One-Dimensional Computational Topology

I. Encoding and decoding planar curves

Jeff Erickson
University of Illinois, Urbana-Champaign

School on Low-Dimensional Geometry and Topology:
Discrete and Algorithmic Aspects

Institut Henri Poincaré, Paris, France

June 18, 2018

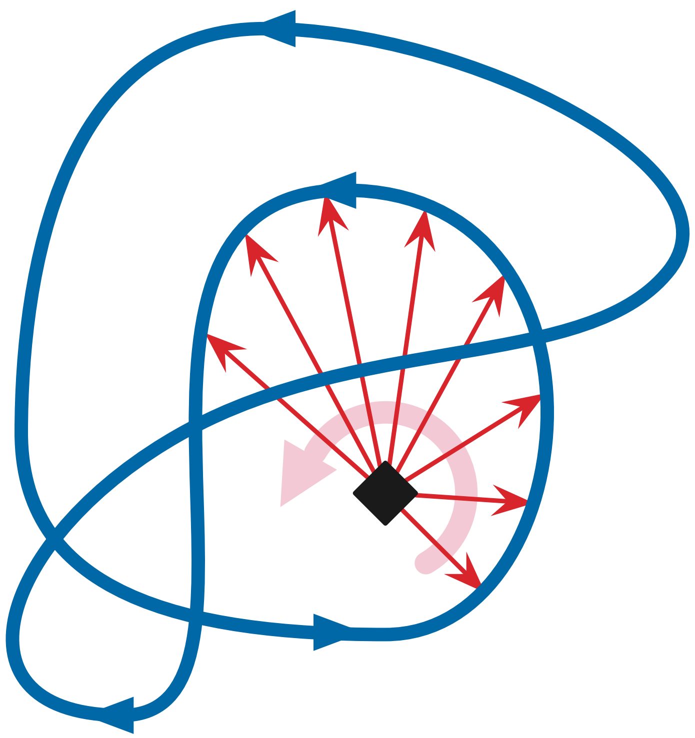
Early combinatorial topology

Geometria ſpeculatiua
Thome brauardini recoligens omnes concluſiones
geometricas ſtudentibus artium & philoſo=
phie ariſtotelis valde neceſſarias ſimul
cum quodam tractatu de qua=
dratura circuli noui=
ter edito.

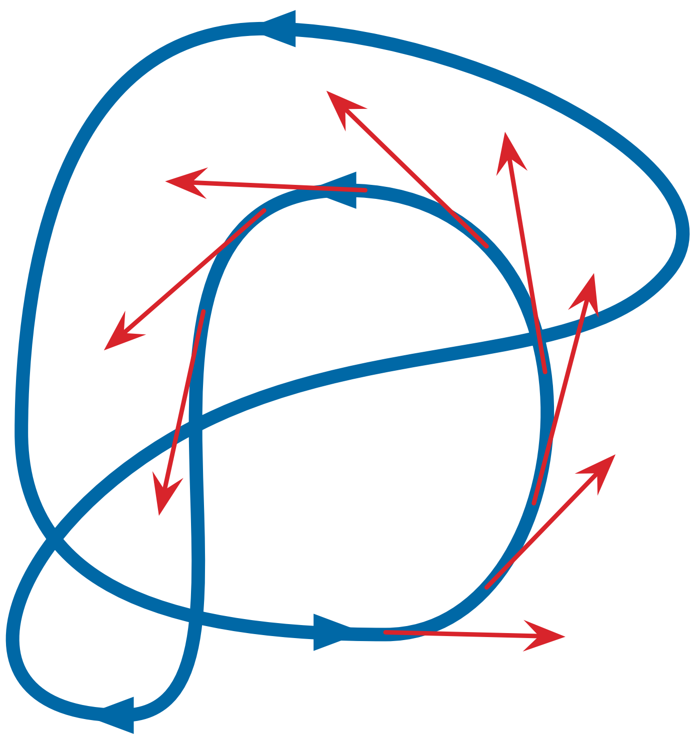
M. ALBERT. LVDOV. FRID. MEISTER
GENERALIA DE GENESI FIGVRARVM
PLANARVM,
ET
INDE PENDENTIBVS EARVM
AFFECTIONIBVS
D. VI. IAN. CIƆ IƆCCLXX.

Two invariants

- Winding number = number of times a curve winds around a point
- Rotation number = number of times the tangent vector of a curve rotates around its base



$wind(C, p) = 1$



$turn(C) = 1$

GEOMETIA·MAGISTRI·TH
OME·BRDVARDINI·INCIPIT

EOME
TRIA
ASSE
CVTI
VA·Ē
ARIT
HMETICE·NAM
POSTERIORIS
DINIS·EST·ET:
PASSIONES NVME
RORVM·IN·MAGNI
TVDINIBVS·DE
SERVIUNT·PROPTER·

Geometria Speculativa

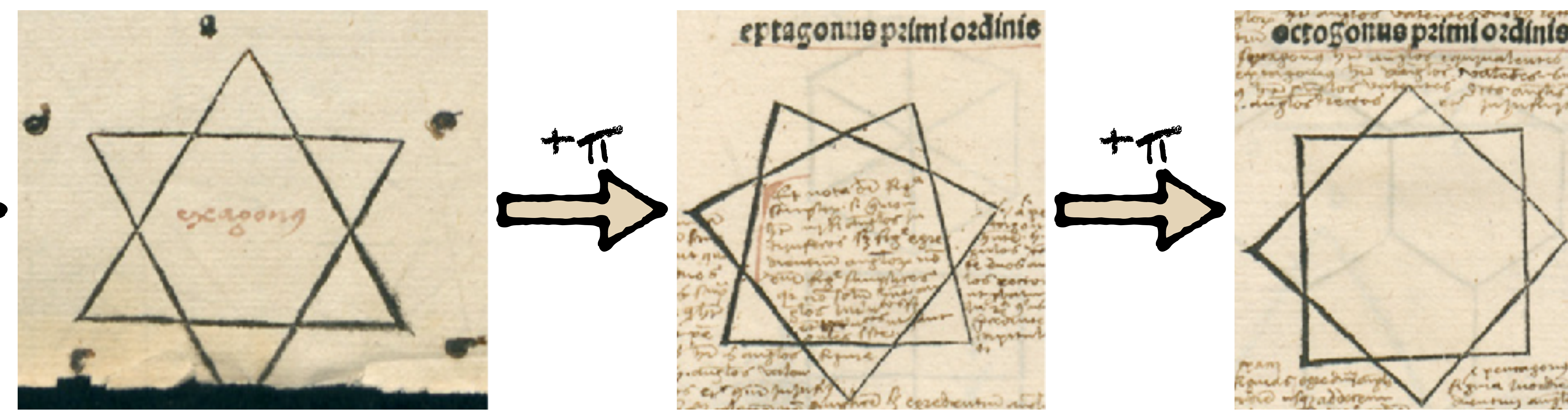
[Bradwardine c.1320]

- Interior angles of a pentagram total two right angles.
- Each additional vertex adds two right angles.
- Increasing the “order” removes four right angles.

Geometria Speculativa

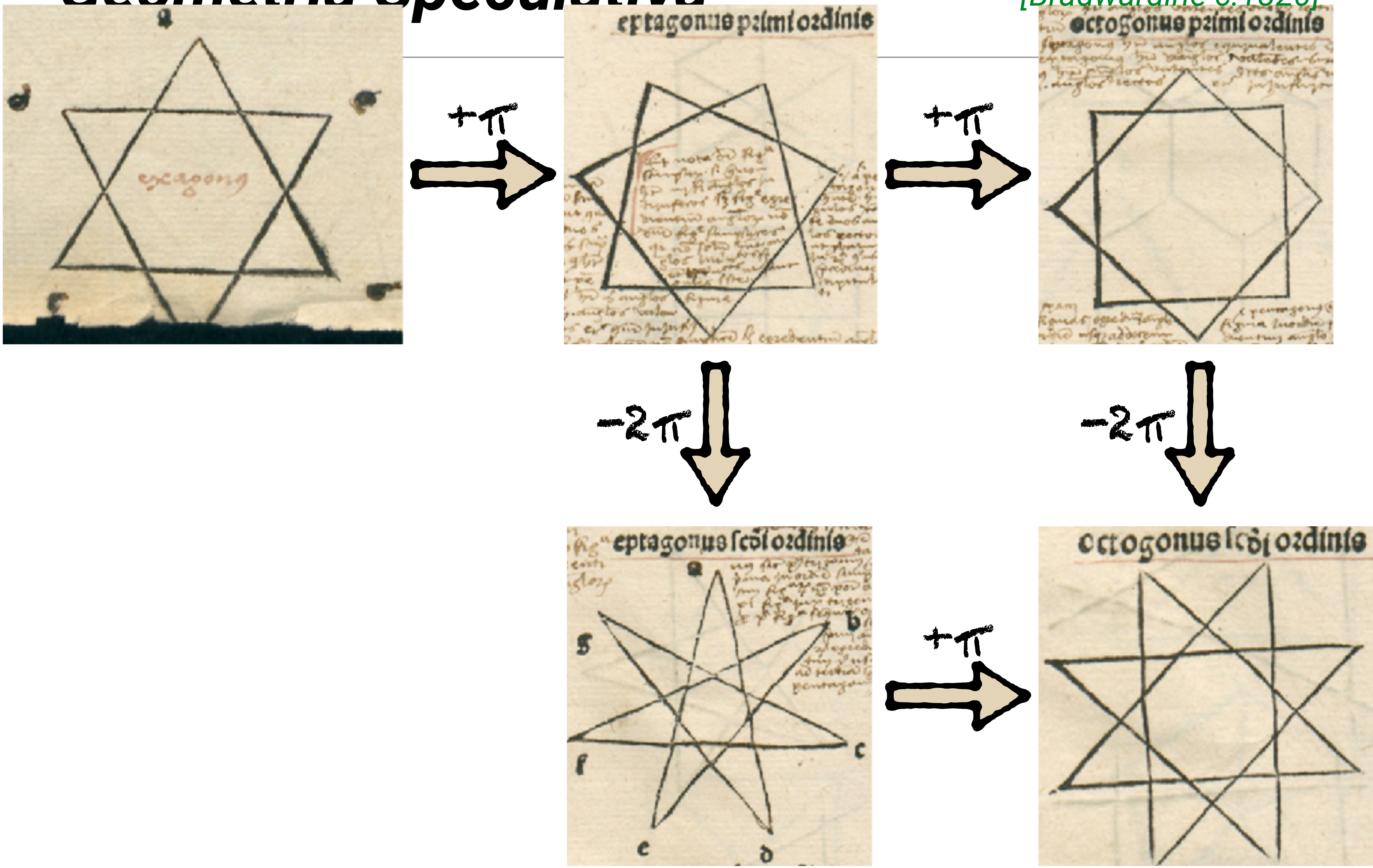
[Bradwardine c.1320]

- Interior angles of a pentagram total two right angles.
- Each additional vertex adds two right angles.
- Increasing the “order” removes four right angles.



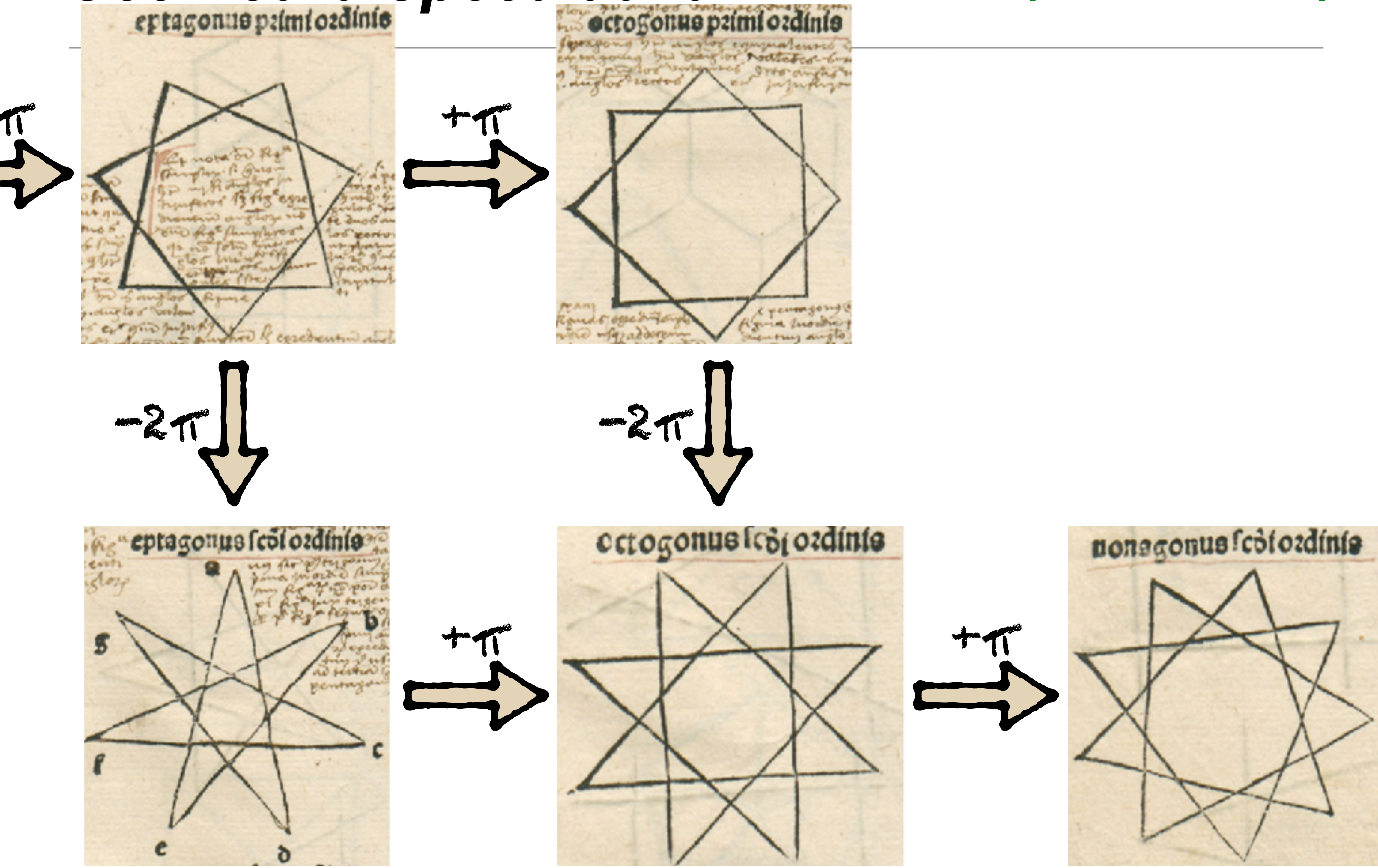
Geometria Speculativa

[Bradwardine c.1320]



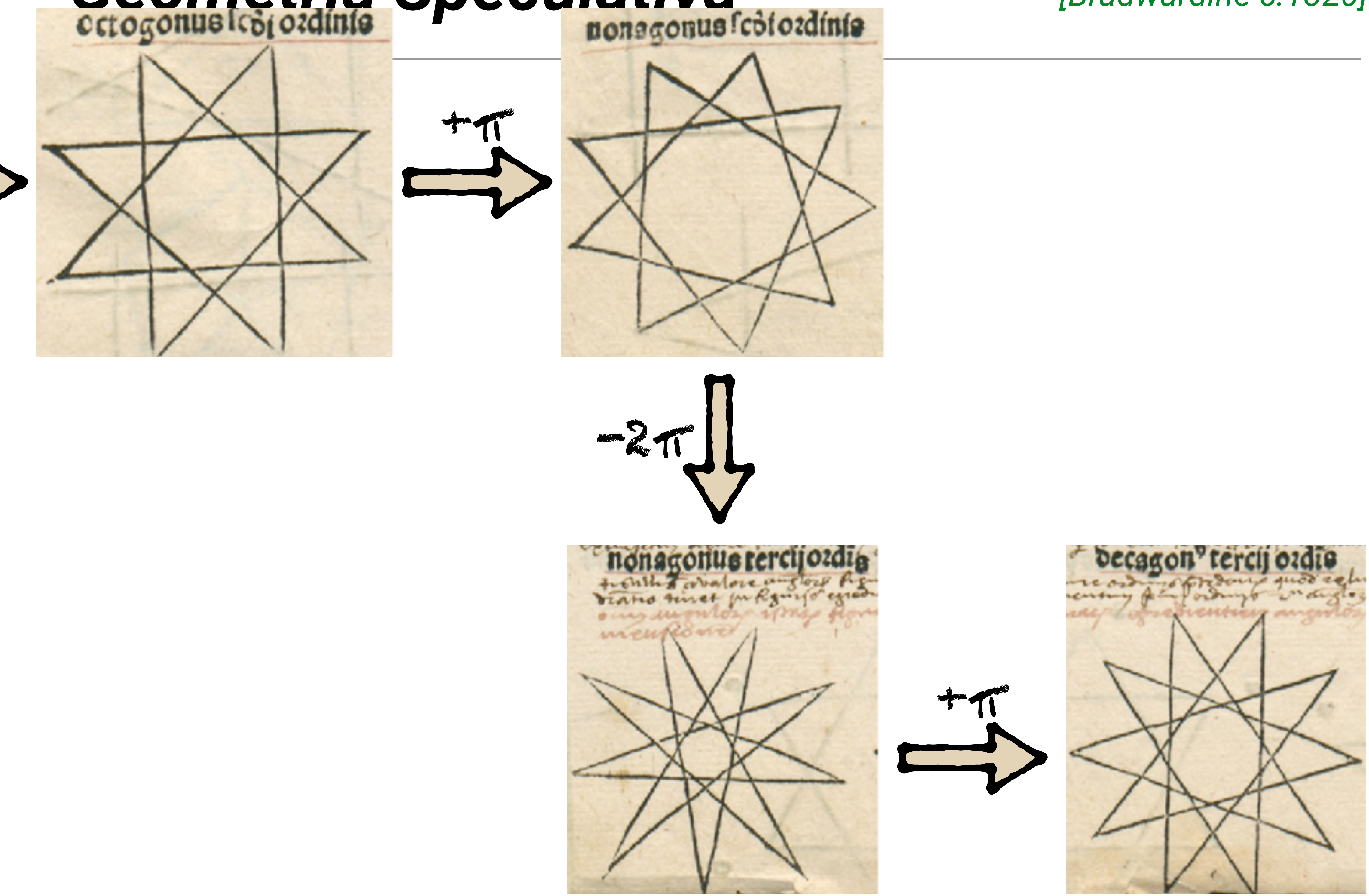
Geometria Speculativa

[Bradwardine c.1320]



Geometria Speculativa

[Bradwardine c.1320]

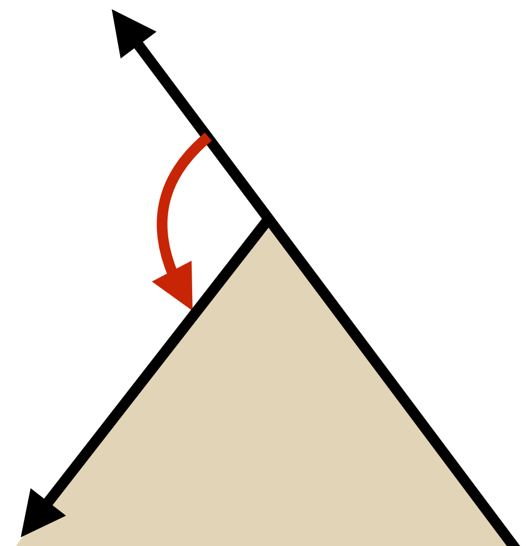


Geometria Speculativa

[Bradwardine c.1320]

- Interior angles of a pentagram total two right angles.
- Each additional vertex adds two right angles.
- Increasing the “order” removes four right angles.

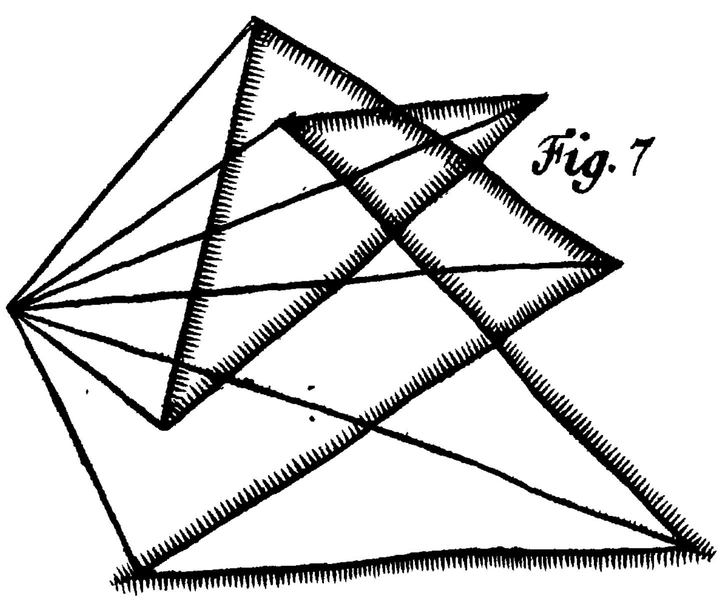
- In modern language:
  Exterior angles of a regular {*p*/*q*}-polygon sum to $2\pi q$.

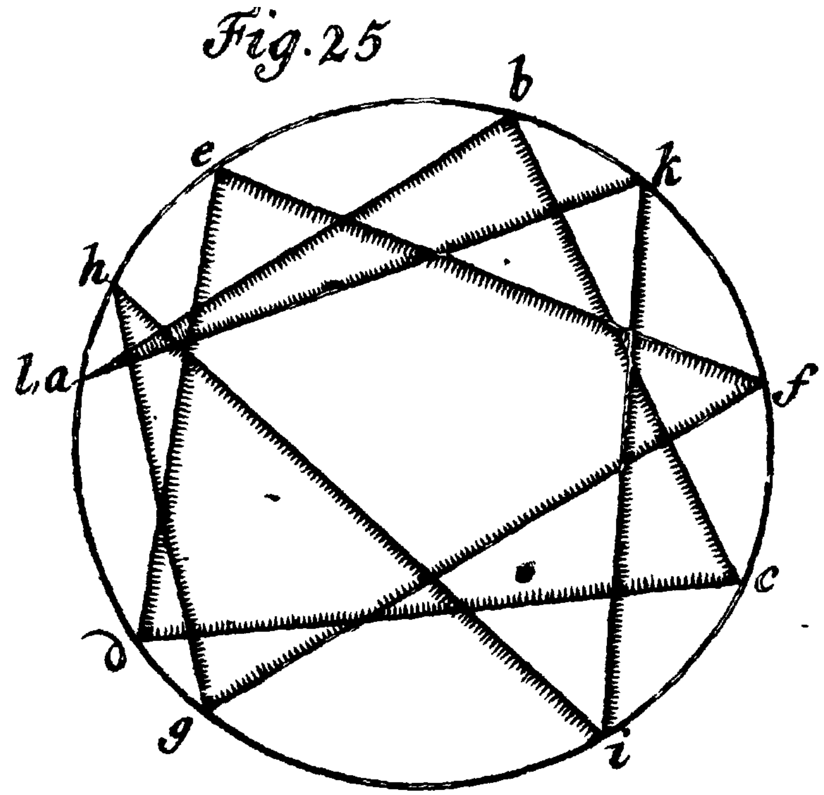


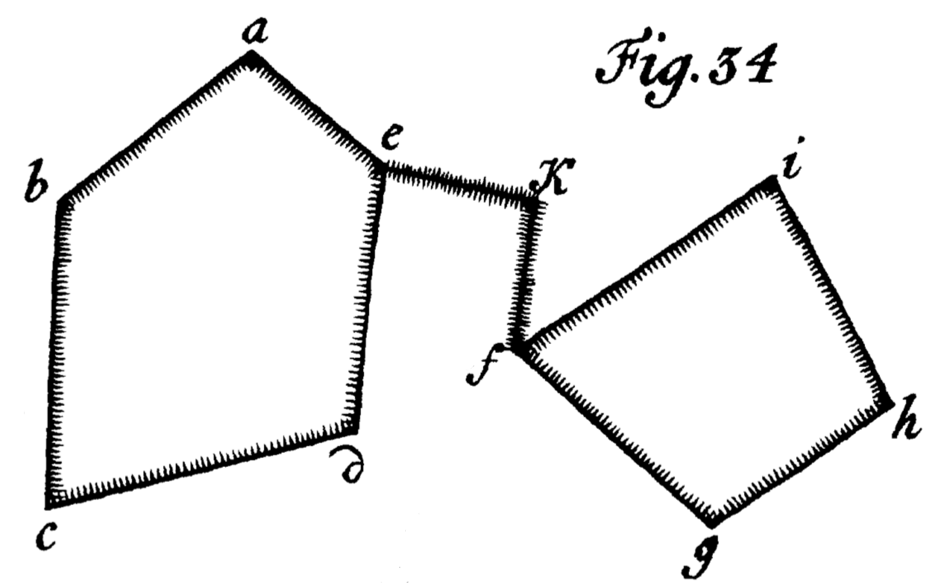
Polygon

[Meister 1770]

A cyclic sequence of points connected by line segments



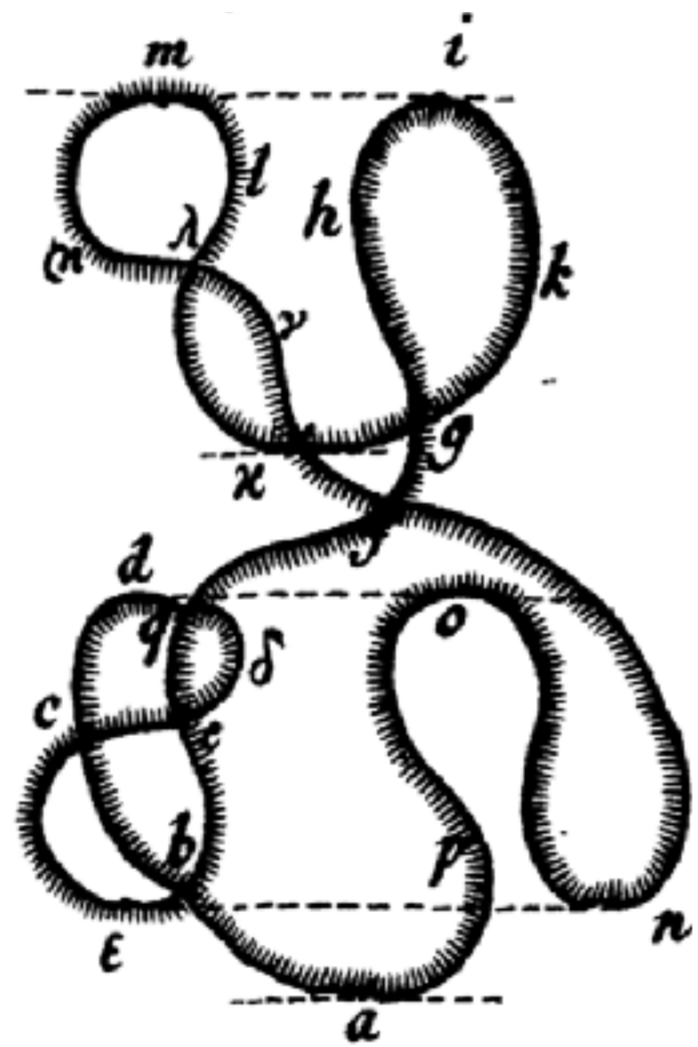




Signed area

[Meister 1770]

- Split the curve into *simple* loops at crossing points
- Add area of positive loops.
  Subtract area of negative loops.
- Contribution of any region is
  area × winding number

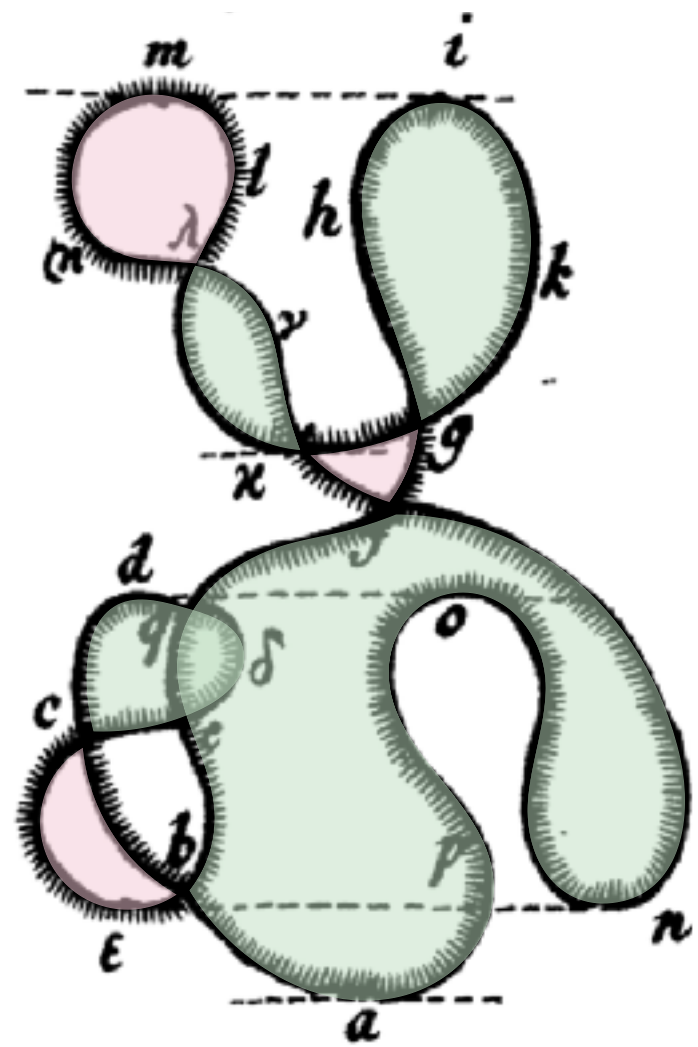


Sunt enim affirmativae + ghikg + κλνkκ + ceδqdc + abeqfnopa, ita ut pars eqδe bis cenſeatur; negativae vero — lmμλl — kgf — bεcb.

Signed area

[Meister 1770]

- Split the curve into *simple* loops at crossing points
- Add area of positive loops.
  Subtract area of negative loops.
- Contribution of any region is
  area × winding number

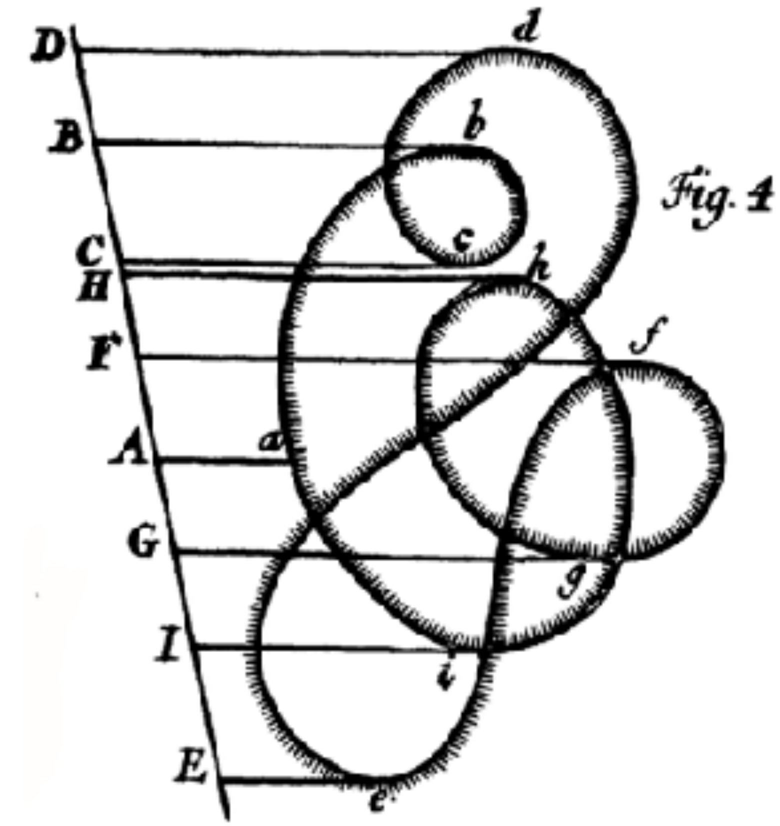


Sunt enim affirmativae + ghikg + κλνkκ + ceδqdc + abeqfnopa, ita ut pars eqδe bis cenſeatur; negativae vero — lmμλl — kgf — bεcb.

