

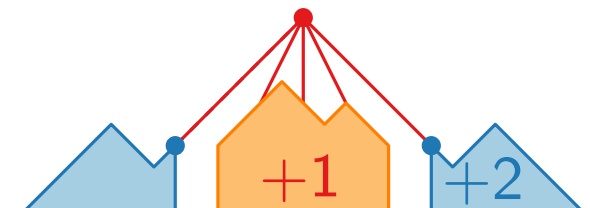
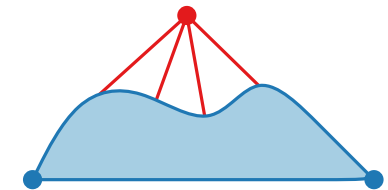
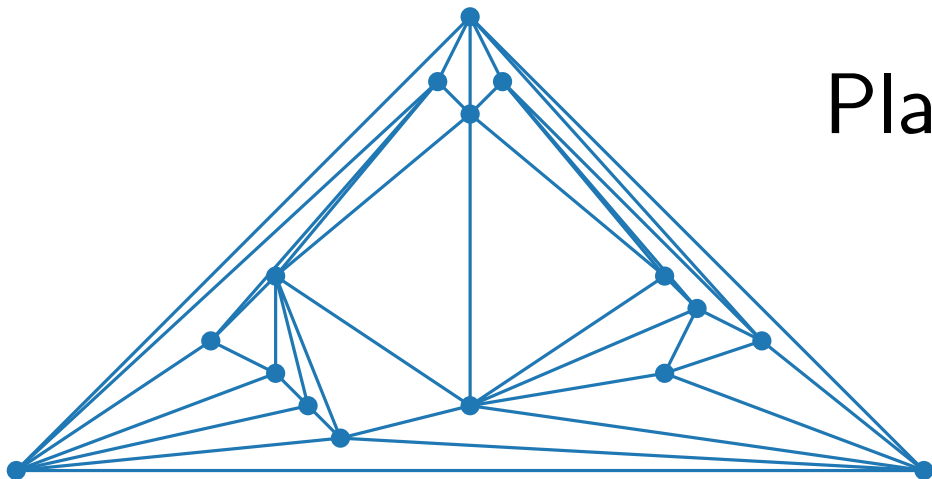
# Visualization of Graphs

## Lecture 3:

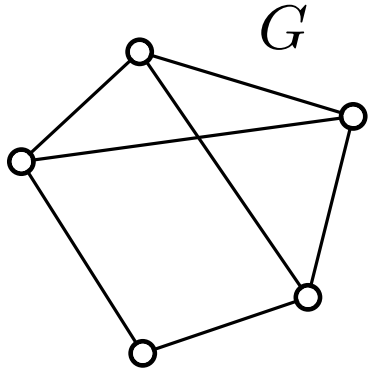
### Straight-Line Drawings of Planar Graphs I: Canonical Ordering and the Shift Method

#### Part I: Planar Straight-Line Drawings

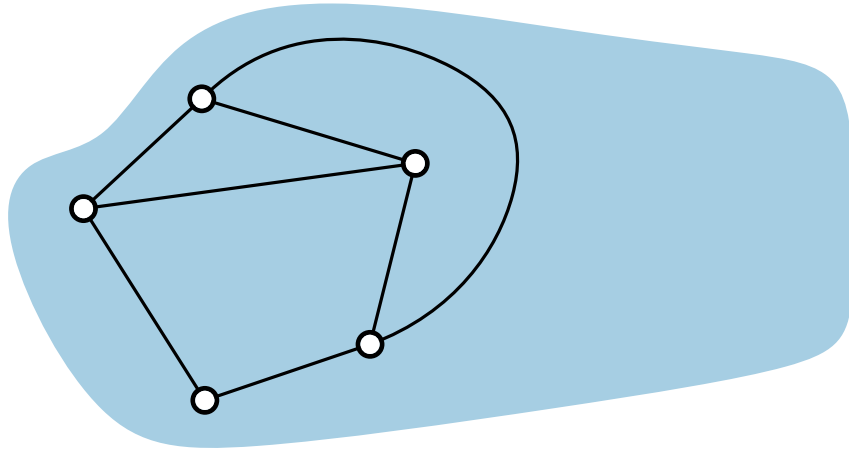
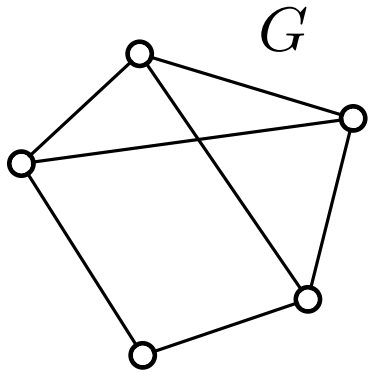
Alexander Wolff



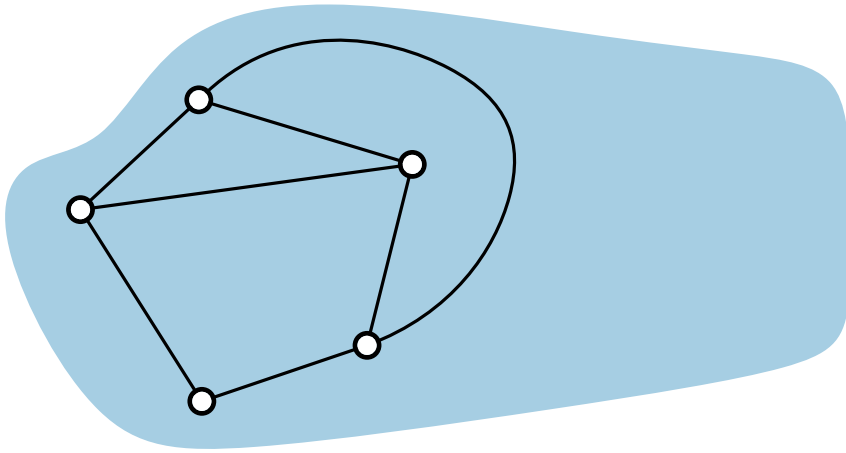
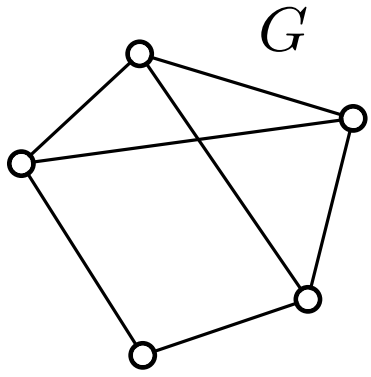
# Planar Graphs



# Planar Graphs



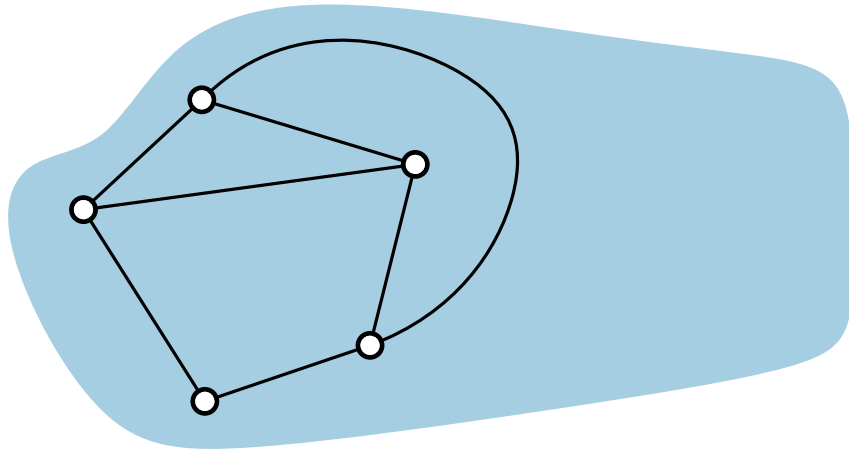
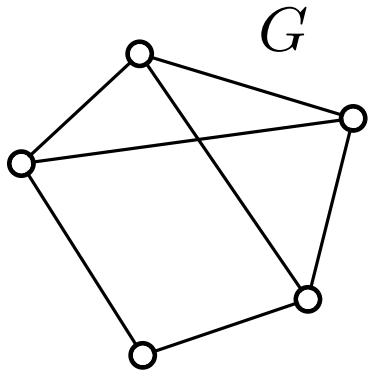
# Planar Graphs



$G$  is **planar**:

it can be drawn in such a way that  
no edges cross each other.

# Planar Graphs



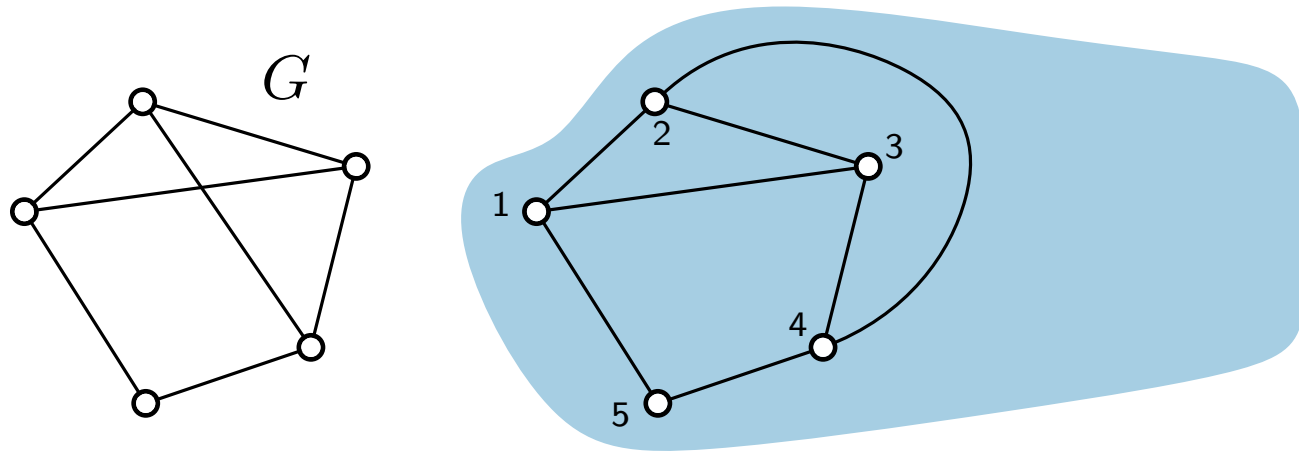
$G$  is **planar**:

it can be drawn in such a way that no edges cross each other.

**planar embedding:**

Clockwise orientation of adjacent vertices around each vertex.

# Planar Graphs



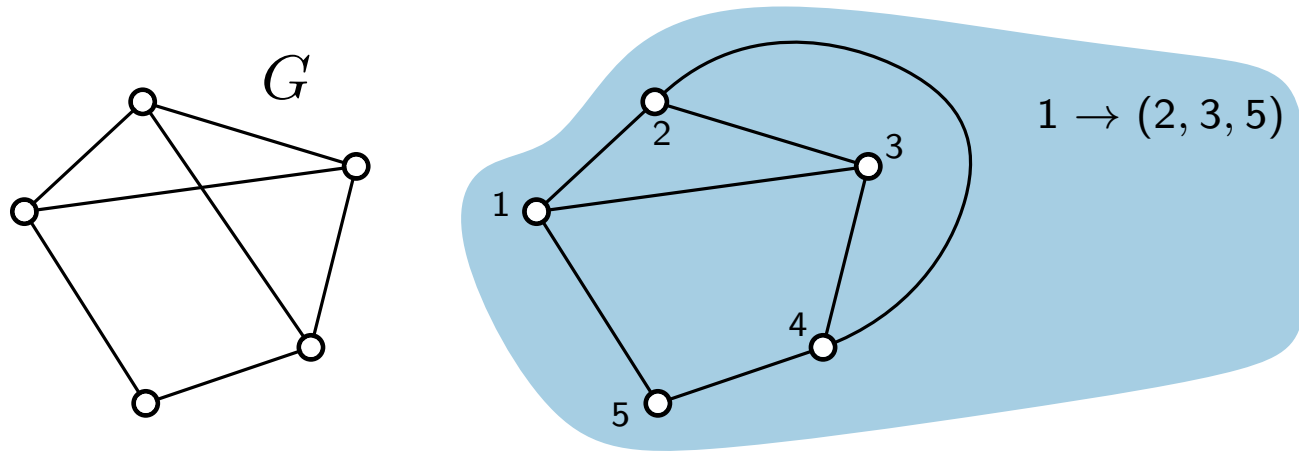
$G$  is **planar**:

it can be drawn in such a way that  
no edges cross each other.

**planar embedding:**

Clockwise orientation of adjacent  
vertices around each vertex.

# Planar Graphs



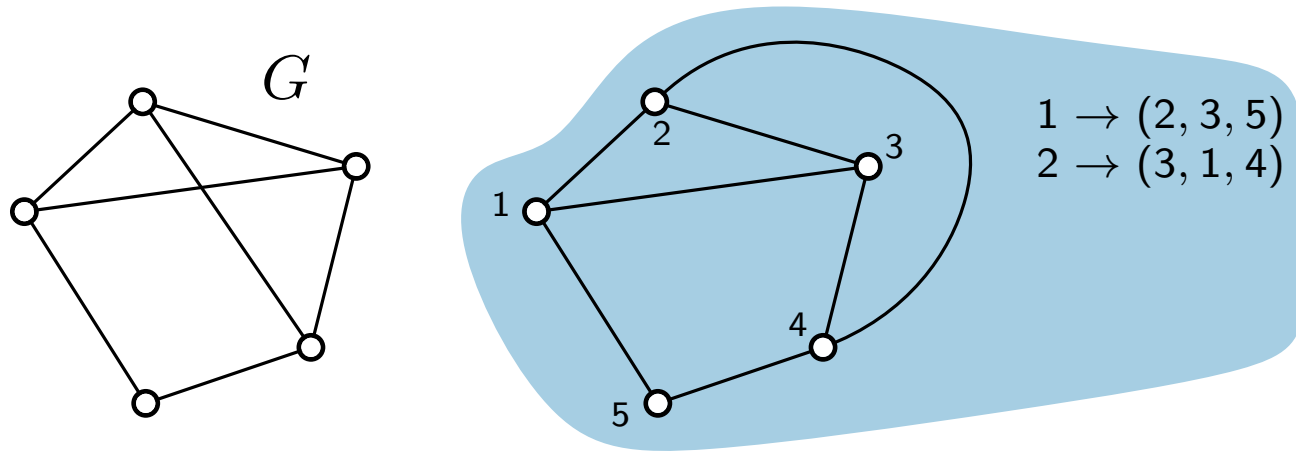
$G$  is **planar**:

it can be drawn in such a way that no edges cross each other.

**planar embedding:**

Clockwise orientation of adjacent vertices around each vertex.

# Planar Graphs



$G$  is **planar**:

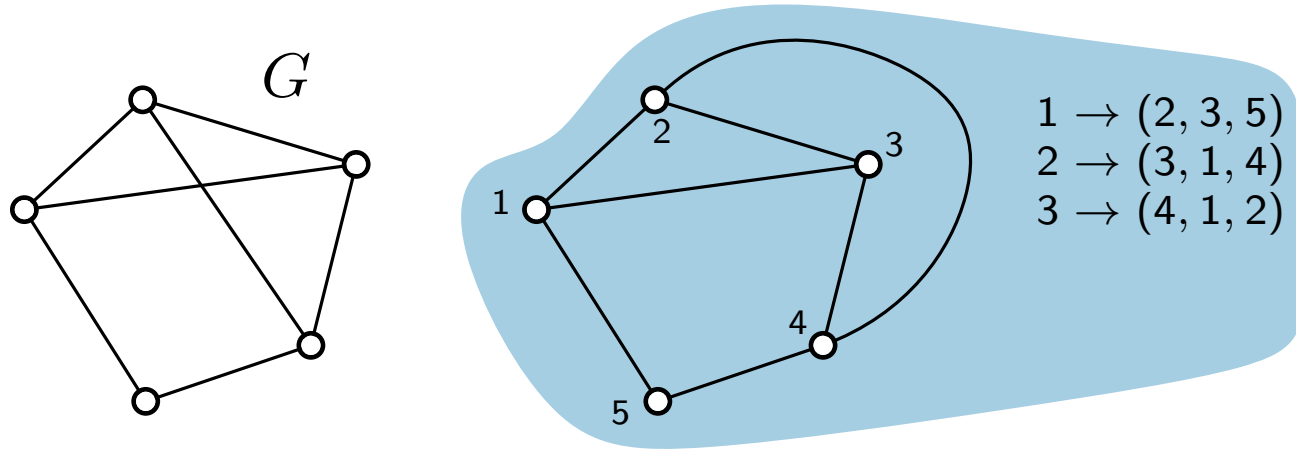
it can be drawn in such a way that no edges cross each other.

**planar embedding:**

Clockwise orientation of adjacent vertices around each vertex.



# Planar Graphs



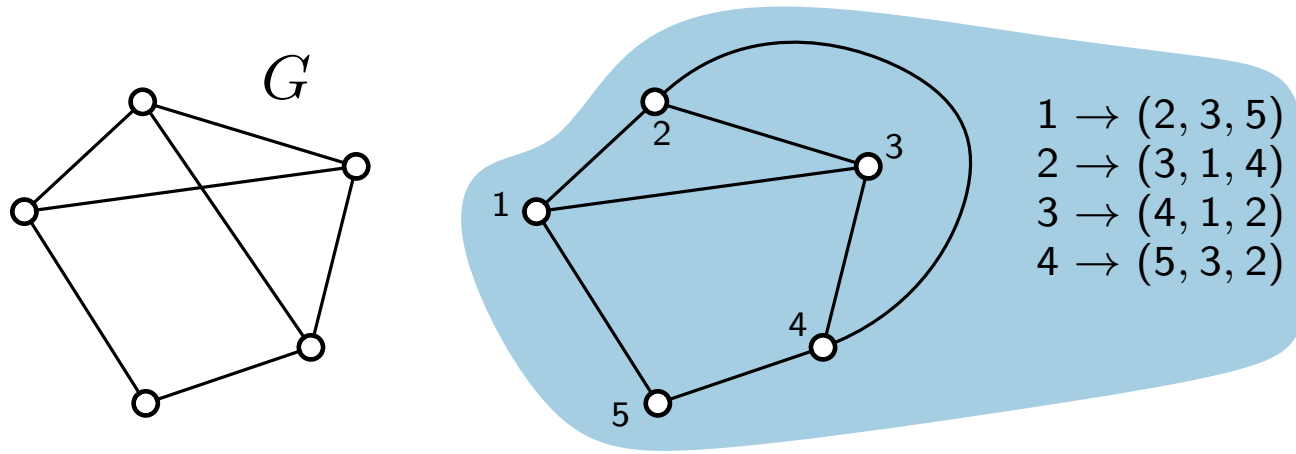
$G$  is **planar**:

it can be drawn in such a way that no edges cross each other.

**planar embedding:**

Clockwise orientation of adjacent vertices around each vertex.

# Planar Graphs



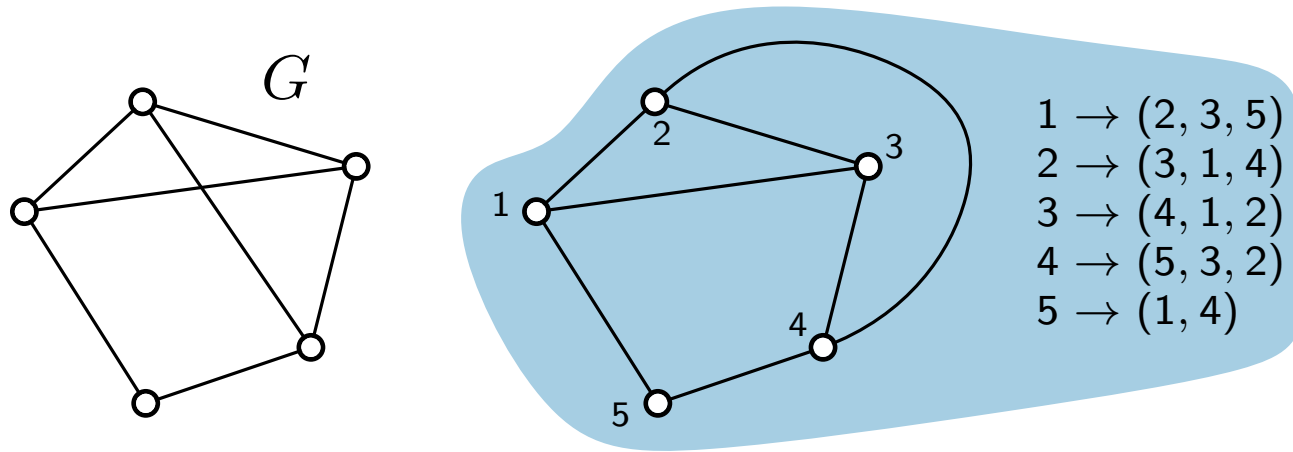
$G$  is **planar**:

it can be drawn in such a way that no edges cross each other.

**planar embedding:**

Clockwise orientation of adjacent vertices around each vertex.

# Planar Graphs



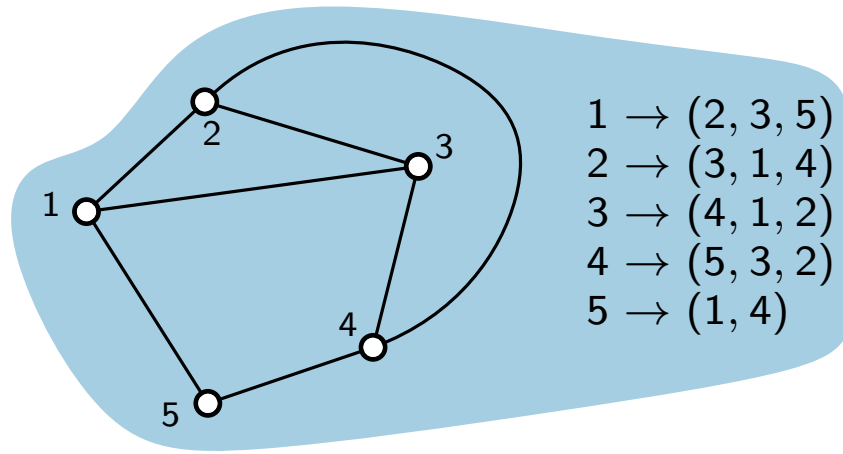
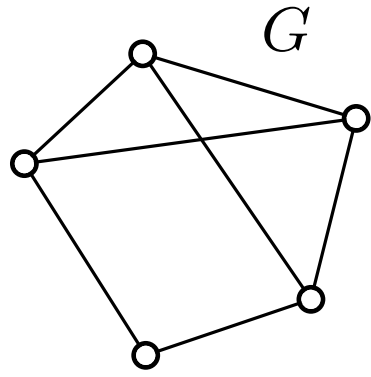
$G$  is **planar**:

it can be drawn in such a way that no edges cross each other.

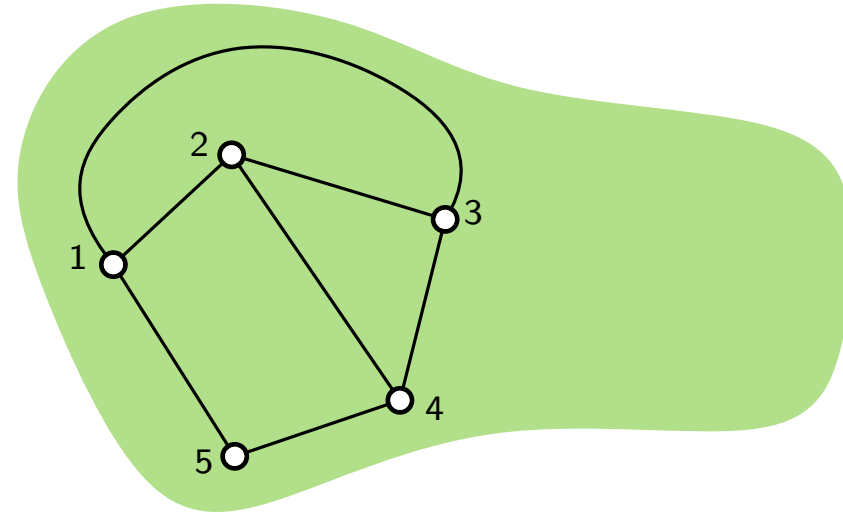
**planar embedding:**

Clockwise orientation of adjacent vertices around each vertex.

# Planar Graphs



$1 \rightarrow (2, 3, 5)$   
 $2 \rightarrow (3, 1, 4)$   
 $3 \rightarrow (4, 1, 2)$   
 $4 \rightarrow (5, 3, 2)$   
 $5 \rightarrow (1, 4)$



$G$  is **planar**:

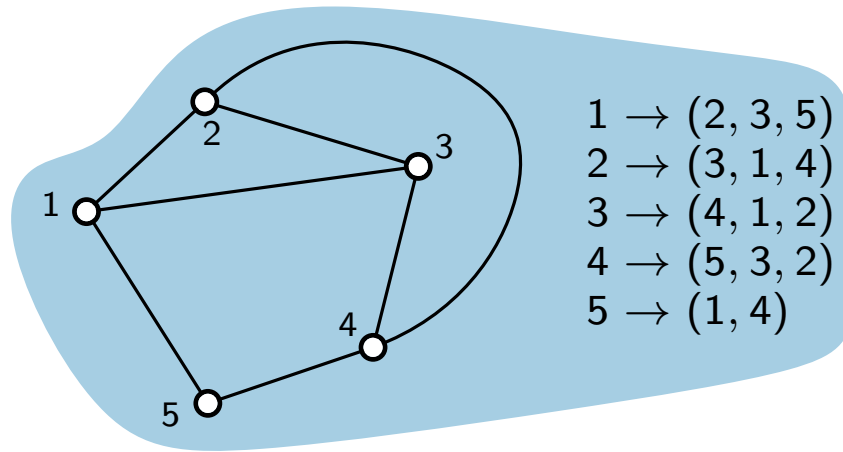
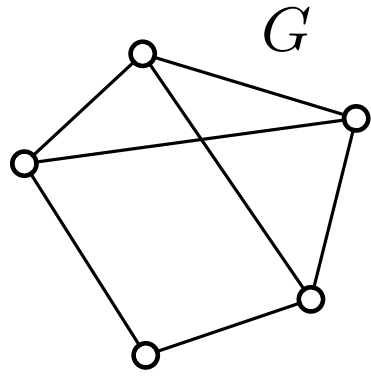
it can be drawn in such a way that no edges cross each other.

**planar embedding:**

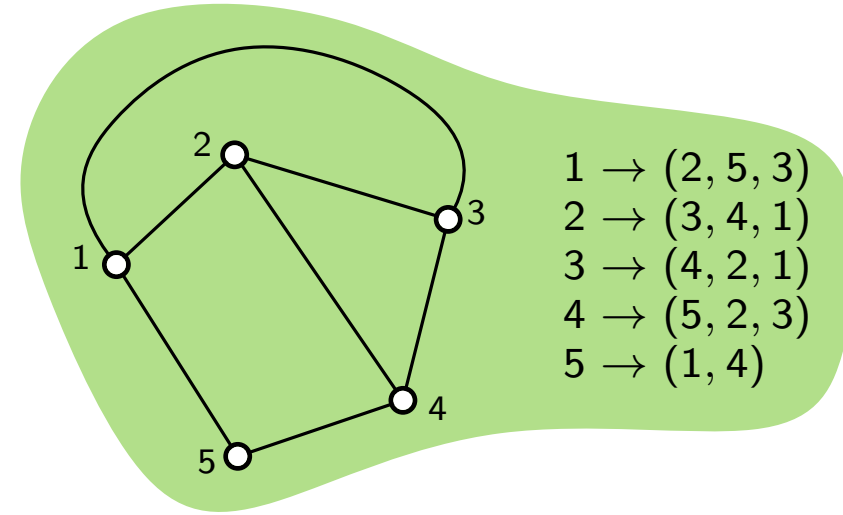
Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

# Planar Graphs



$1 \rightarrow (2, 3, 5)$   
 $2 \rightarrow (3, 1, 4)$   
 $3 \rightarrow (4, 1, 2)$   
 $4 \rightarrow (5, 3, 2)$   
 $5 \rightarrow (1, 4)$



$1 \rightarrow (2, 5, 3)$   
 $2 \rightarrow (3, 4, 1)$   
 $3 \rightarrow (4, 2, 1)$   
 $4 \rightarrow (5, 2, 3)$   
 $5 \rightarrow (1, 4)$

$G$  is **planar**:

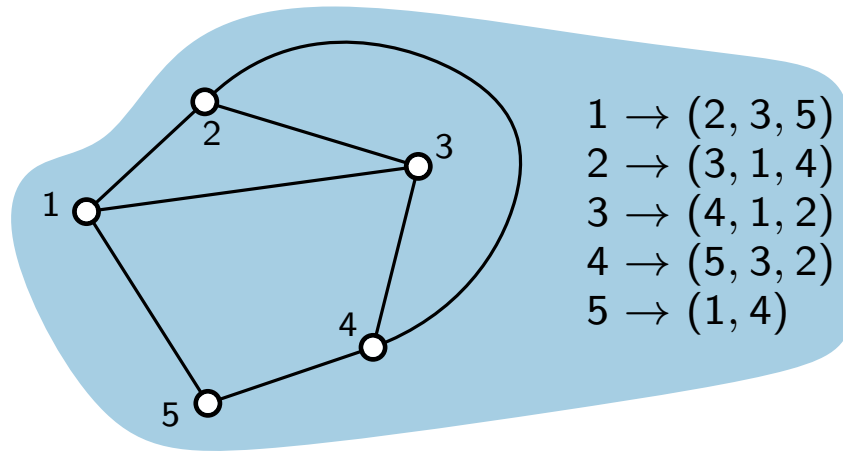
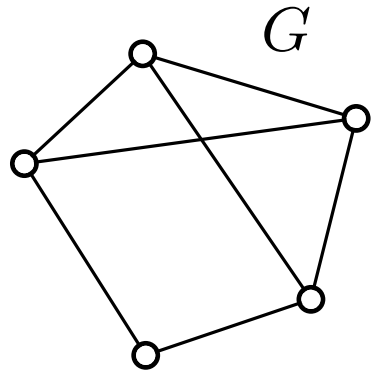
it can be drawn in such a way that no edges cross each other.

**planar embedding:**

Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

# Planar Graphs



$1 \rightarrow (2, 3, 5)$   
 $2 \rightarrow (3, 1, 4)$   
 $3 \rightarrow (4, 1, 2)$   
 $4 \rightarrow (5, 3, 2)$   
 $5 \rightarrow (1, 4)$

$G$  is **planar**:

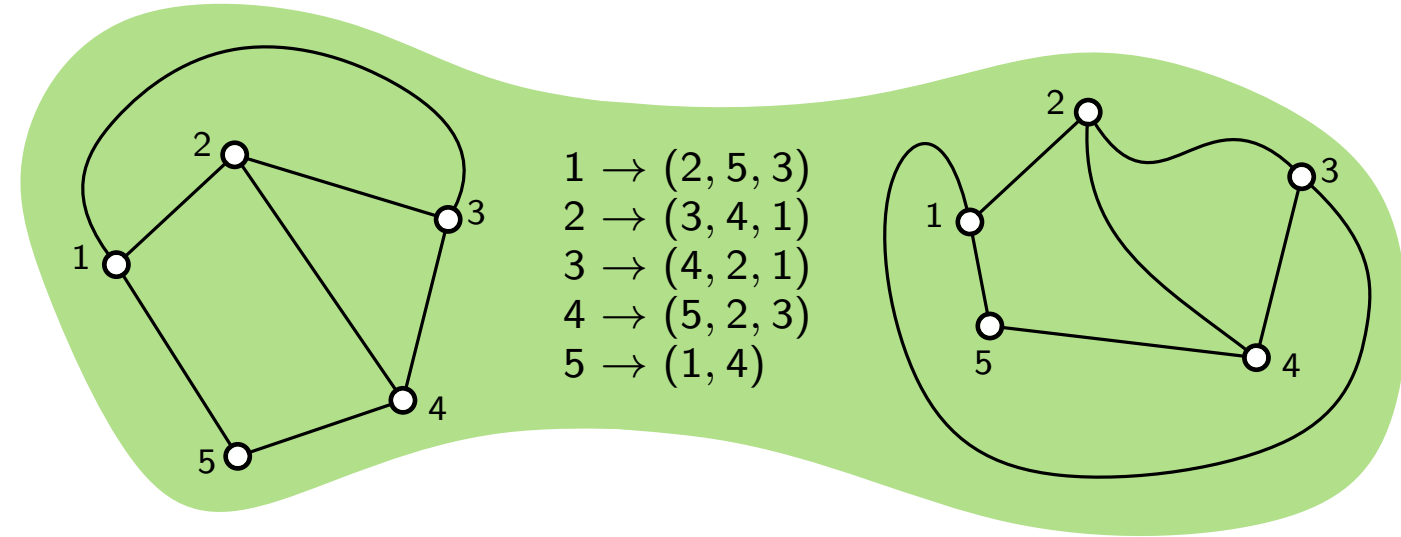
it can be drawn in such a way that no edges cross each other.

**planar embedding:**

Clockwise orientation of adjacent vertices around each vertex.

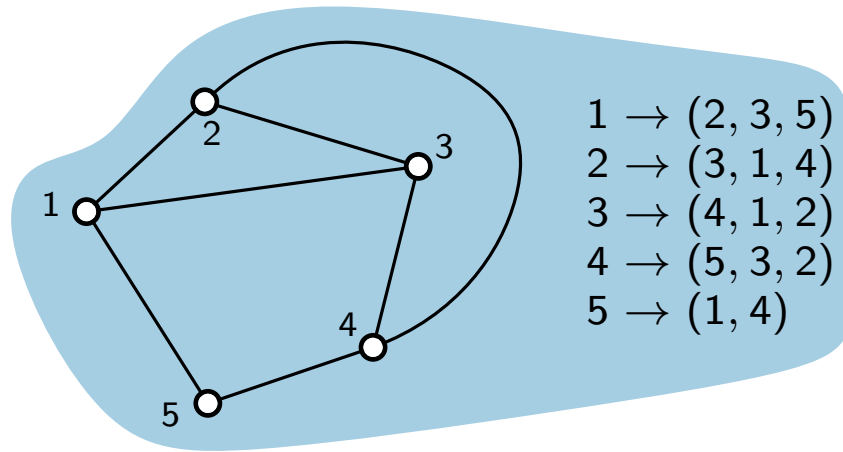
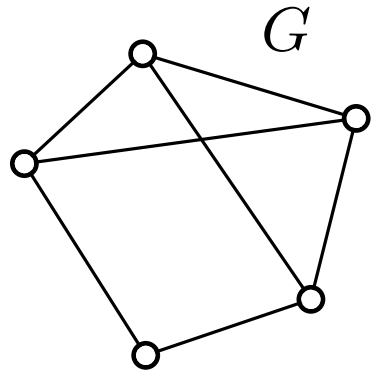
A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



$1 \rightarrow (2, 5, 3)$   
 $2 \rightarrow (3, 4, 1)$   
 $3 \rightarrow (4, 2, 1)$   
 $4 \rightarrow (5, 2, 3)$   
 $5 \rightarrow (1, 4)$

# Planar Graphs



$G$  is **planar**:

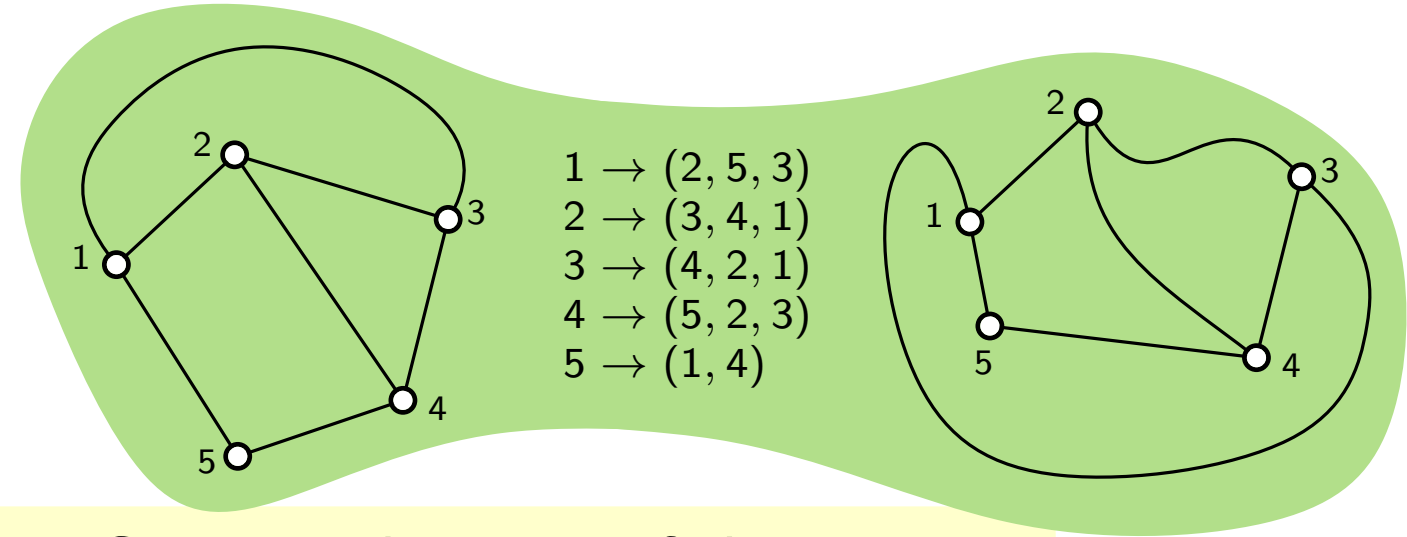
it can be drawn in such a way that no edges cross each other.

**planar embedding**:

Clockwise orientation of adjacent vertices around each vertex.

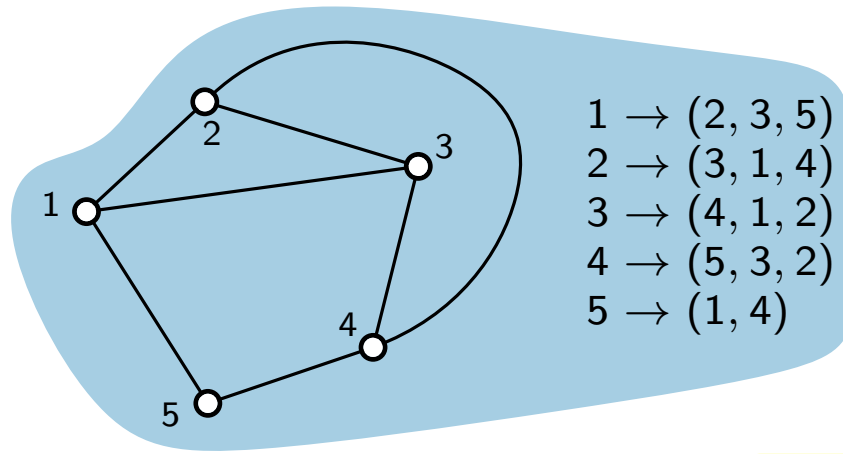
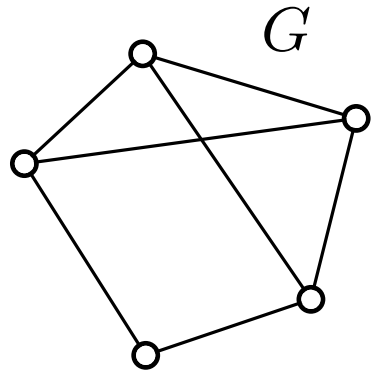
A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



**faces**: Connected region of the plane bounded by edges

# Planar Graphs



$G$  is **planar**:

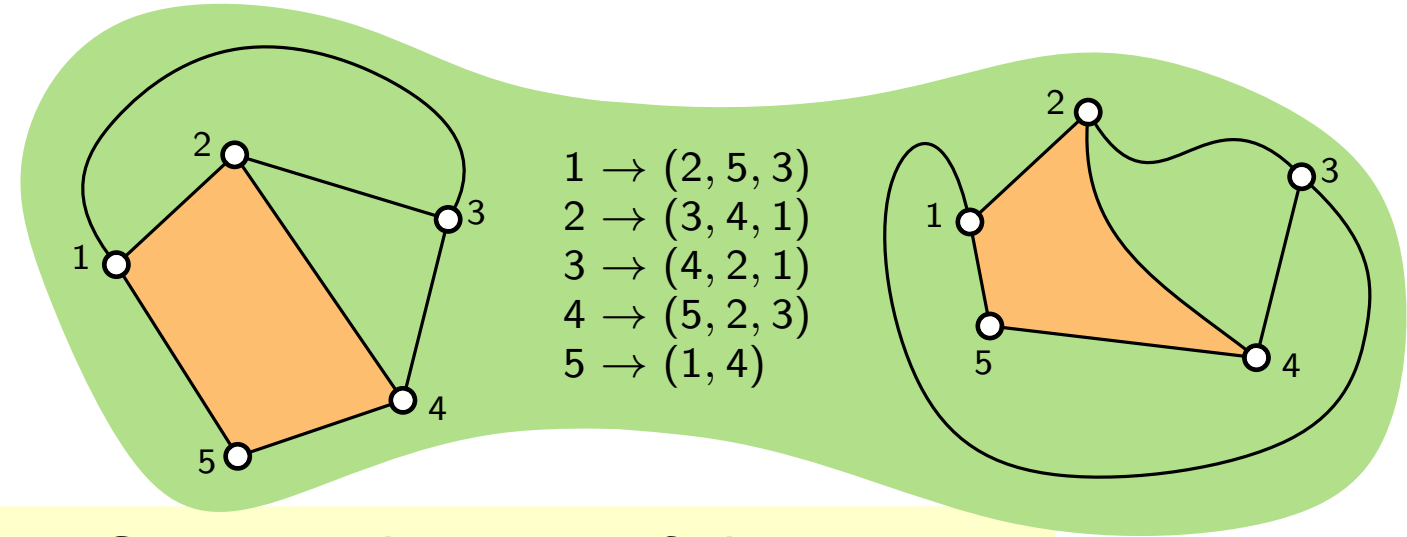
it can be drawn in such a way that no edges cross each other.

**planar embedding**:

Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

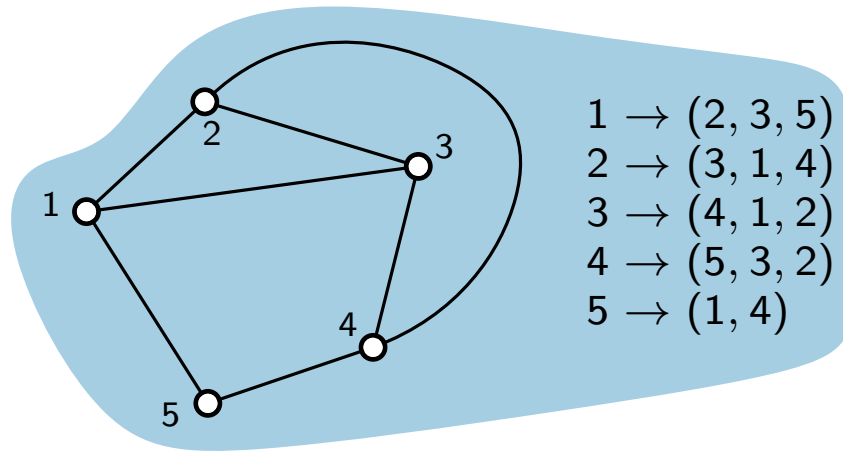
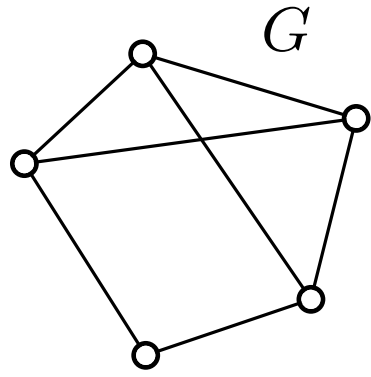
A planar embedding can have many planar drawings!



**faces**: Connected region of the plane bounded by edges



# Planar Graphs



$G$  is **planar**:

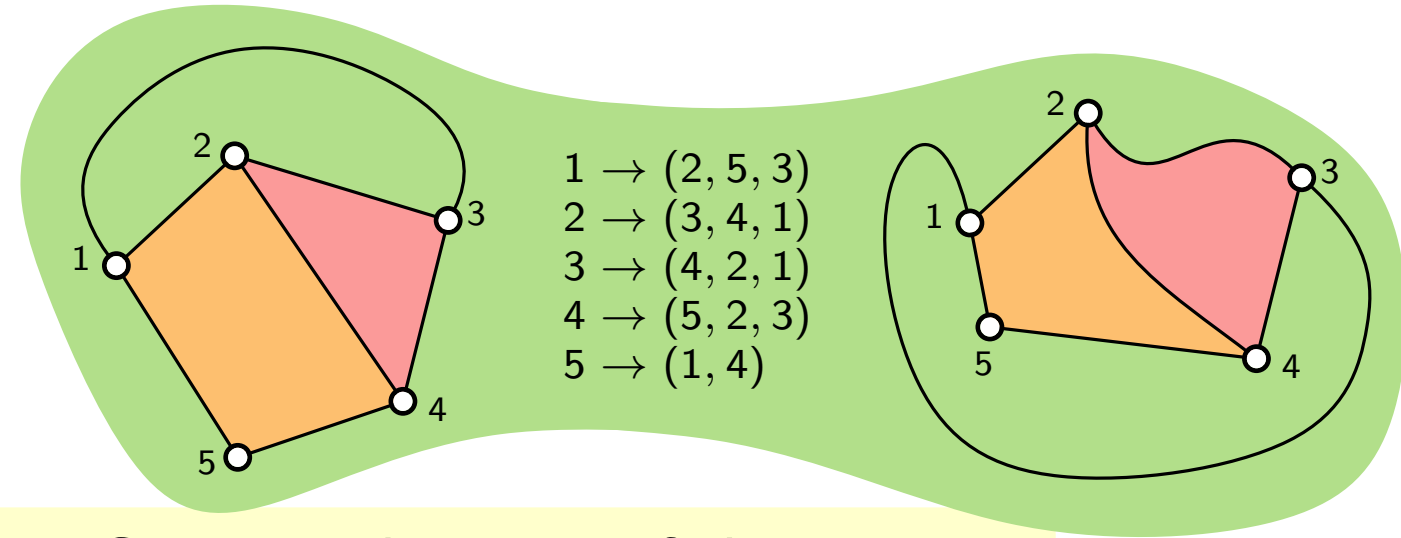
it can be drawn in such a way that no edges cross each other.

**planar embedding**:

Clockwise orientation of adjacent vertices around each vertex.

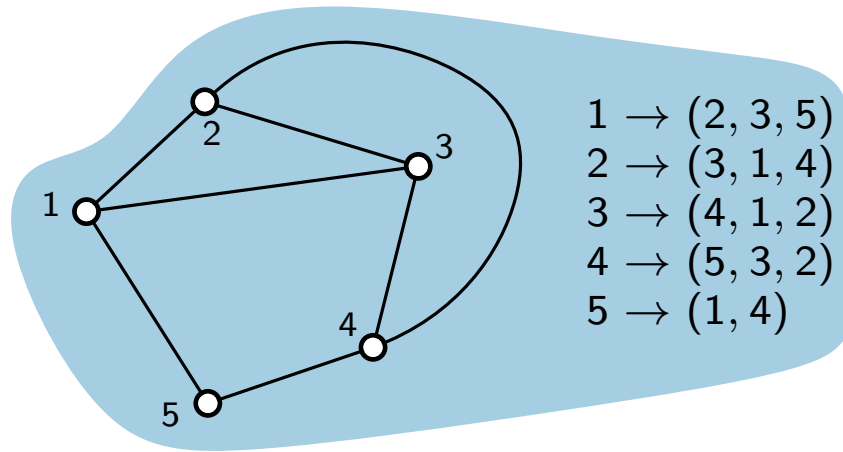
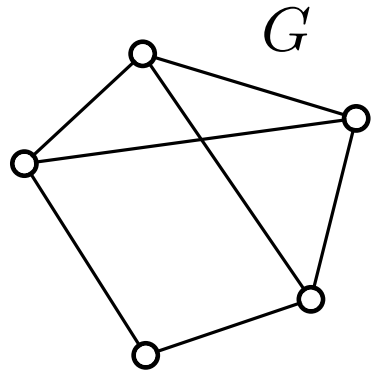
A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



**faces**: Connected region of the plane bounded by edges

# Planar Graphs



$G$  is **planar**:

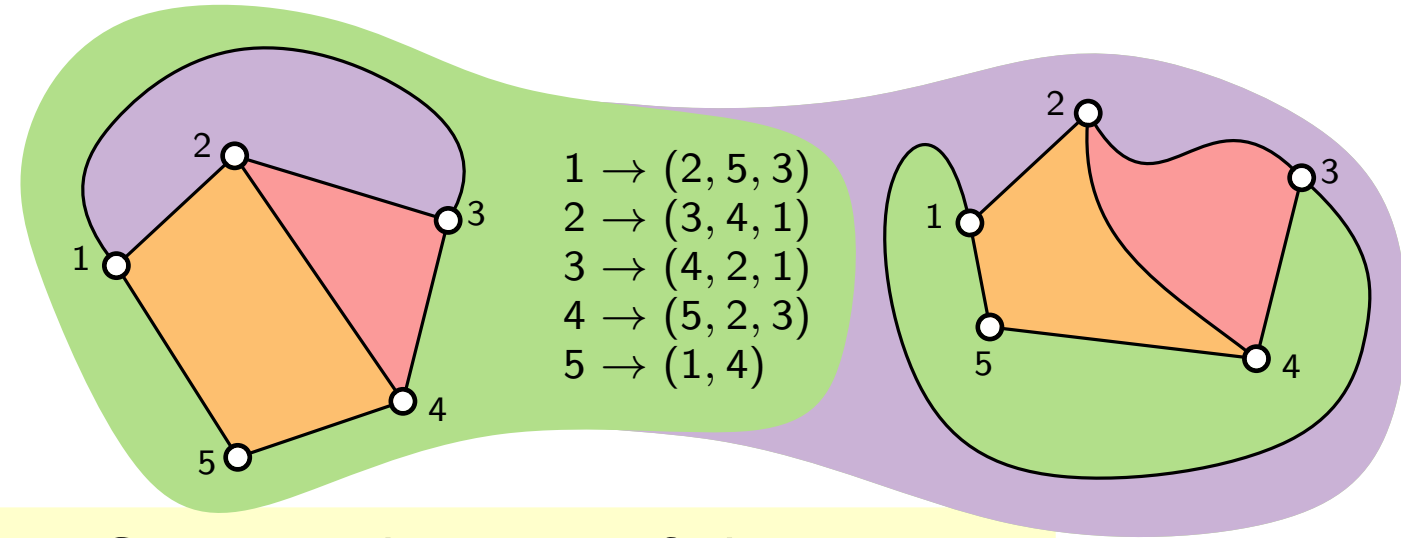
it can be drawn in such a way that no edges cross each other.

**planar embedding**:

Clockwise orientation of adjacent vertices around each vertex.

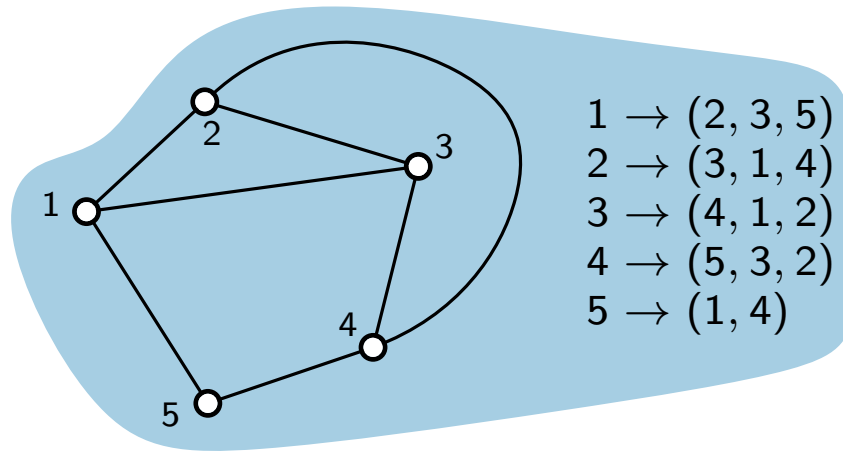
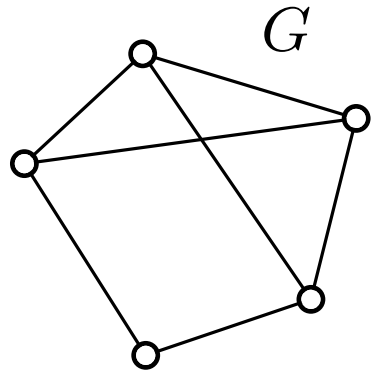
A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



**faces**: Connected region of the plane bounded by edges

# Planar Graphs



$G$  is **planar**:

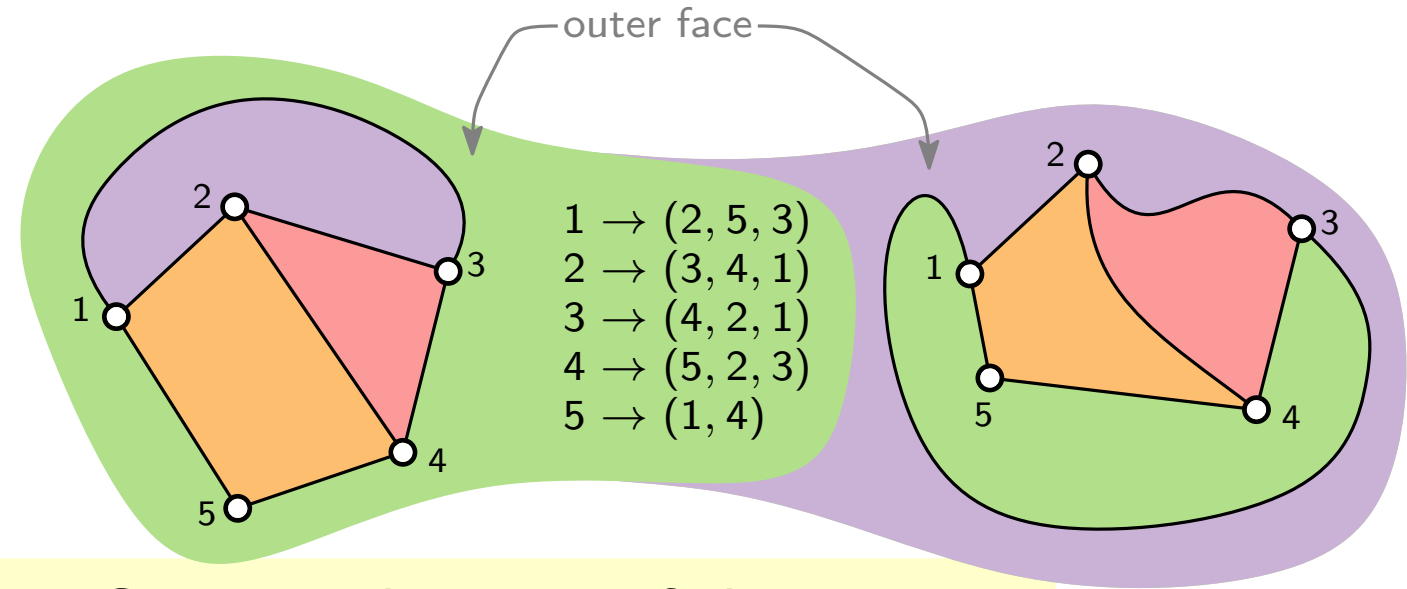
it can be drawn in such a way that no edges cross each other.

**planar embedding**:

Clockwise orientation of adjacent vertices around each vertex.

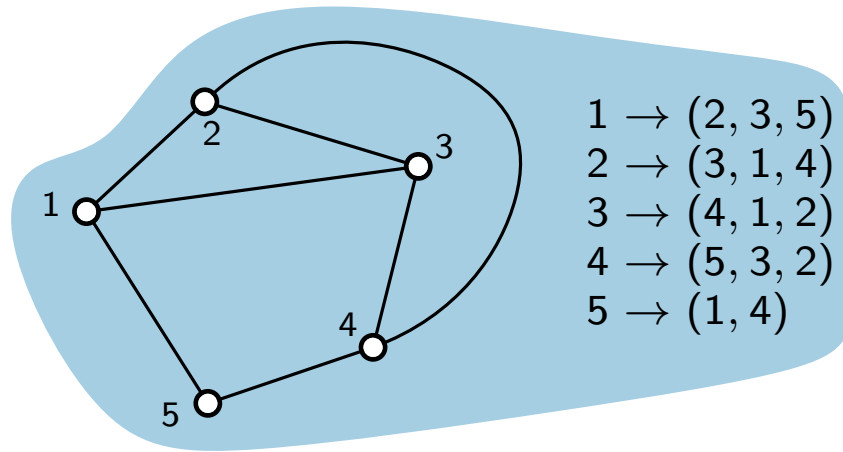
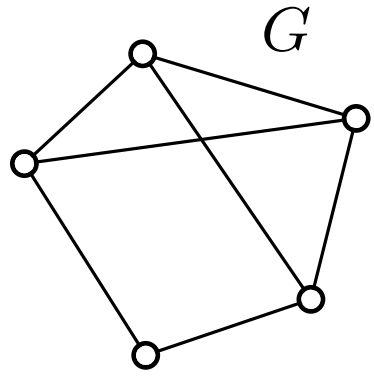
A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



**faces:** Connected region of the plane bounded by edges

# Planar Graphs



$G$  is **planar**:

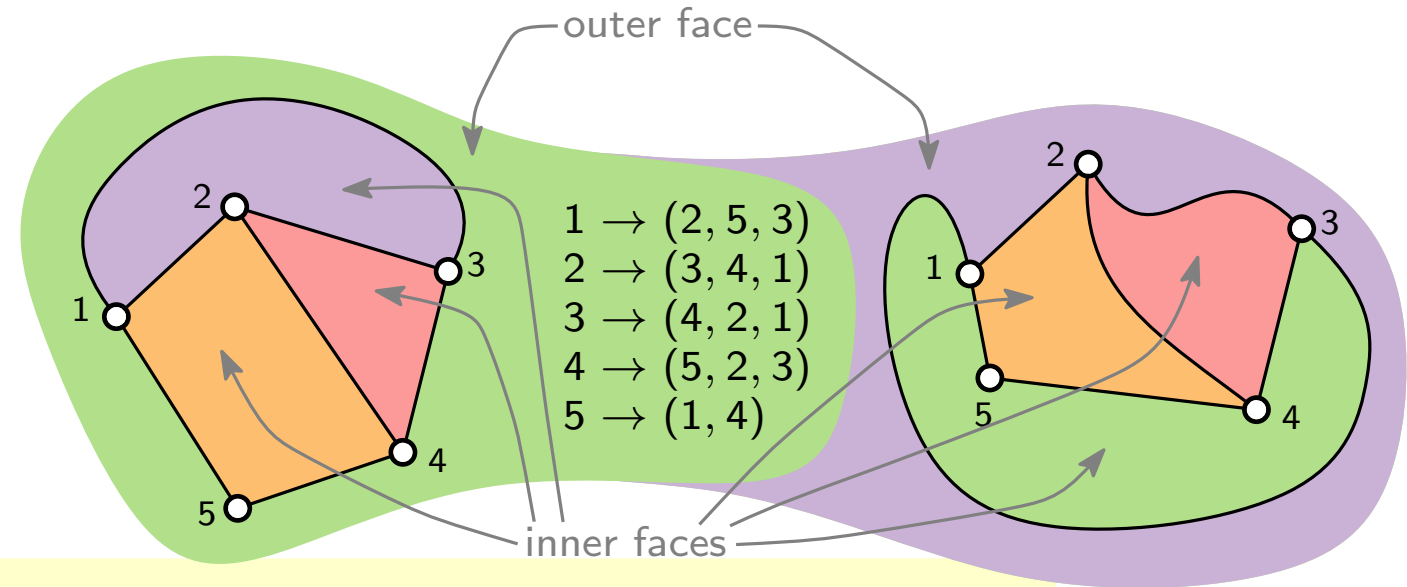
it can be drawn in such a way that no edges cross each other.

**planar embedding**:

Clockwise orientation of adjacent vertices around each vertex.

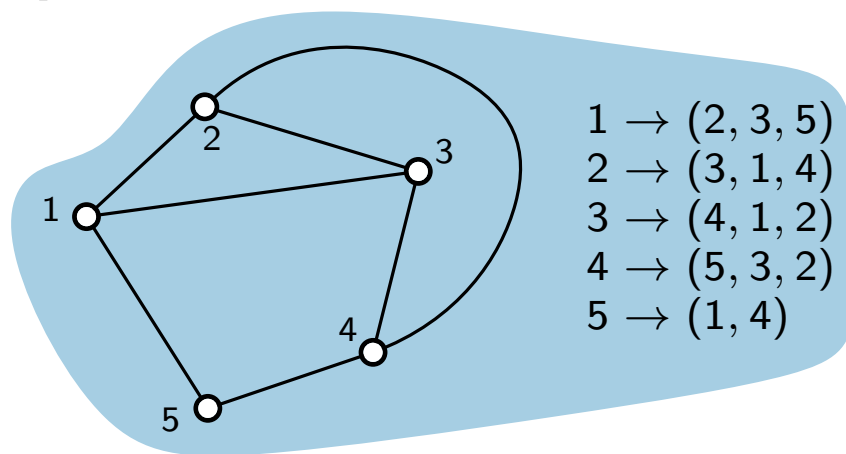
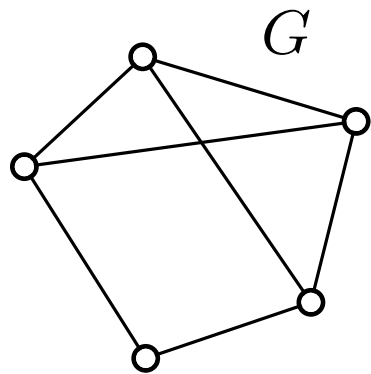
A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



**faces**: Connected region of the plane bounded by edges

# Planar Graphs



$G$  is **planar**:

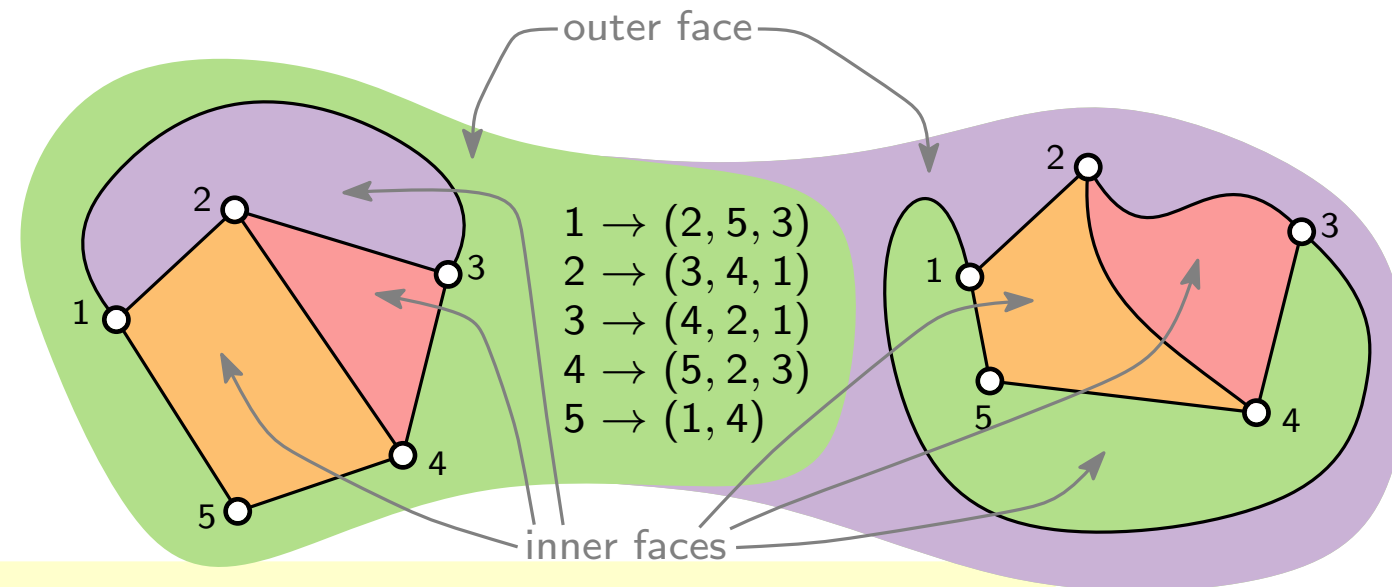
it can be drawn in such a way that no edges cross each other.

**planar embedding**:

Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!

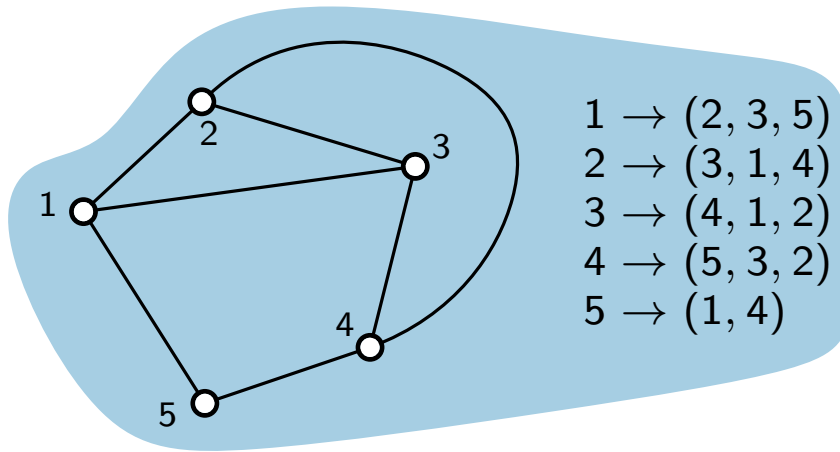
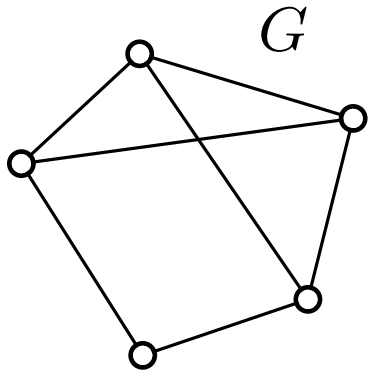


**faces:** Connected region of the plane bounded by edges

**Euler's polyhedra formula.**

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

# Planar Graphs



$G$  is **planar**:

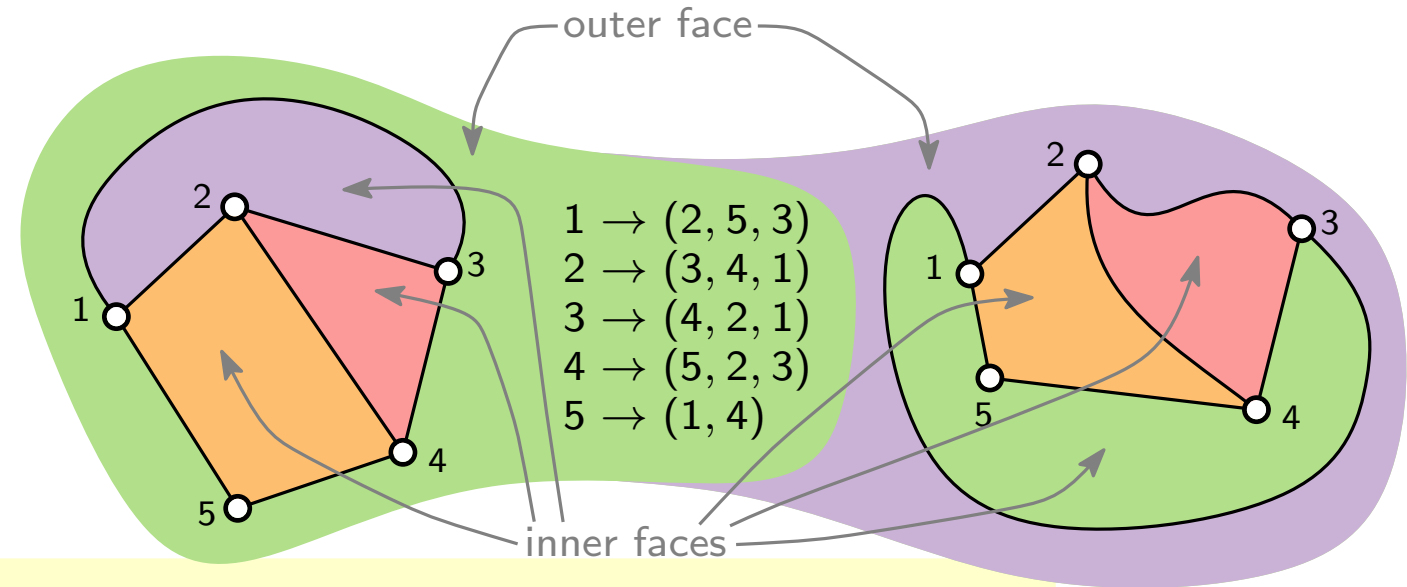
it can be drawn in such a way that no edges cross each other.

**planar embedding**:

Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



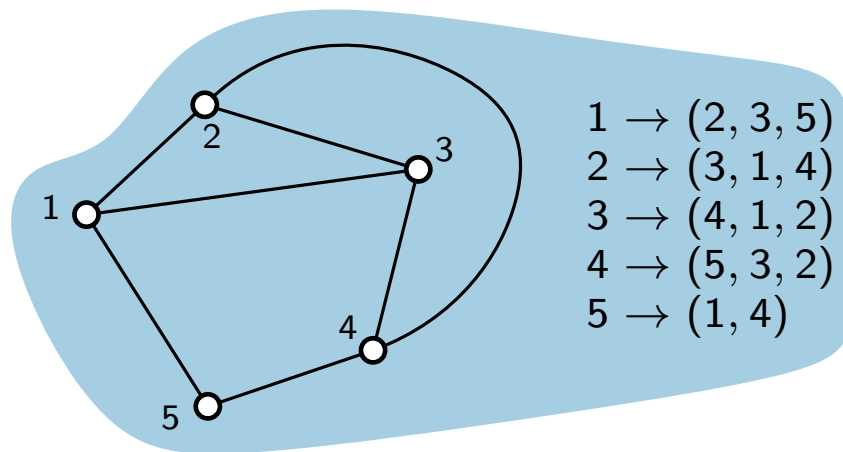
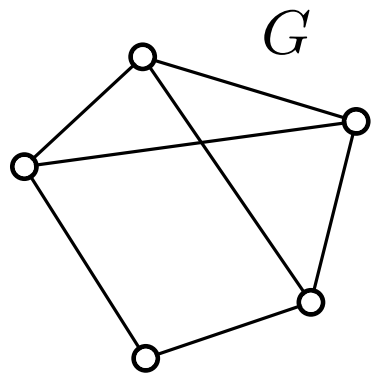
**faces:** Connected region of the plane bounded by edges

**Euler's polyhedra formula.**

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Proof.**

# Planar Graphs



$G$  is **planar**:

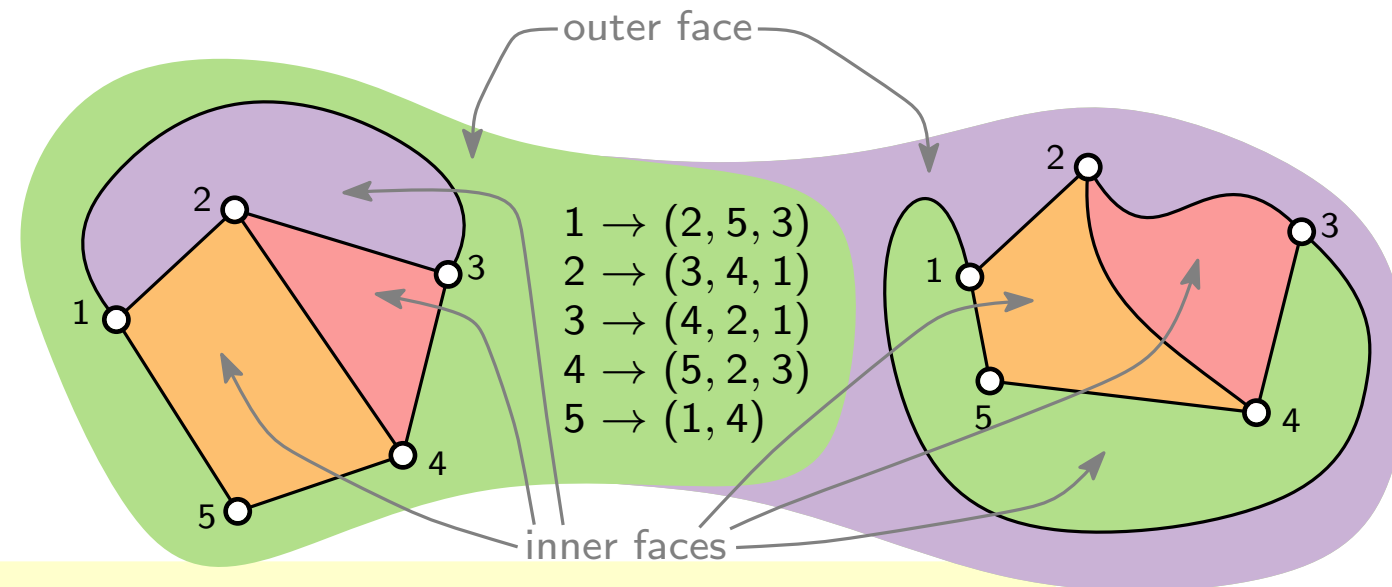
it can be drawn in such a way that no edges cross each other.

**planar embedding**:

Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



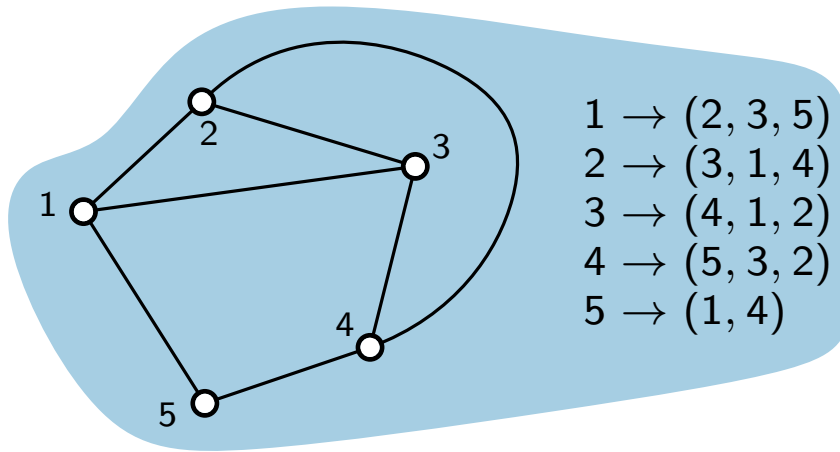
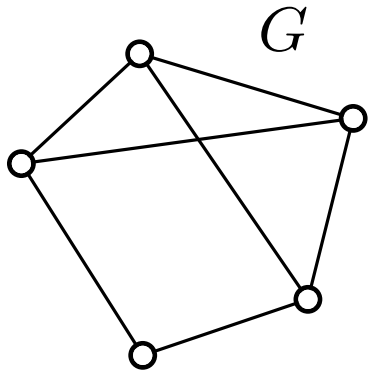
**faces:** Connected region of the plane bounded by edges

**Euler's polyhedra formula.**

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Proof.** By induction on  $m$ :

# Planar Graphs



$G$  is **planar**:

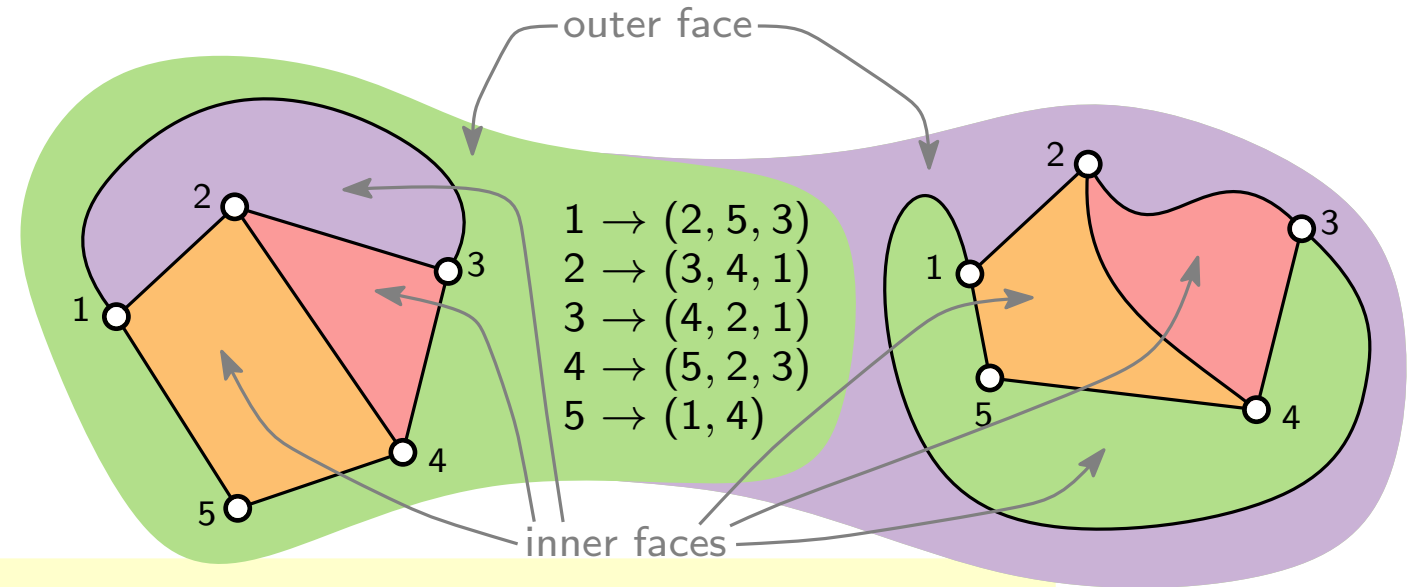
it can be drawn in such a way that no edges cross each other.

**planar embedding:**

Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



**faces:** Connected region of the plane bounded by edges

**Euler's polyhedra formula.**

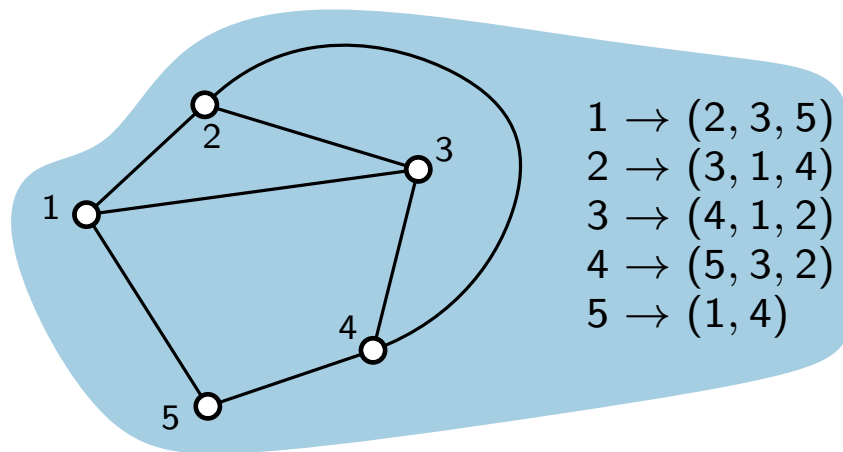
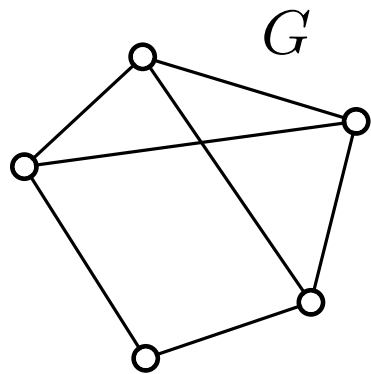
$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Proof.** By induction on  $m$ :

$$m = 0 \Rightarrow$$



# Planar Graphs



$G$  is **planar**:

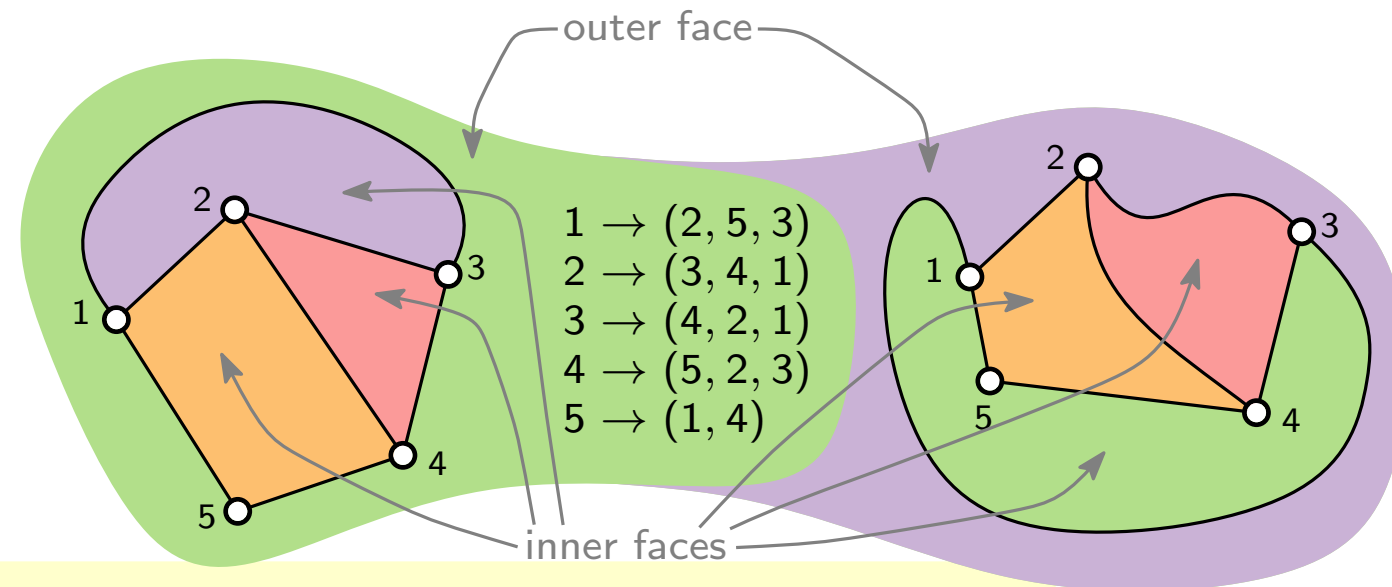
it can be drawn in such a way that no edges cross each other.

**planar embedding**:

Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



**faces:** Connected region of the plane bounded by edges

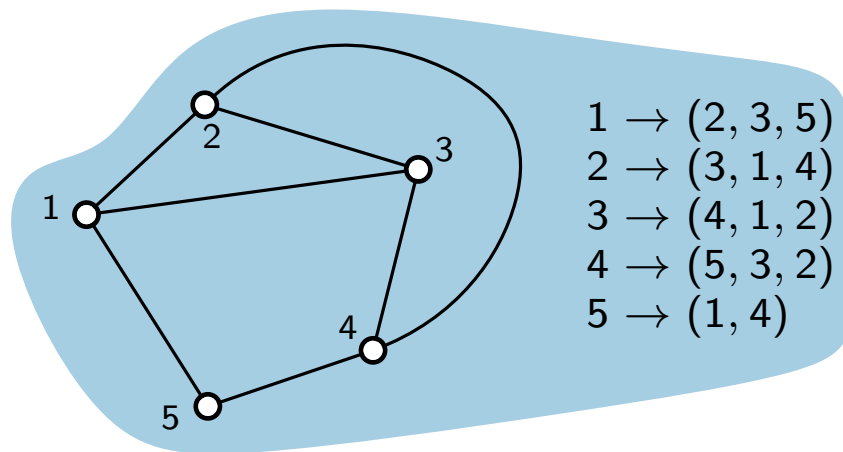
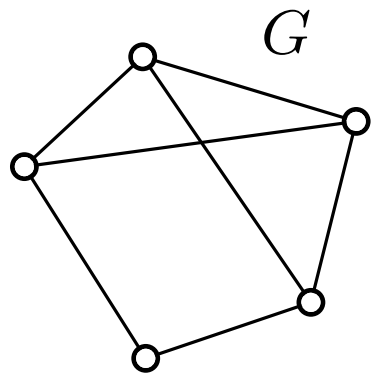
**Euler's polyhedra formula.**

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Proof.** By induction on  $m$ :

$$m = 0 \Rightarrow f = ? \text{ and } c = ?$$

# Planar Graphs



$G$  is **planar**:

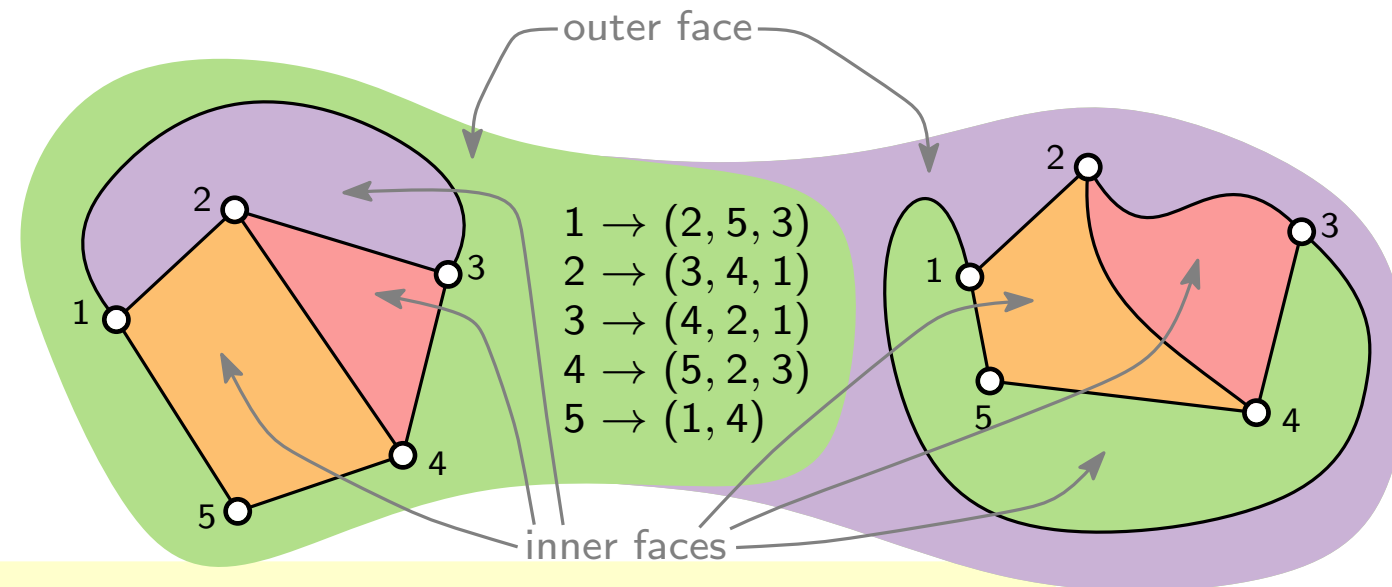
it can be drawn in such a way that no edges cross each other.

**planar embedding:**

Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



**faces:** Connected region of the plane bounded by edges

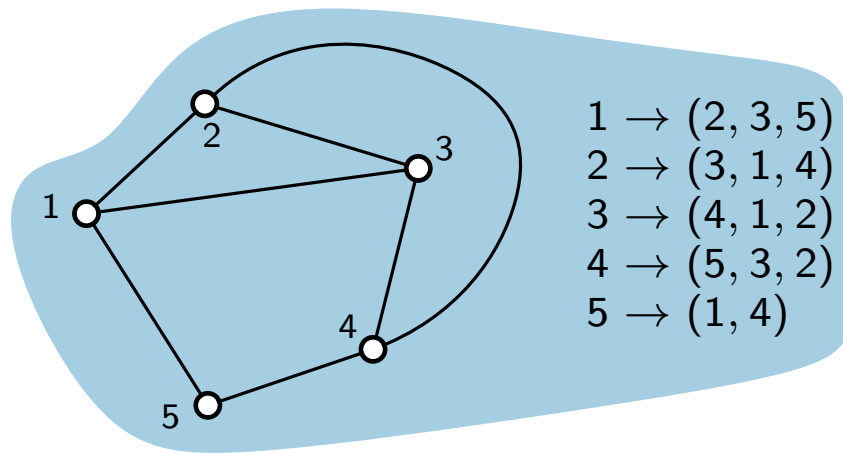
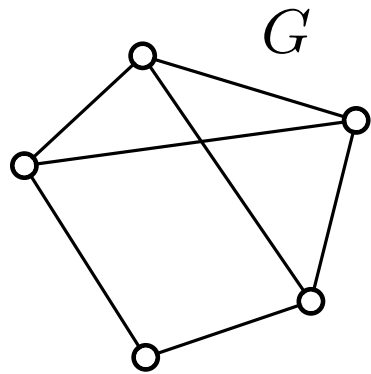
**Euler's polyhedra formula.**

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Proof.** By induction on  $m$ :

$$m = 0 \Rightarrow f = 1 \text{ and } c = n$$

# Planar Graphs



$G$  is **planar**:

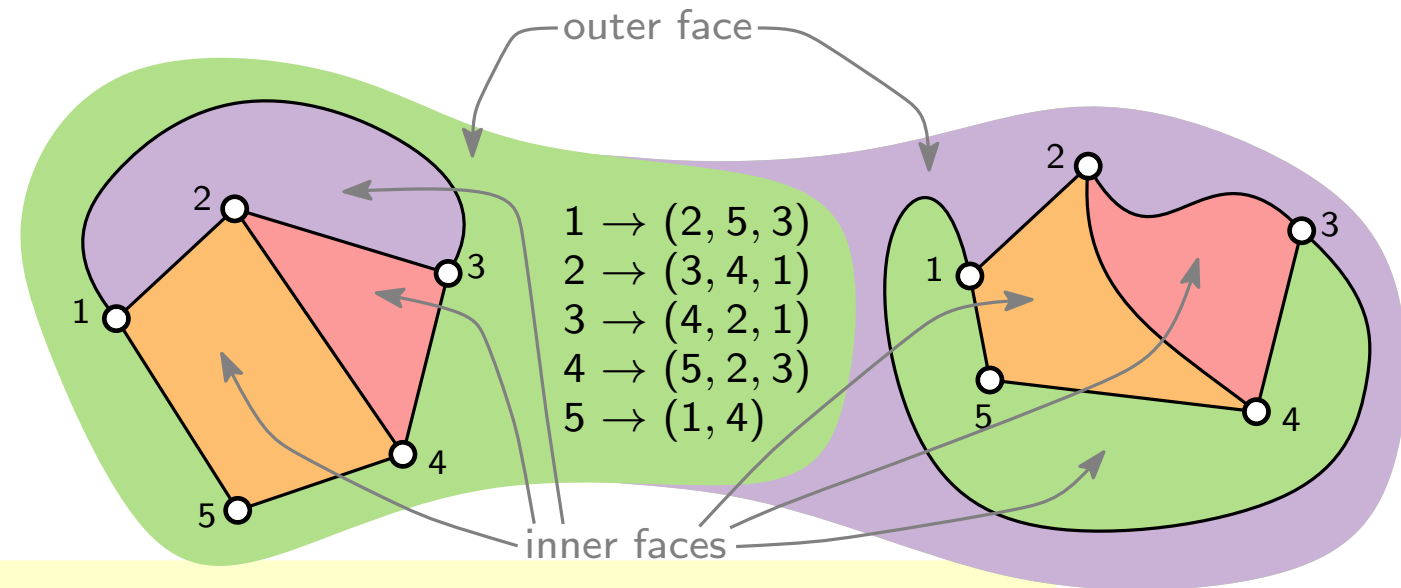
it can be drawn in such a way that no edges cross each other.

**planar embedding:**

Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



**faces:** Connected region of the plane bounded by edges

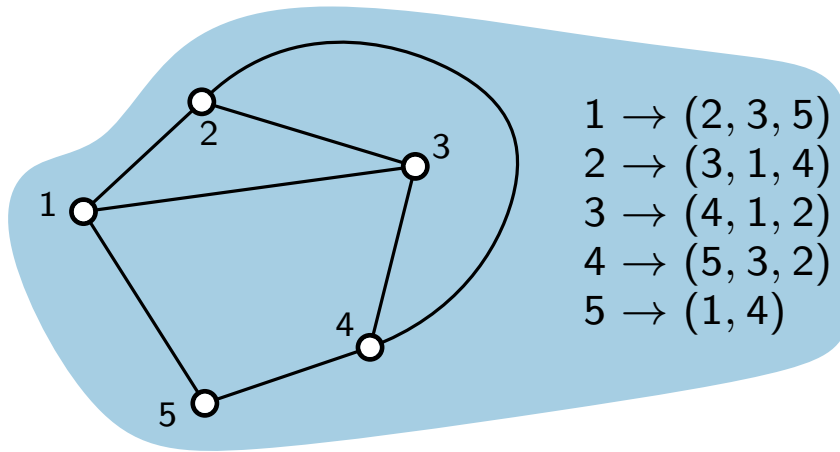
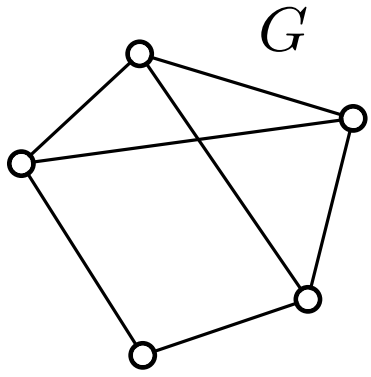
**Euler's polyhedra formula.**

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Proof.** By induction on  $m$ :

$$\begin{aligned} m = 0 &\Rightarrow f = 1 \text{ and } c = n \\ &\Rightarrow 1 - 0 + n = n + 1 \end{aligned}$$

# Planar Graphs



$G$  is **planar**:

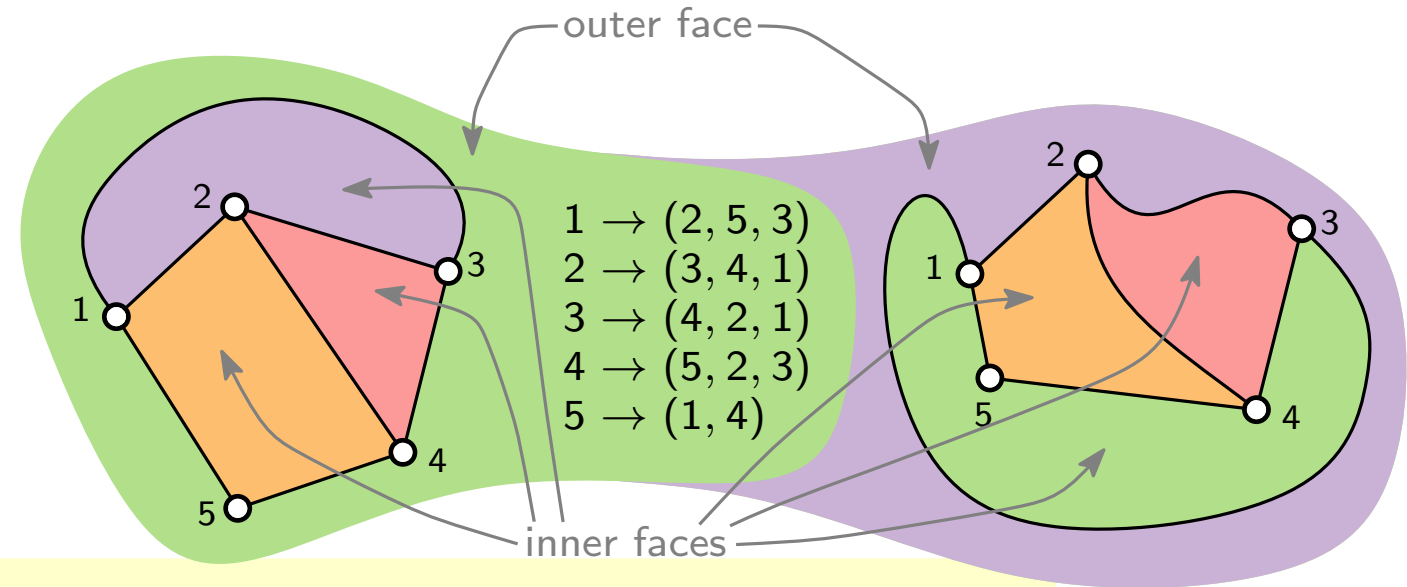
it can be drawn in such a way that no edges cross each other.

**planar embedding:**

Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



**faces:** Connected region of the plane bounded by edges

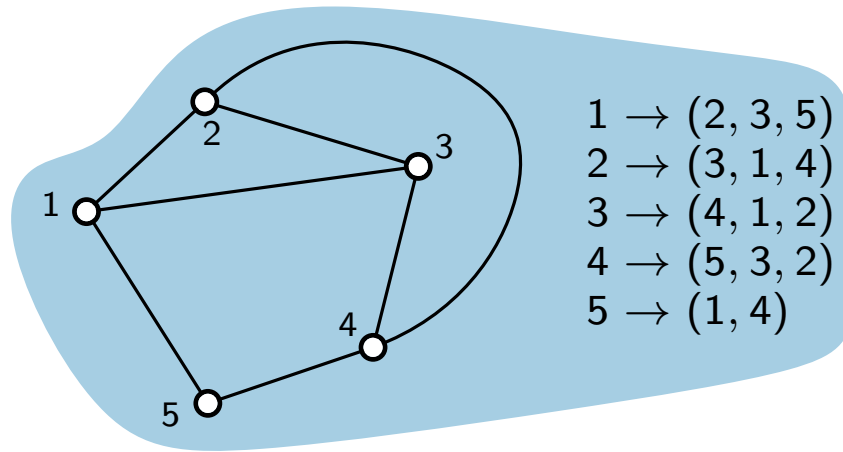
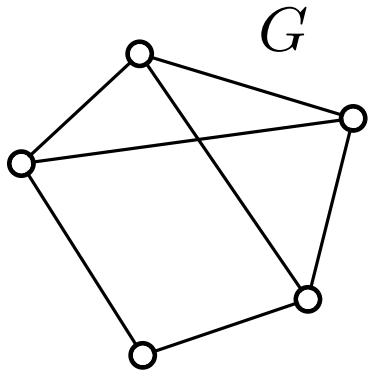
**Euler's polyhedra formula.**

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Proof.** By induction on  $m$ :

$$\begin{aligned} m = 0 &\Rightarrow f = 1 \text{ and } c = n \\ &\Rightarrow 1 - 0 + n = n + 1 \quad \checkmark \end{aligned}$$

# Planar Graphs



$G$  is **planar**:

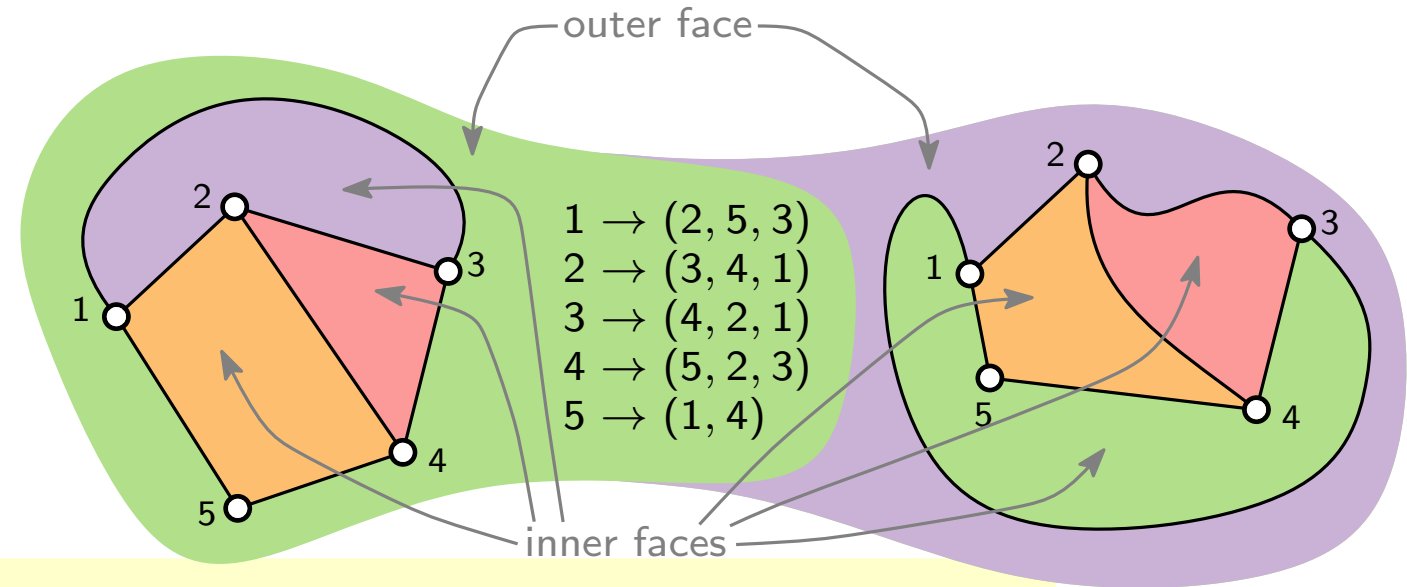
it can be drawn in such a way that no edges cross each other.

