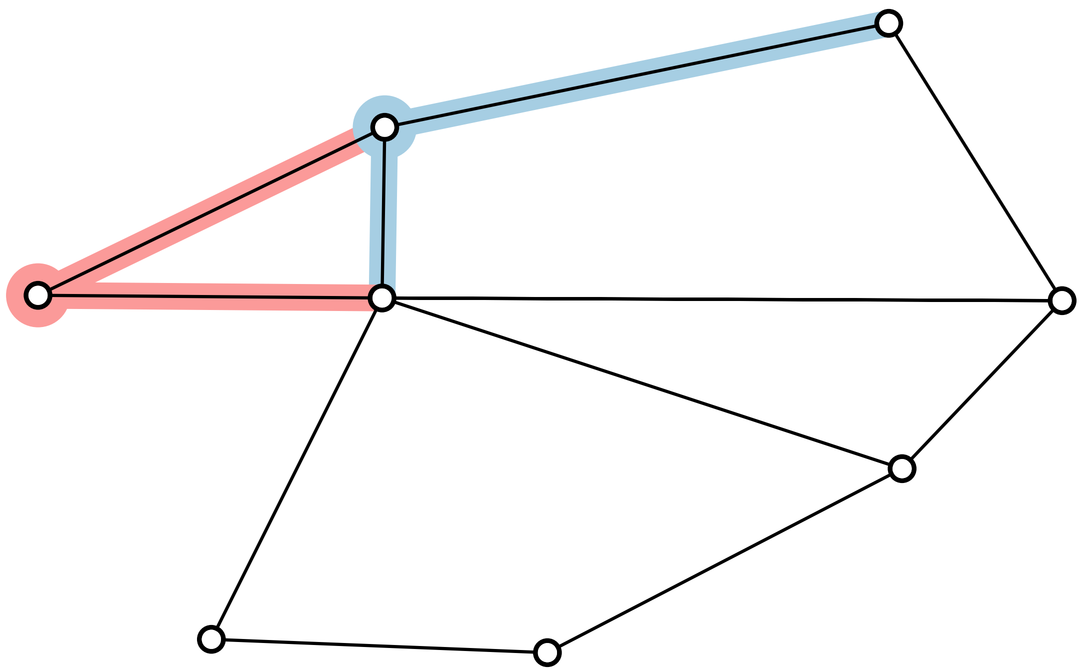
Out: a minimum **vertex cover**: a minimum vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, either $u \in V'$ or $v \in V'$).



VertexCover (card.)

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

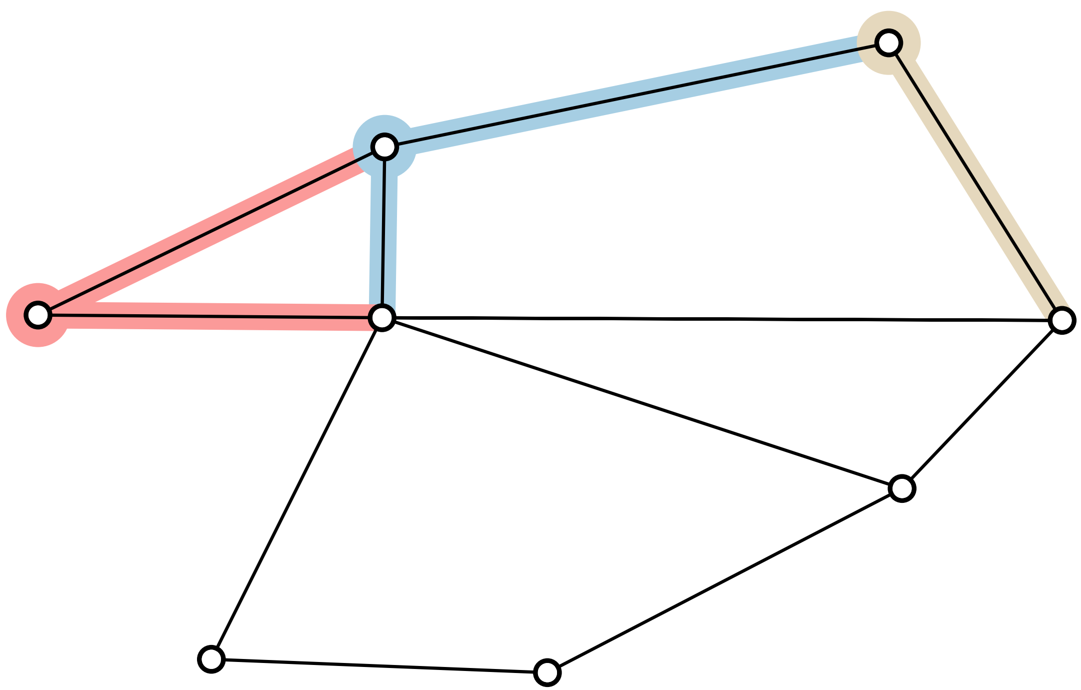
Out: a minimum **vertex cover**: a minimum vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, either $u \in V'$ or $v \in V'$).



VertexCover (card.)

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

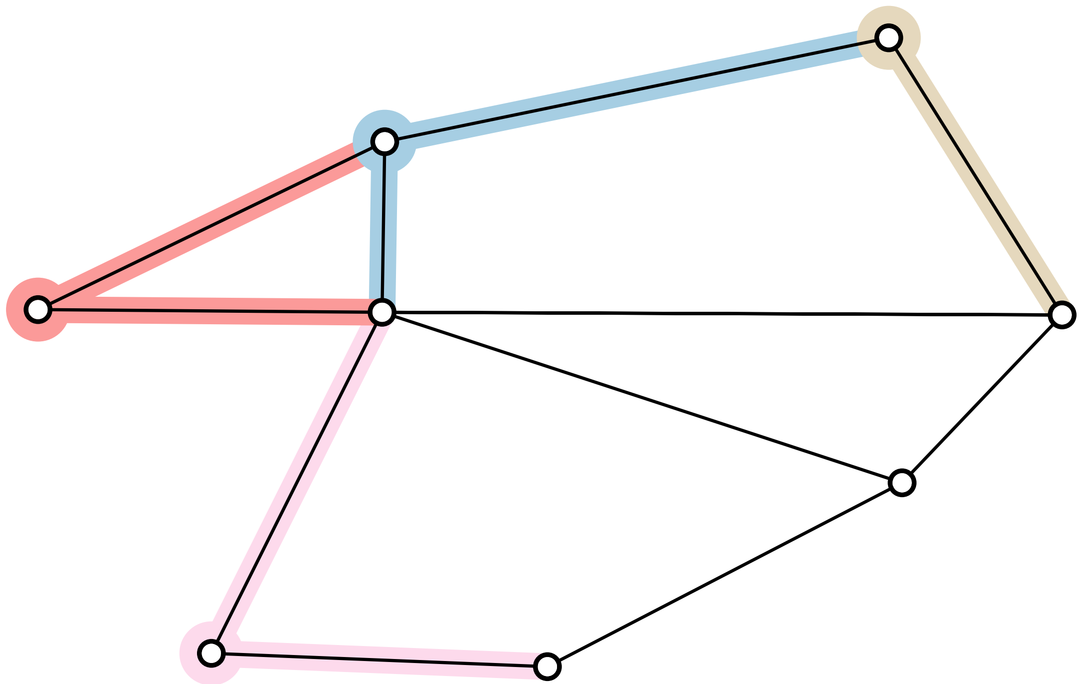
Out: a minimum **vertex cover**: a minimum vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, either $u \in V'$ or $v \in V'$).



VertexCover (card.)

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

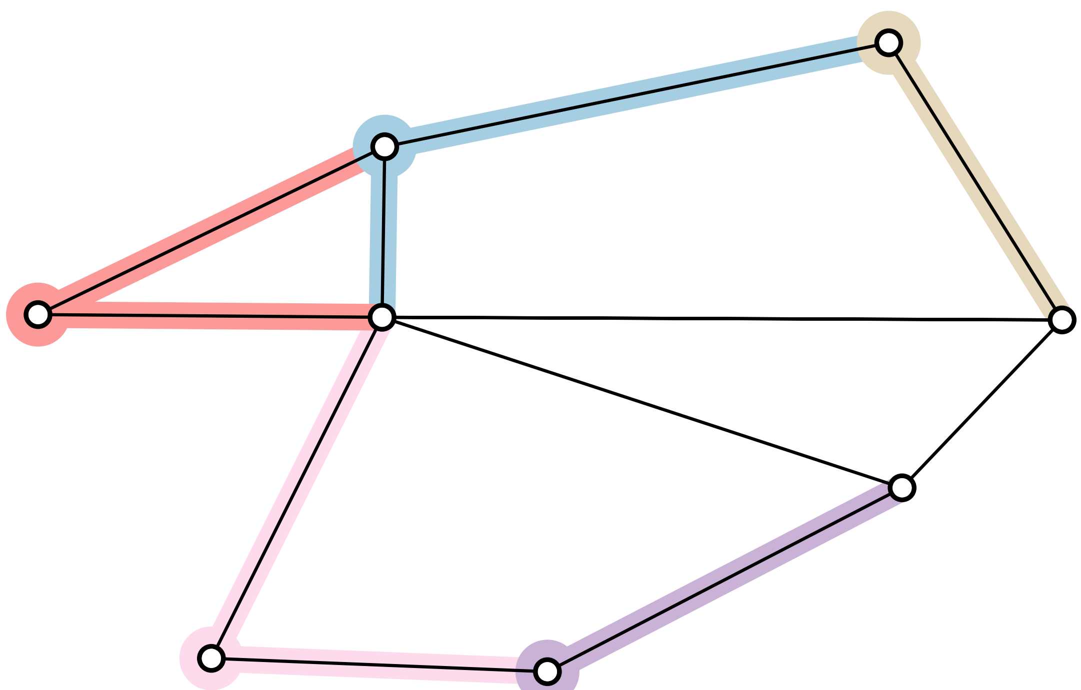
Out: a minimum **vertex cover**: a minimum vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, either $u \in V'$ or $v \in V'$).



VertexCover (card.)

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

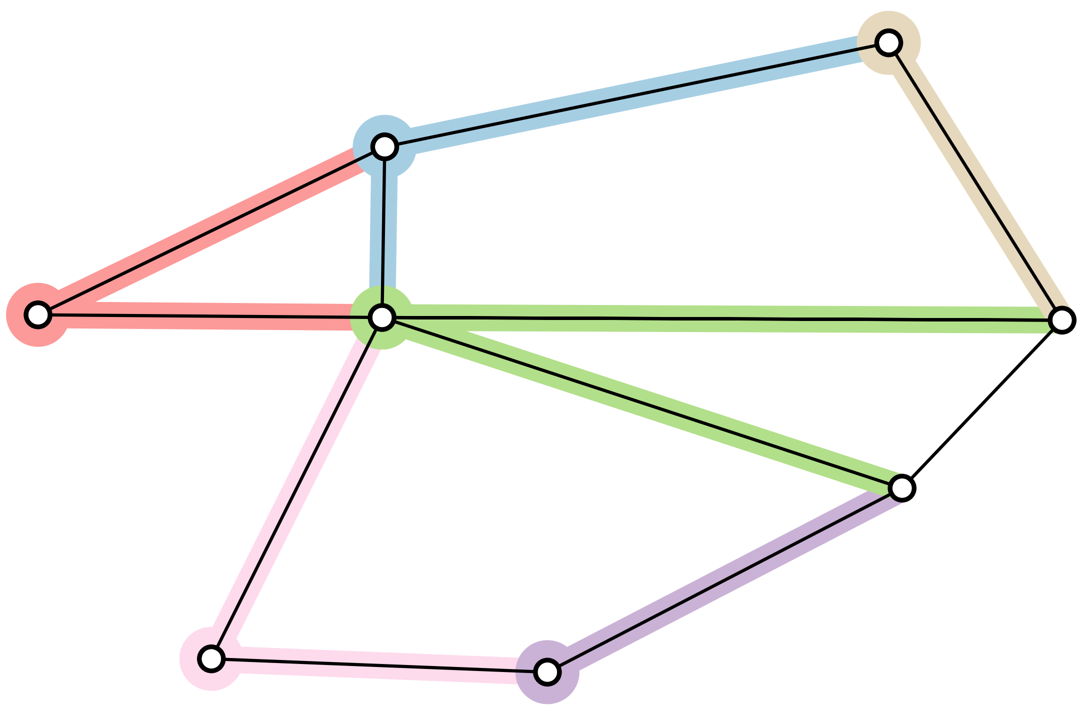
Out: a minimum **vertex cover**: a minimum vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, either $u \in V'$ or $v \in V'$).



