Realizations of (v_r) configurations – Lecture 5

Marko Boben

Discrete Mathematics 2 + Configurations

Today's Topics

- We will define several types of *realizations* of configurations (in the Euclidean plane)
- We will present several methods (algorithms) one can use to realize configurations in the plane.
- We will define *polycyclic configurations* and discuss realizations of some special types of polycyclic configurations.

Today's Topics

- We will define several types of *realizations* of configurations (in the Euclidean plane)
- We will present several methods (algorithms) one can use to realize configurations in the plane.
- We will define *polycyclic configurations* and discuss realizations of some special types of polycyclic configurations.

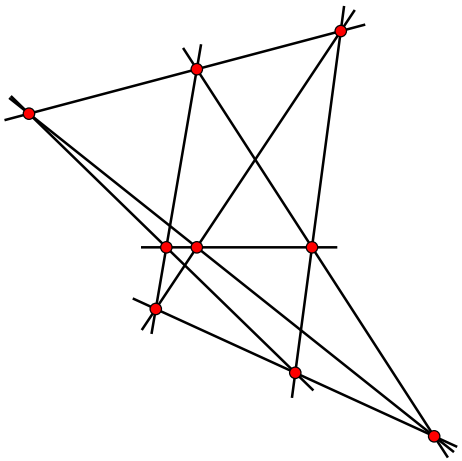
Today's Topics

- We will define several types of *realizations* of configurations (in the Euclidean plane)
- We will present several methods (algorithms) one can use to realize configurations in the plane.
- We will define *polycyclic configurations* and discuss realizations of some special types of polycyclic configurations.

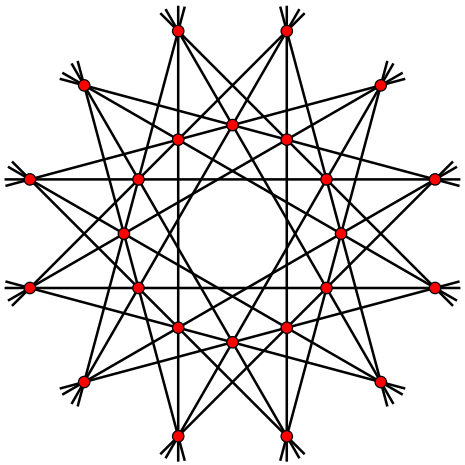
Today's Topics

- We will define several types of *realizations* of configurations (in the Euclidean plane)
- We will present several methods (algorithms) one can use to realize configurations in the plane.
- We will define *polycyclic configurations* and discuss realizations of some special types of polycyclic configurations.

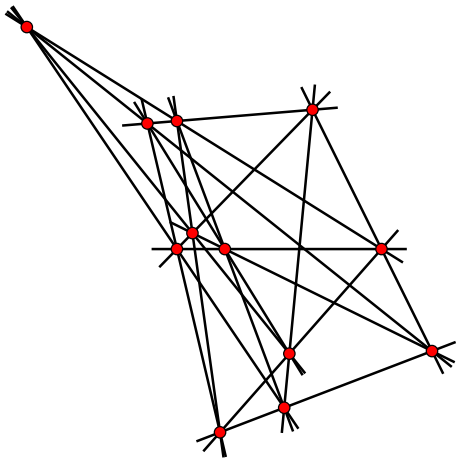
Examples of configurations *realized* in the plane (i.e. *geometric* configurations)



(9_3)



(24_4)



$(12_4, 16_3)$

Definition

A *realization* of a configuration $\mathcal{C}$ with point-set $\mathcal{C}$ and line-set $\mathcal{L}$ in the plane $\mathbb{R}^2$ is an 1-to-1 mapping φ which maps the points of $\mathcal{C}$ to the points of $\mathbb{R}^2$, the lines of $\mathcal{C}$ to the lines of $\mathbb{R}^2$ and preserves incidence, i.e.

$$p \text{ is incident with } L \text{ in } \mathcal{C} \implies \varphi(p) \text{ is incident with } \varphi(L) \text{ in } \mathbb{R}^2$$

for each $p \in \mathcal{C}$ and $L \in \mathcal{L}$.

A realization is called *strong* if

$$\varphi(p) \text{ is incident with } \varphi(L) \text{ in } \mathbb{R}^2 \implies p \text{ is incident with } L \text{ in } \mathcal{C}$$

for each $p \in \mathcal{C}$ and $L \in \mathcal{L}$.

Definition

A *realization* of a configuration $\mathcal{C}$ with point-set $\mathcal{C}$ and line-set $\mathcal{L}$ in the plane $\mathbb{R}^2$ is an 1-to-1 mapping φ which maps the points of $\mathcal{C}$ to the points of $\mathbb{R}^2$, the lines of $\mathcal{C}$ to the lines of $\mathbb{R}^2$ and preserves incidence, i.e.

$$p \text{ is incident with } L \text{ in } \mathcal{C} \implies \varphi(p) \text{ is incident with } \varphi(L) \text{ in } \mathbb{R}^2$$

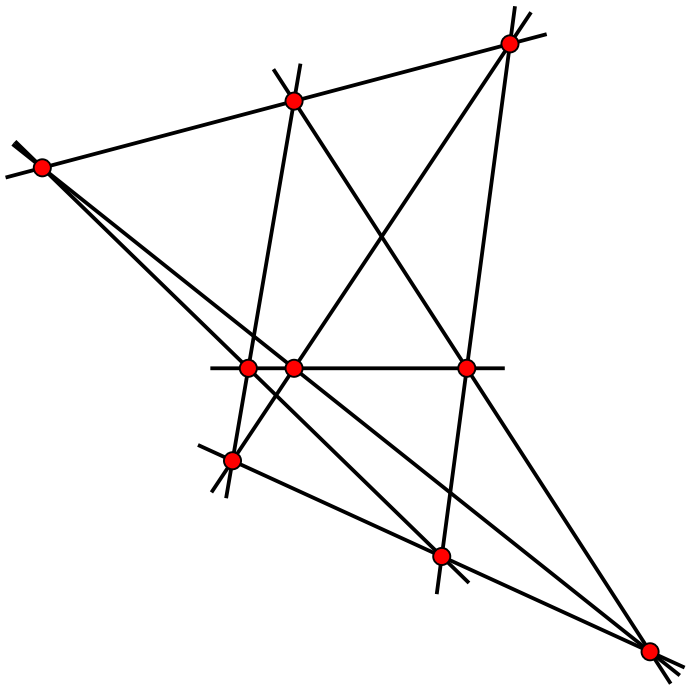
for each $p \in \mathcal{C}$ and $L \in \mathcal{L}$.

A realization is called *strong* if

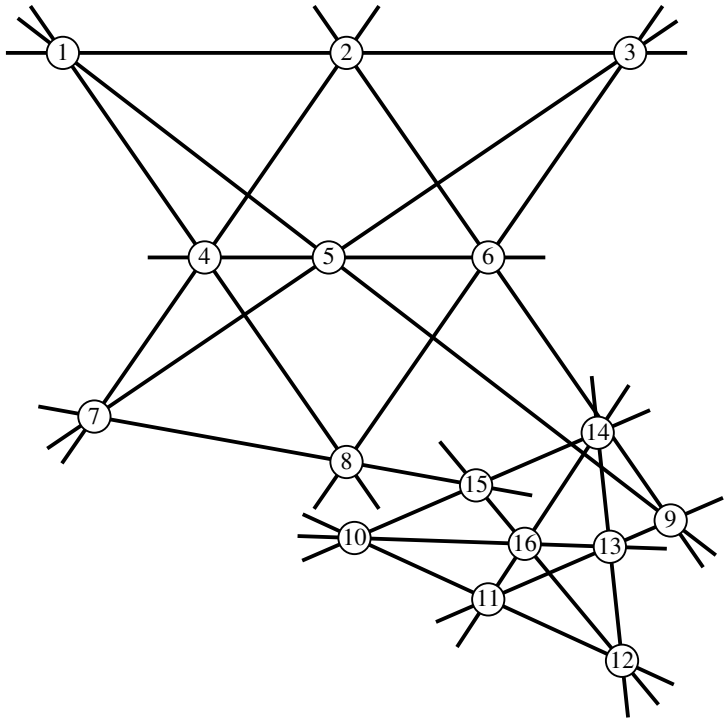
$$\varphi(p) \text{ is incident with } \varphi(L) \text{ in } \mathbb{R}^2 \implies p \text{ is incident with } L \text{ in } \mathcal{C}$$

for each $p \in \mathcal{C}$ and $L \in \mathcal{L}$.

