Visualization of Graphs

Lecture 3:
Straight-Line Drawings of Planar Graphs I:
Canonical Ordering and Shift Method

Part I:
Planar Straight-Line Drawings

Jonathan Klawitter
Planar Graphs

$G$
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.

$G$ is drawn in such a way that no edges cross each other:

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

$G$ is planar:

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

Diagram showing the planar graph $G$ with vertices and edges labeled accordingly.
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.

$G = \{1 \rightarrow (2, 3, 5), 2 \rightarrow (3, 1, 4), 3 \rightarrow (4, 1, 2), 4 \rightarrow (5, 3, 2)\}$
**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.

1 → (2, 3, 5)  
2 → (3, 1, 4)  
3 → (4, 1, 2)  
4 → (5, 3, 2)  
5 → (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.
Planar Graphs

A planar graph can have many planar embeddings.

$G$ 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.

A planar graph can have many
planar embeddings.

A planar embedding can have many
planar drawings!

$G$ is planar:
$1 \rightarrow (2, 3, 5)$
$2 \rightarrow (3, 1, 4)$
$3 \rightarrow (4, 1, 2)$
$4 \rightarrow (5, 3, 2)$
$5 \rightarrow (1, 4)$

$G$ is planar:
$1 \rightarrow (2, 5, 3)$
$2 \rightarrow (3, 4, 1)$
$3 \rightarrow (4, 2, 1)$
$4 \rightarrow (5, 2, 3)$
$5 \rightarrow (1, 4)$
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.

$$\text{#faces} - \text{#edges} + \text{#vertices} = \text{#conn.comp.} + 1$$

$$f - m + n = c + 1$$
Planar Graphs

A planar graph 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!

**Euler’s polyhedra formula.**

\[
\text{faces} - \text{edges} + \text{vertices} = \text{conn. comp.} + 1
\]

\[
f - m + n = c + 1
\]

**Proof.**
A planar graph can have many planar embeddings.

A planar embedding can have many planar drawings!

Euler’s polyhedra formula.

\[ f - m + n = c + 1 \]

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.

$\#\text{faces} - \#\text{edges} + \#\text{vertices} = \#\text{conn.comp.} + 1$

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.**

\[ \text{#faces} - \text{#edges} + \text{#vertices} = \text{#conn.comp.} + 1 \]

\[ f - m + n = c + 1 \]

**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.**

\[ \#\text{faces} - \#\text{edges} + \#\text{vertices} = \#\text{conn.comp.} + 1 \]

**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.**

$\#\text{faces} - \#\text{edges} + \#\text{vertices} = \#\text{conn.comp.} + 1$

$\quad f - m + n = c + 1$

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

$m = 0 \Rightarrow f = 1$ and $c = n$

$\Rightarrow 1 - 0 + n = n + 1$
**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.**

$$\text{#faces} - \text{#edges} + \text{#vertices} = \text{#conn.comp.} + 1$$

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

$m = 0 \Rightarrow f = 1$ and $c = n$

$$\Rightarrow 1 - 0 + n = n + 1 \checkmark$$
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.**

\[
\text{faces} - \text{edges} + \text{vertices} = \text{conn. comp.} + 1
\]

\[
f - m + n = c + 1
\]

**Proof.** By induction on \( m \):

\( m = 0 \Rightarrow f = 1 \) and \( c = n \)

\[
\Rightarrow 1 - 0 + n = n + 1 \checkmark
\]

\( m > 1 \Rightarrow \)
**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.**

\[
\text{#faces} - \text{#edges} + \text{#vertices} = \text{#conn.comp.} + 1
\]

\[
f - m + n = c + 1
\]

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

- \( m = 0 \Rightarrow f = 1 \) and \( c = n \)
  \[
  1 - 0 + n = n + 1 \checkmark
  \]
- \( m > 1 \Rightarrow \) remove 1 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.**

\[
f - m + n = c + 1
\]

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

$m = 0 \Rightarrow f = 1$ and $c = n$

\[
\Rightarrow 1 - 0 + n = n + 1 \checkmark
\]

$m > 1 \Rightarrow$ remove 1 edge $e \Rightarrow m - 1$
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.**

\[
\text{faces} - \text{edges} + \text{vertices} = \text{#conn.comp.} + 1
\]

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

$m = 0 \Rightarrow f = 1$ and $c = n$

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

$m > 1 \Rightarrow$ remove 1 edge $e \Rightarrow m - 1$
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.**
\[
\text{faces} - \text{edges} + \text{vertices} = \text{conn.comp.} + 1
\]
\[
f - m + n = c + 1
\]

**Proof.** By induction on \(m\):

- \(m = 0 \Rightarrow f = 1\) and \(c = n\)
  \[
  \Rightarrow 1 - 0 + n = n + 1 \checkmark
  \]
- \(m > 1 \Rightarrow \text{remove 1 edge } e \Rightarrow m - 1\)
  \[
  \Rightarrow c + 1
  \]
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.**

\[
\text{faces} - \text{edges} + \text{vertices} = \text{conn. comp.} + 1
\]

\[
f - m + n = c + 1
\]

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

$m = 0 \Rightarrow f = 1$ and $c = n$

\[
\Rightarrow 1 - 0 + n = n + 1 \checkmark
\]

$m > 1 \Rightarrow \text{remove 1 edge } e \Rightarrow m - 1$

\[
\Rightarrow c + 1
\]

\[
\Rightarrow f - 1
\]
Properties of Planar Graphs

Euler’s polyhedra formula.
\[
\text{#faces} - \text{#edges} + \text{#vertices} = \text{#conn.comp.} + 1
\]
\[
f - m + n = c + 1
\]
Properties of Planar Graphs

**Euler’s polyhedra formula.**

\[ f - m + n = c + 1 \]

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

\[ \text{#faces} - \text{#edges} + \text{#vertices} = \text{#conn.comp.} + 1 \]
## Properties of Planar Graphs

### Euler’s polyhedron formula.

\[
\text{#faces} - \text{#edges} + \text{#vertices} = \text{#conn.comp.} + 1
\]

\[
f - m + n = c + 1
\]

### Theorem. \( G \) simple planar graph with \( n \geq 3 \).

1. \( m \leq 3n - 6 \)
Properties of Planar Graphs

Euler’s polyhedra formula.

\[ \#\text{faces} - \#\text{edges} + \#\text{vertices} = \#\text{conn.comp.} + 1 \]

\[ f - m + n = c + 1 \]

Theorem. \( G \) simple planar graph with \( n \geq 3 \).
1. \( m \leq 3n - 6 \)

Proof. 1.
Properties of Planar Graphs

**Euler’s polyhedra formula.**

\[ f - m + n = c + 1 \]

**Theorem.** \( G \) simple planar graph with \( n \geq 3 \).
1. \( m \leq 3n - 6 \)

**Proof.** 1. Every edge incident to \( \leq 2 \) faces
Properties of Planar Graphs

**Theorem.** $G$ simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

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

**Euler’s polyhedra formula.**

$$f - m + n = c + 1$$
Properties of Planar Graphs

**Euler’s polyhedra formula.**

\[ \text{#faces} - \text{#edges} + \text{#vertices} = \text{#conn.comp.} + 1 \]