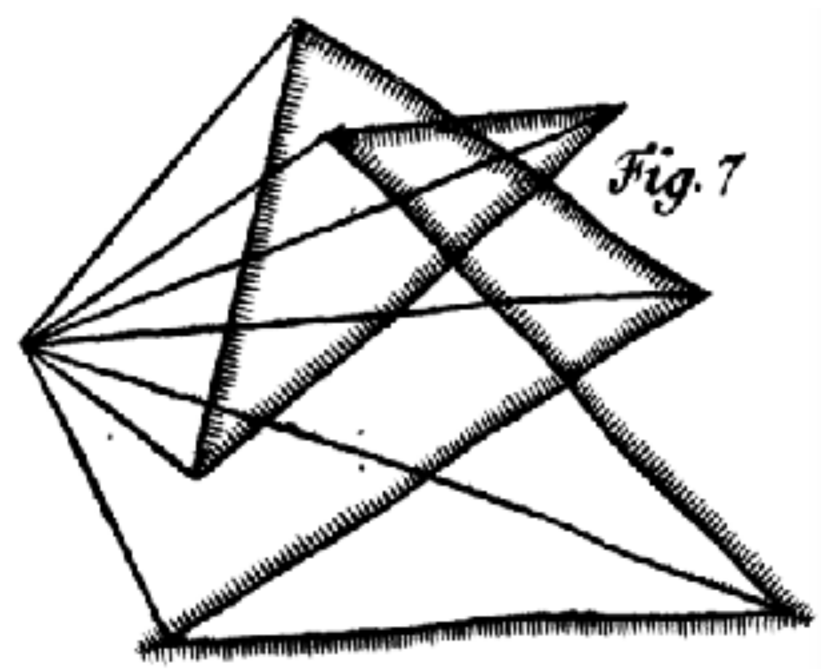
“Shoelace algorithm”

[Meister 1770]

- Computing the signed area of a curve
  - Split curve at horizontal tangent points
  - Measure area between each segment and a line
  - Add positive areas; subtract negative areas



- Signed area of a polygon =
  Sum of signed triangle areas



“Shoelace algorithm”

[Meister 1770]

- Computing the signed area of a curve
  - Split curve at horizontal tangent points
  - Measure area between each segment and a line
  - Add positive areas; subtract negative areas
- Signed area of a polygon =
  Sum of signed triangle areas

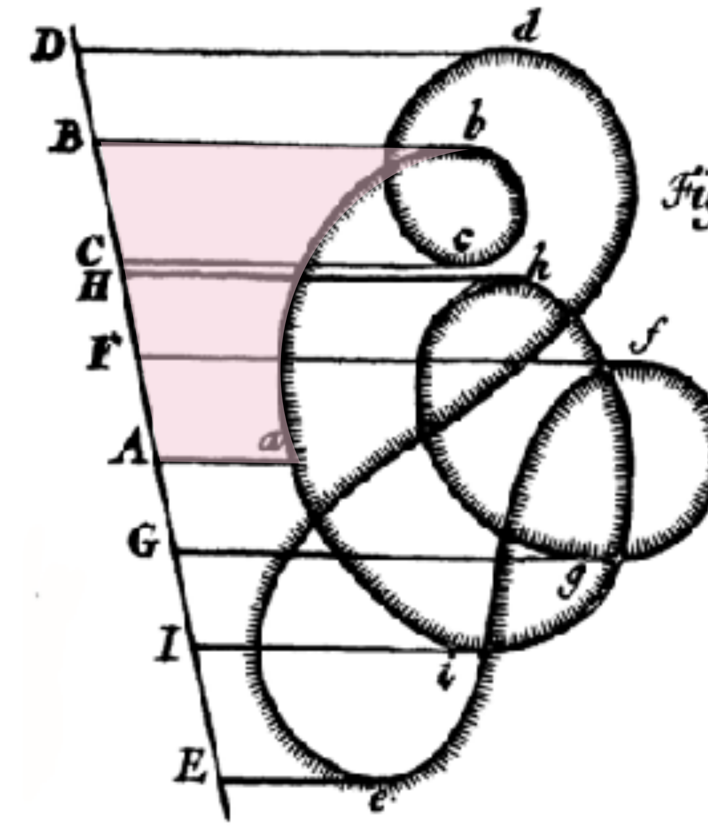


Fig. 4

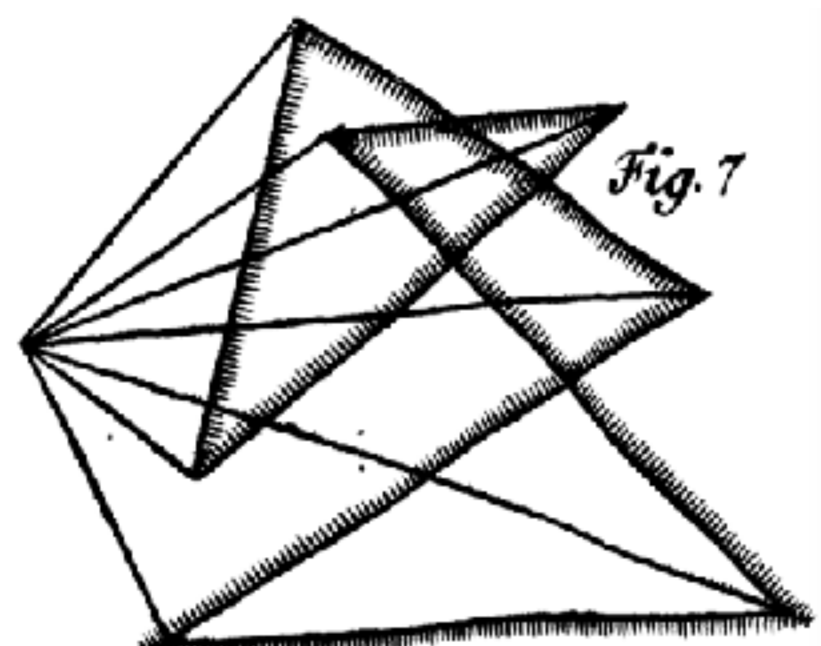
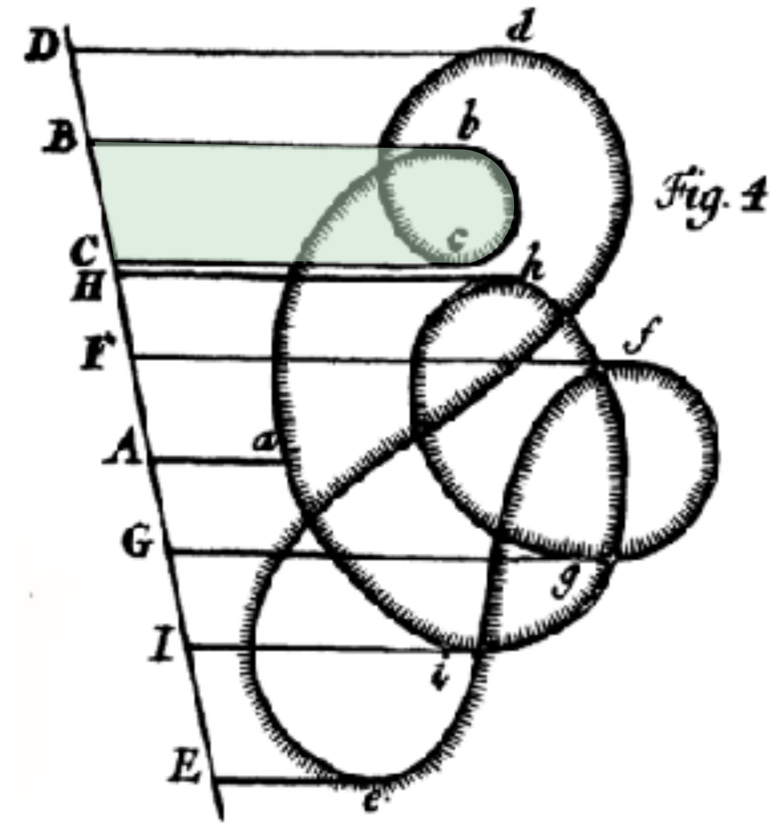


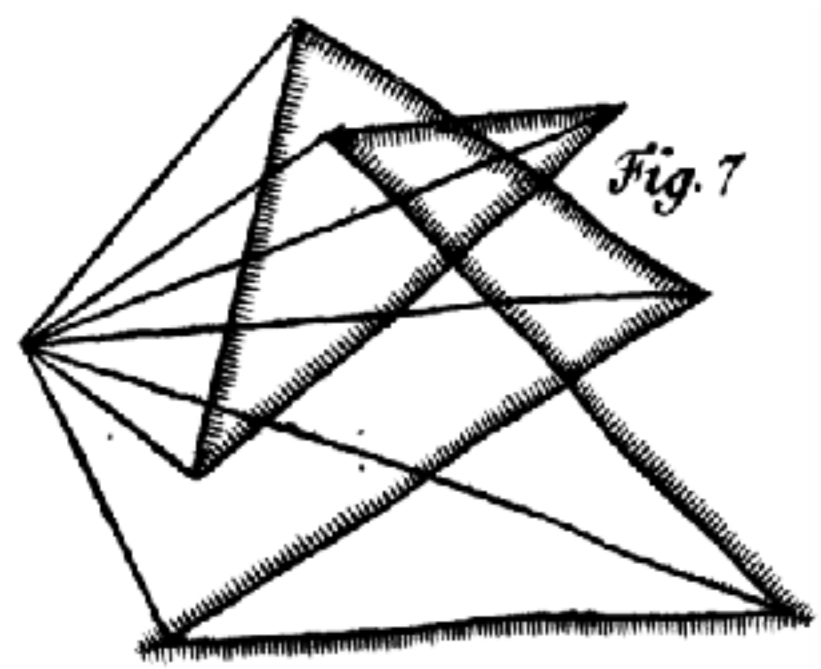
Fig. 7

“Shoelace algorithm”

[Meister 1770]

- Computing the signed area of a curve
  - Split curve at horizontal tangent points
  - Measure area between each segment and a line
  - Add positive areas; subtract negative areas
- Signed area of a polygon =
  Sum of signed triangle areas





“Shoelace algorithm”

[Meister 1770]

- Computing the signed area of a curve
  - Split curve at horizontal tangent points
  - Measure area between each segment and a line
  - Add positive areas; subtract negative areas

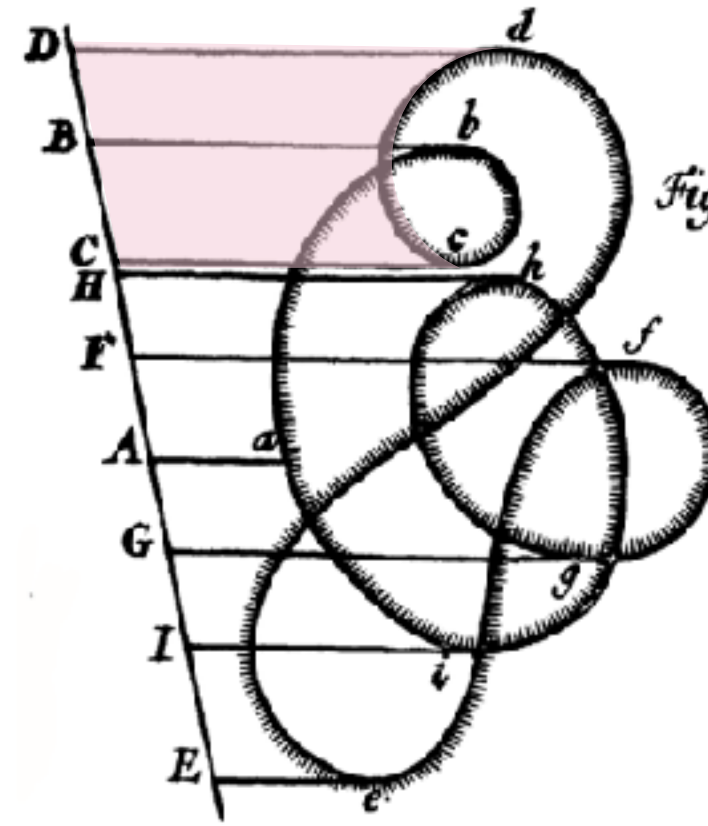


Fig. 4

- Signed area of a polygon = Sum of signed triangle areas

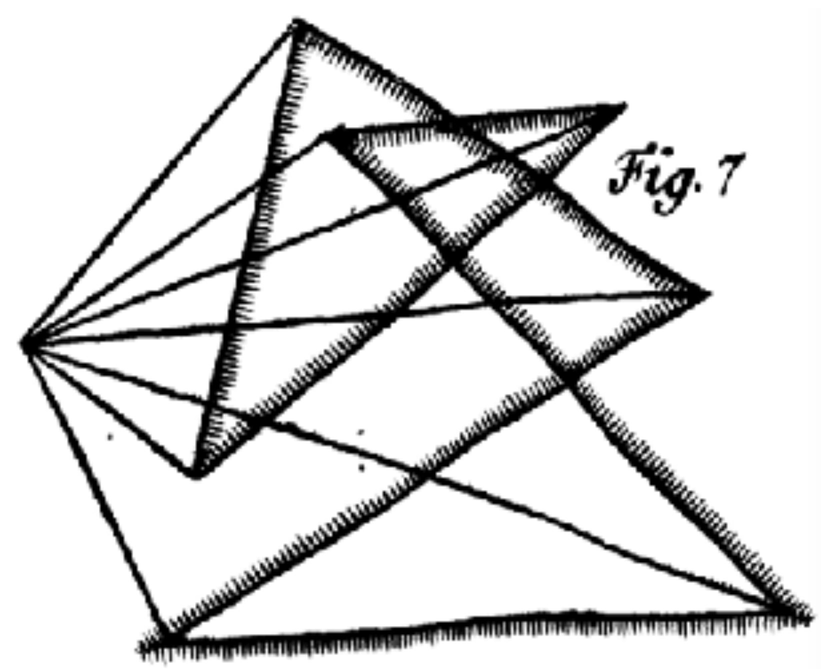
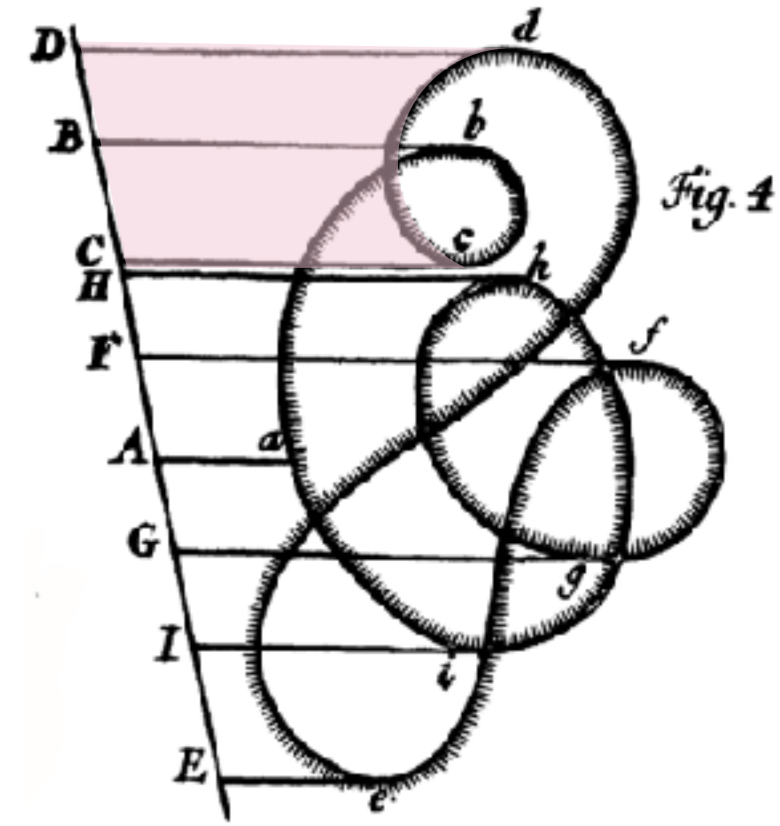


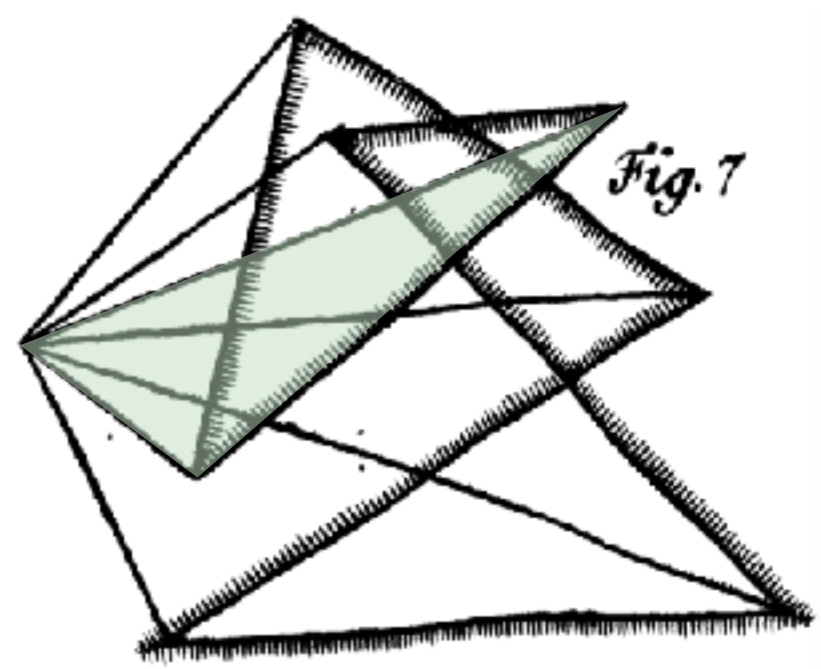
Fig. 7

“Shoelace algorithm”

[Meister 1770]

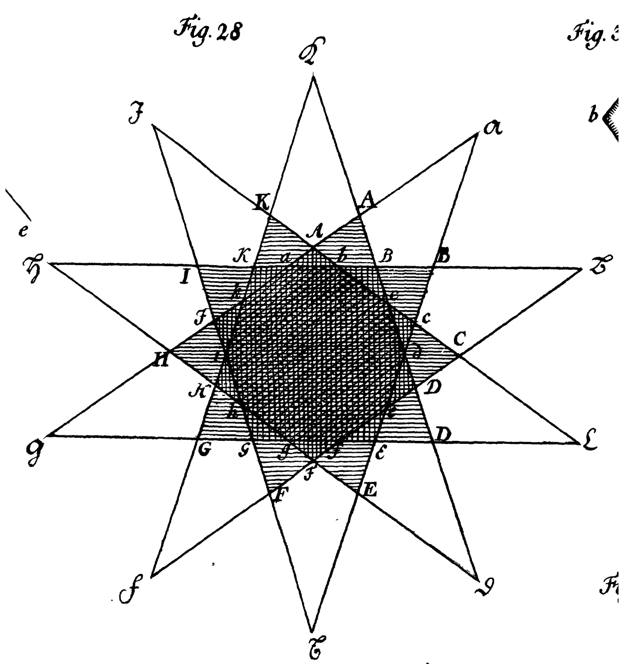
- Computing the signed area of a curve
  - Split curve at horizontal tangent points
  - Measure area between each segment and a line
  - Add positive areas; subtract negative areas
- Signed area of a polygon =
  Sum of signed triangle areas



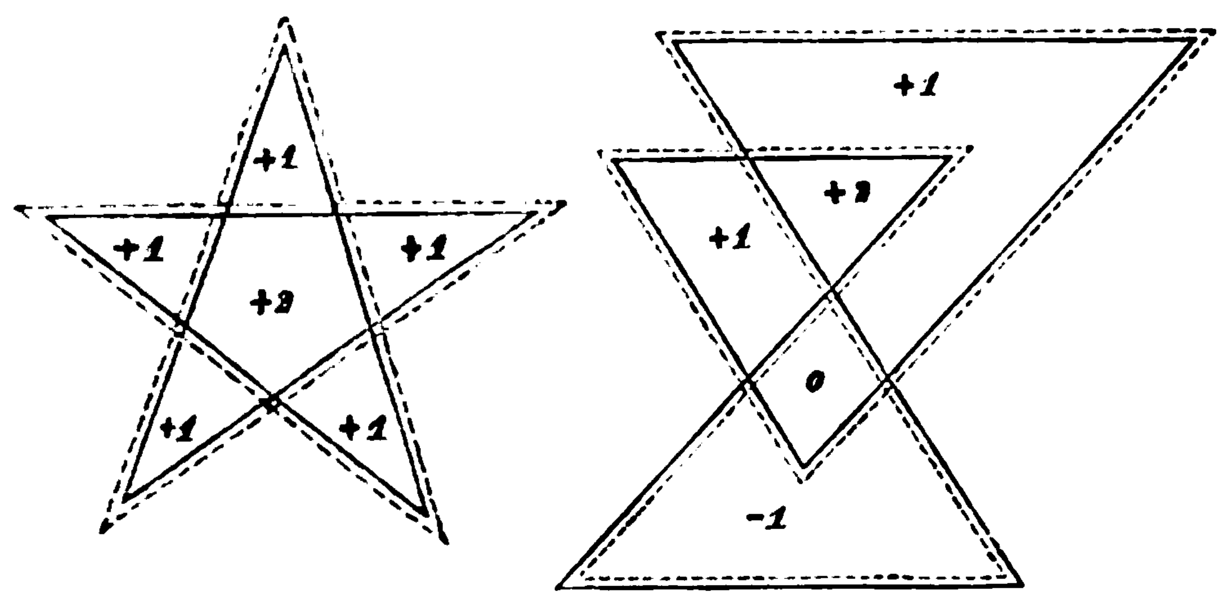


“Alexander” numbering

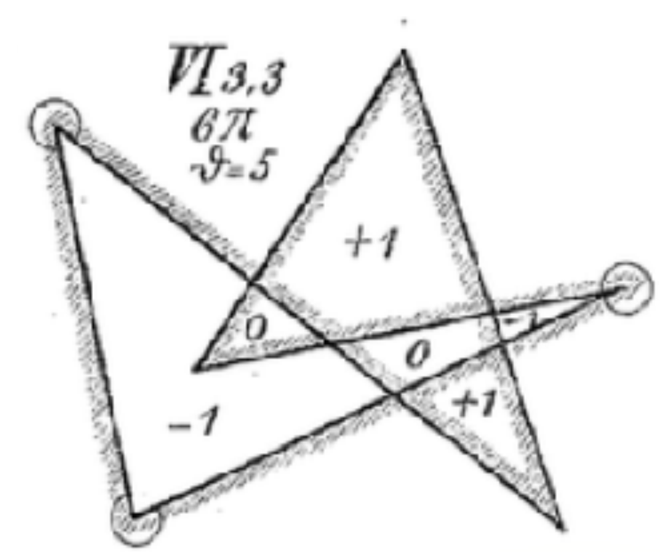
- Winding numbers are constant within each face.
- The outer face has winding number 0.
- At any regular point on the curve, the winding number on the left is 1 more than the winding number on the right.



[Meister 1770]



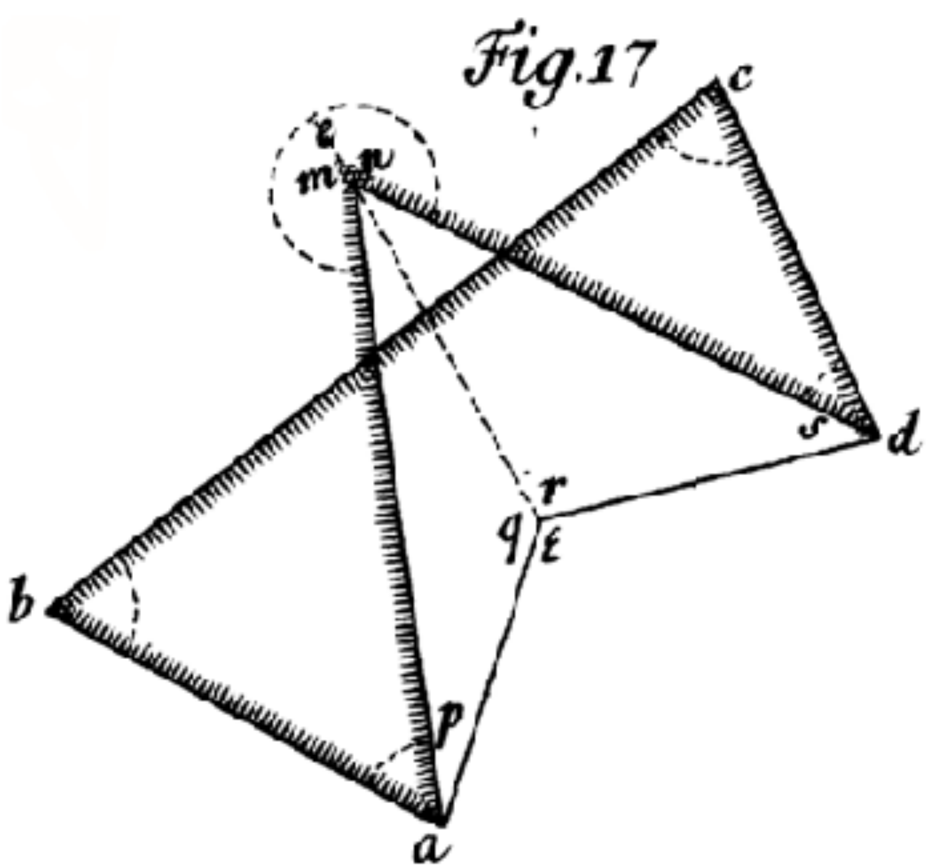
[Möbius 1865]

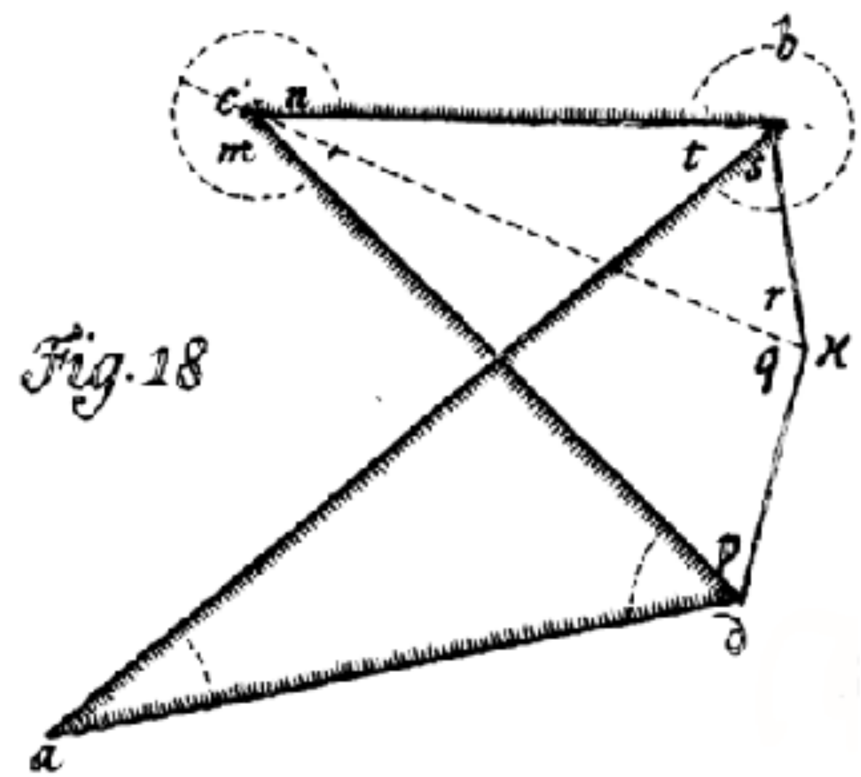


[Brückner 1900]

Rotation number

- A *topological* invariant!
  - *Fig. 17*: Moving e to ε doesn't change the sum of angles
  - *Fig. 18*: Moving c to κ changes the sum of angles by 2π.

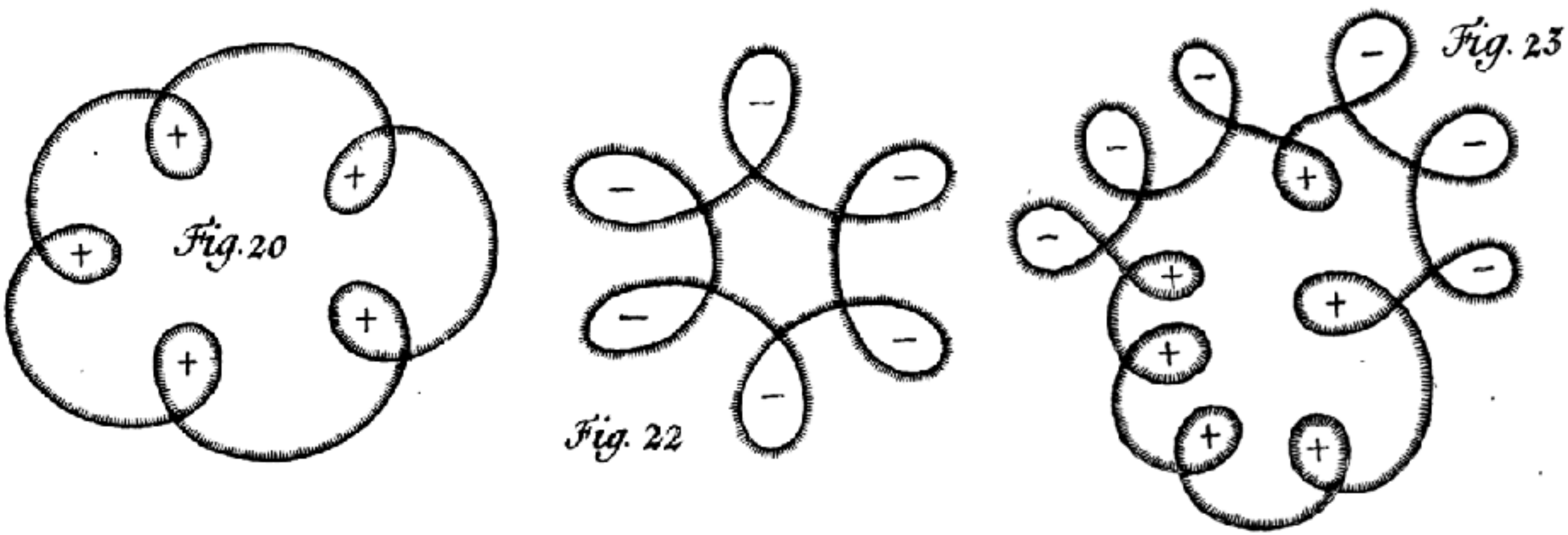




Rotation number

[Meister 1770]

- Informal statement of the *Whitney-Graustein theorem*: Two curves are *regularly homotopic* if and only if they have the same rotation number. *[Boy 1933] [Whitney 1936]*



Early *computational* topology

Point in polygon algorithm

[Gauss c.1850]

Shoot a ray to the right. If the number of positive crossings (α) equals the number of negative crossings (β), the point is outside; otherwise, the point is inside.

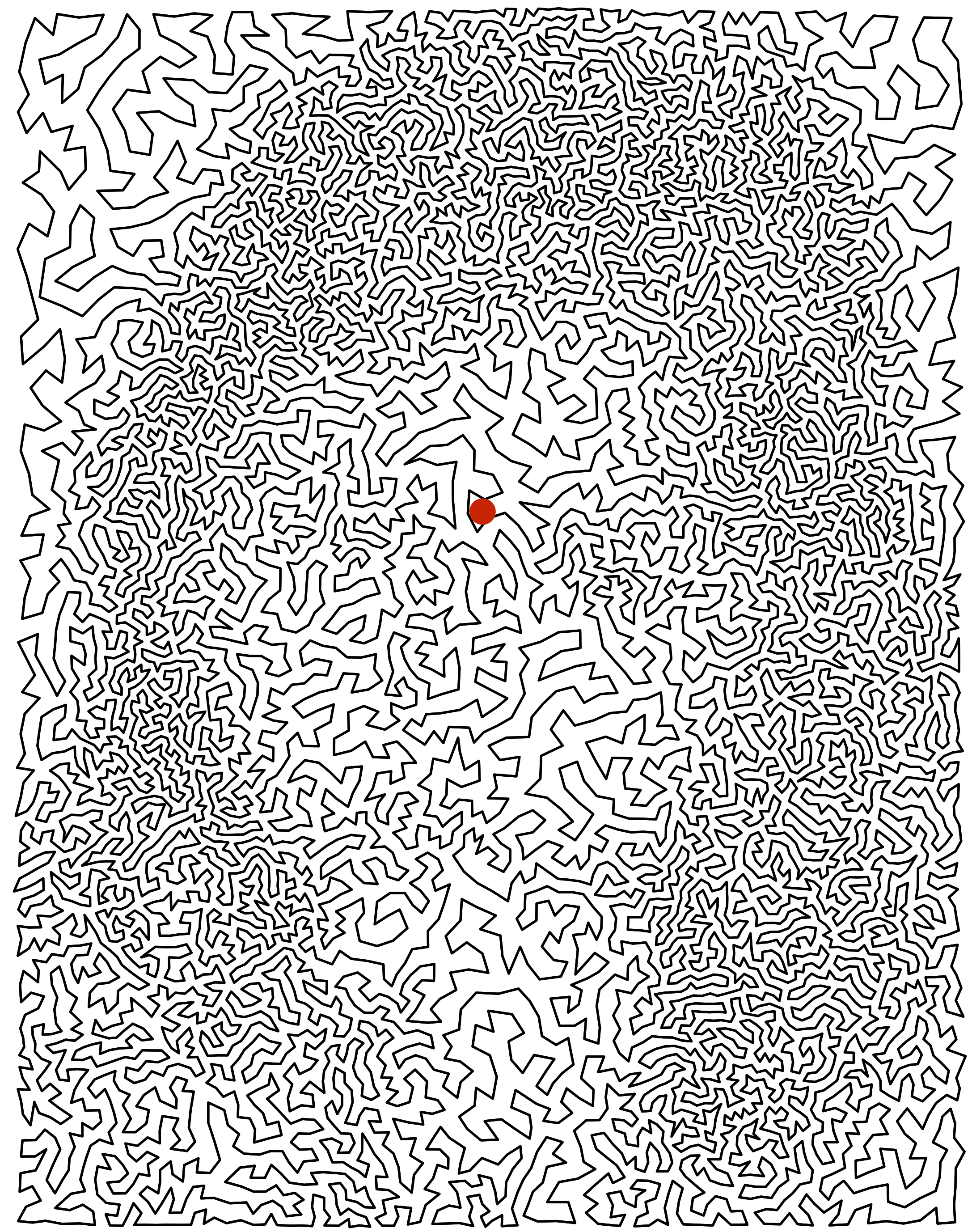
Eine interessante Aufgabe scheint zu sein, die Bedingung analytisch anzugeben, ob ein gegebener Punkt innerhalb oder ausserhalb der Figur fällt.
Die Auflösung ist leicht. Indem man den Punkt zum Anfangspunkt der Coordinaten wählt, zähle man alle Punkte

$$\begin{array}{llll} \alpha, & \text{wo} \quad y, & -y', & xy'-yx' \\ \beta, & \text{wo} \quad y, & -y', & yx'-xy' \\ \gamma, & \text{wo} \quad -y, & y', & xy'-yx' \\ \delta, & \text{wo} \quad -y, & y', & yx'-xy' \end{array}$$

positiv sind; man hat dann

$$\alpha = \gamma, \qquad \beta = \delta.$$

Ist nun $\alpha - \beta = 0$, so liegt der Punkt ausserhalb, ist $\alpha - \beta = \pm 1$, so liegt er innerhalb.



Point in polygon algorithm

[Gauss c.1850]

Shoot a ray to the right. If the number of positive crossings (α) equals the number of negative crossings (β), the point is outside; otherwise, the point is inside.

Eine interessante Aufgabe scheint zu sein, die Bedingung analytisch anzugeben, ob ein gegebener Punkt innerhalb oder ausserhalb der Figur fällt.

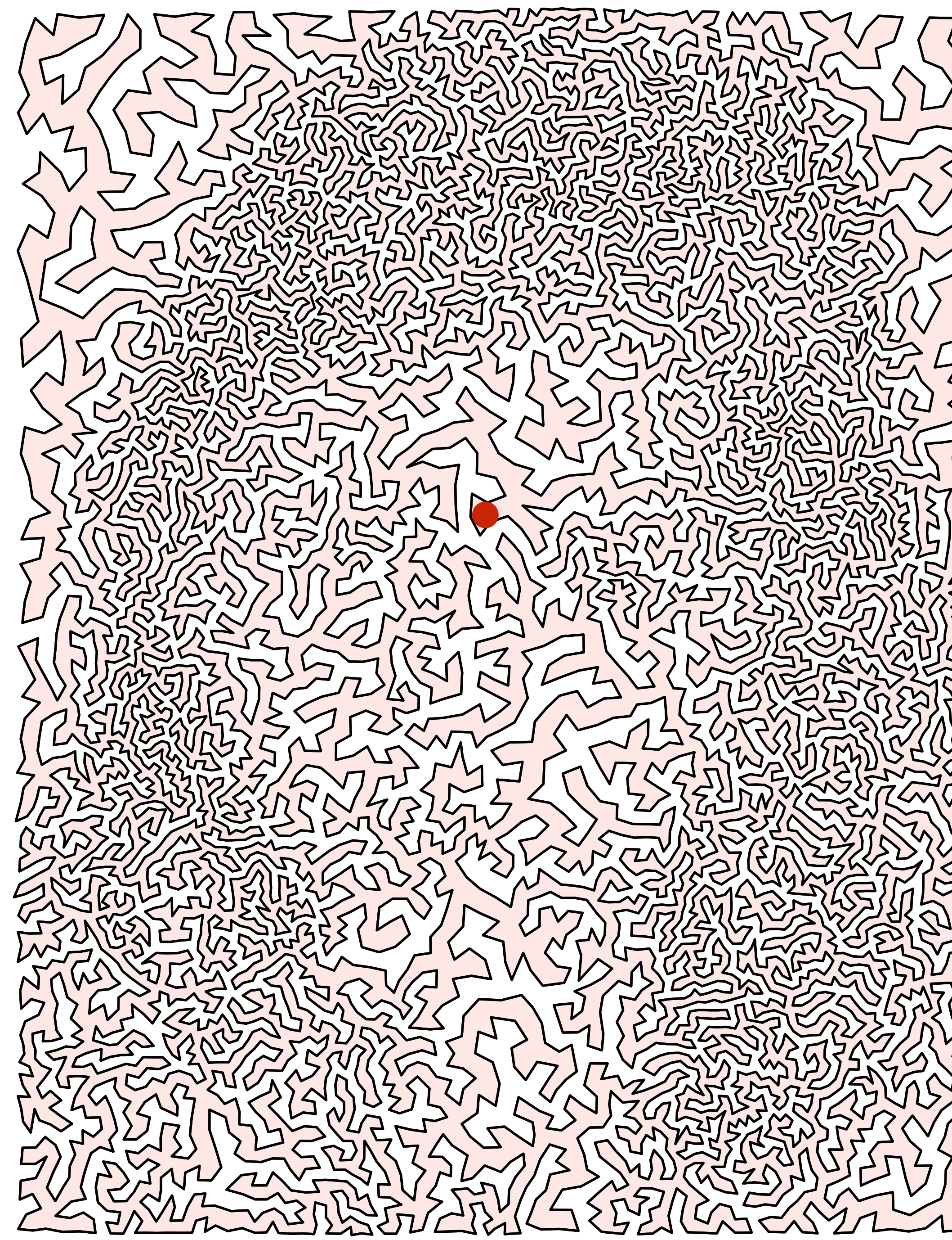
Die Auflösung ist leicht. Indem man den Punkt zum Anfangspunkt der Coordinaten wählt, zähle man alle Punkte

$$
\begin{array}{llll}
\alpha, & \text{wo} & y, \; -y', & xy' - yx' \\
\beta, & \text{wo} & y, \; -y', & yx' - xy' \\
\gamma, & \text{wo} & -y, \; y', & xy' - yx' \\
\delta, & \text{wo} & -y, \; y', & yx' - xy'
\end{array}
$$

positiv sind; man hat dann

$$\alpha = \gamma, \qquad \beta = \delta.$$

Ist nun $\alpha - \beta = 0$, so liegt der Punkt ausserhalb, ist $\alpha - \beta = \pm 1$, so liegt er innerhalb.



Point in polygon algorithm

[Gauss c.1850]

Shoot a ray to the right. If the number of positive crossings (α) equals the number of negative crossings (β), the point is outside; otherwise, the point is inside.

Eine interessante Aufgabe scheint zu sein, die Bedingung analytisch anzugeben, ob ein gegebener Punkt innerhalb oder ausserhalb der Figur fällt.