**planar embedding:**

Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



**faces:** Connected region of the plane bounded by edges

**Euler's polyhedra formula.**

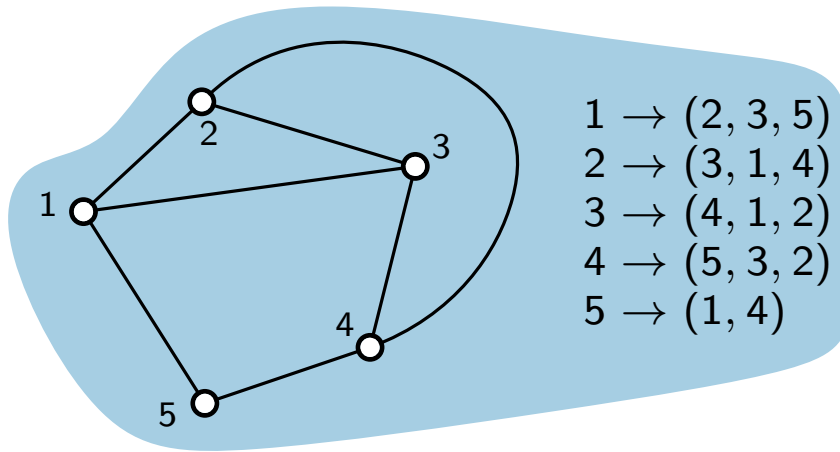
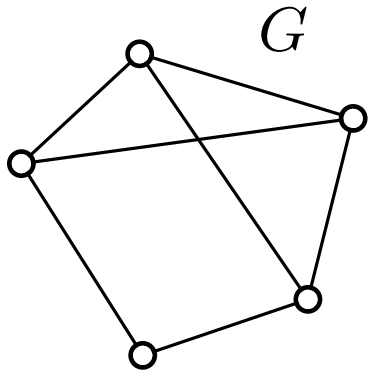
$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Proof.** By induction on  $m$ :

$$\begin{aligned} m = 0 &\Rightarrow f = 1 \text{ and } c = n \\ &\Rightarrow 1 - 0 + n = n + 1 \quad \checkmark \end{aligned}$$

$$m \geq 1 \Rightarrow$$

# Planar Graphs



$G$  is **planar**:

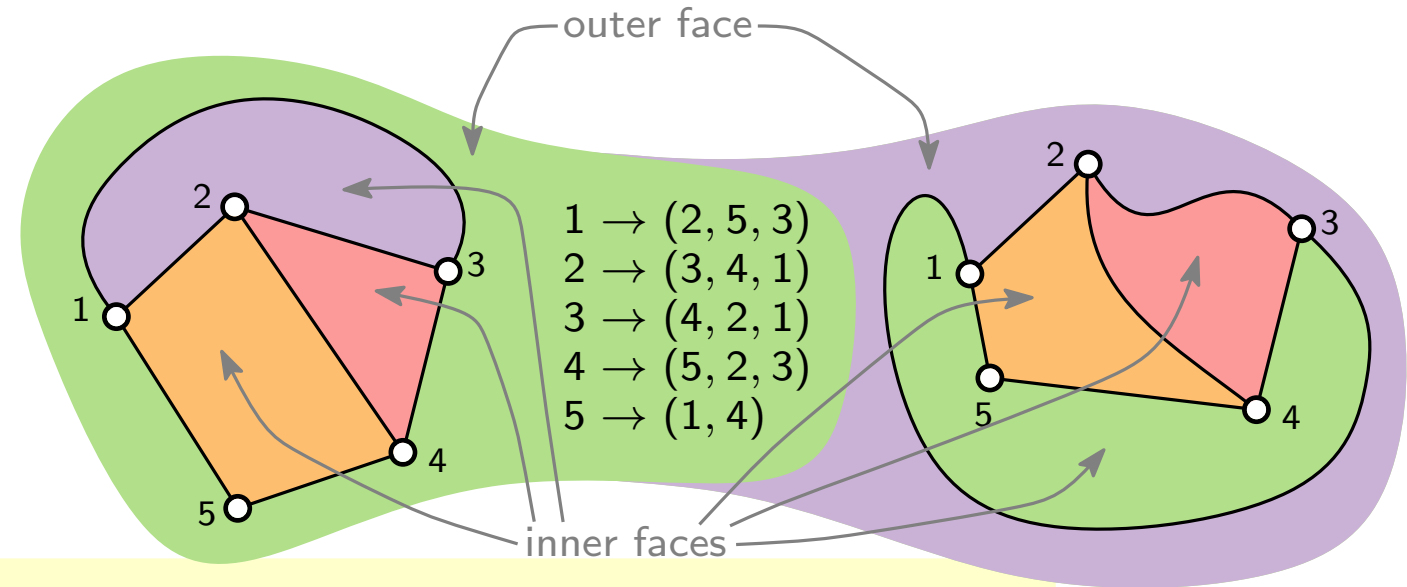
it can be drawn in such a way that no edges cross each other.

**planar embedding:**

Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



**faces:** Connected region of the plane bounded by edges

**Euler's polyhedra formula.**

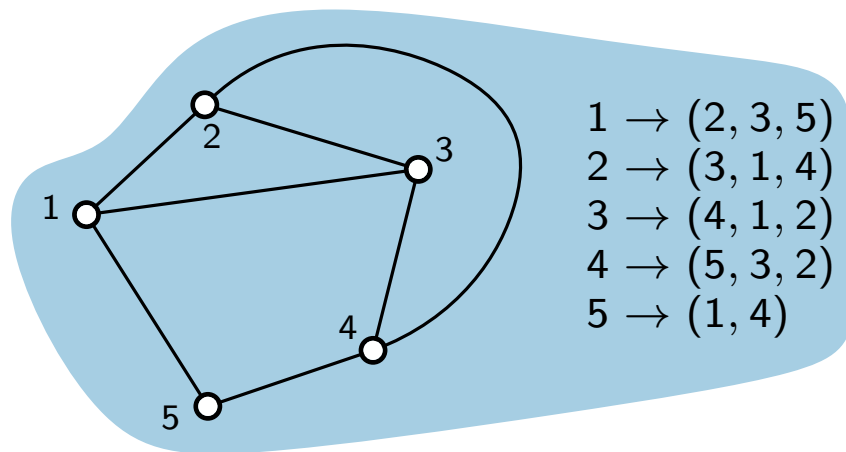
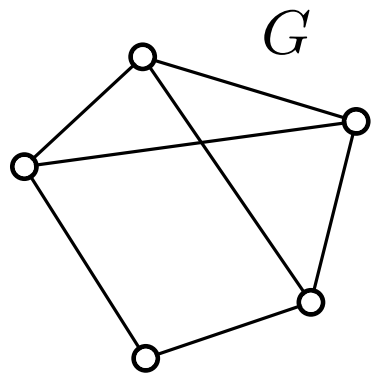
$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Proof.** By induction on  $m$ :

$$\begin{aligned} m = 0 &\Rightarrow f = 1 \text{ and } c = n \\ &\Rightarrow 1 - 0 + n = n + 1 \quad \checkmark \end{aligned}$$

$$m \geq 1 \Rightarrow \text{remove some edge } e$$

# Planar Graphs



$G$  is **planar**:

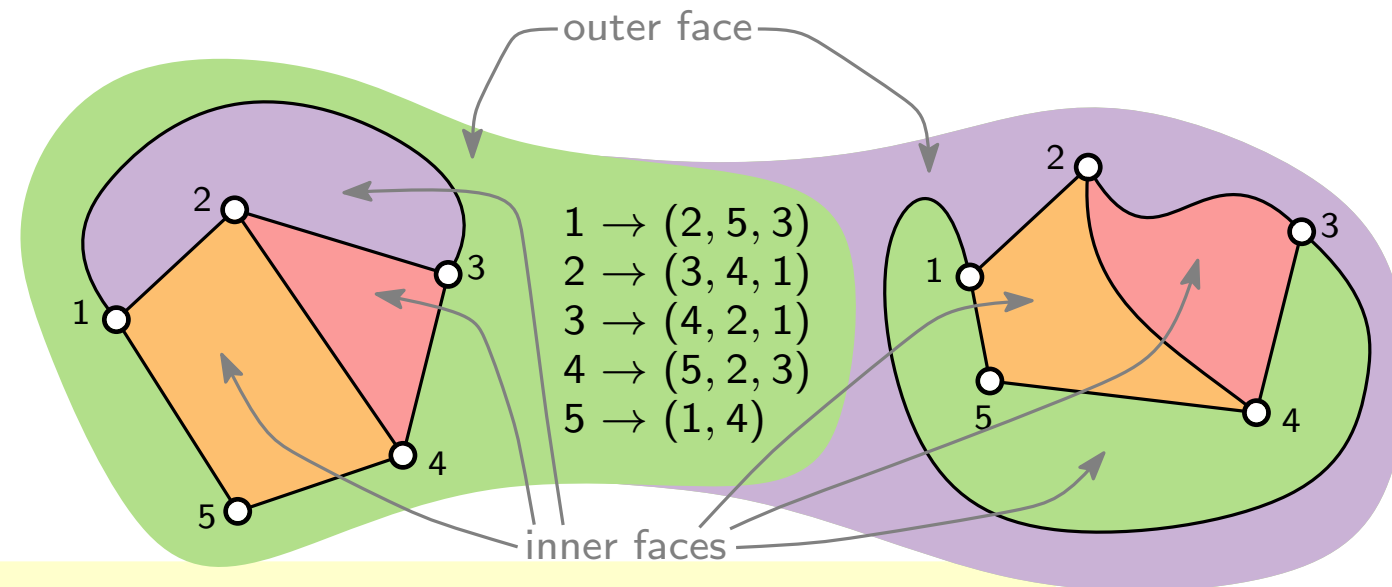
it can be drawn in such a way that no edges cross each other.

**planar embedding:**

Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



**faces:** Connected region of the plane bounded by edges

**Euler's polyhedra formula.**

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Proof.** By induction on  $m$ :

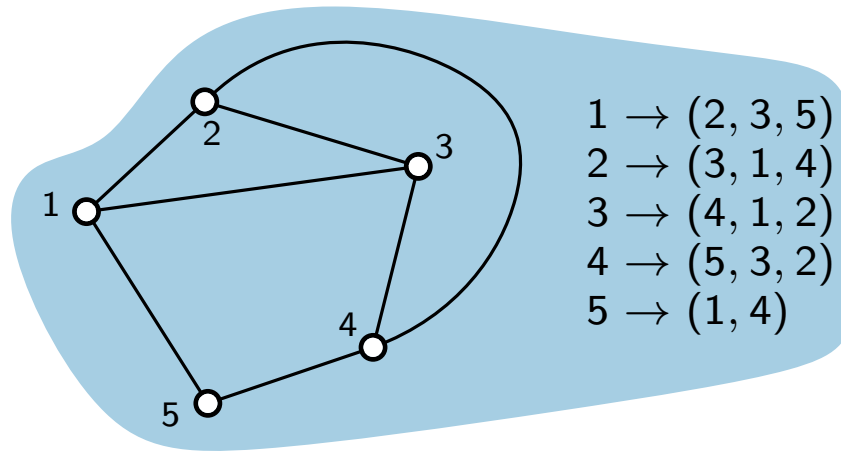
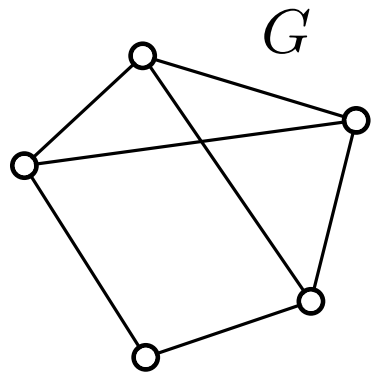
$$m = 0 \Rightarrow f = 1 \text{ and } c = n$$

$$\Rightarrow 1 - 0 + n = n + 1 \quad \checkmark$$

$$m \geq 1 \Rightarrow \text{remove some edge } e \Rightarrow m \rightarrow m - 1$$



# Planar Graphs



$G$  is **planar**:

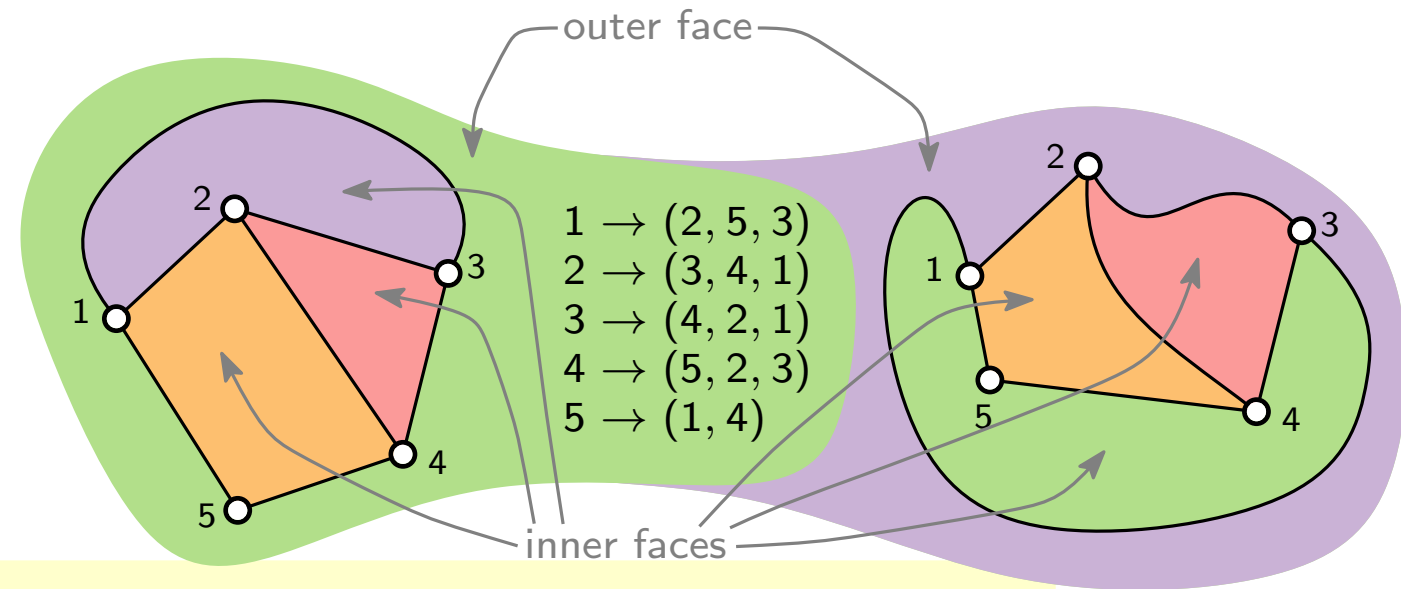
it can be drawn in such a way that no edges cross each other.

**planar embedding:**

Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



**faces:** Connected region of the plane bounded by edges

**Euler's polyhedra formula.**

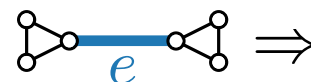
$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Proof.** By induction on  $m$ :

$$m = 0 \Rightarrow f = 1 \text{ and } c = n$$

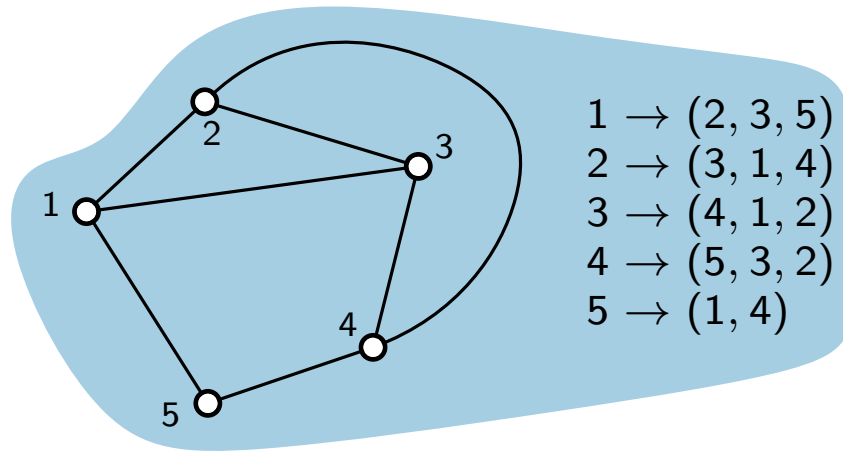
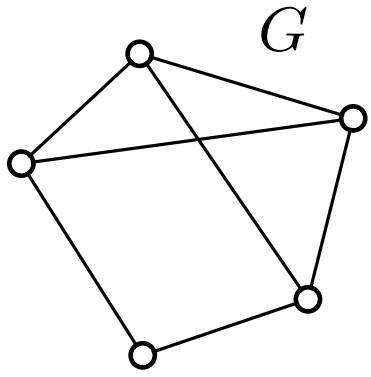
$$\Rightarrow 1 - 0 + n = n + 1 \quad \checkmark$$

$$m \geq 1 \Rightarrow \text{remove some edge } e \Rightarrow m \rightarrow m - 1$$





# Planar Graphs



$G$  is **planar**:

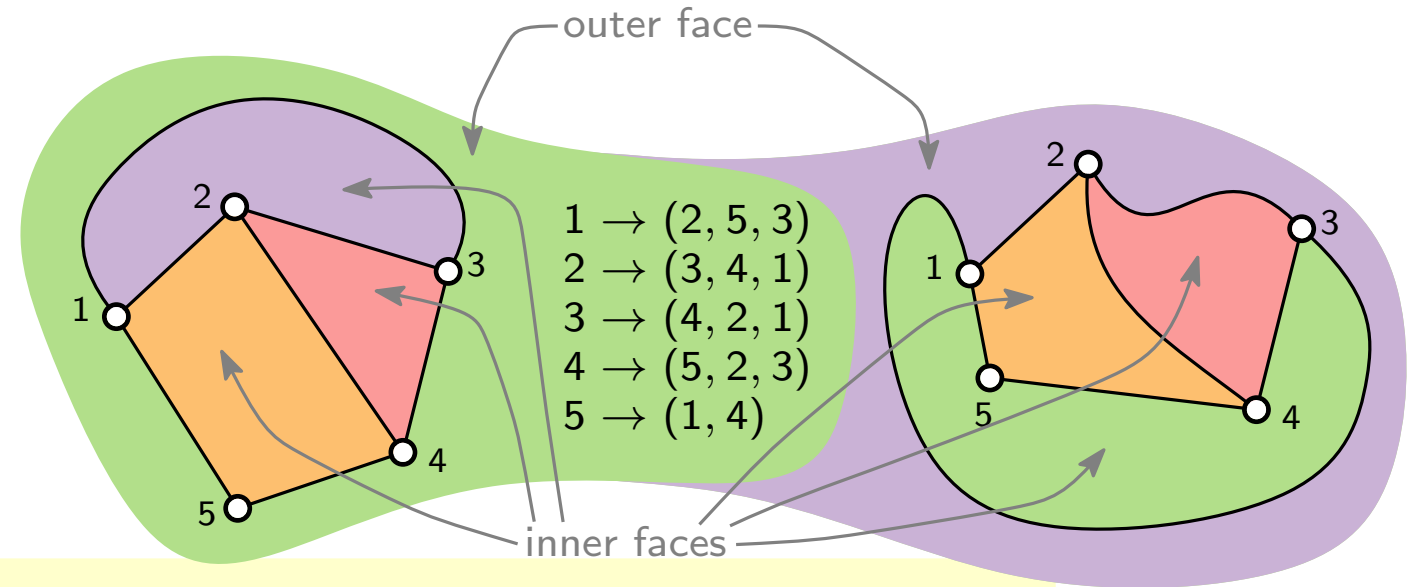
it can be drawn in such a way that no edges cross each other.

**planar embedding:**

Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



**faces:** Connected region of the plane bounded by edges

**Euler's polyhedra formula.**

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Proof.** By induction on  $m$ :

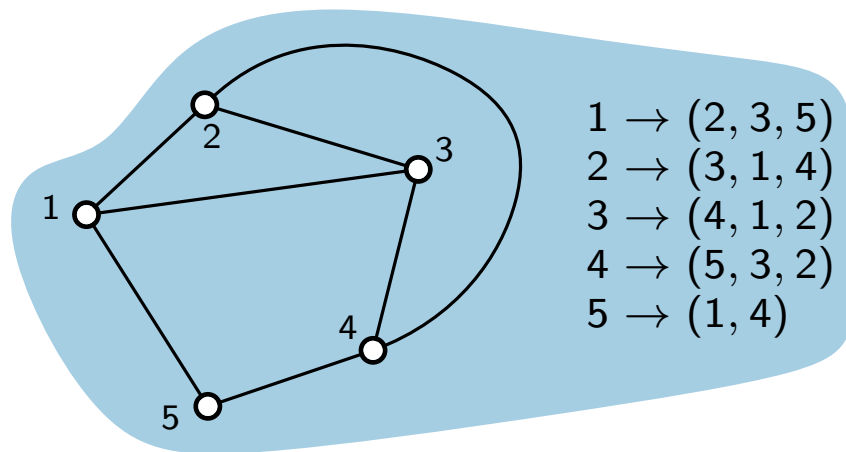
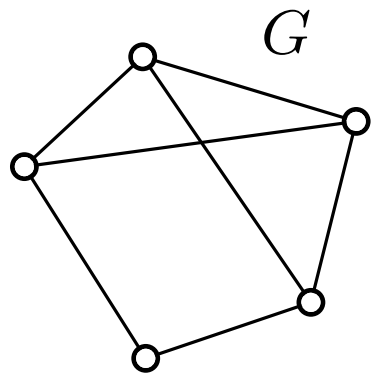
$$m = 0 \Rightarrow f = 1 \text{ and } c = n$$

$$\Rightarrow 1 - 0 + n = n + 1 \quad \checkmark$$

$$m \geq 1 \Rightarrow \text{remove some edge } e \Rightarrow m \rightarrow m - 1$$

$$\text{graph} \xrightarrow{e} \text{graph} \Rightarrow c \rightarrow c + 1$$

# Planar Graphs



$G$  is **planar**:

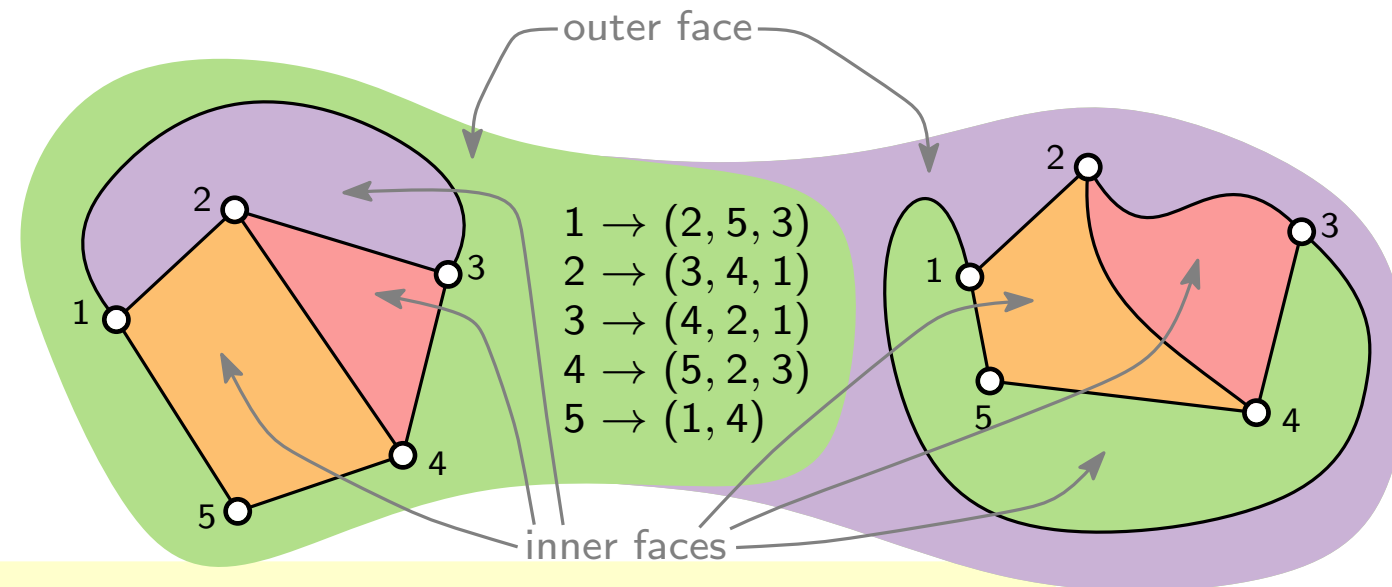
it can be drawn in such a way that no edges cross each other.

**planar embedding**:

Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



**faces:** Connected region of the plane bounded by edges

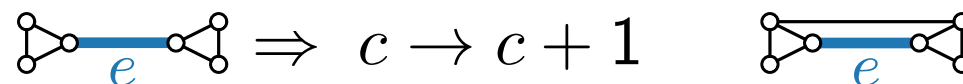
**Euler's polyhedra formula.**

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

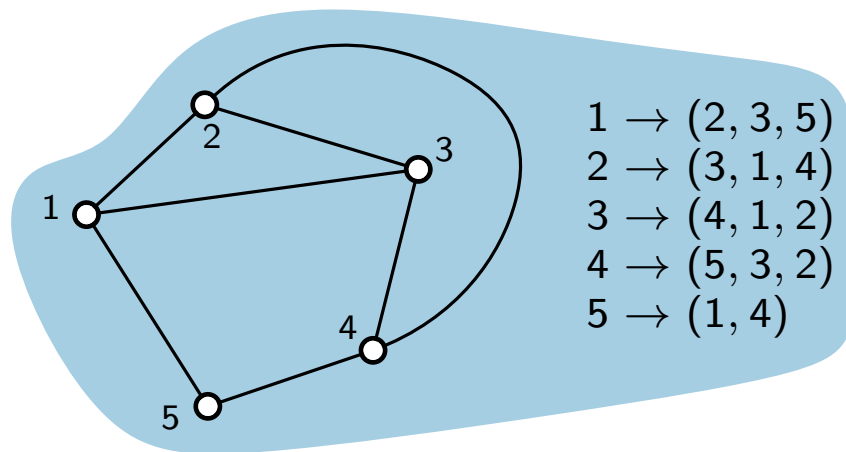
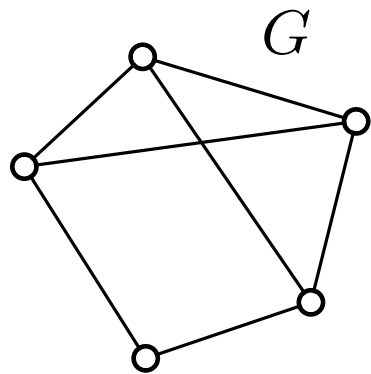
**Proof.** By induction on  $m$ :

$$\begin{aligned} m = 0 &\Rightarrow f = 1 \text{ and } c = n \\ &\Rightarrow 1 - 0 + n = n + 1 \quad \checkmark \end{aligned}$$

$$m \geq 1 \Rightarrow \text{remove some edge } e \Rightarrow m \rightarrow m - 1$$



# Planar Graphs



$G$  is **planar**:

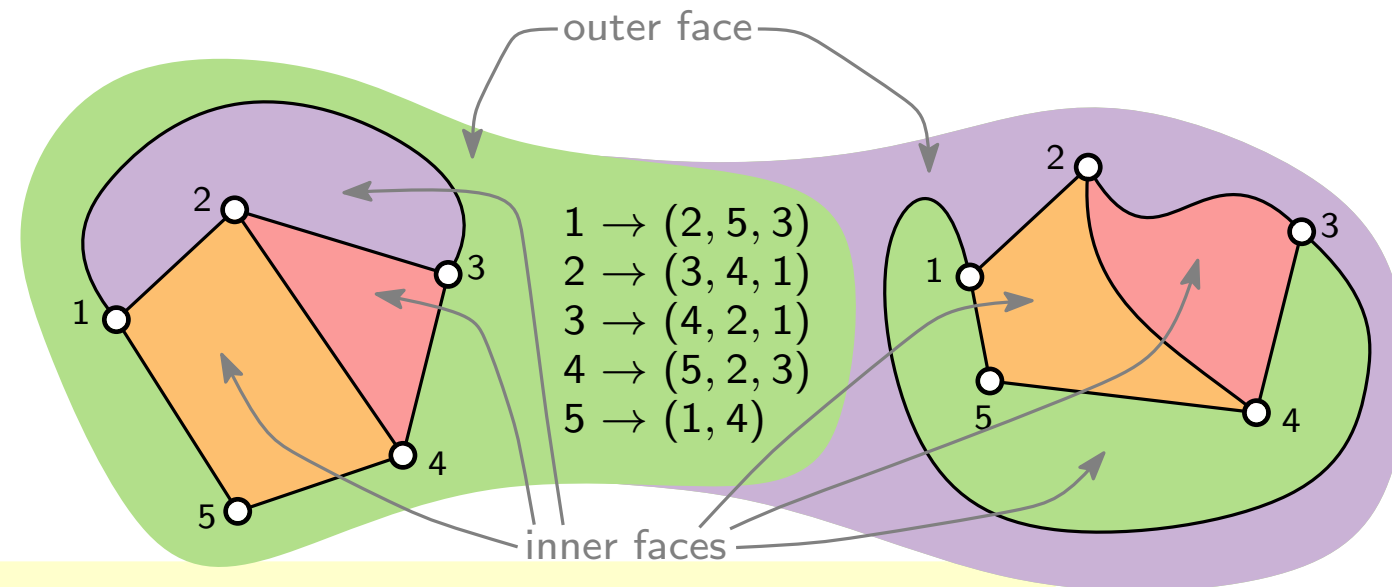
it can be drawn in such a way that no edges cross each other.

**planar embedding**:

Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



**faces:** Connected region of the plane bounded by edges

**Euler's polyhedra formula.**

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

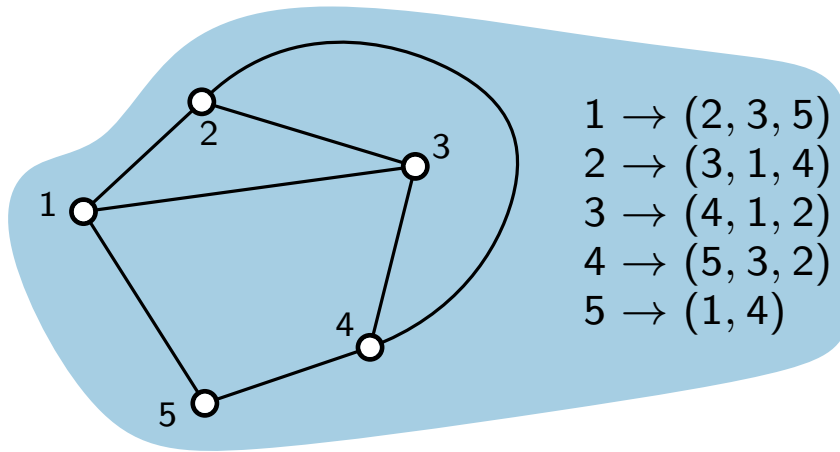
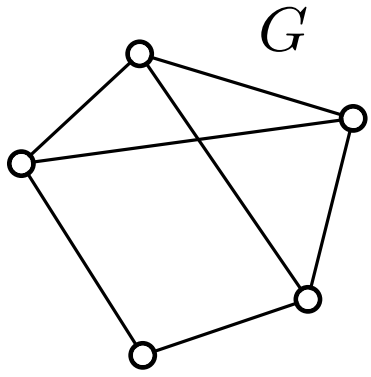
**Proof.** By induction on  $m$ :

$$\begin{aligned} m = 0 &\Rightarrow f = 1 \text{ and } c = n \\ &\Rightarrow 1 - 0 + n = n + 1 \quad \checkmark \end{aligned}$$

$$m \geq 1 \Rightarrow \text{remove some edge } e \Rightarrow m \rightarrow m - 1$$

$$\begin{array}{cc} \text{graph with edge } e & \Rightarrow c \rightarrow c + 1 \\ \text{graph with edge } e & \Rightarrow f \rightarrow f - 1 \end{array}$$

# Planar Graphs



$G$  is **planar**:

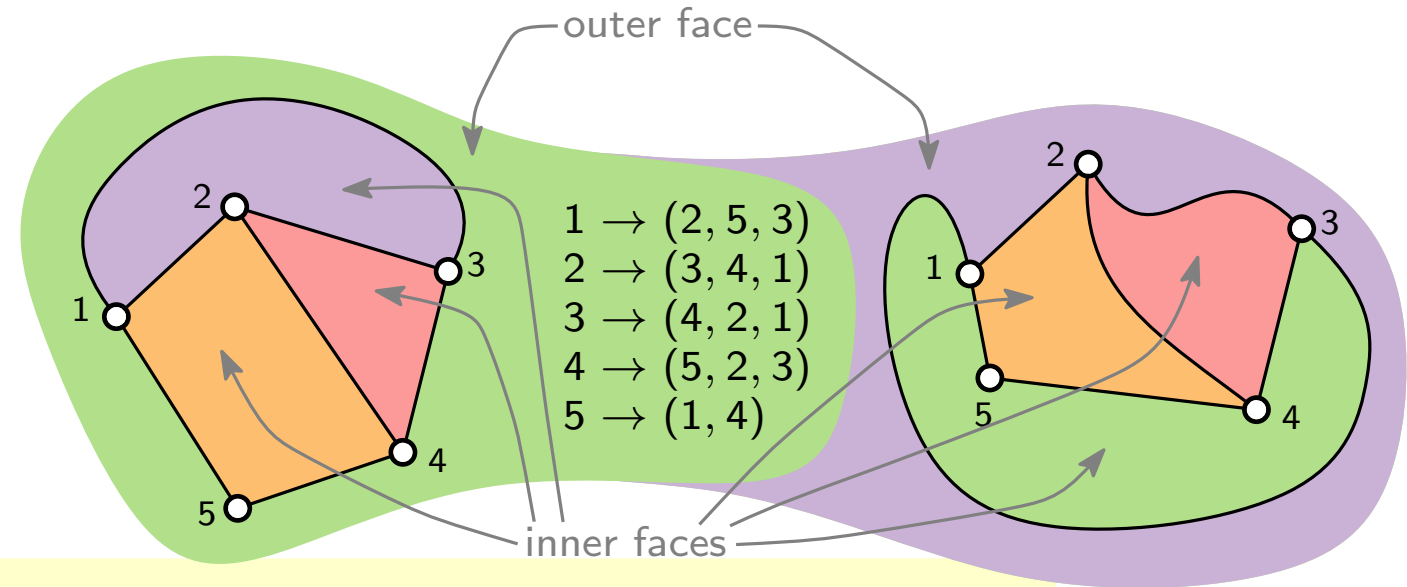
it can be drawn in such a way that no edges cross each other.

**planar embedding**:

Clockwise orientation of adjacent vertices around each vertex.

A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!



**faces:** Connected region of the plane bounded by edges

**Euler's polyhedra formula.**

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Proof.** By induction on  $m$ :

$$m = 0 \Rightarrow f = 1 \text{ and } c = n$$

$$\Rightarrow 1 - 0 + n = n + 1 \quad \checkmark$$

$$m \geq 1 \Rightarrow \text{remove some edge } e \Rightarrow m \rightarrow m - 1$$



$$\Rightarrow c \rightarrow c + 1$$



$$\Rightarrow f \rightarrow f - 1 \quad \checkmark$$

# Properties of Planar Graphs

**Euler's polyhedra formula.**

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

1.  $m \leq 3n - 6$

# Properties of Planar Graphs

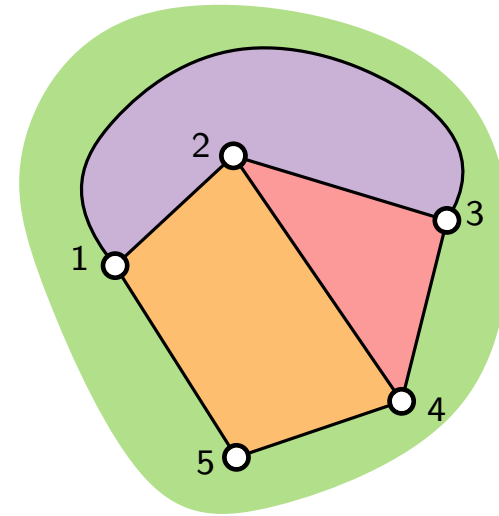
## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

1.  $m \leq 3n - 6$

**Proof.** 1.





# Properties of Planar Graphs

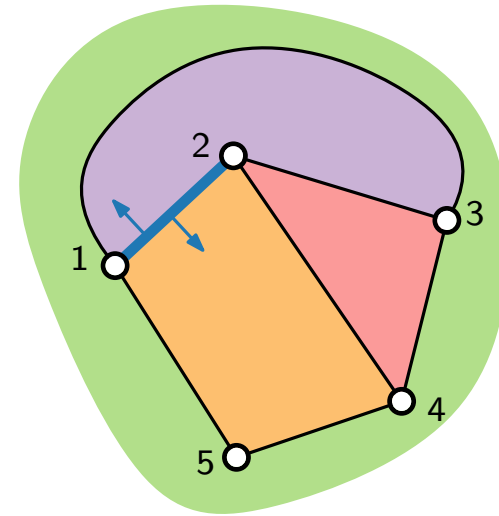
## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

1.  $m \leq 3n - 6$

**Proof.** 1. Every **edge** incident to  $\leq 2$  faces



# Properties of Planar Graphs

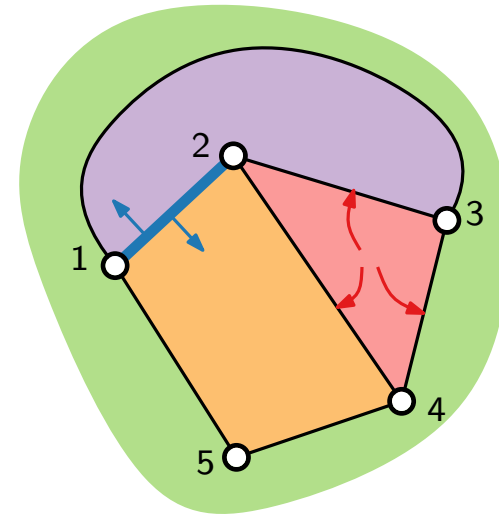
## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

1.  $m \leq 3n - 6$

**Proof.** 1. Every **edge** incident to  $\leq 2$  faces  
Every **face** incident to  $\geq 3$  edges



# Properties of Planar Graphs

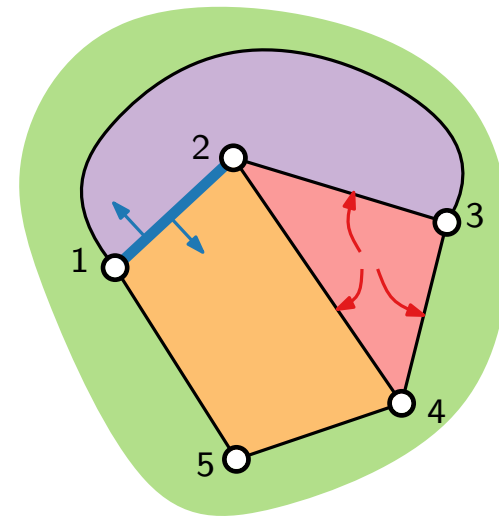
## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

1.  $m \leq 3n - 6$

**Proof.** 1. Every **edge** incident to  $\leq 2$  faces  
 Every **face** incident to  $\geq 3$  edges  
 $\Rightarrow 3f \leq 2m$



# Properties of Planar Graphs

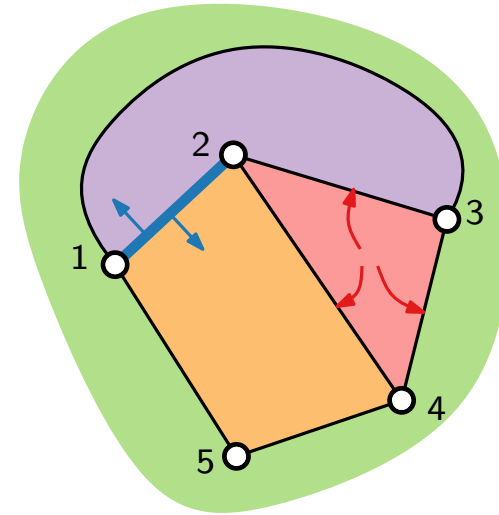
## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

1.  $m \leq 3n - 6$

**Proof.** 1. Every **edge** incident to  $\leq 2$  faces  
 Every **face** incident to  $\geq 3$  edges  
 $\Rightarrow 3f \leq 2m$



# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

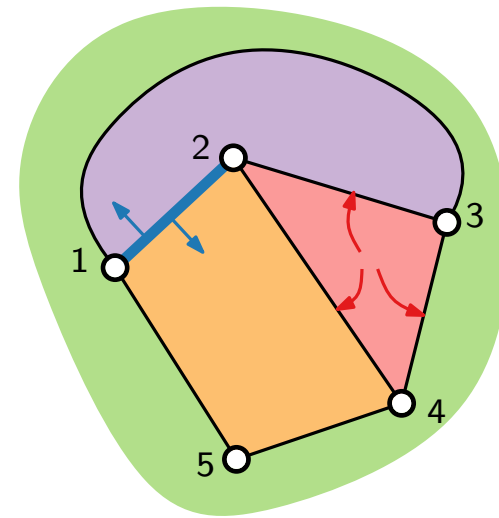
1.  $m \leq 3n - 6$

**Proof.** 1. Every **edge** incident to  $\leq 2$  faces

Every **face** incident to  $\geq 3$  edges

$$\Rightarrow 3f \leq 2m$$

$$\Rightarrow 3c + 3 \leq 3f - 3m + 3n$$



# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

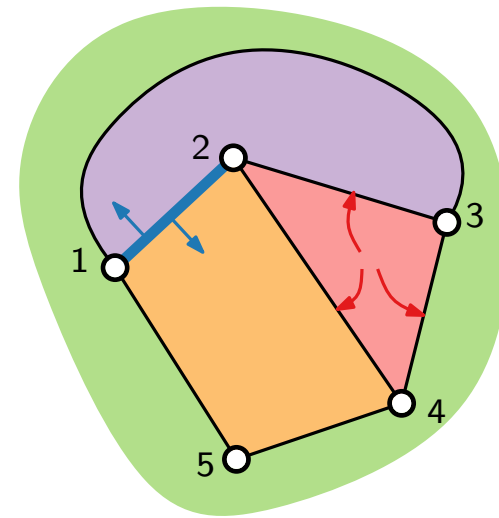
1.  $m \leq 3n - 6$

**Proof.** 1. Every **edge** incident to  $\leq 2$  faces

Every **face** incident to  $\geq 3$  edges

$$\Rightarrow 3f \leq 2m$$

$$\Rightarrow \leq 3c + 3 \leq 3f - 3m + 3n$$



# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

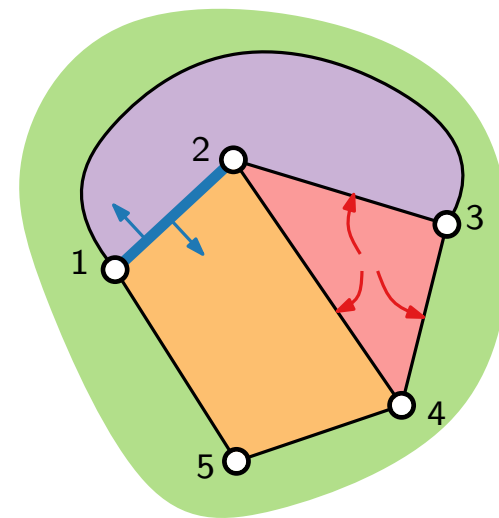
1.  $m \leq 3n - 6$

**Proof.** 1. Every **edge** incident to  $\leq 2$  faces

Every **face** incident to  $\geq 3$  edges

$$\Rightarrow 3f \leq 2m$$

$$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n$$



# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

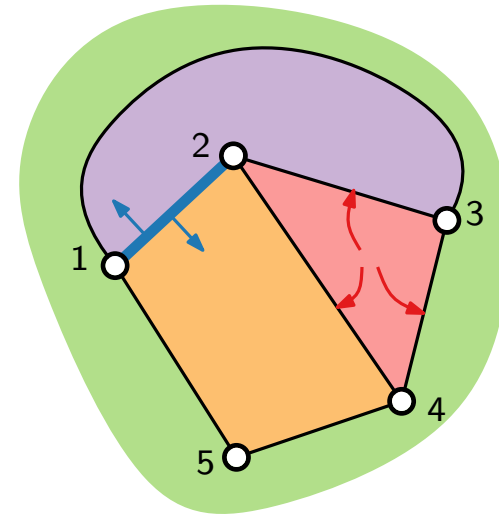
1.  $m \leq 3n - 6$

**Proof.** 1. Every **edge** incident to  $\leq 2$  faces

Every **face** incident to  $\geq 3$  edges

$$\Rightarrow 3f \leq 2m$$

$$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n$$





# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

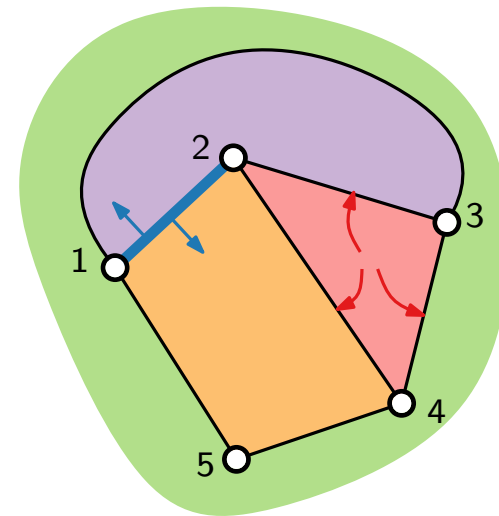
1.  $m \leq 3n - 6$

**Proof.** 1. Every **edge** incident to  $\leq 2$  faces

Every **face** incident to  $\geq 3$  edges

$$\Rightarrow 3f \leq 2m$$

$$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n$$



# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

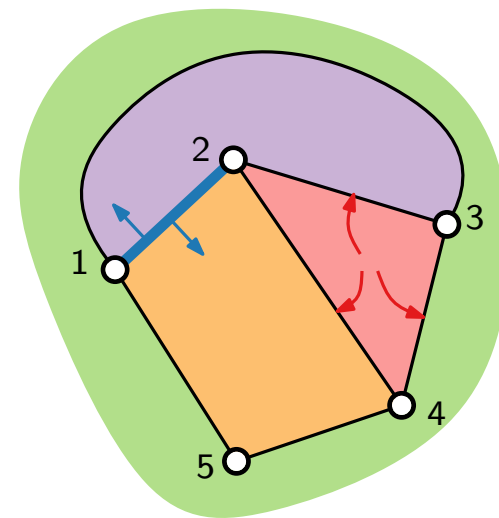
**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

1.  $m \leq 3n - 6$

**Proof.** 1. Every **edge** incident to  $\leq 2$  faces  
Every **face** incident to  $\geq 3$  edges

$$\Rightarrow 3f \leq 2m$$

$$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$$



# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

1.  $m \leq 3n - 6$

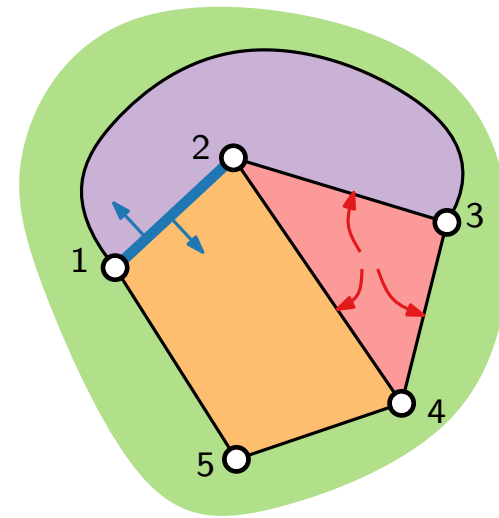
**Proof.** 1. Every **edge** incident to  $\leq 2$  faces

Every **face** incident to  $\geq 3$  edges

$$\Rightarrow 3f \leq 2m$$

$$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$$

$$\Rightarrow m \leq 3n - 6$$



# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

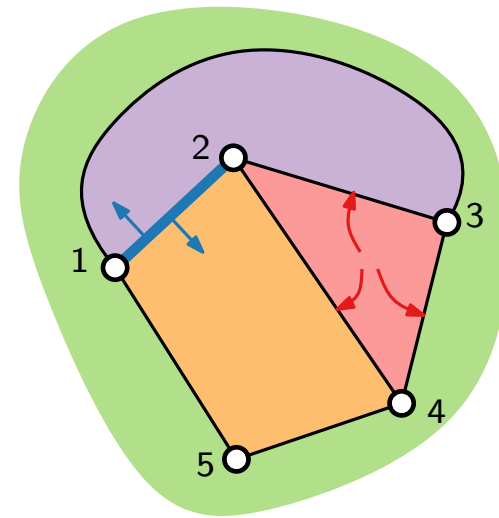
$$1. m \leq 3n - 6 \qquad 2. f \leq 2n - 4$$

**Proof.** 1. Every **edge** incident to  $\leq 2$  faces  
Every **face** incident to  $\geq 3$  edges

$$\Rightarrow 3f \leq 2m$$

$$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$$

$$\Rightarrow m \leq 3n - 6$$



# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

$$1. m \leq 3n - 6 \qquad 2. f \leq 2n - 4$$

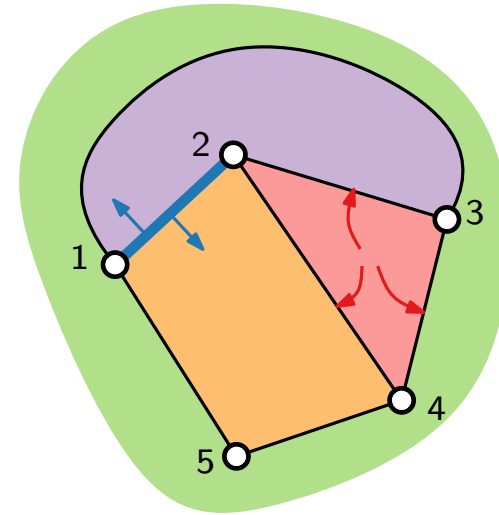
**Proof.** 1. Every **edge** incident to  $\leq 2$  faces  
Every **face** incident to  $\geq 3$  edges

$$\Rightarrow 3f \leq 2m$$

$$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$$

$$\Rightarrow m \leq 3n - 6$$

$$2. 3f \leq 2m$$



# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

$$1. m \leq 3n - 6 \qquad 2. f \leq 2n - 4$$

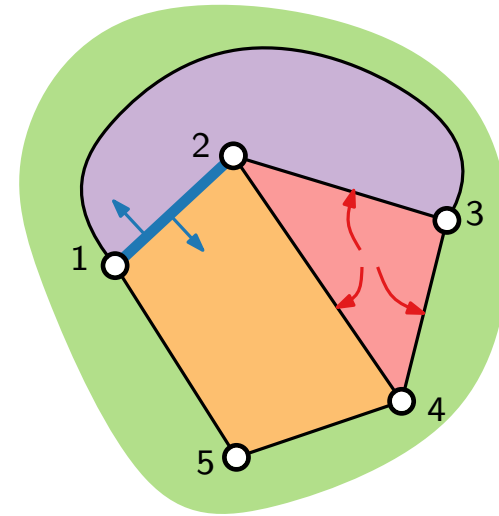
**Proof.** 1. Every **edge** incident to  $\leq 2$  faces  
Every **face** incident to  $\geq 3$  edges

$$\Rightarrow 3f \leq 2m$$

$$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$$

$$\Rightarrow m \leq 3n - 6$$

$$2. 3f \leq 2m \leq 6n - 12$$



# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

$$1. m \leq 3n - 6 \qquad 2. f \leq 2n - 4$$

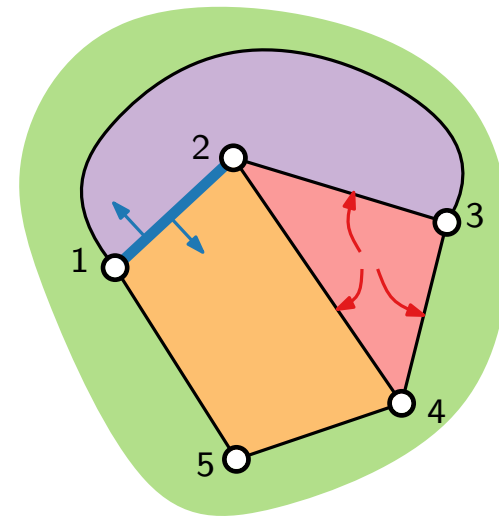
**Proof.** 1. Every **edge** incident to  $\leq 2$  faces  
Every **face** incident to  $\geq 3$  edges

$$\Rightarrow 3f \leq 2m$$

$$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$$

$$\Rightarrow m \leq 3n - 6$$

$$2. 3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$$



# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

1.  $m \leq 3n - 6$
2.  $f \leq 2n - 4$
3. There is a vertex of degree at most 5.

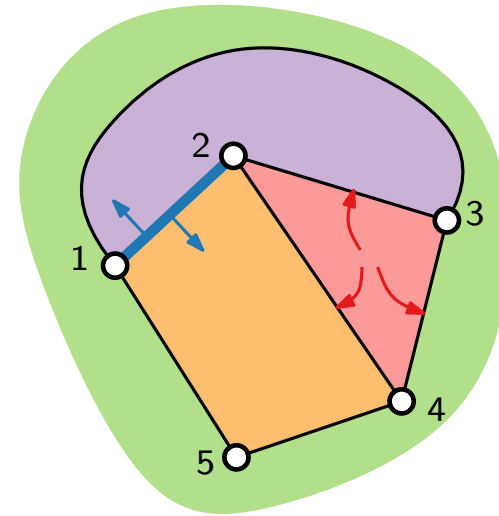
**Proof.** 1. Every **edge** incident to  $\leq 2$  faces  
Every **face** incident to  $\geq 3$  edges

$$\Rightarrow 3f \leq 2m$$

$$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$$

$$\Rightarrow m \leq 3n - 6$$

$$2. \quad 3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$$





# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

1.  $m \leq 3n - 6$
2.  $f \leq 2n - 4$
3. There is a vertex of degree at most 5.

**Proof.** 1. Every **edge** incident to  $\leq 2$  faces  
Every **face** incident to  $\geq 3$  edges

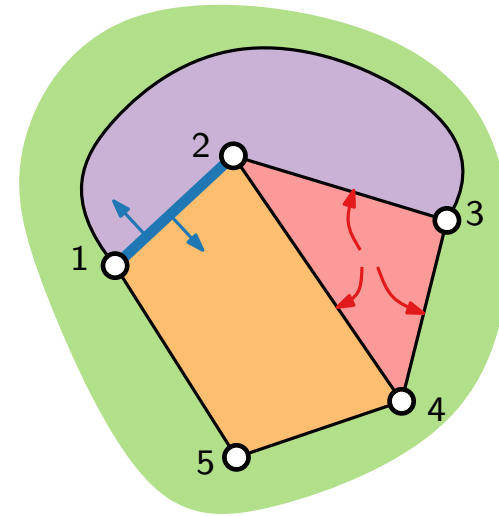
$$\Rightarrow 3f \leq 2m$$

$$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$$

$$\Rightarrow m \leq 3n - 6$$

$$2. \quad 3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$$

$$3. \quad \sum_{v \in V} \deg(v)$$



# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

1.  $m \leq 3n - 6$
2.  $f \leq 2n - 4$
3. There is a vertex of degree at most 5.

**Proof.** 1. Every **edge** incident to  $\leq 2$  faces  
Every **face** incident to  $\geq 3$  edges

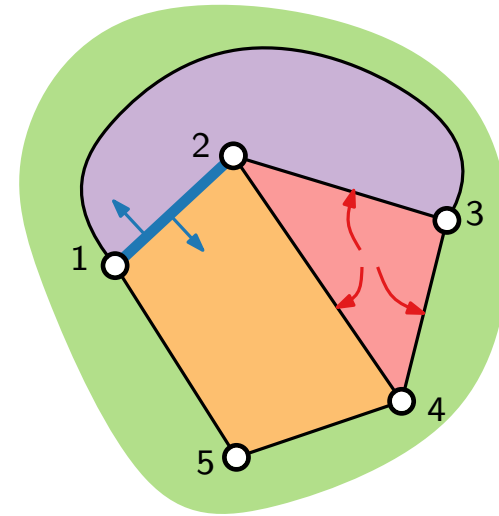
$$\Rightarrow 3f \leq 2m$$

$$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$$

$$\Rightarrow m \leq 3n - 6$$

$$2. \quad 3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$$

$$3. \quad \sum_{v \in V} \deg(v)$$



## Handshaking lemma.

$$\sum_{v \in V} \deg(v) = 2|E|$$

# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

1.  $m \leq 3n - 6$
2.  $f \leq 2n - 4$
3. There is a vertex of degree at most 5.

**Proof.** 1. Every **edge** incident to  $\leq 2$  faces  
Every **face** incident to  $\geq 3$  edges

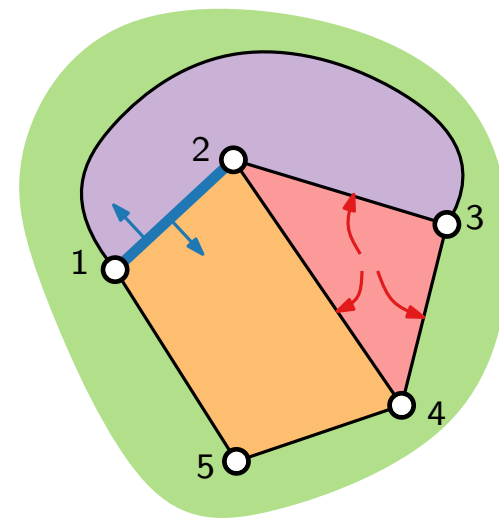
$$\Rightarrow 3f \leq 2m$$

$$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$$

$$\Rightarrow m \leq 3n - 6$$

$$2. \quad 3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$$

$$3. \quad \sum_{v \in V} \deg(v) = 2m$$



## Handshaking lemma.

$$\sum_{v \in V} \deg(v) = 2|E|$$

# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

1.  $m \leq 3n - 6$
2.  $f \leq 2n - 4$
3. There is a vertex of degree at most 5.

**Proof.** 1. Every **edge** incident to  $\leq 2$  faces  
Every **face** incident to  $\geq 3$  edges

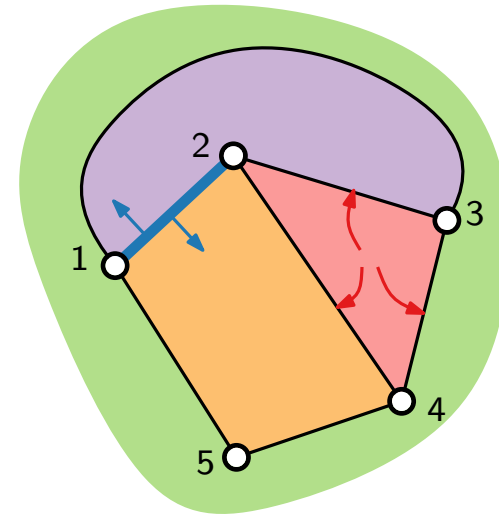
$$\Rightarrow 3f \leq 2m$$

$$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$$

$$\Rightarrow m \leq 3n - 6$$

$$2. \quad 3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$$

$$3. \quad \sum_{v \in V} \deg(v) = 2m \leq 6n - 12$$



## Handshaking lemma.

$$\sum_{v \in V} \deg(v) = 2|E|$$

# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

1.  $m \leq 3n - 6$
2.  $f \leq 2n - 4$
3. There is a vertex of degree at most 5.

**Proof.** 1. Every **edge** incident to  $\leq 2$  faces  
Every **face** incident to  $\geq 3$  edges

$$\Rightarrow 3f \leq 2m$$

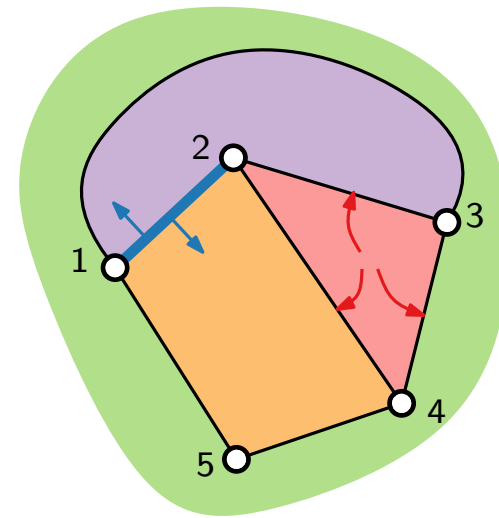
$$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$$

$$\Rightarrow m \leq 3n - 6$$

$$2. \quad 3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$$

$$3. \quad \sum_{v \in V} \deg(v) = 2m \leq 6n - 12$$

$$\Rightarrow \min_{v \in V} \deg(v)$$



## Handshaking lemma.

$$\sum_{v \in V} \deg(v) = 2|E|$$

# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

1.  $m \leq 3n - 6$
2.  $f \leq 2n - 4$
3. There is a vertex of degree at most 5.

**Proof.** 1. Every **edge** incident to  $\leq 2$  faces  
Every **face** incident to  $\geq 3$  edges

$$\Rightarrow 3f \leq 2m$$

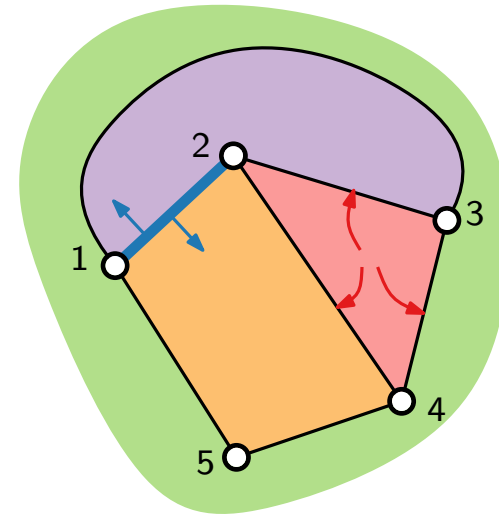
$$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$$

$$\Rightarrow m \leq 3n - 6$$

$$2. \quad 3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$$

$$3. \quad \sum_{v \in V} \deg(v) = 2m \leq 6n - 12$$

$$\Rightarrow \min_{v \in V} \deg(v) \leq \text{average degree}(G)$$



## Handshaking lemma.

$$\sum_{v \in V} \deg(v) = 2|E|$$

# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

1.  $m \leq 3n - 6$
2.  $f \leq 2n - 4$
3. There is a vertex of degree at most 5.

**Proof.** 1. Every **edge** incident to  $\leq 2$  faces  
Every **face** incident to  $\geq 3$  edges

$$\Rightarrow 3f \leq 2m$$

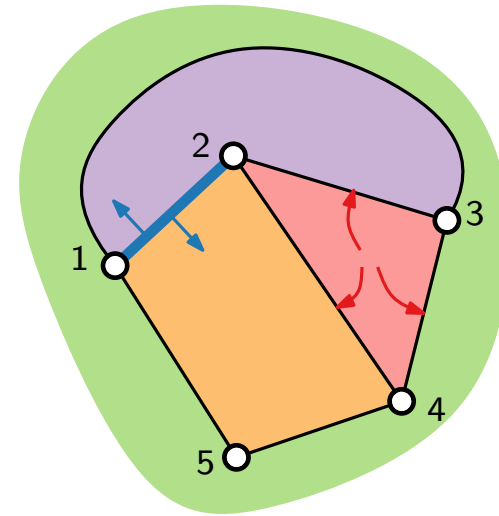
$$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$$

$$\Rightarrow m \leq 3n - 6$$

$$2. \quad 3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$$

$$3. \quad \sum_{v \in V} \deg(v) = 2m \leq 6n - 12$$

$$\Rightarrow \min_{v \in V} \deg(v) \leq \text{average degree}(G) = 1/n \sum_{v \in V} \deg(v)$$



## Handshaking lemma.

$$\sum_{v \in V} \deg(v) = 2|E|$$

# Properties of Planar Graphs

## Euler's polyhedra formula.

$$\begin{array}{ccccccc} \# \text{faces} & - & \# \text{edges} & + & \# \text{vertices} & = & \# \text{conn.comp.} + 1 \\ f & - & m & + & n & = & c + 1 \end{array}$$

**Theorem.**  $G$  simple planar graph with  $n \geq 3$  vtc.

1.  $m \leq 3n - 6$
2.  $f \leq 2n - 4$
3. There is a vertex of degree at most 5.

**Proof.** 1. Every **edge** incident to  $\leq 2$  faces  
Every **face** incident to  $\geq 3$  edges

$$\Rightarrow 3f \leq 2m$$

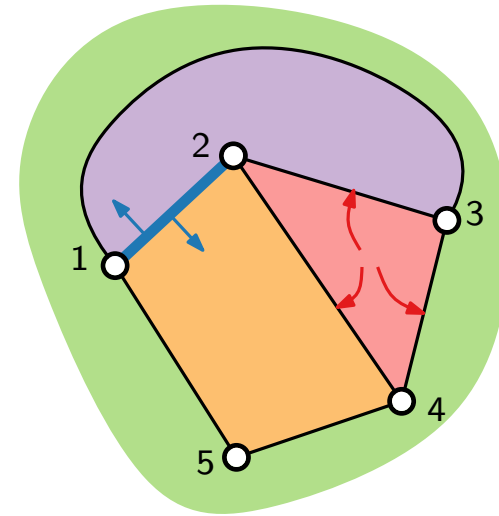
$$\Rightarrow 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n = 3n - m$$

$$\Rightarrow m \leq 3n - 6$$

$$2. \quad 3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$$

$$3. \quad \sum_{v \in V} \deg(v) = 2m \leq 6n - 12$$

$$\Rightarrow \min_{v \in V} \deg(v) \leq \text{average degree}(G) = 1/n \sum_{v \in V} \deg(v) < 6$$



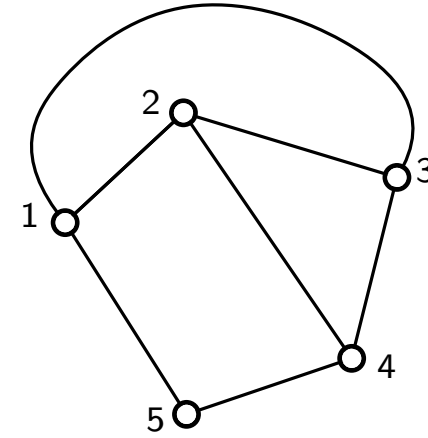
## Handshaking lemma.

$$\sum_{v \in V} \deg(v) = 2|E|$$



# Triangulations

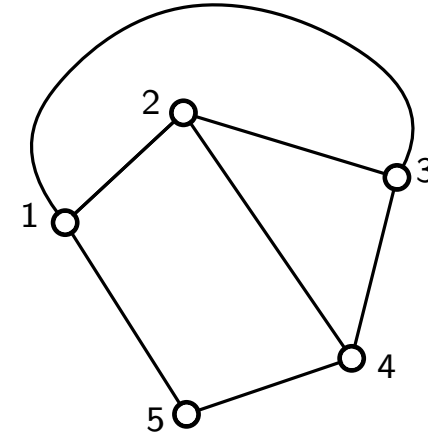
A **plane triangulation** is a plane graph where every face is a triangle.



# Triangulations

with planar embedding

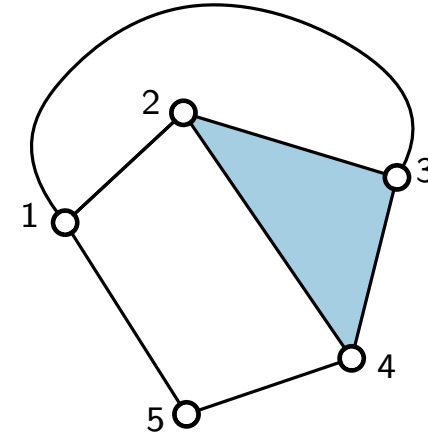
A **plane triangulation** is a plane graph where every face is a triangle.



# Triangulations

with planar embedding

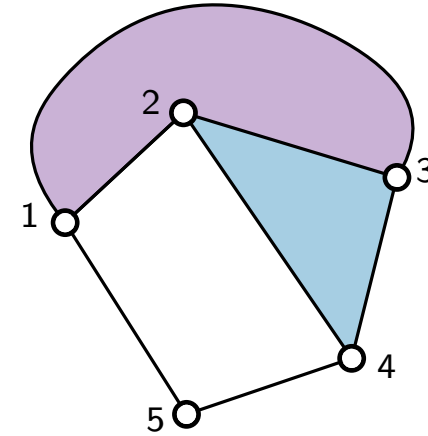
A **plane triangulation** is a plane graph where every face is a triangle.



# Triangulations

with planar embedding

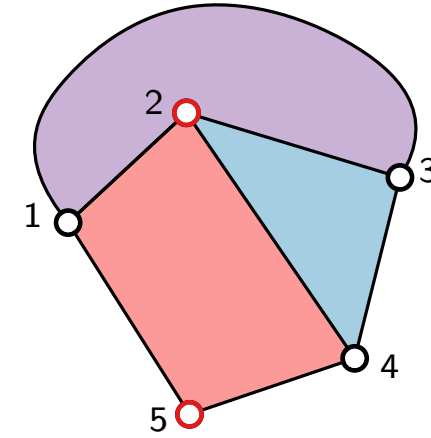
A **plane triangulation** is a plane graph where every face is a triangle.



# Triangulations

with planar embedding

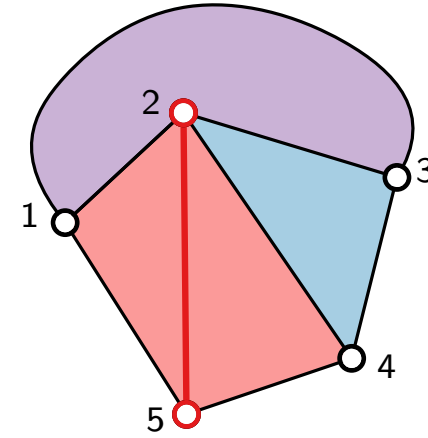
A **plane triangulation** is a plane graph where every face is a triangle.



# Triangulations

with planar embedding

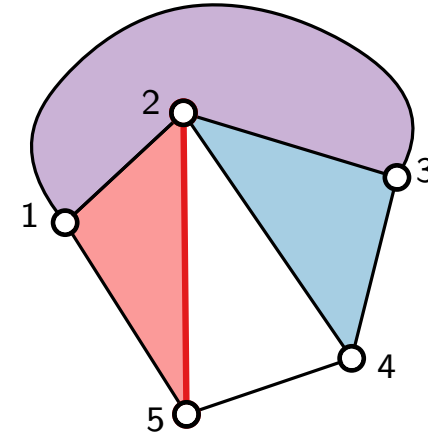
A **plane triangulation** is a plane graph where every face is a triangle.



# Triangulations

with planar embedding

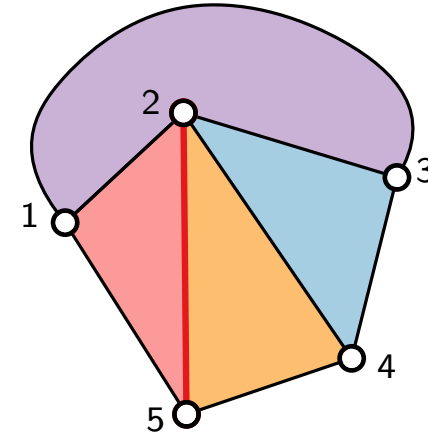
A **plane triangulation** is a plane graph where every face is a triangle.



# Triangulations

with planar embedding

A **plane triangulation** is a plane graph where every face is a triangle.

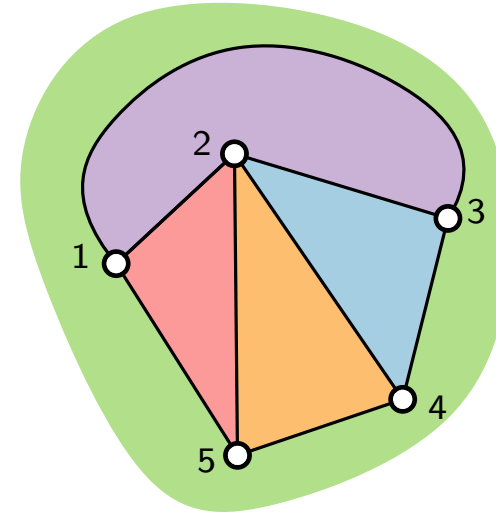




# Triangulations

with planar embedding

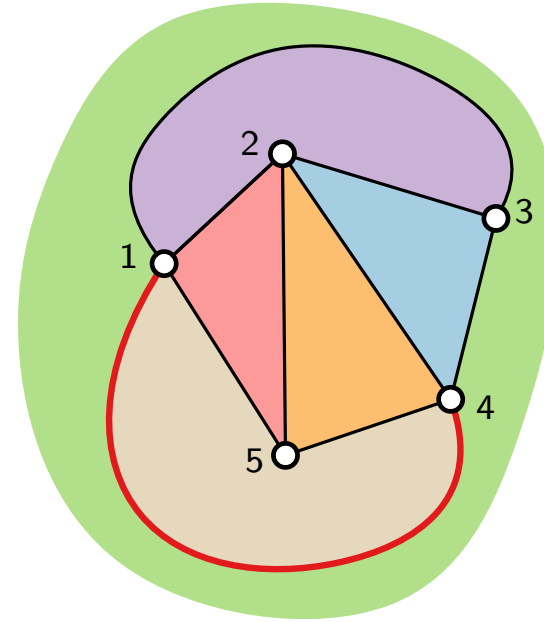
A **plane triangulation** is a plane graph where every face is a triangle.



# Triangulations

with planar embedding

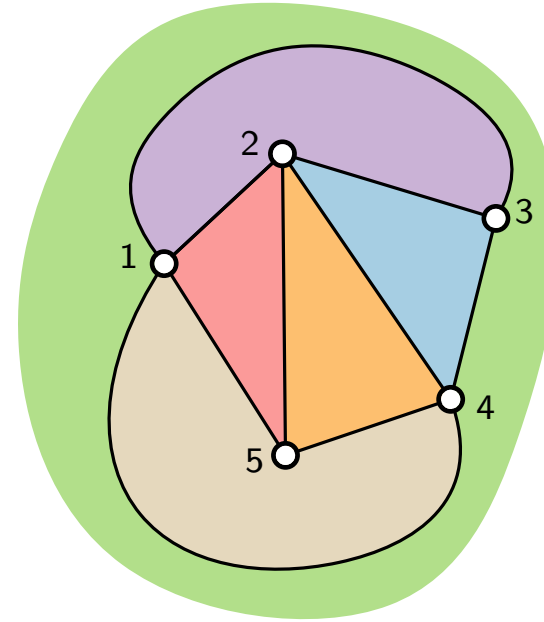
A **plane triangulation** is a plane graph where every face is a triangle.



# Triangulations

with planar embedding

A **plane (inner) triangulation** is a plane graph where every (inner) face is a triangle.

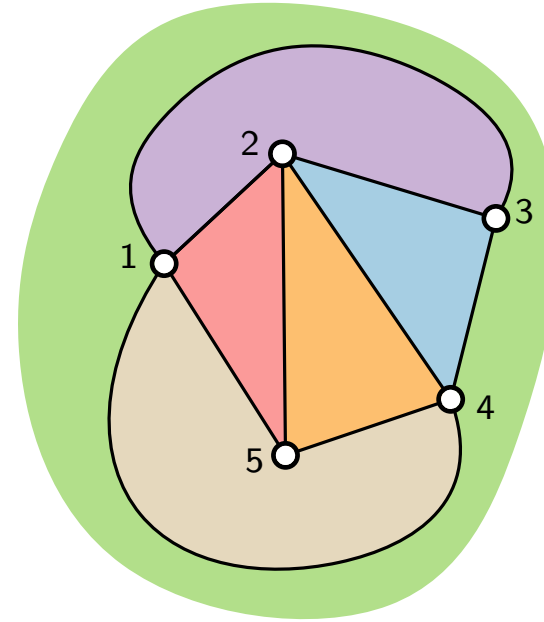


# Triangulations

with planar embedding

A **plane (inner) triangulation** is a plane graph where every (inner) face is a triangle.

A **maximal planar graph** is a planar graph where adding any edge would violate planarity.

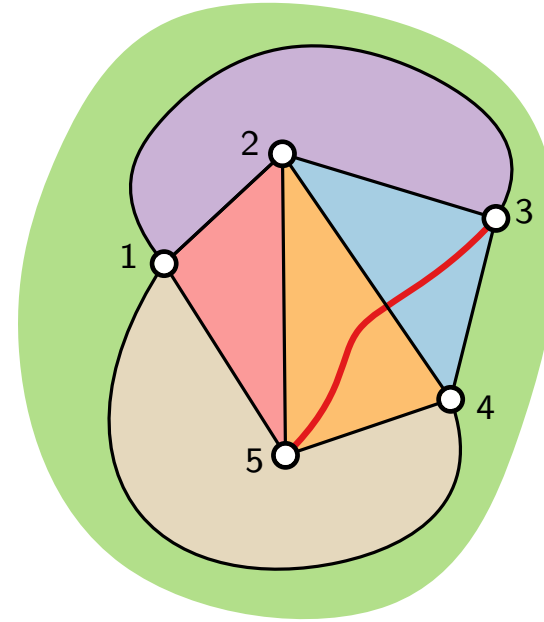


# Triangulations

with planar embedding

A **plane (inner) triangulation** is a plane graph where every (inner) face is a triangle.

A **maximal planar graph** is a planar graph where adding any edge would violate planarity.

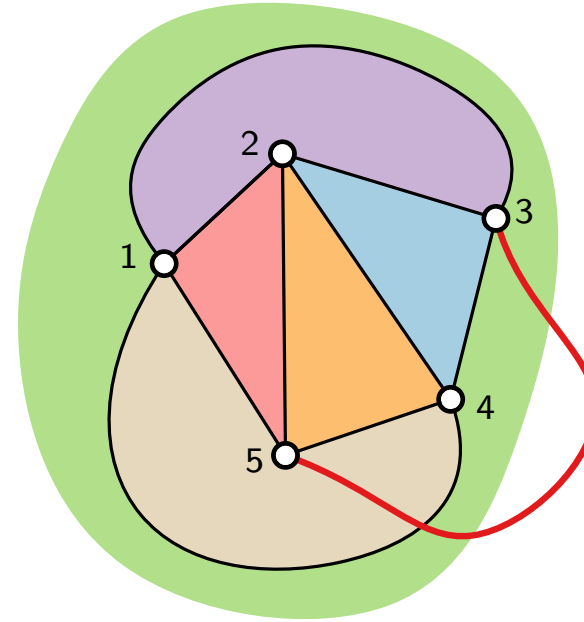


# Triangulations

with planar embedding

A **plane (inner) triangulation** is a plane graph where every (inner) face is a triangle.

A **maximal planar graph** is a planar graph where adding any edge would violate planarity.

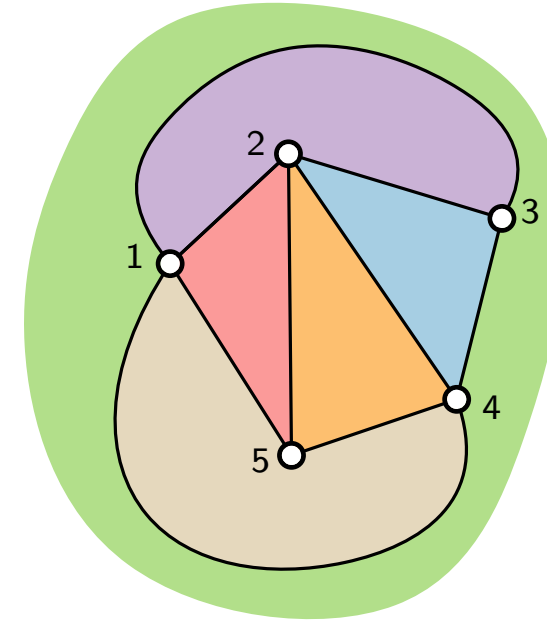


# Triangulations

with planar embedding

A **plane (inner) triangulation** is a plane graph where every (inner) face is a triangle.

A **maximal planar graph** is a planar graph where adding any edge would violate planarity.



# Triangulations

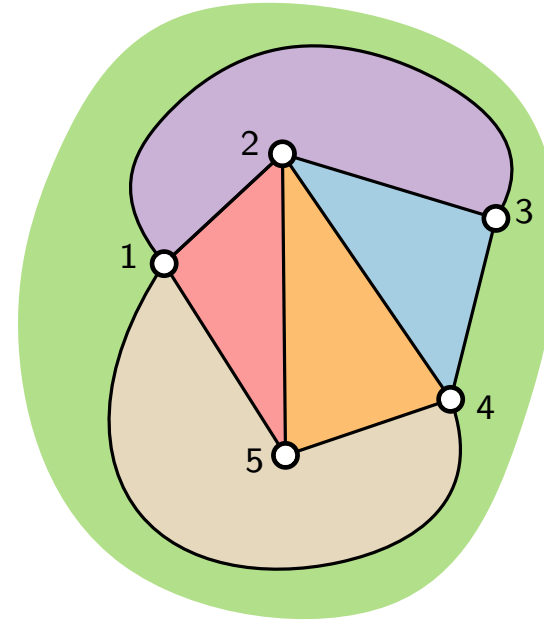
with planar embedding

A **plane (inner) triangulation** is a plane graph where every (inner) face is a triangle.

A **maximal planar graph** is a planar graph where adding any edge would violate planarity.

## Observation.

A maximal plane graph is a plane triangulation.





# Triangulations

with planar embedding

A **plane (inner) triangulation** is a plane graph where every (inner) face is a triangle.

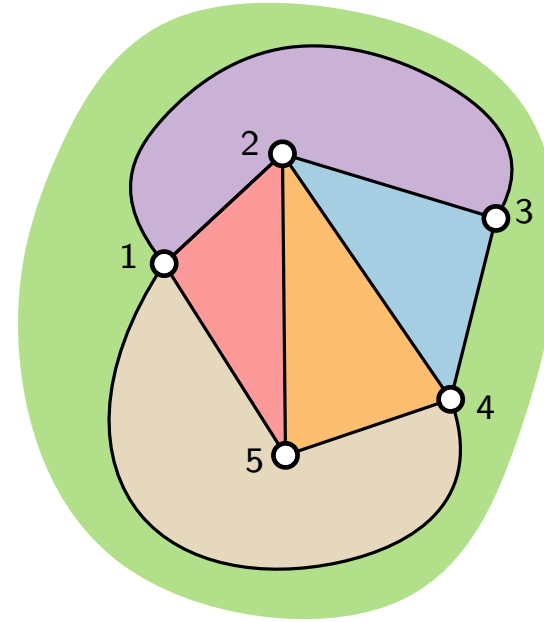
A **maximal planar graph** is a planar graph where adding any edge would violate planarity.

## Observation.

A maximal plane graph is a plane triangulation.

## Lemma.

A plane triangulation is at least 3-connected and thus has a unique planar embedding.



# Triangulations

with planar embedding

A **plane (inner) triangulation** is a plane graph where every (inner) face is a triangle.

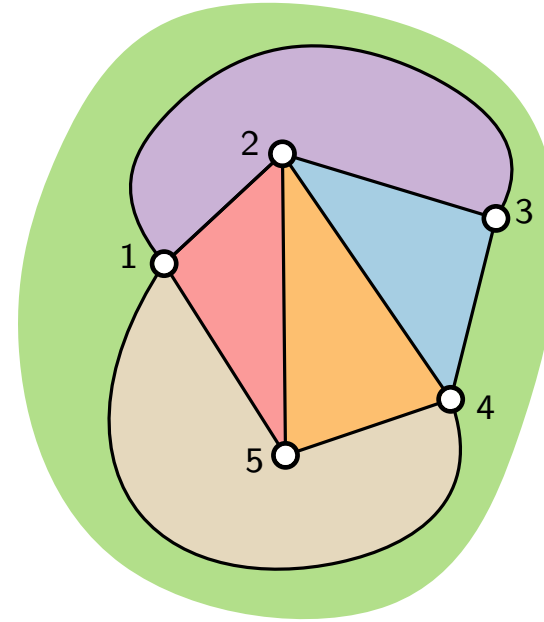
A **maximal planar graph** is a planar graph where adding any edge would violate planarity.

## Observation.

A maximal plane graph is a plane triangulation.

## Lemma.

A plane triangulation is at least 3-connected and thus has a unique planar embedding.



We focus on plane triangulations:

# Triangulations

with planar embedding

A **plane (inner) triangulation** is a plane graph where every (inner) face is a triangle.

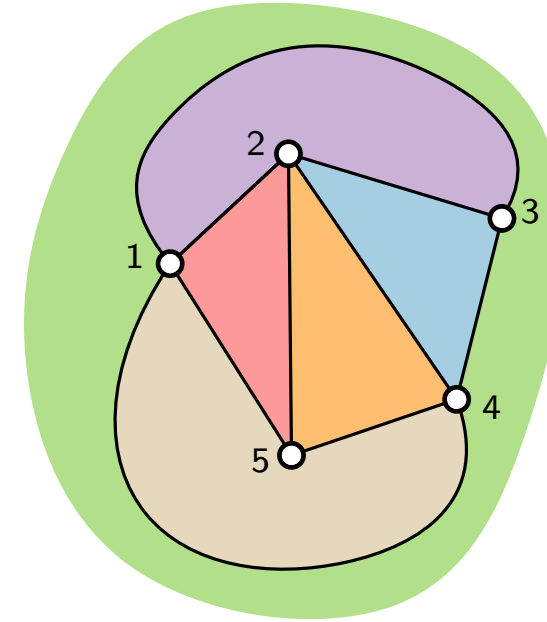
A **maximal planar graph** is a planar graph where adding any edge would violate planarity.

## Observation.

A maximal plane graph is a plane triangulation.

## Lemma.

A plane triangulation is at least 3-connected and thus has a unique planar embedding.



We focus on plane triangulations:

## Lemma.

Every plane graph is subgraph of a plane triangulation.

# Triangulations

with planar embedding

A **plane (inner) triangulation** is a plane graph where every (inner) face is a triangle.

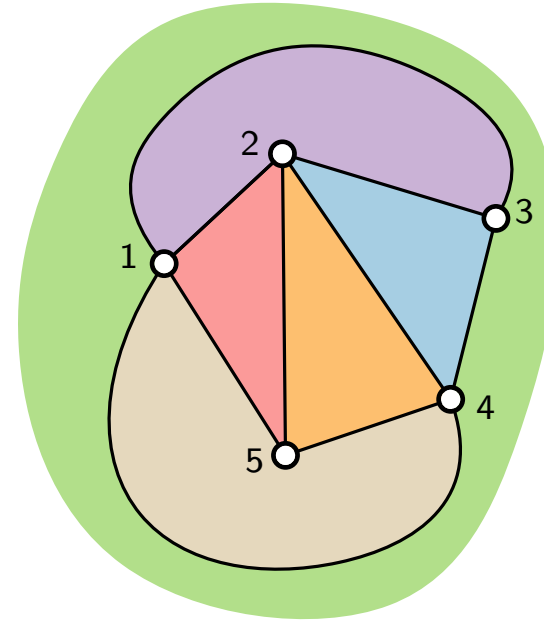
A **maximal planar graph** is a planar graph where adding any edge would violate planarity.

## Observation.

A maximal plane graph is a plane triangulation.

## Lemma.

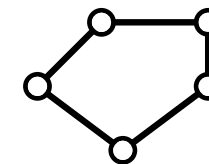
A plane triangulation is at least 3-connected and thus has a unique planar embedding.



We focus on plane triangulations:

## Lemma.

Every plane graph is subgraph of a plane triangulation.



# Triangulations

with planar embedding

A **plane (inner) triangulation** is a plane graph where every (inner) face is a triangle.

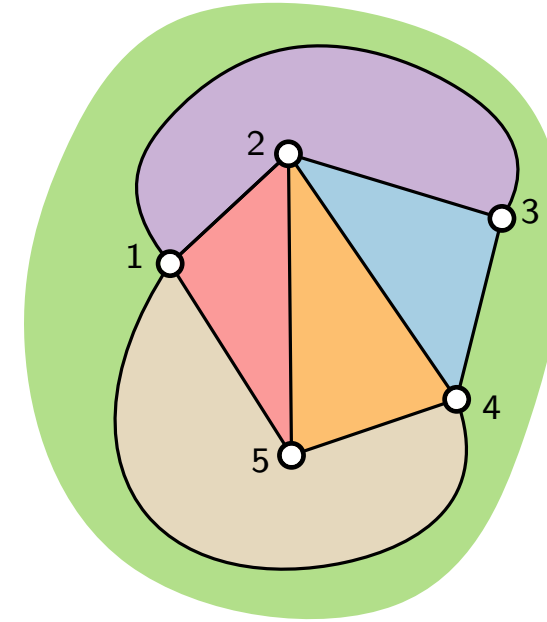
A **maximal planar graph** is a planar graph where adding any edge would violate planarity.

## Observation.

A maximal plane graph is a plane triangulation.

## Lemma.

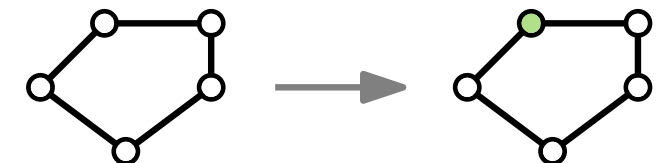
A plane triangulation is at least 3-connected and thus has a unique planar embedding.



We focus on plane triangulations:

## Lemma.

Every plane graph is subgraph of a plane triangulation.



# Triangulations

with planar embedding

A **plane (inner) triangulation** is a plane graph where every (inner) face is a triangle.

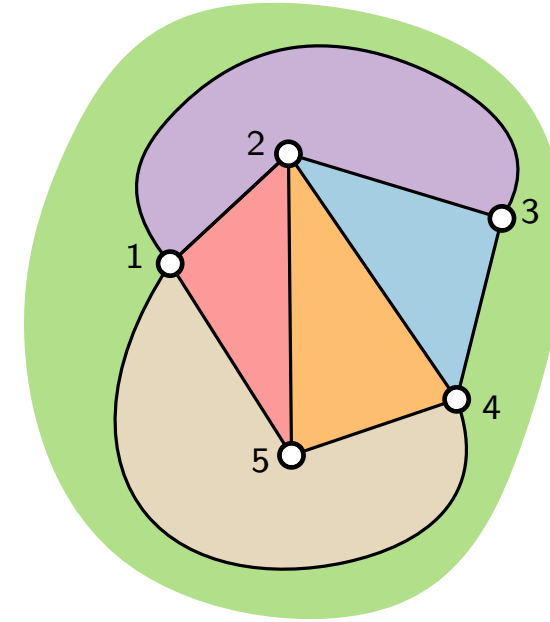
A **maximal planar graph** is a planar graph where adding any edge would violate planarity.

## Observation.

A maximal plane graph is a plane triangulation.

## Lemma.

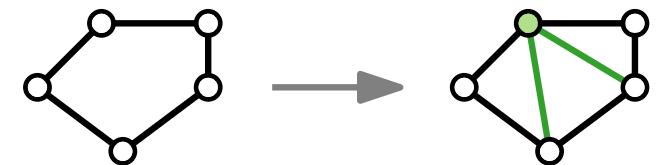
A plane triangulation is at least 3-connected and thus has a unique planar embedding.



We focus on plane triangulations:

## Lemma.

Every plane graph is subgraph of a plane triangulation.



# Triangulations

with planar embedding

A **plane (inner) triangulation** is a plane graph where every (inner) face is a triangle.

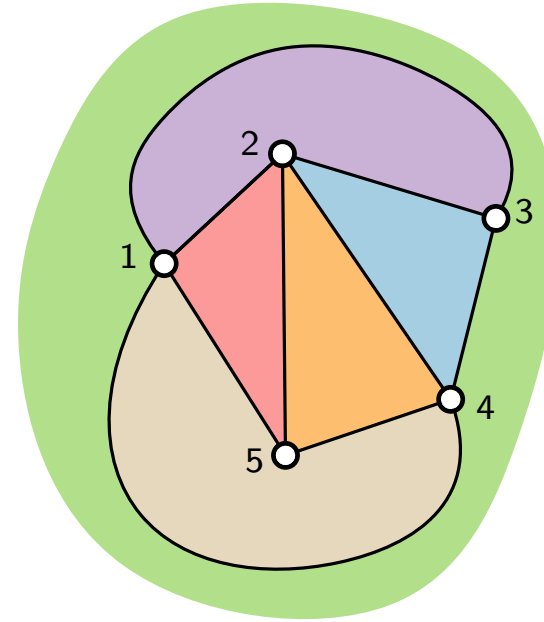
A **maximal planar graph** is a planar graph where adding any edge would violate planarity.

## Observation.

A maximal plane graph is a plane triangulation.

## Lemma.

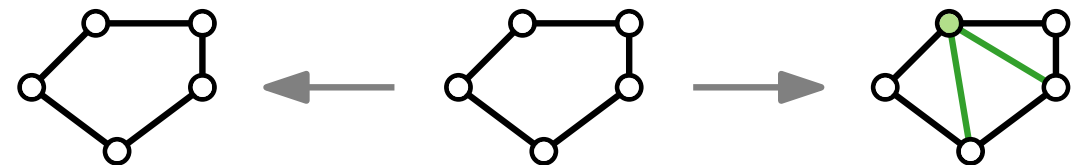
A plane triangulation is at least 3-connected and thus has a unique planar embedding.



We focus on plane triangulations:

## Lemma.

Every plane graph is subgraph of a plane triangulation.



# Triangulations

with planar embedding

A **plane (inner) triangulation** is a plane graph where every (inner) face is a triangle.

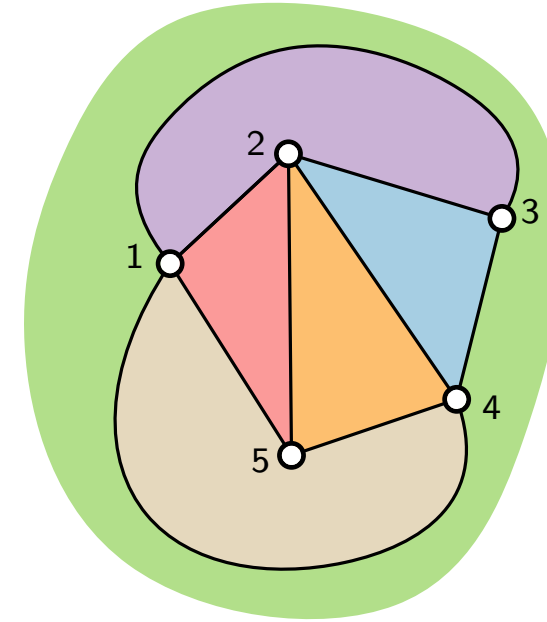
A **maximal planar graph** is a planar graph where adding any edge would violate planarity.

## Observation.

A maximal plane graph is a plane triangulation.

## Lemma.

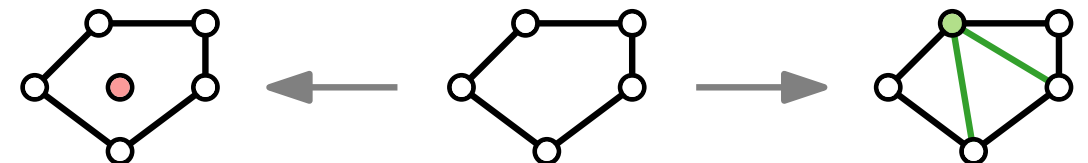
A plane triangulation is at least 3-connected and thus has a unique planar embedding.



We focus on plane triangulations:

## Lemma.

Every plane graph is subgraph of a plane triangulation.





# Triangulations

with planar embedding

A **plane (inner) triangulation** is a plane graph where every (inner) face is a triangle.

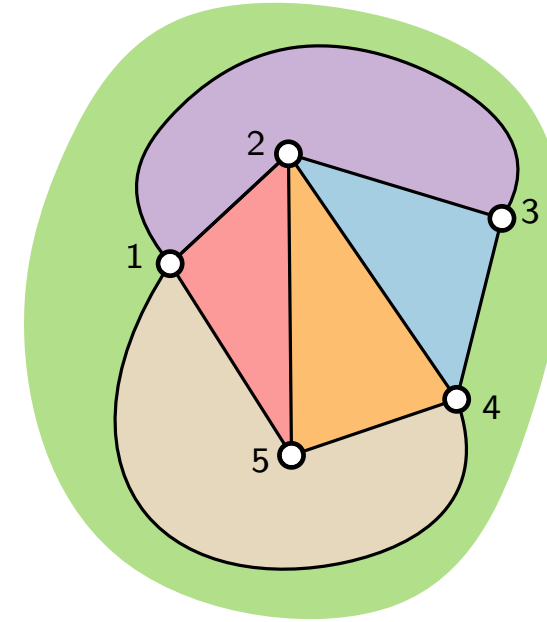
A **maximal planar graph** is a planar graph where adding any edge would violate planarity.

## Observation.

A maximal plane graph is a plane triangulation.

## Lemma.

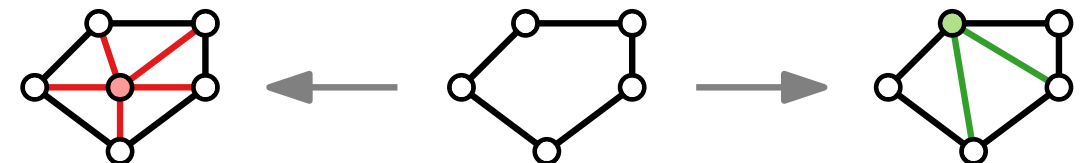
A plane triangulation is at least 3-connected and thus has a unique planar embedding.



We focus on plane triangulations:

## Lemma.

Every plane graph is subgraph of a plane triangulation.



# Triangulations

with planar embedding

A **plane (inner) triangulation** is a plane graph where every (inner) face is a triangle.

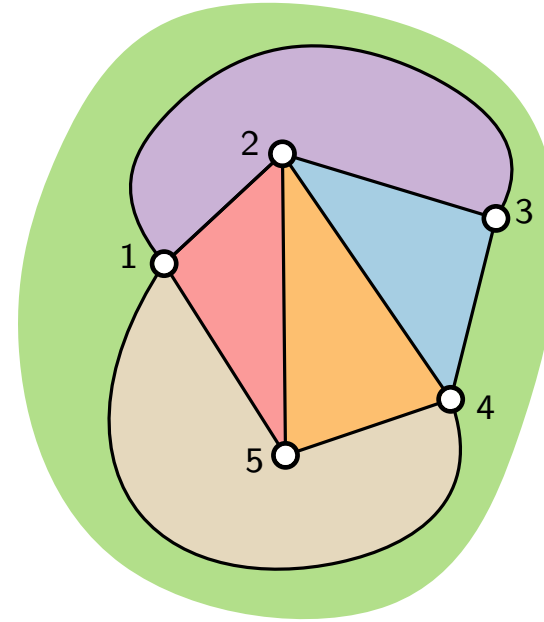
A **maximal planar graph** is a planar graph where adding any edge would violate planarity.

## Observation.

A maximal plane graph is a plane triangulation.

## Lemma.

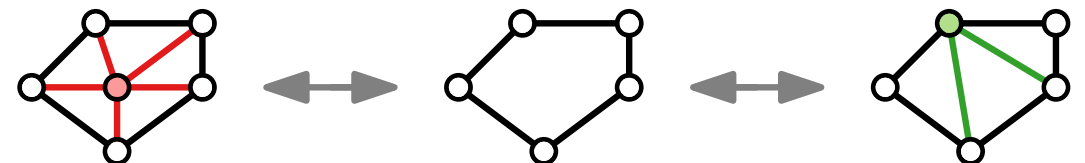
A plane triangulation is at least 3-connected and thus has a unique planar embedding.



We focus on plane triangulations:

## Lemma.

Every plane graph is subgraph of a plane triangulation.



# Motivation

- Why planar and straight-line?

# Motivation

- Why planar and straight-line?

[Bennett, Ryall, Spaltzholz and Gooch '07]

## 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

- Why planar and straight-line?

[Bennett, Ryall, Spaltzholz and Gooch '07]

## 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

- Why planar and straight-line?

[Bennett, Ryall, Spaltzholz and Gooch '07]

## 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

- Why planar and straight-line?

[Bennett, Ryall, Spaltzholz and Gooch '07]

## 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.

### Drawing conventions

- No crossings  $\Rightarrow$  planar
- No bends  $\Rightarrow$  straight-line

# Motivation

- Why planar and straight-line?

[Bennett, Ryall, Spaltzholz and Gooch '07]

## 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.

### Drawing conventions

- No crossings  $\Rightarrow$  planar
- No bends  $\Rightarrow$  straight-line

### Drawing aesthetics

- Area



# Towards Straight-Line Drawings

# Towards Straight-Line Drawings

**Characterization**

# Towards Straight-Line Drawings

**Characterization**

**Recognition**

# Towards Straight-Line Drawings

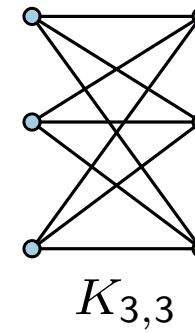
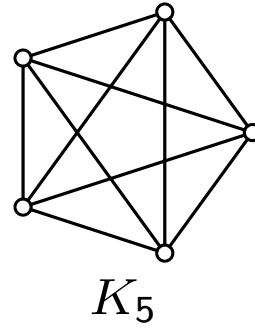
**Characterization**

**Recognition**

**Drawing**

# Towards Straight-Line Drawings

**Theorem.** [Kuratowski 1930]  
 $G$  planar  $\Leftrightarrow$   
neither  $K_5$  nor  $K_{3,3}$  minor of  $G$



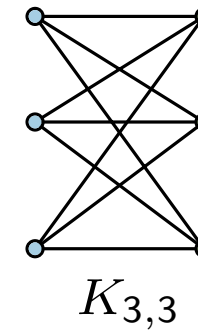
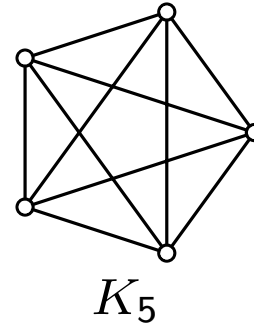
Characterization

Recognition

Drawing

# Towards Straight-Line Drawings

**Theorem.** [Kuratowski 1930]  
 $G$  planar  $\Leftrightarrow$   
neither  $K_5$  nor  $K_{3,3}$  minor of  $G$



**Characterization**

**Theorem.** [Hopcroft & Tarjan 1974]  
Let  $G$  be a graph with  $n$  vertices. There is an  
 $\mathcal{O}(n)$ -time algorithm to test whether  $G$  is planar.

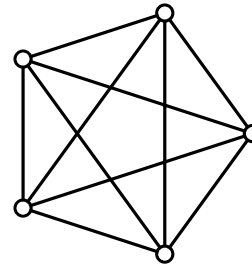
**Recognition**

**Drawing**

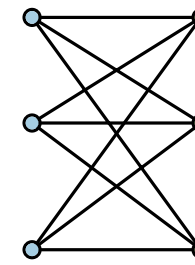
# Towards Straight-Line Drawings

**Theorem.** [Kuratowski 1930]

$G$  planar  $\Leftrightarrow$   
neither  $K_5$  nor  $K_{3,3}$  minor of  $G$



$K_5$



$K_{3,3}$

**Characterization**

**Theorem.** [Hopcroft & Tarjan 1974]

Let  $G$  be a graph with  $n$  vertices. There is an  $\mathcal{O}(n)$ -time algorithm to test whether  $G$  is planar.

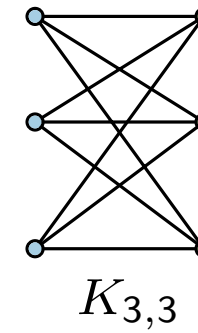
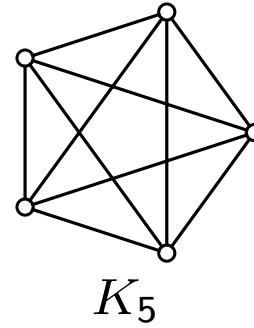
**Recognition**

Also computes a planar embedding in  $\mathcal{O}(n)$  time.

**Drawing**

# Towards Straight-Line Drawings

**Theorem.** [Kuratowski 1930]  
 $G$  planar  $\Leftrightarrow$   
neither  $K_5$  nor  $K_{3,3}$  minor of  $G$



Characterization

**Theorem.** [Hopcroft & Tarjan 1974]  
Let  $G$  be a graph with  $n$  vertices. There is an  
 $\mathcal{O}(n)$ -time algorithm to test whether  $G$  is planar.

Recognition

Also computes a planar embedding in  $\mathcal{O}(n)$  time.

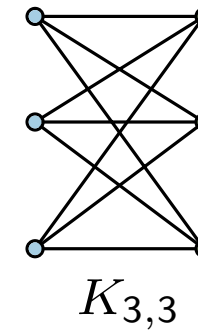
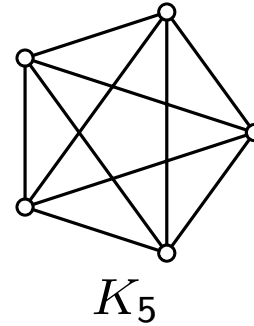
**Theorem.** [Wagner 1936, Fáry 1948, Stein 1951]  
Every planar graph has a planar drawing  
where the edges are straight-line segments.

Drawing



# Towards Straight-Line Drawings

**Theorem.** [Kuratowski 1930]  
 $G$  planar  $\Leftrightarrow$   
neither  $K_5$  nor  $K_{3,3}$  minor of  $G$



Characterization

**Theorem.** [Hopcroft & Tarjan 1974]  
Let  $G$  be a graph with  $n$  vertices. There is an  
 $\mathcal{O}(n)$ -time algorithm to test whether  $G$  is planar.

Recognition

Also computes a planar embedding in  $\mathcal{O}(n)$  time.

