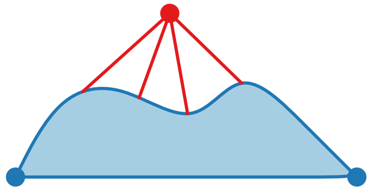
Visualisation of graphs

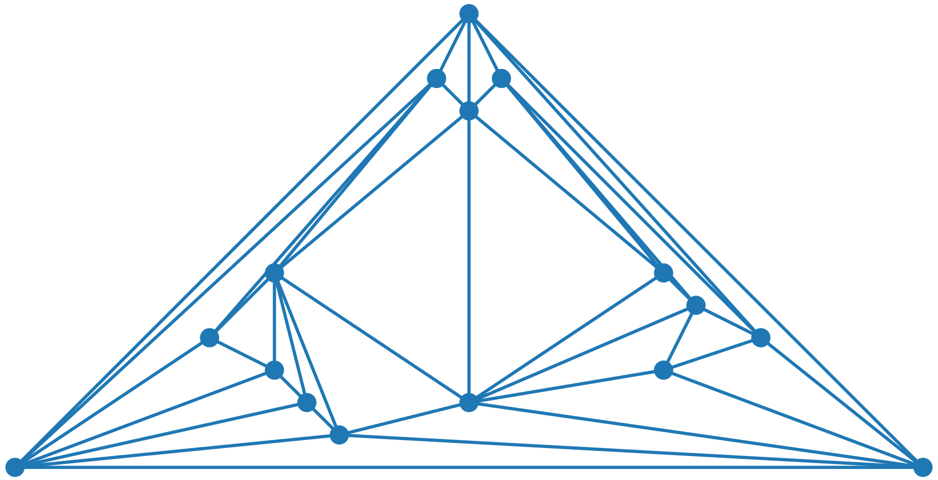
Planar straight-line drawings

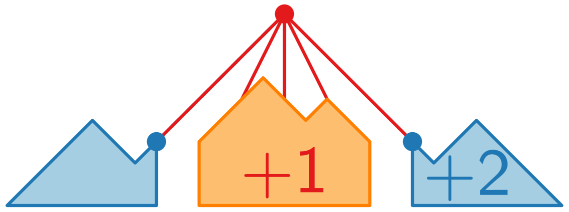
Canonical order

Antonios Symvonis · Chrysanthi Raftopoulou

Fall semester 2022







The original slides of this presentation were created by researchers at Karlsruhe Institute of Technology (KIT), TU Wien, U Wuerzburg, U Konstanz, ...
The original presentation was modified/updated by A. Symvonis and C. Raftopoulou

Motivation

- So far we looked at planar and straight-line drawings of trees and series-parallel graphs.

Motivation

- So far we looked at planar and straight-line drawings of trees and series-parallel graphs.
- Why straight-line? Why planar?

Motivation

- So far we looked at planar and straight-line drawings of trees and series-parallel graphs.
- Why straight-line? Why planar?
- Bennett, Ryall, Spaltzeholz and Gooch, 2007
  "The Aesthetics of Graph Visualization"

3.2. Edge Placement Heuristics

By far the most agreed-upon edge placement heuristic is to *minimize the number of edge crossings* in a graph [BMRW98, Har98, DH96, Pur02, TR05, TBB88]. The importance of avoiding edge crossings has also been extensively validated in terms of user preference and performance (see Section 4). Similarly, based on perceptual principles, it is beneficial to *minimize the number of edge bends* within a graph [Pur02, TR05, TBB88]. Edge bends make edges more difficult to follow because an edge with a sharp bend is more likely to be perceived as two separate objects. This leads to the heuristic of *keeping edge bends uniform* with respect to the bend's position on the edge and its angle [TR05]. If an edge must be bent to satisfy other aesthetic criteria, the angle of the bend should be as little as possible, and the bend placement should evenly divide the edge.

Motivation

- So far we looked at planar and straight-line drawings of trees and series-parallel graphs.
- Why straight-line? Why planar?
- Bennett, Ryall, Spaltzeholz and Gooch, 2007
  "The Aesthetics of Graph Visualization"

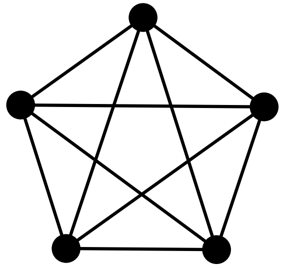
3.2. Edge Placement Heuristics

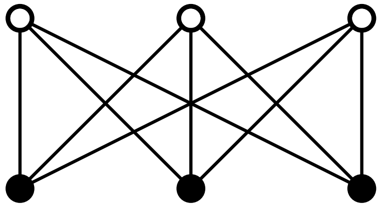
By far the most agreed-upon edge placement heuristic is to *minimize the number of edge crossings* in a graph [BMRW98, Har98, DH96, Pur02, TR05, TBB88]. The importance of avoiding edge crossings has also been extensively validated in terms of user preference and performance (see Section 4). Similarly, based on perceptual principles, it is beneficial to *minimize the number of edge bends* within a graph [Pur02, TR05, TBB88]. Edge bends make edges more difficult to follow because an edge with a sharp bend is more likely to be perceived as two separate objects. This leads to the heuristic of *keeping edge bends uniform* with respect to the bend's position on the edge and its angle [TR05]. If an edge must be bent to satisfy other aesthetic criteria, the angle of the bend should be as little as possible, and the bend placement should evenly divide the edge.

- crossings reduce readability
- bends reduce readability

Planar graphs

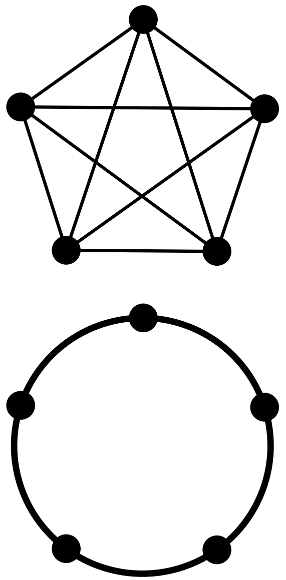
- **Characterisation:** A graph is **planar** iff it contains neither a K_5 nor a $K_{3,3}$ minor.
  [Kuratowski 1930, Wagner 1936]

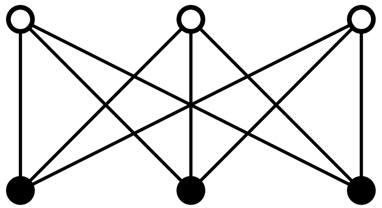




Planar graphs

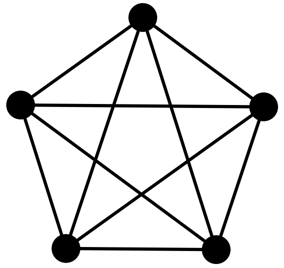
- **Characterisation:** A graph is **planar** iff it contains neither a K_5 nor a $K_{3,3}$ minor. [Kuratowski 1930, Wagner 1936]

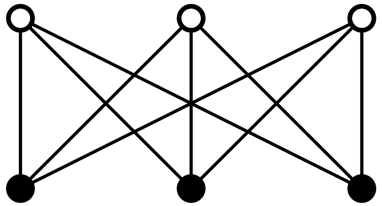


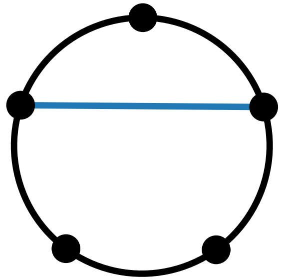


Planar graphs

- **Characterisation:** A graph is **planar** iff it contains neither a K_5 nor a $K_{3,3}$ minor. [Kuratowski 1930, Wagner 1936]

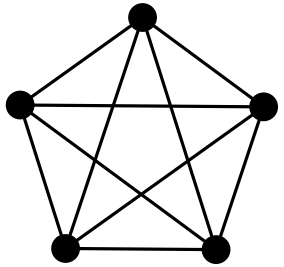


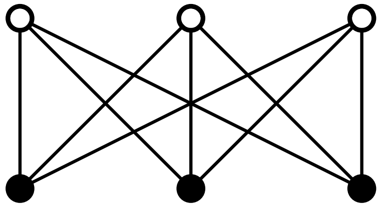


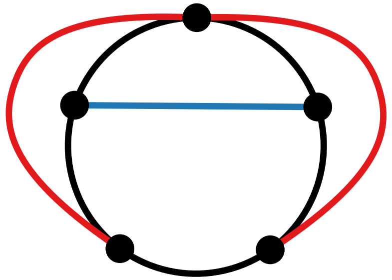


Planar graphs

- **Characterisation:** A graph is **planar** iff it contains neither a K_5 nor a $K_{3,3}$ minor.
  [Kuratowski 1930, Wagner 1936]

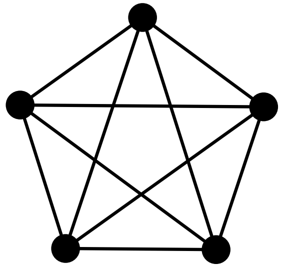


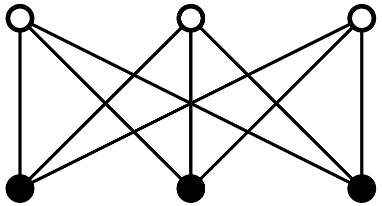


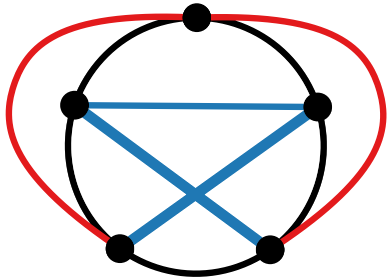


Planar graphs

- **Characterisation:** A graph is **planar** iff it contains neither a K_5 nor a $K_{3,3}$ minor. [Kuratowski 1930, Wagner 1936]

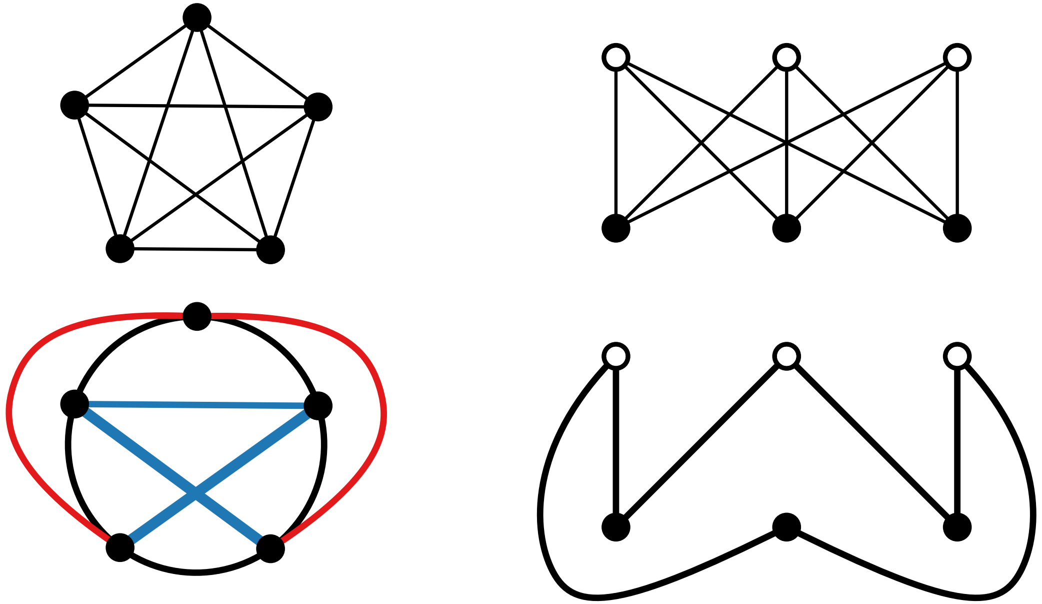






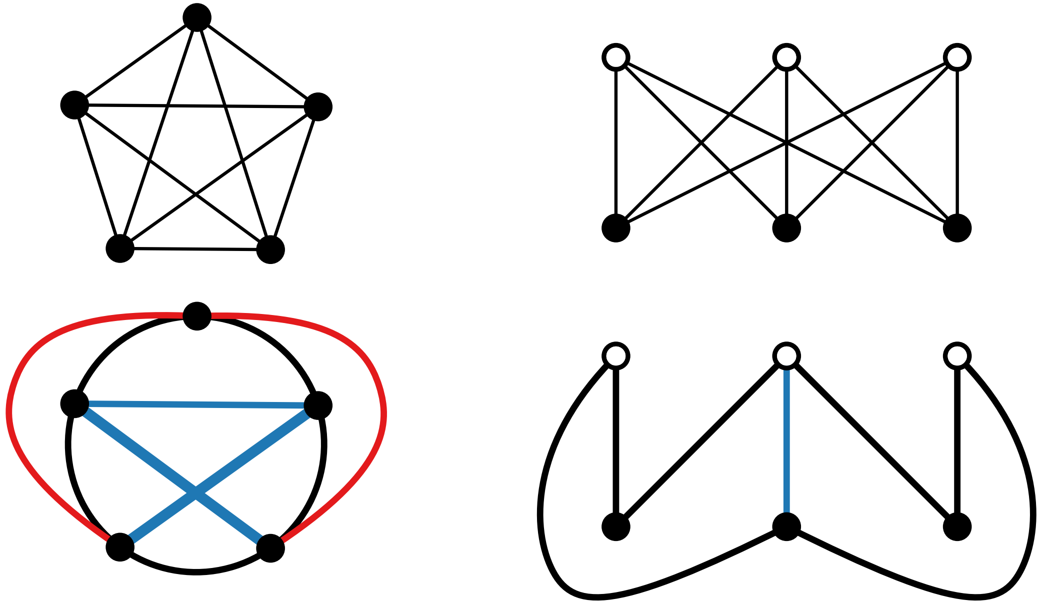
Planar graphs

- **Characterisation:** A graph is **planar** iff it contains neither a K_5 nor a $K_{3,3}$ minor.
  [Kuratowski 1930, Wagner 1936]



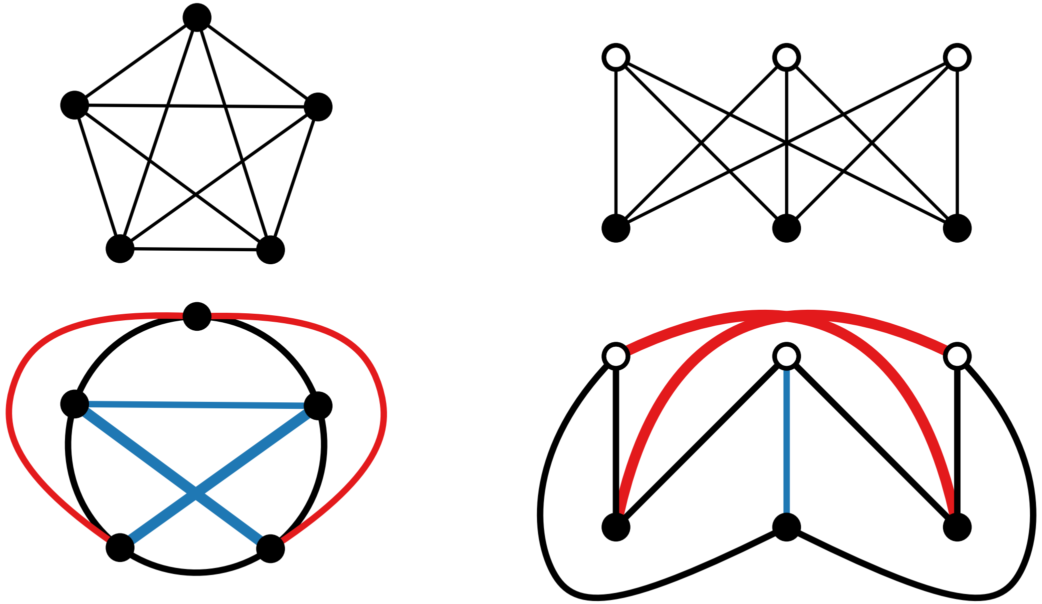
Planar graphs

- **Characterisation:** A graph is **planar** iff it contains neither a K_5 nor a $K_{3,3}$ minor.
  [Kuratowski 1930, Wagner 1936]



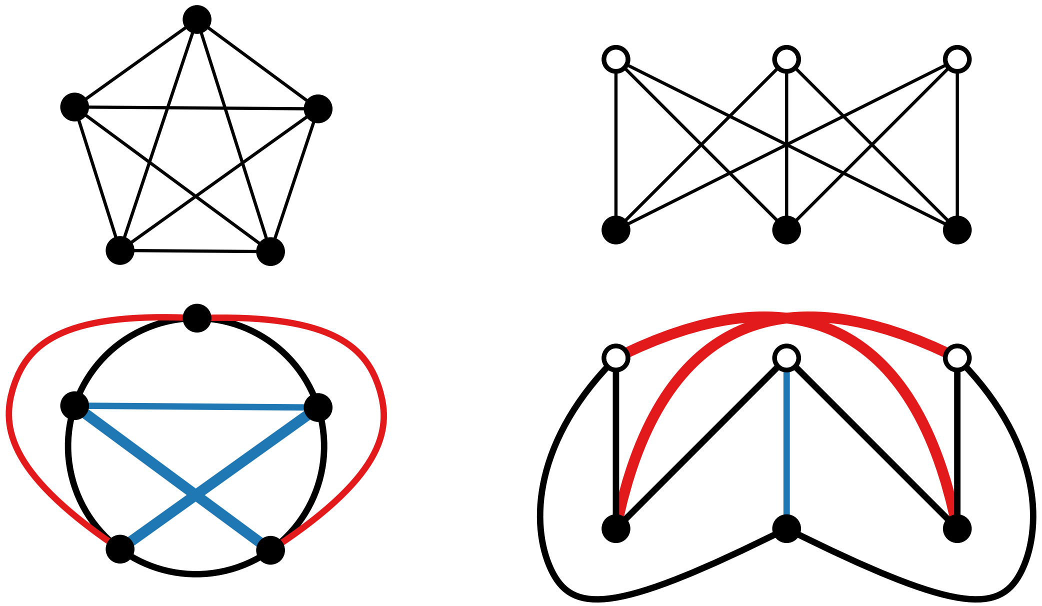
Planar graphs

- **Characterisation:** A graph is **planar** iff it contains neither a K_5 nor a $K_{3,3}$ minor.
  [Kuratowski 1930, Wagner 1936]



Planar graphs

- **Characterisation:** A graph is **planar** iff it contains neither a K_5 nor a $K_{3,3}$ minor. [Kuratowski 1930, Wagner 1936]



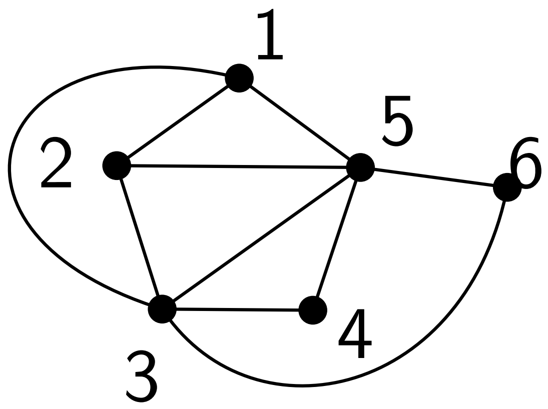
- **Recognition:** For a graph G with n vertices, there is an $\mathcal{O}(n)$ time algorithm to test if G is planar. [Hopcroft & Tarjan 1974]
  - Also computes an *embedding* in $\mathcal{O}(n)$.

Planar graphs

- **Embedding of planar graph:**
  - clockwise circular order of the edges incident to each vertex
  - outerface (clockwise order of edges)

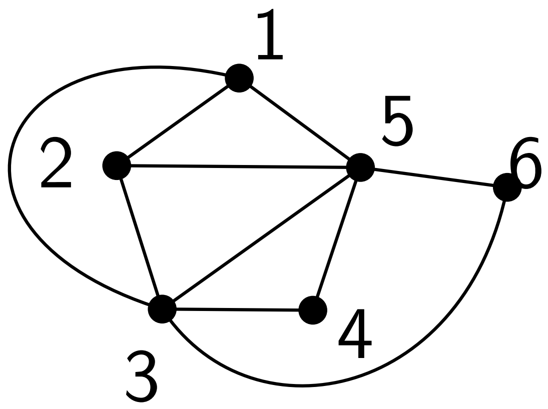
Planar graphs

- **Embedding of planar graph:**
  - clockwise circular order of the edges incident to each vertex
  - outerface (clockwise order of edges)

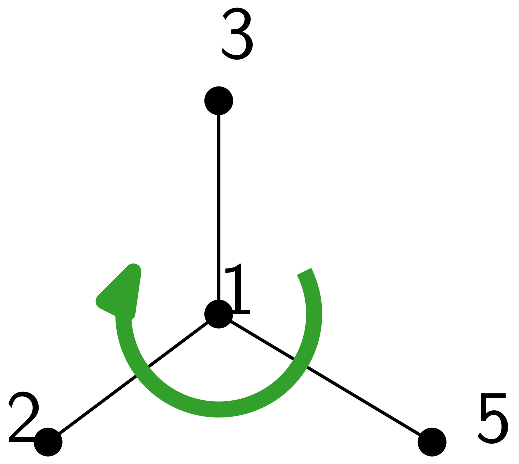


Planar graphs

- **Embedding of planar graph:**
  - clockwise circular order of the edges incident to each vertex
  - outerface (clockwise order of edges)

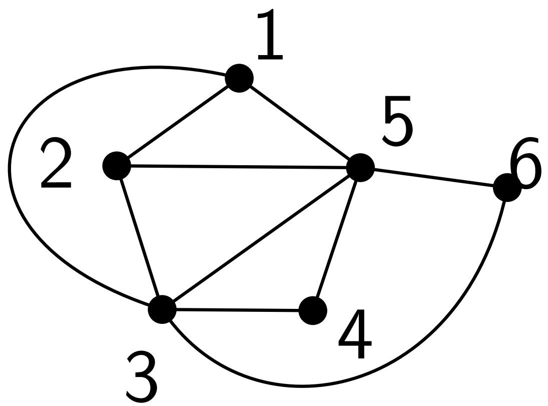


- **Edges:**
  $1 : \{(1,5),(1,2),(1,3)\}$



Planar graphs

- **Embedding of planar graph:**
  - clockwise circular order of the edges incident to each vertex
  - outerface (clockwise order of edges)

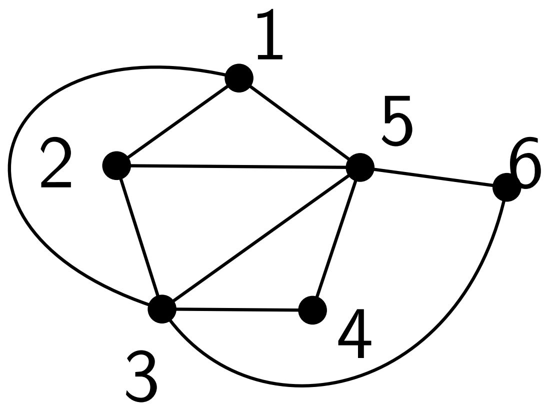


- **Edges:**

$1 : \{(1,5),(1,2),(1,3)\}$
$2 : \{(2,1),(2,5),(2,3)\}$
$3 : \{(3,1),(3,2),(3,5),(3,4),(3,6)\}$
$4 : \{(4,3),(4,5)\}$
$5 : \{(5,6),(5,4),(5,3),(5,2),(5,1)\}$
$6 : \{(6,3),(6,5)\}$

Planar graphs

- **Embedding of planar graph:**
  - clockwise circular order of the edges incident to each vertex
  - outerface (clockwise order of edges)



- **Edges:**

  $1 : \{(1,5),(1,2),(1,3)\}$
  $2 : \{(2,1),(2,5),(2,3)\}$
  $3 : \{(3,1),(3,2),(3,5),(3,4),(3,6)\}$
  $4 : \{(4,3),(4,5)\}$
  $5 : \{(5,6),(5,4),(5,3),(5,2),(5,1)\}$
  $6 : \{(6,3),(6,5)\}$

- **Outerface:**

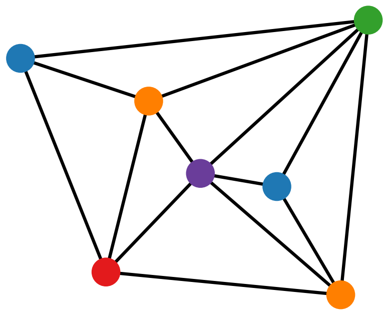
  $1 : \{(1,3),(3,6),(6,5),(5,1)\}$

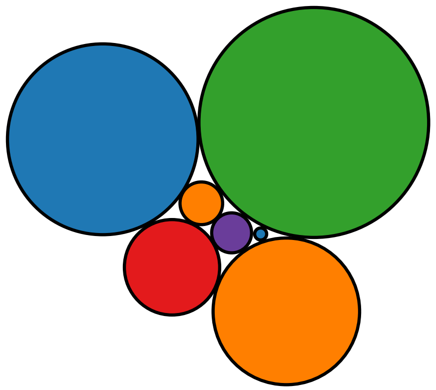
Planar graphs

- **Straight-line drawing:** Every planar graph has an embedding where the edges are straight-line segments. [Wagner 1936, Fáry 1948, Stein 1951]
  - The algorithms implied by this theory produce drawings with area *not bounded* by any polynomial on n.

Planar graphs

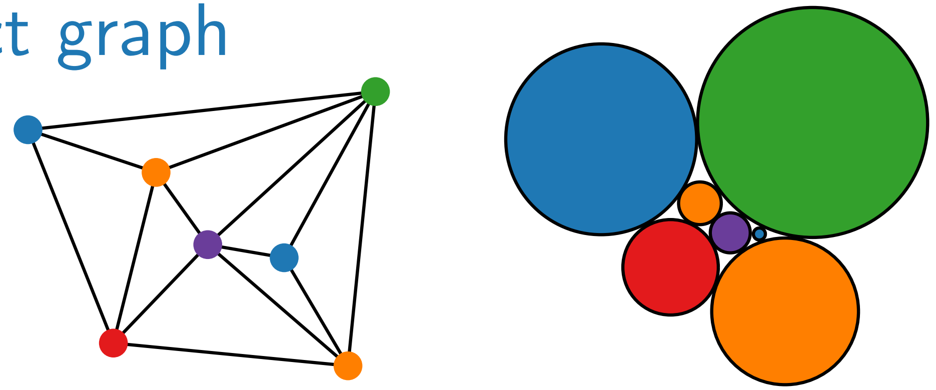
- **Straight-line drawing:** Every planar graph has an embedding where the edges are straight-line segments. [Wagner 1936, Fáry 1948, Stein 1951]
  - The algorithms implied by this theory produce drawings with area *not bounded* by any polynomial on n.
- **Coin graph:** Every planar graph is a circle contact graph (implies straight-line drawing). [Koebe 1936]





Planar graphs

- **Straight-line drawing:** Every planar graph has an embedding where the edges are straight-line segments. [Wagner 1936, Fáry 1948, Stein 1951]
  - The algorithms implied by this theory produce drawings with area *not bounded* by any polynomial on n.
- **Coin graph:** Every planar graph is a circle contact graph (implies straight-line drawing). [Koebe 1936]



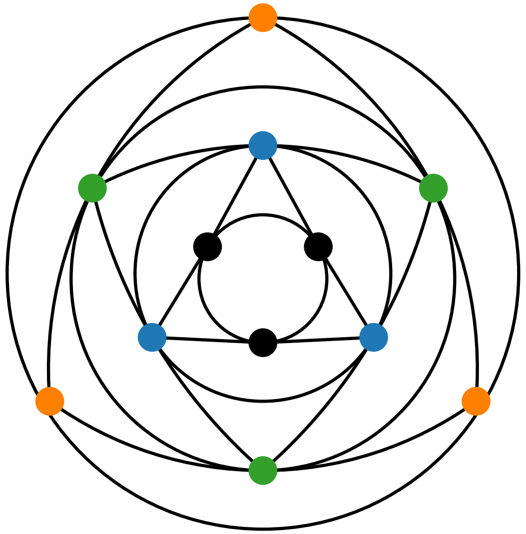
- Every 3-connected planar graph has an embedding with convex polygons as its faces (i.e., implies straight lines). [Tutte 1963: How to draw a graph]
  - Idea: Place vertices in the barycentre of neighbours.
  - Drawback: Requires large grids.

Planar graphs

- **Coin graph:**
  Exponential area

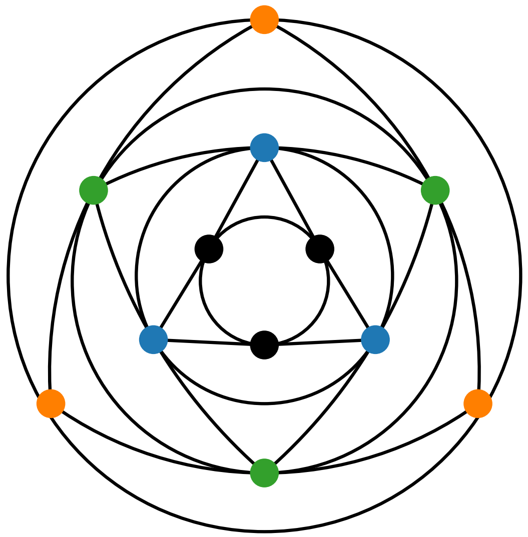
Planar graphs

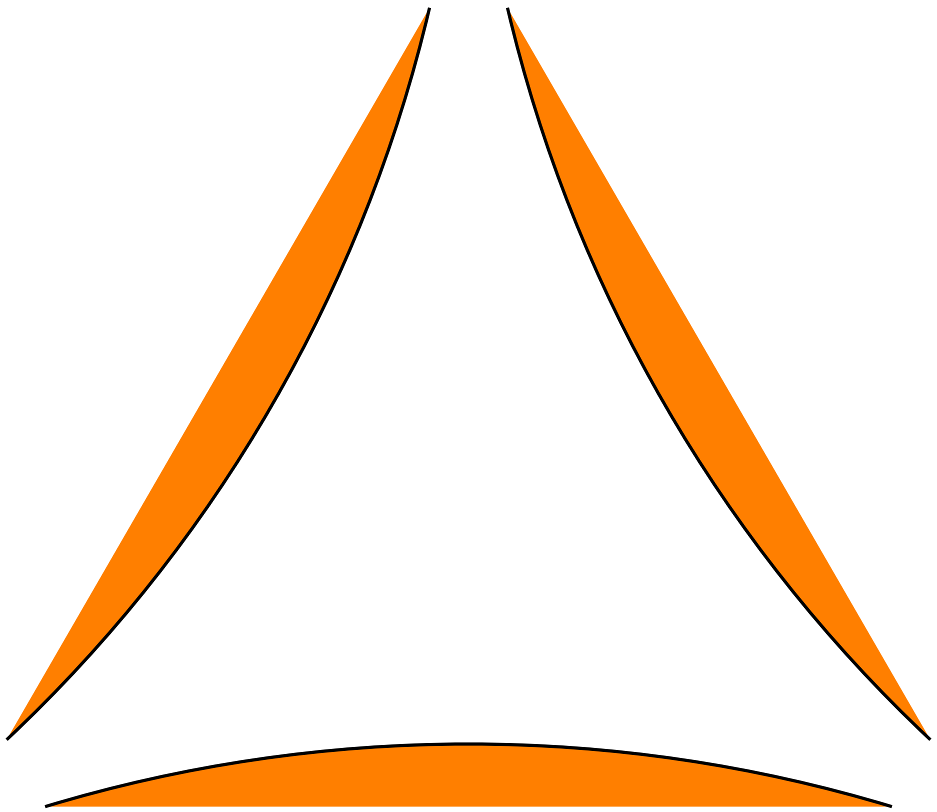
- **Coin graph:**
  Exponential area



Planar graphs

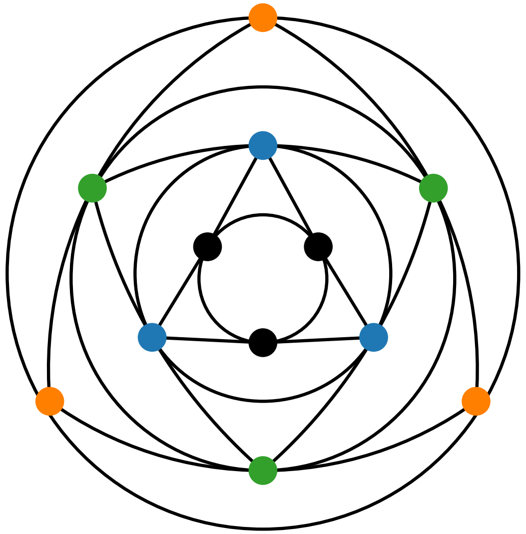
- **Coin graph:**
  Exponential area

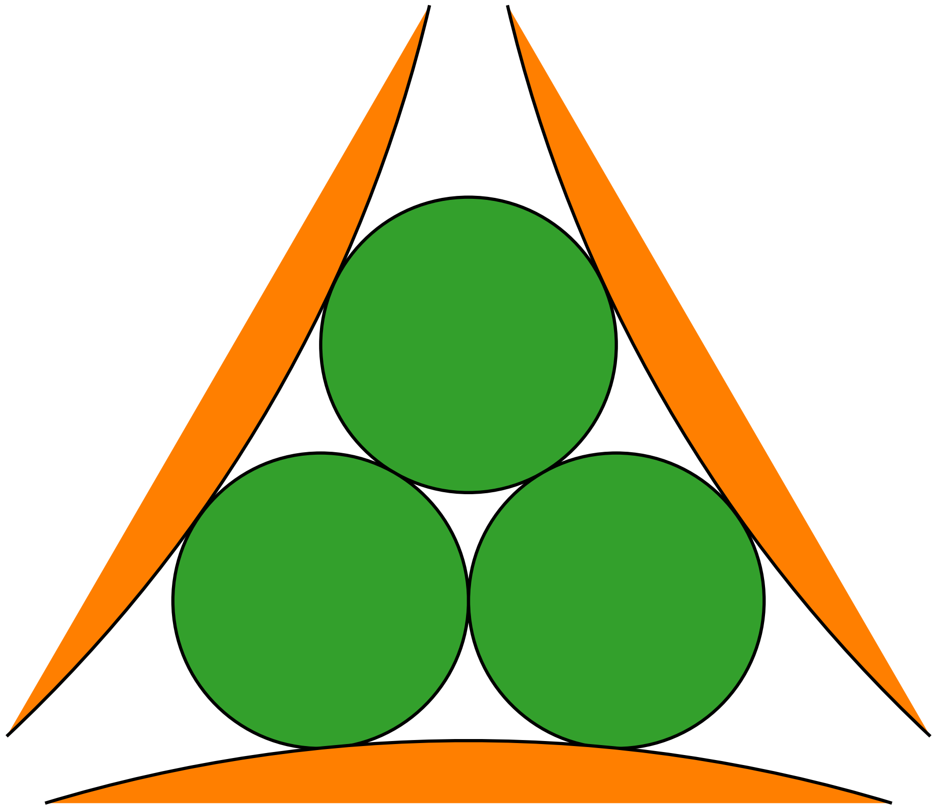




Planar graphs

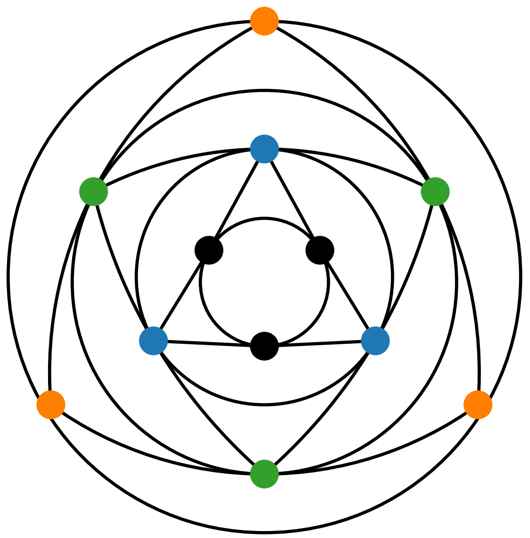
- **Coin graph:**
  Exponential area

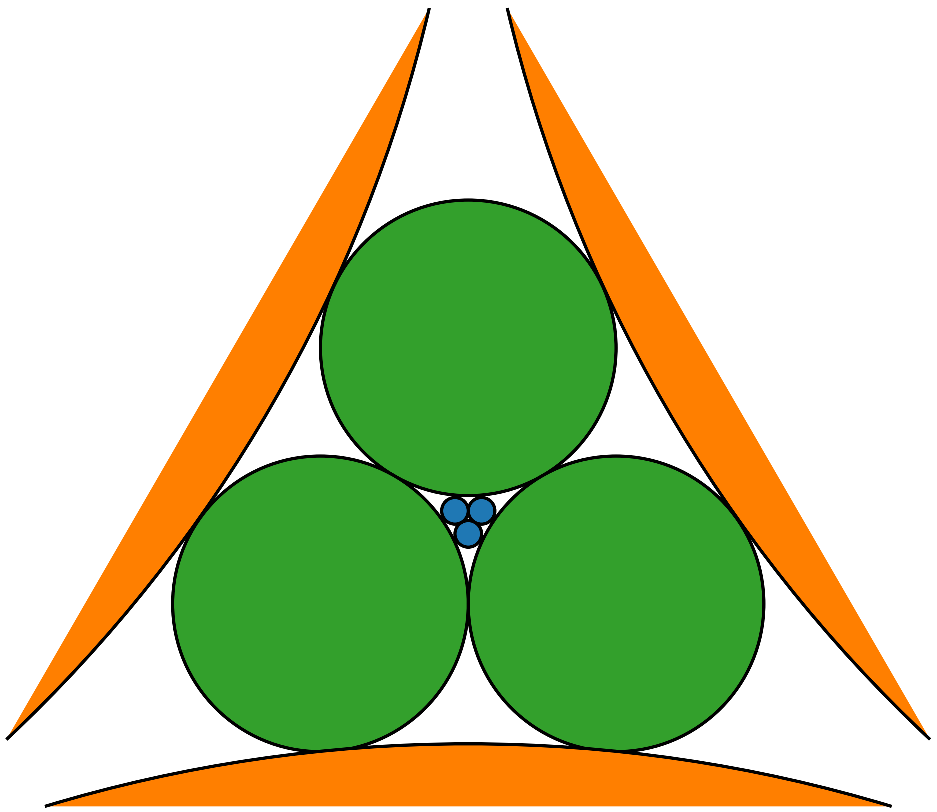




Planar graphs

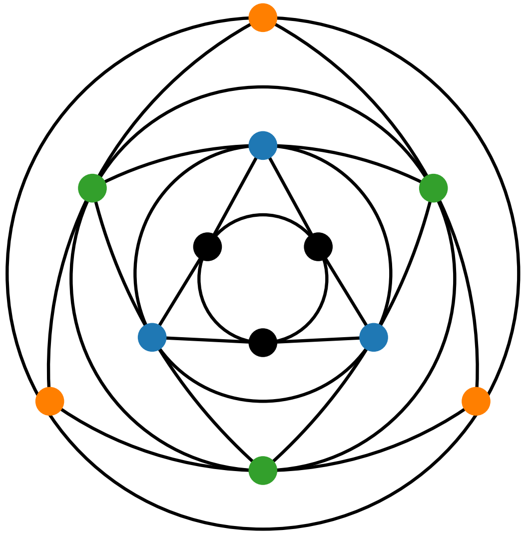
- **Coin graph:**
  Exponential area

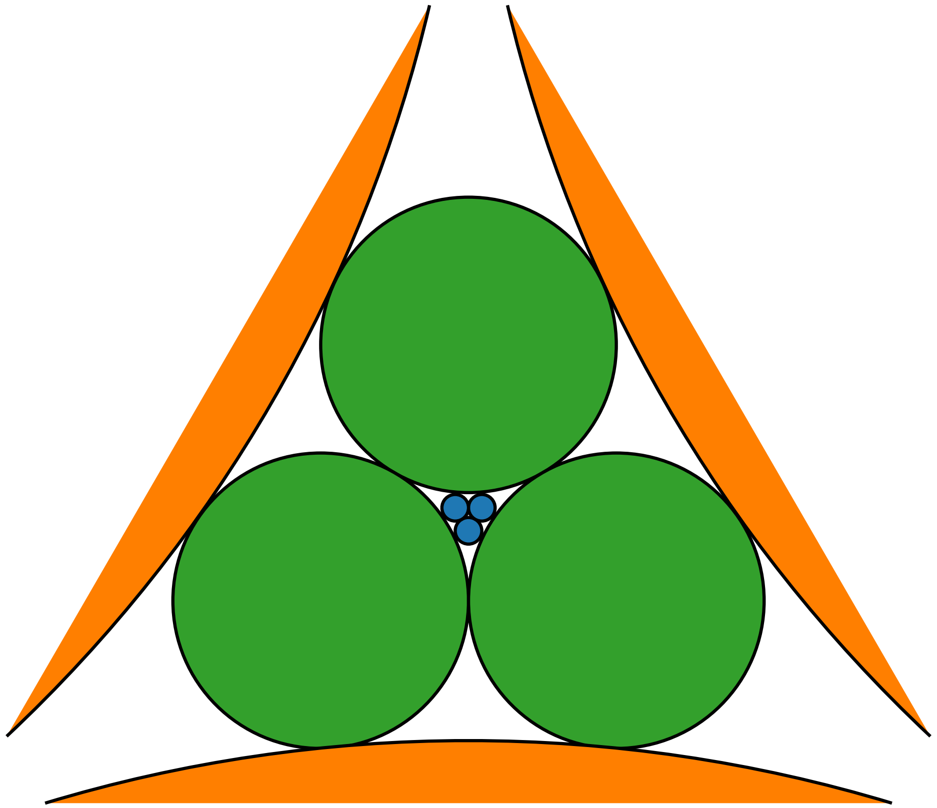




Planar graphs

- **Coin graph:**
  Exponential area

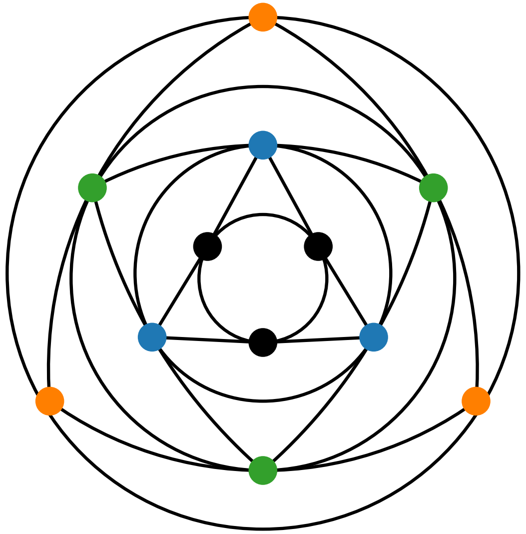


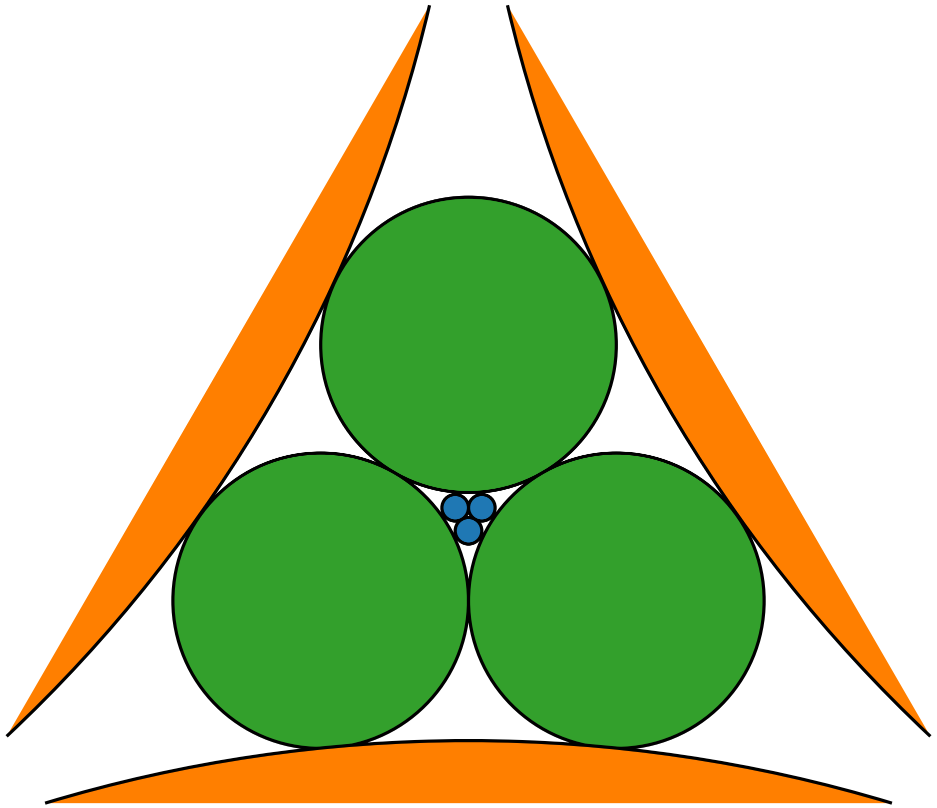


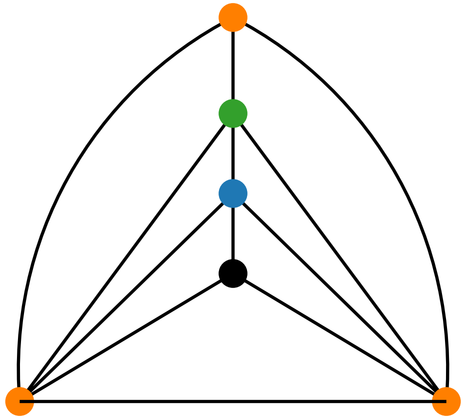
- **Barycentric representation:**
  Exponential area

