MPRI 2-38-1: Algorithms and combinatorics for geometric graphs

Lecture 4

Planar straight-line grid drawings

Chapter I: FPP algorithm (and canonical orderings)

october 9, 2024

Luca Castelli Aleardi

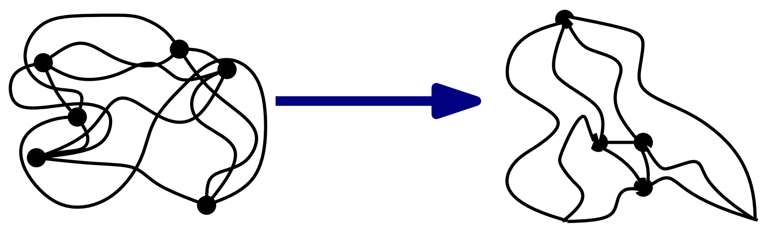
(some slides are provided by Eric Fusy)

Straight-line planar drawings of planar graphs

Problem definition (Planarity testing, Embedding a planar graph)

Input: a planar graph

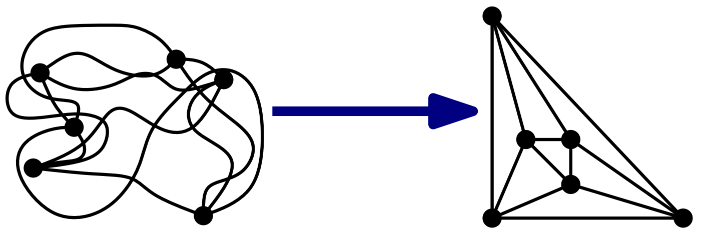
Output: the planar map (cellulaly embedded graph)



Problem definition (drawing in the plane)

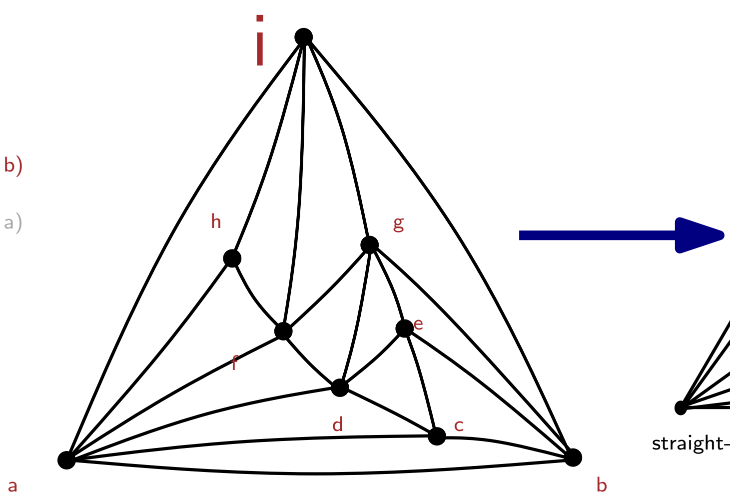
Input: a planar map

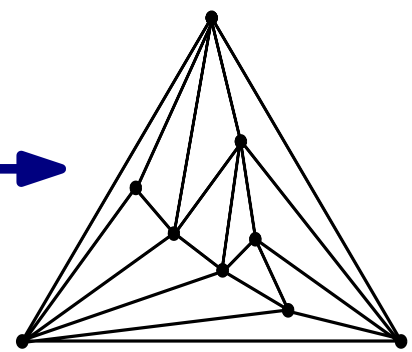
Output: a straight-line planar drawing (crossing-free)



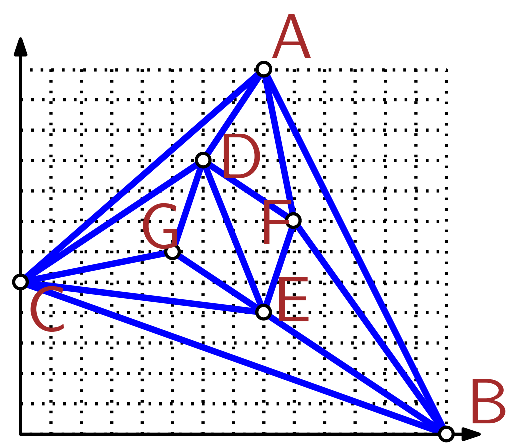
Input of the problem: planar map

(a, b, c)	(d, e, g)	(i, g, b)
(a, c, d)	(e, b, g)	(i, b, a)
(d, c, e)	(a, f, h)	
(c, b, e)	(a, h, i)	
(a, d, f)	(i, h, f)	
(f, d, g)	(i, f, g)	



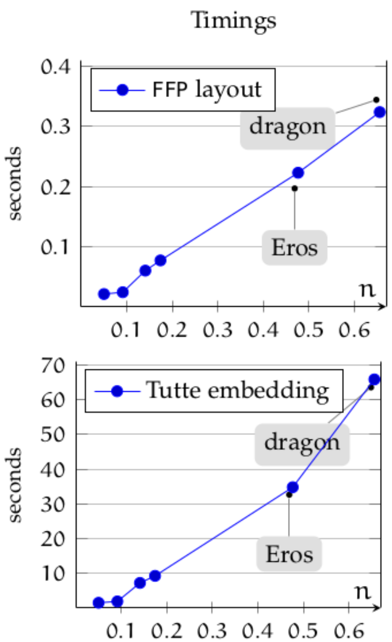


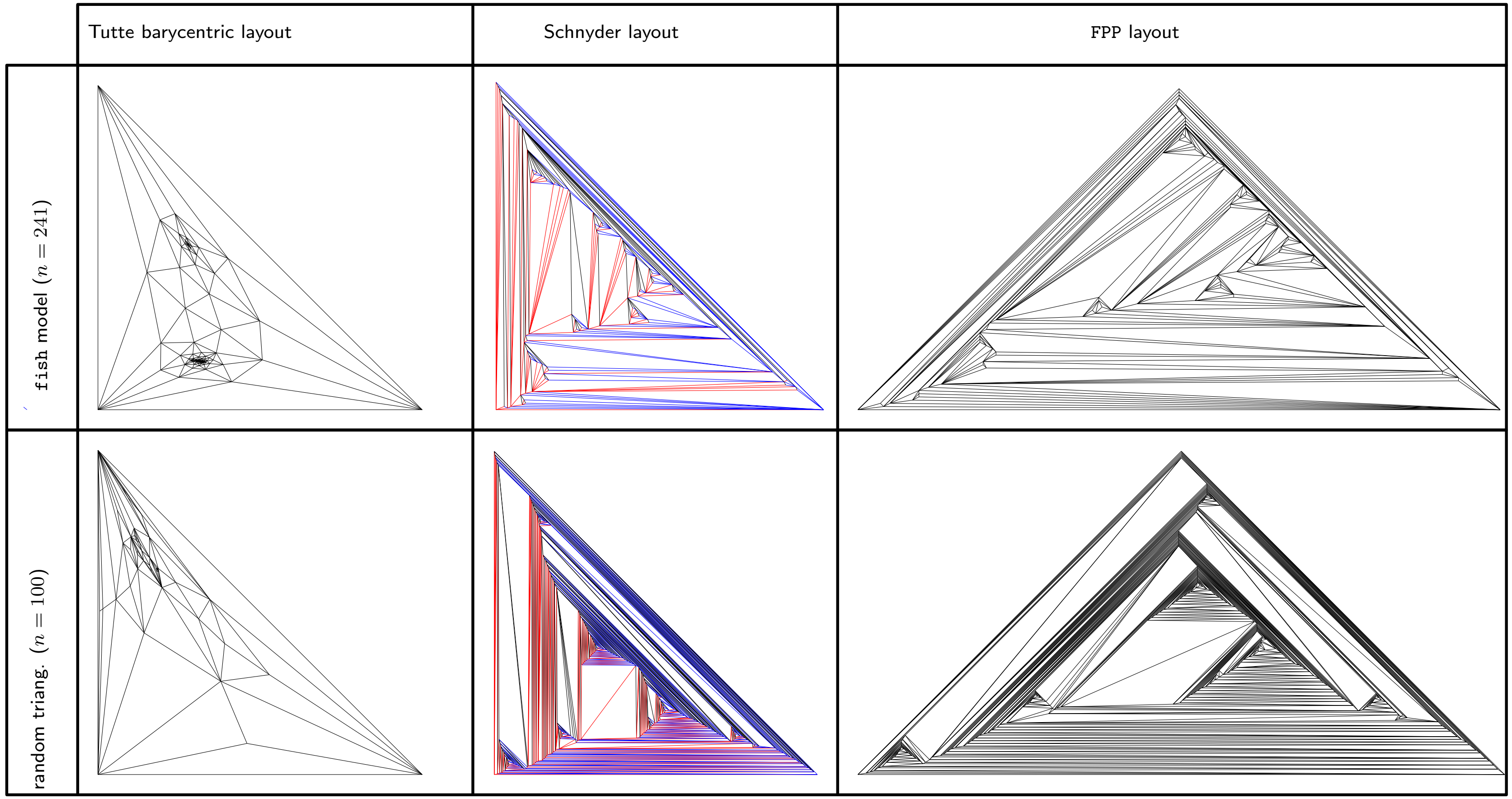
straight-line crossing-free drawing



straight-line grid drawing

Spring embedding vs. grid drawing





Canonical orderings

(the definition)

Canonical orderings: definition

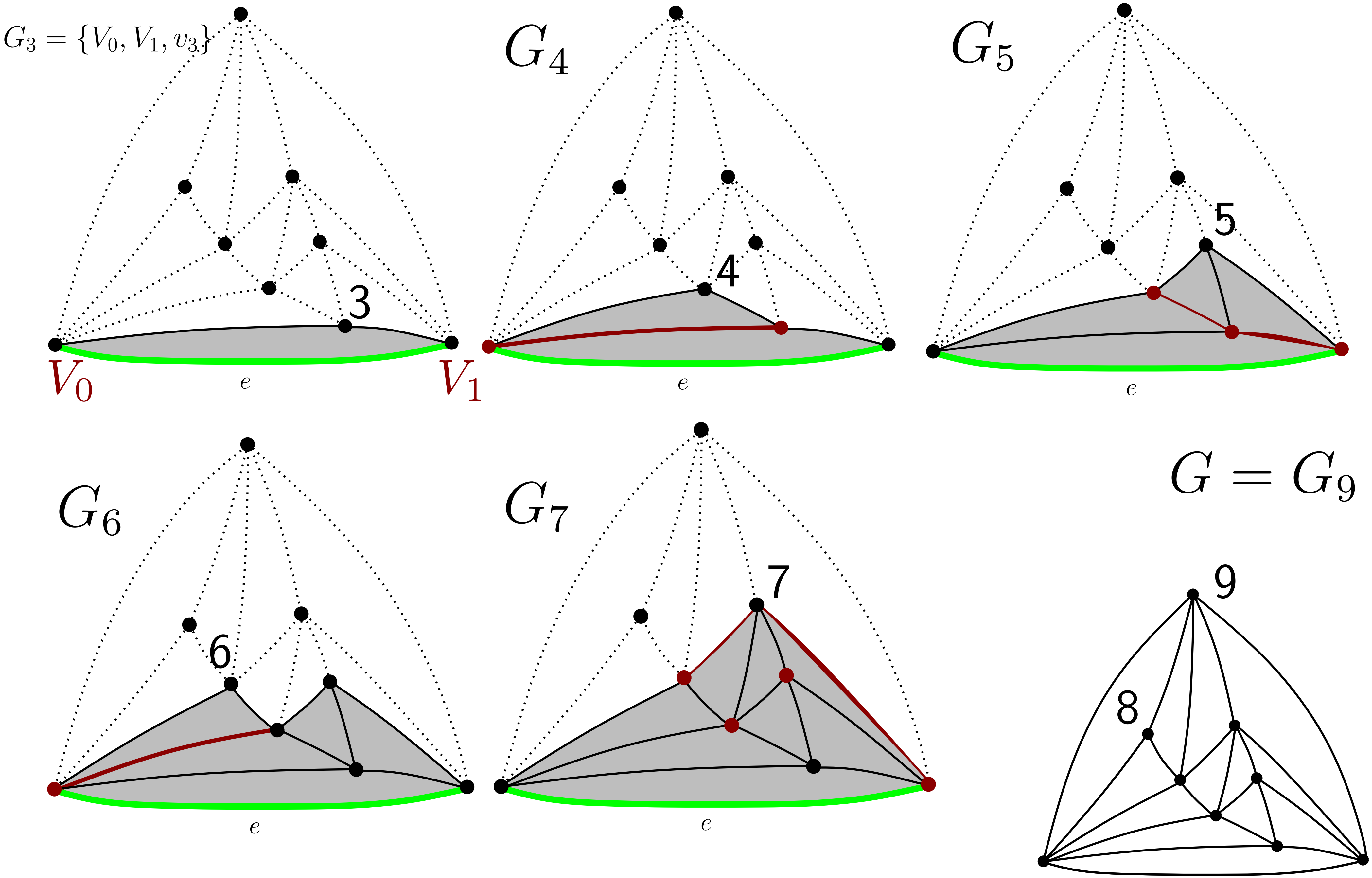
[de Fraysseix Pach Pollack]

Definition 2.6 ([FPP90]) *Let* $\mathcal{T}$ *be a plane triangulation, whose vertices on the outer (root) face are denoted* V_0, V_1, V_2. *An ordering* $\pi = \{v_1, v_2, \ldots, v_n\}$ *of the* n *vertices of* $\mathcal{T}$ *is called a* canonical ordering *if the subgraphs* G_k $(3 \leqslant k \leqslant n)$ *induced by the vertices* $v_1, \ldots, v_k$ *satisfy the following conditions (where we denote by* B_k *the cycle bounding the outer face of* G_k*):*

- G_k *is 2-connected and internally triangulated, and* $G_n = \mathcal{T}$*;*
- v_1 *and* v_2 *belong to the outer face* (V_0, V_1, V_2)*;*
- *for each* $k \geqslant 3$ *the vertex* v_k *is on the* B_k *and its neighbors in* G_{k-1} *are consecutive on* B_{k-1}.

Canonical orderings: definition

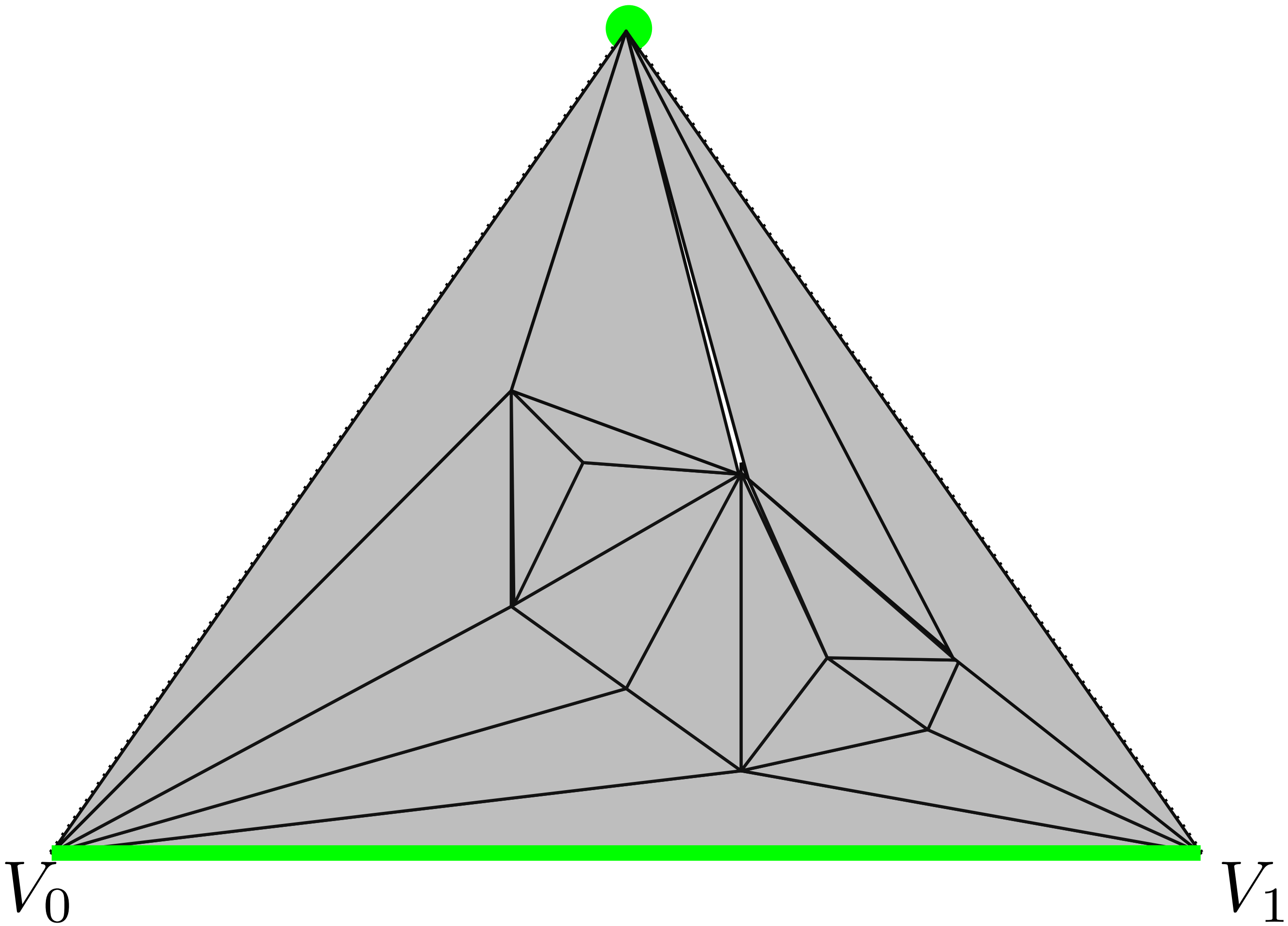
[de Fraysseix Pach Pollack]



Canonical orderings: existence

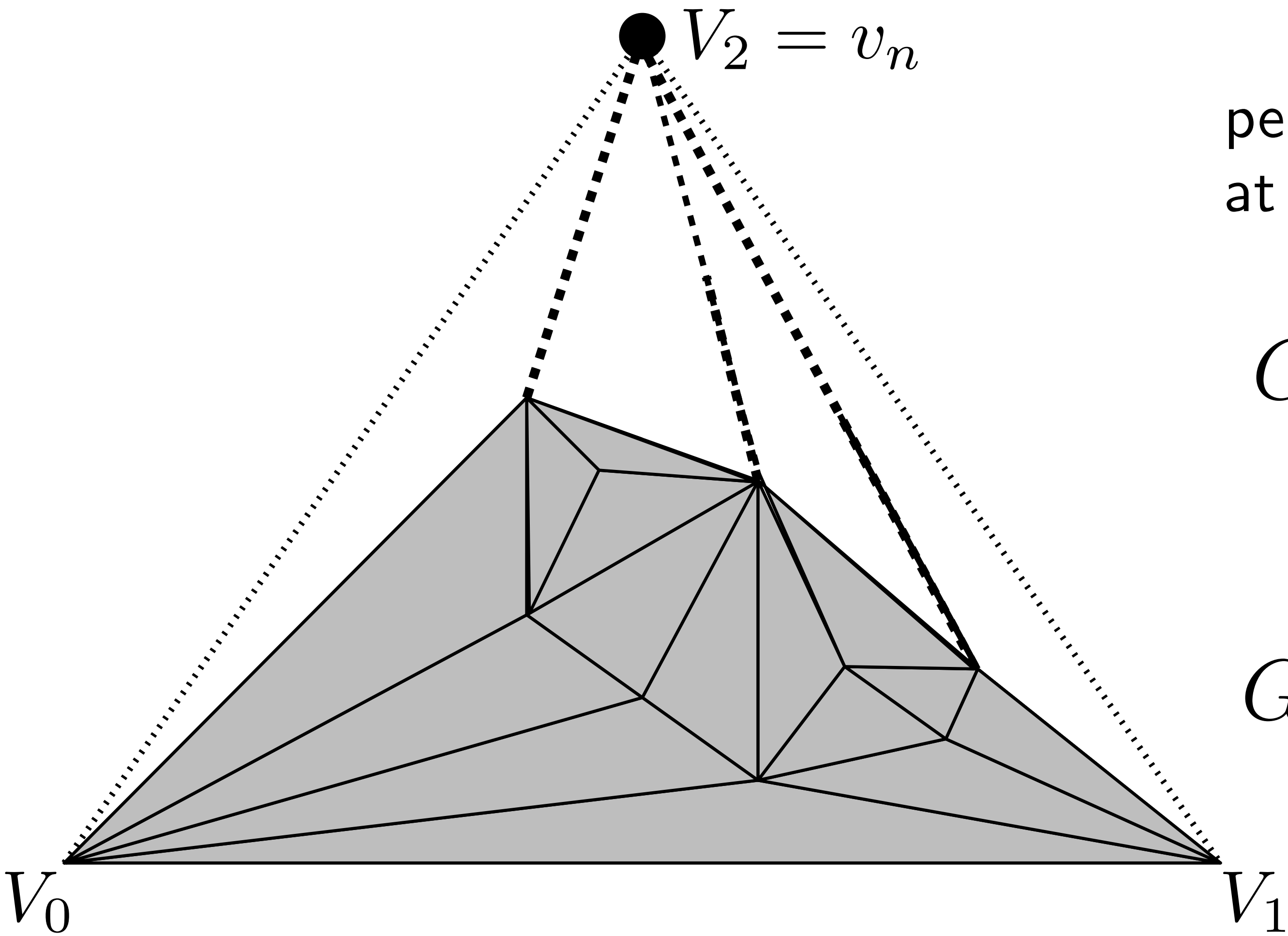
Theorem
Every planar triangulation admits a **Canonical Ordering**,
which can be computed in linear time.