**Theorem.** [Wagner 1936, Fáry 1948, Stein 1951]  
Every planar graph has a planar drawing  
where the edges are straight-line segments.

Drawing

The algorithms implied by these theorems produce drawings  
whose area is **not** bounded by any polynomial in  $n$ .

# Planar straight-line drawings

**Theorem.** [De Fraysseix, Pach, Pollack '90]

Every  $n$ -vertex planar graph has a planar straight-line drawing of size  $(2n - 4) \times (n - 2)$ .

**Theorem.** [Schnyder '90]

Every  $n$ -vertex planar graph has a planar straight-line drawing of size  $(n - 2) \times (n - 2)$ .

# Planar straight-line drawings

**Theorem.** [De Fraysseix, Pach, Pollack '90]

Every  $n$ -vertex planar graph has a planar straight-line drawing of size  $(2n - 4) \times (n - 2)$ .

**Theorem.** [Schnyder '90]

Every  $n$ -vertex planar graph has a planar straight-line drawing of size  $(n - 2) \times (n - 2)$ .

# Planar straight-line drawings

**Theorem.** [De Fraysseix, Pach, Pollack '90]

Every  $n$ -vertex planar graph has a planar straight-line drawing of size  $(2n - 4) \times (n - 2)$ .

**Idea.**

**Theorem.** [Schnyder '90]

Every  $n$ -vertex planar graph has a planar straight-line drawing of size  $(n - 2) \times (n - 2)$ .

# Planar straight-line drawings

**Theorem.** [De Fraysseix, Pach, Pollack '90]

Every  $n$ -vertex planar graph has a planar straight-line drawing of size  $(2n - 4) \times (n - 2)$ .

**Idea.**

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



**Theorem.** [Schnyder '90]

Every  $n$ -vertex planar graph has a planar straight-line drawing of size  $(n - 2) \times (n - 2)$ .

# Planar straight-line drawings

**Theorem.** [De Fraysseix, Pach, Pollack '90]

Every  $n$ -vertex planar graph has a planar straight-line drawing of size  $(2n - 4) \times (n - 2)$ .

**Idea.**

- Start with single 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  $G_i$ .



**Theorem.** [Schnyder '90]

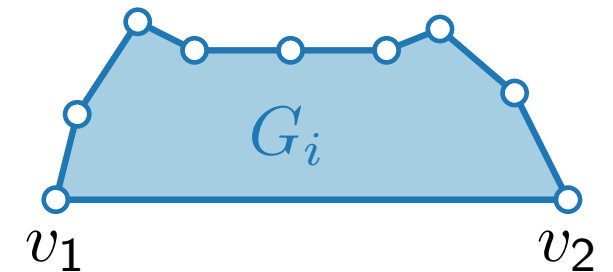
Every  $n$ -vertex planar graph has a planar straight-line drawing of size  $(n - 2) \times (n - 2)$ .

# Planar straight-line drawings

**Theorem.** [De Fraysseix, Pach, Pollack '90]  
Every  $n$ -vertex planar graph has a planar straight-line drawing of size  $(2n - 4) \times (n - 2)$ .

## Idea.

- Start with single 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  $G_i$ .



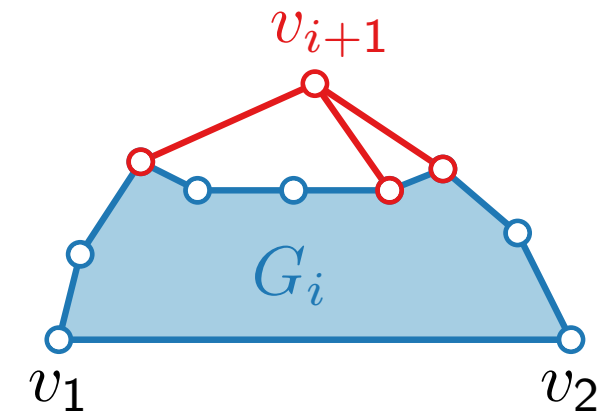
**Theorem.** [Schneider '90]  
Every  $n$ -vertex planar graph has a planar straight-line drawing of size  $(n - 2) \times (n - 2)$ .

# Planar straight-line drawings

**Theorem.** [De Fraysseix, Pach, Pollack '90]  
Every  $n$ -vertex planar graph has a planar straight-line drawing of size  $(2n - 4) \times (n - 2)$ .

## Idea.

- Start with single 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  $G_i$ .



**Theorem.** [Schneider '90]  
Every  $n$ -vertex planar graph has a planar straight-line drawing of size  $(n - 2) \times (n - 2)$ .



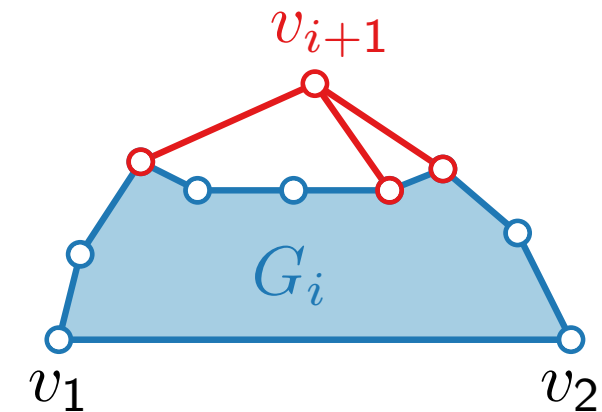
# Planar straight-line drawings

**Theorem.** [De Fraysseix, Pach, Pollack '90]

Every  $n$ -vertex planar graph has a planar straight-line drawing of size  $(2n - 4) \times (n - 2)$ .

**Idea.**

- Start with single 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.



**Theorem.** [Schnyder '90]

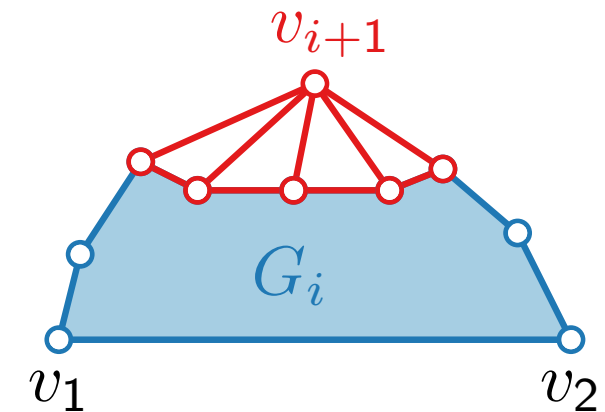
Every  $n$ -vertex planar graph has a planar straight-line drawing of size  $(n - 2) \times (n - 2)$ .

# Planar straight-line drawings

**Theorem.** [De Fraysseix, Pach, Pollack '90]  
Every  $n$ -vertex planar graph has a planar straight-line drawing of size  $(2n - 4) \times (n - 2)$ .

## Idea.

- Start with single 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.

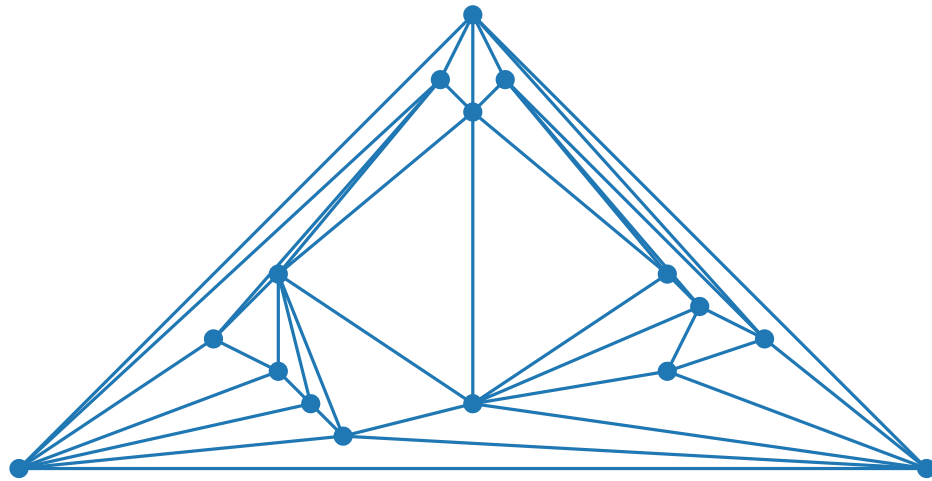


**Theorem.** [Schneider '90]  
Every  $n$ -vertex planar graph has a planar straight-line drawing of size  $(n - 2) \times (n - 2)$ .

# Visualization of Graphs

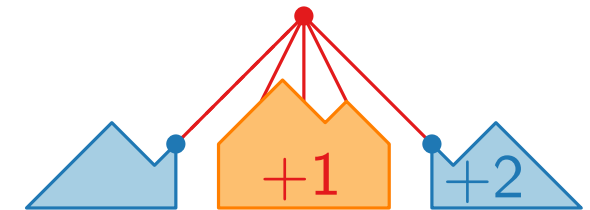
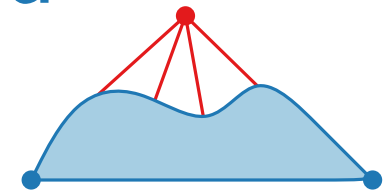
## Lecture 3:

### Straight-Line Drawings of Planar Graphs I: Canonical Ordering and Shift Method



#### Part II: Canonical Order

Alexander Wolff



# Canonical Order – Definition

**Definition.**

Let  $G = (V, E)$  be a triangulated plane graph on  $n \geq 3$  vertices.

# Canonical Order – Definition

## Definition.

Let  $G = (V, E)$  be a triangulated plane graph on  $n \geq 3$  vertices. An ordering  $\pi = (v_1, v_2, \dots, v_n)$  of  $V$  is called a **canonical order** if the following conditions hold for each  $k \in \{3, 4, \dots, n\}$ :

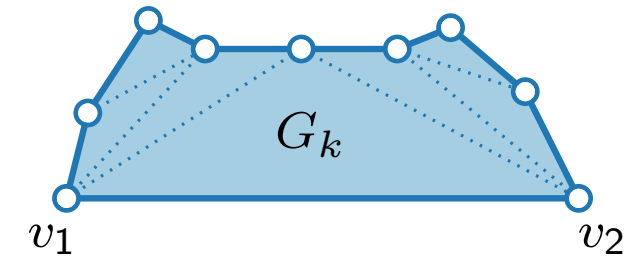
# Canonical Order – Definition

## Definition.

Let  $G = (V, E)$  be a triangulated plane graph on  $n \geq 3$  vertices.

An ordering  $\pi = (v_1, v_2, \dots, v_n)$  of  $V$  is called a **canonical order** if the following conditions hold for each  $k \in \{3, 4, \dots, n\}$ :

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



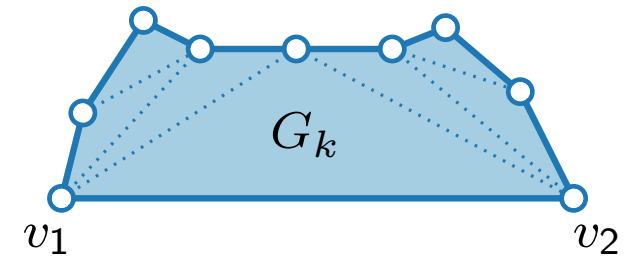
# Canonical Order – Definition

## Definition.

Let  $G = (V, E)$  be a triangulated plane graph on  $n \geq 3$  vertices.

An ordering  $\pi = (v_1, v_2, \dots, v_n)$  of  $V$  is called a **canonical order** if the following conditions hold for each  $k \in \{3, 4, \dots, n\}$ :

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

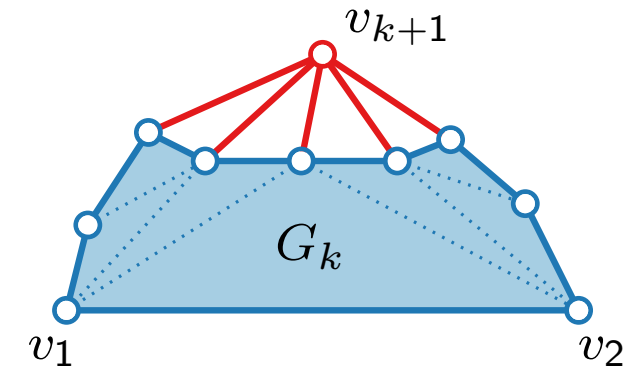


# Canonical Order – Definition

## Definition.

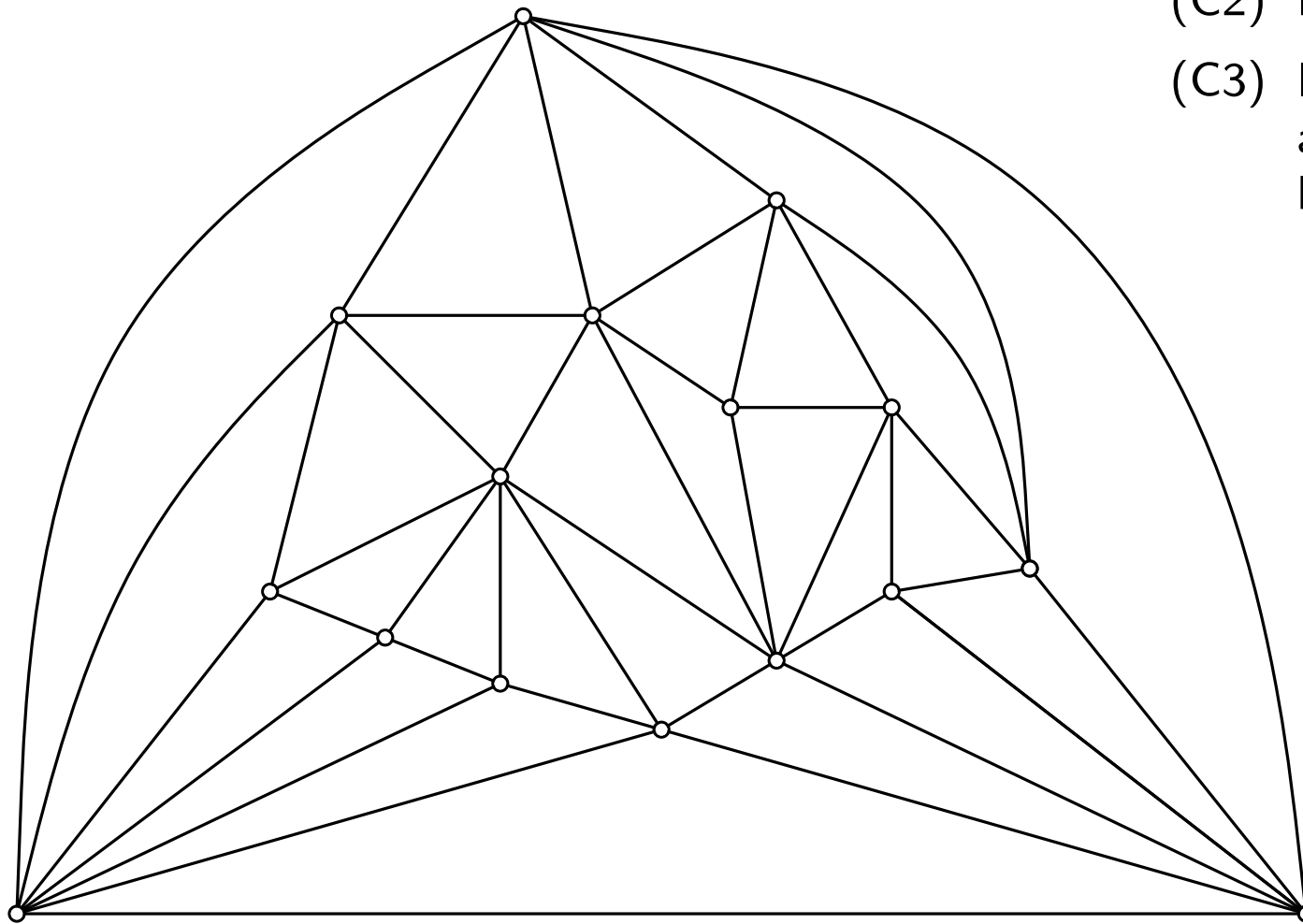
Let  $G = (V, E)$  be a triangulated plane graph on  $n \geq 3$  vertices. An ordering  $\pi = (v_1, v_2, \dots, v_n)$  of  $V$  is called a **canonical order** if the following conditions hold for each  $k \in \{3, 4, \dots, n\}$ :

- (C1) Vertices  $\{v_1, \dots, v_k\}$  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 the neighbors of  $v_{k+1}$  form a path on the boundary of  $G_k$ .





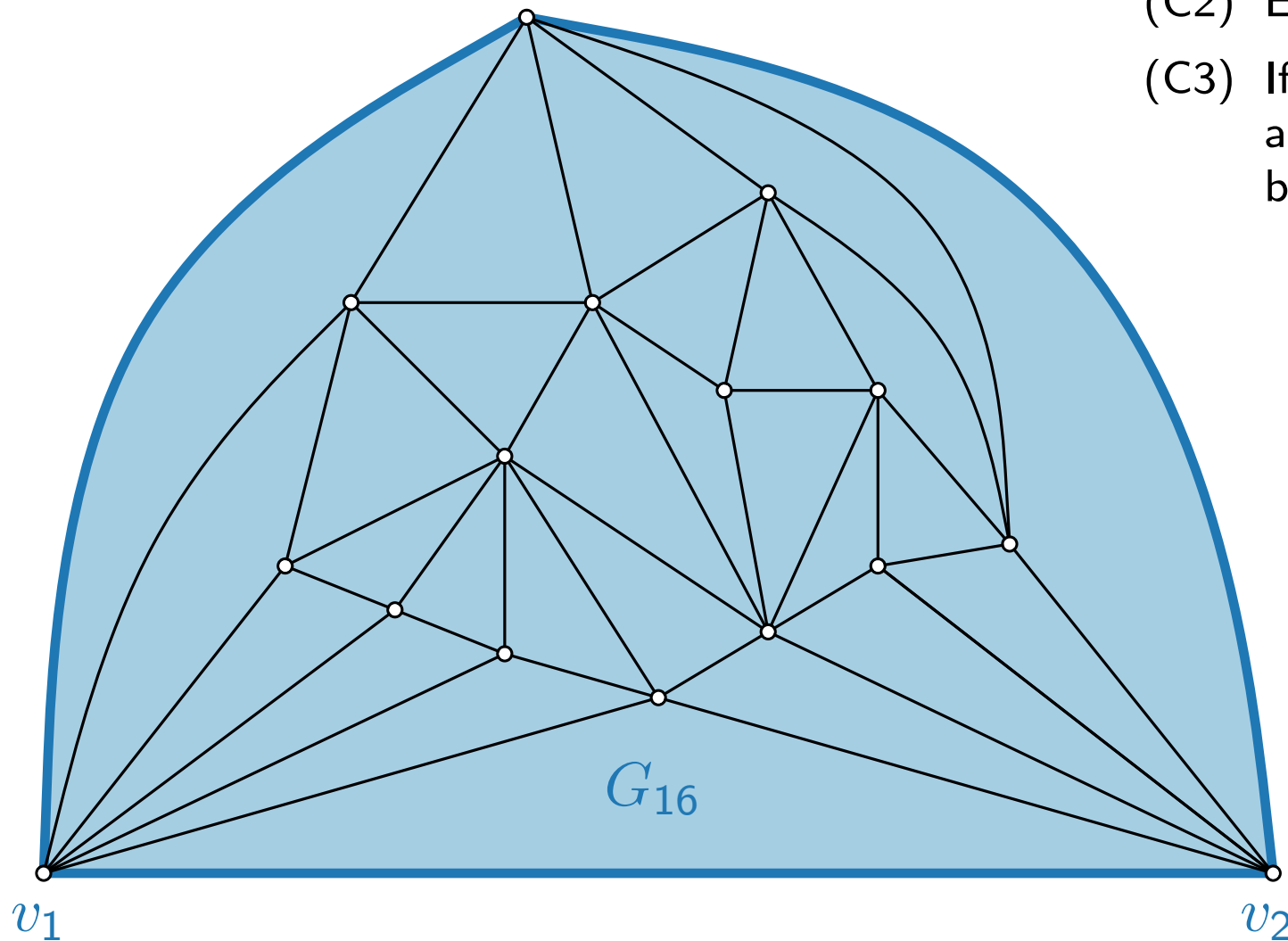
# Canonical Order – Example



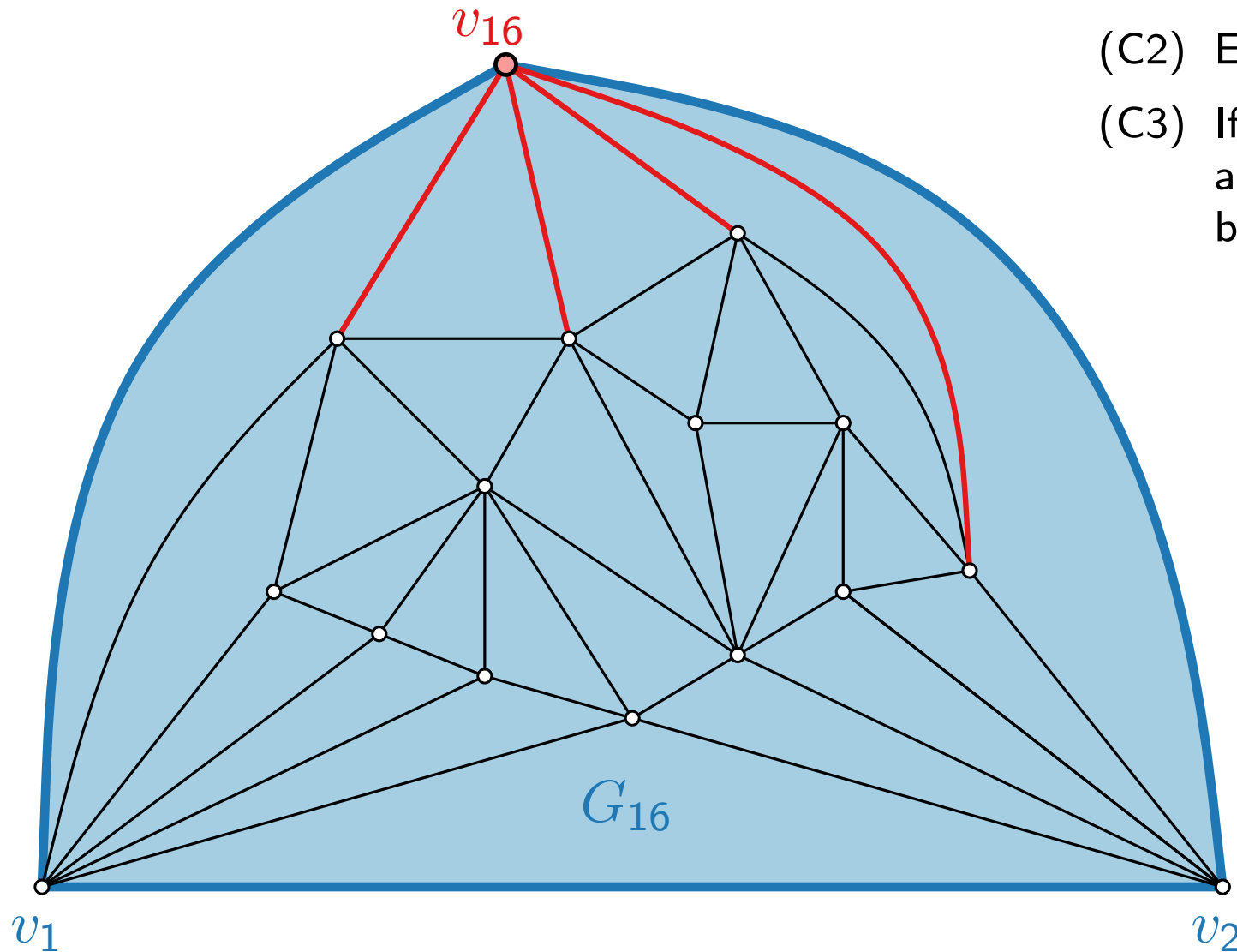
- (C1) Vertices  $\{v_1, \dots, v_k\}$  induce a biconnected internally triangulated graph; call it  $G_k$ .
- (C2) Edge  $(v_1, v_2)$  belongs to the outer face of  $G_k$ .
- (C3) If  $k < n$  then vertex  $v_{k+1}$  lies in the outer face of  $G_k$ , and the neighbors of  $v_{k+1}$  form a path on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

# Canonical Order – Example

- (C1) Vertices  $\{v_1, \dots, v_k\}$  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 the neighbors of  $v_{k+1}$  form a path on the boundary of  $G_k$ .

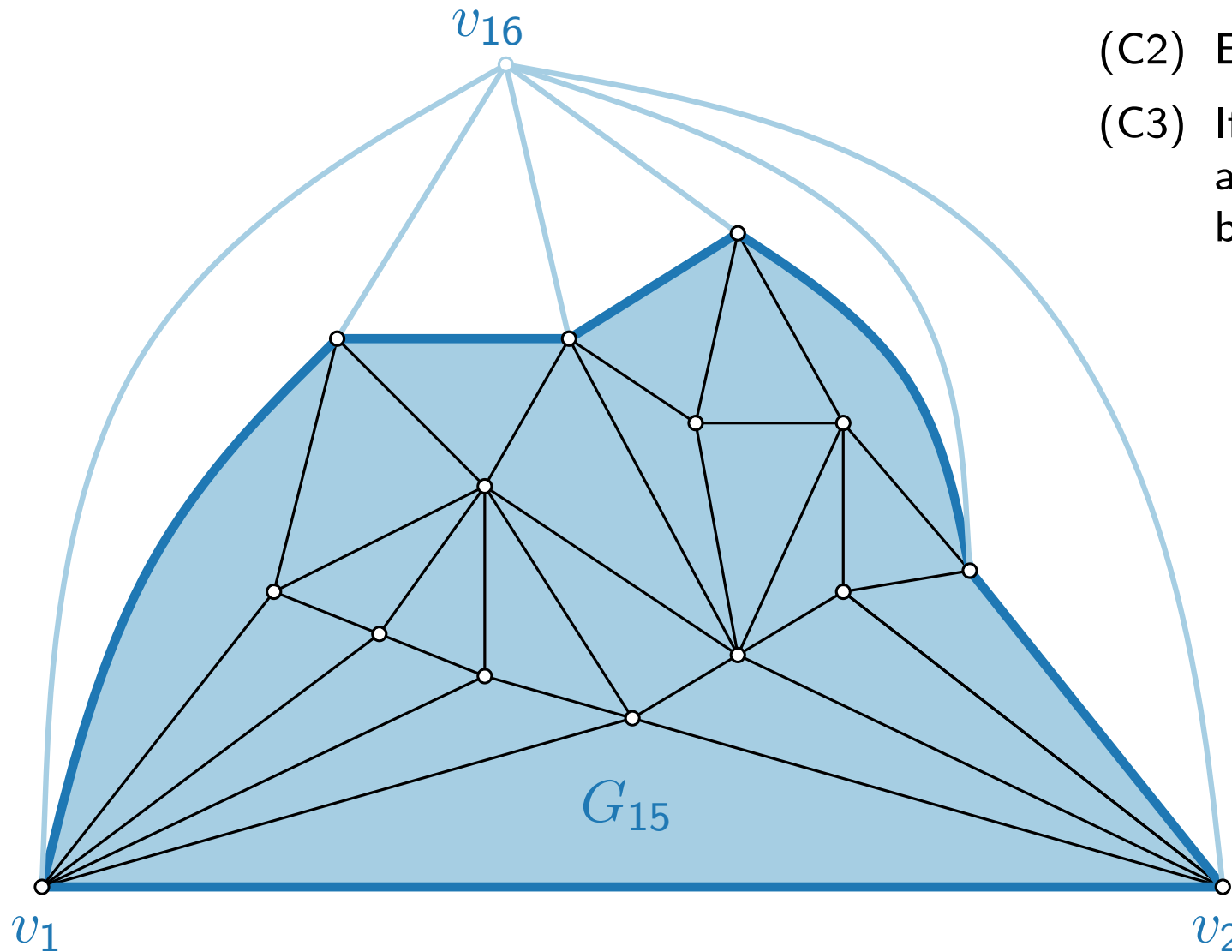


# Canonical Order – Example



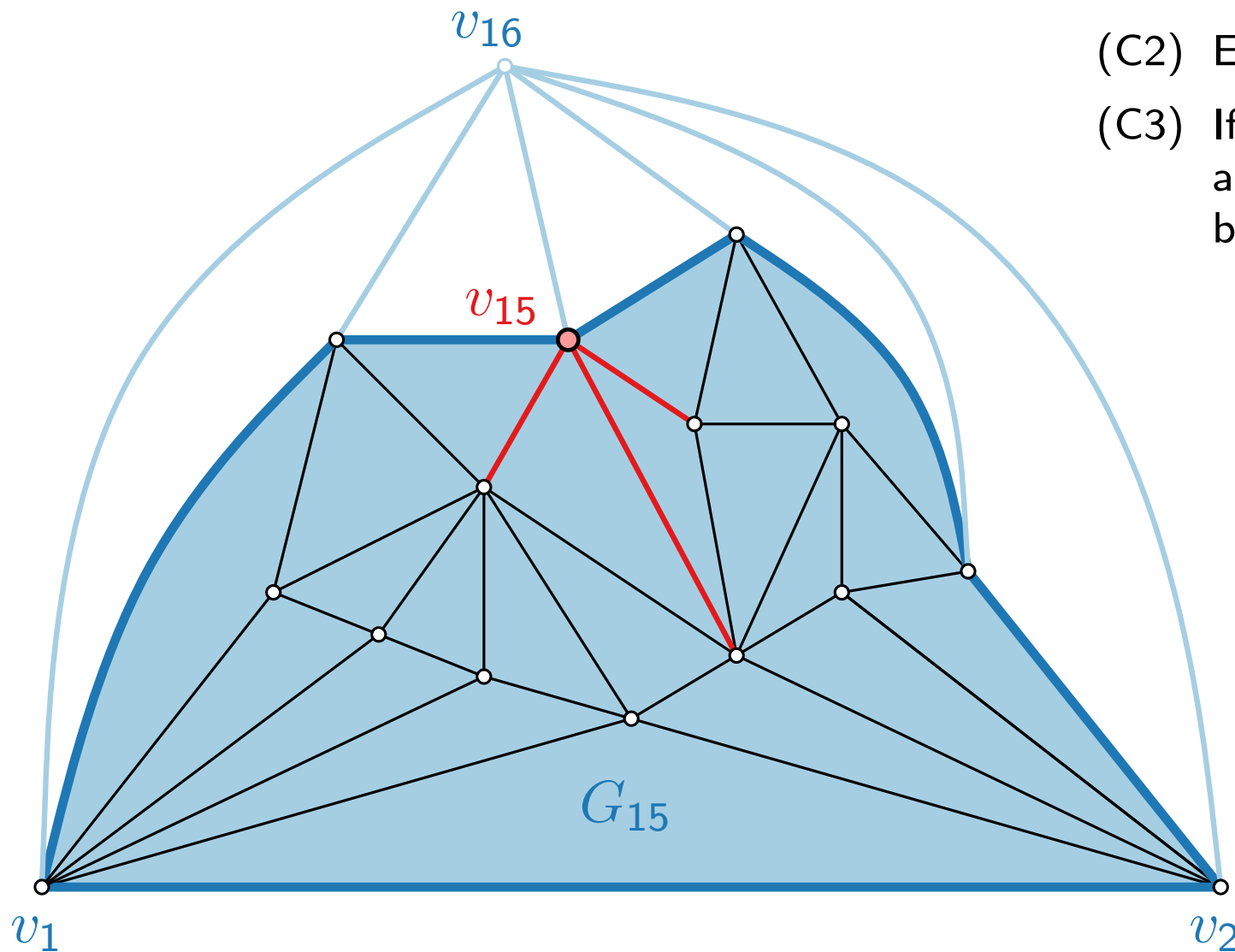
- (C1) Vertices  $\{v_1, \dots, v_k\}$  induce a biconnected internally triangulated graph; call it  $G_k$ .
- (C2) Edge  $(v_1, v_2)$  belongs to the outer face of  $G_k$ .
- (C3) If  $k < n$  then vertex  $v_{k+1}$  lies in the outer face of  $G_k$ , and the neighbors of  $v_{k+1}$  form a path on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

# Canonical Order – Example



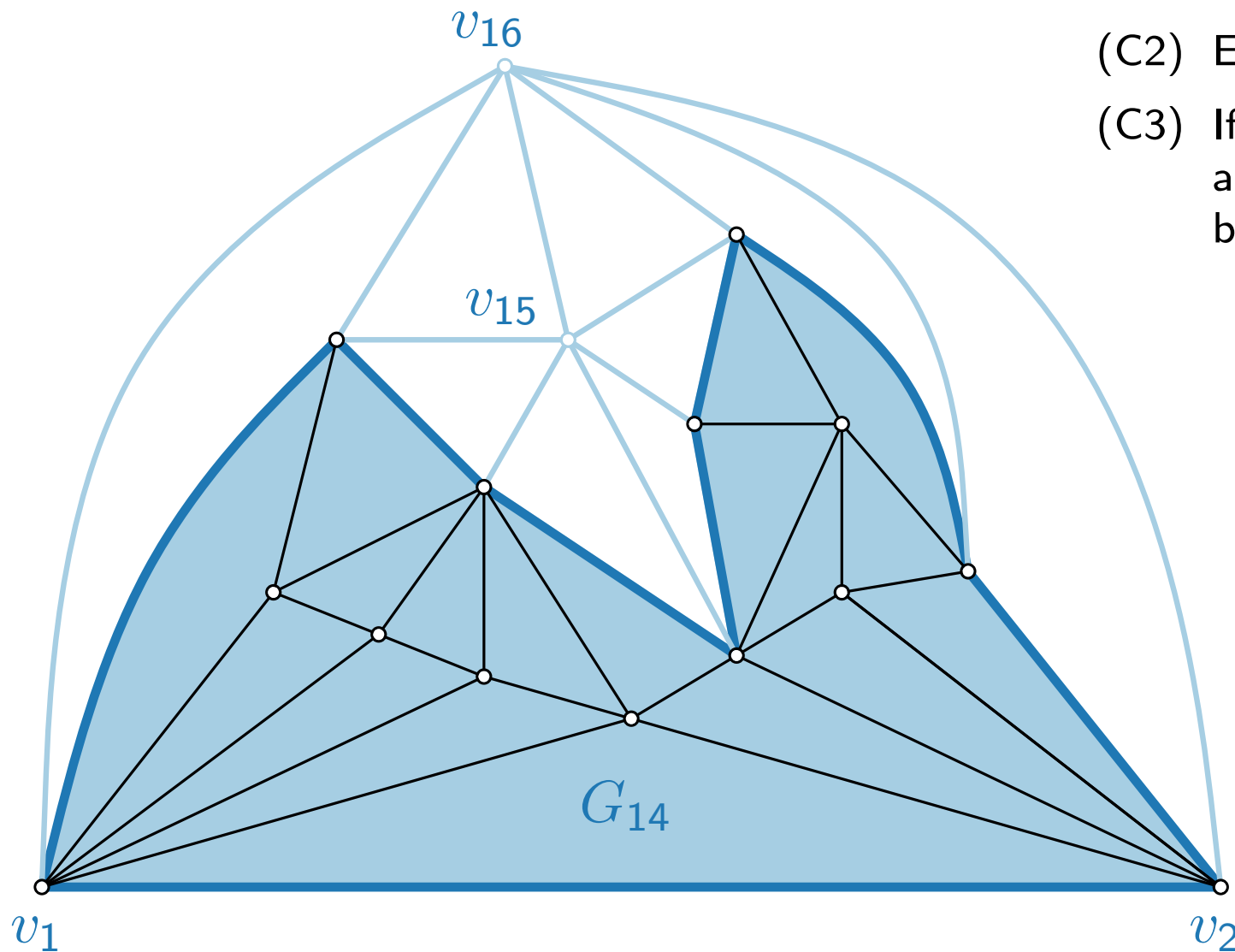
- (C1) Vertices  $\{v_1, \dots, v_k\}$  induce a biconnected internally triangulated graph; call it  $G_k$ .
- (C2) Edge  $(v_1, v_2)$  belongs to the outer face of  $G_k$ .
- (C3) If  $k < n$  then vertex  $v_{k+1}$  lies in the outer face of  $G_k$ , and the neighbors of  $v_{k+1}$  form a path on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

# Canonical Order – Example



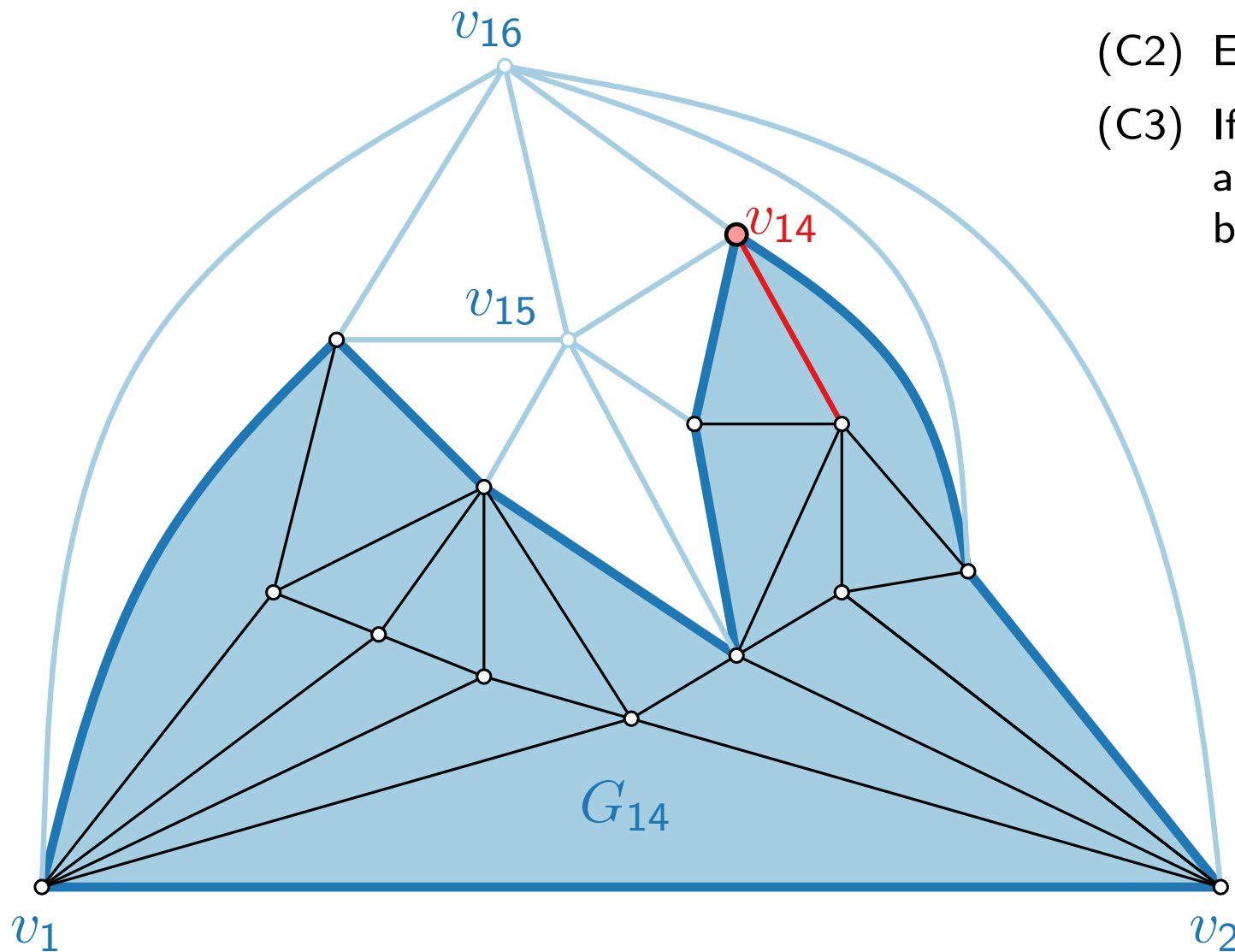
- (C1) Vertices  $\{v_1, \dots, v_k\}$  induce a biconnected internally triangulated graph; call it  $G_k$ .
- (C2) Edge  $(v_1, v_2)$  belongs to the outer face of  $G_k$ .
- (C3) If  $k < n$  then vertex  $v_{k+1}$  lies in the outer face of  $G_k$ , and the neighbors of  $v_{k+1}$  form a path on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

# Canonical Order – Example



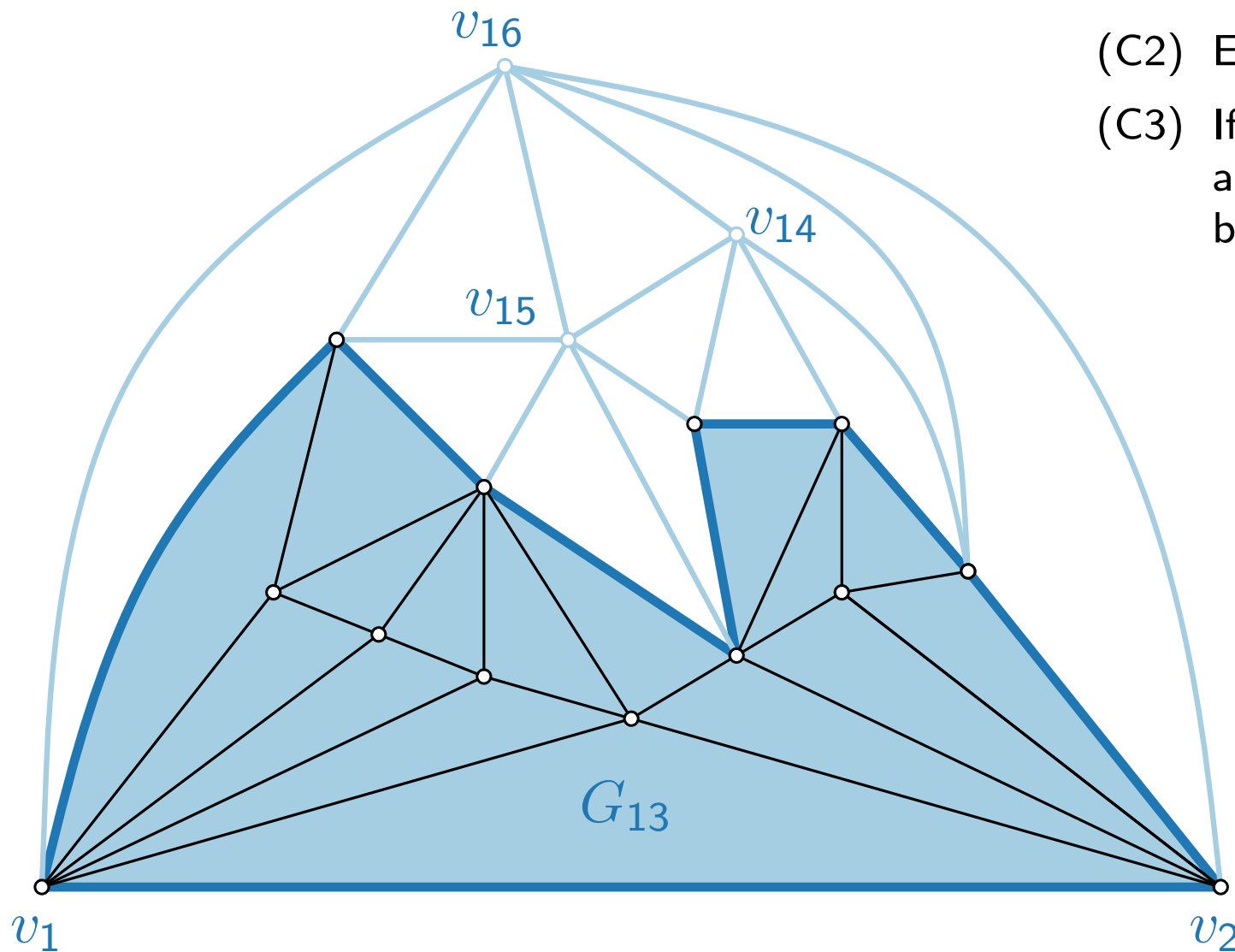
- (C1) Vertices  $\{v_1, \dots, v_k\}$  induce a biconnected internally triangulated graph; call it  $G_k$ .
- (C2) Edge  $(v_1, v_2)$  belongs to the outer face of  $G_k$ .
- (C3) If  $k < n$  then vertex  $v_{k+1}$  lies in the outer face of  $G_k$ , and the neighbors of  $v_{k+1}$  form a path on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

# Canonical Order – Example



- (C1) Vertices  $\{v_1, \dots, v_k\}$  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 the neighbors of  $v_{k+1}$  form a path on the boundary of  $G_k$ .

# Canonical Order – Example

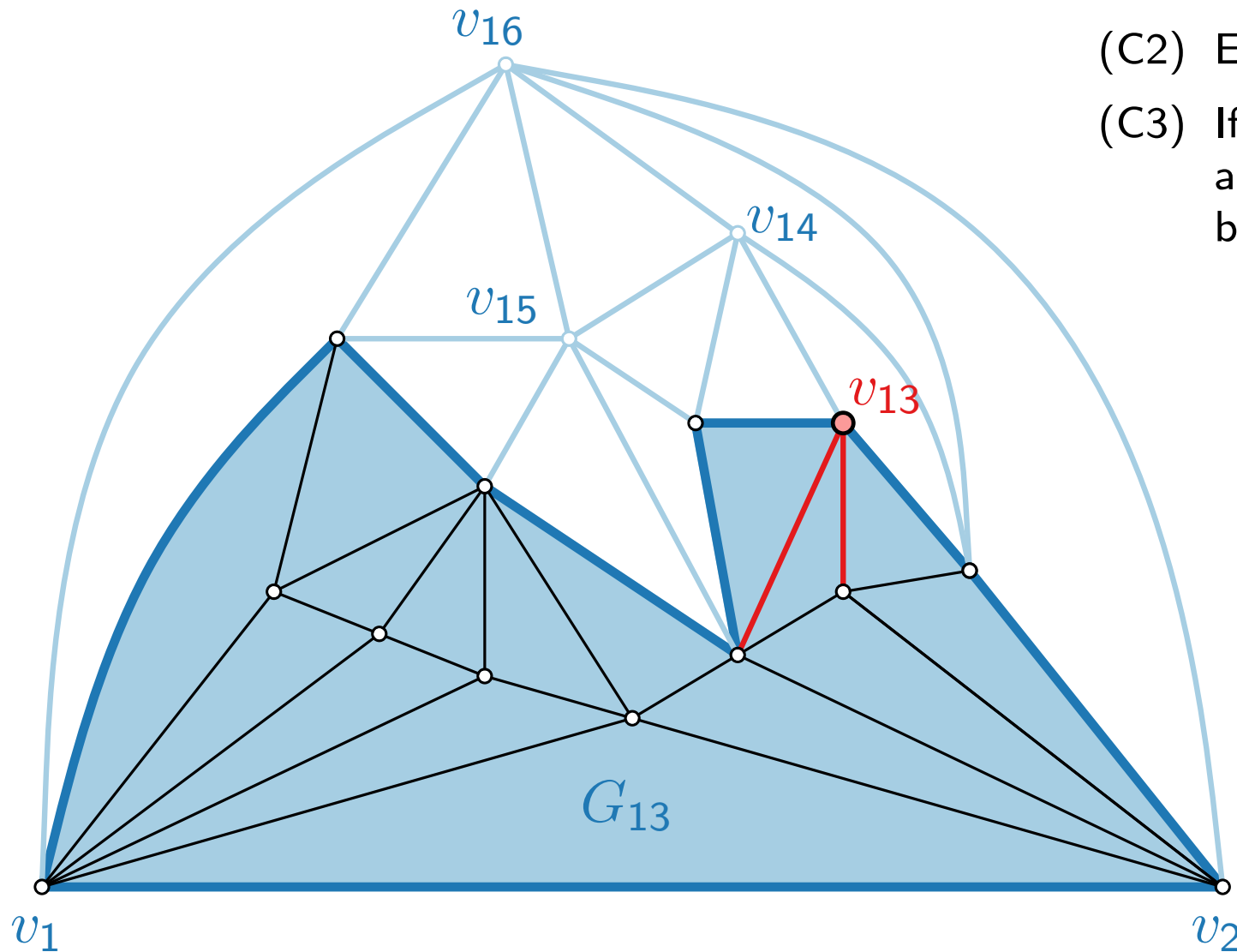


- (C1) Vertices  $\{v_1, \dots, v_k\}$  induce a biconnected internally triangulated graph; call it  $G_k$ .
- (C2) Edge  $(v_1, v_2)$  belongs to the outer face of  $G_k$ .
- (C3) If  $k < n$  then vertex  $v_{k+1}$  lies in the outer face of  $G_k$ , and the neighbors of  $v_{k+1}$  form a path on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

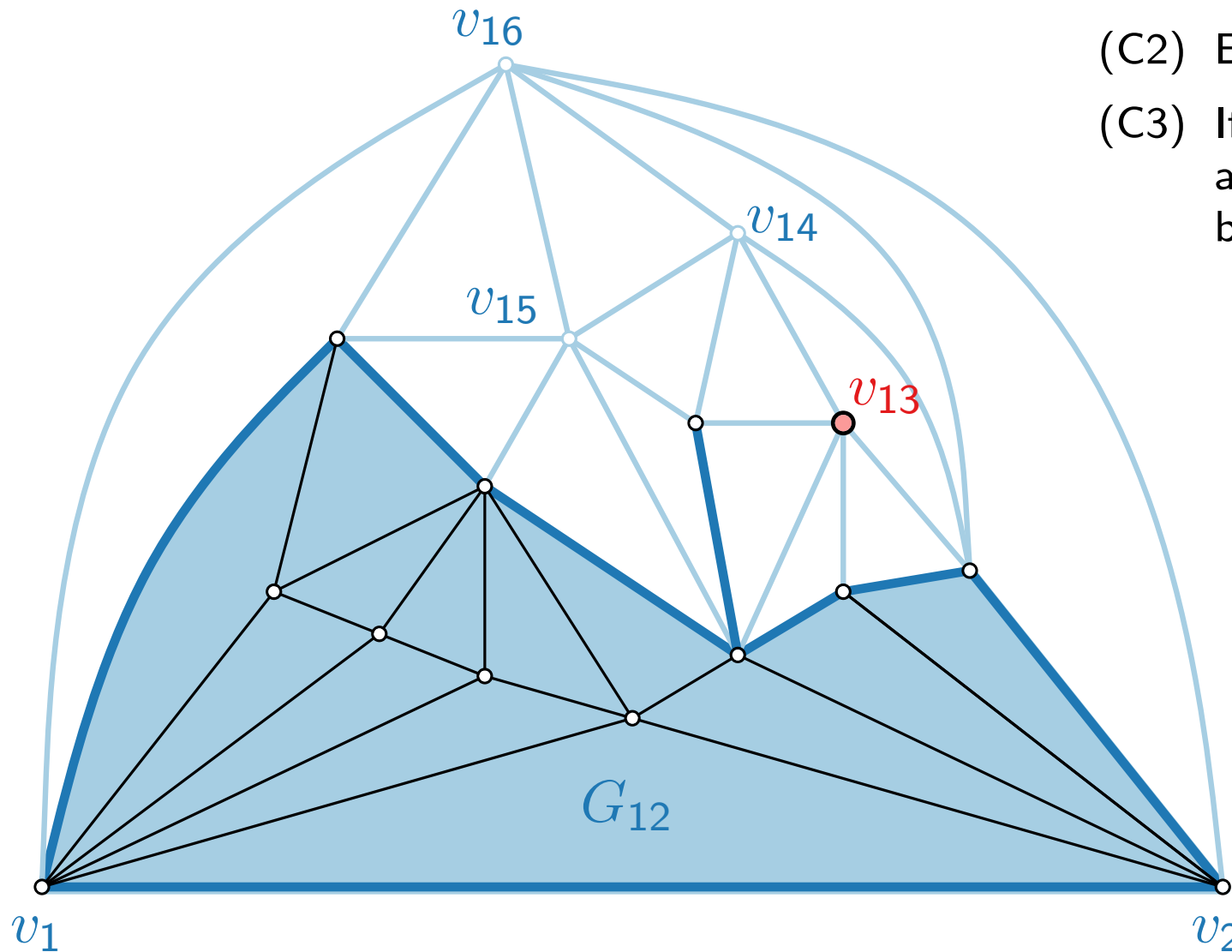


# Canonical Order – Example

- (C1) Vertices  $\{v_1, \dots, v_k\}$  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 the neighbors of  $v_{k+1}$  form a path on the boundary of  $G_k$ .

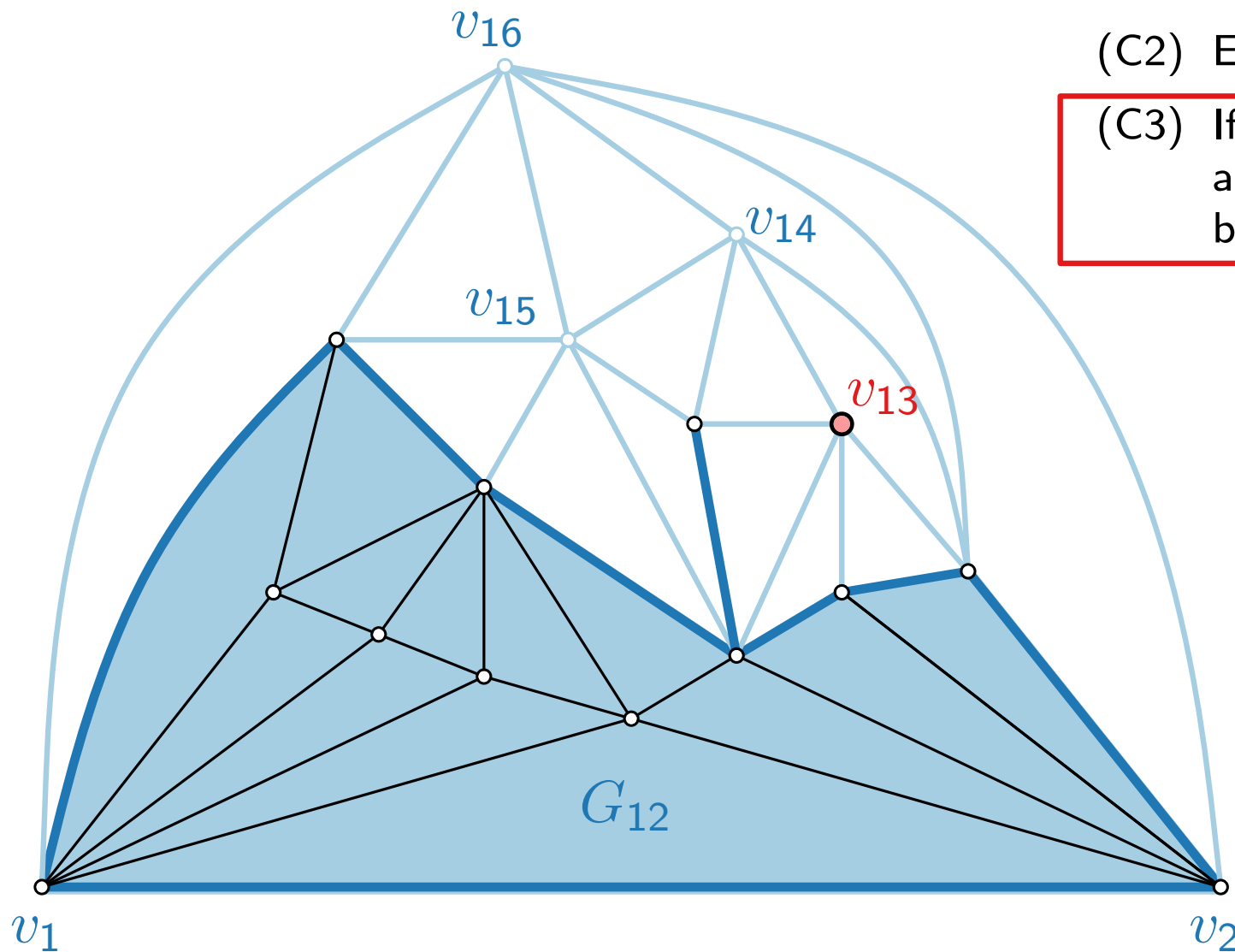


# Canonical Order – Example



- (C1) Vertices  $\{v_1, \dots, v_k\}$  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 the neighbors of  $v_{k+1}$  form a path on the boundary of  $G_k$ .

# Canonical Order – Example



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

(C2) Edge  $(v_1, v_2)$  belongs to the outer face of  $G_k$ .

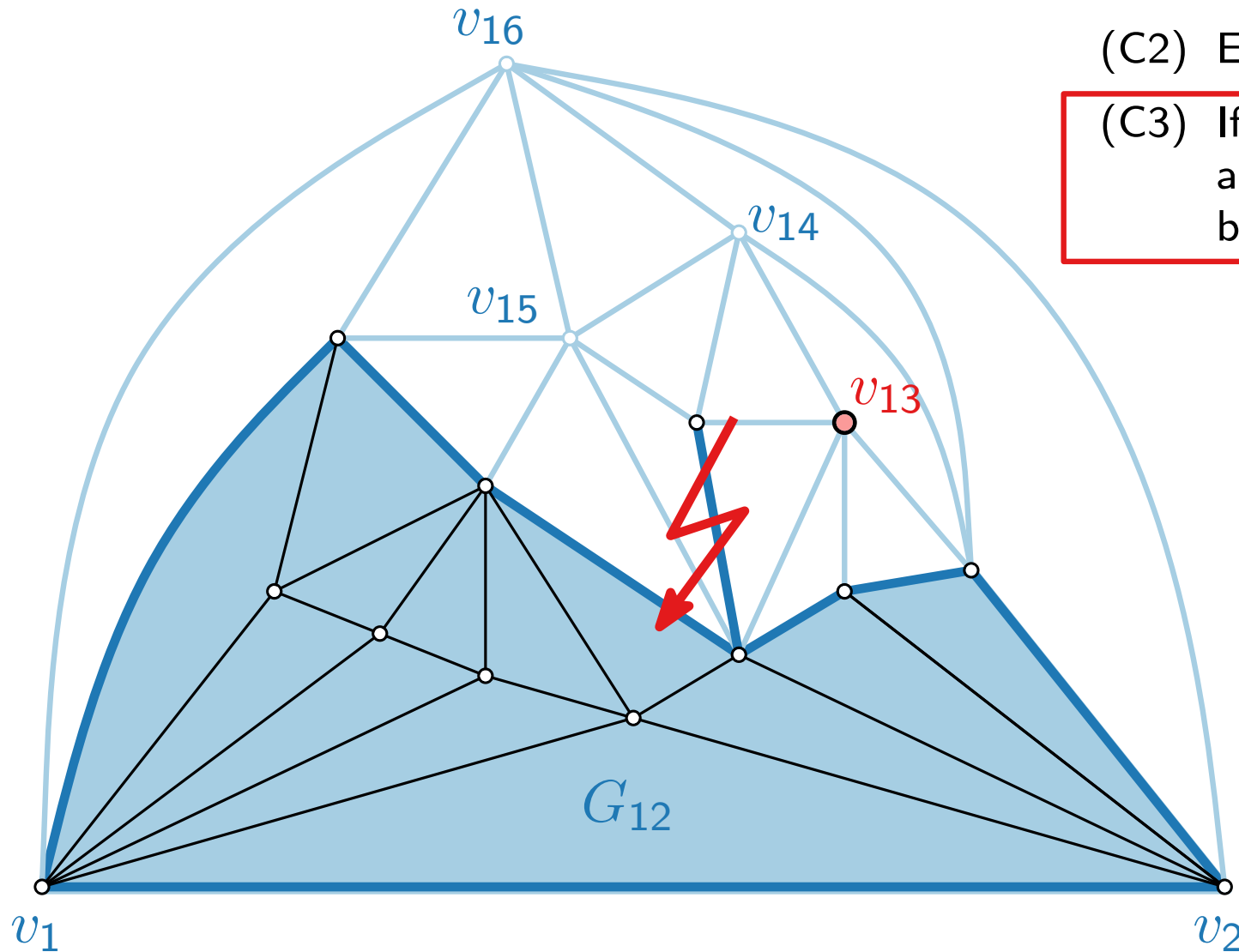
(C3) If  $k < n$  then vertex  $v_{k+1}$  lies in the outer face of  $G_k$ , and the neighbors of  $v_{k+1}$  form a path on the boundary of  $G_k$ .

# Canonical Order – Example

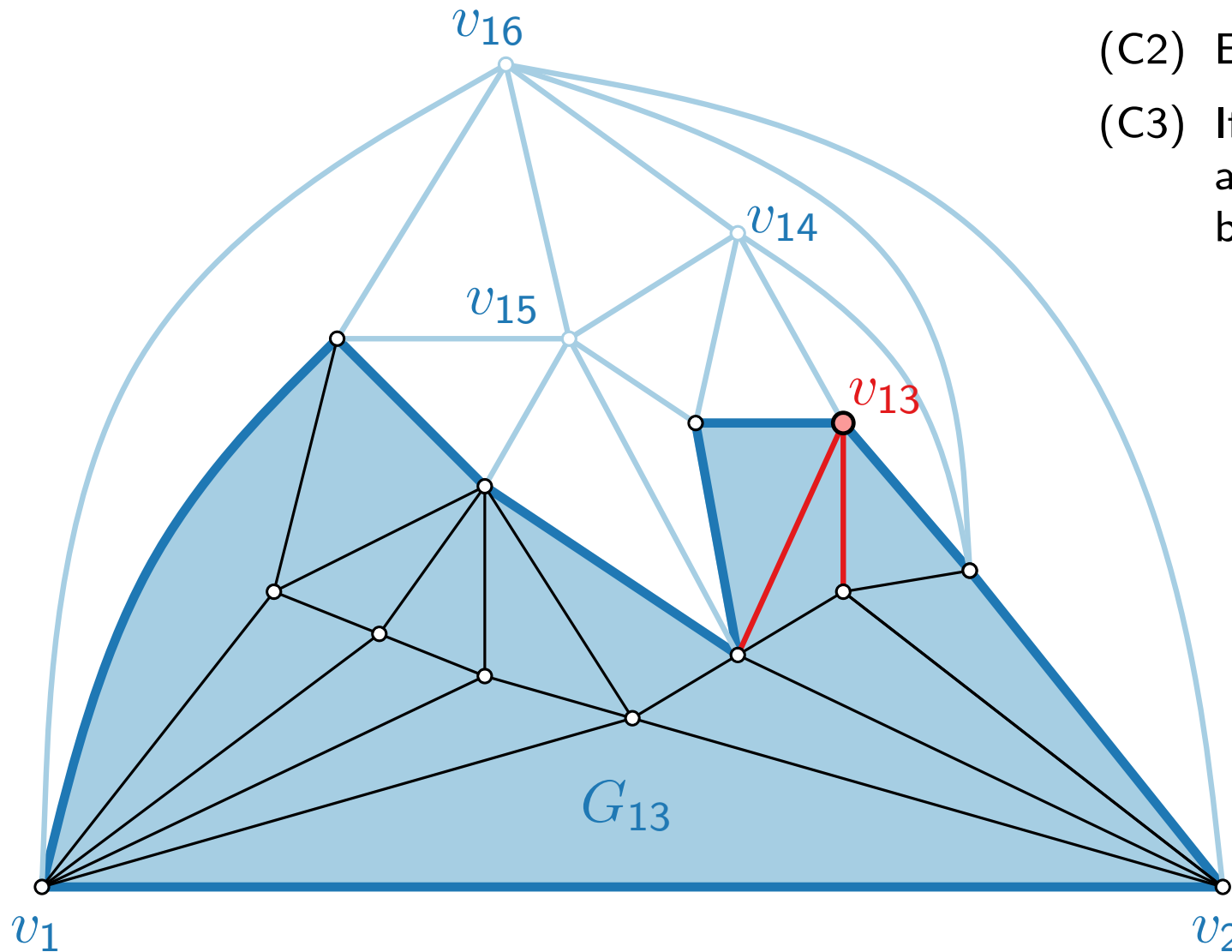
(C1) Vertices  $\{v_1, \dots, v_k\}$  induce a biconnected internally triangulated graph; call it  $G_k$ .

(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 the neighbors of  $v_{k+1}$  form a path on the boundary of  $G_k$ .

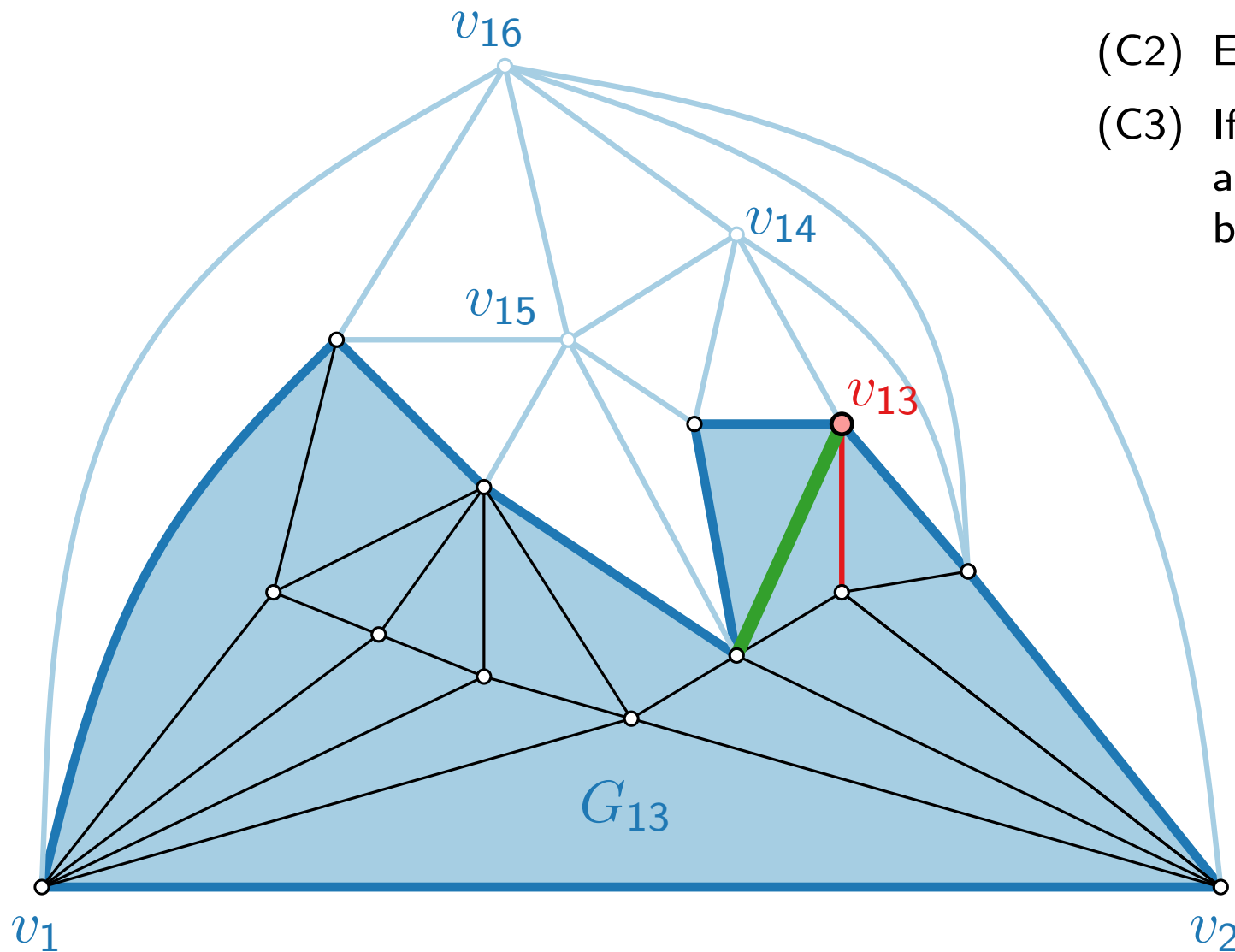


# Canonical Order – Example



- (C1) Vertices  $\{v_1, \dots, v_k\}$  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 the neighbors of  $v_{k+1}$  form a path on the boundary of  $G_k$ .

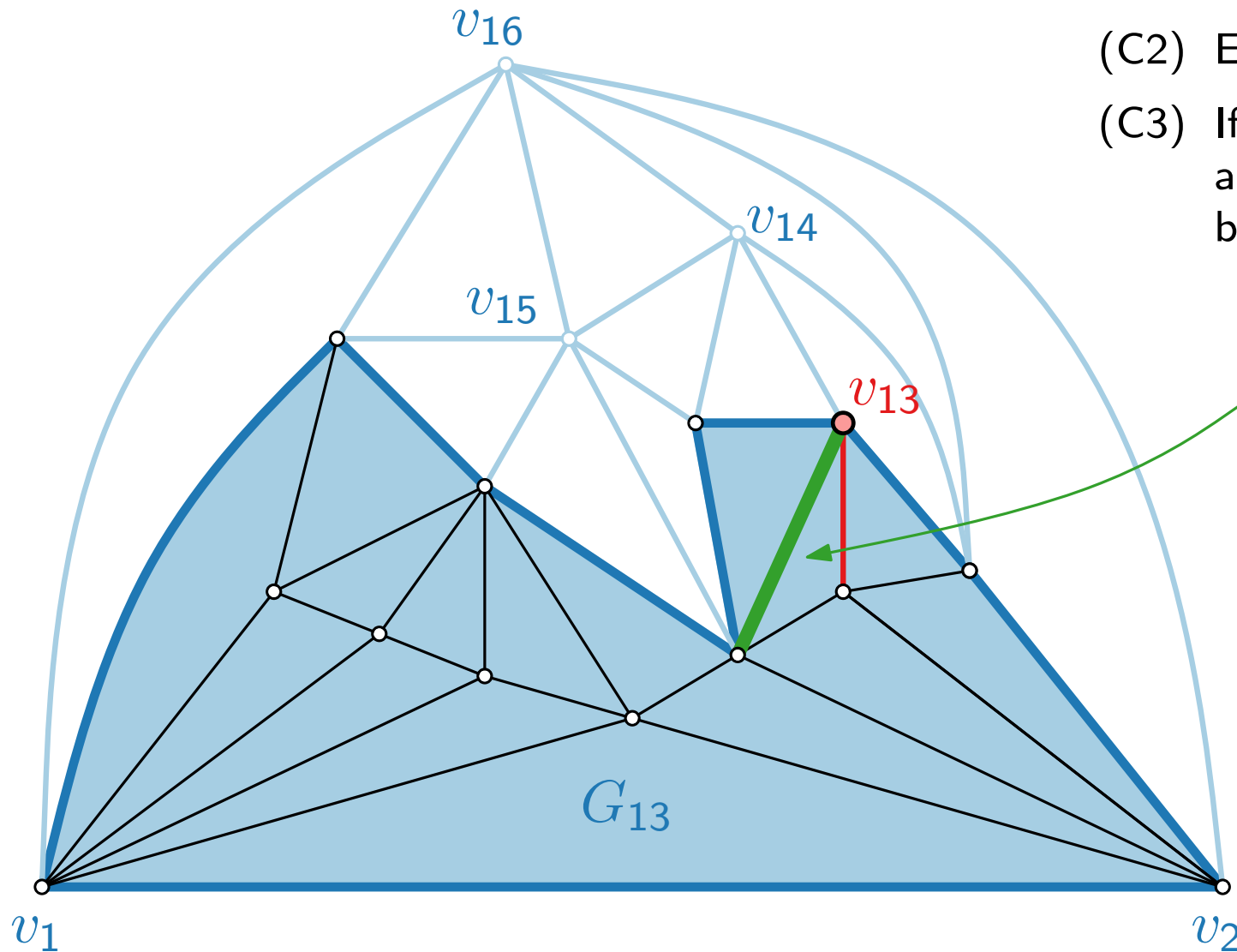
# Canonical Order – Example



- (C1) Vertices  $\{v_1, \dots, v_k\}$  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 the neighbors of  $v_{k+1}$  form a path on the boundary of  $G_k$ .

# Canonical Order – Example

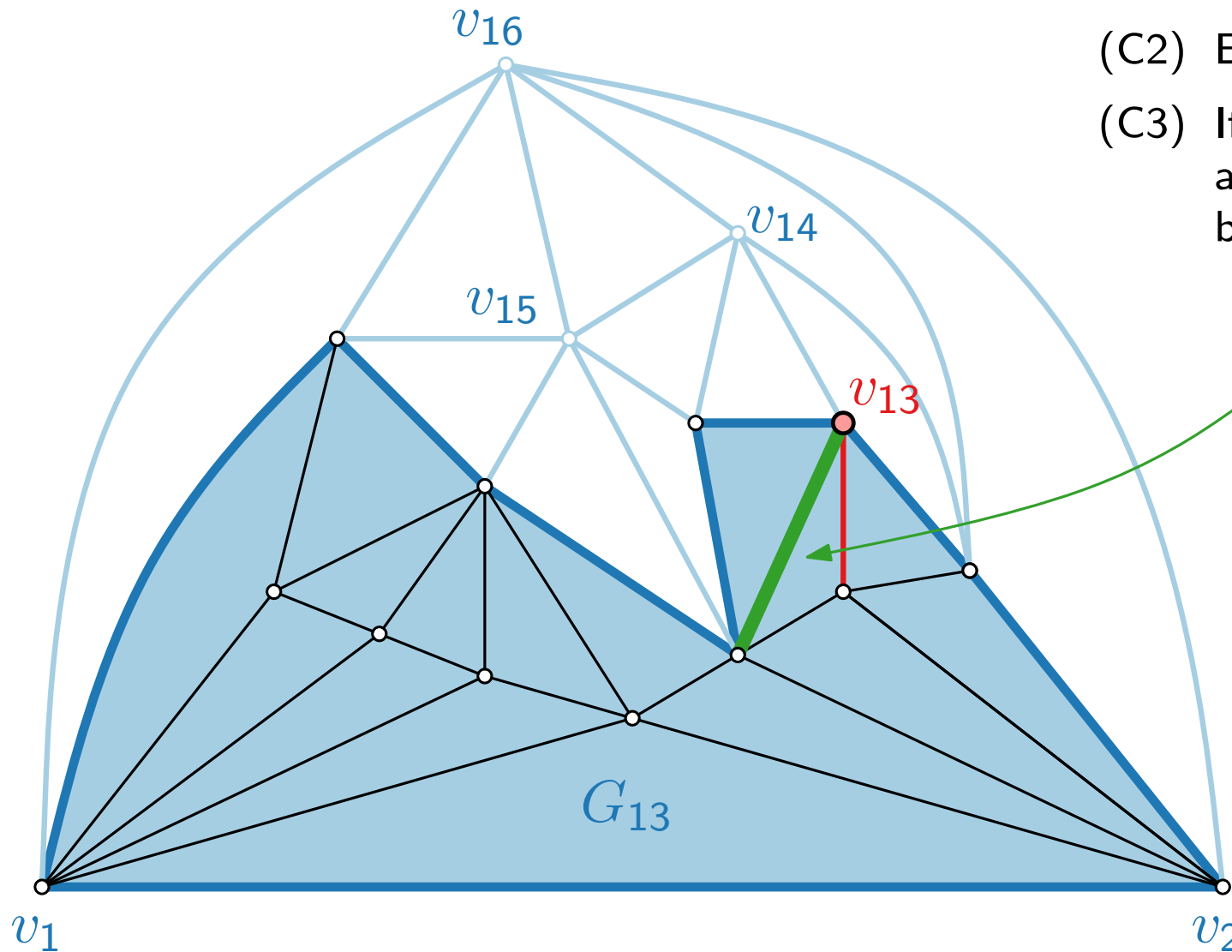
- (C1) Vertices  $\{v_1, \dots, v_k\}$  induce a biconnected internally triangulated graph; call it  $G_k$ .
- (C2) Edge  $(v_1, v_2)$  belongs to the outer face of  $G_k$ .
- (C3) If  $k < n$  then vertex  $v_{k+1}$  lies in the outer face of  $G_k$ , and the neighbors of  $v_{k+1}$  form a path on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



chord:

# Canonical Order – Example

- (C1) Vertices  $\{v_1, \dots, v_k\}$  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 the neighbors of  $v_{k+1}$  form a path on the boundary of  $G_k$ .

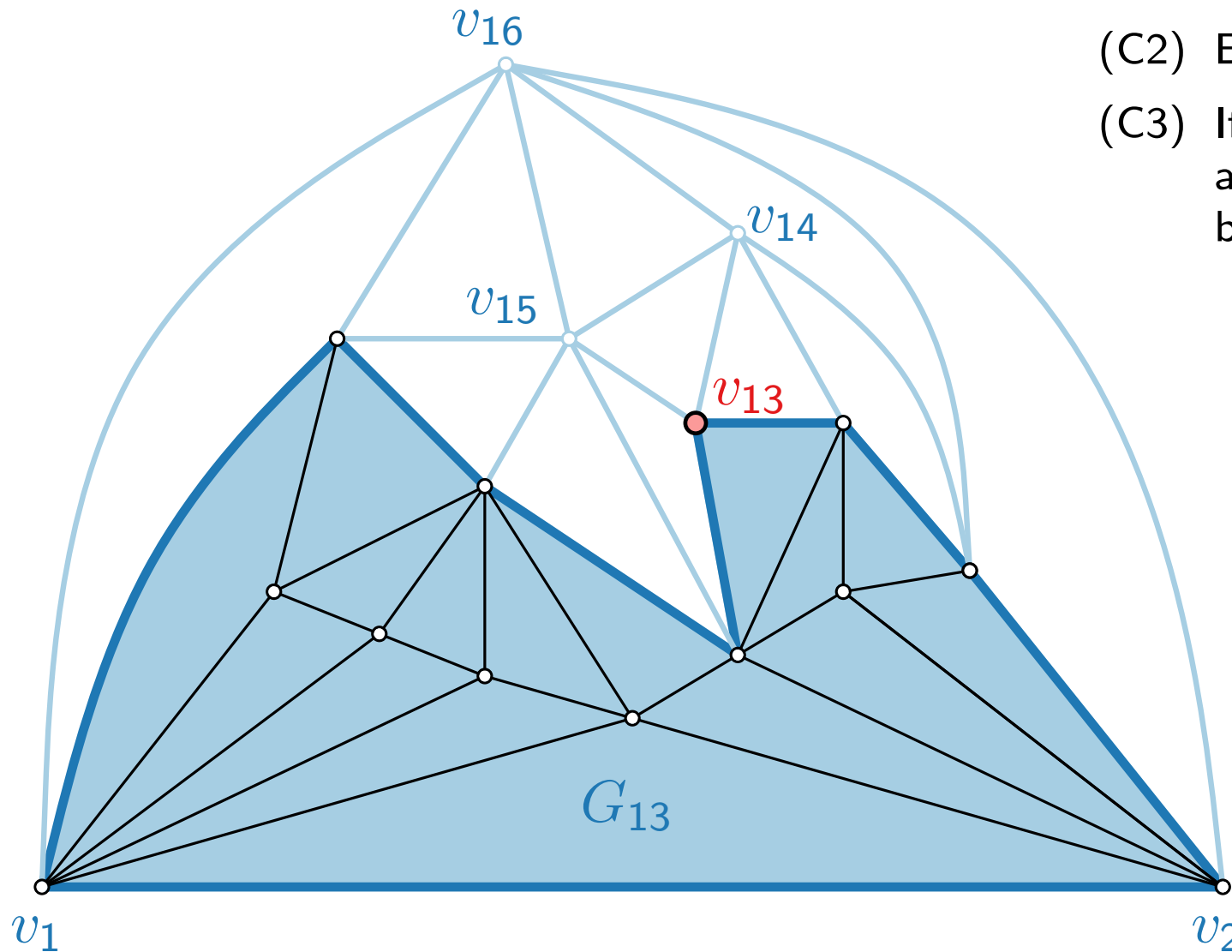


*chord:*

edge joining two  
nonadjacent  
vertices in a cycle

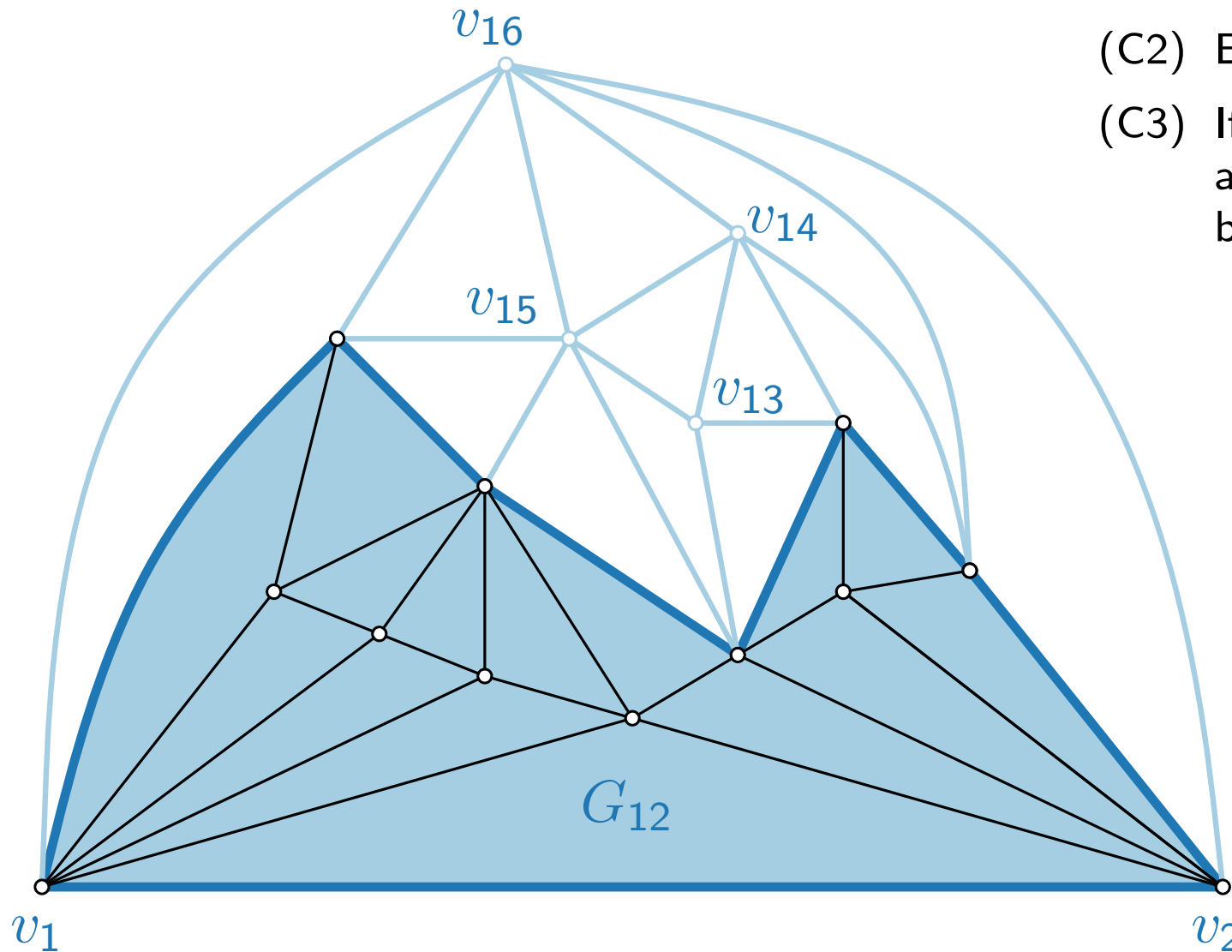


# Canonical Order – Example



- (C1) Vertices  $\{v_1, \dots, v_k\}$  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 the neighbors of  $v_{k+1}$  form a path on the boundary of  $G_k$ .

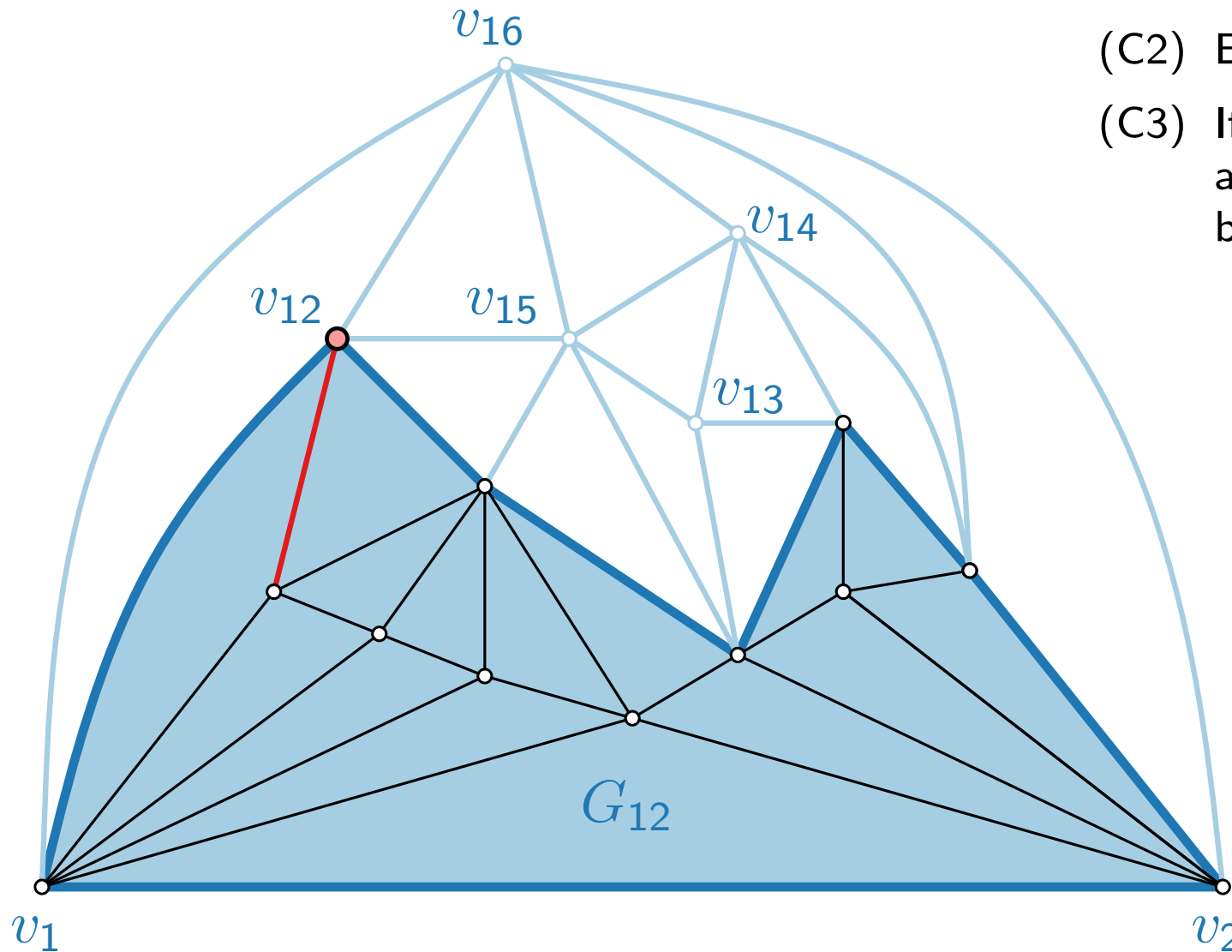
# Canonical Order – Example



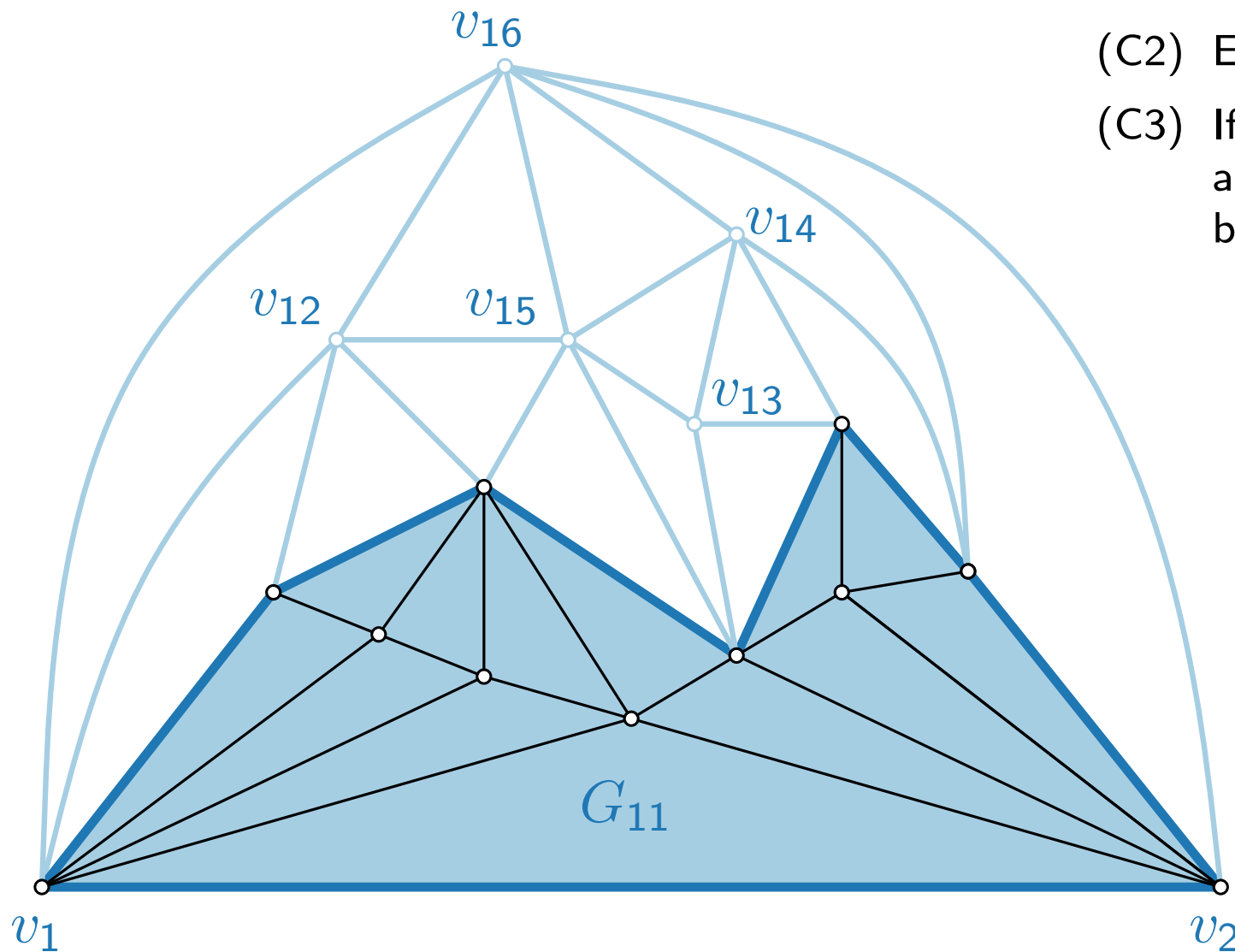
- (C1) Vertices  $\{v_1, \dots, v_k\}$  induce a biconnected internally triangulated graph; call it  $G_k$ .
- (C2) Edge  $(v_1, v_2)$  belongs to the outer face of  $G_k$ .
- (C3) If  $k < n$  then vertex  $v_{k+1}$  lies in the outer face of  $G_k$ , and the neighbors of  $v_{k+1}$  form a path on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

# Canonical Order – Example

- (C1) Vertices  $\{v_1, \dots, v_k\}$  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 the neighbors of  $v_{k+1}$  form a path on the boundary of  $G_k$ .

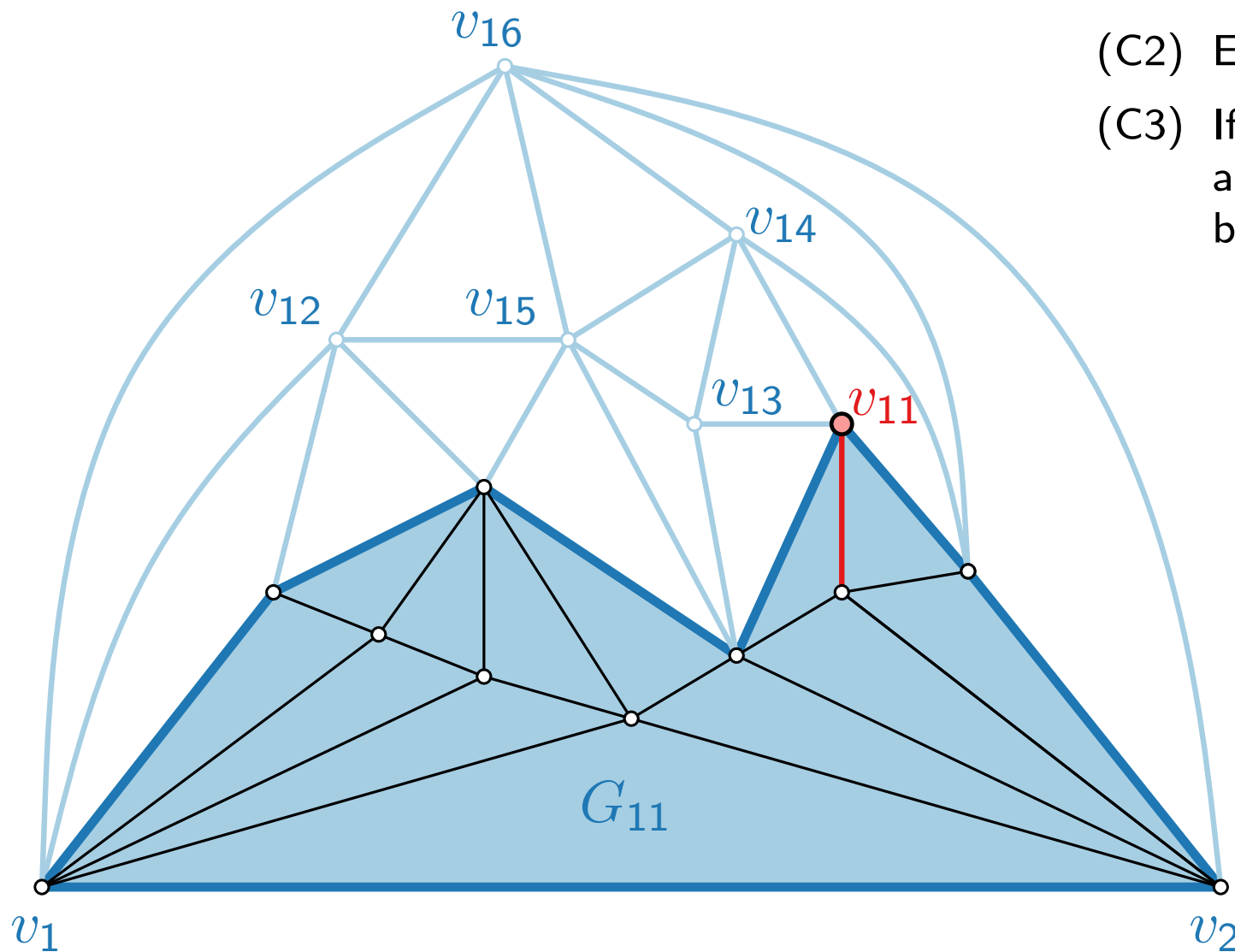


# Canonical Order – Example



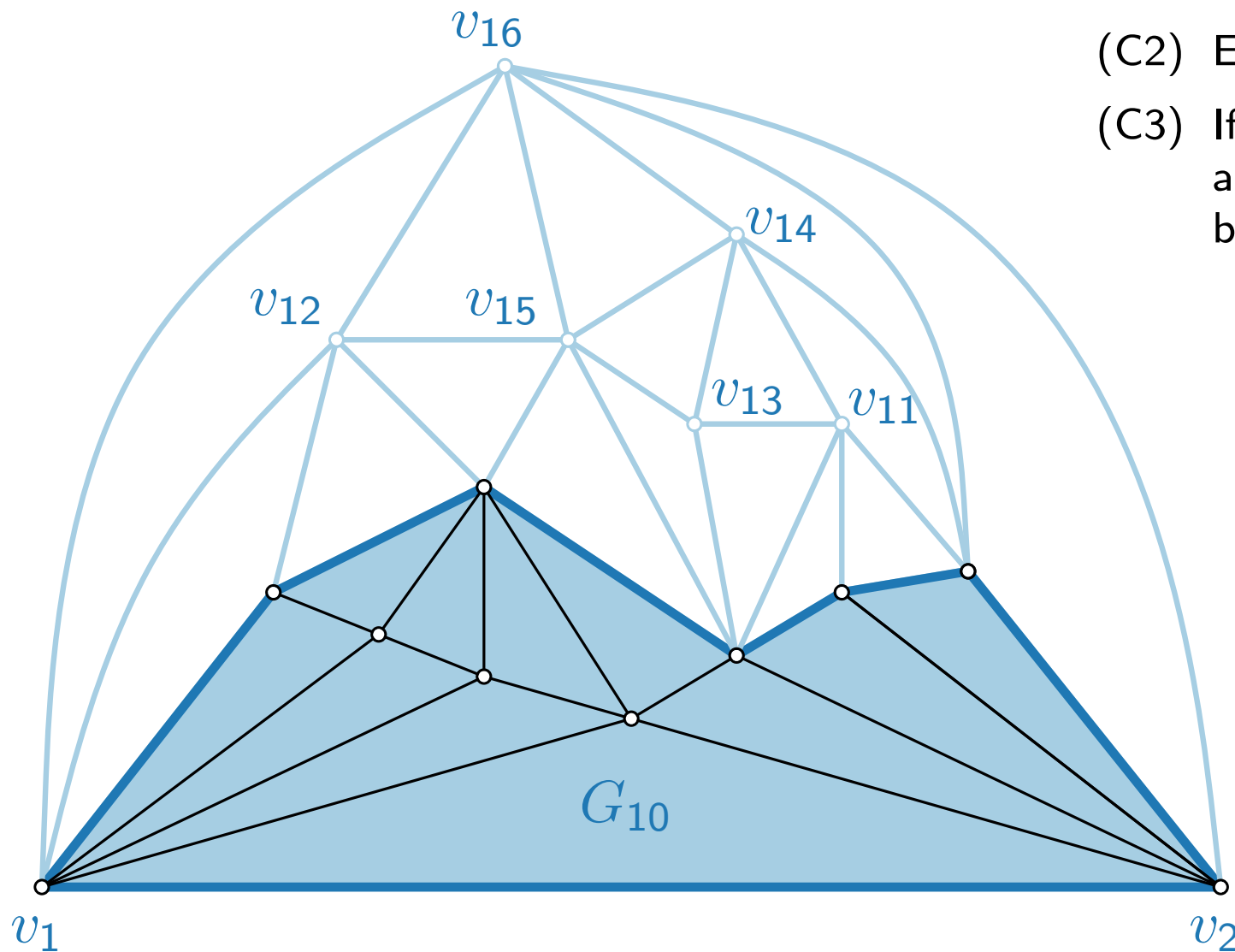
- (C1) Vertices  $\{v_1, \dots, v_k\}$  induce a biconnected internally triangulated graph; call it  $G_k$ .
- (C2) Edge  $(v_1, v_2)$  belongs to the outer face of  $G_k$ .
- (C3) If  $k < n$  then vertex  $v_{k+1}$  lies in the outer face of  $G_k$ , and the neighbors of  $v_{k+1}$  form a path on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

# Canonical Order – Example



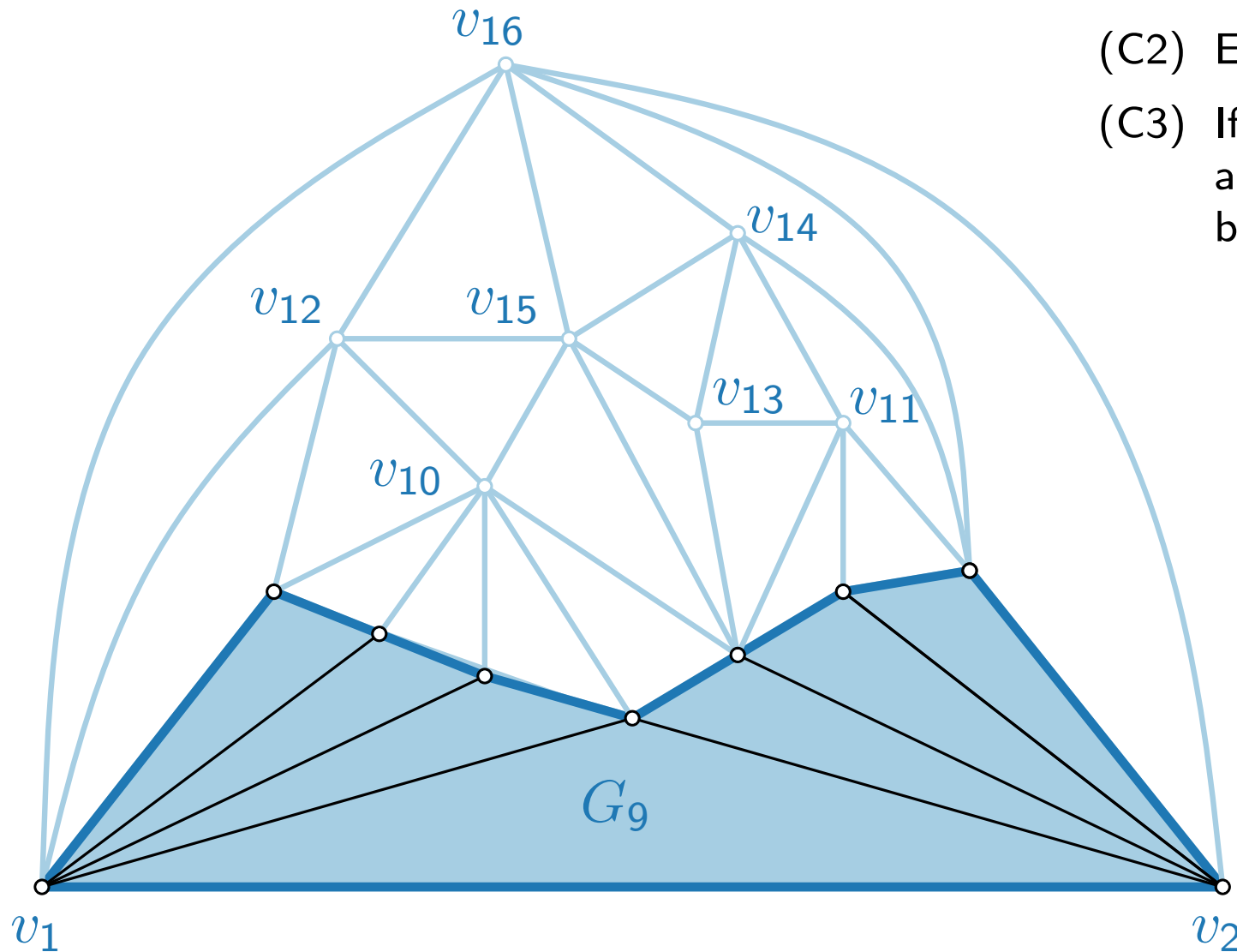
- (C1) Vertices  $\{v_1, \dots, v_k\}$  induce a biconnected internally triangulated graph; call it  $G_k$ .
- (C2) Edge  $(v_1, v_2)$  belongs to the outer face of  $G_k$ .
- (C3) If  $k < n$  then vertex  $v_{k+1}$  lies in the outer face of  $G_k$ , and the neighbors of  $v_{k+1}$  form a path on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

# Canonical Order – Example



- (C1) Vertices  $\{v_1, \dots, v_k\}$  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 the neighbors of  $v_{k+1}$  form a path on the boundary of  $G_k$ .

