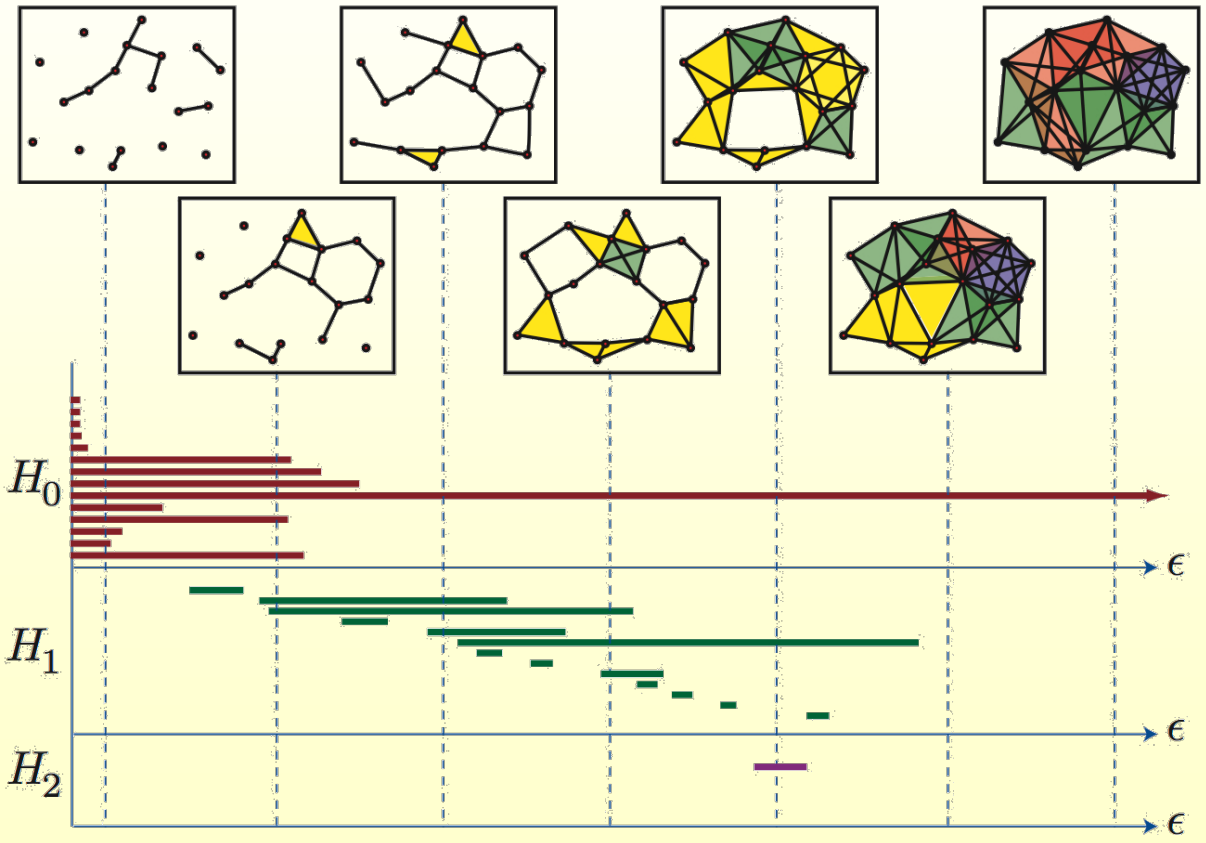
CS233: Geometric and Topological Data Analysis

Homology,

Persistent Homology

25 April 2018

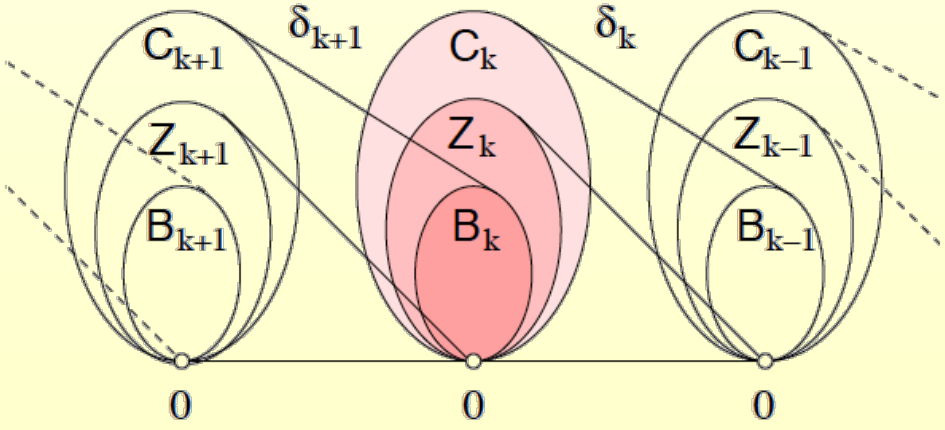


Last Time: Simplicial Homology

- The *k*th homology group is

$$\mathsf{H}_k = \mathsf{Z}_k/\mathsf{B}_k = \ker \partial_k / \operatorname{im} \partial_{k+1}.$$

- If $z_1 = z_2 + \mathsf{B}_k, z_1, z_2 \in \mathsf{Z}_k$, we say z_1 and z_2 are homologous
- $z_1 \sim z_2$.

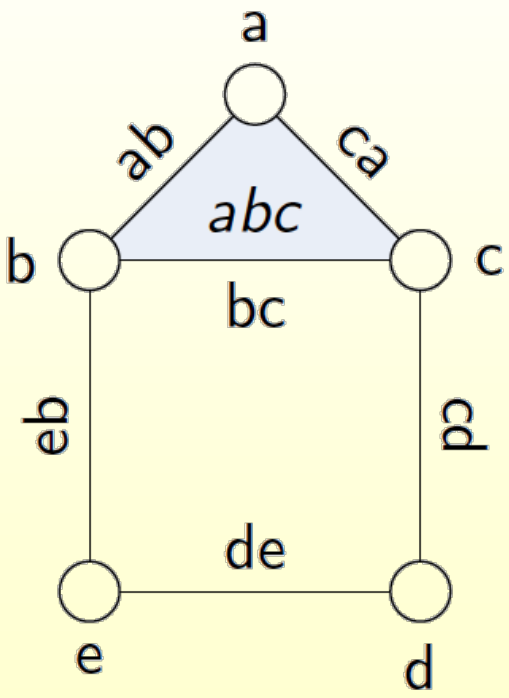


Today

- Homology
- Persistent homology

Linear algebra

Simplicial Complexes



A simplicial complex is a collection of simplices

- Each simplex has a dimension.
- Collection is closed under subset relation.
- Simplices of dimension d have $d + 1$ vertices
- Each simplex represented by an ordered list of vertices

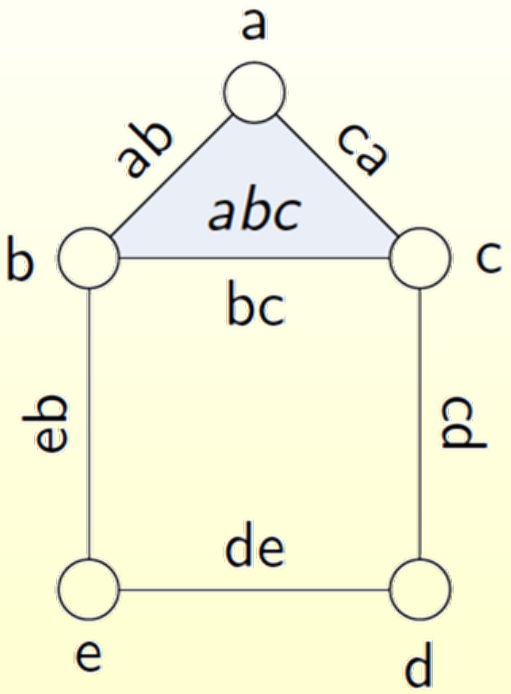
Simplicial Complexes to Chains

- List of simplices
- 0-dim (vertices)

  $\{a,b,c,d,e\}$
- 1-dim (edges)

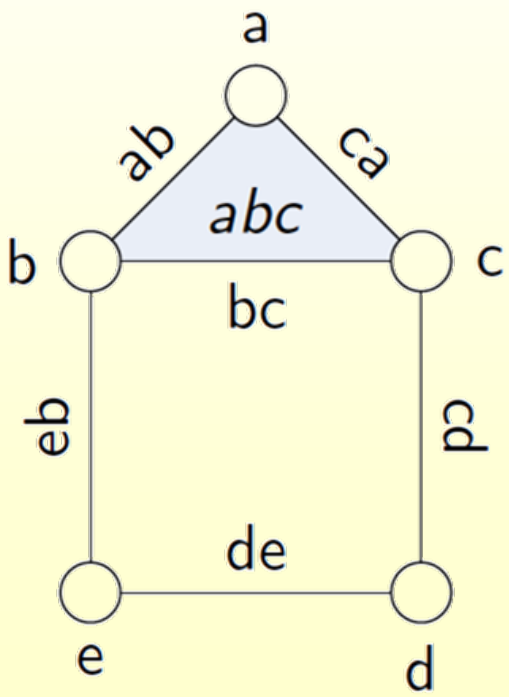
  $\{ab,ca,bc,cd,de,eb\}$
- 2-dim (triangles)

  $\{abc\}$



Boundary Operators

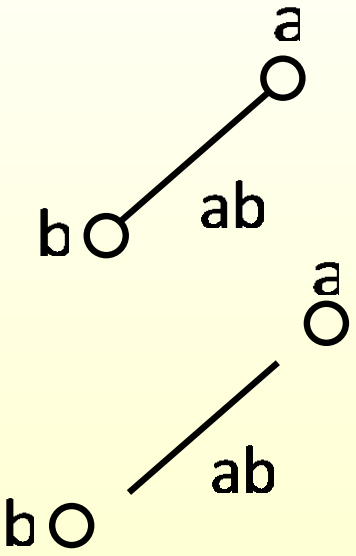
- Recall: Topology changes when we glue and cut
- Some assembly required: gluing maps



$\{a,b,c,d,e\}$

$\{ab,ca,bc,cd,de,eb\}$

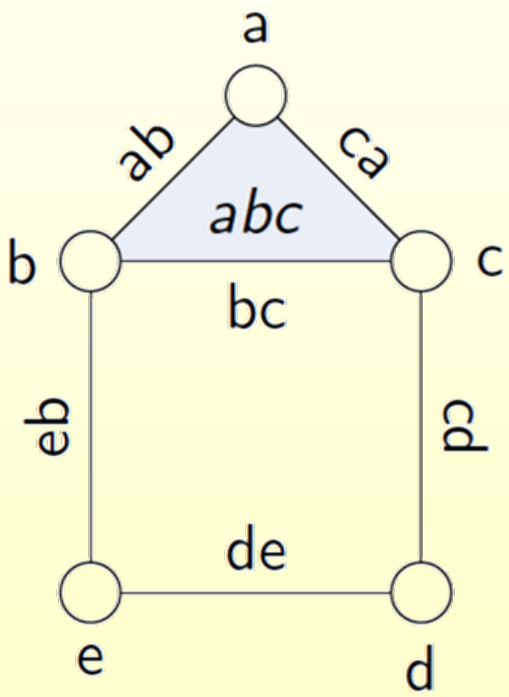
$\{abc\}$



$ab \mapsto a,b$

Boundary Operators

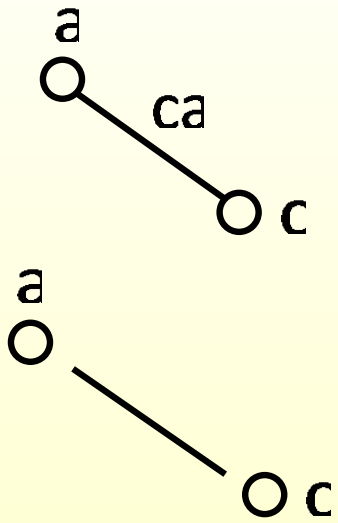
- Recall: Topology changes when we glue and cut
- Some assembly required: gluing maps



$\{a,b,c,d,e\}$

$\{ab,ca,bc,cd,de,eb\}$

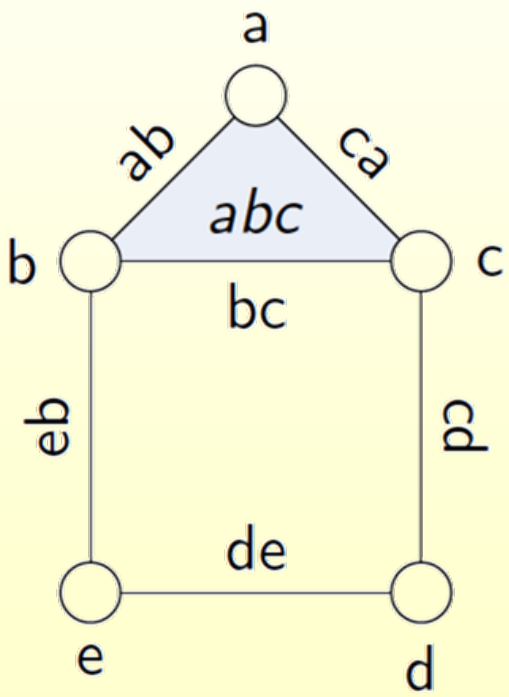
$\{abc\}$



$ca \mapsto a,c$

Boundary Operators

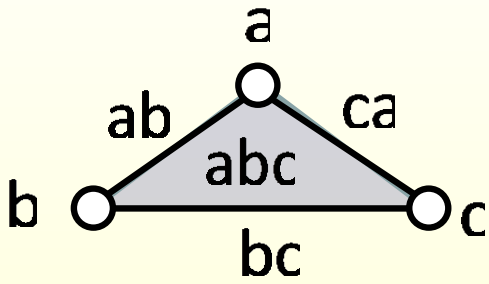
- Recall: Topology changes when we glue and cut
- Some assembly required: gluing maps

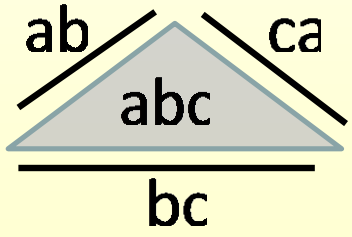


$\{a,b,c,d,e\}$

$\{ab,ca,bc,cd,de,eb\}$

$\{abc\}$

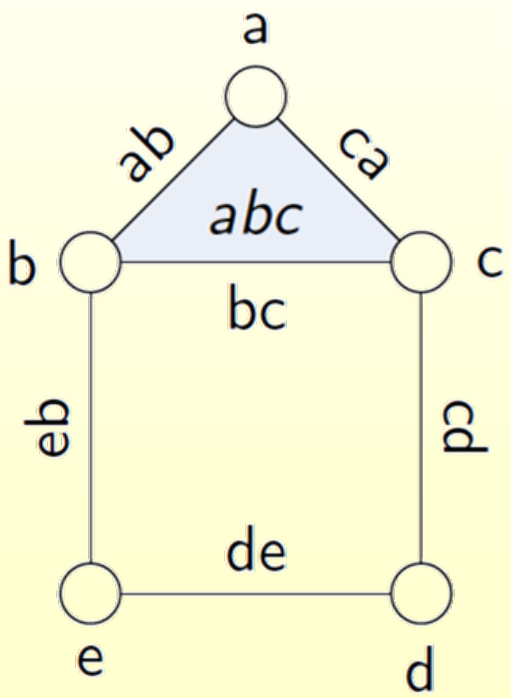




$abc \mapsto ab, ca, bc$

Boundary Operators

- Reorganize information into vectors



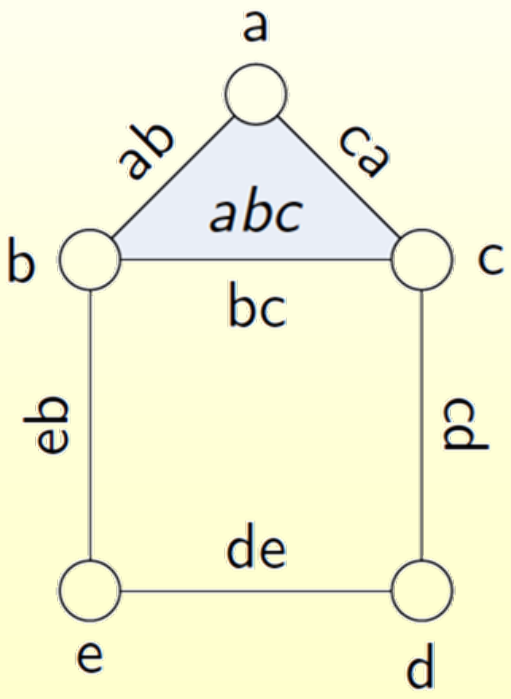
$ab \mapsto a,b$ $\begin{bmatrix} a \\ b \end{bmatrix}$

$ca \mapsto a,c$ $\begin{bmatrix} a \\ c \end{bmatrix}$

$abc \mapsto ab, ca, bc$ $\begin{bmatrix} ab \\ ac \\ bc \end{bmatrix}$

Boundary Operators

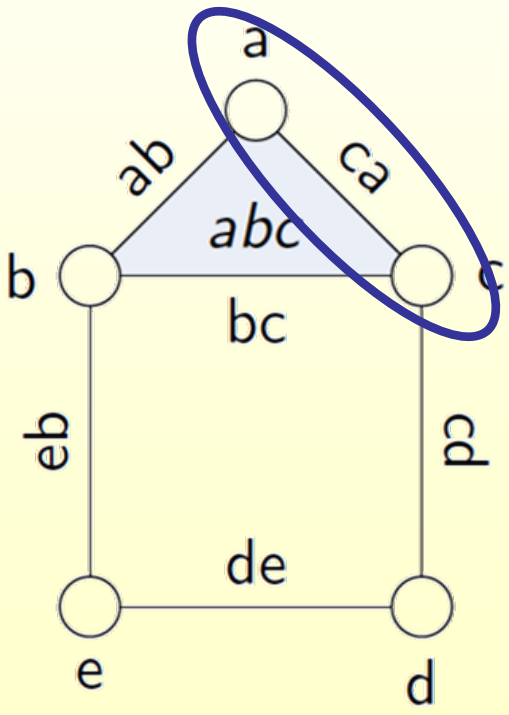
- Reorganize information into vectors
- Vector space of chains (e.g. chain groups)



$$
\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ ab \\ ca \\ bc \\ cd \\ eb \\ de \\ abc \end{array}
\begin{array}{c}
\begin{array}{ccccccc} ab & ca & bc & cd & eb & de & abc \end{array} \\
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

Boundary Operators

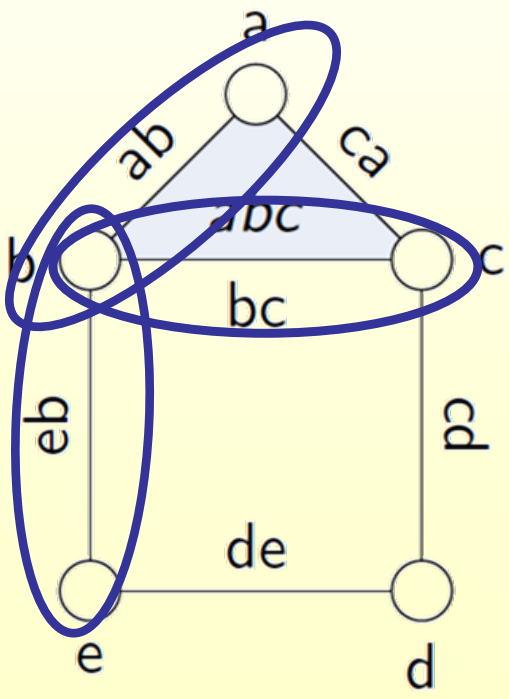
- Reorganize information into vectors
- Vector space of chains (e.g. chain groups)



$$
\begin{array}{c}
\\
a \\ b \\ c \\ d \\ e \\ ab \\ ca \\ bc \\ cd \\ eb \\ de \\ abc
\end{array}
\begin{array}{c}
\begin{array}{ccccccc} ab & ca & bc & cd & eb & de & abc \end{array} \\
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

Boundary Operators

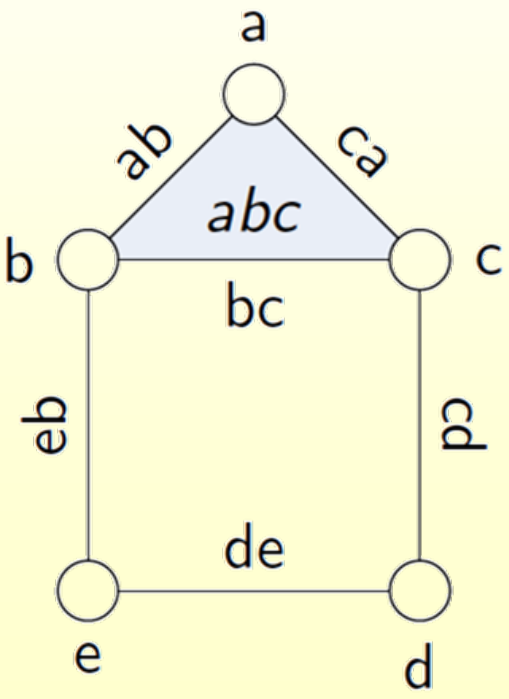
- Reorganize information into vectors
- Vector space of chains (e.g. chain groups)



$$
\begin{array}{c}
\\
a \\ b \\ c \\ d \\ e \\ ab \\ ca \\ bc \\ cd \\ eb \\ de \\ abc
\end{array}
\begin{array}{c}
\begin{array}{ccccccc} ab & ca & bc & cd & eb & de & abc \end{array} \\
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

Boundary Operators

- Reorganize information into vectors
- Vector space of chains (e.g. chain groups)

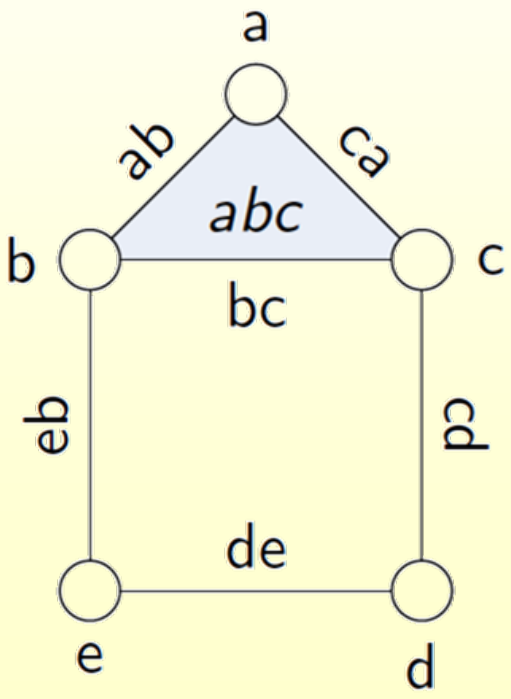


Graph adjacency matrix

	ab	ca	bc	cd	eb	de	abc
a	1	1	0	0	0	0	0
b	1	0	1	0	1	0	0
c	0	1	1	1	0	0	0
d	0	0	0	1	0	1	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0