The traversal starts from the root face

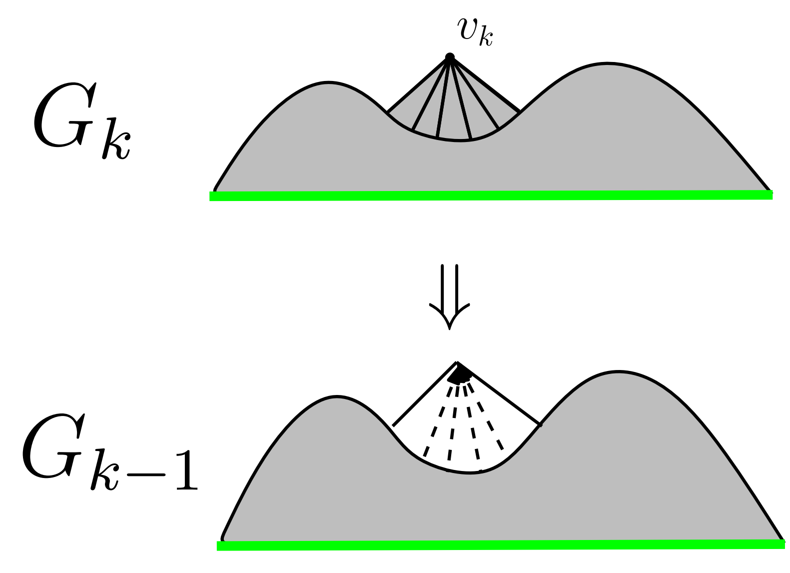


Canonical orderings: existence

The traversal starts from the root face

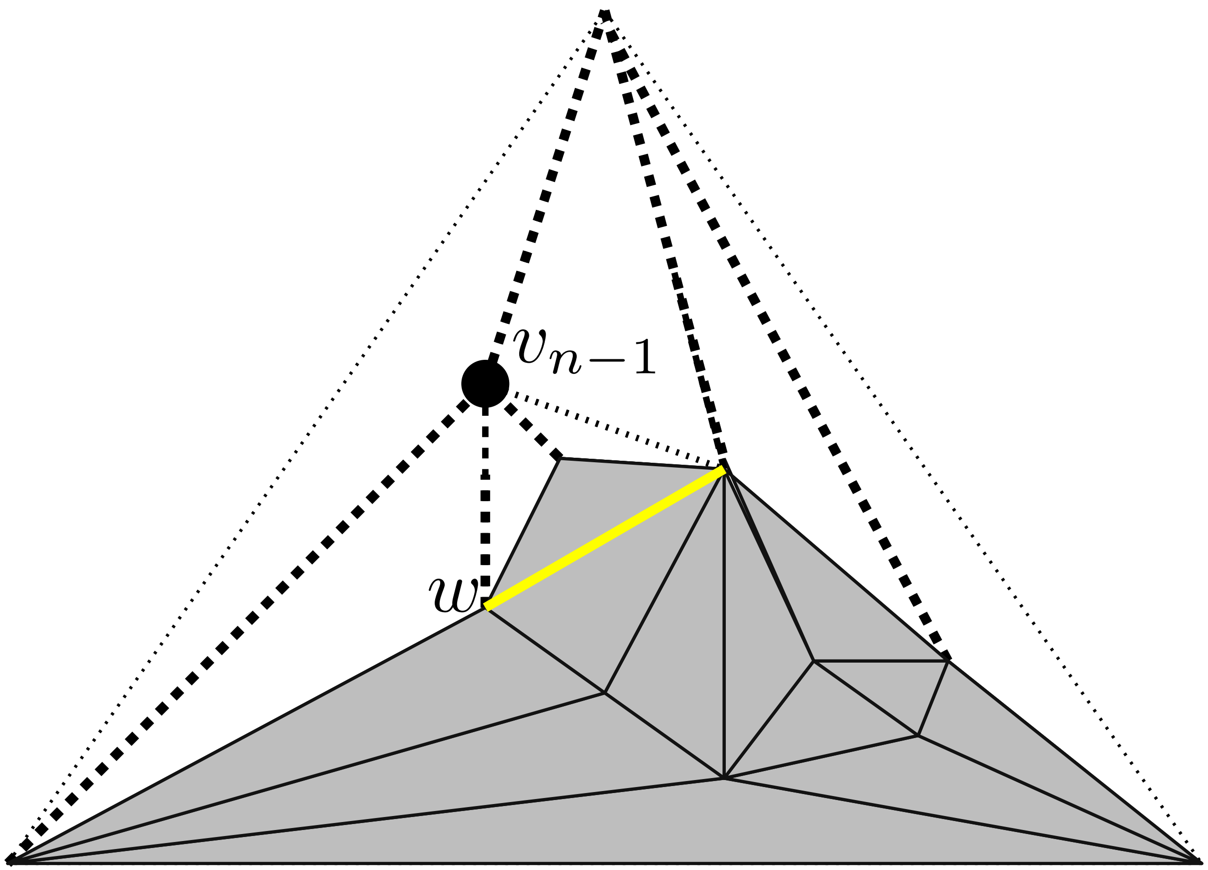


perform a vertex conquest at each step

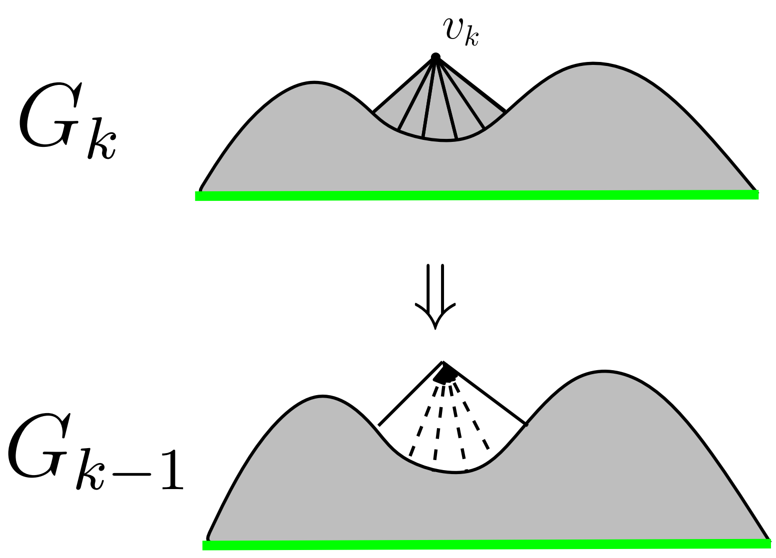


Canonical orderings: existence

The traversal starts from the root face

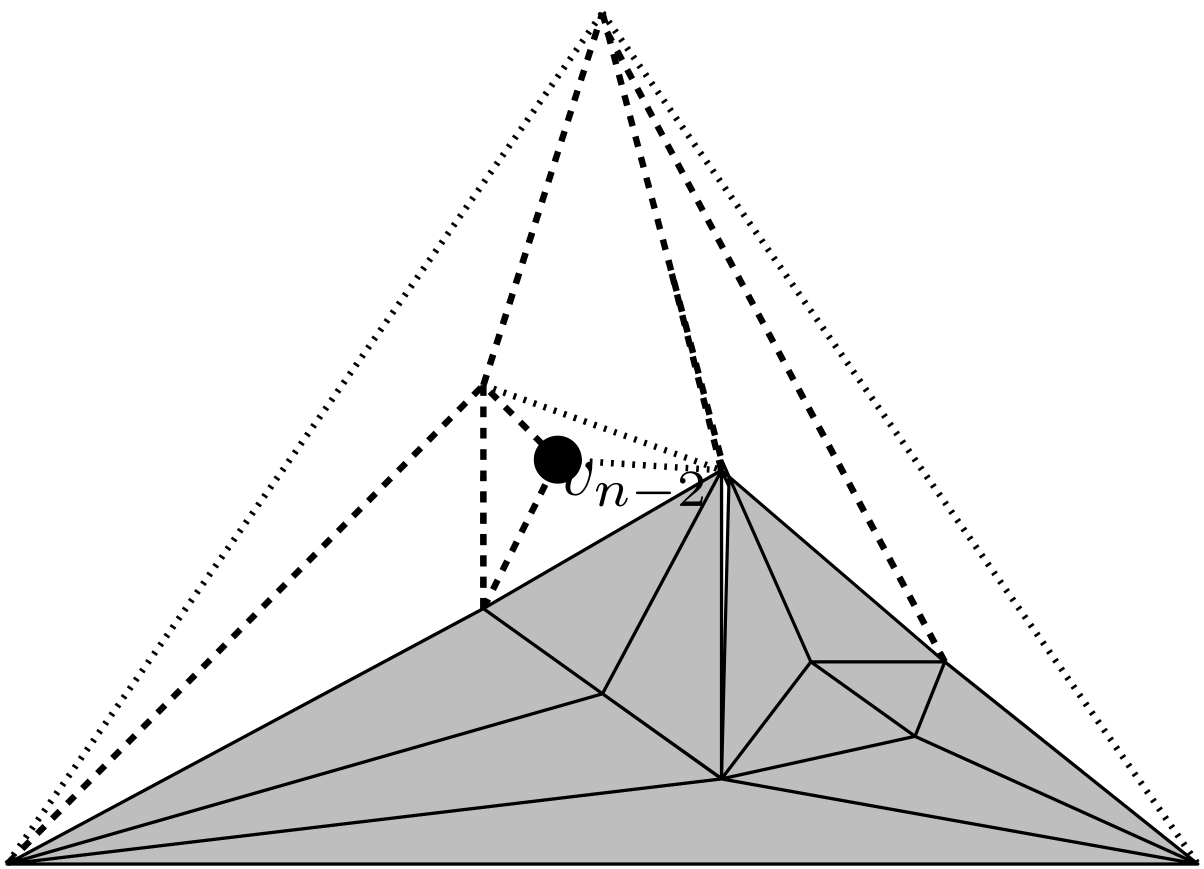


perform a vertex conquest at each step

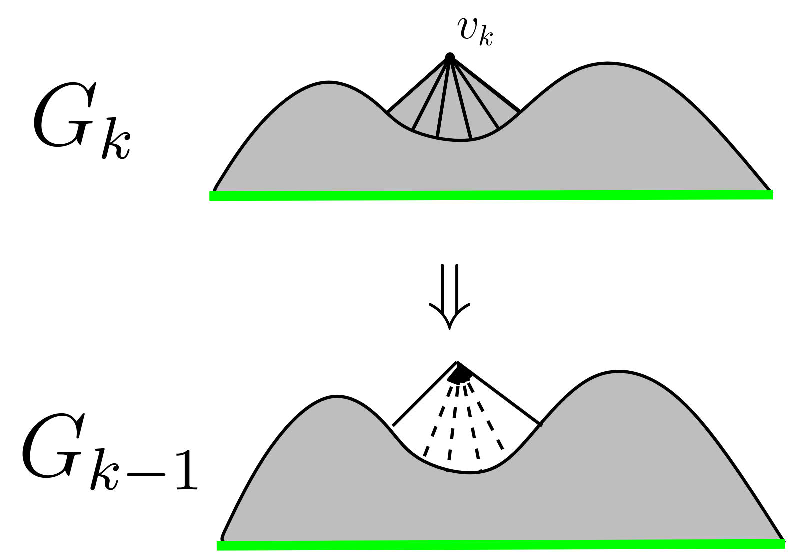


Canonical orderings: existence

The traversal starts from the root face

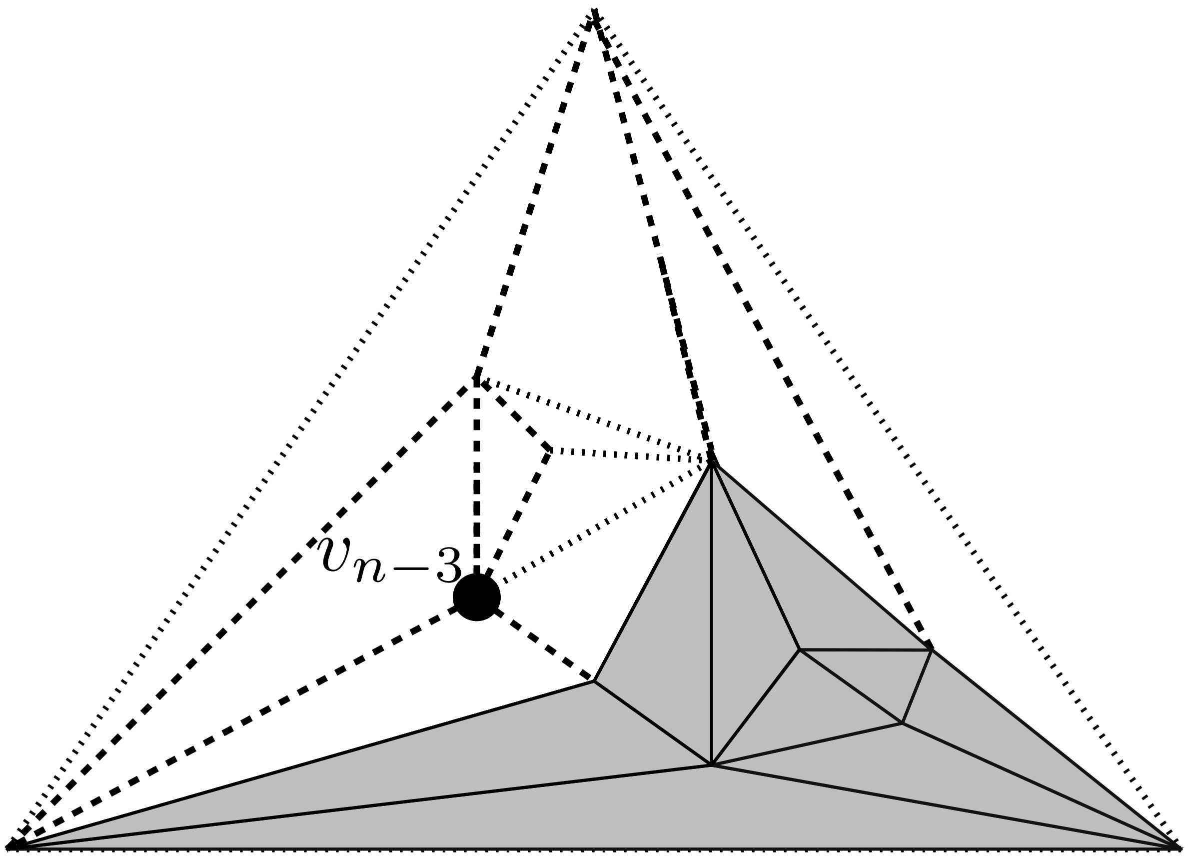


perform a vertex conquest
at each step



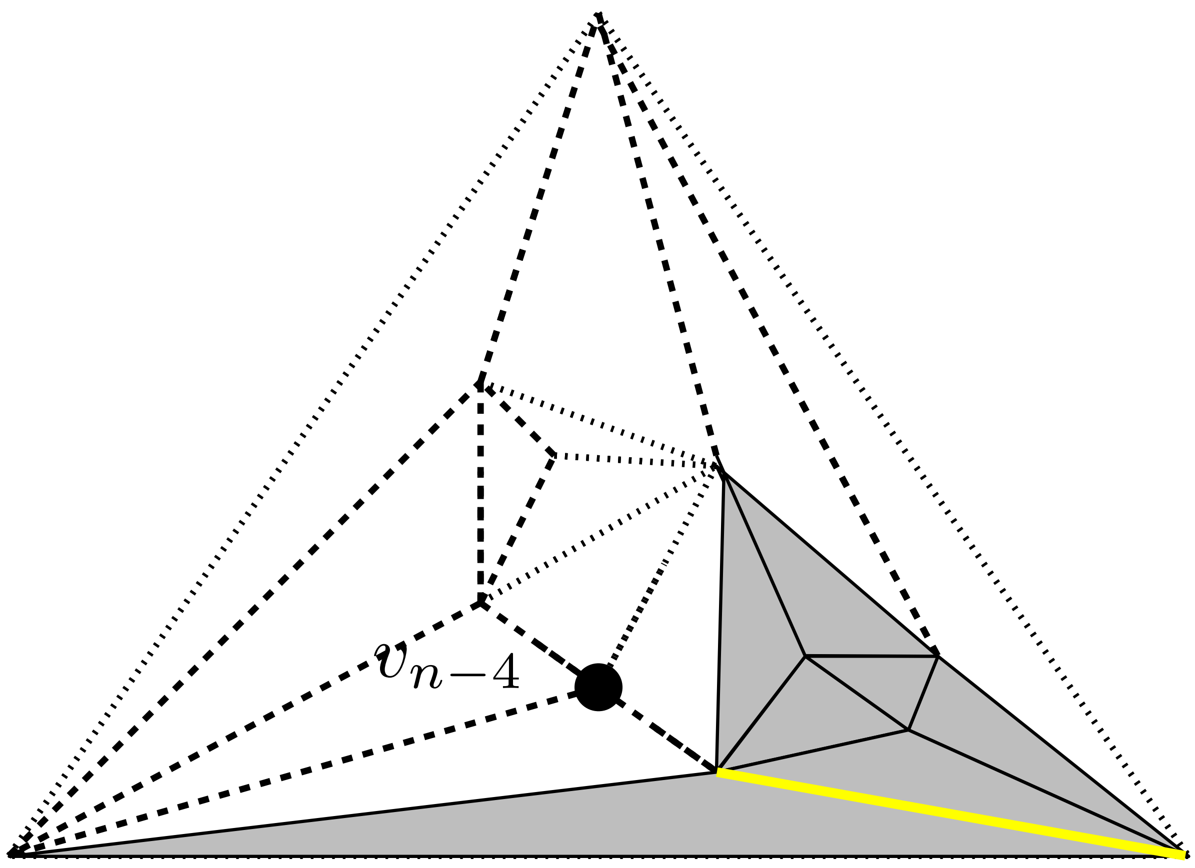
Canonical orderings: existence

The traversal starts from the root face



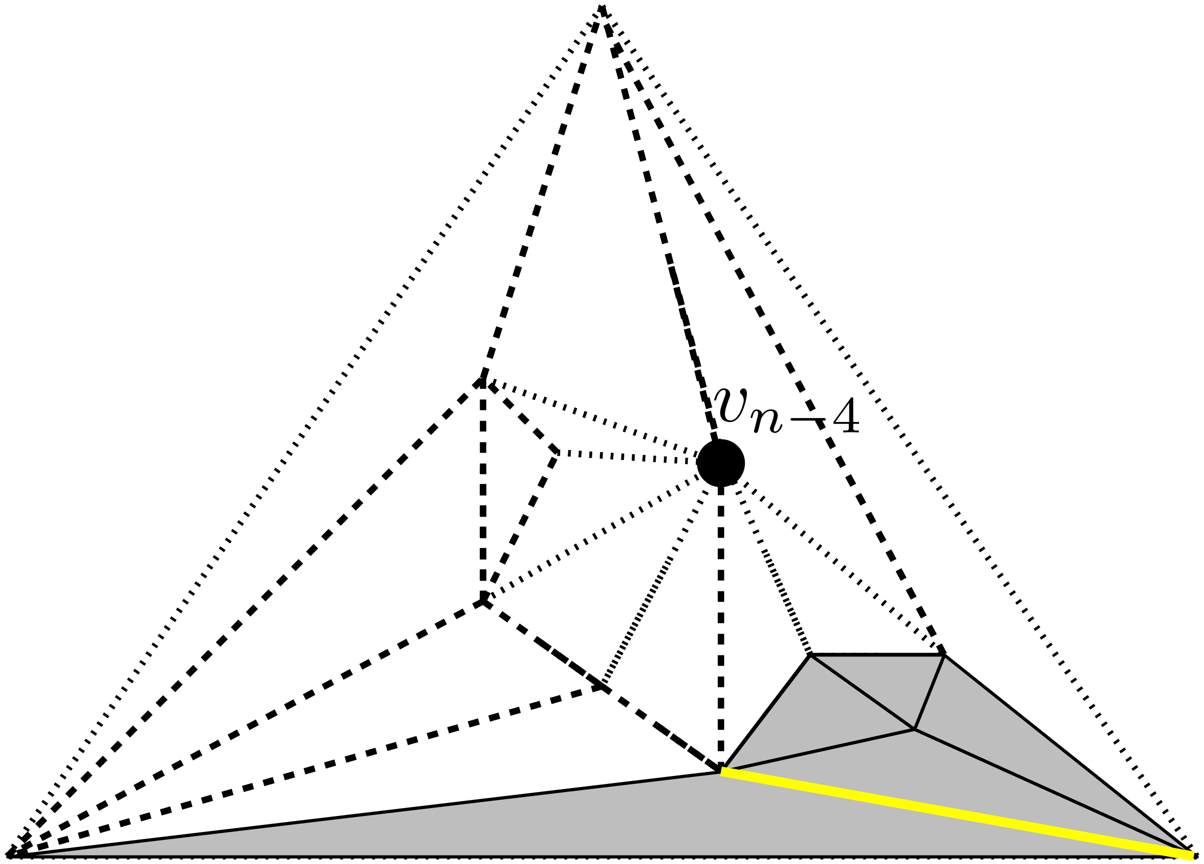
Canonical orderings: existence

The traversal starts from the root face

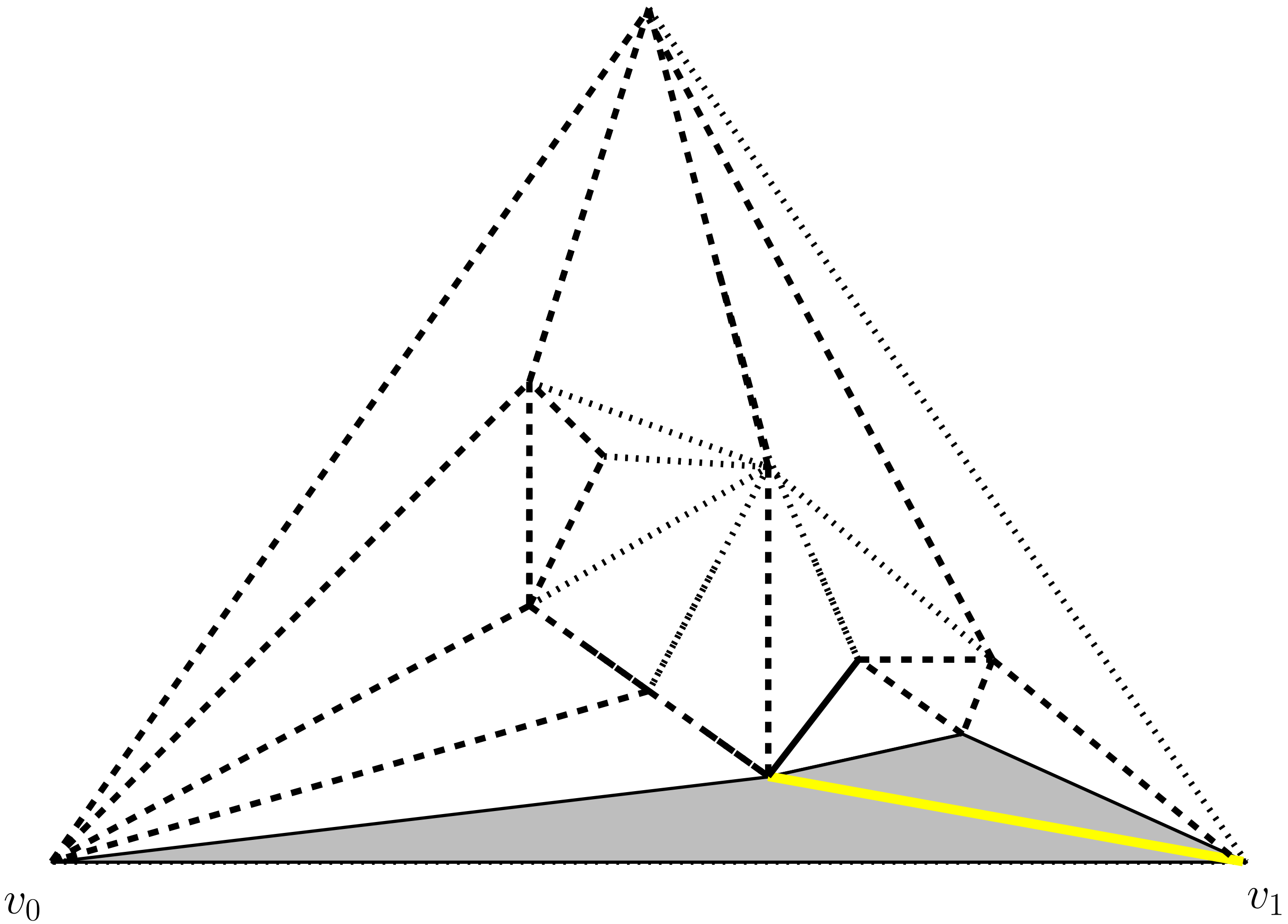


Canonical orderings: existence

The traversal starts from the root face

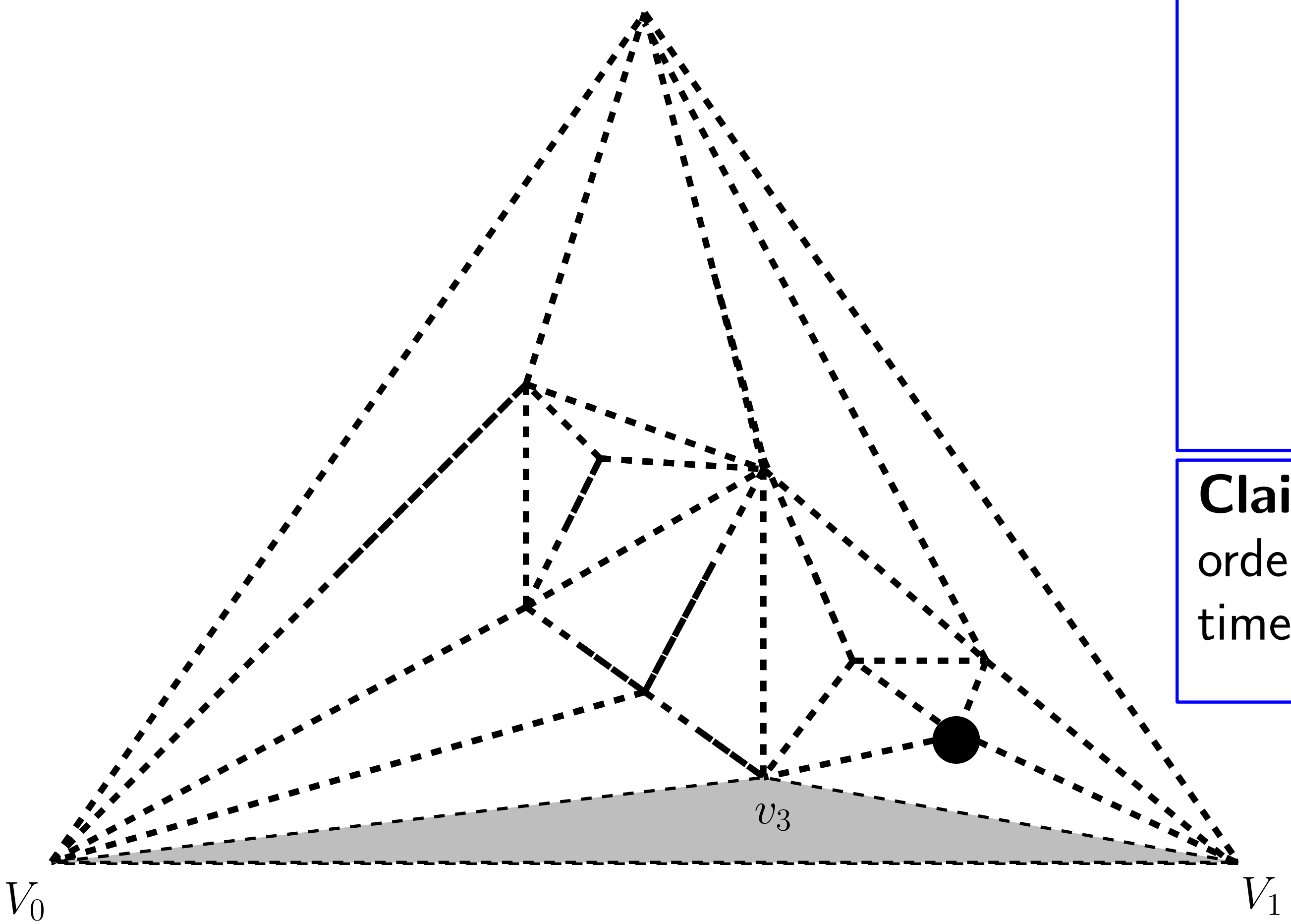


The traversal starts from the root face



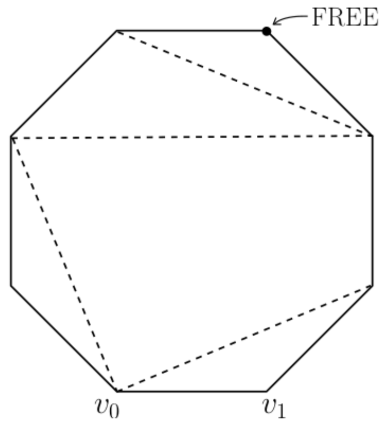
Canonical orderings: existence

The traversal starts from the root face



Claim (correctness): the shelling procedure terminates computing a canonical ordering

There must be a free vertex v (not V_0 nor V_1) without chords



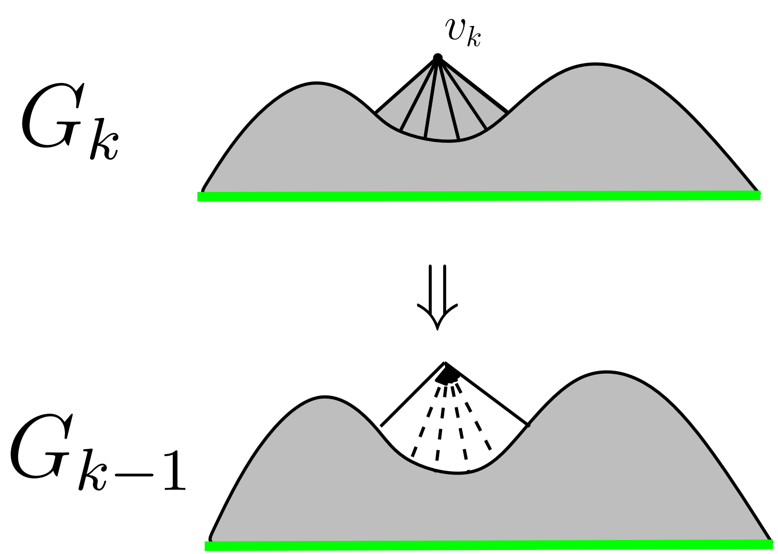
Claim (complexity): the canonical ordering can be computed in $O(n)$ time

Canonical orderings: exercices

exercice 1

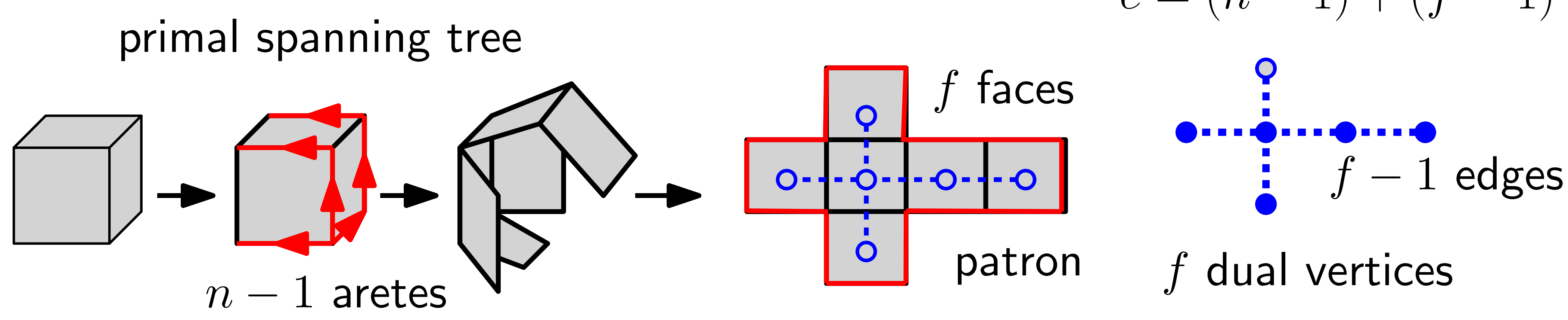
Give a proof of Euler formula using vertex shellings

$$\boxed{n - e + f = 2}$$



Famous proof of Euler formula

$$e = (n-1) + (f-1)$$



Canonical orderings: exercices

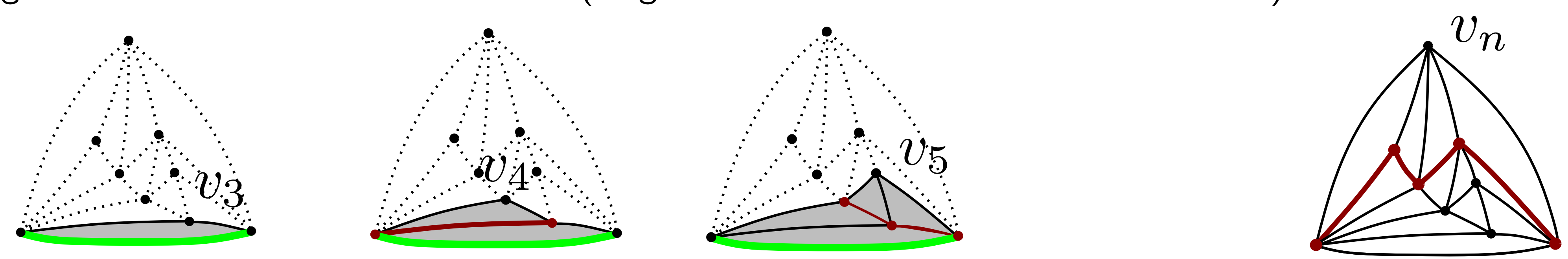
Application (exercise)

Design a (simple) linear time algorithm to embed a **maximal planar graph** G (it outputs `false` if the input graph is not a planar triangulation)

Assume first you are given a 3-cycle (V_0, V_1, V_2) which is a face of G

Step 1: simulate the computation of the canonical ordering on the graph G

Step 2: compute the embedding of G (its faces) by adding the vertices $v_3, v_4, \ldots, v_n$ incrementally together with the fan of incident faces (neighbors are consecutive on the outer face)



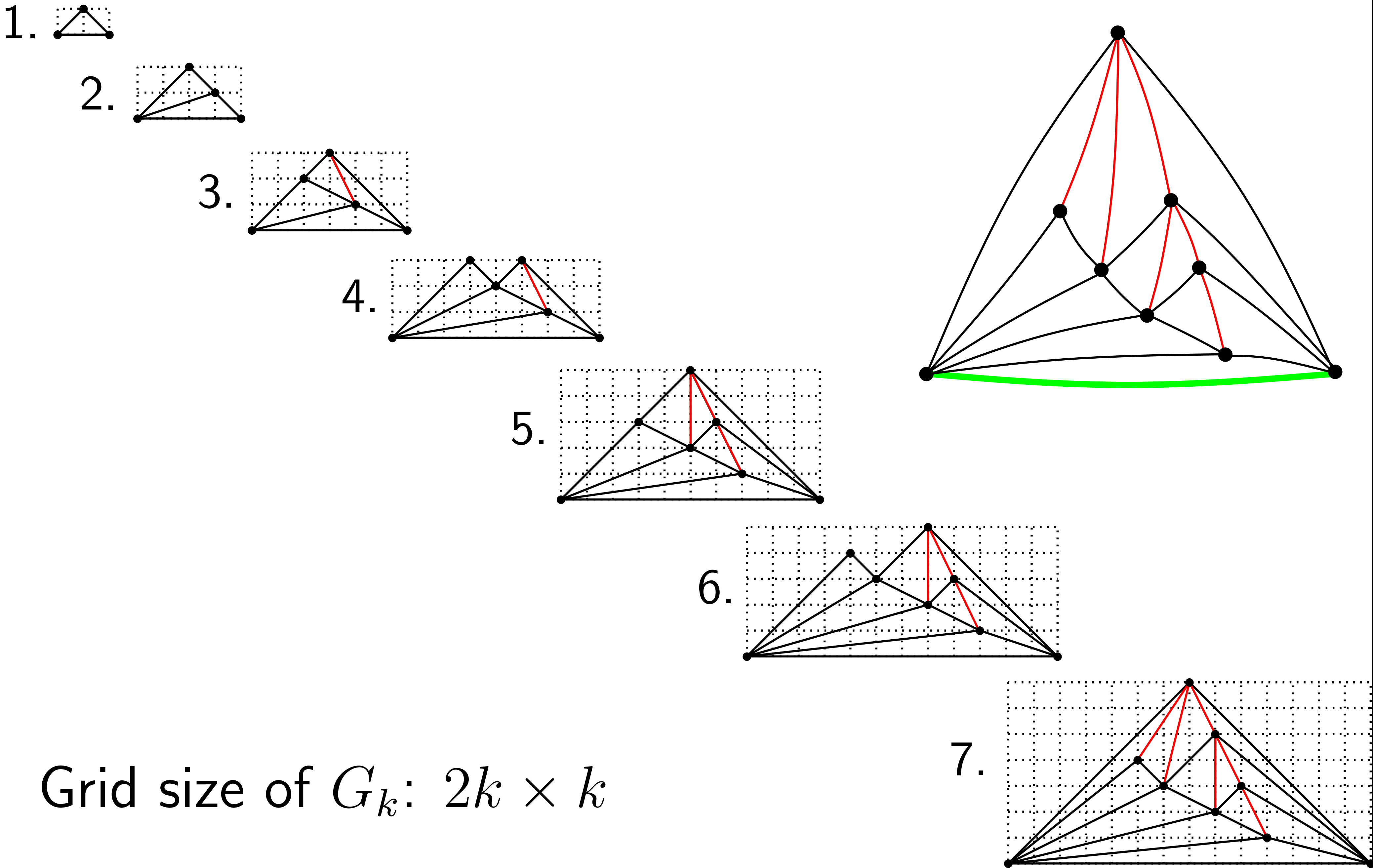
Remark: what happens if the 3-cycle is not a face of G?

Planar straight-line drawings

(FPP algorithm)

Incremental drawing algorithm

[de Fraysseix, Pollack, Pach'89]

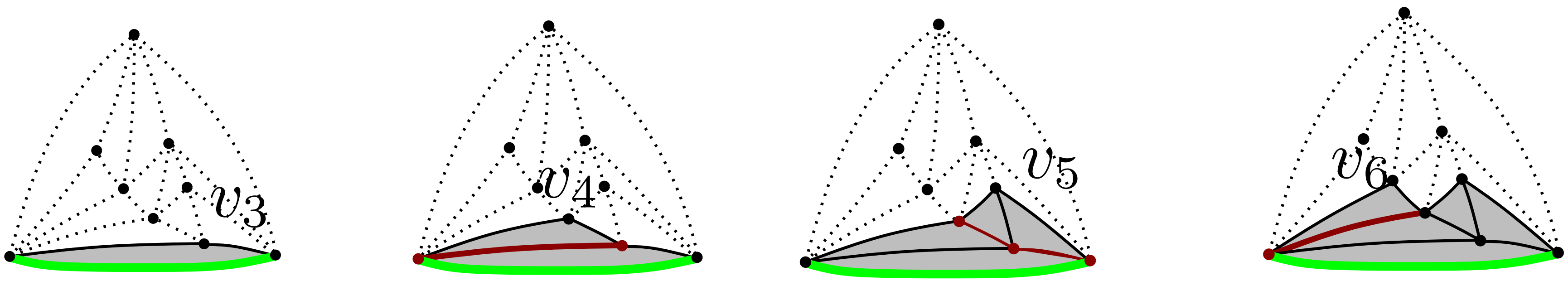


Grid size of G_k: $2k \times k$

Incremental drawing algorithm

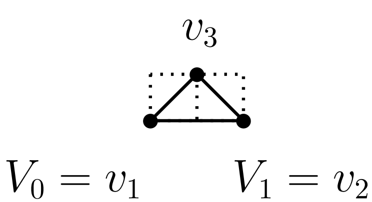
[de Fraysseix, Pollack, Pach'89]

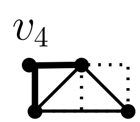
Idea: add vertices incrementally (according to the canonical ordering) together with their incident faces (in the outer face)

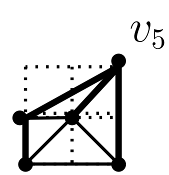


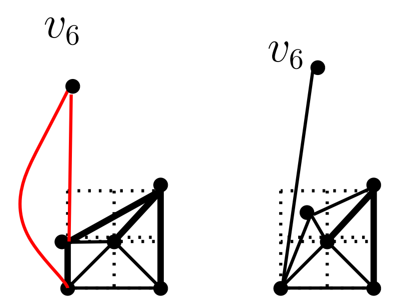
Step 1:
Add first face

Step 2:
Add v_4



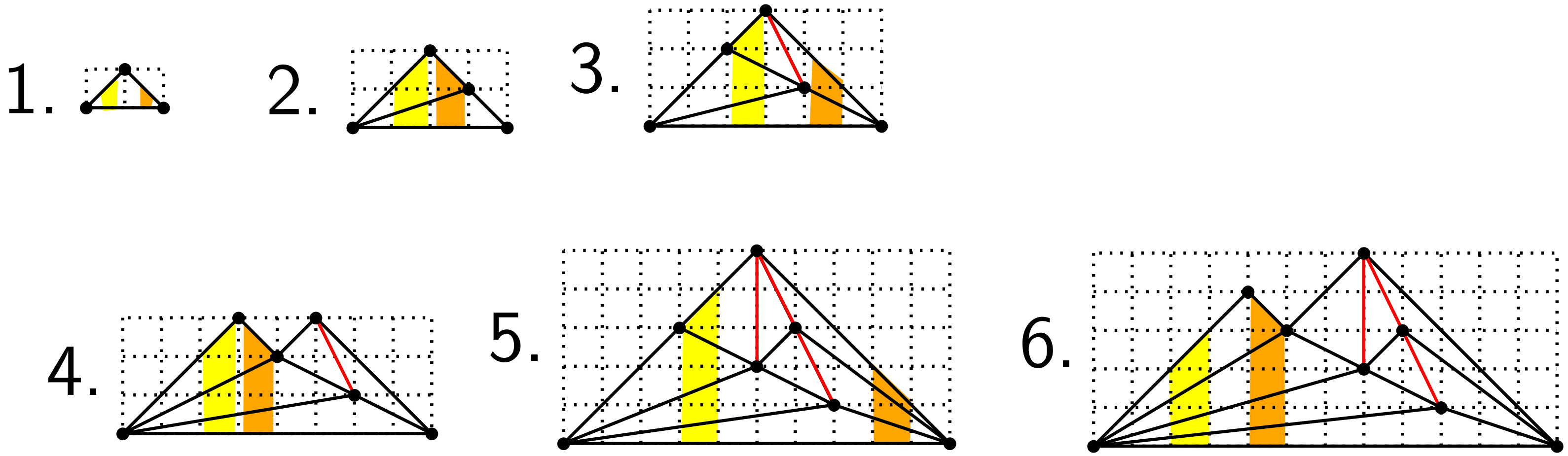






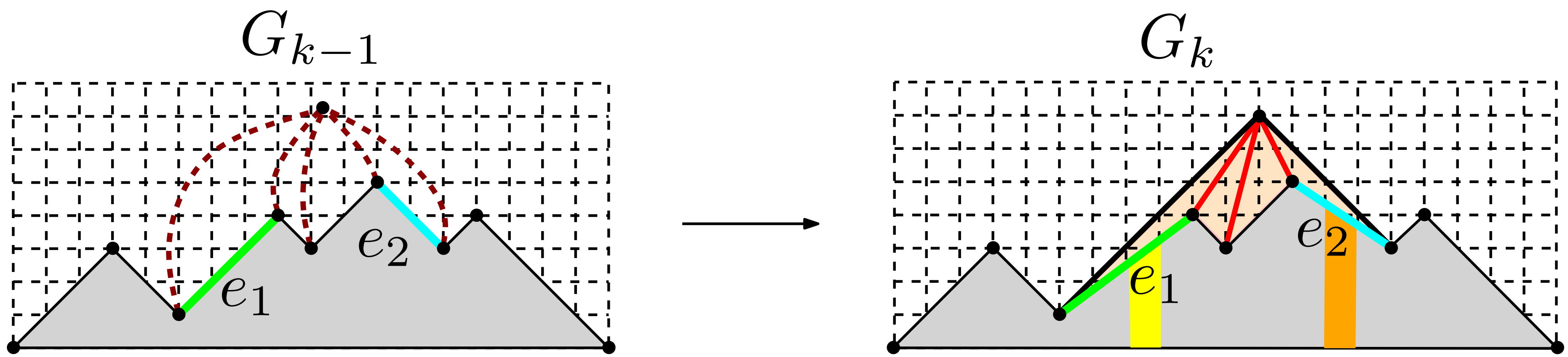
Step 4:
problems: either the vertices are not visible, or the grid becomes too big

incremental shift algorithm (original FPP)



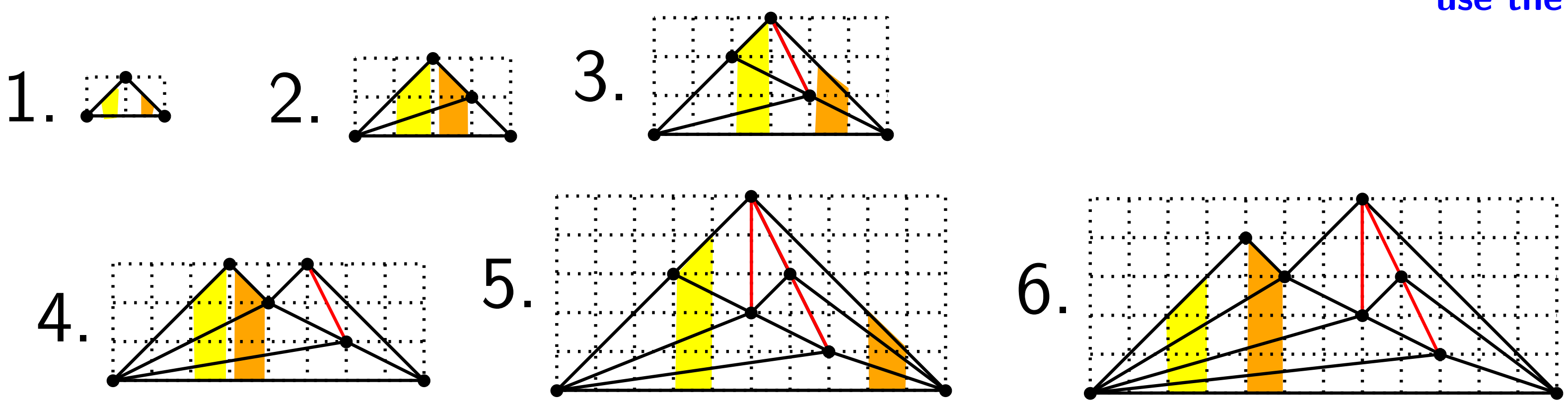
1. Make the grid large enough: add two vertical strips (of width 1)
2. Add the edges incident to v_k (leftmost and rightmost) of slope $+1$ and -1

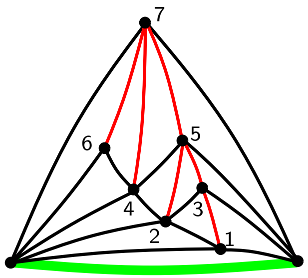
stretch horizontally edges e_1 and e_2 of 1



incremental shift (FPP) algorithm

use the canonical ordering

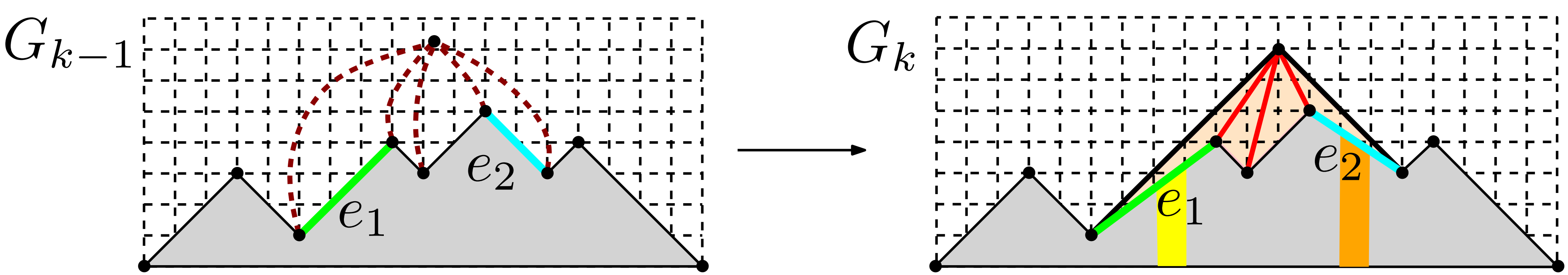




Claims:

1. Vertices are drawn as grid points
2. the grid is polynomial: $O(n) \times O(n)$
3. the execution takes $O(n)$ time
4. the drawing is planar: no edge crossings

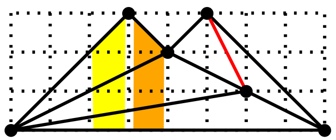
stretch horizontally edges e_1 and e_2 of 1



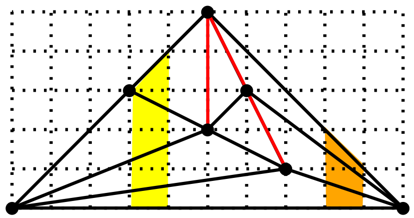
incremental shift (FPP) algorithm

Let us make things more precise

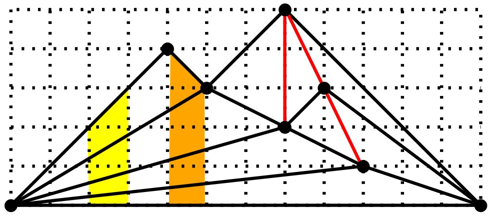
4.



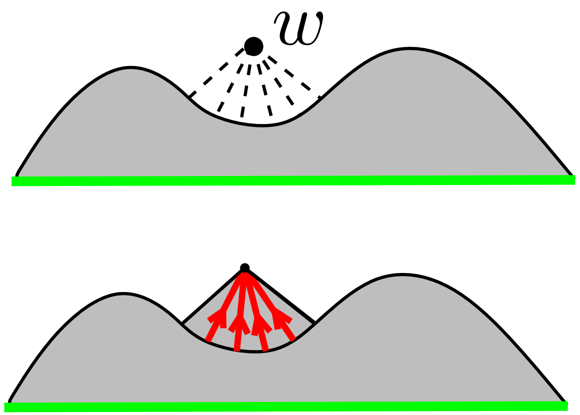
5.

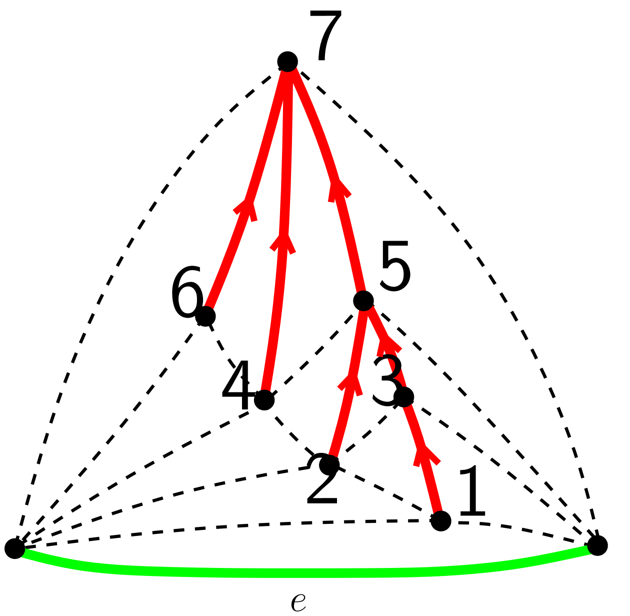


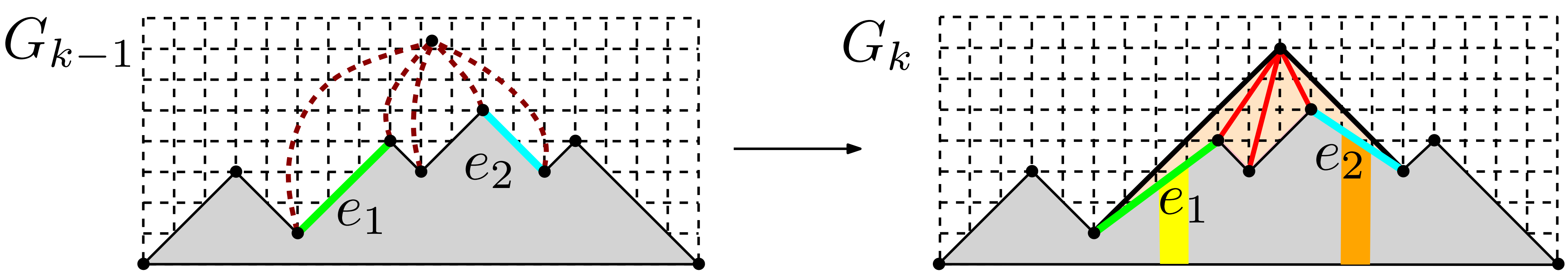
6.



1. Consider the following primal (red) tree:
connect v to its largest neighbor w

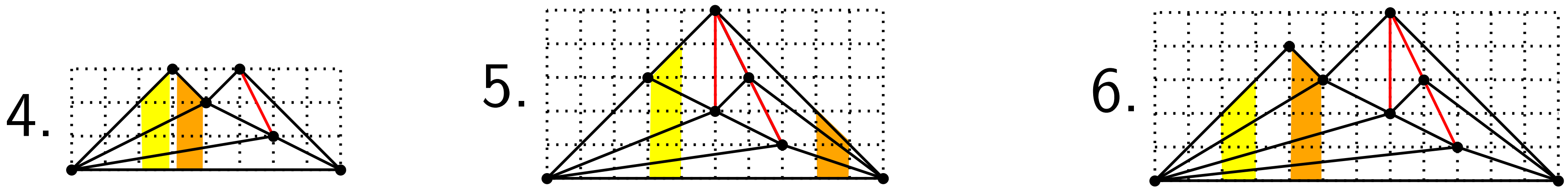




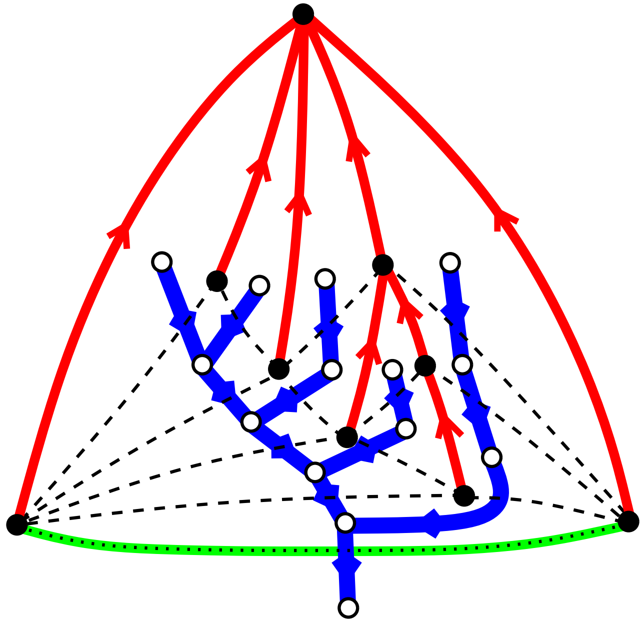


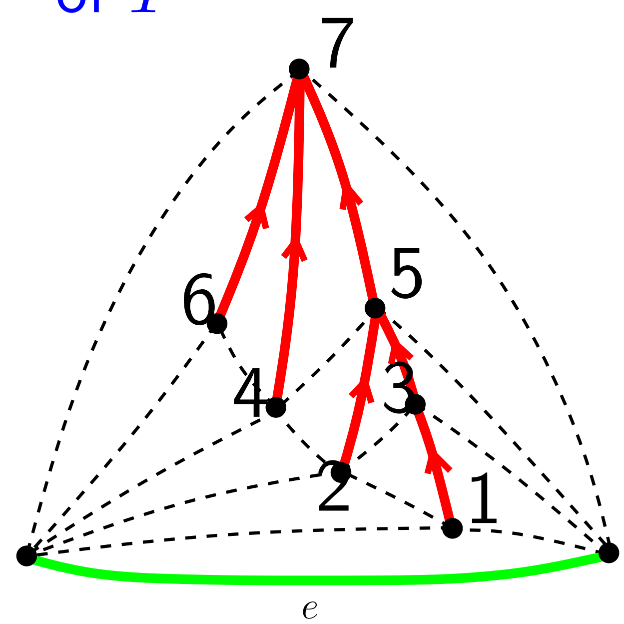
incremental shift (FPP) algorithm

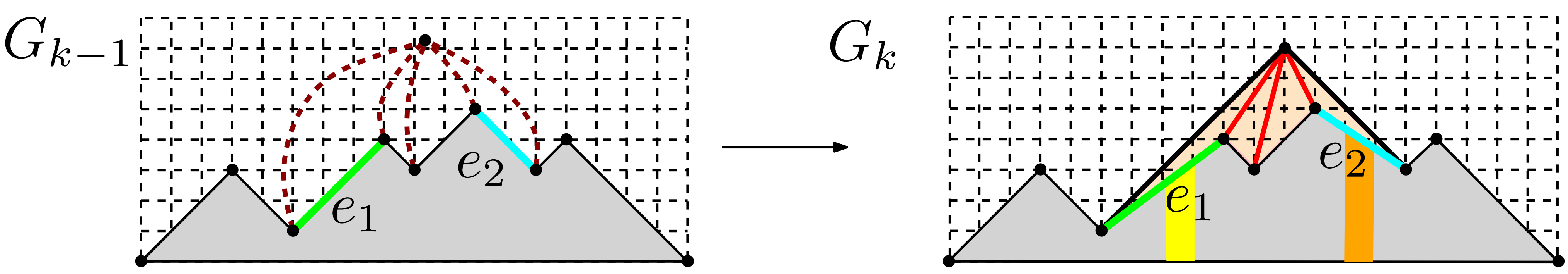
Let us make things more precise



2. Consider the dual (blue) spanning tree T^* of T

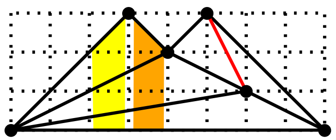


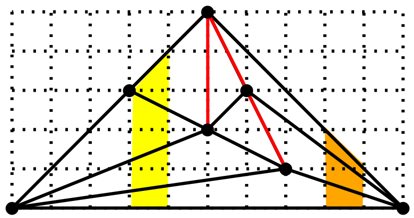


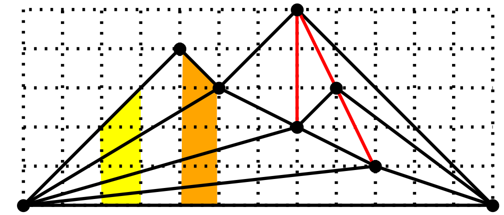


incremental shift (FPP) algorithm

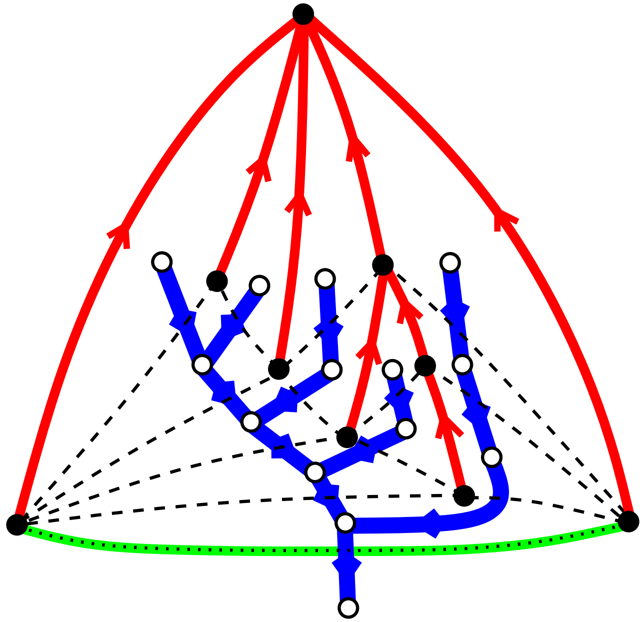
Let us make things more precise

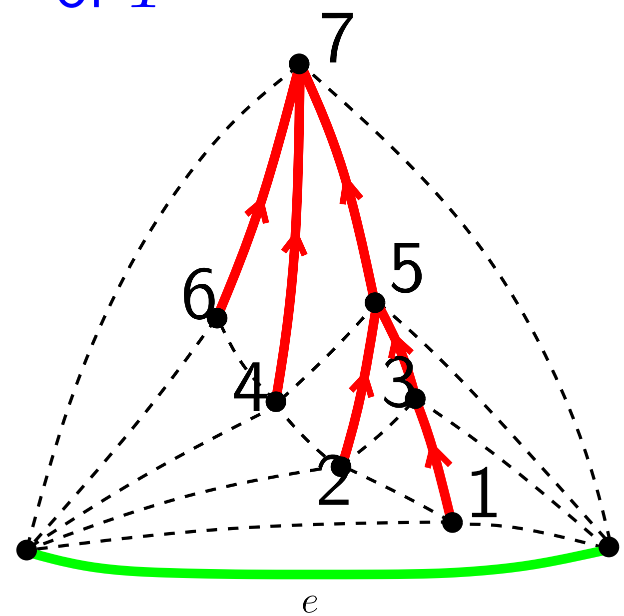
4. 

5. 

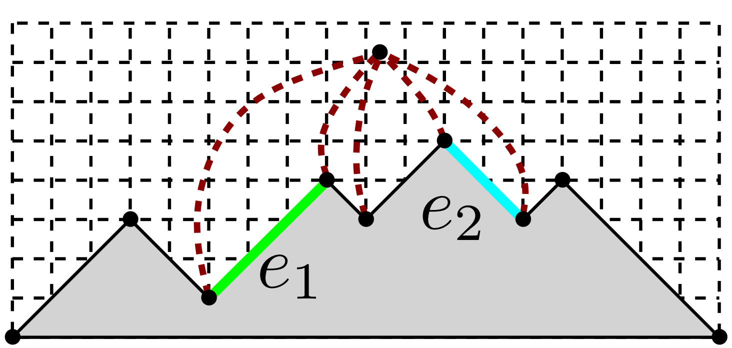
6. 

