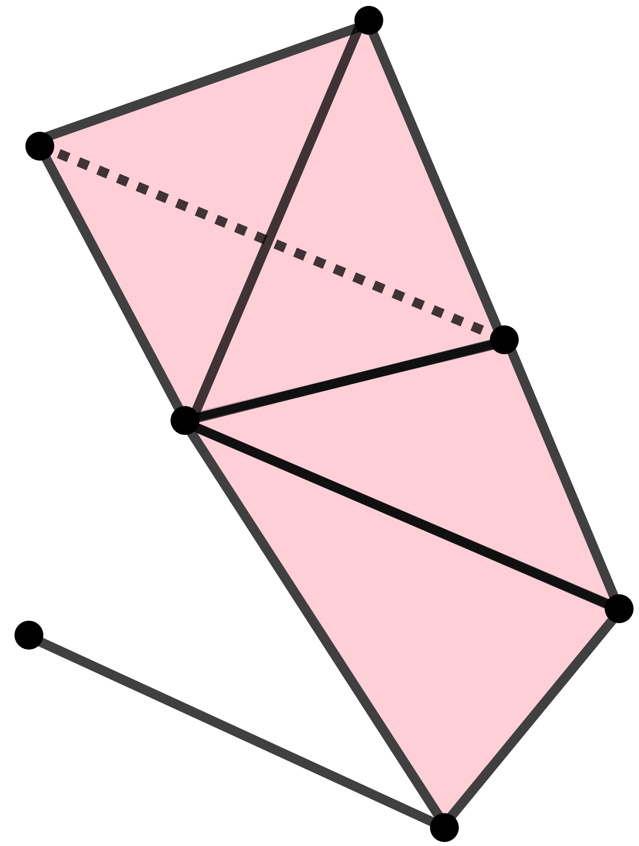
Abstract simplex and simplicial complex

Def: Let $P = \{p_1, \cdots, p_n\}$ be a (finite) set. An abstract simplicial complex K with vertex set P is a set of subsets of P satisfying the two conditions:

- *(i)* the elements of P belong to K,
- *(ii)* if $\tau \in K$ and $\sigma \subseteq \tau$, then $\sigma \in K$.

The elements of K are the simplices.

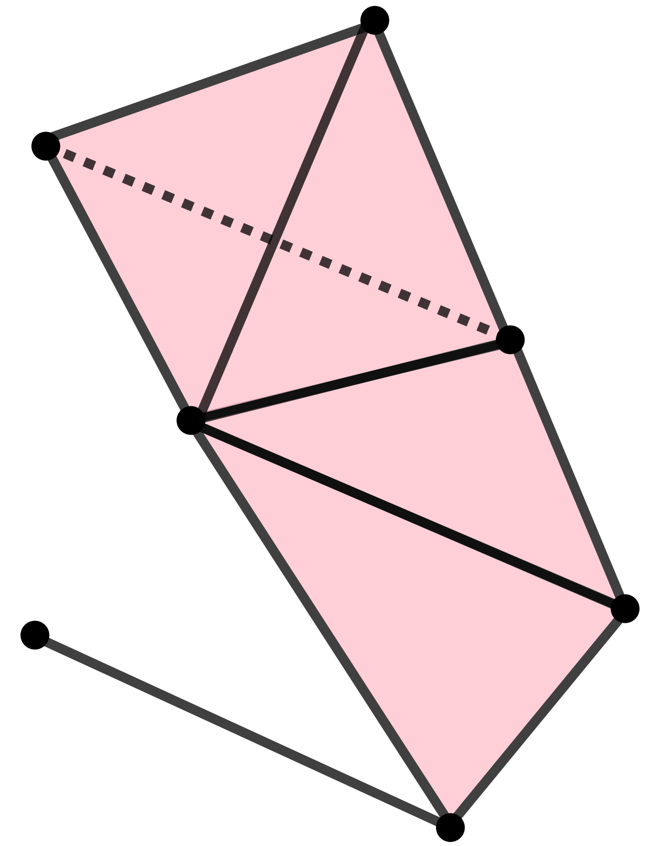


Abstract simplex and simplicial complex

Def: Let $P = \{p_1, \cdots, p_n\}$ be a (finite) set. An abstract simplicial complex K with vertex set P is a set of subsets of P satisfying the two conditions:

- *(i)* the elements of P belong to K,
- *(ii)* if $\tau \in K$ and $\sigma \subseteq \tau$, then $\sigma \in K$.

The elements of K are the simplices.



IMPORTANT

Simplicial complexes can be seen at the same time as geometric/topological spaces (good for topological/geometrical inference) and as combinatorial objects (abstract simplicial complexes, good for computations).

Abstract simplex and simplicial complex

Def: A realization of an abstract simplicial complex K is a geometric simplicial complex K' who is isomorphic to K, i.e., there exists a bijection

$$f : \mathrm{vert}(K) \to \mathrm{vert}(K'),$$

such that $\sigma \in K \iff f(\sigma) \in K'$.

Abstract simplex and simplicial complex

Def: A realization of an abstract simplicial complex K is a geometric simplicial complex K' who is isomorphic to K, i.e., there exists a bijection

$$f : \mathrm{vert}(K) \to \mathrm{vert}(K'),$$

such that $\sigma \in K \iff f(\sigma) \in K'$.

Any abstract simplicial complex with n vertices can be realized in $\mathbb{R}^n$.

Q: Prove it.

Abstract simplex and simplicial complex

Def: A realization of an abstract simplicial complex K is a geometric simplicial complex K' who is isomorphic to K, i.e., there exists a bijection

$$f : \mathrm{vert}(K) \to \mathrm{vert}(K'),$$

such that $\sigma \in K \iff f(\sigma) \in K'$.

Any abstract simplicial complex with n vertices can be realized in $\mathbb{R}^n$.

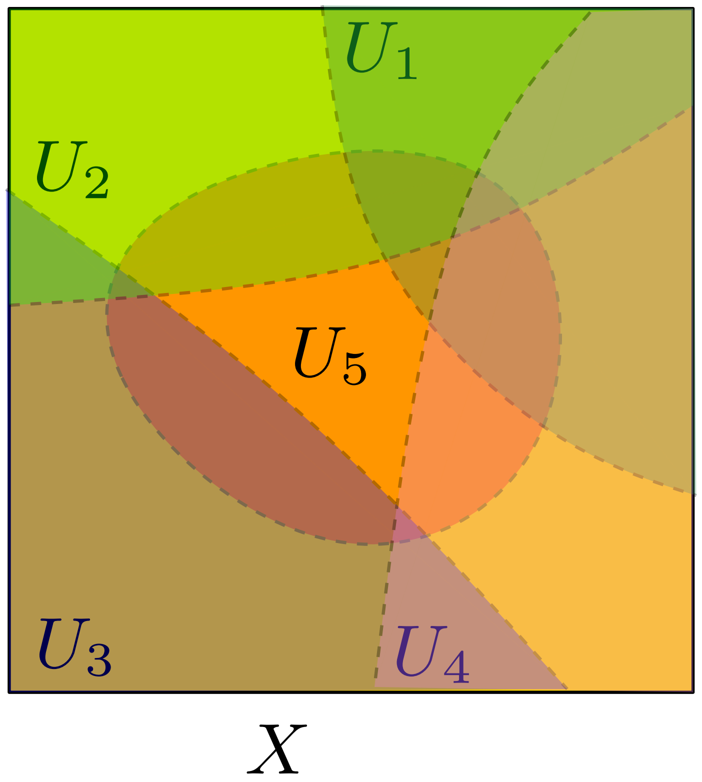
Q: Prove it.

Abstract simplicial complexes and their realizations are *homeomorphic*.

Nerve complex

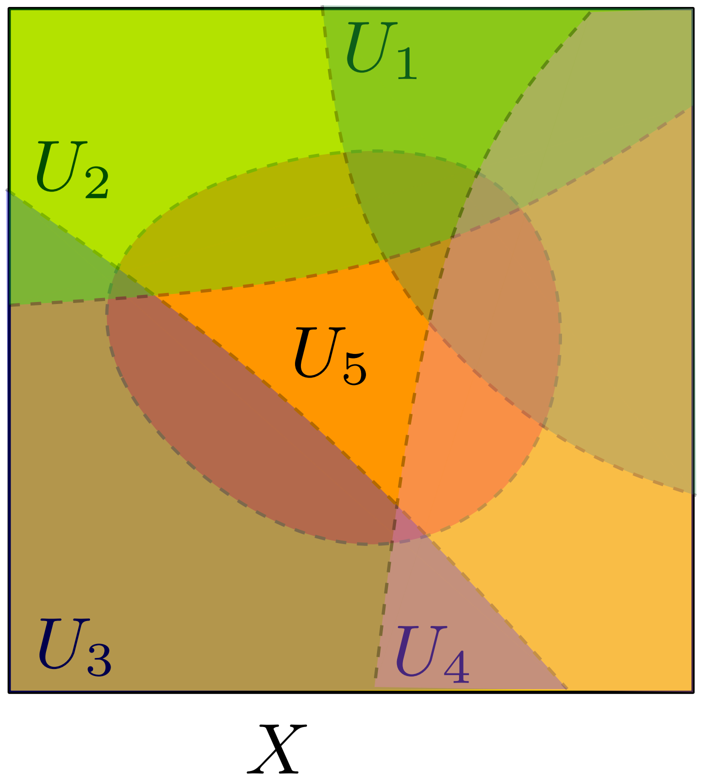
Def: An open cover of a topological space X is a collection $\mathcal{U} = (U_i)_{i \in I}$ of open subsets $U_i \subseteq X$, $i \in I$ where I is a set, such that $X \subseteq \cup_{i \in I} U_i$.

Nerve complex



Def: An open cover of a topological space X is a collection $\mathcal{U} = (U_i)_{i \in I}$ of open subsets $U_i \subseteq X$, $i \in I$ where I is a set, such that $X \subseteq \cup_{i \in I} U_i$.

Nerve complex

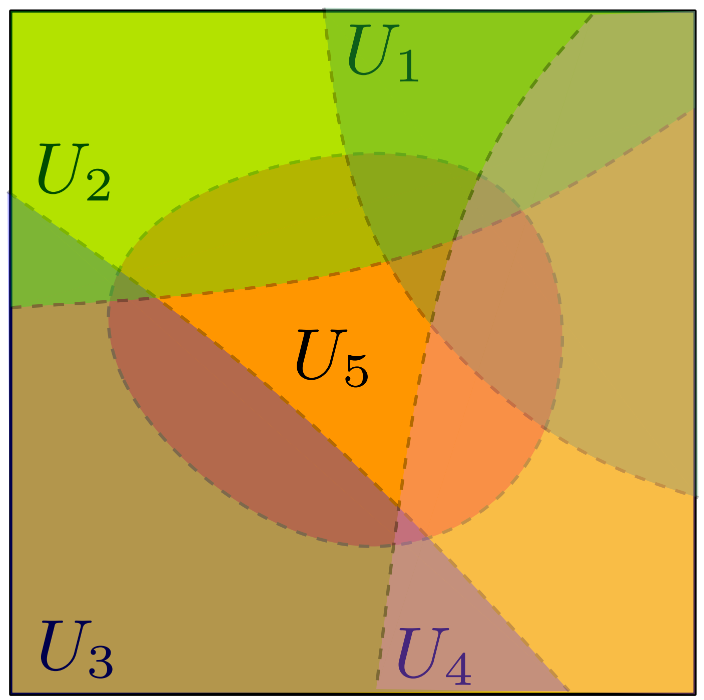


Def: An open cover of a topological space X is a collection $\mathcal{U} = (U_i)_{i \in I}$ of open subsets $U_i \subseteq X$, $i \in I$ where I is a set, such that $X \subseteq \cup_{i \in I} U_i$.

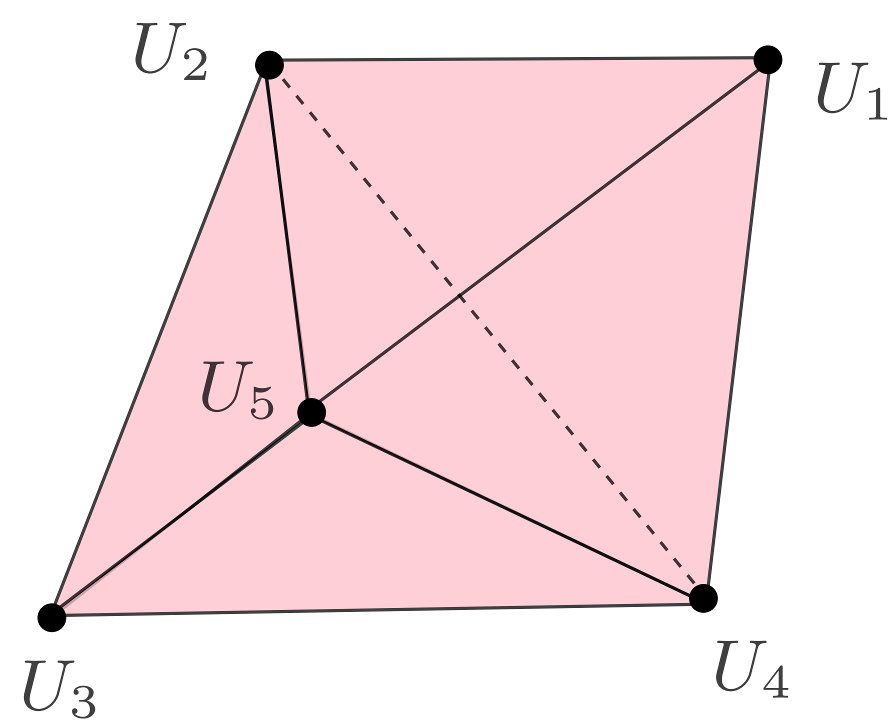
Def: Given a cover of a topological space X, $\mathcal{U} = (U_i)_{i \in I}$, its nerve is the abstract simplicial complex $C(\mathcal{U})$ whose vertex set is $\mathcal{U}$ and s.t.

$$\sigma = [U_{i_0}, U_{i_1}, \ldots, U_{i_k}] \in C(\mathcal{U}) \text{ if and only if } \cap_{j=0}^{k} U_{i_j} \neq \emptyset.$$

Nerve complex



X



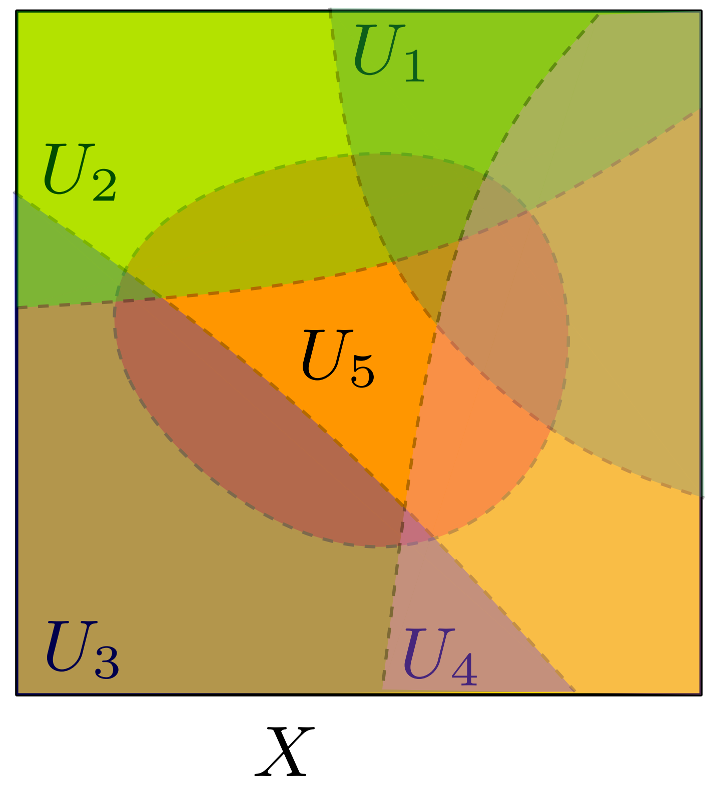
Def: An open cover of a topological space X is a collection $\mathcal{U} = (U_i)_{i \in I}$ of open subsets $U_i \subseteq X$, $i \in I$ where I is a set, such that $X \subseteq \cup_{i \in I} U_i$.

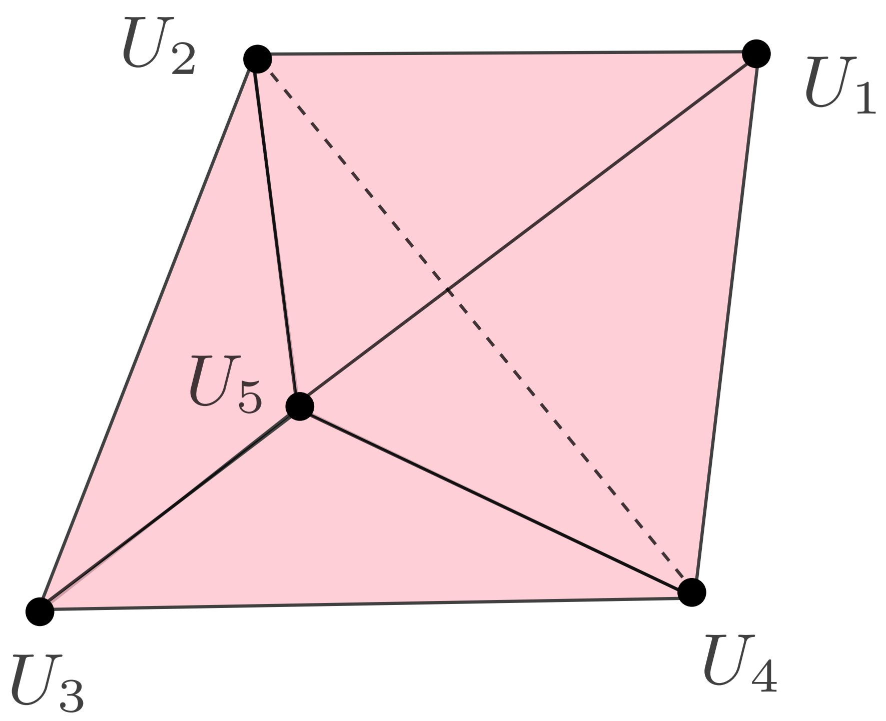
Def: Given a cover of a topological space X, $\mathcal{U} = (U_i)_{i \in I}$, its nerve is the abstract simplicial complex $C(\mathcal{U})$ whose vertex set is $\mathcal{U}$ and s.t.

$$\sigma = [U_{i_0}, U_{i_1}, \ldots, U_{i_k}] \in C(\mathcal{U}) \quad \text{if and only if} \quad \cap_{j=0}^{k} U_{i_j} \neq \emptyset.$$

Nerve complex

[*On the imbedding of systems of compacta in simplicial complexes*, Borsuk, Fund. Math., 1948]





The Nerve Theorem: Let $\mathcal{U} = (U_i)_{i \in I}$ be a finite open cover of a subset X of $\mathbb{R}^d$ such that any intersection of the U_i's is either empty or contractible. Then X and $C(\mathcal{U})$ are homotopy equivalent.

In particular, every convex set is contractible.

Čech and (Vietoris)-Rips complexes

Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Čech complex of radius $r > 0$ is the abstract simplicial complex $C(P, r)$ s.t. $\mathrm{vert}(C(P,r)) = P$ and

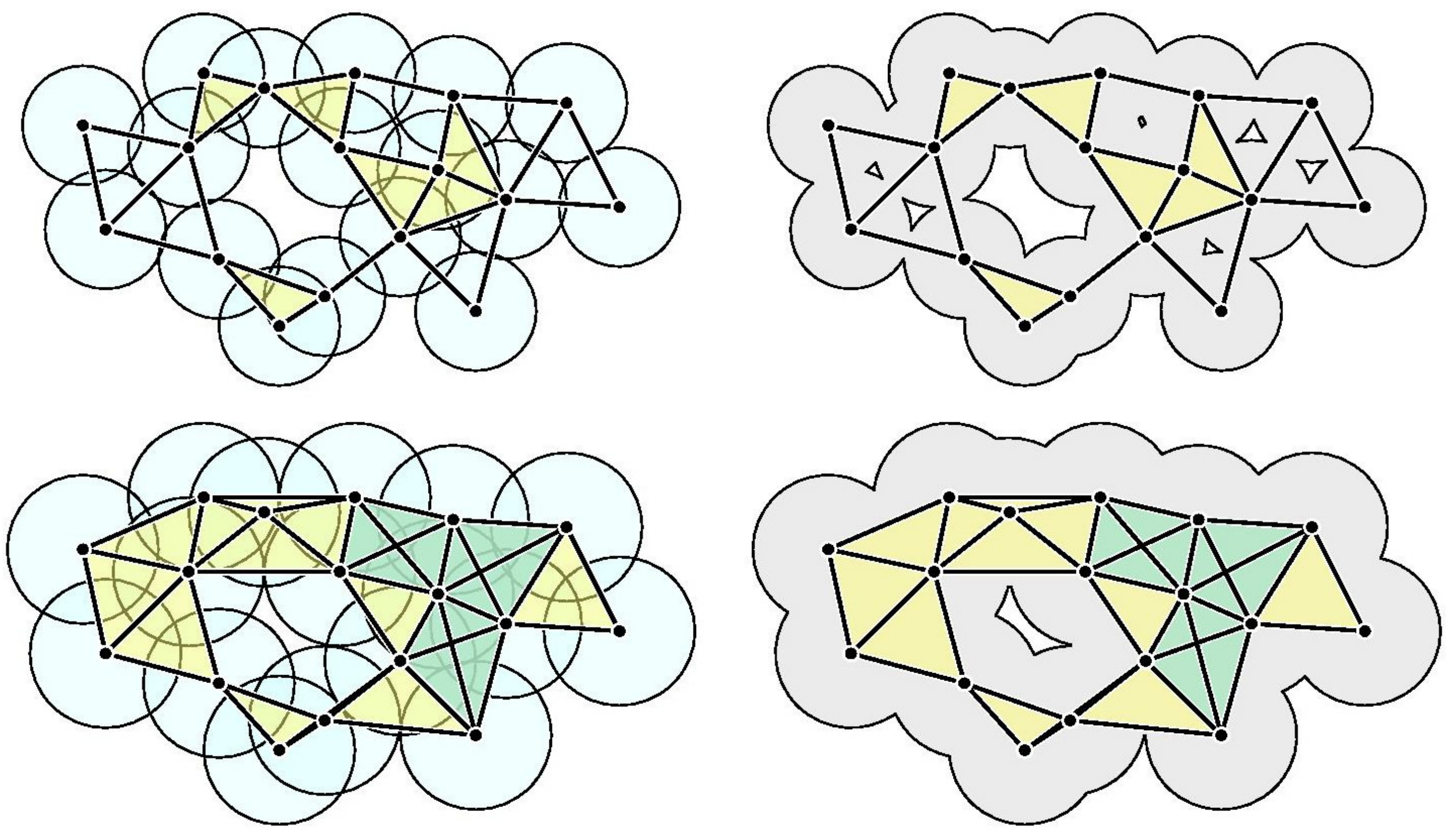
$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in C(P, r) \quad \text{iif} \quad \cap_{j=0}^{k} B(P_{i_j}, r) \neq \emptyset.$$

Čech and (Vietoris)-Rips complexes

Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Čech complex of radius $r > 0$ is the abstract simplicial complex $C(P, r)$ s.t. $\mathrm{vert}(C(P,r)) = P$ and

$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in C(P, r) \quad \text{iif} \quad \cap_{j=0}^{k} B(P_{i_j}, r) \neq \emptyset.$$

Q: Does the Nerve Theorem apply to Čech complexes?



