Computing Homology

$$\left[\begin{array}{ccc|c} b_1 & & 0 & \\ & \ddots & & 0 \\ 0 & & b_{l_k} & \\ \hline & 0 & & 0 \end{array}\right]$$

Afra Zomorodian

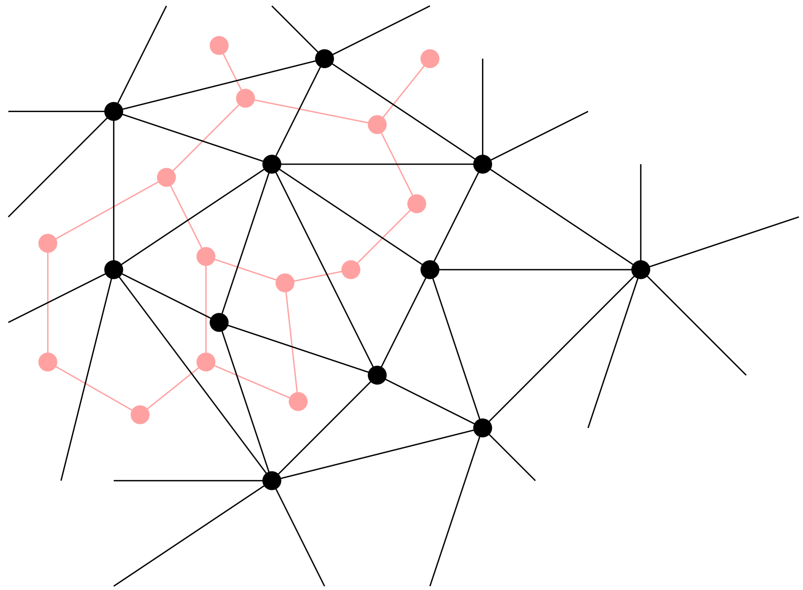
CS 468 – Lecture 7

2-25-4

OVERVIEW

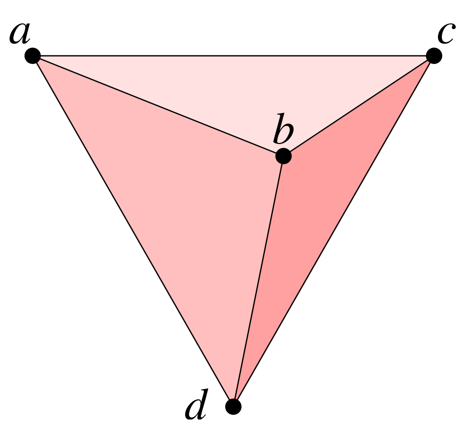
- Duality
- Data structures
  - Quad-Edge
  - Edge-Facet
- Algorithms
  - Reduction Algorithm
  - Incremental Algorithm

STRUCTURE

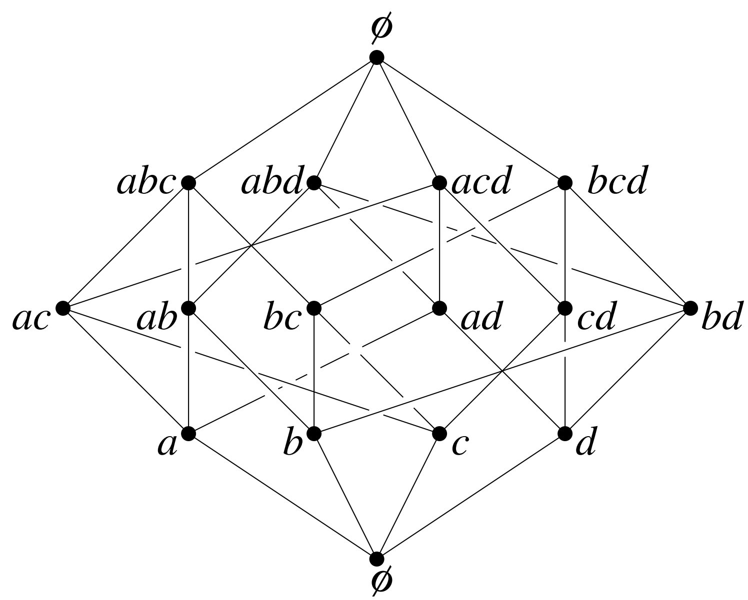


- Triangulation K of d-manifold
- All d-simplices are principal
- Quick access to topology: k-simplex to $(d-k)$-simplex