VertexCover (card.)

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

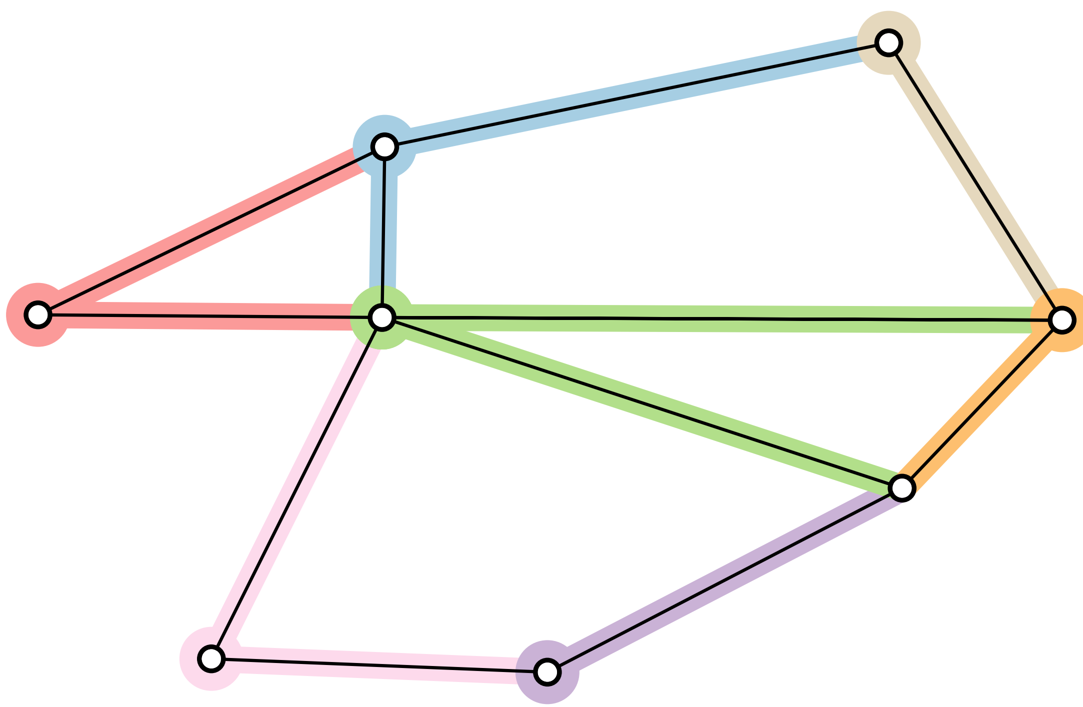
Out: a minimum **vertex cover**: a minimum vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, either $u \in V'$ or $v \in V'$).



VertexCover (card.)

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

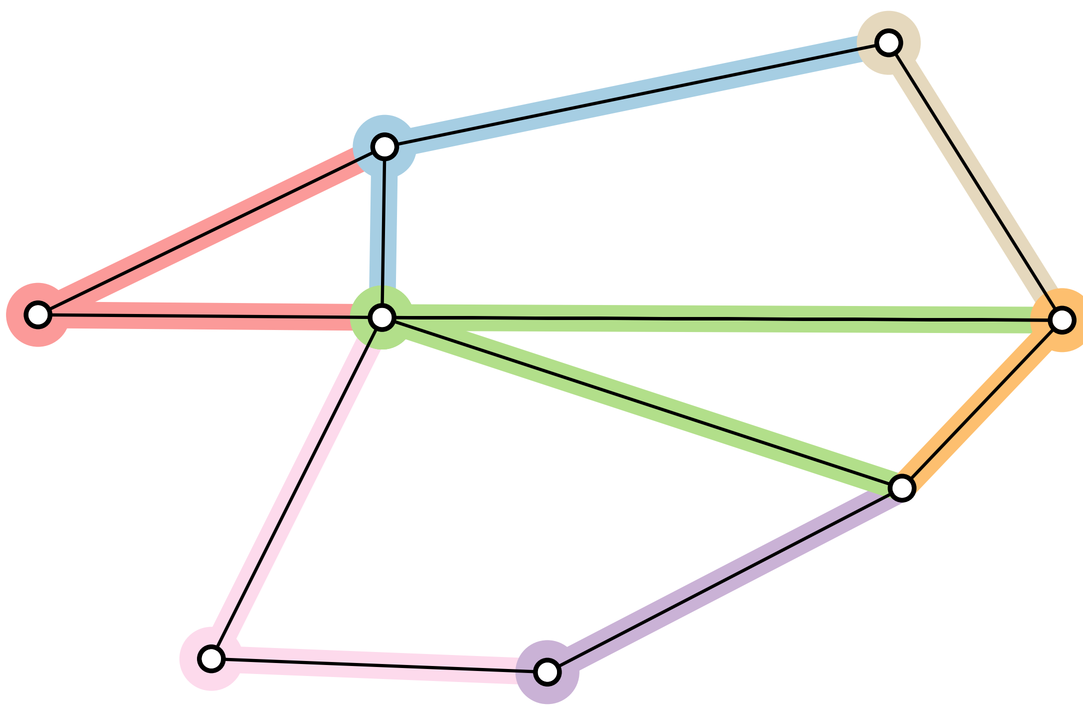
Out: a minimum **vertex cover**: a minimum vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, either $u \in V'$ or $v \in V'$).



VertexCover (card.)

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

Out: a minimum **vertex cover**: a minimum vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, either $u \in V'$ or $v \in V'$).

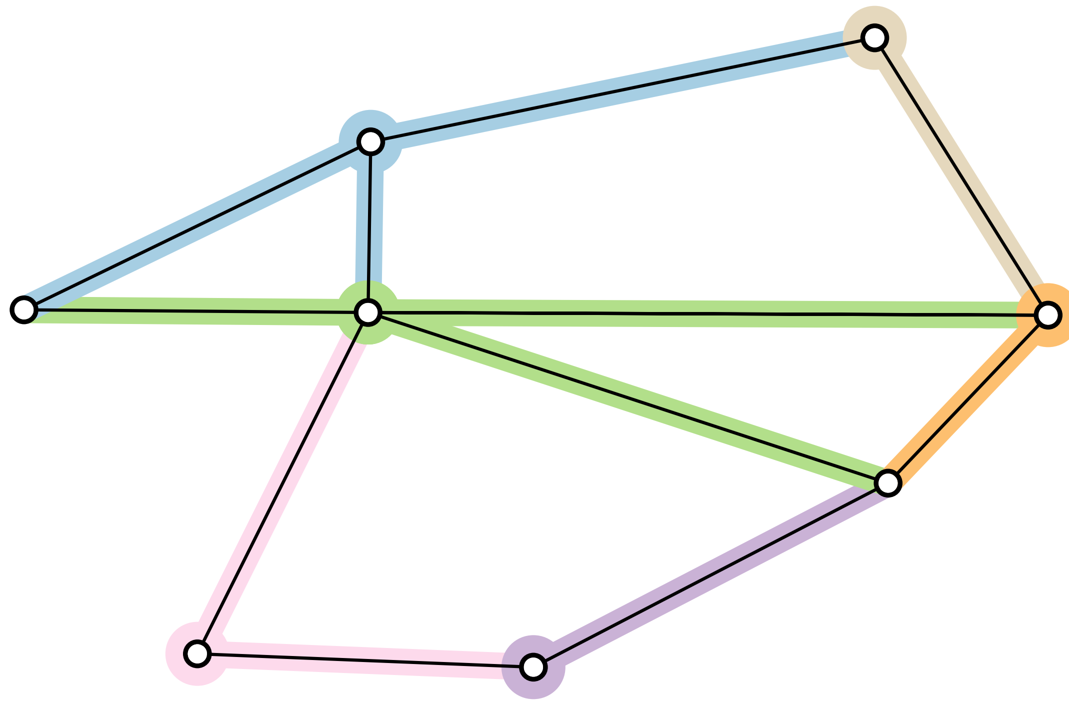


any vertex cover

VertexCover (card.)

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

Out: a minimum **vertex cover**: a minimum vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, either $u \in V'$ or $v \in V'$).

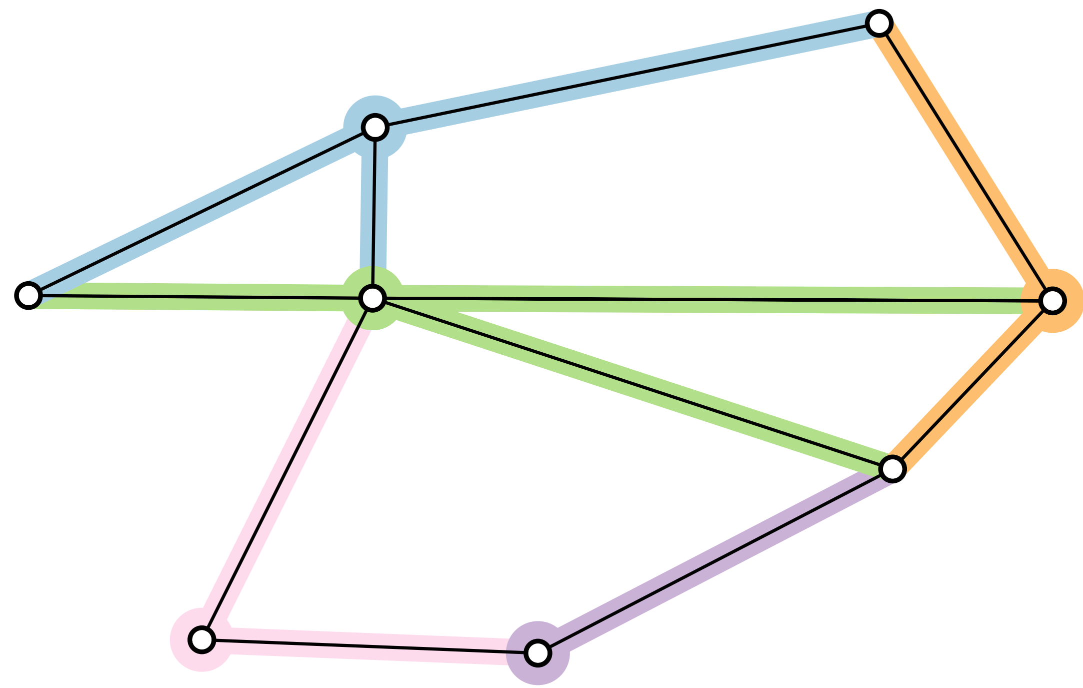


any vertex cover

VertexCover (card.)

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

Out: a minimum **vertex cover**: a minimum vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, either $u \in V'$ or $v \in V'$).

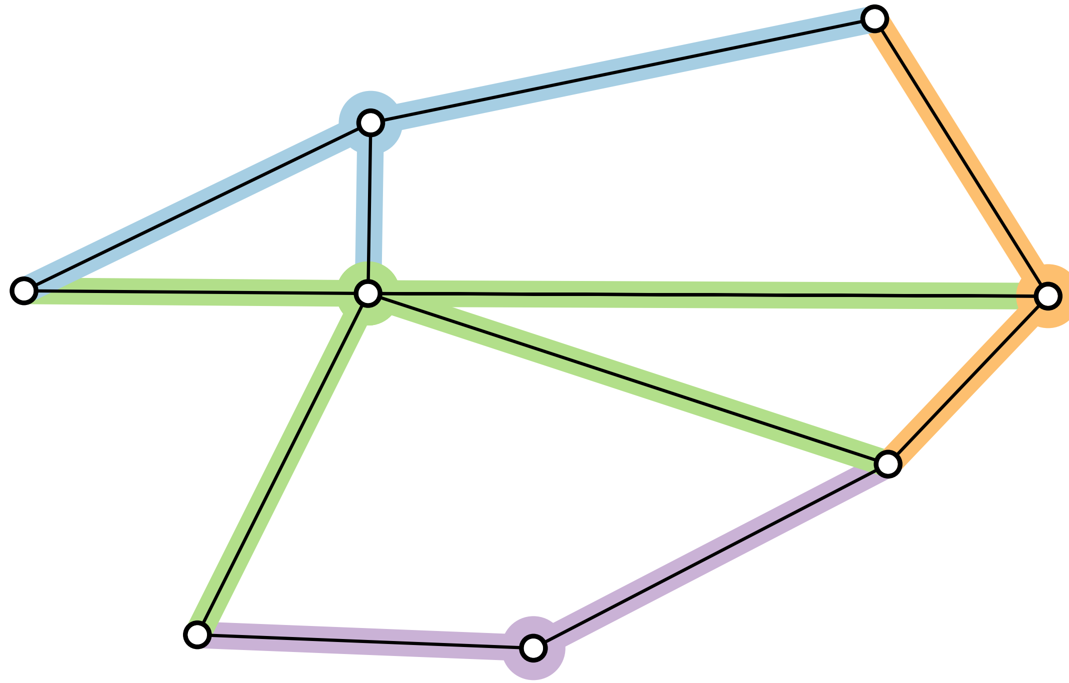


any vertex cover

VertexCover (card.)