Čech and (Vietoris)-Rips complexes

Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Čech complex of radius $r > 0$ is the abstract simplicial complex $C(P, r)$ s.t. $\mathrm{vert}(C(P, r)) = P$ and

$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in C(P, r) \quad \text{iif} \quad \cap_{j=0}^{k} B(P_{i_j}, r) \neq \emptyset.$$

Pbm: Čech complexes can be quite hard to compute.

Čech and (Vietoris)-Rips complexes

Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Čech complex of radius $r > 0$ is the abstract simplicial complex $C(P, r)$ s.t. $\mathrm{vert}(C(P, r)) = P$ and

$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in C(P, r) \quad \text{iif} \quad \cap_{j=0}^{k} B(P_{i_j}, r) \neq \emptyset.$$

Pbm: Čech complexes can be quite hard to compute.

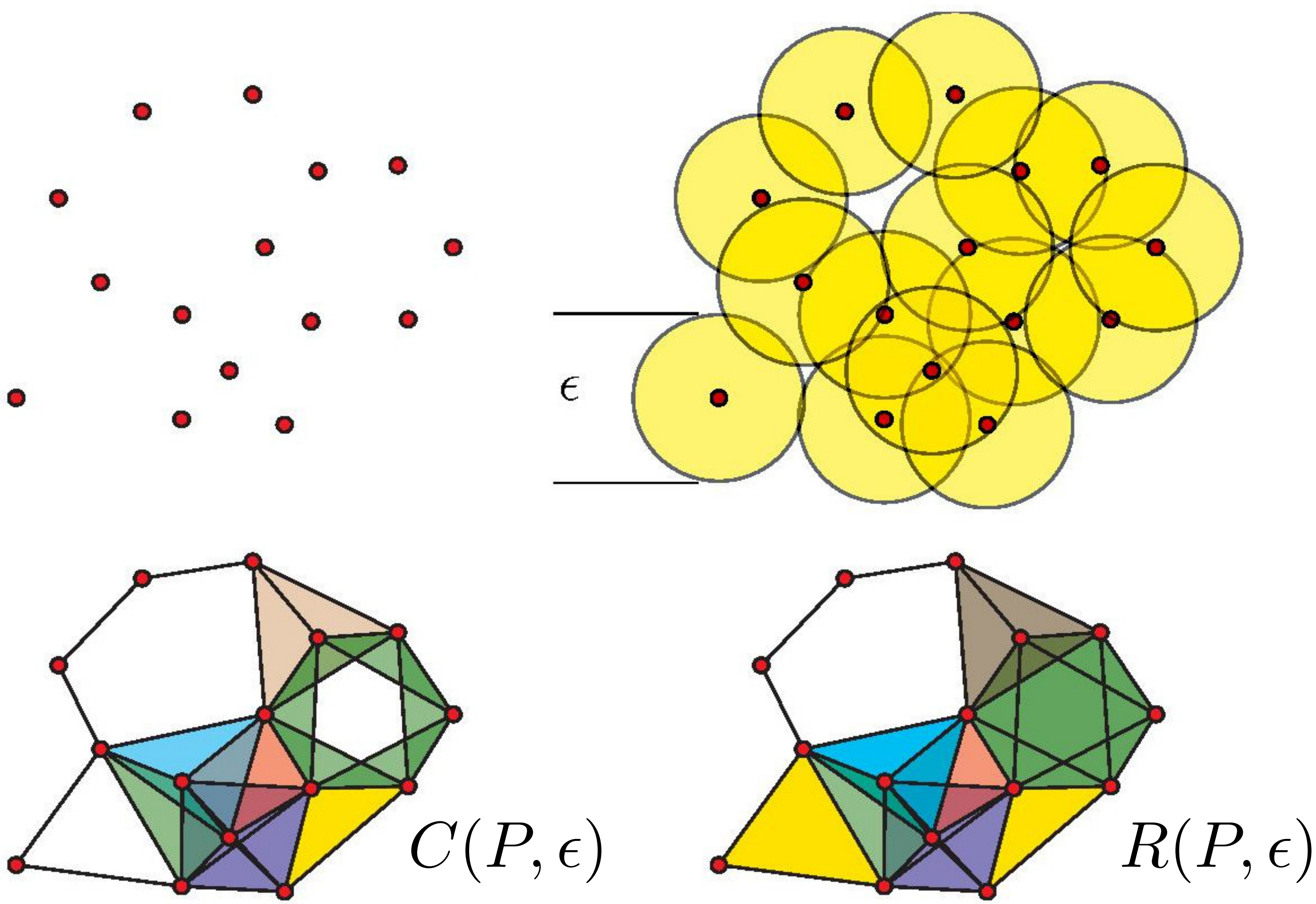
Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Rips complex of radius $r > 0$ is the abstract simplicial complex $R(P, r)$ s.t. $\mathrm{vert}(R(P, r)) = P$ and

$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in R(P, r) \quad \text{iif} \quad \|P_{i_j} - P_{i_{j'}}\| \leq 2r, \forall 1 \leq j, j' \leq k.$$

Čech and (Vietoris)-Rips complexes

Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Čech complex of radius $r > 0$ is the abstract simplicial complex $C(P, r)$ s.t. $\mathrm{vert}(C(P, r)) = P$ and

$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in C(P, r) \quad \text{iif} \quad \cap_{j=0}^{k} B(P_{i_j}, r) \neq \emptyset.$$



Čech and (Vietoris)-Rips complexes

Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Čech complex of radius $r > 0$ is the abstract simplicial complex $C(P, r)$ s.t. $\mathrm{vert}(C(P, r)) = P$ and

$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in C(P, r) \quad \text{iif} \quad \cap_{j=0}^{k} B(P_{i_j}, r) \neq \emptyset.$$

Pbm: Čech complexes can be quite hard to compute.

Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Rips complex of radius $r > 0$ is the abstract simplicial complex $R(P, r)$ s.t. $\mathrm{vert}(R(P, r)) = P$ and

$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in R(P, r) \quad \text{iif} \quad \|P_{i_j} - P_{i_{j'}}\| \leq 2r, \forall 1 \leq j, j' \leq k.$$

Good news is that Rips and Čech complexes are related:

Čech and (Vietoris)-Rips complexes

Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Čech complex of radius $r > 0$ is the abstract simplicial complex $C(P,r)$ s.t. $\mathrm{vert}(C(P,r)) = P$ and

$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in C(P,r) \quad \text{iif} \quad \cap_{j=0}^{k} B(P_{i_j}, r) \neq \emptyset.$$

Pbm: Čech complexes can be quite hard to compute.

Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Rips complex of radius $r > 0$ is the abstract simplicial complex $R(P,r)$ s.t. $\mathrm{vert}(R(P,r)) = P$ and

$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in R(P,r) \quad \text{iif} \quad \|P_{i_j} - P_{i_{j'}}\| \leq 2r, \forall 1 \leq j, j' \leq k.$$

Good news is that Rips and Čech complexes are related:

Prop: $R(P, r/2) \subseteq C(P,r) \subseteq R(P,r)$.

Q: Prove it.

Storing simplicial complexes

[*The Simplex Tree: An Efficient Data Structure for General Simplicial Complexes*, Boissonnat, Maria, Algorithmica, 2014]

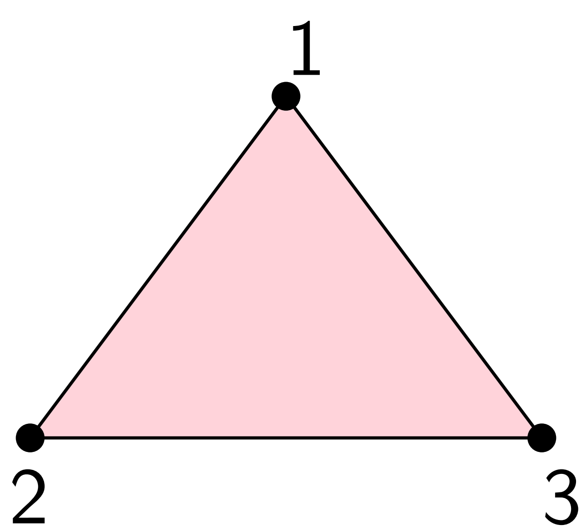
We want to store simplicial complexes with a data structure that allows to perform standard operations (insertion of a simplex, checking if a simplex is present, etc) in a fast and easy way.

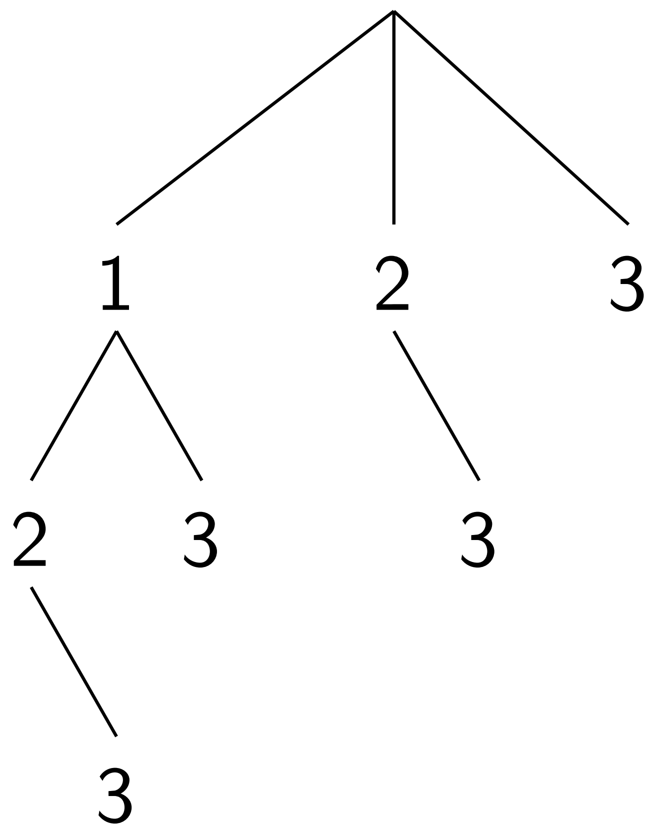
Storing simplicial complexes

[*The Simplex Tree: An Efficient Data Structure for General Simplicial Complexes*, Boissonnat, Maria, Algorithmica, 2014]

We want to store simplicial complexes with a data structure that allows to perform standard operations (insertion of a simplex, checking if a simplex is present, etc) in a fast and easy way.

Idea: store sorted simplices in a prefix tree (also called *trie)*.



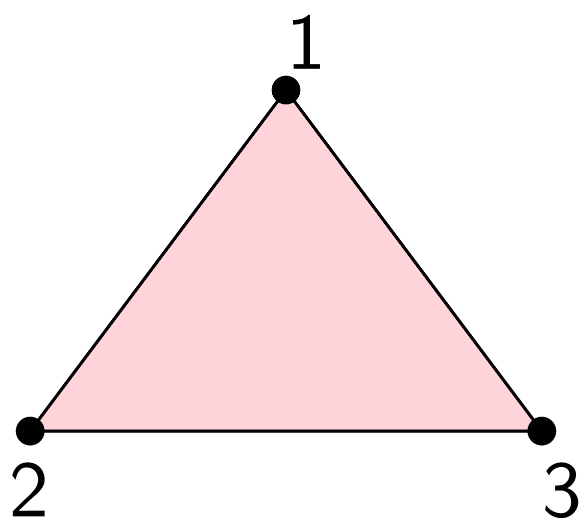


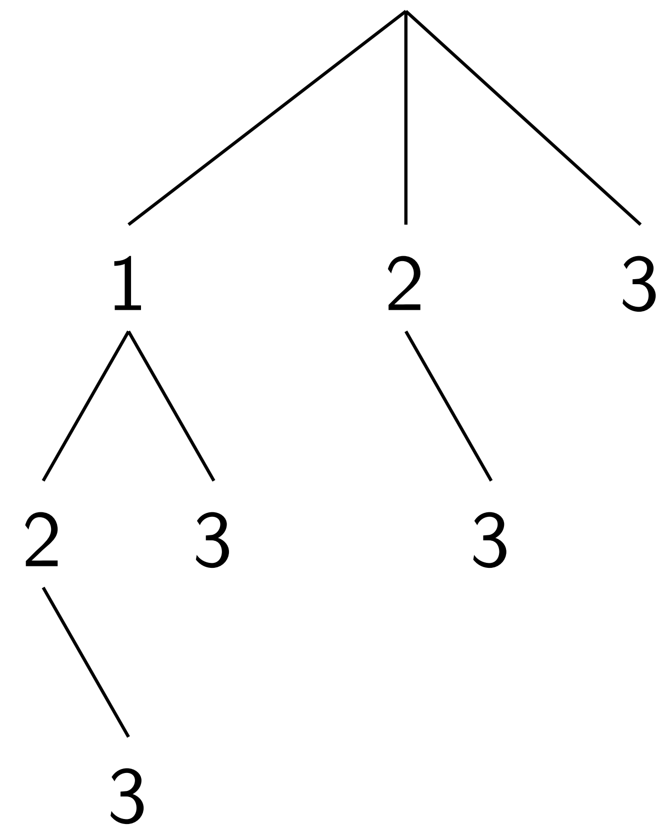
Storing simplicial complexes

[*The Simplex Tree: An Efficient Data Structure for General Simplicial Complexes*, Boissonnat, Maria, Algorithmica, 2014]

We want to store simplicial complexes with a data structure that allows to perform standard operations (insertion of a simplex, checking if a simplex is present, etc) in a fast and easy way.

Idea: store sorted simplices in a prefix tree (also called *trie)*.





This is called the *simplex tree*.

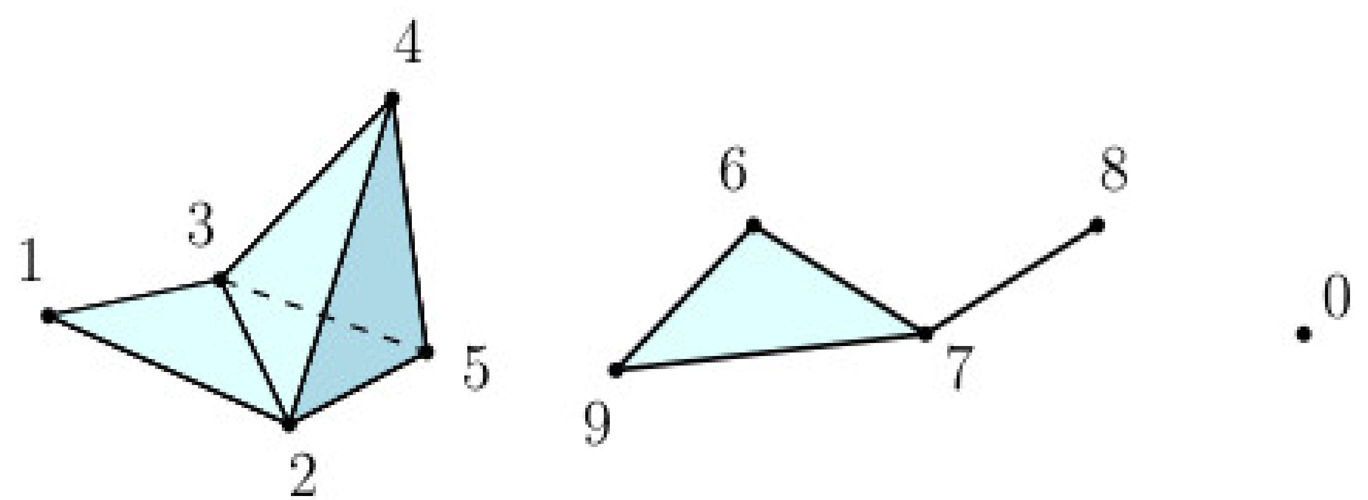
It allows to store all simplices explicitly without storing all adjacency relations, while maintaining low complexity for basic operations.

Storing simplicial complexes

[*The Simplex Tree: An Efficient Data Structure for General Simplicial Complexes*, Boissonnat, Maria, Algorithmica, 2014]

We want to store simplicial complexes with a data structure that allows to perform standard operations (insertion of a simplex, checking if a simplex is present, etc) in a fast and easy way.

Q: build the simplex tree of

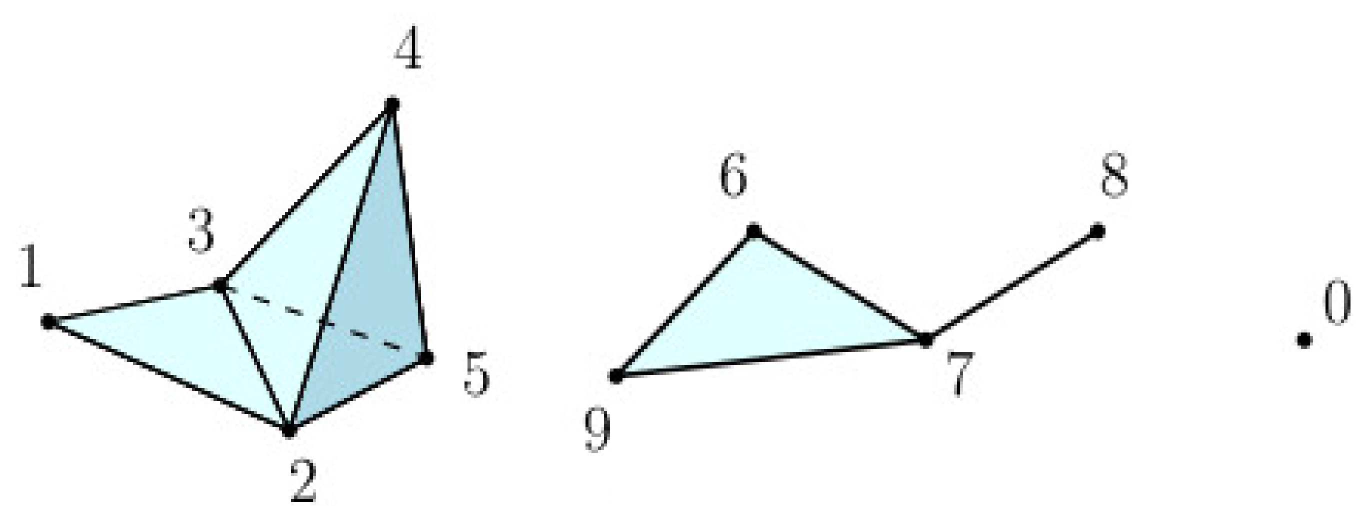


Storing simplicial complexes

[*The Simplex Tree: An Efficient Data Structure for General Simplicial Complexes*, Boissonnat, Maria, Algorithmica, 2014]

We want to store simplicial complexes with a data structure that allows to perform standard operations (insertion of a simplex, checking if a simplex is present, etc) in a fast and easy way.

Q: build the simplex tree of



Number of nodes in simplex tree $=$ number of simplices

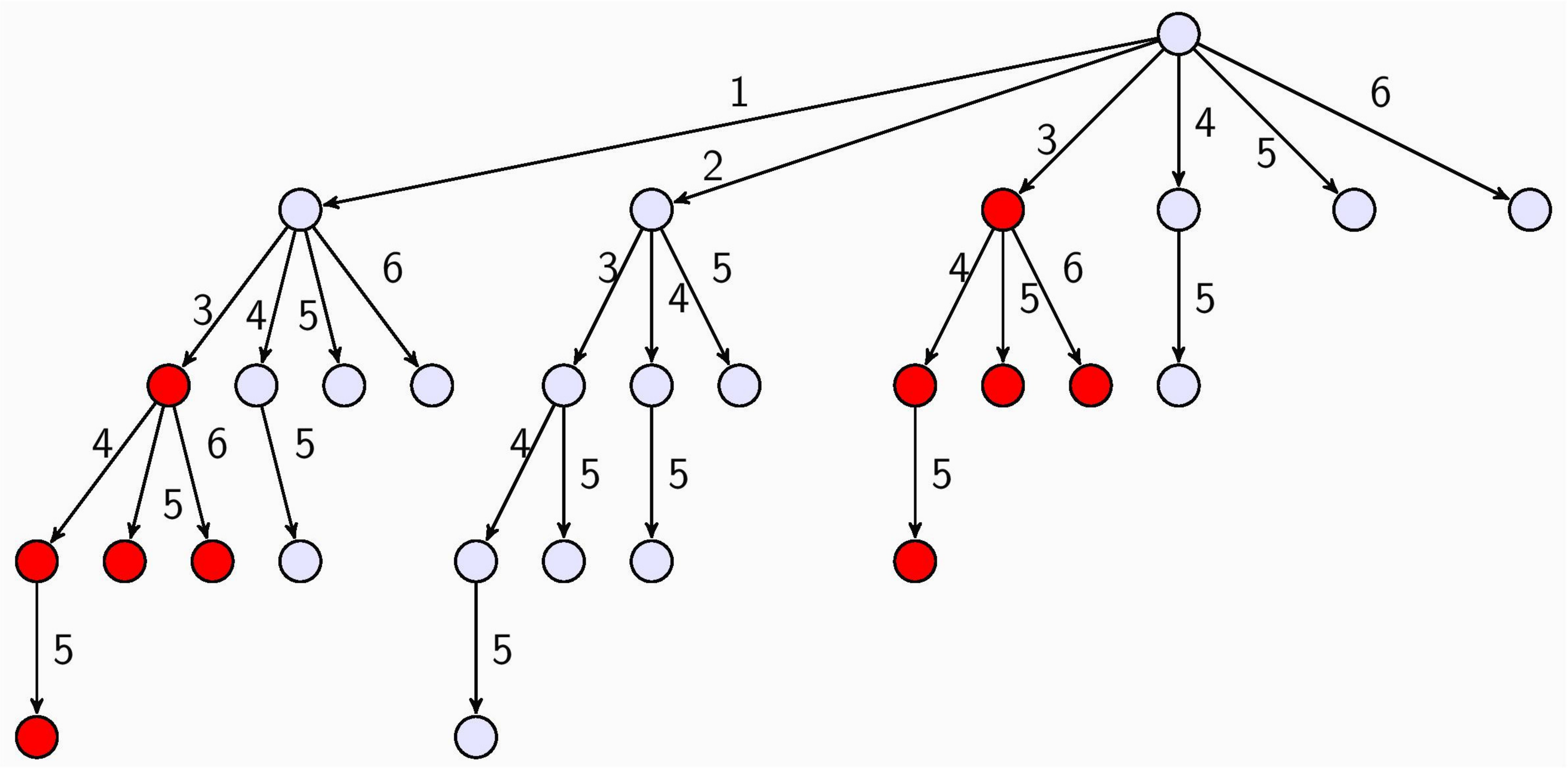
Depth of simplex tree $= 1 +$ dimension of complex

Storing simplicial complexes

[*The Simplex Tree: An Efficient Data Structure for General Simplicial Complexes*, Boissonnat, Maria, Algorithmica, 2014]

We want to store simplicial complexes with a data structure that allows to perform standard operations (insertion of a simplex, checking if a simplex is present, etc) in a fast and easy way.

