

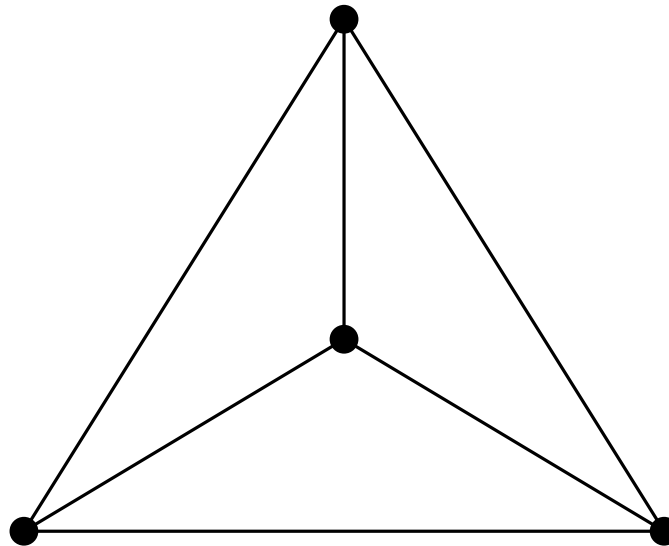
Approximation Algorithms

Lecture 3: STEINERTREE and MULTIWAYCUT

Part I: STEINERTREE

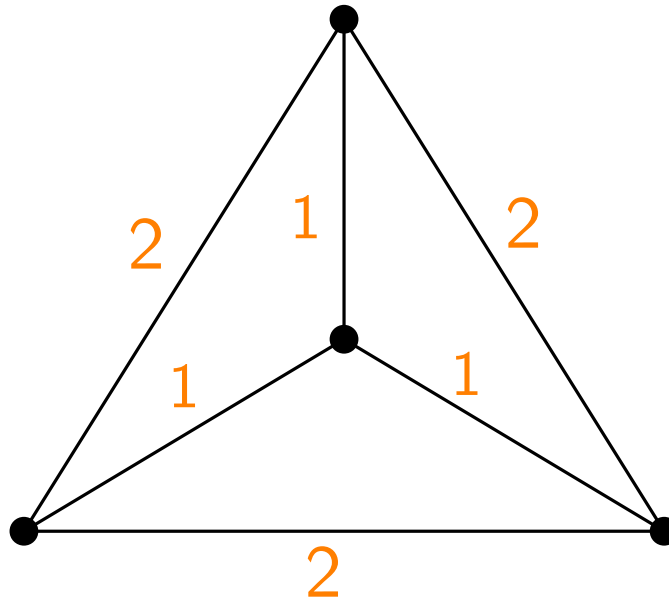
STEINERTREE

Given: A graph G



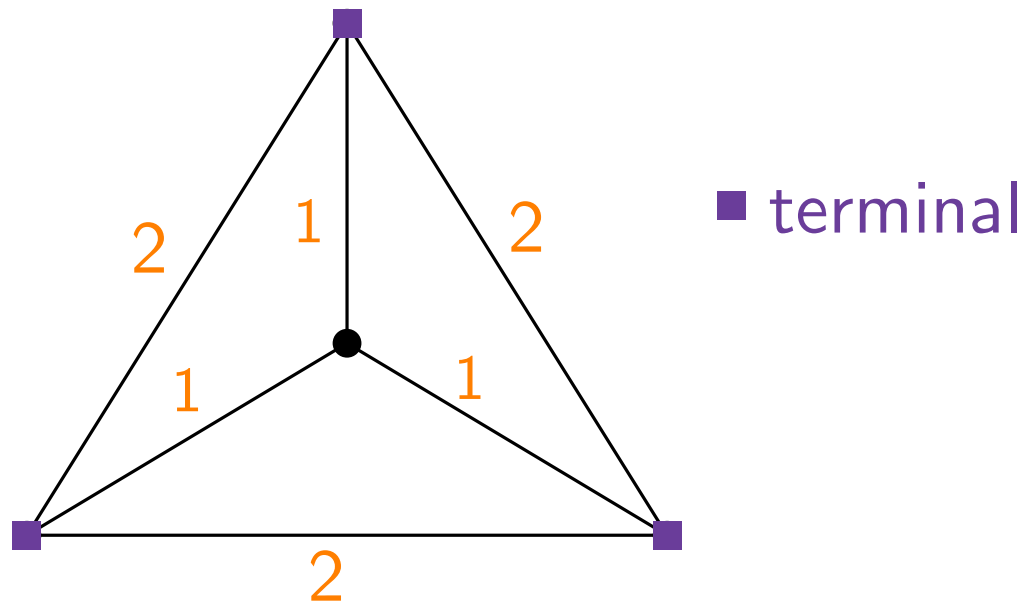
STEINERTREE

Given: A graph G with edge weights $c: E(G) \rightarrow \mathbb{Q}^+$



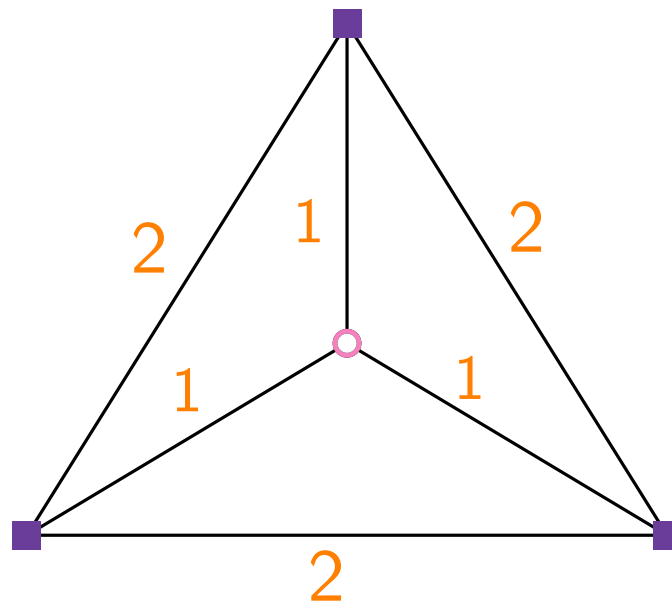
STEINERTREE

Given: A graph G with **edge weights** $c: E(G) \rightarrow \mathbb{Q}^+$
and a partition of $V(G)$ into a set T of **terminals**



STEINERTREE

Given: A graph G with **edge weights** $c: E(G) \rightarrow \mathbb{Q}^+$
and a partition of $V(G)$ into a set T of **terminals**
and a set S of **Steiner vertices**.



■ terminal

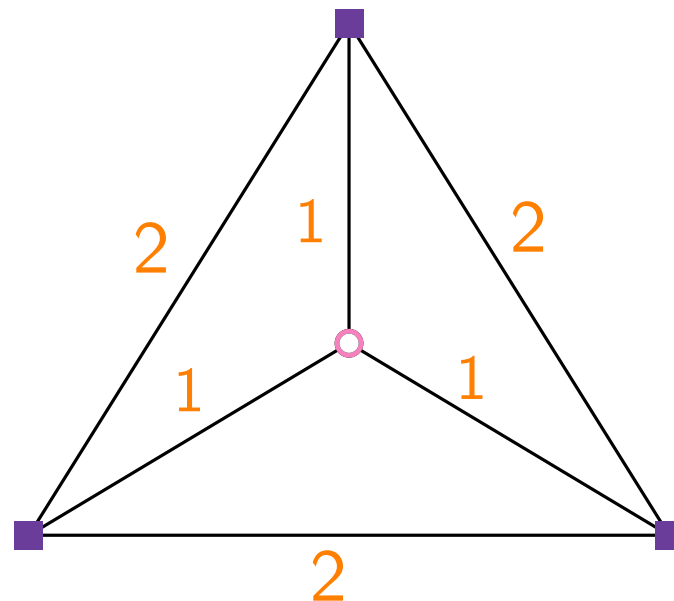
○ Steiner vertex

STEINERTREE

Given: A graph G with **edge weights** $c: E(G) \rightarrow \mathbb{Q}^+$ and a partition of $V(G)$ into a set T of **terminals** and a set S of **Steiner vertices**.

Find: A **subtree** B of G that

- contains all **terminals** (i.e., $T \subseteq V(B)$) and



■ terminal

○ Steiner vertex

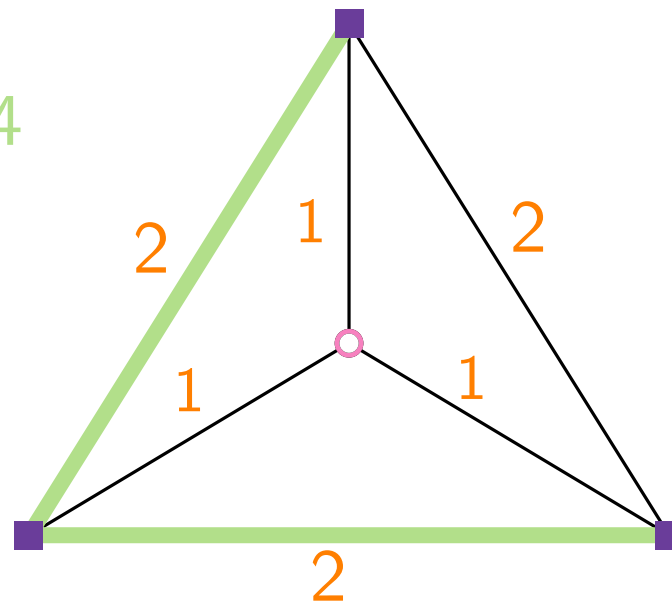
STEINERTREE

Given: A graph G with edge weights $c: E(G) \rightarrow \mathbb{Q}^+$ and a partition of $V(G)$ into a set T of **terminals** and a set S of **Steiner vertices**.

Find: A subtree B of G that

- contains all **terminals** (i.e., $T \subseteq V(B)$) and

valid solution with cost 4



■ terminal

○ Steiner vertex

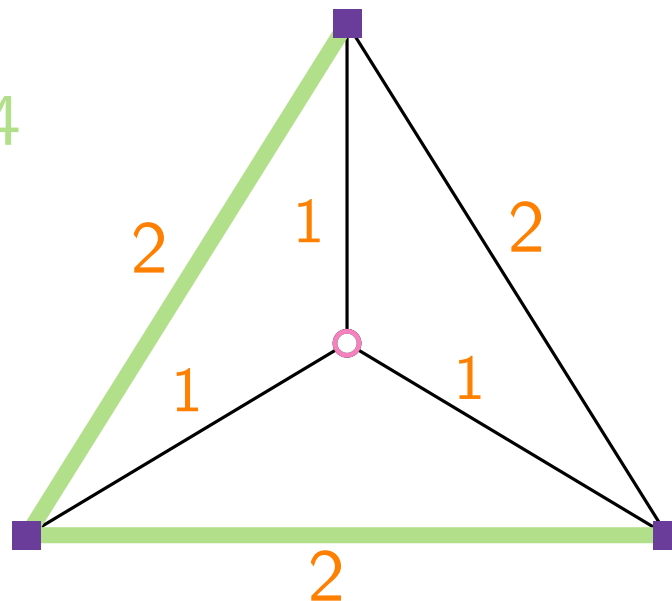
STEINERTREE

Given: A graph G with **edge weights** $c: E(G) \rightarrow \mathbb{Q}^+$ and a partition of $V(G)$ into a set T of **terminals** and a set S of **Steiner vertices**.

Find: A **subtree** B of G that

- contains all **terminals** (i.e., $T \subseteq V(B)$) and
- has **minimum cost** $c(B) := \sum_{e \in E(B)} c(e)$ among all subtrees with this property.

valid solution with cost 4



■ terminal

○ Steiner vertex

STEINERTREE

Given: A graph G with **edge weights** $c: E(G) \rightarrow \mathbb{Q}^+$ and a partition of $V(G)$ into a set T of **terminals** and a set S of **Steiner vertices**.

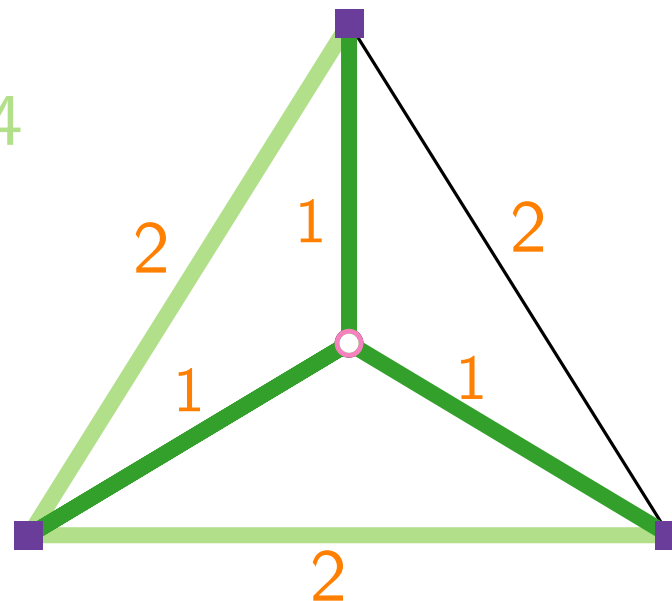
Find: A **subtree** B of G that

- contains all **terminals** (i.e., $T \subseteq V(B)$) and
- has **minimum cost** $c(B) := \sum_{e \in E(B)} c(e)$ among all subtrees with this property.

valid solution with cost 4

optimum solution

with cost 3



■ terminal

○ Steiner vertex

METRICSTEINERTREE

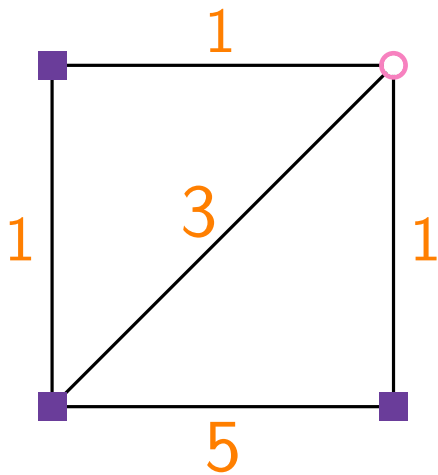
Restriction of STEINERTREE where the graph G is complete and the cost function is **metric**

METRICSTEINERTREE

Restriction of STEINERTREE where the graph G is complete and the cost function is **metric**, i.e., for every triple u, v, w of vertices, we have $c(u, w) \leq c(u, v) + c(v, w)$.

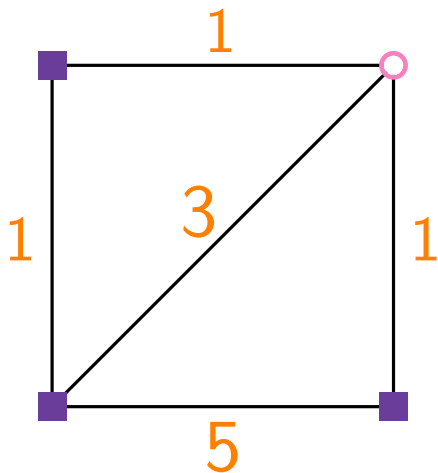
METRICSTEINERTREE

Restriction of STEINERTREE where the graph G is complete and the cost function is **metric**, i.e., for every triple u, v, w of vertices, we have $c(u, w) \leq c(u, v) + c(v, w)$.



METRICSTEINERTREE

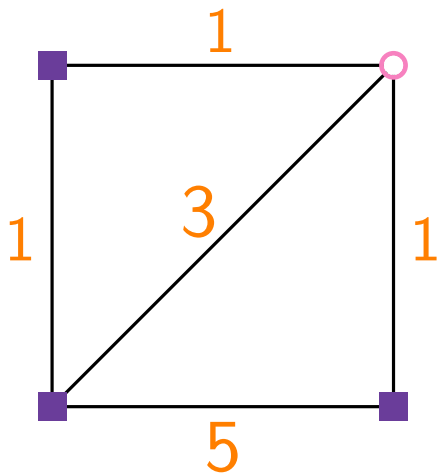
Restriction of STEINERTREE where the graph G is complete and the cost function is **metric**, i.e., for every triple u, v, w of vertices, we have $c(u, w) \leq c(u, v) + c(v, w)$.



– not complete

METRICSTEINERTREE

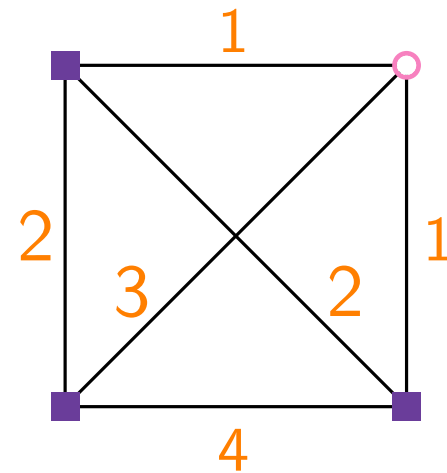
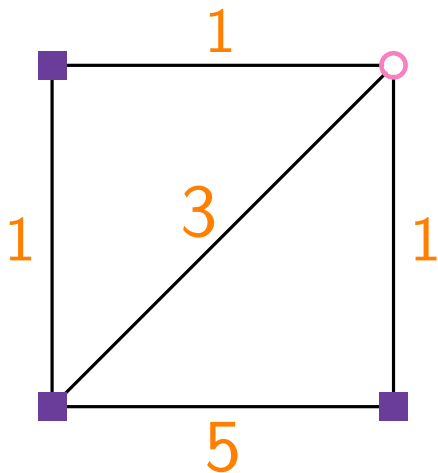
Restriction of STEINERTREE where the graph G is complete and the cost function is **metric**, i.e., for every triple u, v, w of vertices, we have $c(u, w) \leq c(u, v) + c(v, w)$.



- not complete
- not metric

METRICSTEINERTREE

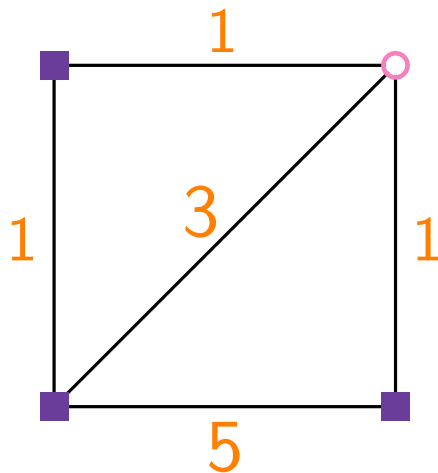
Restriction of STEINERTREE where the graph G is complete and the cost function is **metric**, i.e., for every triple u, v, w of vertices, we have $c(u, w) \leq c(u, v) + c(v, w)$.



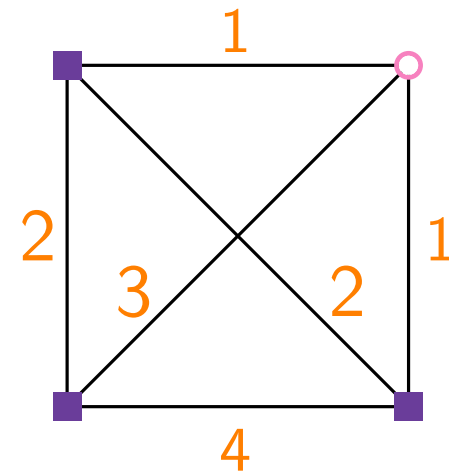
- not complete
- not metric

METRICSTEINERTREE

Restriction of STEINERTREE where the graph G is complete and the cost function is **metric**, i.e., for every triple u, v, w of vertices, we have $c(u, w) \leq c(u, v) + c(v, w)$.



- not complete
- not metric



- complete
- metric

Approximation Algorithms

Lecture 3:

STEINERTREE and MULTIWAYCUT

Part II:

Approximation-Preserving Reduction

Approximation-Preserving Reduction

Let Π_1, Π_2 be minimization problems.

Approximation-Preserving Reduction

Let Π_1, Π_2 be minimization problems.

problems

Π_1

Π_2

Approximation-Preserving Reduction

Let Π_1, Π_2 be minimization problems. An **approximation-preserving reduction** from Π_1 to Π_2 is a tuple (f, g) of poly-time computable functions with the following properties.

- For each instance I_1 of Π_1 ,

problems

Π_1

Π_2

Approximation-Preserving Reduction

Let Π_1, Π_2 be minimization problems. An **approximation-preserving reduction** from Π_1 to Π_2 is a tuple (f, g) of poly-time computable functions with the following properties.

- For each instance I_1 of Π_1 ,
 $I_2 = f(I_1)$ is an instance of Π_2 with $\text{OPT}_{\Pi_2}(I_2) \leq \text{OPT}_{\Pi_1}(I_1)$.

problems

Π_1

Π_2

Approximation-Preserving Reduction

Let Π_1, Π_2 be minimization problems. An **approximation-preserving reduction** from Π_1 to Π_2 is a tuple (f, g) of poly-time computable functions with the following properties.

- For each instance I_1 of Π_1 ,
 $I_2 = f(I_1)$ is an instance of Π_2 with $\text{OPT}_{\Pi_2}(I_2) \leq \text{OPT}_{\Pi_1}(I_1)$.

problems

Π_1

Π_2

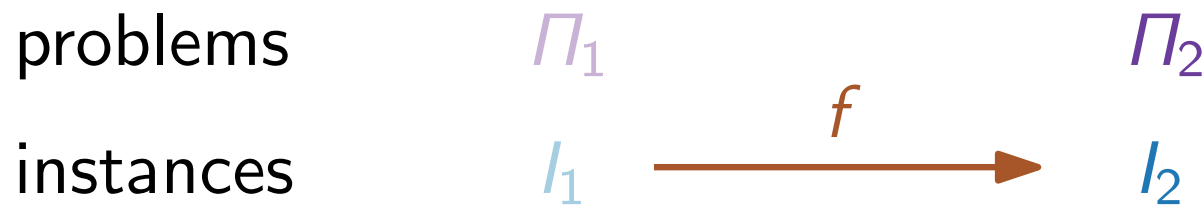
instances

I_1

Approximation-Preserving Reduction

Let Π_1, Π_2 be minimization problems. An **approximation-preserving reduction** from Π_1 to Π_2 is a tuple (f, g) of poly-time computable functions with the following properties.

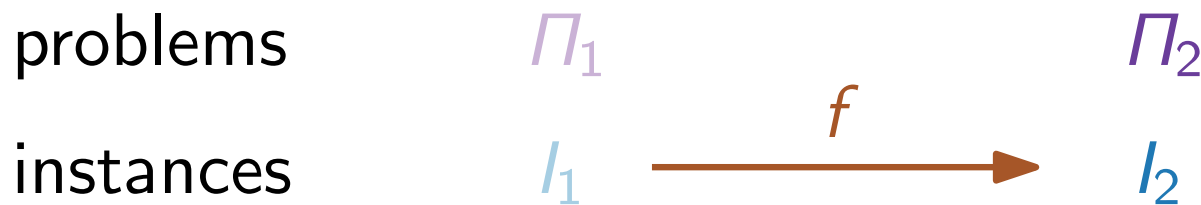
- For each instance I_1 of Π_1 ,
 $I_2 = f(I_1)$ is an instance of Π_2 with $\text{OPT}_{\Pi_2}(I_2) \leq \text{OPT}_{\Pi_1}(I_1)$.



Approximation-Preserving Reduction

Let Π_1, Π_2 be minimization problems. An **approximation-preserving reduction** from Π_1 to Π_2 is a tuple (f, g) of poly-time computable functions with the following properties.

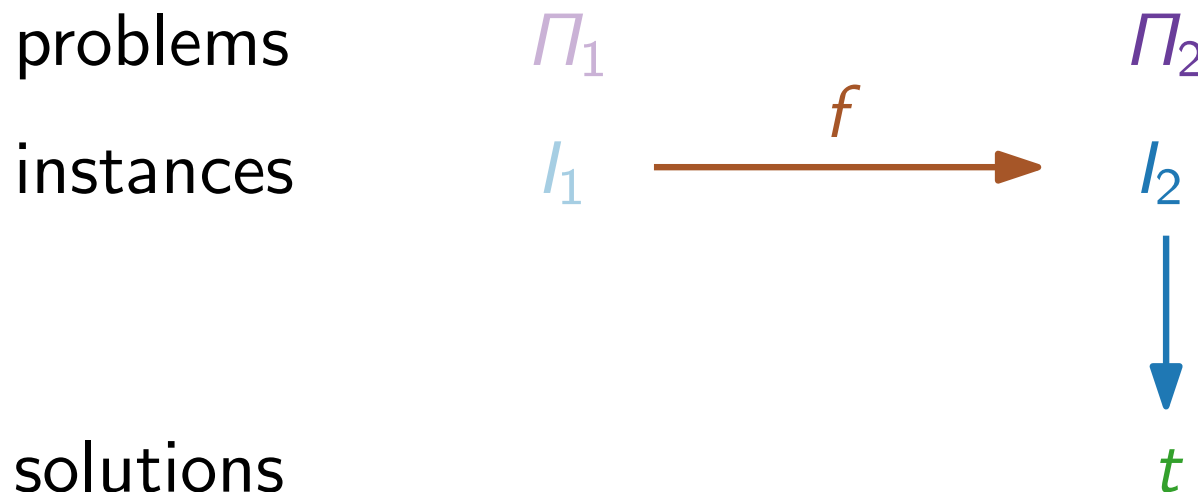
- For each instance l_1 of Π_1 ,
 $l_2 = f(l_1)$ is an instance of Π_2 with $\text{OPT}_{\Pi_2}(l_2) \leq \text{OPT}_{\Pi_1}(l_1)$.
- For each feasible solution t of l_2 ,
 $s = g(l_1, t)$ is a feasible sol. of l_1 with $\text{obj}_{\Pi_1}(l_1, s) \leq \text{obj}_{\Pi_2}(l_2, t)$.



Approximation-Preserving Reduction

Let Π_1, Π_2 be minimization problems. An **approximation-preserving reduction** from Π_1 to Π_2 is a tuple (f, g) of poly-time computable functions with the following properties.

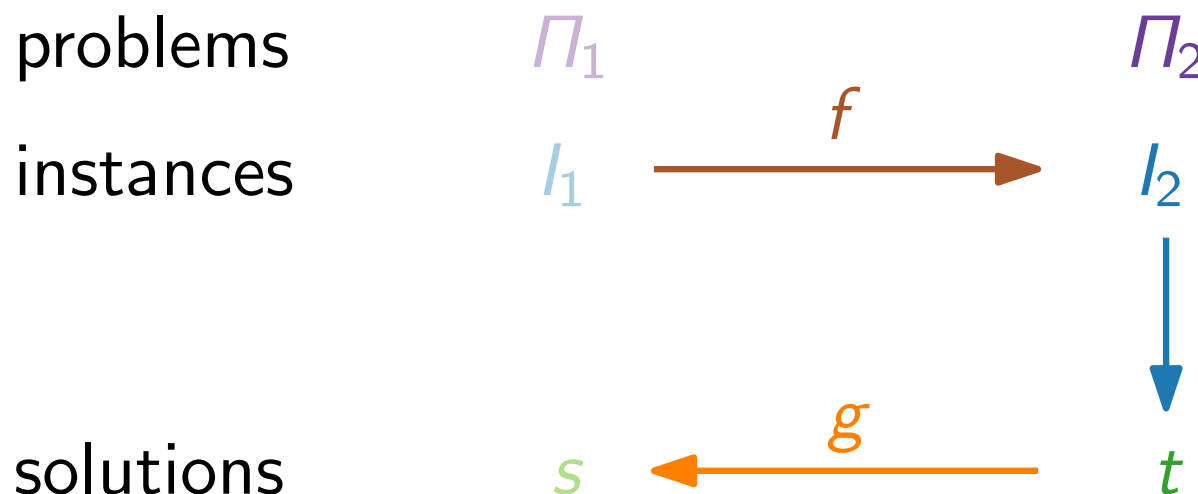
- For each instance l_1 of Π_1 ,
 $l_2 = f(l_1)$ is an instance of Π_2 with $\text{OPT}_{\Pi_2}(l_2) \leq \text{OPT}_{\Pi_1}(l_1)$.
- For each feasible solution t of l_2 ,
 $s = g(l_1, t)$ is a feasible sol. of l_1 with $\text{obj}_{\Pi_1}(l_1, s) \leq \text{obj}_{\Pi_2}(l_2, t)$.



Approximation-Preserving Reduction

Let Π_1, Π_2 be minimization problems. An **approximation-preserving reduction** from Π_1 to Π_2 is a tuple (f, g) of poly-time computable functions with the following properties.

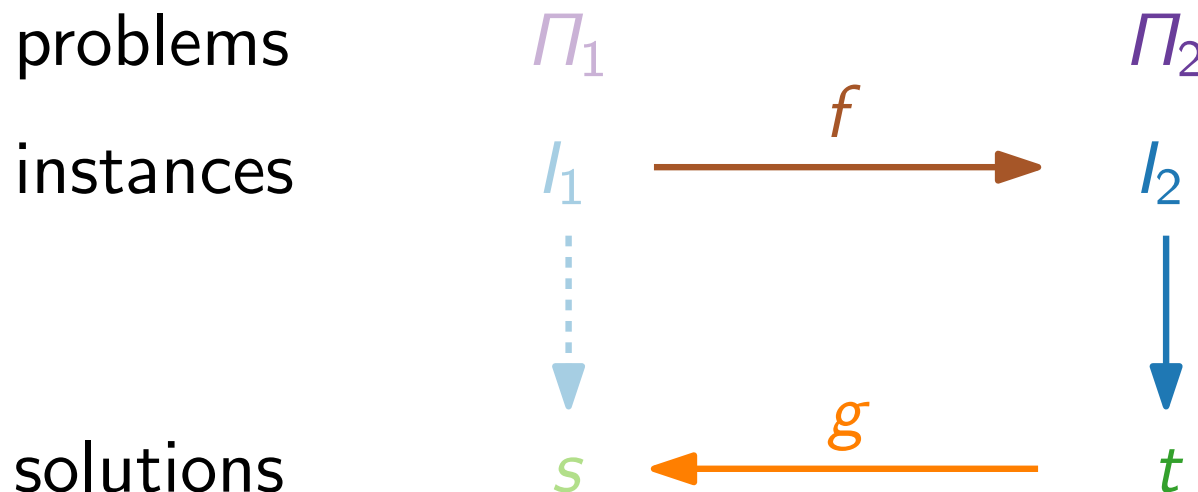
- For each instance I_1 of Π_1 ,
 $I_2 = f(I_1)$ is an instance of Π_2 with $\text{OPT}_{\Pi_2}(I_2) \leq \text{OPT}_{\Pi_1}(I_1)$.
- For each feasible solution t of I_2 ,
 $s = g(I_1, t)$ is a feasible sol. of I_1 with $\text{obj}_{\Pi_1}(I_1, s) \leq \text{obj}_{\Pi_2}(I_2, t)$.



Approximation-Preserving Reduction

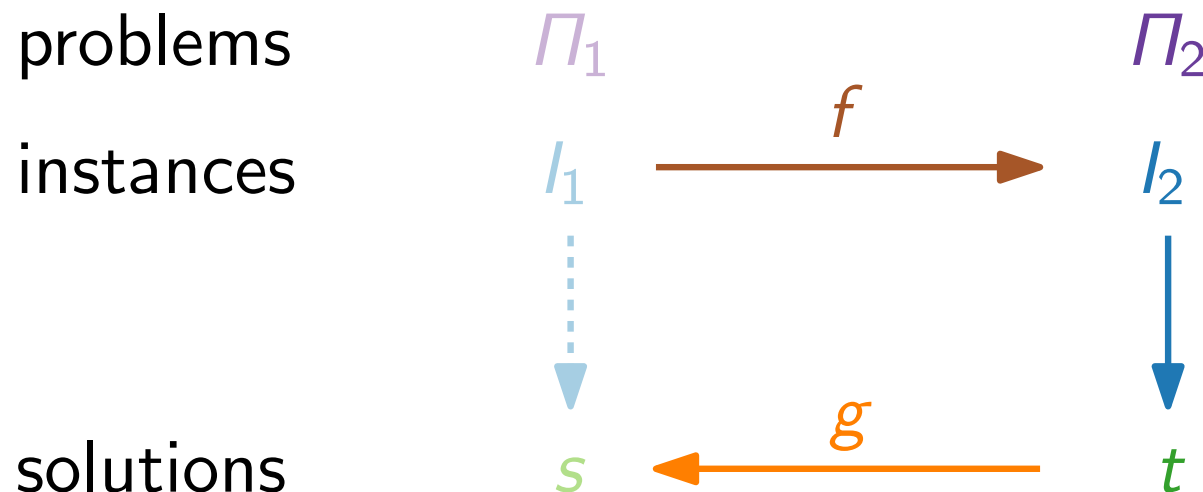
Let Π_1, Π_2 be minimization problems. An **approximation-preserving reduction** from Π_1 to Π_2 is a tuple (f, g) of poly-time computable functions with the following properties.

- For each instance l_1 of Π_1 ,
 $l_2 = f(l_1)$ is an instance of Π_2 with $\text{OPT}_{\Pi_2}(l_2) \leq \text{OPT}_{\Pi_1}(l_1)$.
- For each feasible solution t of l_2 ,
 $s = g(l_1, t)$ is a feasible sol. of l_1 with $\text{obj}_{\Pi_1}(l_1, s) \leq \text{obj}_{\Pi_2}(l_2, t)$.



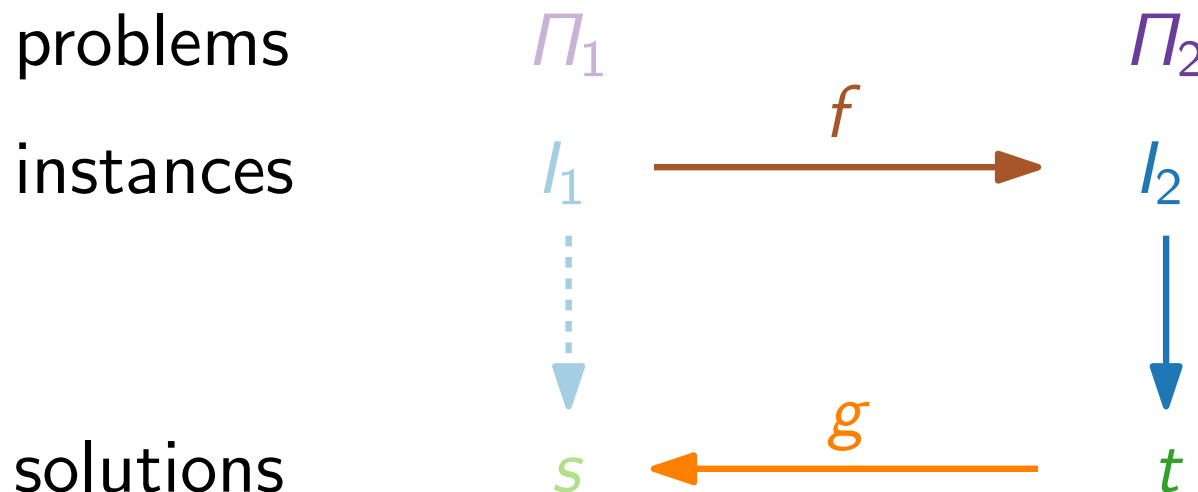
Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2 .



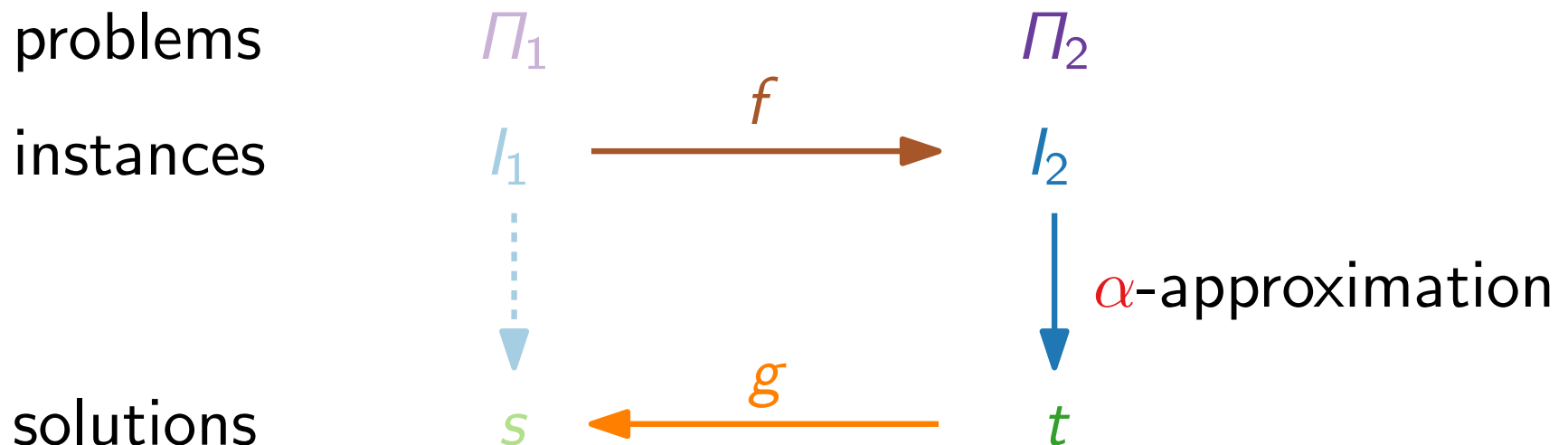
Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2 . If there is a factor- α approximation algorithm for Π_2 , then there is a factor-? approximation algorithm for Π_1 .



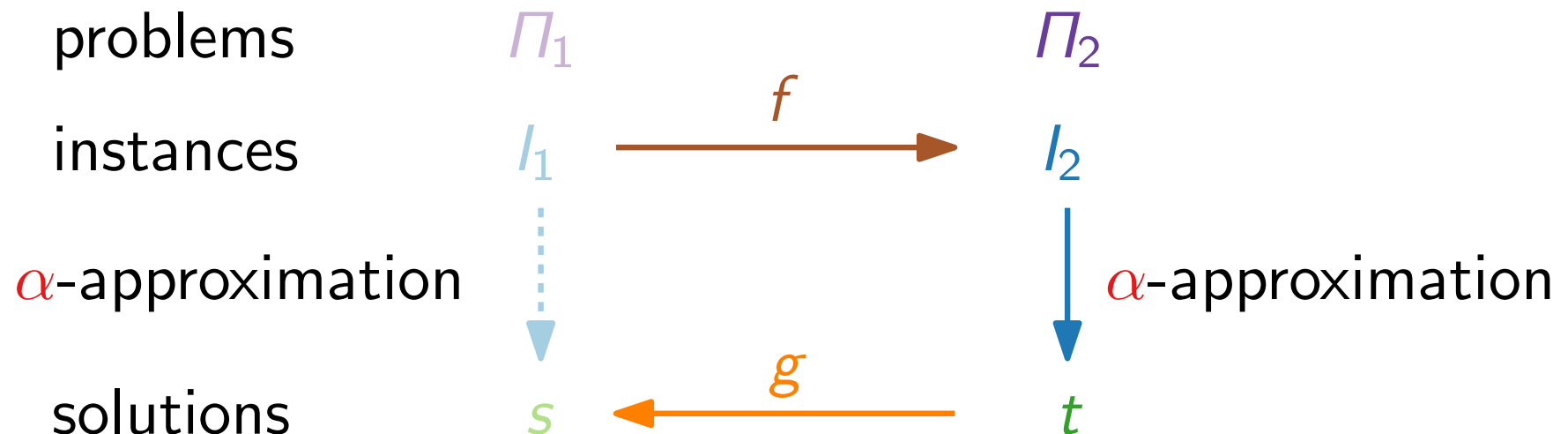
Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2 . If there is a factor- α approximation algorithm for Π_2 , then there is a factor-? approximation algorithm for Π_1 .



Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2 . If there is a factor- α approximation algorithm for Π_2 , then there is a factor- α approximation algorithm for Π_1 .

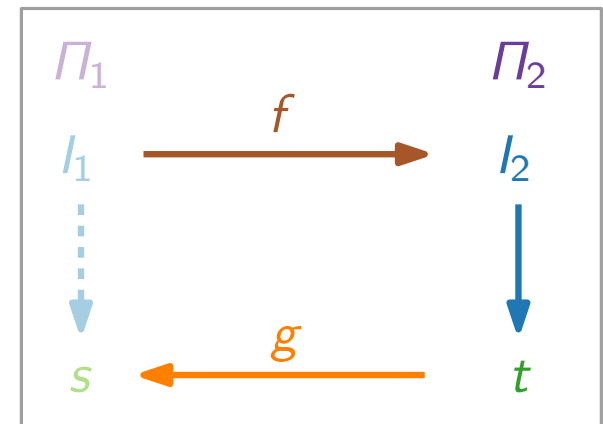


Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2 . If there is a factor- α approximation algorithm for Π_2 , then there is a factor- α approximation algorithm for Π_1 .

Proof.

Let A be a factor- α approx. alg. for Π_2 .



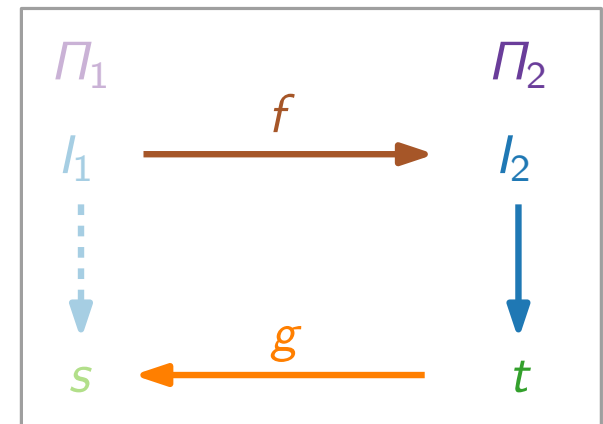
Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2 . If there is a factor- α approximation algorithm for Π_2 , then there is a factor- α approximation algorithm for Π_1 .

Proof.

Let A be a factor- α approx. alg. for Π_2 .

Let l_1 be an instance of Π_1 .



Approximation-Preserving Reduction

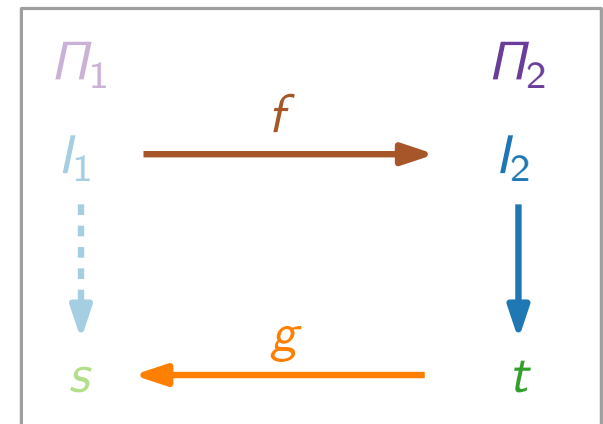
Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2 . If there is a factor- α approximation algorithm for Π_2 , then there is a factor- α approximation algorithm for Π_1 .

Proof.

Let A be a factor- α approx. alg. for Π_2 .

Let l_1 be an instance of Π_1 .

Set $l_2 :=$ $t :=$ and $s :=$



Approximation-Preserving Reduction

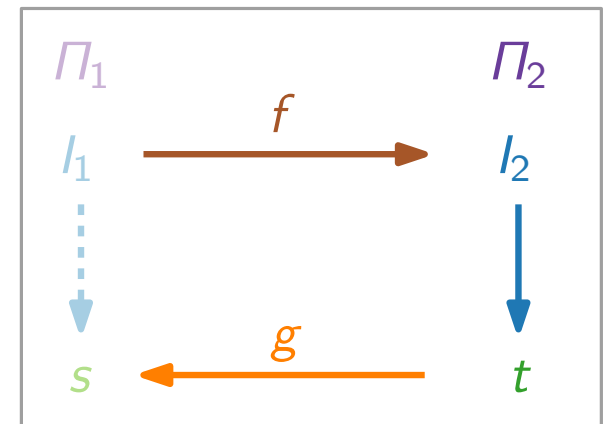
Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2 . If there is a factor- α approximation algorithm for Π_2 , then there is a factor- α approximation algorithm for Π_1 .

Proof.