Duality of Structure



(a) Tetrahedron

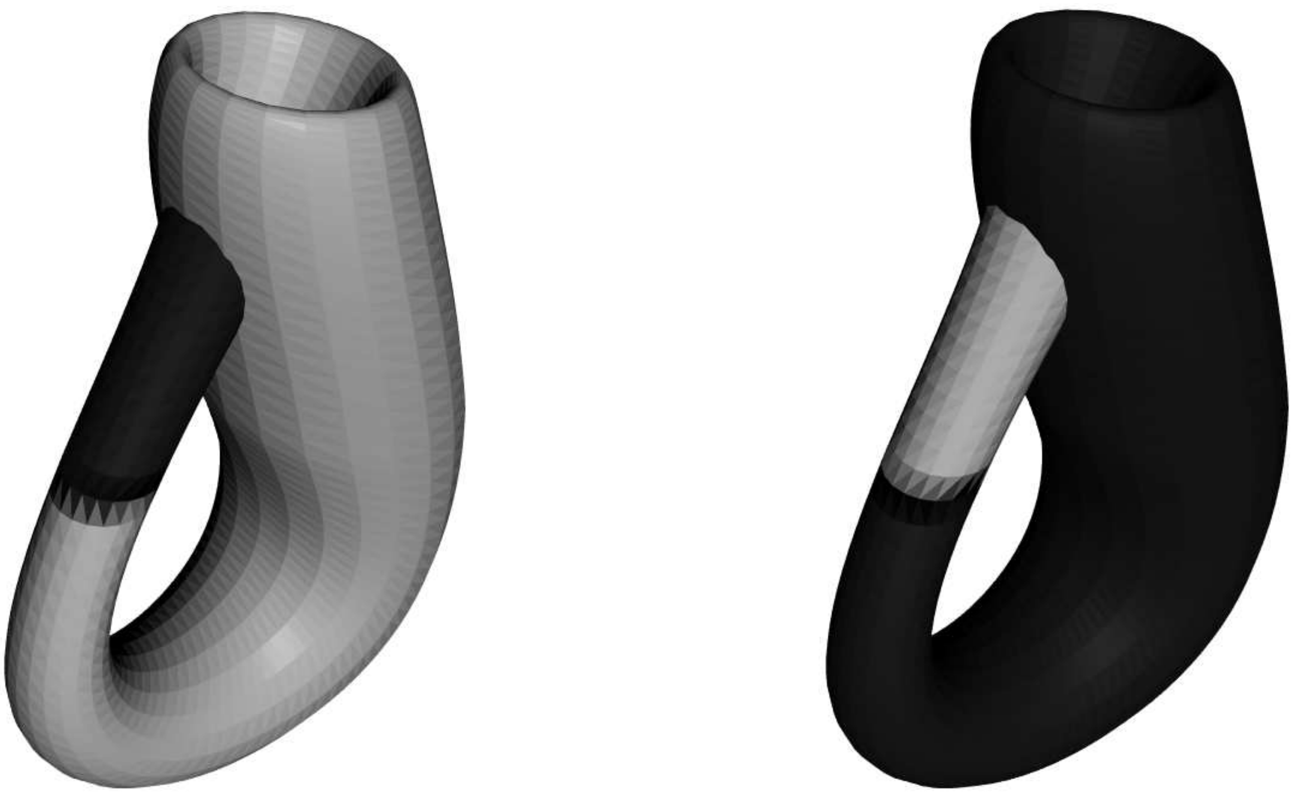


(b) Poset

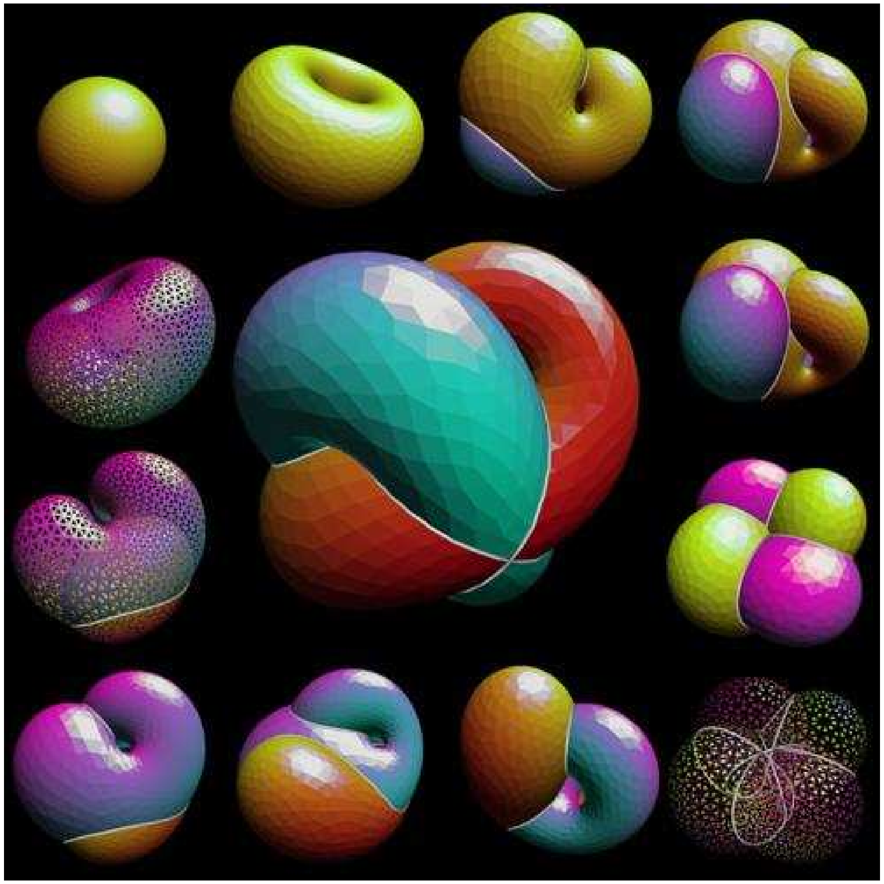
PLATONIC SOLIDS

solid	vertices	edges	faces
tetrahedron	4	6	4
cube	8	12	6
octahedron	6	12	8
dodecahedron	20	30	12
icosahedron	12	30	20

ORIENTATION

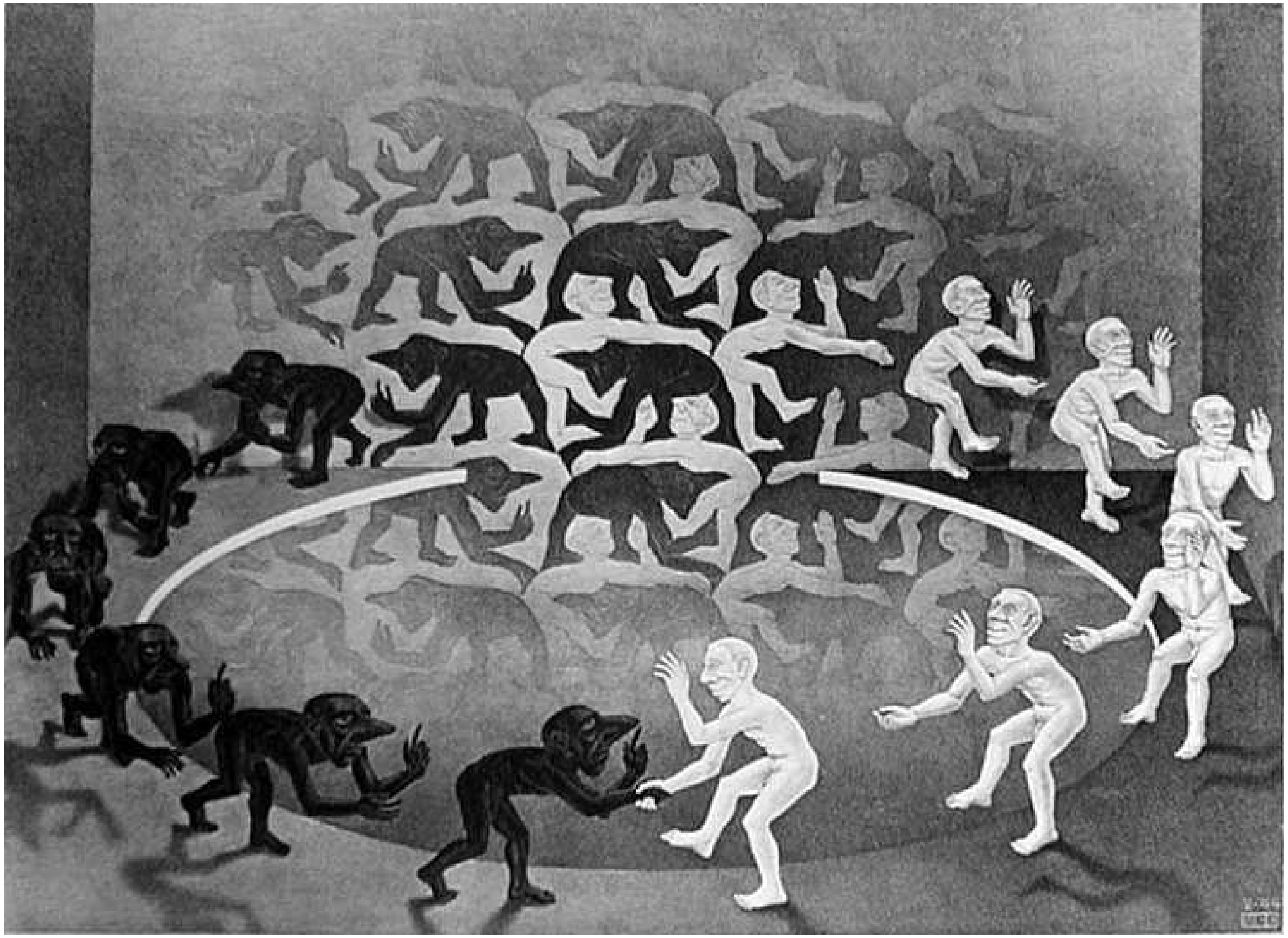


INSIDE / OUTSIDE DUALITY



The Optiverse [Sullivan '98]

