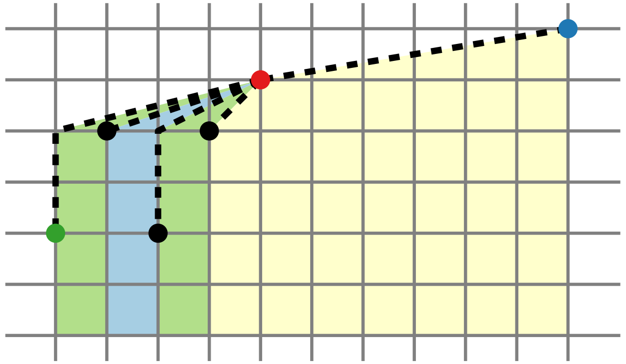
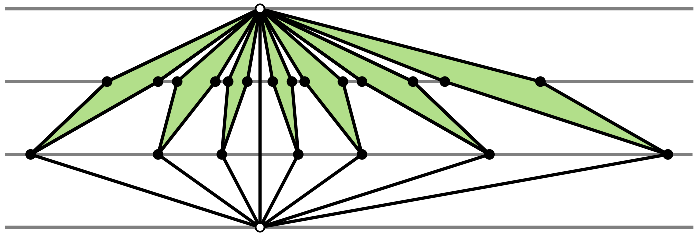
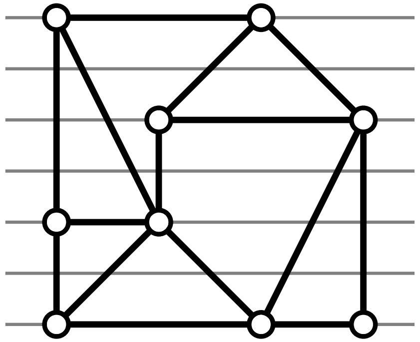
UNIVERSITÄT TRIER

ALGO

Weakly Leveled Planarity with Bounded Span







Michael Bekos

Giordano Da Lozzo

Fabrizio Frati

Siddharth Gupta

Philipp Kindermann

Giuseppe Liotta

Ignaz Rutter

Ioannis Tollis

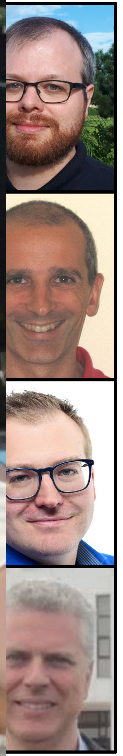
GNV 2023
Graph & Network Vis. Workshop

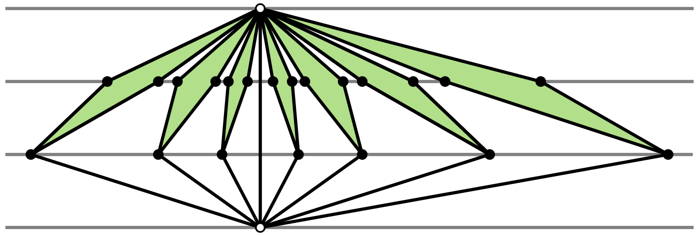
GD 2024
Graph Drawing & Network Vis.

UNIVERSITÄT TRIER

ALGO

Weakly Leveled Planarity with Bounded Span





Philipp Kindermann

Giuseppe Liotta

Ignaz Rutter

Ioannis Tollis

GNV 2023
Graph & Network Vis. Workshop

GD 2024
Graph Drawing & Network Vis.

UNIVERSITÄT TRIER

ALGO

Weakly Leveled Planarity with Bounded Span

(Weakly) leveled planar drawings

Leveled planar drawing

(Weakly) leveled planar drawings

Leveled planar drawing

- Set of horizontal lines (*levels*)

(Weakly) leveled planar drawings

Leveled planar drawing

- Set of horizontal lines (*levels*)

(Weakly) leveled planar drawings

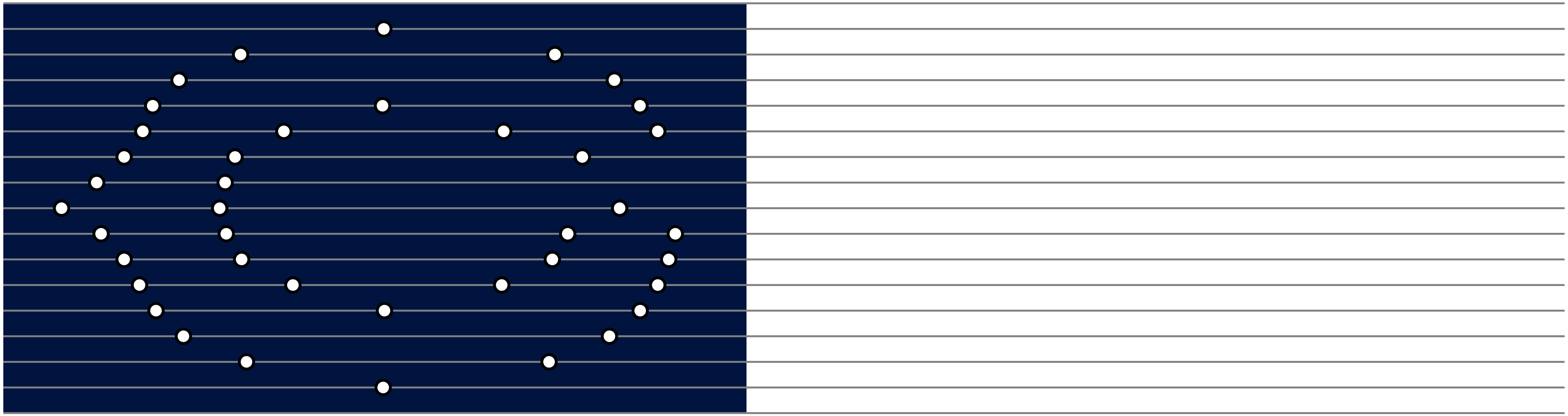
Leveled planar drawing

- Set of horizontal lines (*levels*)
- Every vertex is a point on a level

(Weakly) leveled planar drawings

Leveled planar drawing

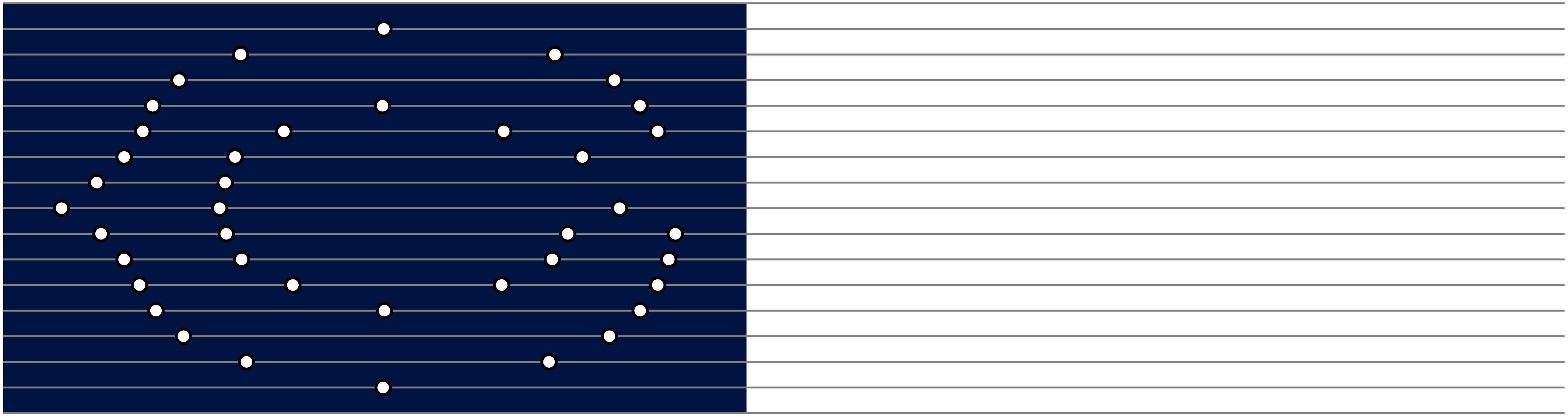
- Set of horizontal lines (*levels*)
- Every vertex is a point on a level



(Weakly) leveled planar drawings

Leveled planar drawing

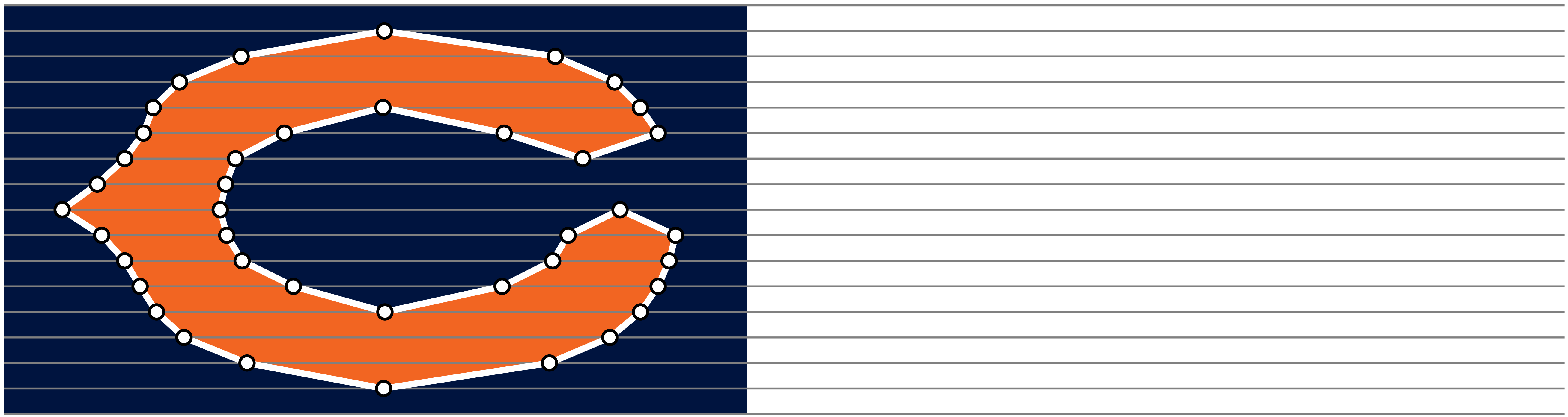
- Set of horizontal lines (*levels*)
- Every vertex is a point on a level
- Edges are only between adjacent levels



(Weakly) leveled planar drawings

Leveled planar drawing

- Set of horizontal lines (*levels*)
- Every vertex is a point on a level
- Edges are only between adjacent levels

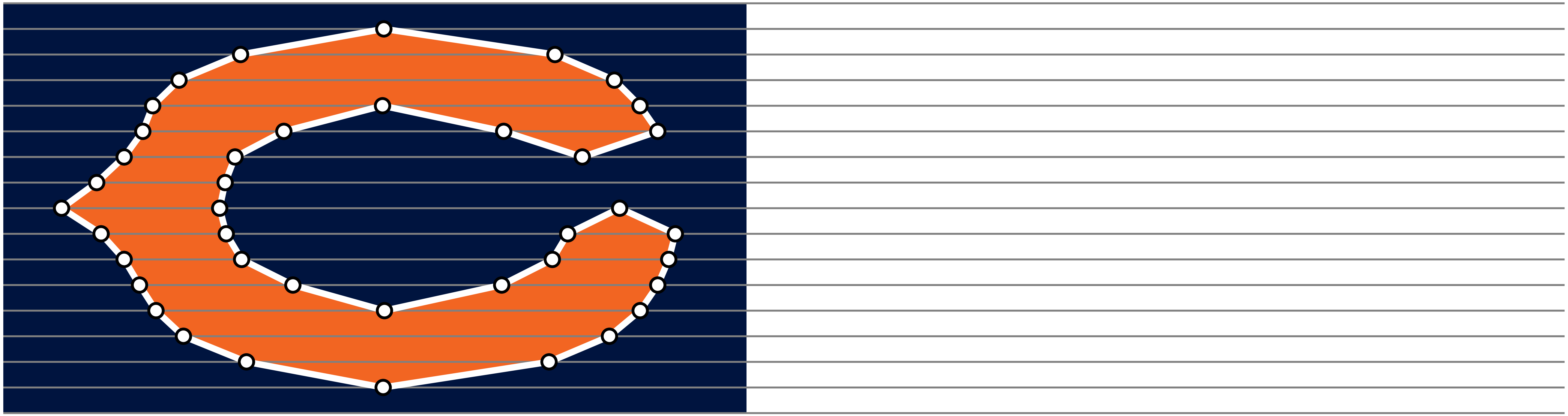


(Weakly) leveled planar drawings

Leveled planar drawing

- Set of horizontal lines (*levels*)
- Every vertex is a point on a level
- Edges are only between adjacent levels

- Drawing is planar



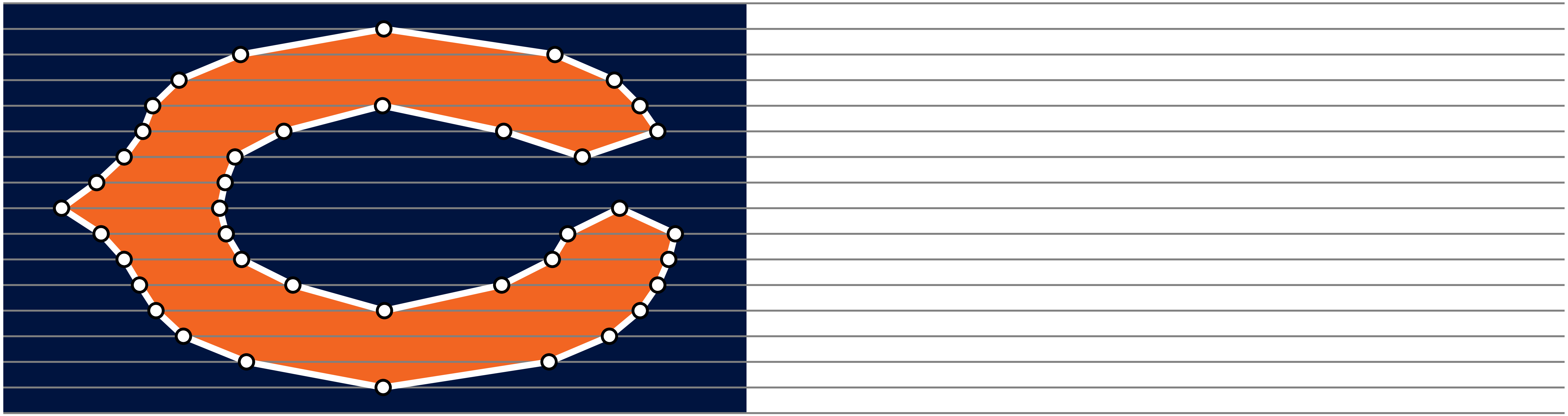
(Weakly) leveled planar drawings

Leveled planar drawing

- Set of horizontal lines (*levels*)
- Every vertex is a point on a level
- Edges are only between adjacent levels

- Drawing is planar

Weakly leveled planar drawing



(Weakly) leveled planar drawings

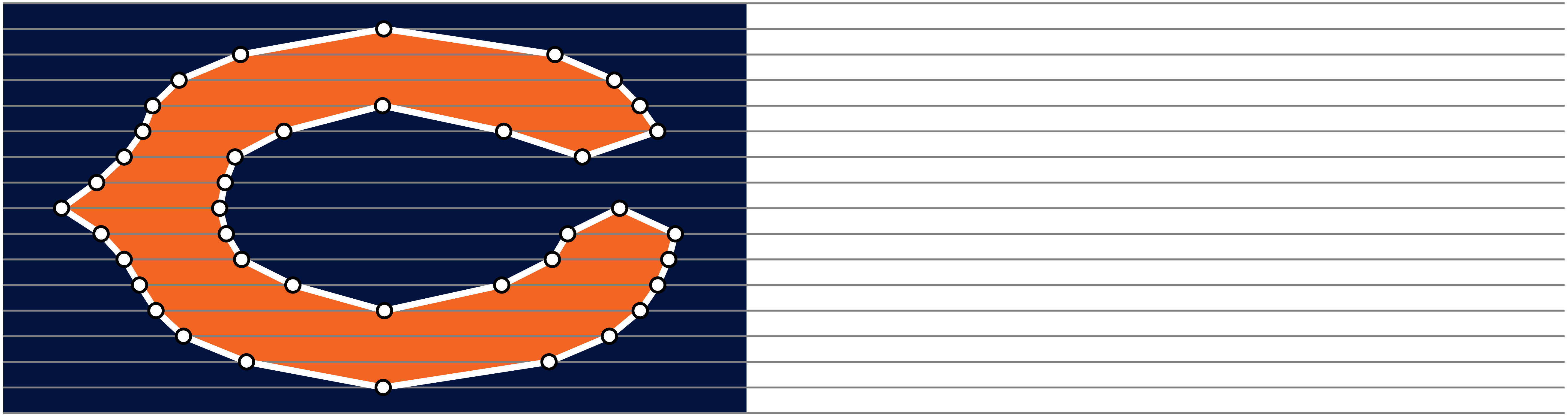
Leveled planar drawing

- Set of horizontal lines (*levels*)
- Every vertex is a point on a level
- Edges are only between adjacent levels

- Drawing is planar

Weakly leveled planar drawing

- . . . or horizontal segments



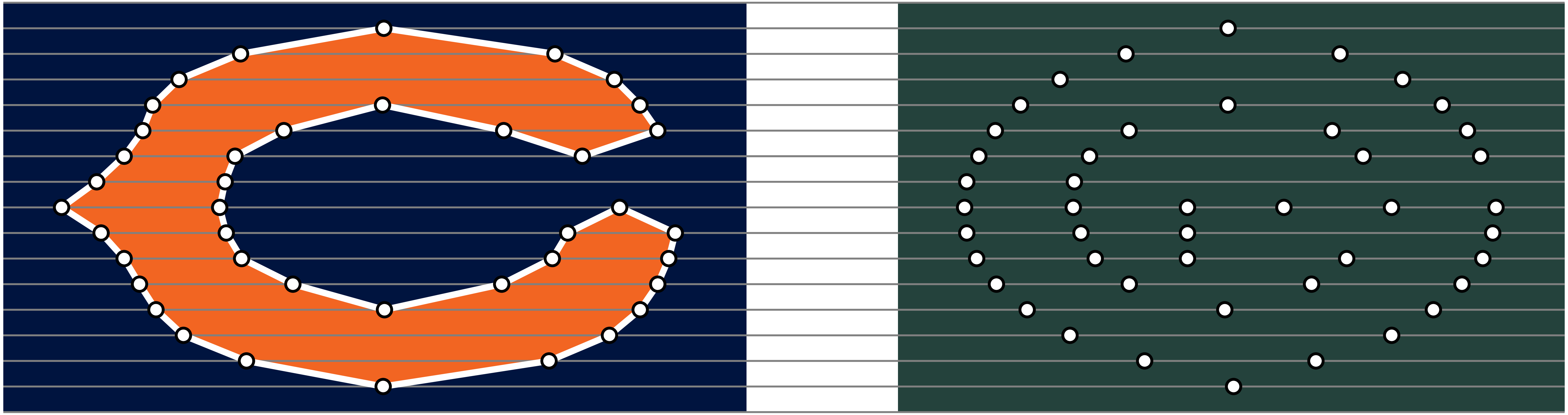
(Weakly) leveled planar drawings

Leveled planar drawing

- Set of horizontal lines (*levels*)
- Every vertex is a point on a level
- Edges are only between adjacent levels
- Drawing is planar

Weakly leveled planar drawing

- . . . or horizontal segments



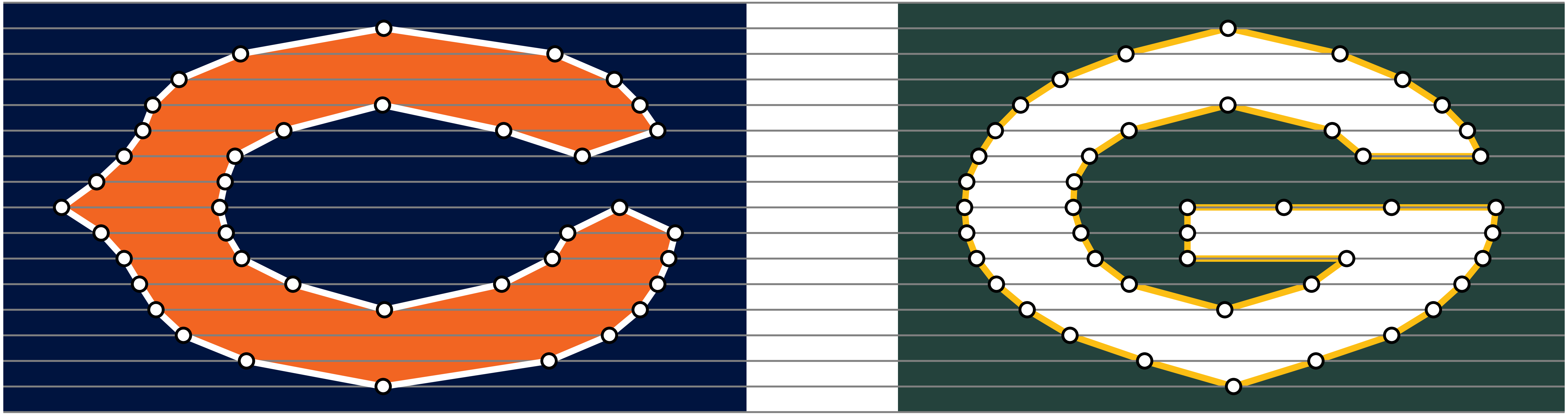
(Weakly) leveled planar drawings

Leveled planar drawing

- Set of horizontal lines (*levels*)
- Every vertex is a point on a level
- Edges are only between adjacent levels
- Drawing is planar

Weakly leveled planar drawing

- ... or horizontal segments



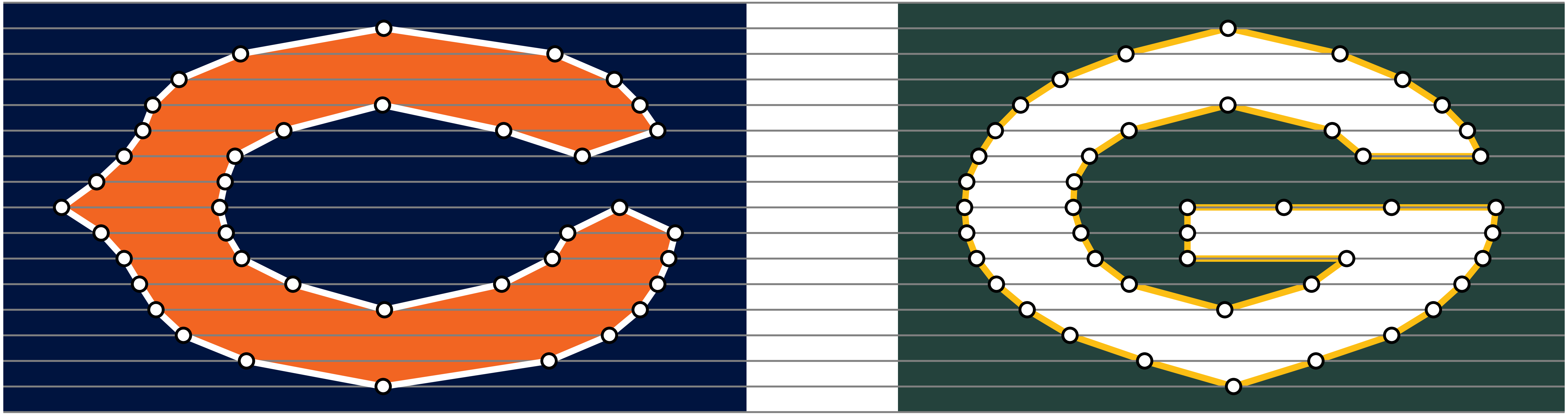
(Weakly) leveled planar drawings

s-span leveled planar drawing

- Set of horizontal lines (*levels*)
- Every vertex is a point on a level
- Edges are only between adjacent levels
- Drawing is planar

s-span weakly leveled planar drawing

- … or horizontal segments



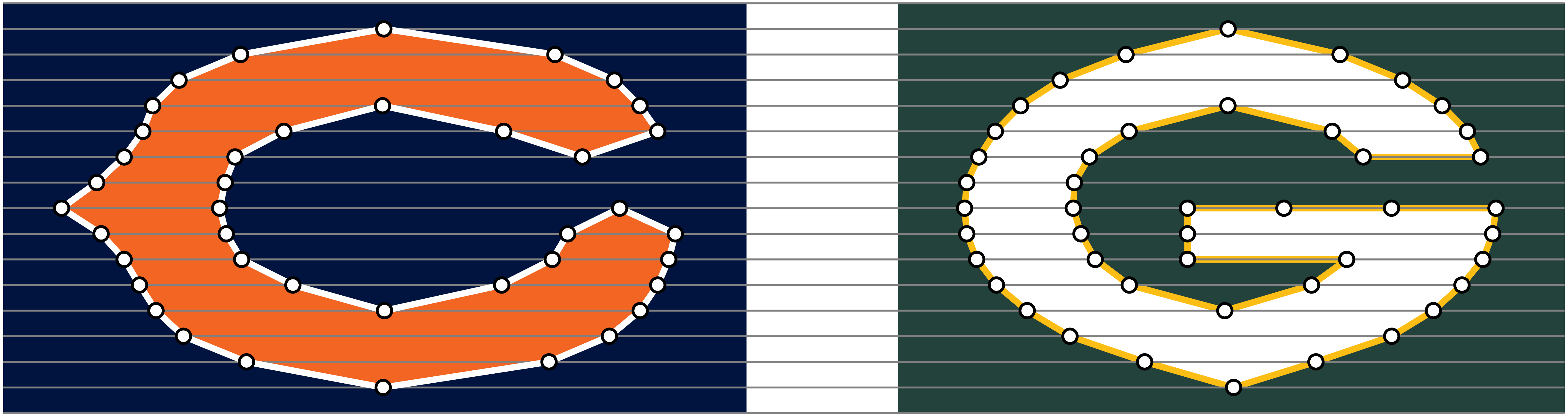
(Weakly) leveled planar drawings

s-span leveled planar drawing

- Set of horizontal lines (*levels*)
- Every vertex is a point on a level
- Edges are y-monotone curves
- Drawing is planar

s-span weakly leveled planar drawing

- ... or horizontal segments



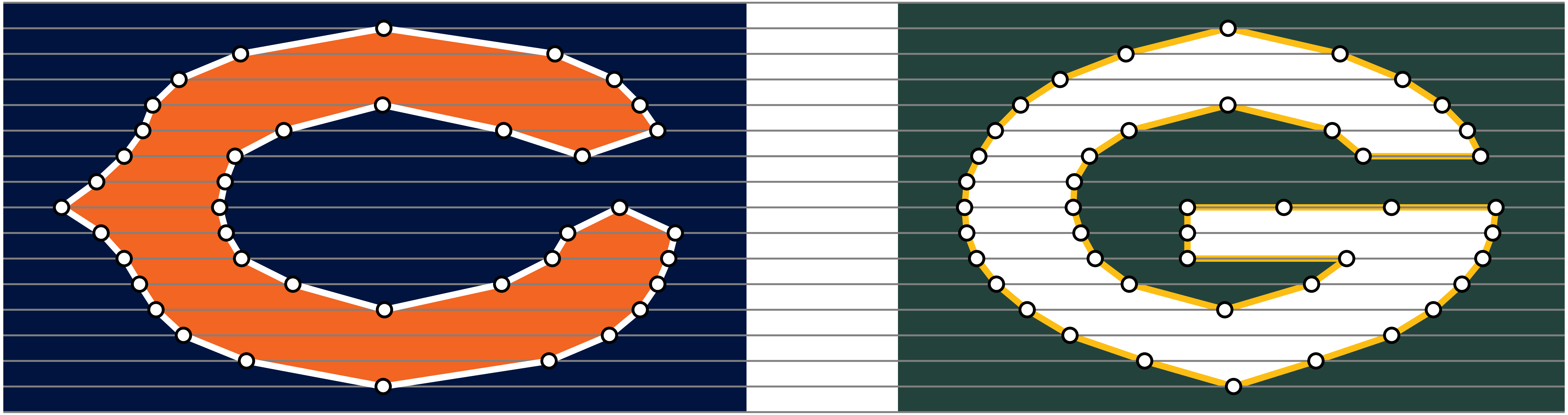
(Weakly) leveled planar drawings

s-span leveled planar drawing

- Set of horizontal lines (*levels*)
- Every vertex is a point on a level
- Edges are y-monotone curves and span at most s levels
- Drawing is planar

s-span weakly leveled planar drawing

- . . . or horizontal segments



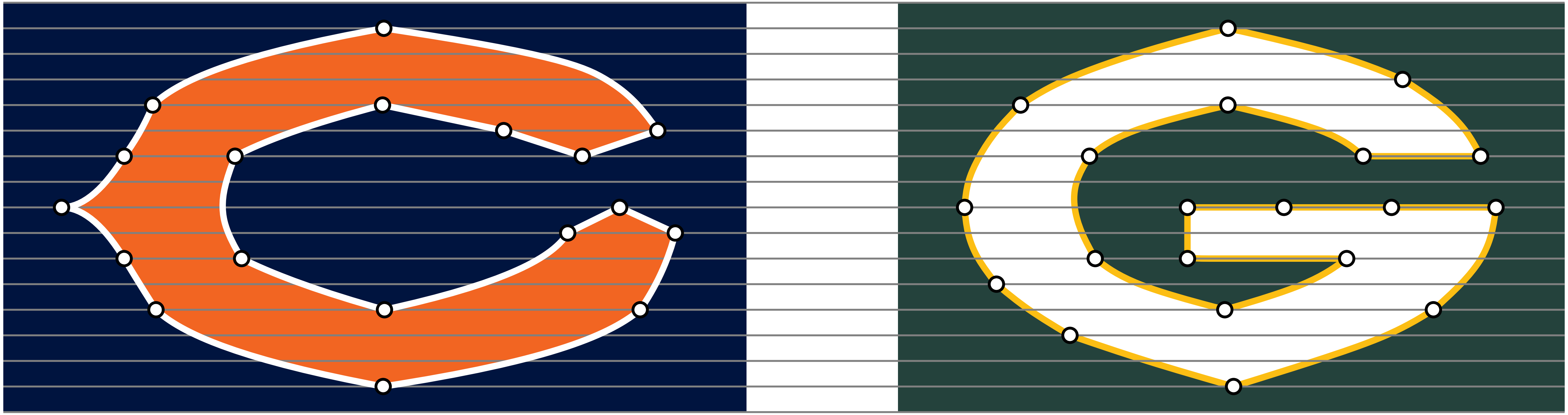
(Weakly) leveled planar drawings

s-span leveled planar drawing

- Set of horizontal lines (*levels*)
- Every vertex is a point on a level
- Edges are y-monotone curves and span at most s levels
- Drawing is planar

s-span weakly leveled planar drawing

- ... or horizontal segments



Overview

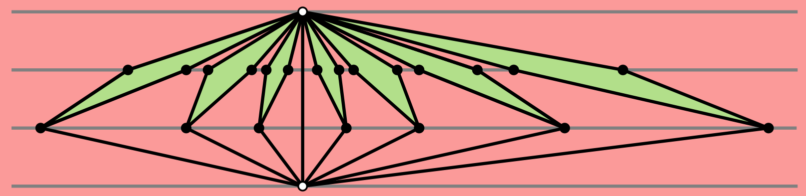
Overview

NP-hardness

Overview

NP-hardness

Reduction from leveled planar

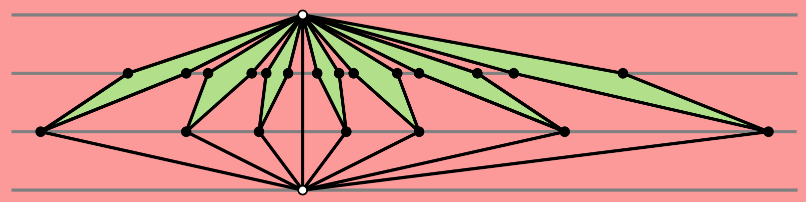


Overview

FPT algorithms

NP-hardness

Reduction from leveled planar

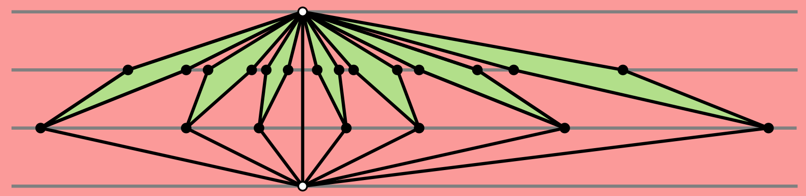


Overview

FPT algorithms

NP-hardness

Reduction from leveled planar



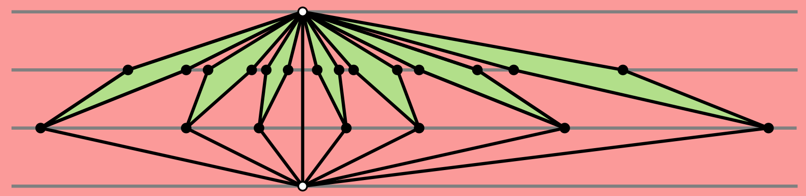
Combinatorial results

Overview

FPT algorithms

NP-hardness

Reduction from leveled planar



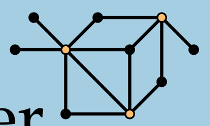
Combinatorial results

Implications

Overview

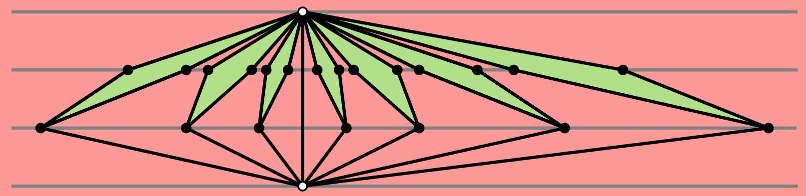
FPT algorithms

Linear kernel in size of vertex cover



NP-hardness

Reduction from leveled planar



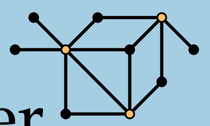
Combinatorial results

Implications

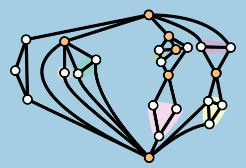
Overview

FPT algorithms

Linear kernel in size of vertex cover

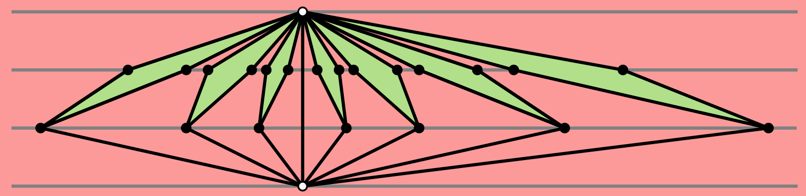


FPT in size of b-modulator $(+b)$



NP-hardness

Reduction from leveled planar

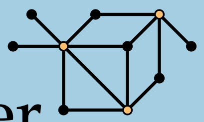


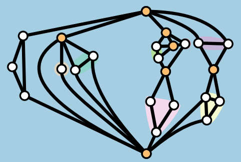
Combinatorial results

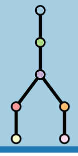
Implications

Overview

FPT algorithms

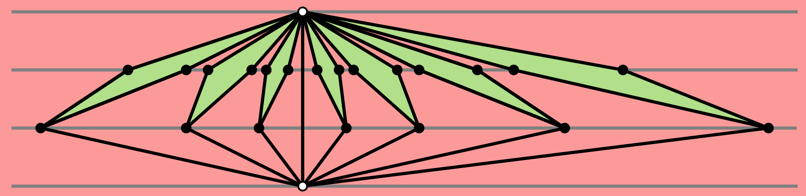
Linear kernel in size of vertex cover 

FPT in size of b-modulator $(+b)$ 

FPT in treedepth 

NP-hardness

Reduction from leveled planar



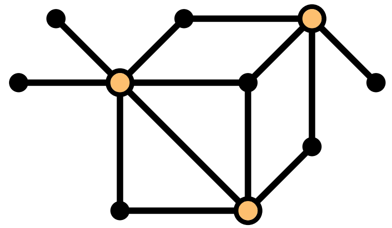
Combinatorial results

Implications

Vertex Cover

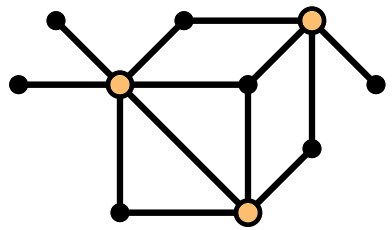
A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$.

Vertex Cover



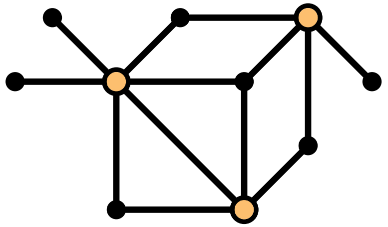
A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$.

Vertex Cover



A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

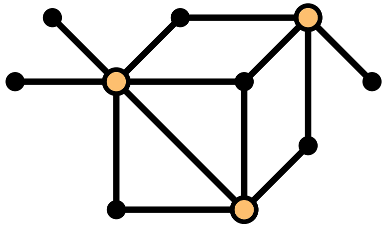
Vertex Cover



A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

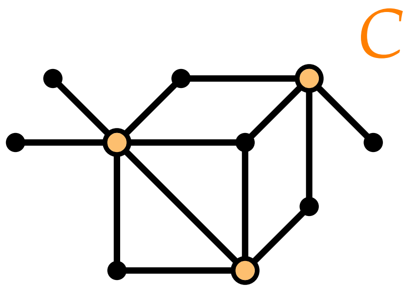
Vertex Cover



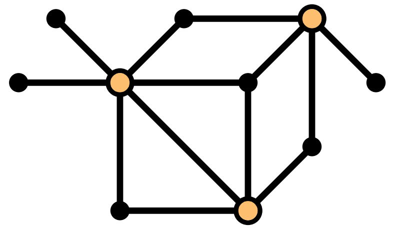
A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

Let C be a vertex cover of G of size k.



Vertex Cover

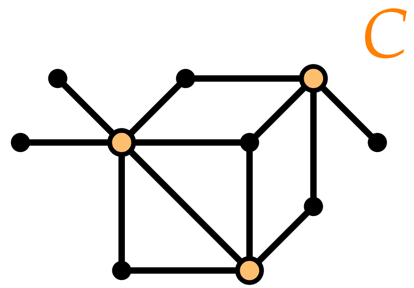


A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

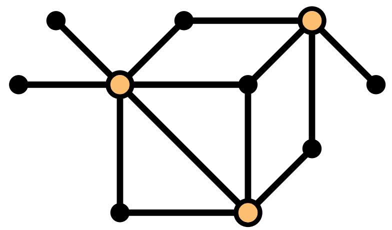
Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

Let C be a vertex cover of G of size k.

1. Remove all degree-1 vertices from $G \setminus C$



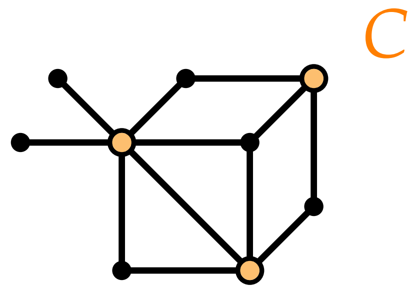
Vertex Cover



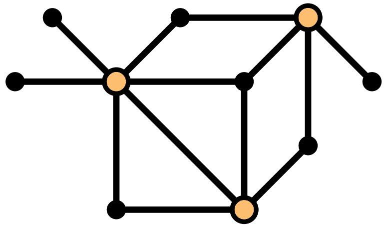
A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

Let C be a vertex cover of G of size k.
1. Remove all degree-1 vertices from $G \setminus C$



Vertex Cover

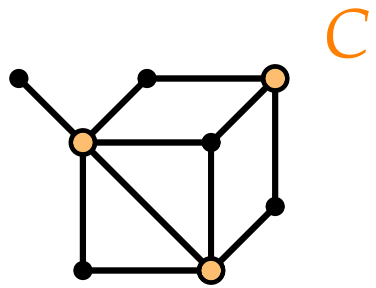


A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

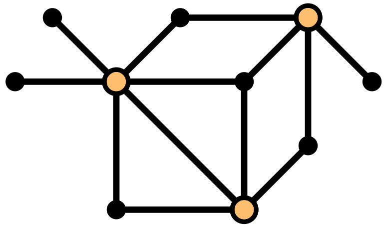
Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

Let C be a vertex cover of G of size k.

1. Remove all degree-1 vertices from $G \setminus C$



Vertex Cover

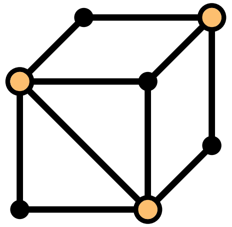


A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

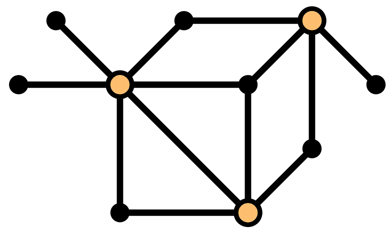
Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

Let C be a vertex cover of G of size k.
1. Remove all degree-1 vertices from $G \setminus C$

C



Vertex Cover

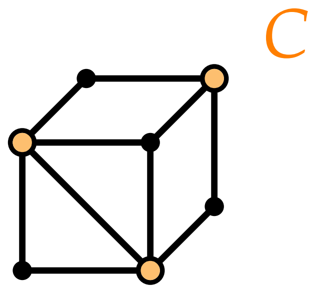


A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

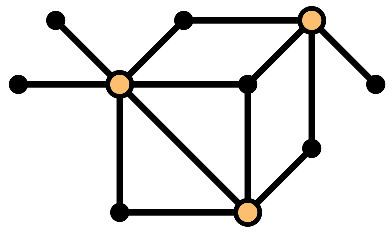
Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

Let C be a vertex cover of G of size k.

1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$



Vertex Cover

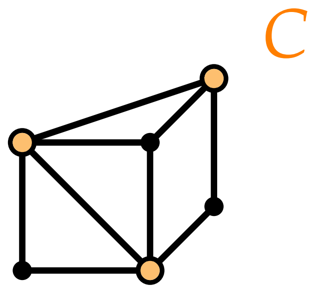


A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

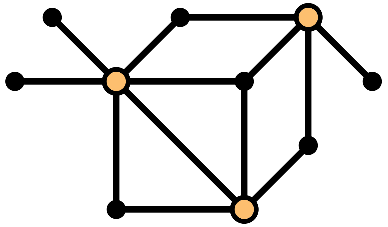
Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

Let C be a vertex cover of G of size k.

1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$



Vertex Cover

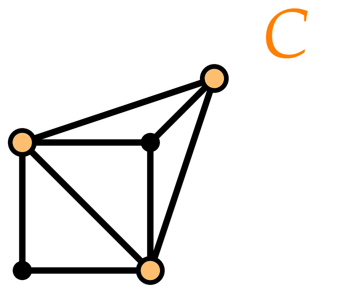


A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

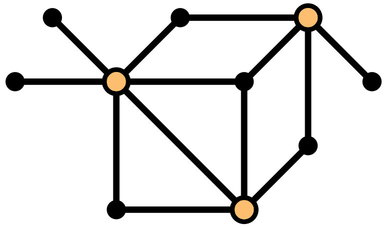
Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

Let C be a vertex cover of G of size k.

1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$



Vertex Cover

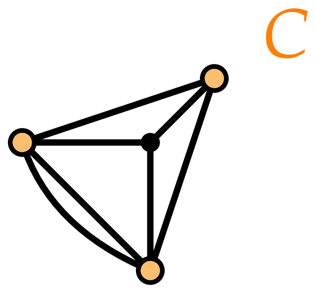


A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

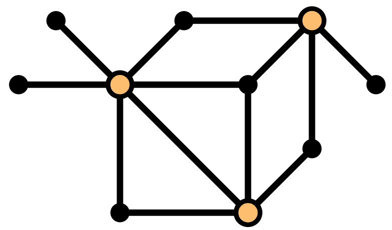
Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

Let C be a vertex cover of G of size k.

1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$



Vertex Cover



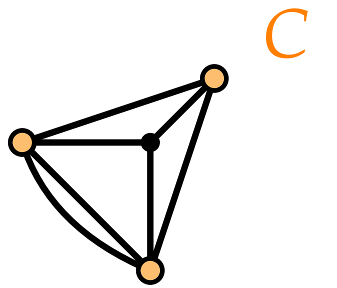
A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

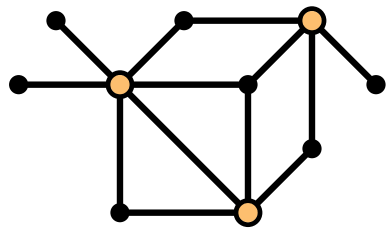
Let C be a vertex cover of G of size k.

1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$

➡ The trimmed (multi-)graph has at most $3k$ vertices.



Vertex Cover

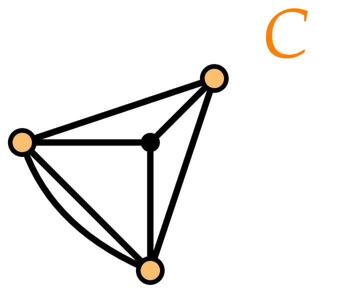


A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

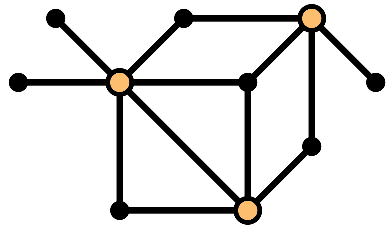
Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

Let C be a vertex cover of G of size k.

1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$
   ➡ The trimmed (multi-)graph has at most $3k$ vertices.
3. Create a $3k$-span (weakly) leveled planar drawing.



Vertex Cover

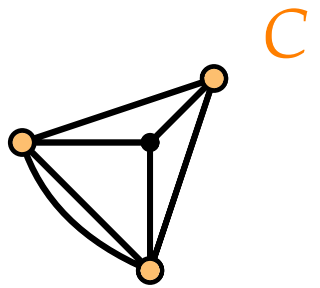


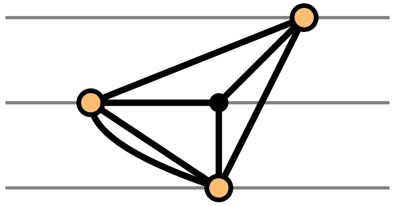
A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

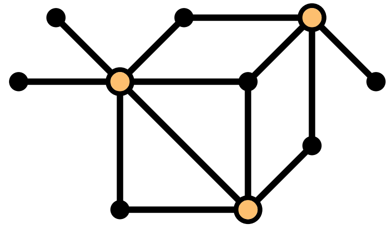
Let C be a vertex cover of G of size k.

1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$
   ➡ The trimmed (multi-)graph has at most $3k$ vertices.
3. Create a $3k$-span (weakly) leveled planar drawing.





Vertex Cover

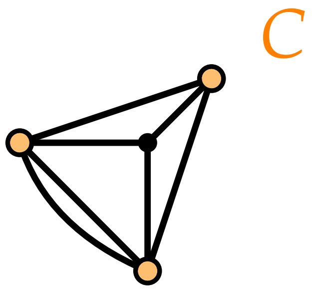


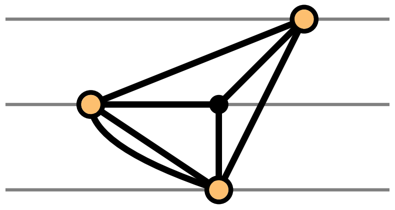
A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

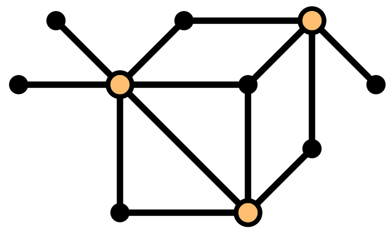
Let C be a vertex cover of G of size k.

1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$
   ➡ The trimmed (multi-)graph has at most $3k$ vertices.
3. Create a $3k$-span (weakly) leveled planar drawing.
4. Double the levels





Vertex Cover

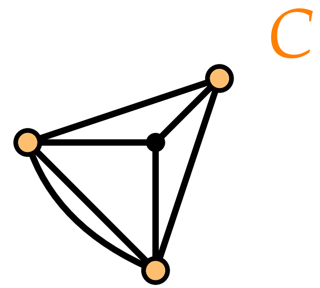


A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

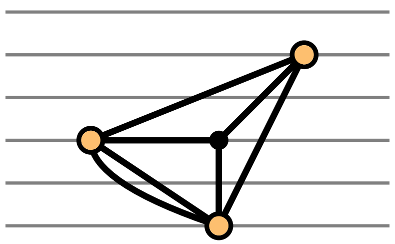
Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

Let C be a vertex cover of G of size k.

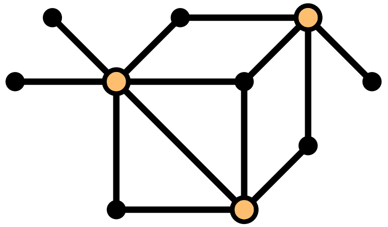
1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$
   ➡ The trimmed (multi-)graph has at most $3k$ vertices.



3. Create a $3k$-span (weakly) leveled planar drawing.
4. Double the levels



Vertex Cover

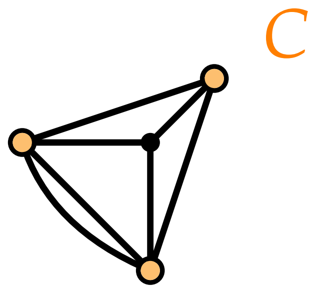


A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

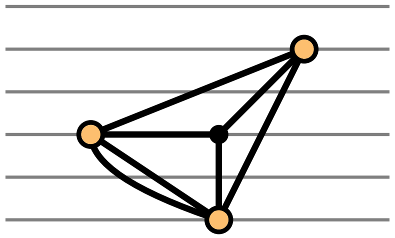
Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

Let C be a vertex cover of G of size k.

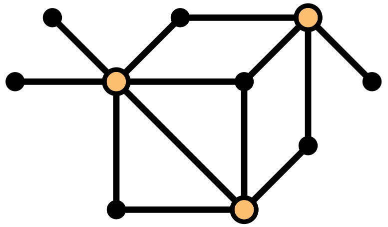
1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$
   ➡ The trimmed (multi-)graph has at most $3k$ vertices.



3. Create a $3k$-span (weakly) leveled planar drawing.
4. Double the levels
   ➡ The span becomes at most $6k$



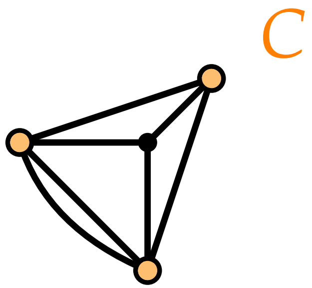
Vertex Cover



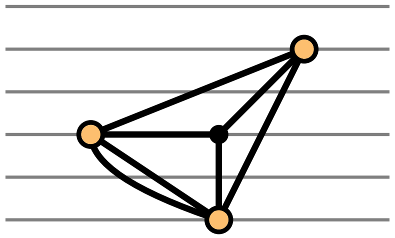
A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

Let C be a vertex cover of G of size k.

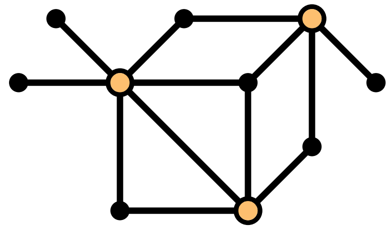


1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$
   ➡ The trimmed (multi-)graph has at most $3k$ vertices.
3. Create a $3k$-span (weakly) leveled planar drawing.
4. Double the levels
   ➡ The span becomes at most $6k$



5. Re-insert degree-1 vertices above their neighbor

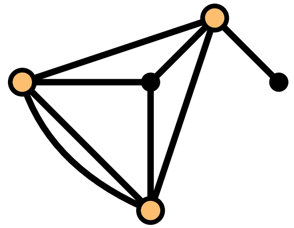
Vertex Cover



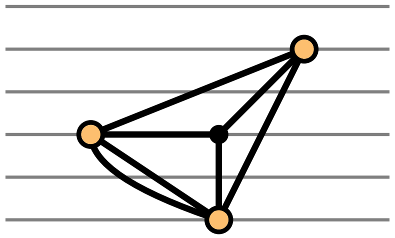
A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

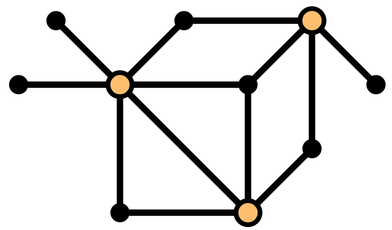
Let C be a vertex cover of G of size k.



1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$
   ➡ The trimmed (multi-)graph has at most $3k$ vertices.
3. Create a $3k$-span (weakly) leveled planar drawing.
4. Double the levels
   ➡ The span becomes at most $6k$
5. Re-insert degree-1 vertices above their neighbor



Vertex Cover

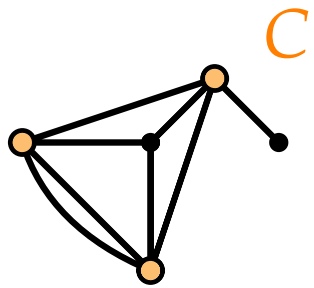


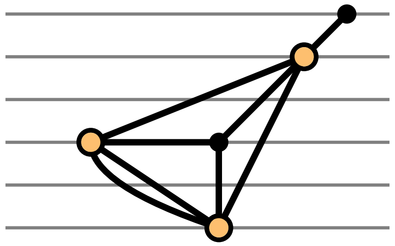
A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

> **Lemma.**
> Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

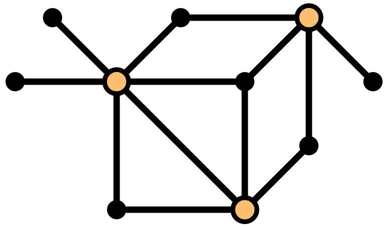
Let C be a vertex cover of G of size k.

1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$
   ➡ The trimmed (multi-)graph has at most $3k$ vertices.
3. Create a $3k$-span (weakly) leveled planar drawing.
4. Double the levels
   ➡ The span becomes at most $6k$
5. Re-insert degree-1 vertices above their neighbor





Vertex Cover

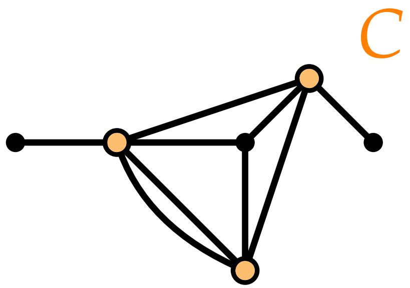


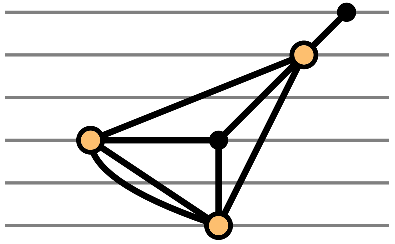
A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

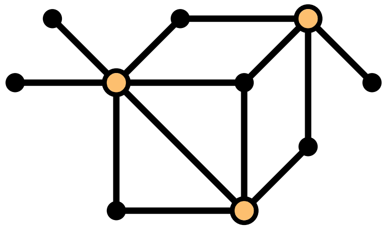
Let C be a vertex cover of G of size k.

1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$
   ➡ The trimmed (multi-)graph has at most $3k$ vertices.
3. Create a $3k$-span (weakly) leveled planar drawing.
4. Double the levels
   ➡ The span becomes at most $6k$
5. Re-insert degree-1 vertices above their neighbor





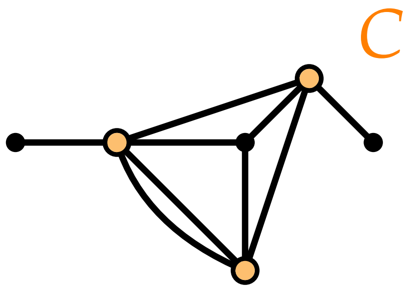
Vertex Cover



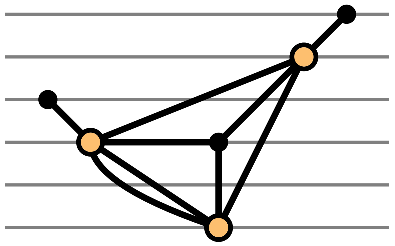
A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

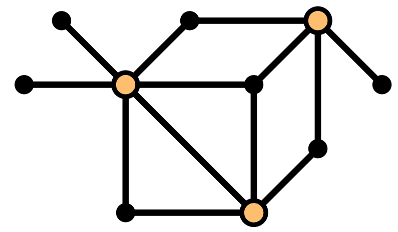
Let C be a vertex cover of G of size k.



1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$
   ➡ The trimmed (multi-)graph has at most $3k$ vertices.
3. Create a $3k$-span (weakly) leveled planar drawing.
4. Double the levels
   ➡ The span becomes at most $6k$
5. Re-insert degree-1 vertices above their neighbor



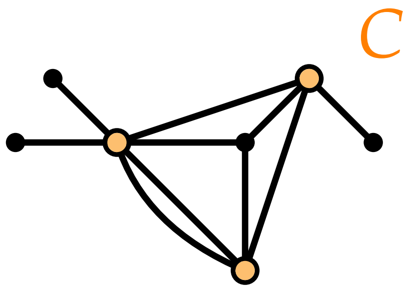
Vertex Cover



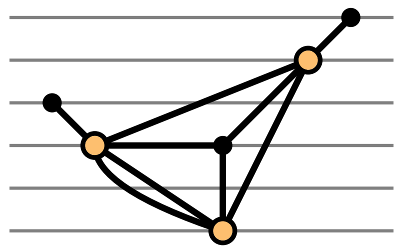
A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

Let C be a vertex cover of G of size k.

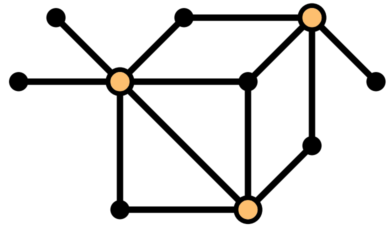


1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$
   ➡ The trimmed (multi-)graph has at most $3k$ vertices.
3. Create a $3k$-span (weakly) leveled planar drawing.
4. Double the levels
   ➡ The span becomes at most $6k$



5. Re-insert degree-1 vertices above their neighbor

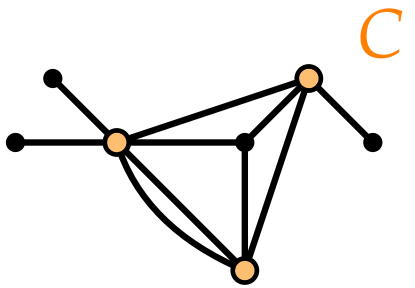
Vertex Cover



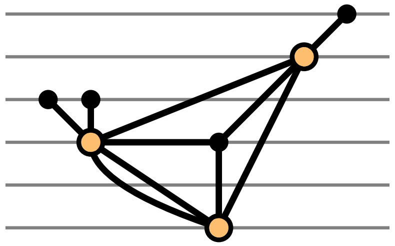
A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

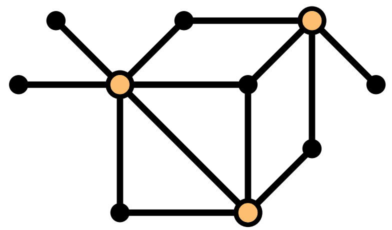
Let C be a vertex cover of G of size k.



1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$
   ➡ The trimmed (multi-)graph has at most $3k$ vertices.
3. Create a $3k$-span (weakly) leveled planar drawing.
4. Double the levels
   ➡ The span becomes at most $6k$
5. Re-insert degree-1 vertices above their neighbor



Vertex Cover

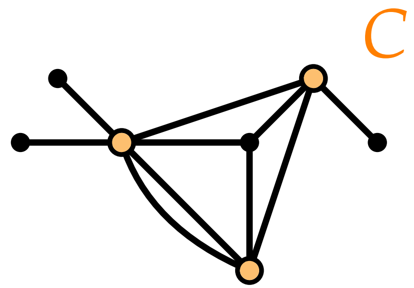


A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

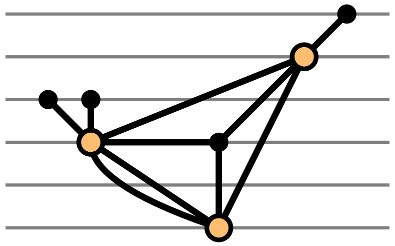
Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

Let C be a vertex cover of G of size k.

1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$
   ➡ The trimmed (multi-)graph has at most $3k$ vertices.

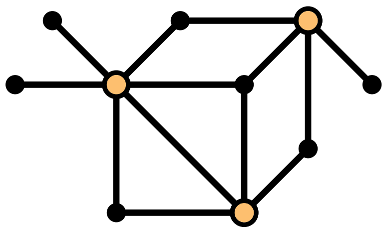


3. Create a $3k$-span (weakly) leveled planar drawing.
4. Double the levels
   ➡ The span becomes at most $6k$



5. Re-insert degree-1 vertices above their neighbor
6. Re-insert degree-2 vertices at intersection of contracted edge with level

Vertex Cover

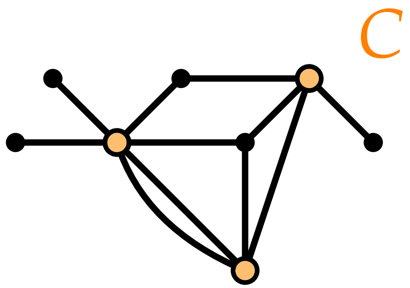


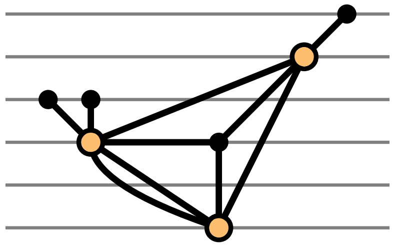
A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

> **Lemma.**
> Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

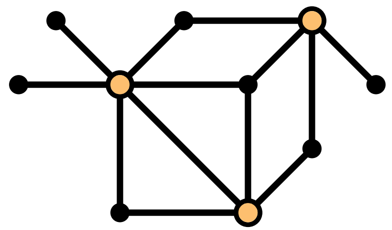
Let C be a vertex cover of G of size k.

1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$
   ➡ The trimmed (multi-)graph has at most $3k$ vertices.
3. Create a $3k$-span (weakly) leveled planar drawing.
4. Double the levels
   ➡ The span becomes at most $6k$
5. Re-insert degree-1 vertices above their neighbor
6. Re-insert degree-2 vertices at intersection of contracted edge with level





Vertex Cover

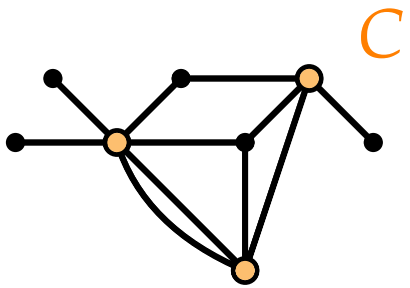


A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

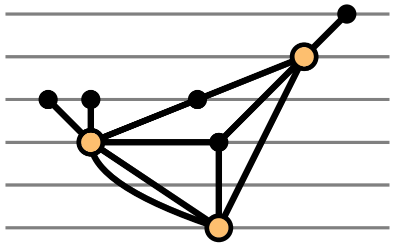
Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

Let C be a vertex cover of G of size k.

1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$
   ➡ The trimmed (multi-)graph has at most $3k$ vertices.

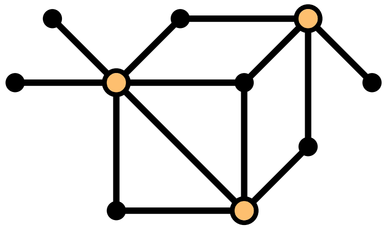


3. Create a $3k$-span (weakly) leveled planar drawing.
4. Double the levels
   ➡ The span becomes at most $6k$



5. Re-insert degree-1 vertices above their neighbor
6. Re-insert degree-2 vertices at intersection of contracted edge with level

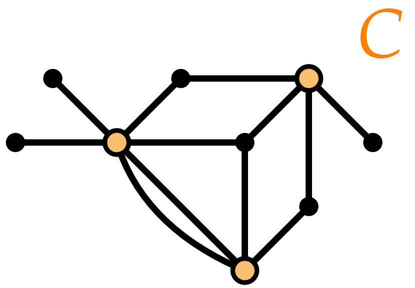
Vertex Cover



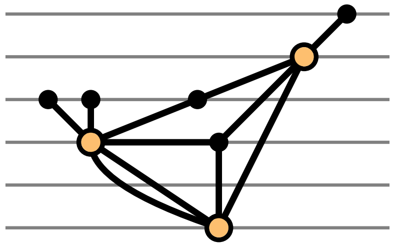
A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

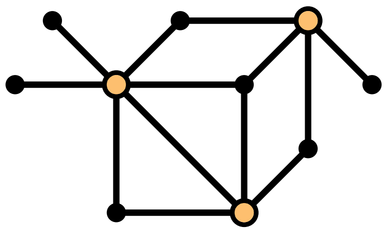
Let C be a vertex cover of G of size k.



1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$
   ➡ The trimmed (multi-)graph has at most $3k$ vertices.
3. Create a $3k$-span (weakly) leveled planar drawing.
4. Double the levels
   ➡ The span becomes at most $6k$
5. Re-insert degree-1 vertices above their neighbor
6. Re-insert degree-2 vertices at intersection of contracted edge with level



Vertex Cover

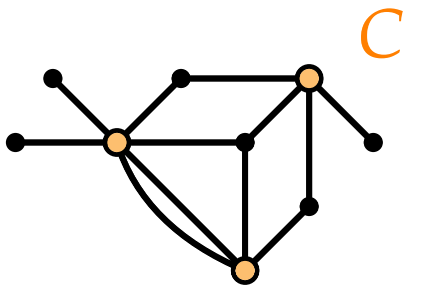


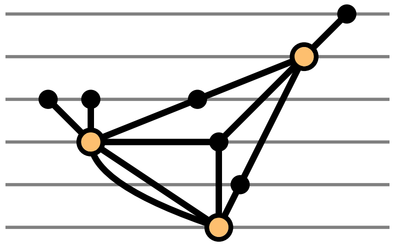
A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

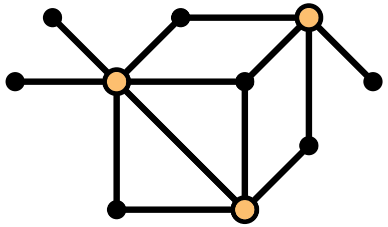
Let C be a vertex cover of G of size k.

1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$
   ➡ The trimmed (multi-)graph has at most $3k$ vertices.
3. Create a $3k$-span (weakly) leveled planar drawing.
4. Double the levels
   ➡ The span becomes at most $6k$
5. Re-insert degree-1 vertices above their neighbor
6. Re-insert degree-2 vertices at intersection of contracted edge with level





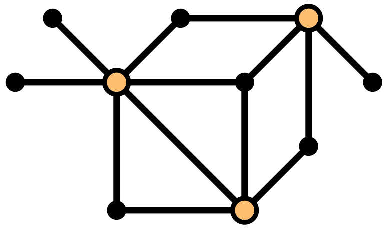
Vertex Cover



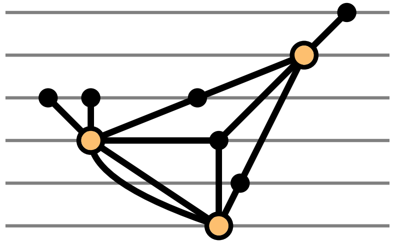
A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

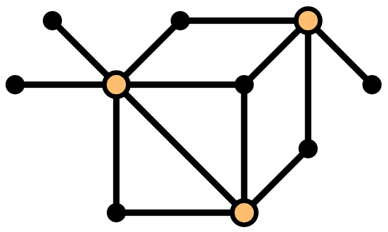
Let C be a vertex cover of G of size k.



1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$
   ➡ The trimmed (multi-)graph has at most $3k$ vertices.
3. Create a $3k$-span (weakly) leveled planar drawing.
4. Double the levels
   ➡ The span becomes at most $6k$
5. Re-insert degree-1 vertices above their neighbor
6. Re-insert degree-2 vertices at intersection of contracted edge with level



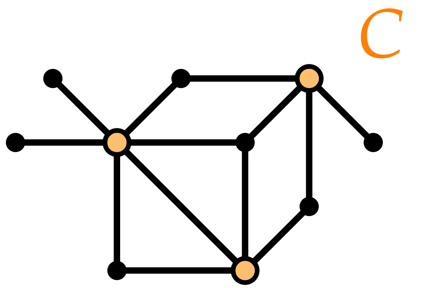
Vertex Cover



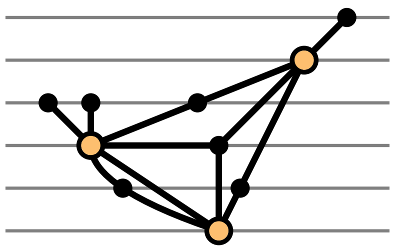
A **vertex cover** of a graph G is a set V' of vertices such that, for every edge $uv \in E$, $u \in V'$ or $v \in V'$. We want to find a poly. kernel in the size of the smallest vertex cover.

Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

Let C be a vertex cover of G of size k.



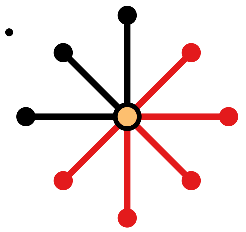
1. Remove all degree-1 vertices from $G \setminus C$
2. Contract all degree-2 vertices from $G \setminus C$
   ➡ The trimmed (multi-)graph has at most $3k$ vertices.
3. Create a $3k$-span (weakly) leveled planar drawing.
4. Double the levels
   ➡ The span becomes at most $6k$
5. Re-insert degree-1 vertices above their neighbor
6. Re-insert degree-2 vertices at intersection of contracted edge with level



Vertex Cover: Reduction Rules

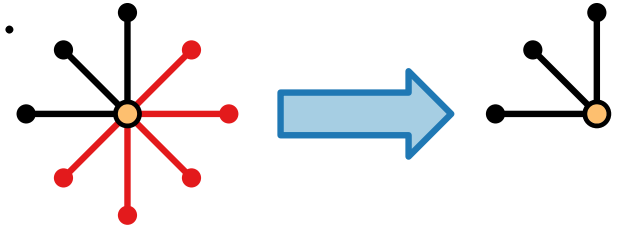
Vertex Cover: Reduction Rules

Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.



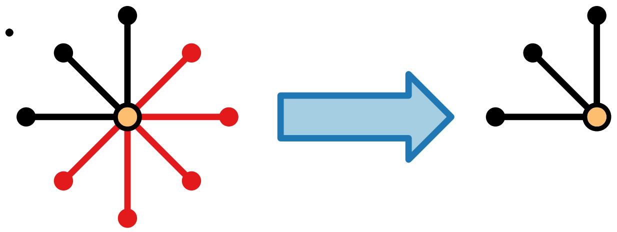
Vertex Cover: Reduction Rules

Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.



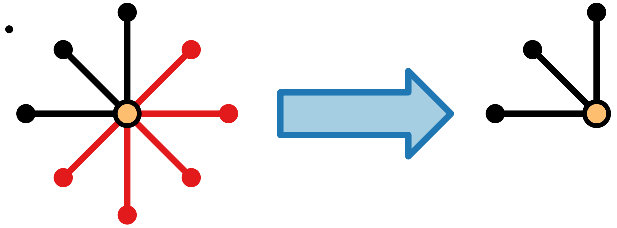
Vertex Cover: Reduction Rules

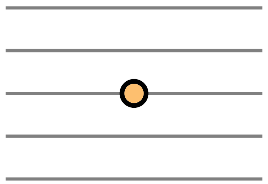
Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.



Vertex Cover: Reduction Rules

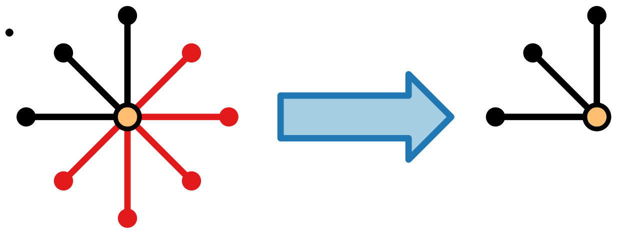
Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.

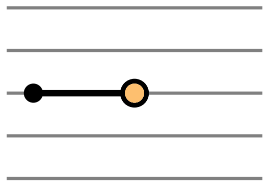




Vertex Cover: Reduction Rules

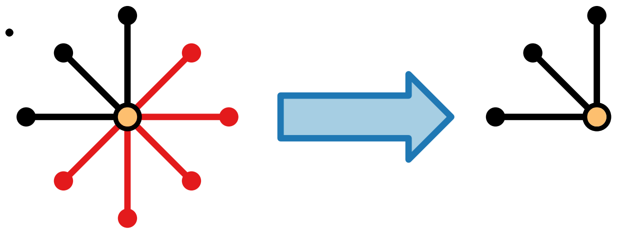
Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.

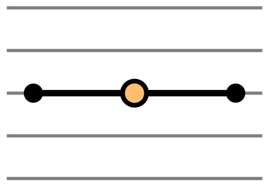




Vertex Cover: Reduction Rules

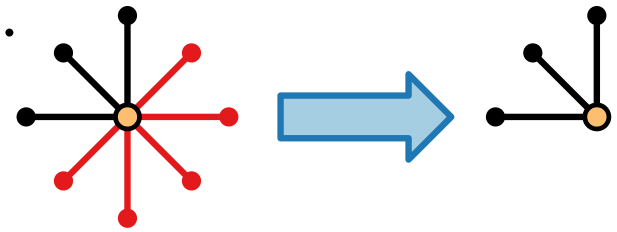
Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.

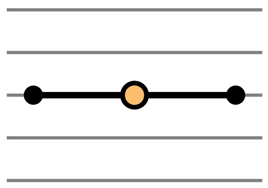




Vertex Cover: Reduction Rules

Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.

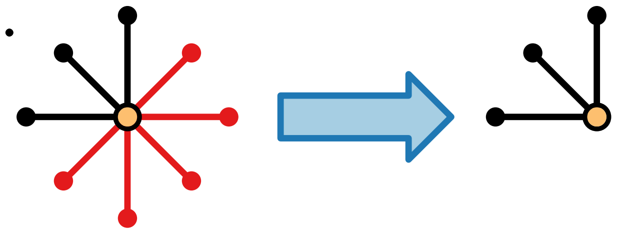


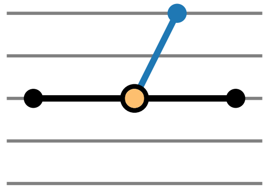


At most 2 neighbors on same level

Vertex Cover: Reduction Rules

Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.

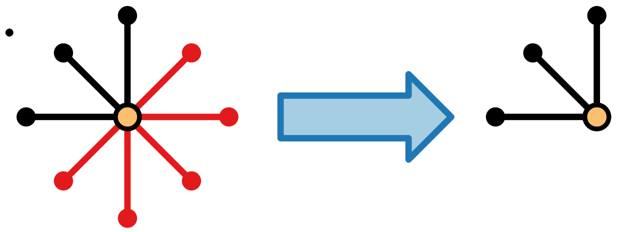


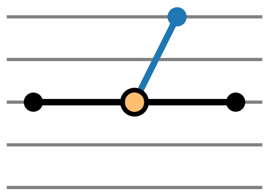


At most 2 neighbors on same level

Vertex Cover: Reduction Rules

Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.

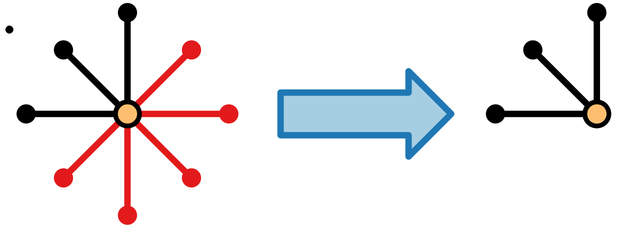


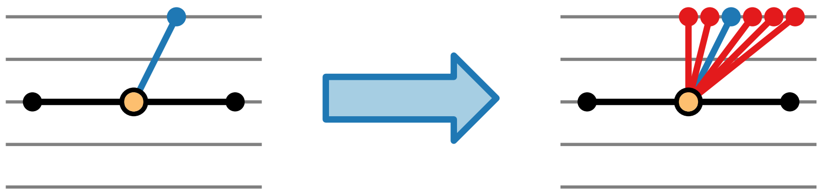


At most 2 neighbors on same level ➡ insert all other next to third neighbor

Vertex Cover: Reduction Rules

Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.

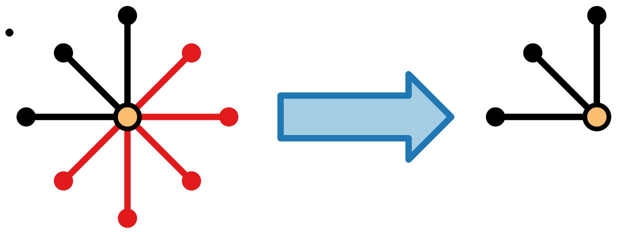


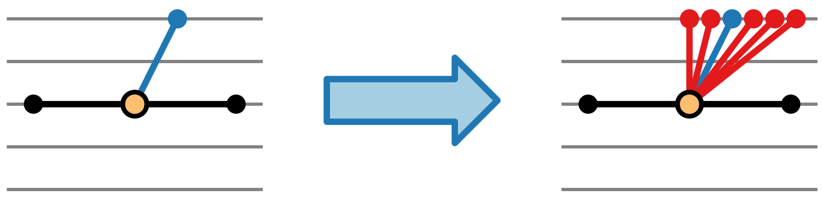


At most 2 neighbors on same level ➡ insert all other next to third neighbor

Vertex Cover: Reduction Rules

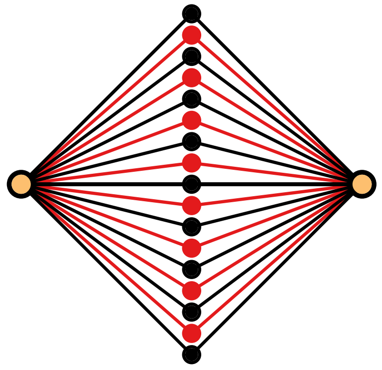
Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.





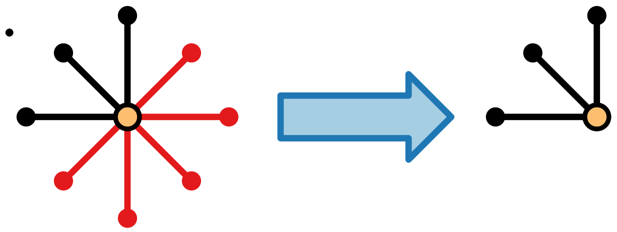
At most 2 neighbors on same level ➡ insert all other next to third neighbor

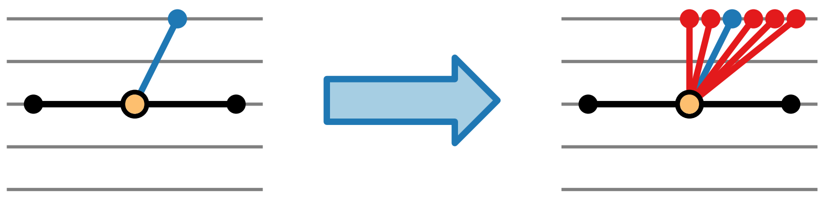
Rule 2: If any two vertices $c, d \in C$ have more than $4s + 5$ common degree-2 neighbors, remove all but $4s + 5$ of them.



Vertex Cover: Reduction Rules

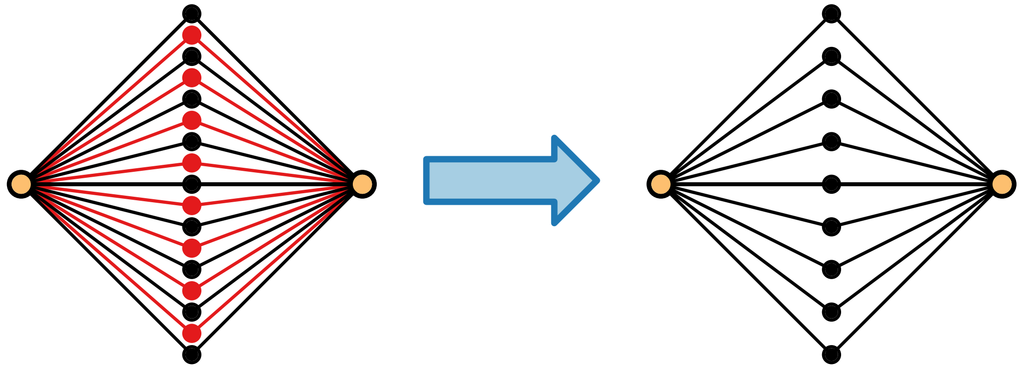
Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.





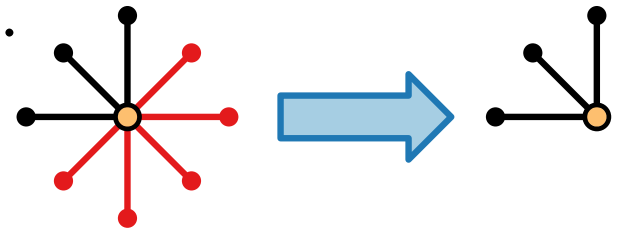
At most 2 neighbors on same level ➡ insert all other next to third neighbor

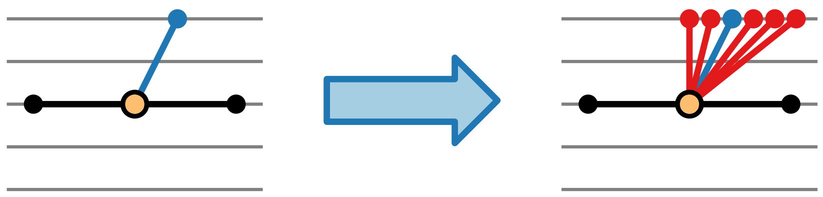
Rule 2: If any two vertices $c, d \in C$ have more than $4s + 5$ common degree-2 neighbors, remove all but $4s + 5$ of them.



Vertex Cover: Reduction Rules

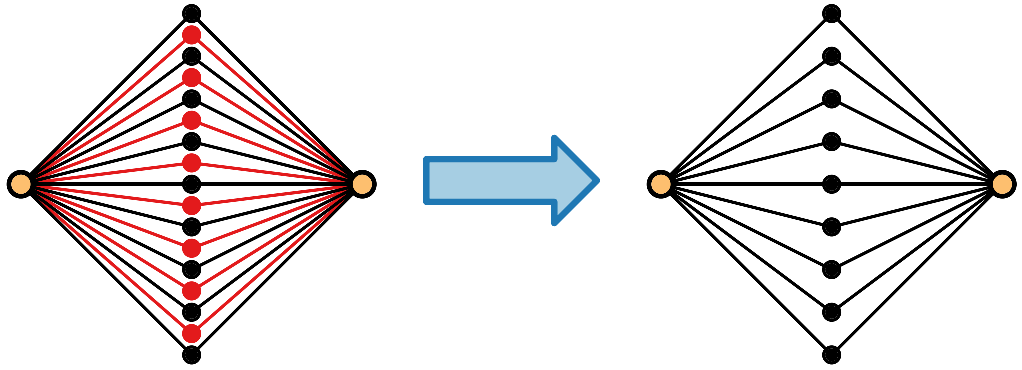
Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.

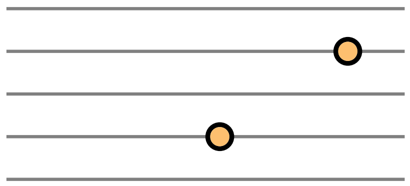




At most 2 neighbors on same level ➡ insert all other next to third neighbor

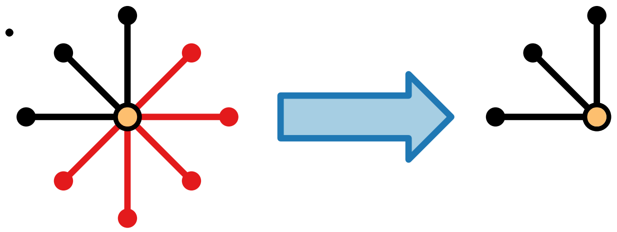
Rule 2: If any two vertices $c, d \in C$ have more than $4s + 5$ common degree-2 neighbors, remove all but $4s + 5$ of them.

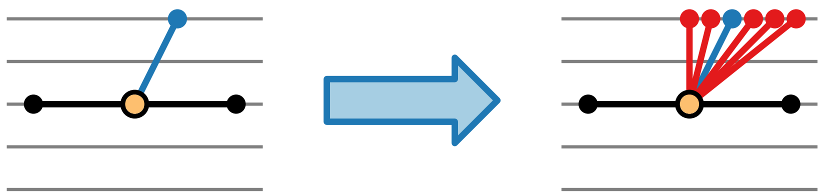




Vertex Cover: Reduction Rules

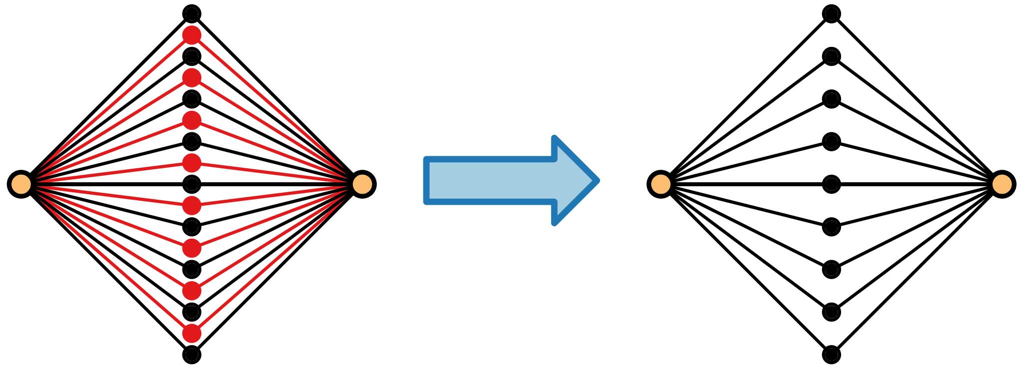
Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.

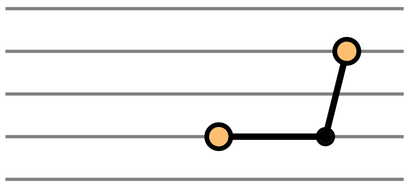




At most 2 neighbors on same level ➡ insert all other next to third neighbor

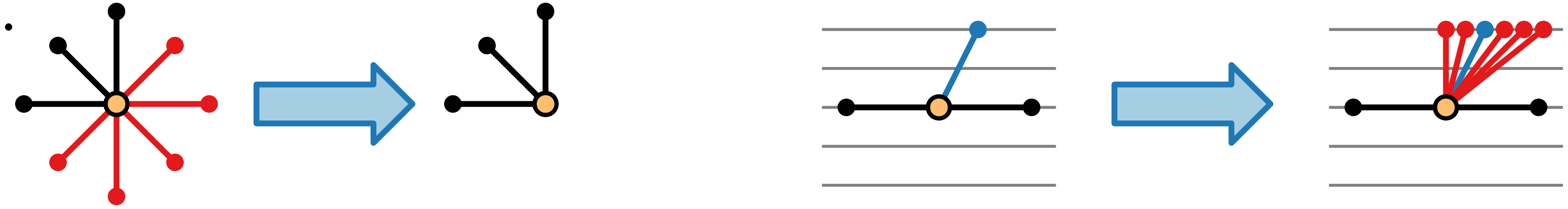
Rule 2: If any two vertices $c, d \in C$ have more than $4s + 5$ common degree-2 neighbors, remove all but $4s + 5$ of them.





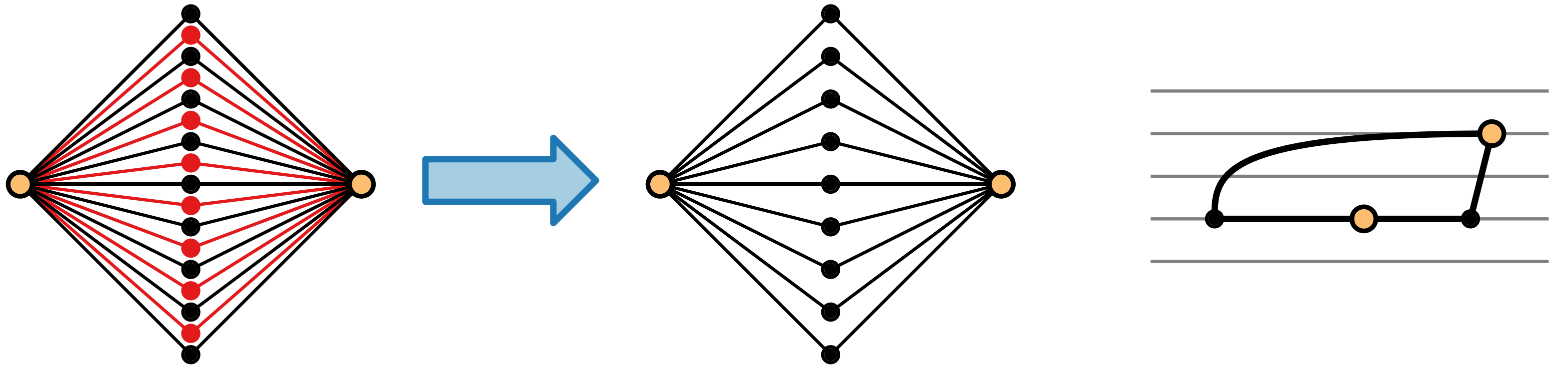
Vertex Cover: Reduction Rules

Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.



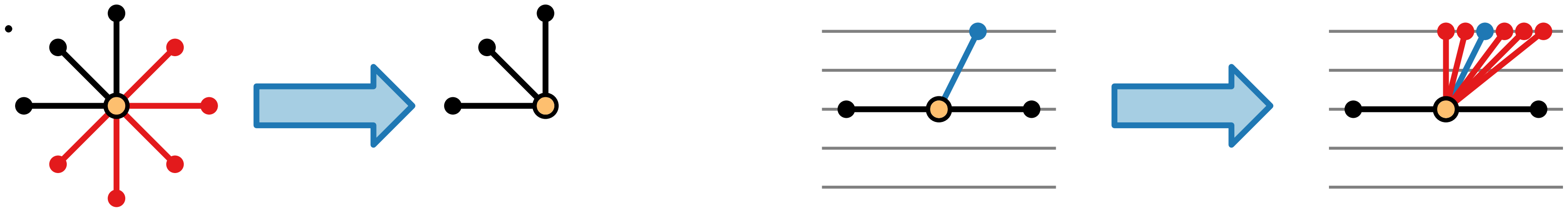
At most 2 neighbors on same level ➡ insert all other next to third neighbor

Rule 2: If any two vertices $c, d \in C$ have more than $4s + 5$ common degree-2 neighbors, remove all but $4s + 5$ of them.



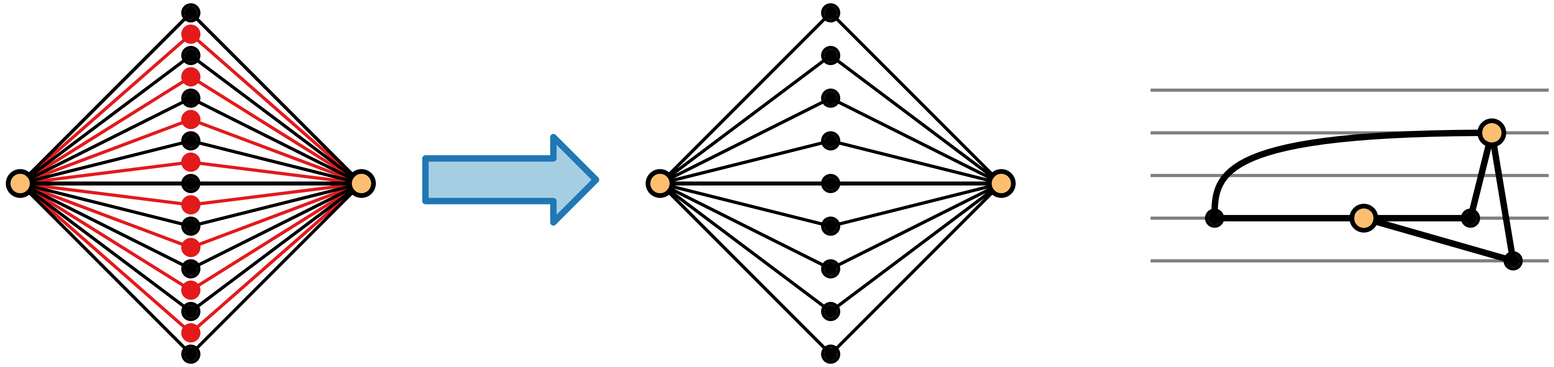
Vertex Cover: Reduction Rules

Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.



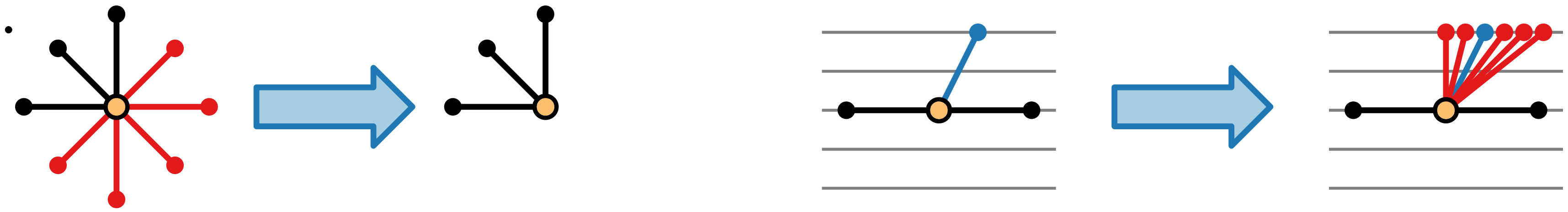
At most 2 neighbors on same level ➡ insert all other next to third neighbor

Rule 2: If any two vertices $c, d \in C$ have more than $4s + 5$ common degree-2 neighbors, remove all but $4s + 5$ of them.



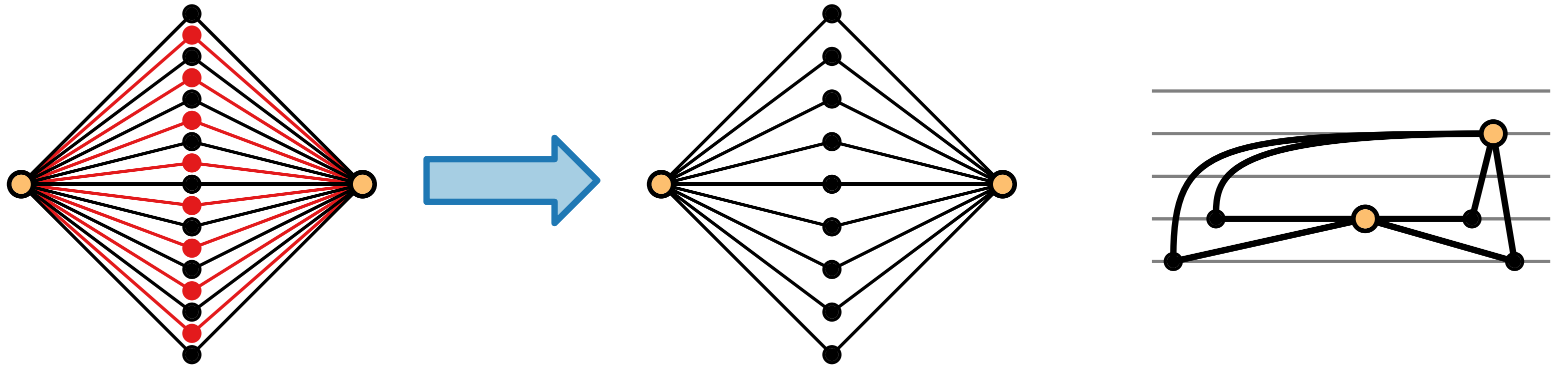
Vertex Cover: Reduction Rules

Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.



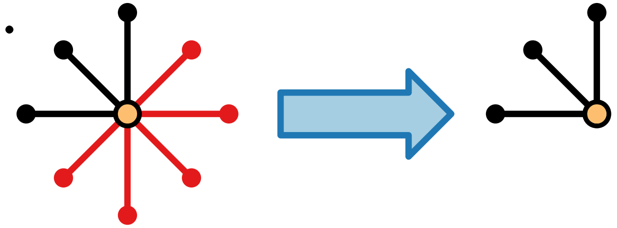
At most 2 neighbors on same level ➡ insert all other next to third neighbor

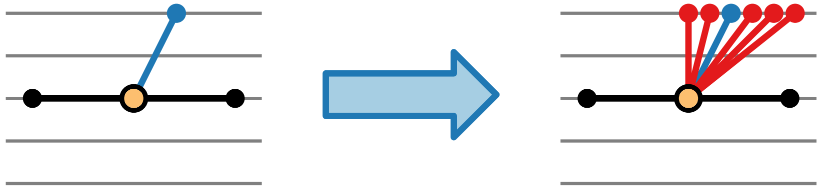
Rule 2: If any two vertices $c, d \in C$ have more than $4s + 5$ common degree-2 neighbors, remove all but $4s + 5$ of them.



Vertex Cover: Reduction Rules

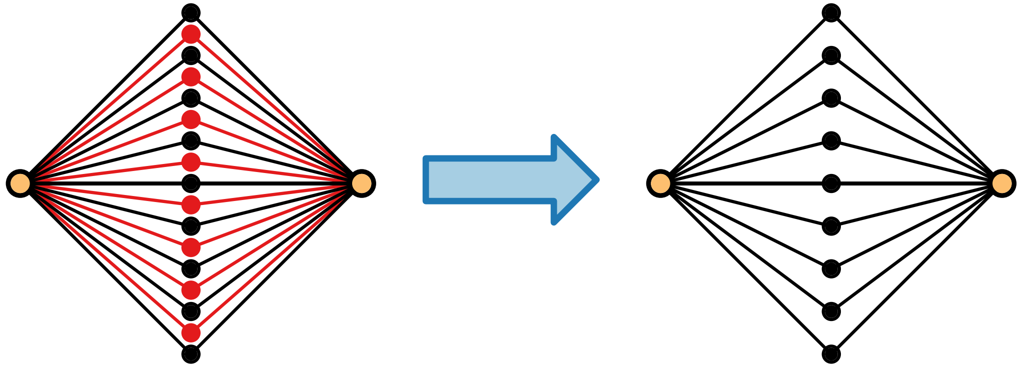
Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.

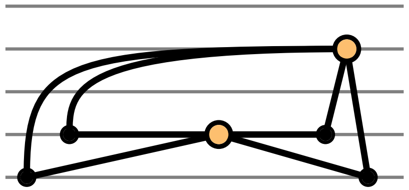




At most 2 neighbors on same level ➡ insert all other next to third neighbor

Rule 2: If any two vertices $c, d \in C$ have more than $4s + 5$ common degree-2 neighbors, remove all but $4s + 5$ of them.

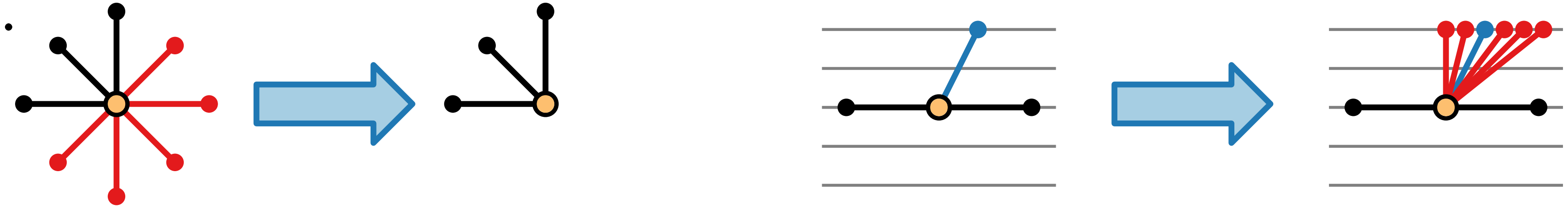




At most 2 neighbors on each level not between c and d

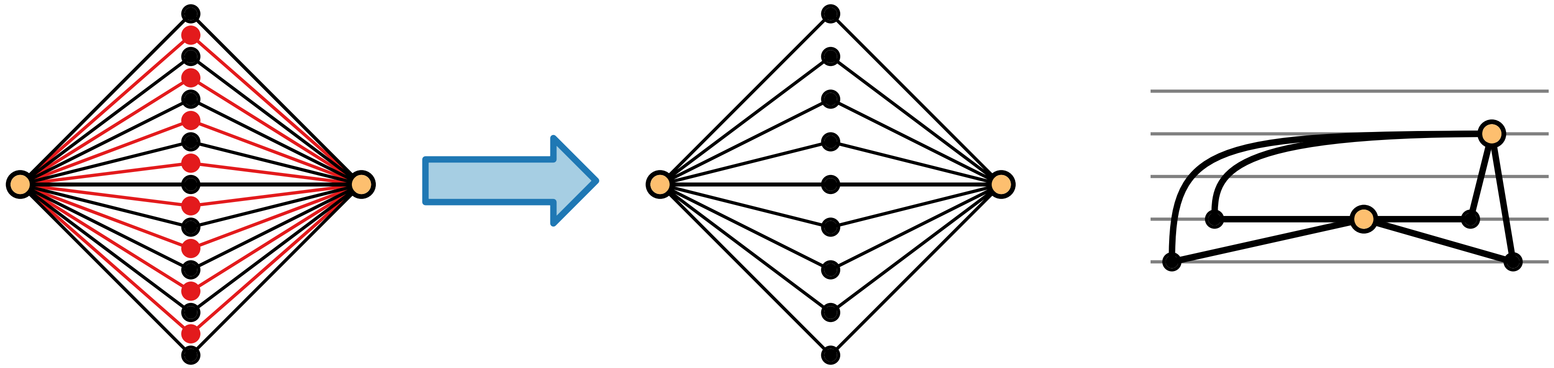
Vertex Cover: Reduction Rules

Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.



At most 2 neighbors on same level ➡ insert all other next to third neighbor

Rule 2: If any two vertices $c, d \in C$ have more than $4s + 5$ common degree-2 neighbors, remove all but $4s + 5$ of them.

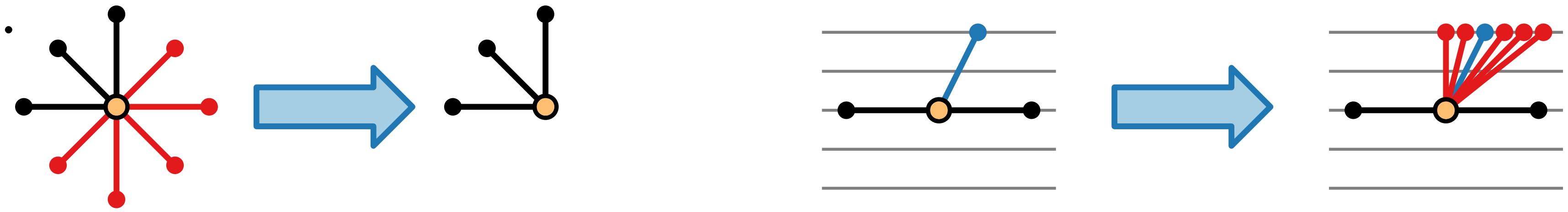


At most 2 neighbors on each level not between c and d

At most $2s + 2$ levels within span s of c and d

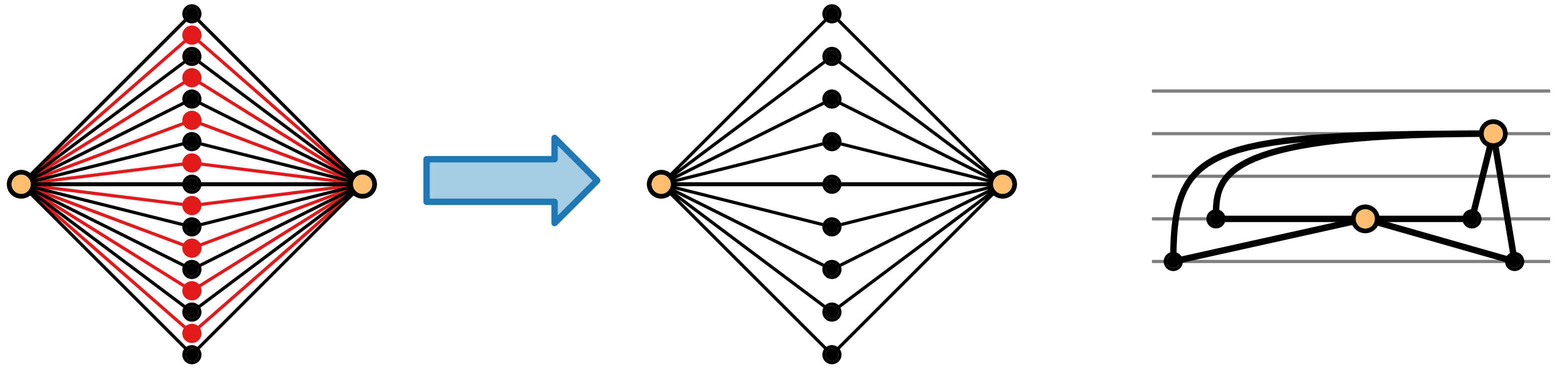
Vertex Cover: Reduction Rules

Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.



At most 2 neighbors on same level ➡ insert all other next to third neighbor

Rule 2: If any two vertices $c, d \in C$ have more than $4s + 5$ common degree-2 neighbors, remove all but $4s + 5$ of them.



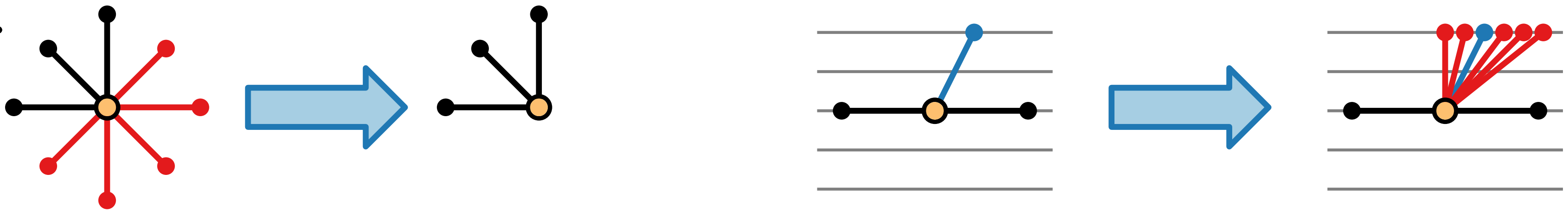
At most 2 neighbors on each level not between c and d

At most $2s + 2$ levels within span s of c and d

➡ at least one neighbor between c and d

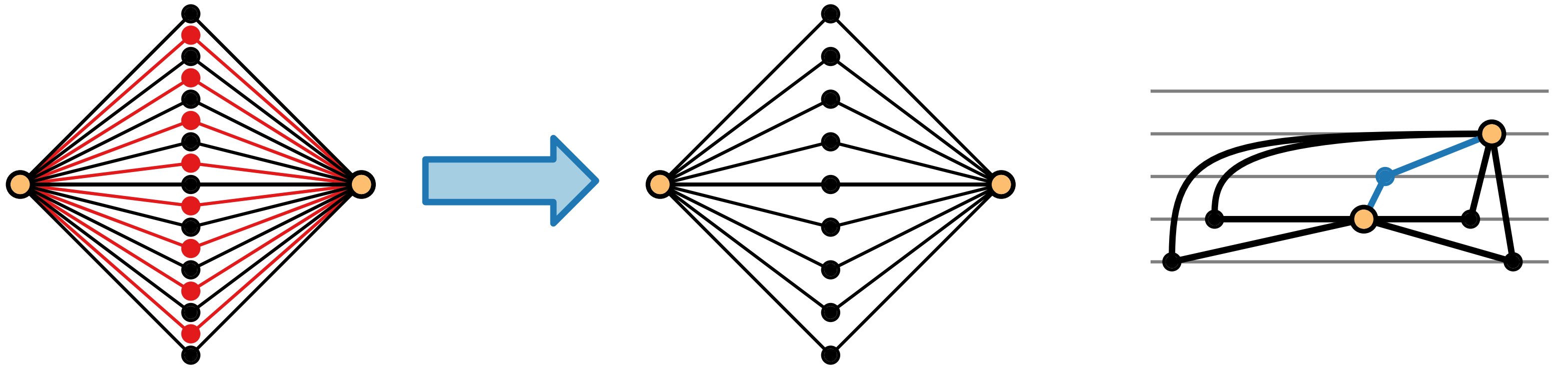
Vertex Cover: Reduction Rules

Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.



At most 2 neighbors on same level ➡ insert all other next to third neighbor

Rule 2: If any two vertices $c, d \in C$ have more than $4s + 5$ common degree-2 neighbors, remove all but $4s + 5$ of them.



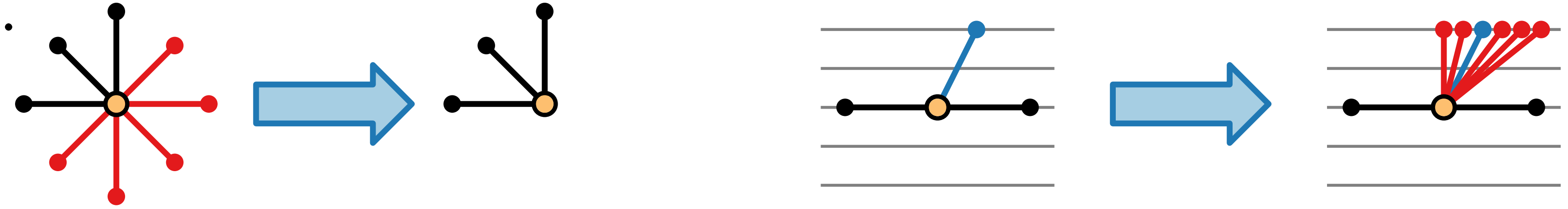
At most 2 neighbors on each level not between c and d

At most $2s + 2$ levels within span s of c and d

➡ at least one neighbor between c and d

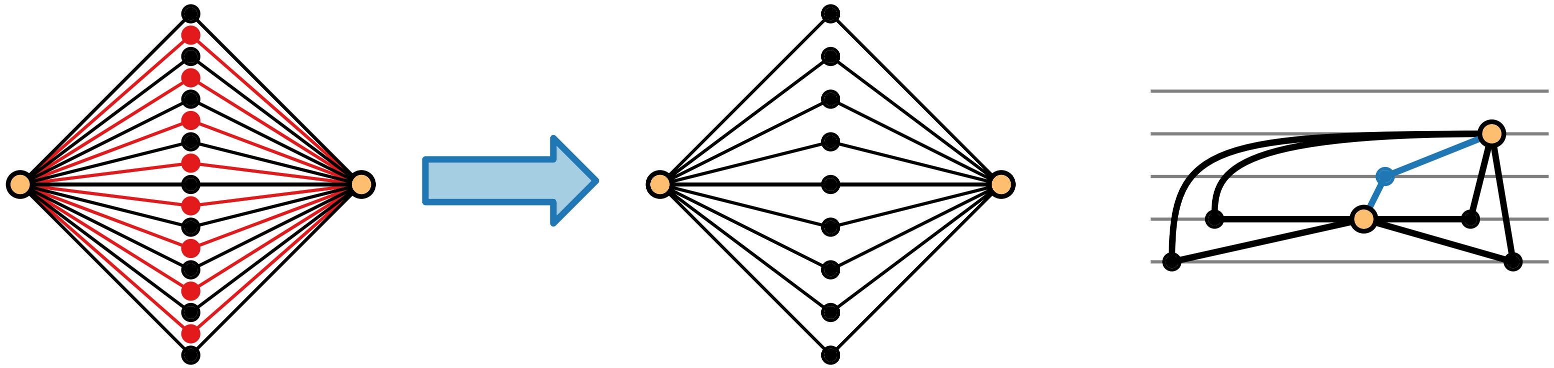
Vertex Cover: Reduction Rules

Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.



At most 2 neighbors on same level ➡ insert all other next to third neighbor

Rule 2: If any two vertices $c, d \in C$ have more than $4s + 5$ common degree-2 neighbors, remove all but $4s + 5$ of them.



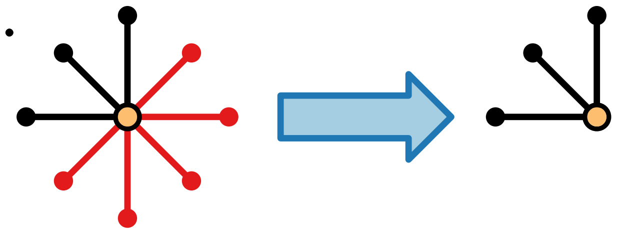
At most 2 neighbors on each level not between c and d

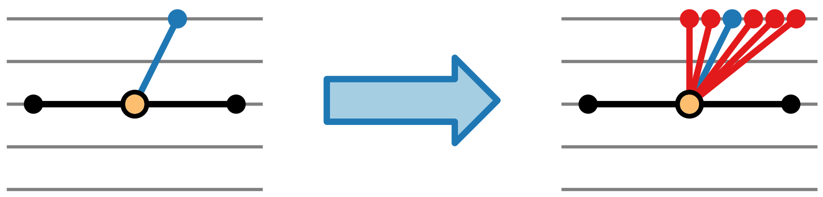
At most $2s + 2$ levels within span s of c and d

➡ at least one neighbor between c and d ➡ insert all others next to it

Vertex Cover: Reduction Rules

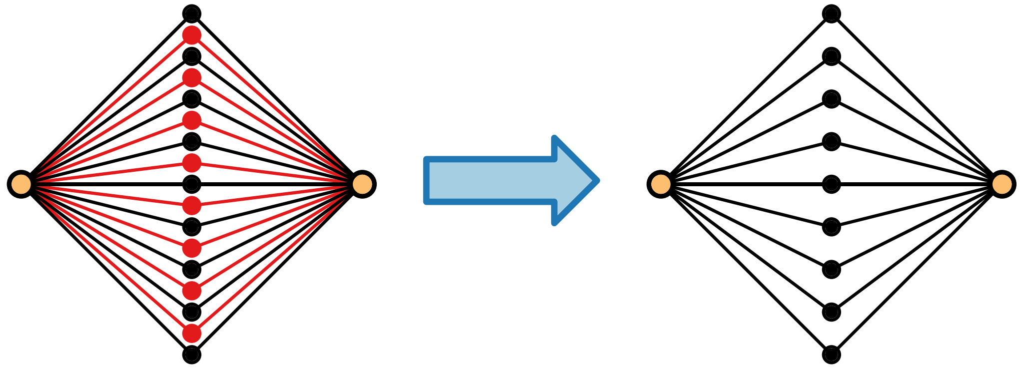
Rule 1: If any vertex $c \in C$ has more than 3 degree-1 neighbors in $V \setminus C$, remove all but 3 of them.

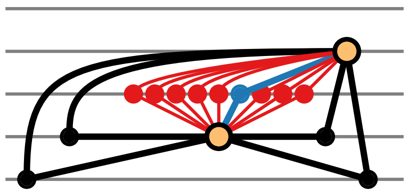




At most 2 neighbors on same level ➡ insert all other next to third neighbor

Rule 2: If any two vertices $c, d \in C$ have more than $4s + 5$ common degree-2 neighbors, remove all but $4s + 5$ of them.



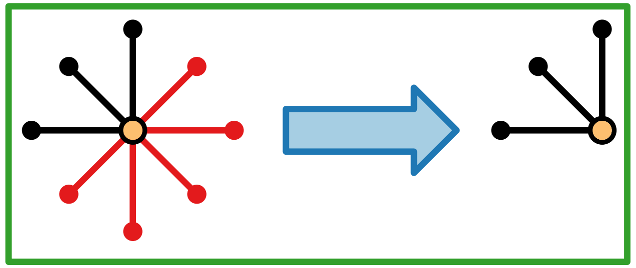


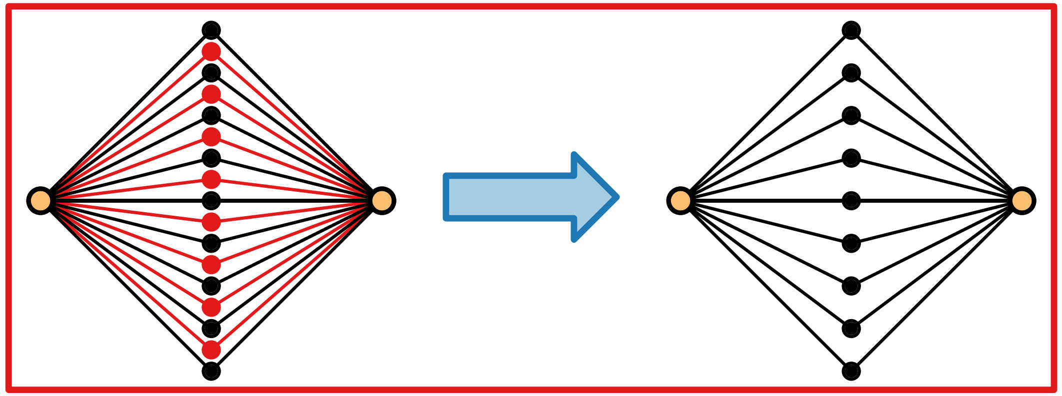
At most 2 neighbors on each level not between c and d

At most $2s + 2$ levels within span s of c and d

➡ at least one neighbor between c and d ➡ insert all others next to it

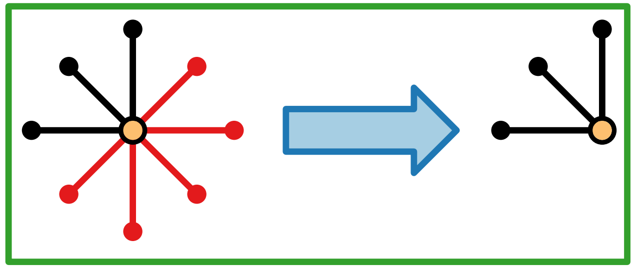
Vertex Cover: Result

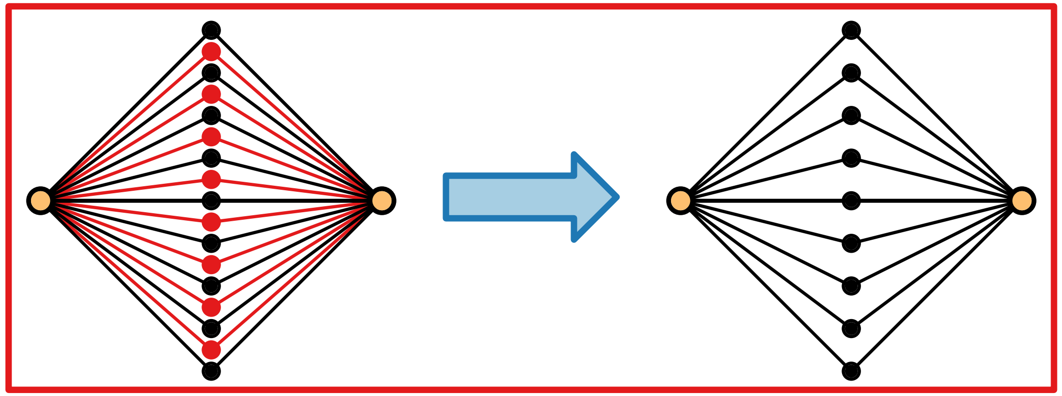




The reduced graph has $\mathcal{O}(k \cdot s)$ vertices.

Vertex Cover: Result

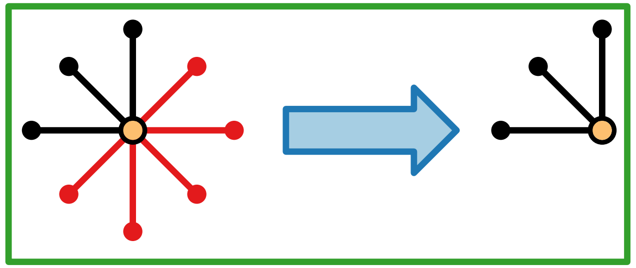


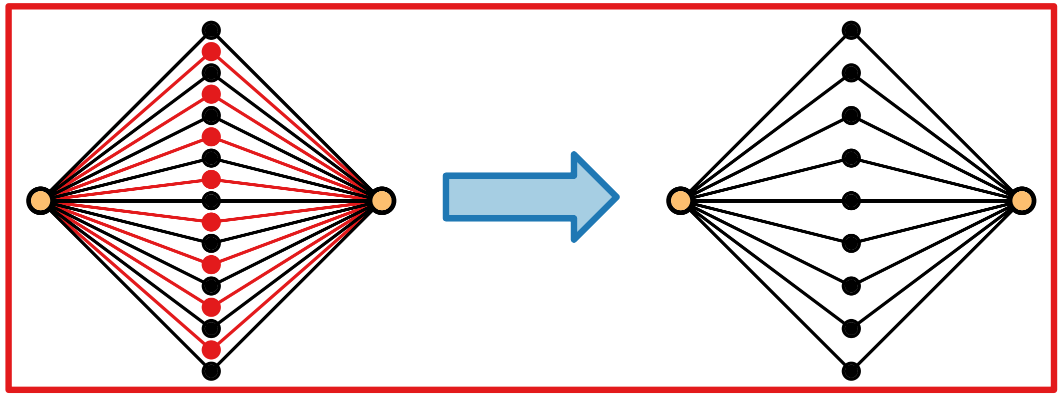


The reduced graph has $\mathcal{O}(k \cdot s)$ vertices.

Theorem.
s-Span Weakly Leveled Planarity for graphs with a vertex cover of size k admits a kernel of size $\mathcal{O}(k \cdot s)$.

Vertex Cover: Result



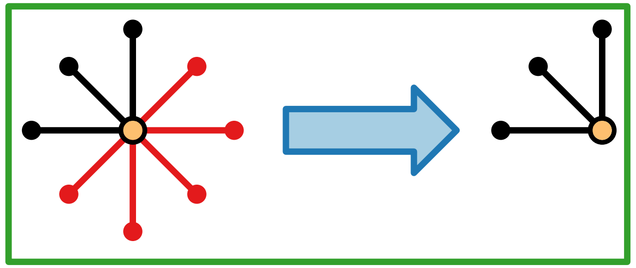


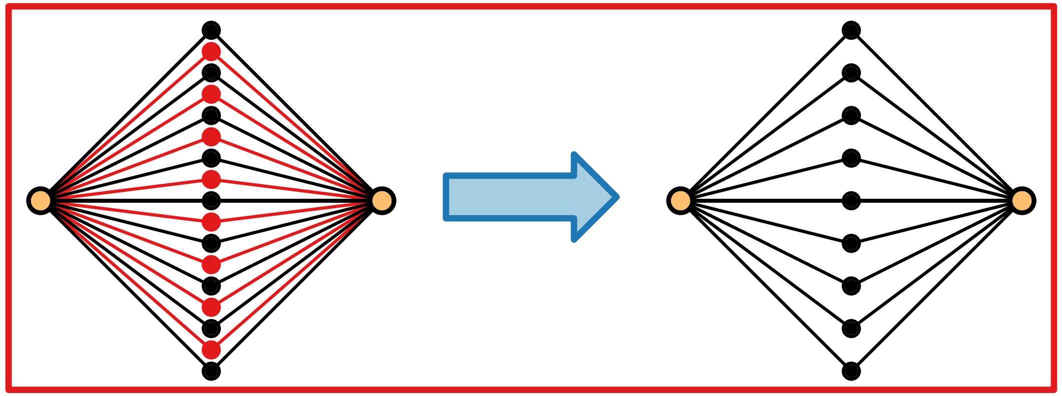
The reduced graph has $\mathcal{O}(k \cdot s)$ vertices.

Theorem.
s-Span Weakly Leveled Planarity for graphs with a vertex cover of size k admits a kernel of size $\mathcal{O}(k \cdot s)$.

Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

Vertex Cover: Result





The reduced graph has $\mathcal{O}(k \cdot s)$ vertices.

Theorem.
s-Span Weakly Leveled Planarity for graphs with a vertex cover of size k admits a kernel of size $\mathcal{O}(k \cdot s)$.

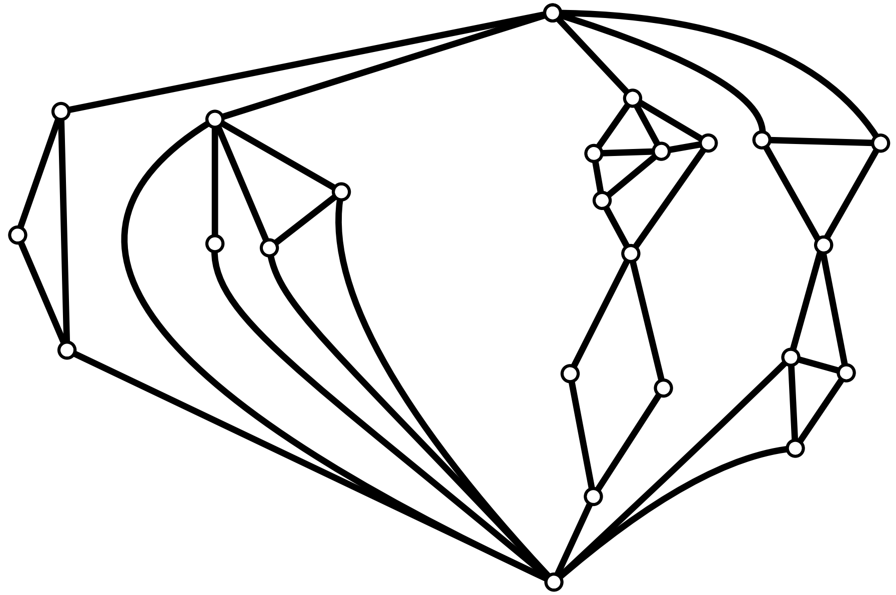
Lemma.
Every graph with a vertex cover of size k admits a $6k$-span (weakly) leveled planar drawing.

Theorem.
s-Span Weakly Leveled Planarity for graphs with a vertex cover of size k admits a kernel of size $\mathcal{O}(k^2)$. Hence, it is FPT with respect to the size of a vertex cover.

b-modulator

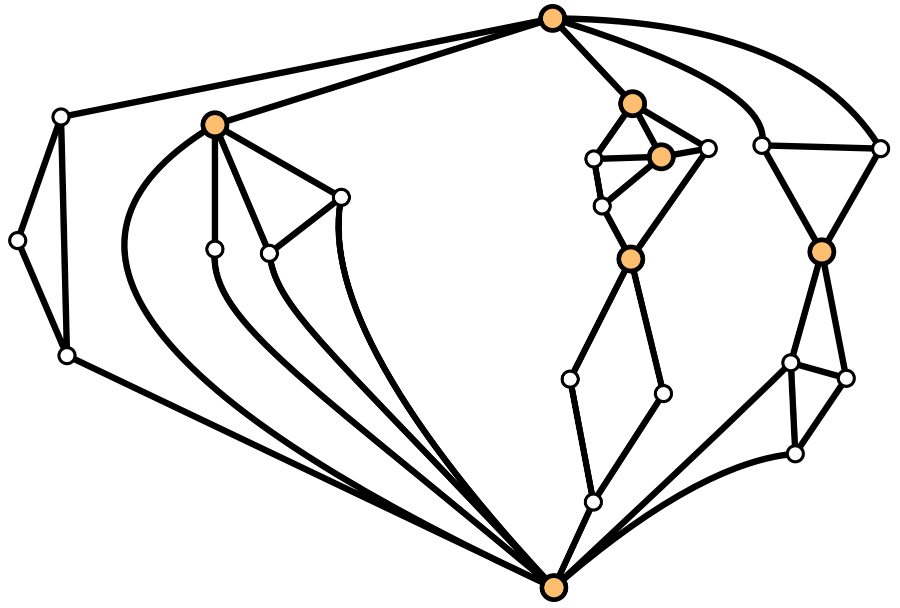
A ***b*-modulator** of a graph G is a set V' of vertices such that every connected component of $G - V'$ has size at most b.

b-modulator



A ***b*-modulator** of a graph G is a set V' of vertices such that every connected component of $G - V'$ has size at most b.

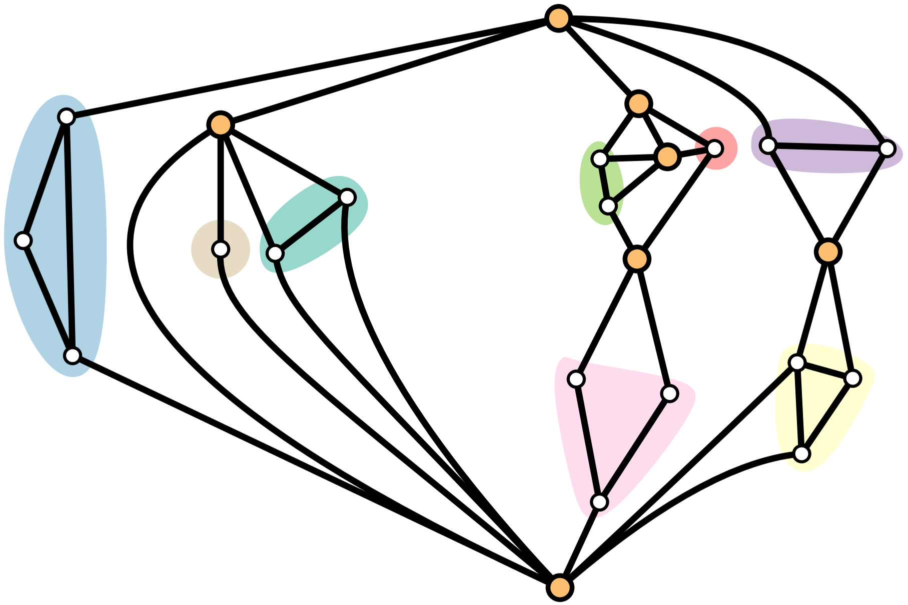
b-modulator



A **b-modulator** of a graph G is a set V' of vertices such that every connected component of $G - V'$ has size at most b.

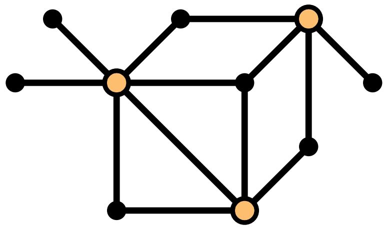
b-modulator

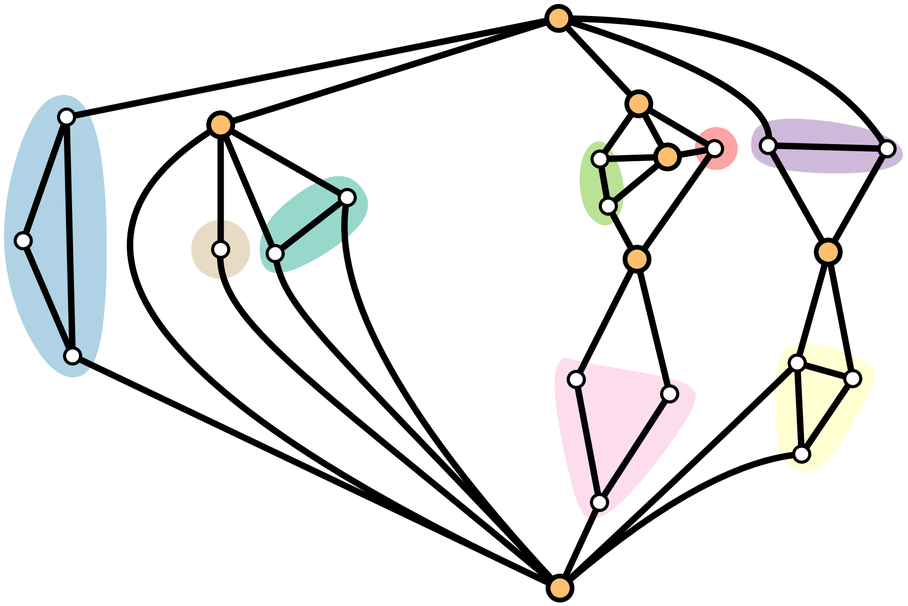
A **b-modulator** of a graph G is a set V' of vertices such that every connected component of $G - V'$ has size at most b.



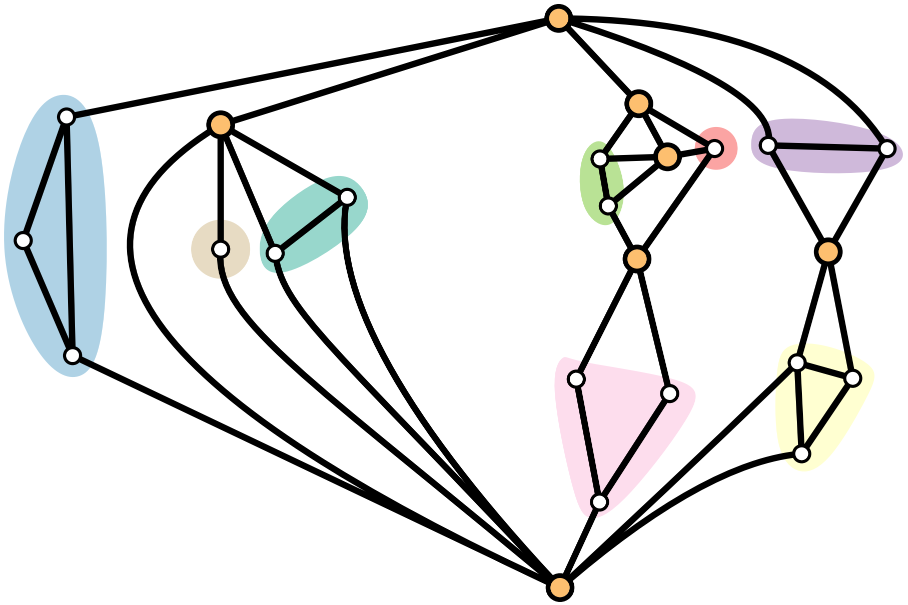
b-modulator

A **b-modulator** of a graph G is a set V' of vertices such that every connected component of $G - V'$ has size at most b.
A vertex cover is a 1-modulator.

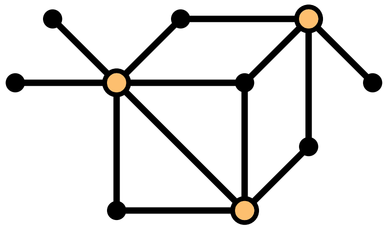




b-modulator

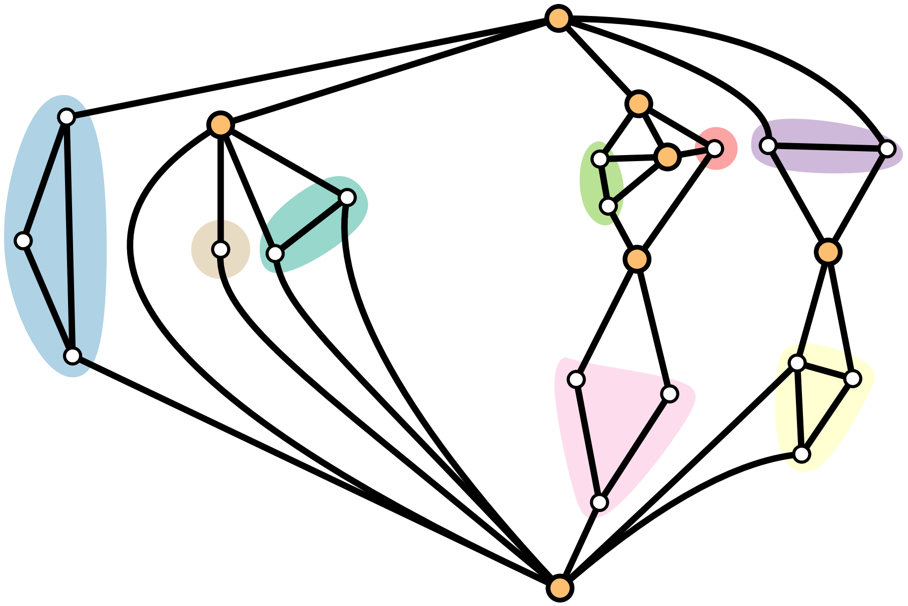


A **b-modulator** of a graph G is a set V' of vertices such that every connected component of $G - V'$ has size at most b.
A vertex cover is a 1-modulator.

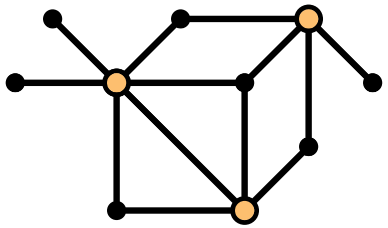


Theorem.
s-Span Weakly Leveled Planarity for graphs with a b-modulator of size k admits a kernel of size $\mathcal{O}(f(b) \cdot k \cdot s)$.

b-modulator



A **b-modulator** of a graph G is a set V' of vertices such that every connected component of $G - V'$ has size at most b.
A vertex cover is a 1-modulator.