Examples of a strong realization, and of a *weak* realization (a realization which is not strong).

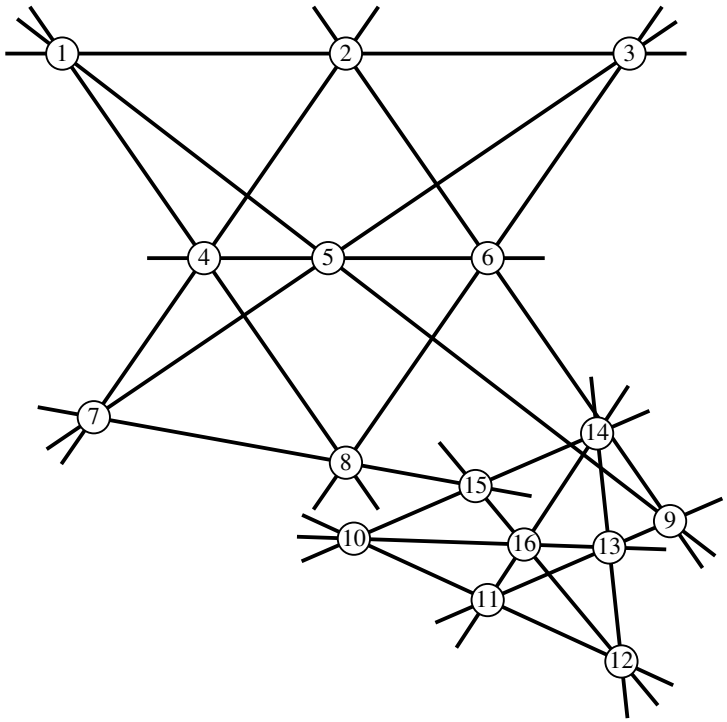


Strong realization of Pappus configuration

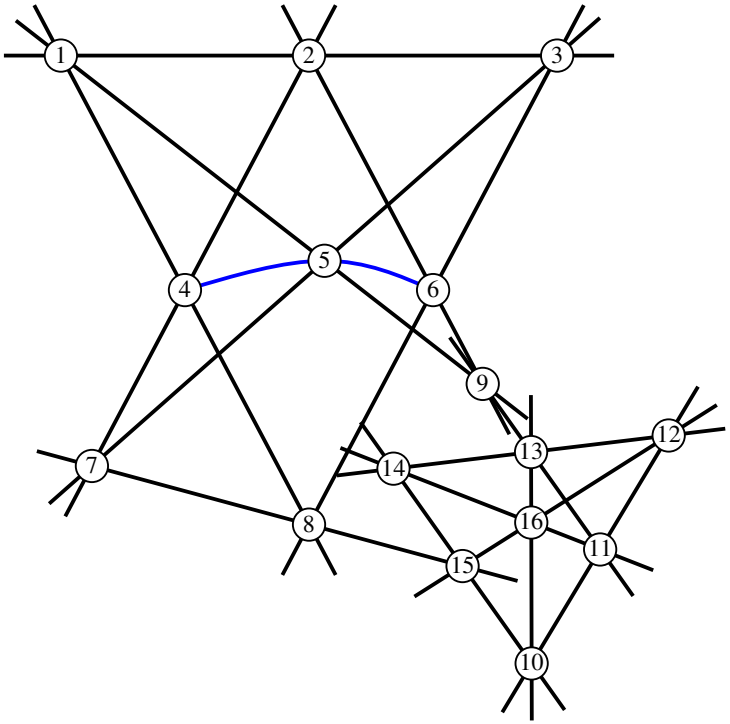


Weak realization of a (16_3) configuration.
(Pappus theorem!)

Some theory about realizations
Not strongly realizable configurations

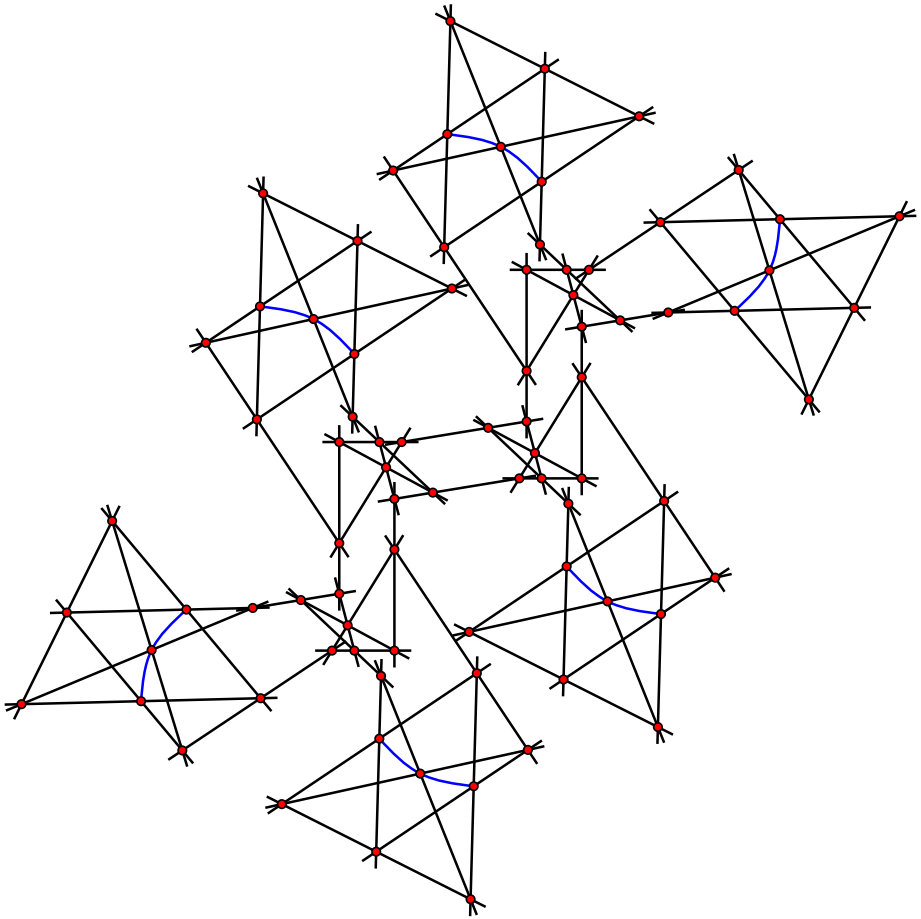


Strongly non-realizable configuration $\mathcal{C}$



Strongly realizable incidence structure ($\mathcal{C}$ without line $\{4, 5, 6\}$)

There exist arbitrary large strongly non-realizable configurations...



Considering a reprezentation of configurations in the plane we try to find

- a drawing with “as little curved lines as possible” (i.e. a realization of an incidence structure obtained from that configuration by “not removing too many lines”);
- a realization;
- a “nice” drawing or realization (e.g. with geometric symmetries reflecting some combinatorial symmetries);

Considering a reprezentation of configurations in the plane we try to find

- a drawing with “as little curved lines as possible” (i.e. a realization of an incidence structure obtained from that configuration by “not removing too many lines”);
- a realization;
- a “nice” drawing or realization (e.g. with geometric symmetries reflecting some combinatorial symmetries);

Considering a reprezentation of configurations in the plane we try to find

- a drawing with “as little curved lines as possible” (i.e. a realization of an incidence structure obtained from that configuration by “not removing too many lines”);
- a realization;
- a “nice” drawing or realization (e.g. with geometric symmetries reflecting some combinatorial symmetries);

Considering a reprezentation of configurations in the plane we try to find

- a drawing with “as little curved lines as possible” (i.e. a realization of an incidence structure obtained from that configuration by “not removing too many lines”);
- a realization;
- a “nice” drawing or realization (e.g. with geometric symmetries reflecting some combinatorial symmetries);

Theorem

In 1894 E. Steinitz proved a theorem which says that it is possible to draw each connected (v_3) configuration in Euclidean plane with at most one curved line.