Let A be a factor- α approx. alg. for Π_2 .

Let l_1 be an instance of Π_1 .

Set $l_2 := f(l_1)$, $t :=$ and $s :=$



Approximation-Preserving Reduction

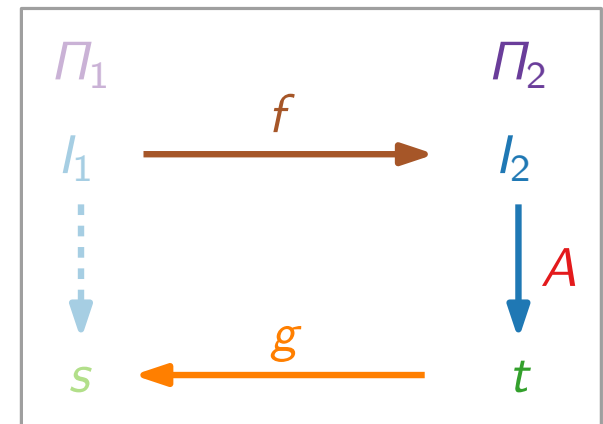
Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2 . If there is a factor- α approximation algorithm for Π_2 , then there is a factor- α approximation algorithm for Π_1 .

Proof.

Let A be a factor- α approx. alg. for Π_2 .

Let l_1 be an instance of Π_1 .

Set $l_2 := f(l_1)$, $t := A(l_2)$ and $s :=$



Approximation-Preserving Reduction

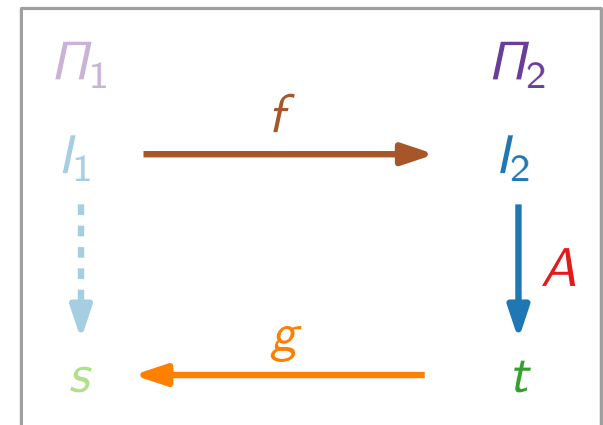
Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2 . If there is a factor- α approximation algorithm for Π_2 , then there is a factor- α approximation algorithm for Π_1 .

Proof.

Let A be a factor- α approx. alg. for Π_2 .

Let l_1 be an instance of Π_1 .

Set $l_2 := f(l_1)$, $t := A(l_2)$ and $s := g(l_1, t)$.



Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2 . If there is a factor- α approximation algorithm for Π_2 , then there is a factor- α approximation algorithm for Π_1 .

Proof.

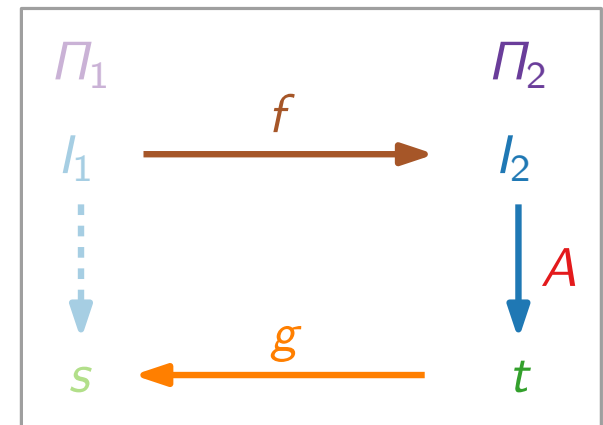
Let A be a factor- α approx. alg. for Π_2 .

Let l_1 be an instance of Π_1 .

Set $l_2 := f(l_1)$, $t := A(l_2)$ and $s := g(l_1, t)$.

Then:

$$\text{obj}_{\Pi_1}(l_1, s) \leq$$



Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2 . If there is a factor- α approximation algorithm for Π_2 , then there is a factor- α approximation algorithm for Π_1 .

Proof.

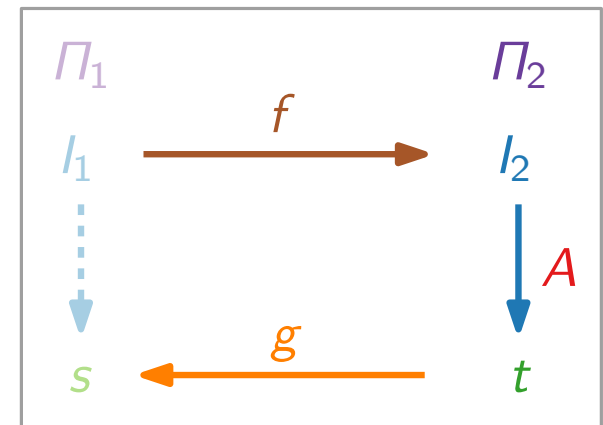
Let A be a factor- α approx. alg. for Π_2 .

Let l_1 be an instance of Π_1 .

Set $l_2 := f(l_1)$, $t := A(l_2)$ and $s := g(l_1, t)$.

Then:

$$\text{obj}_{\Pi_1}(l_1, s) \leq \text{obj}_{\Pi_2}(l_2, t) \leq$$



Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2 . If there is a factor- α approximation algorithm for Π_2 , then there is a factor- α approximation algorithm for Π_1 .

Proof.

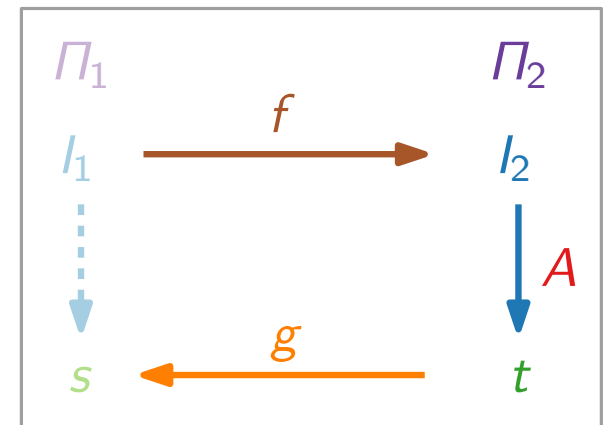
Let A be a factor- α approx. alg. for Π_2 .

Let l_1 be an instance of Π_1 .

Set $l_2 := f(l_1)$, $t := A(l_2)$ and $s := g(l_1, t)$.

Then:

$$\text{obj}_{\Pi_1}(l_1, s) \leq \text{obj}_{\Pi_2}(l_2, t) \leq \alpha \cdot \text{OPT}_{\Pi_2}(l_2) \leq$$



Approximation-Preserving Reduction

Theorem. Let Π_1, Π_2 be minimization problems with an approximation-preserving reduction (f, g) from Π_1 to Π_2 . If there is a factor- α approximation algorithm for Π_2 , then there is a factor- α approximation algorithm for Π_1 .

Proof.

Let A be a factor- α approx. alg. for Π_2 .

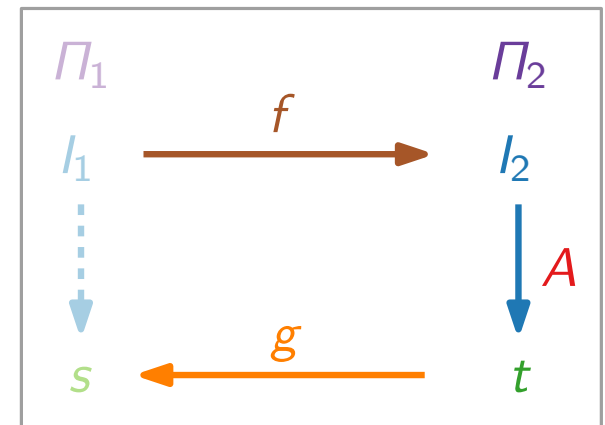
Let l_1 be an instance of Π_1 .

Set $l_2 := f(l_1)$, $t := A(l_2)$ and $s := g(l_1, t)$.

Then:

$$\text{obj}_{\Pi_1}(l_1, s) \leq \text{obj}_{\Pi_2}(l_2, t) \leq \alpha \cdot \text{OPT}_{\Pi_2}(l_2) \leq \alpha \cdot \text{OPT}_{\Pi_1}(l_1).$$

□



Approximation Algorithms

Lecture 3:

STEINERTREE and MULTIWAYCUT

Part III:

Reduction to METRICSTEINERTREE

METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $l_1 \xrightarrow{f} l_2$

METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $I_1 \xrightarrow{f} I_2$

Instance I_1 of STEINERTREE:

Graph $G_1 = (V, E_1)$, edge weights c_1 , partition $V = T \cup S$

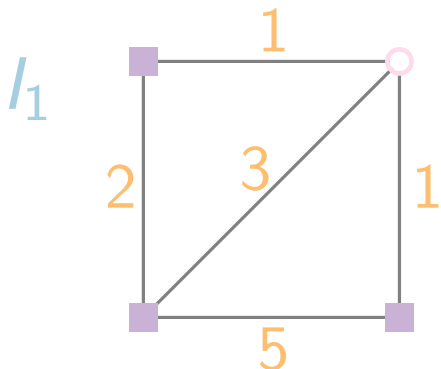
METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $I_1 \xrightarrow{f} I_2$

Instance I_1 of STEINERTREE:

Graph $G_1 = (V, E_1)$, edge weights c_1 , partition $V = T \cup S$



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

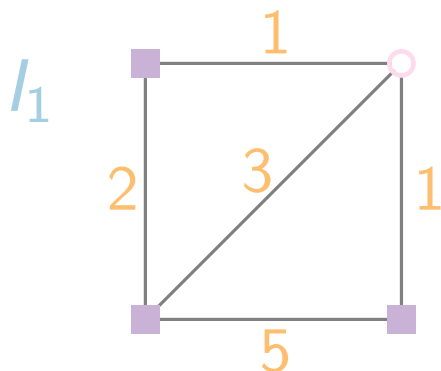
Proof. (1) Mapping f $I_1 \xrightarrow{f} I_2$

Instance I_1 of STEINERTREE:

Graph $G_1 = (V, E_1)$, edge weights c_1 , partition $V = T \cup S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

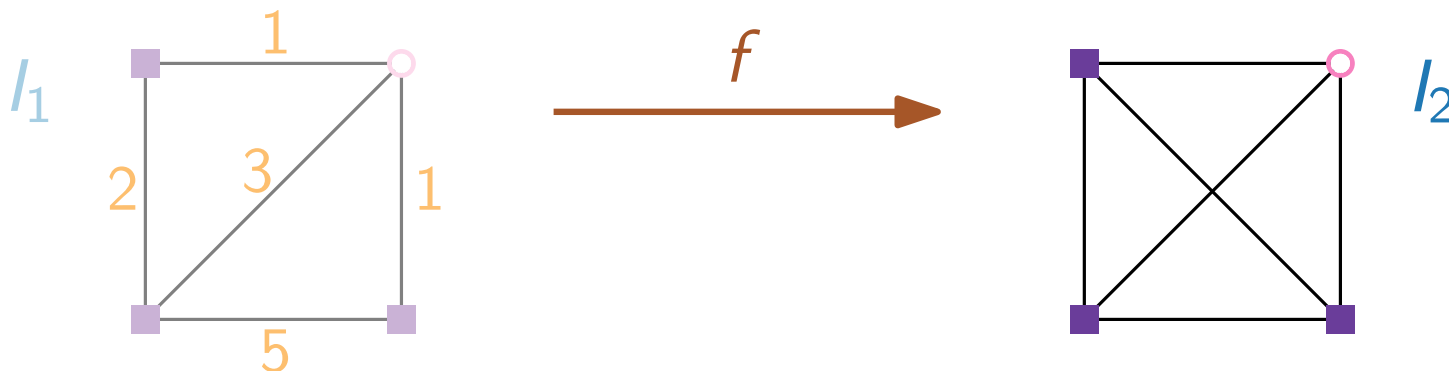
Proof. (1) Mapping f $I_1 \xrightarrow{f} I_2$

Instance I_1 of STEINERTREE:

Graph $G_1 = (V, E_1)$, edge weights c_1 , partition $V = T \cup S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $I_1 \xrightarrow{f} I_2$

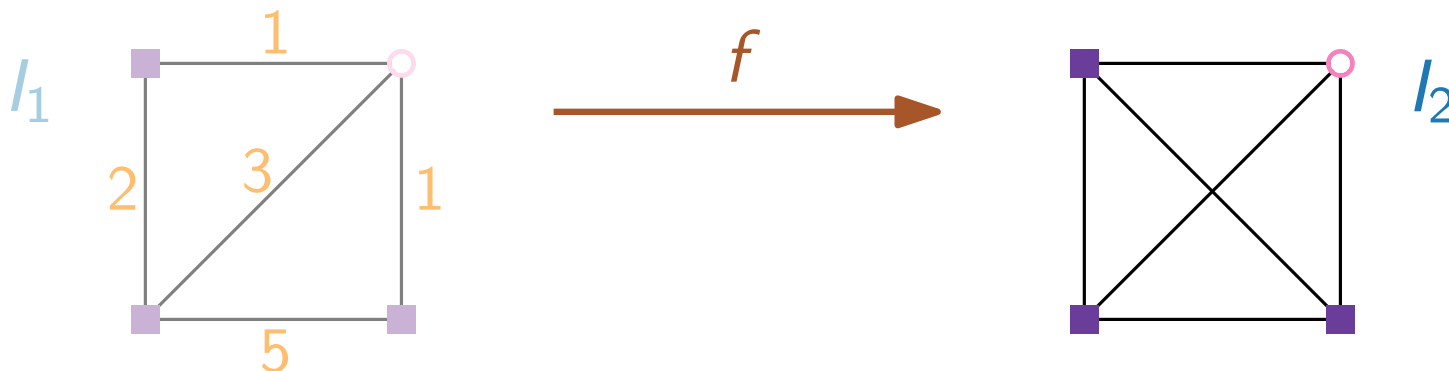
Instance I_1 of STEINERTREE:

Graph $G_1 = (V, E_1)$, edge weights c_1 , partition $V = T \cup S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) := \text{length of a shortest } u-v \text{ path in } G_1$.



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $I_1 \xrightarrow{f} I_2$

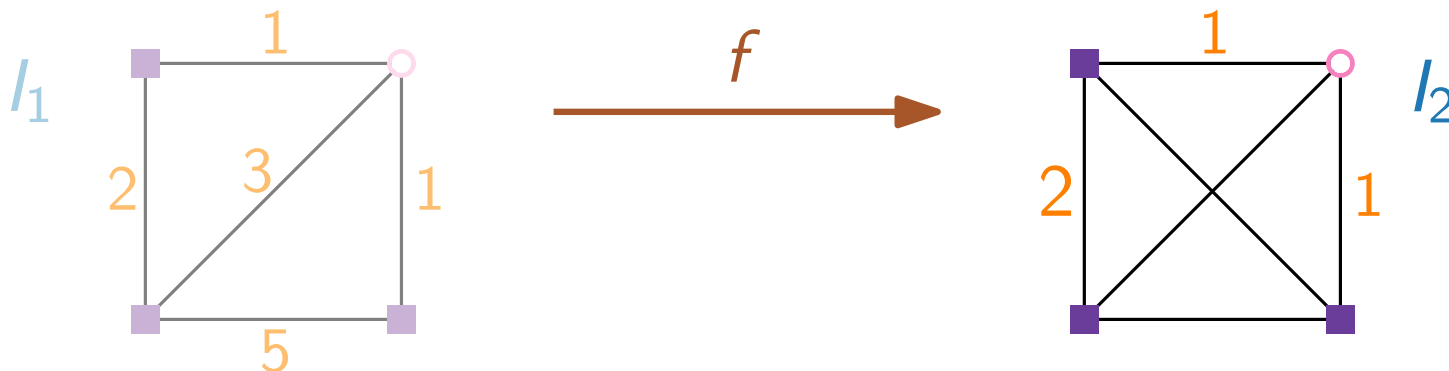
Instance I_1 of STEINERTREE:

Graph $G_1 = (V, E_1)$, edge weights c_1 , partition $V = T \cup S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) :=$ length of a shortest $u-v$ path in G_1 .



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $I_1 \xrightarrow{f} I_2$

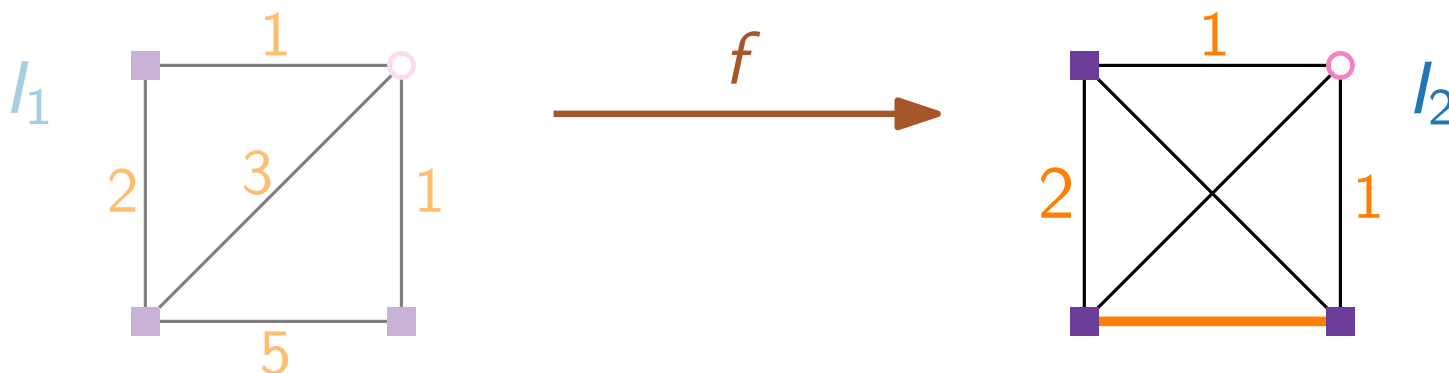
Instance I_1 of STEINERTREE:

Graph $G_1 = (V, E_1)$, edge weights c_1 , partition $V = T \cup S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) :=$ length of a shortest $u-v$ path in G_1 .



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $I_1 \xrightarrow{f} I_2$

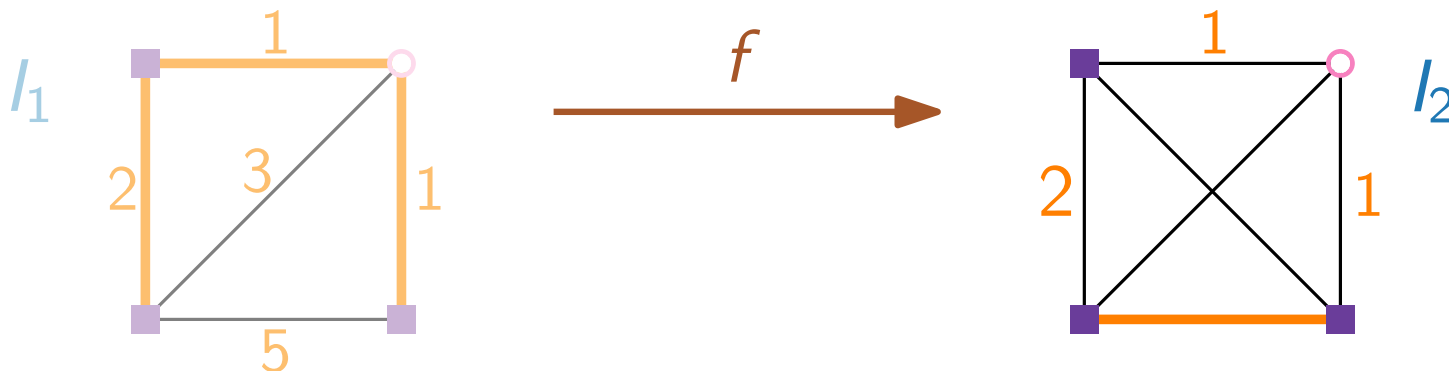
Instance I_1 of STEINERTREE:

Graph $G_1 = (V, E_1)$, edge weights c_1 , partition $V = T \cup S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) := \text{length of a shortest } u-v \text{ path in } G_1$.



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $I_1 \xrightarrow{f} I_2$

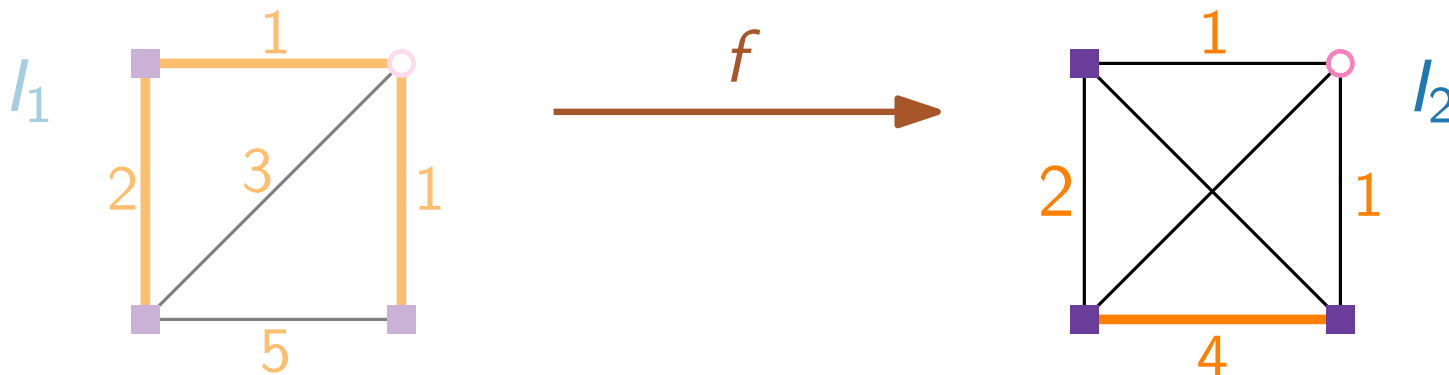
Instance I_1 of STEINERTREE:

Graph $G_1 = (V, E_1)$, edge weights c_1 , partition $V = T \cup S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) := \text{length of a shortest } u-v \text{ path in } G_1$.



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $I_1 \xrightarrow{f} I_2$

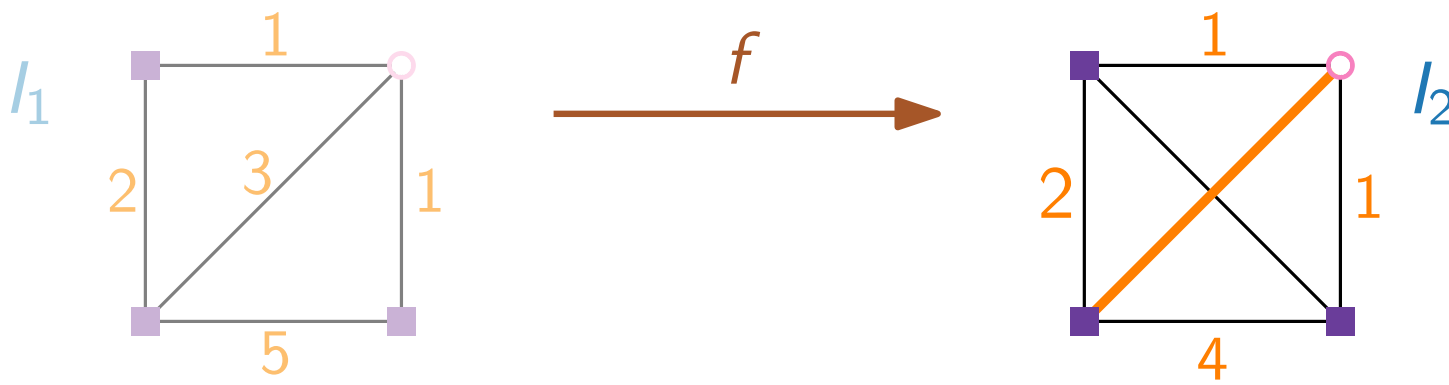
Instance I_1 of STEINERTREE:

Graph $G_1 = (V, E_1)$, edge weights c_1 , partition $V = T \cup S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) :=$ length of a shortest $u-v$ path in G_1 .



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $I_1 \xrightarrow{f} I_2$

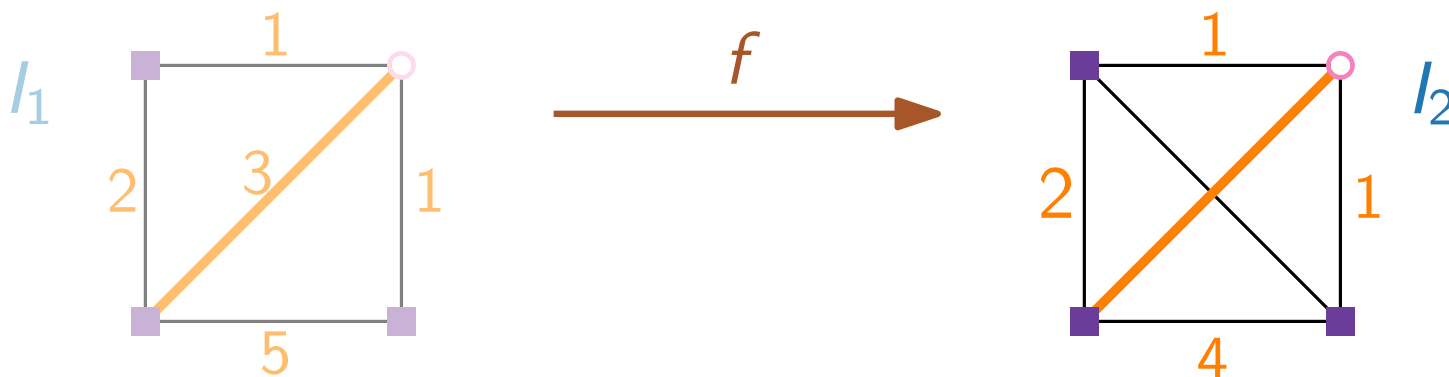
Instance I_1 of STEINERTREE:

Graph $G_1 = (V, E_1)$, edge weights c_1 , partition $V = T \cup S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) :=$ length of a shortest $u-v$ path in G_1 .



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $I_1 \xrightarrow{f} I_2$

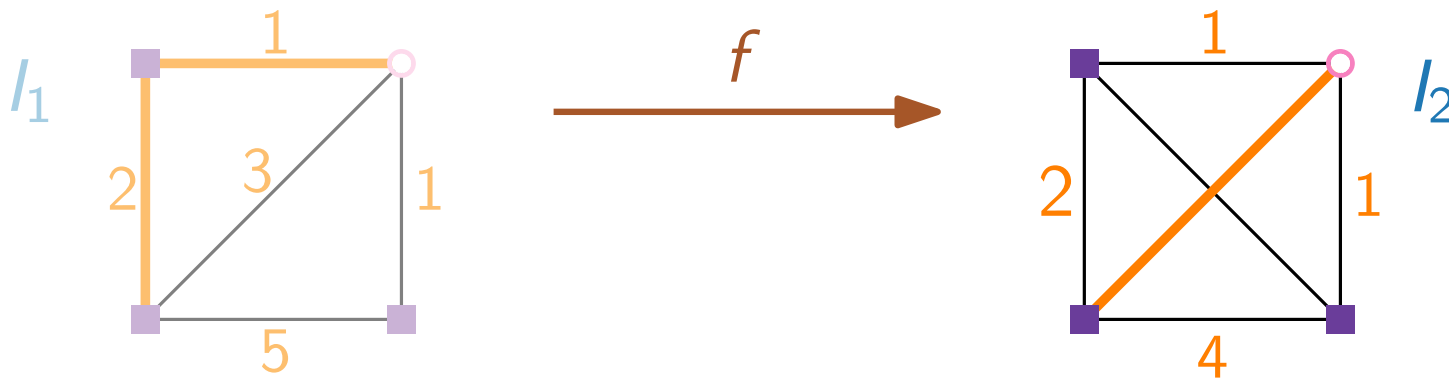
Instance I_1 of STEINERTREE:

Graph $G_1 = (V, E_1)$, edge weights c_1 , partition $V = T \cup S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) := \text{length of a shortest } u-v \text{ path in } G_1$.



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $I_1 \xrightarrow{f} I_2$

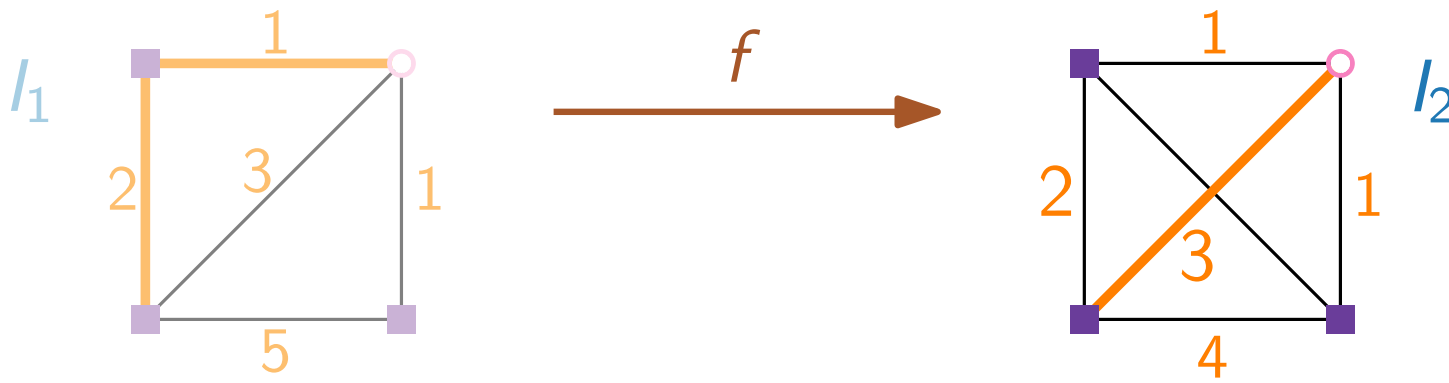
Instance I_1 of STEINERTREE:

Graph $G_1 = (V, E_1)$, edge weights c_1 , partition $V = T \cup S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) := \text{length of a shortest } u-v \text{ path in } G_1$.



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $I_1 \xrightarrow{f} I_2$

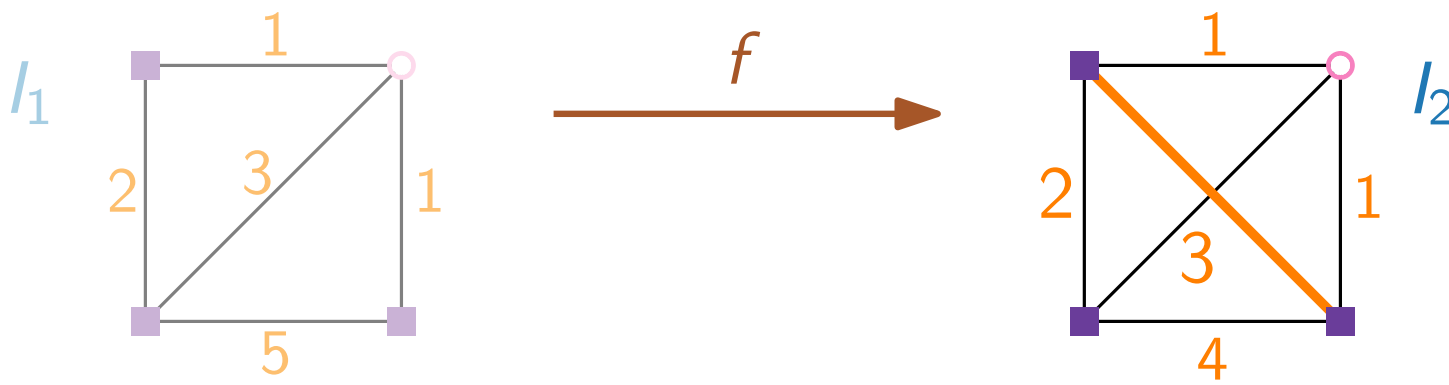
Instance I_1 of STEINERTREE:

Graph $G_1 = (V, E_1)$, edge weights c_1 , partition $V = T \cup S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) := \text{length of a shortest } u-v \text{ path in } G_1$.



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $I_1 \xrightarrow{f} I_2$

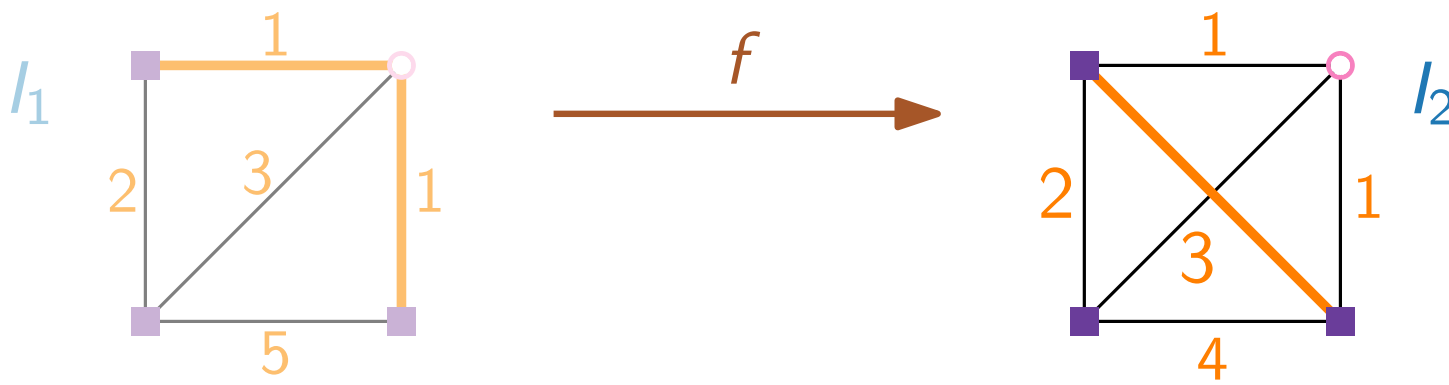
Instance I_1 of STEINERTREE:

Graph $G_1 = (V, E_1)$, edge weights c_1 , partition $V = T \cup S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) := \text{length of a shortest } u-v \text{ path in } G_1$.



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $I_1 \xrightarrow{f} I_2$

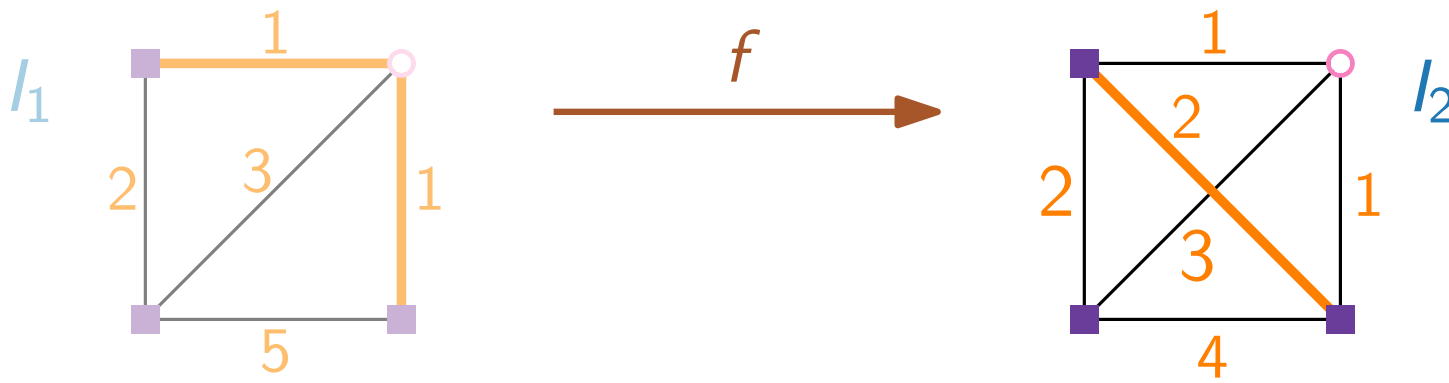
Instance I_1 of STEINERTREE:

Graph $G_1 = (V, E_1)$, edge weights c_1 , partition $V = T \cup S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) := \text{length of a shortest } u-v \text{ path in } G_1$.



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $I_1 \xrightarrow{f} I_2$

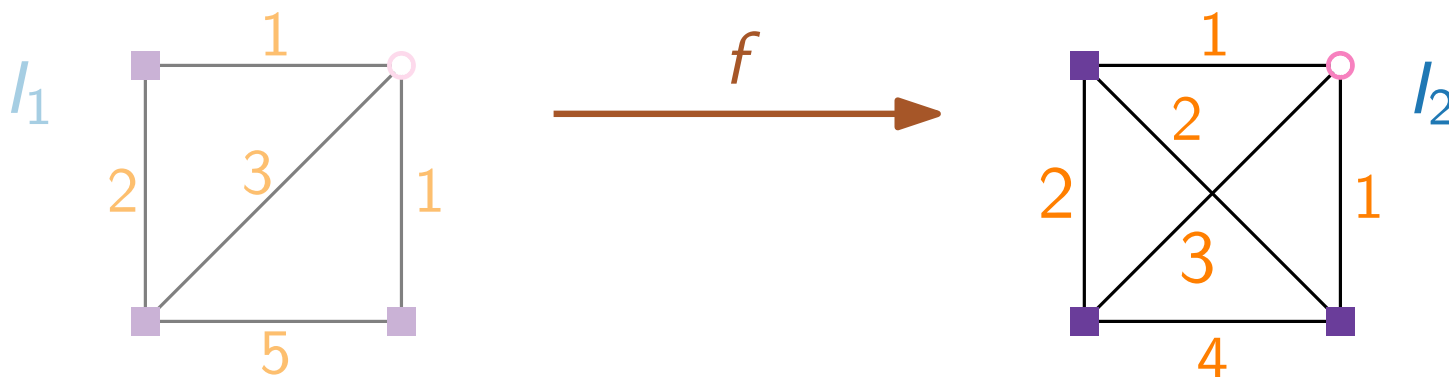
Instance I_1 of STEINERTREE:

Graph $G_1 = (V, E_1)$, edge weights c_1 , partition $V = T \cup S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) :=$ length of a shortest $u-v$ path in G_1 .



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $I_1 \xrightarrow{f} I_2$

Instance I_1 of STEINERTREE:

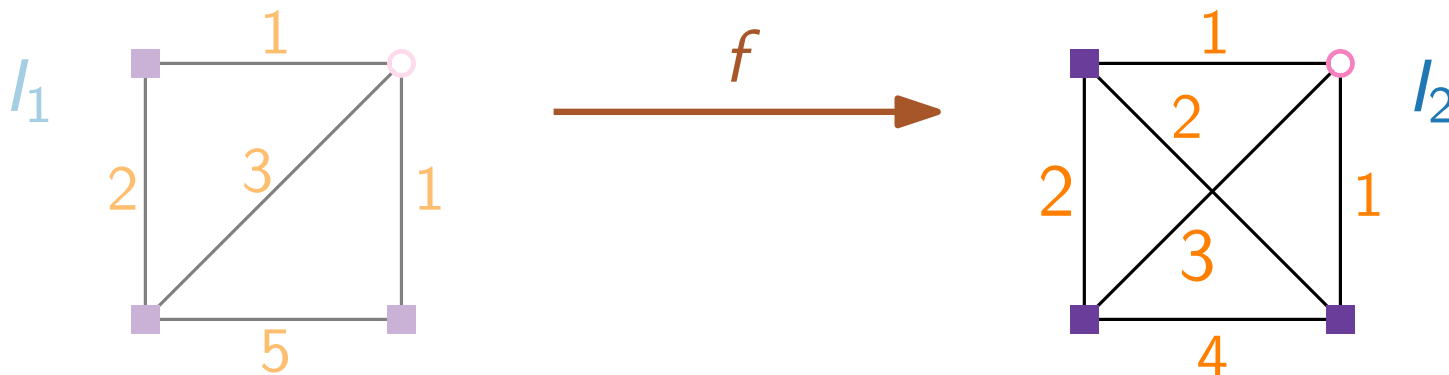
Graph $G_1 = (V, E_1)$, edge weights c_1 , partition $V = T \cup S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) :=$ length of a shortest $u-v$ path in G_1 .

Note: For every edge $(u, v) \in E_1$,



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (1) Mapping f $I_1 \xrightarrow{f} I_2$

Instance I_1 of STEINERTREE:

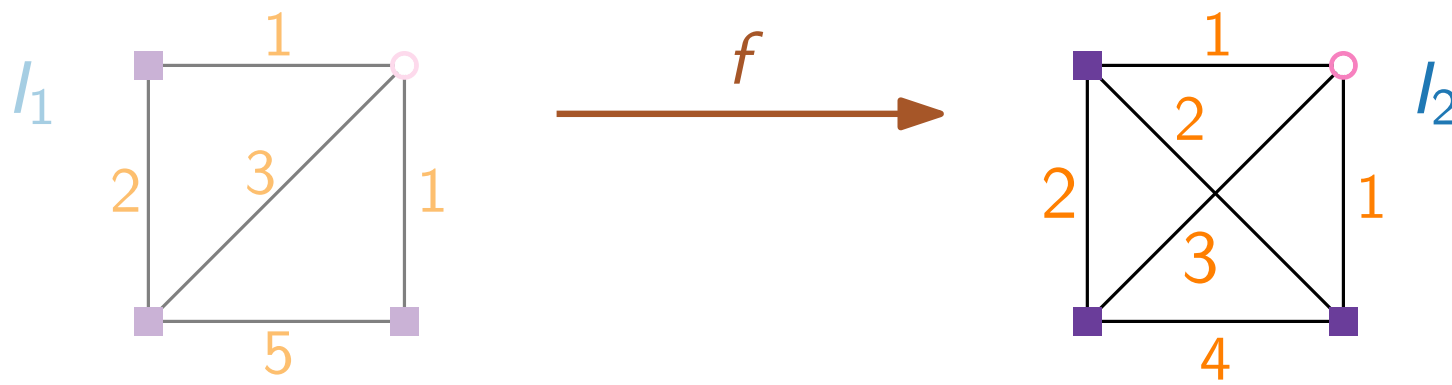
Graph $G_1 = (V, E_1)$, edge weights c_1 , partition $V = T \cup S$

Metric instance $I_2 := f(I_1)$:

Complete graph $G_2 = (V, E_2)$, partition T, S as in I_1

$c_2(u, v) :=$ length of a shortest $u-v$ path in G_1 .

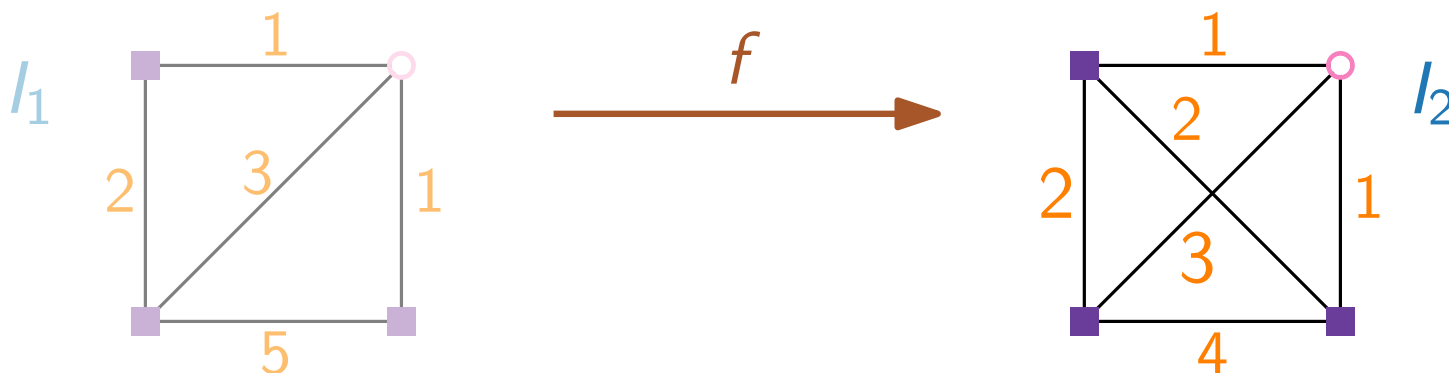
Note: For every edge $(u, v) \in E_1$, $c_2(u, v) \leq c_1(u, v)$.