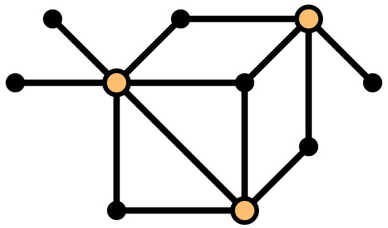


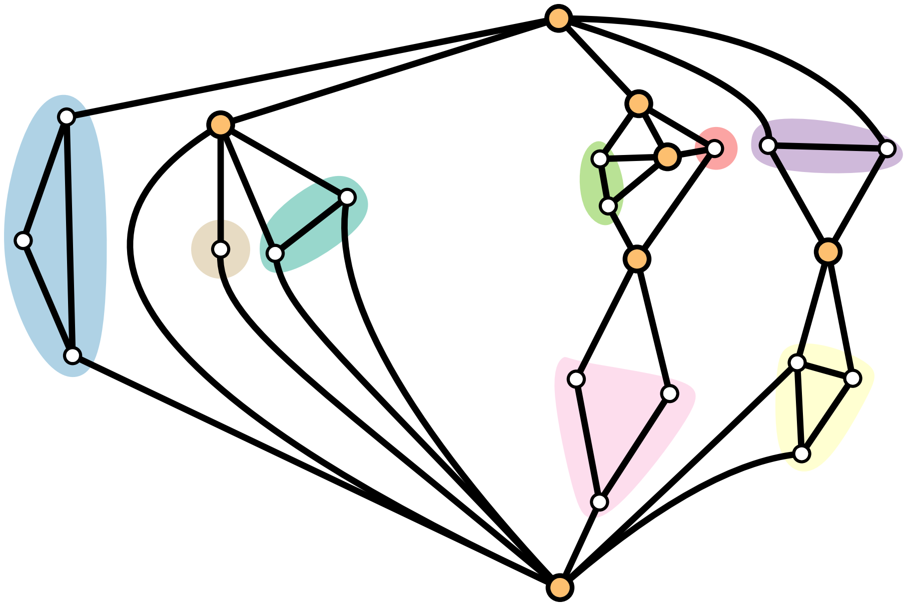
Theorem.
s-Span Weakly Leveled Planarity for graphs with a b-modulator of size k admits a kernel of size $\mathcal{O}(f(b) \cdot k \cdot s)$.

Lemma.
Every graph with a b-modulator of size k admits a $\mathcal{O}(b^2 \cdot k)$-span (weakly) leveled planar drawing.

b-modulator

A **b-modulator** of a graph G is a set V' of vertices such that every connected component of $G - V'$ has size at most b.
A vertex cover is a 1-modulator.





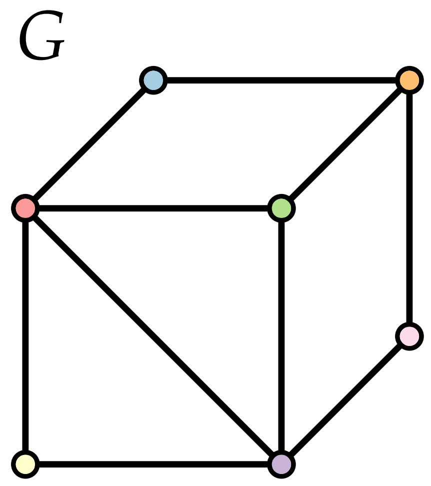
Theorem.
s-Span Weakly Leveled Planarity for graphs with a b-modulator of size k admits a kernel of size $\mathcal{O}(f(b) \cdot k \cdot s)$.

Lemma.
Every graph with a b-modulator of size k admits a $\mathcal{O}(b^2 \cdot k)$-span (weakly) leveled planar drawing.

Theorem.
s-Span Weakly Leveled Planarity for graphs with a b-modulator of size k admits a kernel of size $\mathcal{O}(f(b) \cdot k^2)$. Hence, it is FPT with respect to $k + b$.

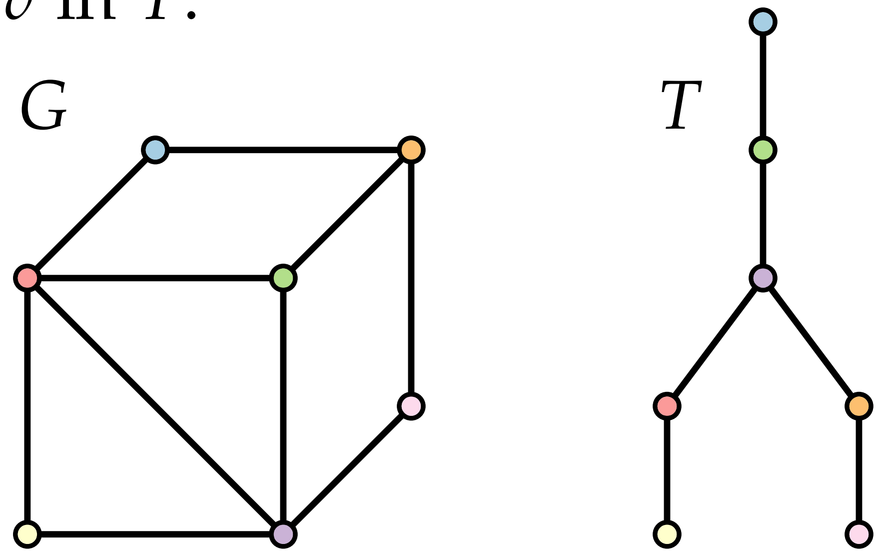
Treedepth

A **treedepth decomposition** of a graph G is a tree T on V s.t. for every edge uv there is an ancestor-descendant relationship between u and v in T.



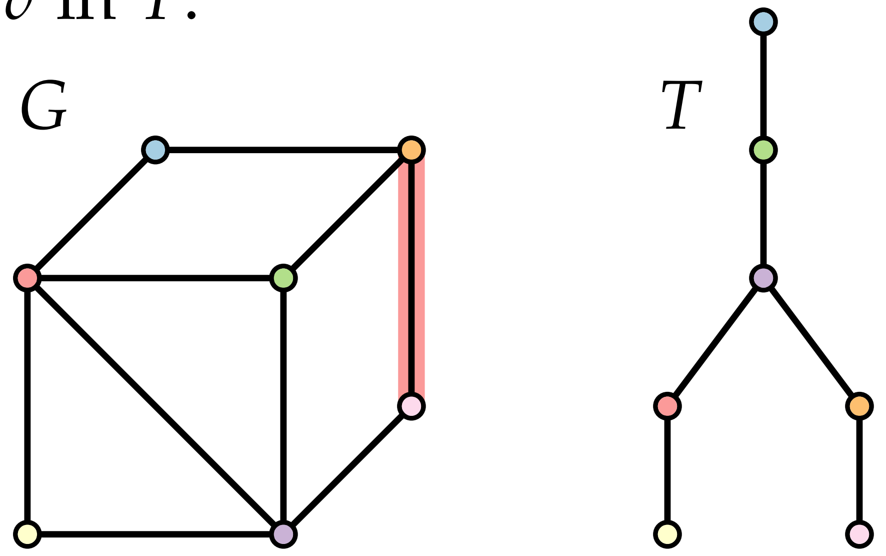
Treedepth

A **treedepth decomposition** of a graph G is a tree T on V s.t. for every edge uv there is an ancestor-descendant relationship between u and v in T.



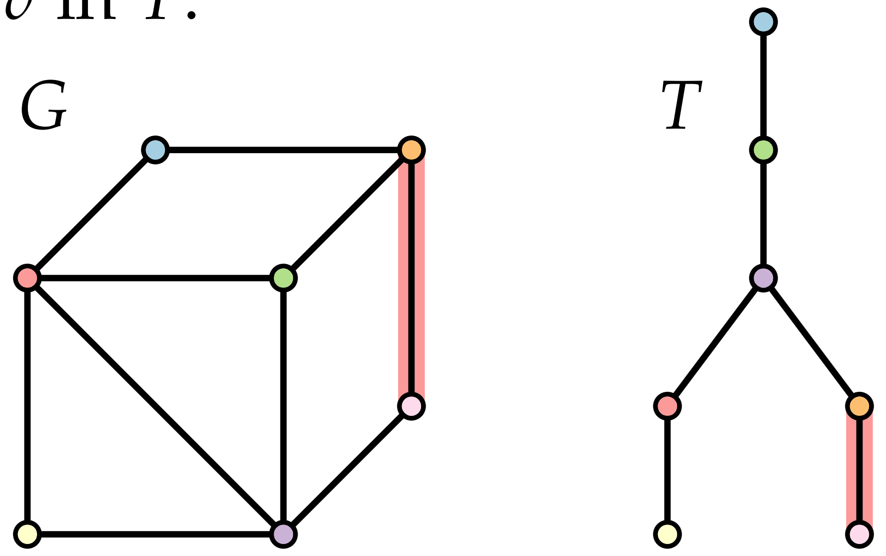
Treedepth

A **treedepth decomposition** of a graph G is a tree T on V s.t. for every edge uv there is an ancestor-descendant relationship between u and v in T.



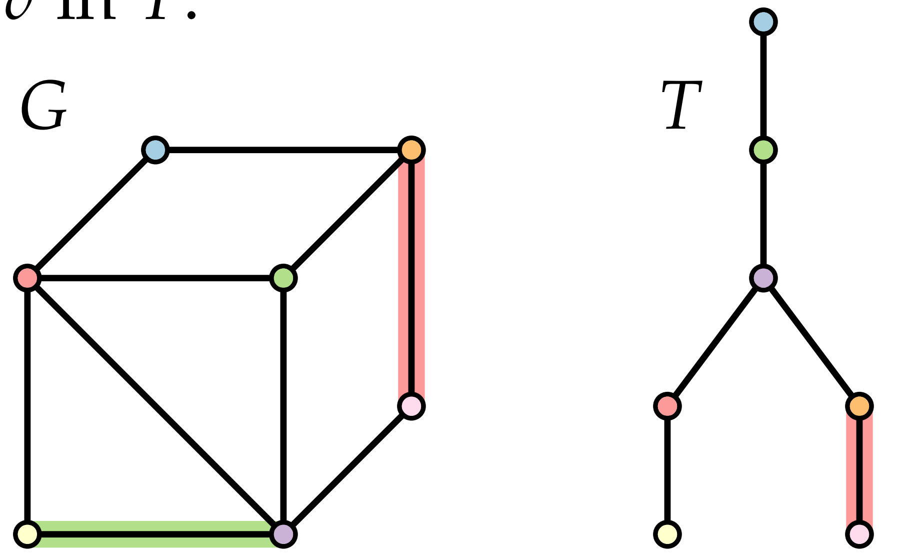
Treedepth

A **treedepth decomposition** of a graph G is a tree T on V s.t. for every edge uv there is an ancestor-descendant relationship between u and v in T.



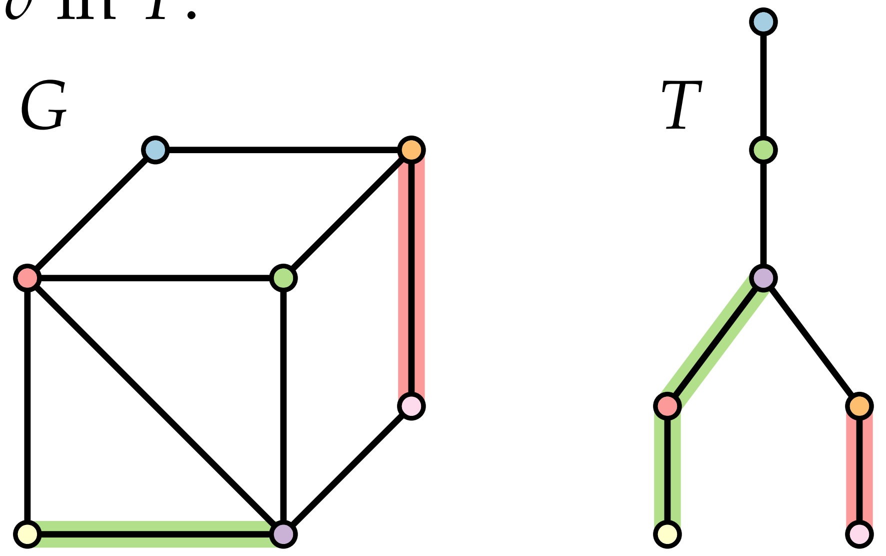
Treedepth

A **treedepth decomposition** of a graph G is a tree T on V s.t. for every edge uv there is an ancestor-descendant relationship between u and v in T.



Treedepth

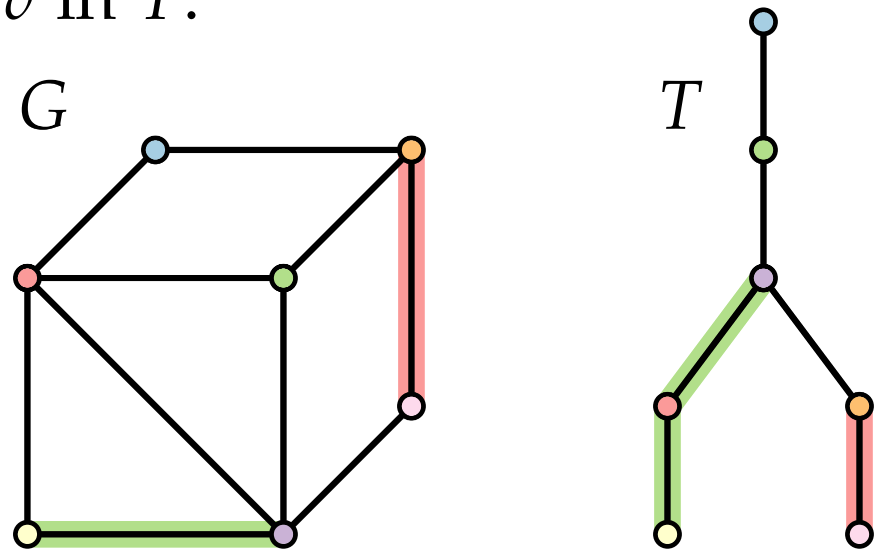
A **treedepth decomposition** of a graph G is a tree T on V s.t. for every edge uv there is an ancestor-descendant relationship between u and v in T.



Treedepth

A **treedepth decomposition** of a graph G is a tree T on V s.t. for every edge uv there is an ancestor-descendant relationship between u and v in T.

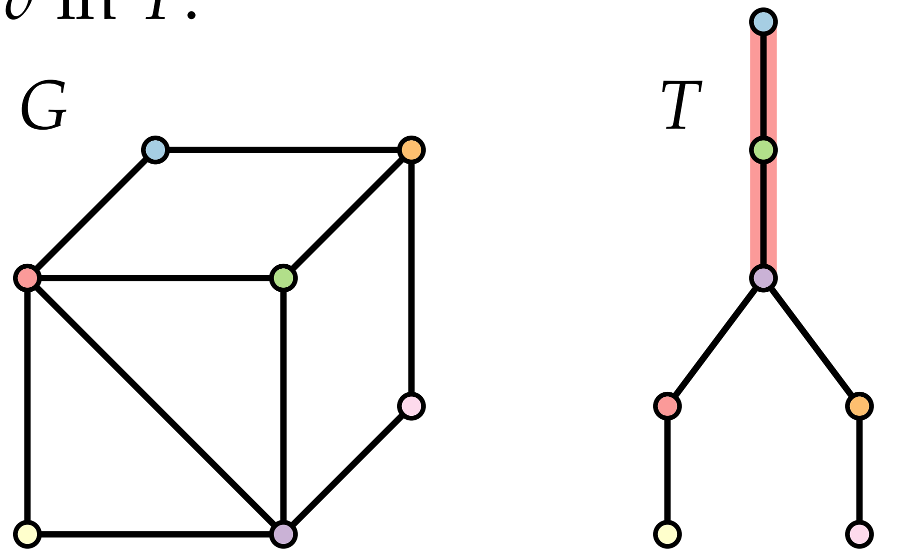
The **treedepth** of G is the minimum depth of a treedepth decomposition.



Treedepth

A **treedepth decomposition** of a graph G is a tree T on V s.t. for every edge uv there is an ancestor-descendant relationship between u and v in T.

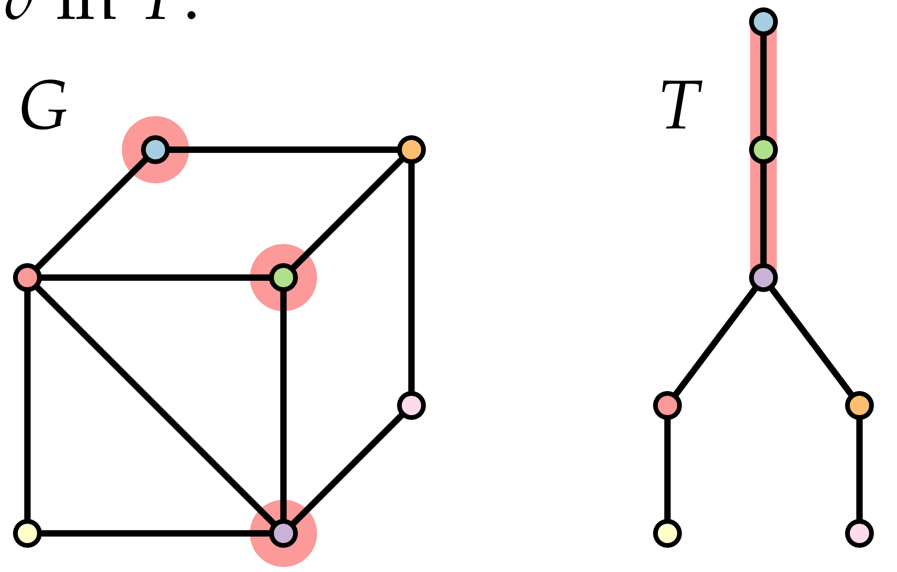
The **treedepth** of G is the minimum depth of a treedepth decomposition.



Treedepth

A **treedepth decomposition** of a graph G is a tree T on V s.t. for every edge uv there is an ancestor-descendant relationship between u and v in T.

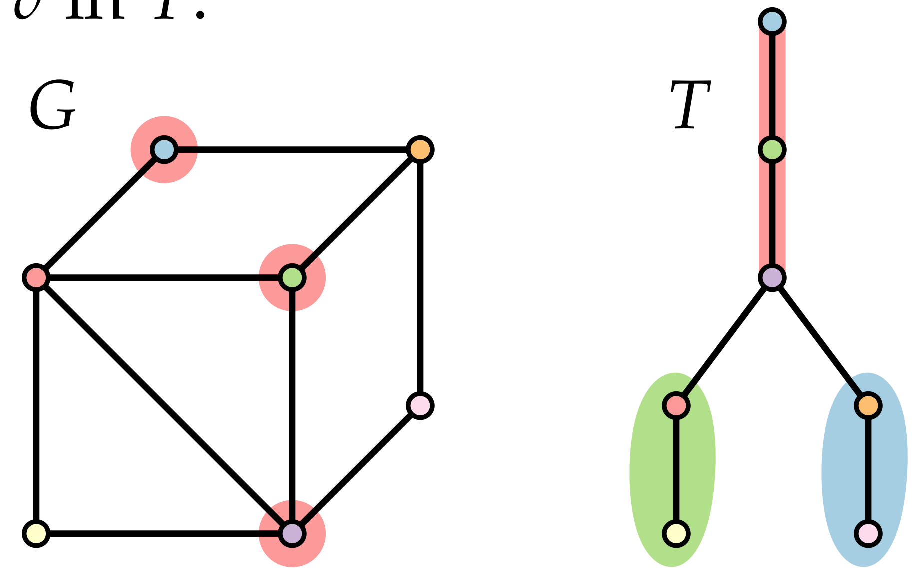
The **treedepth** of G is the minimum depth of a treedepth decomposition.



Treedepth

A **treedepth decomposition** of a graph G is a tree T on V s.t. for every edge uv there is an ancestor-descendant relationship between u and v in T.

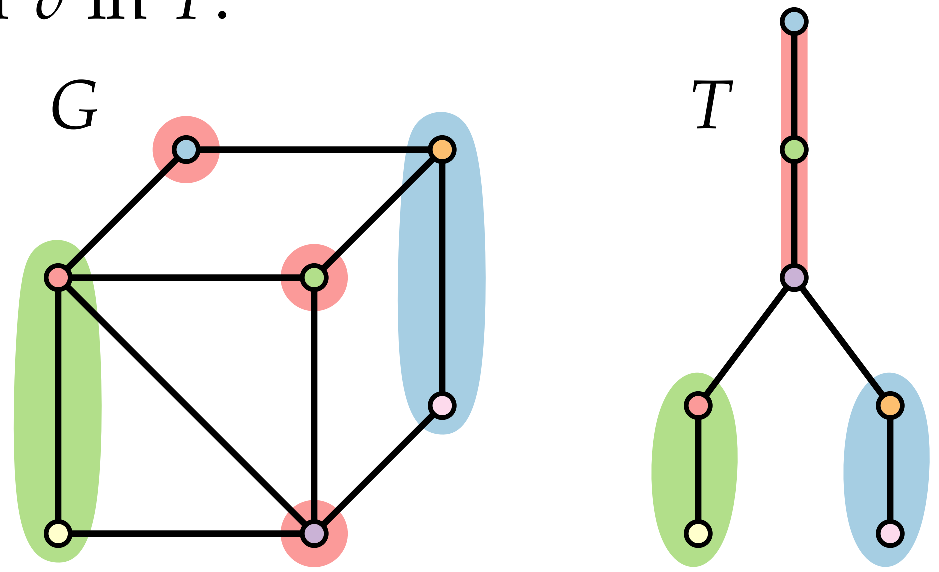
The **treedepth** of G is the minimum depth of a treedepth decomposition.



Treedepth

A **treedepth decomposition** of a graph G is a tree T on V s.t. for every edge uv there is an ancestor-descendant relationship between u and v in T.

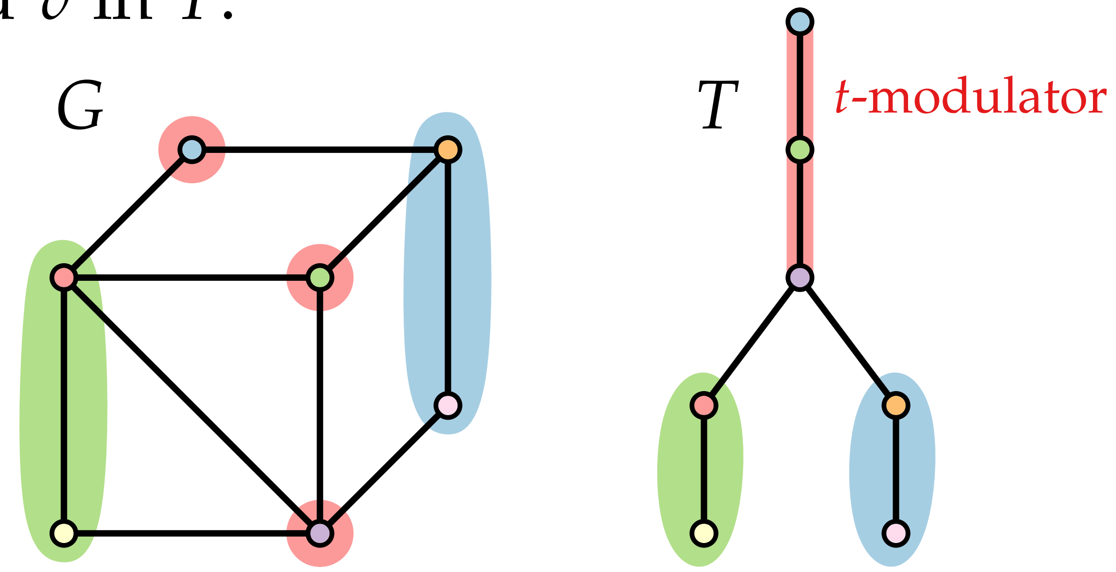
The **treedepth** of G is the minimum depth of a treedepth decomposition.



Treedepth

A **treedepth decomposition** of a graph G is a tree T on V s.t. for every edge uv there is an ancestor-descendant relationship between u and v in T.

The **treedepth** of G is the minimum depth of a treedepth decomposition.

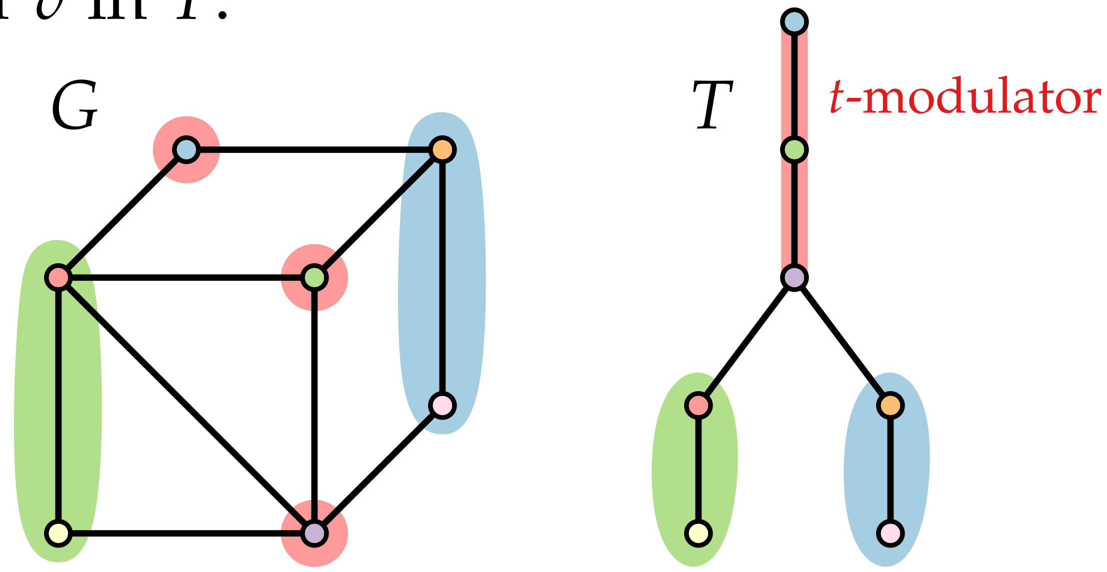


Trim until every vertex has outdegree $\leq 5t$.

Treedepth

A **treedepth decomposition** of a graph G is a tree T on V s.t. for every edge uv there is an ancestor-descendant relationship between u and v in T.

The **treedepth** of G is the minimum depth of a treedepth decomposition.



Trim until every vertex has outdegree $\leq 5t$.

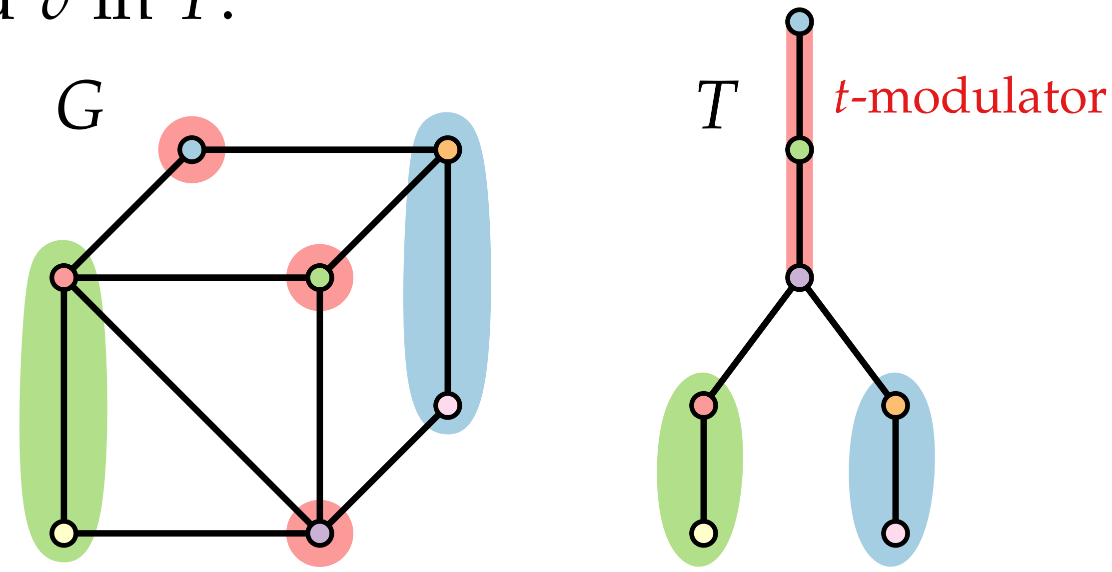
Lemma.
Every graph with treedepth t admits a $\mathcal{O}^*(t^{t^t})$-span (weakly) leveled planar drawing.

Treedepth

A **treedepth decomposition** of a graph G is a tree T on V s.t. for every edge uv there is an ancestor-descendant relationship between u and v in T.

The **treedepth** of G is the minimum depth of a treedepth decomposition.

Trim until every vertex has outdegree $\leq 5t$.

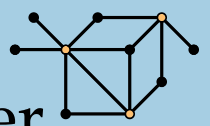


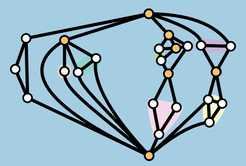
Lemma.
Every graph with treedepth t admits a $\mathcal{O}^*(t^{t^t})$-span (weakly) leveled planar drawing.

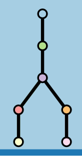
Theorem.
s-Span Weakly Leveled Planarity for graphs with treewidth t admits a kernel of size $\mathcal{O}(g(t))$. Hence, it is FPT with respect to t.

Overview

FPT algorithms

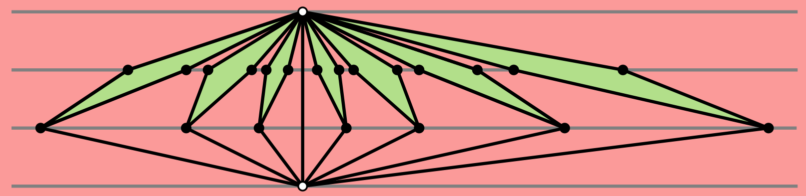
Linear kernel in size of vertex cover 

FPT in size of b-modulator $(+b)$ 

FPT in treedepth 

NP-hardness

Reduction from leveled planar

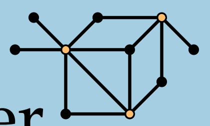


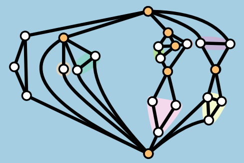
Combinatorial results

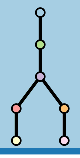
Implications

Overview

FPT algorithms

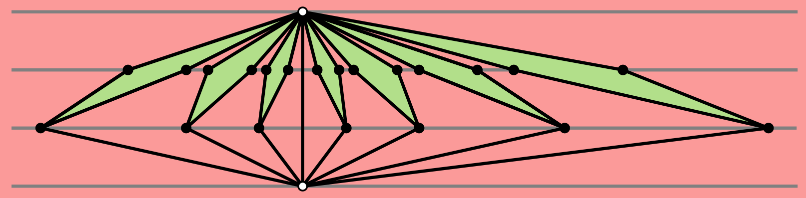
Linear kernel in size of vertex cover 

FPT in size of b-modulator $(+b)$ 

FPT in treedepth 

NP-hardness

Reduction from leveled planar



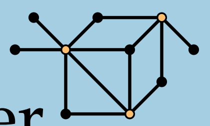
Combinatorial results

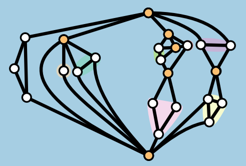
Graph class	**LB**	**UB**
2-outerplanar	$\Omega(n)$	n

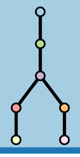
Implications

Overview

FPT algorithms

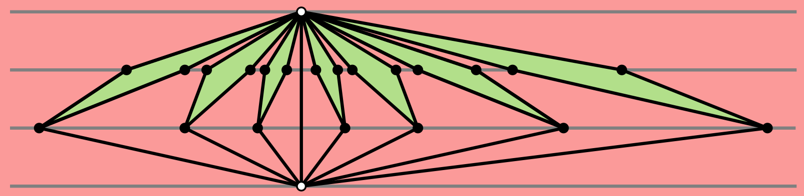
Linear kernel in size of vertex cover 

FPT in size of b-modulator $(+b)$ 

FPT in treedepth 

NP-hardness

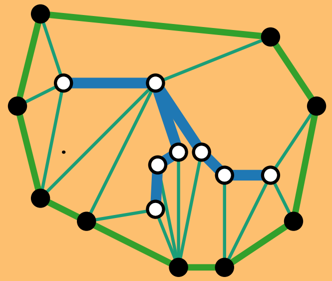
Reduction from leveled planar



Combinatorial results

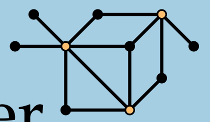
Graph class	LB	UB
2-outerplanar	$\Omega(n)$	n
3-connected cycle-tree	4	4

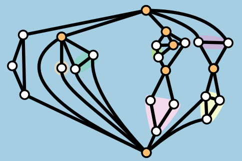
Implications

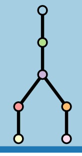


Overview

FPT algorithms

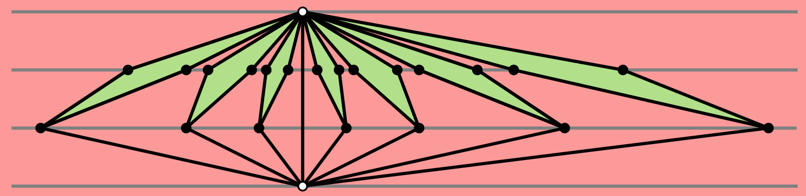
Linear kernel in size of vertex cover 

FPT in size of b-modulator $(+b)$ 

FPT in treedepth 

NP-hardness

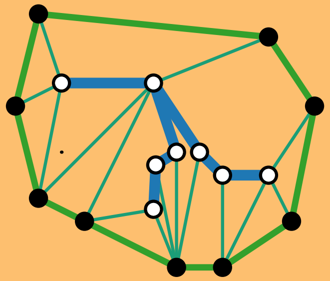
Reduction from leveled planar



Combinatorial results

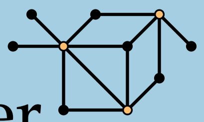
Graph class	**LB**	**UB**
2-outerplanar	$\Omega(n)$	n
3-connected cycle-tree	4	4
cycle-tree	$\Omega(\log n)$	$\mathcal{O}(\log n)$

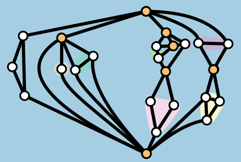
Implications

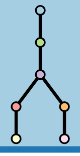


Overview

FPT algorithms

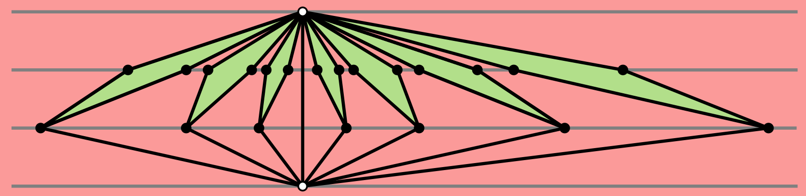
Linear kernel in size of vertex cover 

FPT in size of b-modulator $(+b)$ 

FPT in treedepth 

NP-hardness

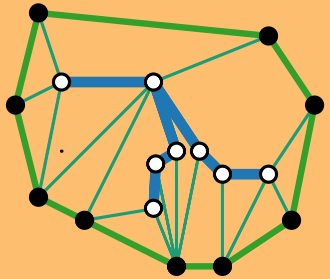
Reduction from leveled planar



Combinatorial results

Graph class	**LB**	**UB**
2-outerplanar	$\Omega(n)$	n
3-connected cycle-tree	4	4
cycle-tree	$\Omega(\log n)$	$\mathcal{O}(\log n)$
treewidth 2	$2^{\Omega(\sqrt{\log n})}$	$\mathcal{O}(\sqrt{n})$

Implications



Combinatoric Bounds

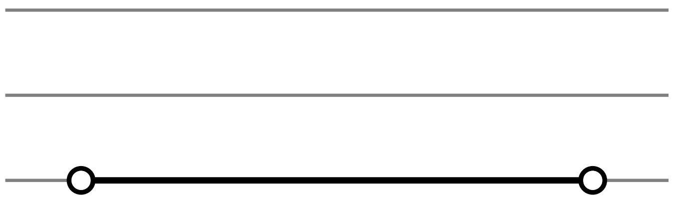
Theorem. [Felsner, Liotta & Wismath '01]
Every outerplanar graph admits a 1-span weakly leveled planar drawing.

Combinatoric Bounds

Theorem. [Felsner, Liotta & Wismath '01]
Every outerplanar graph admits a 1-span weakly leveled planar drawing.

Combinatoric Bounds

Theorem. [Felsner, Liotta & Wismath '01]
Every outerplanar graph admits a 1-span weakly leveled planar drawing.

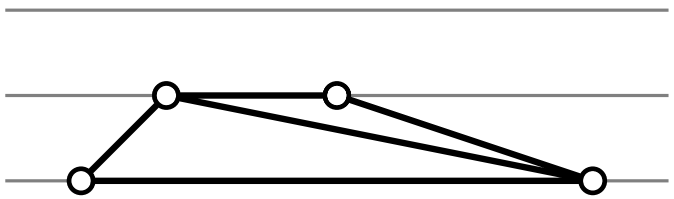


Combinatoric Bounds

Theorem. [Felsner, Liotta & Wismath '01]
Every outerplanar graph admits a 1-span weakly leveled planar drawing.

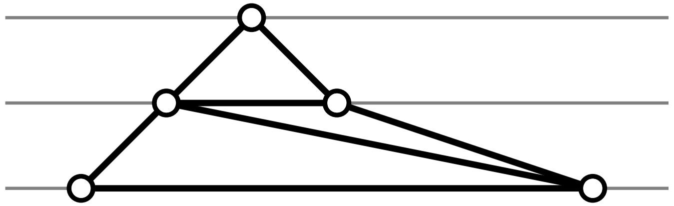
Combinatoric Bounds

Theorem. [Felsner, Liotta & Wismath '01]
Every outerplanar graph admits a 1-span weakly leveled planar drawing.



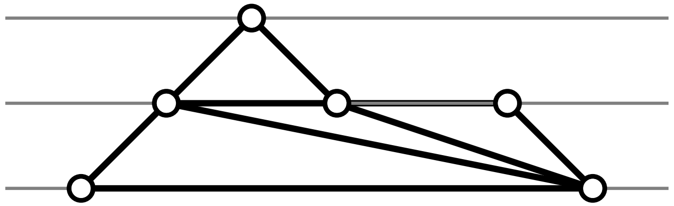
Combinatoric Bounds

Theorem. [Felsner, Liotta & Wismath '01]
Every outerplanar graph admits a 1-span weakly leveled planar drawing.



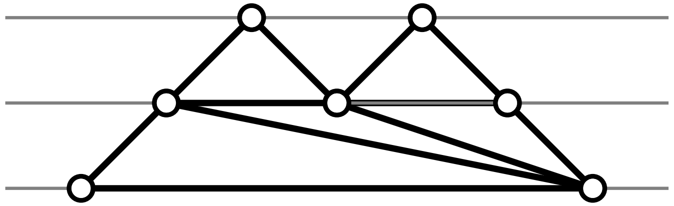
Combinatoric Bounds

Theorem. [Felsner, Liotta & Wismath '01]
Every outerplanar graph admits a 1-span weakly leveled planar drawing.



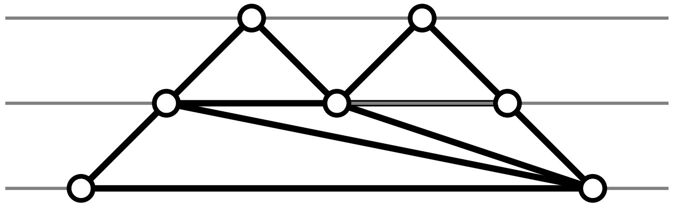
Combinatoric Bounds

Theorem. [Felsner, Liotta & Wismath '01]
Every outerplanar graph admits a 1-span weakly leveled planar drawing.



Combinatoric Bounds

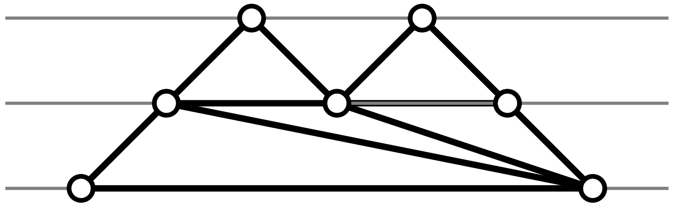
Theorem. [Felsner, Liotta & Wismath '01]
Every outerplanar graph admits a 1-span weakly leveled planar drawing.

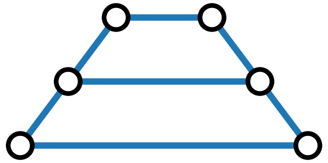


Theorem.
Some 2-outerplanar graphs require span $\Omega(n)$ in any weakly leveled planar drawing.

Combinatoric Bounds

Theorem. [Felsner, Liotta & Wismath '01]
Every outerplanar graph admits a 1-span weakly leveled planar drawing.

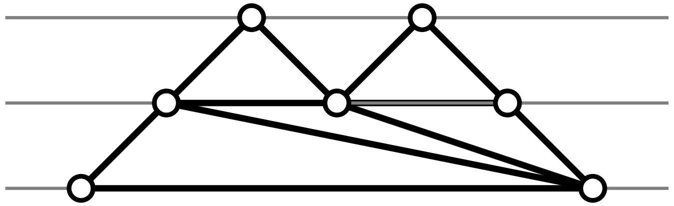


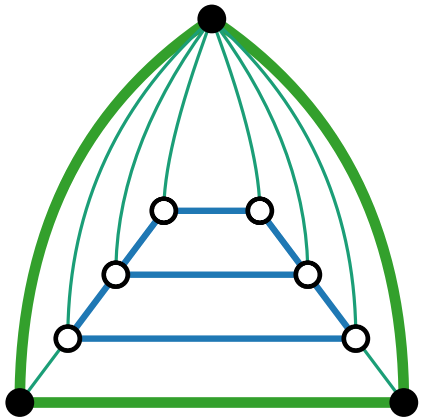


Theorem.
Some 2-outerplanar graphs require span $\Omega(n)$ in any weakly leveled planar drawing.

Combinatoric Bounds

Theorem. [Felsner, Liotta & Wismath '01]
Every outerplanar graph admits a 1-span weakly leveled planar drawing.

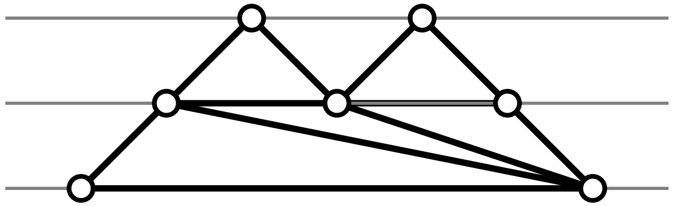




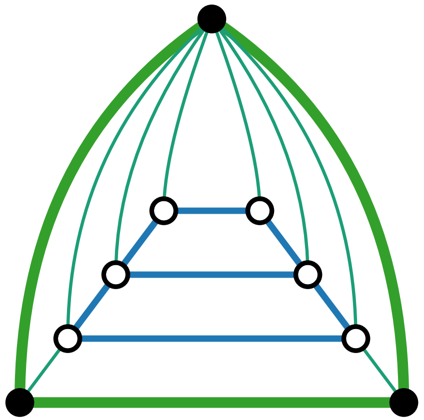
Theorem.
Some 2-outerplanar graphs require span $\Omega(n)$ in any weakly leveled planar drawing.

Combinatoric Bounds

Theorem. [Felsner, Liotta & Wismath '01]
Every outerplanar graph admits a 1-span weakly leveled planar drawing.



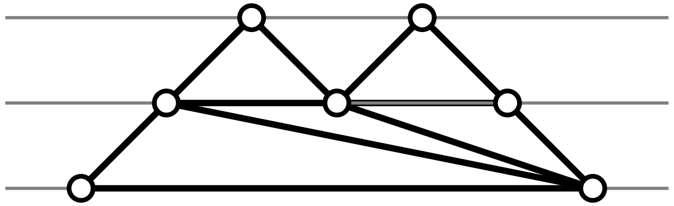
Theorem. [Di Giacomo et al. '24]
Every Halin graph admits a 1-span weakly leveled planar drawing.



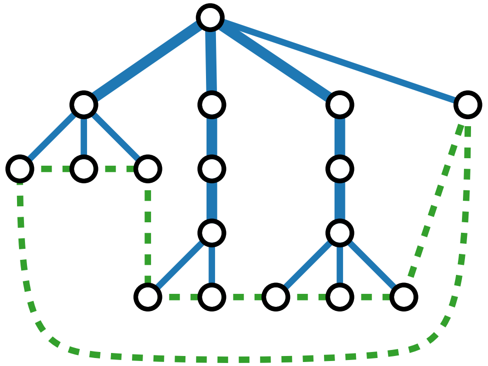
Theorem.
Some 2-outerplanar graphs require span $\Omega(n)$ in any weakly leveled planar drawing.

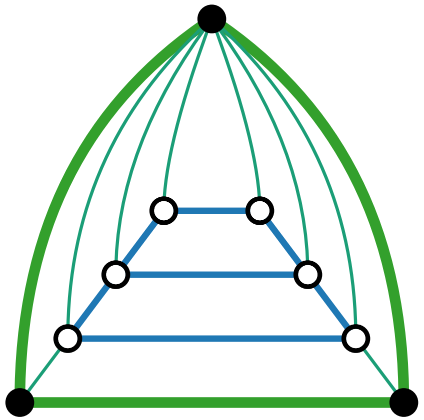
Combinatoric Bounds

Theorem. [Felsner, Liotta & Wismath '01]
Every outerplanar graph admits a 1-span weakly leveled planar drawing.



Theorem. [Di Giacomo et al. '24]
Every Halin graph admits a 1-span weakly leveled planar drawing.

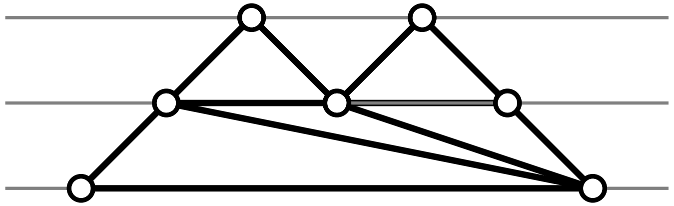




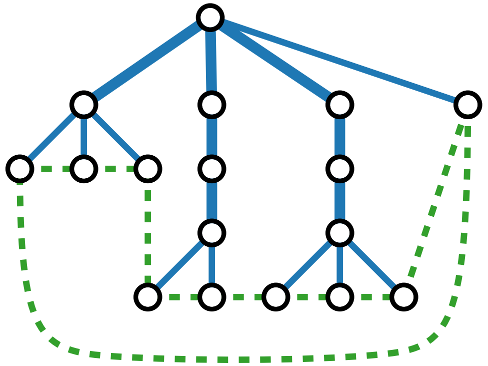
Theorem.
Some 2-outerplanar graphs require span $\Omega(n)$ in any weakly leveled planar drawing.

Combinatoric Bounds

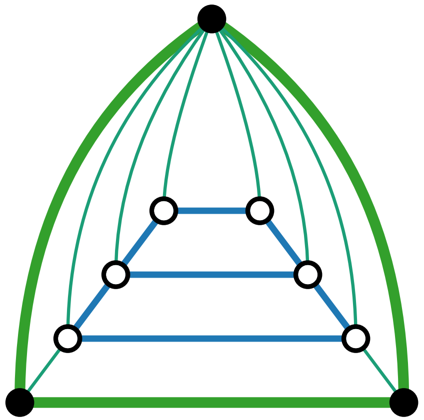
Theorem. [Felsner, Liotta & Wismath '01]
Every outerplanar graph admits a 1-span weakly leveled planar drawing.



Theorem. [Di Giacomo et al. '24]
Every Halin graph admits a 1-span weakly leveled planar drawing.



?



Theorem.
Some 2-outerplanar graphs require span $\Omega(n)$ in any weakly leveled planar drawing.

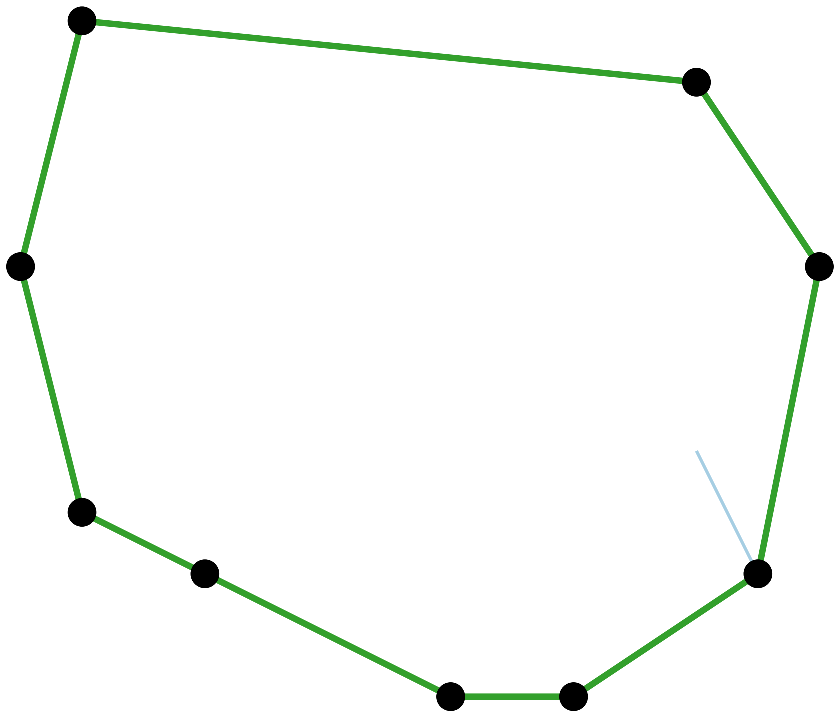
Cycle- and Path-trees

A **cycle-tree** consists of

Cycle- and Path-trees

A **cycle-tree** consists of

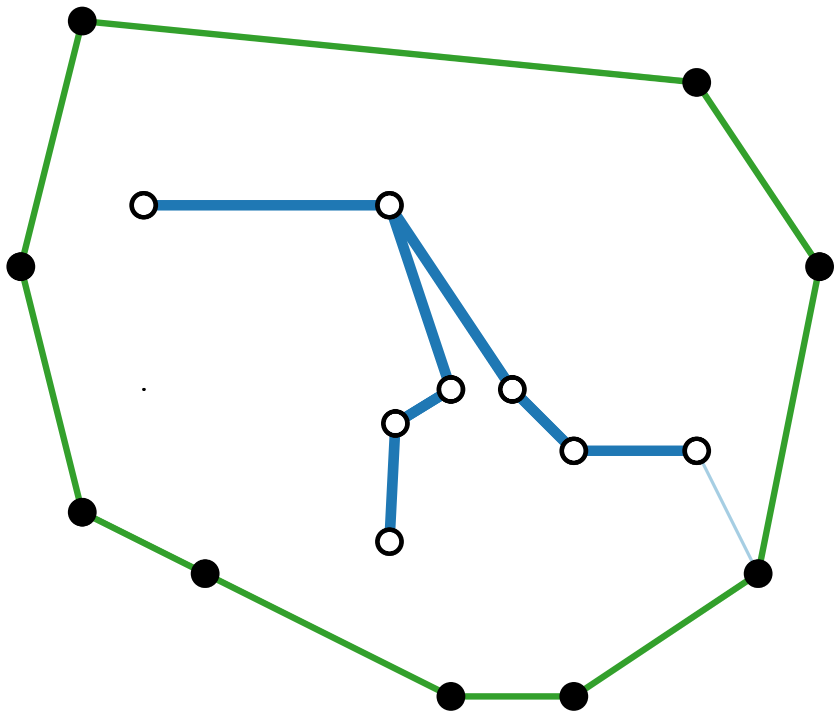
- A **cycle**



Cycle- and Path-trees

A **cycle-tree** consists of

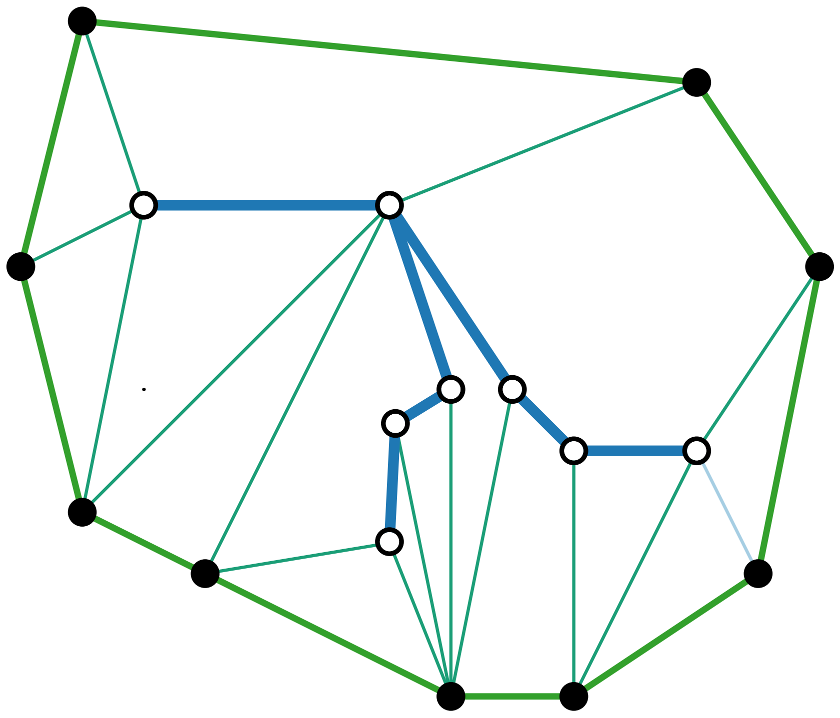
- A **cycle**
- A **tree** inside the cycle



Cycle- and Path-trees

A **cycle-tree** consists of

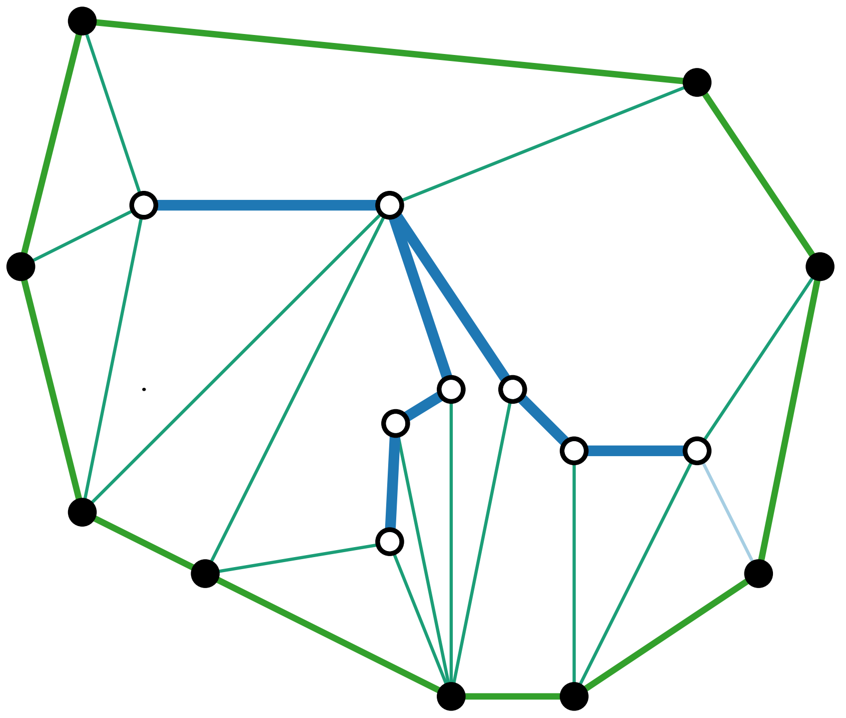
- A **cycle**
- A **tree** inside the cycle
- A **set of edges** between the cycle and the tree



Cycle- and Path-trees

A **cycle-tree** consists of

- A **cycle**
- A **tree** inside the cycle
- A **set of edges** between the cycle and the tree

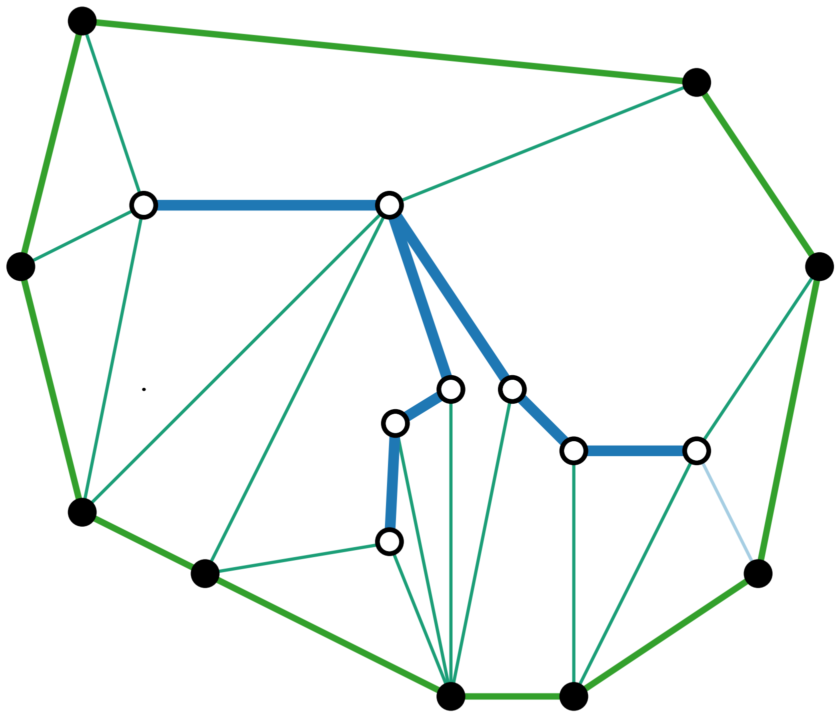


Every **Halin graph** is the subdivision of a 3-connected cycle-tree

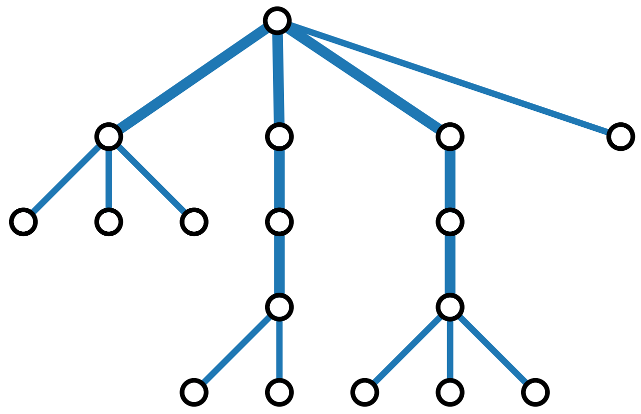
Cycle- and Path-trees

A **cycle-tree** consists of

- A **cycle**
- A **tree** inside the cycle
- A **set of edges** between the cycle and the tree



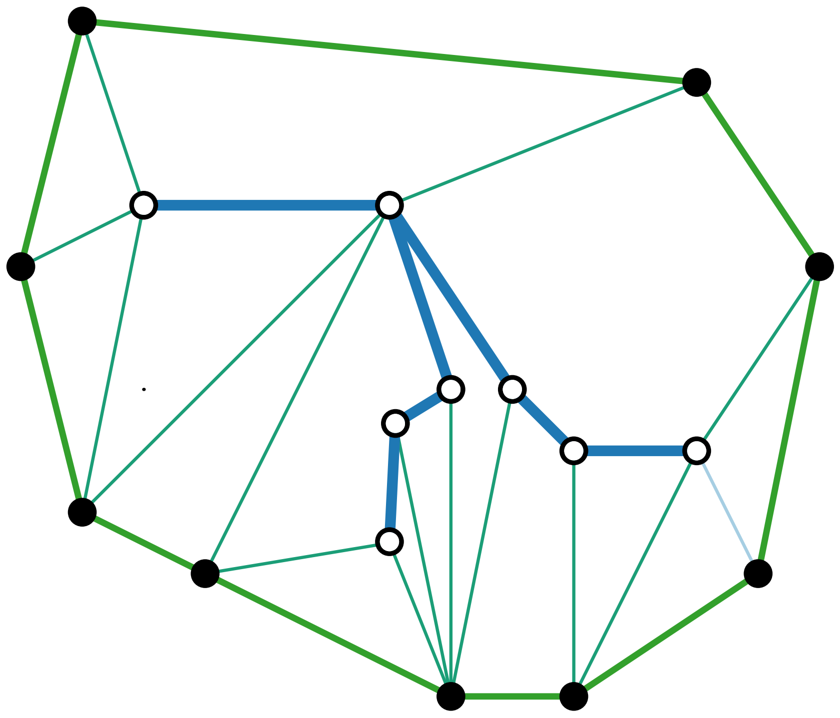
Every **Halin graph** is the subdivision of a 3-connected cycle-tree



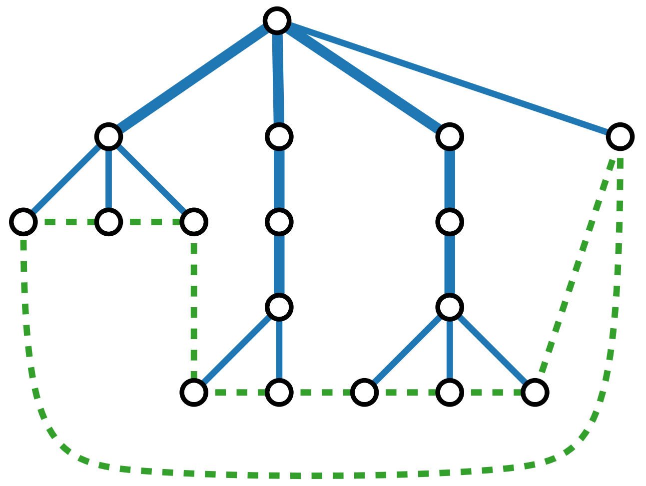
Cycle- and Path-trees

A **cycle-tree** consists of

- A **cycle**
- A **tree** inside the cycle
- A **set of edges** between the cycle and the tree



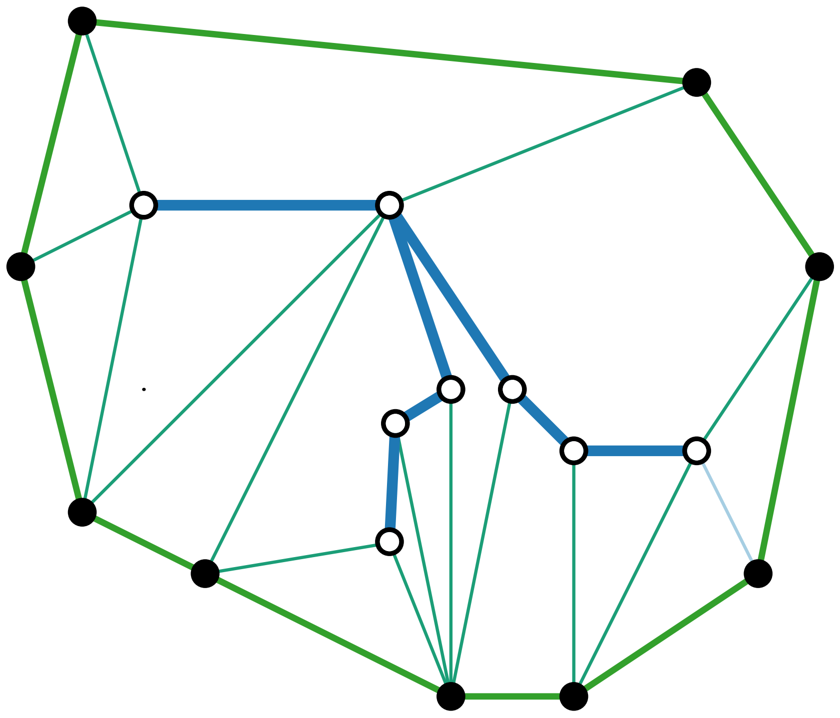
Every **Halin graph** is the subdivision of a 3-connected cycle-tree



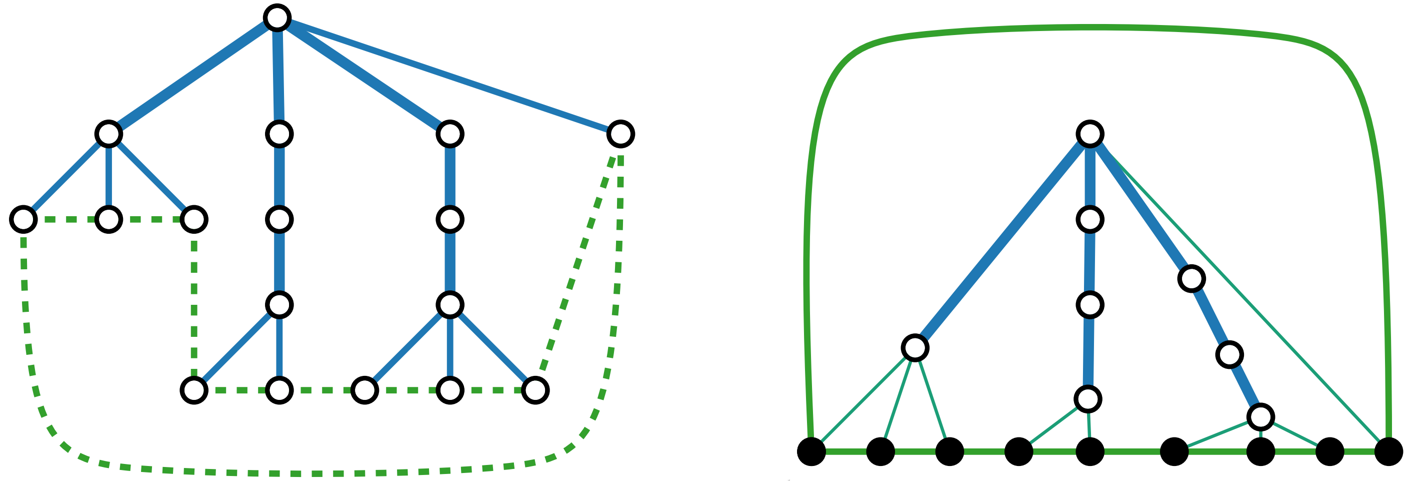
Cycle- and Path-trees

A **cycle-tree** consists of

- A **cycle**
- A **tree** inside the cycle
- A **set of edges** between the cycle and the tree



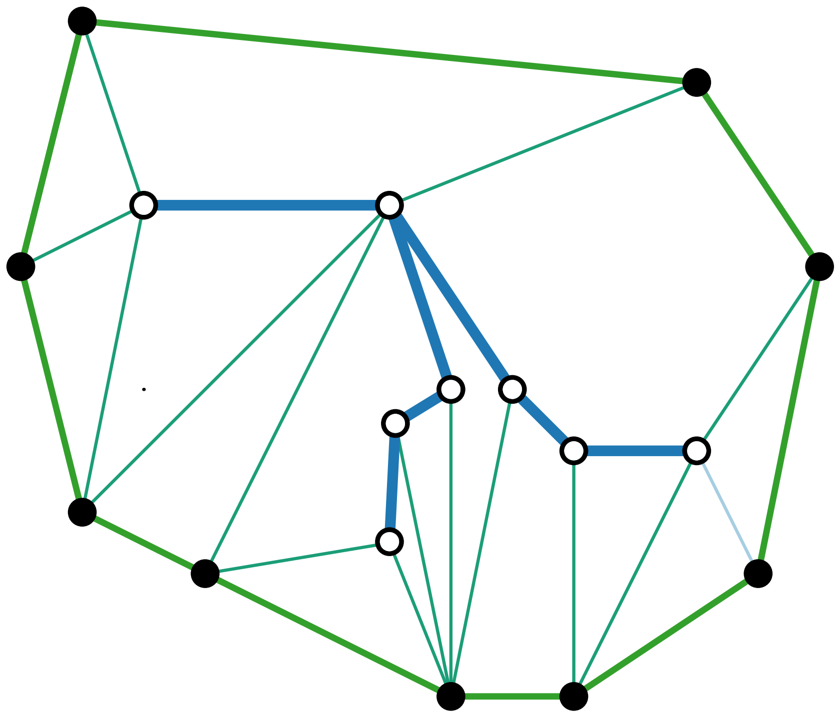
Every **Halin graph** is the subdivision of a 3-connected cycle-tree