Boundary Operators

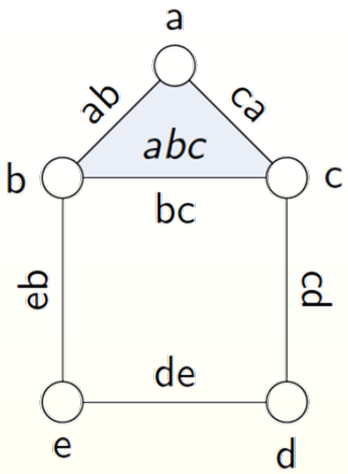
- Reorganize information into vectors
- Vector space of chains (e.g. chain groups)



∂_1

	ab	ca	bc	cd	eb	de	abc
a	1	1	0	0	0	0	0
b	1	0	1	0	1	0	0
c	0	1	1	1	0	0	0
d	0	0	0	1	0	1	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0

∂_2



Chain Complex

$$\ldots \to C_2 \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0 \to 0$$

$\{abc\}$ $\xrightarrow{\partial_2}$ $\{ab,ca,bc,cd,de,eb\}$ $\xrightarrow{\partial_1}$ $\{a,b,c,d,e\}$

$$\begin{array}{c} \\ ab \\ ca \\ bc \\ cd \\ eb \\ de \end{array} \begin{array}{c} abc \\ \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{array}$$

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} ab\ ca\ bc\ cd\ eb\ de \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

Z_2 Operations

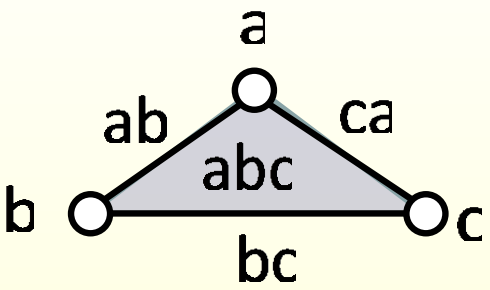
$$\begin{bmatrix}1\\0\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}1\\1\end{bmatrix}$$

$$\begin{bmatrix}1\\1\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix}$$

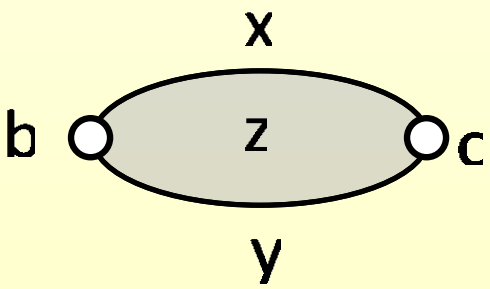
$$\begin{bmatrix}1\\1\end{bmatrix} + \begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$$

Cellular Complexes

- Simplicial complexes are easy to construct but big



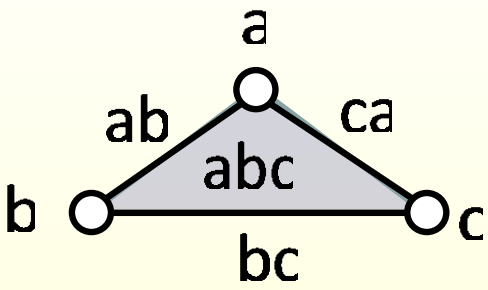
	ab	ca	bc	abc
a	1	1	0	0
b	1	0	1	0
c	0	1	1	0
ab	0	0	0	1
ca	0	0	0	1
bc	0	0	0	1



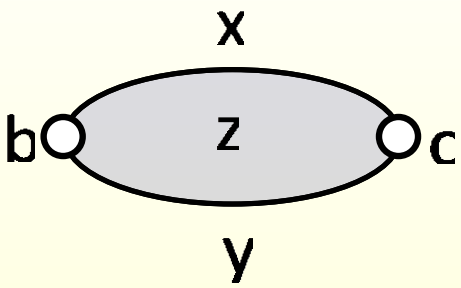
	x	y	z
b	1	1	0
c	1	1	0
x	0	0	1
y	0	0	1

Cellular Complexes

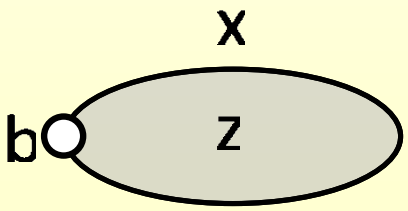
- Simplicial complexes are easy to construct but big



	ab	ca	bc	abc
a	1	1	0	0
b	1	0	1	0
c	0	1	1	0
ab	0	0	0	1
ca	0	0	0	1
bc	0	0	0	1



	x	y	z
b	1	1	0
c	1	1	0
x	0	0	1
y	0	0	1



	x	z
b	0	0
x	0	1

Spans and Bases

- Given a matrix, find a basis (a set of elements which span the space)
- A vector is spanned iff it can be written as a linear combination of elements
- Linear combinations in Z_2 are subsets
- Algorithm: Gaussian elimination

$$
\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array}
\begin{array}{c}
\begin{array}{cccccc} ab & ca & bc & cd & eb & de \end{array} \\
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1
\end{bmatrix}
\end{array}
$$

Spans and Bases

- Given a matrix, find a basis (a set of elements which span the space)
- A vector is spanned iff it can be written as a linear combination of elements
- Linear combinations in Z_2 are subsets
- Algorithm: Gaussian elimination

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Gaussian Elimination

- Perform left to right
- Check column vectors
- Don’t worry about pivots

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Gaussian Elimination

- Perform left to right
- Check column vectors
- Don’t worry about pivots

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Gaussian Elimination

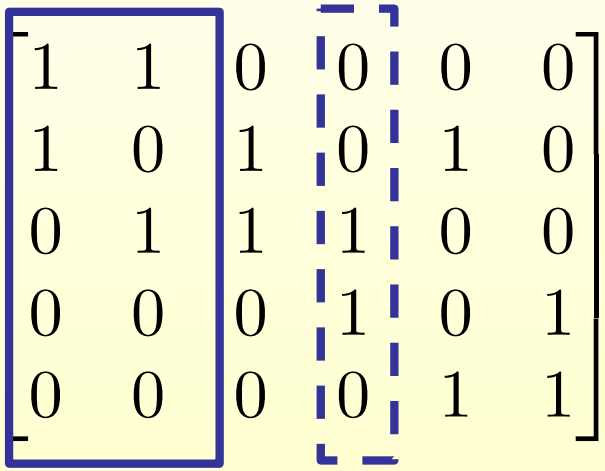
- Perform left to right
- Check column vectors
- Don’t worry about pivots

$$\left[\begin{array}{cc|c|ccc} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right]$$

Zero!

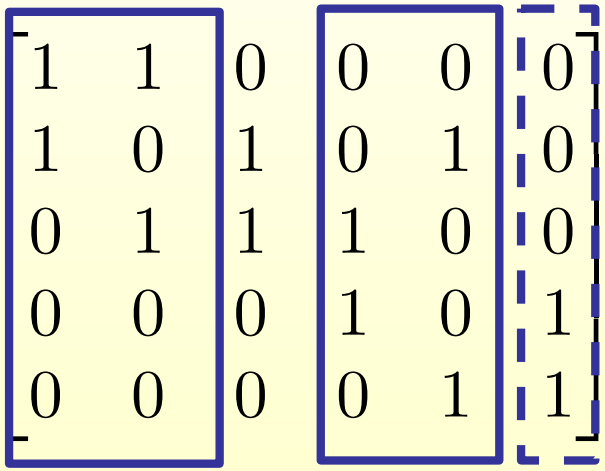
Gaussian Elimination

- Perform left to right
- Check column vectors
- Don’t worry about pivots



Gaussian Elimination

- Perform left to right
- Check column vectors
- Don’t worry about pivots

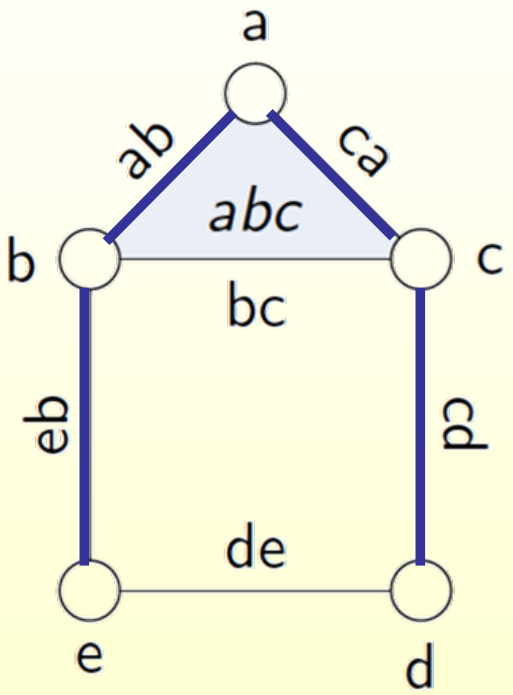


Zero!

Basis

- What does this mean?

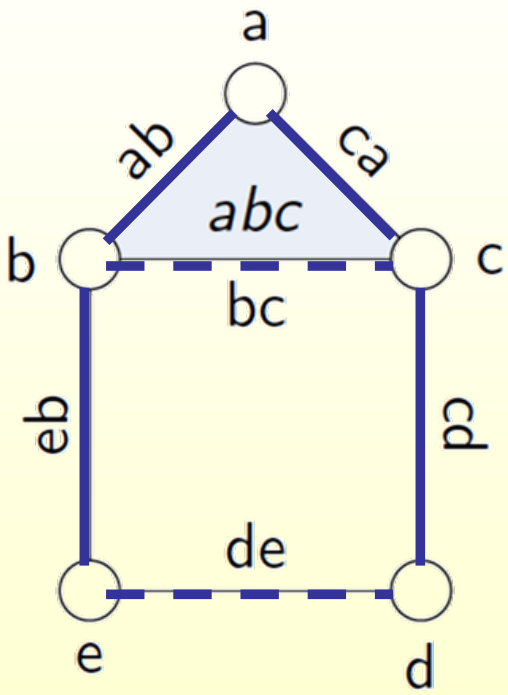
	ab	ca	bc	cd	eb	de
a	1	1	0	0	0	0
b	1	0	1	0	1	0
c	0	1	1	1	0	0
d	0	0	0	1	0	1
e	0	0	0	0	1	1



Basis

- What does this mean?
- Null space

	ab	ca	bc	cd	eb	de
a	1	1	0	0	0	0
b	1	0	1	0	1	0
c	0	1	1	1	0	0
d	0	0	0	1	0	1
e	0	0	0	0	1	1



$$bc = ab + ca$$

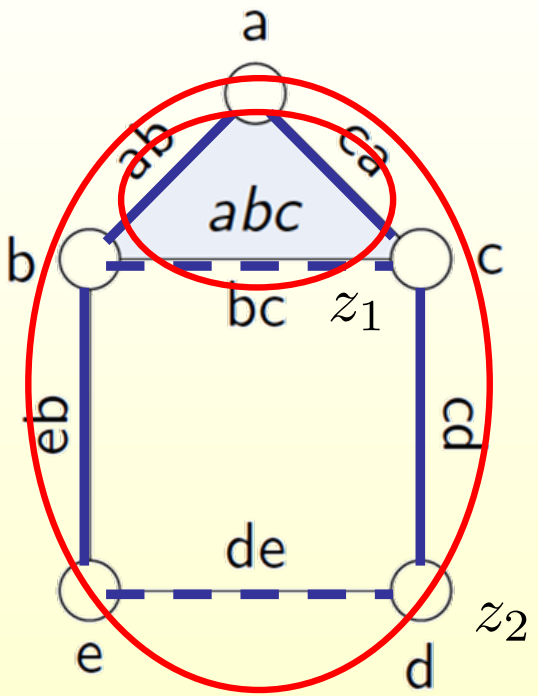
$$bc + ab + ca = 0$$

$$bc + ab + ca + cd + eb + de = 0$$

Basis

- What does this mean?
- Null space

	ab	ca	bc	cd	eb	de
a	1	1	0	0	0	0
b	1	0	1	0	1	0
c	0	1	1	1	0	0
d	0	0	0	1	0	1
e	0	0	0	0	1	1



$$\mathrm{bc} = \mathrm{ab} + \mathrm{ca}$$

$$\mathrm{bc} + \mathrm{ab} + \mathrm{ca} = 0 \quad z_1$$

$$\mathrm{bc} + \mathrm{ab} + \mathrm{ca} + \mathrm{cd} + \mathrm{eb} + \mathrm{de} = 0 \quad z_2$$

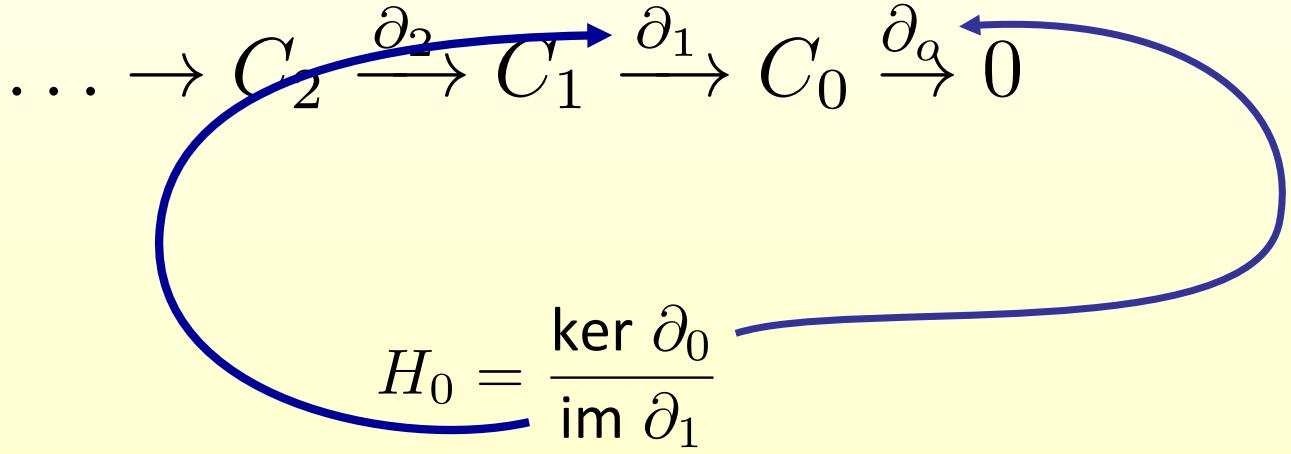
$$Z_1 = [z_1, z_2]$$

$$B_0 = [\mathrm{ab}, \mathrm{ca}, \mathrm{cd}, \mathrm{eb}]$$

0-Dim Homology

- Recall: Definition of homology

$$H_k = \frac{\ker \partial_k}{\text{im } \partial_{k+1}}$$



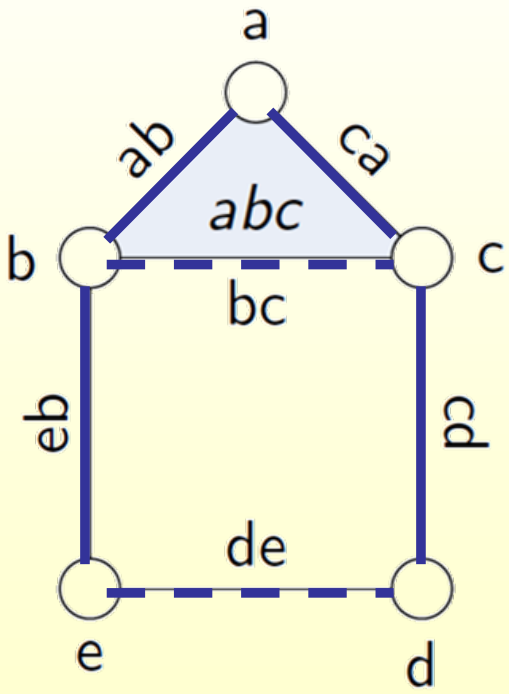
0-Dim Homology

$$H_0 = \frac{\ker \partial_0}{\text{im } \partial_1}$$

$\ker \partial_0 =$ a, b, c, d, e

$\text{im } \partial_1 =$ a+b, c+a, c+d, e+b

$\text{im } \partial_1 \subseteq \ker \partial_0$



Quotient is everything which is cycle but not a boundary

0-Dim Homology

im ∂_1 = a+b, c+a, c+d, e+b

ker ∂_0 = a, b, c, d, e

$$\text{im } \partial_1 \subseteq \ker \partial_0$$

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array}\!\!\begin{array}{c} \begin{array}{cccc} ab & ca & cd & eb \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \longrightarrow \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array}\!\!\begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

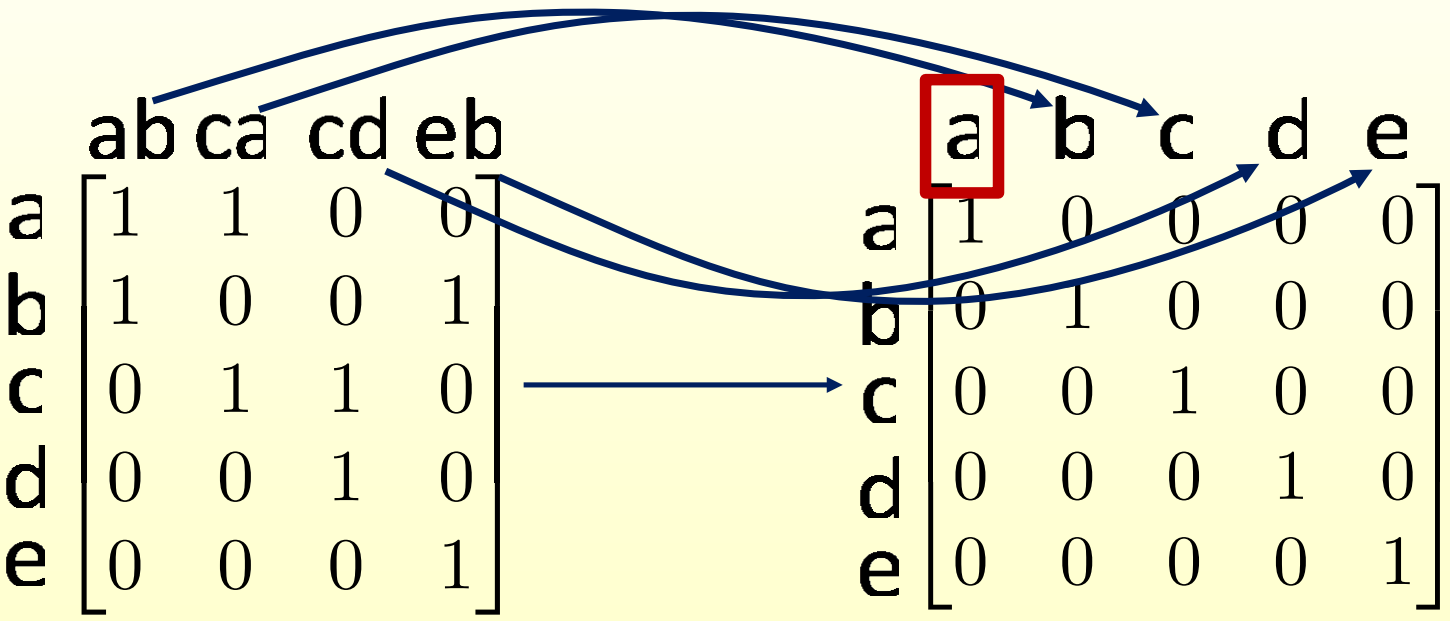
Quotient is everything which isn't a boundary

0-Dim Homology

im ∂_1 = a+b, c+a, c+d, e+b

ker ∂_0 = a, b, c, d, e

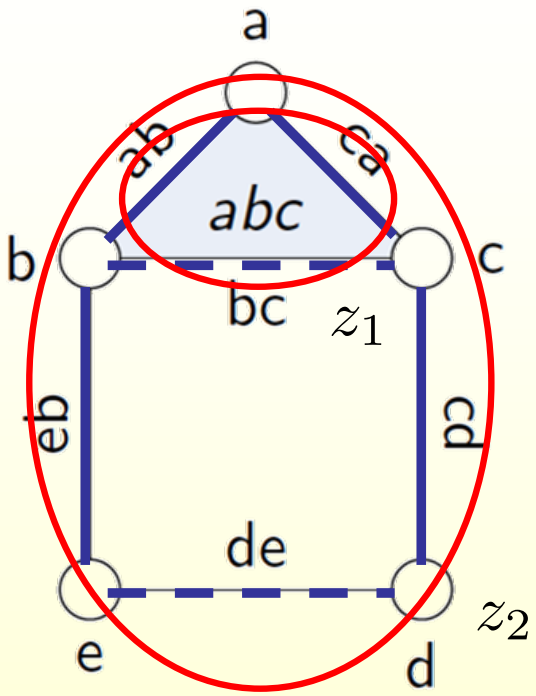
im $\partial_1 \subseteq$ ker ∂_0



Quotient is everything which isn't a boundary

1-Dim Homology

$$\partial_2 = \begin{array}{c} \\ \mathsf{ab} \\ \mathsf{ca} \\ \mathsf{bc} \\ \mathsf{cd} \\ \mathsf{eb} \\ \mathsf{de} \end{array} \begin{array}{c} \mathsf{abc} \\ \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{array} \qquad B_1 = \begin{array}{c} \\ \mathsf{ab} \\ \mathsf{ca} \\ \mathsf{bc} \\ \mathsf{cd} \\ \mathsf{eb} \\ \mathsf{de} \end{array} \begin{array}{c} \mathsf{abc} \\ \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{array}$$