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (2) $\text{OPT}(I_2) \leq \text{OPT}(I_1)$

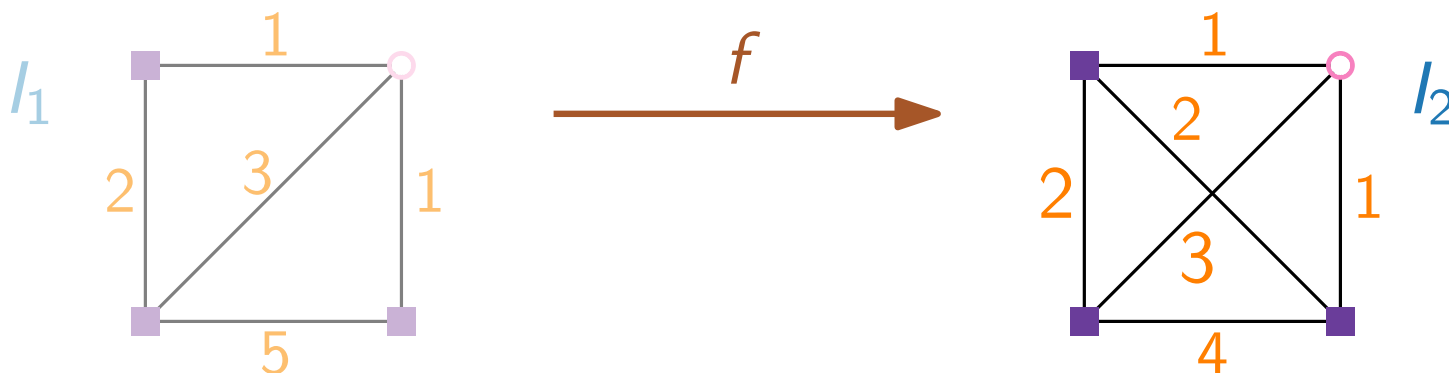


METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (2) $\text{OPT}(I_2) \leq \text{OPT}(I_1)$

Let B^* be an optimal Steiner tree for I_1 .

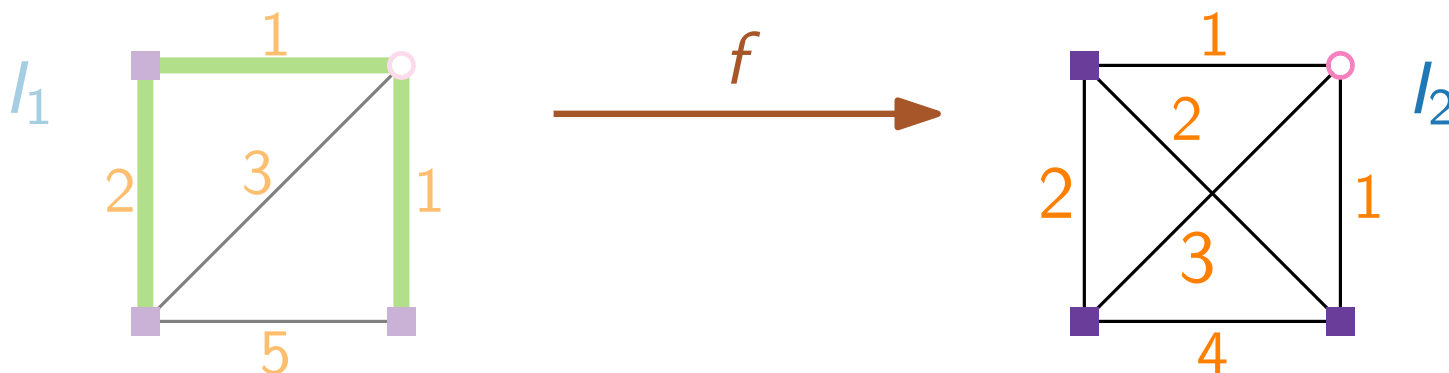


METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (2) $\text{OPT}(I_2) \leq \text{OPT}(I_1)$

Let B^* be an optimal Steiner tree for I_1 .



METRICSTEINERTREE

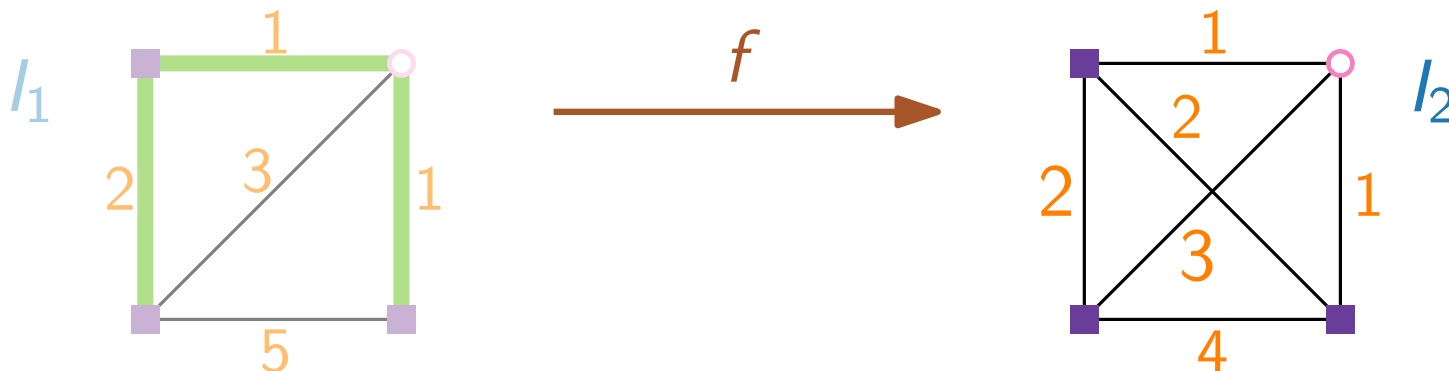
Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (2) $\text{OPT}(I_2) \leq \text{OPT}(I_1)$

Let B^* be an optimal Steiner tree for I_1 .

Note that B^* is also a feasible solution for I_2 :

$E_1 \subseteq E_2$ and the vertex sets V , T , S are the same.



METRICSTEINERTREE

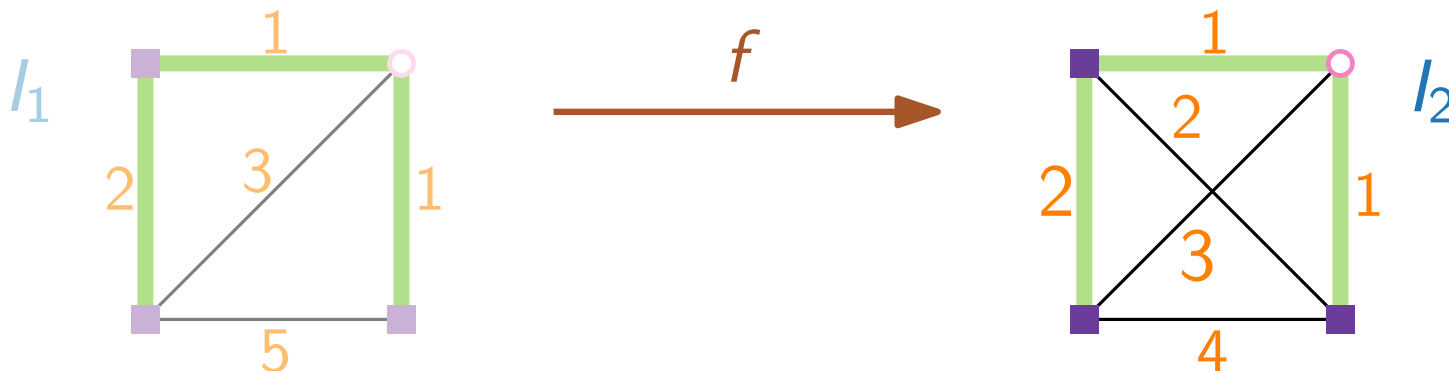
Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (2) $\text{OPT}(I_2) \leq \text{OPT}(I_1)$

Let B^* be an optimal Steiner tree for I_1 .

Note that B^* is also a feasible solution for I_2 :

$E_1 \subseteq E_2$ and the vertex sets V , T , S are the same.



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

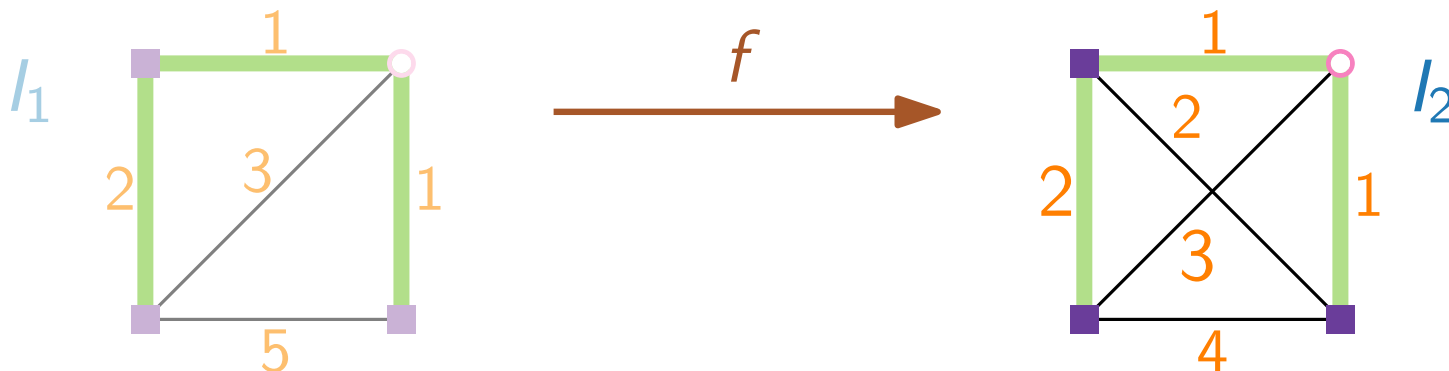
Proof. (2) $\text{OPT}(I_2) \leq \text{OPT}(I_1)$

Let B^* be an optimal Steiner tree for I_1 .

Note that B^* is also a feasible solution for I_2 :

$E_1 \subseteq E_2$ and the vertex sets V , T , S are the same.

$\text{OPT}(I_2)$



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

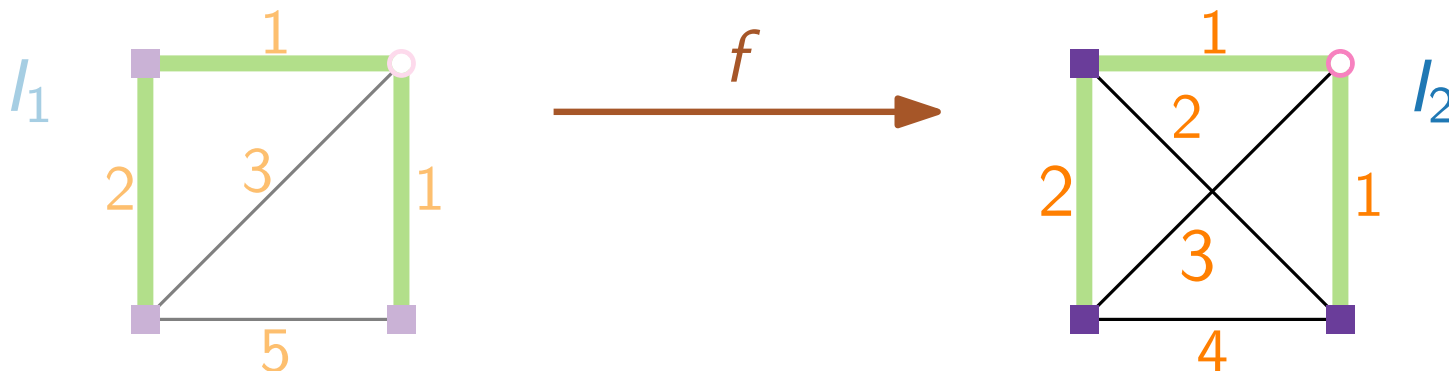
Proof. (2) $\text{OPT}(I_2) \leq \text{OPT}(I_1)$

Let B^* be an optimal Steiner tree for I_1 .

Note that B^* is also a feasible solution for I_2 :

$E_1 \subseteq E_2$ and the vertex sets V , T , S are the same.

$$\text{OPT}(I_2) \leq c_2(B^*)$$



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

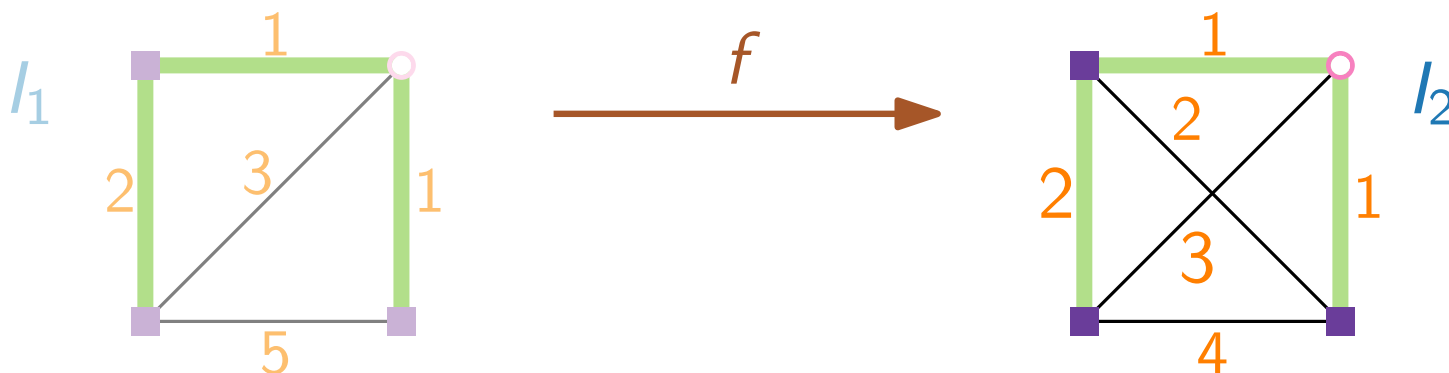
Proof. (2) $\text{OPT}(I_2) \leq \text{OPT}(I_1)$

Let B^* be an optimal Steiner tree for I_1 .

Note that B^* is also a feasible solution for I_2 :

$E_1 \subseteq E_2$ and the vertex sets V , T , S are the same.

$$\text{OPT}(I_2) \leq c_2(B^*) \leq c_1(B^*)$$



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

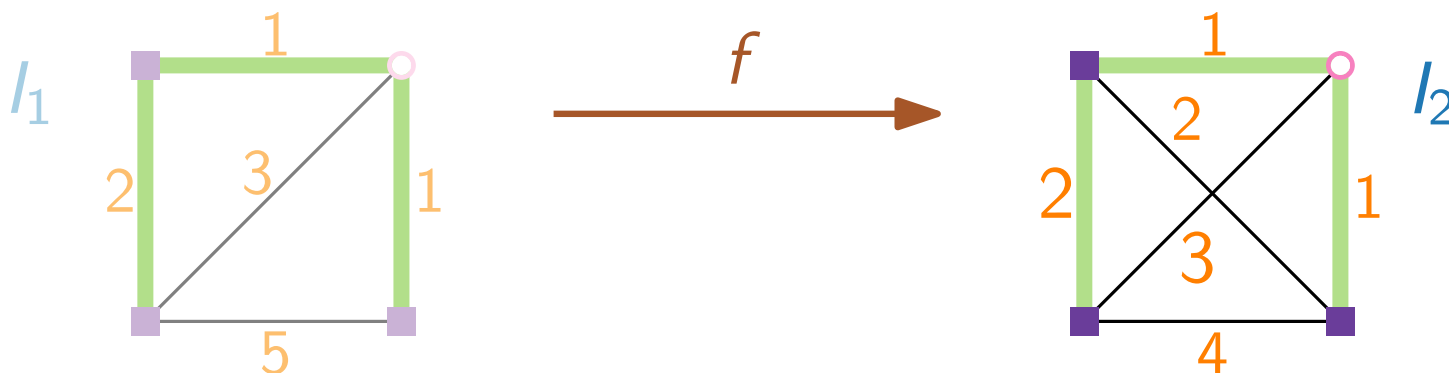
Proof. (2) $\text{OPT}(I_2) \leq \text{OPT}(I_1)$

Let B^* be an optimal Steiner tree for I_1 .

Note that B^* is also a feasible solution for I_2 :

$E_1 \subseteq E_2$ and the vertex sets V , T , S are the same.

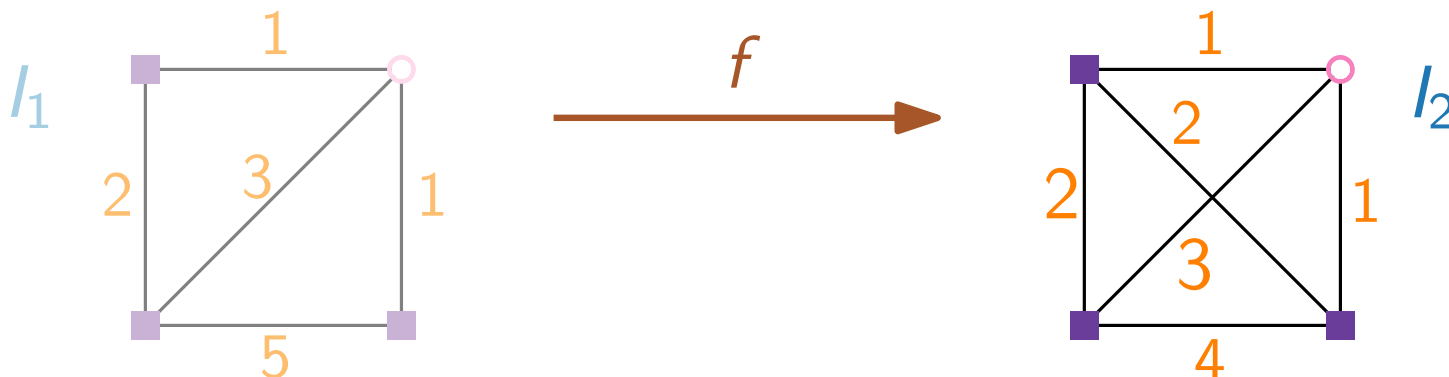
$$\text{OPT}(I_2) \leq c_2(B^*) \leq c_1(B^*) = \text{OPT}(I_1)$$



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (3) Mapping g $s \xleftarrow{g} t$

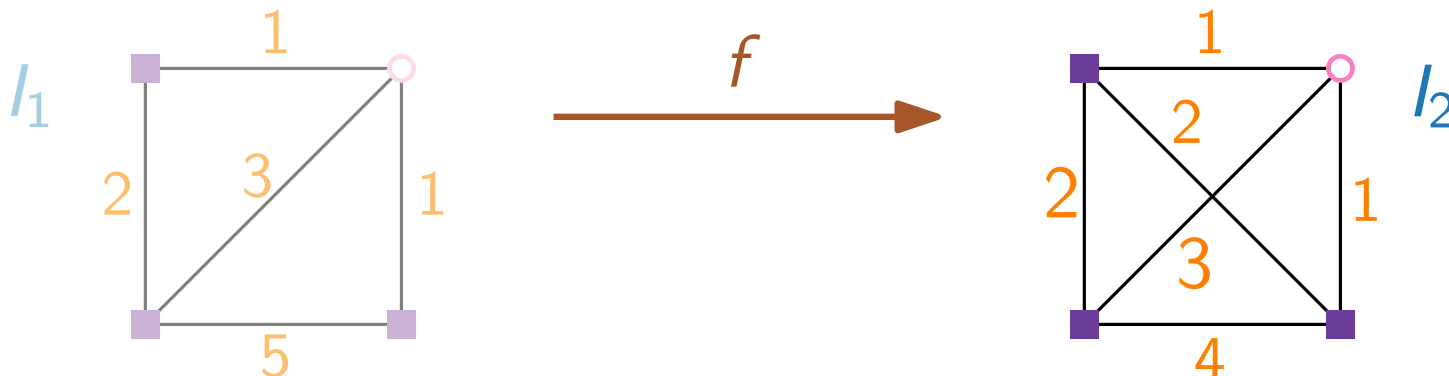


METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (3) Mapping g $s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2 .

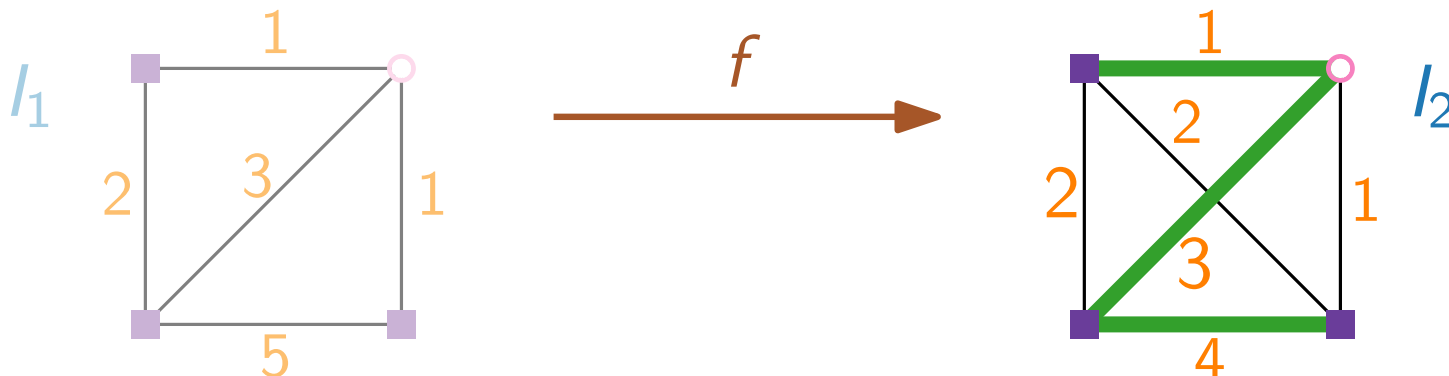


METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (3) Mapping g $s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2 .



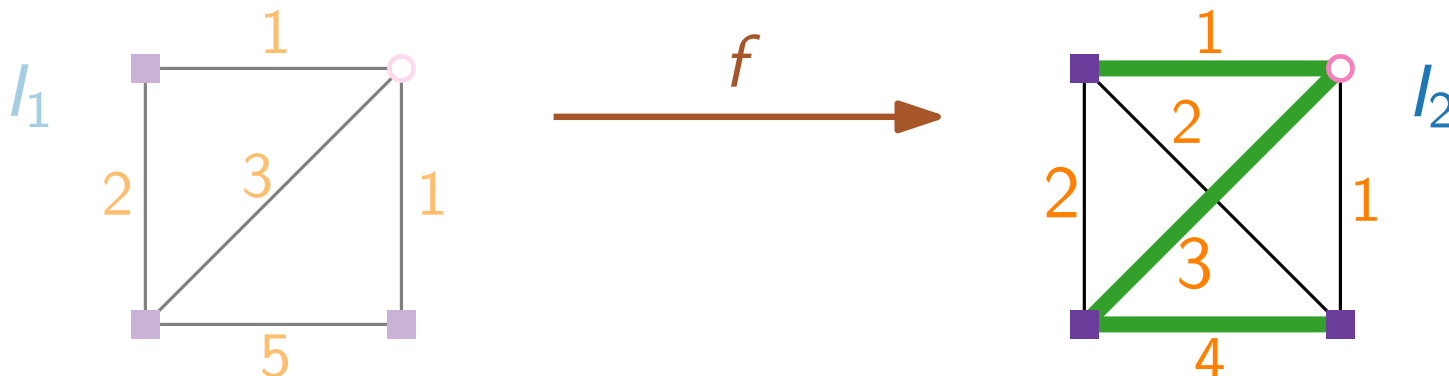
METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (3) Mapping g $s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2 .

Construct $G'_1 \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest $u-v$ path in G_1 .



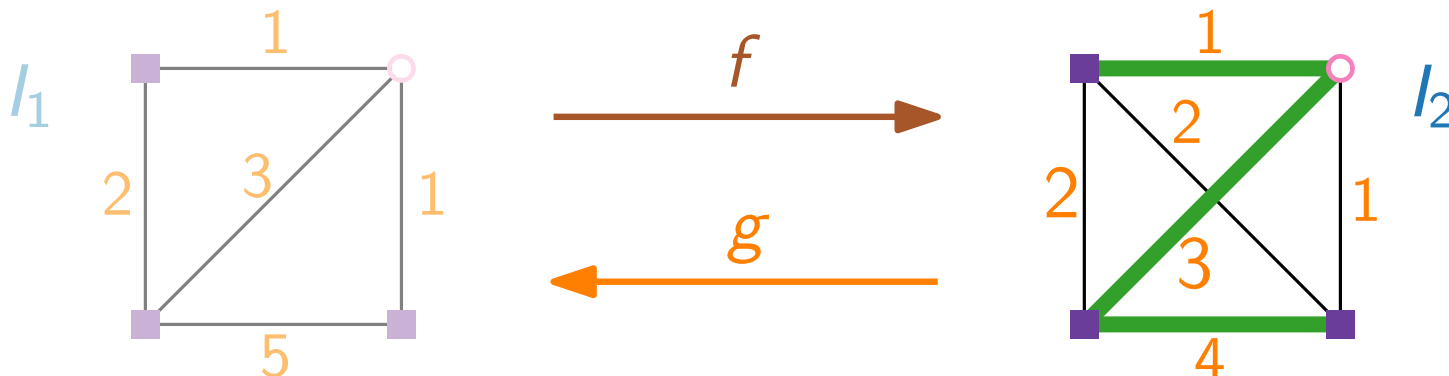
METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (3) Mapping g $s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2 .

Construct $G'_1 \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest $u-v$ path in G_1 .



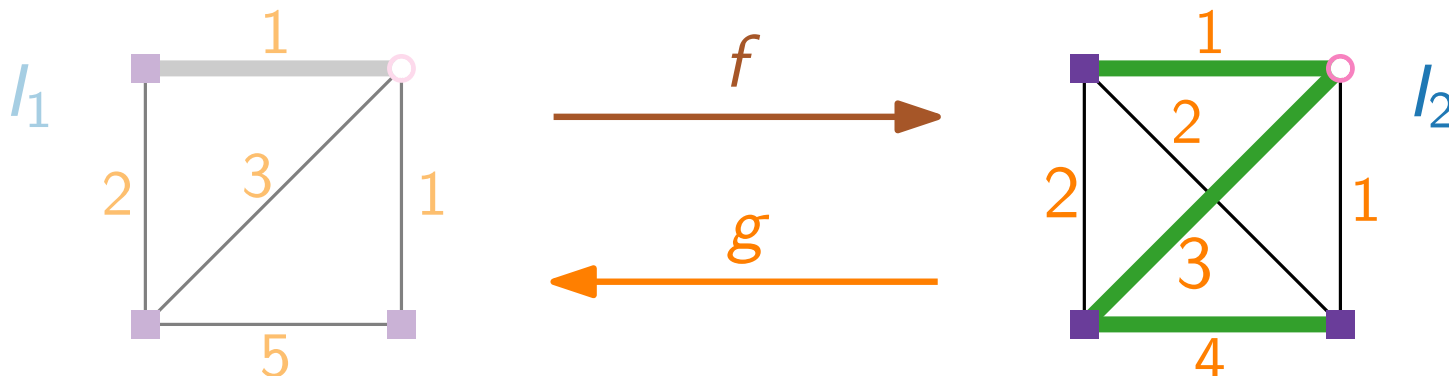
METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (3) Mapping g $s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2 .

Construct $G'_1 \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest $u-v$ path in G_1 .



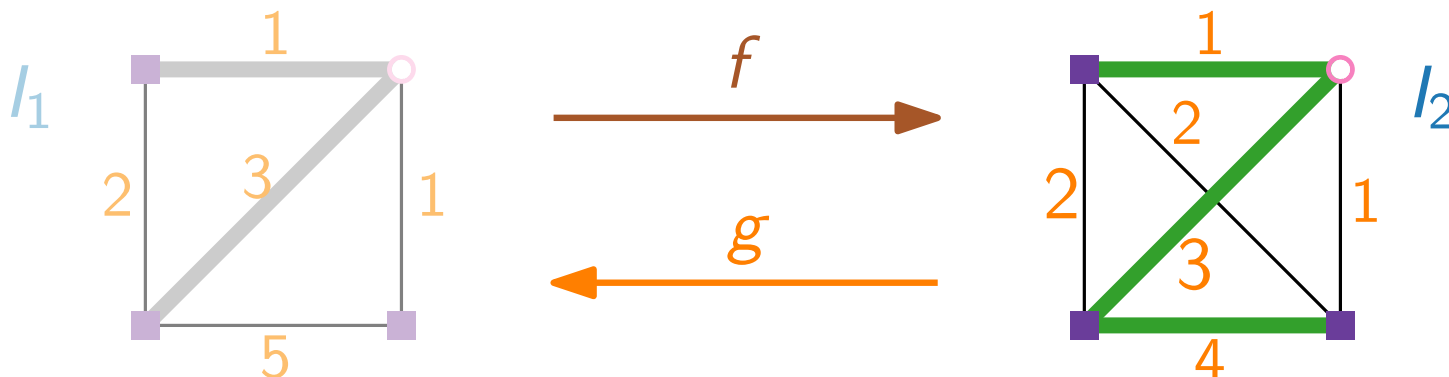
METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (3) Mapping g $s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2 .

Construct $G'_1 \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest $u-v$ path in G_1 .



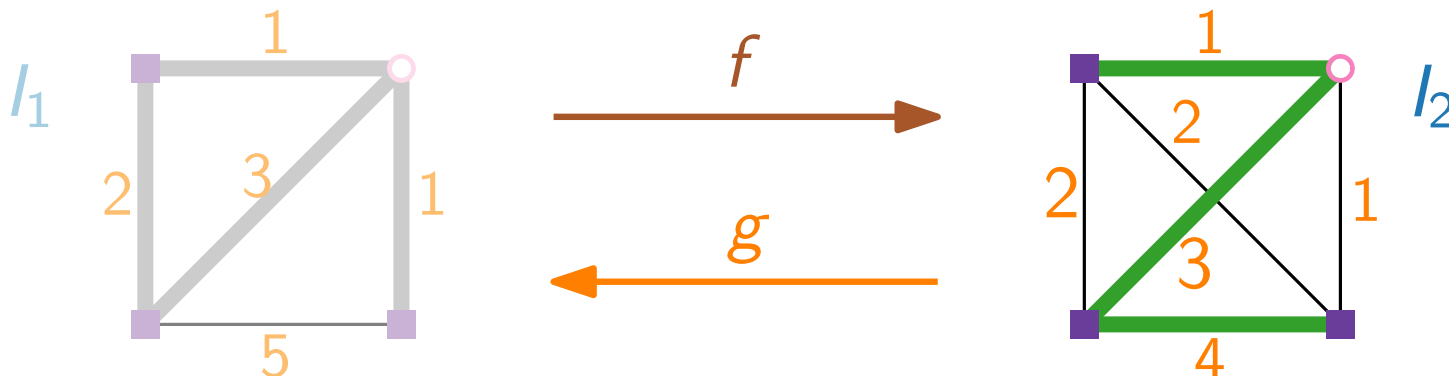
METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (3) Mapping g $s \xleftarrow{g} t$

Let B_2 be a Steiner tree of G_2 .

Construct $G'_1 \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest $u-v$ path in G_1 .



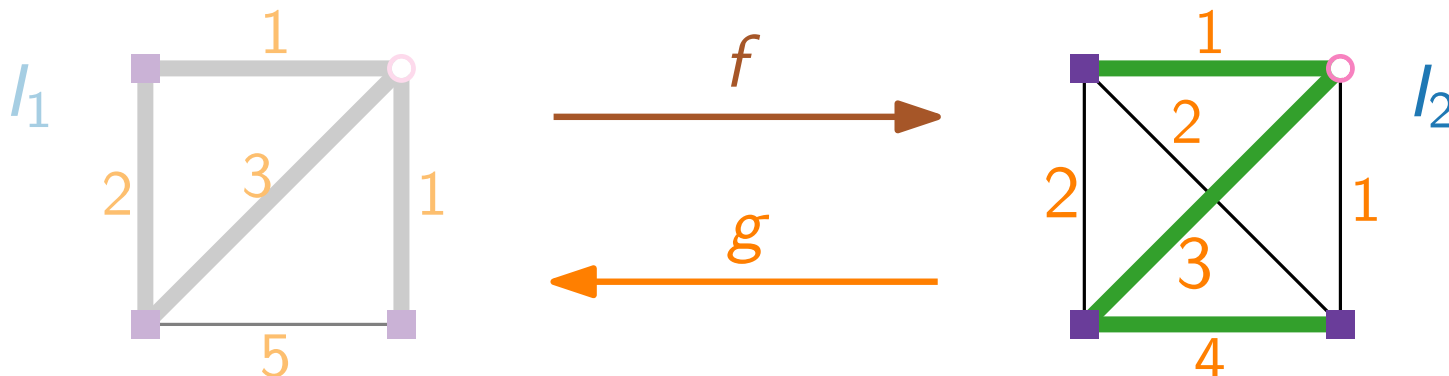
METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (3) Mapping g 

Let B_2 be a Steiner tree of G_2 .

Construct $G'_1 \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest $u-v$ path in G_1 . Keep ≤ 1 copy per edge.



METRICSTEINERTREE

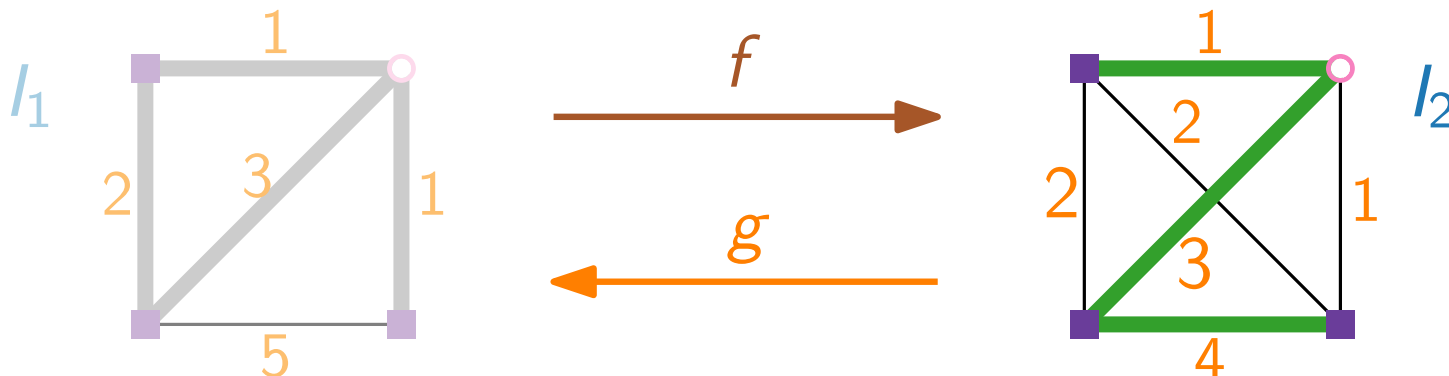
Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (3) Mapping g 

Let B_2 be a Steiner tree of G_2 .

Construct $G'_1 \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest $u-v$ path in G_1 . Keep ≤ 1 copy per edge.

$$c_1(G'_1) \leq c_2(B_2)$$



METRICSTEINERTREE

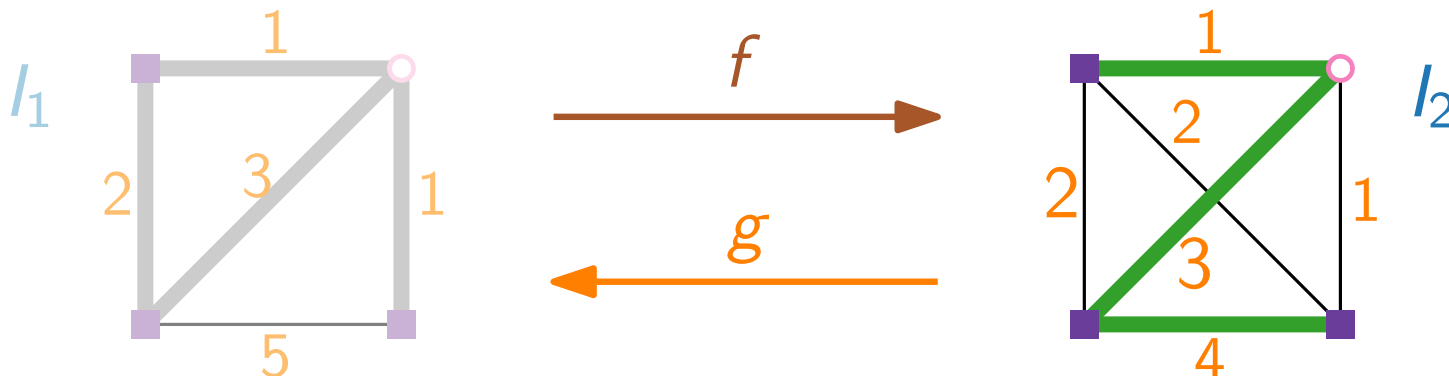
Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (3) Mapping g 

Let B_2 be a Steiner tree of G_2 .

Construct $G'_1 \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest $u-v$ path in G_1 . Keep ≤ 1 copy per edge.

$c_1(G'_1) \leq c_2(B_2)$; G'_1 connects all terminals



METRICSTEINERTREE

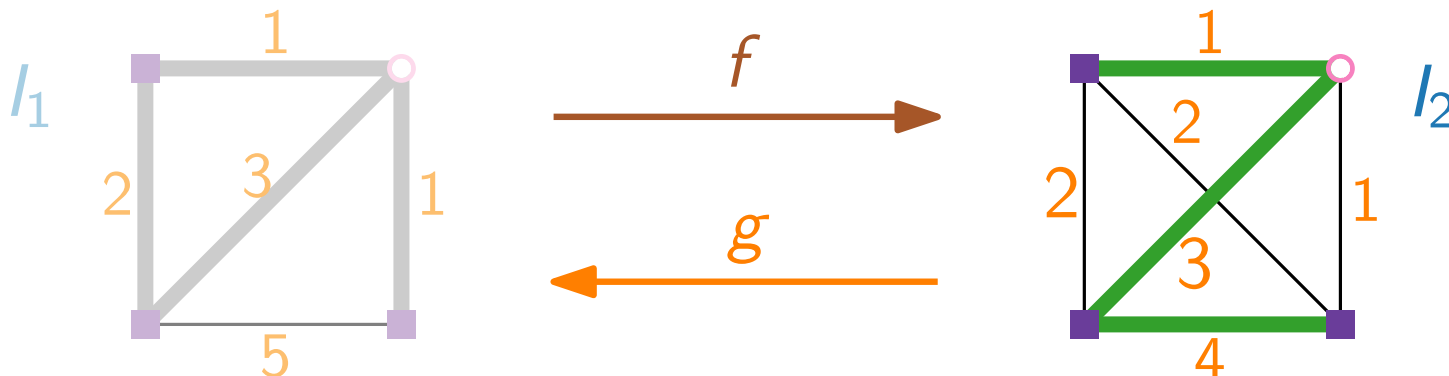
Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (3) Mapping g 

Let B_2 be a Steiner tree of G_2 .

Construct $G'_1 \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest $u-v$ path in G_1 . Keep ≤ 1 copy per edge.

$c_1(G'_1) \leq c_2(B_2)$; G'_1 connects all terminals; maybe not a tree.



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

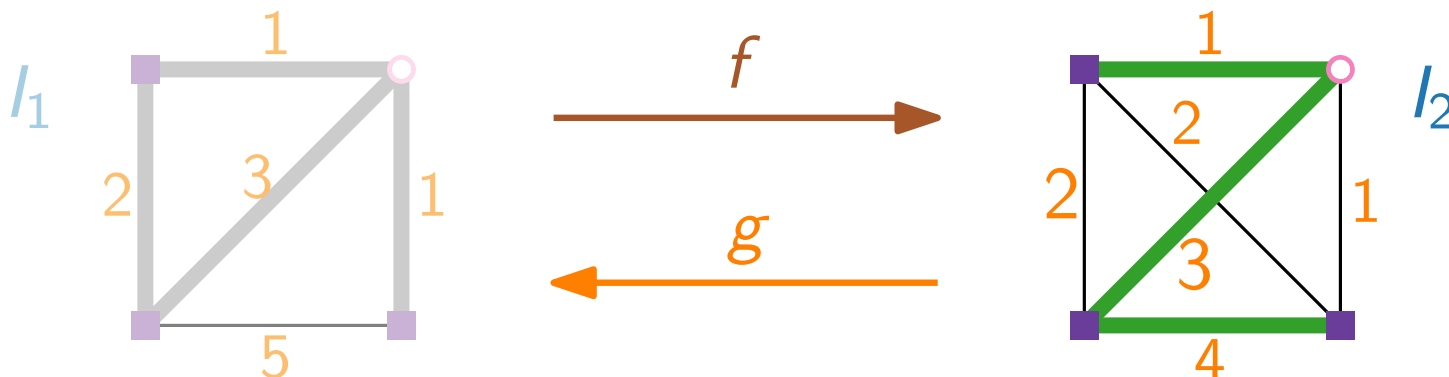
Proof. (3) Mapping g 

Let B_2 be a Steiner tree of G_2 .

Construct $G'_1 \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest $u-v$ path in G_1 . Keep ≤ 1 copy per edge.

$c_1(G'_1) \leq c_2(B_2)$; G'_1 connects all terminals; maybe not a tree.

Consider spanning tree B_1 of G'_1



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

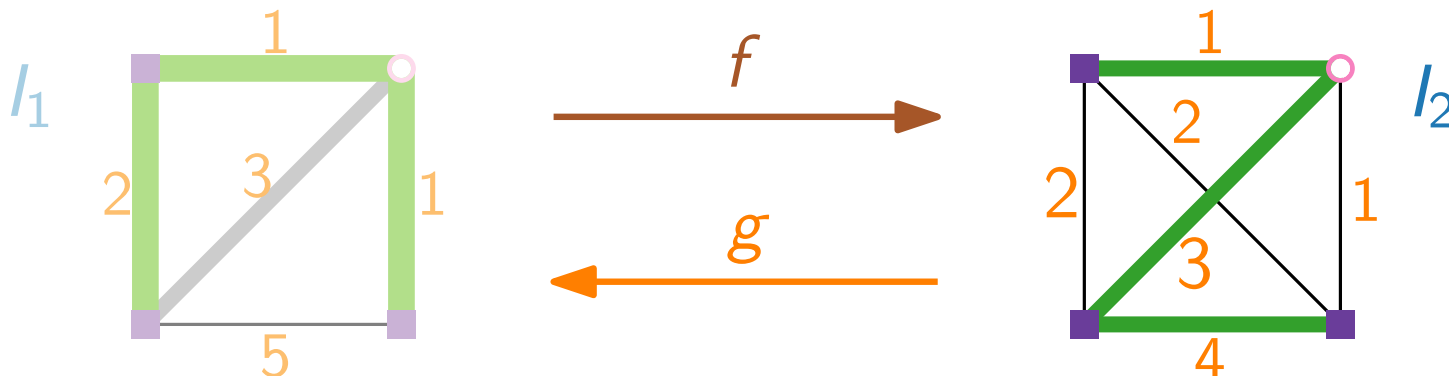
Proof. (3) Mapping g 

Let B_2 be a Steiner tree of G_2 .

Construct $G'_1 \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest $u-v$ path in G_1 . Keep ≤ 1 copy per edge.

$c_1(G'_1) \leq c_2(B_2)$; G'_1 connects all terminals; maybe not a tree.