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

Out: a minimum **vertex cover**: a minimum vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, either $u \in V'$ or $v \in V'$).

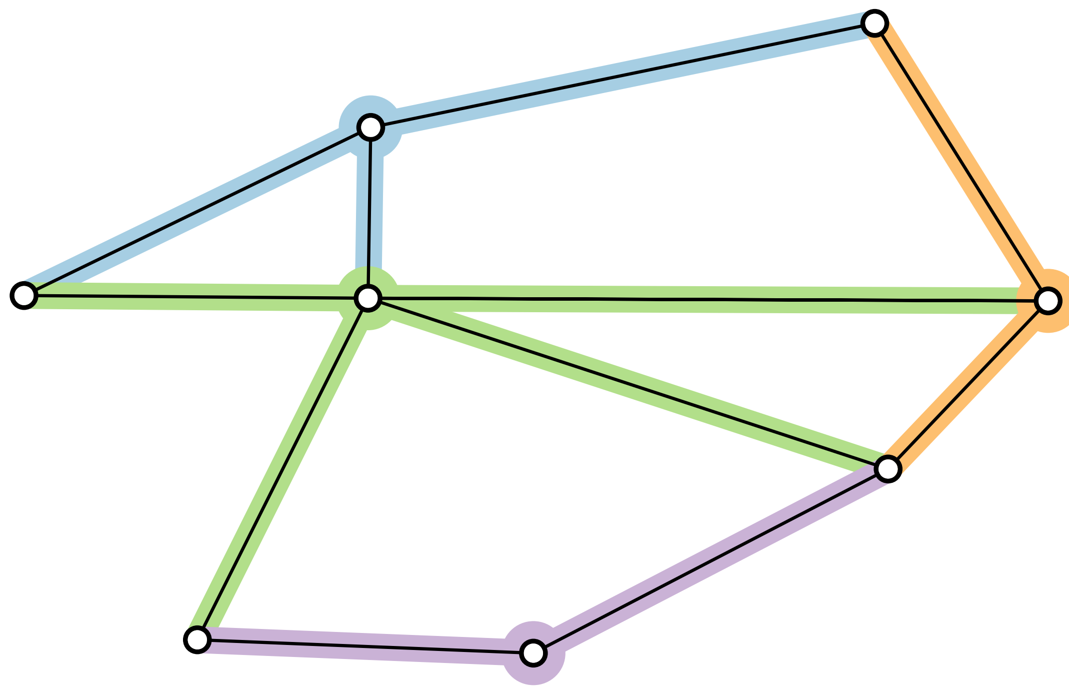


any vertex cover

VertexCover (card.)

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

Out: a minimum **vertex cover**: a minimum vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, either $u \in V'$ or $v \in V'$).

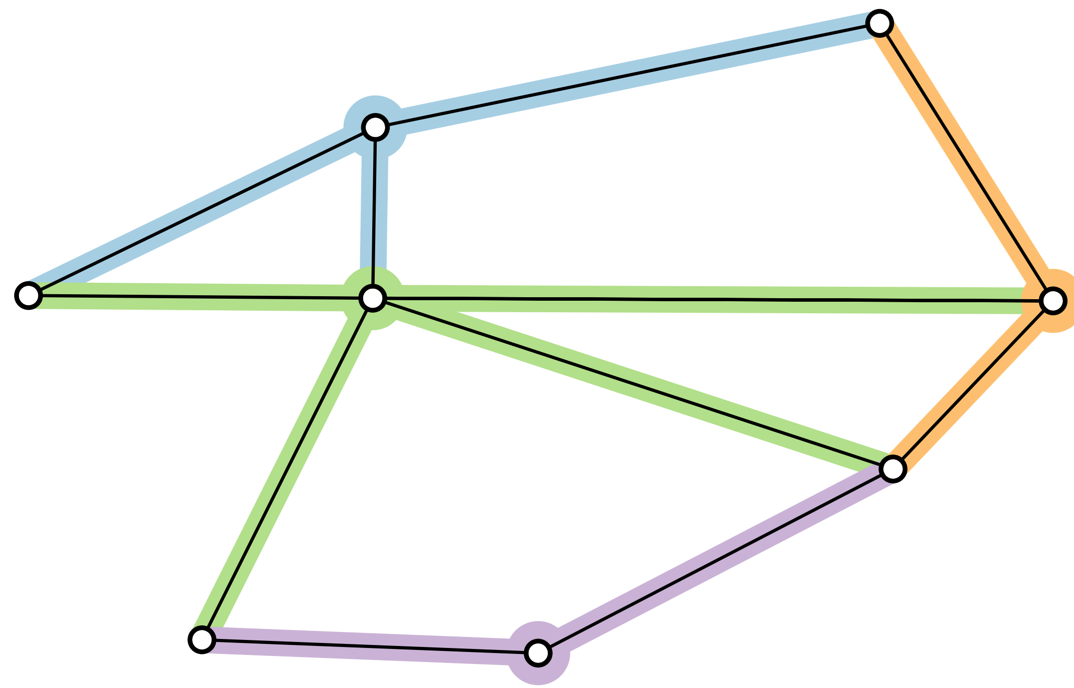


Optimum (OPT $= 4$)

VertexCover (card.)

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

Out: a minimum **vertex cover**: a minimum vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, either $u \in V'$ or $v \in V'$).

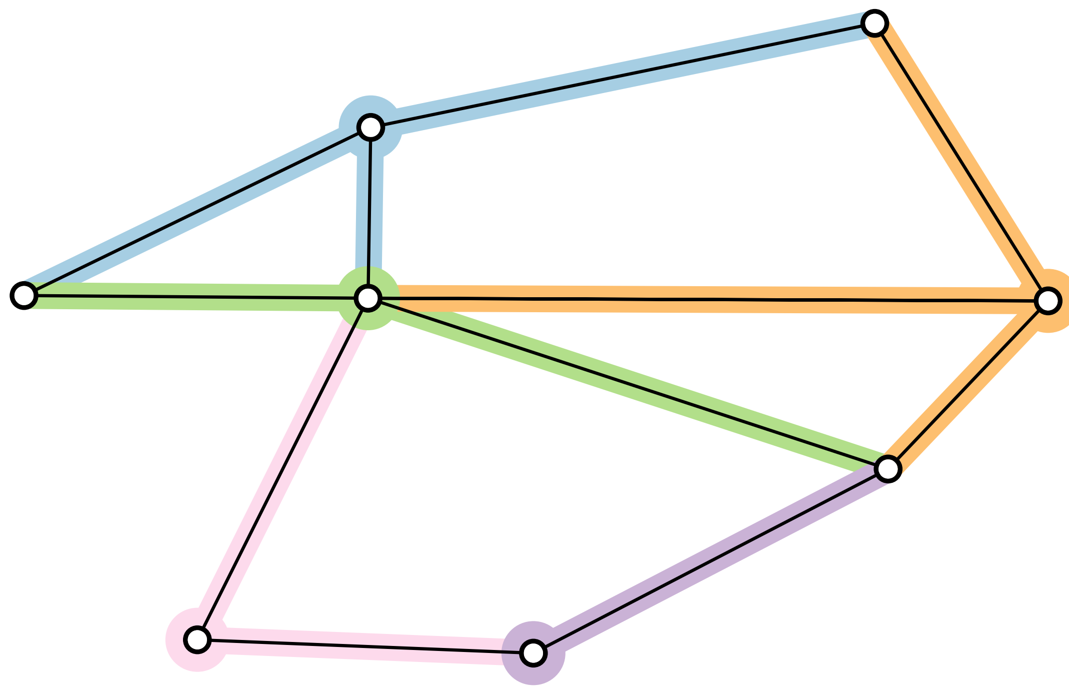


Optimum (OPT $= 4$) – but in general NP-hard to find :-(

VertexCover (card.)

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

Out: a minimum **vertex cover**: a minimum vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, either $u \in V'$ or $v \in V'$).



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

Approximation Algorithms

Lecture 1:
Introduction and Vertex Cover

Part III:
NP-Optimization Problem

Philipp Kindermann

Summer Semester 2020

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
  - for each pair (s, I), there is a polynomial time algorithm to decide 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
  - for each pair (s, I), there is a polynomial time algorithm to decide whether $s \in S_\Pi(I)$.
- A polynomial time computable **target 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
  - for each pair (s, I), there is a polynomial time algorithm to decide whether $s \in S_\Pi(I)$.
- A polynomial time computable **target 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 $\Pi = \textsc{Vertex Cover}$.

$D_\Pi =$

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 M.....imization problem.

VertexCover: NP-Optimization Problem

Task: Fill in the gaps for $\Pi =$ 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 M.....imization problem.

VertexCover: NP-Optimization Problem

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

$D_\Pi =$ Set of all graphs

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

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

- 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 M.....imization problem.

VertexCover: NP-Optimization Problem

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

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: $|I| =$ Number of vertices $|V|$

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

- 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 M.....imization problem.

VertexCover: NP-Optimization Problem

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

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: $|I| =$ Number of vertices $|V|$

$G = (V, E)$ $S_\Pi(I) =$ 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 M.....imization problem.

VertexCover: NP-Optimization Problem

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

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: $|I| =$ Number of vertices $|V|$

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

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

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

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

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

Π is M.....imization problem.

VertexCover: NP-Optimization Problem

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

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: $|I| =$ Number of vertices $|V|$

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

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

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

- 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 M.....imization problem.

VertexCover: NP-Optimization Problem

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

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: $|I| =$ Number of vertices $|V|$

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

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?
  $s \subseteq V \Rightarrow |s| \leq |V| = |I|$
- 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 M.....imization problem.

VertexCover: NP-Optimization Problem

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

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: $|I| =$ Number of vertices $|V|$

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

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?
  $s \subseteq V \Rightarrow |s| \leq |V| = |I|$
- 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 Minimization problem.

Optimum and optimal objective value

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

Optimum and optimal objective value

Let Π be a minimization problem and $I \in D_\Pi$ be an instance of Π.
A feasible solution $s^* \in S_\Pi(I)$ is **optimal** if $\mathrm{obj}_\Pi(I, s^*)$ is minimal among objective values attained by the feasible solutions of I.

Optimum and optimal objective value

maximization problem

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

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

maximal

$\mathrm{obj}_\Pi(I, s^*)$ is minimal among objective values attained by the feasible solutions of I.

Optimum and optimal objective value

maximization problem

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

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

maximal

$\text{obj}_\Pi(I, s^*)$ is minimal among objective values attained by the feasible solutions of I.

The optimal value $\text{obj}_\Pi(I, s^*)$ of the objective function is also denoted by $\text{OPT}_\Pi(I)$ or simply 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 which 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 which provides for **any** instance $I \in D_\Pi$ a feasible solution $s \in S_\Pi(I)$ such that

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

Approximation Algorithms

Let Π be a minimization problem and $\alpha \in \mathbb{Q}^+$.
A factor-α-approximation algorithm for Π is an efficient algorithm which provides for **any** instance $I \in D_\Pi$ a feasible solution $s \in S_\Pi(I)$ such that

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

Approximation Algorithms

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

Let Π be a minimization problem and ~~$\alpha \in \mathbb{Q}^+$~~.
A factor-α-approximation algorithm for Π is an efficient algorithm which provides for **any** instance $I \in D_\Pi$ a feasible solution $s \in S_\Pi(I)$ such that

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

Approximation Algorithms

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

Let Π be a minimization problem and ~~$\alpha \in \mathbb{Q}^+$~~.
A factor-α-approximation algorithm for Π is an efficient algorithm which provides for **any** instance $I \in D_\Pi$ a feasible solution $s \in S_\Pi(I)$ such that

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

Approximation Algorithms

Lecture 1:
Introduction and Vertex Cover

Part IV:
Approximation Algorithm for VertexCover

Philipp Kindermann

Summer Semester 2020

Approximation Alg. for VertexCover

Ideas?