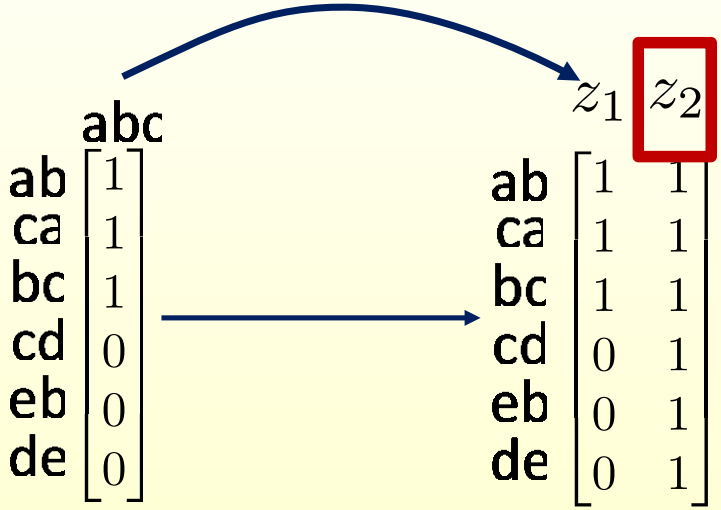


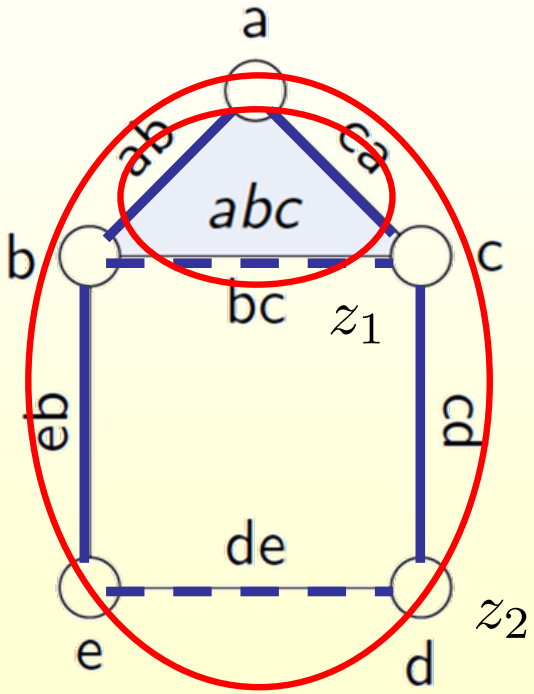
$$\mathsf{bc} + \mathsf{ab} + \mathsf{ca} = 0 \quad z_1$$

$$\mathsf{bc} + \mathsf{ab} + \mathsf{ca} + \mathsf{cd} + \mathsf{eb} + \mathsf{de} = 0 \quad z_2$$

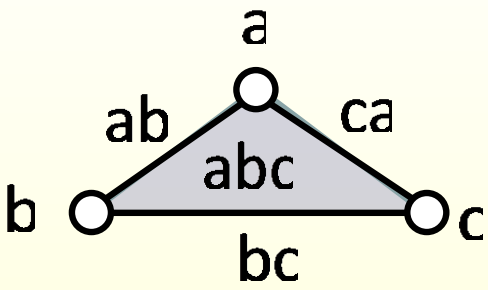
$$Z_1 = [z_1, z_2] \qquad Z_1 = \begin{array}{c} \\ \mathsf{ab} \\ \mathsf{ca} \\ \mathsf{bc} \\ \mathsf{cd} \\ \mathsf{eb} \\ \mathsf{de} \end{array} \begin{array}{c} z_1 \ z_2 \\ \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \end{array}$$

1-Dim Homology

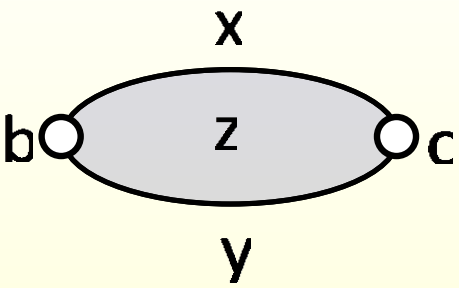




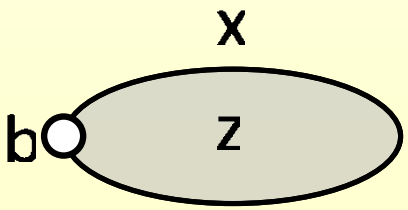
Simpler Example



$$\begin{array}{c} \\ a \\ b \\ c \\ ab \\ ca \\ bc \end{array} \begin{array}{c} \begin{array}{cccc} ab & ca & bc & abc \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$



$$\begin{array}{c} \\ b \\ c \\ x \\ y \end{array} \begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \end{array}$$



$$\begin{array}{c} \\ b \\ x \end{array} \begin{array}{c} \begin{array}{cc} x & z \end{array} \\ \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \end{array}$$

cycles: b, x

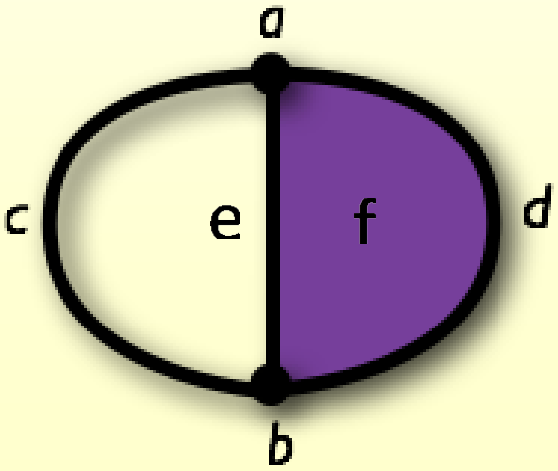
boundaries: $z \mapsto x$

Homology

- Each simplex/cell maps into a cycle

$$\partial_k \partial_{k+1} = 0$$

- Homology is obvious if we have the “right” basis
- Boundaries and cycles should be compatible



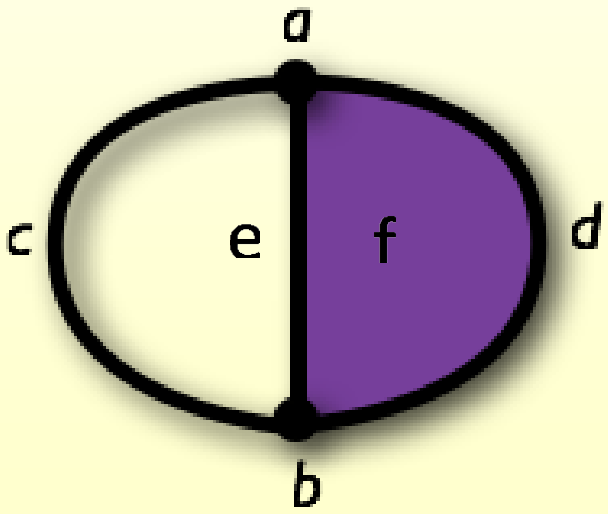
Algorithm

1. Find basis of nullspace of ∂_k (kernel)
2. Find basis of range of ∂_{k+1} (image)
3. Compute map between image and kernel
4. Diagonalize this map (make it one-to-one)

Every step is a form of Gaussian elimination

Example

1. Find basis of nullspace of ∂_k (kernel)
2. Find basis of range of ∂_{k+1} (image)
3. Compute map between image and kernel
4. Diagonalize this map (make it one-to-one)



kernel = 0 dim: a,b

1 dim: c+d, e+c, d+e

image = 0 dim: a+b

1 dim: d+e

$a+b \mapsto b$

$d+e \mapsto d+e$

Possible bases: (a, c+d)

(a, e+c)

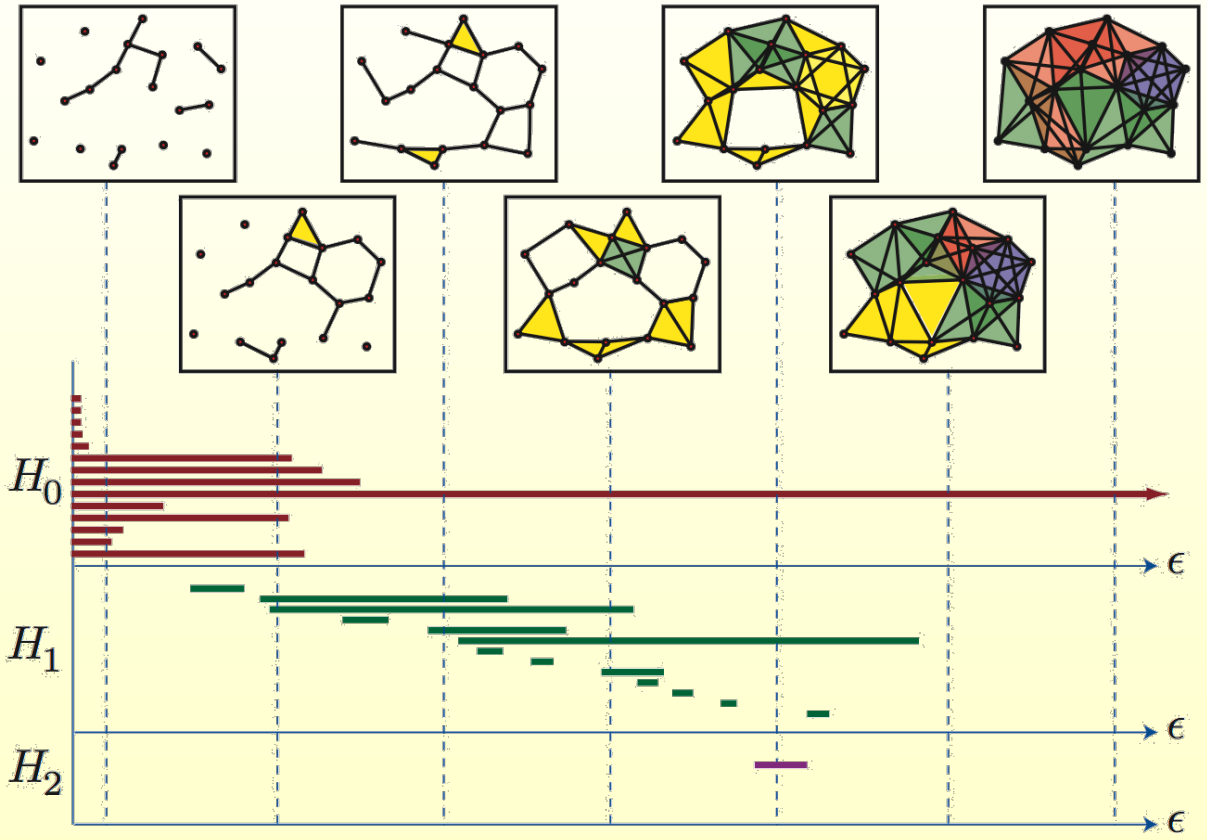
(b, e+c)

(a, c+d)

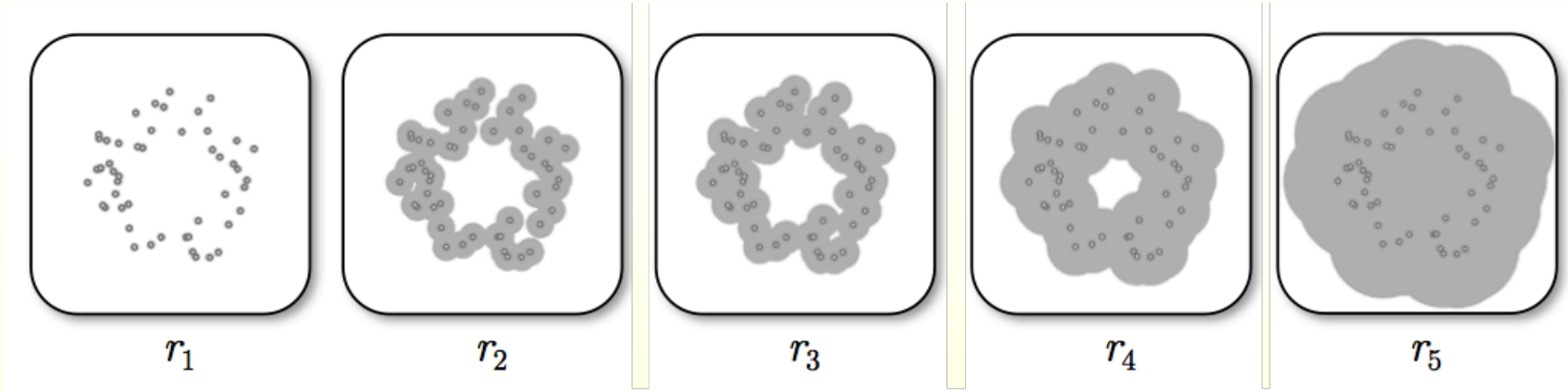
A Few Comments

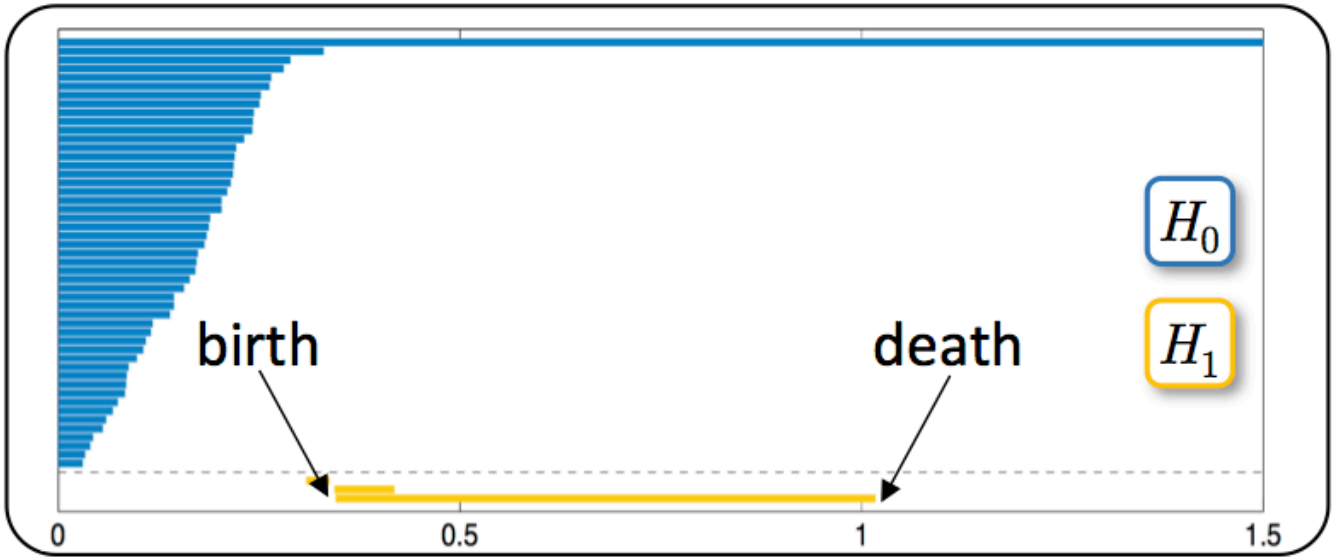
- Combinations of cycles are still cycles
- Homology basis is not unique (we can choose many representatives)
- Think in terms of spans of vectors
- Persistent homology – we can have unique representatives

Persistent Homology

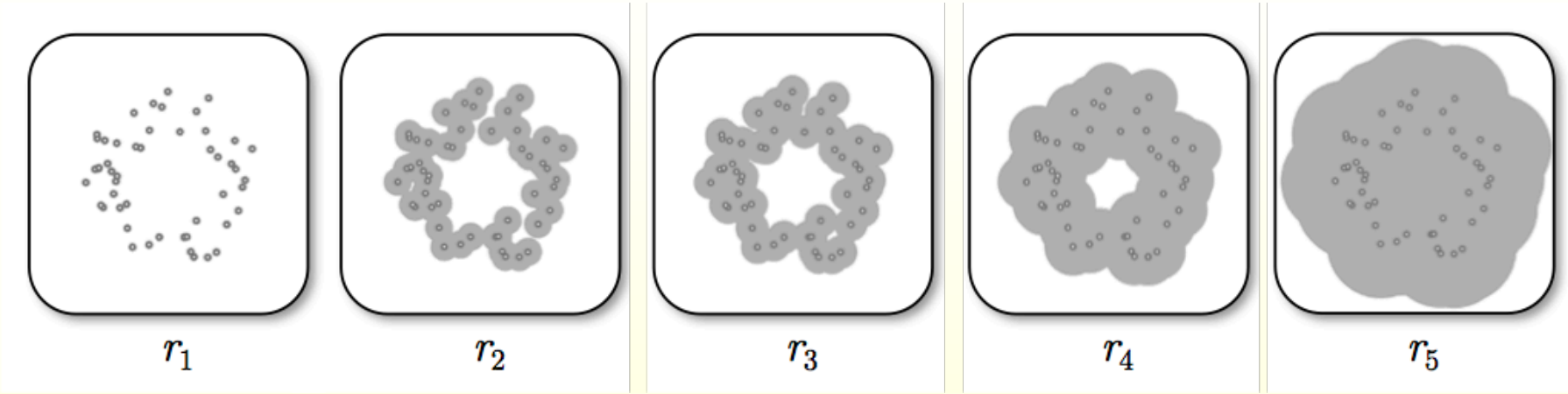


Filtrations





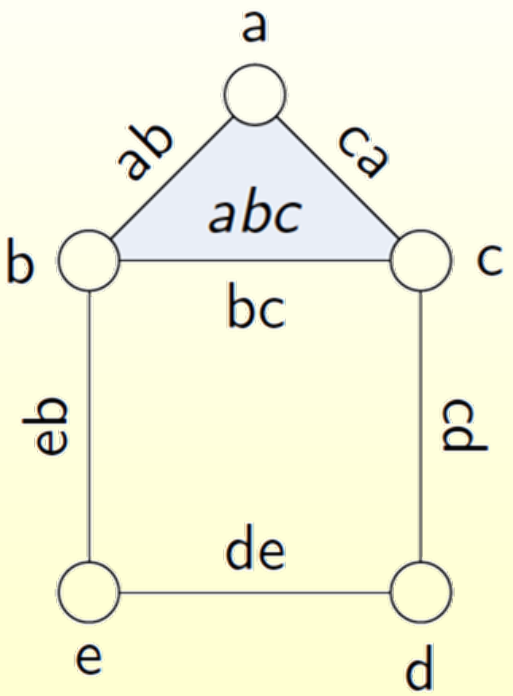
Filtrations



$$X_1 \subseteq X_2 \subseteq X_3 \subseteq X_4 \subseteq X_5$$

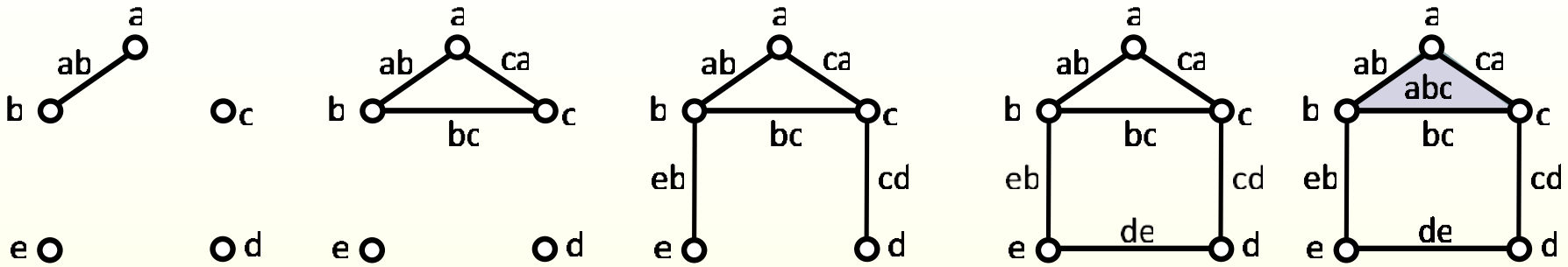
$$\partial(X_1) \subseteq \partial(X_2) \subseteq \partial(X_3) \subseteq \partial(X_4) \subseteq \partial(X_5)$$

Boundary Operators



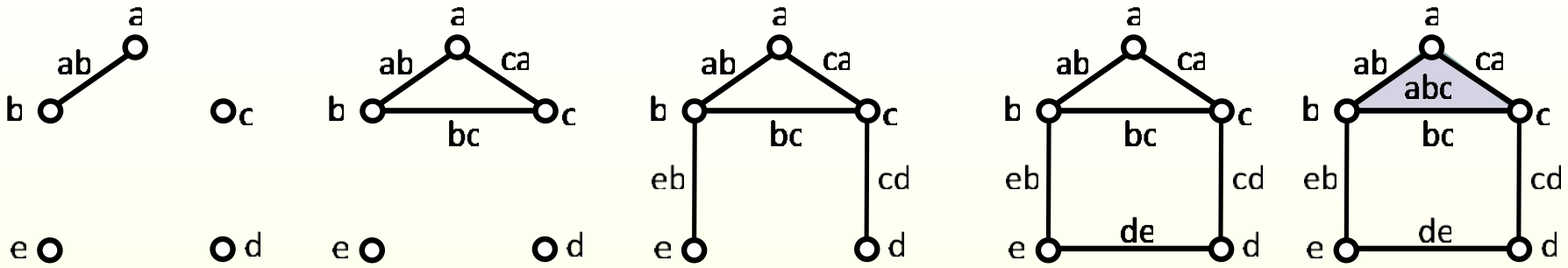
$$
\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ ab \\ ca \\ bc \\ cd \\ eb \\ de \\ abc \end{array}
\begin{array}{c}
\begin{array}{ccccccc} ab & ca & bc & cd & eb & de & abc \end{array} \\
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

Boundary Operators



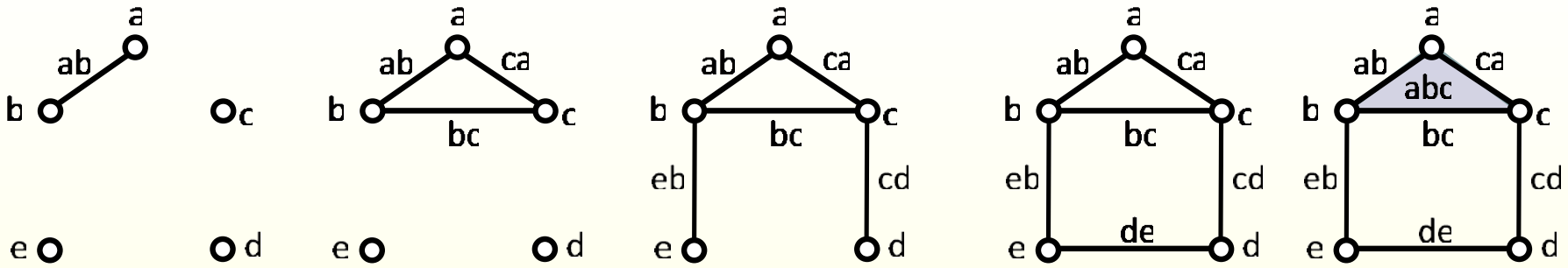
	ab	ca	bc	cd	eb	de	abc
a	1	1	0	0	0	0	0
b	1	0	1	0	1	0	0
c	0	1	1	1	0	0	0
d	0	0	0	1	0	1	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0

Boundary Operators



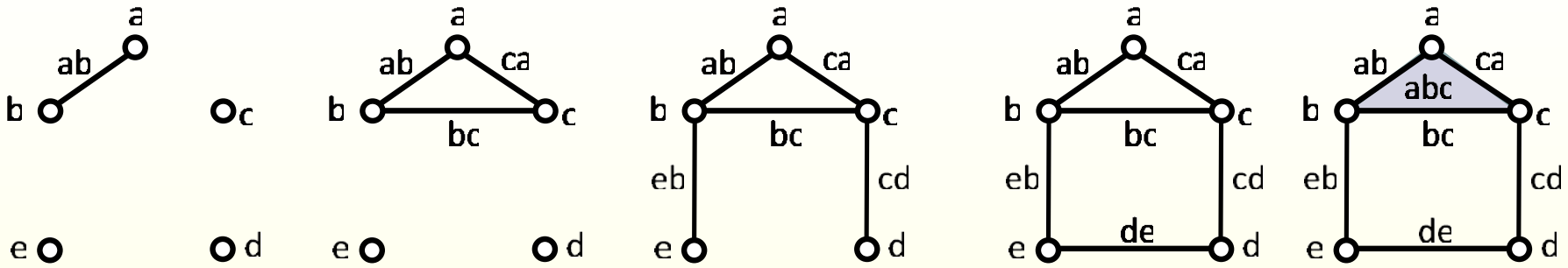
	ab	ca	bc	cd	eb	de	abc
a	1	1	0	0	0	0	0
b	1	0	1	0	1	0	0
c	0	1	1	1	0	0	0
d	0	0	0	1	0	1	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0