2. Consider the dual (blue) spanning tree T^* of T

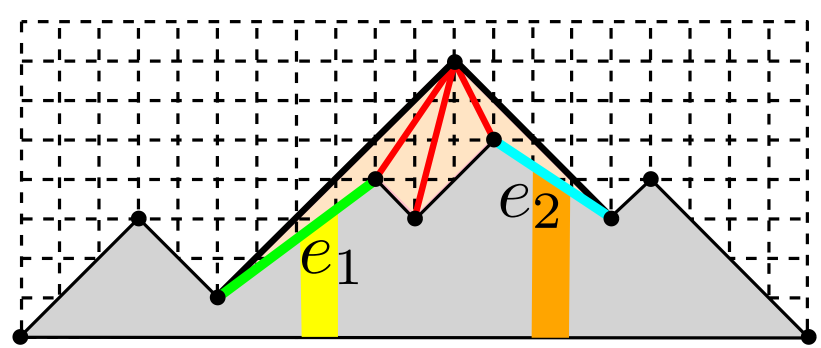




G_{k-1}

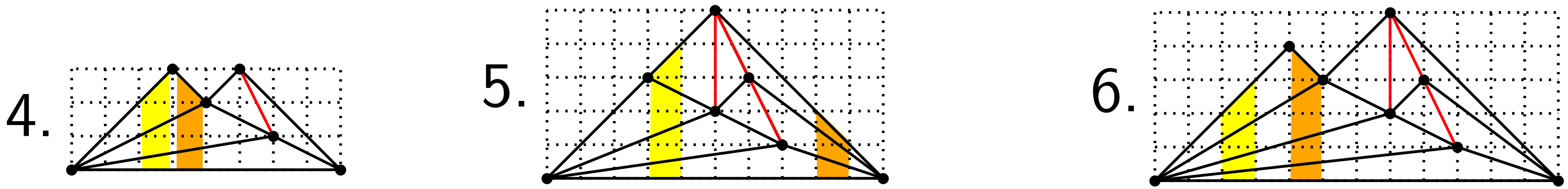


G_k

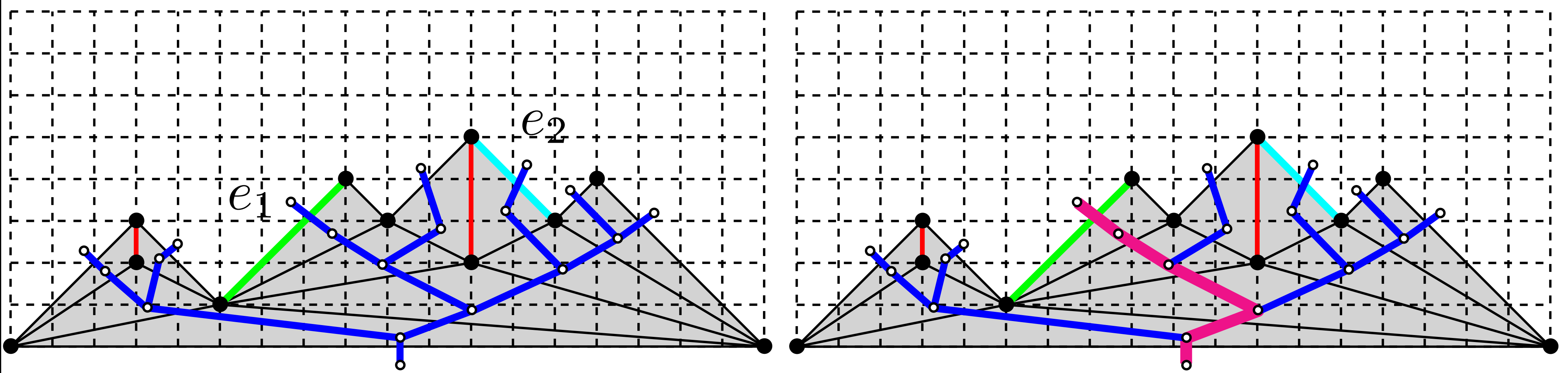


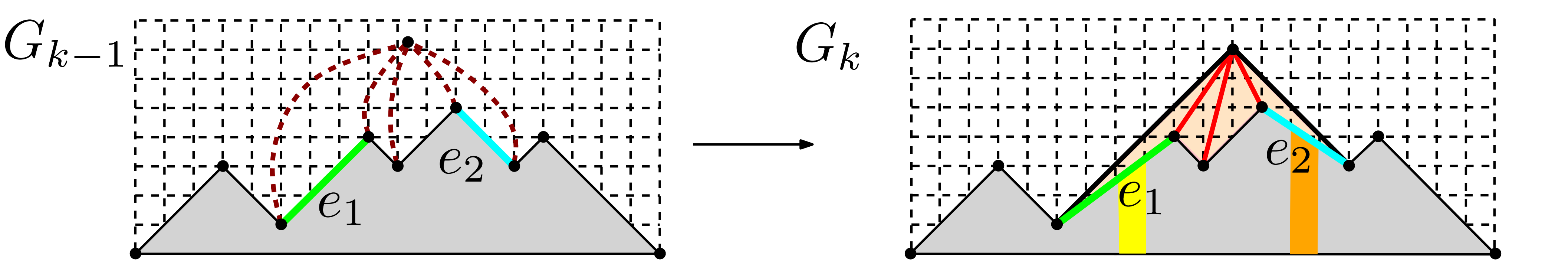
incremental shift (FPP) algorithm

Let us make things more precise



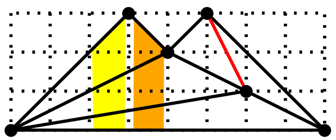
3. Stretch e_1 and e_2 and all edges which are "below"
(e_1, e_2:= leftmost and rightmost edges incident to v_k)

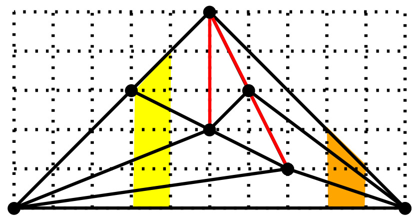


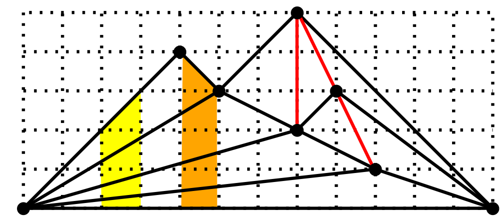


incremental shift (FPP) algorithm

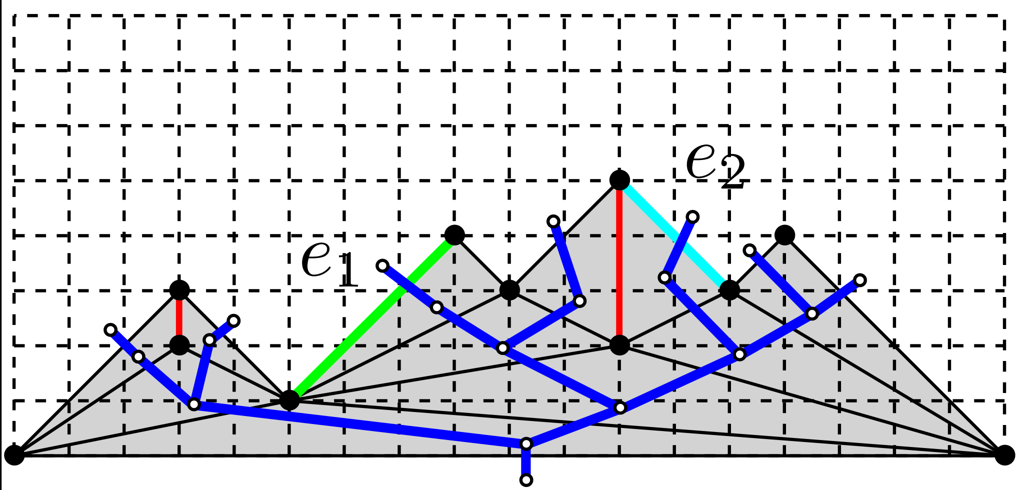
Let us make things more precise

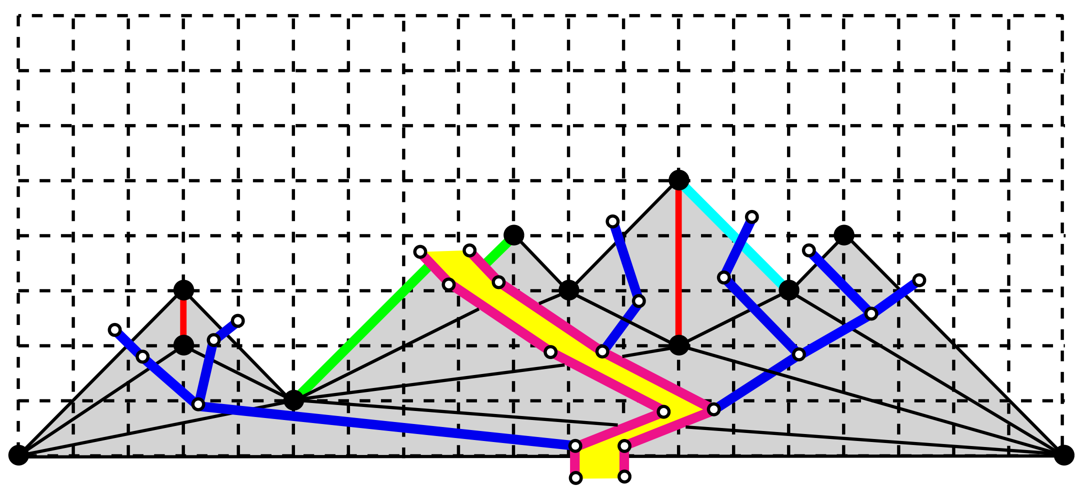
4. 

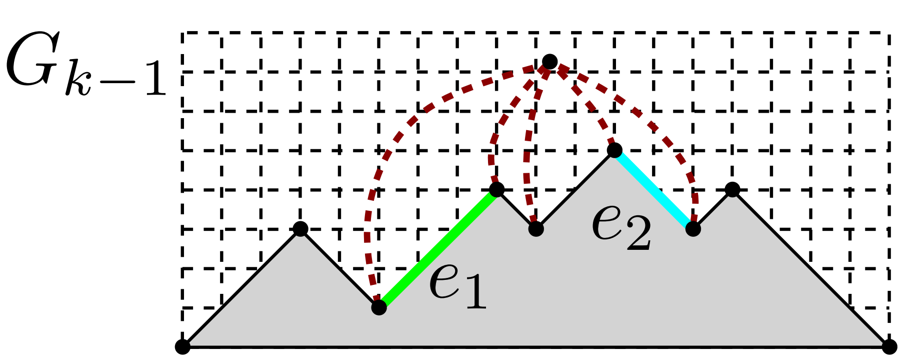
5. 

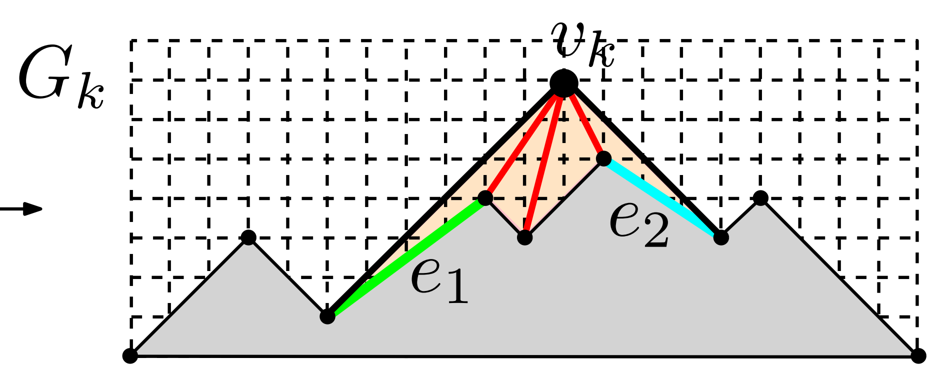
6. 

3. Stretch e_1 and e_2 and all edges which are "below"
(e_1, e_2:= leftmost and rightmost edges incident to v_k)



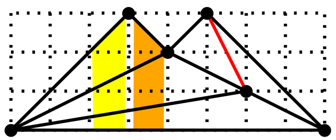


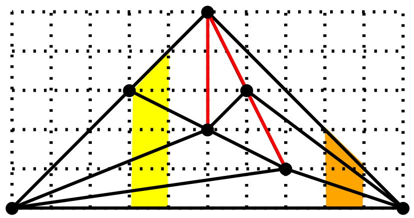


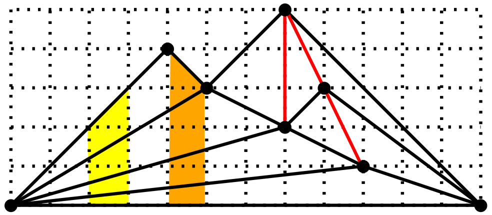


incremental shift (FPP) algorithm

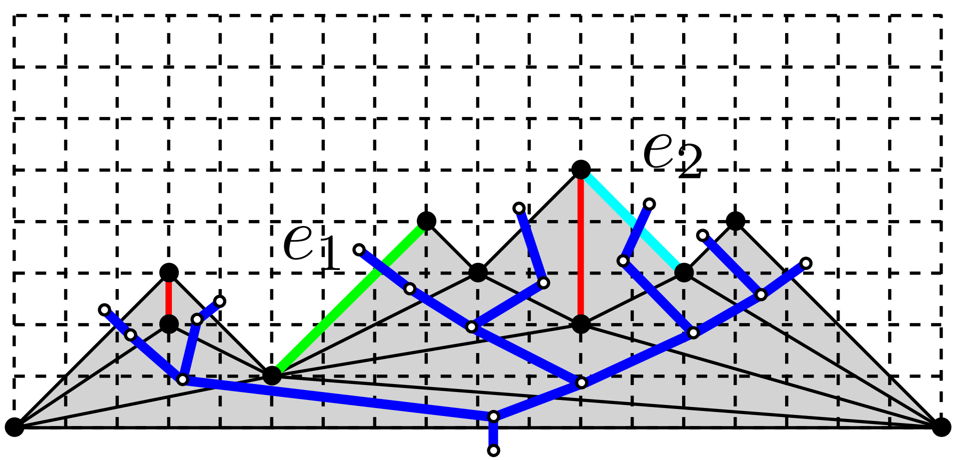
Let us make things more precise

4. 

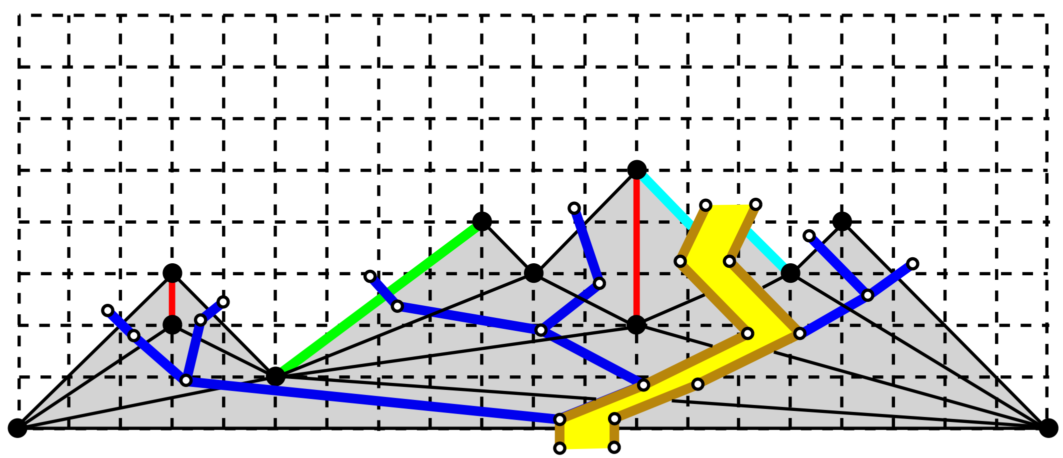
5. 

6. 

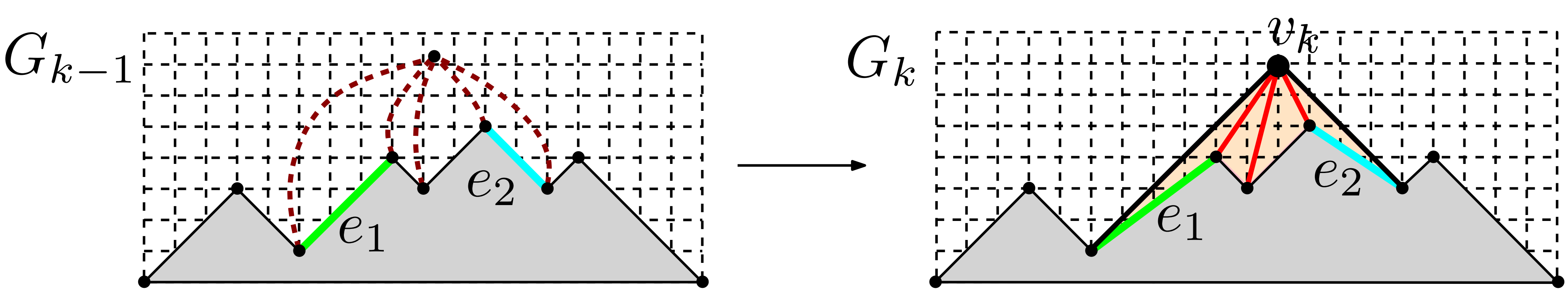
3. Stretch e_1 and e_2 and all edges which are "below"
(e_1, e_2:= leftmost and rightmost edges incident to v_k)



width k



width $k+2$



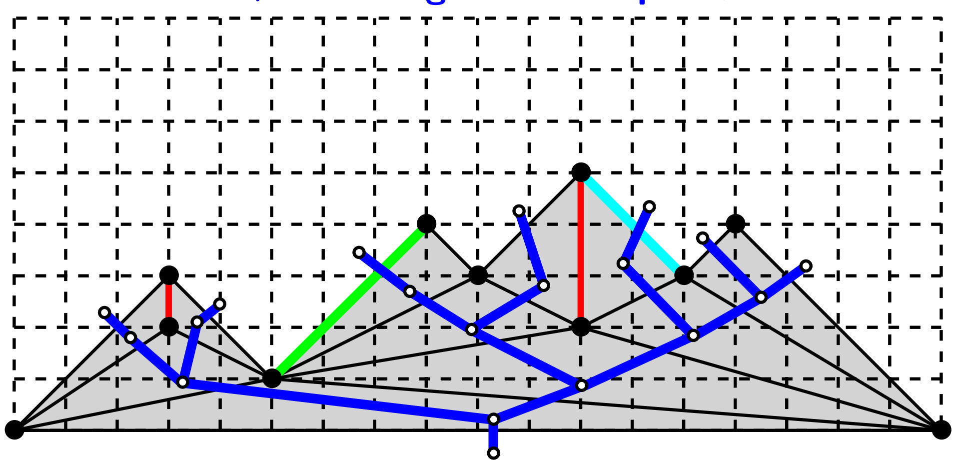
incremental shift (FPP) algorithm

4. add v_k at the crossing of the edges with slopes $+1$ and -1

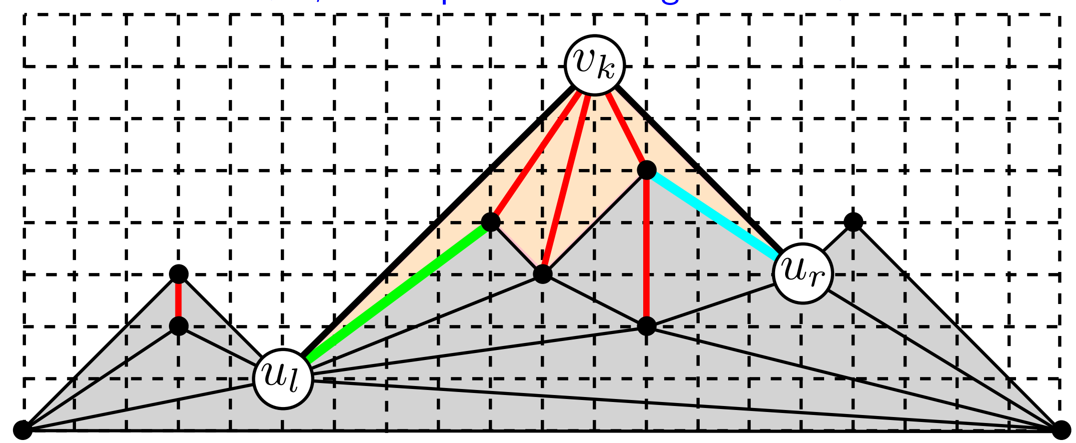
Claim: vertex v_k is a grid point

Proof: the manhattan distance between u_l and u_r is even
(since the slopes of outer edges are always $+1$ or -1)

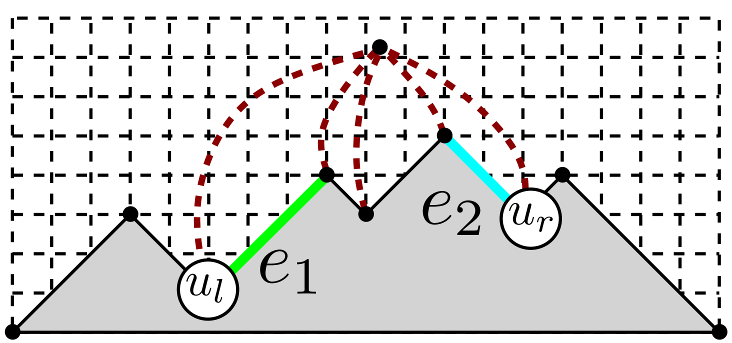
width k, outer edges have slopes $+1$ or -1



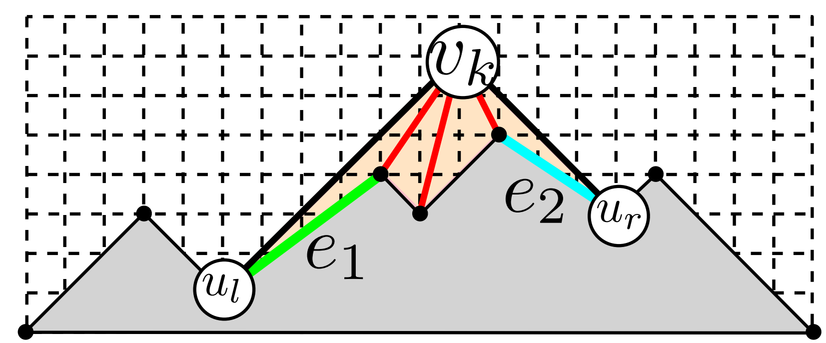
width $k+2$, the slope of outer edges is still $+1$ or -1



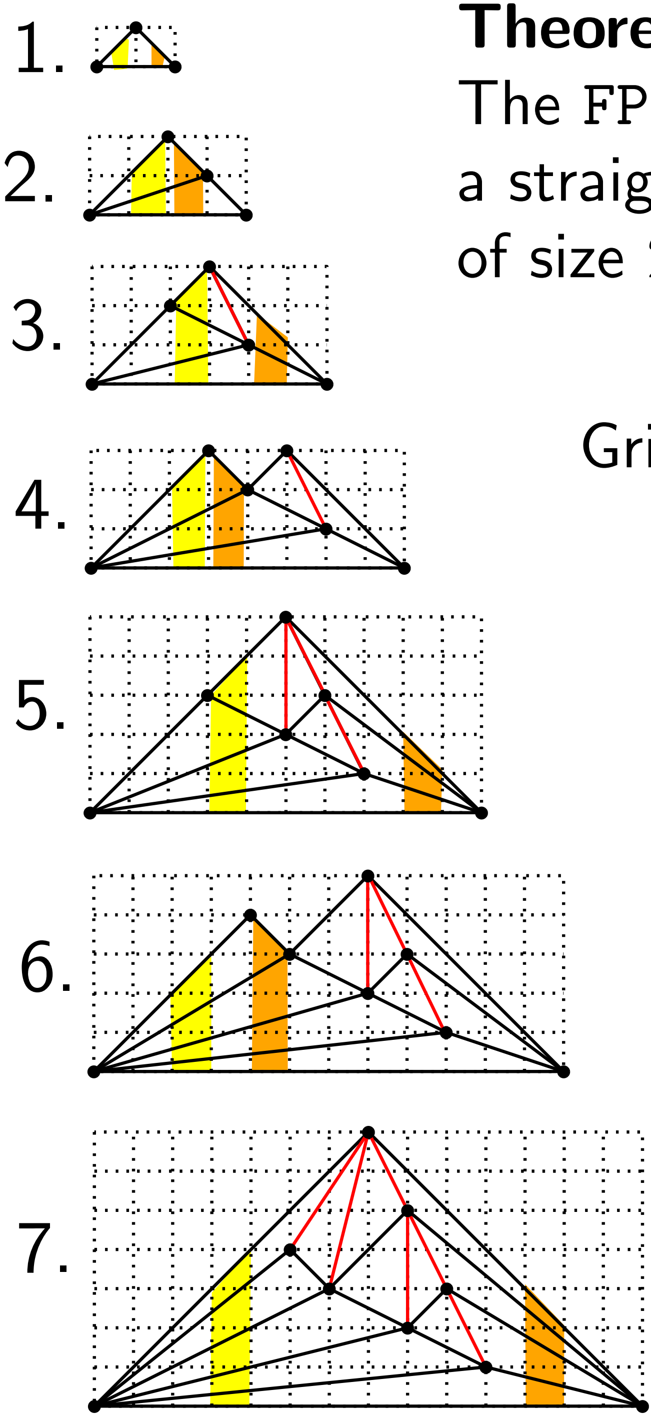
G_{k-1}



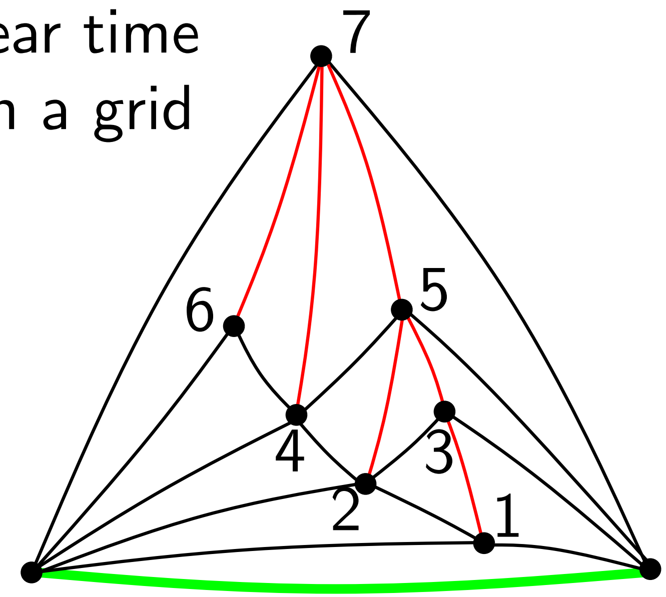
G_k



incremental shift algorithm (original FPP)



Theorem [de Fraysseix, Pollack, Pach'89]
The FPP algorithm computes in linear time a straight-line grid drawing of T, on a grid of size $2n \times n$



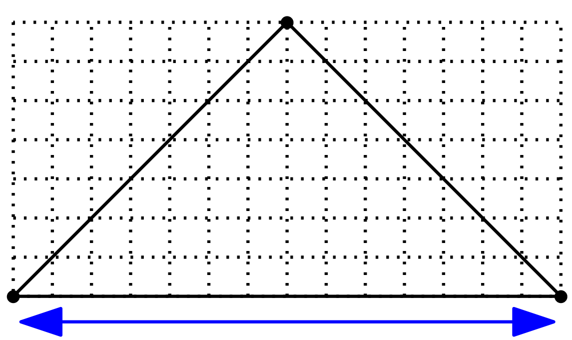
Grid size of G_k: $2k \times k$

1. Vertices are drawn as grid points

Vertex coordinates are integers, because the Manhattan distance between vertices on the outer boundary is even: at each step the edges on the outer face have slopes $+1$ or -1

2. the grid is polynomial: $2n \times n$

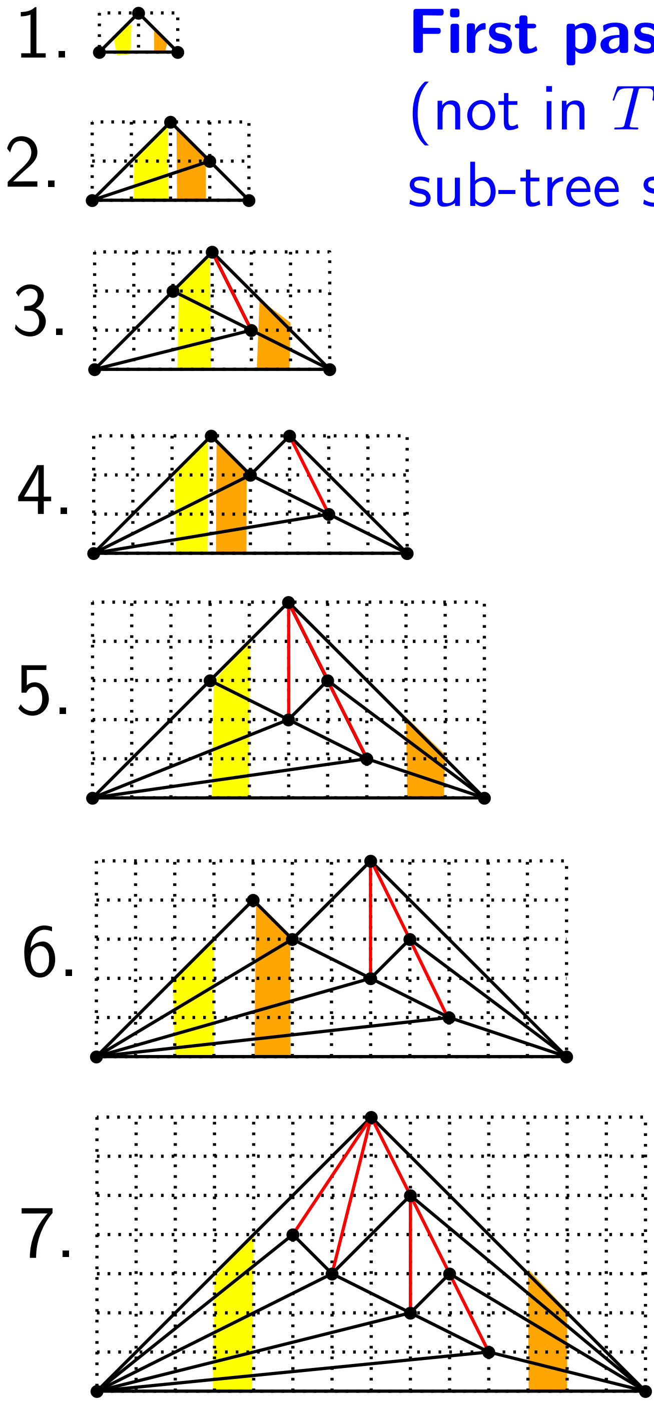
for every vertex we stretch by 2 horizontally

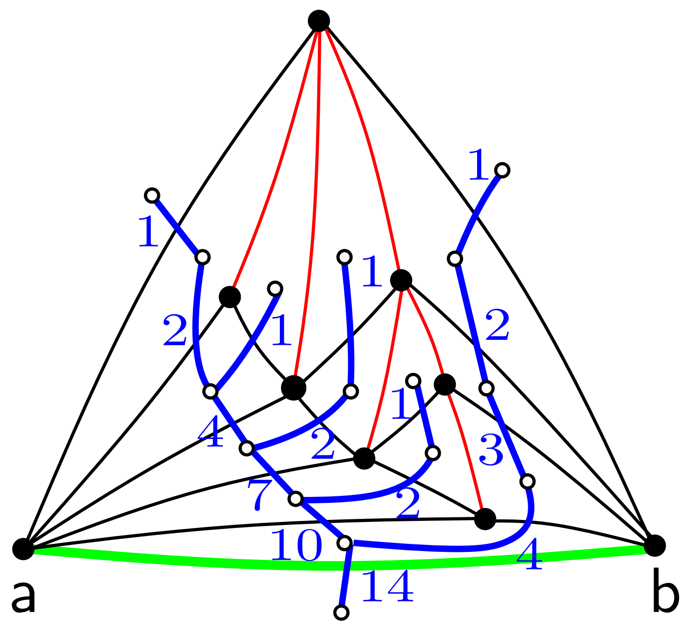


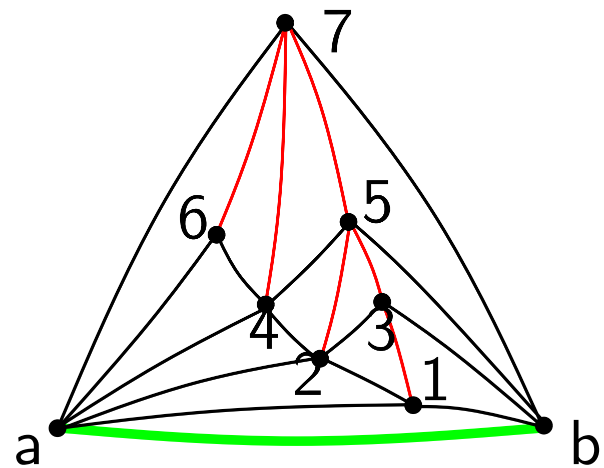
Two-passes implementation: linear-time

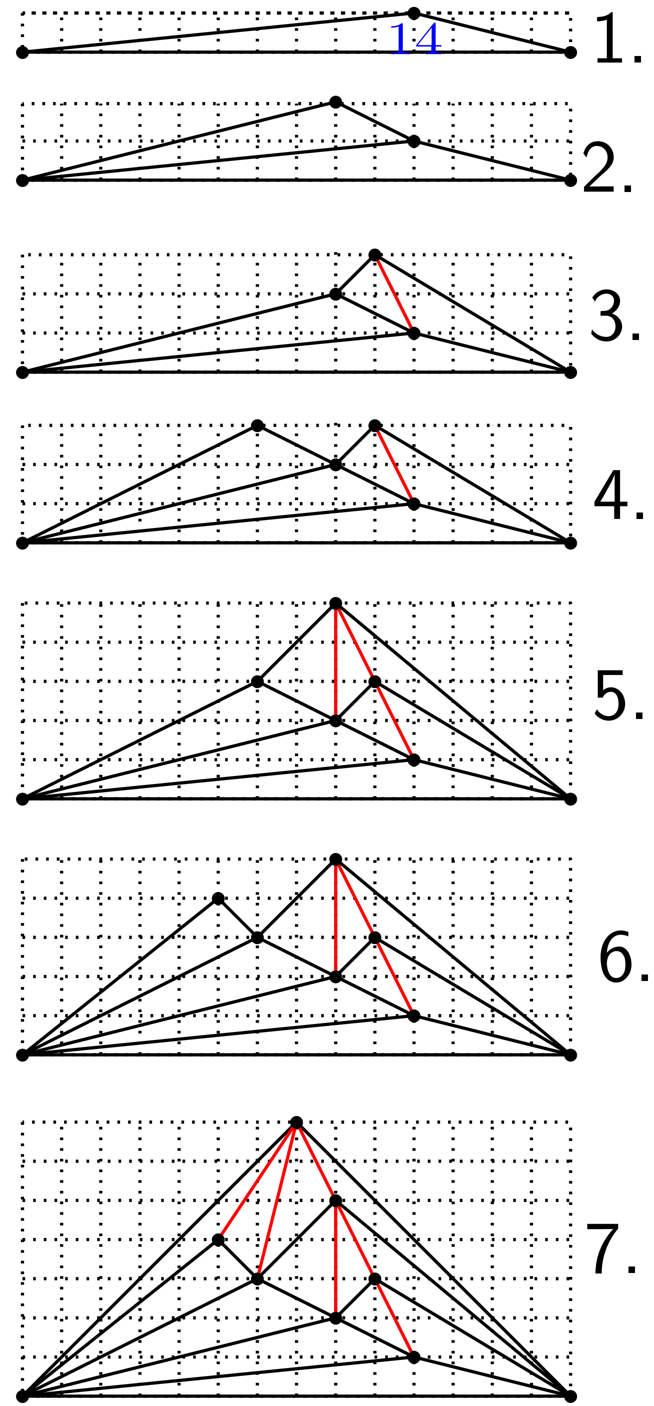
Second pass

First pass: compute for each edge (not in T) the x-**span**, defined by sub-tree size in the dual spanning tree





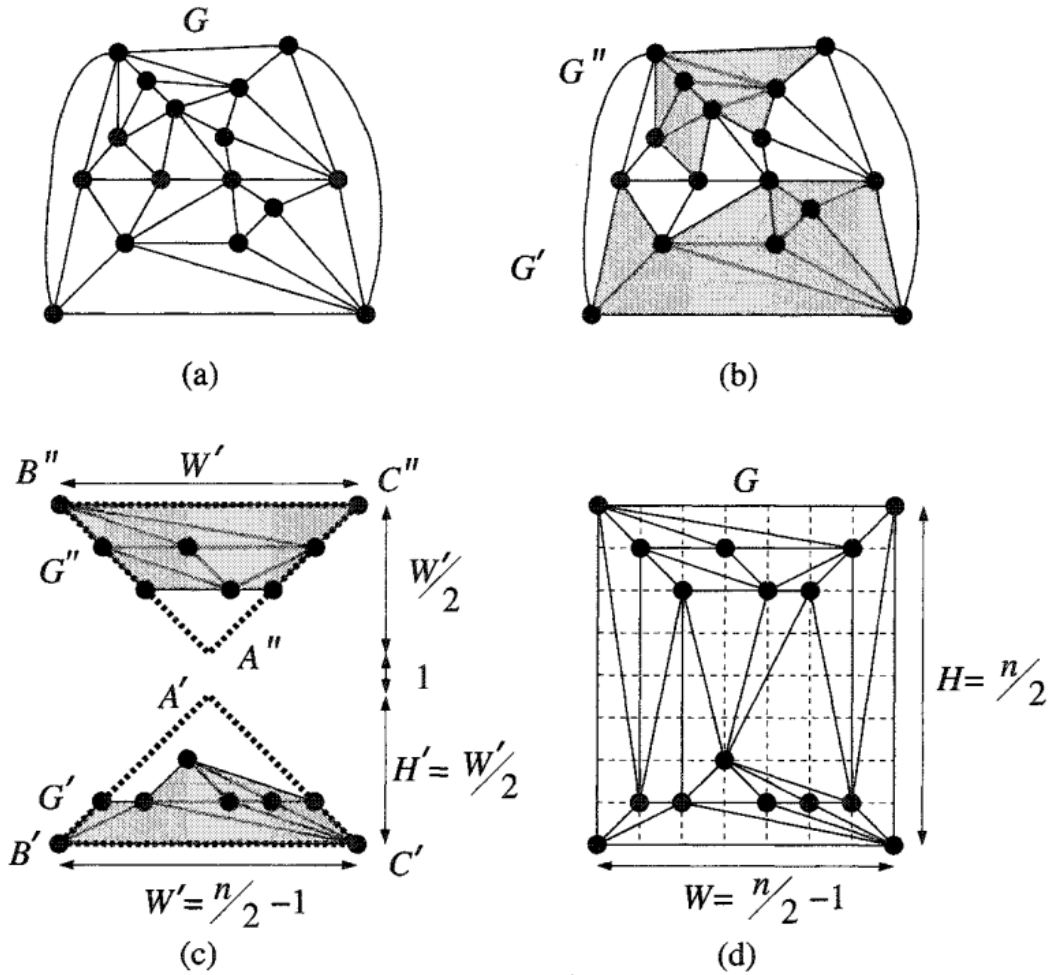




Again on Canonical Orderings

(variants and applications)

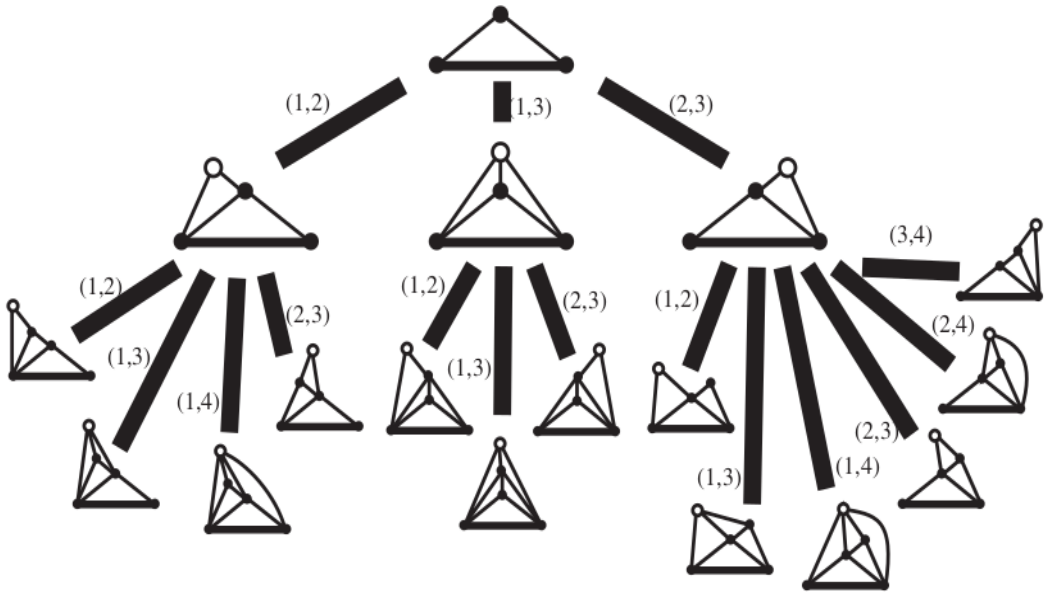
Drawing 4-connected planar triangulations



Theorem

A planar 4-connected triangulation (with at least four vertices on the bounday), admits a straight-line on a grid of size $\frac{n}{2} \times \frac{n}{2}$

Fast enumeration of planar triangulations



[Nakano et al.]