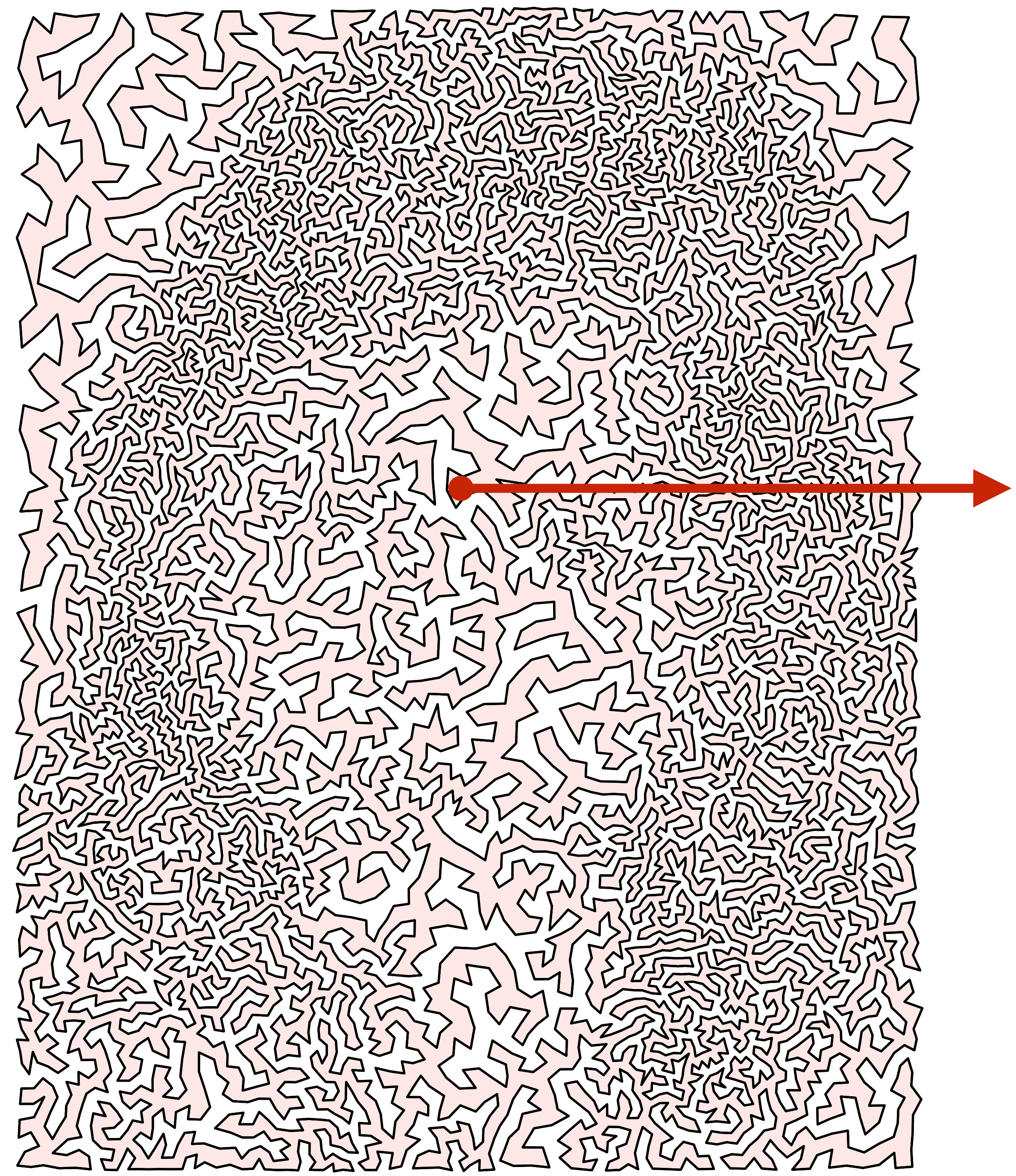
Die Auflösung ist leicht. Indem man den Punkt zum Anfangspunkt der Coordinaten wählt, zähle man alle Punkte

$$\begin{array}{llll} \alpha, & \text{wo} & y, \ -y', & xy'-yx' \\ \beta, & \text{wo} & y, \ -y', & yx'-xy' \\ \gamma, & \text{wo} & -y, \ y', & xy'-yx' \\ \delta, & \text{wo} & -y, \ y', & yx'-xy' \end{array}$$

positiv sind; man hat dann

$$\alpha = \gamma, \qquad \beta = \delta.$$

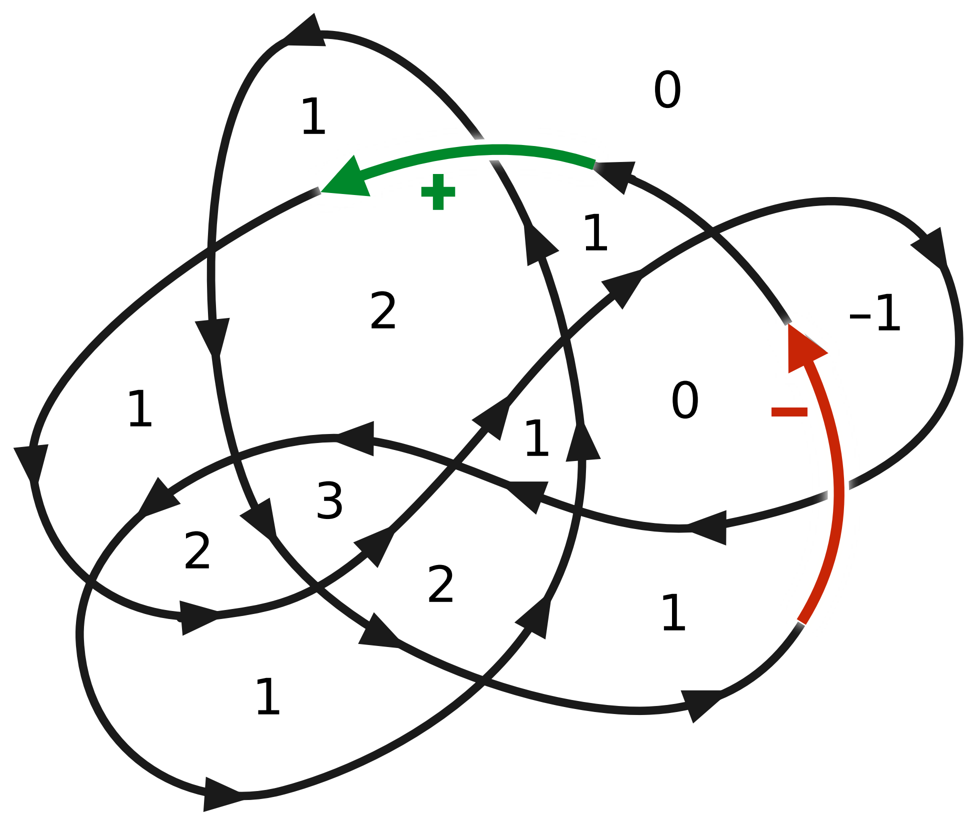
Ist nun $\alpha - \beta = 0$, so liegt der Punkt ausserhalb, ist $\alpha - \beta = \pm 1$, so liegt er innerhalb.



Signed crossings

[Gauss c. 1840]

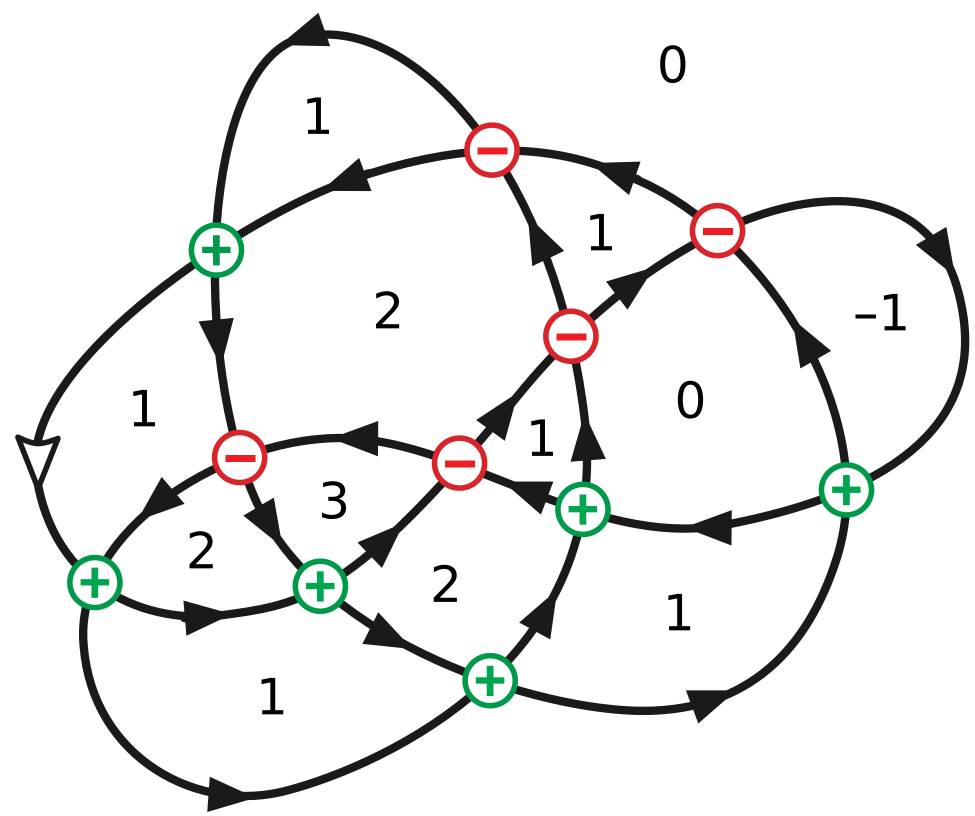
- *positive* = right to left = increasing winding number
- *negative* = left to right = decreasing winding number



Signed vertices

[Gauss c. 1840]

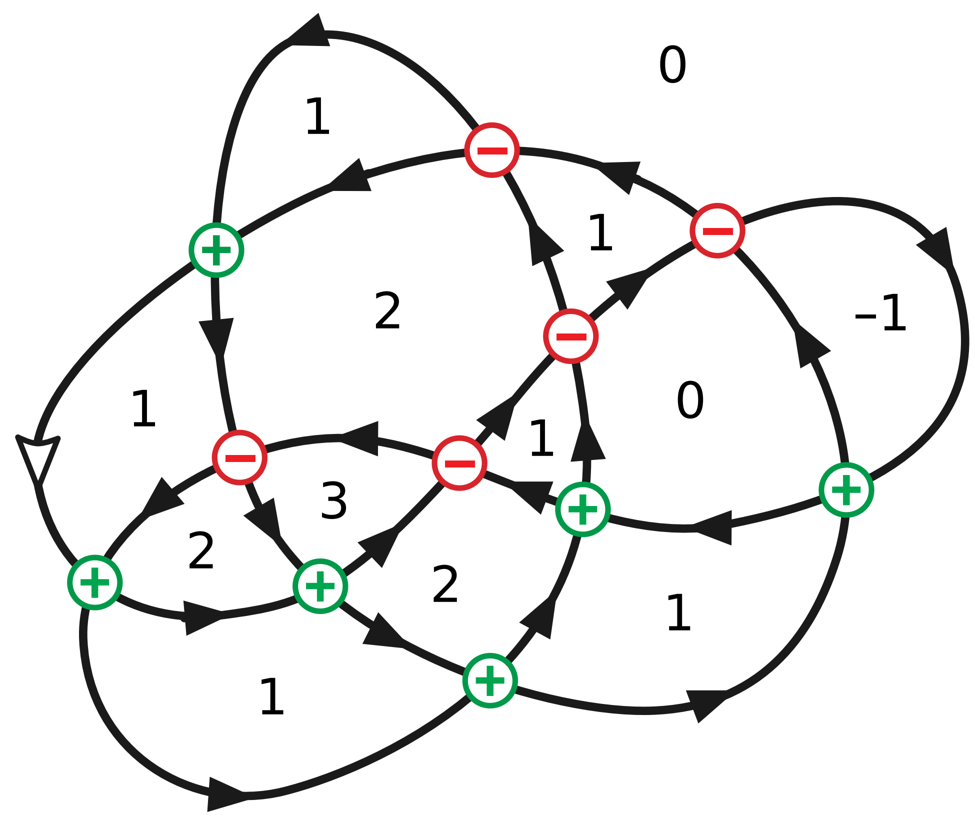
- Fix an arbitrary *basepoint*.
- The sign of a vertex is the sign of its first crossing.



Rotation number formula

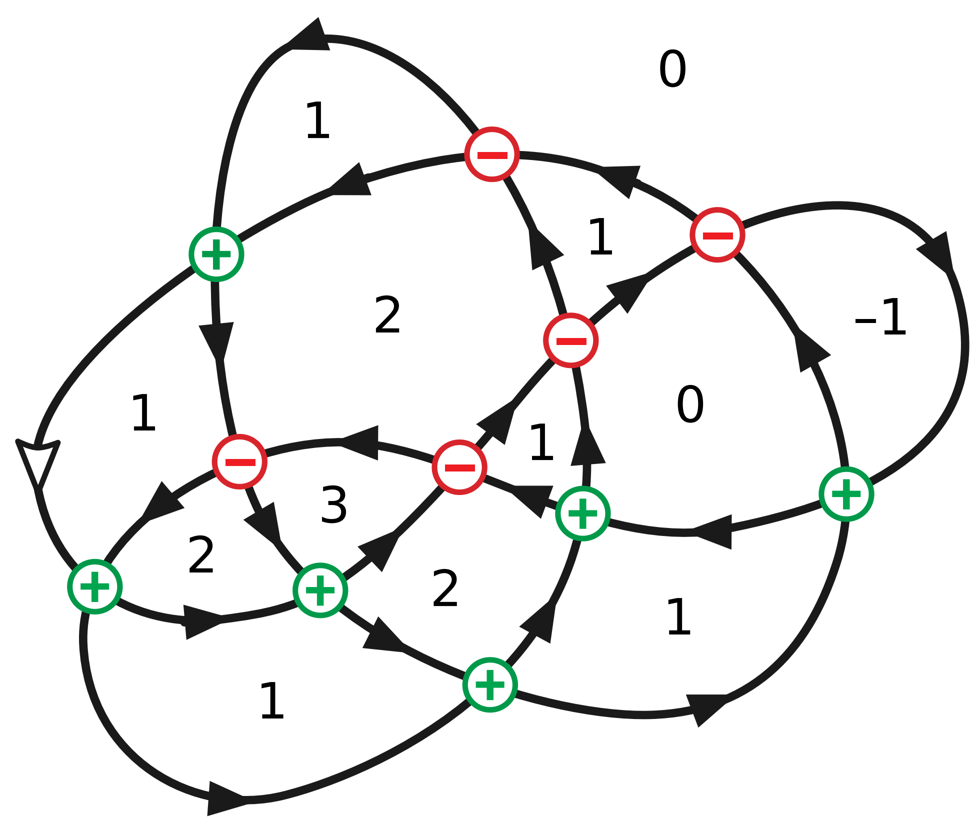
[Gauss c.1840]
[Whitney 1937] [Titus 1960]
[Grünbaum Shephard 1990]

- $\mathrm{rot}(C) = \alpha - \beta + \gamma + \gamma'$, where
  - α = number of *positive* vertices
  - β = number of *negative* vertices
  - γ, γ' = *winding numbers* on either side of the basepoint



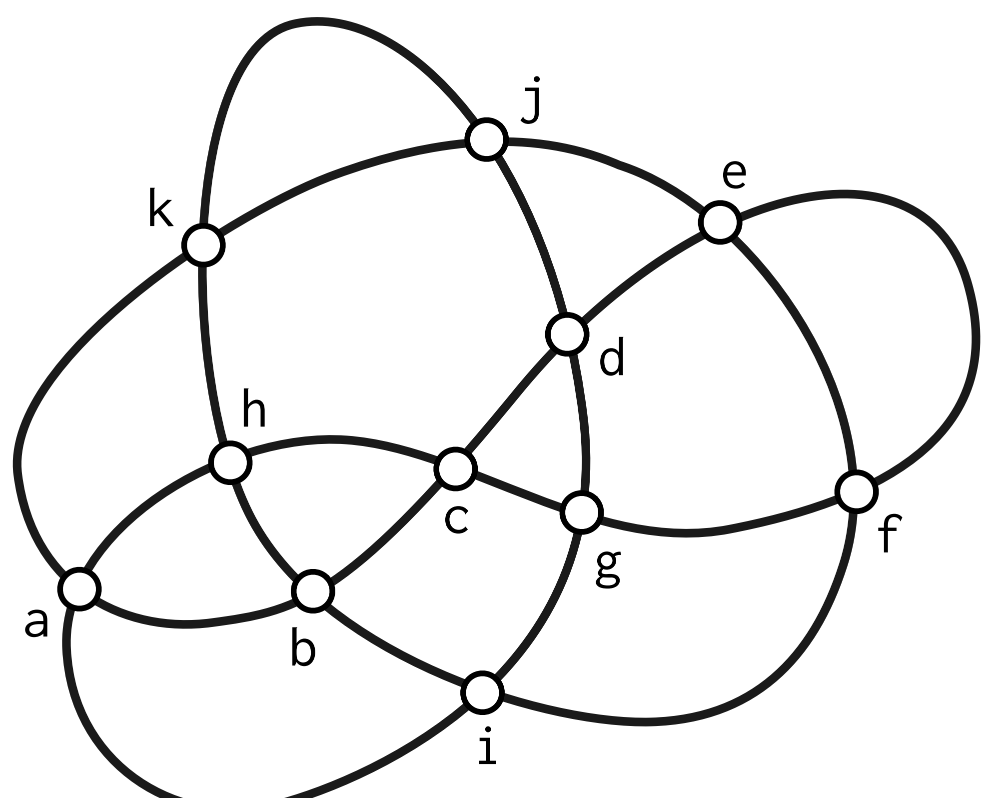
Topology!

- Positive and negative crossings/vertices, winding numbers, and rotation numbers are all *isotopy* invariants.
- We *don’t care* about coordinates, lengths, areas, angles, tangent vectors, derivatives, smoothness, ...



What is a "curve"?

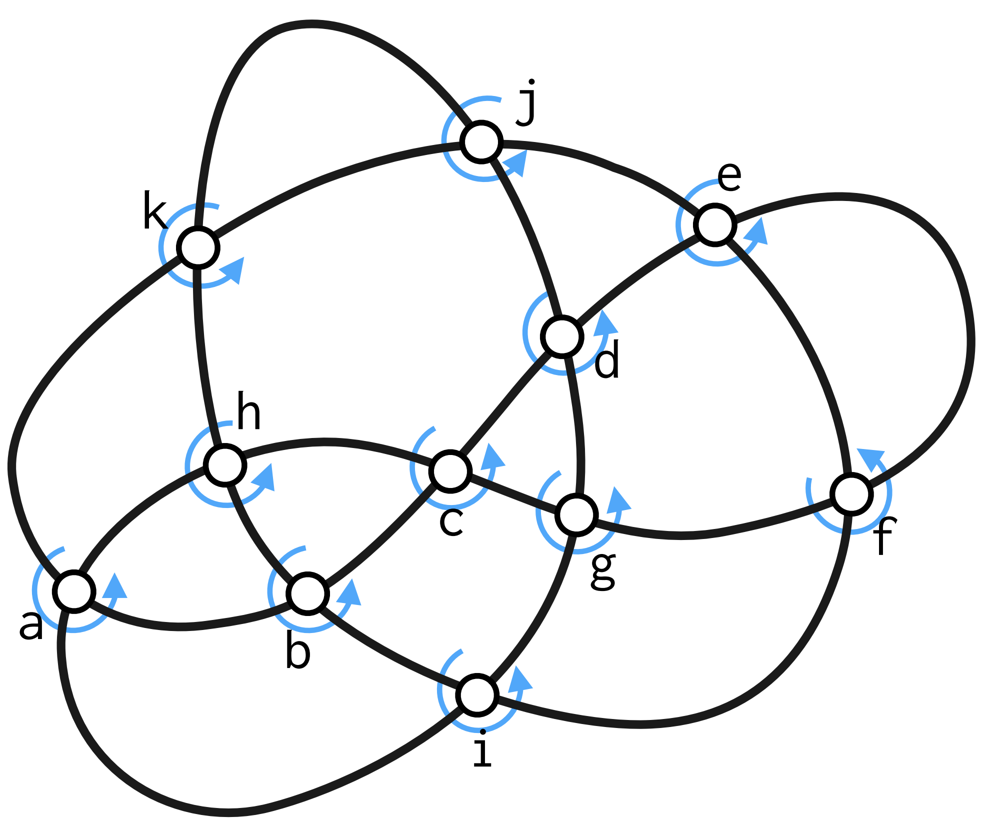
- The image of (nonsimple generic) curve is a 4-regular graph embedded in the plane.
  - Vertices = crossing points
  - Edges = curve segments between crossing points



Rotation system

[Hamilton 1856] [Kirkman 1856] [Cayley 1857] [Heffter 1891] [Brückner 1900]....

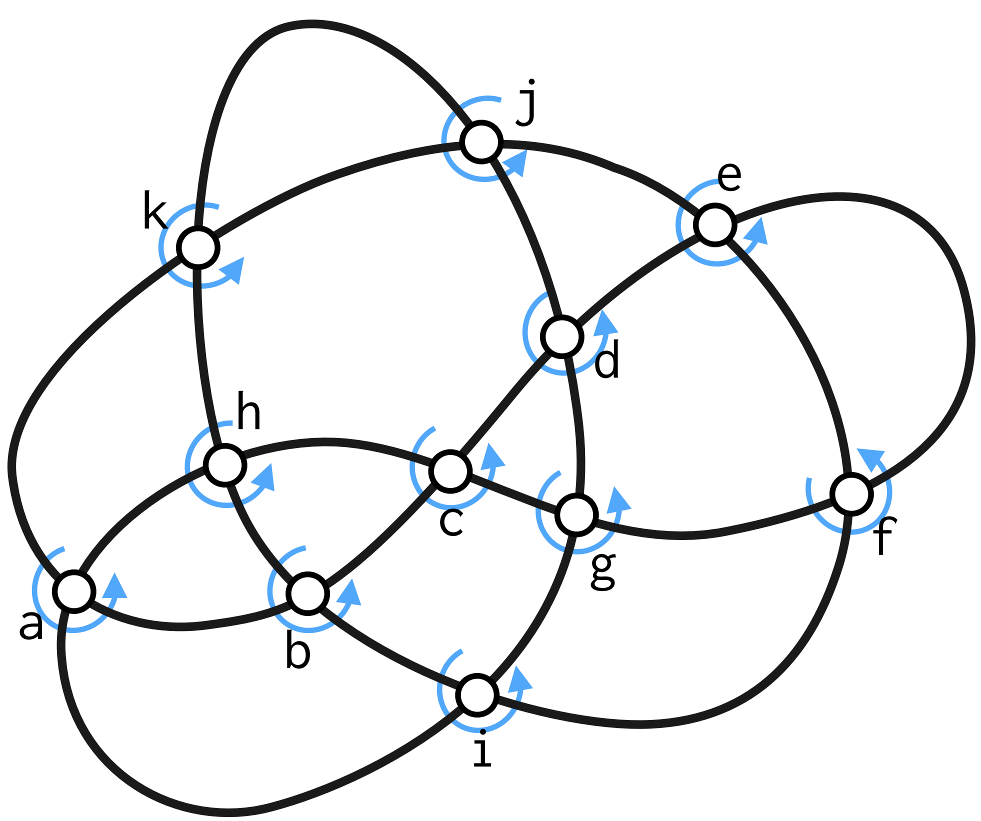
- Counterclockwise order of edges incident to each vertex.
- Specifies the embedding of G on the sphere, up to (ambient) isotopy



a	k	i	b	h
b	h	a	i	c
c	h	b	g	d
d	j	c	g	e
e	j	d	f	f
f	e	g	i	e
g	d	c	i	f
h	k	a	b	c
i	b	a	f	g
j	k	k	d	e
k	j	a	b	j

What is a “curve”?

- A curve is the “straight ahead” Euler tour of its image graph
- The rotation system is a complete isotopy invariant
- For algorithmic purposes, **a “curve” *IS* its rotation system**



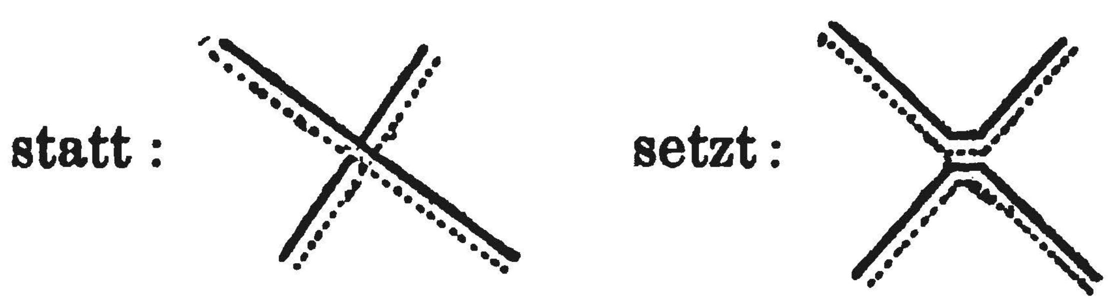
a	k	i	b	h
b	h	a	i	c
c	h	b	g	d
d	j	c	g	e
e	j	d	f	f
f	e	g	i	e
g	d	c	i	f
h	k	a	b	c
i	b	a	f	g
j	k	k	d	e
k	j	a	b	j

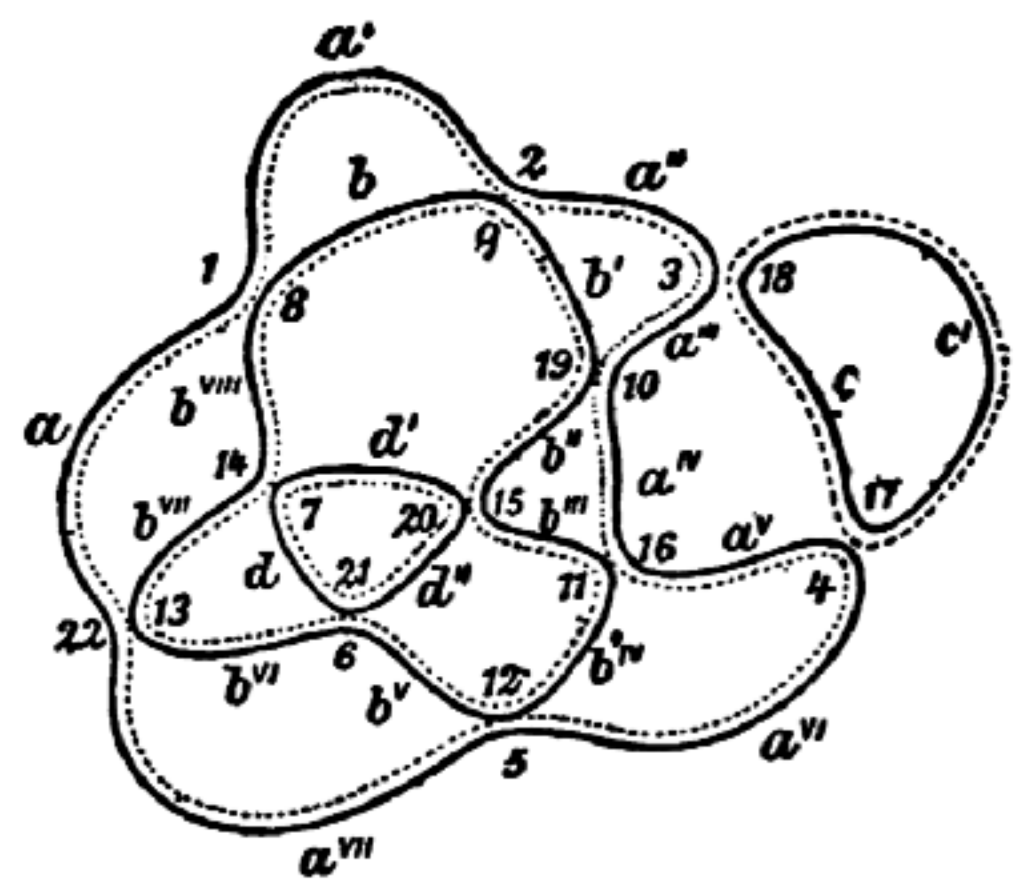
Seifert decomposition

[Gauss c.1845]
[Jacobi c.1850] [Wiener 1865]
[Hermes 1866] [Steinitz 1916]

Uncross/smooth/resolve the curve at every crossing, preserving orientation

- Winding number = sum of individual winding numbers
- Rotation number = sum of individual rotation numbers

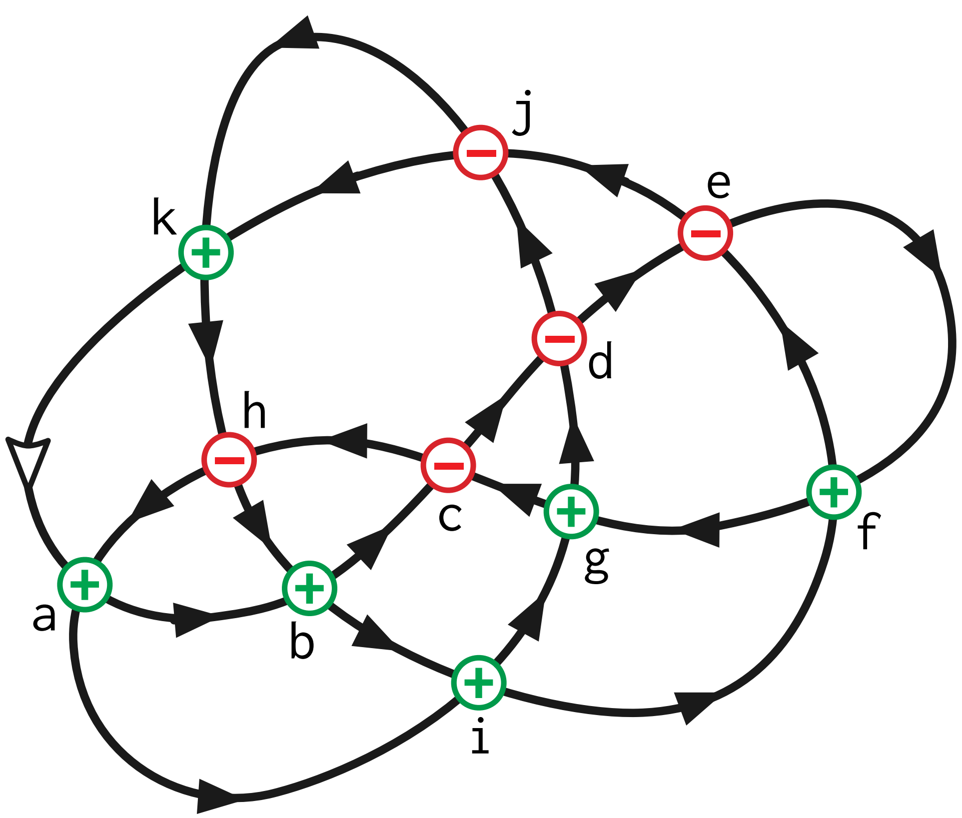




Gauss code

[Gauss c. 1840]

Sequence of crossing labels, either with or without signs

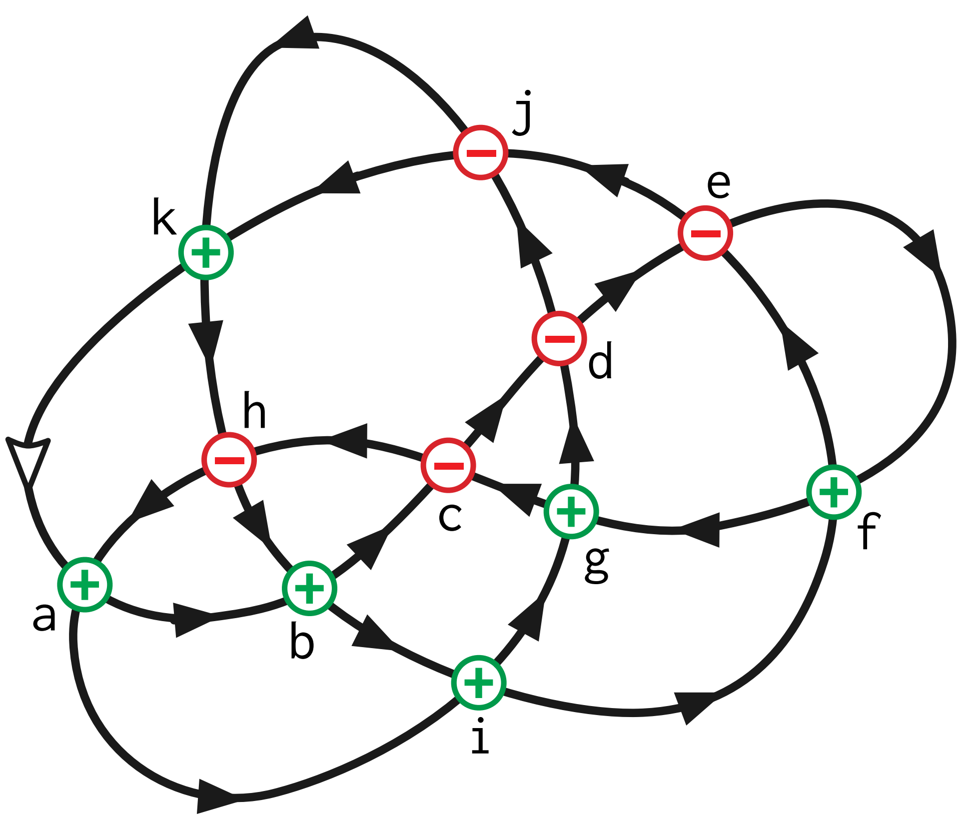



```
++−−−+++−−+−+−++−−−++−
abcdefgchaigdjkhbifejk
```

Gauss’ problem

[Gauss 1844]

Which Gauss codes correspond to planar curves?



++−−−+++−−+−+−++−−−++−

abcdefgchaigdjkhbifejk

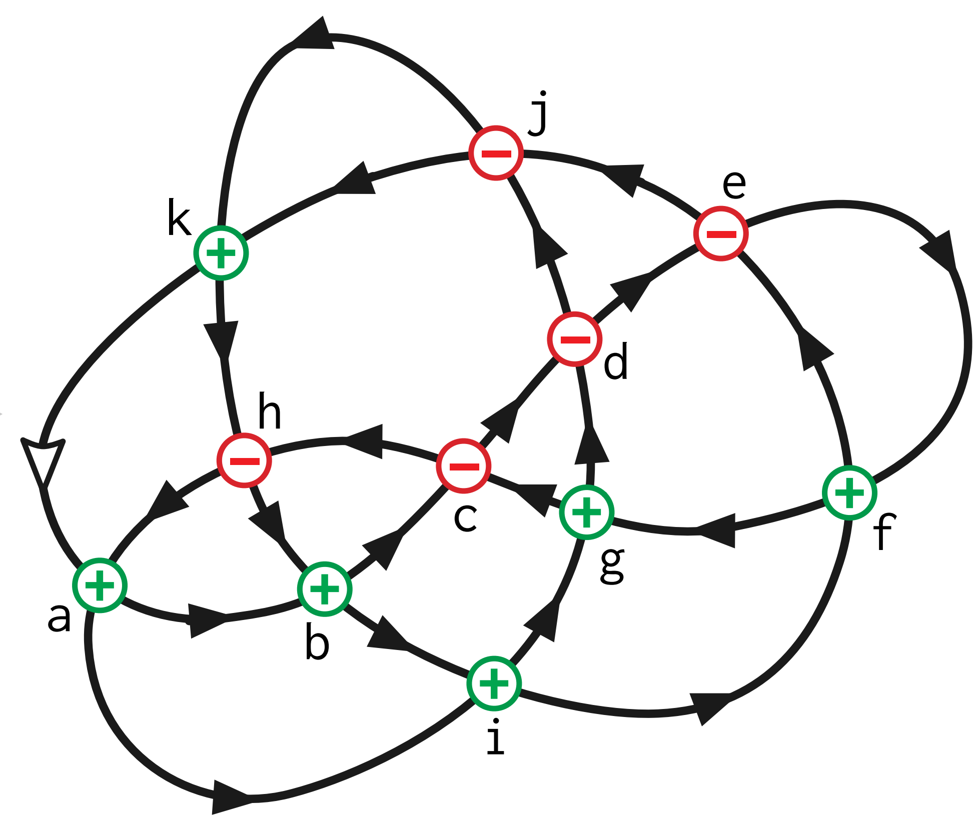
Gauss' problem

[Gauss 1844]

Which Gauss codes correspond to planar curves?

++−−−+++−−+−+−++−−−++−
abcdefgchaigdjkhbifejk

?



Gauss' problem

[Gauss 1844]

Es finden jedoch bei diesen Arrangements einige Bedingungen statt, so dass nicht jedes aus der Luft gegriffene Arrangement möglich ist; jeder Knoten muss einmal an einer geraden, einmal an einer ungeraden Stelle vorkommen; zwischen den beiden Plätzen muss die Summe aller $+\alpha$, $-\beta$ Null werden. Diess reicht aber nicht zu, um die Unmöglichkeit des Schemas

a	*b*	*c*	*b*	*a*	*b*	*c*	*b*
1	2	3	1	4	3	2	4
+	−	+	−	+	−	+	−

zu zeigen; hier müssen die Zeichen von 2 und 3 nothwendig geändert werden*).

*) 1844 Dec. 30 fand ich, dass die Anordnung der Zahlen (mittelste Reihe) zureicht, um auch die zugehörigen Schnittcharactere (+ und −Zeichen in der untersten Reihe) und die Verknüpfung der Tracte (oberste Reihe) daraus abzuleiten, dass aber jene Anordnung selbst nicht willkürlich ist, sondern gewissen Bedingungen unterliegt, deren vollständige Ermittelung Gegenstand neuer Arbeiten sein wird. Es leidet jedoch auch der obige Satz Einschränkungen, z. B.