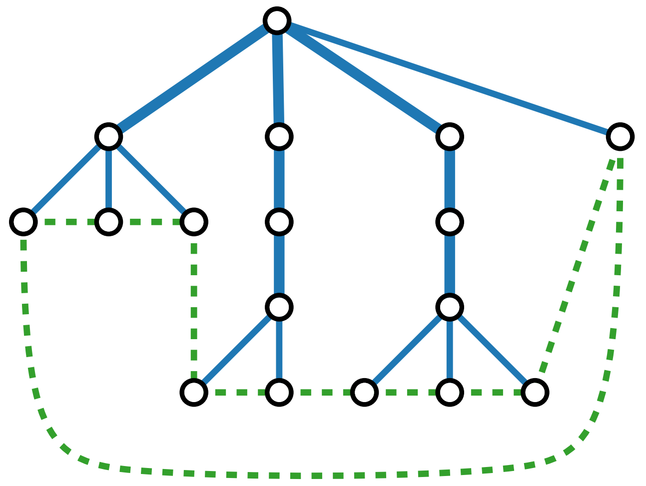
Cycle- and Path-trees

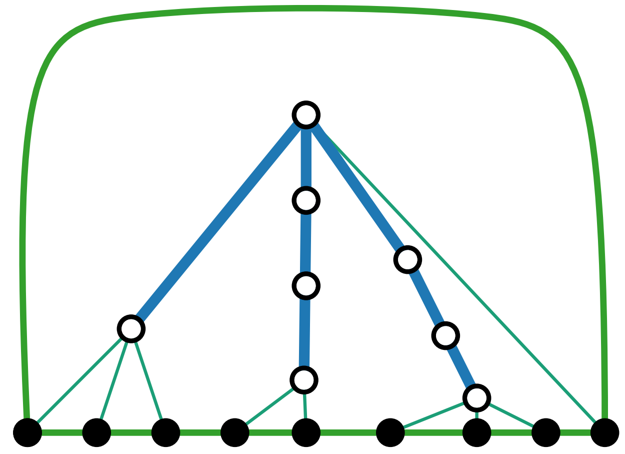
A **cycle-tree** consists of

- A **cycle**
- A **tree** inside the cycle
- A **set of edges** between the cycle and the tree



Every **Halin graph** is the subdivision of a 3-connected cycle-tree

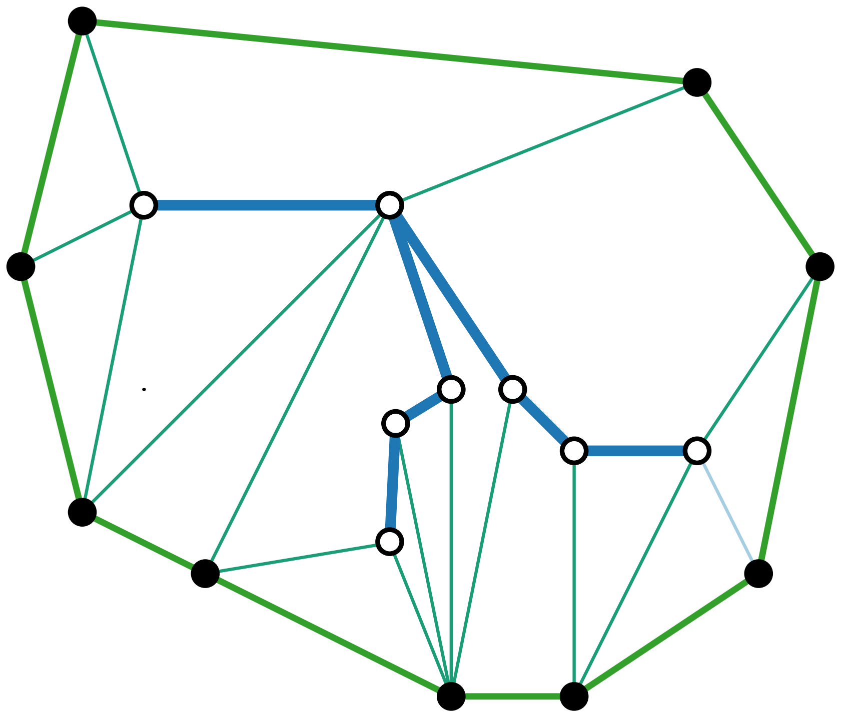




Cycle- and Path-trees

A **cycle-tree** consists of

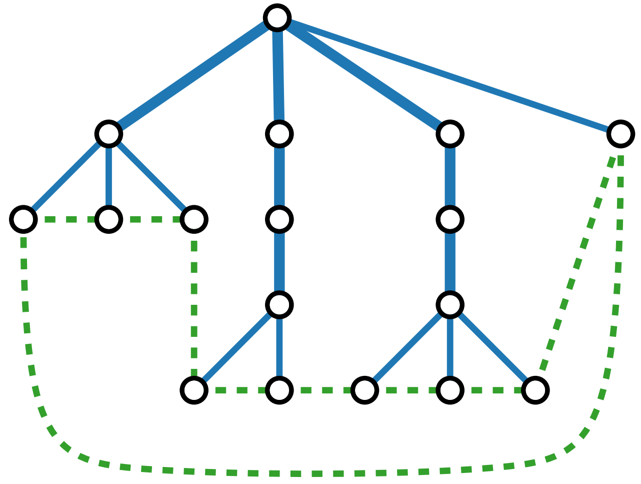
- A **cycle**
- A **tree** inside the cycle
- A **set of edges** between the cycle and the tree

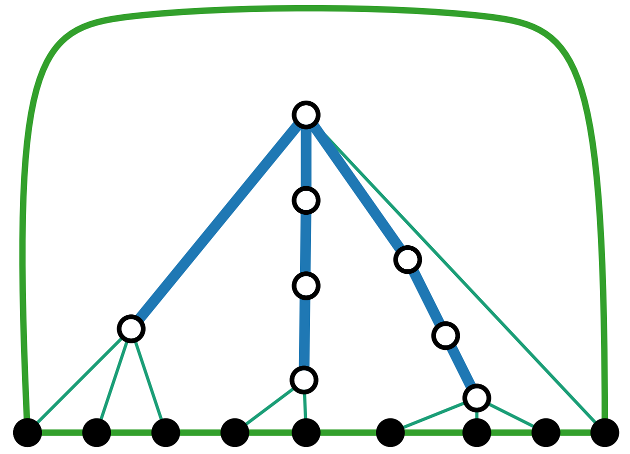


Outer face shares an edge with every other face

Every **Halin graph** is the subdivision of a 3-connected cycle-tree

Outer face shares a vertex with every other face

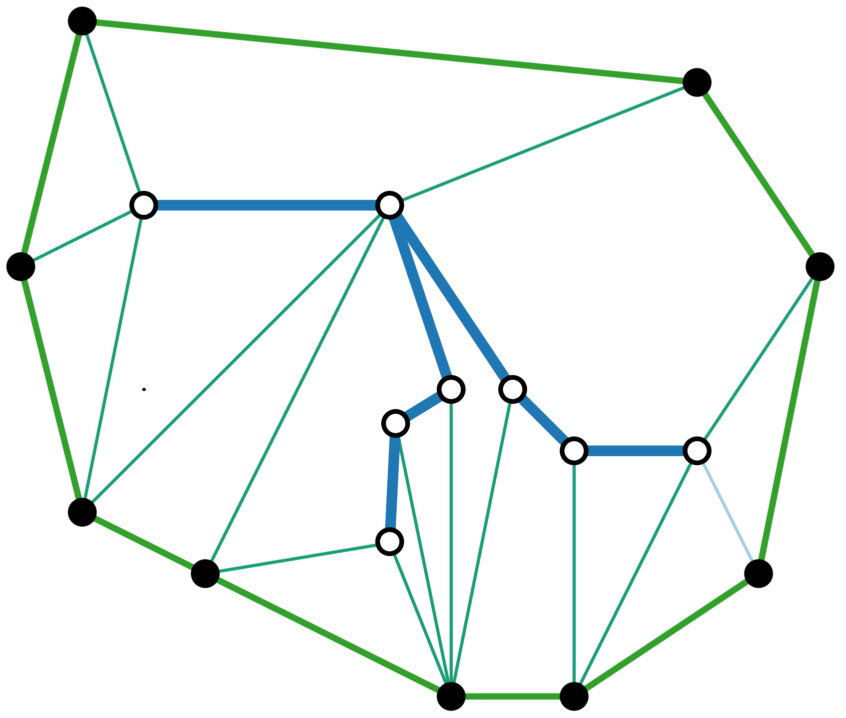




Cycle- and Path-trees

A **cycle-tree** consists of

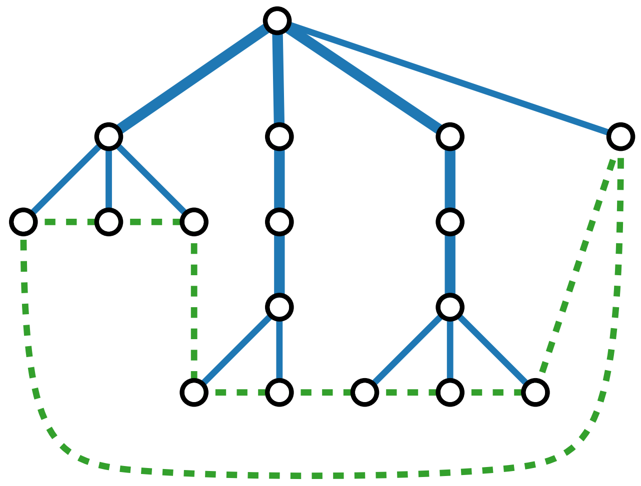
- A **cycle**
- A **tree** inside the cycle
- A **set of edges** between the cycle and the tree

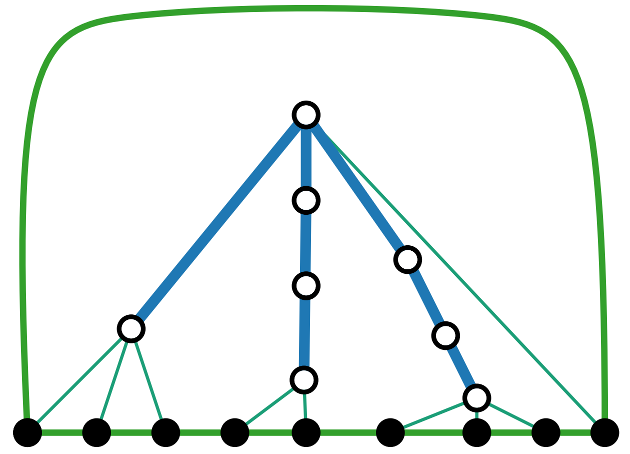


Outer face shares an edge with every other face

Every **Halin graph** is the subdivision of a 3-connected cycle-tree

Outer face shares a vertex with every other face



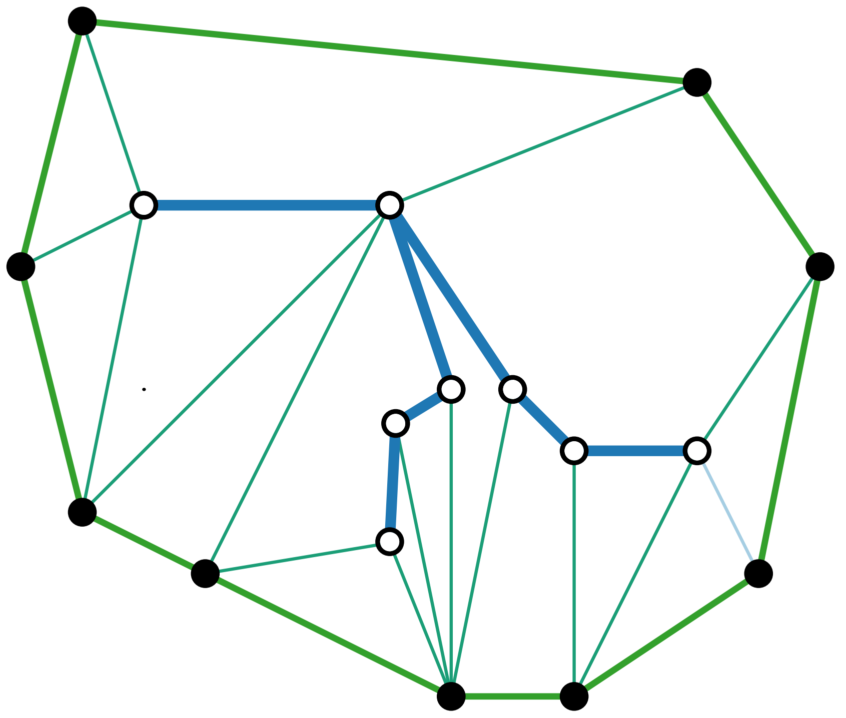


A **path-tree** is a cycle-tree minus one cycle edge

Cycle- and Path-trees

A **cycle-tree** consists of

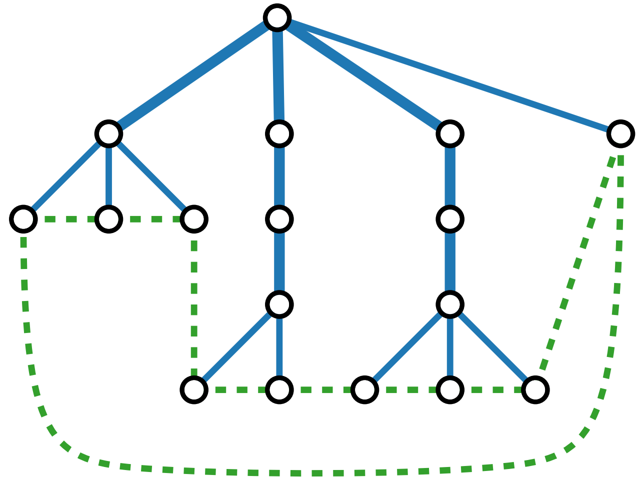
- A **cycle**
- A **tree** inside the cycle
- A **set of edges** between the cycle and the tree

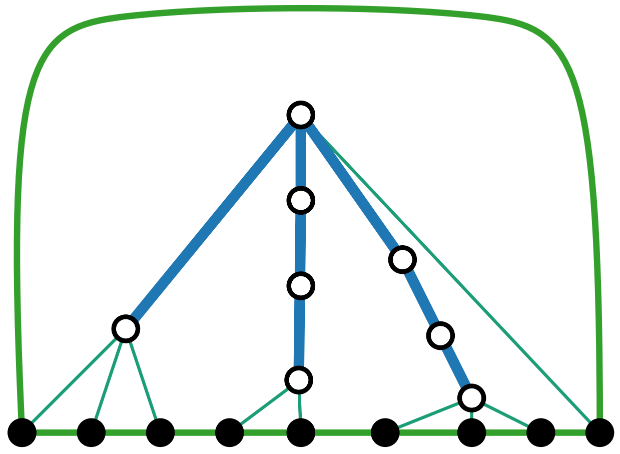


Outer face shares an edge with every other face

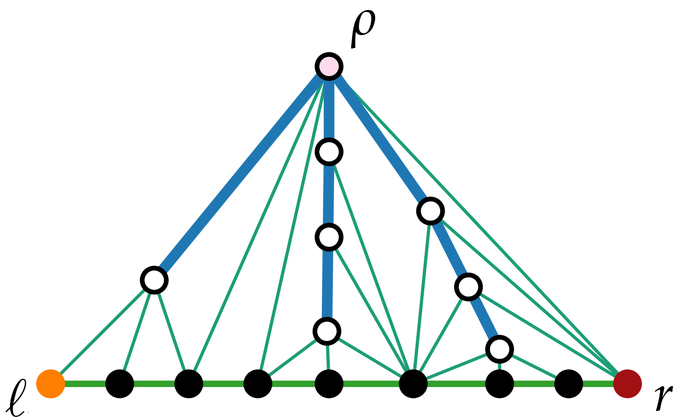
Every **Halin graph** is the subdivision of a 3-connected cycle-tree

Outer face shares a vertex with every other face





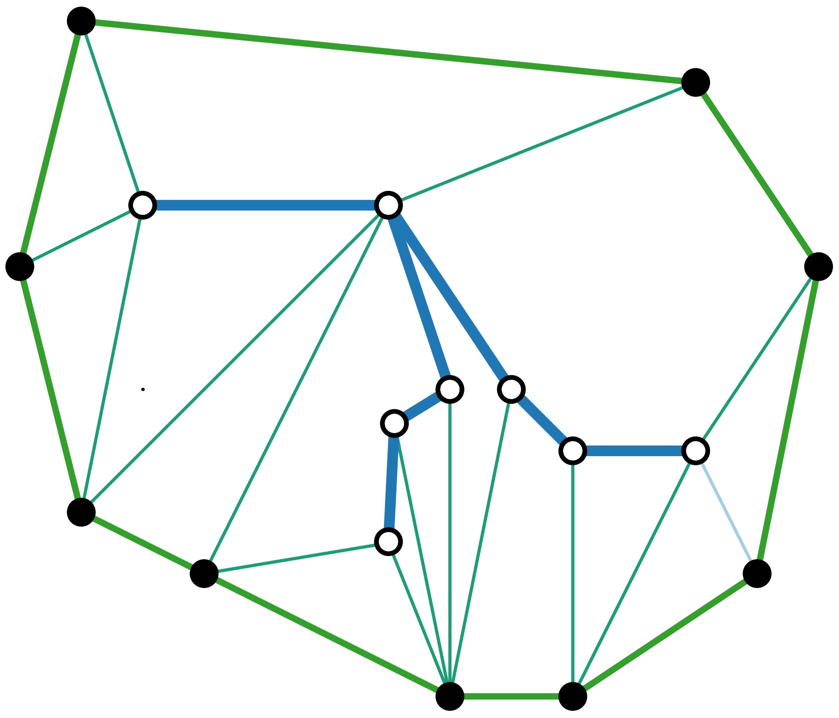
A **path-tree** is a cycle-tree minus one cycle edge



Cycle- and Path-trees

A **cycle-tree** consists of

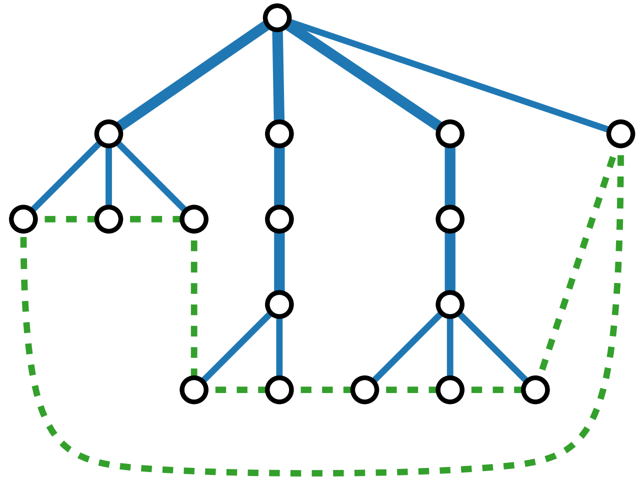
- A **cycle**
- A **tree** inside the cycle
- A **set of edges** between the cycle and the tree

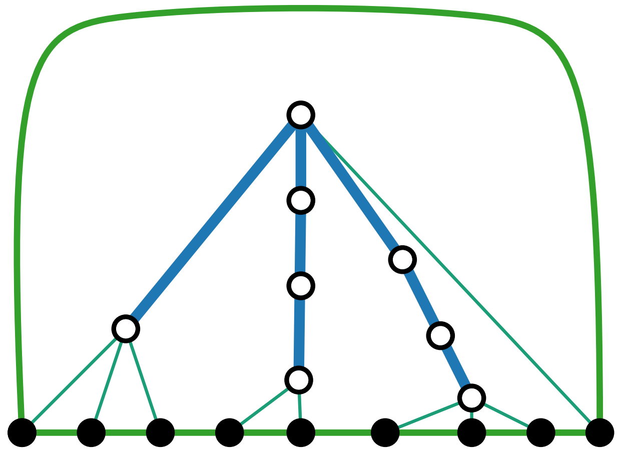


Outer face shares an edge with every other face

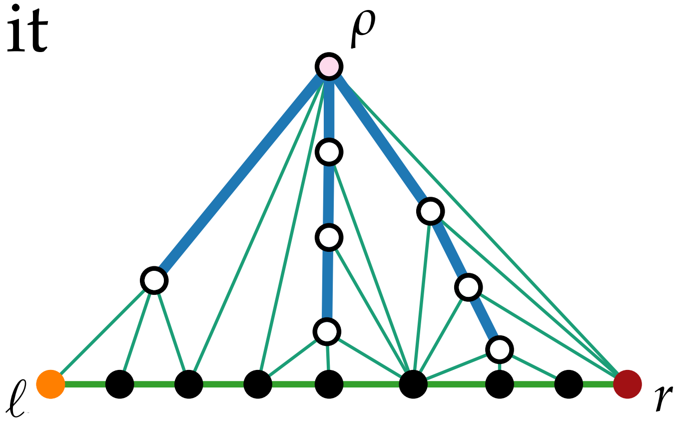
Every **Halin graph** is the subdivision of a 3-connected cycle-tree

Outer face shares a vertex with every other face





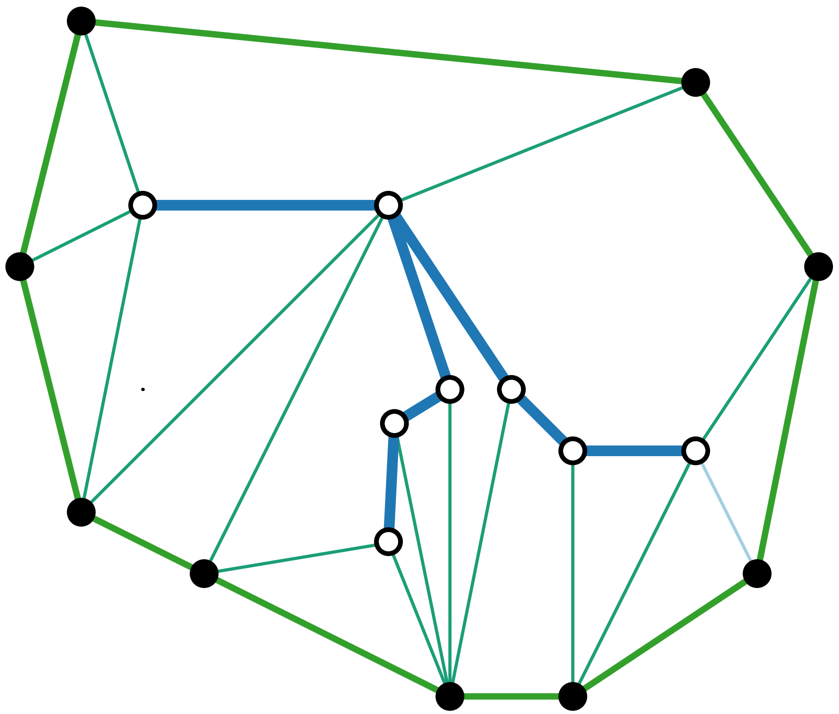
A **path-tree** is a cycle-tree minus one cycle edge
It is **almost-3-connected** if it becomes 3-connected by connecting ρ, ℓ, r



Cycle- and Path-trees

A **cycle-tree** consists of

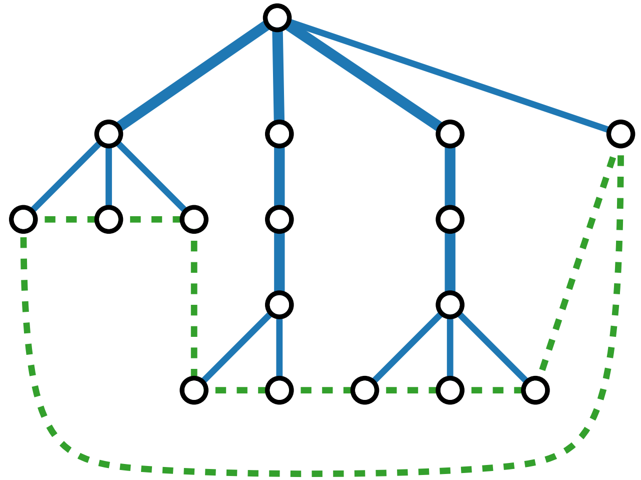
- A **cycle**
- A **tree** inside the cycle
- A **set of edges** between the cycle and the tree

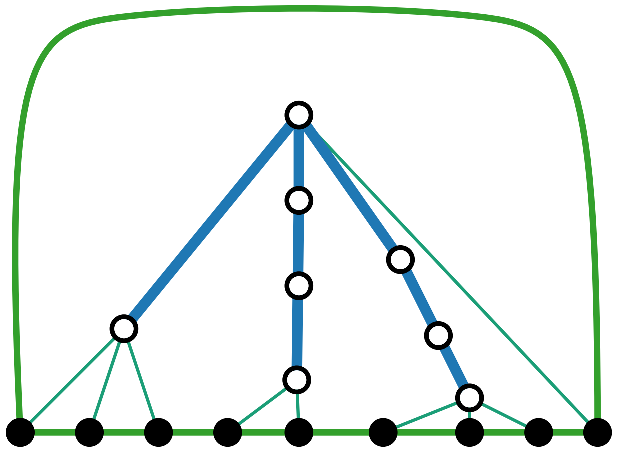


Outer face shares an edge with every other face

Every **Halin graph** is the subdivision of a 3-connected cycle-tree

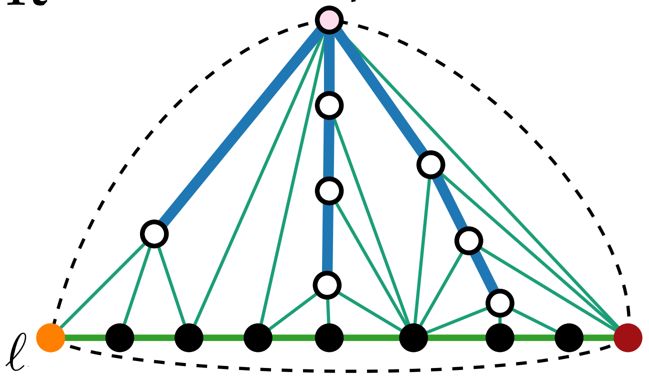
Outer face shares a vertex with every other face





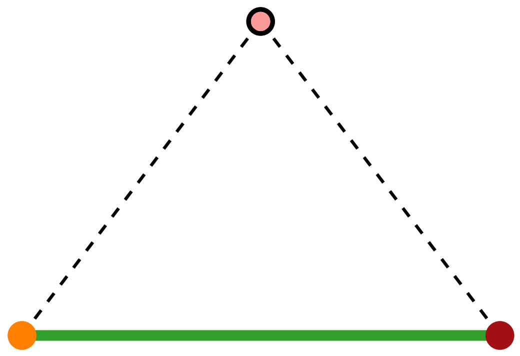
A **path-tree** is a cycle-tree minus one cycle edge

It is **almost-3-connected** if it becomes 3-connected by connecting ρ, ℓ, r



SPQ-(De)composition of Path-Trees

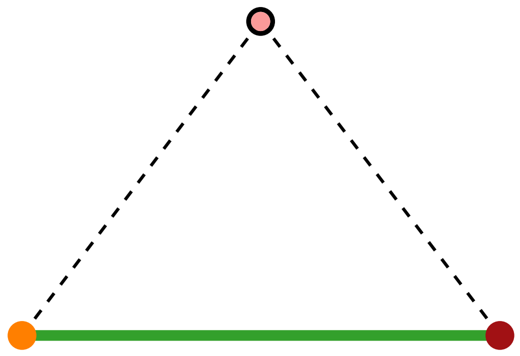
Every almost-3-connected path-tree has an SPQ-decomposition.



SPQ-(De)composition of Path-Trees

Every almost-3-connected path-tree has an SPQ-decomposition.

Solid edges must exist, dashed edges might exist

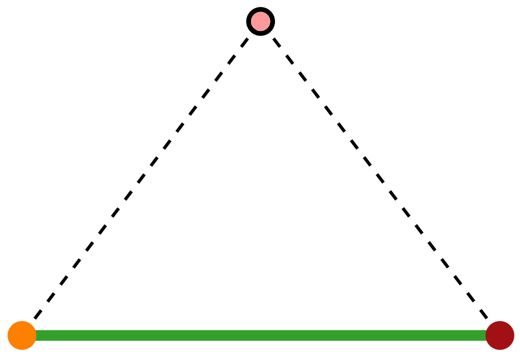


SPQ-(De)composition of Path-Trees

Every almost-3-connected path-tree has an SPQ-decomposition.

Solid edges must exist, dashed edges might exist

- **Q-node:**

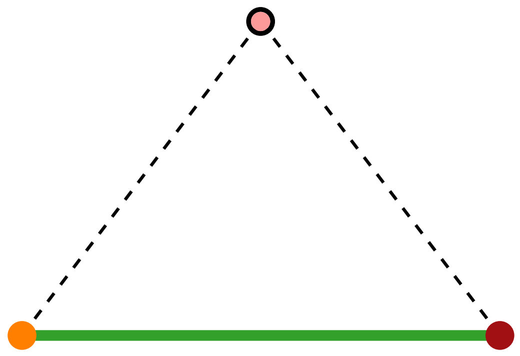


SPQ-(De)composition of Path-Trees

Every almost-3-connected path-tree has an SPQ-decomposition.

Solid edges must exist, dashed edges might exist

- **Q-node:**



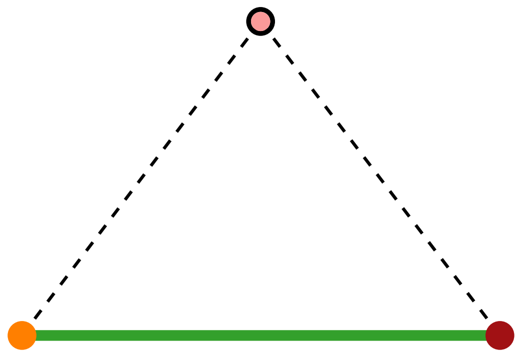
- **S-node:**

SPQ-(De)composition of Path-Trees

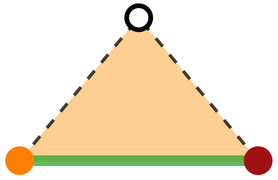
Every almost-3-connected path-tree has an SPQ-decomposition.

Solid edges must exist, dashed edges might exist

- **Q-node:**



- **S-node:**

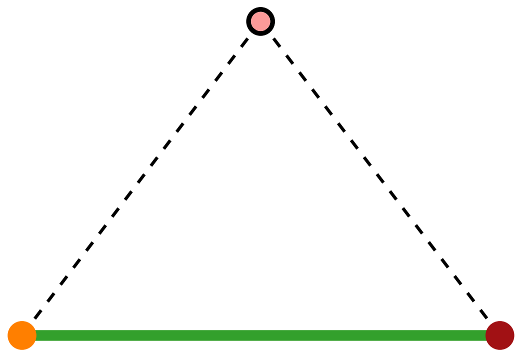


SPQ-(De)composition of Path-Trees

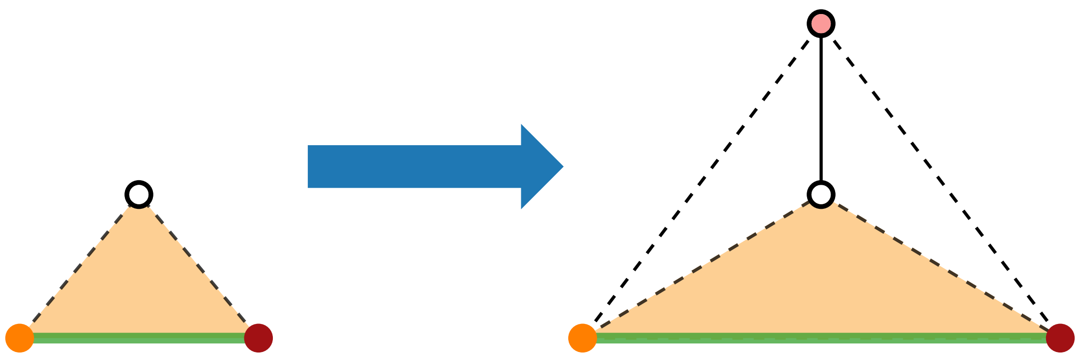
Every almost-3-connected path-tree has an SPQ-decomposition.

Solid edges must exist, dashed edges might exist

- **Q-node:**



- **S-node:**

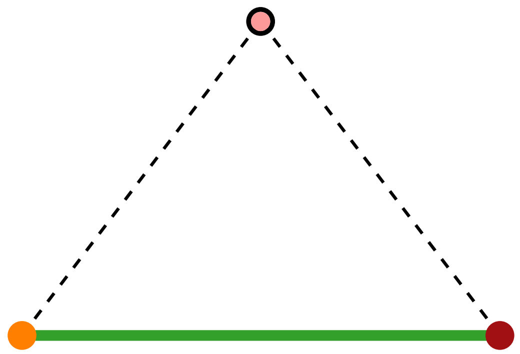


SPQ-(De)composition of Path-Trees

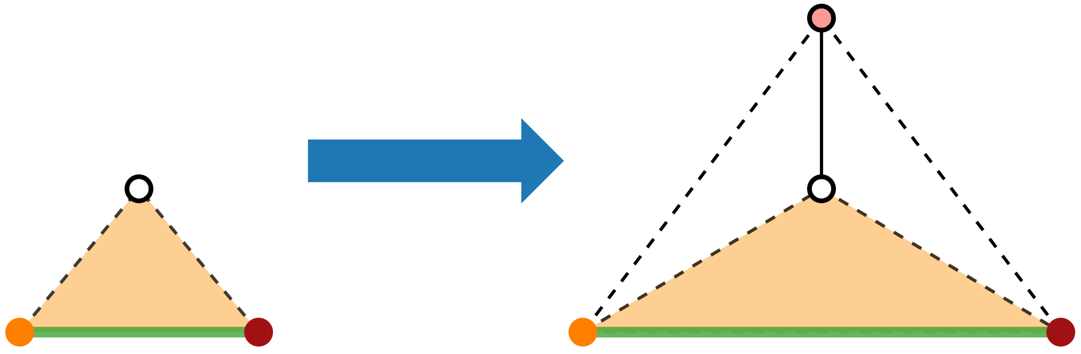
Every almost-3-connected path-tree has an SPQ-decomposition.

Solid edges must exist, dashed edges might exist

- **Q-node:**



- **S-node:**



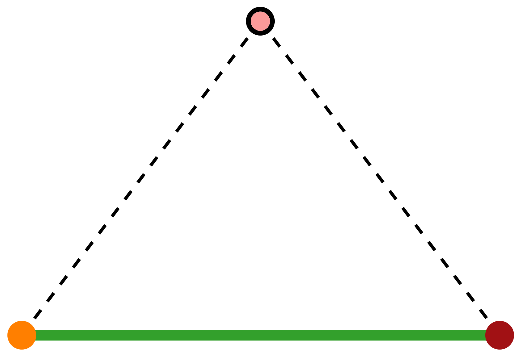
- **P-node:**

SPQ-(De)composition of Path-Trees

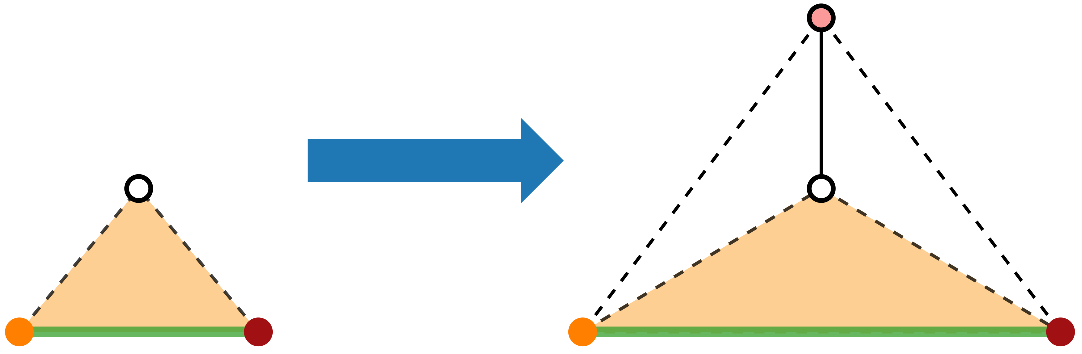
Every almost-3-connected path-tree has an SPQ-decomposition.

Solid edges must exist, dashed edges might exist

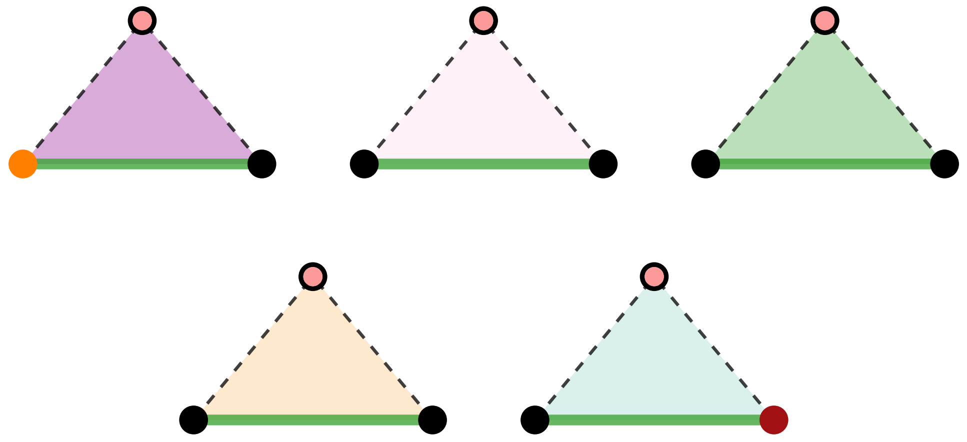
- **Q-node:**



- **S-node:**



- **P-node:**

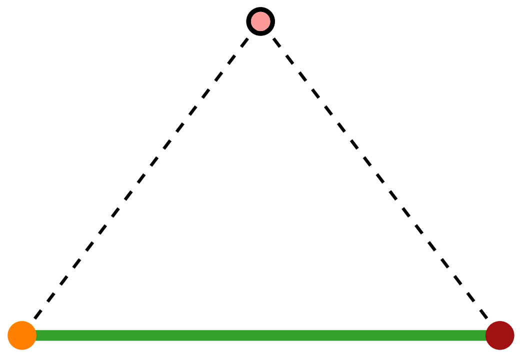


SPQ-(De)composition of Path-Trees

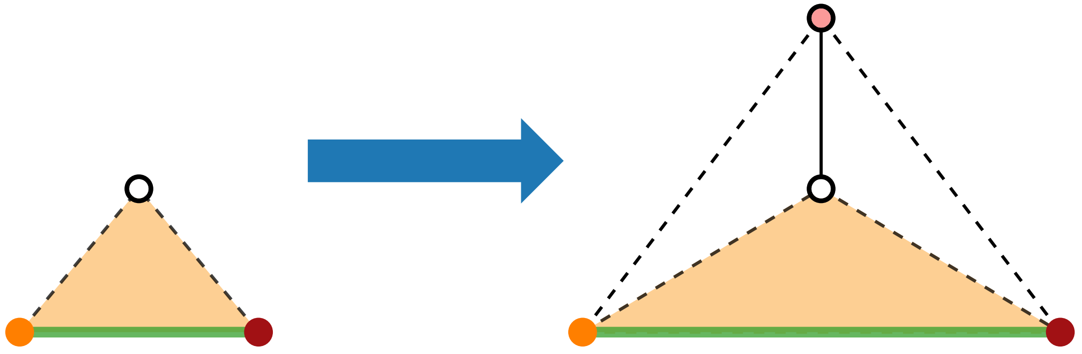
Every almost-3-connected path-tree has an SPQ-decomposition.

Solid edges must exist, dashed edges might exist

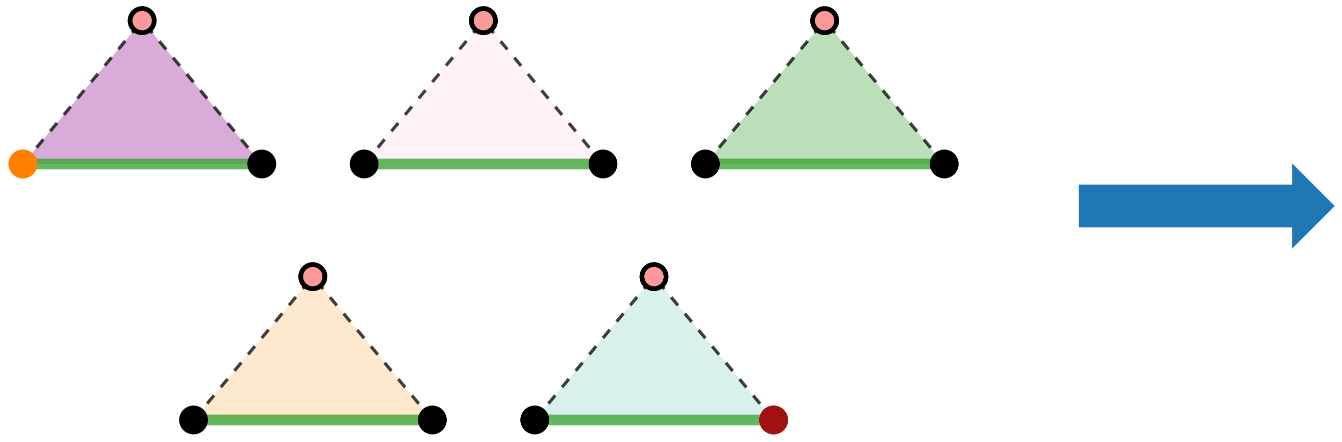
- **Q-node:**



- **S-node:**



- **P-node:**

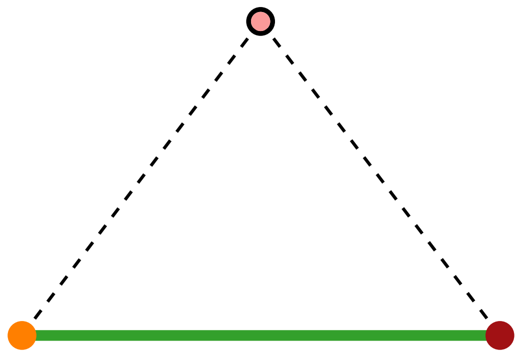


SPQ-(De)composition of Path-Trees

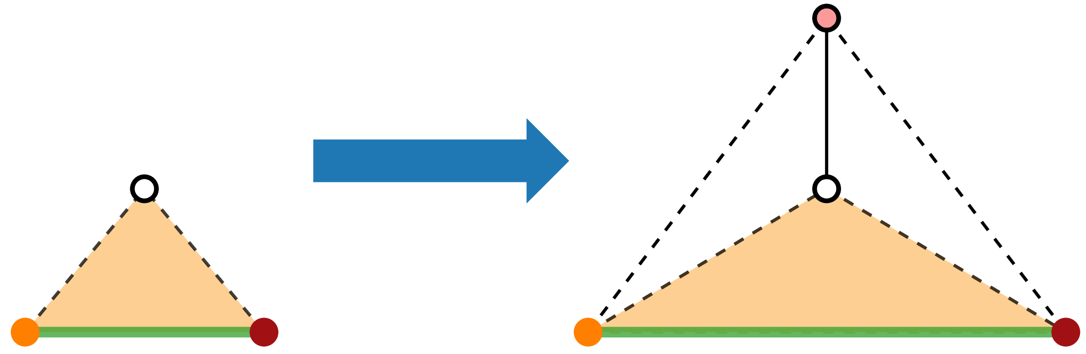
Every almost-3-connected path-tree has an SPQ-decomposition.

Solid edges must exist, dashed edges might exist

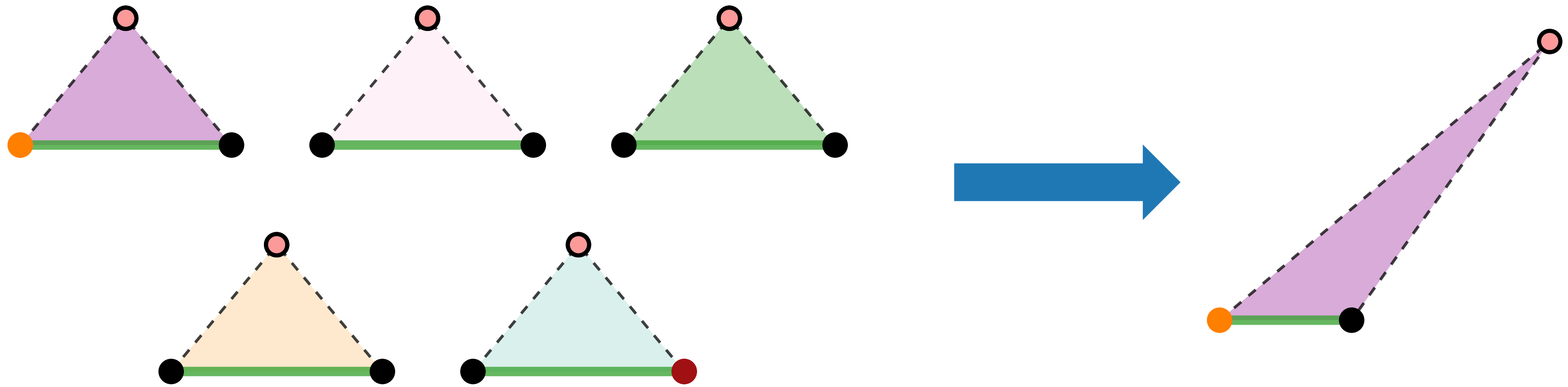
- **Q-node:**



- **S-node:**



- **P-node:**

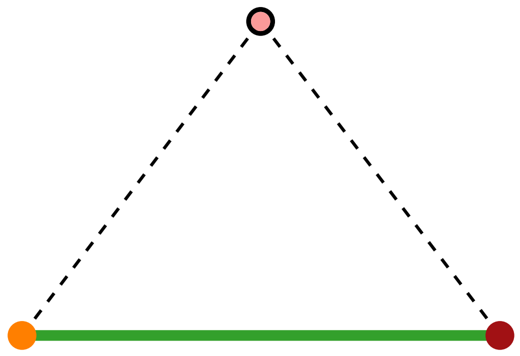


SPQ-(De)composition of Path-Trees

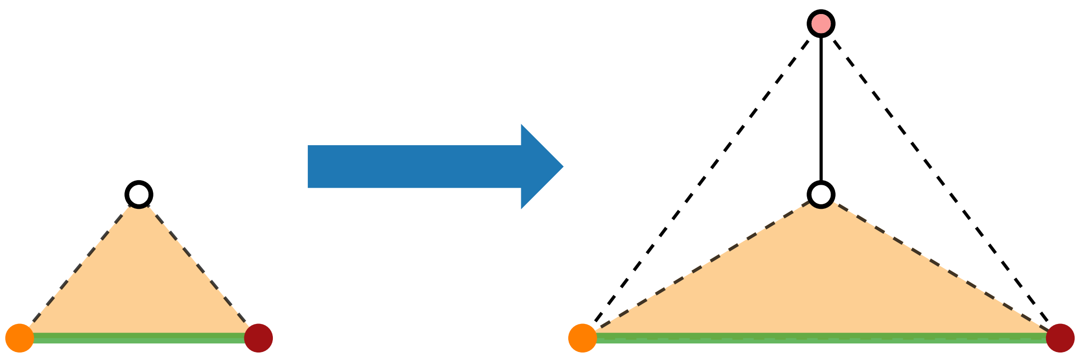
Every almost-3-connected path-tree has an SPQ-decomposition.

Solid edges must exist, dashed edges might exist

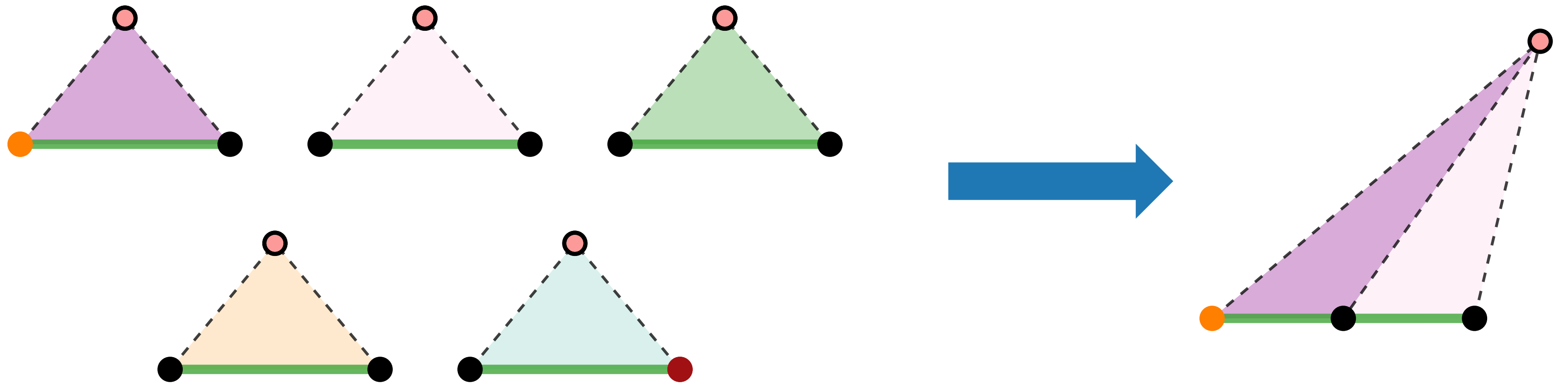
- **Q-node:**



- **S-node:**



- **P-node:**

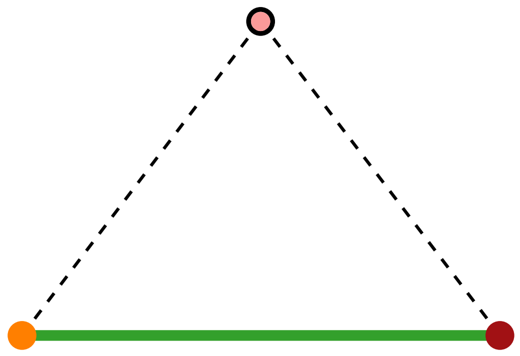


SPQ-(De)composition of Path-Trees

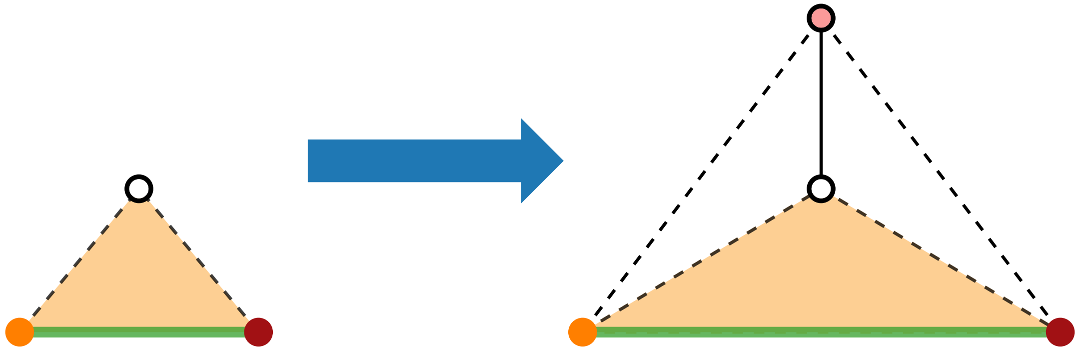
Every almost-3-connected path-tree has an SPQ-decomposition.

Solid edges must exist, dashed edges might exist

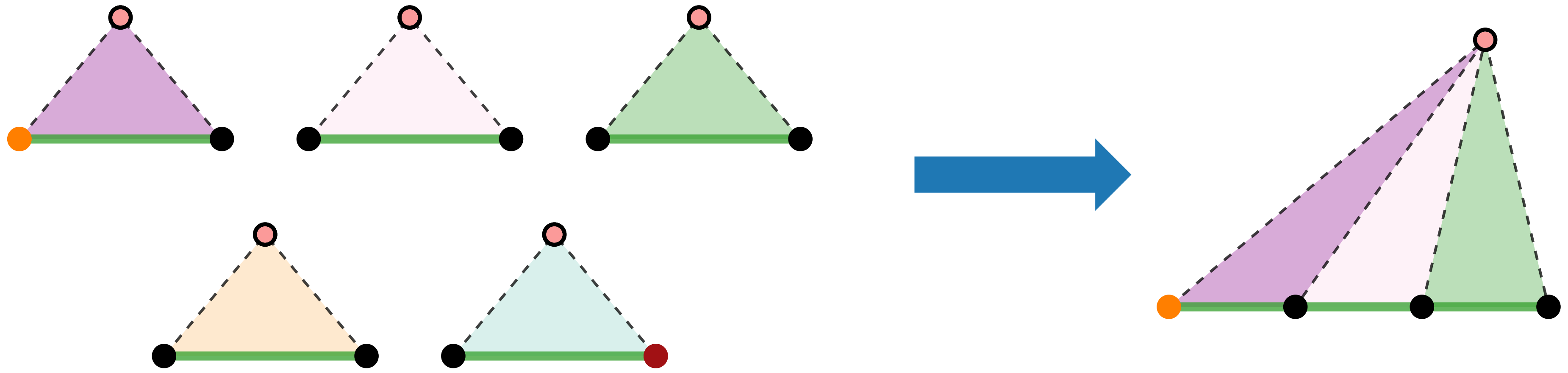
- **Q-node:**



- **S-node:**



- **P-node:**

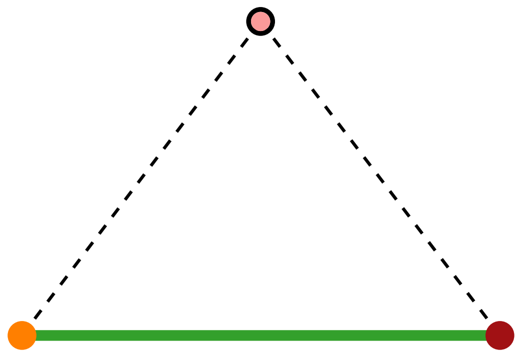


SPQ-(De)composition of Path-Trees

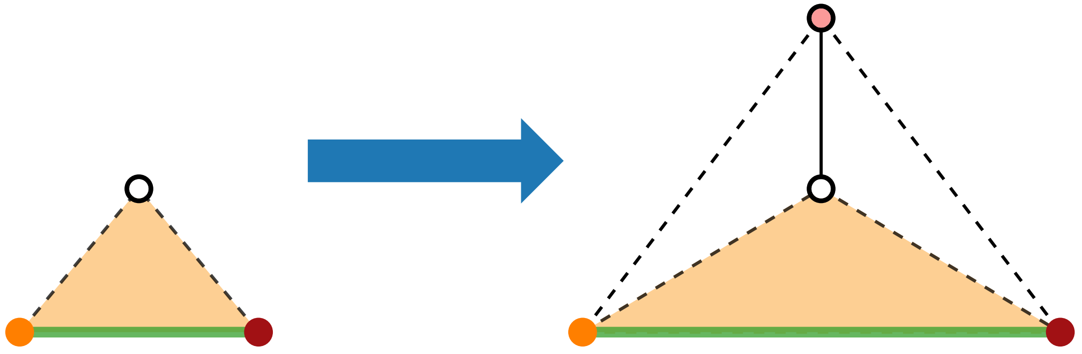
Every almost-3-connected path-tree has an SPQ-decomposition.

Solid edges must exist, dashed edges might exist

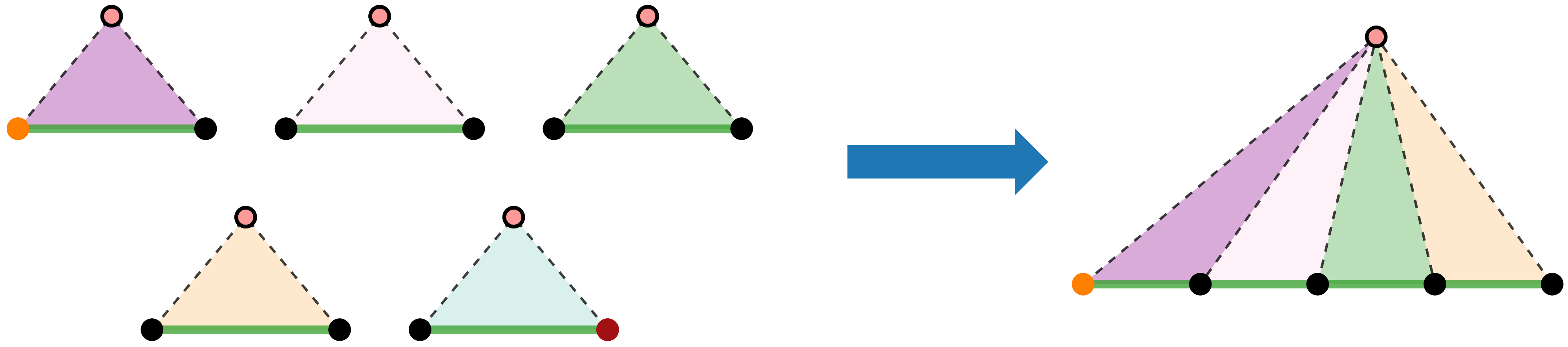
- **Q-node:**



- **S-node:**



- **P-node:**

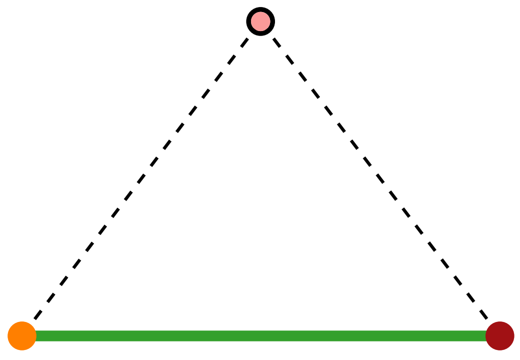


SPQ-(De)composition of Path-Trees

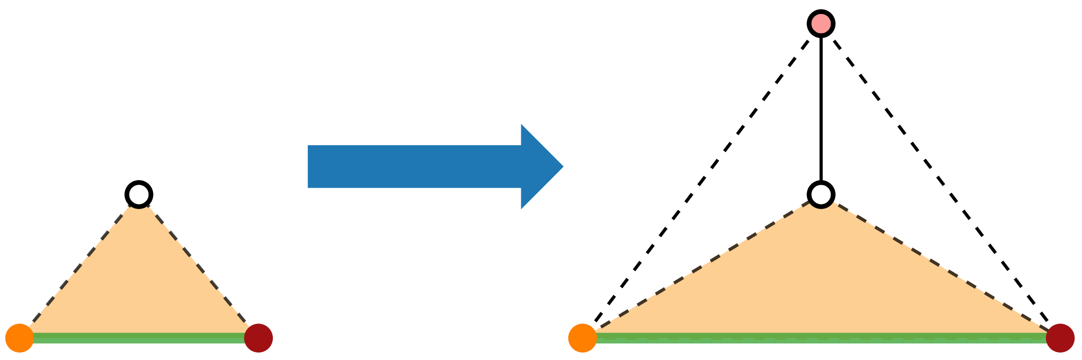
Every almost-3-connected path-tree has an SPQ-decomposition.