Unfortunately, the simplex tree also has redundancies.

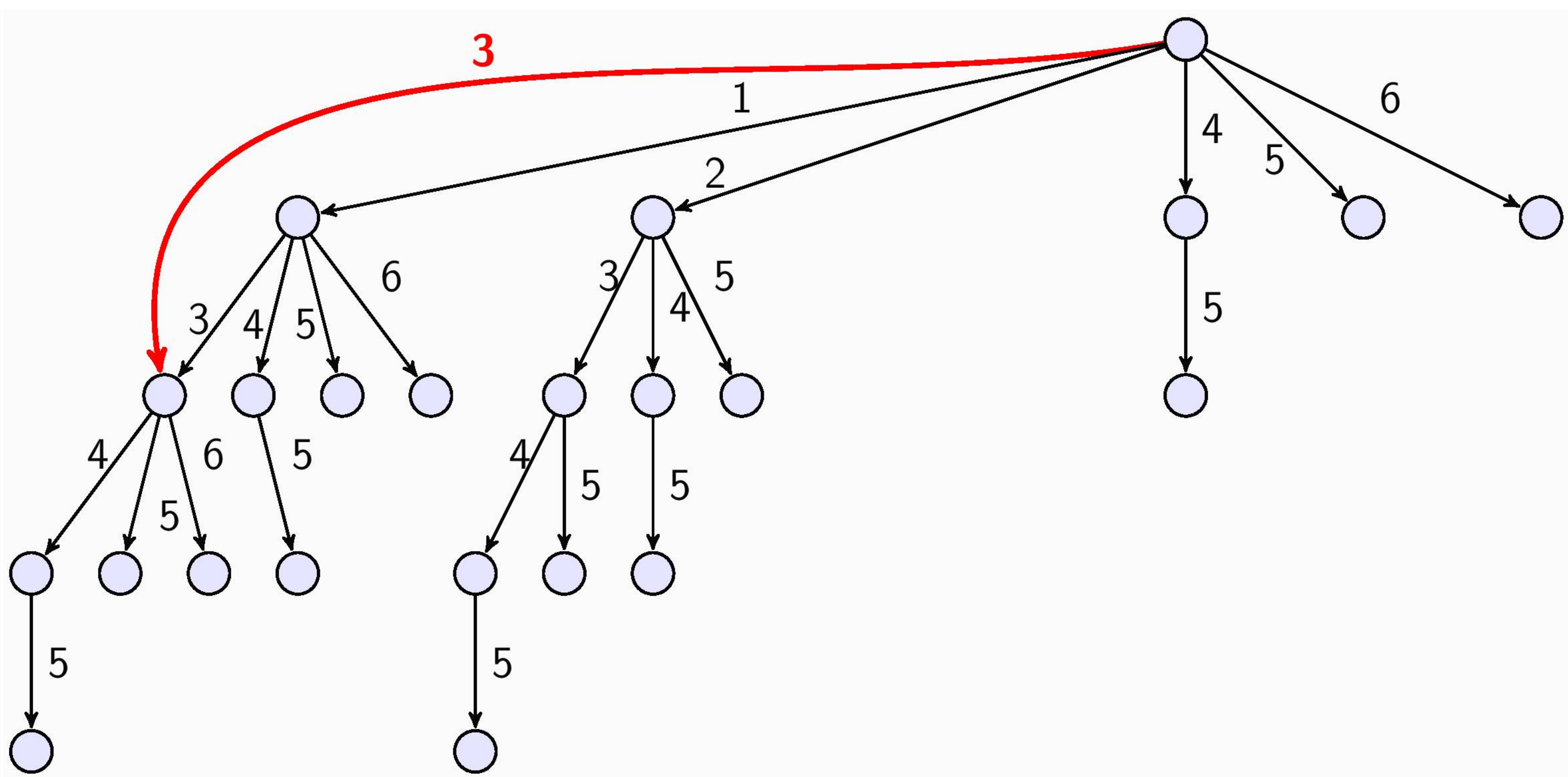


Storing simplicial complexes

[*The Simplex Tree: An Efficient Data Structure for General Simplicial Complexes*, Boissonnat, Maria, Algorithmica, 2014]

We want to store simplicial complexes with a data structure that allows to perform standard operations (insertion of a simplex, checking if a simplex is present, etc) in a fast and easy way.

Unfortunately, the simplex tree also has redundancies.



An invariant fit for computation

Pb 2: Looking for homotopy equivalences/homeomorphisms/isotopies is extremely difficult. Are there mathematical quantities that are invariant to homotopy equivalences **and** easy to compute?

An invariant fit for computation

Pb 2: Looking for homotopy equivalences/homeomorphisms/isotopies is extremely difficult. Are there mathematical quantities that are invariant to homotopy equivalences **and** easy to compute?

A: The *holes*, encoded in the *homology groups* H_k, $k \in \mathbb{N}$

The homology groups

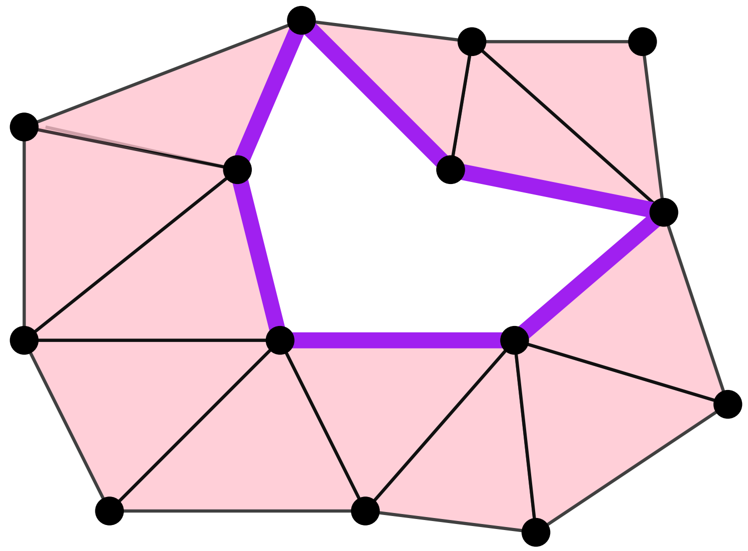
The homology groups

Q: How to characterize a hole in a simplicial complex?

The homology groups

Q: How to characterize a hole in a simplicial complex?

A: A hole (in 1D) is a path whose first and end points are the same, a loop.

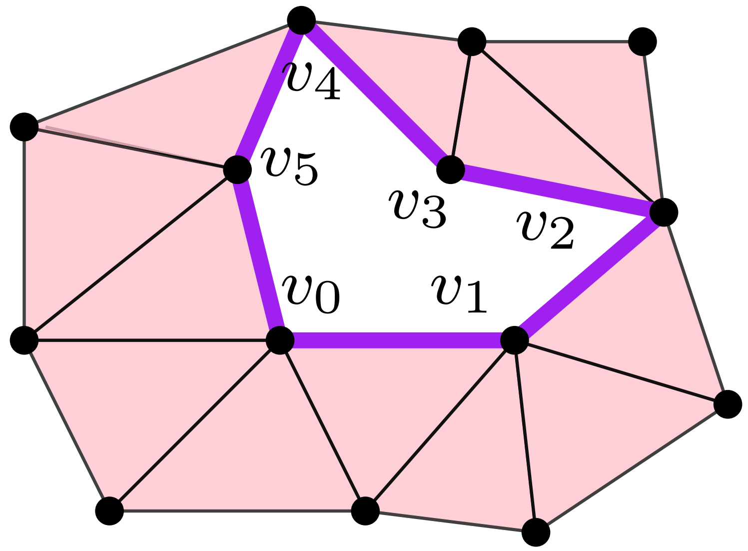


The homology groups

Q: How to characterize a hole in a simplicial complex?

A: A hole (in 1D) is a path whose first and end points are the same, a loop.

The sequence of 1-dimensional simplices $[v_0, v_1]$, $[v_1, v_2]$, $[v_2, v_3]$, $[v_3, v_4]$, $[v_4, v_5]$, $[v_5, v_0]$ is a hole

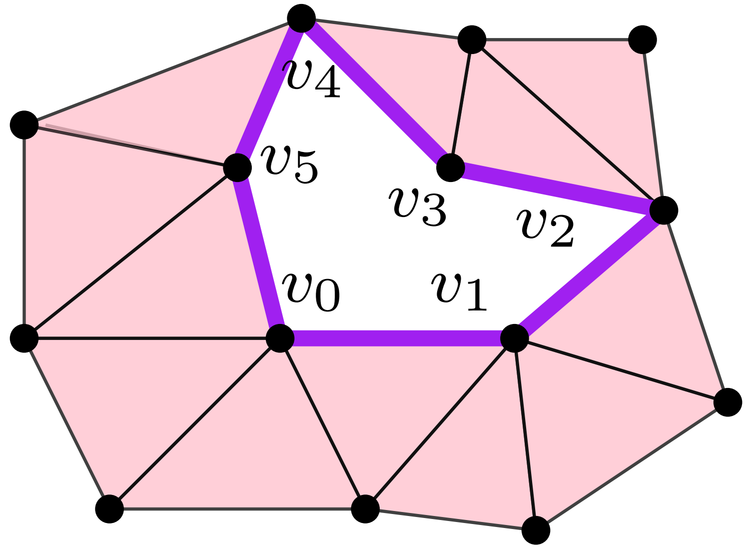


The homology groups

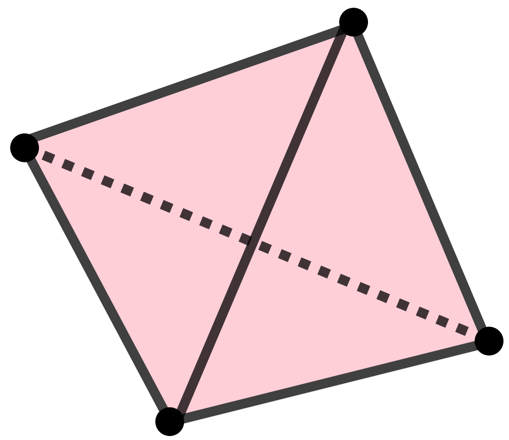
Q: How to characterize a hole in a simplicial complex?

A: A hole (in 1D) is a path whose first and end points are the same, a loop.

The sequence of 1-dimensional simplices $[v_0, v_1]$, $[v_1, v_2]$, $[v_2, v_3]$, $[v_3, v_4]$, $[v_4, v_5]$, $[v_5, v_0]$ is a hole



But what about higher dimensional holes (like the inside of a tetrahedron)?

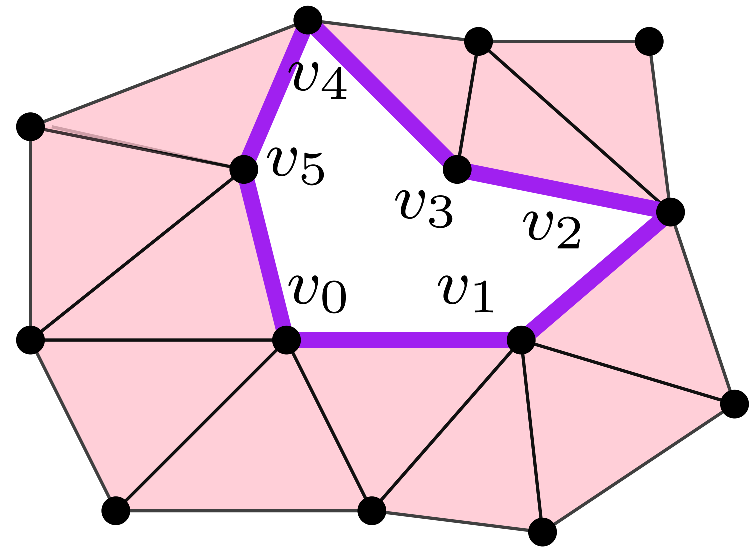


The homology groups

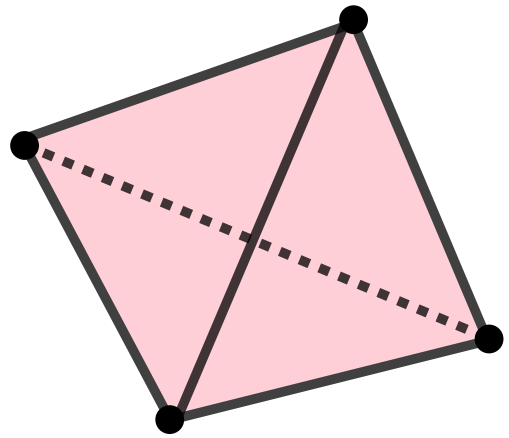
Q: How to characterize a hole in a simplicial complex?

A: A hole (in 1D) is a path whose first and end points are the same, a loop.

The sequence of 1-dimensional simplices $[v_0, v_1]$, $[v_1, v_2]$, $[v_2, v_3]$, $[v_3, v_4]$, $[v_4, v_5]$, $[v_5, v_0]$ is a hole



But what about higher dimensional holes (like the inside of a tetrahedron)?



A: A hole in dimension d is a simplicial complex in which each $(d-1)$-simplex appears an even number of times.

The homology groups

Def: A d-*chain* is a formal sum of d-simplices with coefficients in $\mathbb{Z}/2\mathbb{Z}$.

$$C = [v_0, v_1] + [v_1, v_2] + [v_2, v_3] + [v_3, v_4] + [v_4, v_5] + [v_5, v_0].$$

The homology groups

Def: A d-*chain* is a formal sum of d-simplices with coefficients in $\mathbb{Z}/2\mathbb{Z}$.

$$C = [v_0, v_1] + [v_1, v_2] + [v_2, v_3] + [v_3, v_4] + [v_4, v_5] + [v_5, v_0].$$

Def: The *boundary* of a d-simplex is the chain made of its $(d-1)$-simplices.

The homology groups

Def: A d-*chain* is a formal sum of d-simplices with coefficients in $\mathbb{Z}/2\mathbb{Z}$.

$$C = [v_0, v_1] + [v_1, v_2] + [v_2, v_3] + [v_3, v_4] + [v_4, v_5] + [v_5, v_0].$$

Def: The *boundary* of a d-simplex is the chain made of its $(d-1)$-simplices.

$$\partial_n [v_1, \ldots, v_{n+1}] = \sum_{i=1}^{n+1} [v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{n+1}]$$

The homology groups

Def: A d-*chain* is a formal sum of d-simplices with coefficients in $\mathbb{Z}/2\mathbb{Z}$.

$$C = [v_0, v_1] + [v_1, v_2] + [v_2, v_3] + [v_3, v_4] + [v_4, v_5] + [v_5, v_0].$$

Def: The *boundary* of a d-simplex is the chain made of its $(d-1)$-simplices.

$$\partial_n[v_1, \ldots, v_{n+1}] = \sum_{i=1}^{n+1}[v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{n+1}]$$

$$\partial_1 C = \partial_1[v_0, v_1] + \partial_1[v_1, v_2] + \partial_1[v_2, v_3] + \partial_1[v_3, v_4] + \partial_1[v_4, v_5] + \partial_1[v_5, v_0]$$

The homology groups

Def: A d-*chain* is a formal sum of d-simplices with coefficients in $\mathbb{Z}/2\mathbb{Z}$.

$$C = [v_0, v_1] + [v_1, v_2] + [v_2, v_3] + [v_3, v_4] + [v_4, v_5] + [v_5, v_0].$$

Def: The *boundary* of a d-simplex is the chain made of its $(d-1)$-simplices.

$$\partial_n [v_1, \ldots, v_{n+1}] = \sum_{i=1}^{n+1} [v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{n+1}]$$

$$\partial_1 C = \partial_1 [v_0, v_1] + \partial_1 [v_1, v_2] + \partial_1 [v_2, v_3] + \partial_1 [v_3, v_4] + \partial_1 [v_4, v_5] + \partial_1 [v_5, v_0]$$
$$= [v_0] + [v_1] + [v_1] + [v_2] + [v_2] + [v_3] + [v_3] + [v_4] + [v_4] + [v_5] + [v_5] + [v_0]$$

The homology groups

Def: A d-*chain* is a formal sum of d-simplices with coefficients in $\mathbb{Z}/2\mathbb{Z}$.

$$C = [v_0, v_1] + [v_1, v_2] + [v_2, v_3] + [v_3, v_4] + [v_4, v_5] + [v_5, v_0].$$

Def: The *boundary* of a d-simplex is the chain made of its $(d-1)$-simplices.

$$\partial_n[v_1, \ldots, v_{n+1}] = \sum_{i=1}^{n+1} [v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{n+1}]$$

$$\partial_1 C = \partial_1[v_0, v_1] + \partial_1[v_1, v_2] + \partial_1[v_2, v_3] + \partial_1[v_3, v_4] + \partial_1[v_4, v_5] + \partial_1[v_5, v_0]$$
$$= [v_0] + \cancel{[v_1] + [v_1]} + \cancel{[v_2] + [v_2]} + \cancel{[v_3] + [v_3]} + \cancel{[v_4] + [v_4]} + \cancel{[v_5] + [v_5]} + [v_0]$$
$$= [v_0] + [v_0] = 0.$$

Def: A d-*cycle* is a d-chain C s.t. $\partial C = 0$.

The homology groups

Def: A d-*chain* is a formal sum of d-simplices with coefficients in $\mathbb{Z}/2\mathbb{Z}$.

$$C = [v_0, v_1] + [v_1, v_2] + [v_2, v_3] + [v_3, v_4] + [v_4, v_5] + [v_5, v_0].$$

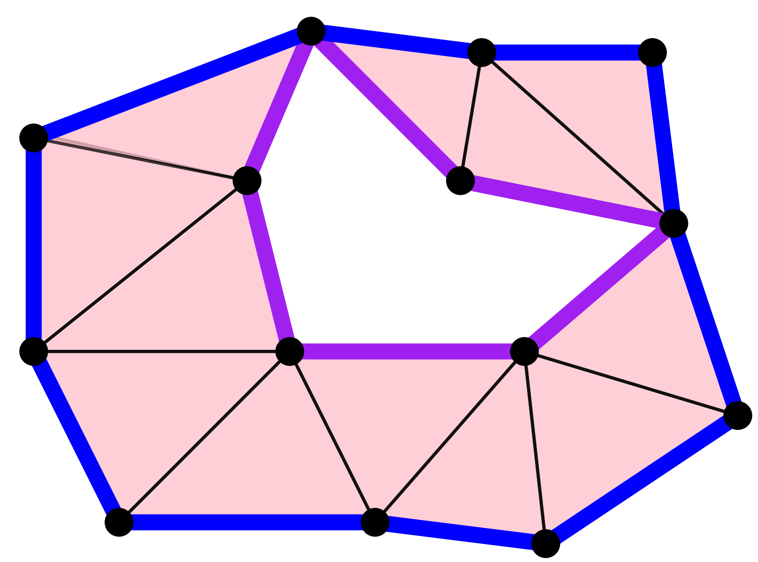
Def: The *boundary* of a d-simplex is the chain made of its $(d-1)$-simplices.

$$\partial_n [v_1, \ldots, v_{n+1}] = \sum_{i=1}^{n+1} [v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{n+1}]$$

$$\partial_1 C = \partial_1 [v_0, v_1] + \partial_1 [v_1, v_2] + \partial_1 [v_2, v_3] + \partial_1 [v_3, v_4] + \partial_1 [v_4, v_5] + \partial_1 [v_5, v_0]$$
$$= [v_0] + \cancel{[v_1] + [v_1]} + \cancel{[v_2] + [v_2]} + \cancel{[v_3] + [v_3]} + \cancel{[v_4] + [v_4]} + \cancel{[v_5] + [v_5]} + [v_0]$$
$$= [v_0] + [v_0] = 0.$$

Def: A d-*cycle* is a d-chain C s.t. $\partial C = 0$.

Pb: Cycles are not holes!!



The homology groups

Lemma: $\partial_{n-1} \circ \partial_n = 0.$

Q: Prove it.

The homology groups

Lemma: $\partial_{n-1} \circ \partial_n = 0.$

Q: Prove it.

Def: Two cycles are the same (homologous) if 'their difference is in $\mathrm{im}(\partial)$':

$$C \sim C' \Longleftrightarrow C + C' \in \mathrm{im}(\partial)$$

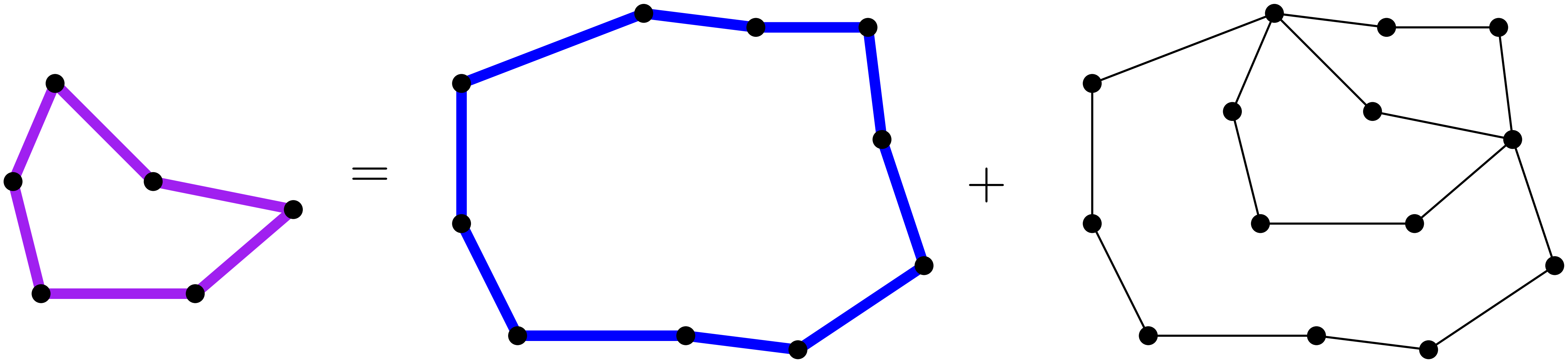
The homology groups

Lemma: $\partial_{n-1} \circ \partial_n = 0.$

Q: Prove it.