Planar graphs

- **Coin graph:**
  Exponential area



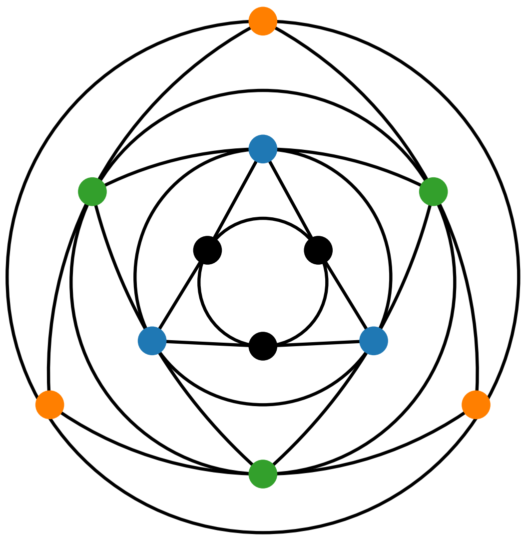


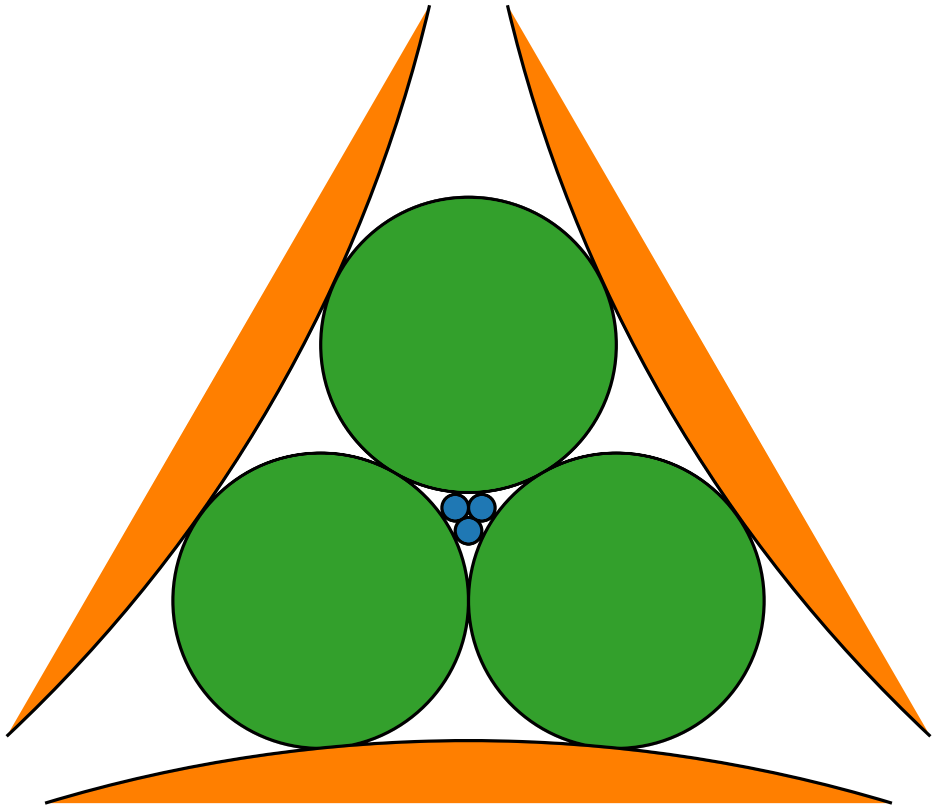


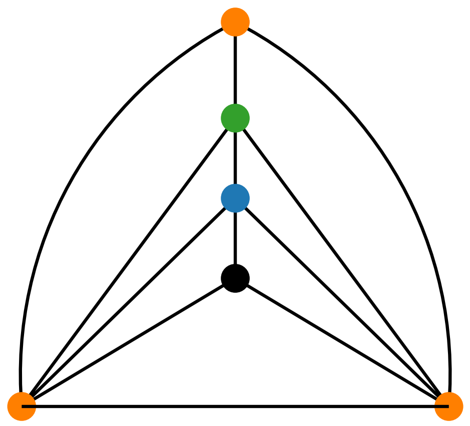
- **Barycentric representation:**
  Exponential area

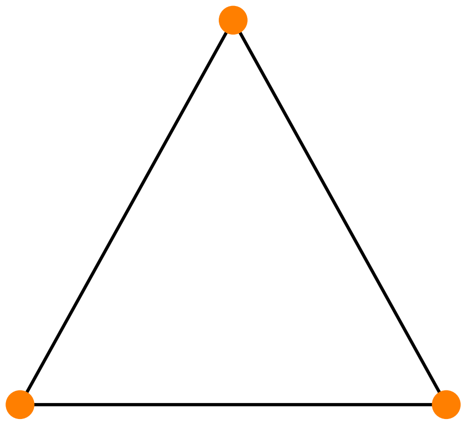
Planar graphs

- **Coin graph:**
  Exponential area





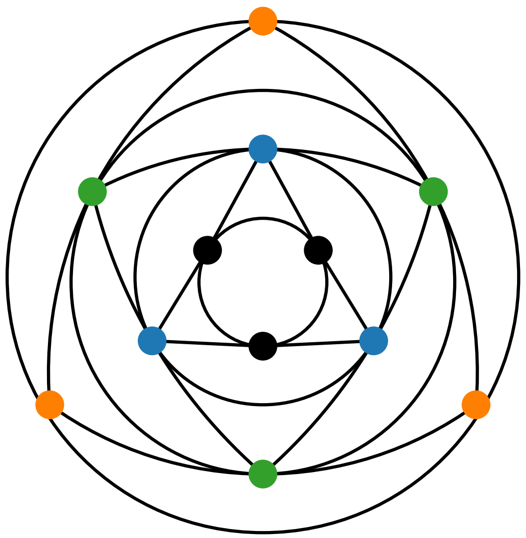


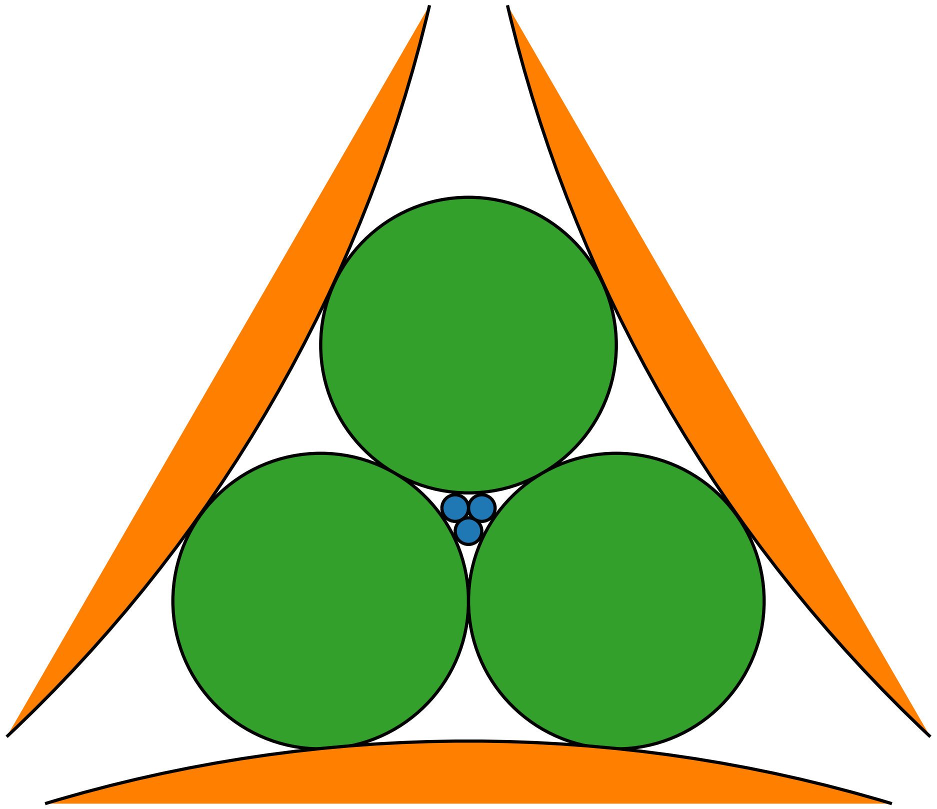


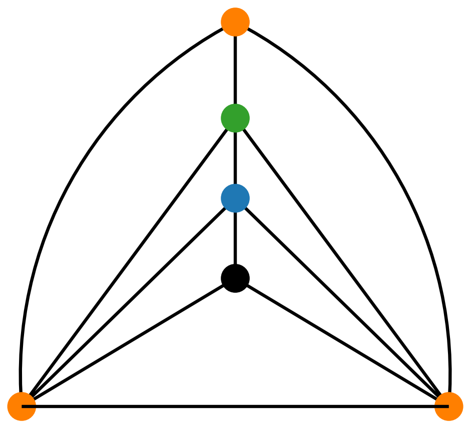
- **Barycentric representation:**
  Exponential area

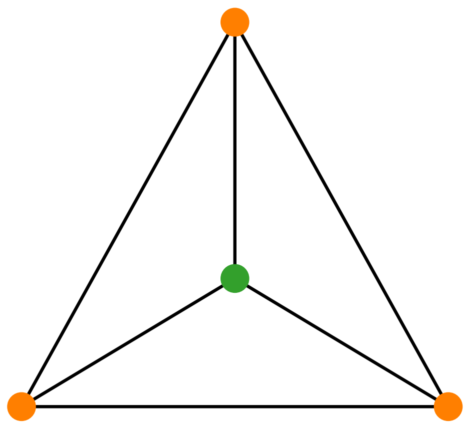
Planar graphs

- **Coin graph:**
  Exponential area





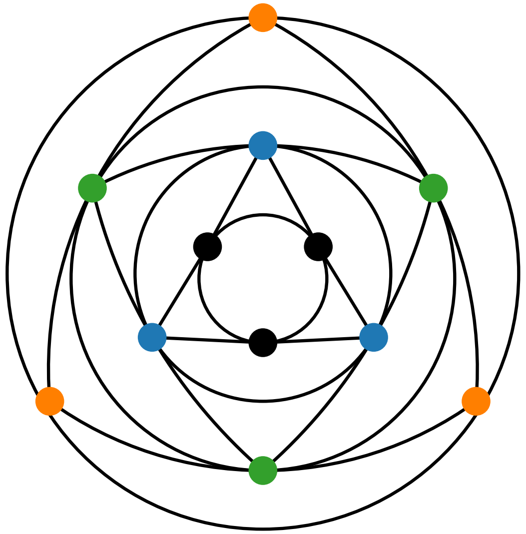


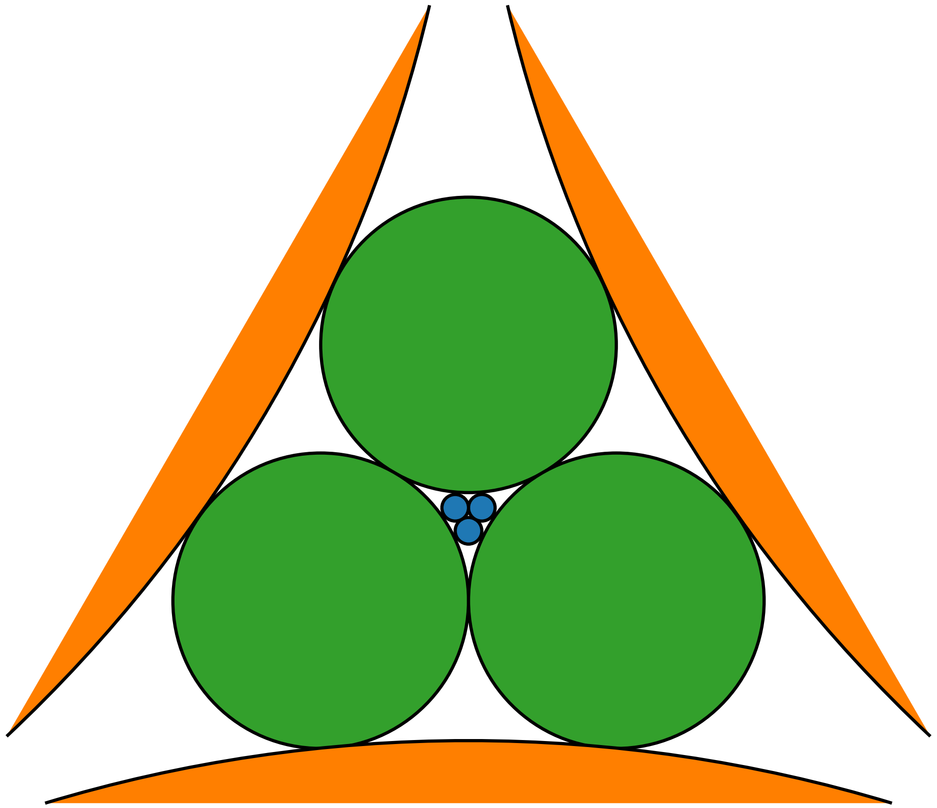


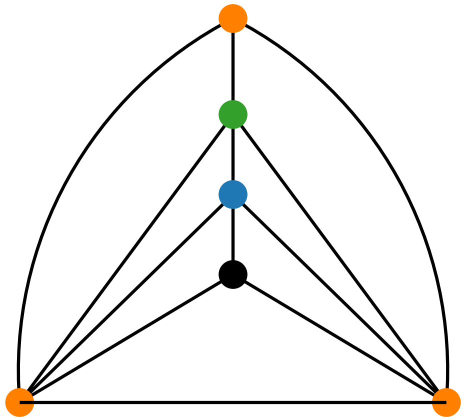
- **Barycentric representation:**
  Exponential area

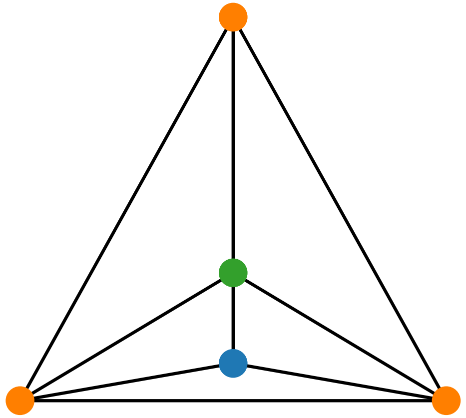
Planar graphs

- **Coin graph:**
  Exponential area





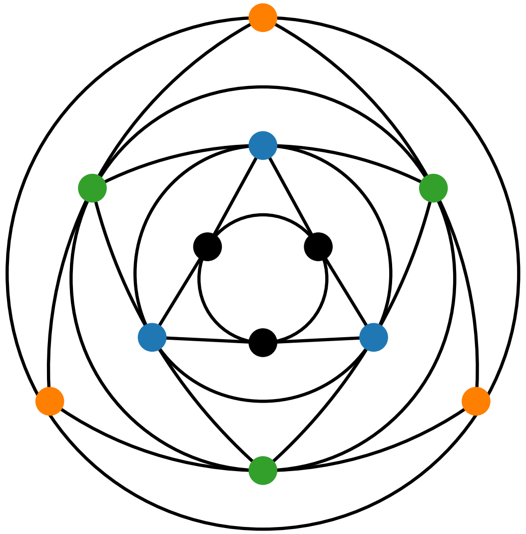


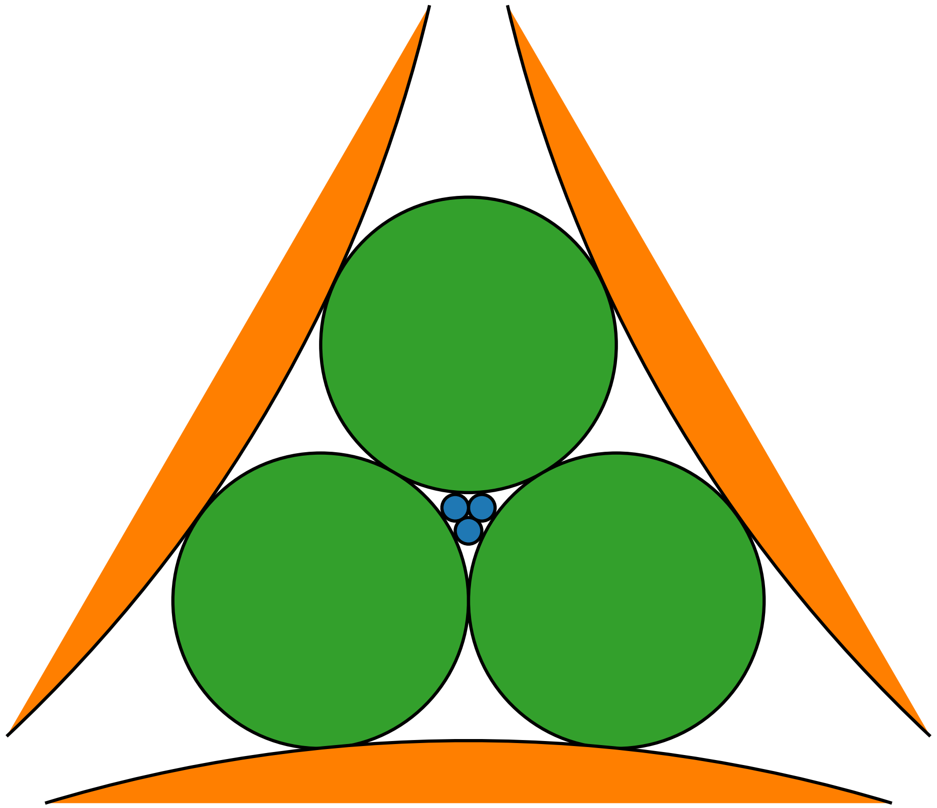


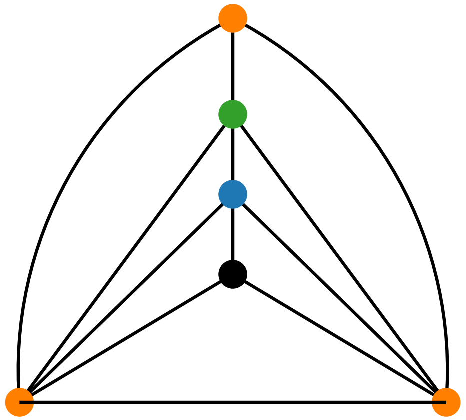
- **Barycentric representation:**
  Exponential area

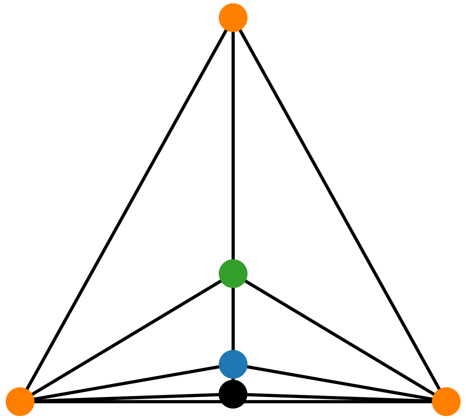
Planar graphs

- **Coin graph:**
  Exponential area









- **Barycentric representation:**
  Exponential area

Planar graphs

- Every planar graph has at most $3n - 6$ edges
- A *planar triangulation* is a planar graph with $3n - 6$ edges

Planar graphs

- Every planar graph has at most $3n - 6$ edges
- A *planar triangulation* is a planar graph with $3n - 6$ edges

- **Properties of planar triangulations:**
  - Every face is a triangle
  - graph is 3-connected
  - Unique embedding (up to choice of outerface)
  - Every plane graph is subgraph of a plane triangulation

Planar graphs

- Every planar graph has at most $3n - 6$ edges
- A *planar triangulation* is a planar graph with $3n - 6$ edges

- **Properties of planar triangulations:**
  - Every face is a triangle
  - graph is 3-connected
  - Unique embedding (up to choice of outerface)
  - Every plane graph is subgraph of a plane triangulation

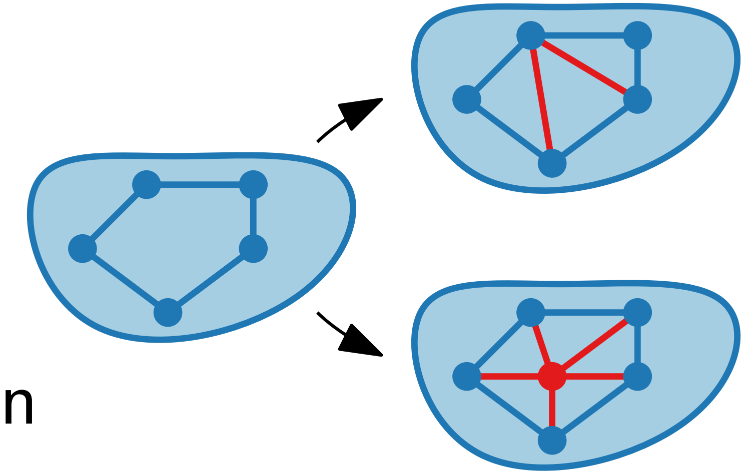
with planar embedding

Planar graphs

- Every planar graph has at most $3n - 6$ edges
- A *planar triangulation* is a planar graph with $3n - 6$ edges

- **Properties of planar triangulations:**
  - Every face is a triangle
  - graph is 3-connected
  - Unique embedding (up to choice of outerface)
  - Every plane graph is subgraph of a plane triangulation

with planar embedding

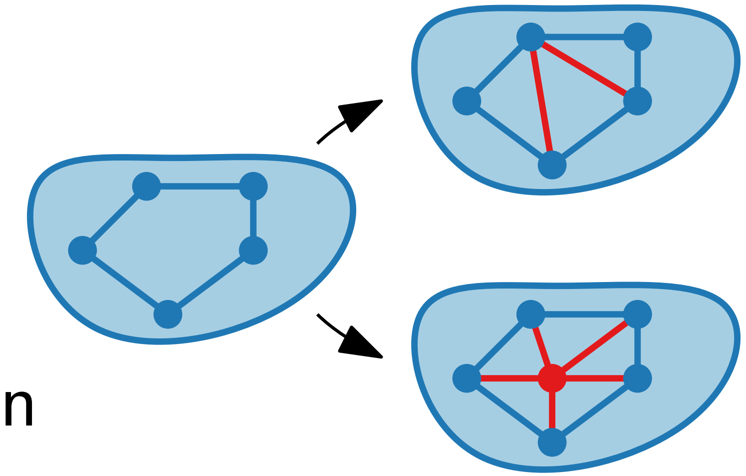


Planar graphs

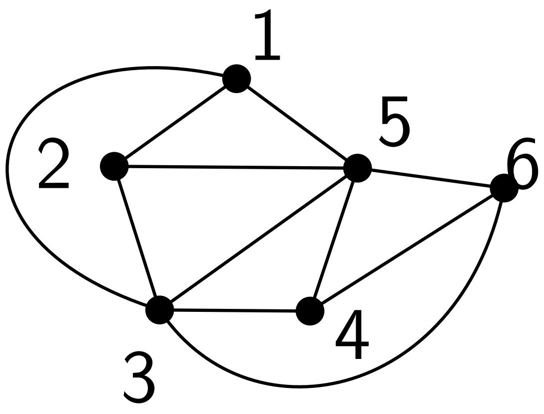
- Every planar graph has at most $3n - 6$ edges
- A *planar triangulation* is a planar graph with $3n - 6$ edges

- **Properties of planar triangulations:**
  - Every face is a triangle
  - graph is 3-connected
  - Unique embedding (up to choice of outerface)
  - Every plane graph is subgraph of a plane triangulation

with planar embedding



- We focus on **triangulations**:
  - A *plane (inner) triangulation* is a plane graph where every (inner) face is a triangle.



Planar straight-line drawings

Goal:
For an n-vertex planar graph create a planar straight-line drawing of size $\mathcal{O}(n^2)$.

Planar straight-line drawings

Goal:
For an n-vertex planar graph create a planar straight-line drawing of size $\mathcal{O}(n^2)$.

Idea.
Create drawing incrementally by adding vertices

Planar straight-line drawings

Goal:
For an n-vertex planar graph create a planar straight-line drawing of size $\mathcal{O}(n^2)$.

Idea.
Create drawing incrementally by adding vertices

Idea (refined).

- Start with singe edge (v_1, v_2). Let this be G_2.

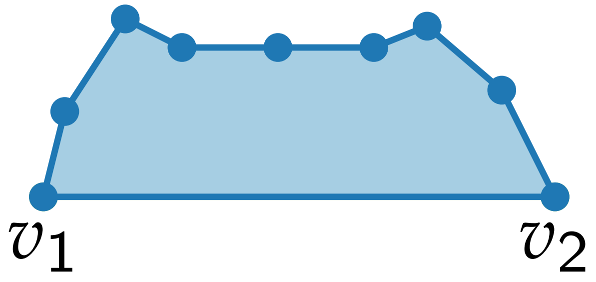
Planar straight-line drawings

Goal:
For an n-vertex planar graph create a planar straight-line drawing of size $\mathcal{O}(n^2)$.

Idea.
Create drawing incrementally by adding vertices

Idea (refined).

- Start with singe edge (v_1, v_2). Let this be G_2.



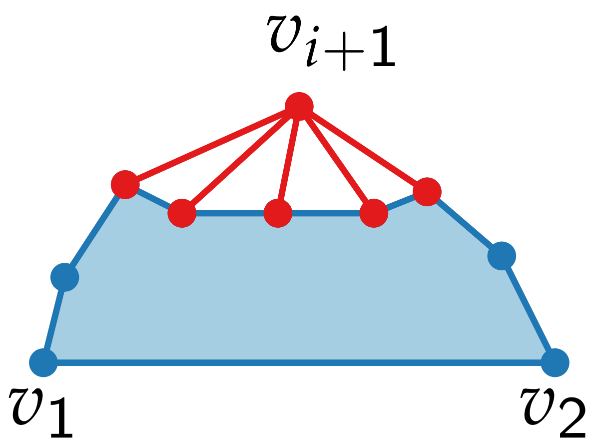
Planar straight-line drawings

Goal:
For an n-vertex planar graph create a planar straight-line drawing of size $\mathcal{O}(n^2)$.

Idea.
Create drawing incrementally by adding vertices

Idea (refined).

- Start with singe edge (v_1, v_2). Let this be G_2.
- To obtain G_{i+1}, add v_{i+1} to G_i so that neighbours of v_{i+1} are on the outer face of G_i.
- Neighbours of v_{i+1} in G_i have to form path of length at least two.



Canonical order – definition

Definition.
Let $G = (V, E)$ be a triangulated plane graph on $n \geq 3$ vertices. An order $\pi = (v_1, v_2, \ldots, v_n)$ is called a **canonical order**, if the following conditions hold for each k, $3 \leq k \leq n$:

Canonical order – definition

Definition.
Let $G = (V, E)$ be a triangulated plane graph on $n \geq 3$ vertices.
An order $\pi = (v_1, v_2, \ldots, v_n)$ is called a **canonical order**, if the following conditions hold for each k, $3 \leq k \leq n$:

- (C1) Vertices $\{v_1, \ldots v_k\}$ induce a biconnected internally triangulated graph; call it G_k.

Canonical order – definition

Definition.
Let $G = (V, E)$ be a triangulated plane graph on $n \geq 3$ vertices.
An order $\pi = (v_1, v_2, \ldots, v_n)$ is called a **canonical order**, if the following conditions hold for each k, $3 \leq k \leq n$:

- (C1) Vertices $\{v_1, \ldots v_k\}$ induce a biconnected internally triangulated graph; call it G_k.
- (C2) Edge (v_1, v_2) belongs to the outer face of G_k.

Canonical order – definition

Definition.
Let $G = (V, E)$ be a triangulated plane graph on $n \geq 3$ vertices.
An order $\pi = (v_1, v_2, \ldots, v_n)$ is called a **canonical order**, if the following conditions hold for each k, $3 \leq k \leq n$:

- (C1) Vertices $\{v_1, \ldots v_k\}$ induce a biconnected internally triangulated graph; call it G_k.
- (C2) Edge (v_1, v_2) belongs to the outer face of G_k.
- (C3) If $k < n$ then vertex v_{k+1} lies in the outer face of G_k, and all neighbors of v_{k+1} in G_k appear on the boundary of G_k consecutively.

Canonical order – definition

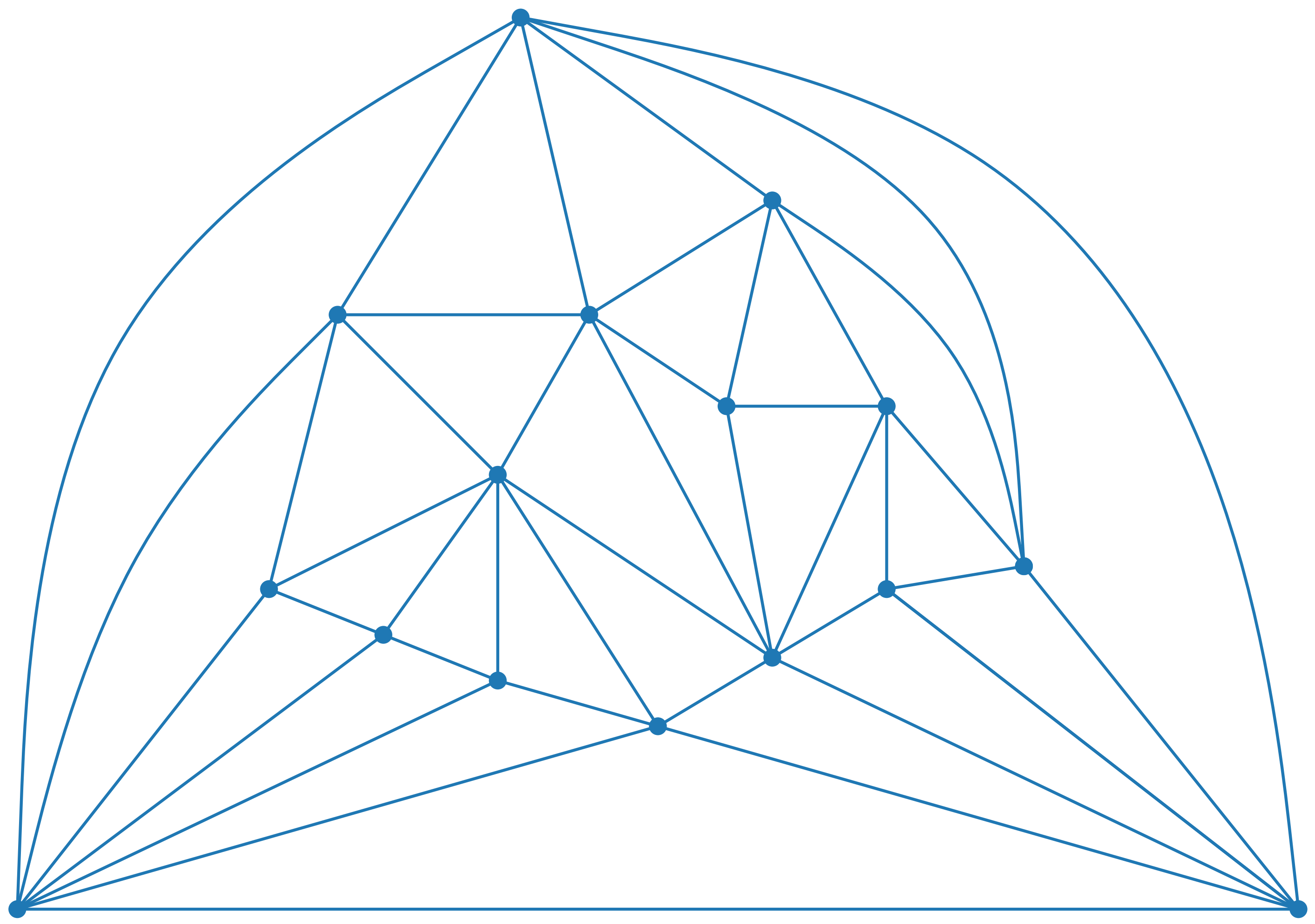
Definition.
Let $G = (V, E)$ be a triangulated plane graph on $n \geq 3$ vertices. An order $\pi = (v_1, v_2, \ldots, v_n)$ is called a **canonical order**, if the following conditions hold for each k, $3 \leq k \leq n$:

- (C1) Vertices $\{v_1, \ldots v_k\}$ induce a biconnected internally triangulated graph; call it G_k.
- (C2) Edge (v_1, v_2) belongs to the outer face of G_k.
- (C3) If $k < n$ then vertex v_{k+1} lies in the outer face of G_k, and all neighbors of v_{k+1} in G_k appear on the boundary of G_k consecutively.