**Theorem.** G simple planar graph with \( n \geq 3 \).

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

**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.**

\[ f - m + n = c + 1 \]

**Theorem.** A simple planar graph with \( n \geq 3 \).

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.
\[ f - m + n = c + 1 \]

Theorem. G simple planar graph with \( n \geq 3 \).
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.

\[ f - m + n = c + 1 \]

Theorem. \( G \) simple planar graph with \( n \geq 3 \).
1. \( m \leq 3n - 6 \)

Proof. 1. Every edge incident to \( \leq 2 \) faces

Every face incident to \( \geq 3 \) edges

\[ 3f \leq 2m \]

\[ 6 \leq 3c + 3 \leq 3f - 3m + 3n \leq 2m - 3m + 3n \]
Properties of Planar Graphs

**Euler’s polyhedra formula.**

\[ f - m + n = c + 1 \]

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

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.
\[ f - m + n = c + 1 \]

Theorem. \( G \) simple planar graph with \( n \geq 3 \).
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.**

\[
\text{#faces} - \text{#edges} + \text{#vertices} = \text{#conn.comp.} + 1
\]

\[
f - m + n = c + 1
\]

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

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

**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.
\[ \text{#faces} - \text{#edges} + \text{#vertices} = \text{#conn.comp.} + 1 \]

**Theorem.** \( G \) simple planar graph with \( n \geq 3 \).
1. \( m \leq 3n - 6 \)
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.
\[ f - m + n = c + 1 \]

Theorem. \( G \) simple planar graph with \( n \geq 3 \).
1. \( m \leq 3n - 6 \)  
2. \( f \leq 2n - 4 \)  
3. There is a vertex of degree at most five

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.**

\[
\text{#faces} - \text{#edges} + \text{#vertices} = \text{#conn. comp.} + 1
\]

\[
f - m + n = c + 1
\]

**Theorem.** \( G \) simple planar graph with \( n \geq 3 \).

1. \( m \leq 3n - 6 \)
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.

\[ f - m + n = c + 1 \]

Theorem. G simple planar graph with \( n \geq 3 \).

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

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

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

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

**Euler’s polyhedra formula.**

$$f - m + n = c + 1$$

**Proof.**

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

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

   $m \leq 3n - 6$

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

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

A simple planar graph $G$ is a graph that can be drawn on a plane in such a way that its edges do not intersect, except at the vertices.
Properties of Planar Graphs

**Euler's polyhedra formula.**

\[ \text{#faces} - \text{#edges} + \text{#vertices} = \text{#conn.comp.} + 1 \]

\[ f - m + n = c + 1 \]

**Theorem.** \( G \) simple planar graph with \( n \geq 3 \).

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

**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. \( 3f \leq 2m \leq 6n - 12 \) \( \Rightarrow f \leq 2n - 4 \)

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

**Handshaking-Lemma.**
Properties of Planar Graphs

**Euler’s polyhedra formula.**

\[
\text{#faces} - \text{#edges} + \text{#vertices} = \#\text{conn.comp.} + 1
\]

\[
f - m + n = c + 1
\]

**Theorem.** \( G \) simple planar graph with \( n \geq 3 \).

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

**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. \( 3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4 \)

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

**Handshaking-Lemma.**

\[
\sum_{v \in V} \deg(v) = 2|E|
\]
Properties of Planar Graphs

**Euler’s polyhedra formula.**
\[
\text{#faces} - \text{#edges} + \text{#vertices} = \text{#conn.comp.} + 1
\]
\[
f - m + n = c + 1
\]

**Theorem.** $G$ simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

**Proof.**

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

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

\[\Rightarrow m \leq 3n - 6\]

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

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

**Handshaking-Lemma.**
\[
\sum_{v \in V} \deg(v) = 2|E|
\]
Properties of Planar Graphs

**Euler’s polyhedra formula.**

\[ \text{#faces} - \text{#edges} + \text{#vertices} = \text{#conn.comp.} + 1 \]

\[ f - m + n = c + 1 \]

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

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

**Proof.**

1. Every edge incident to $\leq 2$ faces
2. 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. $3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$

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

**Handshaking-Lemma.**

\[ \sum_{v \in V} \deg(v) = 2|E| \]
Properties of Planar Graphs

**Euler’s polyhedra formula.**

\[
\text{#faces} - \text{#edges} + \text{#vertices} = \text{#conn.comp.} + 1
\]

\[
f - m + n = c + 1
\]

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

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

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

3. $\sum_{v \in V} \deg(v) = 2m \leq 6n - 12$
   
   \[
   \Rightarrow \min_{v \in V} \deg(v) \leq 1/n \sum_{v \in V} \deg(v)
   \]

**Euler's polyhedra formula.**

\[
\text{#faces} - \text{#edges} + \text{#vertices} = \text{#conn.comp.} + 1
\]

\[
f - m + n = c + 1
\]

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

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

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

3. $\sum_{v \in V} \deg(v) = 2m \leq 6n - 12$
   
   \[
   \Rightarrow \min_{v \in V} \deg(v) \leq 1/n \sum_{v \in V} \deg(v)
   \]
Properties of Planar Graphs

**Euler’s polyhedra formula.**

\[ \text{#faces} - \text{#edges} + \text{#vertices} = \text{#conn.comp.} + 1 \]
\[ f - m + n = c + 1 \]

**Theorem.** $G$ simple planar graph with $n \geq 3$.
1. $m \leq 3n - 6$
2. $f \leq 2n - 4$
3. There is a vertex of degree at most five

**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. $3f \leq 2m \leq 6n - 12 \Rightarrow f \leq 2n - 4$
3. $\sum_{v \in V} \deg(v) = 2m \leq 6n - 12$
   \[ \Rightarrow \min_{v \in V} \deg(v) \leq 1/n \sum_{v \in V} \deg(v) < 6 \]

**Handshaking-Lemma.**
\[ \sum_{v \in V} \deg(v) = 2|E| \]
Triangulations

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

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

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

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

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 destroy planarity.
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 destroy planarity.
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 destroy planarity.

**Observation.**
A maximal plane graph is a plane triangulation.
**Triangulations**

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 destroy 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

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 destroy 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.
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 destroy 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.
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 destroy 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.
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 destroy 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.
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 destroy 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.
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 destroy 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

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 destroy 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.
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 destroy 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.
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 destroy 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, Spaltzeholz 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, Spaltzeholz 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, Spaltzeholz 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 [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, Spaltzeholz 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 ⇒ planar
■ No bends ⇒ straight-line
Motivation

■ Why planar and straight-line?

[Bennett, Ryall, Spaltzeholz and Gooch ’07]
The Aesthetics of Graph Visualization

3.2. Edge Placement Heuristics

By far the most agreed-upon edge placement heuristic is to \textbf{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 \textbf{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 \textit{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.