Gauss’ problem

[Gauss 1844]

However, these arrangements satisfy certain conditions, so not every arrangement pulled out of thin air is possible. Each node must appear once in a positive crossing and once in a negative crossing, and between these two places, the numbers of positive and negative crossings must be equal. But this is not enough to show the impossibility of the following schema:

a	*b*	*c*	*b*	*a*	*b*	*c*	*b*
1	**2**	**3**	**1**	**4**	**3**	**2**	**4**
+	−	+	−	+	−	+	−

Here one must change the signs at nodes 2 and 3.*

*On December 30, 1844, I discovered that **the sequence of numbers (in the middle row) is sufficient** to deduce both the corresponding crossing directions (+ and − signs in the lower row) and the connections of the tract (upper row), but that arrangement itself is not arbitrary, but subject to **certain conditions, the complete determination of which will be the subject of new works**.

Gauss' problem

[Gauss 1844]

However, these arrangements satisfy certain conditions, so not every arrangement pulled out of thin air is possible. Each node must appear once in a positive crossing and once in a negative crossing, and between these two places, the numbers of positive and negative crossings must be equal. But this is not enough to show the impossibility of the following schema:

a	*b*	*c*	*b*	*a*	*b*	*c*	*b*
1	2	3	1	4	3	2	4
+	−	+	−	+	−	+	−

"Seifert" circles

Here one must change the signs at nodes 2 and 3.*

---

*On December 30, 1844, I discovered that **the sequence of numbers (in the middle row) is sufficient** to deduce both the corresponding crossing directions (+ and − signs in the lower row) and the connections of the tract (upper row), but that arrangement itself is not arbitrary, but subject to **certain conditions, the complete determination of which will be the subject of new works**.

Gauss' problem

[Gauss c.1850]

Which Gauss codes correspond to planar curves?

Ein vollständiger Knoten.

1. *aa*

Zwei vollständige Knoten.

1. *aabb*
2. *abab* *
3. *abba*

Drei vollständige Knoten.

1. *aabbcc*	9. *abbcca*
2. *aabcbc* *	10. *abcabc*
3. *aabccb*	11. *abcacb* *
4. *ababcc* *	12. *abcbac* *
5. *abacbc* **	13. *abcbca* **
6. *abaccb* *	14. *abccab* *
7. *abbacc*	15. *abccba*
8. *abbcac* *	

Gauss’ problem

[Gauss c.1850]

Which Gauss codes correspond to planar curves?

19. abacbcdd **
20. abacbdcd **
21. abacbddc **
22. abaccbdd **
23. abaccdbd **
24. abaccddb **
25. abacdbcd **
26. abacdbdc **
27. abacdcbd **
28. abacdcdb **
29. abacddbc **
30. abacddcb **
31. abbaccdd
32. abbacdcd *
33. abbacddc
34. abbcacdd *
35. abbcadcd **

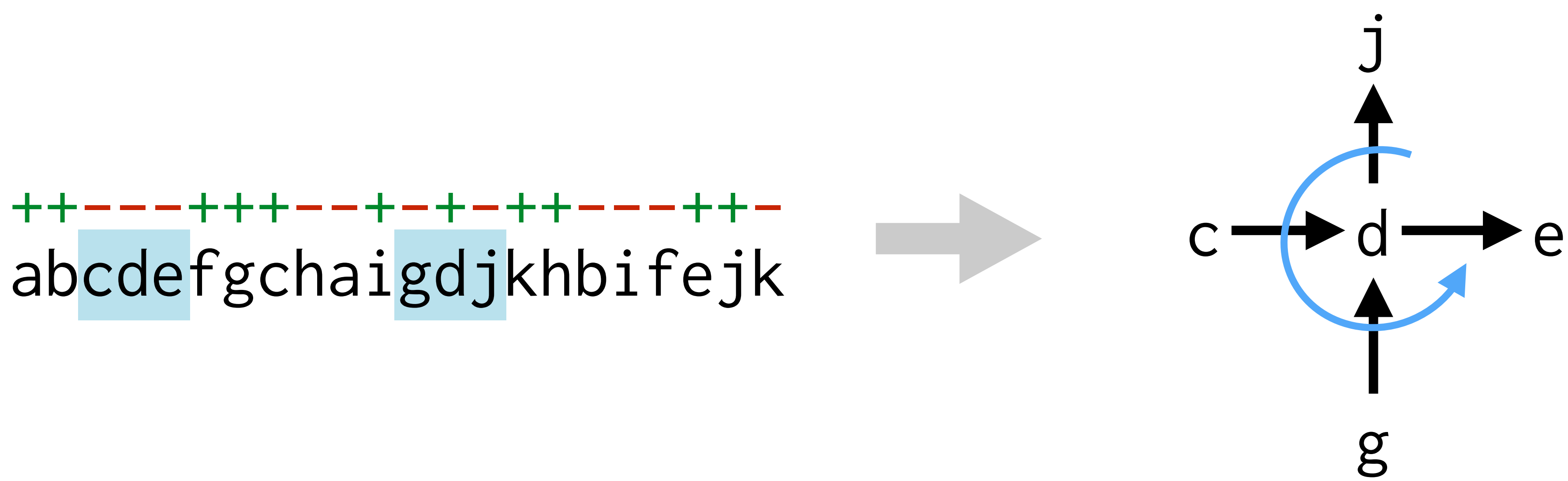
54. abcadcbd
55. abcadcdb **
56. abcaddbc
57. abcaddcb *
58. abcbacdd *
59. abcbadcd **
60. abcbaddc *
61. abcbcadd **
62. abcbcdad **
63. abcbcdda **
64. abcbdacd **
65. abcbdadc **
66. abcbdcad **
67. abcbdcda **
68. abcbddac *
69. abcbddca **
70. abccabdd *

89. abcdbadc
90. abcdbcad **
91. abcdbcda
92. abcdbdac **
93. abcdbdca **
94. abcdcabd **
95. abcdcadb **
96. abcdcbad *
97. abcdcbda **
98. abcdcdab **
99. abcdcdba **
100. abcddabc
101. abcddacb *
102. abcddbac *
103. abcddbca **
104. abcddcab *
105. abcddcba

Signed codes are easy

[Francis 1969]
[Carter 1991]
[Cairns Elton 1993]

- Every signed Gauss code is consistent with a *unique* rotation system.

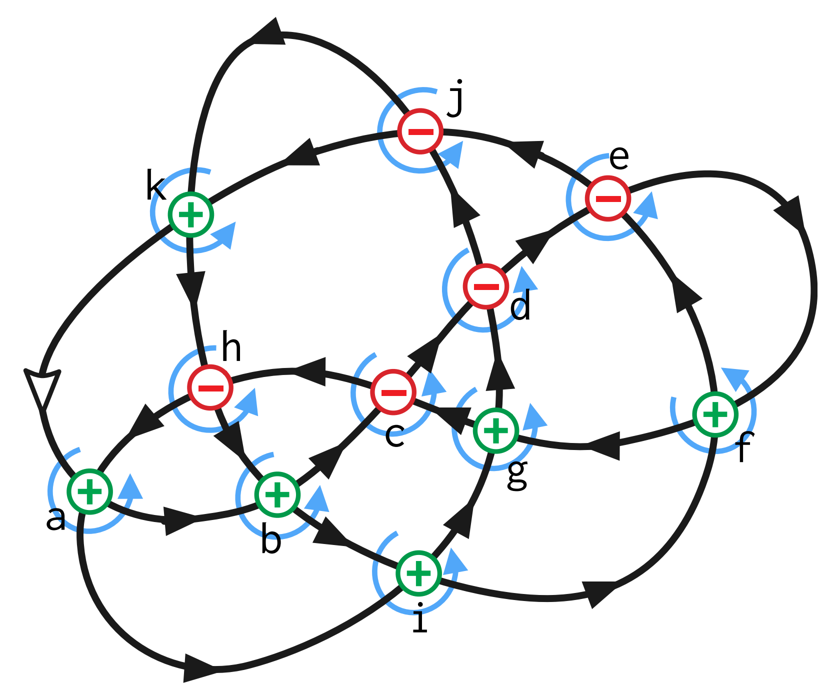


Signed codes are easy

[Francis 1969]
[Carter 1991]
[Cairns Elton 1993]

- Every signed Gauss code is consistent with a *unique* rotation system.
- A rotation system describes a planar embedding if and only if it satisfies *Euler's formula V−E+F=2*.

```
++−−−+++−−+−+−++−−−++−
abcdefgchaigdjkhbifejk
```

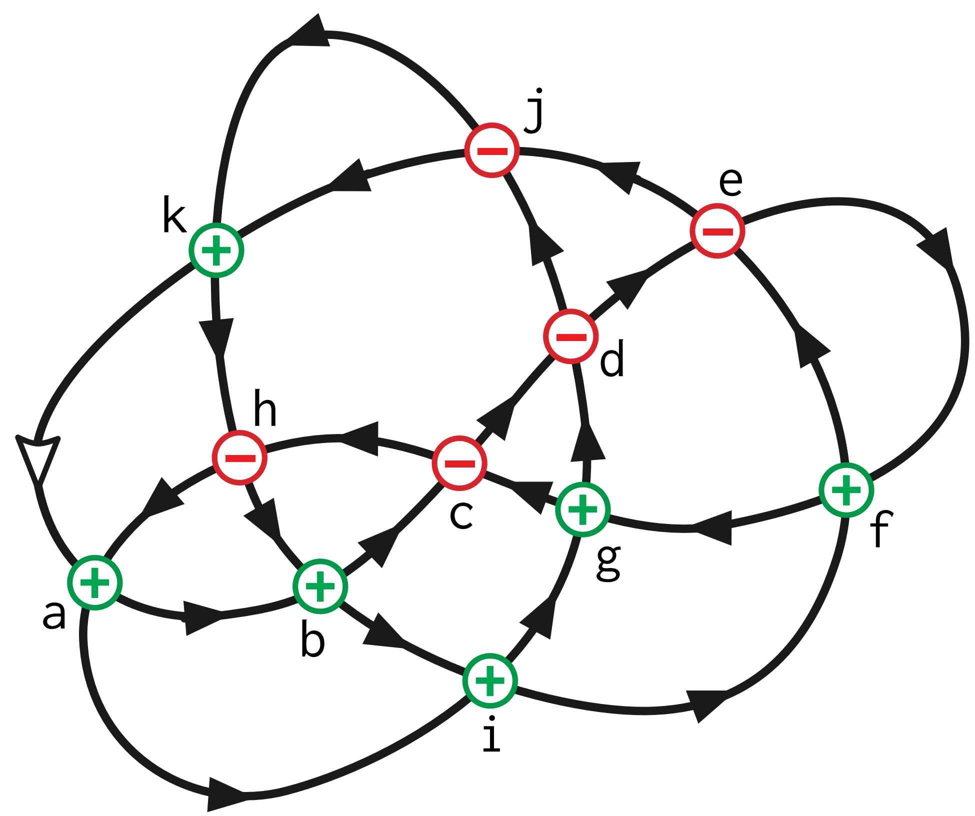


Parity condition

[Gauss c. 1850]
[Tait 1877]

- Any matching pair of symbols must be separated by an even number of other symbols

abcdefgchaigdjkhbifejk

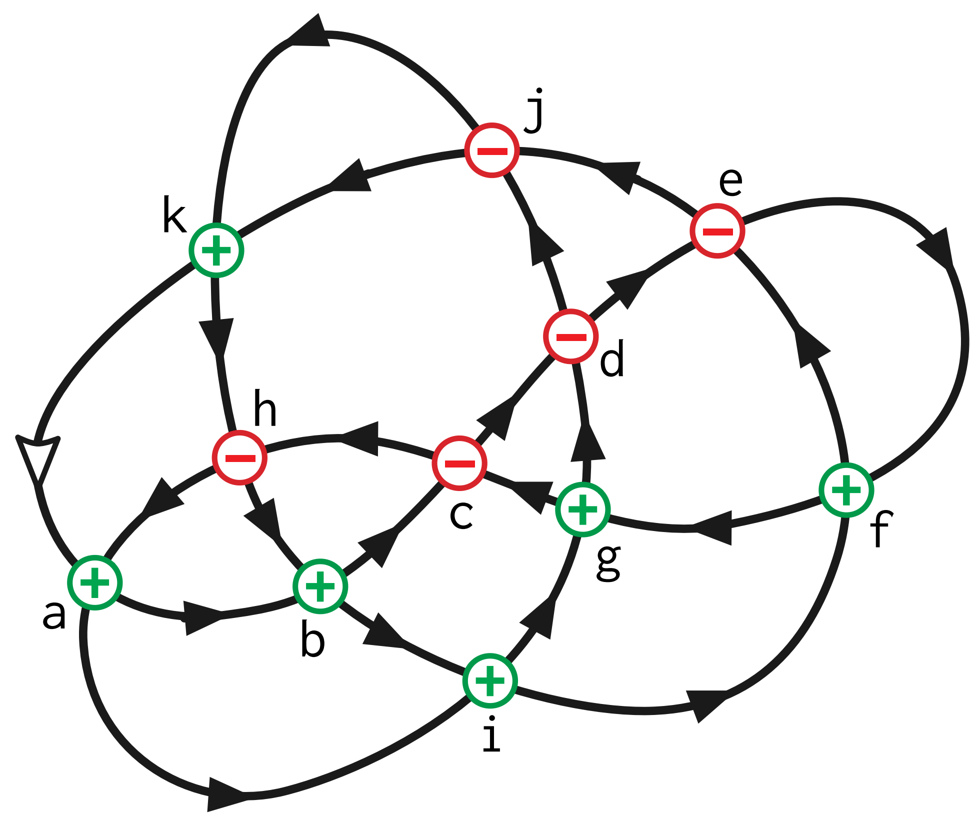


Parity condition

[Gauss c. 1850]
[Tait 1877]

- Any matching pair of symbols must be separated by an even number of other symbols

abcdefgchaigdjkhbifejk

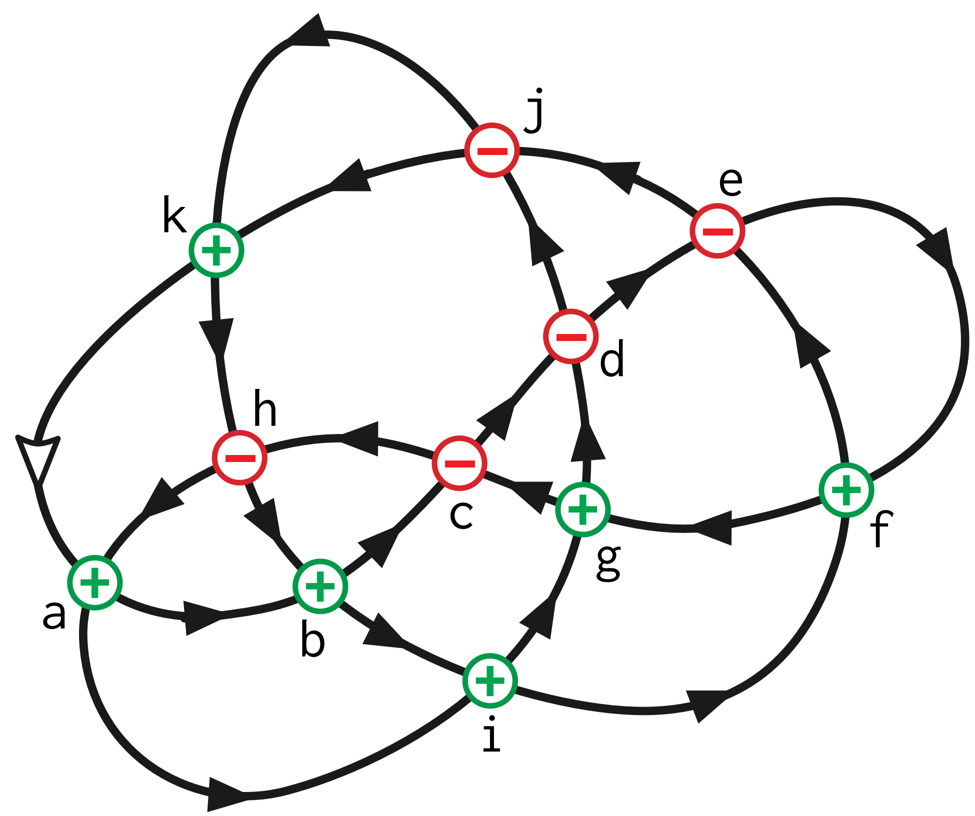


Parity condition

[Gauss c. 1850]
[Tait 1877]

- Any matching pair of symbols must be separated by an even number of other symbols

abcdefgchaigdjkhbifejk

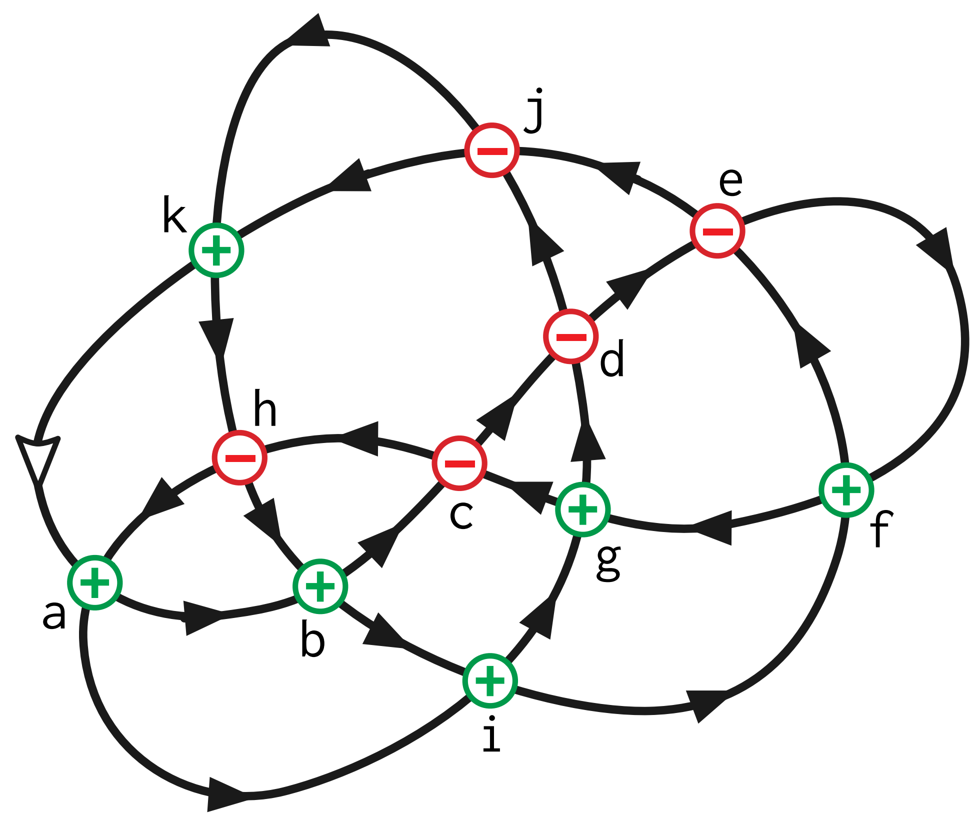


Parity condition

[Gauss c. 1850]
[Tait 1877]

- Any matching pair of symbols must be separated by an even number of other symbols

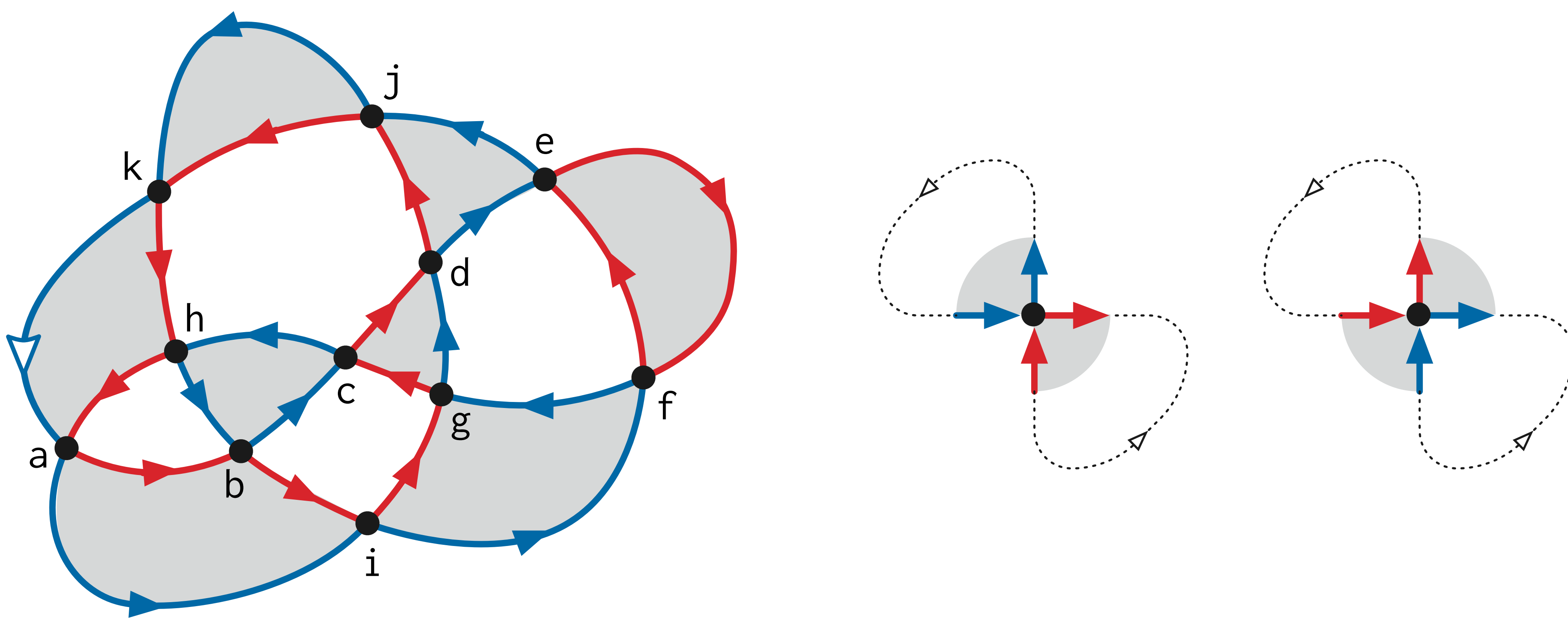
abcdefgchaigdjkhbifejk



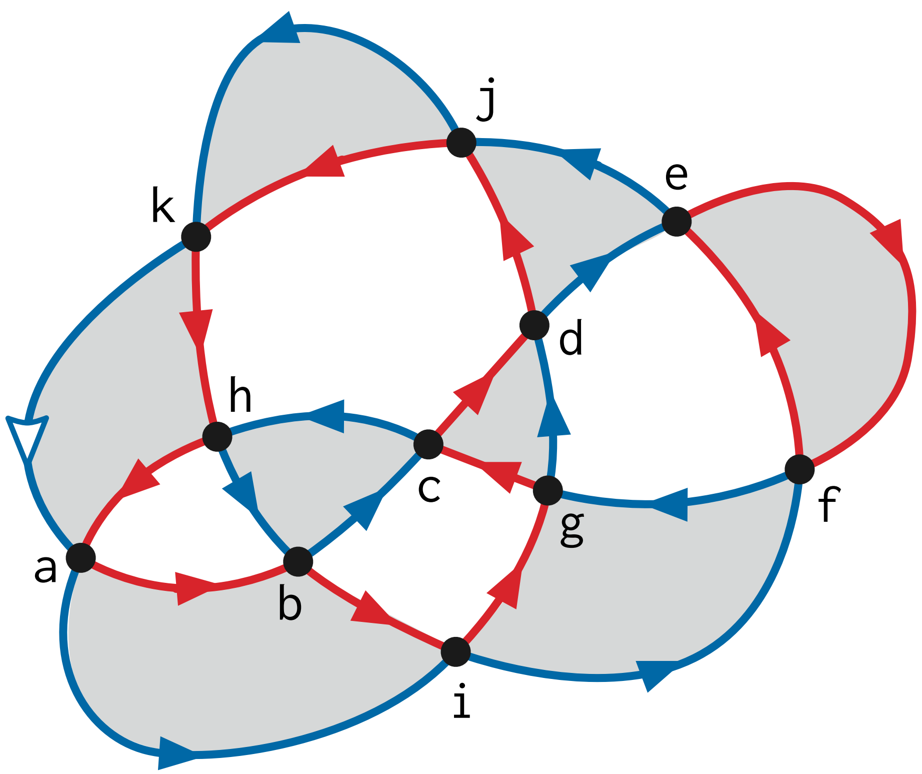
Parity proof

[Nagy 1927]

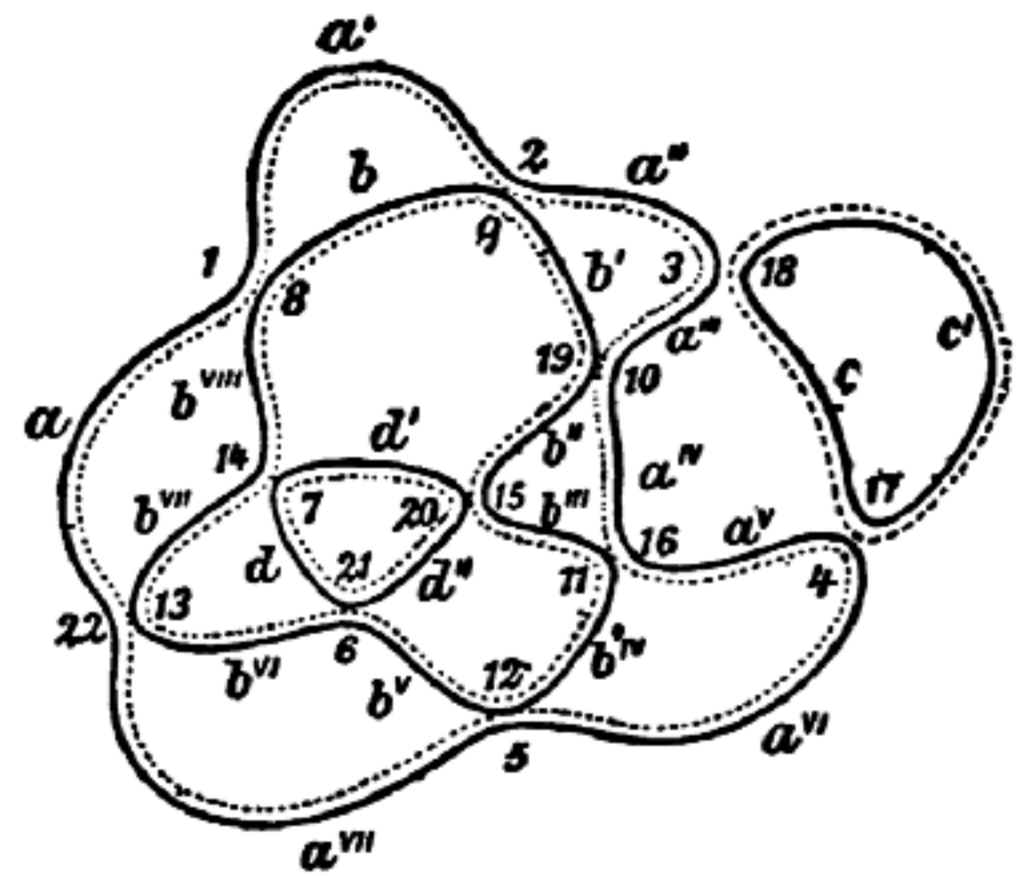
- Color segments of the curve alternately *red* and *blue*.
  - *Red* = odd winding number on the right
  - *bLue* = odd winding number on the Left
- After leaving a vertex along a *red* segment, you must next enter that vertex along a *red* segment.



Look familiar?



[Nagy 1927]



[Gauss c.1845]

Parity condition

- Unfortunately this condition is *not* sufficient.

Dieses Criterium hört aber bei Perioden von mehr als 4 Knoten auf, für die Möglichkeit zureichend zu sein; z. B. ***abcadcedbe*** oder $\begin{pmatrix} 0. & 2. & 4. & 6. & 8 \\ 3. & 5. & 7. & 9. & 1 \end{pmatrix}$ ist, obgleich dem Criterium genügt ist, unmöglich.

Ebenso ist unmöglich ***abcabdecde*** [oder] $\begin{pmatrix} 0. & 2. & 4. & 6. & 8 \\ 3. & 7. & 1. & 9. & 5 \end{pmatrix}$.

[Gauss c.1850]

the number which need here be tried. But, *secondly*, even when this is attended to, the scheme may be an impossible one. Thus, the scheme

$$\mathrm{A}\ D\ \mathrm{B}\ E\ \mathrm{C}\ A\ \mathrm{D}\ B\ \mathrm{E}\ C \mid \mathrm{A}$$

is lawful, but

$$\mathrm{A}\ D\ \mathrm{B}\ A\ \mathrm{C}\ E\ \mathrm{D}\ C\ \mathrm{E}\ B \mid \mathrm{A}$$

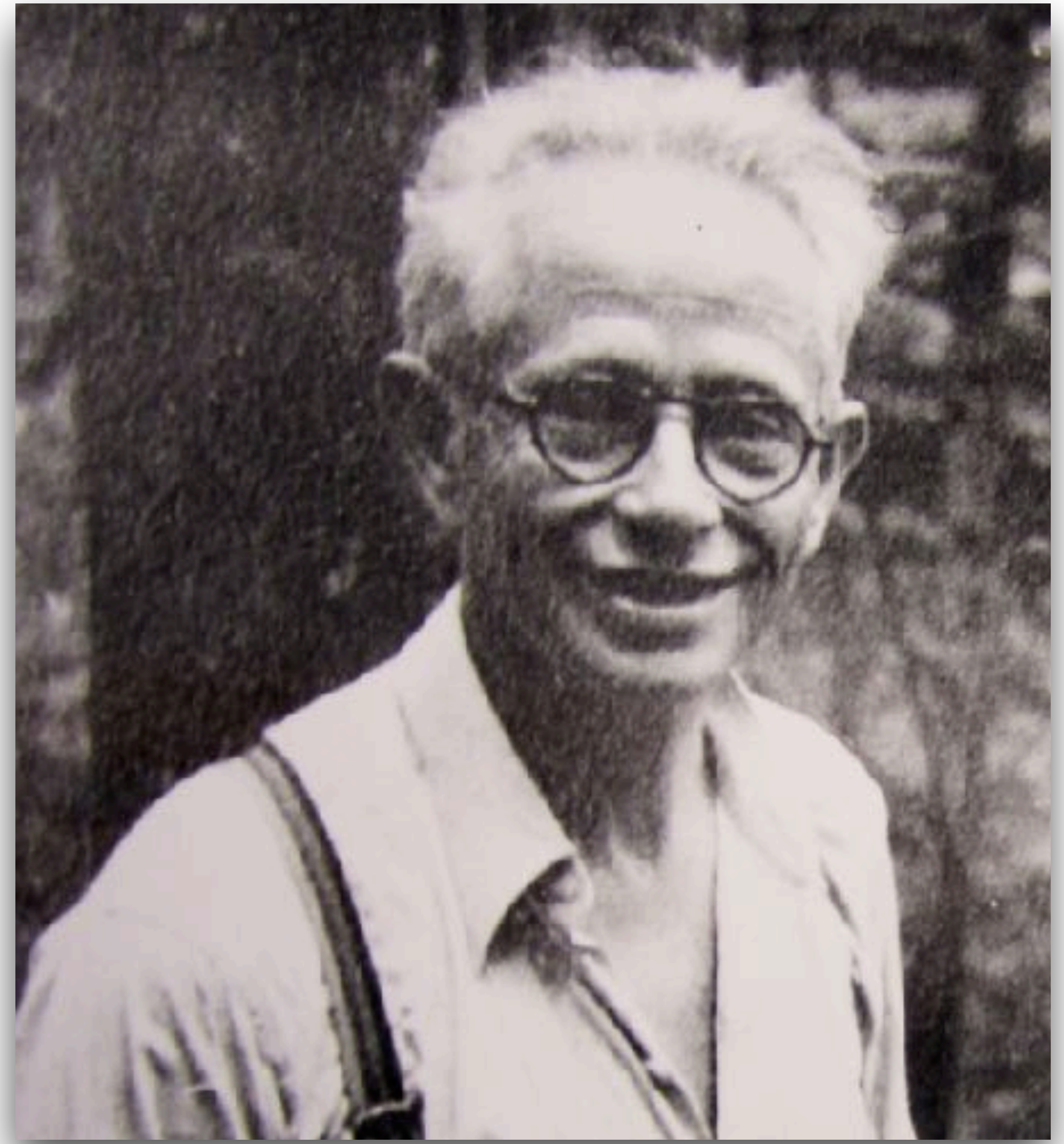
is not.