Consider spanning tree B_1 of G'_1



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

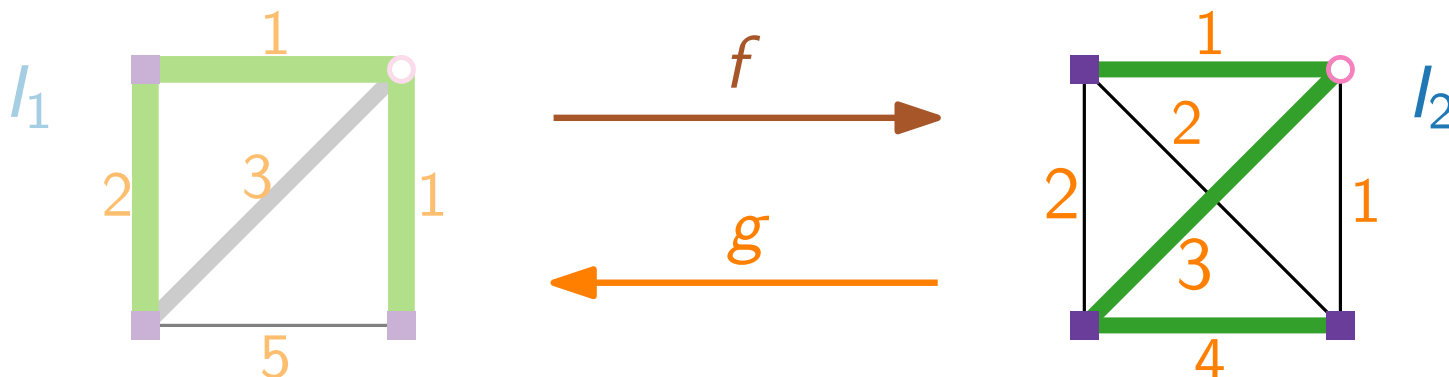
Proof. (3) Mapping g 

Let B_2 be a Steiner tree of G_2 .

Construct $G'_1 \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest $u-v$ path in G_1 . Keep ≤ 1 copy per edge.

$c_1(G'_1) \leq c_2(B_2)$; G'_1 connects all terminals; maybe not a tree.

Consider spanning tree B_1 of $G'_1 \rightsquigarrow$ Steiner tree B_1 of G_1



METRICSTEINERTREE

Theorem. There is an approximation-preserving reduction from STEINERTREE to METRICSTEINERTREE.

Proof. (3) Mapping g $s \xleftarrow{g} t$

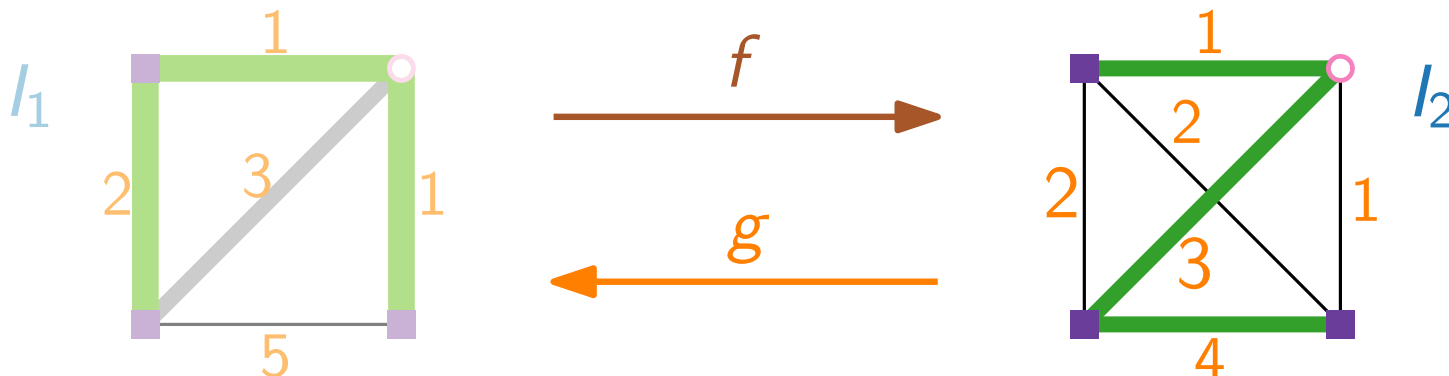
Let B_2 be a Steiner tree of G_2 .

Construct $G'_1 \subseteq G_1$ from B_2 by replacing each edge (u, v) of B_2 by a shortest $u-v$ path in G_1 . Keep ≤ 1 copy per edge.

$c_1(G'_1) \leq c_2(B_2)$; G'_1 connects all terminals; maybe not a tree.

Consider spanning tree B_1 of $G'_1 \rightsquigarrow$ Steiner tree B_1 of G_1

Note that $c_1(B_1) \leq c_1(G'_1) \leq c_2(B_2)$.



Approximation Algorithms

Lecture 3:

STEINERTREE and MULTIWAYCUT

Part IV:

2-Approximation for STEINERTREE

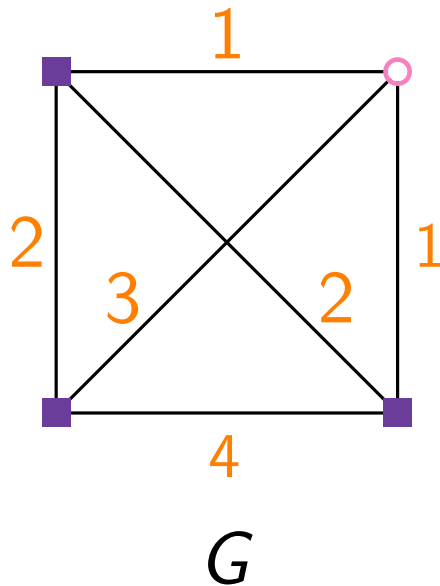
2-Approximation for STEINERTREE

2-Approximation for STEINERTREE

Theorem. For an instance of METRICSTEINERTREE, let B be a minimum spanning tree (MST) of the subgraph $G[T]$ induced by the terminal set T . Then $c(B) \leq 2 \cdot \text{OPT}$.

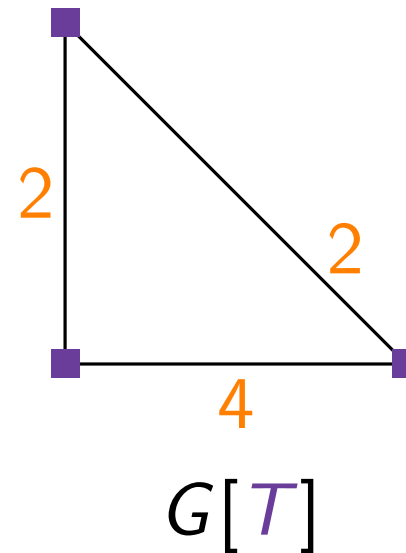
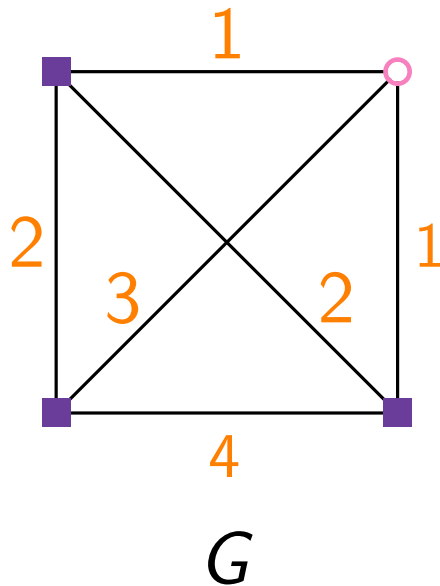
2-Approximation for STEINERTREE

Theorem. For an instance of METRICSTEINERTREE, let B be a minimum spanning tree (MST) of the subgraph $G[T]$ induced by the terminal set T . Then $c(B) \leq 2 \cdot \text{OPT}$.



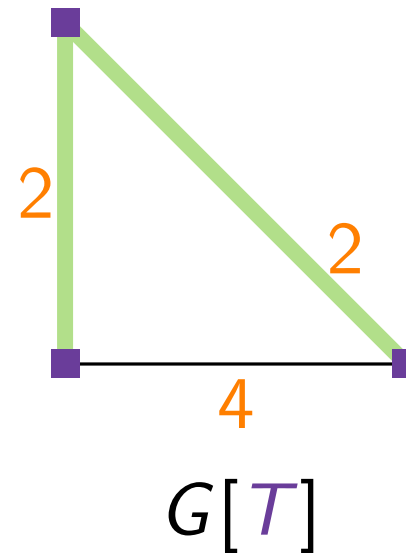
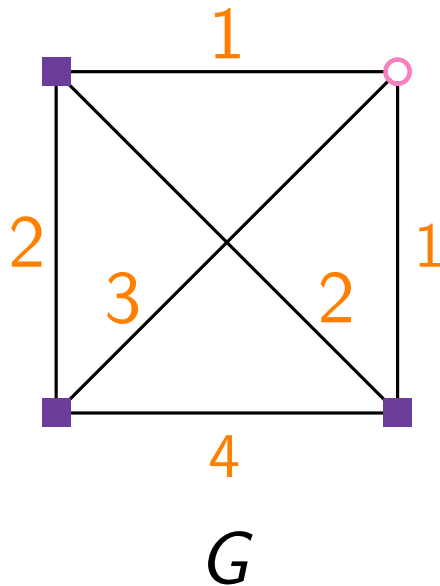
2-Approximation for STEINERTREE

Theorem. For an instance of METRICSTEINERTREE, let B be a minimum spanning tree (MST) of the subgraph $G[T]$ induced by the terminal set T . Then $c(B) \leq 2 \cdot \text{OPT}$.



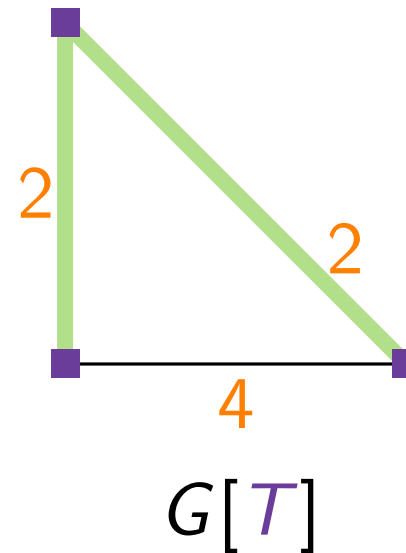
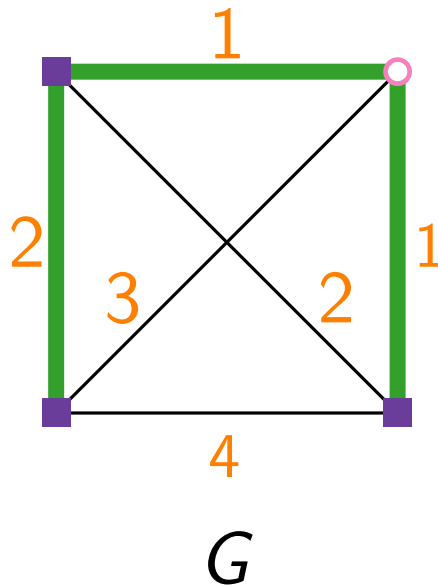
2-Approximation for STEINERTREE

Theorem. For an instance of METRICSTEINERTREE, let B be a minimum spanning tree (MST) of the subgraph $G[T]$ induced by the terminal set T . Then $c(B) \leq 2 \cdot \text{OPT}$.



2-Approximation for STEINERTREE

Theorem. For an instance of METRICSTEINERTREE, let B be a minimum spanning tree (MST) of the subgraph $G[T]$ induced by the terminal set T . Then $c(B) \leq 2 \cdot \text{OPT}$.

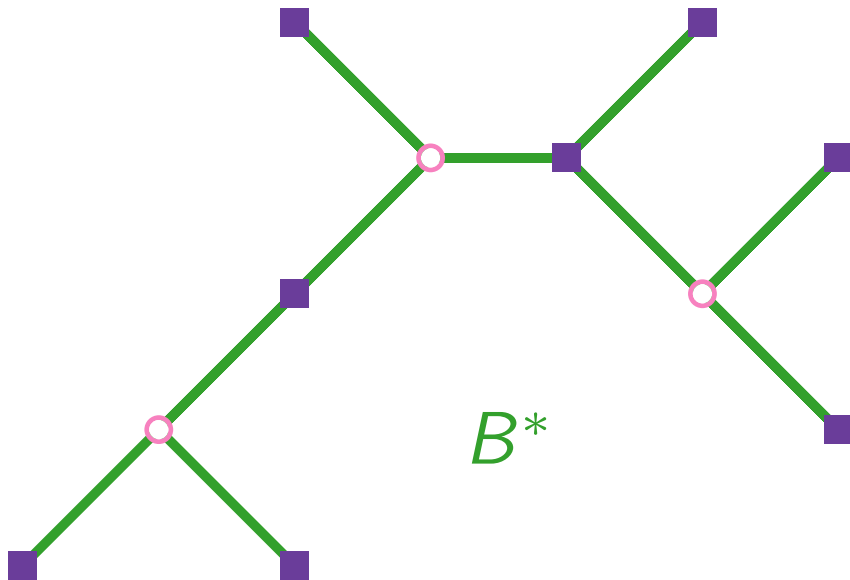


Proof of Approximation Factor

Consider an optimal Steiner tree B^* .

Proof of Approximation Factor

Consider an optimal Steiner tree B^* .

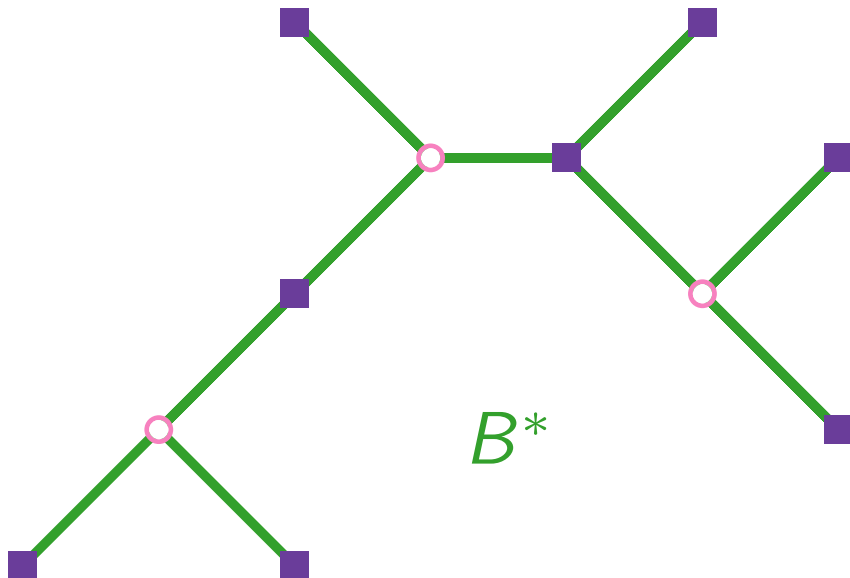


Proof of Approximation Factor

Consider an optimal Steiner tree B^* .

Duplicate all edges of B^* .

$\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \text{OPT}$.

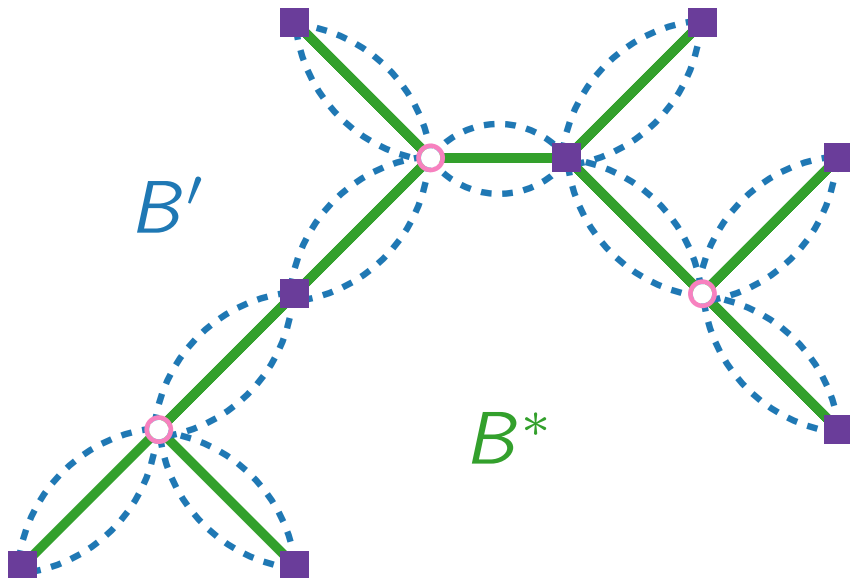


Proof of Approximation Factor

Consider an optimal Steiner tree B^* .

Duplicate all edges of B^* .

$\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \text{OPT}$.



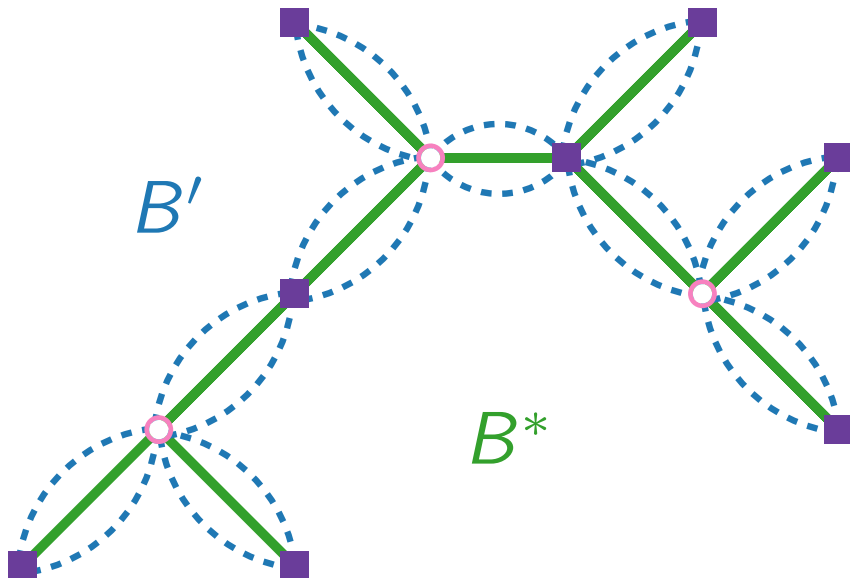
Proof of Approximation Factor

Consider an optimal Steiner tree B^* .

Duplicate all edges of B^* .

$\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \text{OPT}$.

Find a Eulerian tour T' in B'



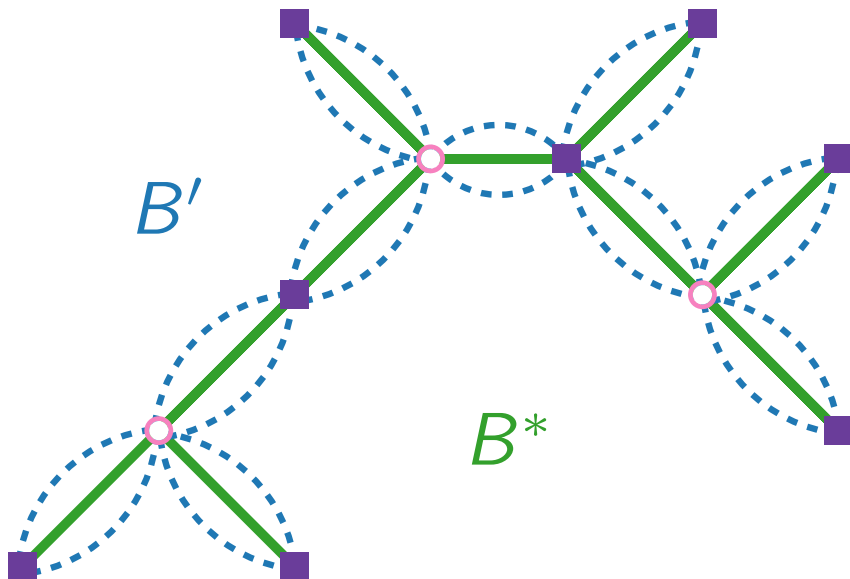
Proof of Approximation Factor

Consider an optimal Steiner tree B^* .

Duplicate all edges of B^* .

$\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \text{OPT}$.

Find a Eulerian tour T' in $B' \Rightarrow c(T') = c(B') = 2 \cdot \text{OPT}$



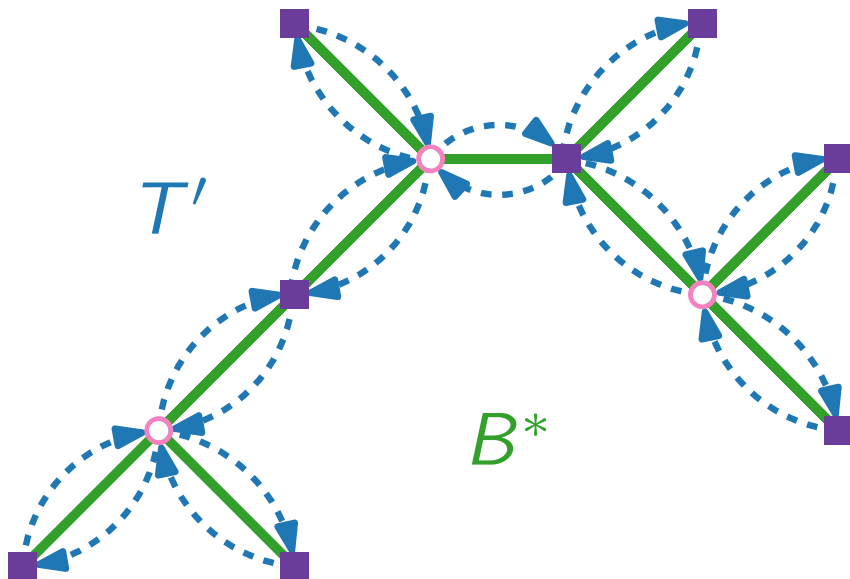
Proof of Approximation Factor

Consider an optimal Steiner tree B^* .

Duplicate all edges of B^* .

$\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \text{OPT}$.

Find a Eulerian tour T' in $B' \Rightarrow c(T') = c(B') = 2 \cdot \text{OPT}$



Proof of Approximation Factor

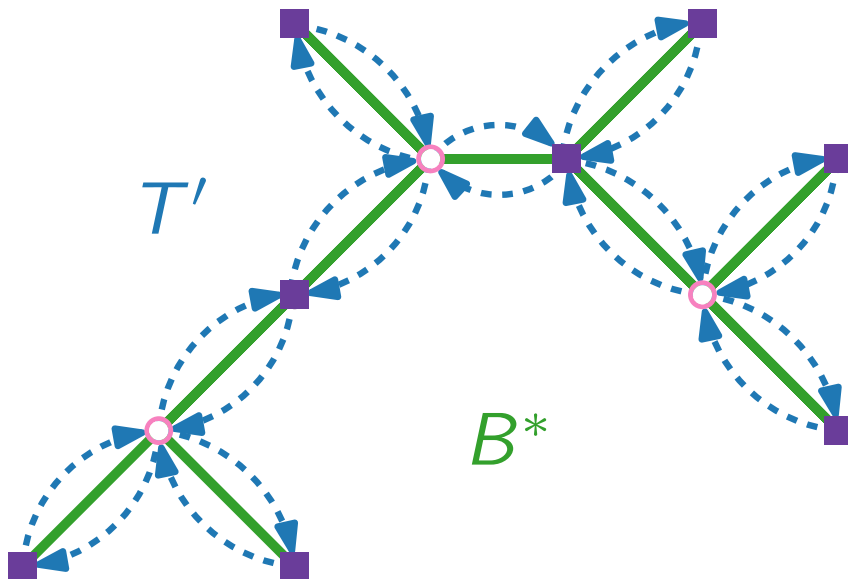
Consider an optimal Steiner tree B^* .

Duplicate all edges of B^* .

$\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \text{OPT}$.

Find a Eulerian tour T' in $B' \Rightarrow c(T') = c(B') = 2 \cdot \text{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and previously visited terminals.



Proof of Approximation Factor

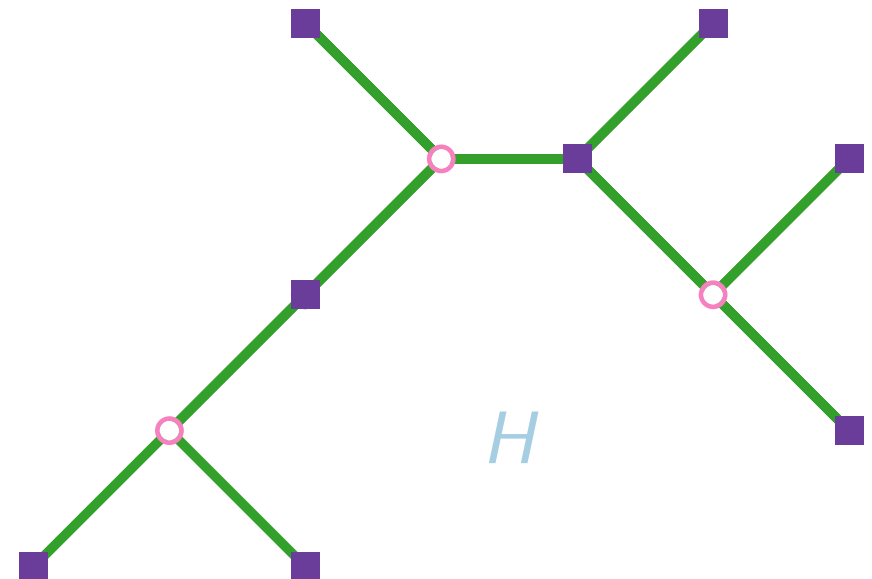
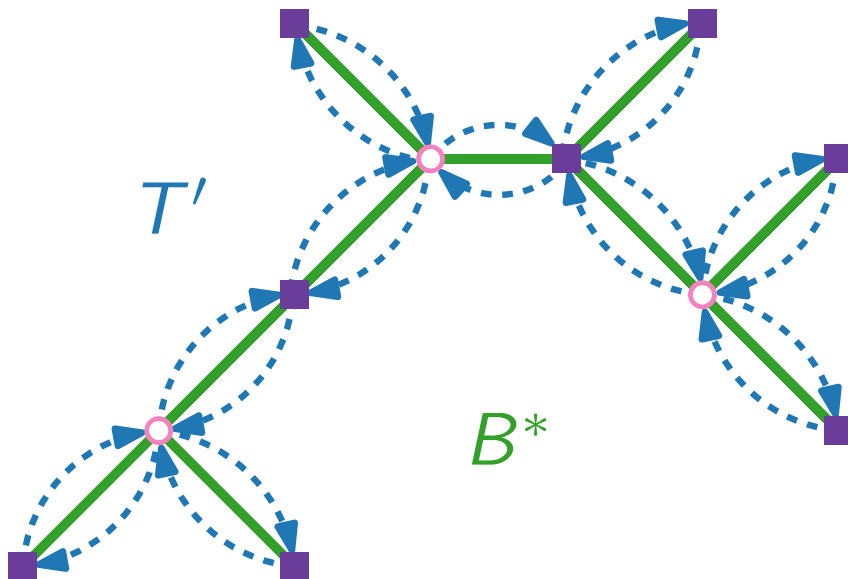
Consider an optimal Steiner tree B^* .

Duplicate all edges of B^* .

$\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \text{OPT}$.

Find a Eulerian tour T' in $B' \Rightarrow c(T') = c(B') = 2 \cdot \text{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and previously visited terminals.



Proof of Approximation Factor

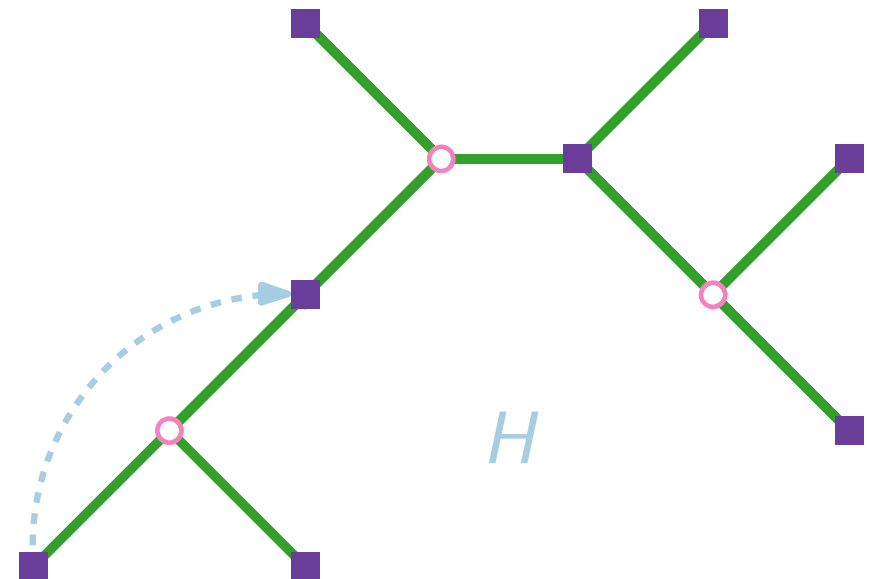
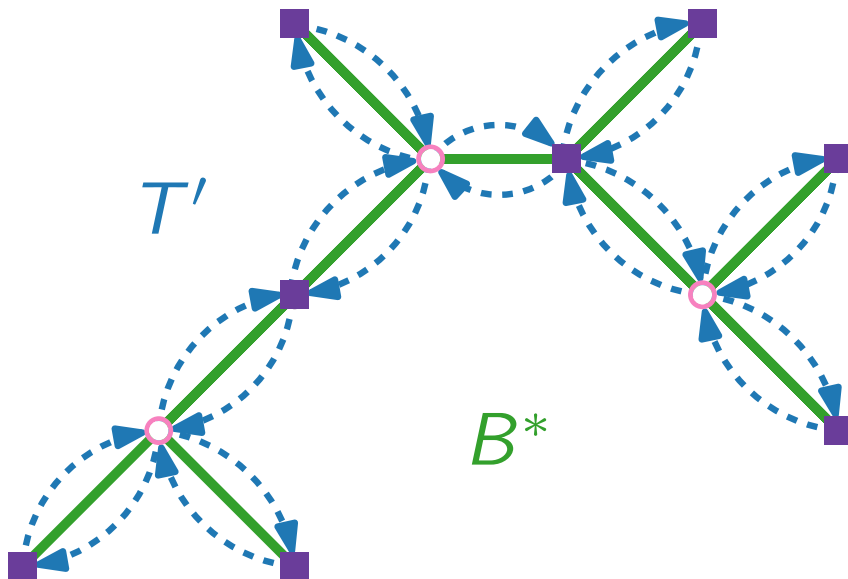
Consider an optimal Steiner tree B^* .

Duplicate all edges of B^* .

$\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \text{OPT}$.

Find a Eulerian tour T' in $B' \Rightarrow c(T') = c(B') = 2 \cdot \text{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and previously visited terminals.



Proof of Approximation Factor

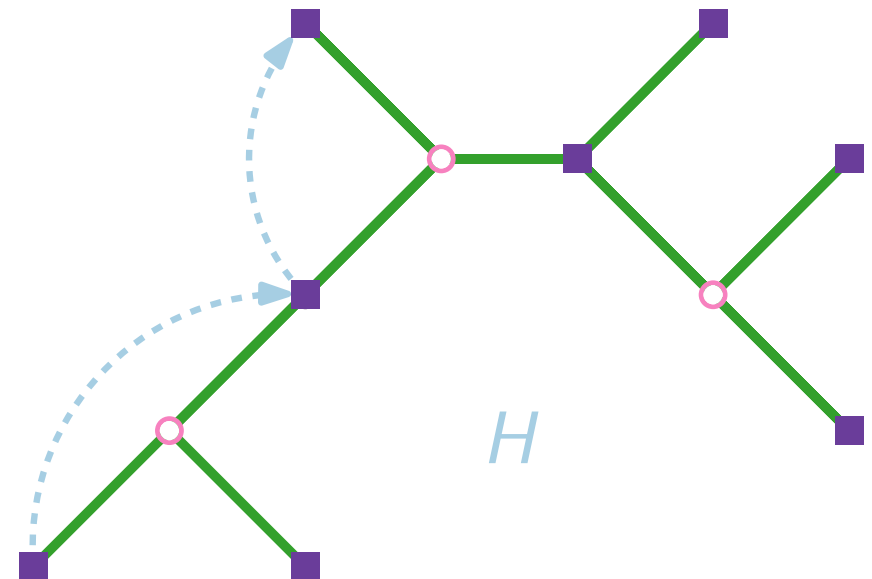
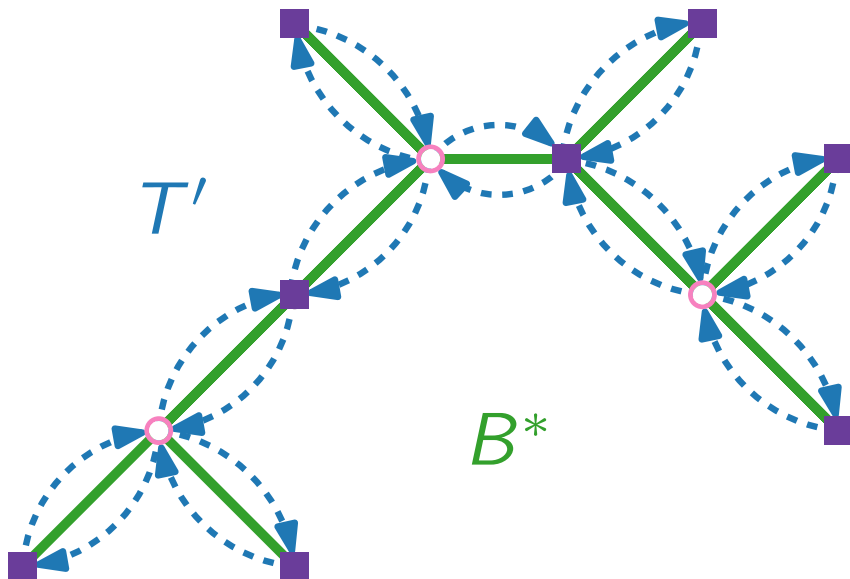
Consider an optimal Steiner tree B^* .

Duplicate all edges of B^* .

$\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \text{OPT}$.

Find a Eulerian tour T' in $B' \Rightarrow c(T') = c(B') = 2 \cdot \text{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and previously visited terminals.



Proof of Approximation Factor

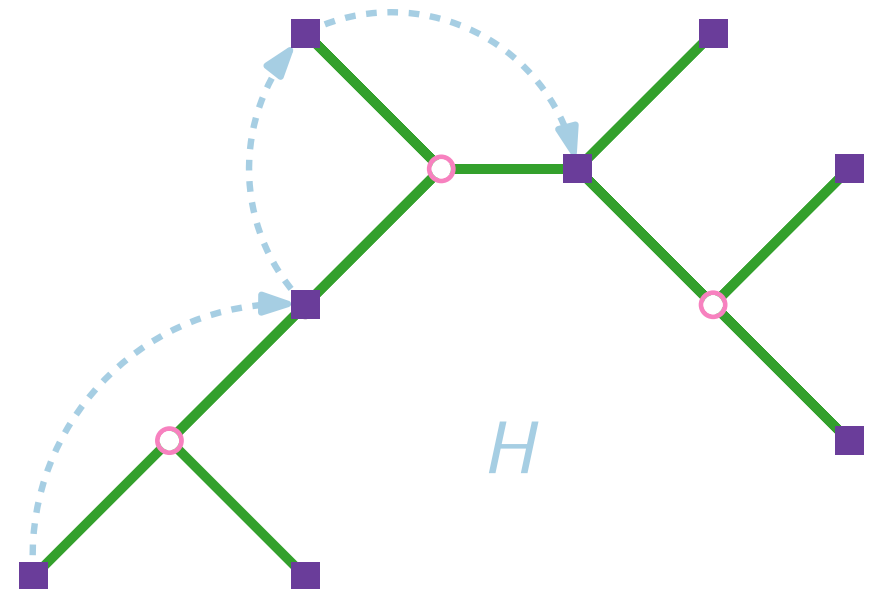
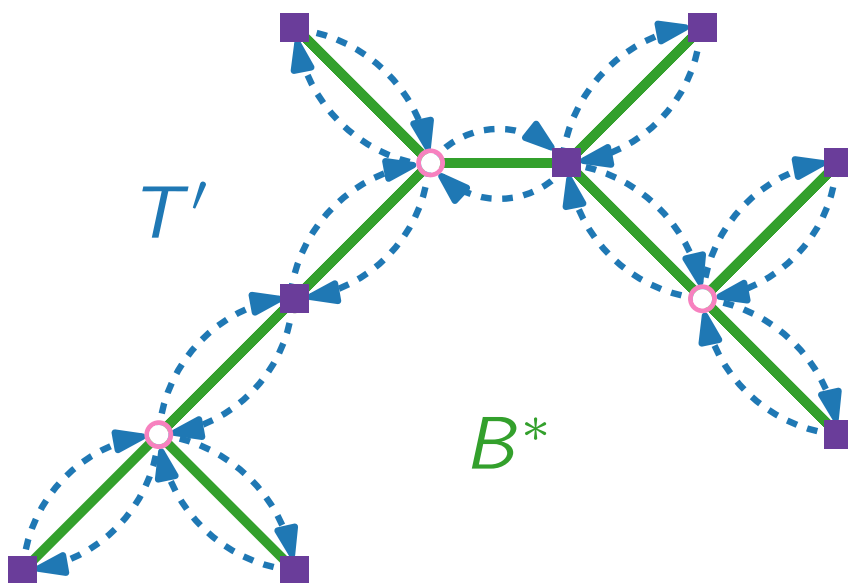
Consider an optimal Steiner tree B^* .

Duplicate all edges of B^* .

$\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \text{OPT}$.

Find a Eulerian tour T' in $B' \Rightarrow c(T') = c(B') = 2 \cdot \text{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and previously visited terminals.



Proof of Approximation Factor

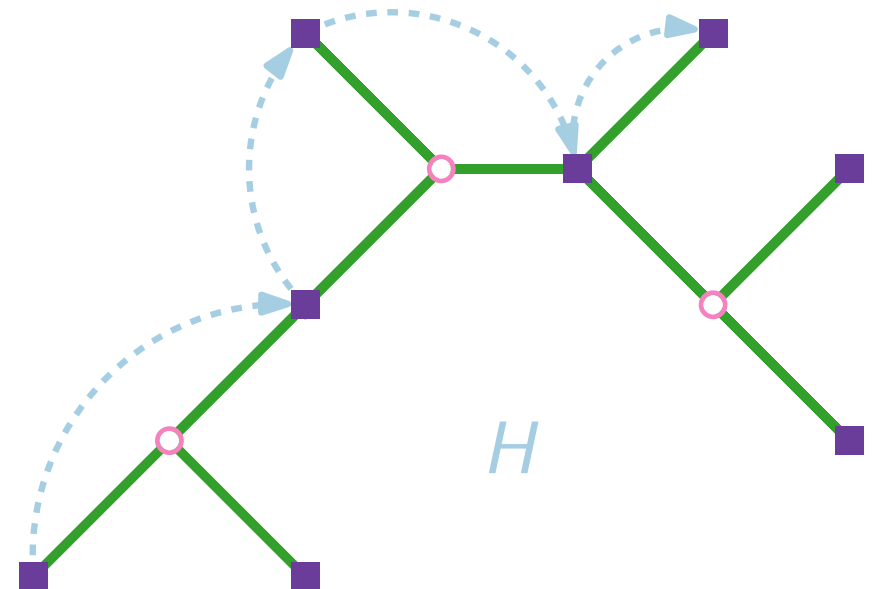
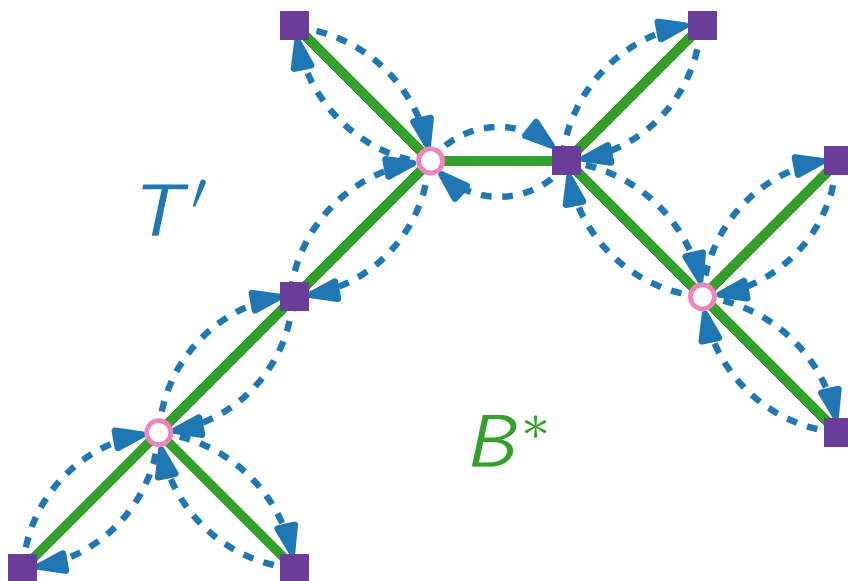
Consider an optimal Steiner tree B^* .

Duplicate all edges of B^* .

$\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \text{OPT}$.

Find a Eulerian tour T' in $B' \Rightarrow c(T') = c(B') = 2 \cdot \text{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and previously visited terminals.



Proof of Approximation Factor

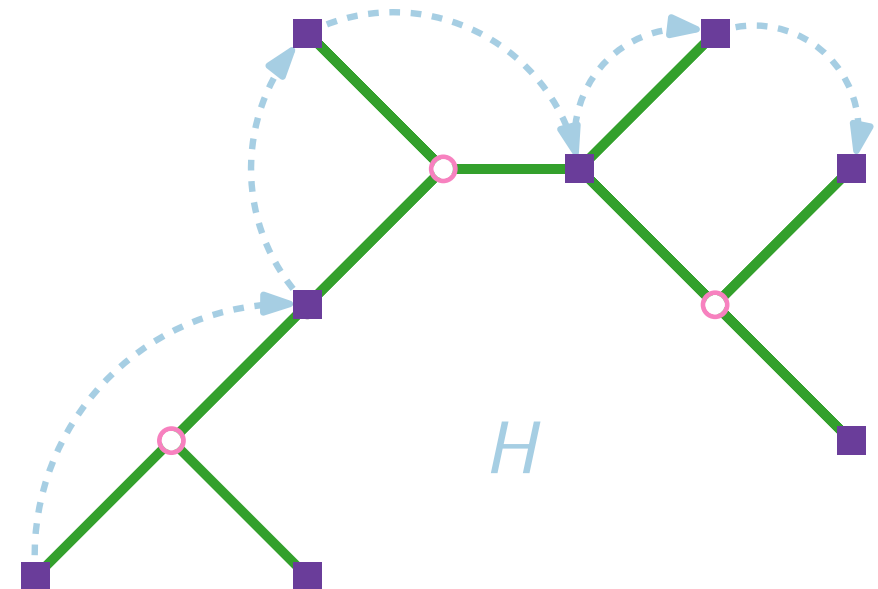
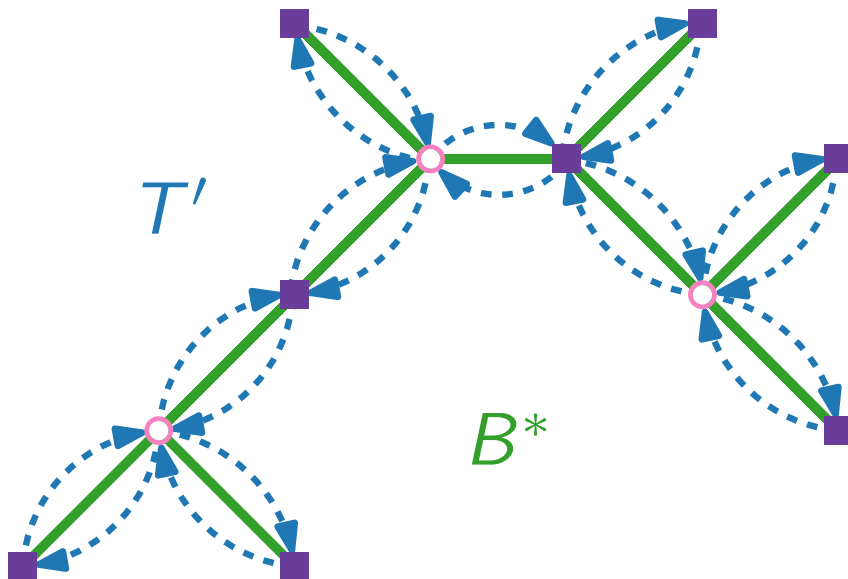
Consider an optimal Steiner tree B^* .