He uses an *iterative method* to prove his theorem:
He successively draws points and lines such that he is always able to construct a point as an arbitrary point on some (already constructed) line or a point which is an itersection of two lines / a line through one or two some already constructed lines.

Theorem

In 1894 E. Steinitz proved a theorem which says that it is possible to draw each connected (v_3) *configuration in Euclidean plane with at most one curved line.*

He uses an *iterative method* to prove his theorem:
He successively draws points and lines such that he is always able to construct a point as an arbitrary point on some (already constructed) line or a point which is an itersection of two lines / a line through one or two some already constructed lines.

We will try to present Steinitz method using *incidence* or *Levi graphs*.

A *Levi graph* $G(\mathcal{C})$ of a (v_r, b_k) configuration $\mathcal{C}$ is a bipartite graph

- with v *black* vertices representing points of $\mathcal{C}$,
- b *white* vetrices representing lines of $\mathcal{C}$;
- there is an edge between black and white vertex iff the corresponding point and line are incident.

(Levi graphs of configurations have girth ≥ 6.)

We will try to present Steinitz method using *incidence* or *Levi graphs*.

A *Levi graph* $G(\mathcal{C})$ of a (v_r, b_k) configuration $\mathcal{C}$ is a bipartite graph

- with v *black* vertices representing points of $\mathcal{C}$,
- b *white* vetrices representing lines of $\mathcal{C}$;
- there is an edge between black and white vertex iff the corresponding point and line are incident.

(Levi graphs of configurations have girth ≥ 6.)

We will try to present Steinitz method using *incidence* or *Levi graphs*.

A *Levi graph* $G(\mathcal{C})$ of a (v_r, b_k) configuration $\mathcal{C}$ is a bipartite graph

- with v *black* vertices representing points of $\mathcal{C}$,
- b *white* vetrices representing lines of $\mathcal{C}$;
- there is an edge between black and white vertex iff the corresponding point and line are incident.

(Levi graphs of configurations have girth ≥ 6.)

We will try to present Steinitz method using *incidence* or *Levi graphs*.

A *Levi graph* $G(\mathcal{C})$ of a (v_r, b_k) configuration $\mathcal{C}$ is a bipartite graph

- with v *black* vertices representing points of $\mathcal{C}$,
- b *white* vetrices representing lines of $\mathcal{C}$;
- there is an edge between black and white vertex iff the corresponding point and line are incident.

(Levi graphs of configurations have girth ≥ 6.)

We will try to present Steinitz method using *incidence* or *Levi graphs*.

A *Levi graph* $G(\mathcal{C})$ of a (v_r, b_k) configuration $\mathcal{C}$ is a bipartite graph

- with v *black* vertices representing points of $\mathcal{C}$,
- b *white* vetrices representing lines of $\mathcal{C}$;
- there is an edge between black and white vertex iff the corresponding point and line are incident.

(Levi graphs of configurations have girth ≥ 6.)

Order of construction can be given by a directed spanning subgraph H of Levi graph G which

- does not have (directed) cycles,
- $\mathrm{outdeg}_H(v) \leq 2$.

An edge $(u, v) \in E(H)$ (and edge "from u to v") means that the object u (either point or line) is constructed from the object v.

The number of actually "constructed" ("realized") lines is the number of vertices w in H representing lines (white vertices) for which

$$\mathrm{outdeg}_H(w) + \mathrm{indeg}_H(w) = \deg_G(w) \qquad (*)$$

Order of construction can be given by a directed spanning subgraph H of Levi graph G which

- does not have (directed) cycles,
- $\mathrm{outdeg}_H(v) \leq 2$.

An edge $(u, v) \in E(H)$ (and edge "from u to v") means that the object u (either point or line) is constructed from the object v.

The number of actually "constructed" ("realized") lines is the number of vertices w in H representing lines (white vertices) for which

$$\mathrm{outdeg}_H(w) + \mathrm{indeg}_H(w) = \deg_G(w) \qquad (*)$$

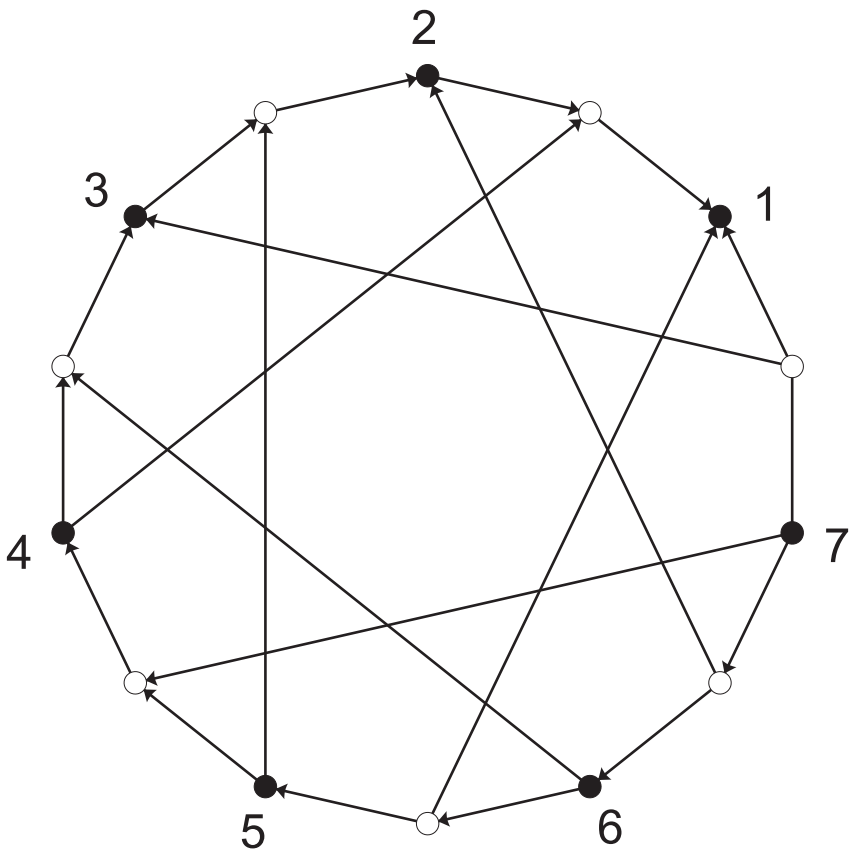
Order of construction can be given by a directed spanning subgraph H of Levi graph G which

- does not have (directed) cycles,
- $\mathrm{outdeg}_H(v) \leq 2$.

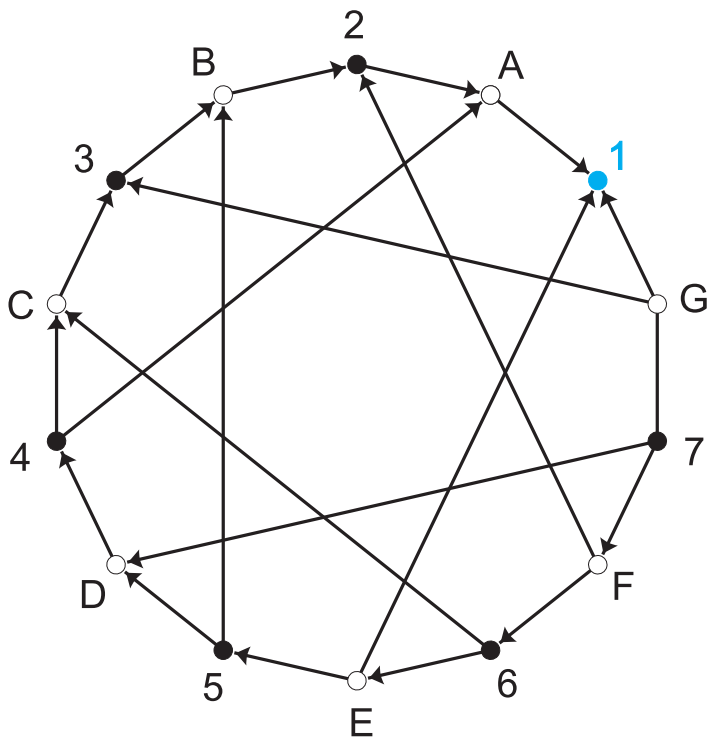
An edge $(u, v) \in E(H)$ (and edge "from u to v") means that the object u (either point or line) is constructed from the object v.