FOREGROUND / BACKGROUND



COMPLEMENTARITY

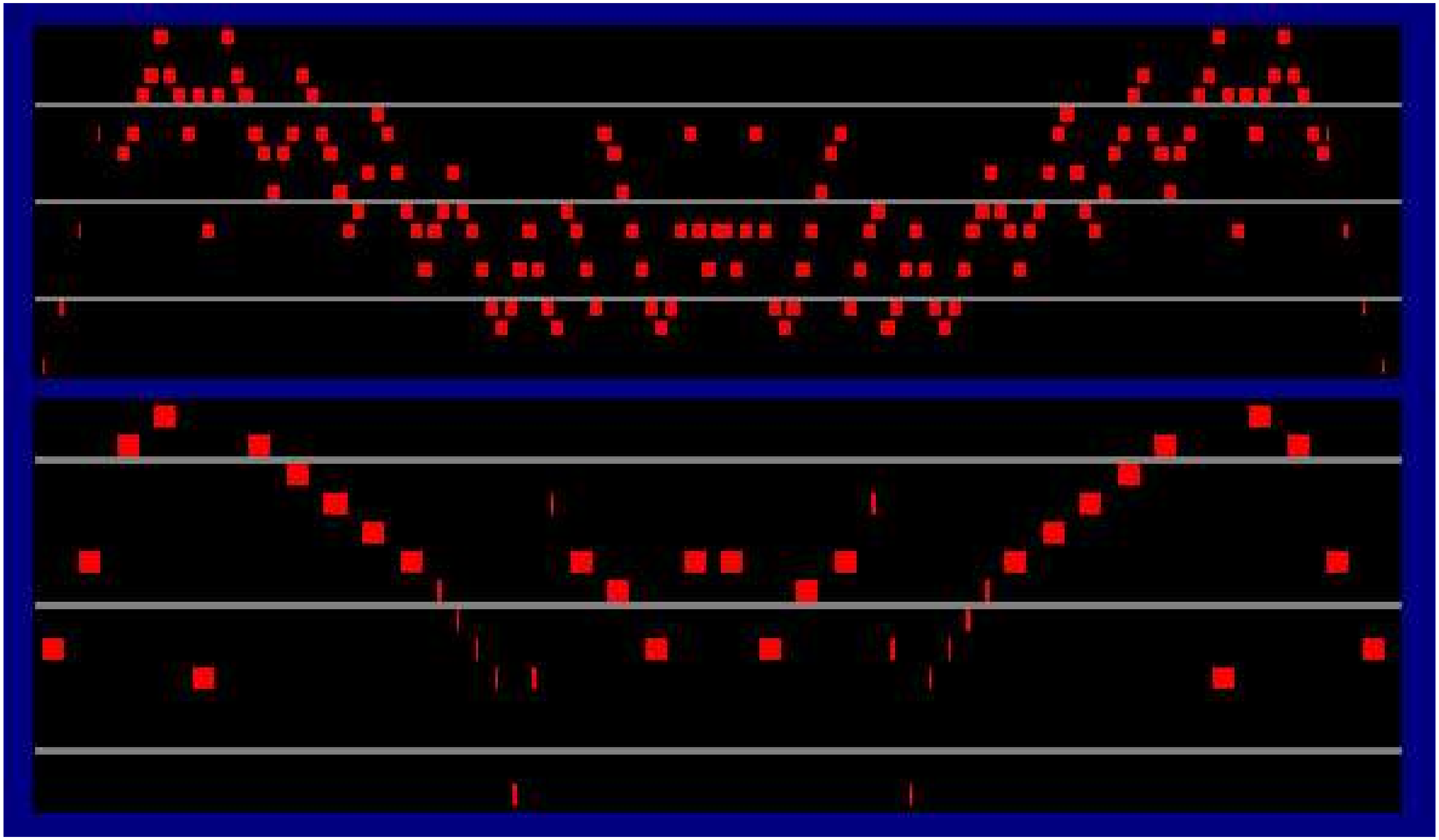
Complementarity

- Changes in primal space affect complement space
- Extrinsic topology
- What is a void?
- Complement of a torus

CANON A 2

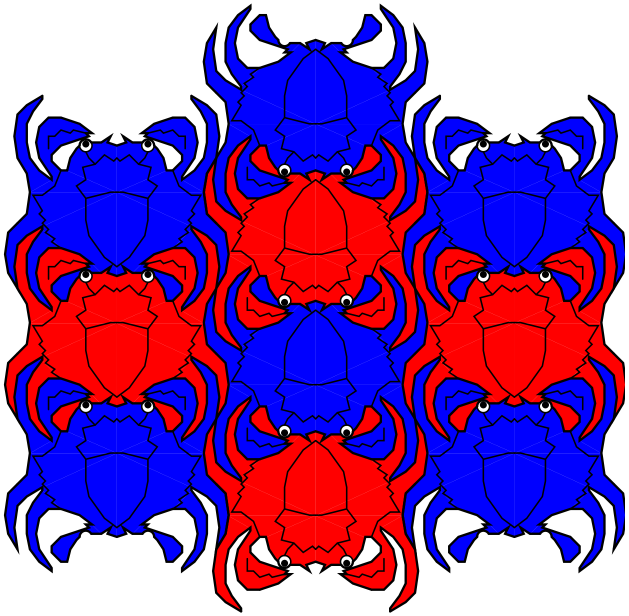

[Presto ♩ = 96]
[Violin I]
[Violin II]
[f marcato]
[f non troppo]
[f non troppo]
[ff]