[Tait 1877]

abcadcedbe abcabdecde abcadebdec

1.	aabbccddee	31.	abbccaddee	61.	abcaddecbe	91.	abcdbceade*
2.	ccdeed	32.	adeed	62.	deebc	92.	ceeda
3.	cddcee	33.	ddaee	63.	ebced	93.	edaec*
4.	cddeec	34.	ddeea	64.	ebdec*	94.	edcea
5.	cdecde	35.	deade	65.	eecbd	95.	eeadc
6.	cdeedc	36.	deeda	66.	eedbc	96.	eecda
7.	aabccbddee	37.	abbcdacdee	67.	abccbaddee	97.	abcddabcee
8.	bdeed	38.	aceed	68.	adeed	98.	abeec
9.	ddbee	39.	aedce	69.	ddaee	99.	aecbe
10.	ddeeb	40.	aeecd	70.	ddeea	100.	aeebc
11.	debde	41.	dcaee	71.	deade	101.	cbaee
12.	deedb	42.	dceea	72.	deeda	102.	cbeea
13.	aabcdbcdee	43.	deace	73.	abccdabdee	103.	ceabe
14.	bceed	44.	deeca	74.	abeed	104.	ceeba
15.	bedce	45.	ecaed	75.	aedbe	105.	ebaec
16.	beecd	46.	ecdea	76.	aeebd	106.	ebcea
17.	dcbee	47.	eeacd	77.	dbaee	107.	eeabc
18.	dceeb	48.	eedca	78.	dbeea	108.	eecba
19.	debce	49.	abcabcddee	79.	deabe	109.	abcdeabcde
20.	deecb	50.	cdeed	80.	deeba	110.	abedc
21.	ecbed	51.	ddcee	81.	ebaed	111.	adcbe
22.	ecdeb	52.	ddeec	82.	ebdea	112.	adebc*
23.	eebcd	53.	decde*	83.	eeabd	113.	cbade
24.	eedcb	54.	deedc	84.	eedba	114.	cbeda
25.	abbaccddee	55.	abcadcbdee	85.	abcdbadcee	115.	cdabe*
26.	cdeed	56.	cbeed	86.	adeec	116.	cdeba
27.	ddcee	57.	cedbe*	87.	aecde	117.	ebadc
28.	ddeec	58.	ceebd	88.	aeedc	118.	ebcda
29.	decde	59.	dbcee	89.	cdaee	119.	edabc
30.	deedc	60.	dbeec	90.	cdeea	120.	edcba

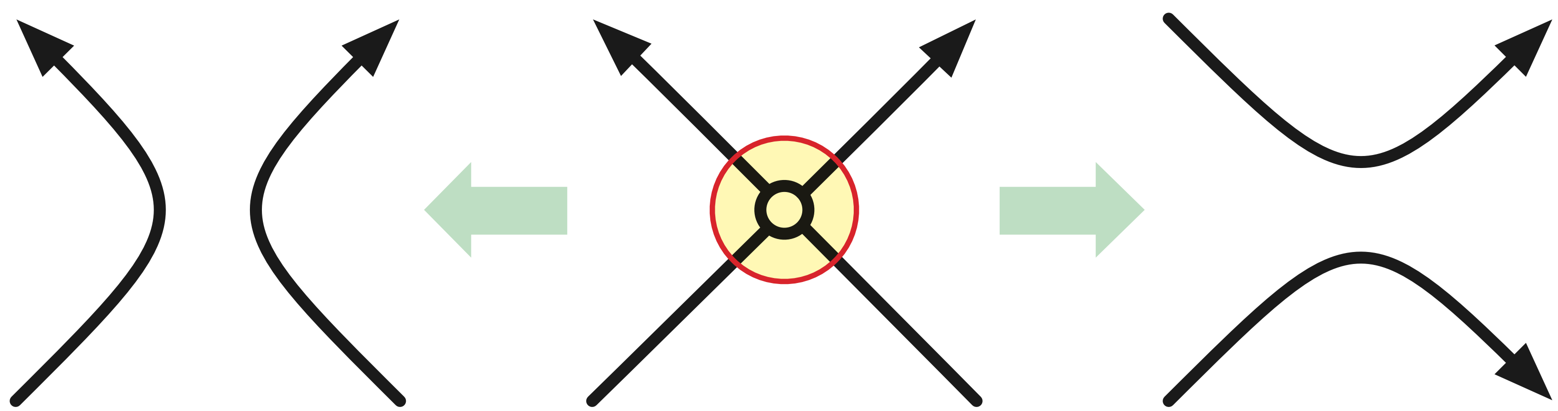
The solution

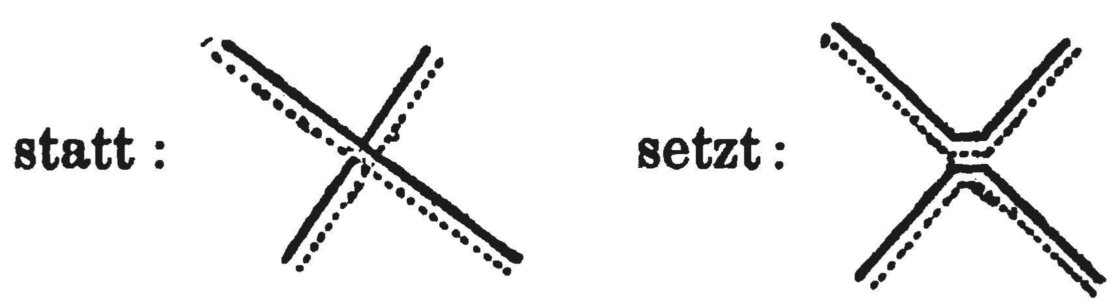


[Trude Guermonprez,
Western Regional Archives,
State Archives of North Carolina]

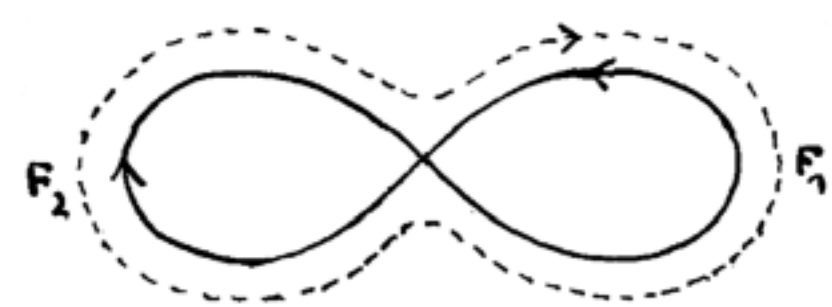
Two ways to smooth a crossing

- Maintain orientation (but disconnect the curve), or
- Maintain connection (but reverse part of the curve)





[Gauss c.1840]



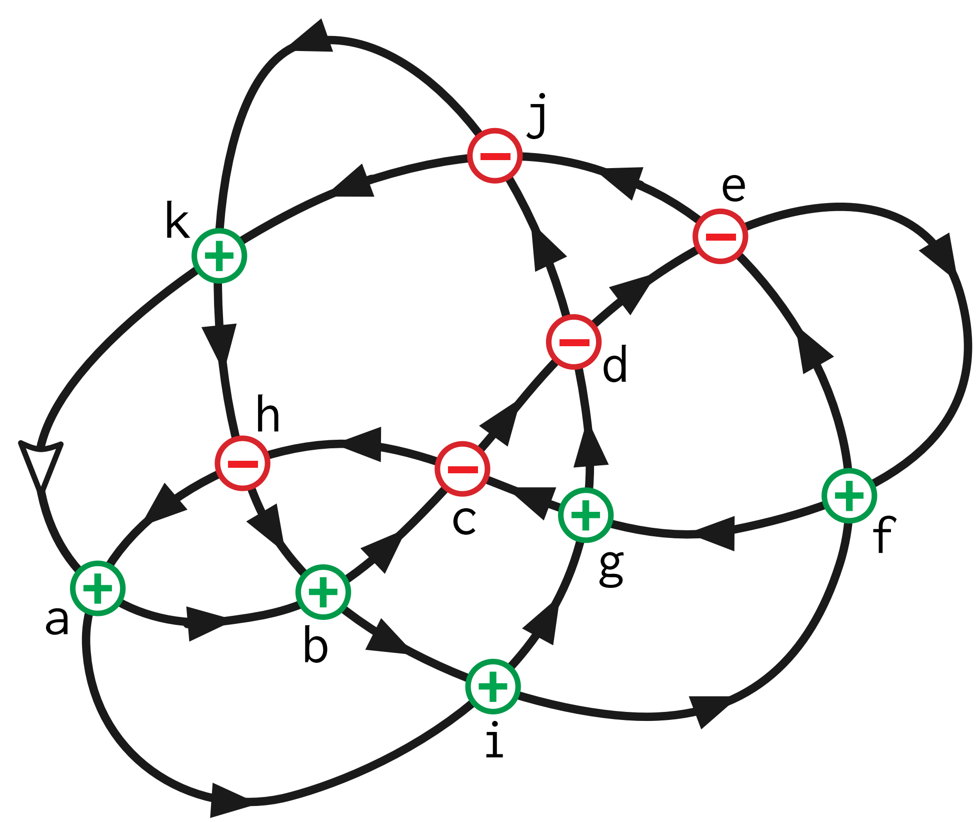
[Dehn 1936]

Untangling

[Dehn 1936]

Reverse every substring bounded by matching symbols

`abcdefgchaigdjkhbifejk`

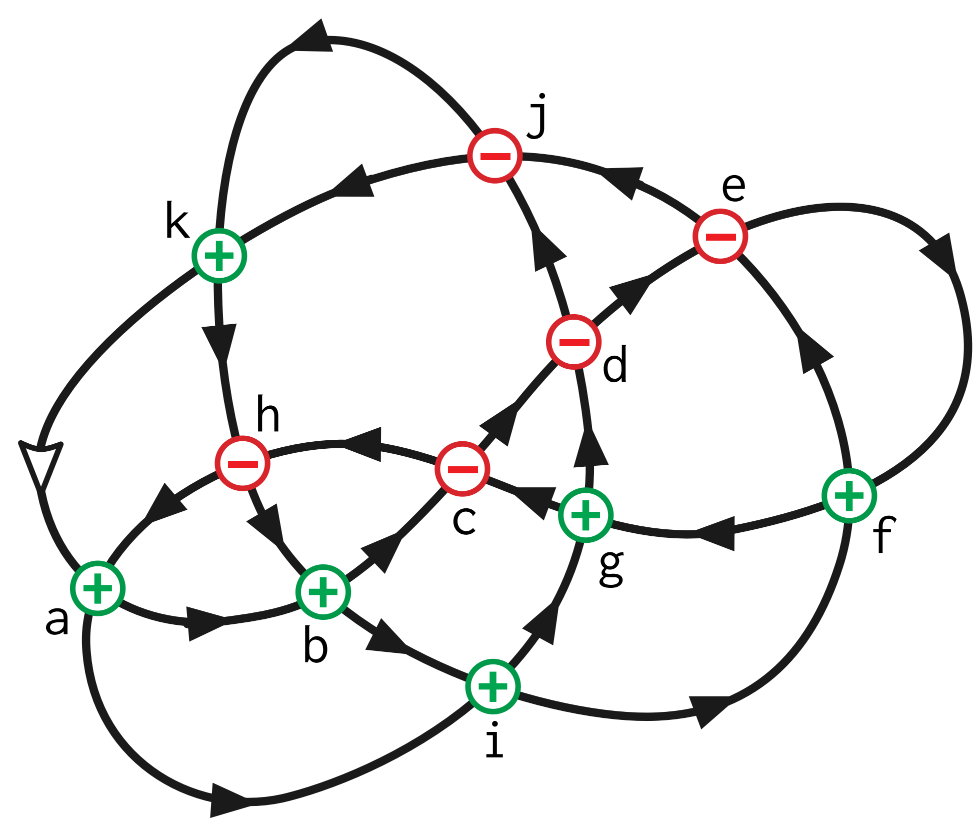


Untangling

[Dehn 1936]

Reverse every substring bounded by matching symbols

```
abcdefgchaigdjkhbifejk
ahcgfedcbaigdjkhbifejk
```



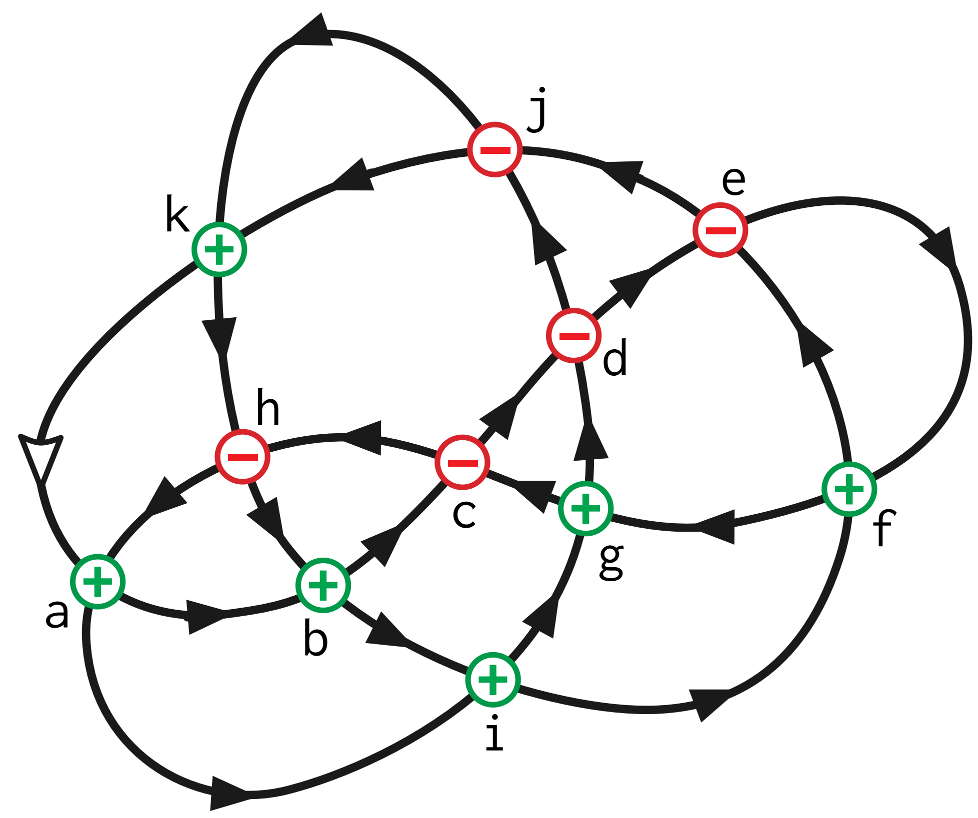
Untangling

[Dehn 1936]

Reverse every substring bounded by matching symbols

```
abcdefgchaigdjkhbifejk
```

ahcgfedcbaigdjkhbifejk

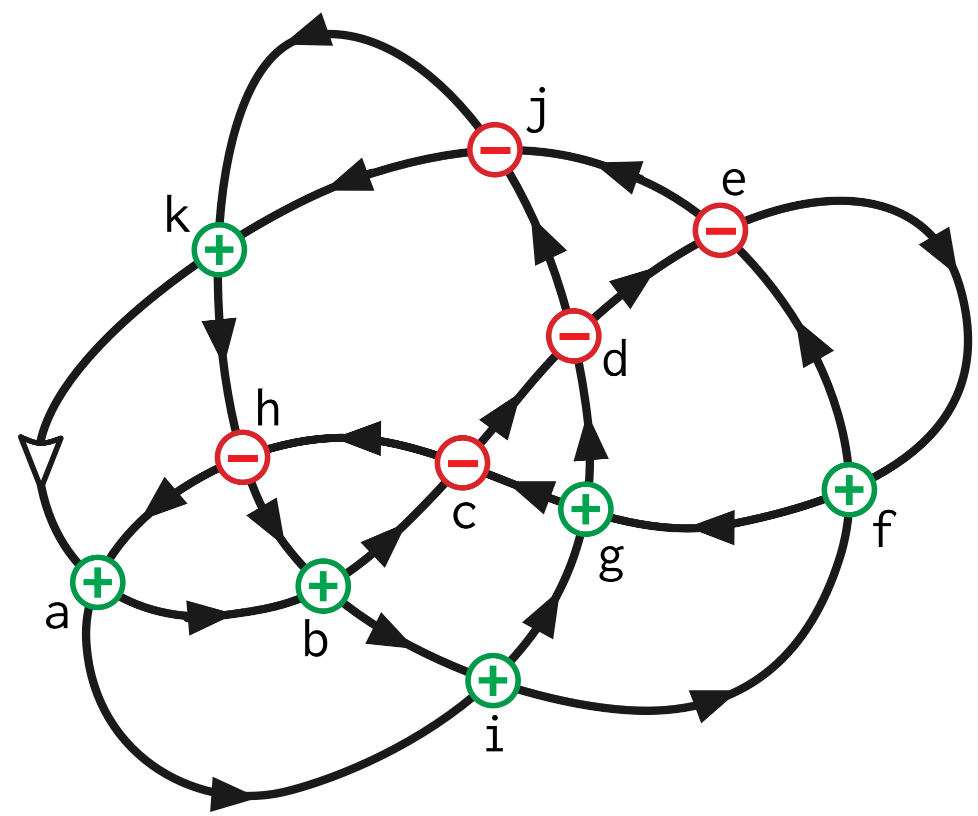


Untangling

[Dehn 1936]

Reverse every substring bounded by matching symbols

```
abcdefgchaigdjkhbifejk
ahcgfedcbaigdjkhbifejk
ahcgfedcbhkjdgiabifejk
```

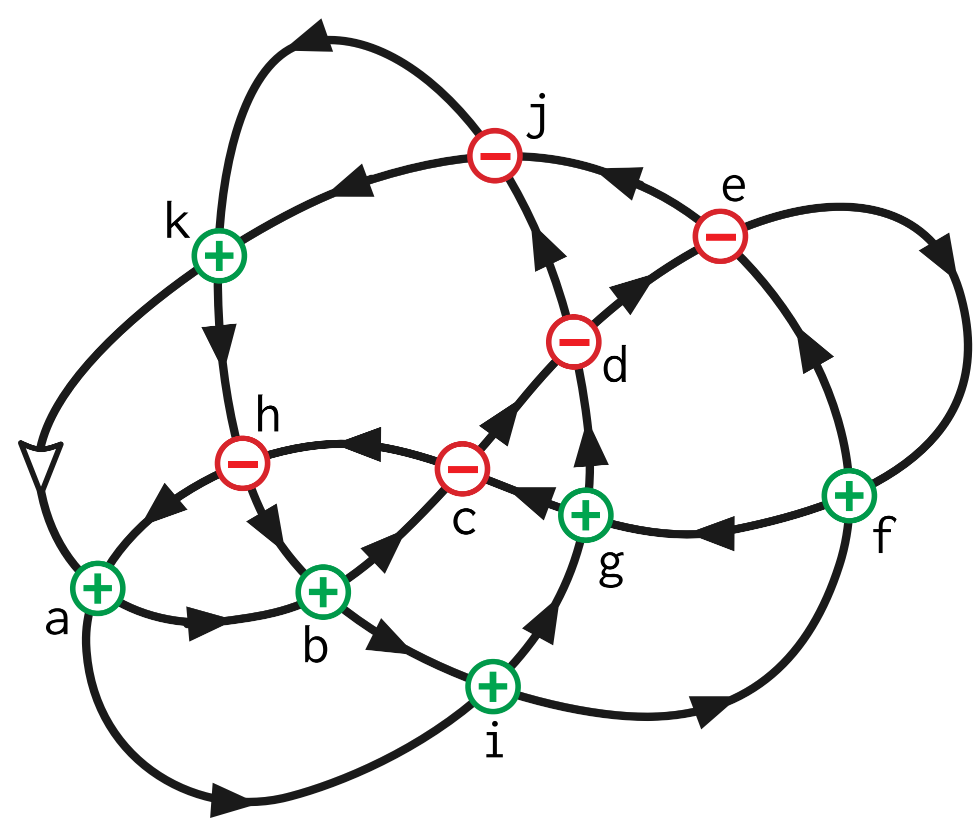


Untangling

[Dehn 1936]

Reverse every substring bounded by matching symbols

abcdefgchaigdjkhbifejk
ahcgfedcbaigdjkhbifejk
ahcgfedc**bhkjdgiab**ifejk
ah**cdefgc**bhkjdgiabifejk
ahc**djkhbcgfed**giabifejk
ahcdjkhbcgf**efibaigde**jk
ahcdjkhbcg**fef**ibaigdejk
ahcdjkhbc**giabifefg**dejk
a**hkjdch**bcgiabifefgdejk
ahkjdchbcg**ibai**fefgdejk
ahk**jedgfefiabigcbhcdj**k
ah**kjdchbcgibaifefgdejk**

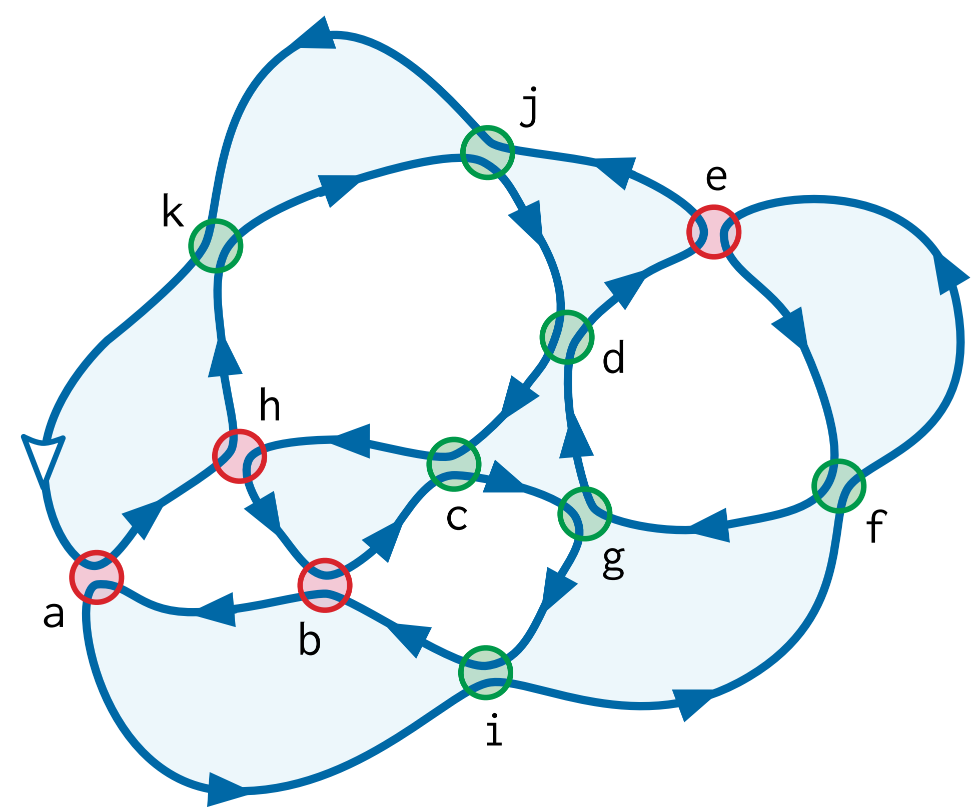


Untangling

[Dehn 1936]

Reverse every substring bounded by matching symbols

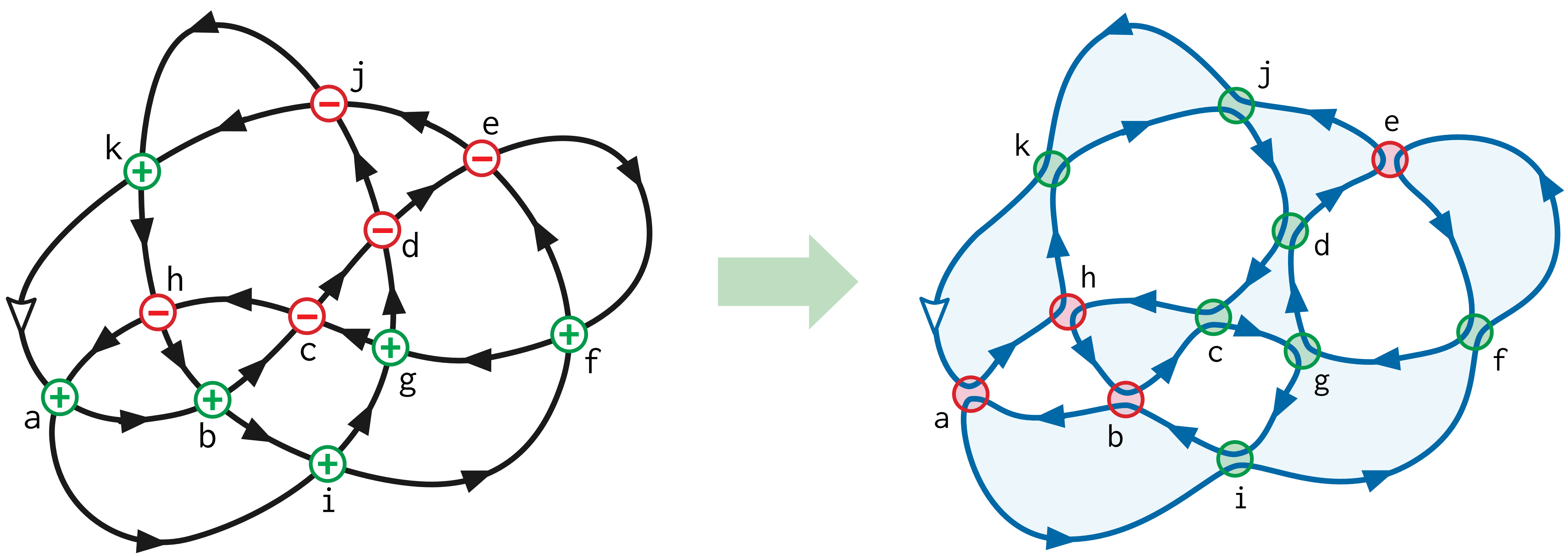
abcdefgchaigdjkhbifejk
ahcgfedcbaigdjkhbifejk
ahcgfedc**bhkjdgiab**ifejk
ah**cdefgc**bhkjdgiabifejk
ahc**djkhbcgfed**giabifejk
ahcdjkhbcgf**efibaigde**jk
ahcdjkhbcg**fef**ibaigdejk
ahcdjkhbc**giabifefg**dejk
a**hkjdch**bcgiabifefgdejk
ahkjdchbcg**ibai**fefgdejk
ahk**jedgfefiabigcbhcdj**k
ah**kjdchbcgibaifefgdejk**



Untangling condition

[Dehn 1936]

The untangled code must be consistent with a *weakly simple* closed curve



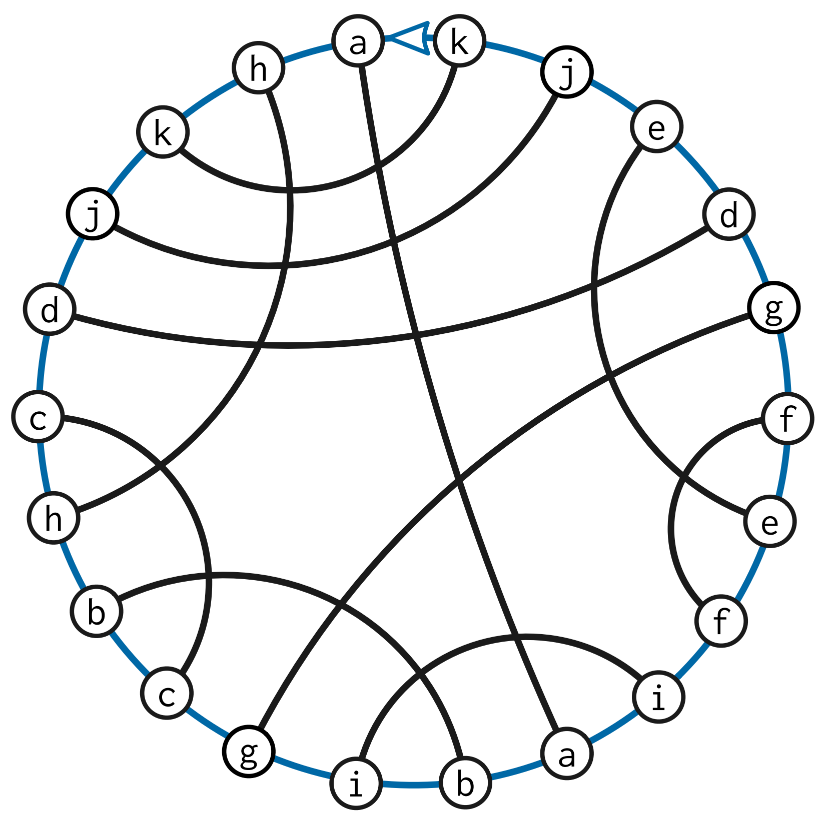
abcdefgchaigdjkhbifejk

ahkjdchbcgibaifefgdejk

Untangling condition

[Dehn 1936]

The *Gauss diagram* of the untangled code must be planar

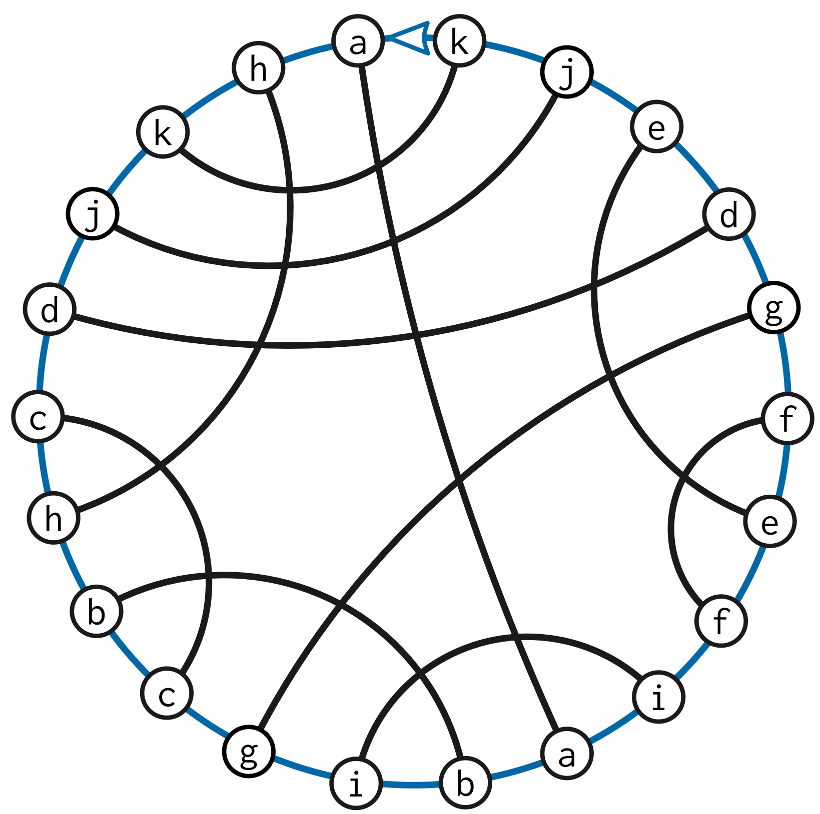


ahkjdchbcgibaifefgdejk

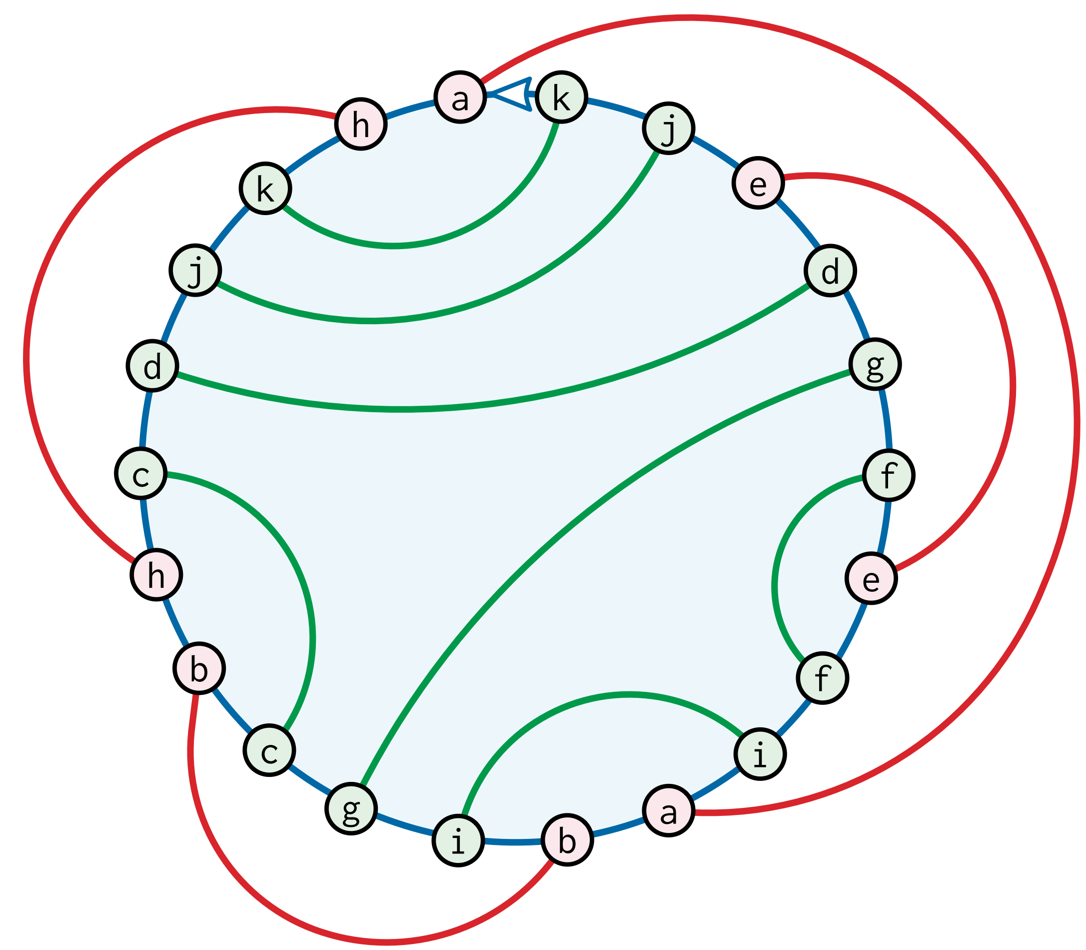
Untangling condition

[Dehn 1936]

The *Gauss diagram* of the untangled code must be planar



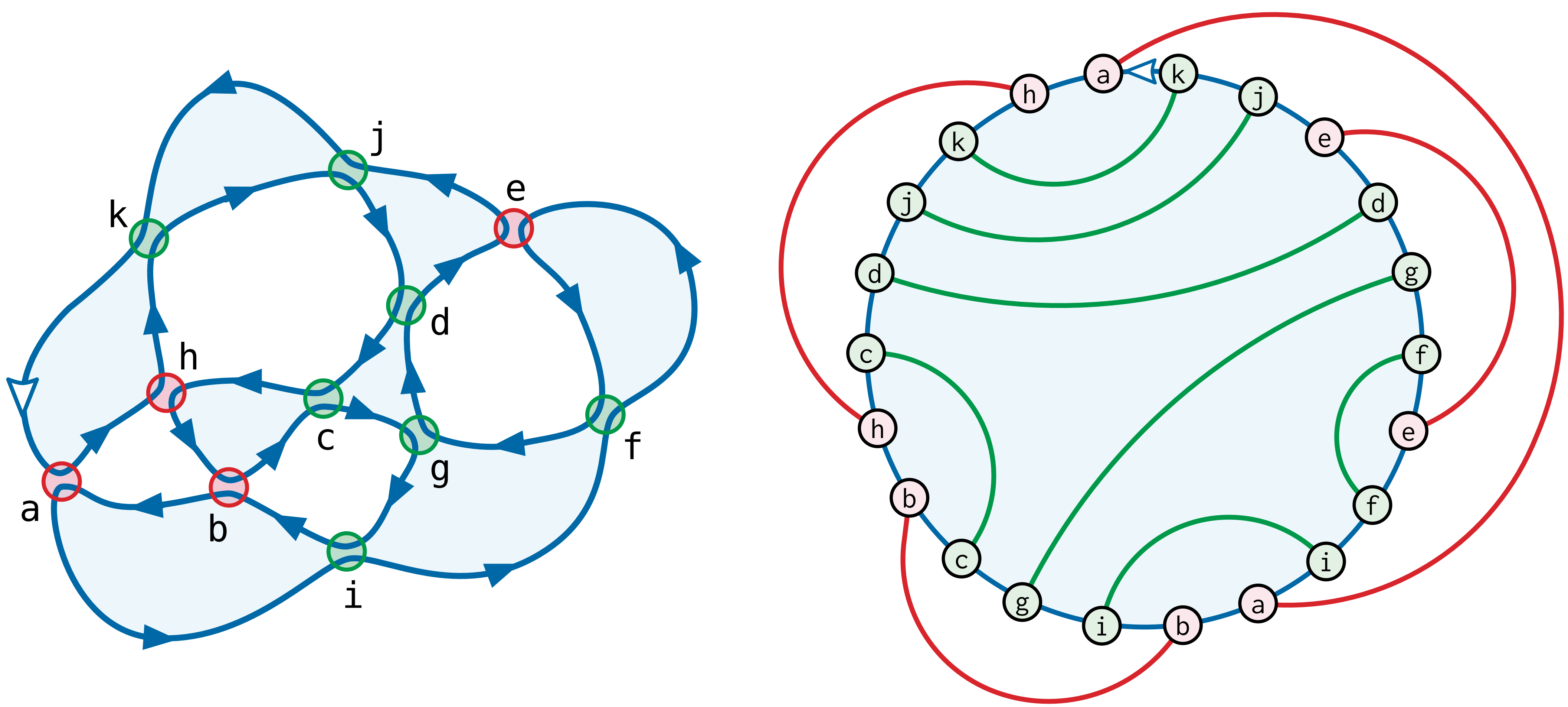
ahkjdchbcgibaifefgdejk



Untangling condition

[Dehn 1936]

The *Gauss diagram* of the untangled code must be planar



Untangling condition

[Dehn 1936]

The *Gauss diagram* of the untangled code must be planar

- “Baum-Zwiebel Figur”

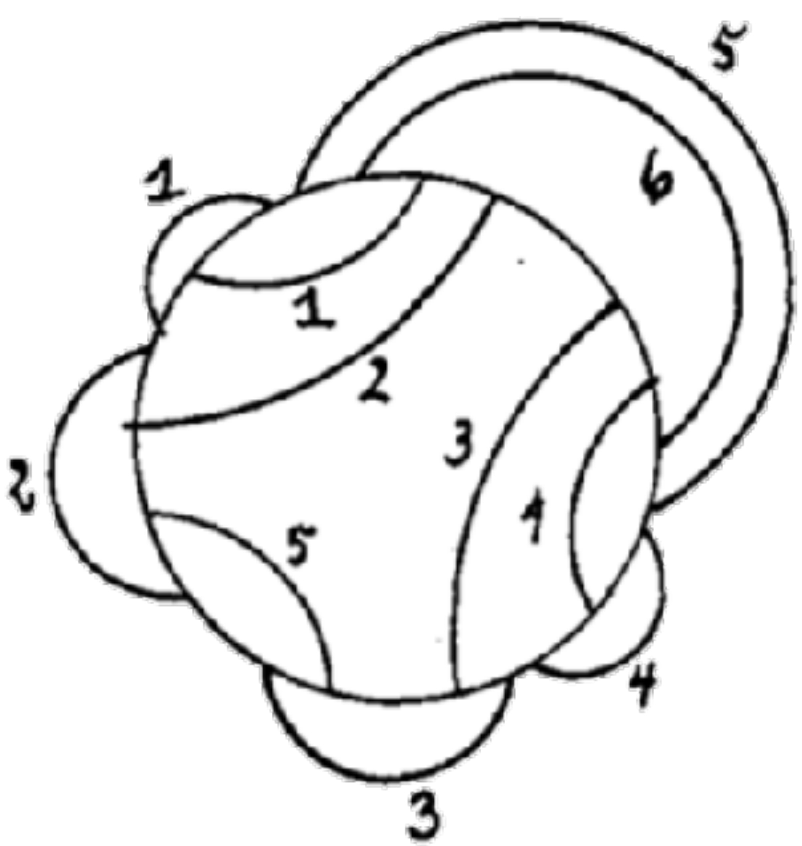


Fig. 14.