Approximation Alg. for VertexCover

Ideas?

- Edge-Greedy

Approximation Alg. for VertexCover

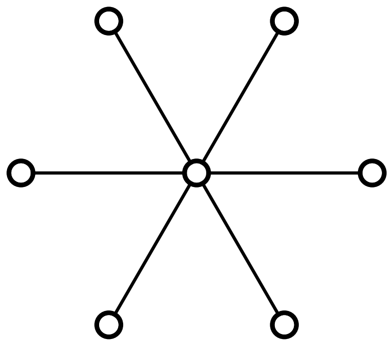
Ideas?

- Edge-Greedy
- Vertex-Greedy

Approximation Alg. for VertexCover

Ideas?

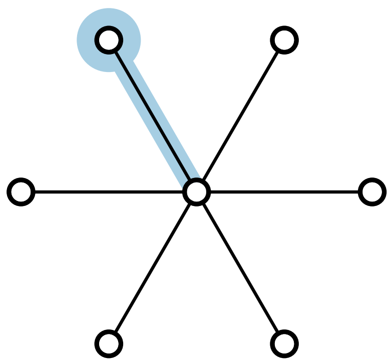
- Edge-Greedy
- Vertex-Greedy



Approximation Alg. for VertexCover

Ideas?

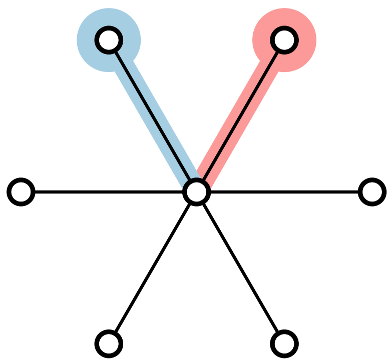
- Edge-Greedy
- Vertex-Greedy



Approximation Alg. for VertexCover

Ideas?

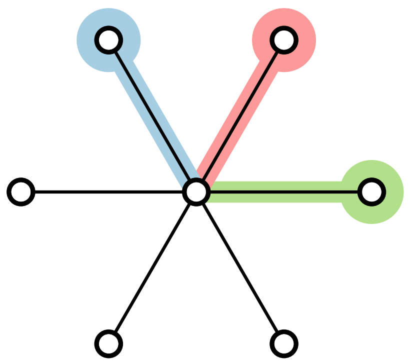
- Edge-Greedy
- Vertex-Greedy



Approximation Alg. for VertexCover

Ideas?

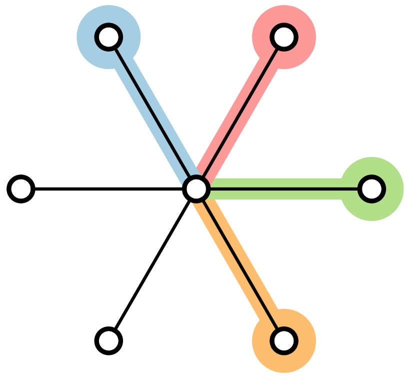
- Edge-Greedy
- Vertex-Greedy



Approximation Alg. for VertexCover

Ideas?

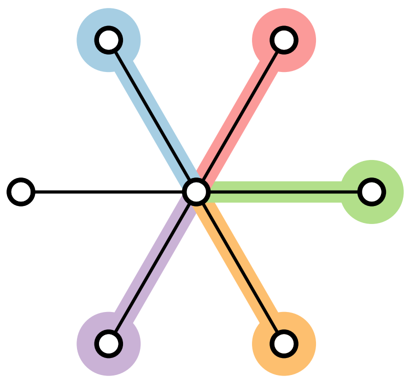
- Edge-Greedy
- Vertex-Greedy



Approximation Alg. for VertexCover

Ideas?

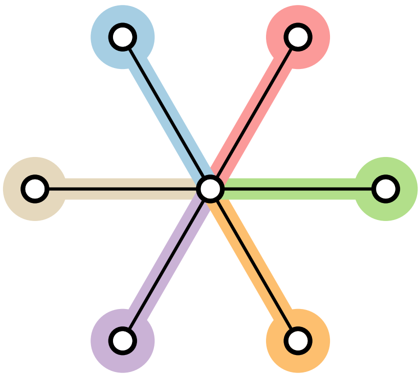
- Edge-Greedy
- Vertex-Greedy



Approximation Alg. for VertexCover

Ideas?

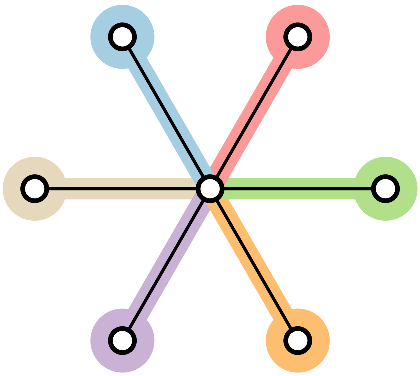
- Edge-Greedy
- Vertex-Greedy



Approximation Alg. for VertexCover

Ideas?

- Edge-Greedy
- Vertex-Greedy

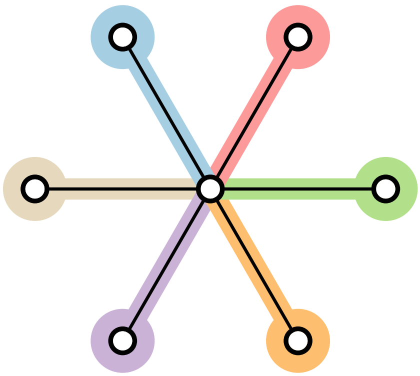


Quality?

Approximation Alg. for VertexCover

Ideas?

- Edge-Greedy
- Vertex-Greedy



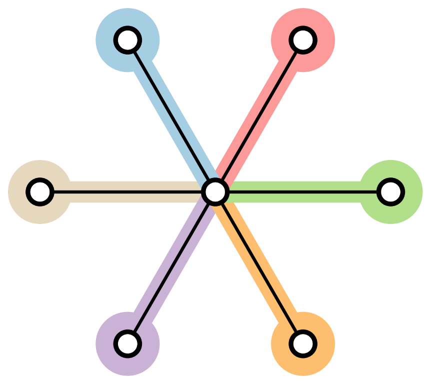
Quality?

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

Approximation Alg. for VertexCover

Ideas?

- Edge-Greedy
- Vertex-Greedy



Quality?

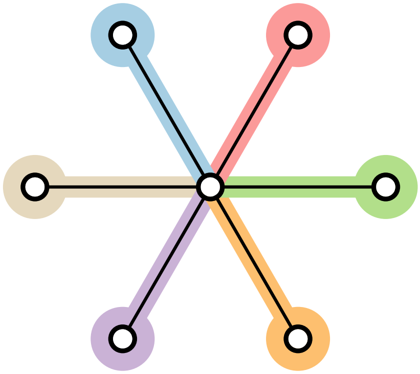
Problem: How can we estimate $\mathrm{obj}_\Pi(I, s)/\mathrm{OPT}$, when it is hard to calculate OPT?

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

Approximation Alg. for VertexCover

Ideas?

- Edge-Greedy
- Vertex-Greedy



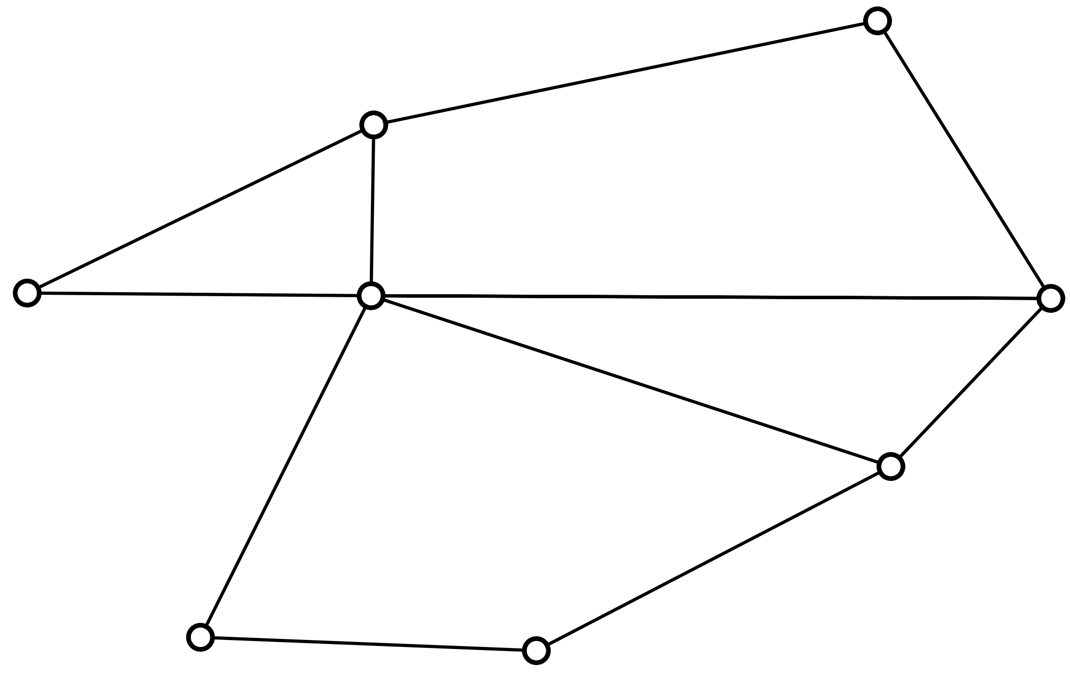
Quality?

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

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

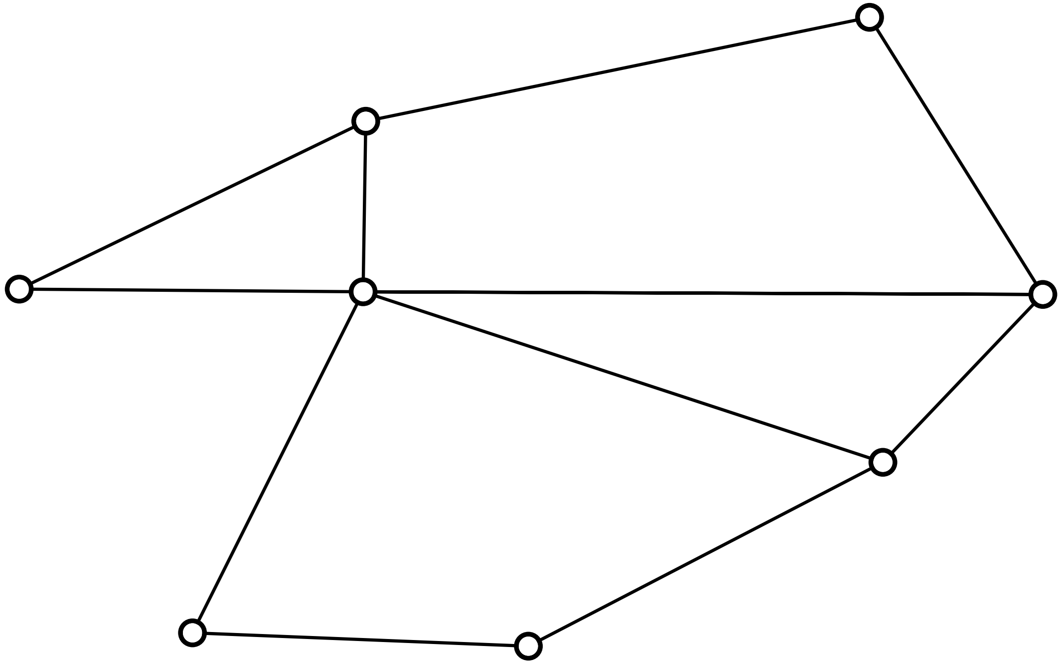
$$\frac{\mathrm{obj}_\Pi(I,s)}{\mathrm{OPT}} \leq \frac{\mathrm{obj}_\Pi(I,s)}{U}$$