CRAB CANON



TIME REVERSAL

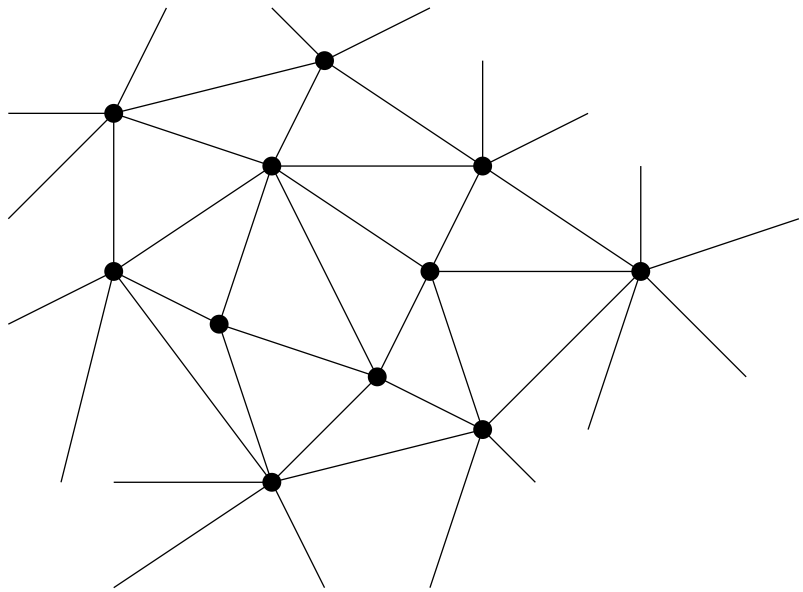
- Self-dual
- Möbius strip
- Escher: complementarity



DUALITY RECAP

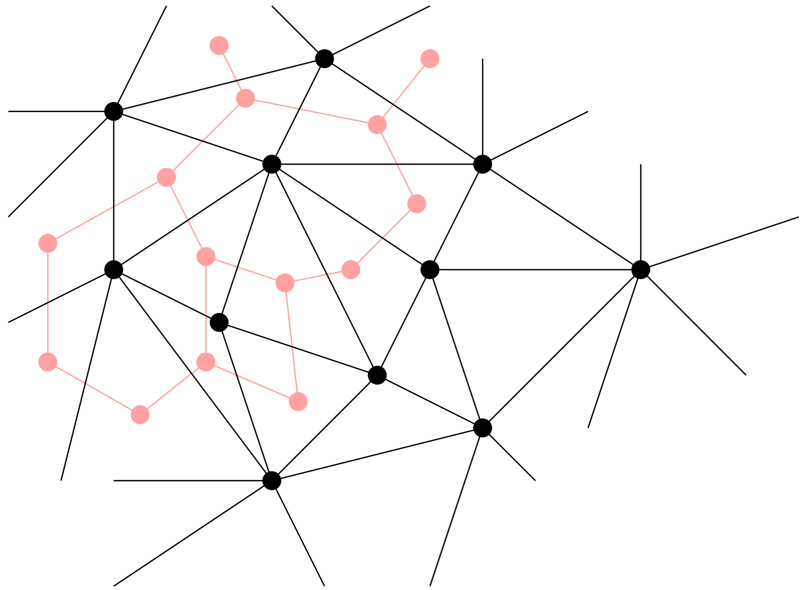
- Structure duality: primal vs. dual
- Orientation duality: inside vs. outside
- Space duality (complementarity): embedded object vs. ambient space
- Time duality: forward vs. backward

2D PRIMAL



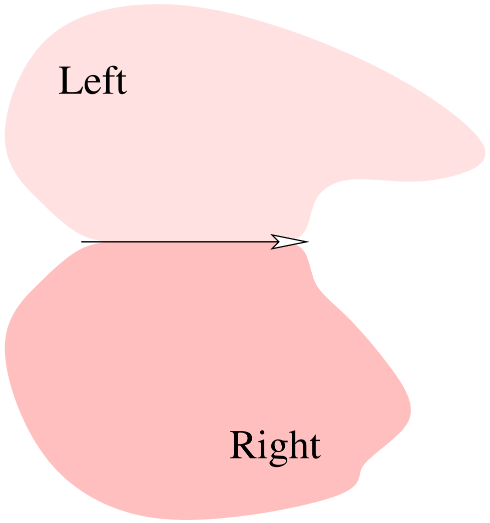
- Vertices V
- Edges E
- Triangles are cycles in $G(V, E)$, the 1-skeleton of K

2D DUAL



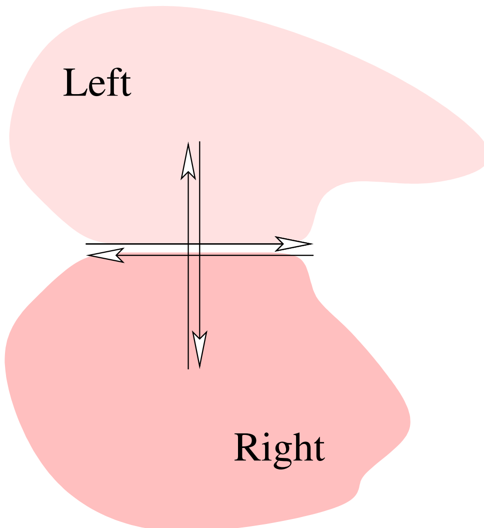
- Vertices ← primal triangles
- Edges ← primal edges
- Faces (cycles) ← primal vertices
- Focus on edges

DIRECTED EDGE



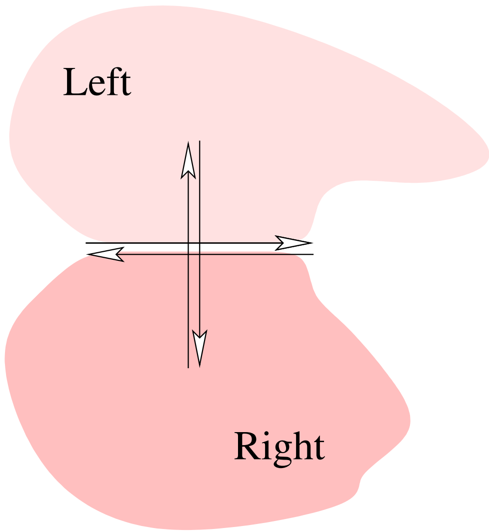
- An edge e has two vertices
- A directed edge goes from Org (e) to Dest (e)
- An edge separates two cells
- Sym (e) goes in the opposite direction

QUAD-EDGE



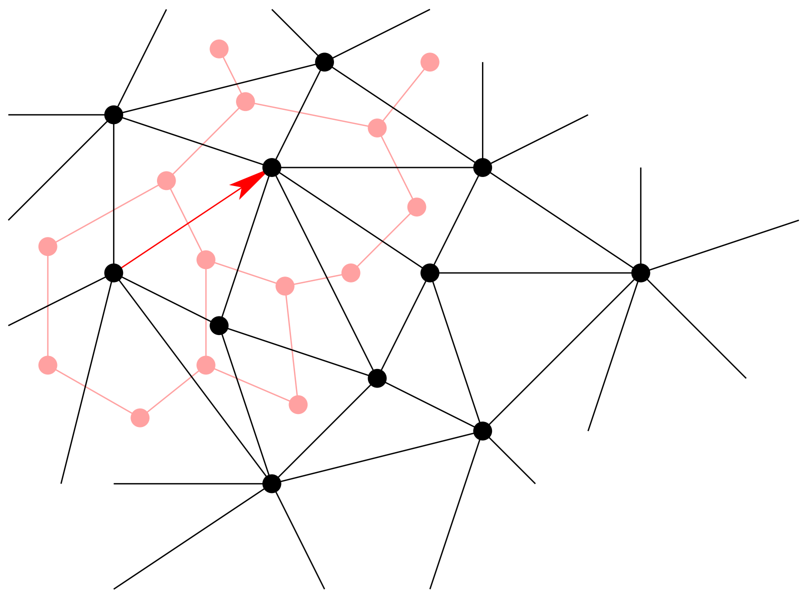
- Rot(e) gives you the dual edge (cw) and Tor(e) (ccw)
- Edge e stores its number and the next (cw) edge with the same origin: Onext (e)
- A Quad-Edge is Edge[4]: edge, rot, sym, tor
- All operations $O(1)$