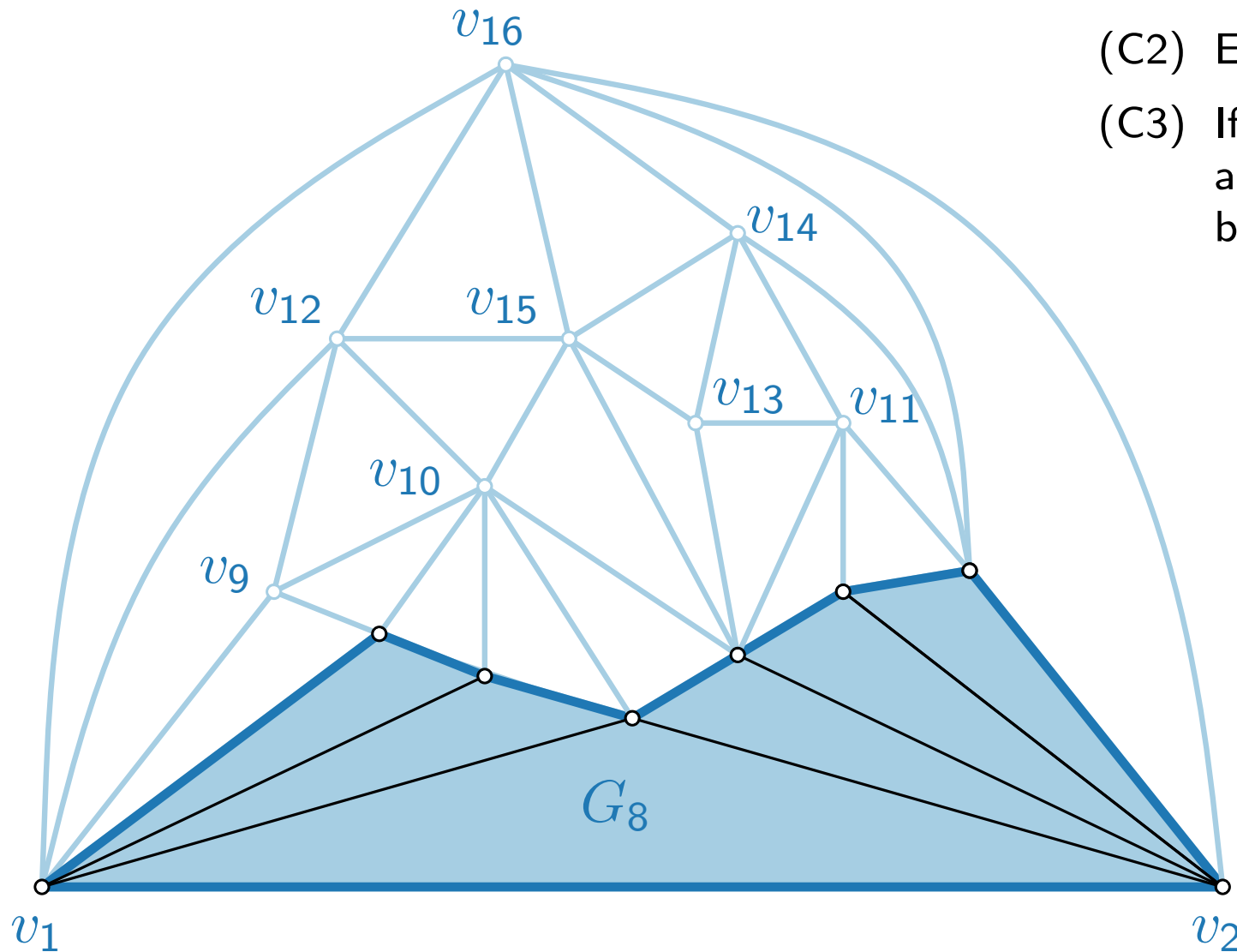
# Canonical Order – Example



- (C1) Vertices  $\{v_1, \dots, v_k\}$  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 the neighbors of  $v_{k+1}$  form a path on the boundary of  $G_k$ .

# Canonical Order – Example

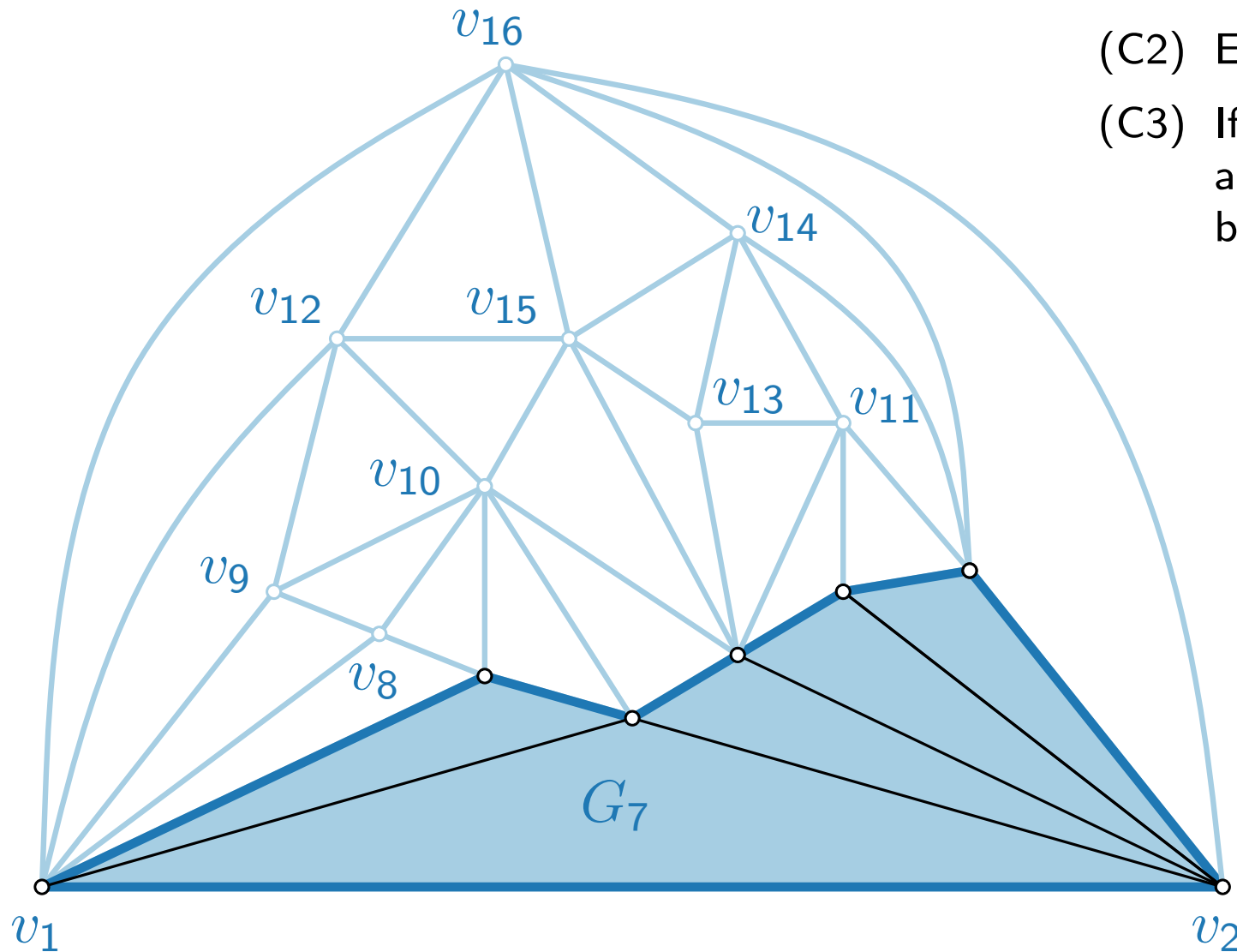
- (C1) Vertices  $\{v_1, \dots, v_k\}$  induce a biconnected internally triangulated graph; call it  $G_k$ .
- (C2) Edge  $(v_1, v_2)$  belongs to the outer face of  $G_k$ .
- (C3) If  $k < n$  then vertex  $v_{k+1}$  lies in the outer face of  $G_k$ , and the neighbors of  $v_{k+1}$  form a path on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

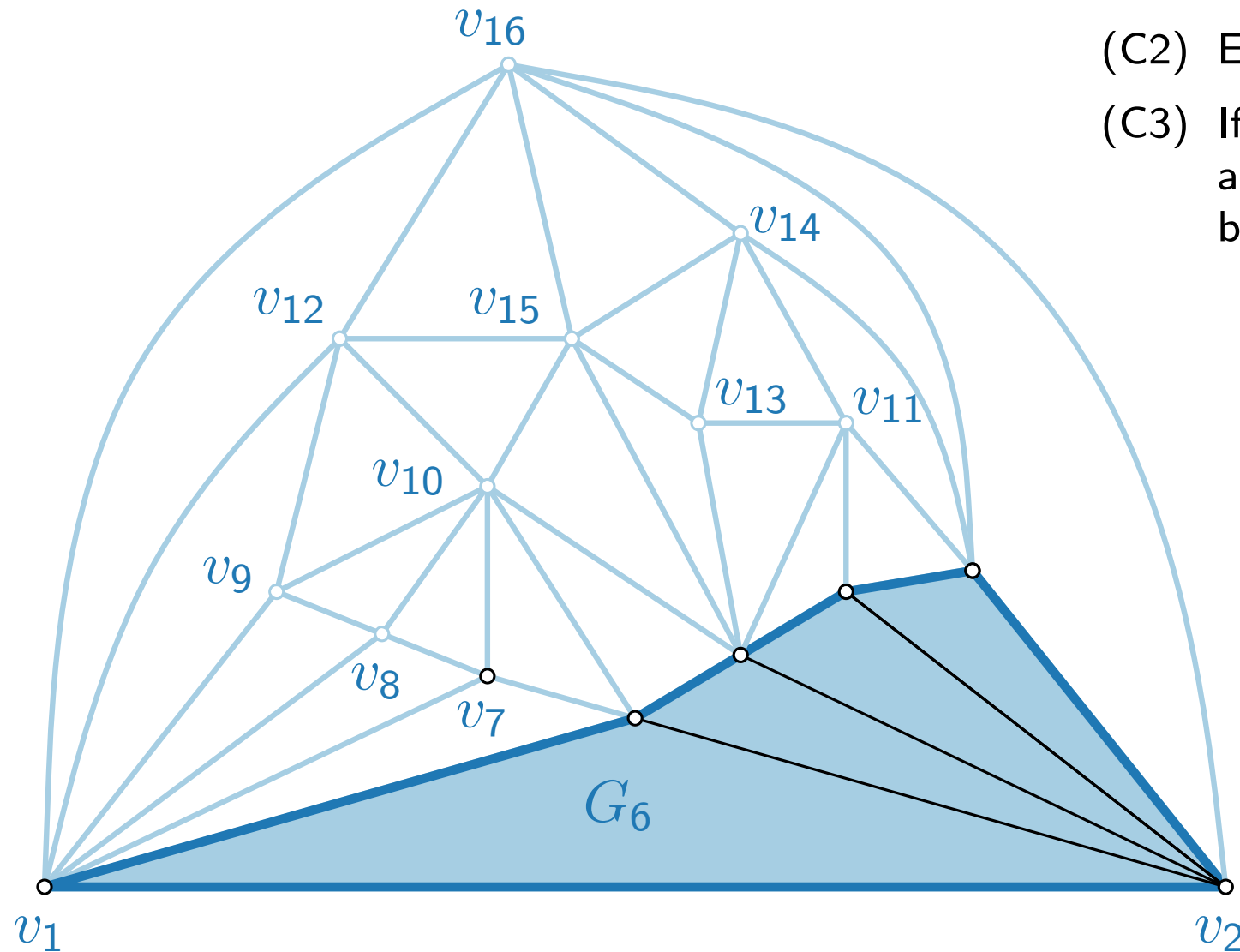




# Canonical Order – Example

- (C1) Vertices  $\{v_1, \dots, v_k\}$  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 the neighbors of  $v_{k+1}$  form a path on the boundary of  $G_k$ .

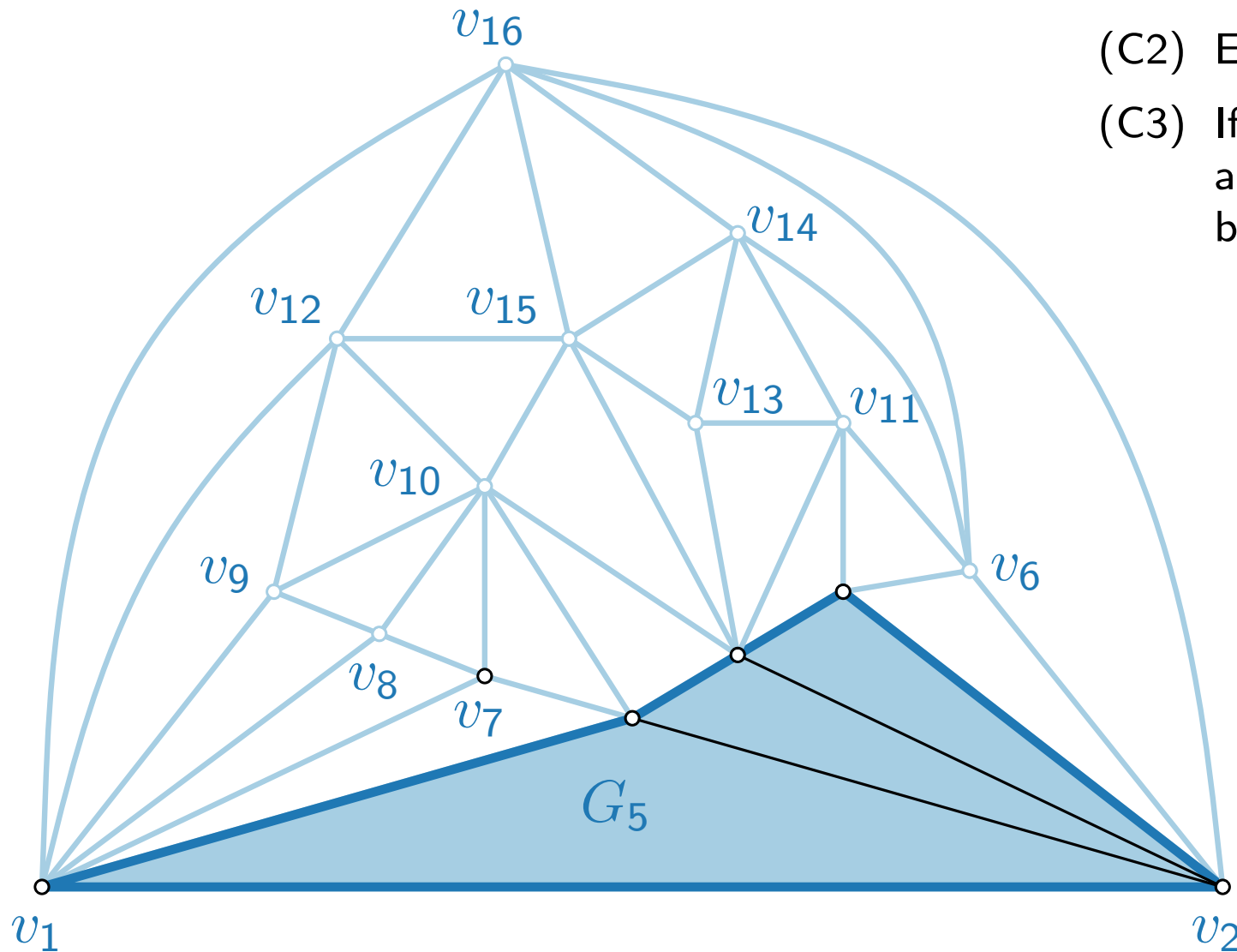




- (C1) Vertices  $\{v_1, \dots, v_k\}$  induce a biconnected internally triangulated graph; call it  $G_k$ .
- (C2) Edge  $(v_1, v_2)$  belongs to the outer face of  $G_k$ .
- (C3) If  $k < n$  then vertex  $v_{k+1}$  lies in the outer face of  $G_k$ , and the neighbors of  $v_{k+1}$  form a path on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

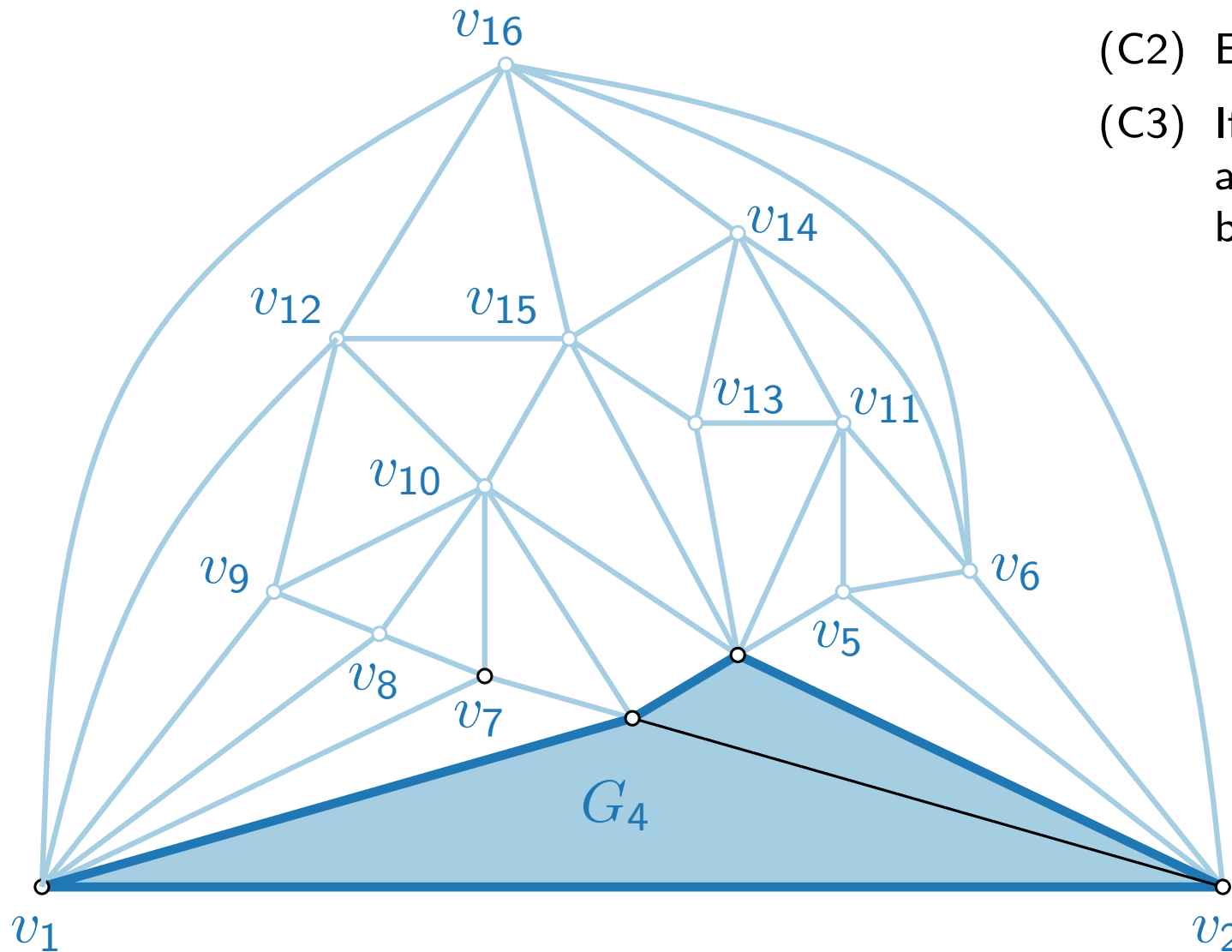
# Canonical Order – Example

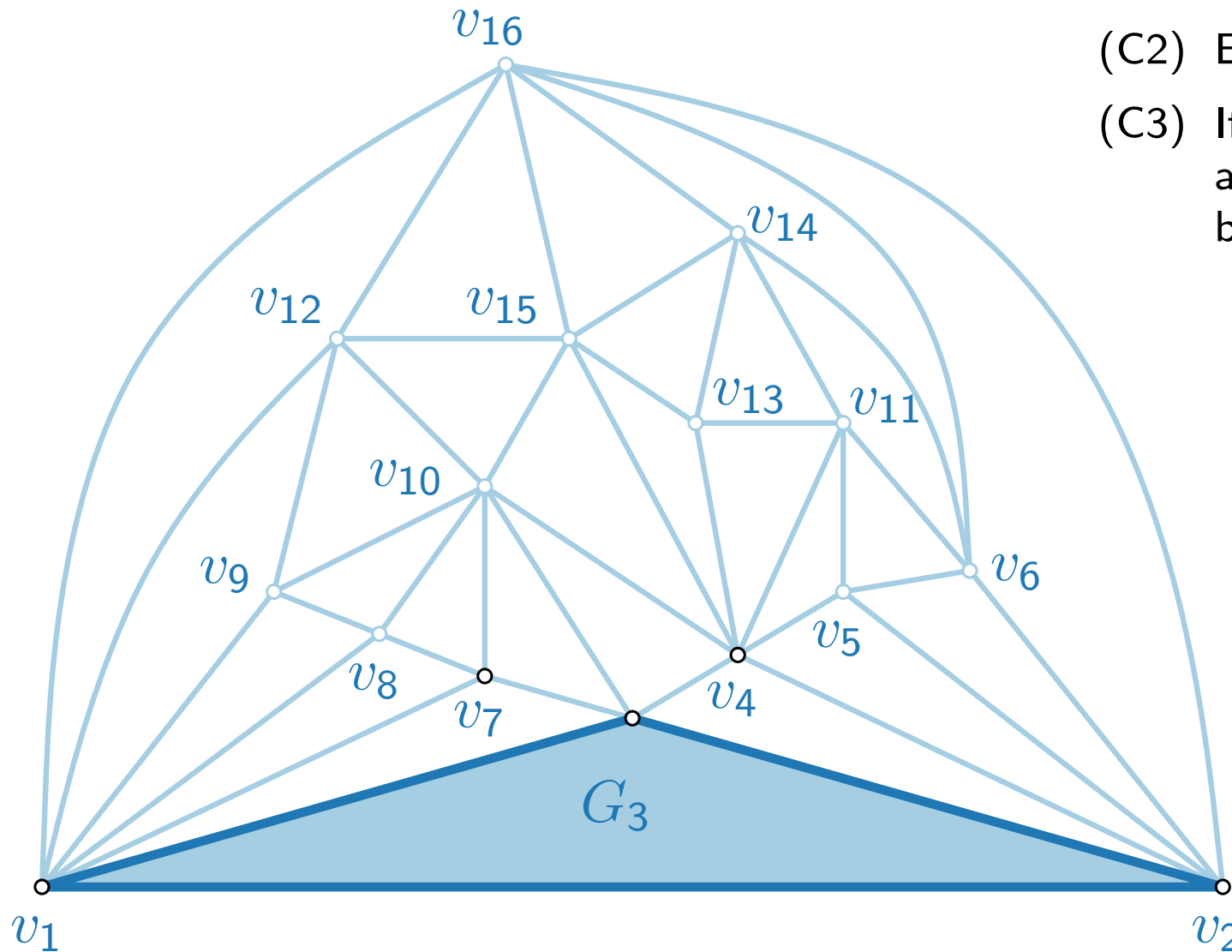
- (C1) Vertices  $\{v_1, \dots, v_k\}$  induce a biconnected internally triangulated graph; call it  $G_k$ .
- (C2) Edge  $(v_1, v_2)$  belongs to the outer face of  $G_k$ .
- (C3) If  $k < n$  then vertex  $v_{k+1}$  lies in the outer face of  $G_k$ , and the neighbors of  $v_{k+1}$  form a path on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



# Canonical Order – Example

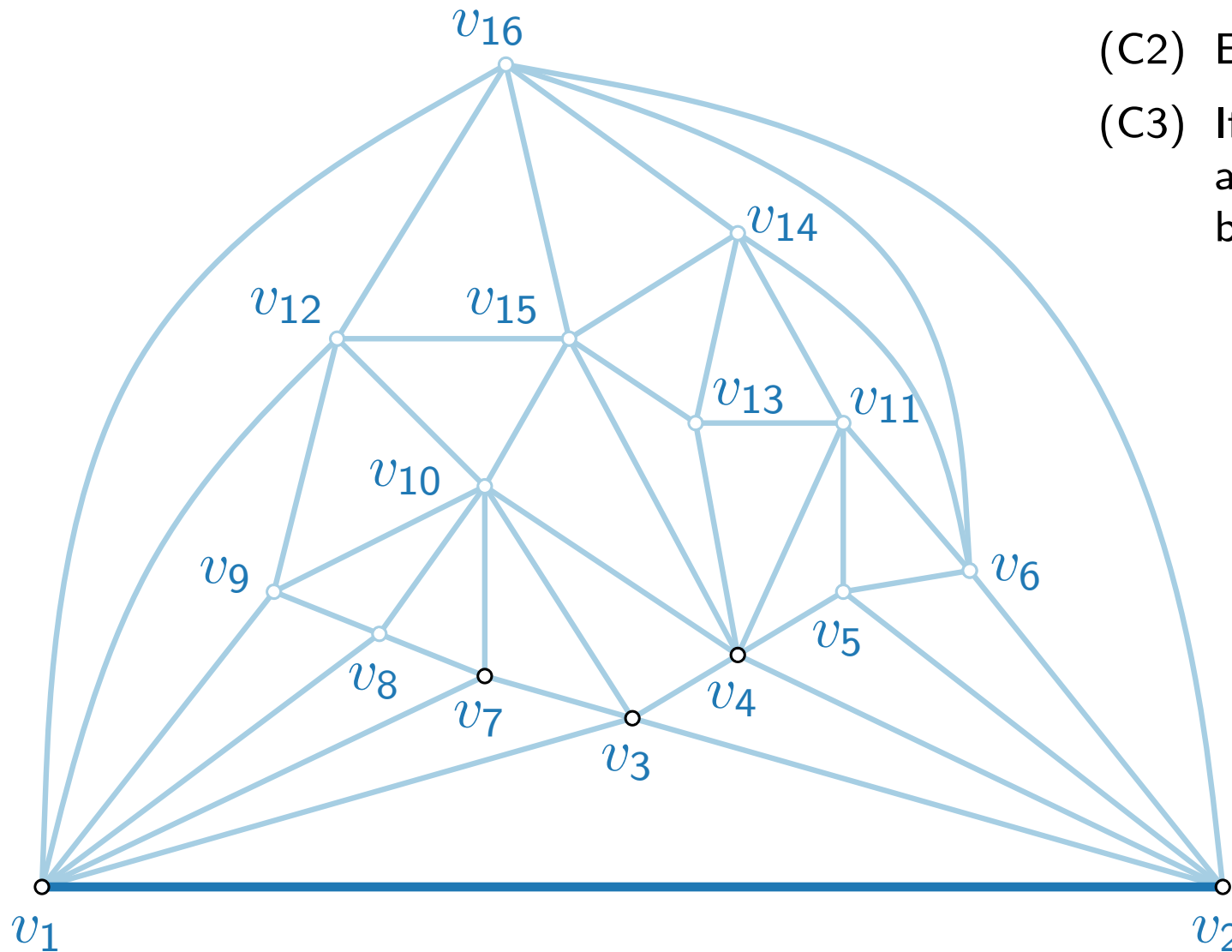
- (C1) Vertices  $\{v_1, \dots, v_k\}$  induce a biconnected internally triangulated graph; call it  $G_k$ .
- (C2) Edge  $(v_1, v_2)$  belongs to the outer face of  $G_k$ .
- (C3) If  $k < n$  then vertex  $v_{k+1}$  lies in the outer face of  $G_k$ , and the neighbors of  $v_{k+1}$  form a path on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





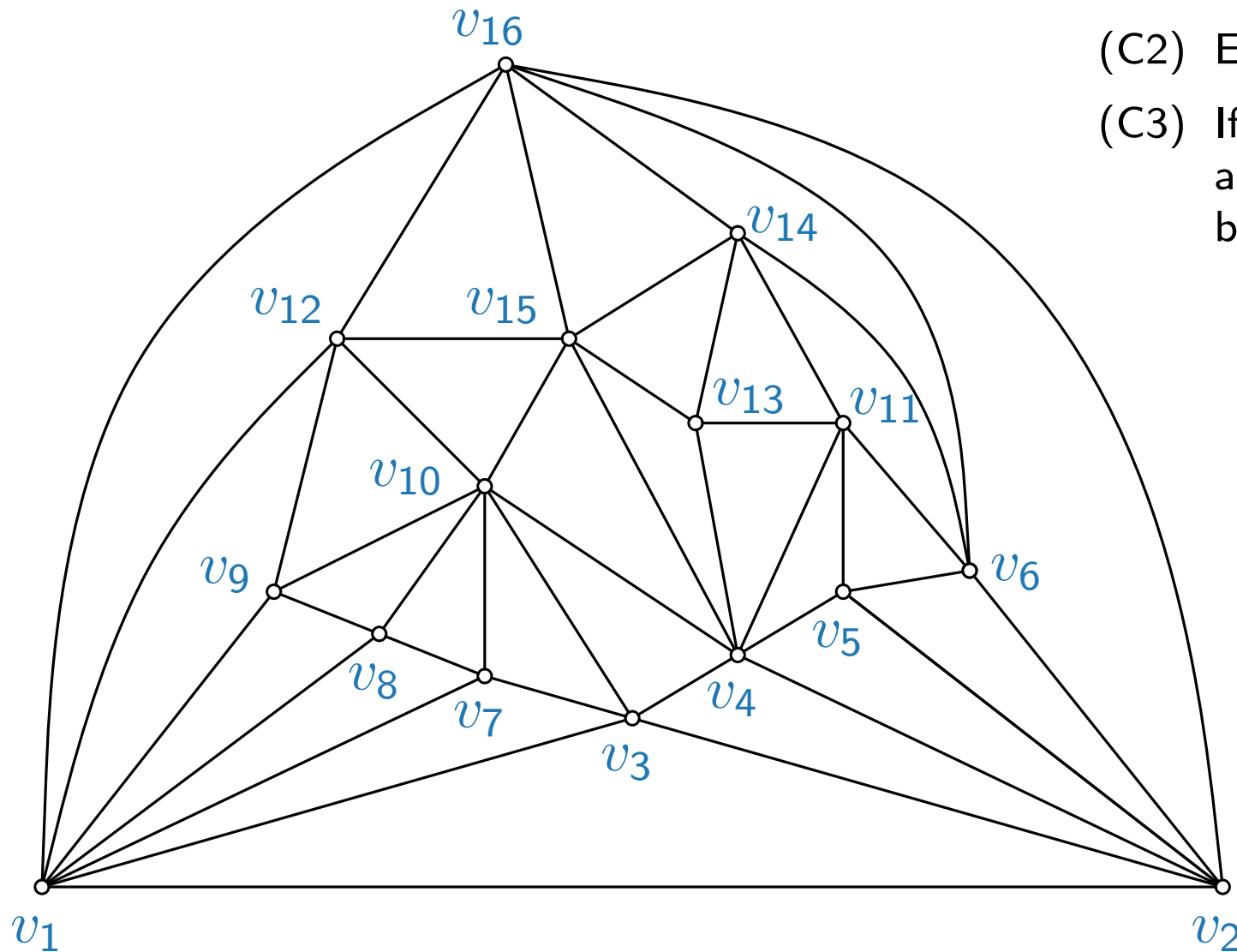
- (C1) Vertices  $\{v_1, \dots, v_k\}$  induce a biconnected internally triangulated graph; call it  $G_k$ .
- (C2) Edge  $(v_1, v_2)$  belongs to the outer face of  $G_k$ .
- (C3) If  $k < n$  then vertex  $v_{k+1}$  lies in the outer face of  $G_k$ , and the neighbors of  $v_{k+1}$  form a path on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

# Canonical Order – Example



- (C1) Vertices  $\{v_1, \dots, v_k\}$  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 the neighbors of  $v_{k+1}$  form a path on the boundary of  $G_k$ .

# Canonical Order – Example



- (C1) Vertices  $\{v_1, \dots, v_k\}$  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 the neighbors of  $v_{k+1}$  form a path on the boundary of  $G_k$ .

# Canonical Order – Existence

**Lemma.**

Every triangulated plane graph has a canonical order.

- (C1)  $G_k$  biconnected and internally triangulated
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$



# Canonical Order – Existence

## Lemma.

Every triangulated plane graph has a canonical order.

Base Case:

Induction hypothesis:

Induction step:

- (C1)  $G_k$  biconnected and internally triangulated
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$

# Canonical Order – Existence

## Lemma.

Every triangulated plane graph has a canonical order.

### Base Case:

Let  $G_n = G$ , and let  $v_1, v_2, v_n$  be the vertices of the outer face of  $G_n$ .

### Induction hypothesis:

### Induction step:

- (C1)  $G_k$  biconnected and internally triangulated
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$

# Canonical Order – Existence

## Lemma.

Every triangulated plane graph has a canonical order.

### Base Case:

Let  $G_n = G$ , and let  $v_1, v_2, v_n$  be the vertices of the outer face of  $G_n$ .

### Induction hypothesis:

### Induction step:

- (C1)  $G_k$  biconnected and internally triangulated ✓
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$

# Canonical Order – Existence

## Lemma.

Every triangulated plane graph has a canonical order.

### Base Case:

Let  $G_n = G$ , and let  $v_1, v_2, v_n$  be the vertices of the outer face of  $G_n$ .

### Induction hypothesis:

### Induction step:

- (C1)  $G_k$  biconnected and internally triangulated ✓
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$  ✓
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$

# Canonical Order – Existence

## Lemma.

Every triangulated plane graph has a canonical order.

### Base Case:

Let  $G_n = G$ , and let  $v_1, v_2, v_n$  be the vertices of the outer face of  $G_n$ .

### Induction hypothesis:

### Induction step:

- (C1)  $G_k$  biconnected and internally triangulated ✓
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$  ✓
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$  ✓

# Canonical Order – Existence

## Lemma.

Every triangulated plane graph has a canonical order.

### Base Case:

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:

### Induction step:

- (C1)  $G_k$  biconnected and internally triangulated ✓
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$  ✓
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$  ✓

# Canonical Order – Existence

## Lemma.

Every triangulated plane graph has a canonical order.

### Base Case:

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  $v_{n-1}, \dots, v_{k+1}$  have been chosen such that conditions (C1)–(C3) hold for  $k+1 \leq i \leq n$ .

### Induction step:

- (C1)  $G_k$  biconnected and internally triangulated
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$

# Canonical Order – Existence

## Lemma.

Every triangulated plane graph has a canonical order.

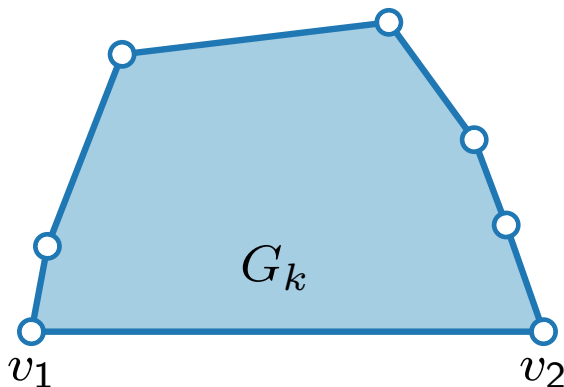
### Base Case:

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  $v_{n-1}, \dots, v_{k+1}$  have been chosen such that conditions (C1)–(C3) hold for  $k+1 \leq i \leq n$ .

Induction step: Consider  $G_k$ .



- (C1)  $G_k$  biconnected and internally triangulated
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$



# Canonical Order – Existence

## Lemma.

Every triangulated plane graph has a canonical order.

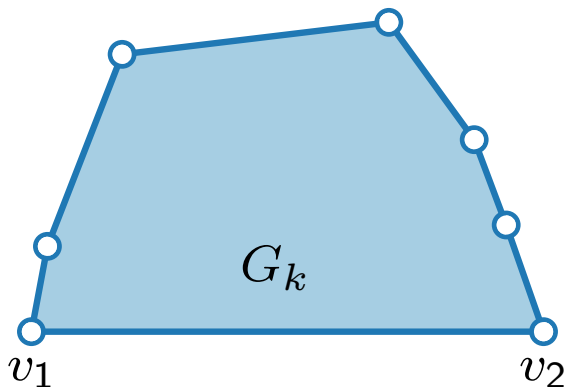
### Base Case:

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  $v_{n-1}, \dots, v_{k+1}$  have been chosen such that conditions (C1)–(C3) hold for  $k+1 \leq i \leq n$ .

Induction step: Consider  $G_k$ . We search for  $v_k$ .



- (C1)  $G_k$  biconnected and internally triangulated
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$

# Canonical Order – Existence

## Lemma.

Every triangulated plane graph has a canonical order.

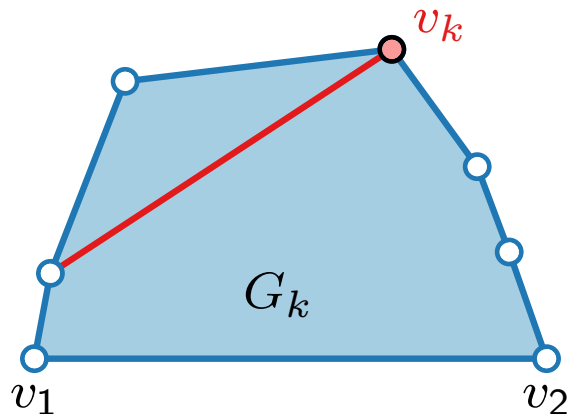
### Base Case:

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  $v_{n-1}, \dots, v_{k+1}$  have been chosen such that conditions (C1)–(C3) hold for  $k+1 \leq i \leq n$ .

**Induction step:** Consider  $G_k$ . We search for  $v_k$ .



- (C1)  $G_k$  biconnected and internally triangulated
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$

# Canonical Order – Existence

## Lemma.

Every triangulated plane graph has a canonical order.

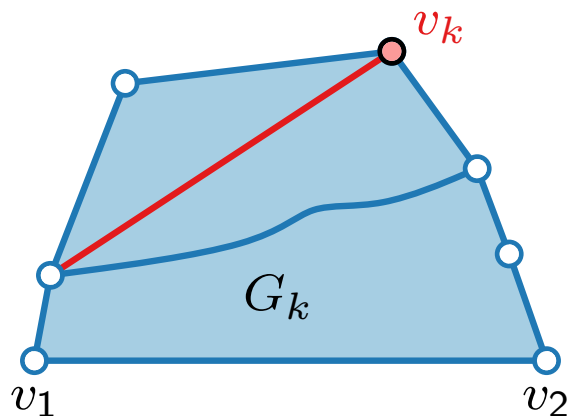
### Base Case:

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  $v_{n-1}, \dots, v_{k+1}$  have been chosen such that conditions (C1)–(C3) hold for  $k+1 \leq i \leq n$ .

Induction step: Consider  $G_k$ . We search for  $v_k$ .



- (C1)  $G_k$  biconnected and internally triangulated
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$

# Canonical Order – Existence

## Lemma.

Every triangulated plane graph has a canonical order.

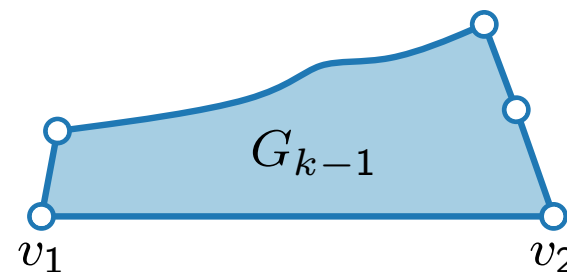
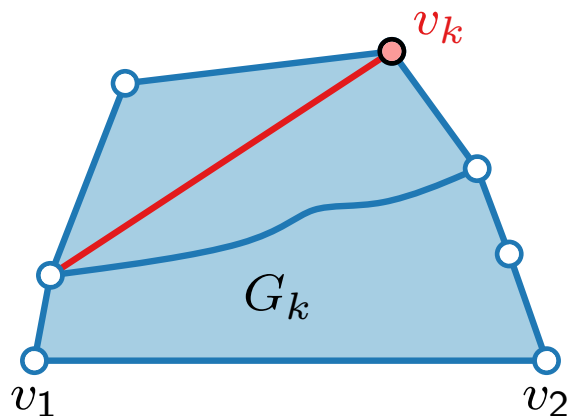
### Base Case:

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  $v_{n-1}, \dots, v_{k+1}$  have been chosen such that conditions (C1)–(C3) hold for  $k+1 \leq i \leq n$ .

Induction step: Consider  $G_k$ . We search for  $v_k$ .



- (C1)  $G_k$  biconnected and internally triangulated
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$

# Canonical Order – Existence

## Lemma.

Every triangulated plane graph has a canonical order.

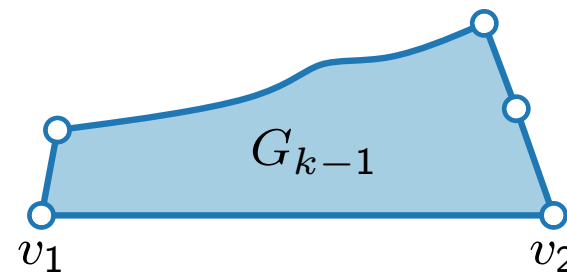
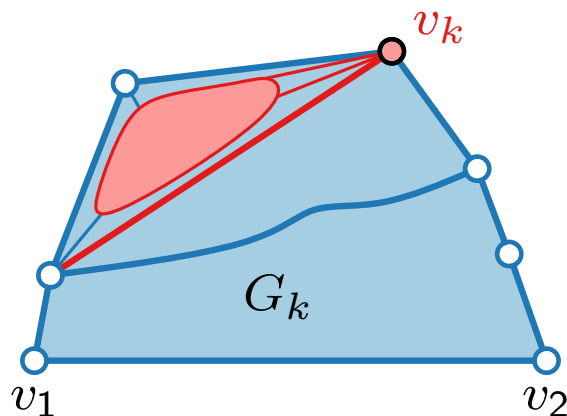
### Base Case:

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  $v_{n-1}, \dots, v_{k+1}$  have been chosen such that conditions (C1)–(C3) hold for  $k+1 \leq i \leq n$ .

Induction step: Consider  $G_k$ . We search for  $v_k$ .



- (C1)  $G_k$  biconnected and internally triangulated
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$

# Canonical Order – Existence

## Lemma.

Every triangulated plane graph has a canonical order.

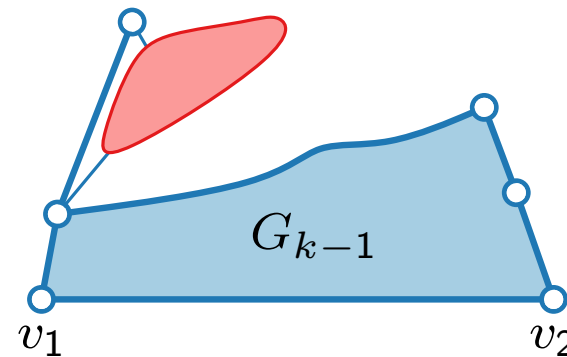
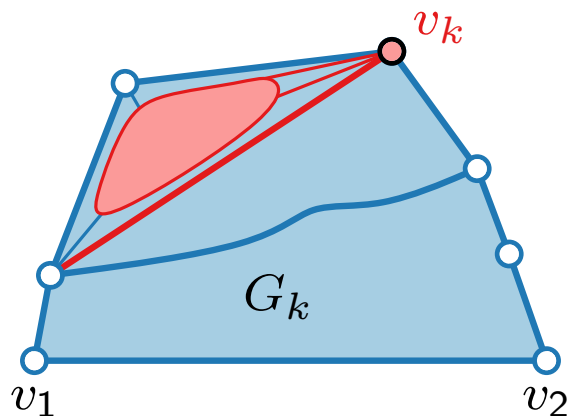
### Base Case:

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  $v_{n-1}, \dots, v_{k+1}$  have been chosen such that conditions (C1)–(C3) hold for  $k+1 \leq i \leq n$ .

Induction step: Consider  $G_k$ . We search for  $v_k$ .



- (C1)  $G_k$  biconnected and internally triangulated
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$

# Canonical Order – Existence

## Lemma.

Every triangulated plane graph has a canonical order.

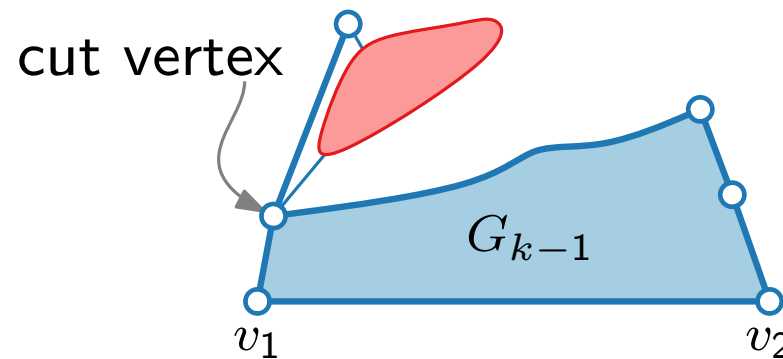
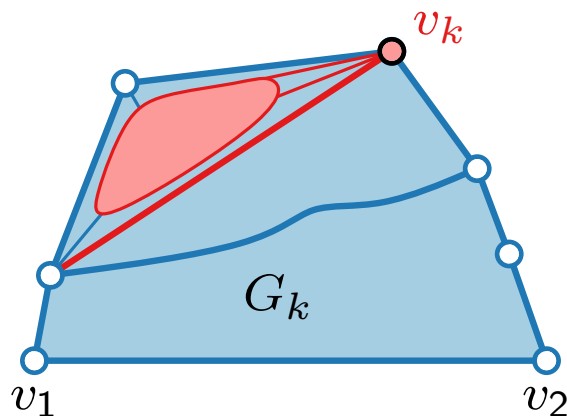
### Base Case:

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  $v_{n-1}, \dots, v_{k+1}$  have been chosen such that conditions (C1)–(C3) hold for  $k+1 \leq i \leq n$ .

Induction step: Consider  $G_k$ . We search for  $v_k$ .



- (C1)  $G_k$  biconnected and internally triangulated
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$

# Canonical Order – Existence

## Lemma.

Every triangulated plane graph has a canonical order.

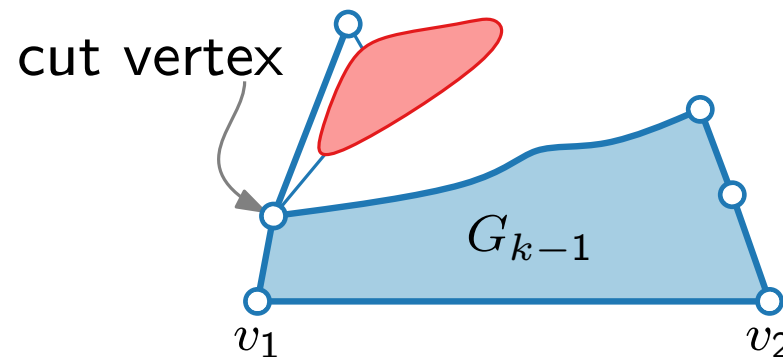
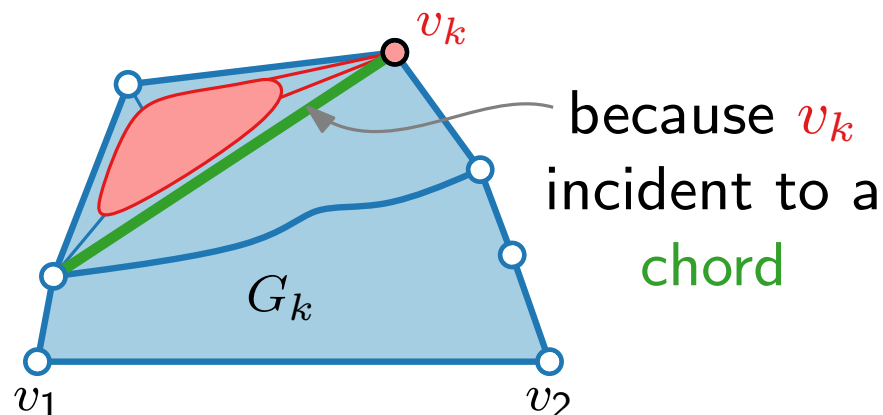
### Base Case:

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  $v_{n-1}, \dots, v_{k+1}$  have been chosen such that conditions (C1)–(C3) hold for  $k+1 \leq i \leq n$ .

**Induction step:** Consider  $G_k$ . We search for  $v_k$ .



- (C1)  $G_k$  biconnected and internally triangulated
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$



# Canonical Order – Existence

## Lemma.

Every triangulated plane graph has a canonical order.

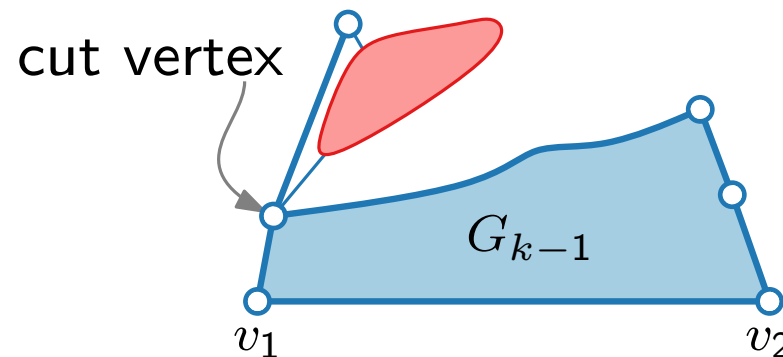
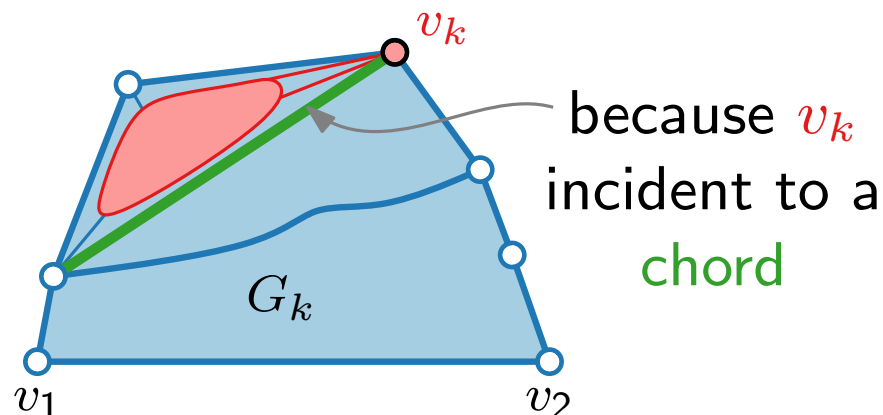
### Base Case:

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  $v_{n-1}, \dots, v_{k+1}$  have been chosen such that conditions (C1)–(C3) hold for  $k+1 \leq i \leq n$ .

**Induction step:** Consider  $G_k$ . We search for  $v_k$ .



- (C1)  $G_k$  biconnected and internally triangulated
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$

Have to show:

# Canonical Order – Existence

## Lemma.

Every triangulated plane graph has a canonical order.

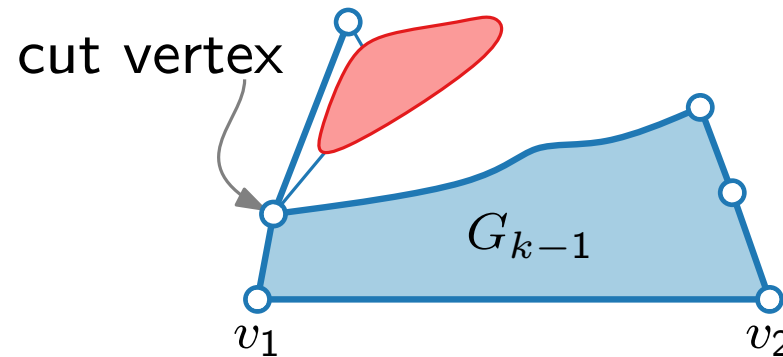
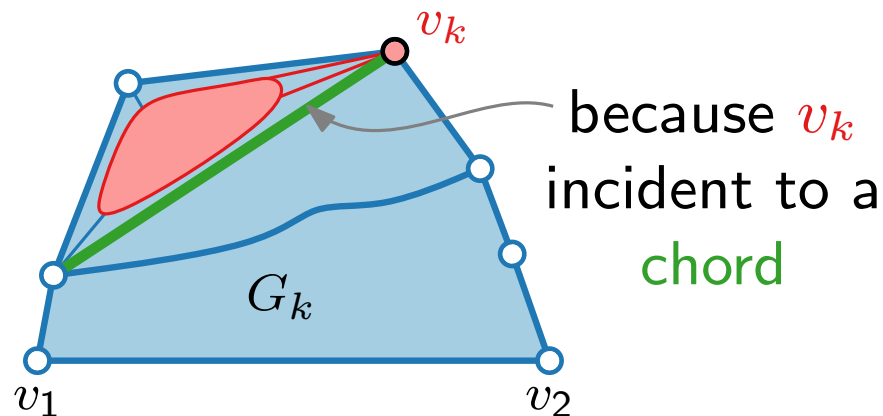
### Base Case:

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  $v_{n-1}, \dots, v_{k+1}$  have been chosen such that conditions (C1)–(C3) hold for  $k+1 \leq i \leq n$ .

**Induction step:** Consider  $G_k$ . We search for  $v_k$ .



- (C1)  $G_k$  biconnected and internally triangulated
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$

### Have to show:

1.  $v_k$  not incident to chord is sufficient

# Canonical Order – Existence

## Lemma.

Every triangulated plane graph has a canonical order.

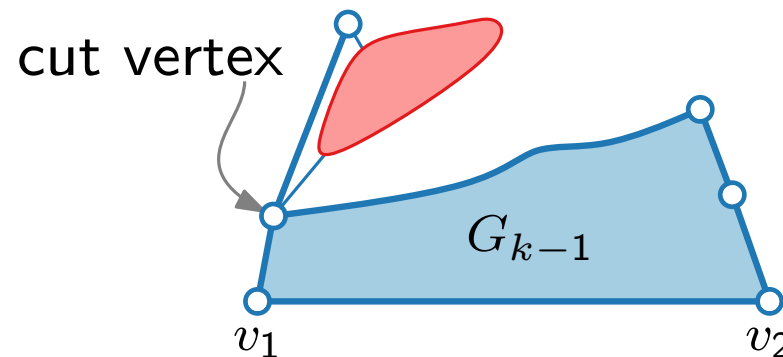
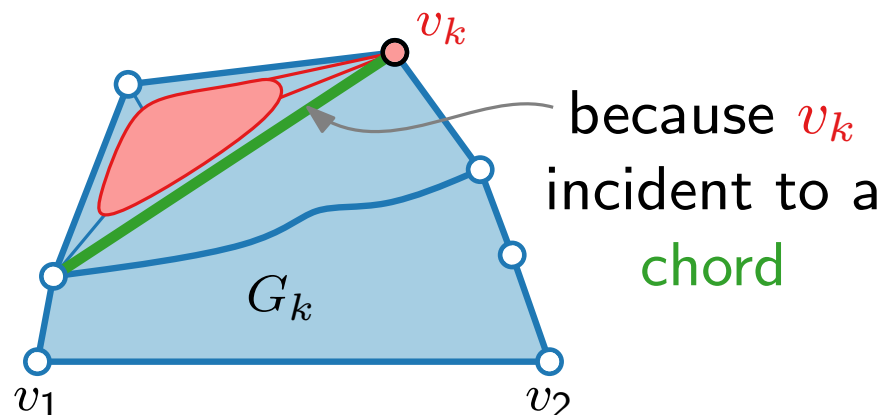
### Base Case:

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  $v_{n-1}, \dots, v_{k+1}$  have been chosen such that conditions (C1)–(C3) hold for  $k+1 \leq i \leq n$ .

Induction step: Consider  $G_k$ . We search for  $v_k$ .



- (C1)  $G_k$  biconnected and internally triangulated
- (C2)  $(v_1, v_2)$  on outer face of  $G_k$
- (C3)  $k < n \Rightarrow v_{k+1}$  in outer face of  $G_k$ , neighbors of  $v_{k+1}$  form path on boundary of  $G_k$

### Have to show:

1.  $v_k$  not incident to chord is sufficient
2. Such  $v_k$  exists

# Canonical Order – Existence

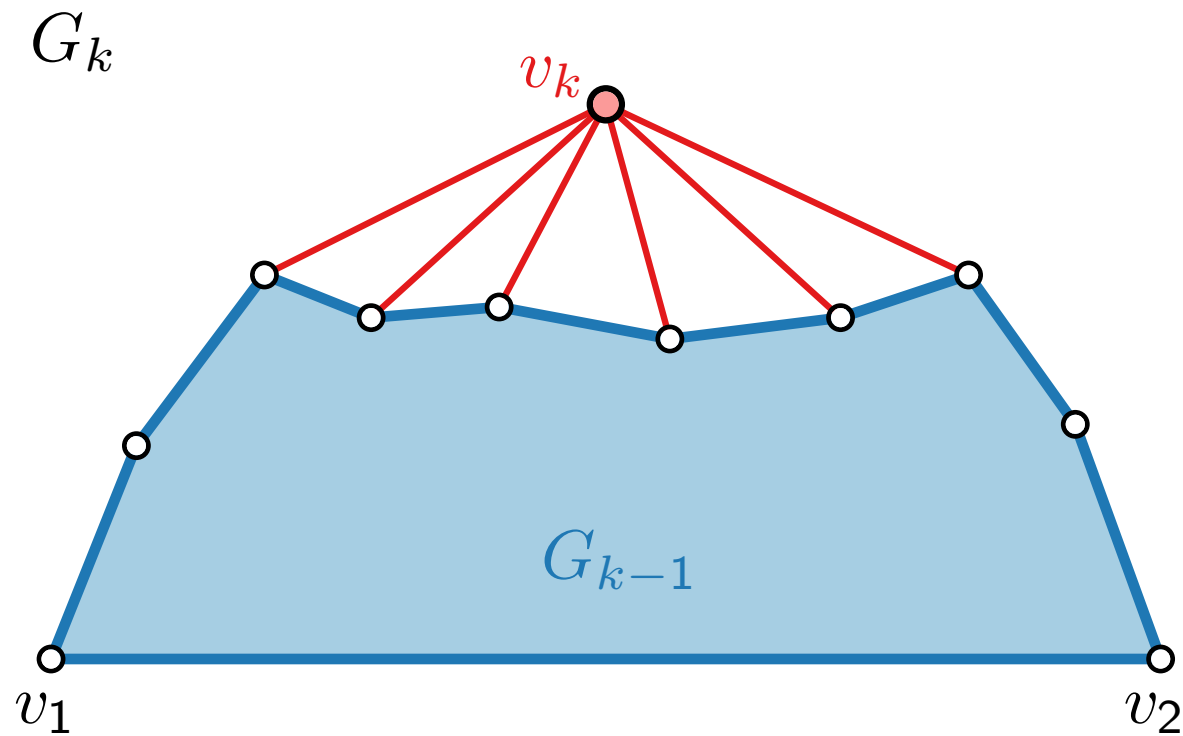
## Claim 1.

If  $v_k$  is not incident to a chord,  
then  $G_{k-1}$  is biconnected.

# Canonical Order – Existence

## Claim 1.

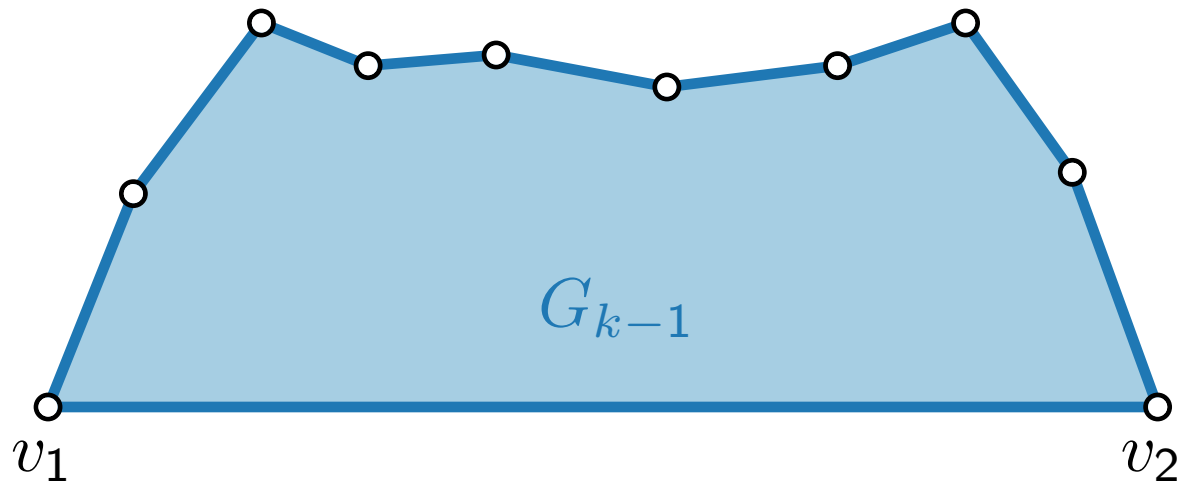
If  $v_k$  is not incident to a **chord**,  
then  $G_{k-1}$  is biconnected.



# Canonical Order – Existence

## Claim 1.

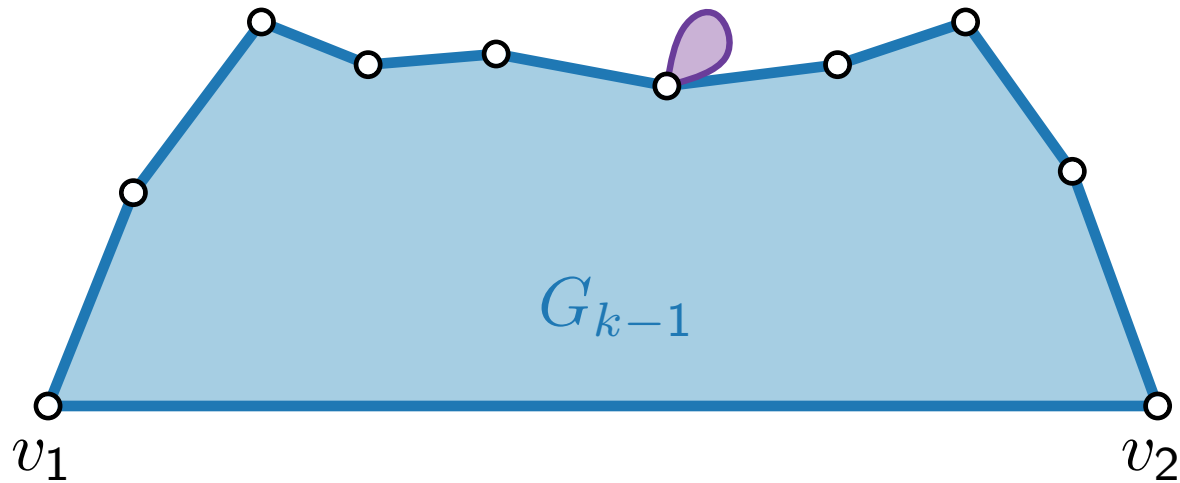
If  $v_k$  is not incident to a chord,  
then  $G_{k-1}$  is biconnected.



# Canonical Order – Existence

## Claim 1.

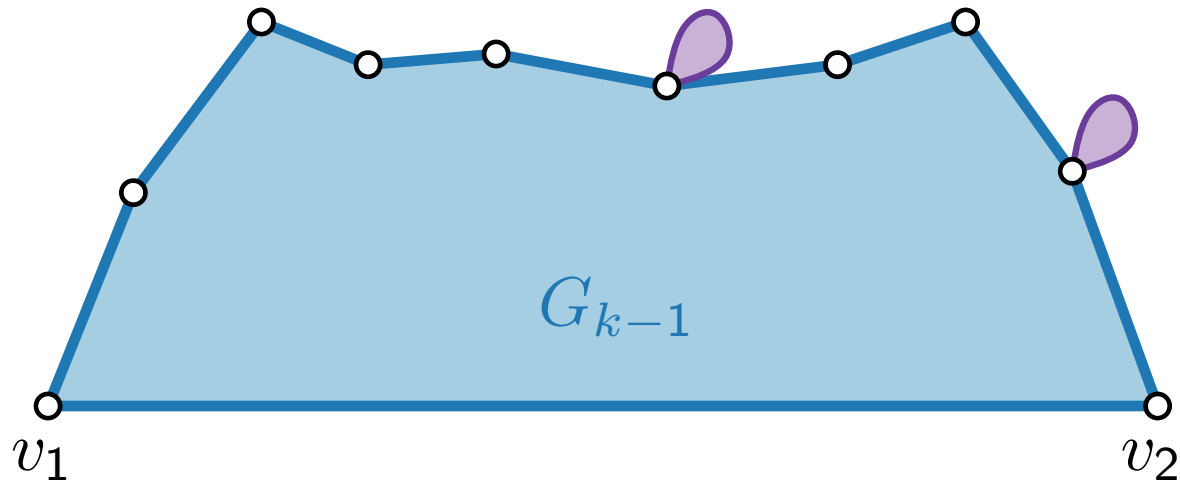
If  $v_k$  is not incident to a **chord**,  
then  $G_{k-1}$  is biconnected.



# Canonical Order – Existence

## Claim 1.

If  $v_k$  is not incident to a **chord**,  
then  $G_{k-1}$  is biconnected.

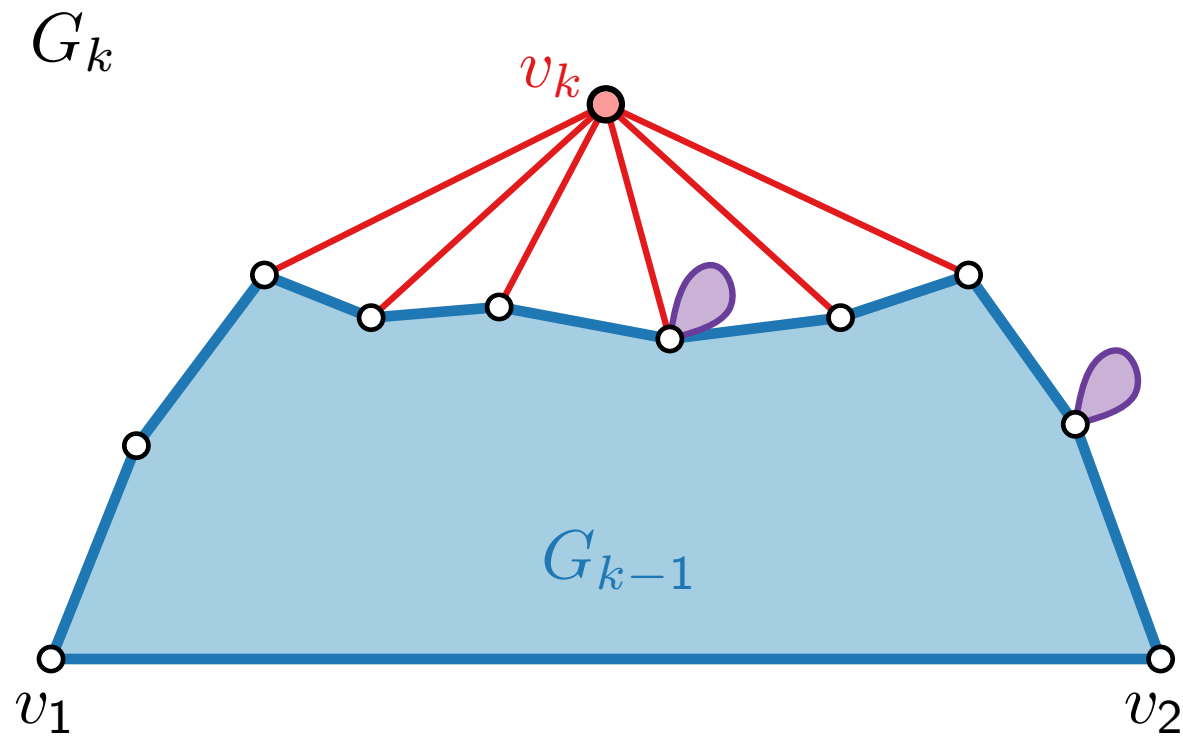




# Canonical Order – Existence

## Claim 1.

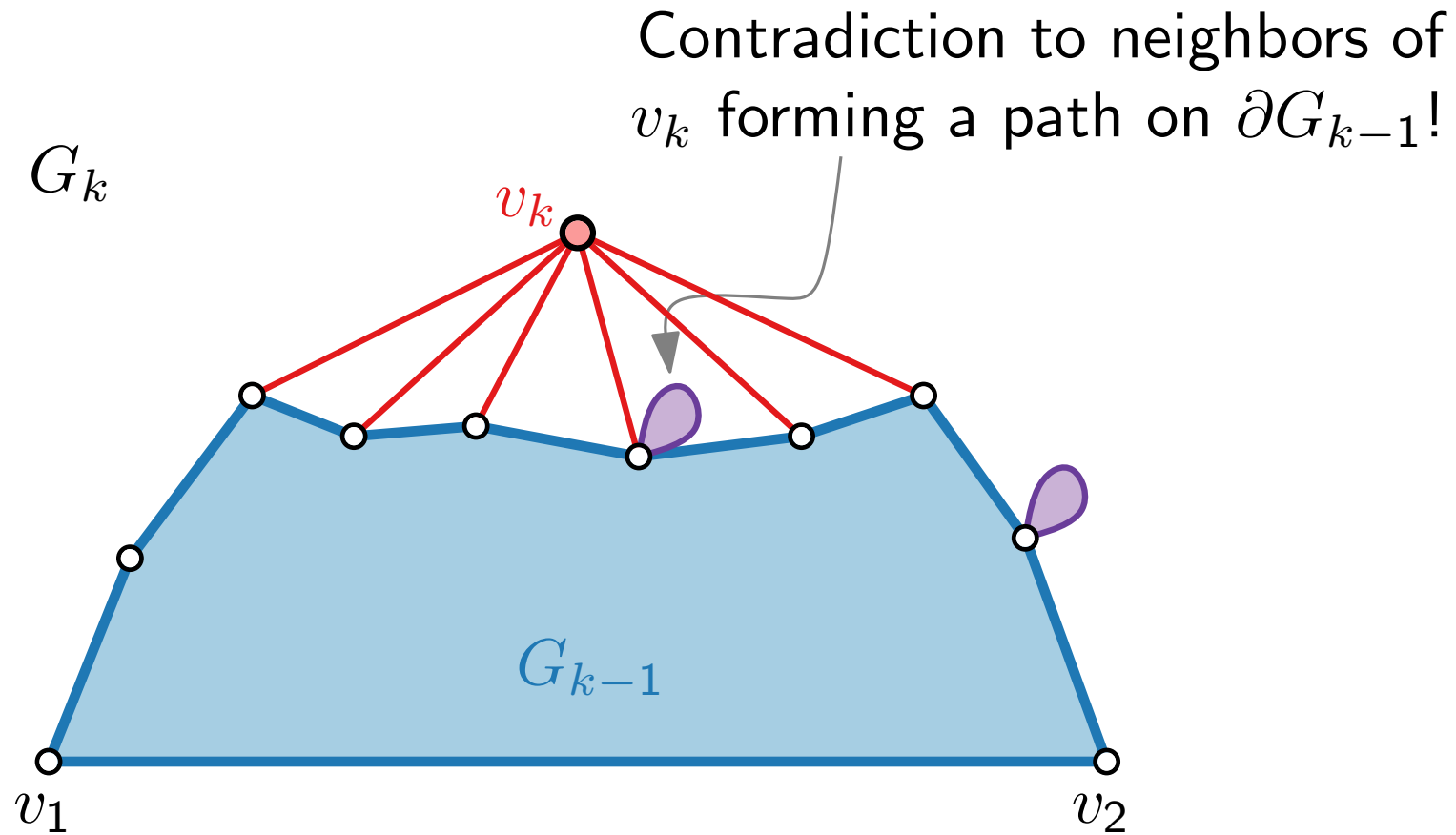
If  $v_k$  is not incident to a **chord**,  
then  $G_{k-1}$  is biconnected.



# Canonical Order – Existence

## Claim 1.

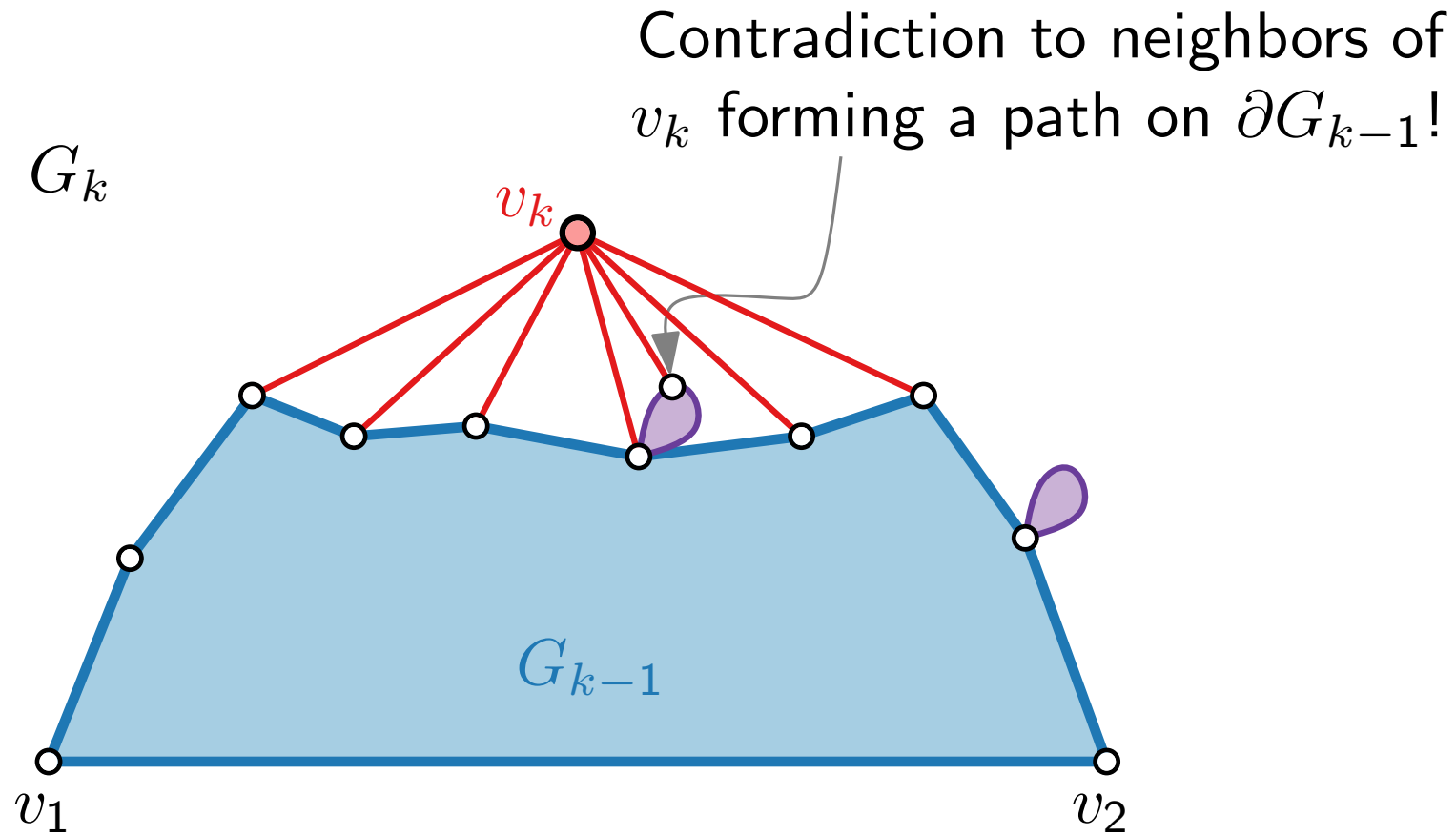
If  $v_k$  is not incident to a **chord**,  
then  $G_{k-1}$  is biconnected.



# Canonical Order – Existence

## Claim 1.

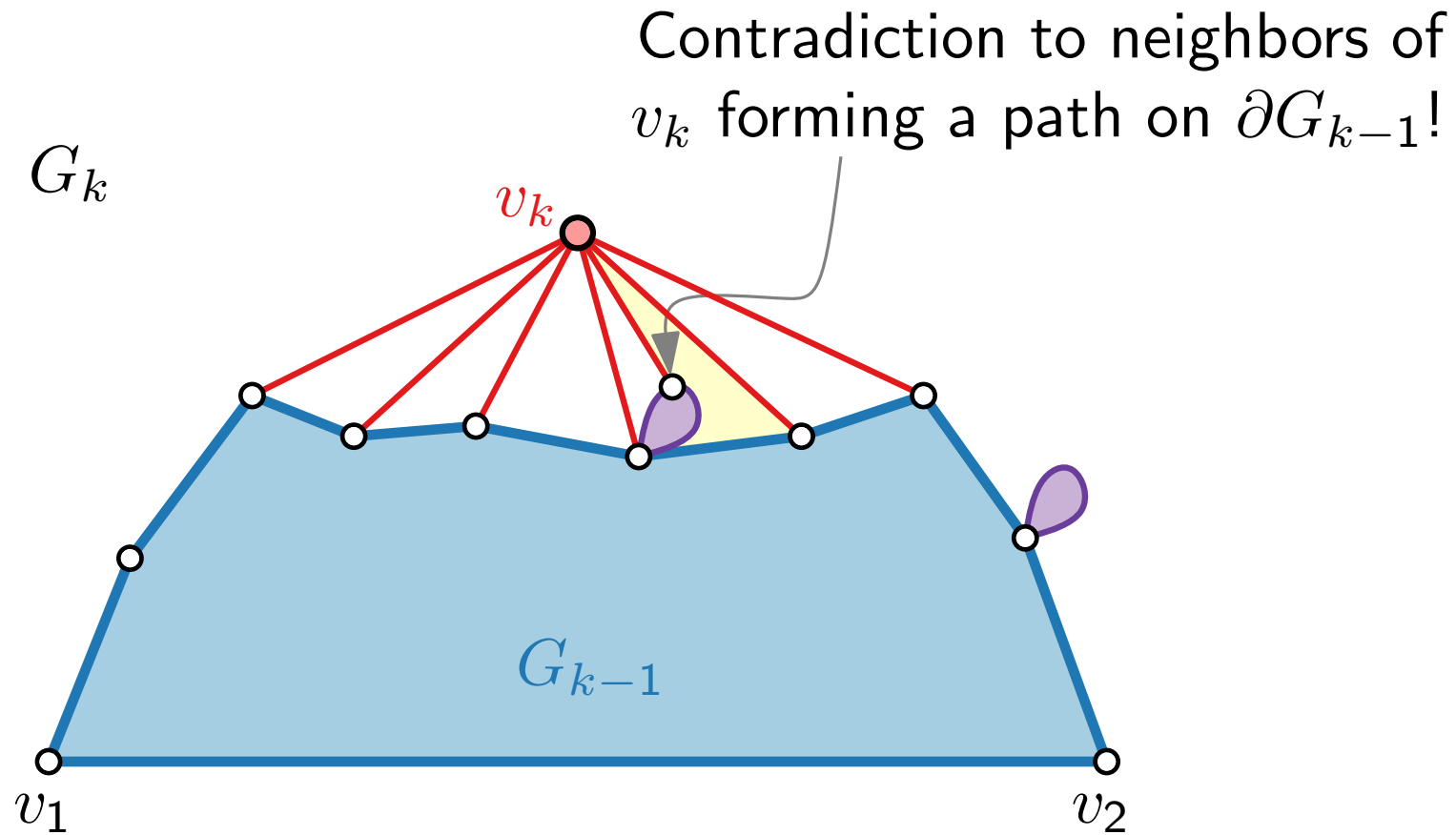
If  $v_k$  is not incident to a **chord**,  
then  $G_{k-1}$  is biconnected.



# Canonical Order – Existence

## Claim 1.

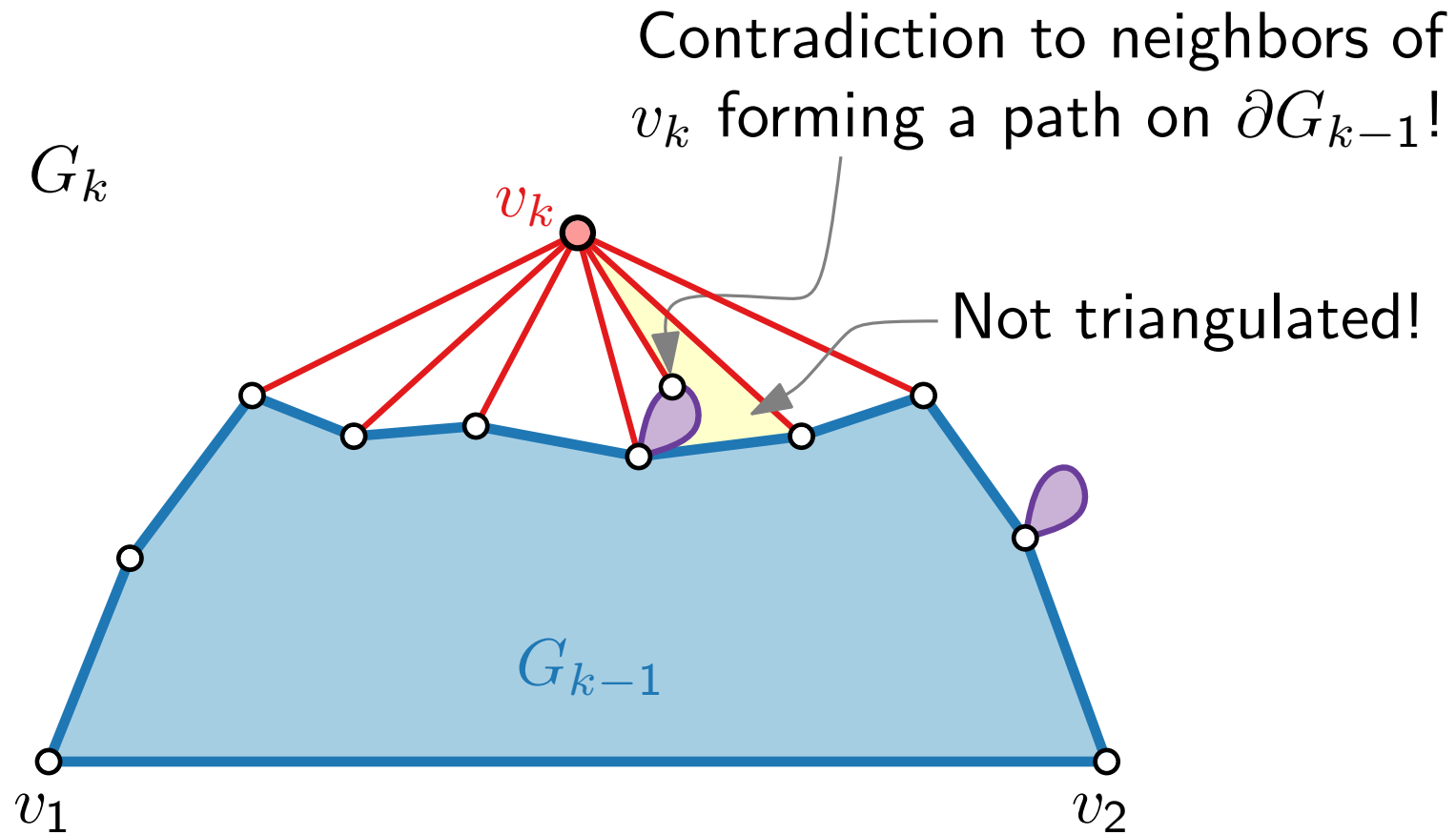
If  $v_k$  is not incident to a **chord**,  
then  $G_{k-1}$  is biconnected.



# Canonical Order – Existence

## Claim 1.

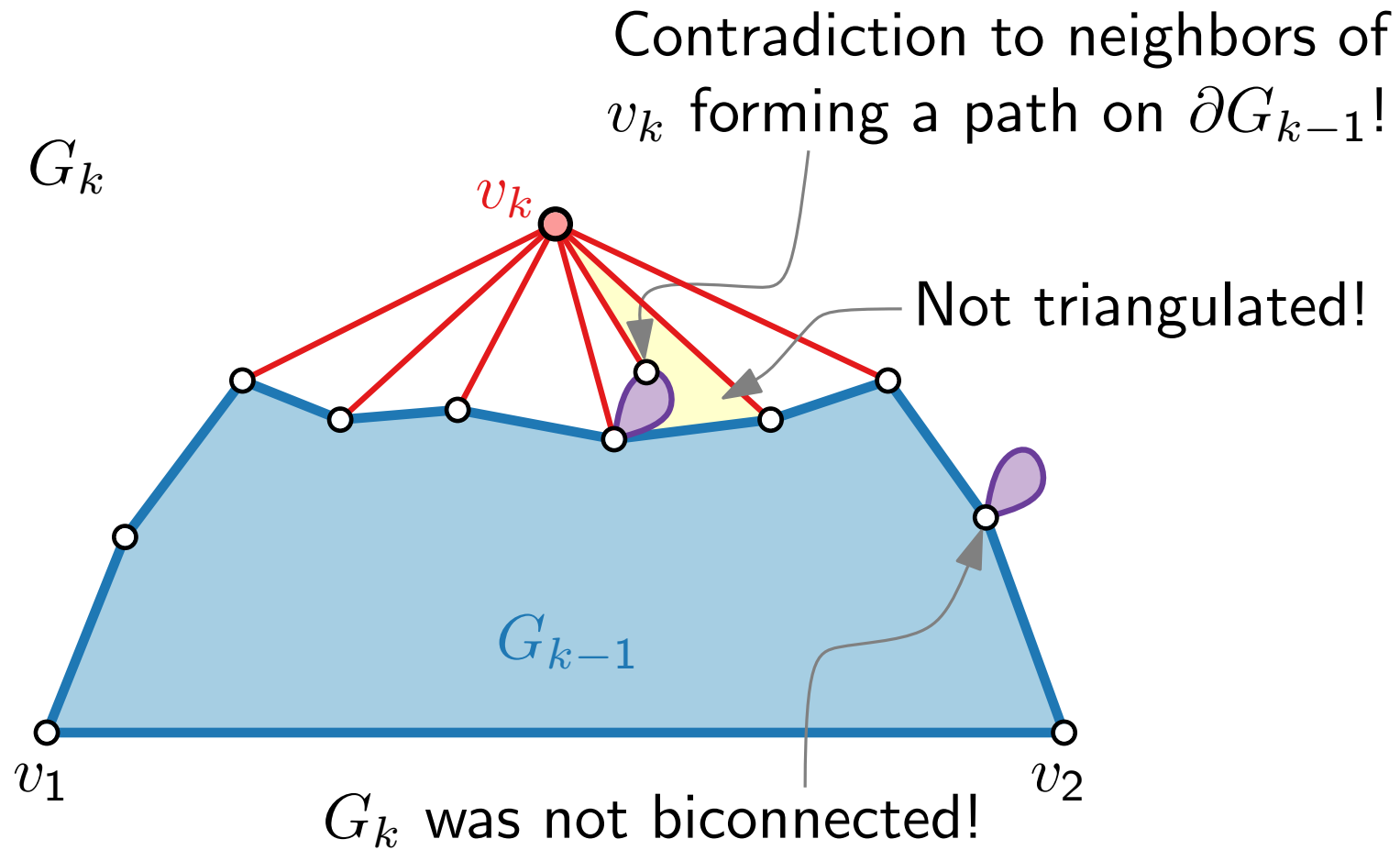
If  $v_k$  is not incident to a **chord**,  
then  $G_{k-1}$  is biconnected.



# Canonical Order – Existence

## Claim 1.

If  $v_k$  is not incident to a **chord**,  
then  $G_{k-1}$  is biconnected.



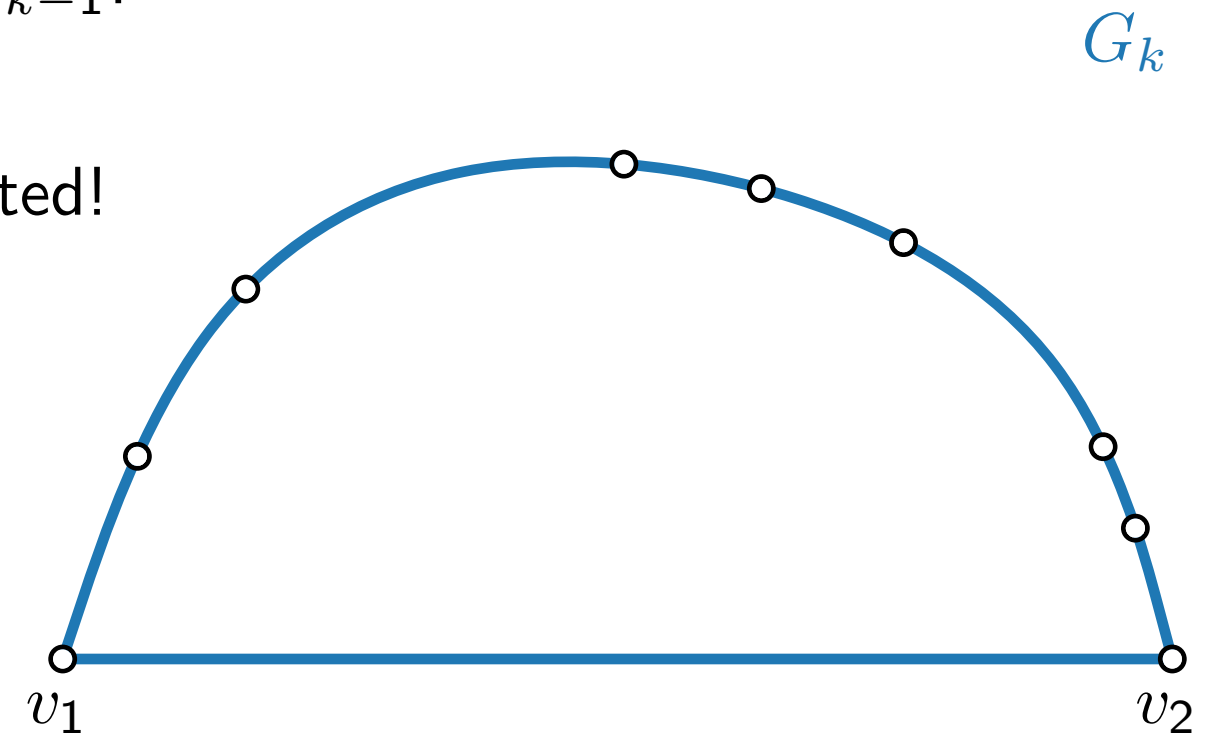
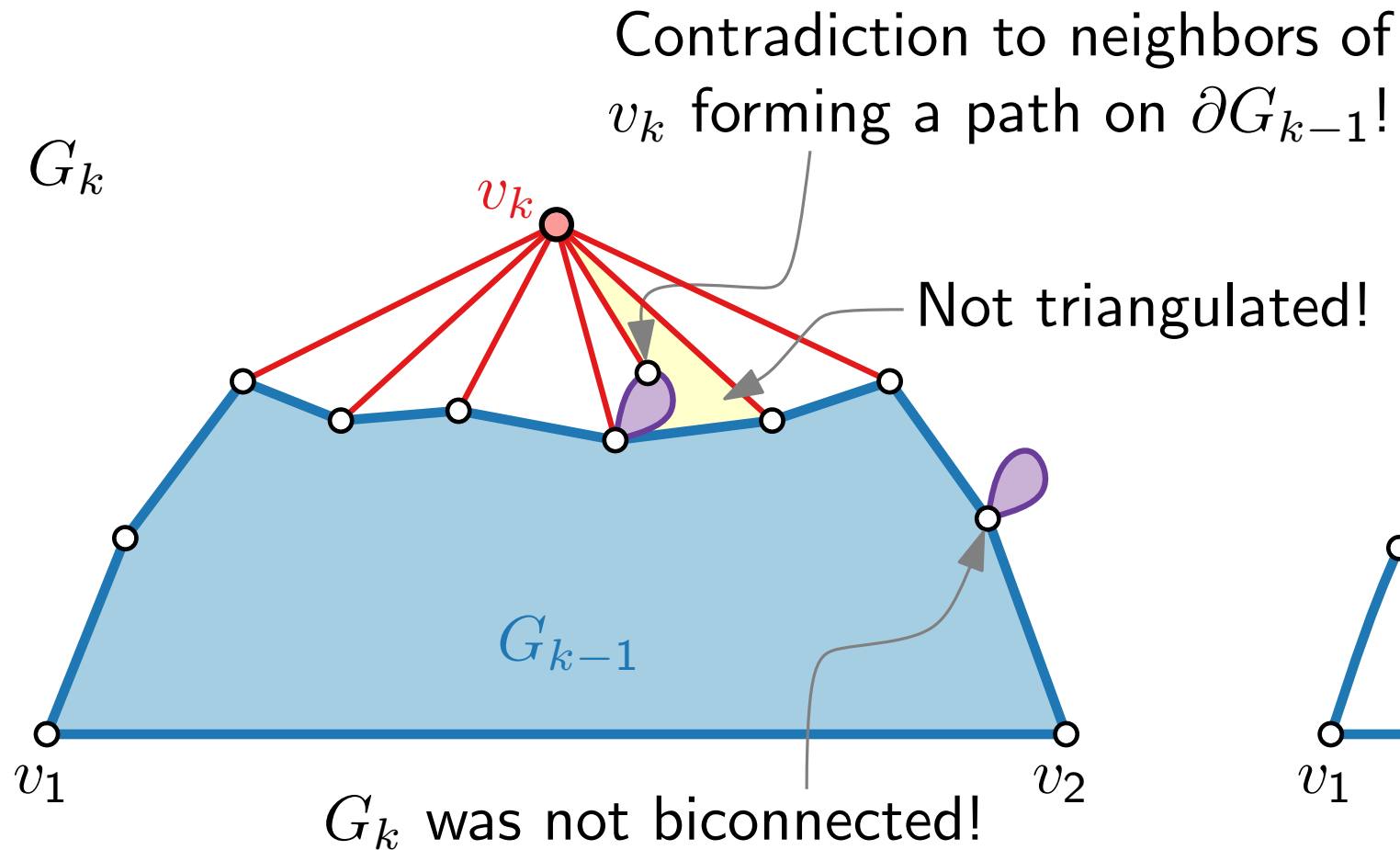
# Canonical Order – Existence

## Claim 1.

If  $v_k$  is not incident to a **chord**,  
then  $G_{k-1}$  is biconnected.

## Claim 2.

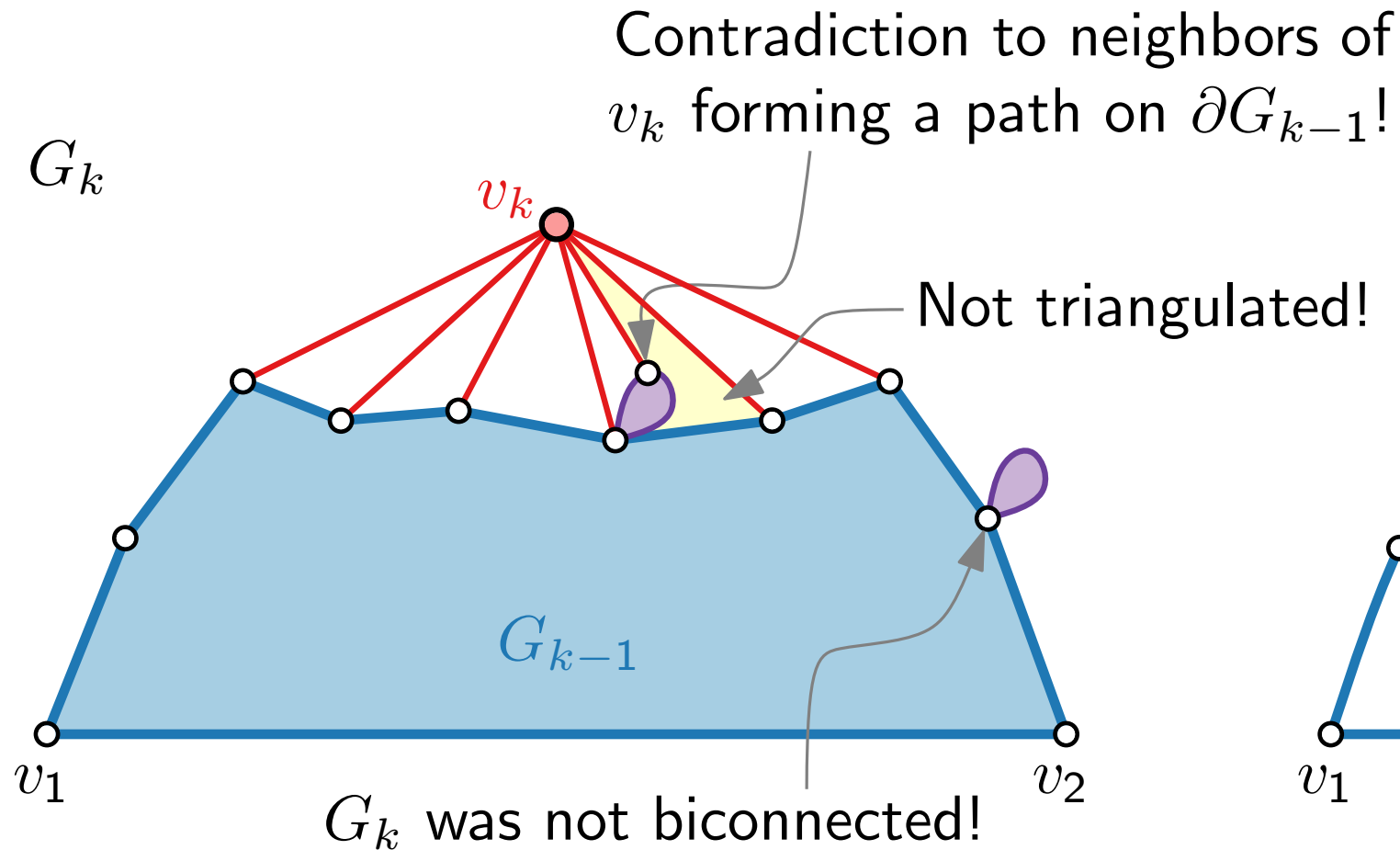
There exists a vertex in  $G_k$  that is not  
incident to a chord as choice for  $v_k$ .



# Canonical Order – Existence

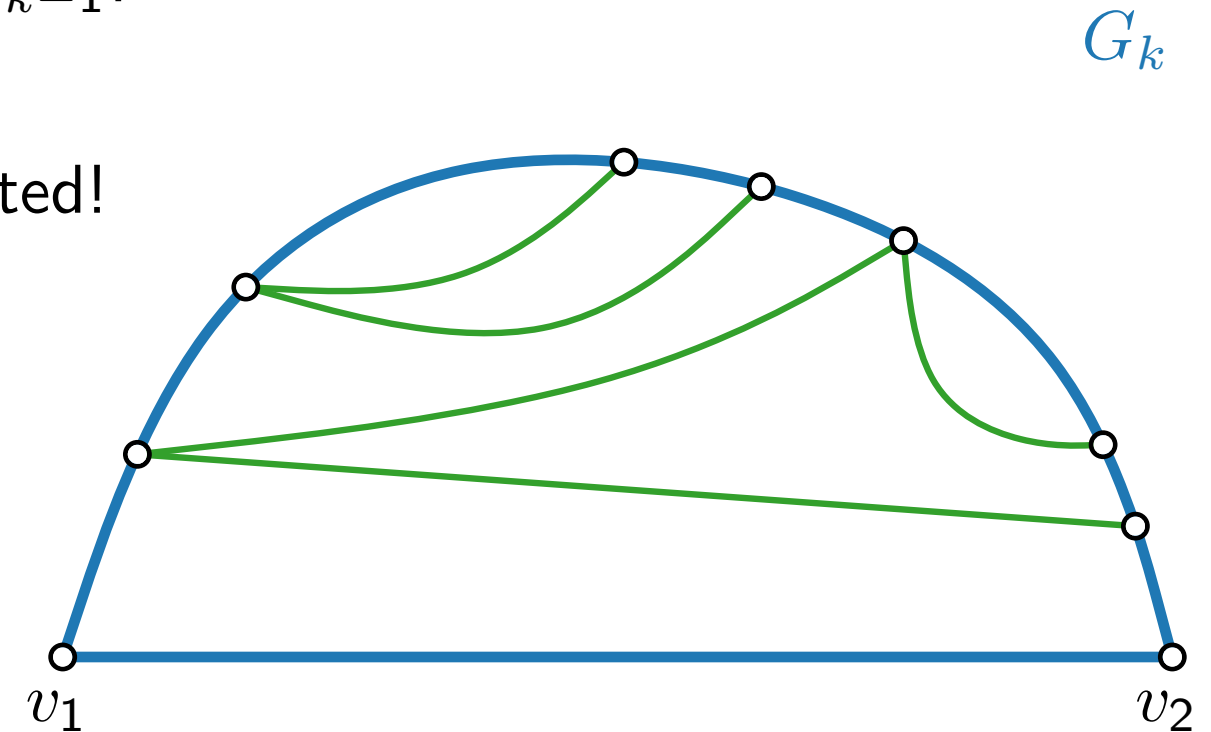
## Claim 1.

If  $v_k$  is not incident to a **chord**,  
then  $G_{k-1}$  is biconnected.



## Claim 2.

There exists a vertex in  $G_k$  that is not incident to a chord as choice for  $v_k$ .