Solid edges must exist, dashed edges might exist

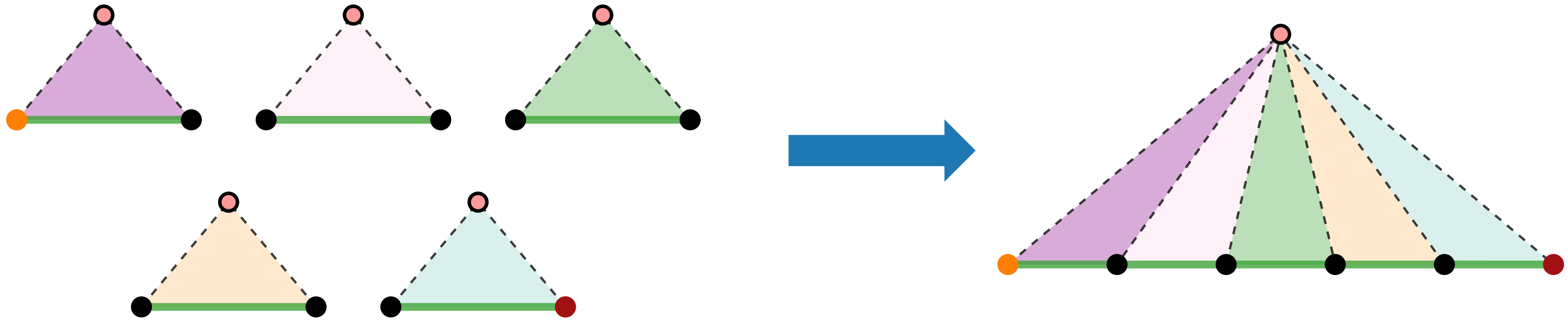
- **Q-node:**



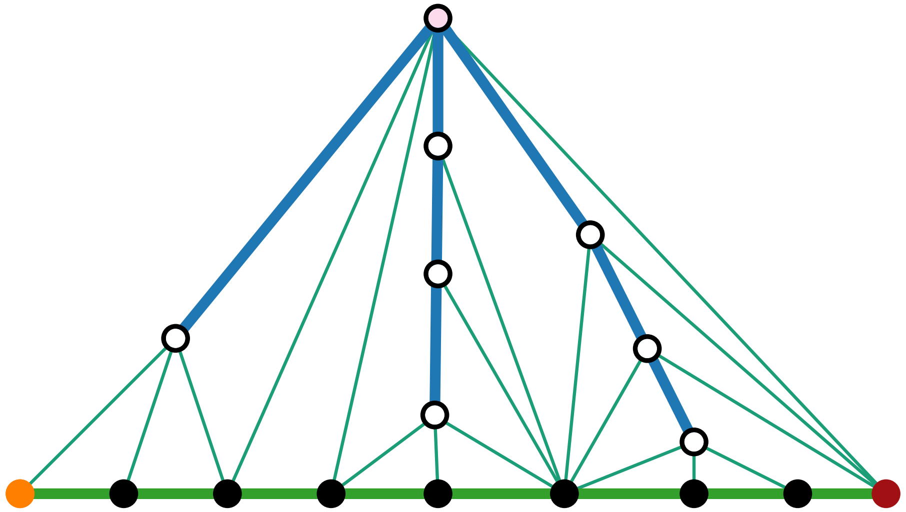
- **S-node:**



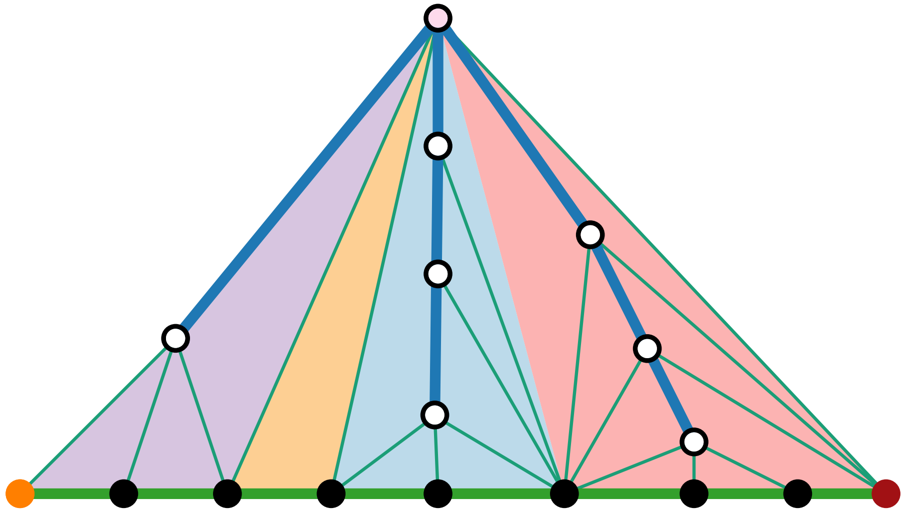
- **P-node:**



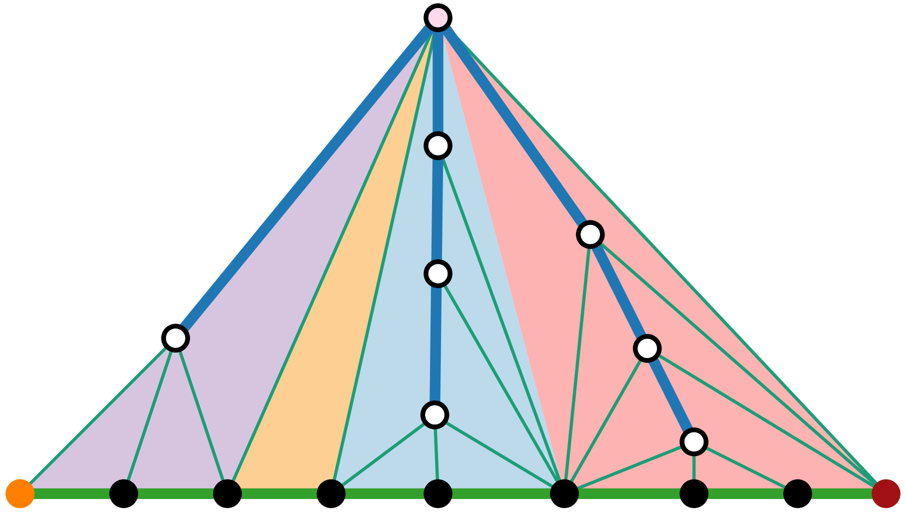
SPQ-Decomposition

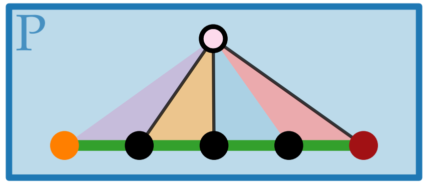


SPQ-Decomposition

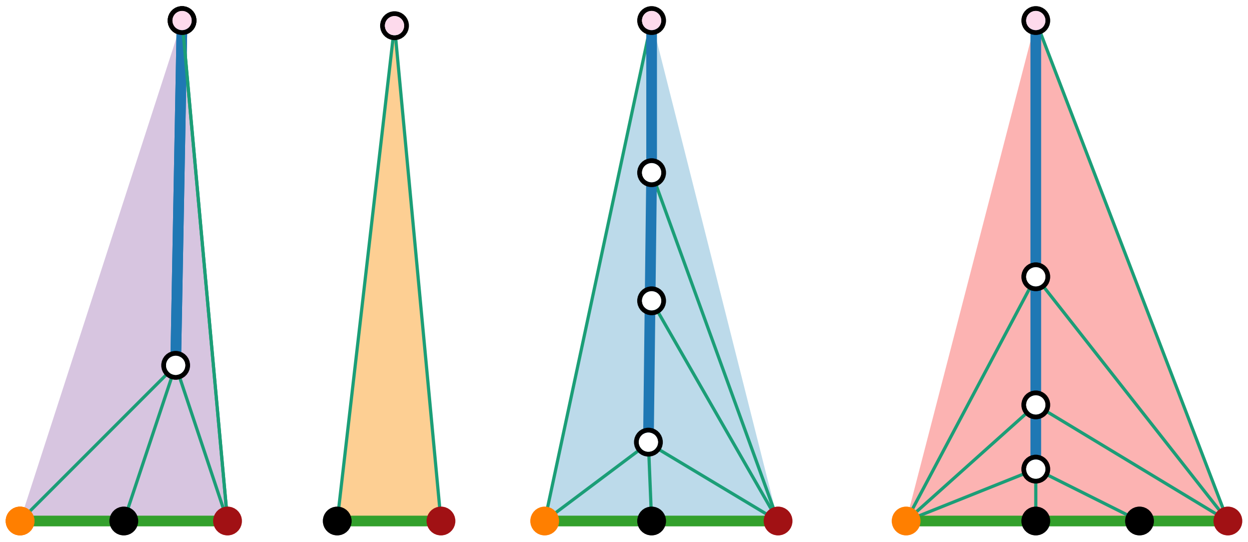


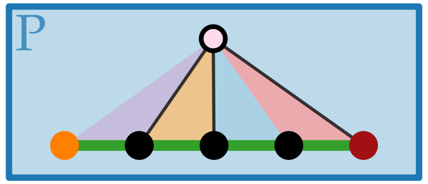
SPQ-Decomposition



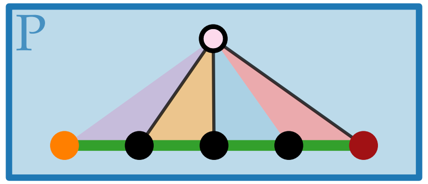


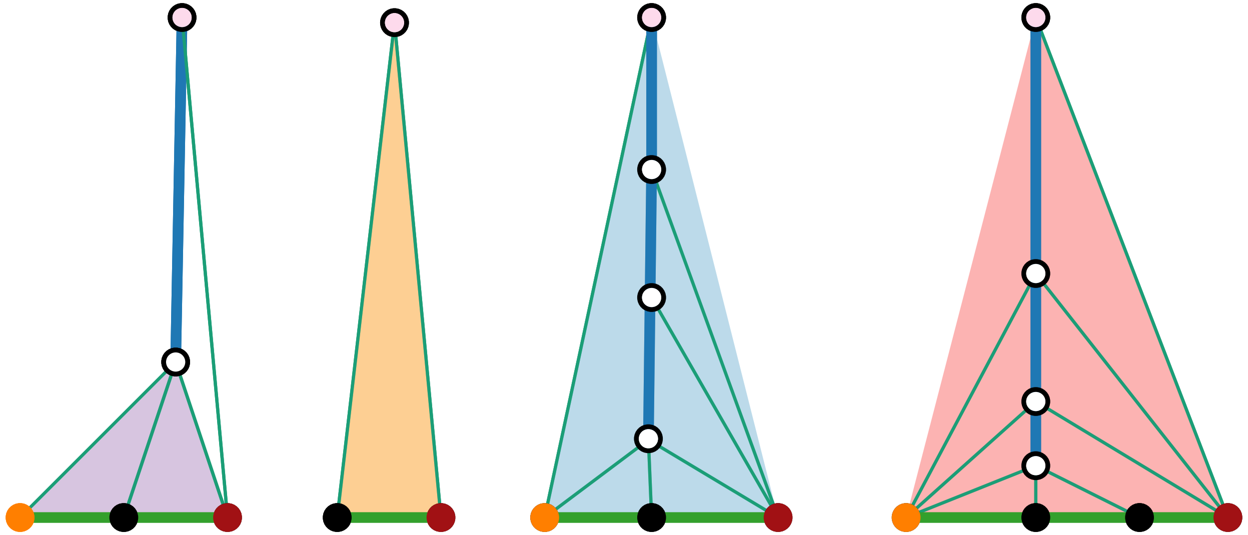
SPQ-Decomposition



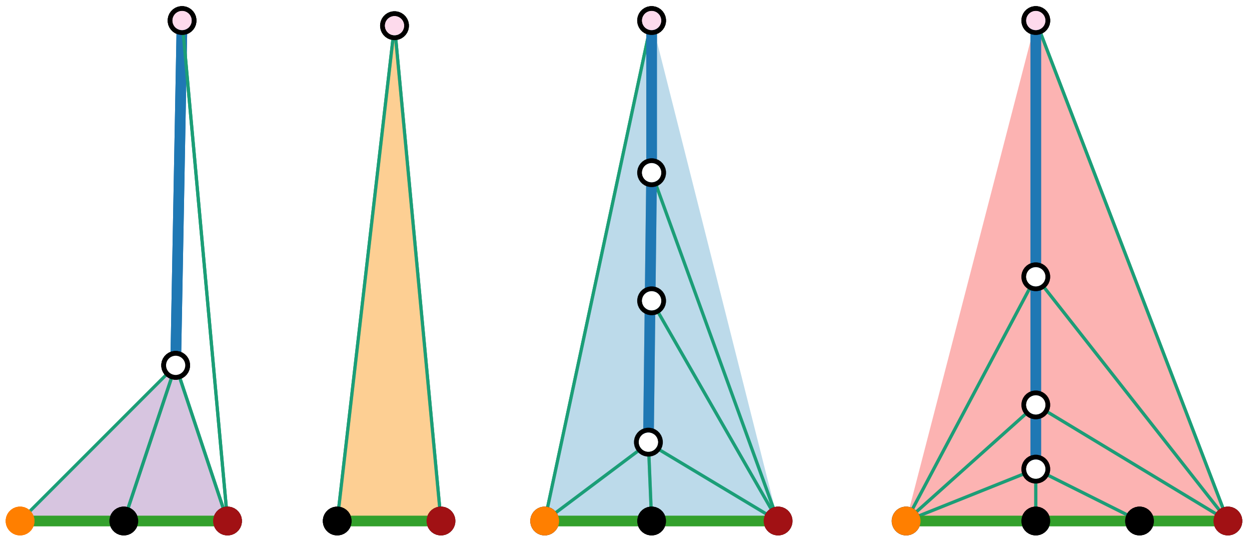


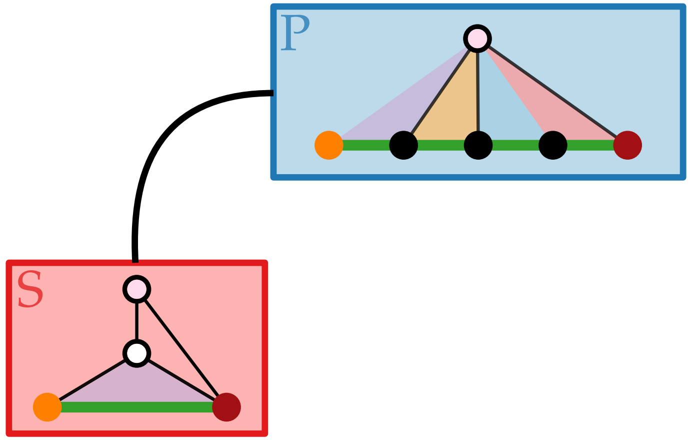
SPQ-Decomposition



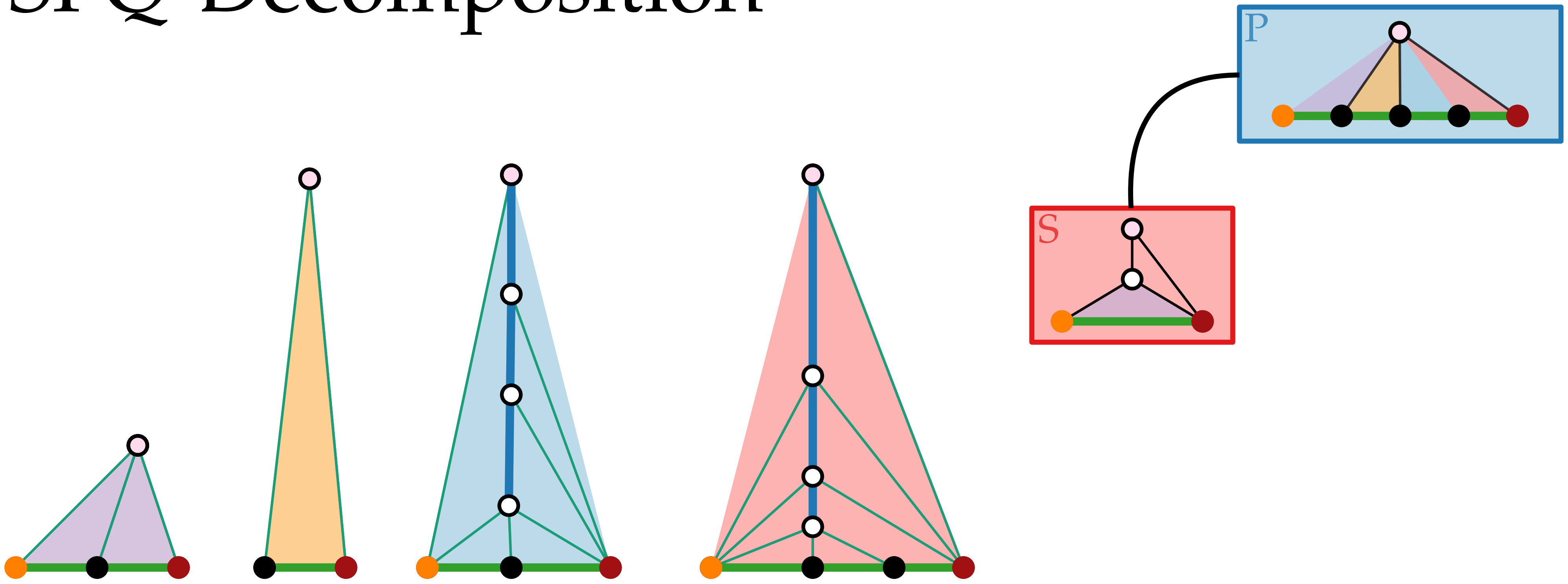


SPQ-Decomposition

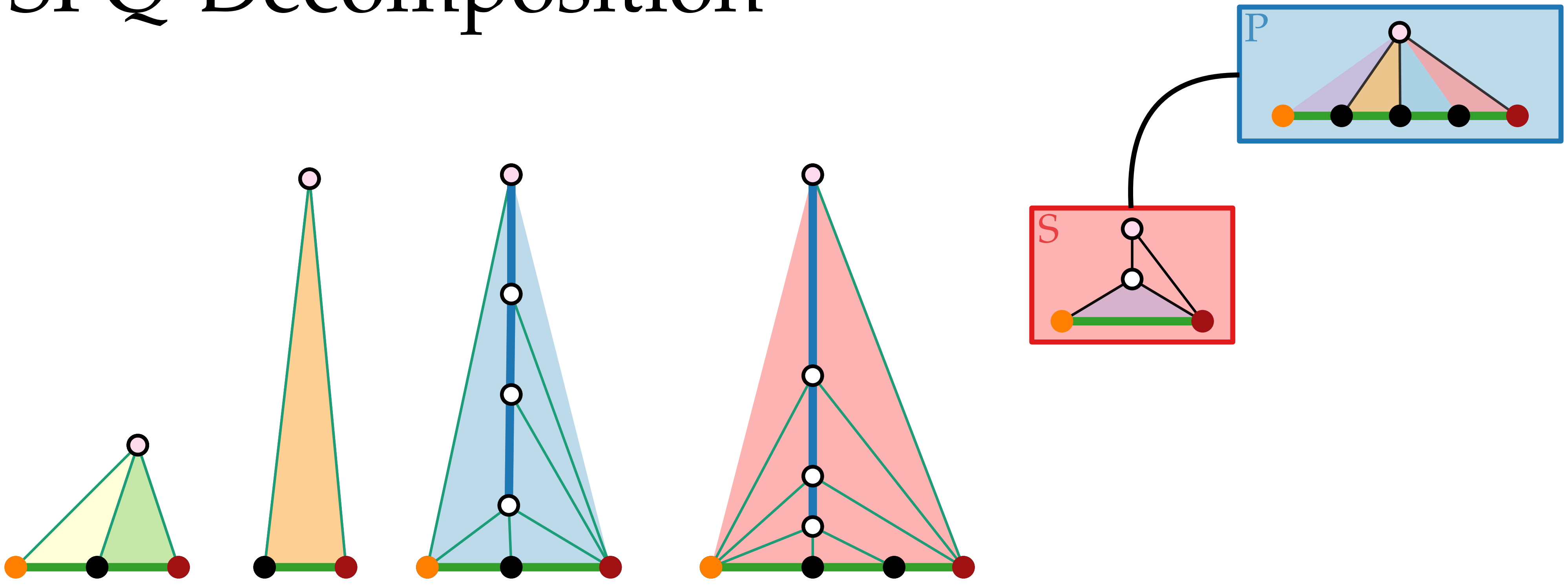




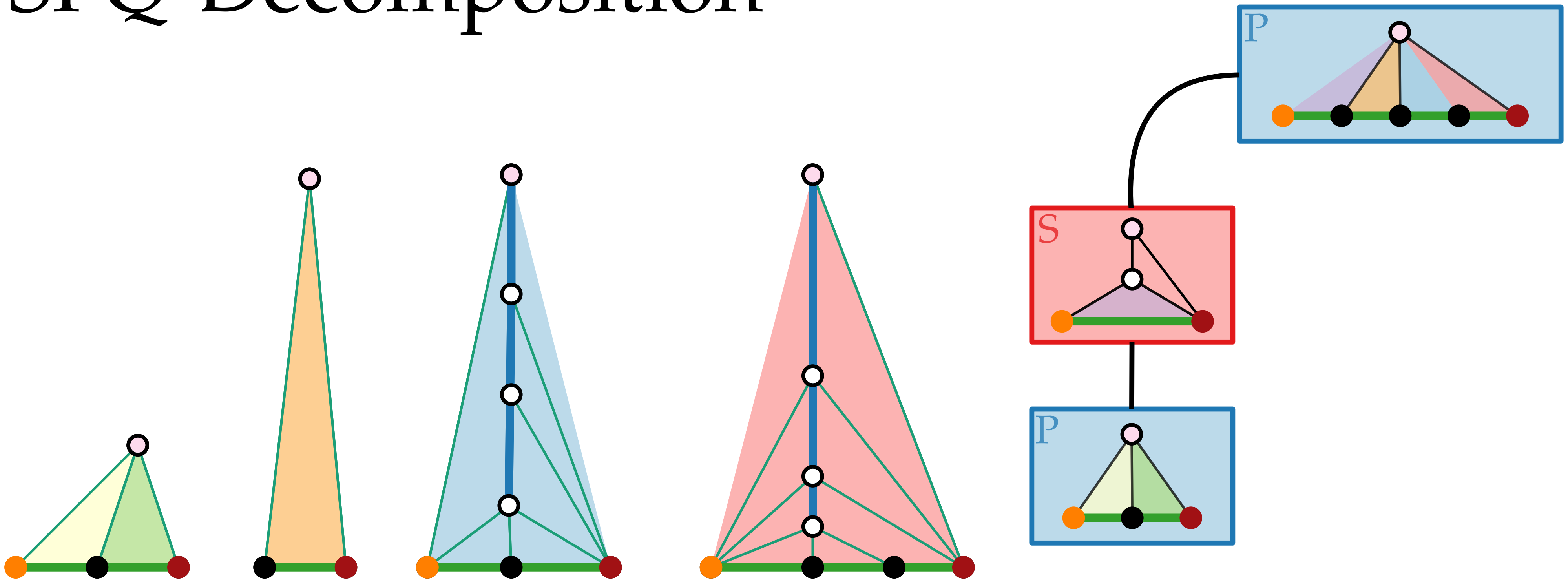
SPQ-Decomposition



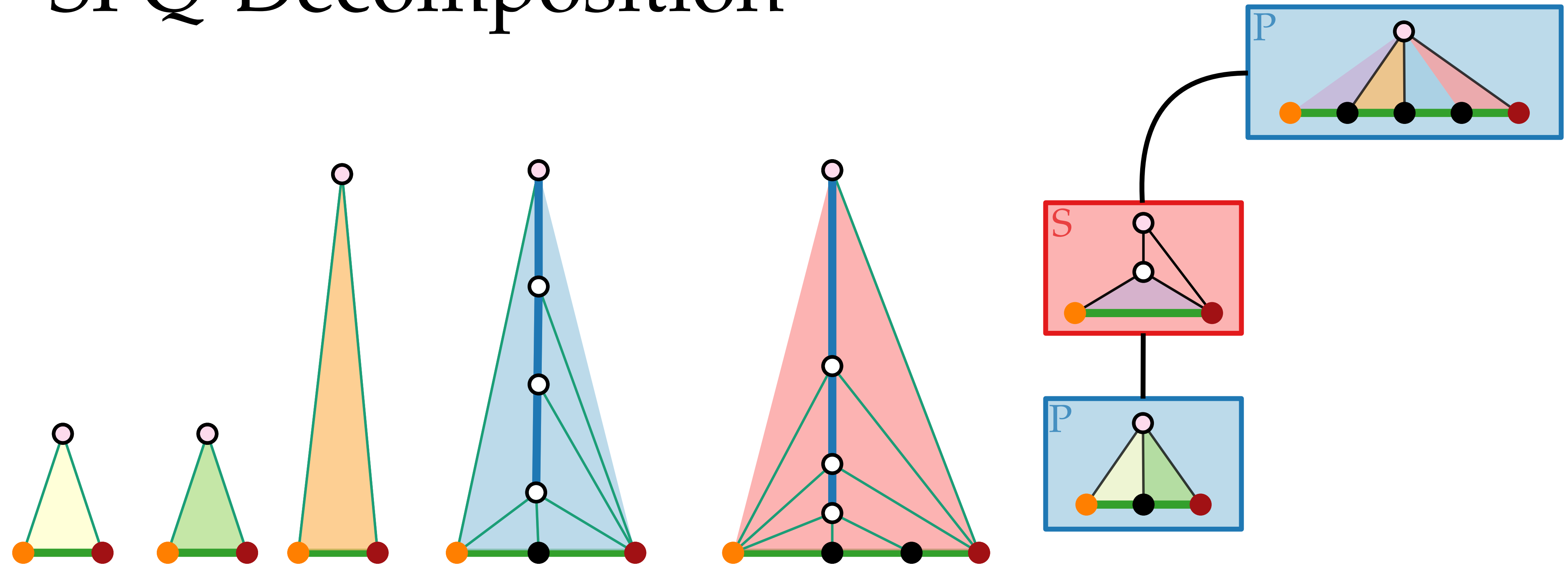
SPQ-Decomposition



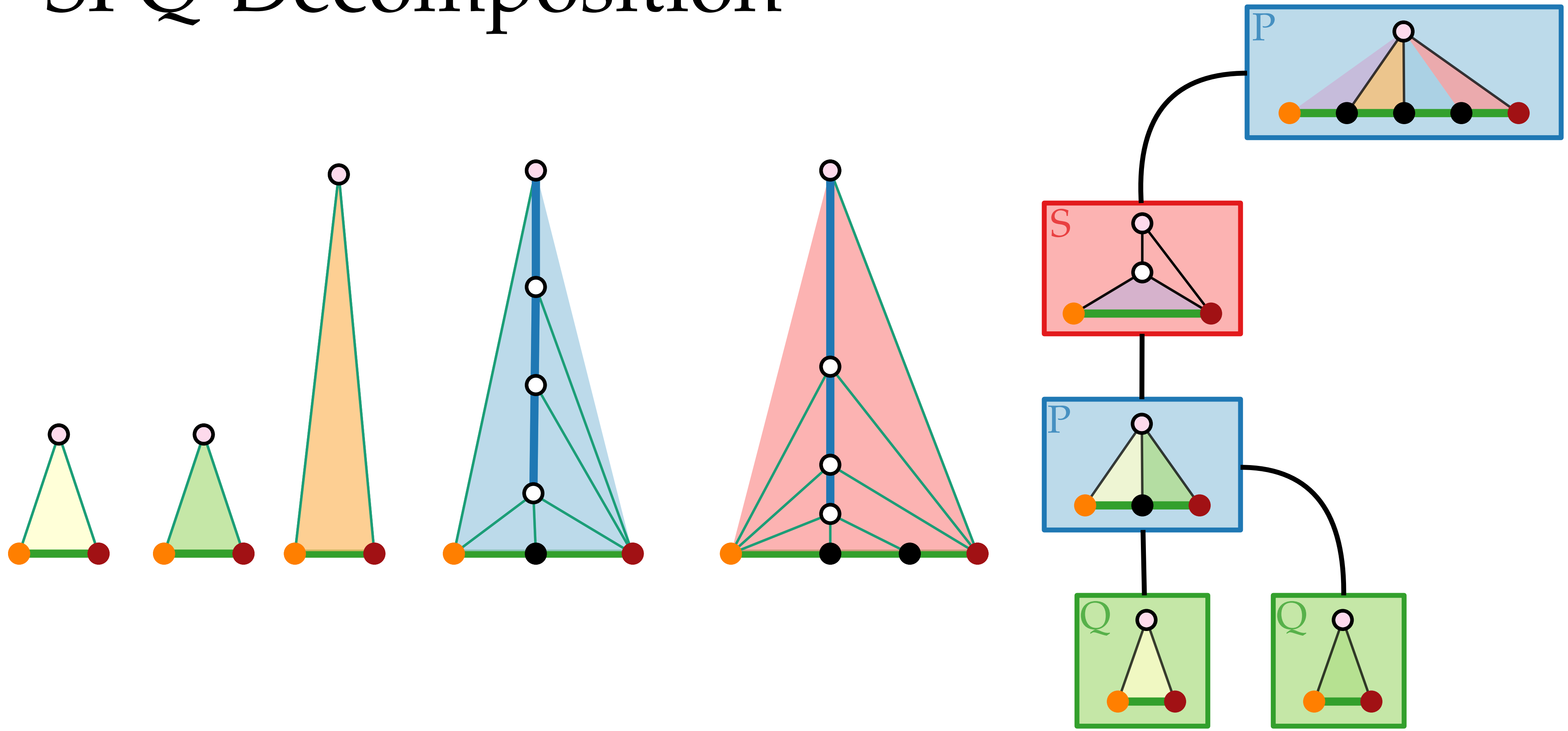
SPQ-Decomposition



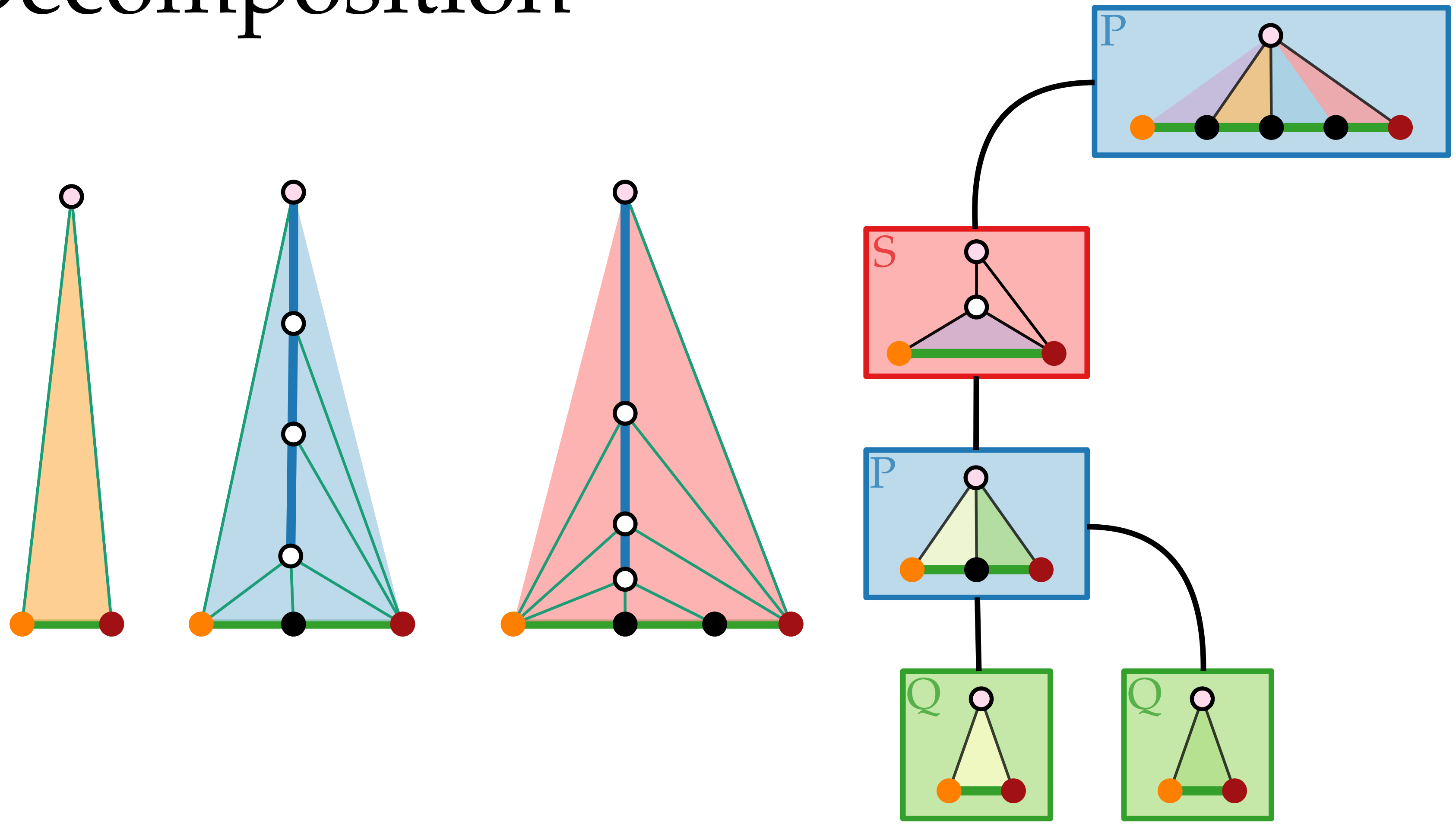
SPQ-Decomposition



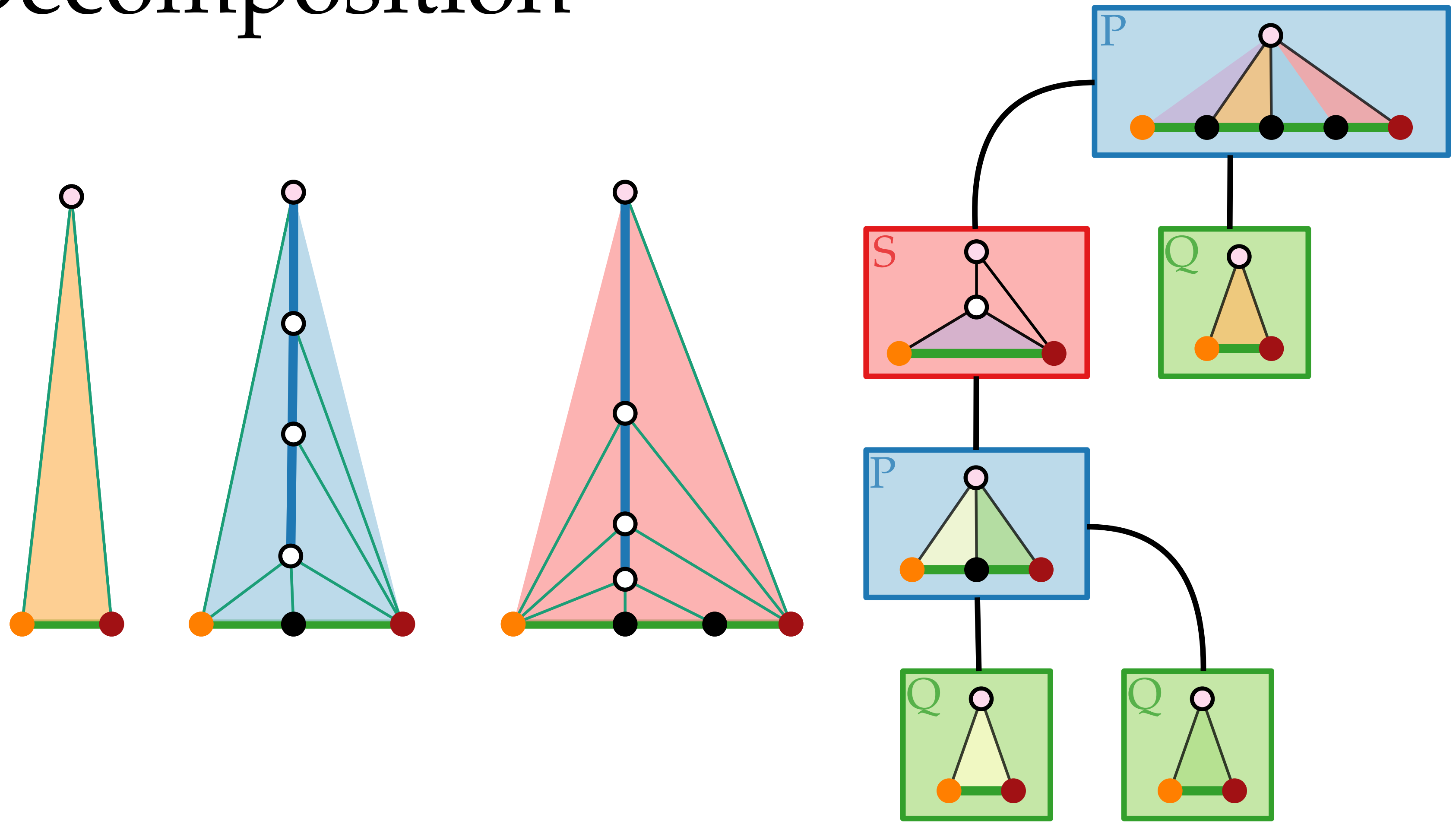
SPQ-Decomposition



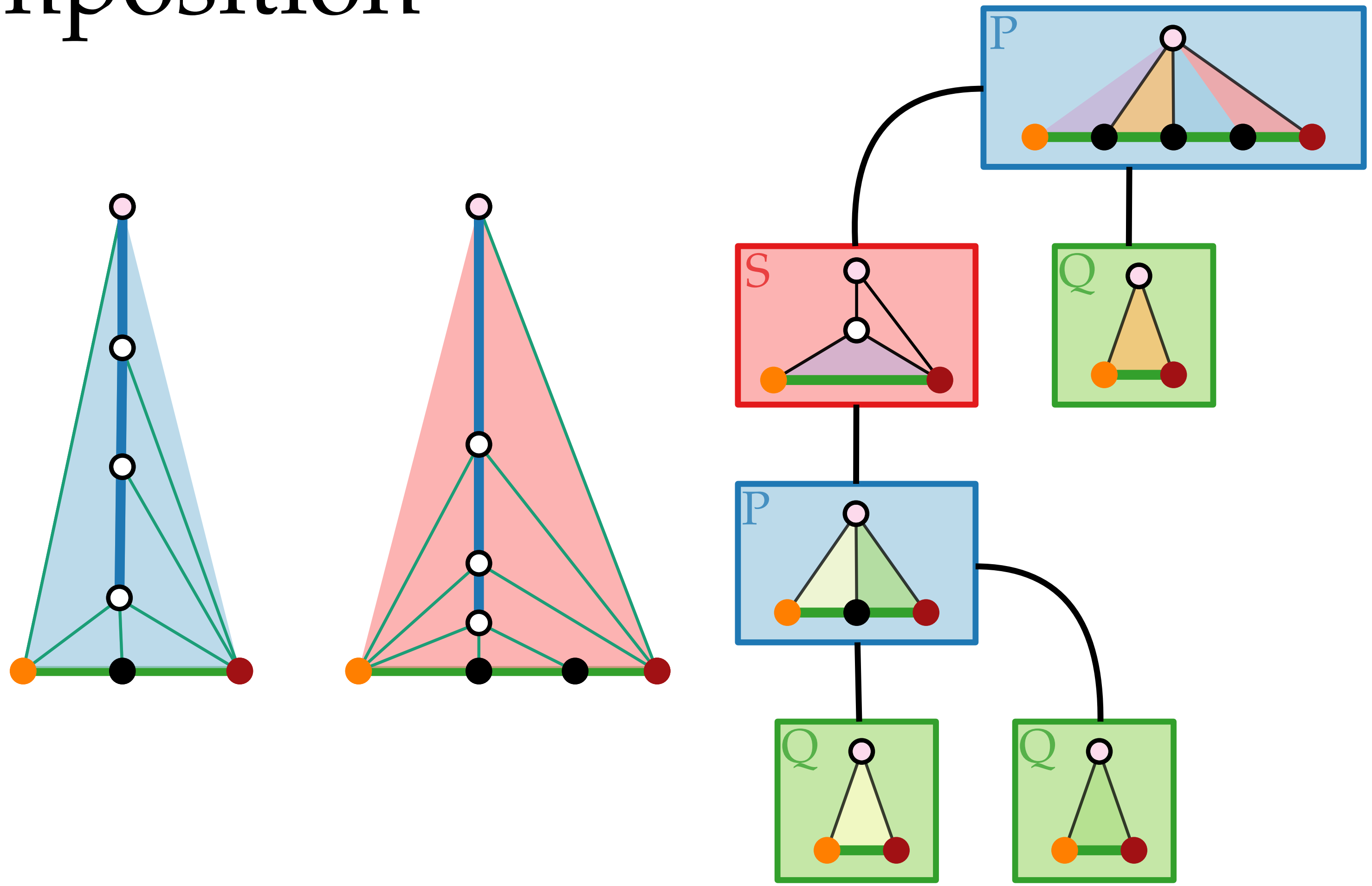
SPQ-Decomposition



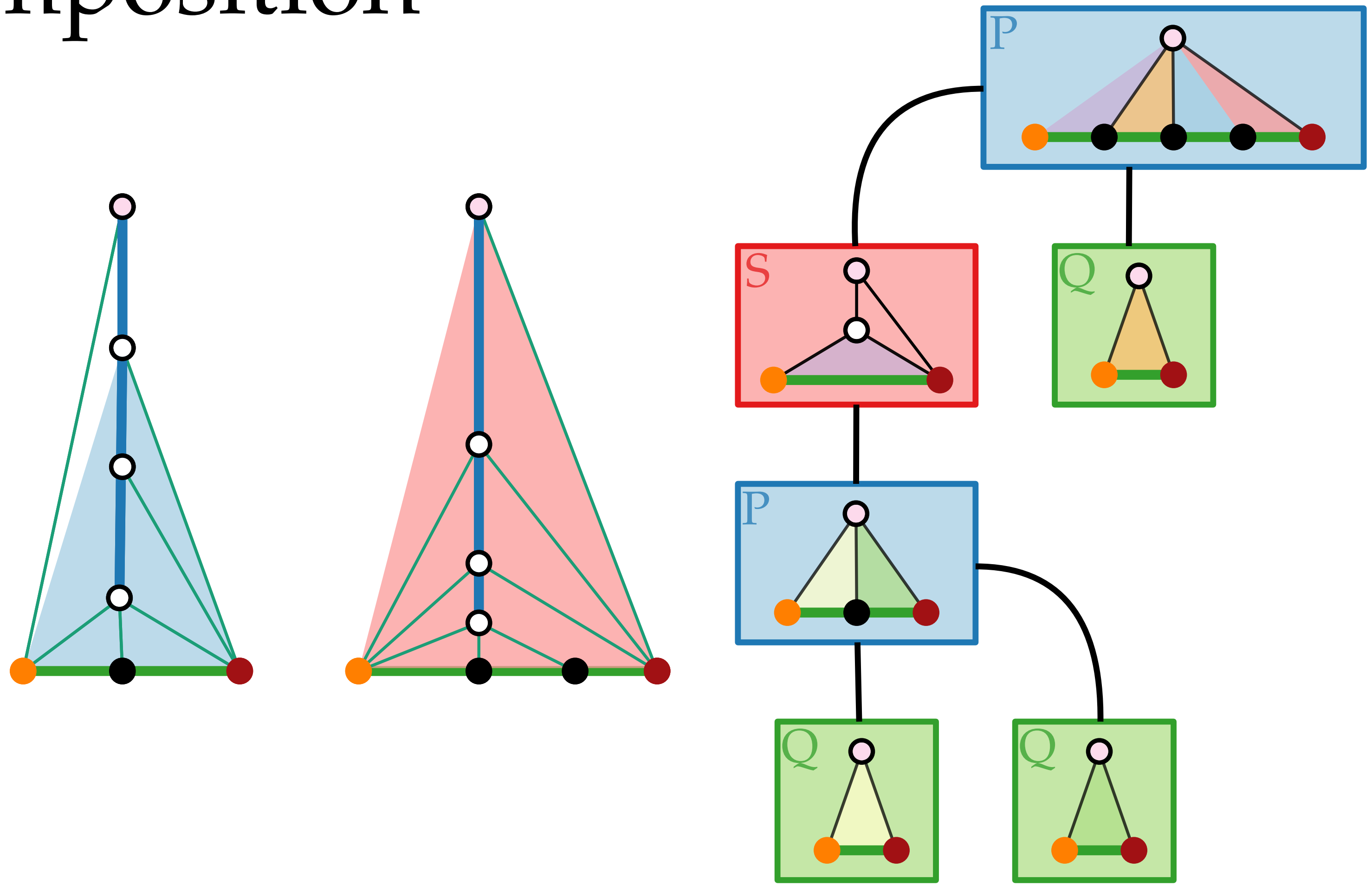
SPQ-Decomposition



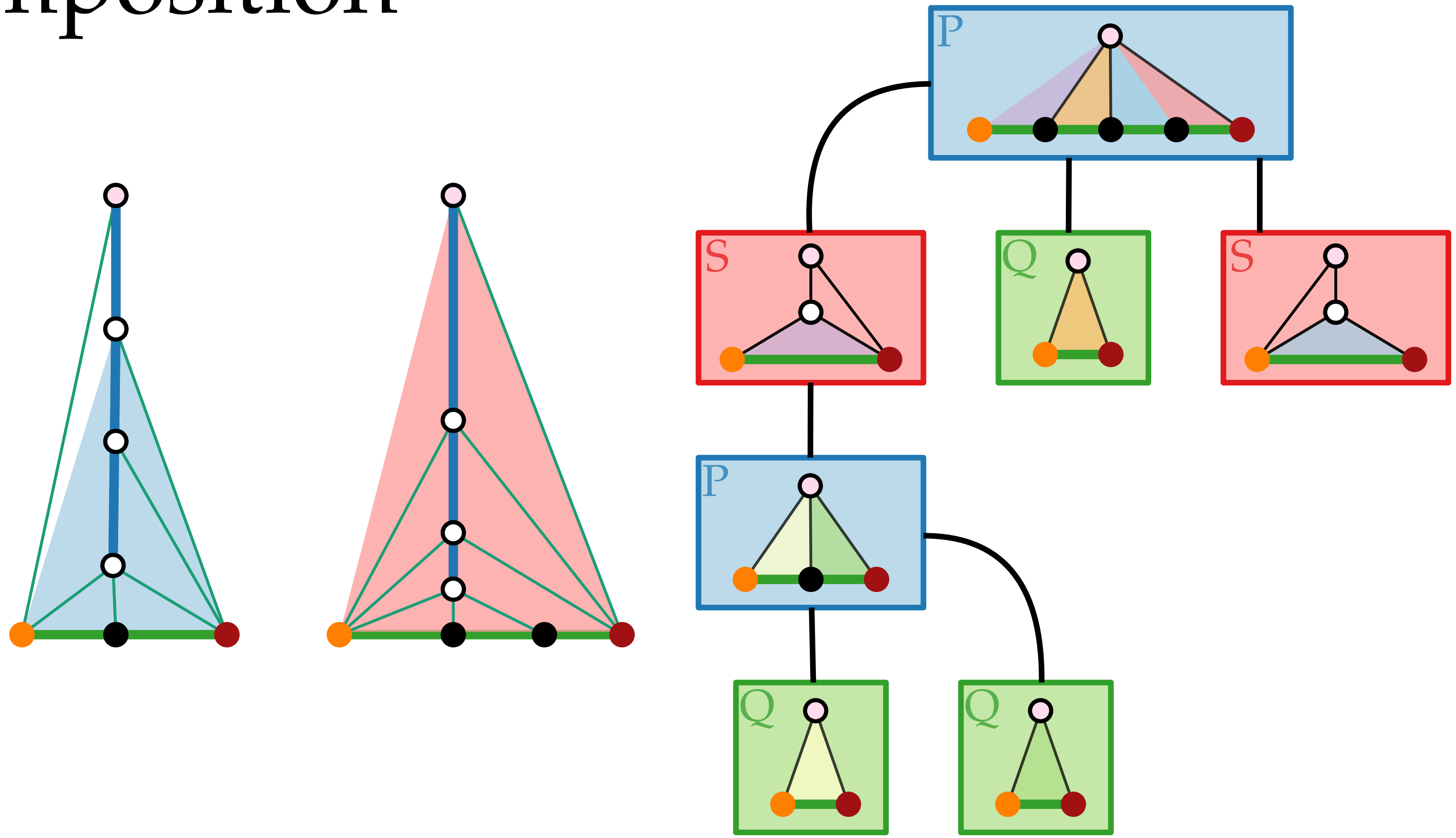
SPQ-Decomposition



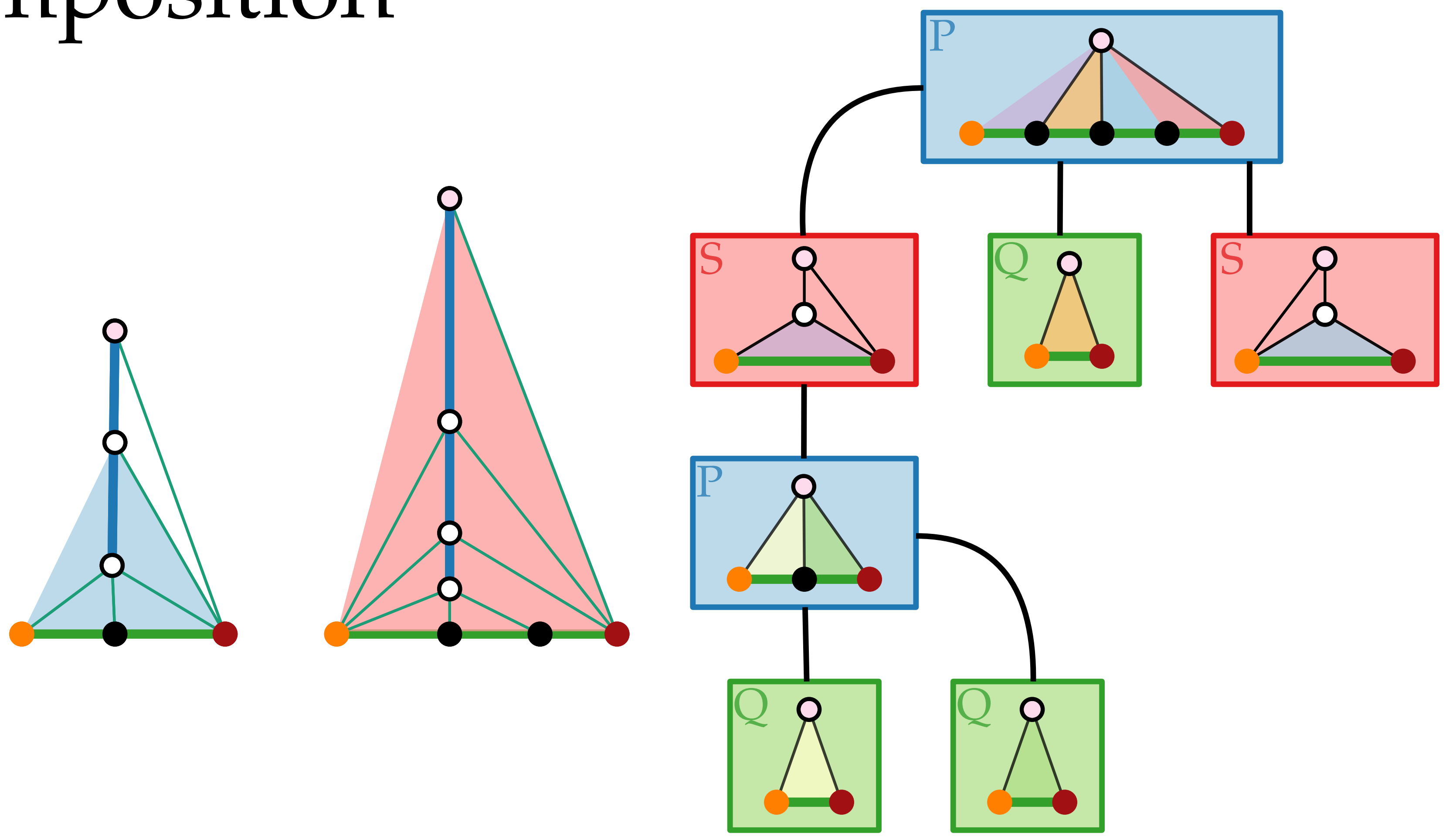
SPQ-Decomposition



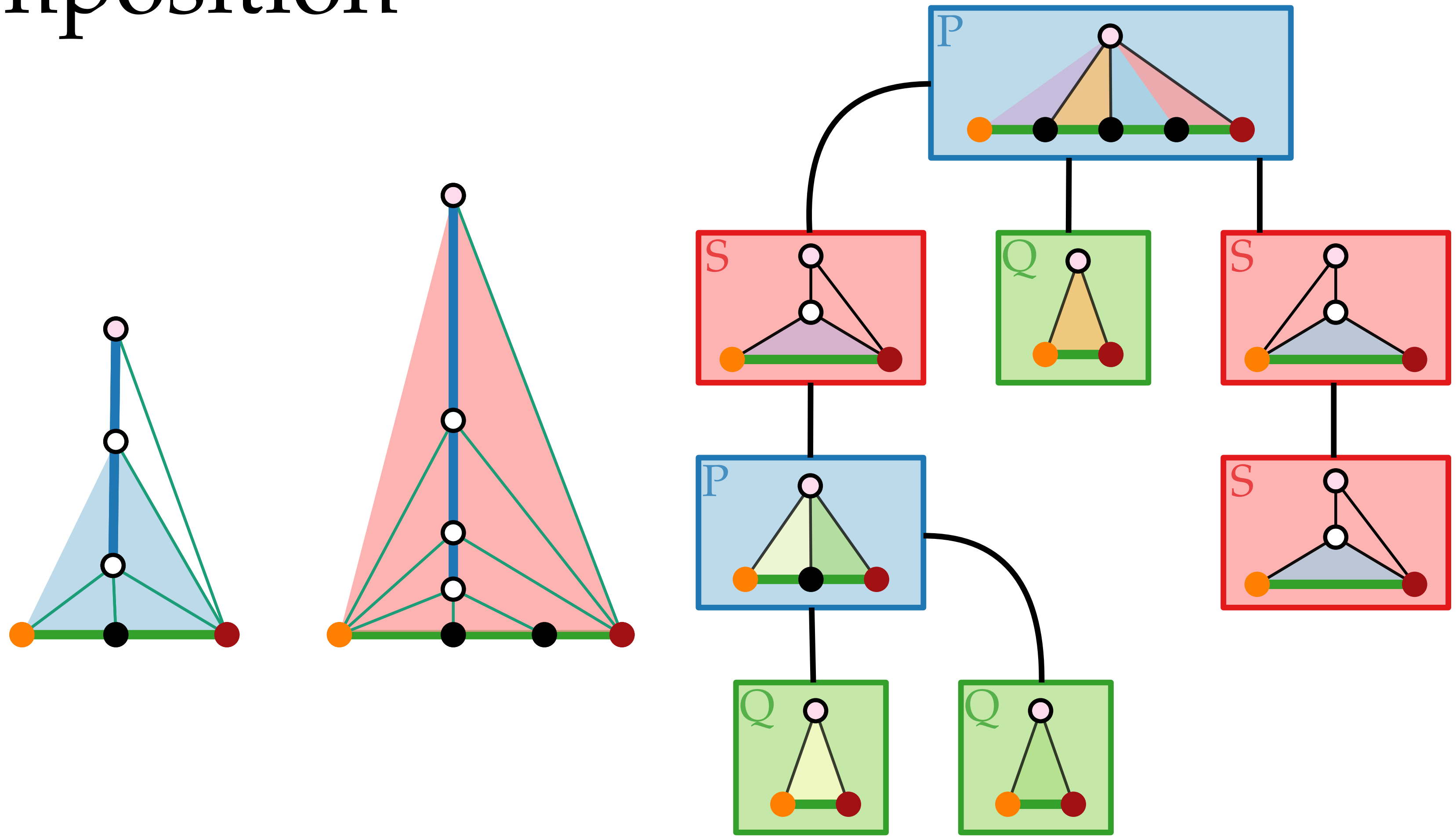
SPQ-Decomposition



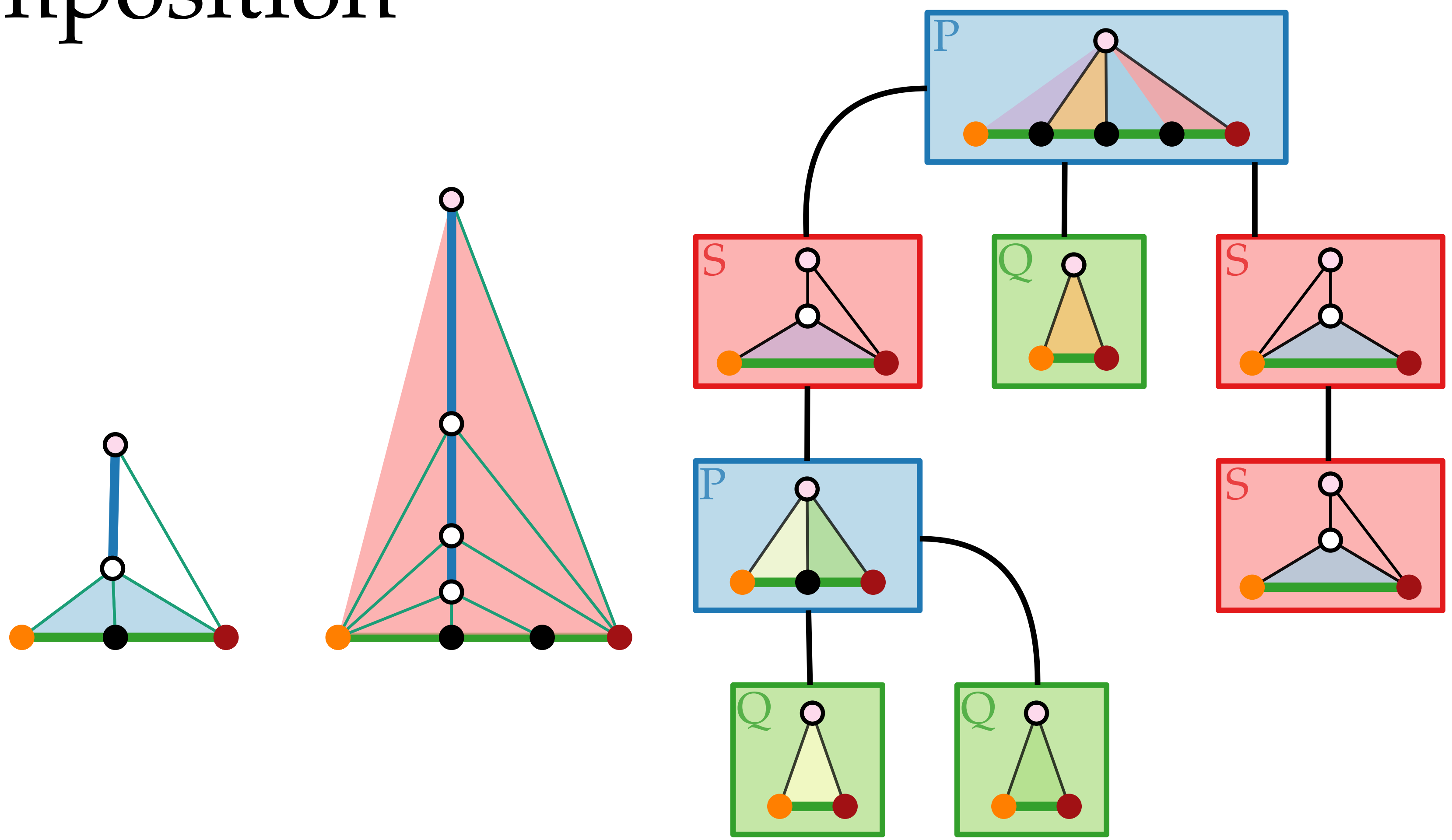
SPQ-Decomposition



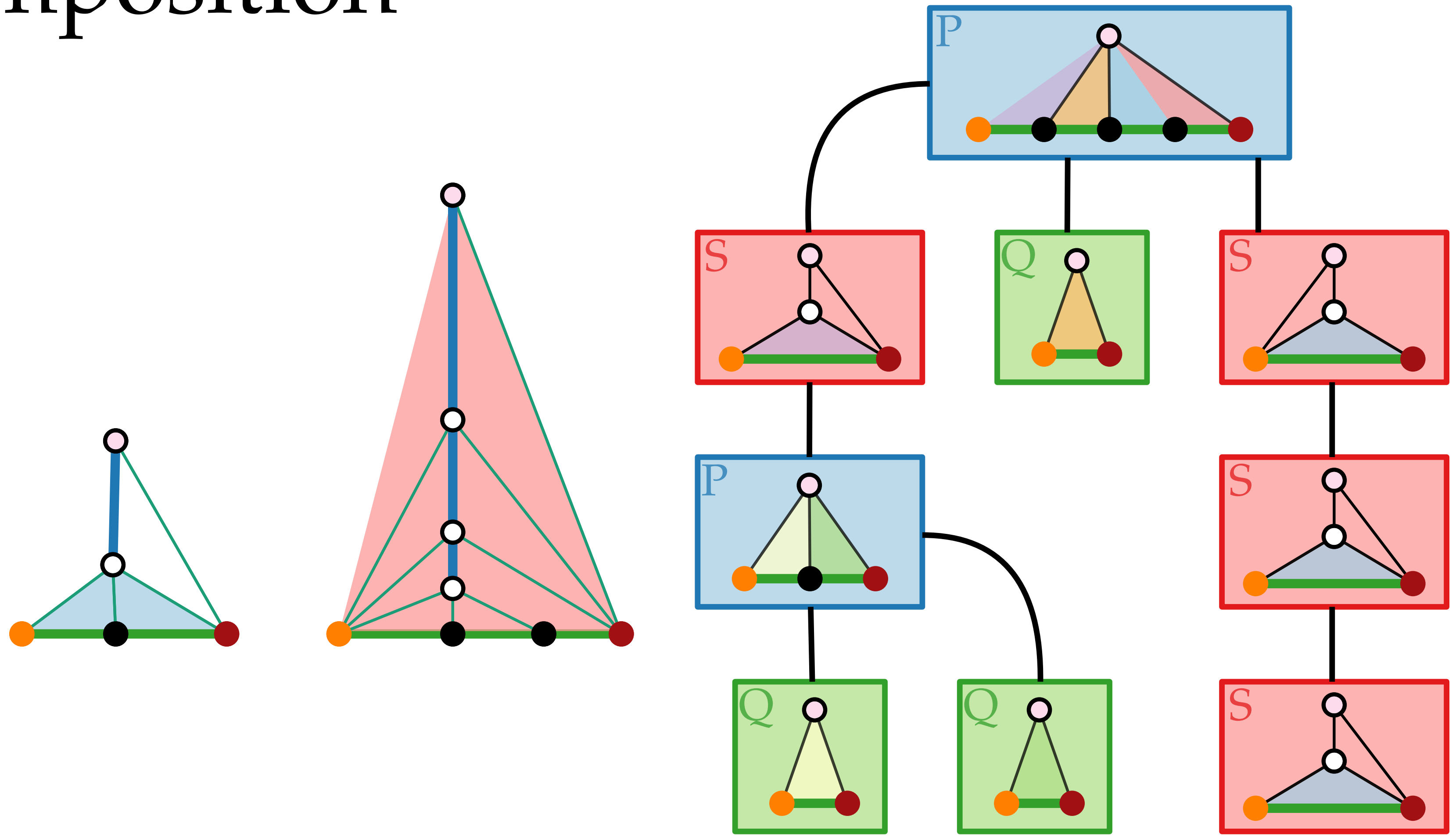
SPQ-Decomposition



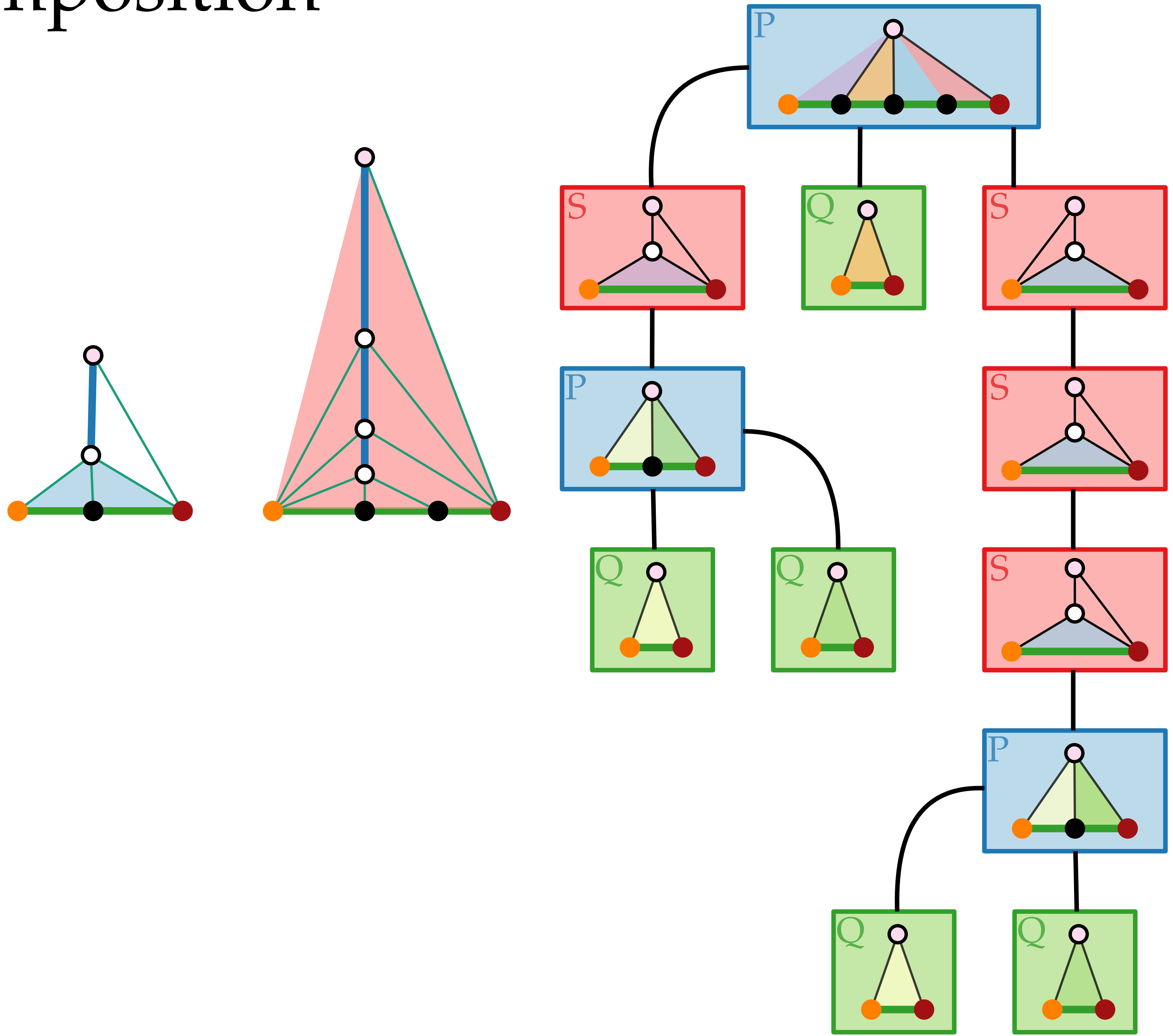
SPQ-Decomposition



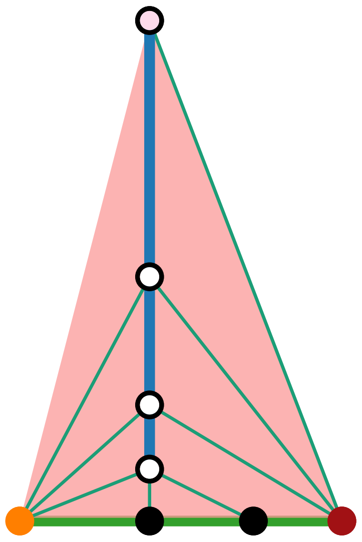
SPQ-Decomposition

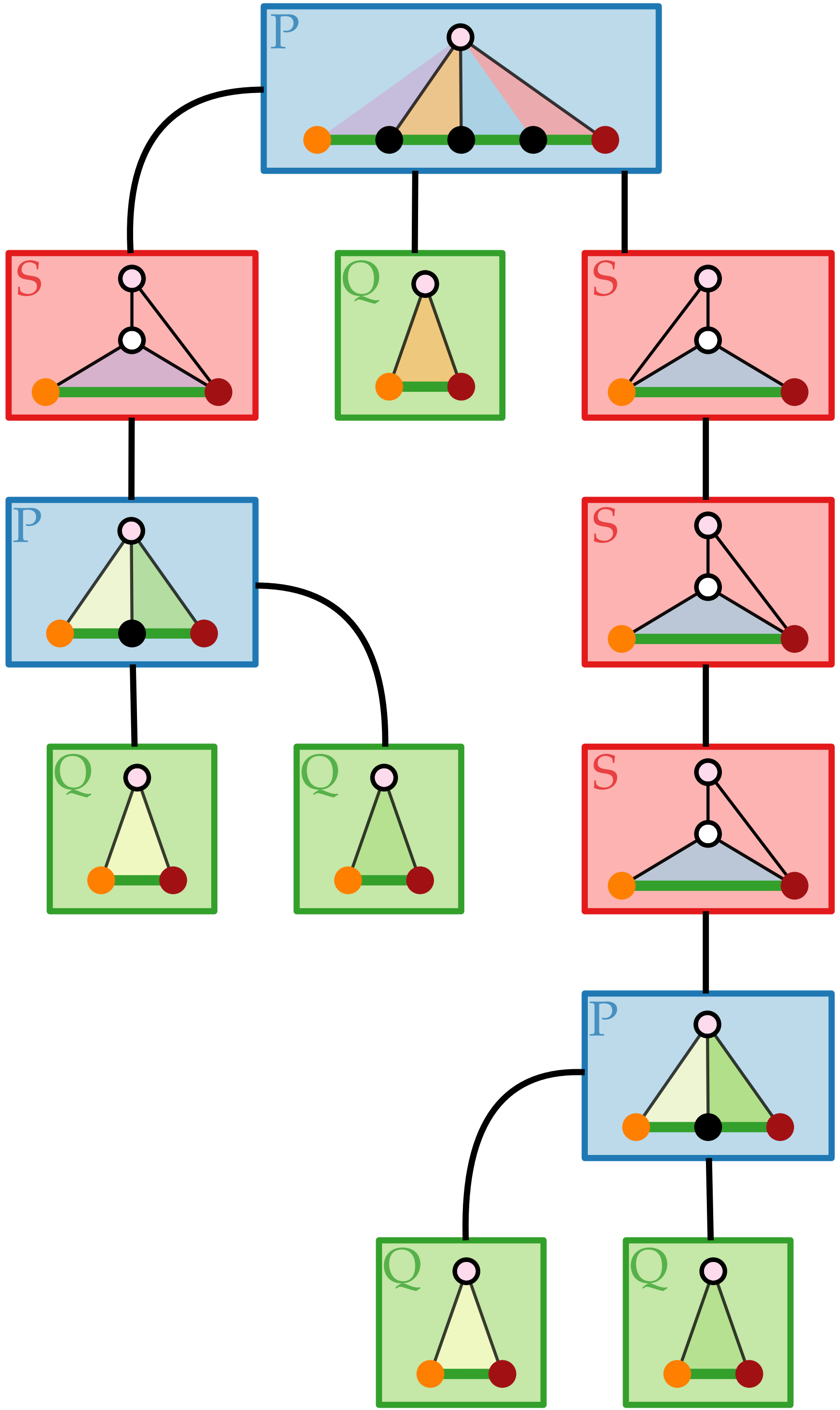


SPQ-Decomposition

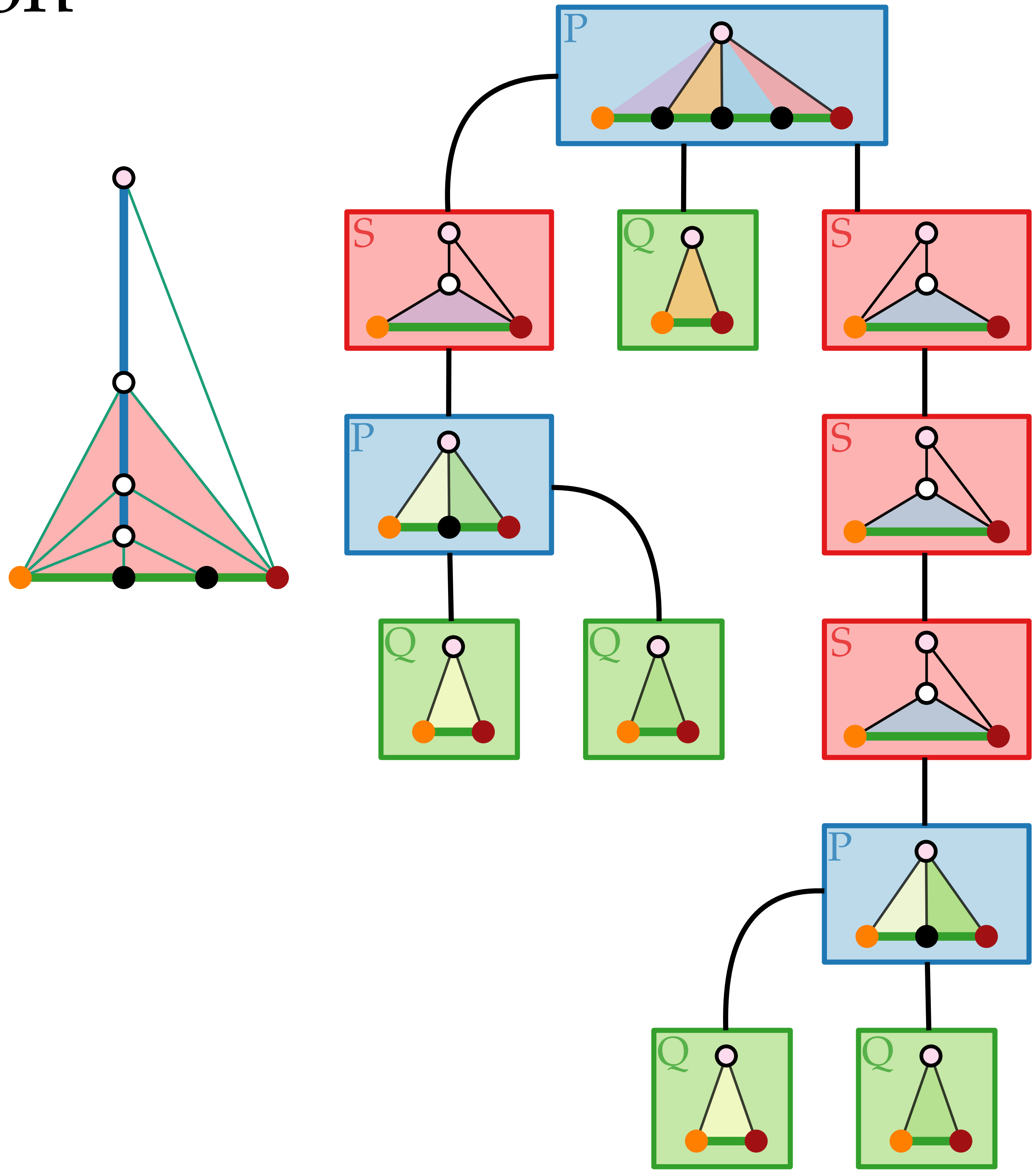


SPQ-Decomposition

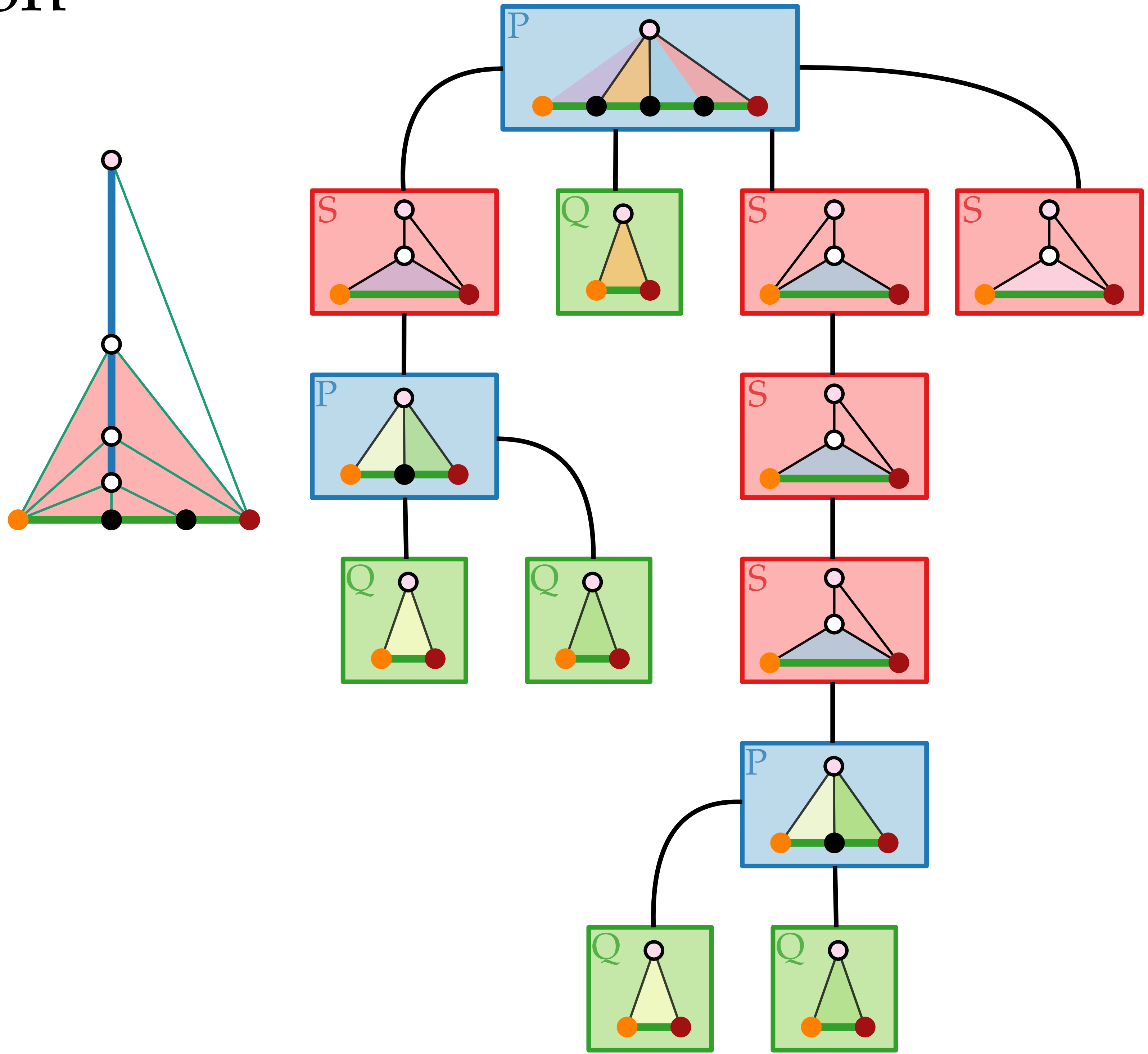




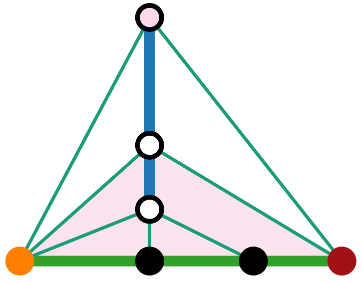
SPQ-Decomposition

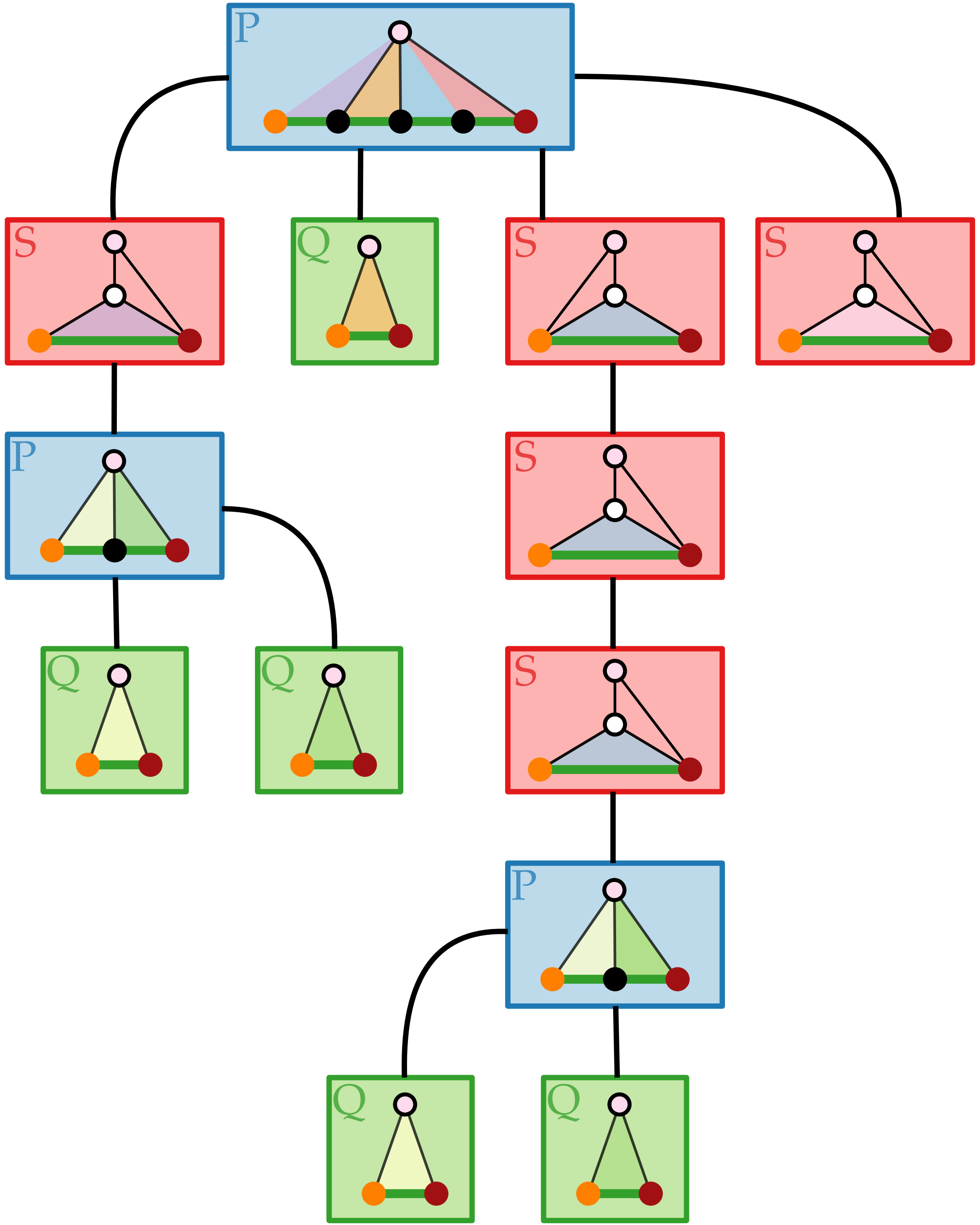


SPQ-Decomposition

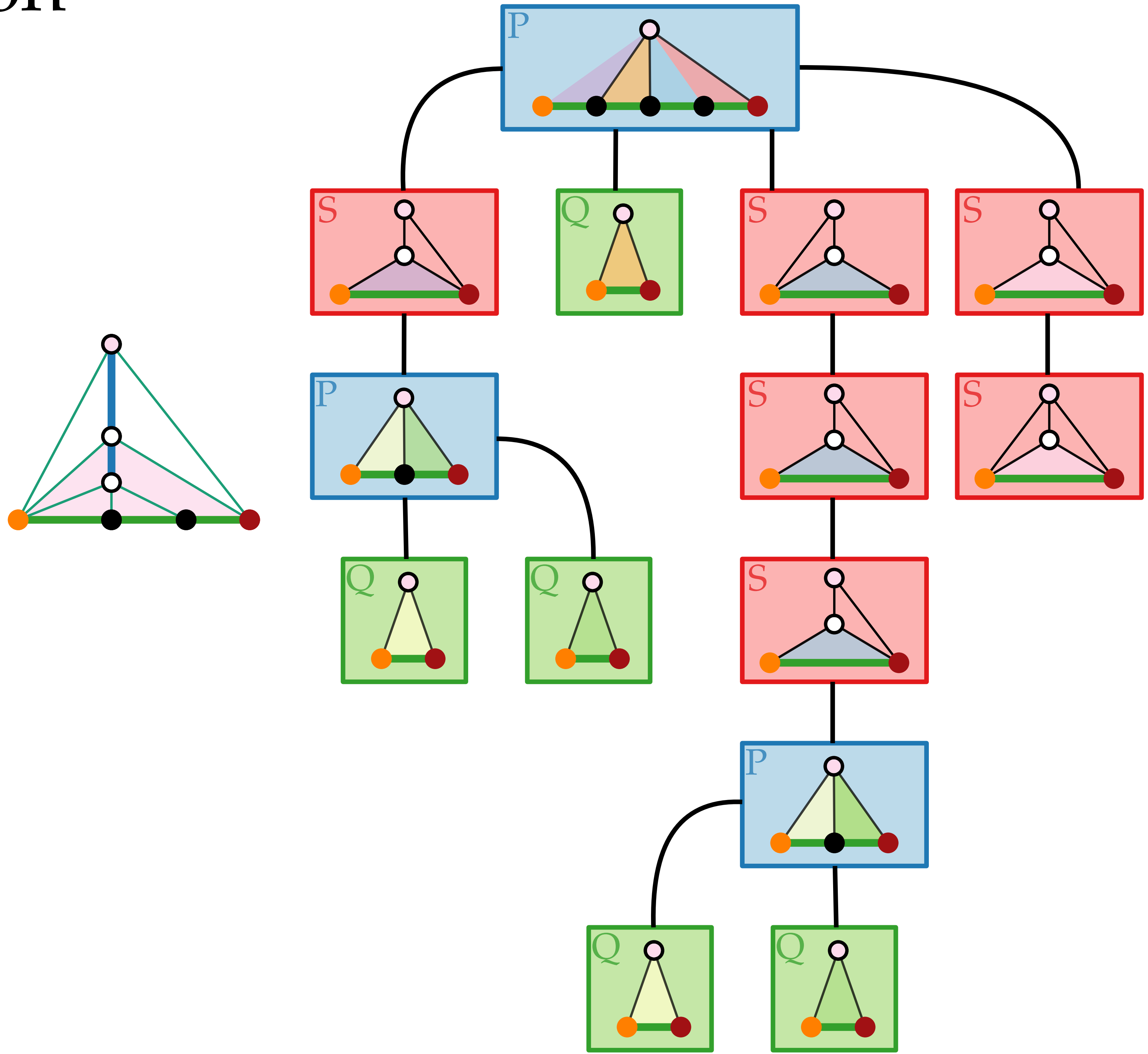


SPQ-Decomposition

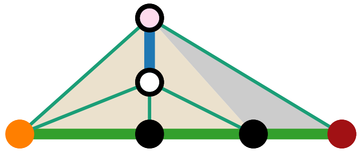


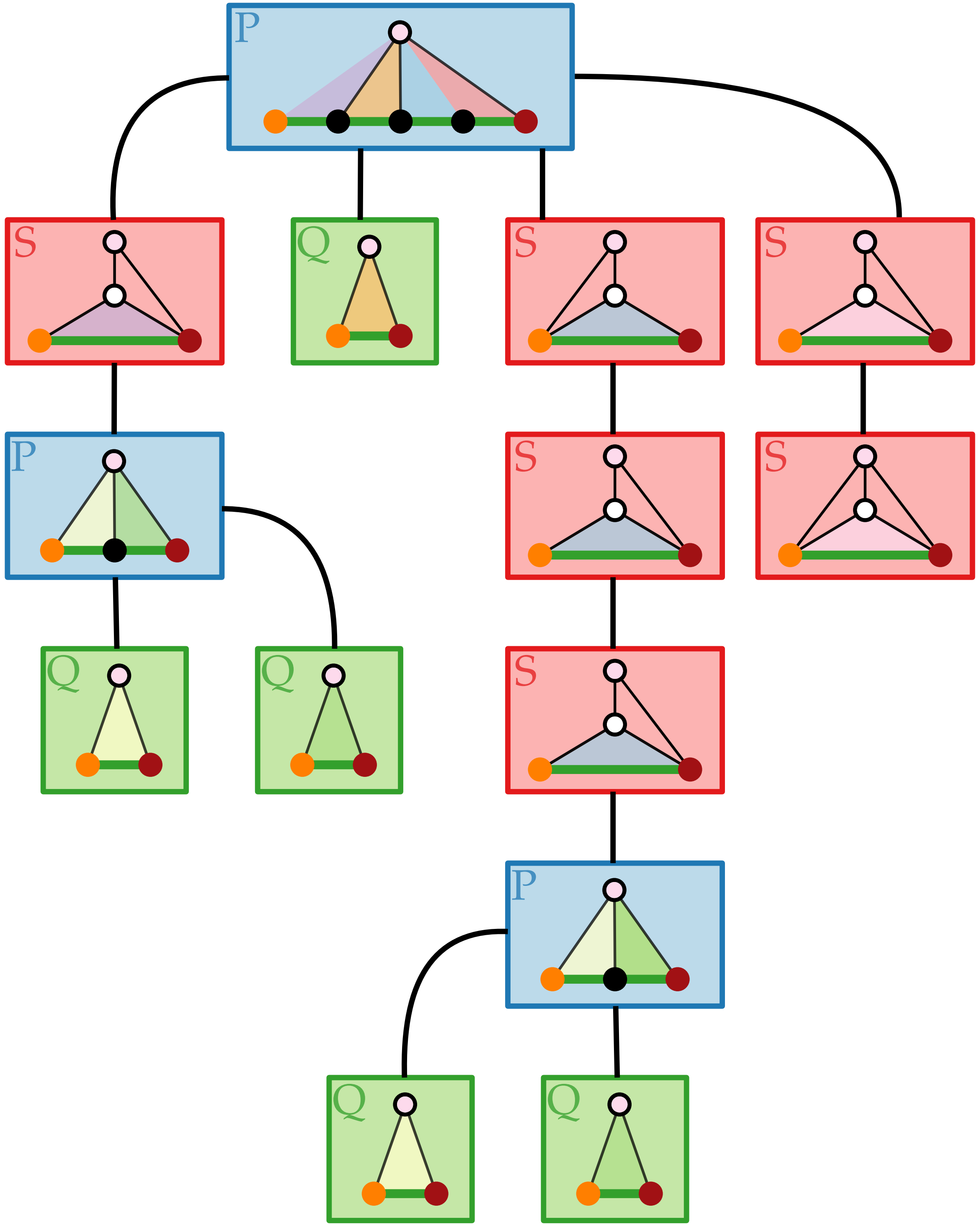


SPQ-Decomposition

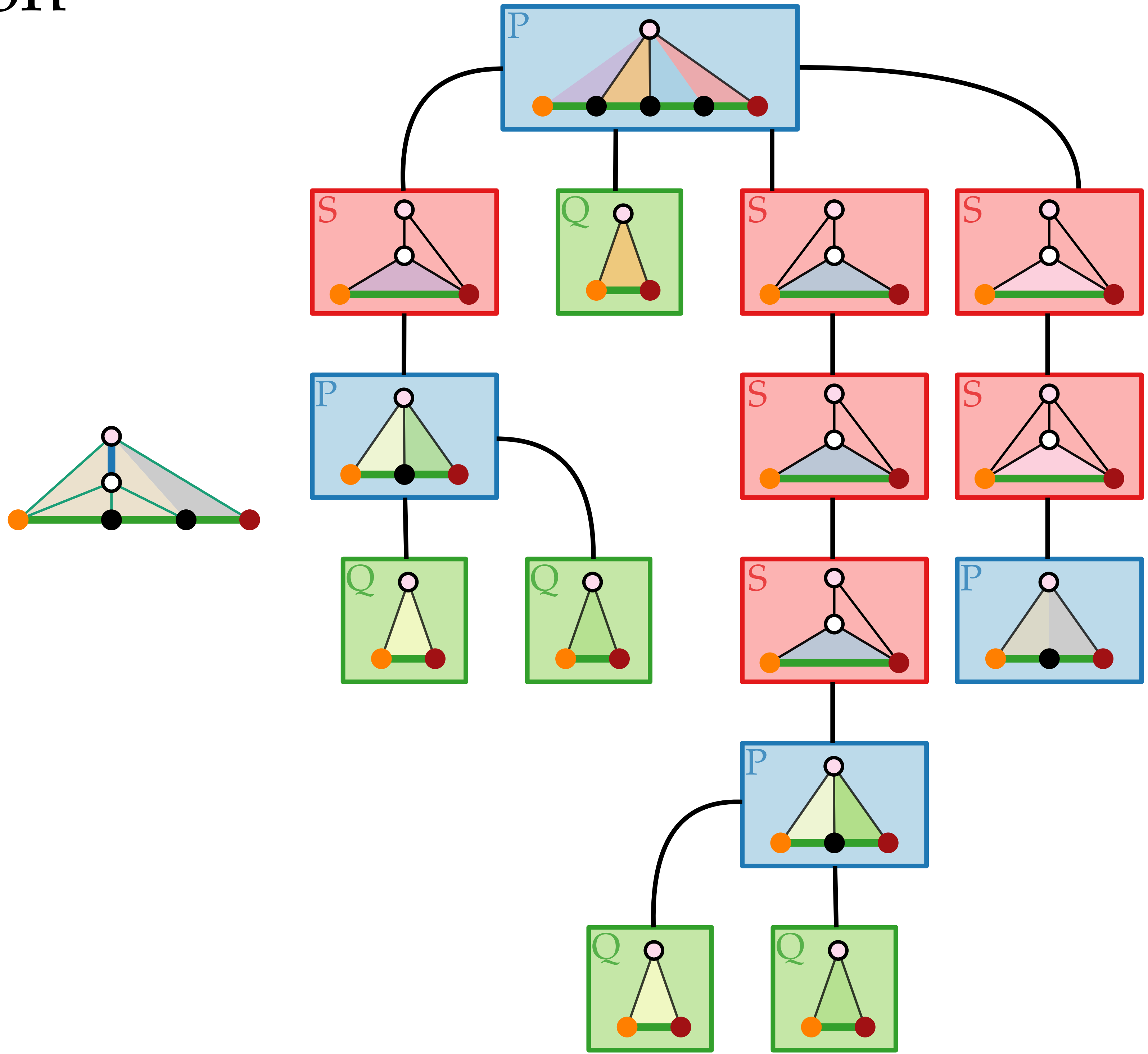


SPQ-Decomposition

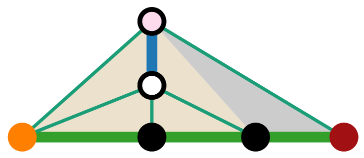


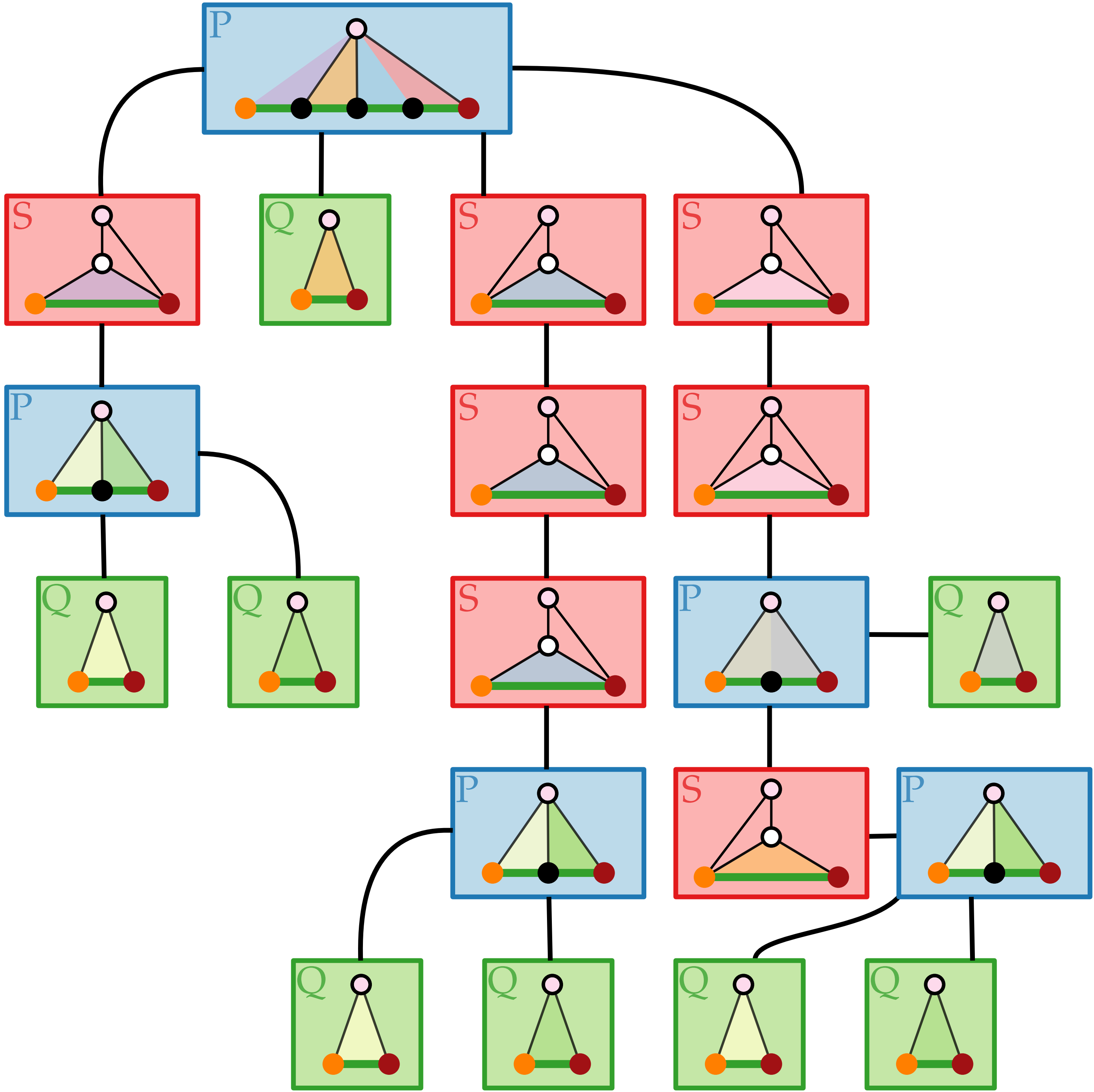


SPQ-Decomposition

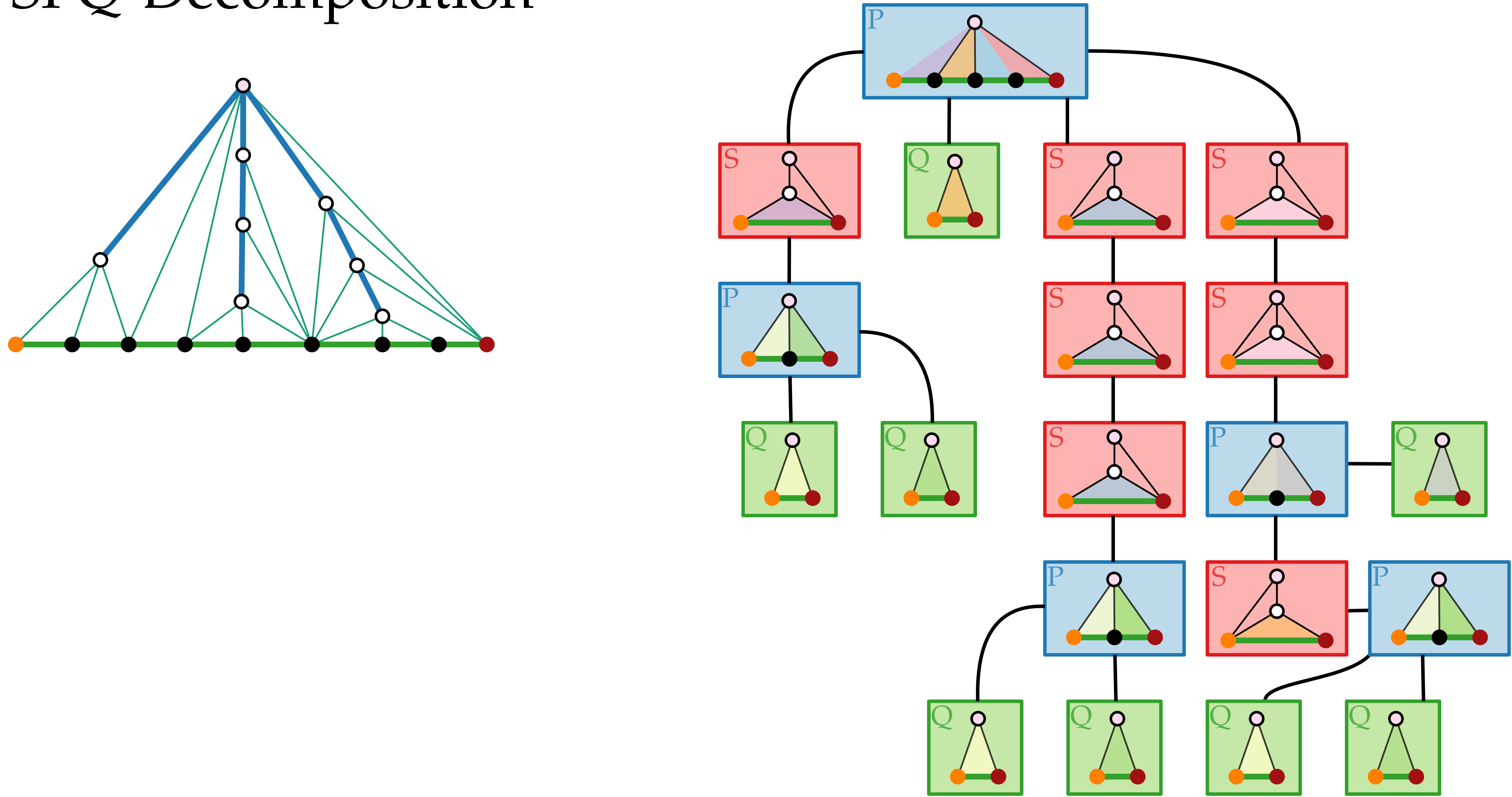


SPQ-Decomposition

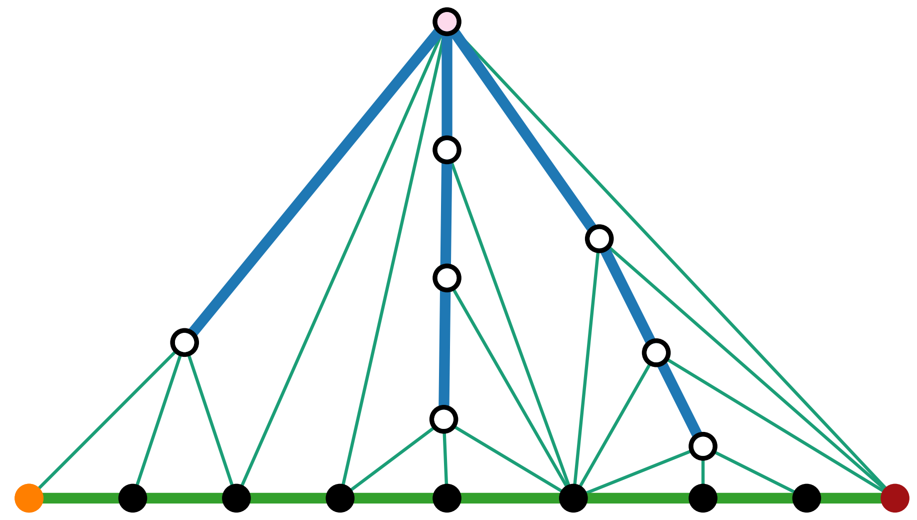


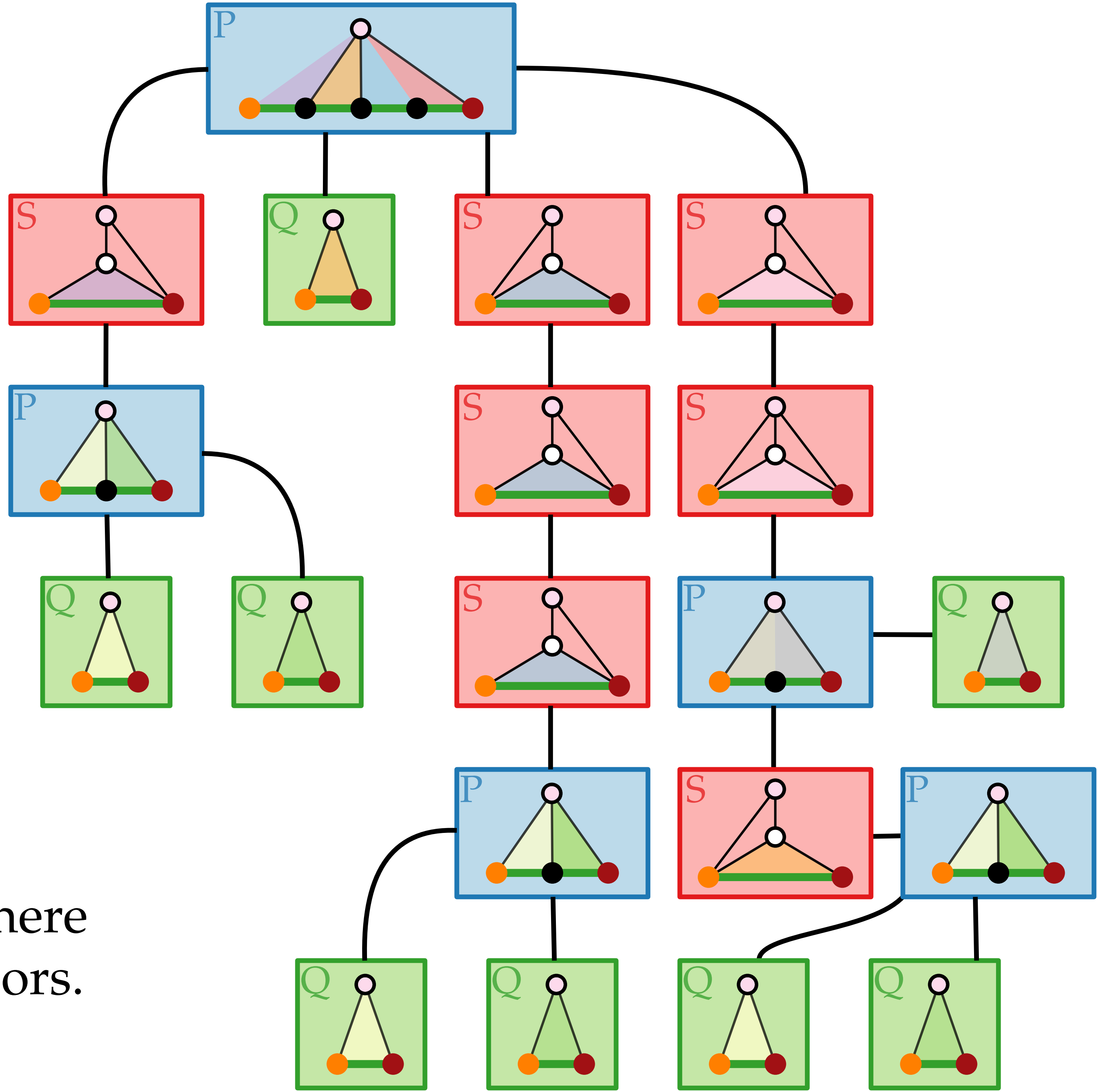


SPQ-Decomposition



SPQ-Decomposition





There is always an SPQ-decomposition where no two nodes of the same type are neighbors.

Shapes

- Always use one of 6 shapes

Shapes

- Always use one of 6 shapes

Shapes

- Always use one of 6 shapes

$(1,1)$-flat

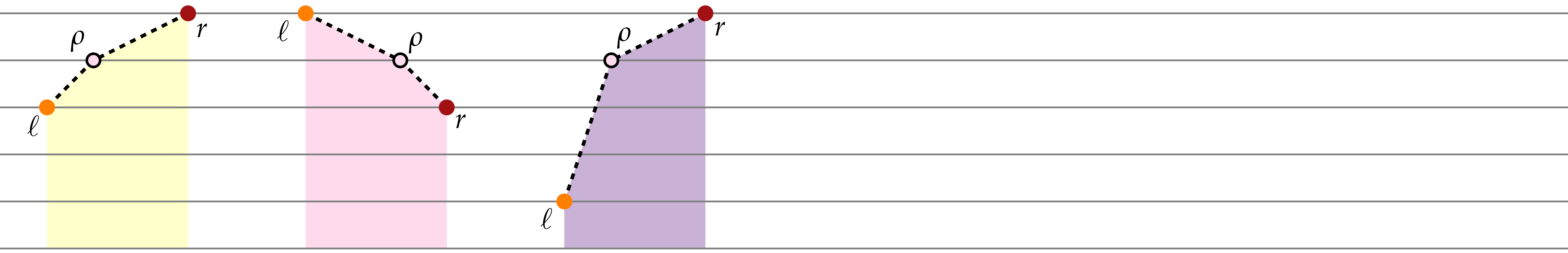
Shapes

- Always use one of 6 shapes