\textbf{Drawing conventions}

■ No crossings $\Rightarrow$ planar

■ No bends $\Rightarrow$ straight-line

\textbf{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 $\iff$ neither $K_5$ nor $K_{3,3}$ minor of $G$
Towards Straight-Line Drawings

**Theorem.** [Kuratowski 1930]

\[ G \text{ planar } \iff \text{ neither } K_5 \text{ nor } K_{3,3} \text{ minor of } G \]

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

Let \( G \) be a graph with \( n \) vertices. There is an \( O(n) \)-time algorithm to test whether \( G \) is planar.
Towards Straight-Line Drawings

**Theorem.** [Kuratowski 1930]

\[ G \text{ planar } \iff \text{ neither } K_5 \text{ nor } K_{3,3} \text{ minor of } G \]

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

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

Also computes a planar embedding in \( O(n) \).
Towards Straight-Line Drawings

**Theorem.** [Kuratowski 1930]

\[ G \text{ planar } \iff \text{neither } K_5 \text{ nor } K_{3,3} \text{ minor of } G \]

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

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

Also computes a planar embedding in \( O(n) \).

**Theorem.** [Wagner 1936, Fáry 1948, Stein 1951]

Every planar graph has an planar drawing where the edges are straight-line segments.
Towards Straight-Line Drawings

**Theorem.** [Kuratowski 1930]

\[ G \text{ planar } \Leftrightarrow \text{ neither } K_5 \text{ nor } K_{3,3} \text{ minor of } G \]

The algorithms implied by this theory produce drawings with area **not** bounded by any polynomial on \( n \).

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

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

Also computes a planar embedding in \( O(n) \).

**Theorem.** [Wagner 1936, Fáry 1948, Stein 1951]

Every planar graph has an planar drawing where the edges are straight-line segments.
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 a 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.** [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 a 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 singe edge $(v_1, v_2)$. Let this be $G_2$.
- To obtain $G_{i+1}$, add $v_{i+1}$ to $G_i$ so that neighbours of $v_{i+1}$ are on the outer face of $G_i$.
- Neighbours of $v_{i+1}$ in $G_i$ have to form path of length at least two.

**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.** [Schnyder '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

Jonathan Klawitter
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 order $\pi = (v_1, v_2, \ldots, v_n)$ is called a **canonical order**, if the following conditions hold for each $k$, $3 \leq k \leq n$:

1. Vertices $\{v_1, \ldots, v_k\}$ induce a biconnected internally triangulated graph; call it $G_k$.
2. Edge $(v_1, v_2)$ belongs to the outer face of $G_k$.
3. If $k < n$, then vertex $v_{k+1}$ lies in the outer face of $G_k$, and all neighbors of $v_{k+1}$ in $G_k$ appear on the boundary of $G_k$ consecutively.
Definition.
Let $G = (V, E)$ be a triangulated plane graph on $n \geq 3$ vertices. An order $\pi = (v_1, v_2, \ldots, v_n)$ is called a **canonical order**, if the following conditions hold for each $k$, $3 \leq k \leq n$:

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

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

(C3) If $k < n$ then vertex $v_{k+1}$ lies in the outer face of $G_k$, and all neighbors of $v_{k+1}$ in $G_k$ appear on the boundary of $G_k$ consecutively.
Definition.
Let $G = (V, E)$ be a triangulated plane graph on $n \geq 3$ vertices. An order $\pi = (v_1, v_2, \ldots, v_n)$ is called a **canonical order**, if the following conditions hold for each $k$, $3 \leq k \leq n$:

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

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

![Diagram](image)
Canonical Order – Definition

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

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

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

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

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

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

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

(C3) If \(k < n\) then vertex \(v_{k+1}\) lies in the outer face of \(G_k\), and all neighbors of \(v_{k+1}\) in \(G_k\) appear on the boundary of \(G_k\) consecutively.
Canonical Order – Example

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

(C3) If \( k < n \) then vertex \( v_{k+1} \) lies in the outer face of \( G_k \), and all neighbors of \( v_{k+1} \) in \( G_k \) appear on the boundary of \( G_k \) consecutively.
 Canonical Order – Example

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

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

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

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

(C3) If \( k < n \) then vertex \( v_{k+1} \) lies in the outer face of \( G_k \), and all neighbors of \( v_{k+1} \) in \( G_k \) appear on the boundary of \( G_k \) consecutively.
Canonical Order – Example

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

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

(C3) If \( k < n \) then vertex \( v_{k+1} \) lies in the outer face of \( G_k \), and all neighbors of \( v_{k+1} \) in \( G_k \) appear on the boundary of \( G_k \) consecutively.
Canonical Order – Example

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

(C3) If \(k < n\) then vertex \(v_{k+1}\) lies in the outer face of \(G_k\), and all neighbors of \(v_{k+1}\) in \(G_k\) appear on the boundary of \(G_k\) consecutively.
Canonical Order – Example

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

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

(C3) If \( k < n \) then vertex \( v_{k+1} \) lies in the outer face of \( G_k \), and all neighbors of \( v_{k+1} \) in \( G_k \) appear on the boundary of \( G_k \) consecutively.
Canonical Order – Example

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

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

(C3) If \( k < n \) then vertex \( v_{k+1} \) lies in the outer face of \( G_k \), and all neighbors of \( v_{k+1} \) in \( G_k \) appear on the boundary of \( G_k \) consecutively.
Canonical Order – Example

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

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

(C3) If \(k < n\) then vertex \(v_{k+1}\) lies in the outer face of \(G_k\), and all neighbors of \(v_{k+1}\) in \(G_k\) appear on the boundary of \(G_k\) consecutively.
Canonical Order – Example

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

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

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

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

(C3) If \( k < n \) then vertex \( v_{k+1} \) lies in the outer face of \( G_k \), and all neighbors of \( v_{k+1} \) in \( G_k \) appear on the boundary of \( G_k \) consecutively.
Canonical Order – Example

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

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

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

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

(C3) If \( k < n \) then vertex \( v_{k+1} \) lies in the outer face of \( G_k \), and all neighbors of \( v_{k+1} \) in \( G_k \) appear on the boundary of \( G_k \) consecutively.
Canonical Order – 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) \( k < n \) ⇒ \( v_{k+1} \) in outer face of \( G_k \), neighbors of \( v_{k+1} \) in \( G_k \) consecutive on boundary
Canonical Order – Existence

**Lemma.** Every triangulated plane graph has a canonical order.

Base Case:

Induction hypothesis:

Induction step:
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:**
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 $v_{k+1}$ in $G_k$ consecutive on boundary
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:

\[\begin{align*}
\text{(C1)} & \quad G_k \text{ biconnected and internally triangulated} \\
\text{(C2)} & \quad (v_1, v_2) \text{ on outer face of } G_k \\
\text{(C3)} & \quad k < n \Rightarrow v_{k+1} \text{ in outer face of } G_k, \text{ neighbors of } v_{k+1} \text{ in } G_k \text{ consecutive on boundary}
\end{align*}\]
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 $v_{k+1}$ in $G_k$ consecutive on boundary
 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 $v_{k+1}$ in $G_k$ consecutive on boundary

Lemma.
Every triangulated plane graph has a canonical order.
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 $v_{k+1}$ in $G_k$ consecutive on boundary
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 $v_{n-1}, \ldots, v_{k+1}$ 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 $v_{k+1}$ in $G_k$ consecutive on boundary
 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 $v_{n-1}, \ldots, v_{k+1}$ have been chosen such that conditions (C1) – (C3) hold for $k + 1 \leq i \leq n$.