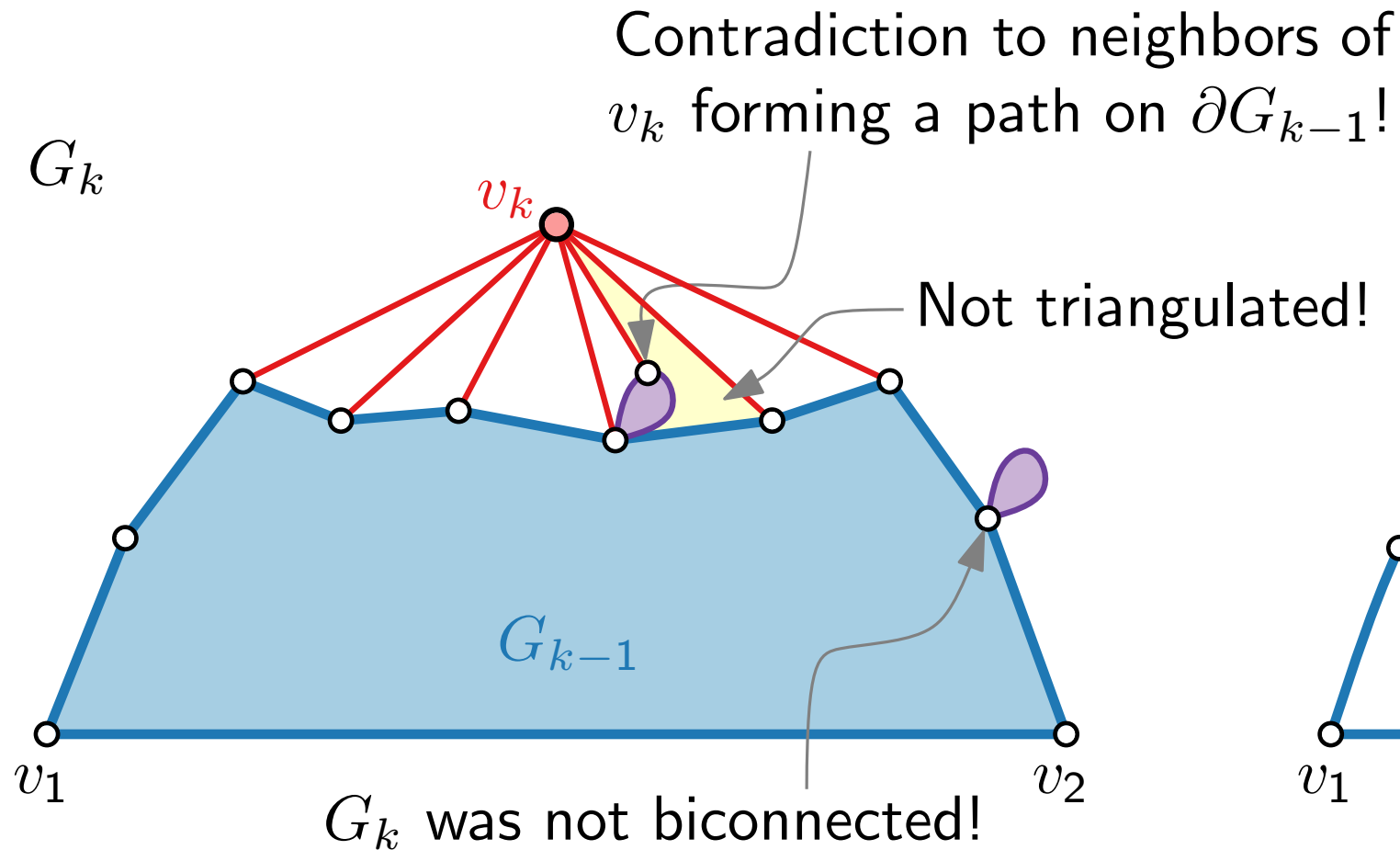




# Canonical Order – Existence

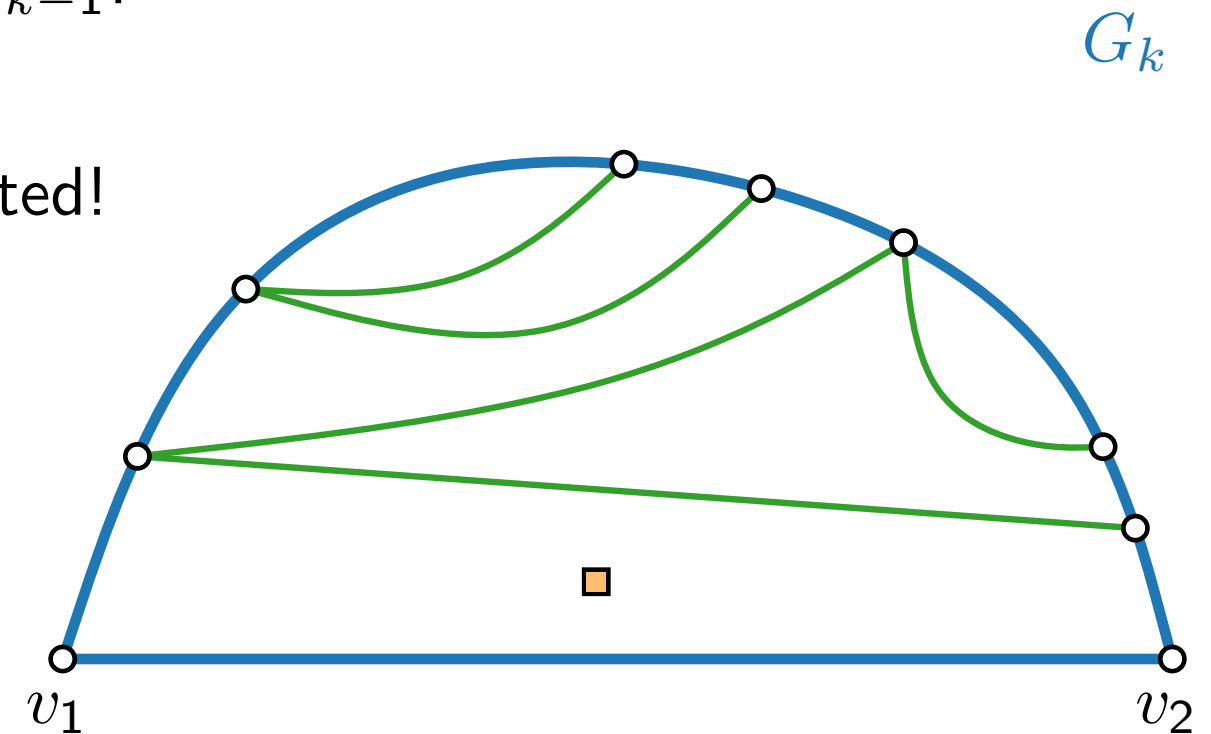
## Claim 1.

If  $v_k$  is not incident to a **chord**, then  $G_{k-1}$  is biconnected.



## Claim 2.

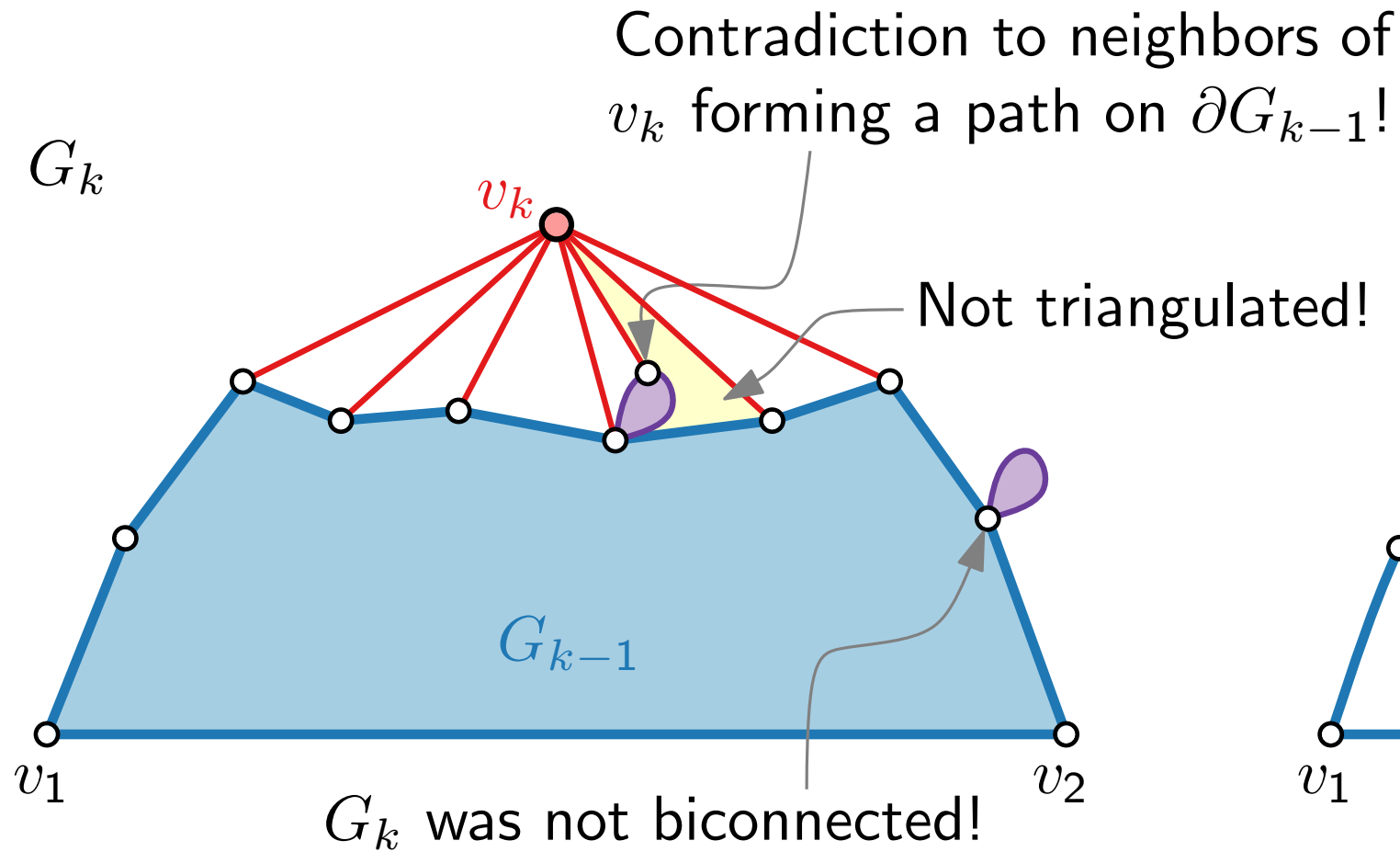
There exists a vertex in  $G_k$  that is not incident to a chord as choice for  $v_k$ .



# Canonical Order – Existence

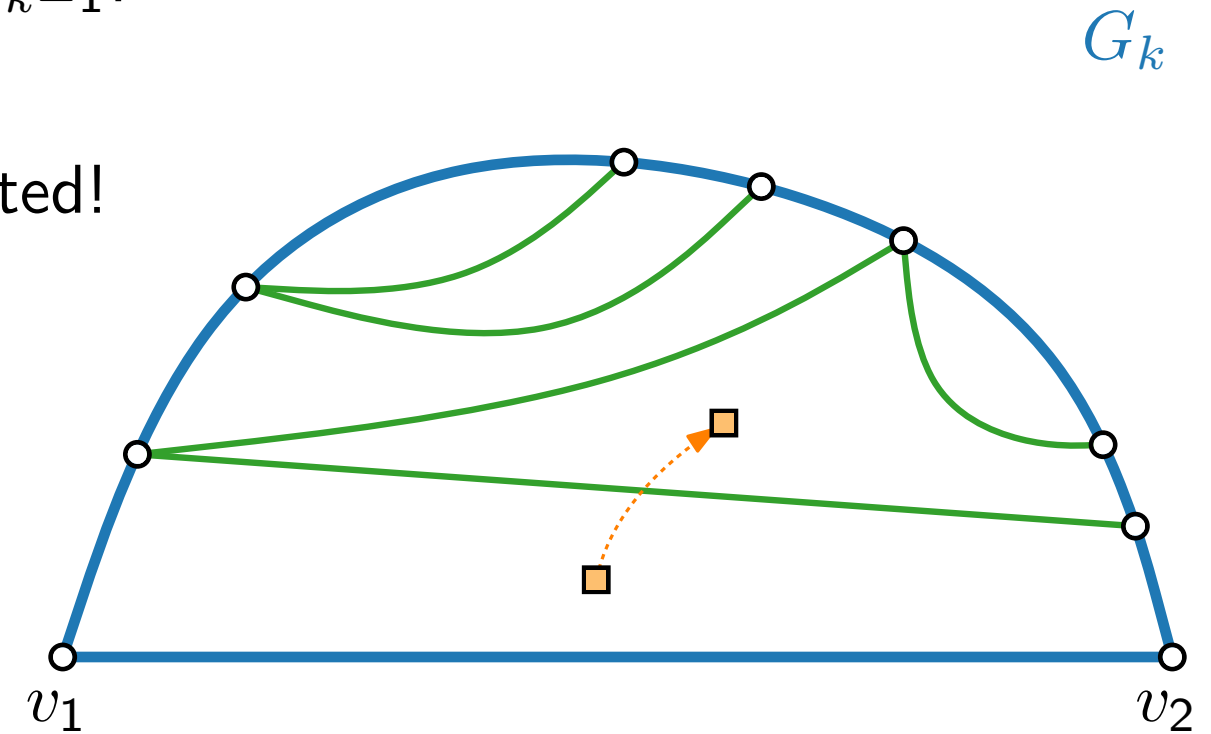
## Claim 1.

If  $v_k$  is not incident to a **chord**,  
then  $G_{k-1}$  is biconnected.



## Claim 2.

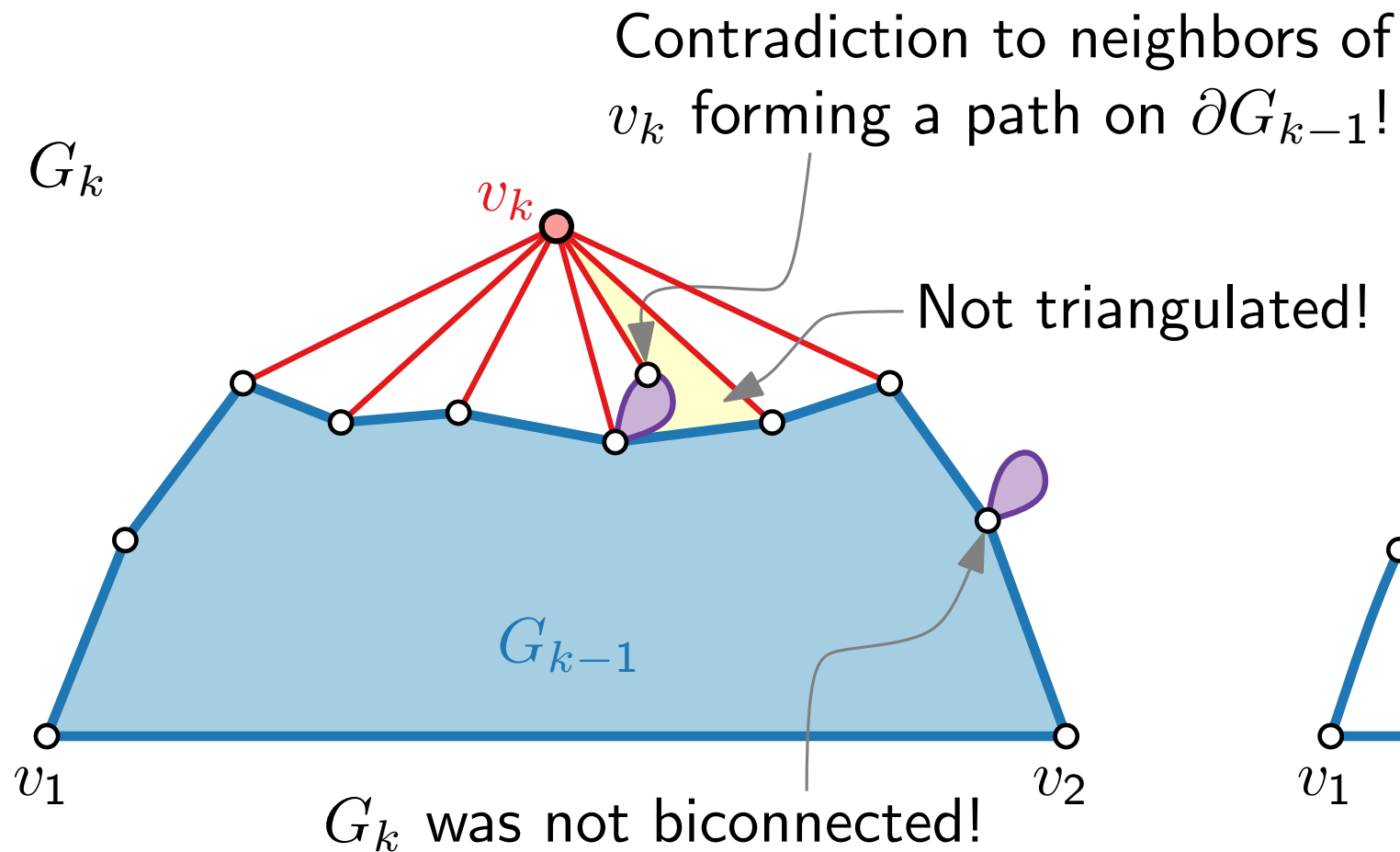
There exists a vertex in  $G_k$  that is not  
incident to a chord as choice for  $v_k$ .



# Canonical Order – Existence

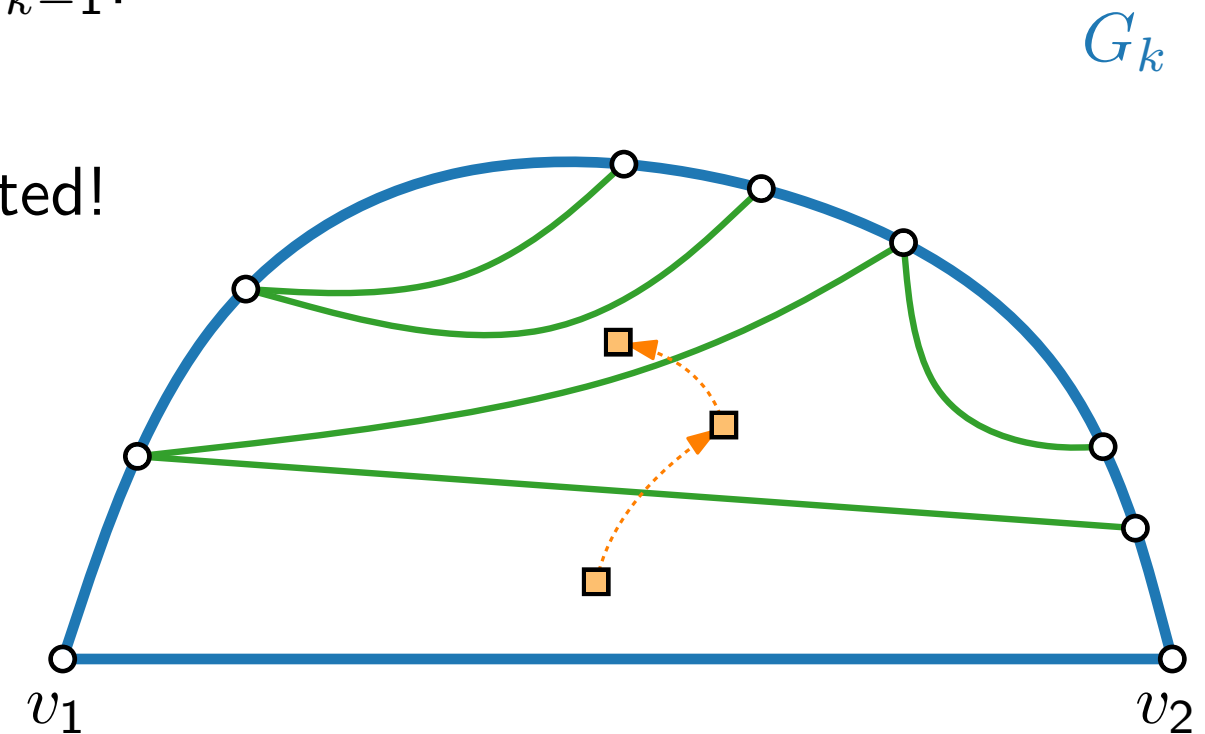
## Claim 1.

If  $v_k$  is not incident to a **chord**,  
then  $G_{k-1}$  is biconnected.



## Claim 2.

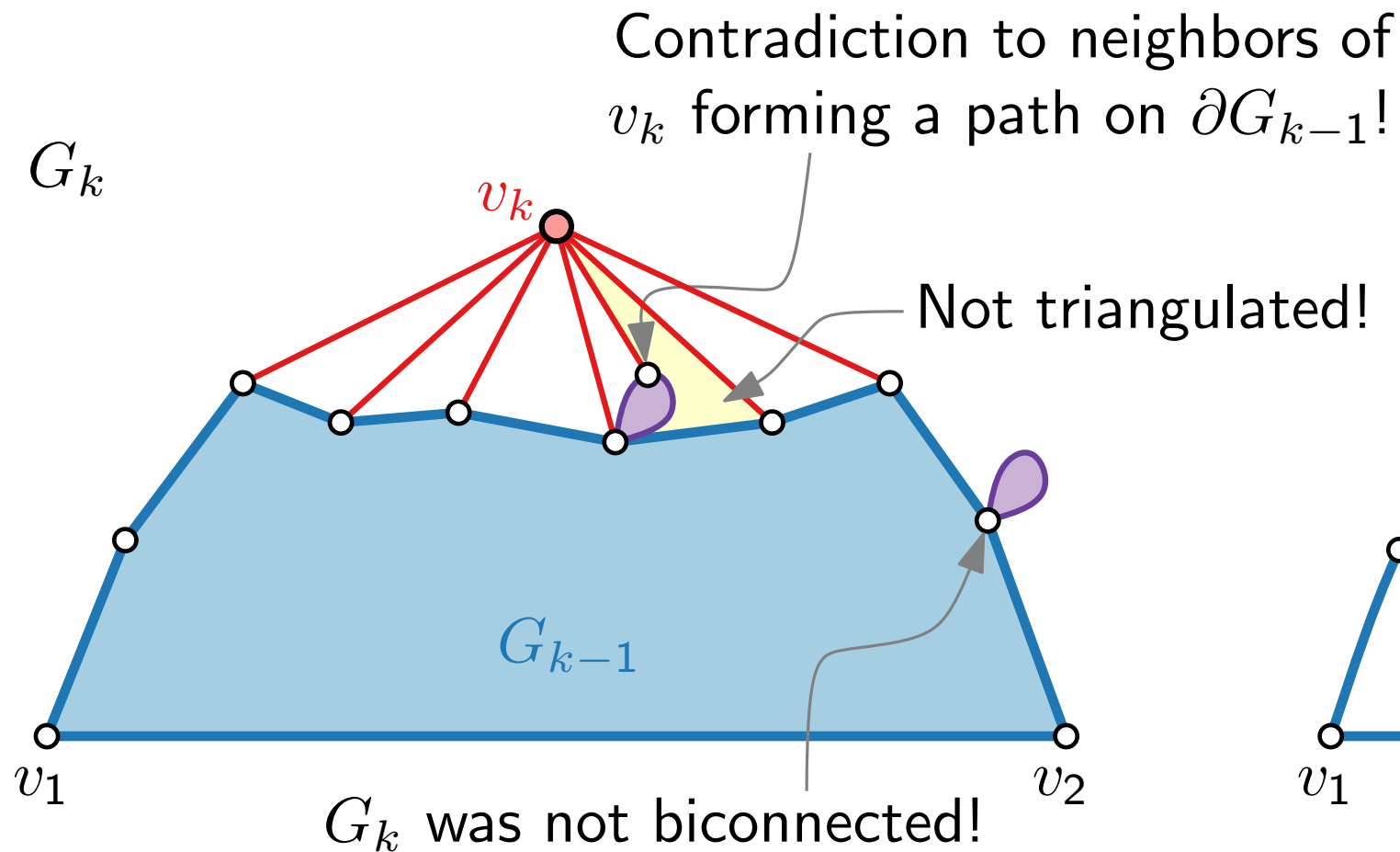
There exists a vertex in  $G_k$  that is not incident to a chord as choice for  $v_k$ .



# Canonical Order – Existence

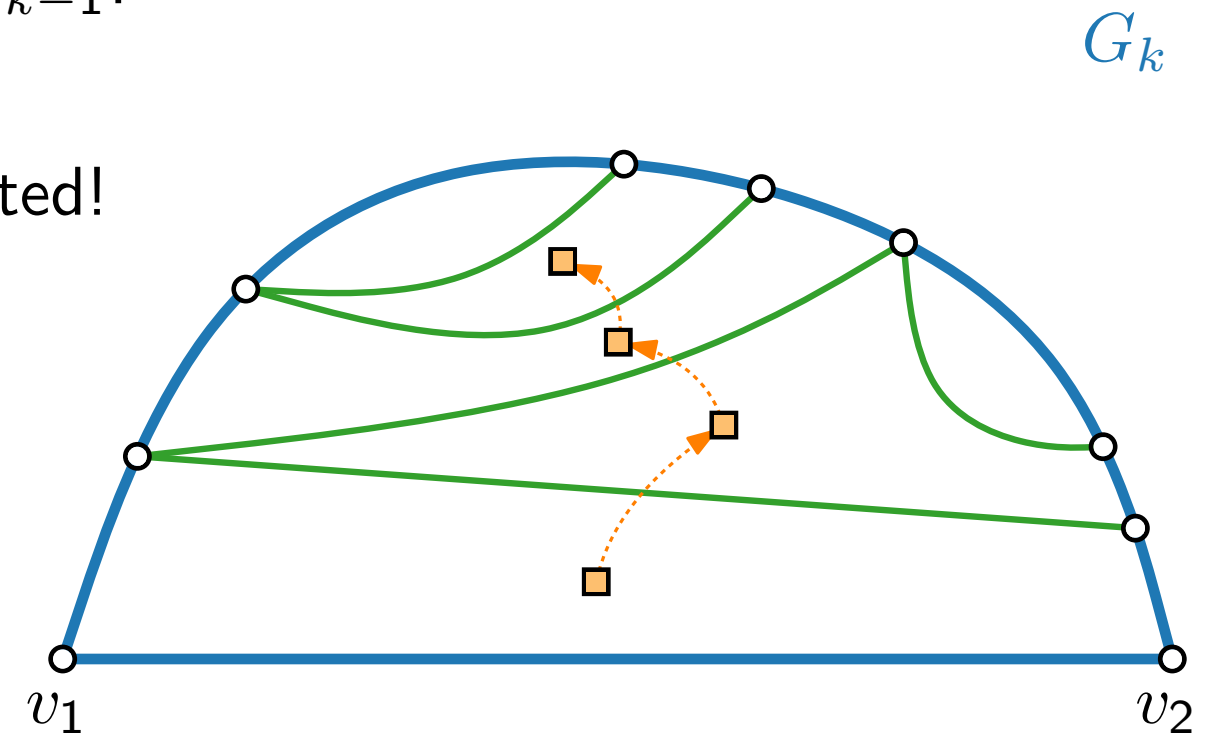
## Claim 1.

If  $v_k$  is not incident to a **chord**,  
then  $G_{k-1}$  is biconnected.



## Claim 2.

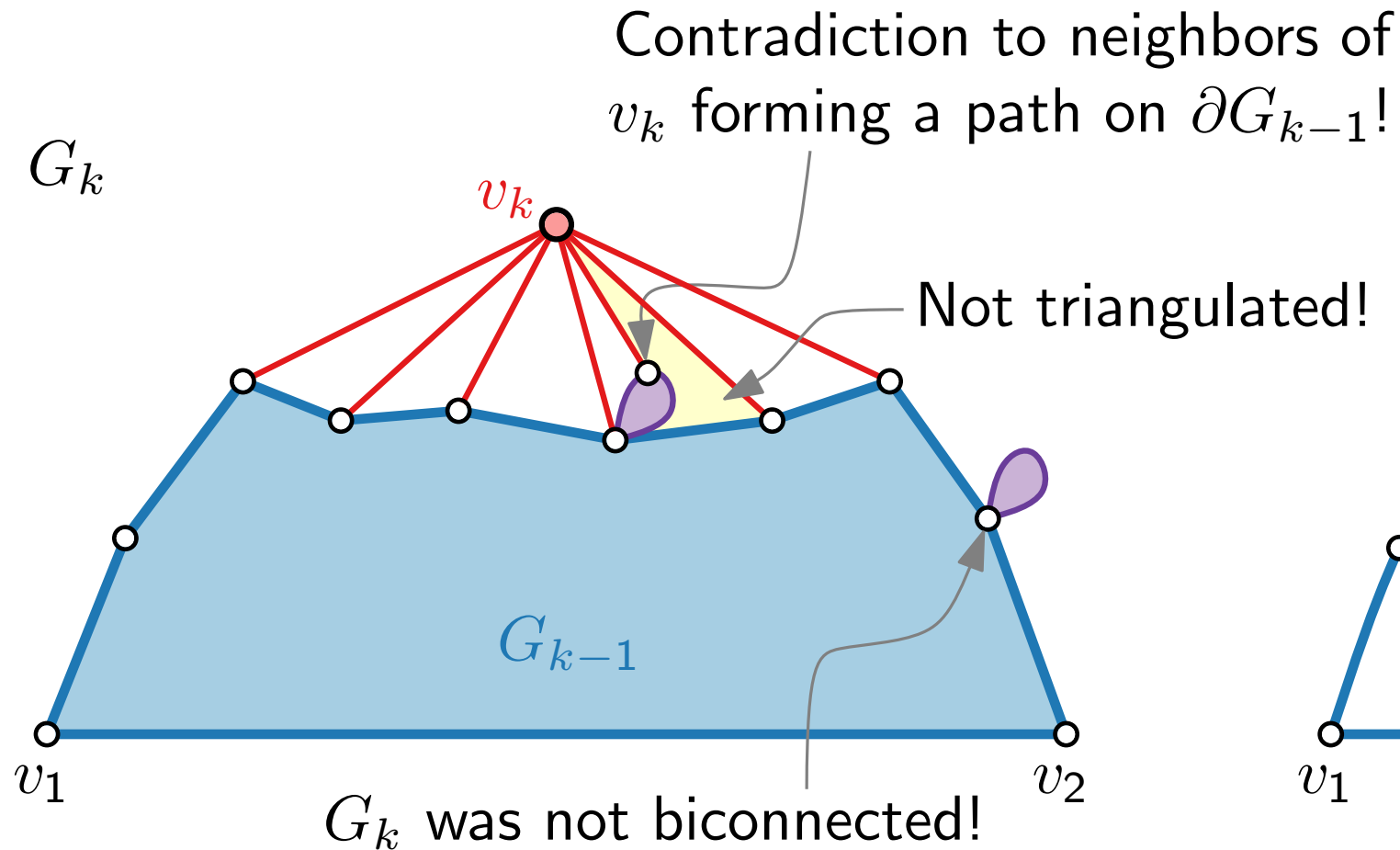
There exists a vertex in  $G_k$  that is not incident to a chord as choice for  $v_k$ .



# Canonical Order – Existence

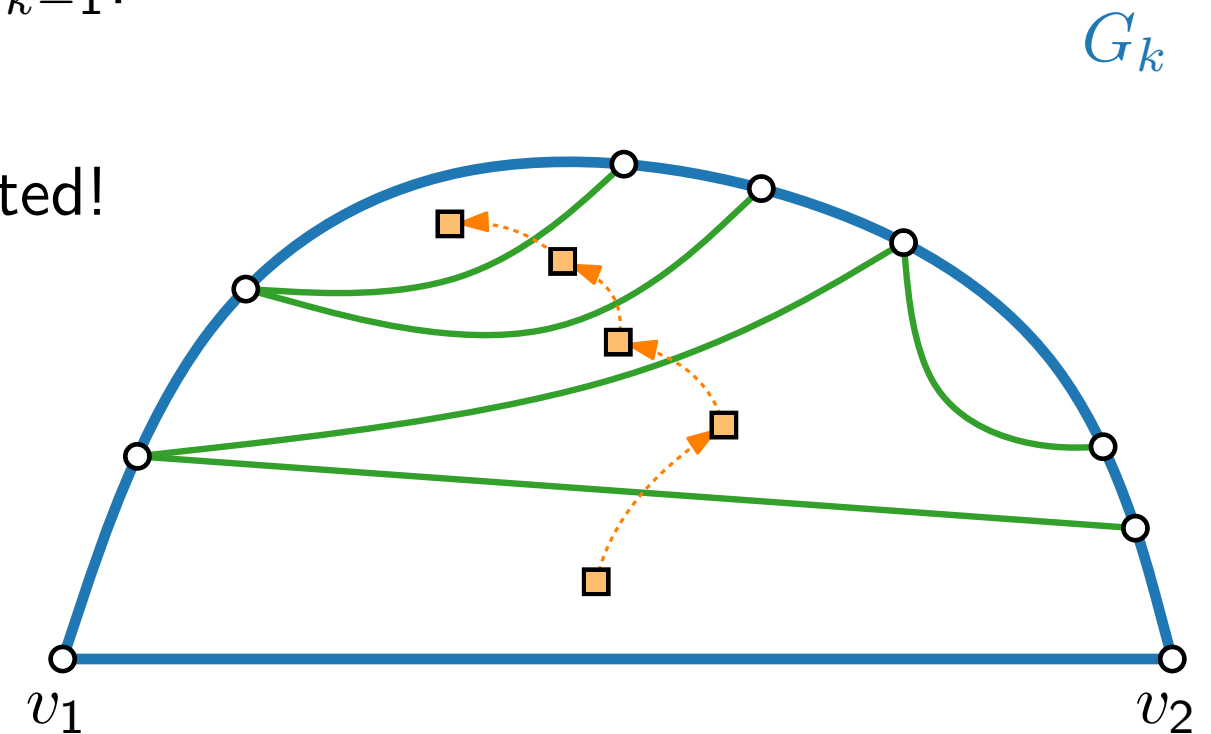
## Claim 1.

If  $v_k$  is not incident to a **chord**,  
then  $G_{k-1}$  is biconnected.



## Claim 2.

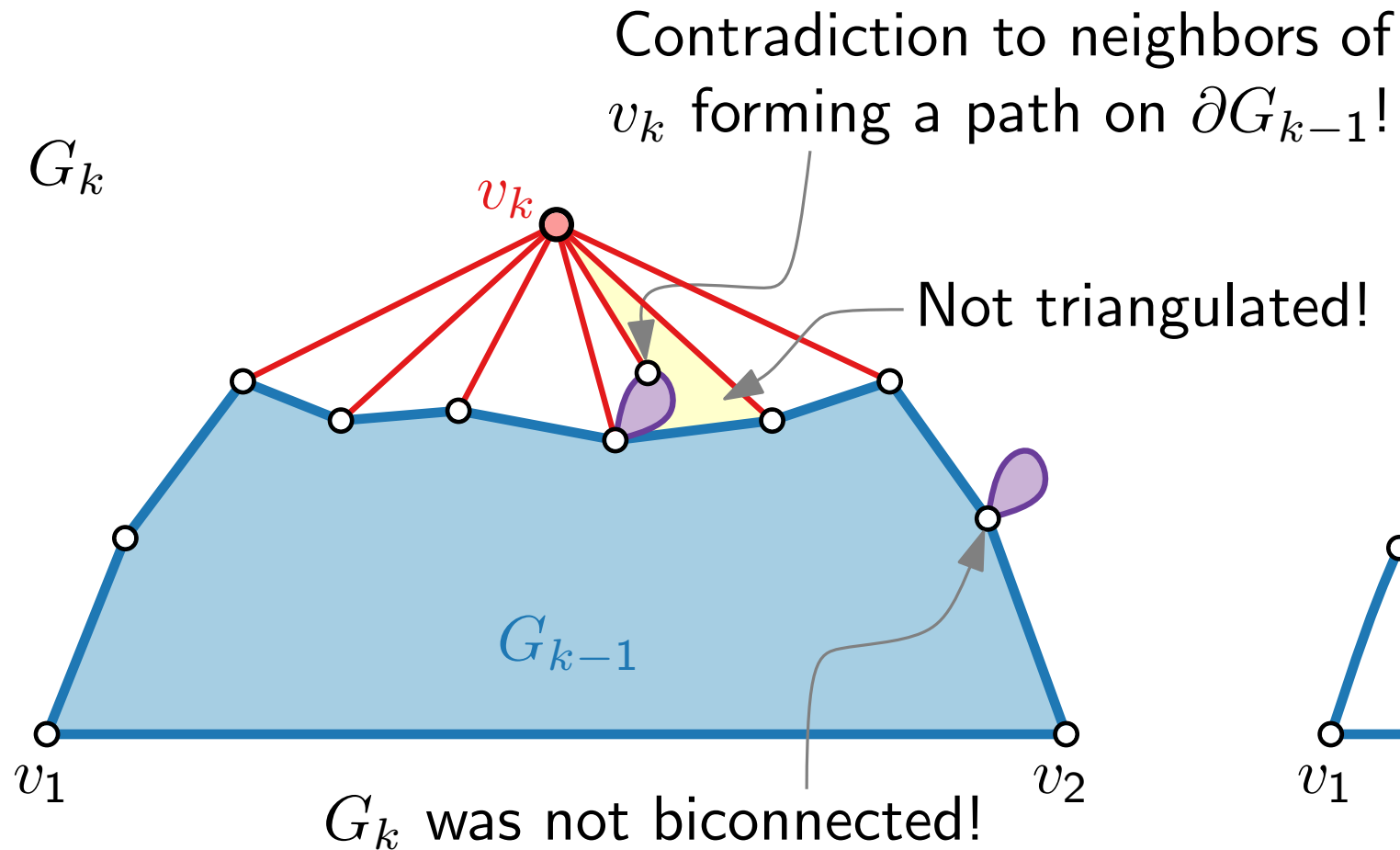
There exists a vertex in  $G_k$  that is not incident to a chord as choice for  $v_k$ .



# Canonical Order – Existence

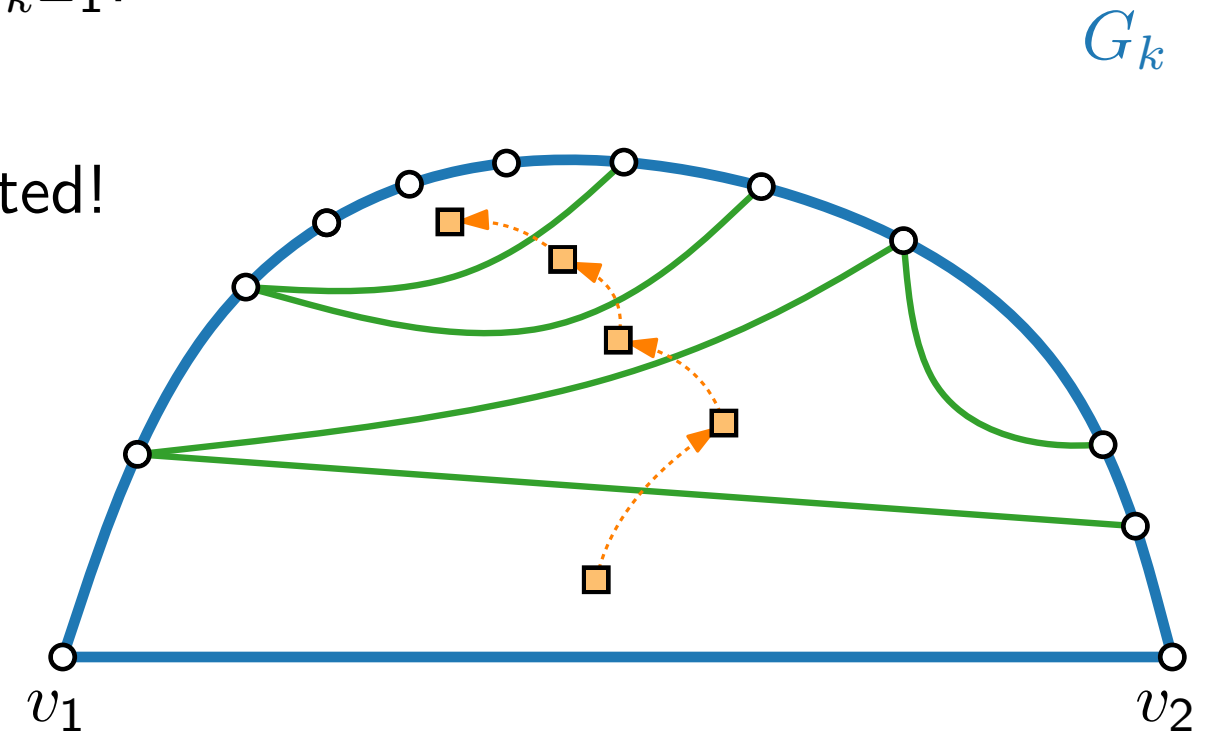
## Claim 1.

If  $v_k$  is not incident to a **chord**,  
then  $G_{k-1}$  is biconnected.



## Claim 2.

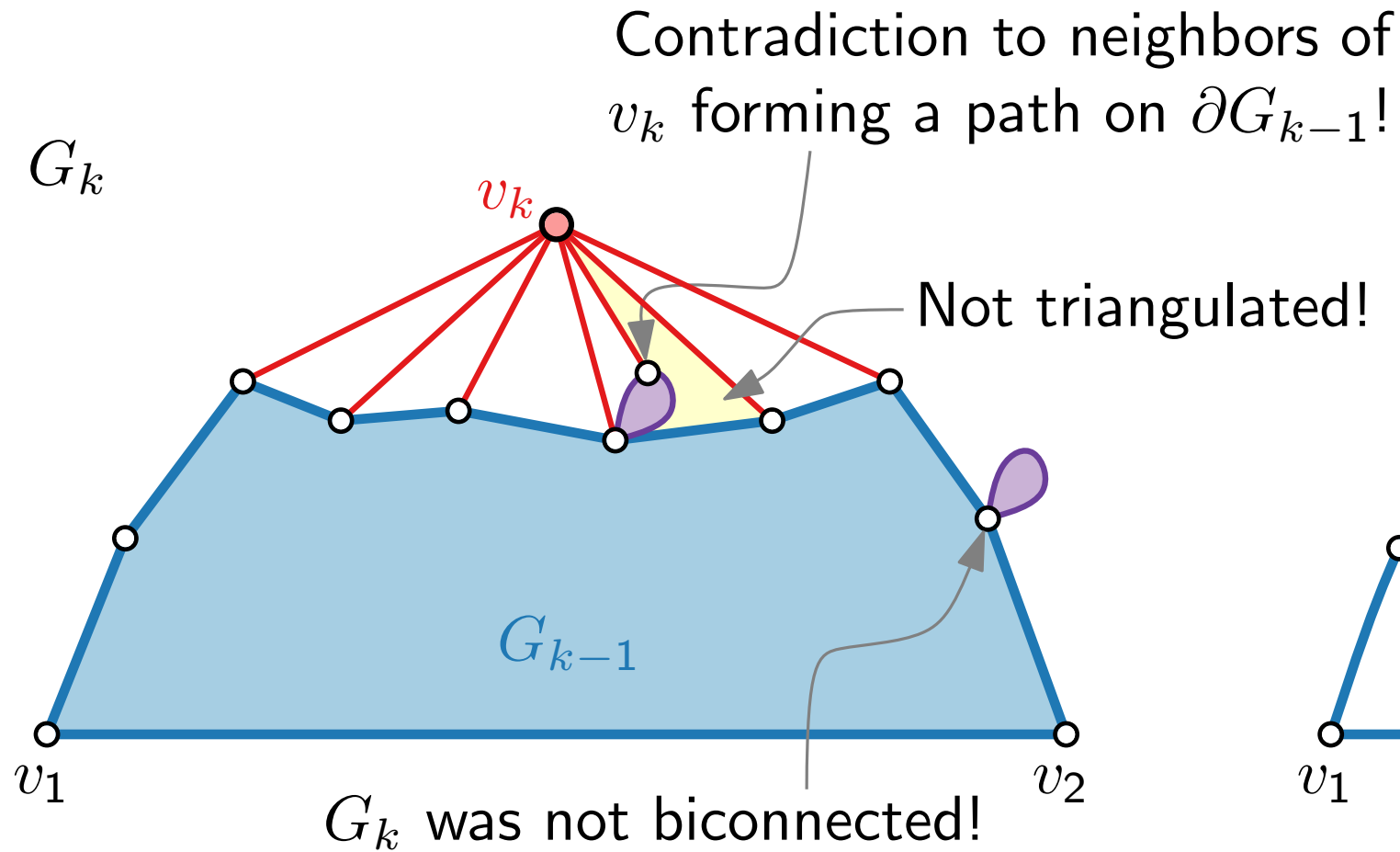
There exists a vertex in  $G_k$  that is not incident to a chord as choice for  $v_k$ .



# Canonical Order – Existence

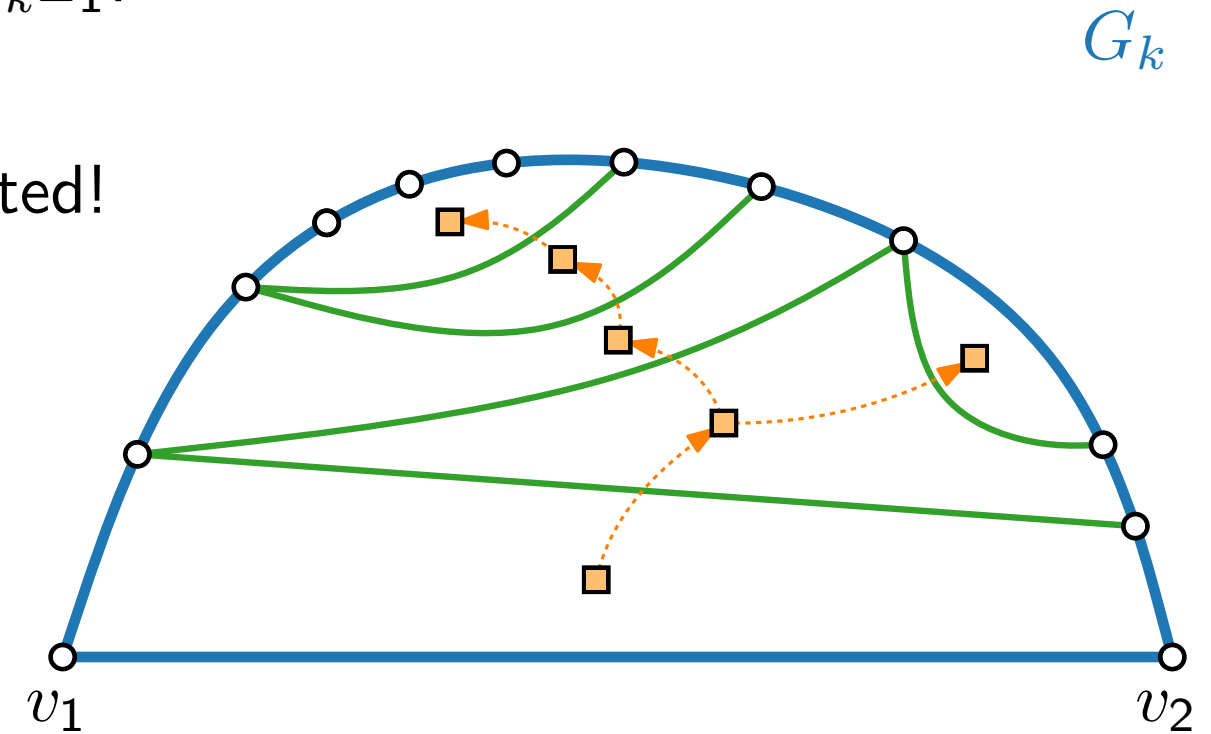
## Claim 1.

If  $v_k$  is not incident to a **chord**,  
then  $G_{k-1}$  is biconnected.



## Claim 2.

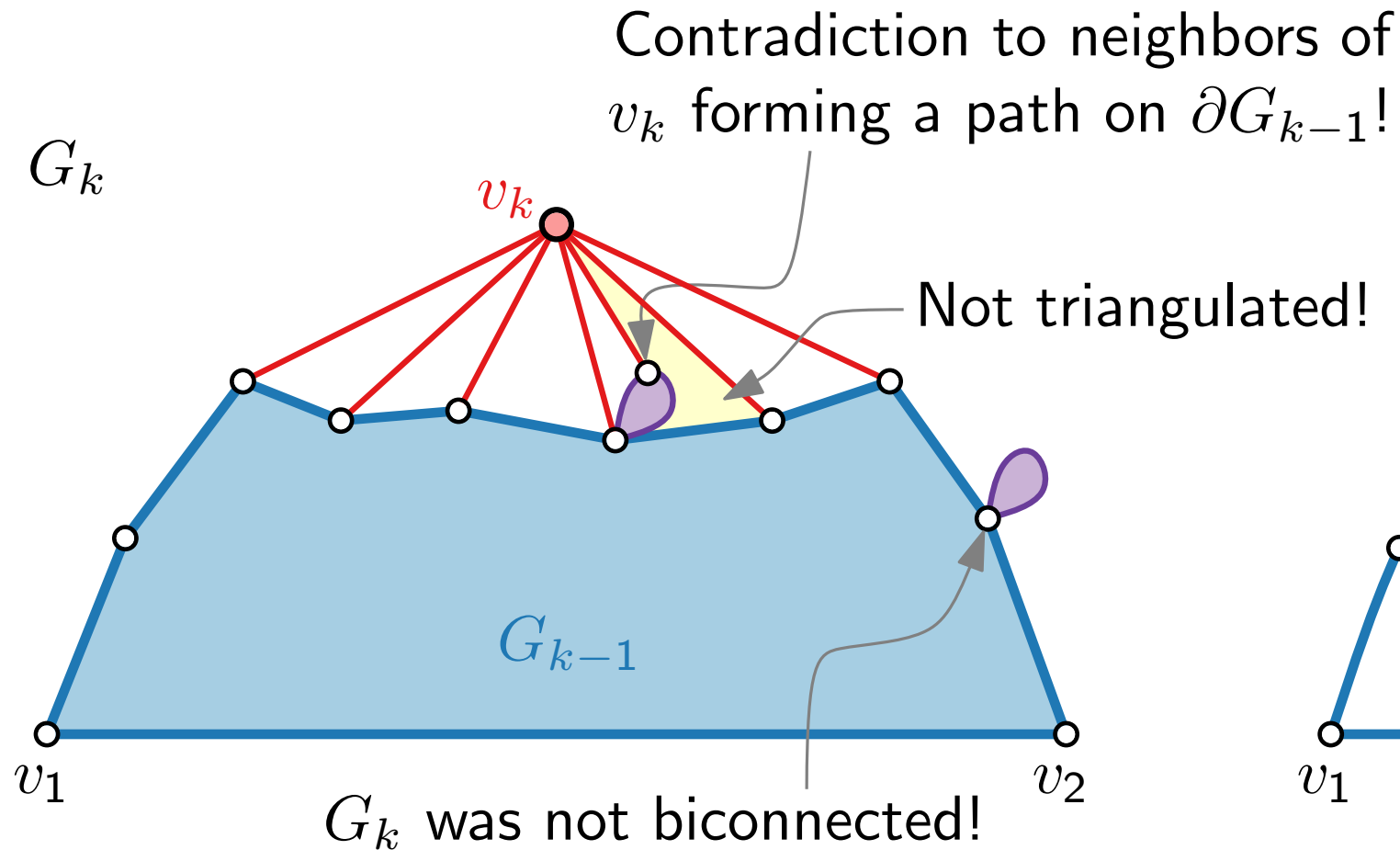
There exists a vertex in  $G_k$  that is not incident to a chord as choice for  $v_k$ .



# Canonical Order – Existence

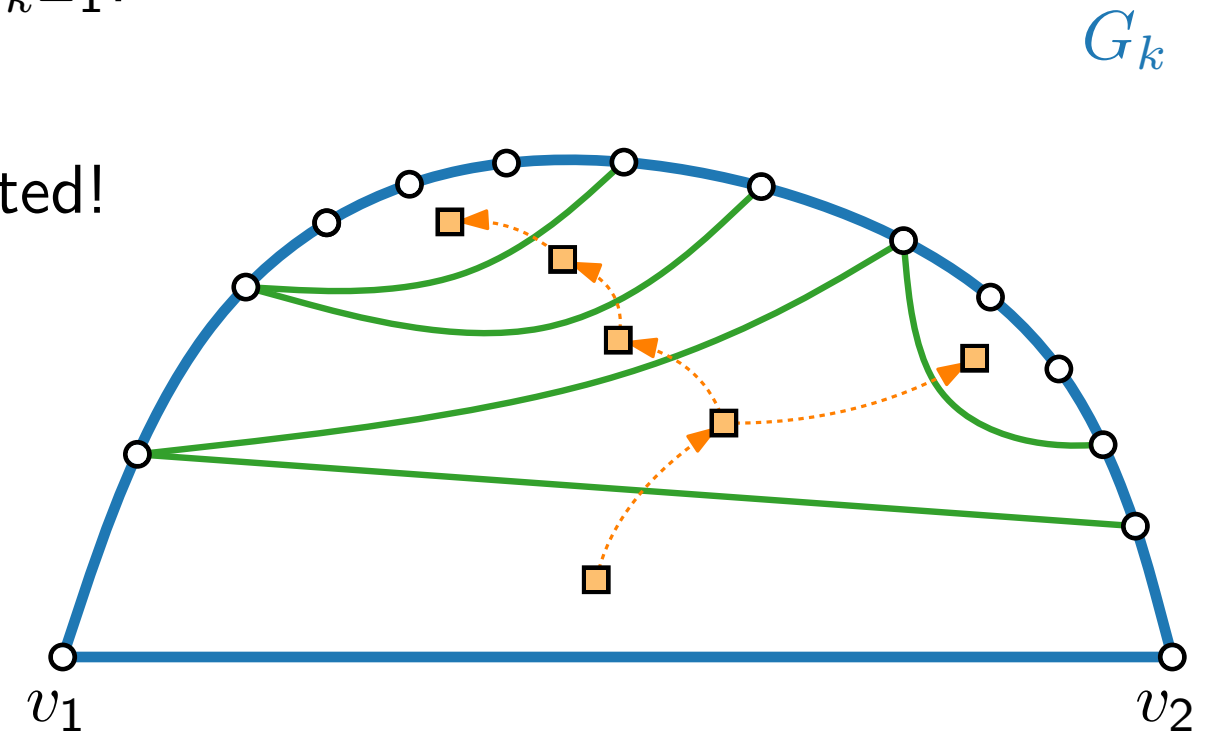
## Claim 1.

If  $v_k$  is not incident to a **chord**, then  $G_{k-1}$  is biconnected.



## Claim 2.

There exists a vertex in  $G_k$  that is not incident to a chord as choice for  $v_k$ .

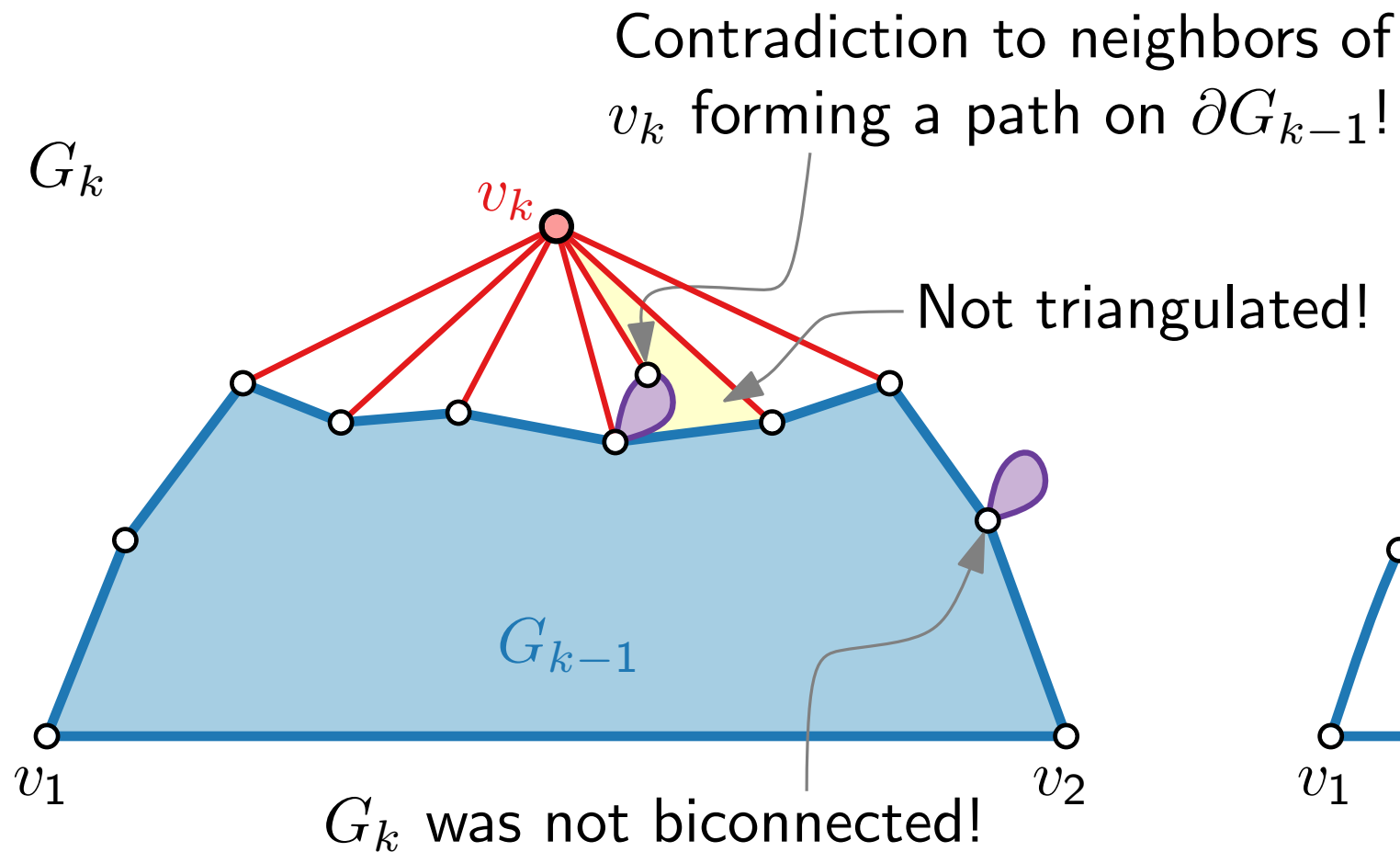




# Canonical Order – Existence

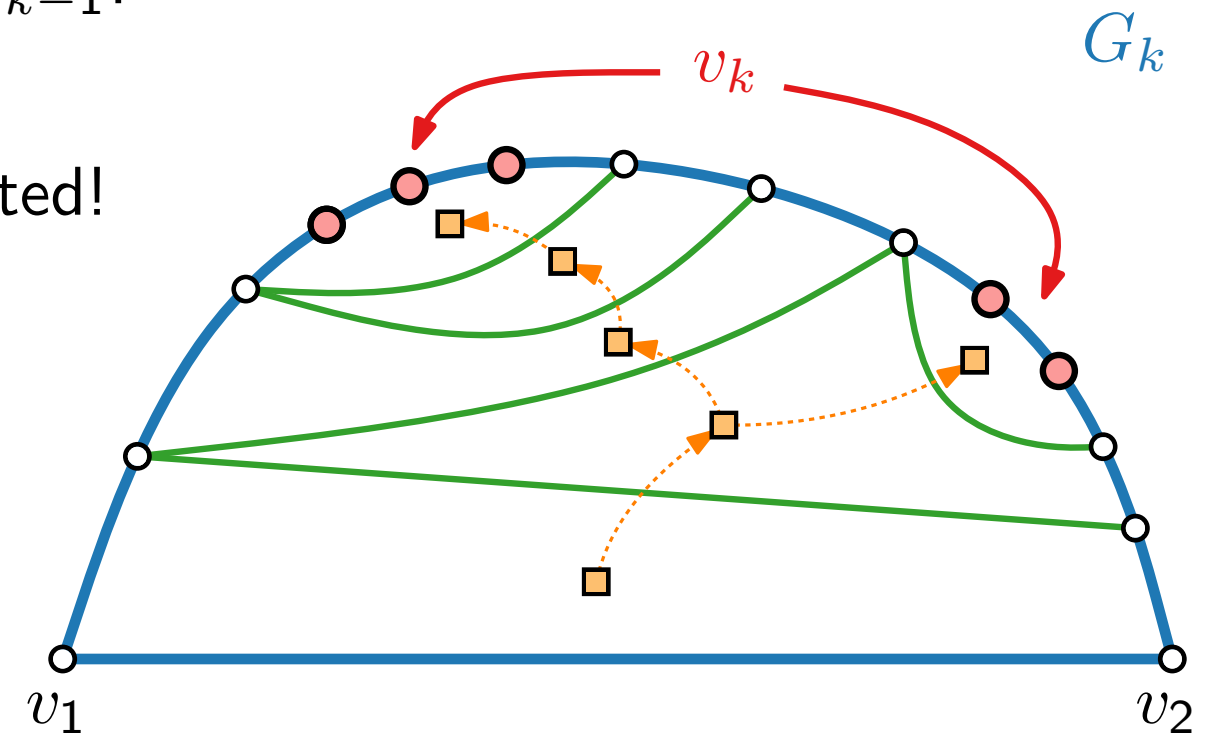
## Claim 1.

If  $v_k$  is not incident to a **chord**, then  $G_{k-1}$  is biconnected.



## Claim 2.

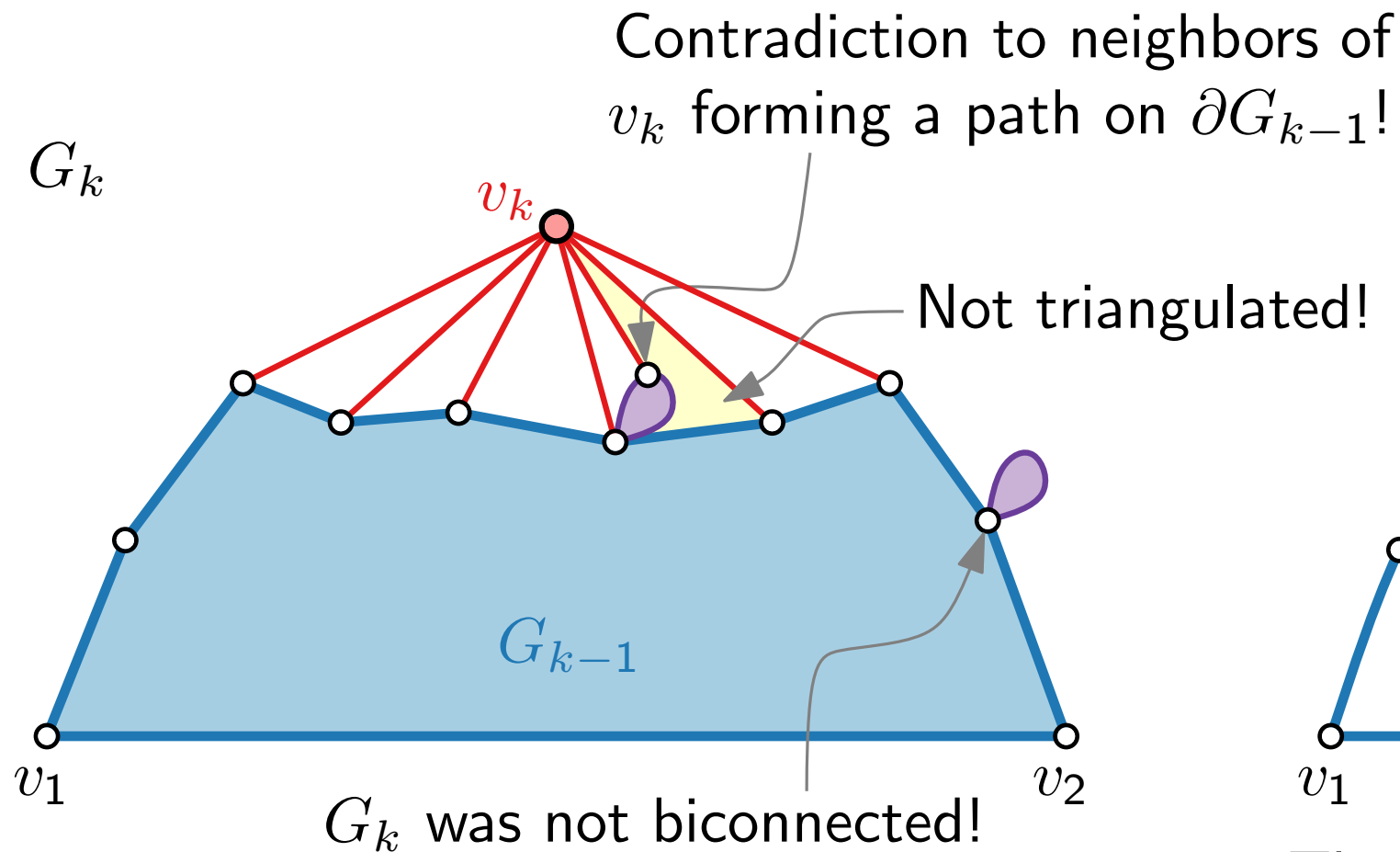
There exists a vertex in  $G_k$  that is not incident to a chord as choice for  $v_k$ .



# Canonical Order – Existence

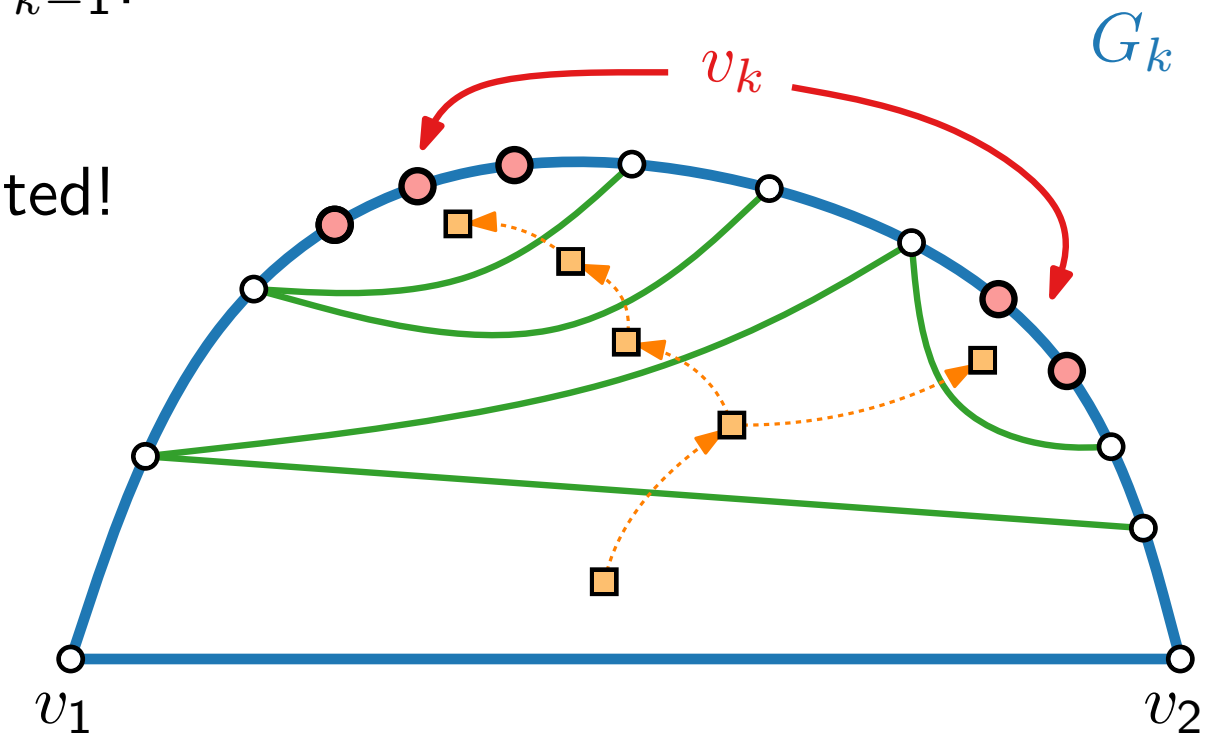
## Claim 1.

If  $v_k$  is not incident to a **chord**, then  $G_{k-1}$  is biconnected.



## Claim 2.

There exists a vertex in  $G_k$  that is not incident to a chord as choice for  $v_k$ .



This completes the proof of the lemma.  $\square$

# Canonical Order – Implementation

CanonicalOrder( $G = (V, E), (v_1, v_2, v_n)$ )

# Canonical Order – Implementation

outer face

$\text{CanonicalOrder}(G = (V, E), (v_1, v_2, v_n))$

# Canonical Order – Implementation

outer face

CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )

**forall**  $v \in V$  **do**

└

└

# Canonical Order – Implementation

outer face

CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )

**forall**  $v \in V$  **do**

└ chords( $v$ )  $\leftarrow 0$ ;

# Canonical Order – Implementation

CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )

**forall**  $v \in V$  **do**  
└  $\text{chords}(v) \leftarrow 0$ ;

outer face

- $\text{chord}(v)$ :  
# chords adjacent to  $v$

# Canonical Order – Implementation

CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )

**forall**  $v \in V$  **do**

└  $\text{chords}(v) \leftarrow 0$ ;  $\text{out}(v) \leftarrow \text{false}$ ;

- $\text{chord}(v)$ :  
# chords adjacent to  $v$



# Canonical Order – Implementation

CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )

**forall**  $v \in V$  **do**

└  $\text{chords}(v) \leftarrow 0$ ;  $\text{out}(v) \leftarrow \text{false}$ ;

outer face

- $\text{chord}(v)$ :  
# chords adjacent to  $v$
- $\text{out}(v) = \text{true}$  iff  $v$  is  
currently outer vertex

# Canonical Order – Implementation

CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )  
 forall  $v \in V$  do  
   └  $\text{chords}(v) \leftarrow 0$ ;  $\text{out}(v) \leftarrow \text{false}$ ;  $\text{mark}(v) \leftarrow \text{false}$

- $\text{chord}(v)$ :  
   # chords adjacent to  $v$
- $\text{out}(v) = \text{true}$  iff  $v$  is  
   currently outer vertex

# Canonical Order – Implementation

CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )  
 forall  $v \in V$  do  
    $\text{chords}(v) \leftarrow 0$ ;  $\text{out}(v) \leftarrow \text{false}$ ;  $\text{mark}(v) \leftarrow \text{false}$

- $\text{chord}(v)$ :  
# chords adjacent to  $v$
- $\text{out}(v) = \text{true}$  iff  $v$  is currently outer vertex
- $\text{mark}(v) = \text{true}$  iff  $v$  has received its number

# Canonical Order – Implementation

outer face

```
CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )
forall  $v \in V$  do
   $\lfloor$  chords( $v$ )  $\leftarrow$  0; out( $v$ )  $\leftarrow$  false; mark( $v$ )  $\leftarrow$  false
  mark( $v_1$ ), mark( $v_2$ ), out( $v_1$ ), out( $v_2$ ), out( $v_n$ )  $\leftarrow$  true
```

- chord( $v$ ):  
# chords adjacent to  $v$
- out( $v$ ) = true iff  $v$  is currently outer vertex
- mark( $v$ ) = true iff  $v$  has received its number

# Canonical Order – Implementation

outer face

```
CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )
forall  $v \in V$  do
   $\text{chords}(v) \leftarrow 0$ ;  $\text{out}(v) \leftarrow \text{false}$ ;  $\text{mark}(v) \leftarrow \text{false}$ 
 $\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow \text{true}$ 
for  $k = n$  downto 3 do
```

- $\text{chord}(v)$ :  
# chords adjacent to  $v$
- $\text{out}(v) = \text{true}$  iff  $v$  is currently outer vertex
- $\text{mark}(v) = \text{true}$  iff  $v$  has received its number

# Canonical Order – Implementation

outer face

```
CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )
forall  $v \in V$  do
   $\text{chords}(v) \leftarrow 0$ ;  $\text{out}(v) \leftarrow \text{false}$ ;  $\text{mark}(v) \leftarrow \text{false}$ 
 $\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow \text{true}$ 
for  $k = n$  downto 3 do
  choose  $v$  such that  $\text{mark}(v) = \text{false}$ ,  $\text{out}(v) = \text{true}$ ,
    and  $\text{chords}(v) = 0$ 
```

- $\text{chord}(v)$ :  
# chords adjacent to  $v$
- $\text{out}(v) = \text{true}$  iff  $v$  is currently outer vertex
- $\text{mark}(v) = \text{true}$  iff  $v$  has received its number

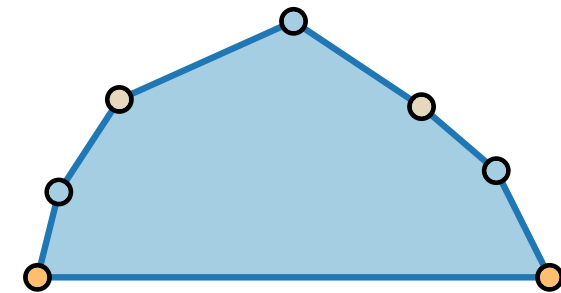
# Canonical Order – Implementation

outer face

```

CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )
forall  $v \in V$  do
   $\text{chords}(v) \leftarrow 0$ ;  $\text{out}(v) \leftarrow \text{false}$ ;  $\text{mark}(v) \leftarrow \text{false}$ 
 $\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow \text{true}$ 
for  $k = n$  downto 3 do
  choose  $v$  such that  $\text{mark}(v) = \text{false}$ ,  $\text{out}(v) = \text{true}$ ,
    and  $\text{chords}(v) = 0$ 
  
```

- $\text{chord}(v)$ :  
# chords adjacent to  $v$
- $\text{out}(v) = \text{true}$  iff  $v$  is currently outer vertex
- $\text{mark}(v) = \text{true}$  iff  $v$  has received its number

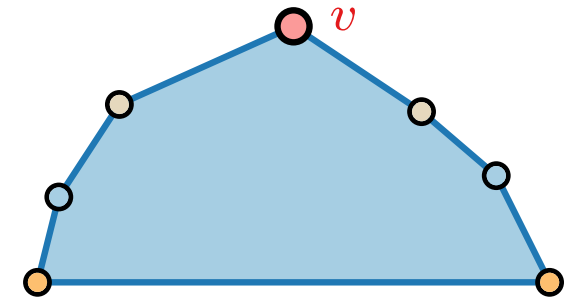


# Canonical Order – Implementation

outer face

```
CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )
forall  $v \in V$  do
   $\text{chords}(v) \leftarrow 0$ ;  $\text{out}(v) \leftarrow \text{false}$ ;  $\text{mark}(v) \leftarrow \text{false}$ 
 $\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow \text{true}$ 
for  $k = n$  downto 3 do
  choose  $v$  such that  $\text{mark}(v) = \text{false}$ ,  $\text{out}(v) = \text{true}$ ,
  and  $\text{chords}(v) = 0$ 
```

- $\text{chord}(v)$ :  
# chords adjacent to  $v$
- $\text{out}(v) = \text{true}$  iff  $v$  is currently outer vertex
- $\text{mark}(v) = \text{true}$  iff  $v$  has received its number





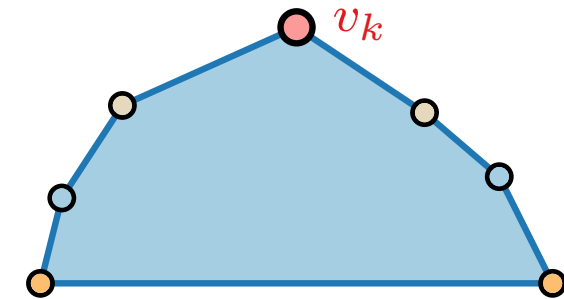
# Canonical Order – Implementation

outer face

```

CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )
forall  $v \in V$  do
     $\text{chords}(v) \leftarrow 0$ ;  $\text{out}(v) \leftarrow \text{false}$ ;  $\text{mark}(v) \leftarrow \text{false}$ 
 $\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow \text{true}$ 
for  $k = n$  downto 3 do
    choose  $v$  such that  $\text{mark}(v) = \text{false}$ ,  $\text{out}(v) = \text{true}$ ,
        and  $\text{chords}(v) = 0$ 
     $v_k \leftarrow v$ ;  $\text{mark}(v) \leftarrow \text{true}$ 
  
```

- $\text{chord}(v)$ :  
# chords adjacent to  $v$
- $\text{out}(v) = \text{true}$  iff  $v$  is currently outer vertex
- $\text{mark}(v) = \text{true}$  iff  $v$  has received its number



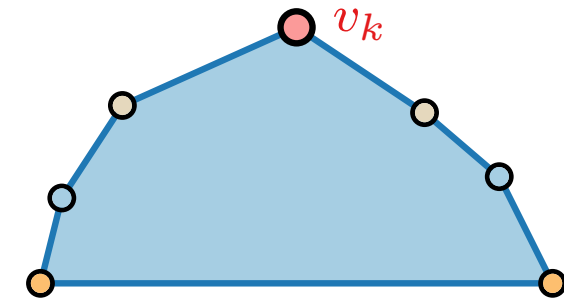
# Canonical Order – Implementation

outer face

```

CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )
forall  $v \in V$  do
   $\text{chords}(v) \leftarrow 0$ ;  $\text{out}(v) \leftarrow \text{false}$ ;  $\text{mark}(v) \leftarrow \text{false}$ 
 $\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow \text{true}$ 
for  $k = n$  downto 3 do
  choose  $v$  such that  $\text{mark}(v) = \text{false}$ ,  $\text{out}(v) = \text{true}$ ,
    and  $\text{chords}(v) = 0$ 
   $v_k \leftarrow v$ ;  $\text{mark}(v) \leftarrow \text{true}$ 
  // Let  $\partial G_{k-1}$  be  $w_1 = v_1, w_2, \dots, w_{t-1}, w_t = v_2$ .
  Let  $w_p, \dots, w_q$  be t
  †
  
```

- $\text{chord}(v)$ :  
# chords adjacent to  $v$
- $\text{out}(v) = \text{true}$  iff  $v$  is currently outer vertex
- $\text{mark}(v) = \text{true}$  iff  $v$  has received its number



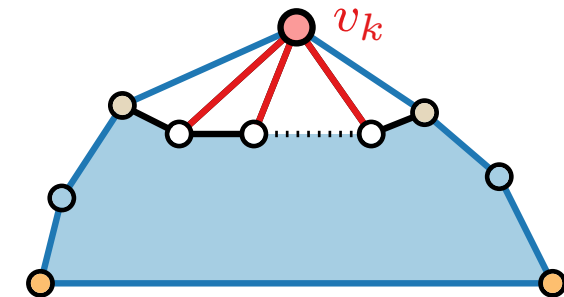
# Canonical Order – Implementation

outer face

```

CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )
forall  $v \in V$  do
   $\text{chords}(v) \leftarrow 0$ ;  $\text{out}(v) \leftarrow \text{false}$ ;  $\text{mark}(v) \leftarrow \text{false}$ 
 $\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow \text{true}$ 
for  $k = n$  downto 3 do
  choose  $v$  such that  $\text{mark}(v) = \text{false}$ ,  $\text{out}(v) = \text{true}$ ,
    and  $\text{chords}(v) = 0$ 
   $v_k \leftarrow v$ ;  $\text{mark}(v) \leftarrow \text{true}$ 
  // Let  $\partial G_{k-1}$  be  $w_1 = v_1, w_2, \dots, w_{t-1}, w_t = v_2$ .
  Let  $w_p, \dots, w_q$  be t
  †
  
```

- $\text{chord}(v)$ :  
# chords adjacent to  $v$
- $\text{out}(v) = \text{true}$  iff  $v$  is currently outer vertex
- $\text{mark}(v) = \text{true}$  iff  $v$  has received its number



# Canonical Order – Implementation

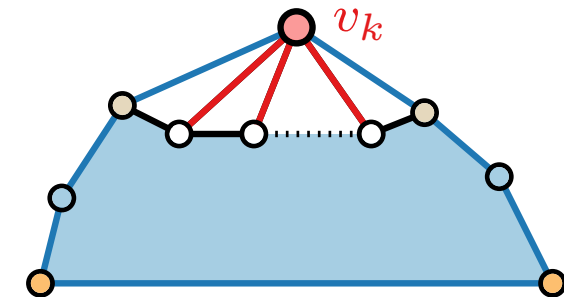
outer face

```

CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )
forall  $v \in V$  do
   $\text{chords}(v) \leftarrow 0$ ;  $\text{out}(v) \leftarrow \text{false}$ ;  $\text{mark}(v) \leftarrow \text{false}$ 
 $\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow \text{true}$ 
for  $k = n$  downto 3 do
  choose  $v$  such that  $\text{mark}(v) = \text{false}$ ,  $\text{out}(v) = \text{true}$ ,
    and  $\text{chords}(v) = 0$ 
   $v_k \leftarrow v$ ;  $\text{mark}(v) \leftarrow \text{true}$ 
  // Let  $\partial G_{k-1}$  be  $w_1 = v_1, w_2, \dots, w_{t-1}, w_t = v_2$ .
  Let  $w_p, \dots, w_q$  be the unmarked neighbors of  $v_k$ .
  for  $i = p$  to  $q$  do

```

- $\text{chord}(v)$ :  
# chords adjacent to  $v$
- $\text{out}(v) = \text{true}$  iff  $v$  is currently outer vertex
- $\text{mark}(v) = \text{true}$  iff  $v$  has received its number



# Canonical Order – Implementation

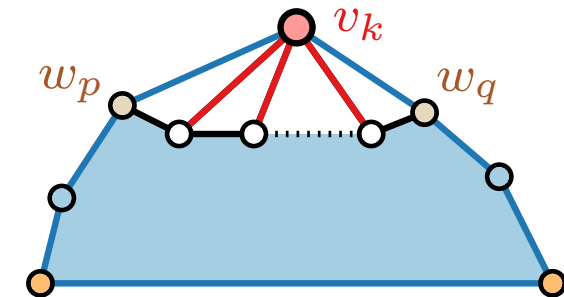
outer face

```

CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )
forall  $v \in V$  do
   $\text{chords}(v) \leftarrow 0$ ;  $\text{out}(v) \leftarrow \text{false}$ ;  $\text{mark}(v) \leftarrow \text{false}$ 
 $\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow \text{true}$ 
for  $k = n$  downto 3 do
  choose  $v$  such that  $\text{mark}(v) = \text{false}$ ,  $\text{out}(v) = \text{true}$ ,
    and  $\text{chords}(v) = 0$ 
   $v_k \leftarrow v$ ;  $\text{mark}(v) \leftarrow \text{true}$ 
  // Let  $\partial G_{k-1}$  be  $w_1 = v_1, w_2, \dots, w_{t-1}, w_t = v_2$ .
  Let  $w_p, \dots, w_q$  be the unmarked neighbors of  $v_k$ .
  for  $i = p$  to  $q$  do

```

- $\text{chord}(v)$ :  
# chords adjacent to  $v$
- $\text{out}(v) = \text{true}$  iff  $v$  is currently outer vertex
- $\text{mark}(v) = \text{true}$  iff  $v$  has received its number



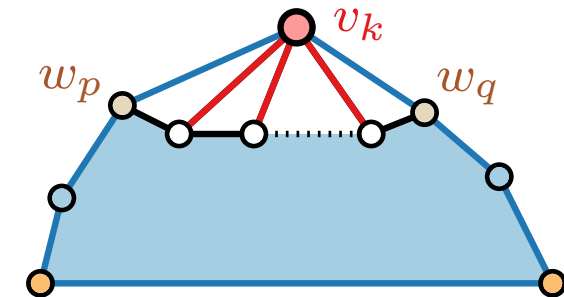
# Canonical Order – Implementation

outer face

```

CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )
forall  $v \in V$  do
  |  $\text{chords}(v) \leftarrow 0$ ;  $\text{out}(v) \leftarrow \text{false}$ ;  $\text{mark}(v) \leftarrow \text{false}$ 
 $\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow \text{true}$ 
for  $k = n$  downto 3 do
  choose  $v$  such that  $\text{mark}(v) = \text{false}$ ,  $\text{out}(v) = \text{true}$ ,
    and  $\text{chords}(v) = 0$ 
   $v_k \leftarrow v$ ;  $\text{mark}(v) \leftarrow \text{true}$ 
  // Let  $\partial G_{k-1}$  be  $w_1 = v_1, w_2, \dots, w_{t-1}, w_t = v_2$ .
  Let  $w_p, \dots, w_q$  be the unmarked neighbors of  $v_k$ .
  for  $i = p$  to  $q$  do
    |  $\text{out}(w_i) \leftarrow \text{true}$ 
  
```

- $\text{chord}(v)$ :  
# chords adjacent to  $v$
- $\text{out}(v) = \text{true}$  iff  $v$  is currently outer vertex
- $\text{mark}(v) = \text{true}$  iff  $v$  has received its number



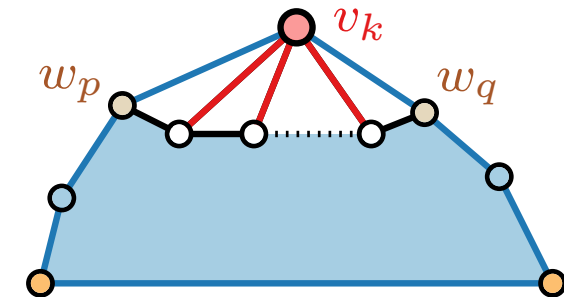
# Canonical Order – Implementation

outer face

```

CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )
forall  $v \in V$  do
   $\text{chords}(v) \leftarrow 0$ ;  $\text{out}(v) \leftarrow \text{false}$ ;  $\text{mark}(v) \leftarrow \text{false}$ 
 $\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow \text{true}$ 
for  $k = n$  downto 3 do
  choose  $v$  such that  $\text{mark}(v) = \text{false}$ ,  $\text{out}(v) = \text{true}$ ,
    and  $\text{chords}(v) = 0$ 
   $v_k \leftarrow v$ ;  $\text{mark}(v) \leftarrow \text{true}$ 
  // Let  $\partial G_{k-1}$  be  $w_1 = v_1, w_2, \dots, w_{t-1}, w_t = v_2$ .
  Let  $w_p, \dots, w_q$  be the unmarked neighbors of  $v_k$ .
  for  $i = p$  to  $q$  do
     $\text{out}(w_i) \leftarrow \text{true}$ 
    update  $\text{chords}(w_i)$ 
    and for its neighbours
  
```

- $\text{chord}(v)$ :  
# chords adjacent to  $v$
- $\text{out}(v) = \text{true}$  iff  $v$  is currently outer vertex
- $\text{mark}(v) = \text{true}$  iff  $v$  has received its number



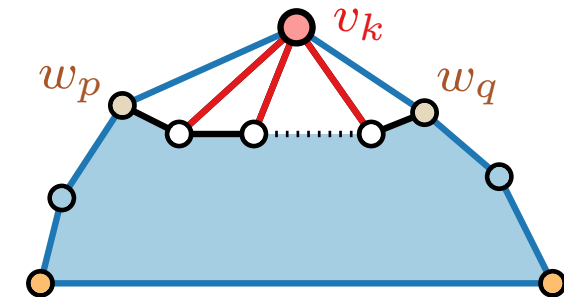
# Canonical Order – Implementation

outer face

```

CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )
  forall  $v \in V$  do
    |  $\text{chords}(v) \leftarrow 0$ ;  $\text{out}(v) \leftarrow \text{false}$ ;  $\text{mark}(v) \leftarrow \text{false}$ 
   $\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow \text{true}$ 
  for  $k = n$  downto 3 do
    choose  $v$  such that  $\text{mark}(v) = \text{false}$ ,  $\text{out}(v) = \text{true}$ ,
      and  $\text{chords}(v) = 0$ 
     $v_k \leftarrow v$ ;  $\text{mark}(v) \leftarrow \text{true}$ 
    // Let  $\partial G_{k-1}$  be  $w_1 = v_1, w_2, \dots, w_{t-1}, w_t = v_2$ .
    Let  $w_p, \dots, w_q$  be the unmarked neighbors of  $v_k$ .
    for  $i = p$  to  $q$  do
      |  $\text{out}(w_i) \leftarrow \text{true}$ 
      | update  $\text{chords}(w_i)$ 
      | and for its neighbours
  
```

- $\text{chord}(v)$ :  
# chords adjacent to  $v$
- $\text{out}(v) = \text{true}$  iff  $v$  is currently outer vertex
- $\text{mark}(v) = \text{true}$  iff  $v$  has received its number



## Lemma.

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



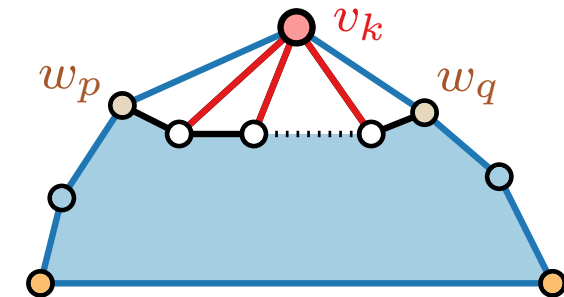
# Canonical Order – Implementation

outer face

```

CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )
forall  $v \in V$  do
  |  $\text{chords}(v) \leftarrow 0$ ;  $\text{out}(v) \leftarrow \text{false}$ ;  $\text{mark}(v) \leftarrow \text{false}$ 
 $\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow \text{true}$ 
for  $k = n$  downto 3 do
  choose  $v$  such that  $\text{mark}(v) = \text{false}$ ,  $\text{out}(v) = \text{true}$ ,
    and  $\text{chords}(v) = 0$  // keep list with candidates
   $v_k \leftarrow v$ ;  $\text{mark}(v) \leftarrow \text{true}$ 
  // Let  $\partial G_{k-1}$  be  $w_1 = v_1, w_2, \dots, w_{t-1}, w_t = v_2$ .
  Let  $w_p, \dots, w_q$  be the unmarked neighbors of  $v_k$ .
  for  $i = p$  to  $q$  do
    |  $\text{out}(w_i) \leftarrow \text{true}$ 
    | update  $\text{chords}(w_i)$ 
    | and for its neighbours
  
```

- $\text{chord}(v)$ :  
# chords adjacent to  $v$
- $\text{out}(v) = \text{true}$  iff  $v$  is currently outer vertex
- $\text{mark}(v) = \text{true}$  iff  $v$  has received its number



## Lemma.

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

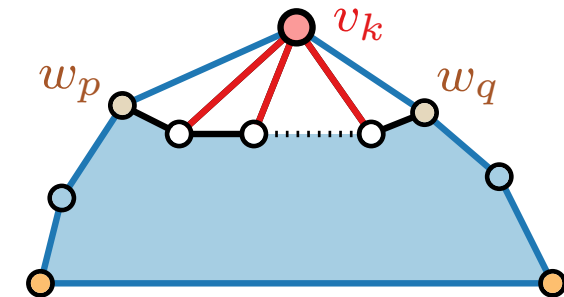
# Canonical Order – Implementation

outer face

```

CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )
forall  $v \in V$  do
   $\text{chords}(v) \leftarrow 0$ ;  $\text{out}(v) \leftarrow \text{false}$ ;  $\text{mark}(v) \leftarrow \text{false}$ 
 $\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow \text{true}$ 
for  $k = n$  downto 3 do
  choose  $v$  such that  $\text{mark}(v) = \text{false}$ ,  $\text{out}(v) = \text{true}$ ,
    and  $\text{chords}(v) = 0$  // keep list with candidates
   $v_k \leftarrow v$ ;  $\text{mark}(v) \leftarrow \text{true}$ 
  // Let  $\partial G_{k-1}$  be  $w_1 = v_1, w_2, \dots, w_{t-1}, w_t = v_2$ .
  Let  $w_p, \dots, w_q$  be the unmarked neighbors of  $v_k$ .
  for  $i = p$  to  $q$  do
     $\text{out}(w_i) \leftarrow \text{true}$  //  $O(n)$  time in total
    update  $\text{chords}(w_i)$ 
    and for its neighbours
  
```

- $\text{chord}(v)$ :  
# chords adjacent to  $v$
- $\text{out}(v) = \text{true}$  iff  $v$  is currently outer vertex
- $\text{mark}(v) = \text{true}$  iff  $v$  has received its number



## Lemma.

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

# Canonical Order – Implementation

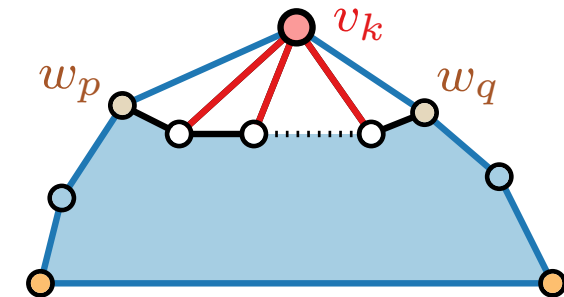
outer face

```

CanonicalOrder( $G = (V, E)$ ,  $(v_1, v_2, v_n)$ )
forall  $v \in V$  do
   $\text{chords}(v) \leftarrow 0$ ;  $\text{out}(v) \leftarrow \text{false}$ ;  $\text{mark}(v) \leftarrow \text{false}$ 
 $\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow \text{true}$ 
for  $k = n$  downto 3 do
  choose  $v$  such that  $\text{mark}(v) = \text{false}$ ,  $\text{out}(v) = \text{true}$ ,
    and  $\text{chords}(v) = 0$            // keep list with candidates
   $v_k \leftarrow v$ ;  $\text{mark}(v) \leftarrow \text{true}$ 
  // Let  $\partial G_{k-1}$  be  $w_1 = v_1, w_2, \dots, w_{t-1}, w_t = v_2$ .
  Let  $w_p, \dots, w_q$  be the unmarked neighbors of  $v_k$ .
  for  $i = p$  to  $q$  do
     $\text{out}(w_i) \leftarrow \text{true}$            //  $O(n)$  time in total
    update  $\text{chords}(w_i)$ 
    and for its neighbours           //  $O(m) = O(n)$  in total

```

- $\text{chord}(v)$ :  
# chords adjacent to  $v$
- $\text{out}(v) = \text{true}$  iff  $v$  is currently outer vertex
- $\text{mark}(v) = \text{true}$  iff  $v$  has received its number



## Lemma.

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

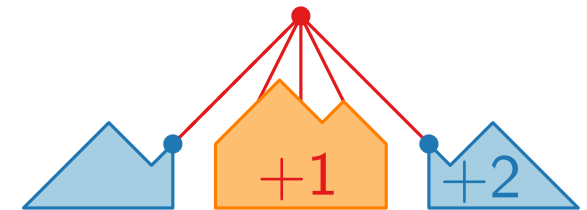
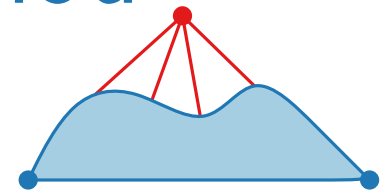
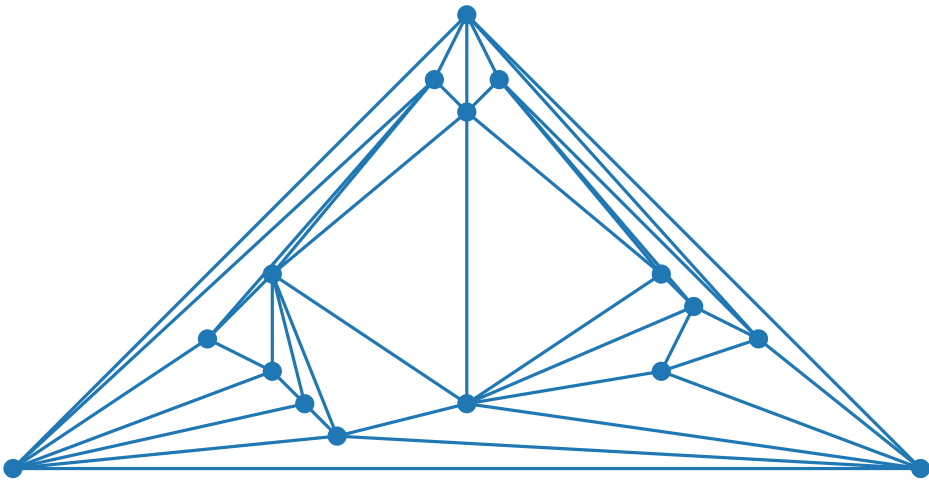
# Visualization of Graphs

## Lecture 3:

### Straight-Line Drawings of Planar Graphs I: Canonical Ordering and the Shift Method

#### Part III: The Shift Method

Alexander Wolff



# Shift Method – Idea

**Drawing invariants:**

$G_{k-1}$  is drawn such that

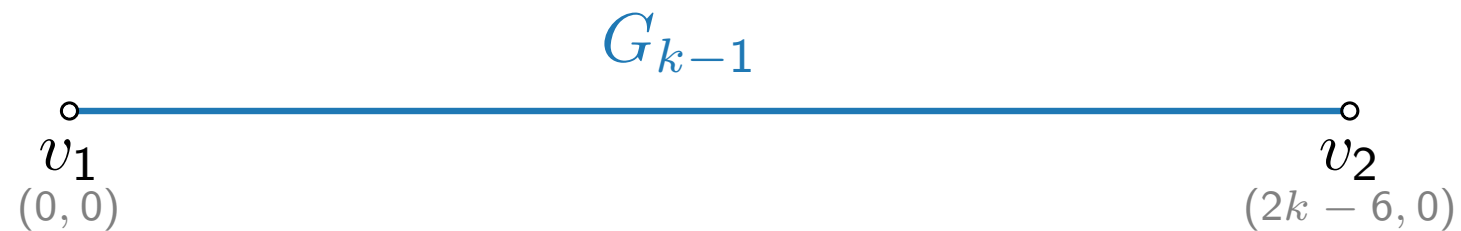
$$G_{k-1}$$

# Shift Method – Idea

## Drawing invariants:

$G_{k-1}$  is drawn such that

- $v_1$  is at  $(0, 0)$ ,  $v_2$  is at  $(2k - 6, 0)$ ,

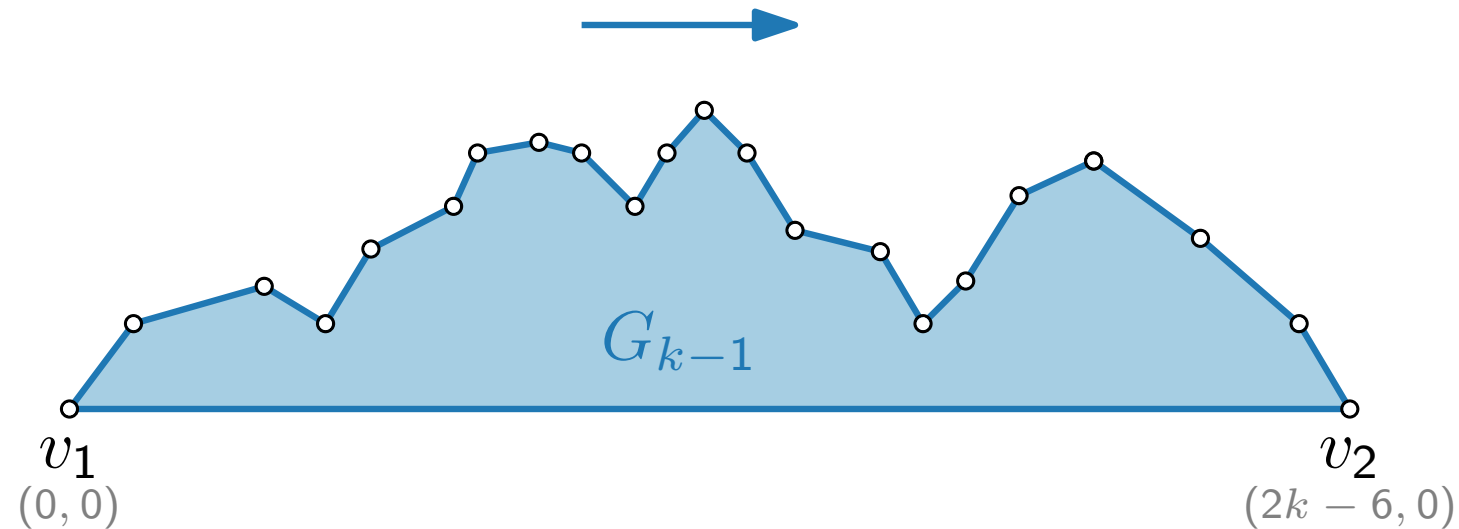


# Shift Method – Idea

## Drawing invariants:

$G_{k-1}$  is drawn such that

- $v_1$  is at  $(0, 0)$ ,  $v_2$  is at  $(2k - 6, 0)$ ,
- boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn  $x$ -monotone,

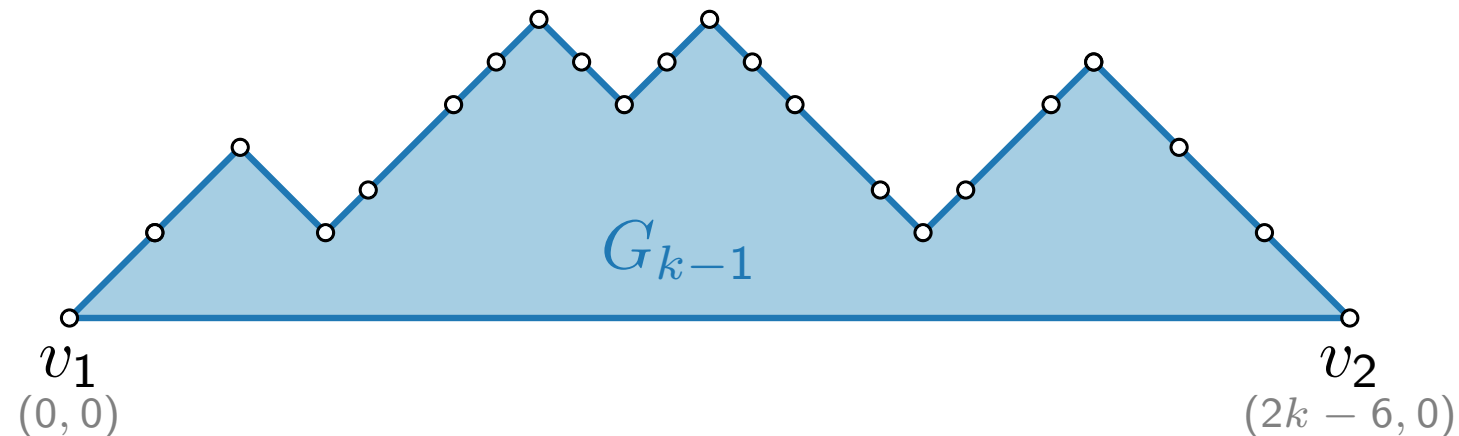


# Shift Method – Idea

## Drawing invariants:

$G_{k-1}$  is drawn such that

- $v_1$  is at  $(0, 0)$ ,  $v_2$  is at  $(2k - 6, 0)$ ,
- boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn  $x$ -monotone,
- each edge of the boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn with slopes  $\pm 1$ .



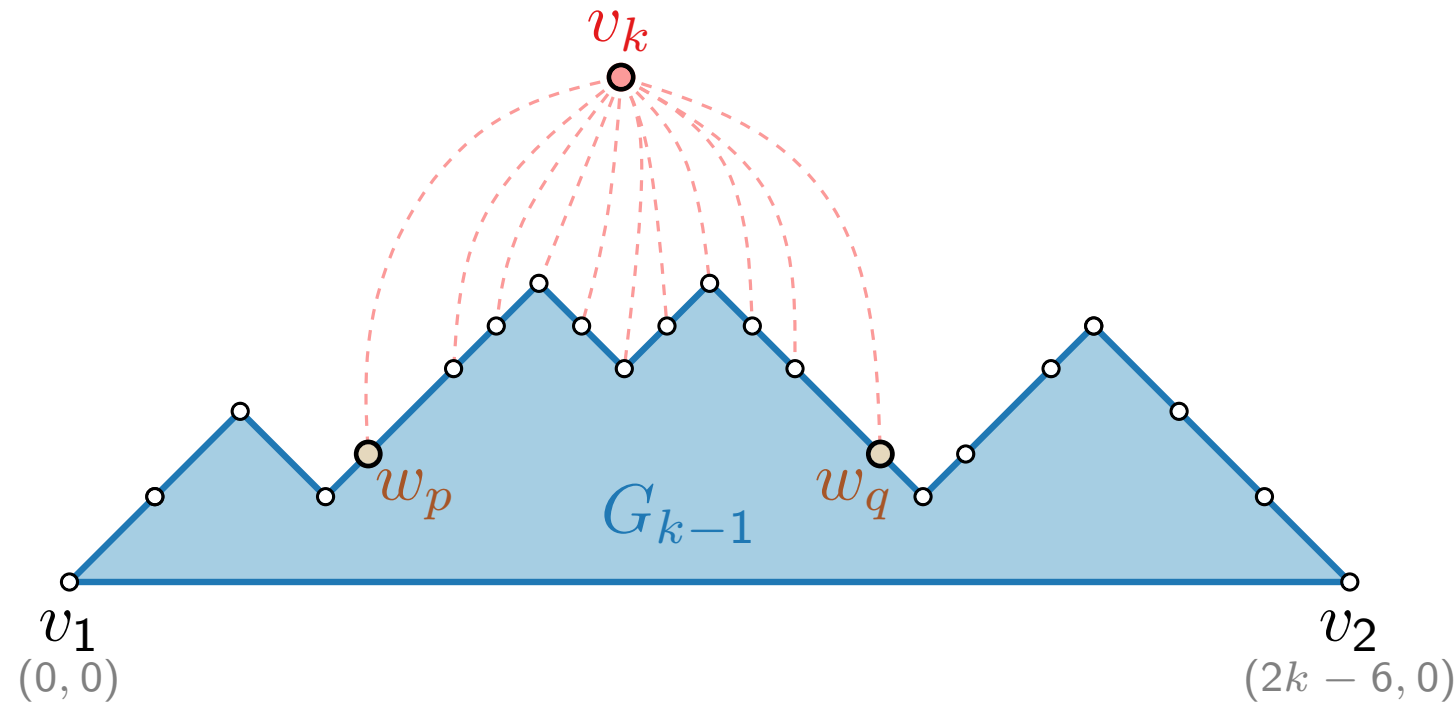


# Shift Method – Idea

## Drawing invariants:

$G_{k-1}$  is drawn such that

- $v_1$  is at  $(0, 0)$ ,  $v_2$  is at  $(2k - 6, 0)$ ,
- boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn  $x$ -monotone,
- each edge of the boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn with slopes  $\pm 1$ .

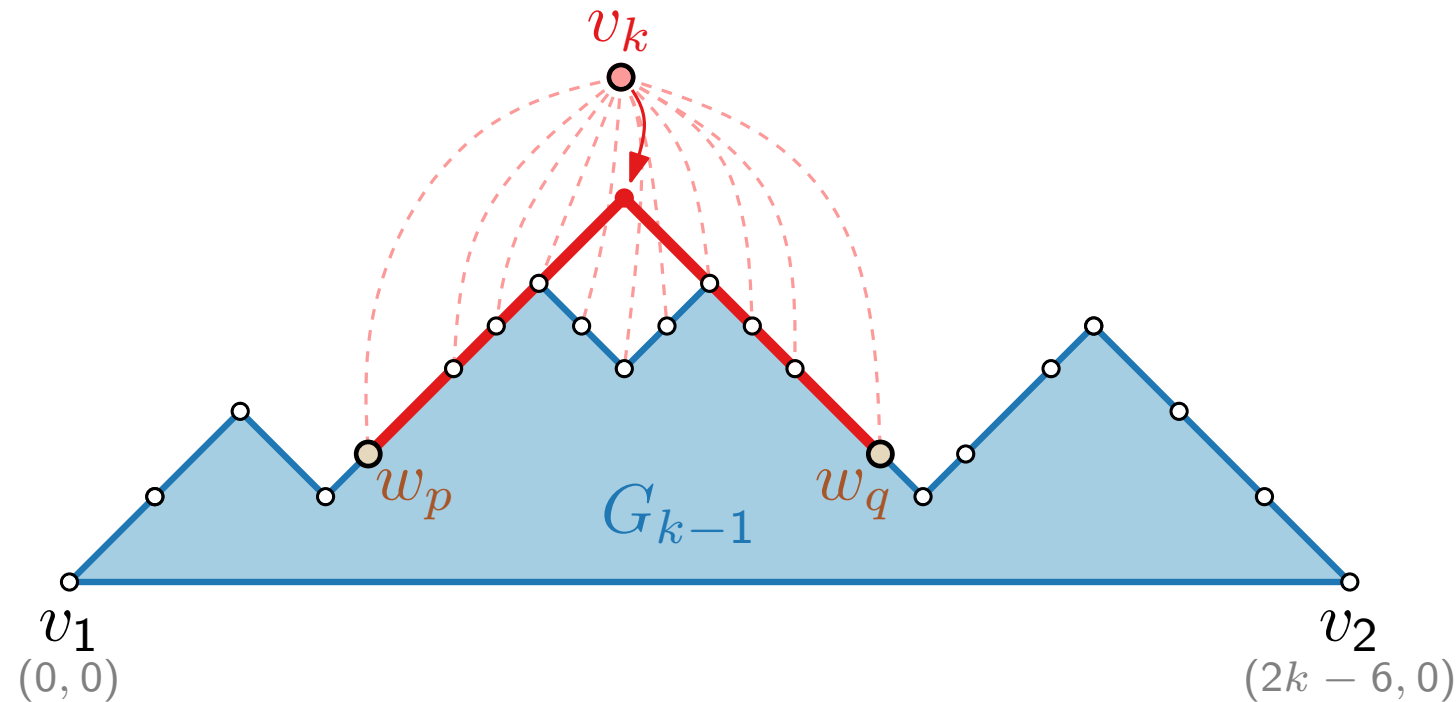


# Shift Method – Idea

## Drawing invariants:

$G_{k-1}$  is drawn such that

- $v_1$  is at  $(0, 0)$ ,  $v_2$  is at  $(2k - 6, 0)$ ,
- boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn  $x$ -monotone,
- each edge of the boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn with slopes  $\pm 1$ .

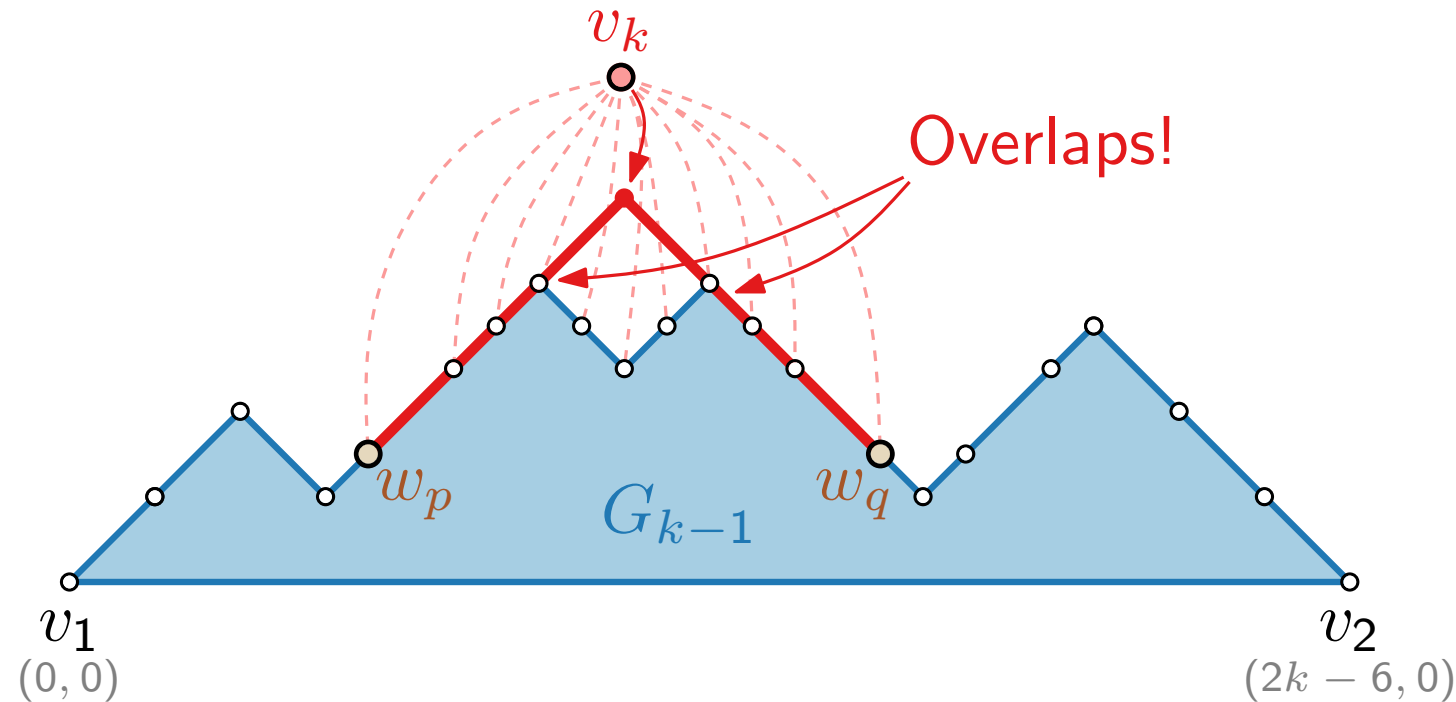


# Shift Method – Idea

## Drawing invariants:

$G_{k-1}$  is drawn such that

- $v_1$  is at  $(0, 0)$ ,  $v_2$  is at  $(2k - 6, 0)$ ,
- boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn  $x$ -monotone,
- each edge of the boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn with slopes  $\pm 1$ .

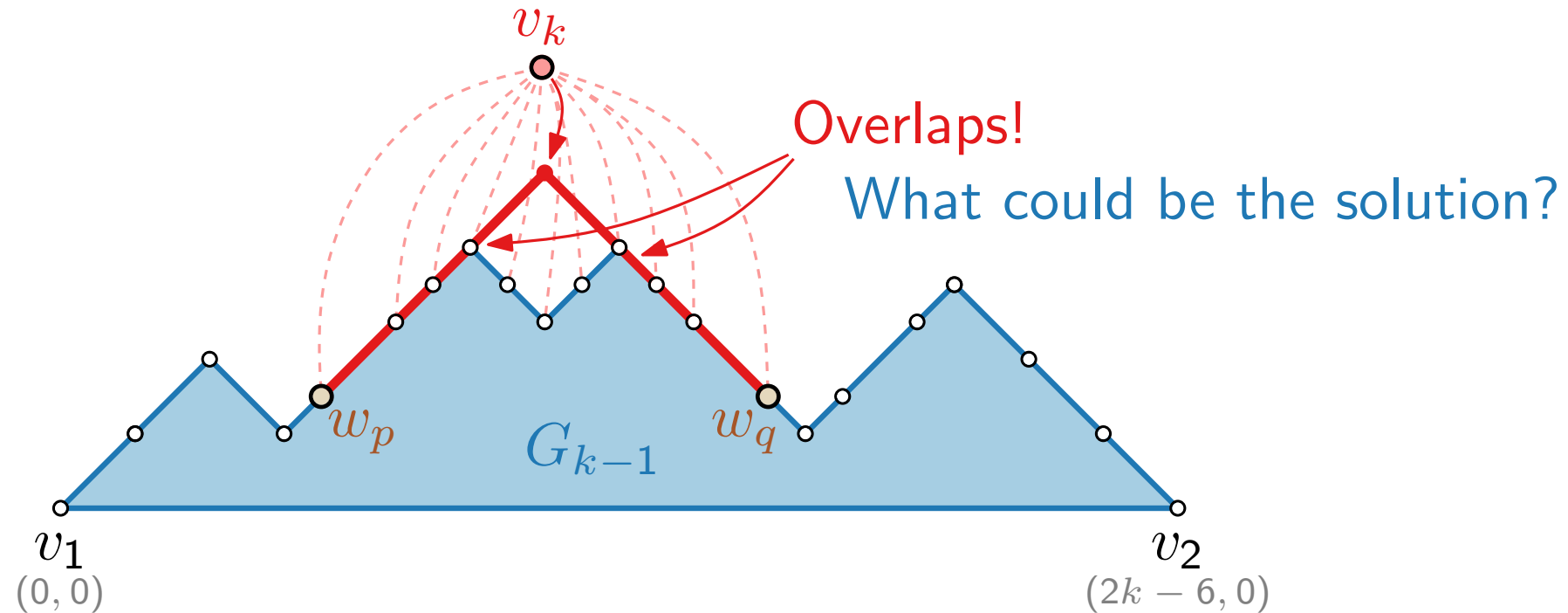


# Shift Method – Idea

## Drawing invariants:

$G_{k-1}$  is drawn such that

- $v_1$  is at  $(0, 0)$ ,  $v_2$  is at  $(2k - 6, 0)$ ,
- boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn  $x$ -monotone,
- each edge of the boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn with slopes  $\pm 1$ .

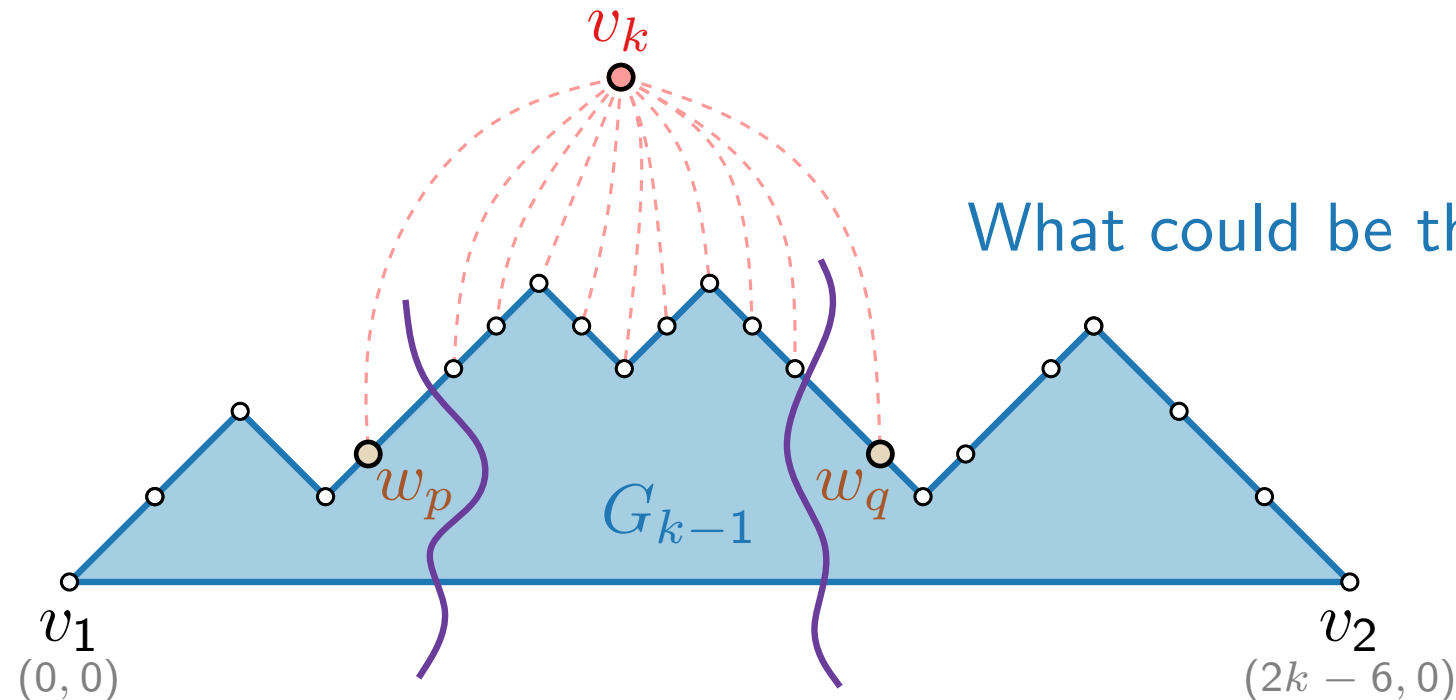


# Shift Method – Idea

## Drawing invariants:

$G_{k-1}$  is drawn such that

- $v_1$  is at  $(0, 0)$ ,  $v_2$  is at  $(2k - 6, 0)$ ,
- boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn  $x$ -monotone,
- each edge of the boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn with slopes  $\pm 1$ .



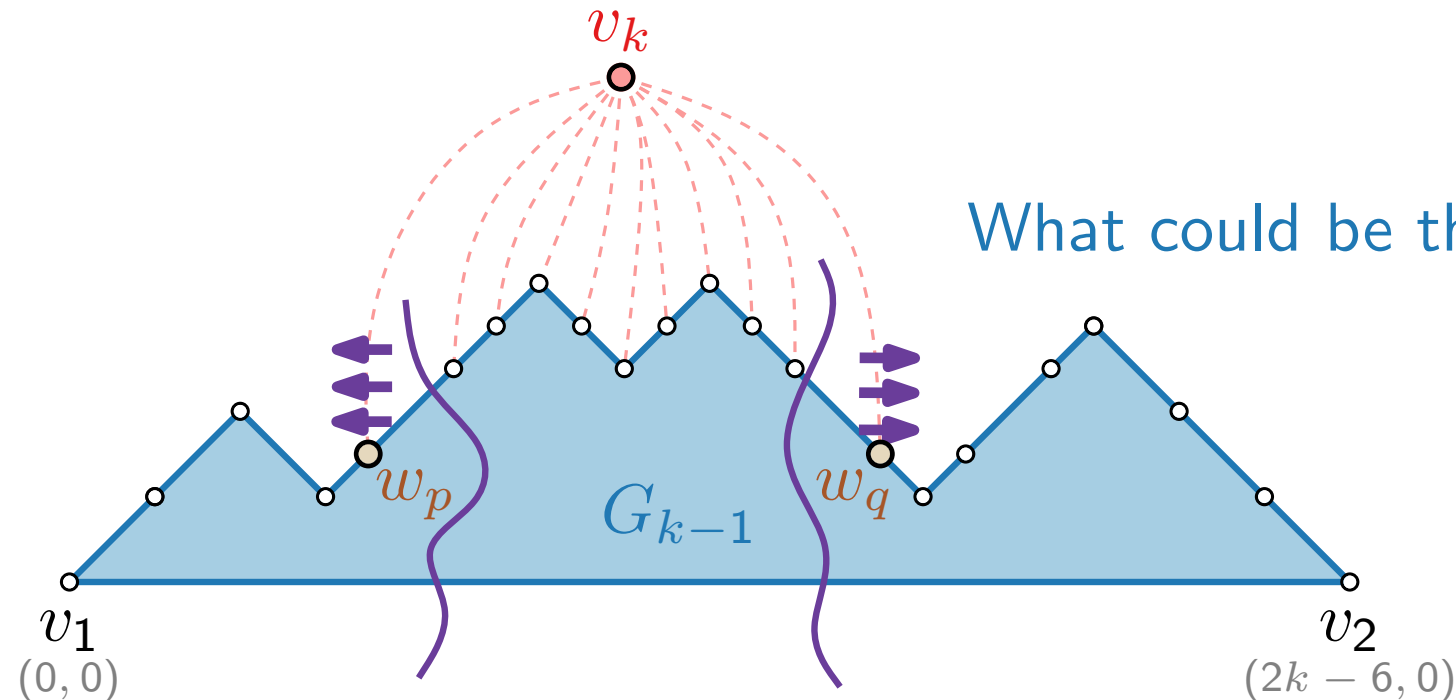
What could be the solution?

# Shift Method – Idea

## Drawing invariants:

$G_{k-1}$  is drawn such that

- $v_1$  is at  $(0, 0)$ ,  $v_2$  is at  $(2k - 6, 0)$ ,
- boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn  $x$ -monotone,
- each edge of the boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn with slopes  $\pm 1$ .

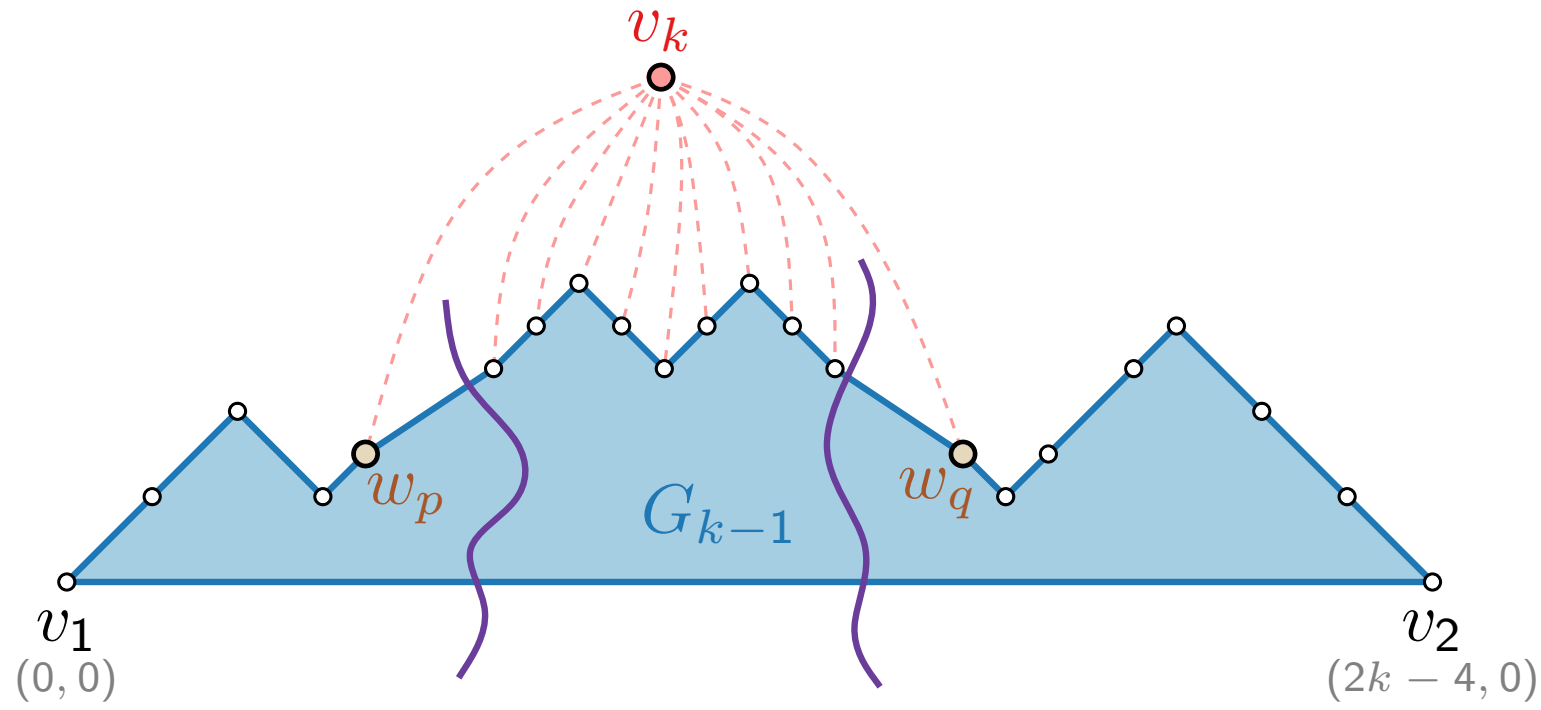


# Shift Method – Idea

## Drawing invariants:

$G_{k-1}$  is drawn such that

- $v_1$  is at  $(0, 0)$ ,  $v_2$  is at  $(2k - 6, 0)$ ,
- boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn  $x$ -monotone,
- each edge of the boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn with slopes  $\pm 1$ .

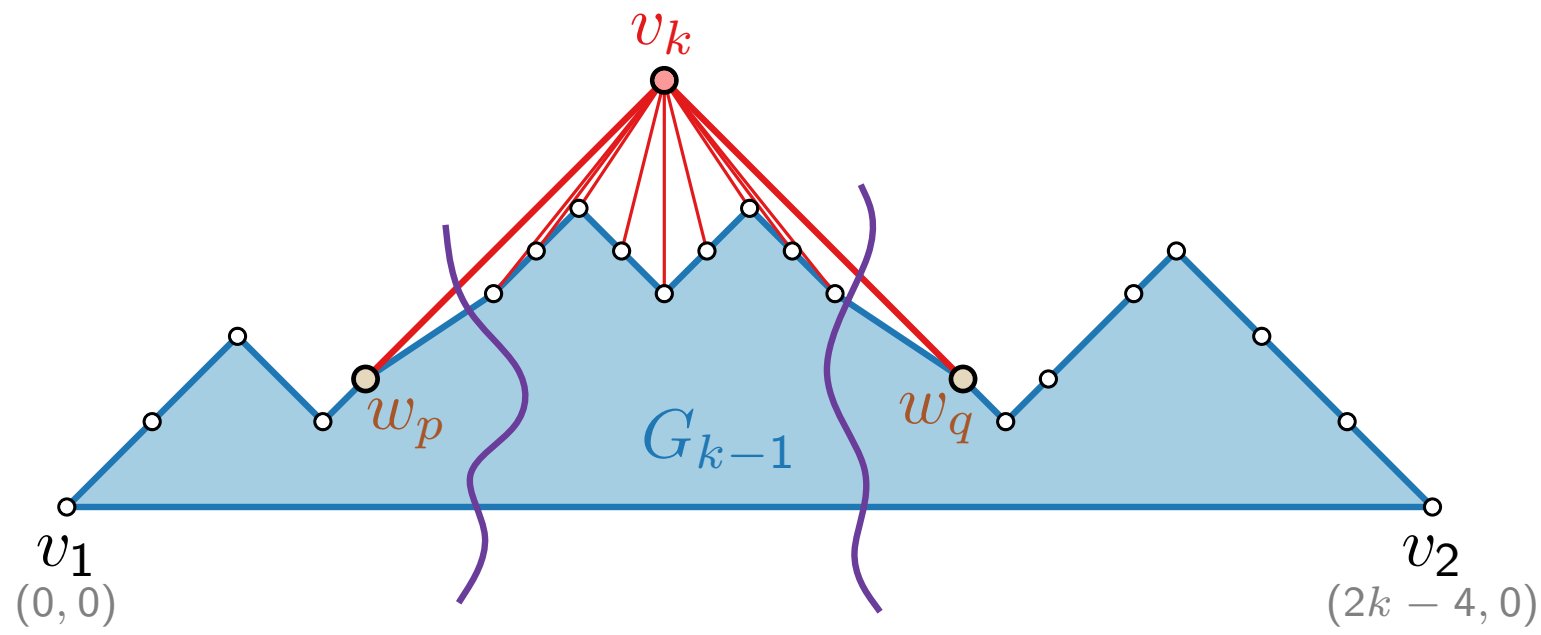


# Shift Method – Idea

## Drawing invariants:

$G_{k-1}$  is drawn such that

- $v_1$  is at  $(0, 0)$ ,  $v_2$  is at  $(2k - 6, 0)$ ,
- boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn  $x$ -monotone,
- each edge of the boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn with slopes  $\pm 1$ .





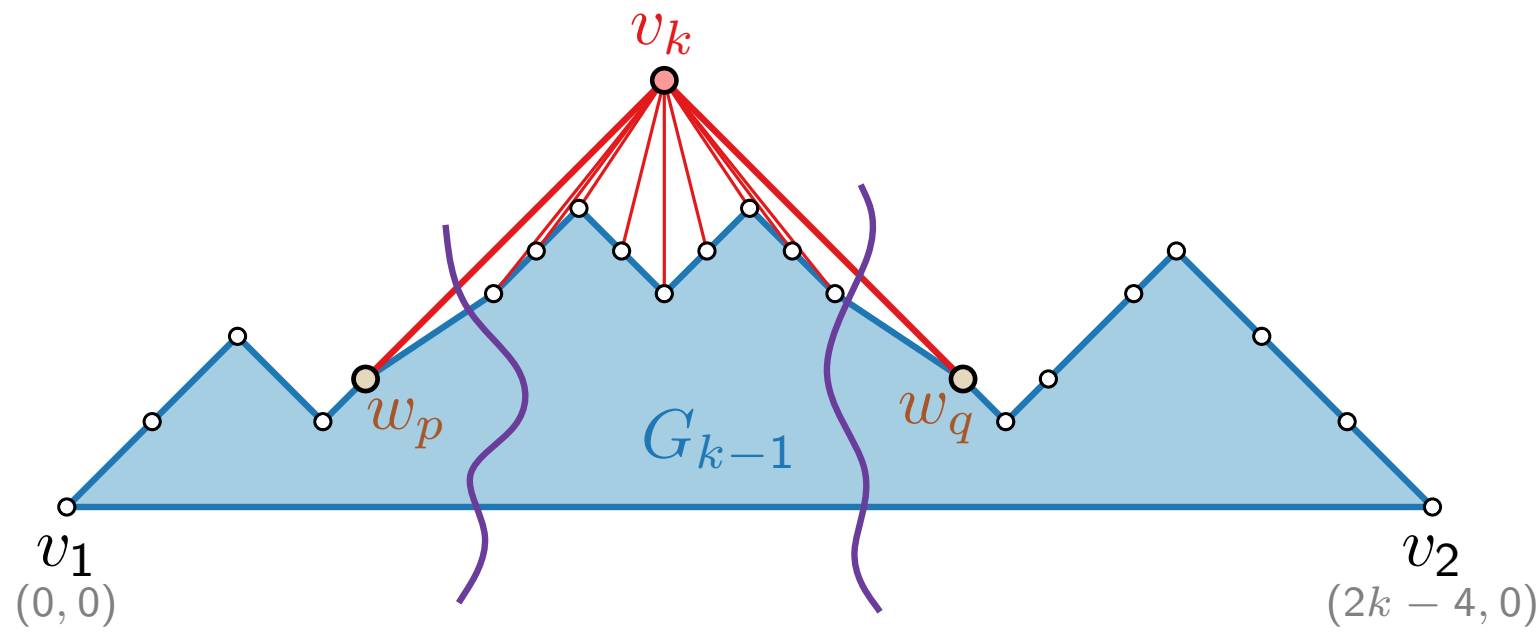
# Shift Method – Idea

## Drawing invariants:

$G_{k-1}$  is drawn such that

- $v_1$  is at  $(0, 0)$ ,  $v_2$  is at  $(2k - 6, 0)$ ,
- boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn  $x$ -monotone,
- each edge of the boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn with slopes  $\pm 1$ .

Will  $v_k$  lie on the grid?



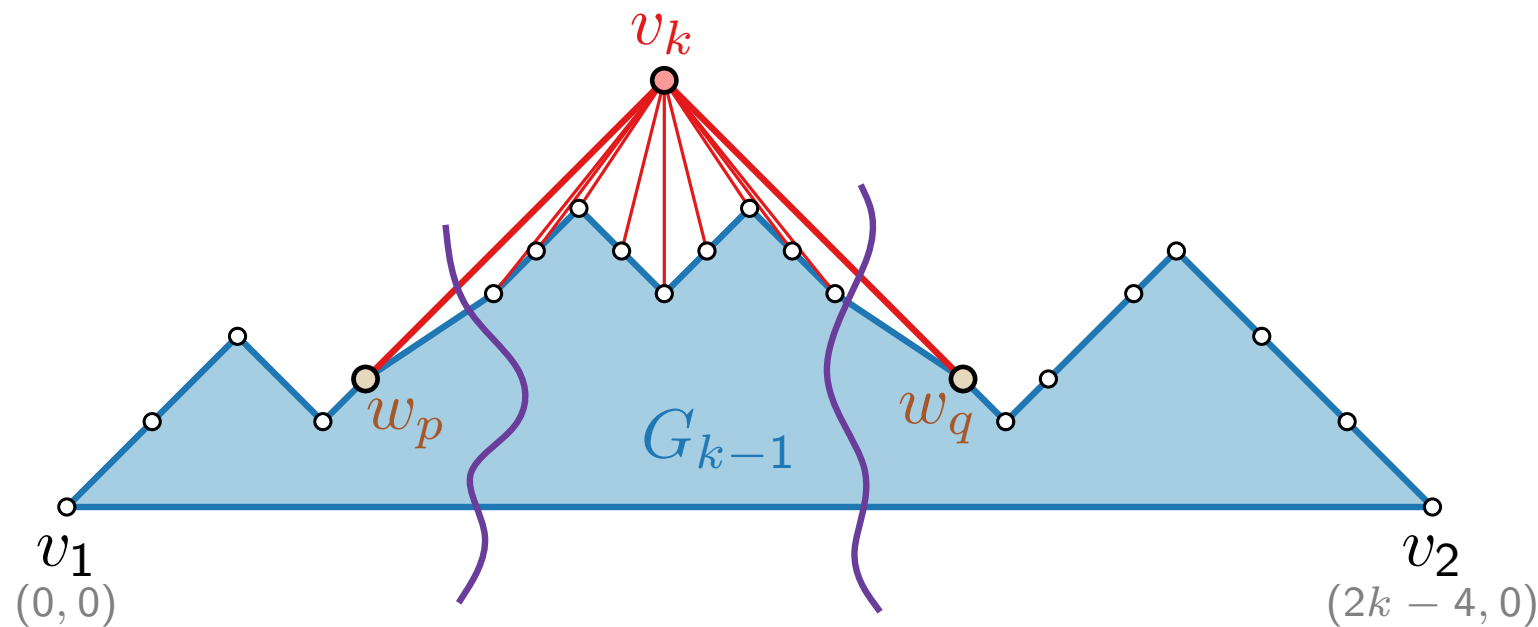
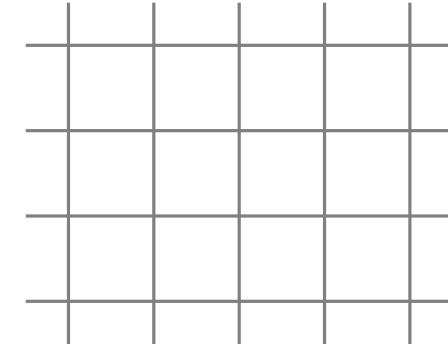
# Shift Method – Idea

## Drawing invariants:

$G_{k-1}$  is drawn such that

- $v_1$  is at  $(0, 0)$ ,  $v_2$  is at  $(2k - 6, 0)$ ,
- boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn  $x$ -monotone,
- each edge of the boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn with slopes  $\pm 1$ .

Will  $v_k$  lie on the grid?



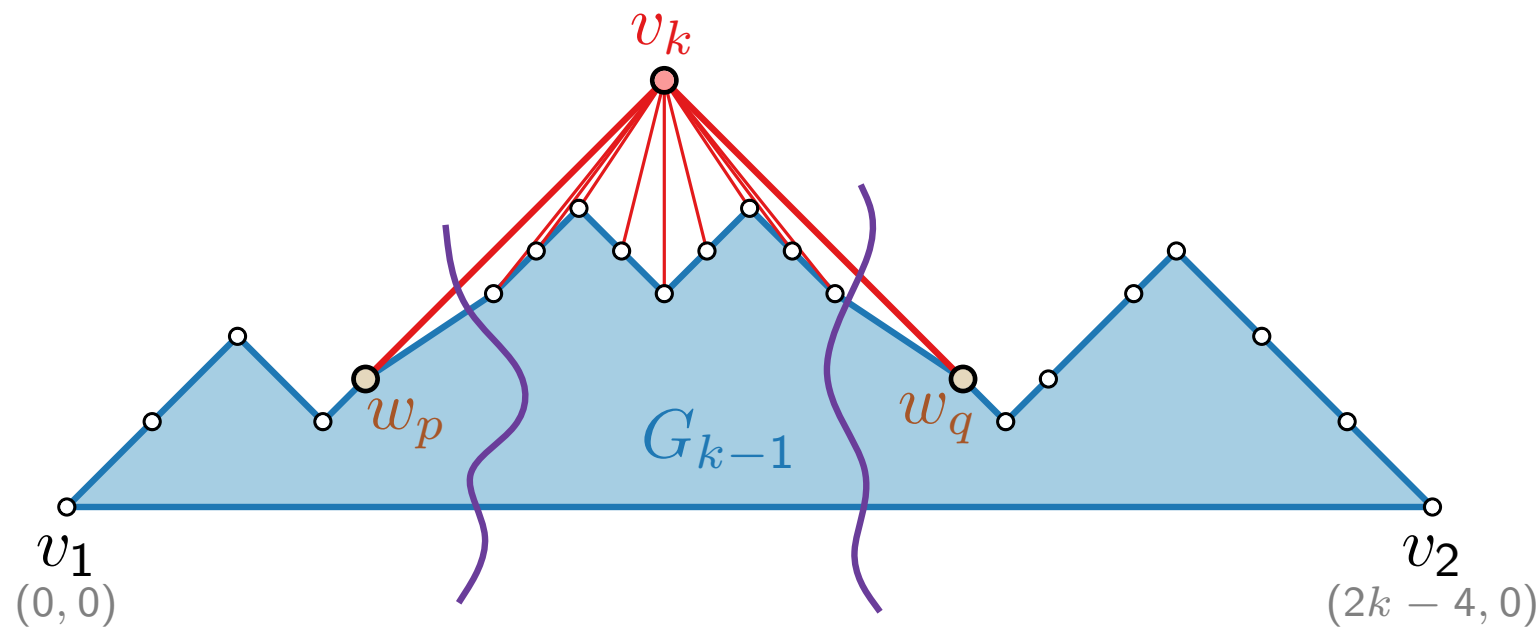
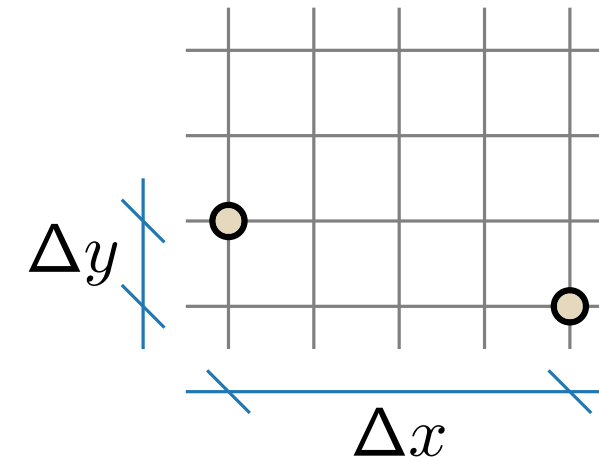
# Shift Method – Idea

## Drawing invariants:

$G_{k-1}$  is drawn such that

- $v_1$  is at  $(0, 0)$ ,  $v_2$  is at  $(2k - 6, 0)$ ,
- boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn  $x$ -monotone,
- each edge of the boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn with slopes  $\pm 1$ .

Will  $v_k$  lie on the grid?



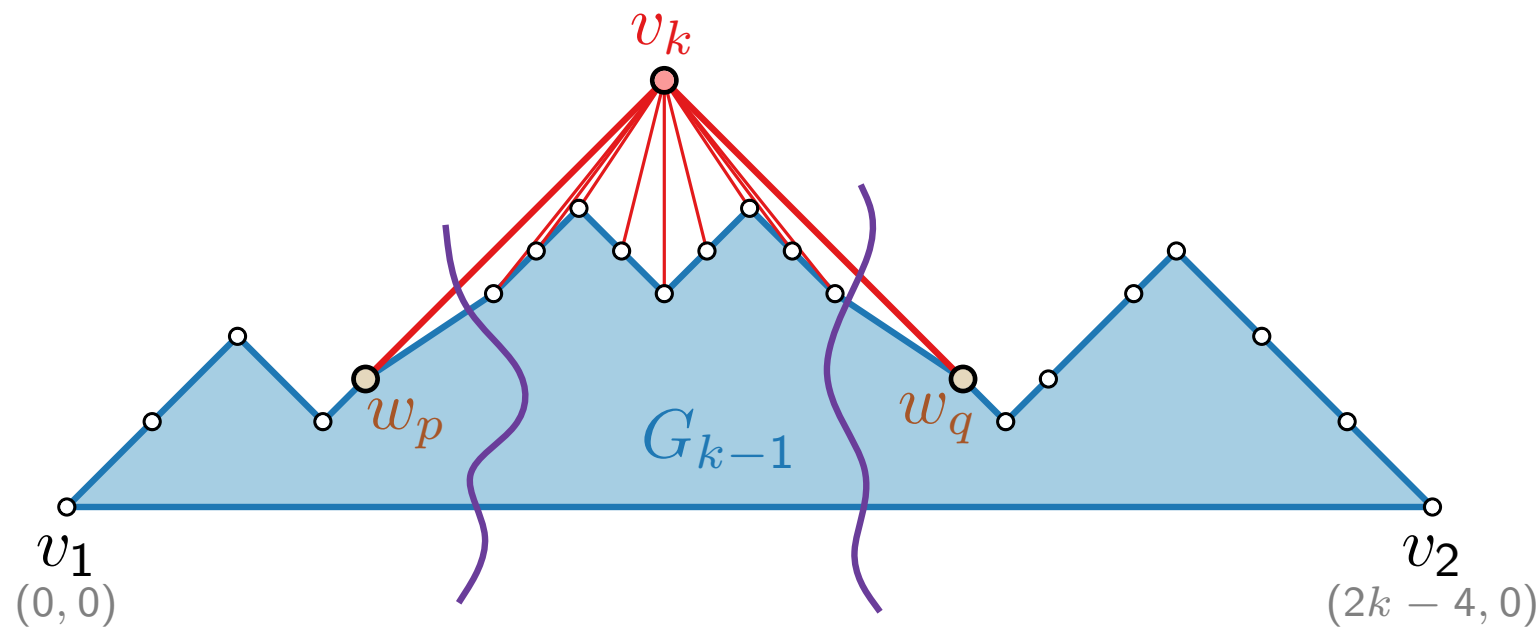
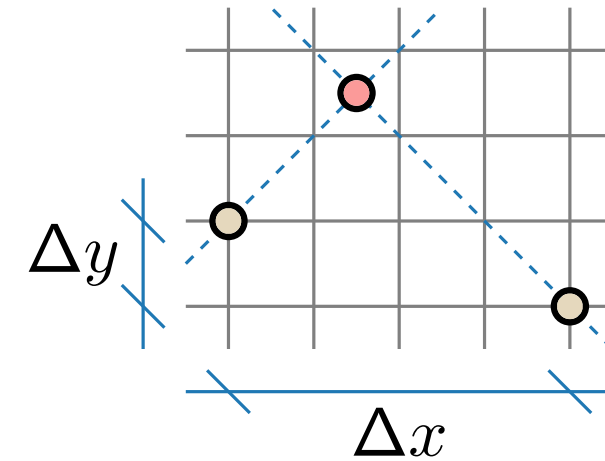
# Shift Method – Idea

## Drawing invariants:

$G_{k-1}$  is drawn such that

- $v_1$  is at  $(0, 0)$ ,  $v_2$  is at  $(2k - 6, 0)$ ,
- boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn  $x$ -monotone,
- each edge of the boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn with slopes  $\pm 1$ .

Will  $v_k$  lie on the grid?



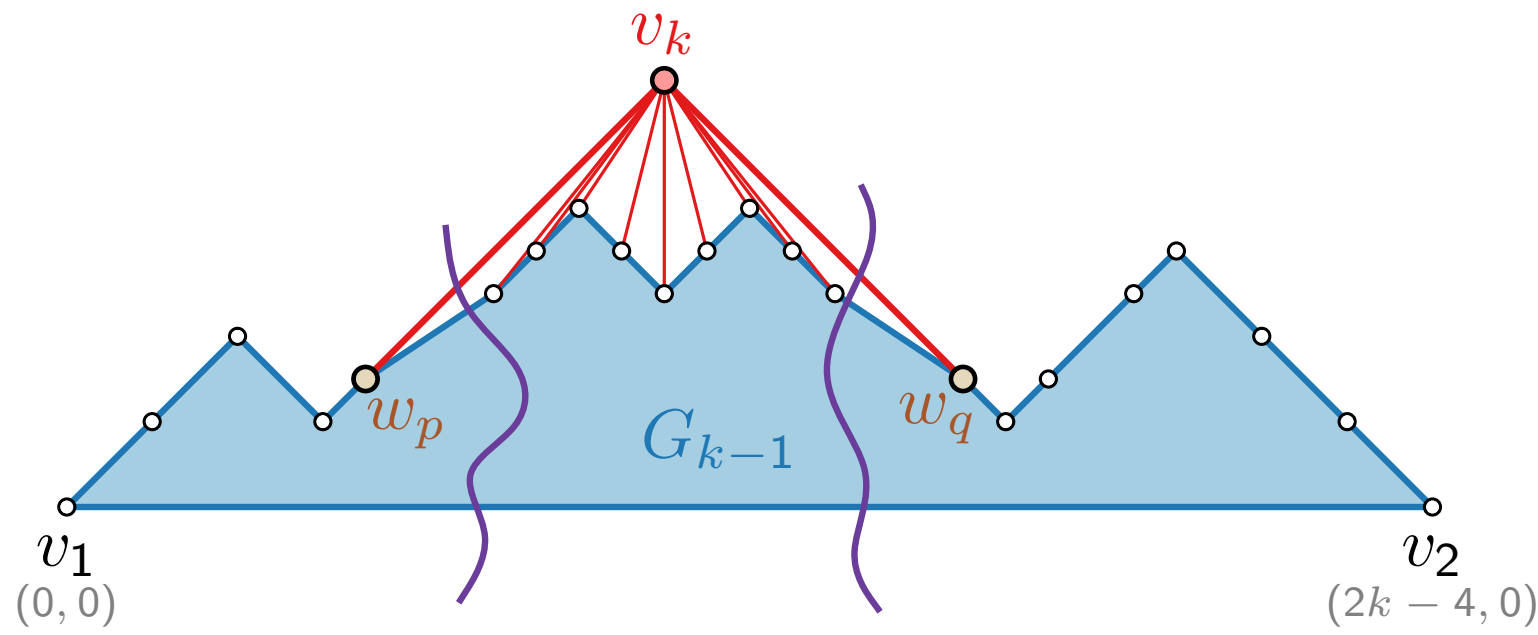
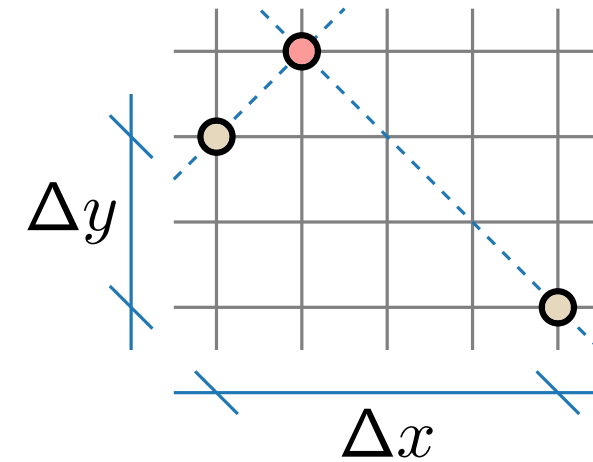
# Shift Method – Idea

## Drawing invariants:

$G_{k-1}$  is drawn such that

- $v_1$  is at  $(0, 0)$ ,  $v_2$  is at  $(2k - 6, 0)$ ,
- boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn  $x$ -monotone,
- each edge of the boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn with slopes  $\pm 1$ .

Will  $v_k$  lie on the grid?



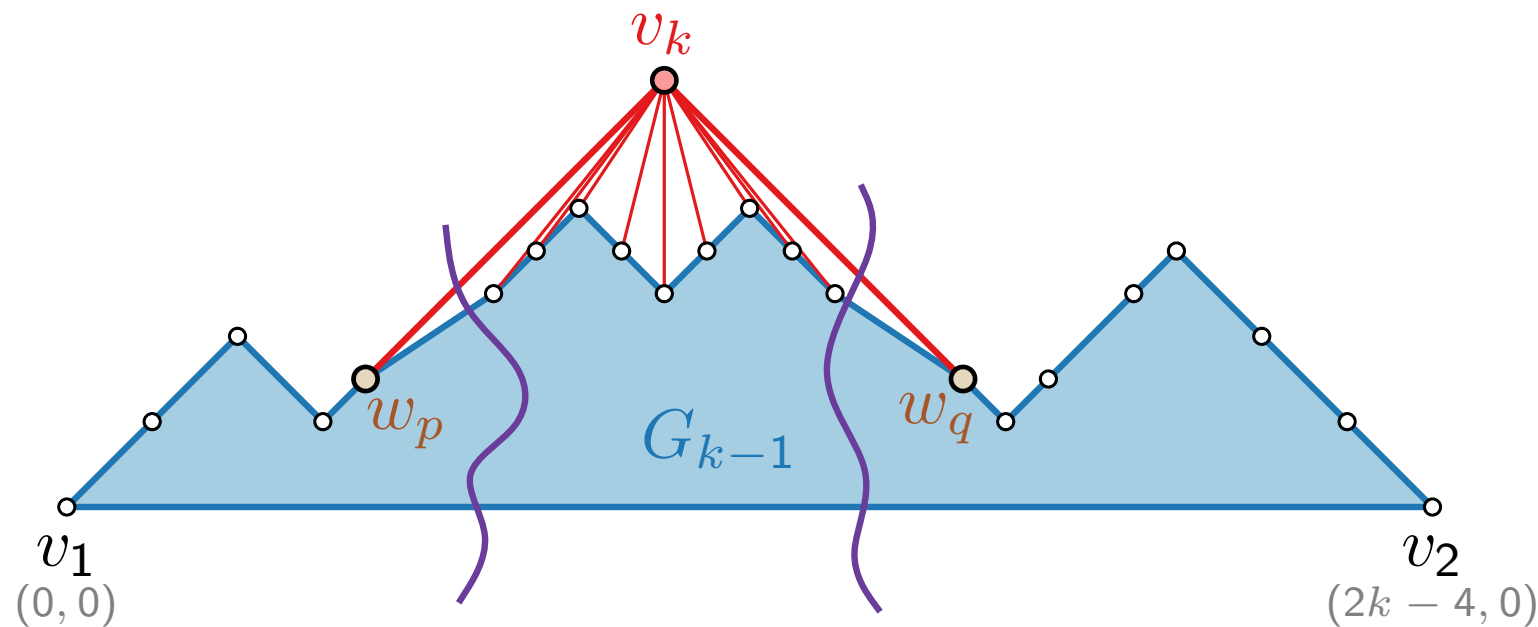
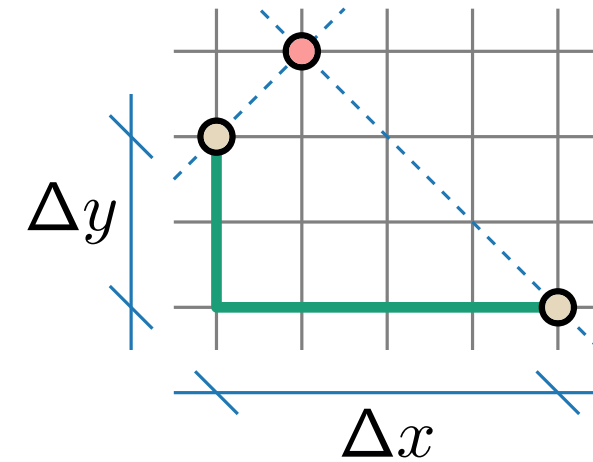
# Shift Method – Idea

## Drawing invariants:

$G_{k-1}$  is drawn such that

- $v_1$  is at  $(0, 0)$ ,  $v_2$  is at  $(2k - 6, 0)$ ,
- boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn  $x$ -monotone,
- each edge of the boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn with slopes  $\pm 1$ .

Will  $v_k$  lie on the grid?



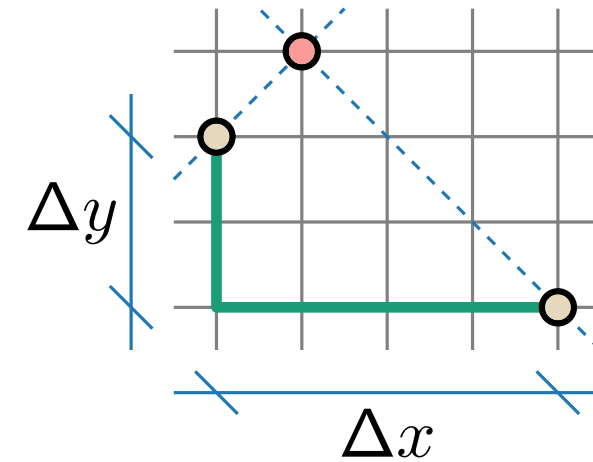
# Shift Method – Idea

## Drawing invariants:

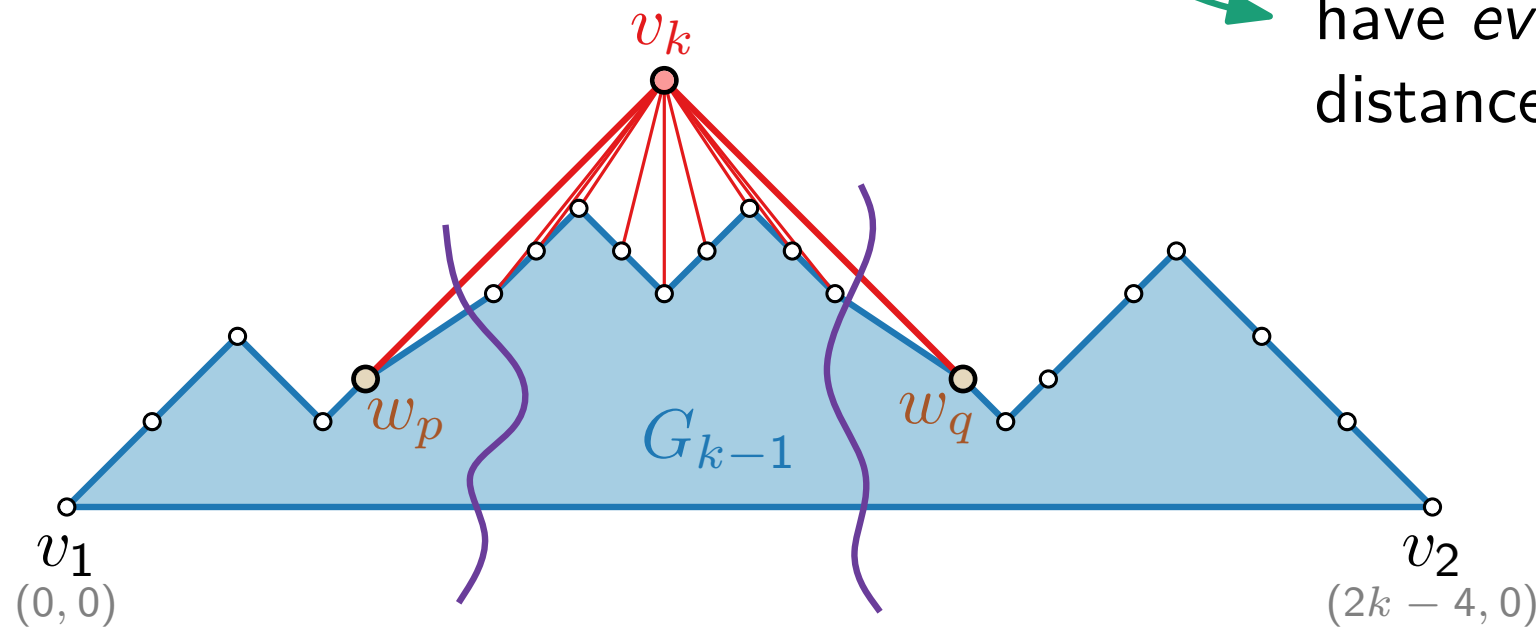
$G_{k-1}$  is drawn such that

- $v_1$  is at  $(0, 0)$ ,  $v_2$  is at  $(2k - 6, 0)$ ,
- boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn  $x$ -monotone,
- each edge of the boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn with slopes  $\pm 1$ .

Will  $v_k$  lie on the grid?



Yes, because  $w_p$  and  $w_q$  have even **Manhattan** distance  $\Delta x + \Delta y$ .



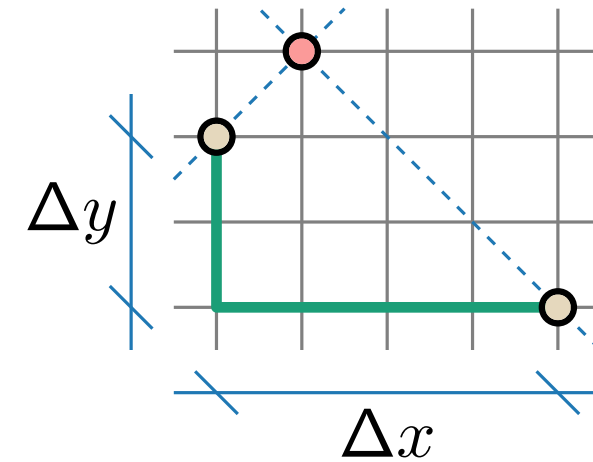
# Shift Method – Idea

## Drawing invariants:

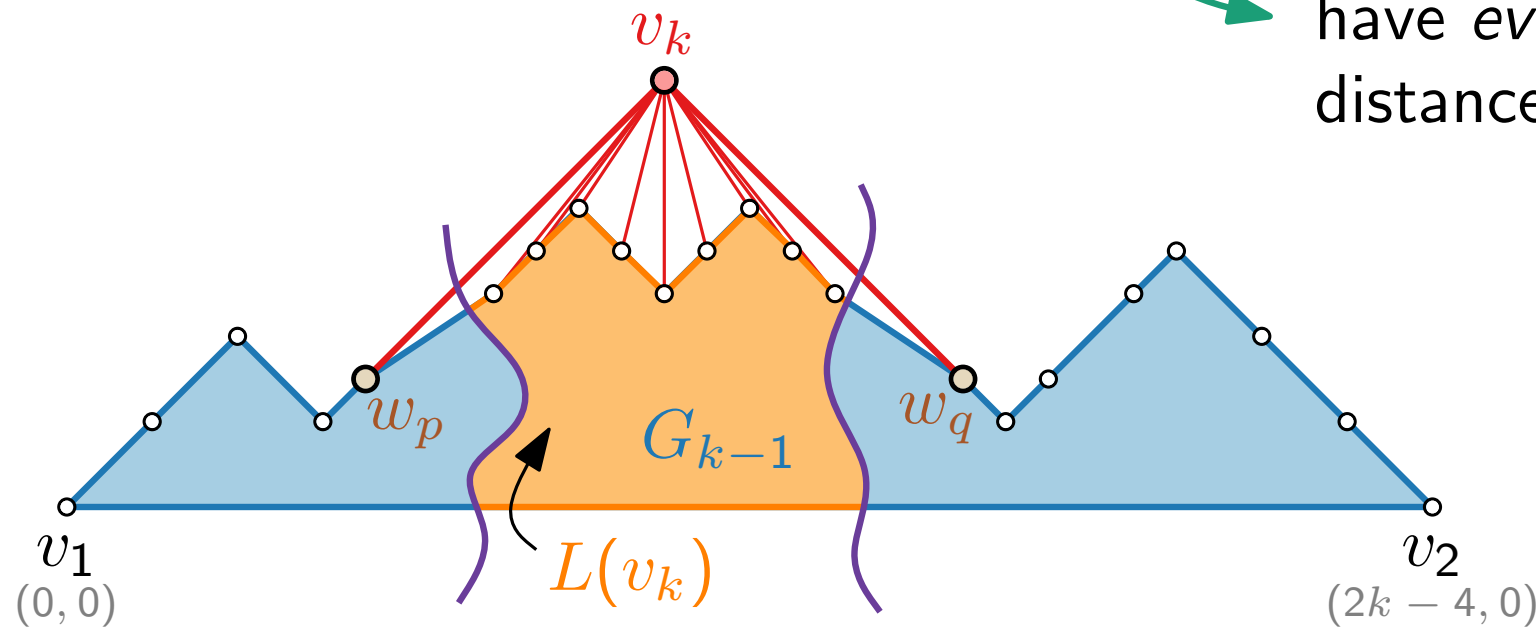
$G_{k-1}$  is drawn such that

- $v_1$  is at  $(0, 0)$ ,  $v_2$  is at  $(2k - 6, 0)$ ,
- boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn  $x$ -monotone,
- each edge of the boundary of  $G_{k-1}$  (minus edge  $(v_1, v_2)$ ) is drawn with slopes  $\pm 1$ .

Will  $v_k$  lie on the grid?

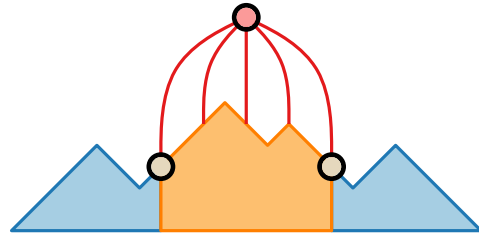


Yes, because  $w_p$  and  $w_q$  have even **Manhattan** distance  $\Delta x + \Delta y$ .

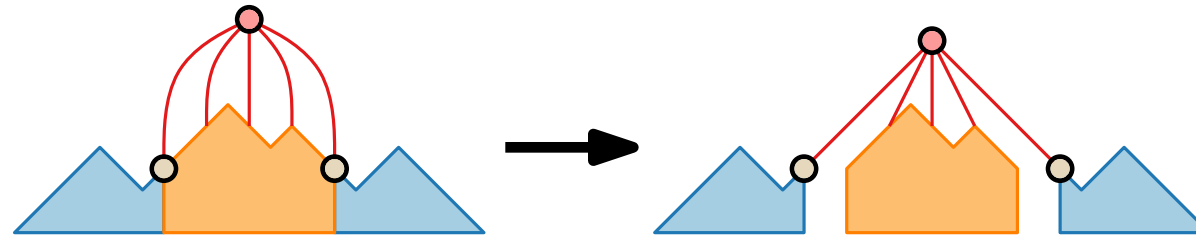




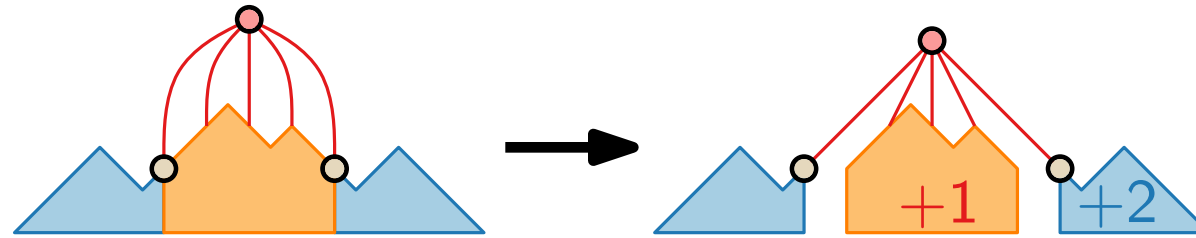
# Shift Method – Example



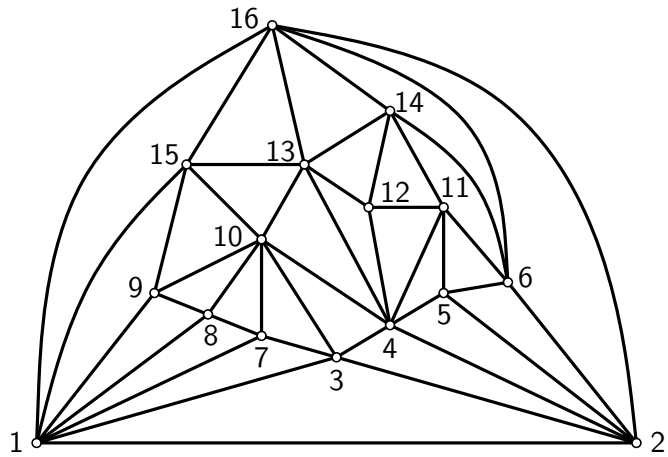
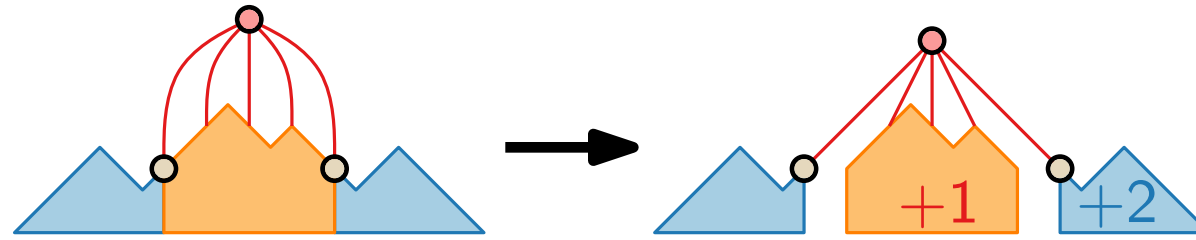
# Shift Method – Example



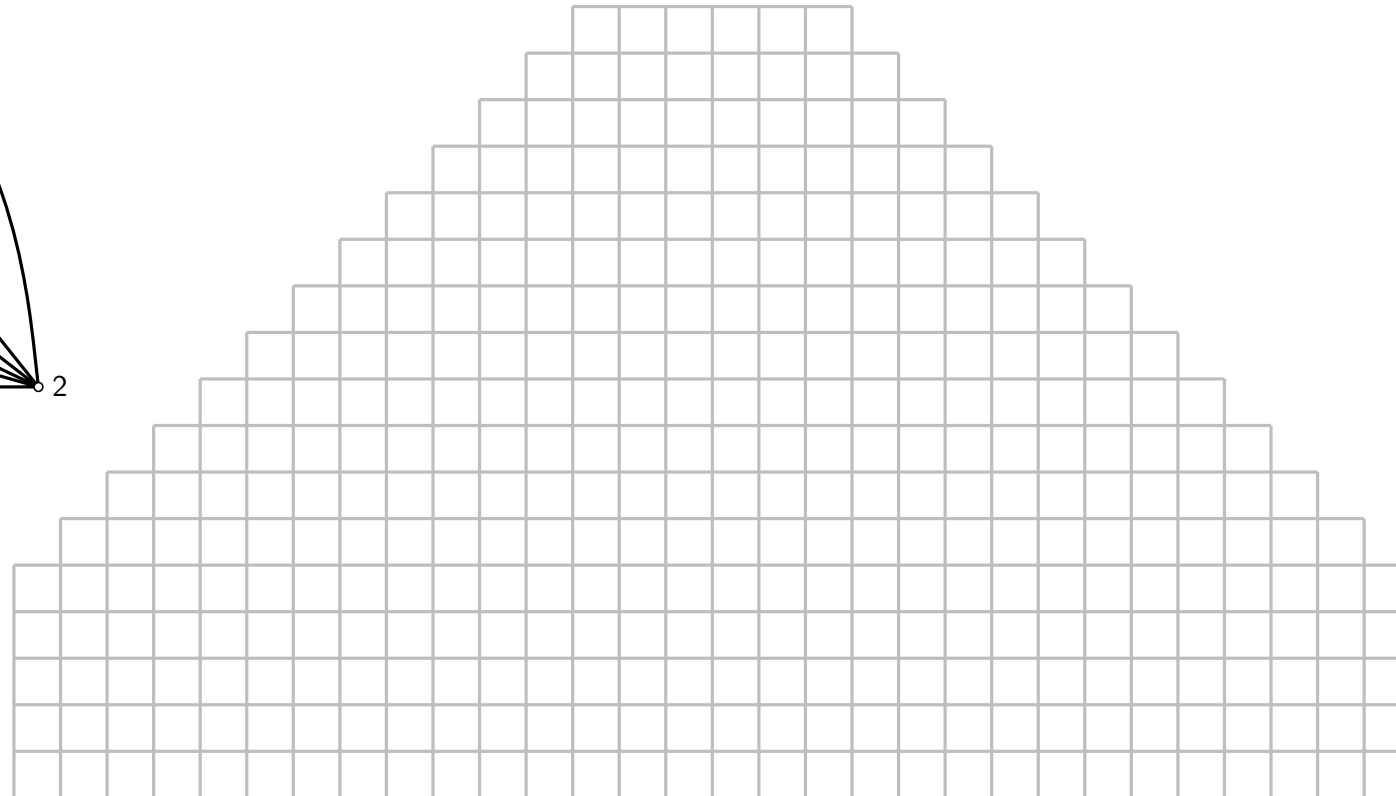
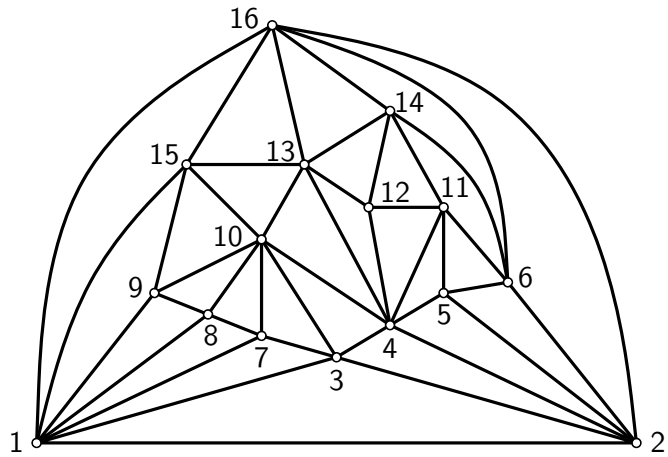
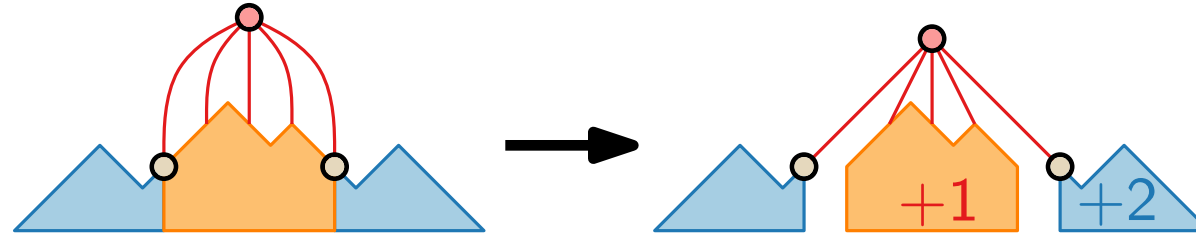
# Shift Method – Example



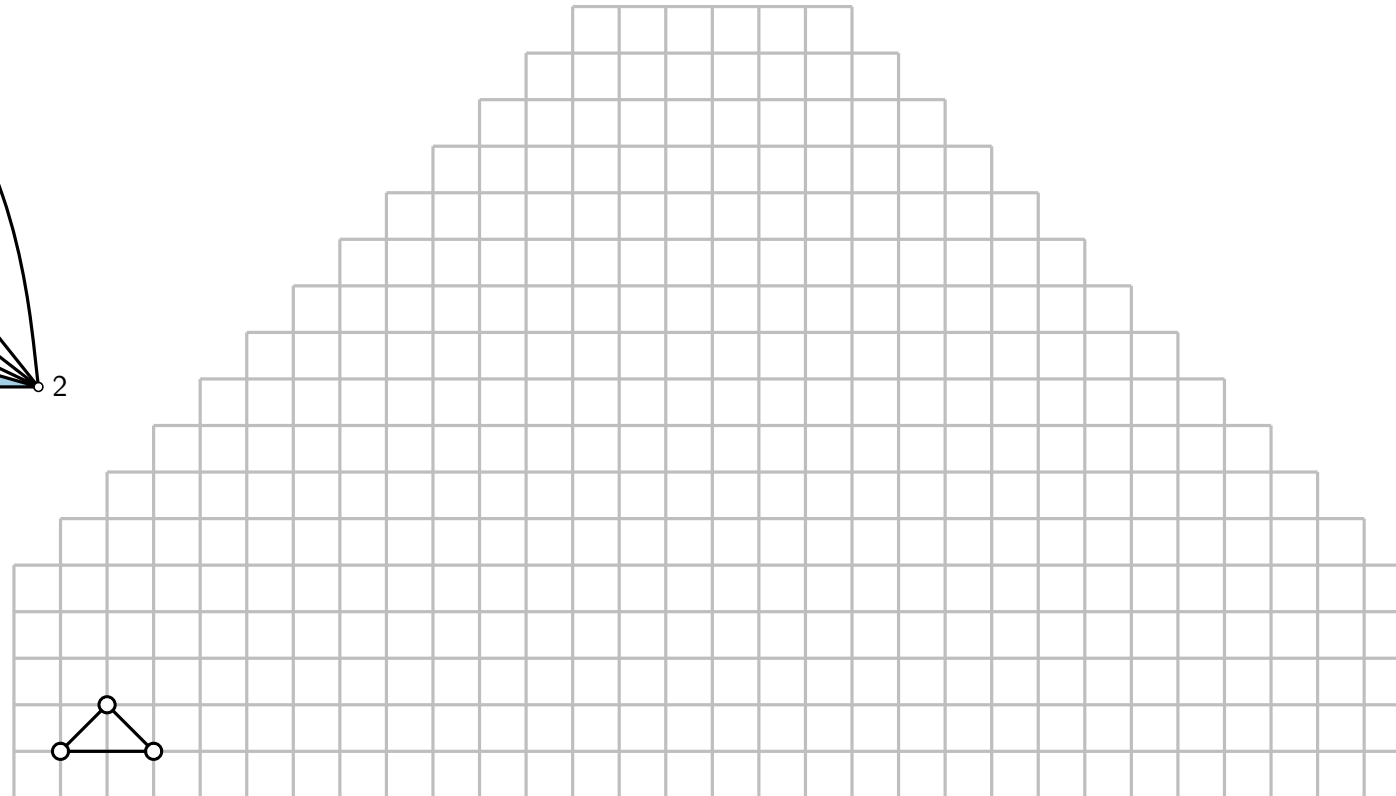
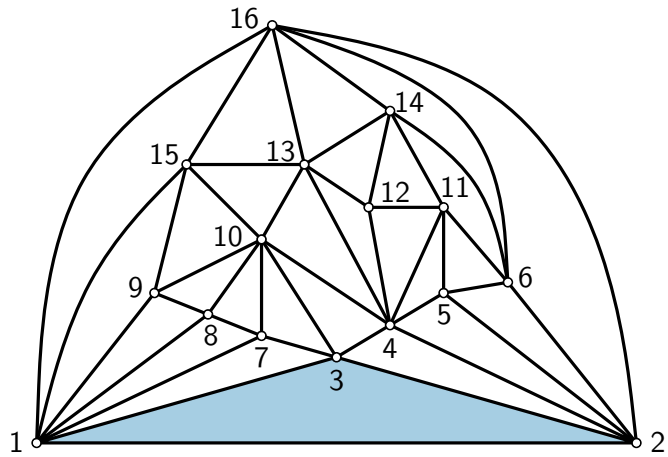
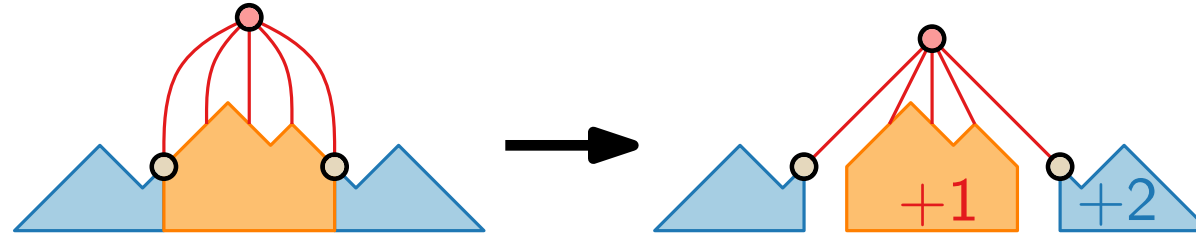
# Shift Method – Example



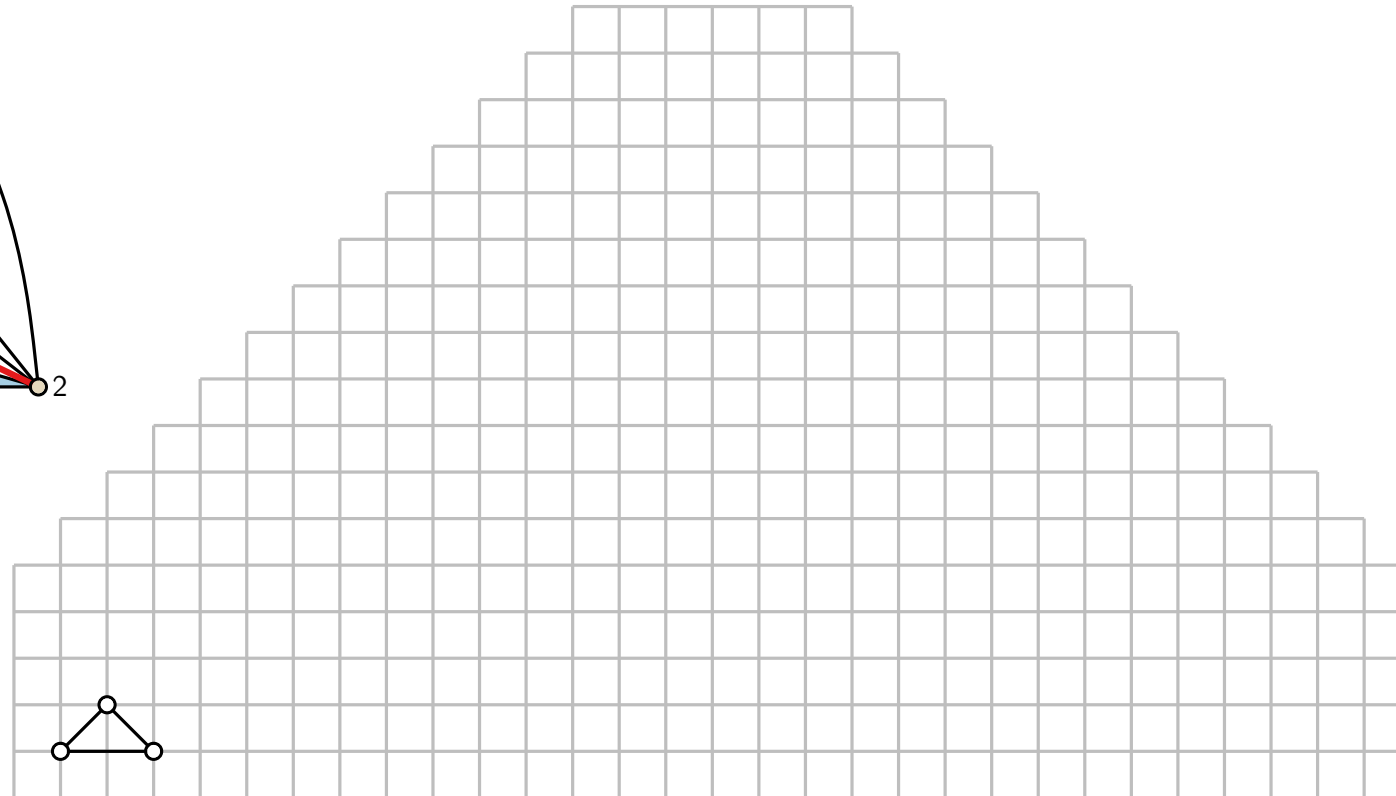
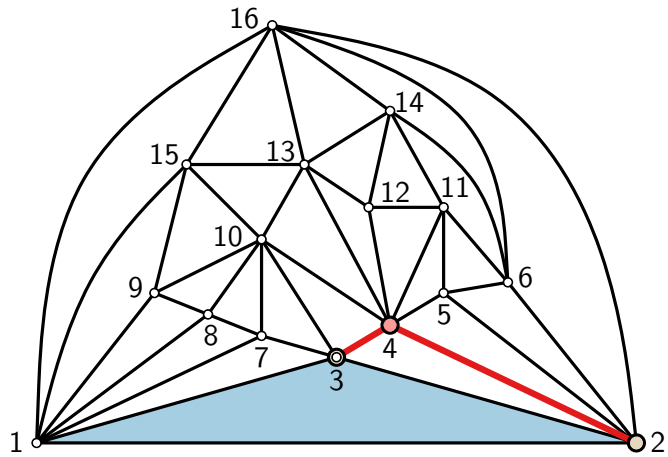
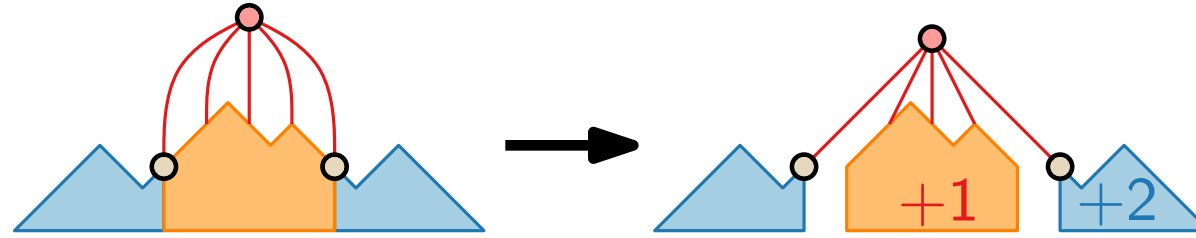
# Shift Method – Example



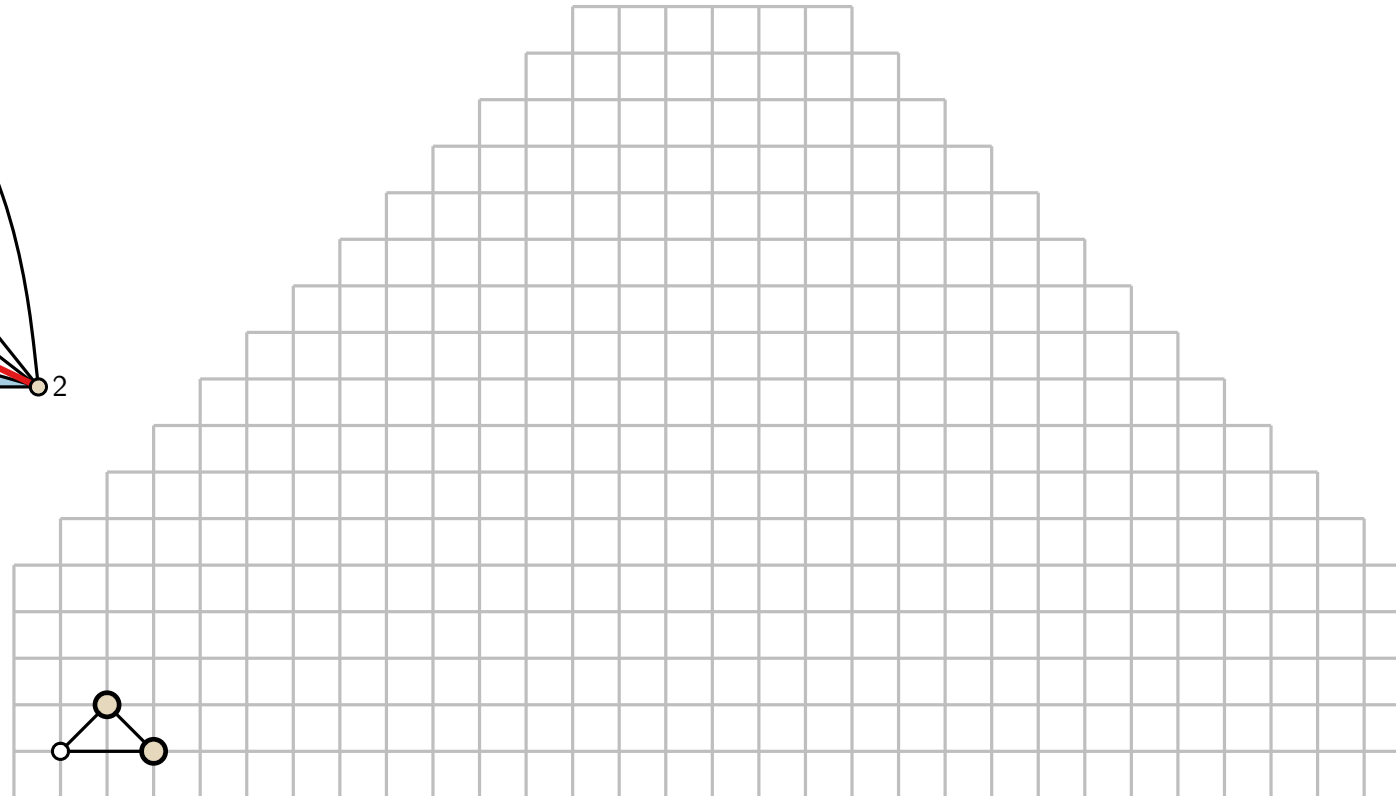
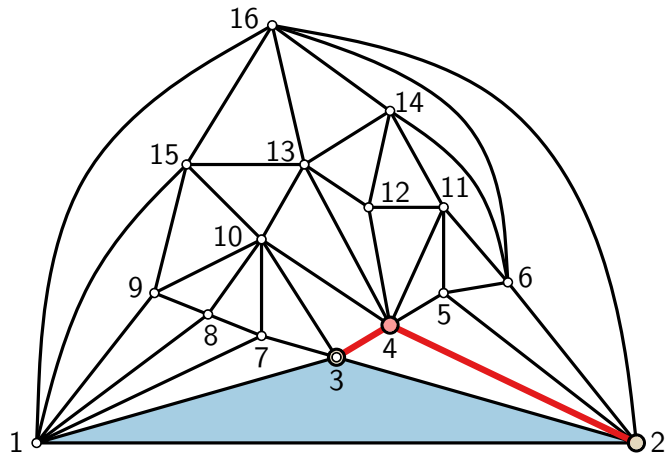
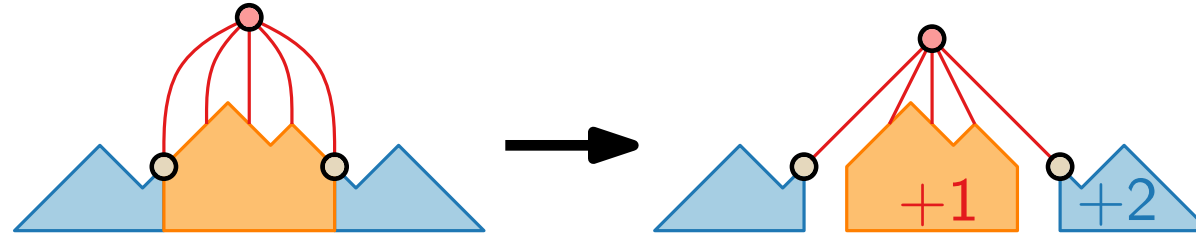
# Shift Method – Example



# Shift Method – Example

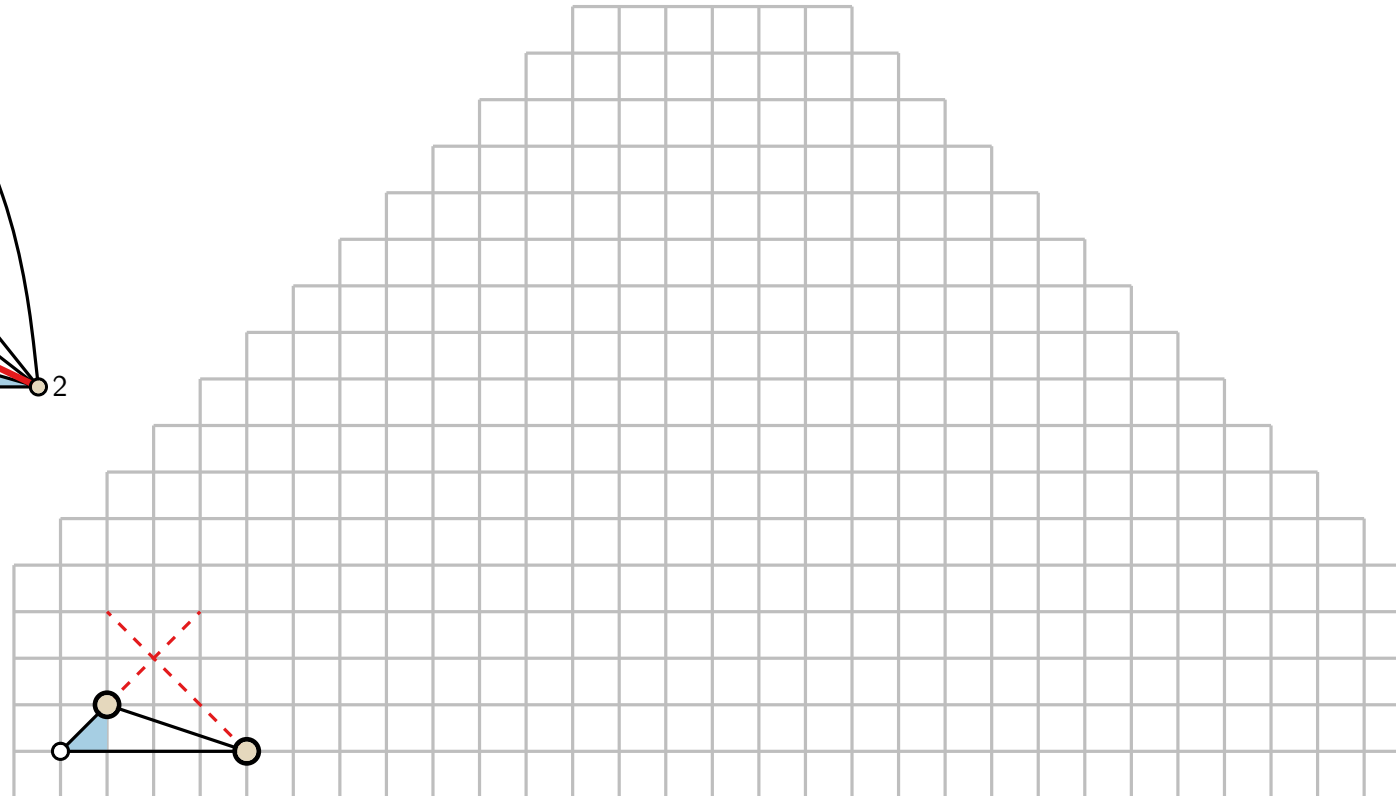
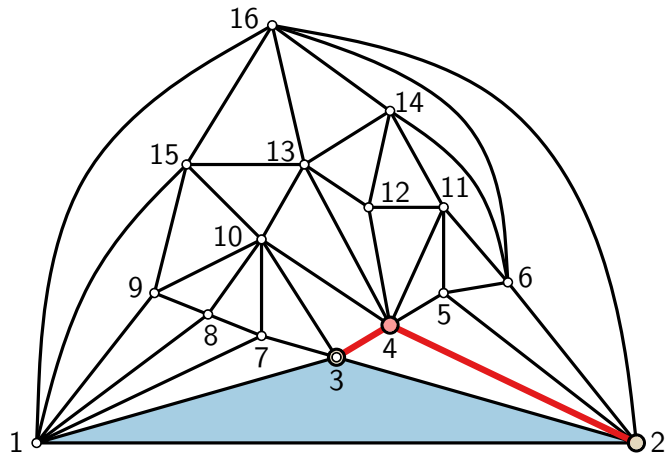
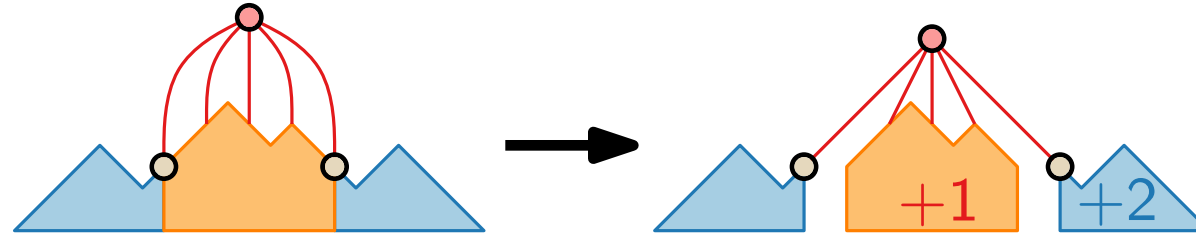


# Shift Method – Example

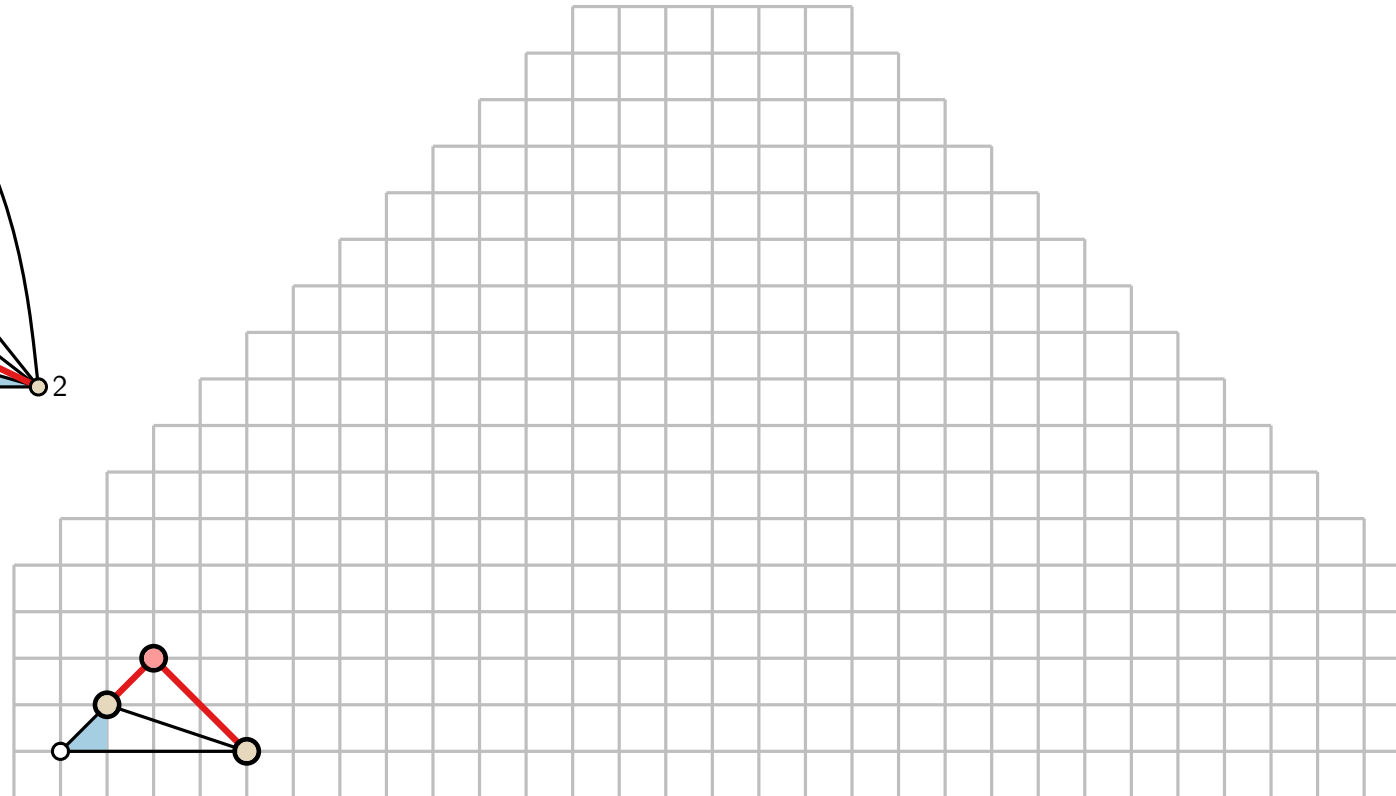
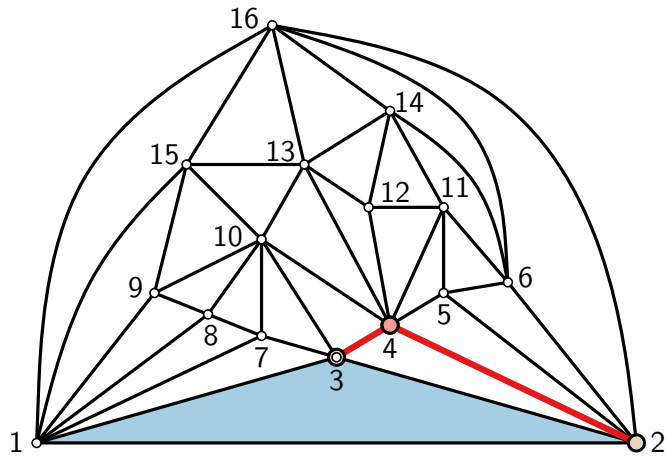
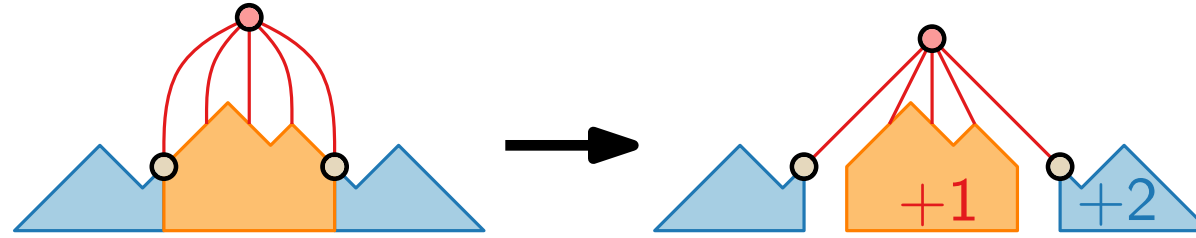




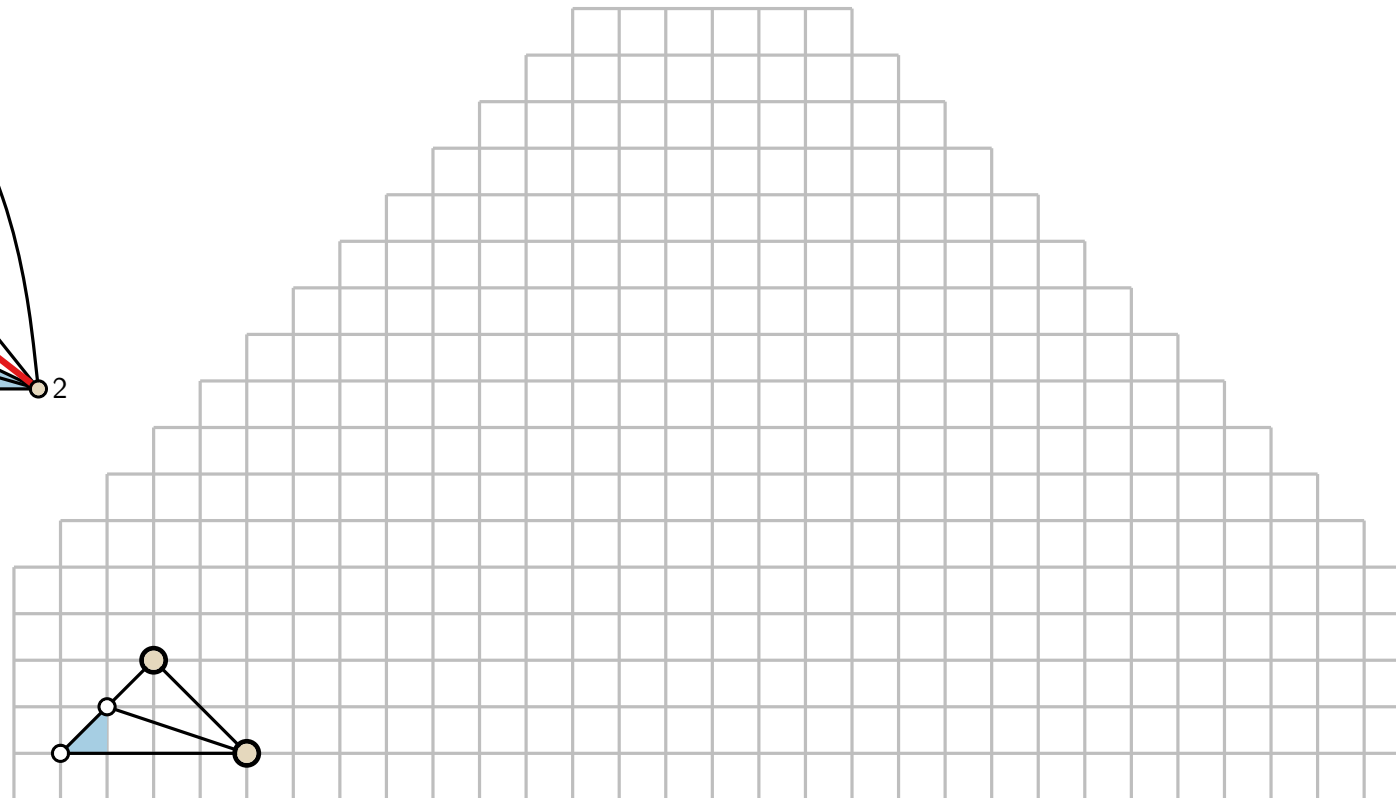
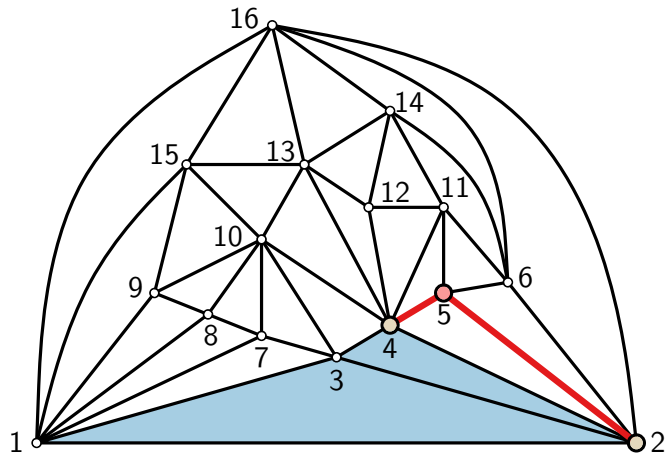
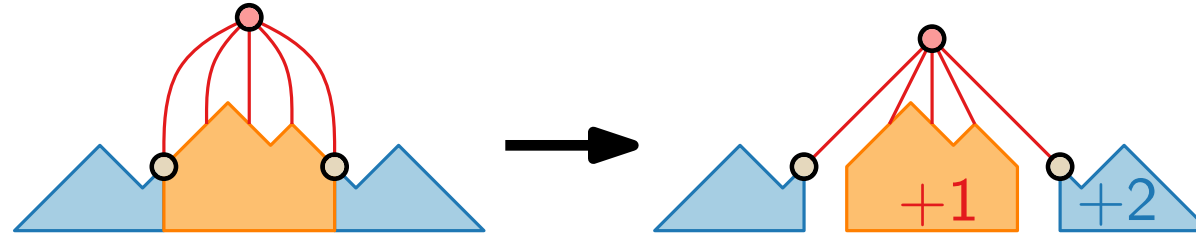
# Shift Method – Example



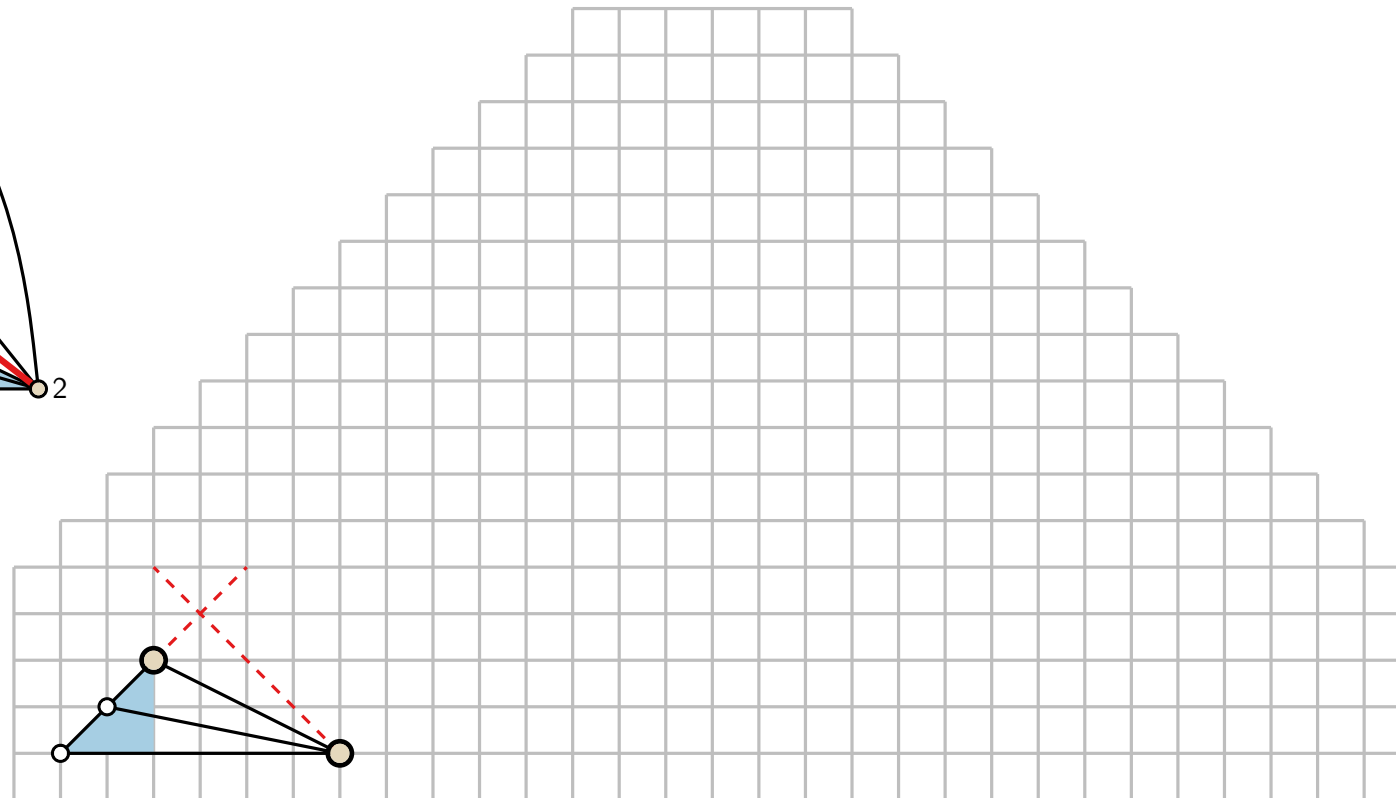
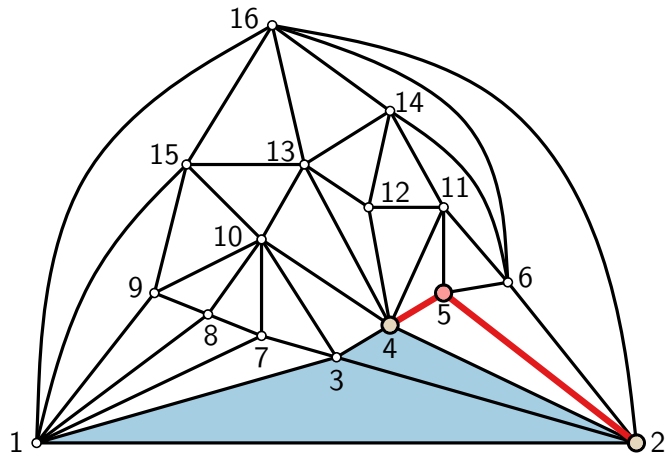
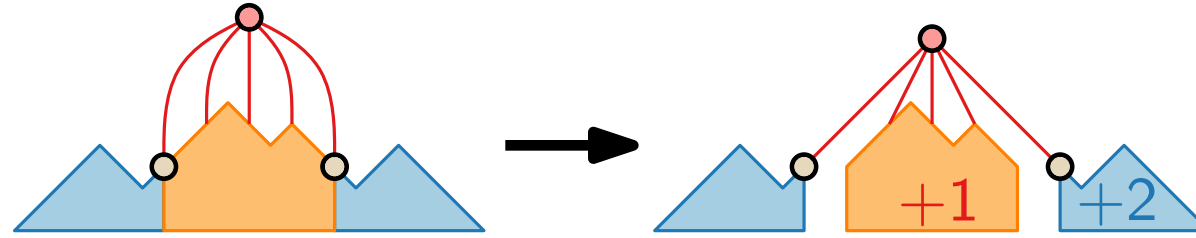
# Shift Method – Example