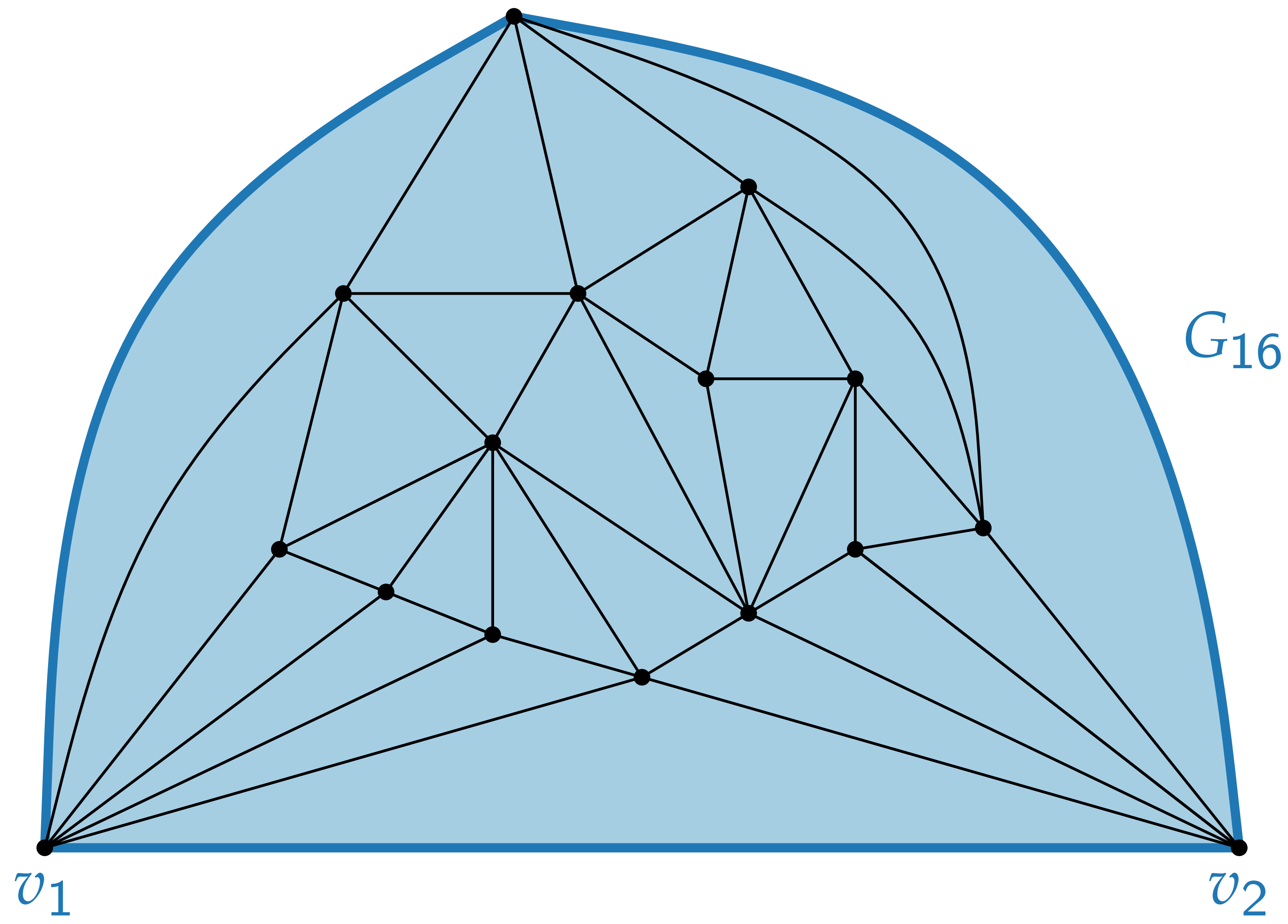
Compute:

- either $\{v_3, v_4, \ldots v_n\}$ (adding vertices)
- or $\{v_n, v_{n-1}, \ldots v_3\}$ (removing vertices)