Duplicate all edges of B^* .

$\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \text{OPT}$.

Find a Eulerian tour T' in $B' \Rightarrow c(T') = c(B') = 2 \cdot \text{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and previously visited terminals.



Proof of Approximation Factor

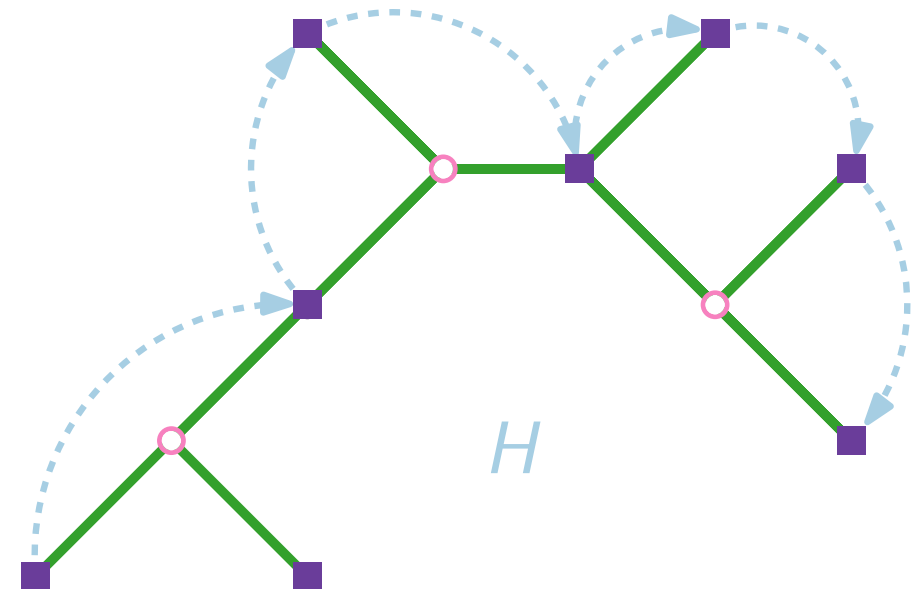
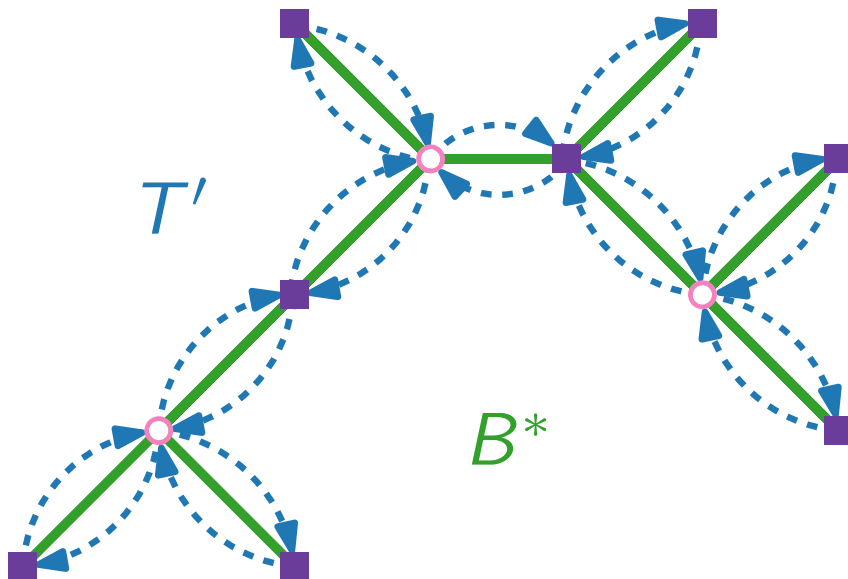
Consider an optimal Steiner tree B^* .

Duplicate all edges of B^* .

$\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \text{OPT}$.

Find a Eulerian tour T' in $B' \Rightarrow c(T') = c(B') = 2 \cdot \text{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and previously visited terminals.



Proof of Approximation Factor

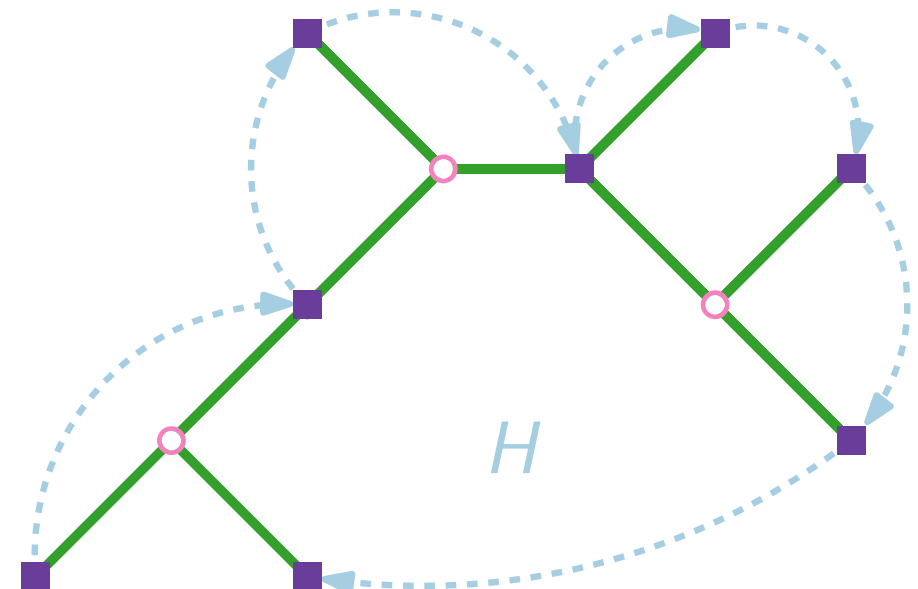
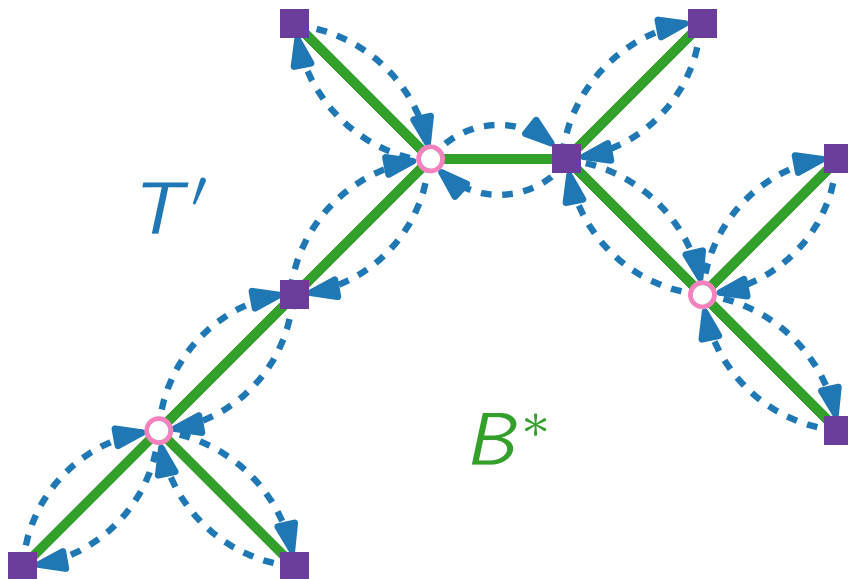
Consider an optimal Steiner tree B^* .

Duplicate all edges of B^* .

$\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \text{OPT}$.

Find a Eulerian tour T' in $B' \Rightarrow c(T') = c(B') = 2 \cdot \text{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and previously visited terminals.



Proof of Approximation Factor

Consider an optimal Steiner tree B^* .

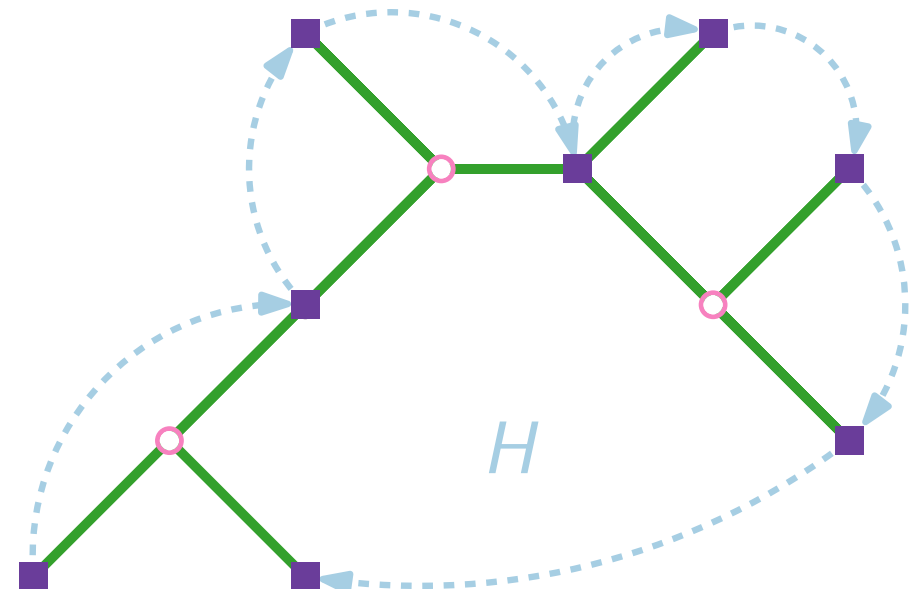
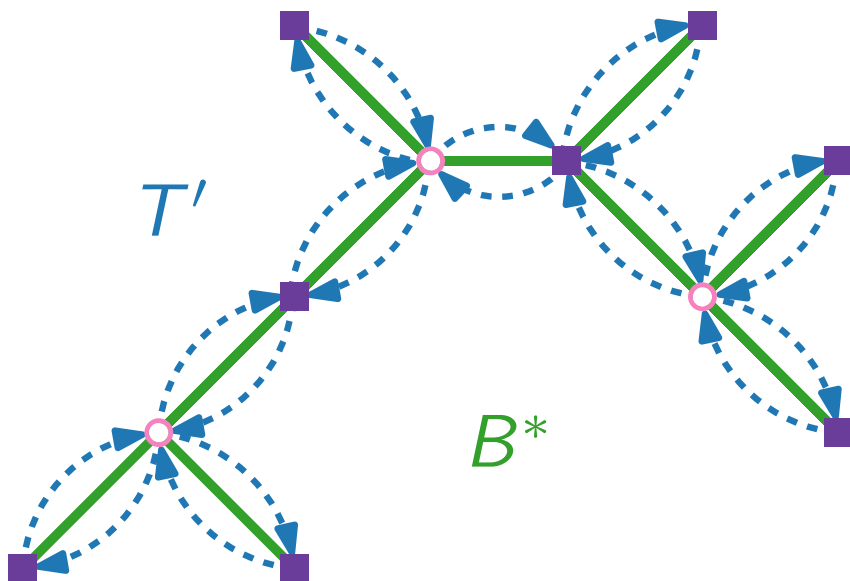
Duplicate all edges of B^* .

$\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \text{OPT}$.

Find a Eulerian tour T' in $B' \Rightarrow c(T') = c(B') = 2 \cdot \text{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and previously visited terminals.

$\Rightarrow c(H) \leq c(T') = 2 \cdot \text{OPT}$ since G is metric.



Proof of Approximation Factor

Consider an optimal Steiner tree B^* .

Duplicate all edges of B^* .

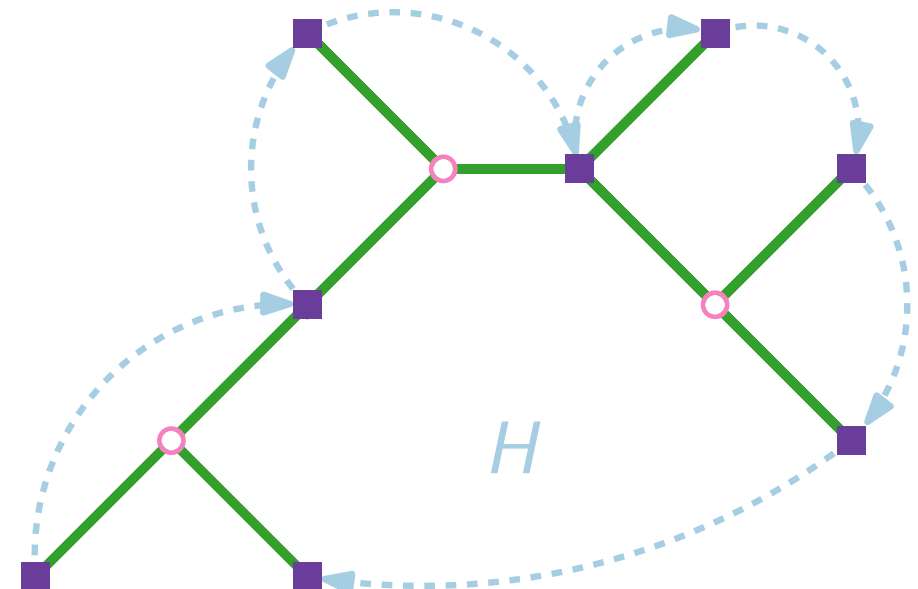
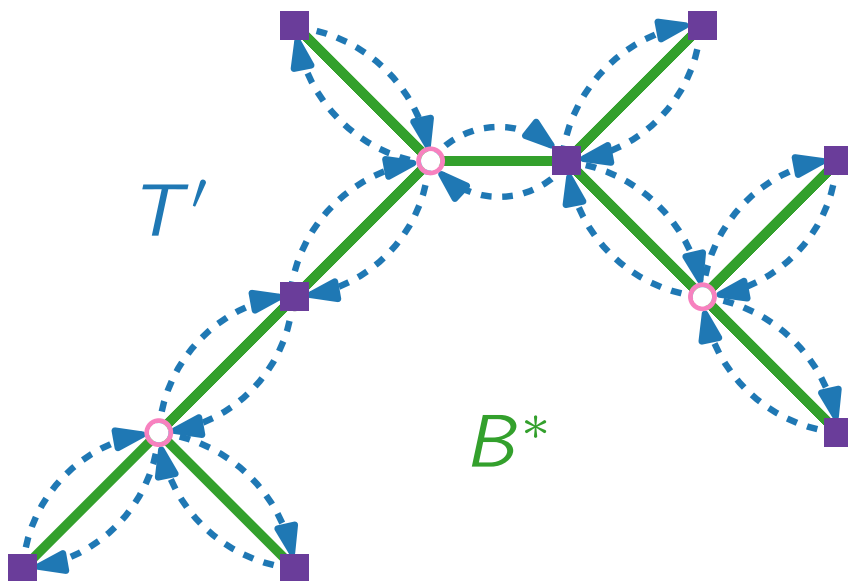
$\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \text{OPT}$.

Find a Eulerian tour T' in $B' \Rightarrow c(T') = c(B') = 2 \cdot \text{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and previously visited terminals.

$\Rightarrow c(H) \leq c(T') = 2 \cdot \text{OPT}$ since G is metric.

MST B of $G[T]$ costs $c(B) \leq c(H) \leq 2 \cdot \text{OPT}$



Proof of Approximation Factor

Consider an optimal Steiner tree B^* .

Duplicate all edges of B^* .

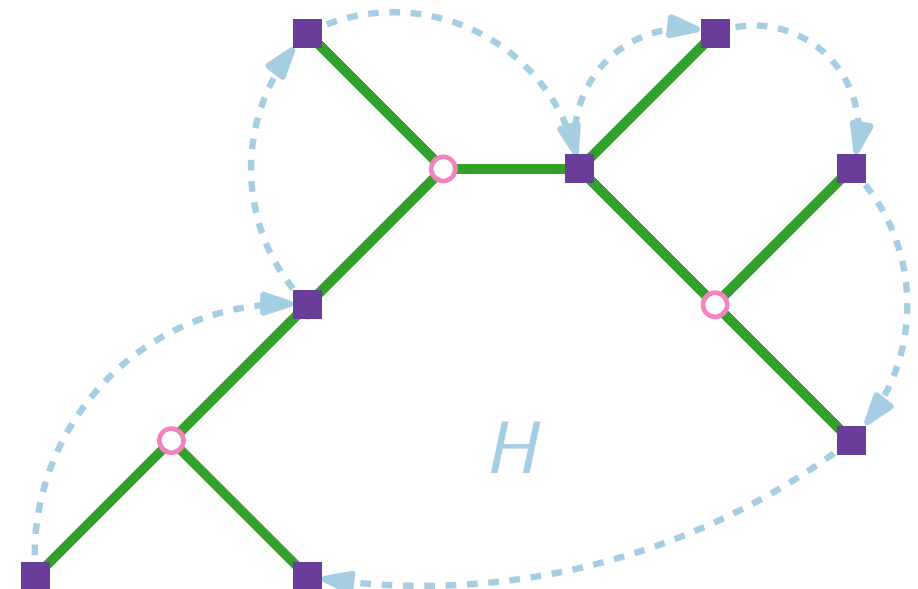
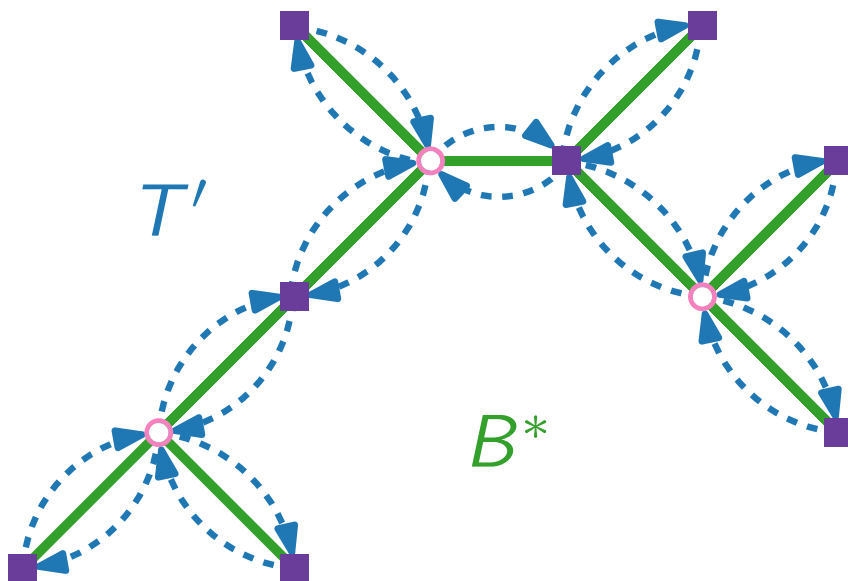
$\Rightarrow$ Eulerian (multi-)graph B' with cost $c(B') = 2 \cdot \text{OPT}$.

Find a Eulerian tour T' in $B' \Rightarrow c(T') = c(B') = 2 \cdot \text{OPT}$

Find a Hamiltonian path H in $G[T]$ by “short-cutting” Steiner vertices and previously visited terminals.

$\Rightarrow c(H) \leq c(T') = 2 \cdot \text{OPT}$ since G is metric.

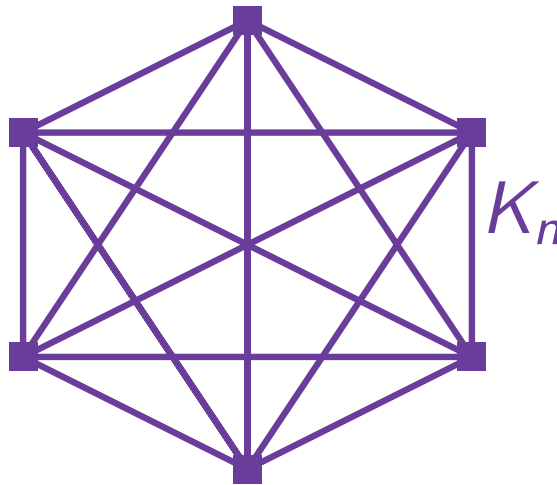
MST B of $G[T]$ costs $c(B) \leq c(H) \leq 2 \cdot \text{OPT}$
since H is a spanning tree of $G[T]$.



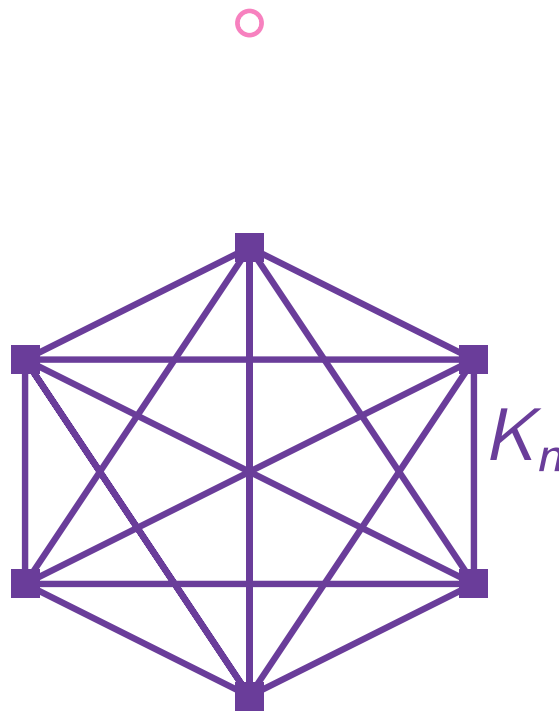
Analysis Tight?

Analysis Tight?

- terminal



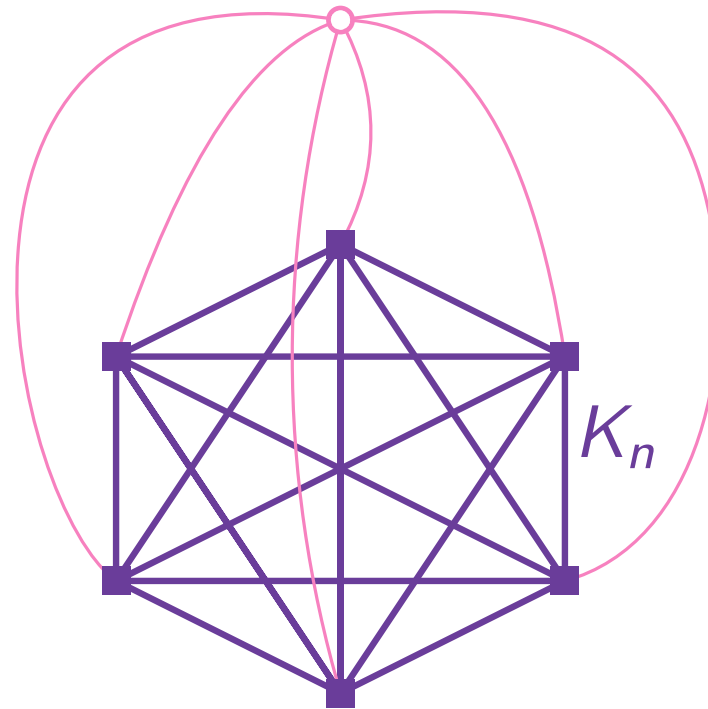
Analysis Tight?



■ terminal

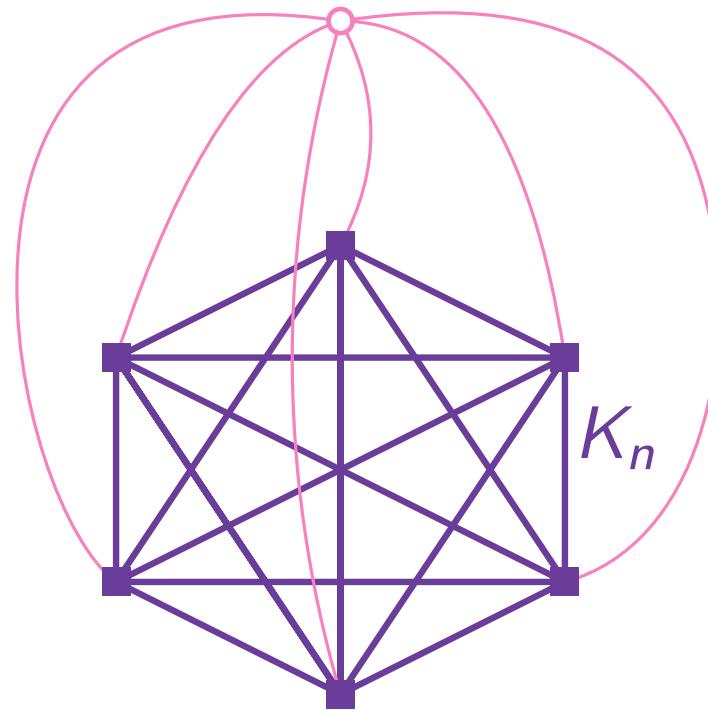
○ Steiner vertex

Analysis Tight?



- terminal
- Steiner vertex

Analysis Tight?



■ terminal

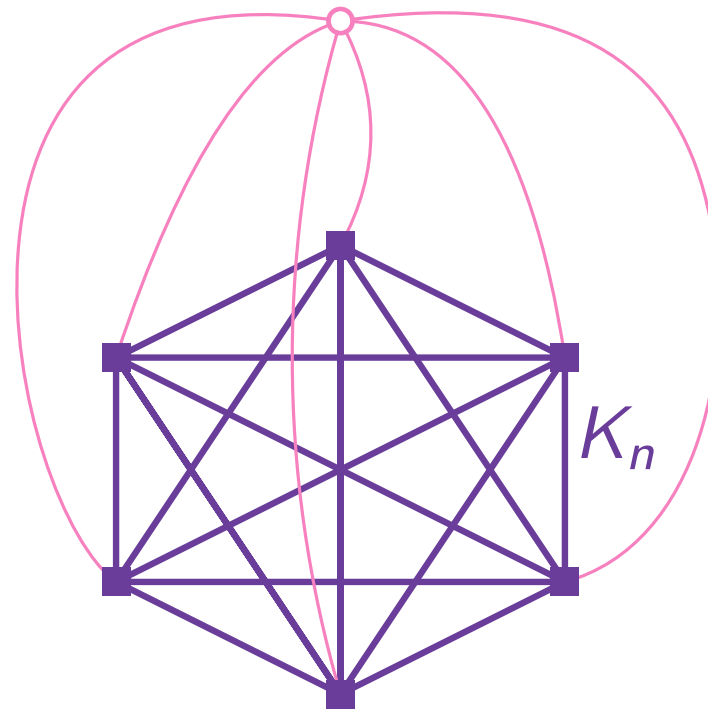
○ Steiner vertex

— cost

— cost



Analysis Tight?



■ terminal

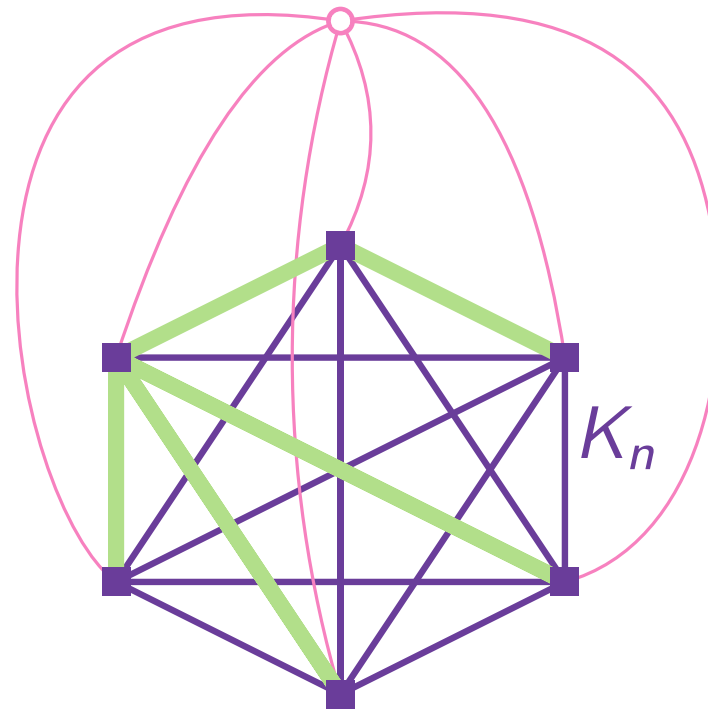
○ Steiner vertex

— cost 1

— cost 2

Analysis Tight?

An **MST** of $G[T]$ has cost



■ terminal

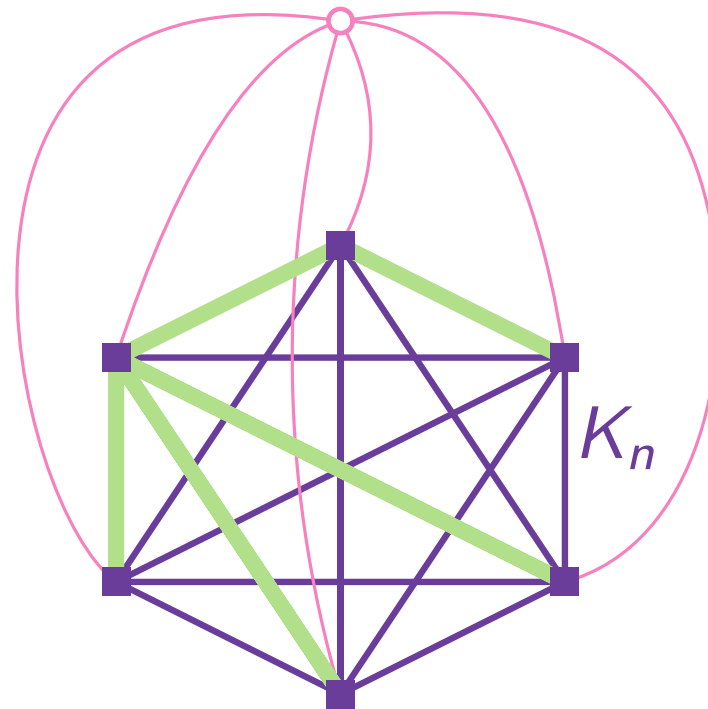
○ Steiner vertex

— cost 1

— cost 2

Analysis Tight?

An **MST** of $G[T]$ has cost $2(n - 1)$.



■ terminal

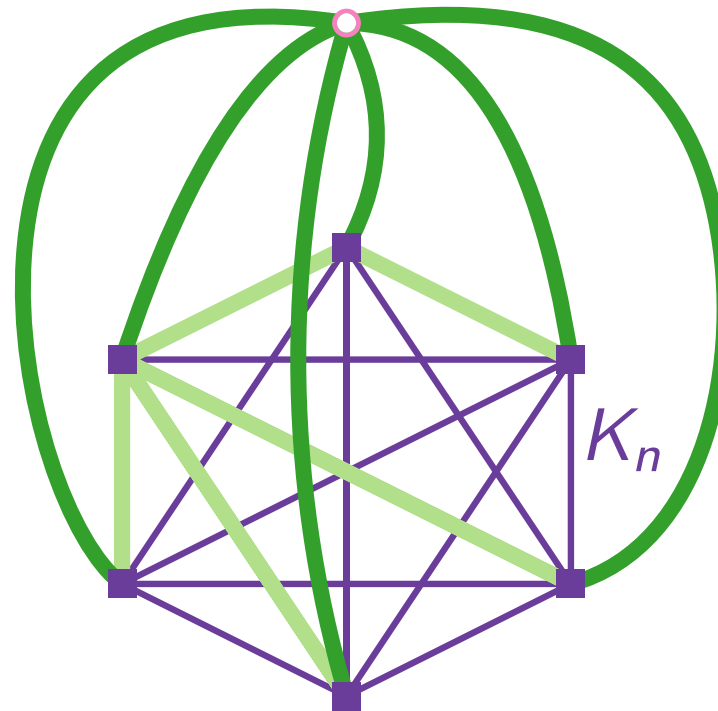
○ Steiner vertex

— cost 1

— cost 2

Analysis Tight?

An **MST** of $G[T]$ has cost $2(n - 1)$.
 The **optimal solution** has cost n .

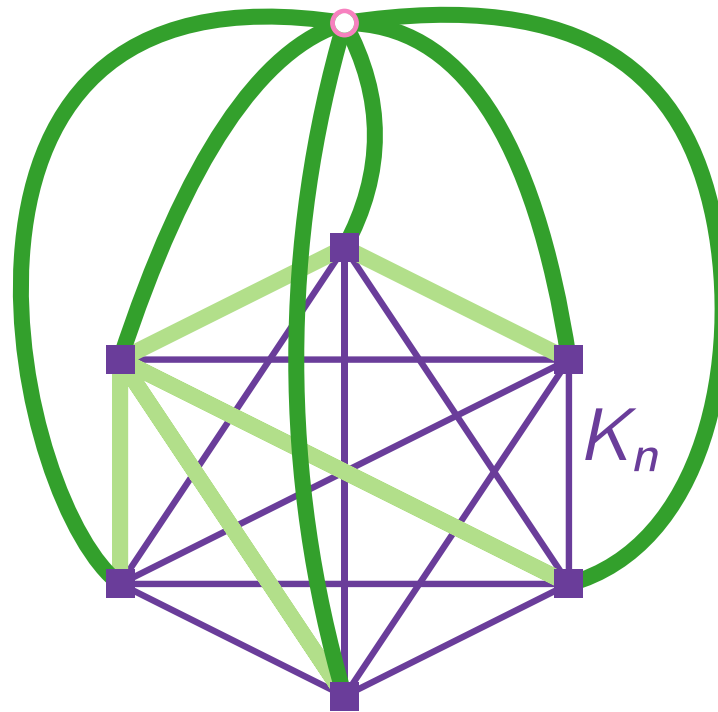


- terminal
- Steiner vertex
- cost 1
- cost 2

Analysis Tight?

An **MST** of $G[T]$ has cost $2(n-1)$.
 The **optimal solution** has cost n .

$$\frac{2(n-1)}{n} \rightarrow 2$$



■ terminal

○ Steiner vertex

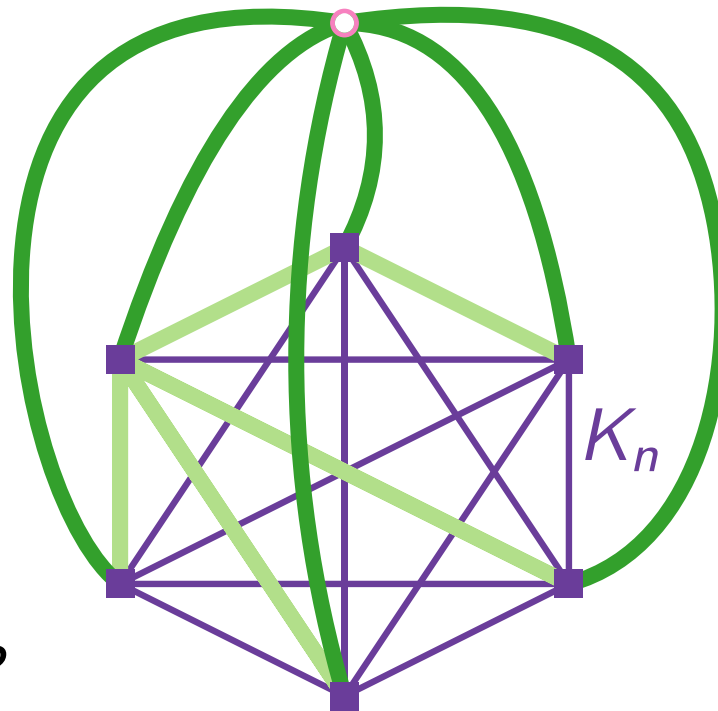
— cost 1

— cost 2

Analysis Tight?

An **MST** of $G[T]$ has cost $2(n-1)$.
The **optimal solution** has cost n .

$$\frac{2(n-1)}{n} \rightarrow 2$$



■ terminal
○ Steiner vertex

— cost 1

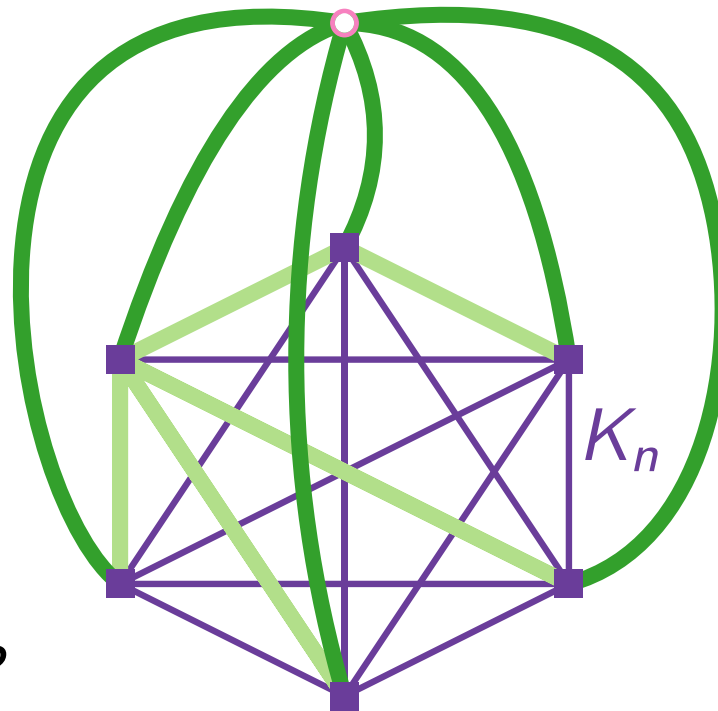
— cost 2

Can we do better?

Analysis Tight?

An **MST** of $G[T]$ has cost $2(n - 1)$.
The **optimal solution** has cost n .

$$\frac{2(n - 1)}{n} \rightarrow 2$$



■ terminal
○ Steiner vertex

— cost 1

— cost 2

Can we do better?

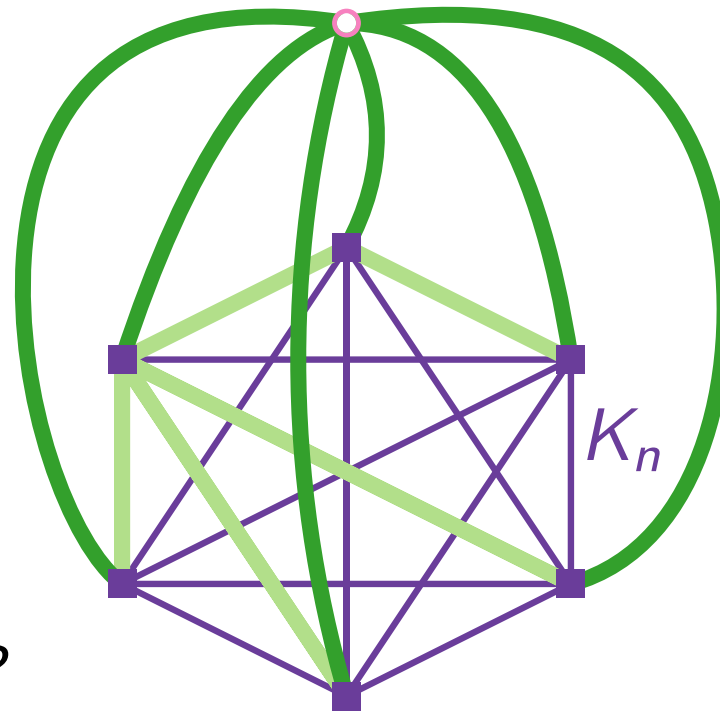
The best known approximation factor for STEINERTREE is $\ln(4) + \varepsilon \approx 1.39$.

[Byrka, Grandoni, Rothvoß & Sanità, J. ACM'13]

Analysis Tight?

An **MST** of $G[T]$ has cost $2(n - 1)$.
The **optimal solution** has cost n .

$$\frac{2(n - 1)}{n} \rightarrow 2$$



■ terminal
○ Steiner vertex

— cost 1

— cost 2

Can we do better?

The best known approximation factor for STEINERTREE is $\ln(4) + \varepsilon \approx 1.39$.

[Byrka, Grandoni, Rothvoß & Sanità, J. ACM'13]

STEINERTREE cannot be approximated within factor $\frac{96}{95} \approx 1.0105$ (unless $P = NP$).