$(1, 1)$-flat $(-1, -1)$-flat

Shapes

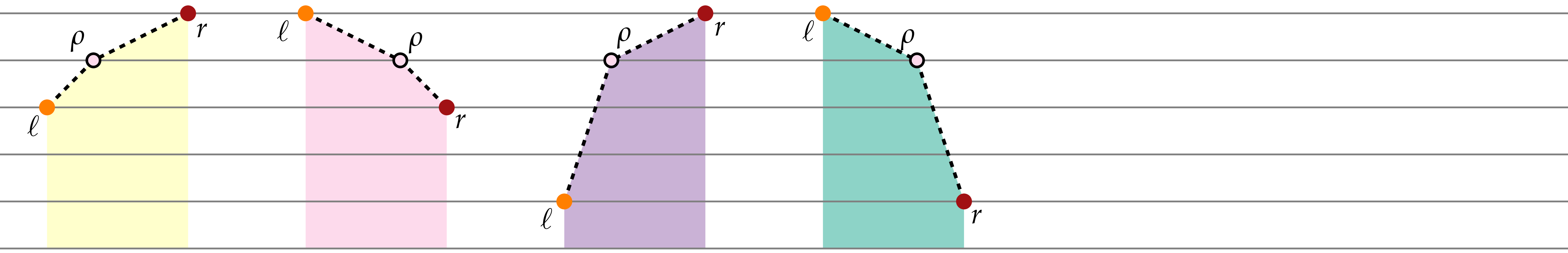
- Always use one of 6 shapes



$(1, 1)$-flat $(-1, -1)$-flat $(3, 1)$-flat

Shapes

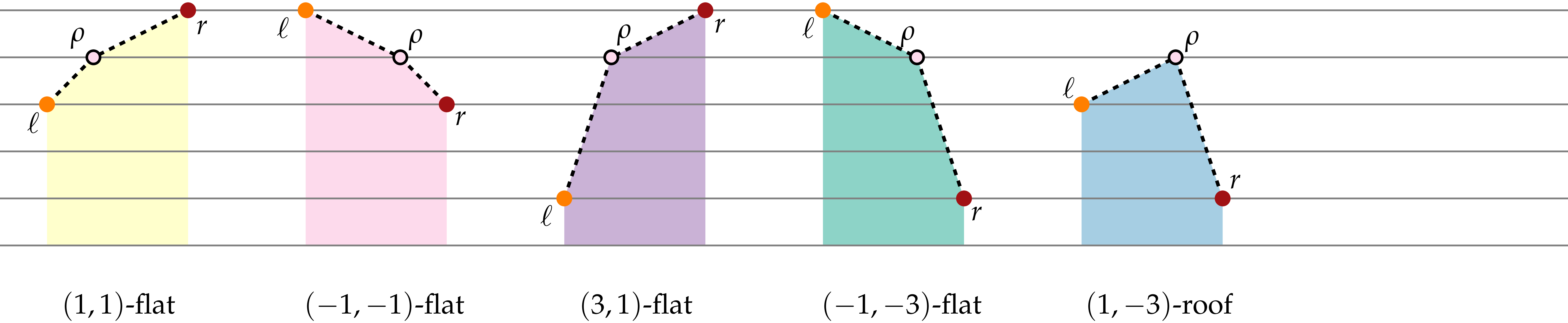
- Always use one of 6 shapes



$(1,1)$-flat $(-1,-1)$-flat $(3,1)$-flat $(-1,-3)$-flat

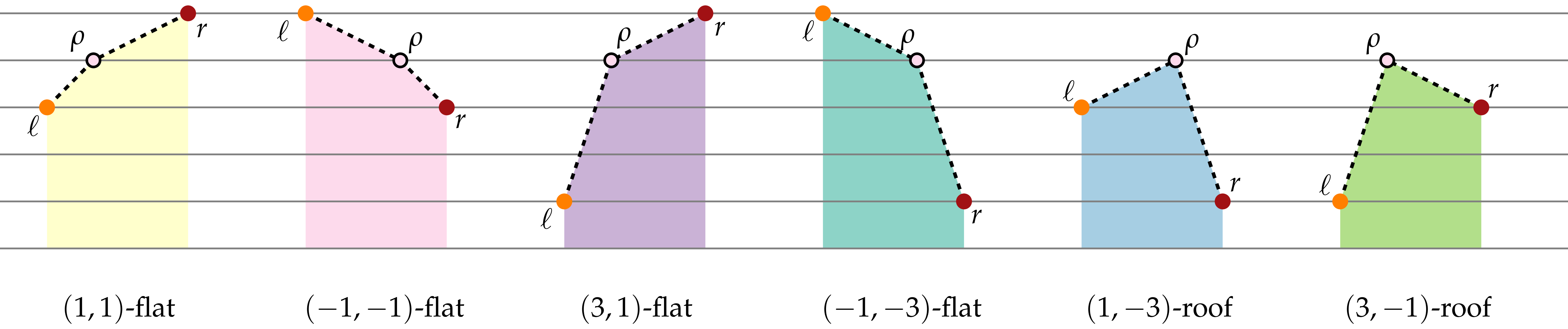
Shapes

- Always use one of 6 shapes



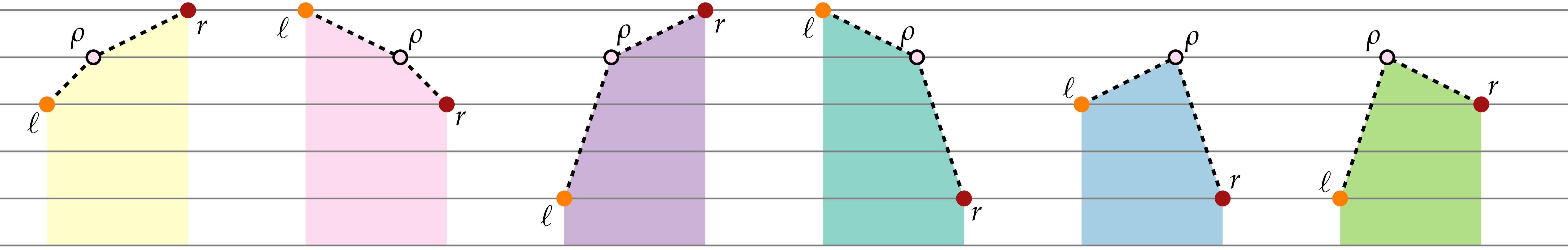
Shapes

- Always use one of 6 shapes



Shapes

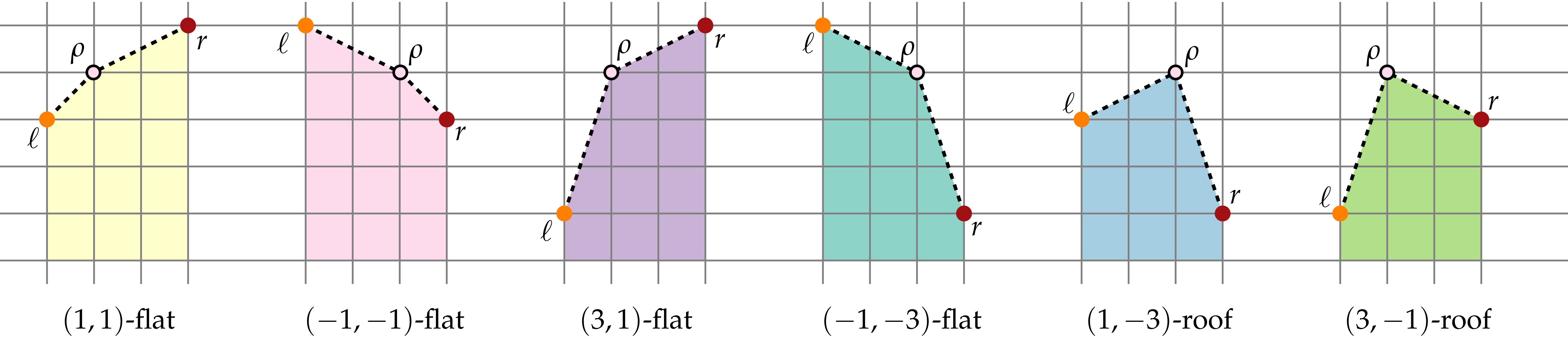
- Always use one of 6 shapes
- Draw with 1 bend per edge on the grid



$(1, 1)$-flat $(-1, -1)$-flat $(3, 1)$-flat $(-1, -3)$-flat $(1, -3)$-roof $(3, -1)$-roof

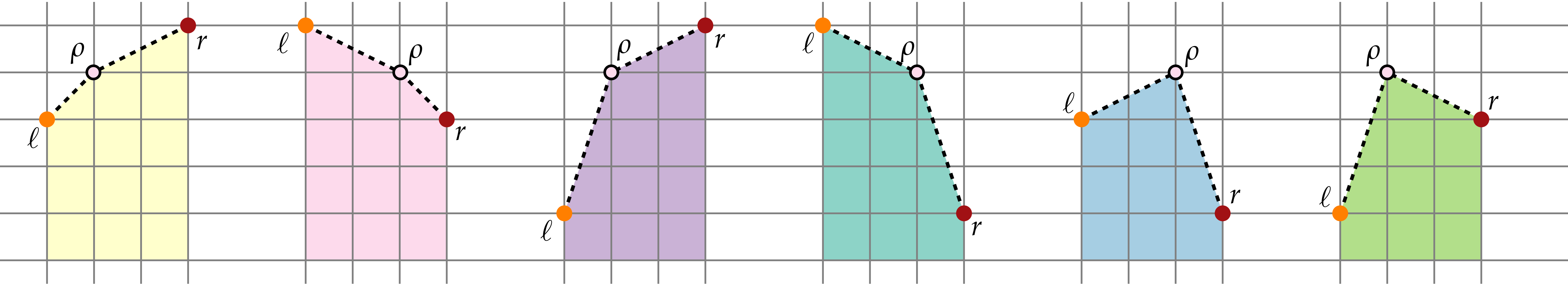
Shapes

- Always use one of 6 shapes
- Draw with 1 bend per edge on the grid



Shapes

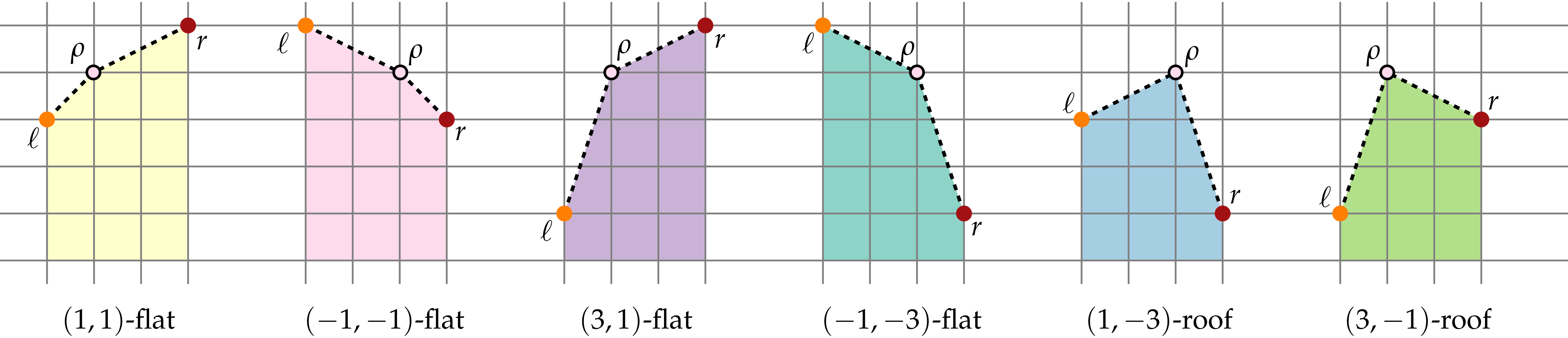
- Always use one of 6 shapes
- Draw with 1 bend per edge on the grid
- P-nodes can only use flat shapes



$(1,1)$-flat $(-1,-1)$-flat $(3,1)$-flat $(-1,-3)$-flat $(1,-3)$-roof $(3,-1)$-roof

Shapes

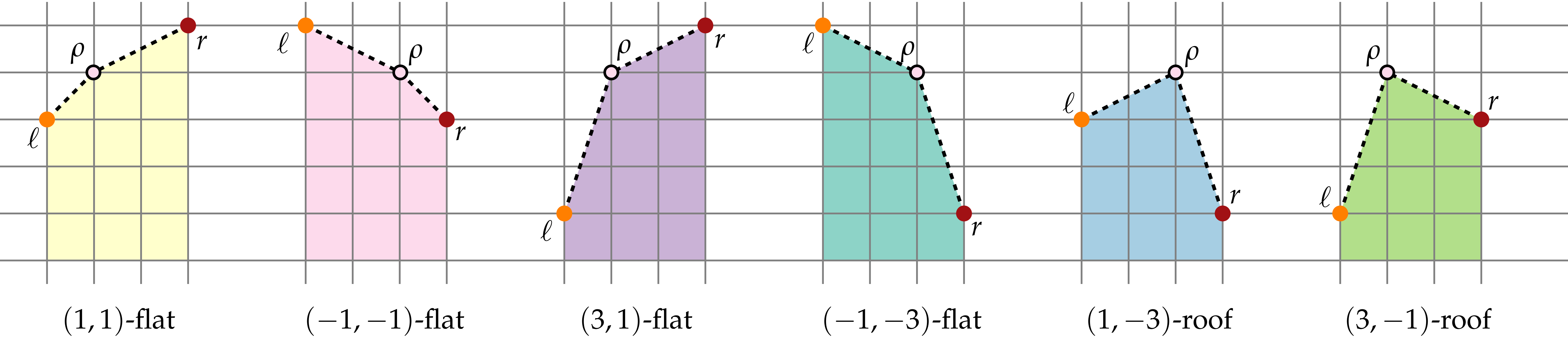
- Always use one of 6 shapes
- Draw with 1 bend per edge on the grid
- P-nodes can only use flat shapes
- Create a drawing for every possible shape ➡ can combine what we need



Shapes

- Always use one of 6 shapes
- Draw with 1 bend per edge on the grid
- P-nodes can only use flat shapes
- Create a drawing for every possible shape ➡ can combine what we need

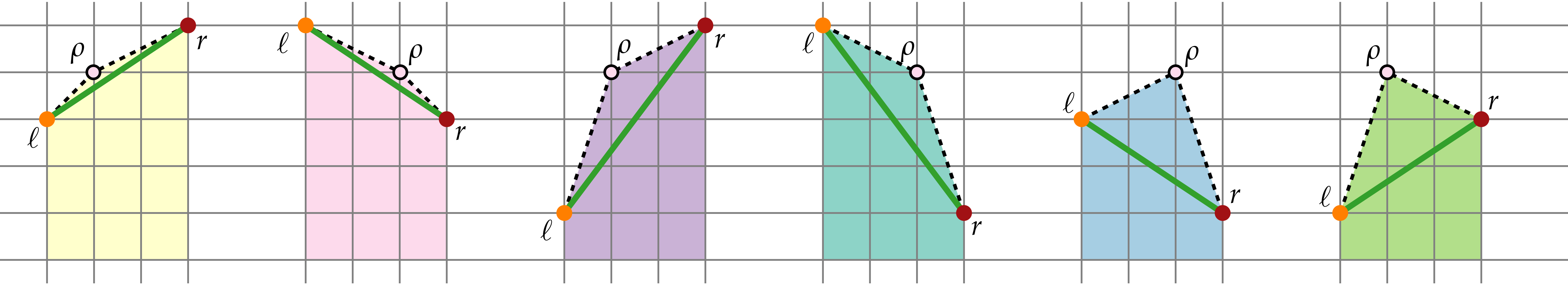
Q-nodes:



Shapes

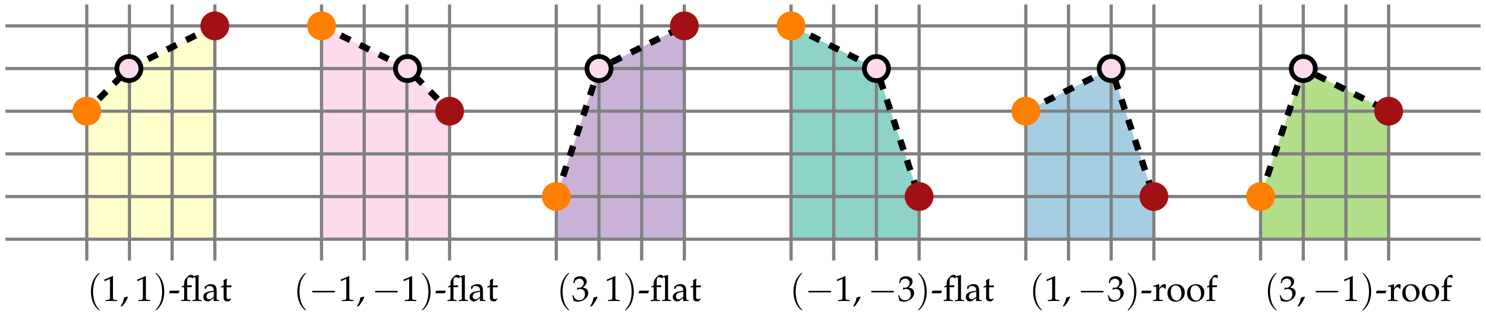
- Always use one of 6 shapes
- Draw with 1 bend per edge on the grid
- P-nodes can only use flat shapes
- Create a drawing for every possible shape ➡ can combine what we need

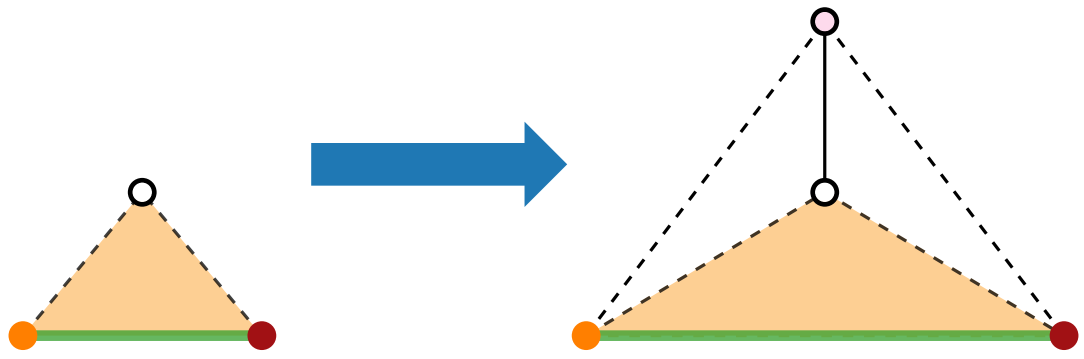
Q-nodes:



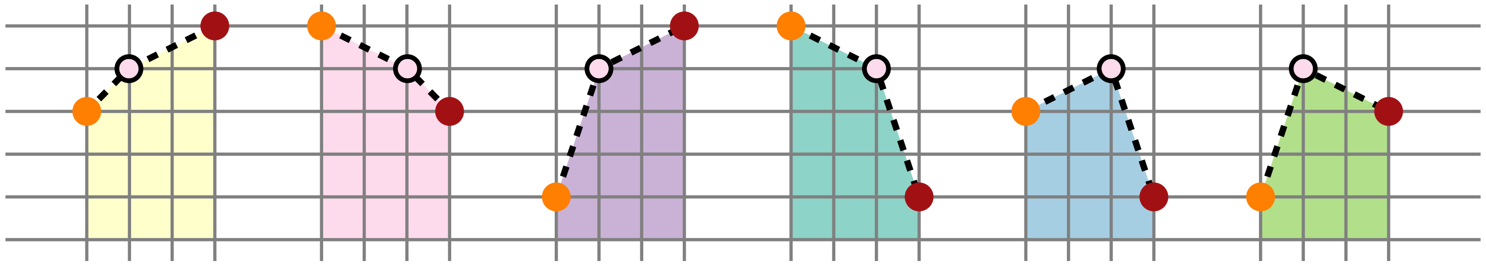
$(1,1)$-flat $(-1,-1)$-flat $(3,1)$-flat $(-1,-3)$-flat $(1,-3)$-roof $(3,-1)$-roof

S-nodes

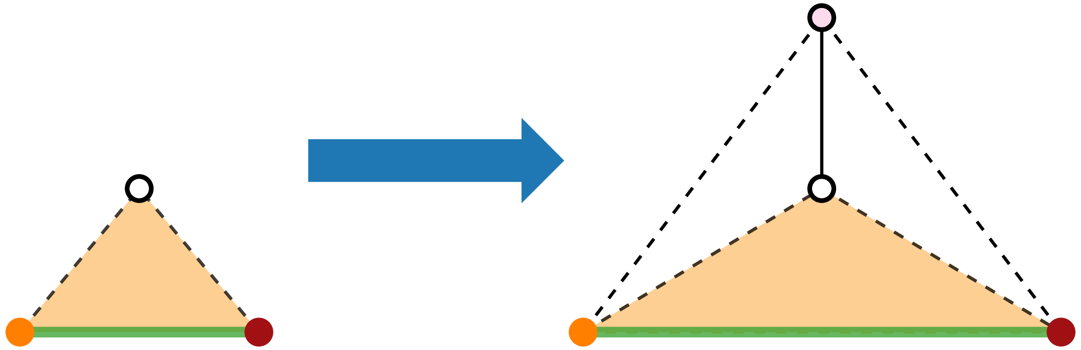




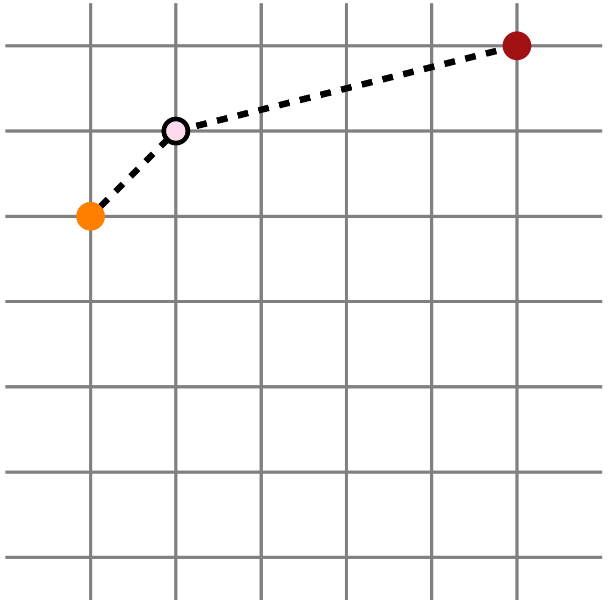
S-nodes



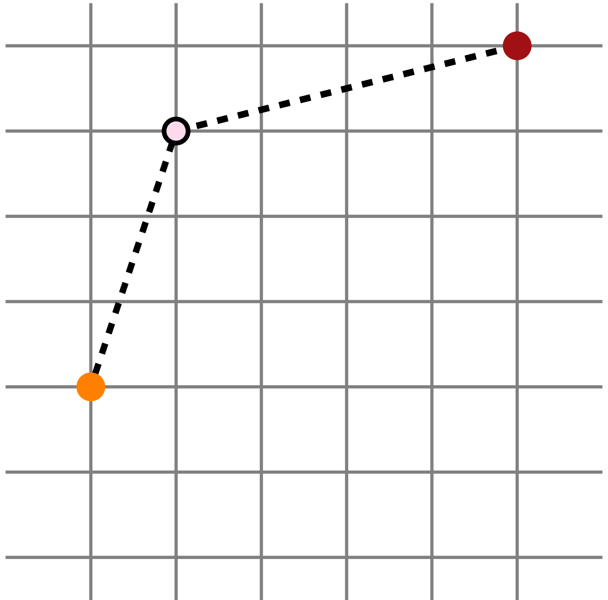
$(1,1)$-flat $(-1,-1)$-flat $(3,1)$-flat $(-1,-3)$-flat $(1,-3)$-roof $(3,-1)$-roof



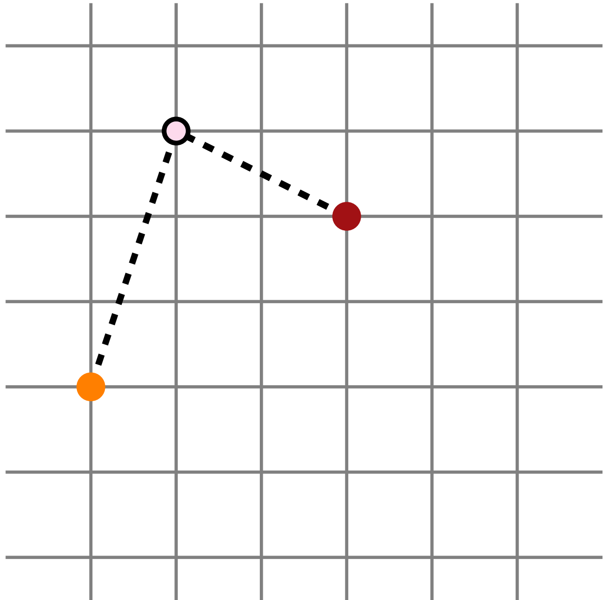
$(1,1)$-flat



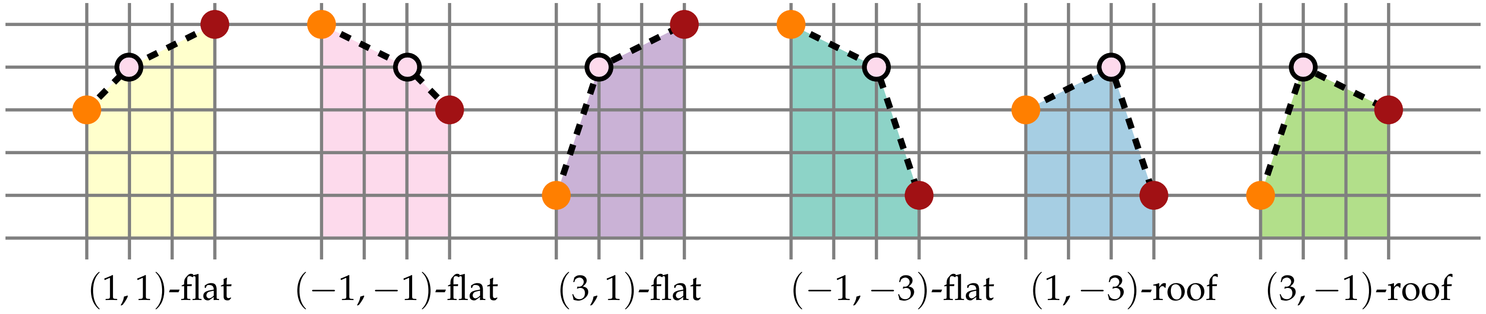
$(3,1)$-flat

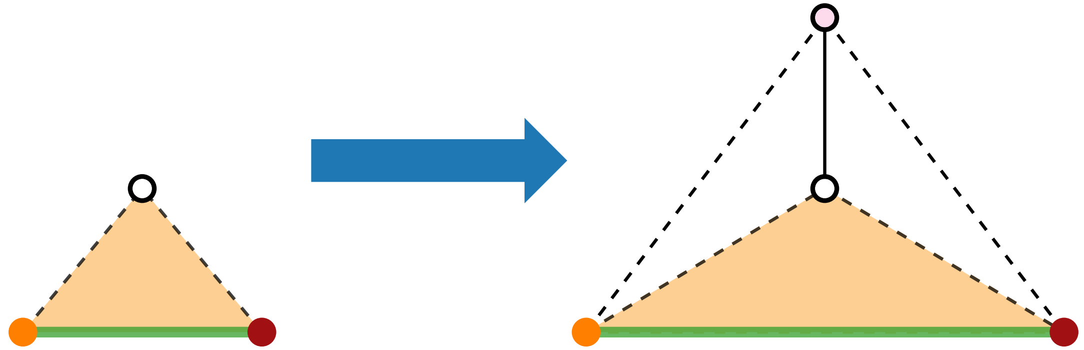


$(3,-1)$-roof

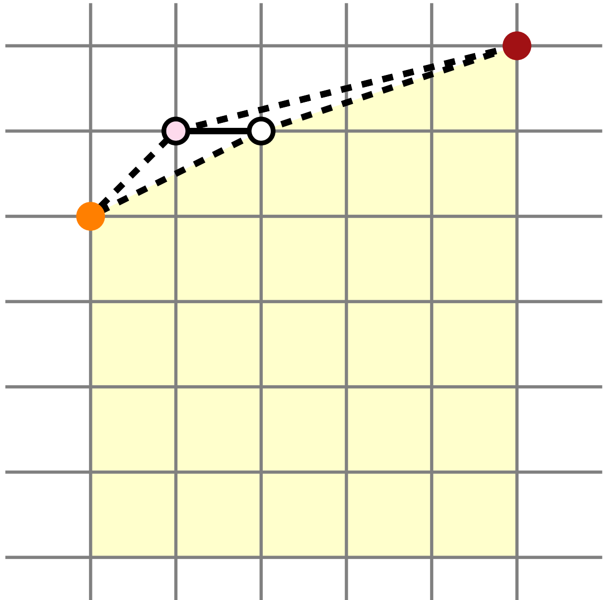


S-nodes

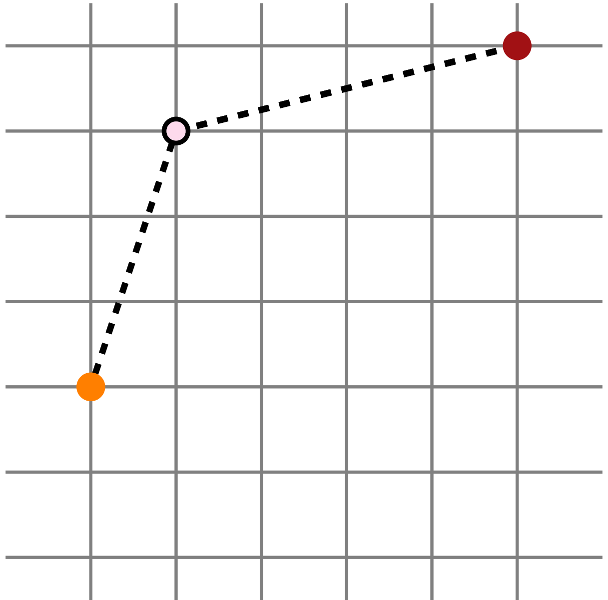




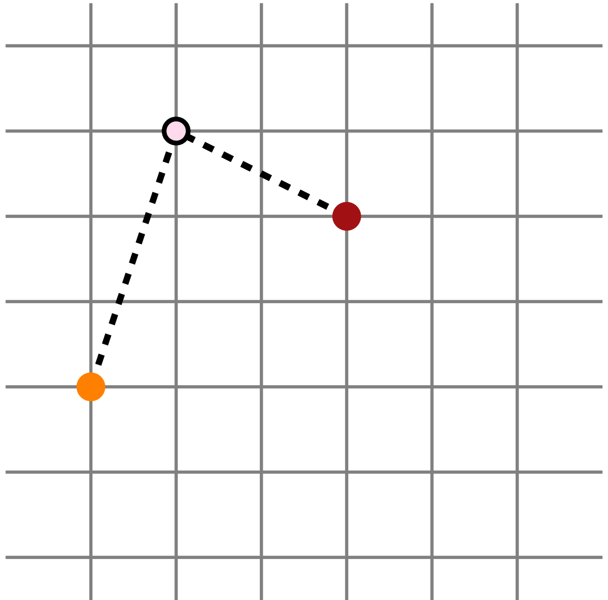
(1, 1)-flat



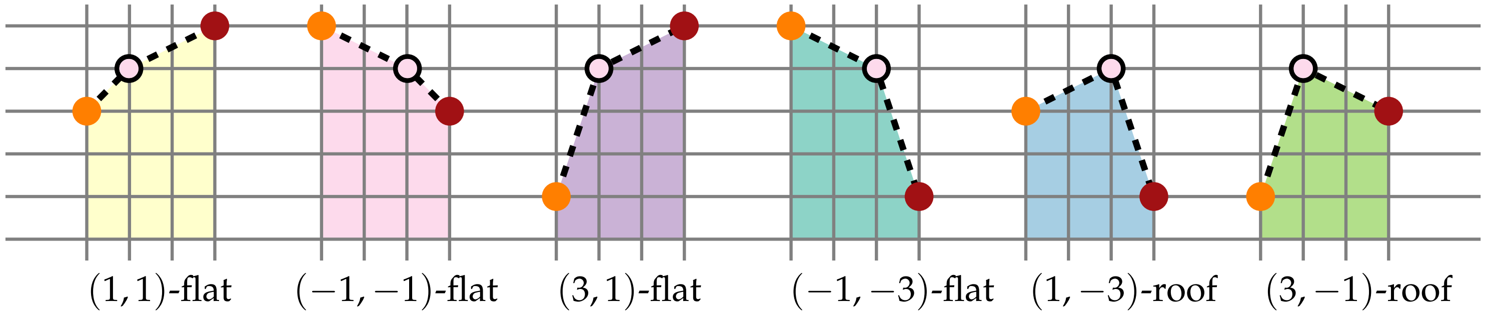
(3, 1)-flat



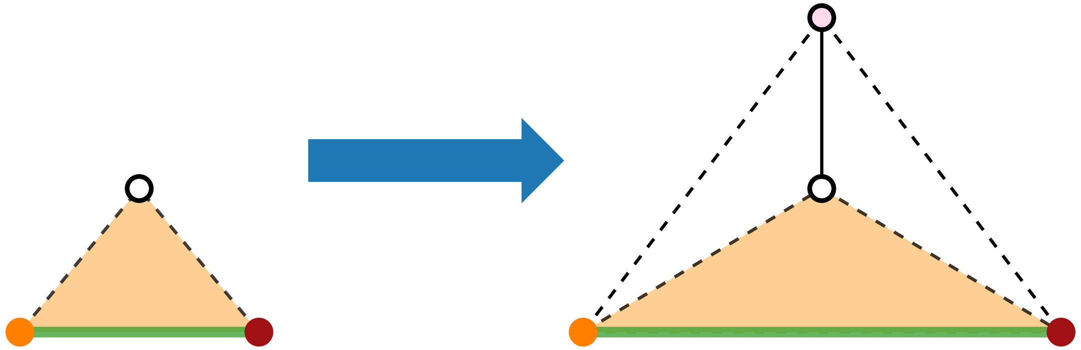
(3, −1)-roof

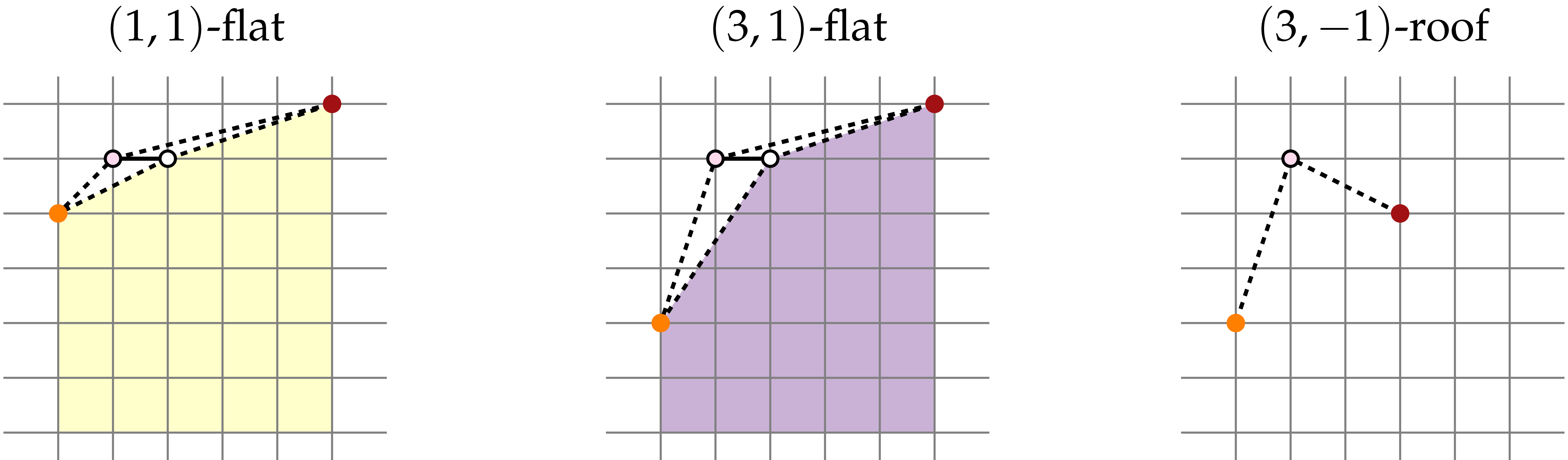


S-nodes



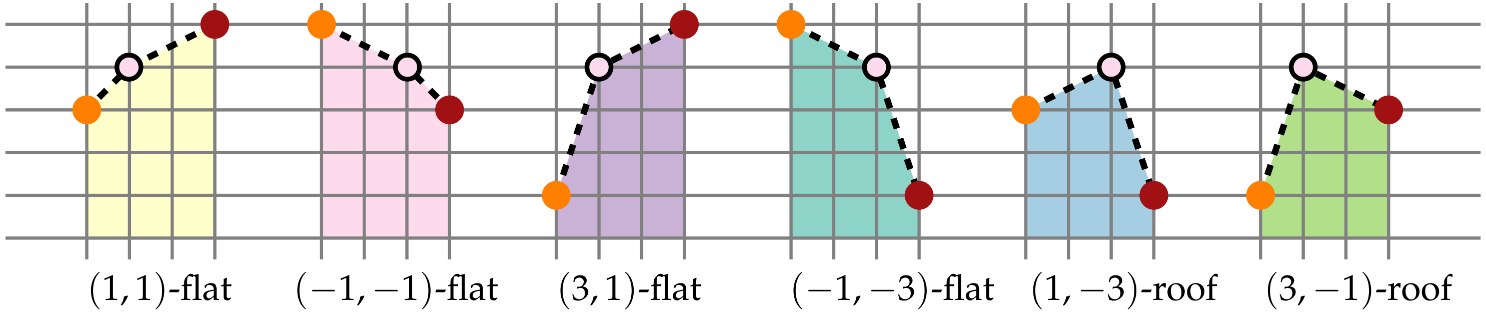
(1, 1)-flat
(−1, −1)-flat
(3, 1)-flat
(−1, −3)-flat
(1, −3)-roof
(3, −1)-roof