Procedure find-all-child-triangulations(G)
begin
  output G { Output the difference from the previous triangulation}
  if G has exactly n vertices **then return**
  for $i = 1$ **to** $s - 1$
    for $j = i + 1$ **to** s
      find-all-child-triangulations$(G(i, j))$ { Case 1}
  for $i = 1$ **to** $s - 1$
    for $j = s + 1$ **to** $q(i)$
      find-all-child-triangulations$(G(i, j))$ { Case 2}
  find-all-child-triangulations$(G(s, s + 1))$ { Case 3}
end

Algorithm find-all-triangulations(T_3)
begin
  output K_3
  $G = K_3$
  find-all-child-triangulations$(G(1, 2))$
  find-all-child-triangulations$(G(2, 3))$
  find-all-child-triangulations$(G(1, 3))$
end

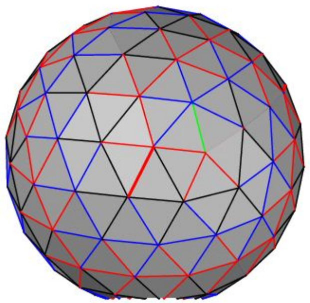
MPRI 2-38-1: Algorithms and combinatorics for geometric graphs

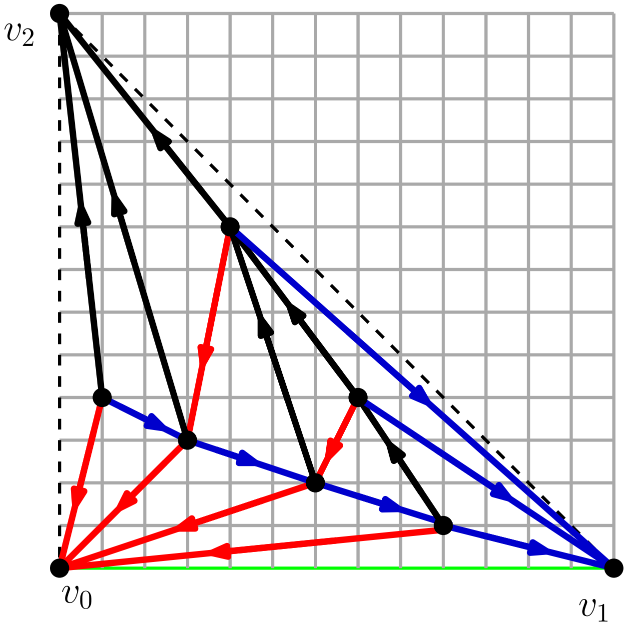
Lecture 4

Chapter II: Schnyder woods

october 9, 2024

Luca Castelli Aleardi





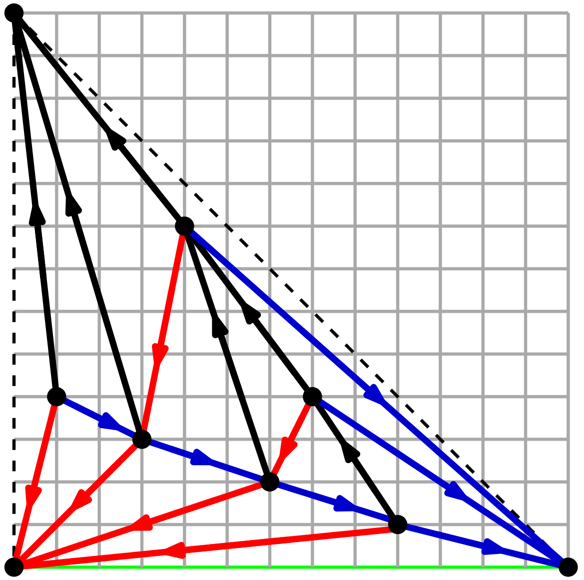
Some facts about planar graphs

("As I have known them")

Some facts about planar graphs

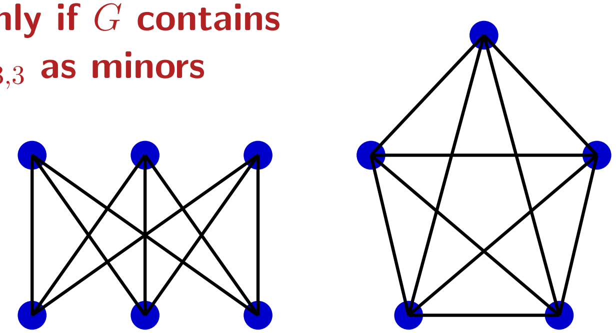
Thm (Schnyder, Trotter, Felsner)

G **planar if and only if** $dim(G) \leq 3$



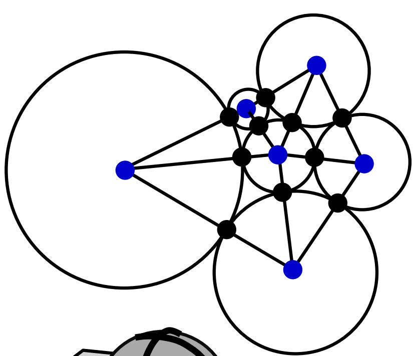
Thm (Kuratowski, excluded minors)

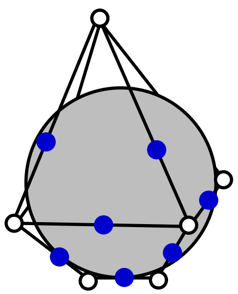
G **planar if and only if** G **contains neither** K_5 **nor** $K_{3,3}$ **as minors**

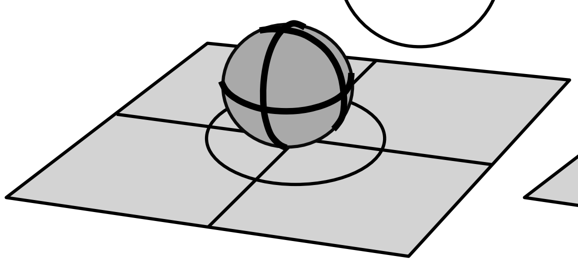


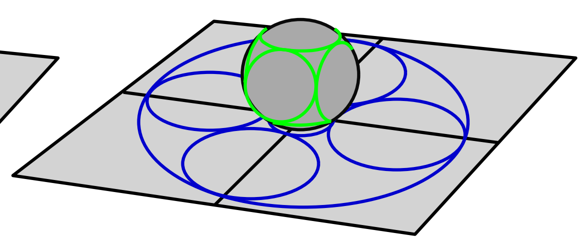
Thm (Koebe-Andreev-Thurston)

Every planar graph with n **vertices is isomorphic to the intersection graph of** n **disks in the plane.**





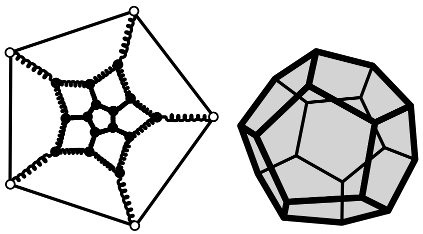




Thm (Tutte)

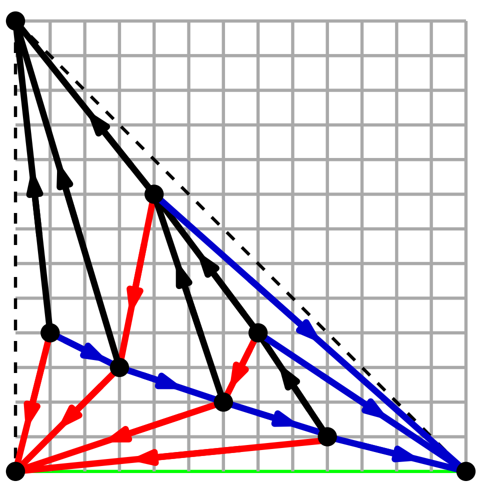
$$E(\rho) := \sum_{(i,j)\in E} |\mathbf{x}(v_i) - \mathbf{x}(v_j)|^2 = \sum_{(i,j)\in E} (x_i - x_j)^2 + (y_i - y_j)^2$$

$$\mathbf{x}(v_i) = \sum_{j\in\mathcal{N}(i)} \frac{1}{deg(v_i)} \mathbf{x}(v_j)$$



Straight-line planar drawings of planar graphs

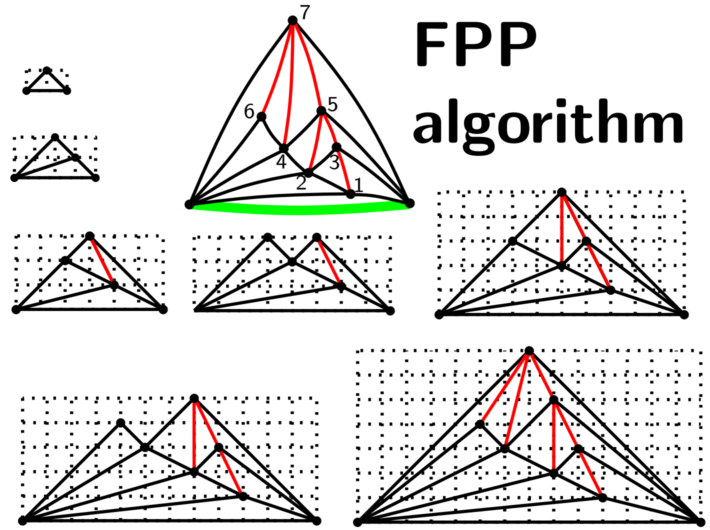
Thm (Schnyder 1990)



face counting via Schnyder woods

Thm (De Fraysseix, Pack Pollack 1989)

FPP algorithm



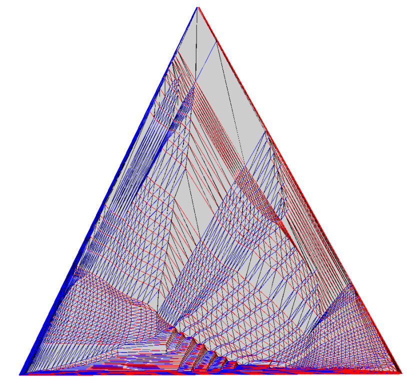
shift algorithm via Canonical orderings

linear time algorithms

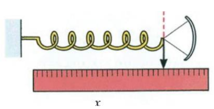
$O(n) \times O(n)$ **grid drawings**

not trivial to implement

extremey fast: they can process millions of vertices per second



Spring embedder (Eades, 1984)
(Fruchterman and Reingold, 1991)



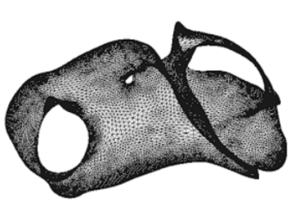
force-directed paradigm

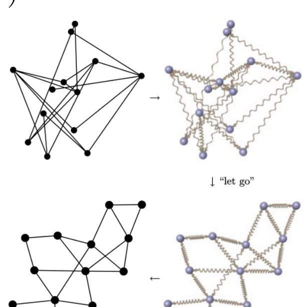
easy to implement

pretty slow: $O(n^2)$ or $O(n \log n)$ time per iteration

$$\mathbf{F}_a(v) = c_1 \cdot \sum_{(u,v) \in E} \log(dist(u,v)/c_2)$$

$$\mathbf{F}_r(v) = c_3 \cdot \sum_{u \in V} \frac{1}{\sqrt{dist(u,v)}}$$





images from Kaufman Wagner (Springer, 2001)

[Tutte'63] Tutte barycentric embedding

minimize the spring energy

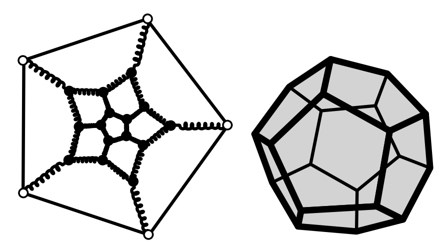
$$E(\rho) := \sum_{(i,j) \in E} |\mathbf{x}(v_i) - \mathbf{x}(v_j)|^2 = \sum_{(i,j) \in E} (x_i - x_j)^2 + (y_i - y_j)^2$$

solve large sparse linear systems

$$\mathbf{x}(v_i) = \sum_{j \in \mathcal{N}(i)} \frac{1}{deg(v_i)} \mathbf{x}(v_j)$$

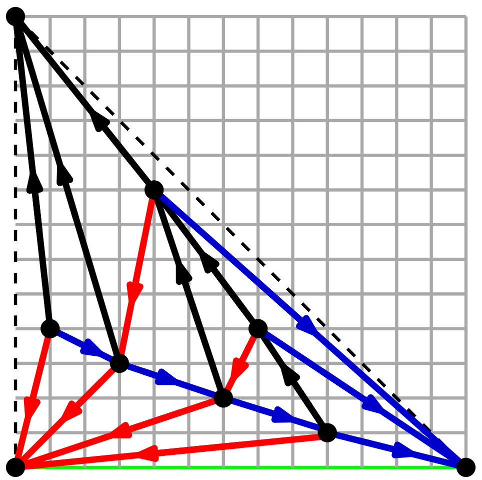
easy to implement

not very fast: they can process $\approx 10^4$ vertices per second



Straight-line planar drawings of planar graphs

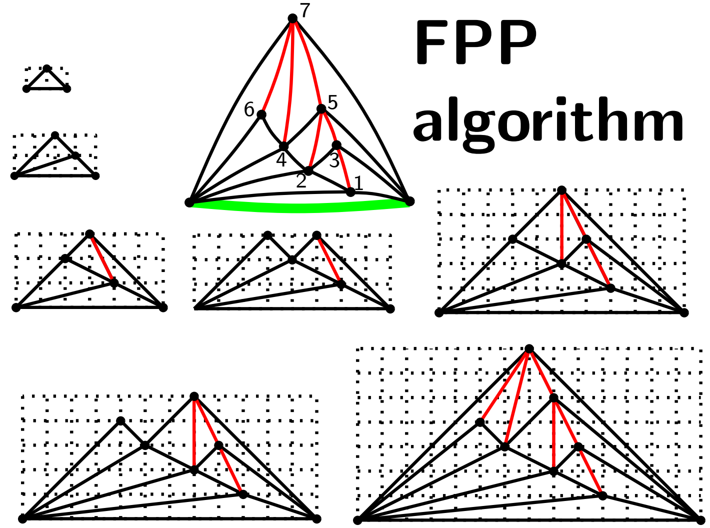
Thm (Schnyder 1990)



face counting via Schnyder woods

Thm (De Fraysseix, Pack Pollack 1989)

FPP algorithm



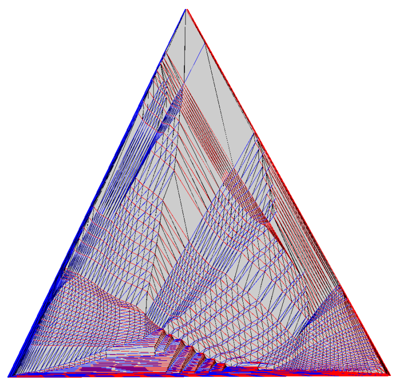
shift algorithm via Canonical orderings

linear time algorithms

$O(n) \times O(n)$ **grid drawings**

not trivial to implement

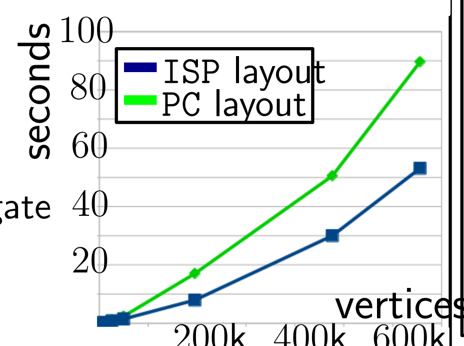
extremey fast: they can process millions of vertices per second



Timing performances

Schnyder drawing or **FPP algorithm**:
less than 1 second
(Java, 2.66GHz Intel i7 CPU)

Chinese dragon ($655k$ vert.)



solve sparse linear systems with the conjugate gradient solver of `MTJ` (Java) library

(numeric precision 10^{-6})

[Tutte'63] Tutte barycentric embedding

minimize the spring energy

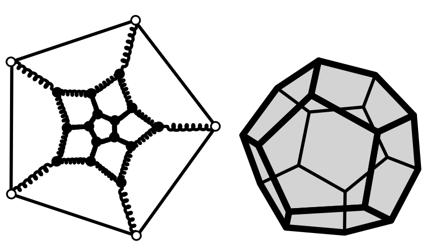
$$E(\rho) := \sum_{(i,j)\in E} |\mathbf{x}(v_i) - \mathbf{x}(v_j)|^2 = \sum_{(i,j)\in E} (x_i - x_j)^2 + (y_i - y_j)^2$$

solve large sparse linear systems

$$\mathbf{x}(v_i) = \sum_{j \in \mathcal{N}(i)} \frac{1}{deg(v_i)} \mathbf{x}(v_j)$$

easy to implement

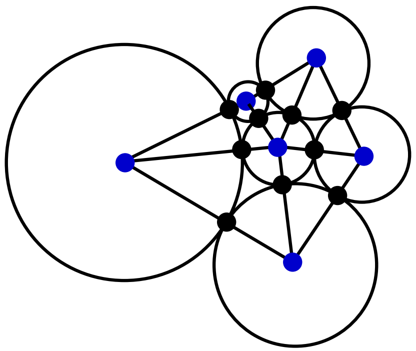
not very fast: they can process $\approx 10^4$ vertices per second

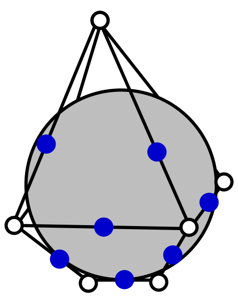


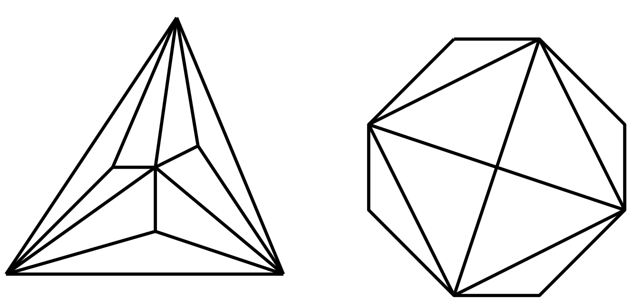
Using circles to measure distances

Thm (Koebe-Andreev-Thurston)

Every planar graph with n vertices is isomorphic to the intersection graph of n disks in the plane.

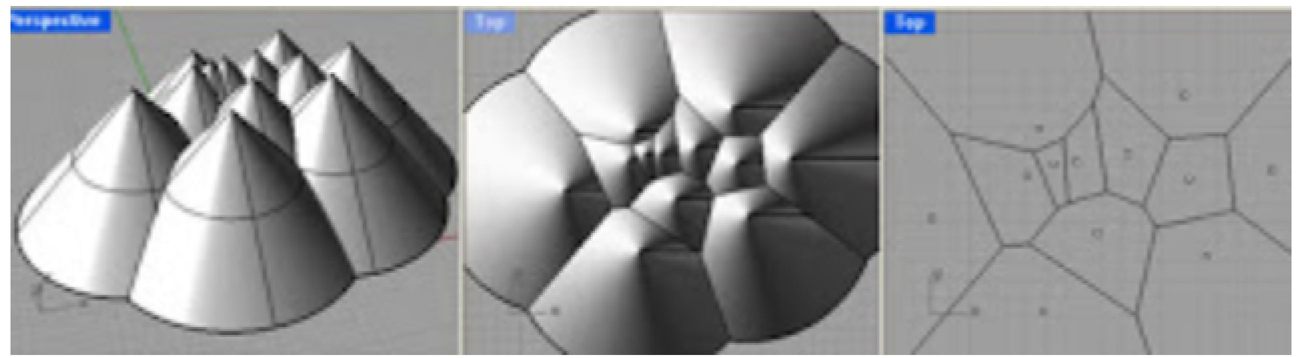




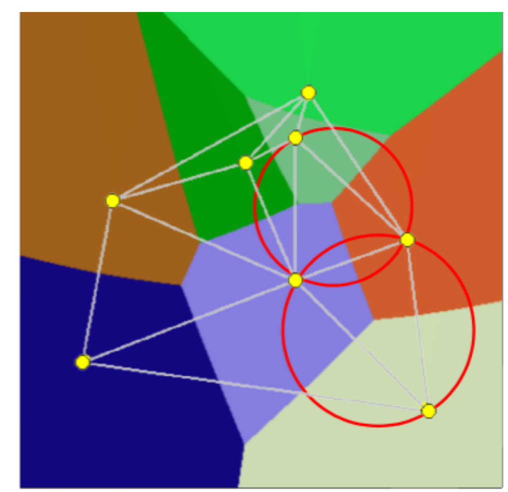


Not every planar triangulation is Delaunay realizable

(images by R. Silveira)



(images by N. Bonichon)

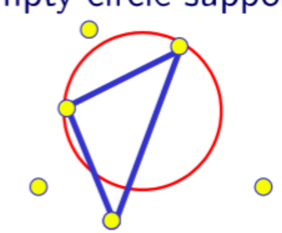


Voronoï cell:
$C(s_i) = \{x / d(s_i, x) \leq d(s_j, x) \forall i \neq j\}$

Delaunay Triangulation:
s_i is a neighbour s_j i f f $C(s_i) \cap C(s_j) \neq \emptyset$

General Position:
No 3 points collinear
No 4 points co_circular.

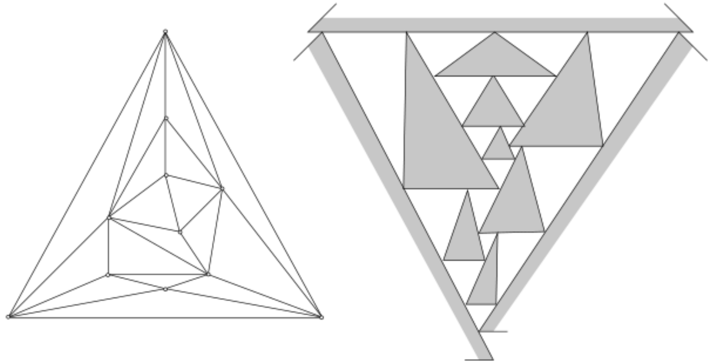
Alternative def: There is an edge (s_i, s_j) iif there is an empty circle supporting s_i and s_j.



$\Rightarrow$: each face is supported by an empty circle.

Using triangles to measure distances

Thm (de Fraysseix, Ossona de Mendez, Rosenstiehl, '94)



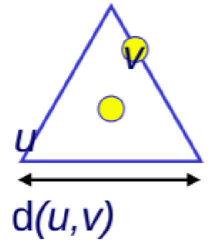
(images by S. Felsner)

Chew, '89

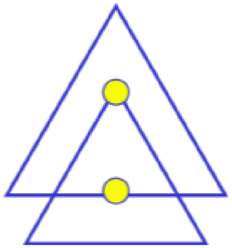
TD-Delaunay: triangular distance Delaunay triangulations

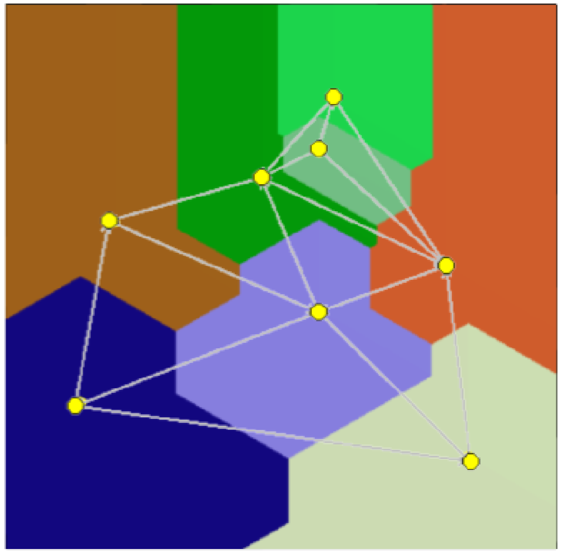
- Distance triangulaire :

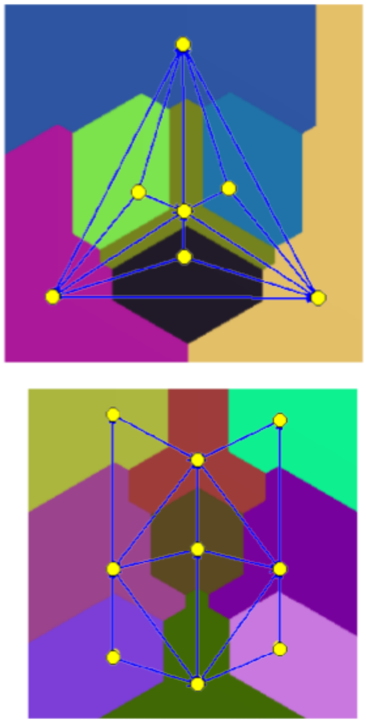
  d(u,v) = taille du plus petit triangle équilatérale à base horizontale centré en u contenant v.



- Rq : d(u,v) ≠ d(v,u) en général







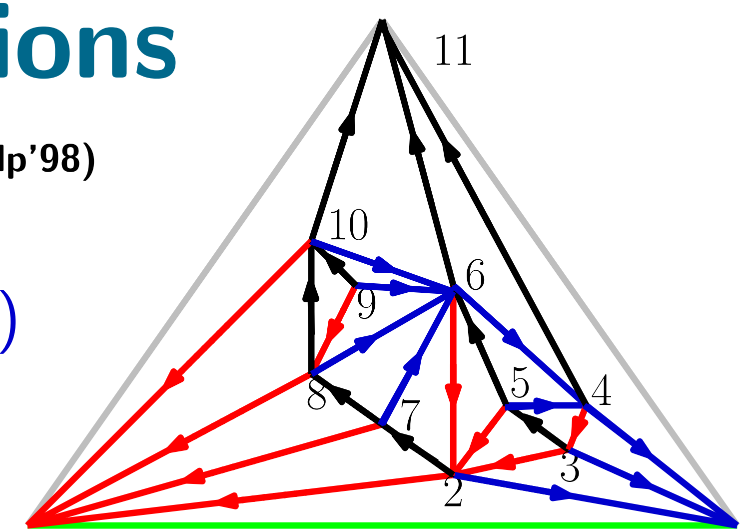
Every planar triangulation is TD-Delaunay realizable

(images by N. Bonichon)

Schnyder woods and canonical orderings: overview of applications

(**graph drawing**, **graph encoding**, succinct representations, compact data structures, exhaustive graph enumeration, bijective counting, greedy drawings, spanners, contact representations, planarity testing, untangling of planar graphs, Steinitz representations of polyhedra, ...)

Some (classical) applications



(Chuang, Garg, He, Kao, Lu, Icalp'98)

(He, Kao, Lu, 1999)

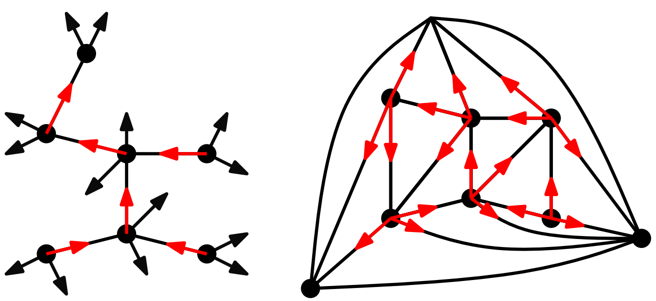
Graph encoding ($4n$ nits)

$\overline{T}_0$ ( ) ((( )) ( ) ( )) ( ) (( )) ( ) ( )

$\overline{T}_2$ 00000101010100110111

(Poulalhon-Schaeffer, Icalp 03)

bijective counting, random generation

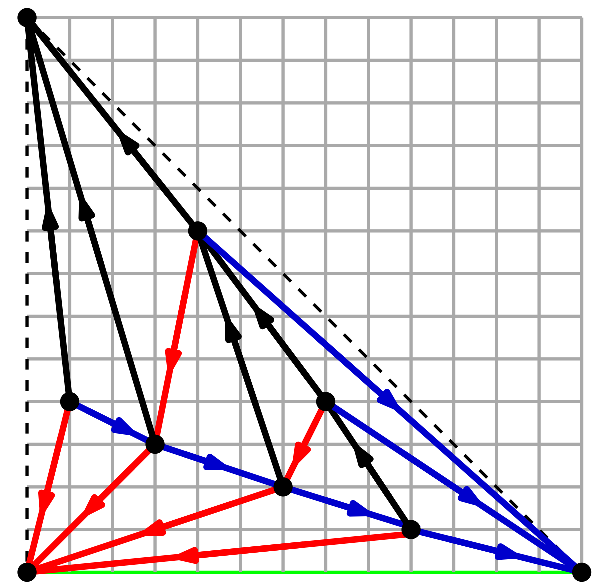


$$\boxed{c_n = \frac{2(4n+1)!}{(3n+2)!(n+1)!}}$$

$\Rightarrow$ optimal encoding ≈ 3.24 bits/vertex

Thm (Schnyder '90)

Planar straight-line grid drawing (on a $O(n \times n)$ grid)



More ("recent") applications

Schnyder woods, TD-Delaunay graphs, orthogonal surfaces and Half-Θ_6-graphs

[Bonichon et al., WG'10, Icalp '10, ...]

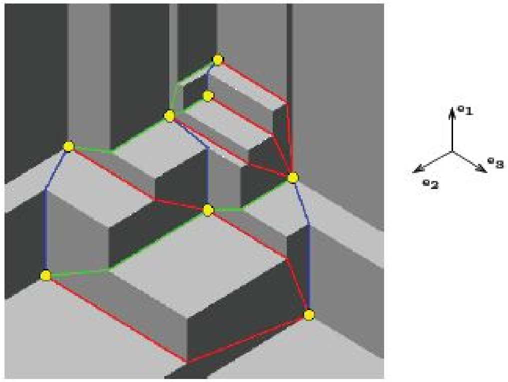
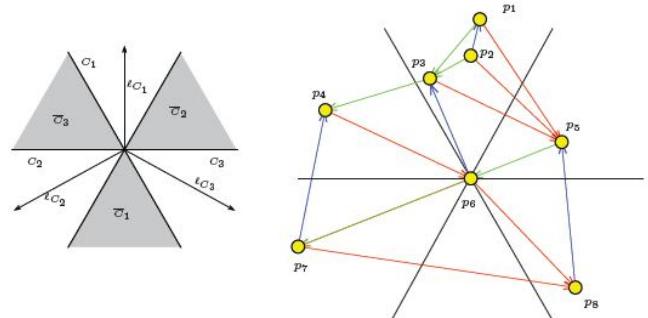


Figure 2: A coplanar orthogonal surface with its geodesic embedding.



(a) (b)

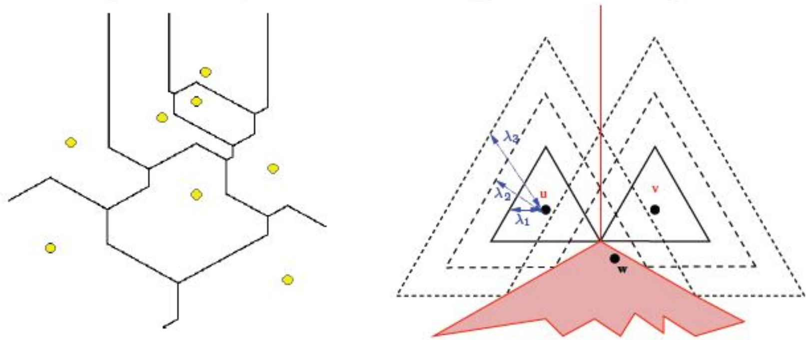
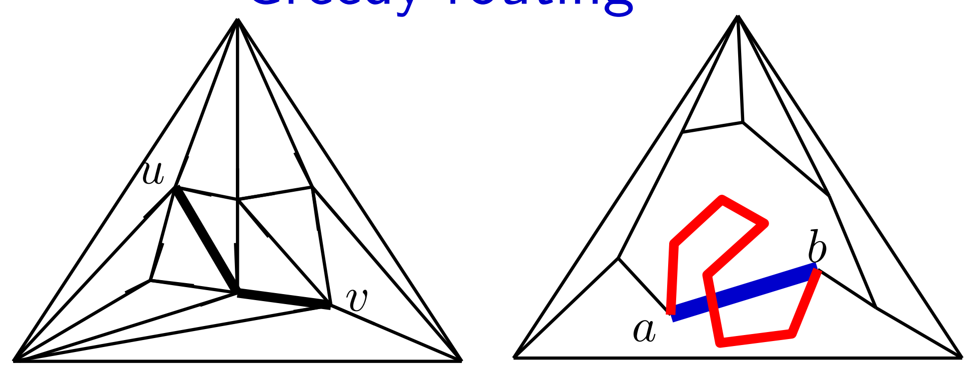


Figure 3: (a) TD-Voronoi diagram. (b) $\lambda_1 < \lambda_2 < \lambda_3$ stand for three triangular distances. Set $\{u, v\}$ is an ambiguous point set, however $\{u, v, w\}$ is non-ambiguous.

Greedy routing



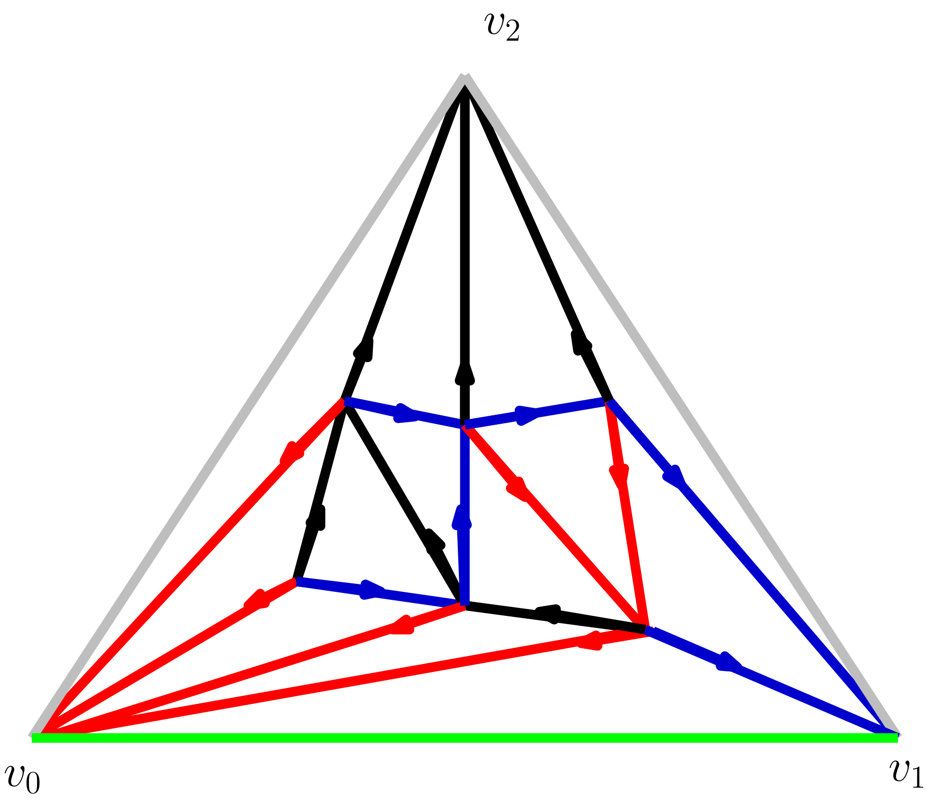
Every planar triangulation admits a **greedy drawing** (Dhandapani, Soda08)

(conjectured by Papadimitriou and Ratajczak for 3-connected planar graphs)

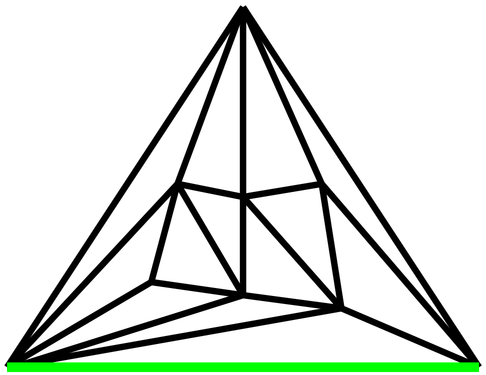
Schnyder woods

(definitions)

Schnyder woods (for triangulations): definition



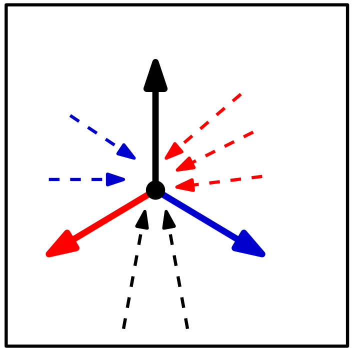
[Schnyder '90]



rooted triangulation on n nodes

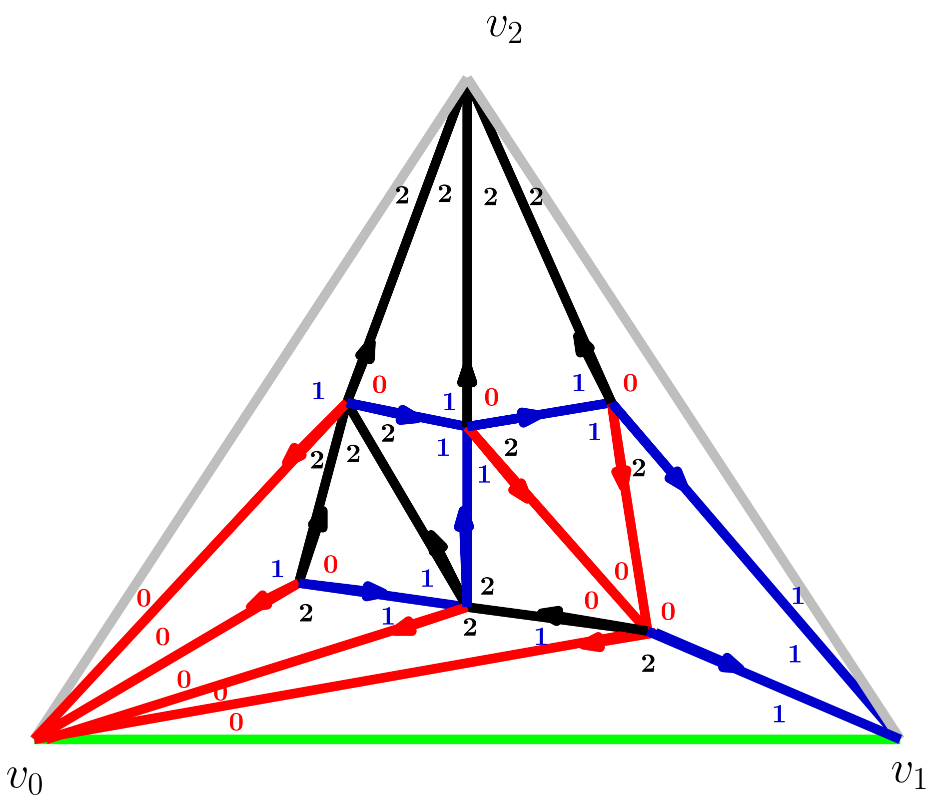
A Schnyder wood of a (rooted) planar triangulation is partition of all inner edges into three sets T_0, T_1 and T_2 such that

i) edge are colored and oriented in such a way that each inner nodes has exaclty one outgoing edge of each color

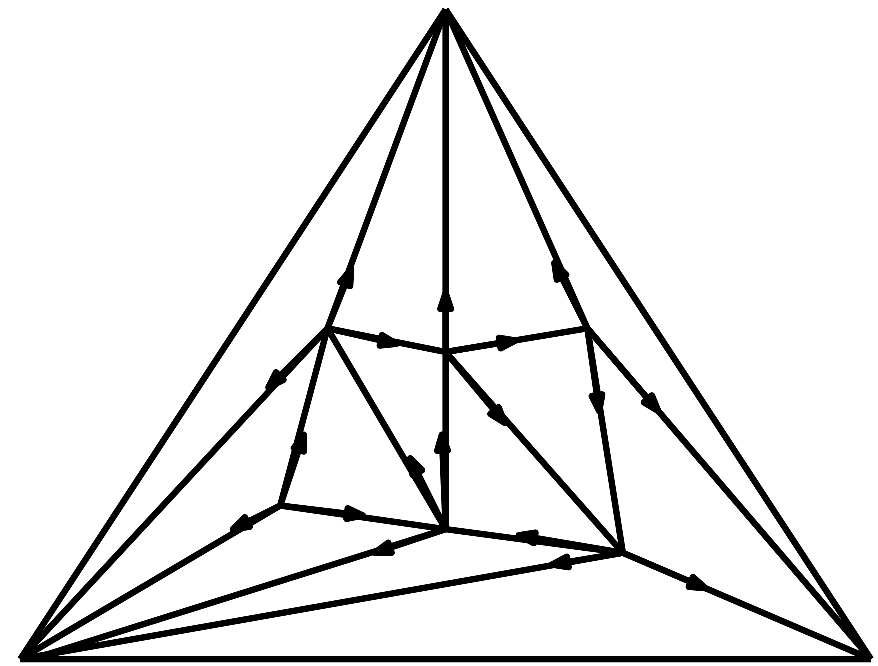


ii) colors and orientations around each inner node must respect the local Schnyder condition

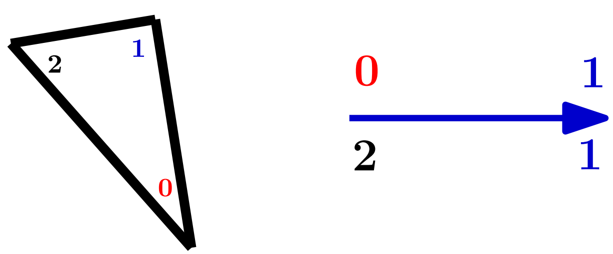
Schnyder woods: equivalent formulations



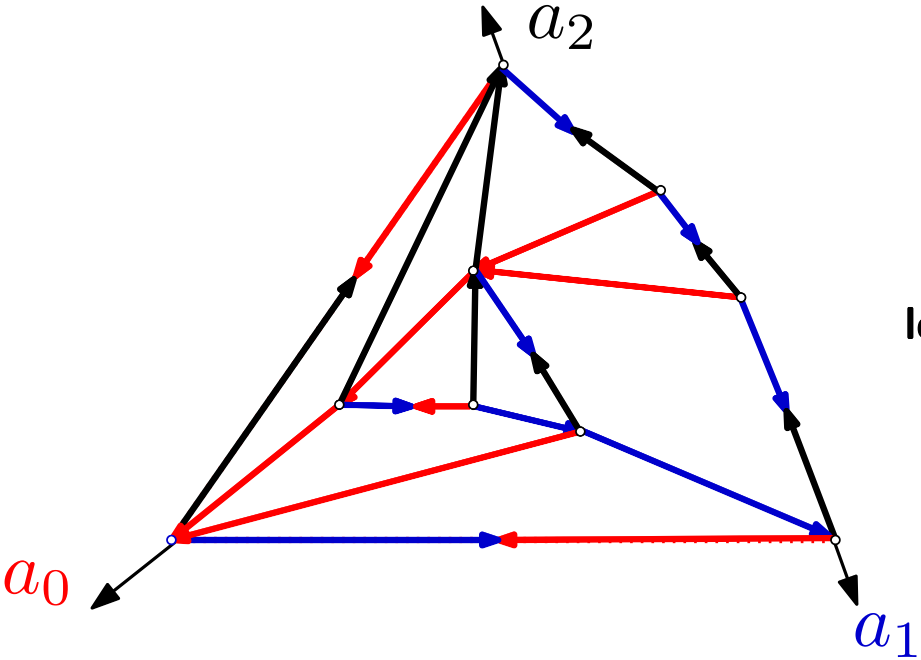
[Schnyder labeling]