Lower Bound by Matchings



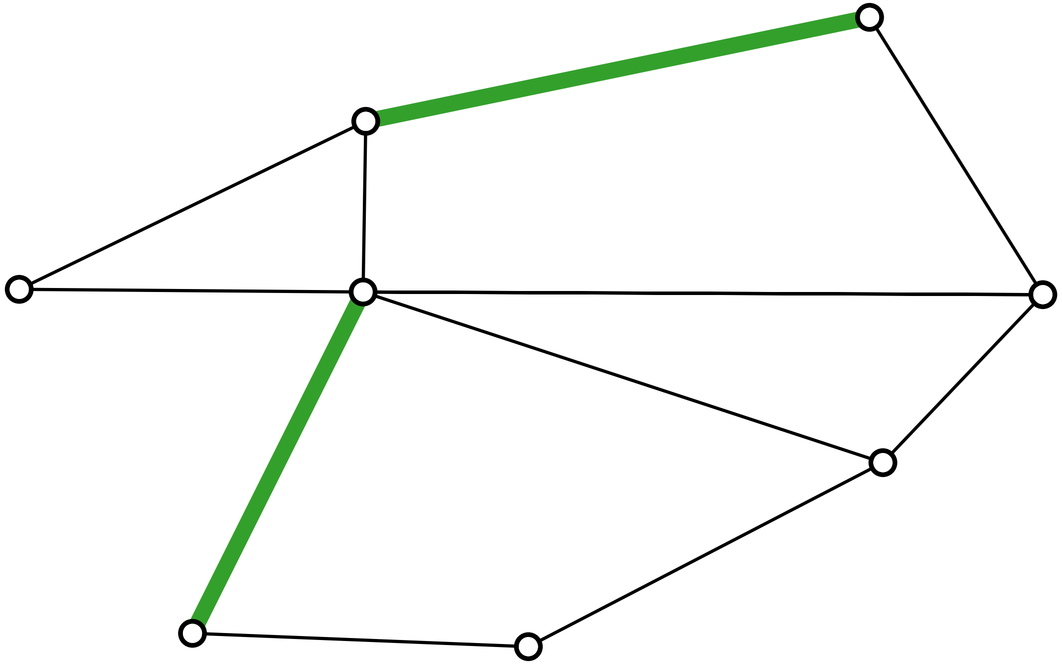
Lower Bound by Matchings

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

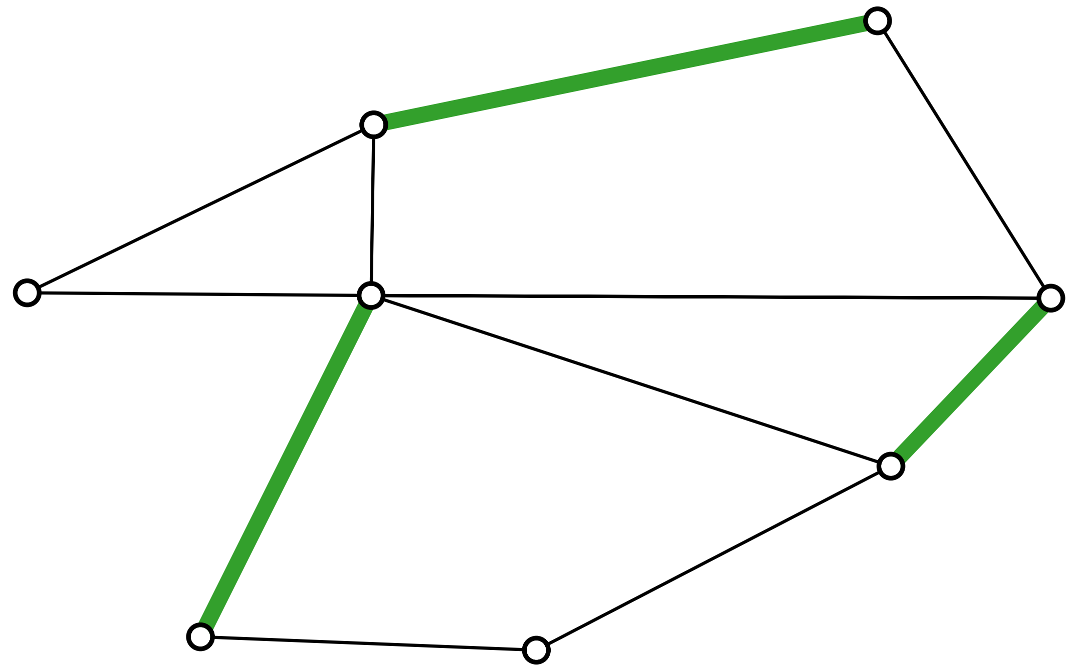
An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a **matching** if no two edges of M are adjacent (i.e., share an end vertex).



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 $M' \supsetneq M$.

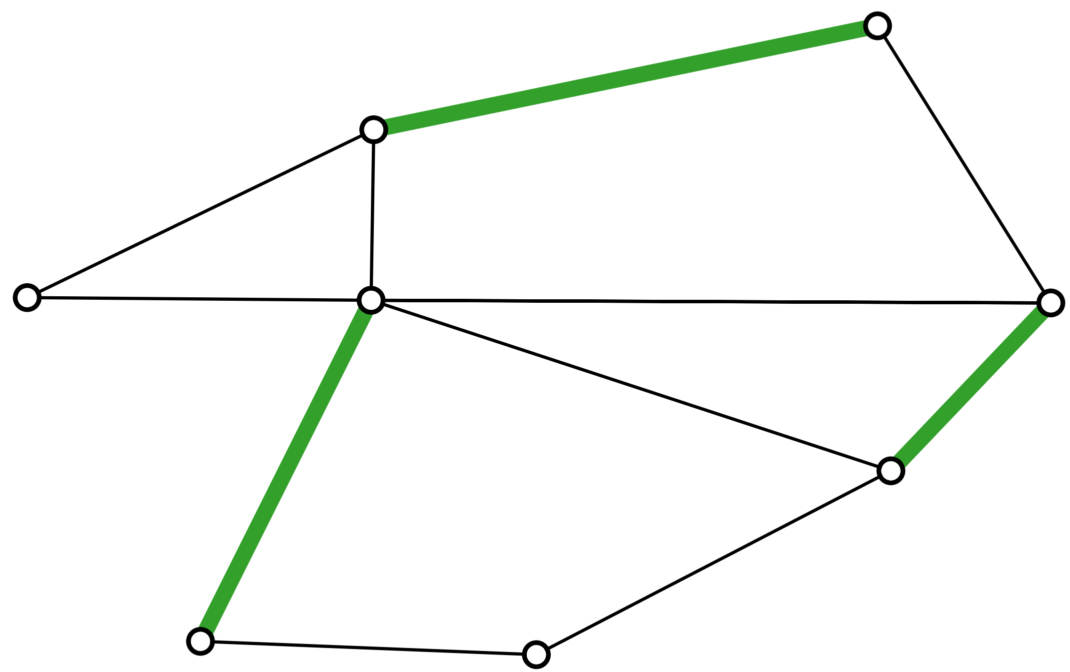


Lower Bound by Matchings

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 $M' \supsetneq M$.

$\mathrm{OPT} \geq$



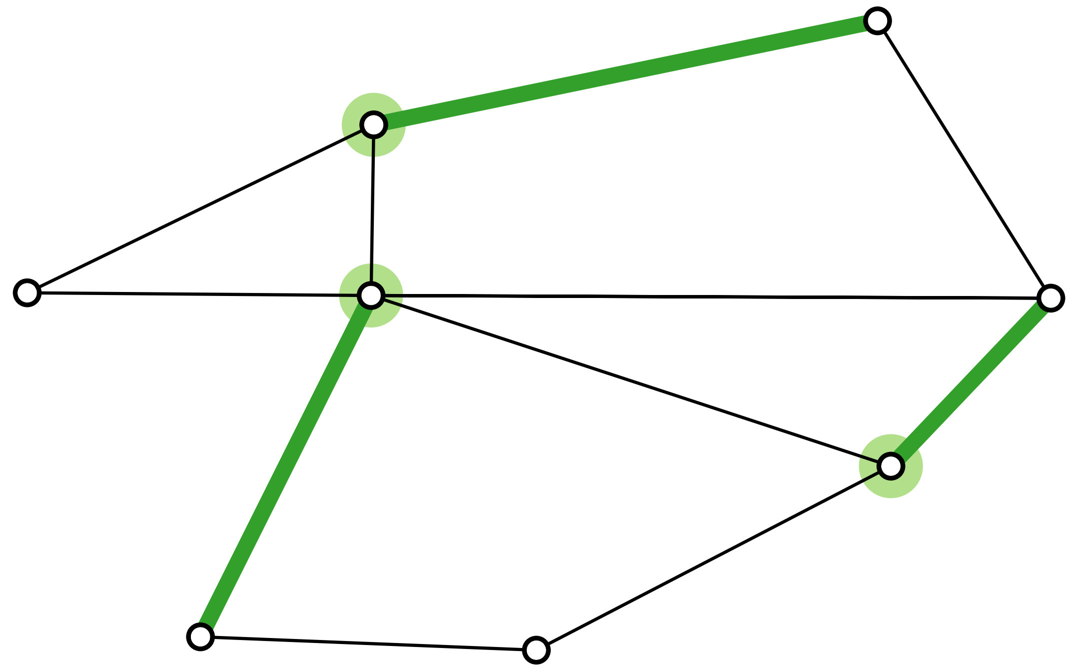
Lower Bound by Matchings

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 $M' \supsetneq M$.

OPT $\geq$

Vertex cover of M



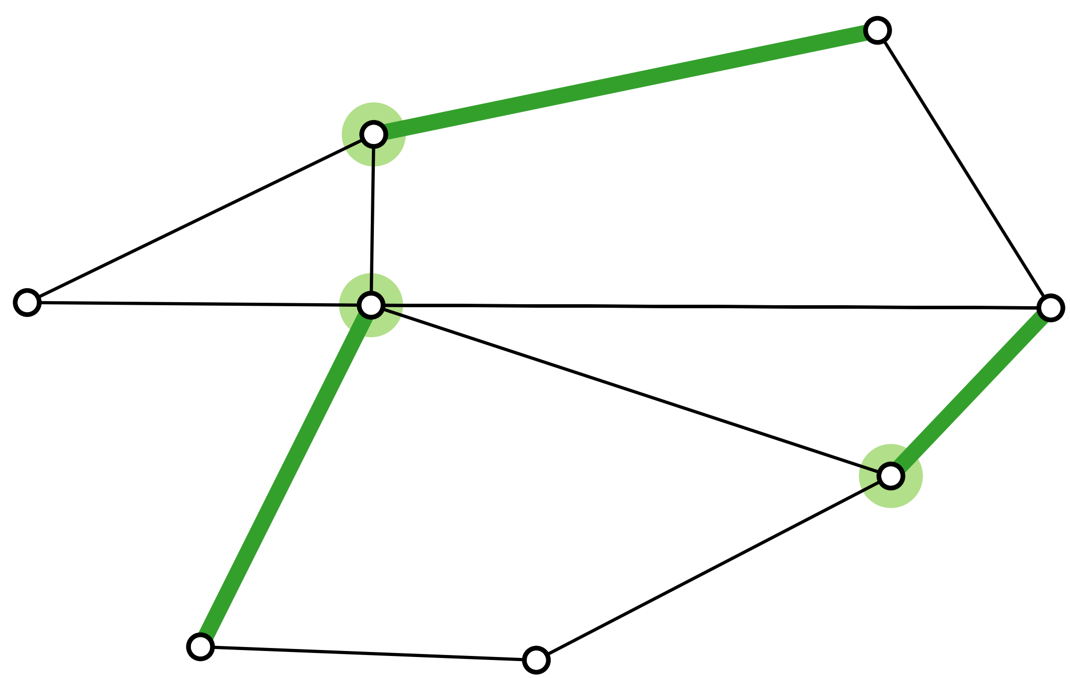
Lower Bound by Matchings

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 $M' \supsetneq M$.