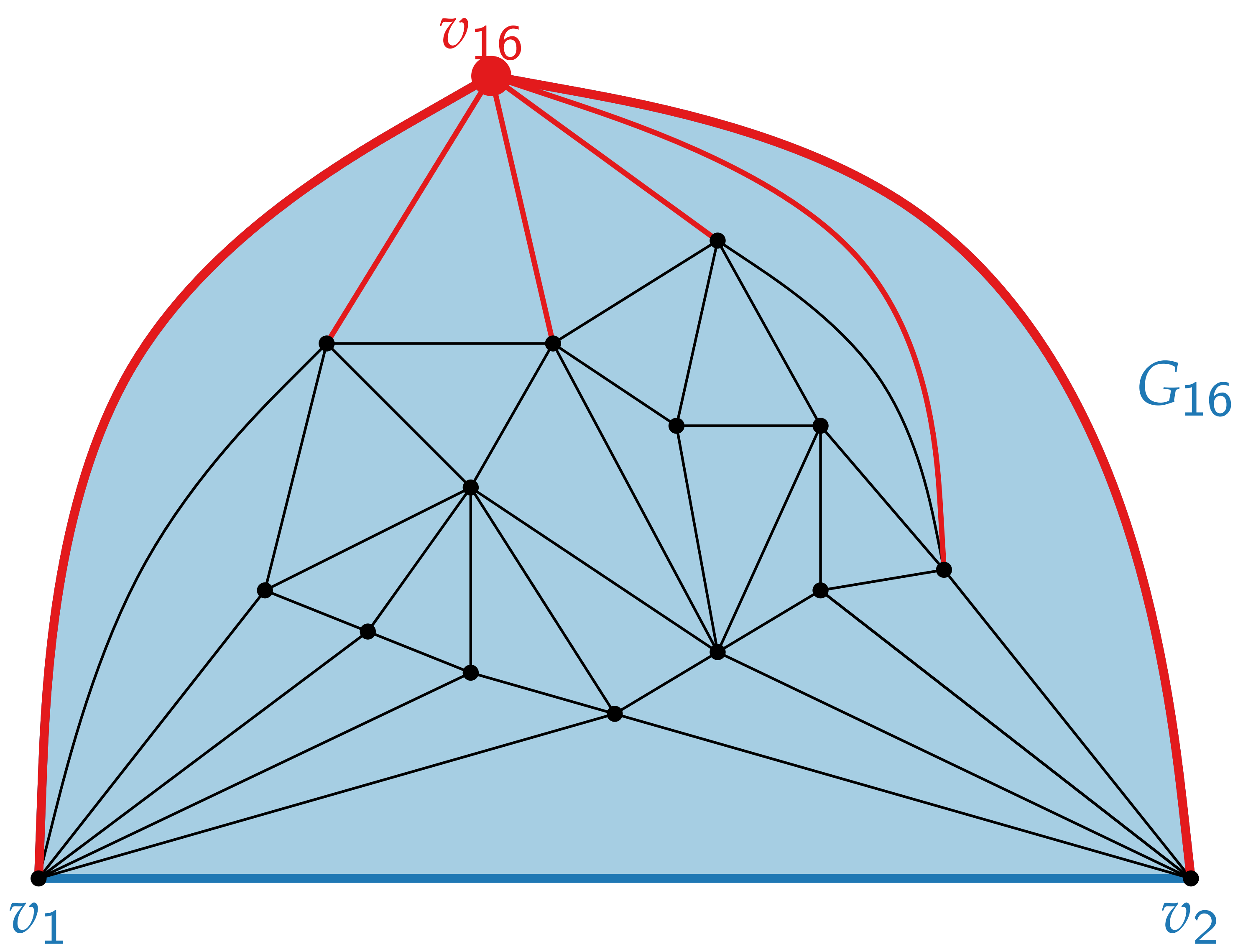
Canonical order – example



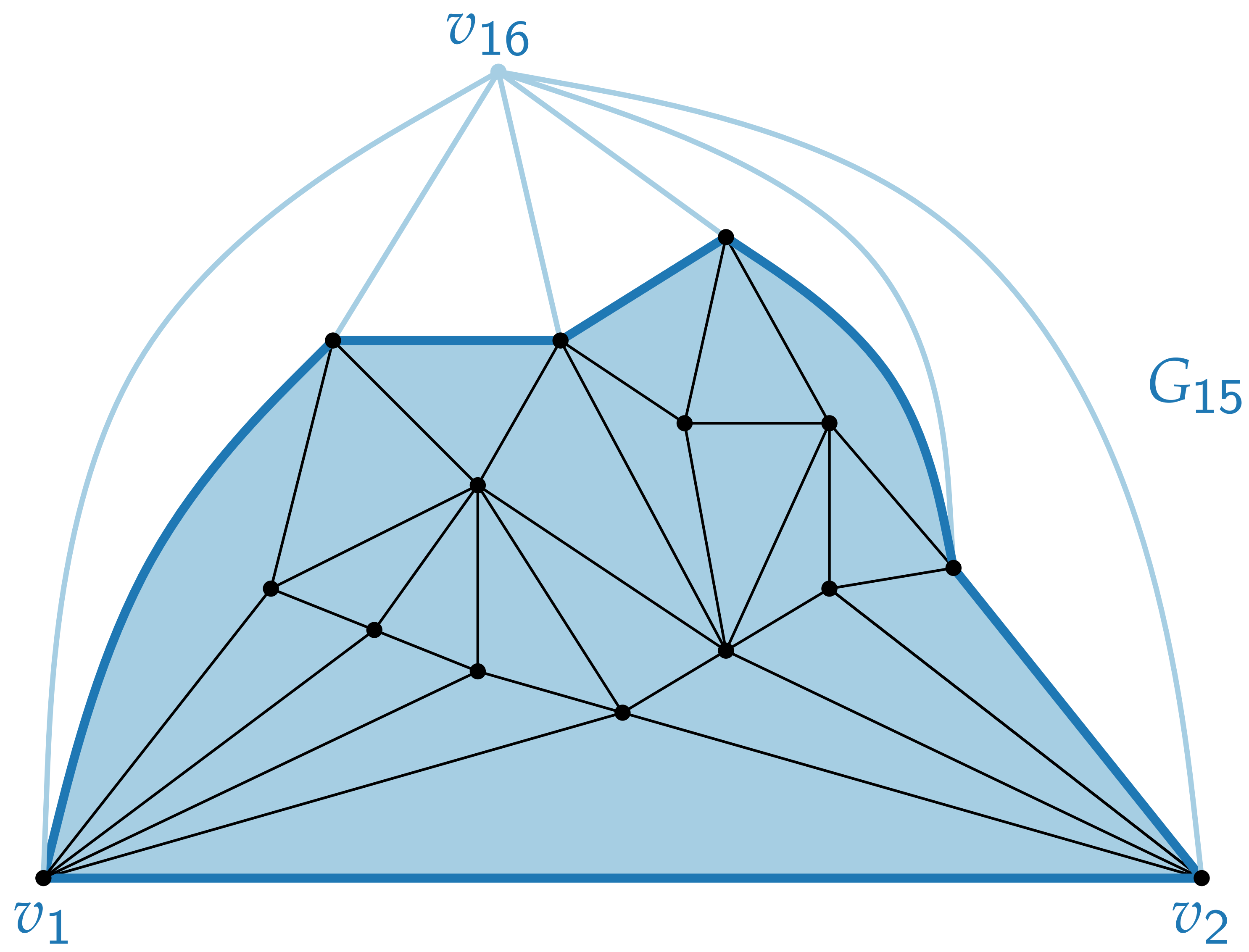
Canonical order – example



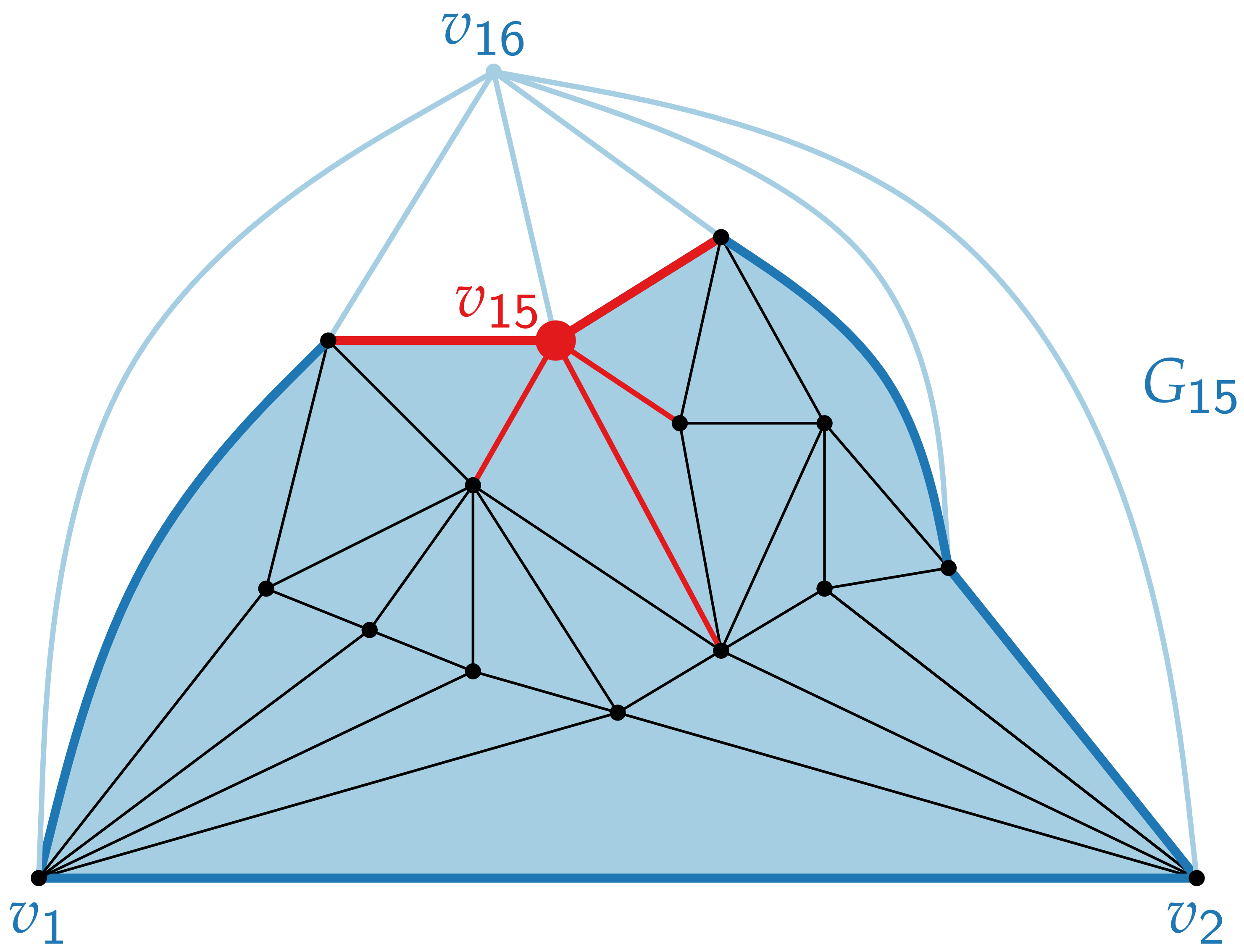
Canonical order – example



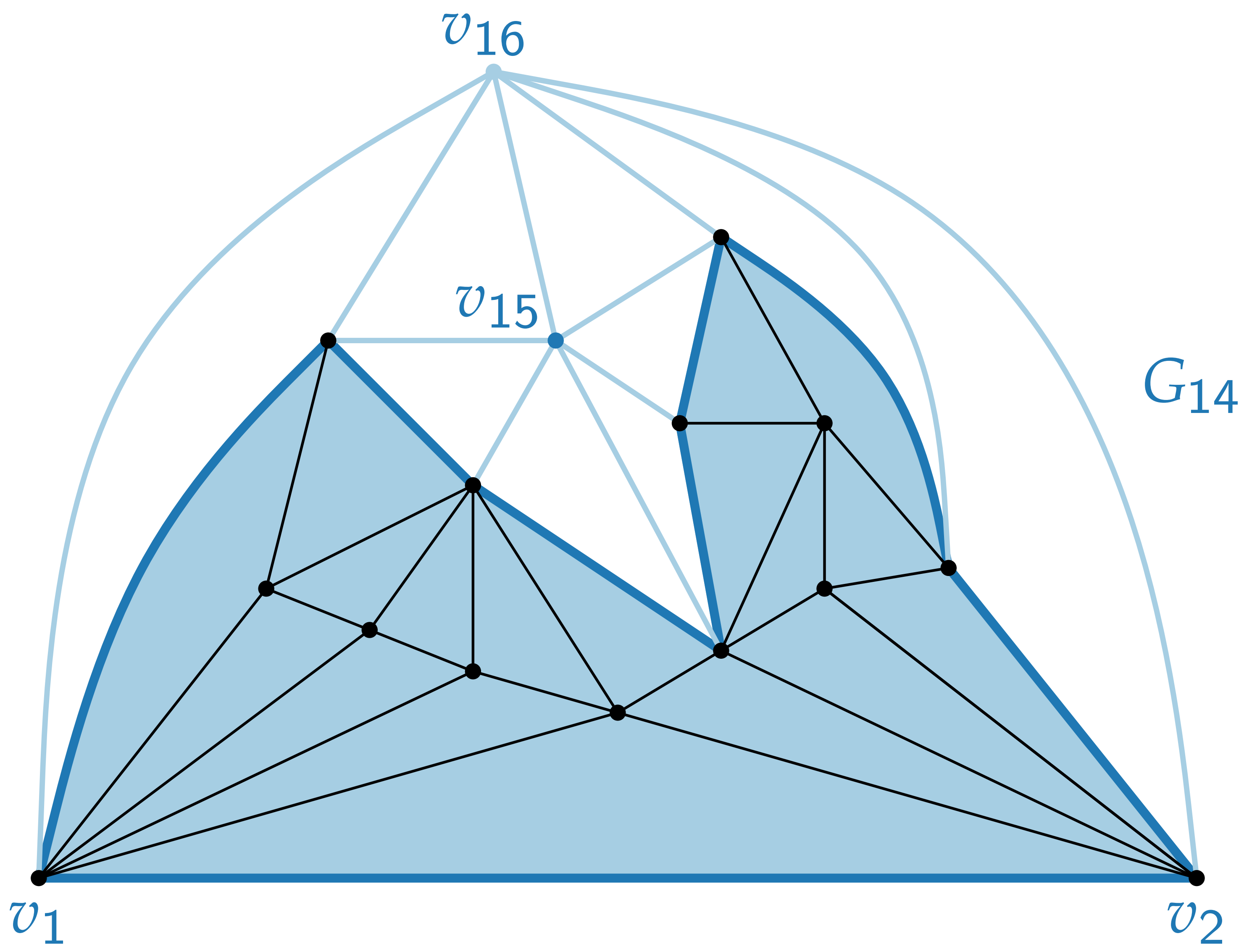
Canonical order – example



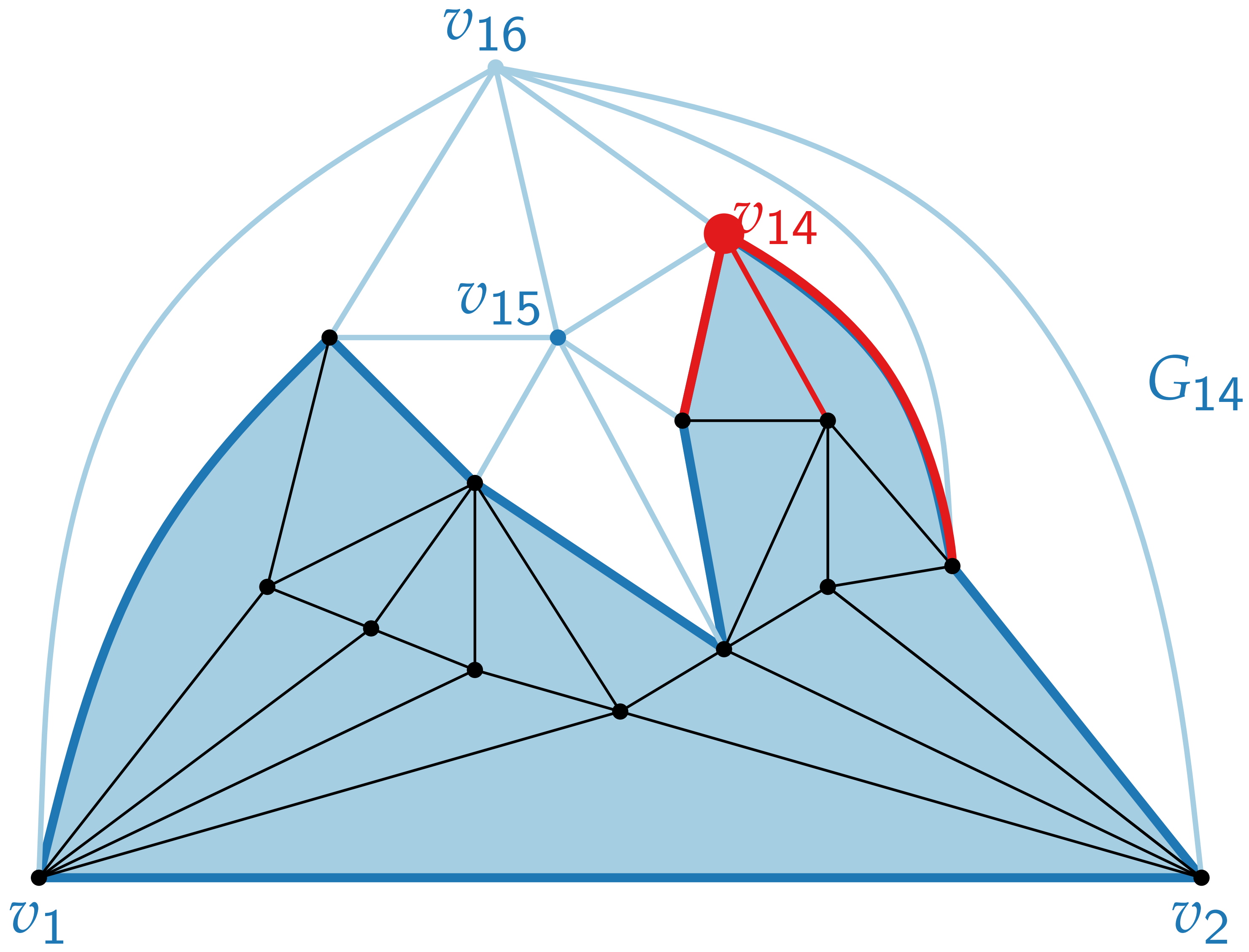
Canonical order – example



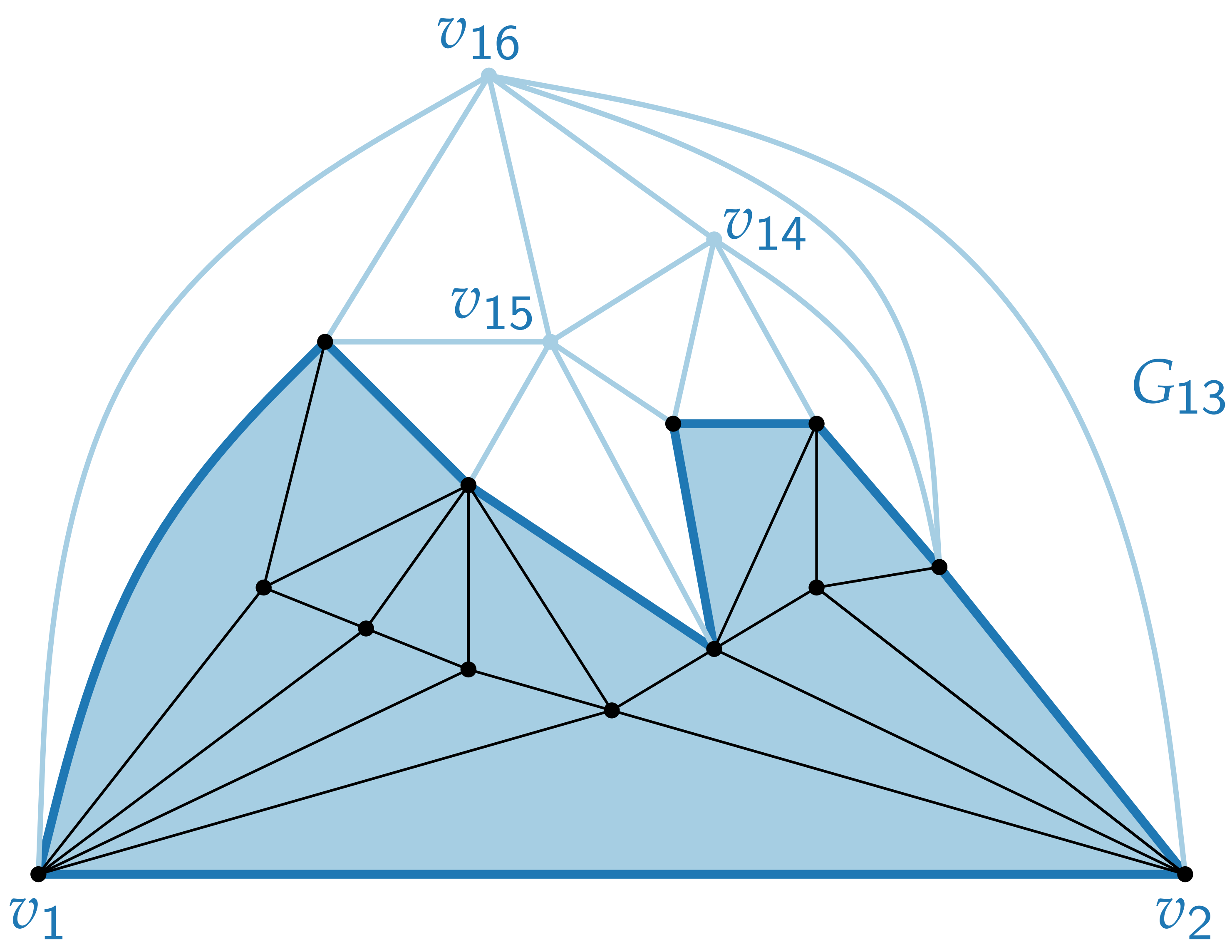
Canonical order – example



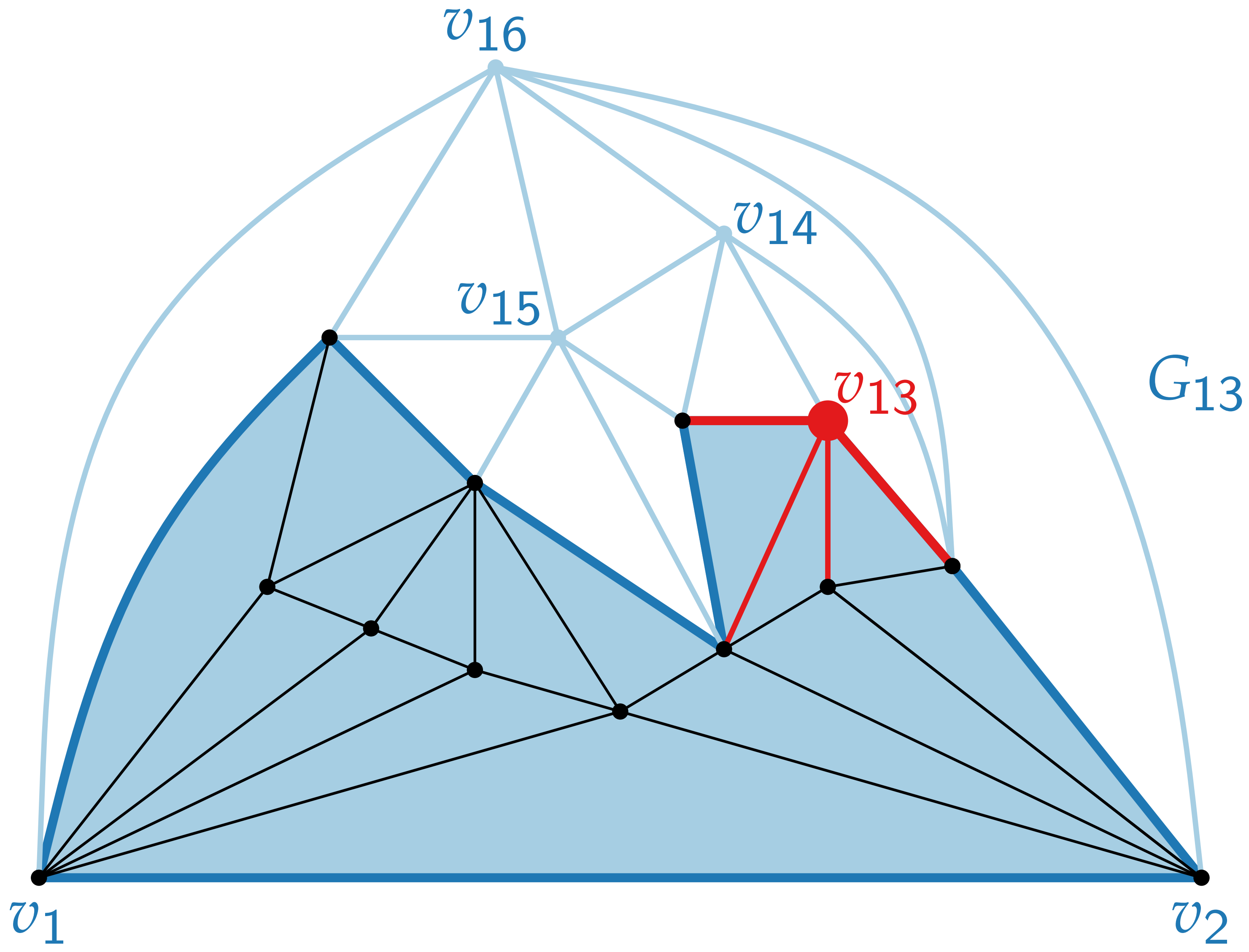
Canonical order – example



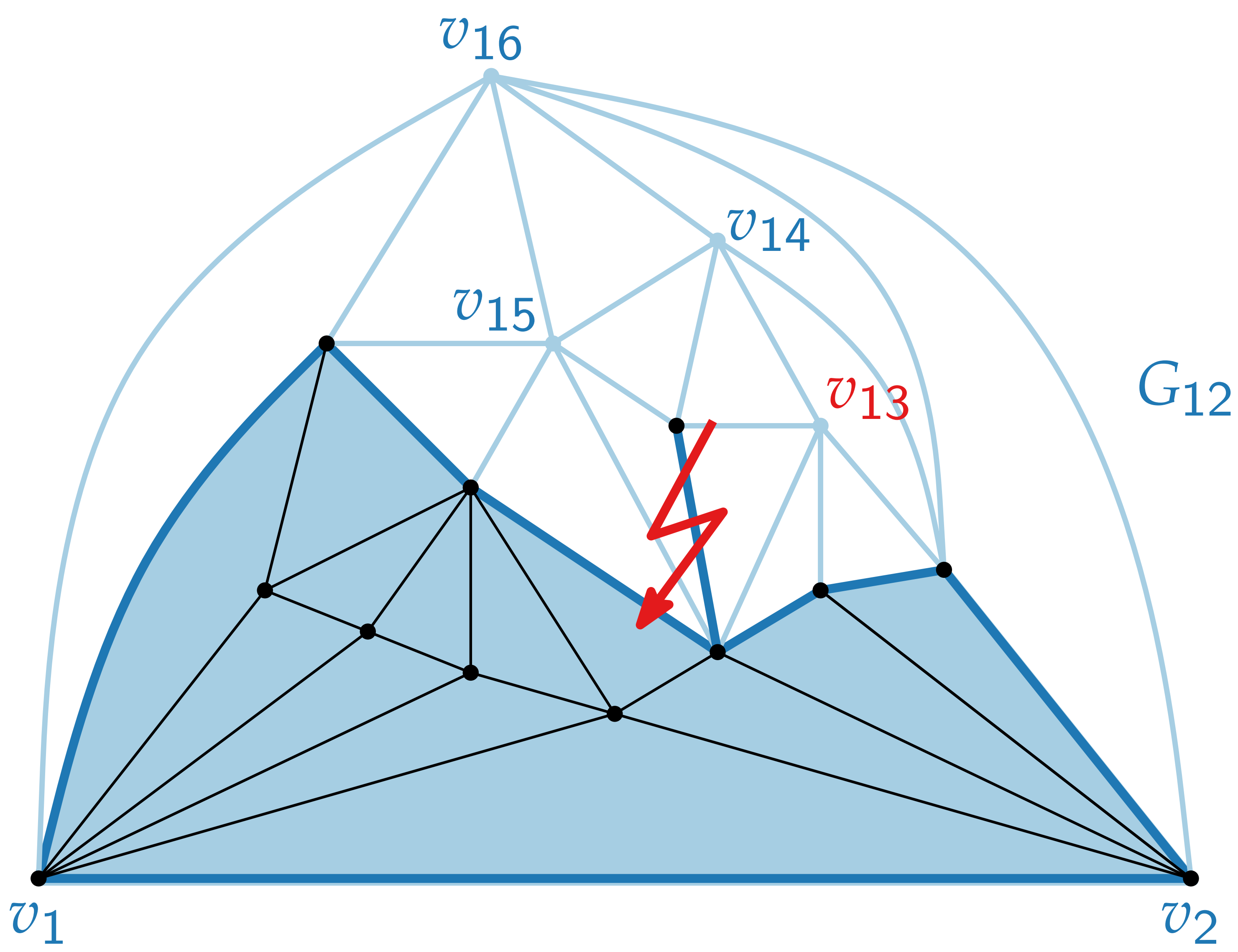
Canonical order – example



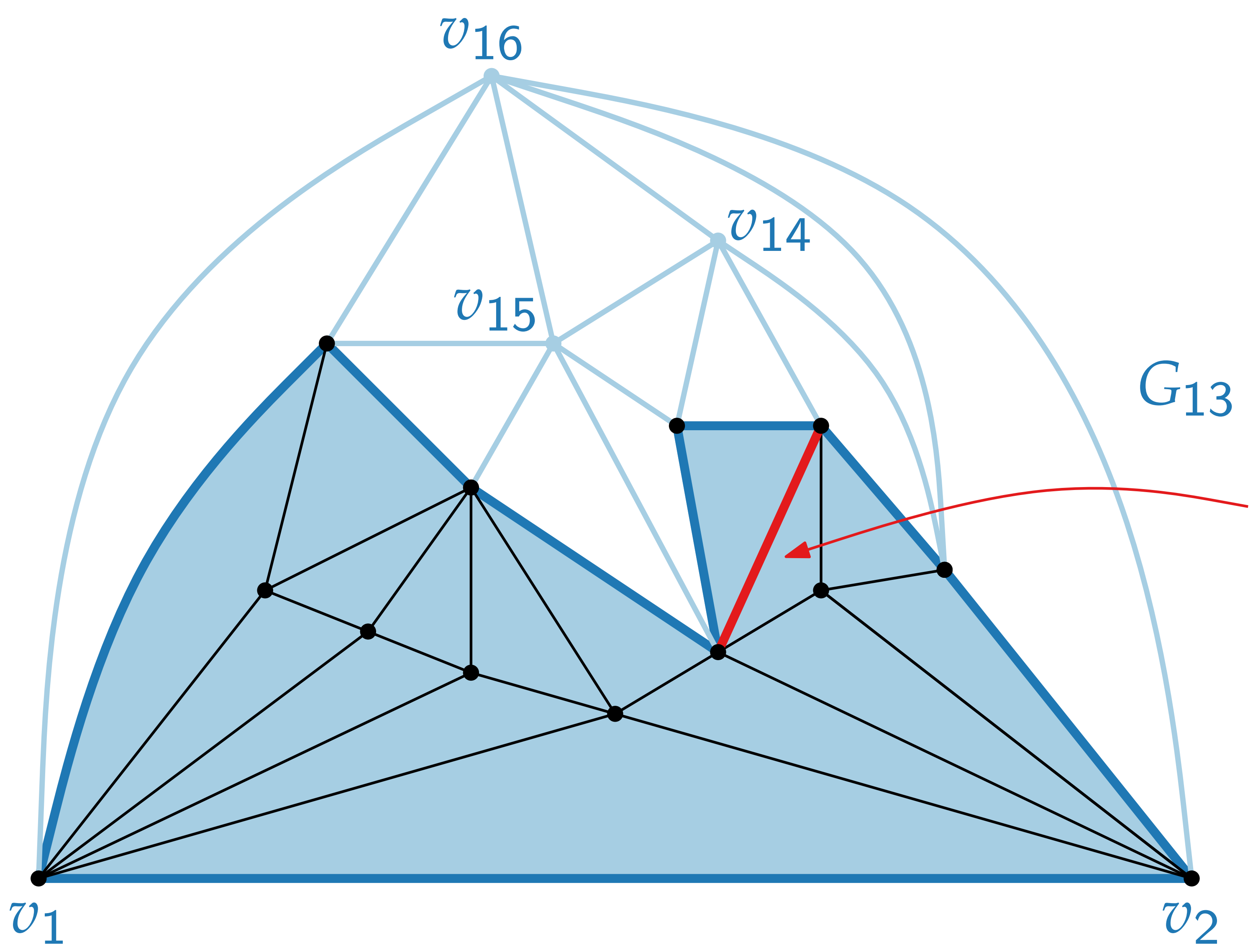
Canonical order – example



Canonical order – example

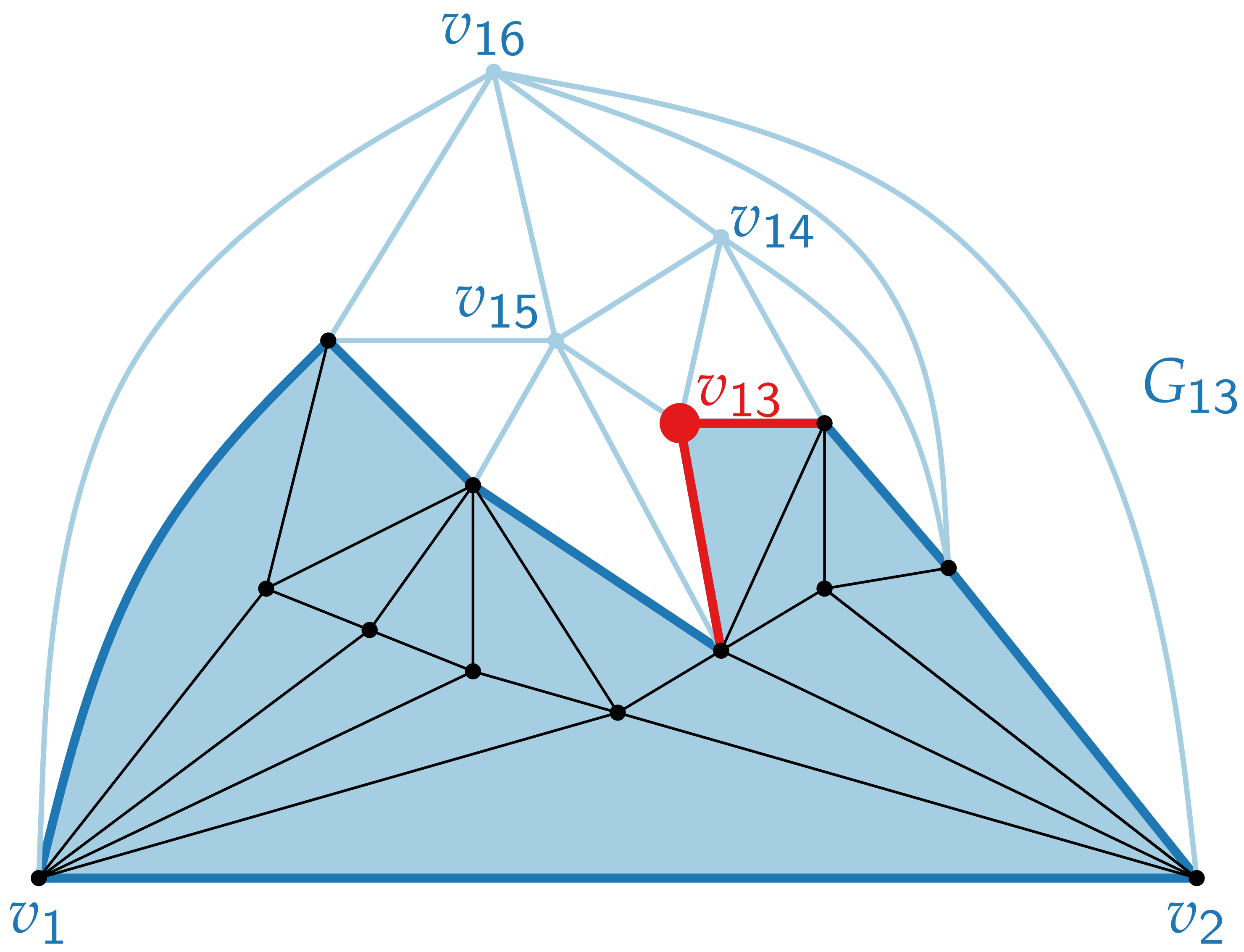


Canonical order – example

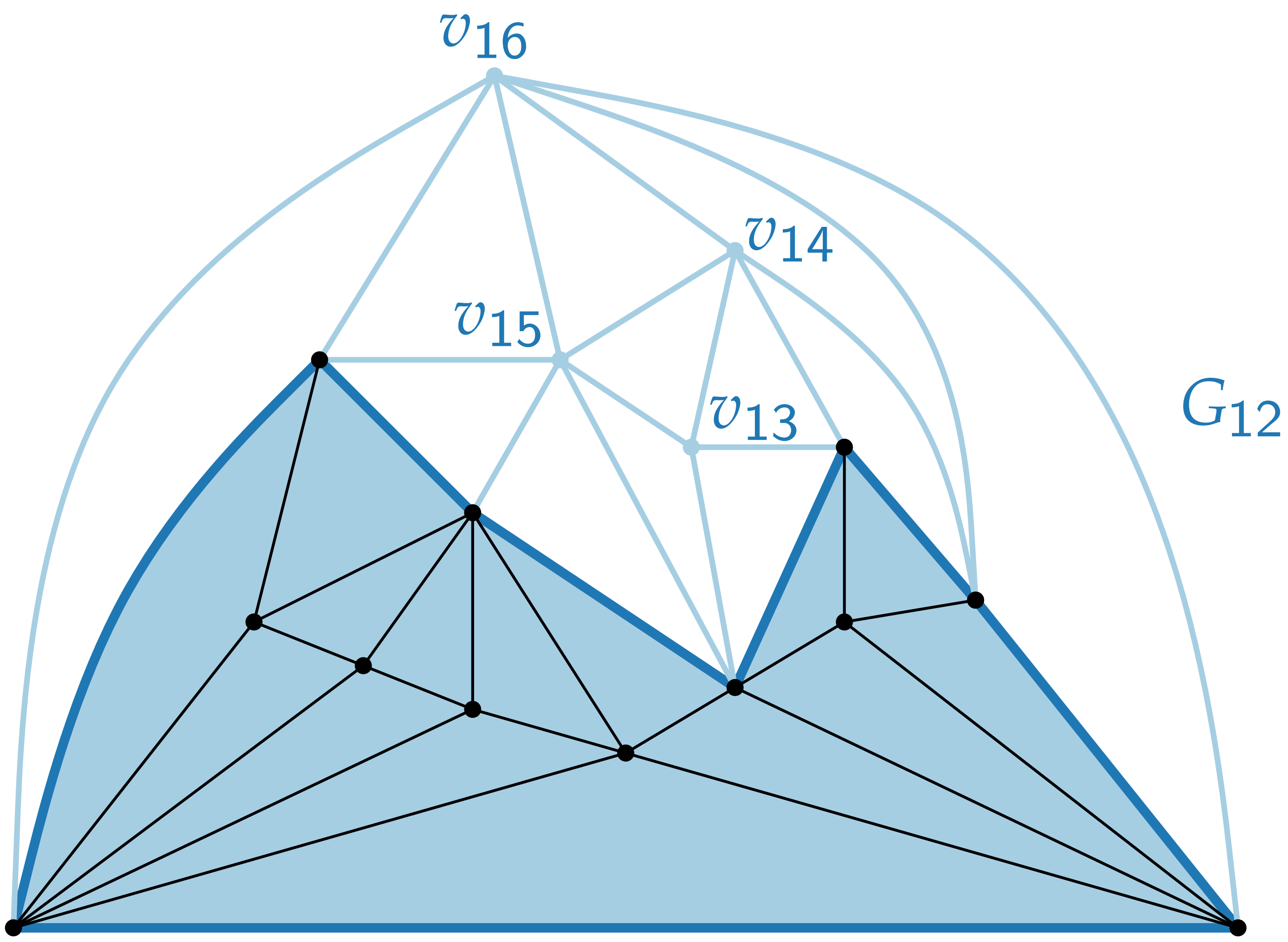


chord
edge joining two nonadjacent vertices in a cycle

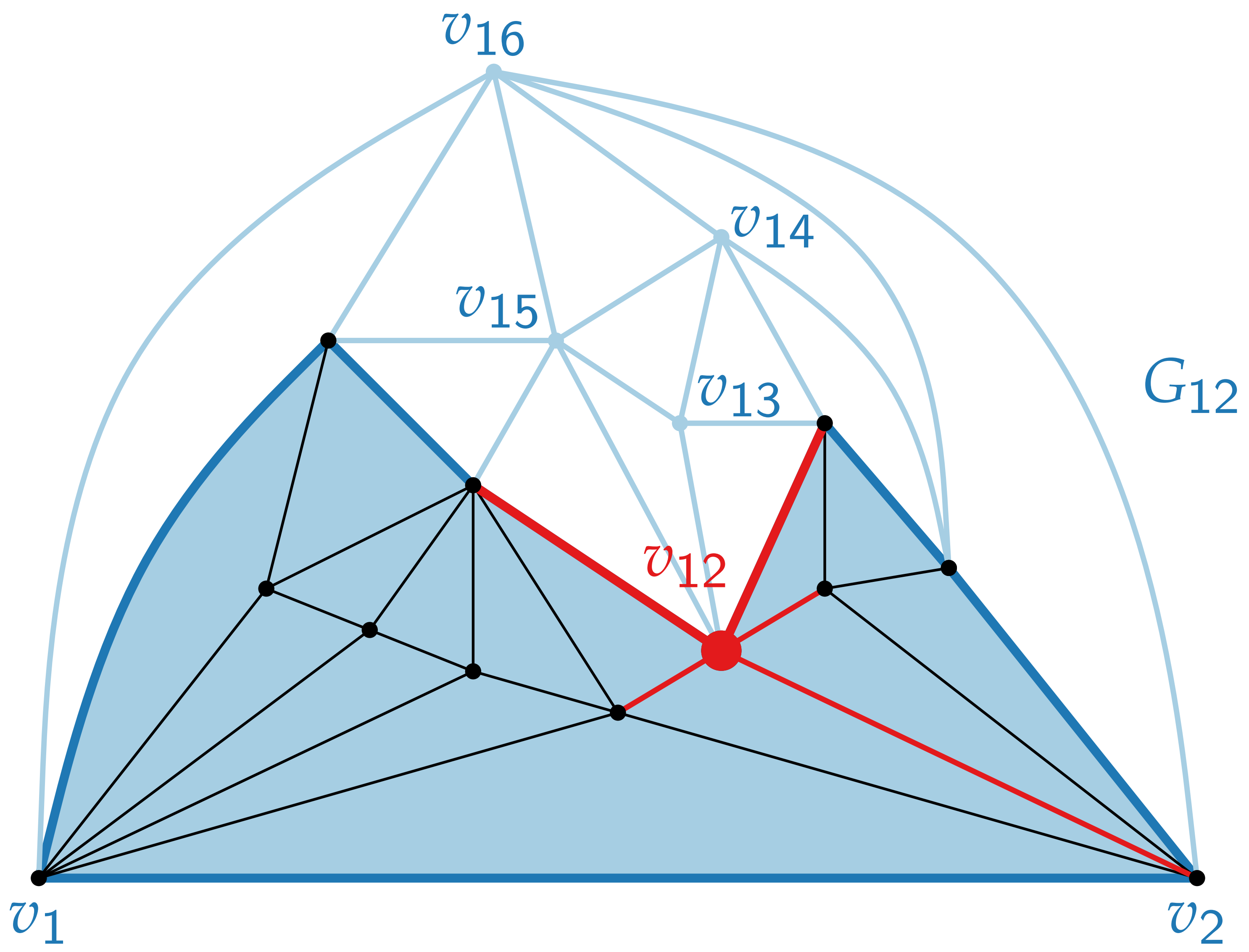
Canonical order – example



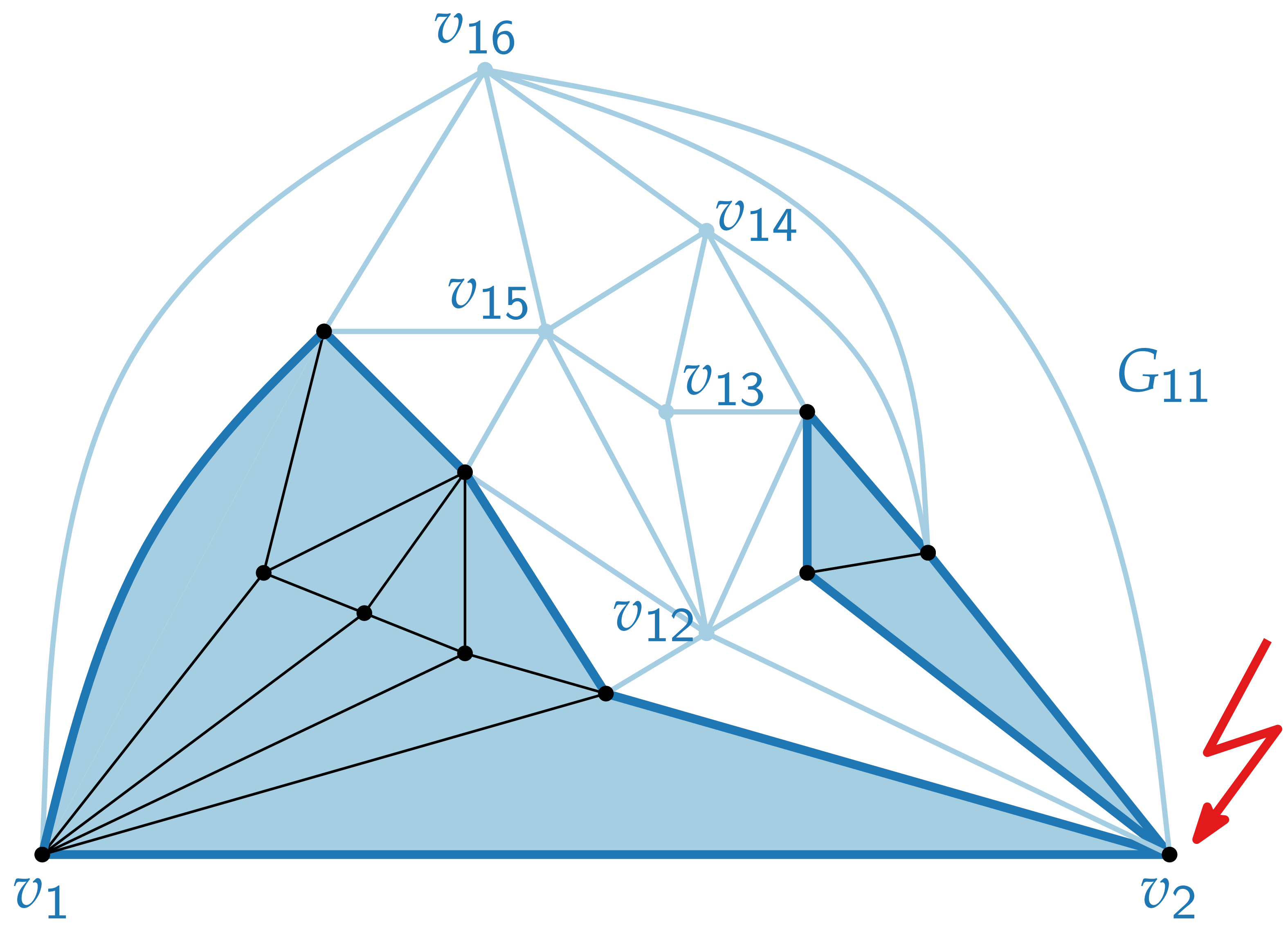
Canonical order – example



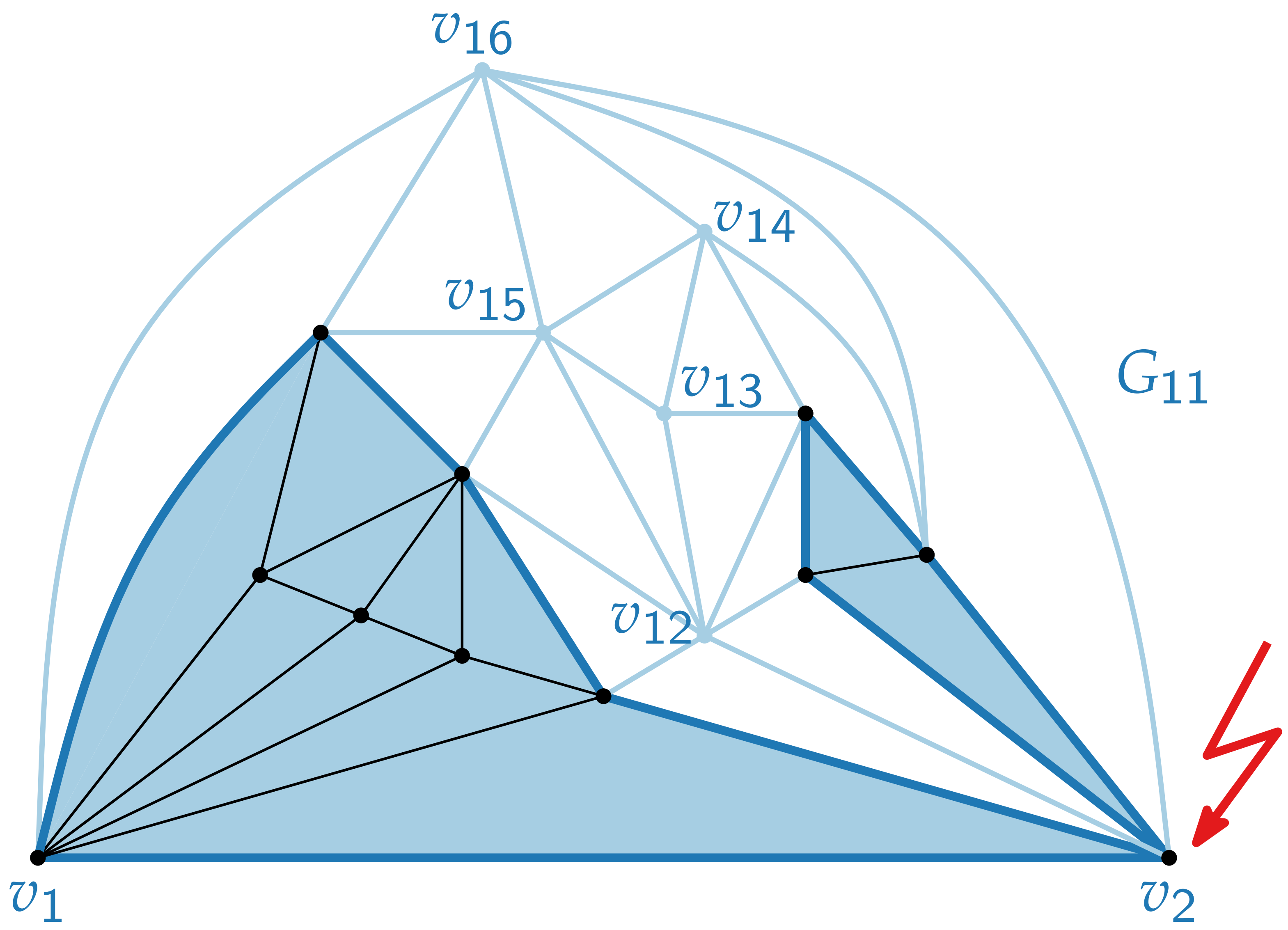
Canonical order – example



Canonical order – example



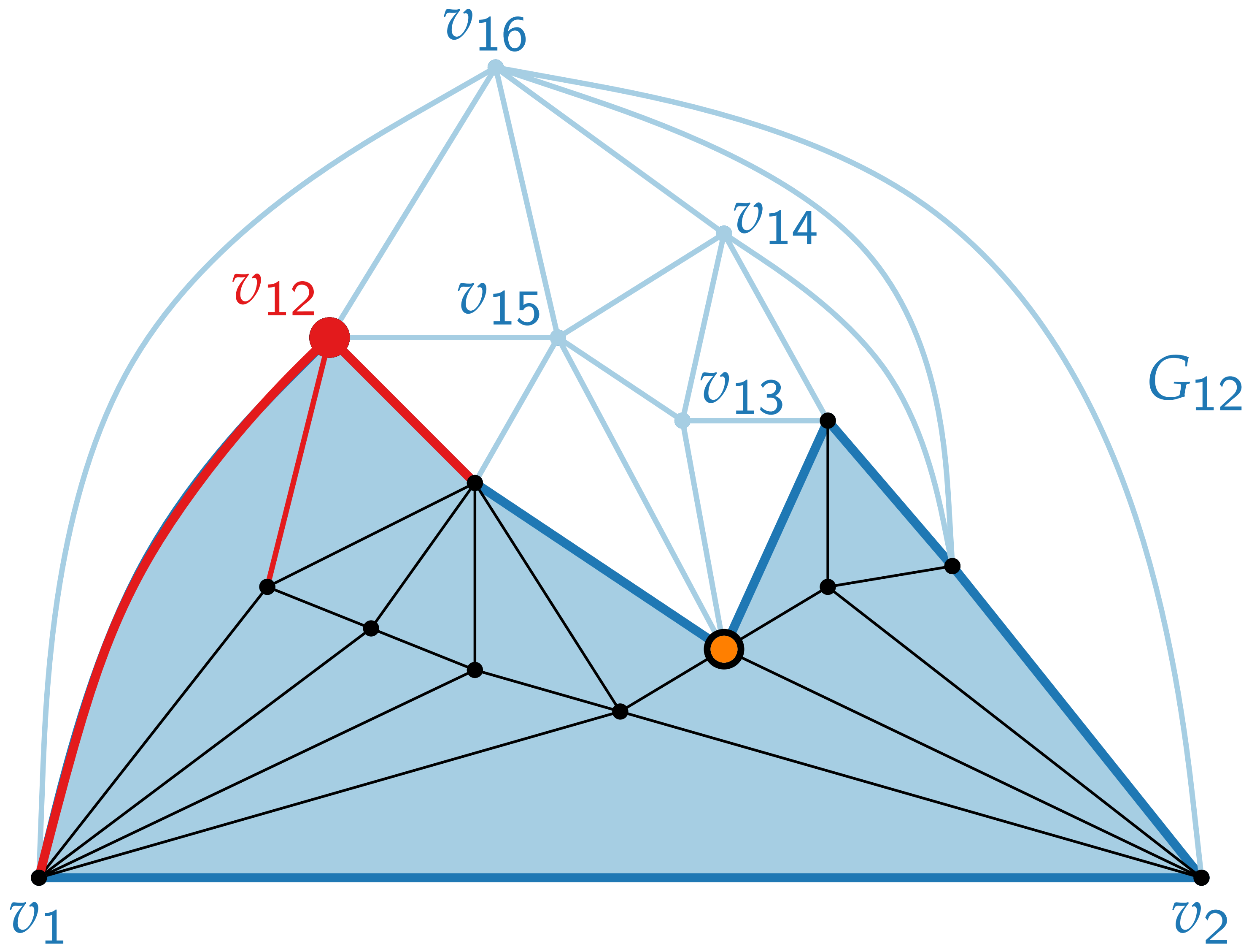
Canonical order – example



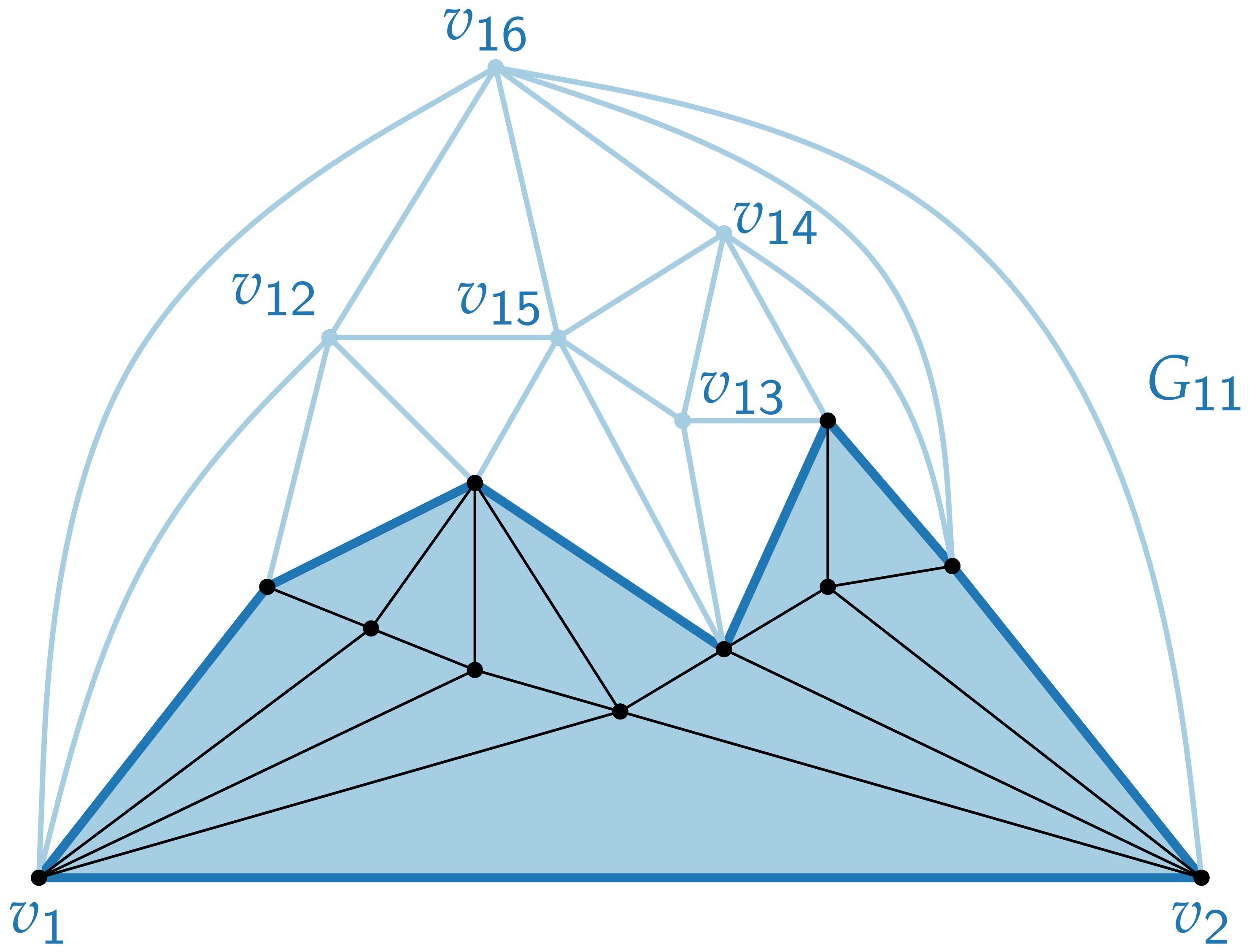
cutvertex

G_{11} is not biconnected

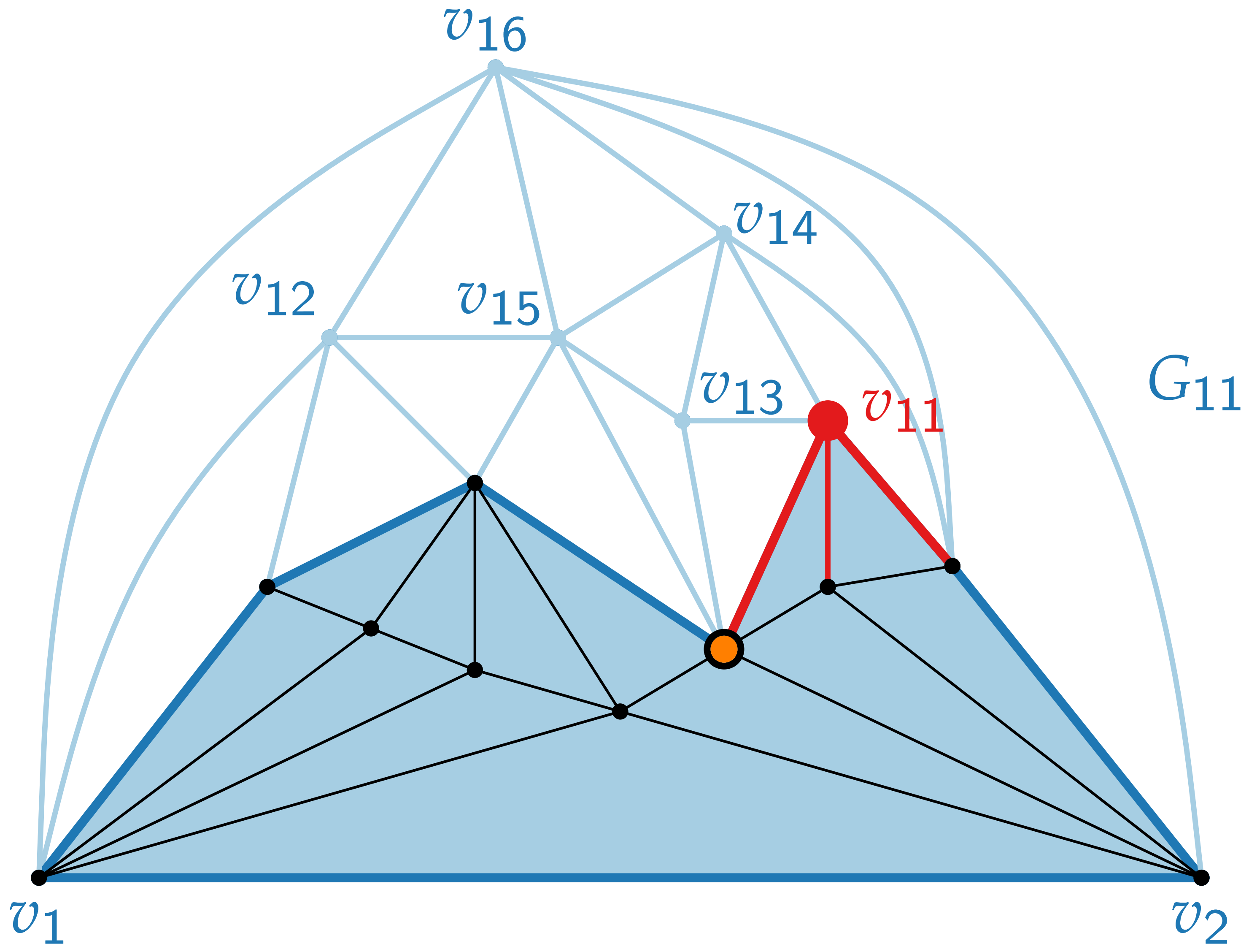
Canonical order – example



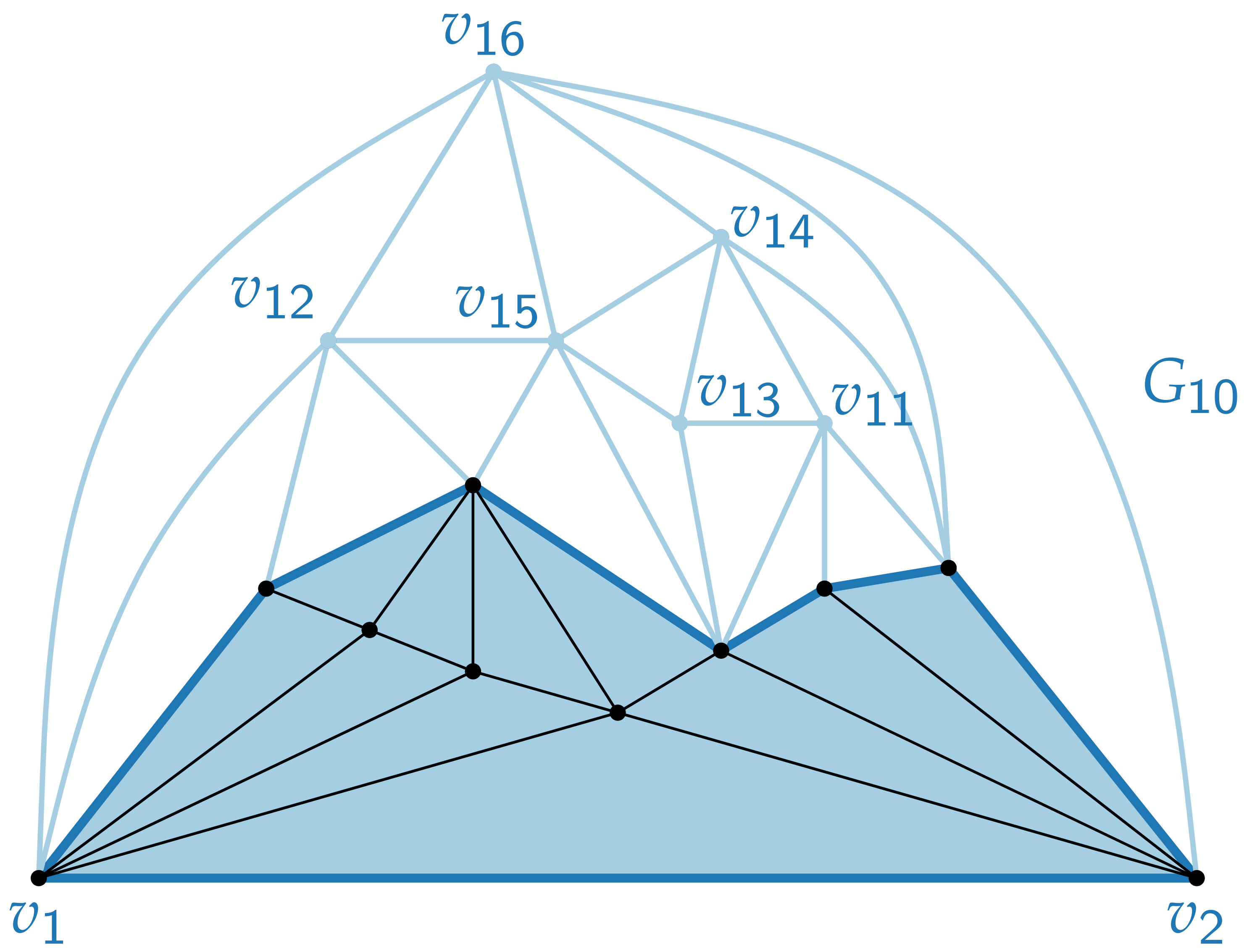
Canonical order – example



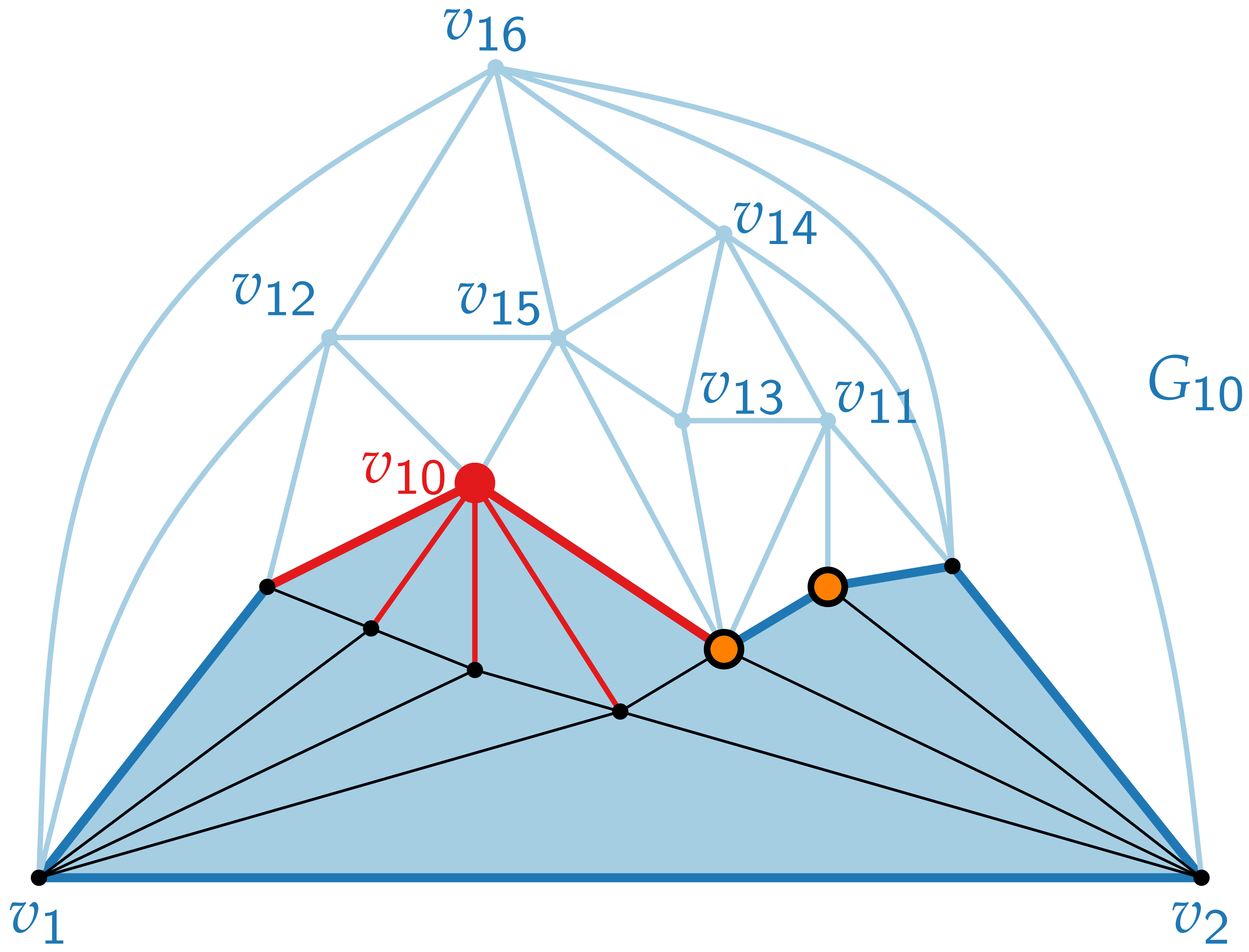
Canonical order – example



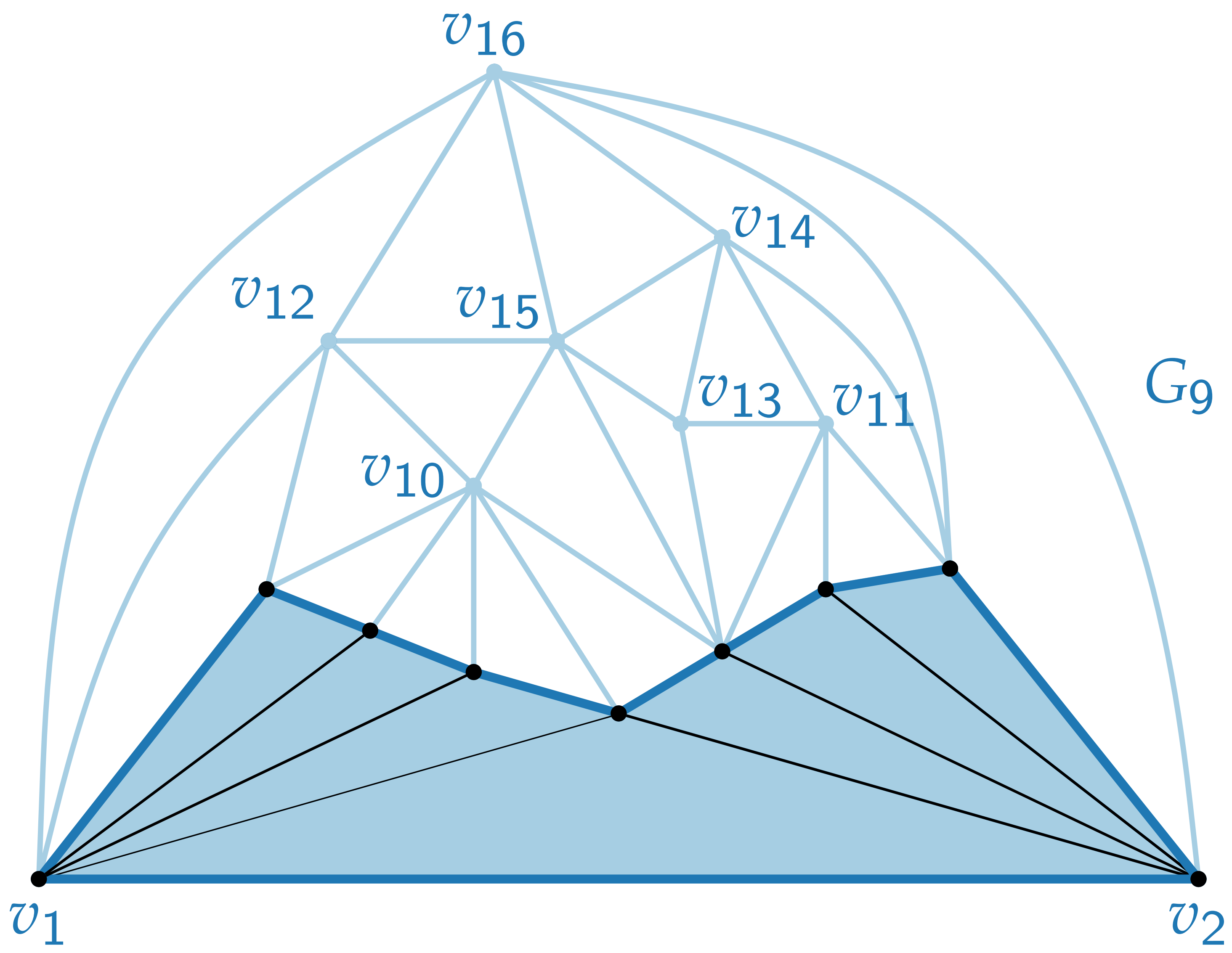
Canonical order – example



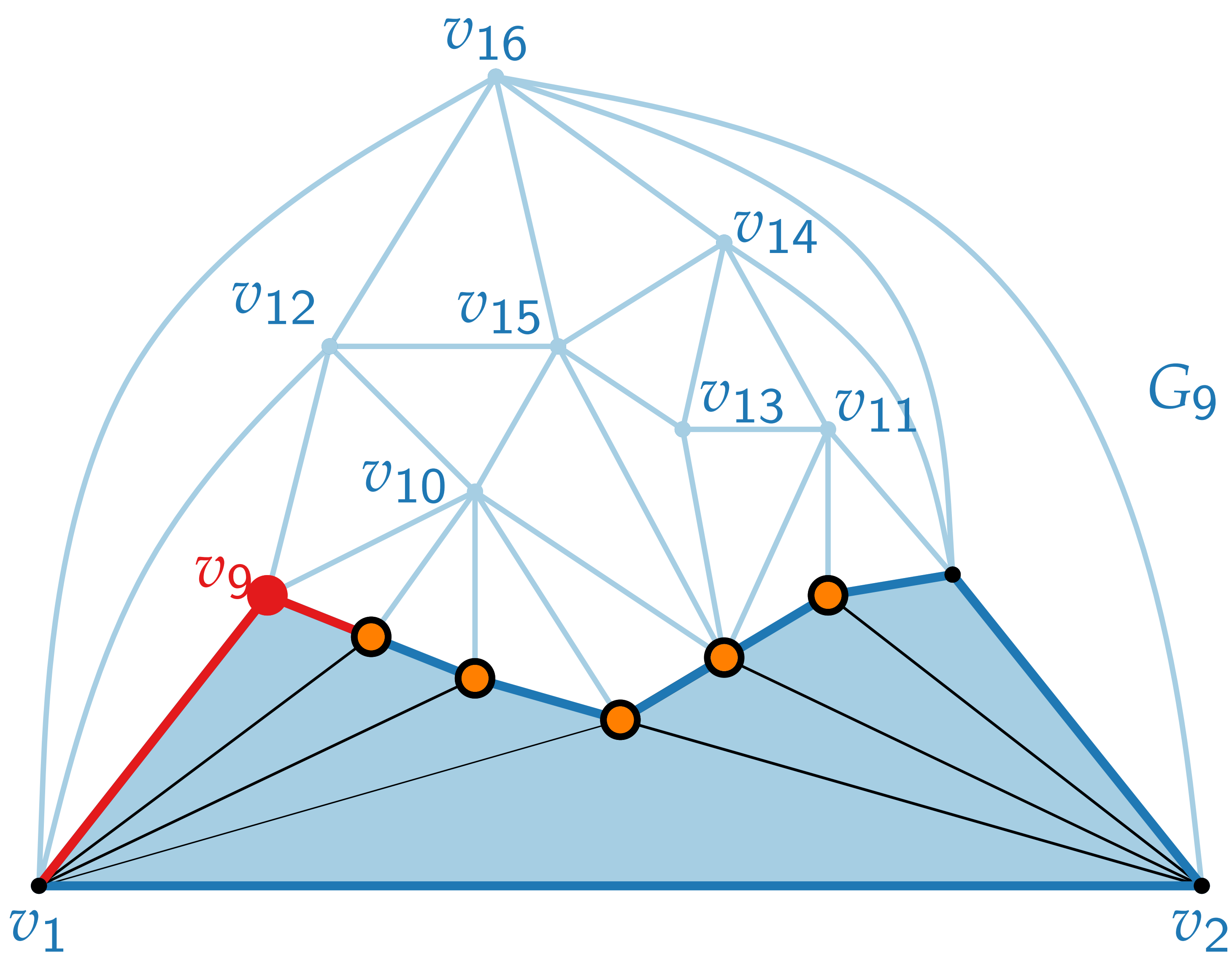
Canonical order – example



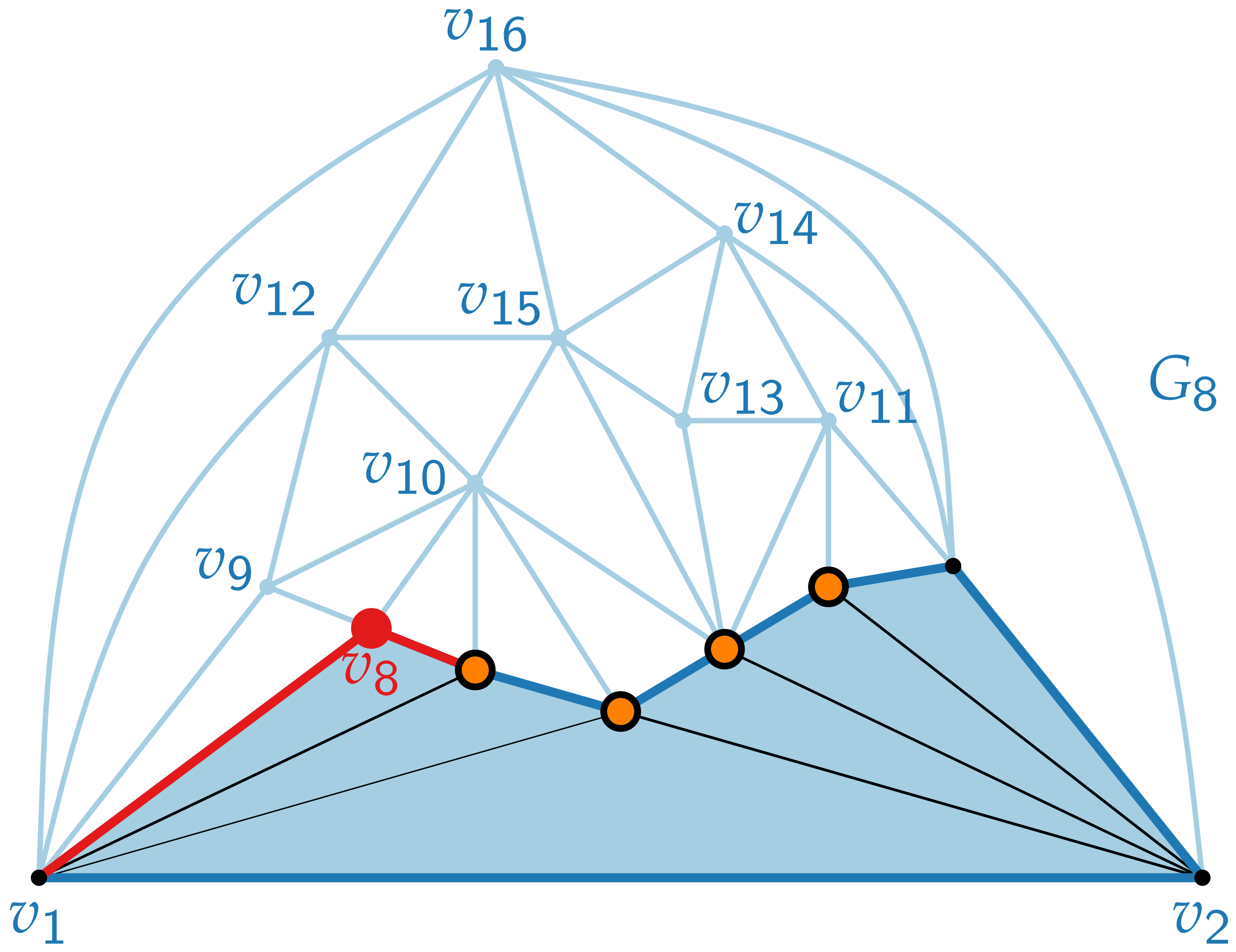
Canonical order – example



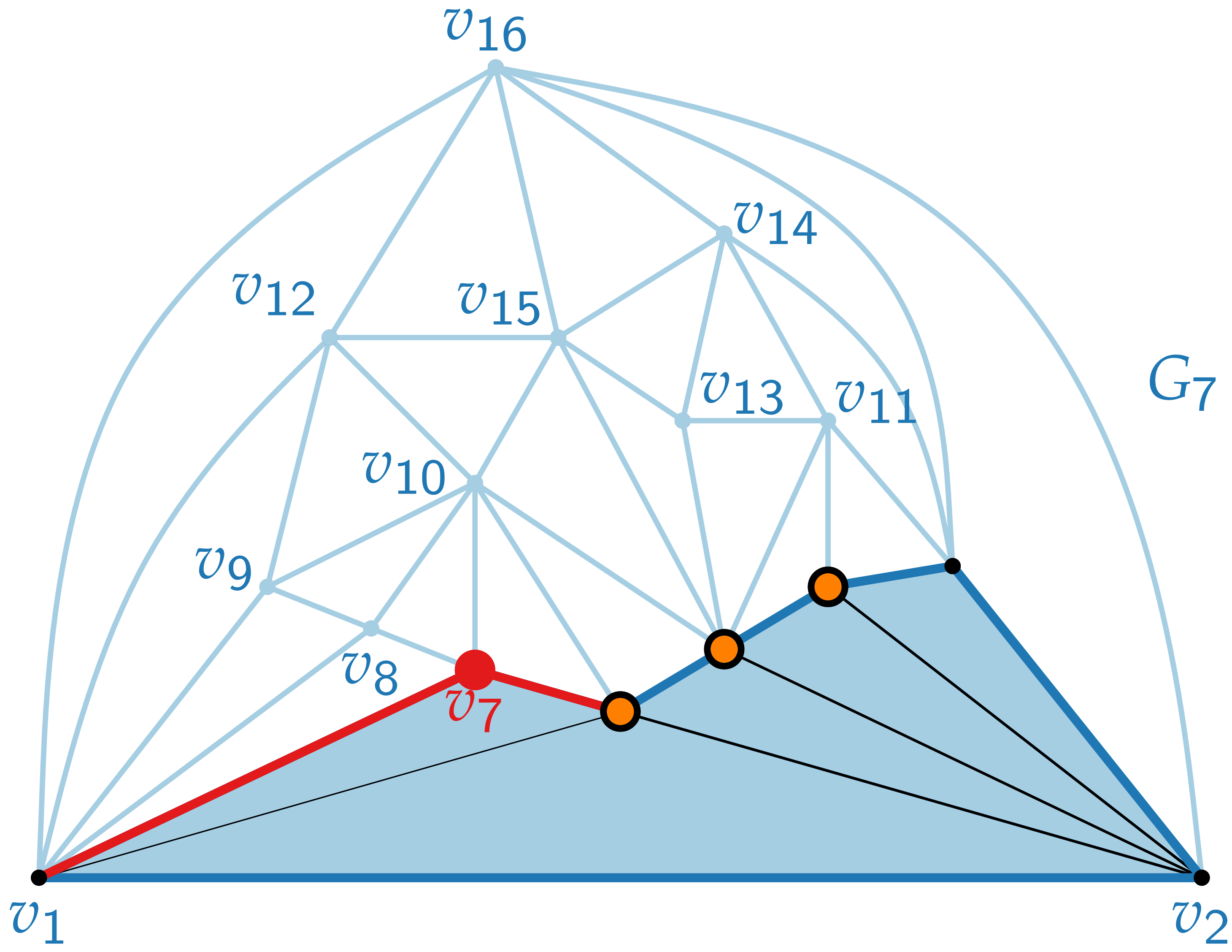
Canonical order – example



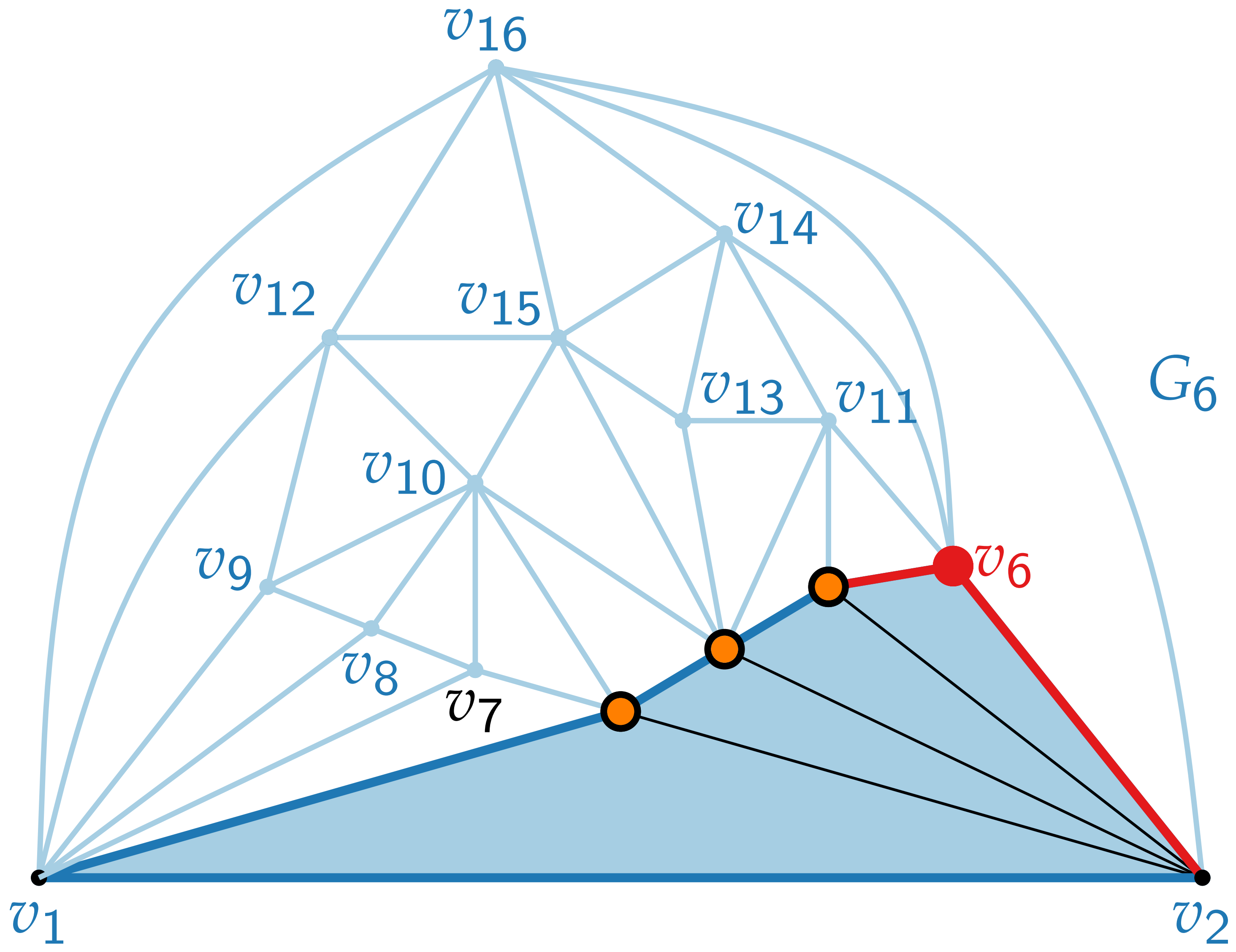
Canonical order – example



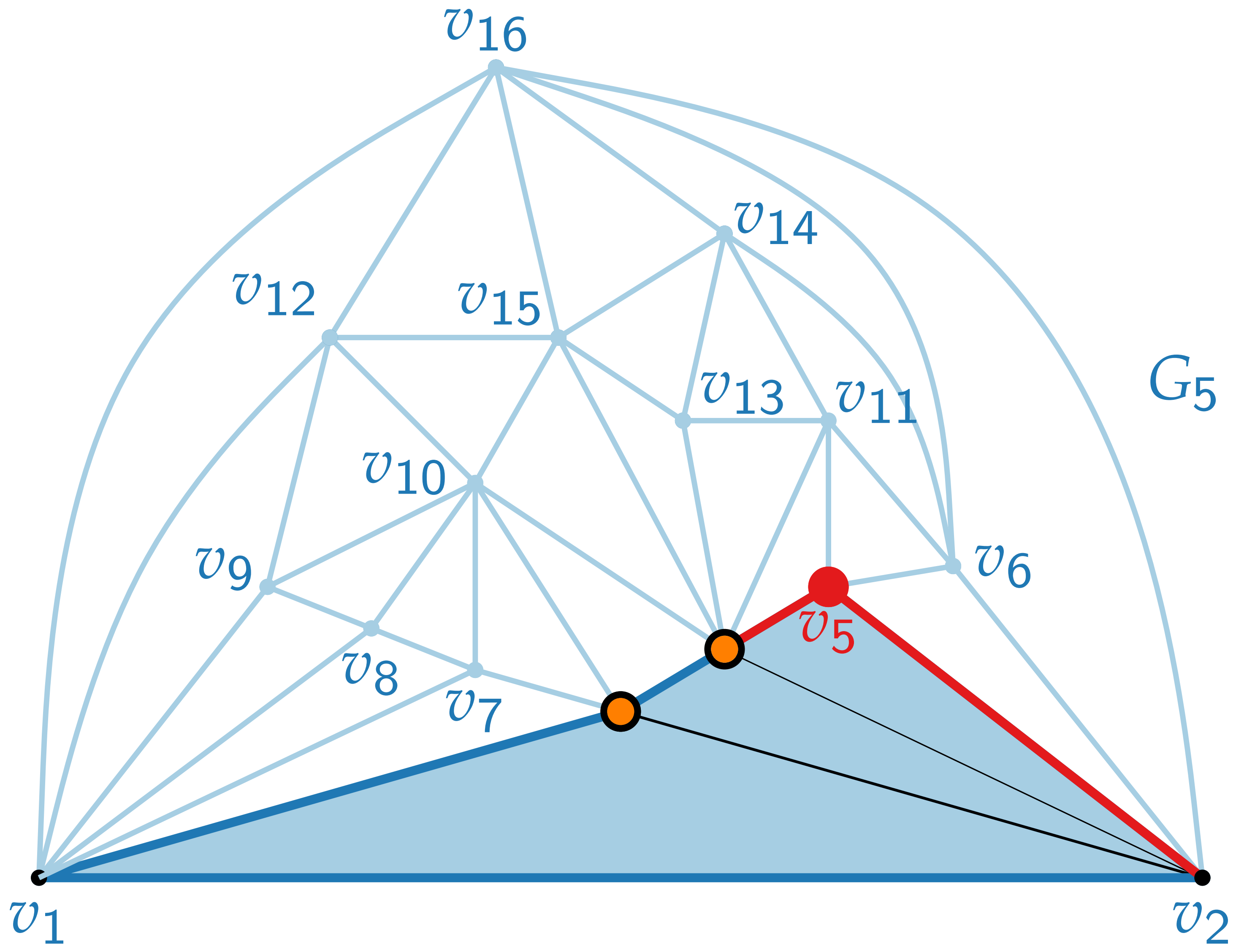
Canonical order – example



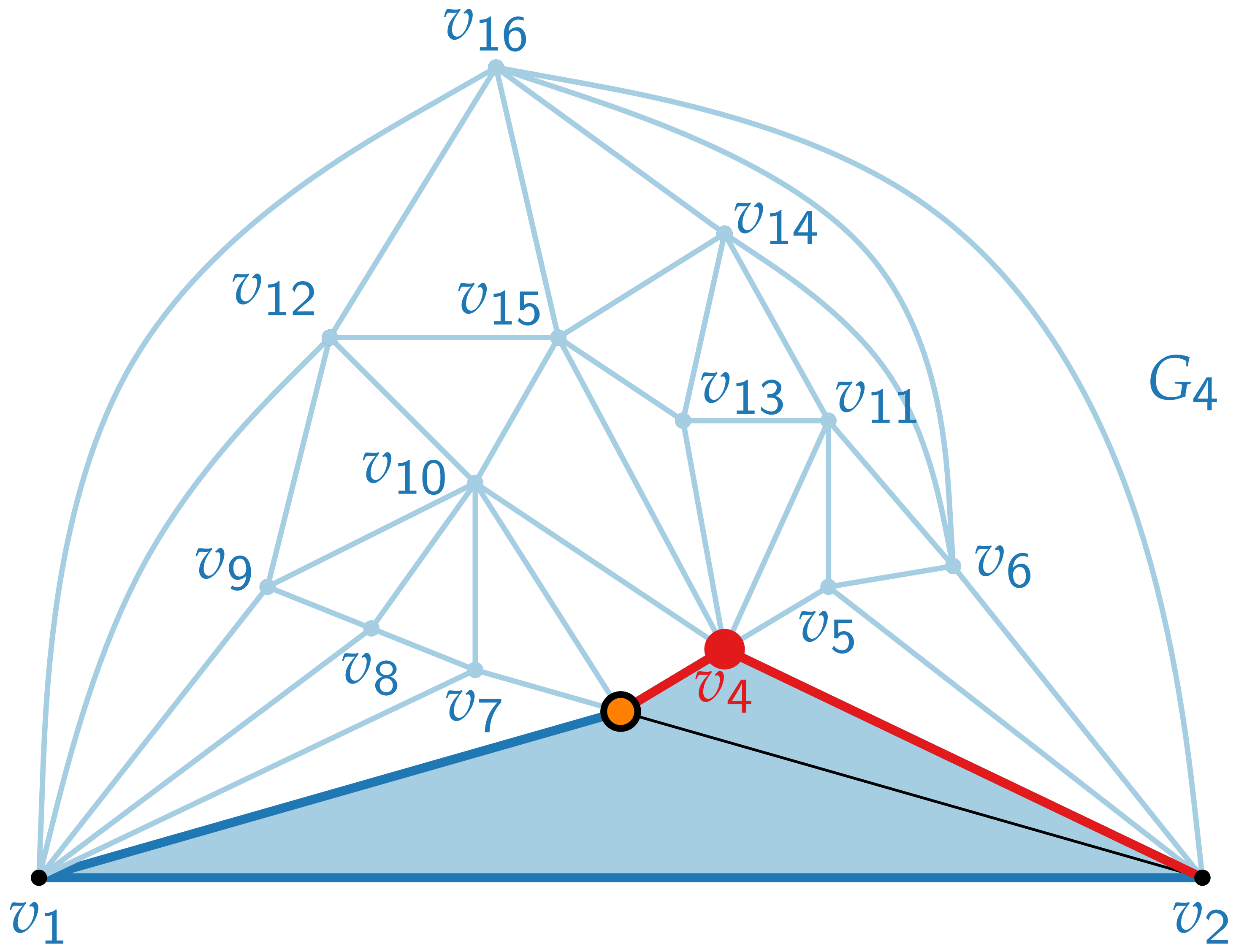
Canonical order – example



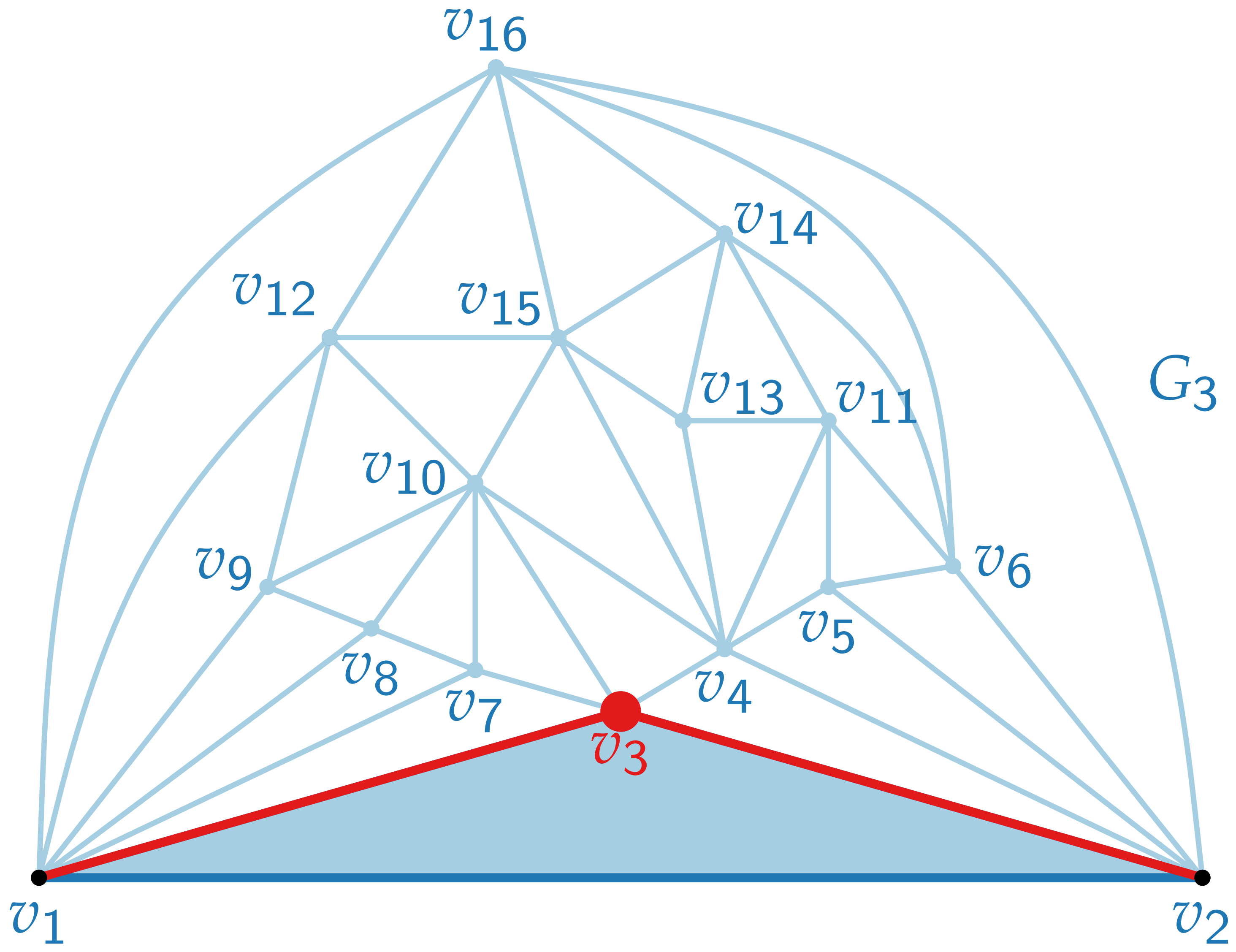
Canonical order – example



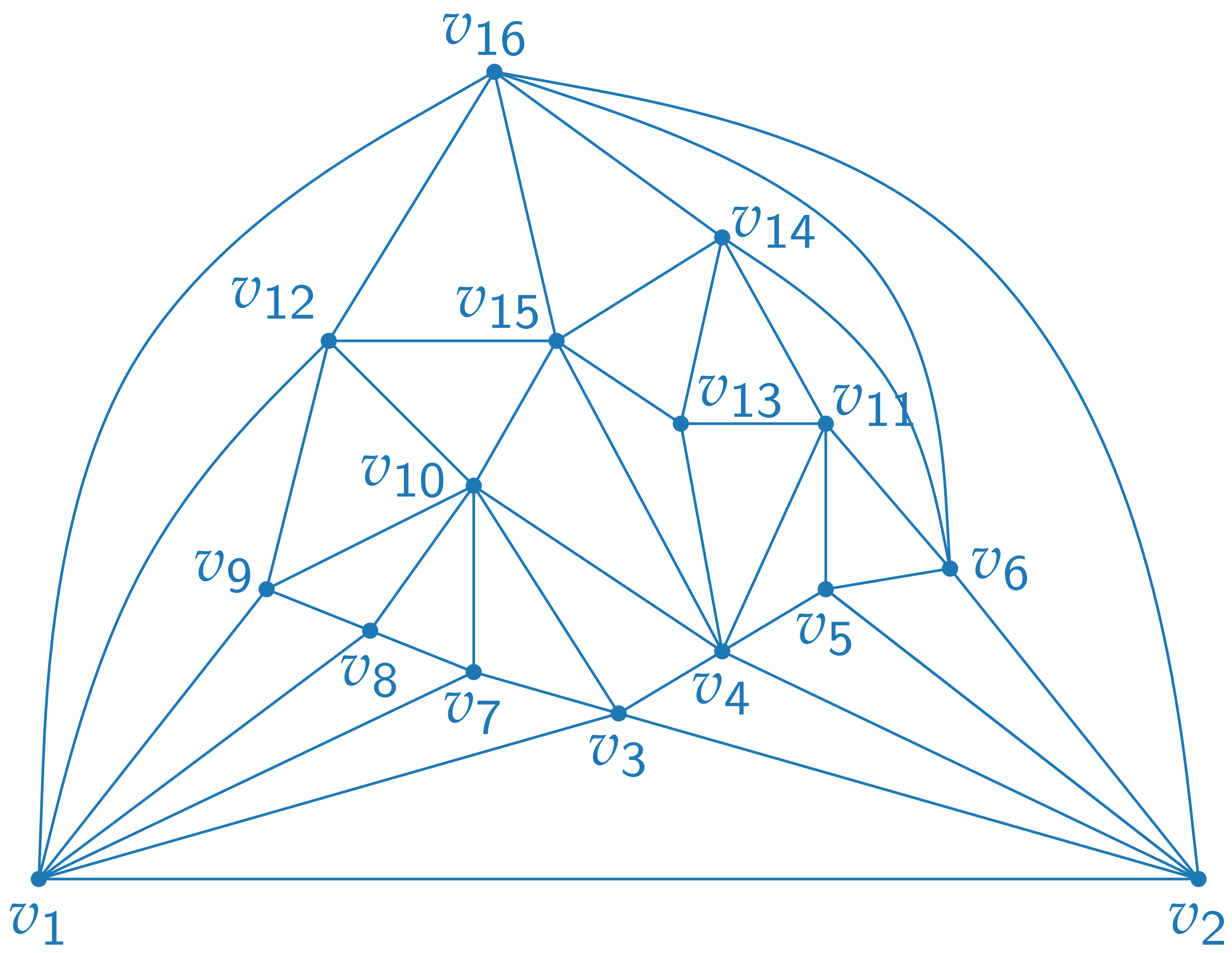
Canonical order – example



Canonical order – example



Canonical order – example



Canonical order – existence

Lemma.
Every triangulated plane graph has a canonical order.

Canonical order – existence

Lemma.
Every triangulated plane graph has a canonical order.

Proof.

- Let $G_n = G$, and let v_1, v_2, v_n be the vertices of the outer face of G_n. Conditions C1-C3 hold.

Canonical order – existence

Lemma.
Every triangulated plane graph has a canonical order.

Proof.

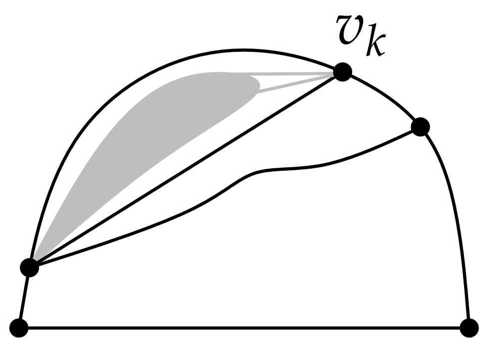
- Let $G_n = G$, and let v_1, v_2, v_n be the vertices of the outer face of G_n. Conditions C1-C3 hold.