OPERATIONS

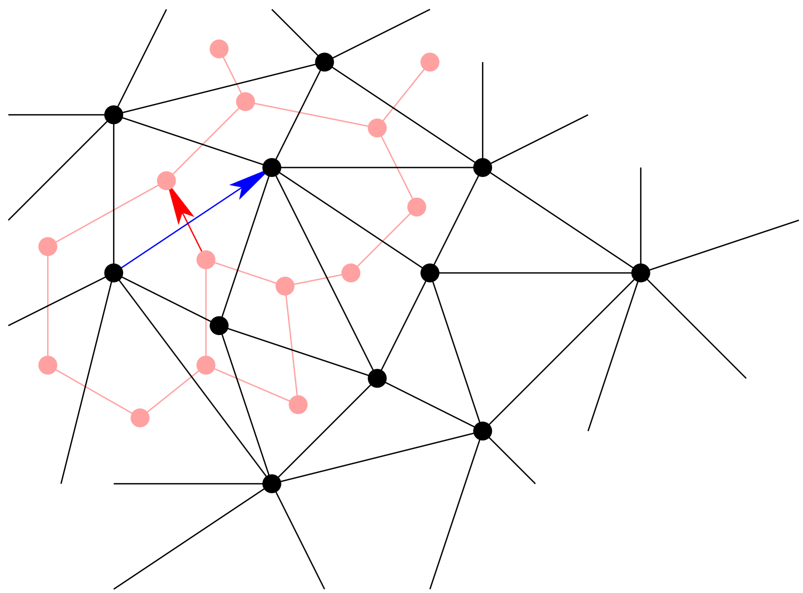


- $\text{Oprev}\,(e) = (\text{Rot} \circ \text{Onext} \circ \text{Rot})(e)$
- $\text{Dnext}\,(e) = (\text{Sym} \circ \text{Onext} \circ \text{Sym})(e)$
- $\text{Lnext}\,(e) = (\text{Tor} \circ \text{Onext} \circ \text{Rot})(e)$

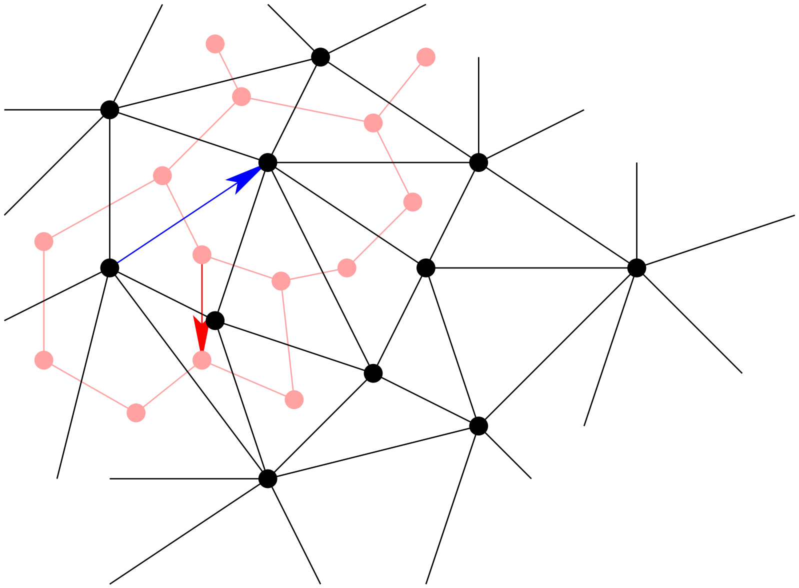
EDGE (e)



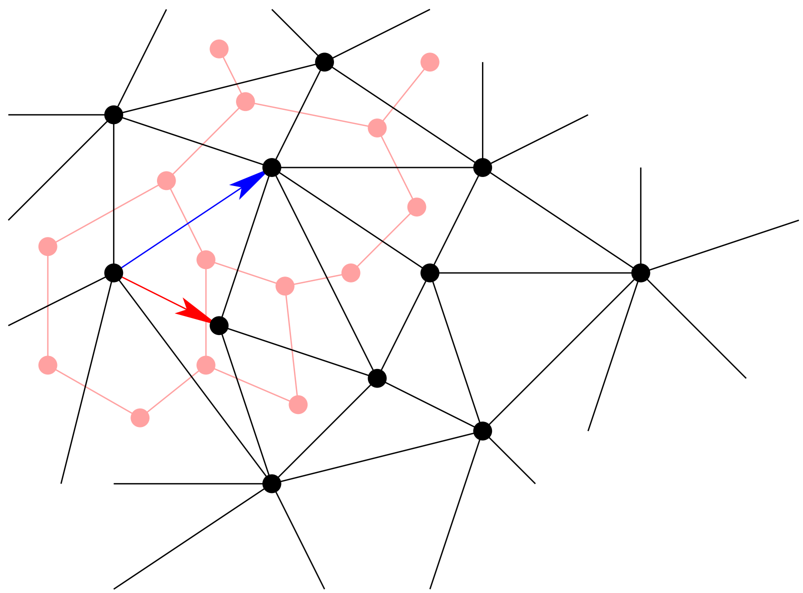
Rot (e)