Induction step: Consider $G_k$.

(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 $v_{k+1}$ in $G_k$ consecutive on boundary
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 $v_{n-1}, \ldots, v_{k+1}$ have been chosen such that conditions (C1) – (C3) hold for $k + 1 \leq i \leq n$.

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

(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 $v_{k+1}$ in $G_k$ consecutive on boundary
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 $v_{n-1}, \ldots, v_{k+1}$ have been chosen such that conditions (C1) – (C3) hold for $k + 1 \leq i \leq n$.

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

(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 $v_{k+1}$ in $G_k$ consecutive on boundary
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 \( v_{n-1}, \ldots, v_{k+1} \) have been chosen such that conditions (C1) – (C3) hold for \( k + 1 \leq i \leq n \).

**Induction step:** Consider \( G_k \). We search for \( v_k \).

(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 \( v_{k+1} \) in \( G_k \) consecutive on boundary
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 $v_{n-1}, \ldots, v_{k+1}$ have been chosen such that conditions (C1) – (C3) hold for $k + 1 \leq i \leq n$.

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

(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 $v_{k+1}$ in $G_k$ consecutive on boundary
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 $v_{n-1}, \ldots, v_{k+1}$ have been chosen such that conditions (C1) – (C3) hold for $k + 1 \leq i \leq n$.

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

(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 $v_{k+1}$ in $G_k$ consecutive on boundary
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 $v_{n-1}, \ldots, v_{k+1}$ have been chosen such that conditions (C1) – (C3) hold for $k + 1 \leq i \leq n$.

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

(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 $v_{k+1}$ in $G_k$ consecutive on boundary
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 $v_{n-1}, \ldots, v_{k+1}$ have been chosen such that conditions (C1) – (C3) hold for $k + 1 \leq i \leq n$.

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

(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 $v_{k+1}$ in $G_k$ consecutive on boundary
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 $v_{n-1}, \ldots, v_{k+1}$ have been chosen such that conditions (C1) – (C3) hold for $k + 1 \leq i \leq n$.

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

(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 $v_{k+1}$ in $G_k$ consecutive on boundary
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 $v_{n-1}, \ldots, v_{k+1}$ have been chosen such that conditions (C1) – (C3) hold for $k + 1 \leq i \leq n$.

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

(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 $v_{k+1}$ in $G_k$ consecutive on boundary

Have to show:

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

because $v_k$ incident to a chord

cut vertex

Lemma.
Every triangulated plane graph has a canonical order.
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 $v_{n-1}, \ldots, v_{k+1}$ have been chosen such that conditions (C1) – (C3) hold for $k + 1 \leq i \leq n$.

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

(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 $v_{k+1}$ in $G_k$ consecutive on boundary

**Have to show:**
1. $v_k$ not incident to chord is sufficient

![Diagram](image)
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 $v_{n-1}, \ldots, v_{k+1}$ have been chosen such that conditions (C1) – (C3) hold for $k + 1 \leq i \leq n$.

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

(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 $v_{k+1}$ in $G_k$ consecutive on boundary

Have to show:
1. $v_k$ not incident to chord is sufficient
2. Such $v_k$ exists
Canonical Order – Existence

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

G_k

\( v_k \)

contradiction to edges being consecutive

not triangulated

\( G_{k-1} \)

\( v_1 \) \( v_2 \)
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 $v_k$.

$G_k$ not biconnected

$G_{k-1}$ not triangulated

contradiction to edges being consecutive
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 $v_k$. 

Contradiction to edges being consecutive

Not triangulated

$G_k$ not biconnected

$G_{k-1}$
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 $v_k$.

\begin{align*}
G_k & \quad \text{not biconnected} \\
G_{k-1} & \quad \text{contradiction to edges being consecutive} \\
\end{align*}
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 $v_k$. 

$G_k$ not connected 

contradiction to edges being consecutive

not triangulated

$G_{k-1}$ not biconnected
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 proof of Lemma. □
Canonical Order – Implementation

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

forall $v \in V$ do
  chords($v$) ← 0; out($v$) ← false; mark($v$) ← false
mark($v_1$), mark($v_2$), out($v_1$), out($v_2$), out($v_n$) ← true
for $k = n \ldots 3$ do
  choose $v$ such that mark($v$) = false, out($v$) = true,
  and chords($v$) = 0
  $v_k$ ← $v$; mark($v$) ← true
// Let $w_1 = v_1$, $w_2, \ldots, w_{t-1}, w_t = v_2$ denote the
boundary of $G_{k-1}$ and let $w_p, \ldots, w_q$ be the
unmarked neighbors of $v_k$
out($w_i$) ← true for all $p < i < q$
update number of chords for $w_i$ and its neighbors

forall $v \in V$ do
  chords($v$) ← 0; out($v$) ← false; mark($v$) ← false
mark($v_1$), mark($v_2$), out($v_1$), out($v_2$), out($v_n$) ← true
for $k = n \ldots q$ be the
unmarked neighbors of $v_k$
out($w_i$) ← true for all $p < i < q$
update number of chords for $w_i$ and its neighbors
Canonical Order – Implementation

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

forall \( v \in V \) do
chords\( (v) \) ← 0; out\( (v) \) ← false; mark\( (v) \) ← false
mark\( (v_1) \), mark\( (v_2) \), out\( (v_1) \), out\( (v_2) \), out\( (v_n) \) ← true

for \( k = n \) to 3 do
choose \( v \) such that mark\( (v) \) = false, out\( (v) \) = true, and chords\( (v) \) = 0
\( v_k \) ← \( v \); mark\( (v) \) ← true

// Let \( w_1 = v_1, w_2, ..., w_{t-1}, w_t = v_2 \) denote the boundary of \( G_{k-1} \) and let \( w_p, ..., w_q \) be the unmarked neighbors of \( v_k \)
out\( (w_i) \) ← true for all \( p < i < q \)
update number of chords for \( w_i \) and its neighbors
Canonical Order – Implementation

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

forall \(v \in V\) do

outer face

forall \(v \in V\) do

\[
\text{chords}(v) \leftarrow 0; \quad \text{out}(v) \leftarrow false; \quad \text{mark}(v) \leftarrow false
\]

\[
\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow true
\]

for \(k = n \ldots 3\) do

choose \(v\) such that \(\text{mark}(v) = false, \text{out}(v) = true,\) and \(\text{chords}(v) = 0\)

\[
v_k \leftarrow v; \quad \text{mark}(v) \leftarrow true
\]

// Let \(w_1, w_2, \ldots, w_{t-1}, w_t = v_2\) denote the boundary of \(G_{k-1}\) and let \(w_p, \ldots, w_q\) be the unmarked neighbors of \(v_k\)

\[
\text{out}(w_i) \leftarrow true \text{ for all } p<i<q
\]

update number of chords for \(w_i\) and its neighbors
Canonical Order – Implementation

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

\[ \textbf{forall } v \in V \textbf{ do} \]
\[ \quad \text{chords}(v) \leftarrow 0; \]
Canonical Order – Implementation