[Chlebík & Chlebíková, TCS'08]

Approximation Algorithms

Lecture 3:

STEINERTREE and MULTIWAYCUT

Part V:

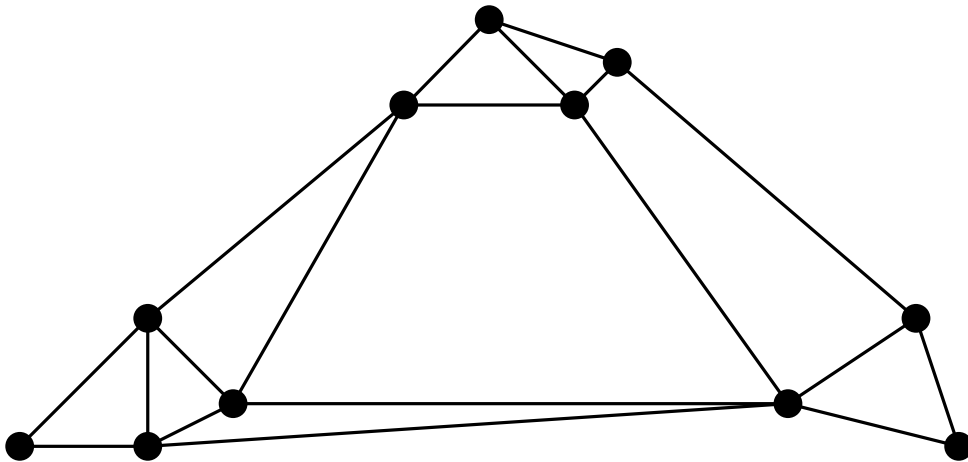
MULTIWAYCUT

MULTIWAYCUT

Given: A connected graph G

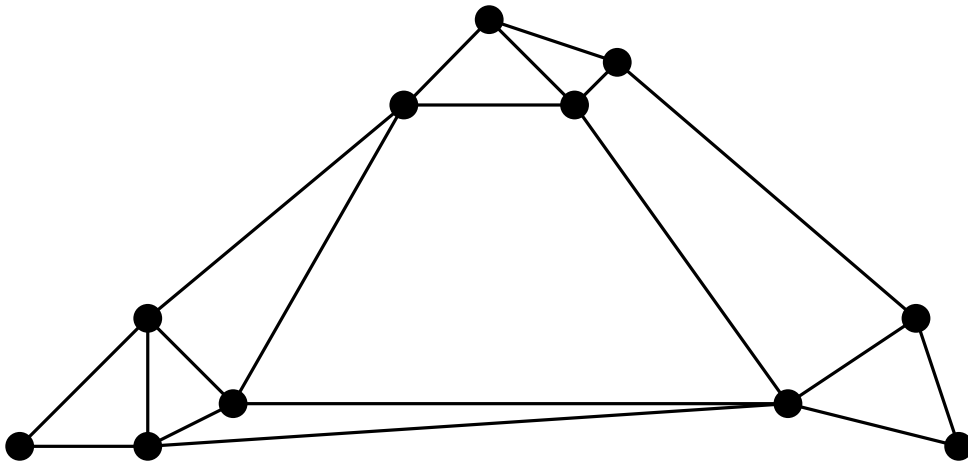
MULTIWAYCUT

Given: A connected graph G



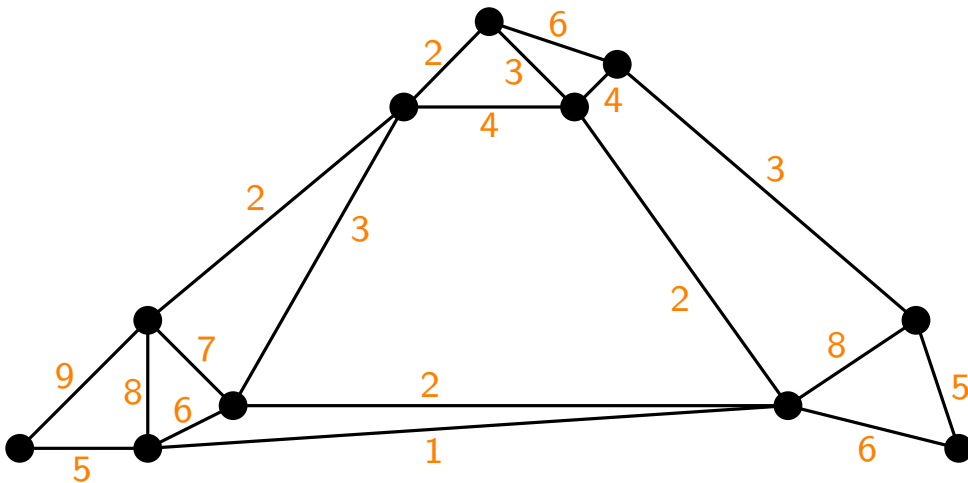
MULTIWAYCUT

Given: A connected graph G with edge costs $c: E(G) \rightarrow \mathbb{Q}^+$



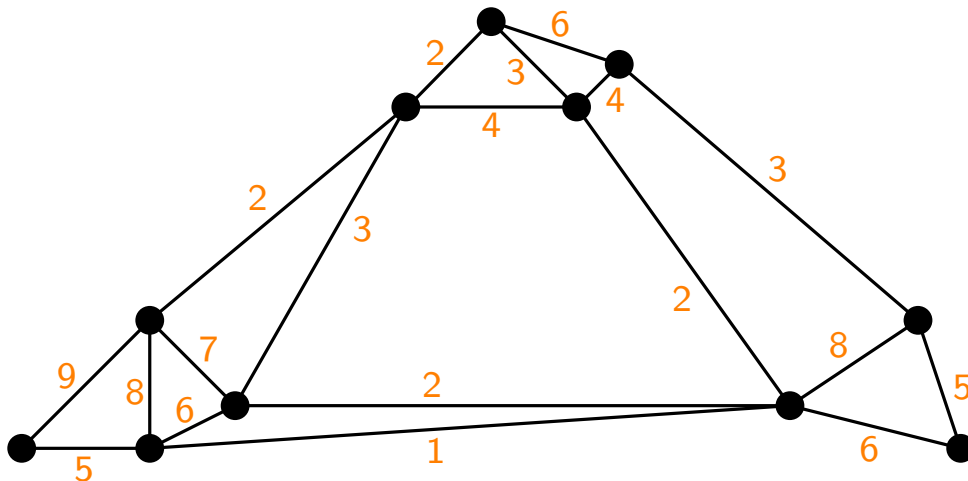
MULTIWAYCUT

Given: A connected graph G with edge costs $c: E(G) \rightarrow \mathbb{Q}^+$



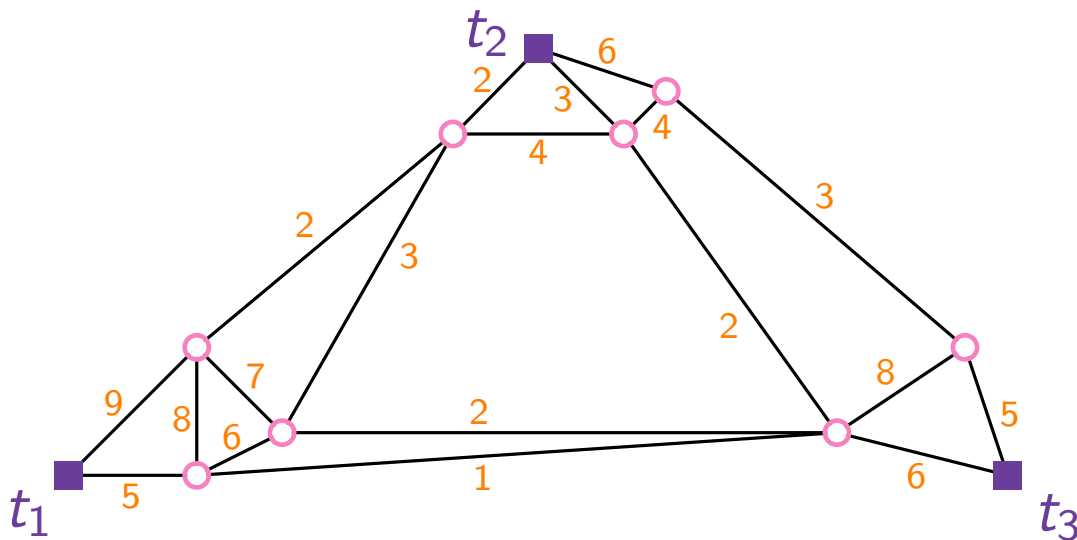
MULTIWAYCUT

Given: A connected graph G with edge costs $c: E(G) \rightarrow \mathbb{Q}^+$ and a set $T = \{t_1, \dots, t_k\} \subseteq V(G)$ of **terminals**.



MULTIWAYCUT

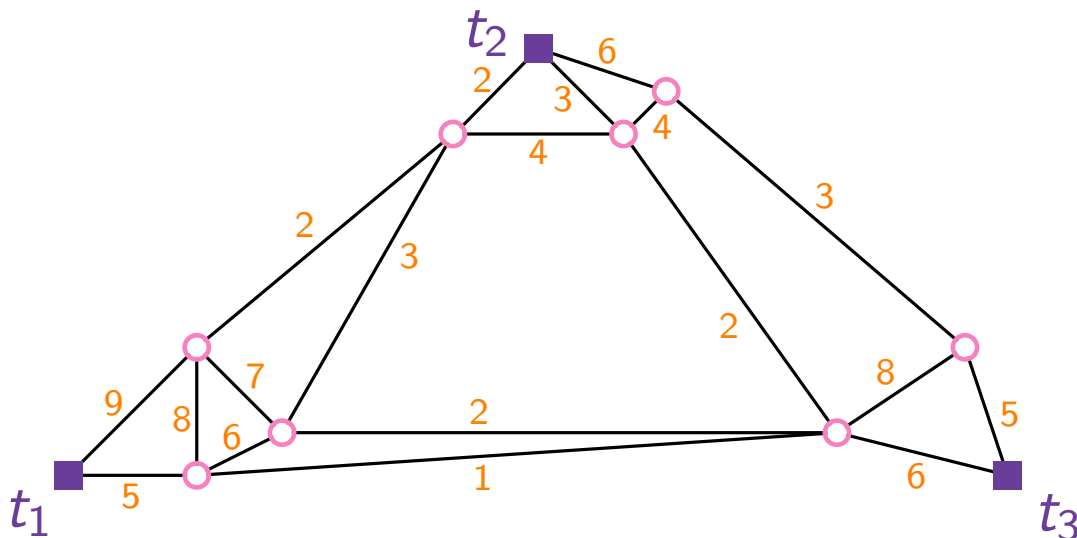
Given: A connected graph G with **edge costs** $c: E(G) \rightarrow \mathbb{Q}^+$ and a set $T = \{t_1, \dots, t_k\} \subseteq V(G)$ of **terminals**.



MULTIWAYCUT

Given: A connected graph G with **edge costs** $c: E(G) \rightarrow \mathbb{Q}^+$ and a set $T = \{t_1, \dots, t_k\} \subseteq V(G)$ of **terminals**.

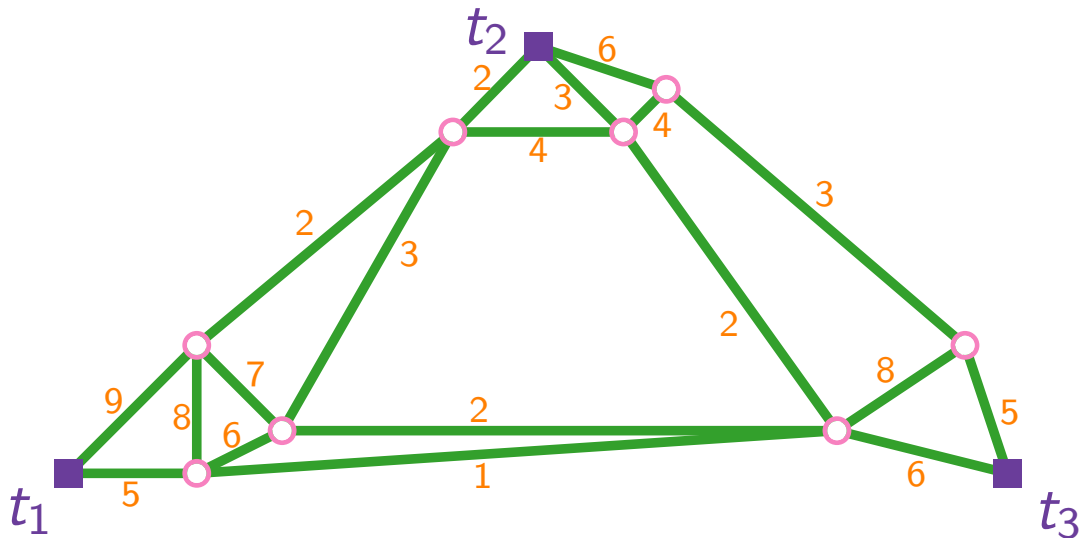
A **multiway cut** of T is a subset E' of edges such that no two terminals in the graph $(V(G), E(G) - E')$ are connected.



MULTIWAYCUT

Given: A connected graph G with **edge costs** $c: E(G) \rightarrow \mathbb{Q}^+$ and a set $T = \{t_1, \dots, t_k\} \subseteq V(G)$ of **terminals**.

A **multiway cut** of T is a subset E' of edges such that no two terminals in the graph $(V(G), E(G) - E')$ are connected.

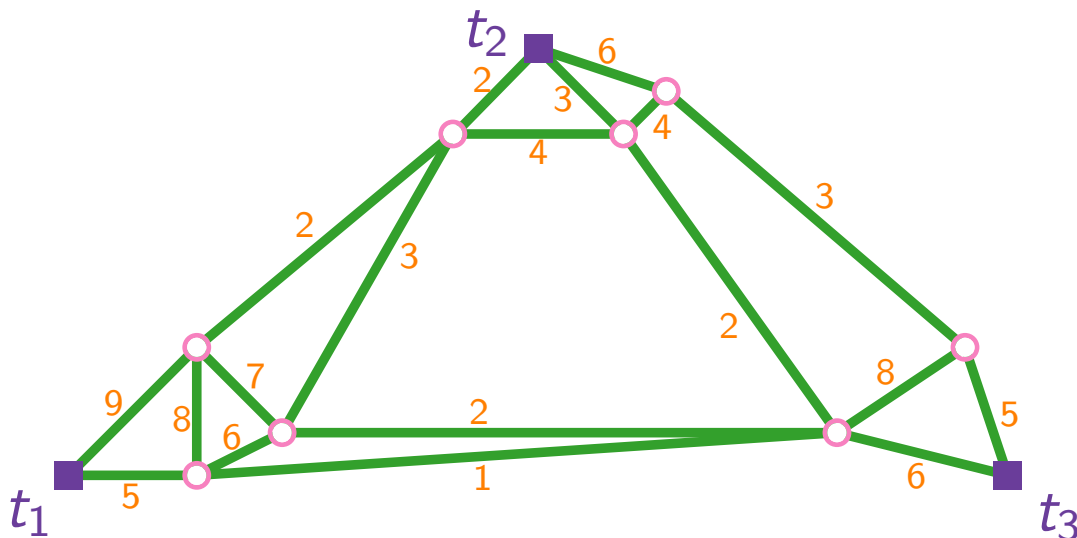


MULTIWAYCUT

Given: A connected graph G with **edge costs** $c: E(G) \rightarrow \mathbb{Q}^+$ and a set $T = \{t_1, \dots, t_k\} \subseteq V(G)$ of **terminals**.

A **multiway cut** of T is a subset E' of edges such that no two terminals in the graph $(V(G), E(G) - E')$ are connected.

Find: A **minimum-cost multiway cut** of T .

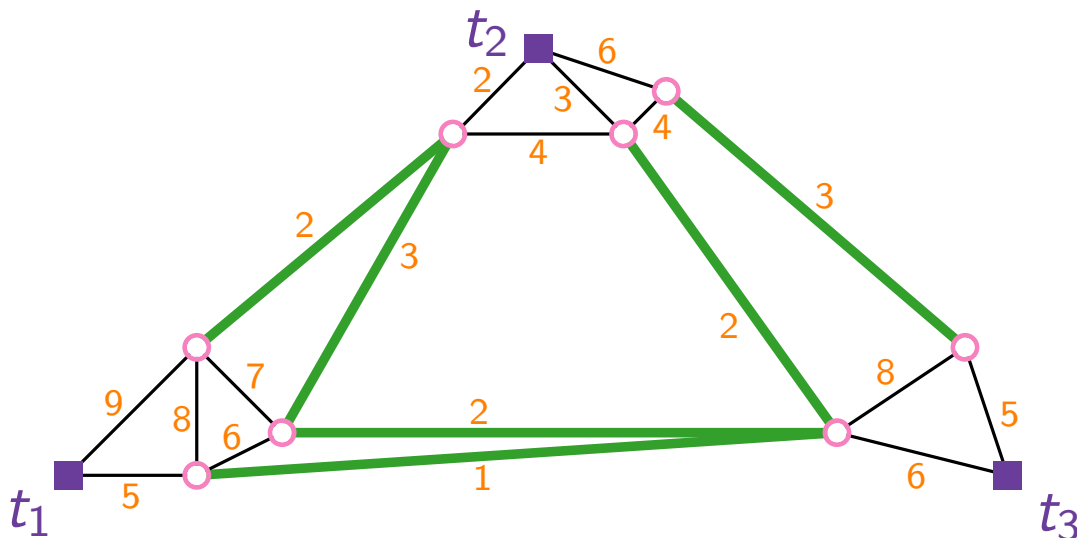


MULTIWAYCUT

Given: A connected graph G with **edge costs** $c: E(G) \rightarrow \mathbb{Q}^+$ and a set $T = \{t_1, \dots, t_k\} \subseteq V(G)$ of **terminals**.

A **multiway cut** of T is a subset E' of edges such that no two terminals in the graph $(V(G), E(G) - E')$ are connected.

Find: A **minimum-cost multiway cut** of T .

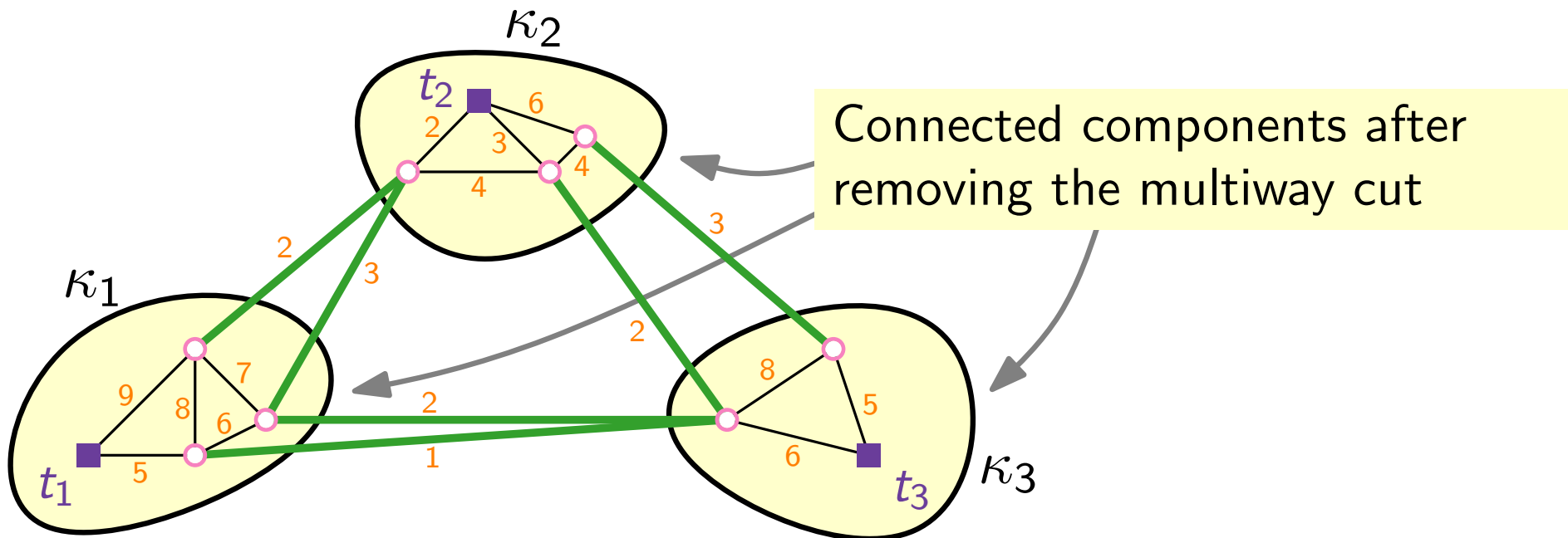


MULTIWAYCUT

Given: A connected graph G with **edge costs** $c: E(G) \rightarrow \mathbb{Q}^+$ and a set $T = \{t_1, \dots, t_k\} \subseteq V(G)$ of **terminals**.

A **multiway cut** of T is a subset E' of edges such that no two terminals in the graph $(V(G), E(G) - E')$ are connected.

Find: A **minimum-cost multiway cut** of T .

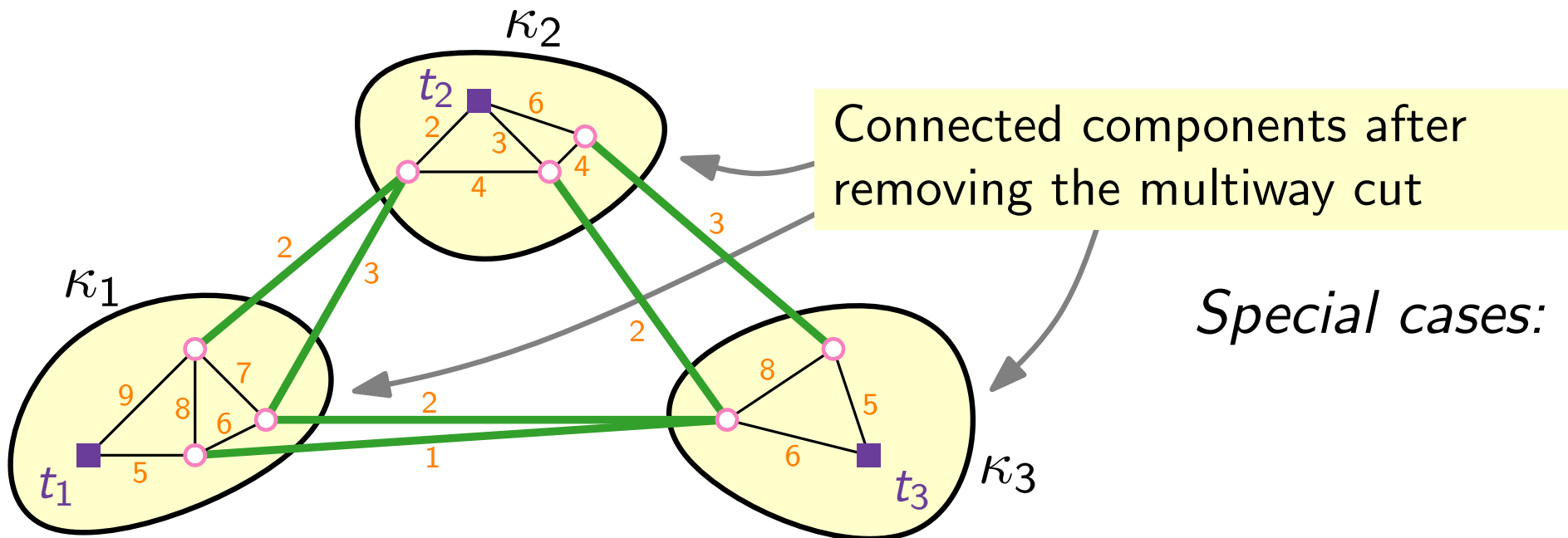


MULTIWAYCUT

Given: A connected graph G with **edge costs** $c: E(G) \rightarrow \mathbb{Q}^+$ and a set $T = \{t_1, \dots, t_k\} \subseteq V(G)$ of **terminals**.

A **multiway cut** of T is a subset E' of edges such that no two terminals in the graph $(V(G), E(G) - E')$ are connected.

Find: A **minimum-cost multiway cut** of T .

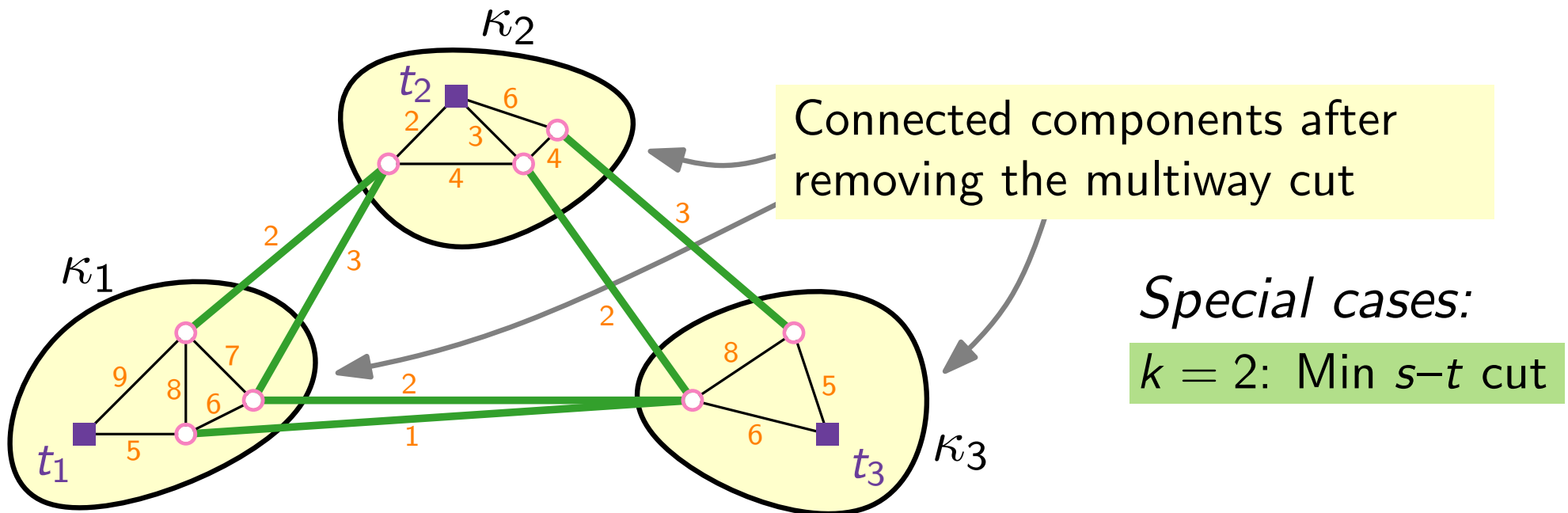


MULTIWAYCUT

Given: A connected graph G with **edge costs** $c: E(G) \rightarrow \mathbb{Q}^+$ and a set $T = \{t_1, \dots, t_k\} \subseteq V(G)$ of **terminals**.

A **multiway cut** of T is a subset E' of edges such that no two terminals in the graph $(V(G), E(G) - E')$ are connected.

Find: A **minimum-cost multiway cut** of T .

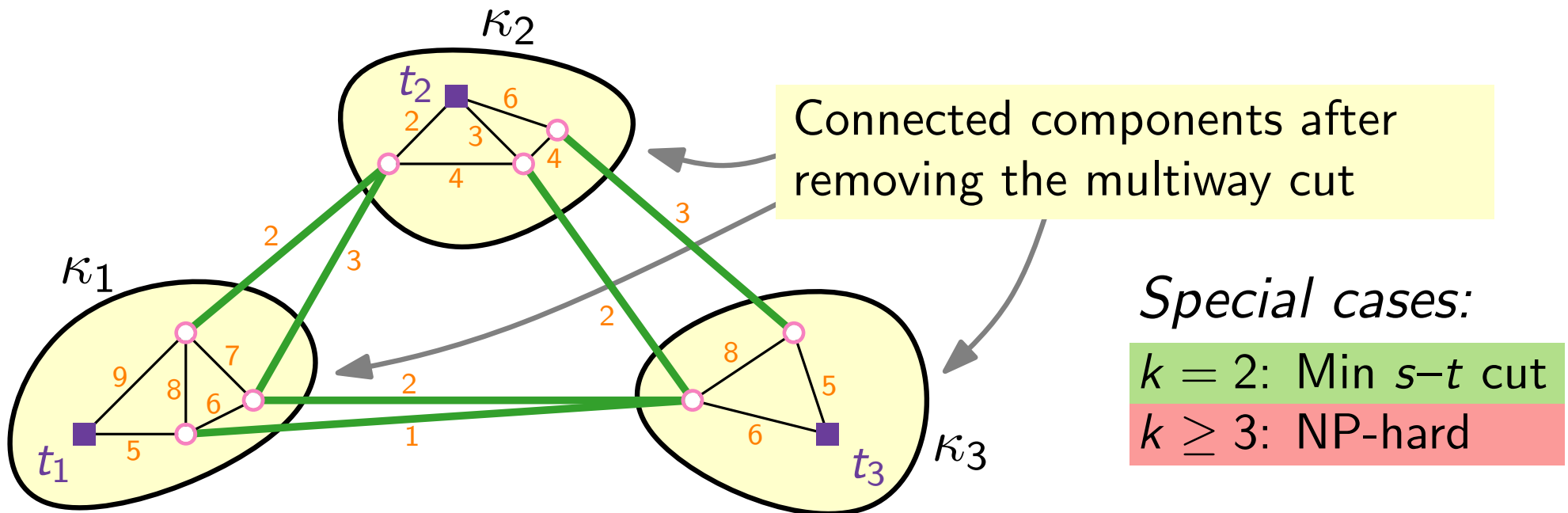


MULTIWAYCUT

Given: A connected graph G with **edge costs** $c: E(G) \rightarrow \mathbb{Q}^+$ and a set $T = \{t_1, \dots, t_k\} \subseteq V(G)$ of **terminals**.

A **multiway cut** of T is a subset E' of edges such that no two terminals in the graph $(V(G), E(G) - E')$ are connected.

Find: A **minimum-cost multiway cut** of T .

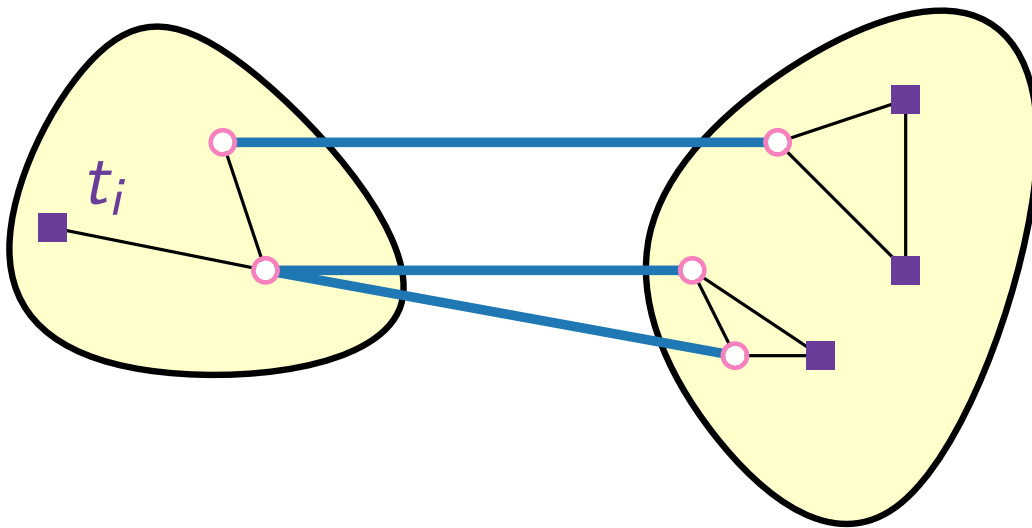


Isolating Cuts

An **isolating cut** for a terminal t_i is a set of edges that disconnects t_i from all other terminals.

Isolating Cuts

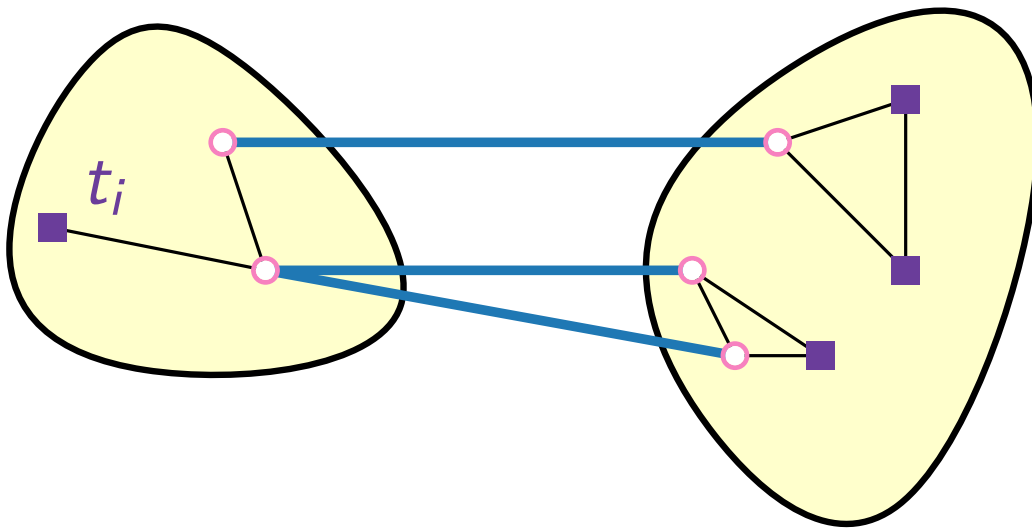
An **isolating cut** for a terminal t_i is a set of edges that disconnects t_i from all other terminals.



Isolating Cuts

An **isolating cut** for a terminal t_i is a set of edges that disconnects t_i from all other terminals.

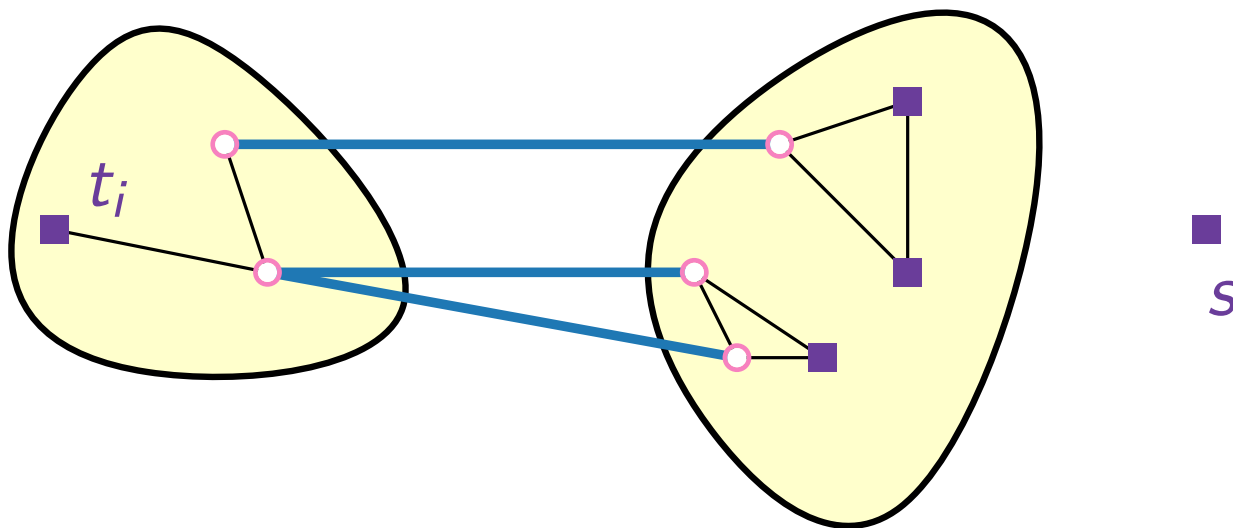
A minimum-cost **isolating cut** for t_i can be computed efficiently:



Isolating Cuts

An **isolating cut** for a terminal t_i is a set of edges that disconnects t_i from all other terminals.

A minimum-cost **isolating cut** for t_i can be computed efficiently:

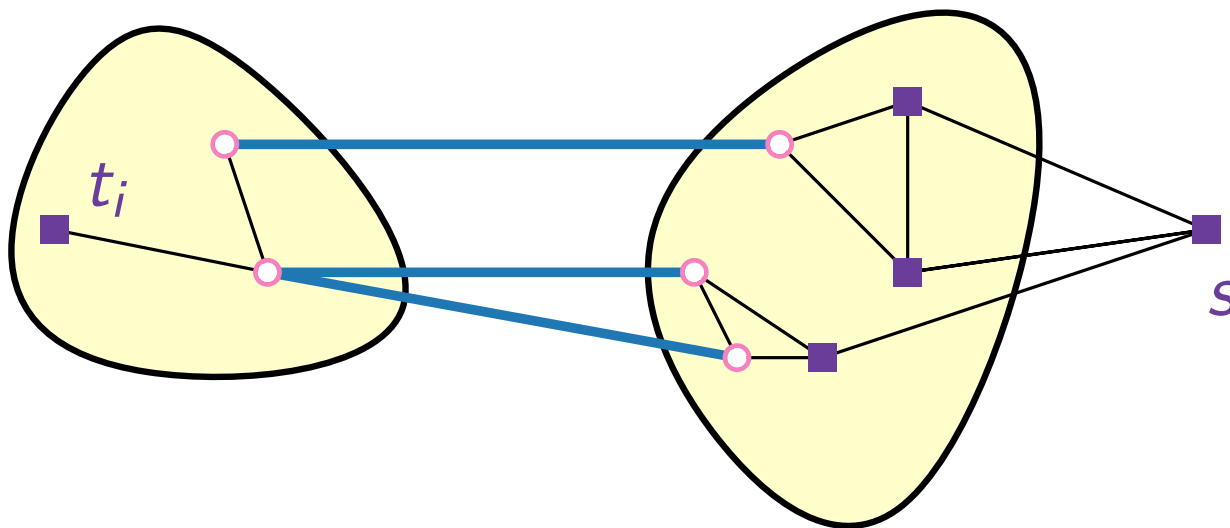


Add dummy terminal s

Isolating Cuts

An **isolating cut** for a terminal t_i is a set of edges that disconnects t_i from all other terminals.

A minimum-cost **isolating cut** for t_i can be computed efficiently:

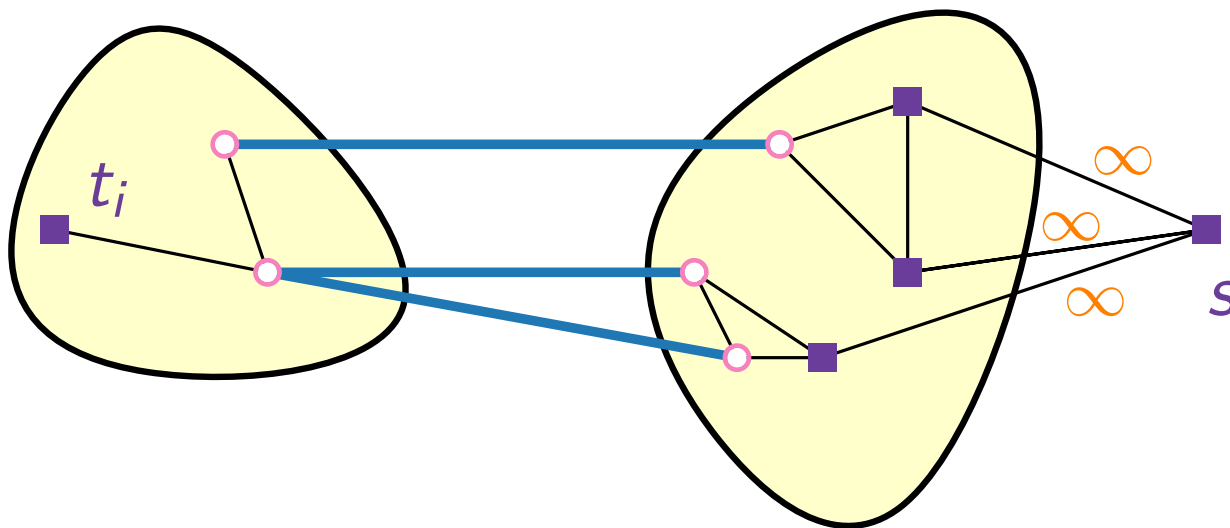


Add dummy terminal s

Isolating Cuts

An **isolating cut** for a terminal t_i is a set of edges that disconnects t_i from all other terminals.

A minimum-cost **isolating cut** for t_i can be computed efficiently:

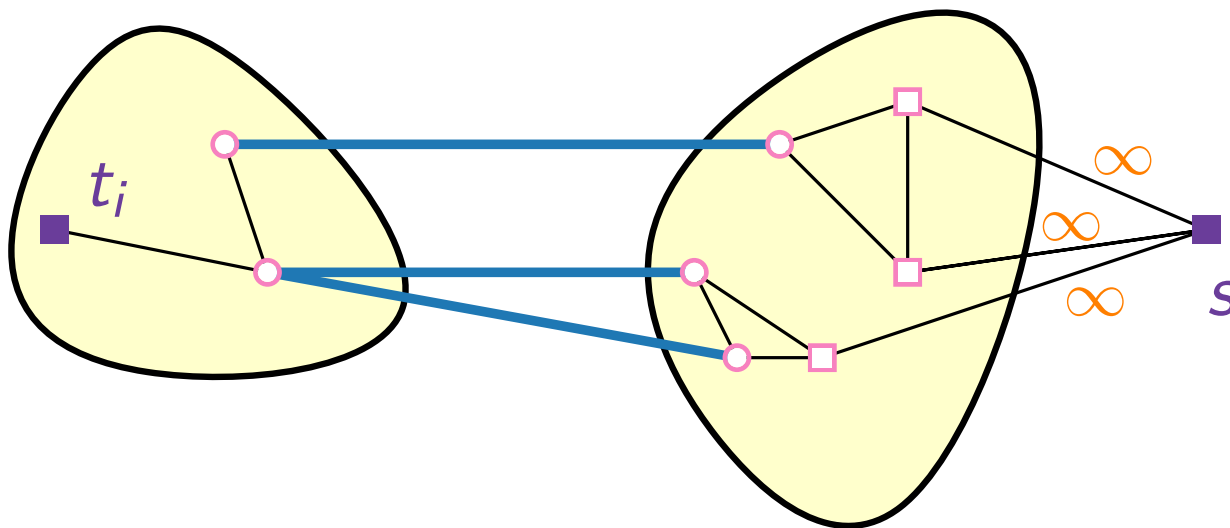


Add dummy terminal s

Isolating Cuts

An **isolating cut** for a terminal t_i is a set of edges that disconnects t_i from all other terminals.

A minimum-cost **isolating cut** for t_i can be computed efficiently:

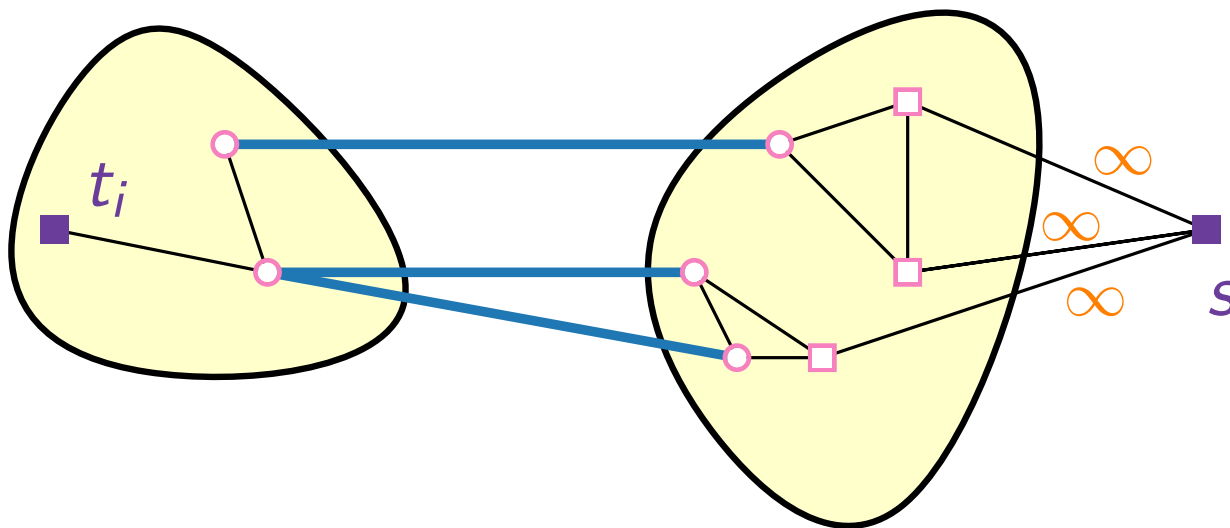


Add dummy terminal s

Isolating Cuts

An **isolating cut** for a terminal t_i is a set of edges that disconnects t_i from all other terminals.

A minimum-cost **isolating cut** for t_i can be computed efficiently:



Add dummy terminal s and find a minimum-cost $s-t_i$ cut.

Approximation Algorithms

Lecture 3:

STEINERTREE and MULTIWAYCUT

Part VI:

Algorithm for MULTIWAYCUT

Algorithm MULTIWAYCUT

For $i = 1, \dots, k$:

Algorithm MULTIWAYCUT

For $i = 1, \dots, k$:

- Compute a minimum-cost isolating cut C_i for t_i .

Algorithm MULTIWAYCUT

For $i = 1, \dots, k$:

- Compute a minimum-cost isolating cut C_i for t_i .
- Return the union $\mathcal{C}$ of the $k - 1$ cheapest such isolating cuts.

Algorithm MULTIWAYCUT

For $i = 1, \dots, k$:

- Compute a minimum-cost isolating cut C_i for t_i .
- Return the union $\mathcal{C}$ of the $k - 1$ cheapest such isolating cuts.

In other words:

Ignore the most expensive one of the isolating cuts $C_1, \dots, C_k$.

Algorithm MULTIWAYCUT

For $i = 1, \dots, k$:

- Compute a minimum-cost isolating cut C_i for t_i .
- Return the union $\mathcal{C}$ of the $k - 1$ cheapest such isolating cuts.

In other words:

Ignore the most expensive one of the isolating cuts $C_1, \dots, C_k$.

$$\Rightarrow c(\mathcal{C}) \quad ? \quad \sum_{i=1}^k c(C_i)$$

Algorithm MULTIWAYCUT

For $i = 1, \dots, k$:

- Compute a minimum-cost isolating cut C_i for t_i .
- Return the union $\mathcal{C}$ of the $k - 1$ cheapest such isolating cuts.

In other words:

Ignore the most expensive one of the isolating cuts $C_1, \dots, C_k$.

$$\Rightarrow c(\mathcal{C}) \leq \sum_{i=1}^k c(C_i)$$

Algorithm MULTIWAYCUT

For $i = 1, \dots, k$:

- Compute a minimum-cost isolating cut C_i for t_i .
- Return the union $\mathcal{C}$ of the $k - 1$ cheapest such isolating cuts.

In other words:

Ignore the most expensive one of the isolating cuts $C_1, \dots, C_k$.

$$\Rightarrow c(\mathcal{C}) \leq \left(1 - \frac{1}{k}\right) \sum_{i=1}^k c(C_i) \text{ because:}$$

Algorithm MULTIWAYCUT

For $i = 1, \dots, k$:

- Compute a minimum-cost isolating cut C_i for t_i .
- Return the union $\mathcal{C}$ of the $k - 1$ cheapest such isolating cuts.

In other words:

Ignore the most expensive one of the isolating cuts $C_1, \dots, C_k$.

$$\Rightarrow c(\mathcal{C}) \leq \left(1 - \frac{1}{k}\right) \sum_{i=1}^k c(C_i) \text{ because:}$$

for the most expensive cut of $C_1, \dots, C_k$, say C_1 , we have

$$c(C_1) \geq$$

Algorithm MULTIWAYCUT

For $i = 1, \dots, k$:

- Compute a minimum-cost isolating cut C_i for t_i .
- Return the union $\mathcal{C}$ of the $k - 1$ cheapest such isolating cuts.