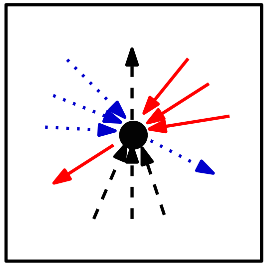


[3-orientation]



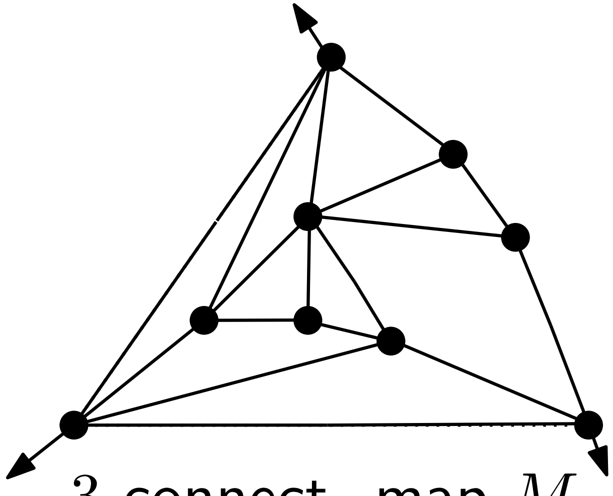
Schnyder woods (3-connected maps): definition





local Schnyder rule

3-connected graphs [Felsner]



3-connect. map M

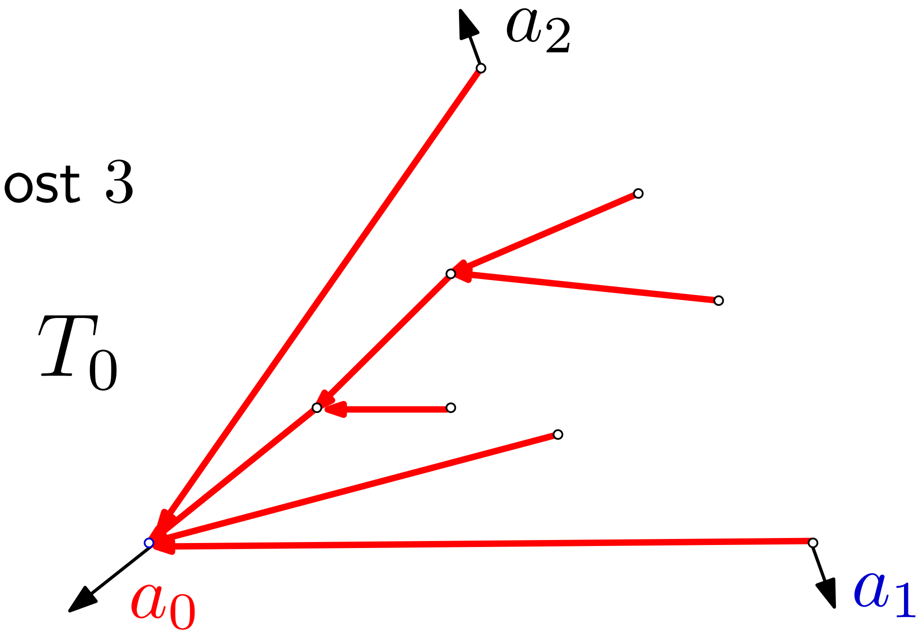
More details: next Lecture

Schnyder woods: spanning property

Theorem **[Schnyder '90]**

$T_i :=$ digraph defined by directed edges of color i

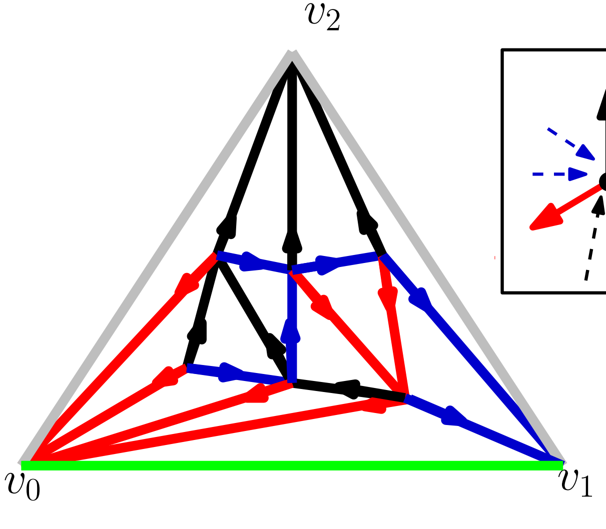
The three sets T_0, T_1, T_2 are spanning trees of the inner vertices of $\mathcal{T}$ (each rooted at vertex v_i)

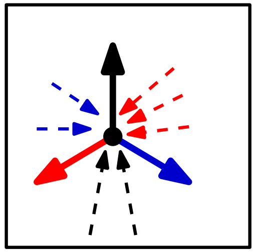


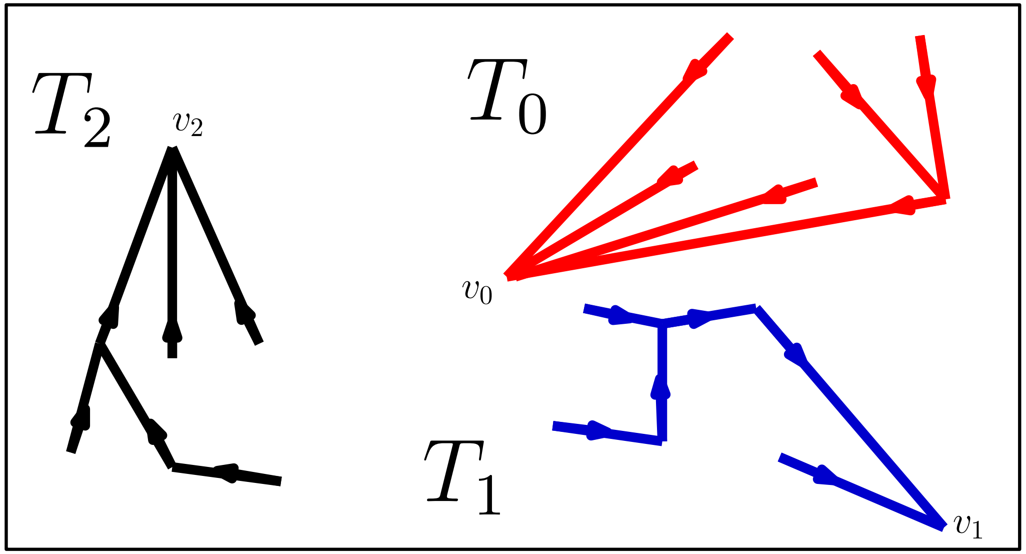
Remark

Planar graphs have **arboricity** at most 3

(minimum number of edge-disjoint spanning forests)







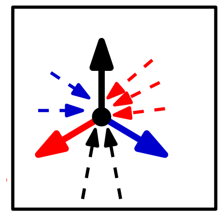
Spanning property for triangulations

Theorem **[Schnyder '90]**
The three sets T_0, T_1, T_2 are spanning trees of the inner vertices of $\mathcal{T}$ (each rooted at vertex v_i)

proof (use a counting argument)

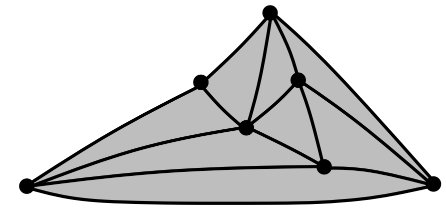
Claim 1: T_i does not contain cycles
(assume there are monochromatic cycles, by contradiction)

local Schnyder rule

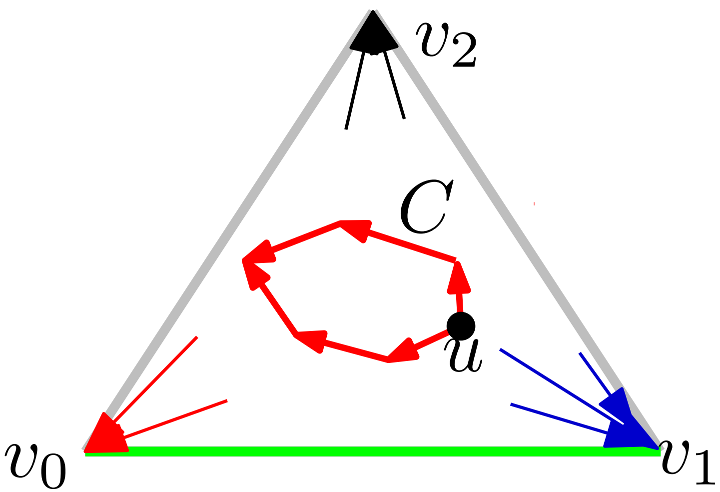


Triangulations with a boundary

$$f_i = 2n_i + k - 2$$
$$e_i = 3n_i + (k-3)$$

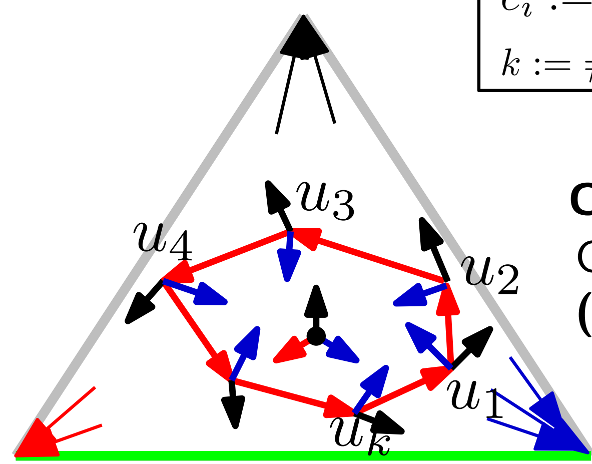


$n_i :=$ #inner vertices
$e_i :=$ # inner edges
$k :=$ #boundary edges=#boundary vertices

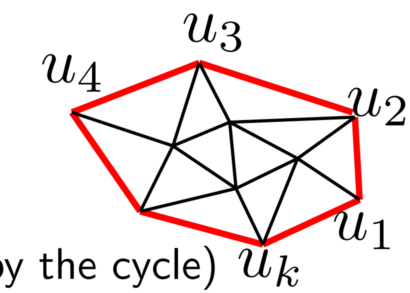


Case 1: C:=non oriented monochromatic cycle of size k

there is a vertex u violating Schnyder rule



Case 2:
C:=monochromatic cycle of size k
(cw or ccw) oriented



Schnyder local rule implies:
(count edges in the triangulation bounded by the cycle)

$$\boxed{e_i = 3n_i + k}$$

3 outgoing edges for inner vertices

1 outgoing edges for boundary vertices

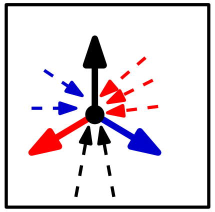
Spanning property for triangulations

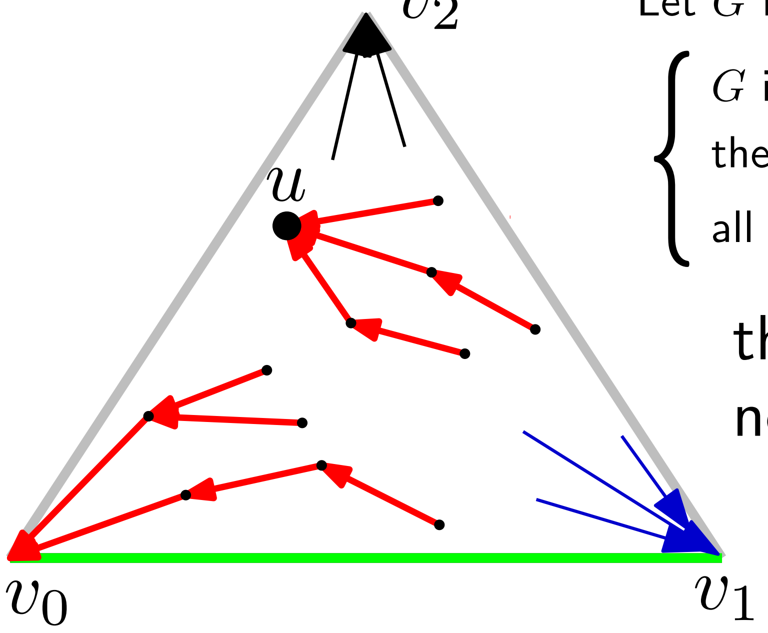
Theorem **[Schnyder '90]**
The three sets T_0, T_1, T_2 are spanning trees of the inner vertices of $\mathcal{T}$ (each rooted at vertex v_i)

proof (use a counting argument)

Claim 2: T_i is connected
(by contradiction, assume there are several disjoint components)

local Schnyder rule





Let G be a connected component not containing v_i

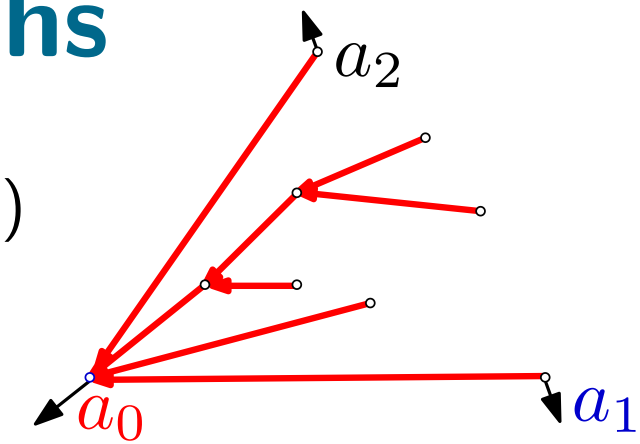
- G is connected and without cycles
- then G is a tree: $|G|$ vertices and $|G|-1$ edges
- all vertices of G are inner vertices (distinct from v_0, v_1 and v_2)

there is a vertex $u \in G$ violating Schnyder rule: no outgoing edge of color i

Non crossing paths

Corollary:

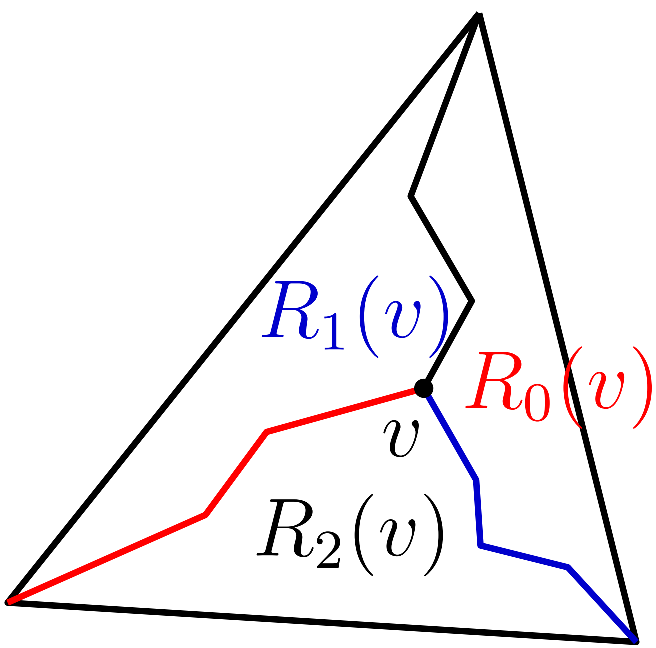
Each sets T_i is spanning tree $\mathcal{M}$ (rooted at vertex a_i)

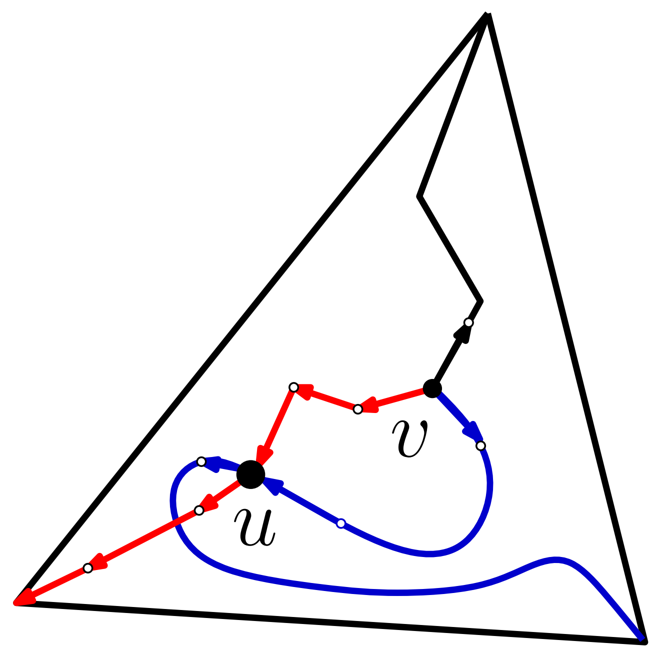


Corollary

For each inner vertex v the three monochromatic paths P_0, P_1, P_2 directed from v toward each vertex a_i are vertex disjoint (except at v) and partition the inner faces into three sets $R_0(v), R_1(v), R_2(v)$

proof: (by contradiction)

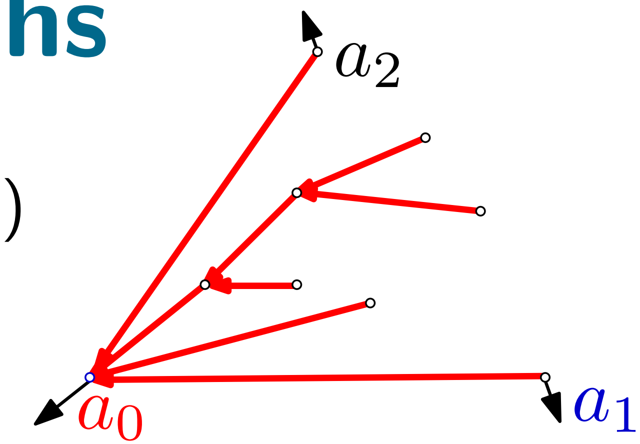




Non crossing paths

Corollary:

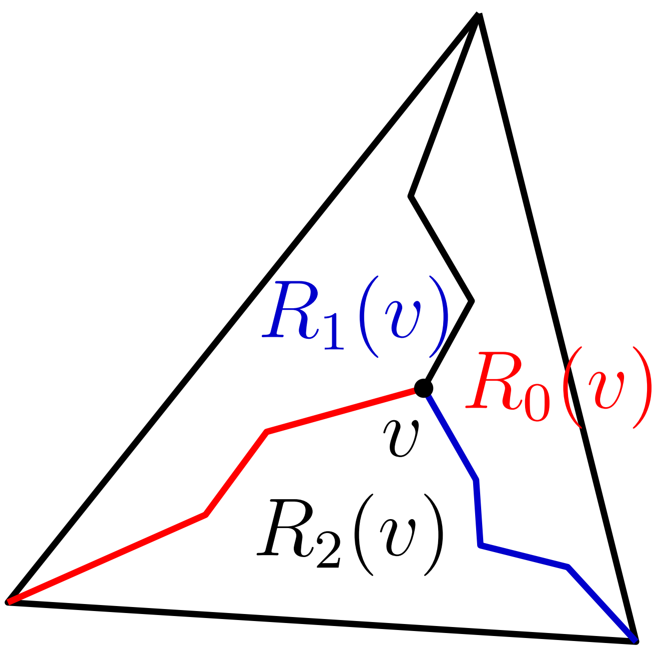
Each sets T_i is spanning tree $\mathcal{M}$ (rooted at vertex a_i)



Corollary

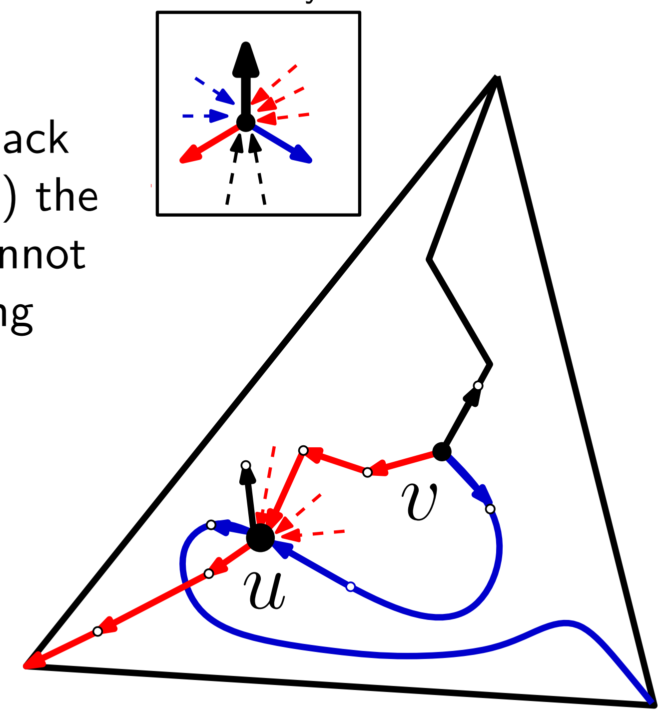
For each inner vertex v the three monochromatic paths P_0, P_1, P_2 directed from v toward each vertex a_i are vertex disjoint (except at v) and partition the inner faces into three sets $R_0(v), R_1(v), R_2(v)$

proof: the existence of two paths $P_i(v)$ and $P_{i+1}(v)$ which are crossing would contradicts previous theorem



Remark: the outgoing black is just after (in ccw order) the last ingoing red and it cannot be followed by an outgoing blue edge

local Schnyder rule



Consequences

Efficient graph data structure for planar graphs

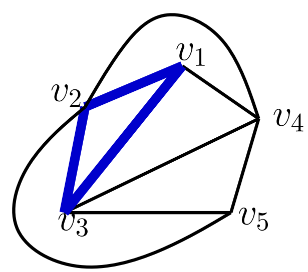
There exist a (simple) data structure of size $O(n \log n)$ bits supporting constant time adjacency test between vertices

Adjacency matrix

$$A_G[i,j] = \begin{cases} 1 & v_i \text{ adjacent } v_j \\ 0 & \text{otherwise} \end{cases} \qquad \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

space: $O(n^2)$ bits

adjacency: $O(1)$ time



$$\begin{array}{c} d_i \quad \text{neighbors} \\ \left[\begin{array}{c:cccc} 3 & 2 & 3 & 4 & \\ 4 & 1 & 4 & 5 & 3 \\ 4 & 5 & 4 & 1 & 2 \\ & & \ldots & & \end{array}\right] \end{array} \begin{array}{l} v_1 \\ v_2 \\ \\ \\ \end{array}$$

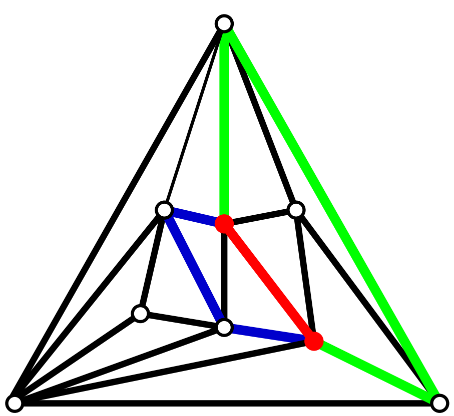
Adjacency lists

space: $O(n \log n)$ bits

adjacency: $O(n)$ time

Menger theorem for planar triangulations

Schnyder woods allows us to compute in linear time, for any pair of vertices (u, v), 3 vertex disjoint paths between u and v



Thm (Menger)

If G is k-connected, then for each pair u, vthere exist k disjoint path from u to v

Consequences

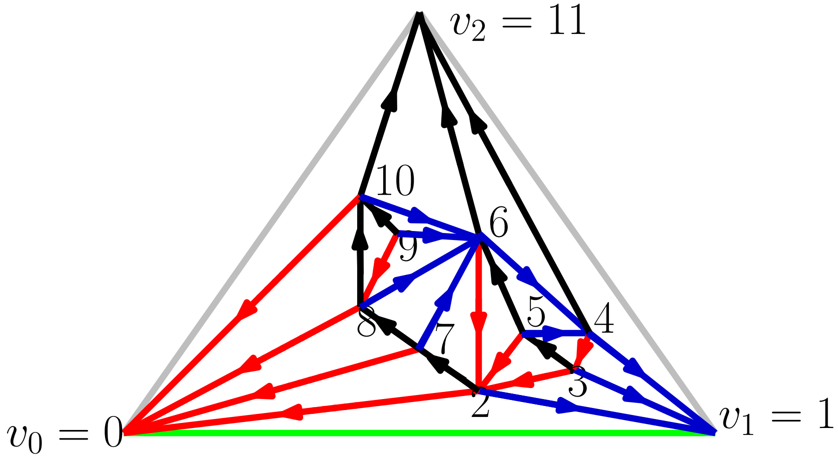
Efficient graph data structure for planar graphs

There exist a (simple) data structure of size $O(n \log n)$ bits supporting constant time adjacency test between vertices

Truncated adjacency lists: store only 3 successors

neighbors (outgoing edges)

$$\begin{array}{c} \\ v_7 \\ v_8 \\ v_9 \\ \\ \end{array} \begin{bmatrix} & \cdots & \\ 0 & 6 & 8 \\ 0 & 6 & 10 \\ 6 & 8 & 10 \\ & \cdots & \end{bmatrix}$$

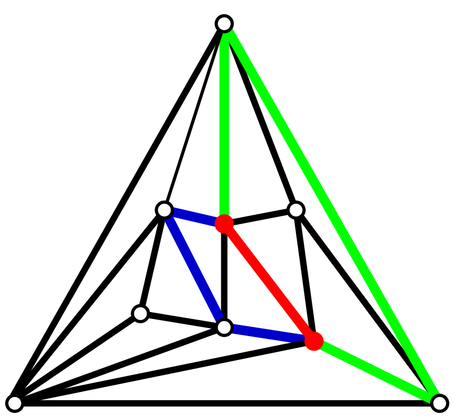


space: $O(n \log n)$ bits

adjacency: $O(1)$ time

Menger theorem for planar triangulations

Schnyder woods allows us to compute in linear time, for any pair of vertices (u, v), 3 vertex disjoint paths between u and v



Consequences

Efficient graph data structure for planar graphs

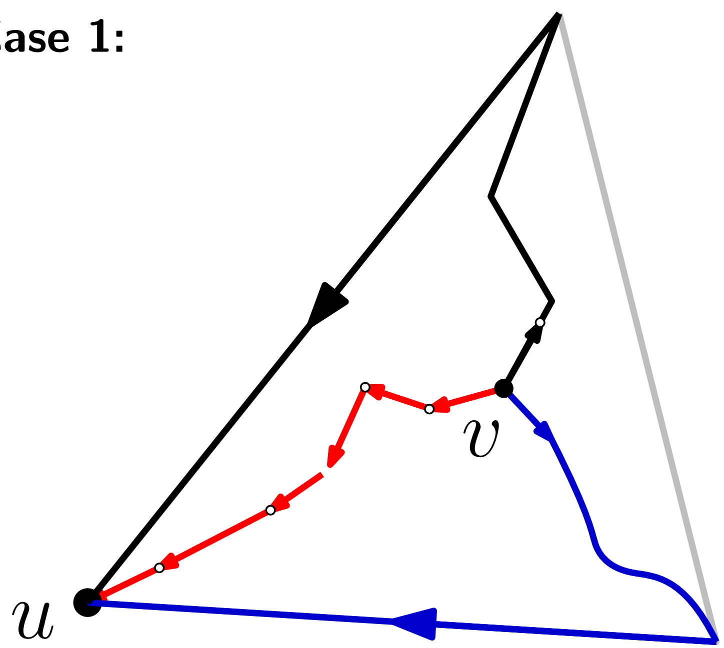
There exist a (simple) data structure of size $O(n \log n)$ bits supporting constant time adjacency test between vertices

Truncated adjacency lists: store only 3 successors

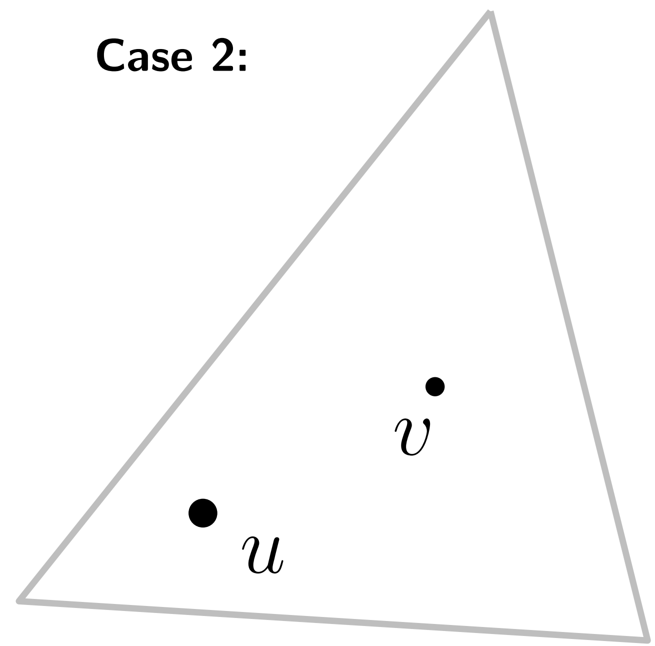
Menger theorem for planar triangulations

Schnyder woods allows us to compute in linear time, for any pair of vertices (u, v), 3 vertex disjoint paths between u and v

Case 1:



Case 2:



Consequences

Efficient graph data structure for planar graphs

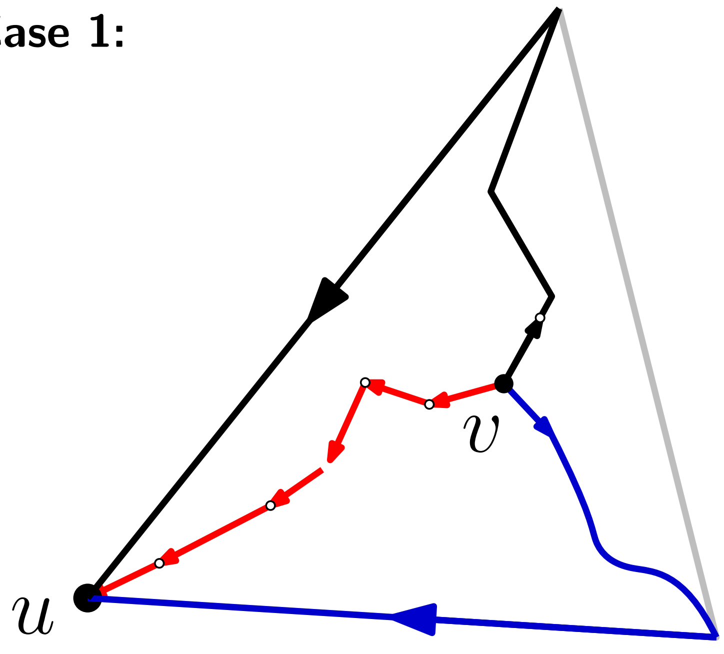
There exist a (simple) data structure of size $O(n \log n)$ bits supporting constant time adjacency test between vertices

Truncated adjacency lists: store only 3 successors

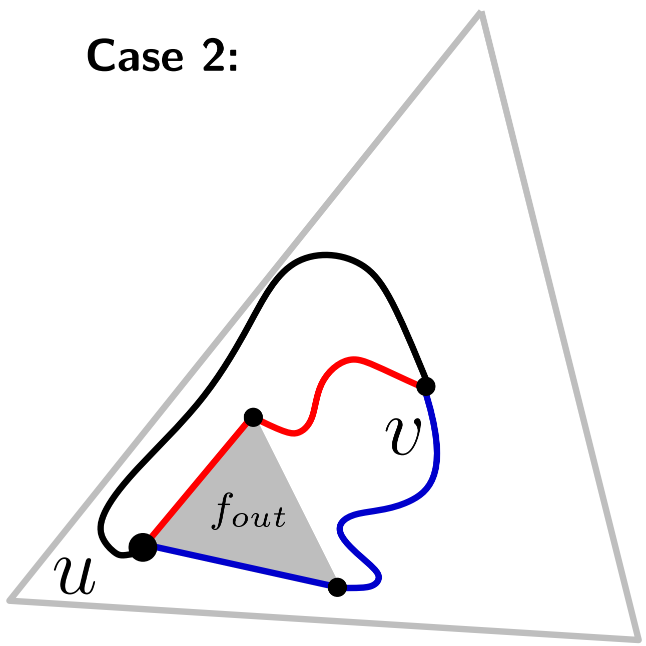
Menger theorem for planar triangulations

Schnyder woods allows us to compute in linear time, for any pair of vertices (u, v), 3 vertex disjoint paths between u and v

Case 1:



Case 2:



Number and structure of Schnyder woods

Counting Schnyder woods: (there are grahs admitting an exponential number)

[Bonichon '05]

\# Schnyder woods of triangulations of size n: $\approx 16^n$
(all Schnyder woods over all distinct triangulations of size n)

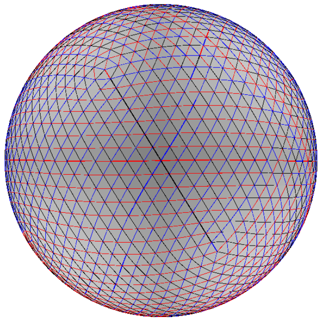
[Felsner Zickfeld '08]

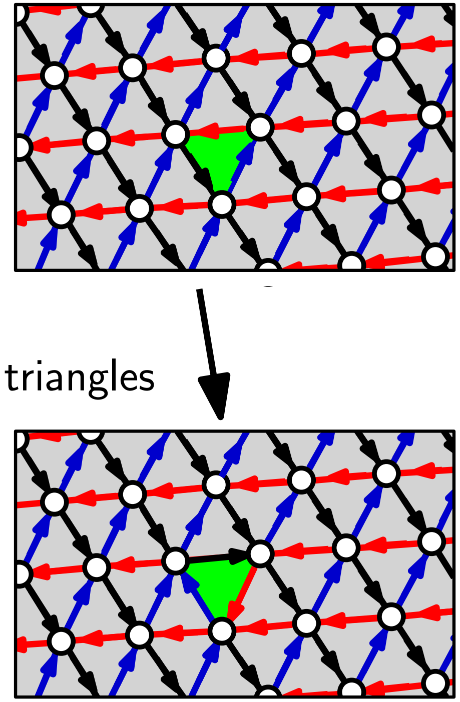
(count of Schnyder woods of a fixed triangulation)

$$2.37^n \leq \max_{T \in \mathcal{T}_n} |SW(T)| \leq 3.56^n$$

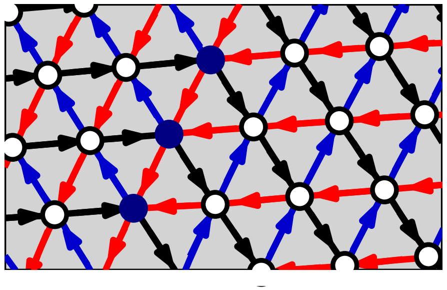
$\mathcal{T}_n :=$ class of planar triangulations of size n

$SW(T) :=$ set of all Schnyder woods of the triangulation T



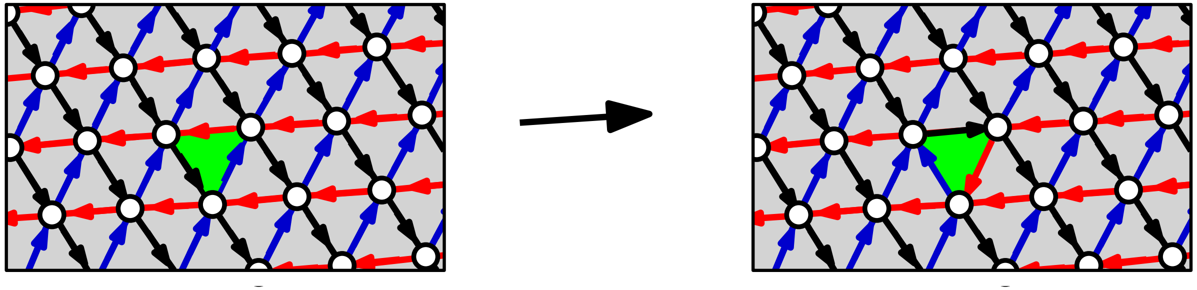


reversal of oriented triangles

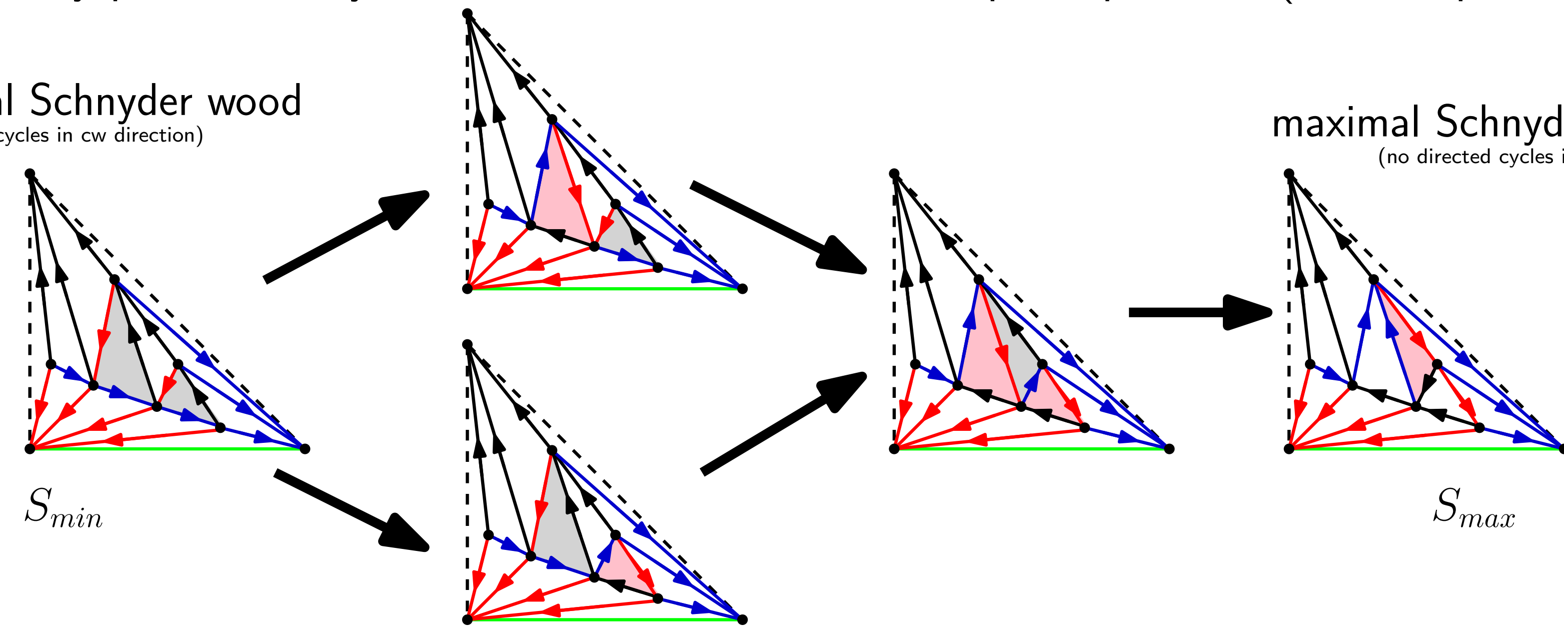


Exercice: there exists a class of planar triangulations admitting a unique Schnyder wood. Which one?

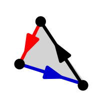
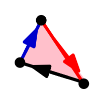
Structure of Schnyder woods: distributive lattice



Thm: [Ossona de Mendez'94], [Felsner'03]
The set $\mathcal{S}(T)$ of all distinct Schnyder woods of a given triangulation T defines a connected graph with respect to the **flip** operation. Furthermore, this set has a **lattice** structure: a partial order such that for every pair of Schnyder woods of T there is an unique supremum (and unique infimum).



$S_a < S_b$ iff S_a can be obtained by S_b by flipping ccw directed cycles

Flip:=  to 

reversal of directed cycles (cycles could bound several faces)

The min is the unique $S_{min} \in \mathcal{S}(T)$ with **no clockwise circuit**

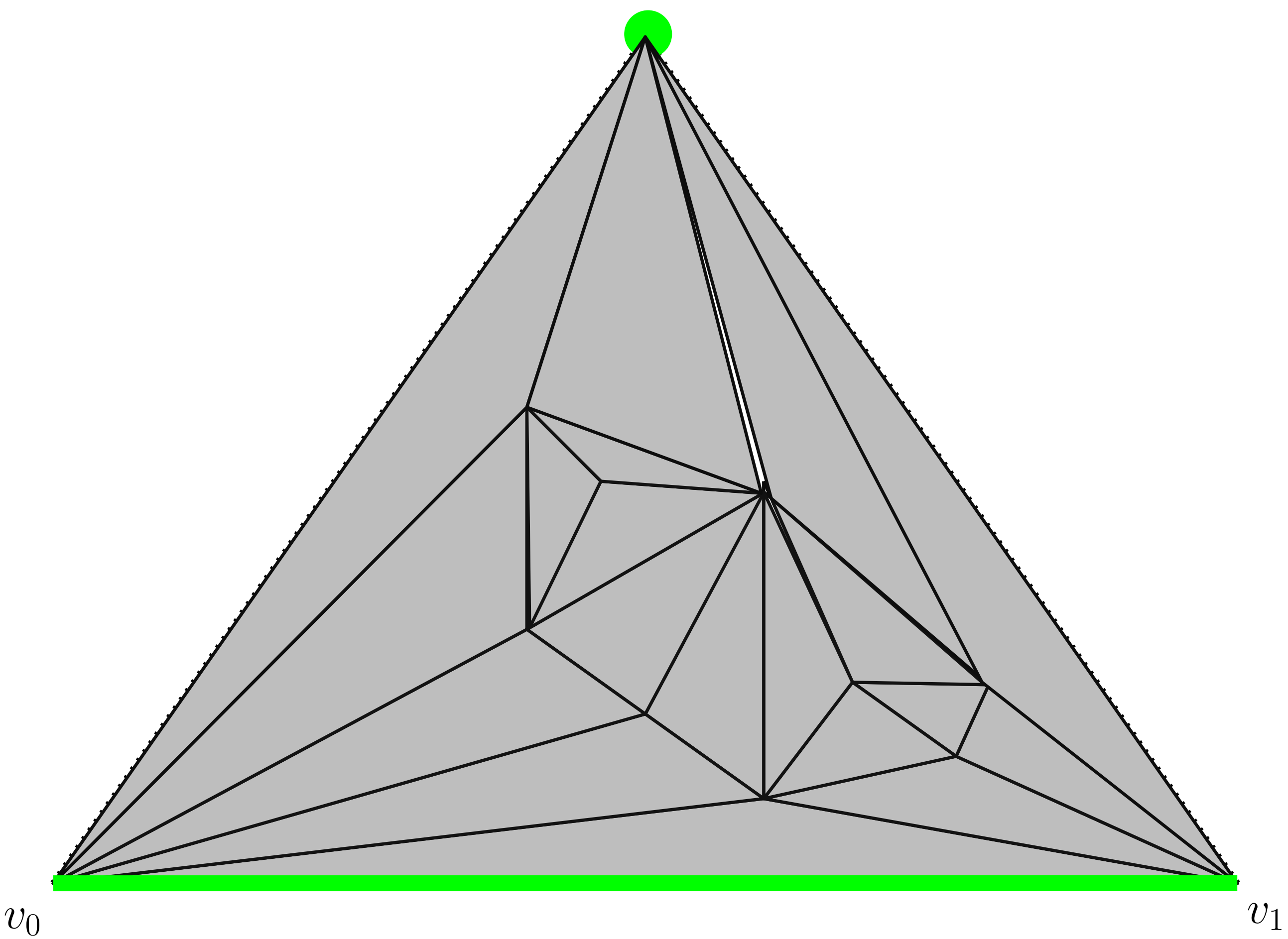
Schnyder woods: existence (algorithm I)

Via **Canonical orderings** (see Lecture 2)

[incremental vertex shelling, Brehm's thesis]

The traversal starts from the root face

Theorem
Every planar triangulation admits a Schnyder wood, which can be computed in linear time.