- Induction hypothesis: Vertices $v_{n-1}, \ldots, v_{k+1}$ have been chosen such that conditions C1-C3 hold for $k+1 \leq i \leq n$.

Canonical order – existence

Lemma.
Every triangulated plane graph has a canonical order.

Proof.

- Let $G_n = G$, and let v_1, v_2, v_n be the vertices of the outer face of G_n. Conditions C1-C3 hold.
- Induction hypothesis: Vertices $v_{n-1}, \ldots, v_{k+1}$ have been chosen such that conditions C1-C3 hold for $k+1 \leq i \leq n$.
- Induction step: Consider G_k. We search for v_k.

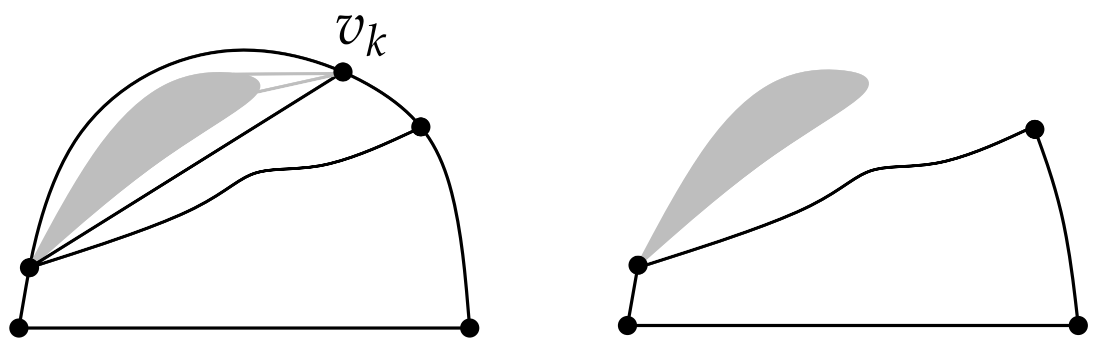


Canonical order – existence

Lemma.
Every triangulated plane graph has a canonical order.

Proof.

- Let $G_n = G$, and let v_1, v_2, v_n be the vertices of the outer face of G_n. Conditions C1-C3 hold.
- Induction hypothesis: Vertices $v_{n-1}, \ldots, v_{k+1}$ have been chosen such that conditions C1-C3 hold for $k+1 \leq i \leq n$.
- Induction step: Consider G_k. We search for v_k.

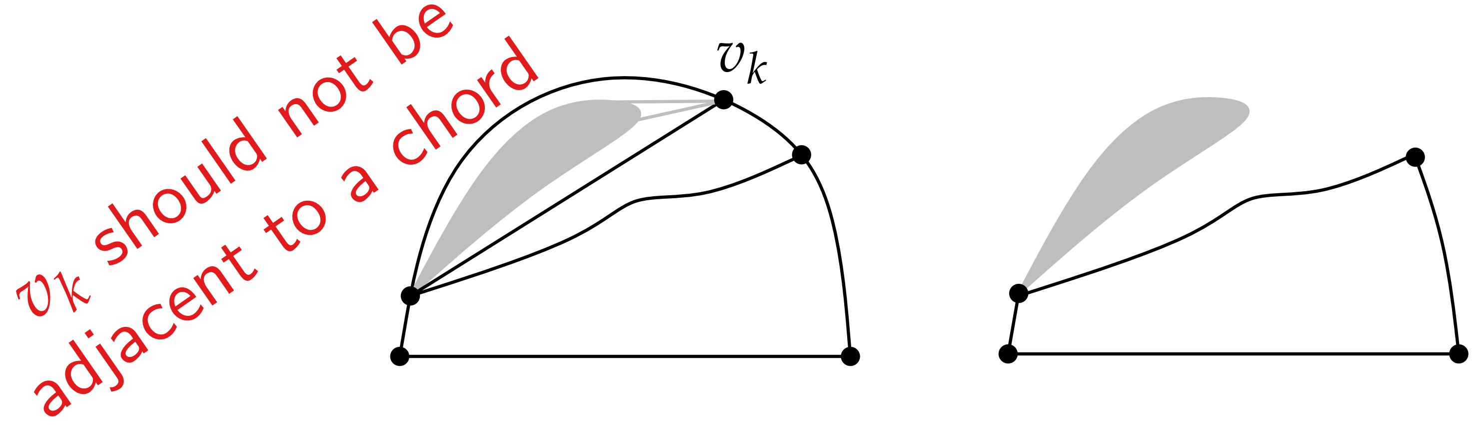


Canonical order – existence

Lemma.
Every triangulated plane graph has a canonical order.

Proof.

- Let $G_n = G$, and let v_1, v_2, v_n be the vertices of the outer face of G_n. Conditions C1-C3 hold.
- Induction hypothesis: Vertices $v_{n-1}, \ldots, v_{k+1}$ have been chosen such that conditions C1-C3 hold for $k+1 \leq i \leq n$.
- Induction step: Consider G_k. We search for v_k.

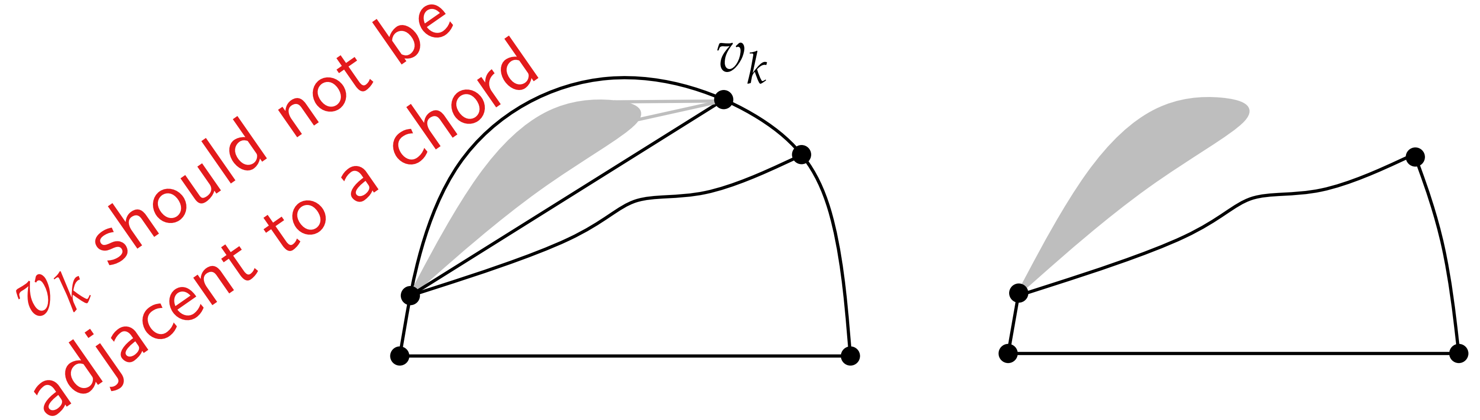


Canonical order – existence

Lemma.
Every triangulated plane graph has a canonical order.

Proof.

- Let $G_n = G$, and let v_1, v_2, v_n be the vertices of the outer face of G_n. Conditions C1-C3 hold.
- Induction hypothesis: Vertices $v_{n-1}, \ldots, v_{k+1}$ have been chosen such that conditions C1-C3 hold for $k+1 \leq i \leq n$.
- Induction step: Consider G_k. We search for v_k.



Have to show:

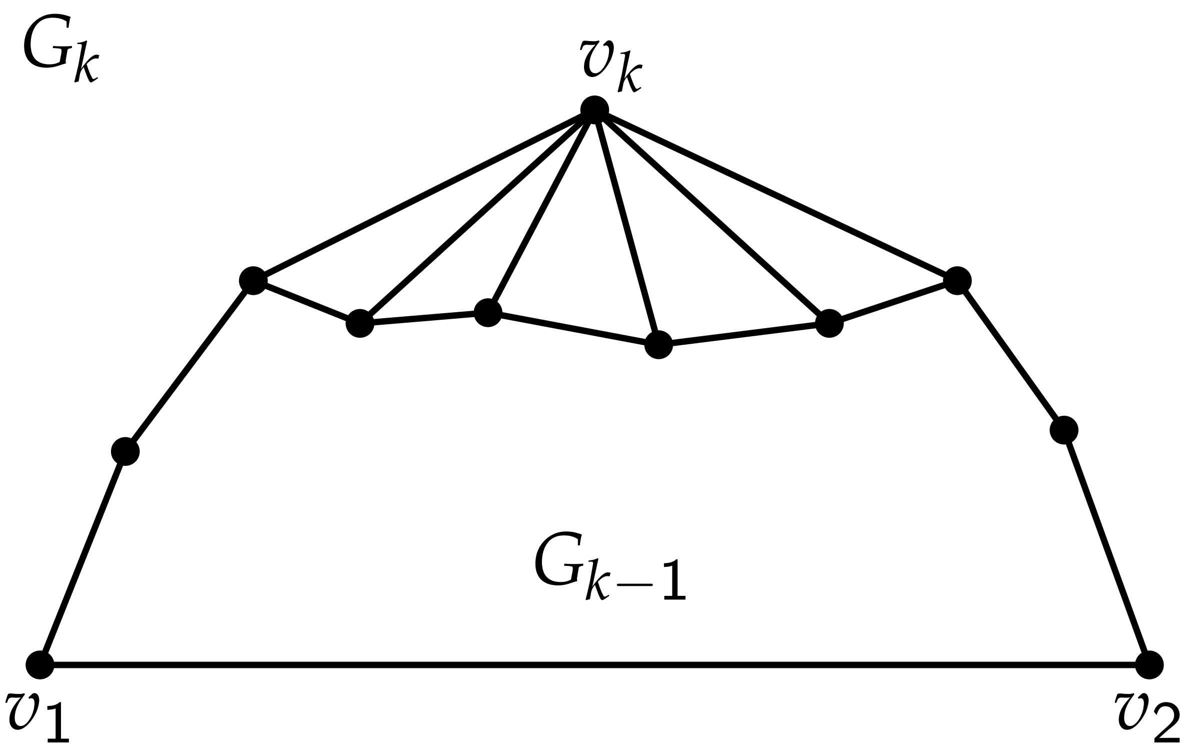
1. v_k not adjacent to chord is sufficient
2. Such v_k exists

Canonical order – existence

Claim 1. If v_k is not adjacent to a chord then removal of v_k leaves the graph biconnected.

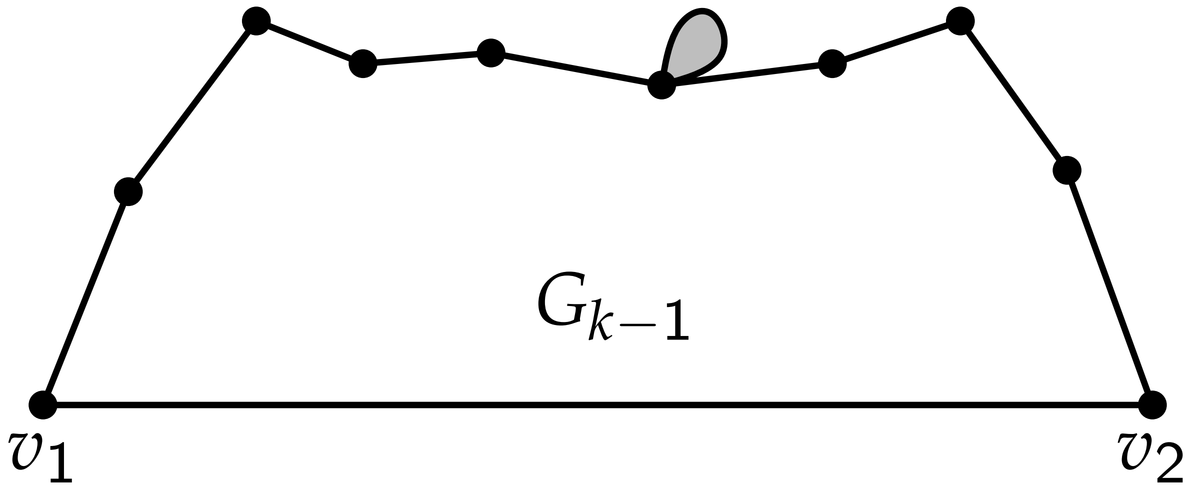
Canonical order – existence

Claim 1. If v_k is not adjacent to a chord then removal of v_k leaves the graph biconnected.



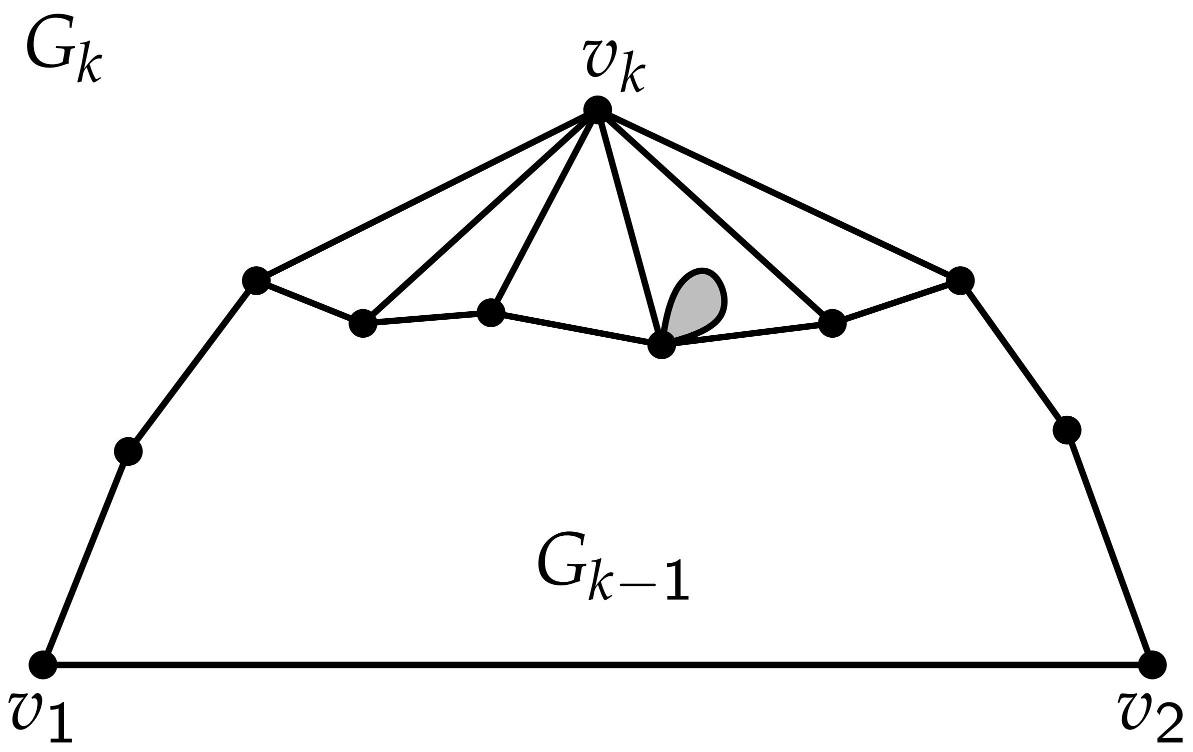
Canonical order – existence

Claim 1. If v_k is not adjacent to a chord then removal of v_k leaves the graph biconnected.



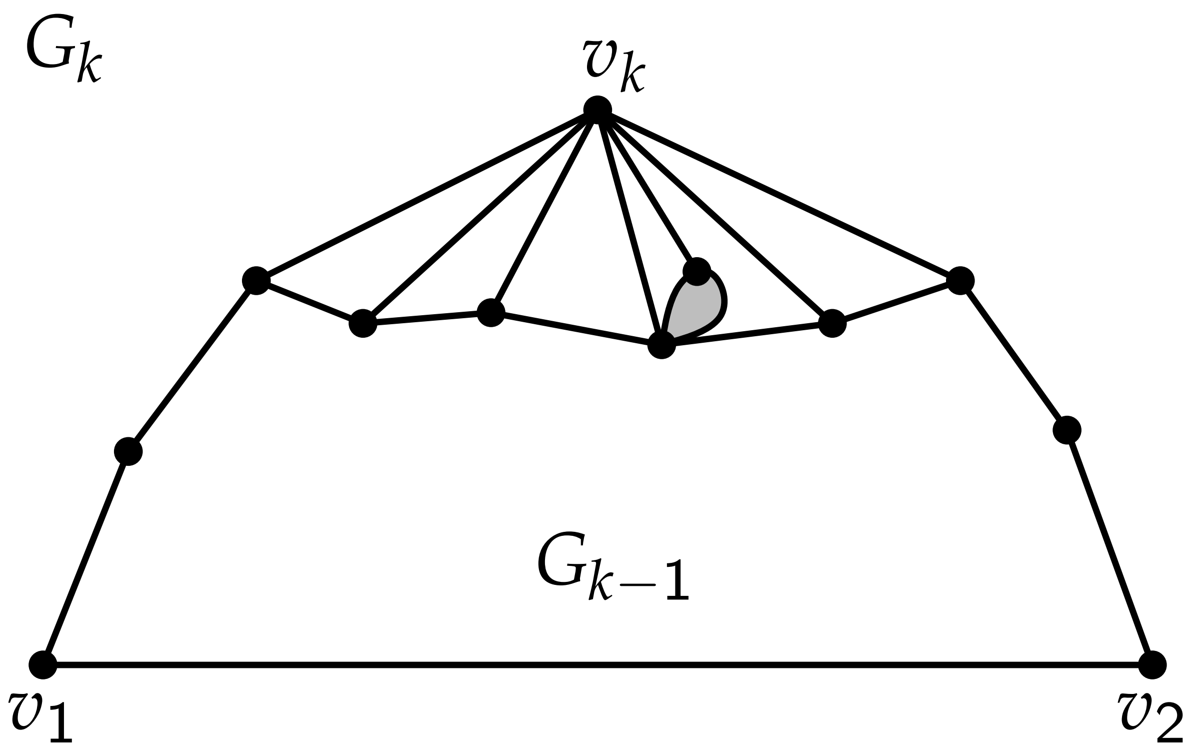
Canonical order – existence

Claim 1. If v_k is not adjacent to a chord then removal of v_k leaves the graph biconnected.