In other words:

Ignore the most expensive one of the isolating cuts $C_1, \dots, C_k$.

$$\Rightarrow c(\mathcal{C}) \leq \left(1 - \frac{1}{k}\right) \sum_{i=1}^k c(C_i) \text{ because:}$$

for the most expensive cut of $C_1, \dots, C_k$, say C_1 , we have

$$c(C_1) \geq \frac{1}{k} \sum_{i=1}^k c(C_i)$$

Algorithm MULTIWAYCUT

For $i = 1, \dots, k$:

- Compute a minimum-cost isolating cut C_i for t_i .
- Return the union $\mathcal{C}$ of the $k - 1$ cheapest such isolating cuts.

In other words:

Ignore the most expensive one of the isolating cuts $C_1, \dots, C_k$.

$$\Rightarrow c(\mathcal{C}) \leq \left(1 - \frac{1}{k}\right) \sum_{i=1}^k c(C_i) \text{ because:}$$

for the most expensive cut of $C_1, \dots, C_k$, say C_1 , we have

$$c(C_1) \geq \frac{1}{k} \sum_{i=1}^k c(C_i) \text{ by the pigeon-hole principle.}$$

Approximation Factor

Theorem. This algorithm is a factor-() approximation algorithm for `MULTIWAYCUT`.

Approximation Factor

Theorem. This algorithm is a factor- $(2 - 2/k)$ approximation algorithm for MULTIWAYCUT.

Approximation Factor

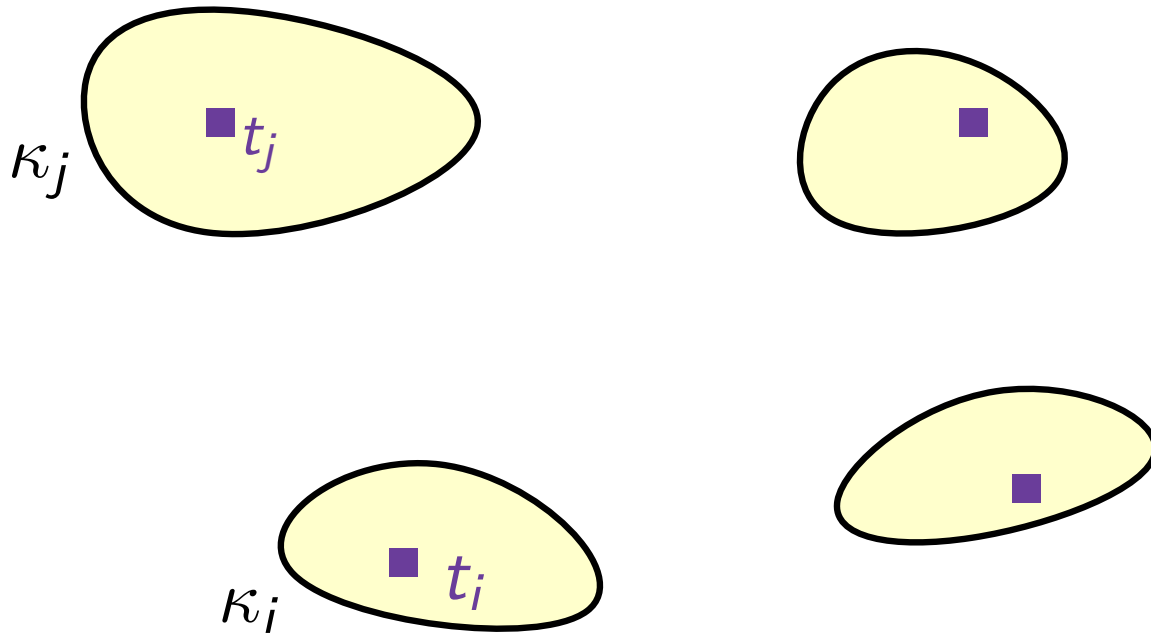
Theorem. This algorithm is a factor- $(2 - 2/k)$ approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$:

Approximation Factor

Theorem. This algorithm is a factor- $(2 - 2/k)$ approximation algorithm for MULTIWAYCUT.

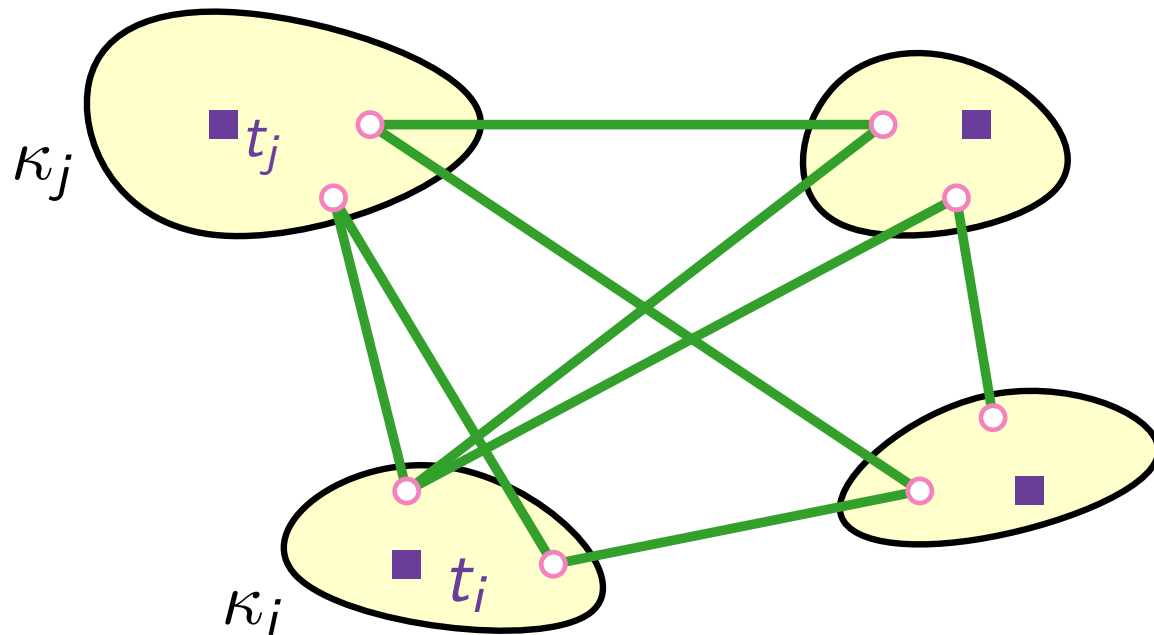
Proof. Consider an opt. multiway cut $\mathcal{A}$:



Approximation Factor

Theorem. This algorithm is a factor- $(2 - 2/k)$ approximation algorithm for MULTIWAYCUT.

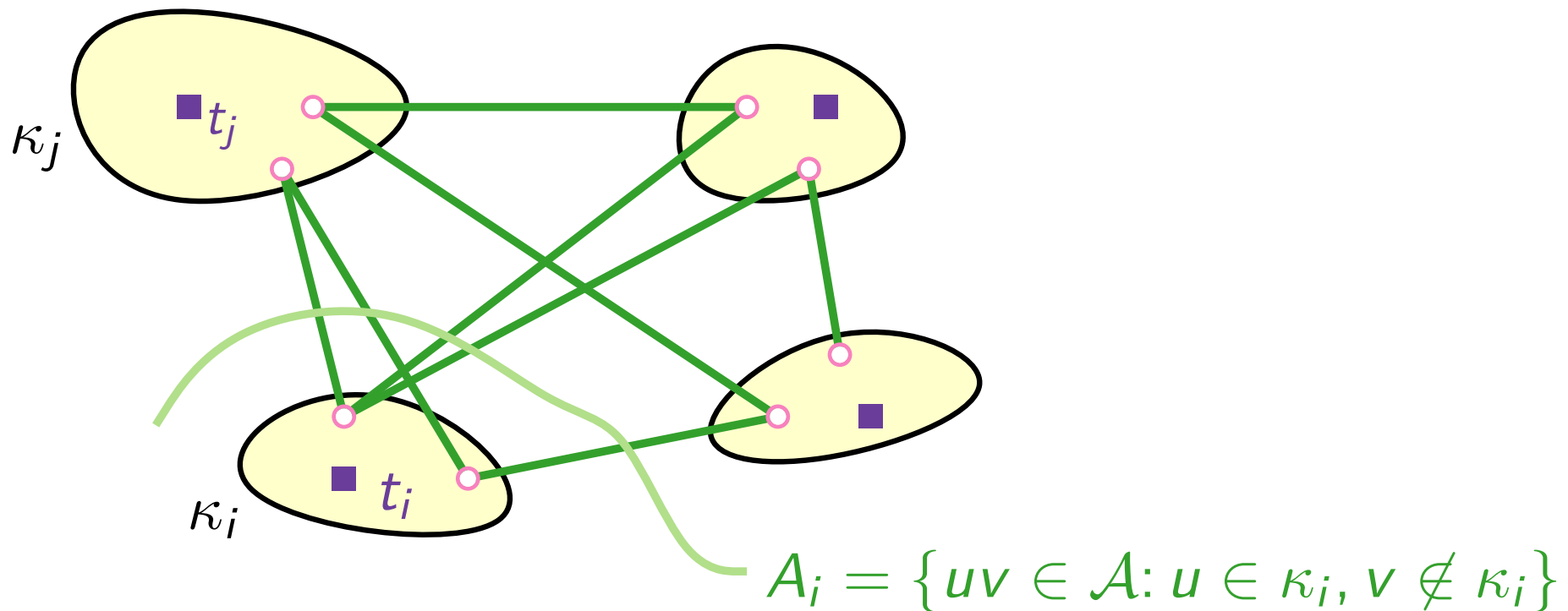
Proof. Consider an opt. multiway cut $\mathcal{A}$:



Approximation Factor

Theorem. This algorithm is a factor- $(2 - 2/k)$ approximation algorithm for MULTIWAYCUT.

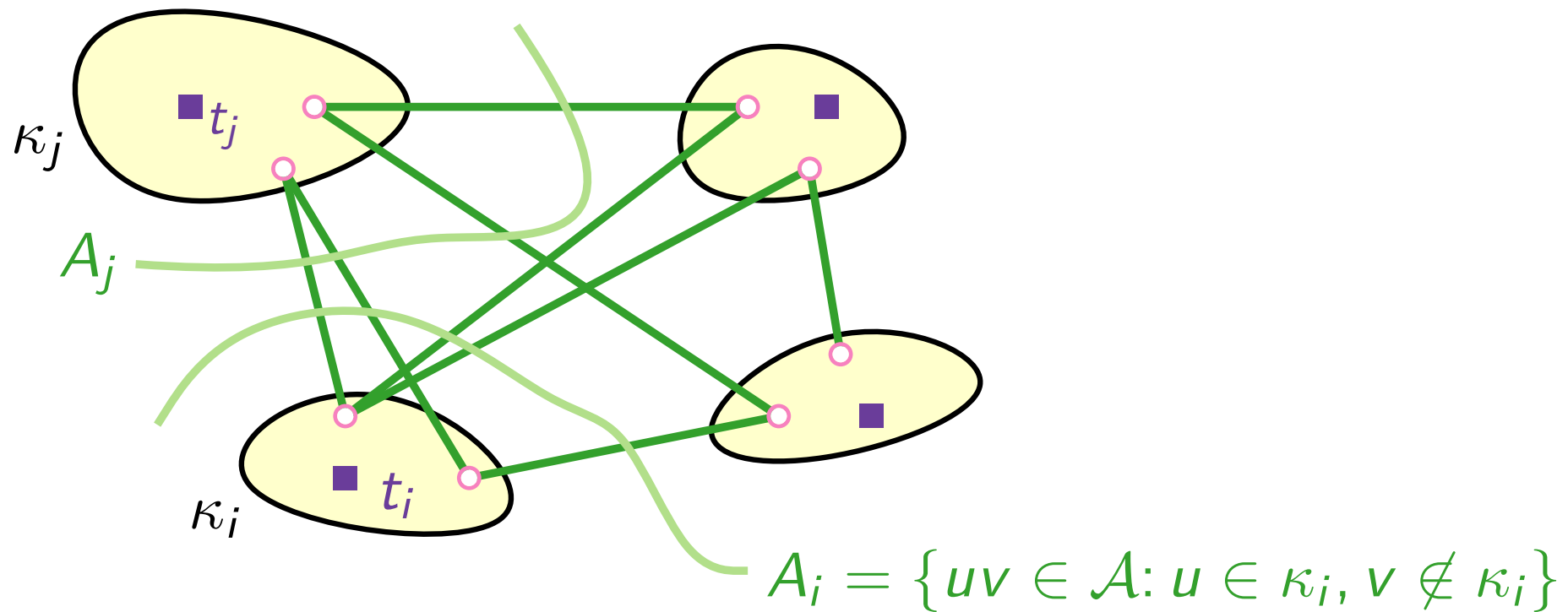
Proof. Consider an opt. multiway cut $\mathcal{A}$:



Approximation Factor

Theorem. This algorithm is a factor- $(2 - 2/k)$ approximation algorithm for MULTIWAYCUT.

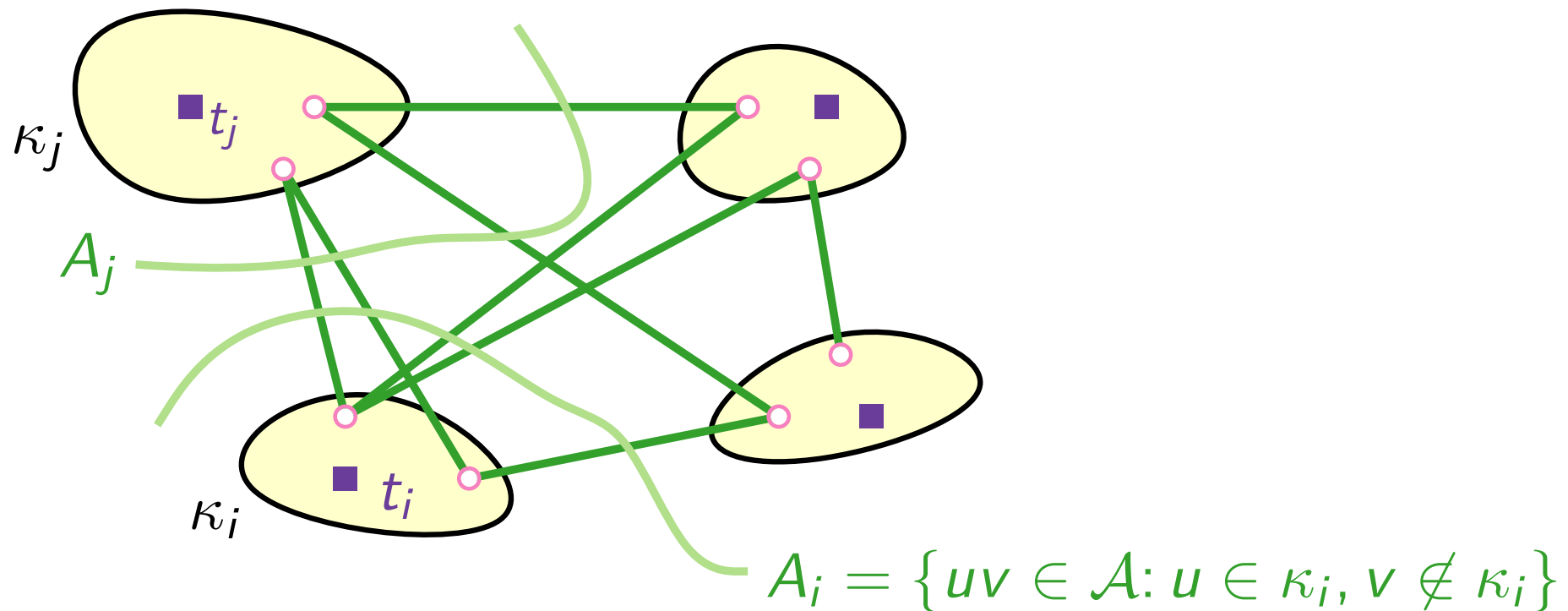
Proof. Consider an opt. multiway cut $\mathcal{A}$:



Approximation Factor

Theorem. This algorithm is a factor- $(2 - 2/k)$ approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$:

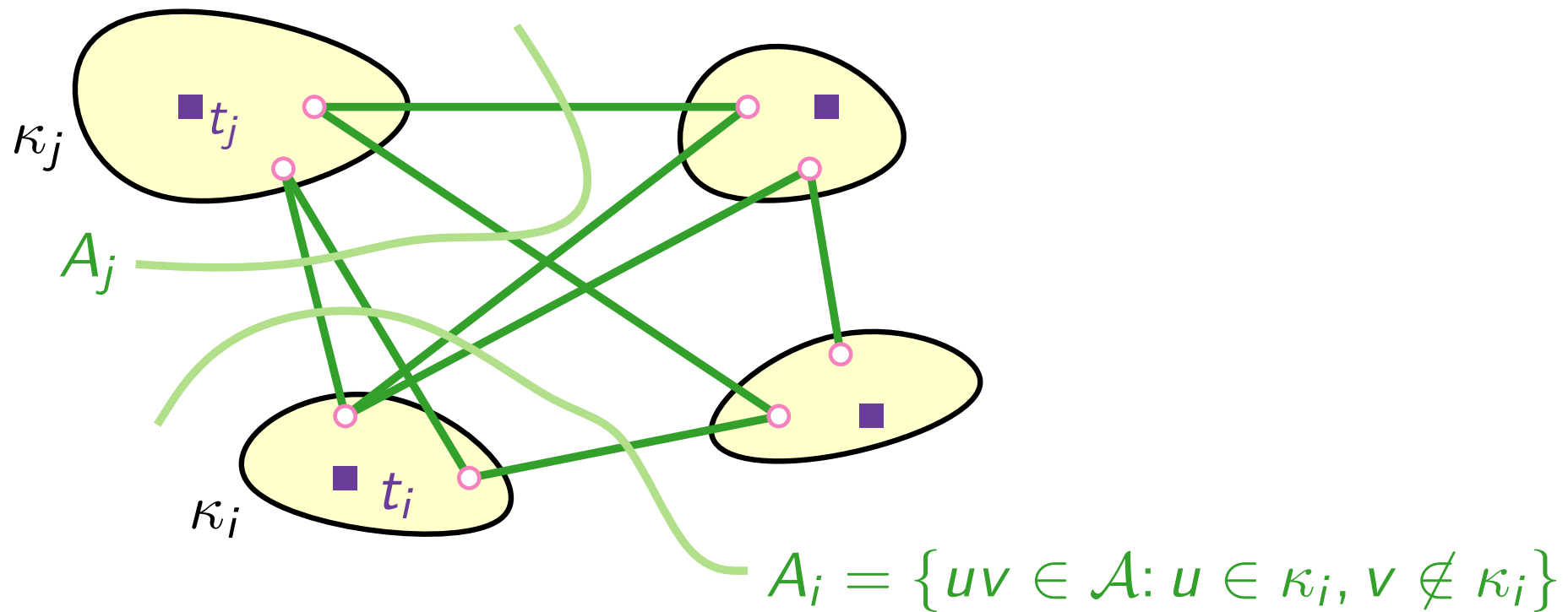


Observation. $\mathcal{A} =$

Approximation Factor

Theorem. This algorithm is a factor- $(2 - 2/k)$ approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$:

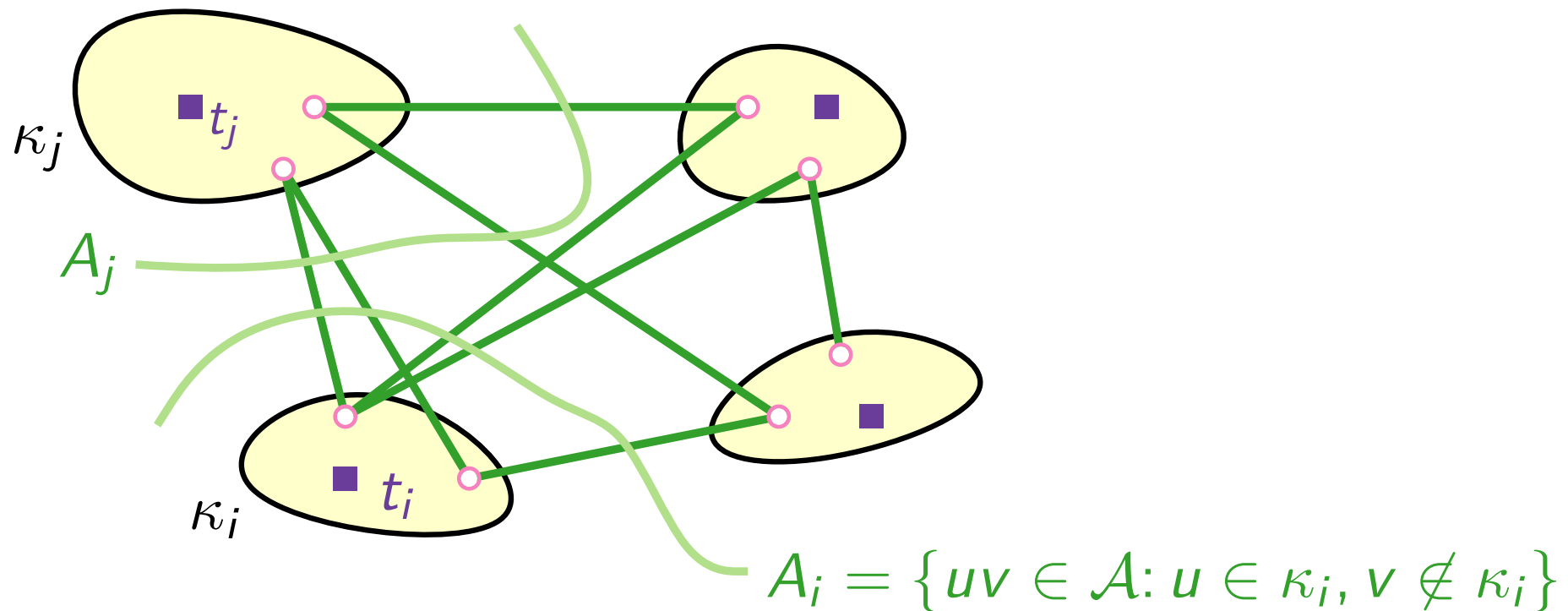


Observation. $\mathcal{A} = \bigcup_{i=1}^k A_i$

Approximation Factor

Theorem. This algorithm is a factor- $(2 - 2/k)$ approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$:

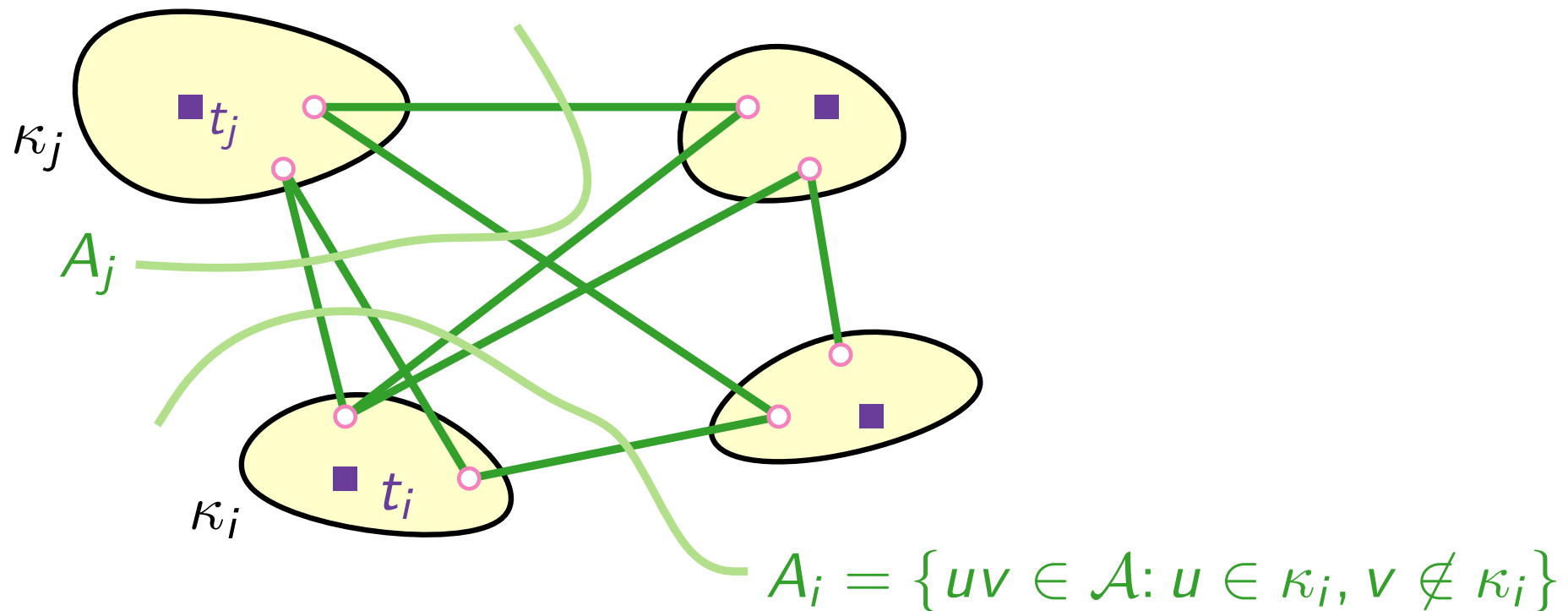


Observation. $\mathcal{A} = \bigcup_{i=1}^k A_i$ and $\sum_{i=1}^k c(A_i) \leq$

Approximation Factor

Theorem. This algorithm is a factor- $(2 - 2/k)$ approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$:

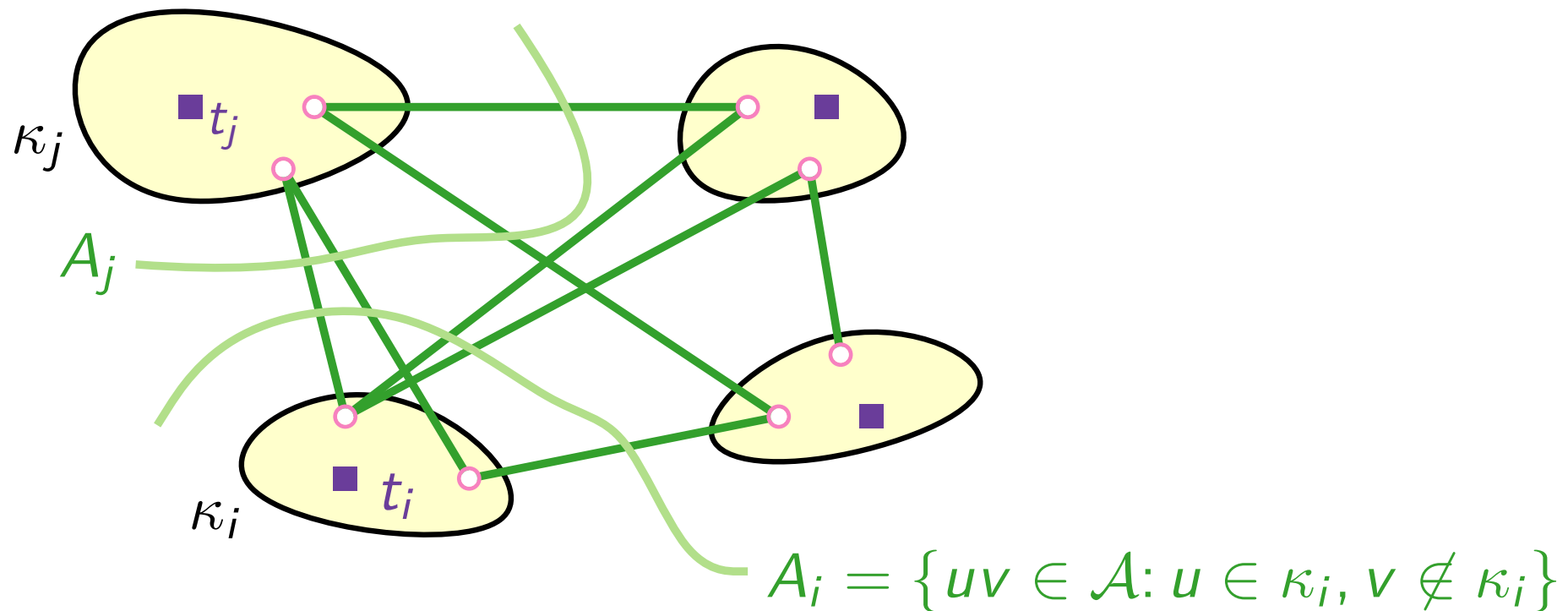


Observation. $\mathcal{A} = \bigcup_{i=1}^k A_i$ and $\sum_{i=1}^k c(A_i) \leq 2 \cdot c(\mathcal{A}) =$

Approximation Factor

Theorem. This algorithm is a factor- $(2 - 2/k)$ approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$:

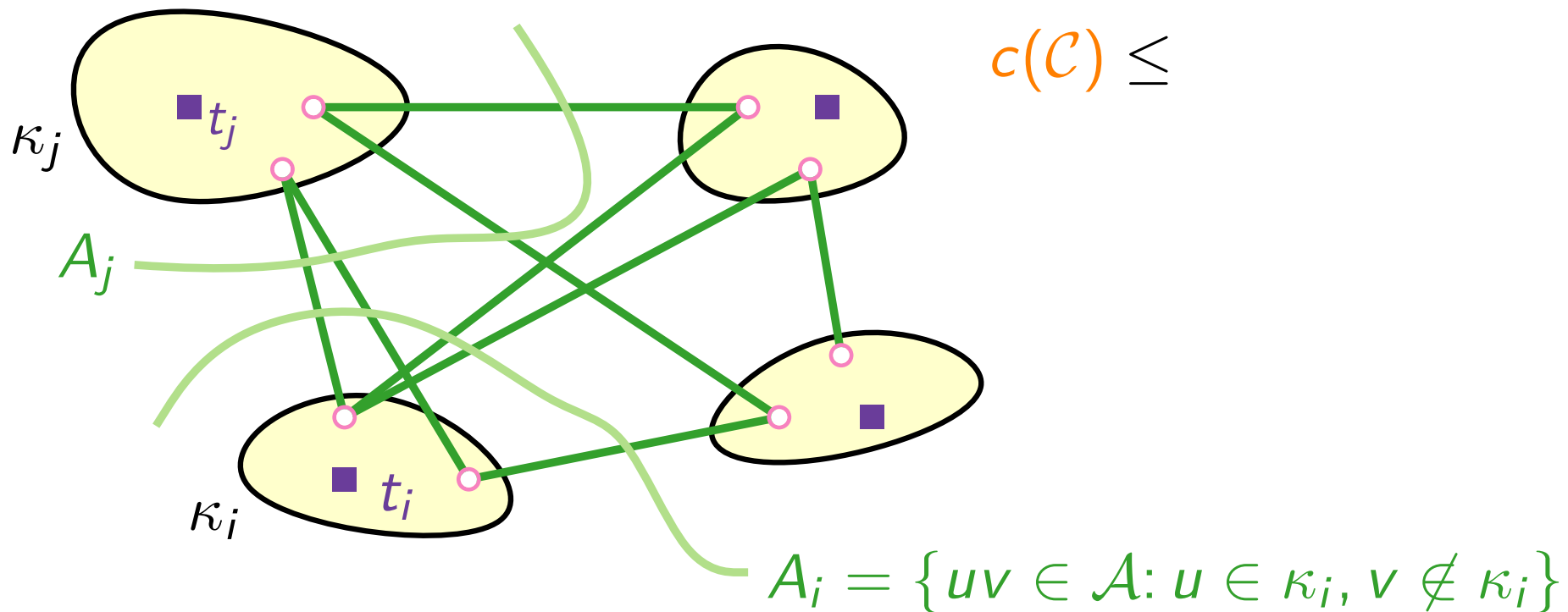


Observation. $\mathcal{A} = \bigcup_{i=1}^k A_i$ and $\sum_{i=1}^k c(A_i) \leq 2 \cdot c(\mathcal{A}) = 2 \cdot \text{OPT}.$

Approximation Factor

Theorem. This algorithm is a factor- $(2 - 2/k)$ approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$: Consider the alg.'s solution $\mathcal{C}$:



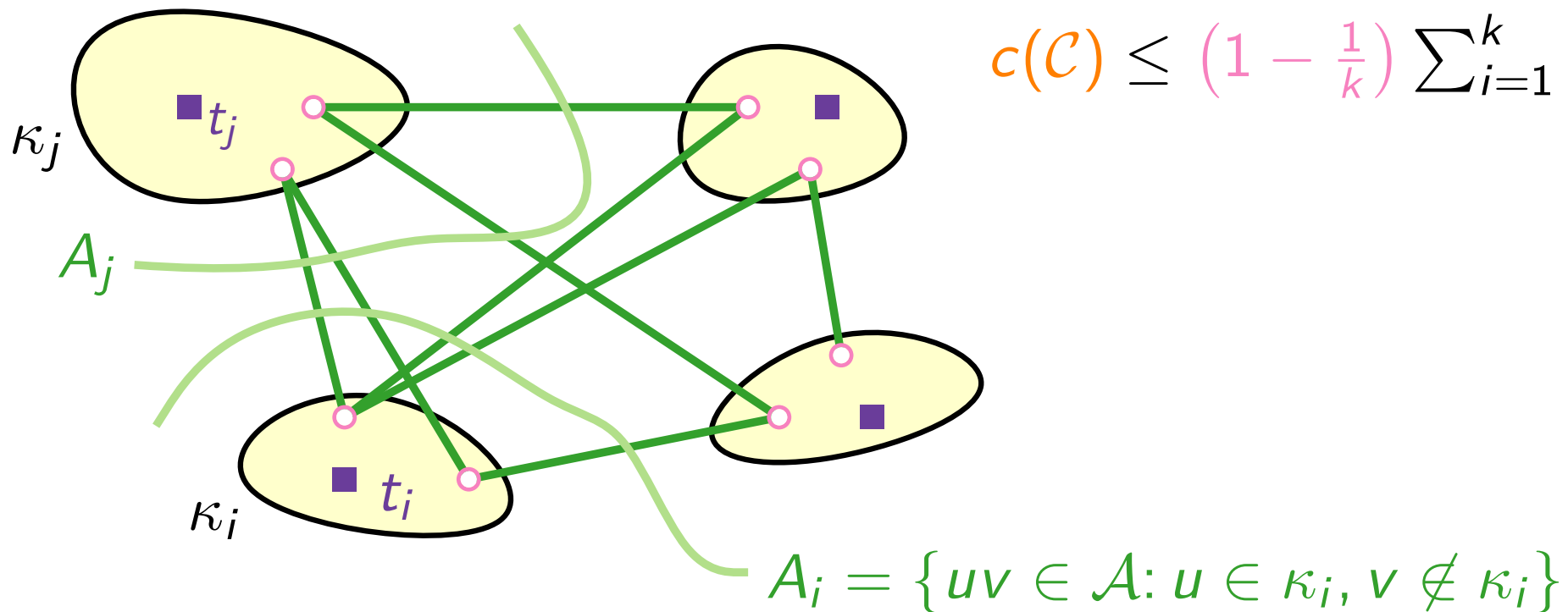
Observation. $\mathcal{A} = \bigcup_{i=1}^k A_i$ and $\sum_{i=1}^k c(A_i) \leq 2 \cdot c(\mathcal{A}) = 2 \cdot \text{OPT}.$

Approximation Factor

Theorem. This algorithm is a factor- $(2 - 2/k)$ approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$: Consider the alg.'s solution $\mathcal{C}$:

$$c(\mathcal{C}) \leq \left(1 - \frac{1}{k}\right) \sum_{i=1}^k c(\mathcal{C}_i)$$

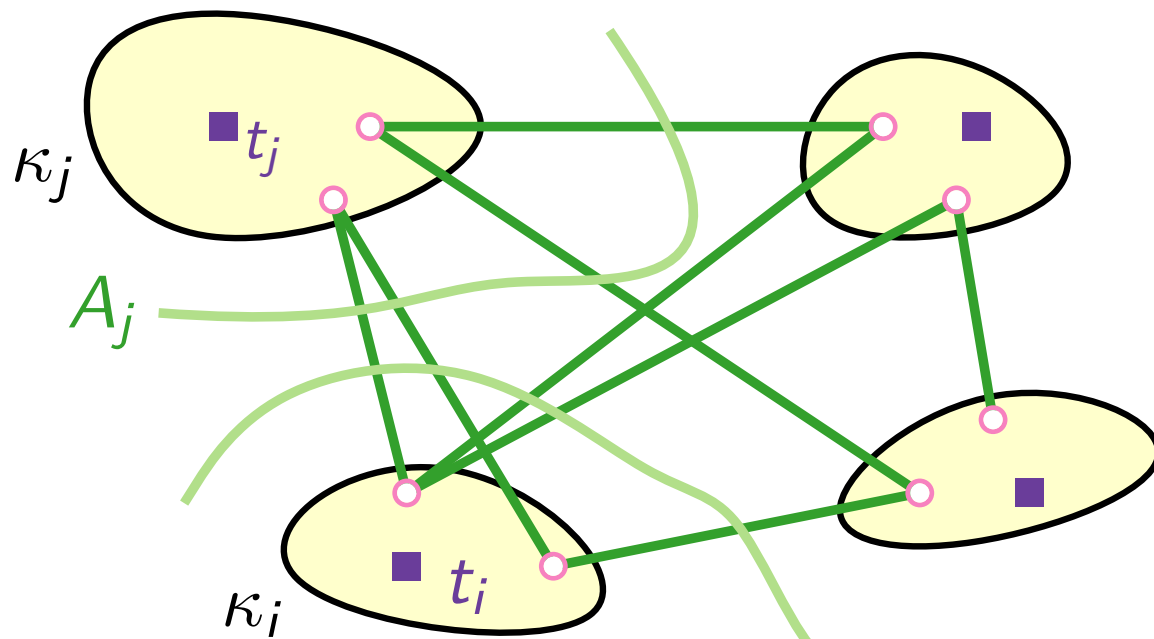


Observation. $\mathcal{A} = \bigcup_{i=1}^k A_i$ and $\sum_{i=1}^k c(A_i) \leq 2 \cdot c(\mathcal{A}) = 2 \cdot \text{OPT}.$

Approximation Factor

Theorem. This algorithm is a factor- $(2 - 2/k)$ approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$: Consider the alg.'s solution $\mathcal{C}$:



$$\begin{aligned} c(\mathcal{C}) &\leq \left(1 - \frac{1}{k}\right) \sum_{i=1}^k c(C_i) \\ &\leq \left(1 - \frac{1}{k}\right) \sum_{i=1}^k c(A_i) \end{aligned}$$

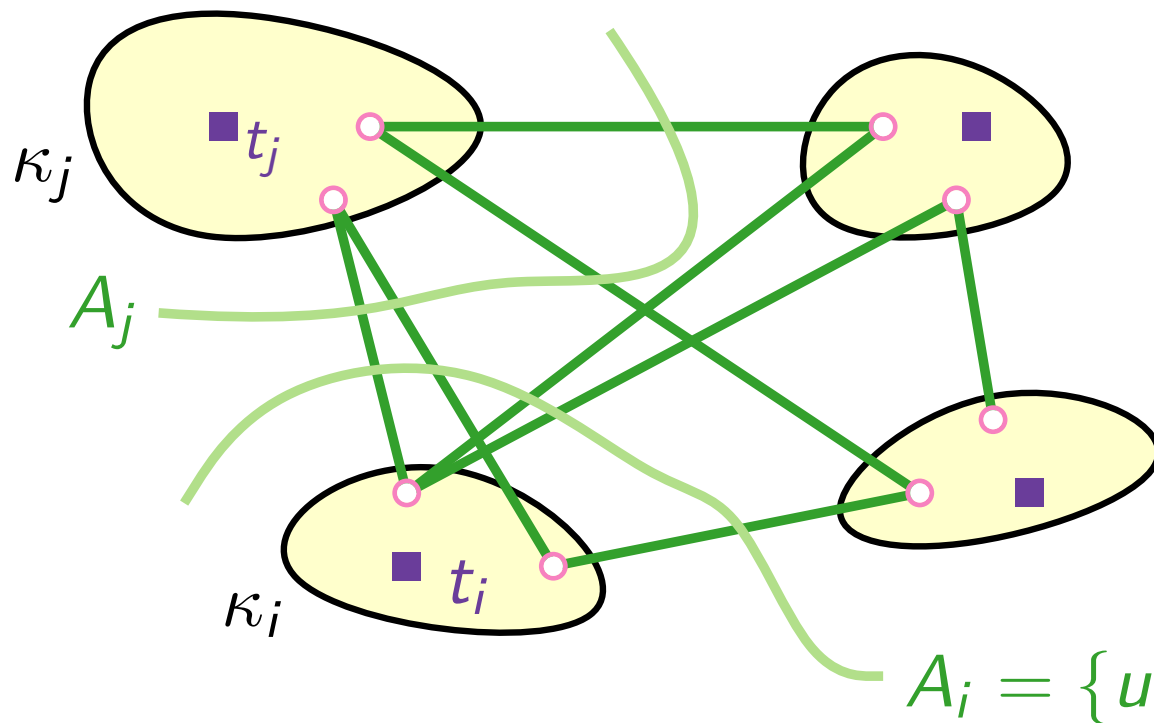
$$A_i = \{uv \in \mathcal{A} : u \in \kappa_i, v \notin \kappa_i\}$$

Observation. $\mathcal{A} = \bigcup_{i=1}^k A_i$ and $\sum_{i=1}^k c(A_i) \leq 2 \cdot c(\mathcal{A}) = 2 \cdot \text{OPT}.$

Approximation Factor

Theorem. This algorithm is a factor- $(2 - 2/k)$ approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$: Consider the alg.'s solution $\mathcal{C}$:



$$\begin{aligned} c(\mathcal{C}) &\leq \left(1 - \frac{1}{k}\right) \sum_{i=1}^k c(\mathcal{C}_i) \\ &\leq \left(1 - \frac{1}{k}\right) \sum_{i=1}^k c(\mathcal{A}_i) \\ &\leq \left(1 - \frac{1}{k}\right) \cdot 2 \cdot c(\mathcal{A}) \end{aligned}$$

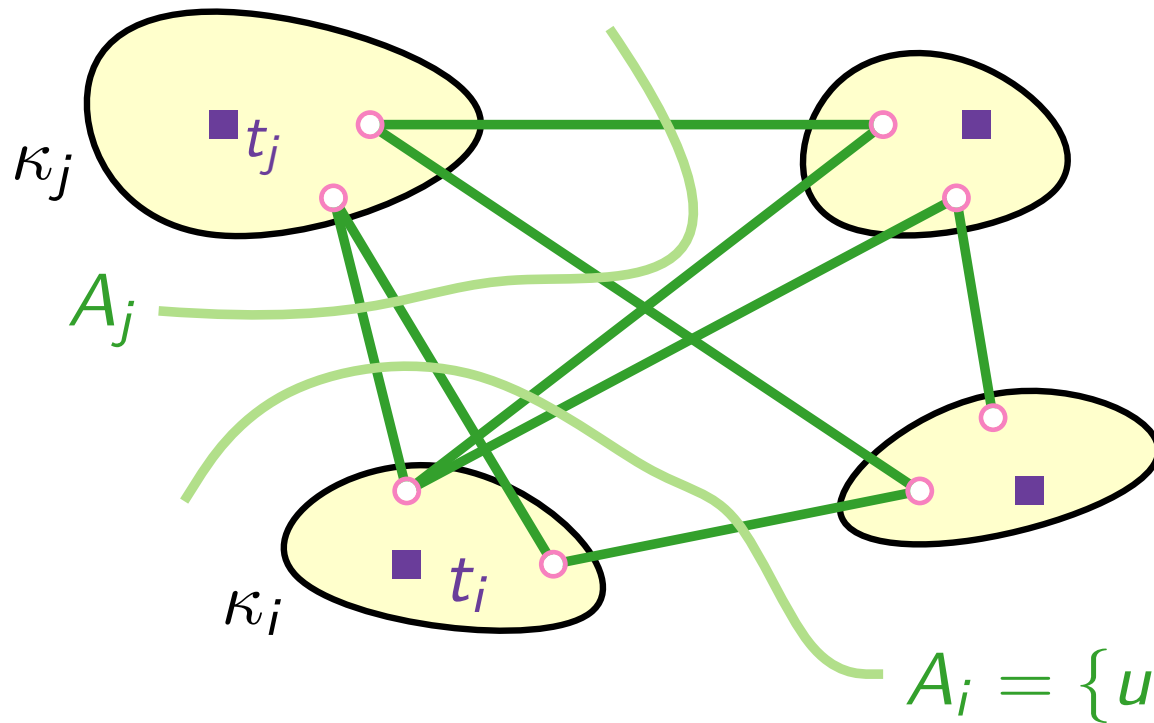
$$\mathcal{A}_i = \{uv \in \mathcal{A} : u \in \kappa_i, v \notin \kappa_i\}$$

Observation. $\mathcal{A} = \bigcup_{i=1}^k \mathcal{A}_i$ and $\sum_{i=1}^k c(\mathcal{A}_i) \leq 2 \cdot c(\mathcal{A}) = 2 \cdot \text{OPT}.$

Approximation Factor

Theorem. This algorithm is a factor- $(2 - 2/k)$ approximation algorithm for MULTIWAYCUT.