Def: Two cycles are the same (homologous) if 'their difference is in $\mathrm{im}(\partial)$':

$$C \sim C' \Longleftrightarrow C + C' \in \mathrm{im}(\partial)$$



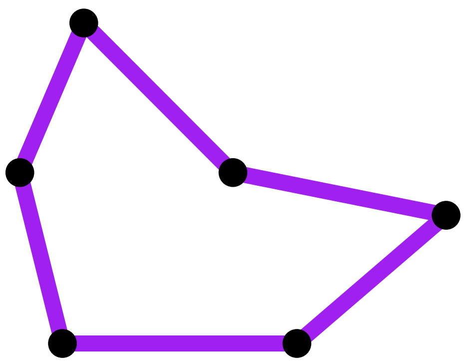
The homology groups

Lemma: $\partial_{n-1} \circ \partial_n = 0.$

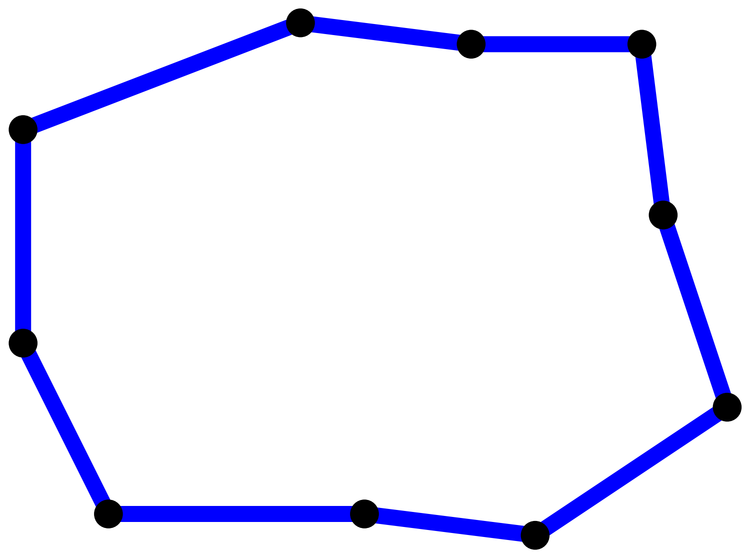
Q: Prove it.

Def: Two cycles are the same (homologous) if 'their difference is in $\mathrm{im}(\partial)$':

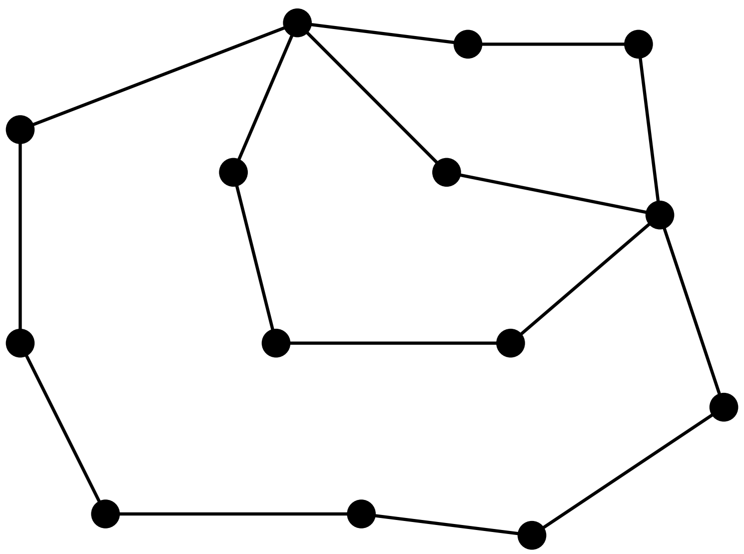
$$C \sim C' \iff C + C' \in \mathrm{im}(\partial)$$



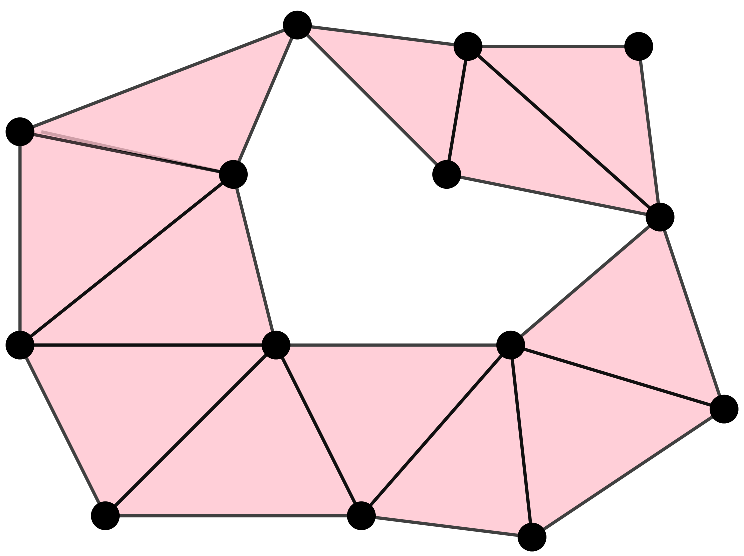
$=$



$+$



$= \partial($



$)$

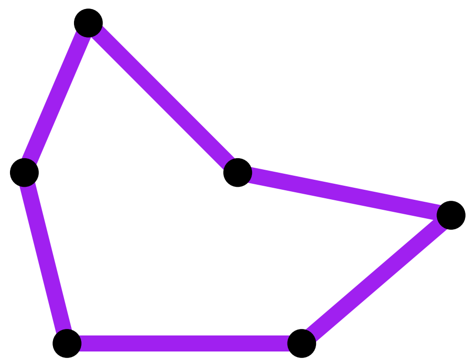
The homology groups

Lemma: $\partial_{n-1} \circ \partial_n = 0.$

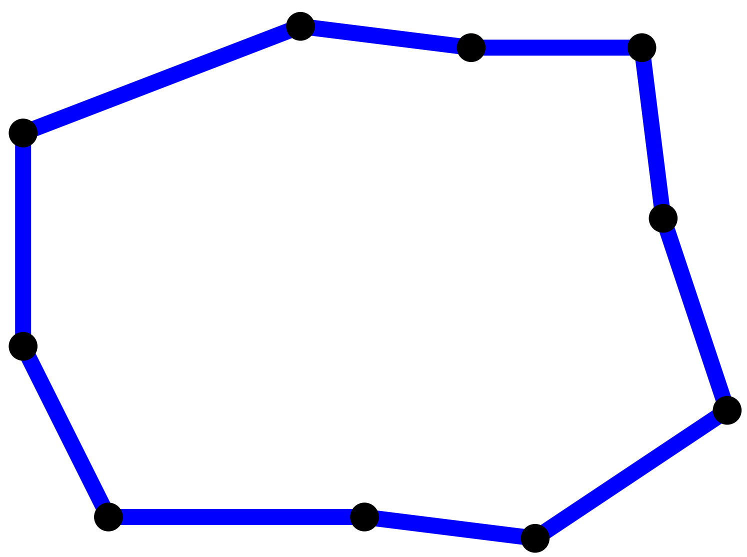
Q: Prove it.

Def: Two cycles are the same (homologous) if 'their difference is in $\mathrm{im}(\partial)$':

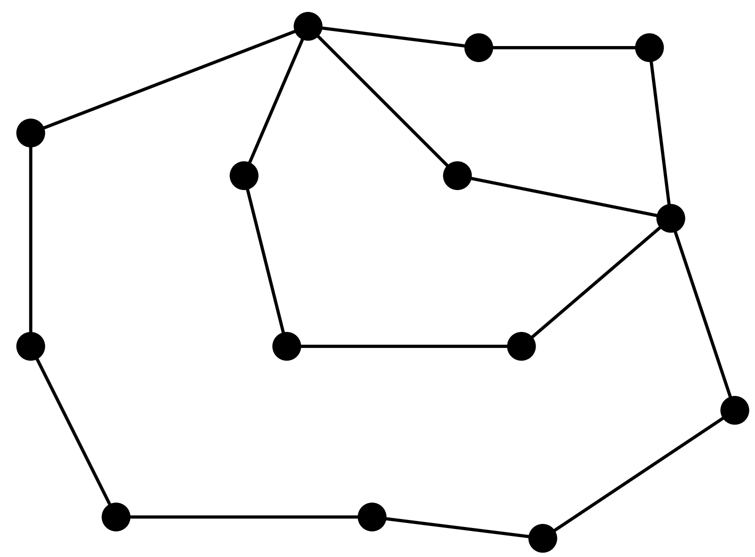
$$C \sim C' \iff C + C' \in \mathrm{im}(\partial)$$



$=$

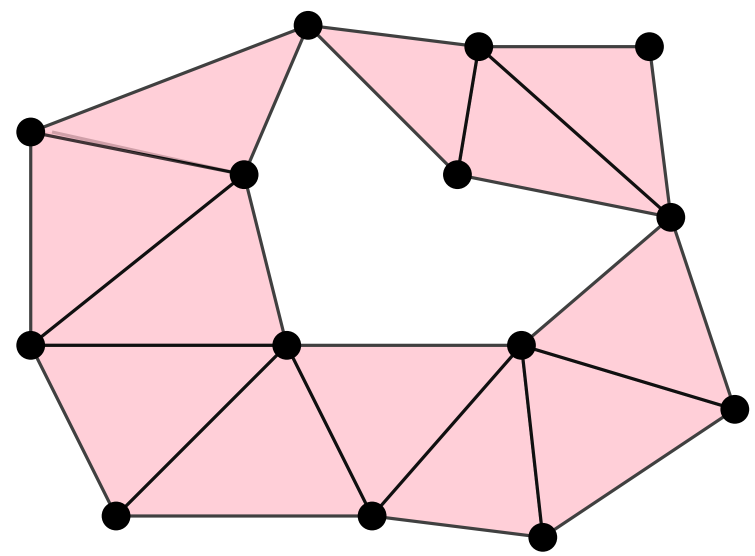


$+$



$$H_k = Z_k / B_k$$

$= \partial($



$)$

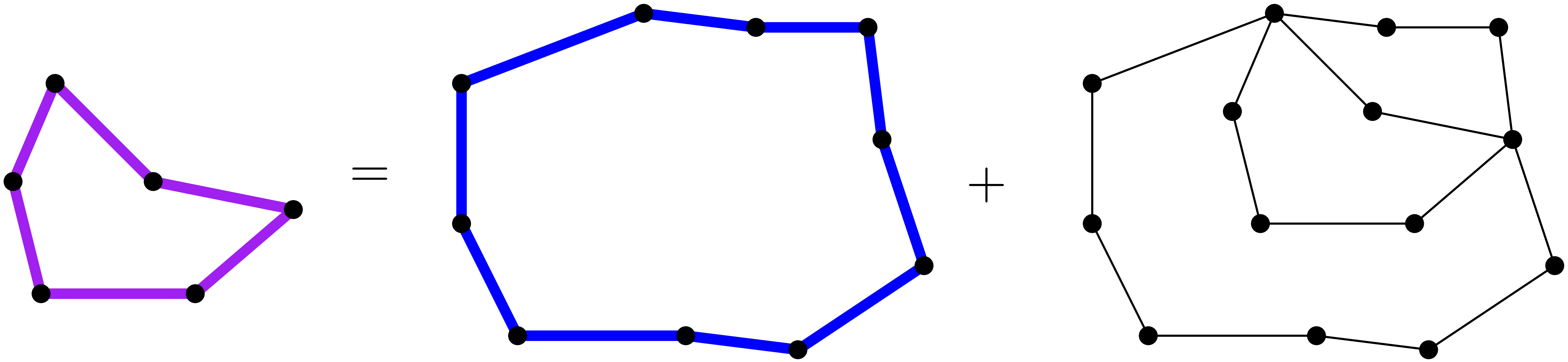
The homology groups

Lemma: $\partial_{n-1} \circ \partial_n = 0.$

Q: Prove it.

Def: Two cycles are the same (homologous) if 'their difference is in $\mathrm{im}(\partial)$':

$$C \sim C' \iff C + C' \in \mathrm{im}(\partial)$$

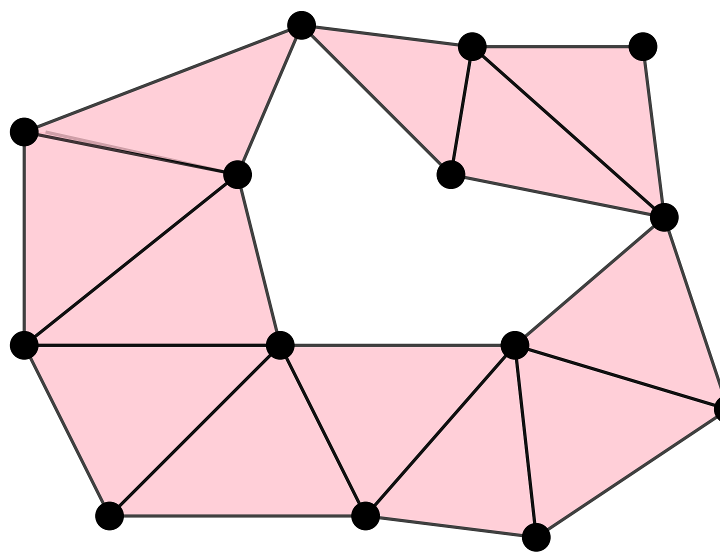


group of k-cycles

$$H_k = Z_k / B_k$$

$\mathrm{im}(\partial_{k+1})$

group of 'cycles minus boundaries'

$= \partial($  $)$

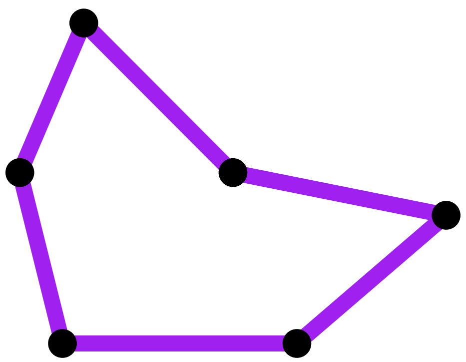
The homology groups

Lemma: $\partial_{n-1} \circ \partial_n = 0.$

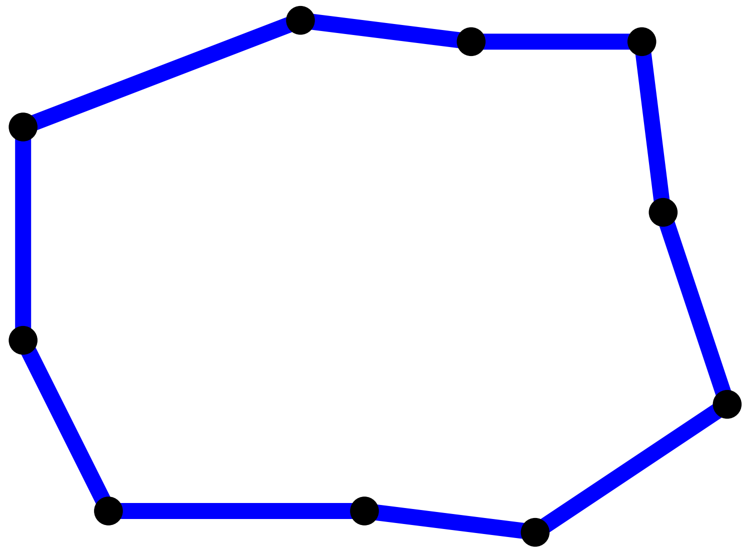
Q: Prove it.

Def: Two cycles are the same (homologous) if 'their difference is in $\mathrm{im}(\partial)$':

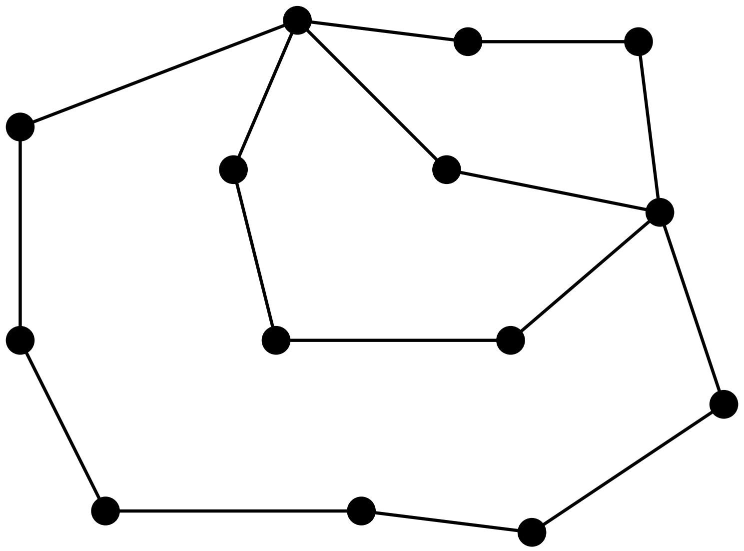
$$C \sim C' \iff C + C' \in \mathrm{im}(\partial)$$



$=$



$+$

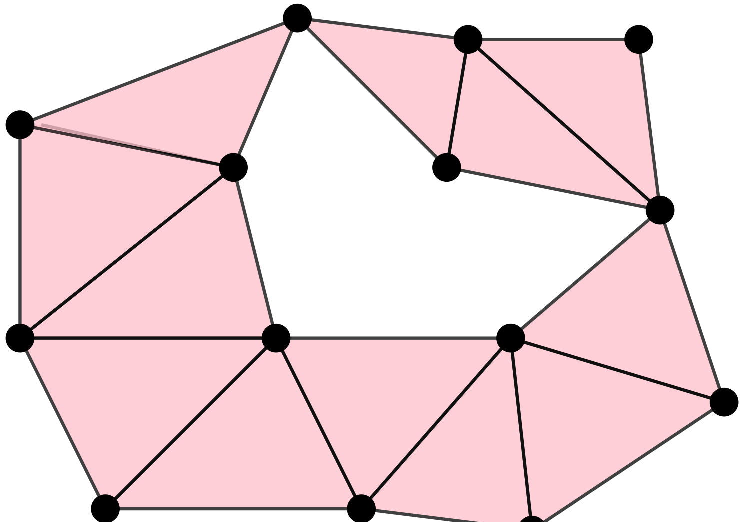


$$H_k = \{[C] \,:\, C \in Z_k\}$$

where

$$[C] = \{C' \,:\, C \sim C'\}$$

$= \partial($



$)$

The homology groups

H_k is a group (vector space) in which each element is an equivalence class of cycles associated to the same hole.

Def: The dimension of H_k is called the *Betti number* β_k.

Minimum number of (classes of) cyles needed to create a basis, i.e., to be able to write *any* cycle as a linear combination of cycles in the basis.

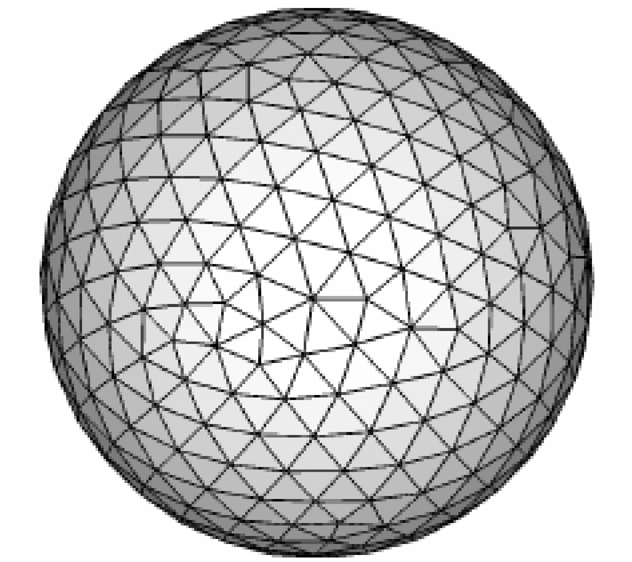
β_0 counts the connected components, β_1 counts the loops, β_2 counts the cavities, and so on...

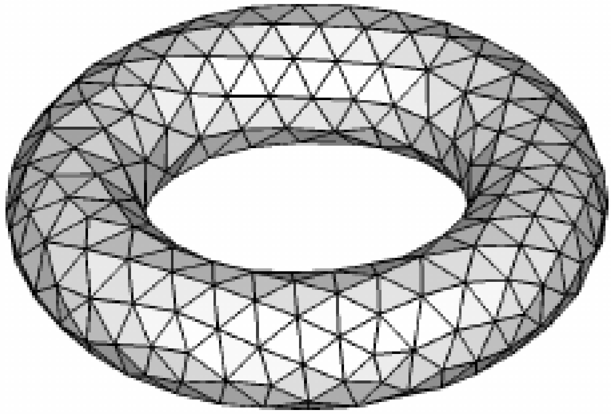
The homology groups

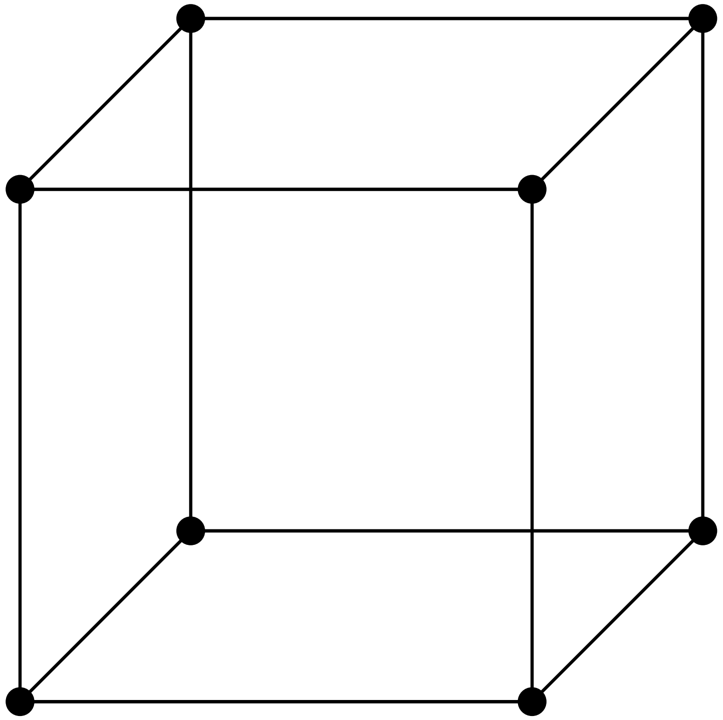
H_k is a group (vector space) in which each element is an equivalence class of cycles associated to the same hole.