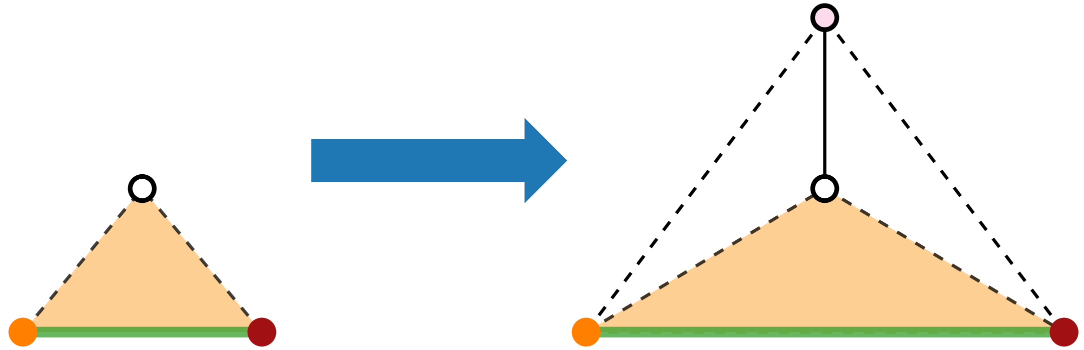




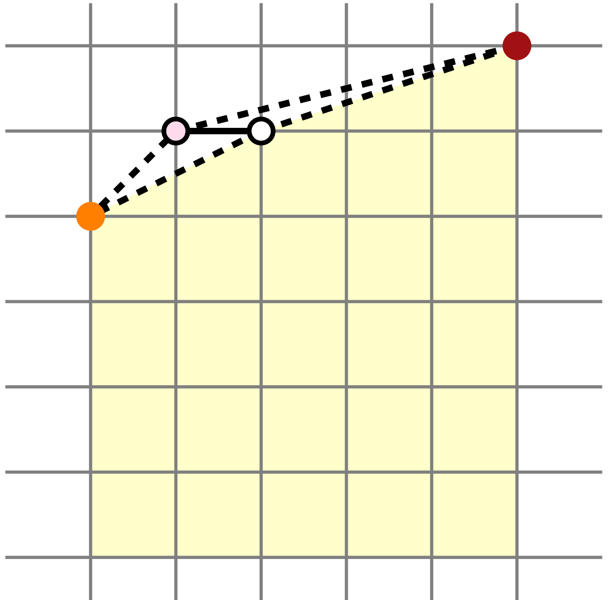
(1, 1)-flat
(3, 1)-flat
(3, −1)-roof

S-nodes

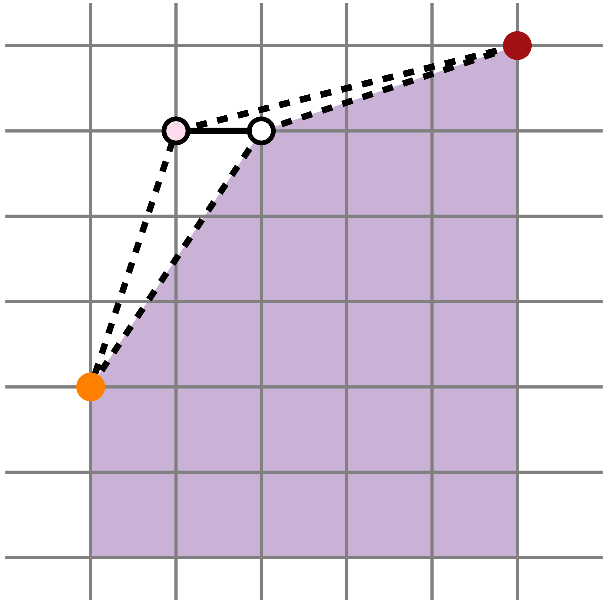




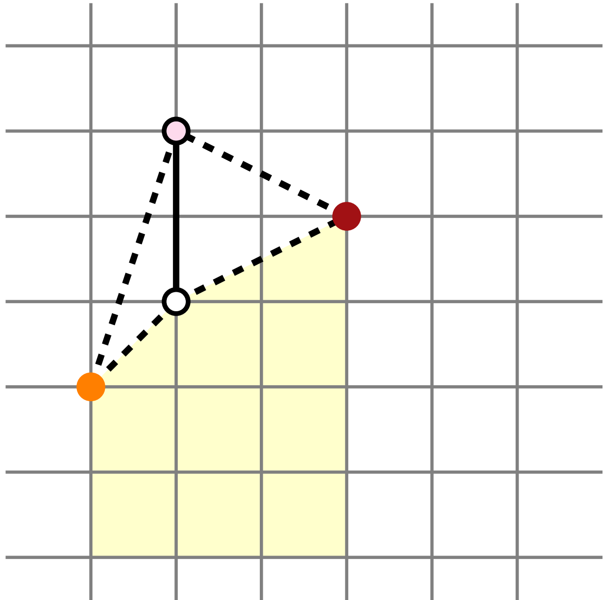
(1, 1)-flat



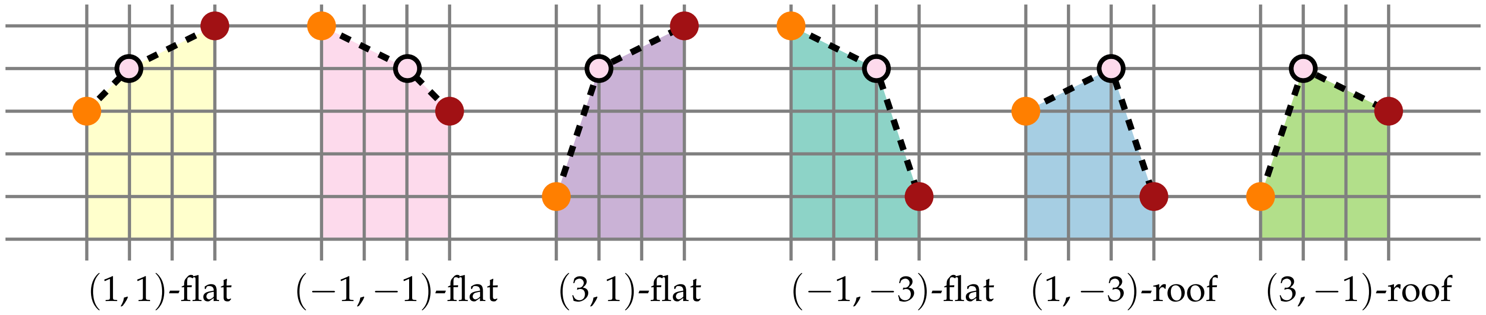
(3, 1)-flat

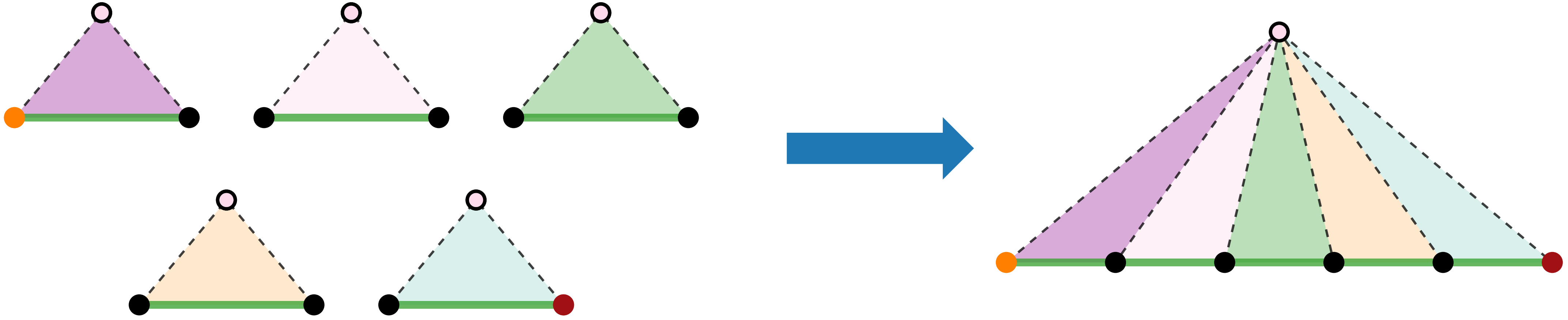


(3, −1)-roof

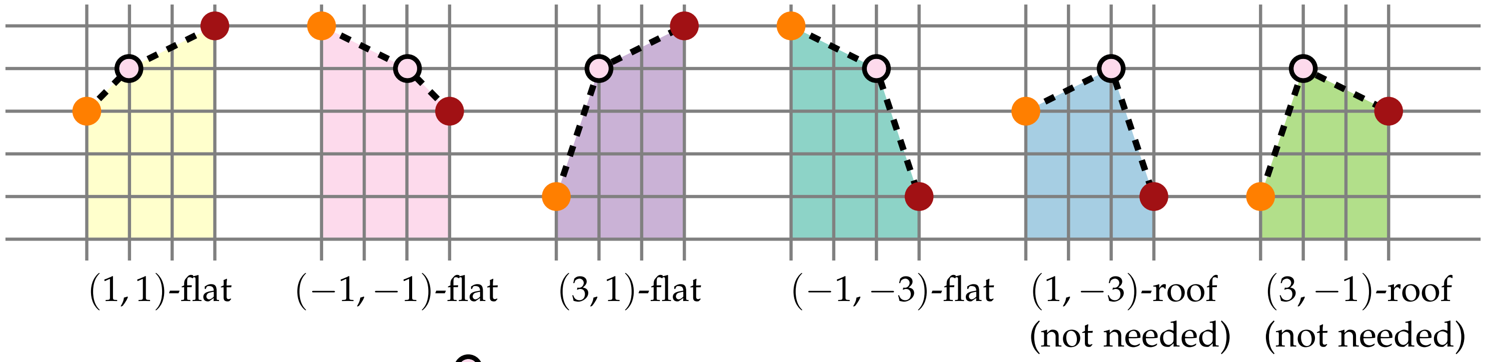


P-nodes

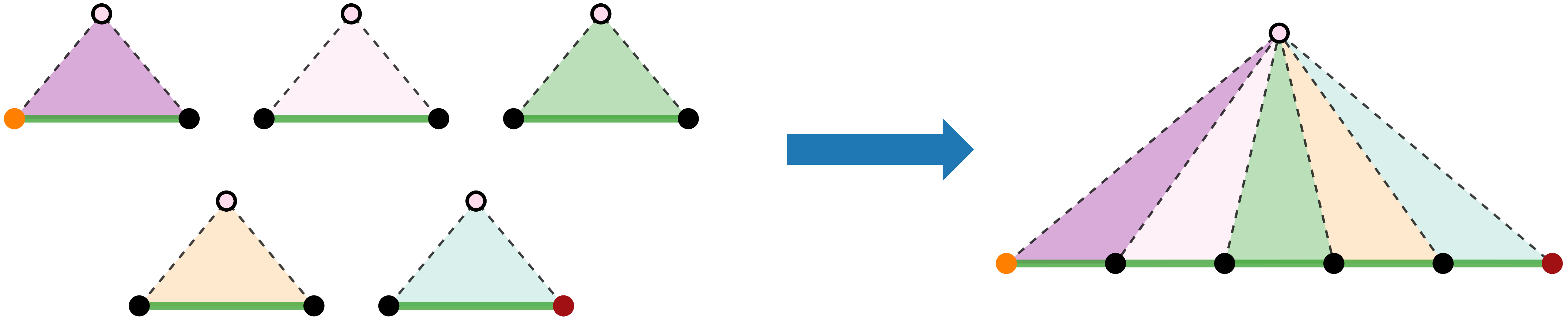




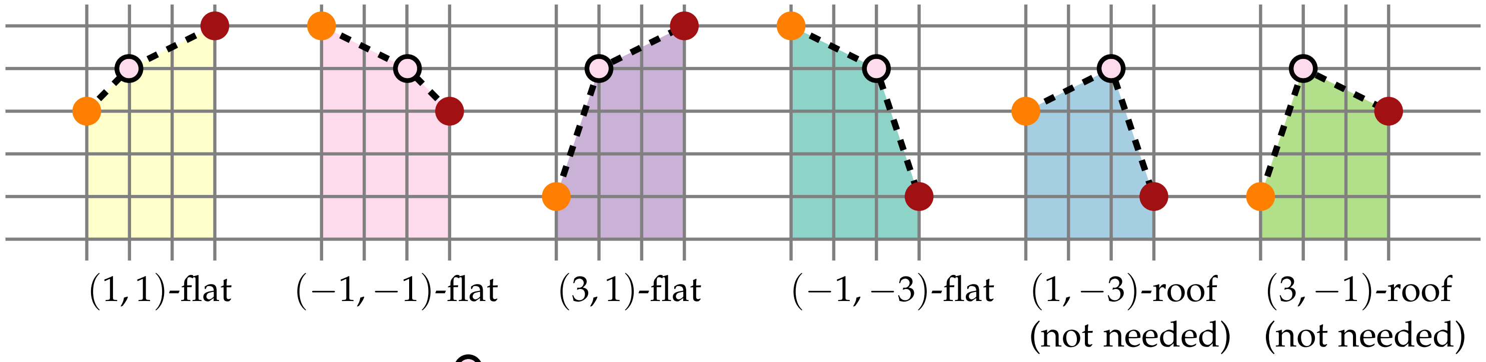
P-nodes

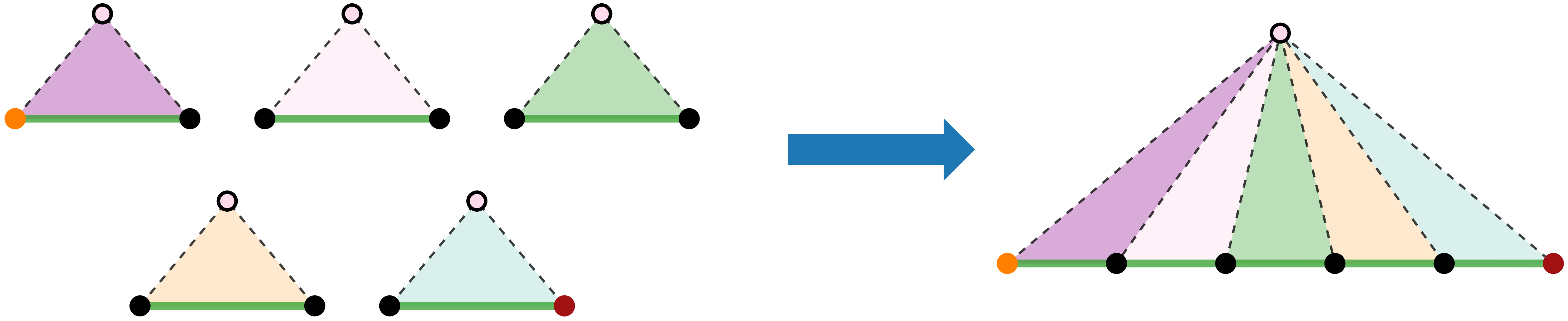


(1, 1)-flat
(−1, −1)-flat
(3, 1)-flat
(−1, −3)-flat
(1, −3)-roof
(not needed)
(3, −1)-roof
(not needed)