Boundary Operators



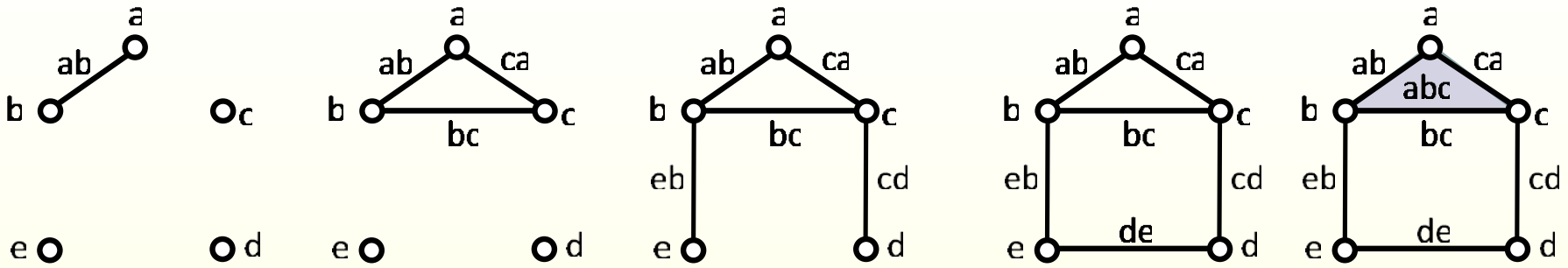
	ab	ca	bc	cd	eb	de	abc
a	1	1	0	0	0	0	0
b	1	0	1	0	1	0	0
c	0	1	1	1	0	0	0
d	0	0	0	1	0	1	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0

Boundary Operators



	ab	ca	bc	cd	eb	de	abc
a	1	1	0	0	0	0	0
b	1	0	1	0	1	0	0
c	0	1	1	1	0	0	0
d	0	0	0	1	0	1	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0

Boundary Operators

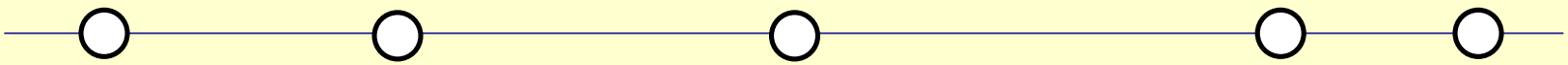


	ab	ca	bc	cd	eb	de	abc
a	1	1	0	0	0	0	0
b	1	0	1	0	1	0	0
c	0	1	1	1	0	0	0
d	0	0	0	1	0	1	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0