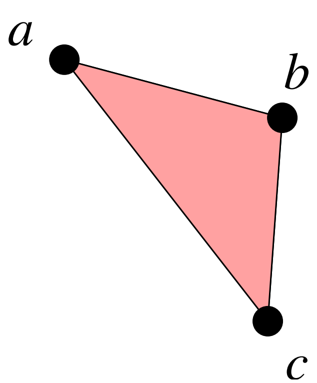
$(\text{ONEXT} \circ \text{ROT})(e)$

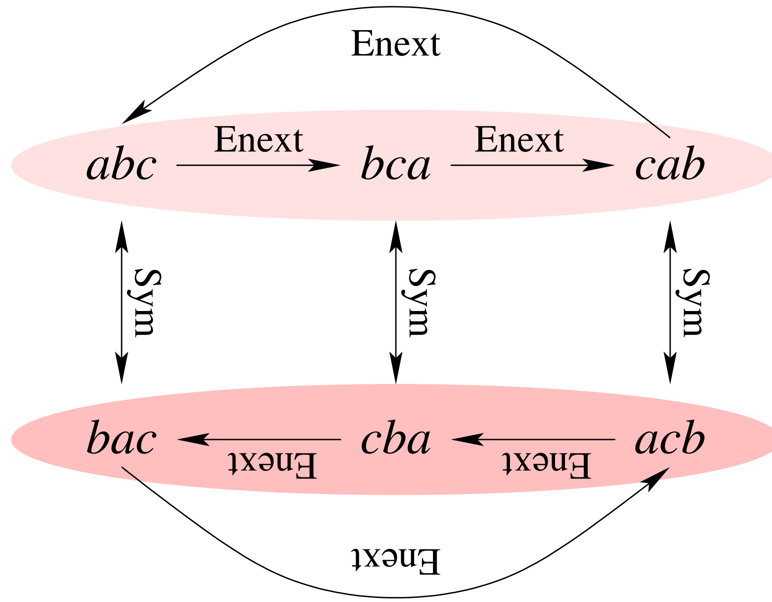


$$\text{OPREV}(e) = (\text{ROT} \circ \text{ONEXT} \circ \text{ROT})(e)$$



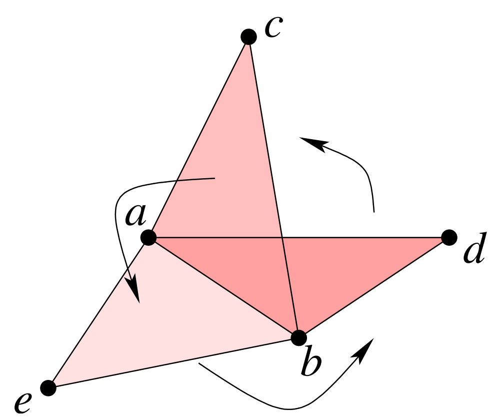
ORIENTED TRIANGLES





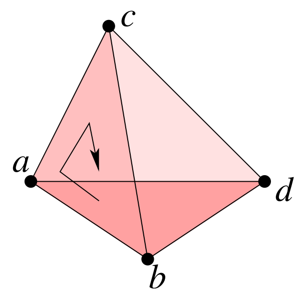
- Sym does one transposition (changes orientation)
- Enext does two transpositions (rotate by 60 degrees clockwise)
- Array of six edge-facets for each triangle
- All operations $O(1)$

FNEXT

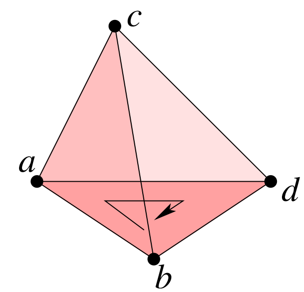


- Fnext $(bac) = bad$
- Fnext $(abd) = abc$
- Fnext $(abc) = abe$
- Each store its Fnext

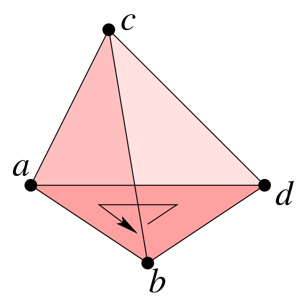
FACES OF A TETRAHEDRON



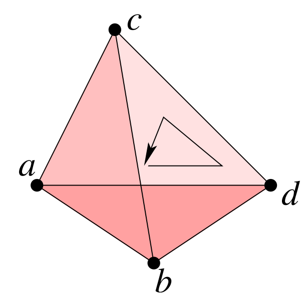
(a) $f = bac$



(b) $\text{Fnext}\,(f) = bad$



(c) $\text{Sym}\,(\text{Fnext}\,(f)) = abd$



(d) $\text{Sym}\,(\text{Fnext}\,(\text{Enext}\,(f))) = cad$

REPRESENTING TOPOLOGY

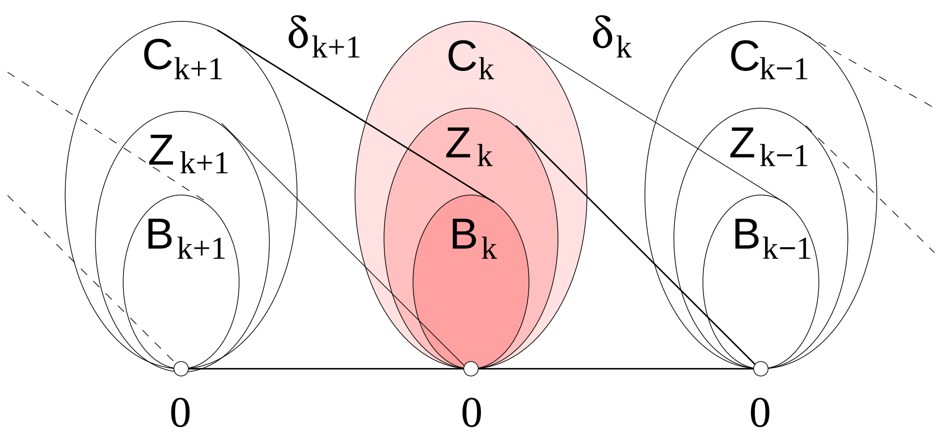
- Store primal and dual structures
- $O(1)$ local operations
- Gives fast boundary operator
- Many available in literature

HOMOLOGY

- The kth homology group is

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

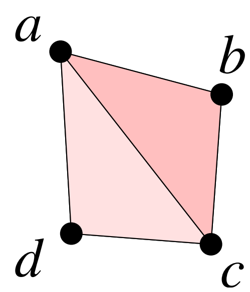
- $\beta_k = \operatorname{rank} \mathsf{Z}_k - \operatorname{rank} \mathsf{B}_k$
- Compute bases for $\ker \partial_k$ and $\operatorname{im} \partial k + 1$



Matrix Representation

- Boundary homomorphism is linear, so it has a matrix
- $\partial_k : \mathsf{C}_k \rightarrow \mathsf{C}_{k-1}$
- Use oriented simplices as bases for domain and codomain
- M_k is the standard matrix representation for ∂_k
- It is an $m_{k-1} \times m_k$ matrix

EXAMPLE



$$M_1 = \left[\begin{array}{c|ccccc} \partial_1 & ab & bc & cd & ad & ac \\ \hline a & -1 & 0 & 0 & -1 & -1 \\ b & 1 & -1 & 0 & 0 & 0 \\ c & 0 & 1 & -1 & 0 & 1 \\ d & 0 & 0 & 1 & 1 & 0 \end{array}\right]$$

ELEMENTARY OPERATIONS

- The elementary row operations on M_k are
  1. exchange row i and row j,
  2. multiply row i by -1,
  3. replace row i by (row i) $+ q$(row j), where q is an integer and $j \neq i$.
- Similar elementary column operations on columns
- Change of bases
- No change in rank

DESCRIPTION

- Homology groups are finitely generated abelian.
- (Theorem) Every finitely generated abelian group is isomorphic to product of cyclic groups of the form

$$\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \ldots \times \mathbb{Z}_{m_r} \times \mathbb{Z} \times \mathbb{Z} \times \ldots \times \mathbb{Z},$$

- $\beta_k = \beta(\mathsf{H}_k)$
- Torsion coefficients $m_1, m_2, \ldots, m_r$

INTUITION

- How do we find cycles?
- How do we find boundaries?
- What does a free generator correspond to?
- How does a torsional element appear?