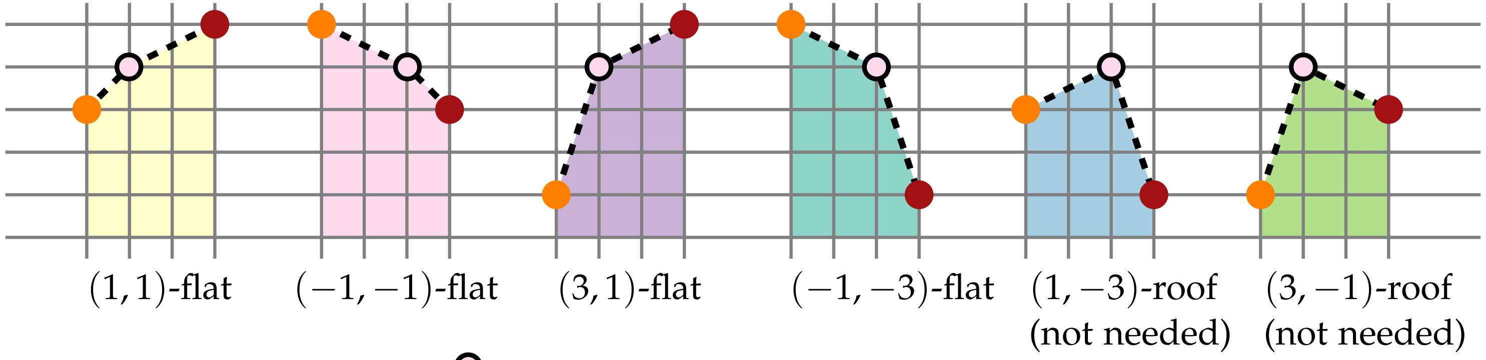
P-nodes

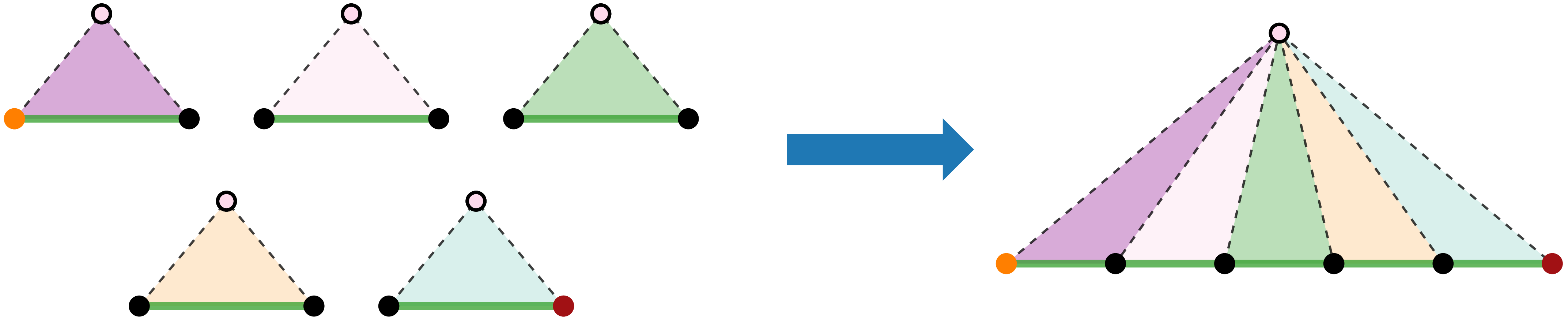




(1,1)-flat

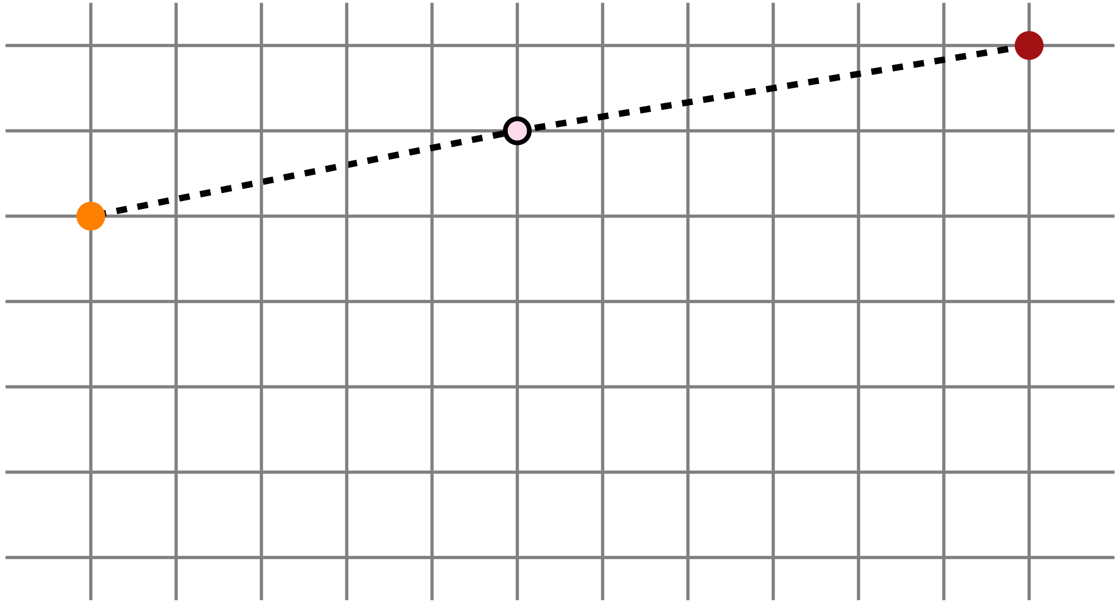
P-nodes



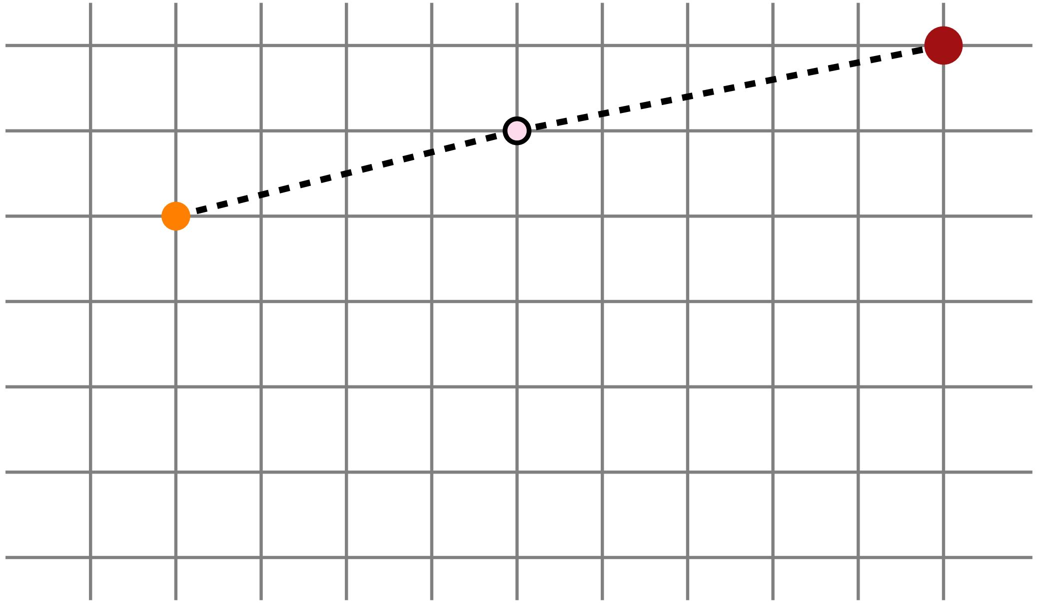


(1,1)-flat

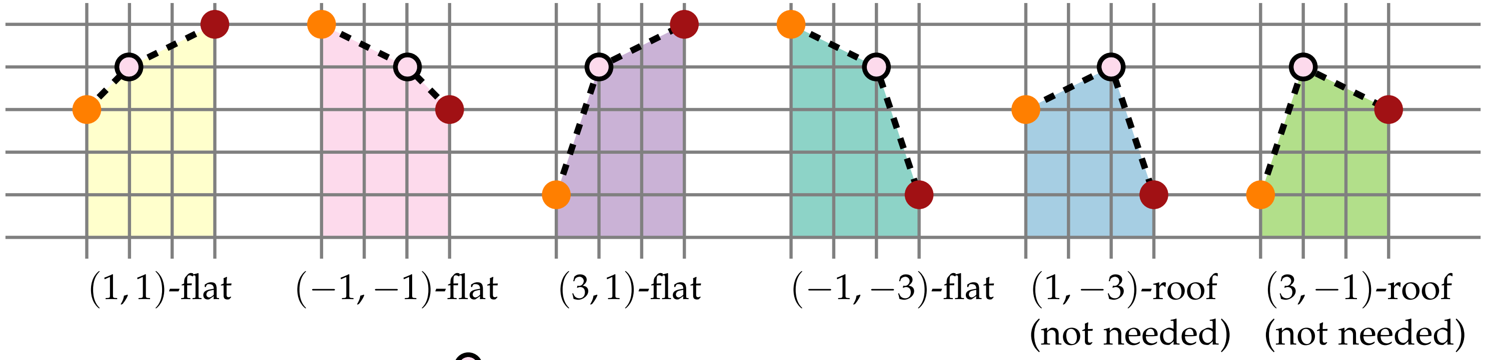
odd degree

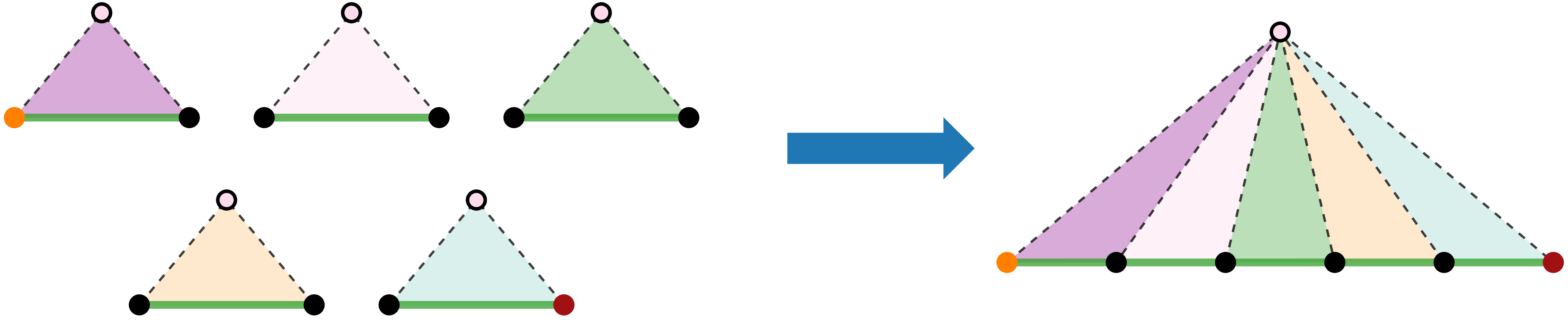


even degree



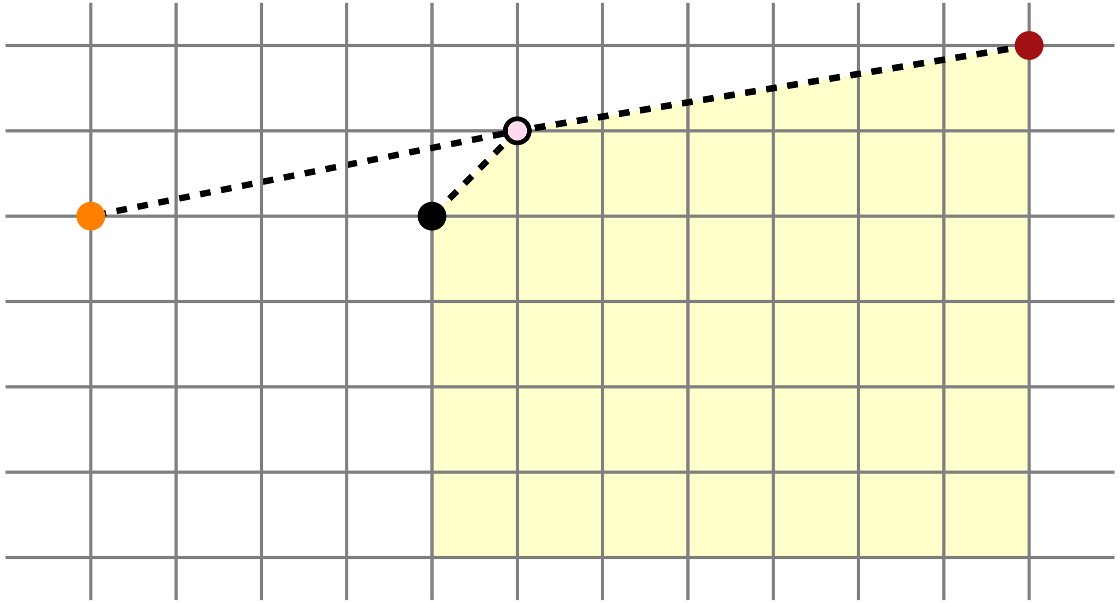
P-nodes



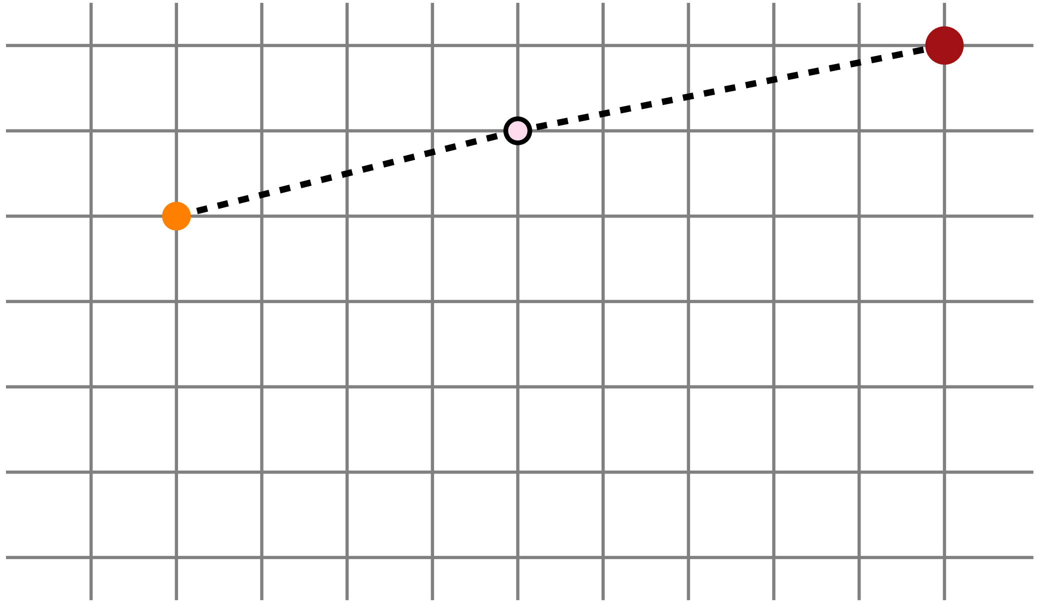


(1,1)-flat

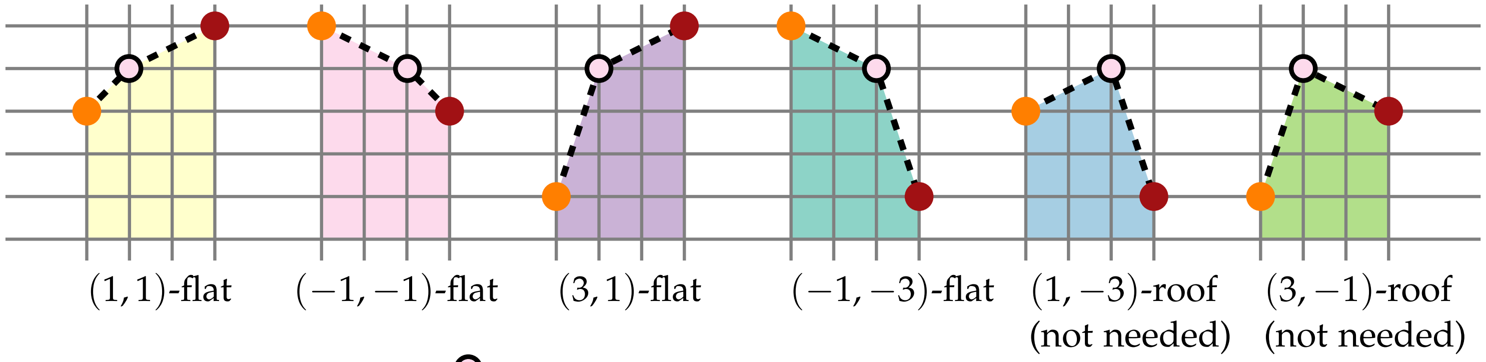
odd degree

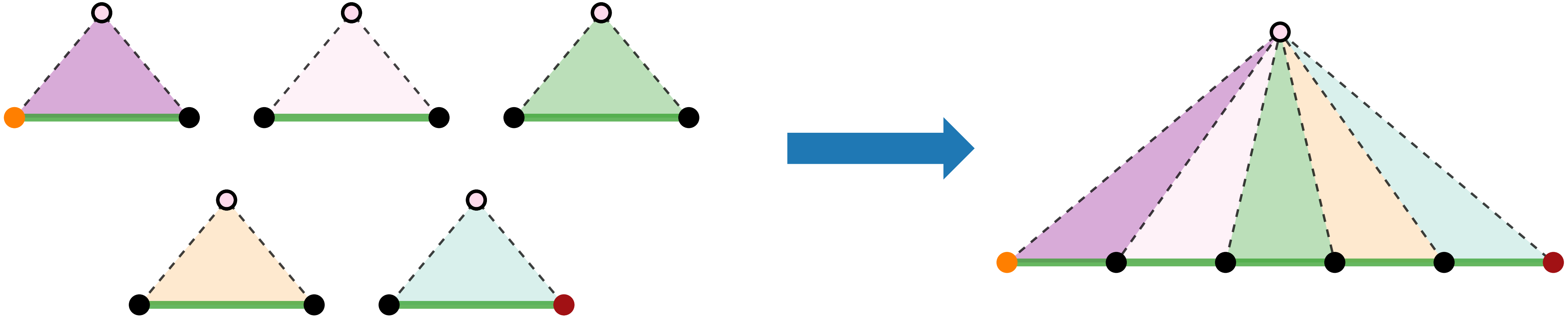


even degree



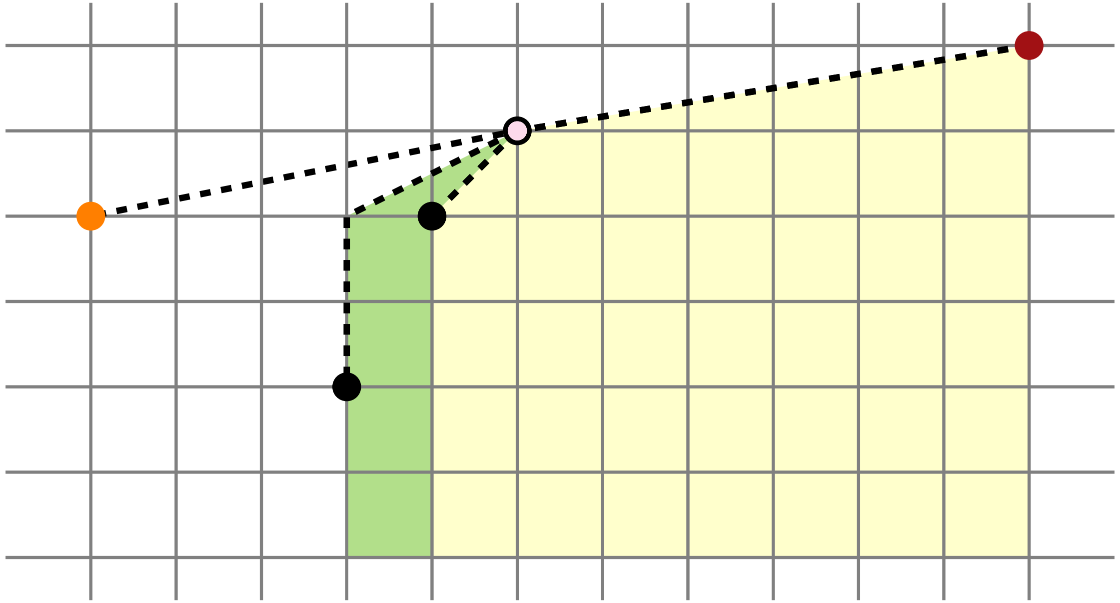
P-nodes



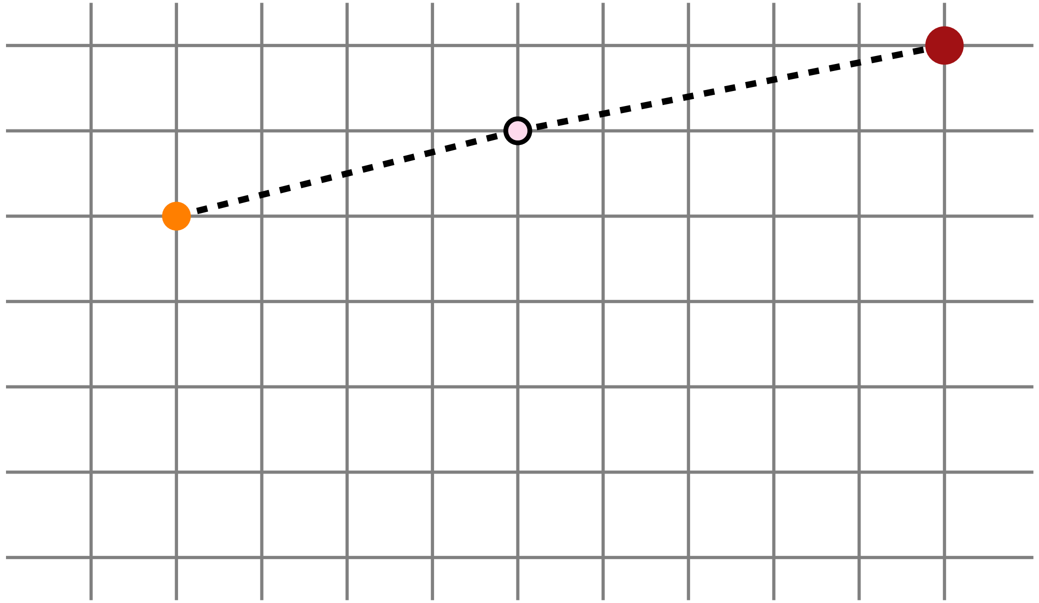


(1,1)-flat

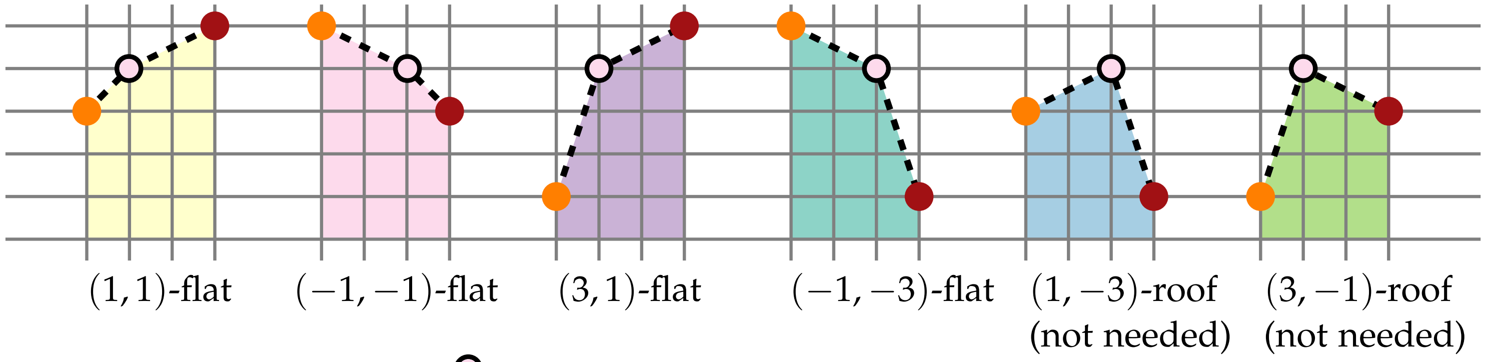
odd degree

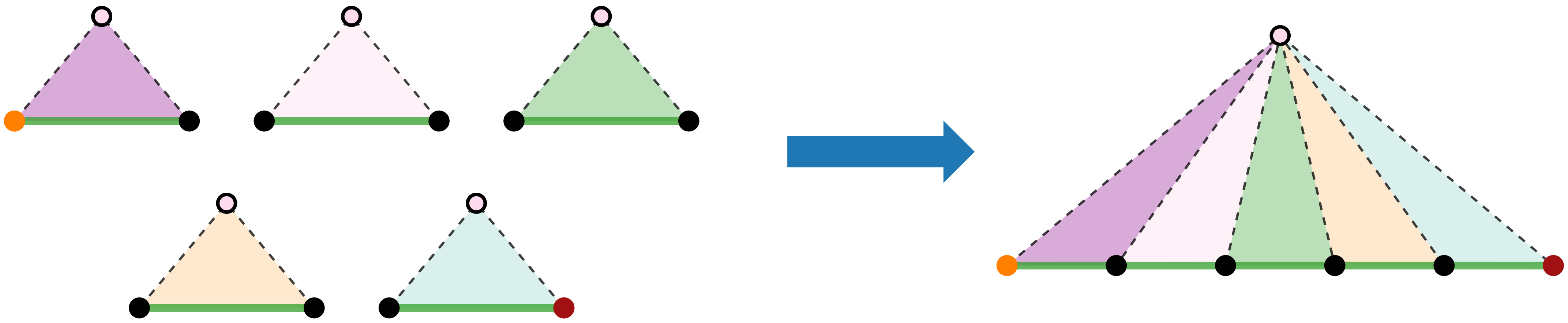


even degree



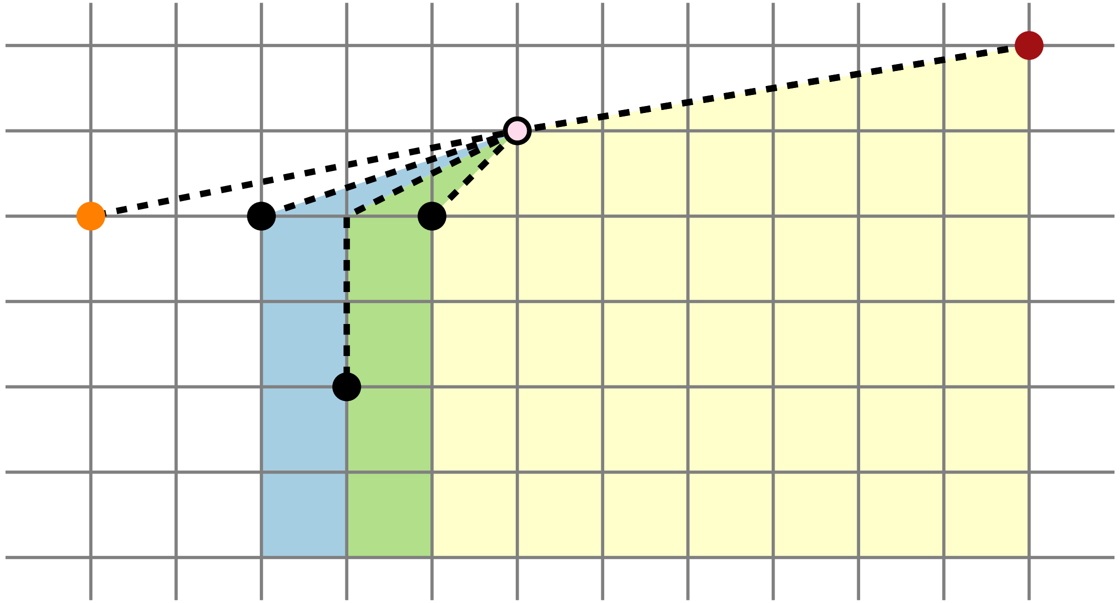
P-nodes



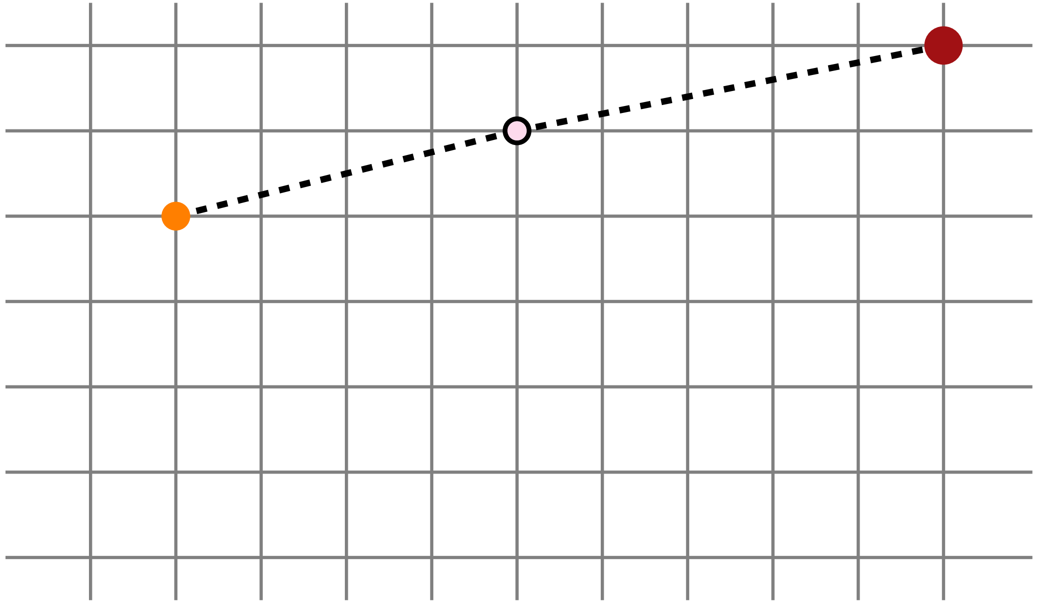


(1,1)-flat

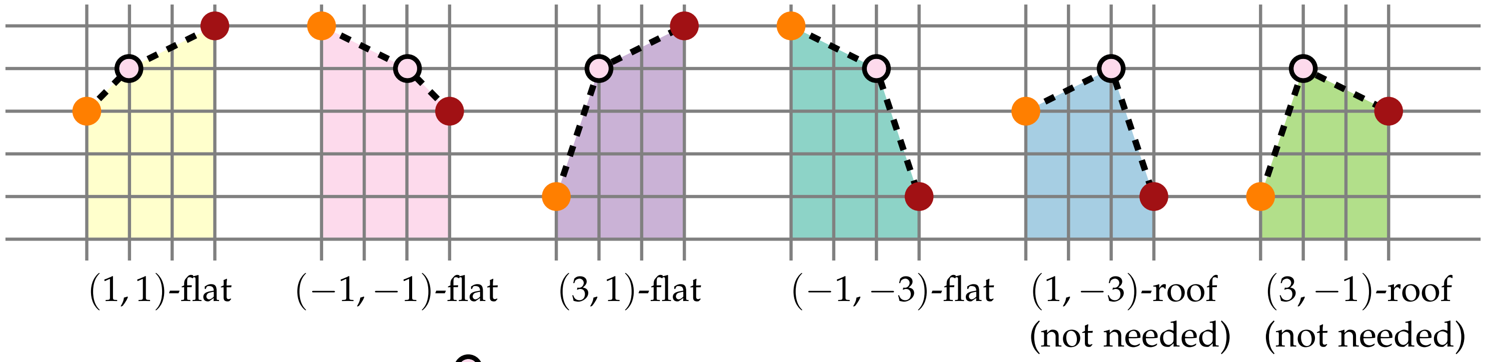
odd degree

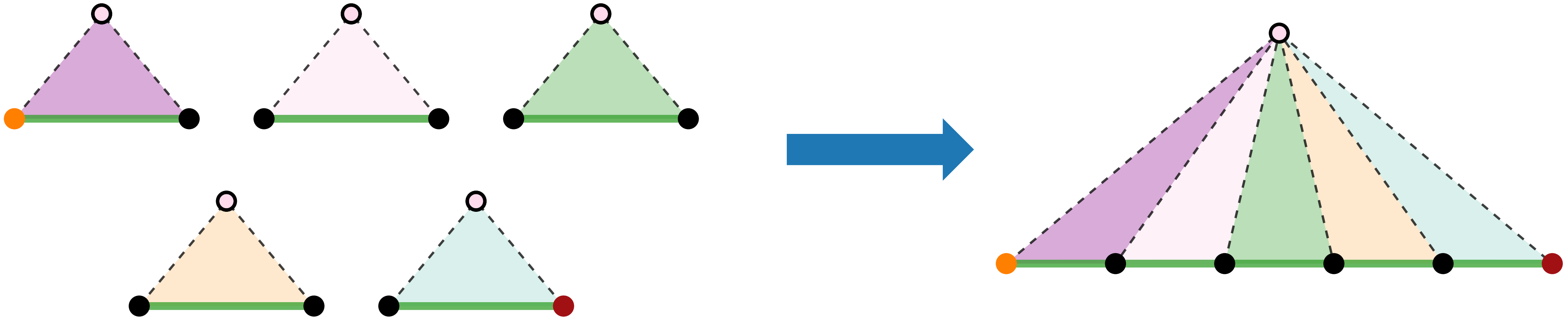


even degree



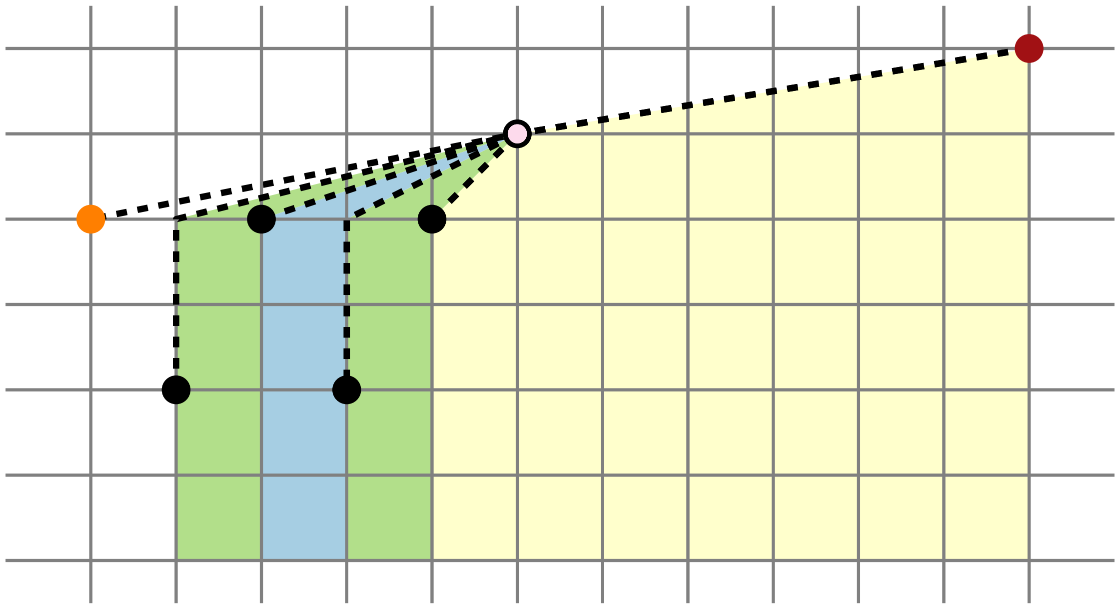
P-nodes



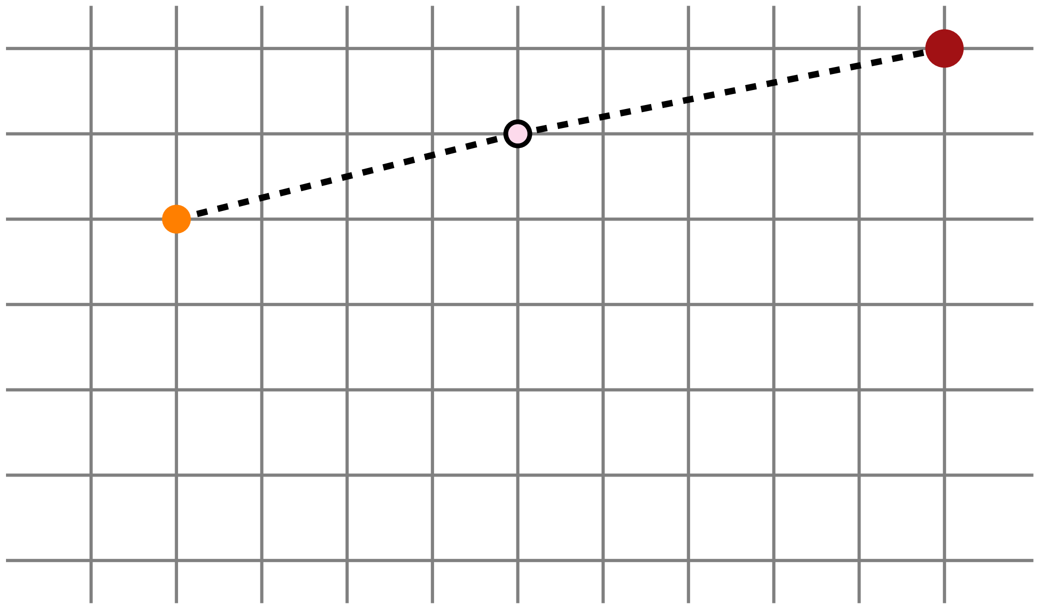


(1,1)-flat

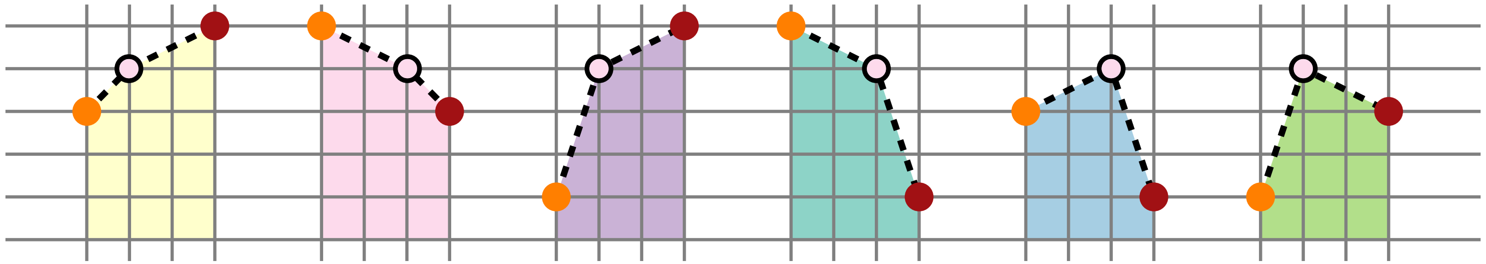
odd degree

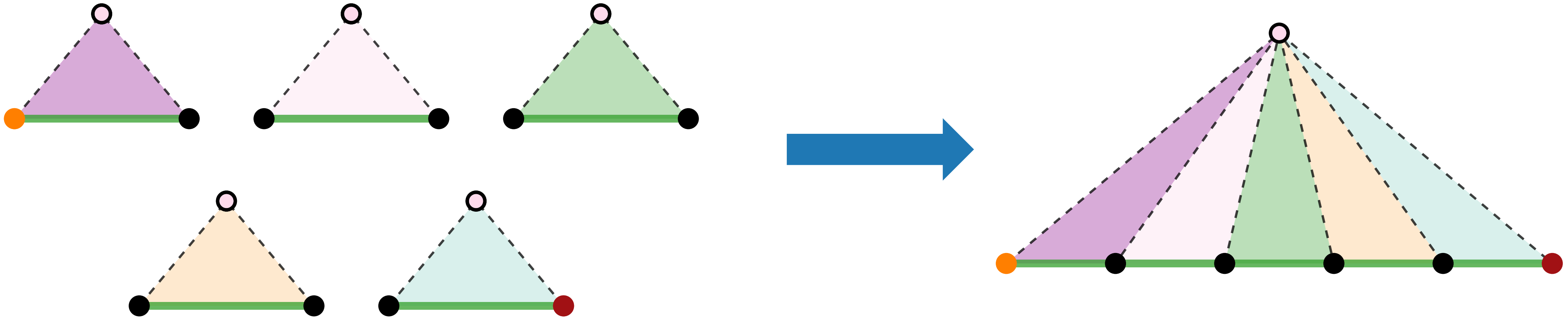


even degree



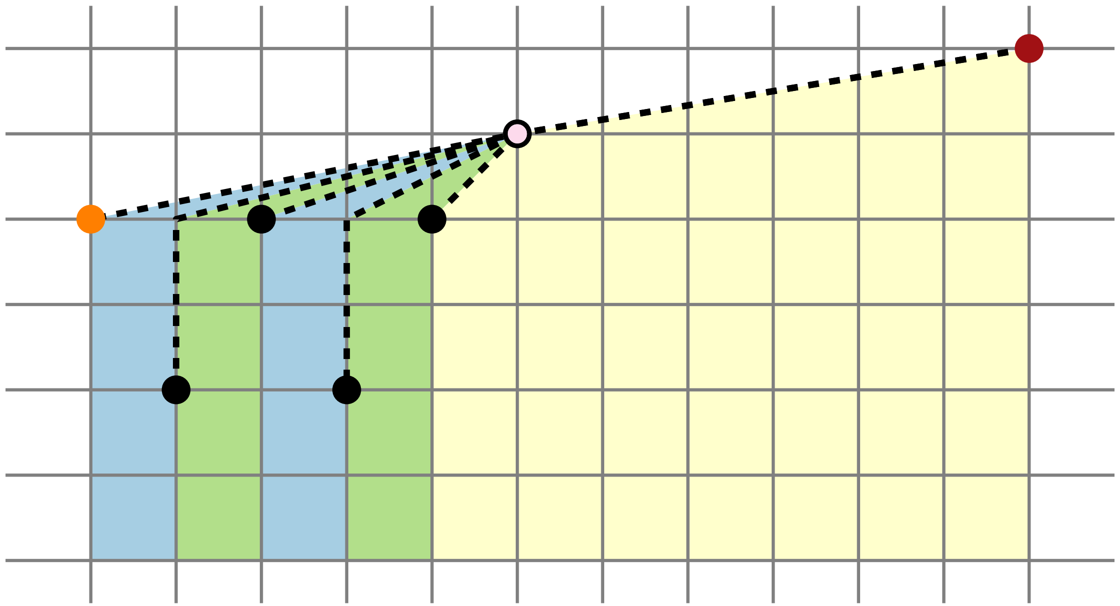
P-nodes



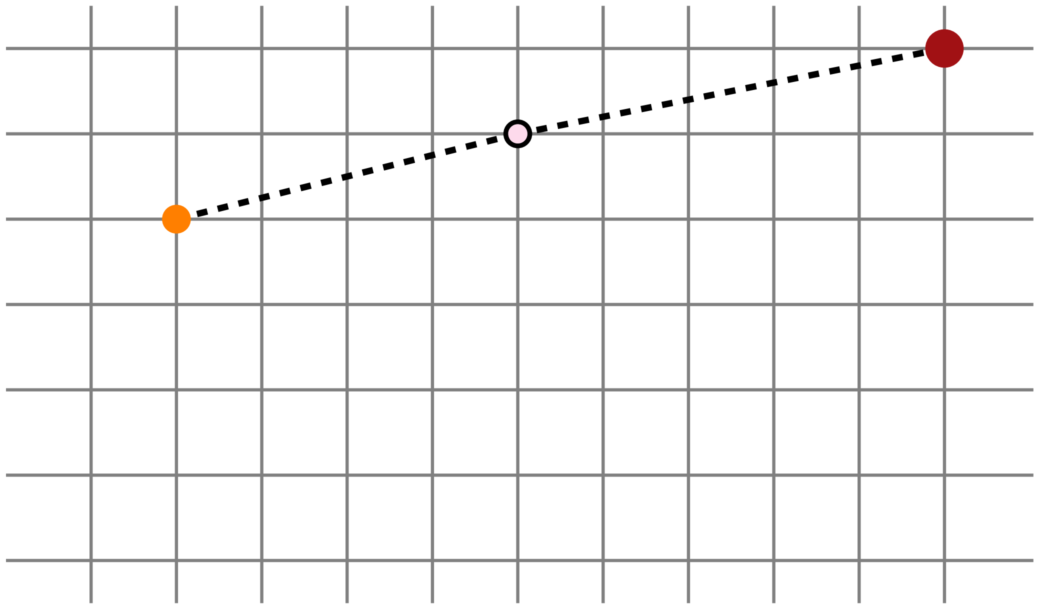


(1,1)-flat

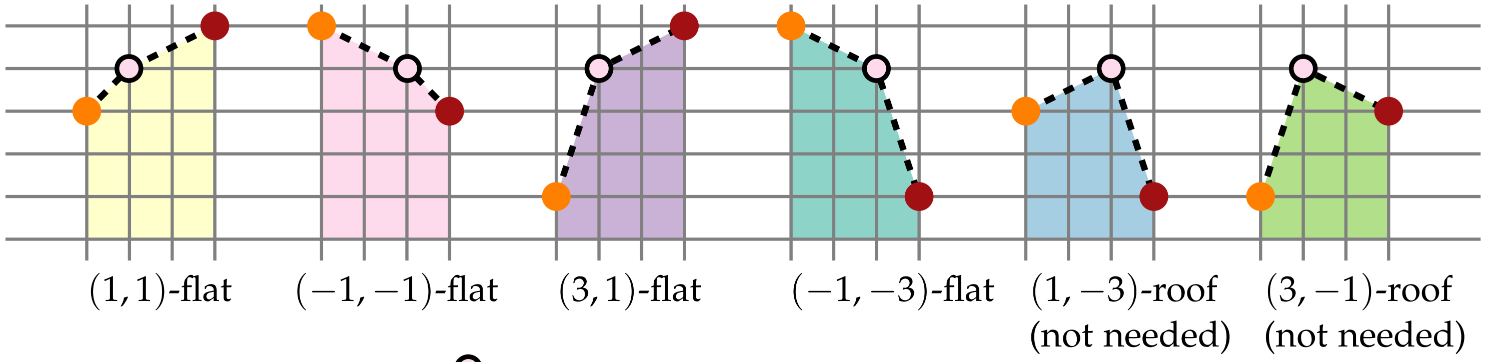
odd degree

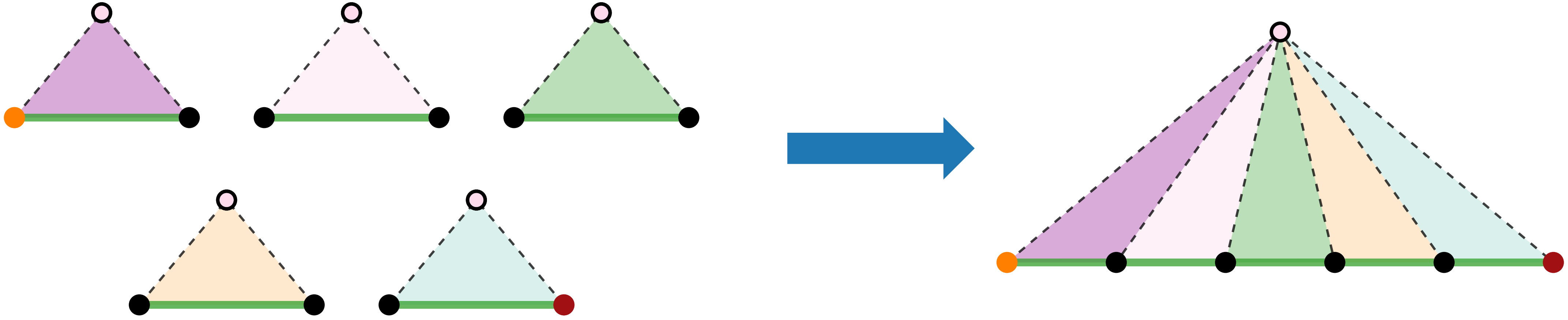


even degree

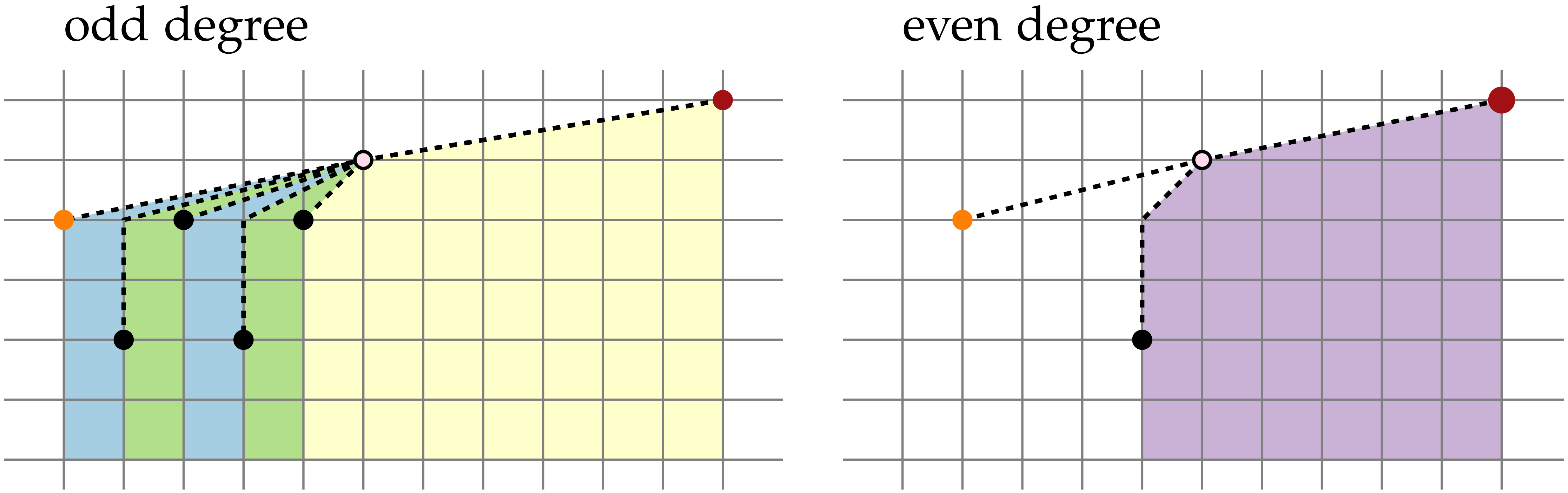


P-nodes

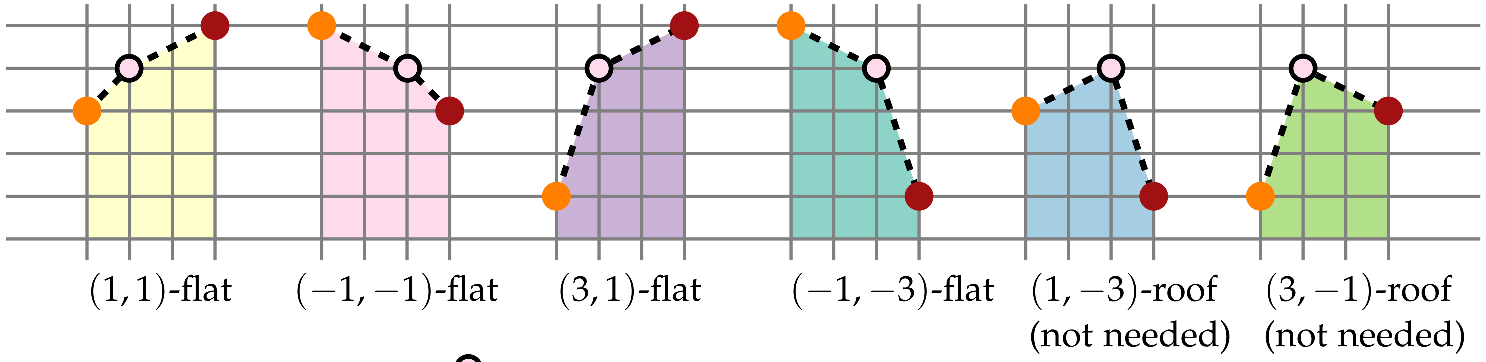


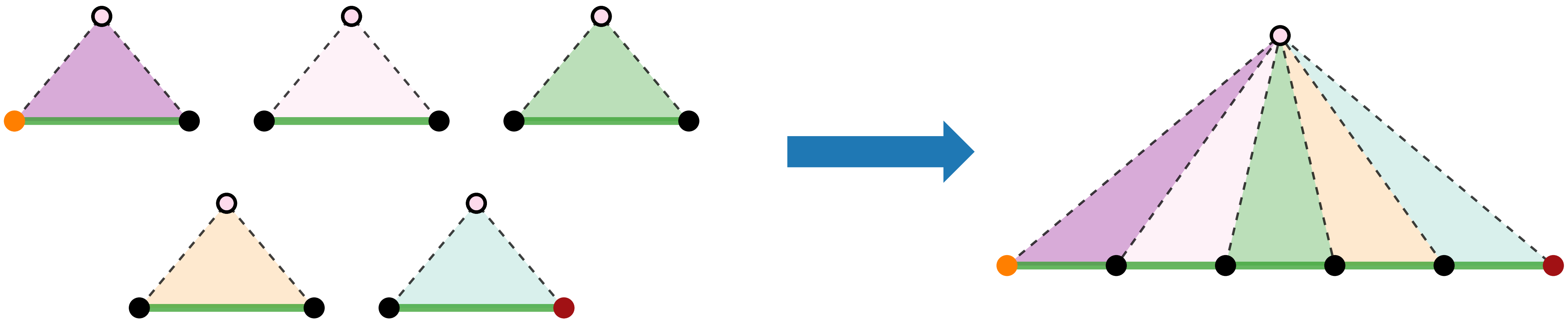


(1,1)-flat



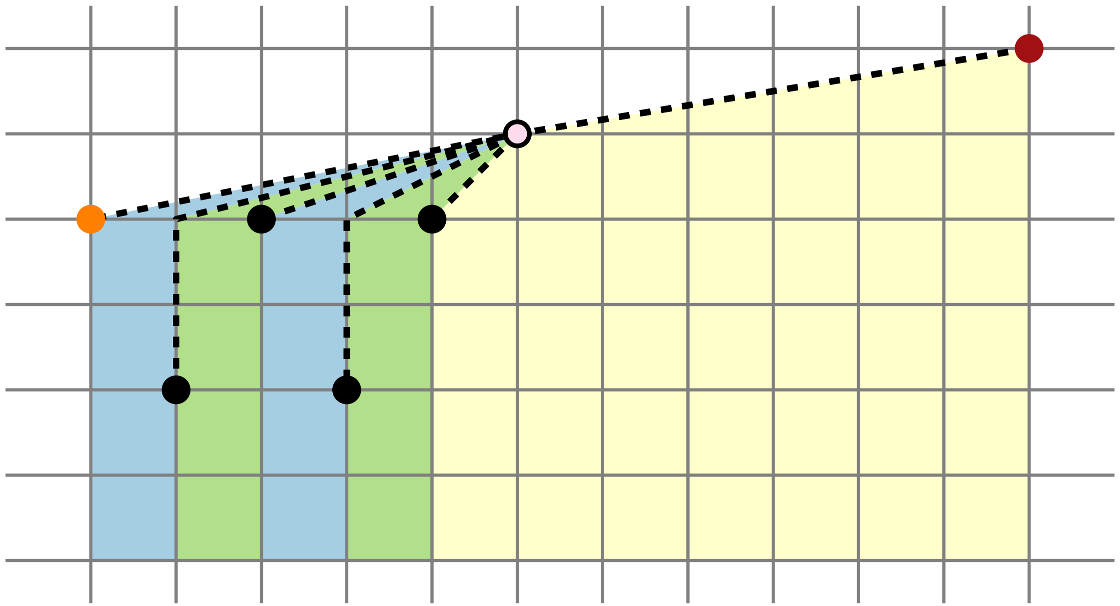
P-nodes



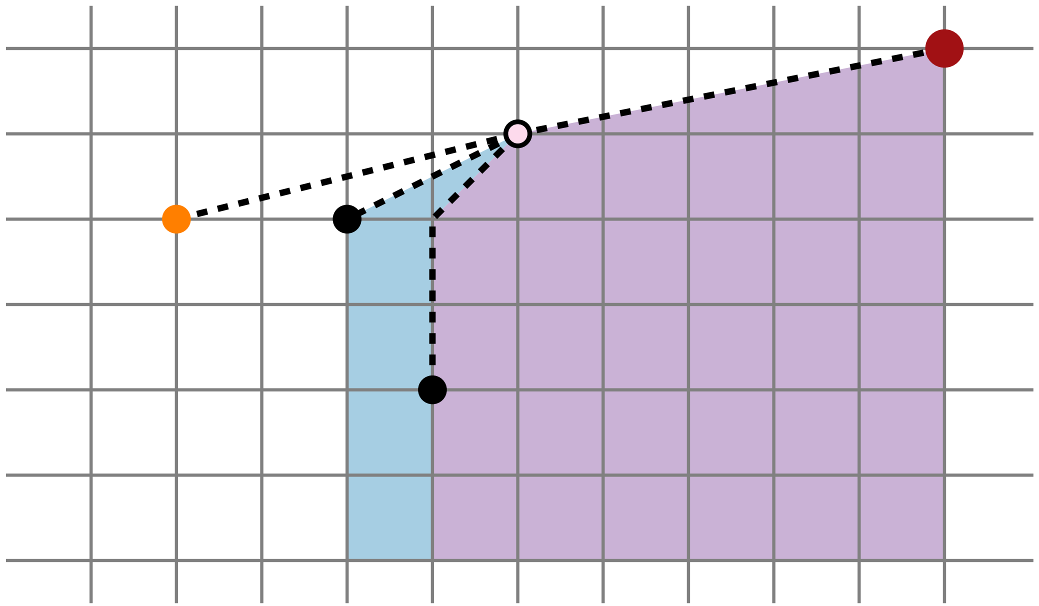


(1,1)-flat

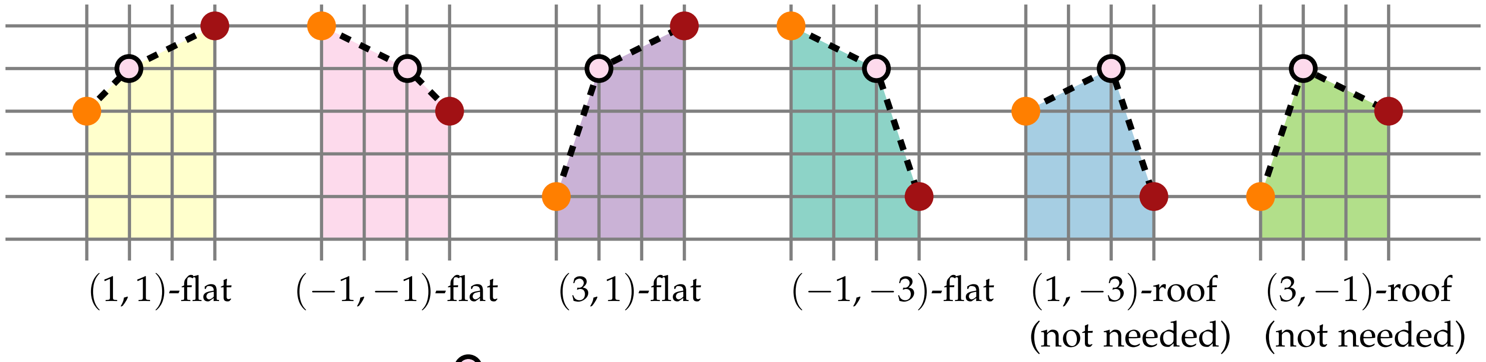
odd degree

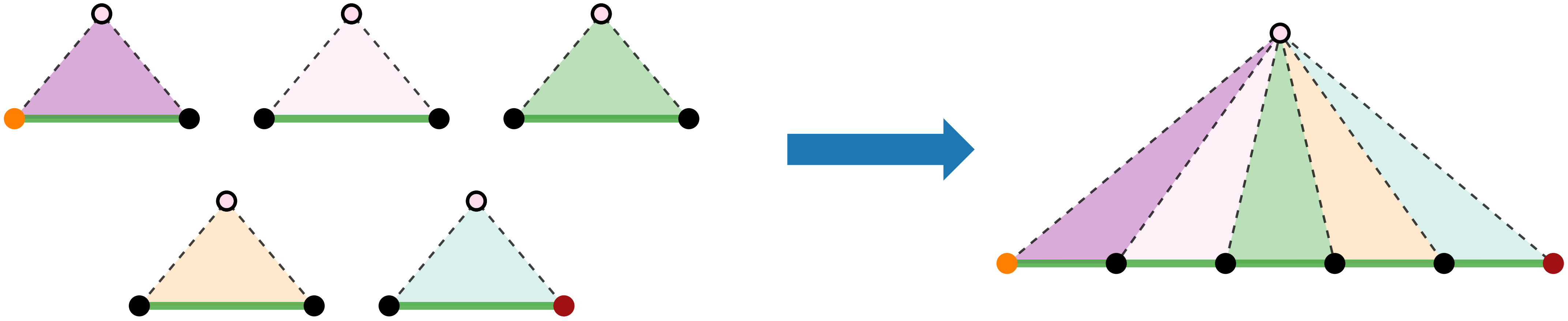


even degree



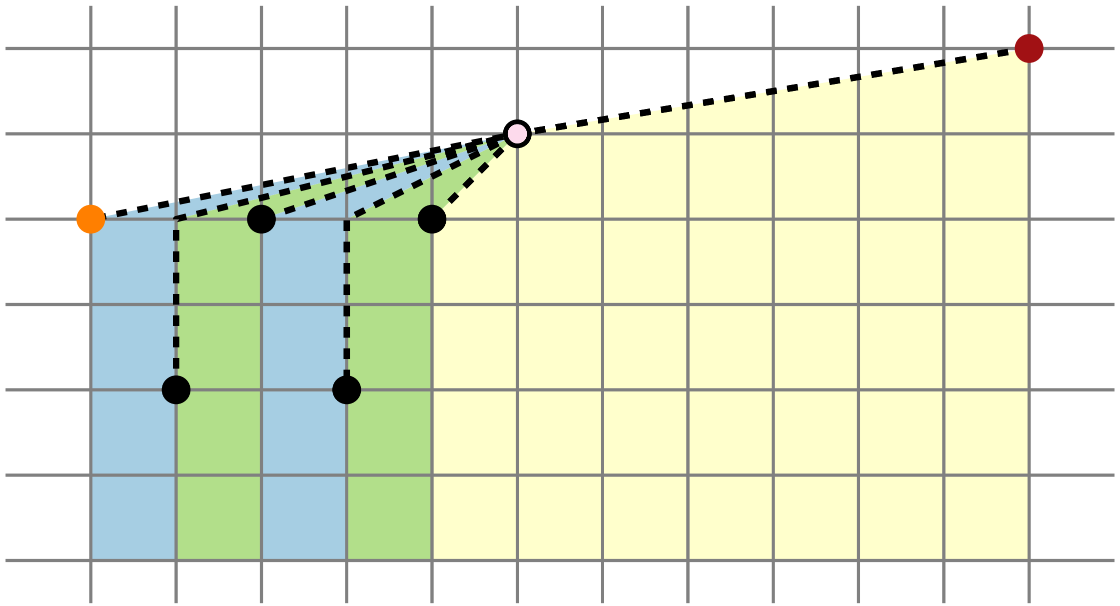
P-nodes



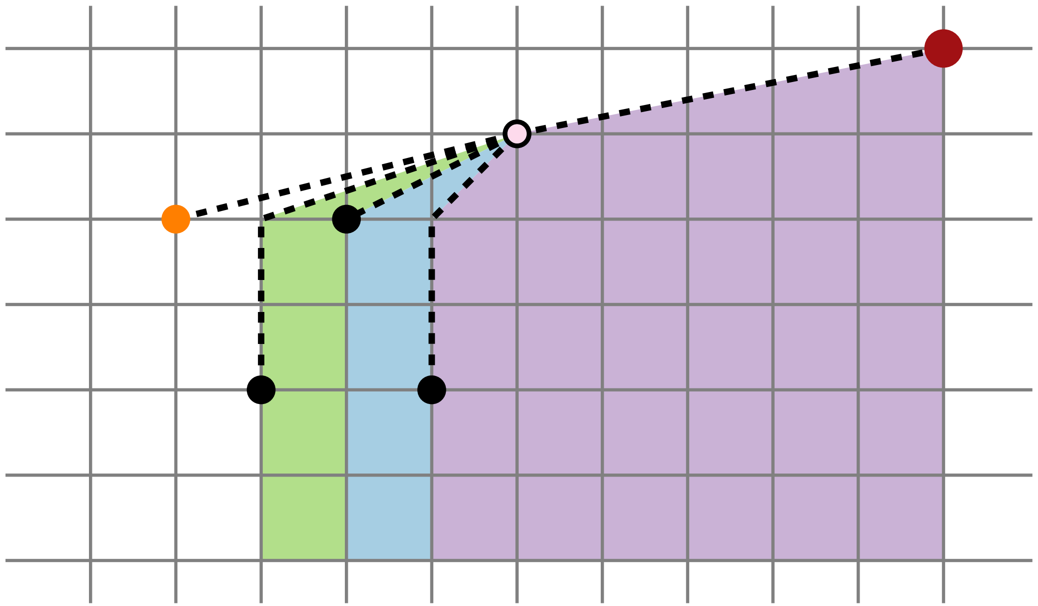


(1,1)-flat

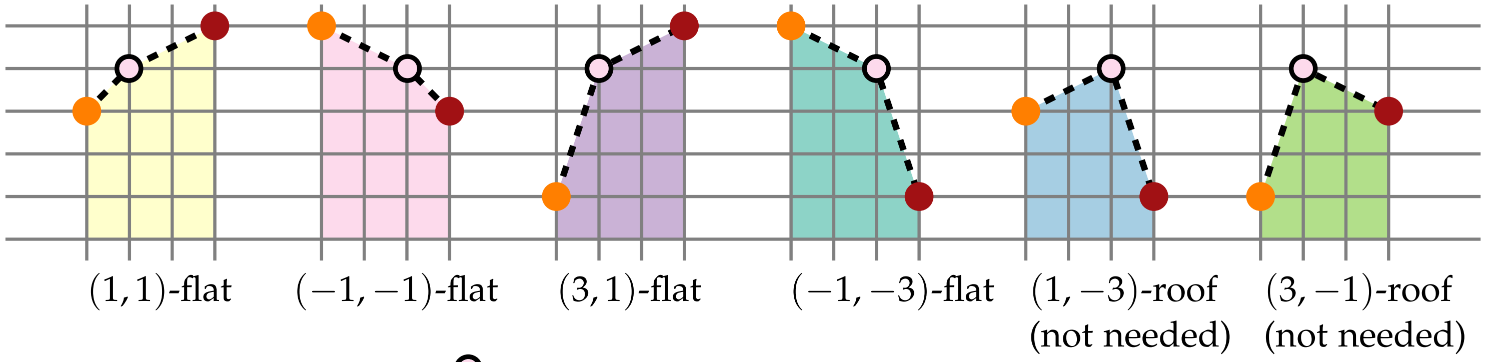
odd degree

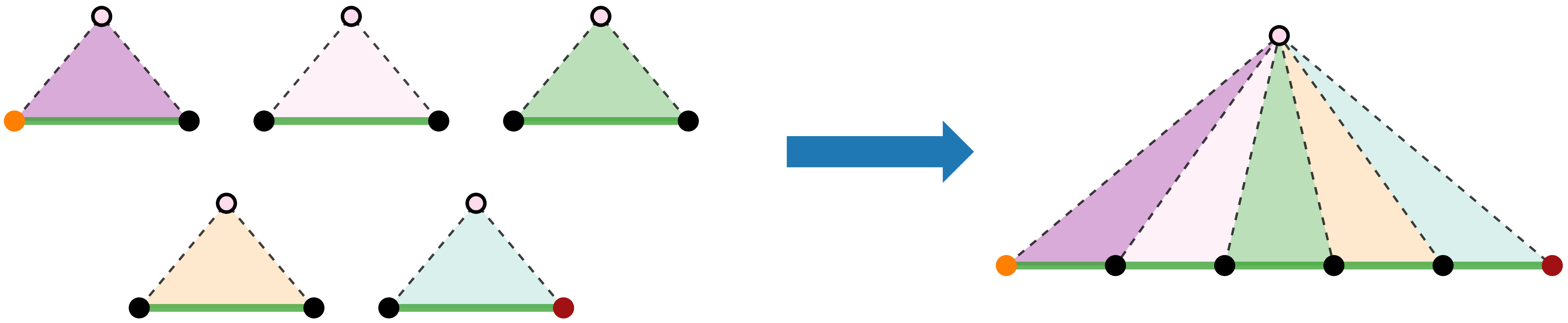


even degree



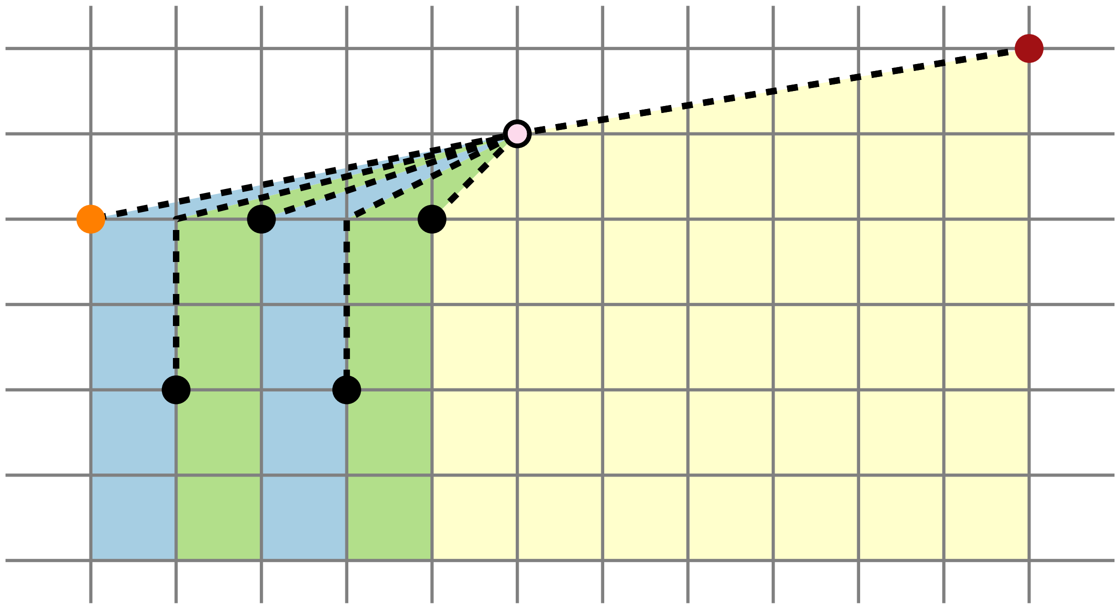
P-nodes



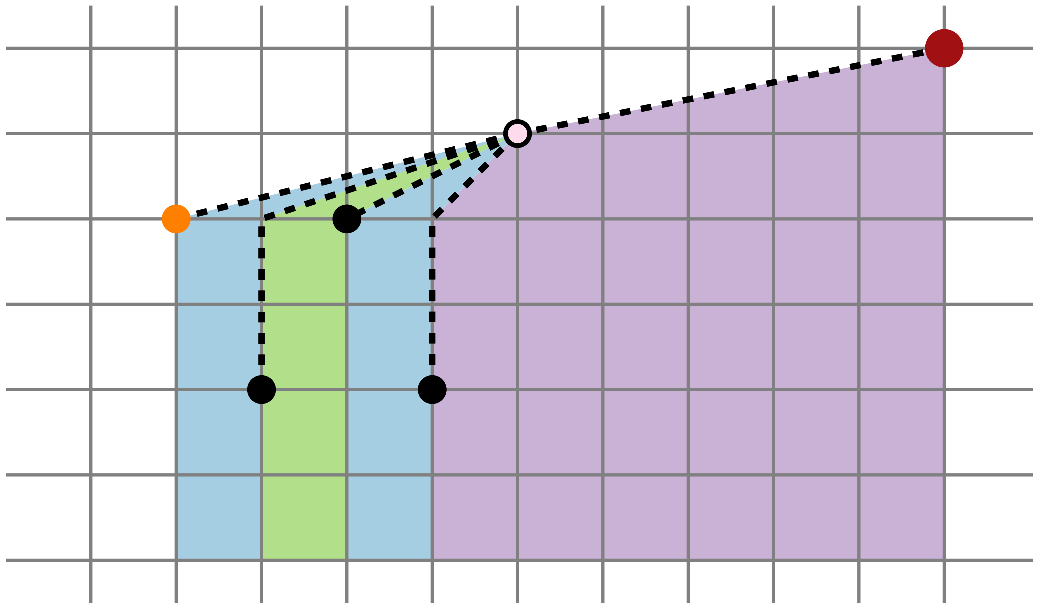


(1,1)-flat

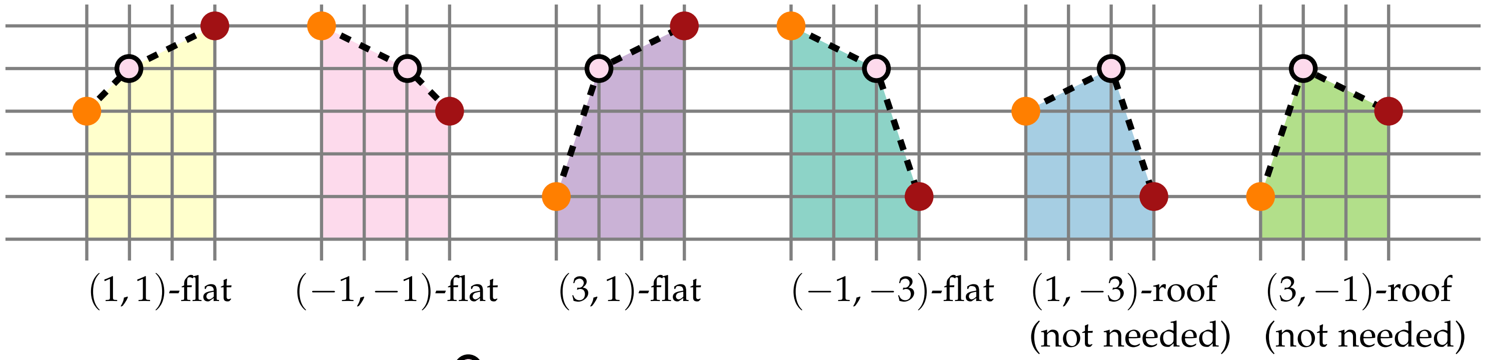
odd degree

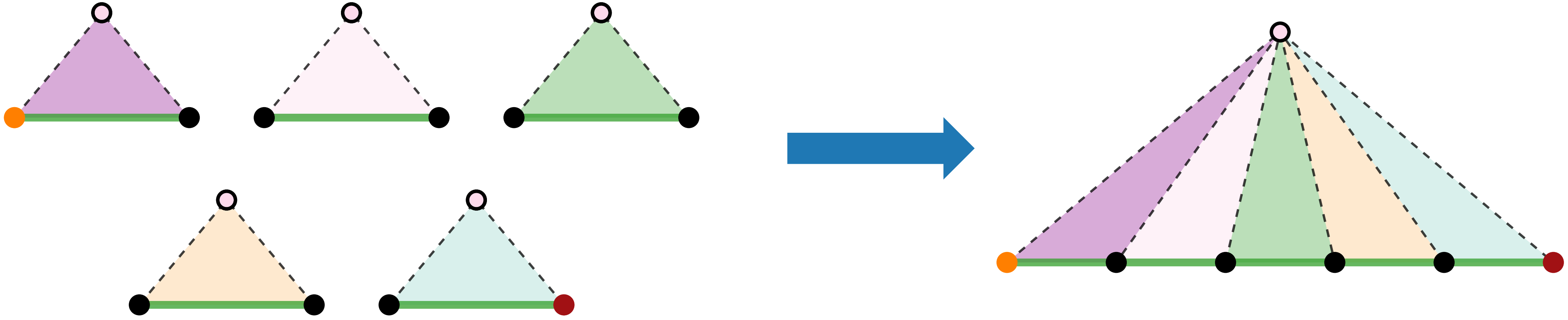


even degree



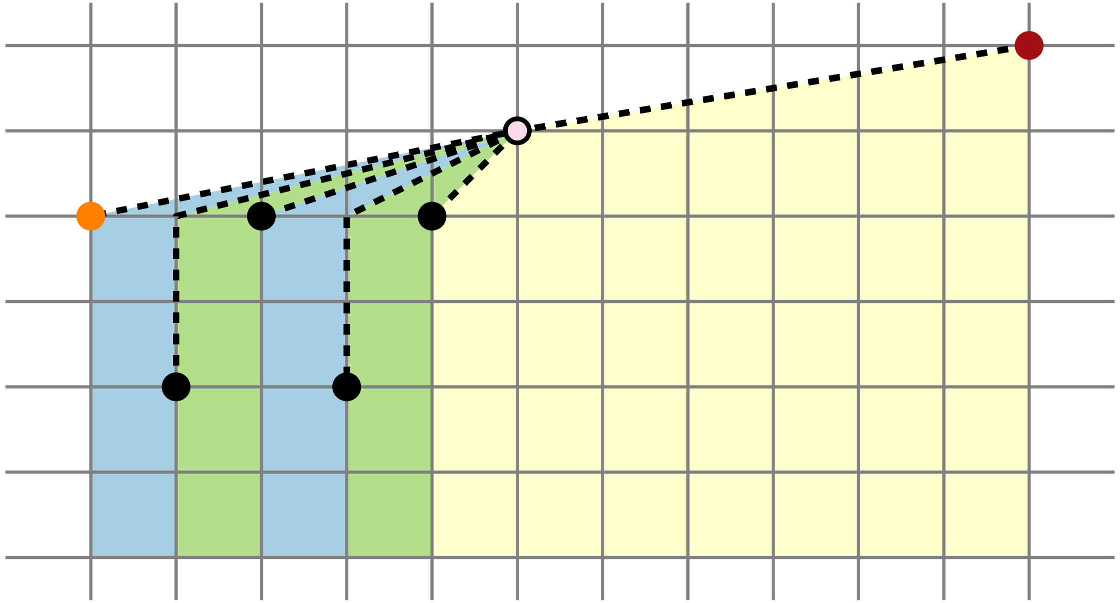
P-nodes



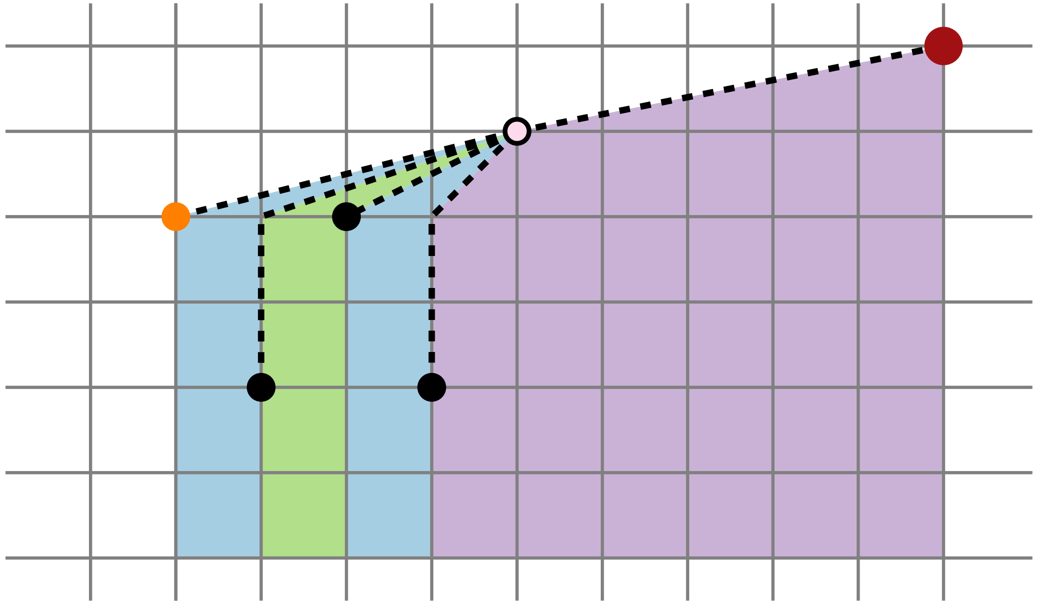


(3,1)-flat

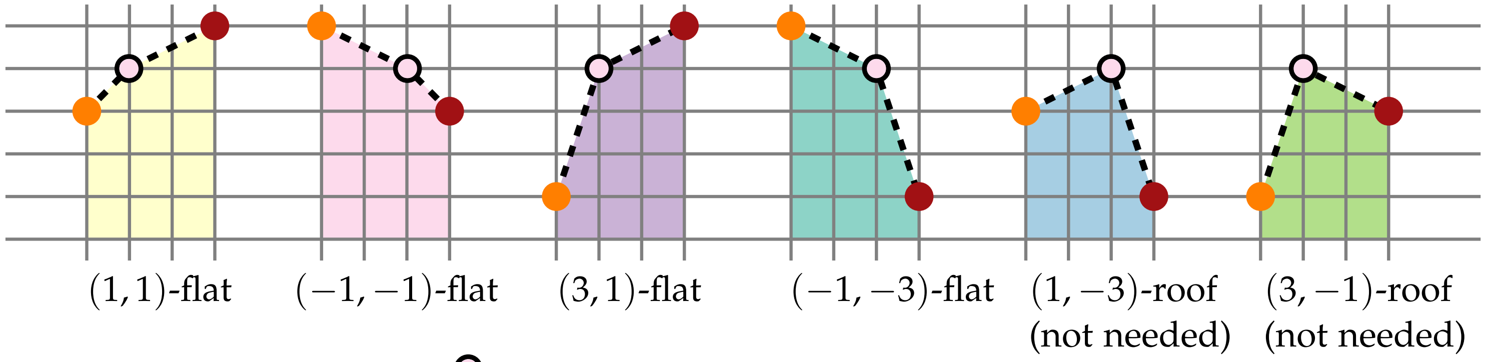
odd degree

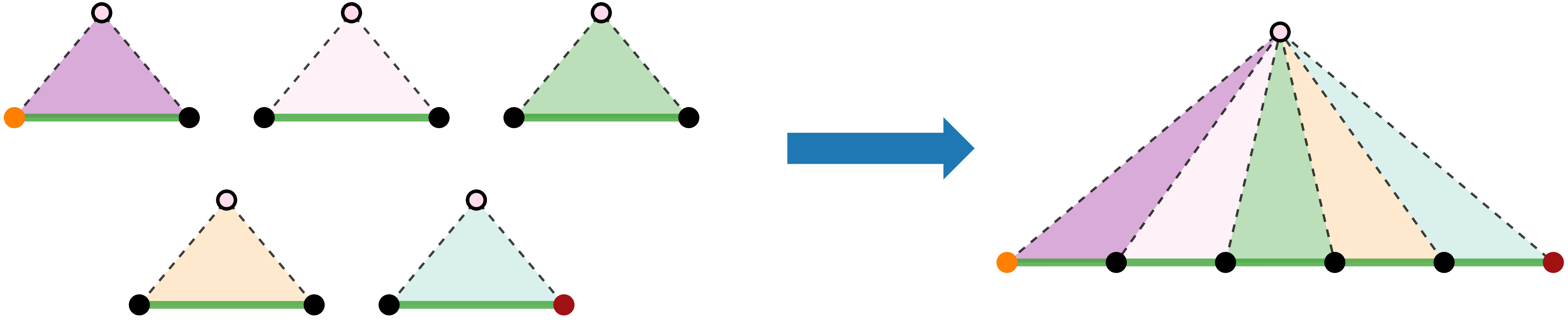


even degree



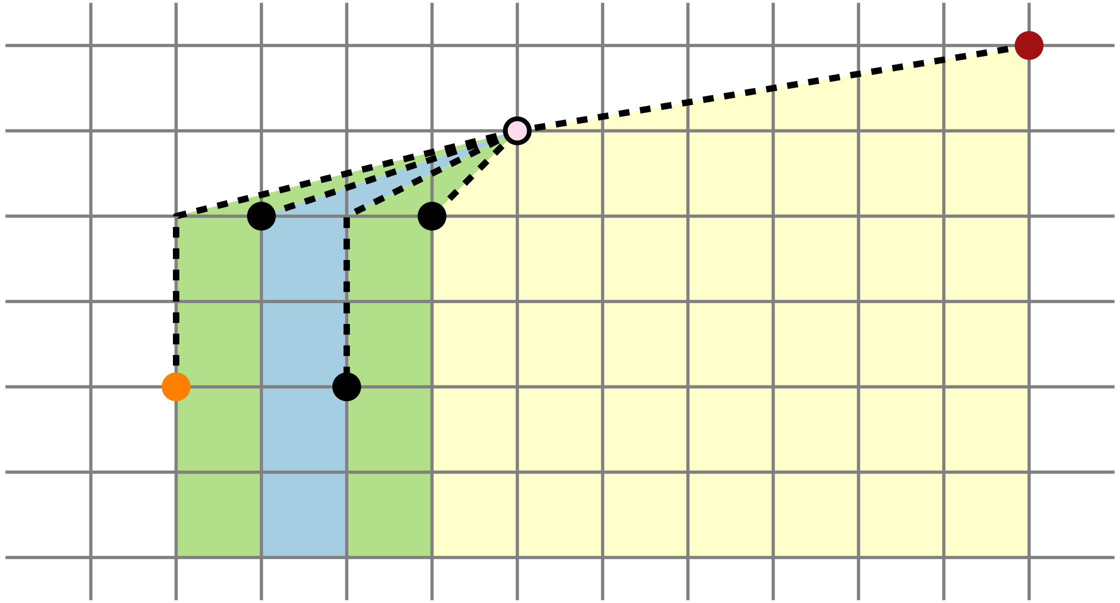
P-nodes



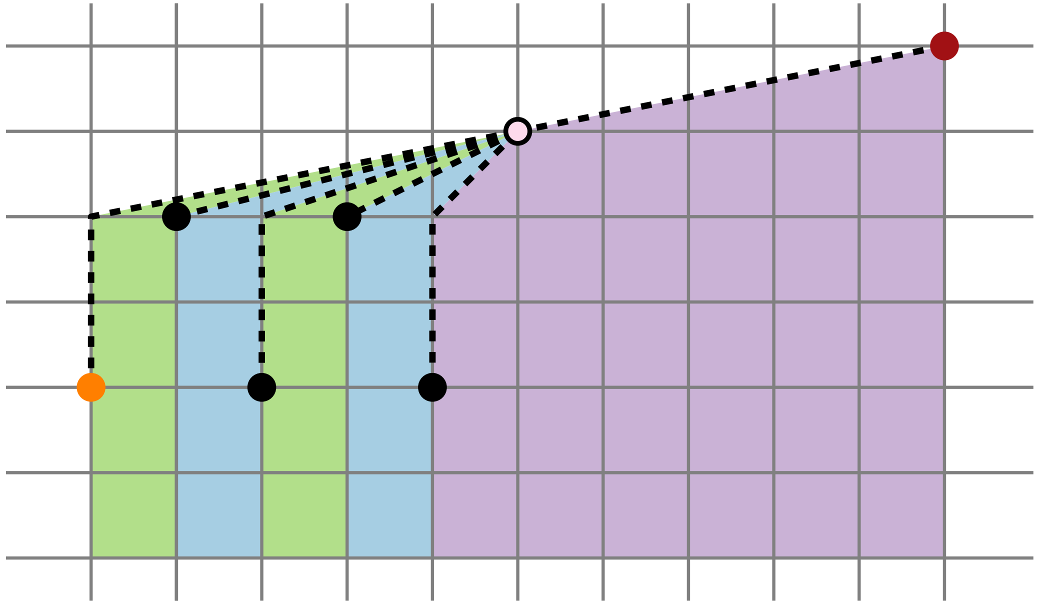


(3,1)-flat

even degree



odd degree



Results: Cycle-trees

Theorem.
Every almost-3-connected path-tree admits a 4-span weakly leveled planar drawing

Results: Cycle-trees

Theorem.
Every almost-3-connected path-tree admits a 4-span weakly leveled planar drawing

Cycle-trees?

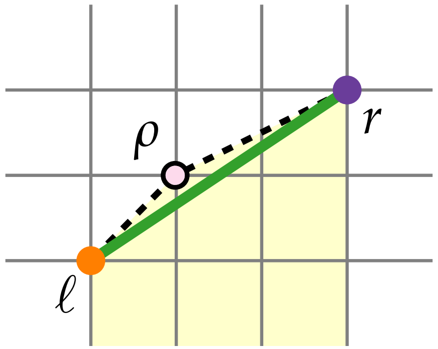
- remove edge (ℓ, r)

Results: Cycle-trees

Theorem.
Every almost-3-connected path-tree admits a 4-span weakly leveled planar drawing

Cycle-trees?

- remove edge (ℓ, r)
- Create (1,1)-flat drawing

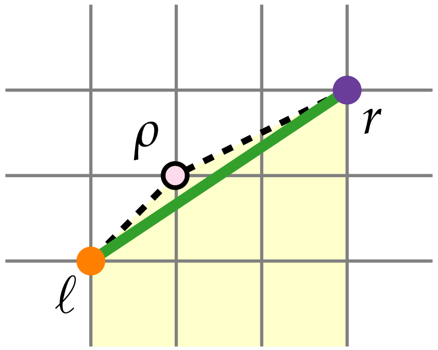


Results: Cycle-trees

Theorem.
Every almost-3-connected path-tree admits a 4-span weakly leveled planar drawing

Cycle-trees?

- remove edge (ℓ, r)
- Create (1,1)-flat drawing
- Reinsert (ℓ, r) with span 2

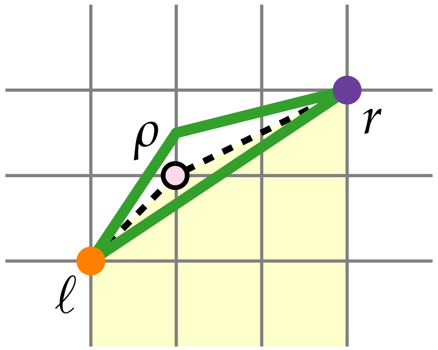


Results: Cycle-trees

Theorem.
Every almost-3-connected path-tree admits a 4-span weakly leveled planar drawing

Cycle-trees?

- remove edge (ℓ, r)
- Create (1,1)-flat drawing
- Reinsert (ℓ, r) with span 2

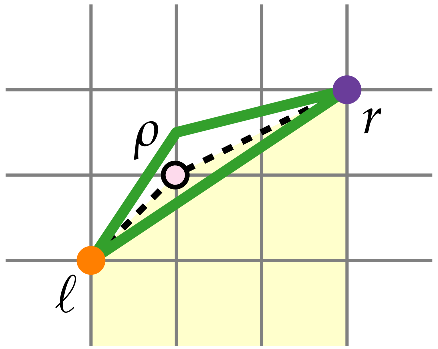


Results: Cycle-trees

Theorem.
Every almost-3-connected path-tree admits a 4-span weakly leveled planar drawing

Cycle-trees?

- remove edge (ℓ, r)
- Create (1,1)-flat drawing
- Reinsert (ℓ, r) with span 2



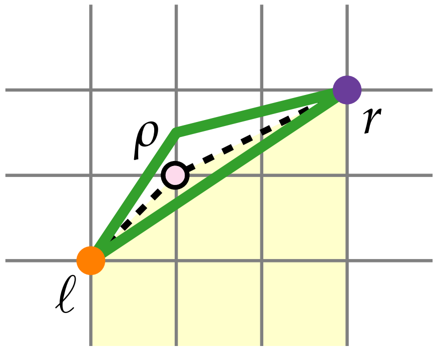
Theorem.
Every 3-connected cycle-tree admits a 4-span weakly leveled planar drawing.

Results: Cycle-trees

Theorem.
Every almost-3-connected path-tree admits a 4-span weakly leveled planar drawing

Cycle-trees?

- remove edge (ℓ, r)
- Create (1,1)-flat drawing
- Reinsert (ℓ, r) with span 2



Theorem.
Every 3-connected cycle-tree admits a 4-span weakly leveled planar drawing.

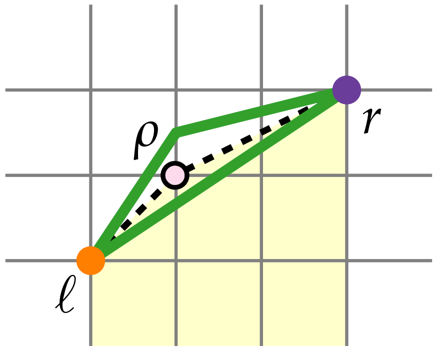
Theorem.
Some 3-connected cycle-trees require span ≥ 4 in any weakly leveled planar drawing.

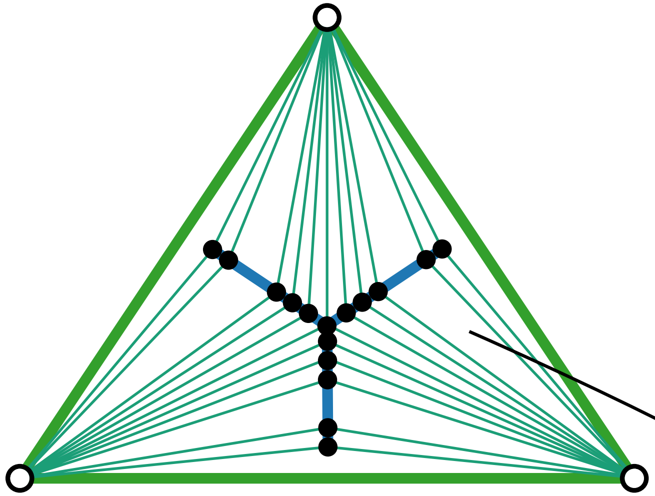
Results: Cycle-trees

Theorem.
Every almost-3-connected path-tree admits a 4-span weakly leveled planar drawing

Cycle-trees?

- remove edge (ℓ, r)
- Create (1,1)-flat drawing
- Reinsert (ℓ, r) with span 2





Theorem.
Every 3-connected cycle-tree admits a 4-span weakly leveled planar drawing.

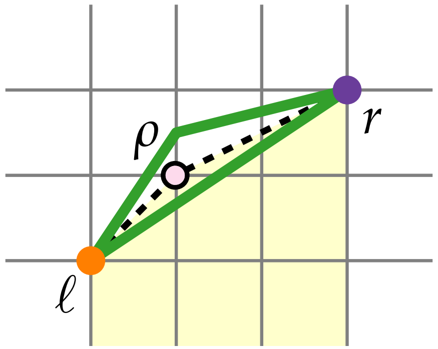
Theorem.
Some 3-connected cycle-trees require span ≥ 4 in any weakly leveled planar drawing.

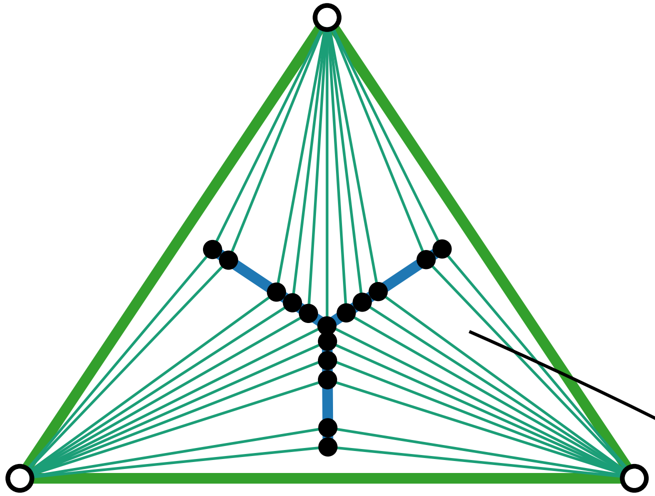
Results: Cycle-trees

Theorem.
Every almost-3-connected path-tree admits a 4-span weakly leveled planar drawing

Cycle-trees?

- remove edge (ℓ, r)
- Create (1,1)-flat drawing
- Reinsert (ℓ, r) with span 2





Theorem.
Every 3-connected cycle-tree admits a 4-span weakly leveled planar drawing.

Theorem.
Some 3-connected cycle-trees require span ≥ 4 in any weakly leveled planar drawing.

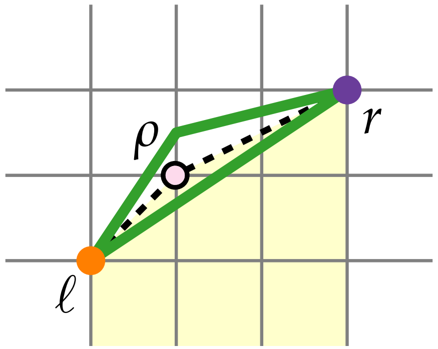
Theorem.
Some cycle-trees require span $\Omega(\log n)$ in any weakly leveled planar drawing.

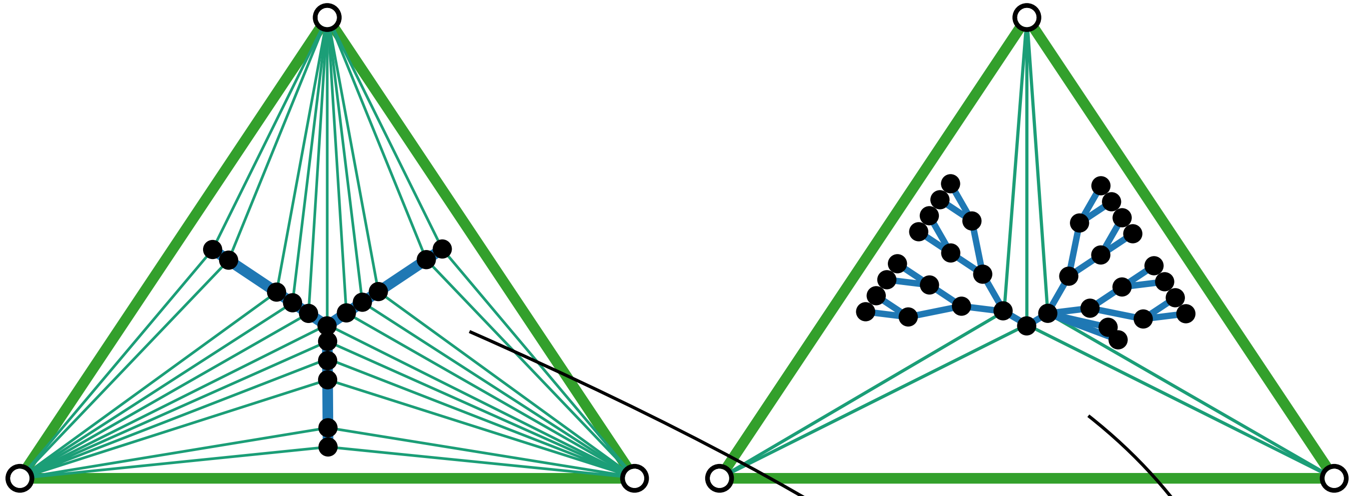
Results: Cycle-trees

Theorem.
Every almost-3-connected path-tree admits a 4-span weakly leveled planar drawing

Cycle-trees?

- remove edge (ℓ, r)
- Create (1,1)-flat drawing
- Reinsert (ℓ, r) with span 2





Theorem.
Every 3-connected cycle-tree admits a 4-span weakly leveled planar drawing.

Theorem.
Some 3-connected cycle-trees require span ≥ 4 in any weakly leveled planar drawing.

Theorem.
Some cycle-trees require span $\Omega(\log n)$ in any weakly leveled planar drawing.

Other bounds and edge ratio

Theorem.
Every treewidth-2 graph admits an $\mathcal{O}(\sqrt{n})$-span weakly leveled planar drawing.

Other bounds and edge ratio

Theorem.
Every treewidth-2 graph admits an $\mathcal{O}(\sqrt{n})$-span weakly leveled planar drawing.

Theorem.
There planar treewidth-2 graphs that require span $2^{\Omega(\sqrt{log n})}$ in any weakly leveled planar drawing.

Other bounds and edge ratio

Theorem.
Every treewidth-2 graph admits an $\mathcal{O}(\sqrt{n})$-span weakly leveled planar drawing.

Theorem.
There planar treewidth-2 graphs that require span $2^{\Omega(\sqrt{log n})}$ in any weakly leveled planar drawing.

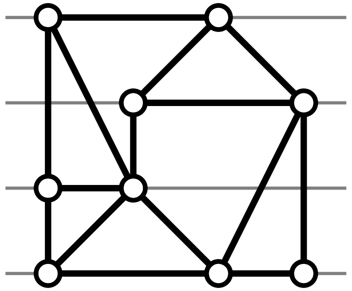
Lemma.
Any graph that admits an s-span weakly leveled planar drawing has edge-length ratio at most $2s + 1$.

Other bounds and edge ratio

Theorem.
Every treewidth-2 graph admits an $\mathcal{O}(\sqrt{n})$-span weakly leveled planar drawing.

Theorem.
There planar treewidth-2 graphs that require span $2^{\Omega(\sqrt{log n})}$ in any weakly leveled planar drawing.

Lemma.
Any graph that admits an s-span weakly leveled planar drawing has edge-length ratio at most $2s + 1$.

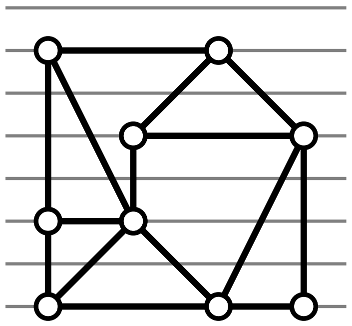


Other bounds and edge ratio

Theorem.
Every treewidth-2 graph admits an $\mathcal{O}(\sqrt{n})$-span weakly leveled planar drawing.

Theorem.
There planar treewidth-2 graphs that require span $2^{\Omega(\sqrt{log n})}$ in any weakly leveled planar drawing.

Lemma.
Any graph that admits an s-span weakly leveled planar drawing has edge-length ratio at most $2s + 1$.

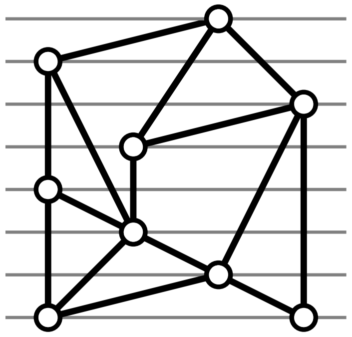


Other bounds and edge ratio

Theorem.
Every treewidth-2 graph admits an $\mathcal{O}(\sqrt{n})$-span weakly leveled planar drawing.

Theorem.
There planar treewidth-2 graphs that require span $2^{\Omega(\sqrt{log n})}$ in any weakly leveled planar drawing.

Lemma.
Any graph that admits an s-span weakly leveled planar drawing has edge-length ratio at most $2s + 1$.

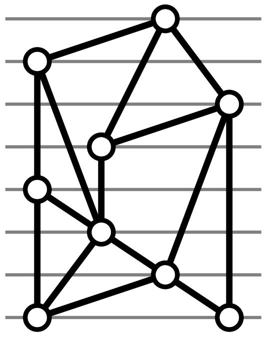


Other bounds and edge ratio

Theorem.
Every treewidth-2 graph admits an $\mathcal{O}(\sqrt{n})$-span weakly leveled planar drawing.

Theorem.
There planar treewidth-2 graphs that require span $2^{\Omega(\sqrt{\log n})}$ in any weakly leveled planar drawing.

Lemma.
Any graph that admits an s-span weakly leveled planar drawing has edge-length ratio at most $2s+1$.

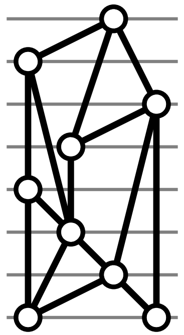


Other bounds and edge ratio

Theorem.
Every treewidth-2 graph admits an $\mathcal{O}(\sqrt{n})$-span weakly leveled planar drawing.

Theorem.
There planar treewidth-2 graphs that require span $2^{\Omega(\sqrt{logn})}$ in any weakly leveled planar drawing.

Lemma.
Any graph that admits an s-span weakly leveled planar drawing has edge-length ratio at most $2s + 1$.

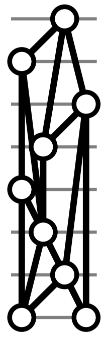


Other bounds and edge ratio

Theorem.
Every treewidth-2 graph admits an $\mathcal{O}(\sqrt{n})$-span weakly leveled planar drawing.

Theorem.
There planar treewidth-2 graphs that require span $2^{\Omega(\sqrt{log n})}$ in any weakly leveled planar drawing.

Lemma.
Any graph that admits an s-span weakly leveled planar drawing has edge-length ratio at most $2s + 1$.

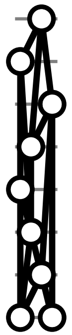


Other bounds and edge ratio

Theorem.
Every treewidth-2 graph admits an $\mathcal{O}(\sqrt{n})$-span weakly leveled planar drawing.

Theorem.
There planar treewidth-2 graphs that require span $2^{\Omega(\sqrt{log n})}$ in any weakly leveled planar drawing.

Lemma.
Any graph that admits an s-span weakly leveled planar drawing has edge-length ratio at most $2s + 1$.



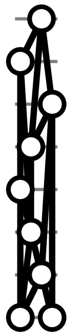
Other bounds and edge ratio

Theorem.
Every treewidth-2 graph admits an $\mathcal{O}(\sqrt{n})$-span weakly leveled planar drawing.

Theorem.
There planar treewidth-2 graphs that require span $2^{\Omega(\sqrt{\log n})}$ in any weakly leveled planar drawing.

Lemma.
Any graph that admits an s-span weakly leveled planar drawing has edge-length ratio at most $2s+1$.

Theorem.
Treewidth-2 graphs have planar edge-length ratio $\mathcal{O}(\sqrt{n})$.



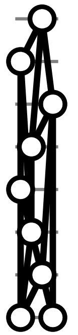
Other bounds and edge ratio

Theorem.
Every treewidth-2 graph admits an $\mathcal{O}(\sqrt{n})$-span weakly leveled planar drawing.

Theorem.
There planar treewidth-2 graphs that require span $2^{\Omega(\sqrt{\log n})}$ in any weakly leveled planar drawing.

Lemma.
Any graph that admits an s-span weakly leveled planar drawing has edge-length ratio at most $2s+1$.

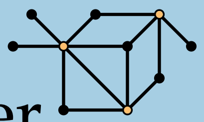
Theorem.
Treewidth-2 graphs have planar edge-length ratio $\mathcal{O}(\sqrt{n})$.

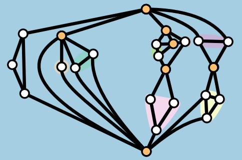


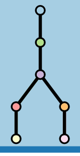
The previously best-known bound was $\mathcal{O}(n^{0.695})$.

Overview

FPT algorithms

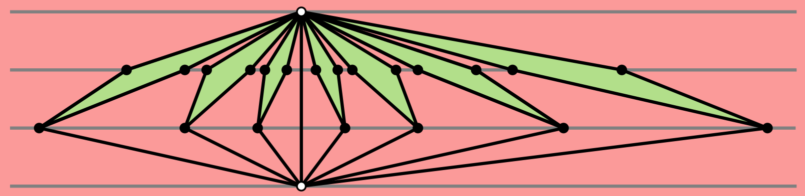
Linear kernel in size of vertex cover 

FPT in size of b-modulator $(+b)$ 

FPT in treedepth 

NP-hardness

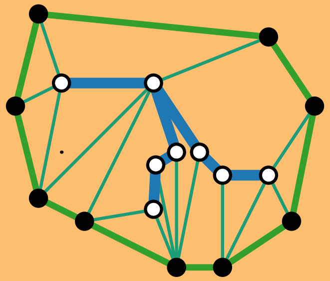
Reduction from leveled planar



Combinatorial results

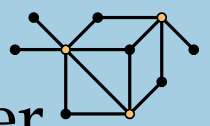
Graph class	**LB**	**UB**
2-outerplanar	$\Omega(n)$	n
3-connected cycle-tree	4	4
cycle-tree	$\Omega(\log n)$	$\mathcal{O}(\log n)$
treewidth 2	$2^{\Omega(\sqrt{\log n})}$	$\mathcal{O}(\sqrt{n})$

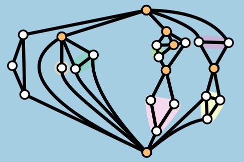
Implications

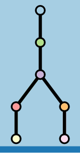


Overview

FPT algorithms

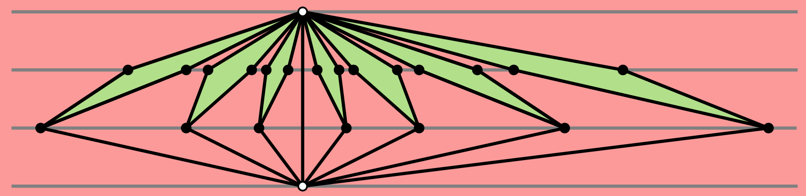
Linear kernel in size of vertex cover 

FPT in size of b-modulator $(+b)$ 

FPT in treedepth 

NP-hardness

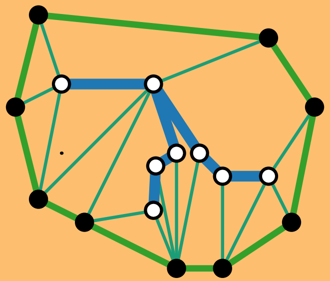
Reduction from leveled planar



Combinatorial results

Graph class	**LB**	**UB**
2-outerplanar	$\Omega(n)$	n
3-connected cycle-tree	4	4
cycle-tree	$\Omega(\log n)$	$\mathcal{O}(\log n)$
treewidth 2	$2^{\Omega(\sqrt{\log n})}$	$\mathcal{O}(\sqrt{n})$

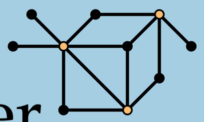
Implications

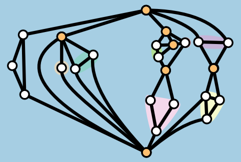


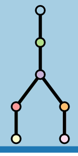
Treewidth-2 graphs have edge-length ratio $\mathcal{O}(\sqrt{n})$

Overview

FPT algorithms

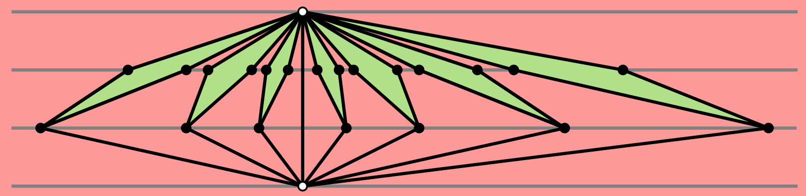
Linear kernel in size of vertex cover 

FPT in size of b-modulator $(+b)$ 

FPT in treedepth 

NP-hardness

Reduction from leveled planar

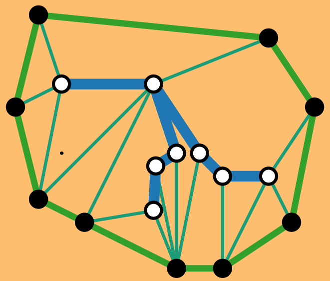


Combinatorial results

Graph class	**LB**	**UB**
2-outerplanar	$\Omega(n)$	n
3-connected cycle-tree	4	4
cycle-tree	$\Omega(\log n)$	$\mathcal{O}(\log n)$
treewidth 2	$2^{\Omega(\sqrt{\log n})}$	$\mathcal{O}(\sqrt{n})$

Implications

3-connected cycle-trees have edge-length ratio ≤ 9

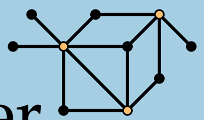


Treewidth-2 graphs have edge-length ratio $\mathcal{O}(\sqrt{n})$

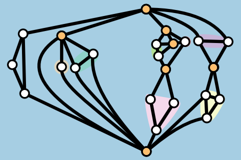
Overview

FPT algorithms

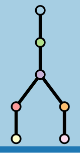
Linear kernel in size of vertex cover



FPT in size of b-modulator ($+b$)

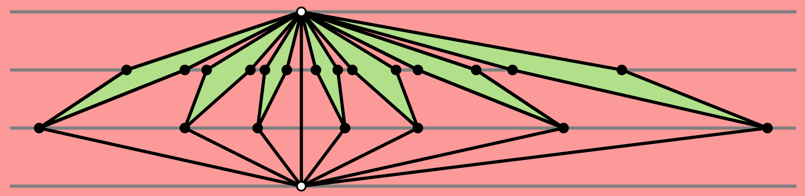


FPT in treedepth



NP-hardness

Reduction from leveled planar

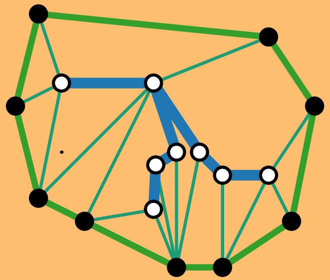


Combinatorial results

Graph class	**LB**	**UB**
2-outerplanar	$\Omega(n)$	n
3-connected cycle-tree	4	4
cycle-tree	$\Omega(\log n)$	$\mathcal{O}(\log n)$
treewidth 2	$2^{\Omega(\sqrt{\log n})}$	$\mathcal{O}(\sqrt{n})$

Implications

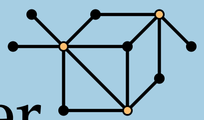
3-connected cycle-trees have
edge-length ratio ≤ 9
queue number ≤ 5

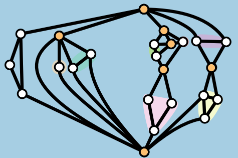


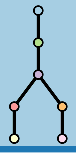
Treewidth-2 graphs have
edge-length ratio $\mathcal{O}(\sqrt{n})$

Overview

FPT algorithms

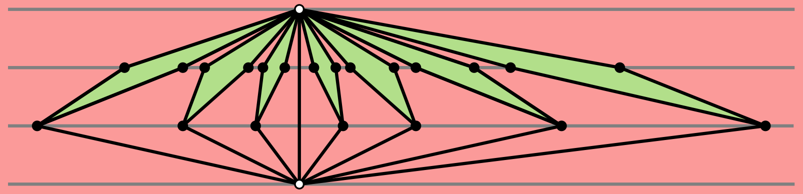
Linear kernel in size of vertex cover 

FPT in size of b-modulator $(+b)$ 

FPT in treedepth  Poly. kernel?

NP-hardness

Reduction from leveled planar

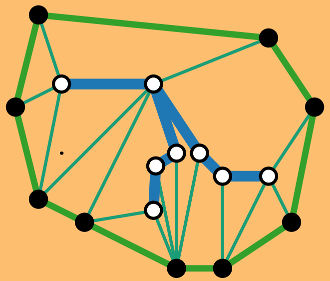


Combinatorial results

Graph class	LB	UB
2-outerplanar	$\Omega(n)$	n
3-connected cycle-tree	4	4
cycle-tree	$\Omega(\log n)$	$\mathcal{O}(\log n)$
treewidth 2	$2^{\Omega(\sqrt{\log n})}$	$\mathcal{O}(\sqrt{n})$

Implications

3-connected cycle-trees have



edge-length ratio ≤ 9

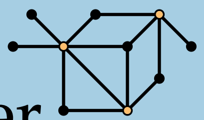
queue number ≤ 5

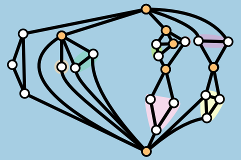
Treewidth-2 graphs have

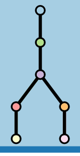
edge-length ratio $\mathcal{O}(\sqrt{n})$

Overview

FPT algorithms

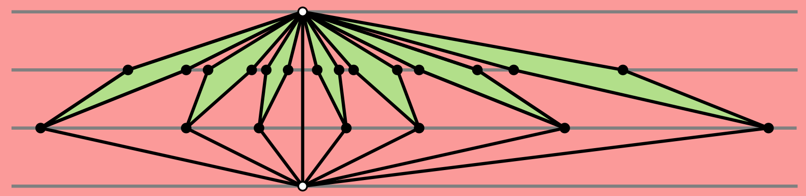
Linear kernel in size of vertex cover 

FPT in size of b-modulator $(+b)$ 

FPT in treedepth  Poly. kernel? Treewidth?

NP-hardness

Reduction from leveled planar

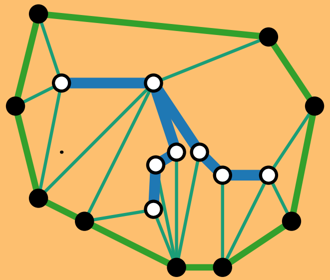


Combinatorial results

Graph class	LB	UB
2-outerplanar	$\Omega(n)$	n
3-connected cycle-tree	4	4
cycle-tree	$\Omega(\log n)$	$\mathcal{O}(\log n)$
treewidth 2	$2^{\Omega(\sqrt{\log n})}$	$\mathcal{O}(\sqrt{n})$

Implications

3-connected cycle-trees have



edge-length ratio ≤ 9

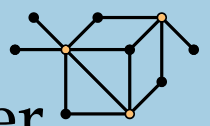
queue number ≤ 5

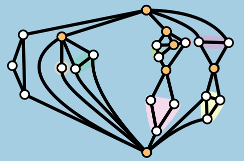
Treewidth-2 graphs have

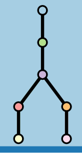
edge-length ratio $\mathcal{O}(\sqrt{n})$

Overview

FPT algorithms

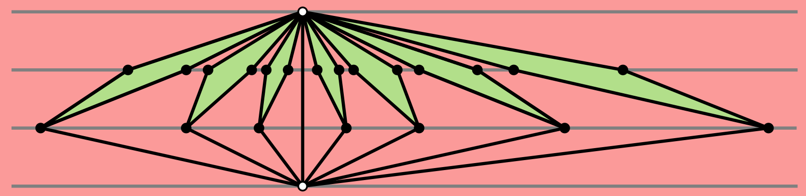
Linear kernel in size of vertex cover 

FPT in size of b-modulator $(+b)$ 

FPT in treedepth  Poly. kernel? Treewidth?

NP-hardness

Reduction from leveled planar



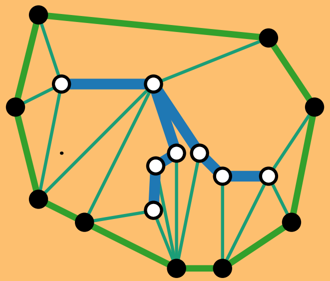
Combinatorial results

Graph class	**LB**	**UB**
2-outerplanar	$\Omega(n)$	n
3-connected cycle-tree	4	4
cycle-tree	$\Omega(\log n)$	$\mathcal{O}(\log n)$
treewidth 2	$2^{\Omega(\sqrt{\log n})}$	$\mathcal{O}(\sqrt{n})$

?

Implications

3-connected cycle-trees have



edge-length ratio ≤ 9

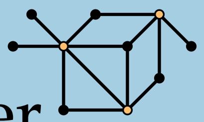
queue number ≤ 5

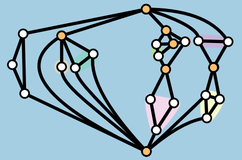
Treewidth-2 graphs have

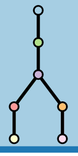
edge-length ratio $\mathcal{O}(\sqrt{n})$

Overview

FPT algorithms

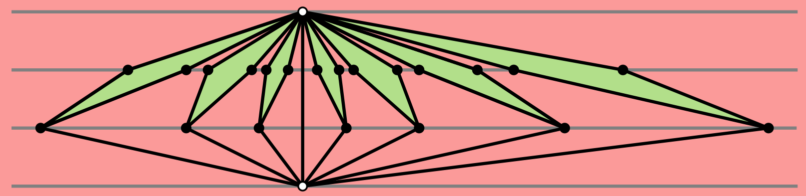
Linear kernel in size of vertex cover 

FPT in size of b-modulator $(+b)$ 

FPT in treedepth  Poly. kernel? Treewidth?

NP-hardness

Reduction from leveled planar



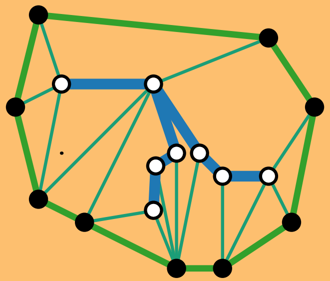
Combinatorial results

Graph class	**LB**	**UB**
2-outerplanar	$\Omega(n)$	n
3-connected cycle-tree	4	4
cycle-tree	$\Omega(\log n)$	$\mathcal{O}(\log n)$
treewidth 2	$2^{\Omega(\sqrt{\log n})}$	$\mathcal{O}(\sqrt{n})$

?

Implications

3-connected cycle-trees have



edge-length ratio ≤ 9

queue number ≤ 5

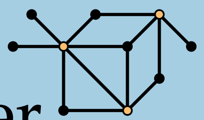
Treewidth-2 graphs have

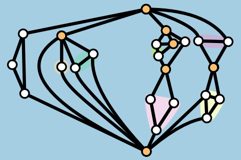
edge-length ratio $\mathcal{O}(\sqrt{n})$

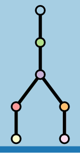
$\Omega(\log n)$

Overview

FPT algorithms

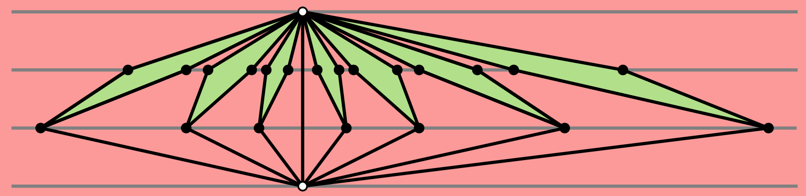
Linear kernel in size of vertex cover 

FPT in size of b-modulator $(+b)$ 

FPT in treedepth  Poly. kernel? Treewidth?

NP-hardness

Reduction from leveled planar



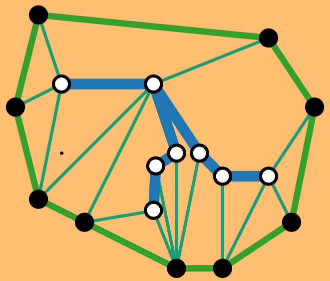
Combinatorial results

Graph class	**LB**	**UB**
2-outerplanar	$\Omega(n)$	n
3-connected cycle-tree	4	4
cycle-tree	$\Omega(\log n)$	$\mathcal{O}(\log n)$
treewidth 2	$2^{\Omega(\sqrt{\log n})}$	$\mathcal{O}(\sqrt{n})$

?

Implications

3-connected cycle-trees have



edge-length ratio ≤ 9

queue number ≤ 5

Treewidth-2 graphs have

edge-length ratio $\mathcal{O}(\sqrt{n})$

$\Omega(\log n)$?