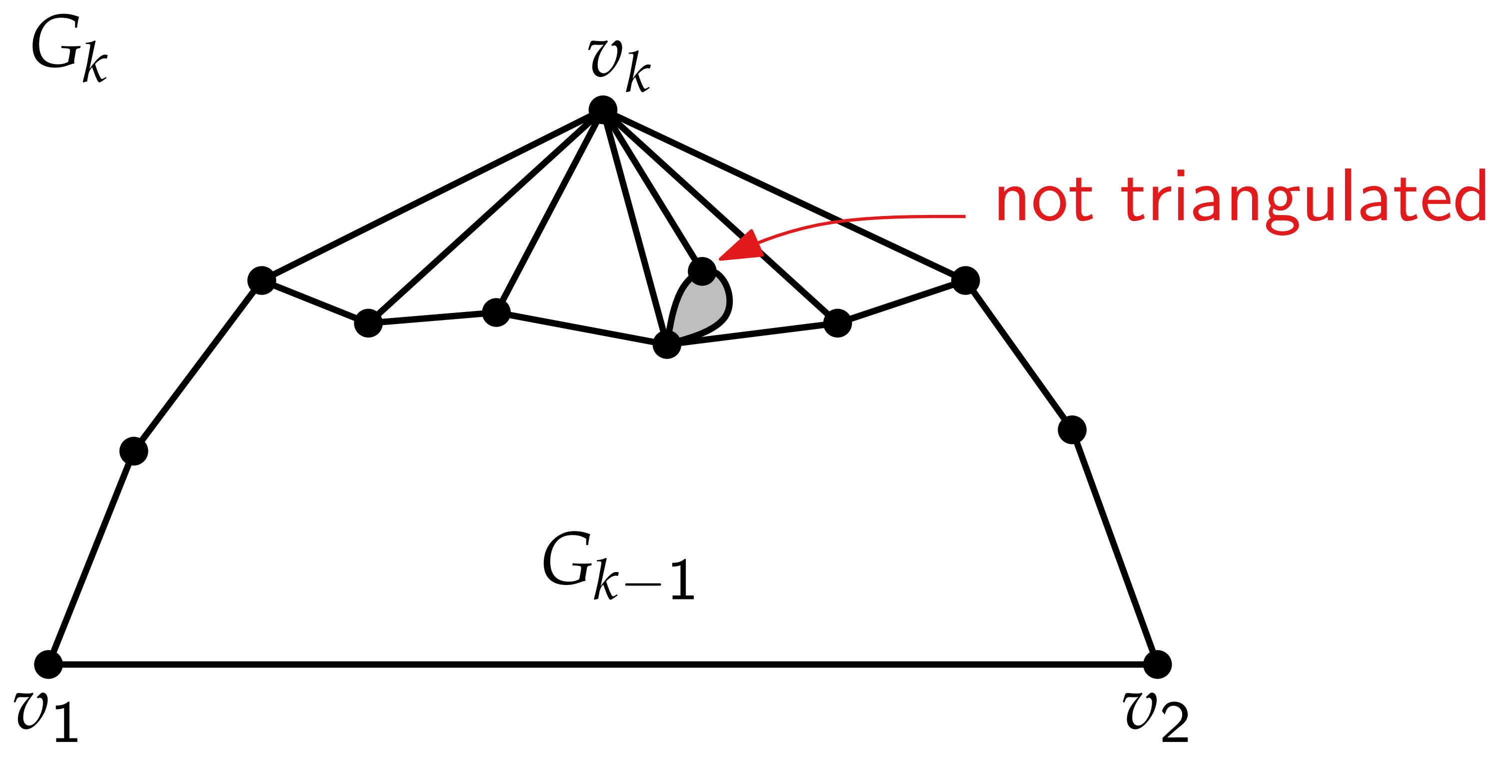
Canonical order – existence

Claim 1. If v_k is not adjacent to a chord then removal of v_k leaves the graph biconnected.



Canonical order – existence

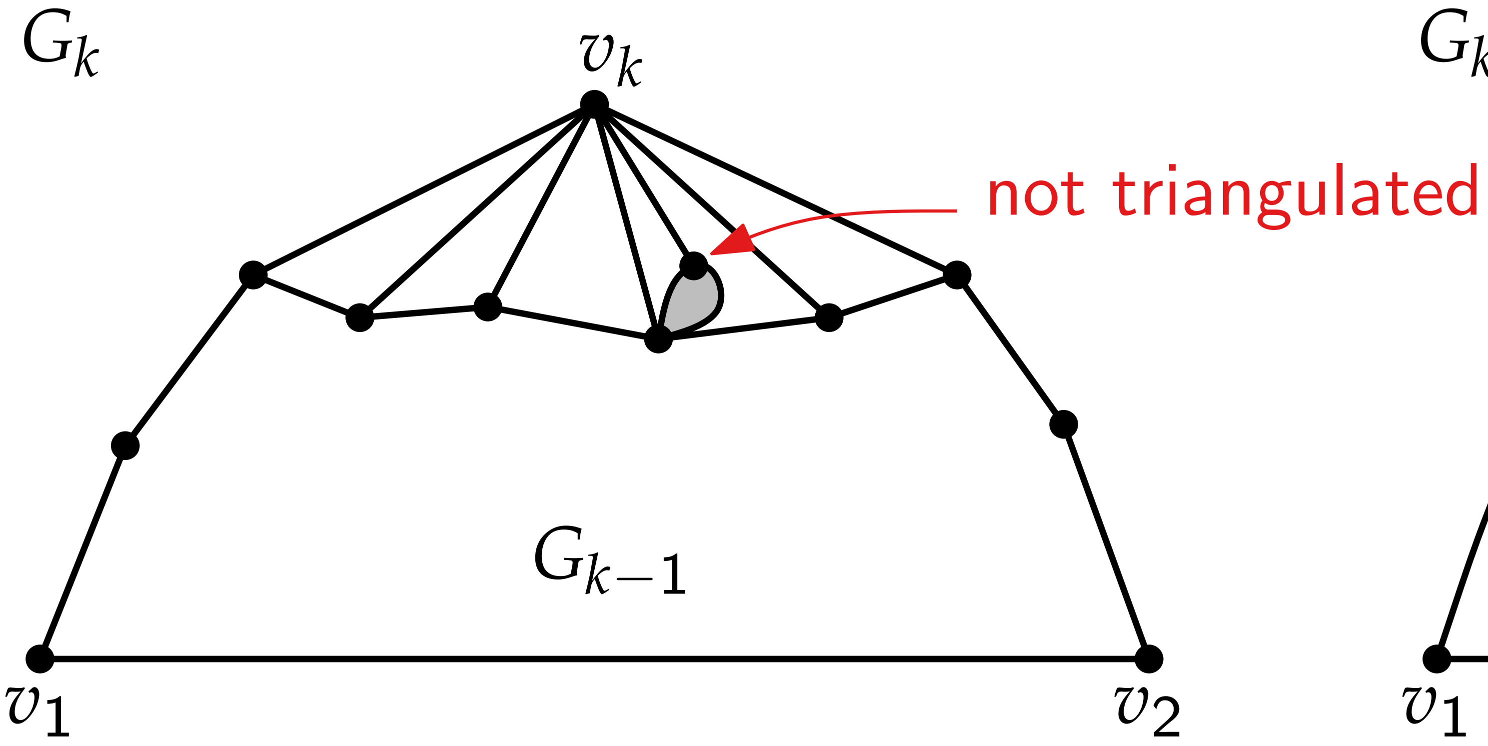
Claim 1. If v_k is not adjacent to a chord then removal of v_k leaves the graph biconnected.

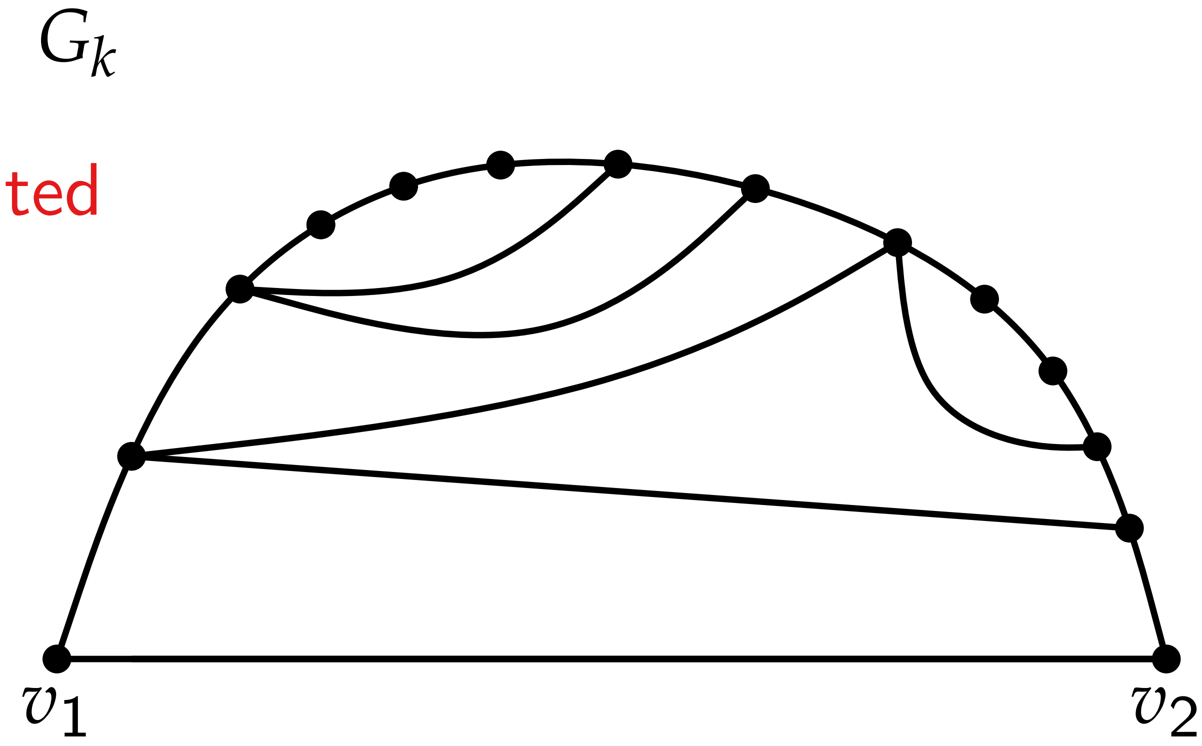


Canonical order – existence

Claim 1. If v_k is not adjacent to a chord then removal of v_k leaves the graph biconnected.

Claim 2. There exists a vertex in G_k that is not adjacent to a chord as choice for v_k.

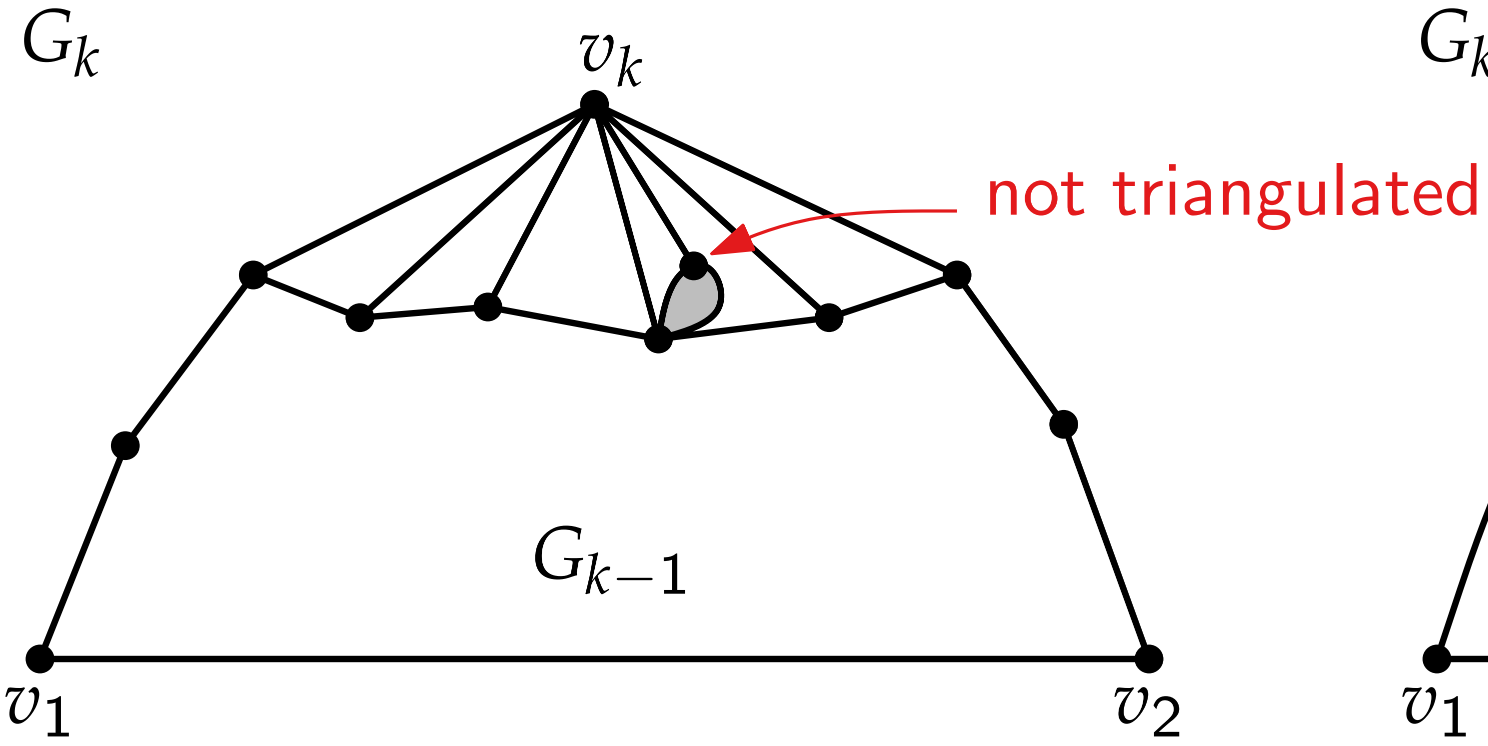


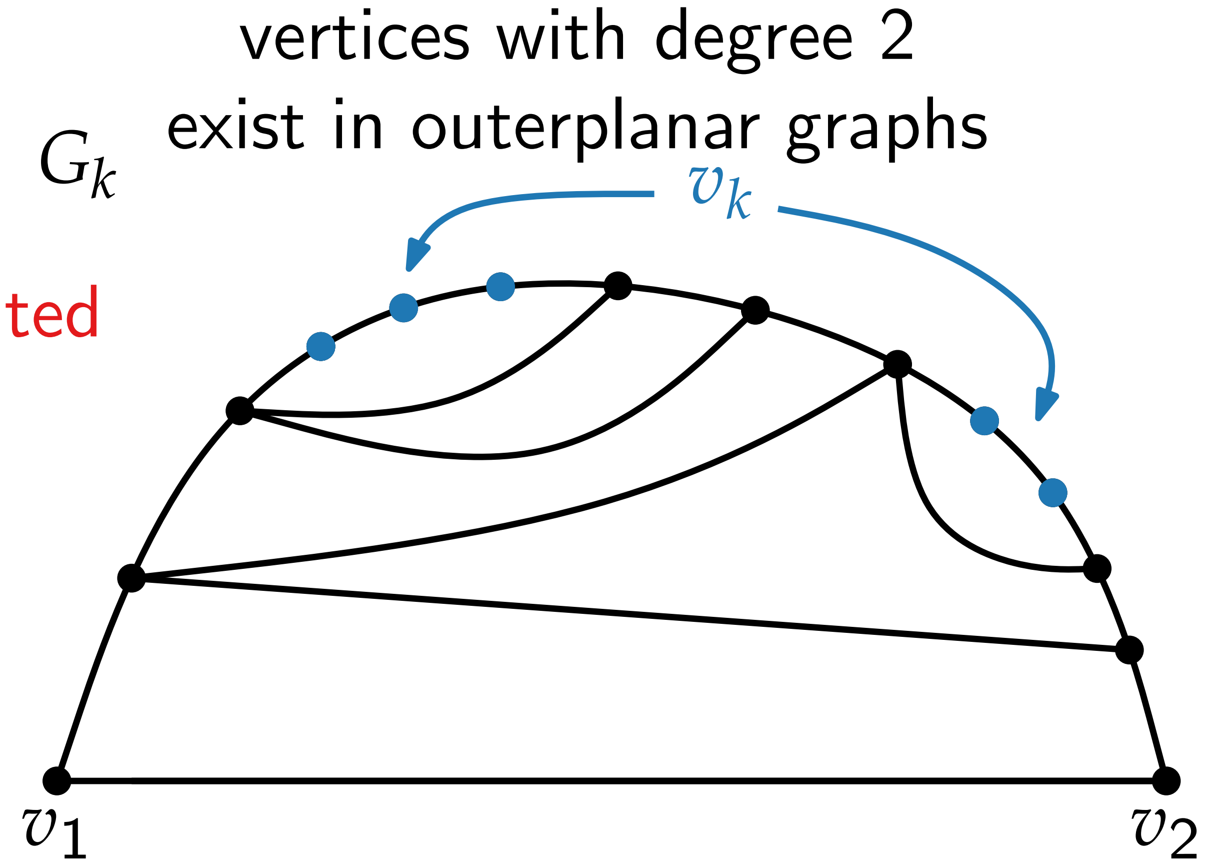


Canonical order – existence

Claim 1. If v_k is not adjacent to a chord then removal of v_k leaves the graph biconnected.

Claim 2. There exists a vertex in G_k that is not adjacent to a chord as choice for v_k.

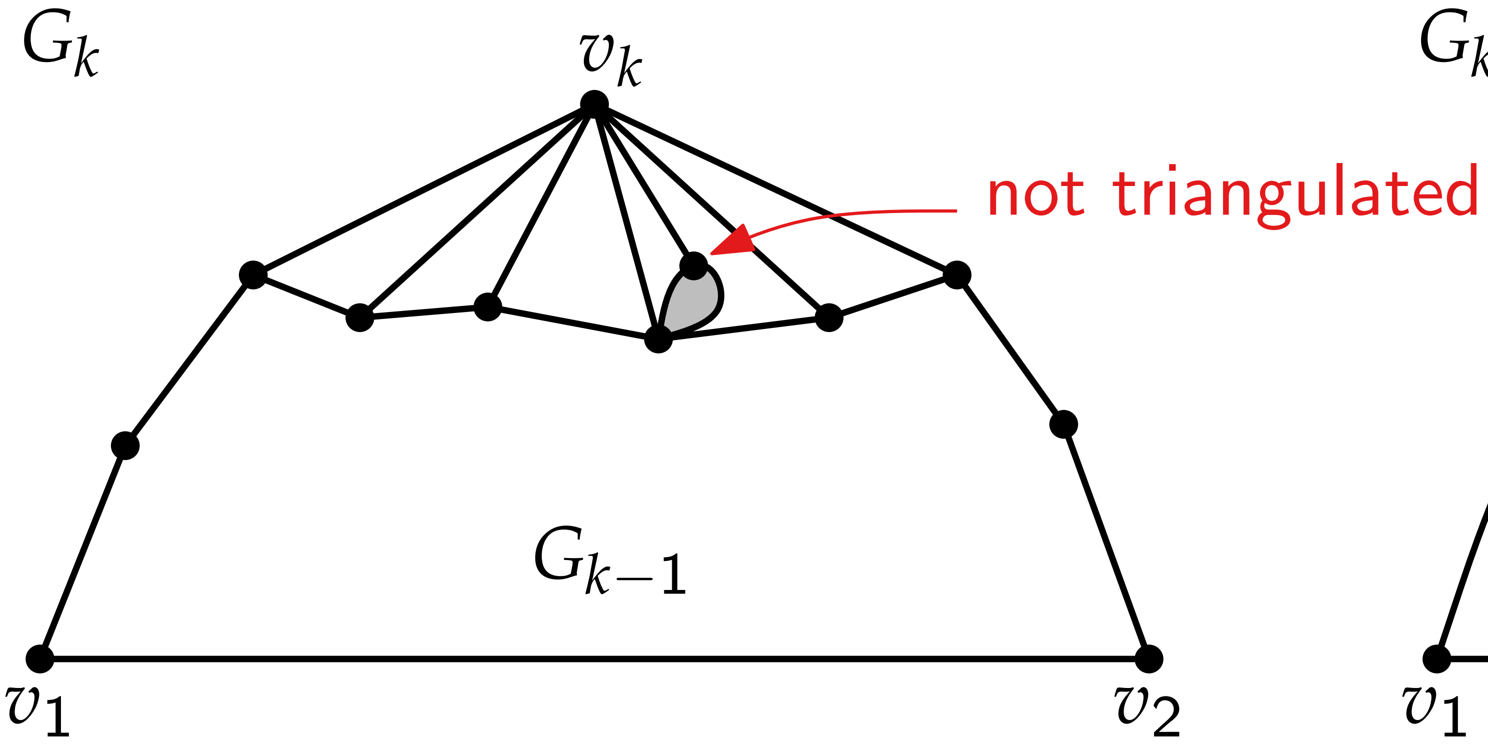


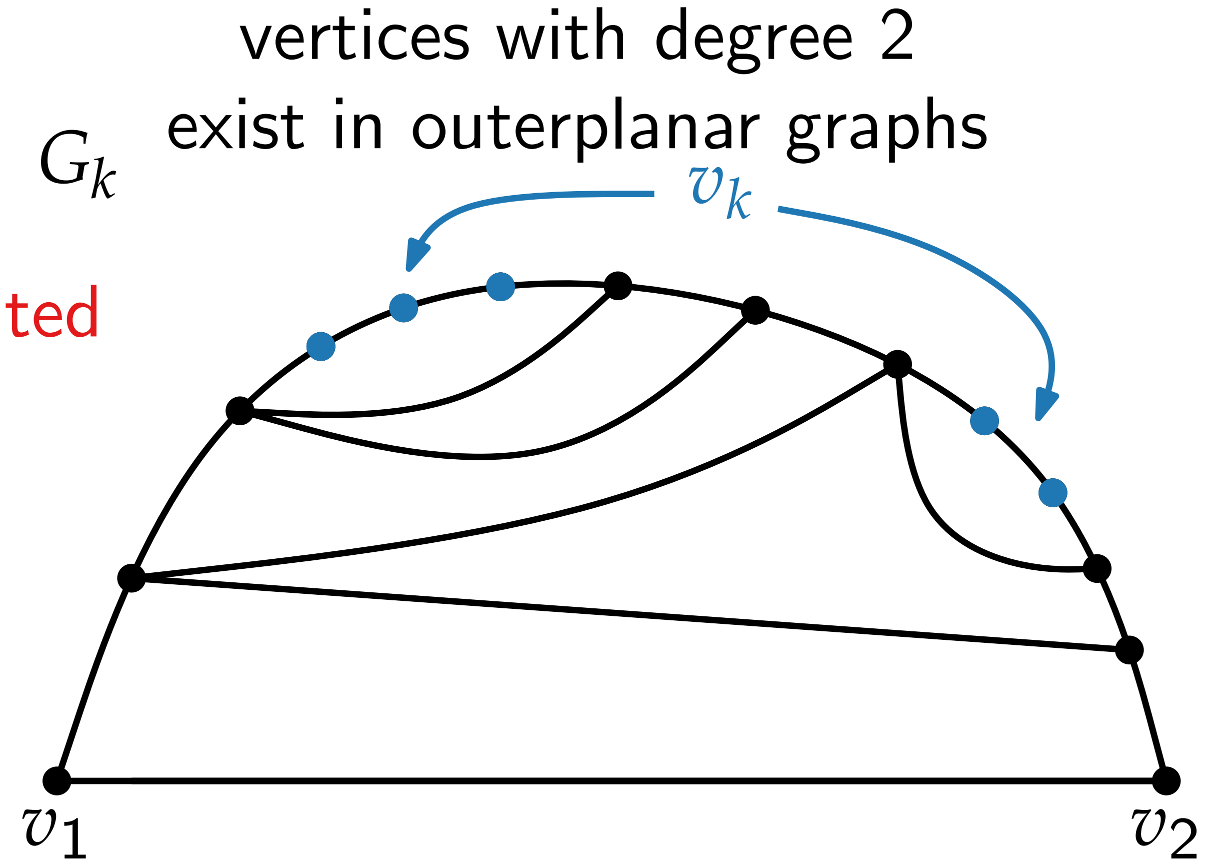


Canonical order – existence

Claim 1. If v_k is not adjacent to a chord then removal of v_k leaves the graph biconnected.

Claim 2. There exists a vertex in G_k that is not adjacent to a chord as choice for v_k.

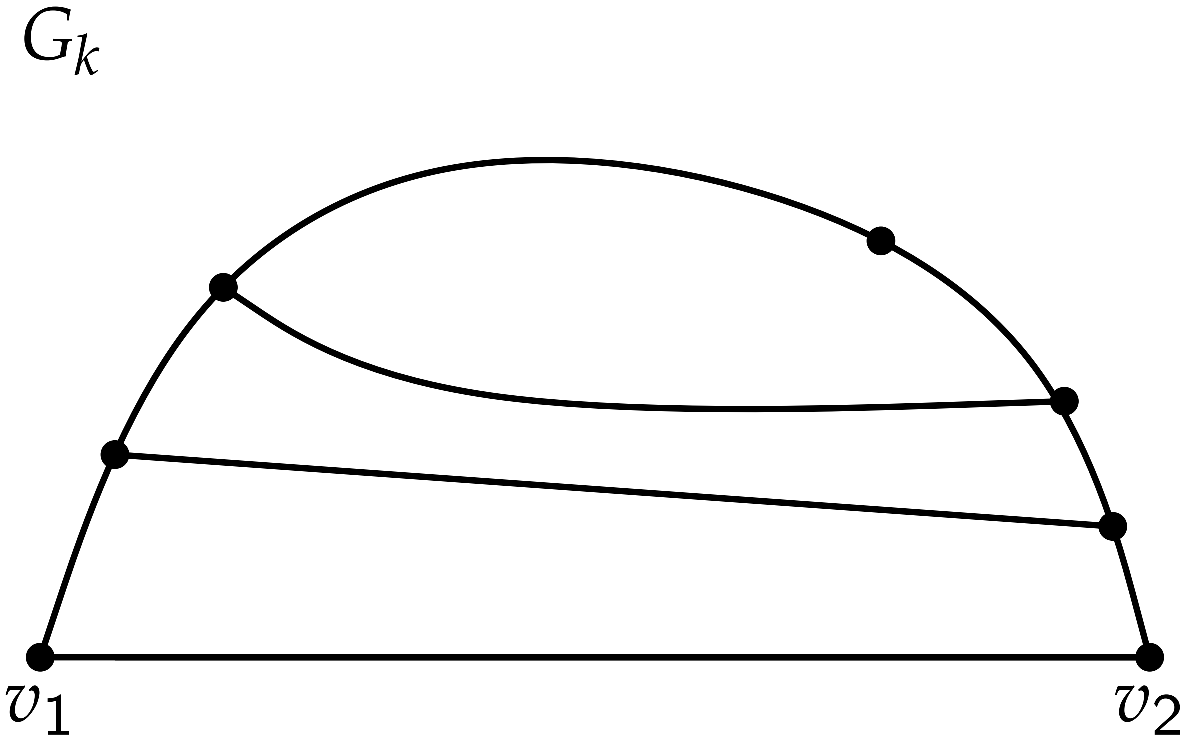




This completes proof of Lemma. □

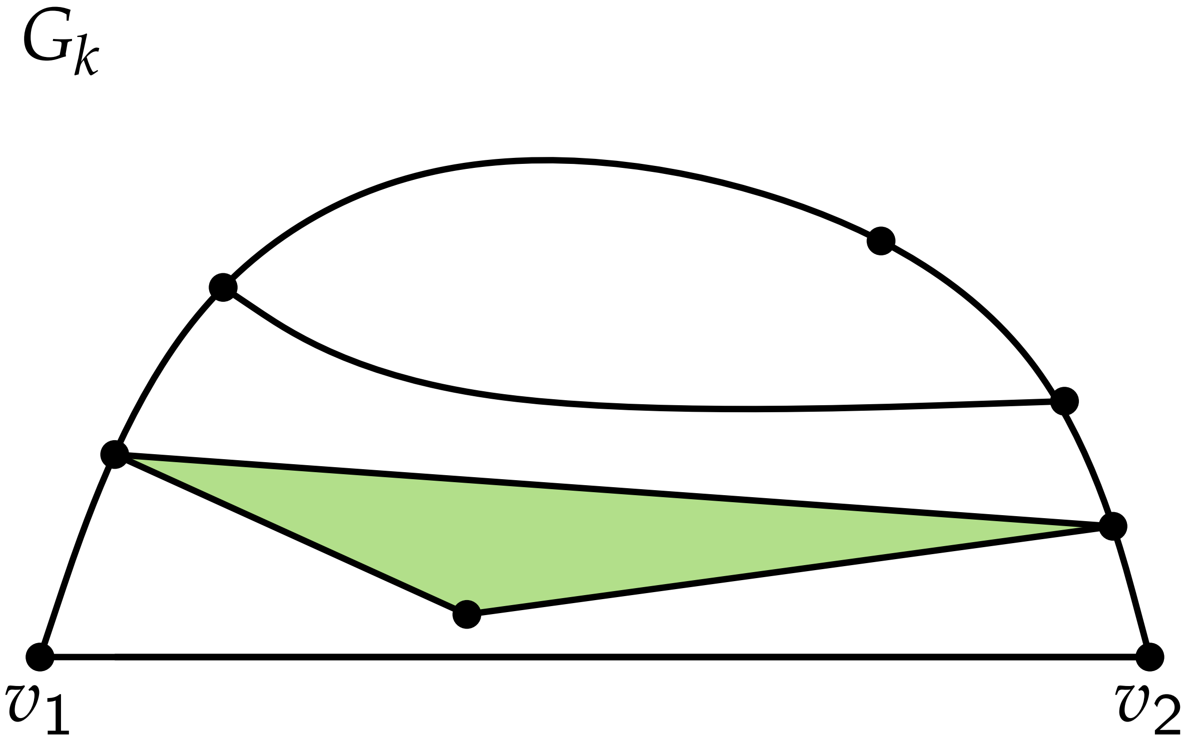
Canonical order – implementation

- chords of G_k belong to faces:



Canonical order – implementation

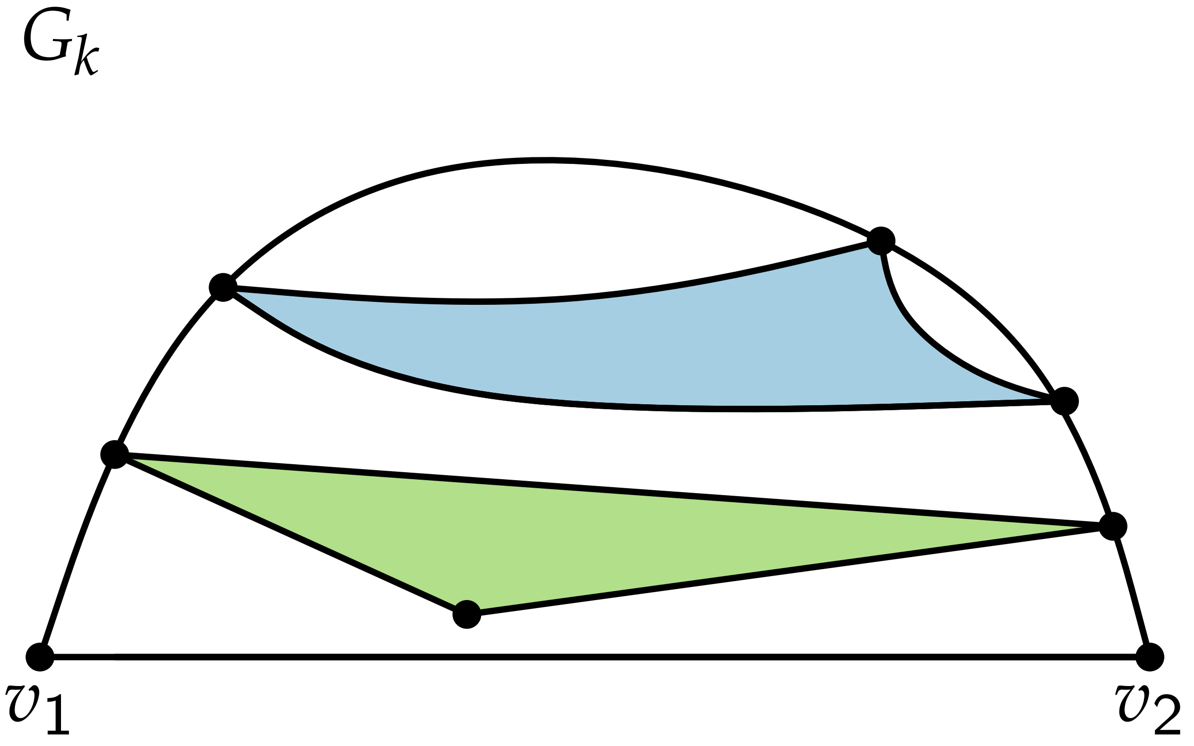
- chords of G_k belong to faces:



- f has two vertices on the outerface and one internal

Canonical order – implementation

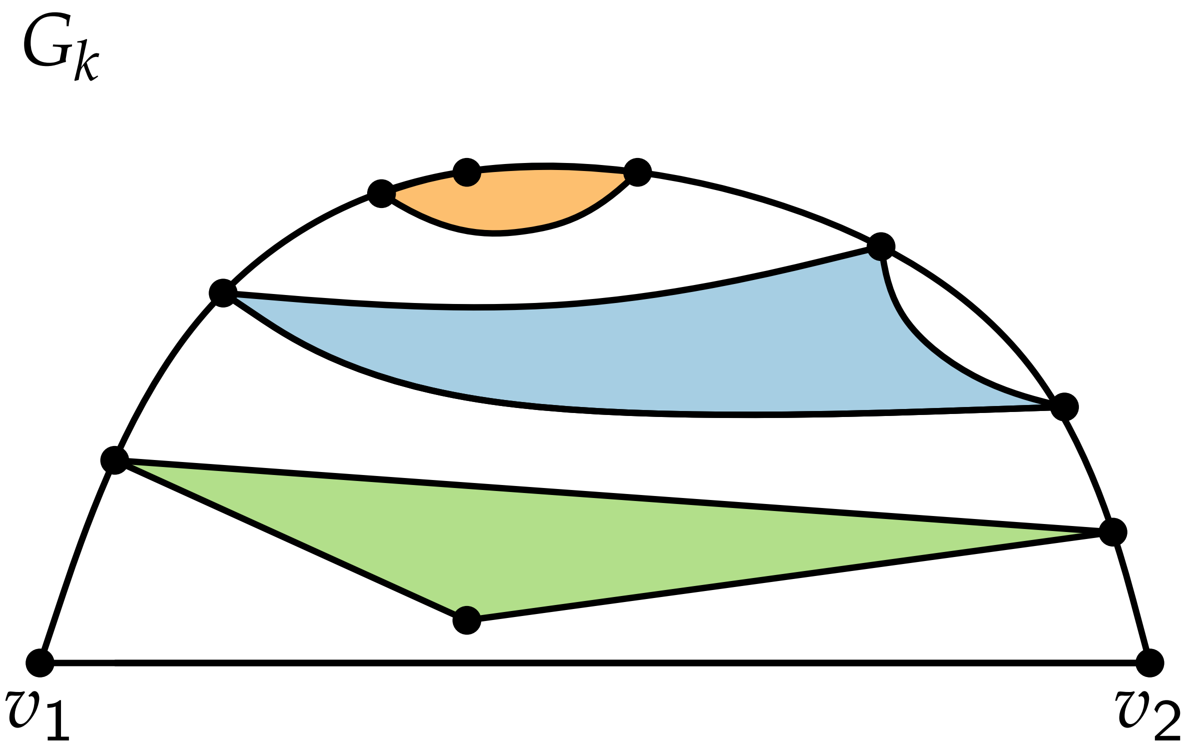
- chords of G_k belong to faces:



- f has two vertices on the outerface and one internal
- f has three vertices on the outerface and at least two chords

Canonical order – implementation

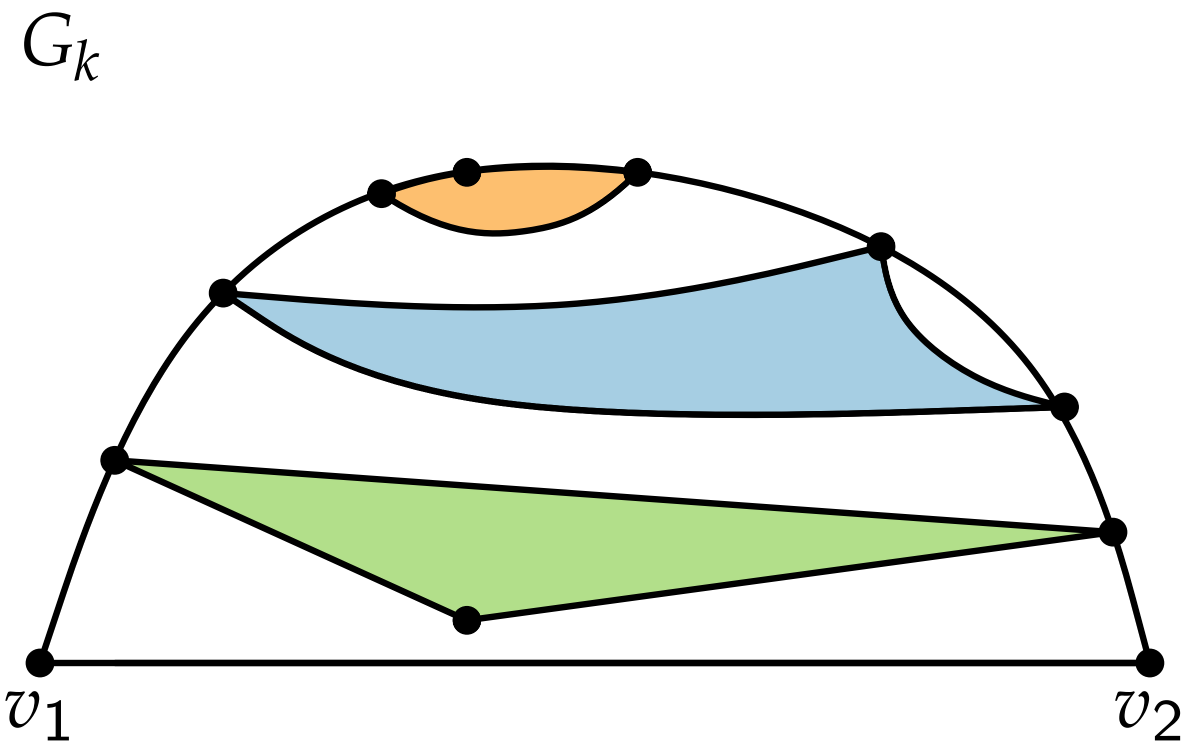
- chords of G_k belong to faces:



- f has two vertices on the outerface and one internal
- f has three vertices on the outerface and at least two chords
- f has three consequtive vertices on the outerface

Canonical order – implementation

- chords of G_k belong to faces:

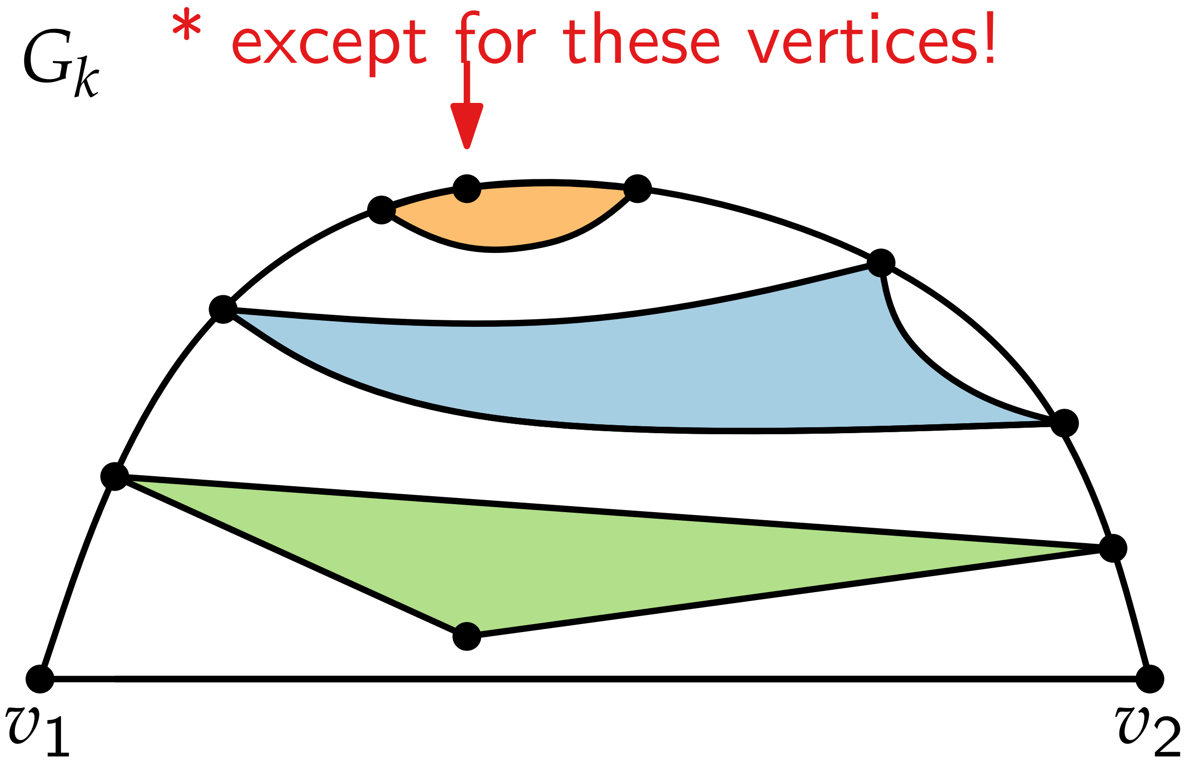


- chords are associated with separating faces
- v_k belongs to no separating faces *

- f has two vertices on the outerface and one internal
- f has three vertices on the outerface and at least two chords
- f has three consequtive vertices on the outerface

Canonical order – implementation

- chords of G_k belong to faces:

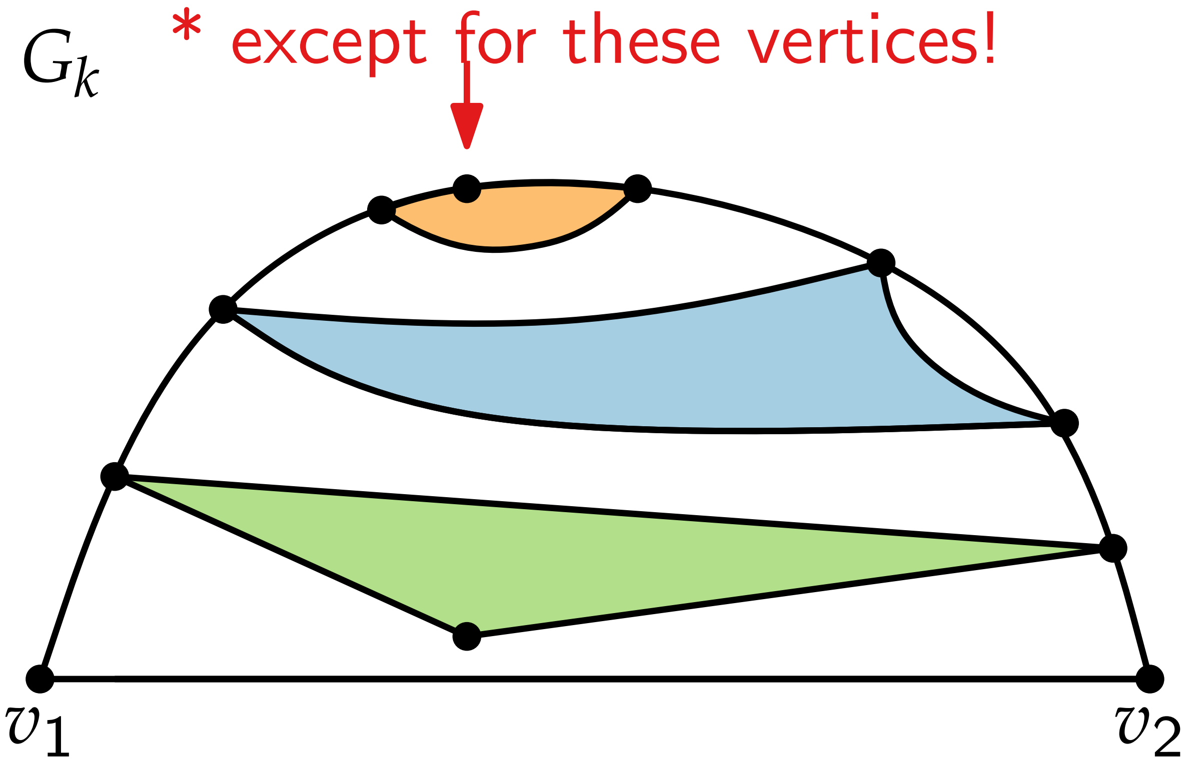


- chords are associated with separating faces
- v_k belongs to no separating faces *

- f has two vertices on the outerface and one internal
- f has three vertices on the outerface and at least two chords
- f has three consequtive vertices on the outerface

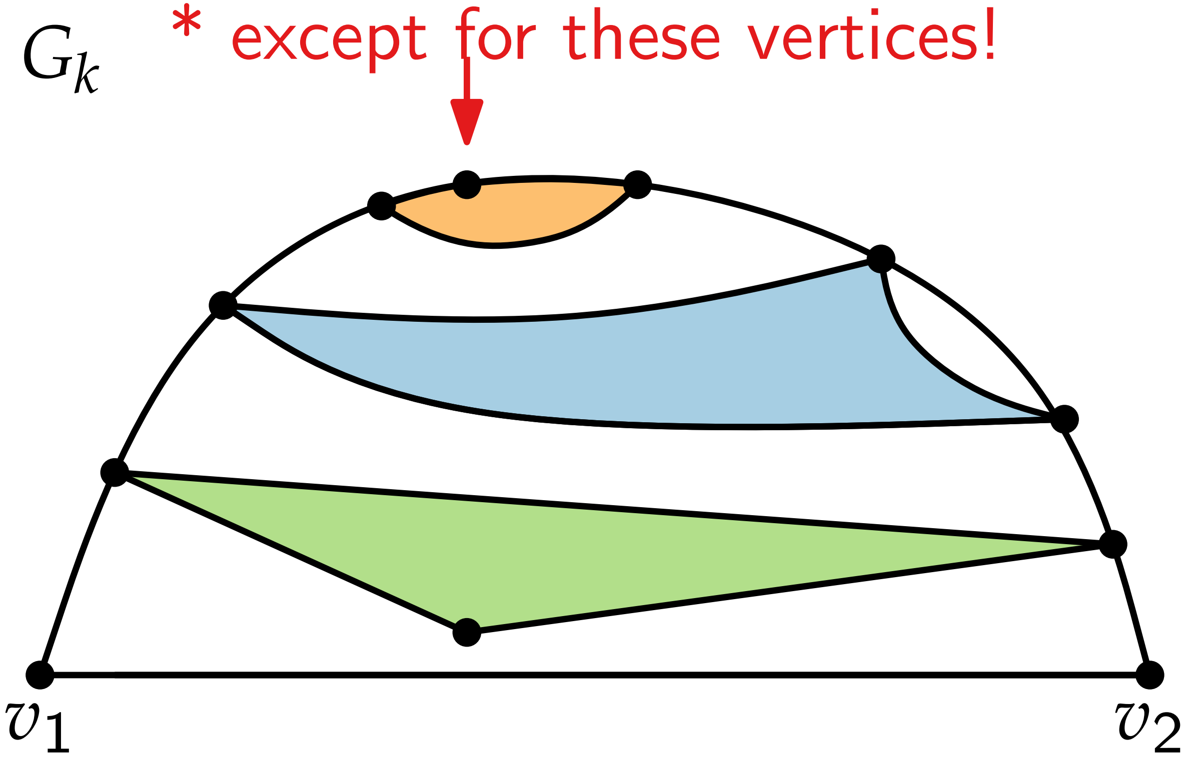
Canonical order – implementation

- f_{out} = current outerface
- $F(v)$ = faces that contain v
- $F(e)$ = faces that contain e



- chords are associated with separating faces
- v_k belongs to no separating faces *

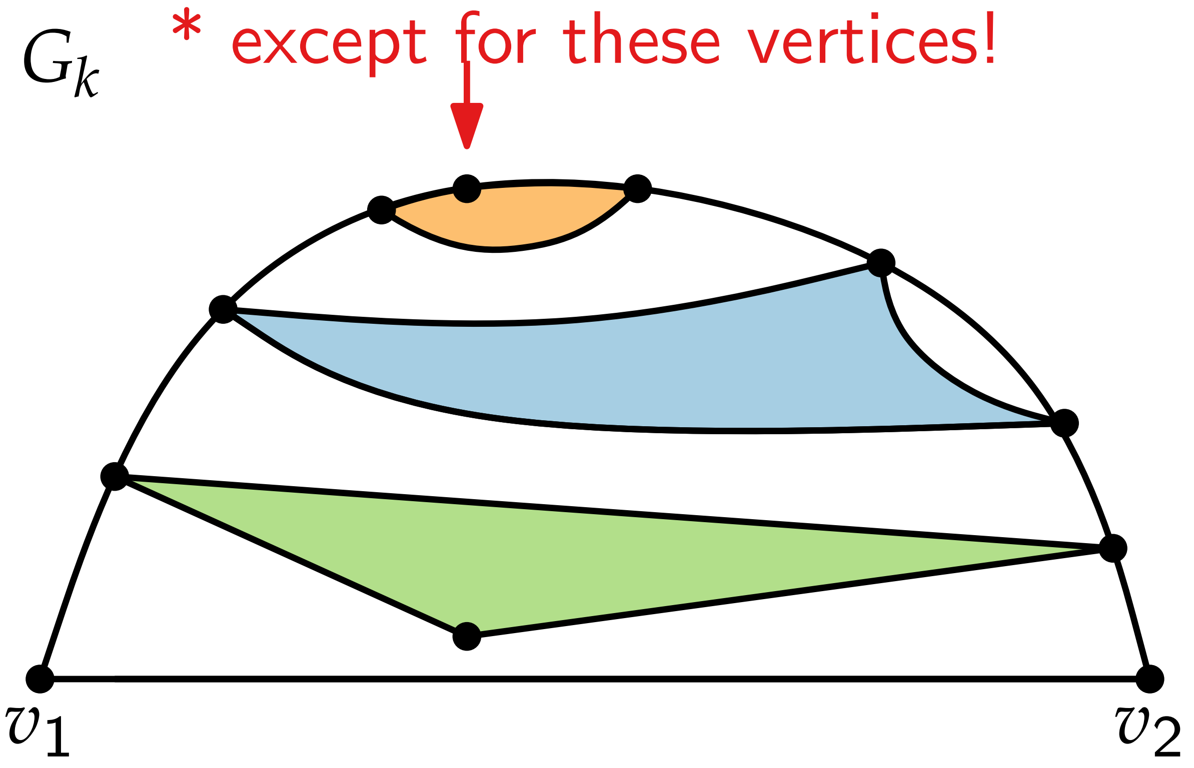
Canonical order – implementation



- chords are associated with separating faces
- v_k belongs to no separating faces *

- f_{out} = current outerface
- $F(v)$ = faces that contain v
- $F(e)$ = faces that contain e
- outV(f) = # vertices of f on f_{out}
- outE(f) = # edges of f on f_{out}
- sepF(v) = # separation faces that contain v

Canonical order – implementation



- chords are associated with separating faces
- v_k belongs to no separating faces *

- f_{out} = current outerface
- $F(v)$ = faces that contain v
- $F(e)$ = faces that contain e
- outV(f) = # vertices of f on f_{out}
- outE(f) = # edges of f on f_{out}
- sepF(v) = # separation faces that contain v

$f \in F(v)$ is separating iff
- outV(f)=3 or
- outV(f)=2 and outE(f)=0

Canonical order – implementation

- f_{out} = current outerface
- $F(v)$ = faces that contain v
- $F(e)$ = faces that contain e
- outV(f) = # vertices of f on f_{out}
- outE(f) = # edges of f on f_{out}
- sepF(v) = # separation faces that contain v

Canonical order – implementation

Algorithm CanonicalOrder- Initialization

forall $v \in V$ **do**
  sepF(v) $\leftarrow$ 0;
forall $f \in F$ **do**
  outV(f), outE(f) $\leftarrow$ 0;

- f_{out} = current outerface
- $F(v)$ = faces that contain v
- $F(e)$ = faces that contain e
- outV(f) = # vertices of f on f_{out}
- outE(f) = # edges of f on f_{out}
- sepF(v) = # separation faces that contain v

Canonical order – implementation

Algorithm CanonicalOrder- Initialization

forall $v \in V$ **do**
  sepF(v) $\leftarrow 0$;

forall $f \in F$ **do**
  outV(f), outE(f) $\leftarrow 0$;

forall $v \in f_{out}$ **do**
  forall $f \in F(v)$: $f \neq f_{out}$ **do**
    outV(f)++;

forall $e \in f_{out}$ **do**
  forall $f \in F(e)$: $f \neq f_{out}$ **do**
    outE(f)++;

- f_{out} = current outerface
- $F(v)$ = faces that contain v
- $F(e)$ = faces that contain e
- outV(f) = # vertices of f on f_{out}
- outE(f) = # edges of f on f_{out}
- sepF(v) = # separation faces that contain v

Canonical order – implementation

Algorithm CanonicalOrder- Initialization

forall $v \in V$ **do**
  sepF(v) $\leftarrow$ 0;

forall $f \in F$ **do**
  outV(f), outE(f) $\leftarrow$ 0;

forall $v \in f_{out}$ **do**
  forall $f \in F(v)$: $f \neq f_{out}$ **do**
    outV(f)++;

forall $e \in f_{out}$ **do**
  forall $f \in F(e)$: $f \neq f_{out}$ **do**
    outE(f)++;

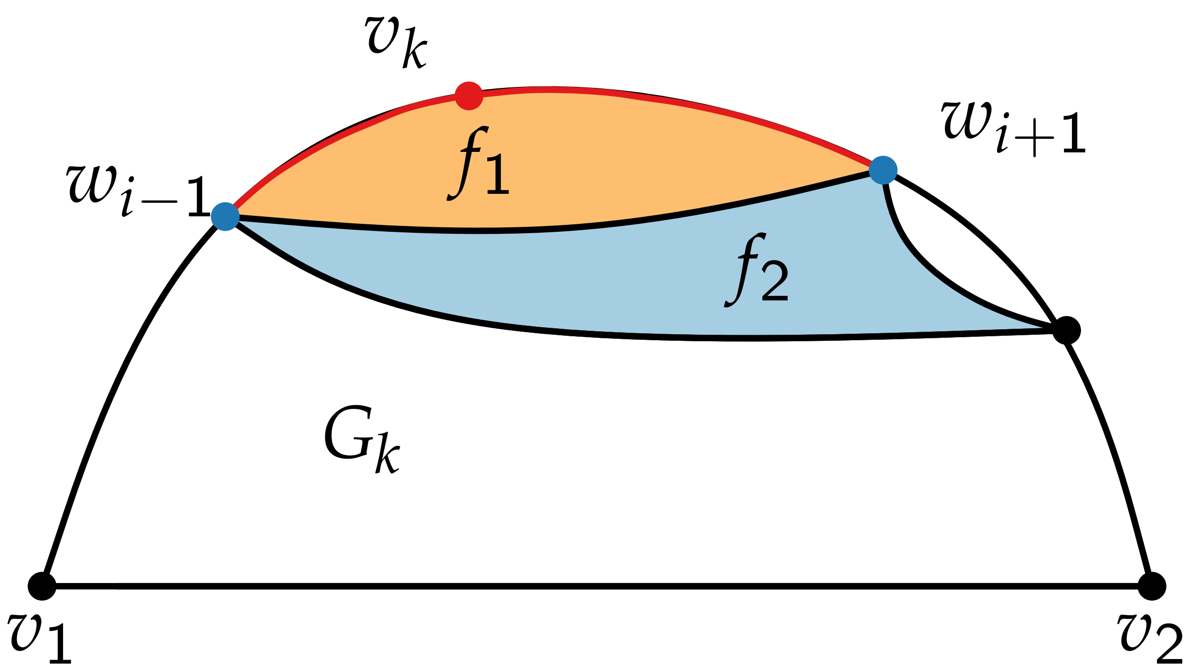
- f_{out} = current outerface
- $F(v)$ = faces that contain v
- $F(e)$ = faces that contain e
- outV(f) = # vertices of f on f_{out}
- outE(f) = # edges of f on f_{out}
- sepF(v) = # separation faces that contain v

forall $v \in f_{out}$ **do**
  forall $f \in F(v)$: $f \neq f_{out}$ **do**
    if outV(f)=3 or outV(f)=2 and outE(f)=0 **then**
      sepF(v)++;

Canonical order – implementation

Remove degree 2 vertex v_k

- f_{out} = current outerface
- $F(v)$ = faces that contain v
- $F(e)$ = faces that contain e
- outV(f) = # vertices of f on f_{out}
- outE(f) = # edges of f on f_{out}
- sepF(v) = # separation faces that contain v

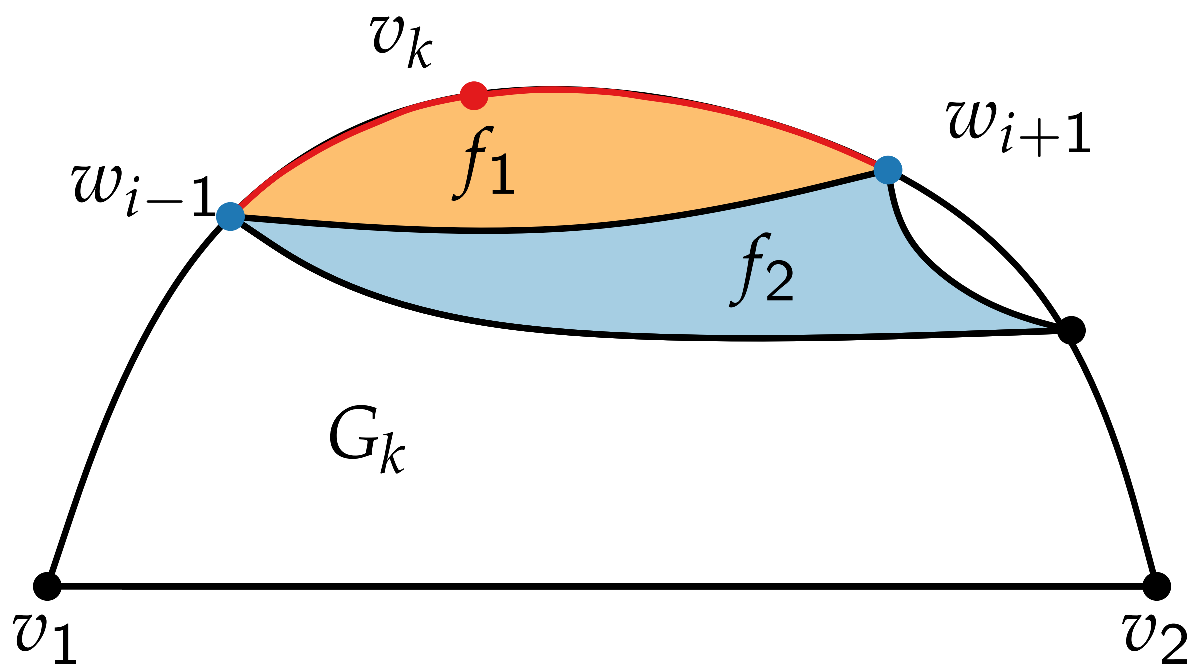


Canonical order – implementation

Remove degree 2 vertex v_k

- v_k and f_1 are removed
- outE(f_2) increases by one
- sepF(w_{i-1}) decreases by one
- sepF(w_{i+1}) decreases by one

- f_{out} = current outerface
- $F(v)$ = faces that contain v
- $F(e)$ = faces that contain e
- outV(f) = # vertices of f on f_{out}
- outE(f) = # edges of f on f_{out}
- sepF(v) = # separation faces that contain v



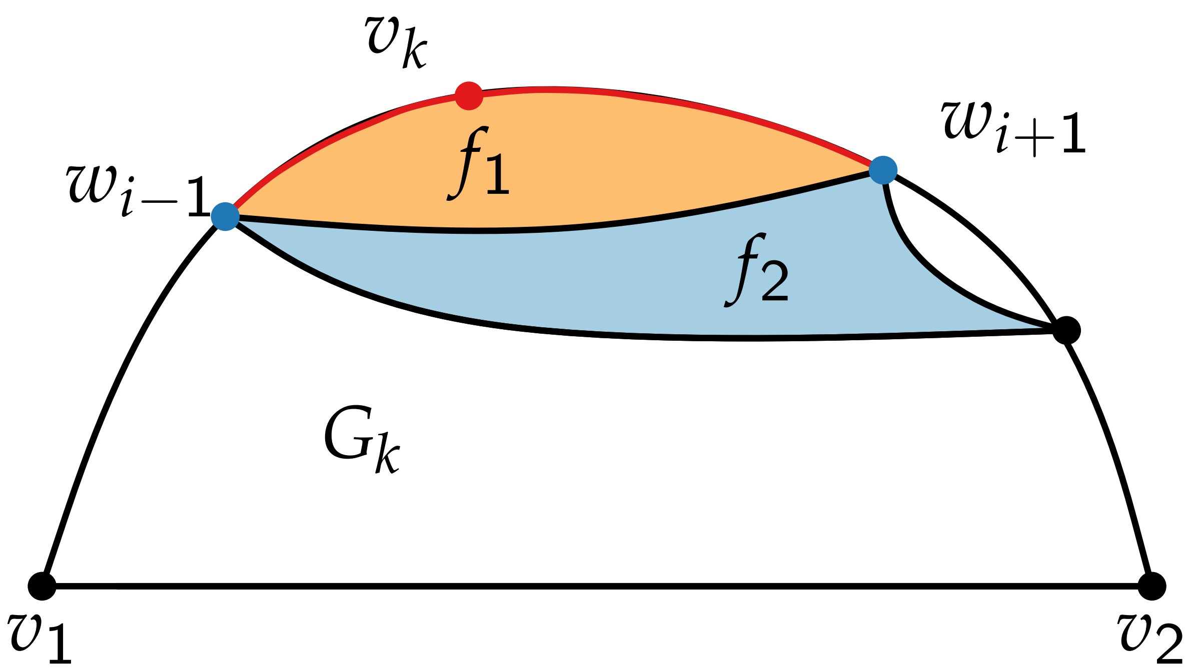
Canonical order – implementation

Remove degree 2 vertex v_k

- v_k and f_1 are removed
- outE(f_2) increases by one
- sepF(w_{i-1}) decreases by one
- sepF(w_{i+1}) decreases by one

- if f_2 has outV(f_2)=2, f_2 is not a separating face
  - sepF(w_{i-1}) decreases by one
  - sepF(w_{i+1}) decreases by one

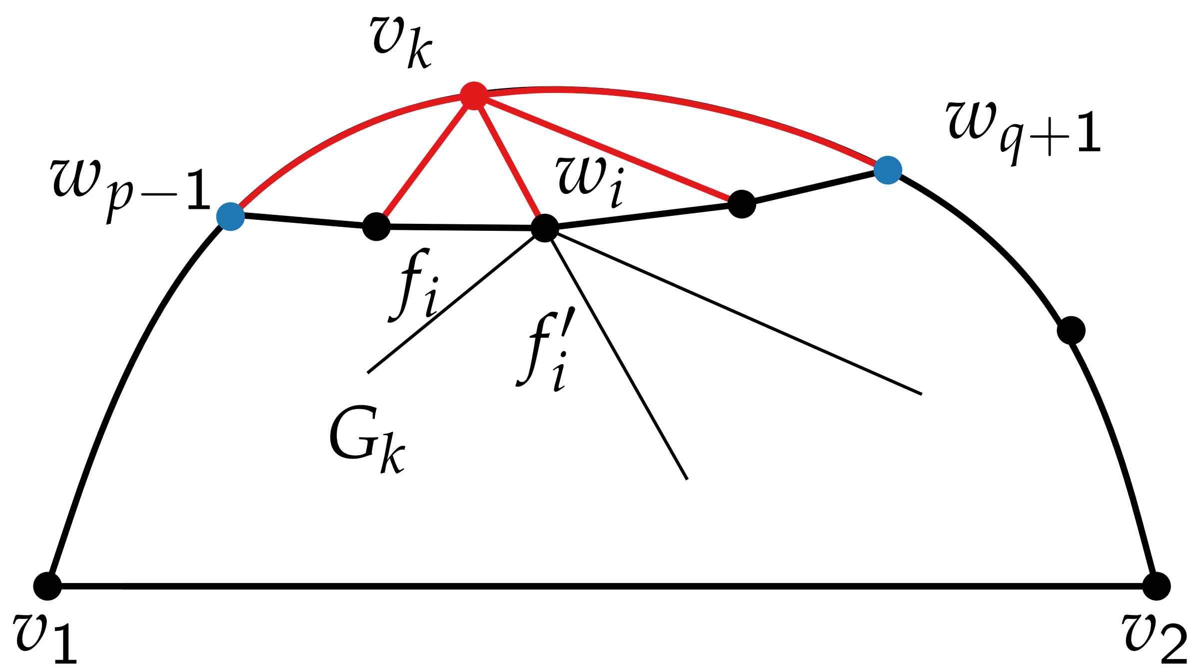
- f_{out} = current outerface
- $F(v)$ = faces that contain v
- $F(e)$ = faces that contain e
- outV(f) = # vertices of f on f_{out}
- outE(f) = # edges of f on f_{out}
- sepF(v) = # separation faces that contain v



Canonical order – implementation

Remove v_k with sepF(v_k)$= 0$

- f_{out} $=$ current outerface
- $F(v)$ $=$ faces that contain v
- $F(e)$ $=$ faces that contain e
- outV(f) $=$ # vertices of f on f_{out}
- outE(f) $=$ # edges of f on f_{out}
- sepF(v) $=$ # separation faces that contain v



- face f_i contains edge (w_{i-1}, w_i) of the outerface of G_{k-1}
- face f_i' contains edges of w_i that are in the interior of G_{k-1}

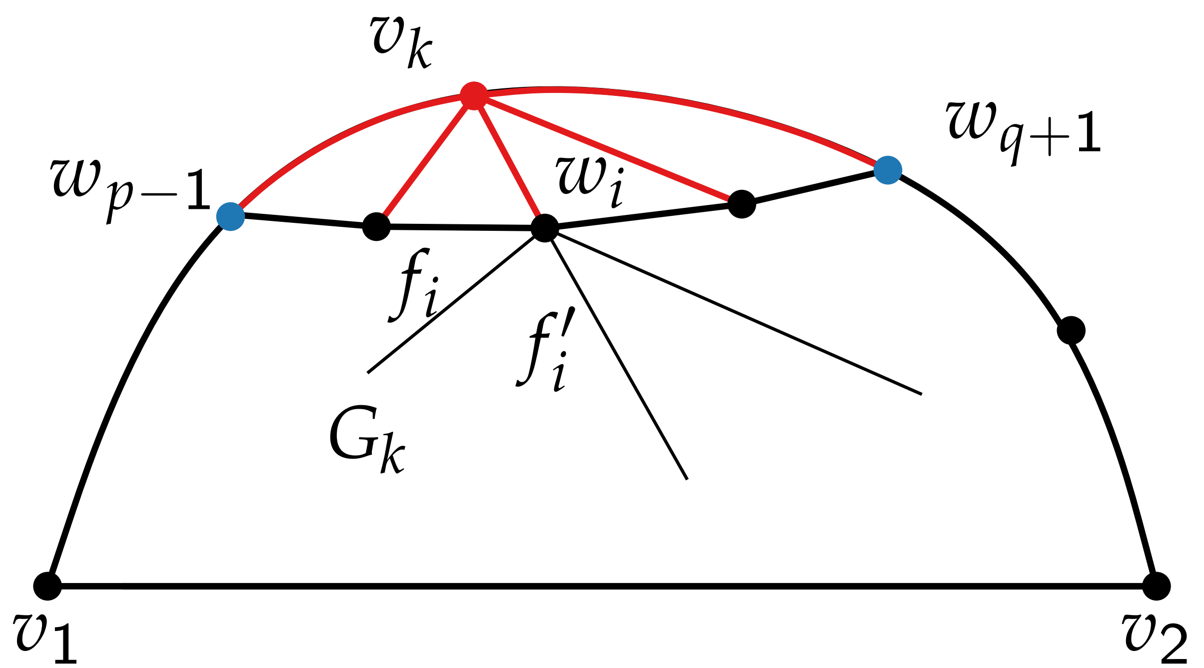
Canonical order – implementation

Remove v_k with sepF(v_k)$= 0$

- v_k and faces that contain v_k are removed
- outV(f_i) increases by two, $p+1 \leq i \leq q$
- outV(f_p), outV(f_{q+1}) increases by one
- outV(f_i') incrases by one, $p \leq i \leq q$
- outE(f_i) increases by one, $p \leq i \leq q+1$

- f_{out} = current outerface
- $F(v)$ = faces that contain v
- $F(e)$ = faces that contain e
- outV(f) = # vertices of f on f_{out}
- outE(f) = # edges of f on f_{out}
- sepF(v) = # separation faces that contain v

- face f_i contains edge (w_{i-1}, w_i) of the outerface of G_{k-1}
- face f_i' contains edges of w_i that are in the interior of G_{k-1}



Canonical order – implementation

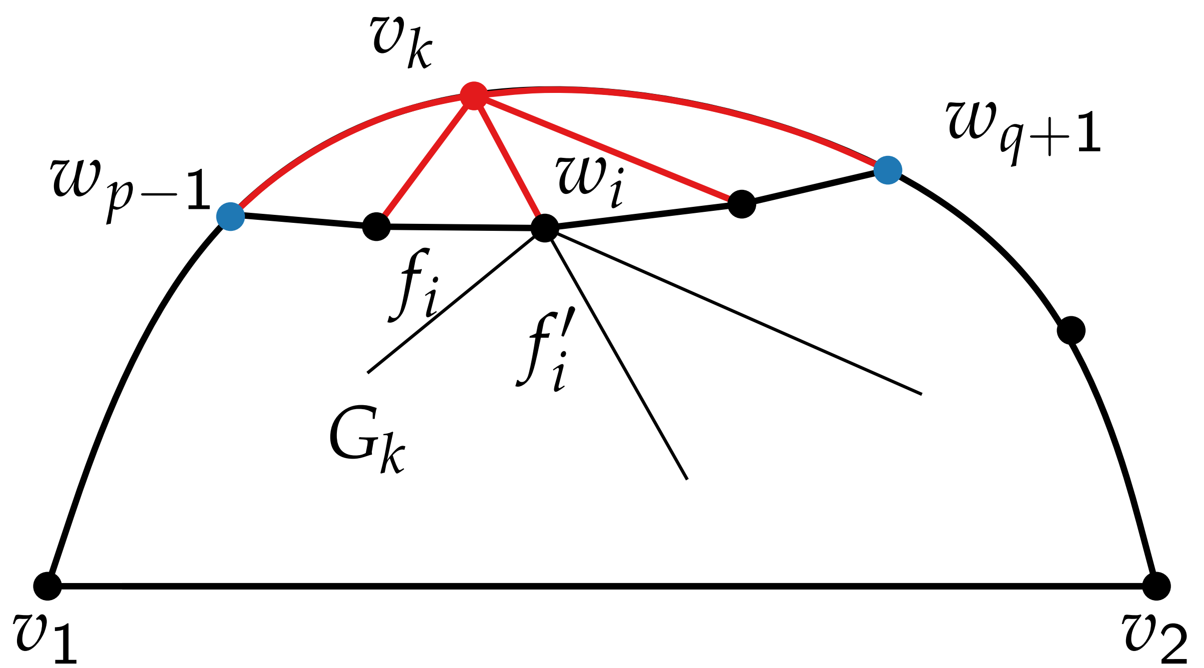
Remove v_k with sepF(v_k)$= 0$

- v_k and faces that contain v_k are removed
- outV(f_i) increases by two, $p+1 \leq i \leq q$
- outV(f_p), outV(f_{q+1}) increases by one
- outV(f_i') incrases by one, $p \leq i \leq q$
- outE(f_i) increases by one, $p \leq i \leq q+1$

- if $f_i$or f_i' becomes separating
  - increase sepF(u) by one for all its vertices u

- face f_i contains edge (w_{i-1}, w_i) of the outerface of G_{k-1}
- face f_i' contains edges of w_i that are in the interior of G_{k-1}

- f_{out} = current outerface
- $F(v)$ = faces that contain v
- $F(e)$ = faces that contain e
- outV(f) = # vertices of f on f_{out}
- outE(f) = # edges of f on f_{out}
- sepF(v) = # separation faces that contain v



Canonical order – implementation

Algorithm CanonicalOrder

initialize;
for $k = n$ **to** 3 **do**
    choose $v_k \neq v_1, v_2$ such that
    – sepf(v)=0 or
    – or $F(v) = \{f\}$, outV(f)=3 and outE(f)=2

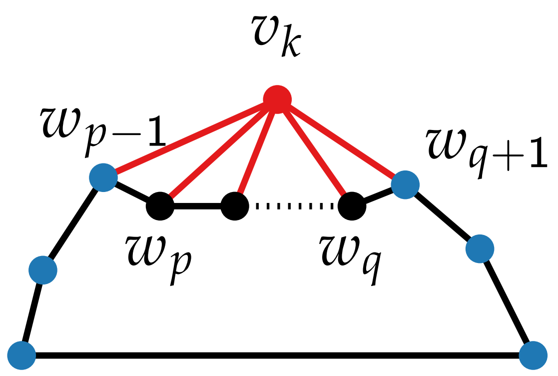
- f_{out} = current outerface
- $F(v)$ = faces that contain v
- $F(e)$ = faces that contain e
- outV(f) = # vertices of f on f_{out}
- outE(f) = # edges of f on f_{out}
- sepF(v) = # separation faces that contain v

Canonical order – implementation

Algorithm CanonicalOrder

initialize;
for $k = n$ **to** 3 **do**
  choose $v_k \neq v_1, v_2$ such that
  – sepf(v)=0 or
  – or $F(v) = \{f\}$, outV(f)=3 and outE(f)=2

- f_{out} = current outerface
- $F(v)$ = faces that contain v
- $F(e)$ = faces that contain e
- outV(f) = # vertices of f on f_{out}
- outE(f) = # edges of f on f_{out}
- sepF(v) = # separation faces that contain v

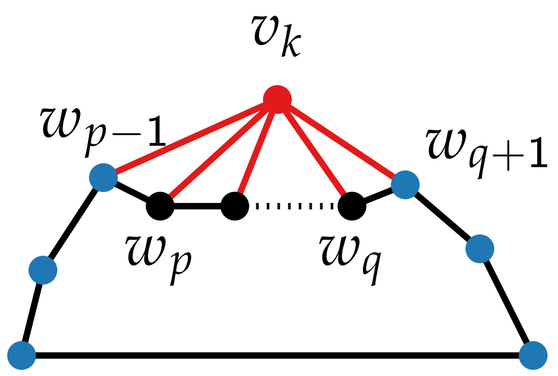


Canonical order – implementation

Algorithm CanonicalOrder

initialize;
for $k = n$ **to** 3 **do**
  choose $v_k \neq v_1, v_2$ such that
  – sepf(v)=0 or
  – or $F(v) = \{f\}$, outV(f)=3 and outE(f)=2
  replace v_k with path $P = w_p \ldots w_q$ in f_{out};

- f_{out} = current outerface
- $F(v)$ = faces that contain v
- $F(e)$ = faces that contain e
- outV(f) = # vertices of f on f_{out}
- outE(f) = # edges of f on f_{out}
- sepF(v) = # separation faces that contain v

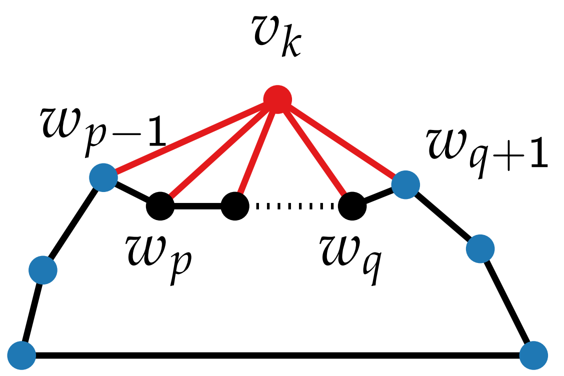


Canonical order – implementation

Algorithm CanonicalOrder

initialize;
for $k = n$ **to** 3 **do**
  choose $v_k \neq v_1, v_2$ such that
  – sepf(v)=0 or
  – or $F(v) = \{f\}$, outV(f)=3 and outE(f)=2
  replace v_k with path $P = w_p \dots w_q$ in f_{out};
  forall $p - 1 \leq i \leq q$ **do**
    remove face $\{v_k, w_i, w_{i+1}\}$ from $F(w_i)$and $F(w_{i+1})$;

- f_{out} = current outerface
- $F(v)$ = faces that contain v
- $F(e)$ = faces that contain e
- outV(f) = # vertices of f on f_{out}
- outE(f) = # edges of f on f_{out}
- sepF(v) = # separation faces that contain v

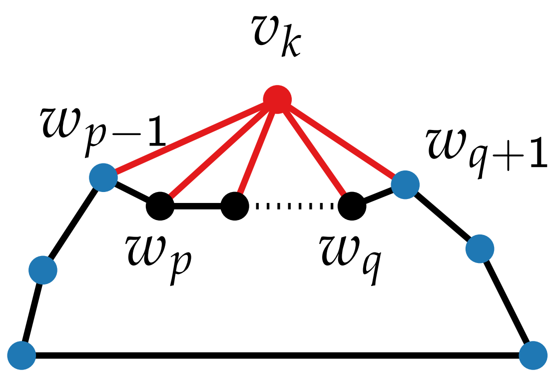


Canonical order – implementation

Algorithm CanonicalOrder

initialize;
for $k = n$ **to** 3 **do**
    choose $v_k \neq v_1, v_2$ such that
    – sepf(v)=0 or
    – or $F(v) = \{f\}$, outV(f)=3 and outE(f)=2
    replace v_k with path $P = w_p \dots w_q$ in f_{out};
    forall $p - 1 \leq i \leq q$ **do**
        remove face $\{v_k, w_i, w_{i+1}\}$ from $F(w_i)$and $F(w_{i+1})$;
    forall $w \in w_{p-1} P w_{q+1}$ **do**
        forall $f \in F(w)$ **do**
            update outV(f);
    forall $e \in w_{p-1} P w_{q+1}$ **do**
        forall $f \in F(e)$ **do**
            update outE(f);

- f_{out} = current outerface
- $F(v)$ = faces that contain v
- $F(e)$ = faces that contain e
- outV(f) = # vertices of f on f_{out}
- outE(f) = # edges of f on f_{out}
- sepF(v) = # separation faces that contain v

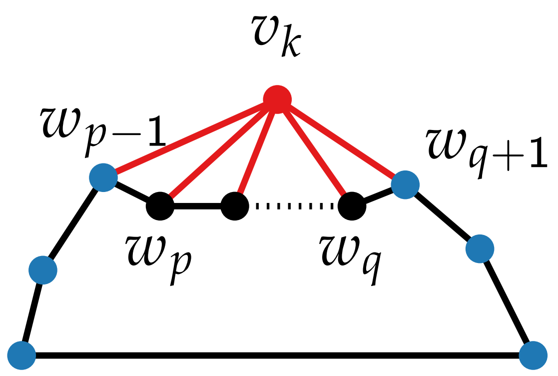


Canonical order – implementation

Algorithm CanonicalOrder

initialize;
for $k = n$ **to** 3 **do**
- choose $v_k \neq v_1, v_2$ such that
  - – sepf(v)=0 or
  - – or $F(v) = \{f\}$, outV(f)=3 and outE(f)=2
- replace v_k with path $P = w_p \dots w_q$ in f_{out};
- **forall** $p - 1 \leq i \leq q$ **do**
  - remove face $\{v_k, w_i, w_{i+1}\}$ from $F(w_i)$and $F(w_{i+1})$;
- **forall** $w \in w_{p-1}Pw_{q+1}$ **do**
  - **forall** $f \in F(w)$ **do**
    - update outV(f);
- **forall** $e \in w_{p-1}Pw_{q+1}$ **do**
  - **forall** $f \in F(e)$ **do**
    - update outE(f);
- **forall** $w \in P \cup N(P)$ **do**
  - **forall** $f \in F(w)$ **do**
    - update sepF(w);

- f_{out} = current outerface
- $F(v)$ = faces that contain v
- $F(e)$ = faces that contain e
- outV(f) = # vertices of f on f_{out}
- outE(f) = # edges of f on f_{out}
- sepF(v) = # separation faces that contain v



Canonical order – implementation

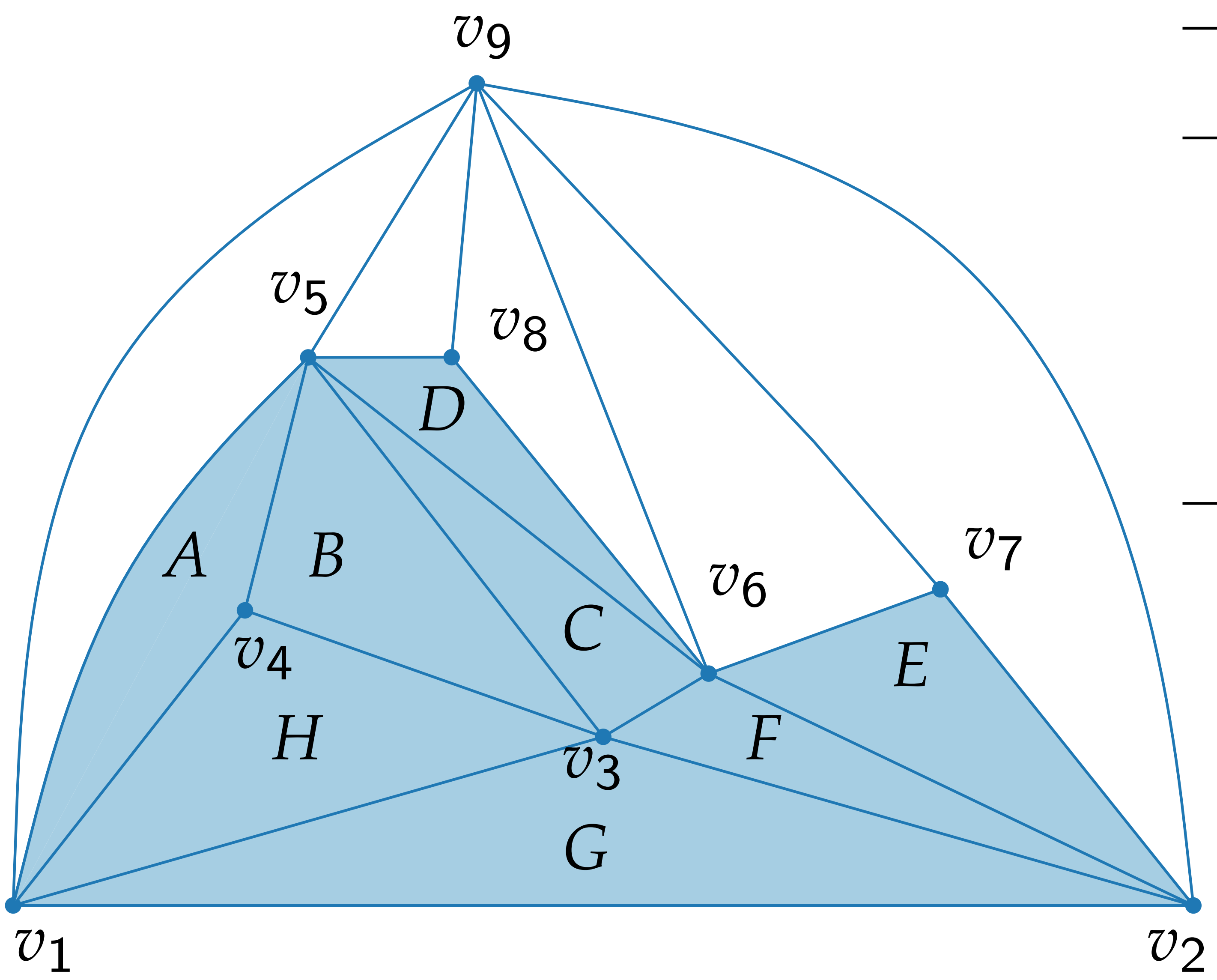
Algorithm CanonicalOrder

initialize;
for $k = n$ **to** 3 **do**
- choose $v_k \neq v_1, v_2$ such that
  - – sepf(v)=0 or
  - – or $F(v) = \{f\}$, outV(f)=3 and outE(f)=2
- replace v_k with path $P = w_p \dots w_q$ in f_{out};
- **forall** $p - 1 \leq i \leq q$ **do**
  - remove face $\{v_k, w_i, w_{i+1}\}$ from $F(w_i)$and $F(w_{i+1})$;
- **forall** $w \in w_{p-1} P w_{q+1}$ **do**
  - **forall** $f \in F(w)$ **do**
    - update outV(f);
- **forall** $e \in w_{p-1} P w_{q+1}$ **do**
  - **forall** $f \in F(e)$ **do**
    - update outE(f);
- **forall** $w \in P \cup N(P)$ **do**
  - **forall** $f \in F(w)$ **do**
    - update sepF(w);

- f_{out} = current outerface
- $F(v)$ = faces that contain v
- $F(e)$ = faces that contain e
- outV(f) = # vertices of f on f_{out}
- outE(f) = # edges of f on f_{out}
- sepF(v) = # separation faces that contain v

Lemma.
Algorithm CanonicalOrder computes a canonical order of a plane graph in $\mathcal{O}(n)$ time.