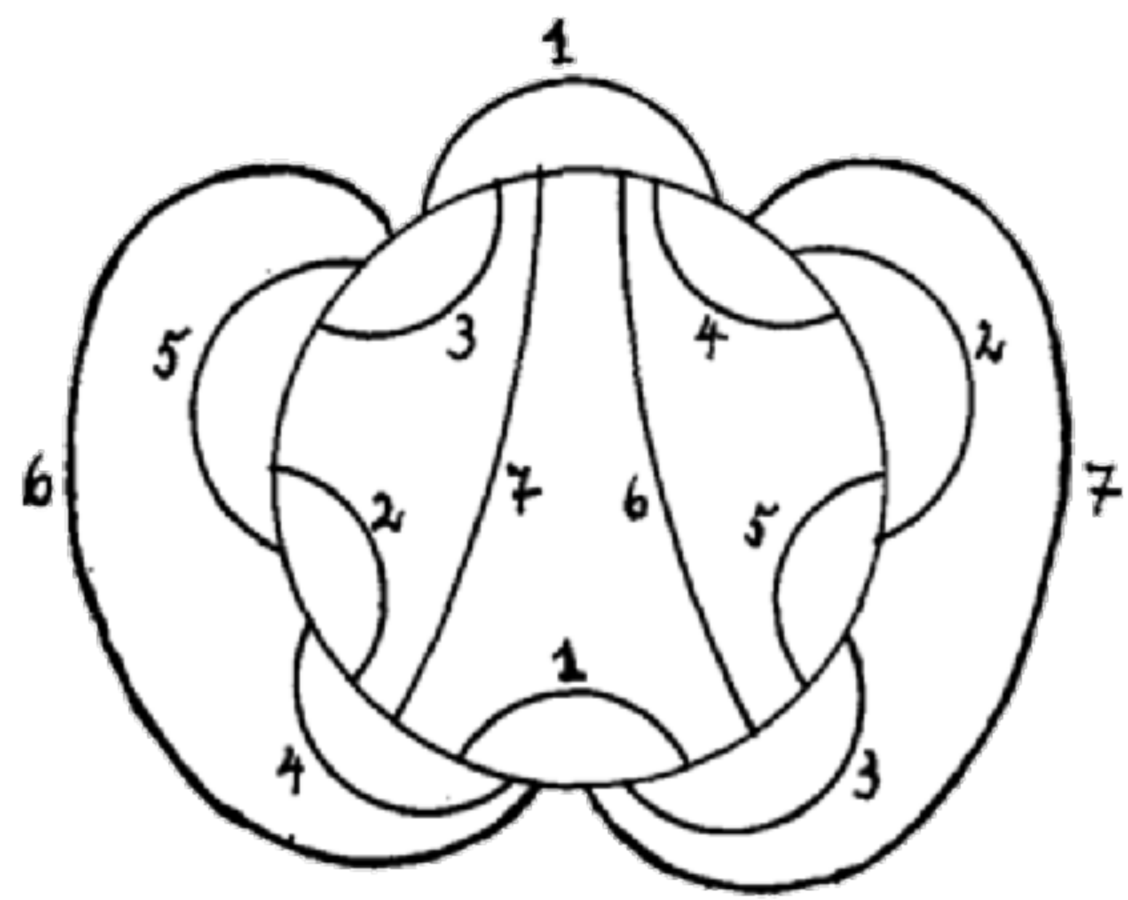
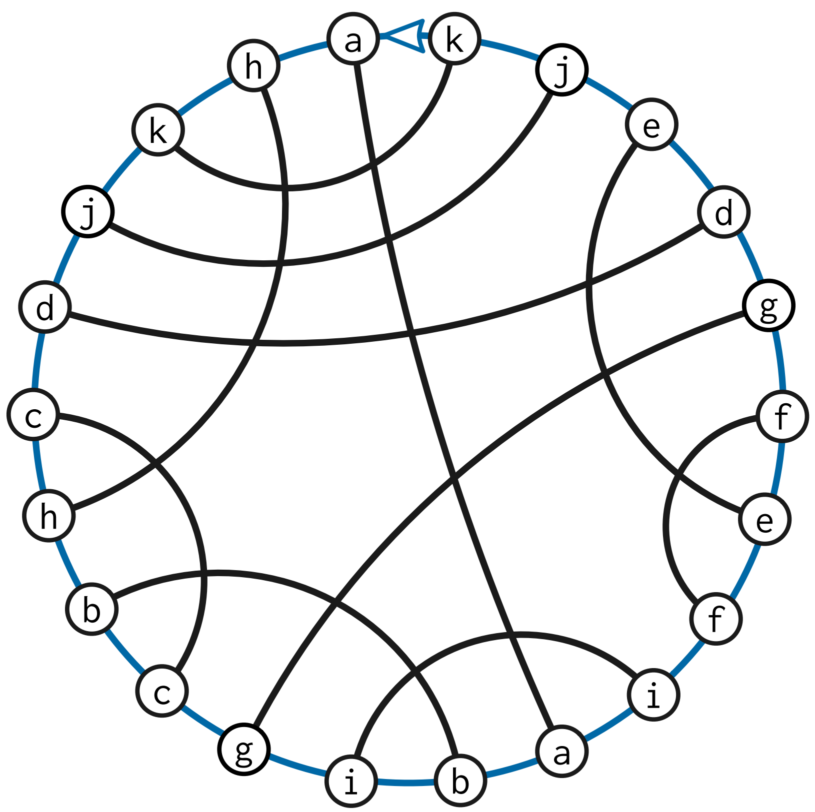


Fig. 15.

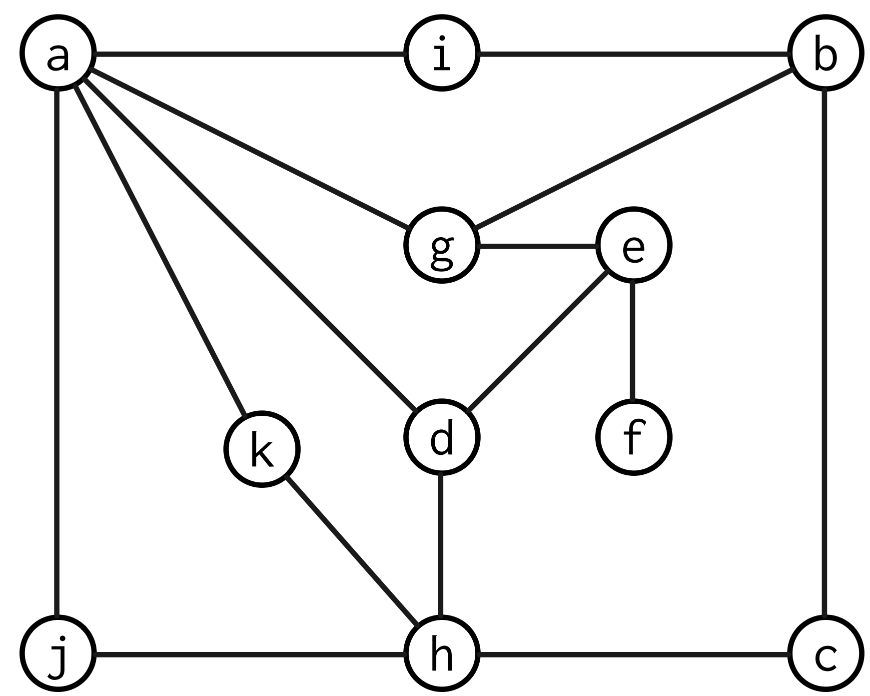
Untangling condition

[Dehn 1936]
[Rosenstiehl 1976]

The *interlacement graph* of the untangled code
must be *bipartite*



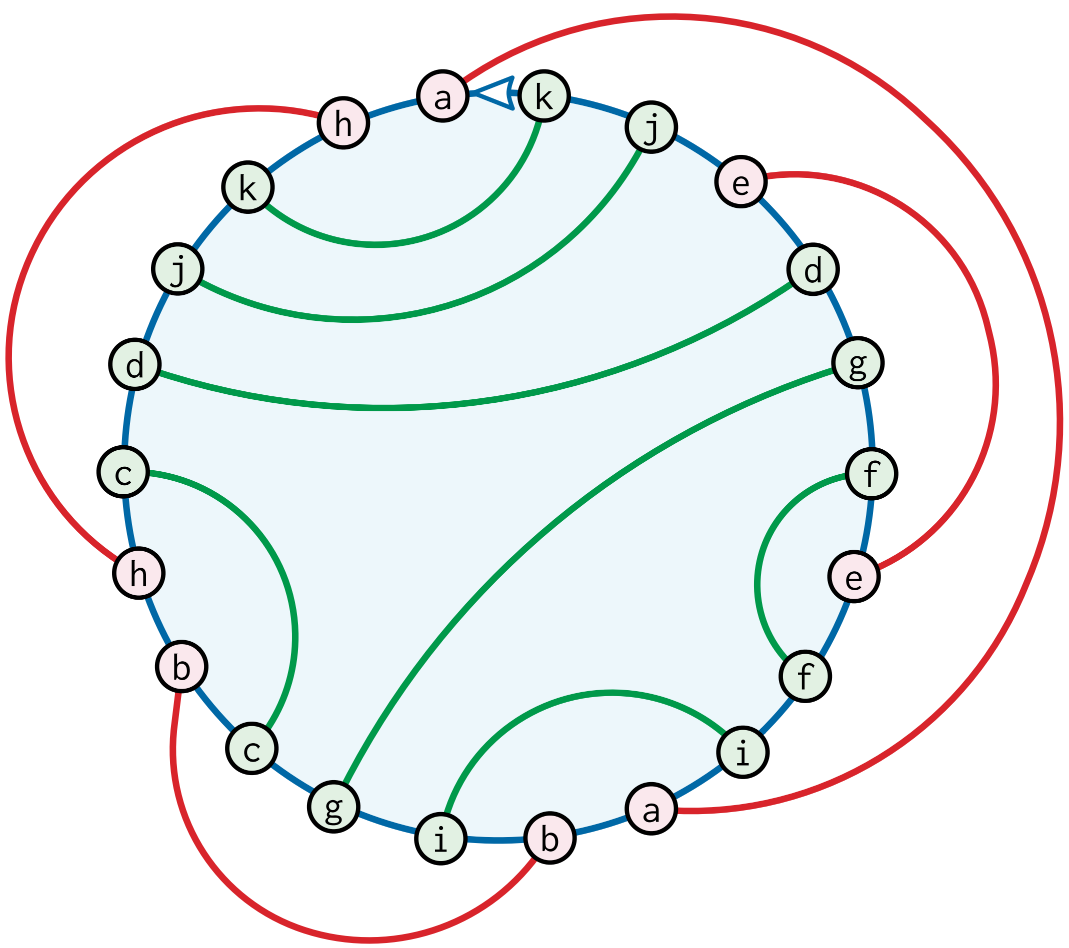
ahkjdchbcgibaifefgdejk



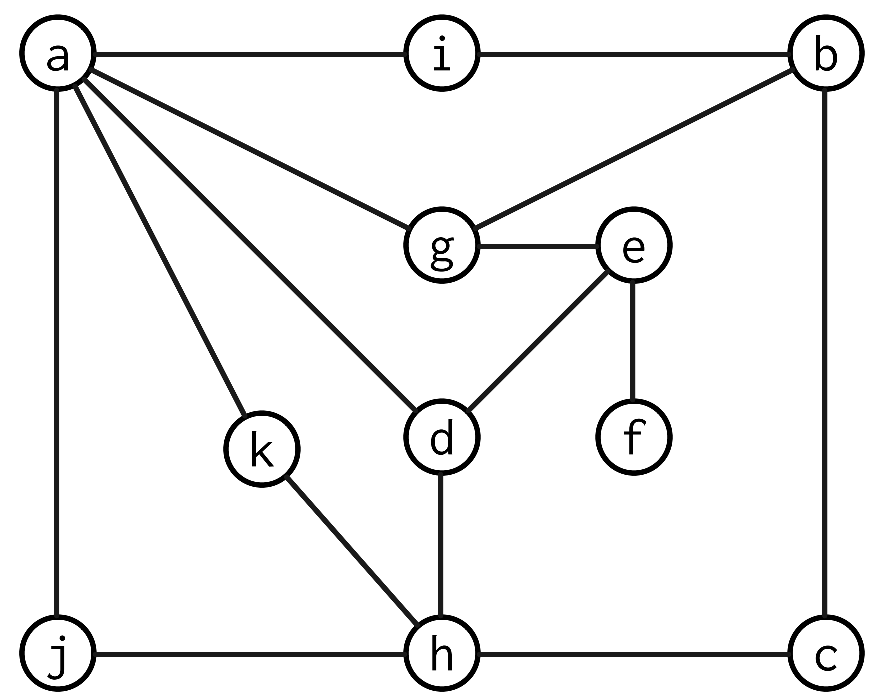
Untangling condition

[Dehn 1936]
[Rosenstiehl 1976]

The *interlacement graph* of the untangled code must be *bipartite*



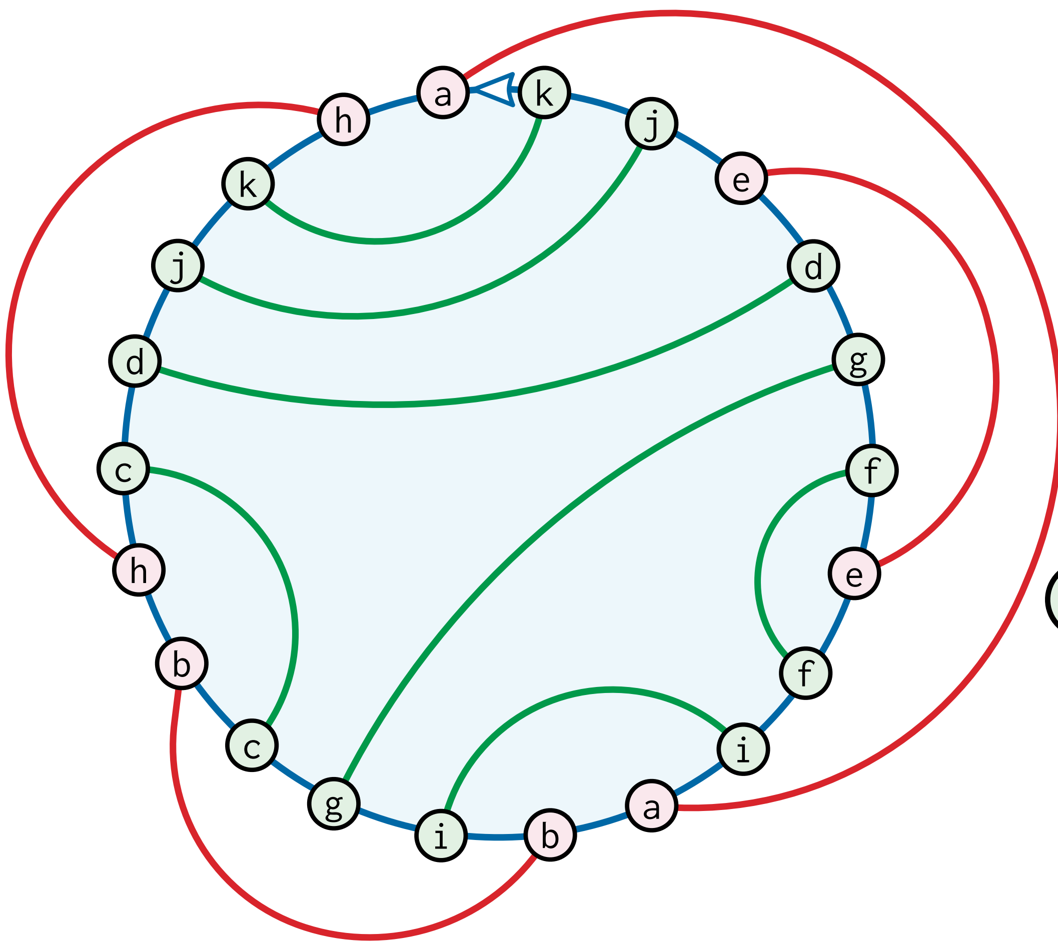
ahkjdchbcgibaifefgdejk



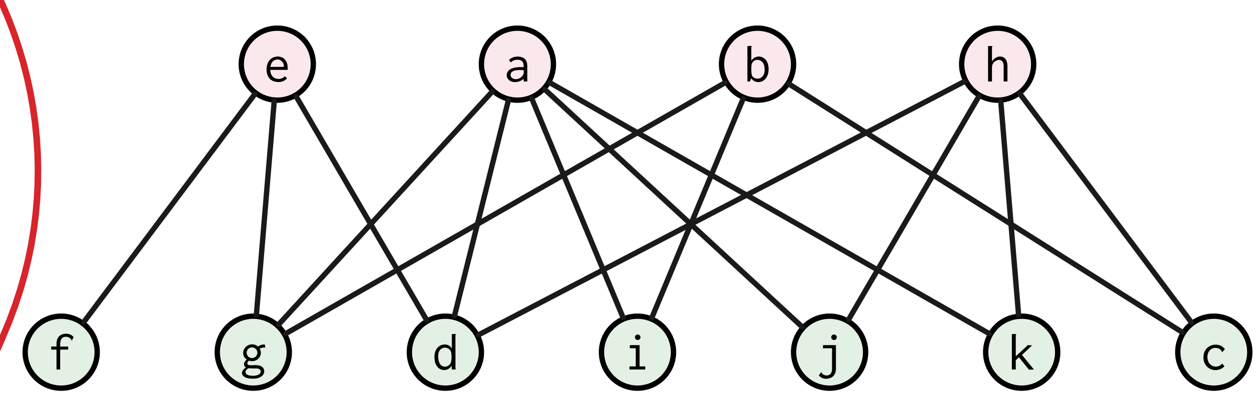
Untangling condition

[Dehn 1936]
[Rosenstiehl 1976]

The *interlacement graph* of the untangled code must be *bipartite*



ahkjdchbcgibaifefgdejk

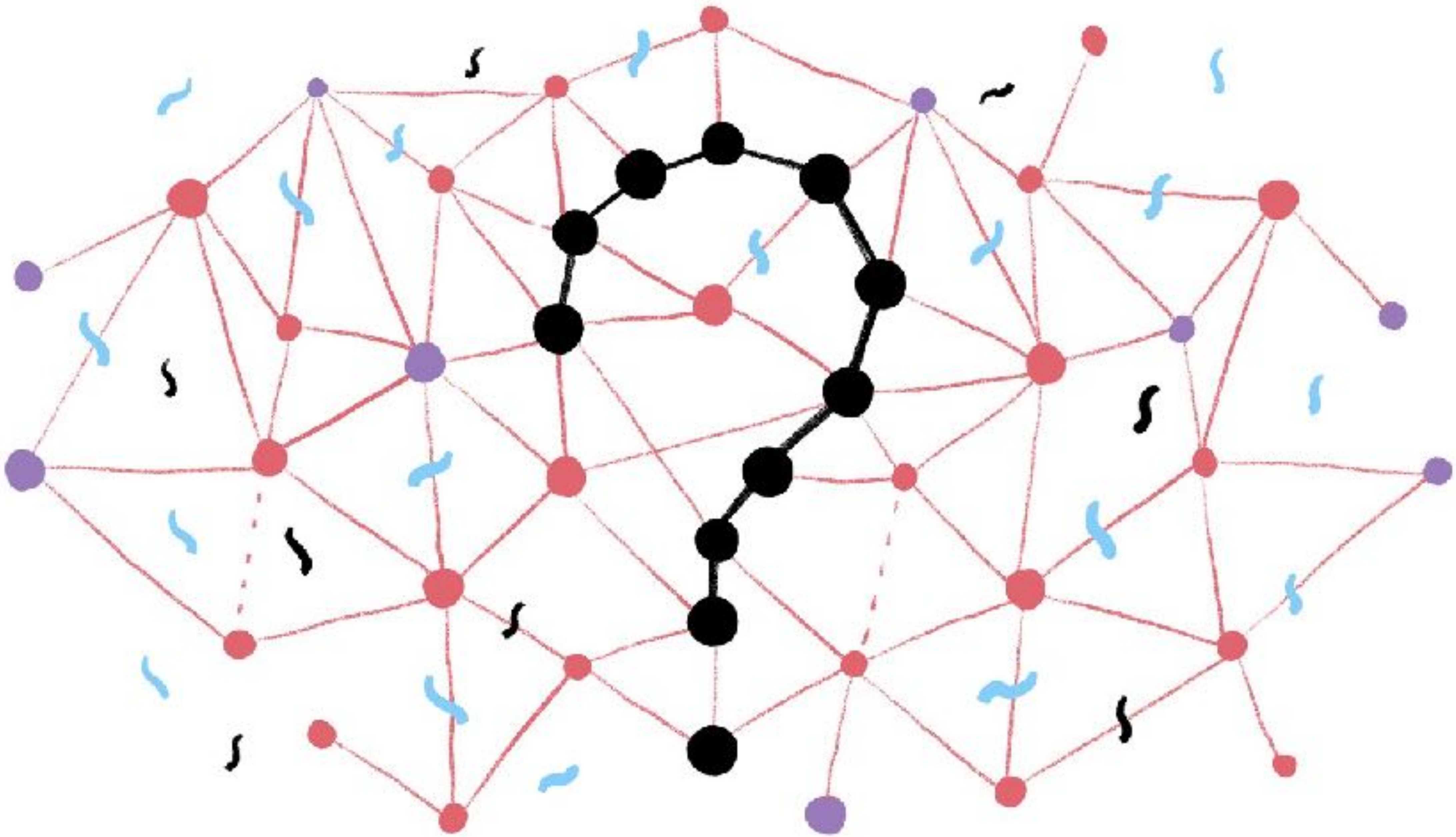


These two conditions suffice!

[Dehn 1936]

A string is the unsigned Gauss code of a planar curve
if and only if it satisfies *both*
Gauss' parity condition and Dehn's untangling condition.

Decoding Algorithm



Algorithm

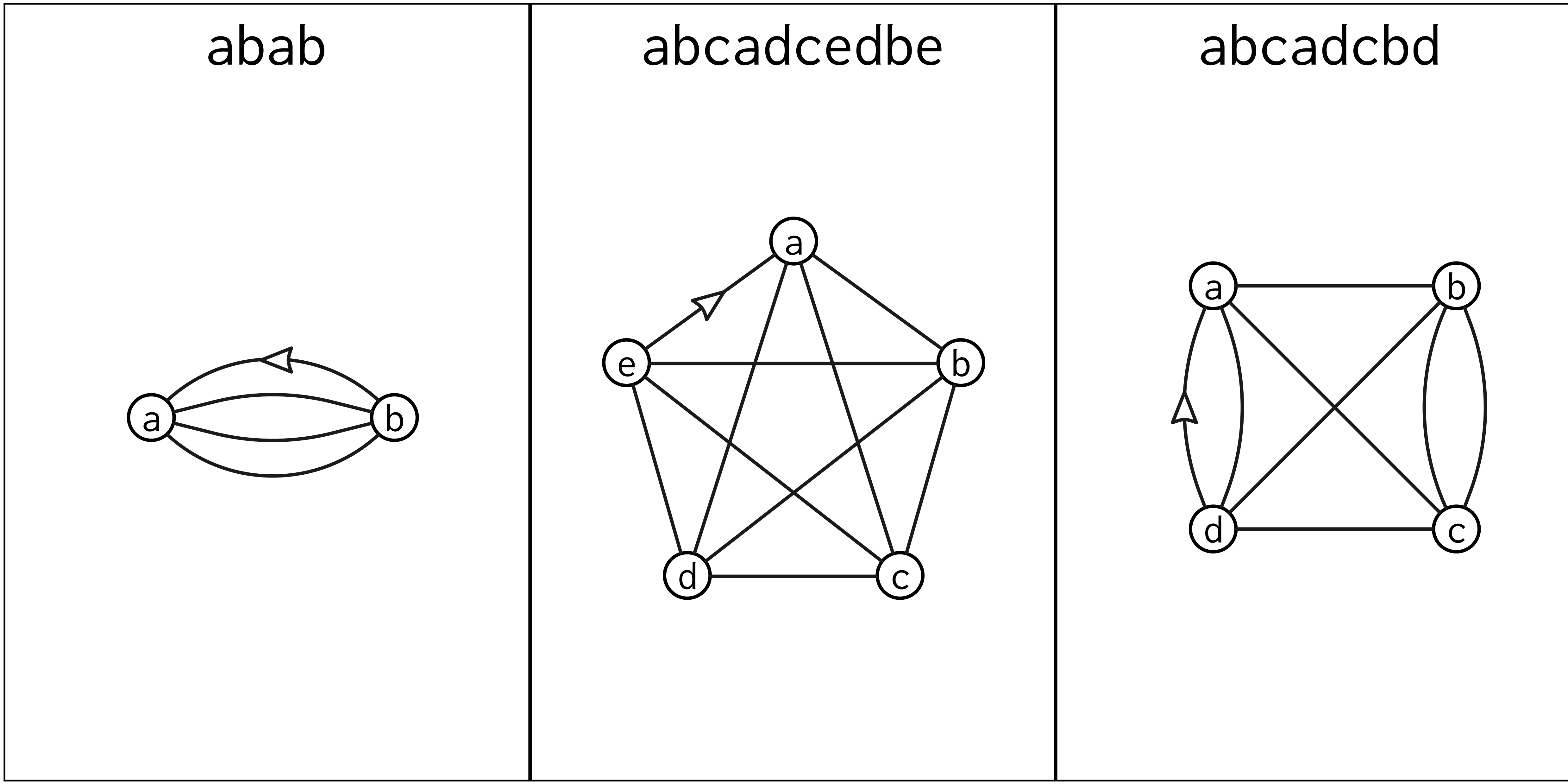
[Nagy 1927] + [Dehn 1936] + [Rosenstiehl 1976]

1. Build a 4-regular graph G from the input code.
2. Alternately direct the edges of G forward and backward
3. Find an Euler tour of G *(or fail)*
4. Extract an untangled code from the Euler tour
5. Build the interlacement graph of the untangled code
6. Find a bipartition of the interlacement graph *(or fail)*
7. Embed the untangled Gauss diagram into the plane
8. Contract the arcs of the embedded Gauss diagram

Three examples

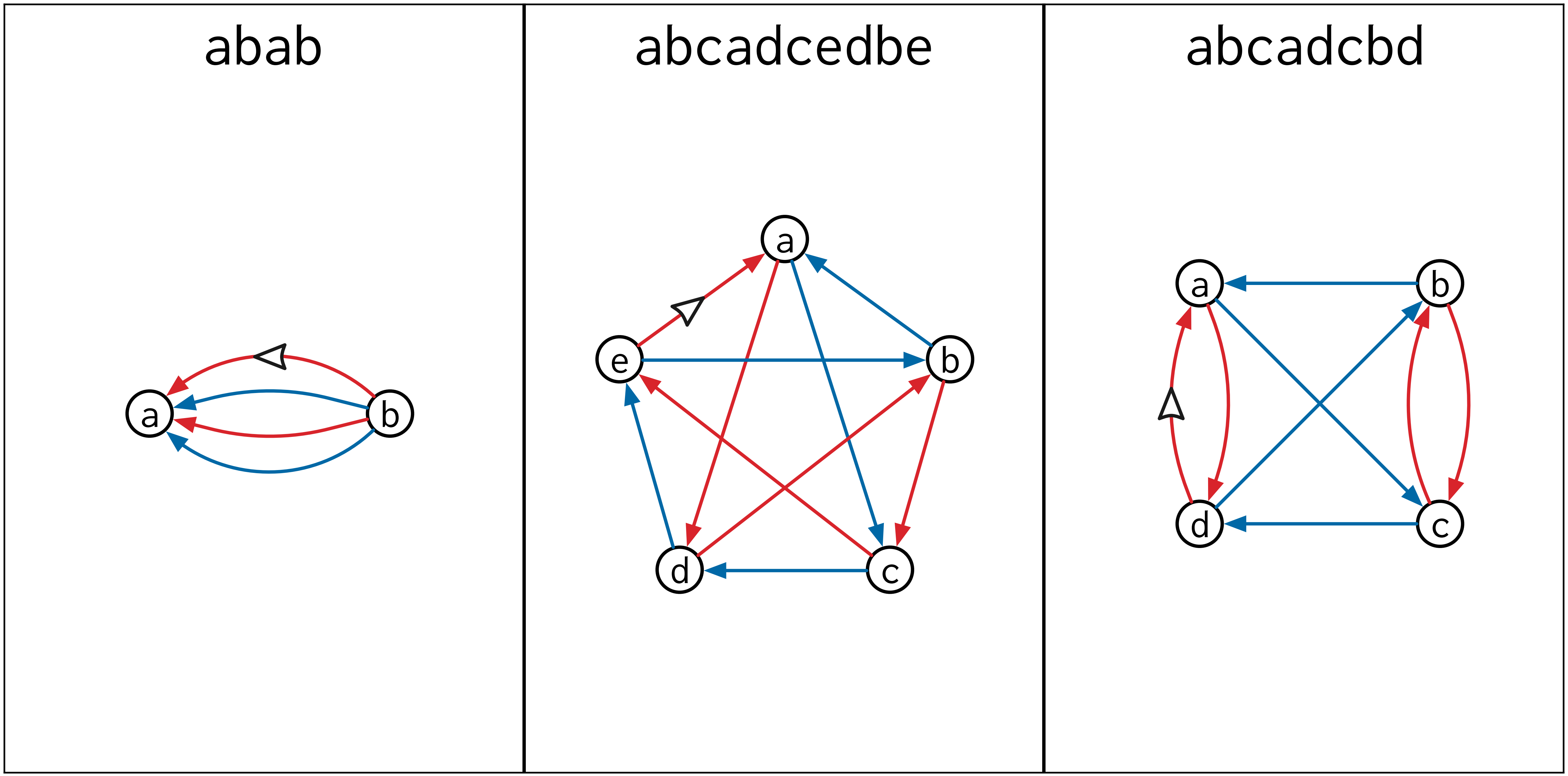
abab	abcadcedbe	abcadcbd

Build 4-regular graph from code



- O(n) time by brute force

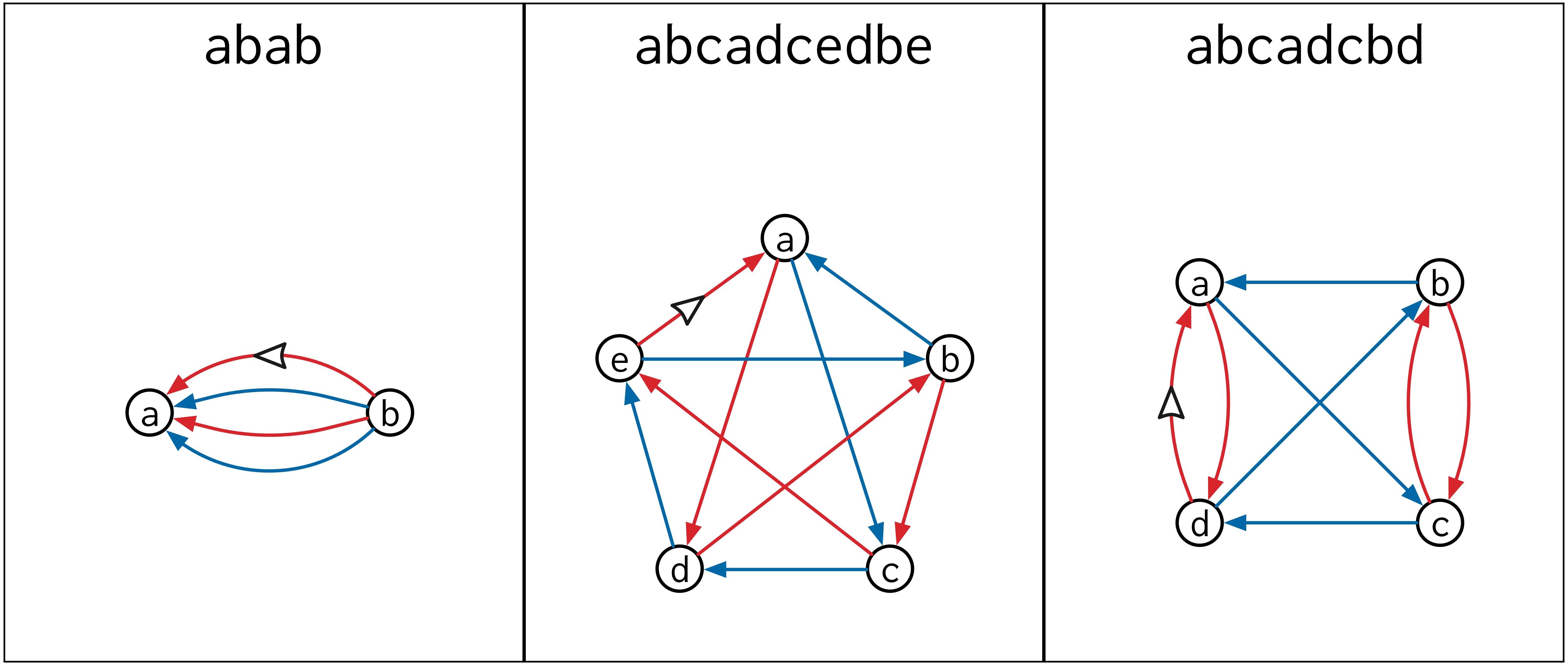
Alternate directions



- $O(n)$ time by brute force

Alternate directions

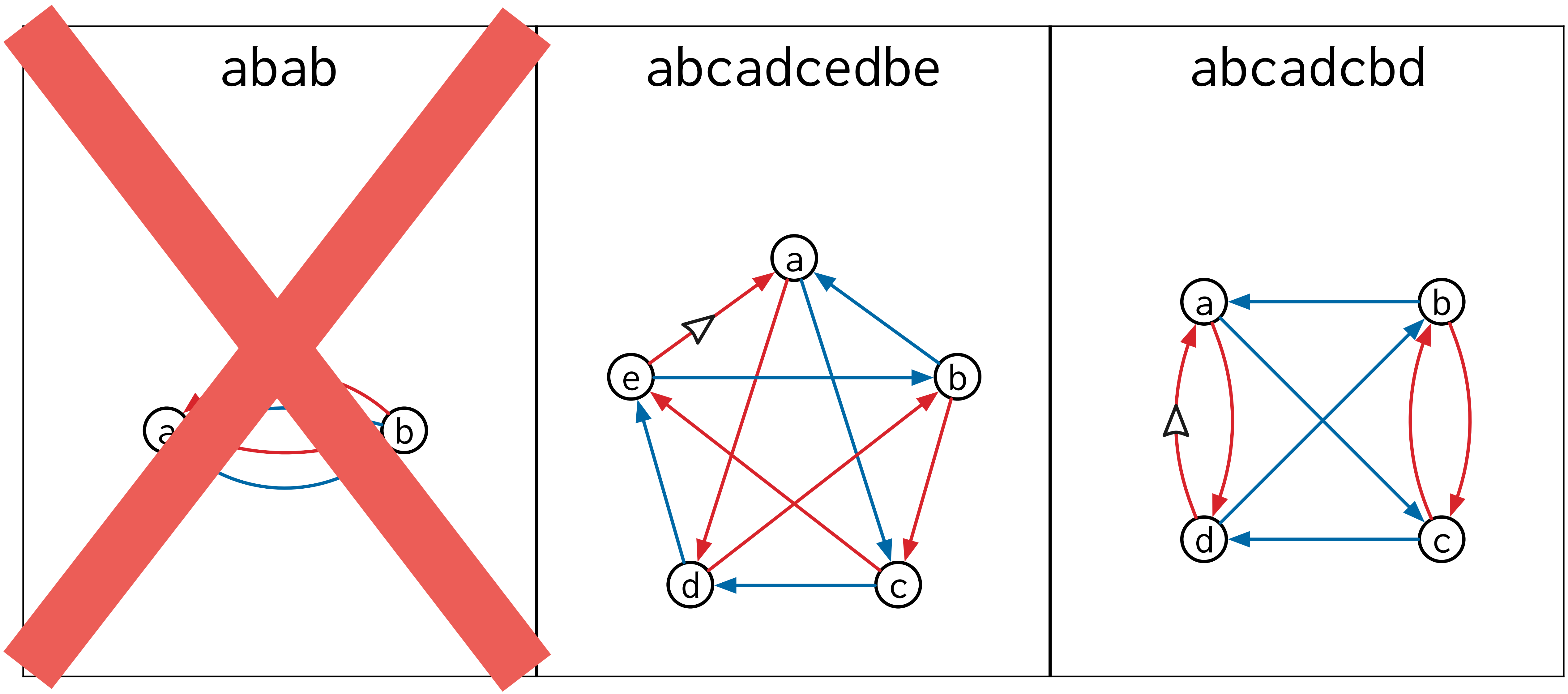
[Nagy 1927]



Gauss' parity condition holds *if and only if* every vertex in *G* has in-degree 2 and out-degree 2

Alternate directions

[Nagy 1927]

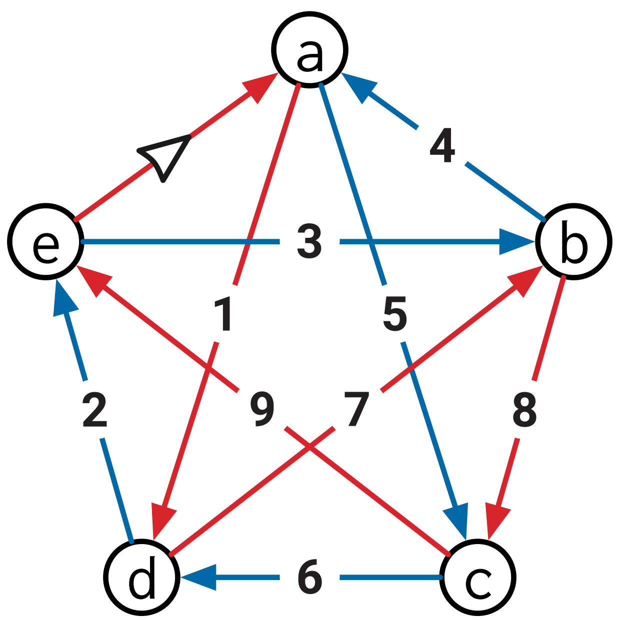


Gauss' parity condition holds *if and only if*
every vertex in *G* has in-degree 2 and out-degree 2