Tracking Features

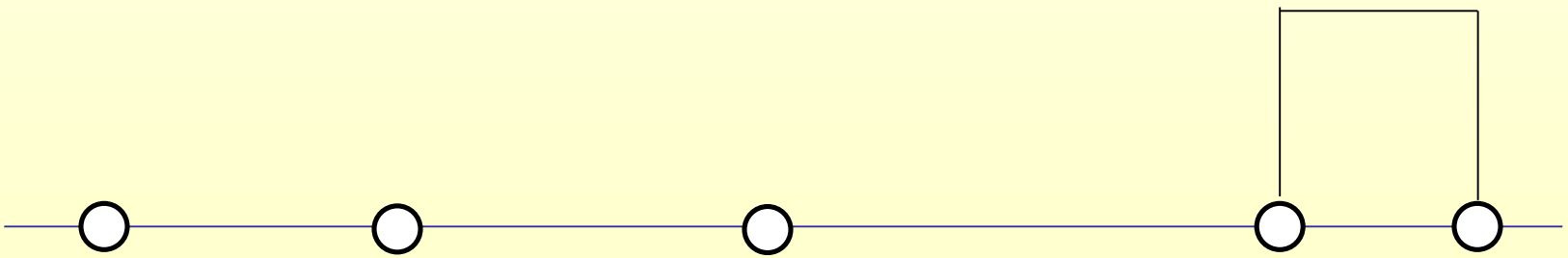
0-Dimensional Persistence

- Single Linkage Clustering



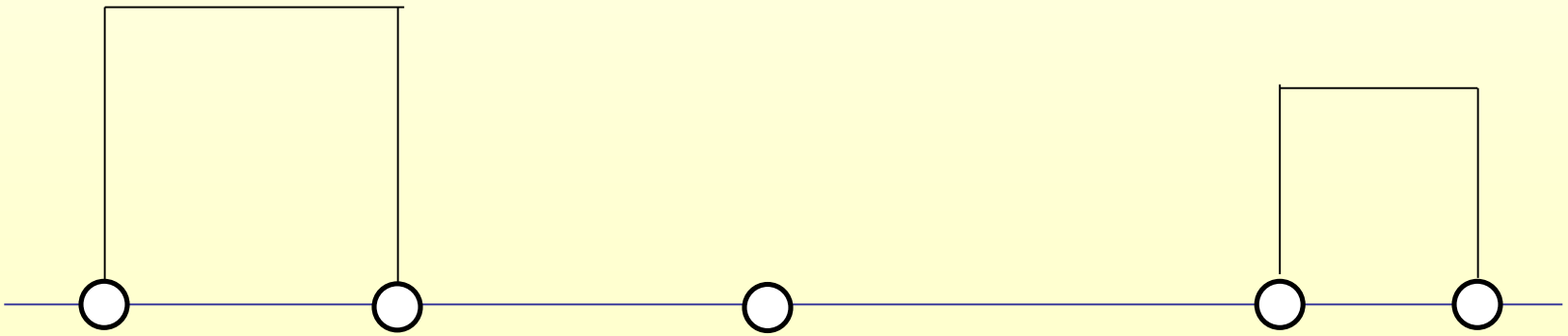
0-Dimensional Persistence

- Single Linkage Clustering



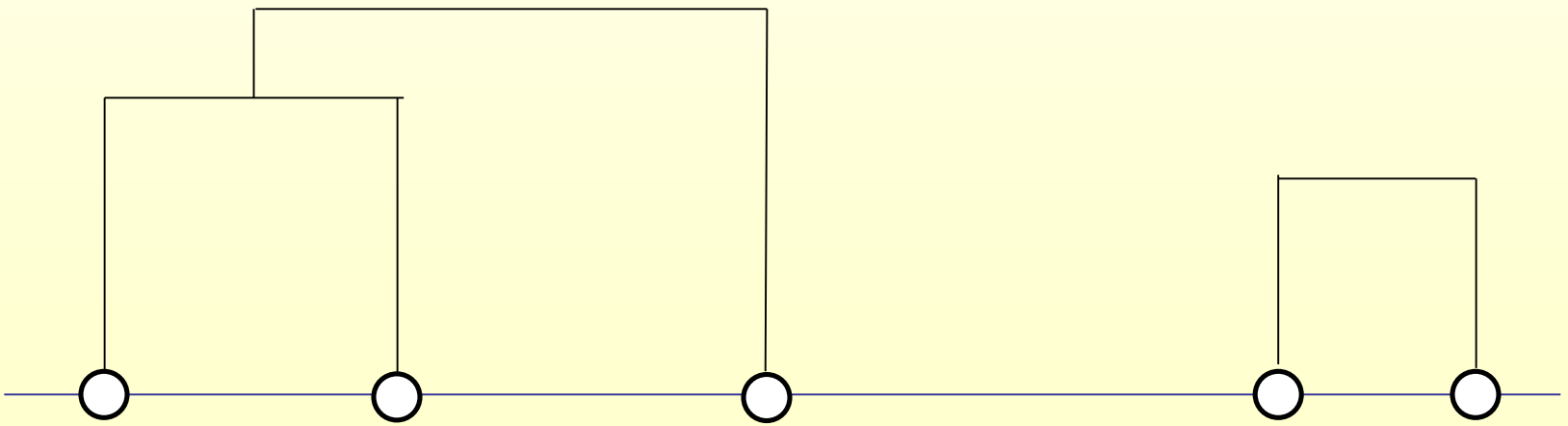
0-Dimensional Persistence

- Single Linkage Clustering



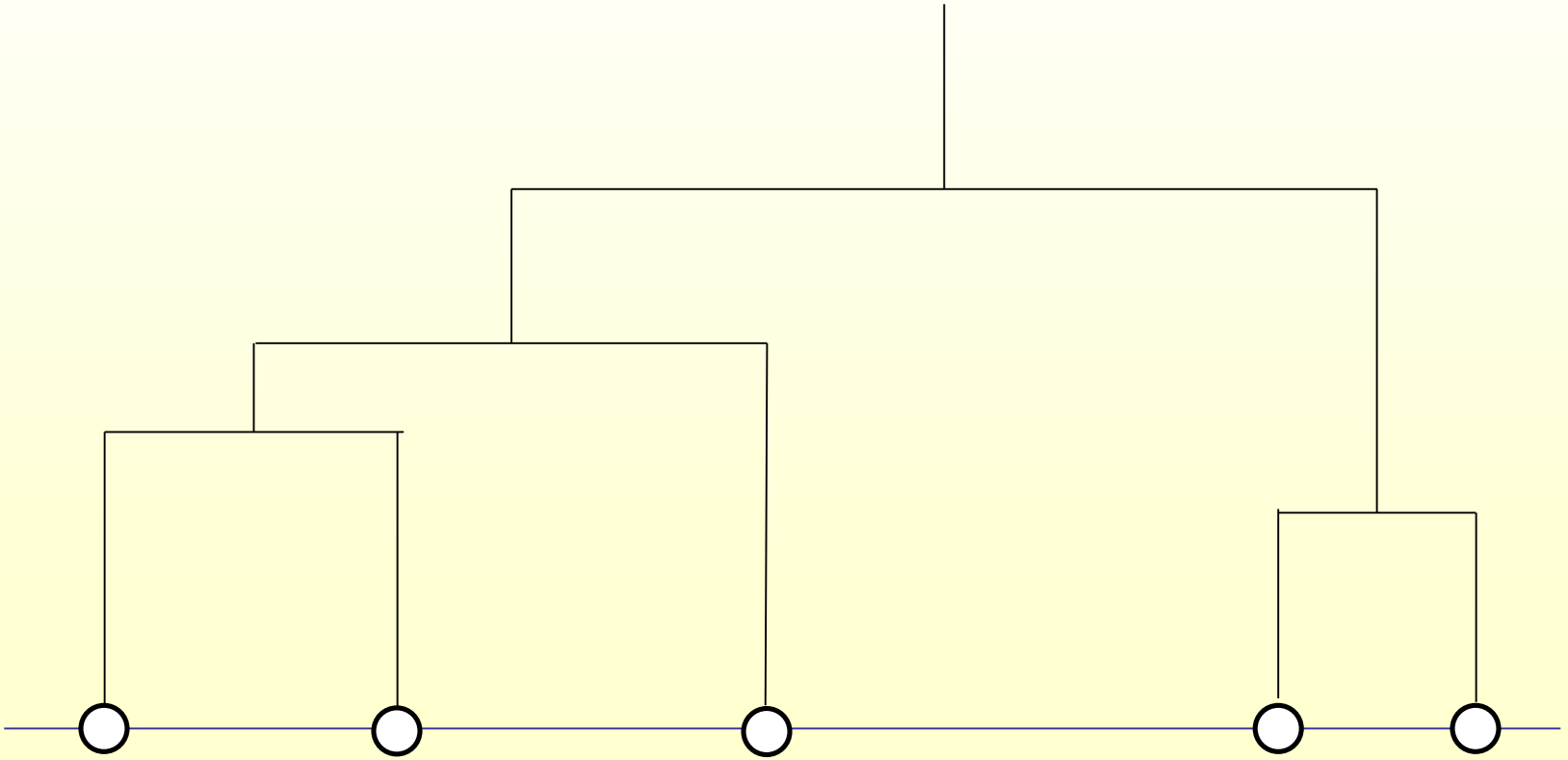
0-Dimensional Persistence

- Single Linkage Clustering



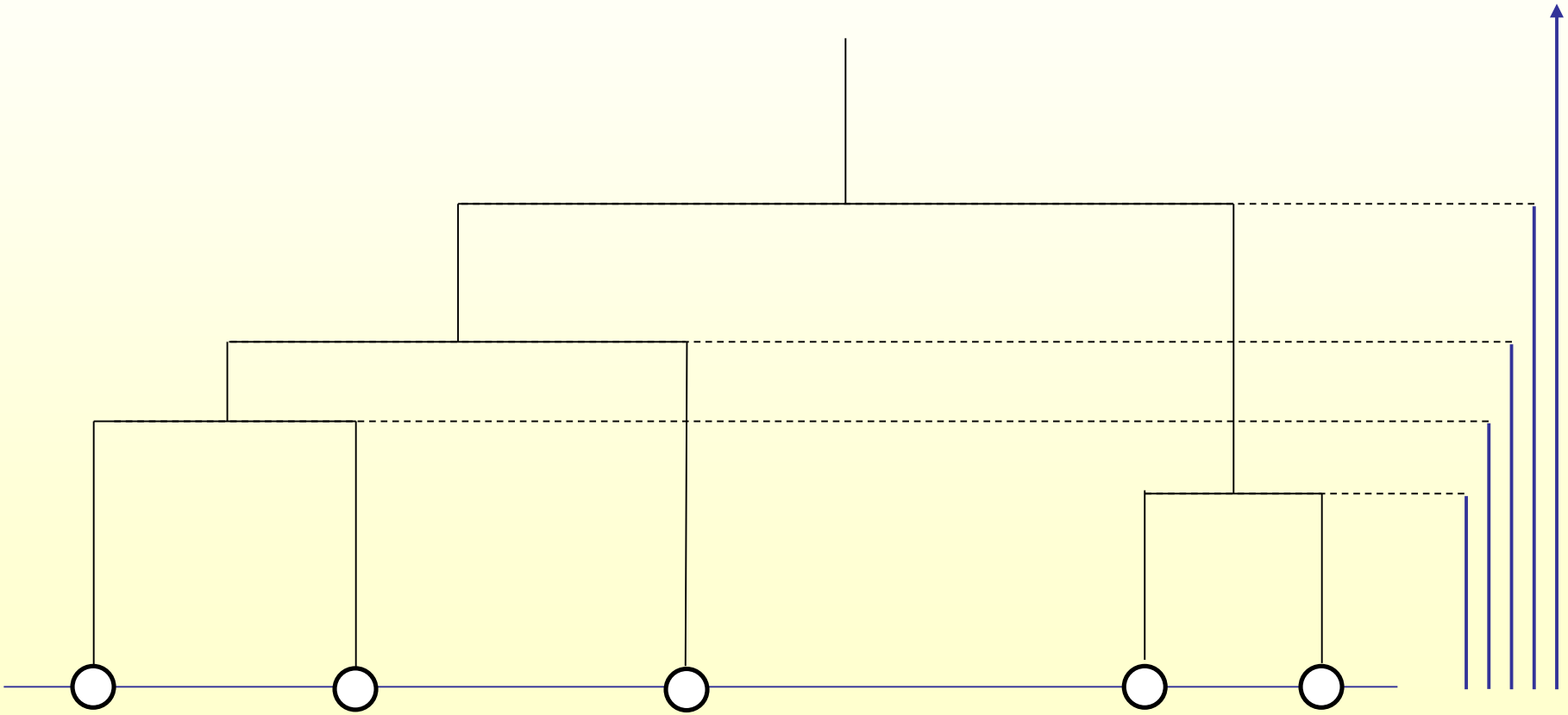
0-Dimensional Persistence

- Single Linkage Clustering



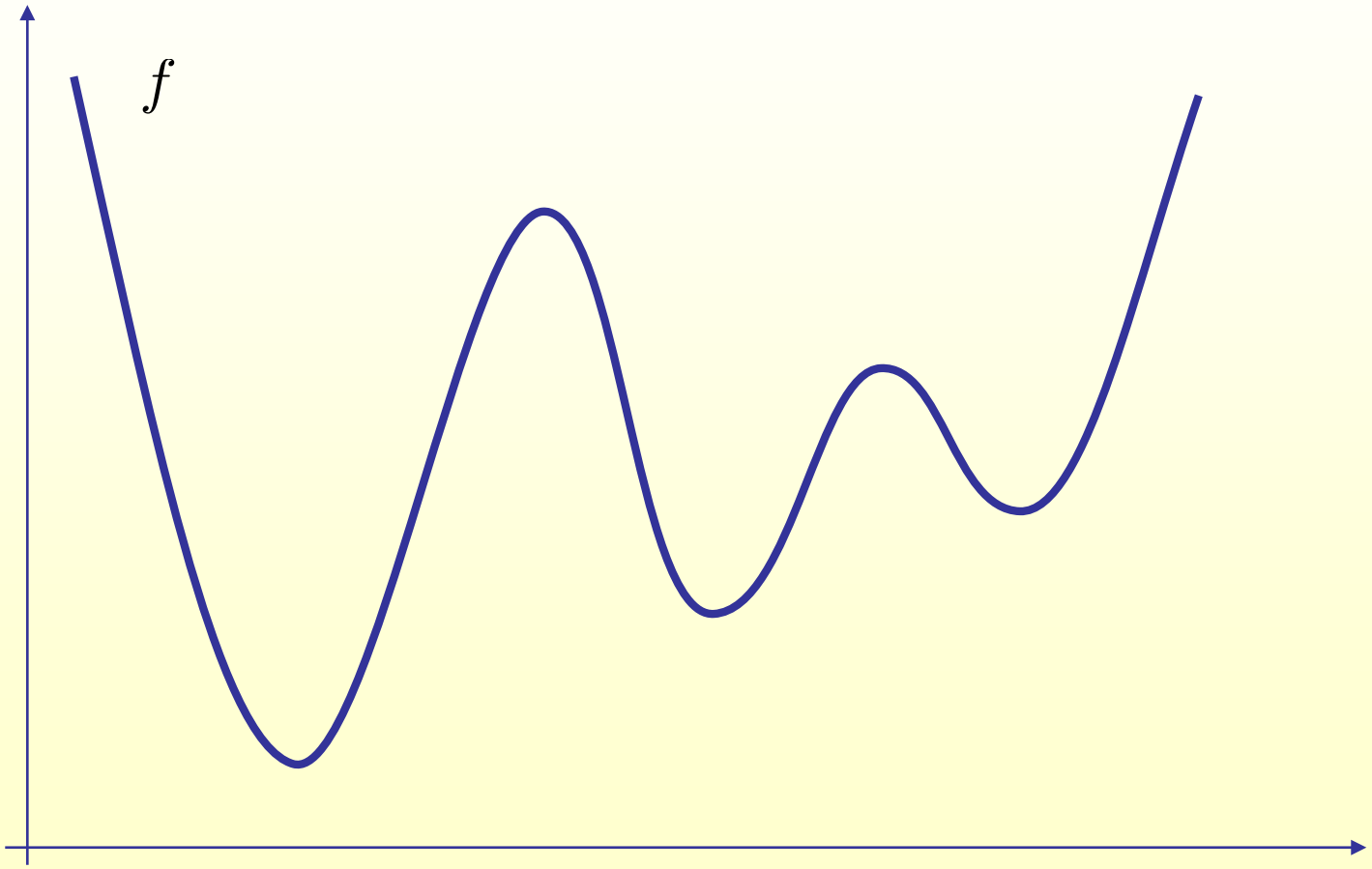
Barcodes

- Single Linkage Clustering

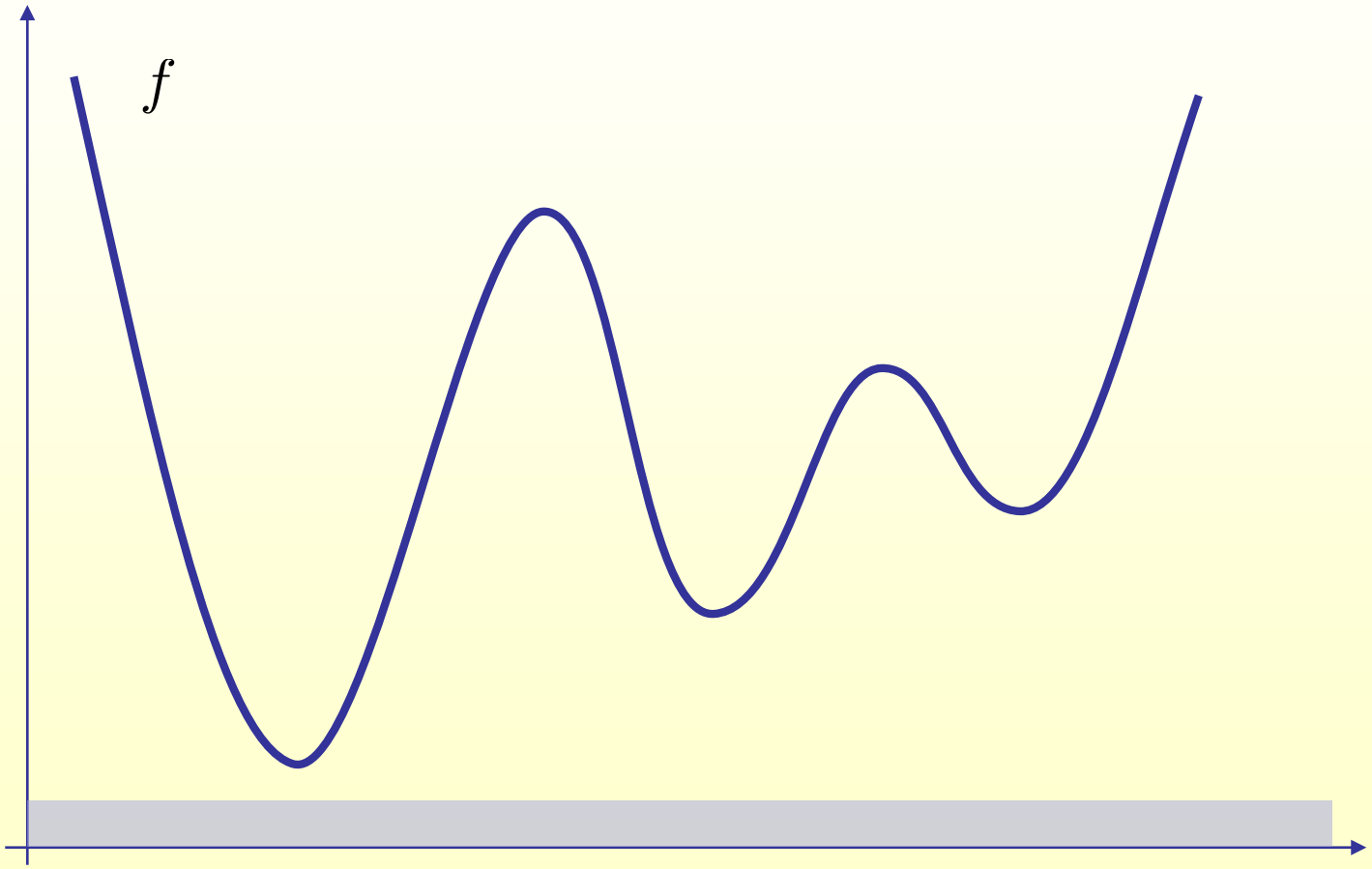


Let’s Generalize

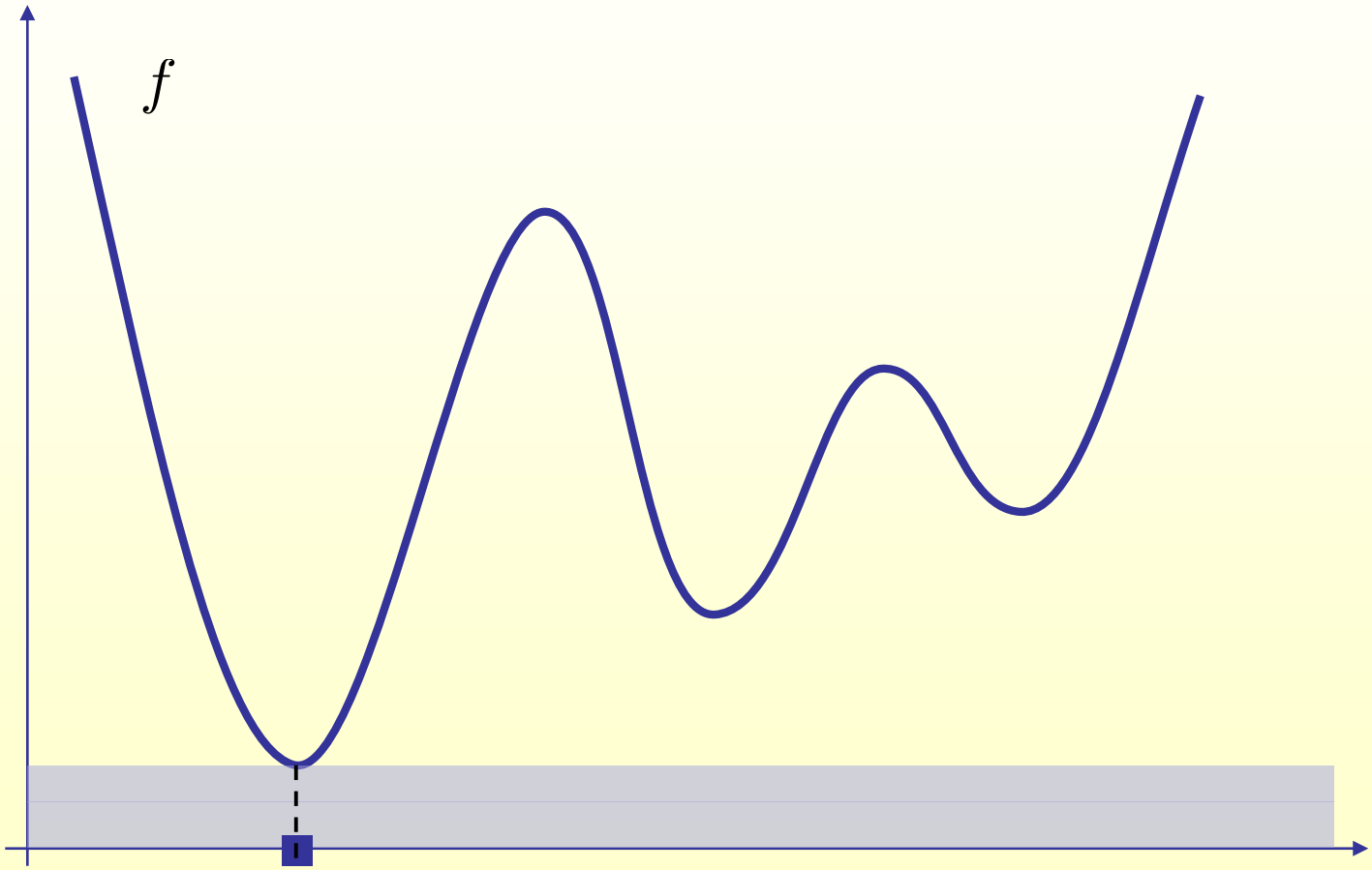
Sublevel Set Persistence



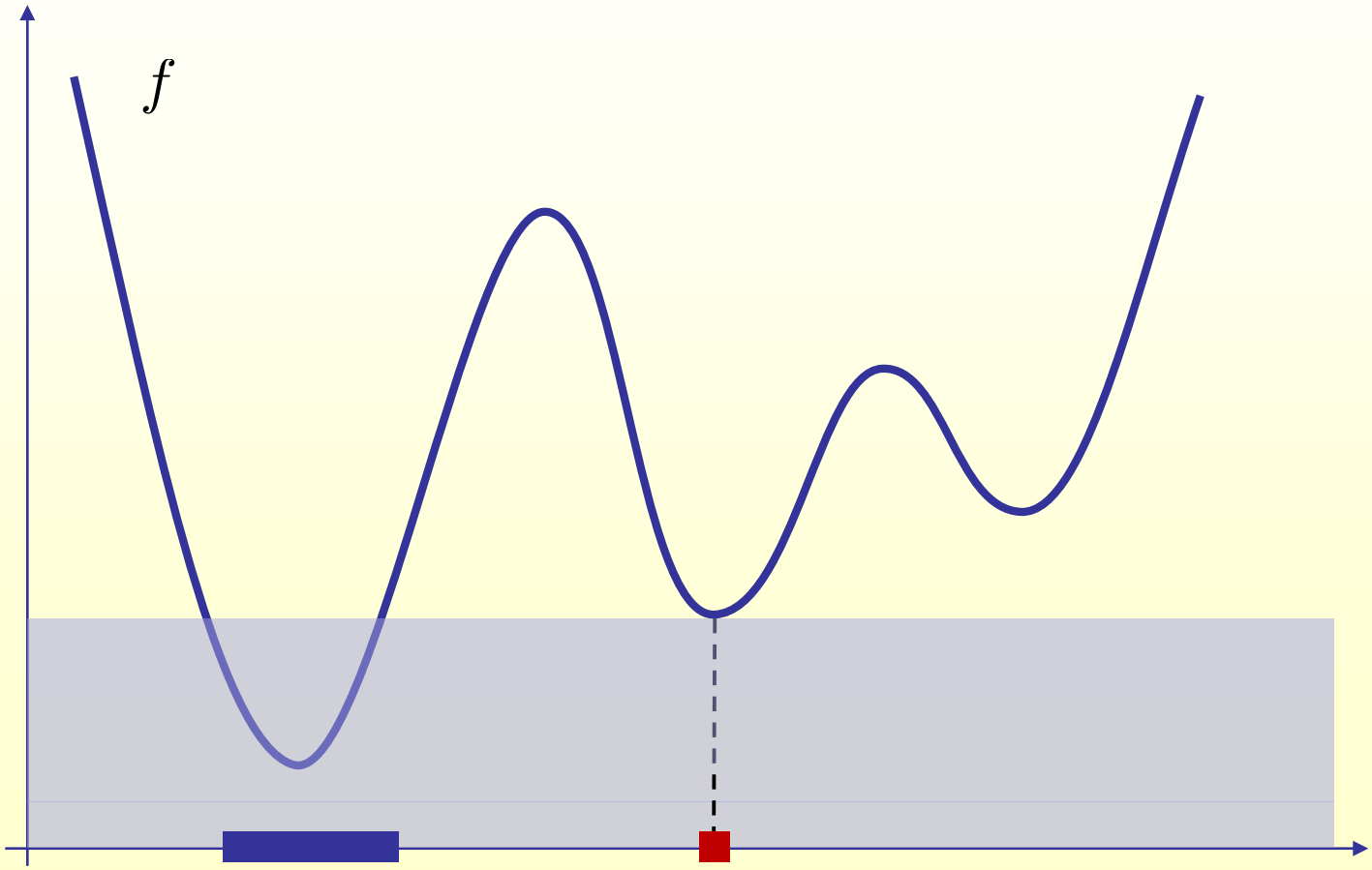
Sublevel Set Persistence



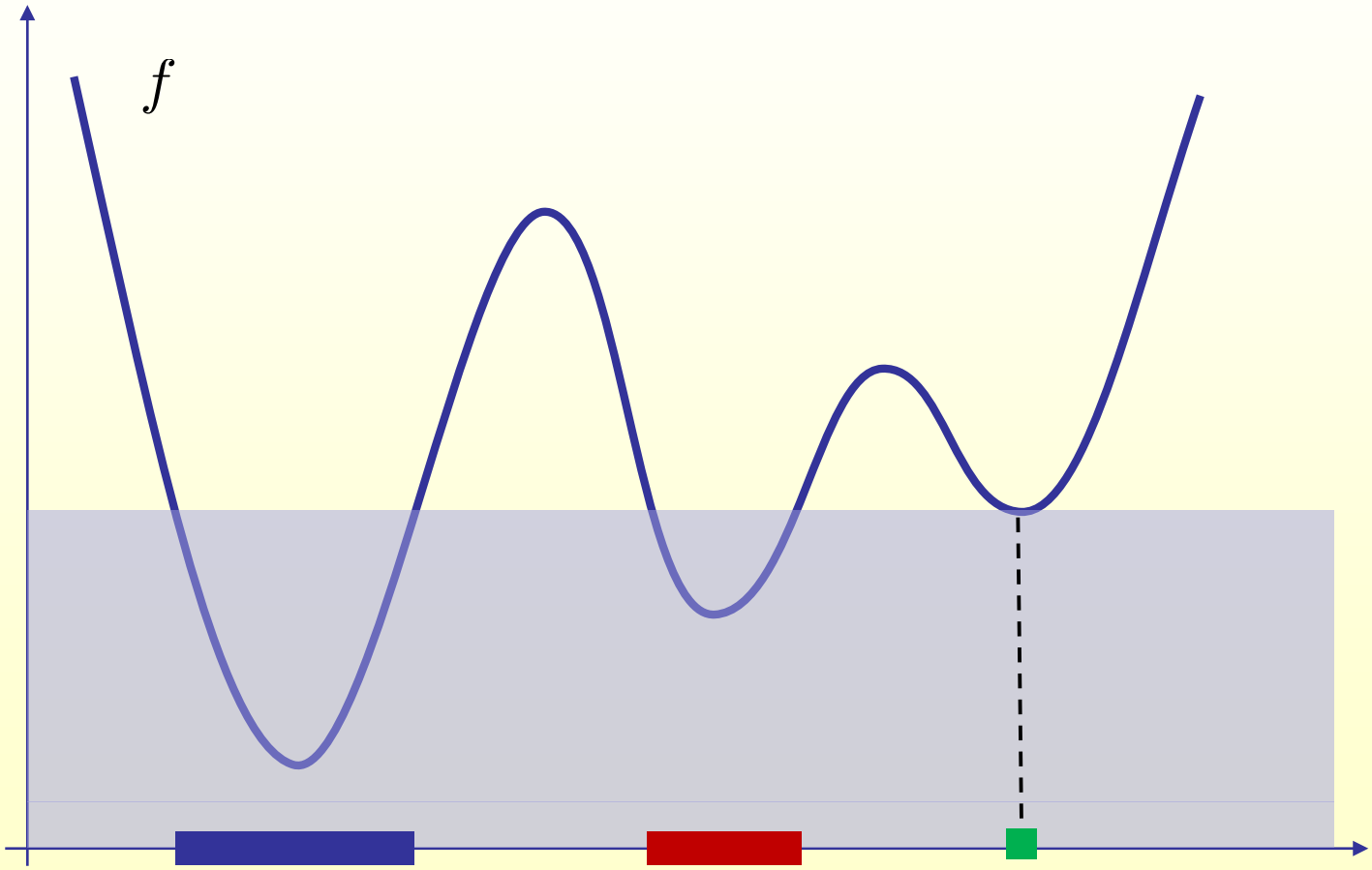
Sublevel Set Persistence



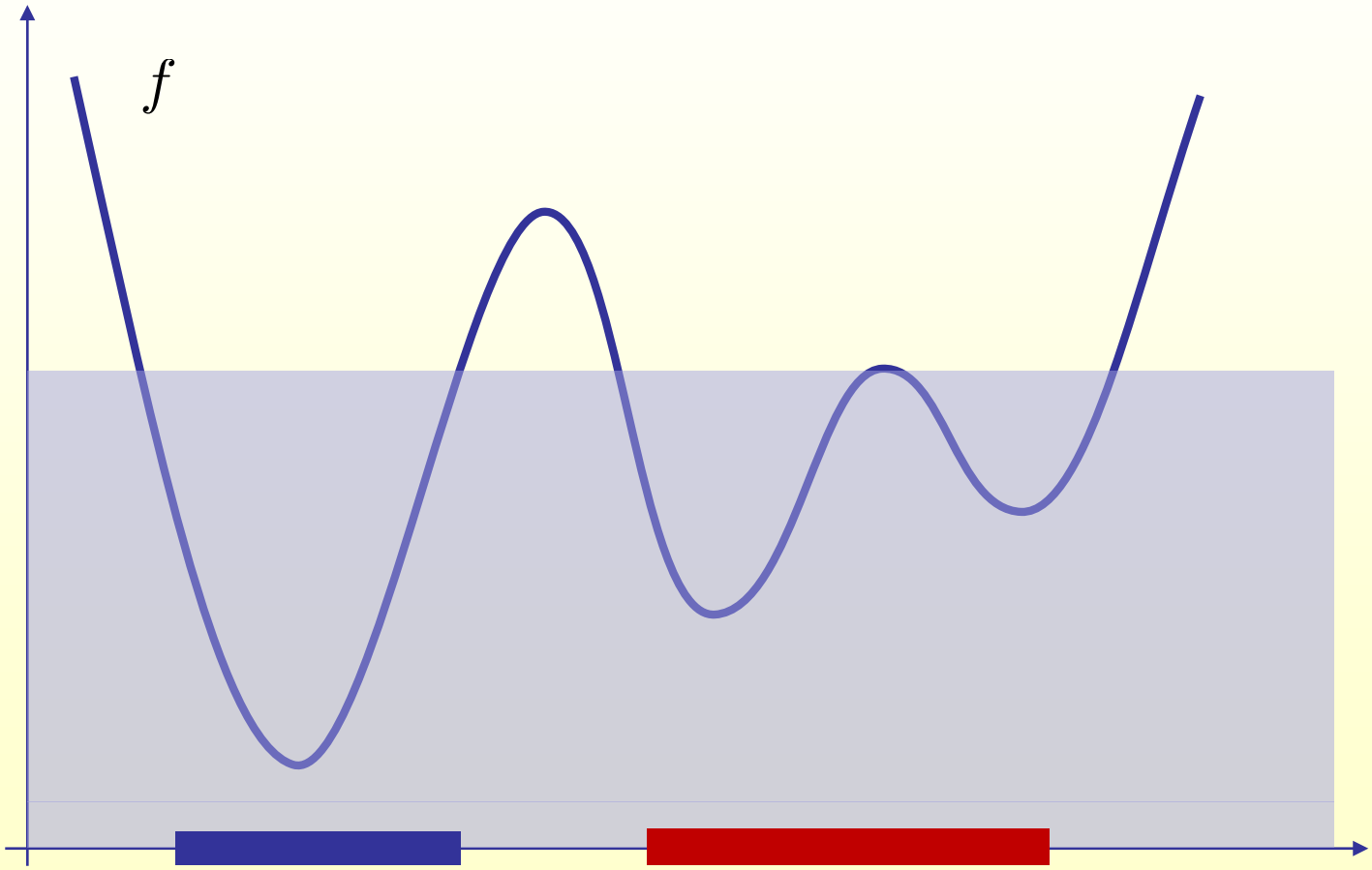
Sublevel Set Persistence



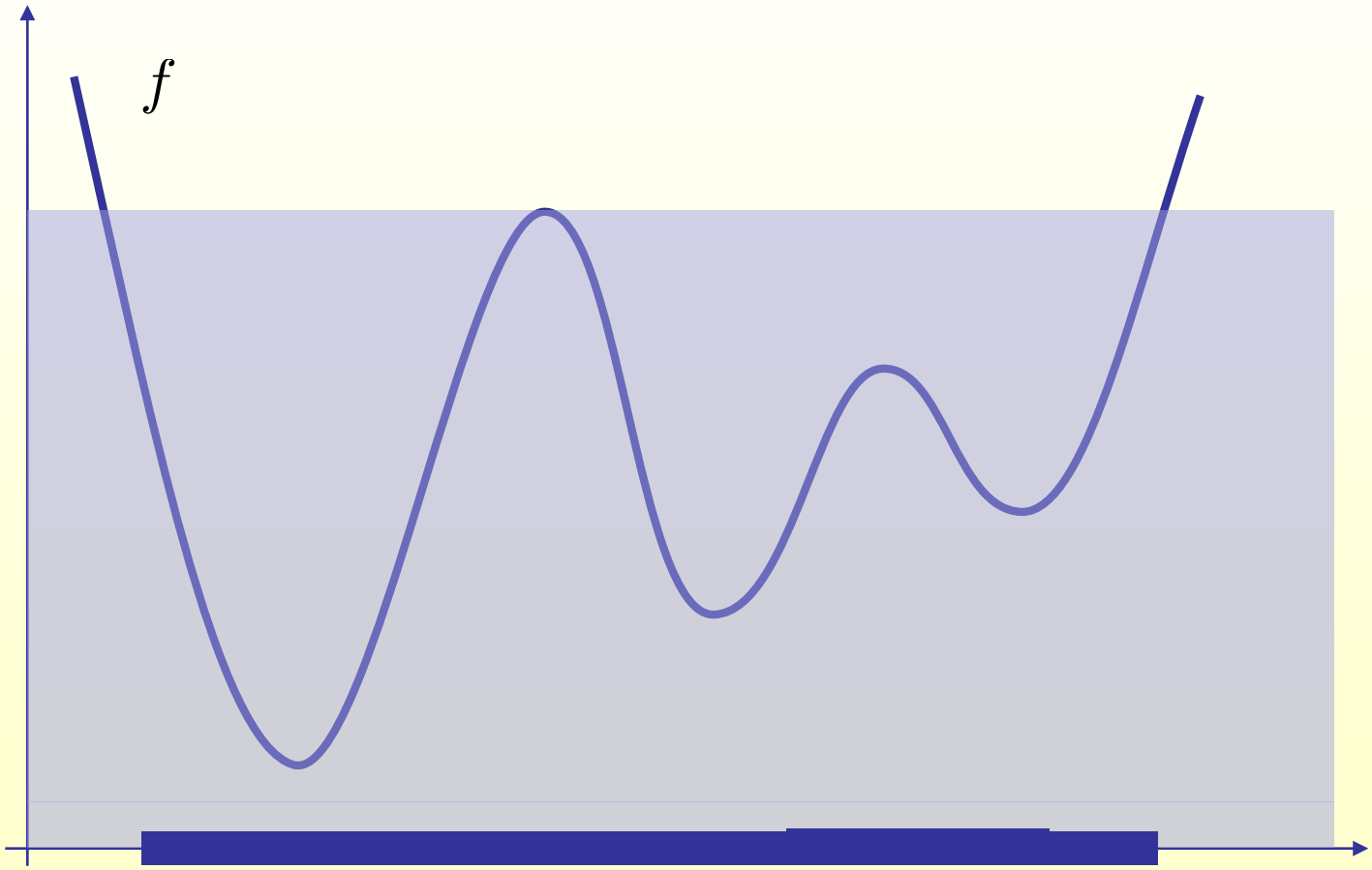
Sublevel Set Persistence



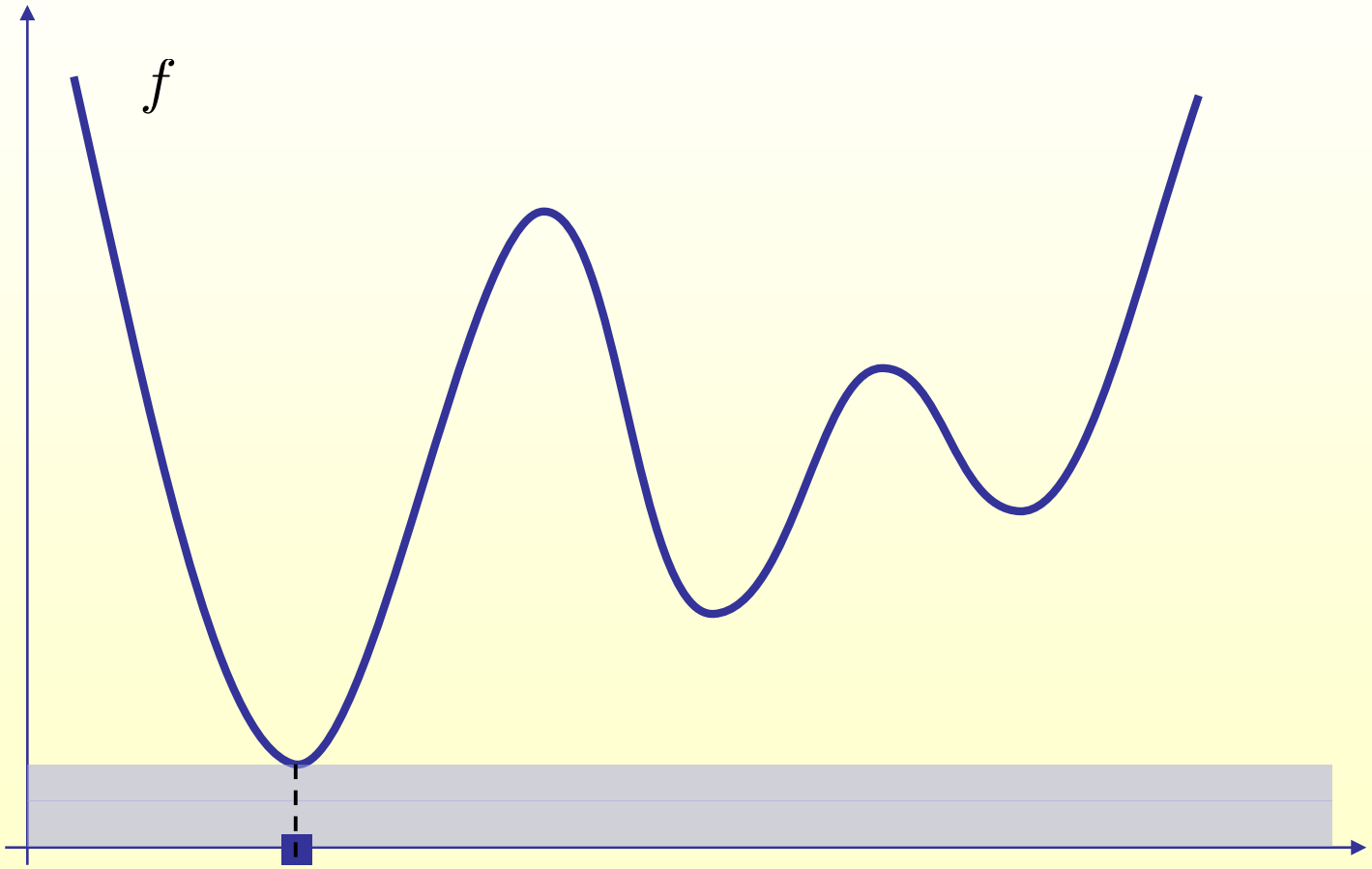
Sublevel Set Persistence



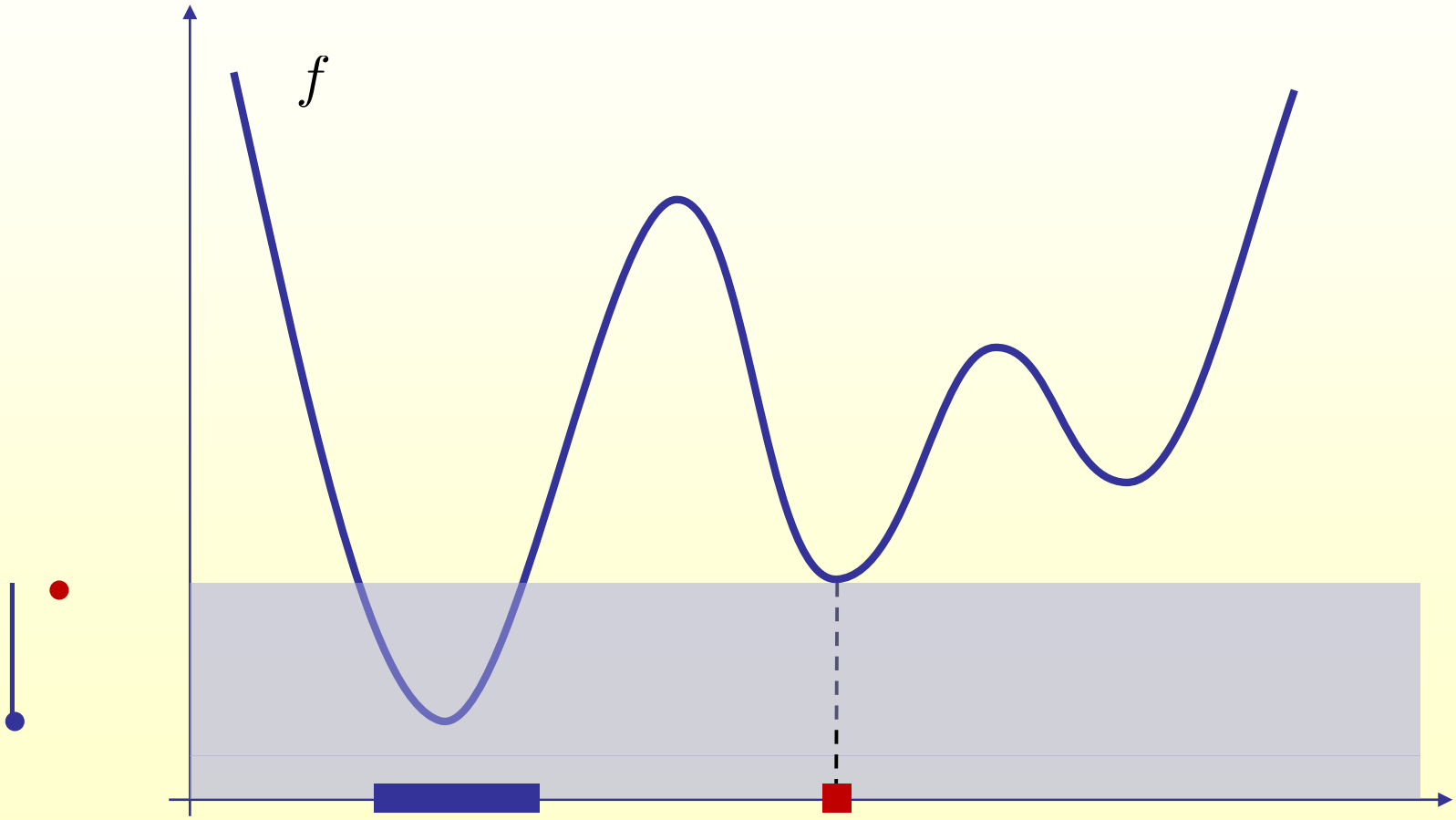
Sublevel Set Persistence



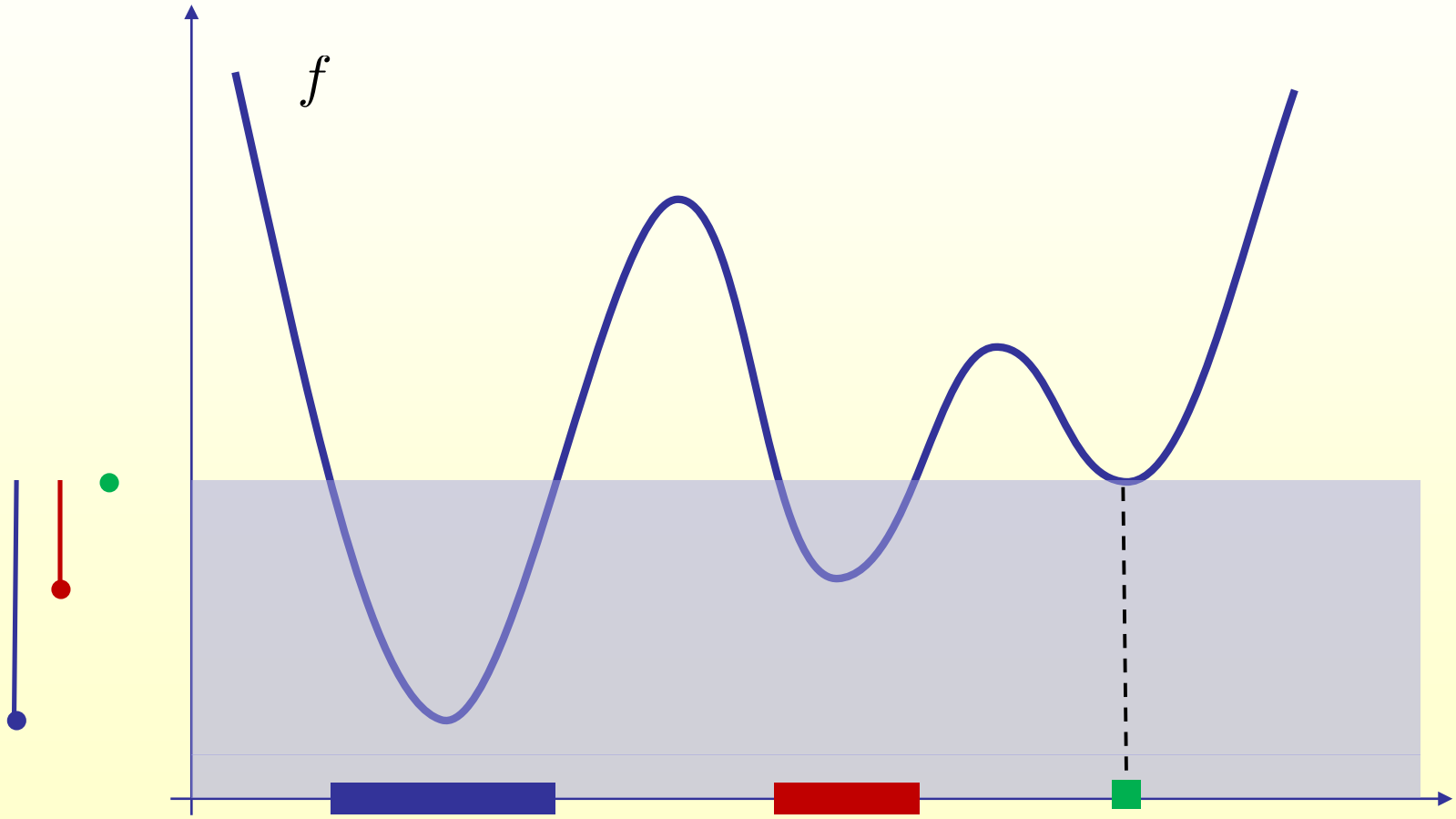
Sublevel Set Persistence



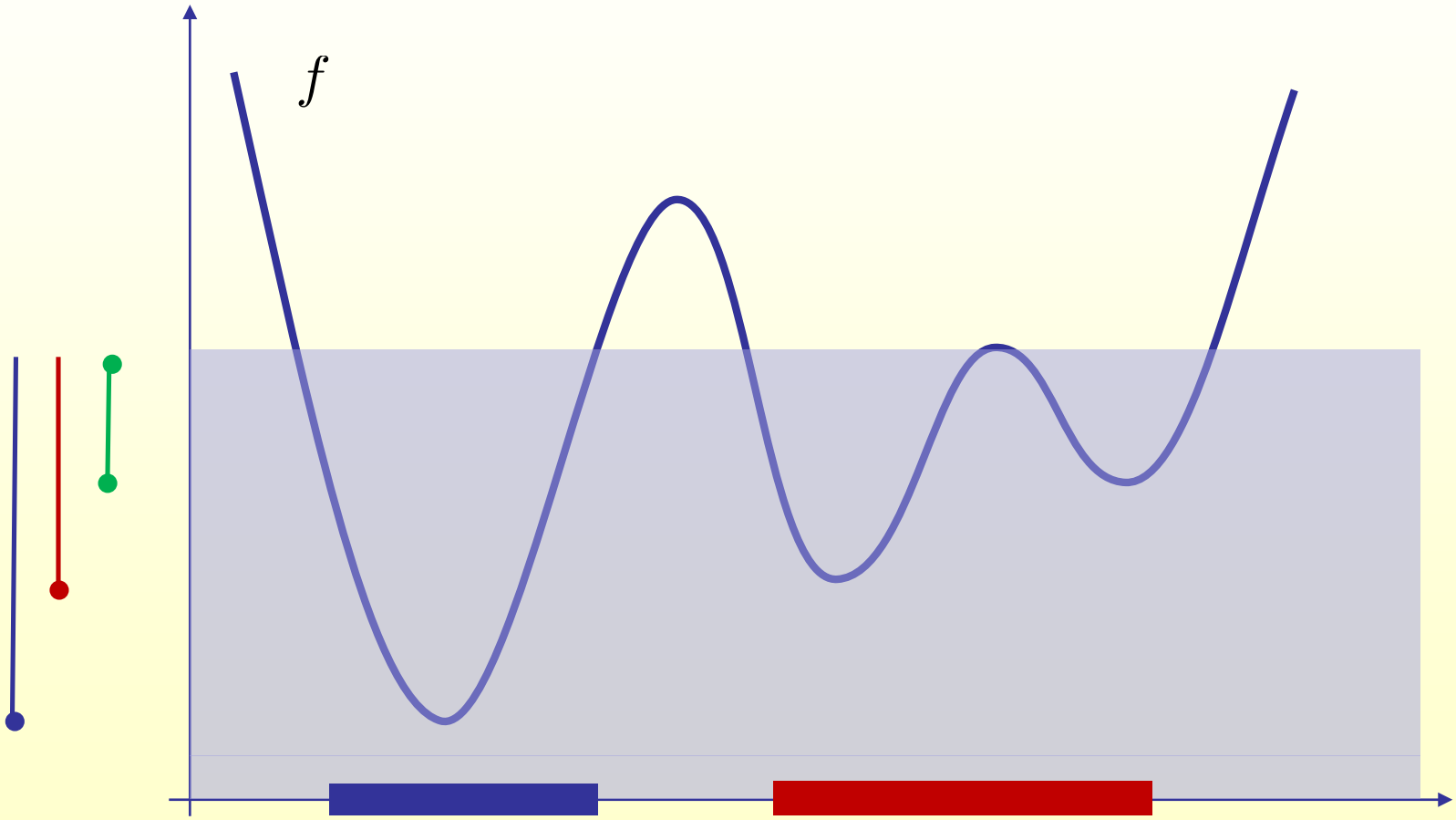
Sublevel Set Persistence



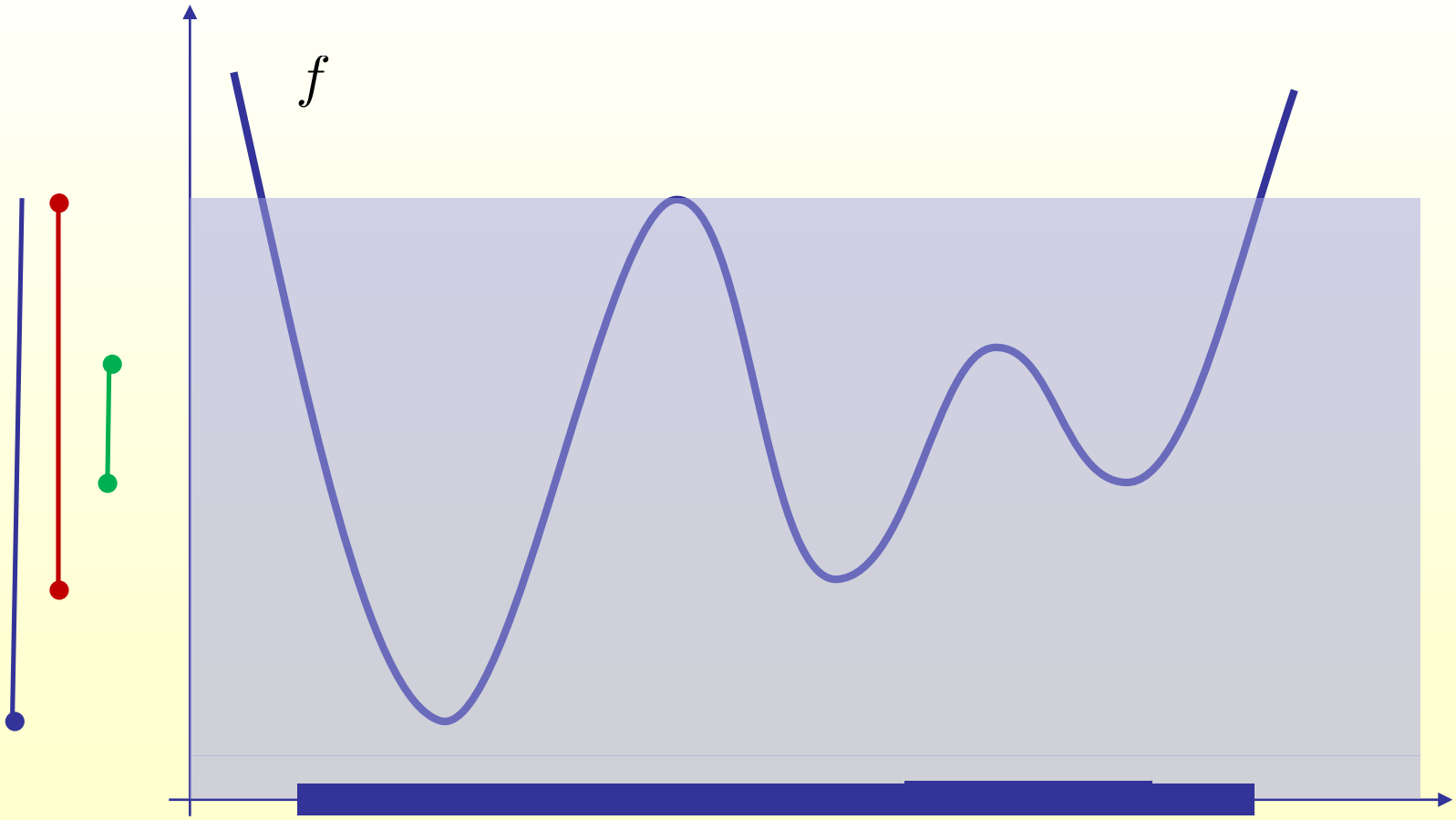
Sublevel Set Persistence



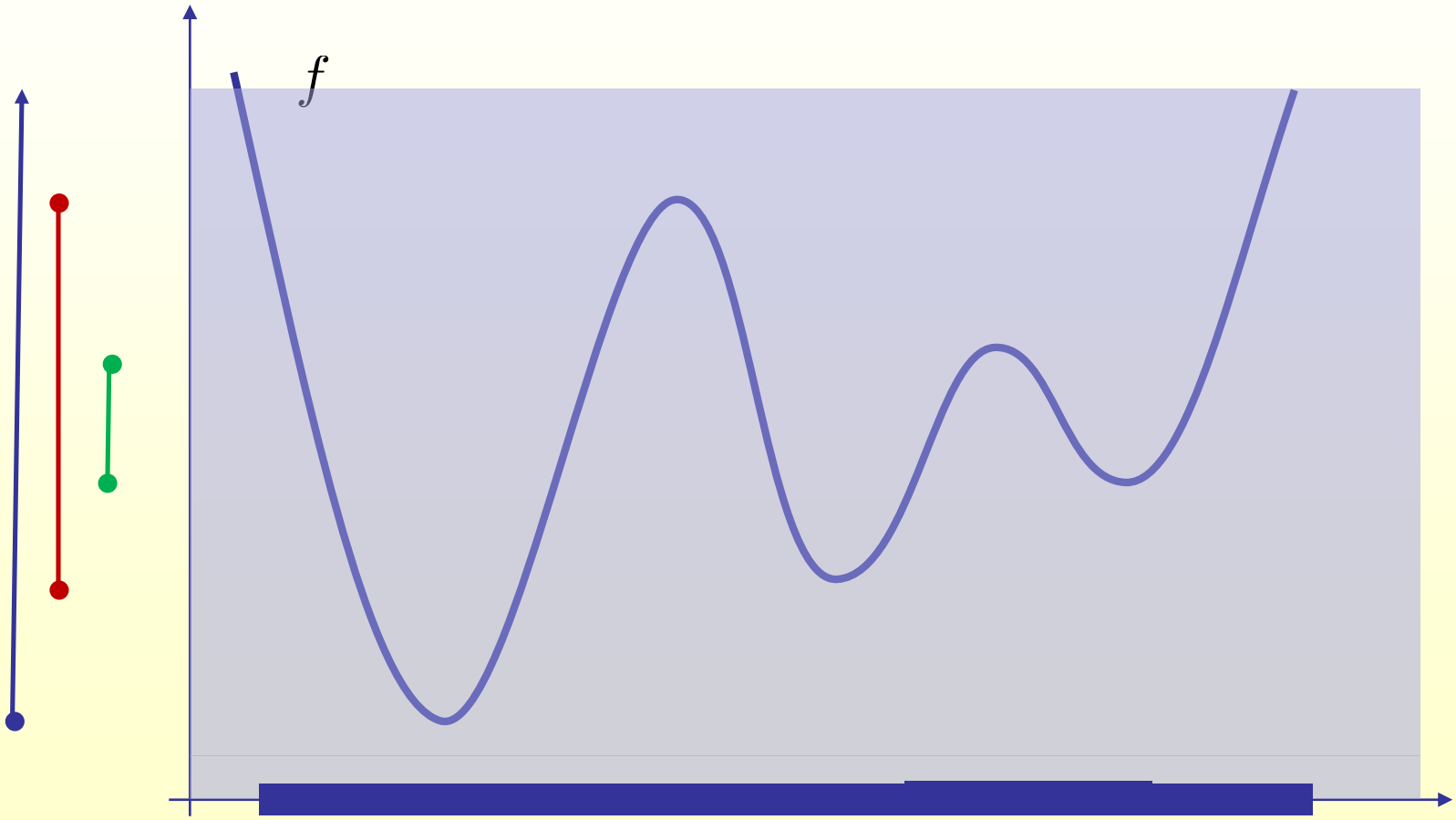