Def: The dimension of H_k is called the *Betti number* β_k.

Q: What are the Betti numbers of:





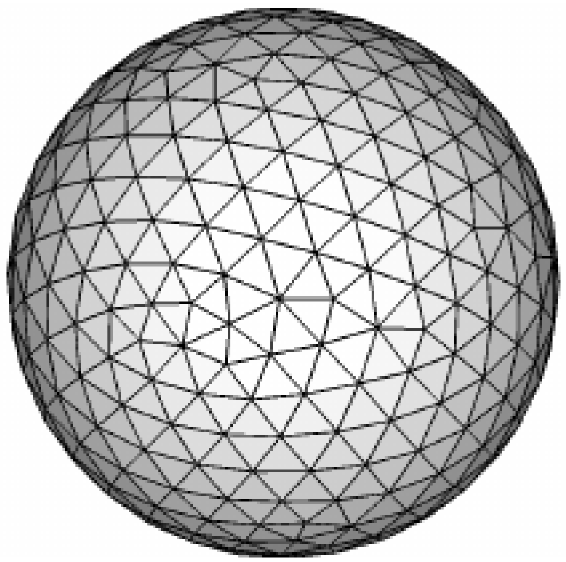


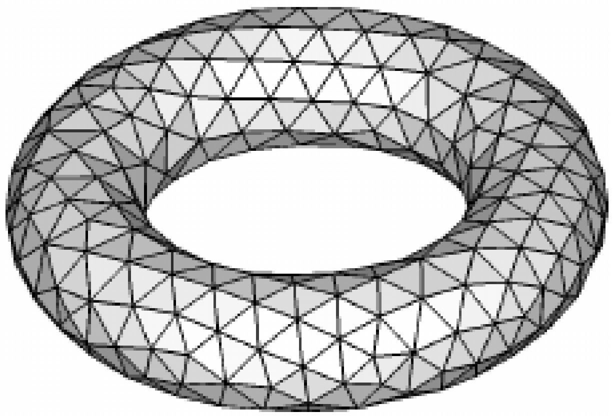
The homology groups

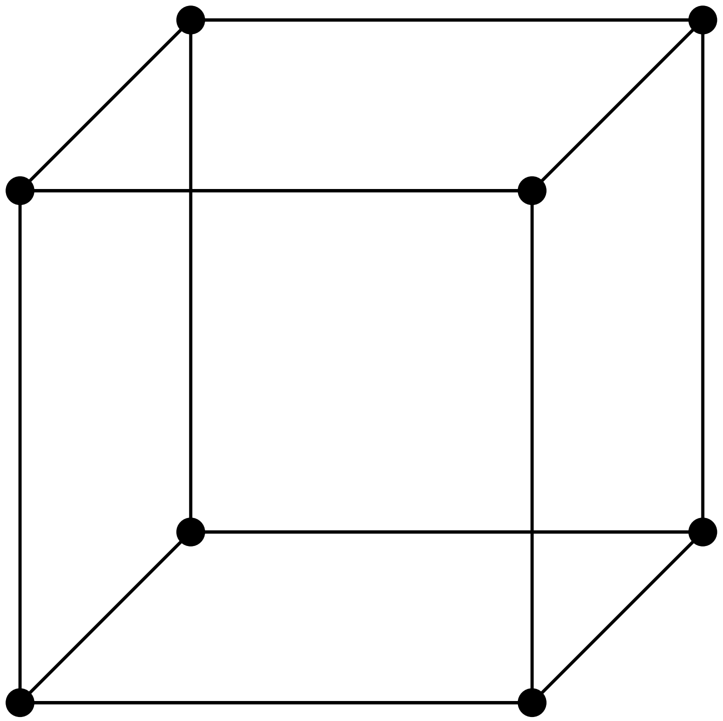
H_k is a group (vector space) in which each element is an equivalence class of cycles associated to the same hole.

Def: The dimension of H_k is called the *Betti number* β_k.

Q: What are the Betti numbers of:







The whole point of homology groups and Betti numbers is that they satisfy:

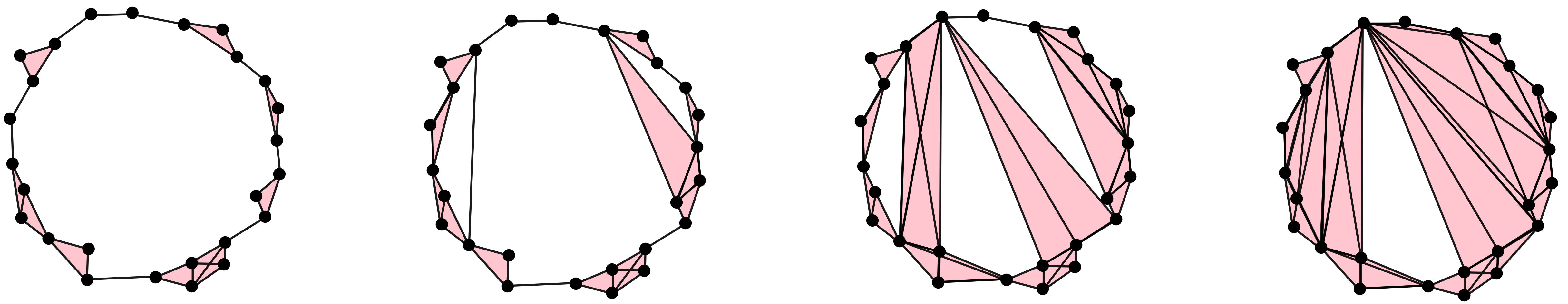
$$H_k(X) \not\sim H_k(Y) \Longrightarrow X \not\sim Y$$

Computation with filtrations and matrix reduction

Algorithms to compute the homology groups of a simplicial complex work by *decomposing* the simplicial complex, with a so-called *filtration*.

Computation with filtrations and matrix reduction

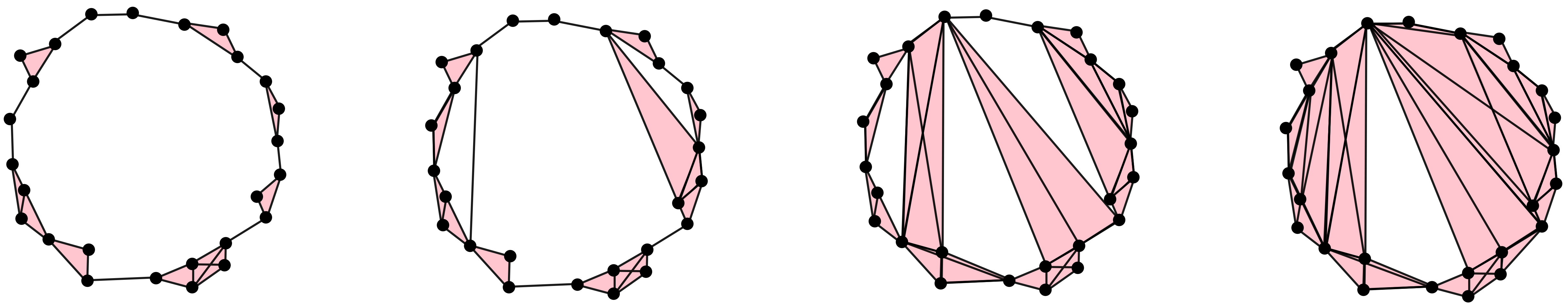
Algorithms to compute the homology groups of a simplicial complex work by *decomposing* the simplicial complex, with a so-called *filtration*.



Def: A filtered simplicial complex S is a family $\{S_a\}_{a\in\mathbb{R}}$ of subcomplexes of some fixed simplicial complex S s.t. $S_a \subseteq S_b$ for any $a \leq b$.

Computation with filtrations and matrix reduction

Algorithms to compute the homology groups of a simplicial complex work by *decomposing* the simplicial complex, with a so-called *filtration*.



Def: A filtered simplicial complex S is a family $\{S_a\}_{a \in \mathbb{R}}$ of subcomplexes of some fixed simplicial complex S s.t. $S_a \subseteq S_b$ for any $a \leq b$.

Def: Let f be a real valued function defined on the vertices of K. For $\sigma = [v_0, \ldots, v_k] \in K$, let $f(\sigma) = \max_{i=0,\ldots,k} f(v_i)$, and order the simplices of K in increasing order w.r.t. the function f values (and break ties with dimension in case some simplices have the same function value).

Q: Show that this is a filtration.

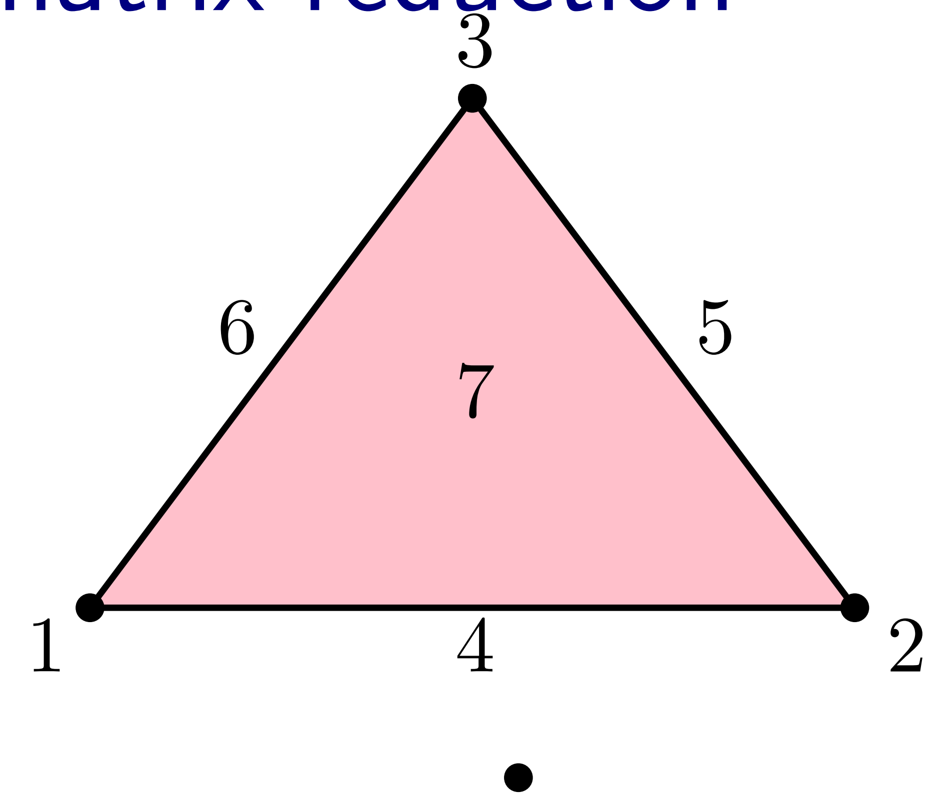
Computation with filtrations and matrix reduction

Input: simplicial filtration

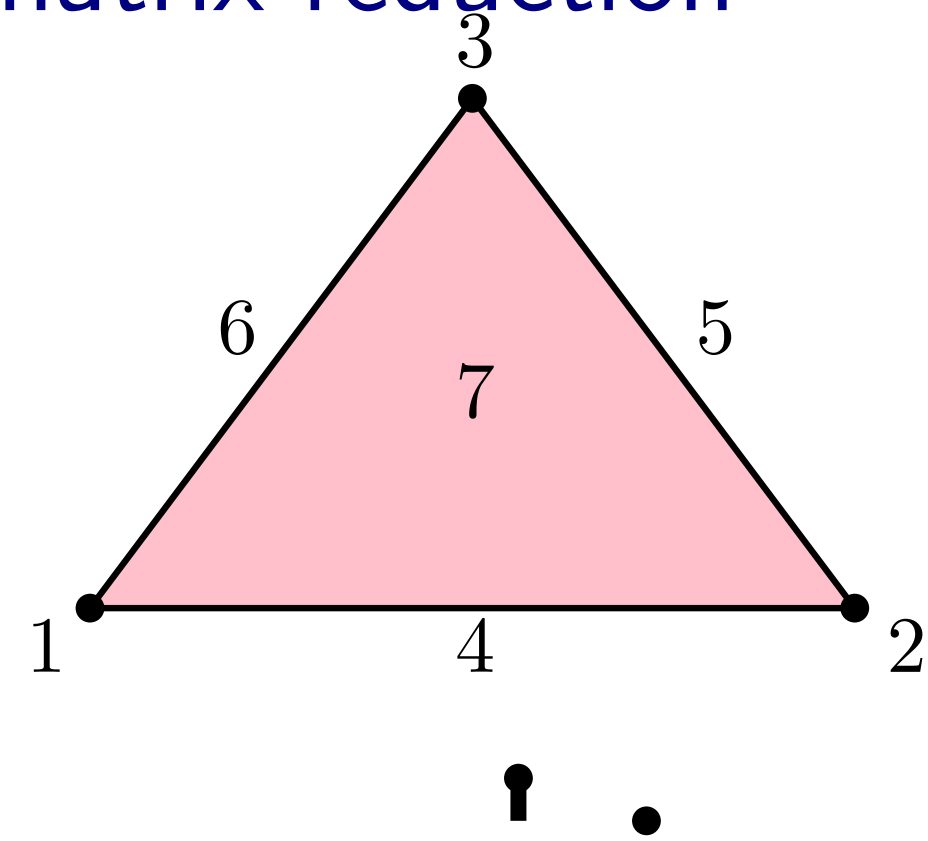
Homology can be computed by using the fact that each simplex is either:

positive, i.e., it *creates a new homology class*

negative, i.e., it *destroys an homology class*



Computation with filtrations and matrix reduction

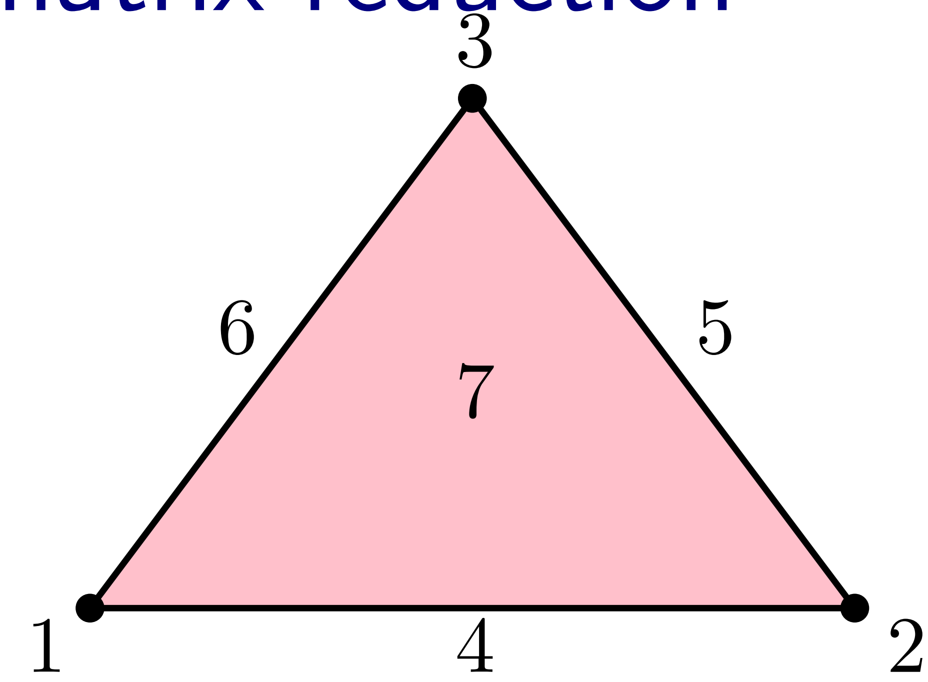


Input: simplicial filtration

Homology can be computed by using the fact that each simplex is either:

- *positive*, i.e., it *creates a new homology class*
- *negative*, i.e., it *destroys an homology class*

Computation with filtrations and matrix reduction

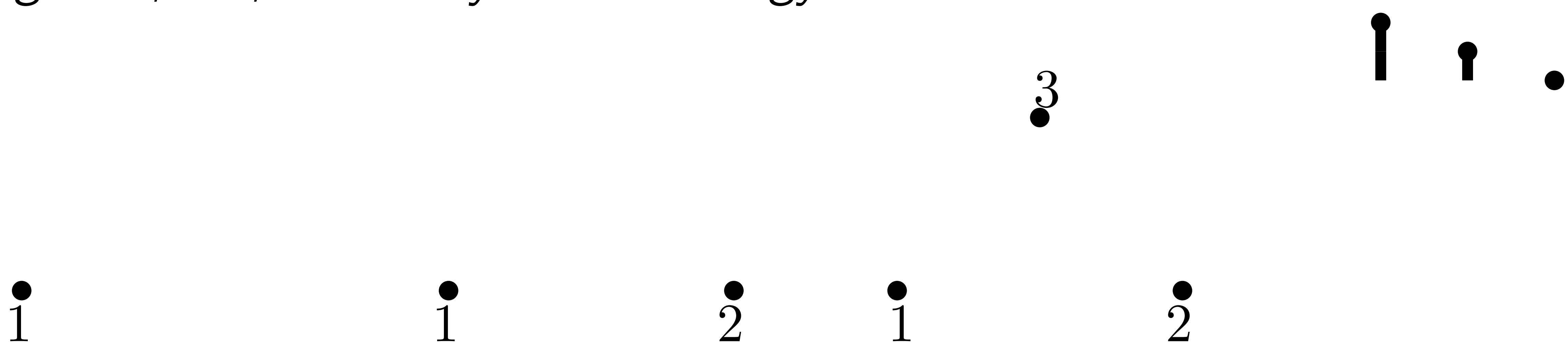


Input: simplicial filtration

Homology can be computed by using the fact that each simplex is either:

positive, i.e., it *creates a new homology class*

negative, i.e., it *destroys an homology class*



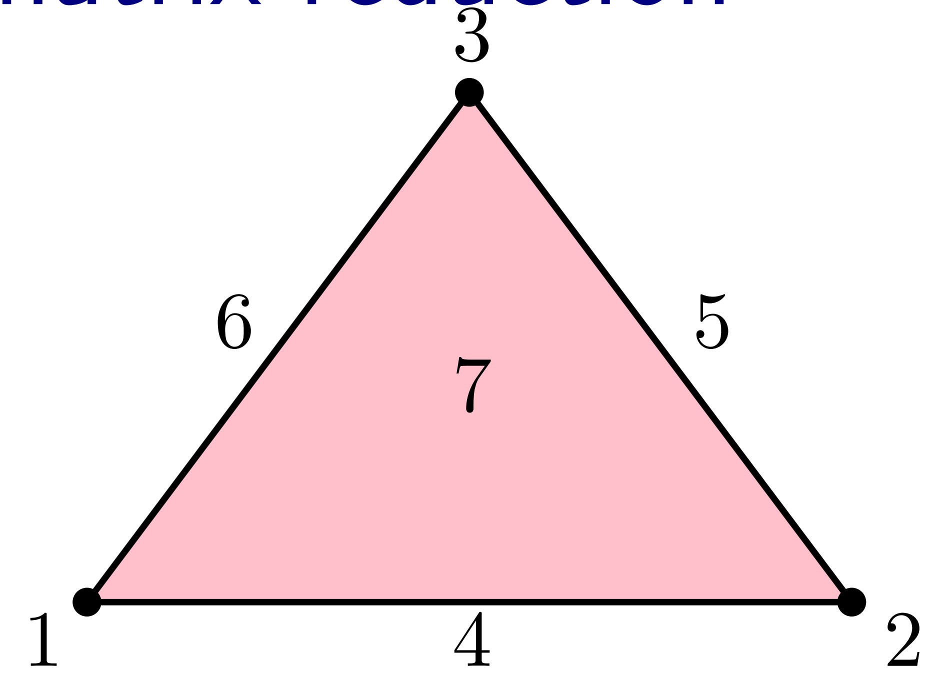
Computation with filtrations and matrix reduction