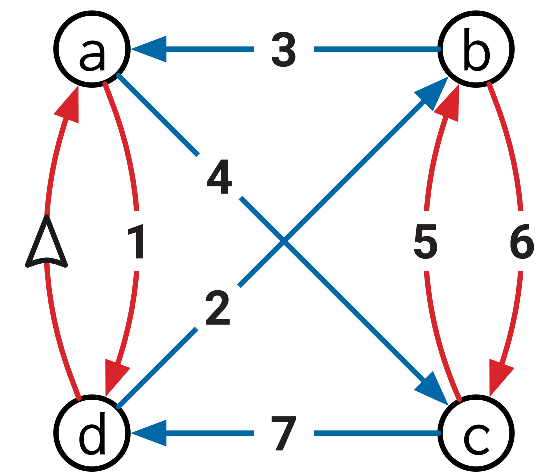
Euler tour

[Euler 1736]
[Hierholzer 1873]
[Good 1946]

abcadcedbe



abcadcbd

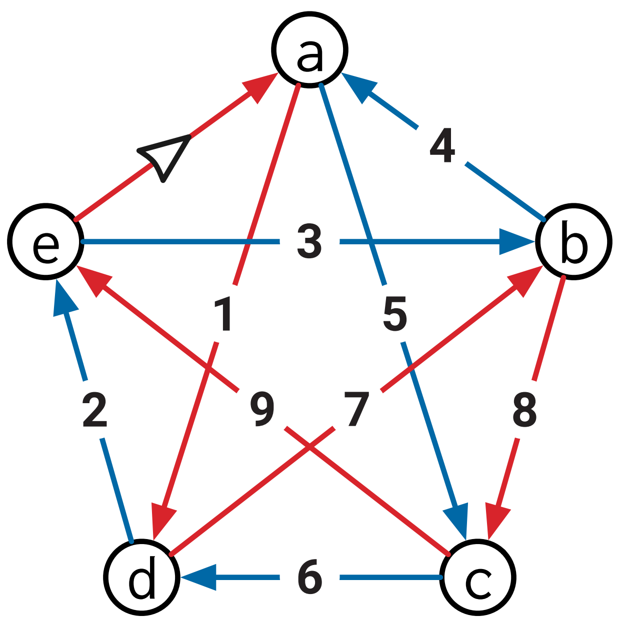


- $O(n)$ time via depth-first search [Hierholzer 1873]

Untangled code

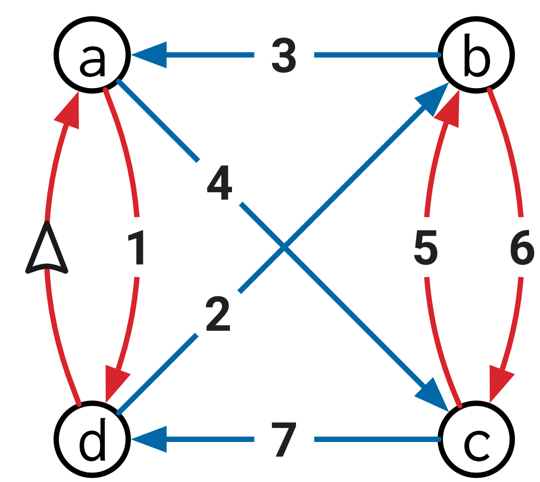
abcadcedbe

adebacdbce



abcadcbd

adbacbcd

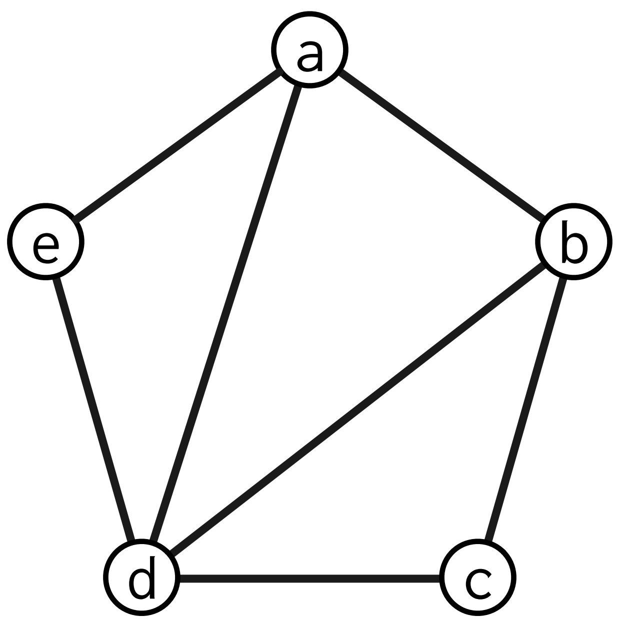


- $O(n)$ time by brute force

Interlacement graph

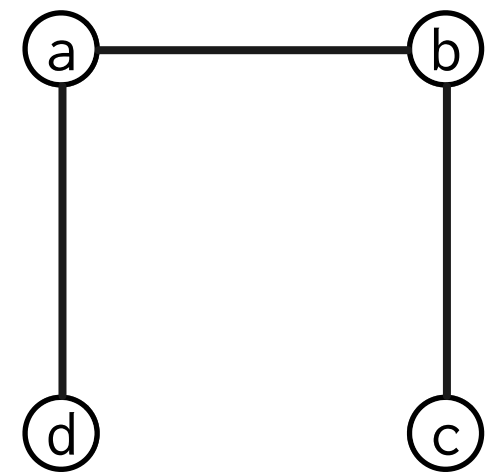
abcadcedbe

adebacdbce



abcadcbd

adbacbcd

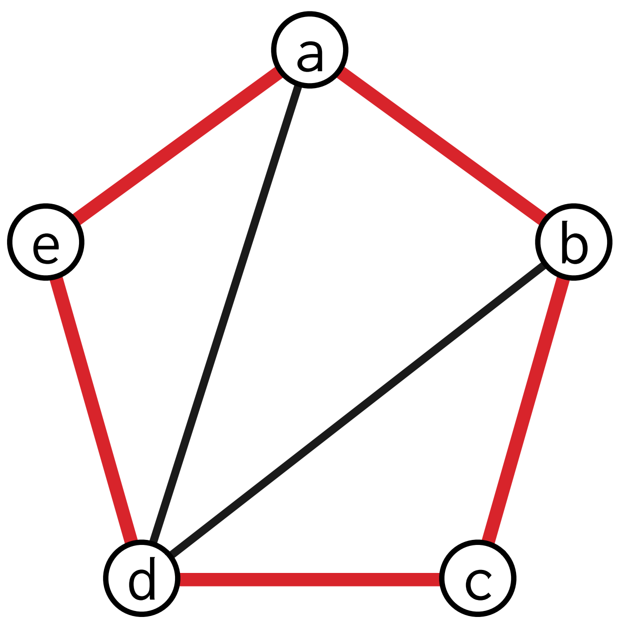


- $O(n^2)$ time by brute force

Bipartition

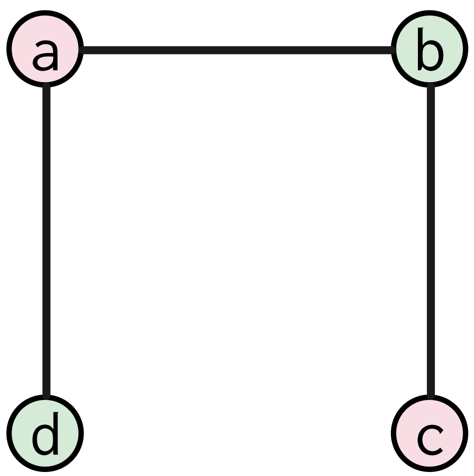
abcadcedbe

adebacdbce



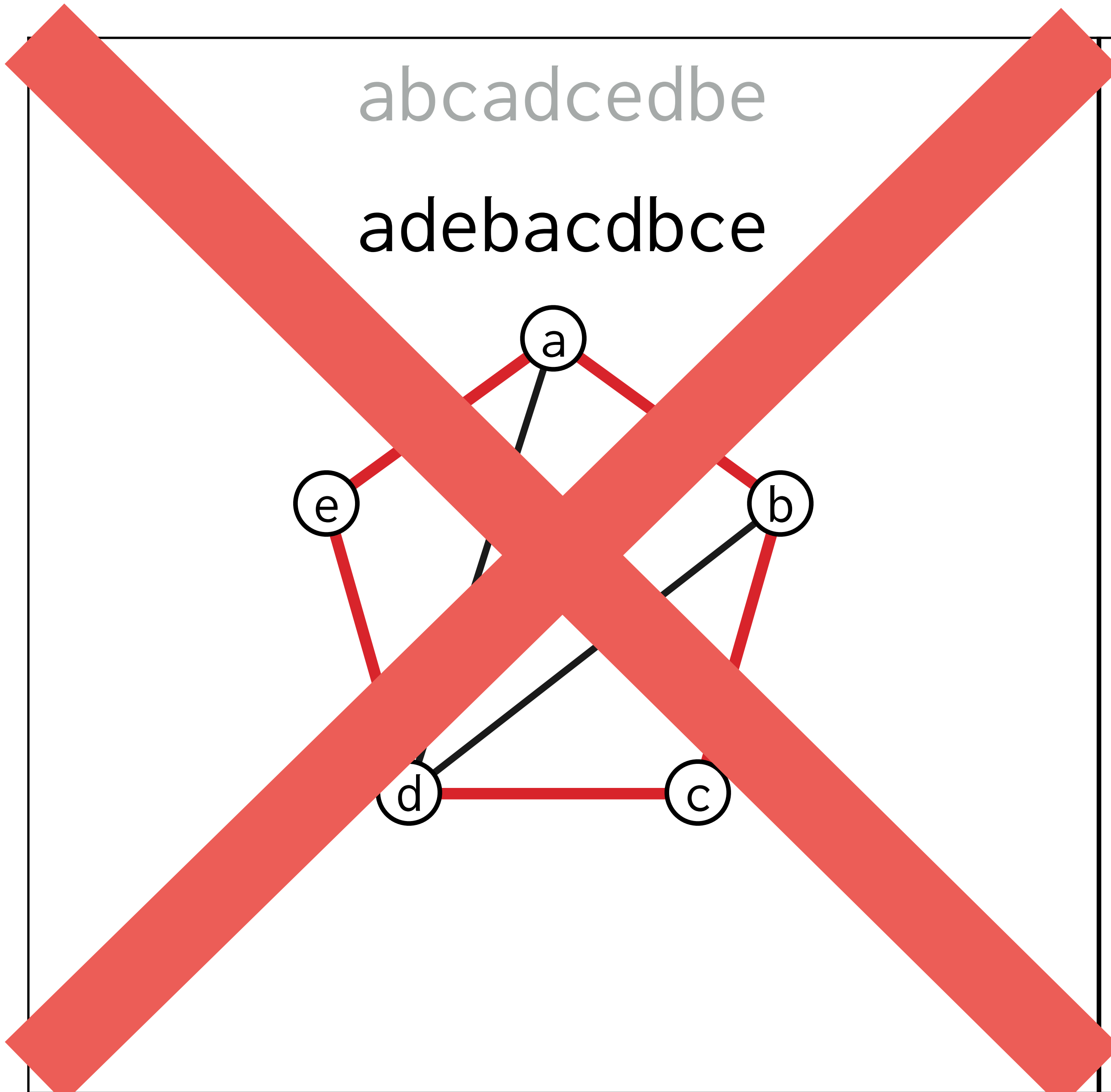
abcadcbd

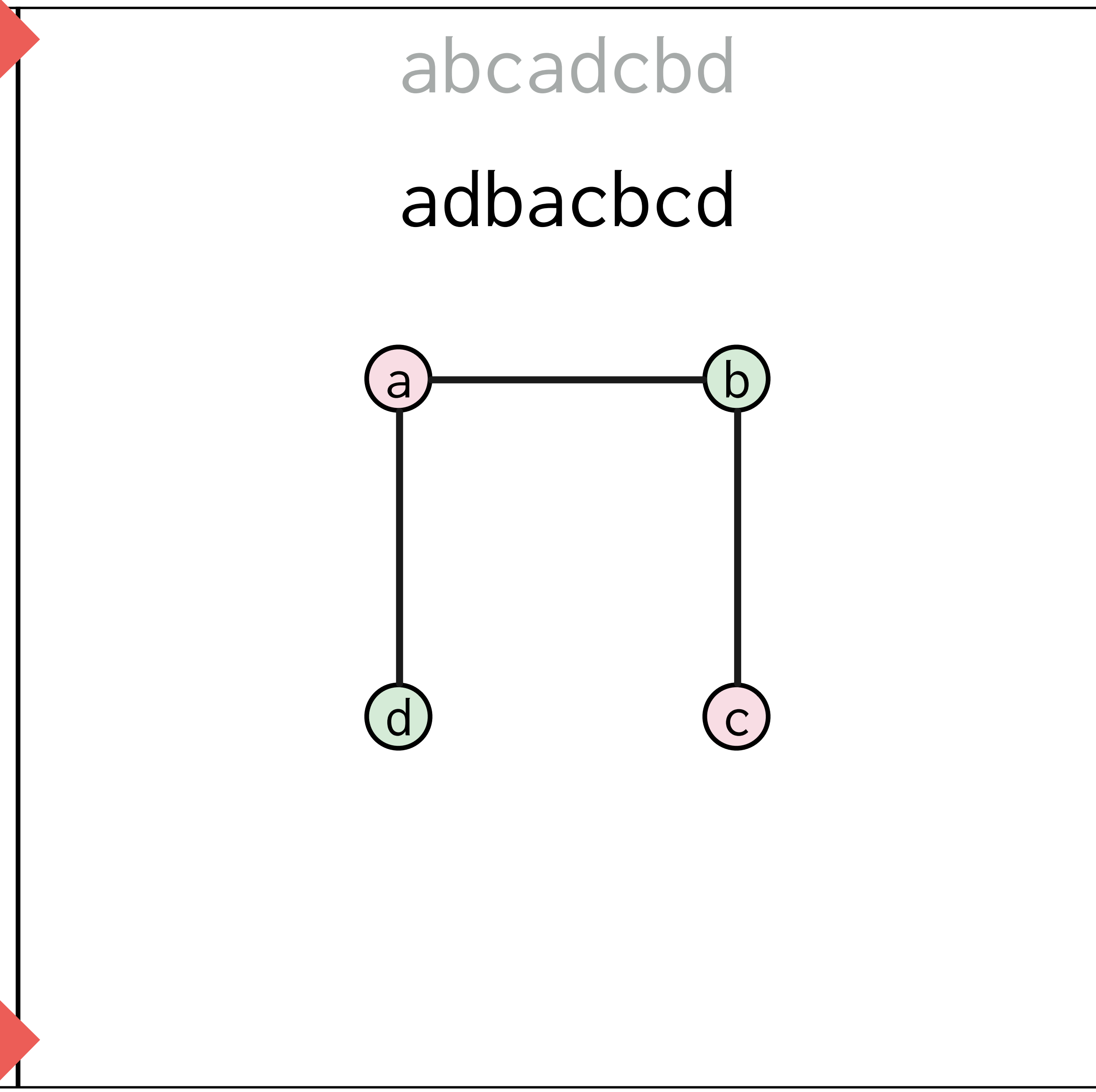
adbacbcd



- $O(n^2)$ time by whatever-first search

Bipartition



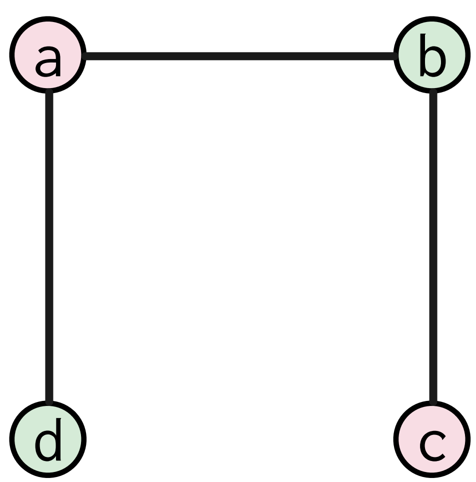


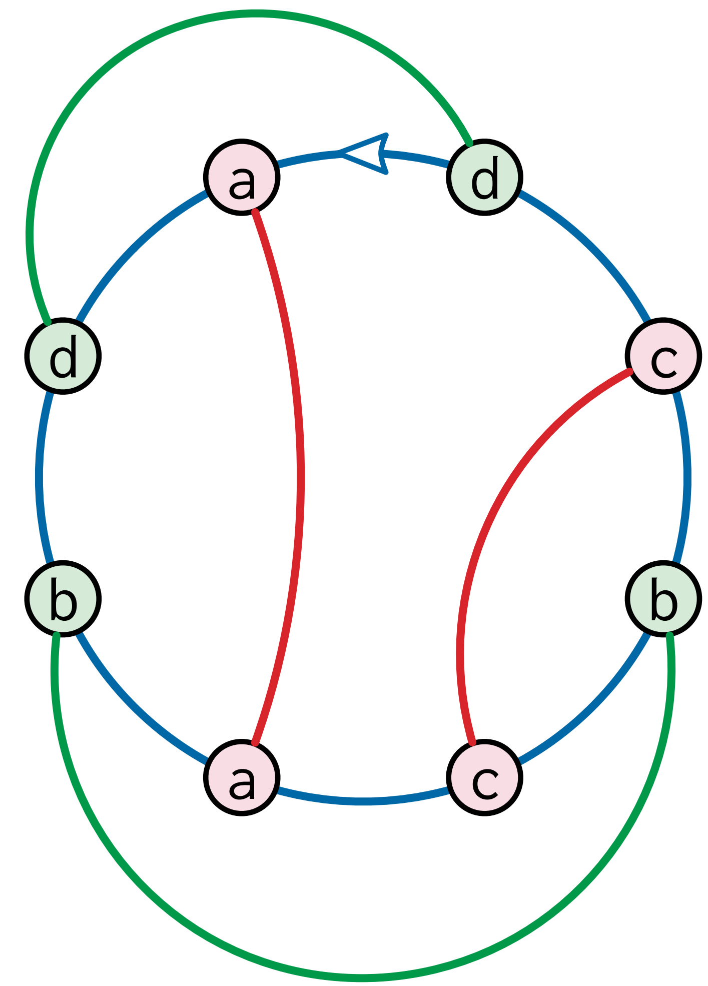
- $O(n^2)$ time by whatever-first search

Embed the Gauss diagram

abcadcbd

adbacbcd

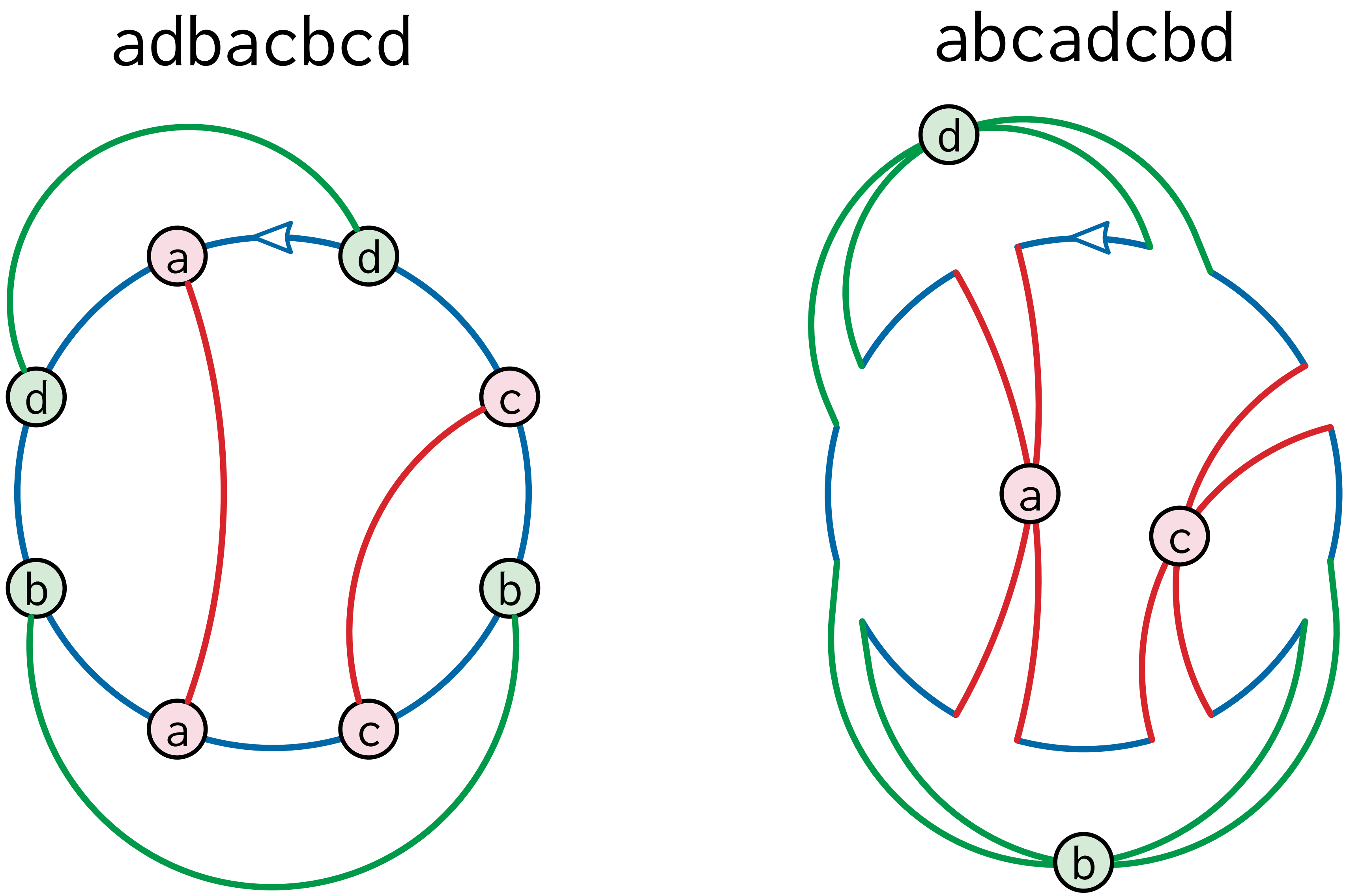




- $O(n)$ time by brute force

Contract diagram arcs

[Dehn 1936]



- O(1) time per edge = O(*n*) time total
- The final curve is consistent with the original Gauss code!

Contract diagram arcs

[Dehn 1936]

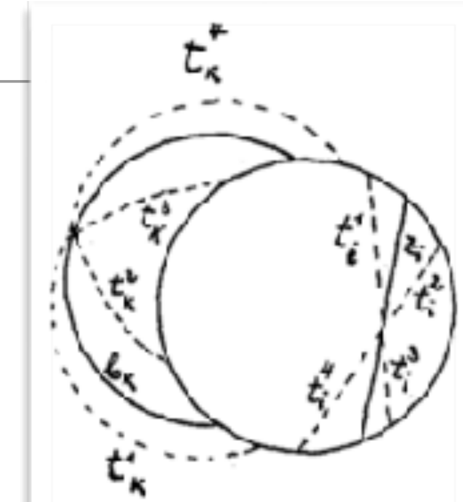
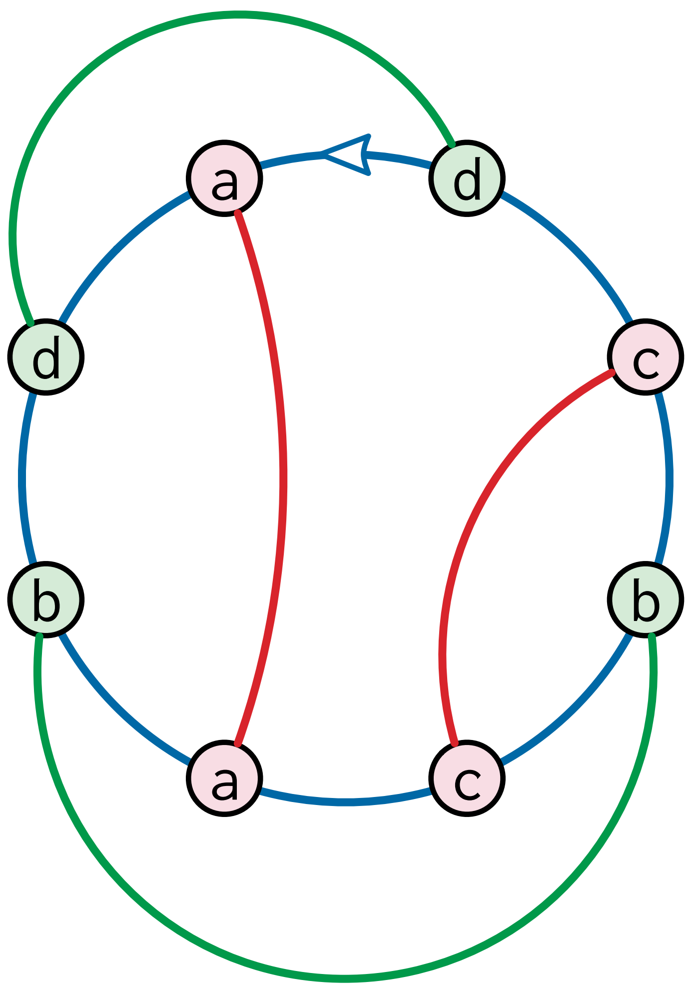
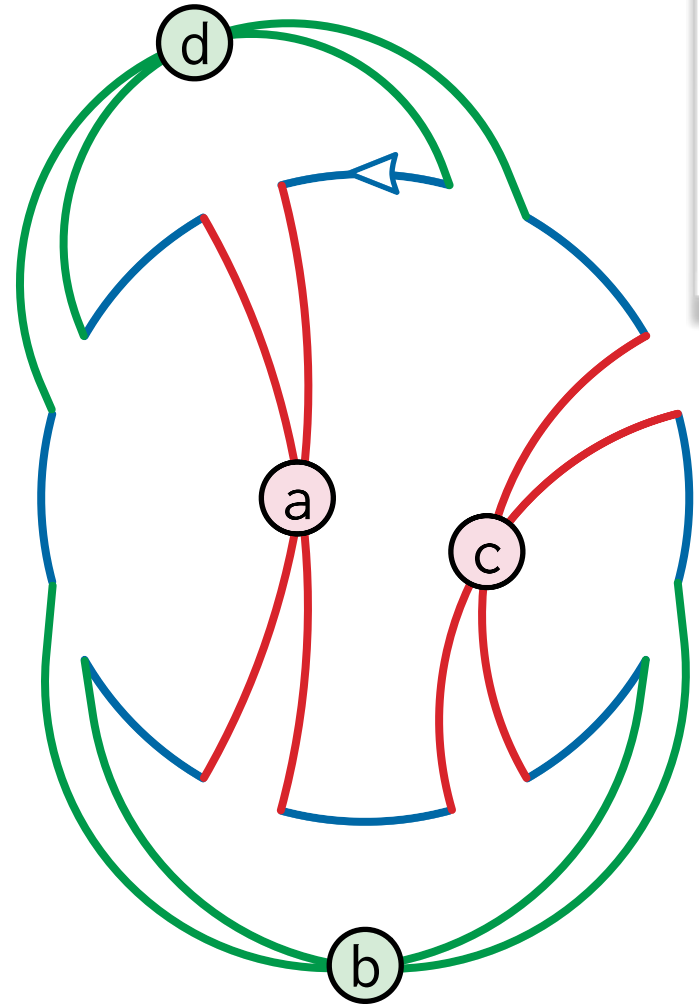


Fig. 18.

adbacbcd



abcadcbd



- O(1) time per edge = O(n) time total
- The final curve is consistent with the original Gauss code!

Algorithm summary

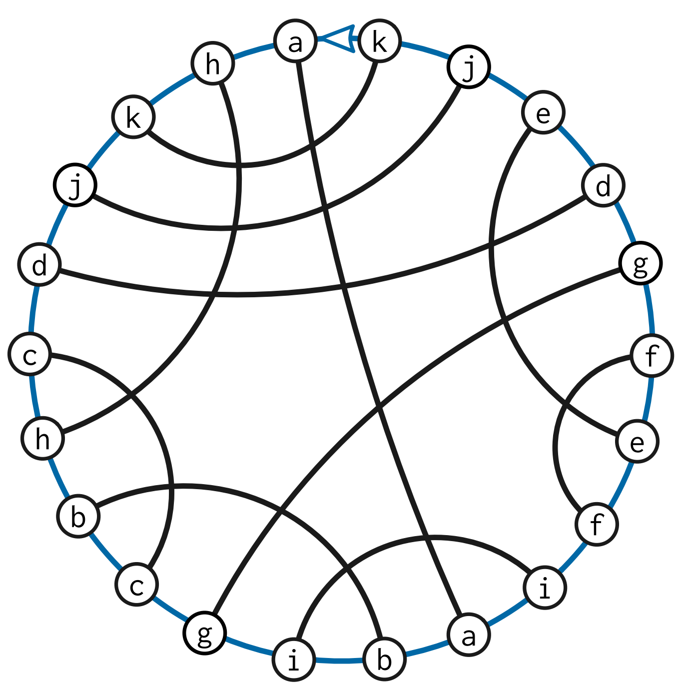
1. Build a 4-regular graph G from the input code
2. Alternately direct the edges of G forward and backward
3. Find an Euler tour of G *(or fail)*
4. Extract an untangled code from the Euler tour
5. Build the interlacement graph of the untangled code
6. Find a bipartition of the interlacement graph *(or fail)*
7. Embed the untangled Gauss diagram into the plane
8. Contract the arcs of the embedded Gauss diagram

- The entire algorithm runs in $O(n^2)$ time.
- Testing Dehn's untangling condition is the bottleneck, but there are faster algorithms for that.

Pile of twin stacks algorithm

[Rosenstiehl Tarjan 1984]

Classifies each arc of the Gauss diagram as “left” (inside) or “right” (outside) in $O(n)$ time, *without* explicitly building the interlacement graph.



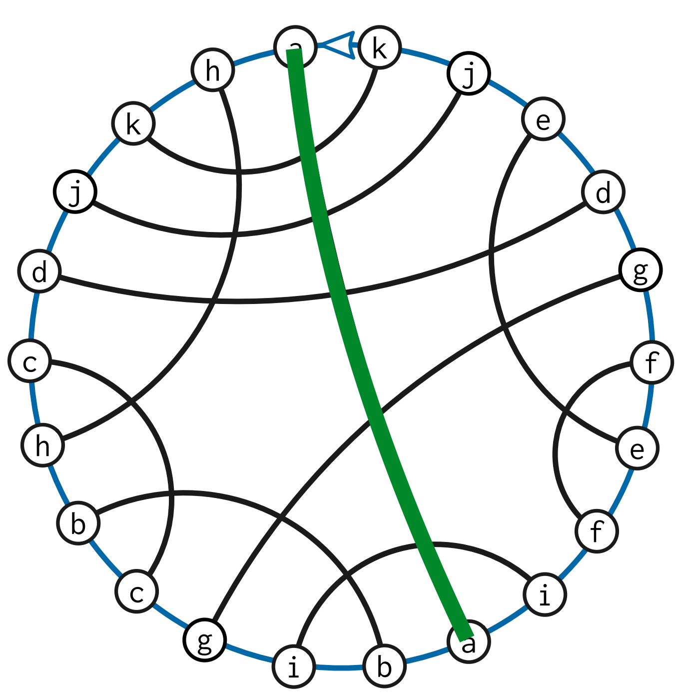
ahkjdchbcgibaifefgdejk

i	$T[i]$	$t[i]$	Pile of twin stacks	Operations	Interleaves found
0	a	12	$[12 \mid \bullet]$	new pair, push left	
1	h	6	$[6 \mid \bullet]$, $[12 \mid \bullet]$	new pair, push left	
2	k	21	$[6, 12 \mid 21]$	meld, push right	ka, kh
3	j	20	$[6, 12 \mid 20, 21]$	push right	jh
4	d	18	$[6, 12 \mid 18, 20, 21]$	push right	dh
5	c	8	$[6, 12 \mid 8, 18, 20, 21]$	push right	ch
6	h	1	$[12 \mid 8, 18, 20, 21]$	pop left	
7	b	11	$[11, 12 \mid 8, 18, 20, 21]$	push left	bc
8	c	5	$[11, 12 \mid 18, 20, 21]$	pop right	
9	g	17	$[11, 12 \mid 17, 18, 20, 21]$	push right	gb
10	i	13	$[11, 12 \mid 13, 17, 18, 20, 21]$	push right	ib
11	b	7	$[12 \mid 13, 17, 18, 20, 21]$	pop left	
12	a	0	$[\bullet \mid 13, 17, 18, 20, 21]$	pop left	
13	i	10	$[\bullet \mid 17, 18, 20, 21]$	pop right	
14	f	16	$[16 \mid \bullet]$, $[\bullet \mid 17, 18, 20, 21]$	new pair, push left	
15	e	19	$[19 \mid 16, 17, 18, 20, 21]$	swap top pair, meld, push right	ef, eg
16	f	14	$[19 \mid 17, 18, 20, 21]$	pop right	
17	g	9	$[19 \mid 18, 20, 21]$	pop right	
18	d	4	$[19 \mid 20, 21]$	pop right	
19	e	15	$[\bullet \mid 20, 21]$	pop left	
20	j	3	$[\bullet \mid 21]$	pop right	
21	k	2	$\varnothing$	pop right, pop empty pair	

Pile of twin stacks algorithm

[Rosenstiehl Tarjan 1984]

Classifies each arc of the Gauss diagram as “left” (inside) or “right” (outside) in O(n) time, *without* explicitly building the interlacement graph.



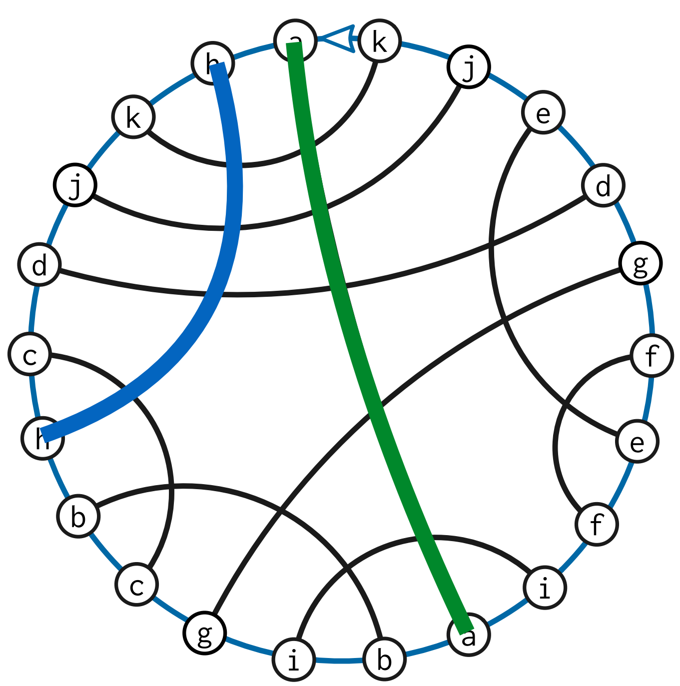
ahkjdchbcgibaifefgdejk

i	$T[i]$	$t[i]$	Pile of twin stacks	Operations	Interleaves found
0	a	12	[12 | •]	new pair, push left	
1	h	6	[6 | •] , [12 | •]	new pair, push left	
2	k	21	[6, 12 | 21]	meld, push right	ka, kh
3	j	20	[6, 12 | 20, 21]	push right	jh
4	d	18	[6, 12 | 18, 20, 21]	push right	dh
5	c	8	[6, 12 | 8, 18, 20, 21]	push right	ch
6	h	1	[12 | 8, 18, 20, 21]	pop left	
7	b	11	[11, 12 | 8, 18, 20, 21]	push left	bc
8	c	5	[11, 12 | 18, 20, 21]	pop right	
9	g	17	[11, 12 | 17, 18, 20, 21]	push right	gb
10	i	13	[11, 12 | 13, 17, 18, 20, 21]	push right	ib
11	b	7	[12 | 13, 17, 18, 20, 21]	pop left	
12	a	0	[• | 13, 17, 18, 20, 21]	pop left	
13	i	10	[• | 17, 18, 20, 21]	pop right	
14	f	16	[16 | •] , [• | 17, 18, 20, 21]	new pair, push left	
15	e	19	[19 | 16, 17, 18, 20, 21]	swap top pair, meld, push right	ef, eg
16	f	14	[19 | 17, 18, 20, 21]	pop right	
17	g	9	[19 | 18, 20, 21]	pop right	
18	d	4	[19 | 20, 21]	pop right	
19	e	15	[• | 20, 21]	pop left	
20	j	3	[• | 21]	pop right	
21	k	2	∅	pop right, pop empty pair	

Pile of twin stacks algorithm

[Rosenstiehl Tarjan 1984]

Classifies each arc of the Gauss diagram as “left” (inside) or “right” (outside) in O(n) time, *without* explicitly building the interlacement graph.