The number of actually "constructed" ("realized") lines is the number of vertices w in H representing lines (white vertices) for which

$$\mathrm{outdeg}_H(w) + \mathrm{indeg}_H(w) = \deg_G(w) \qquad (*)$$

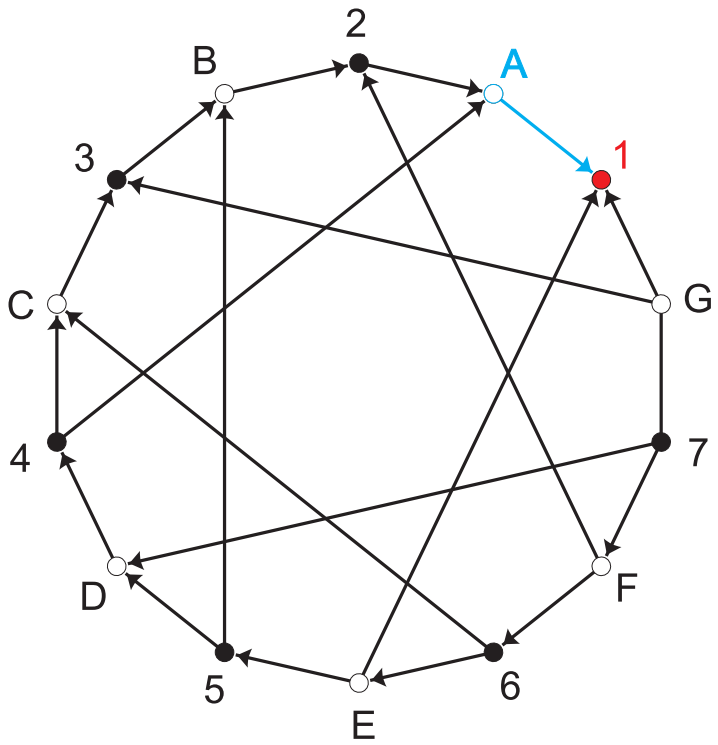


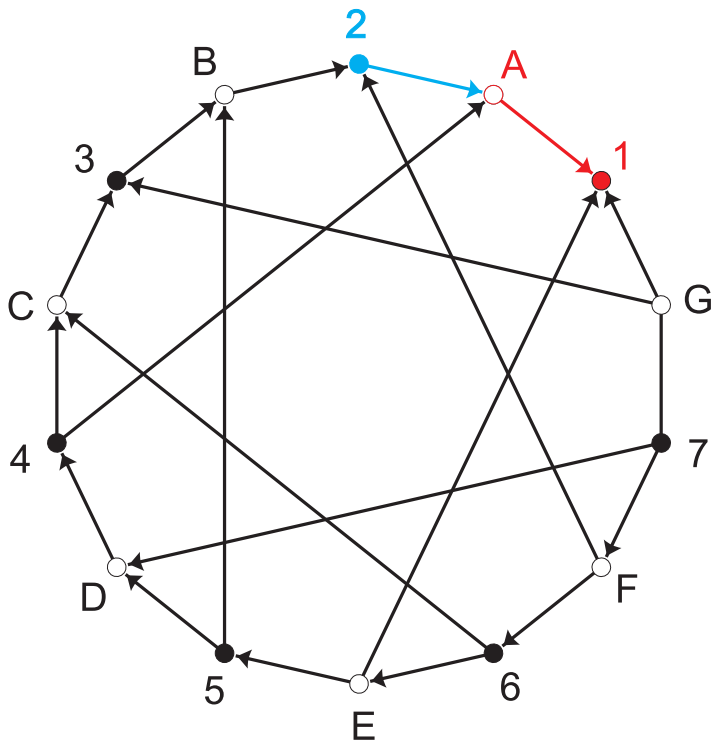
Heawood graph – Levi graph of Fano plane