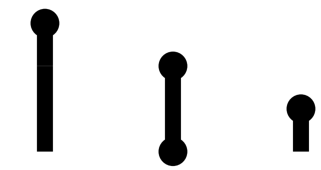
Input: simplicial filtration

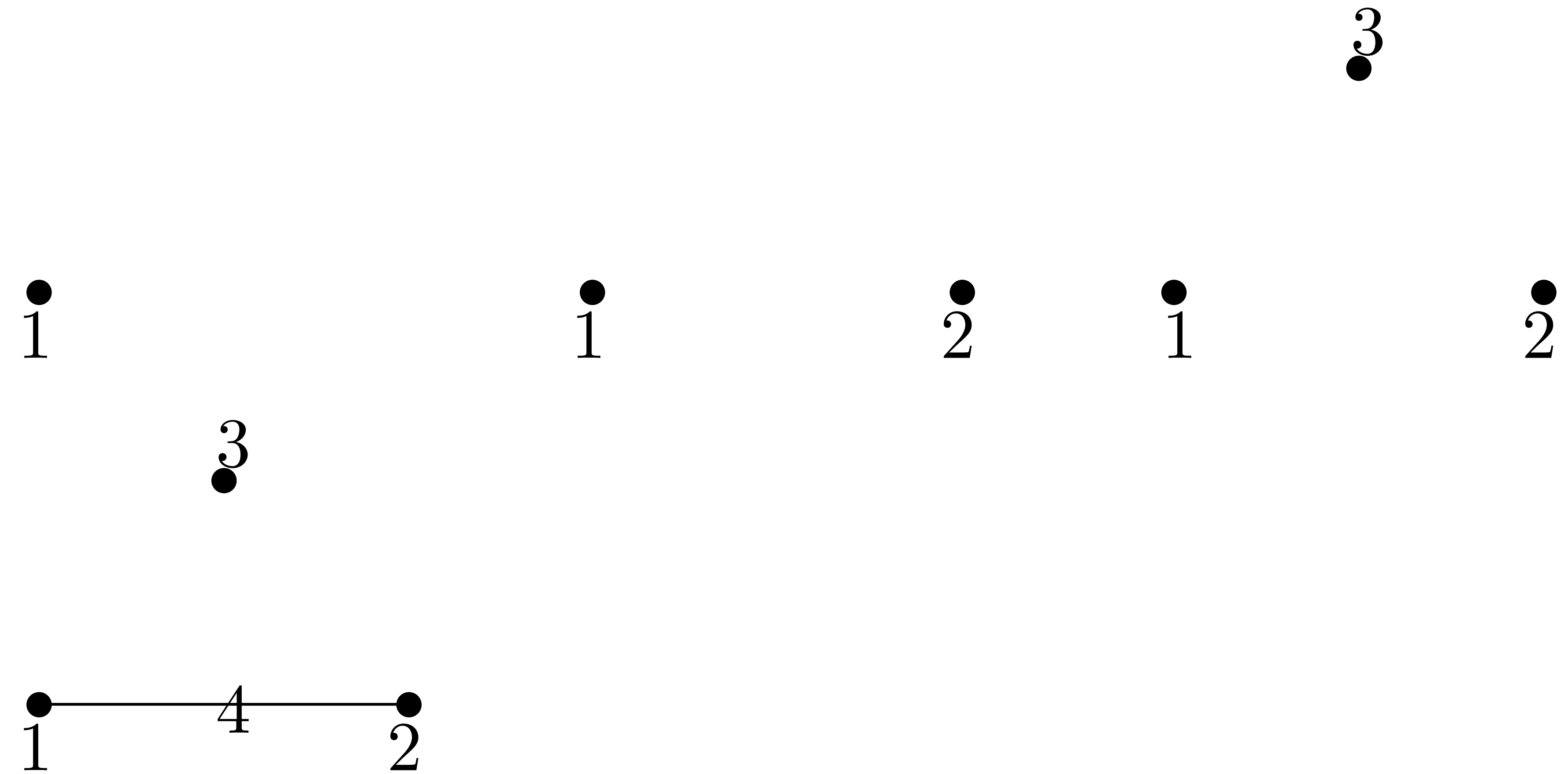
Homology can be computed by using the fact that each simplex is either:

positive, i.e., it *creates a new homology class*

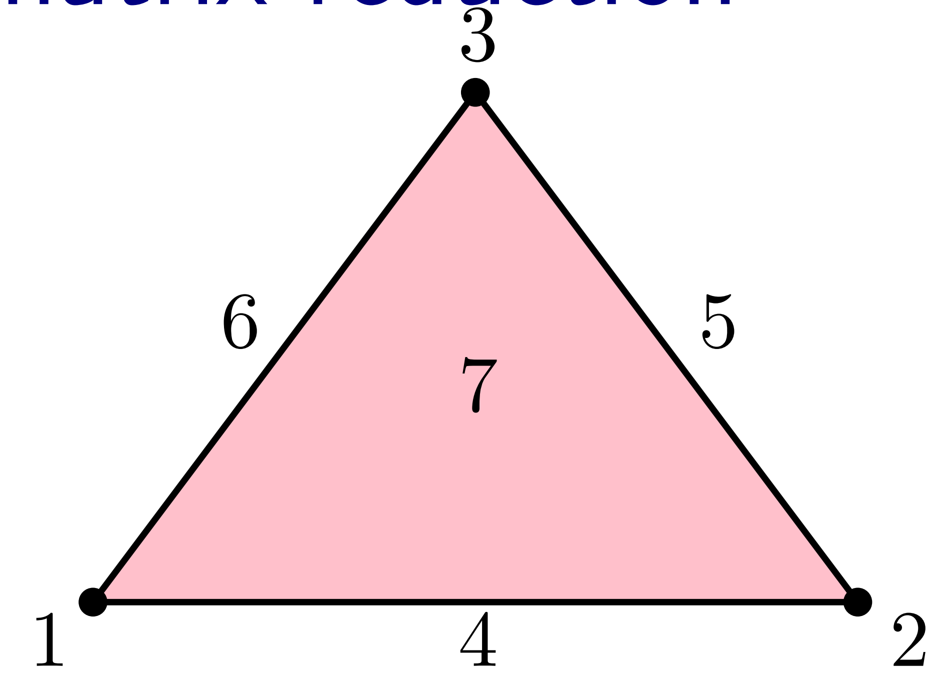
negative, i.e., it *destroys an homology class*







Computation with filtrations and matrix reduction

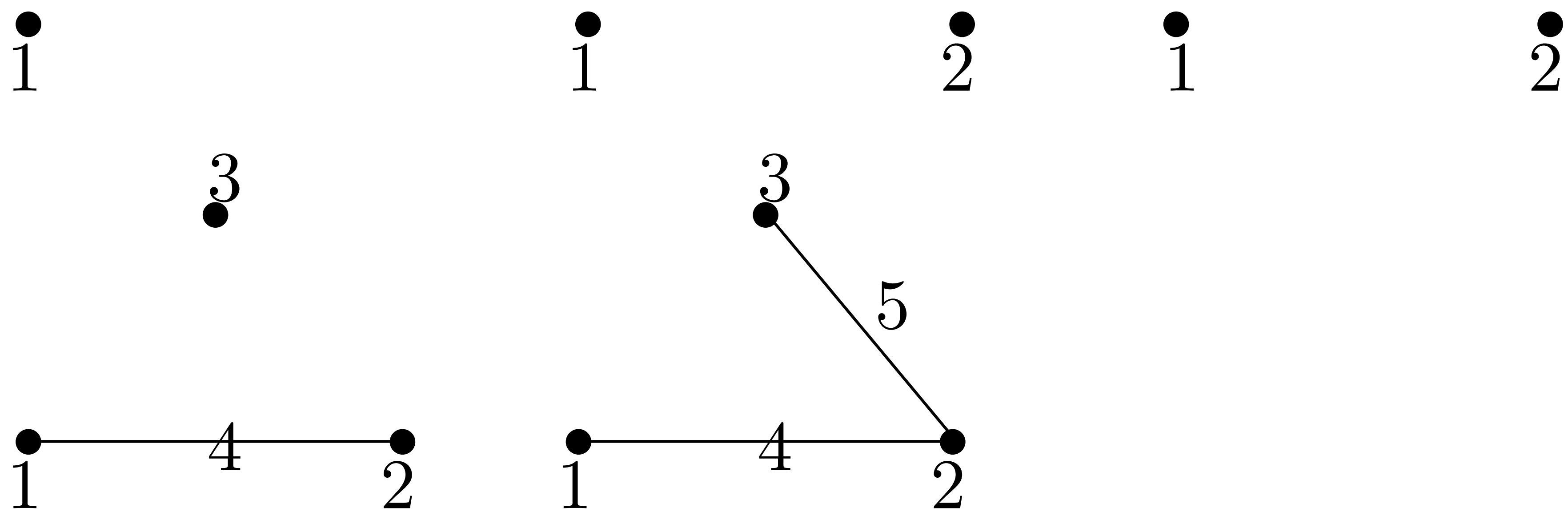


Input: simplicial filtration

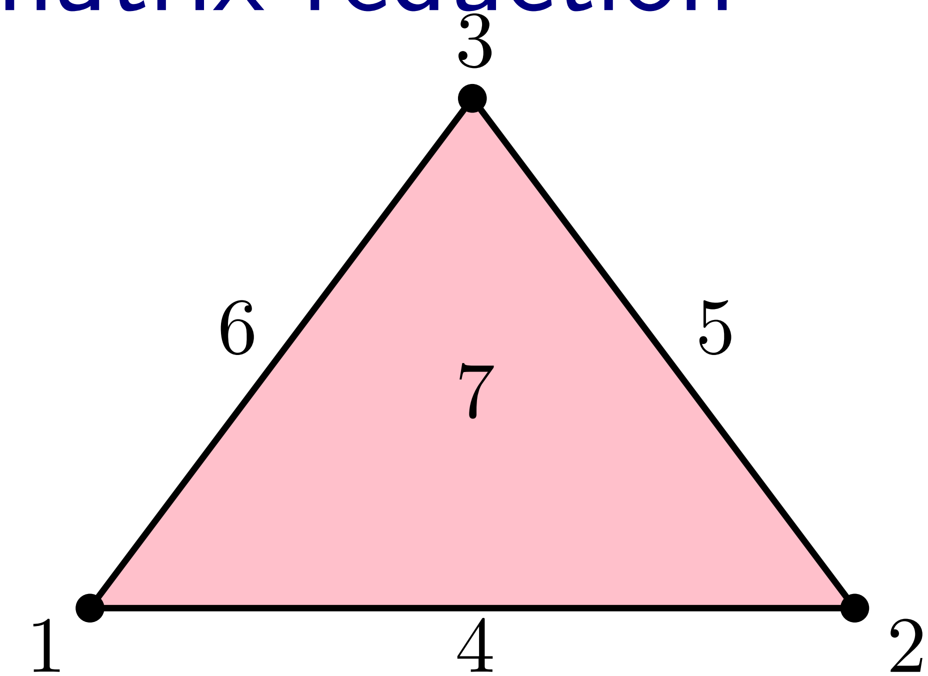
Homology can be computed by using the fact that each simplex is either:

positive, i.e., it *creates a new homology class*

negative, i.e., it *destroys an homology class*



Computation with filtrations and matrix reduction

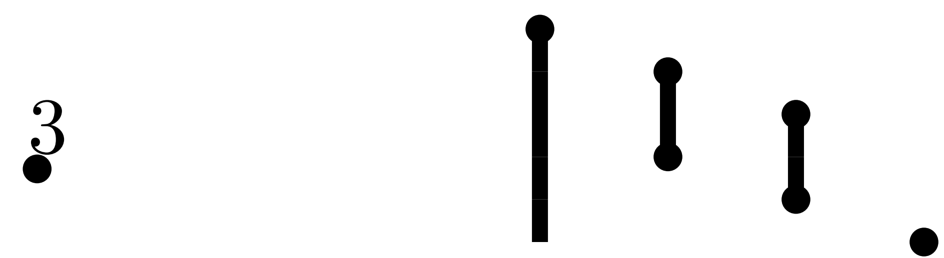


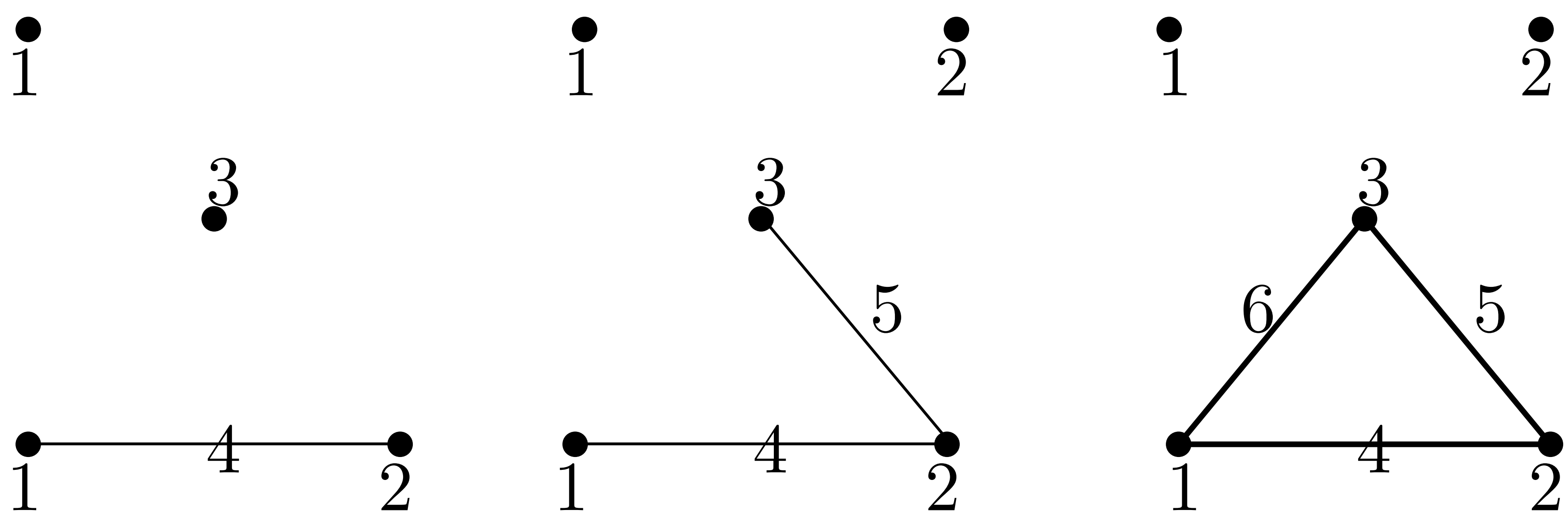
Input: simplicial filtration

Homology can be computed by using the fact that each simplex is either:

positive, i.e., it *creates a new homology class*

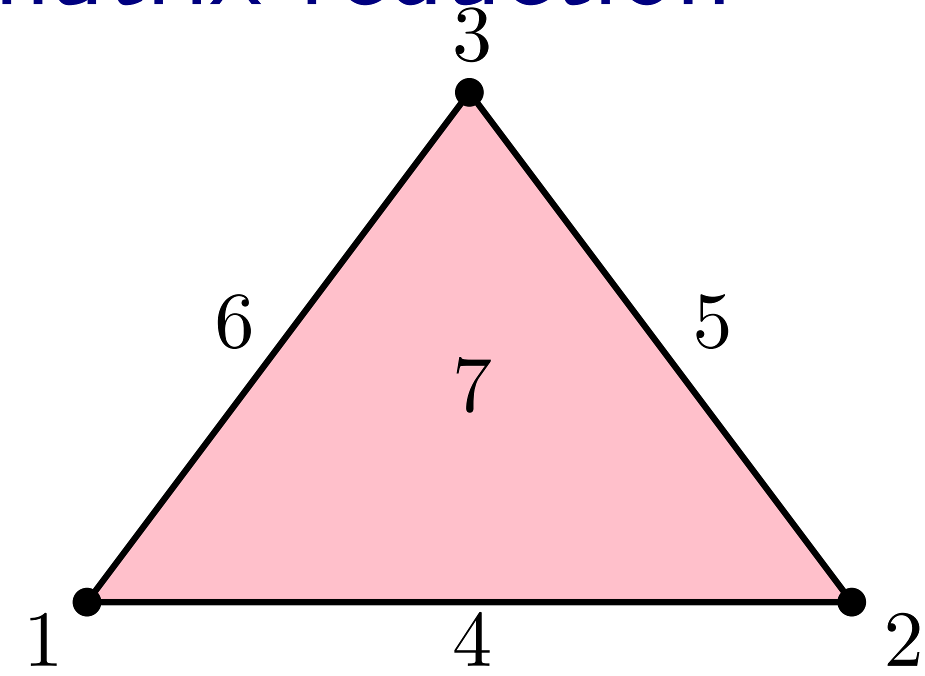
negative, i.e., it *destroys an homology class*





Computation with filtrations and matrix reduction

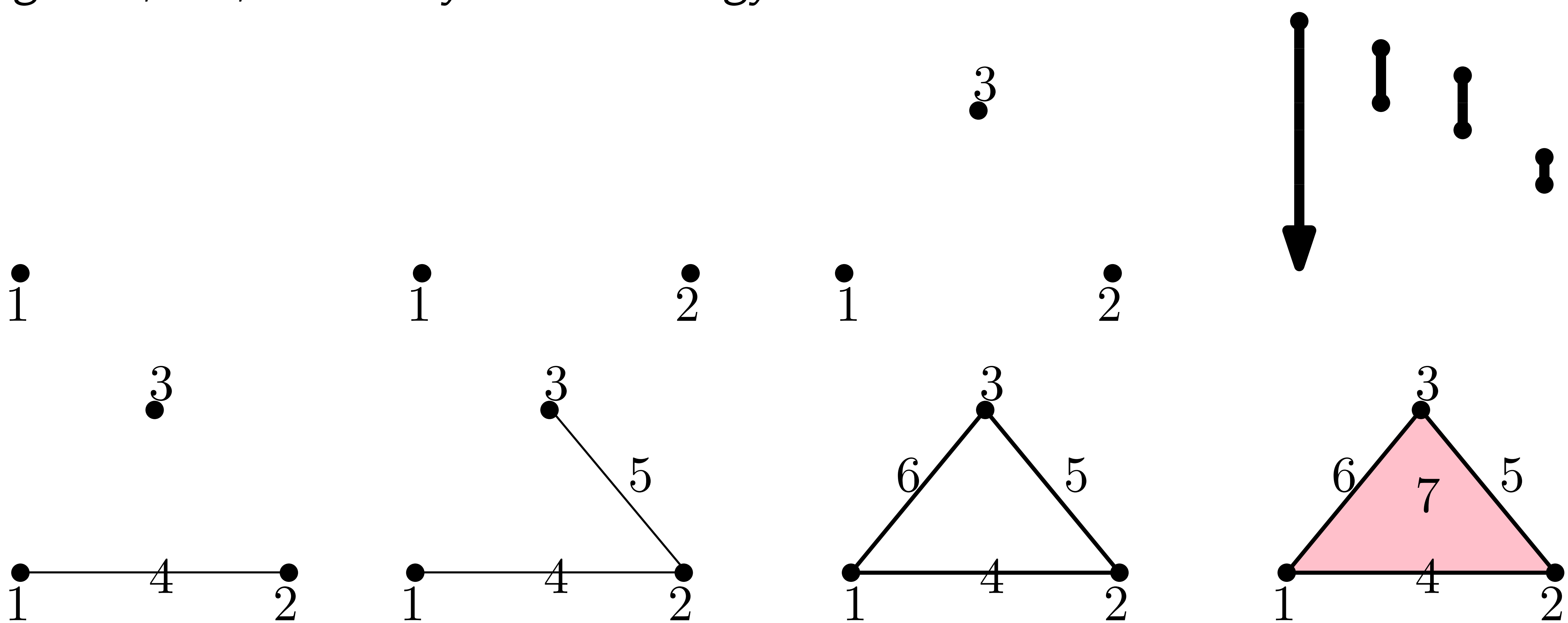
Input: simplicial filtration



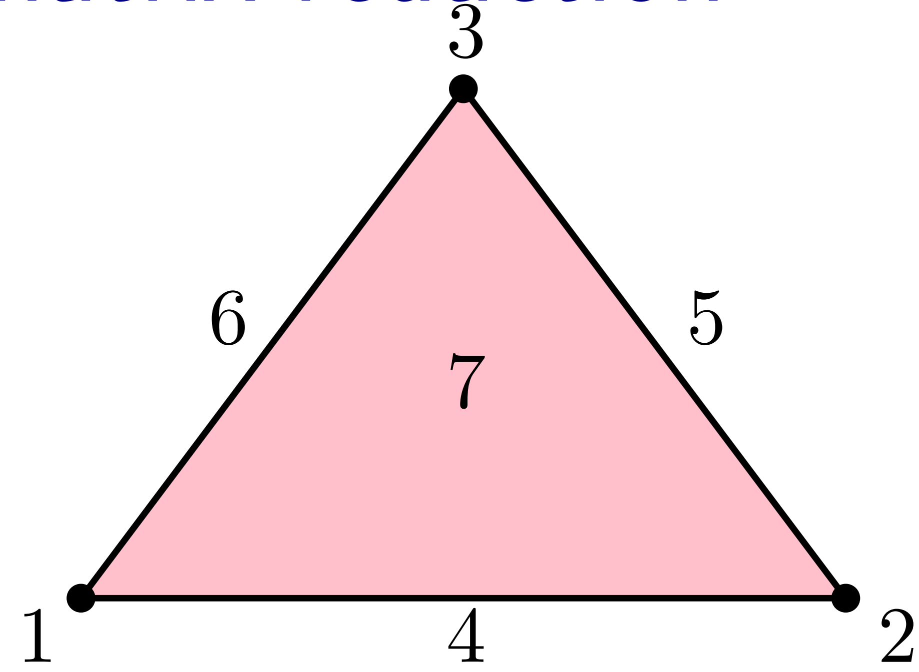
Homology can be computed by using the fact that each simplex is either:

positive, i.e., it *creates a new homology class*

negative, i.e., it *destroys an homology class*



Computation with filtrations and matrix reduction

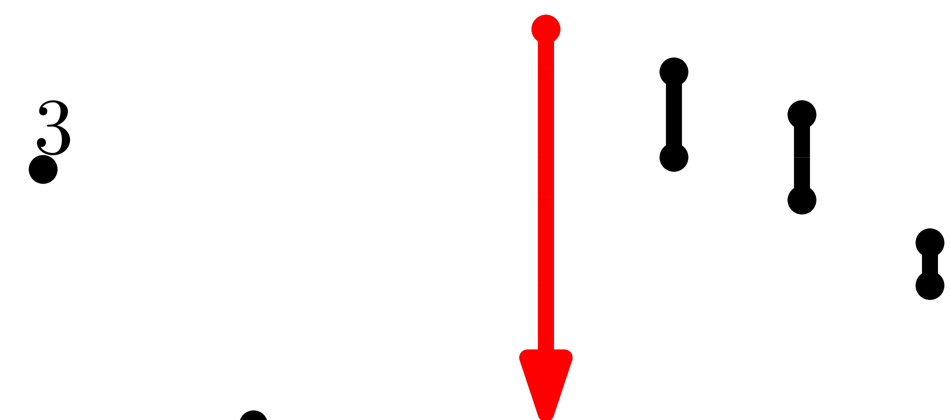


Input: simplicial filtration

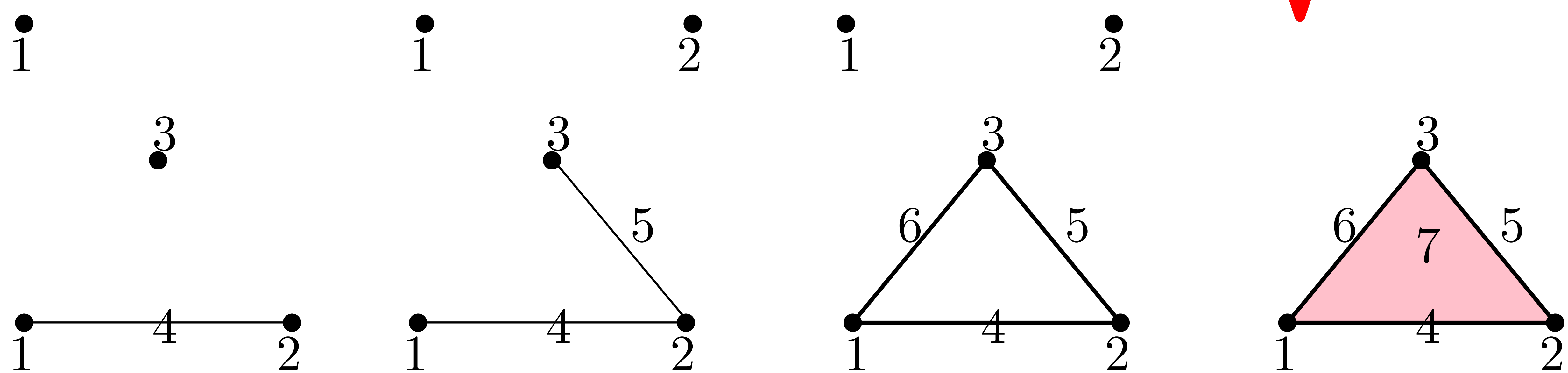
Homology can be computed by using the fact that each simplex is either:

positive, i.e., it *creates a new homology class*

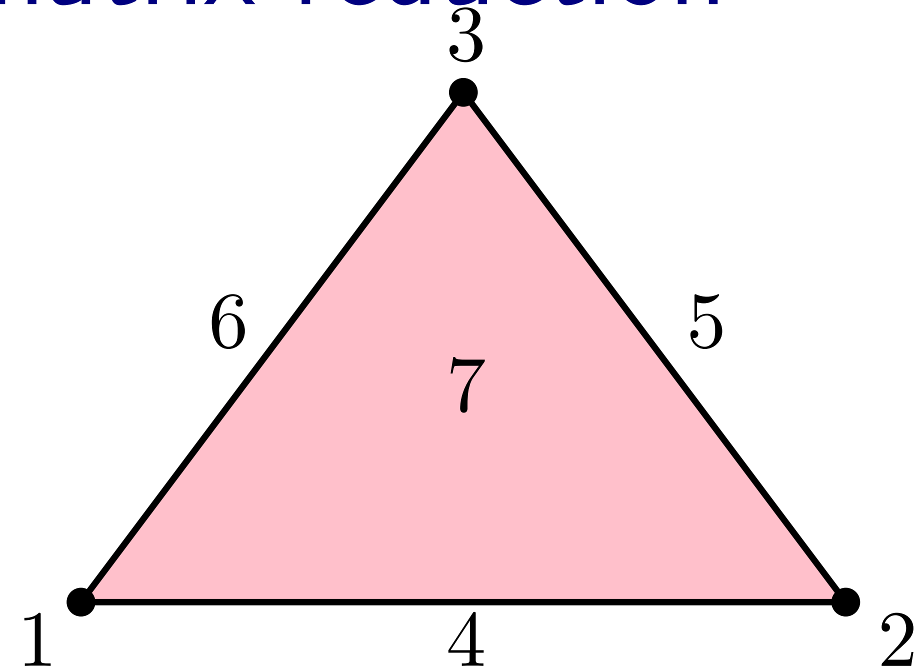
negative, i.e., it *destroys an homology class*