OPT $\geq |M|$

Vertex cover of M



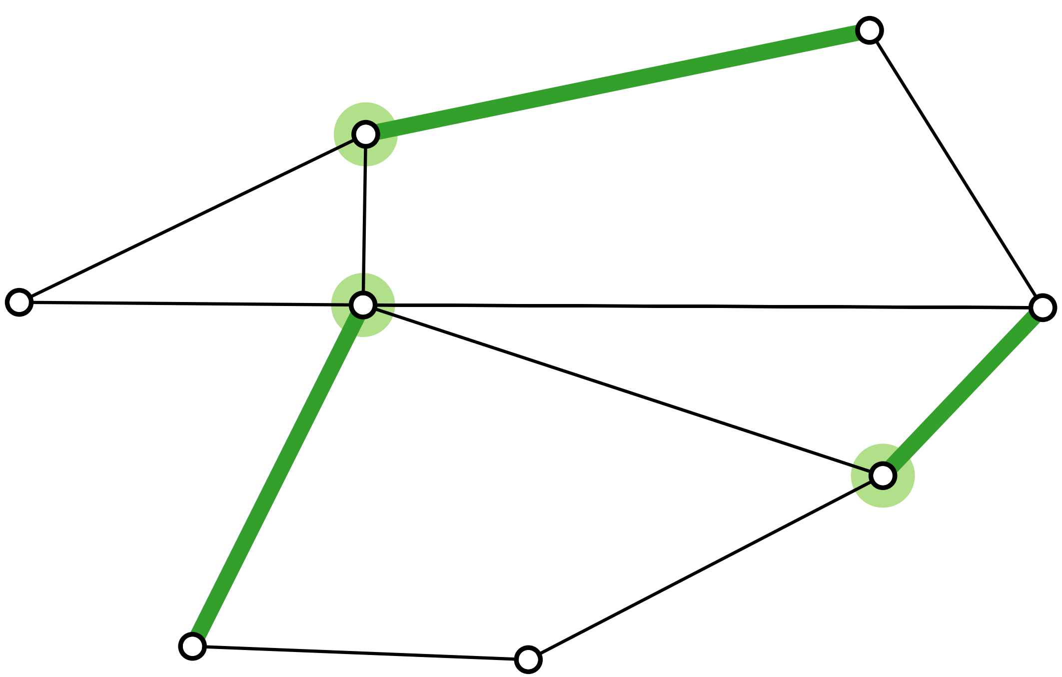
Lower Bound by Matchings

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 $M' \supsetneq M$.

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

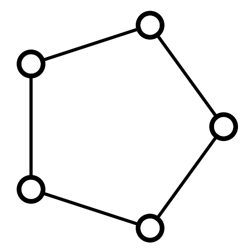
Vertex cover of M



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).

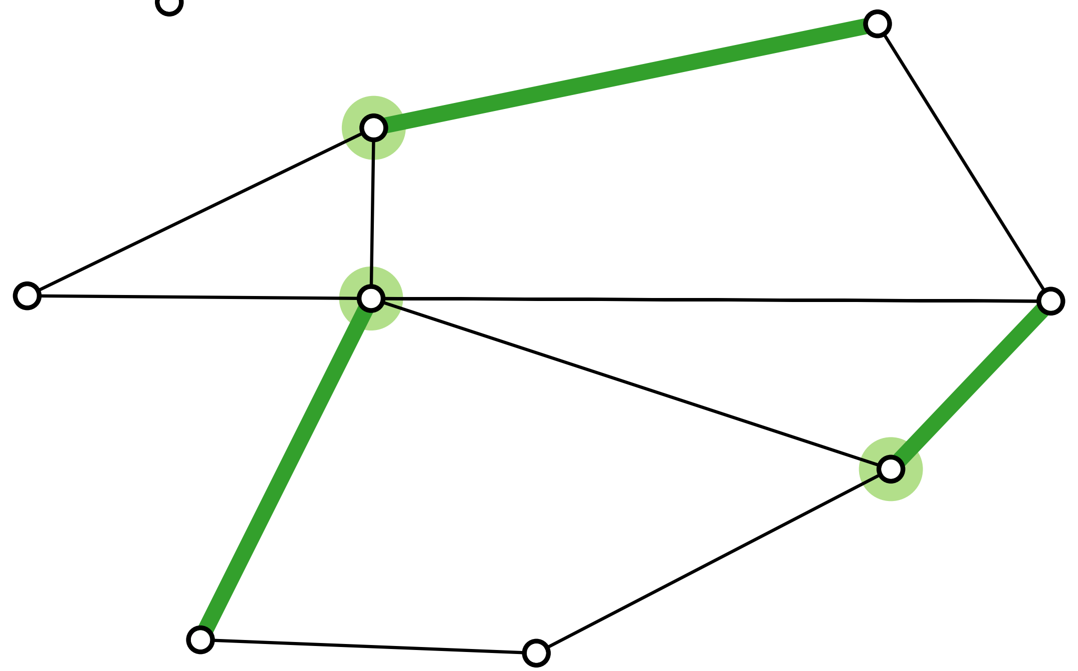
M is **maximal** if there is no matching M' with $M' \supsetneq M$.



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

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

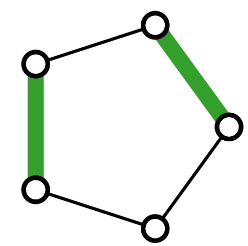
Vertex cover of M



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).

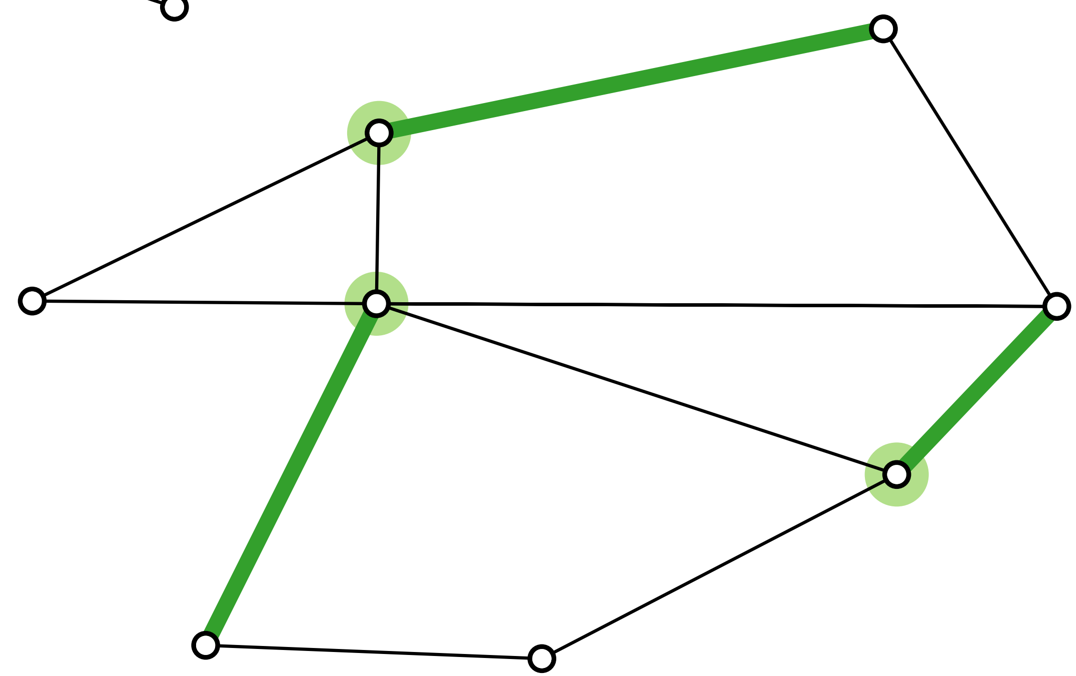
M is **maximal** if there is no matching M' with $M' \supsetneq M$.



OPT $\geq |M|$

OPT $= |M|$?

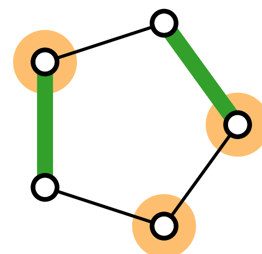
Vertex cover of M



Lower Bound by Matchings

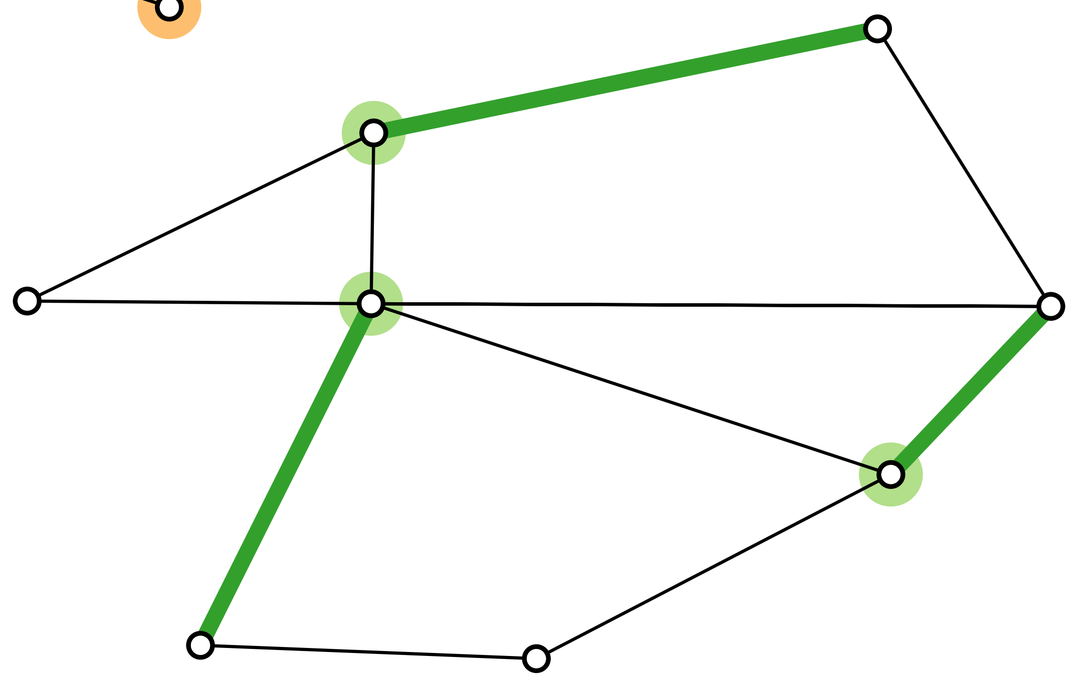
An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a **matching** if no two edges of M are adjacent (i.e., share an end vertex).

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



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

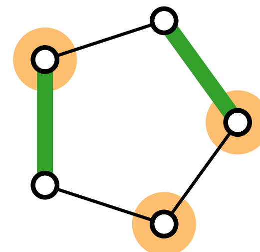
Vertex cover of M



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).

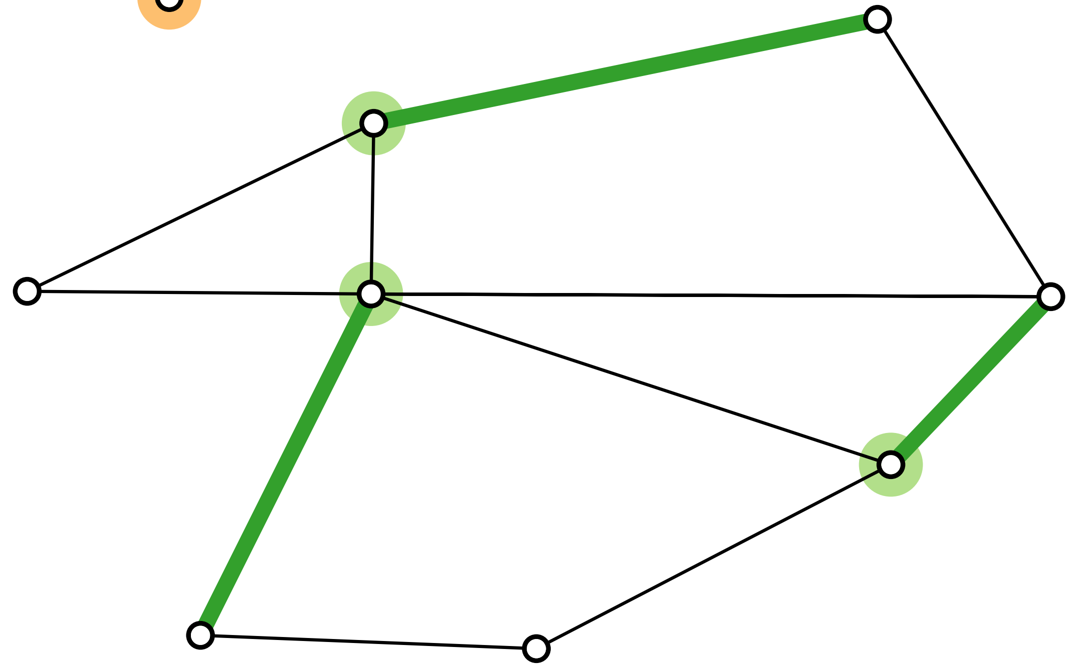
M is **maximal** if there is no matching M' with $M' \supsetneq M$.