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

\[
\text{forall } v \in V \text{ do}
\]
\[
\text{chords}(v) \leftarrow 0;
\]
Canonical Order – Implementation

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

\[
\text{forall } v \in V \text{ do}
\]
\[
\text{chords}(v) \leftarrow 0; \text{ out}(v) \leftarrow \text{false};
\]
Canonical Order – Implementation

**CanonicalOrder**

\[
G = (V, E), (v_1, v_2, v_n)
\]

**forall** \( v \in V \) do

\[\text{chords}(v) \leftarrow 0; \text{out}(v) \leftarrow \text{false};\]

- **chord**(
  - \# chords adjacent to \( v \)
)
- **out**(
  - true iff \( v \) is currently outer vertex
)

outer face
Canonical Order – Implementation

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

\[\forall v \in V \text{ do} \]
\[\text{chords}(v) \leftarrow 0; \text{out}(v) \leftarrow \text{false}; \text{mark}(v) \leftarrow \text{false}\]
Canonical Order – Implementation

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

forall \(v \in V\) do

\[\text{chords}(v) \leftarrow 0; \ \text{out}(v) \leftarrow \text{false}; \ \text{mark}(v) \leftarrow \text{false}\]

- **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

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

\[
\text{for all } v \in V \text{ do }
\begin{align*}
\text{chords}(v) &\leftarrow 0; \text{out}(v) \leftarrow \text{false}; \text{mark}(v) \leftarrow \text{false} \\
\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) &\leftarrow \text{true}
\end{align*}
\]

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

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

\(\text{forall } v \in V \text{ do} \)
\(\quad \text{chords}(v) \leftarrow 0; \text{out}(v) \leftarrow \text{false}; \text{mark}(v) \leftarrow \text{false}\)
\(\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow \text{true}\)
\(\text{for } k = n \text{ to } 3 \text{ do}\)

\(\quad\)

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

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

forall \( v \in V \) do
chords\( (v) \) ← 0; out\( (v) \) ← false; mark\( (v) \) ← false
mark\( (v_1) \), mark\( (v_2) \), out\( (v_1) \), out\( (v_2) \), out\( (v_n) \) ← true
for \( k = n \) to 3 do
choose \( v \) such that mark\( (v) \) = false, out\( (v) \) = true, and chords\( (v) \) = 0

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

outer face

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

outer face
Canonical Order – Implementation

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

\[
\text{forall } v \in V \text{ do}
\]
\[
\text{chords}(v) \leftarrow 0; \text{out}(v) \leftarrow \text{false}; \text{mark}(v) \leftarrow \text{false}
\]
\[
\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow \text{true}
\]

\[
\text{for } k = n \text{ to } 3 \text{ do}
\]
\[
\text{choose } v \text{ such that mark}(v) = \text{false}, \text{out}(v) = \text{true}, \text{and chords}(v) = 0
\]

- **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

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

forall $v \in V$ do
  chords($v$) ← 0; out($v$) ← false; mark($v$) ← false
  mark($v_1$), mark($v_2$), out($v_1$), out($v_2$), out($v_n$) ← true

for $k = n$ to 3 do
  choose $v$ such that mark($v$) = false, out($v$) = true, and chords($v$) = 0

- 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

```
CanonicalOrder(G = (V, E), (v1, v2, vn))
for all v ∈ V do
    chords(v) ← 0; out(v) ← false; mark(v) ← false
mark(v1), mark(v2), out(v1), out(v2), out(vn) ← true
for k = n to 3 do
    choose v such that mark(v) = false, out(v) = true, and chords(v) = 0
    vk ← v; mark(v) ← 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

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

\[
\begin{align*}
\text{forall } v \in V & \text{ do} \\
\text{chords}(v) & \leftarrow 0; \text{out}(v) \leftarrow \text{false}; \text{mark}(v) \leftarrow \text{false} \\
\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) & \leftarrow \text{true} \\
\text{for } k = n \text{ to } 3 & \text{ do} \\
\text{choose } v & \text{ such that mark}(v) = \text{false}, \text{out}(v) = \text{true}, \\
\text{and chords}(v) = 0 \\
v_k & \leftarrow v; \text{mark}(v) \leftarrow \text{true} \\
& // \text{Let } w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2 \text{ denote the boundary of } G_{k-1}
\end{align*}
\]
Canonical Order – Implementation

CanonicalOrder(G = (V, E), (v₁, v₂, vₙ))

forall v ∈ V do
  chords(v) ← 0; out(v) ← false; mark(v) ← false
  mark(v₁), mark(v₂), out(v₁), out(v₂), out(vₙ) ← true
for k = n to 3 do
  choose v such that mark(v) = false, out(v) = true, and chords(v) = 0
  vᵥk ← v; mark(v) ← true
  // Let w₁ = v₁, w₂, ..., wᵗ₋₁, wᵗ = v₂ denote the boundary of Gᵏ₋₁

- 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

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

forall \(v \in V\) do
- chords\((v)\) ← 0; out\((v)\) ← false; mark\((v)\) ← false
- mark\((v_1)\), mark\((v_2)\), out\((v_1)\), out\((v_2)\), out\((v_n)\) ← true

for \(k = n\) to 3 do
  choose \(v\) such that mark\((v)\) = false, out\((v)\) = true, and chords\((v)\) = 0
  \(v_k \leftarrow v\); mark\((v)\) ← true

// Let \(w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2\) denote the boundary of \(G_{k-1}\) and let \(w_p, \ldots, w_q\) be the unmarked neighbors of \(v_k\)

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

\(v_k\)
Canonical Order – Implementation

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

\[
\begin{align*}
\forall v \in V & \quad \text{do} \\
\quad \text{chords}(v) \gets 0; \quad \text{out}(v) \gets \text{false}; \quad \text{mark}(v) \gets \text{false} \\
\quad \text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \gets \text{true} \\
\text{for } k = n \text{ to } 3 \text{ do} \\
\quad \text{choose } v \text{ such that } \text{mark}(v) = \text{false}, \text{out}(v) = \text{true}, \text{and } \text{chords}(v) = 0 \\
\quad v_k \gets v; \quad \text{mark}(v) \gets \text{true} \\
\quad \text{// Let } w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2 \text{ denote the boundary of } G_{k-1} \text{ and let } w_p, \ldots, w_q \text{ be the unmarked neighbors of } v_k
\end{align*}
\]

- **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

CanonicalOrder(G = (V, E), (v1, v2, vn))

forall v ∈ V do
  chords(v) ← 0; out(v) ← false; mark(v) ← false
mark(v1), mark(v2), out(v1), out(v2), out(vn) ← true
for k = n to 3 do
  choose v such that mark(v) = false, out(v) = true, and chords(v) = 0
  vk ← v; mark(v) ← true
  // Let w1 = v1, w2, ..., wt−1, wt = v2 denote the boundary of Gk−1 and let wp, ..., wq be the unmarked neighbors of vk
  out(wi) ← true for all p < i < q

- 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

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

\[
\text{forall } v \in V \text{ do}
\]
\[
\begin{align*}
\text{chords}(v) & \leftarrow 0; \text{out}(v) \leftarrow \text{false; mark}(v) \leftarrow \text{false} \\
\text{mark}(v_1), \text{mark}(v_2), \text{out}(v_1), \text{out}(v_2), \text{out}(v_n) & \leftarrow \text{true}
\end{align*}
\]

\[
\text{for } k = n \text{ to } 3 \text{ do}
\]
\[
\begin{align*}
\text{choose } v \text{ such that mark}(v) = \text{false, out}(v) = \text{true, and chords}(v) = 0} \\
v_k & \leftarrow v; \text{mark}(v) \leftarrow \text{true}
\end{align*}
\]

// Let \( w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2 \) denote the boundary of \( G_{k-1} \) and let \( w_p, \ldots, w_q \) be the unmarked neighbors of \( v_k \)
\[
\text{out}(w_i) \leftarrow \text{true for all } p < i < q
\]
update number of chords for \( w_i \) and its neighbours

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

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

forall $v \in V$ do
  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
for $k = n$ to 3 do
  choose $v$ such that mark($v$) = false, out($v$) = true, and chords($v$) = 0
  $v_k \leftarrow v$; mark($v$) $\leftarrow$ true
  // Let $w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2$ denote the boundary of $G_{k-1}$ and let $w_p, \ldots, w_q$ be the unmarked neighbors of $v_k$
  out($w_i$) $\leftarrow$ true for all $p < i < q$
  update number of chords for $w_i$ and its neighbours

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

Lemma.
Algorithm CanonicalOrder computes a canonical order of a plane graph in $O(n)$ time.
**Canonical Order – Implementation**