The Betti number is equal to the number of bars that are still alive when the full complex is reached in the filtration



Computation with filtrations and matrix reduction

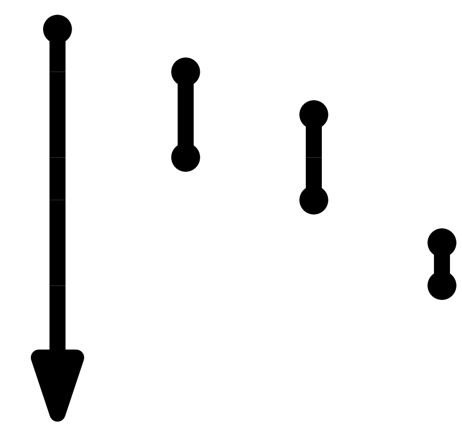


Input: simplicial filtration

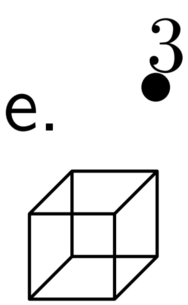
Homology can be computed by using the fact that each simplex is either:

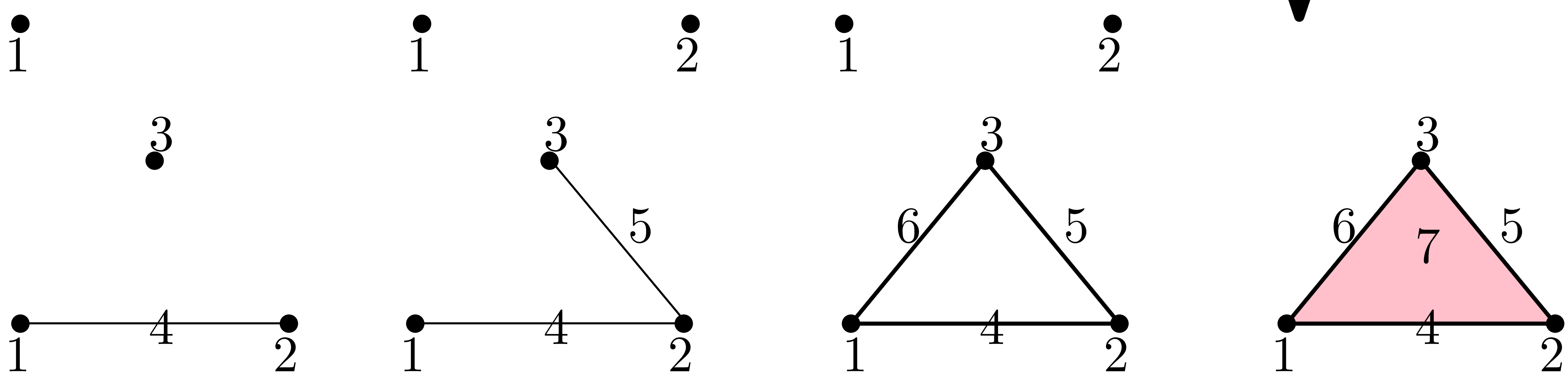
positive, i.e., it *creates a new homology class*

negative, i.e., it *destroys an homology class*



Q: Do the same for the homology of the cube.



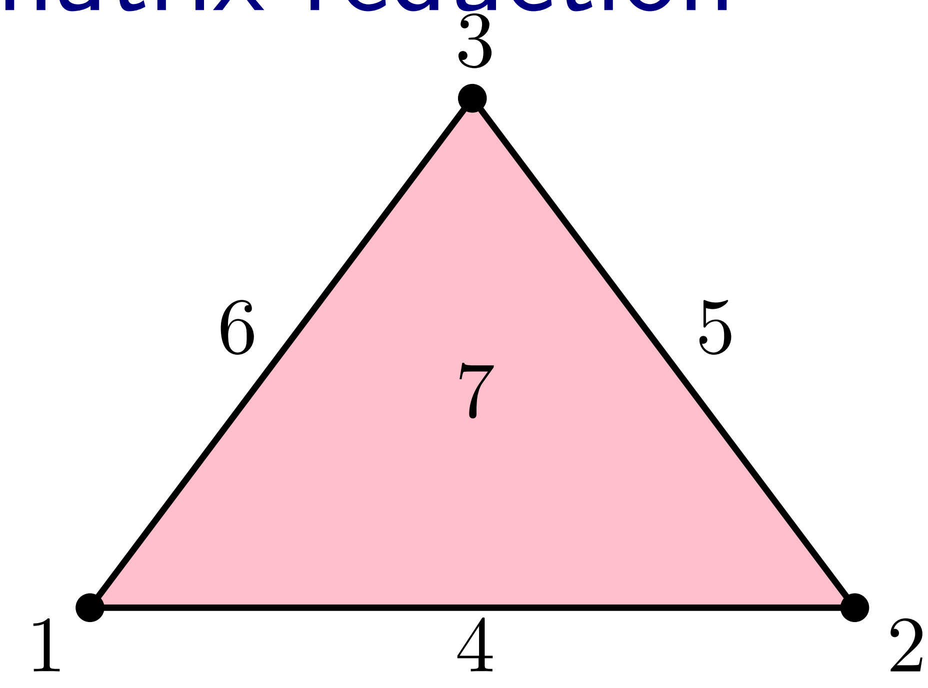


Computation with filtrations and matrix reduction

Input: simplicial filtration

given as *boundary matrix*

	1	2	3	4	5	6	7
1							
2							
3							
4							
5							
6							
7							

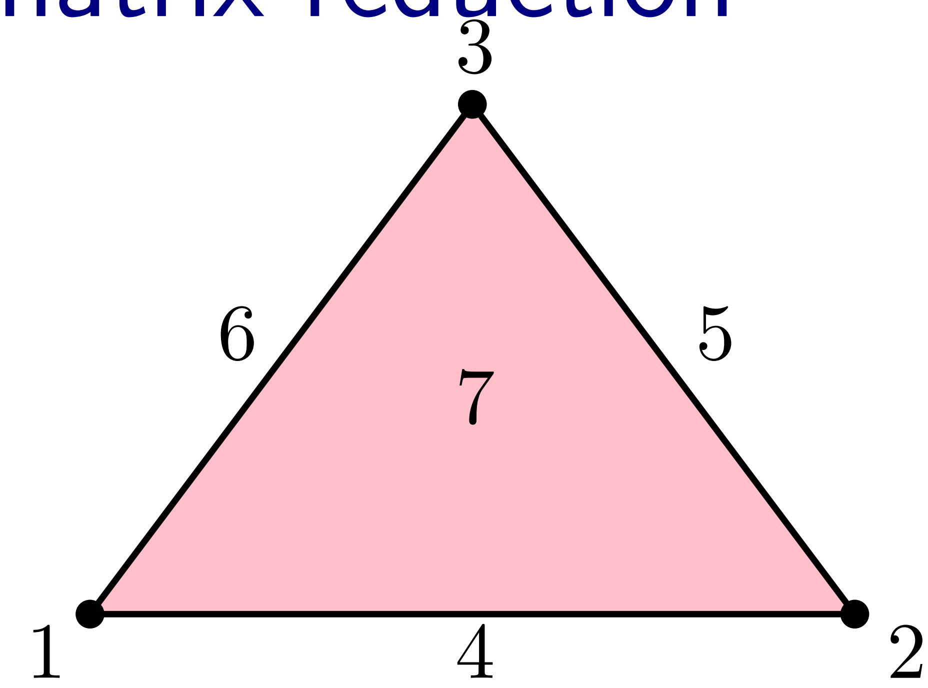


Computation with filtrations and matrix reduction

Input: simplicial filtration

given as *boundary matrix*

	1	2	3	4	5	6	7
1				•			
2				•			
3							
4							
5							
6							
7							

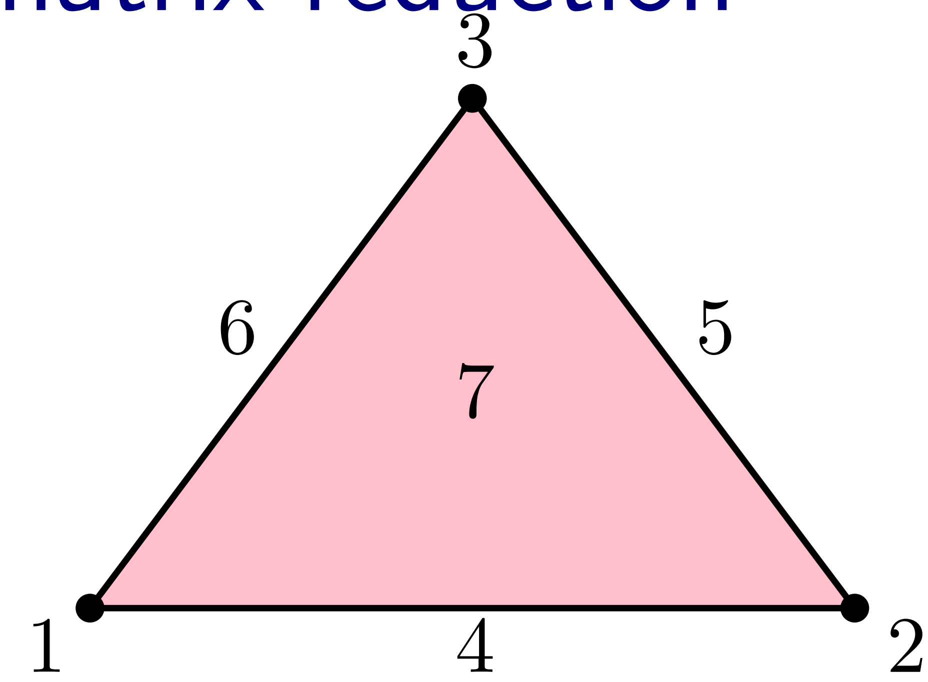


Computation with filtrations and matrix reduction

Input: simplicial filtration

given as *boundary matrix*

	1	2	3	4	5	6	7
1				•			
2				•	•		
3					•		
4							
5							
6							
7							

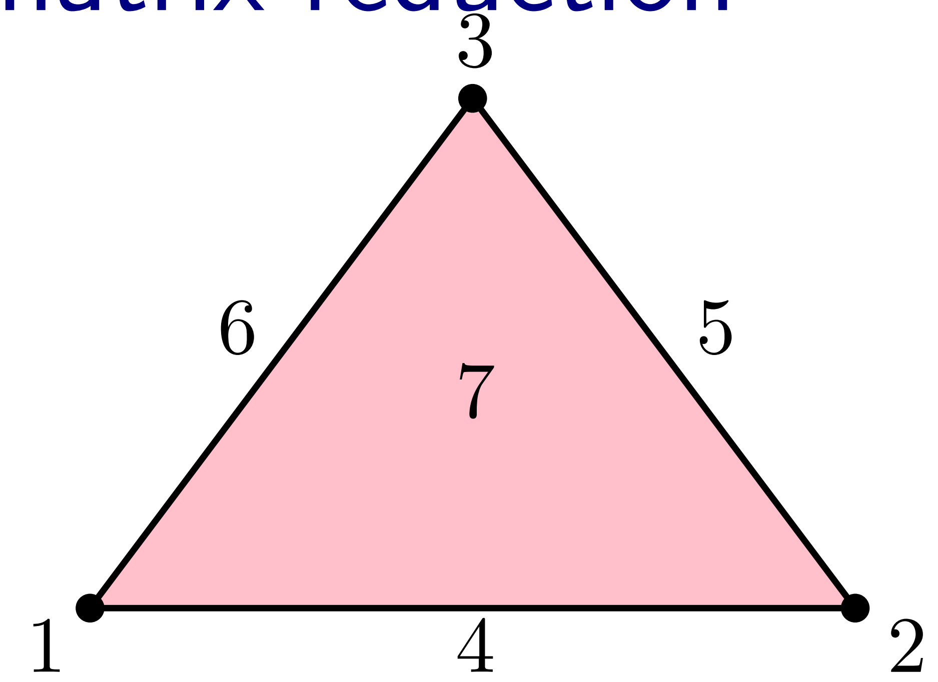


Computation with filtrations and matrix reduction

Input: simplicial filtration

given as *boundary matrix*

	1	2	3	4	5	6	7
1				•		•	
2				•	•		
3					•	•	
4							
5							
6							
7							

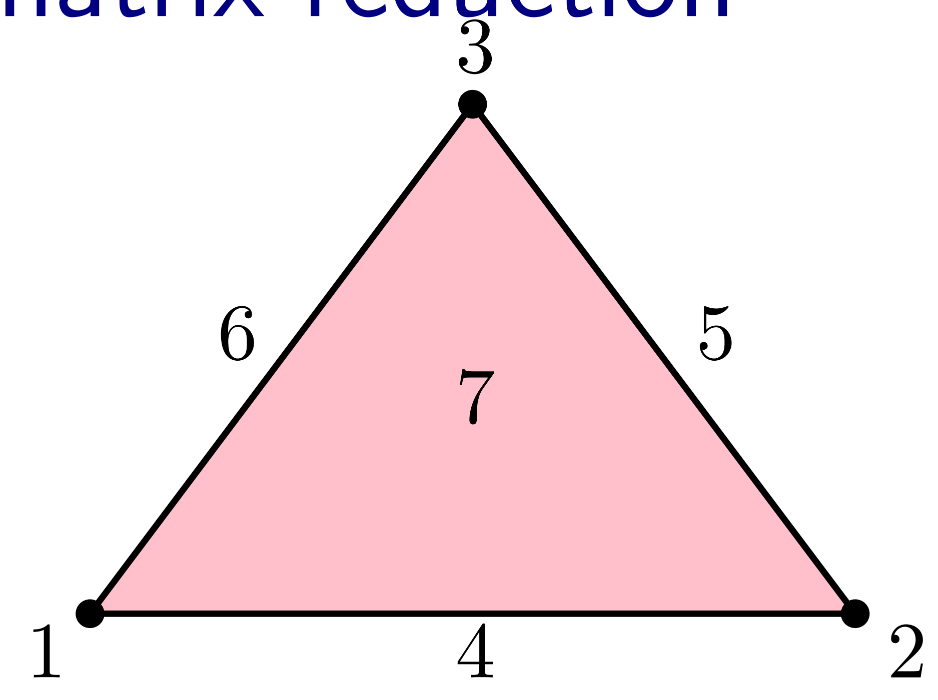


Computation with filtrations and matrix reduction

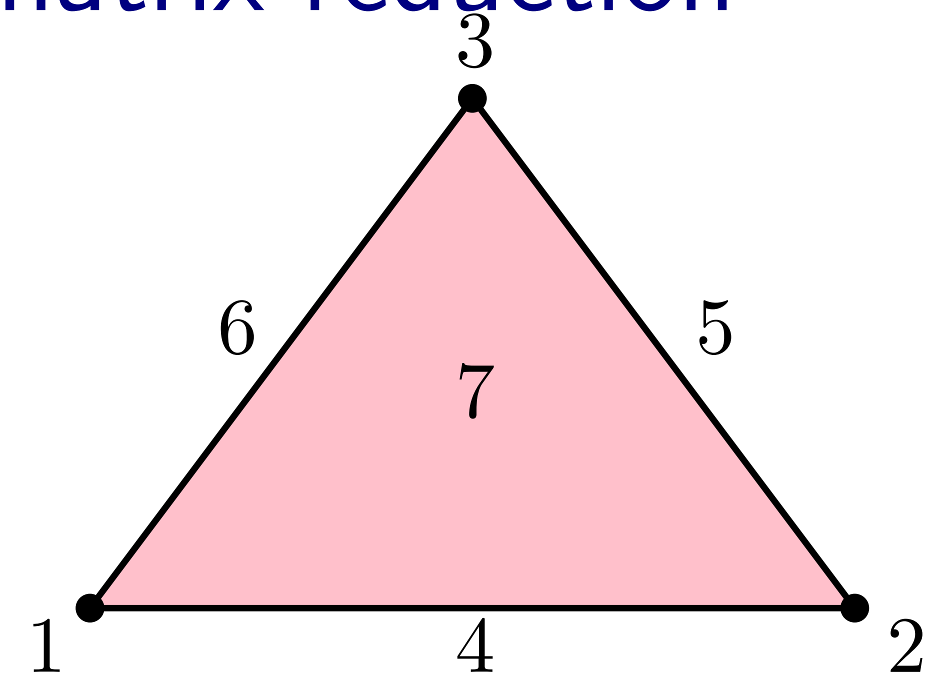
Input: simplicial filtration

given as *boundary matrix*

	1	2	3	4	5	6	7
1				•		•	
2				•	•		
3					•	•	
4							•
5							•
6							•
7							



Computation with filtrations and matrix reduction



Input: simplicial filtration

given as *boundary matrix*

	1	2	3	4	5	6	7
1				•		•	
2				•	•		
3					•	•	
4							•
5							•
6							•
7							

`for` j=1 `to` m `do:`

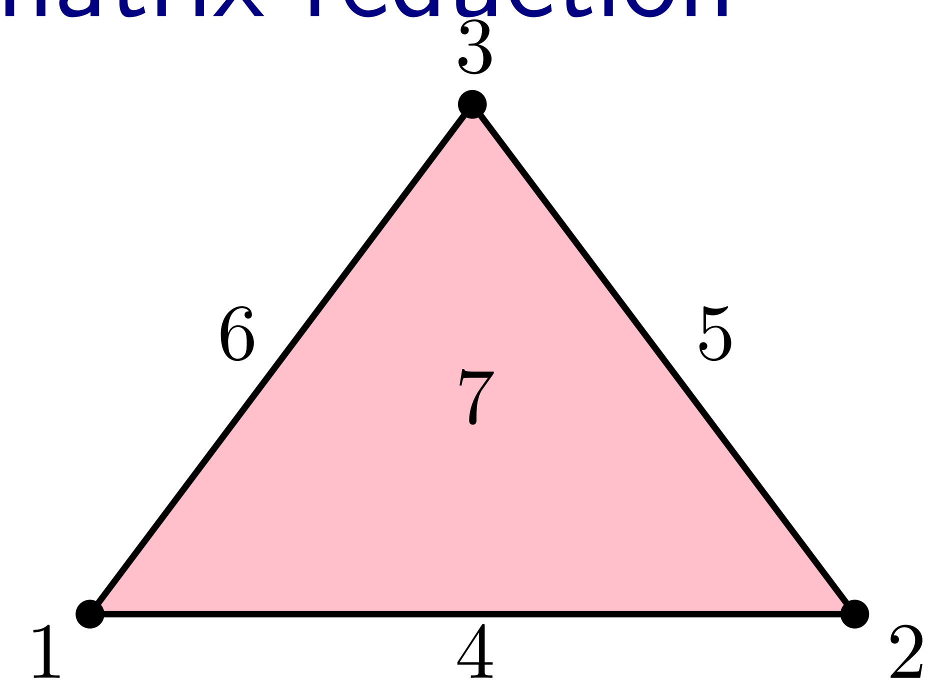
`while` $\exists k < j$ `s.t.` low(k) ==low(j) `do:`

col(j) =col(j)+col(k)

Computation with filtrations and matrix reduction

Input: simplicial filtration

given as *boundary matrix*



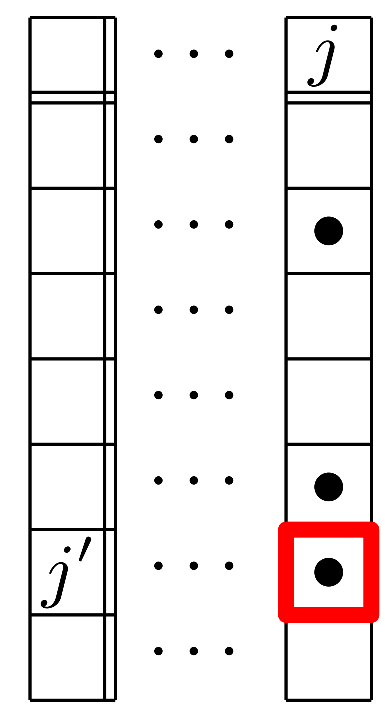
	1	2	3	4	5	6	7
1				●		●	
2				●	●		
3					●	●	
4							●
5							●
6							●
7							