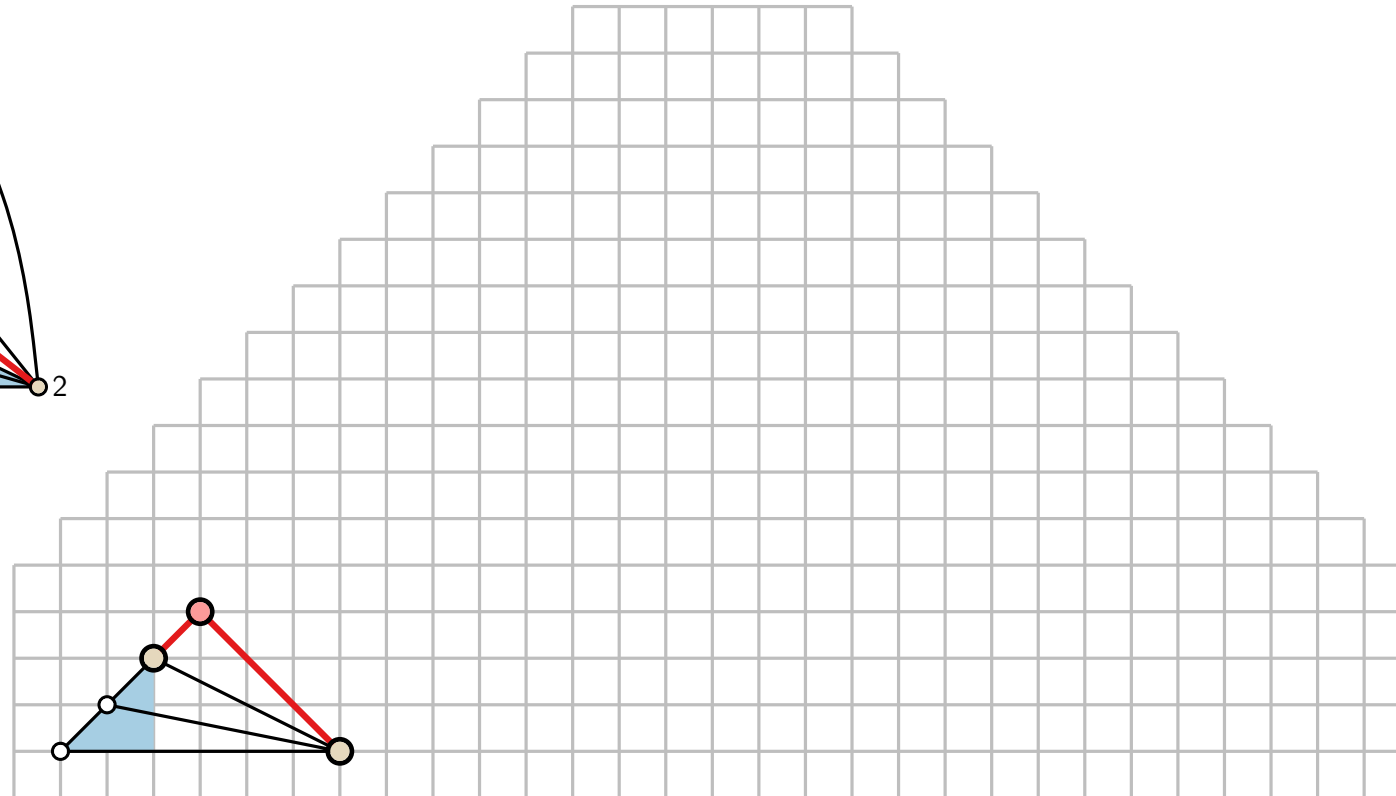
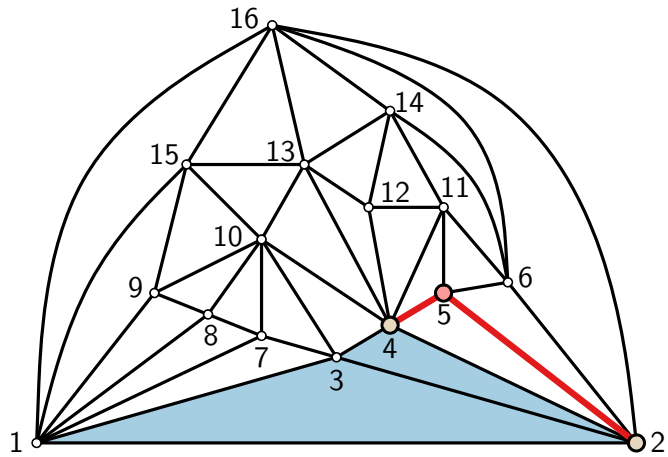
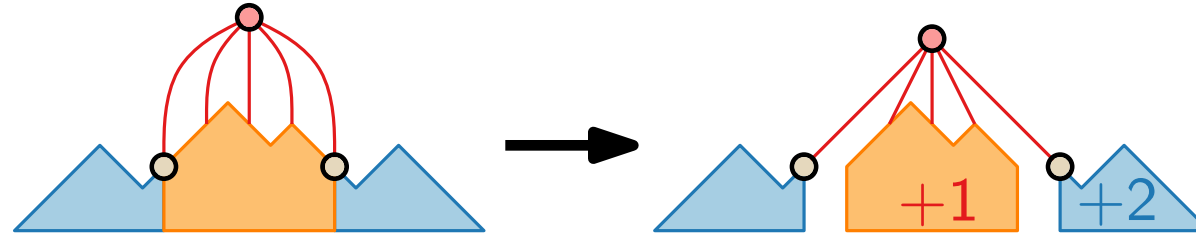
# Shift Method – Example



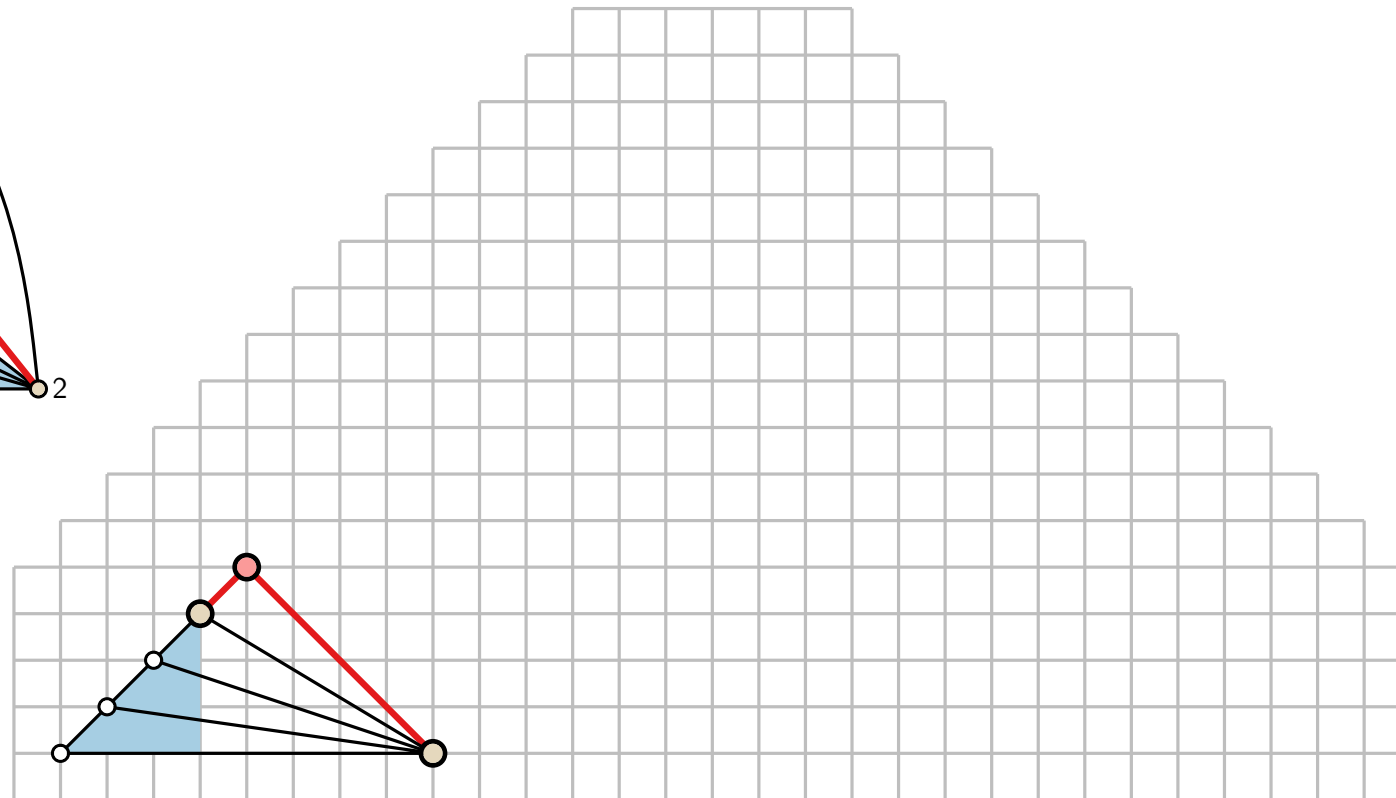
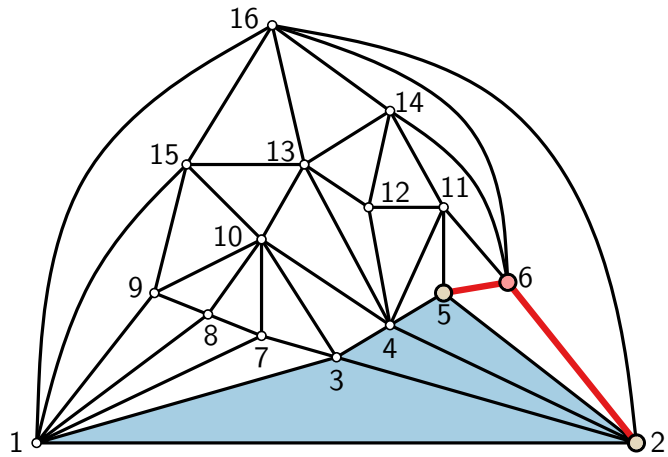
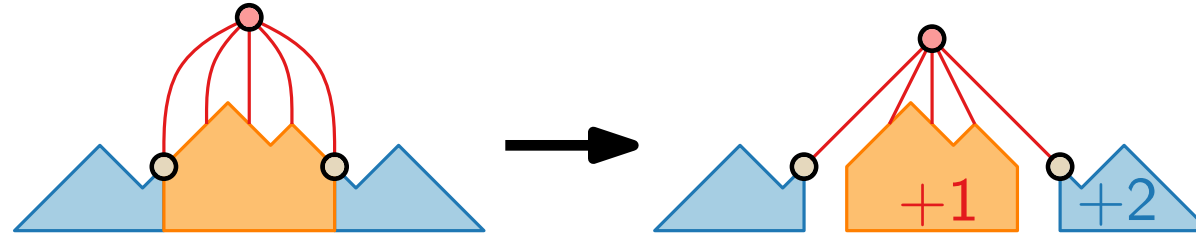
# Shift Method – Example



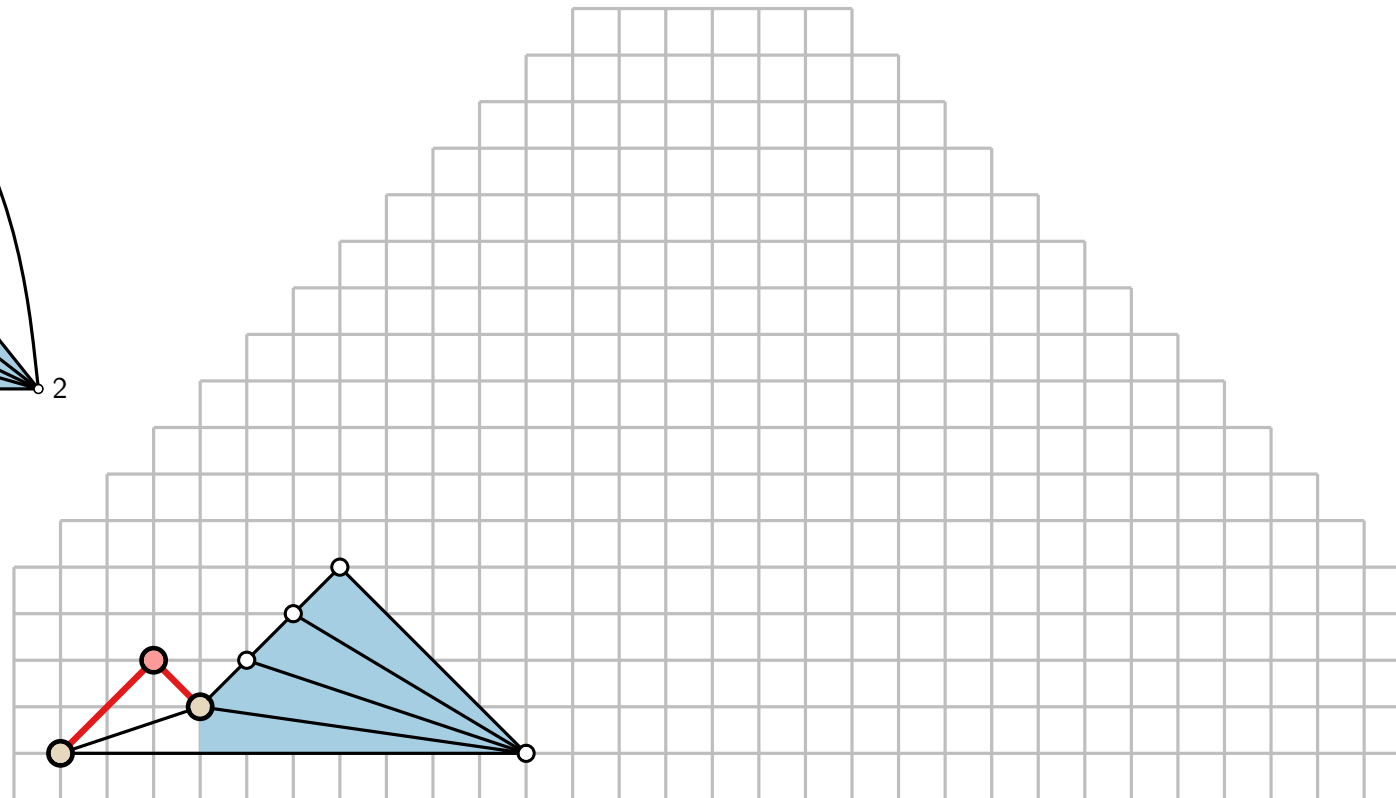
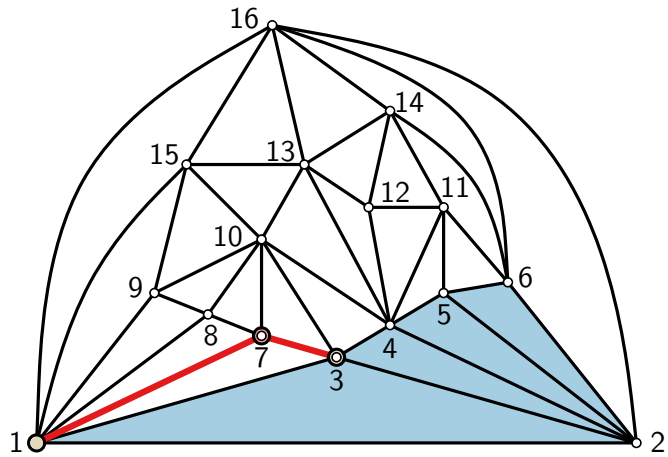
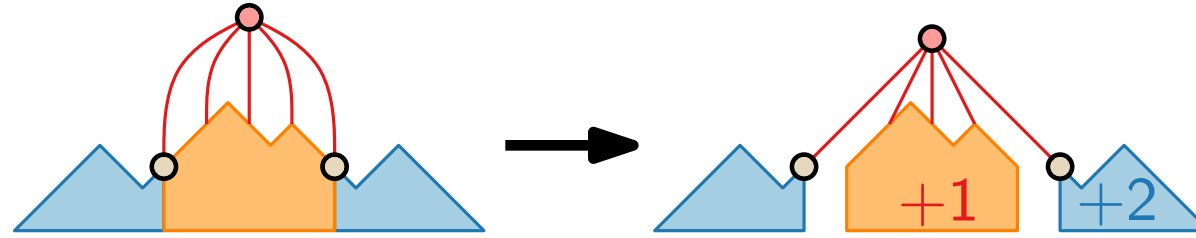
# Shift Method – Example



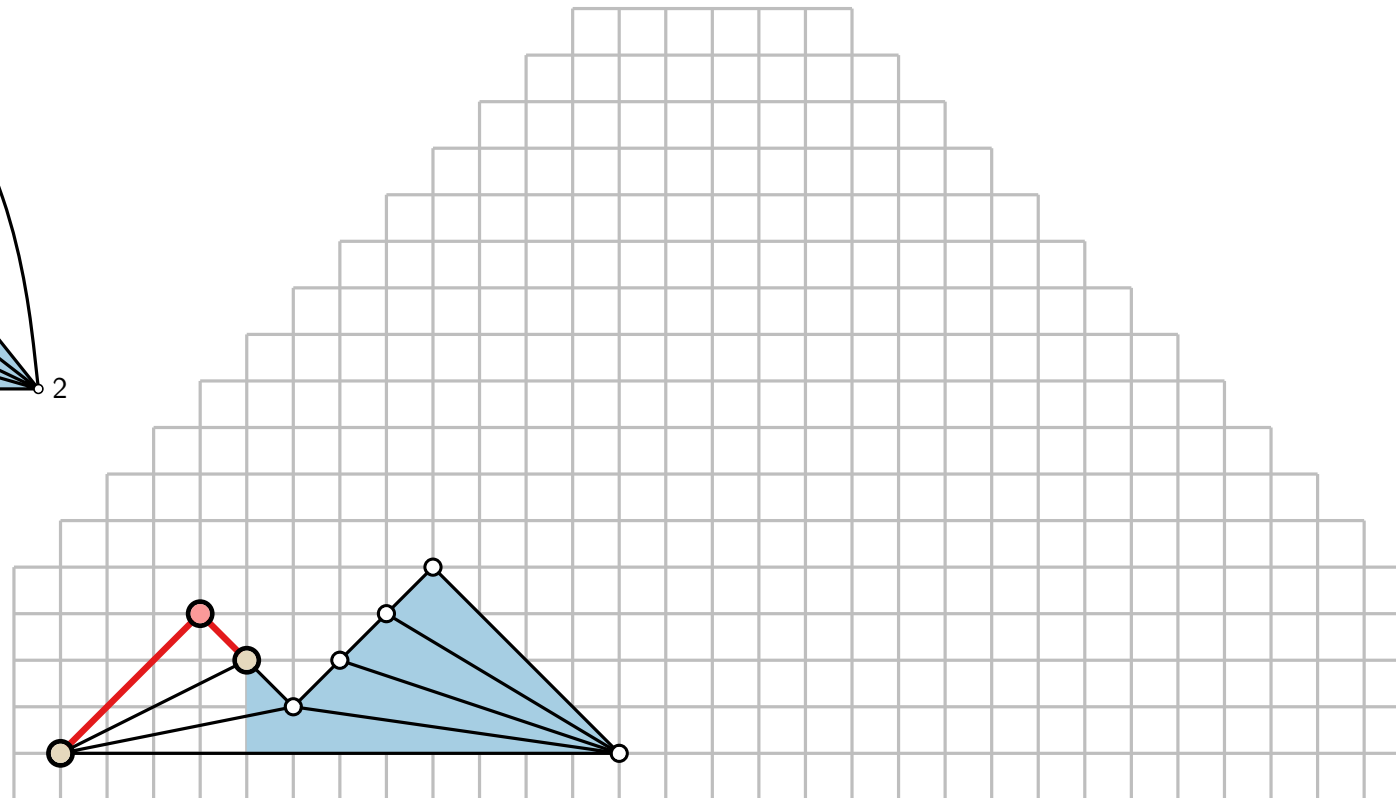
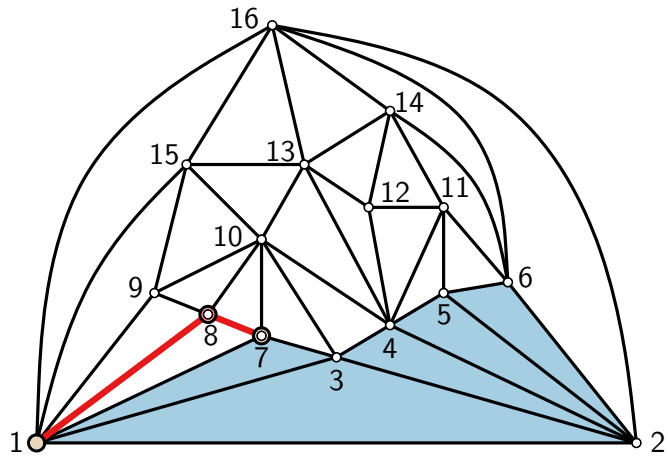
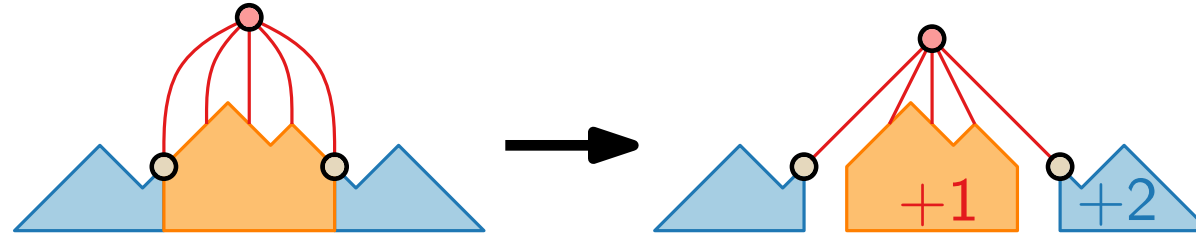
# Shift Method – Example



# Shift Method – Example

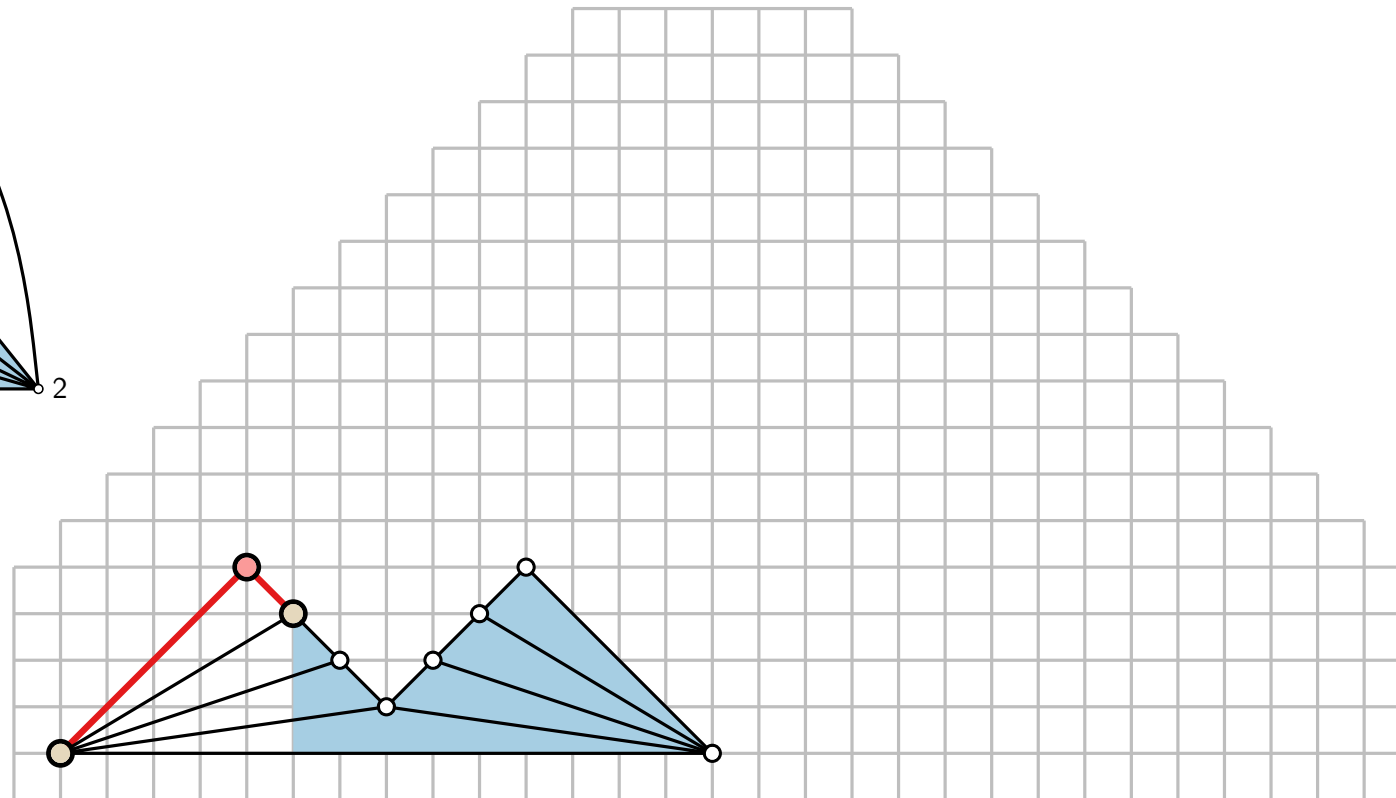
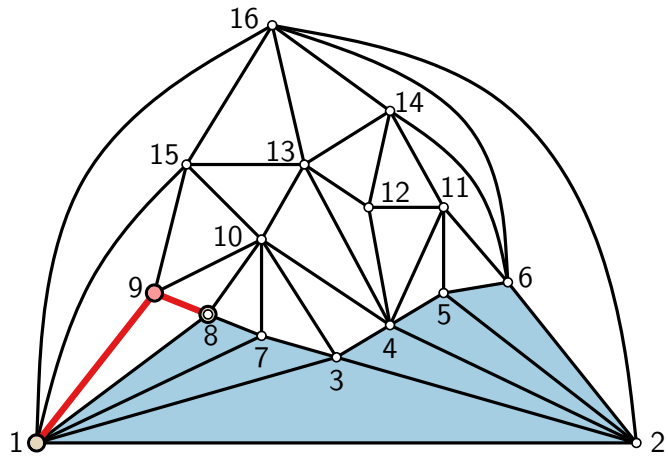
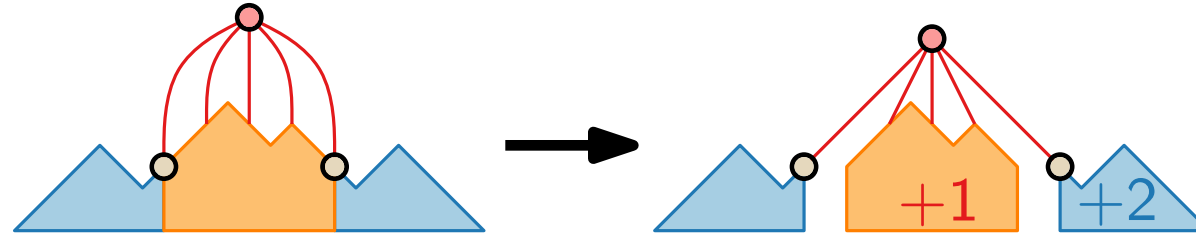


# Shift Method – Example

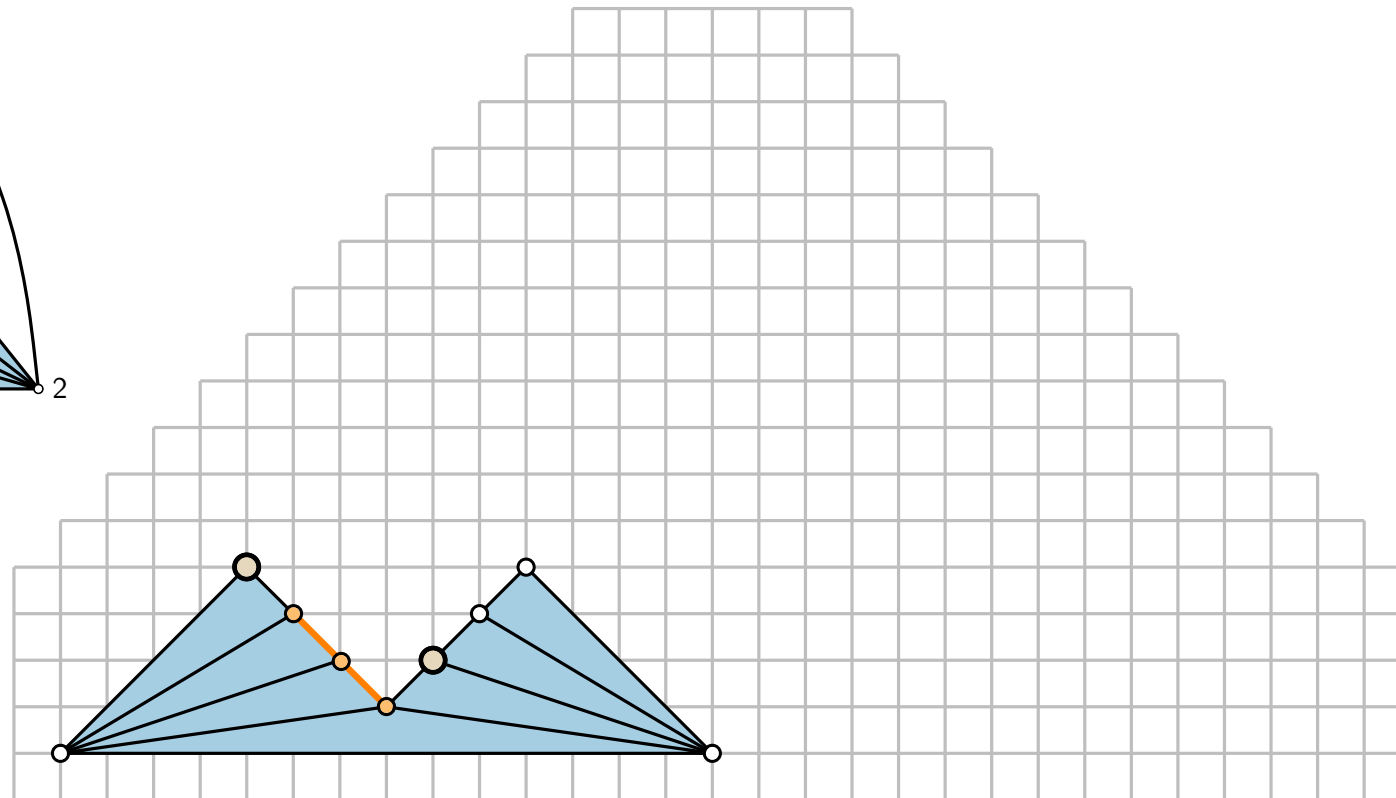
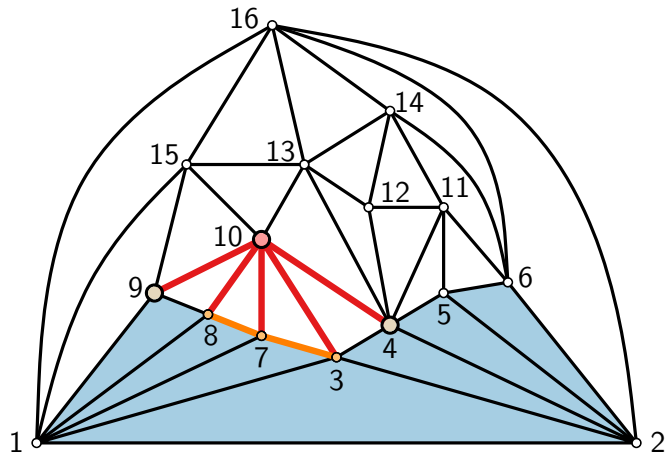
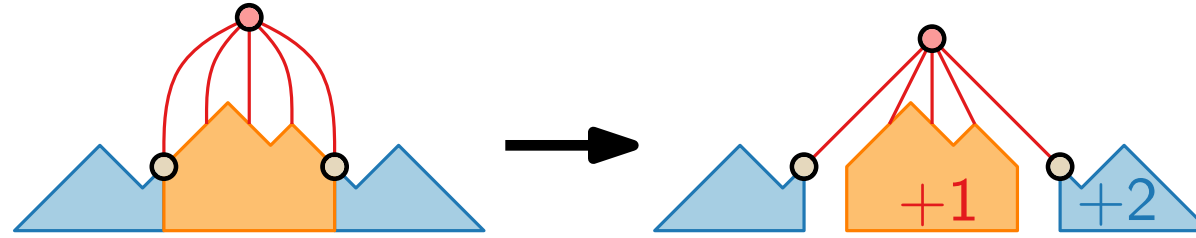




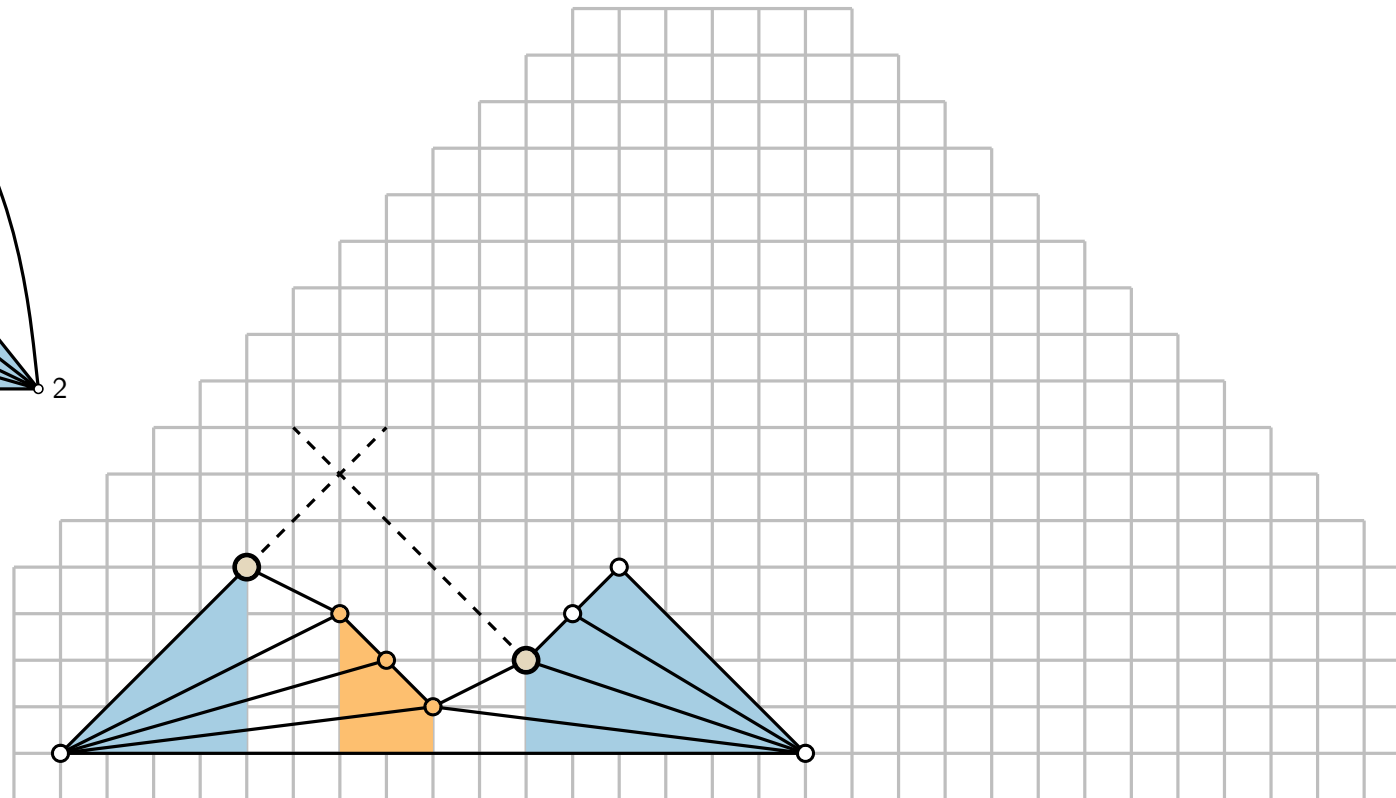
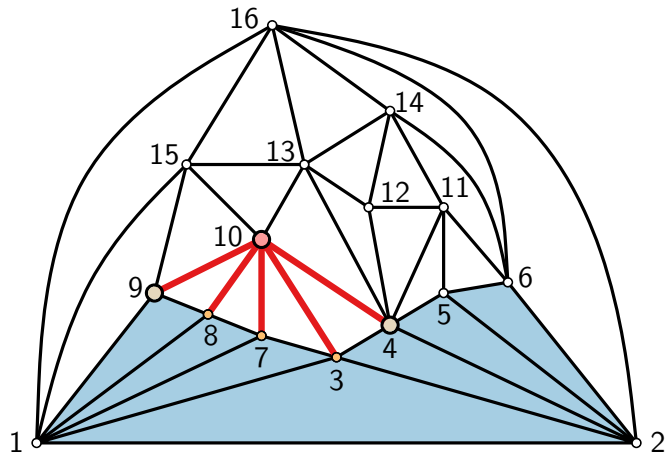
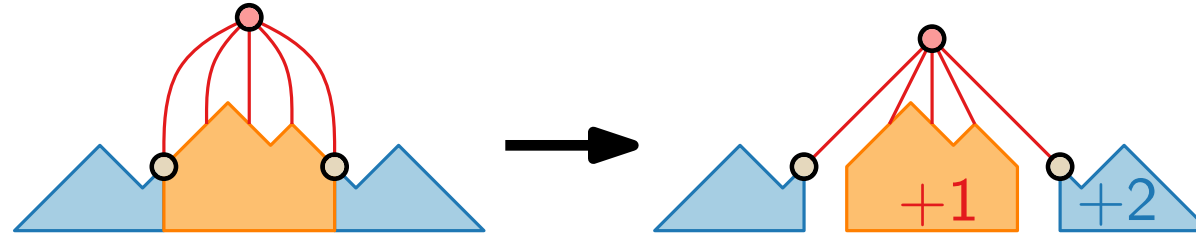
# Shift Method – Example



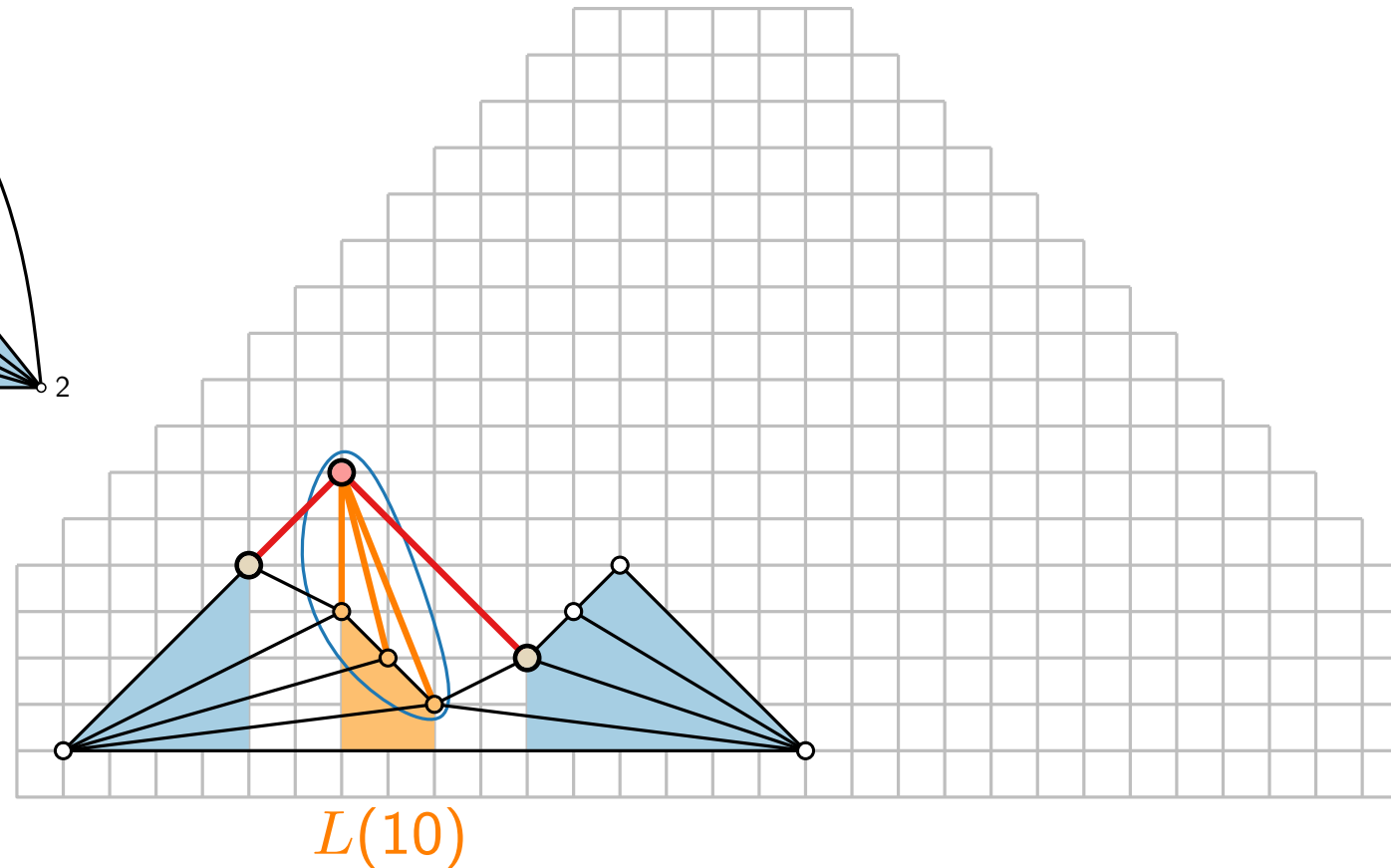
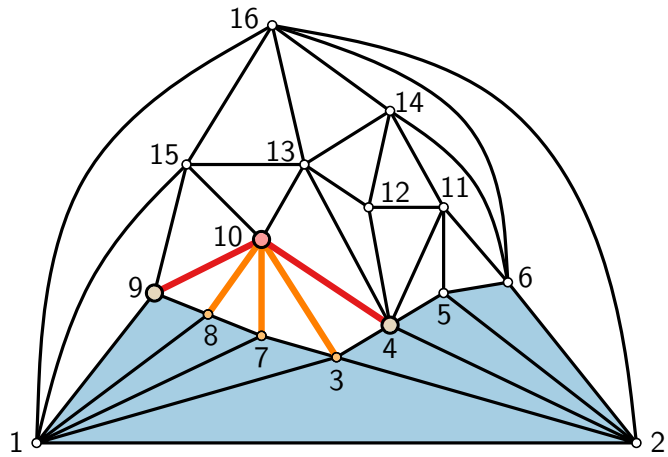
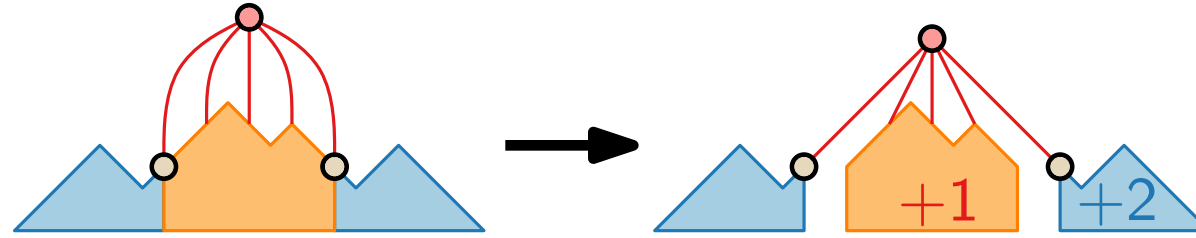
# Shift Method – Example



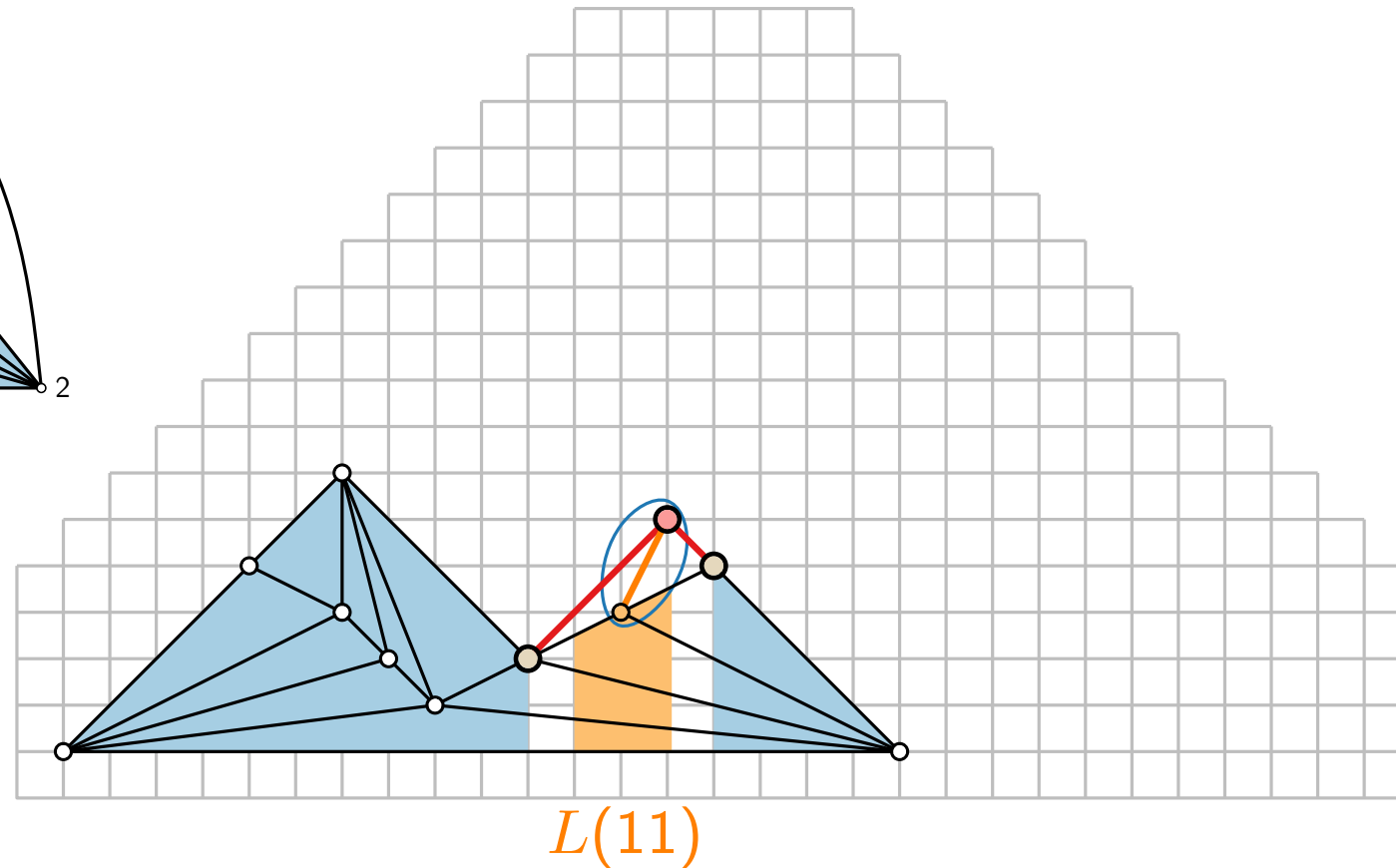
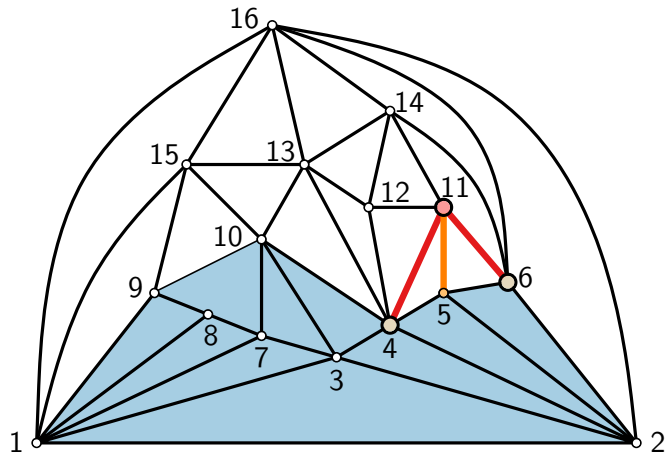
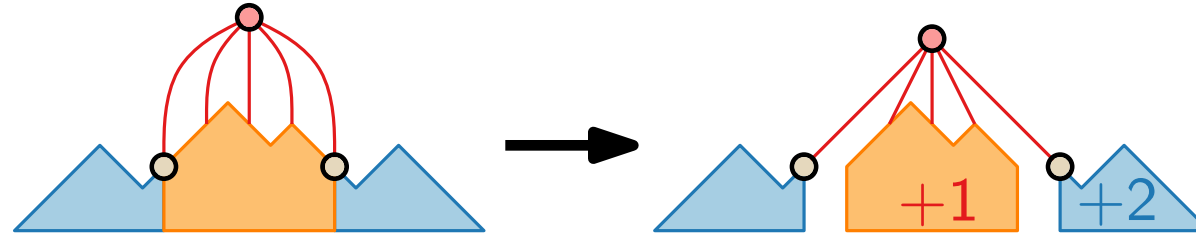
# Shift Method – Example



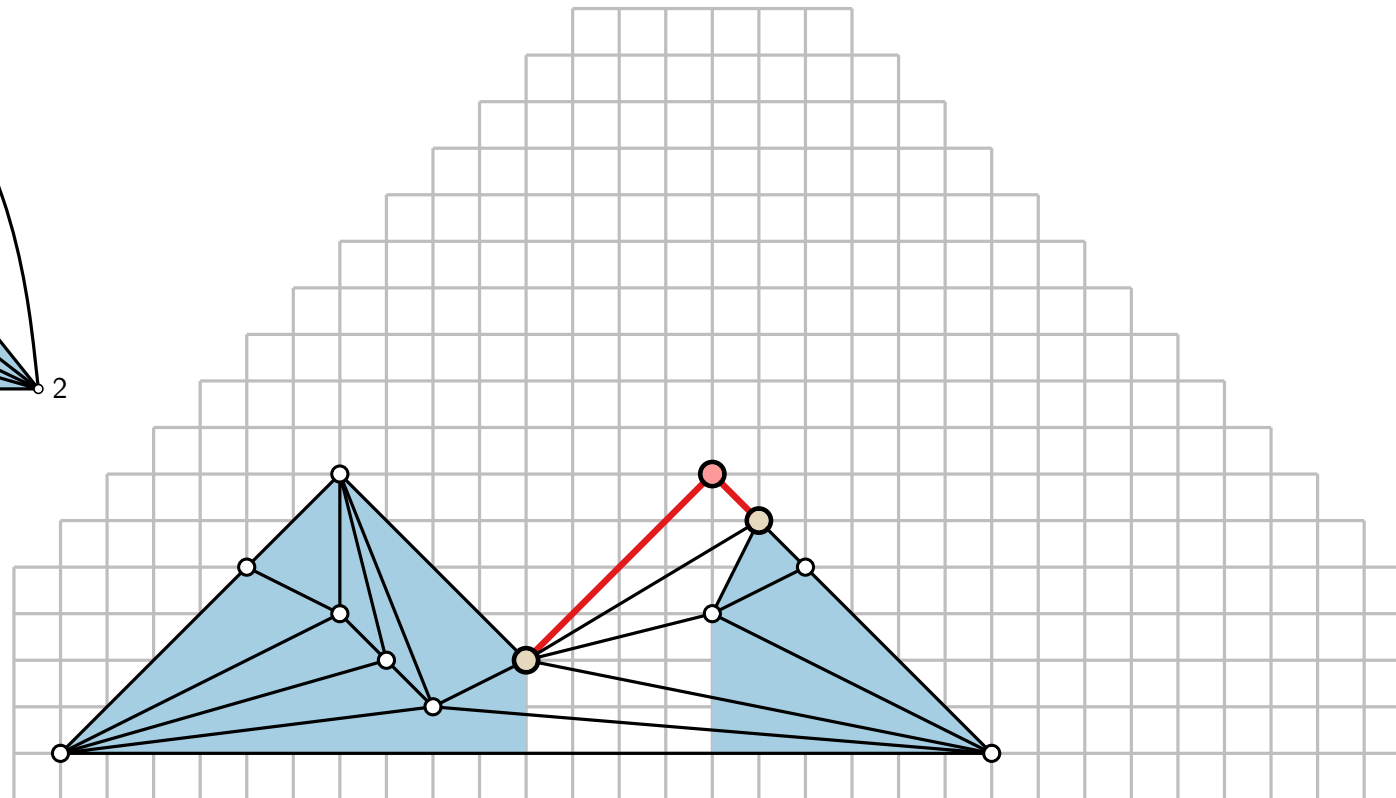
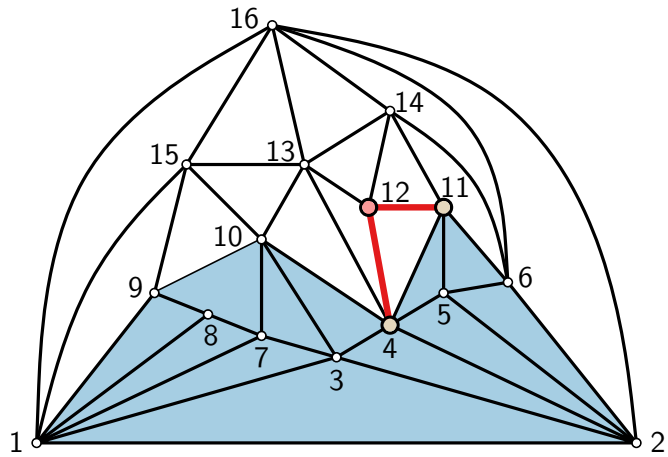
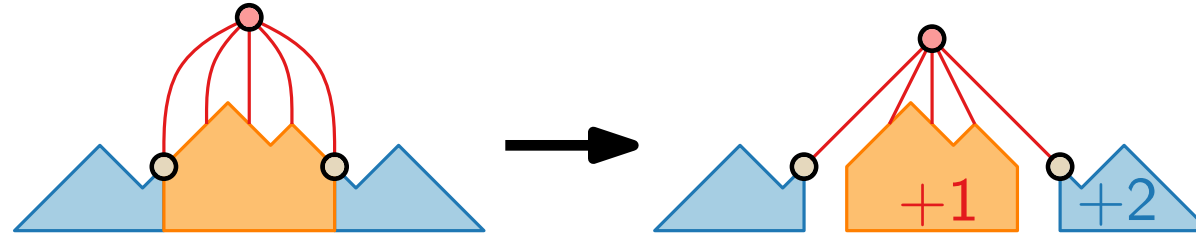
# Shift Method – Example



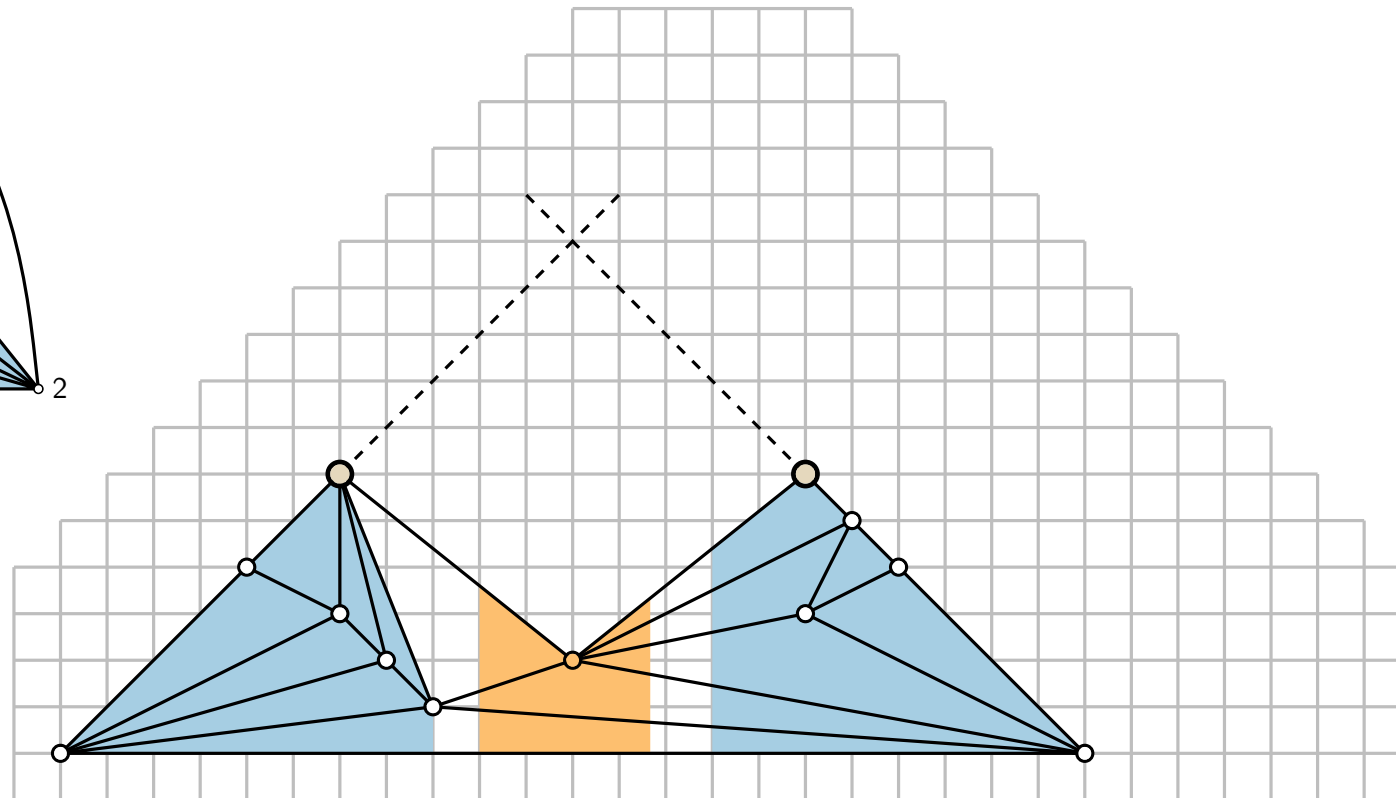
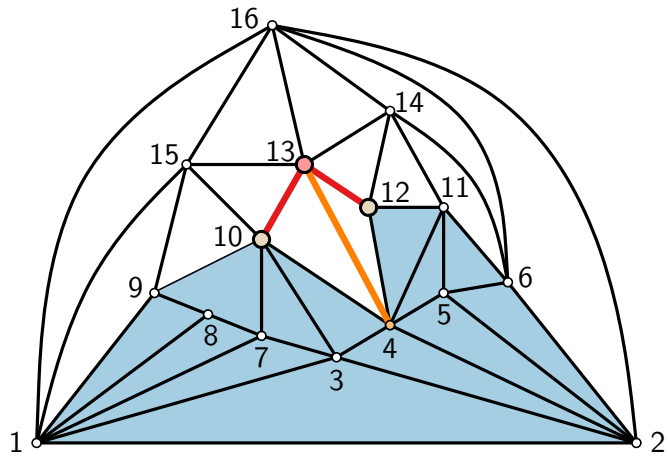
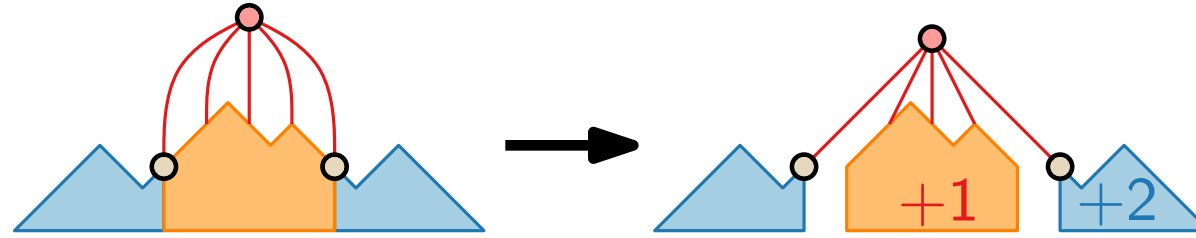
# Shift Method – Example



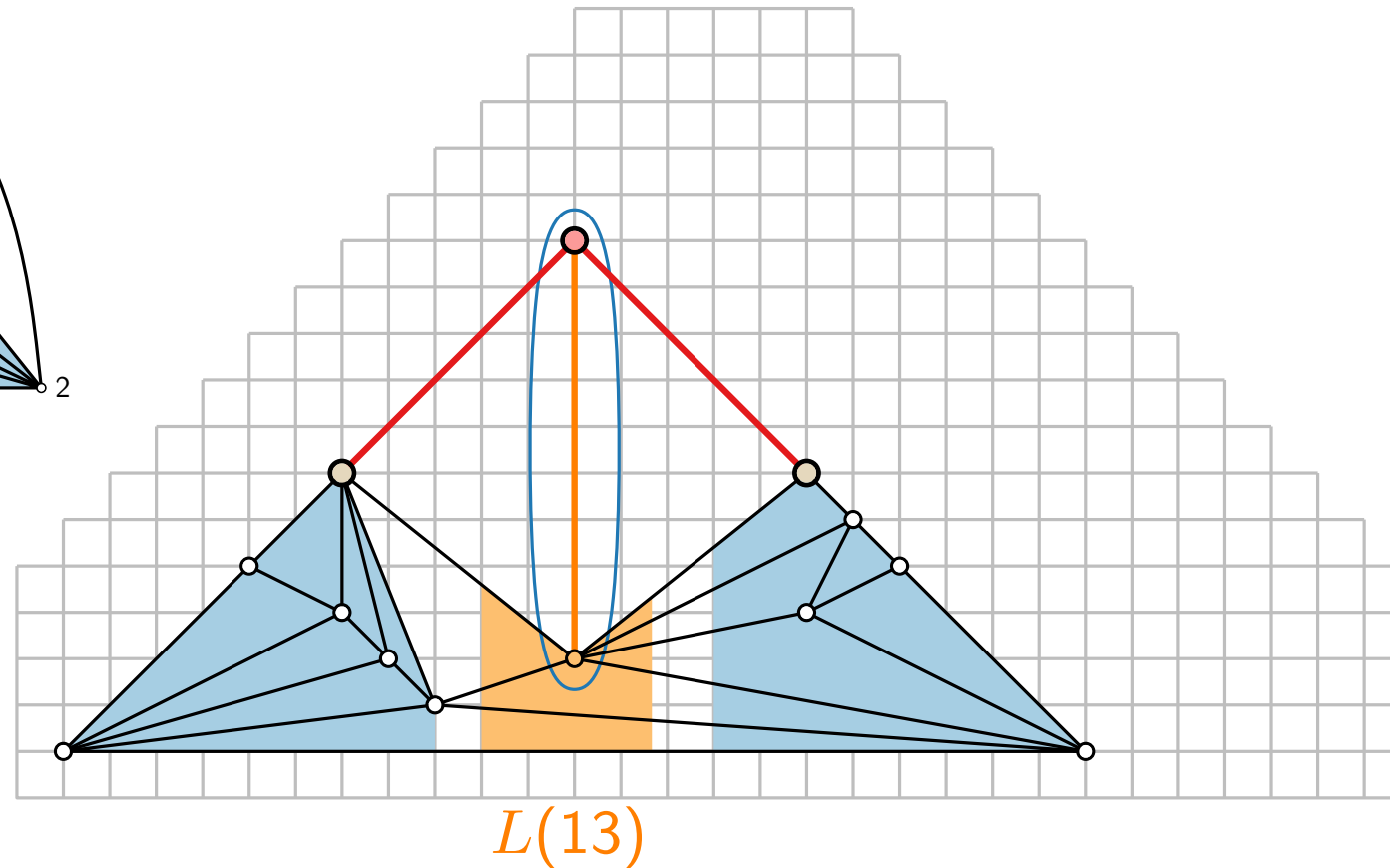
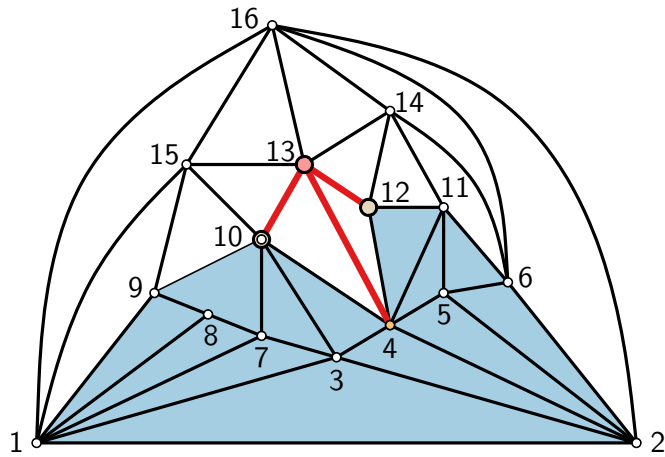
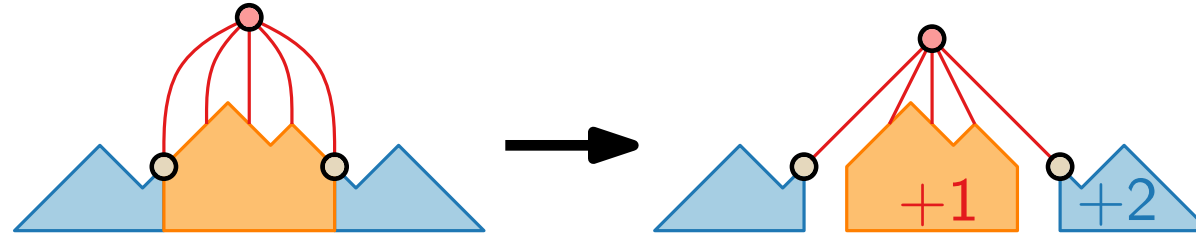
# Shift Method – Example



# Shift Method – Example

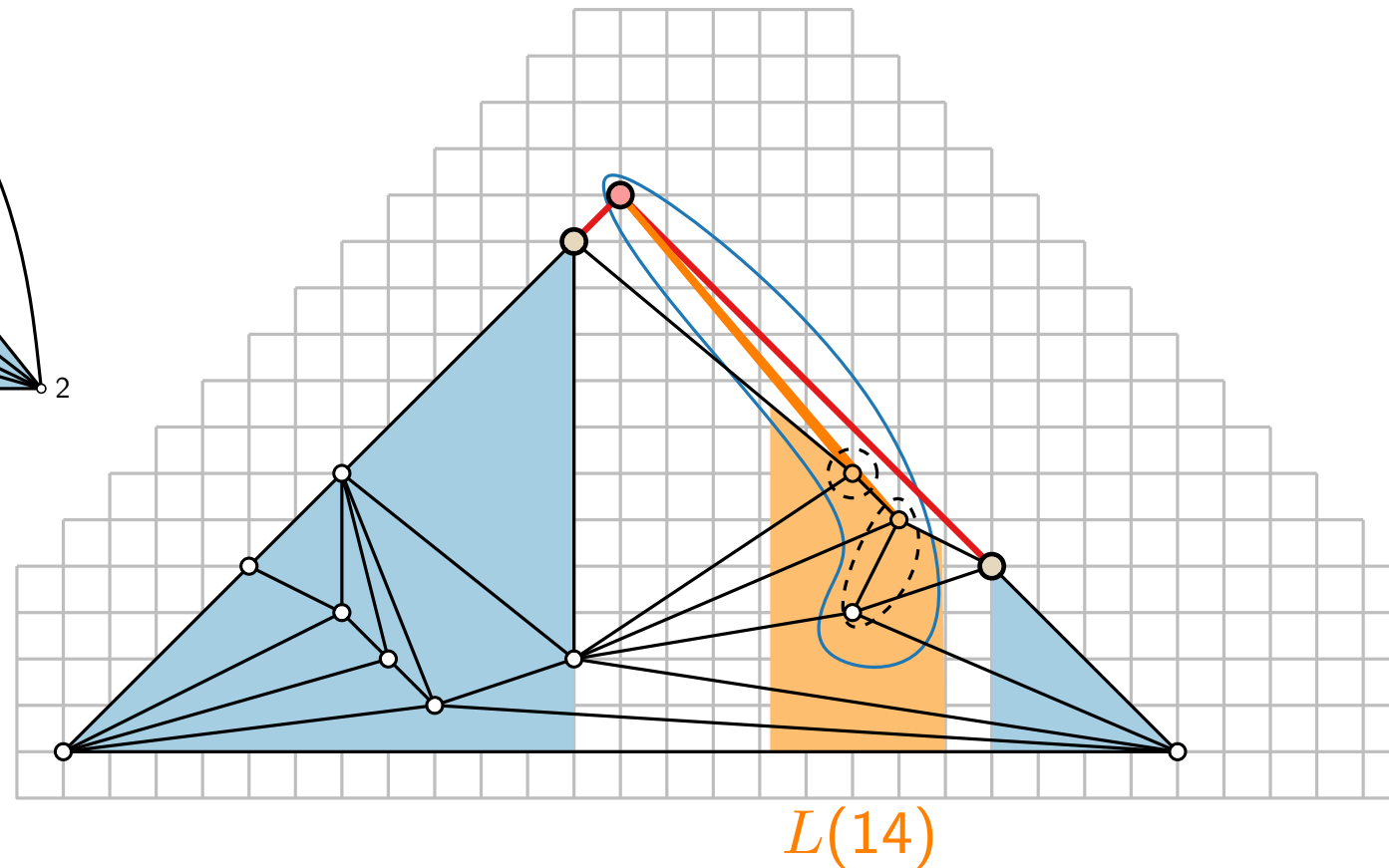
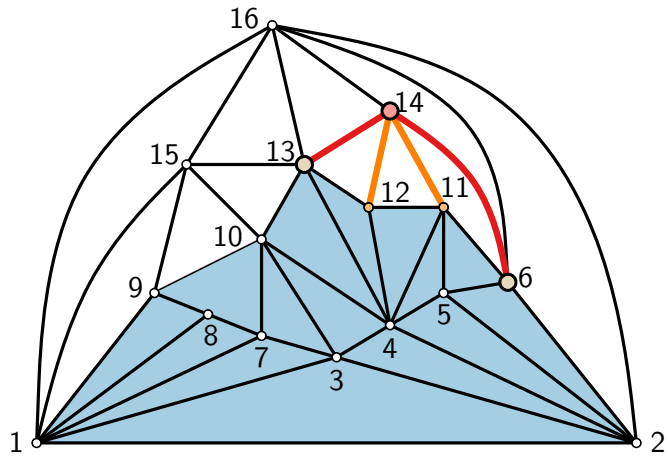
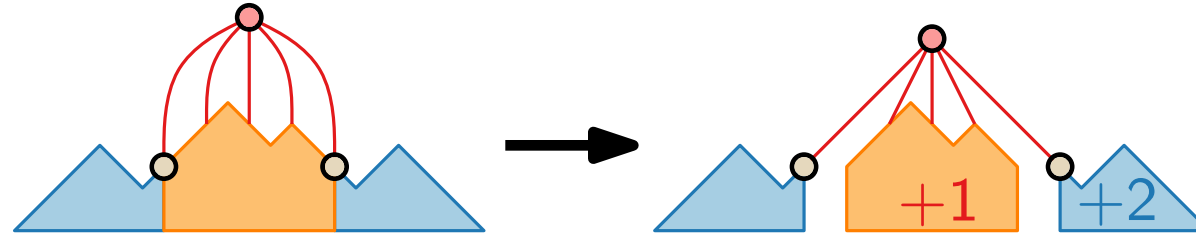


# Shift Method – Example

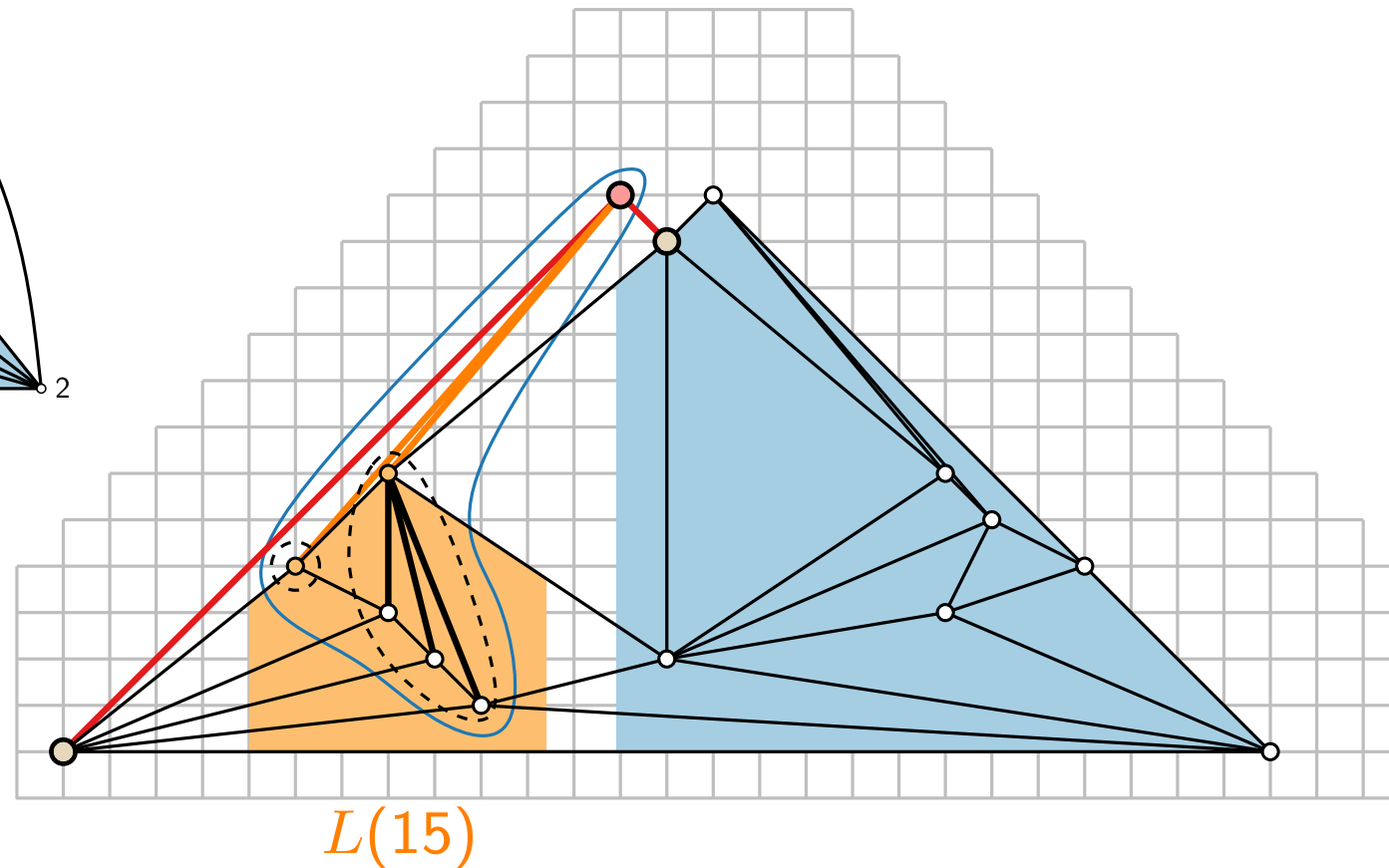
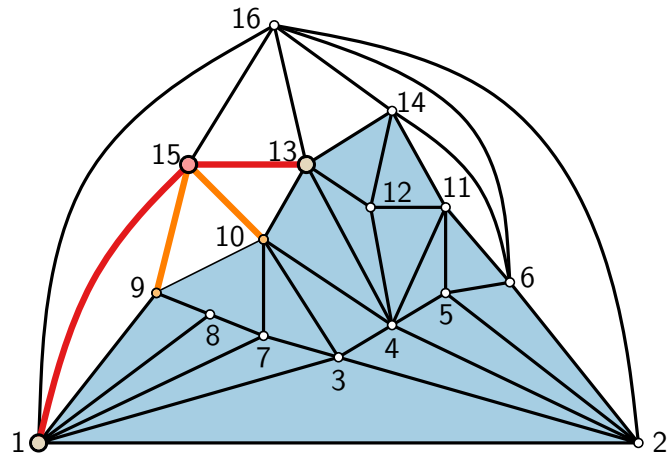
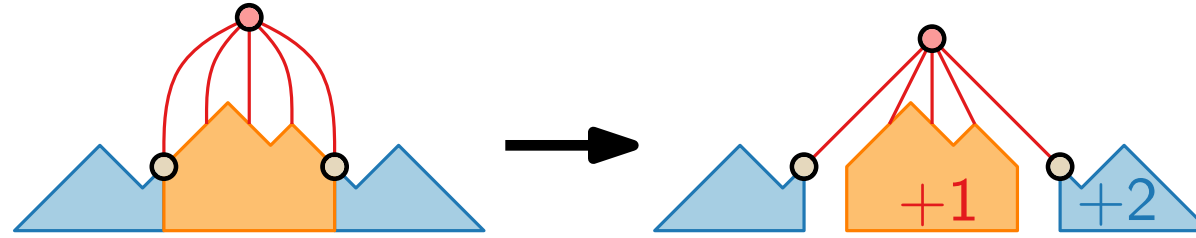




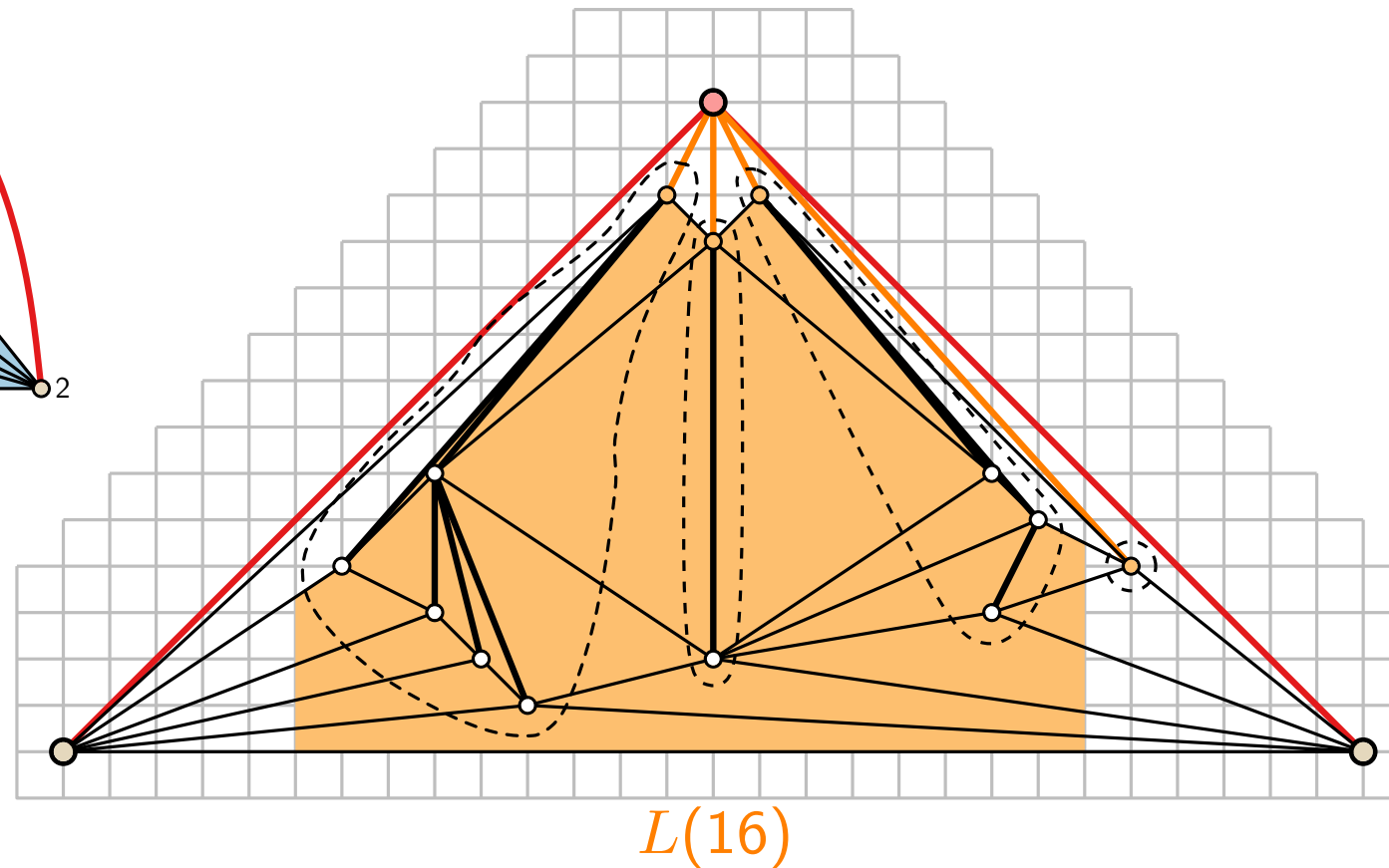
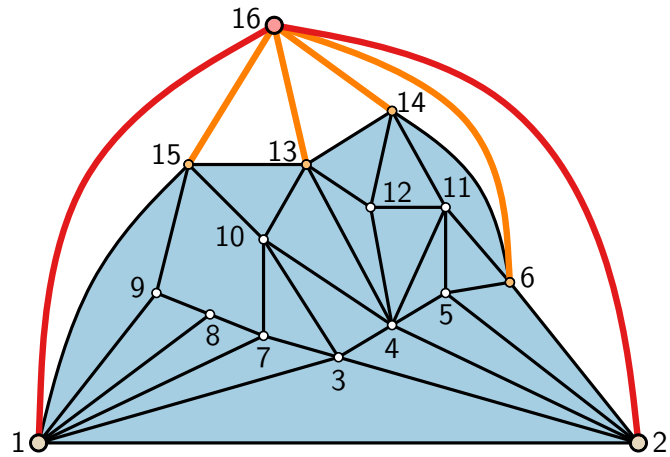
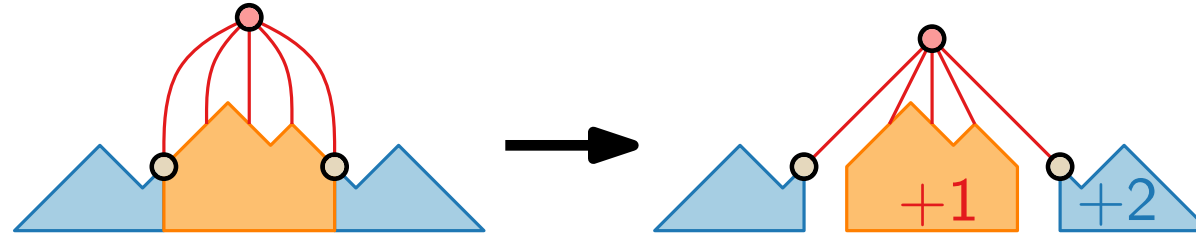
# Shift Method – Example



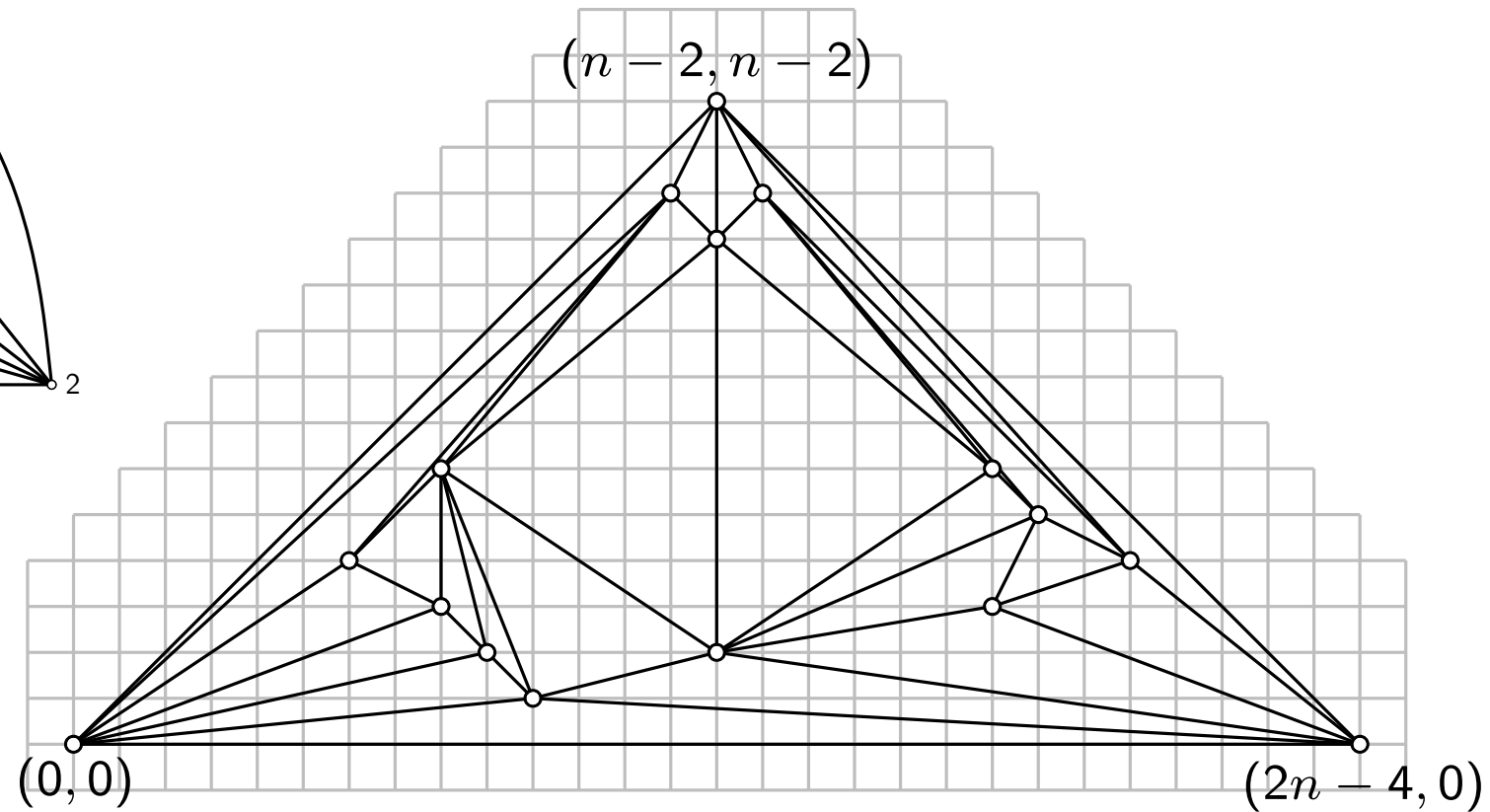
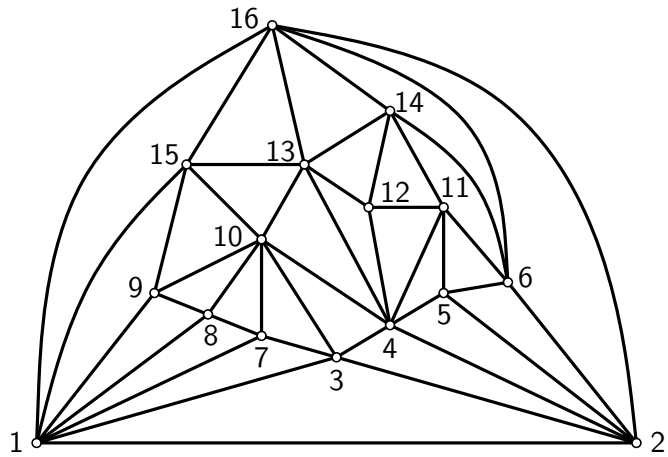
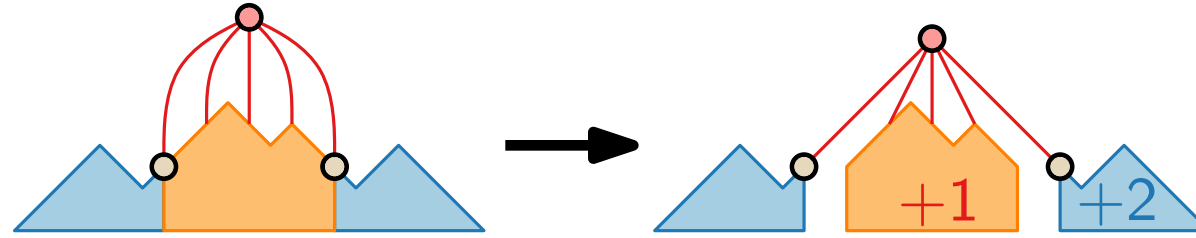
# Shift Method – Example



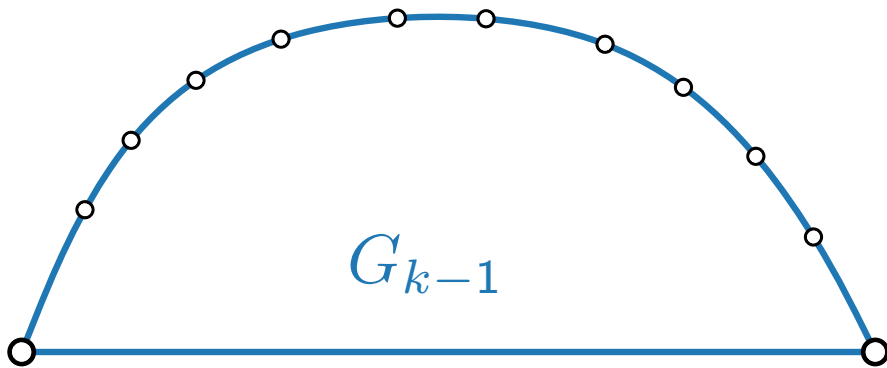
# Shift Method – Example



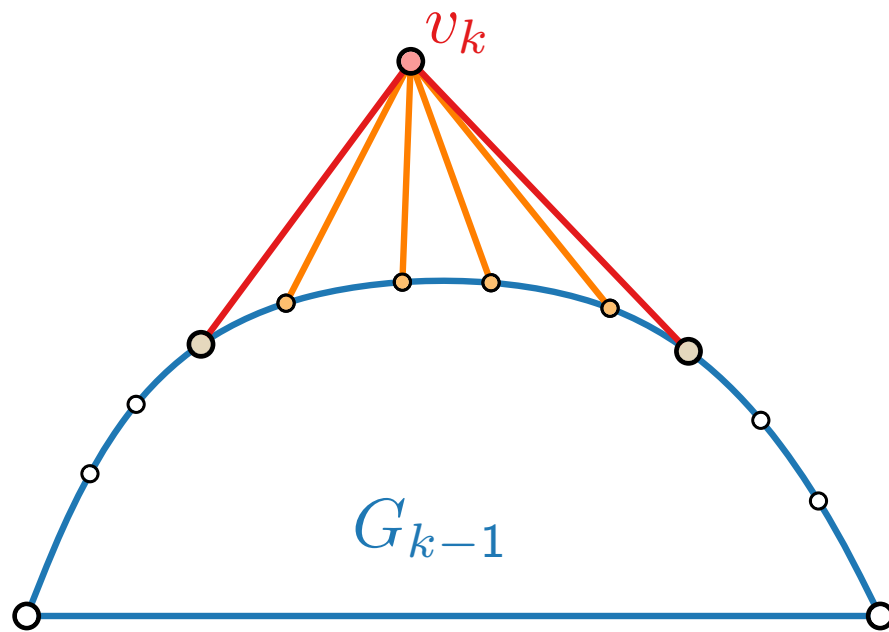
# Shift Method – Example



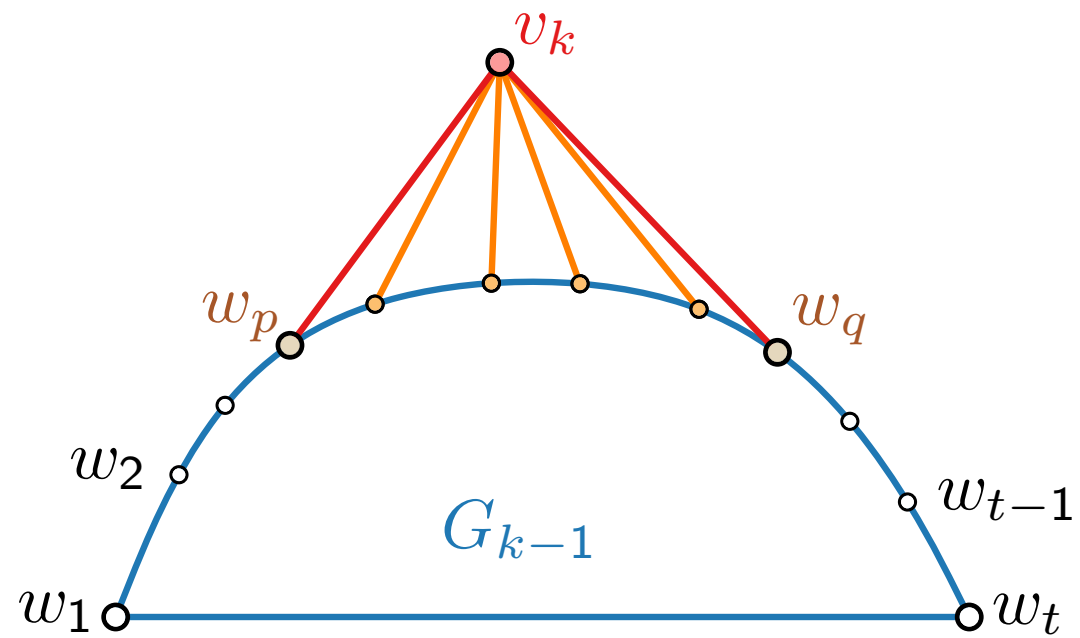
# Shift Method – Planarity



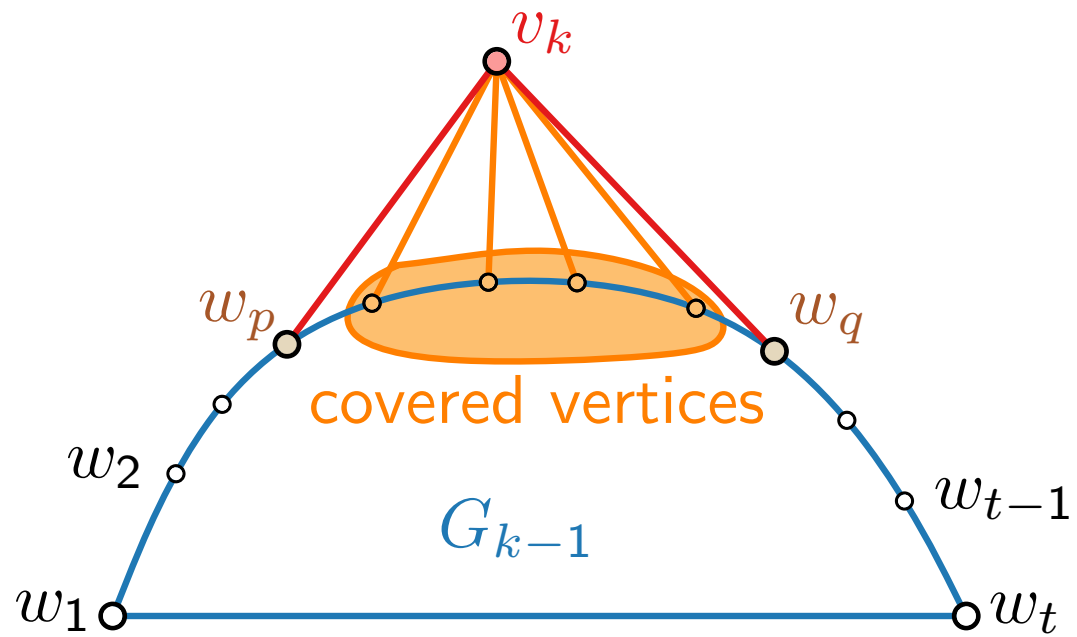
# Shift Method – Planarity



# Shift Method – Planarity



# Shift Method – Planarity

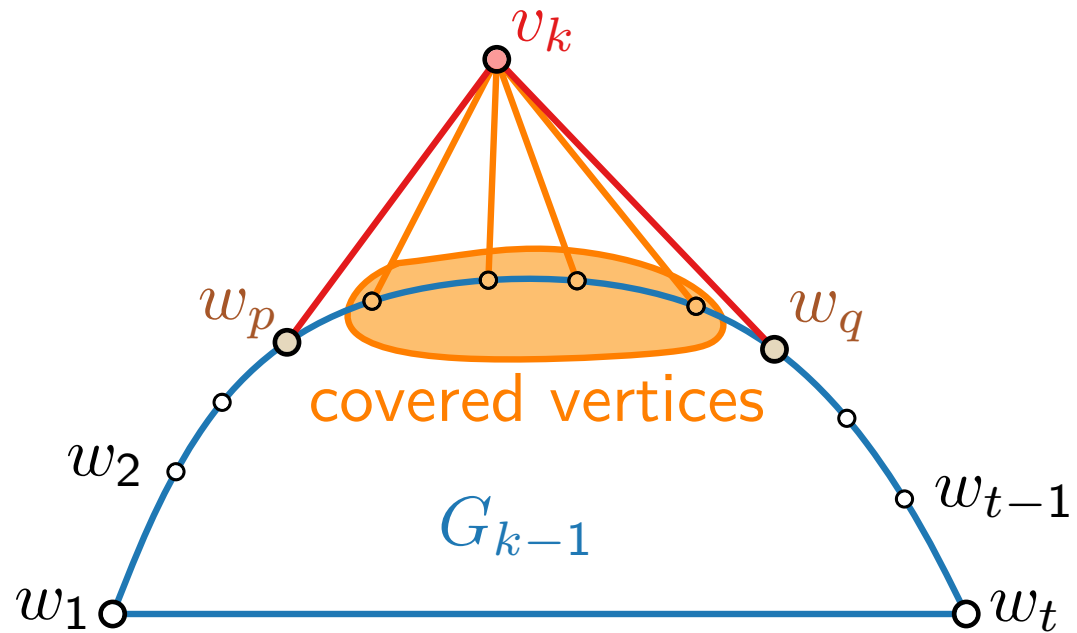




# Shift Method – Planarity

## Observations.

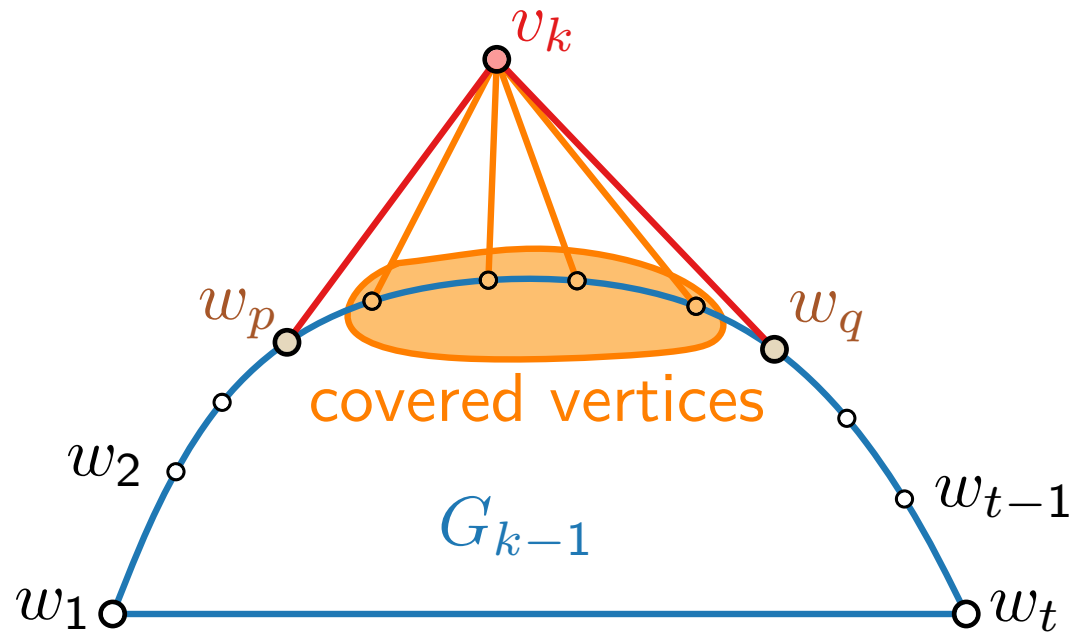
- Each internal vertex is **covered** exactly once.