Sublevel Set Persistence



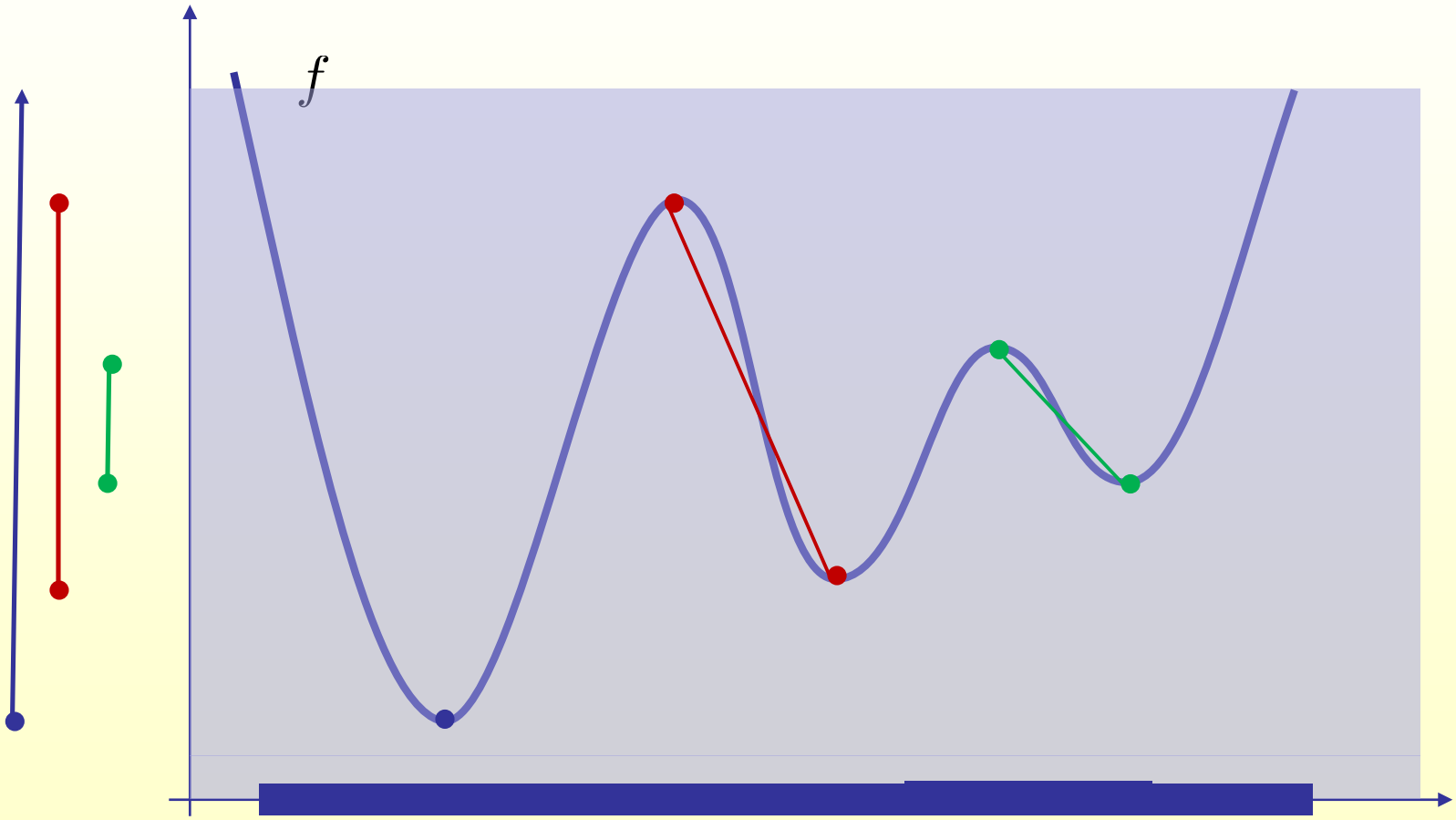
Sublevel Set Persistence



Persistence Gives a Pairing

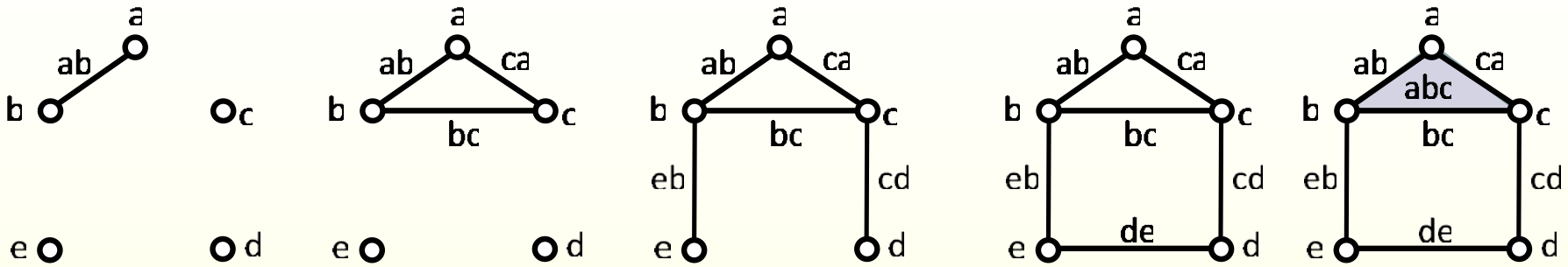


Elder Rule



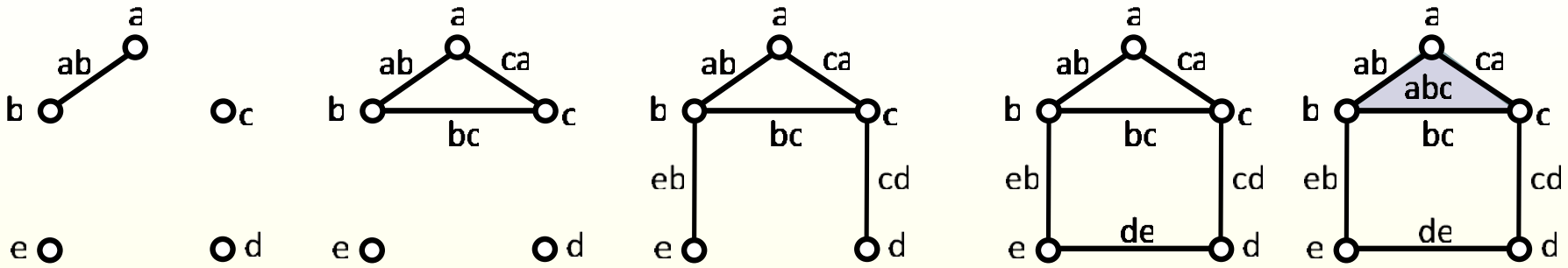
Back to Linear Algebra

Incremental Algorithm



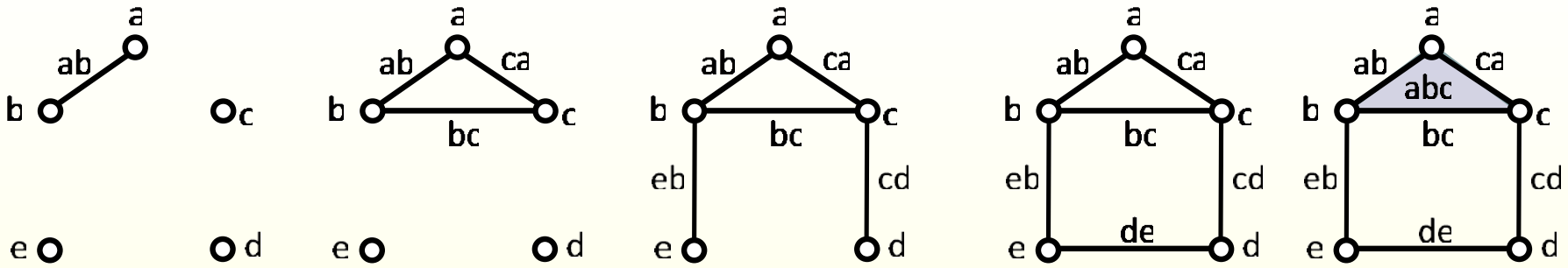
	ab	ca	bc	cd	eb	de	abc
a	1	1	0	0	0	0	0
b	1	0	1	0	1	0	0
c	0	1	1	1	0	0	0
d	0	0	0	1	0	1	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0

Incremental Algorithm



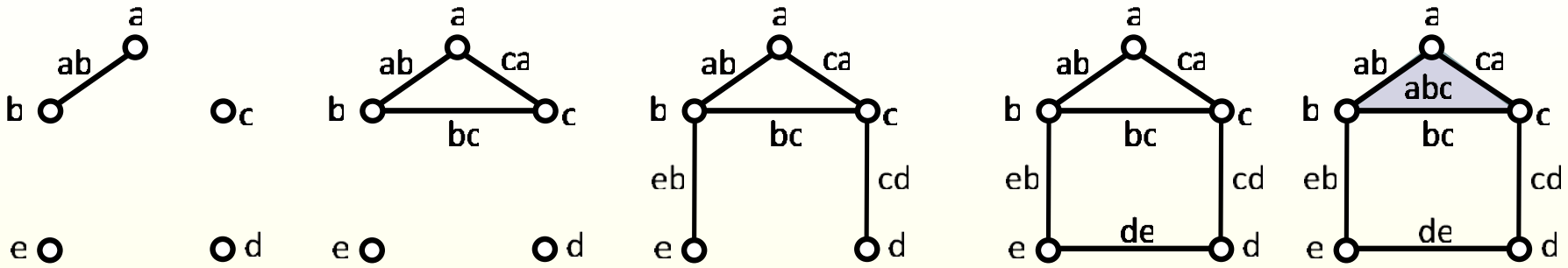
	ab	ca	bc	cd	eb	de	abc
a	1	1	0	0	0	0	0
b	1	0	1	0	1	0	0
c	0	1	1	1	0	0	0
d	0	0	0	1	0	1	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0

Incremental Algorithm



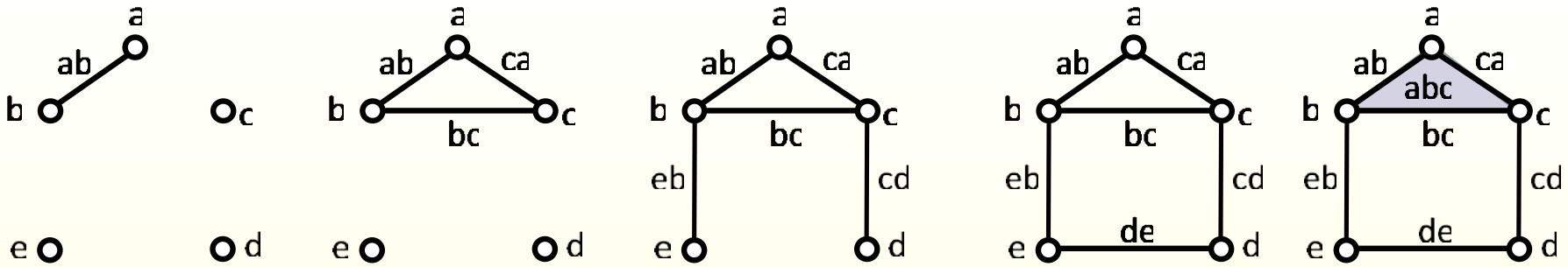
	ab	ca	bc	cd	eb	de	abc
a	1	1	0	0	0	0	0
b	1	0	1	0	1	0	0
c	0	1	1	1	0	0	0
d	0	0	0	1	0	1	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0

Incremental Algorithm



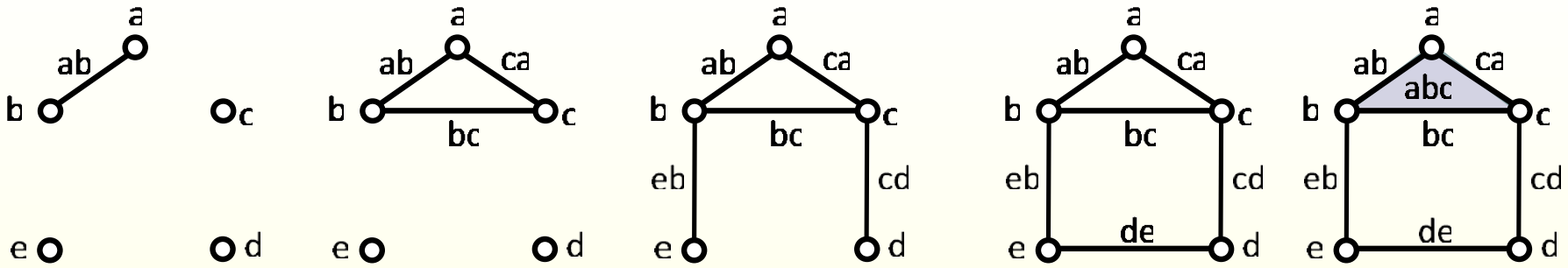
	ab	ca	bc	cd	eb	de	abc
a	1	1	0	0	0	0	0
b	1	0	1	0	1	0	0
c	0	1	1	1	0	0	0
d	0	0	0	1	0	1	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0

Incremental Algorithm



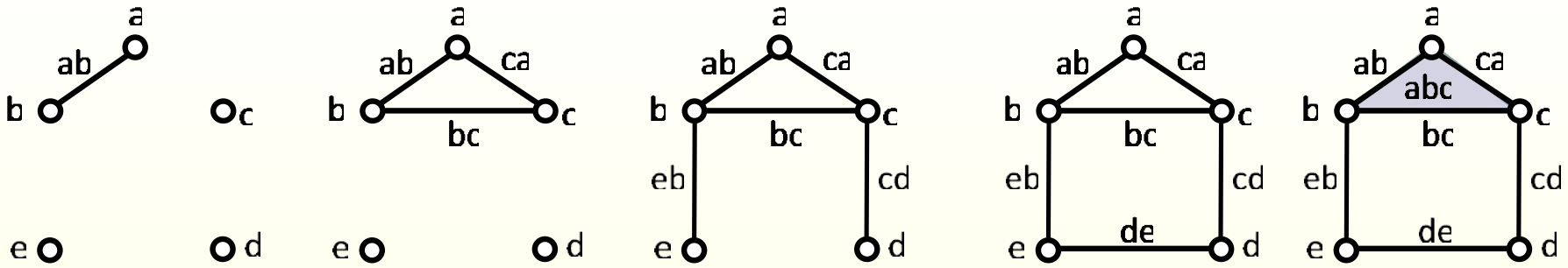
	ab	ca	bc	cd	eb	de	abc
a	1	1	0	0	0	0	0
b	1	0	1	0	1	0	0
c	0	1	1	1	0	0	0
d	0	0	0	1	0	1	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0

Incremental Algorithm



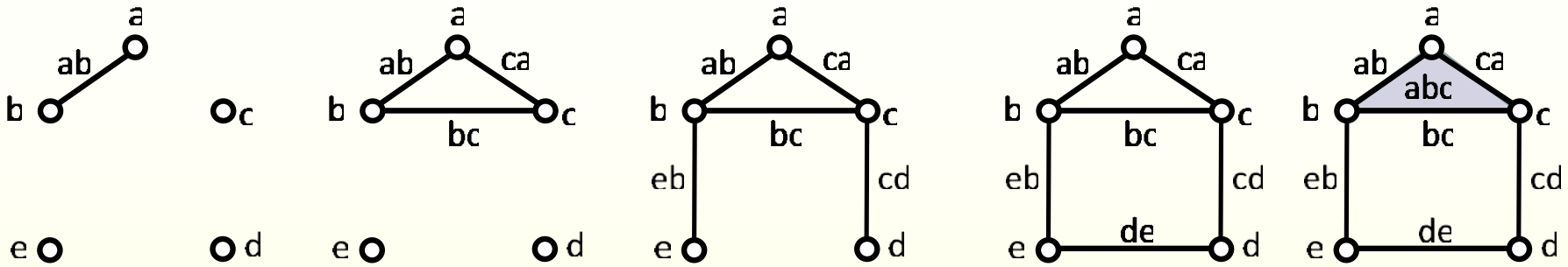
	ab	ca	bc	cd	eb	de	abc
a	1	1	0	0	0	0	0