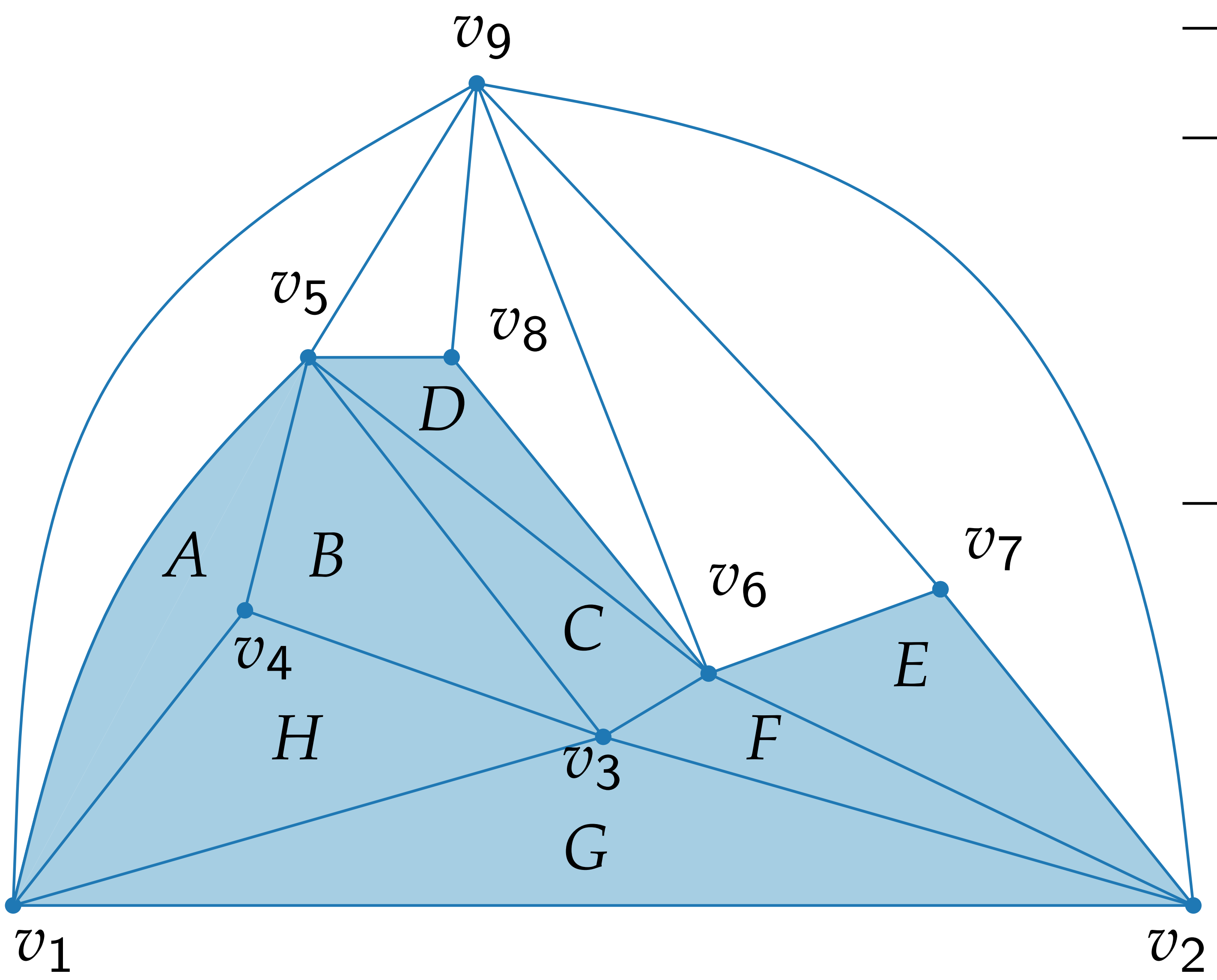
Canonical order – example



	A	B	C	D	E	F	G	H
outV(f)								
outE(f)								

	v_3	v_4	v_5	v_6	v_7	v_8
sepF(v)						

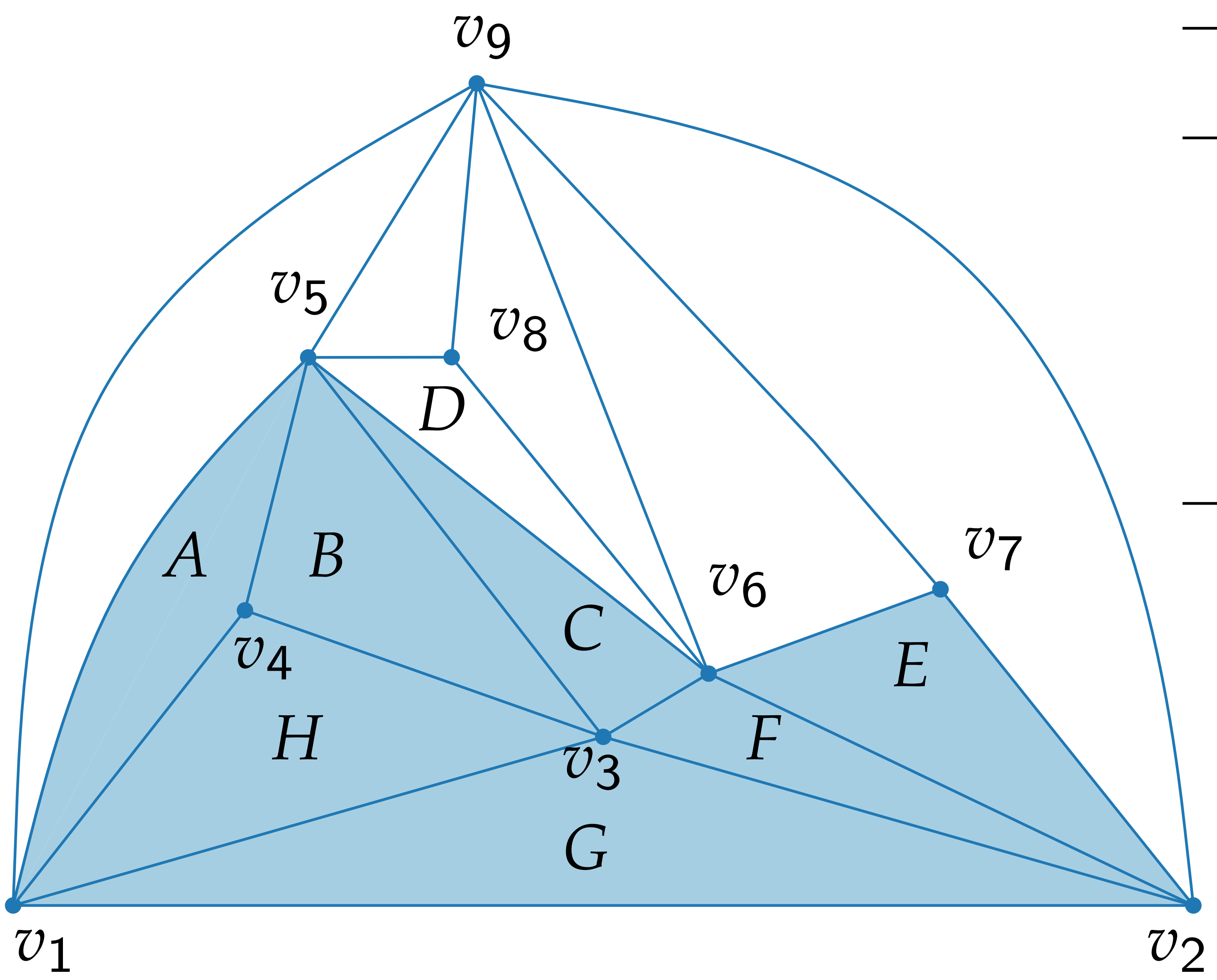
Canonical order – example



	A	B	C	D	E	F	G	H
outV(f)	2	1	2	3	3	2	2	1
outE(f)	1	0	0	2	2	0	1	0

	v_3	v_4	v_5	v_6	v_7	v_8
sepF(v)			2	4	1	1

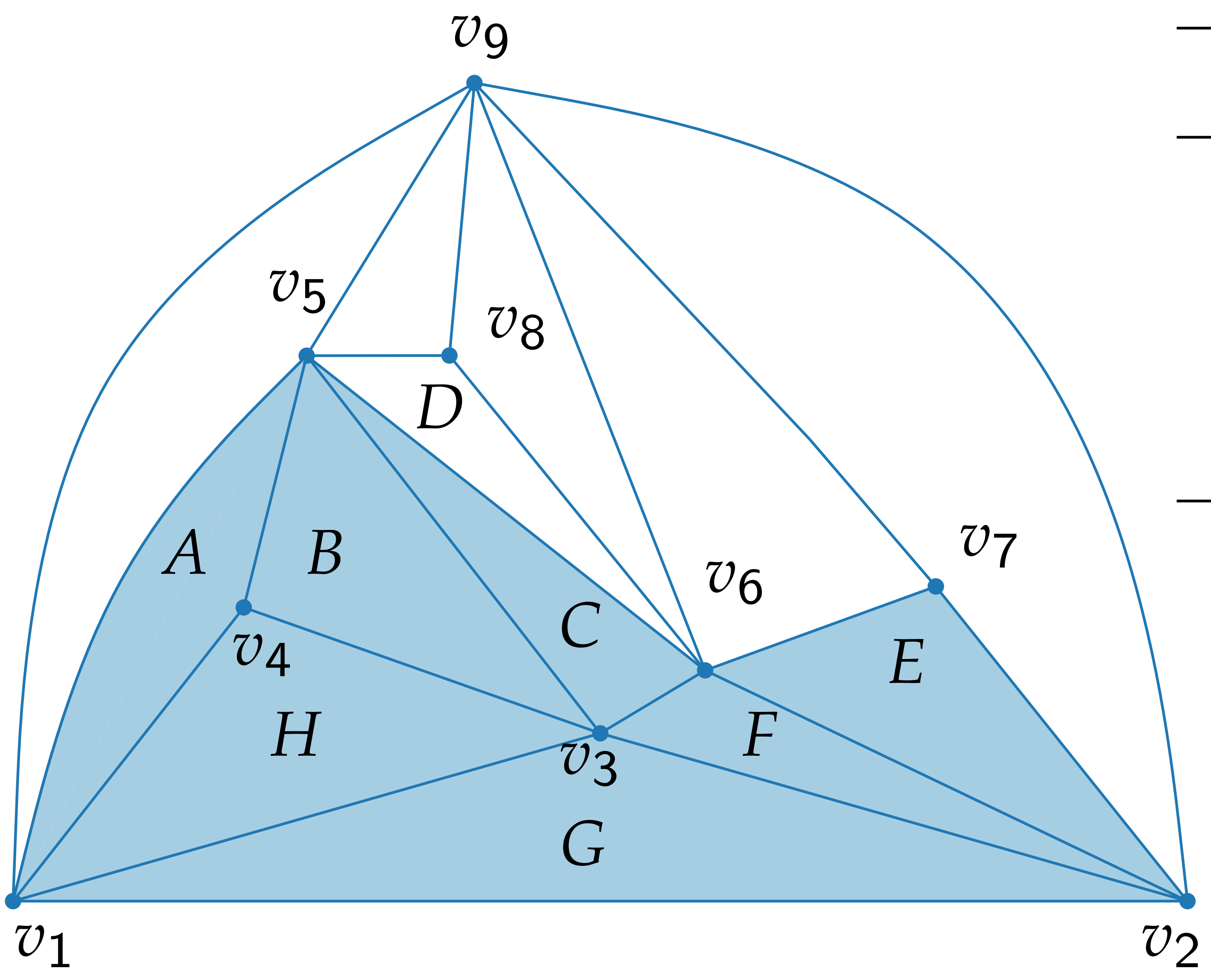
Canonical order – example



	A	B	C	D	E	F	G	H
outV(f)	2	1	2	3	3	2	2	1
outE(f)	1	0	0	2	2	0	1	0

	v_3	v_4	v_5	v_6	v_7	v_8
sepF(v)			2	4	1	1

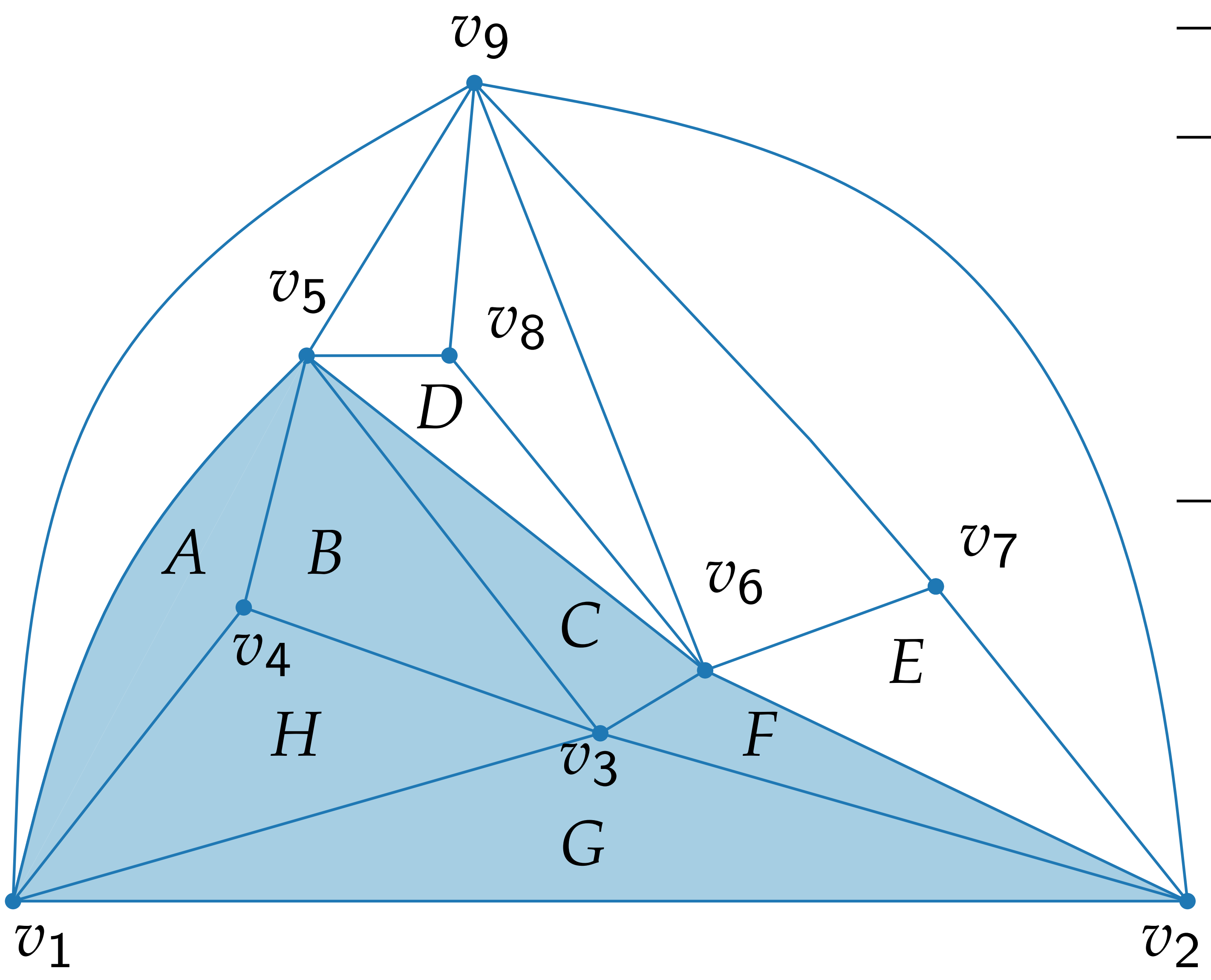
Canonical order – example



	A	B	C	D	E	F	G	H
outV(f)	2	1	2		3	2	2	1
outE(f)	1	0	1		2	0	1	0

	v_3	v_4	v_5	v_6	v_7	v_8
sepF(v)			0	2	1	

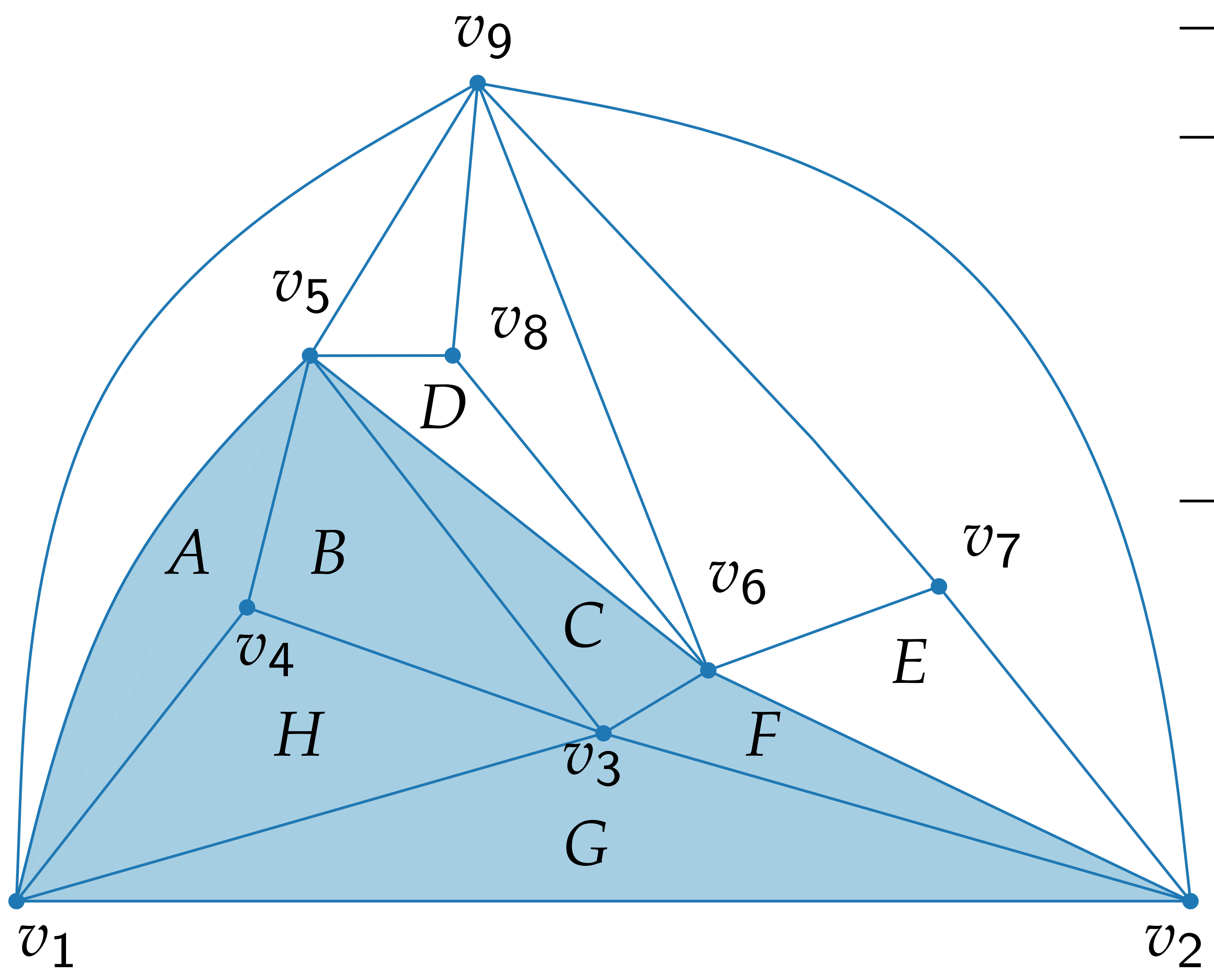
Canonical order – example



	A	B	C	D	E	F	G	H
outV(f)	2	1	2		3	2	2	1
outE(f)	1	0	1		2	0	1	0

	v_3	v_4	v_5	v_6	v_7	v_8
sepF(v)			0	2	1	

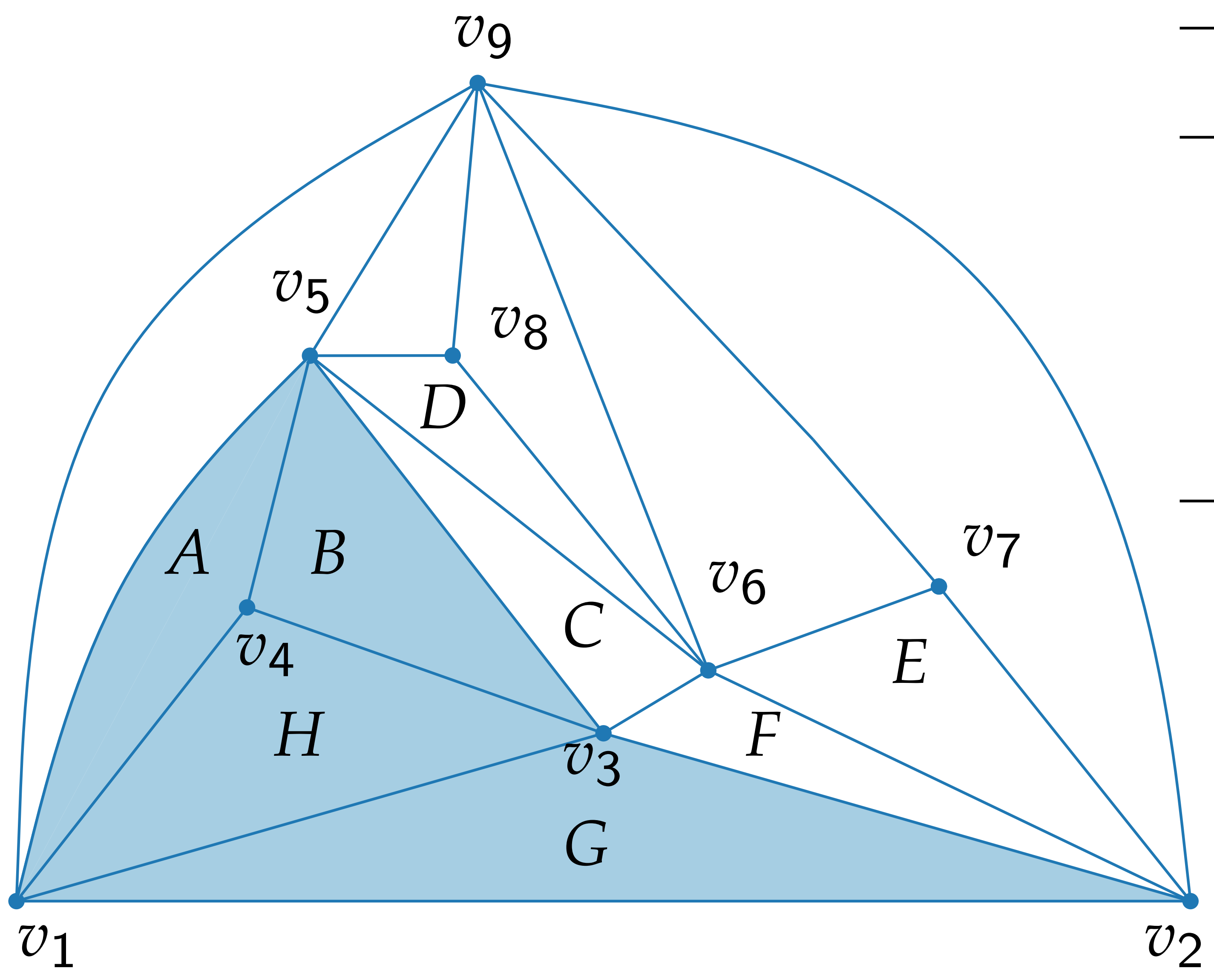
Canonical order – example



	A	B	C	D	E	F	G	H
outV(f)	2	1	2			2	2	1
outE(f)	1	0	1			1	1	0

	v_3	v_4	v_5	v_6	v_7	v_8
sepF(v)			0	0		

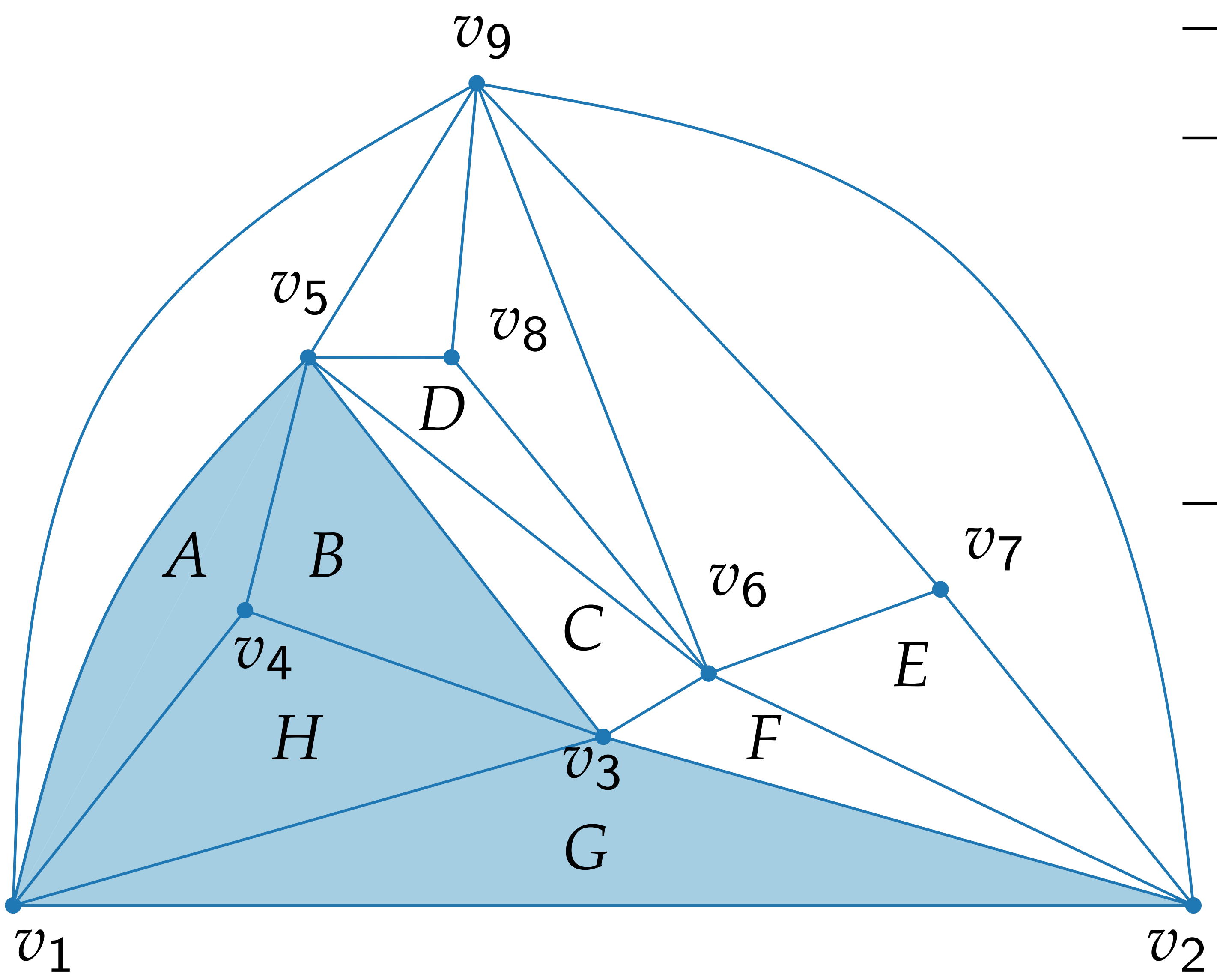
Canonical order – example



	A	B	C	D	E	F	G	H
outV(f)	2	1	2			2	2	1
outE(f)	1	0	1			1	1	0

	v_3	v_4	v_5	v_6	v_7	v_8
sepF(v)			0	0		

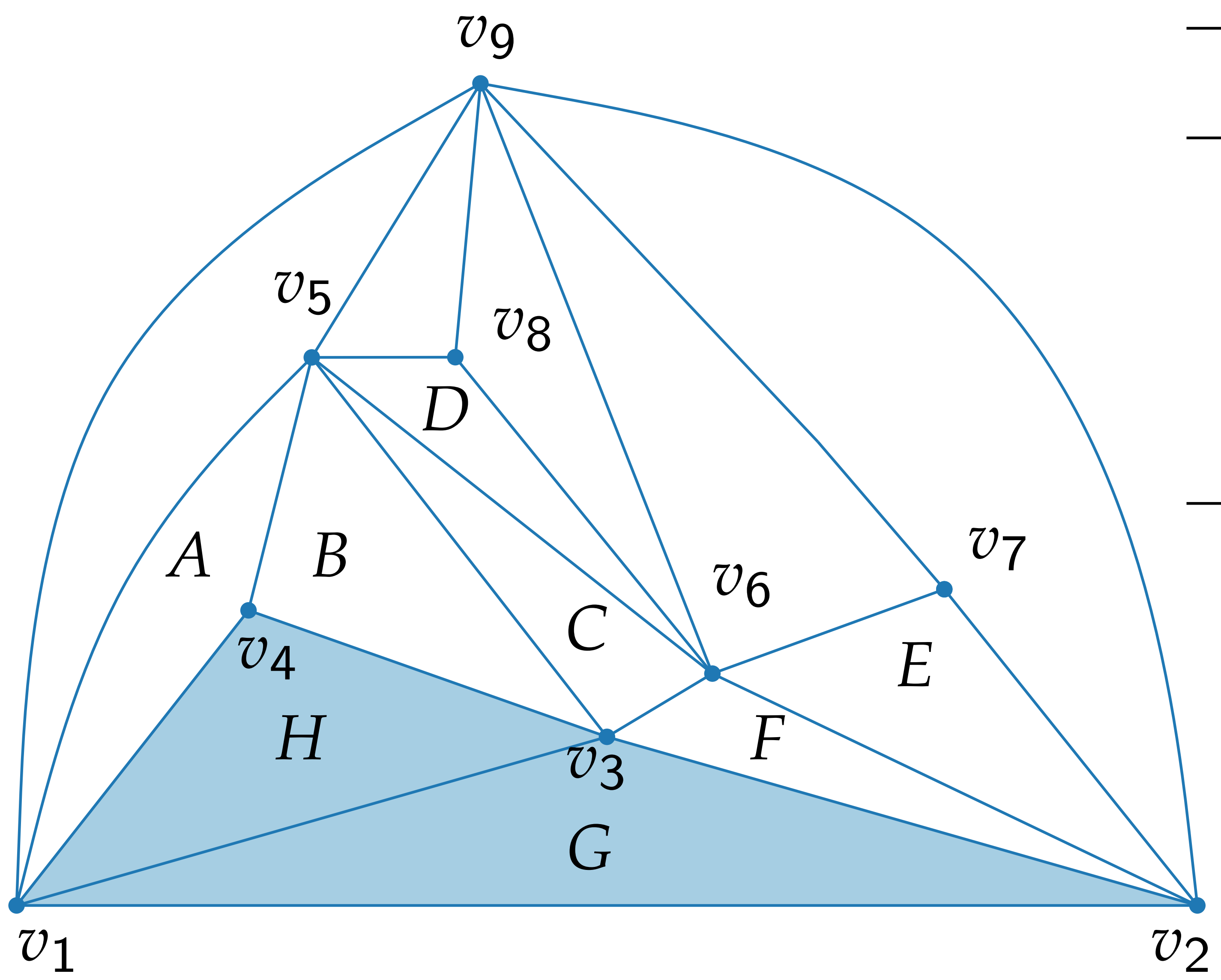
Canonical order – example



	A	B	C	D	E	F	G	H
outV(f)	2	2					3	2
outE(f)	1	1					2	0

	v_3	v_4	v_5	v_6	v_7	v_8
sepF(v)	2		0			

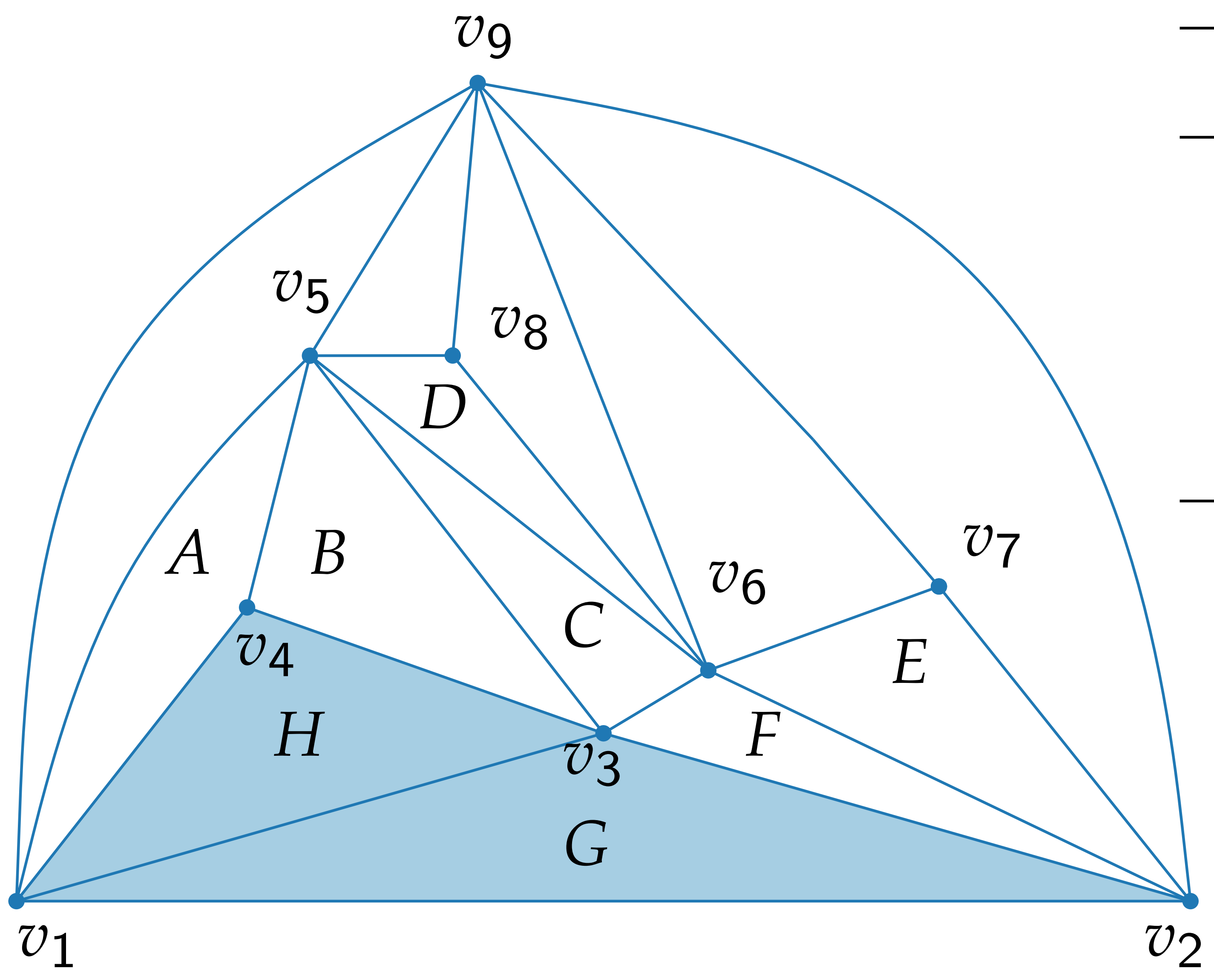
Canonical order – example



	A	B	C	D	E	F	G	H
outV(f)	2	2					3	2
outE(f)	1	1					2	0

	v_3	v_4	v_5	v_6	v_7	v_8
sepF(v)	2		0			

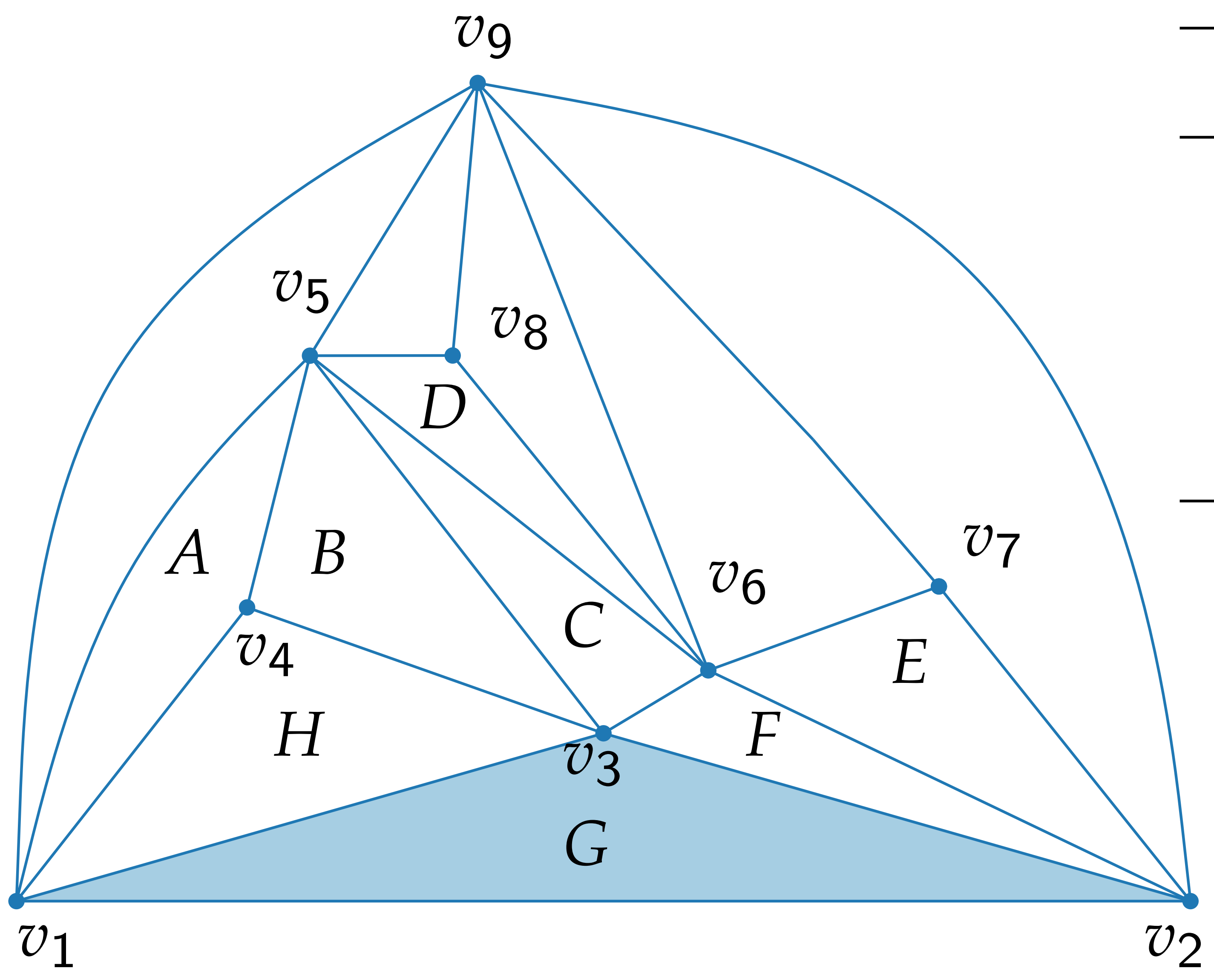
Canonical order – example



	A	B	C	D	E	F	G	H
outV(f)							3	3
outE(f)							2	2

	v_3	v_4	v_5	v_6	v_7	v_8
sepF(v)	2	1				

Canonical order – example



	A	B	C	D	E	F	G	H
outV(f)							3	3
outE(f)							2	2

	v_3	v_4	v_5	v_6	v_7	v_8
sepF(v)	2	1				

Order:

$\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9\}$

Literature

- [HGD Ch. 6.5] canonical order
- [dFPP90] de Fraysseix, Pach, Pollack *"How to draw a planar graph on a grid"*, Combinatorica, 1990
- [Kant96] Kant *"Drawing planar graphs using the canonical ordering"*, Algorithmica, 1996
- [BBC11] Badent, Brandes, Cornelsen *"More Canonical Ordering"*, JGAA, 2011