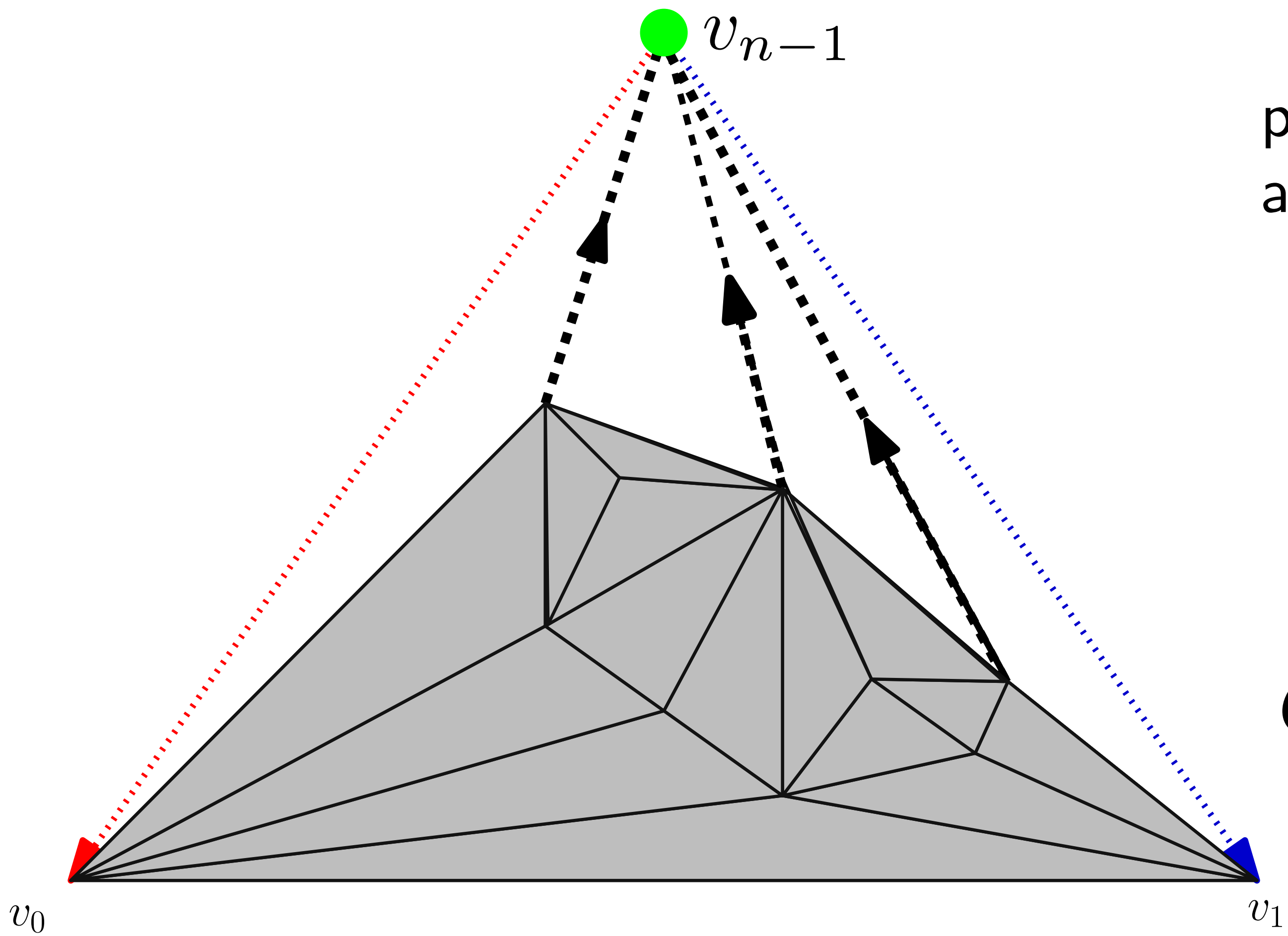
Schnyder woods: existence (algorithm I)

Via **Canonical orderings** (see Lecture 2)

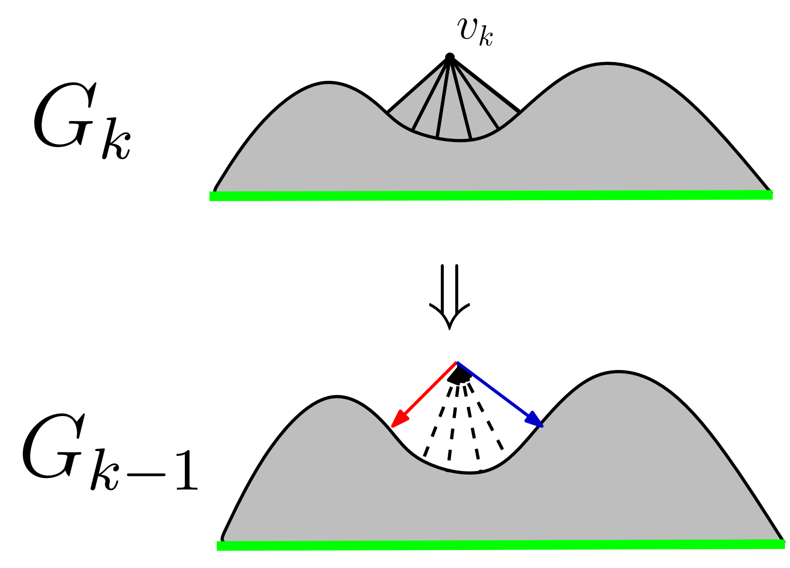
[incremental vertex shelling, Brehm's thesis]

The traversal starts from the root face

Theorem
Every planar triangulation admits a Schnyder wood, which can be computed in linear time.



perform a vertex conquest at each step



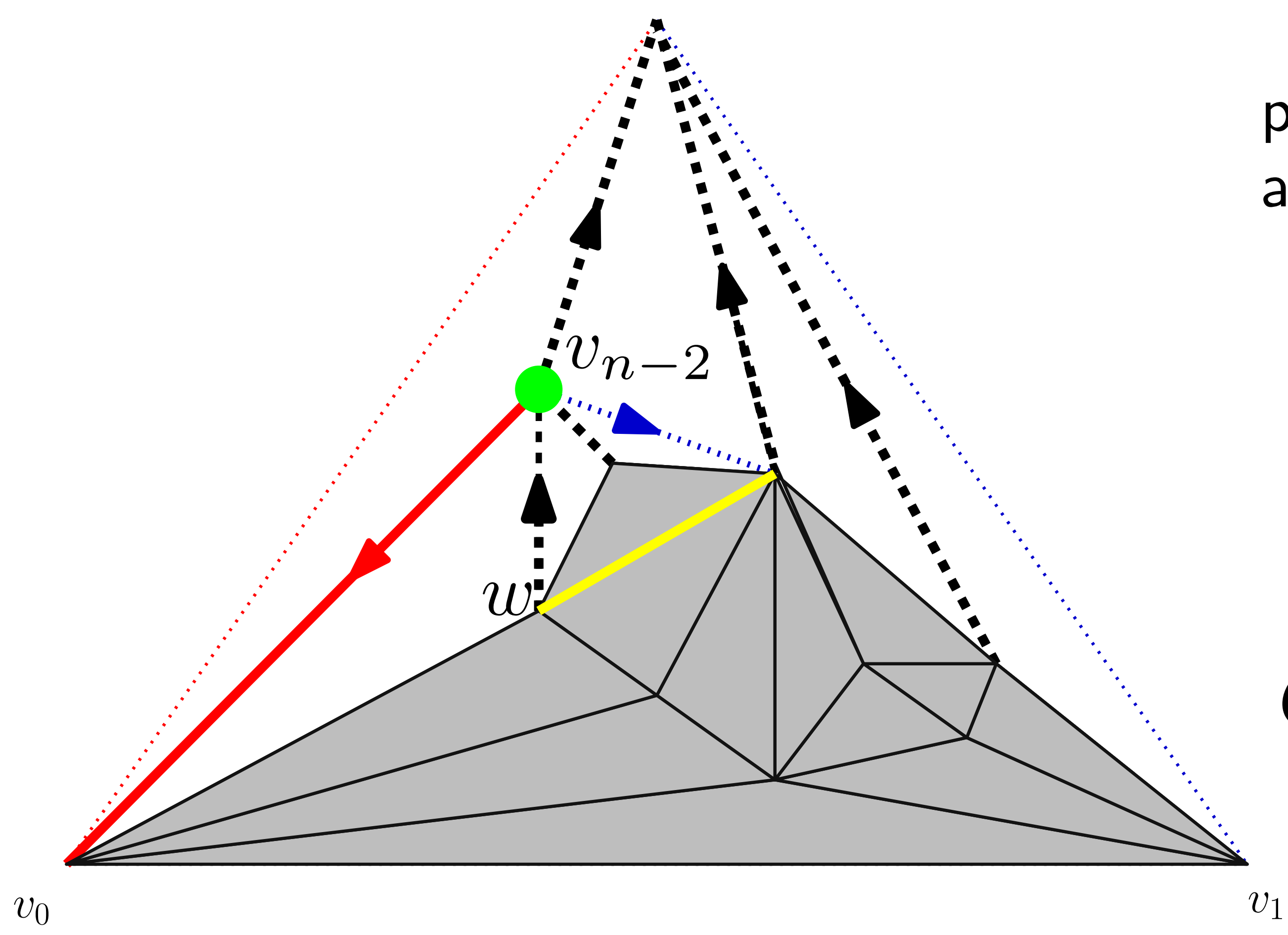
Schnyder woods: existence (algorithm I)

Via **Canonical orderings** (see Lecture 4)

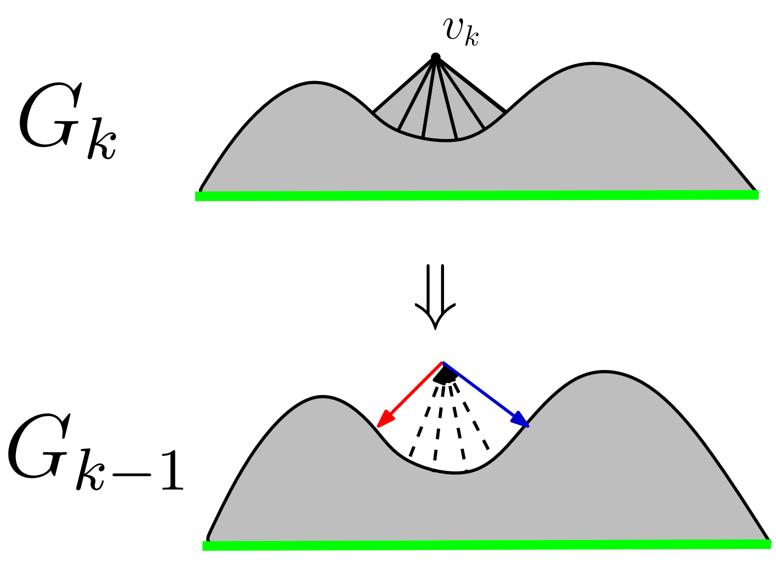
[incremental vertex shelling, Brehm's thesis]

The traversal starts from the root face

Theorem
Every planar triangulation admits a Schnyder wood, which can be computed in linear time.



perform a vertex conquest at each step

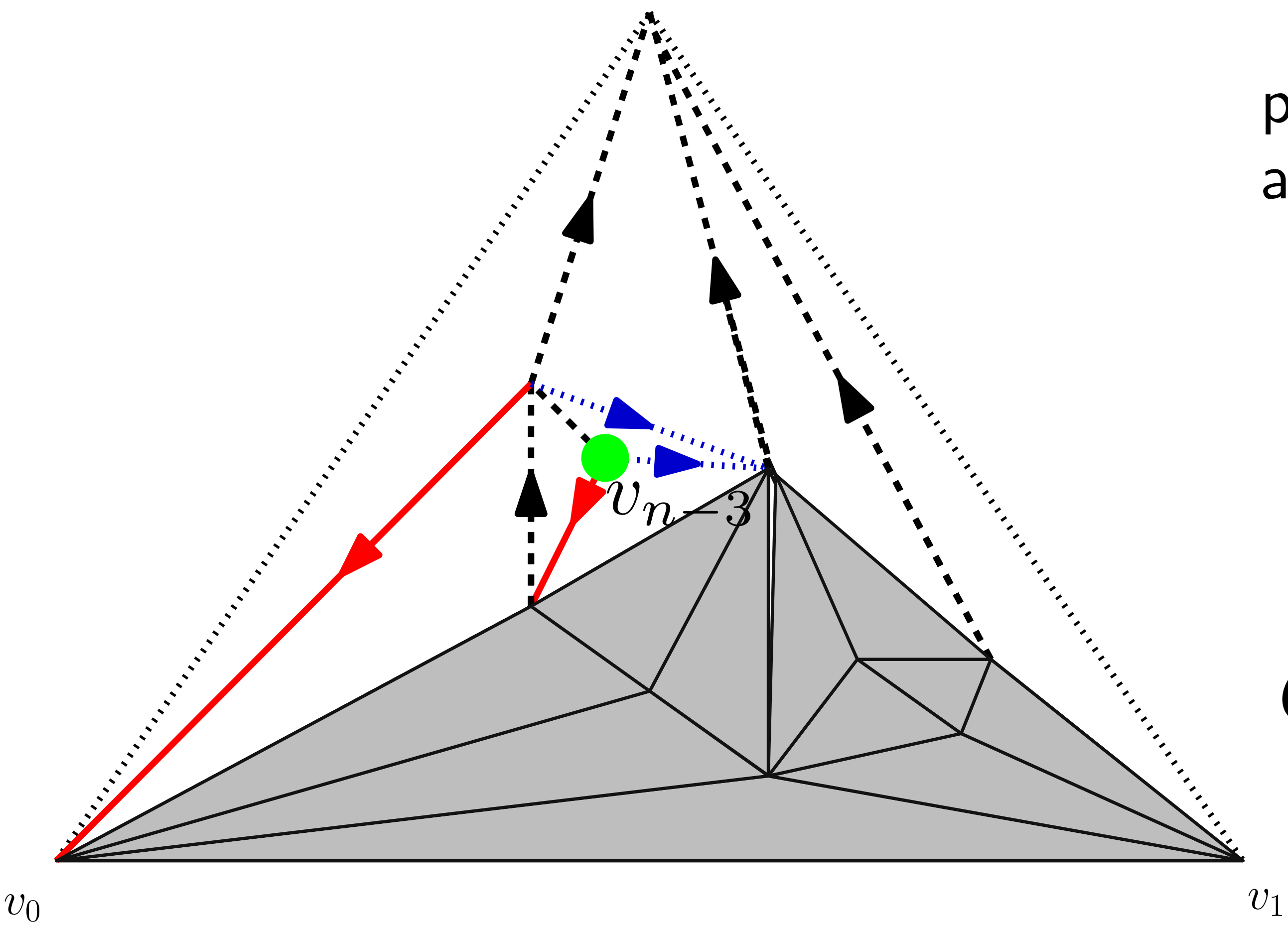


Schnyder woods: existence (algorithm I)

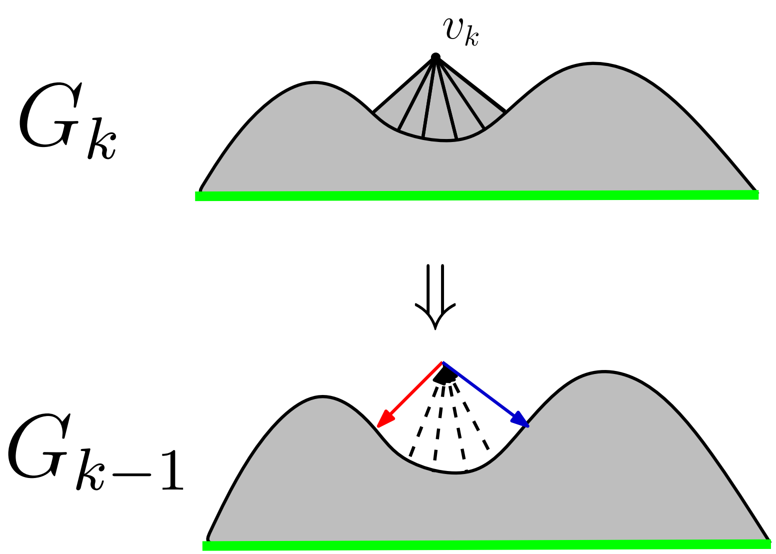
[incremental vertex shelling, Brehm's thesis]

The traversal starts from the root face

Theorem
Every planar triangulation admits a Schnyder wood, which can be computed in linear time.



perform a vertex conquest at each step

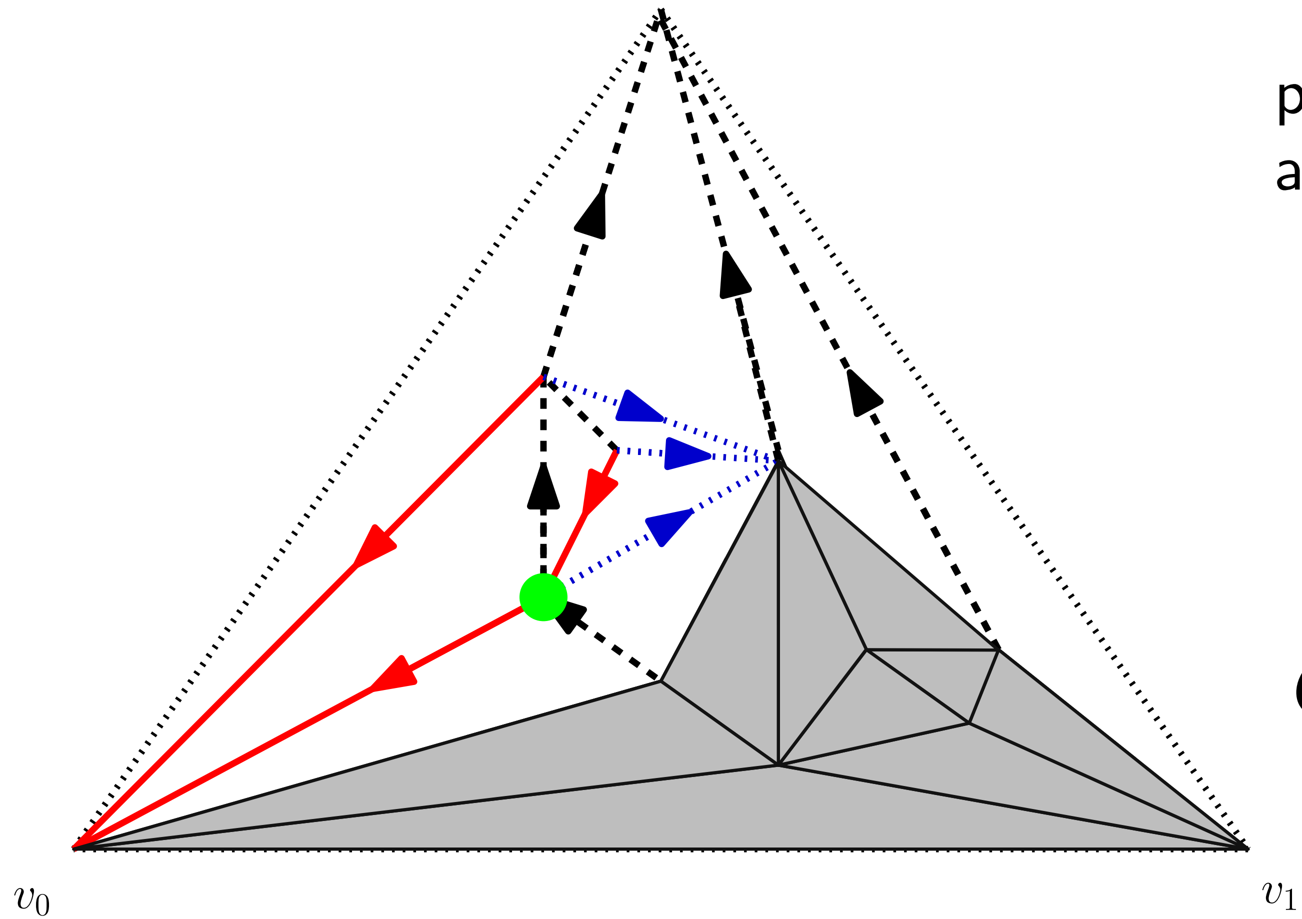


Schnyder woods: existence (algorithm I)

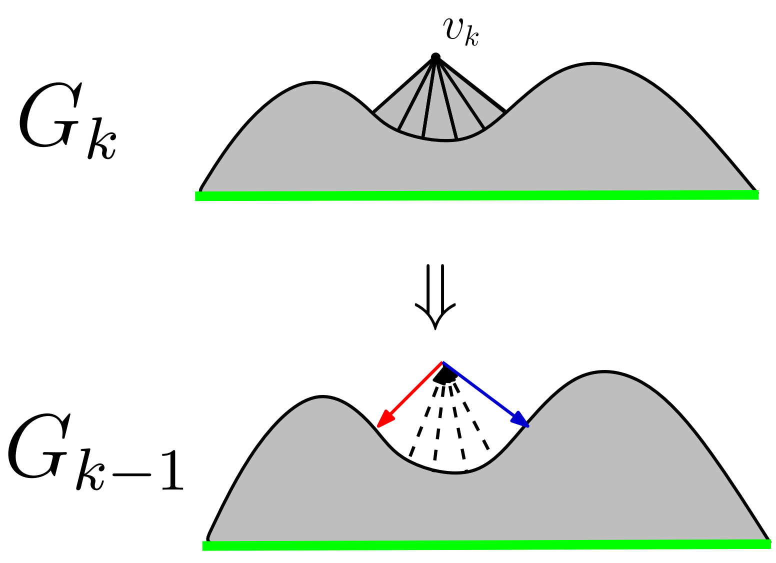
[incremental vertex shelling, Brehm's thesis]

The traversal starts from the root face

Theorem
Every planar triangulation admits a Schnyder wood, which can be computed in linear time.



perform a vertex conquest at each step

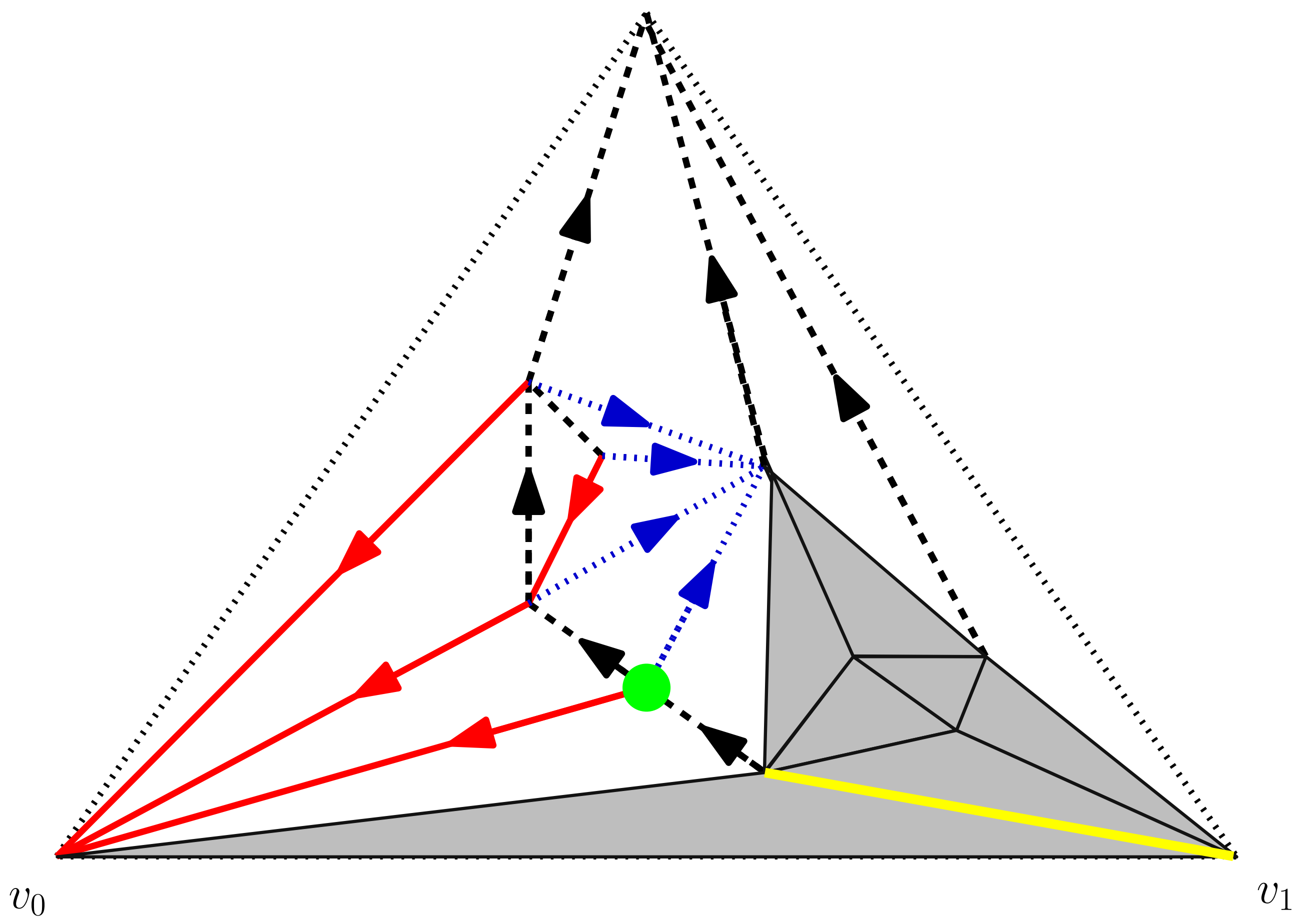


Schnyder woods: existence (algorithm I)

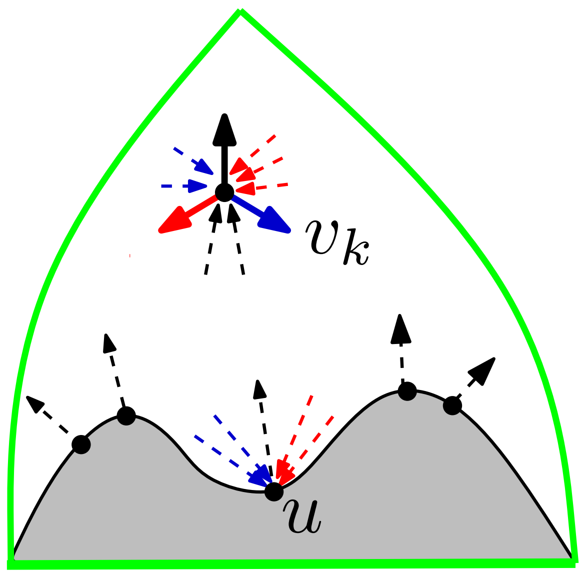
[incremental vertex shelling, Brehm's thesis]

The traversal starts from the root face

Theorem
Every planar triangulation admits a Schnyder wood, which can be computed in linear time.



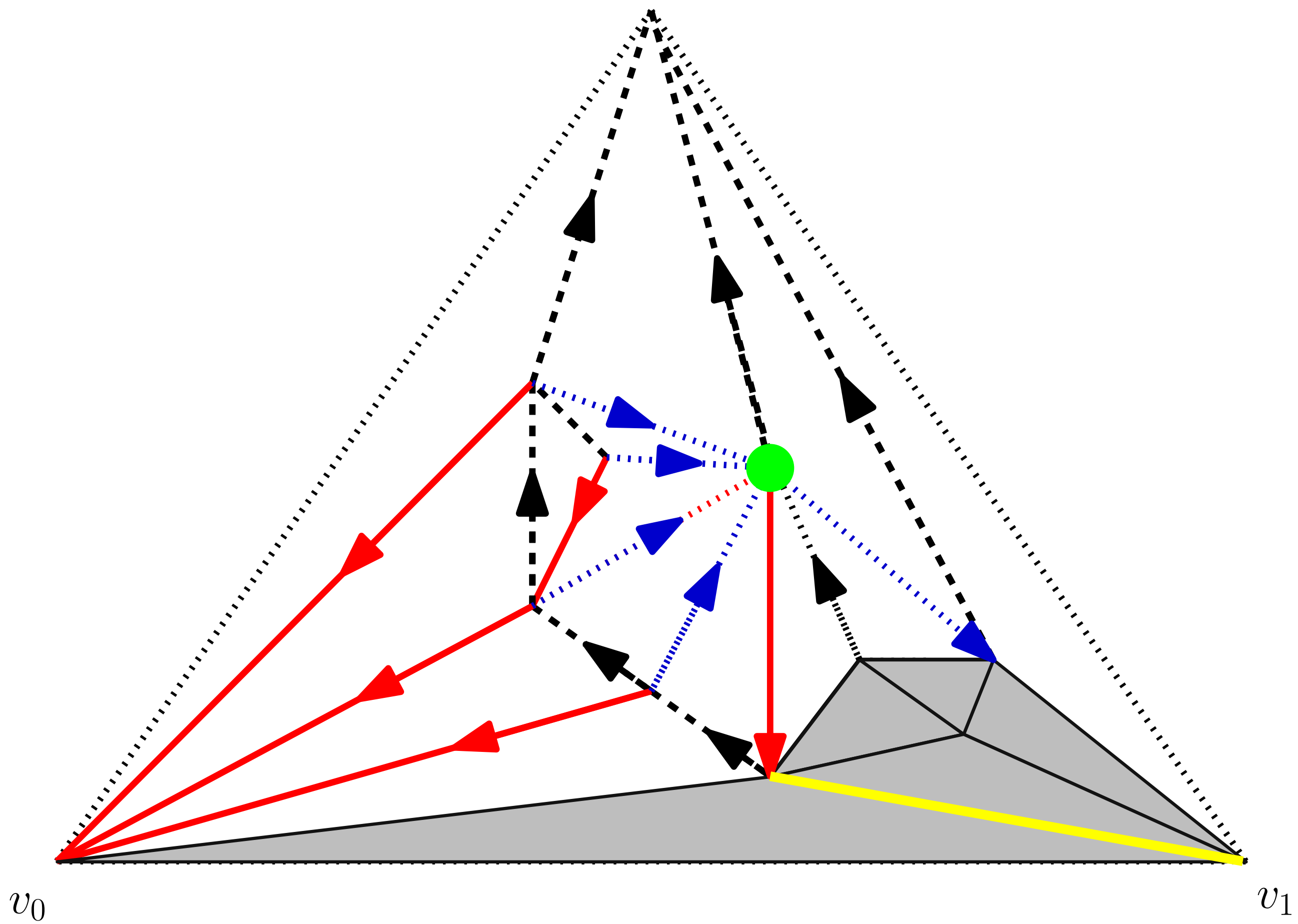
Invariant:



Schnyder woods: existence (algorithm I)

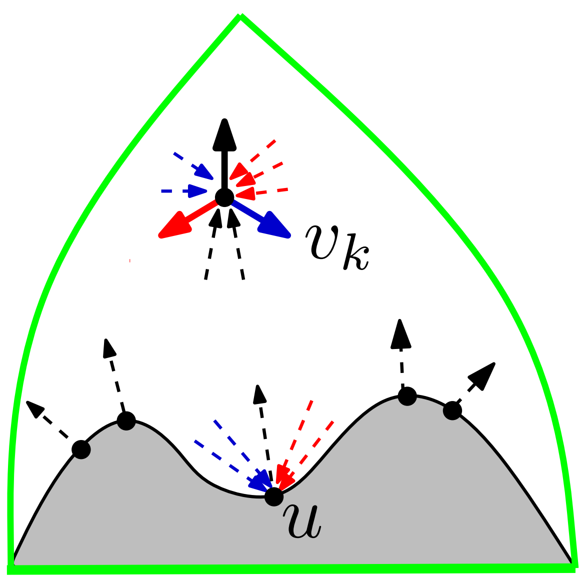
[incremental vertex shelling, Brehm's thesis]

The traversal starts from the root face



Theorem
Every planar triangulation admits a Schnyder wood, which can be computed in linear time.

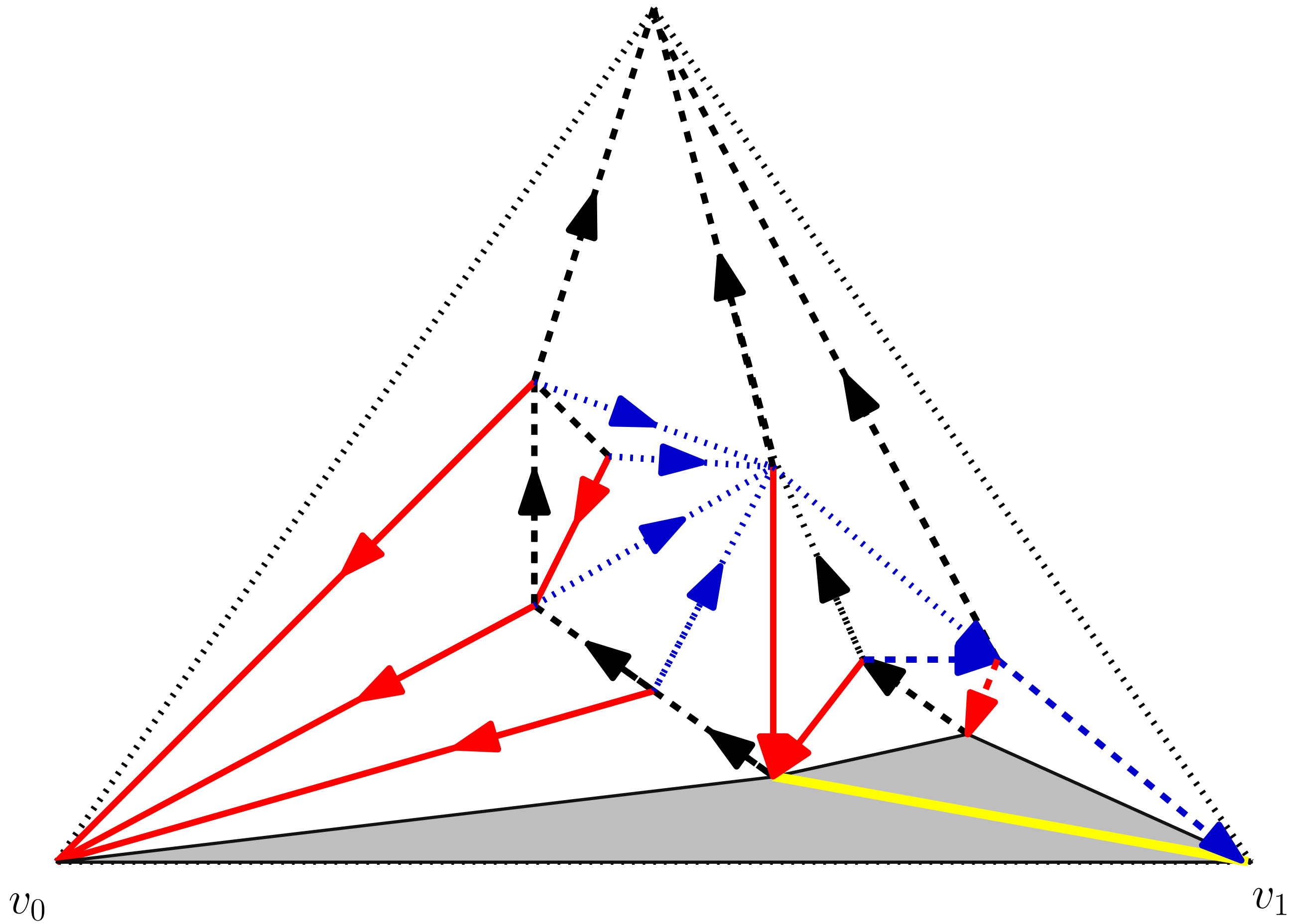
Invariant:



Schnyder woods: existence (algorithm I)

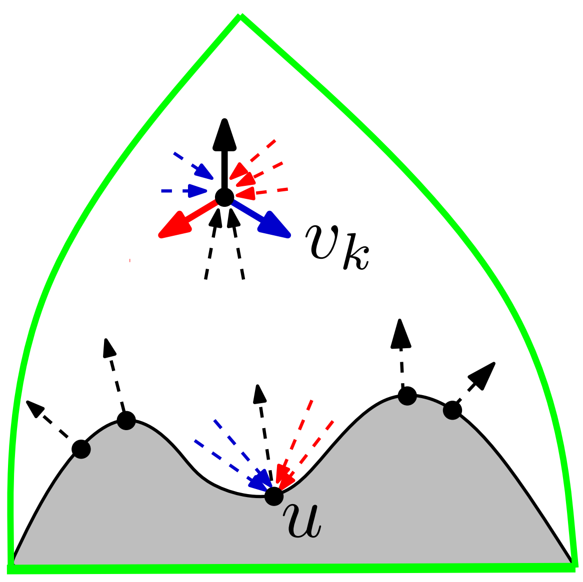
[incremental vertex shelling, Brehm's thesis]

The traversal starts from the root face



Theorem
Every planar triangulation admits a Schnyder wood, which can be computed in linear time.

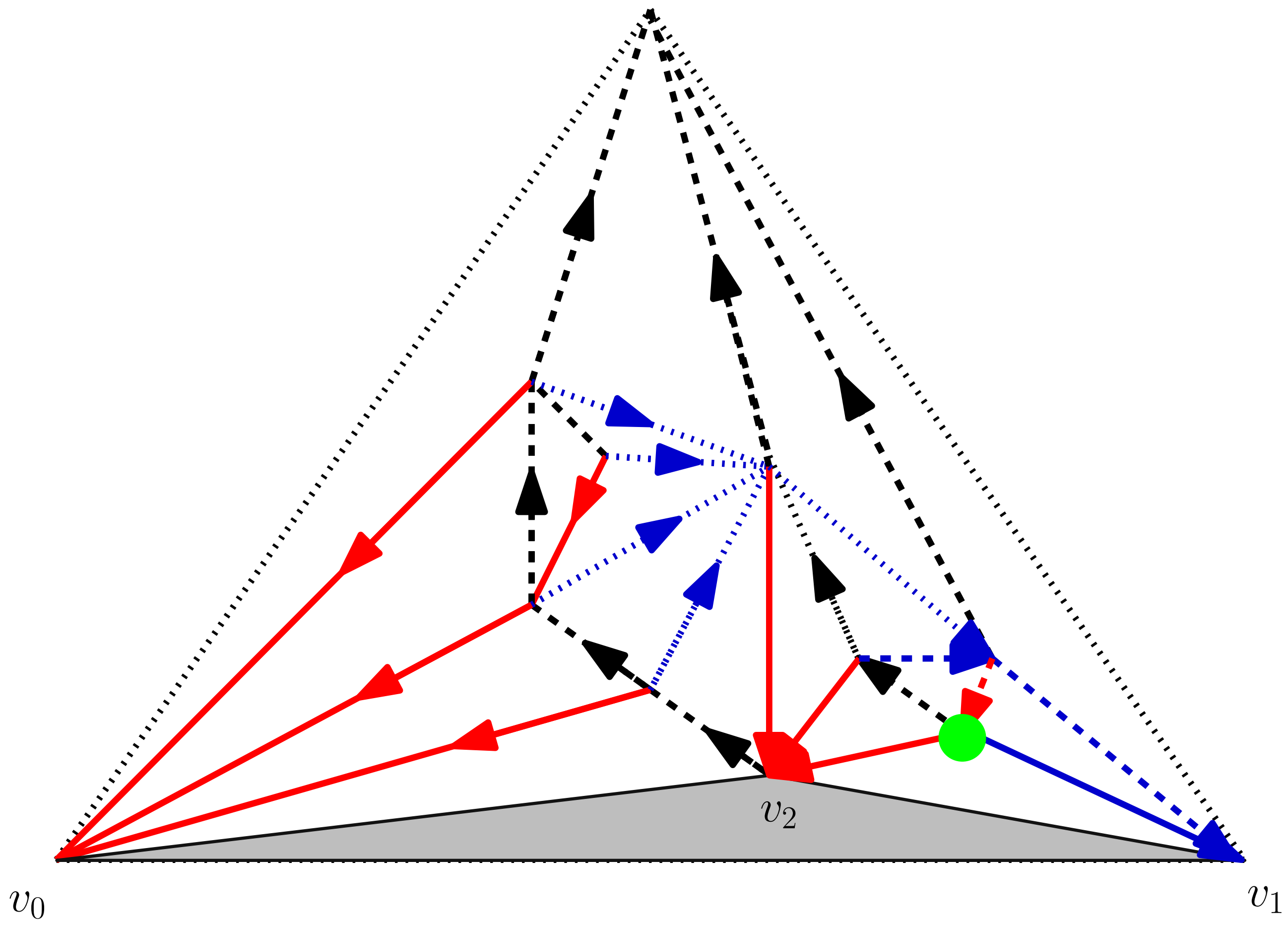
Invariant:



Schnyder woods: existence (algorithm I)

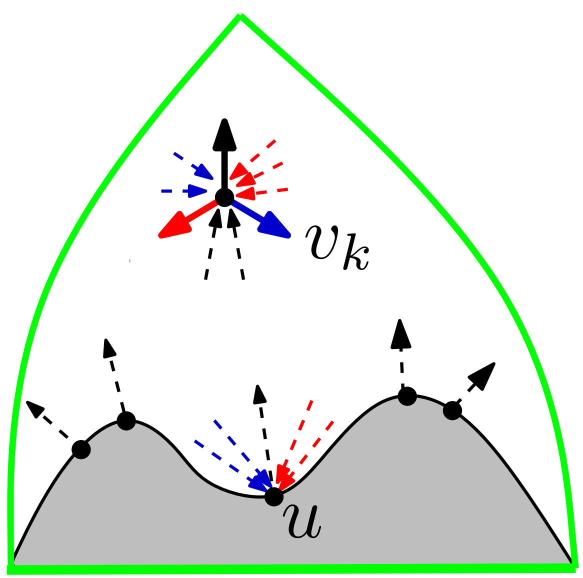
[incremental vertex shelling, Brehm's thesis]

The traversal starts from the root face



Theorem
Every planar triangulation admits a Schnyder wood, which can be computed in linear time.