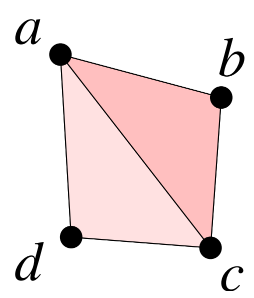
Reduction Algorithm

- Like Gaussian elimination, we keep changing the basis to get to the (Smith) normal form:

$$\tilde{M}_k = \left[\begin{array}{ccc|c} b_1 & & 0 & \\ & \ddots & & 0 \\ 0 & & b_{l_k} & \\ \hline & 0 & & 0 \\ & & & \end{array}\right]$$

- $l_k = \text{rank } M_k = \text{rank } \tilde{M}_k$, $b^i \geq 1$
- $b_i | b_{i+1}$ for all $1 \leq i < l_k$

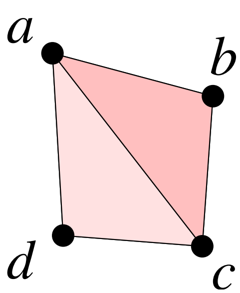
Reduced Example



$$\tilde{M}_1 = \left[\begin{array}{c|ccccc} & cd & bc & ab & z_1 & z_2 \\ \hline d-c & 1 & 0 & 0 & 0 & 0 \\ c-b & 0 & 1 & 0 & 0 & 0 \\ b-a & 0 & 0 & 1 & 0 & 0 \\ a & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

- $z_1 = ad - bc - cd - ab$ and $z_2 = ac - bc - ab$ form a basis for Z_1
- $\{d - c, c - b, b - a\}$ is a basis for B_0

REDUCED EXAMPLE



$$M_2 = \left[\begin{array}{c|cc} & abc & acd \\ \hline ac & -1 & 1 \\ ad & 0 & -1 \\ cd & 0 & 1 \\ bc & 1 & 0 \\ ab & 1 & 0 \end{array}\right]$$

$$\tilde{M}_2 = \left[\begin{array}{c|cc} & -abc & -acd + abc \\ \hline ac - bc - ab & 1 & 0 \\ ad - cd - bc - ab & 0 & 1 \\ cd & 0 & 0 \\ bc & 0 & 0 \\ ab & 0 & 0 \end{array}\right]$$

NORMAL FORM

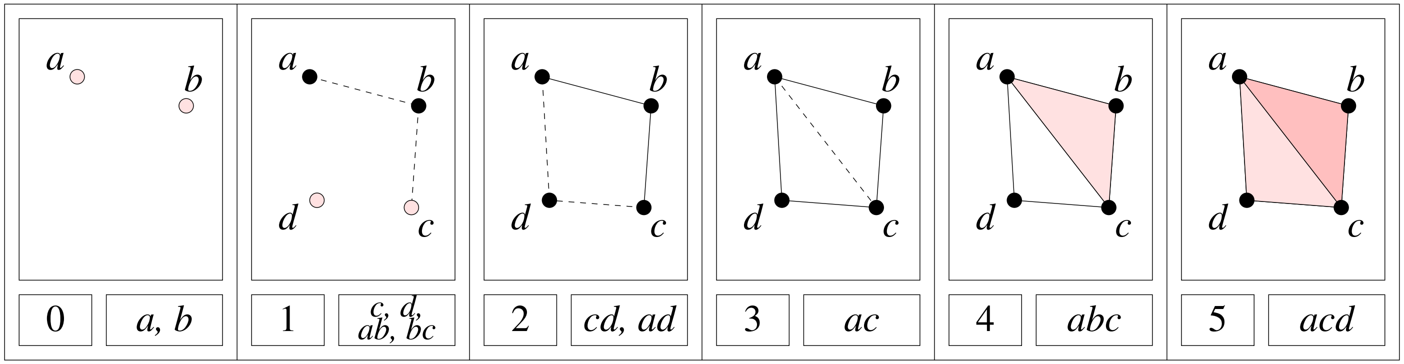
$$\tilde{M}_k = \left[\begin{array}{c|ccc|ccc}
\partial_k & e_1 & \cdots & e_{l_k} & e_{l_k+1} & \dots & e_{m_k} \\
\hline
\hat{e}_1 & b_1 & & 0 & & & \\
\vdots & & \ddots & & & 0 & \\
\hat{e}_{l_k} & 0 & & b_{l_k} & & & \\
\hline
\hat{e}_{l_k+1} & & & & & & \\
\vdots & & 0 & & & 0 & \\
\hat{e}_{m_{k-1}} & & & & & &
\end{array}\right]$$

1. the torsion coefficients of H_{k-1} are $b_i \geq 1$.
2. $\{e_i \mid l_k + 1 \leq i \leq m_k\}$ is a basis for Z_k. $\Rightarrow \text{rank}\, \mathsf{Z}_k = m_k - l_k$.
3. $\{b_i \hat{e}_i \mid 1 \leq i \leq l_k\}$ is a basis for B_{k-1}. $\Rightarrow \text{rank}\, \mathsf{B}_k = l_{k+1}$.
4. $\beta_k = \text{rank}\, \mathsf{Z}_k - \text{rank}\, \mathsf{B}_k = m_k - l_k - l_{k+1}$

In $\mathbb{S}^3$

- Algorithm takes $O(m^3)$ operations, but integers can get large
- Subcomplexes are torsion-free, so we don't need the force!
- k-chain: $c = \sum_i n_i[\sigma_i], n_i \in \mathbb{Z}, \sigma_i \in K$
- Different view, n_i are coefficients
- We can multiply, but not divide (in $\mathbb{Z}$)
- We can also change to other coefficients, such as $\mathbb{R}$, $\mathbb{Q}$, etc.
- $\mathbb{Z}_2$ Homology
  - restrict to 0,1, so unoriented simplices
  - $-\sigma = \sigma$
  - Addition is symmetric sum: $c + d = (c \cup d) - (c \cap d)$.