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

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

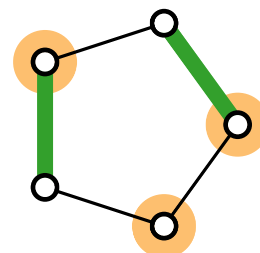
Vertex cover of M



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 $M' \supsetneq M$.

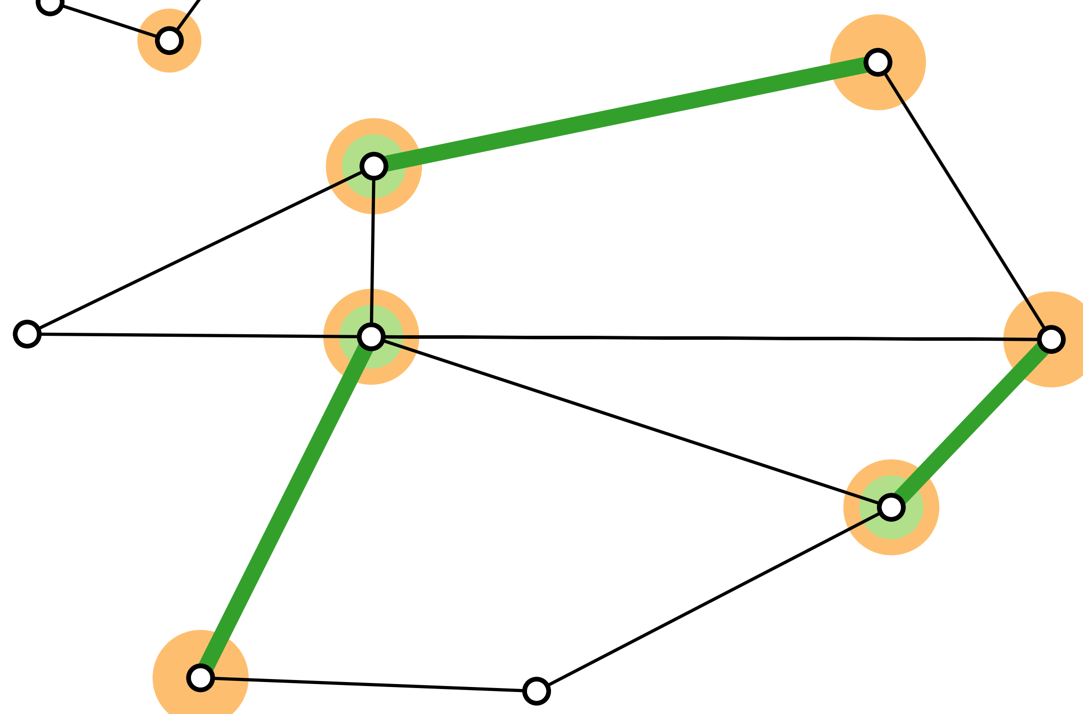


$\mathrm{OPT} \geq |M|$

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

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).

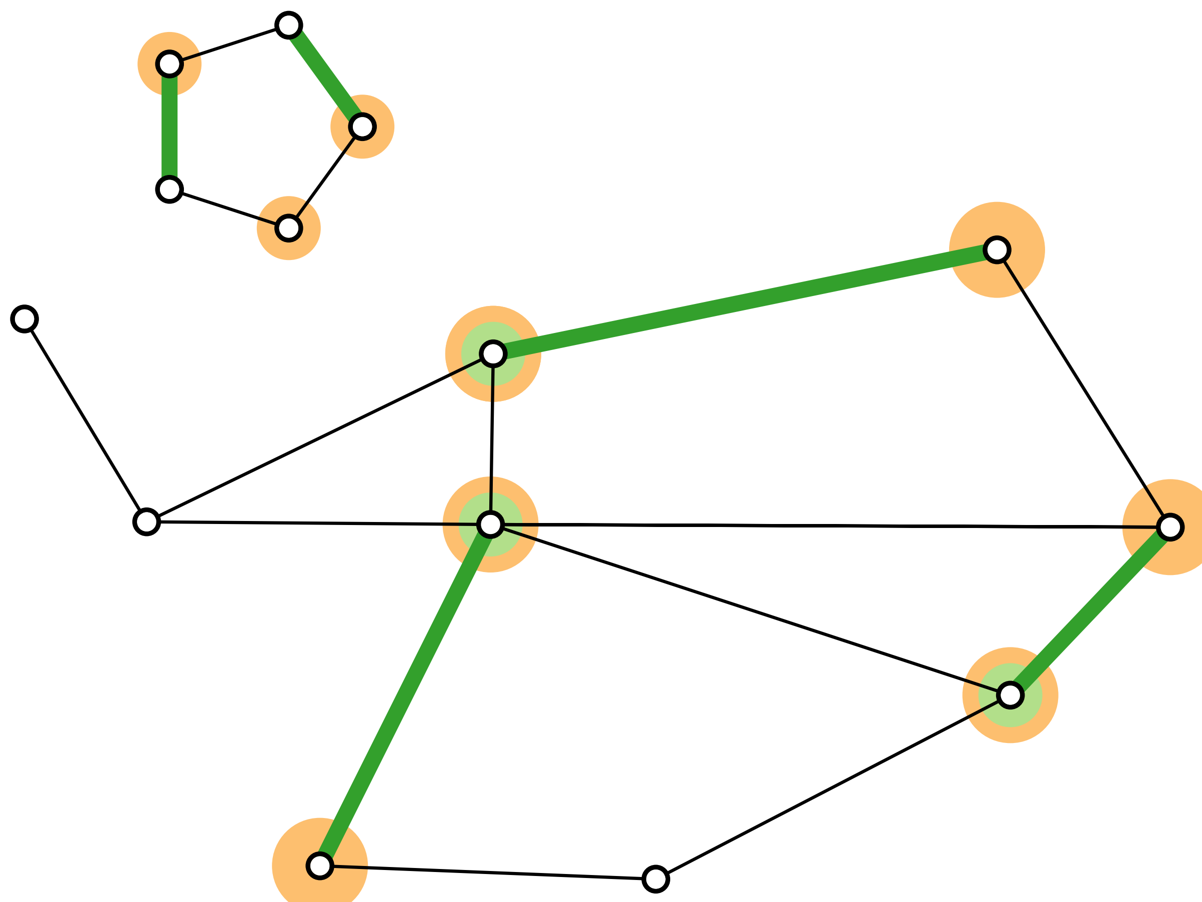
M is **maximal** if there is no matching M' with $M' \supsetneq M$.

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

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

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).

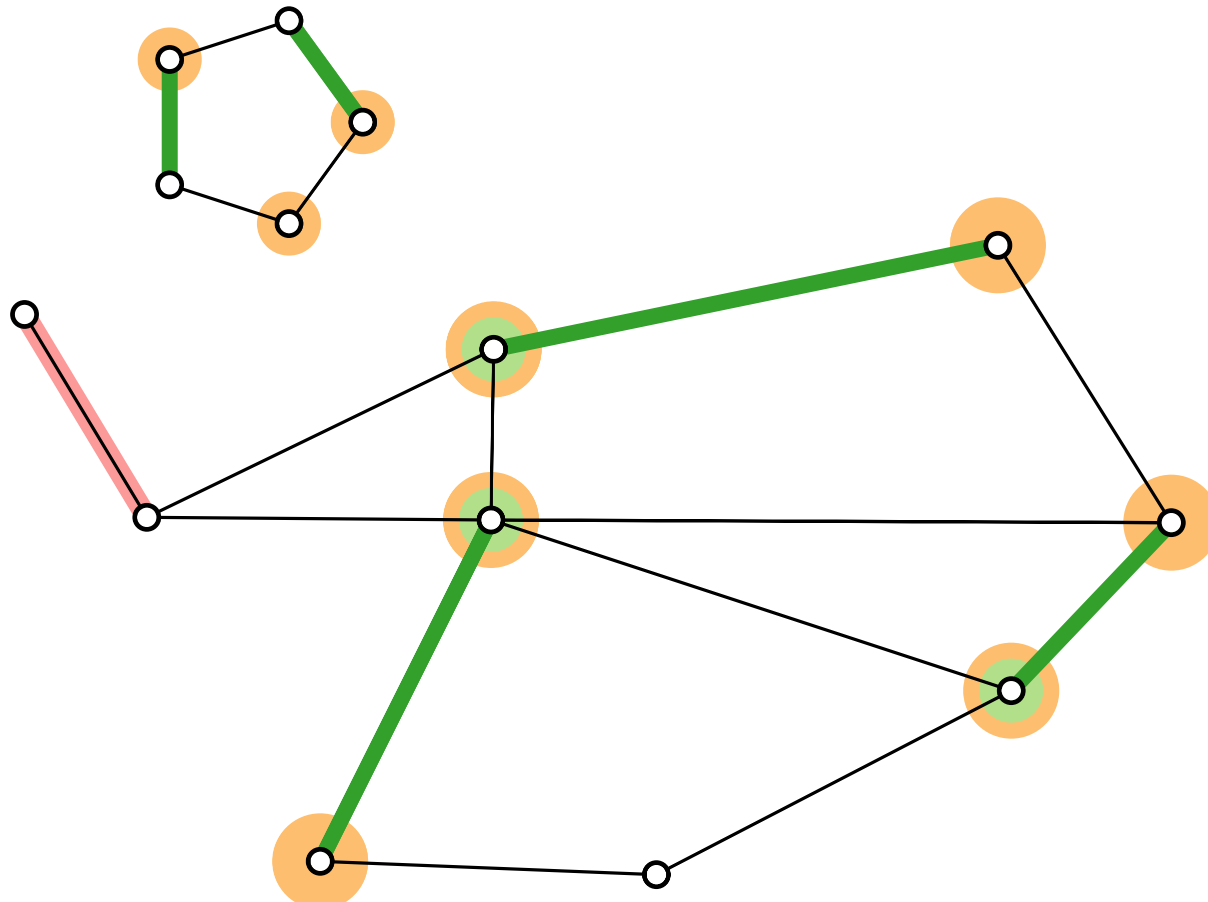
M is **maximal** if there is no matching M' with $M' \supsetneq M$.

OPT $\geq |M|$

~~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).