```
CanonicalOrder(G = (V, E), (v1, v2, vn))

forall v ∈ V do
  chords(v) ← 0; out(v) ← false; mark(v) ← false
mark(v1), mark(v2), out(v1), out(v2), out(vn) ← true
for k = n to 3 do
  choose v such that mark(v) = false, out(v) = true, and chords(v) = 0
  vk ← v; mark(v) ← true
  // Let w1 = v1, w2, ..., wt−1, wt = v2 denote the boundary of Gk−1 and let wp, ..., wq be the unmarked neighbors of vk
  out(wi) ← true for all p < i < q
  update number of chords for wi and its neighbours
```

**Lemma.**
Algorithm CanonicalOrder computes a canonical order of a plane graph in $O(n)$ time.

- **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

```
CanonicalOrder(G = (V, E), (v1, v2, vn))

forall v ∈ V do
  chords(v) ← 0; out(v) ← false; mark(v) ← false
mark(v1), mark(v2), out(v1), out(v2), out(vn) ← true
for k = n to 3 do
  choose v such that mark(v) = false, out(v) = true,
  and chords(v) = 0  // keep list with candidates
  vk ← v; mark(v) ← true
  // Let w1 = v1, w2, ..., wt-1, wt = v2 denote the
  boundary of Gk-1 and let w_p, ..., w_q be the
  unmarked neighbors of vk
  out(w_i) ← true for all p < i < q  // O(n) in total
  update number of chords for w_i
  and its neighbours
```

**Lemma.**
Algorithm CanonicalOrder computes a canonical order of a plane graph in $O(n)$ time.
Canonical Order – Implementation

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

forall $v \in V$ do
  chords($v$) ← 0; out($v$) ← false; mark($v$) ← false
  mark($v_1$), mark($v_2$), out($v_1$), out($v_2$), out($v_n$) ← true

for $k = n$ to 3 do
  choose $v$ such that mark($v$) = false, out($v$) = true,
  and chords($v$) = 0  // keep list with candidates
  $v_k ← v$; mark($v$) ← true
  // Let $w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2$ denote the
  // boundary of $G_{k-1}$ and let $w_p, \ldots, w_q$ be the
  // unmarked neighbors of $v_k$
  out($w_i$) ← true for all $p < i < q$  // $O(n)$ in total
  update number of chords for $w_i$
  and its neighbours  // $O(m) = O(n)$ in total

**Lemma.**
Algorithm CanonicalOrder computes a canonical order of a plane graph in $O(n)$ time.

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

Lecture 3:
Straight-Line Drawings of Planar Graphs I:
Canonical Ordering and Shift Method

Part III:
Shift Method

Jonathan Klawitter
Shift Method – Idea

**Drawing invariants:**

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

- $v_1$ is on $(0,0)$, $v_2$ is on $(2k-6,0)$,
- 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 $\pm 1$. 

$G_{k-1}$
Shift Method – Idea

**Drawing invariants:**

\( G_{k-1} \) is drawn such that

- \( v_1 \) is on \((0, 0)\), \( v_2 \) is on \((2k - 6, 0)\),
- 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 on $(0, 0)$, $v_2$ is on $(2k - 6, 0)$,
  - 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 on $(0, 0)$, $v_2$ is on $(2k - 6, 0)$,
- 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 $\pm 1$. 

![Diagram of $G_{k-1}$ with vertices $v_1$ and $v_2$.]
Shift Method – Idea

**Drawing invariants:**

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

- $v_1$ is on $(0, 0)$, $v_2$ is on $(2k - 6, 0)$,
- 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 on $(0, 0)$, $v_2$ is on $(2k - 6, 0)$,
- 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 $\pm 1$. 

![Diagram of graph $G_{k-1}$ with vertices $v_1$, $v_2$, and edges connecting them, along with a vertex $u_k$ and edges connecting $u_k$ to other vertices. The graph is drawn in a way that demonstrates the drawing invariants mentioned.](image-url)
Shift Method – Idea

**Drawing invariants:**

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

- $v_1$ is on $(0, 0)$, $v_2$ is on $(2k - 6, 0)$,
- 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 $\pm 1$. 

![Diagram showing the drawing invariants for $G_{k-1}$ with $v_1$, $v_2$, $w_p$, $w_q$, and $v_k$ marked, along with an indication of an overlap between $G_{k-1}$ and another shape.]
Shift Method – Idea

Drawing invariants:

$G_{k-1}$ is drawn such that
- $v_1$ is on $(0, 0)$, $v_2$ is on $(2k - 6, 0)$,
- 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 $\pm 1$.

Overlaps!

What could be the solution?
Shift Method – Idea

**Drawing invariants:**

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

- $v_1$ is on $(0,0)$, $v_2$ is on $(2k-6,0)$,
- 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 on $(0,0)$, $v_2$ is on $(2k-6,0)$,
- 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 on \((0, 0)\), \( v_2 \) is on \((2k - 6, 0)\),
- 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 on $(0, 0)$, $v_2$ is on $(2k - 6, 0)$,
- 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 $\pm 1$. 

![Diagram of $G_{k-1}$ with vertices $v_1$, $v_2$, $v_k$, $w_p$, and $w_q$. The boundary of $G_{k-1}$ is drawn $x$-monotone, and each edge of the boundary is drawn with slopes $\pm 1$.](image-url)
Shift Method – Idea

**Drawing invariants:**

\( G_{k-1} \) is drawn such that

- \( v_1 \) is on \((0,0)\), \( v_2 \) is on \((2k-6,0)\),
- 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 \( \pm 1 \).

Does \( v_k \) land on grid?
Shift Method – Idea

**Drawing invariants:**

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

- $v_1$ is on $(0, 0)$, $v_2$ is on $(2k - 6, 0)$,
- 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 $\pm 1$.

Does $v_k$ land on grid?
Shift Method – Idea

**Drawing invariants:**

- $G_{k-1}$ is drawn such that
  - $v_1$ is on $(0, 0)$, $v_2$ is on $(2k - 6, 0)$,
  - 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 $\pm 1$. 

Does $v_k$ land on grid?
Shift Method – Idea

**Drawing invariants:**

- $G_{k-1}$ is drawn such that:
  - $v_1$ is on $(0,0)$, $v_2$ is on $(2k-6,0)$,
  - 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 $\pm 1$.

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

Does $v_k$ land on grid?
Shift Method – Idea

**Drawing invariants:**

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

- $v_1$ is on $(0, 0)$, $v_2$ is on $(2k - 6, 0)$,
- 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 $\pm 1$.

Does $v_k$ land on grid?
Shift Method – Idea

**Drawing invariants:**

- $G_{k-1}$ is drawn such that
  - $v_1$ is on $(0,0)$, $v_2$ is on $(2k-6,0)$,
  - 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 $\pm 1$.

Does $v_k$ land on grid?

![Diagram](image)
Shift Method – Idea

**Drawing invariants:**

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