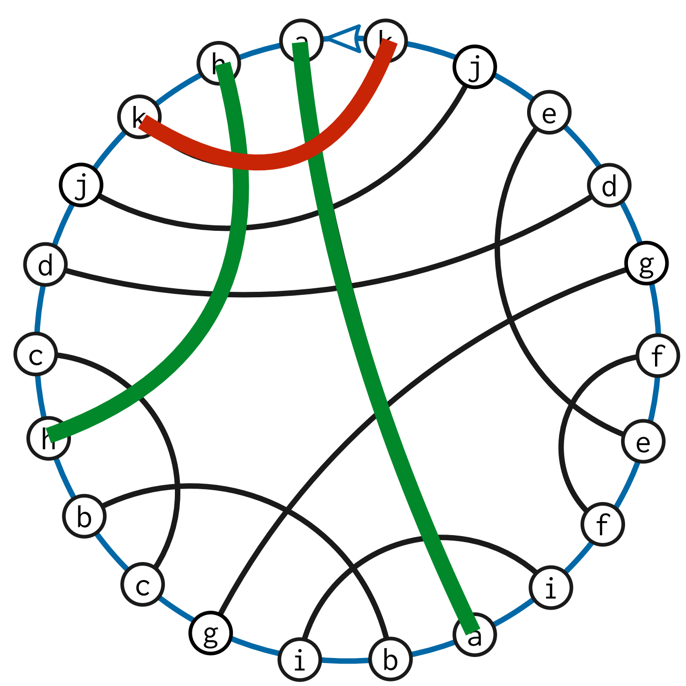
ahkjdchbcgibaifefgdejk

i	$T[i]$	$t[i]$	Pile of twin stacks	Operations	Interleaves found
0	a	12	[12 | •]	new pair, push left	
1	h	6	[6 | •] , [12 | •]	new pair, push left	
2	k	21	[6, 12 | 21]	meld, push right	ka, kh
3	j	20	[6, 12 | 20, 21]	push right	jh
4	d	18	[6, 12 | 18, 20, 21]	push right	dh
5	c	8	[6, 12 | 8, 18, 20, 21]	push right	ch
6	h	1	[12 | 8, 18, 20, 21]	pop left	
7	b	11	[11, 12 | 8, 18, 20, 21]	push left	bc
8	c	5	[11, 12 | 18, 20, 21]	pop right	
9	g	17	[11, 12 | 17, 18, 20, 21]	push right	gb
10	i	13	[11, 12 | 13, 17, 18, 20, 21]	push right	ib
11	b	7	[12 | 13, 17, 18, 20, 21]	pop left	
12	a	0	[• | 13, 17, 18, 20, 21]	pop left	
13	i	10	[• | 17, 18, 20, 21]	pop right	
14	f	16	[16 | •] , [• | 17, 18, 20, 21]	new pair, push left	
15	e	19	[19 | 16, 17, 18, 20, 21]	swap top pair, meld, push right	ef, eg
16	f	14	[19 | 17, 18, 20, 21]	pop right	
17	g	9	[19 | 18, 20, 21]	pop right	
18	d	4	[19 | 20, 21]	pop right	
19	e	15	[• | 20, 21]	pop left	
20	j	3	[• | 21]	pop right	
21	k	2	∅	pop right, pop empty pair	

Pile of twin stacks algorithm

[Rosenstiehl Tarjan 1984]

Classifies each arc of the Gauss diagram as "left" (inside) or "right" (outside) in $O(n)$ time, *without* explicitly building the interlacement graph.



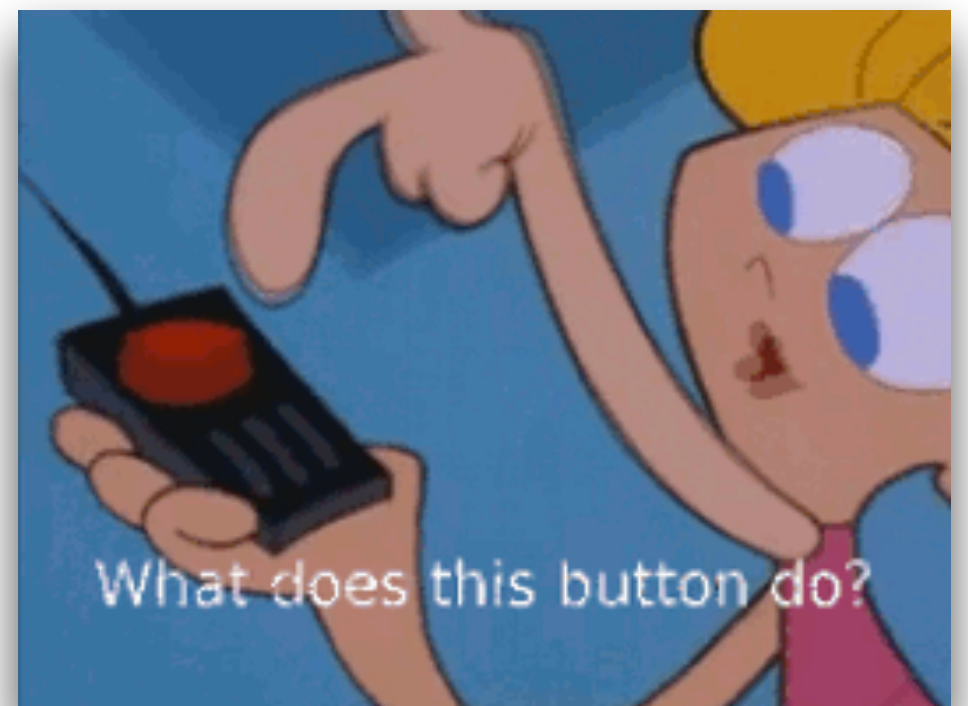
ahkjdchbcgibaifefgdejk

i	$T[i]$	$t[i]$	Pile of twin stacks	Operations	Interleaves found
0	a	12	[12 | •]	new pair, push left	
1	h	6	[6 | •] , [12 | •]	new pair, push left	
2	k	21	[6, 12 | 21]	meld, push right	ka, kh
3	j	20	[6, 12 | 20, 21]	push right	jh
4	d	18	[6, 12 | 18, 20, 21]	push right	dh
5	c	8	[6, 12 | 8, 18, 20, 21]	push right	ch
6	h	1	[12 | 8, 18, 20, 21]	pop left	
7	b	11	[11, 12 | 8, 18, 20, 21]	push left	bc
8	c	5	[11, 12 | 18, 20, 21]	pop right	
9	g	17	[11, 12 | 17, 18, 20, 21]	push right	gb
10	i	13	[11, 12 | 13, 17, 18, 20, 21]	push right	ib
11	b	7	[12 | 13, 17, 18, 20, 21]	pop left	
12	a	0	[• | 13, 17, 18, 20, 21]	pop left	
13	i	10	[• | 17, 18, 20, 21]	pop right	
14	f	16	[16 | •] , [• | 17, 18, 20, 21]	new pair, push left	
15	e	19	[19 | 16, 17, 18, 20, 21]	swap top pair, meld, push right	ef, eg
16	f	14	[19 | 17, 18, 20, 21]	pop right	
17	g	9	[19 | 18, 20, 21]	pop right	
18	d	4	[19 | 20, 21]	pop right	
19	e	15	[• | 20, 21]	pop left	
20	j	3	[• | 21]	pop right	
21	k	2	∅	pop right, pop empty pair	

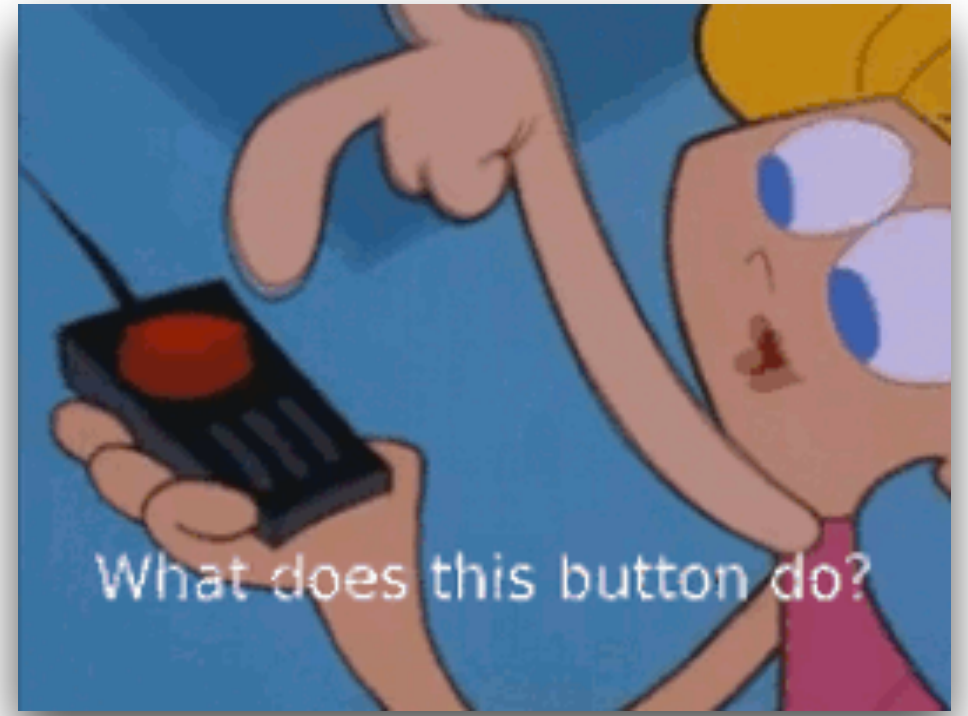
Other characterizations

- *[Tait 1877]*
- *[Nagy 1927]*
- *[Treybig 1968]*
- *[Marx 1967, 1969]*
- *[Bouchet 1972]*
- *[Lovász Marx 1976]*
- *[Rosenstiehl 1976]*
- *[Read Rosenstiehl 1976]*
- *[Dowker Thistlethwaite 1983]*
- *[Chaves Weber 1994]*
- *[Cairns Elton 1996]*
- *[de Fraysseix, Ossona de Mendez 1999]*
- *[Burckel 2001]*
- *[Grinblat Lopatkin 2017]*

Extensions and Open Problems



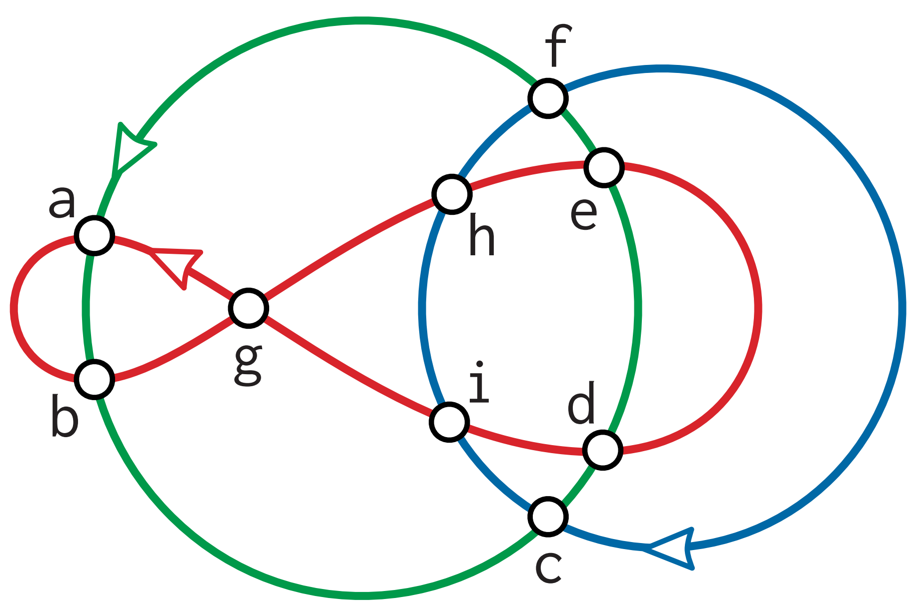
Extensions and Open Problems



Multiple curves

[Dehn 1936]

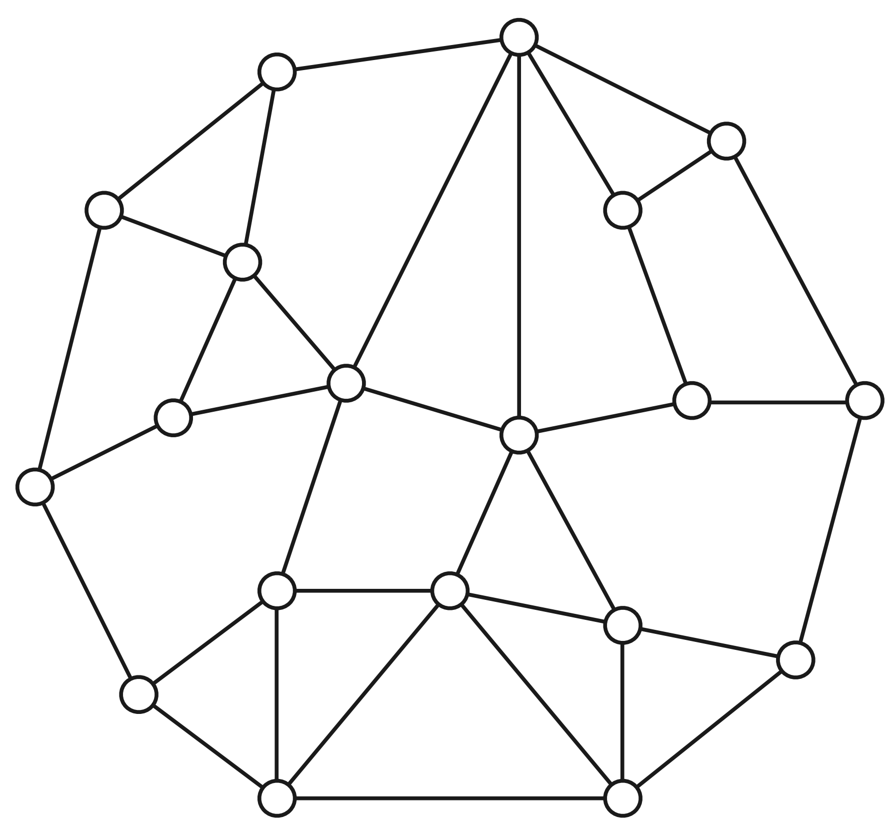
- "Gauss paragraphs"
- Need two additional parity conditions
  - Each "sentence" has even length
  - For any two symbols that appear in the same two "sentences", the substrings they delimit have even total length
- The algorithm is essentially unchanged



abcdef · abghedig · hfci

Left-Right Planarity Test

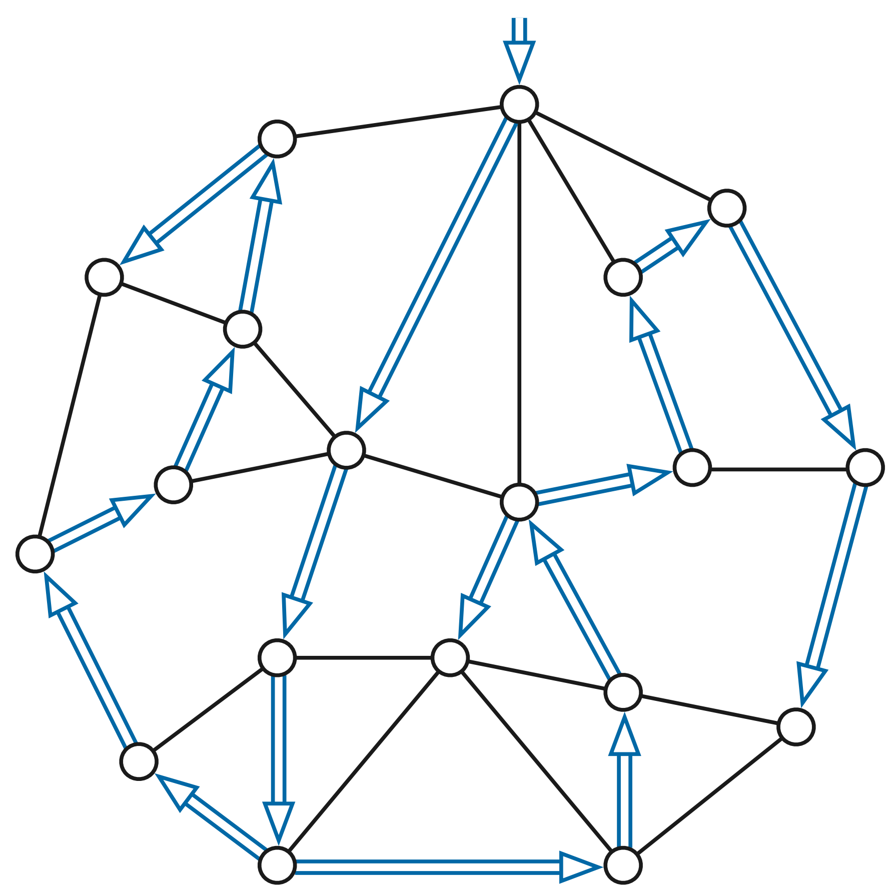
[Wu 1955, 1985] [Liu 1978, 1988]
[de Fraysseix, Rosenstiehl 1982, 1985]
[Xu 1989] [Cai Han Tarjan 1993]
[de Fraysseix, Ossona de Mendez, Rosenstiehl 2006]



Left-Right Planarity Test

[Wu 1955, 1985] [Liu 1978, 1988]
[de Fraysseix, Rosenstiehl 1982, 1985]
[Xu 1989] [Cai Han Tarjan 1993]
[de Fraysseix, Ossona de Mendez, Rosenstiehl 2006]

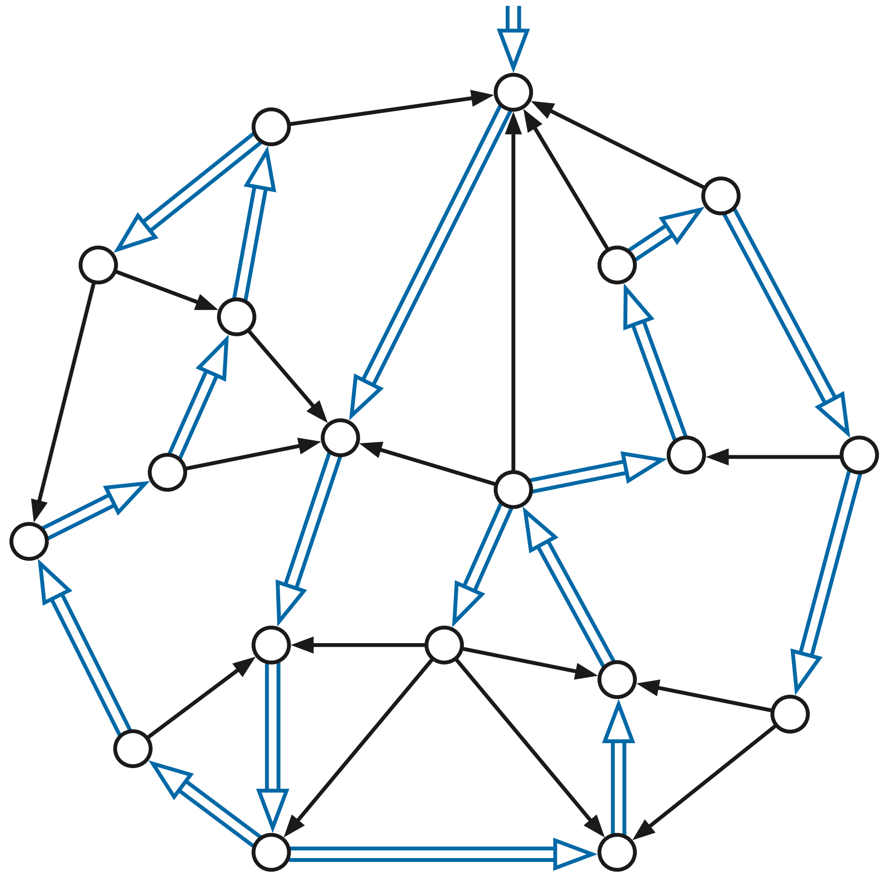
- Fix a depth-first search tree of G.
  [Wiener 1873] [Trémaux c.1882]



Left-Right Planarity Test

[Wu 1955, 1985] [Liu 1978, 1988]
[de Fraysseix, Rosenstiehl 1982, 1985]
[Xu 1989] [Cai Han Tarjan 1993]
[de Fraysseix, Ossona de Mendez, Rosenstiehl 2006]

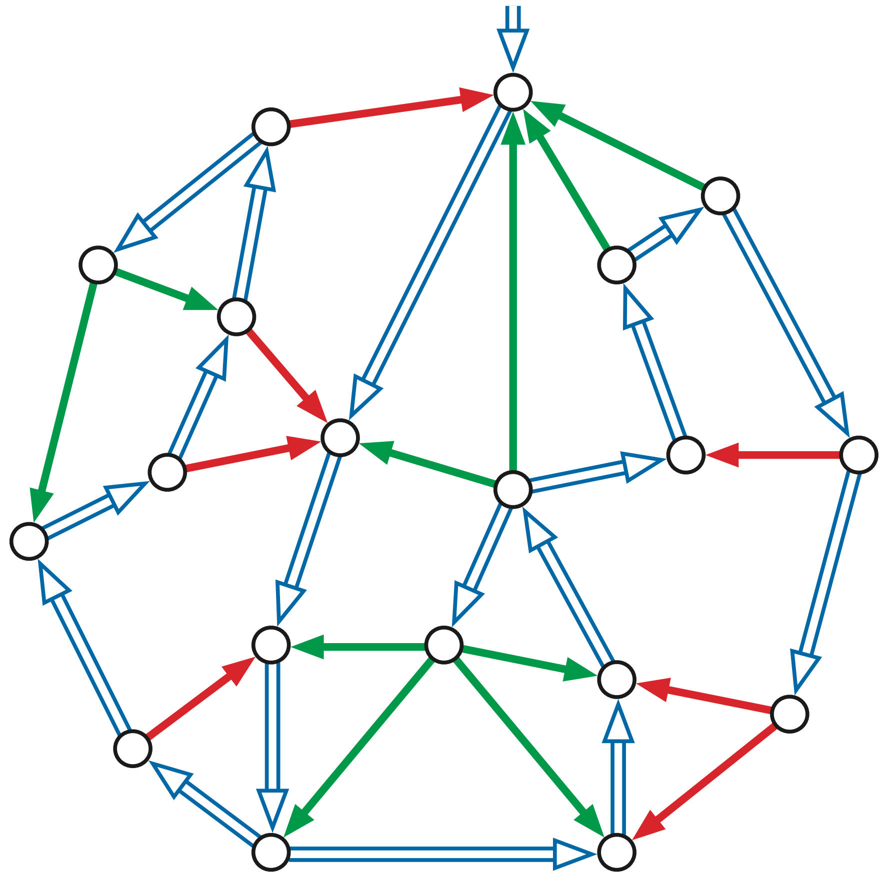
- Fix a depth-first search tree of G.
  [Wiener 1873] [Trémaux c.1882]
- Each non-tree edge defines a directed fundamental cycle.



Left-Right Planarity Test

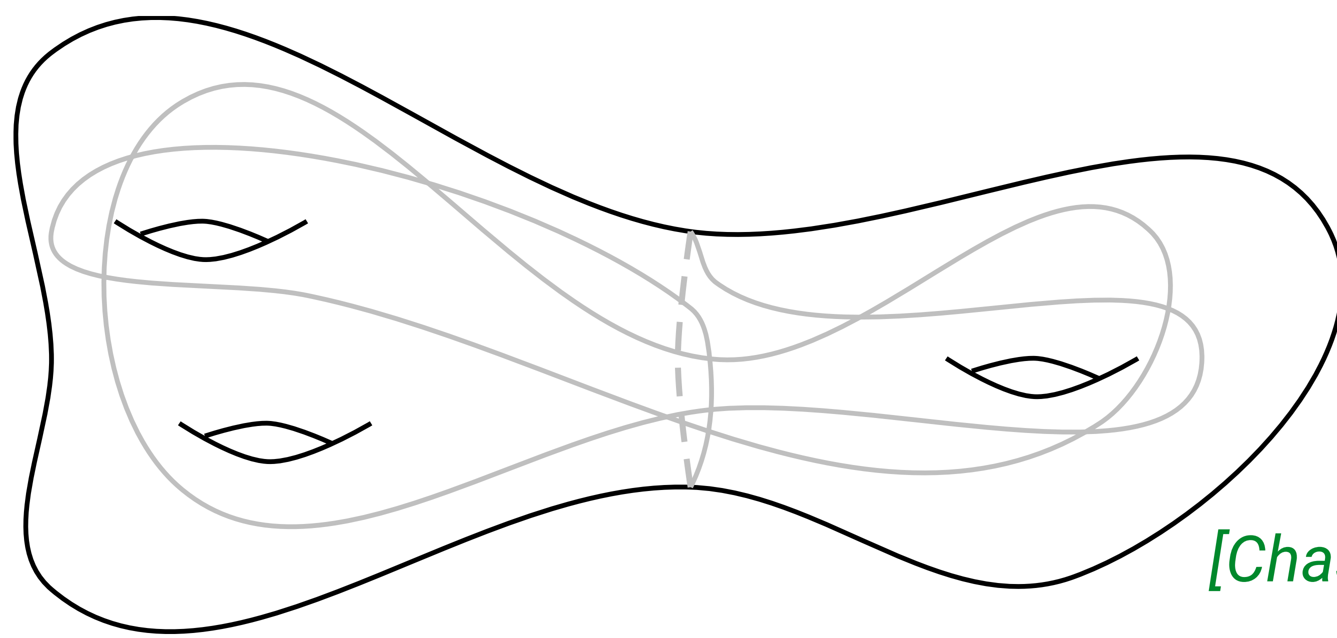
[Wu 1955, 1985] [Liu 1978, 1988]
[de Fraysseix, Rosenstiehl 1982, 1985]
[Xu 1989] [Cai Han Tarjan 1993]
[de Fraysseix, Ossona de Mendez, Rosenstiehl 2006]

- Fix a depth-first search tree of G.
  [Wiener 1873] [Trémaux c.1882]
- Each non-tree edge defines a directed fundamental cycle.
- G is planar iff the interlacement graph of these fundamental cycles is bipartite.



Open problem

- How quickly can we recognize Gauss codes of curves in more complicated surfaces? *[Dehn 1936]*
  - Signed codes are still easy! (Does $V-E+F = 2-2g$?)
  - Only existing algorithm for unsigned codes: Try all 2^n signings!



[Chas 2007]

g=3, `abcdefghiejabcdghifj`

Open problems

- How quickly can we recognize Gauss codes of curves in more complicated surfaces? *[Dehn 1936]*

- ...of *null-homologous* curves...?
  - = has a checkerboard coloring = has an Alexander numbering = "*lacet*"
    [Lins Richter Shank 1987] [Crapo Rosenstiehl 2001] [Lins, Oliveira-Lima, Silva 2008]

For a given Π, there is now an obvious algorithm to determine the surface S of least connectivity in which Π arises as a left-right path try each of the $2^{|E|}$ choices for K and pick one that gives the smallest rank for $J^2 + J + JK + KJ$ Is there an algorithm to determine S whose running time is bounded by a polynomial in $|E|$?

Proposition 4 *(Theorem on orientable Gauss codes). Let $\overline{P}$ be an orientable Gauss code. Then the set of lacets for $\overline{P}$ in a orientable surface of genus g are in 1–1 correspondence with the tight solutions of the quadratic system of n^2 equations*

$$\kappa_{ij} + (1 + t_i + t_j)\lambda_{ij} = \sum_{h=1}^{g} (x_{ih} y_{jh} + x_{jh} y_{ih}) \quad \forall (i, j), \qquad (3)$$

where the unknowns are t_i, x_{ih} and y_{ih}, $1 \leqslant i \leqslant n$, $1 \leqslant h \leqslant g$.

Open problems

- How quickly can we recognize Gauss codes of curves in more complicated surfaces? *[Dehn 1936]*
- ...of *null-homologous* curves...?
  - = has a checkerboard coloring = has an Alexander numbering = "*lacet*" *[Lins Richter Shank 1987] [Crapo Rosenstiehl 2001] [Lins, Oliveira-Lima, Silva 2008]*
- ...of *contractible* curves...?
  - = continuously deformable to a point — **Come back tomorrow!**

Open problems

- How quickly can we recognize Gauss codes of curves in more complicated surfaces? *[Dehn 1936]*
- ...of *null-homologous* curves...?
  - = has a checkerboard coloring = has an Alexander numbering = "*lacet*" *[Lins Richter Shank 1987] [Crapo Rosenstiehl 2001] [Lins, Oliveira-Lima, Silva 2008]*
- ...of *contractible* curves...?
  - = continuously deformable to a point — **Come back tomorrow!**
- ...of curves in minimal position...?

Open problems

- How quickly can we recognize Gauss codes of curves in more complicated surfaces? *[Dehn 1936]*

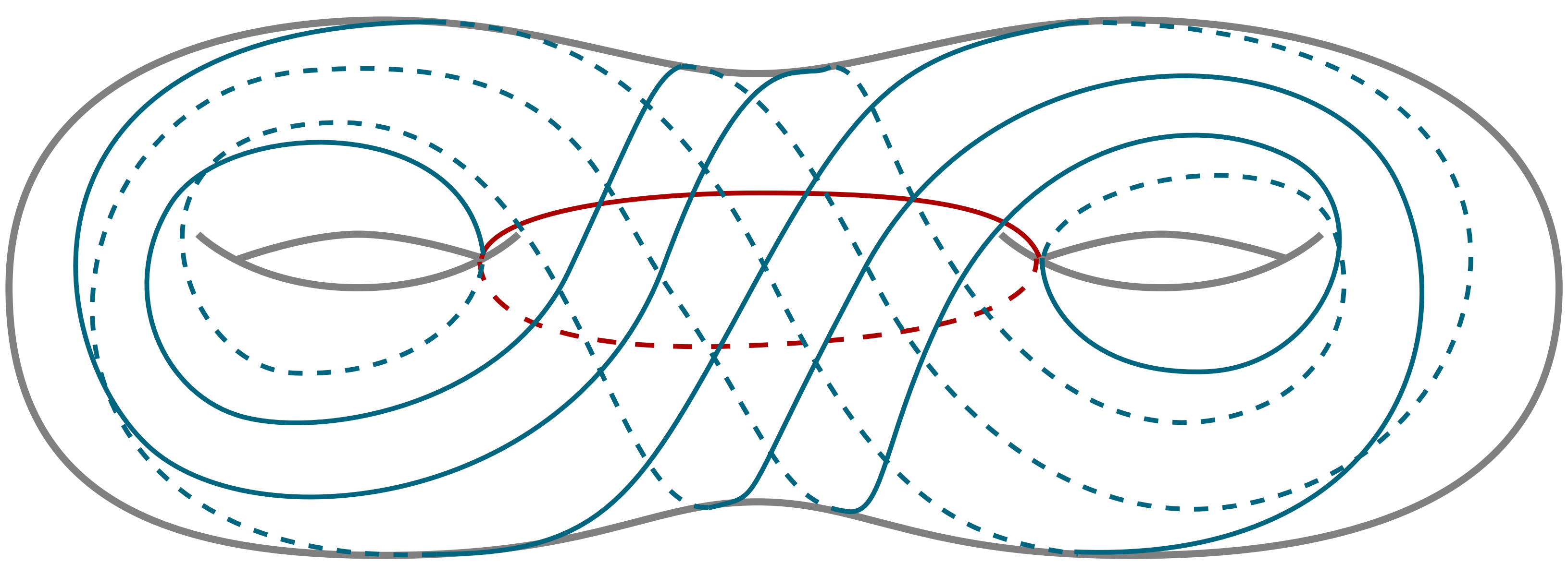
- ...of *null-homologous* curves...?
  - = has a checkerboard coloring = has an Alexander numbering = "*lacet*" *[Lins Richter Shank 1987] [Crapo Rosenstiehl 2001] [Lins, Oliveira-Lima, Silva 2008]*

- ...of *contractible* curves...?
  - = continuously deformable to a point — **Come back tomorrow!**

- ...of curves in minimal position...?
- ...of curves homotopic to simple curves...?

Open problems

- How quickly can we recognize Gauss *paragraphs* of *multicurves* in more complicated surfaces?
- ...of *null-homologous* multicurves...?
  - = has a checkerboard coloring = has an Alexander numbering



g=2, abcdefghijkl·ahcjefgbidkl

[Birman Margalit Menasco 2014]

Open problems

How efficiently can we compute interesting properties of (multi)curves on surfaces from their unsigned Gauss codes?

Open problems

How efficiently can we compute interesting properties of (multi)curves on surfaces from their unsigned Gauss codes?

- **Conjecture:** *All* of these problems are *NP-hard* if the underlying surface is part of the input.

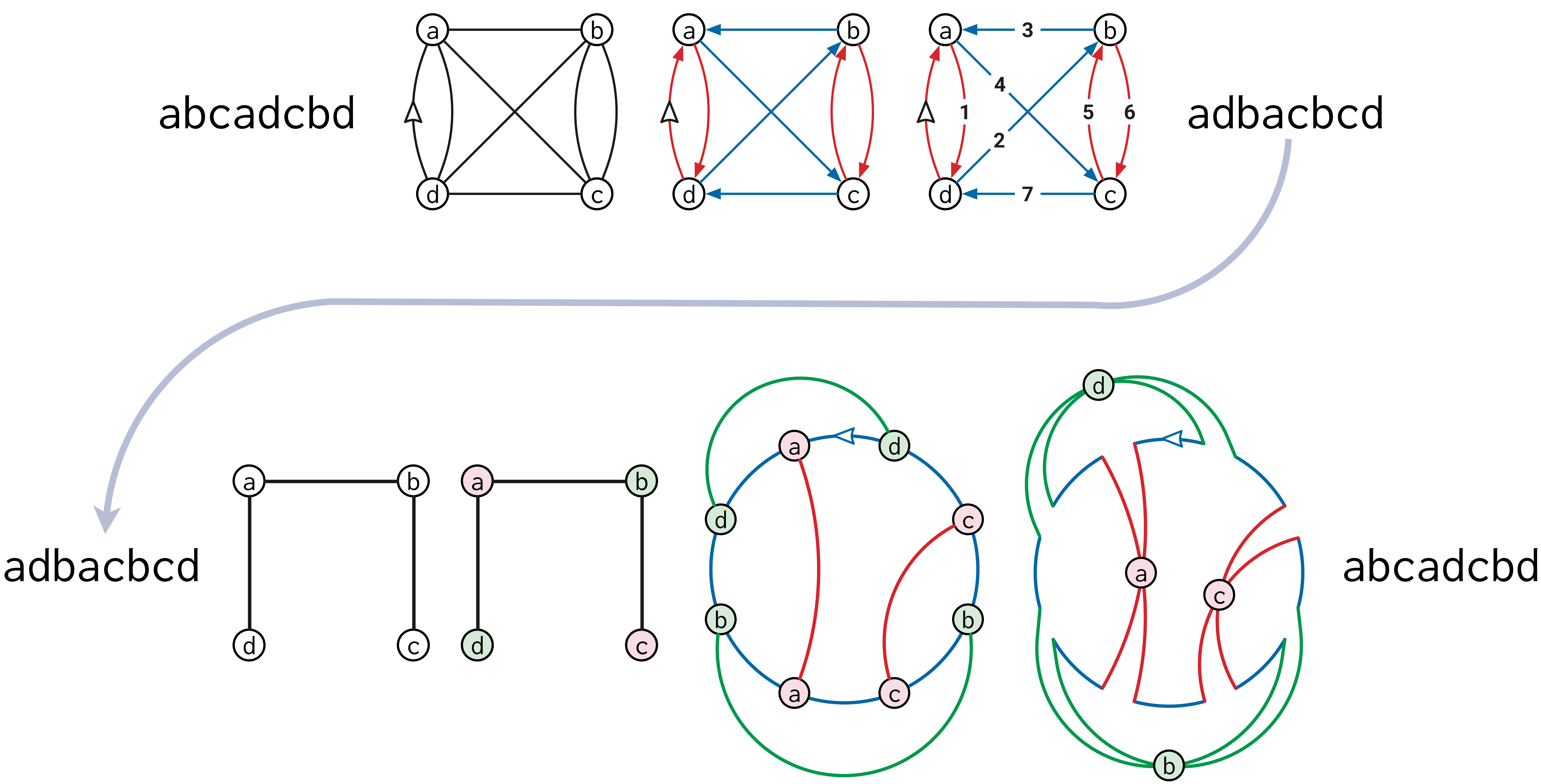
Open problems

How efficiently can we compute interesting properties of (multi)curves on surfaces from their unsigned Gauss codes?

- **Conjecture:** *All* of these problems are *NP-hard* if the underlying surface is part of the input.

- **Conjecture:** *Some* of these problems can be solved in polynomial (or even linear?) time for any *fixed* surface.

Thank you!


abcadcbd
a b c d
a b c d
a b c d
3
4
1
2
5
6
7
adbacbcd
adbacbcd
a b c d
a b c d
a d c b a c b d
d b a c b
abcadcbd