b	1	0	1	0	1	0	0
c	0	1	1	1	0	0	0
d	0	0	0	1	0	1	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0

Incremental Algorithm



	ab	ca	bc	cd	eb	de	abc
a	1	1	0	0	0	0	0
b	1	0	1	0	1	0	0
c	0	1	1	1	0	0	0
d	0	0	0	1	0	1	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0

Incremental Algorithm



	ab	ca	bc	cd	eb	de	abc
a	1	1	0	0	0	0	0
b	1	0	1	0	1	0	0
c	0	1	1	1	0	0	0
d	0	0	0	1	0	1	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0

Algorithm

Input: Ordering on K, $B = \emptyset$

For each σ in K

if $\partial\sigma \subseteq span(B)$

mark σ positive

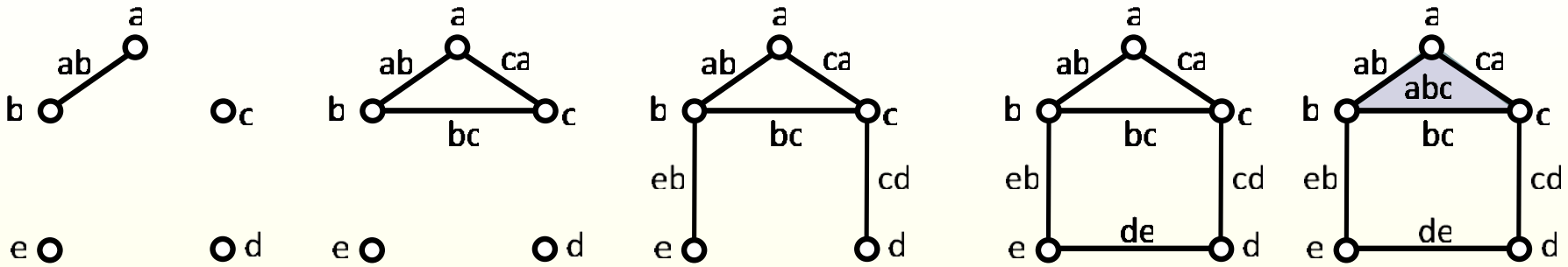
else

$B = B \cup \partial\sigma$

return interval $(\ell(\sigma), \sigma)$

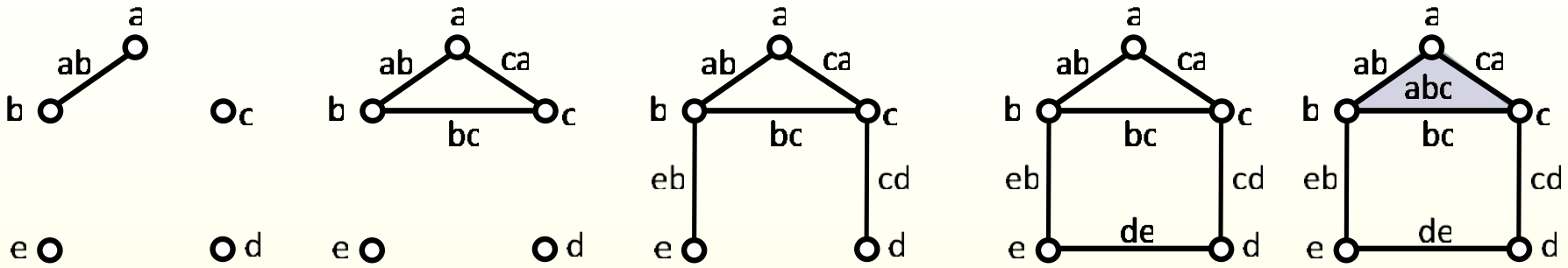
$\ell(\sigma)$ = last positive simplex with non-zero entry in reduced vector

Incremental Algorithm



	ab	ca	bc	cd	eb	de	abc
a	1	1	0	0	0	0	0
b	1	0	1	0	1	0	0
c	0	1	1	1	0	0	0
d	0	0	0	1	0	1	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0

Pivots are Persistence Pairings



	ab	ca	bc	cd	eb	de	abc
a	1	1	0	0	0	0	0
b	1	0	1	0	1	0	0
c	0	1	1	1	0	0	0
d	0	0	0	1	0	1	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0

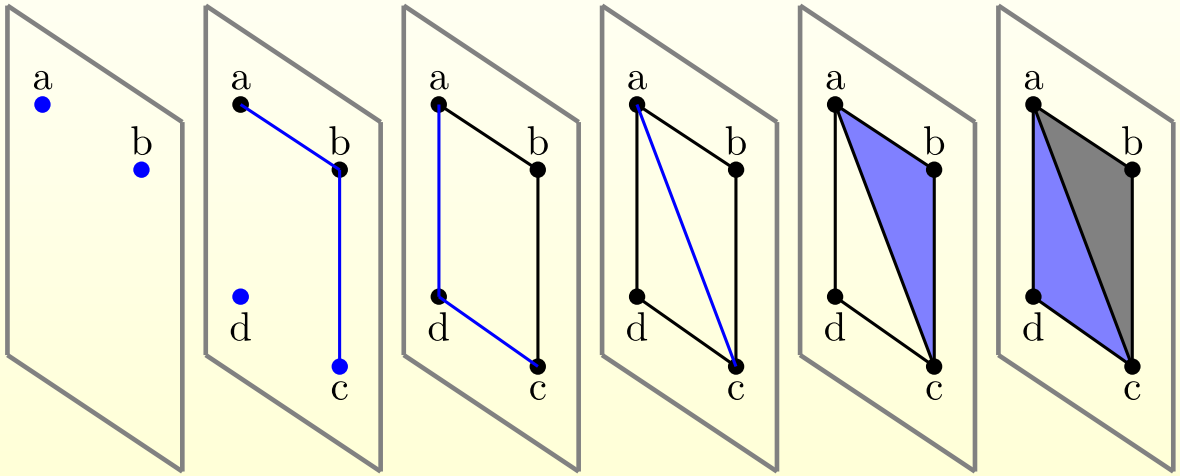
(t(b),t(ab))

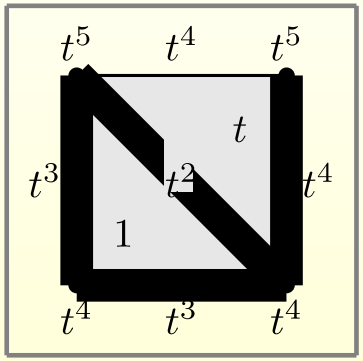
(t(c),t(ca))

(t(bc),t(abc))

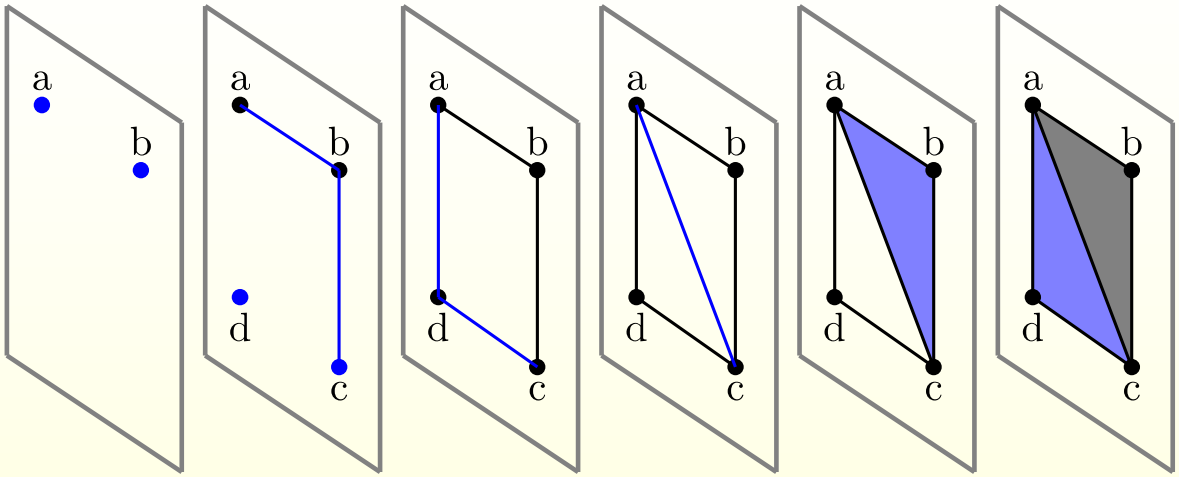
Why does this work?

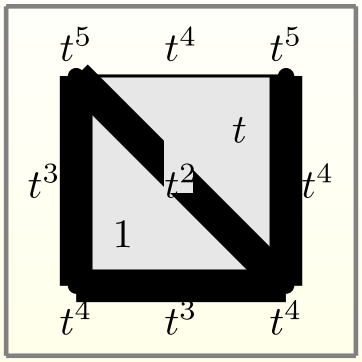
Why does this work?