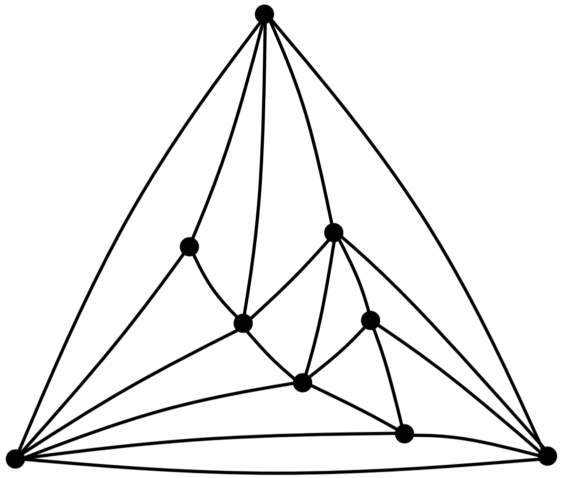
Invariant:



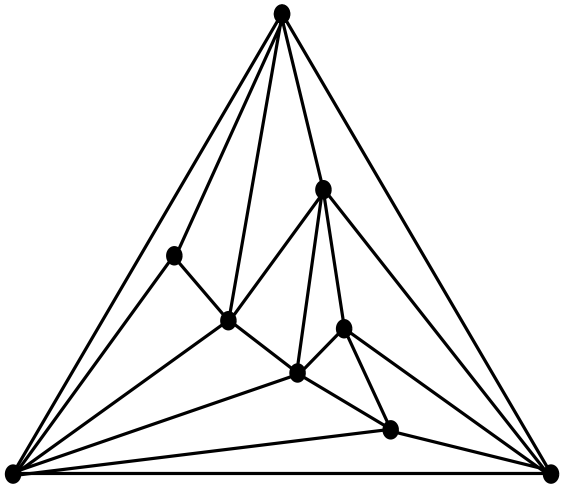
Planar straight-line drawings

(of planar graphs)

Planar straight-line drawings



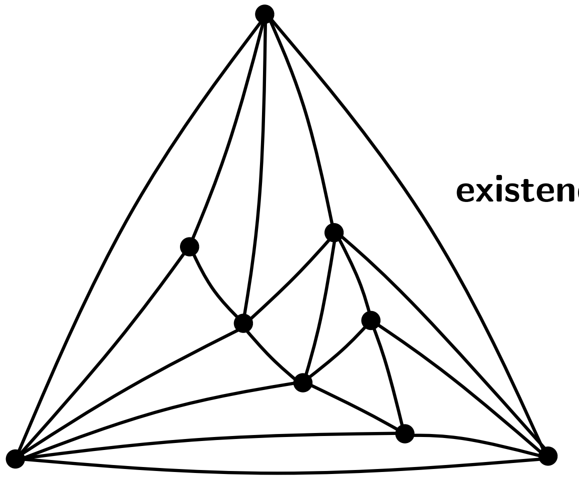
$\Longrightarrow$



[Wagner'36]

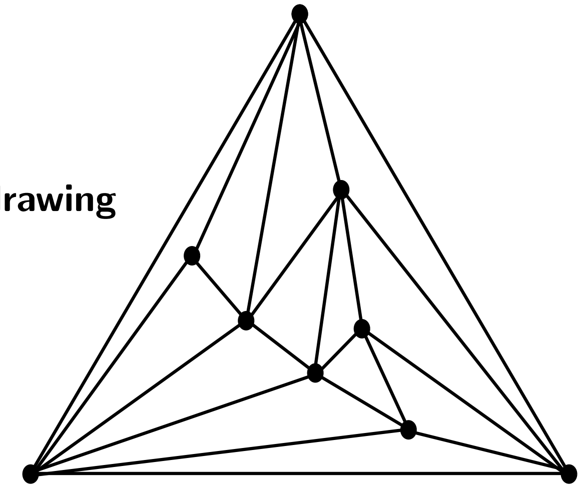
[Fary'48]

Planar straight-line drawings



existence of straight-line drawing

$\Rightarrow$

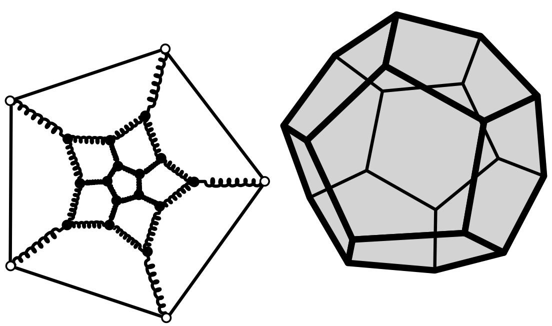


[Wagner'36]

[Fary'48]

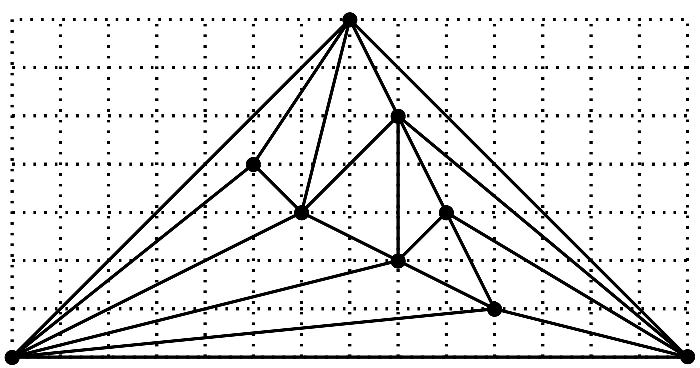
[Stein'51]

Classical algorithms:



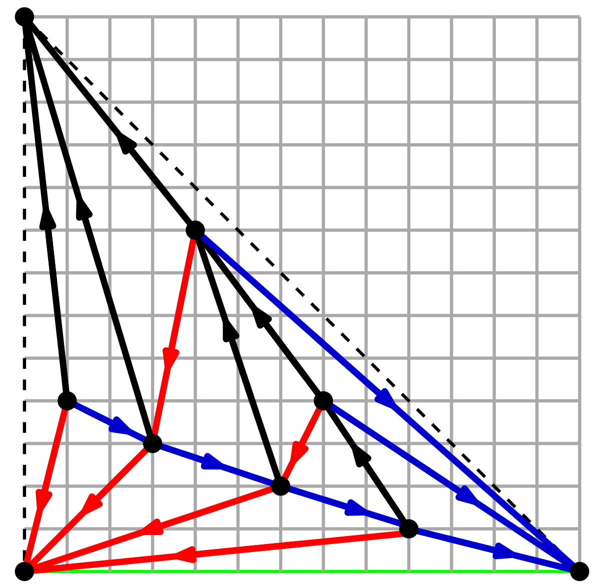
[Tutte'63]

spring-embedding



[De Fraysseix, Pach, Pollack 89]

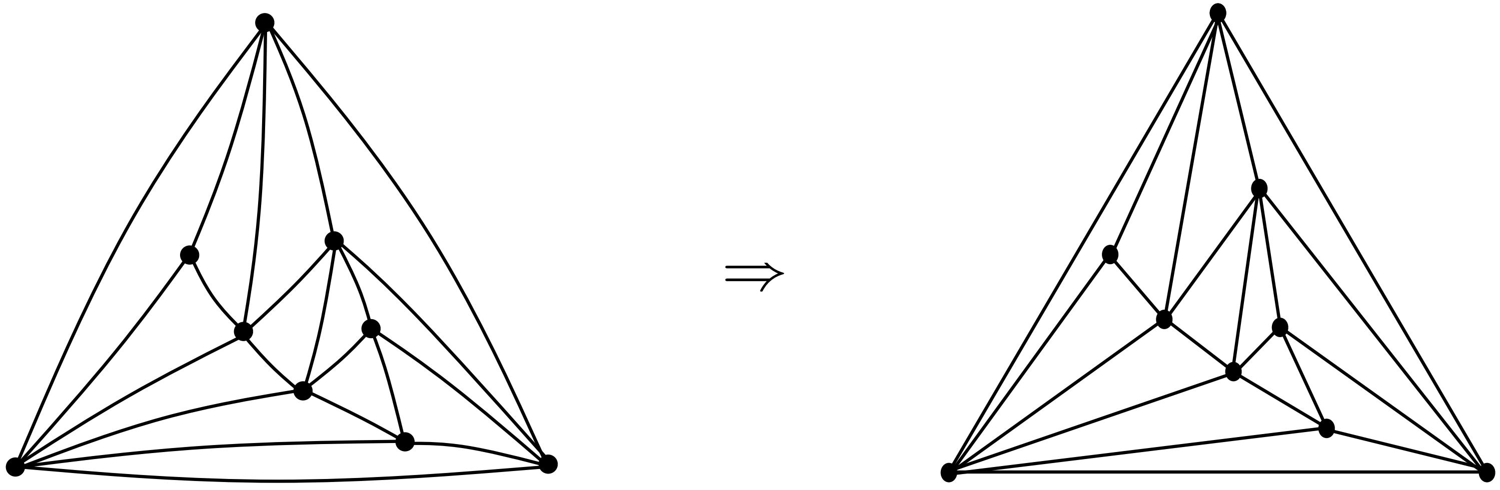
incremental (**Shift-algorithm**)



[Schnyder'90]

face-counting principle

Planar straight-line drawings



[Wagner'36]

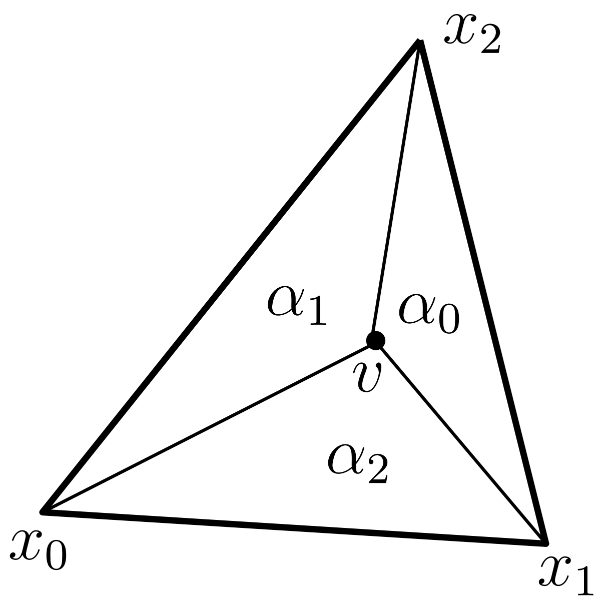
[Fary'48]

Face counting algorithm

(Schnyder algorithm, 1990)

Face counting algorithm

Geometric interpretation

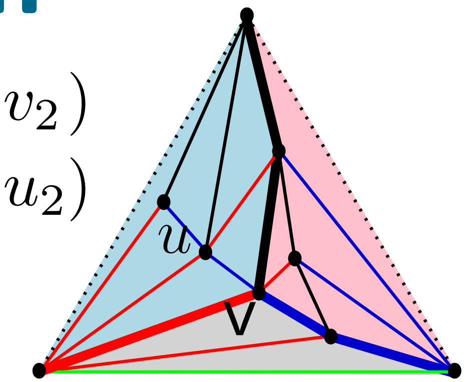


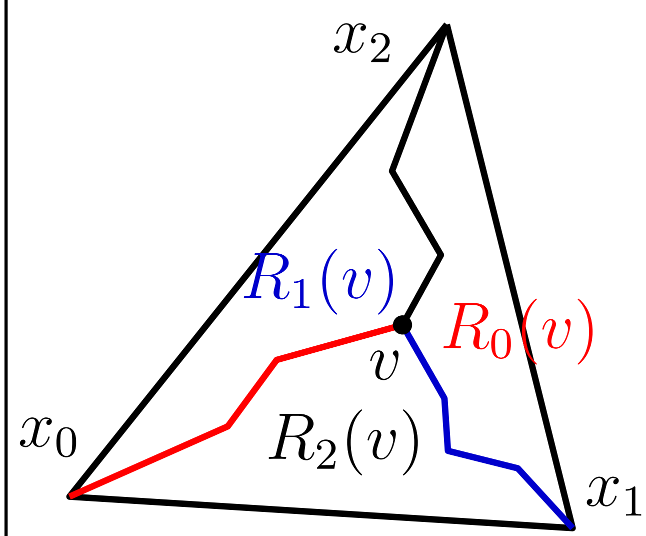
$v = \alpha_0 x_0 + \alpha_1 x_1 + \alpha_2 x_2$

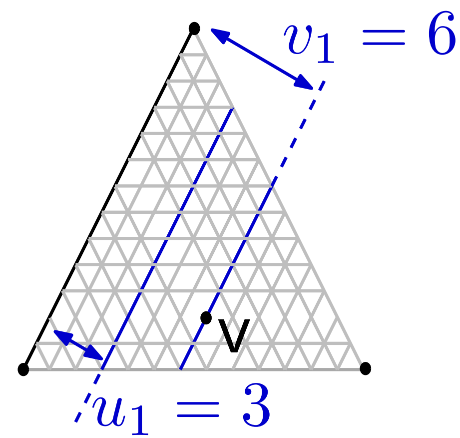
where α_i is the normalized area

$\mathbf{v} \rightarrow (5, 6, 2) := (v_0, v_1, v_2)$

$\mathbf{u} \rightarrow (7, 3, 3) := (u_0, u_1, u_2)$







$v = \frac{|R_0(v)|}{|F|-1} x_0 + \frac{|R_1(v)|}{|F|-1} x_1 + \frac{|R_2(v)|}{|F|-1} x_2$

$|R_i(v)|$ is the number of triangles in $R_i(v)$

Theorem

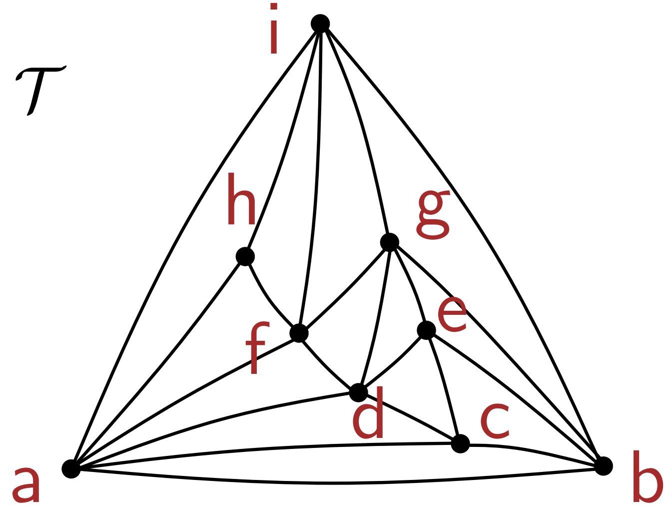
For a 3-connected planar map $\mathcal{M}$ having f vertices, there is drawing on a grid of size $(f-1) \times (f-1)$

Theorem (Schnyder, Soda '90)

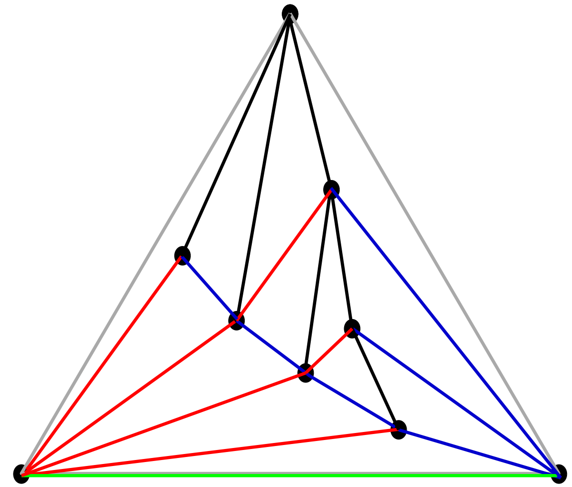
For a triangulation $\mathcal{T}$ having n vertices, we can draw it on a grid of size $(2n-5) \times (2n-5)$, by setting $x_0 = (2n-5, 0)$, $x_1 = (0,0)$ and $x_2 = (0, 2n-5)$.

Face counting algorithm: example

Input: $\mathcal{T}$



$\Rightarrow$



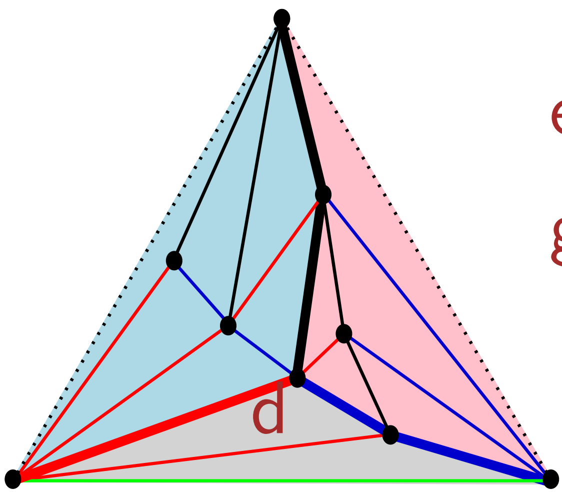
$\mathcal{T}$ endowed with a Schnyder wood

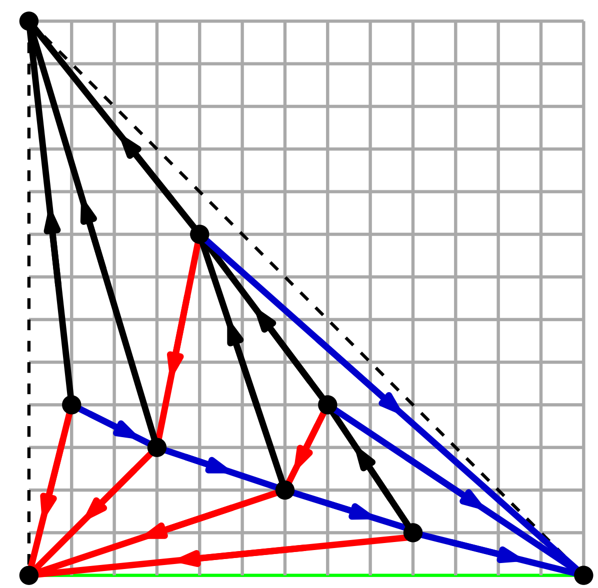
a $\rightarrow (0,0)$ b $\rightarrow (0,1)$ i $\rightarrow (1,0)$

c $\rightarrow (\frac{9}{13}, \frac{1}{13})$ **d** $\rightarrow (\boldsymbol{\frac{5}{13}}, \boldsymbol{\frac{6}{13}})$

e $\rightarrow (\frac{7}{13}, \frac{4}{13})$ f $\rightarrow (\frac{3}{13}, \frac{3}{13})$

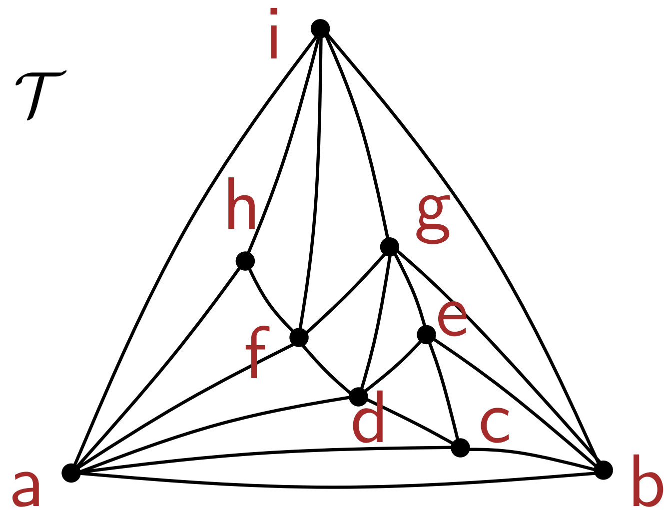
g $\rightarrow (\frac{4}{13}, \frac{8}{13})$ h $\rightarrow (\frac{1}{13}, \frac{4}{13})$



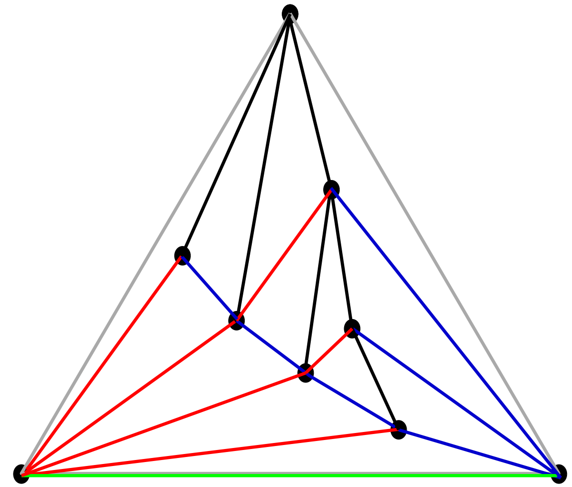


Face counting algorithm: example

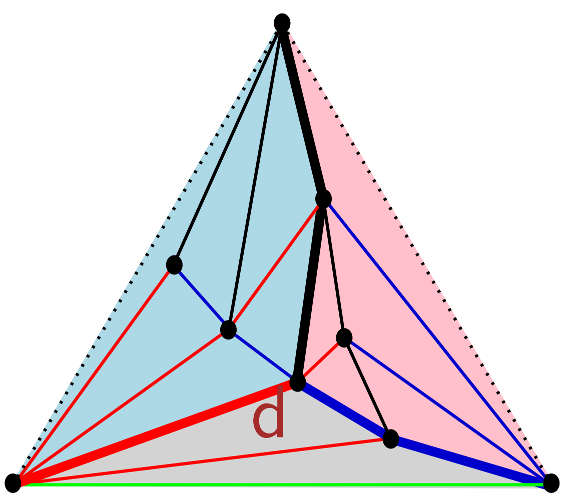
Input: $\mathcal{T}$



$\Rightarrow$



$\mathcal{T}$ endowed with a Schnyder wood



a $\rightarrow (13, 0, 0)$

b $\rightarrow (0, 13, 0)$

c $\rightarrow (9, 3, 1)$

d $\rightarrow (5, 6, 2)$

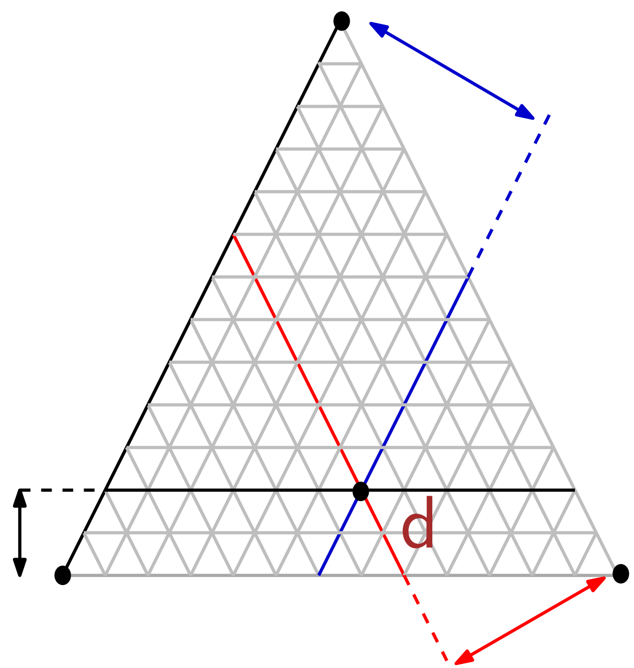
e $\rightarrow (2, 7, 4)$

f $\rightarrow (7, 3, 3)$

g $\rightarrow (1, 4, 8)$

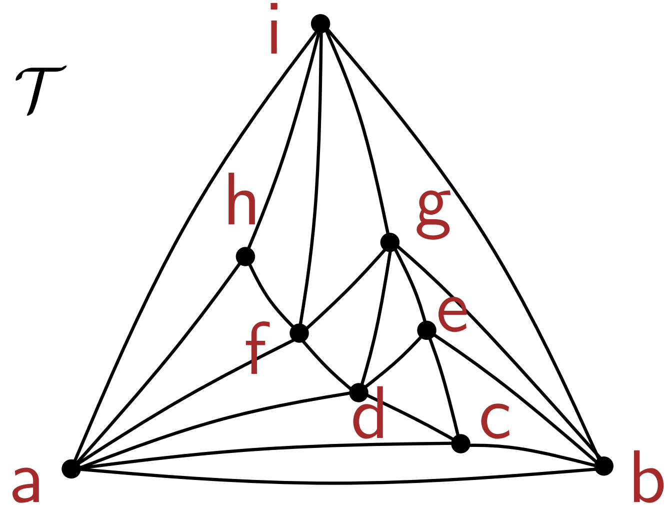
h $\rightarrow (8, 1, 4)$

i $\rightarrow (0, 0, 13)$

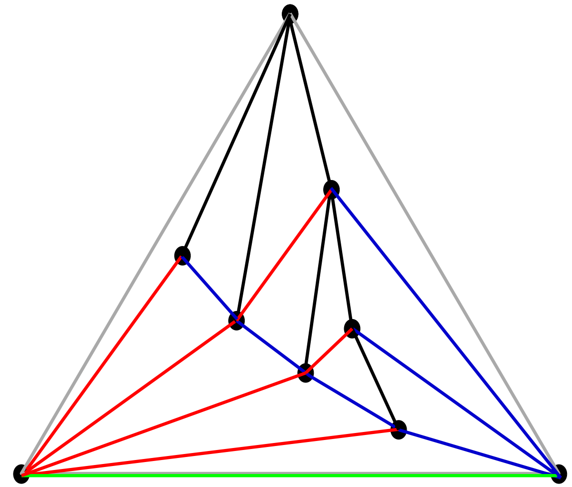


Face counting algorithm: example

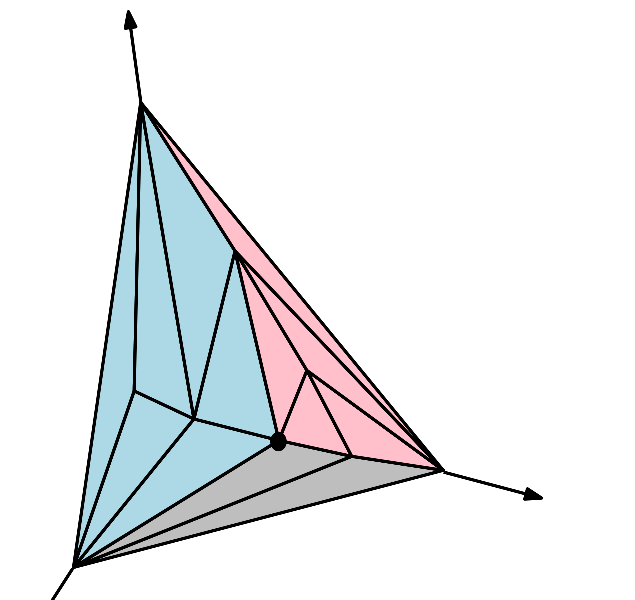
Input: $\mathcal{T}$



$\Rightarrow$



$\mathcal{T}$ endowed with a Schnyder wood



$x + y + z = 2n - 5$

a $\rightarrow (13, 0, 0)$

b $\rightarrow (0, 13, 0)$

c $\rightarrow (9, 3, 1)$

d $\rightarrow (5, 6, 2)$

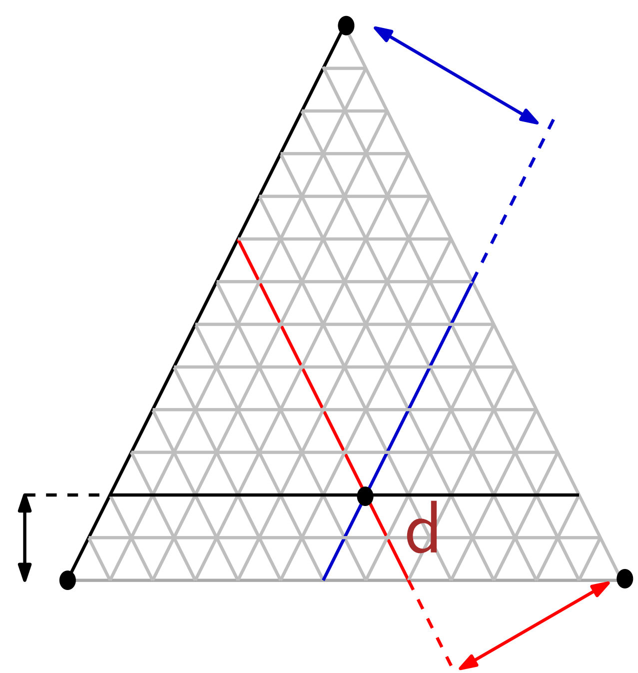
e $\rightarrow (2, 7, 4)$

f $\rightarrow (7, 3, 3)$

g $\rightarrow (1, 4, 8)$

h $\rightarrow (8, 1, 4)$

i $\rightarrow (0, 0, 13)$



Barycentric representation of a planar graph

(validity of the Schnyder layout)

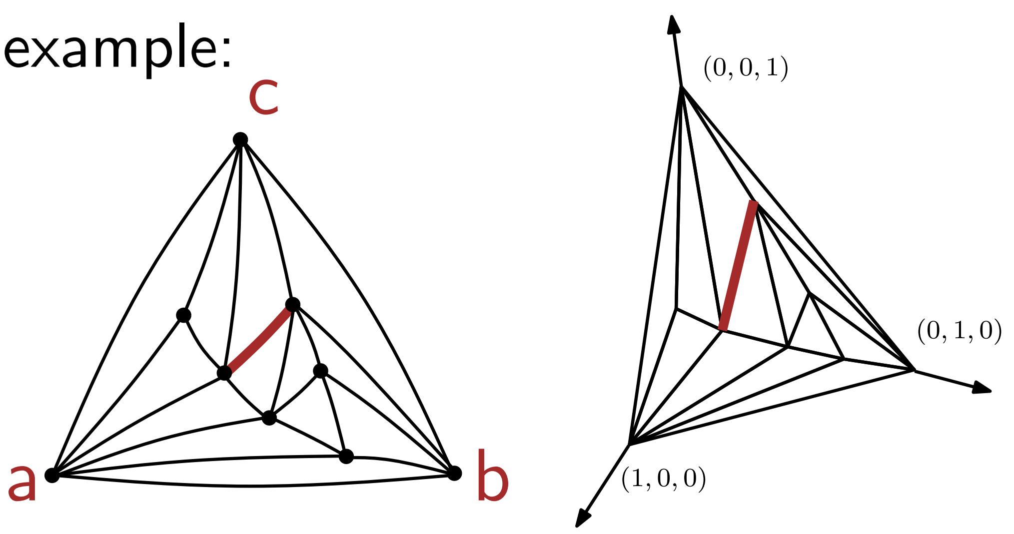
Barycentric representation of a planar graph

Definition: A **barycentric representation** of a graph G is defined by a mapping $f(v) \longrightarrow (v_0, v_1, v_2) \in R^3$ satisfying:

- $v_0 + v_1 + v_2 = 1$, for each vertex v
- for each edge $(x, y) \in E$ and each vertex $z \notin \{x, y\}$ there is an index $k \in \{0, 1, 2\}$ such that

$$x_k < z_k$$
$$y_k < z_k$$

example:



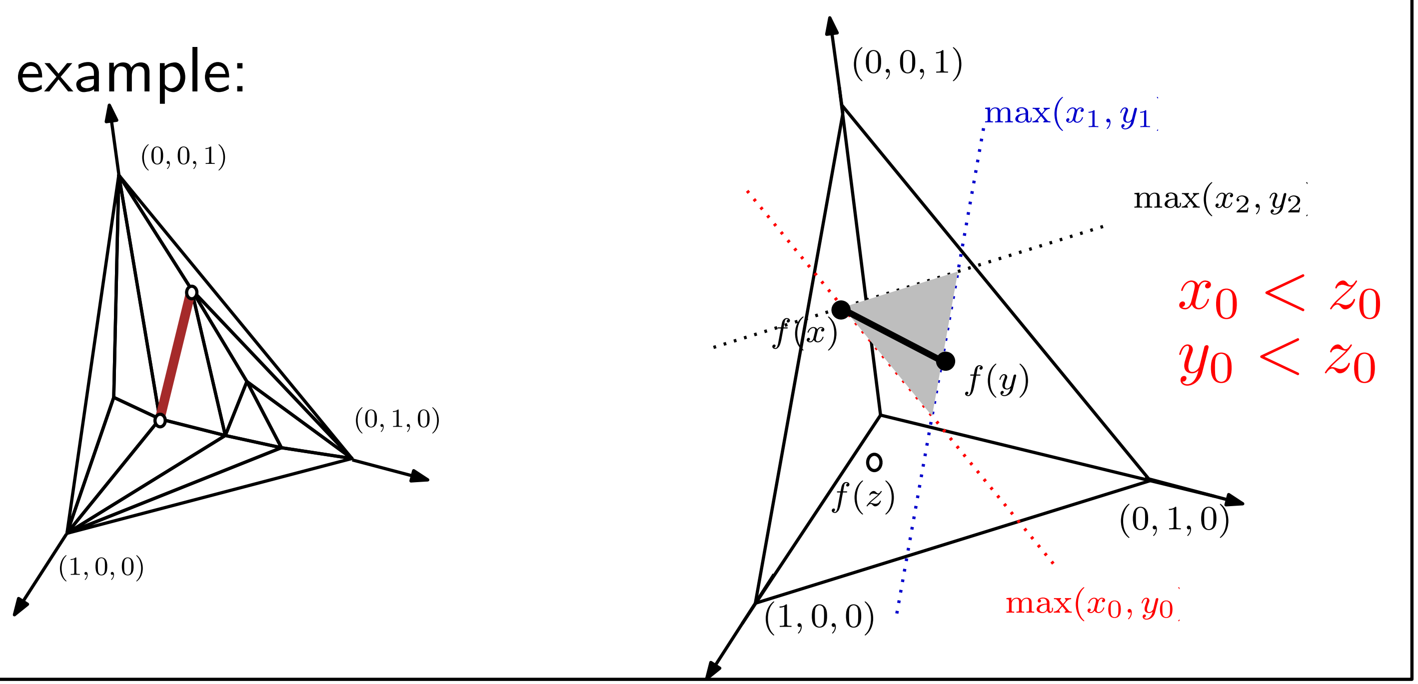
Barycentric representation of a planar graph

Definition: A **barycentric representation** of a graph G is defined by a mapping $f(v) \longrightarrow (v_0, v_1, v_2) \in R^3$ satisfying:

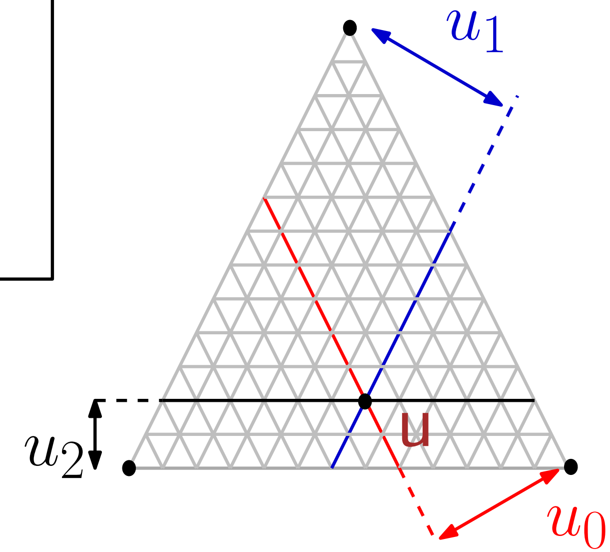
- $v_0 + v_1 + v_2 = 1$, for each vertex v
- for each edge $(x, y) \in E$ and each vertex $z \notin \{x, y\}$ there is an index $k \in \{0, 1, 2\}$ such that

$$x_k < z_k$$
$$y_k < z_k$$

example:



Intuition: no vertex z in the gray triangle defined by $f(x), f(y)$



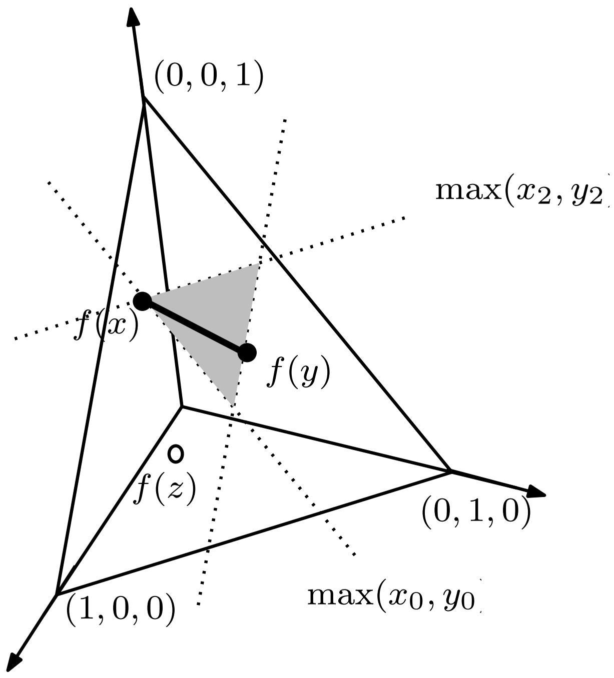
Barycentric representation of a planar graph

Theorem
A barycentric representation defines a planar straight-line (crossing-free) drawing of G, in the plane spanned by $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$.

Claim 1: for each edge $(x,y) \in E$ and each vertex $z \notin \{x,y\}$, $f(z)$ cannot lie on $(f(x), f(y))$

proof: **by contradiction:** assume $f(z) \in (f(x), f(y))$, so we can write

$$f(z) = tf(x) + (1-t)f(y) \text{ , for some } t \in [0,1]$$



f is a barycentric representation, so there is $k \in \{0,1,2\}$ s. t.

$$x_k < z_k$$
$$y_k < z_k$$

so get a contradiction

$$z_k = tx_k + (1-t)y_k < tz_k + (1-t)z_k = z_k$$

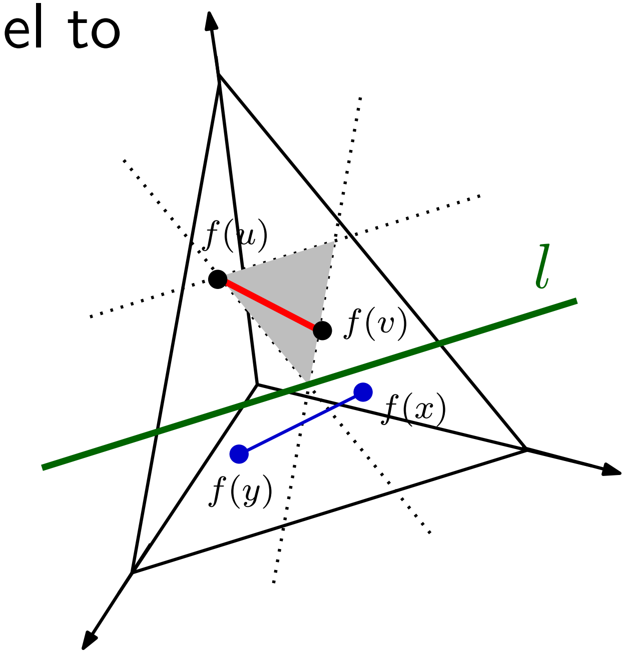
Barycentric representation of a planar graph

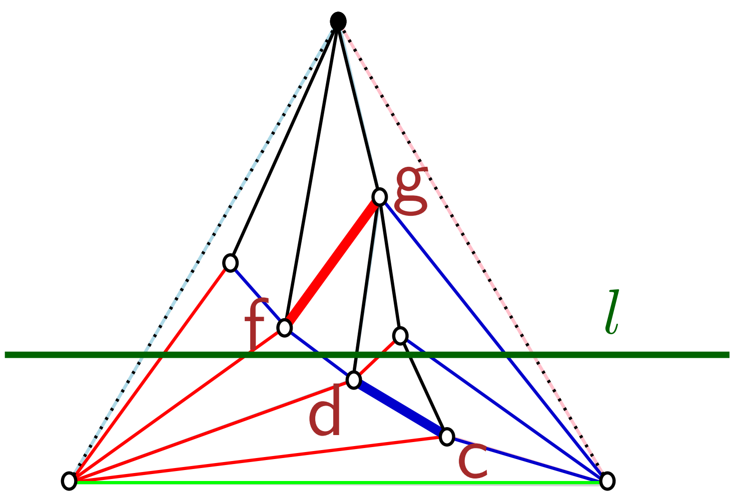
Theorem

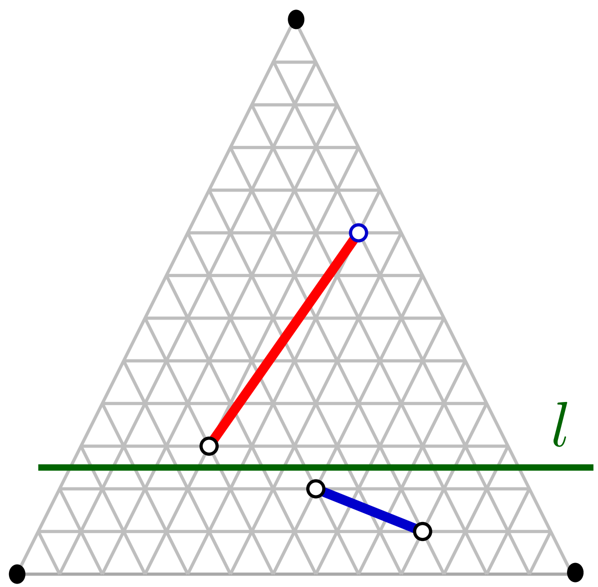
A barycentric representation defines a planar straight-line (crossing-free) drawing of G, in the plane spanned by $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$.