- $v_1$ is on $(0, 0)$, $v_2$ is on $(2k - 6, 0)$,
- 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 $\pm 1$.

Does $v_k$ land on grid?

yes, because $w_p$ and $w_q$ have even Manhattan distance.
Shift Method – Idea

**Drawing invariants:**

\(G_{k-1}\) is drawn such that

- \(v_1\) is on \((0,0)\), \(v_2\) is on \((2k-6,0)\),
- 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 \(\pm 1\).

Does \(v_k\) land on grid?

yes, because \(w_p\) and \(w_q\) have even Manhattan distance

\(G_{k-1}\)
Shift Method – Example
Shift Method – Example
Shift Method – Example
Shift Method – Example

![Diagram showing a shift method example](image-url)
Shift Method – Example
Shift Method – Example
Shift Method – Example
Shift Method – Example
Shift Method – Example
Shift Method – Example

[Diagram showing a shift method example with numbers and arrows indicating movement.]
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

$L(10)$
Shift Method – Example

$L(11)$
Shift Method – Example
Shift Method – Example
Shift Method – Example

$L(13)$
Shift Method – Example

\[ L(14) \]
Shift Method – Example
Shift Method – Example
Shift Method – Example

\[(0, 0) \rightarrow (2n - 4, 0)\]
Shift Method – Planarity

$G_{k-1}$
Shift Method – Planarity

$G_{k-1}$
Shift Method – Planarity
Shift Method – Planarity

$G_{k-1}$

covered vertices

$w_1 \rightarrow w_2 \rightarrow \ldots \rightarrow w_{t-1} \rightarrow w_t$
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 $G_i$, $1 \leq i \leq n - 1$. 

![Diagram of graph with vertices and edges labeled as $w_1$, $w_2$, $v_k$, $w_p$, $w_q$, $G_{k-1}$, $w_{t-1}$, and $w_t$.]
Shift Method – Planarity

**Observations.**

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

![Diagram of graph](image)
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 $0 < \delta_1 \leq \delta_2 \leq \cdots \leq \delta_t \in \mathbb{N}$, 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 $0 < \delta_1 \leq \delta_2 \leq \cdots \leq \delta_t \in \mathbb{N}$, 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 $0 < \delta_1 \leq \delta_2 \leq \cdots \leq \delta_t \in \mathbb{N}$, 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 $G_k$. 

![Diagram of graph and vertices](Image)

$G_{k-1}$

$w_1$ $w_2$ $w_p$ $w_q$ $v_k$ $w_t$ $w_{t-1}$ $L(w_i)$
Let $v_1, \ldots, v_n$ be a canonical order of $G$

for $i = 1$ to $3$ do

for $i = 4$ to $n$ do
Shift Method – Pseudocode

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

for $i = 1$ to $3$ do

\[ L(v_i) \leftarrow \{v_i\} \]

for $i = 4$ to $n$ do

\[ L(v_i) \leftarrow \bigcup_{j=p+1}^{q-1} L(w_j) \cup \{v_i\} \]
Let $v_1, \ldots, v_n$ 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

\[
L(v_i) \leftarrow \bigcup_{j=p+1}^{q-1} L(w_j) \cup \{v_i\}
\]
Let $v_1, \ldots, v_n$ be a canonical order of $G$

for $i = 1$ to $3$

1. $L(v_i) \leftarrow \{v_i\}$
2. $P(v_1) \leftarrow (0,0); P(v_2) \leftarrow (2,0), P(v_3) \leftarrow (1,1)$

for $i = 4$ to $n$

1. Let $w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2$
2. denote the boundary of $G_{i-1}$
3. and let $w_p, \ldots, w_q$ be the neighbours of $v_i$
Let $v_1, \ldots, v_n$ 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 $w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2$

denote the boundary of $G_{i-1}$

and let $w_p, \ldots, w_q$ be the neighbours of $v_i$
Let $v_1, \ldots, v_n$ 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 $w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2$

denote the boundary of $G_{i-1}$

and let $w_p, \ldots, w_q$ be the neighbours of $v_i$

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

\[ x(v) \leftarrow x(v) + 2 \]

$P(v_i) \leftarrow \text{intersection of } +1/-1 \text{ diagonals through } P(w_p) \text{ and } P(w_q)$

$L(v_i) \leftarrow \bigcup_{j=p+1}^{q-1} L(w_j) \cup \{v_i\}$
Let \( v_1, \ldots, v_n \) 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 \( w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2 \)
denote the boundary of \( G_{k-1} \) and let \( w_p, \ldots, w_q \) be the neighbours of \( v_i \)

for \( \forall v \in \bigcup_{j=p+1}^{q-1} L(w_j) \) do

\[ x(v) \leftarrow x(v) + 1 \]
Let $v_1, \ldots, v_n$ 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 $w_1 = v_1$, $w_2, \ldots, w_{t-1}$, $w_t = v_2$
  denote the boundary of $G_{i-1}$
  and let $w_p, \ldots, w_q$ be the neighbours of $v_i$
  for $\forall v \in \bigcup_{j=p+1}^{q-1} L(w_j)$ do
    $x(v) \leftarrow x(v) + 2$
  for $\forall v \in \bigcup_{j=q}^{t} L(w_j)$ do
    $x(v) \leftarrow x(v) + 1$

Let $v_1, \ldots, v_n$ 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 $w_1 = v_1$, $w_2, \ldots, w_{t-1}$, $w_t = v_2$
  denote the boundary of $G_{i-1}$
  and let $w_p, \ldots, w_q$ be the neighbours of $v_i$
  for $\forall v \in \bigcup_{j=p+1}^{q-1} L(w_j)$ do
    $x(v) \leftarrow x(v) + 2$
  for $\forall v \in \bigcup_{j=q}^{t} L(w_j)$ do
    $x(v) \leftarrow x(v) + 1$
Shift Method – Pseudocode

Let $v_1, \ldots, v_n$ 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 $w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2$
  denote the boundary of $G_{i-1}$
  and let $w_p, \ldots, w_q$ be the neighbours of $v_i$
  for $\forall v \in \bigcup_{j=p+1}^{q-1} L(w_j)$ do
    $x(v) \leftarrow x(v) + 1$
  for $\forall v \in \bigcup_{j=q}^{t} L(w_j)$ do
    $x(v) \leftarrow x(v) + 2$
Let $v_1, \ldots, v_n$ 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 $w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2$
denote the boundary of $G_{i-1}$
and let $w_p, \ldots, w_q$ be the neighbours of $v_i$

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

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

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

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

$P(v_i) \leftarrow$ intersection of $+1/-1$ diagonals
through $P(w_p)$ and $P(w_q)$
Let $v_1, \ldots, v_n$ be a canonical order of $G$

for $i = 1$ to $3$ do

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

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

for $i = 4$ to $n$ do

Let $w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2$
denote the boundary of $G_{i-1}$
and let $w_p, \ldots, w_q$ be the neighbours of $v_i$

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

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

for $\forall v \in \bigcup_{j=q}^{t} \text{L}(w_j)$ do