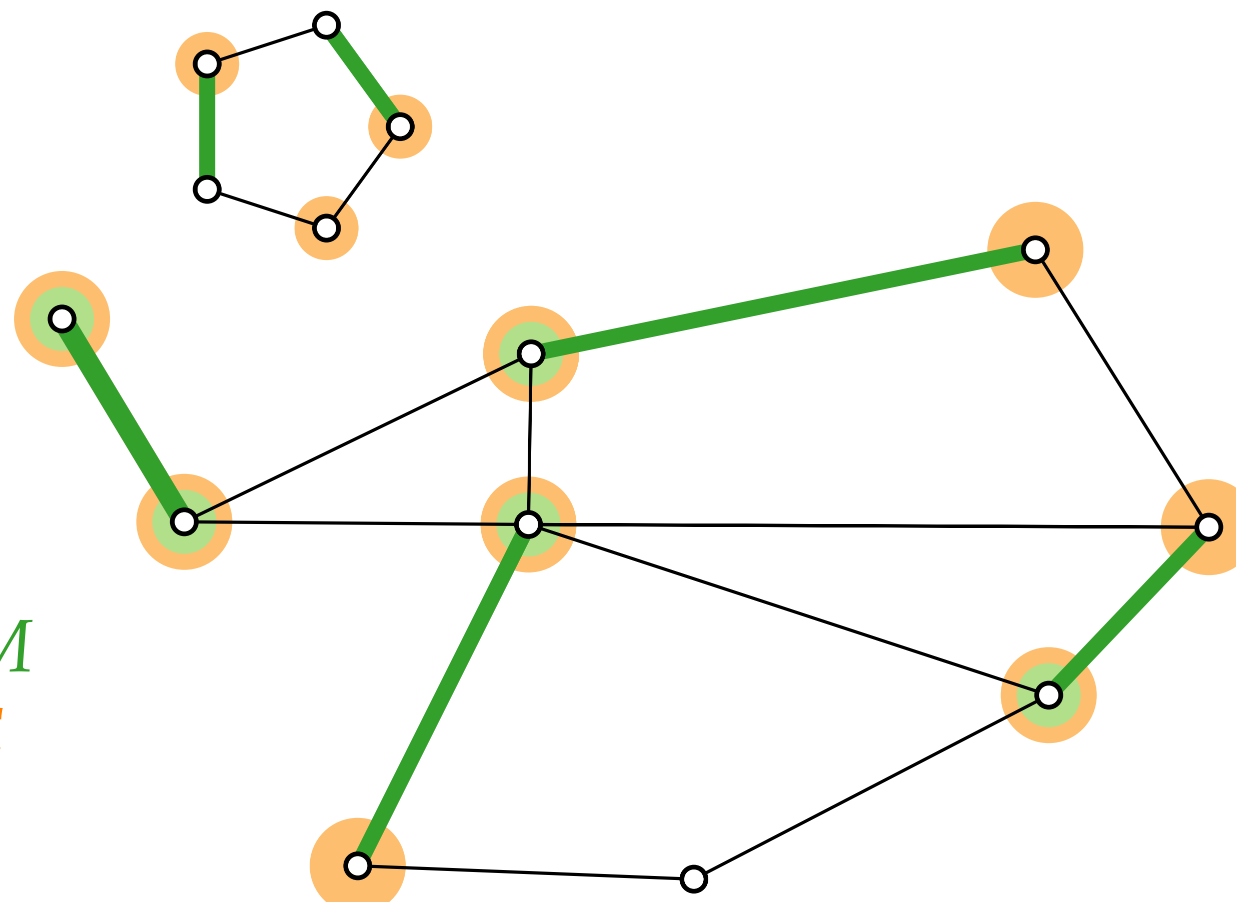
M is **maximal** if there is no matching M' with $M' \supsetneq M$.

OPT $\geq |M|$

~~OPT $= |M|$~~ ?

Vertex cover of M

Vertex cover of E



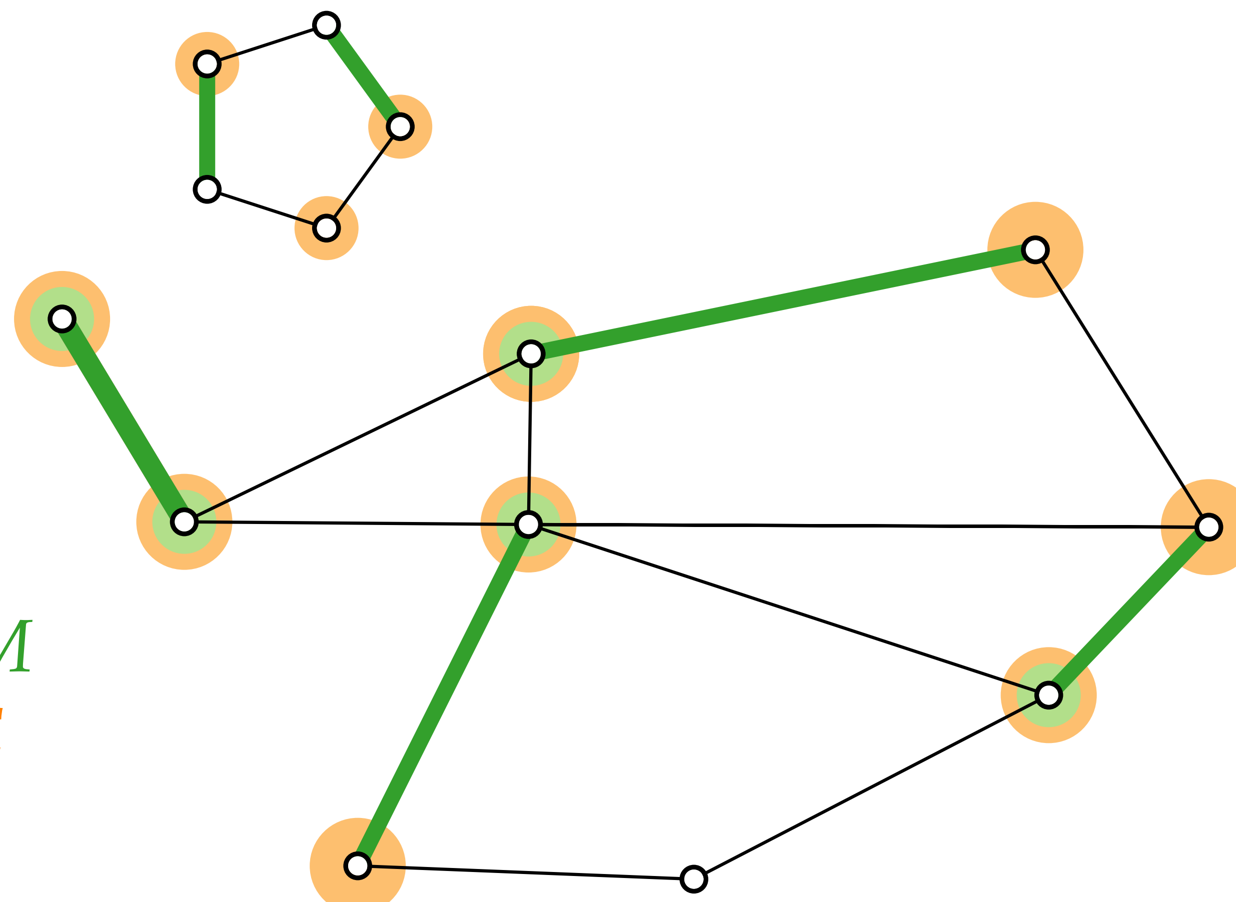
Lower Bound by Matchings

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

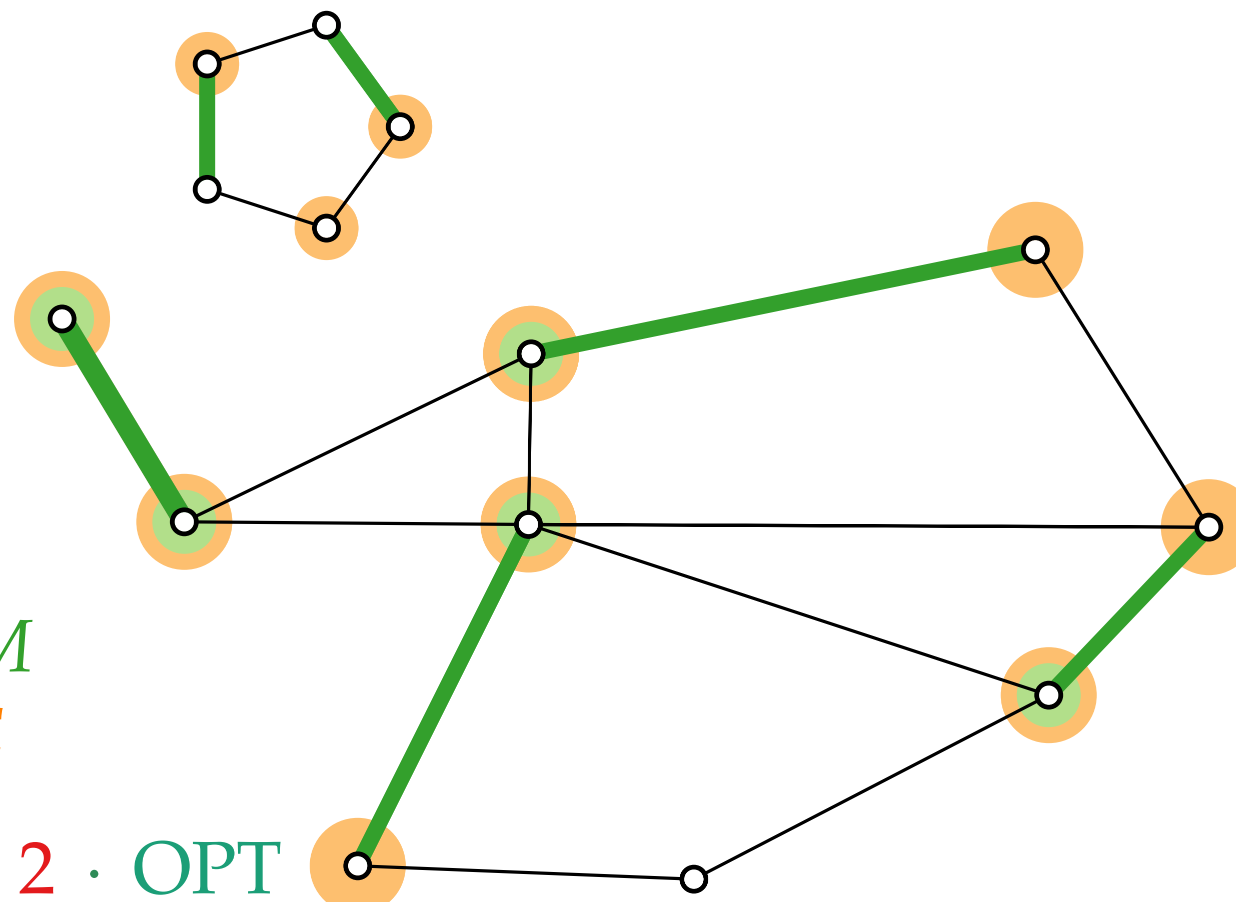
Vertex cover of E

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

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 $M' \supsetneq M$.



OPT $\geq |M|$

~~OPT $= |M|$~~ ?

Vertex cover of M

Vertex cover of E

ALG $= 2 \cdot |M| \leq 2 \cdot$ OPT

Approximation Alg. for VertexCover

Algorithm VertexCover(G)

$M \leftarrow \emptyset$

Approximation Alg. for VertexCover

Algorithm VertexCover(G)

$M \leftarrow \emptyset$

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

Approximation Alg. for VertexCover

Algorithm VertexCover(G)

$M \leftarrow \emptyset$

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

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

Approximation Alg. for VertexCover

Algorithm VertexCover(G)

$M \leftarrow \emptyset$
foreach $e \in E(G)$ **do**
  if e not adjacent to edge in M **then**
    $M \leftarrow M \cup \{e\}$

Approximation Alg. for VertexCover

Algorithm VertexCover(G)

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

Approximation Alg. for VertexCover

Algorithm VertexCover(G)

$M \leftarrow \emptyset$
foreach $e \in E(G)$ **do**
  if e not adjacent to 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 Alg. for VertexCover

Algorithm VertexCover(G)

$M \leftarrow \emptyset$
foreach $e \in E(G)$ **do**
  if e not adjacent to 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.

The best-known approximation factor for VertexCover is

Approximation Alg. for VertexCover

Algorithm VertexCover(G)

$M \leftarrow \emptyset$
foreach $e \in E(G)$ **do**
  if e not adjacent to 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.

The best-known approximation factor for VertexCover is $2 - \Theta(1/\sqrt{\log n})$

Approximation Alg. for VertexCover

Algorithm VertexCover(G)

```
M ← ∅
foreach e ∈ E(G) do
    if e not adjacent to edge in M then
        M ← M ∪ {e}
return { u, v | uv ∈ M }
```

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

The best-known approximation factor for VertexCover is $2 - \Theta(1/\sqrt{\log n})$

VertexCover cannot be approximated within factor 1.3606 (unless P=NP)

Approximation Alg. for VertexCover

Algorithm VertexCover(G)

$M \leftarrow \emptyset$
foreach $e \in E(G)$ **do**
  if e not adjacent to 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.

The best-known approximation factor for VertexCover is $2 - \Theta(1/\sqrt{\log n})$

VertexCover cannot be approximated within factor 1.3606 (unless P=NP)

VertexCover cannot be approximated within factor $2 - \Theta(1)$, if "Unique Game Conjecture" holds.