2
B
A
3
1
C
G
4
7
D
F
5
6
E


1

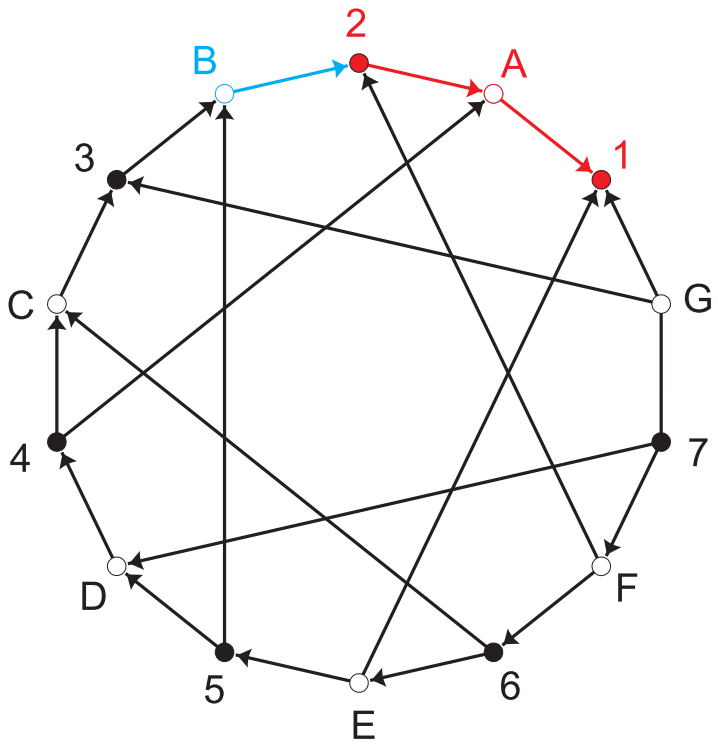
Algorithms
Example

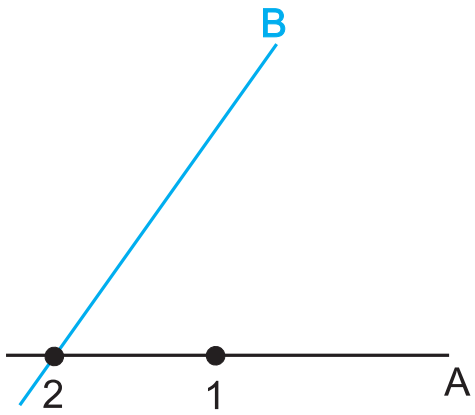


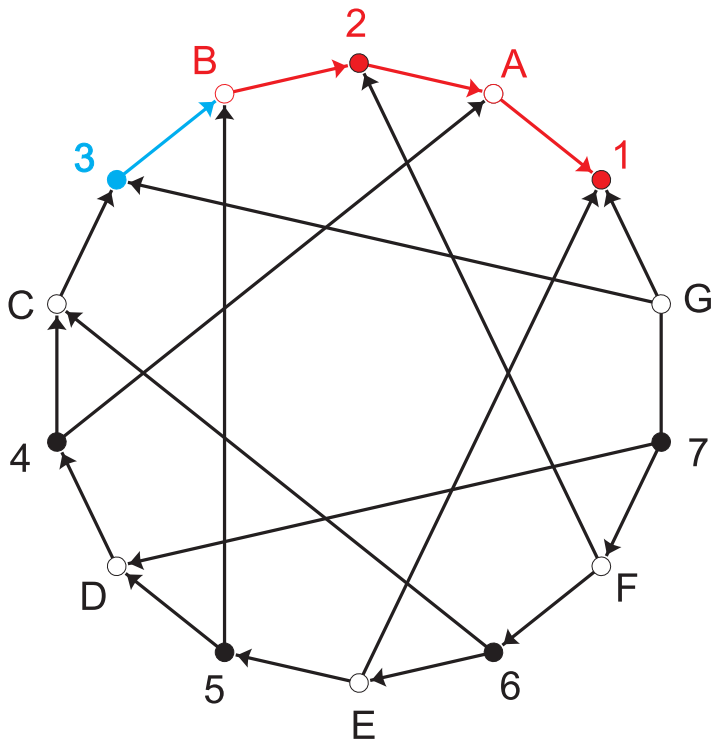

2
B
A
1
3
C
G
4
7
D
F
5
6
E

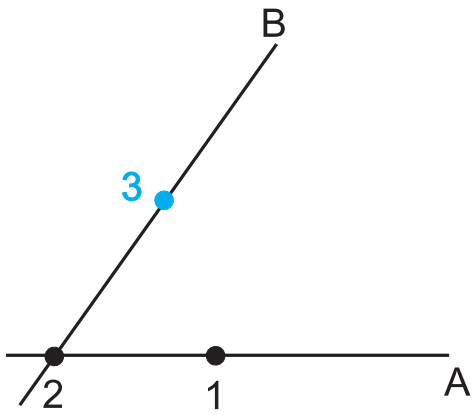

2
1
A

Algorithms
Example

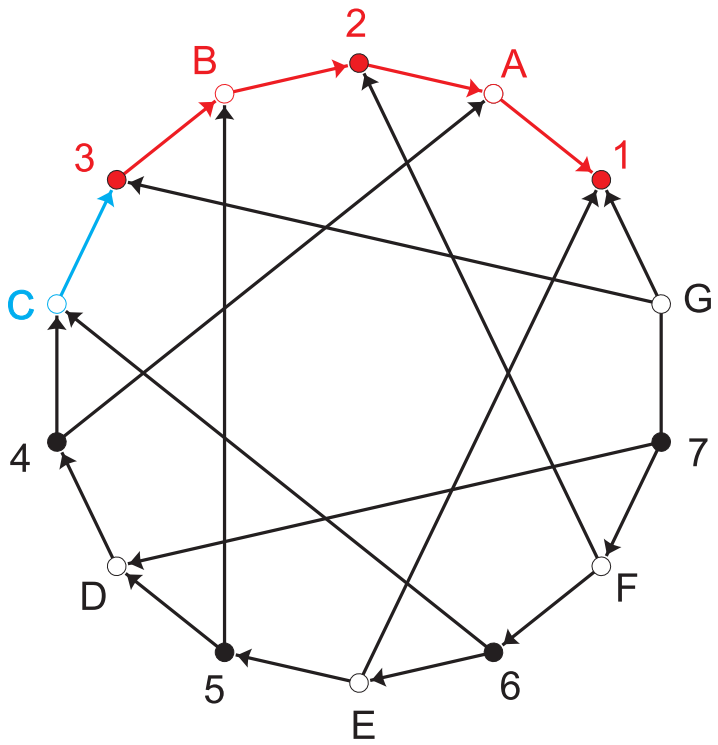


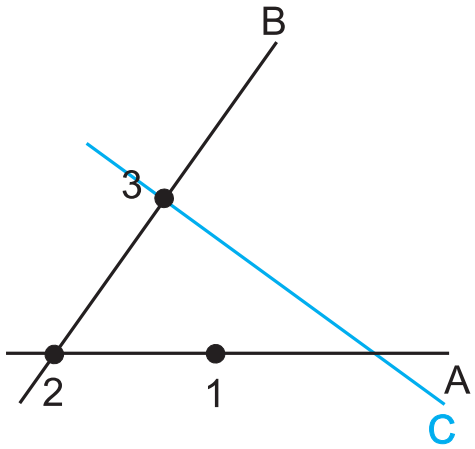


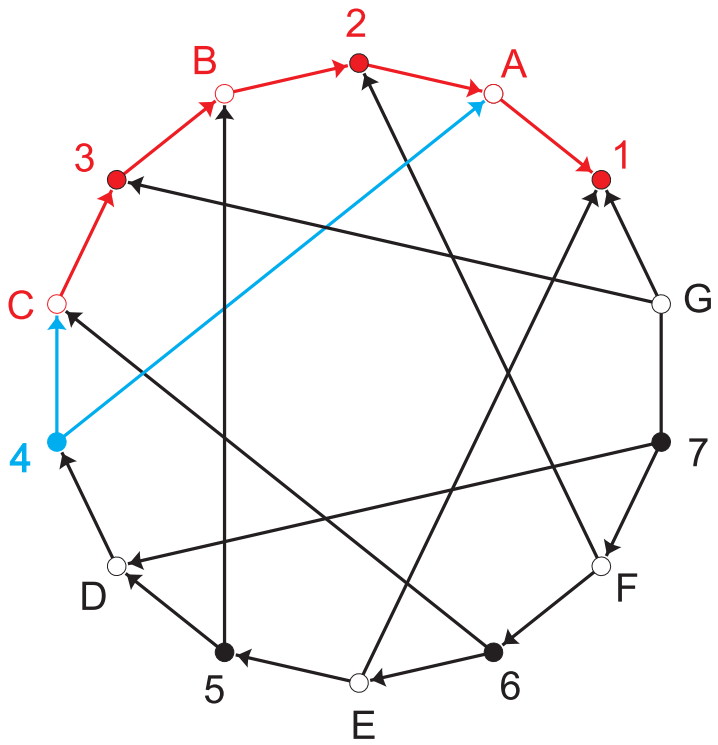

2
B
A
3
1
C
G
4
7
D
F
5
6
E

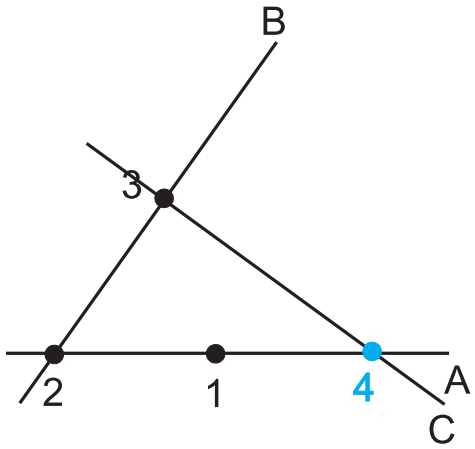

B
3
2
1
A

Algorithms
Example

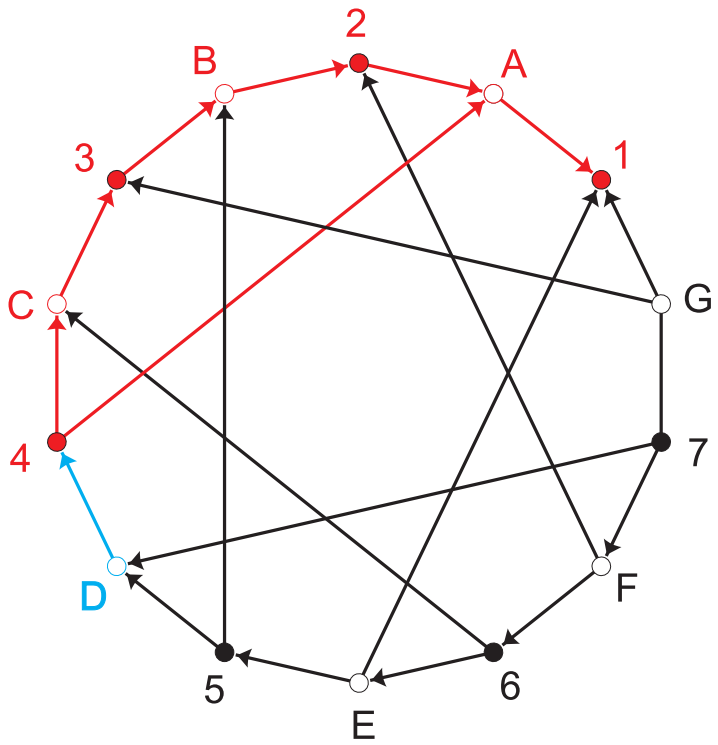


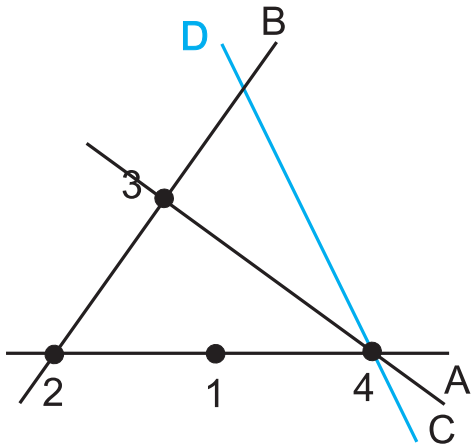



2
B
A
3
1
C
G
4
7
D
F
5
6
E

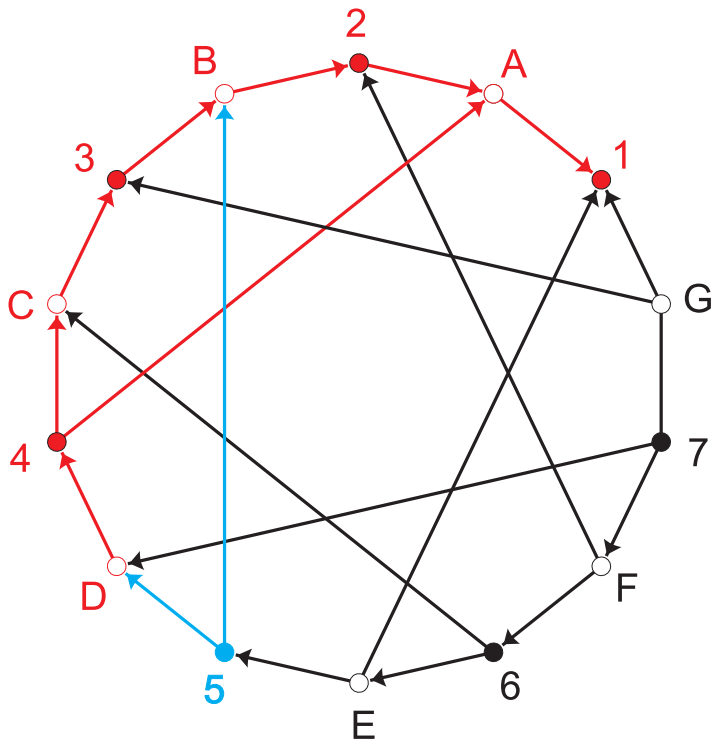

B
3
2
1
4
A
C

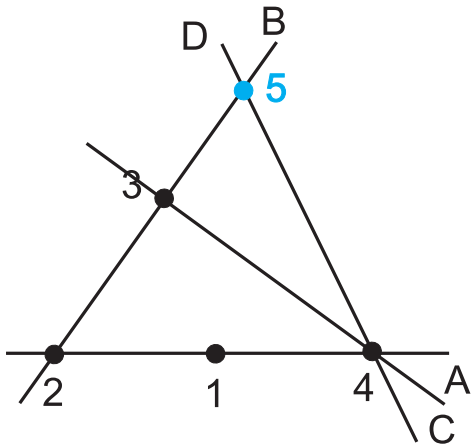
Algorithms
Example



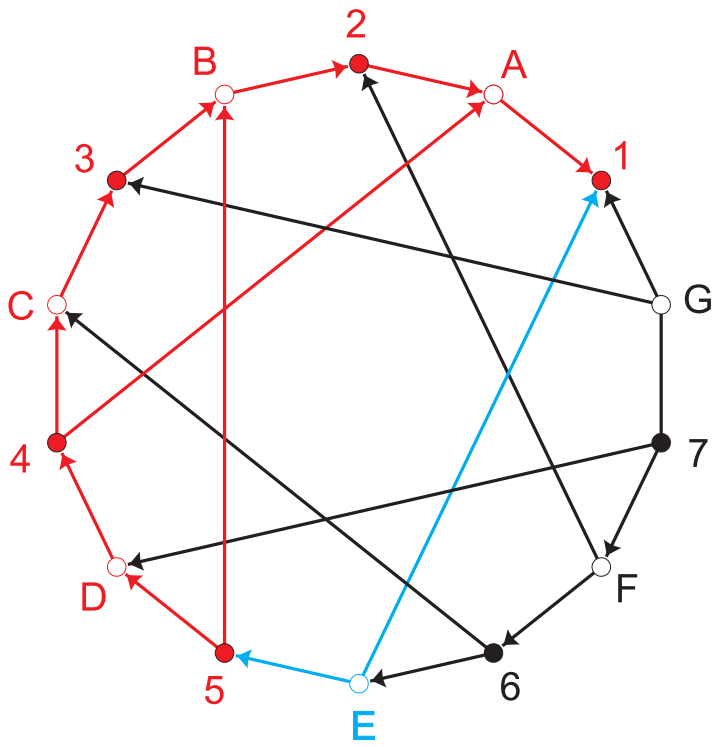


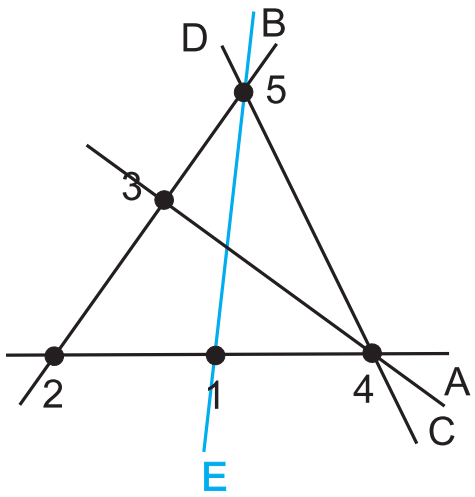
Algorithms
Example