Proof. Consider an opt. multiway cut $\mathcal{A}$: Consider the alg.'s solution $\mathcal{C}$:



$$\begin{aligned}
 c(\mathcal{C}) &\leq \left(1 - \frac{1}{k}\right) \sum_{i=1}^k c(C_i) \\
 &\leq \left(1 - \frac{1}{k}\right) \sum_{i=1}^k c(A_i) \\
 &\leq \left(1 - \frac{1}{k}\right) \cdot 2 \cdot c(\mathcal{A}) \\
 &\leq \left(2 - \frac{2}{k}\right) \cdot \text{OPT}
 \end{aligned}$$

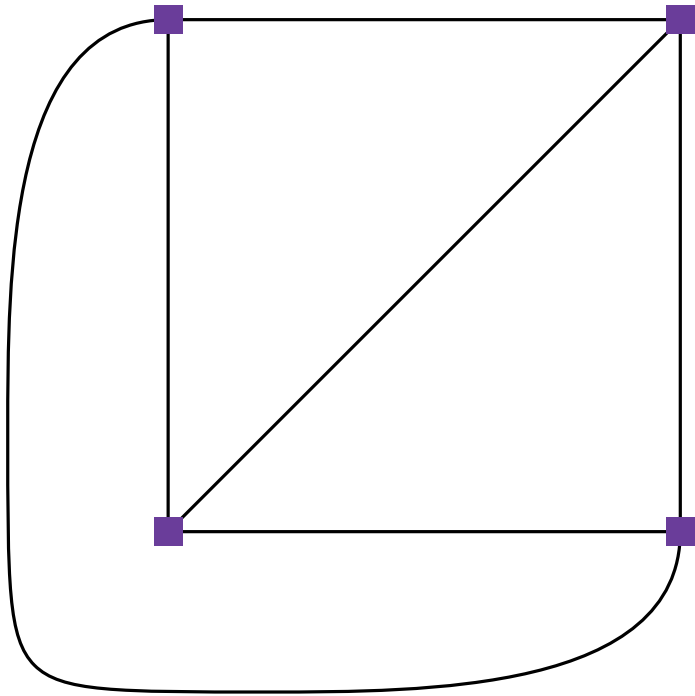
$$A_i = \{uv \in \mathcal{A} : u \in \kappa_i, v \notin \kappa_i\}$$

Observation. $\mathcal{A} = \bigcup_{i=1}^k A_i$ and $\sum_{i=1}^k c(A_i) \leq 2 \cdot c(\mathcal{A}) = 2 \cdot \text{OPT}$.

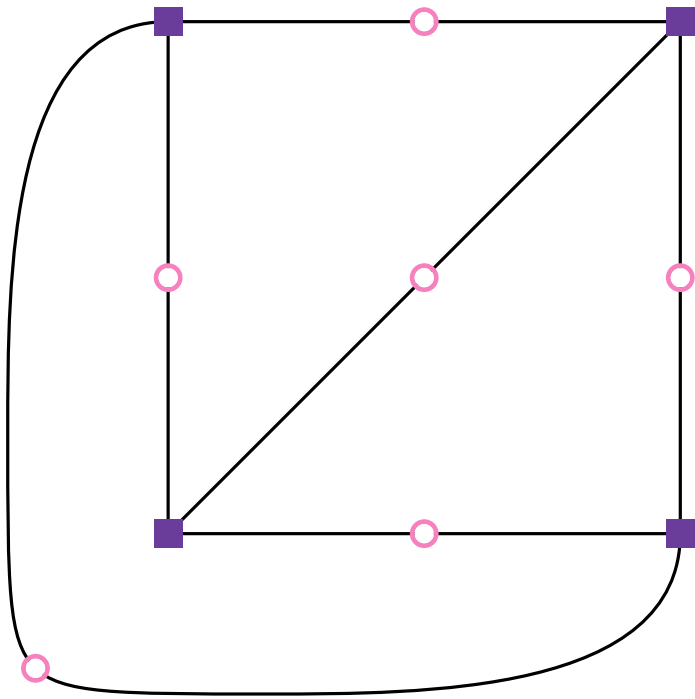
Analysis Tight?

K_k

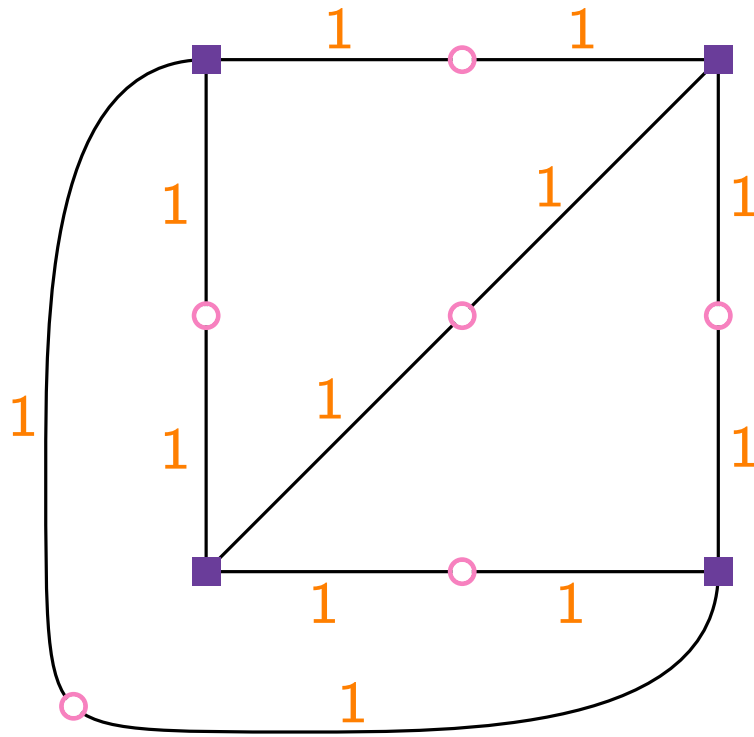
Analysis Tight?

 K_k 

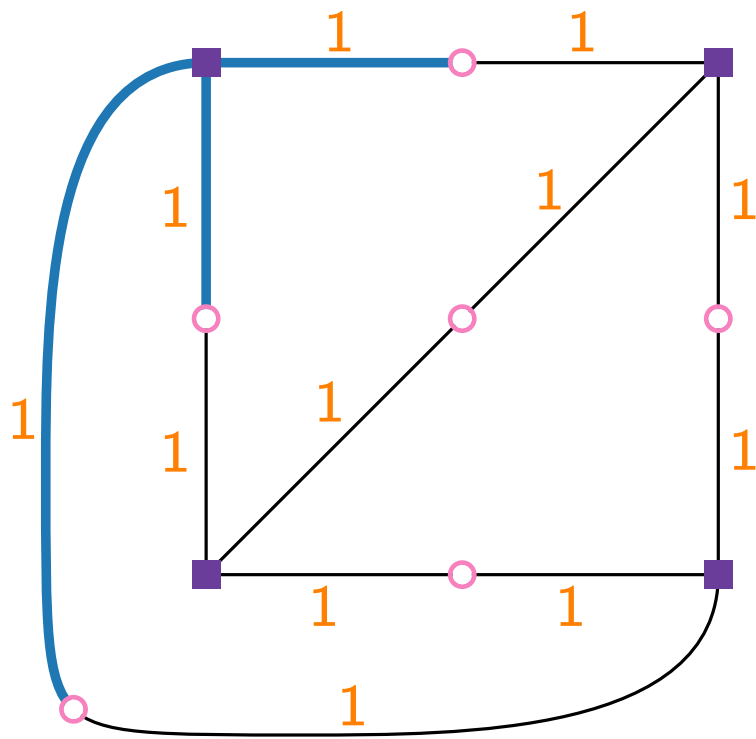
Analysis Tight?

 K_k 

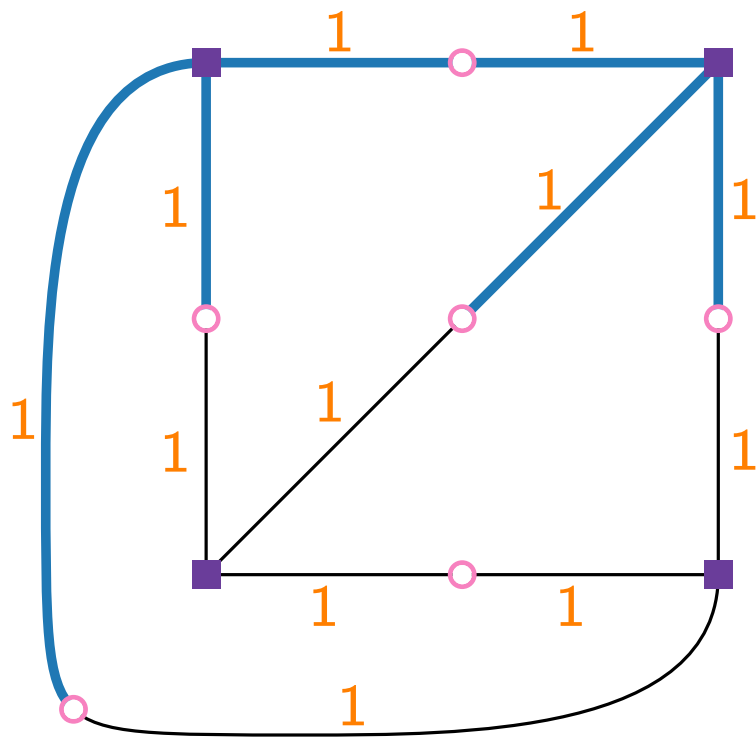
Analysis Tight?

 K_k 

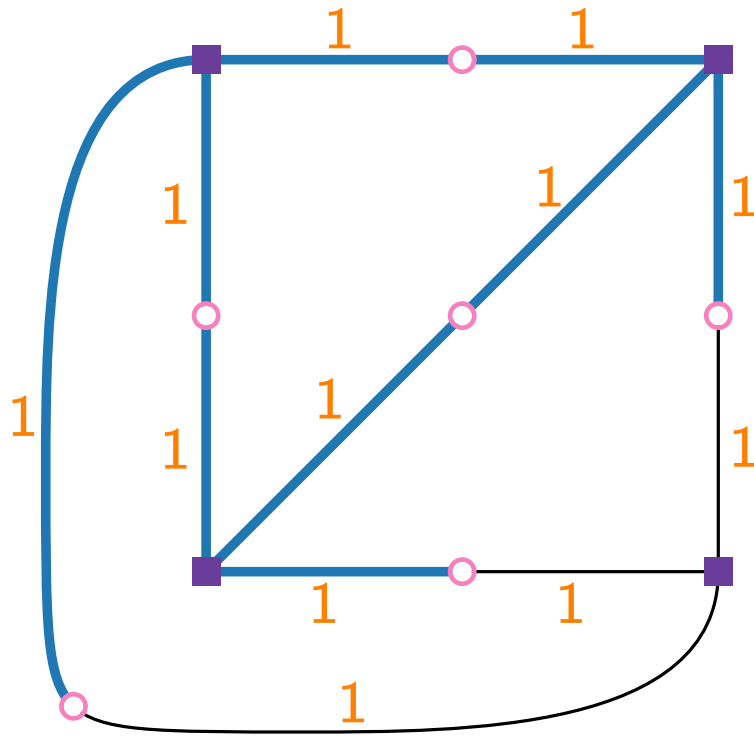
Analysis Tight?

 K_k 

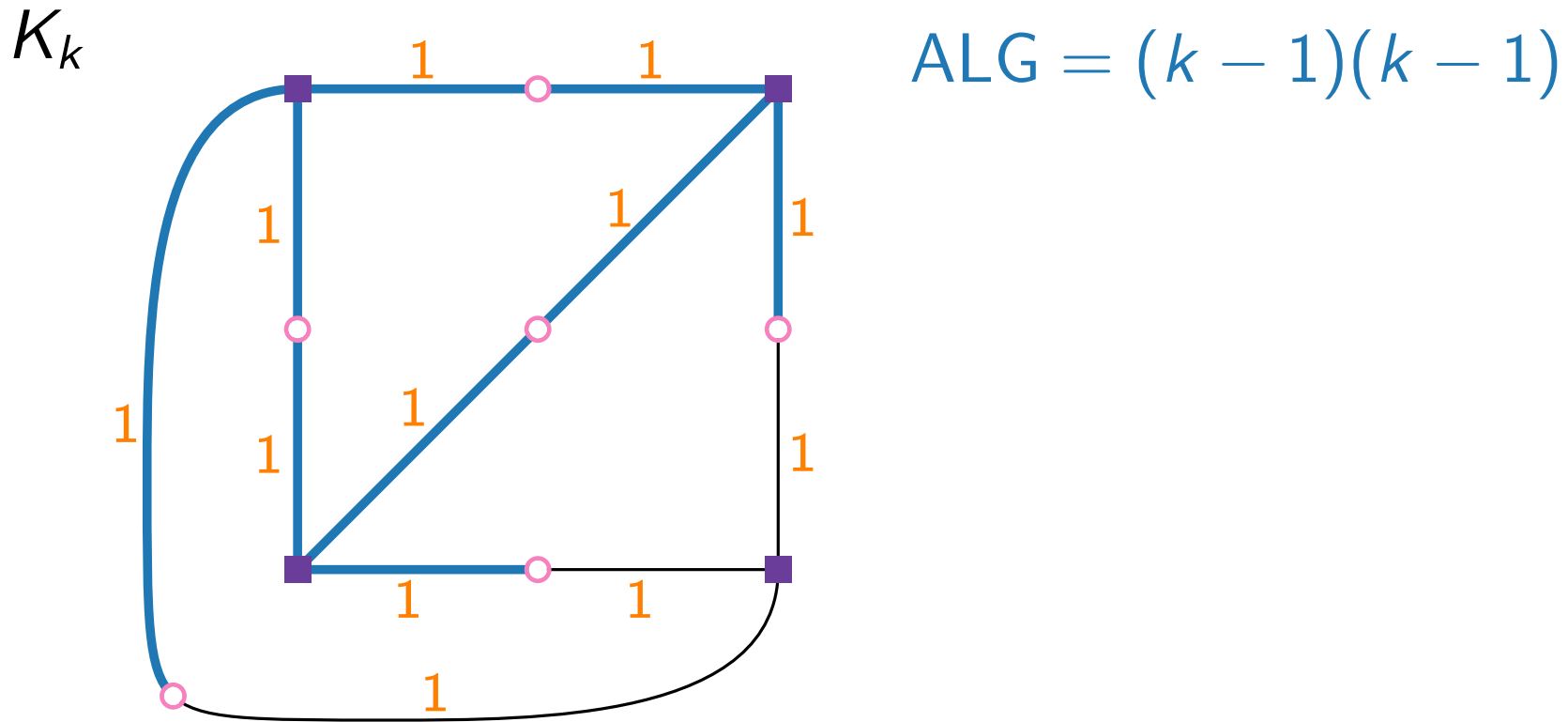
Analysis Tight?

 K_k 

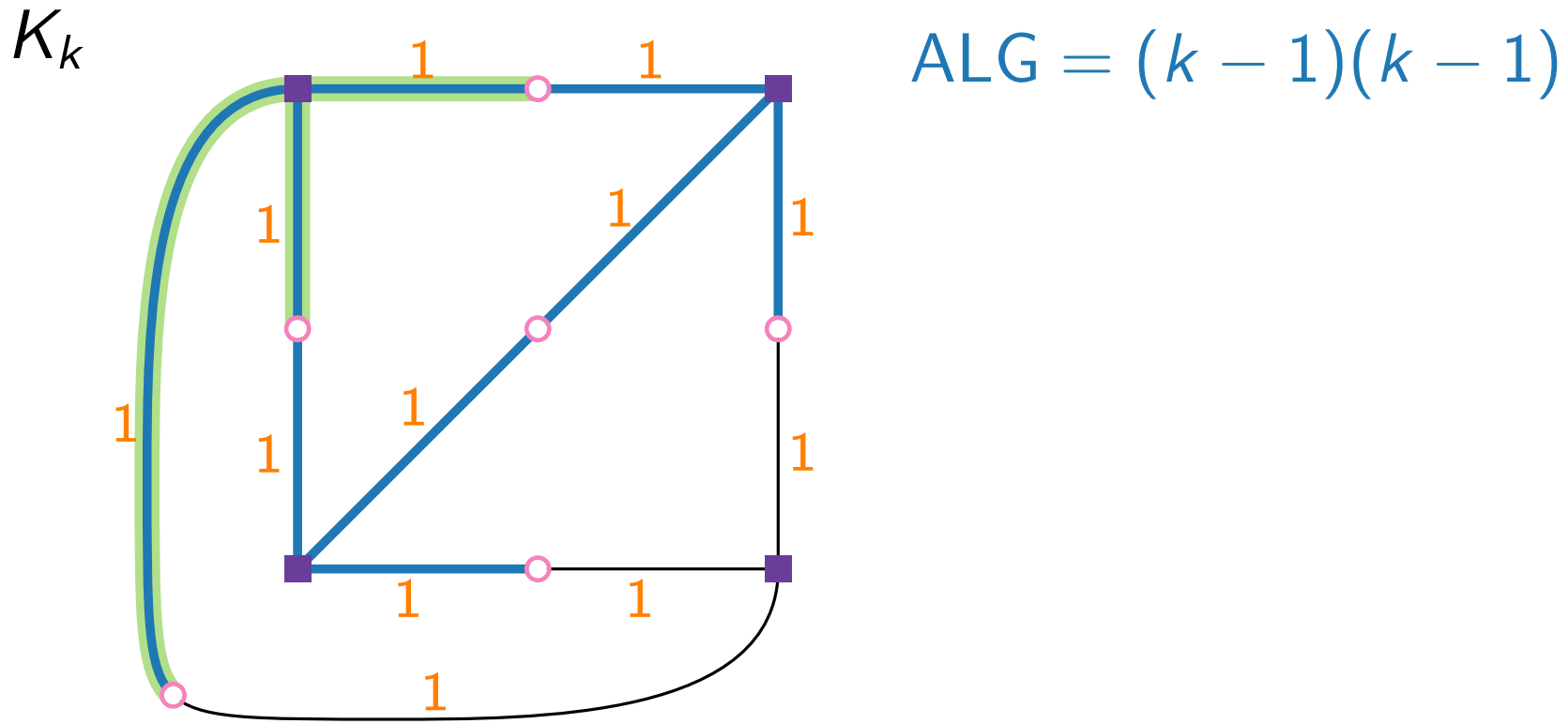
Analysis Tight?

 K_k 

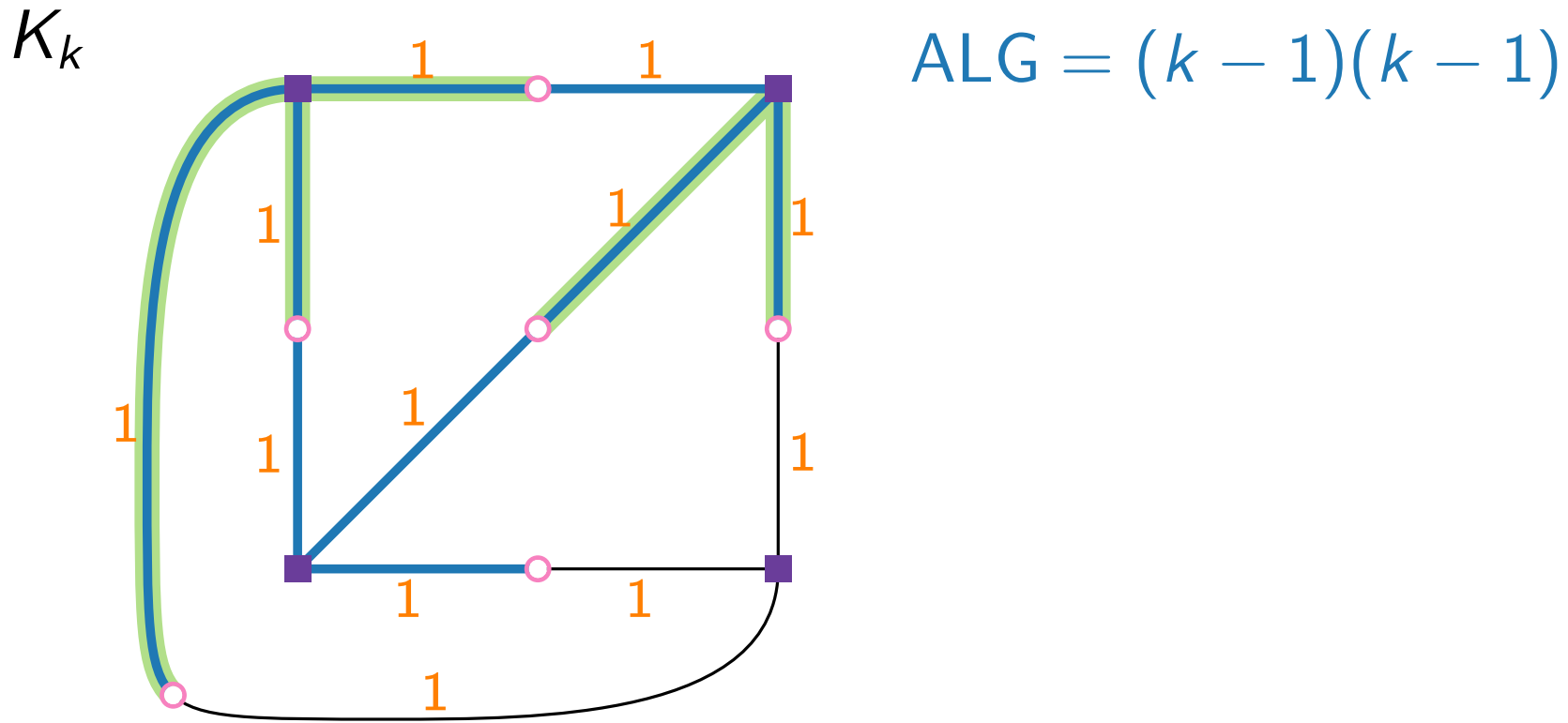
Analysis Tight?



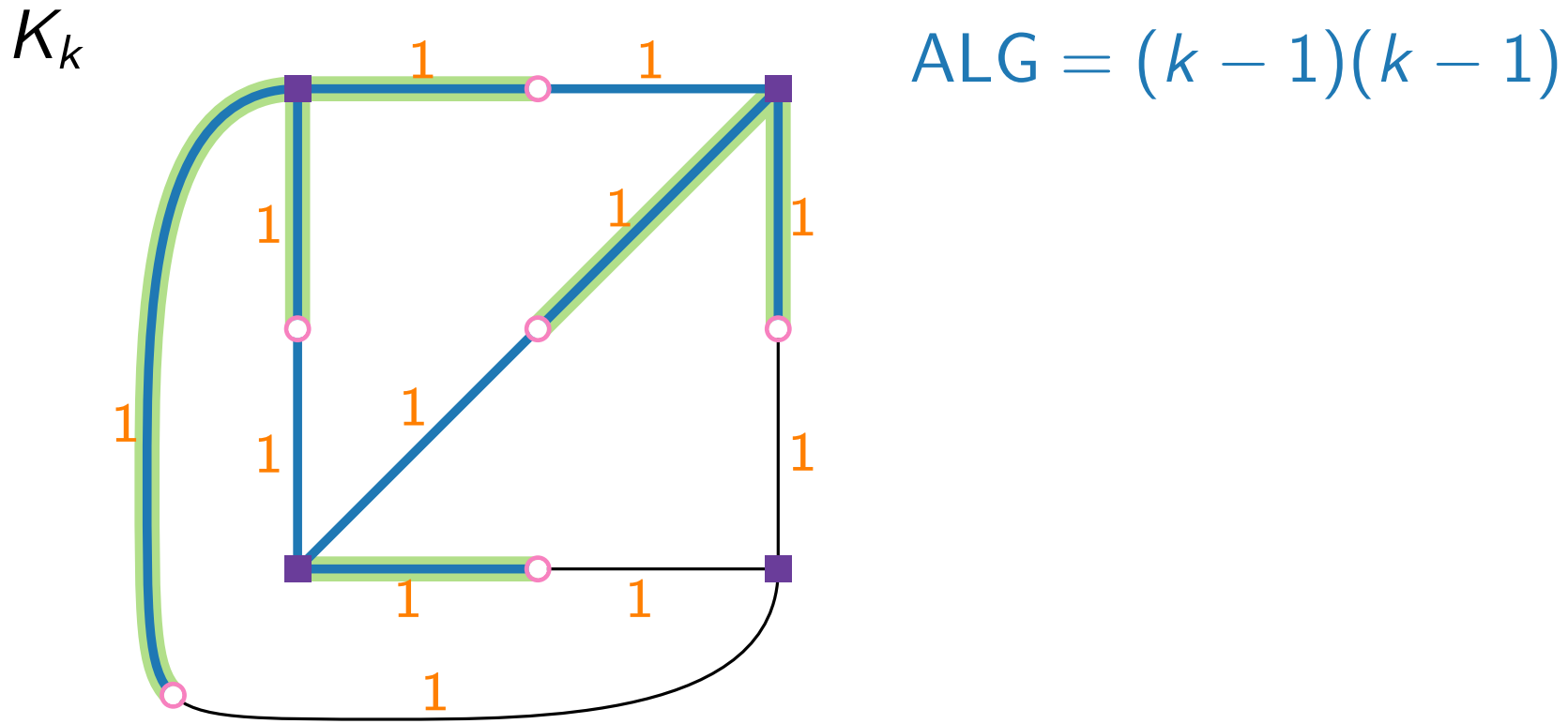
Analysis Tight?



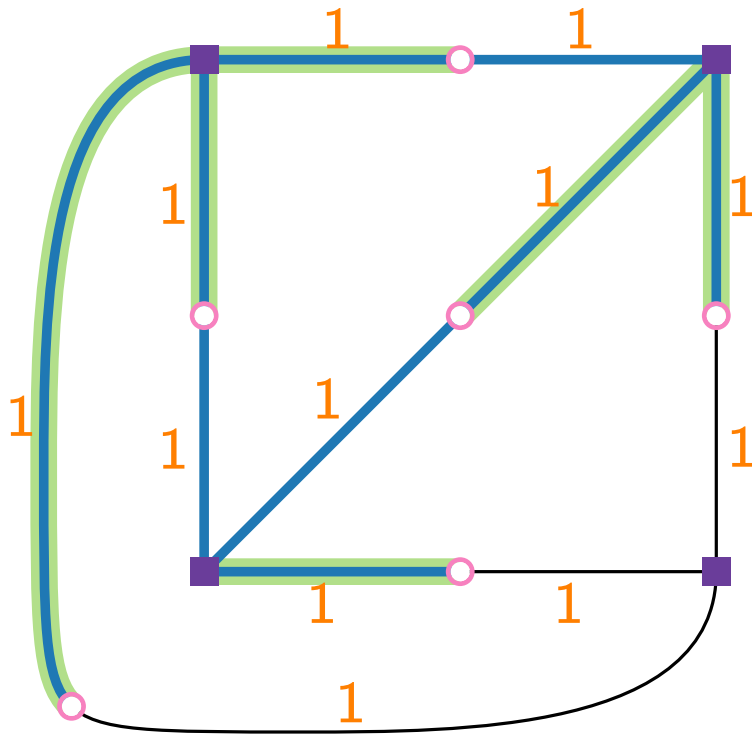
Analysis Tight?



Analysis Tight?



Analysis Tight?

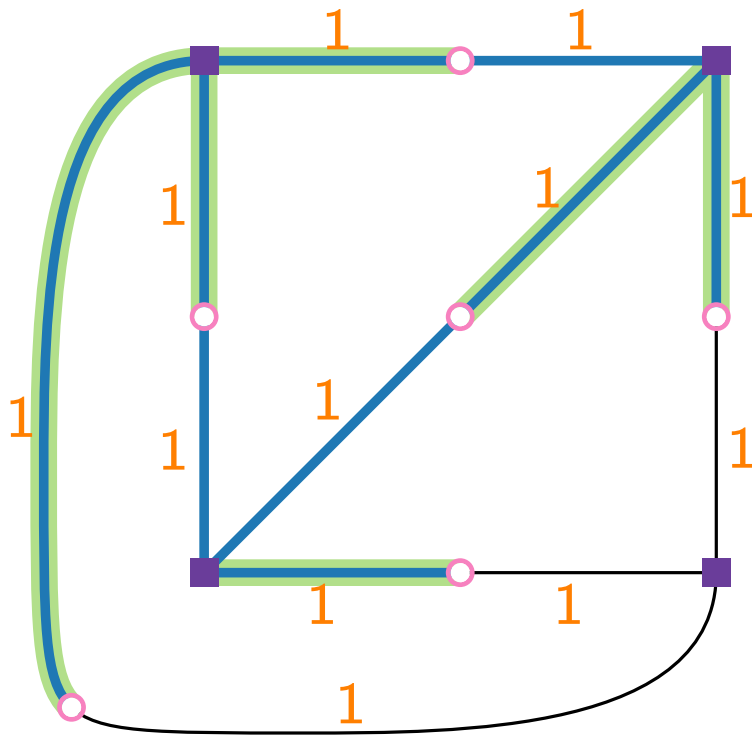
 K_k


$$\text{ALG} = (k - 1)(k - 1)$$

$$\text{OPT} = \sum_{i=1}^{k-1} i =$$

Analysis Tight?

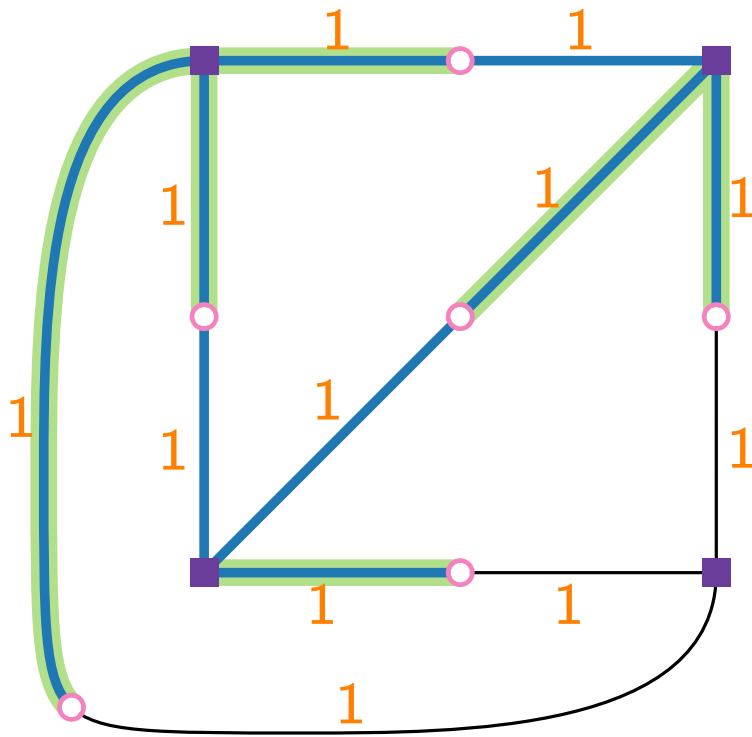
K_k



$$\text{ALG} = (k - 1)(k - 1)$$

$$\text{OPT} = \sum_{i=1}^{k-1} i = \frac{k \cdot (k-1)}{2}$$

Analysis Tight?

 K_k


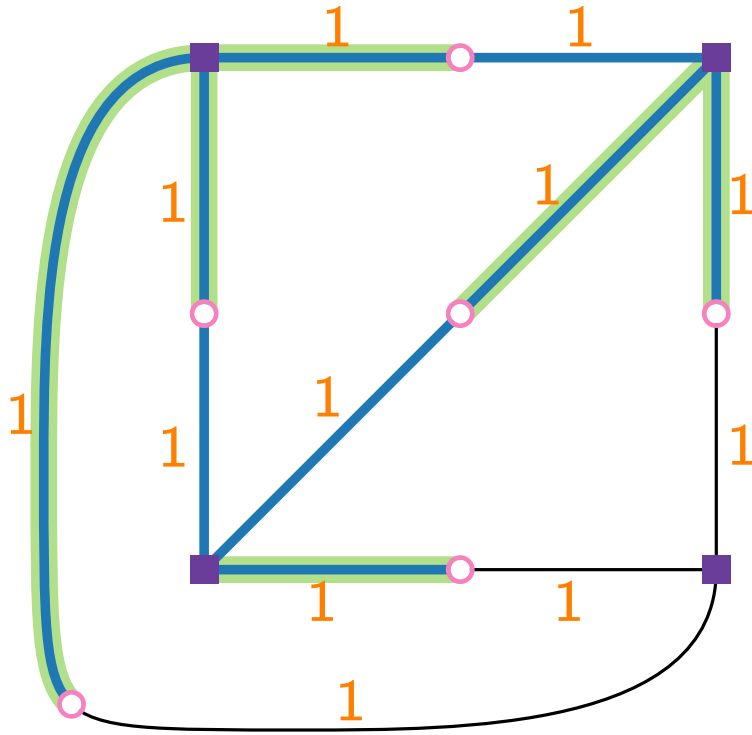
$$\text{ALG} = (k - 1)(k - 1)$$

$$\text{OPT} = \sum_{i=1}^{k-1} i = \frac{k \cdot (k-1)}{2}$$

$$\text{ALG} / \text{OPT} =$$

Analysis Tight?

K_k



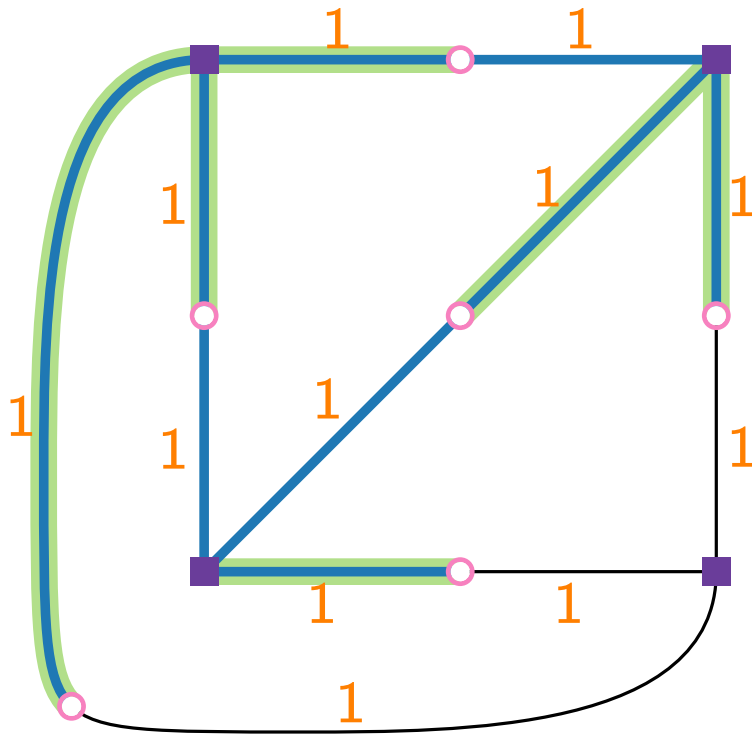
$$\text{ALG} = (k - 1)(k - 1)$$

$$\text{OPT} = \sum_{i=1}^{k-1} i = \frac{k \cdot (k-1)}{2}$$

$$\text{ALG} / \text{OPT} = \frac{2(k-1)}{k} =$$

Analysis Tight?

K_k



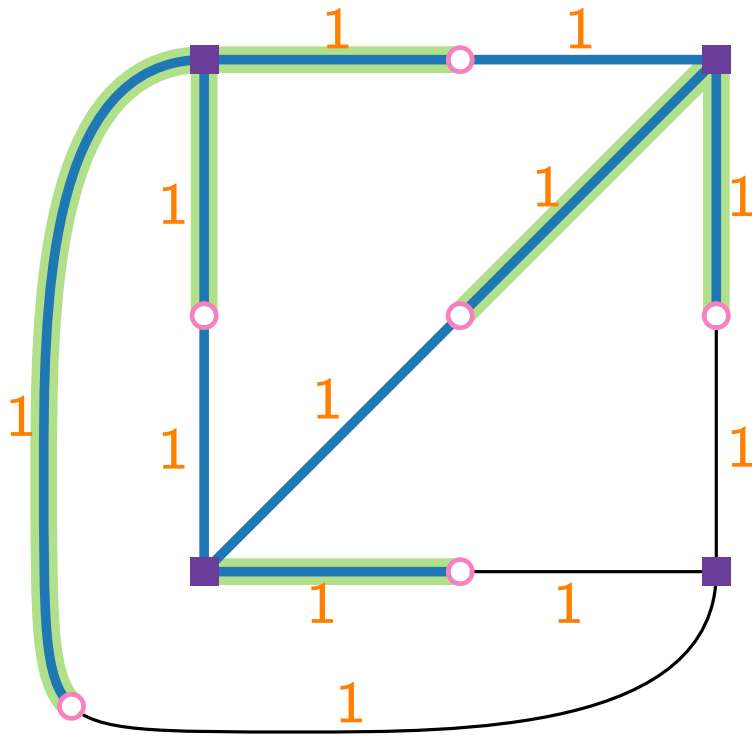
$$\text{ALG} = (k - 1)(k - 1)$$

$$\text{OPT} = \sum_{i=1}^{k-1} i = \frac{k \cdot (k-1)}{2}$$

$$\text{ALG} / \text{OPT} = \frac{2(k-1)}{k} = 2 - \frac{2}{k}$$

Analysis Tight?

K_k



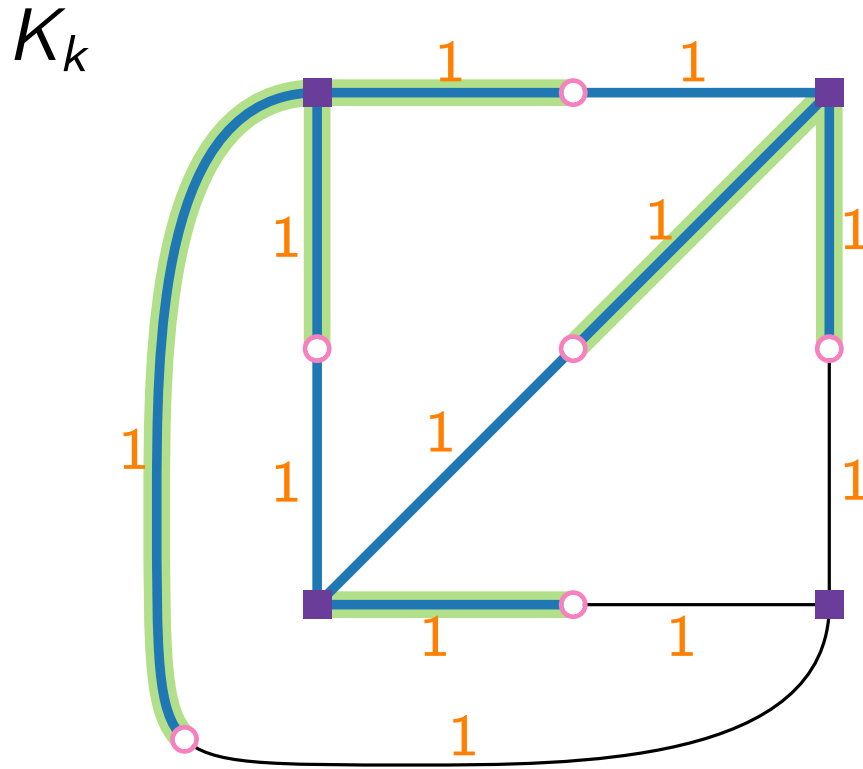
$$\text{ALG} = (k - 1)(k - 1)$$

$$\text{OPT} = \sum_{i=1}^{k-1} i = \frac{k \cdot (k-1)}{2}$$

$$\text{ALG}/\text{OPT} = \frac{2(k-1)}{k} = 2 - \frac{2}{k}$$

Can we do better?

Analysis Tight?



$$\text{ALG} = (k - 1)(k - 1)$$

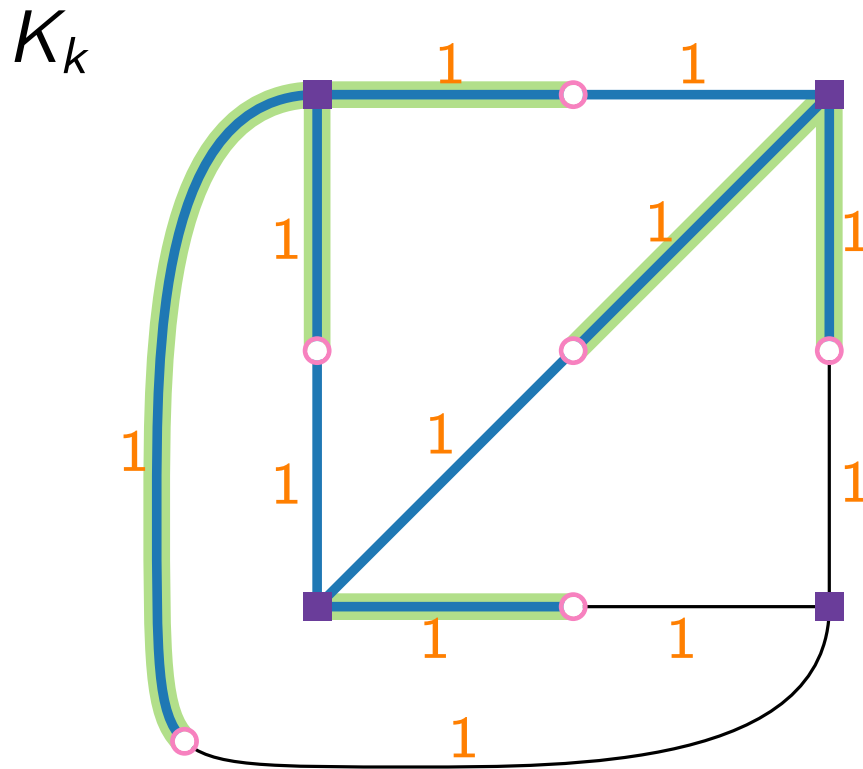
$$\text{OPT} = \sum_{i=1}^{k-1} i = \frac{k \cdot (k-1)}{2}$$

$$\text{ALG}/\text{OPT} = \frac{2(k-1)}{k} = 2 - \frac{2}{k}$$

Can we do better?

The best known approximation factor for **MULTIWAYCUT** is $1.2965 - \frac{1}{k}$.
 [Sharma & Vondrák, STOC'14]

Analysis Tight?



$$\text{ALG} = (k - 1)(k - 1)$$

$$\text{OPT} = \sum_{i=1}^{k-1} i = \frac{k \cdot (k-1)}{2}$$

$$\text{ALG}/\text{OPT} = \frac{2(k-1)}{k} = 2 - \frac{2}{k}$$

Can we do better?

The best known approximation factor for **MULTIWAYCUT** is $1.2965 - \frac{1}{k}$.
[Sharma & Vondrák, STOC'14]

MULTIWAYCUT cannot be approximated within factor $1.20016 - O(1/k)$
(unless $P = NP$). [Bérczi, Chandrasekaran, Király & Madan, MP'18]