FILTRATION



- A filtration of a complex K is $\emptyset = K^0 \subseteq K^1 \subseteq \ldots \subseteq K^m = K$.
- A filtration is a partial ordering
- Sort according to dimension
- Break other ties arbitrarily
- Algorithm for $K = \mathbb{S}^3$

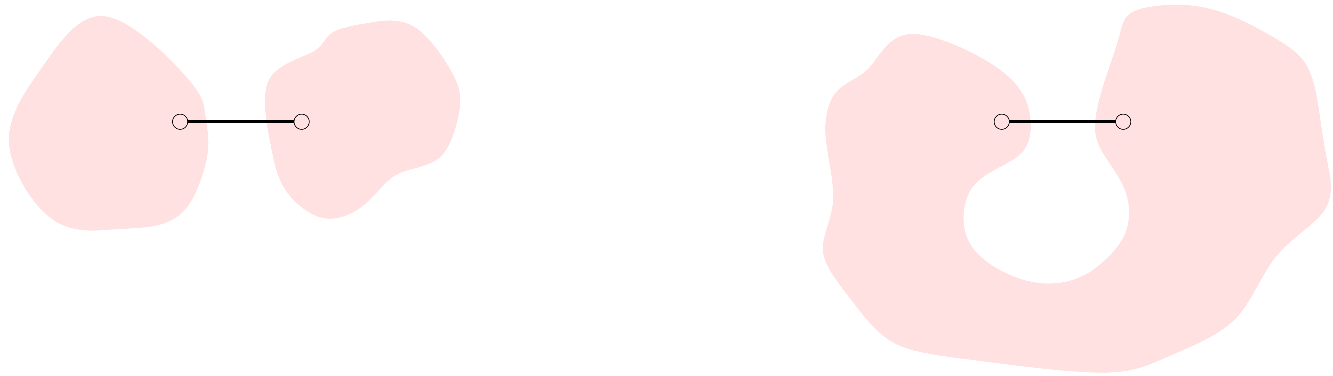
ALEXANDER DUALITY

- Alexander Duality:
  - β_0 measures the number of components of the complex.
  - β_1 is the rank of a basis for the tunnels.
  - β_2 counts the number of voids in the complex.
- An incremental approach:
  - add each simplex in turn
  - check to see if we form a new cycle class or destroy one.

VERTICES

- Vertices always add a new component, so β_0++. Why?
- Union-find data-structure:
  - MAKESET: initializes a set with an item
  - FIND: finds the set an element belongs to
  - UNION: forms the union of two sets
- Very simple to implement
- $O(n)$ space
- Amortized $\alpha(m)$ FIND, UNION
- MAKESET for each vertex

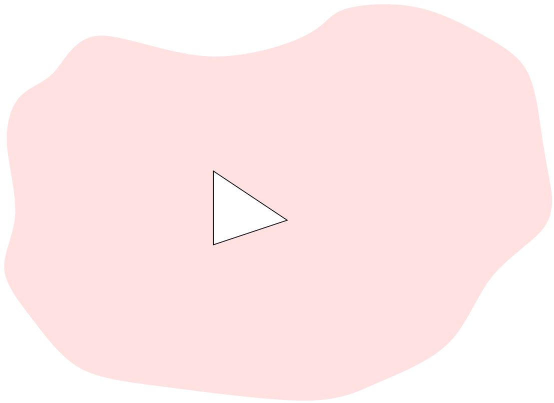
EDGES



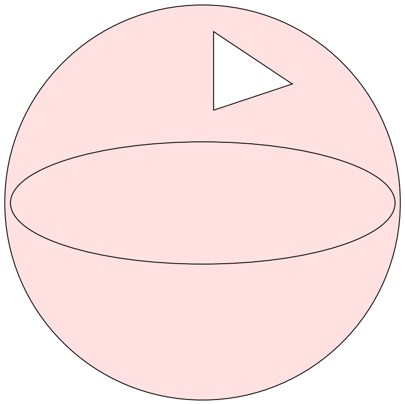
(a) $\beta_0{}^{--}$ (b) $\beta_1{}^{++}$

- (a) Two FINDS, one UNION
- (b) Two FINDS

TRIANGLES AND TETRAHEDRA



(a) $\beta_1{}^{--}$



(b) $\beta_2{}^{++}$

- Tetrahedra always fill voids, so $\beta_2{}^{--}$

COMPUTING VOIDS

$$1 \text{ void} \xrightarrow{+\text{tri}} 2 \text{ voids} \xrightarrow{+\text{tet}} 1 \text{ void} \xrightarrow{+\text{tet}} 1 \text{ component}$$

Computing Voids

$$
\begin{array}{ccccccc}
\text{1 void} & \xrightarrow{\text{+tri}} & \text{2 voids} & \xrightarrow{\text{+tet}} & \text{1 void} & \xrightarrow{\text{+tet}} & \text{1 component} \\
\downarrow \text{complement} & & \downarrow & & \downarrow & & \text{complement} \downarrow \\
\text{1 component} & \xrightarrow{\text{-tri}} & \text{2 components} & \xrightarrow{\text{-tet}} & \text{1 component} & \xrightarrow{\text{-tet}} & \text{1 void}
\end{array}
$$

- Dualize in space

COMPUTING VOIDS

$$
\begin{array}{ccccccc}
\text{1 void} & \xrightarrow{\text{+tri}} & \text{2 voids} & \xrightarrow{\text{+tet}} & \text{1 void} & \xrightarrow{\text{+tet}} & \text{1 component} \\
\downarrow \text{complement} & & \downarrow & & \downarrow & & \text{complement} \downarrow \\
\text{1 component} & \xrightarrow{\text{-tri}} & \text{2 components} & \xrightarrow{\text{-tet}} & \text{1 component} & \xrightarrow{\text{-tet}} & \text{1 void} \\
\downarrow \text{dual} & & \downarrow & & \downarrow & & \text{dual} \downarrow \\
\text{1 vertex} & \xrightarrow{\text{-edge}} & \text{2 vertices} & \xrightarrow{\text{-vertex}} & \text{1 vertex} & \xrightarrow{\text{-vertex}} & \emptyset
\end{array}
$$

- Dualize in space
- Dualize in structure

COMPUTING VOIDS

$$
\begin{array}{ccccccc}
\text{1 void} & \xrightarrow{\text{+tri}} & \text{2 voids} & \xrightarrow{\text{+tet}} & \text{1 void} & \xrightarrow{\text{+tet}} & \text{1 component} \\
\big\downarrow \text{complement} & & \big\downarrow & & \big\downarrow & & \text{complement} \big\downarrow \\
\text{1 component} & \xrightarrow{\text{-tri}} & \text{2 components} & \xrightarrow{\text{-tet}} & \text{1 component} & \xrightarrow{\text{-tet}} & \text{1 void} \\
\big\downarrow \text{dual} & & \big\downarrow & & \big\downarrow & & \text{dual} \big\downarrow \\
\text{1 vertex} & \xleftarrow{\text{+edge}} & \text{2 vertices} & \xleftarrow[\text{reverse}]{\text{+vertex}} & \text{1 vertex} & \xleftarrow{\text{+vertex}} & \emptyset
\end{array}
$$

- Dualize in space
- Dualize in structure
- Dualize in time

INCREMENTAL ALGORITHM

- Three passes:
  - One pass to identify negative edges
  - One reverse dual pass on the complement space to get positive triangles
  - One pass to compute them all (the Betti numbers)
- $O(m\alpha(m))$

dim	0	1	2
0	++		
1	−−	++	
2		−−	++
3			−−

What to Remember

- Connectivity is the dual in structure
  - Data structure must store duality
  - $O(1)$ local operations
- Knowledge of ambient space topology helps
  - Alexander duality in triangulations of $\mathbb{S}^3$
  - Instrinsic and extrinsic topology are related
- Hardest problems are when co-dimension is high
- Play with time