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

$P(v_i) \leftarrow \text{intersection of } +1/-1 \text{ diagonals}$
through $P(w_p)$ and $P(w_q)$

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

Let \( v_1, \ldots, v_n \) 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 \( w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2 \)

denote the boundary of \( G_{i-1} \)

and let \( w_p, \ldots, w_q \) be the neighbours of \( v_i \)

for \( \forall v \in \bigcup_{j=p+1}^{q-1} L(w_j) \) do

\[
x(v) \leftarrow x(v) + 1
\]

for \( \forall v \in \bigcup_{j=q}^{t} L(w_j) \) do

\[
x(v) \leftarrow x(v) + 2
\]

\[
P(v_i) \leftarrow \text{intersection of } +1/-1 \text{ diagonals through } P(w_p) \text{ and } P(w_q)
\]

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

Running Time?
Shift Method – Pseudocode

Let $v_1, \ldots, v_n$ 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 $w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2$ denote the boundary of $G_{i-1}$ and let $w_p, \ldots, w_q$ be the neighbours of $v_i$
  for $\forall v \in \bigcup_{j=p+1}^{q-1} L(w_j)$ do // $O(n^2)$ in total
    $x(v) \leftarrow x(v) + 1$
  for $\forall v \in \bigcup_{j=q}^{t} L(w_j)$ do // $O(n^2)$ in total
    $x(v) \leftarrow x(v) + 2$
  $P(v_i) \leftarrow$ intersection of $+1/-1$ 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
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 $x(w_q) - x(w_p)$.
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 \( x(w_q) - x(w_p) \)

\[
\begin{align*}
(1) \quad x(v_k) &= \frac{1}{2} (x(w_q) + x(w_p) + y(w_q) - y(w_p)) \\
(2) \quad y(v_k) &= \frac{1}{2} (x(w_q) - x(w_p) + y(w_q) + y(w_p)) \\
(3) \quad x(v_k) - x(w_p) &= \frac{1}{2} (x(w_q) - x(w_p) + y(w_q) - y(w_p))
\end{align*}
\]
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 $x(w_q) - x(w_p)$.

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

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

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

\[
\begin{align*}
(1) \quad x(v_k) &= \frac{1}{2} (x(w_q) + x(w_p) + y(w_q) - y(w_p)) \\
(2) \quad y(v_k) &= \frac{1}{2} (x(w_q) - x(w_p) + y(w_q) + y(w_p)) \\
(3) \quad x(v_k) - x(w_p) &= \frac{1}{2} (x(w_q) - x(w_p) + y(w_q) - y(w_p))
\end{align*}
\]
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 x distance for \(v_n\) computed, use preorder
traversal to compute all x-coordinates.

\[
\begin{align*}
(1) \quad x(v_k) &= \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p)) \\
(2) \quad y(v_k) &= \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p)) \\
(3) \quad x(v_k) - x(w_p) &= \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))
\end{align*}
\]
Shift Method – Linear Time Implementation

**Relative x distance tree.**
For each vertex $v$ store
- $x$-offset $\Delta x(v)$ from parent
- $y$-coordinate $y(v)$

\[
\begin{align*}
(1) \quad x(v_k) &= \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p)) \\
(2) \quad y(v_k) &= \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p)) \\
(3) \quad x(v_k) - x(w_p) &= \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))
\end{align*}
\]
Shift Method – Linear Time Implementation

Relative x distance tree.
For each vertex $v$ store
- x-offset $\Delta_x(v)$ from parent
- y-coordinate $y(v)$

\begin{align*}
(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{align*}
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 $y(v)$

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

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

$3) \quad x(v_k) - x(w_p) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))$
Shift Method – Linear Time Implementation

Relative x distance tree.
For each vertex \( v \) store
- x-offset \( \Delta_x(v) \) from parent
- y-coordinate \( y(v) \)

\[
\text{(1)} \quad x(v_k) = \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p))
\]
\[
\text{(2)} \quad y(v_k) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p))
\]
\[
\text{(3)} \quad x(v_k) - x(w_p) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))
\]
Shift Method – Linear Time Implementation

Relative x distance tree.
For each vertex \( v \) store
- x-offset \( \Delta_x(v) \) from parent
- y-coordinate \( y(v) \)

\[
\begin{align*}
(1) \quad x(v_k) &= \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p)) \\
(2) \quad y(v_k) &= \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p)) \\
(3) \quad x(v_k) - x(w_p) &= \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))
\end{align*}
\]
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.
- $\Delta x(w_{p+1})^{++}$, $\Delta x(w_q)^{++}$

\[
\begin{align*}
(1) \quad x(v_k) &= \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p)) \\
(2) \quad y(v_k) &= \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p)) \\
(3) \quad x(v_k) - x(w_p) &= \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))
\end{align*}
\]
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.
- \( \Delta_x(w_{p+1})++ \), \( \Delta_x(w_q)++ \)
- \( \Delta_x(w_p, w_q) = \Delta_x(w_{p+1}) + \ldots + \Delta_x(w_q) \)

(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.
- $\Delta x(w_{p+1})$, $\Delta x(w_q)$
- $\Delta x(w_p, w_q) = \Delta x(w_{p+1}) + \ldots + \Delta x(w_q)$
- $\Delta x(v_k)$ by (3)

\[
(1) \quad x(v_k) = \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p))
\]
\[
(2) \quad y(v_k) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p))
\]
\[
(3) \quad x(v_k) - x(w_p) = \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))
\]
Shift Method – Linear Time Implementation

Relative x distance tree.
For each vertex $v$ store
- x-offset $\Delta x(v)$ from parent
- y-coordinate $y(v)$

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

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

\[
\begin{align*}
(1) \quad x(v_k) &= \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p)) \\
(2) \quad y(v_k) &= \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p)) \\
(3) \quad x(v_k) - x(w_p) &= \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))
\end{align*}
\]
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.
- $\Delta x(w_{p+1})$, $\Delta x(w_q)$
- $\Delta x(w_p, w_q) = \Delta x(w_{p+1}) + \ldots + \Delta x(w_q)$
- $\Delta x(v_k)$ 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.

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

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