Claim 2: given two edges (x,y), (u,v) of G they cannot cross

proof (intuition): we can find a straight-line l (parallel to one of the 3 axis) separating the two edges







c $\rightarrow (3,9,1)$ f $\rightarrow (7,3,3)$

d $\rightarrow (5,6,2)$ g $\rightarrow (1,4,8)$

$c_2, d_2 < f_2$

$d_2, c_2 < g_2$

Barycentric representation of a planar graph

Theorem

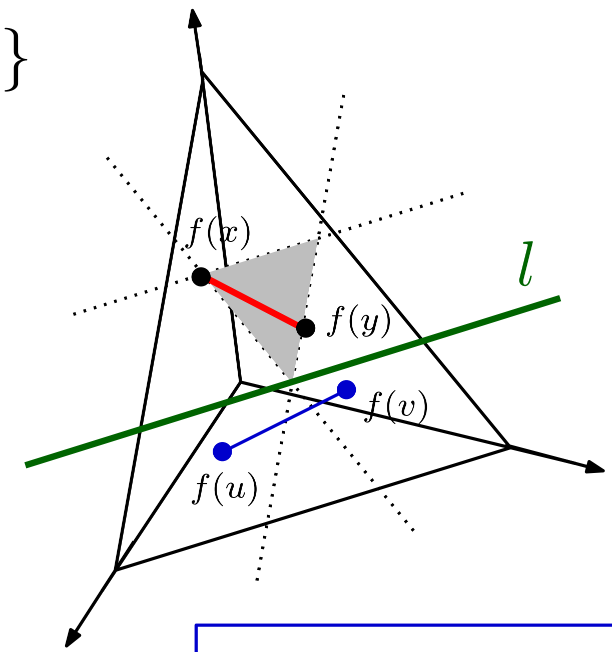
A barycentric representation defines a planar straight-line drawing of G, in the plane spanned by $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$.

Claim 2: given two edges (x,y), (u,v) of G they cannot cross

proof:

by definition there are four indices $i,j,k,l \in \{0,1,2\}$

$$u_i, v_i < x_i \qquad x_k, y_k < u_k$$
$$u_j, v_j < y_j \qquad x_l, y_l < v_l$$



Fact: $i \neq k$

if $i = k$ we would have

$u_k < x_k$

$v_k < x_k$

contradicting

$x_k, y_k < u_k$

$i \neq k, l$ and $j \neq k, l$ $\longrightarrow$ $i = j$ or $k = l$

there exists a separating line l parallel to one of the sides of the outer triangle, that separates (u,v) and (x,y)

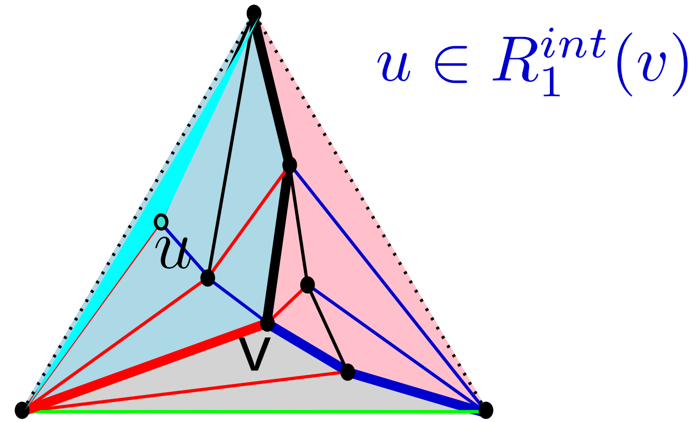
In the example above we have $i = j = 2$ $\longrightarrow$ the line l parallel to $[(1,0,0),(0,1,0)]$ separates (u,v) and (x,y)

The Schnyder layout defines a barycentric representation

(validity of the Schnyder layout)

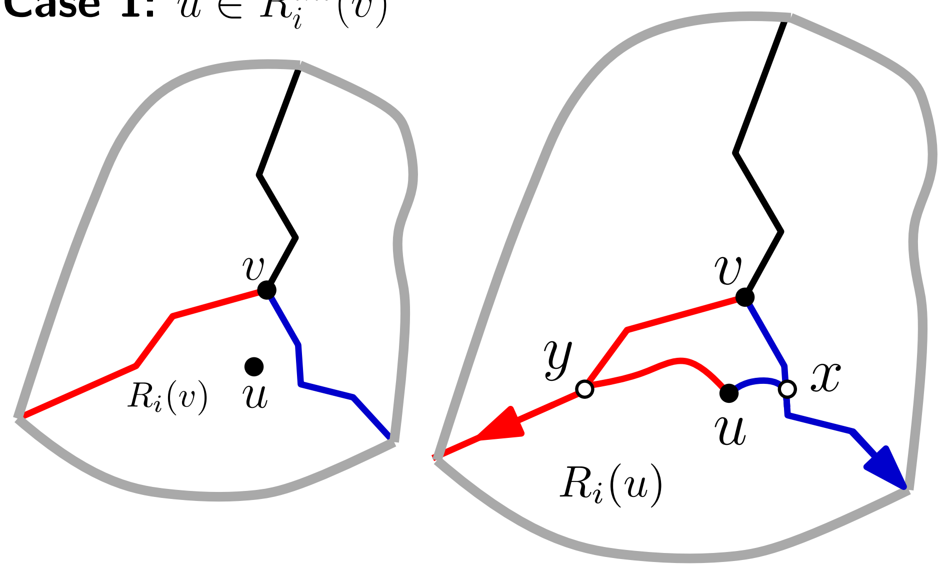
Paths and regions

Lemma Let (T_0, T_1, T_2) a Schnyder wood of $\mathcal{M}$.
If $u \in R_i(v)$ then $R_i(u) \subseteq R_i(v)$
If $u \in R_i^{int}(v)$ then $R_i(u) \subset R_i(v)$



proof:

Case 1: $u \in R_i^{int}(v)$

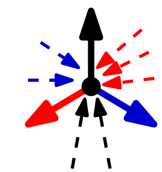


first step: compute the paths $P_{i+1}(u)$ and $P_{i-1}(u)$

They must intersect the boundary of $R_i(v)$ at x and y

Remark: x and y are different from v

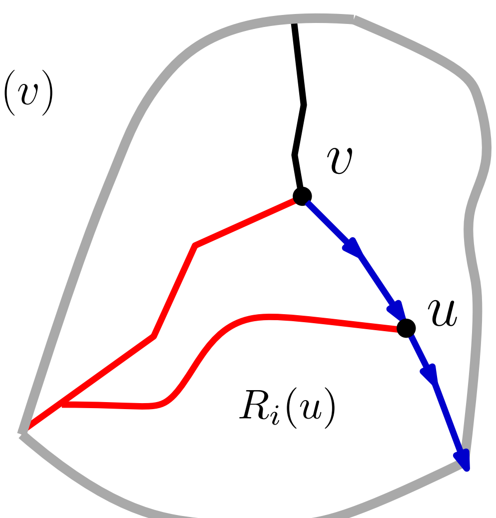
and we have $y \in P_{i+1}(u)$ and $x \in P_{i-1}(u)$
(because of Schnyder rule)



so we have: $R_i(u) \subset R_i(v)$

Case 2a: $u \in P_{i-1}(v)$

$R_i(u) \subset R_i(v)$

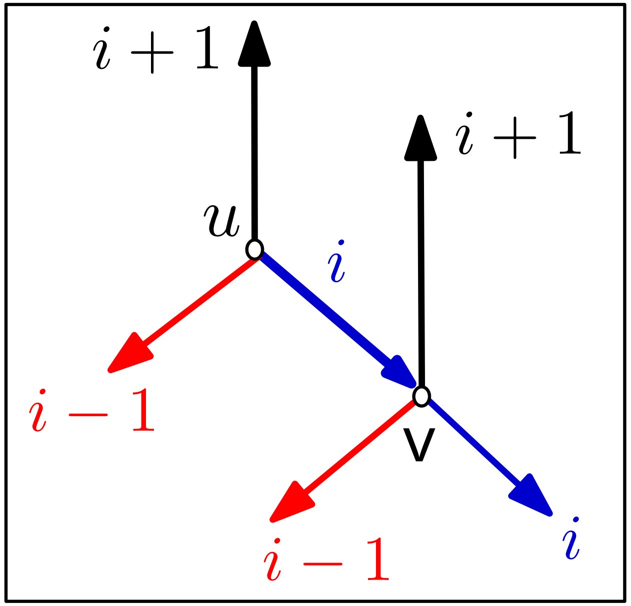


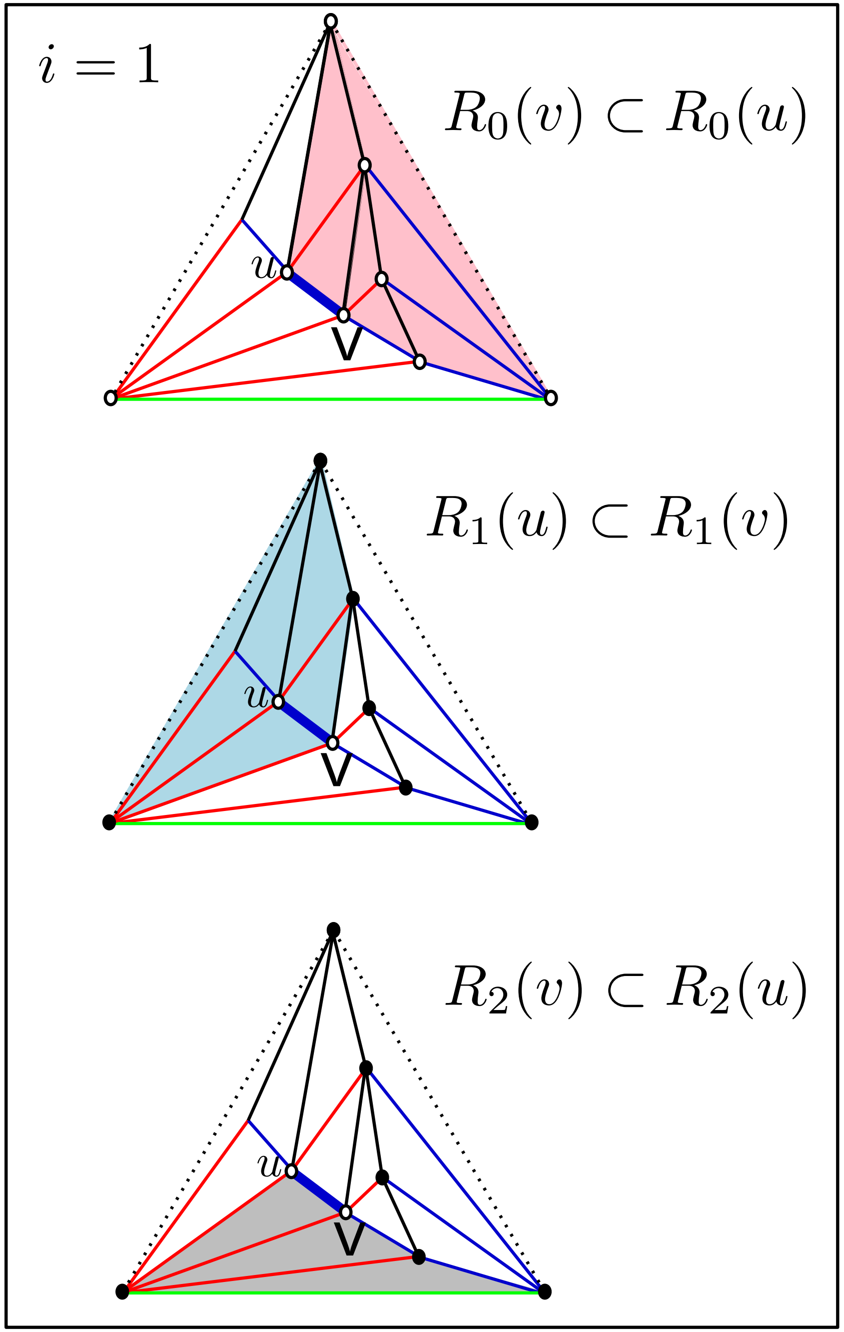
Paths and regions

Remarks: Let (u, v) be an edge of color i oriented from u to v

$$v \in P_i(u) \longrightarrow \begin{cases} v \in R_{i+1}(u) \\ v \in R_{i-1}(u) \\ u \in R_i(v) \end{cases}$$

$$R_i(u) \subset R_i(v)$$
$$R_{i+1}(v) \subset R_{i+1}(u)$$
$$R_{i-1}(v) \subset R_{i-1}(u)$$

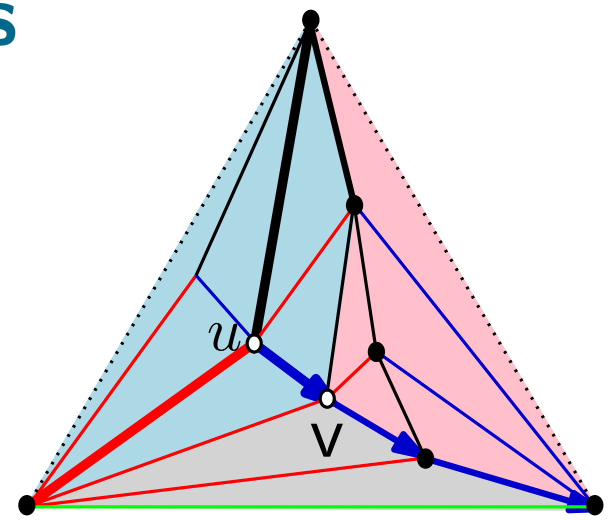




Regions and coordinates

Schnyder coordinates

$$v =: \frac{|R_0(v)|}{|F|-1}x_0 + \frac{|R_1(v)|}{|F|-1}x_1 + \frac{|R_2(v)|}{|F|-1}x_2 =$$
$$= \frac{v_0}{|F|-1}x_0 + \frac{v_1}{|F|-1}x_1 + \frac{v_2}{|F|-1}x_2$$



v $(5, 6, 2) := (v_0, v_1, v_2)$

u $(7, 3, 3) := (u_0, u_1, u_2)$

Given (u, v) of color i oriented from u to v we have:

- $R_i(u) \subseteq R_i(v) \longrightarrow |R_i(u)| \leq |R_i(v)| \longrightarrow \boxed{u_i \leq v_i}$

- $R_i(u) \subset R_i(v)$, $R_{i+1}(v) \subset R_{i+1}(u)$, $R_{i-1}(v) \subset R_{i-1}(u)$ $\longrightarrow \begin{cases} u_i < v_i \\ u_{i+1} > v_{i+1} \\ u_{i-1} > v_{i-1} \end{cases}$

- $v_0 + v_1 + v_2 = f - 1$
- For every edge (u, v) there are some indices $i, j \in \{0, 1, 2\}$ s.t.

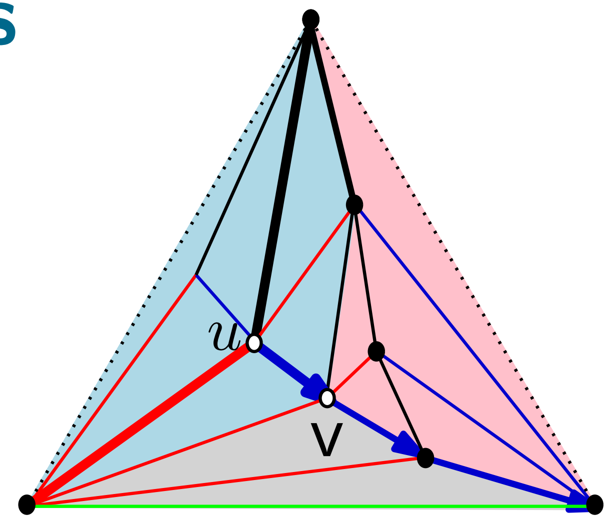
$$\boxed{\begin{matrix} u_i < v_i \\ u_j > v_j \end{matrix}}$$

Lemma: The Schnyder layout is a barycentric representation

Corollary: The Schnyder layout is crossing free

Regions and coordinates

Remarks: Let (u,v) of color i oriented from u to v



$$v =: \frac{|R_0(v)|}{|F|-1}x_0 + \frac{|R_1(v)|}{|F|-1}x_1 + \frac{|R_2(v)|}{|F|-1}x_2 =$$
$$= \frac{v_0}{|F|-1}x_0 + \frac{v_1}{|F|-1}x_1 + \frac{v_2}{|F|-1}x_2$$

- $R_i(u) \subseteq R_i(v) \longrightarrow |R_i(u)| \leq |R_i(v)| \longrightarrow \boxed{u_i \leq v_i}$

v $(5, 6, 2) := (v_0, v_1, v_2)$

u $(7, 3, 3) := (u_0, u_1, u_2)$

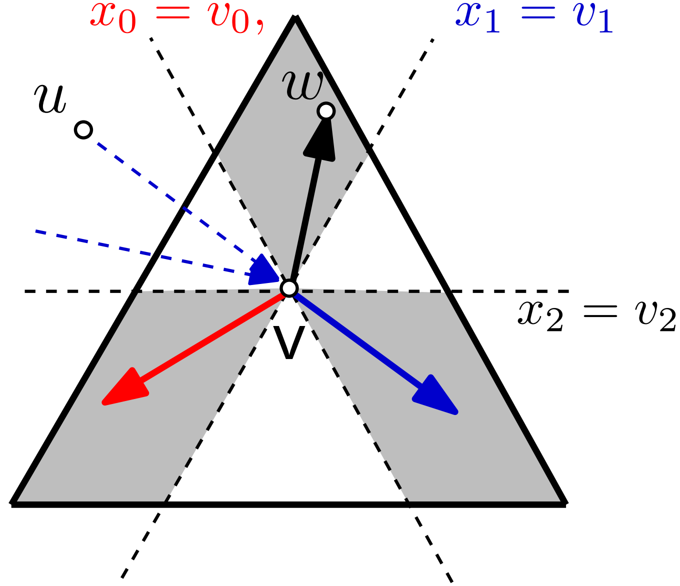
- $v_0 + v_1 + v_2 = f - 1$

- $\begin{matrix} R_i(u) \subset R_i(v) \\ R_{i+1}(v) \subset R_{i+1}(u) \\ R_{i-1}(v) \subset R_{i-1}(u) \end{matrix} \longrightarrow \left\{ \begin{matrix} u_i < v_i \\ u_{i+1} > v_{i+1} \\ u_{i-1} > v_{i-1} \end{matrix} \right.$

Remark:

is $u_i < v_i$ the u lies in the white sector $\begin{matrix} x_{i+1} > u_{i+1} \\ x_{i-1} > v_{i-1} \end{matrix}$

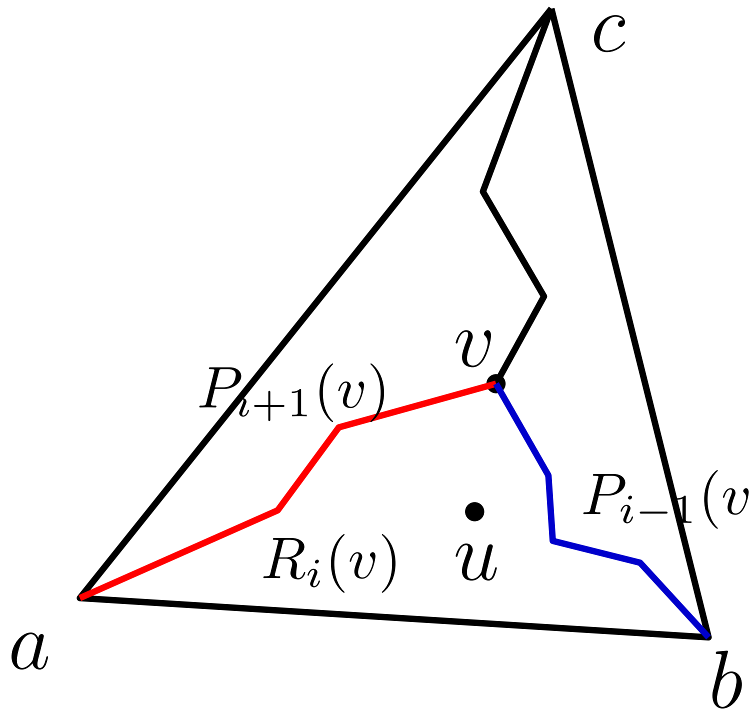
the outgoing edges (v, w) lie in the gray sectors

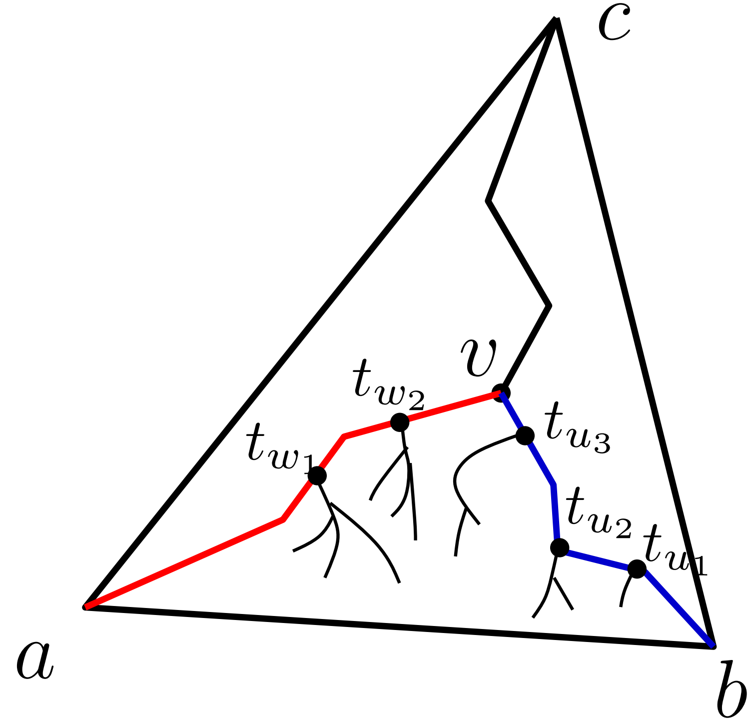


Linear-time implementation

(how to efficiently perform region counting)

Linear-time implementation

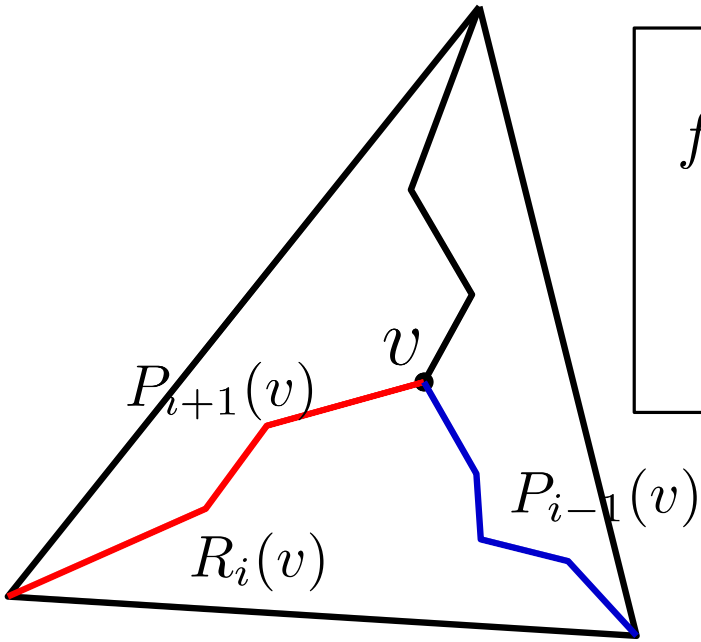




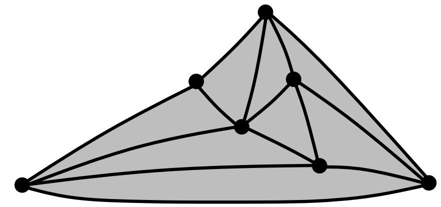
Problem: how to efficiently compute $|R_i(v)|$ (for all $v \in V$)?

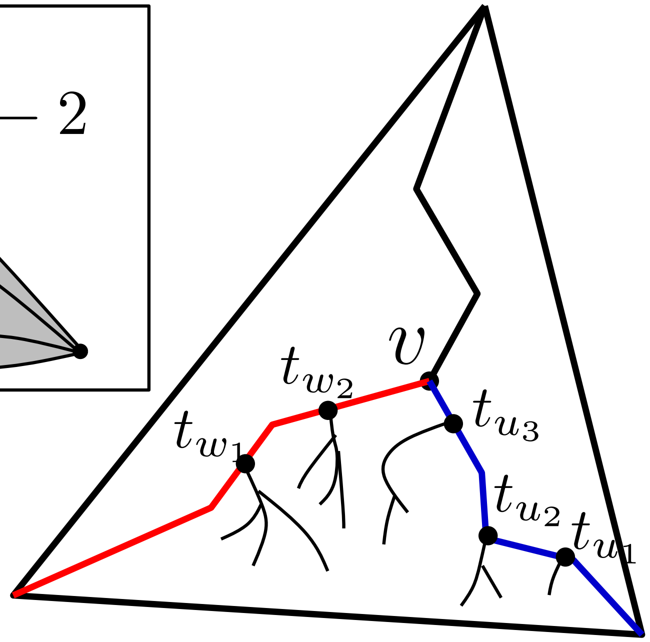
Remark: the number of faces $|R_i(v)|$ can be retrieved from: the number of inner vertices and the number of vertices on the path $P_{i+1}(v)$ and $P_{i-1}(v)$

Linear-time implementation



$$f_i = 2n_i + k - 2$$





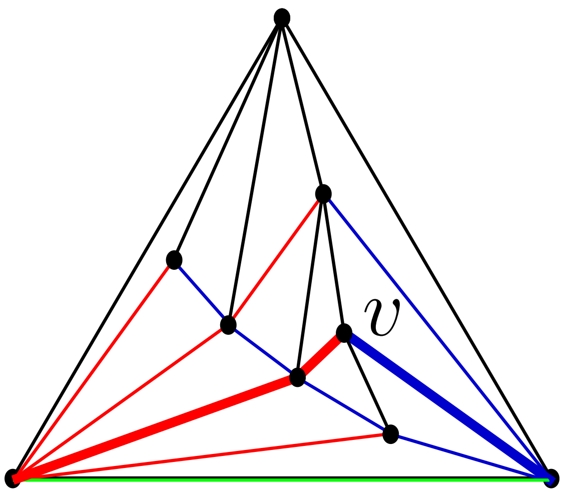
Problem: how to efficiently compute $|R_i(v)|$ (for all $v \in V$)?

Remark: the number of faces $|R_i(v)|$ can be retrieved from: the number of inner vertices and the number of vertices on the path $P_{i+1}(v)$ and $P_{i-1}(v)$

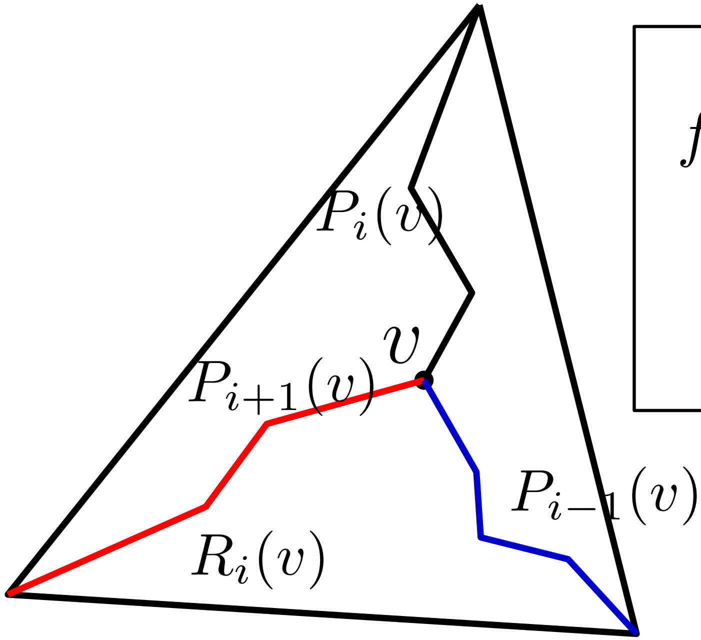
$R_i(v) = 4$
(inner faces)

$\partial R_i(v) := (P_{i+1}(v) + P_{i-1}(v)) - 1 = 4$
(outer vertices)

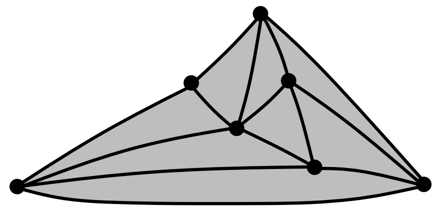
$\sum_{w \in P_{i+1}} |t_w| + \sum_u |t_u| = 1$ (inner vertices)

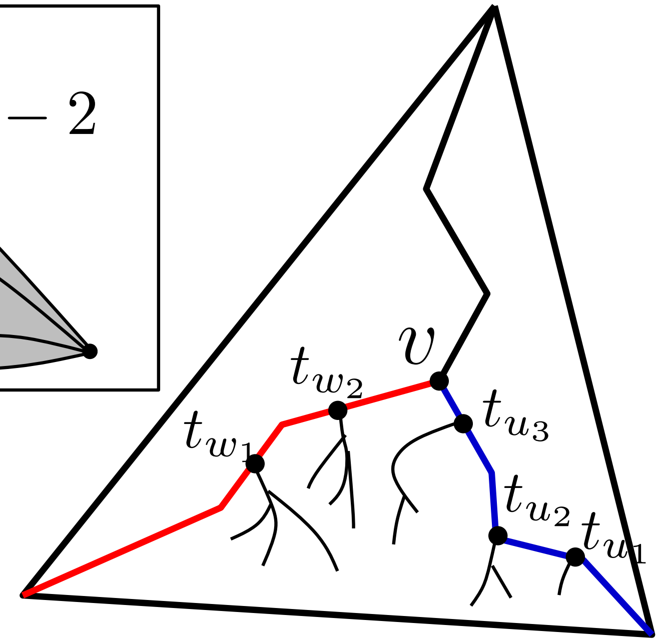


Linear-time implementation



$$f_i = 2n_i + k - 2$$



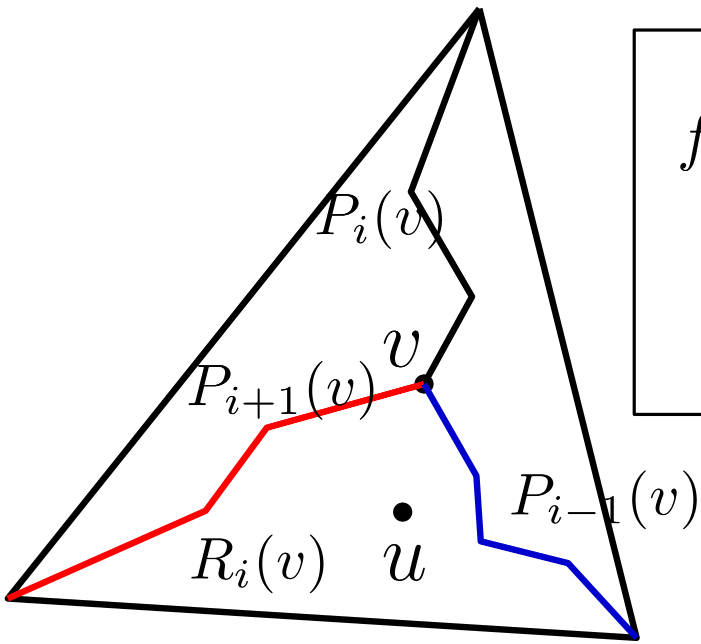


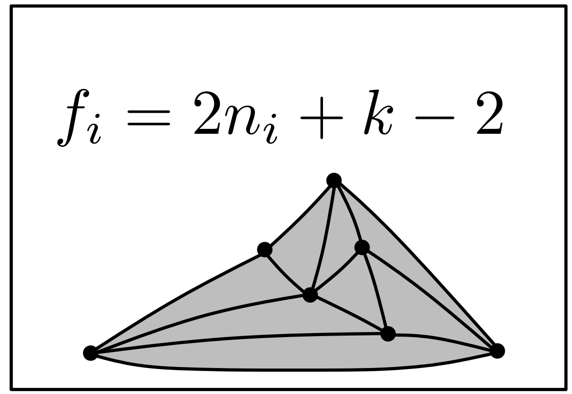
Problem: how to efficiently compute $|R_i(v)|$ (for all $v \in V$)?

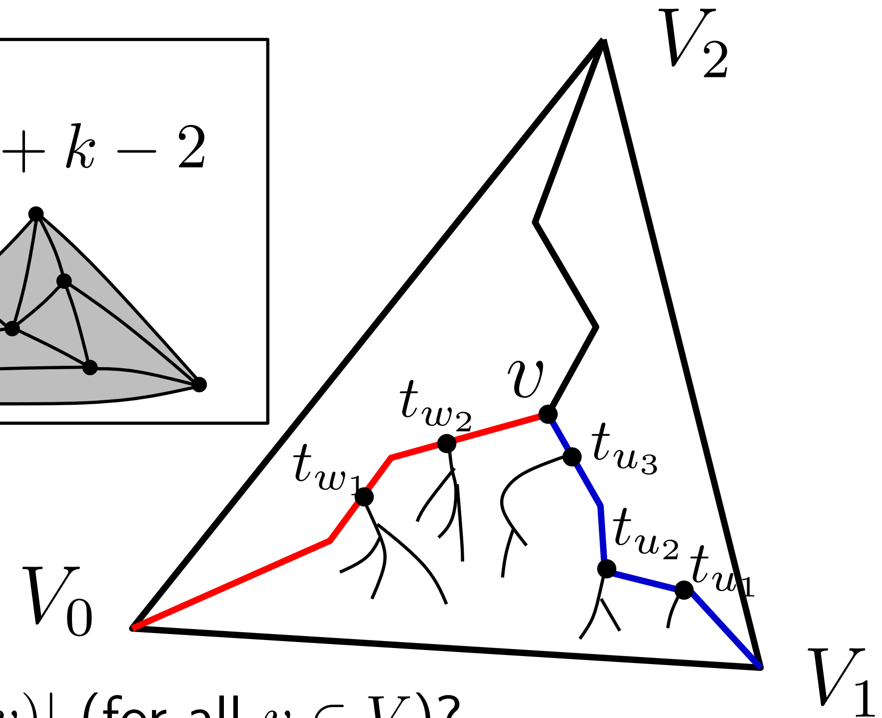
```
/* computes number of nodes in tree */
int size(Node node)
{
    if (node == null)
        return 0;
    else
        return (size(node.left) + 1 + size(node.right));
}
```

- Compute and store for each vertex v the subtree size of $T_0(v), T_1(v), T_2(v)$
- Compute the length of the paths $P_0(v), P_1(v), P_2(v)$
- cumulate the size of sub-trees for all vertices w_k, u_j on the paths $P_{i+1}(v), P_{i-1}(v)$

Linear-time implementation







Problem: how to efficiently compute $|R_i(v)|$ (for all $v \in V$)?

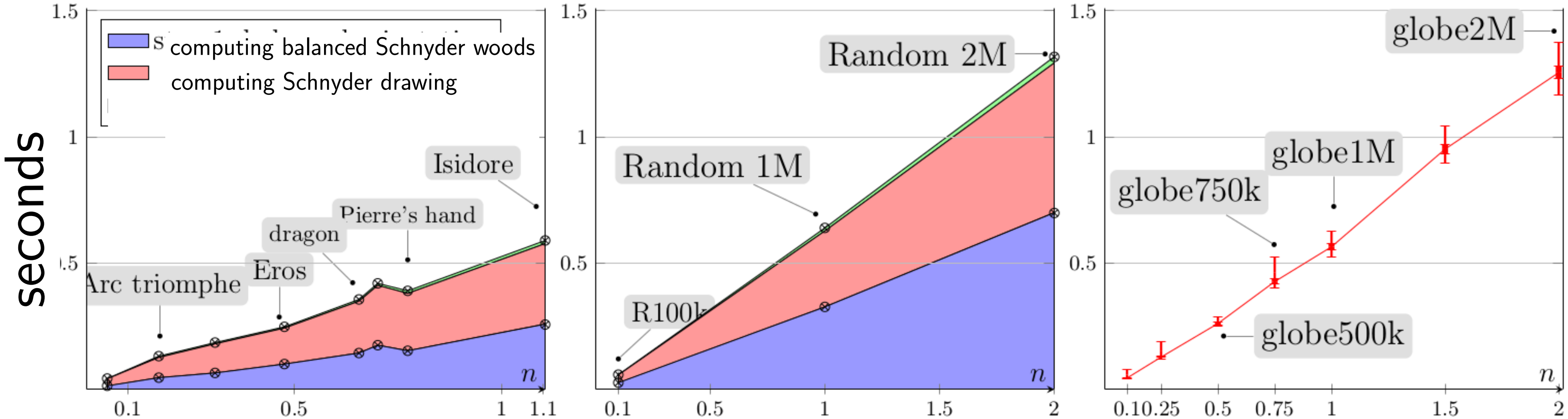
```
private static int finalSum = 0;
public static int nodeDepths(BinaryTree root) {
      // Write your code here.
      int runningSum = 0;
      depthHelper(root.left, runningSum);
      depthHelper(root.right, runningSum);
      return finalSum;
}

  private static void depthHelper(BinaryTree node, int runningSum) {
      if(node == null) return;
      runningSum++;
      finalSum += runningSum;
      depthHelper(node.left, runningSum);
      depthHelper(node.right, runningSum);
  }
```

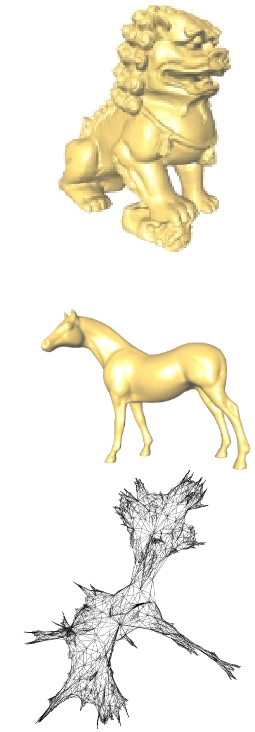
$$P_0(v) = \{V_0, w_1, w_2, \ldots v\}$$
$$P_1(v) = \{V_1, u_1, u_2, \ldots v\}$$

- Compute and store for each vertex v the subtree size of $T_0(v), T_1(v), T_2(v)$
- Compute the length of the paths $P_0(v), P_1(v), P_2(v)$
- cumulate the size of sub-trees for all vertices w_k, u_j on the paths $P_{i+1}(v), P_{i-1}(v)$

Practical performances

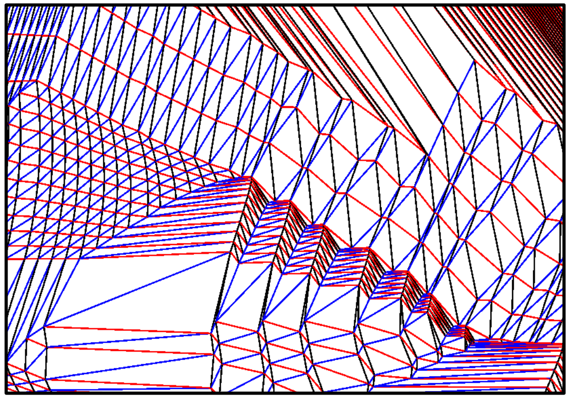
average timings (over 100 executions)

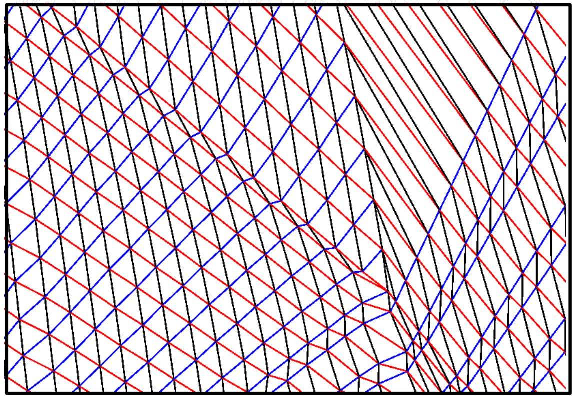




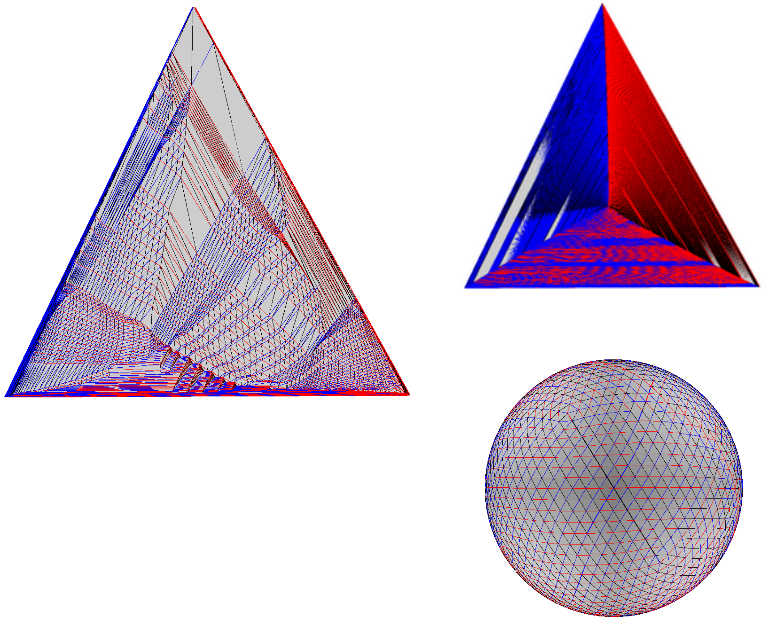
Timing performances (pure **Java**, on a core i7-5600 U, 2.60GHz, 1GB Ram):
Schnyder woods can process $\approx 1.43M - 1.92M$ vertices/seconds

---





Two Schnyder drawings of a sphere graph



Practical performances