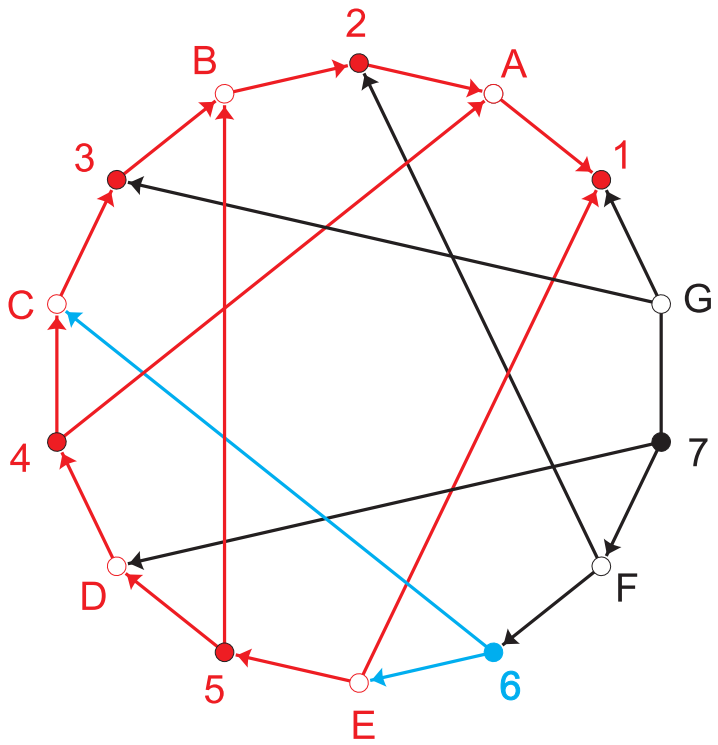


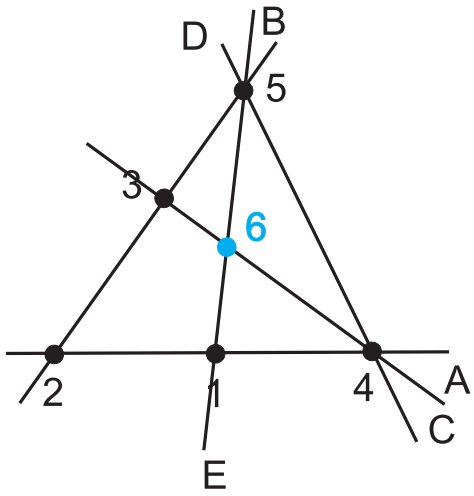
Algorithms
Example



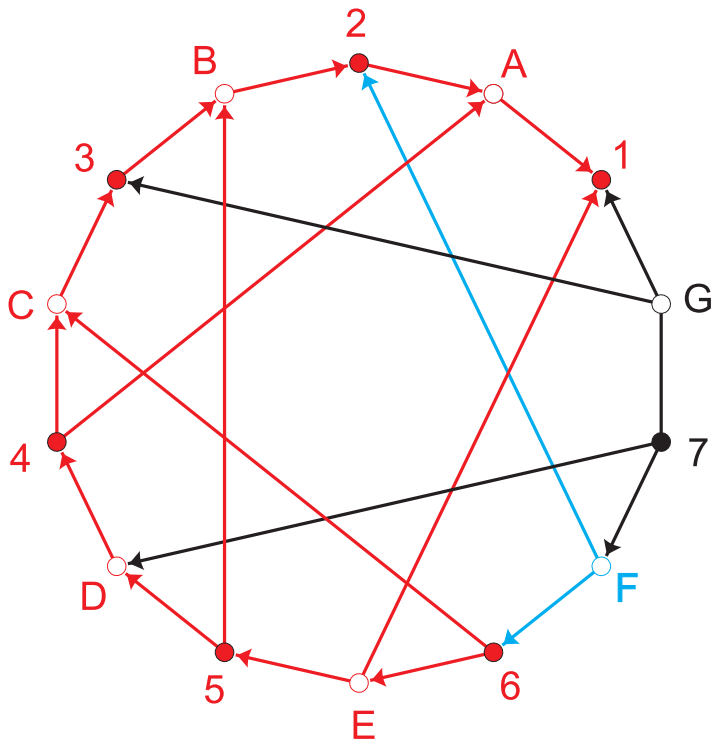


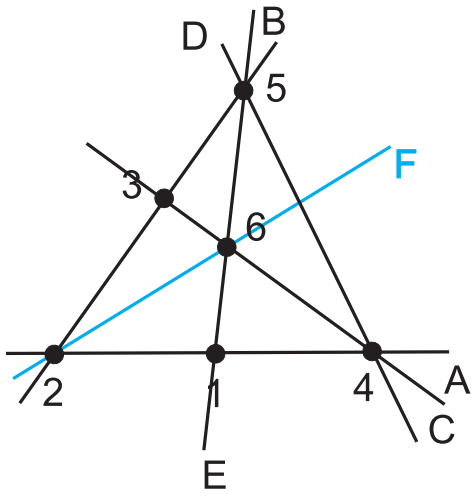
Algorithms
Example



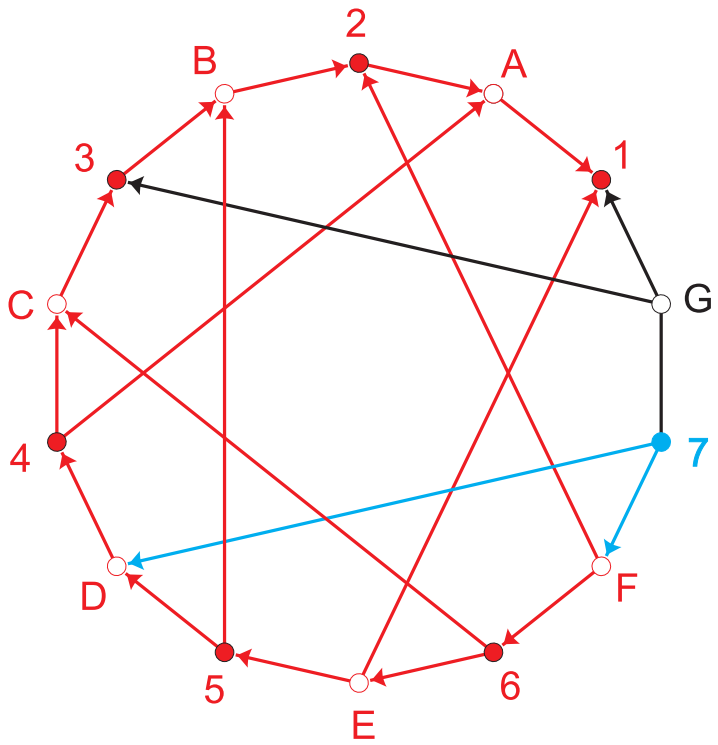


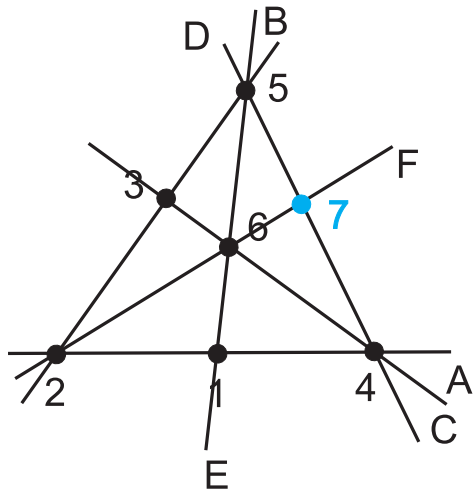
Algorithms
Example

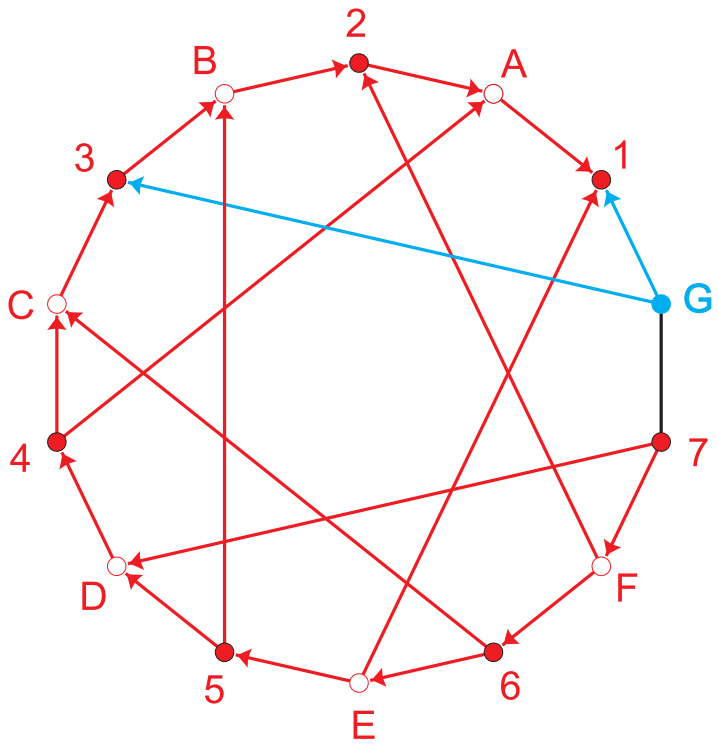


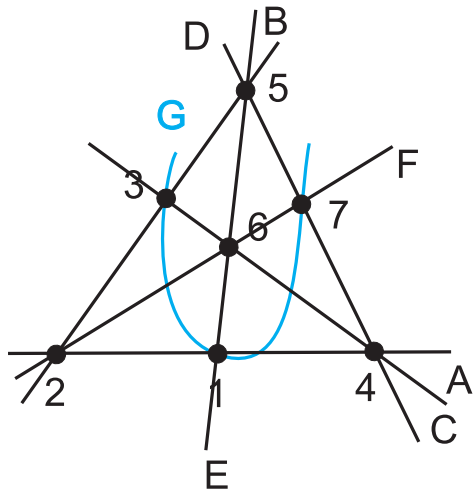


Algorithms
Example






2
B
A
3
1
C
G
4
7
D
F
5
6
E


D
B
5
G
F
3
7
6
2
1
4
A
C
E

Steinitz theorem says that in the case of connected (v_3) configurations, only one line "violates" the condition (*).

Sketch of the (original) proof:

- Every bipartite cubic graph contains a 2-factor (i.e. one can find a decomposition of the graph into ℓ disjoint cycles, covering all vertices).
- Next, order the cycles (2-factors) in the following way:
  - choose the first cycle C_1 arbitrarily,
  - each next cycle C_i must have an edge e_i with one end-vertex, v_i, in C_i and the other one, w_j in C_j, $j < i$ (this can be done since the graph is connected).

Steinitz theorem says that in the case of connected (v_3) configurations, only one line "violates" the condition (*).

Sketch of the (original) proof:

- Every bipartite cubic graph contains a 2-factor (i.e. one can find a decomposition of the graph into ℓ disjoint cycles, covering all vertices).
- Next, order the cycles (2-factors) in the following way:
  - choose the first cycle C_1 arbitrarily,
  - each next cycle C_i must have an edge e_i with one end-vertex, v_i, in C_i and the other one, w_j in C_j, $j < i$ (this can be done since the graph is connected).

Steinitz theorem says that in the case of connected (v_3) configurations, only one line "violates" the condition (*).