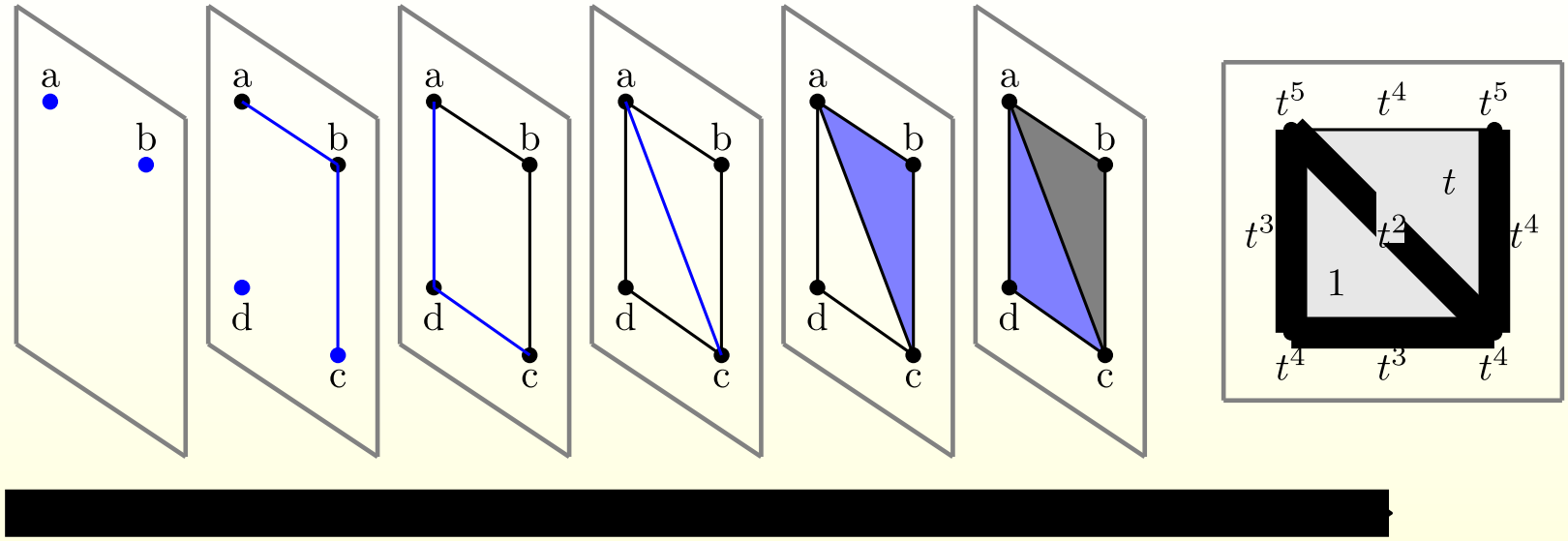
Why does this work?





$$\begin{pmatrix} -t & \cdot & \cdot & -t^2 & -t^3 & \cdot & \cdot \\ t & -t & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & -t & \cdot & t^2 & \cdot & \cdot \\ \cdot & \cdot & t & t & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & t^3 \\ \cdot & \cdot & \cdot & \cdot & \cdot & t & -t^2 \end{pmatrix}$$

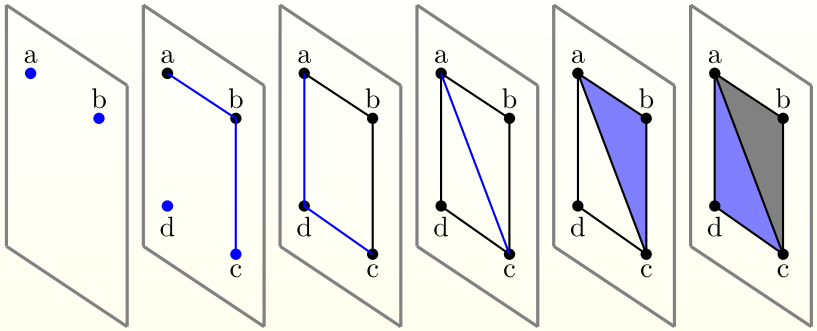
Why does this work?

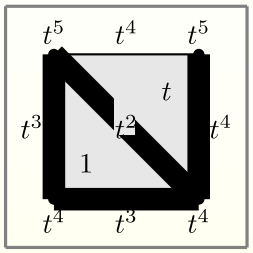


$$z_1 = a \qquad z_3 = c \qquad z_5 = tab + tbc + cd - ad$$

$$z_2 = b \qquad z_4 = d \qquad z_6 = t^2ab + t^2bc - ac$$

Why does this work?

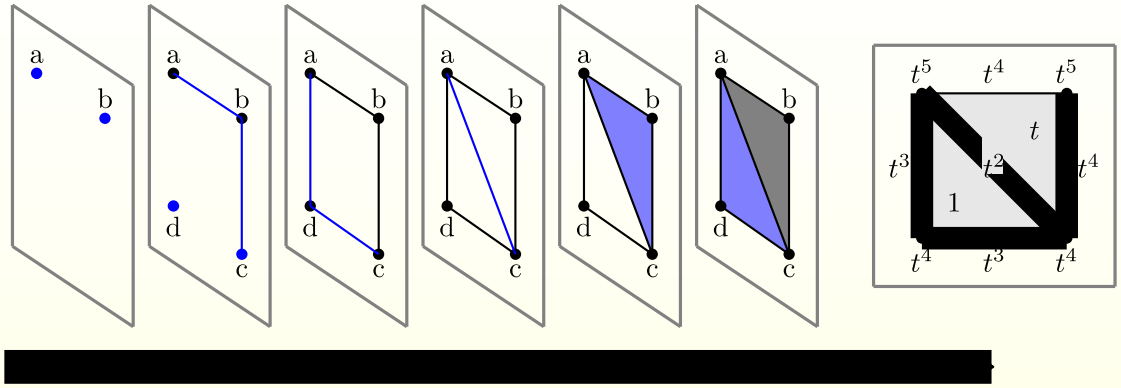




$$\begin{pmatrix} -t & \cdot & \cdot & -t^2 & -t^3 & \cdot & \cdot \\ t & -t & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & -t & \cdot & t^2 & \cdot & \cdot \\ \cdot & \cdot & t & t & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & t^3 \\ \cdot & \cdot & \cdot & \cdot & \cdot & t & -t^2 \end{pmatrix} \longrightarrow \begin{pmatrix} \cdot & -t & \cdot & -t^2 & -t^3 & \cdot & \cdot \\ t & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & -t & \cdot & t^2 & \cdot & \cdot \\ \cdot & \cdot & t & t & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & t^3 \\ \cdot & \cdot & \cdot & \cdot & \cdot & t & -t^2 \end{pmatrix}$$

$$\begin{pmatrix} \cdot & \cdot & -t^2 & -t^2 & \cdot & \cdot & \cdot \\ t & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & t & t & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & t^3 \\ \cdot & \cdot & \cdot & \cdot & \cdot & t & -t^2 \end{pmatrix} \longrightarrow \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ t & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & t & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & t^3 \\ \cdot & \cdot & \cdot & \cdot & \cdot & t & -t^2 \end{pmatrix} \longrightarrow \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ t & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & t & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & t^3 \\ \cdot & \cdot & \cdot & \cdot & \cdot & t & \cdot \end{pmatrix}$$

Why does this work?



$$\begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ t & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & t & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & t^3 \\ \cdot & \cdot & \cdot & \cdot & \cdot & t & \cdot \end{pmatrix}$$

$r_1 \mapsto tz_2 - tz_1$

$r_2 \mapsto z_3 - tz_2$

$r_3 \mapsto tz_4 - tz_3$

$r_4 \mapsto tz_4 - t^2 z_1$

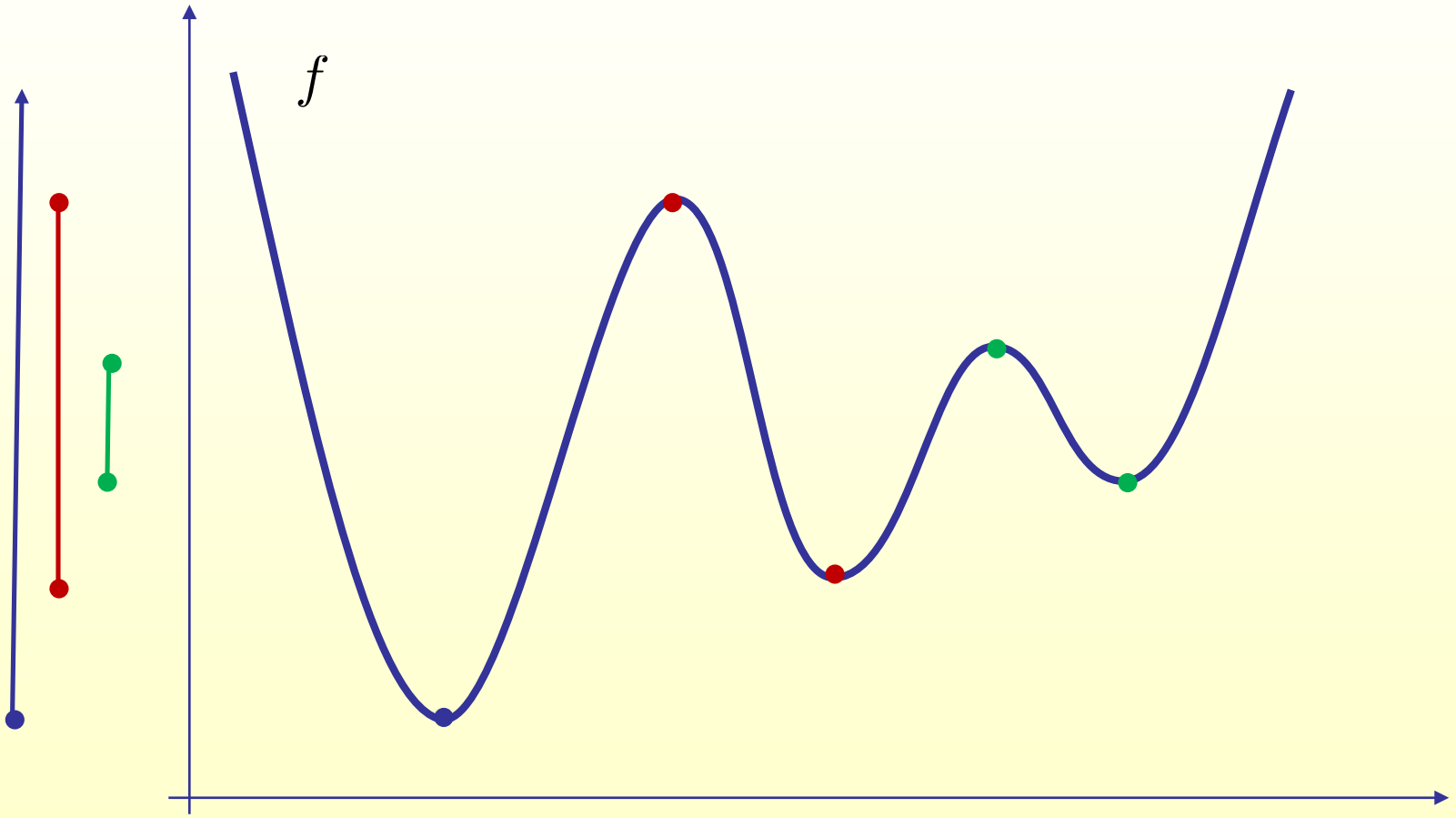
$r_5 \mapsto t^2 z_3 - t^3 z_1$

$r_6 \mapsto tz_6$

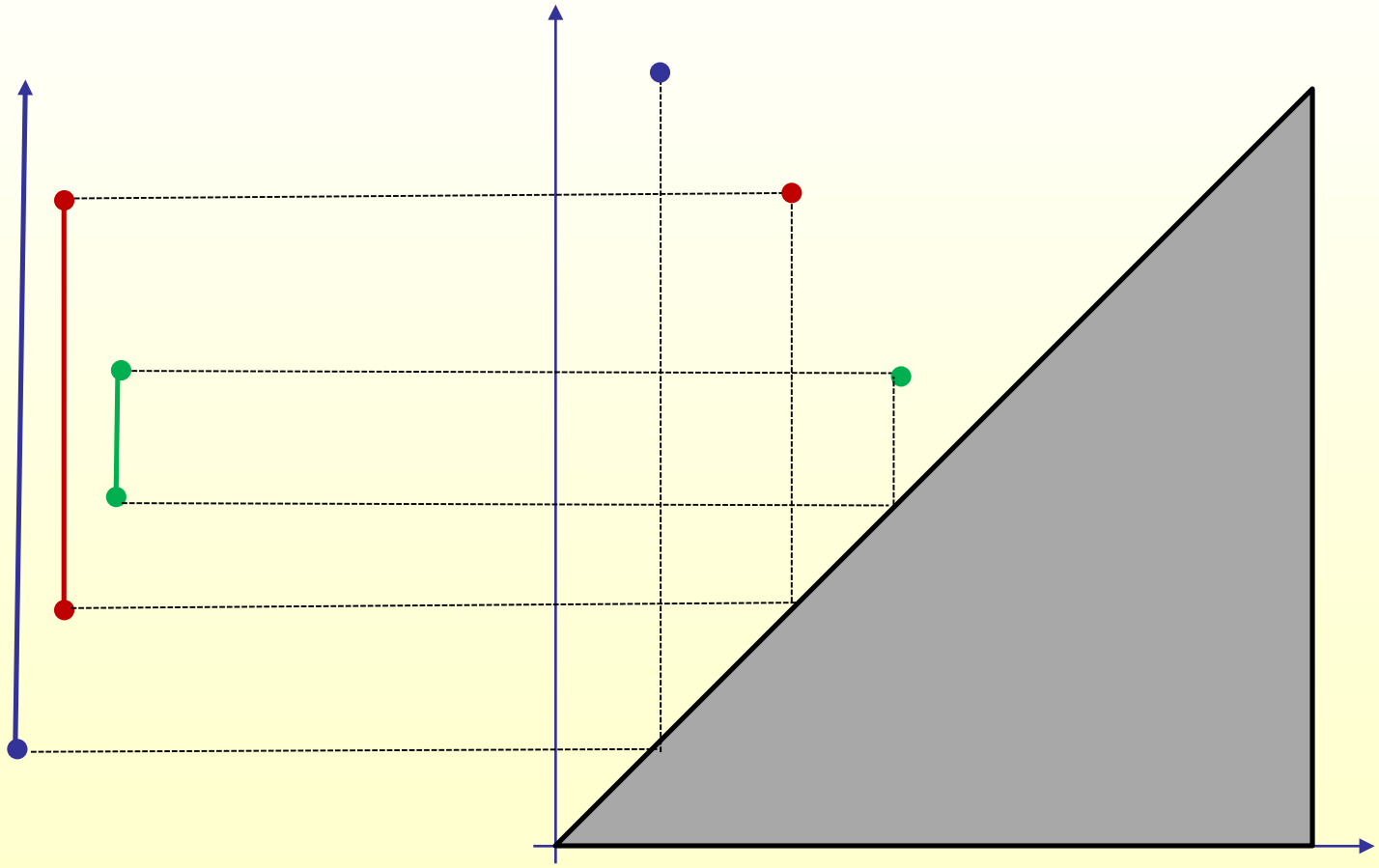
$r_7 \mapsto t^3 z_5 - t^2 z_6$

Persistence Diagrams

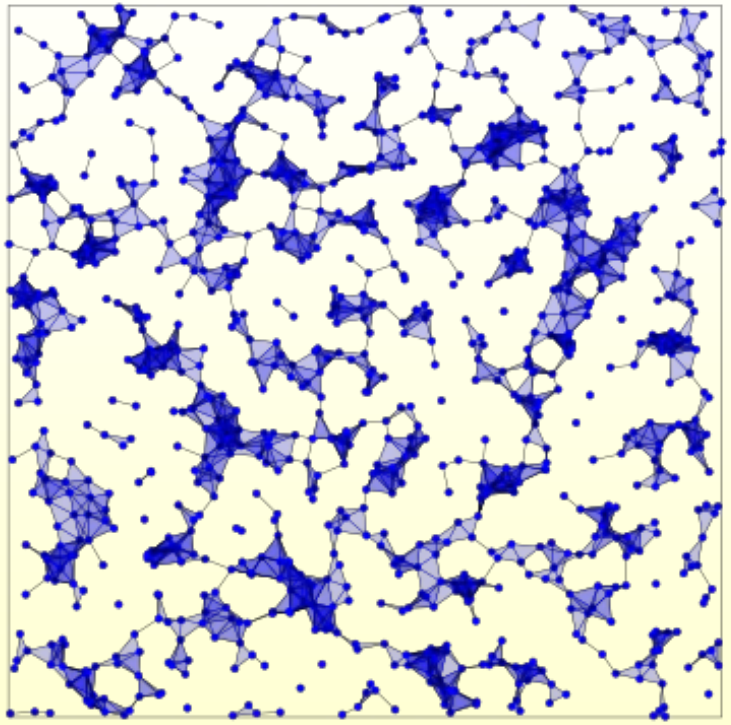
Persistence Diagrams

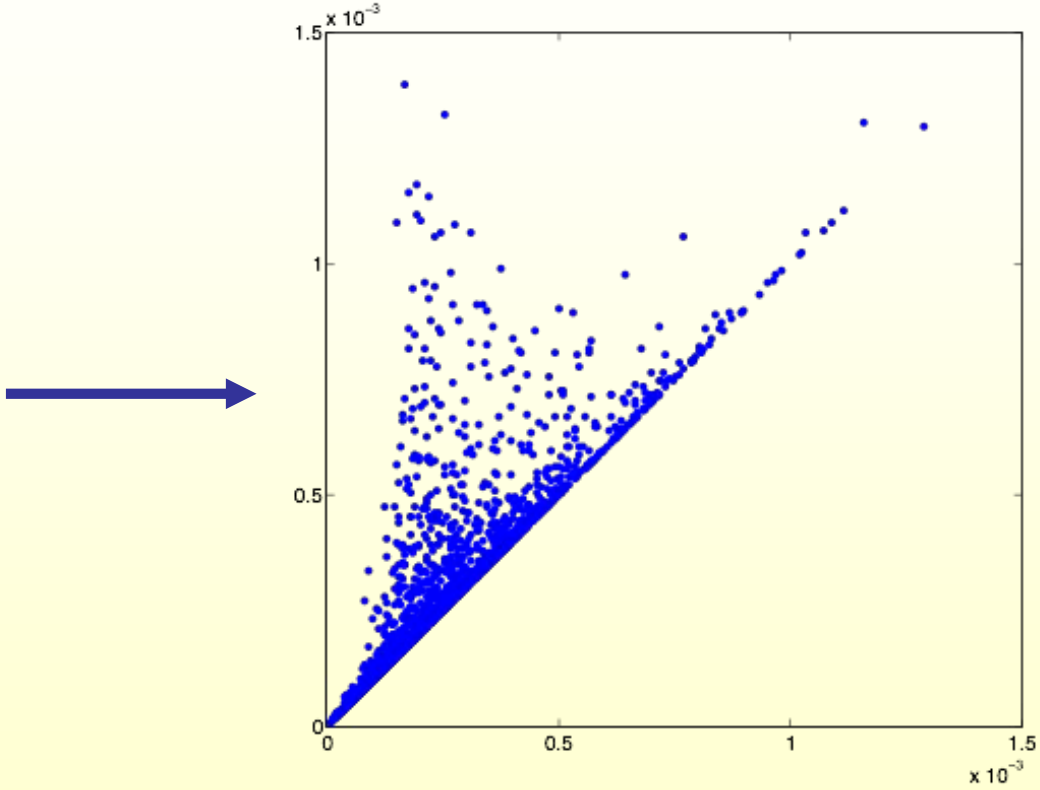


Persistence Diagrams



Example





“Modern” Approaches

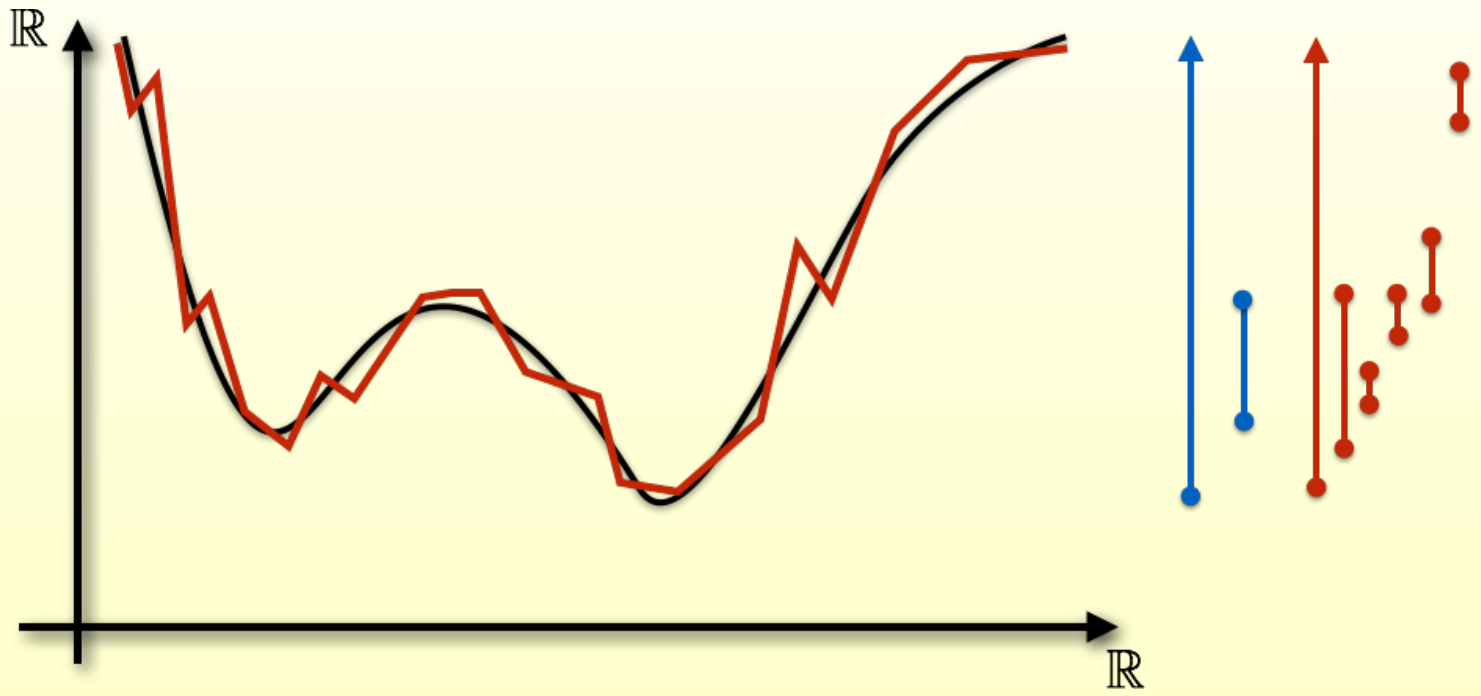
- Worst case: cubic
- Using Fast Matrix Multiplication (as fast)
- In practice, numerous optimization
  - Persistent cohomology
  - Discrete Morse Theory
  - Matroid Theory
  - Many more…

Computing in Practice

- Javaplex - https://appliedtopology.github.io/javaplex/
- Gudhi - http://gudhi.gforge.inria.fr/
- Perseus - http://people.maths.ox.ac.uk/nanda/perseus/
- Dionysus - http://www.mrzv.org/software/dionysus2/
- Eirene - https://github.com/Eetion/Eirene.jl
- Ripser - https://github.com/Ripser/ripser
  http://live.ripser.org/
- Phat - https://bitbucket.org/phat-code/phat
- TDA package in R

Stability

- Key property of persistence



Where to From Here?

- Types of complexes, functions
- Statistics of persistence diagrams
- Mathematical connections to
  - Representation theory
  - Algebraic geometry
  - Category theory
  - ….