# Shift Method – Planarity

## Observations.

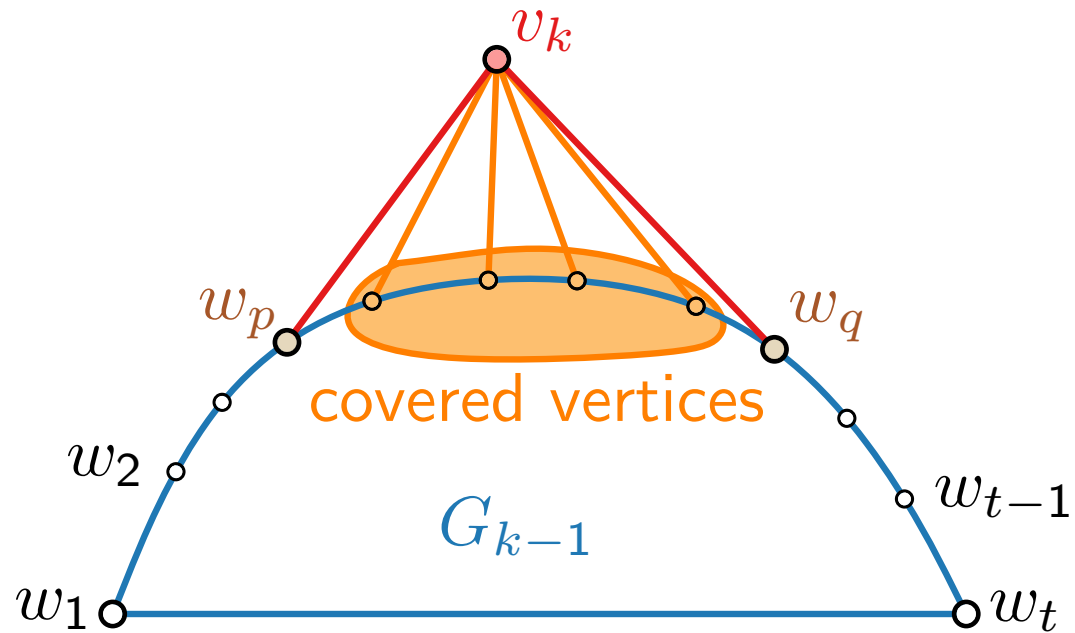
- Each internal vertex is **covered** exactly once.
- Covering relation defines a tree in  $G$



# Shift Method – Planarity

## Observations.

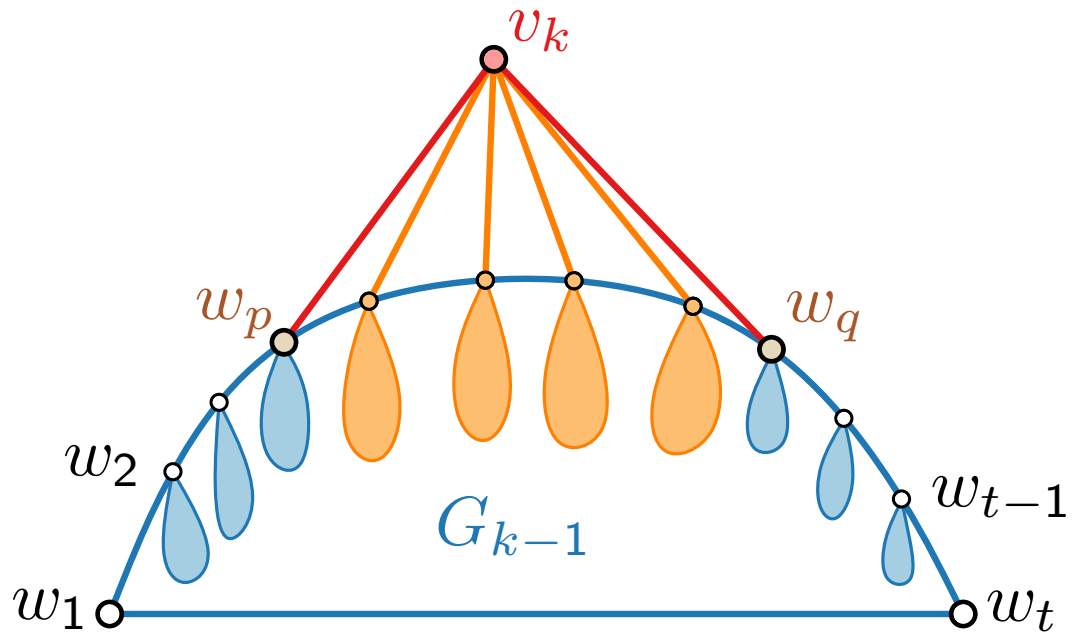
- Each internal vertex is **covered** exactly once.
- Covering relation defines a tree in  $G$
- and a forest in  $G_i$ ,  $1 \leq i \leq n - 1$ .



# Shift Method – Planarity

## Observations.

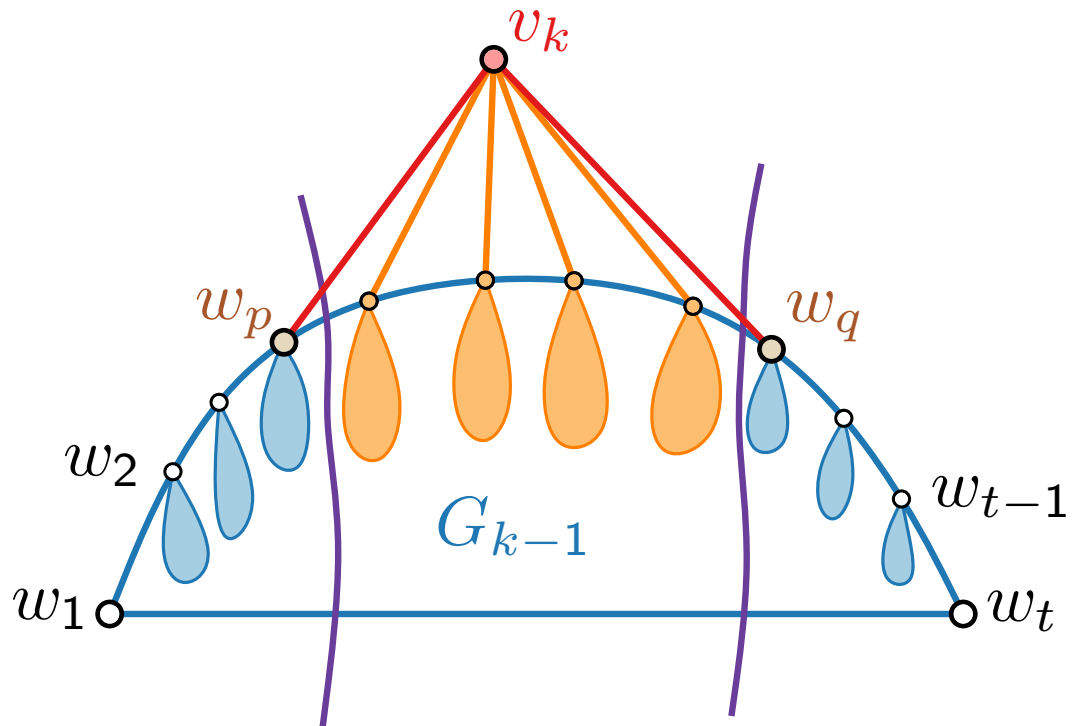
- Each internal vertex is **covered** exactly once.
- Covering relation defines a tree in  $G$
- and a forest in  $G_i$ ,  $1 \leq i \leq n - 1$ .



# Shift Method – Planarity

## Observations.

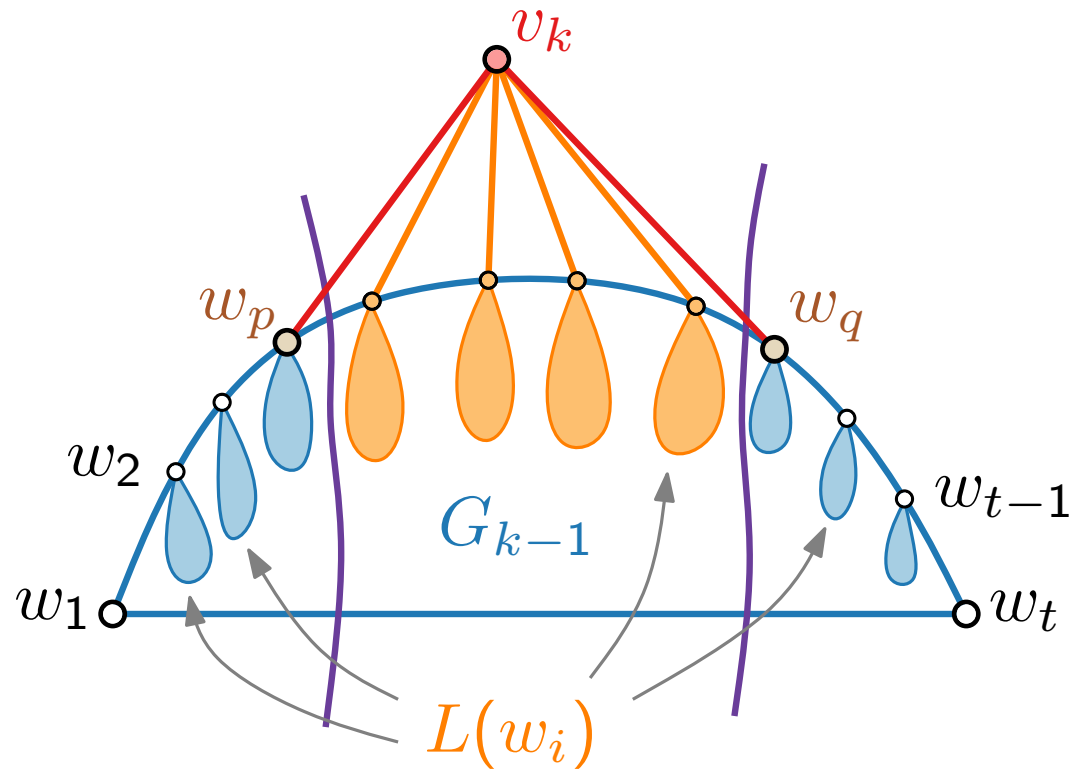
- Each internal vertex is **covered** exactly once.
- Covering relation defines a tree in  $G$
- and a forest in  $G_i$ ,  $1 \leq i \leq n - 1$ .



# Shift Method – Planarity

## Observations.

- Each internal vertex is **covered** exactly once.
- Covering relation defines a tree in  $G$
- and a forest in  $G_i$ ,  $1 \leq i \leq n - 1$ .



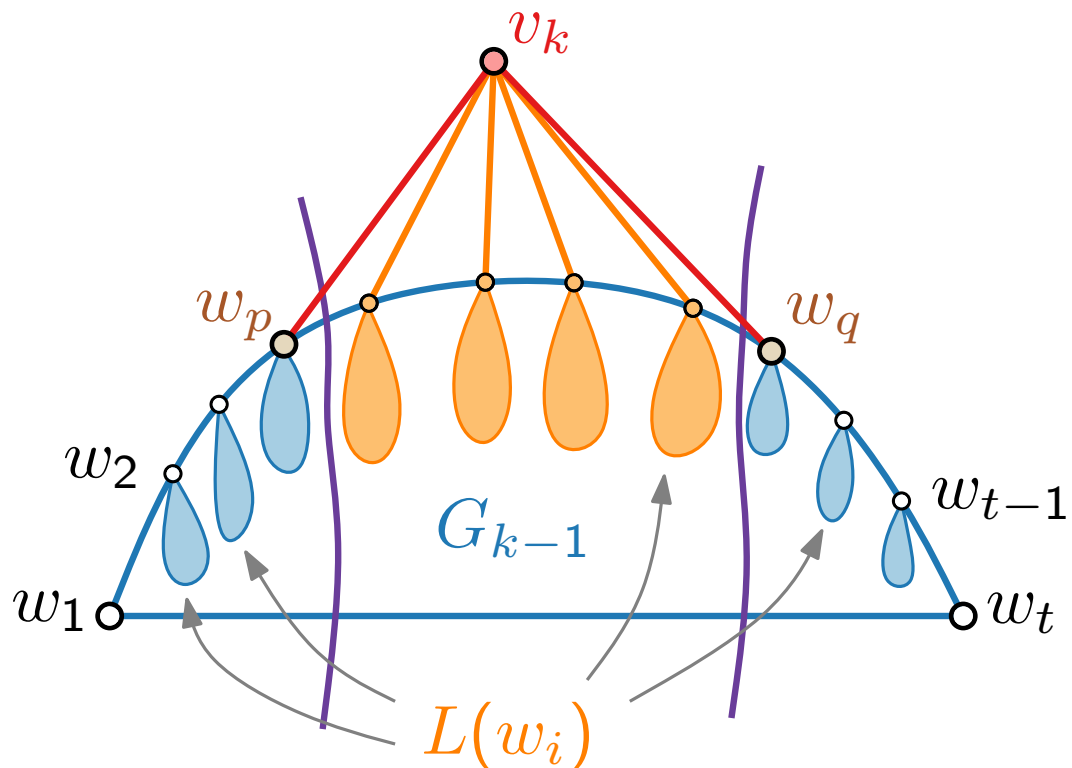
# Shift Method – Planarity

## Observations.

- Each internal vertex is **covered** exactly once.
- Covering relation defines a tree in  $G$
- and a forest in  $G_i$ ,  $1 \leq i \leq n - 1$ .

## Lemma.

Let  $0 < \delta_1 \leq \delta_2 \leq \dots \leq \delta_t \in \mathbb{N}$ , such that  $\delta_q - \delta_p \geq 2$  and even.



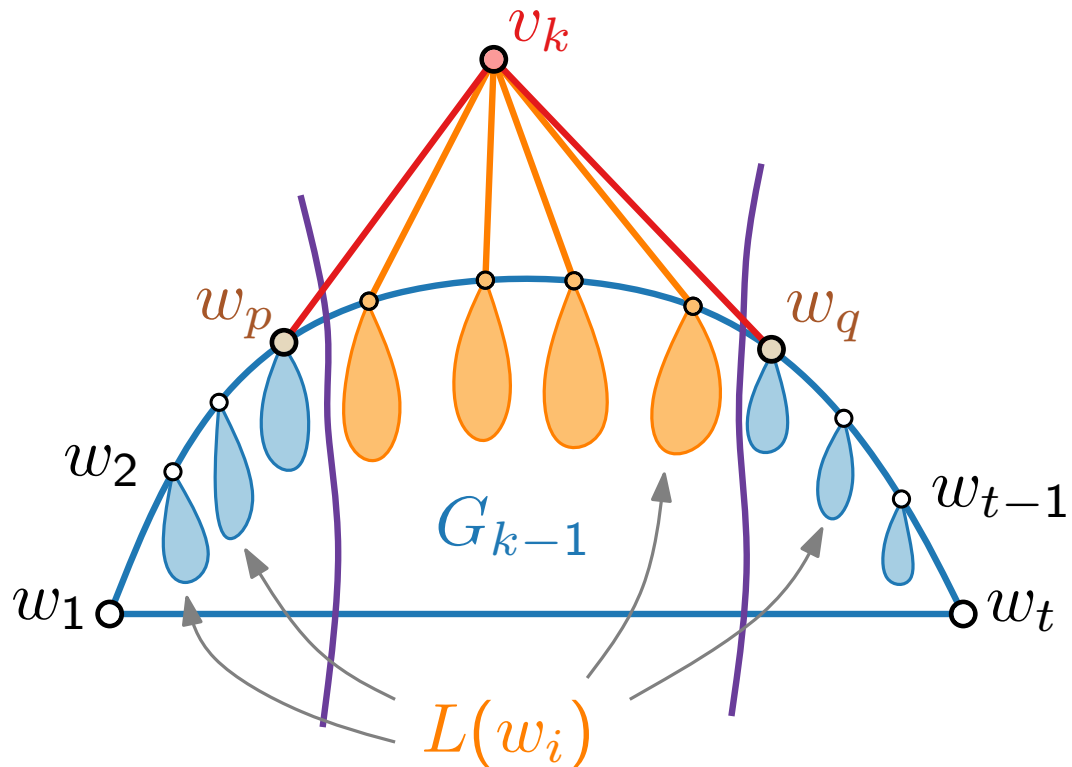
# Shift Method – Planarity

## Observations.

- Each internal vertex is **covered** exactly once.
- Covering relation defines a tree in  $G$
- and a forest in  $G_i$ ,  $1 \leq i \leq n - 1$ .

## Lemma.

Let  $0 < \delta_1 \leq \delta_2 \leq \dots \leq \delta_t \in \mathbb{N}$ , such that  $\delta_q - \delta_p \geq 2$  and even. If we shift  $L(w_i)$  by  $\delta_i$  to the right, then we get a planar straight-line drawing.





# Shift Method – Planarity

## Observations.

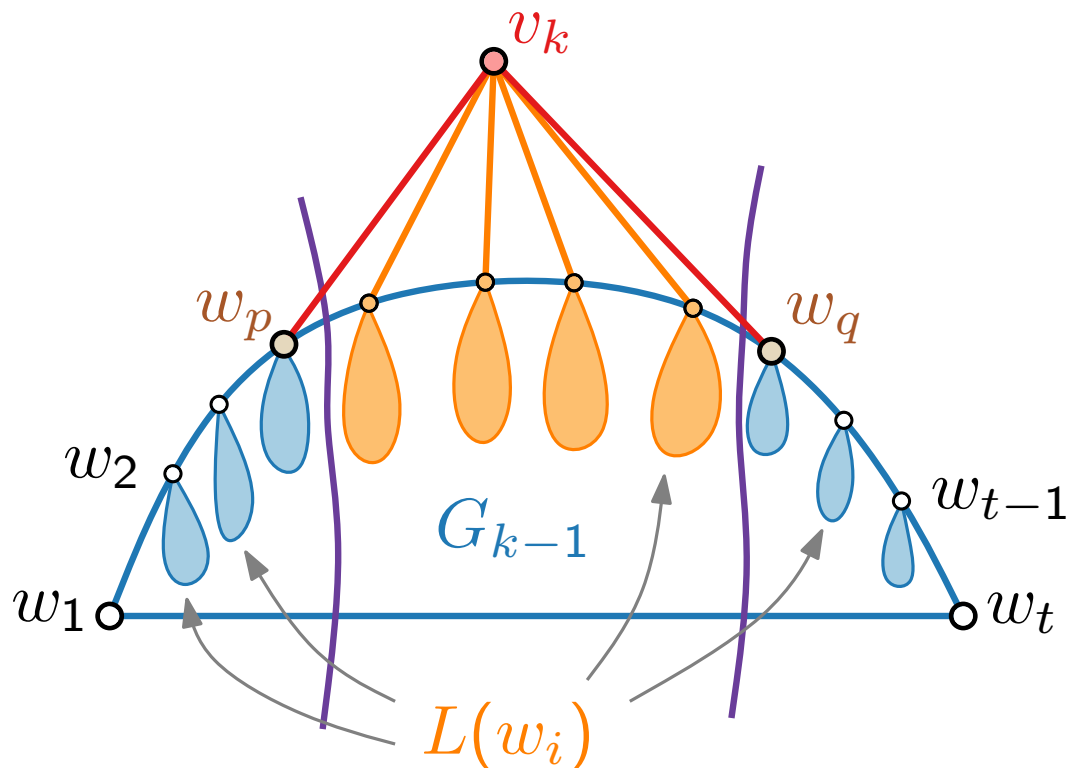
- Each internal vertex is **covered** exactly once.
- Covering relation defines a tree in  $G$
- and a forest in  $G_i$ ,  $1 \leq i \leq n - 1$ .

## Lemma.

Let  $0 < \delta_1 \leq \delta_2 \leq \dots \leq \delta_t \in \mathbb{N}$ , such that  $\delta_q - \delta_p \geq 2$  and even. If we shift  $L(w_i)$  by  $\delta_i$  to the right, then we get a planar straight-line drawing.

Proof by induction:

If  $G_{k-1}$  is drawn planar and straight-line, then so is  $G_k$ .



# Shift Method – Pseudocode

Let  $v_1, \dots, v_n$  be a canonical order of  $G$ .

**for**  $i = 1$  to  $3$  **do**

└

**for**  $i = 4$  to  $n$  **do**

└

# Shift Method – Pseudocode

Let  $v_1, \dots, v_n$  be a canonical order of  $G$ .

**for**  $i = 1$  to  $3$  **do**

└  $L(v_i) \leftarrow \{v_i\}$

**for**  $i = 4$  to  $n$  **do**

└

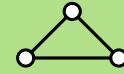
# Shift Method – Pseudocode

Let  $v_1, \dots, v_n$  be a canonical order of  $G$ .

**for**  $i = 1$  to  $3$  **do**

$L(v_i) \leftarrow \{v_i\}$

$P(v_1) \leftarrow (0, 0); P(v_2) \leftarrow (2, 0), P(v_3) \leftarrow (1, 1)$



**for**  $i = 4$  to  $n$  **do**

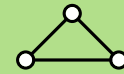
# Shift Method – Pseudocode

Let  $v_1, \dots, v_n$  be a canonical order of  $G$ .

**for**  $i = 1$  to 3 **do**

$L(v_i) \leftarrow \{v_i\}$

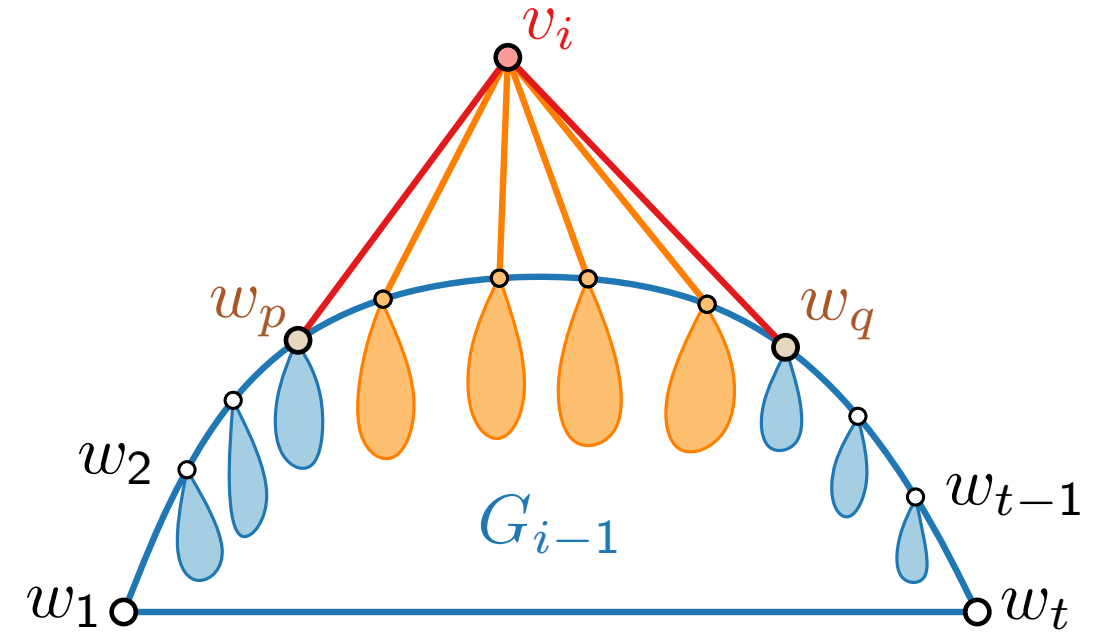
$P(v_1) \leftarrow (0, 0); P(v_2) \leftarrow (2, 0), P(v_3) \leftarrow (1, 1)$



**for**  $i = 4$  to  $n$  **do**

    Let  $\partial G_{i-1}$  be  $v_1 = w_1, w_2, \dots, w_{t-1}, w_t = v_2$ .

    Let  $w_p, \dots, w_q$  be the neighbors of  $v_i$ .



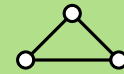
# Shift Method – Pseudocode

Let  $v_1, \dots, v_n$  be a canonical order of  $G$ .

**for**  $i = 1$  to  $3$  **do**

$L(v_i) \leftarrow \{v_i\}$

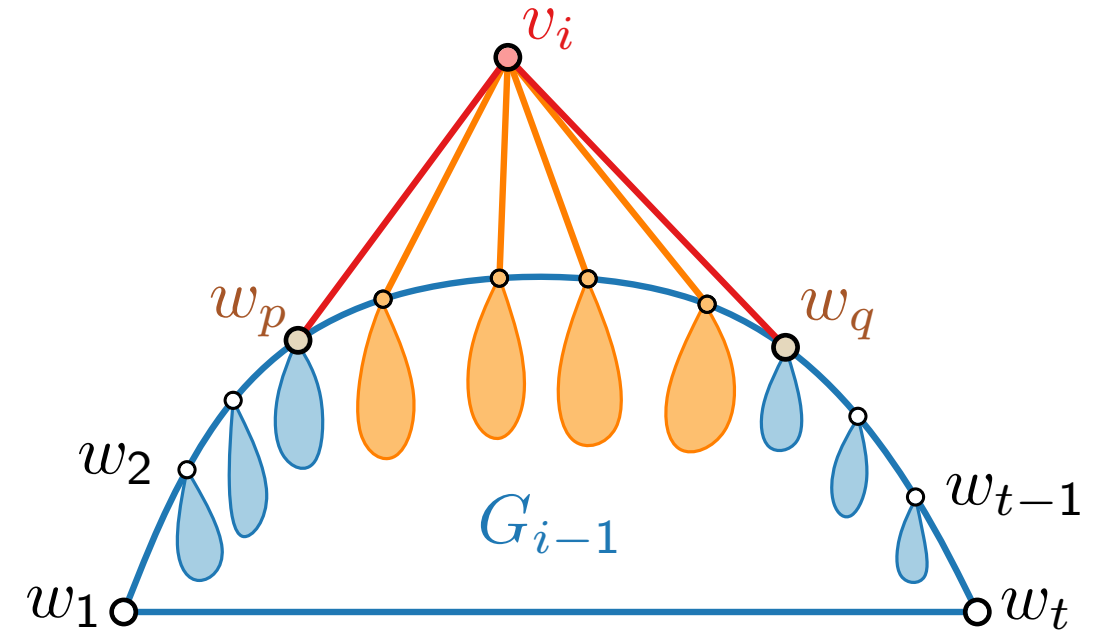
$P(v_1) \leftarrow (0, 0); P(v_2) \leftarrow (2, 0), P(v_3) \leftarrow (1, 1)$



**for**  $i = 4$  to  $n$  **do**

    Let  $\partial G_{i-1}$  be  $v_1 = w_1, w_2, \dots, w_{t-1}, w_t = v_2$ .

    Let  $w_p, \dots, w_q$  be the neighbors of  $v_i$ .



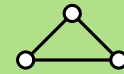
# Shift Method – Pseudocode

Let  $v_1, \dots, v_n$  be a canonical order of  $G$ .

**for**  $i = 1$  to 3 **do**

└  $L(v_i) \leftarrow \{v_i\}$

$P(v_1) \leftarrow (0, 0); P(v_2) \leftarrow (2, 0), P(v_3) \leftarrow (1, 1)$



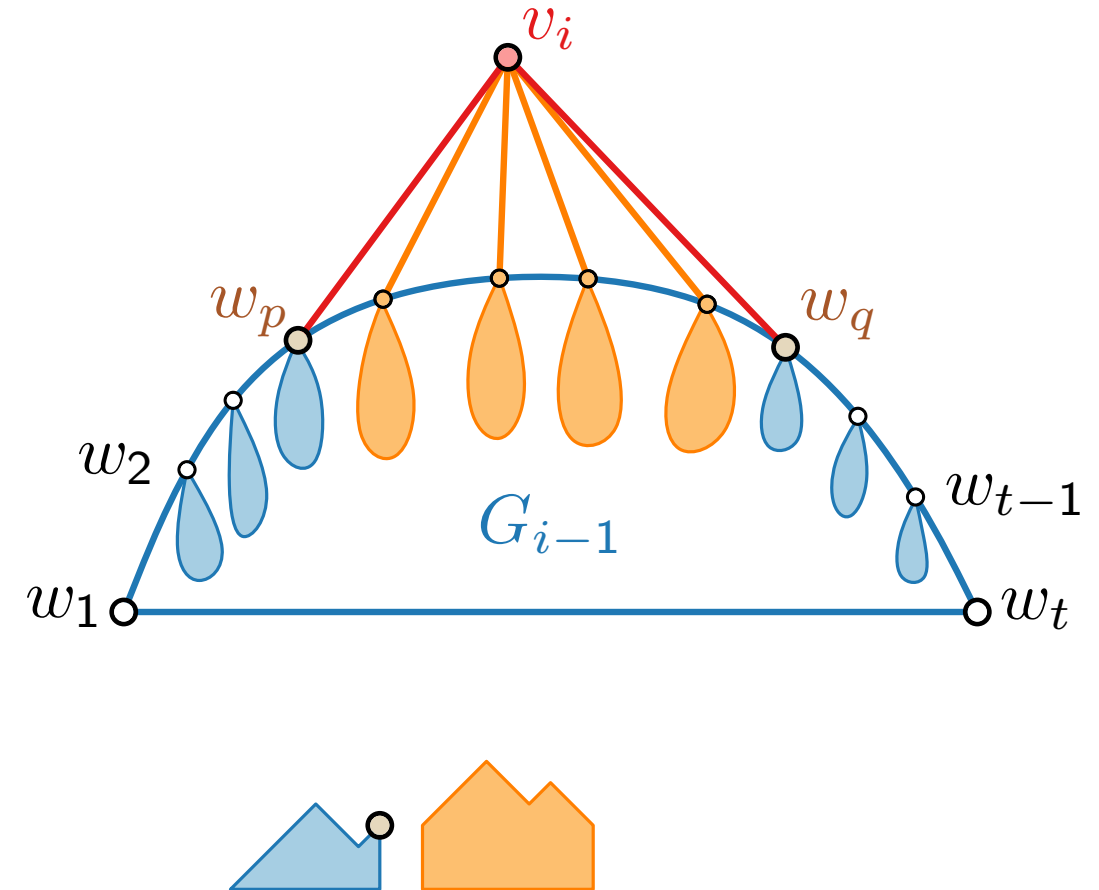
**for**  $i = 4$  to  $n$  **do**

Let  $\partial G_{i-1}$  be  $v_1 = w_1, w_2, \dots, w_{t-1}, w_t = v_2$ .

Let  $w_p, \dots, w_q$  be the neighbors of  $v_i$ .

**foreach**  $v \in \bigcup_{j=p+1}^{q-1} L(w_j)$  **do**

└



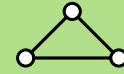
# Shift Method – Pseudocode

Let  $v_1, \dots, v_n$  be a canonical order of  $G$ .

**for**  $i = 1$  to 3 **do**

$L(v_i) \leftarrow \{v_i\}$

$P(v_1) \leftarrow (0, 0); P(v_2) \leftarrow (2, 0), P(v_3) \leftarrow (1, 1)$



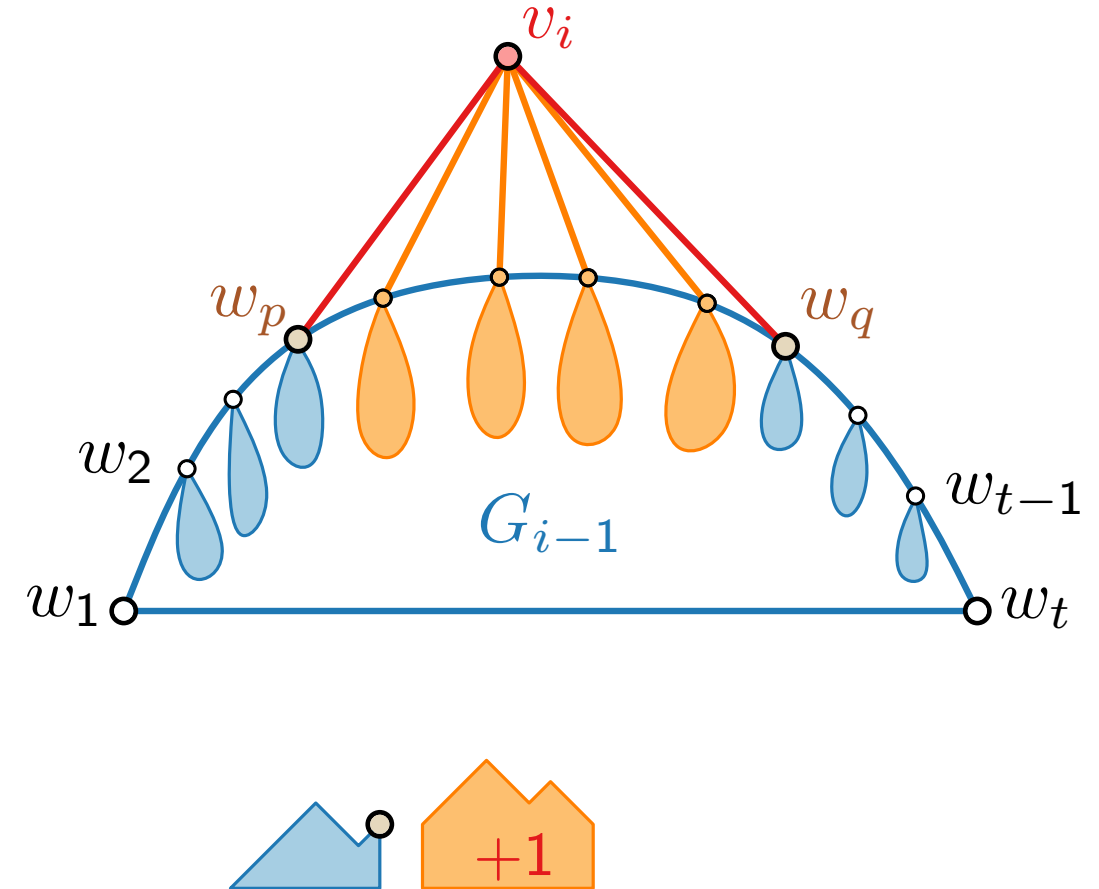
**for**  $i = 4$  to  $n$  **do**

    Let  $\partial G_{i-1}$  be  $v_1 = w_1, w_2, \dots, w_{t-1}, w_t = v_2$ .

    Let  $w_p, \dots, w_q$  be the neighbors of  $v_i$ .

**foreach**  $v \in \bigcup_{j=p+1}^{q-1} L(w_j)$  **do**

$x(v) \leftarrow x(v) + 1$





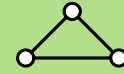
# Shift Method – Pseudocode

Let  $v_1, \dots, v_n$  be a canonical order of  $G$ .

**for**  $i = 1$  to 3 **do**

$L(v_i) \leftarrow \{v_i\}$

$P(v_1) \leftarrow (0, 0); P(v_2) \leftarrow (2, 0), P(v_3) \leftarrow (1, 1)$



**for**  $i = 4$  to  $n$  **do**

    Let  $\partial G_{i-1}$  be  $v_1 = w_1, w_2, \dots, w_{t-1}, w_t = v_2$ .

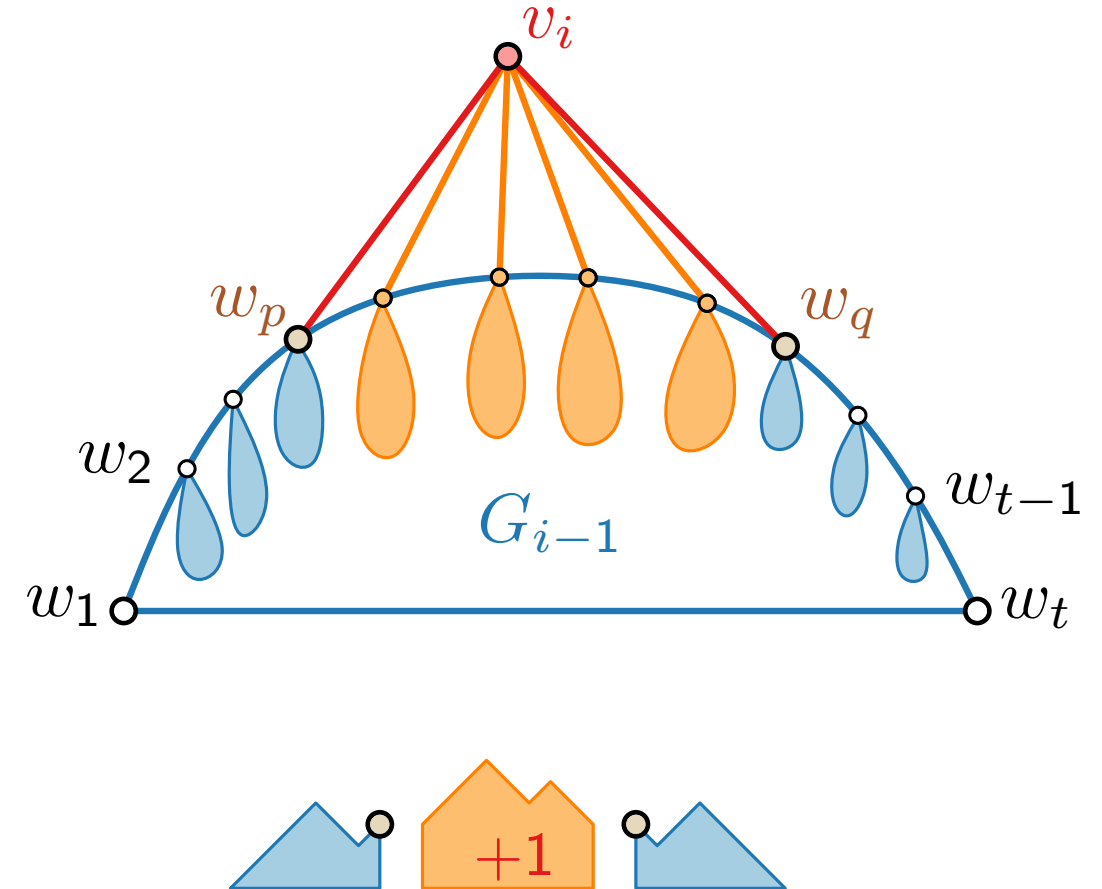
    Let  $w_p, \dots, w_q$  be the neighbors of  $v_i$ .

**foreach**  $v \in \bigcup_{j=p+1}^{q-1} L(w_j)$  **do**

$x(v) \leftarrow x(v) + 1$

**foreach**  $v \in \bigcup_{j=q}^t L(w_j)$  **do**

        ...



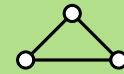
# Shift Method – Pseudocode

Let  $v_1, \dots, v_n$  be a canonical order of  $G$ .

**for**  $i = 1$  to 3 **do**

$L(v_i) \leftarrow \{v_i\}$

$P(v_1) \leftarrow (0, 0); P(v_2) \leftarrow (2, 0), P(v_3) \leftarrow (1, 1)$



**for**  $i = 4$  to  $n$  **do**

    Let  $\partial G_{i-1}$  be  $v_1 = w_1, w_2, \dots, w_{t-1}, w_t = v_2$ .

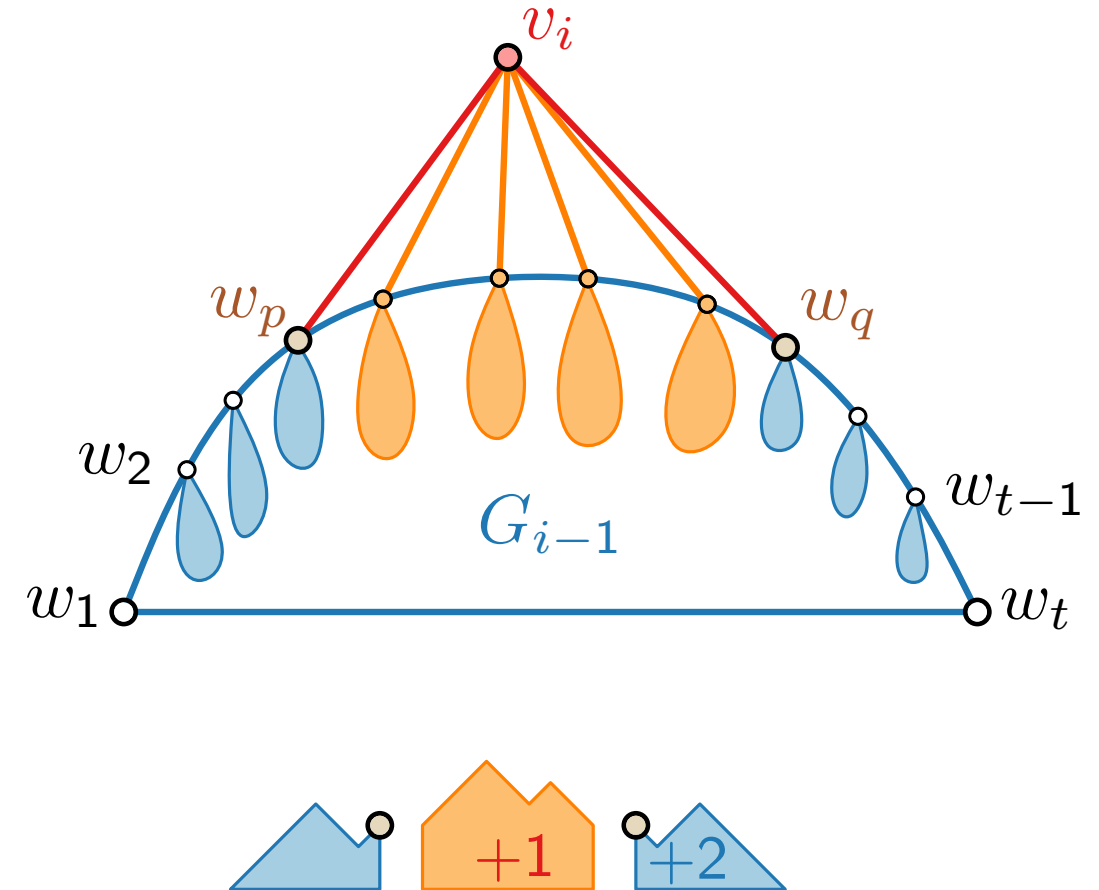
    Let  $w_p, \dots, w_q$  be the neighbors of  $v_i$ .

**foreach**  $v \in \bigcup_{j=p+1}^{q-1} L(w_j)$  **do**

$x(v) \leftarrow x(v) + 1$

**foreach**  $v \in \bigcup_{j=q}^t L(w_j)$  **do**

$x(v) \leftarrow x(v) + 2$



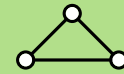
# Shift Method – Pseudocode

Let  $v_1, \dots, v_n$  be a canonical order of  $G$ .

**for**  $i = 1$  to 3 **do**

$L(v_i) \leftarrow \{v_i\}$

$P(v_1) \leftarrow (0, 0); P(v_2) \leftarrow (2, 0), P(v_3) \leftarrow (1, 1)$



**for**  $i = 4$  to  $n$  **do**

    Let  $\partial G_{i-1}$  be  $v_1 = w_1, w_2, \dots, w_{t-1}, w_t = v_2$ .

    Let  $w_p, \dots, w_q$  be the neighbors of  $v_i$ .

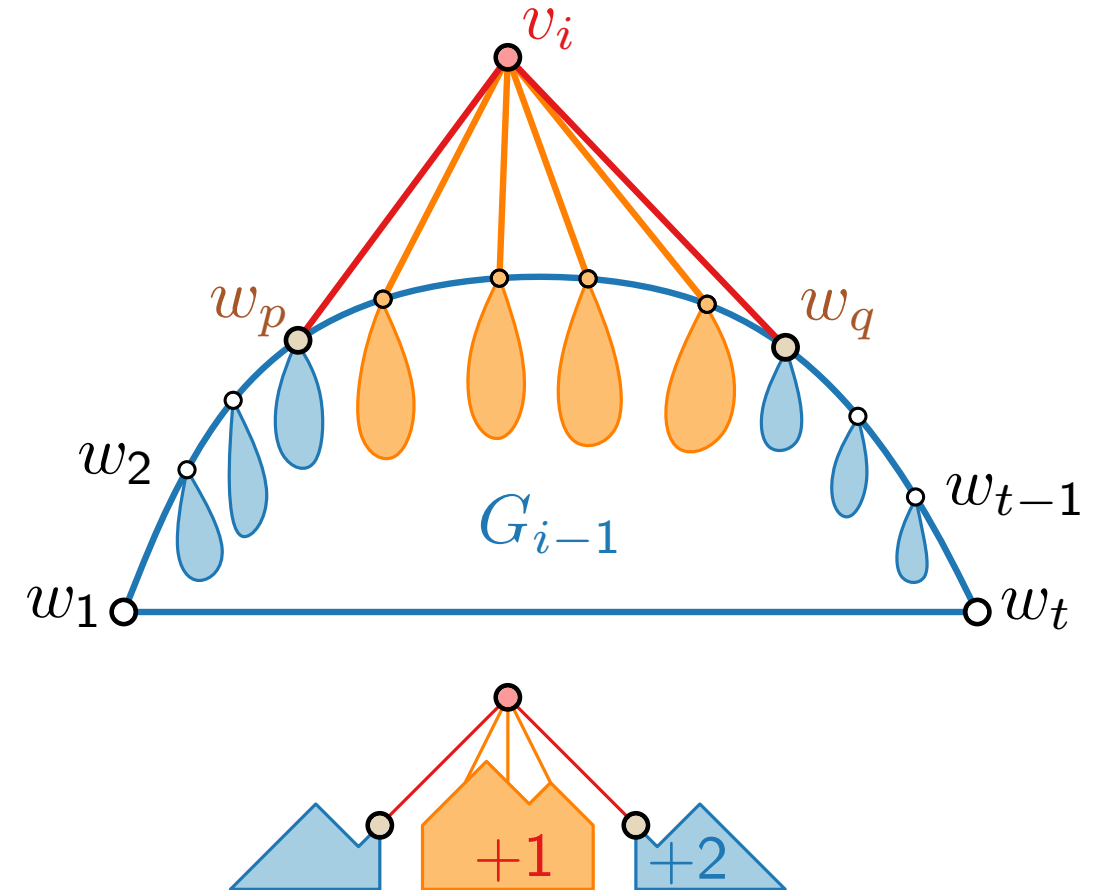
**foreach**  $v \in \bigcup_{j=p+1}^{q-1} L(w_j)$  **do**

$x(v) \leftarrow x(v) + 1$

**foreach**  $v \in \bigcup_{j=q}^t L(w_j)$  **do**

$x(v) \leftarrow x(v) + 2$

$P(v_i) \leftarrow$  intersection of slope- $\pm 1$  diagonals  
        through  $P(w_p)$  and  $P(w_q)$



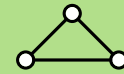
# Shift Method – Pseudocode

Let  $v_1, \dots, v_n$  be a canonical order of  $G$ .

**for**  $i = 1$  to  $3$  **do**

$L(v_i) \leftarrow \{v_i\}$

$P(v_1) \leftarrow (0, 0); P(v_2) \leftarrow (2, 0), P(v_3) \leftarrow (1, 1)$



**for**  $i = 4$  to  $n$  **do**

    Let  $\partial G_{i-1}$  be  $v_1 = w_1, w_2, \dots, w_{t-1}, w_t = v_2$ .

    Let  $w_p, \dots, w_q$  be the neighbors of  $v_i$ .

**foreach**  $v \in \bigcup_{j=p+1}^{q-1} L(w_j)$  **do**

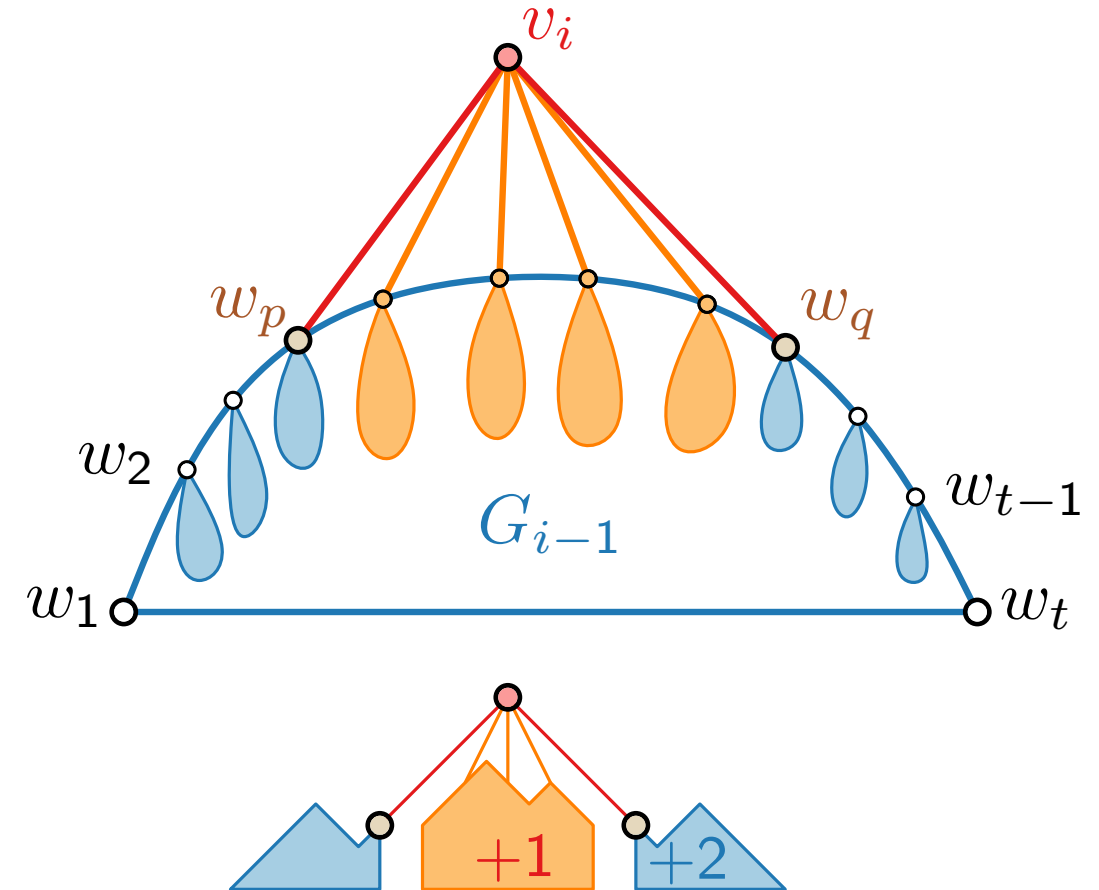
$x(v) \leftarrow x(v) + 1$

**foreach**  $v \in \bigcup_{j=q}^t L(w_j)$  **do**

$x(v) \leftarrow x(v) + 2$

$P(v_i) \leftarrow$  intersection of slope- $\pm 1$  diagonals  
        through  $P(w_p)$  and  $P(w_q)$

$L(v_i) \leftarrow \bigcup_{j=p+1}^{q-1} L(w_j) \cup \{v_i\}$



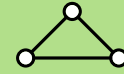
# Shift Method – Pseudocode

Let  $v_1, \dots, v_n$  be a canonical order of  $G$ .

**for**  $i = 1$  to 3 **do**

$L(v_i) \leftarrow \{v_i\}$

$P(v_1) \leftarrow (0, 0); P(v_2) \leftarrow (2, 0), P(v_3) \leftarrow (1, 1)$



**for**  $i = 4$  to  $n$  **do**

    Let  $\partial G_{i-1}$  be  $v_1 = w_1, w_2, \dots, w_{t-1}, w_t = v_2$ .

    Let  $w_p, \dots, w_q$  be the neighbors of  $v_i$ .

**foreach**  $v \in \bigcup_{j=p+1}^{q-1} L(w_j)$  **do**

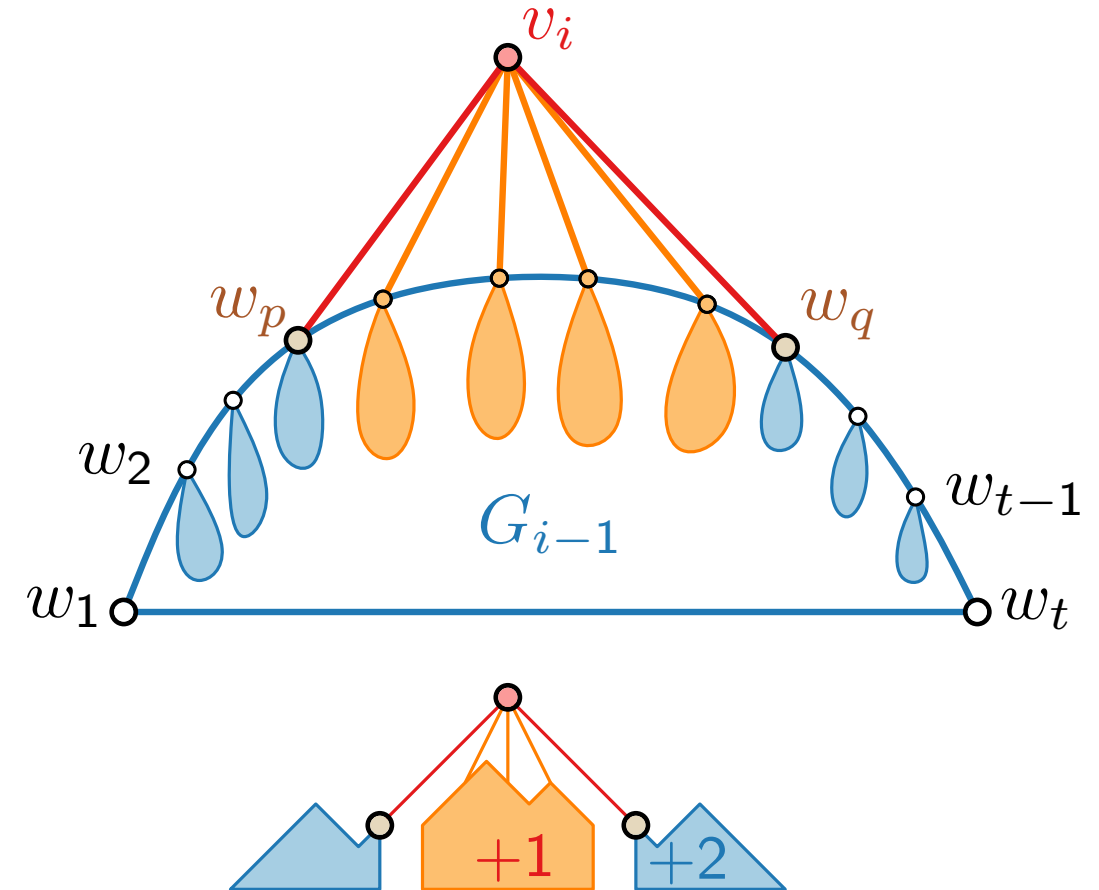
$x(v) \leftarrow x(v) + 1$

**foreach**  $v \in \bigcup_{j=q}^t L(w_j)$  **do**

$x(v) \leftarrow x(v) + 2$

$P(v_i) \leftarrow$  intersection of slope- $\pm 1$  diagonals  
        through  $P(w_p)$  and  $P(w_q)$

$L(v_i) \leftarrow \bigcup_{j=p+1}^{q-1} L(w_j) \cup \{v_i\}$



**Running Time?**

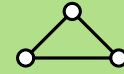
# Shift Method – Pseudocode

Let  $v_1, \dots, v_n$  be a canonical order of  $G$ .

**for**  $i = 1$  to  $3$  **do**

$L(v_i) \leftarrow \{v_i\}$

$P(v_1) \leftarrow (0, 0); P(v_2) \leftarrow (2, 0), P(v_3) \leftarrow (1, 1)$



**for**  $i = 4$  to  $n$  **do**

    Let  $\partial G_{i-1}$  be  $v_1 = w_1, w_2, \dots, w_{t-1}, w_t = v_2$ .

    Let  $w_p, \dots, w_q$  be the neighbors of  $v_i$ .

**foreach**  $v \in \bigcup_{j=p+1}^{q-1} L(w_j)$  **do**      *//  $\mathcal{O}(n^2)$  in total*

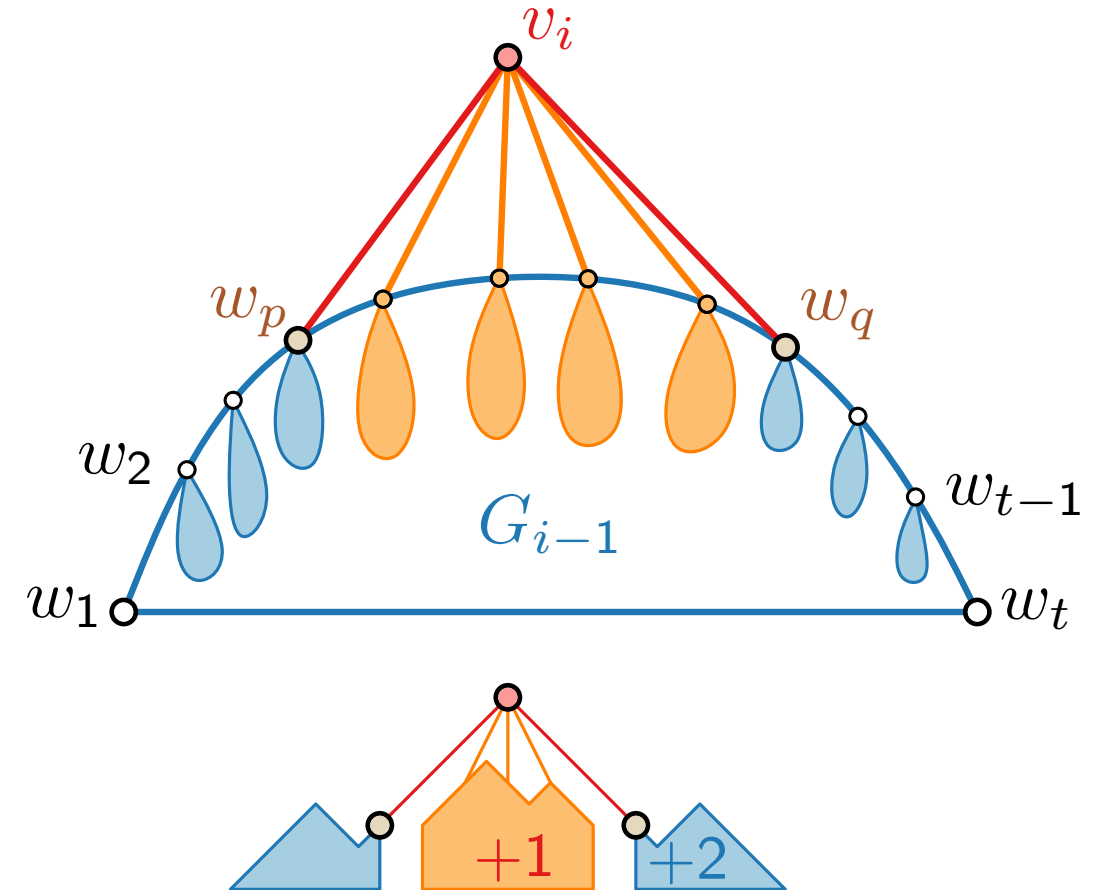
$x(v) \leftarrow x(v) + 1$

**foreach**  $v \in \bigcup_{j=q}^t L(w_j)$  **do**      *//  $\mathcal{O}(n^2)$  in total*

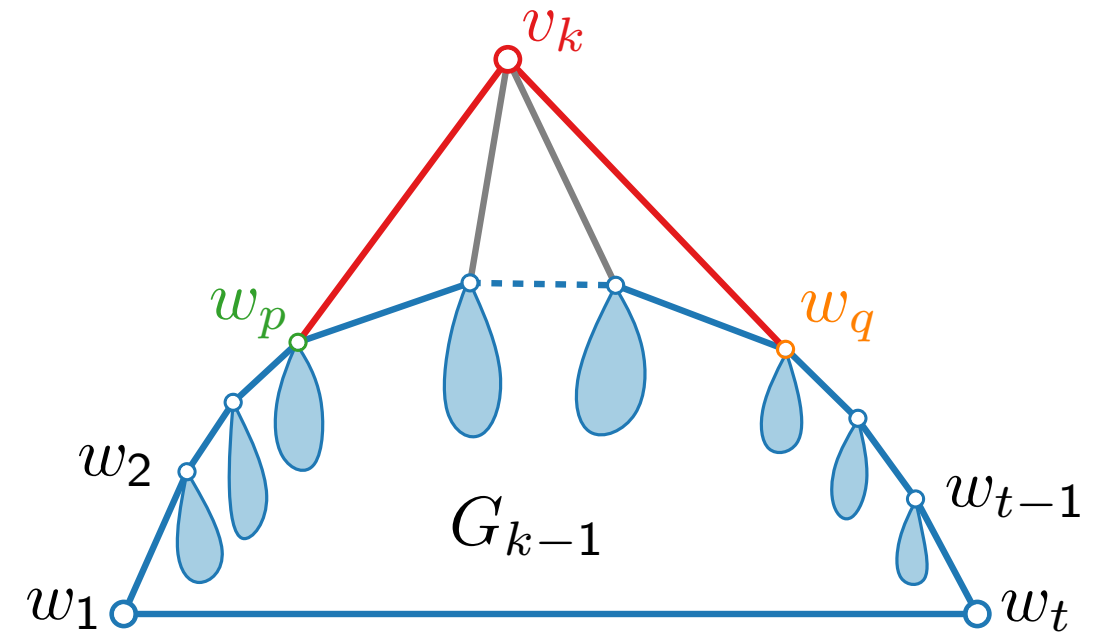
$x(v) \leftarrow x(v) + 2$

$P(v_i) \leftarrow$  intersection of slope- $\pm 1$  diagonals  
        through  $P(w_p)$  and  $P(w_q)$

$L(v_i) \leftarrow \bigcup_{j=p+1}^{q-1} L(w_j) \cup \{v_i\}$



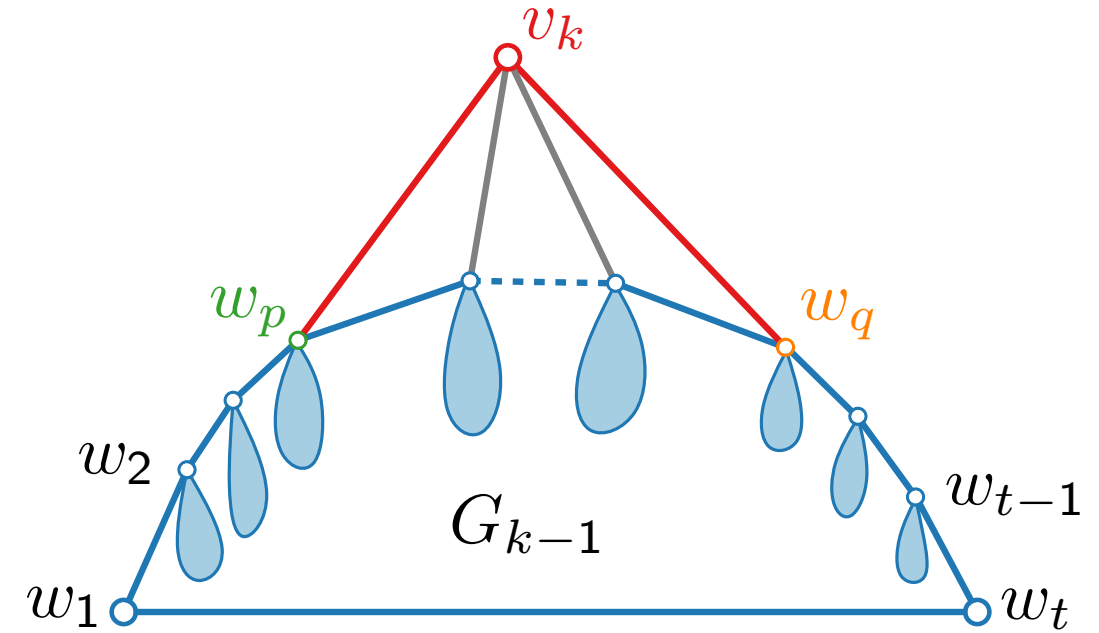
**Running Time?**



# Shift Method – Linear-Time Implementation

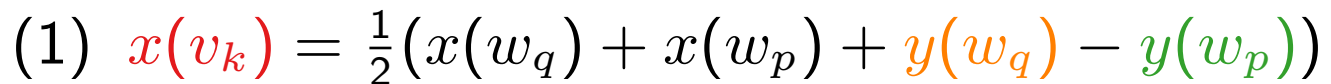
## Idea 1.

To compute  $x(v_k)$  &  $y(v_k)$ ,  
we only need  $y(w_p)$  and  $y(w_q)$  and  $x(w_q) - x(w_p)$

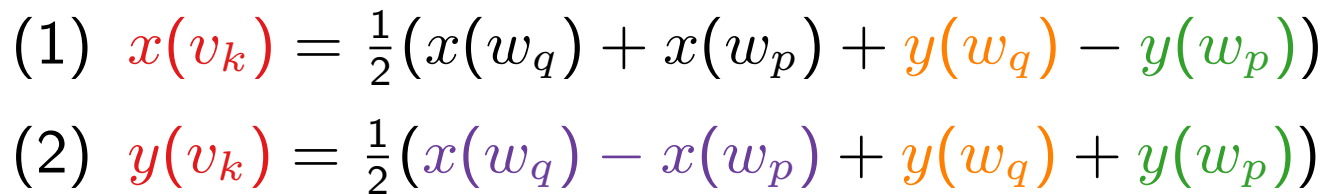




To compute  $x(v_k)$  &  $y(v_k)$ ,  
we only need  $y(w_p)$  and  $y(w_q)$  and  $x(w_q) - x(w_p)$

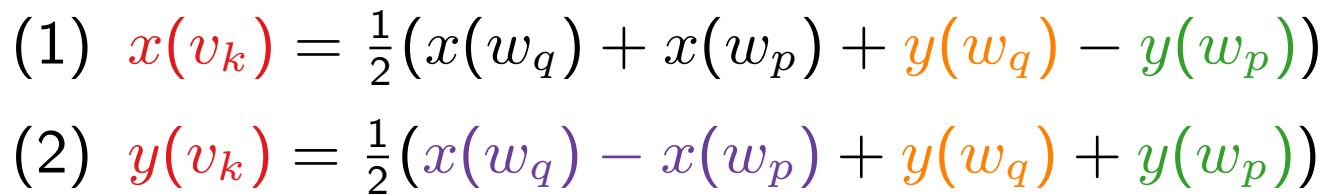


To compute  $x(v_k)$  &  $y(v_k)$ ,  
we only need  $y(w_p)$  and  $y(w_q)$  and  $x(w_q) - x(w_p)$



To compute  $x(v_k)$  &  $y(v_k)$ ,  
we only need  $y(w_p)$  and  $y(w_q)$  and  $x(w_q) - x(w_p)$

Instead of storing explicit x-coordinates,  
we store x-distances.



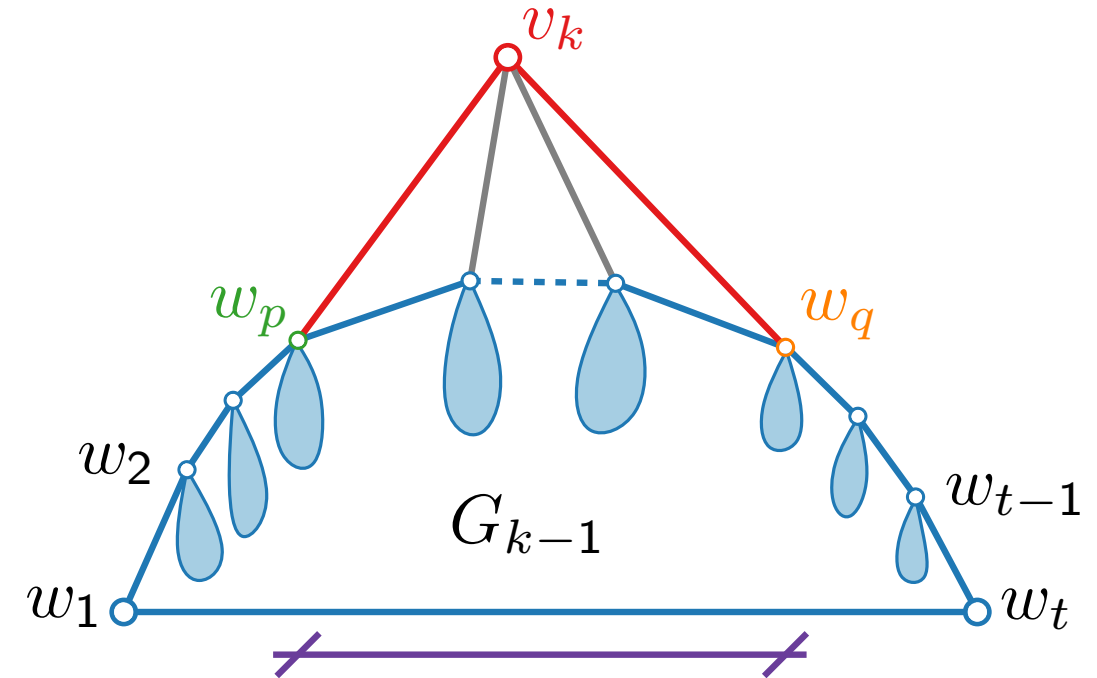
# Shift Method – Linear-Time Implementation

## Idea 1.

To compute  $x(v_k)$  &  $y(v_k)$ ,  
we only need  $y(w_p)$  and  $y(w_q)$  and  $x(w_q) - x(w_p)$

## Idea 2.

Instead of storing explicit x-coordinates,  
we store x-distances.



$$(1) \quad x(v_k) = \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p))$$

$$(2) \quad y(v_k) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p))$$

$$(3) \quad x(v_k) - x(w_p) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))$$

# Shift Method – Linear-Time Implementation

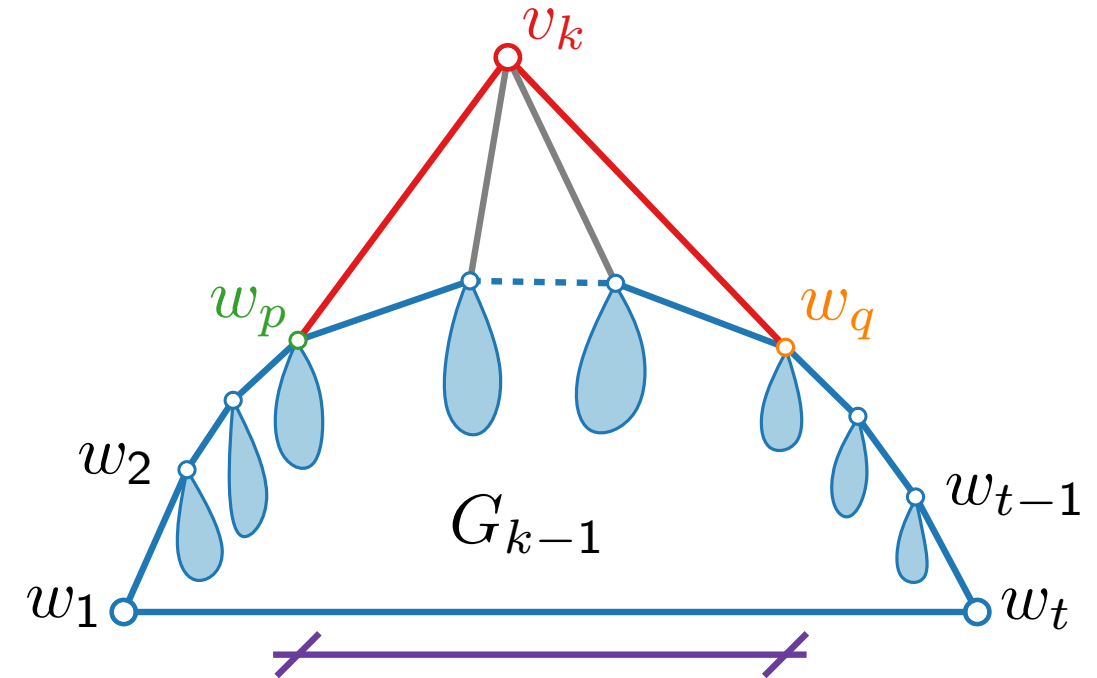
## Idea 1.

To compute  $x(v_k)$  &  $y(v_k)$ ,  
we only need  $y(w_p)$  and  $y(w_q)$  and  $x(w_q) - x(w_p)$

## Idea 2.

Instead of storing explicit x-coordinates,  
we store x-distances.

After an x-distance is computed for each  $v_k$ ,  
use preorder traversal to compute all x-coordinates.



$$(1) \quad x(v_k) = \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p))$$

$$(2) \quad y(v_k) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p))$$

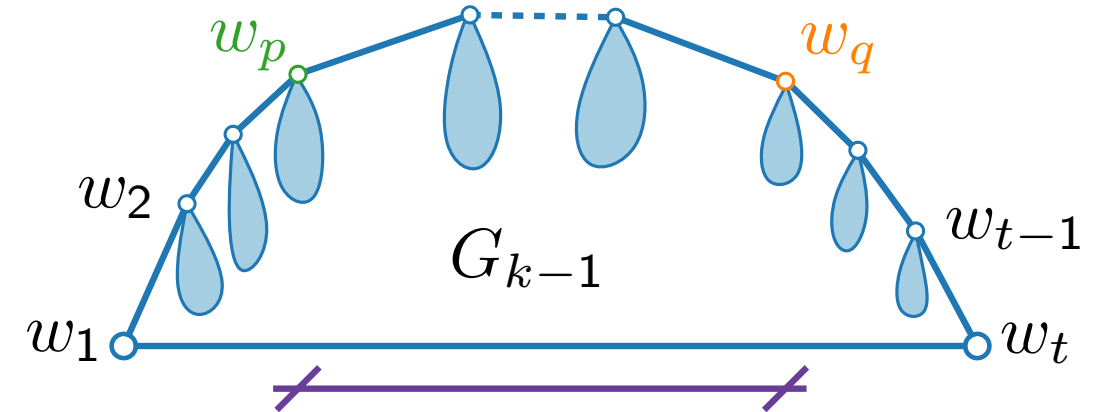
$$(3) \quad x(v_k) - x(w_p) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))$$

# Shift Method – Linear-Time Implementation

## Relative x-distance tree.

For each vertex  $v$  store

- x-offset  $\Delta_x(v)$  from parent
- y-coordinate  $y(v)$



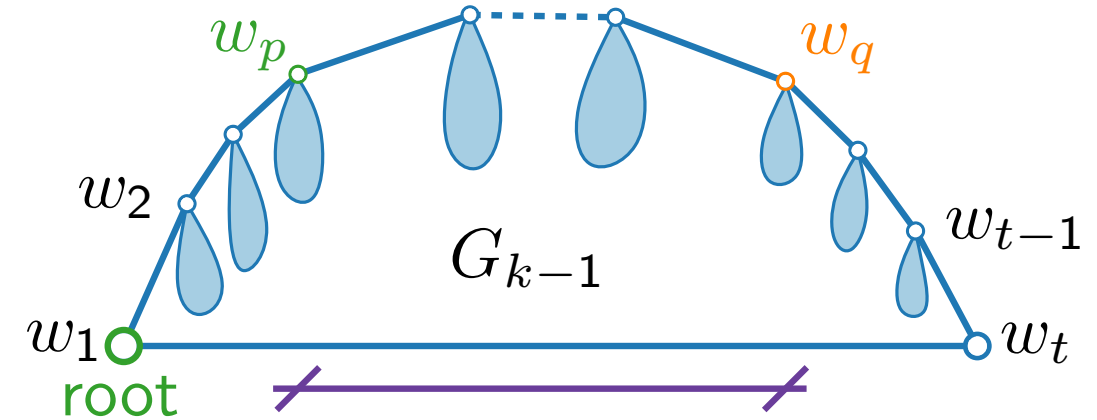
- (1)  $x(v_k) = \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p))$
- (2)  $y(v_k) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p))$
- (3)  $x(v_k) - x(w_p) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))$

# Shift Method – Linear-Time Implementation

## Relative x-distance tree.

For each vertex  $v$  store

- x-offset  $\Delta_x(v)$  from parent
- y-coordinate  $y(v)$



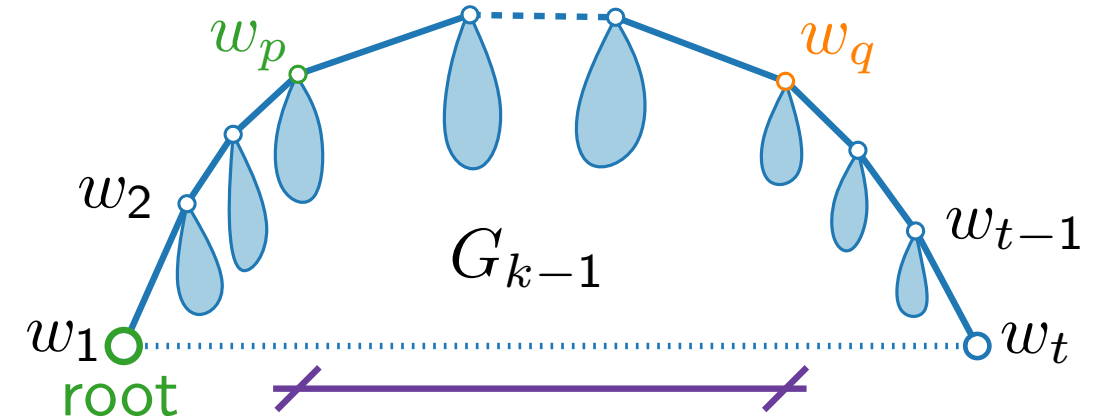
- (1)  $x(v_k) = \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p))$
- (2)  $y(v_k) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p))$
- (3)  $x(v_k) - x(w_p) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))$

# Shift Method – Linear-Time Implementation

## Relative x-distance tree.

For each vertex  $v$  store

- x-offset  $\Delta_x(v)$  from parent
- y-coordinate  $y(v)$



- (1)  $x(v_k) = \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p))$
- (2)  $y(v_k) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p))$
- (3)  $x(v_k) - x(w_p) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))$

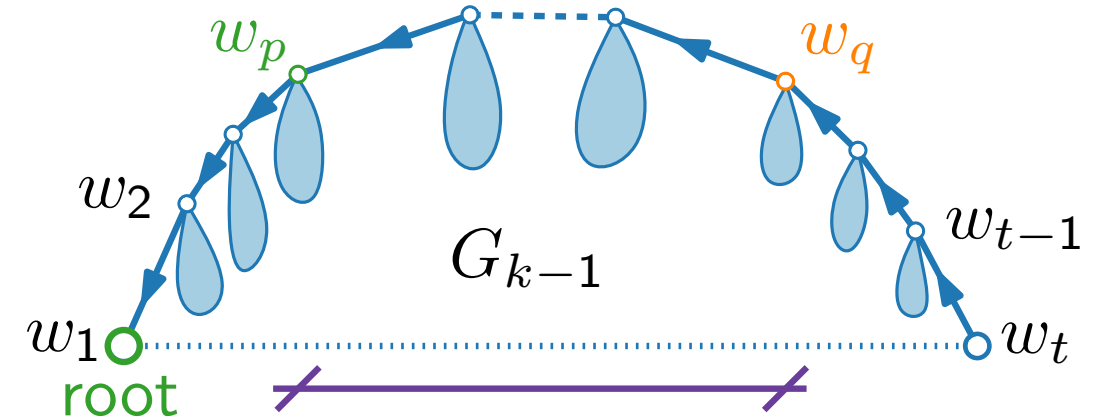


# Shift Method – Linear-Time Implementation

## Relative x-distance tree.

For each vertex  $v$  store

- x-offset  $\Delta_x(v)$  from parent
- y-coordinate  $y(v)$



- (1)  $x(v_k) = \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p))$
- (2)  $y(v_k) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p))$
- (3)  $x(v_k) - x(w_p) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))$

For each vertex  $v$  store

- 
- The diagram illustrates a tree structure  $G_{k-1}$  with root  $v_k$ . The root  $v_k$  is a red node. The tree has a sequence of nodes  $w_1, w_2, \dots, w_t$  along its left and right sides.  $w_1$  is a green node labeled "root".  $w_2$  is a black node.  $w_p$  is a green node.  $w_q$  is an orange node.  $w_t$  is an orange node. The tree has blue edges and blue teardrop shapes. A horizontal dotted line separates the tree from a purple line with slashes below it.

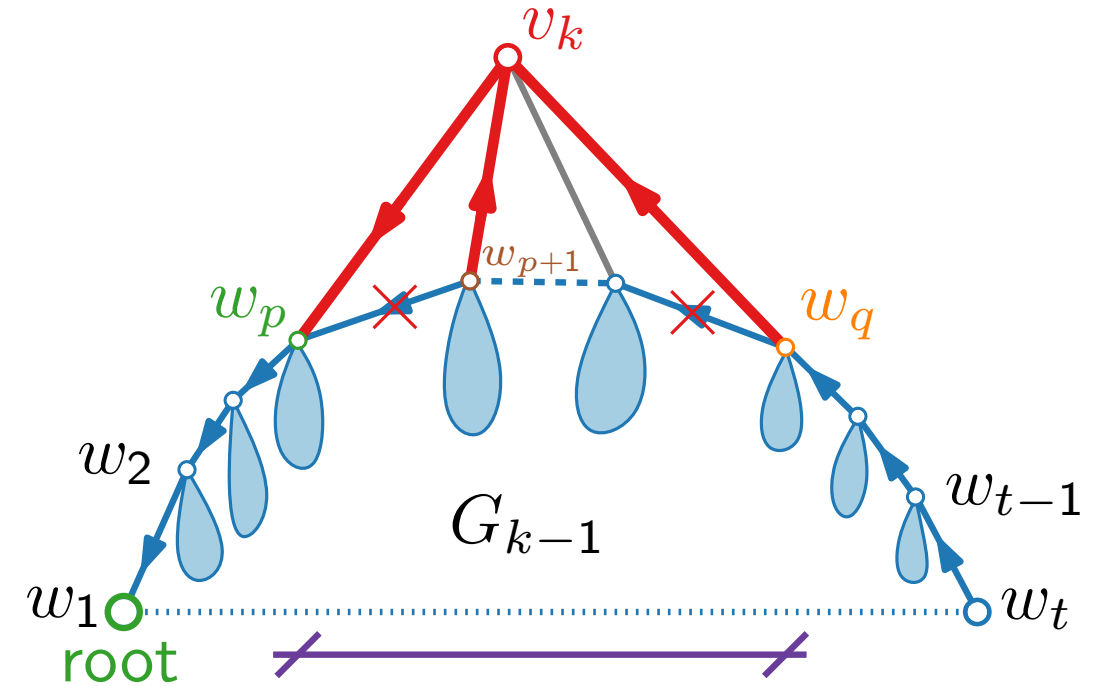
$$(3) \quad x(v_k) - x(w_p) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))$$

# Shift Method – Linear-Time Implementation

## Relative x-distance tree.

For each vertex  $v$  store

- x-offset  $\Delta_x(v)$  from parent
- y-coordinate  $y(v)$



- (1)  $x(v_k) = \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p))$
- (2)  $y(v_k) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p))$
- (3)  $x(v_k) - x(w_p) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))$

# Shift Method – Linear-Time Implementation

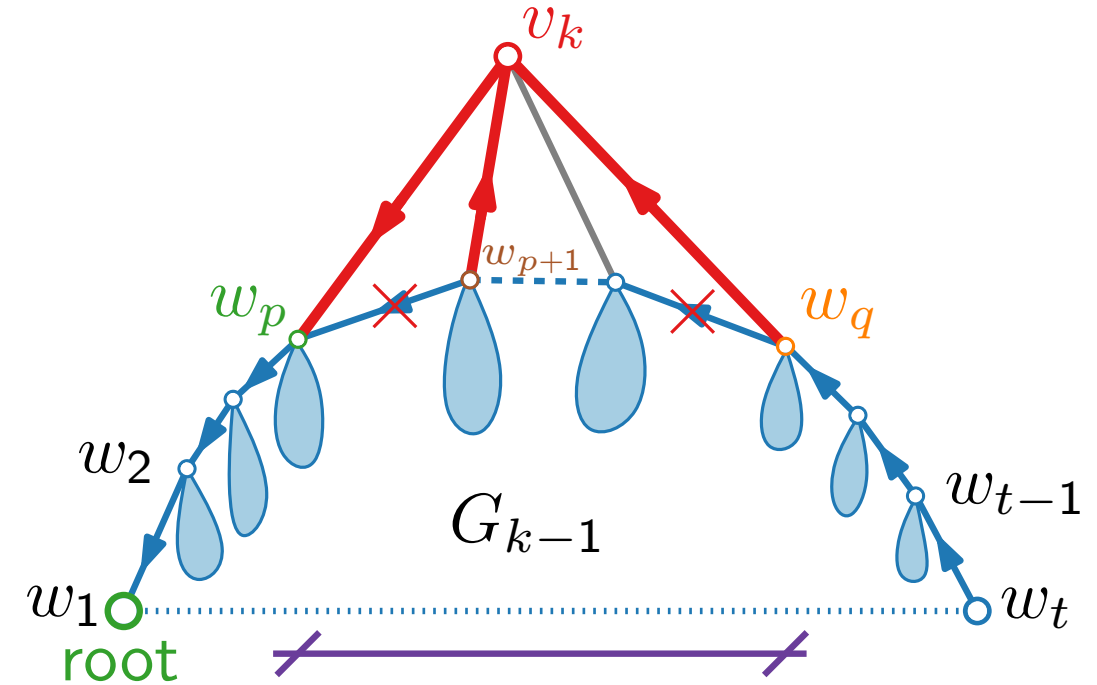
## Relative x-distance tree.

For each vertex  $v$  store

- x-offset  $\Delta_x(v)$  from parent
- y-coordinate  $y(v)$

## Calculations.

- $\Delta_x(w_{p+1})^{++}, \Delta_x(w_q)^{++}$



- (1)  $x(v_k) = \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p))$
- (2)  $y(v_k) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p))$
- (3)  $x(v_k) - x(w_p) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))$

# Shift Method – Linear-Time Implementation

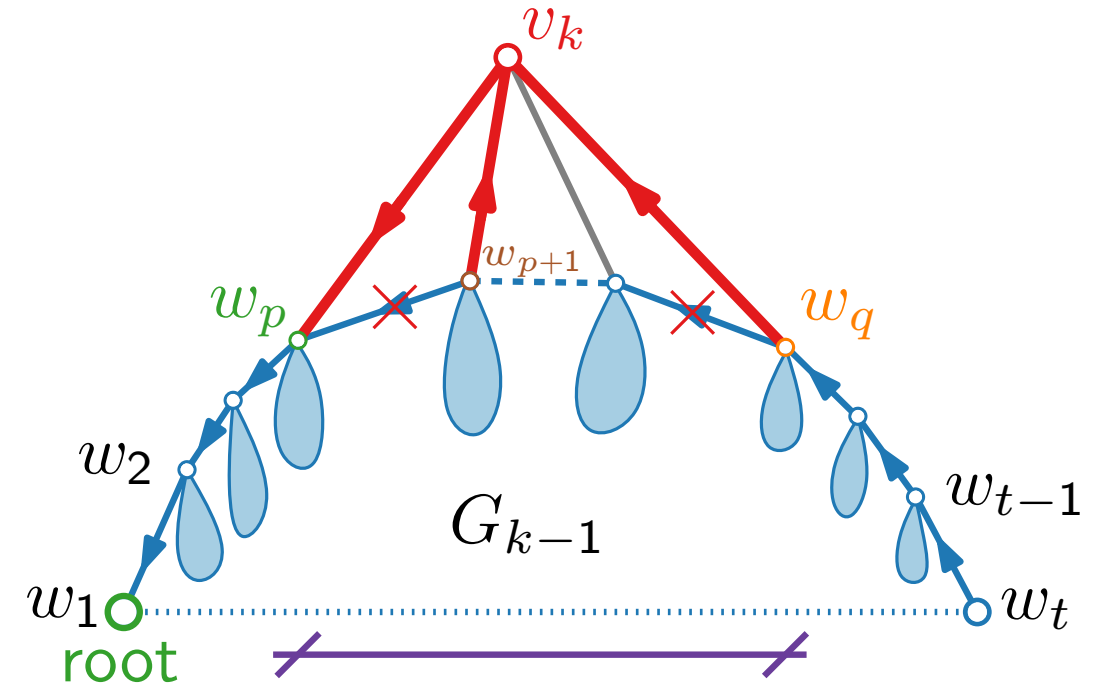
## Relative x-distance tree.

For each vertex  $v$  store

- x-offset  $\Delta_x(v)$  from parent
- y-coordinate  $y(v)$

## Calculations.

- $\Delta_x(w_{p+1})^{++}, \Delta_x(w_q)^{++}$
- $\Delta_x(w_p, w_q) = \Delta_x(w_{p+1}) + \dots + \Delta_x(w_q)$



- (1)  $x(v_k) = \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p))$
- (2)  $y(v_k) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p))$
- (3)  $x(v_k) - x(w_p) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))$

# Shift Method – Linear-Time Implementation

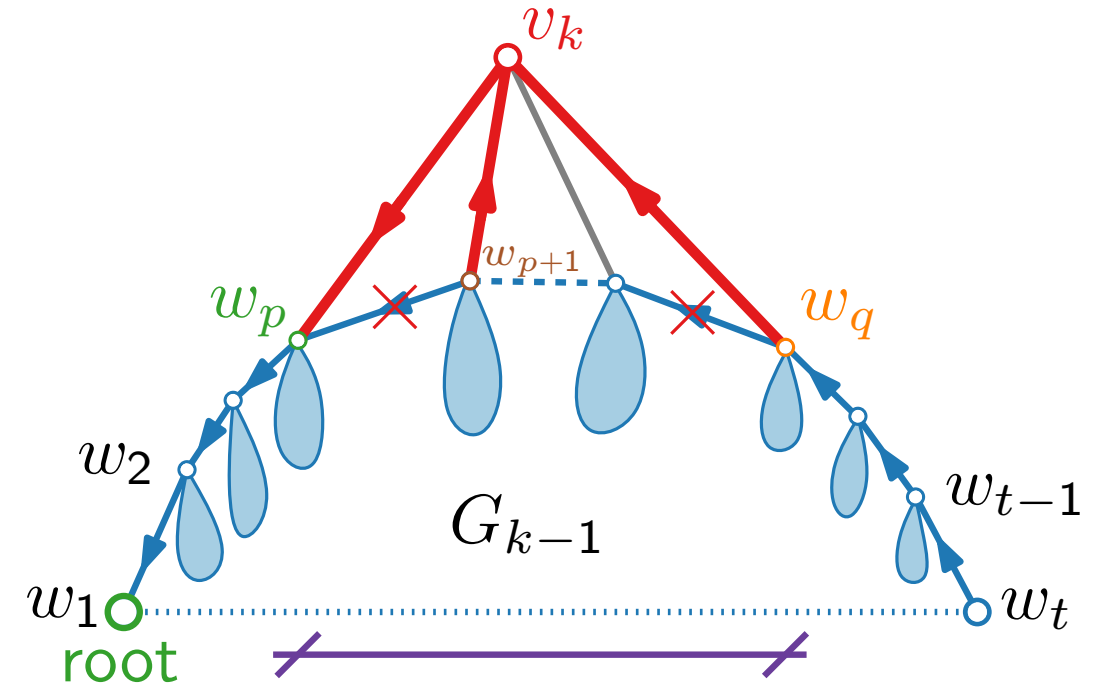
## Relative x-distance tree.

For each vertex  $v$  store

- x-offset  $\Delta_x(v)$  from parent
- y-coordinate  $y(v)$

## Calculations.

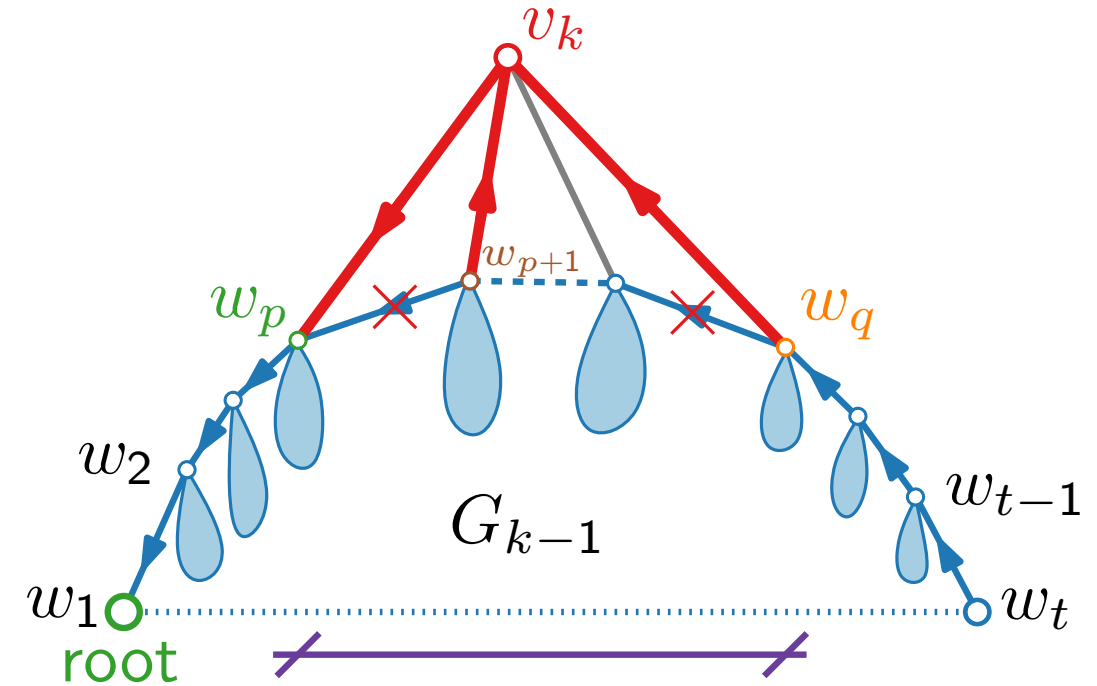
- $\Delta_x(w_{p+1})^{++}, \Delta_x(w_q)^{++}$
- $\Delta_x(w_p, w_q) = \Delta_x(w_{p+1}) + \dots + \Delta_x(w_q)$
- $\Delta_x(v_k)$  by (3)



- (1)  $x(v_k) = \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p))$
- (2)  $y(v_k) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p))$
- (3)  $x(v_k) - x(w_p) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))$

For each vertex  $v$  store

- $\Delta_x(w_{p+1})^{++}, \Delta_x(w_q)^{++}$
- $\Delta_x(w_p, w_q) = \Delta_x(w_{p+1}) + \dots + \Delta_x(w_q)$
- $\Delta_x(v_k)$  by (3)      ■  $y(v_k)$  by (2)



$$\begin{aligned} (1) \quad x(v_k) &= \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p)) \\ (2) \quad y(v_k) &= \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p)) \\ (3) \quad x(v_k) - x(w_p) &= \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p)) \end{aligned}$$

# Shift Method – Linear-Time Implementation

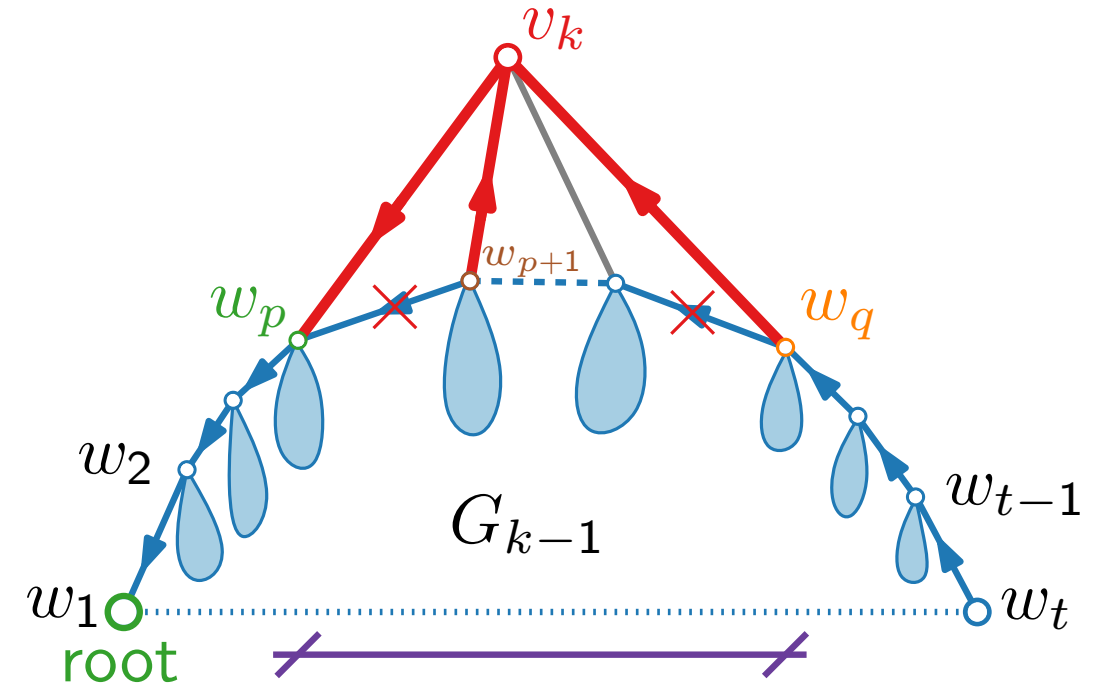
## Relative x-distance tree.

For each vertex  $v$  store

- x-offset  $\Delta_x(v)$  from parent
- y-coordinate  $y(v)$

## Calculations.

- $\Delta_x(w_{p+1})++$ ,  $\Delta_x(w_q)++$
- $\Delta_x(w_p, w_q) = \Delta_x(w_{p+1}) + \dots + \Delta_x(w_q)$
- $\Delta_x(v_k)$  by (3)      ■  $y(v_k)$  by (2)
- $\Delta_x(w_q) = \Delta_x(w_p, w_q) - \Delta_x(v_k)$



- (1)  $x(v_k) = \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p))$
- (2)  $y(v_k) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p))$
- (3)  $x(v_k) - x(w_p) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))$



# Shift Method – Linear-Time Implementation

## Relative x-distance tree.

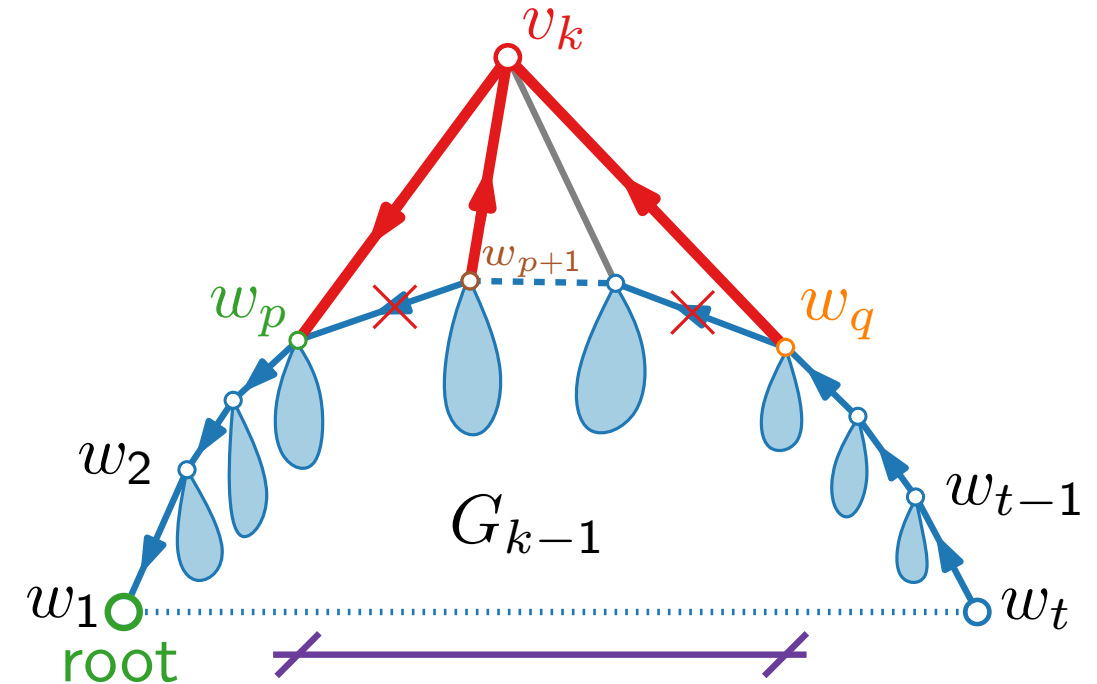
For each vertex  $v$  store

- x-offset  $\Delta_x(v)$  from parent
- y-coordinate  $y(v)$

## Calculations.

- $\Delta_x(w_{p+1})++$ ,  $\Delta_x(w_q)++$
- $\Delta_x(w_p, w_q) = \Delta_x(w_{p+1}) + \dots + \Delta_x(w_q)$
- $\Delta_x(v_k)$  by (3)      ■  $y(v_k)$  by (2)
- $\Delta_x(w_q) = \Delta_x(w_p, w_q) - \Delta_x(v_k)$
- $\Delta_x(w_{p+1}) = \Delta_x(w_{p+1}) - \Delta_x(v_k)$

- (1)  $x(v_k) = \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p))$
- (2)  $y(v_k) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p))$
- (3)  $x(v_k) - x(w_p) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))$



# Shift Method – Linear-Time Implementation

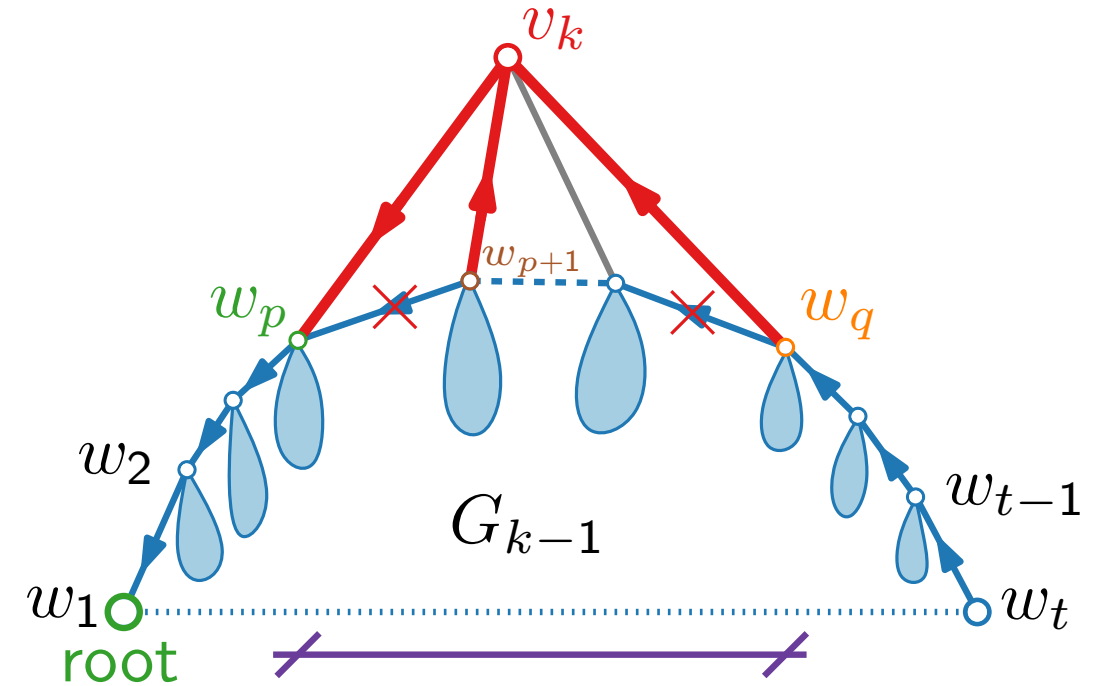
## Relative x-distance tree.

For each vertex  $v$  store

- x-offset  $\Delta_x(v)$  from parent
- y-coordinate  $y(v)$

## Calculations.

- $\Delta_x(w_{p+1})++$ ,  $\Delta_x(w_q)++$
- $\Delta_x(w_p, w_q) = \Delta_x(w_{p+1}) + \dots + \Delta_x(w_q)$
- $\Delta_x(v_k)$  by (3)      ■  $y(v_k)$  by (2)
- $\Delta_x(w_q) = \Delta_x(w_p, w_q) - \Delta_x(v_k)$
- $\Delta_x(w_{p+1}) = \Delta_x(w_{p+1}) - \Delta_x(v_k)$



$\mathcal{O}(n)$  in total

- (1)  $x(v_k) = \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p))$
- (2)  $y(v_k) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p))$
- (3)  $x(v_k) - x(w_p) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))$

# Literature

- [PGD Ch. 4.2] for detailed explanation of shift method
- [de Fraysseix, Pach, Pollack 1990] “How to draw a planar graph on a grid”
  - original paper on shift method