Sketch of the (original) proof:

- Every bipartite cubic graph contains a 2-factor (i.e. one can find a decomposition of the graph into ℓ disjoint cycles, covering all vertices).
- Next, order the cycles (2-factors) in the following way:
  - choose the first cycle C_1 arbitrarily,
  - each next cycle C_i must have an edge e_i with one end-vertex, v_i, in C_i and the other one, w_j in C_j, $j < i$ (this can be done since the graph is connected).

Algorithms
...

- Direct edges on each cycle such that v_i is the "last" vertex on that cycle (on v_1 take an arbitrary vertex for the "last" one, v_1).
- Direct edges e_i from w_j to v_i.
- other edges (except for one edge incident to v_1) should be directed s.t. the end vertex belongs to the same or to one of the next cycles.

This construction gives the construction subgraph missing only one edge of the original graph.

It follows that only one line can not be constructed.

- Direct edges on each cycle such that v_i is the "last" vertex on that cycle (on v_1 take an arbitrary vertex for the "last" one, v_1).
- Direct edges e_i from w_j to v_i.
- other edges (except for one edge incident to v_1) should be directed s.t. the end vertex belongs to the same or to one of the next cycles.

This construction gives the construction subgraph missing only one edge of the original graph.

It follows that only one line can not be constructed.

Alternative construction:

- Take a distance partition of the levi graph starting from one (line) vertex v, i.e. partition vertices into disjoint sets $D_0 = \{v\}, D_1, \ldots, D_k$. such that $\mathrm{dist}(v, u) = i$ for each $u \in D_i$.
- Direct an edge between D_i and D_{i+1} from the vertex in D_i to the vertex in D_{i+1} (This works for all edges except for one adjacent to v)
- This gives an construction subraph leaving only one line (the one represented by v) “unconstructed”.

Algorithms

...

An approach in *Computational Synthetic Geometry* by J. Bokowski and B. Sturmfels works similarly, it just gives a different order of construction (a different directed sub-graph of the Levi graph).

If, for some vertex $v \in H$ (directed spanning subgraph of the configuration), we have $\mathrm{outdeg}_H(v) < 2$, it means that we have one or two free parameters.

For the lines which we were not able to realize, we get additional conditions in the sense of algebraic equation(s).

Algorithms

...

An approach in *Computational Synthetic Geometry* by J. Bokowski and B. Sturmfels works similarly, it just gives a different order of construction (a different directed sub-graph of the Levi graph).

If, for some vertex $v \in H$ (directed spanning subgraph of the configuration), we have $\mathrm{outdeg}_H(v) < 2$, it means that we have one or two free parameters.

For the lines which we were not able to realize, we get additional conditions in the sense of algebraic equation(s).

Algorithms
...

An approach in *Computational Synthetic Geometry* by J. Bokowski and B. Sturmfels works similarly, it just gives a different order of construction (a different directed sub-graph of the Levi graph).

If, for some vertex $v \in H$ (directed spanning subgraph of the configuration), we have $\mathrm{outdeg}_H(v) < 2$, it means that we have one or two free parameters.

For the lines which we were not able to realize, we get additional conditions in the sense of algebraic equation(s).

In *Mathematica*. . .

We can use the following “naïve” algorithm:

1. Place the points of the configuration $\mathcal{C}$ in the plane
   - randomly;
   - using an algorithm described above (leaving some lines “curved”).
2. Select points which should be on the same line L.
3. Place a line $\hat{L}$ through these points s.t. the sum of the squares of distances between a point an $\hat{L}$ is minimal.
4. Project the points to $\hat{L}$.
5. Repeat step 2 until all lines of $\mathcal{C}$ are “approximately” straight.

We can use the following "naïve" algorithm:

1. Place the points of the configuration $\mathcal{C}$ in the plane
   - randomly;
   - using an algorithm described above (leaving some lines "curved").
2. Select points which should be on the same line L.
3. Place a line $\hat{L}$ through these points s.t. the sum of the squares of distances between a point an $\hat{L}$ is minimal.
4. Project the points to $\hat{L}$.
5. Repeat step 2 until all lines of $\mathcal{C}$ are "approximately" straight.

We can use the following "naïve" algorithm:

1. Place the points of the configuration $\mathcal{C}$ in the plane
   - randomly;
   - using an algorithm described above (leaving some lines "curved").
2. Select points which should be on the same line L.
3. Place a line $\hat{L}$ through these points s.t. the sum of the squares of distances between a point an $\hat{L}$ is minimal.
4. Project the points to $\hat{L}$.
5. Repeat step 2 until all lines of $\mathcal{C}$ are "approximately" straight.

We can use the following "naïve" algorithm:

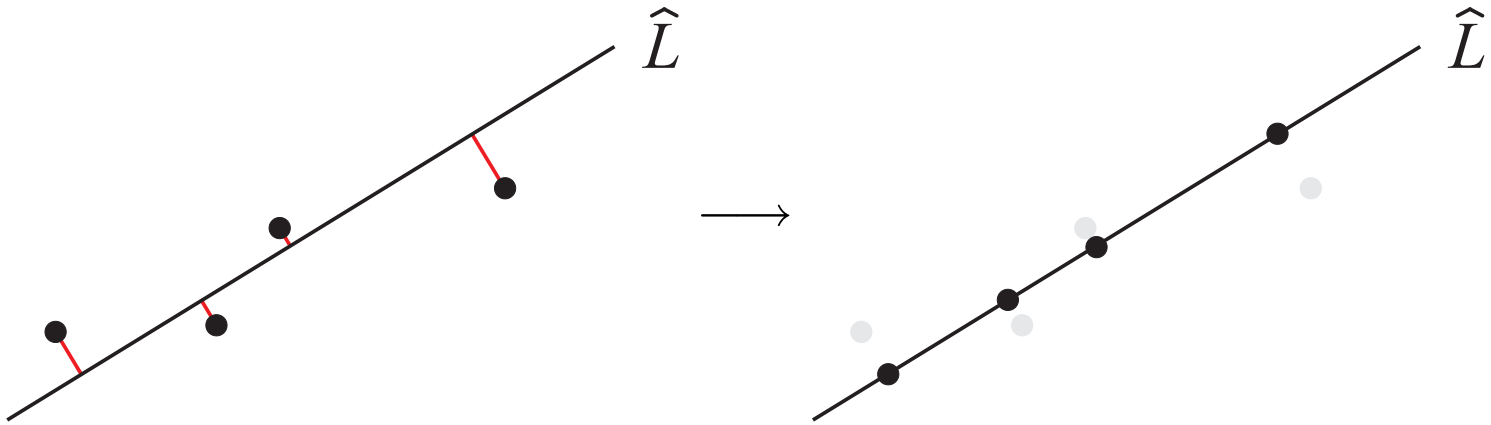
1. Place the points of the configuration $\mathcal{C}$ in the plane
   - randomly;
   - using an algorithm described above (leaving some lines "curved").
2. Select points which should be on the same line L.
3. Place a line $\hat{L}$ through these points s.t. the sum of the squares of distances between a point an $\hat{L}$ is minimal.
4. Project the points to $\hat{L}$.
5. Repeat step 2 until all lines of $\mathcal{C}$ are "approximately" straight.

We can use the following "naïve" algorithm:

1. Place the points of the configuration $\mathcal{C}$ in the plane
   - randomly;
   - using an algorithm described above (leaving some lines "curved").
2. Select points which should be on the same line L.
3. Place a line $\hat{L}$ through these points s.t. the sum of the squares of distances between a point an $\hat{L}$ is minimal.
4. Project the points to $\hat{L}$.
5. Repeat step 2 until all lines of $\mathcal{C}$ are "approximately" straight.

We can use the following "naïve" algorithm:

1. Place the points of the configuration $\mathcal{C}$ in the plane
   - randomly;
   - using an algorithm described above (leaving some lines "curved").
2. Select points which should be on the same line L.
3. Place a line $\hat{L}$ through these points s.t. the sum of the squares of distances between a point an $\hat{L}$ is minimal.
4. Project the points to $\hat{L}$.
5. Repeat step 2 until all lines of $\mathcal{C}$ are "approximately" straight.


$\widehat{L}$
$\widehat{L}$

In *Mathematica*. . .