`for` j=1 `to m do:`

`while` $\exists k < j$ `s.t.` low(k) ==low(j) `do:`

col(j) =col(j)+col(k)

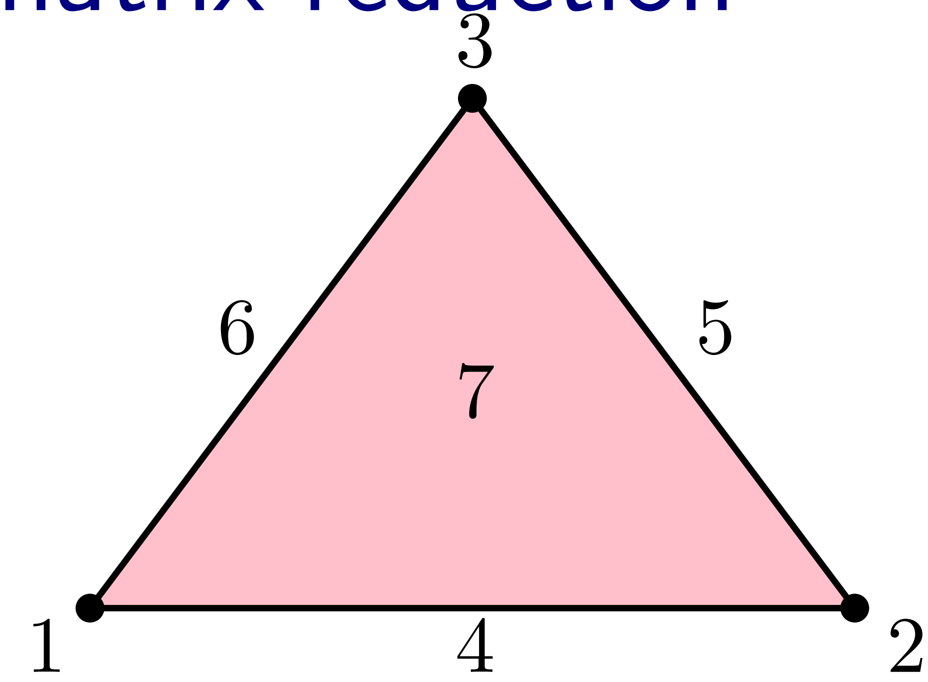
low$(j) = j'$



Computation with filtrations and matrix reduction

Input: simplicial filtration

given as *boundary matrix*



	1	2	3	4	5	6	7
1				•		•	
2				•	•		
3					$\boxed{•}$	$\boxed{•}$	
4							•
5							•
6							•
7							

for j=1 to m do:

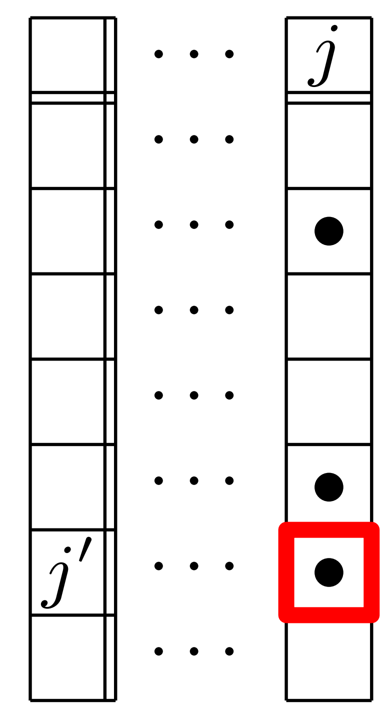
while $\exists k < j$ s.t. low(k) ==low(j) do:

col(j) =col(j)+col(k)

5	6
	•
•	
•	•

6 = 6+5

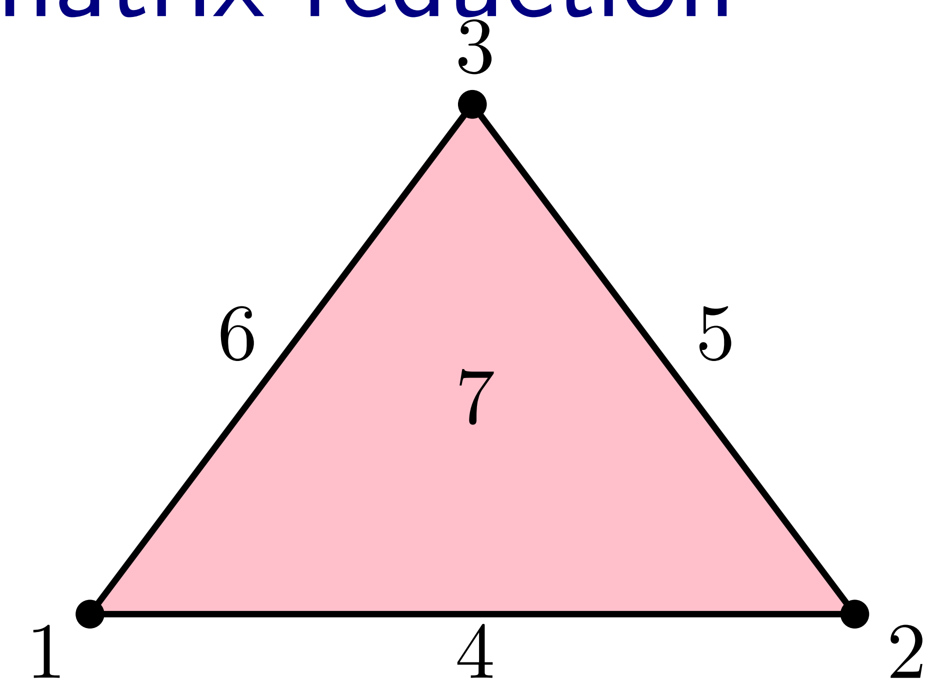
low$(j) = j'$



Computation with filtrations and matrix reduction

Input: simplicial filtration

given as *boundary matrix*



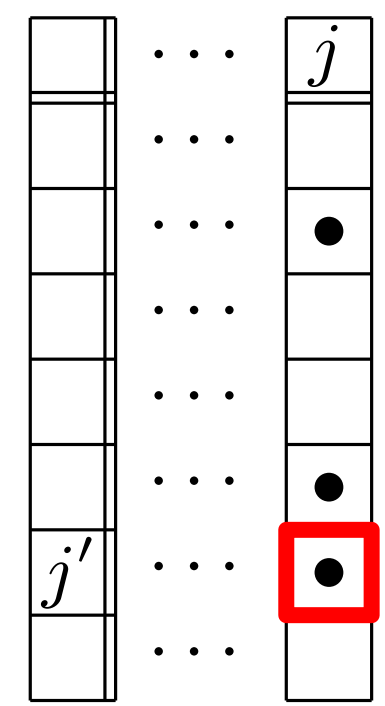
	1	2	3	4	5	6	7
1				•		•	
2				•	•	•	
3					•		
4							•
5							•
6							•
7							

```
for j=1 to m do:
    while ∃k < j s.t. low(k) ==low(j) do:
        col(j) =col(j)+col(k)
```

5	6
	•
•	•
•	

6 = 6+5

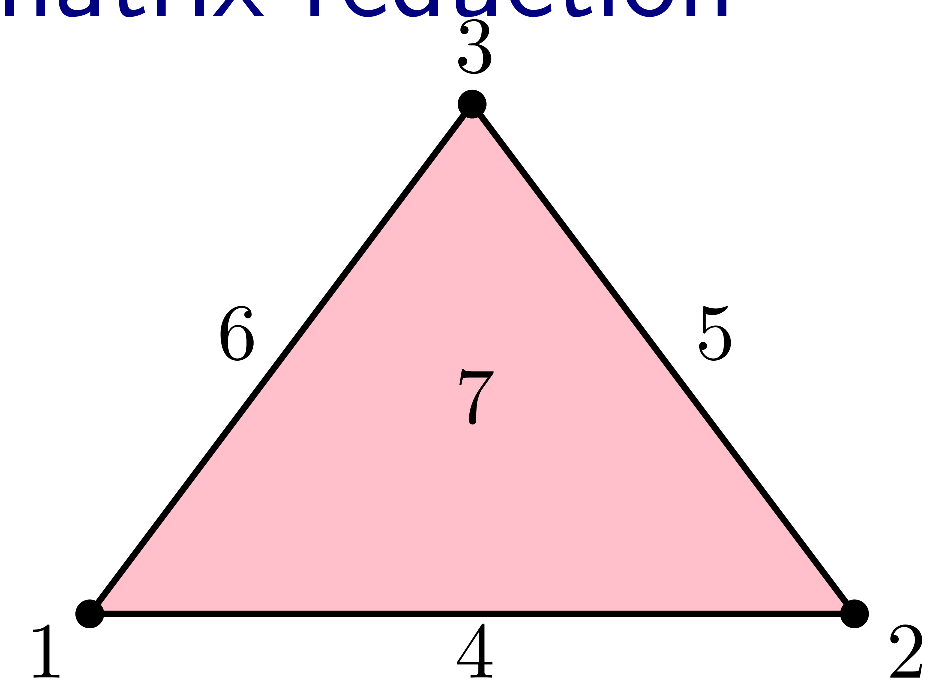
$\mathsf{low}(j) = j'$



Computation with filtrations and matrix reduction

Input: simplicial filtration

given as *boundary matrix*



	1	2	3	4	5	6	7
1				•		•	
2				$\boxed{•}$	•	$\boxed{•}$	
3					•		
4							•
5							•
6							•
7							

`for j=1 to m do:`

`while` $\exists k < j$ `s.t.` low(k) ==low(j) `do:`

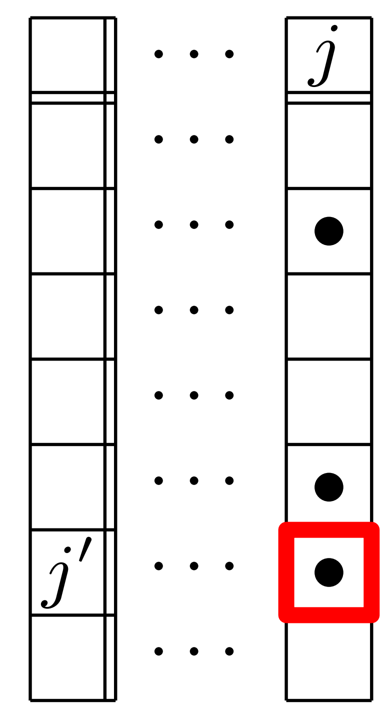
col(j) =col(j)+col(k)

5	6
	•
•	•
•	

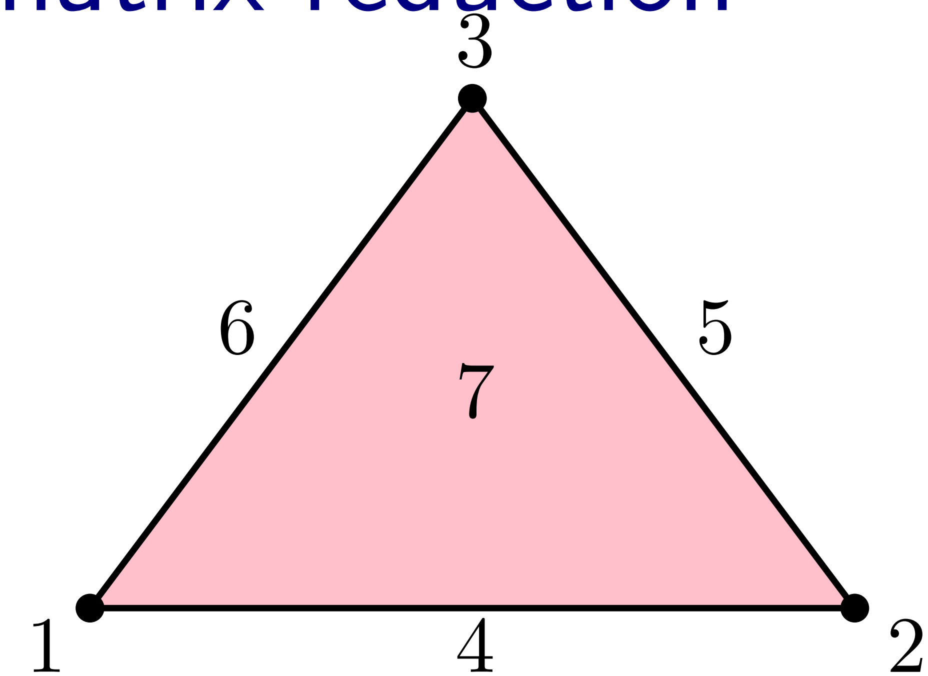
4		6
•		•
•		•

6 = 6+5
6 = 6+4

low$(j) = j'$



Computation with filtrations and matrix reduction



Input: simplicial filtration

given as *boundary matrix*

	1	2	3	4	5	6	7
1				●			
2				●	●		
3					●		
4							●
5							●
6							●
7							

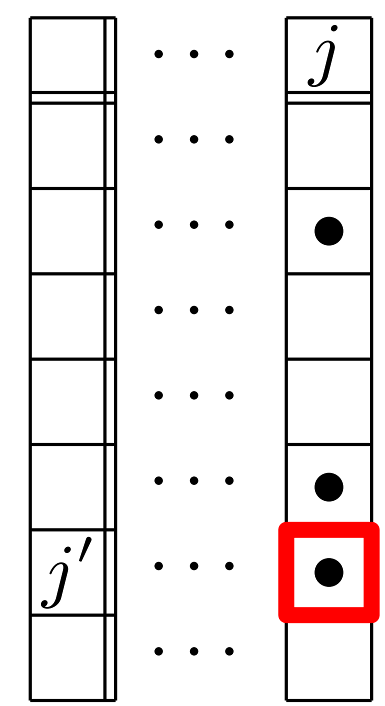
```
for j=1 to m do:
    while ∃k < j s.t. low(k) ==low(j) do:
        col(j) =col(j)+col(k)
```

5	6
	●
●	●
●	

4		6
●		
●		

$$6 = 6{+}5$$
$$6 = 6{+}4$$

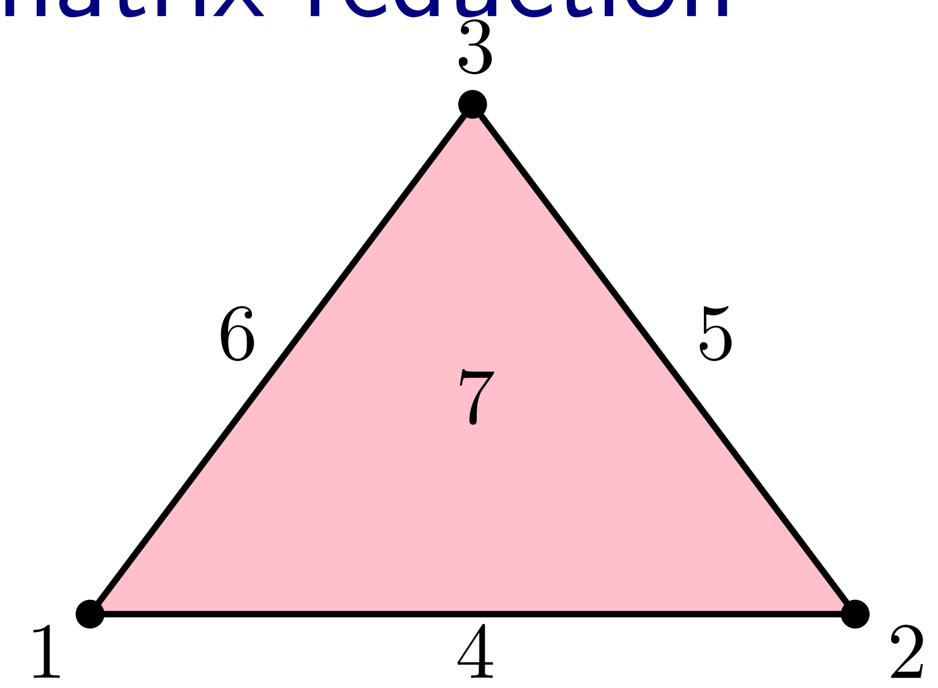
$\mathrm{low}(j) = j'$



Computation with filtrations and matrix reduction

Input: simplicial filtration

Output: boundary matrix

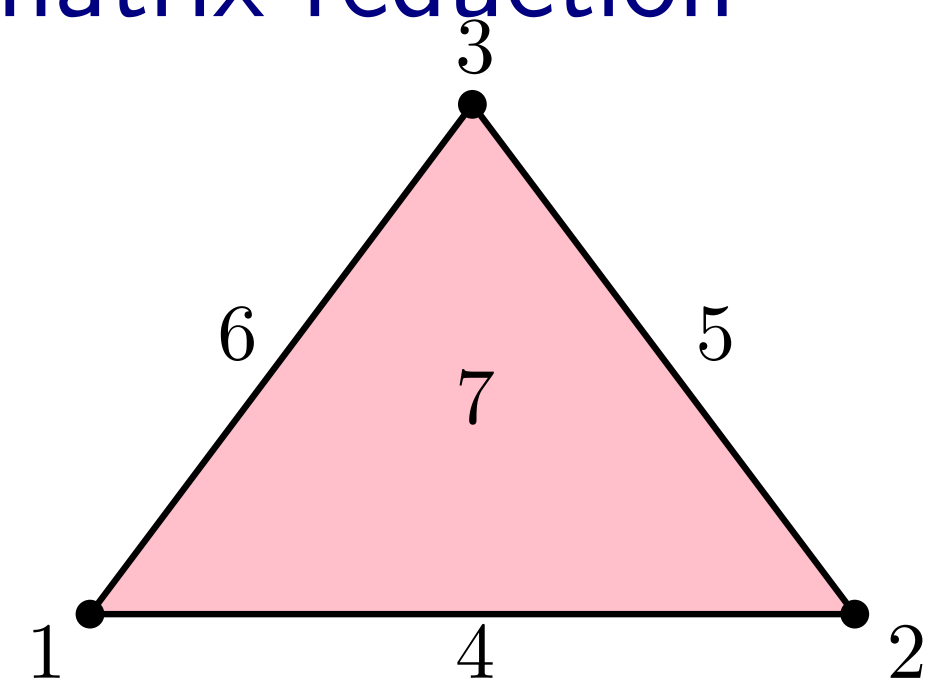


	1	2	3	4	5	6	7
1				*		*	
2				*	*		
3					*	*	
4							*
5							*
6							*
7							

Computation with filtrations and matrix reduction

Input: simplicial filtration

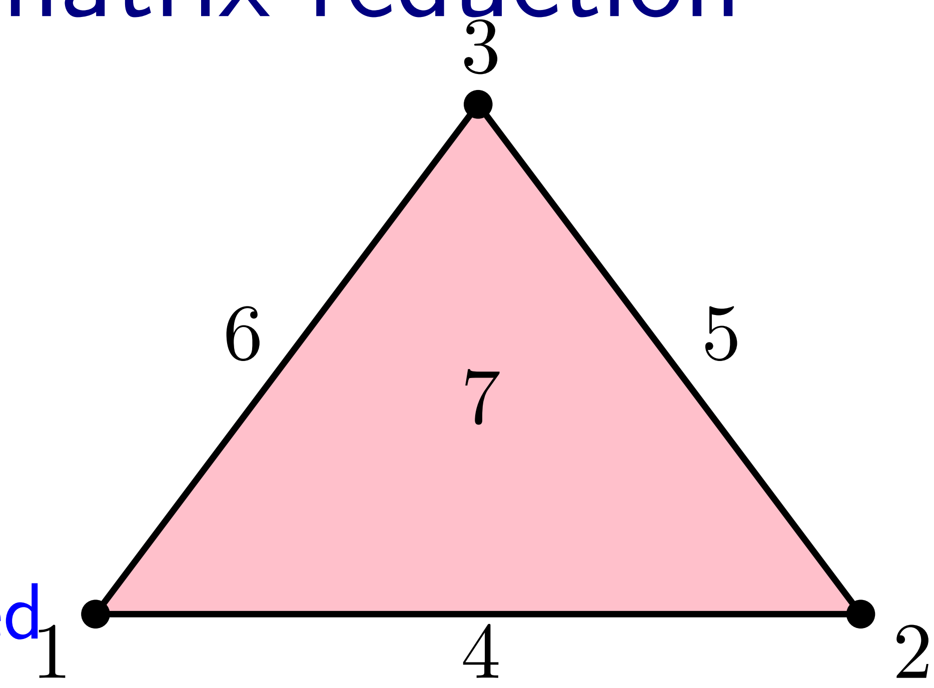
Output: boundary matrix
reduced to column-echelon form



	1	2	3	4	5	6	7
1				*		*	
2				*	*		
3					*	*	
4							*
5							*
6							*
7							

	1	2	3	4	5	6	7
1				*			
2				1	*		
3					1		
4							*
5							*
6							1
7							

Computation with filtrations and matrix reduction



Input: simplicial filtration

Output: boundary matrix
reduced to column-echelon form

some positive-negative simplices are paired
$[2,4)$, $[3,5)$, $[6,7)$

unpaired simplices provide homology basis: $[1,+\infty)$

	1	2	3	4	5	6	7
1				*		*	
2				*	*		
3					*	*	
4							*
5							*
6							*
7							

	1	2	3	4	5	6	7
1				*			
2				1	*		
3					1		
4							*
5							*
6							1
7							

Computation with filtrations and matrix reduction

Input: simplicial filtration

Output: boundary matrix
reduced to column-echelon form

Q: Complexity?

Computation with filtrations and matrix reduction

Input: simplicial filtration

Output: boundary matrix
reduced to column-echelon form

Q: Complexity?

PLU factorization:

- Gaussian elimination
- fast matrix multiplication (divide-and-conquer)
- random projections?

Computation with filtrations and matrix reduction

Input: simplicial filtration

Output: boundary matrix
reduced to column-echelon form

Q: Complexity?

PLU factorization:

- Gaussian elimination
  - PLEX / JavaPLEX (http://appliedtopology.github.io/javaplex/)
  - Dionysus (http://www.mrzv.org/software/dionysus/)
  - Perseus (http://www.sas.upenn.edu/~vnanda/perseus/)
  - Gudhi (http://gudhi.gforge.inria.fr/)
  - PHAT (https://bitbucket.org/phat-code/phat)
  - DIPHA (https://github.com/DIPHA/dipha/)
  - CTL (https://github.com/appliedtopology/ctl)

Computation with filtrations and matrix reduction

Q: Triangulate and compute homology of dunce cap: