Eliminating Popular Faces
(in curve arrangements)

Maarten Löffler
Utrecht University / Tulane University

Joint work with

Phoebe de Nooijer
Alexandra Weinberger
Soeren Terziadis
Zuzana Masárová
Tamara Mchedlidze
Günter Rote

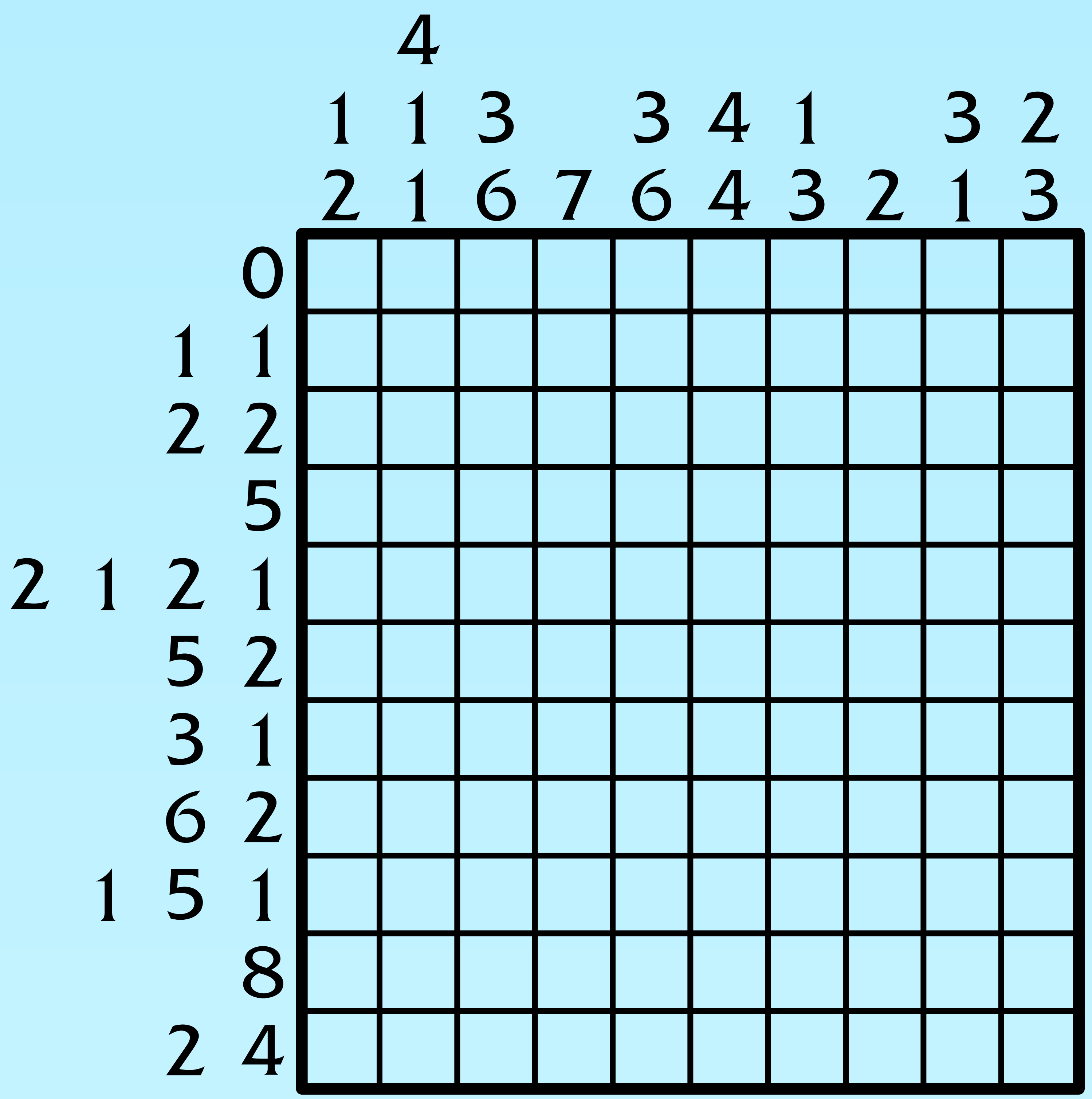
Part 1
NONOGRAMS

Part 2
POPULAR FACES

Part 3
FPT

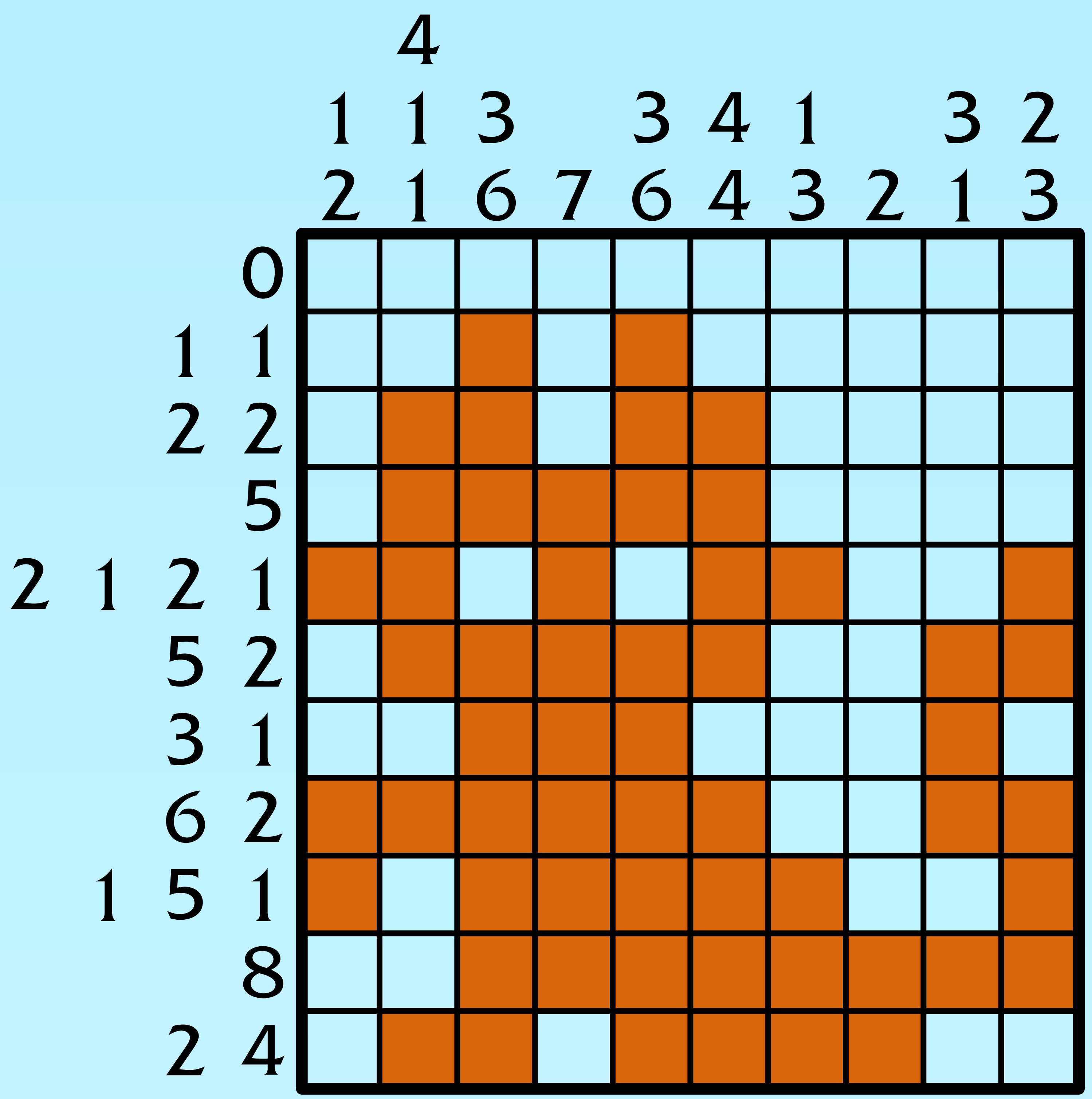
Part 1
NONOGRAMS

Let's talk about Nonograms (a.k.a. Griddlers, Japanese puzzles, Hanjie, Picross, ...)



Let's talk about Nonograms (a.k.a. Griddlers, Japanese puzzles, Hanjie, Picross, ...)

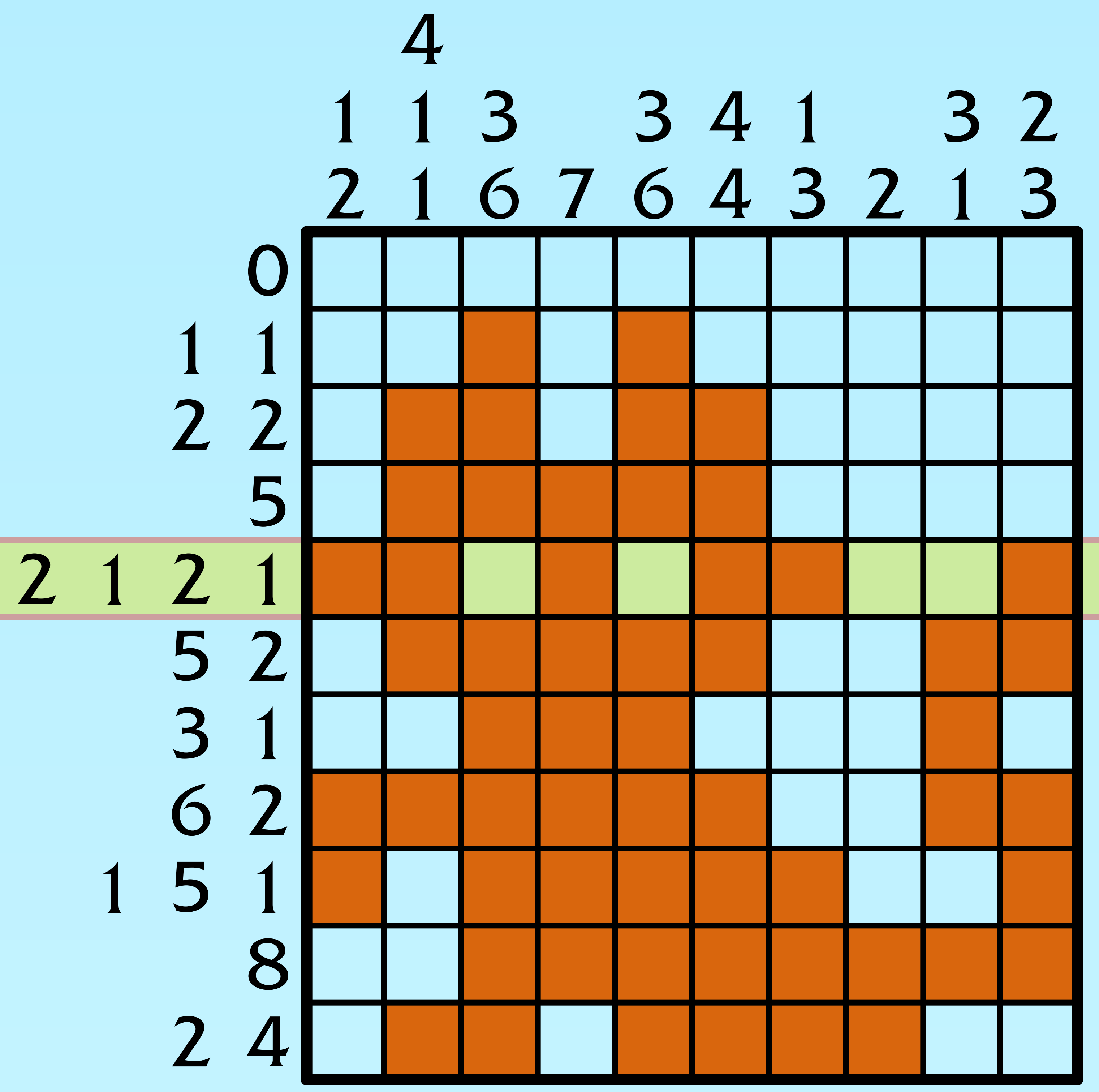
Input: a grid with some numbers next to and above it.



Let's talk about Nonograms (a.k.a. Griddlers, Japanese puzzles, Hanjie, Picross, ...)

Input: a grid with some numbers next to and above it.

Output: a black/white coloring of the grid cells.



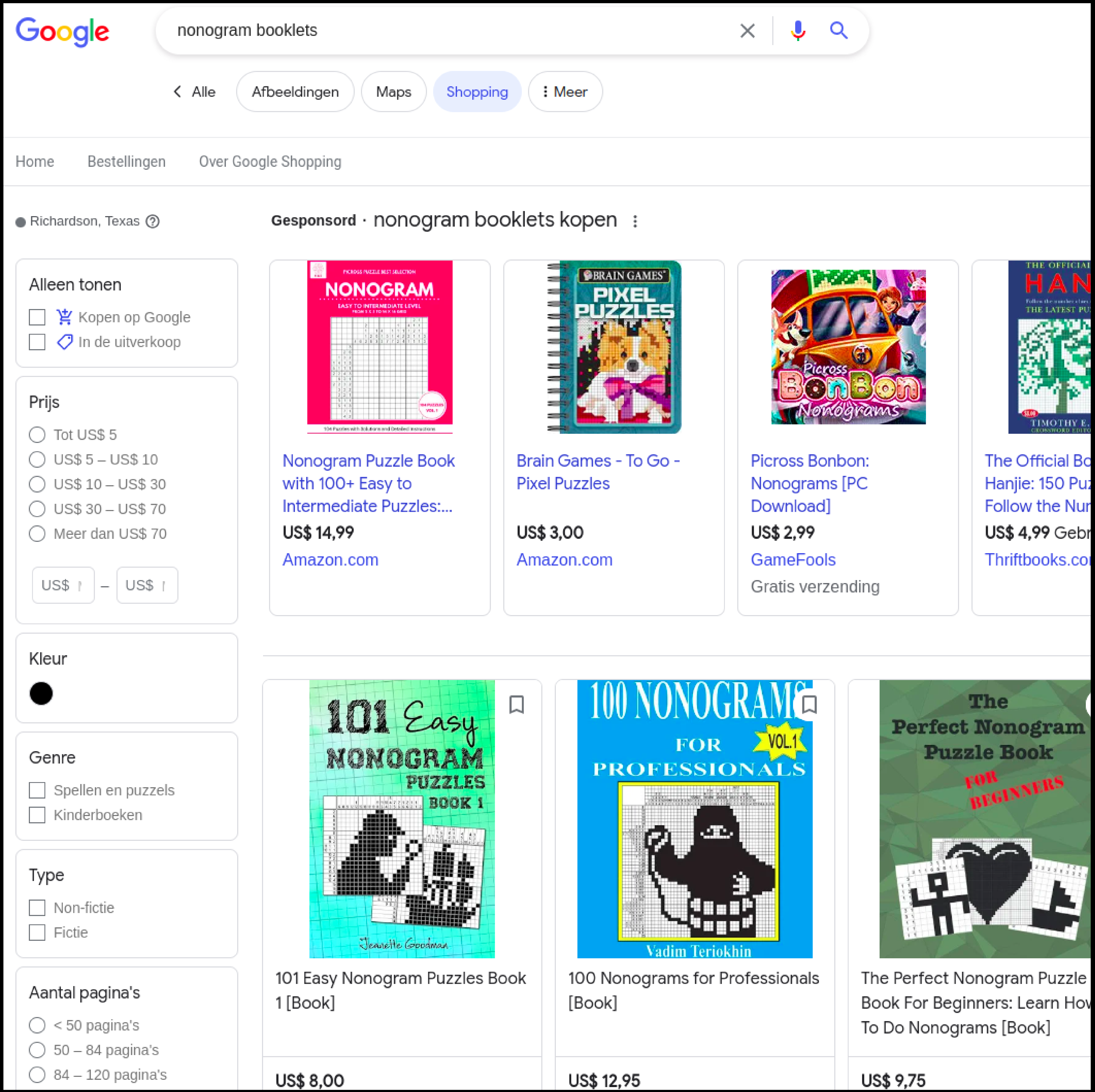
Let's talk about Nonograms (a.k.a. Griddlers, Japanese puzzles, Hanjie, Picross, ...)

Input: a grid with some numbers next to and above it.

Output: a black/white coloring of the grid cells.

Rule: in each row and column, the numbers correspond to the sizes of the consecutive blocks of black cells.

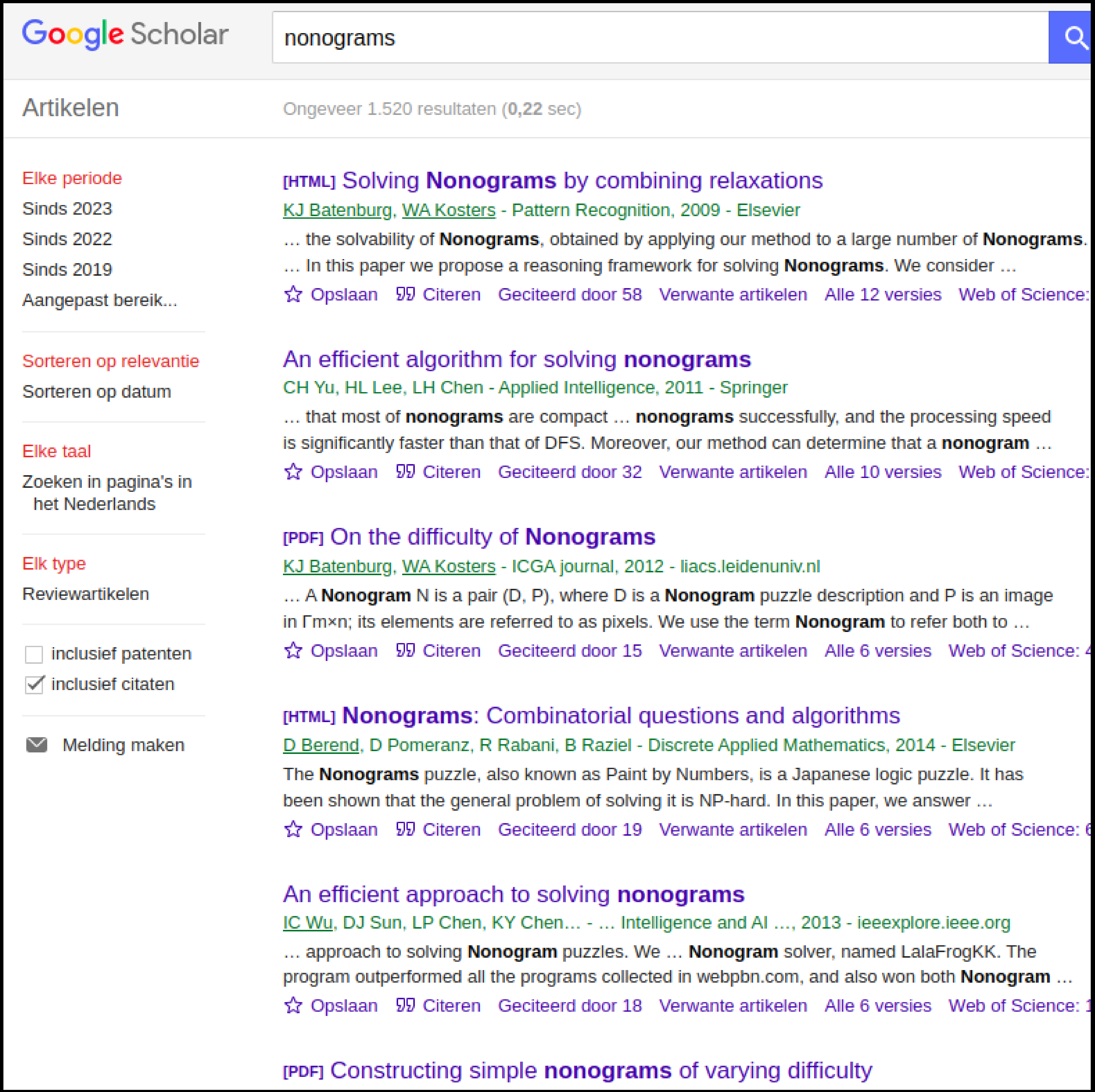
Let's talk about Nonograms (a.k.a. Griddlers, Japanese puzzles, Hanjie, Picross, ...)



Let's talk about Nonograms (a.k.a. Griddlers, Japanese puzzles, Hanjie, Picross, ...)

Very popular with consumers

...



Google Scholar | nonograms

Artikelen

Ongeveer 1.520 resultaten (0,22 sec)

Elke periode
Sinds 2023
Sinds 2022
Sinds 2019
Aangepast bereik...

Sorteren op relevantie
Sorteren op datum

Elke taal
Zoeken in pagina's in het Nederlands

Elk type
Reviewartikelen

inclusief patenten
inclusief citaten

Melding maken

[HTML] Solving **Nonograms** by combining relaxations
KJ Batenburg, WA Kosters - Pattern Recognition, 2009 - Elsevier
… the solvability of **Nonograms**, obtained by applying our method to a large number of **Nonograms**. … In this paper we propose a reasoning framework for solving **Nonograms**. We consider …
Opslaan Citeren Geciteerd door 58 Verwante artikelen Alle 12 versies Web of Science:

An efficient algorithm for solving **nonograms**
CH Yu, HL Lee, LH Chen - Applied Intelligence, 2011 - Springer
… that most of **nonograms** are compact … **nonograms** successfully, and the processing speed is significantly faster than that of DFS. Moreover, our method can determine that a **nonogram** …
Opslaan Citeren Geciteerd door 32 Verwante artikelen Alle 10 versies Web of Science:

[PDF] On the difficulty of **Nonograms**
KJ Batenburg, WA Kosters - ICGA journal, 2012 - liacs.leidenuniv.nl
… A **Nonogram** N is a pair (D, P), where D is a **Nonogram** puzzle description and P is an image in Γm×n; its elements are referred to as pixels. We use the term **Nonogram** to refer both to …
Opslaan Citeren Geciteerd door 15 Verwante artikelen Alle 6 versies Web of Science: 4

[HTML] **Nonograms**: Combinatorial questions and algorithms
D Berend, D Pomeranz, R Rabani, B Raziel - Discrete Applied Mathematics, 2014 - Elsevier
The **Nonograms** puzzle, also known as Paint by Numbers, is a Japanese logic puzzle. It has been shown that the general problem of solving it is NP-hard. In this paper, we answer …
Opslaan Citeren Geciteerd door 19 Verwante artikelen Alle 6 versies Web of Science: 6

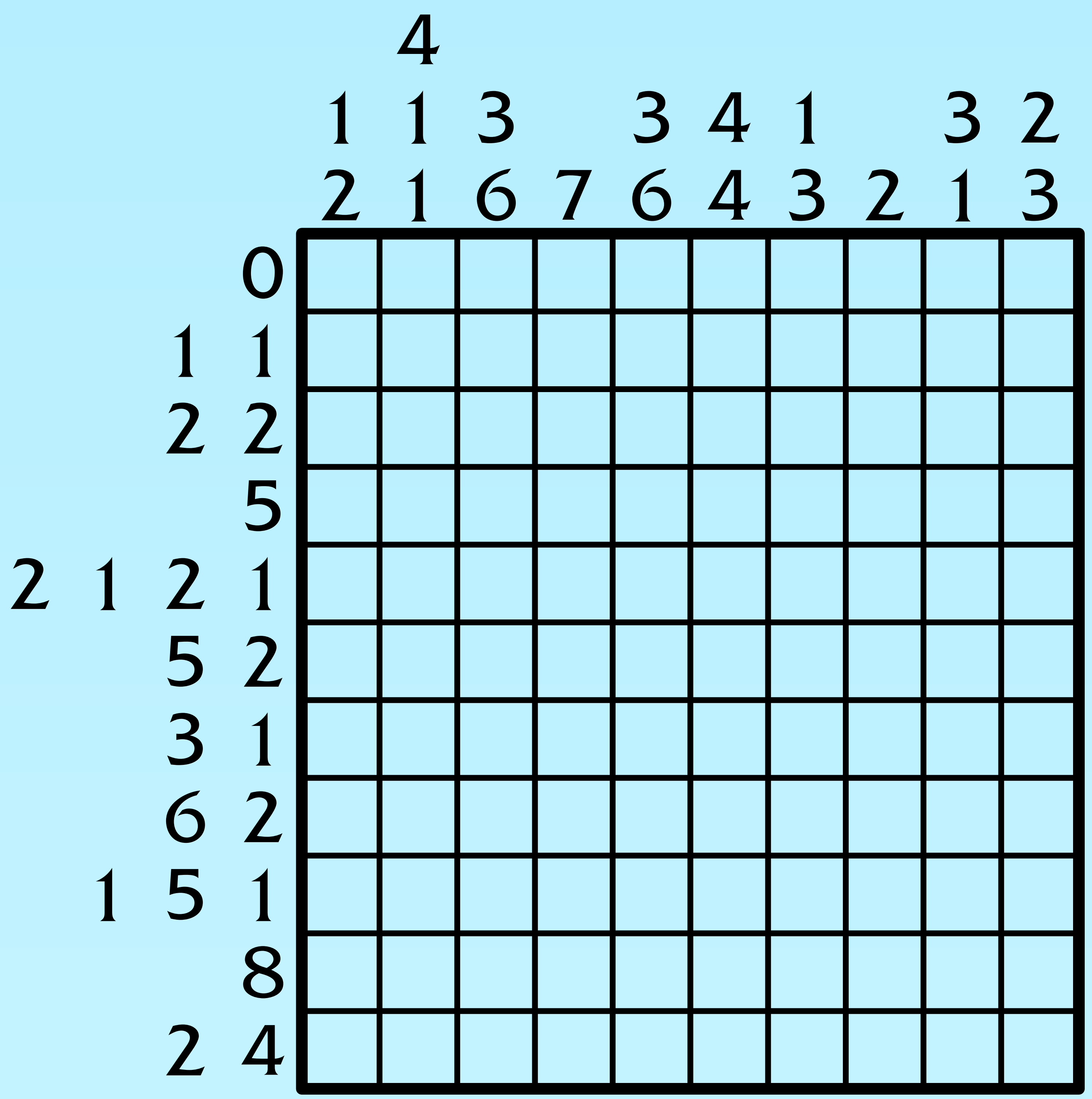
An efficient approach to solving **nonograms**
IC Wu, DJ Sun, LP Chen, KY Chen… - … Intelligence and AI …, 2013 - ieeexplore.ieee.org
… approach to solving **Nonogram** puzzles. We … **Nonogram** solver, named LalaFrogKK. The program outperformed all the programs collected in webpbn.com, and also won both **Nonogram** …
Opslaan Citeren Geciteerd door 18 Verwante artikelen Alle 6 versies Web of Science: 1

[PDF] Constructing simple **nonograms** of varying difficulty

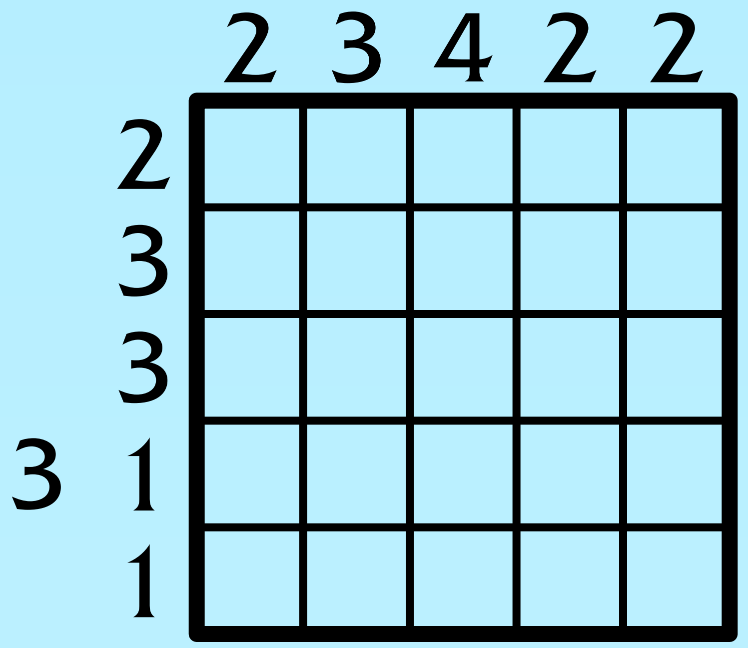
Let's talk about Nonograms (a.k.a. Griddlers, Japanese puzzles, Hanjie, Picross, …)

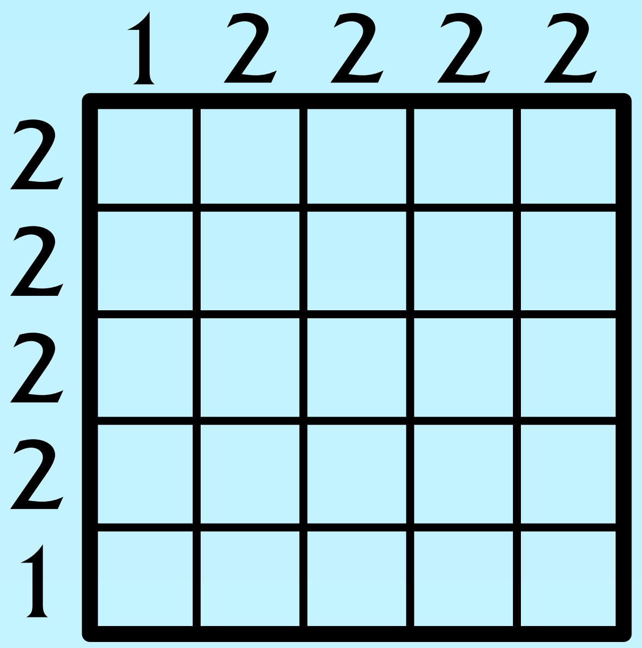
Very popular with consumers …

… and scientists.

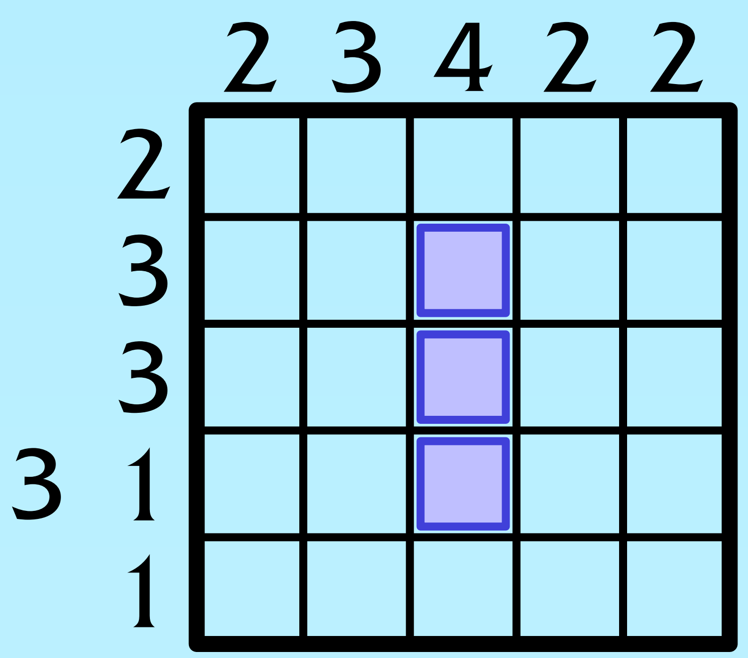


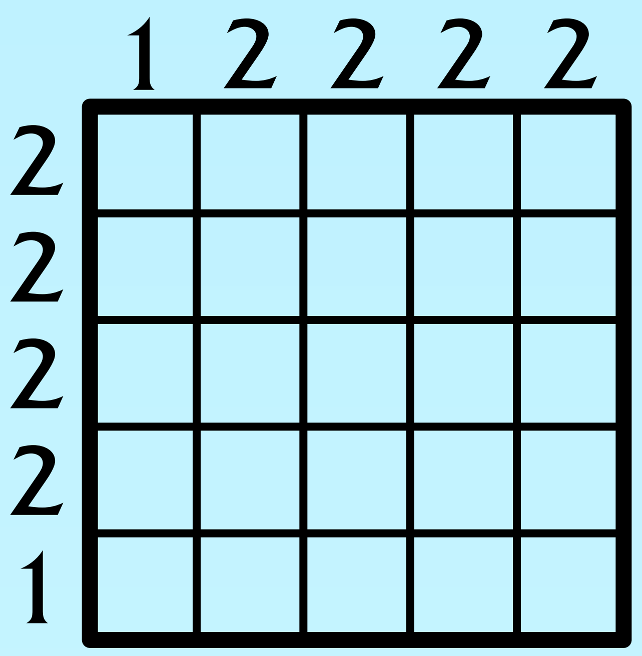
A nonogram is simple if it can be solved by only ever looking at one row or column at a time.



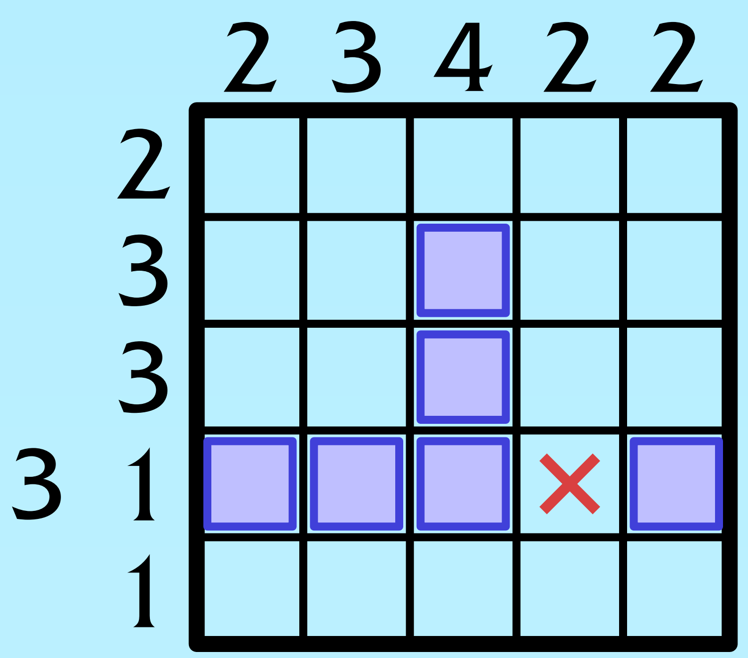


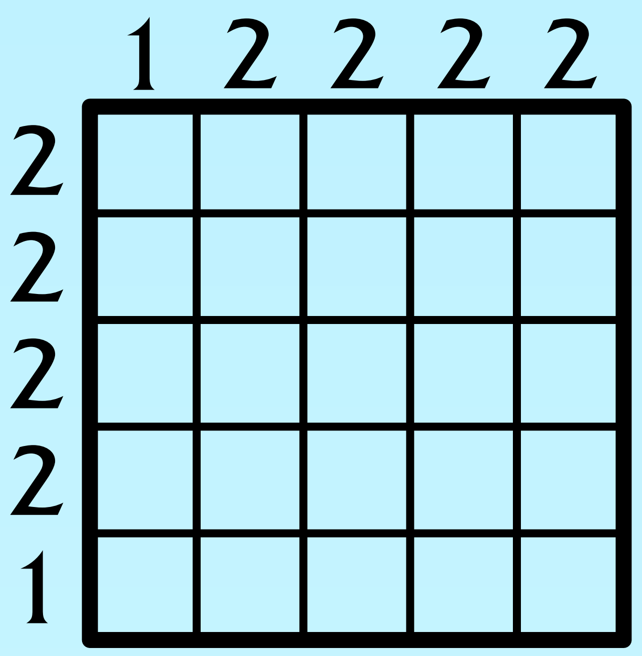
A nonogram is simple if it can be solved by only ever looking at one row or column at a time.



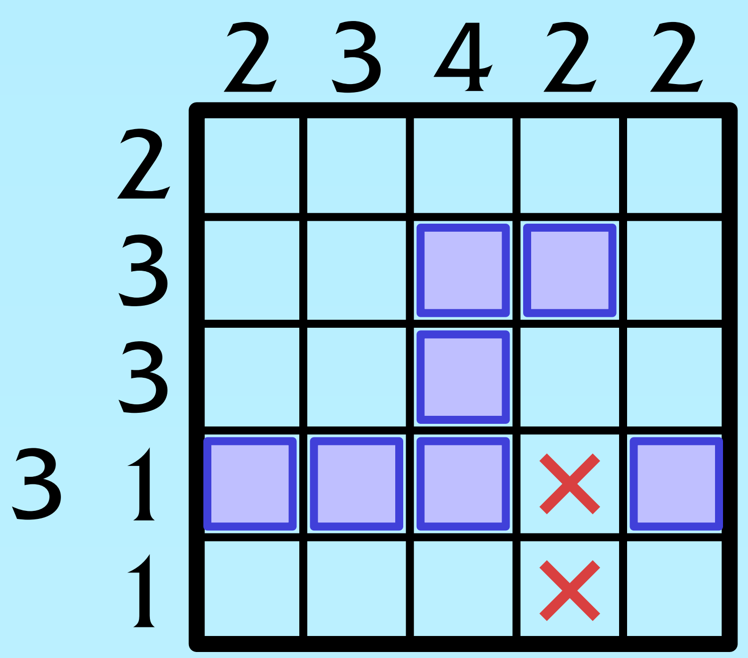


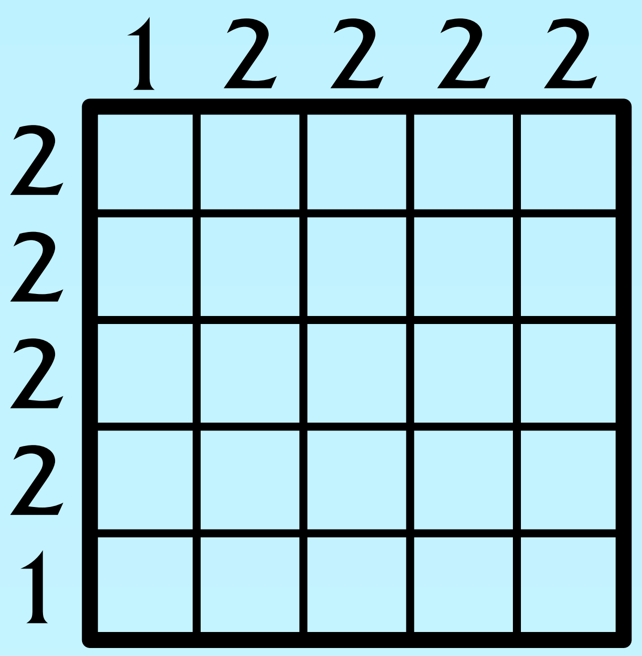
A nonogram is simple if it can be solved by only ever looking at one row or column at a time.



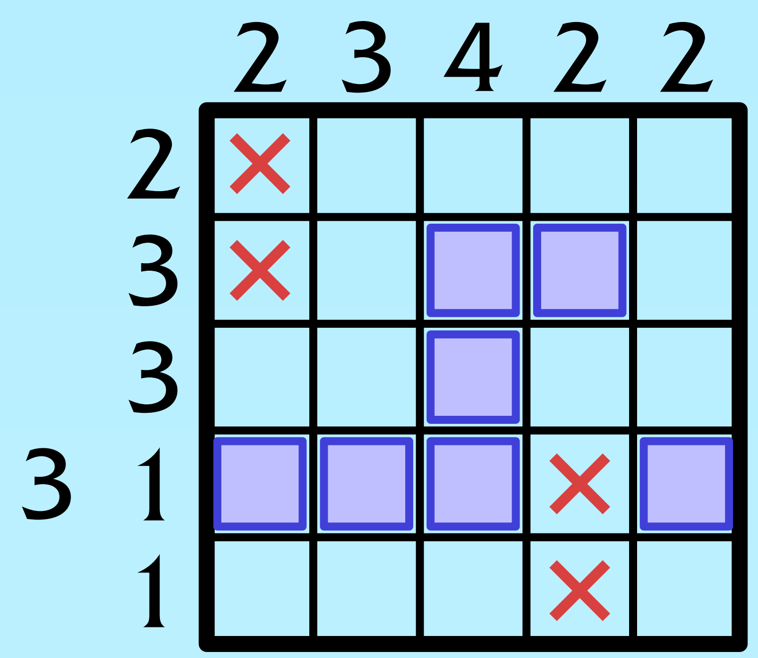


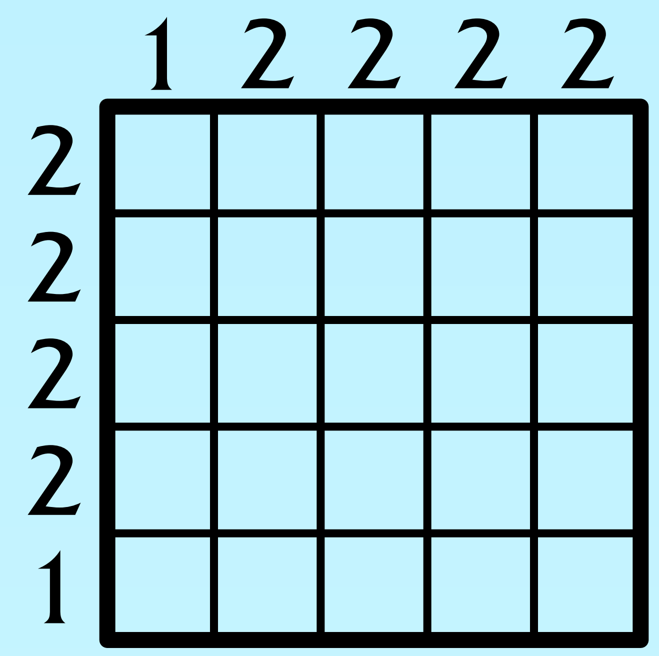
A nonogram is simple if it can be solved by only ever looking at one row or column at a time.



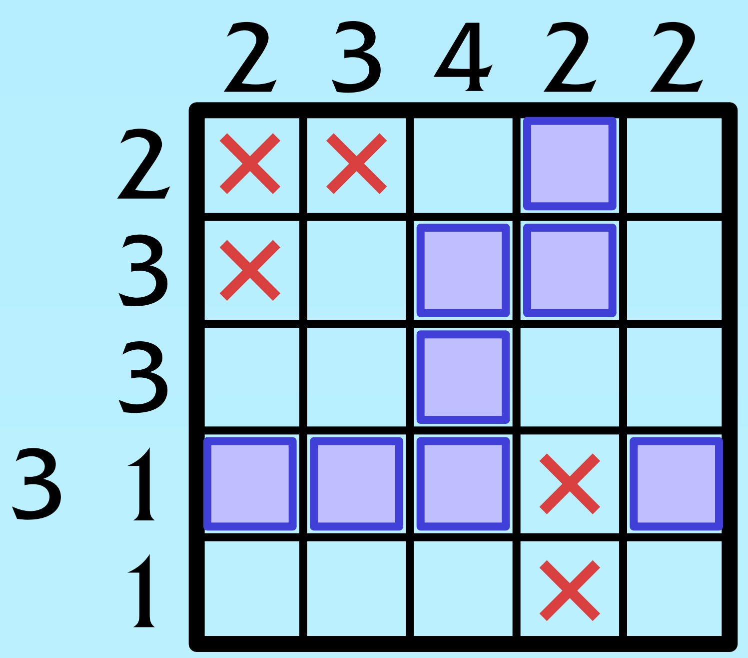


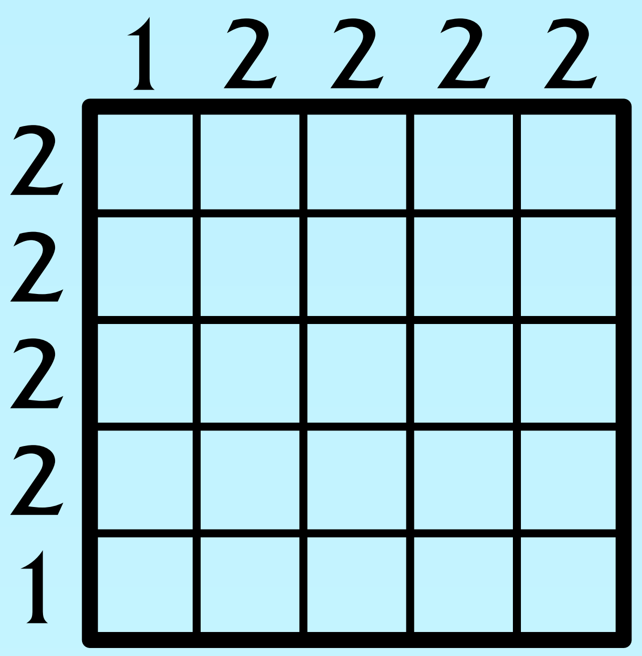
A nonogram is simple if it can be solved by only ever looking at one row or column at a time.



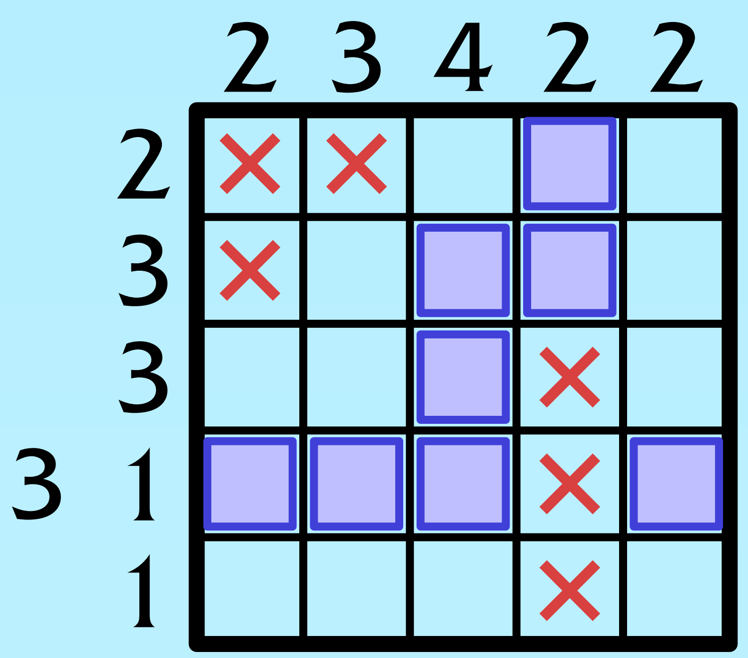


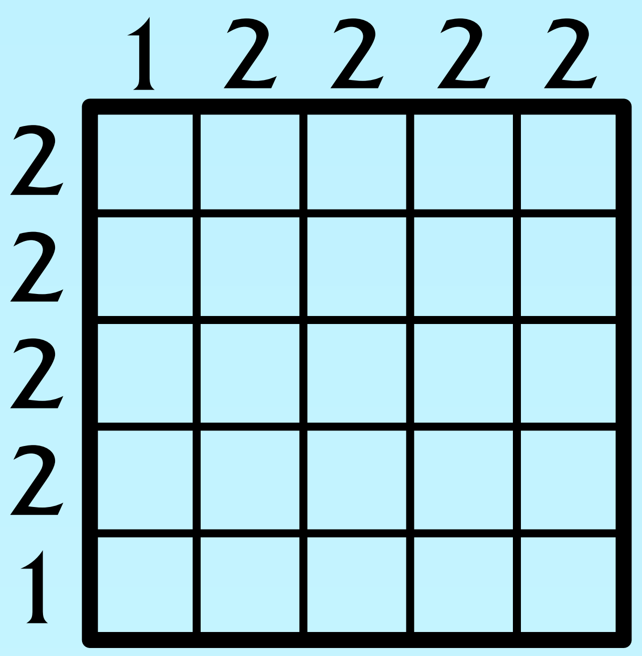
A nonogram is simple if it can be solved by only ever looking at one row or column at a time.



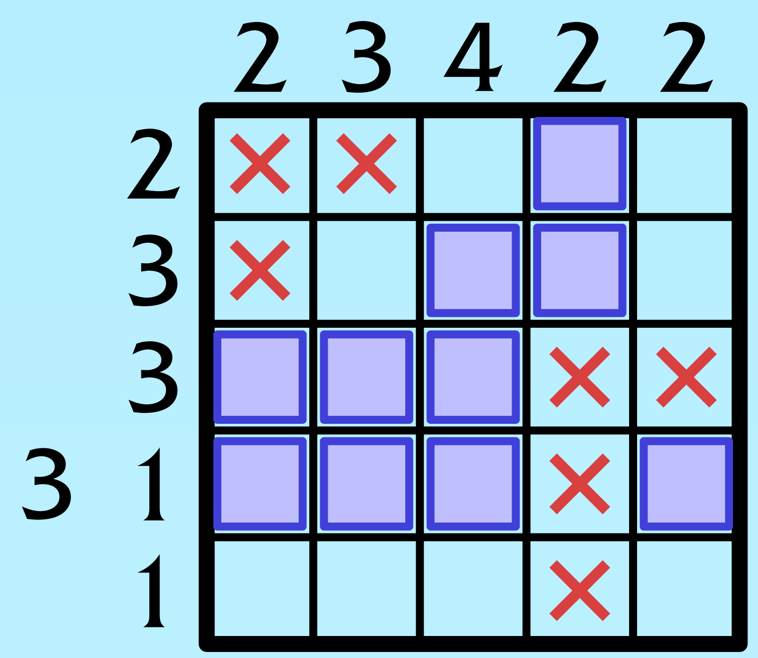


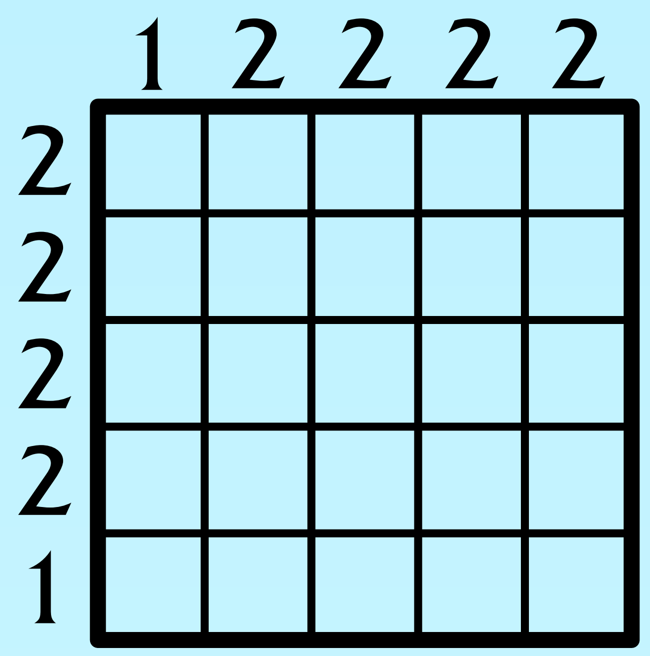
A nonogram is simple if it can be solved by only ever looking at one row or column at a time.



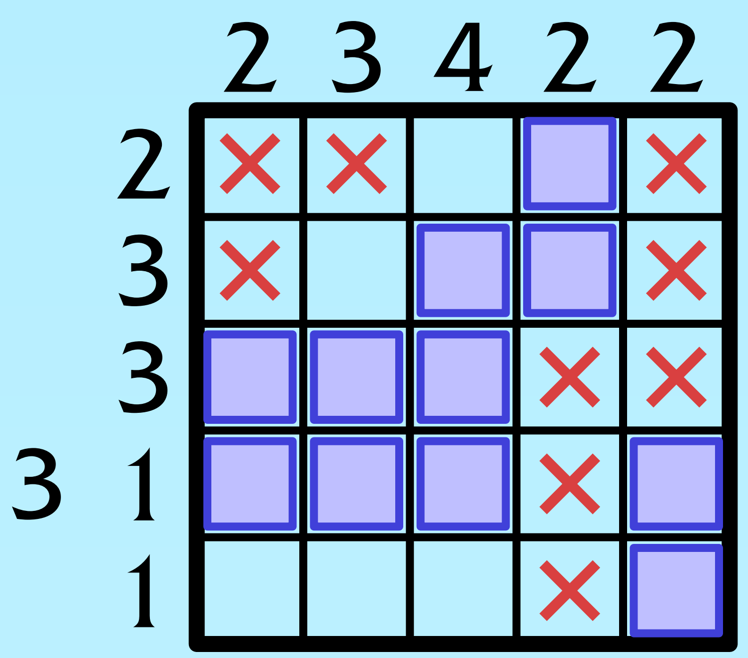


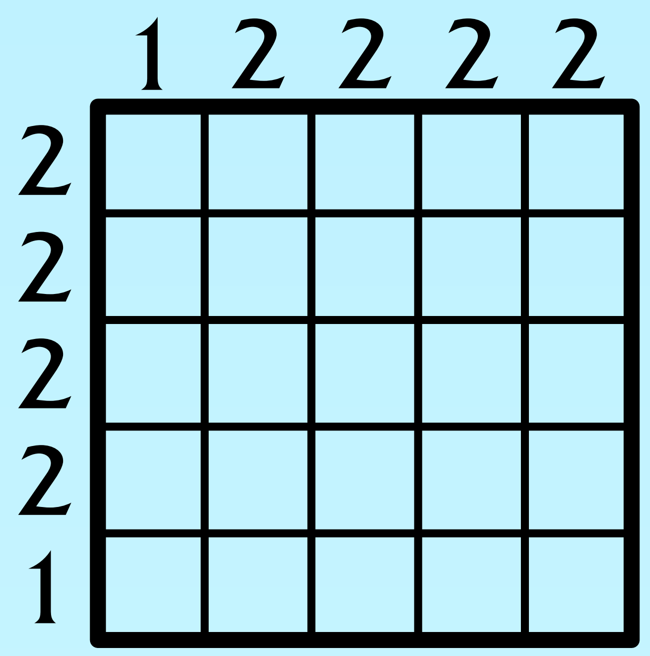
A nonogram is simple if it can be solved by only ever looking at one row or column at a time.



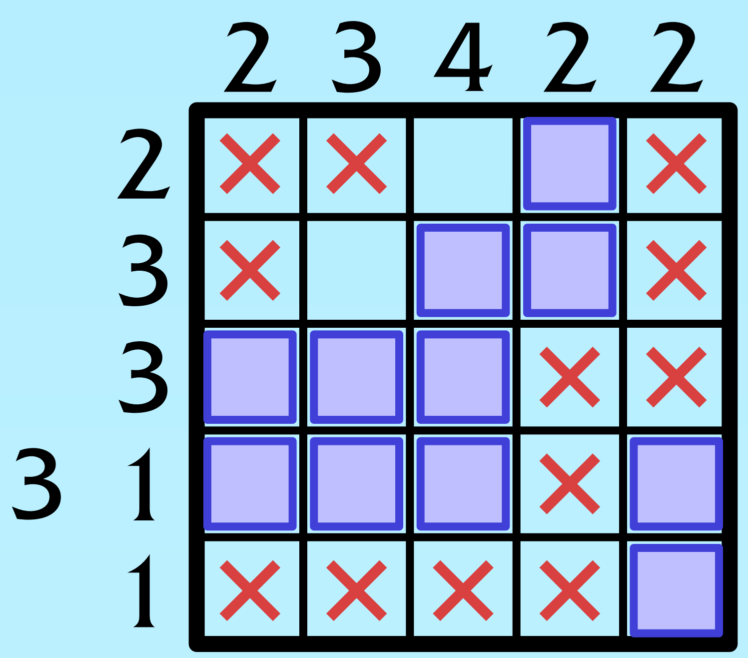


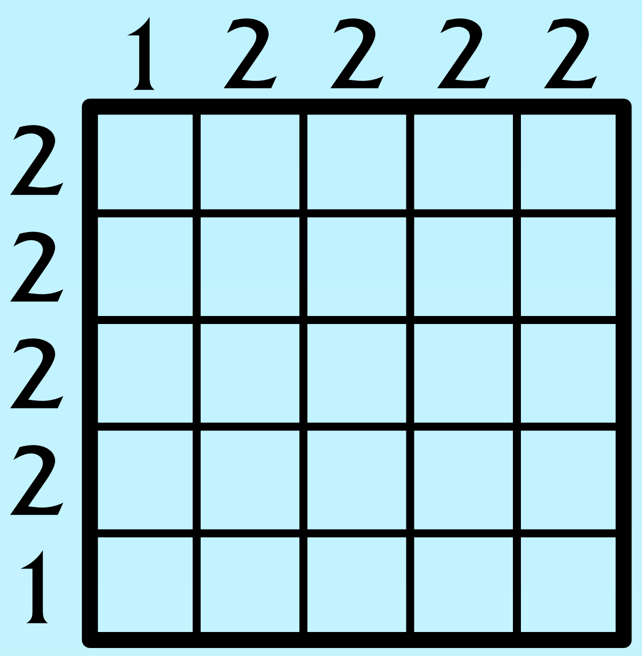
A nonogram is simple if it can be solved by only ever looking at one row or column at a time.



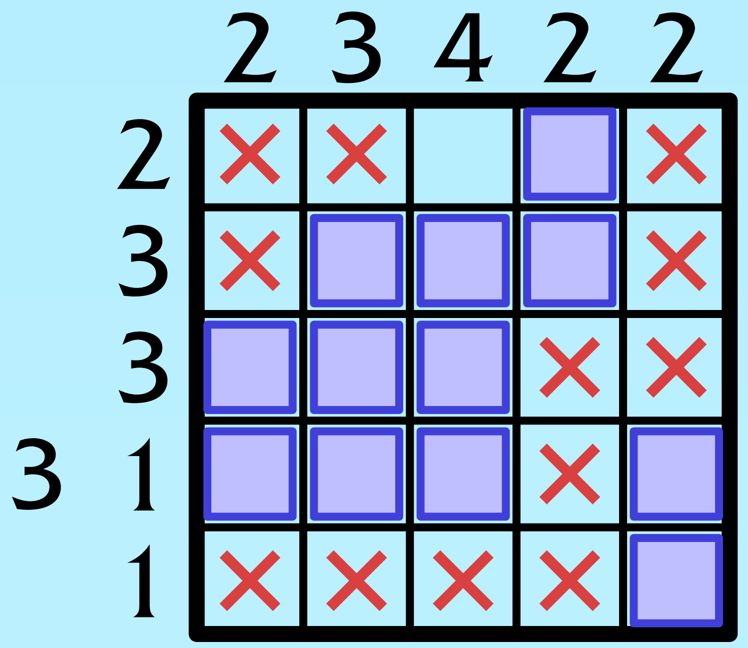


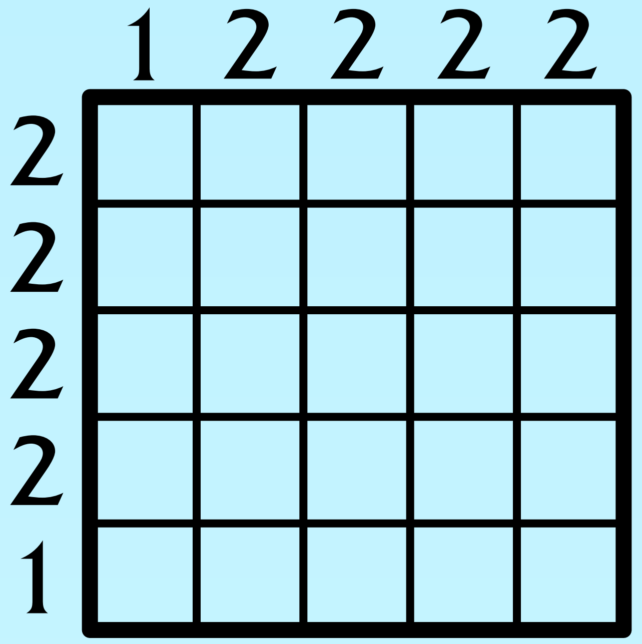
A nonogram is simple if it can be solved by only ever looking at one row or column at a time.



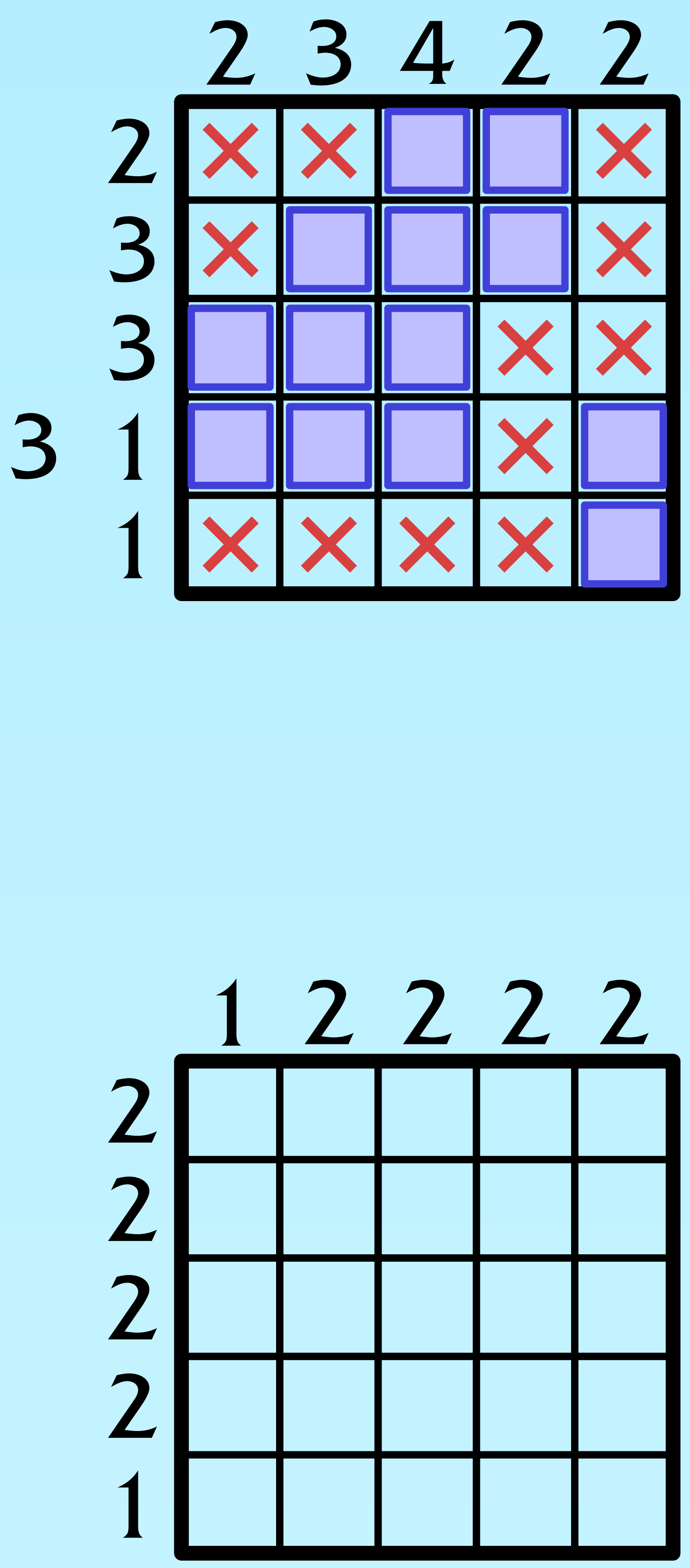


A nonogram is simple if it can be solved by only ever looking at one row or column at a time.

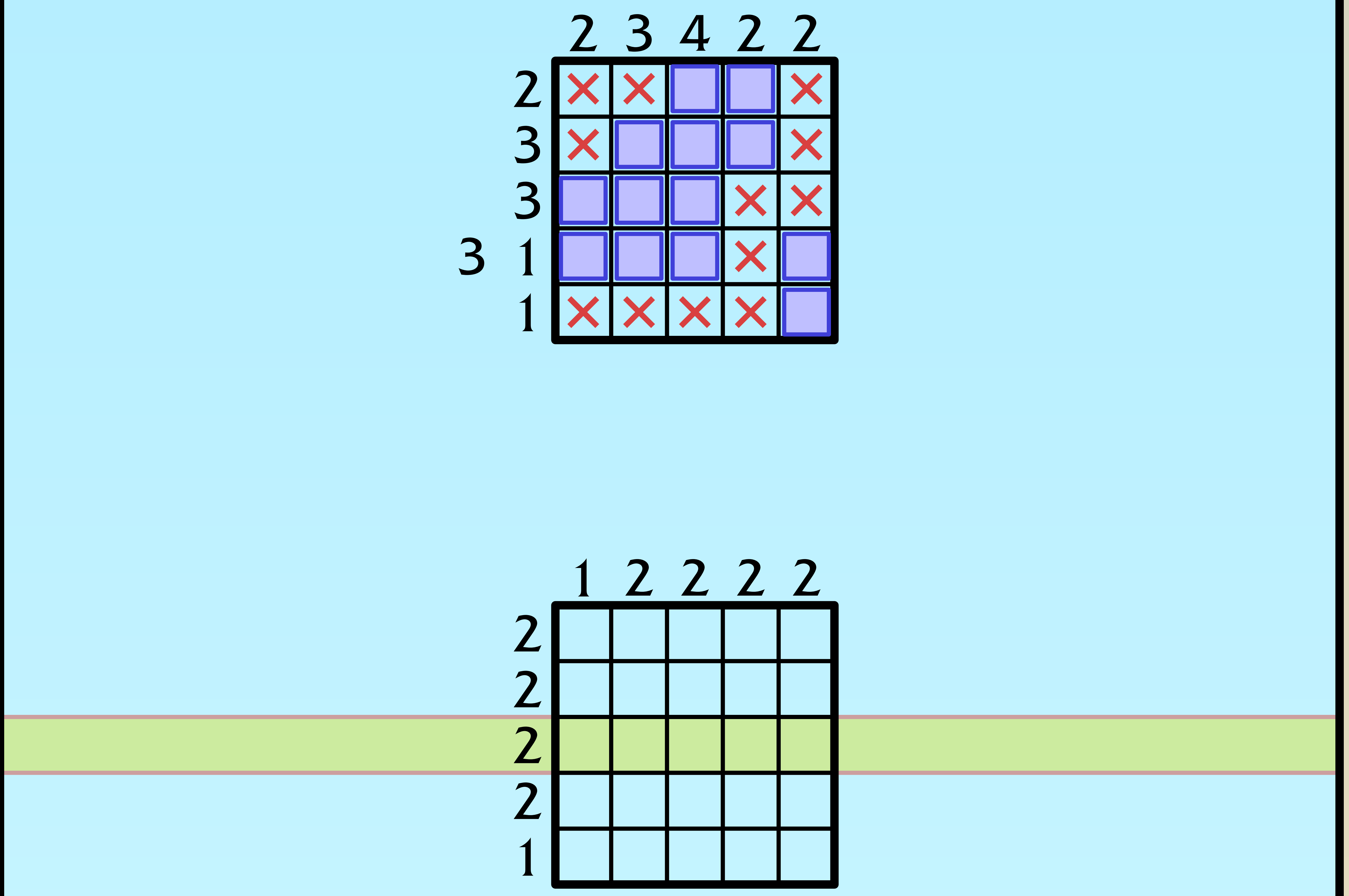




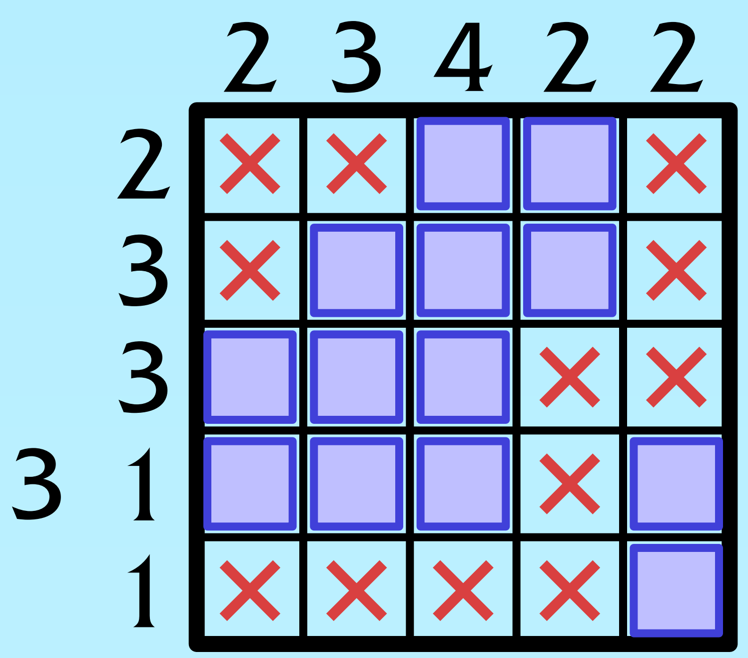
A nonogram is simple if it can be solved by only ever looking at one row or column at a time.

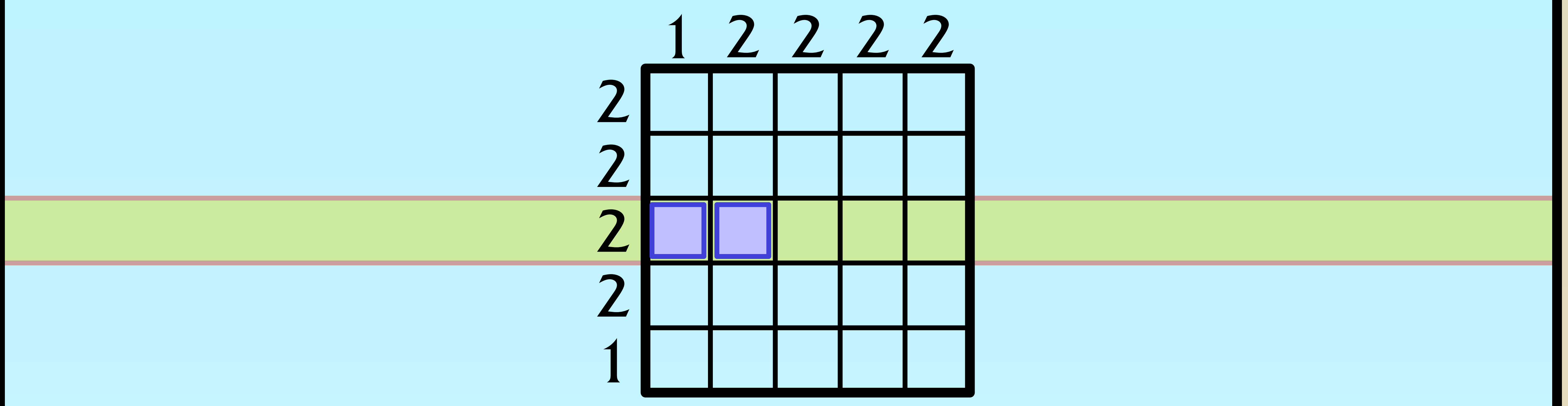


A nonogram is simple if it can be solved by only ever looking at one row or column at a time.

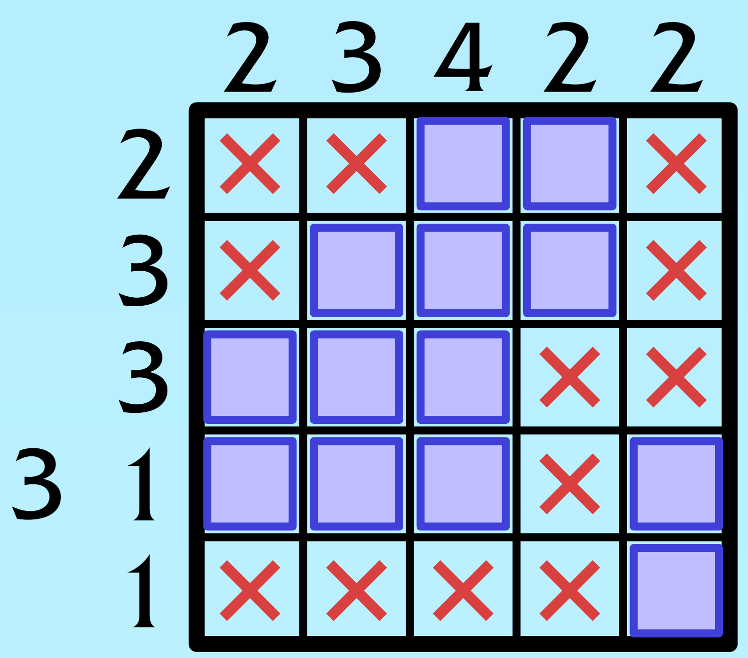


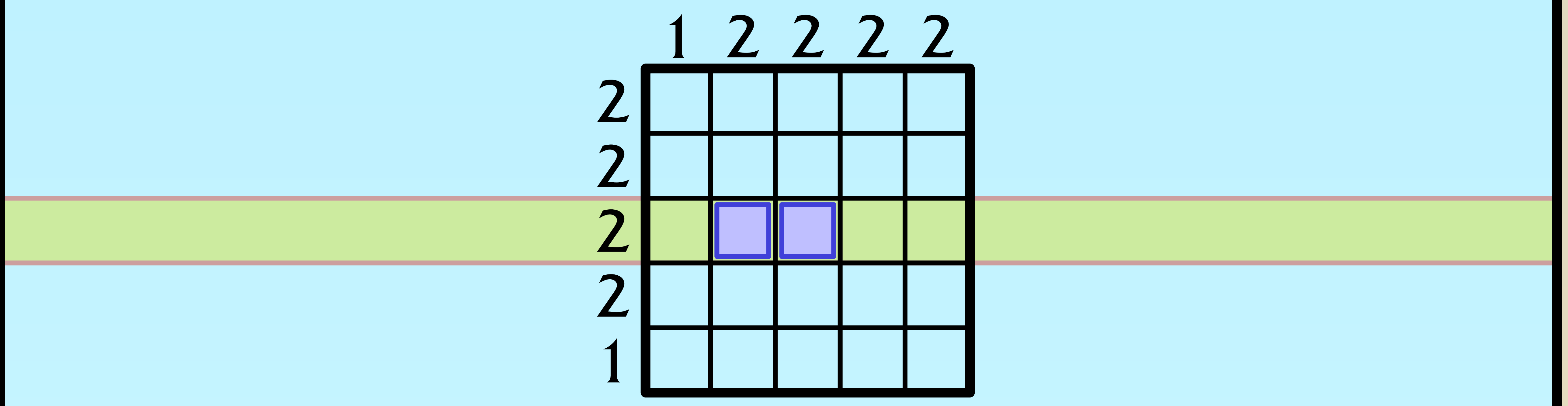
A nonogram is simple if it can be solved by only ever looking at one row or column at a time.



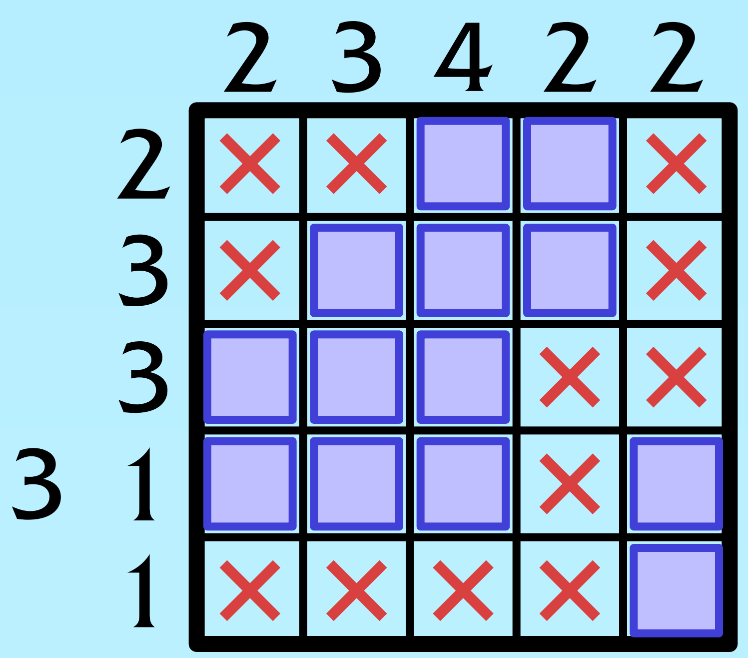


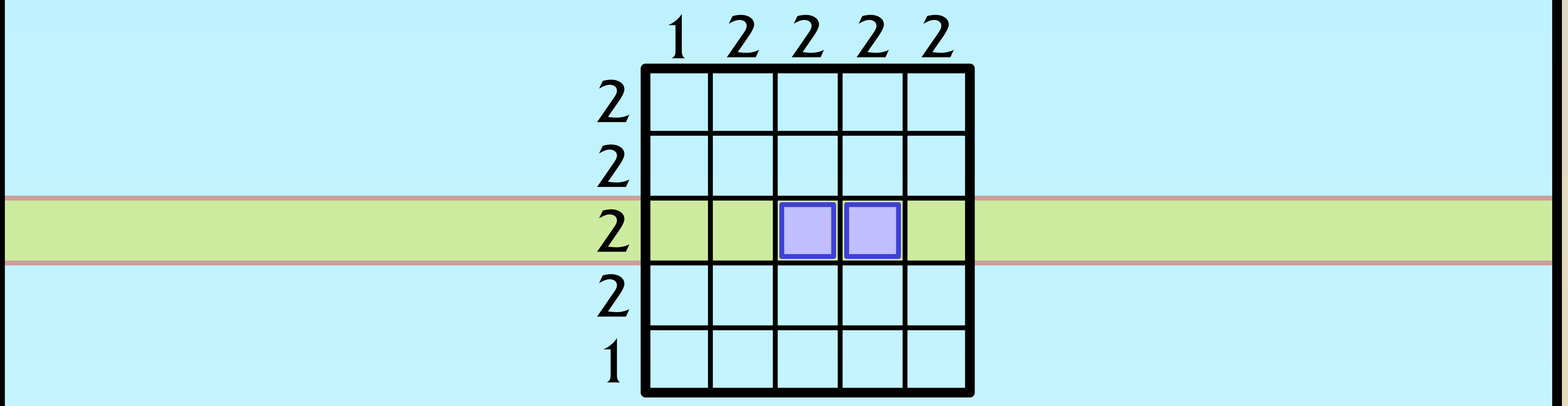
A nonogram is simple if it can be solved by only ever looking at one row or column at a time.



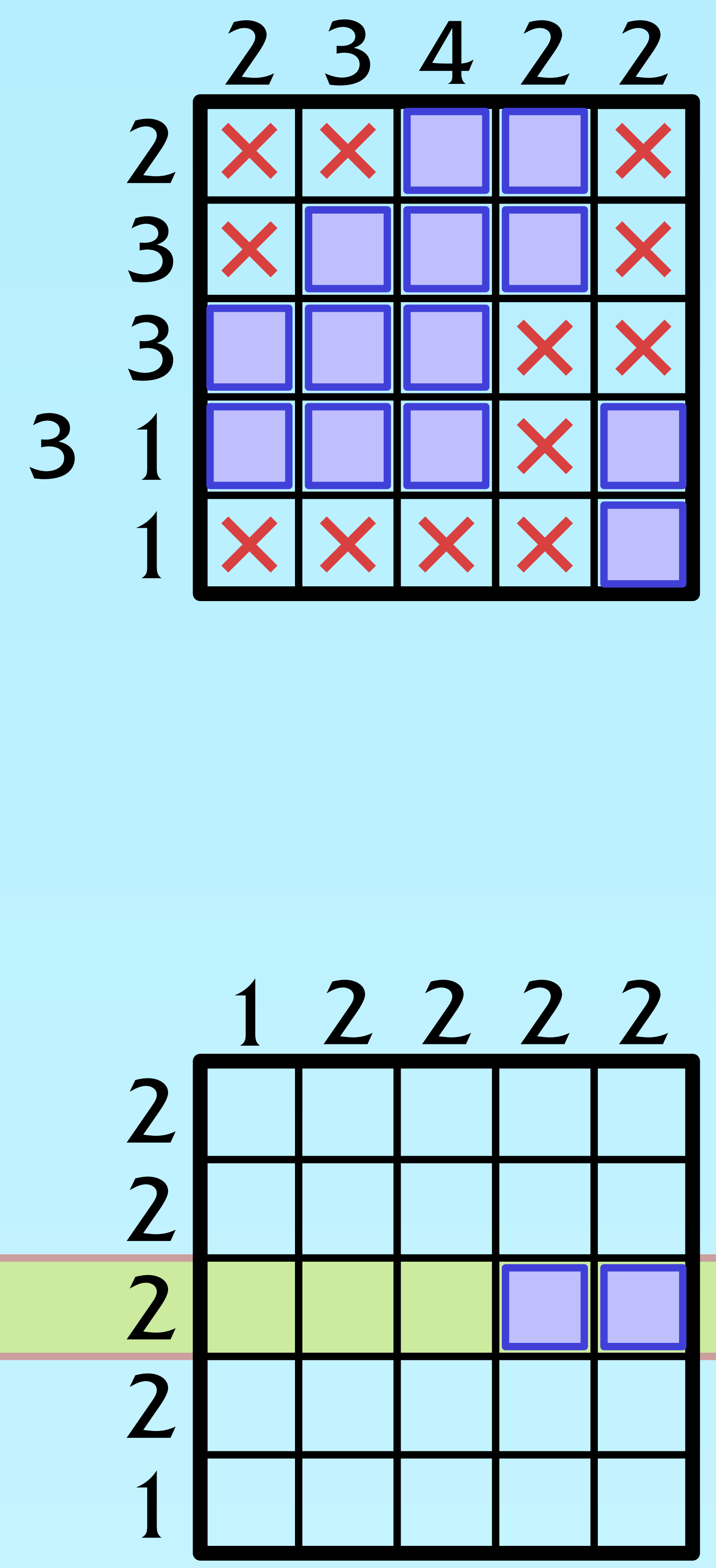


A nonogram is simple if it can be solved by only ever looking at one row or column at a time.

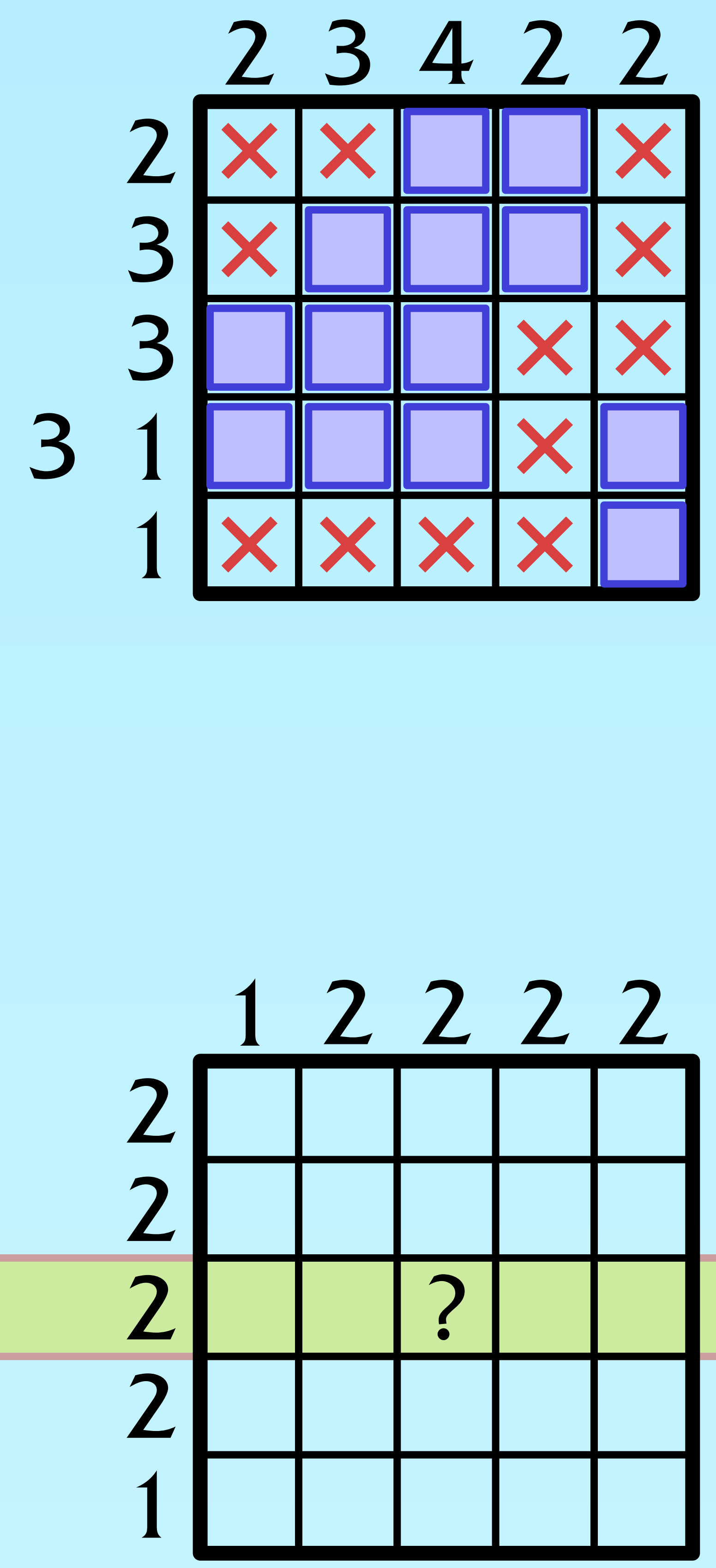




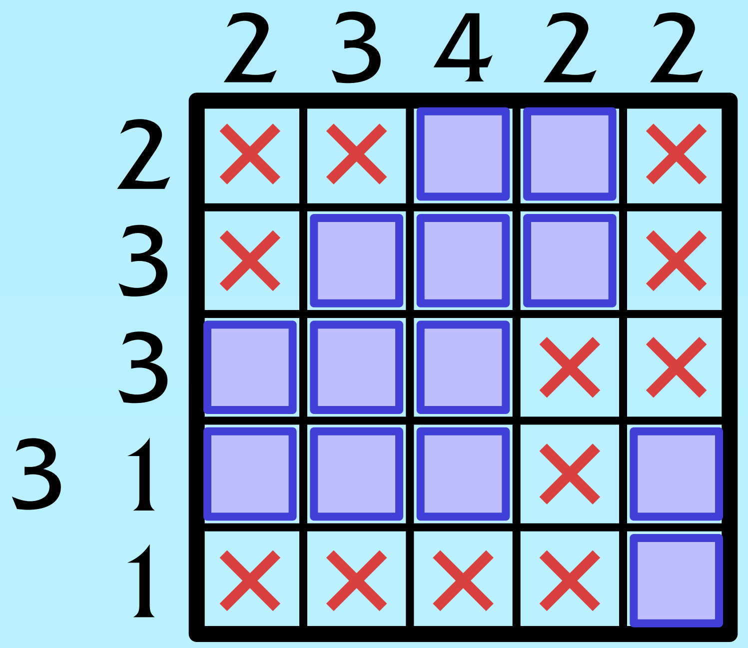
A nonogram is simple if it can be solved by only ever looking at one row or column at a time.

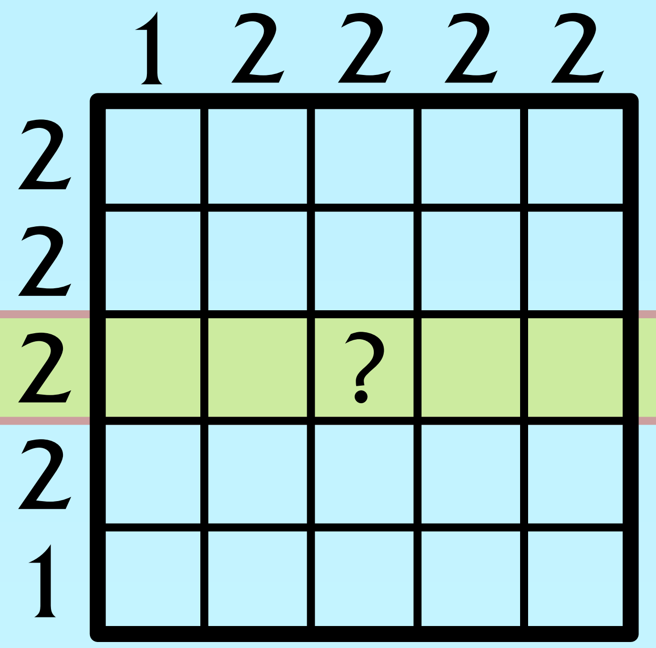


A nonogram is simple if it can be solved by only ever looking at one row or column at a time.



A nonogram is simple if it can be solved by only ever looking at one row or column at a time.

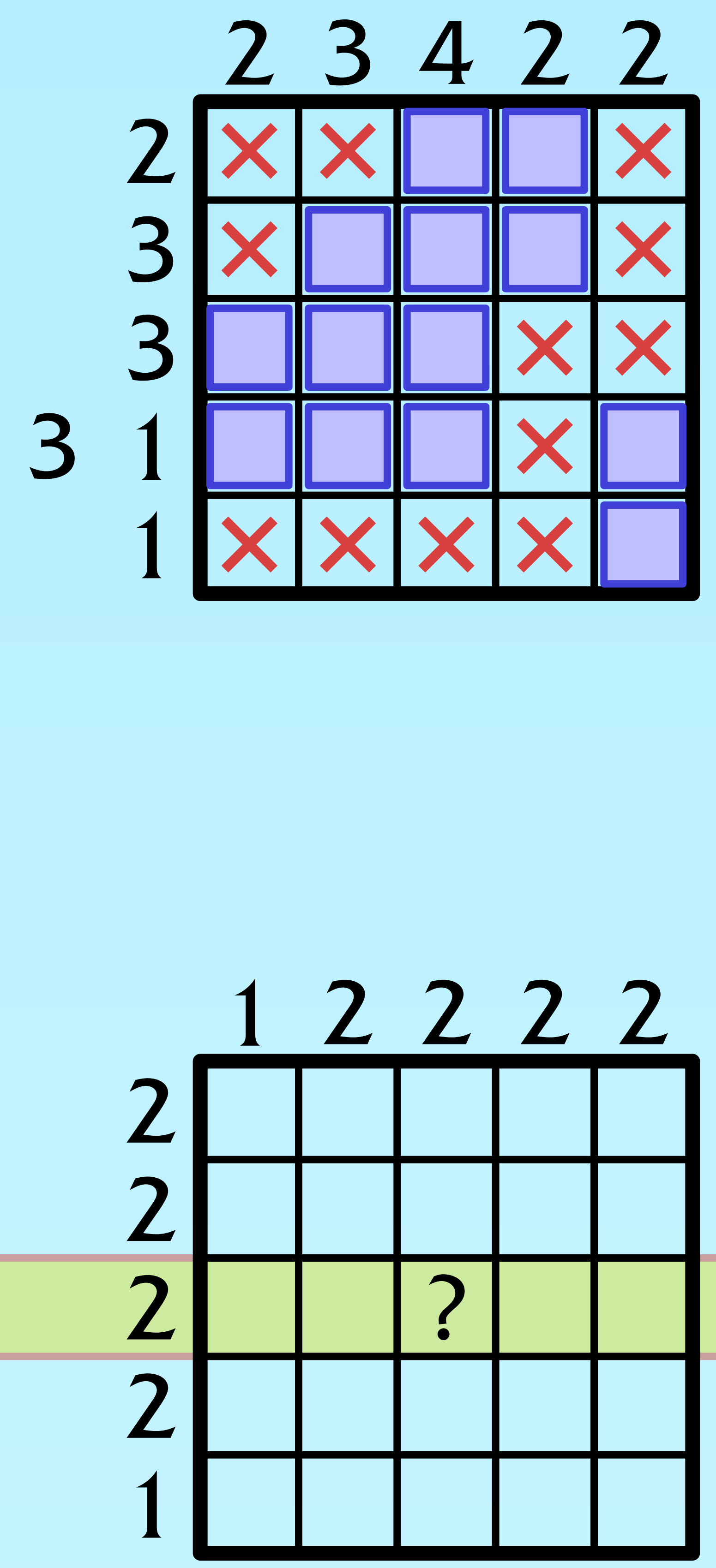




A nonogram is simple if it can be solved by only ever looking at one row or column at a time.

Solving nonograms is NP-hard, but solving simple nonograms is not.

[Batenburg & Kosters, ICGA Journal, 2012]

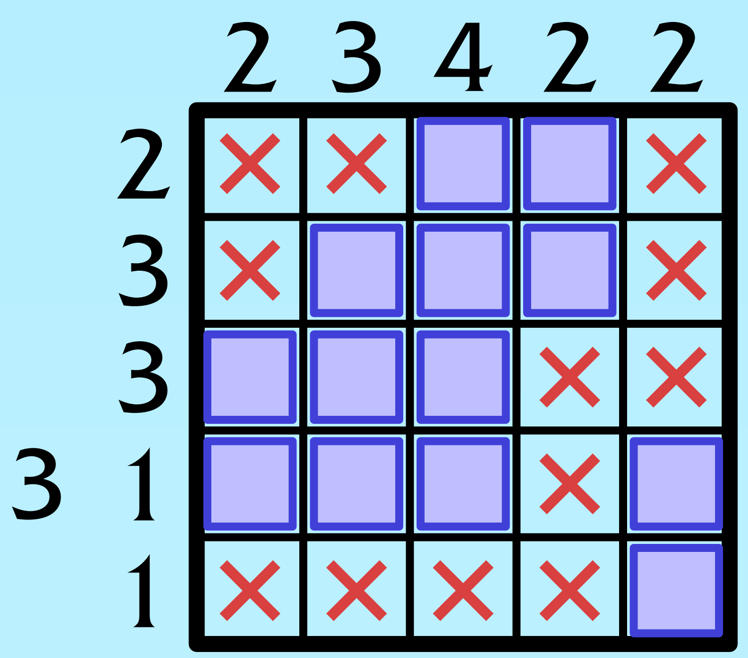


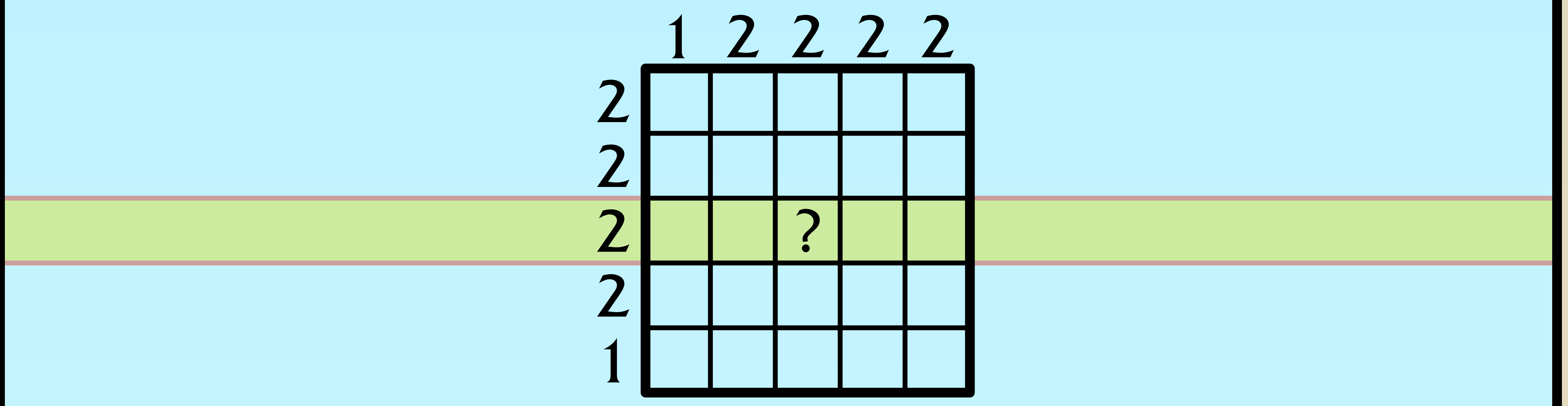
A nonogram is simple if it can be solved by only ever looking at one row or column at a time.

Solving nonograms is NP-hard, but solving simple nonograms is not.

→ It might be of interest to generate simple nonograms.

[Batenburg & Kosters, ICGA Journal, 2012]





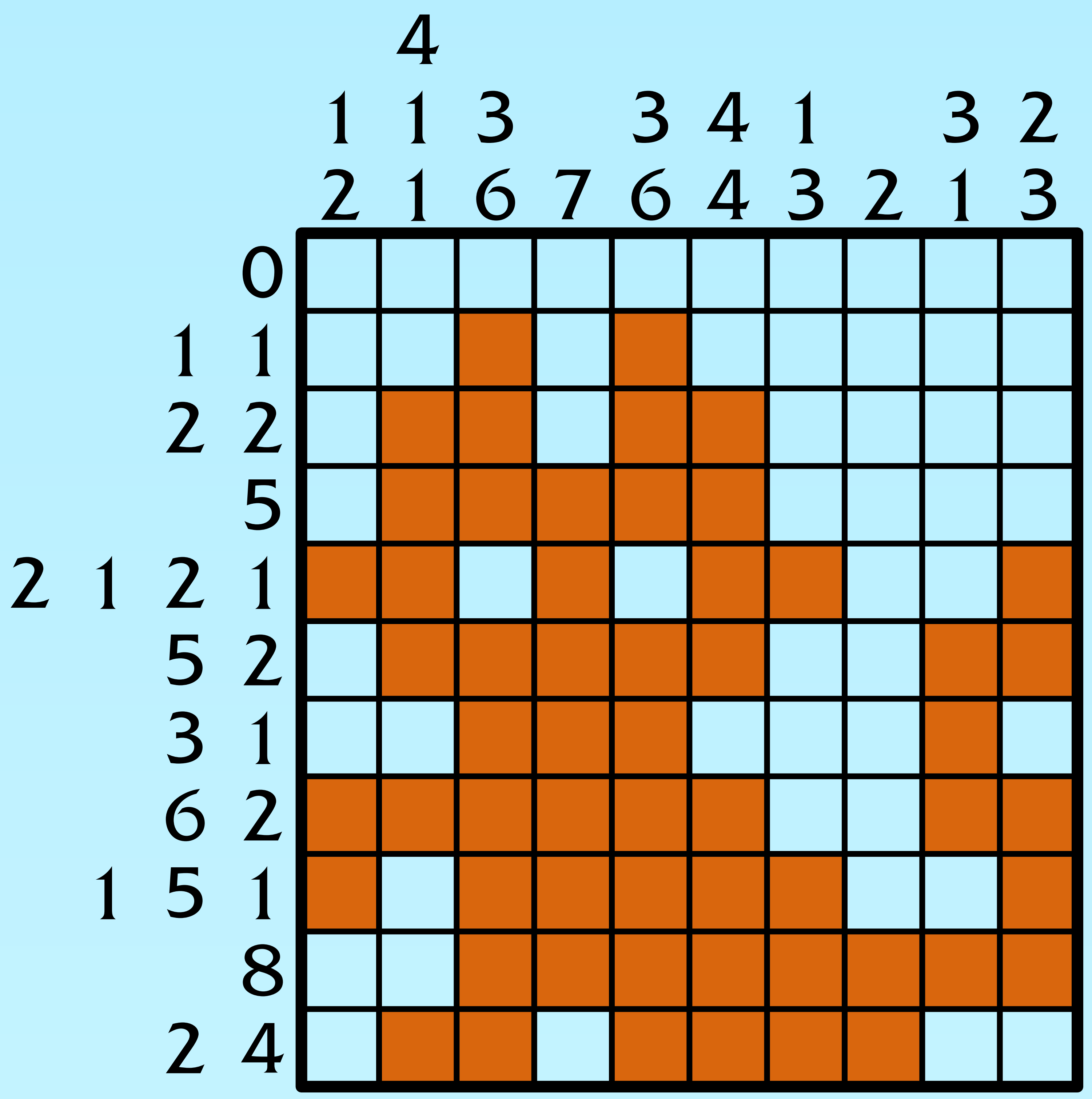
A nonogram is simple if it can be solved by only ever looking at one row or column at a time.

Solving nonograms is NP-hard, but solving simple nonograms is not.

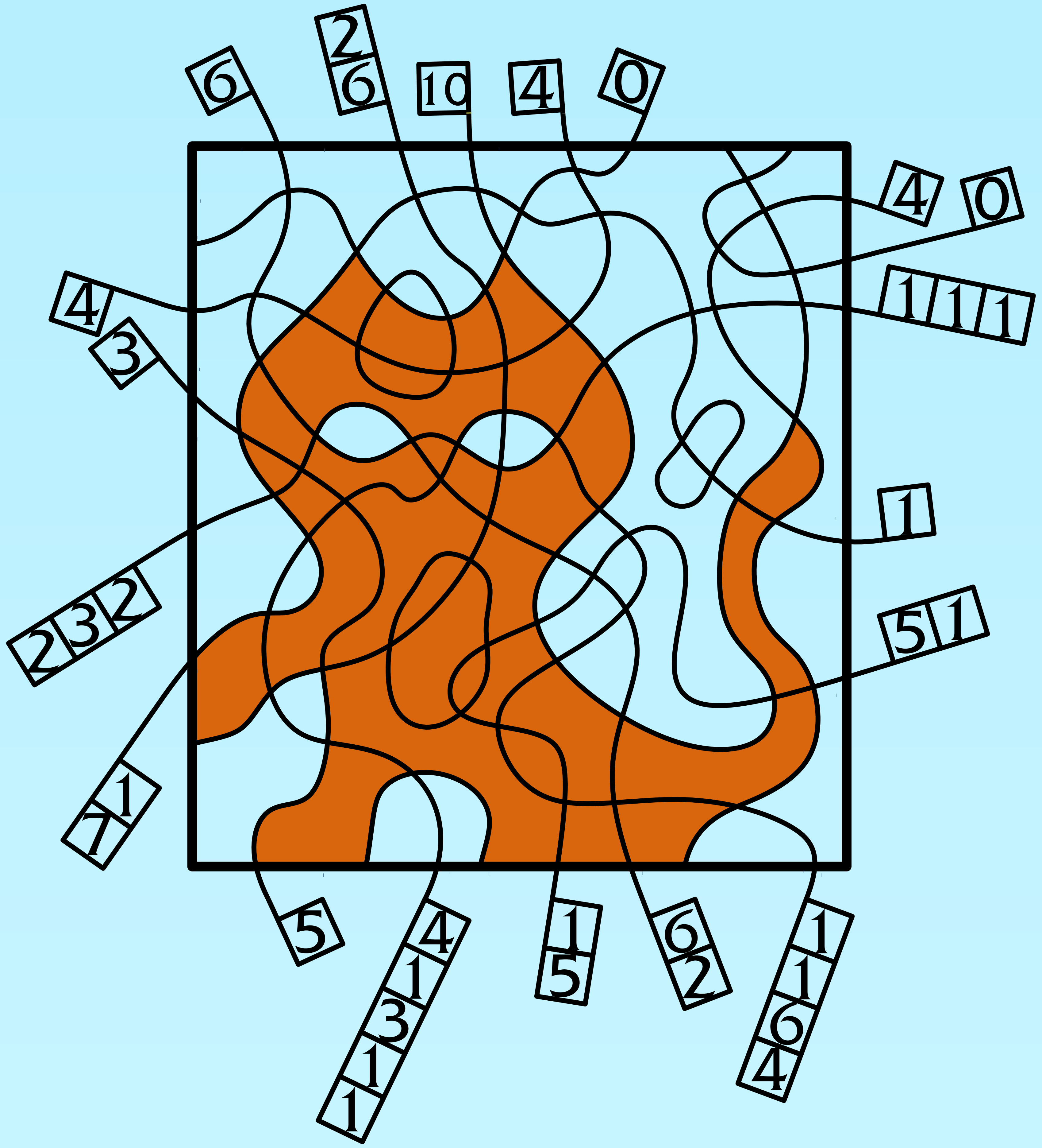
→ It might be of interest to generate simple nonograms.

More fine-grained measures of difficulty have also been studied.

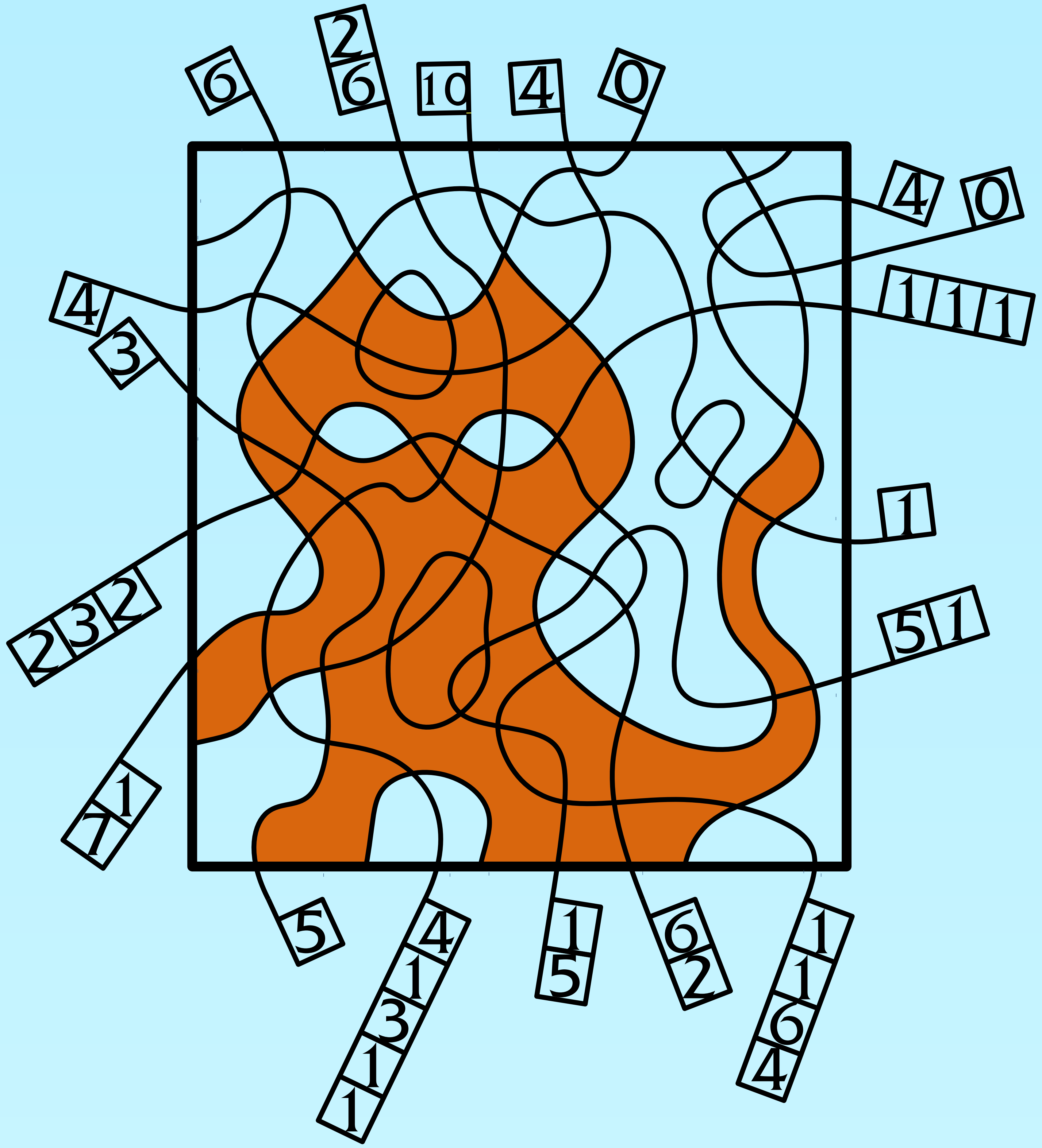
[Batenburg & Kosters, ICGA Journal, 2012]



Curved nonograms are an adaptation of nonograms that work on curve arrangements rather than grids.

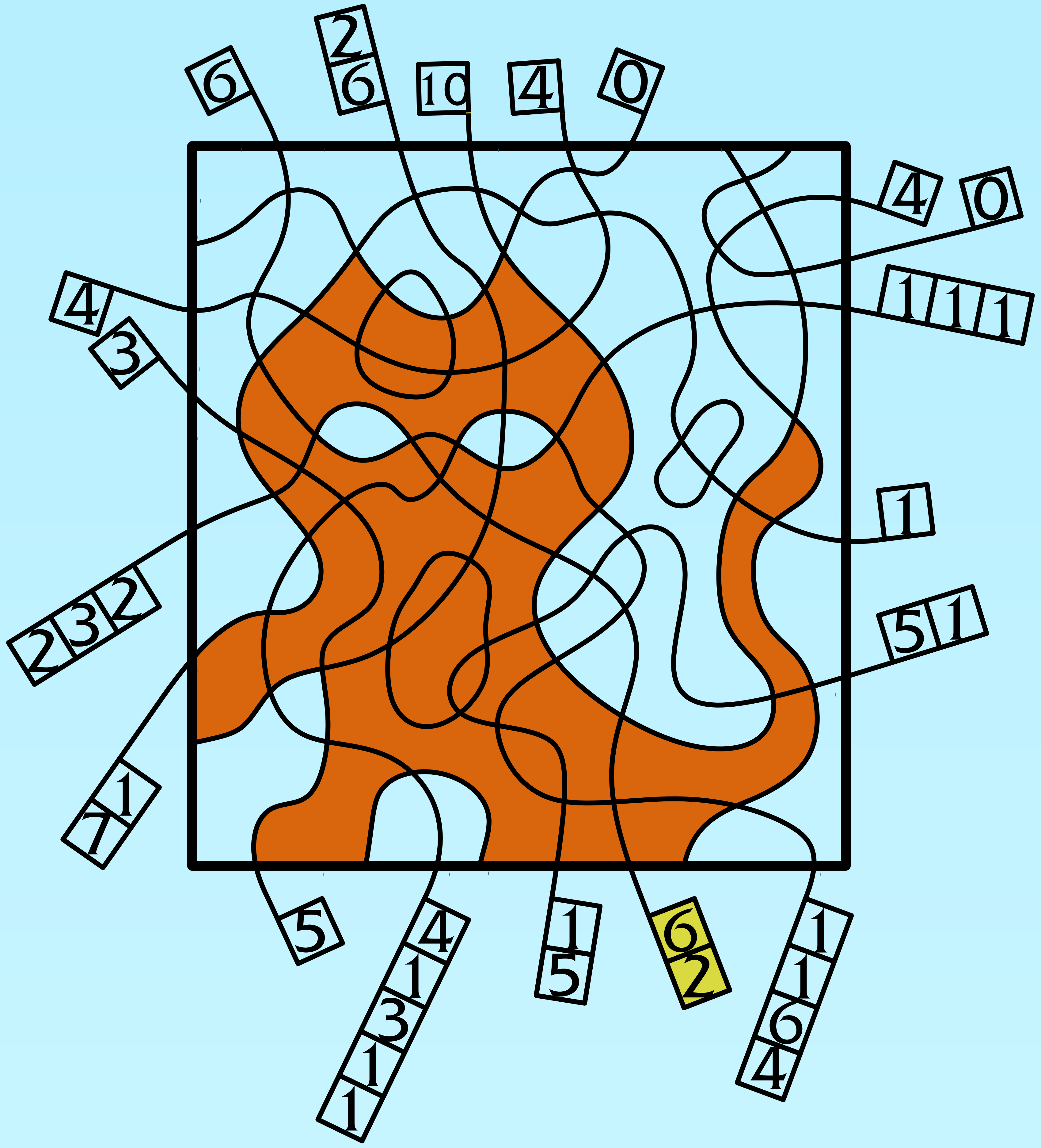


Curved nonograms are an adaptation of nonograms that work on curve arrangements rather than grids.



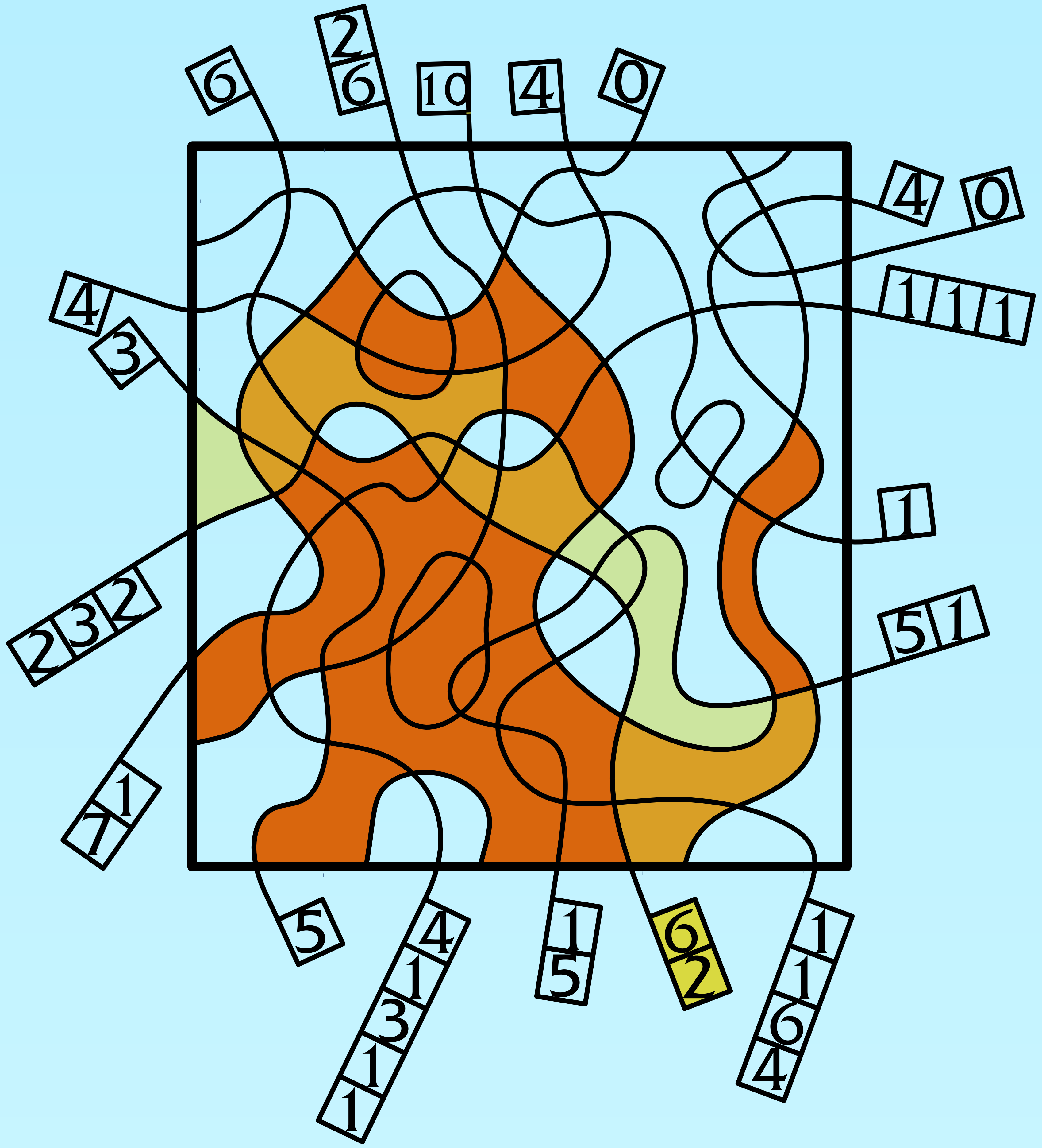
Curved nonograms are an adaptation of nonograms that work on curve arrangements rather than grids.

Similar rules as classic nonograms; clues apply to sequences of faces adjacent to a common curve.



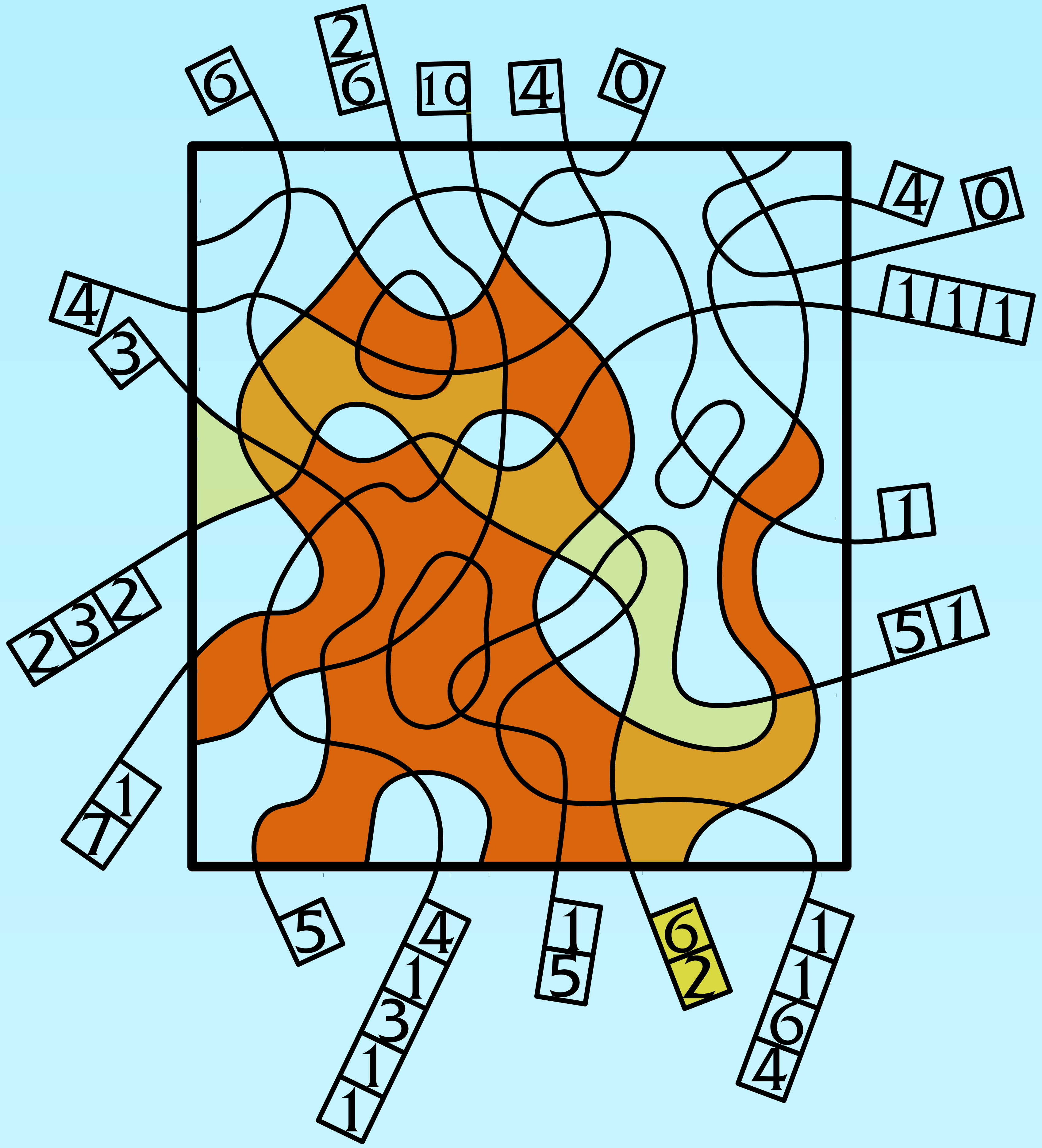
Curved nonograms are an adaptation of nonograms that work on curve arrangements rather than grids.

Similar rules as classic nonograms; clues apply to sequences of faces adjacent to a common curve.



Curved nonograms are an adaptation of nonograms that work on curve arrangements rather than grids.

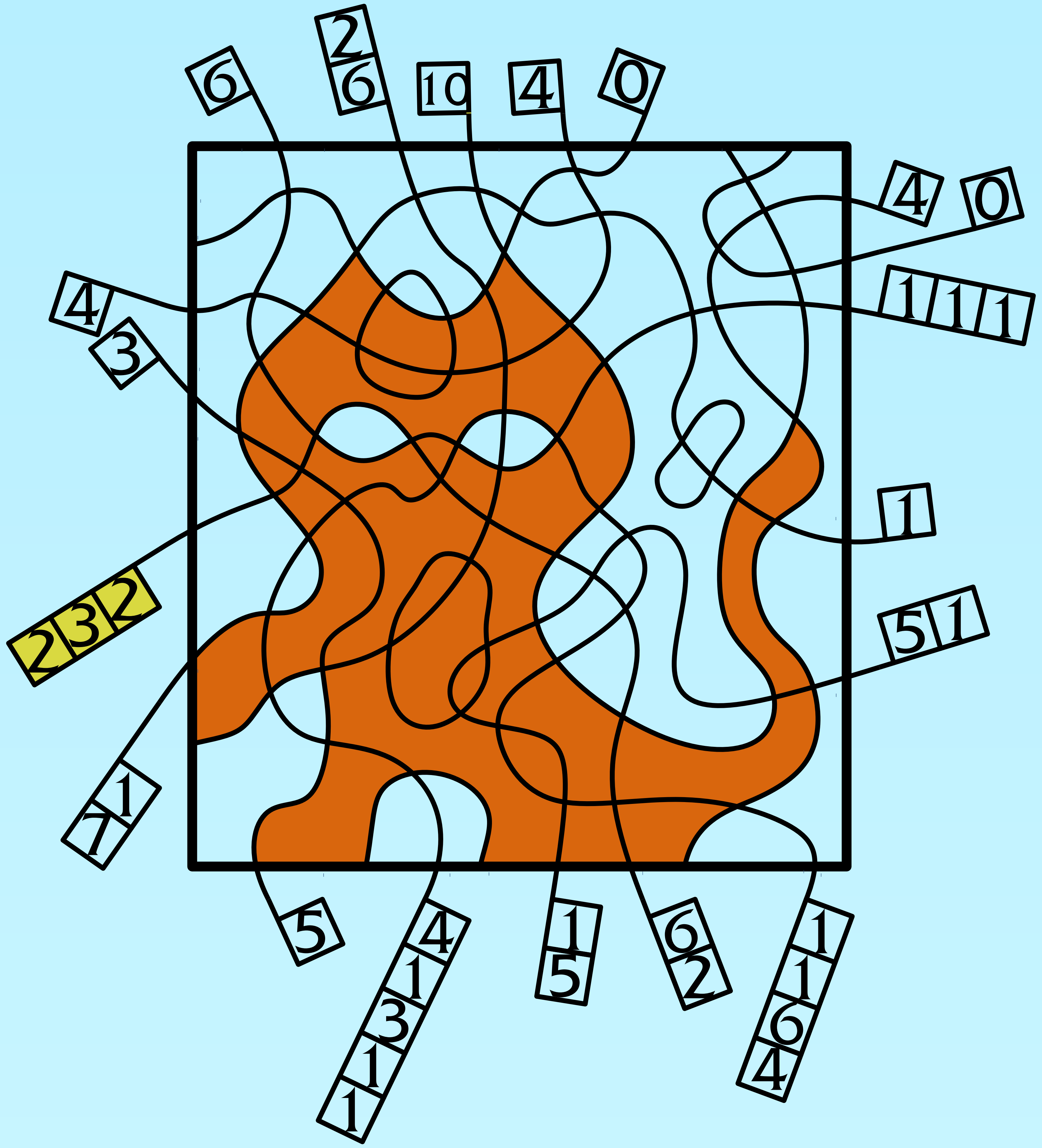
Similar rules as classic nonograms; clues apply to sequences of faces adjacent to a common curve.



Curved nonograms are an adaptation of nonograms that work on curve arrangements rather than grids.

Similar rules as classic nonograms; clues apply to sequences of faces adjacent to a common curve.

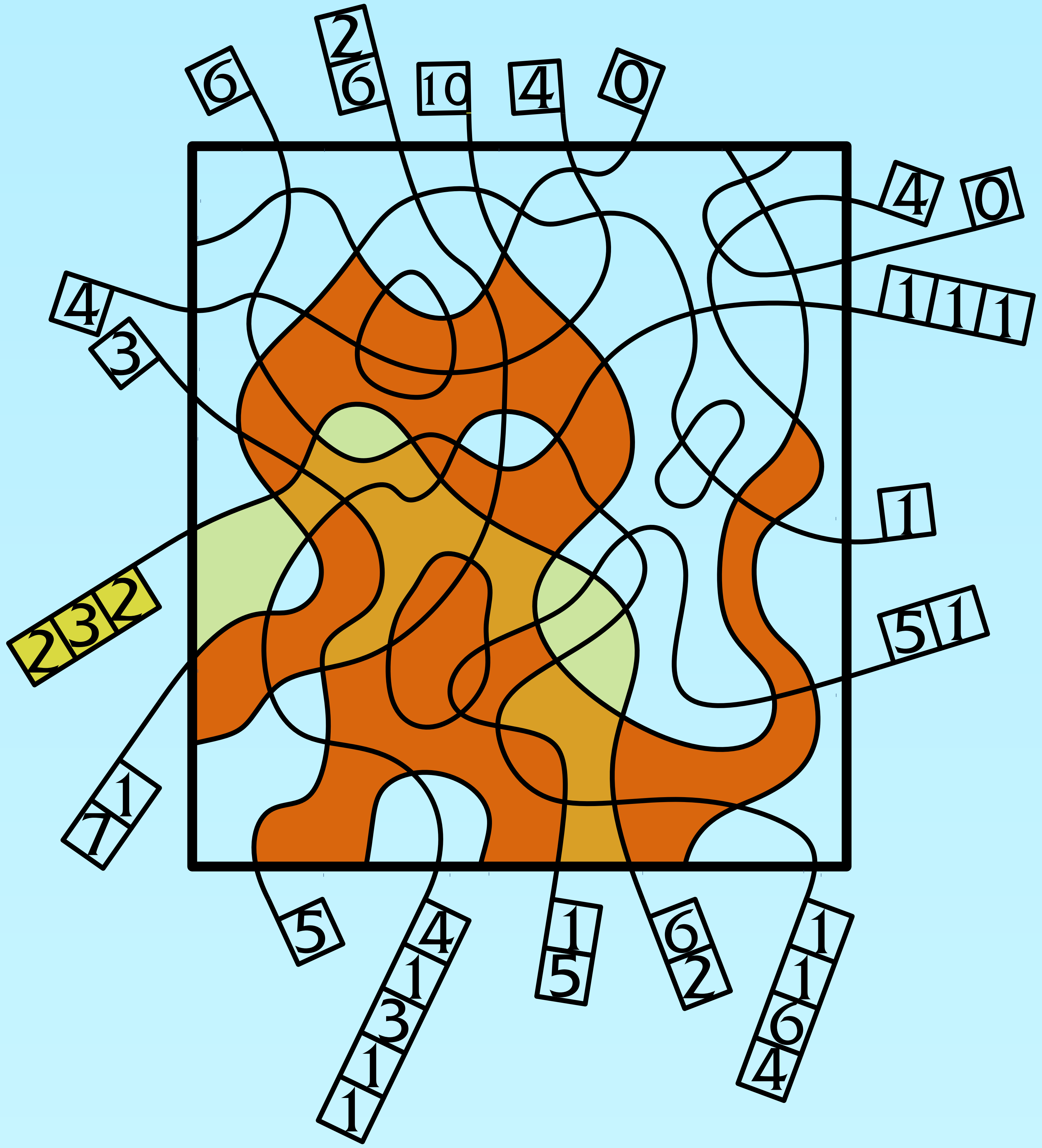
Curves have two sides with separate clues.



Curved nonograms are an adaptation of nonograms that work on curve arrangements rather than grids.

Similar rules as classic nonograms; clues apply to sequences of faces adjacent to a common curve.

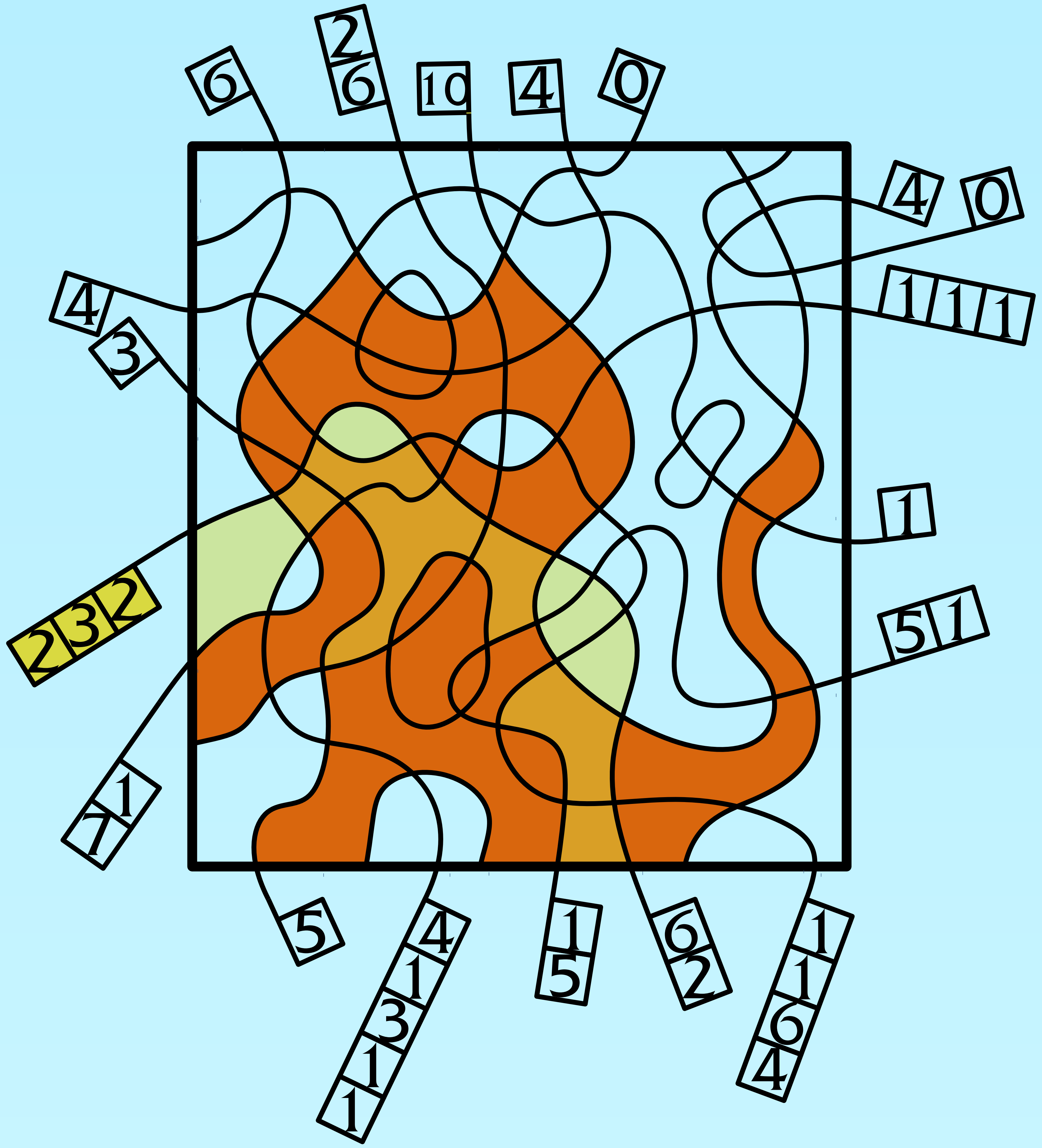
Curves have two sides with separate clues.



Curved nonograms are an adaptation of nonograms that work on curve arrangements rather than grids.

Similar rules as classic nonograms; clues apply to sequences of faces adjacent to a common curve.

Curves have two sides with separate clues.

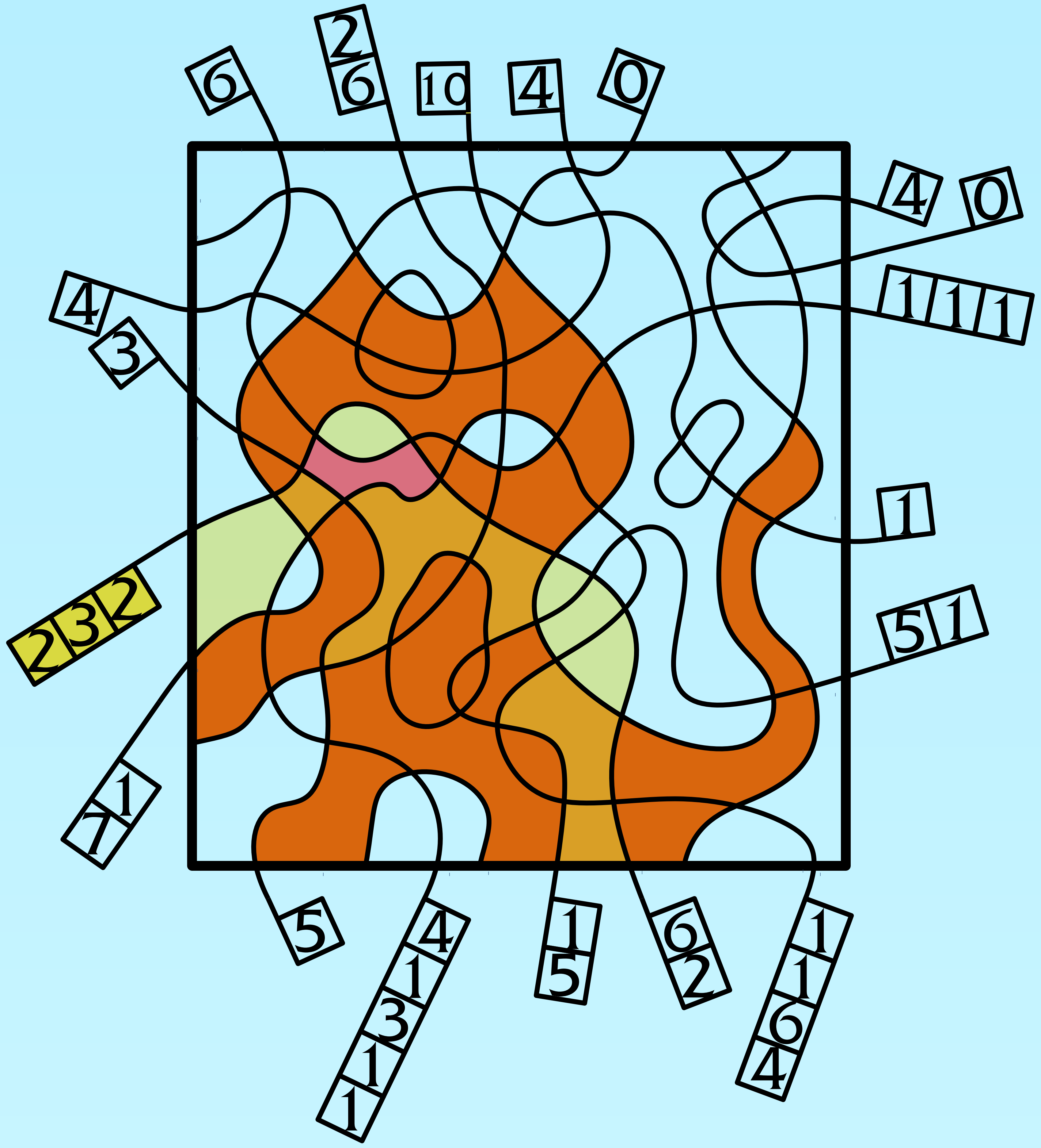


Curved nonograms are an adaptation of nonograms that work on curve arrangements rather than grids.

Similar rules as classic nonograms; clues apply to sequences of faces adjacent to a common curve.

Curves have two sides with separate clues.

Cells can appear twice!



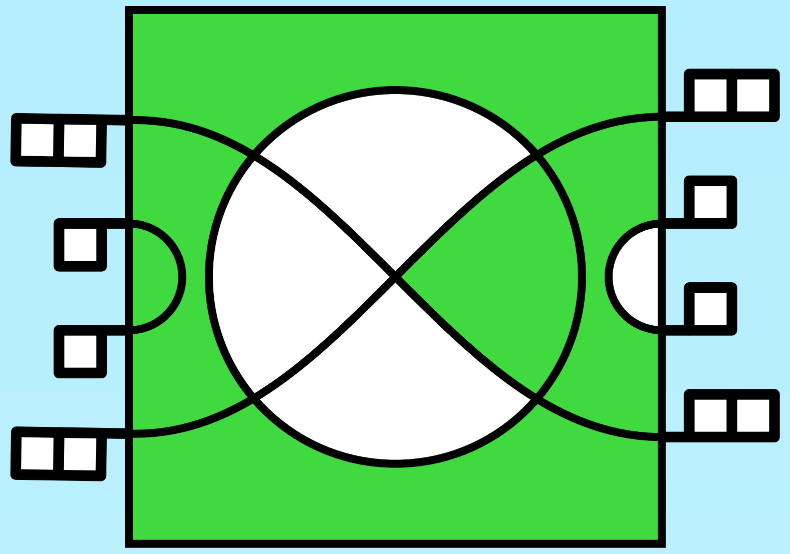
Curved nonograms are an adaptation of nonograms that work on curve arrangements rather than grids.

Similar rules as classic nonograms; clues apply to sequences of faces adjacent to a common curve.

Curves have two sides with separate clues.

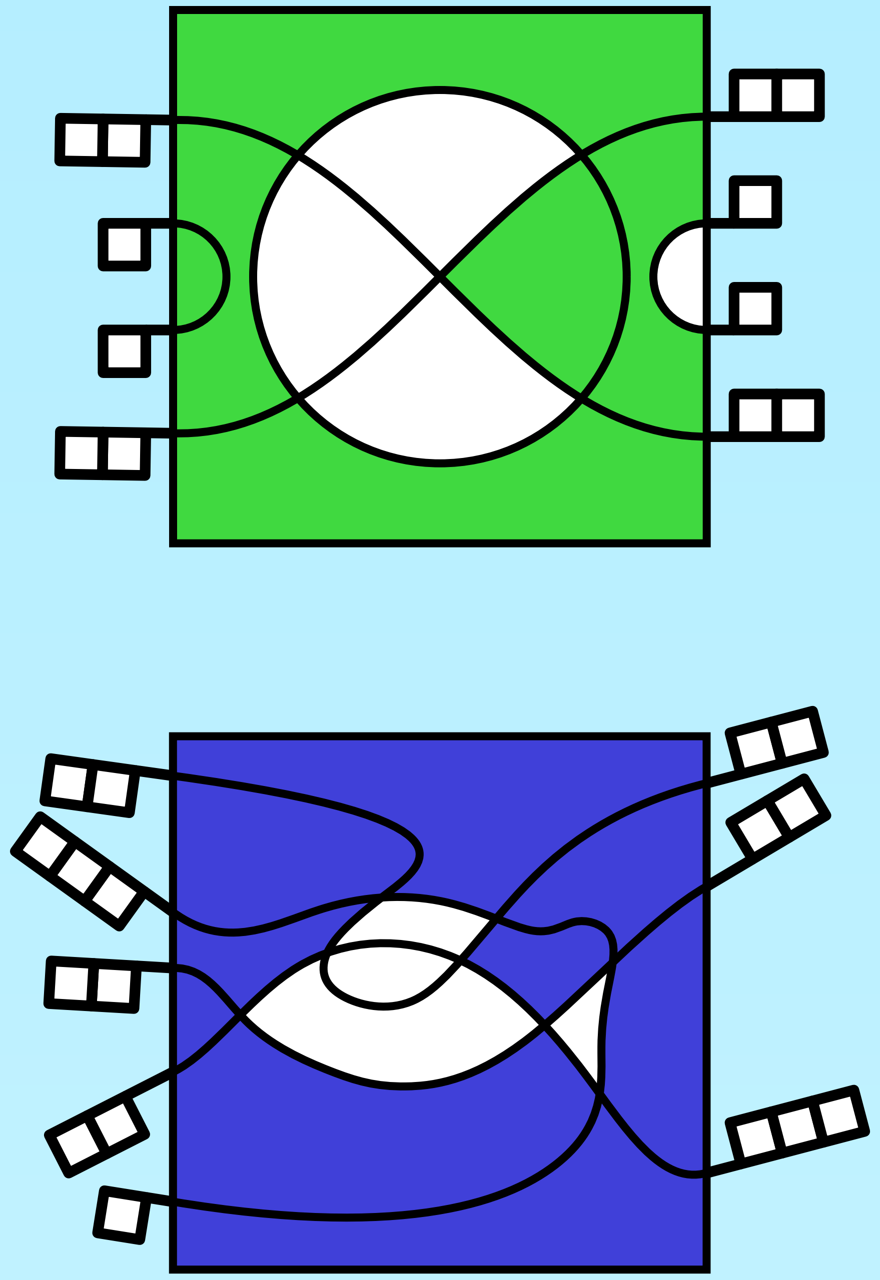
Cells can appear twice!

We can identify three levels of "conceptual difficulty".



We can identify three levels of "conceptual difficulty".

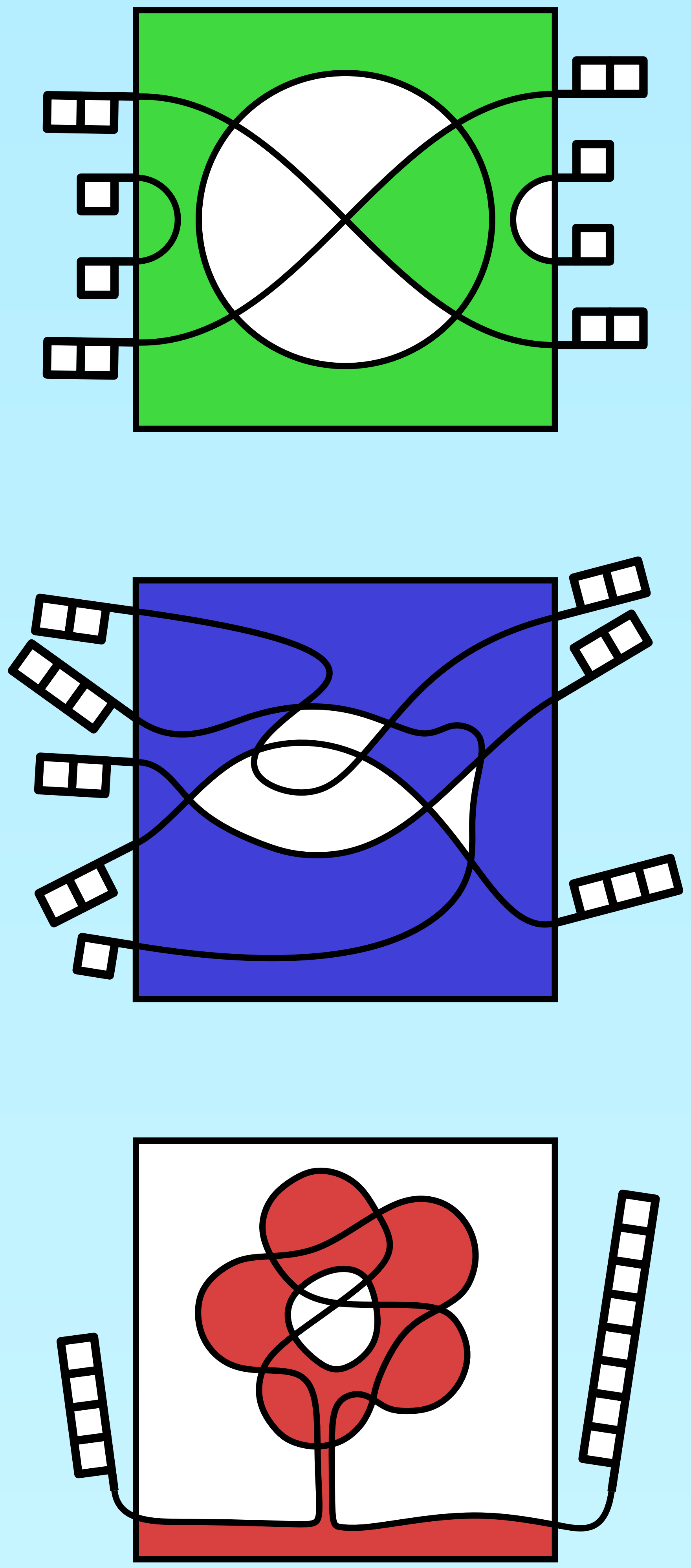
Basic nonograms do not have any duplicate cells adjacent to the same curve.



We can identify three levels of “conceptual difficulty”.

Basic nonograms do not have any duplicate cells adjacent to the same curve.

Advanced nonograms may have duplicate cells adjacent to the same side of a curve.

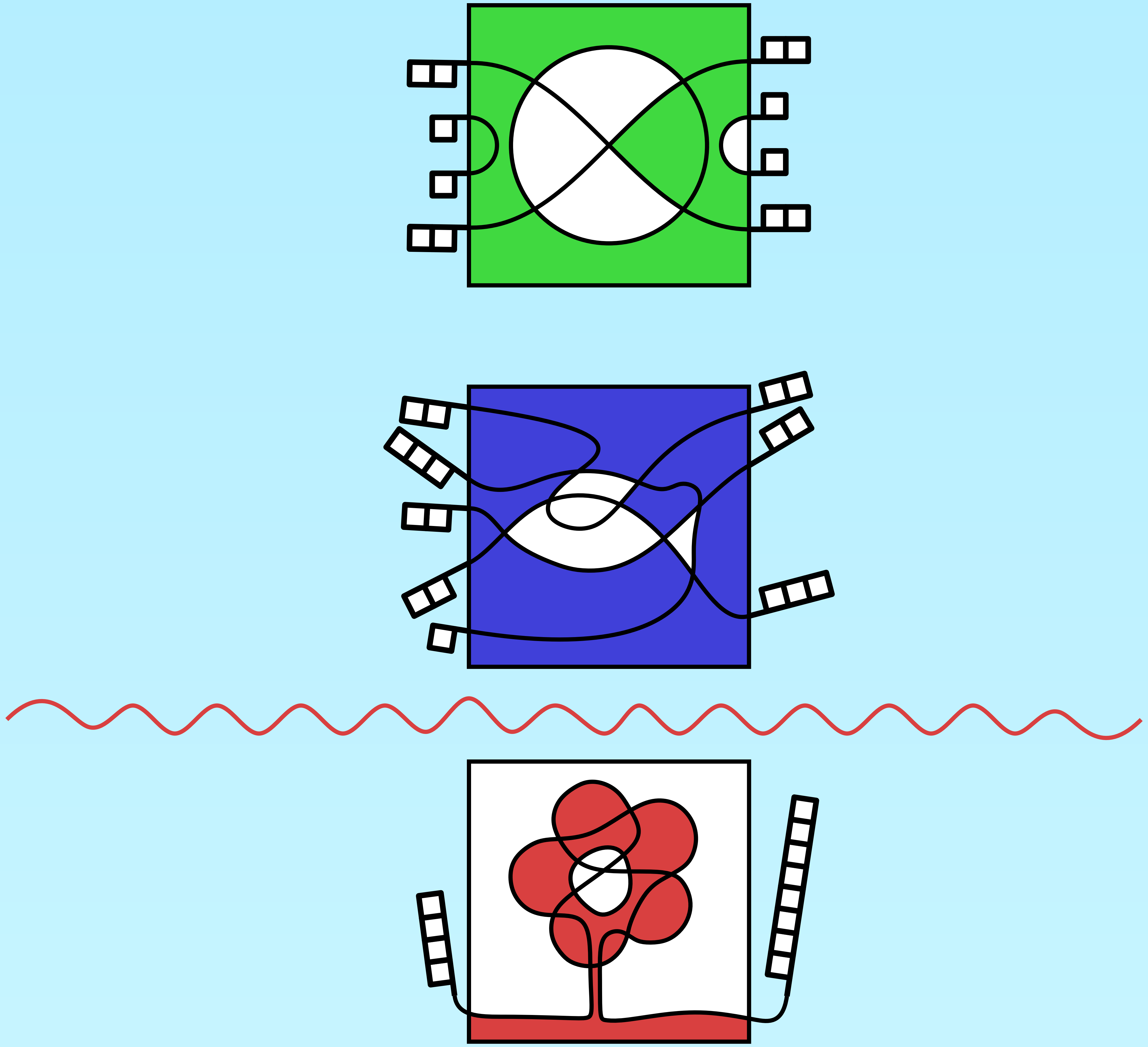


We can identify three levels of "conceptual difficulty".

Basic nonograms do not have any duplicate cells adjacent to the same curve.

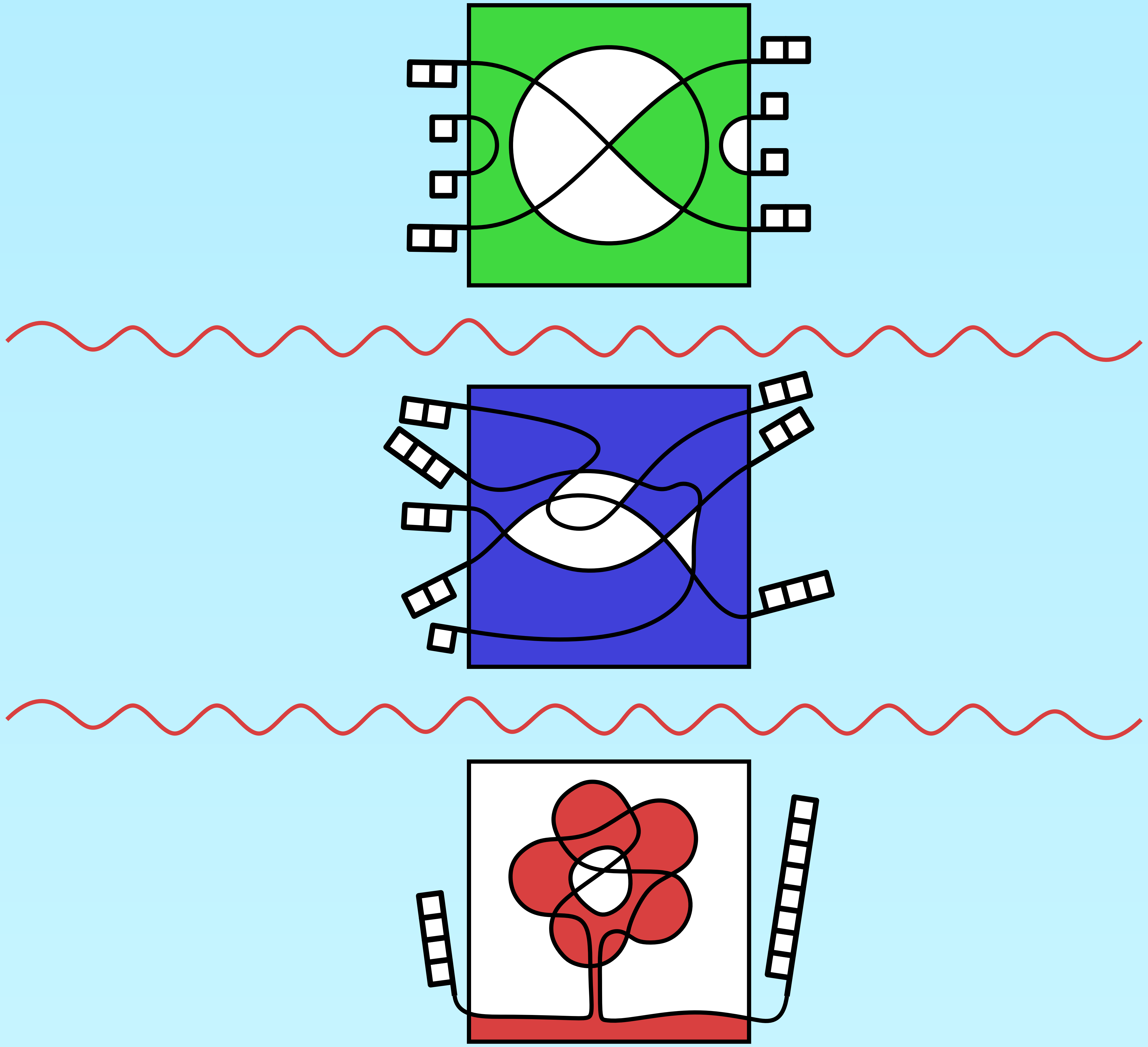
Advanced nonograms may have duplicate cells adjacent to the same side of a curve.

Expert nonograms have duplicate cells adjacent to opposite sides of a curve.



We can identify three levels of “conceptual difficulty”.

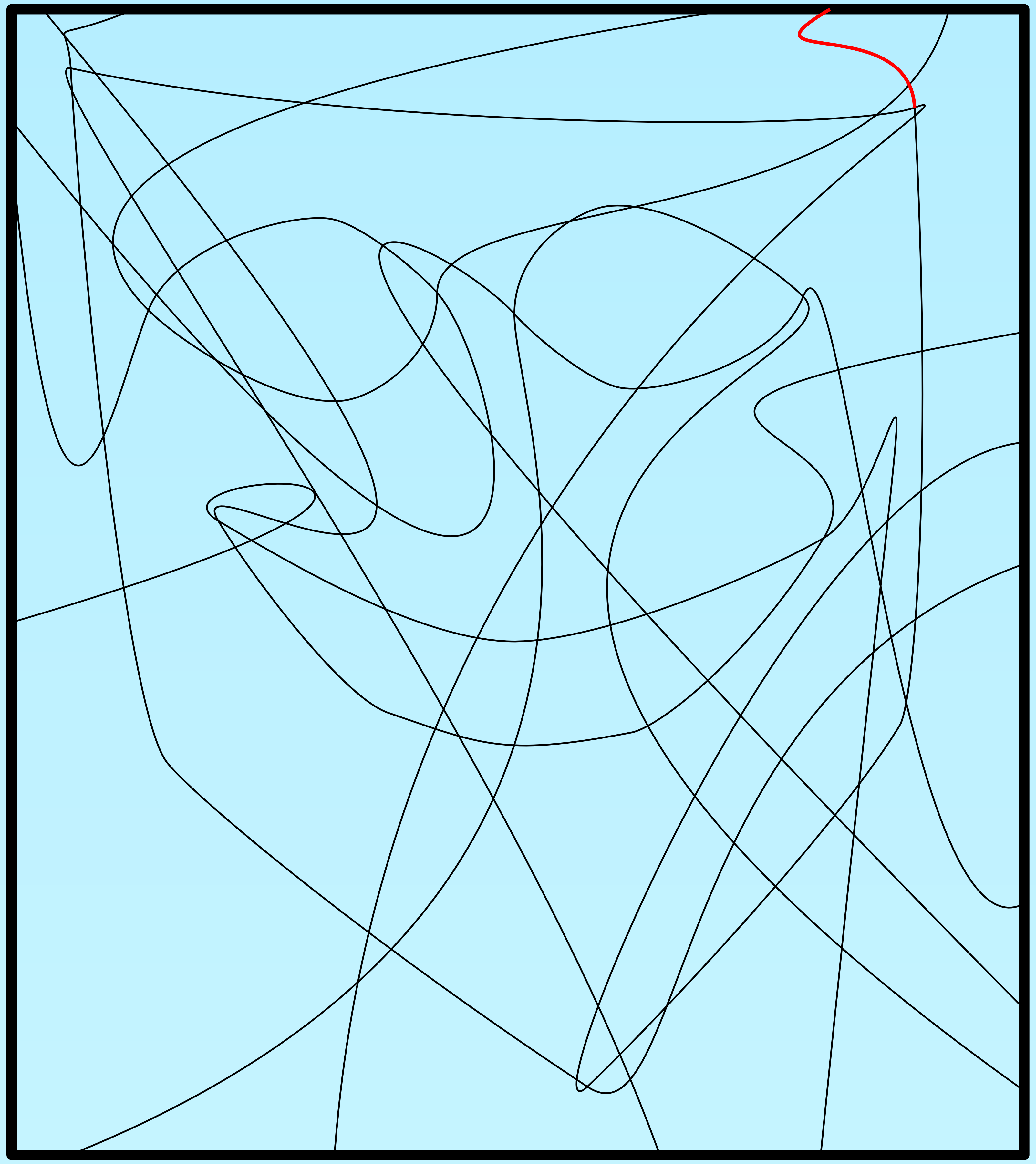
Advanced versus Expert: presence of self-intersections



We can identify three levels of "conceptual difficulty".

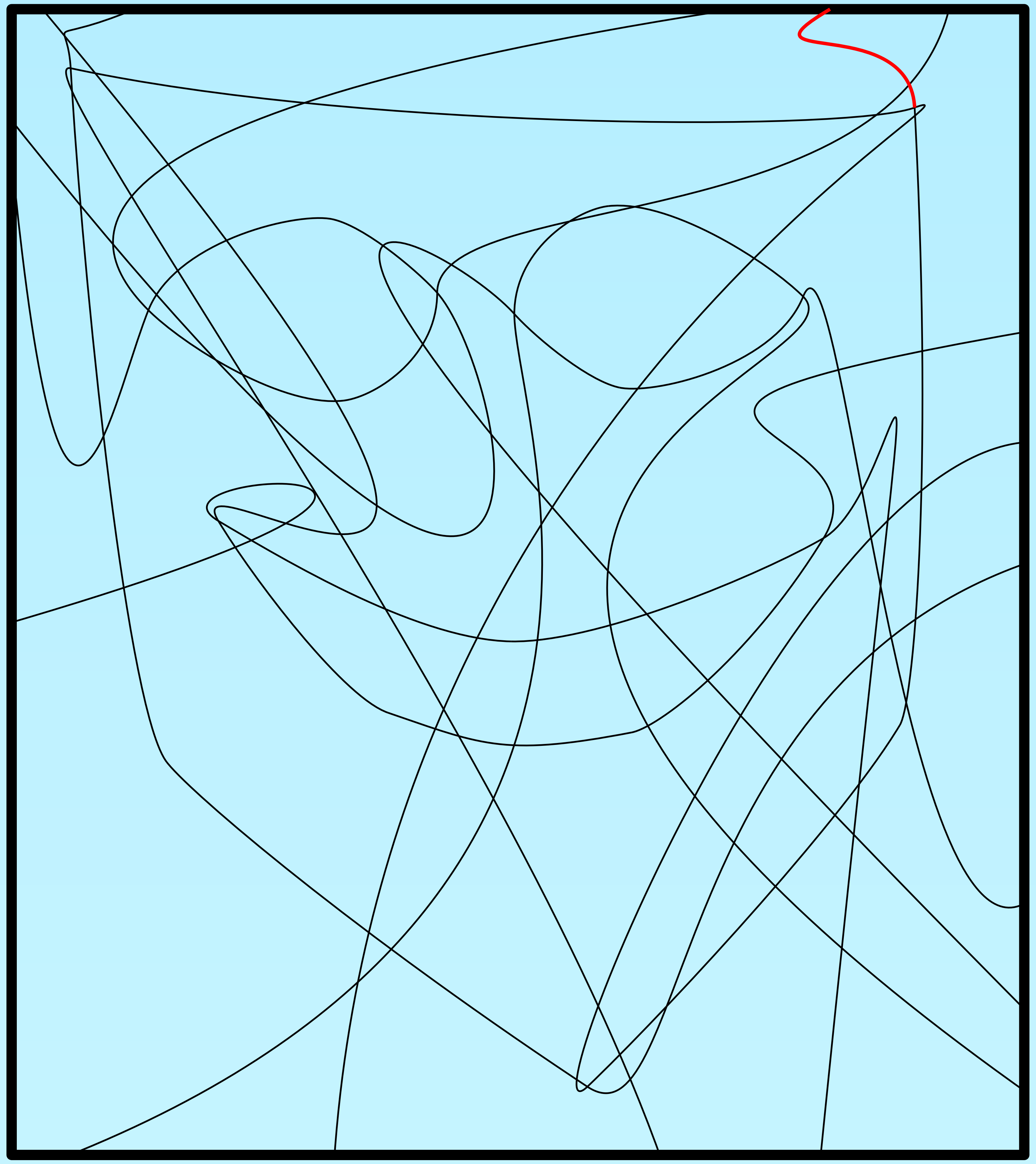
Basic versus Advanced:
presence of popular faces

Advanced versus Expert:
presence of self-intersections



One can generate nonograms from desired output pictures.

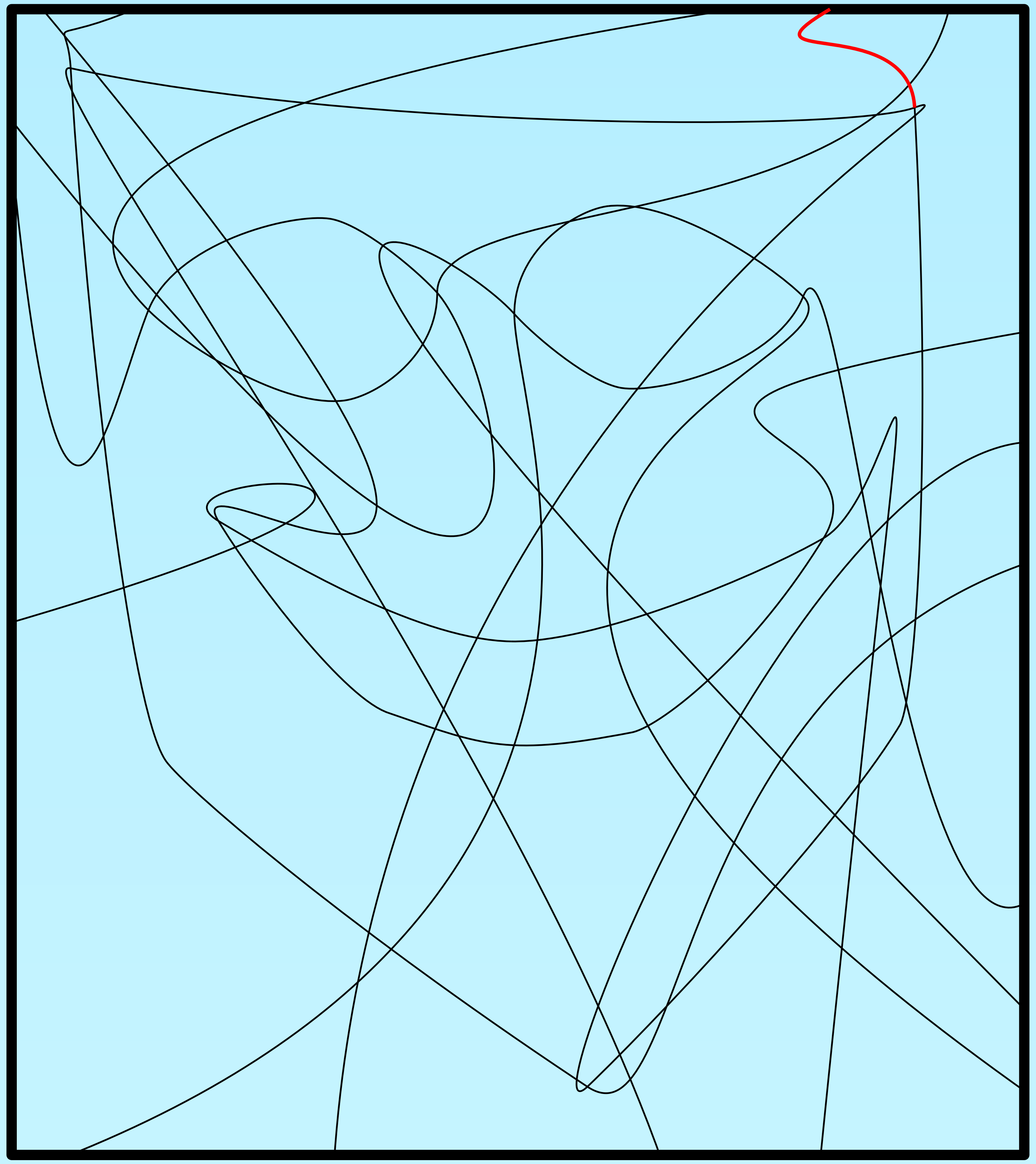
[van de Kerkhof, de Jong, Parment, L, Vaxman & van Kreveld, EuroGraphics 2019]



One can generate nonograms from desired output pictures.

Generated puzzles are often advanced.

[van de Kerkhof, de Jong, Parment, L, Vaxman & van Kreveld, EuroGraphics 2019]

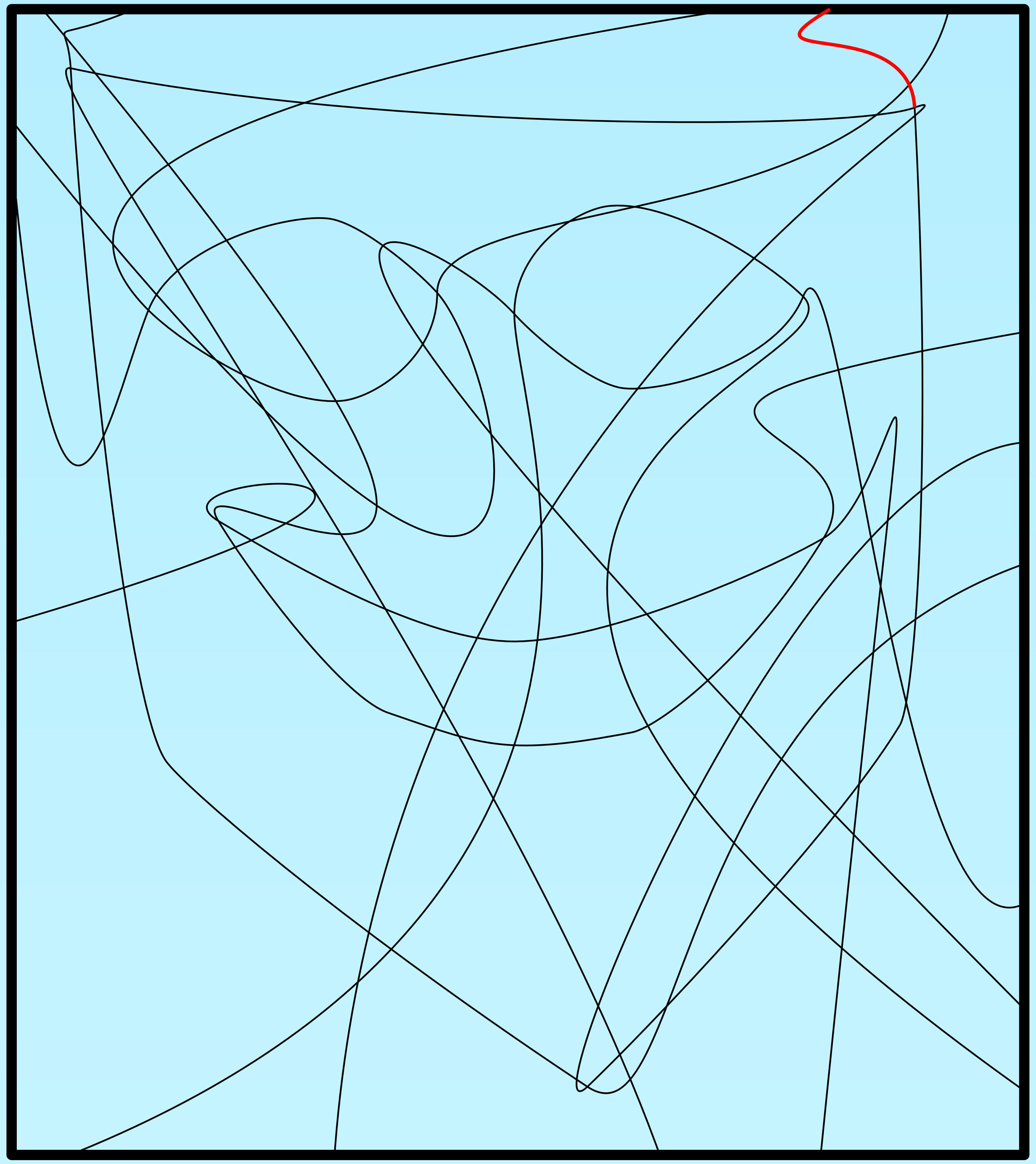


One can generate nonograms from desired output pictures.

Generated puzzles are often advanced.

User studies suggest a higher demand for basic puzzles.

[van de Kerkhof, de Jong, Parment, L, Vaxman & van Kreveld, EuroGraphics 2019]



One can generate nonograms from desired output pictures.

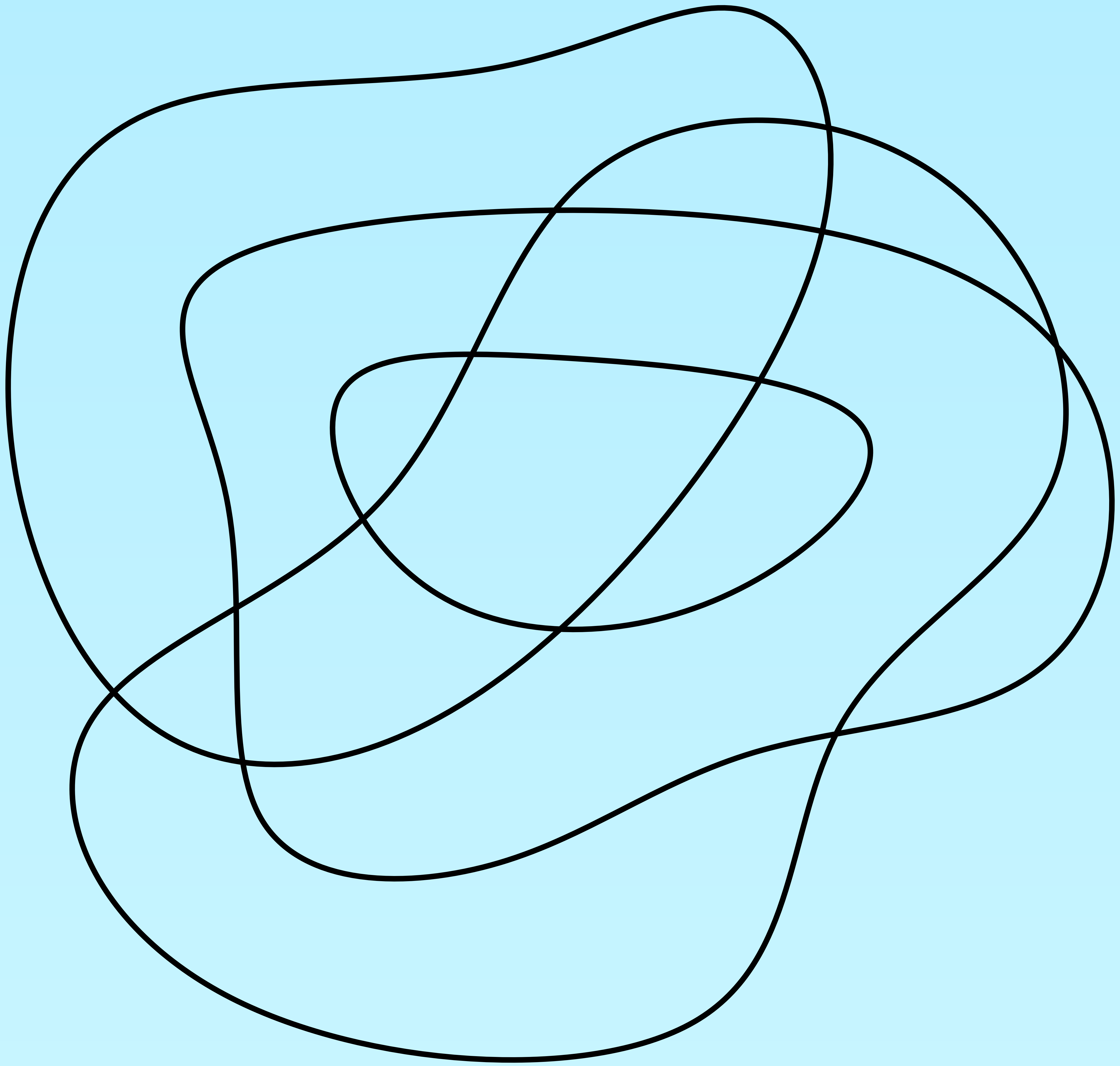
Generated puzzles are often advanced.

User studies suggest a higher demand for basic puzzles.

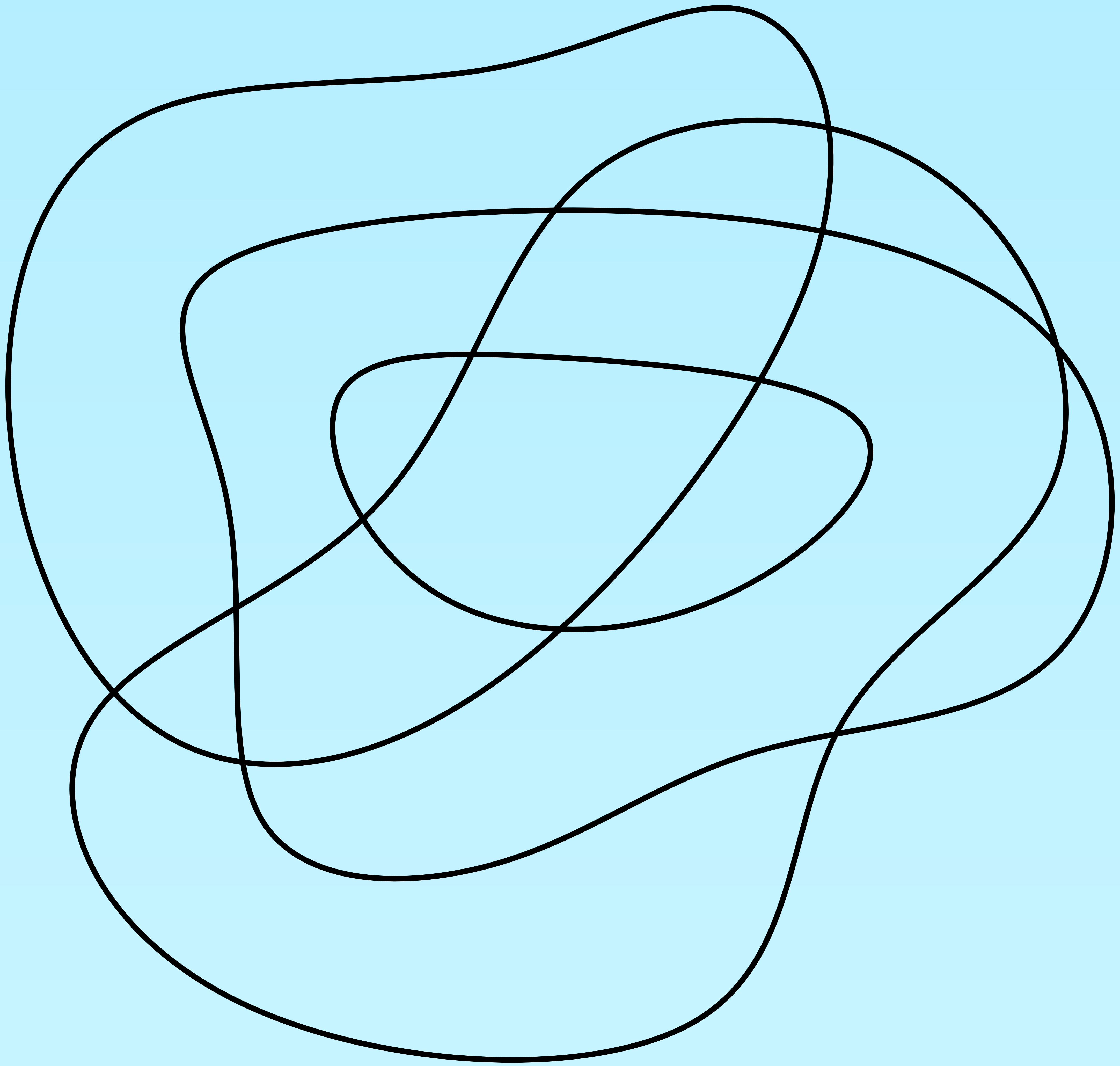
Can we turn advanced puzzles into basic ones?

[van de Kerkhof, de Jong, Parment, L, Vaxman & van Kreveld, EuroGraphics 2019]

Part 2
POPULAR FACES

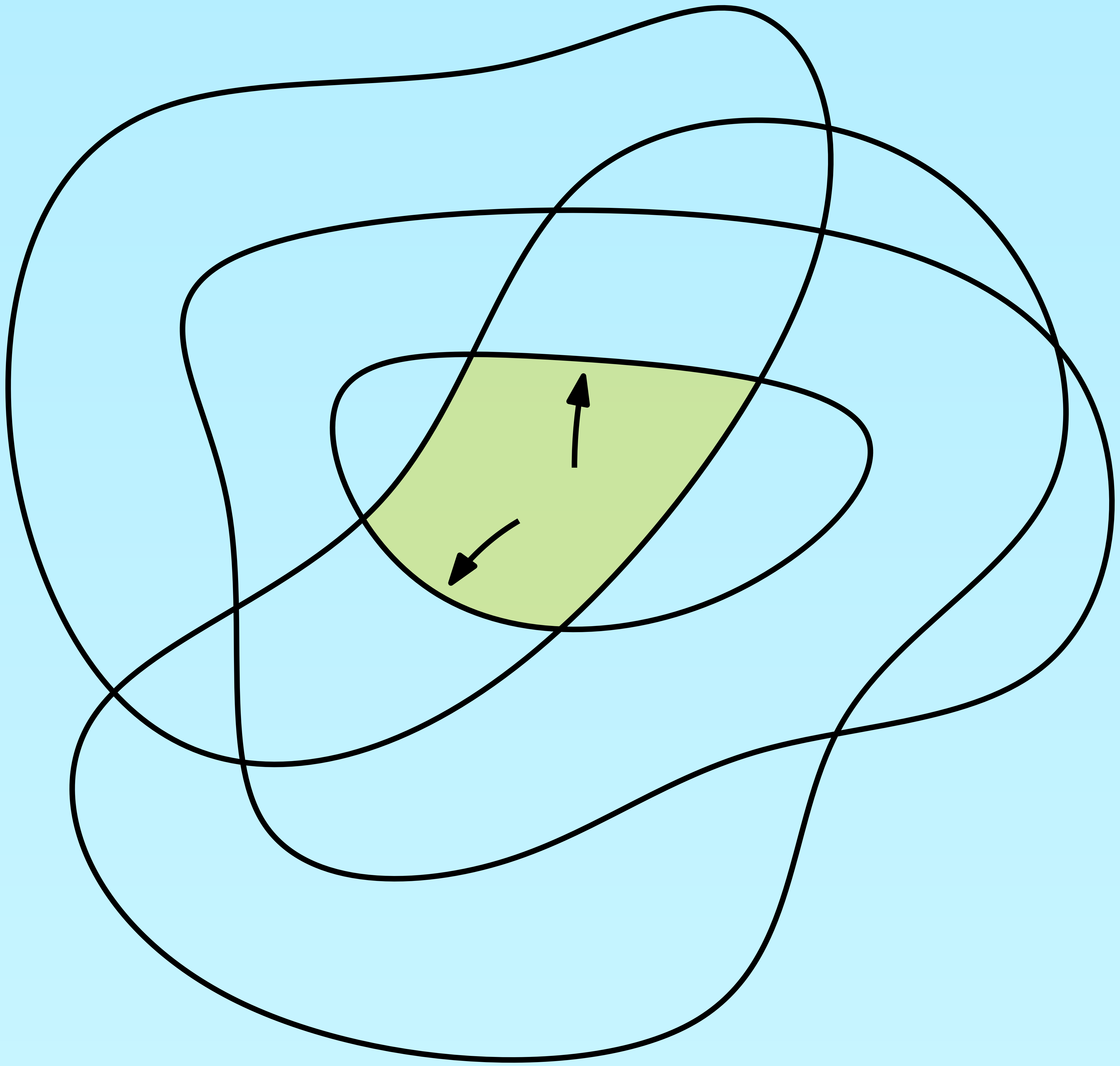


Consider an arrangement of simple closed curves in the plane.



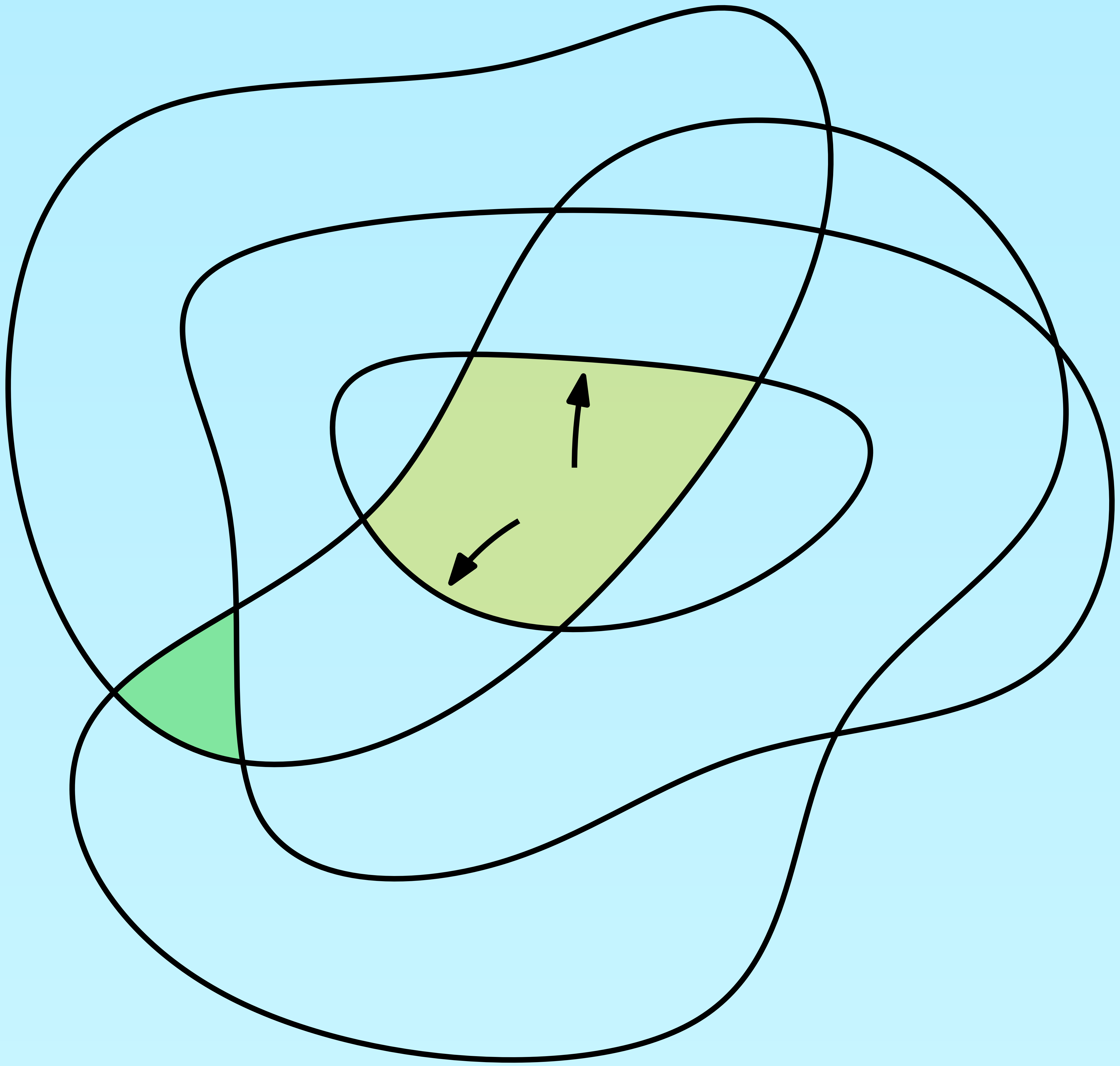
Consider an arrangement of simple closed curves in the plane.

A face of the arrangement is popular if it is adjacent to the same curve twice.



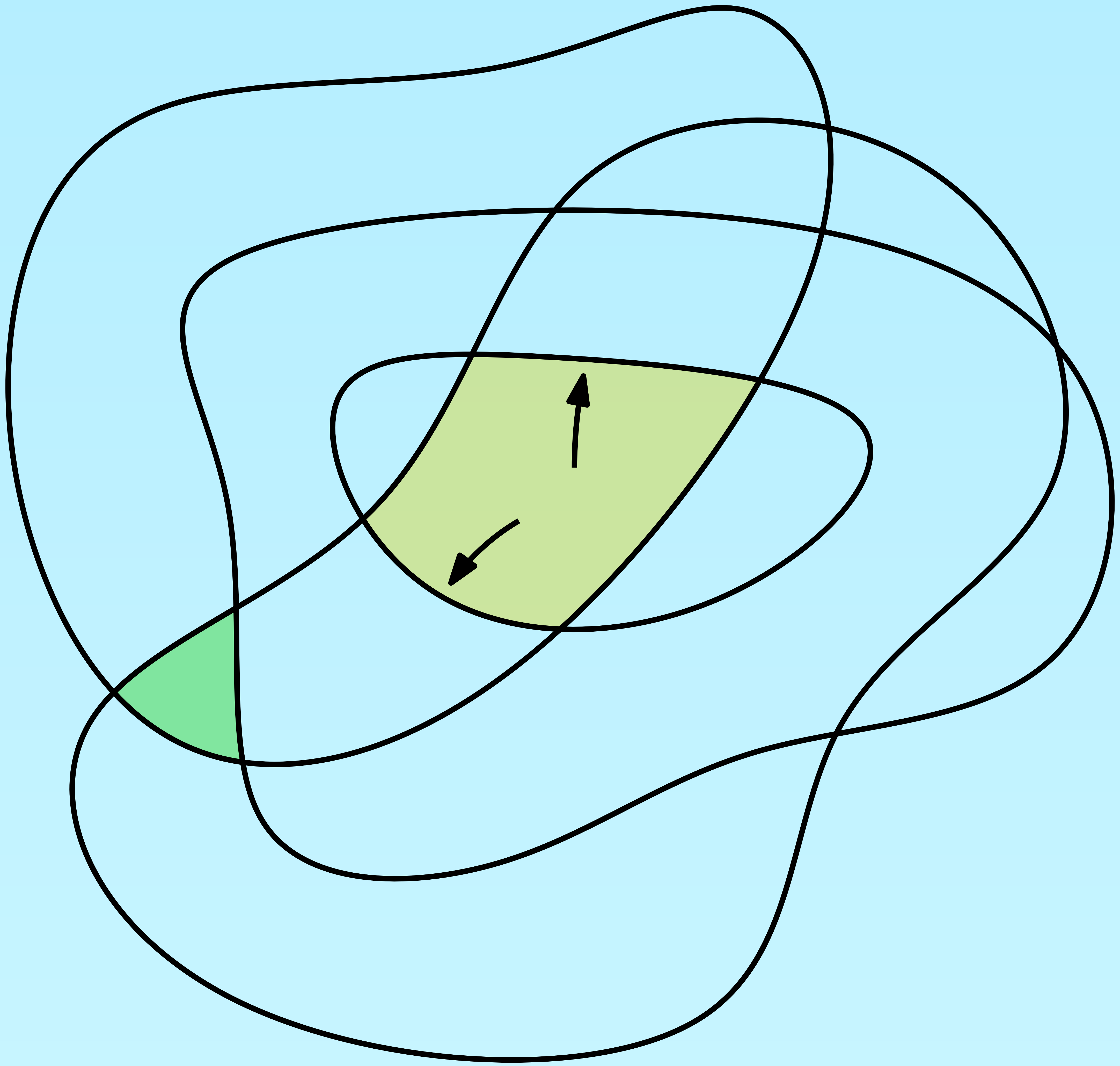
Consider an arrangement of simple closed curves in the plane.

A face of the arrangement is popular if it is adjacent to the same curve twice.



Consider an arrangement of simple closed curves in the plane.

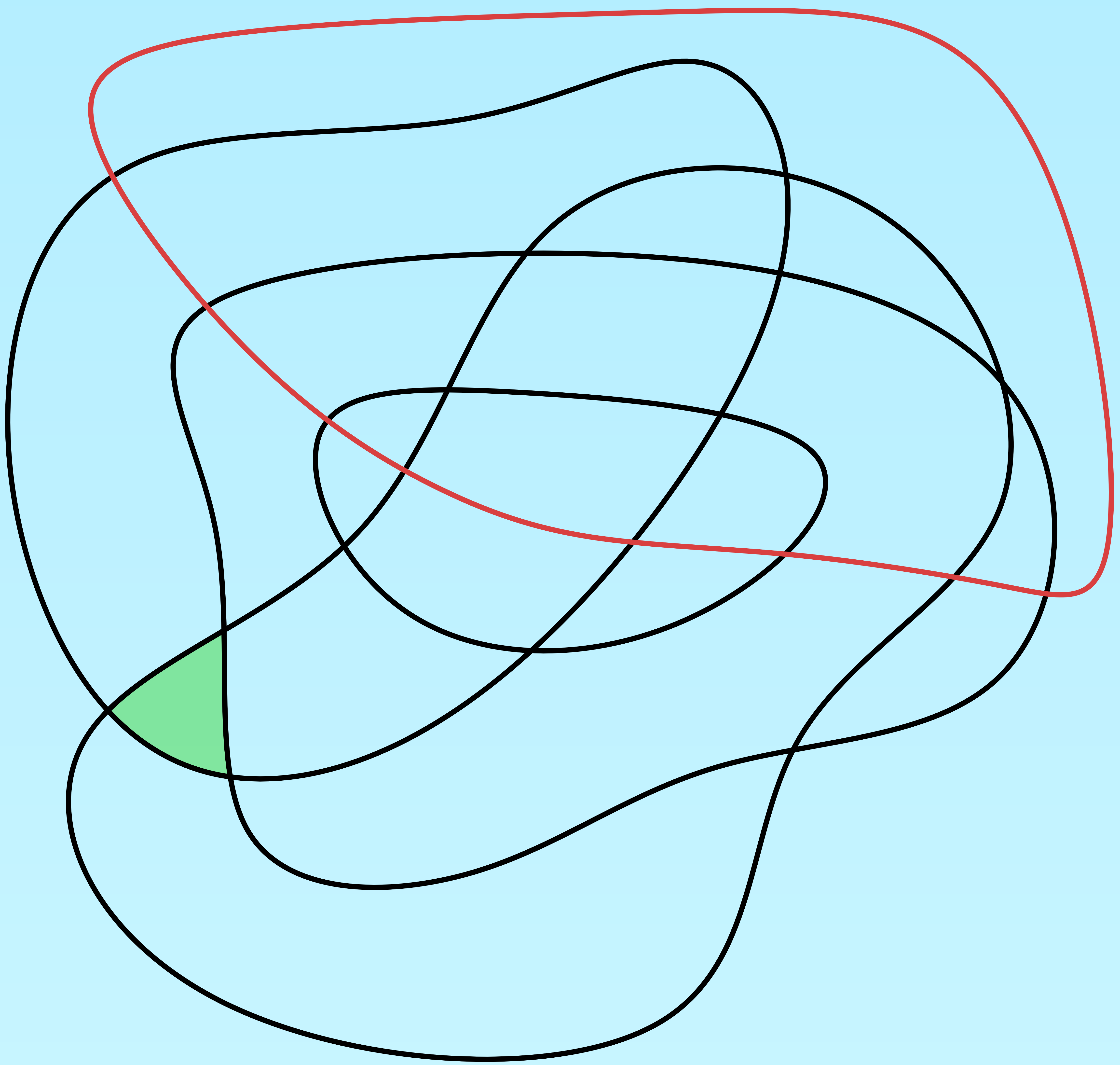
A face of the arrangement is popular if it is adjacent to the same curve twice.



Consider an arrangement of simple closed curves in the plane.

A face of the arrangement is popular if it is adjacent to the same curve twice.

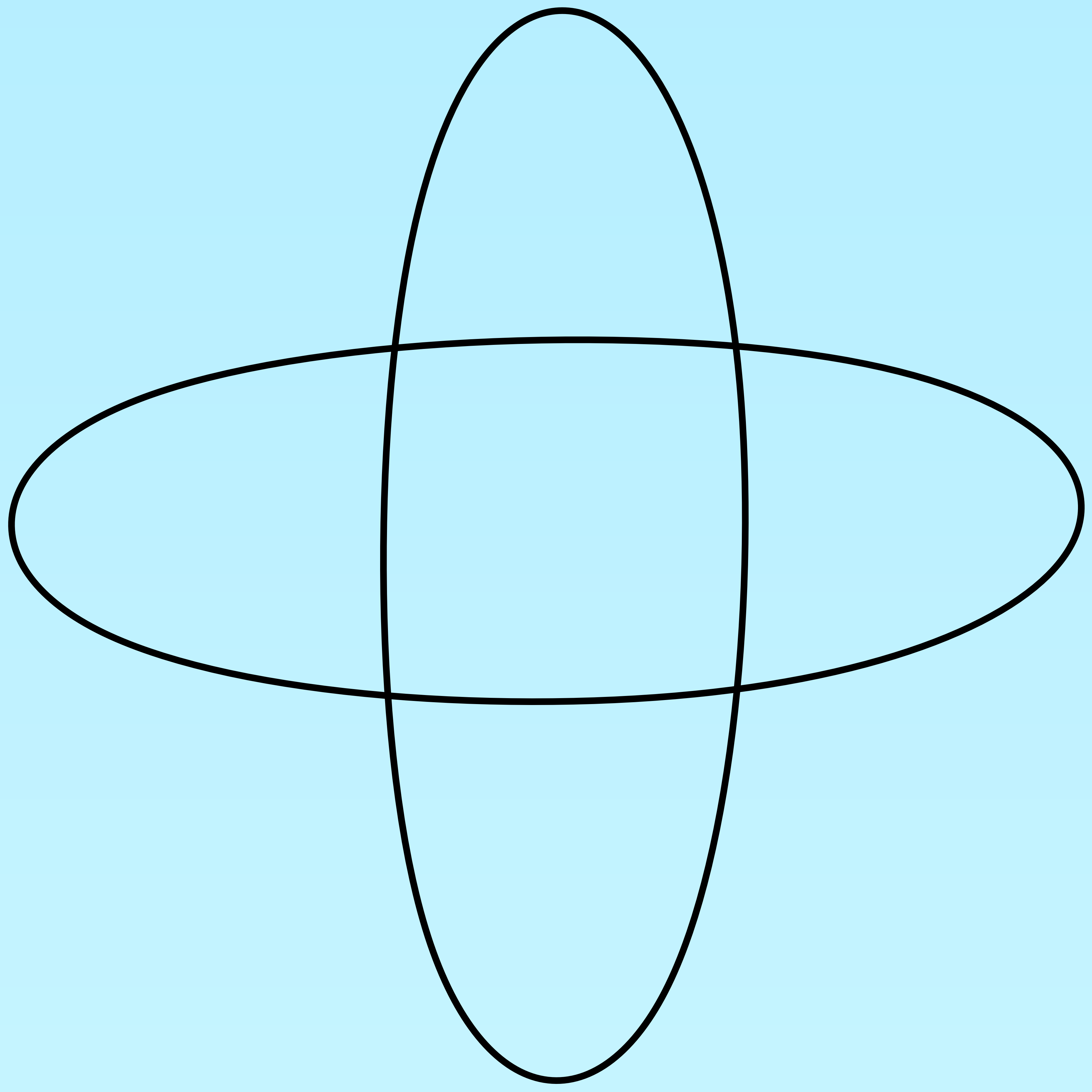
Problem. Can we add a curve to the arrangement, such that the resulting arrangement has no popular faces?



Consider an arrangement of simple closed curves in the plane.

A face of the arrangement is popular if it is adjacent to the same curve twice.

Problem. Can we add a curve to the arrangement, such that the resulting arrangement has no popular faces?

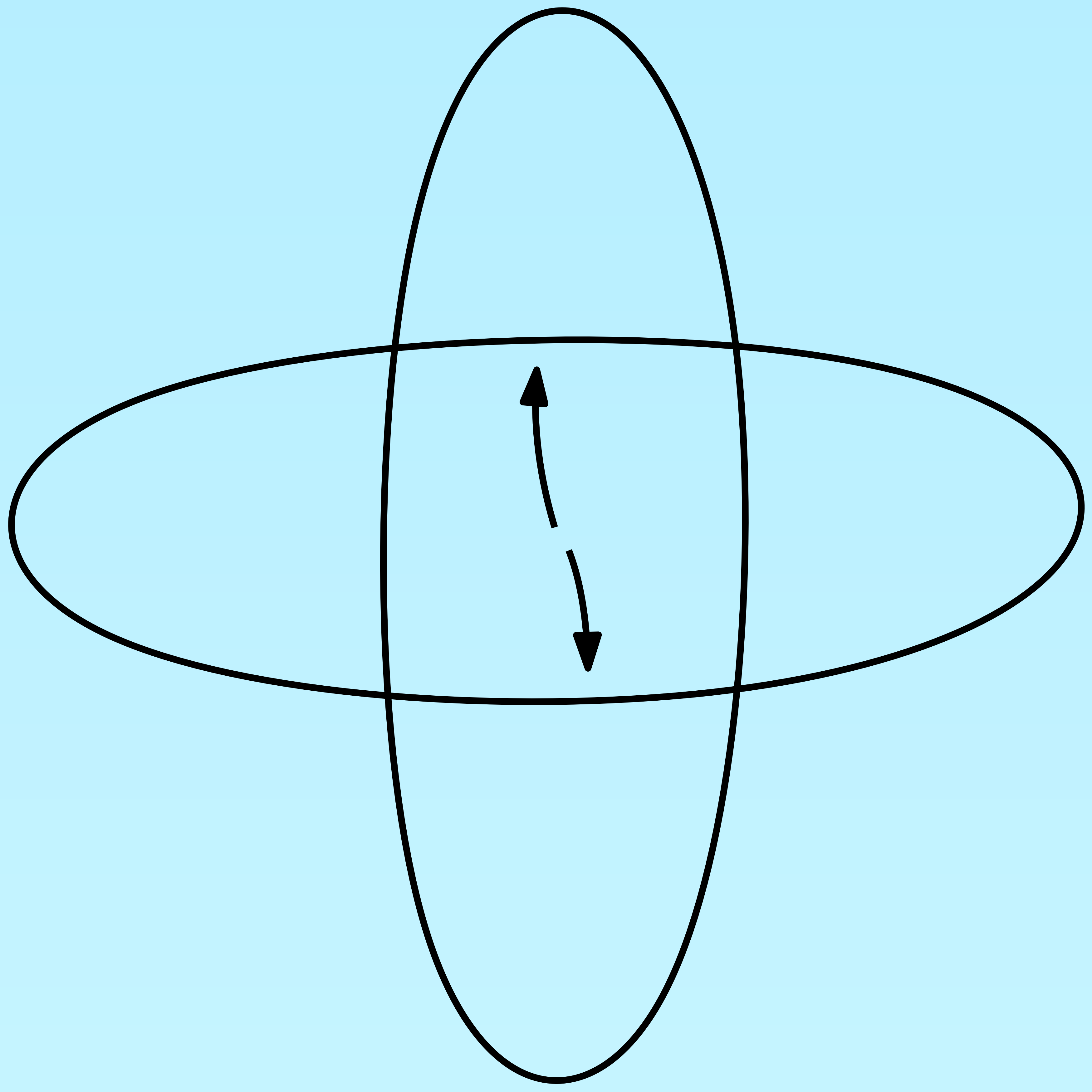


Consider an arrangement of simple closed curves in the plane.

A face of the arrangement is popular if it is adjacent to the same curve twice.

Problem. Can we add a curve to the arrangement, such that the resulting arrangement has no popular faces?

Not always!

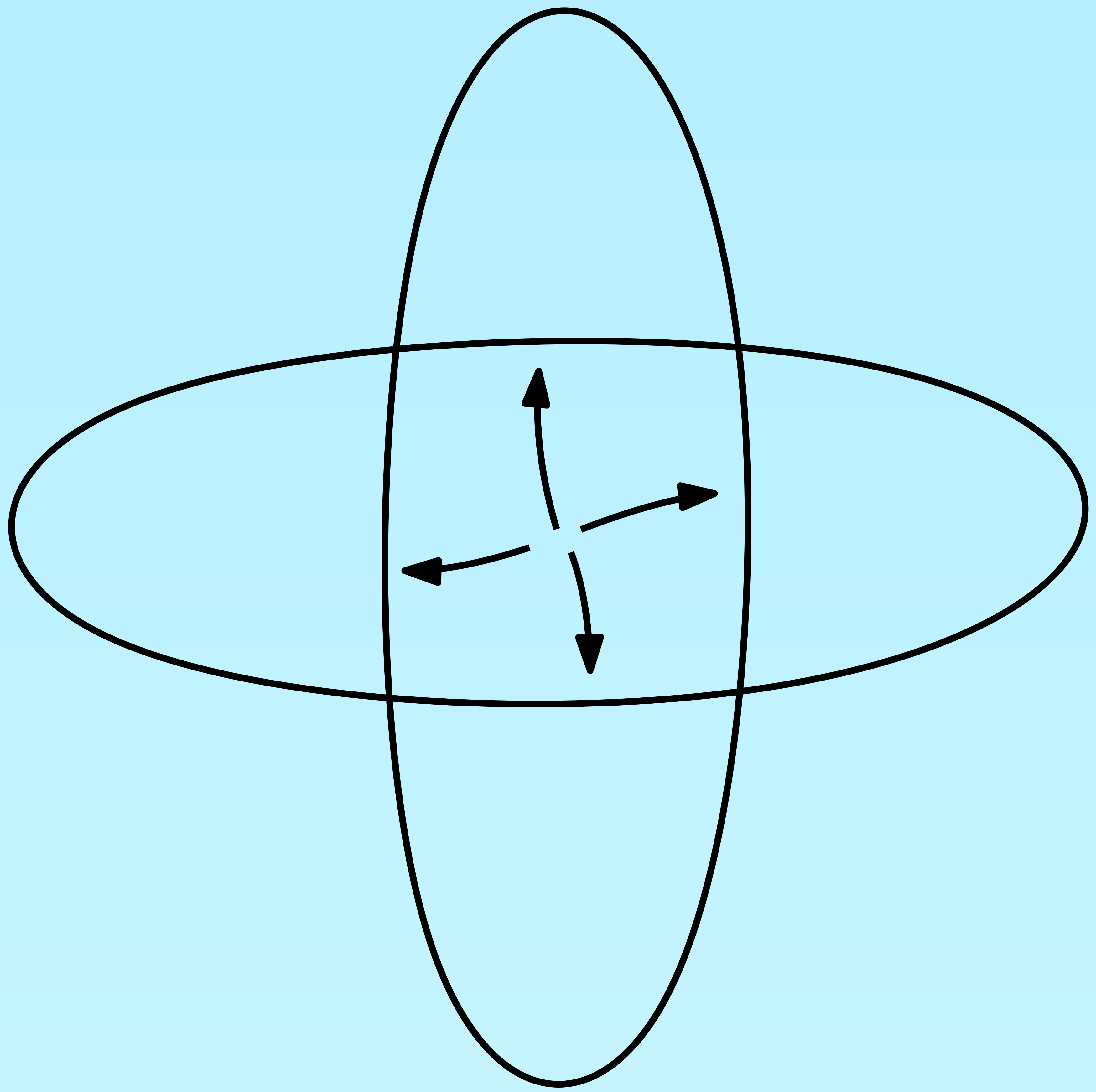


Consider an arrangement of simple closed curves in the plane.

A face of the arrangement is popular if it is adjacent to the same curve twice.

Problem. Can we add a curve to the arrangement, such that the resulting arrangement has no popular faces?

Not always!

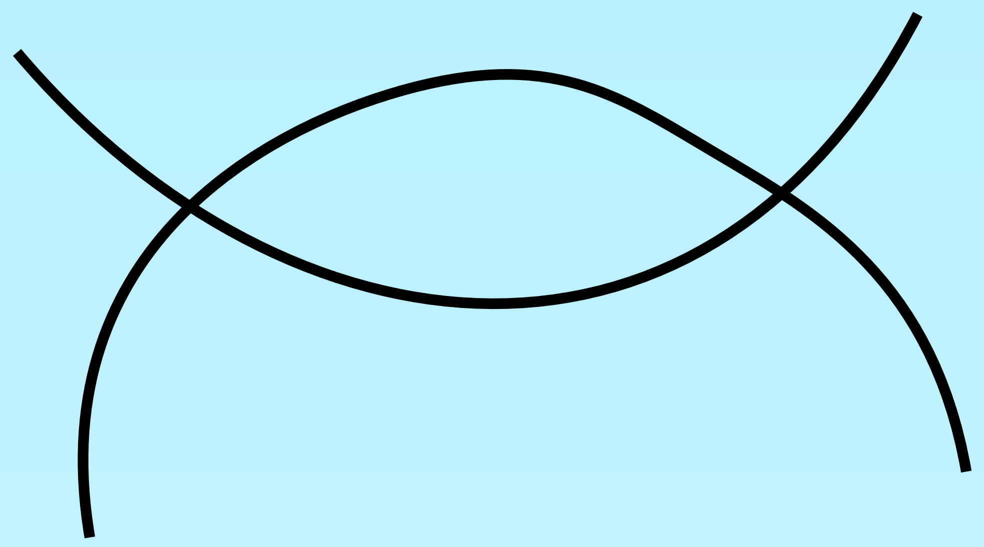


Consider an arrangement of simple closed curves in the plane.

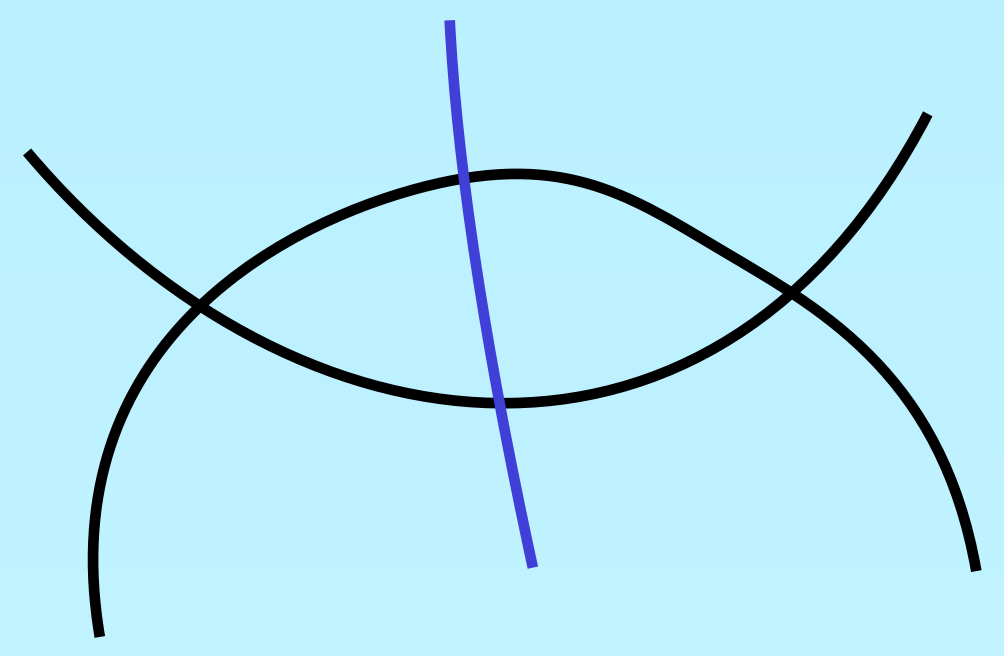
A face of the arrangement is popular if it is adjacent to the same curve twice.

Problem. Can we add a curve to the arrangement, such that the resulting arrangement has no popular faces?

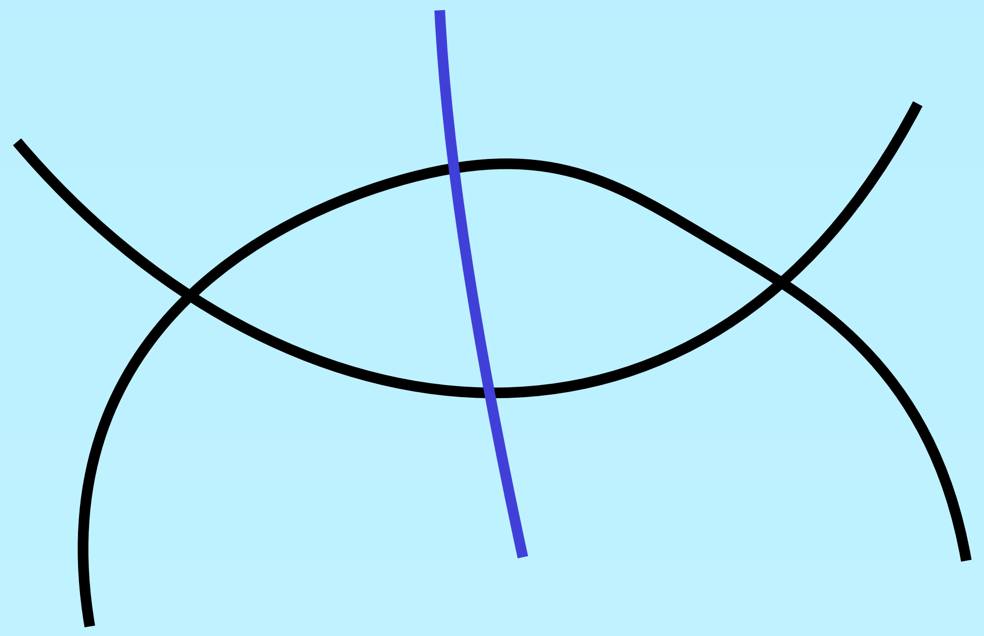
Not always!



Popular faces present in the input restrict the possible route of the new curve.

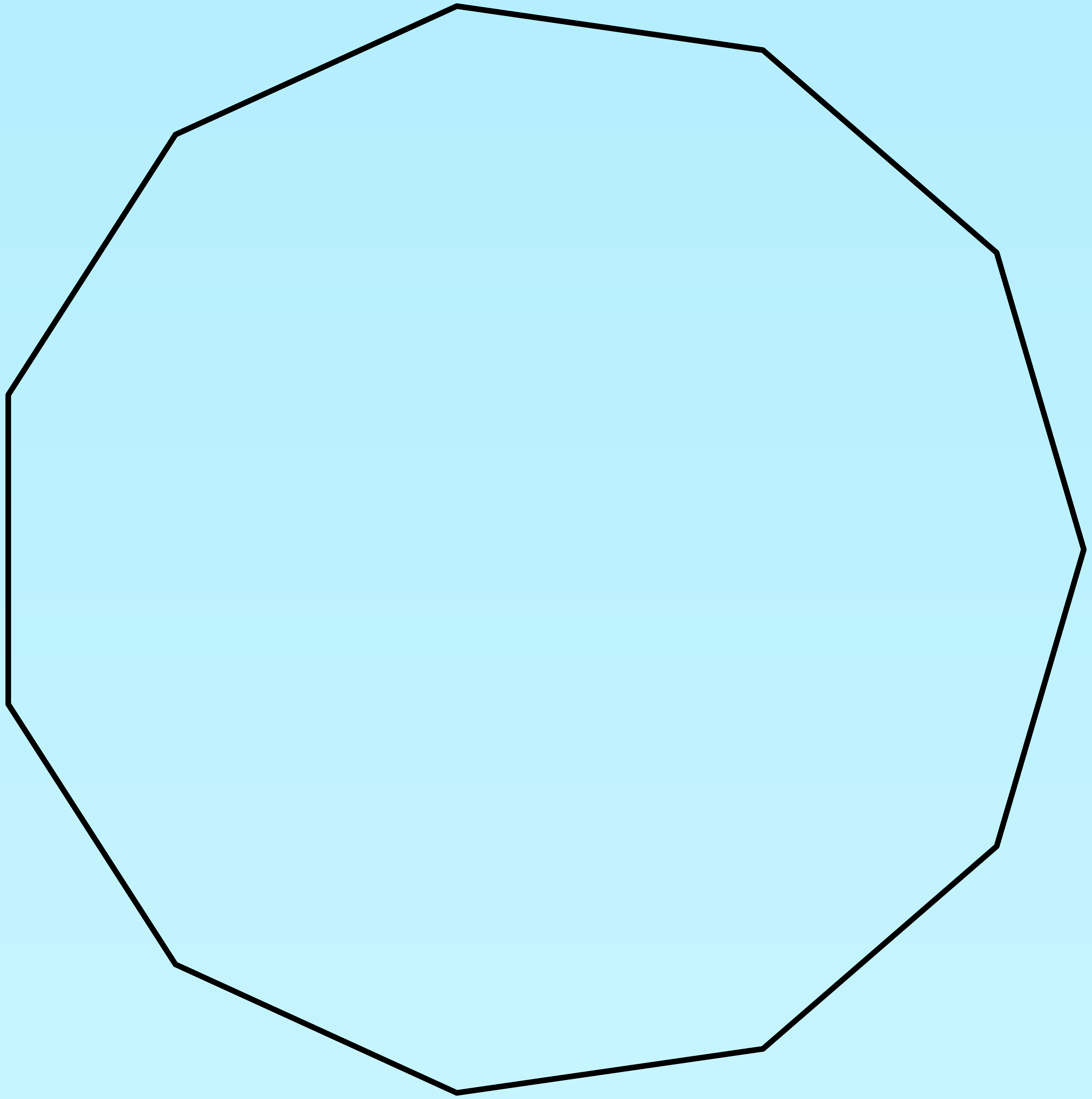


Popular faces present in the input restrict the possible route of the new curve.



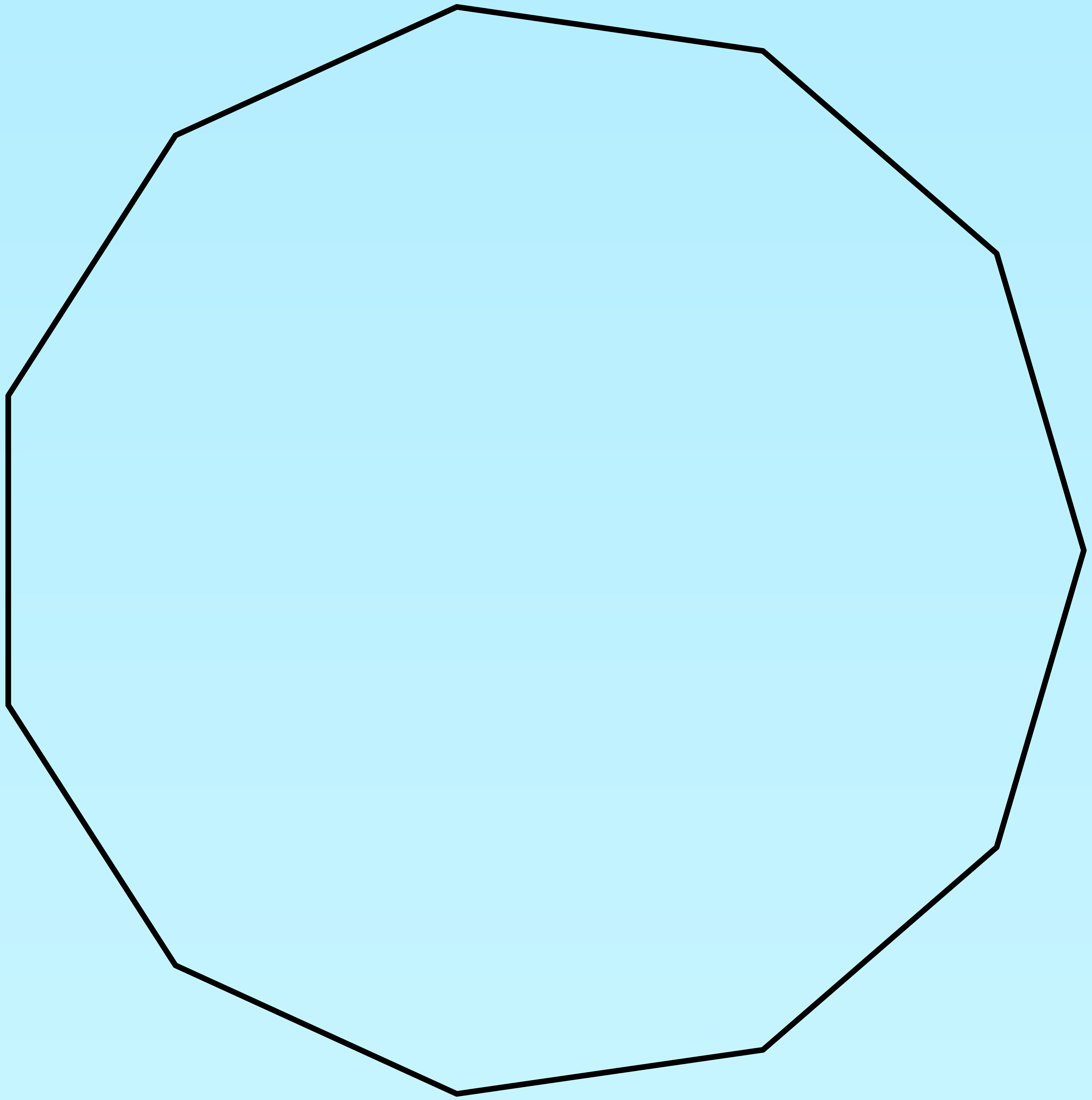
Popular faces present in the input restrict the possible route of the new curve.

Consider one popular face.



Popular faces present in the input restrict the possible route of the new curve.

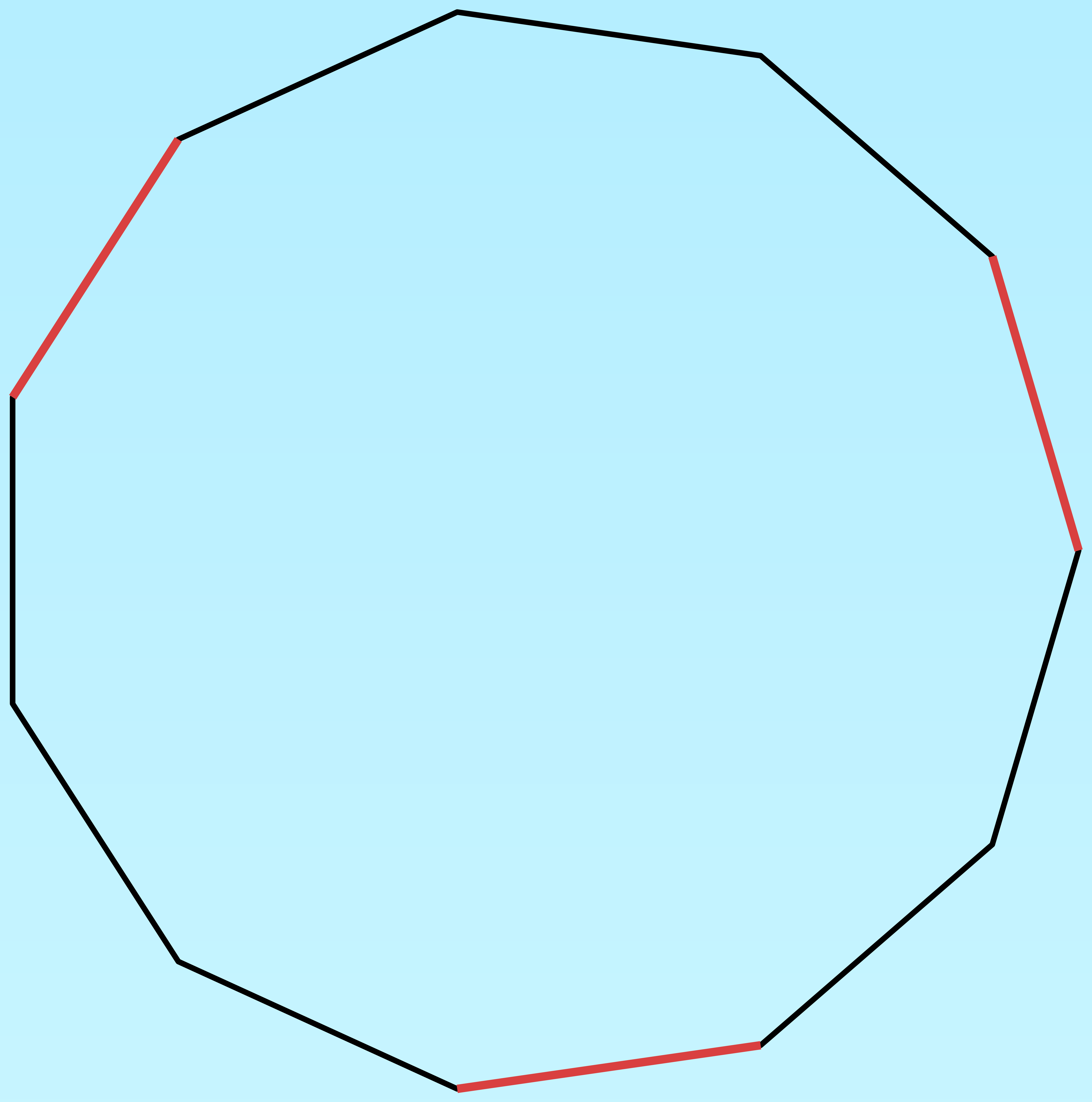
Consider one popular face.



Popular faces present in the input restrict the possible route of the new curve.

Consider one popular face.

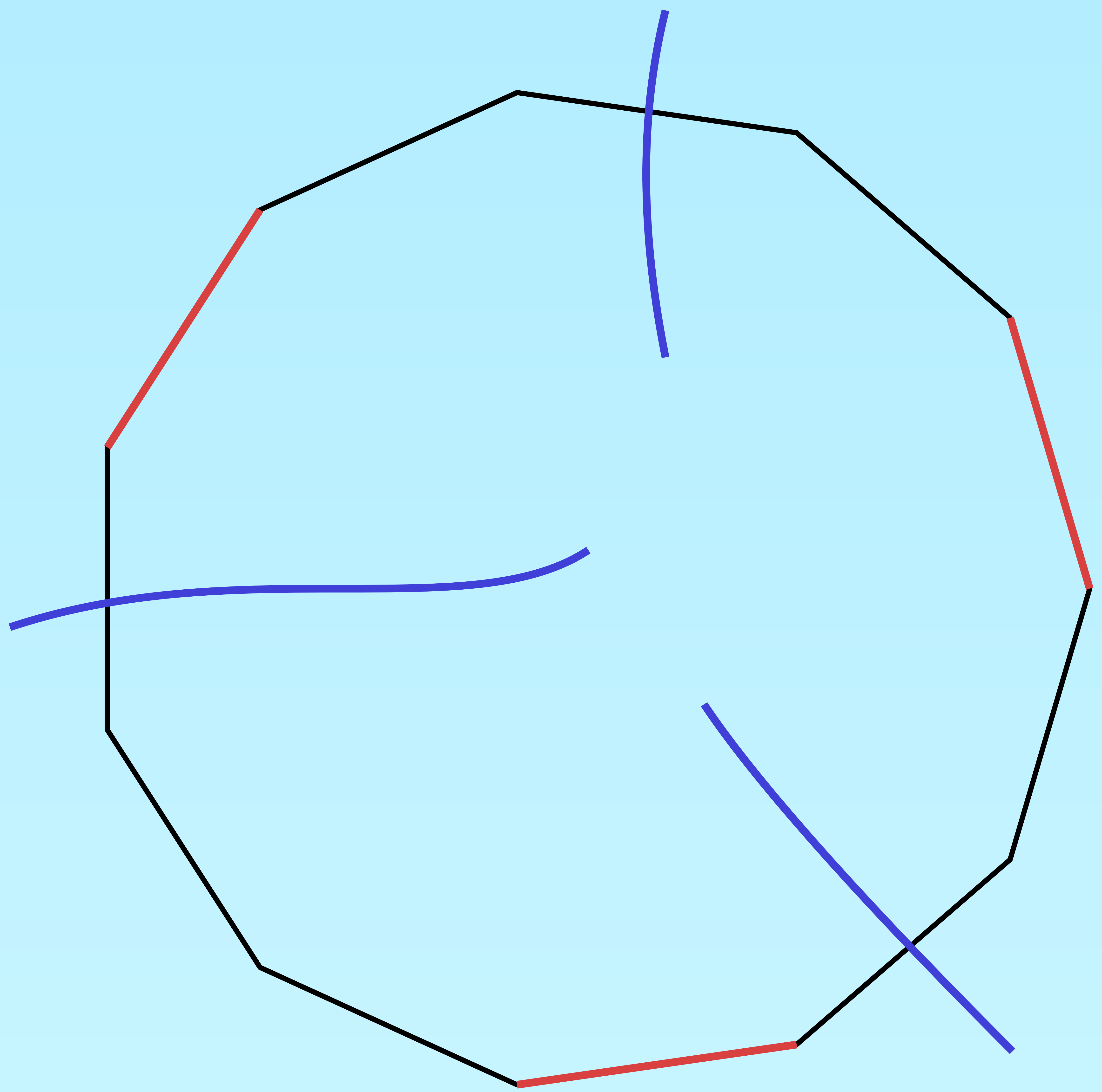
Observation: No curve can visit the face more than twice.



Popular faces present in the input restrict the possible route of the new curve.

Consider one popular face.

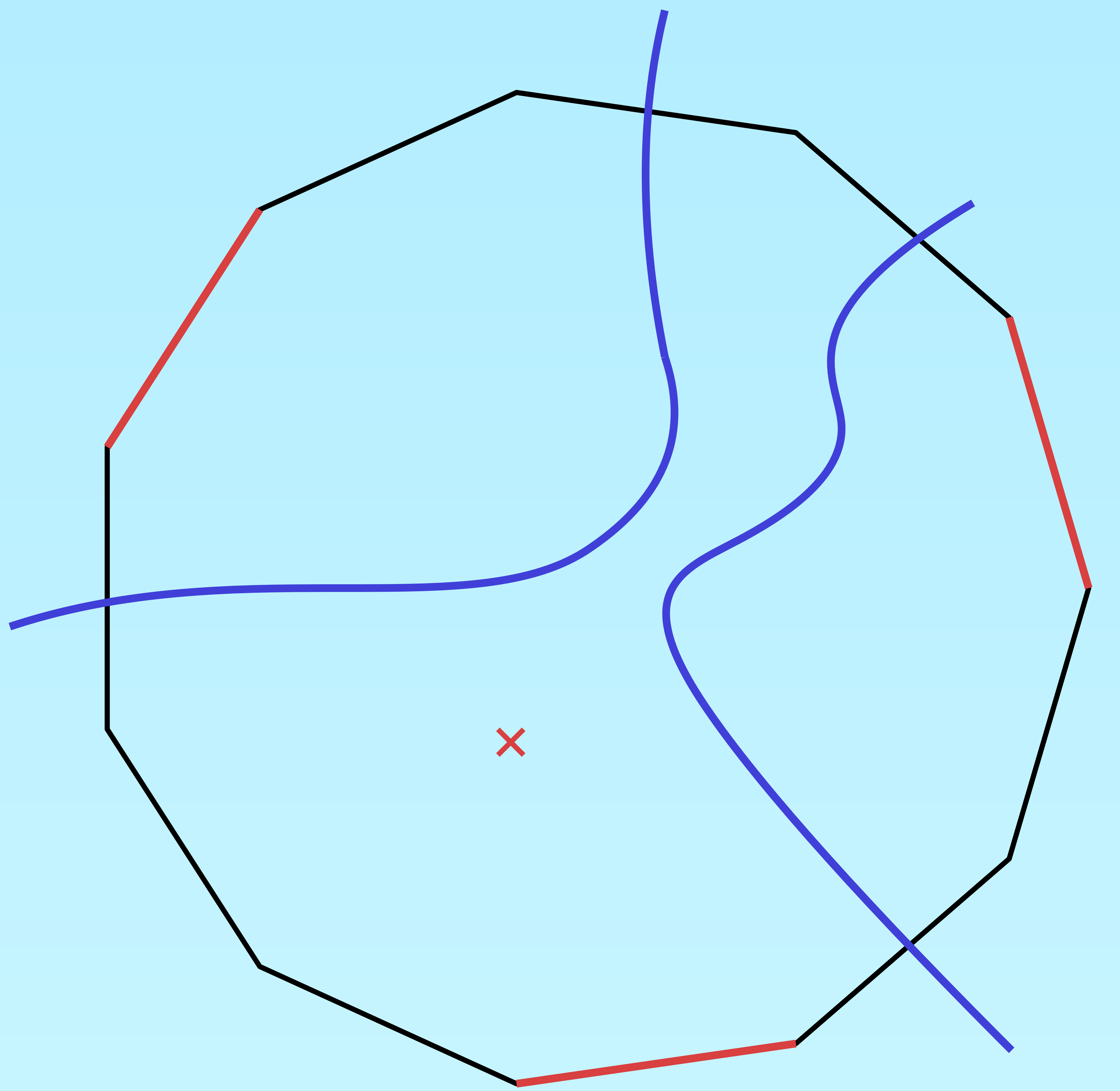
Observation: No curve can visit the face more than twice.



Popular faces present in the input restrict the possible route of the new curve.

Consider one popular face.

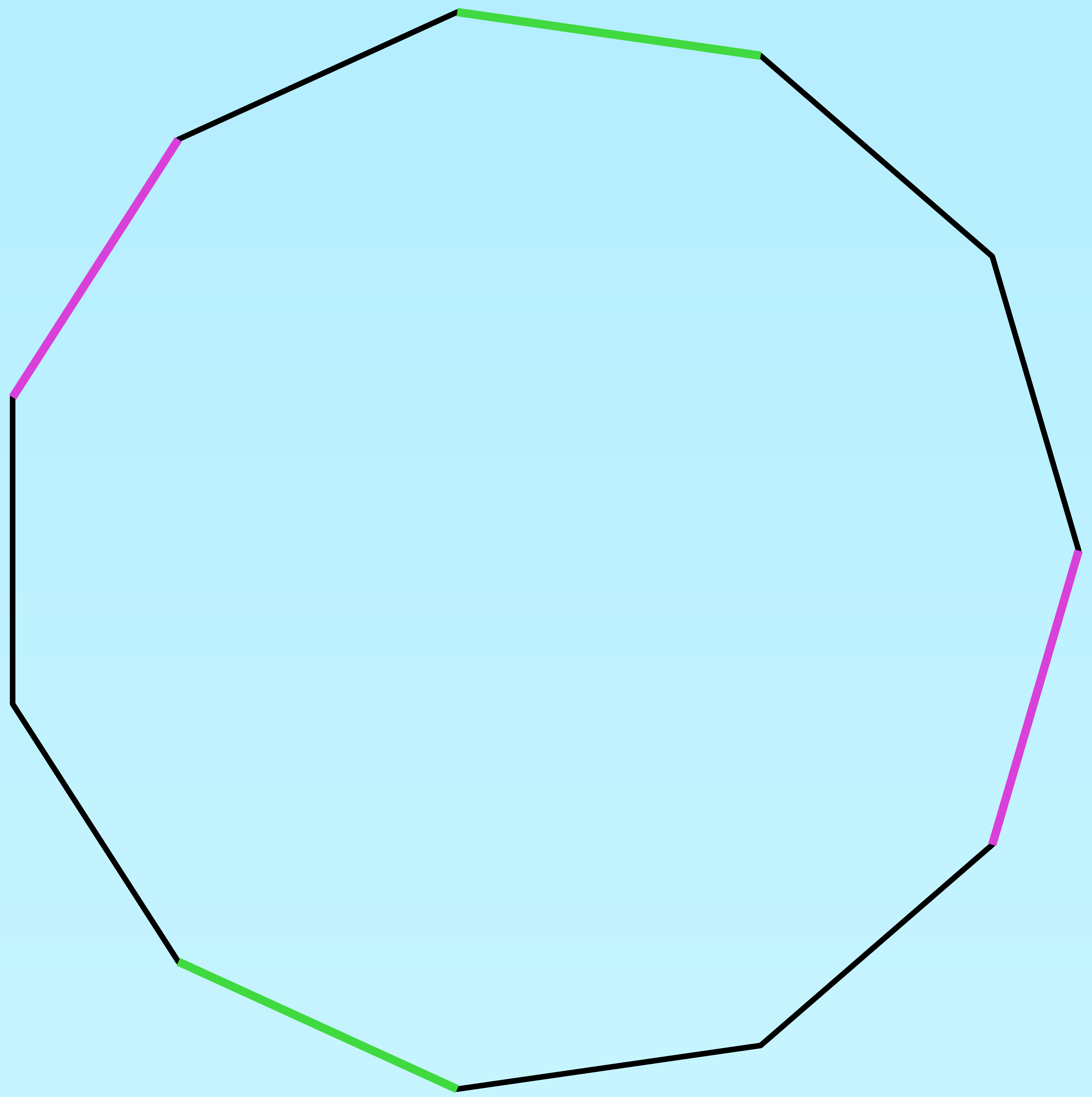
Observation: No curve can visit the face more than twice.



Popular faces present in the input restrict the possible route of the new curve.

Consider one popular face.

Observation: No curve can visit the face more than twice.



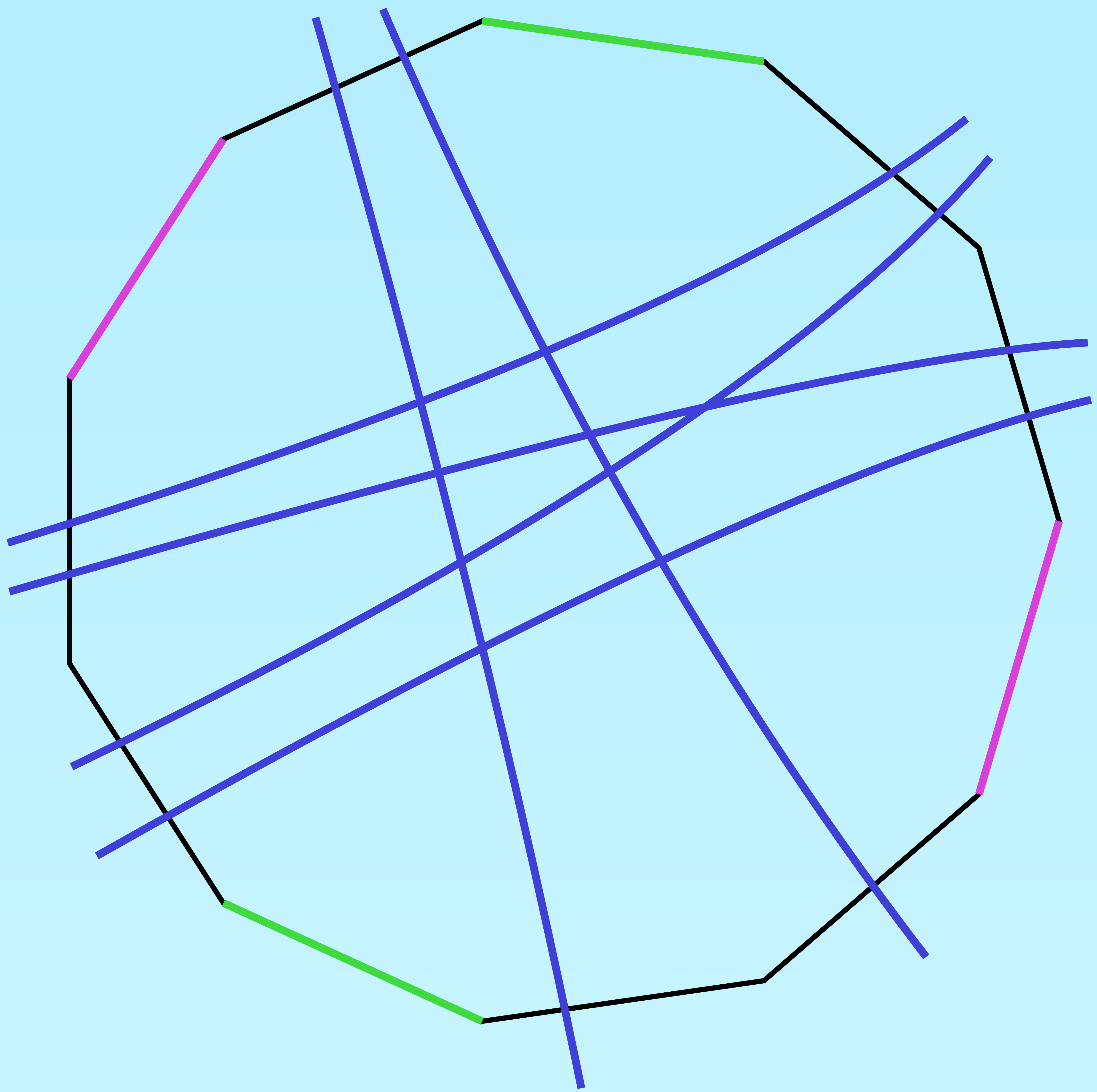
Popular faces present in the input restrict the possible route of the new curve.

Consider one popular face.

Observation: No curve can visit the face more than twice.

The new curve must separate each pair.

We can enumerate all possible resolutions.



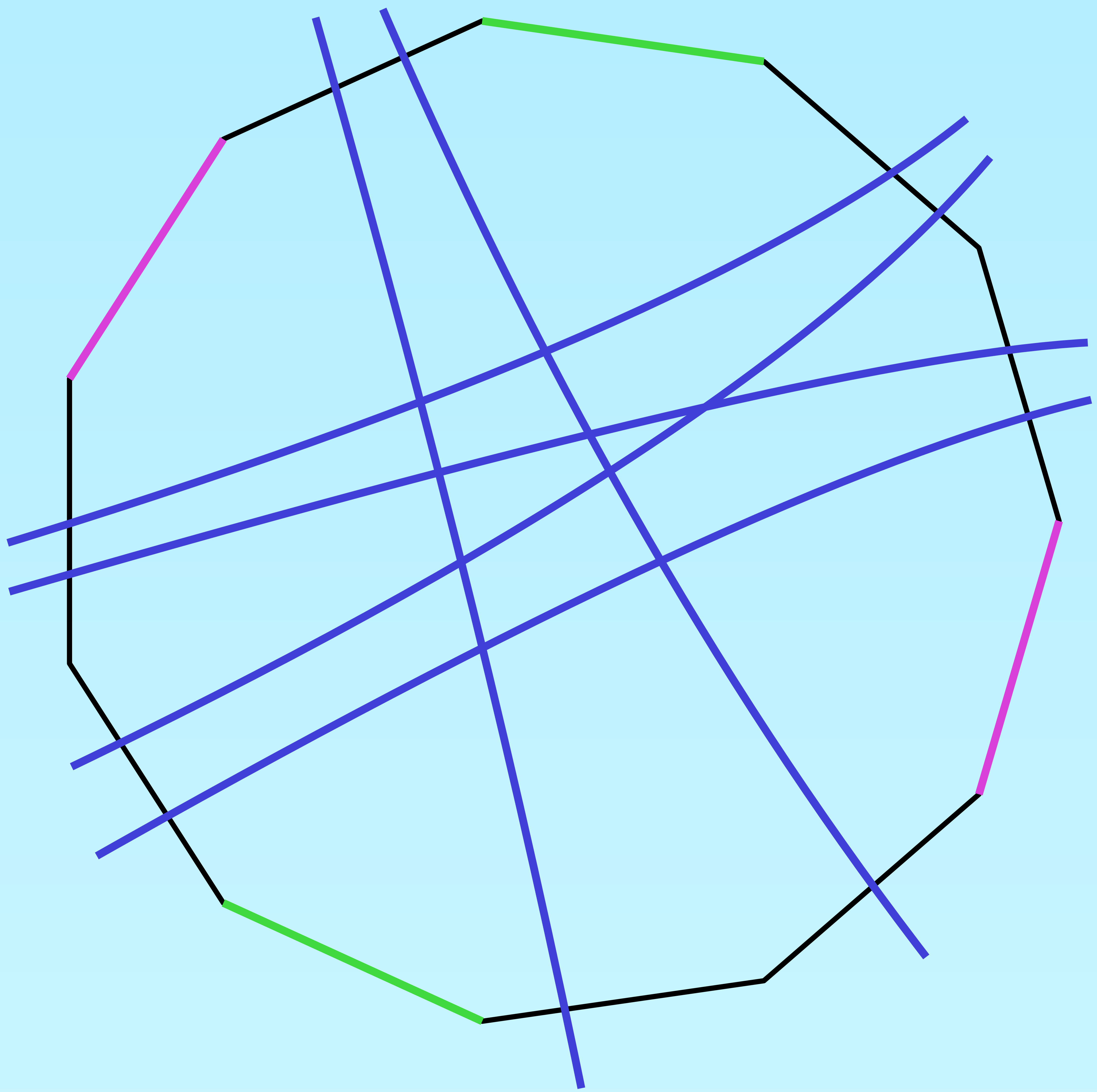
Popular faces present in the input restrict the possible route of the new curve.

Consider one popular face.

Observation: No curve can visit the face more than twice.

The new curve must separate each pair.

We can enumerate all possible resolutions.



Popular faces present in the input restrict the possible route of the new curve.

Consider one popular face.

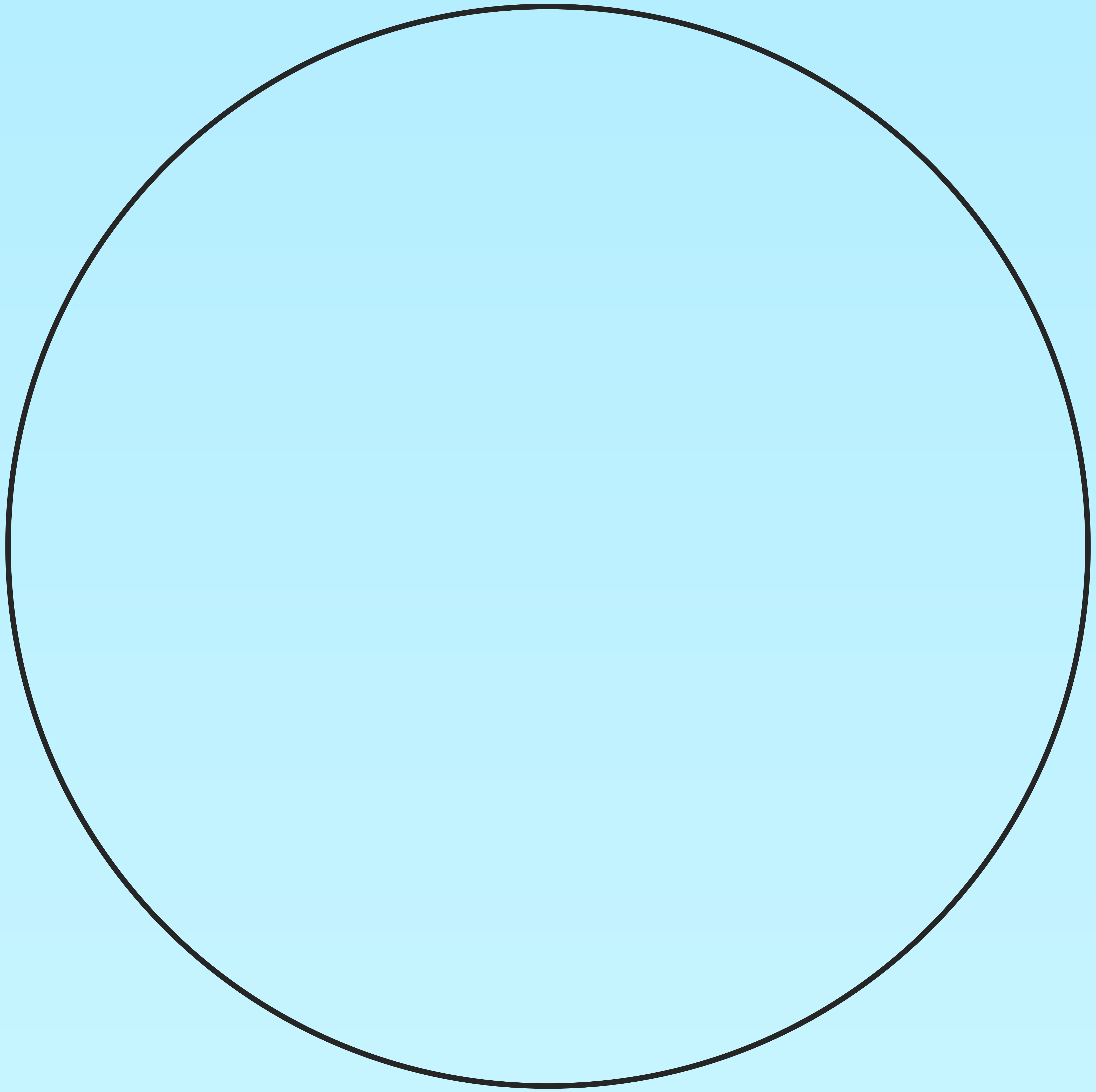
Observation: No curve can visit the face more than twice.

The new curve must separate each pair.

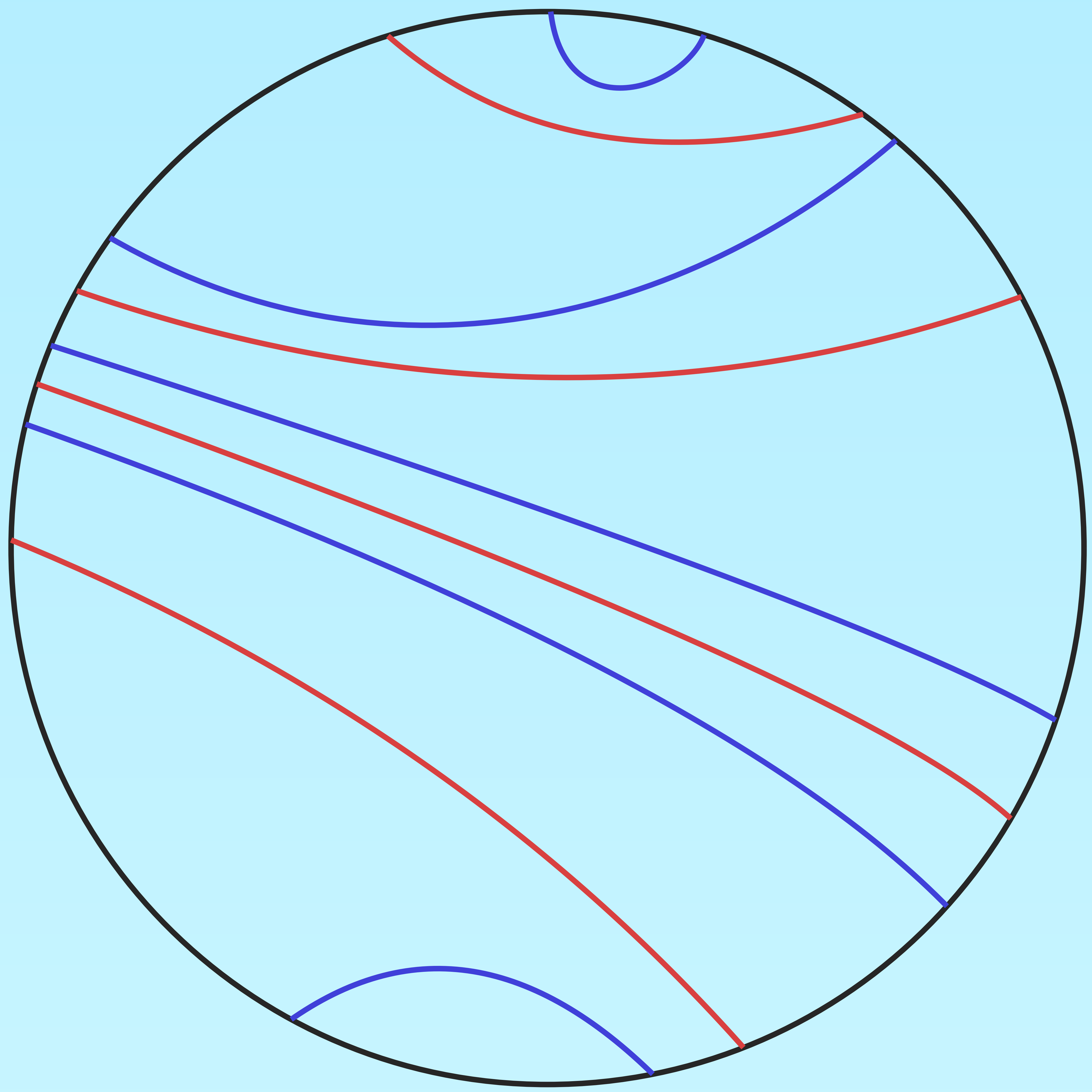
We can enumerate all possible resolutions.

At most $O(n^2)$ options!

Just for fun: what if we allowed to insert 2 extra curves...?



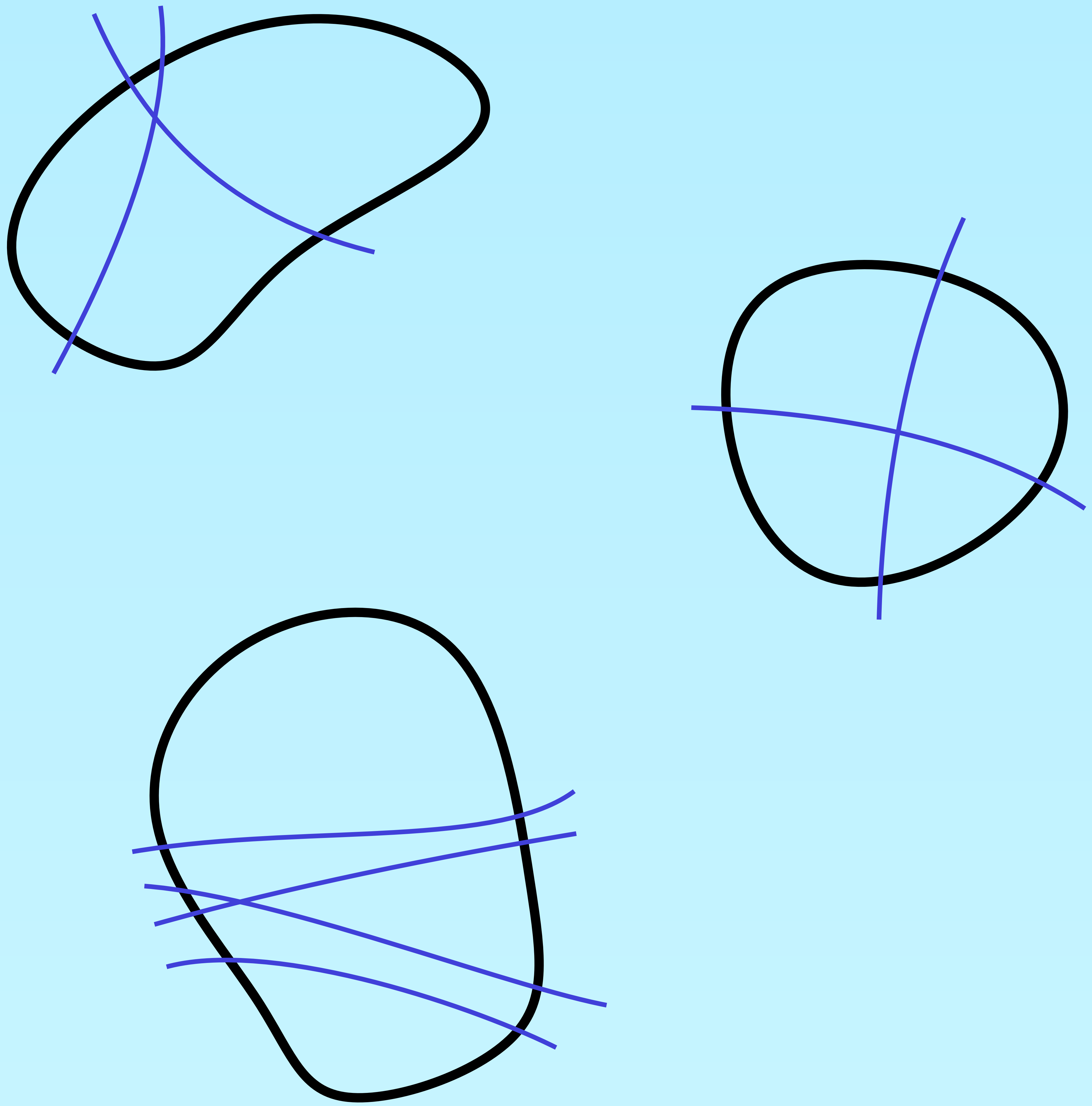
Just for fun: what if we allowed to insert 2 extra curves...?



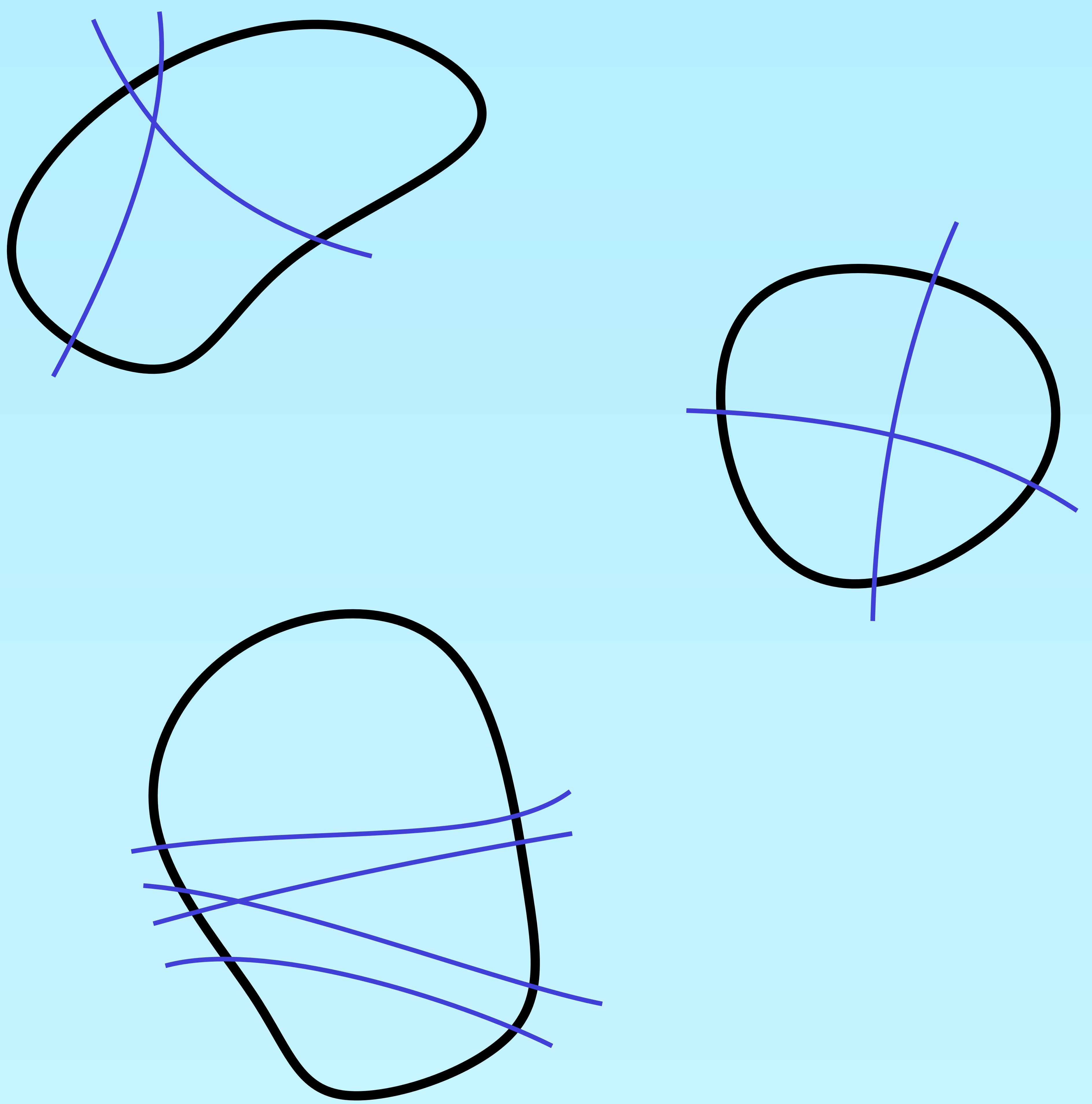
Just for fun: what if we allowed to insert 2 extra curves...?

If the curves cooperate, each face can be visited many times!

Task: We must choose one resolving piece from each popular face, and then connect them.

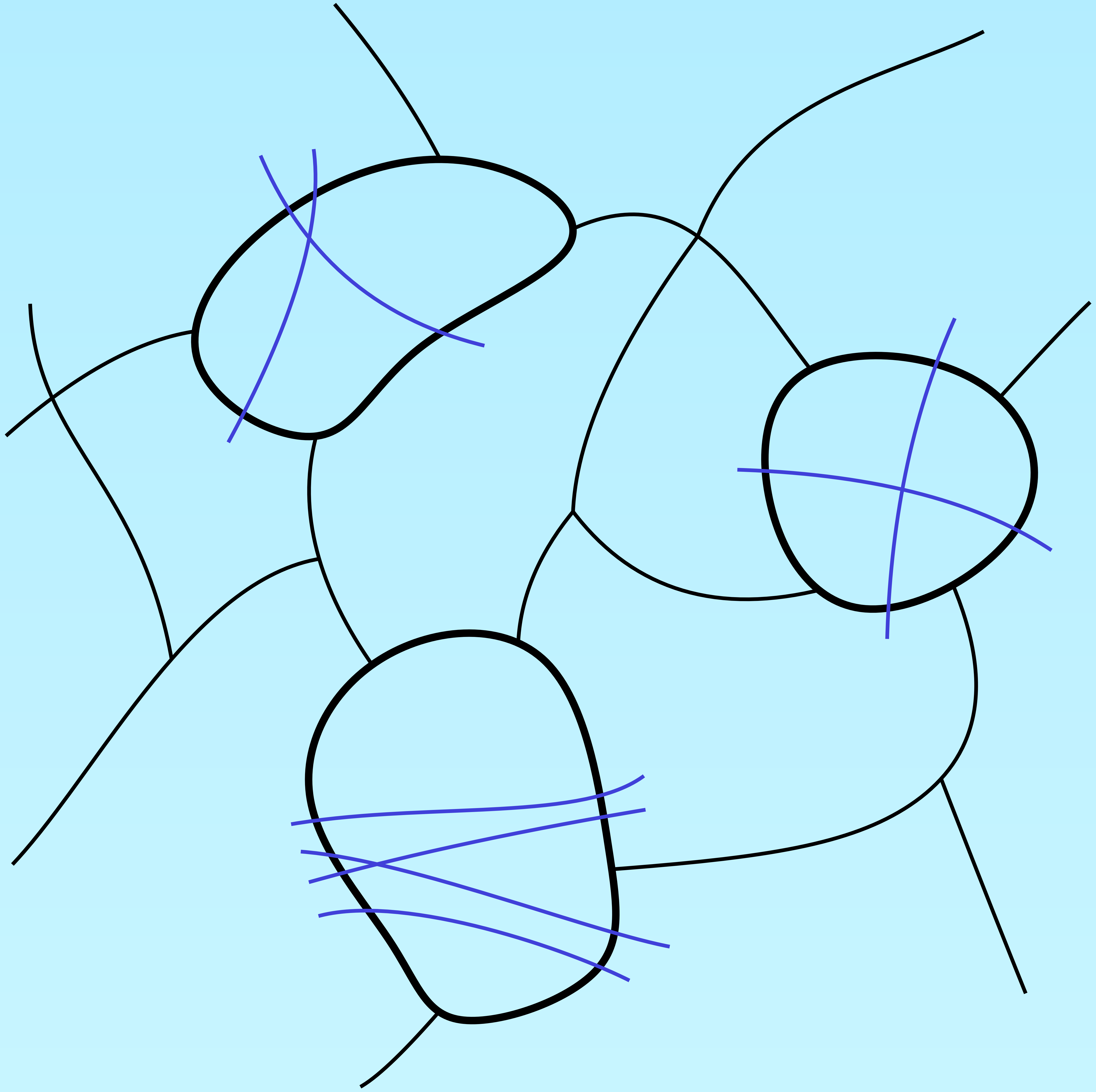


Task: We must choose one resolving piece from each popular face, and then connect them.



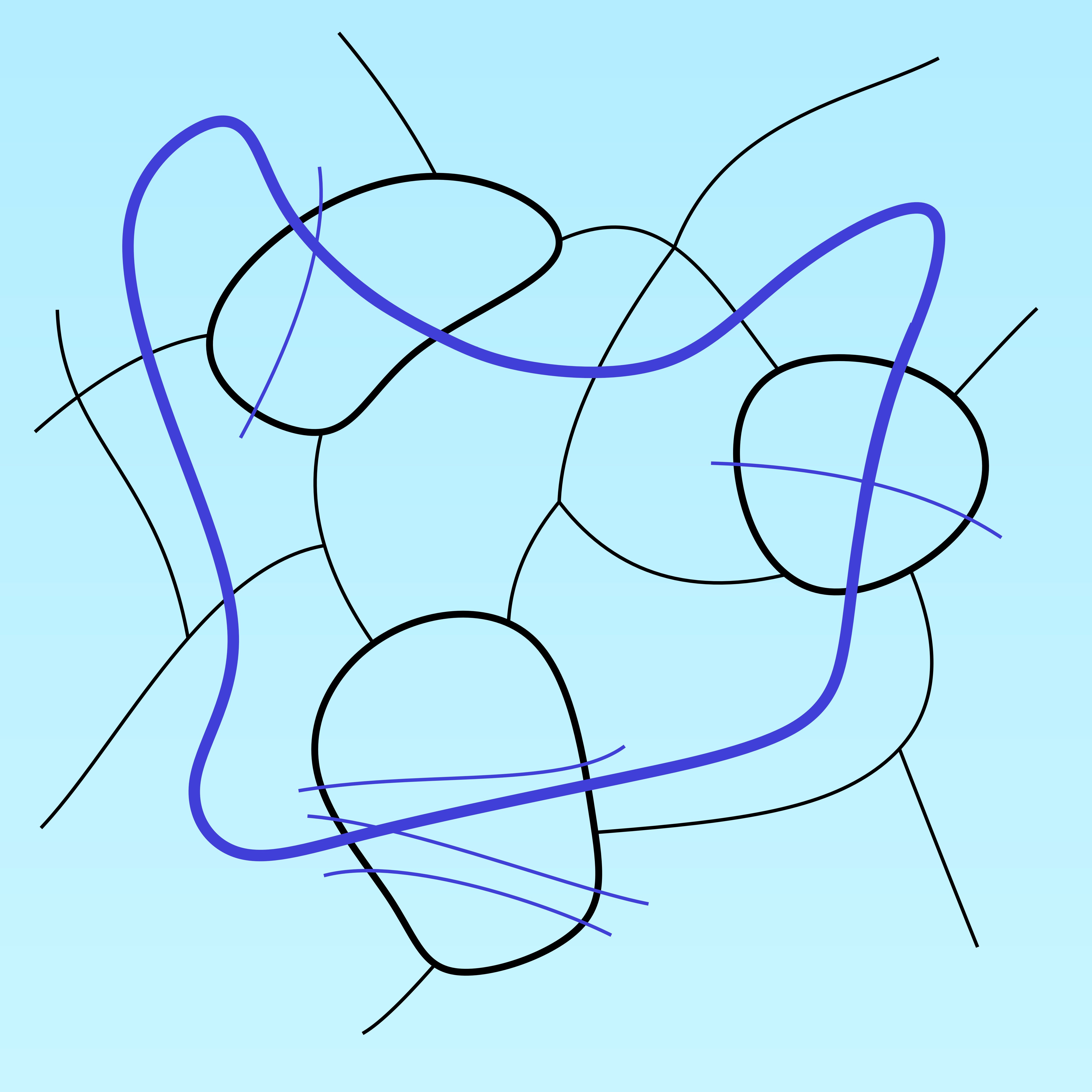
Task: We must choose one resolving piece from each popular face, and then connect them.

We can only go through each cell of the arrangement at most once.



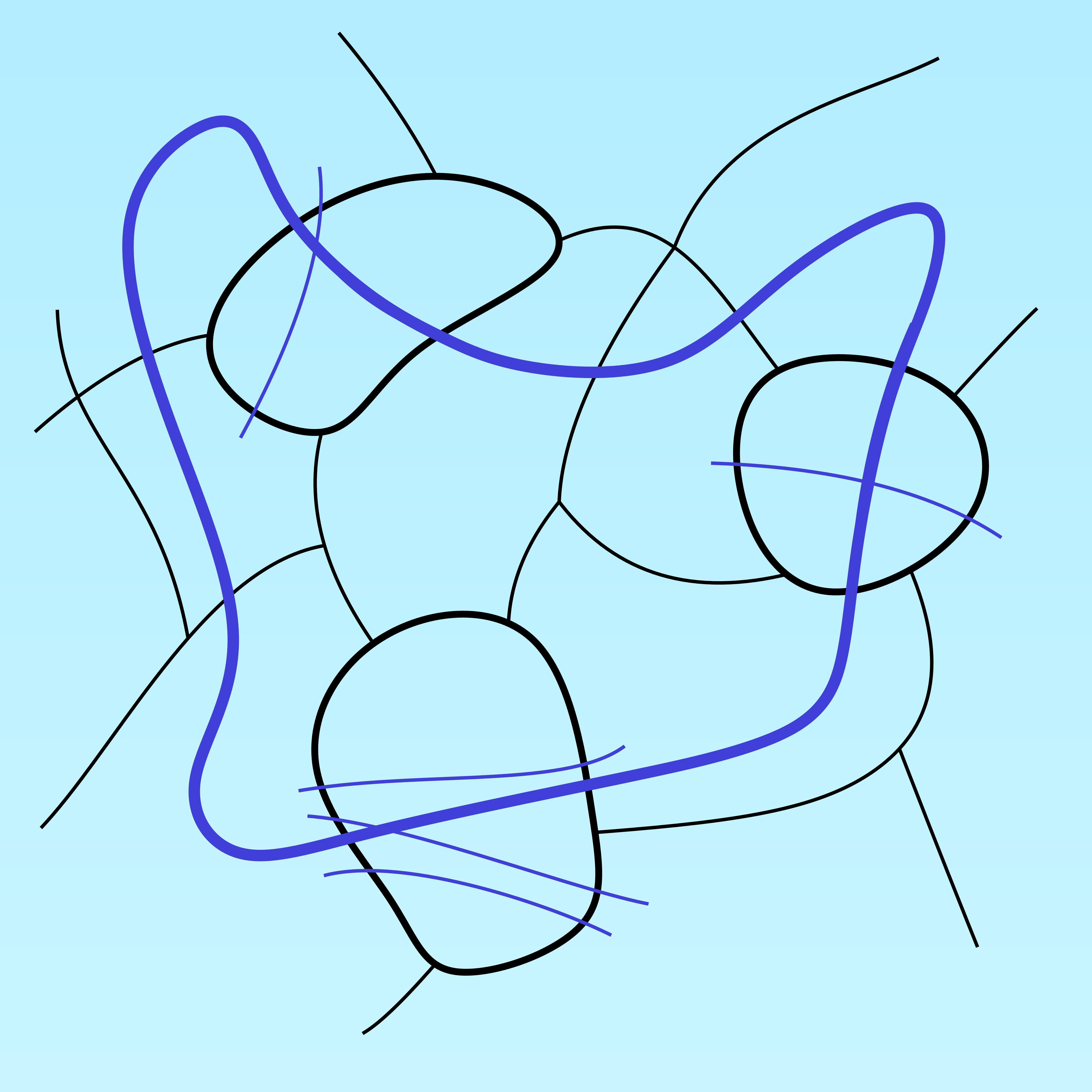
Task: We must choose one resolving piece from each popular face, and then connect them.

We can only go through each cell of the arrangement at most once.



Task: We must choose one resolving piece from each popular face, and then connect them.

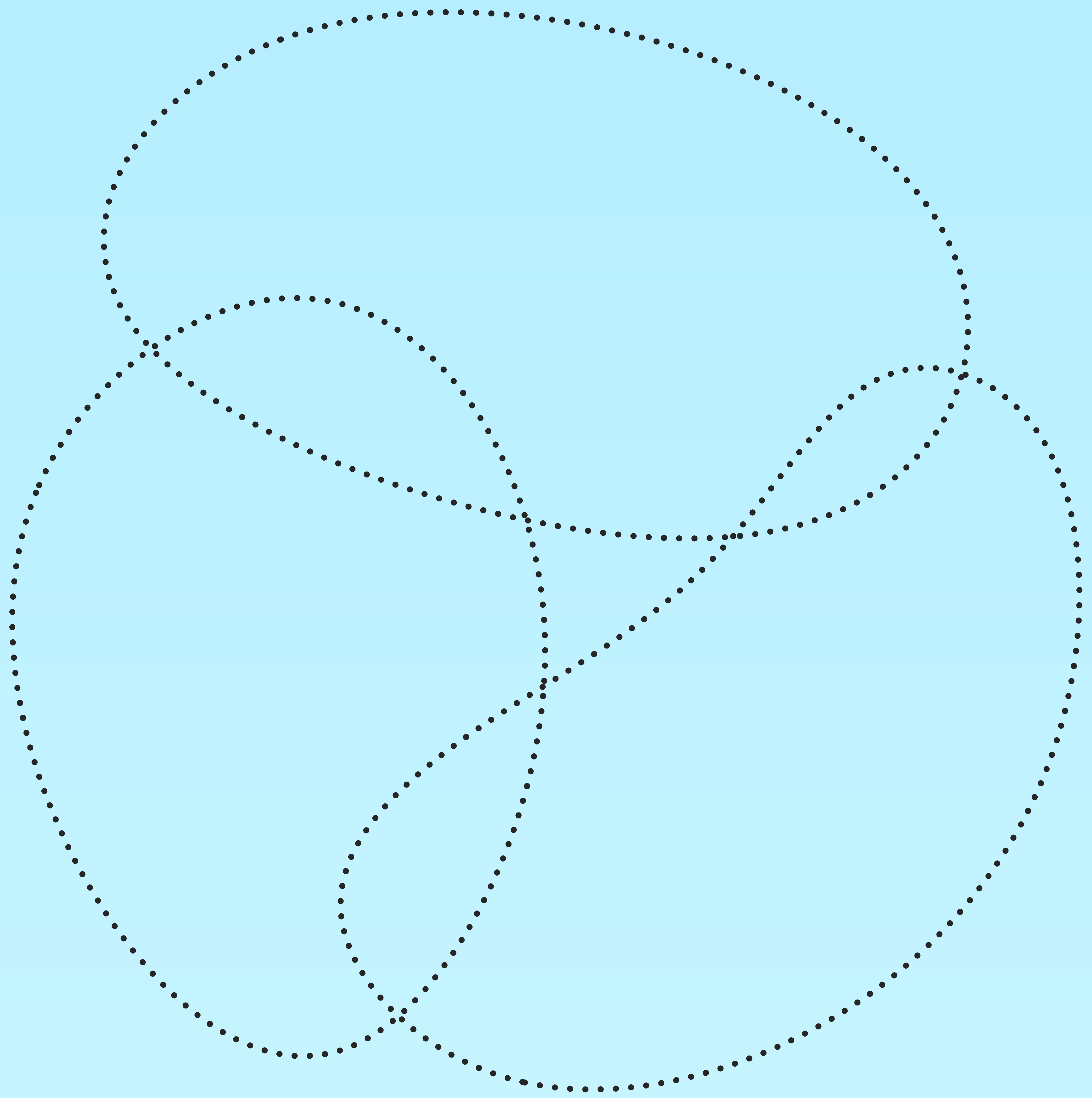
We can only go through each cell of the arrangement at most once.



Task: We must choose one resolving piece from each popular face, and then connect them.

We can only go through each cell of the arrangement at most once.

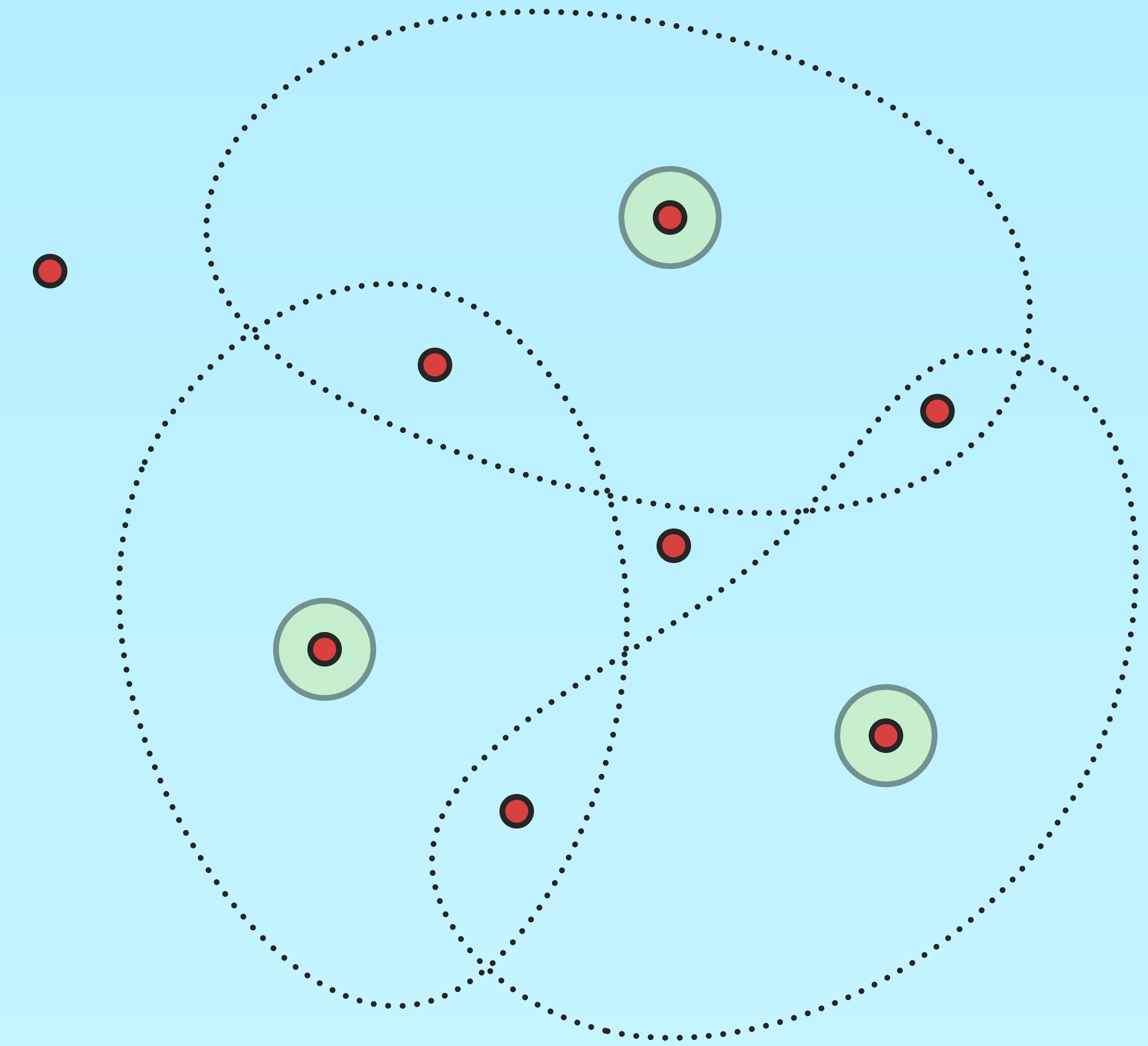
Look at the dual graph!



Task: We must choose one resolving piece from each popular face, and then connect them.

We can only go through each cell of the arrangement at most once.

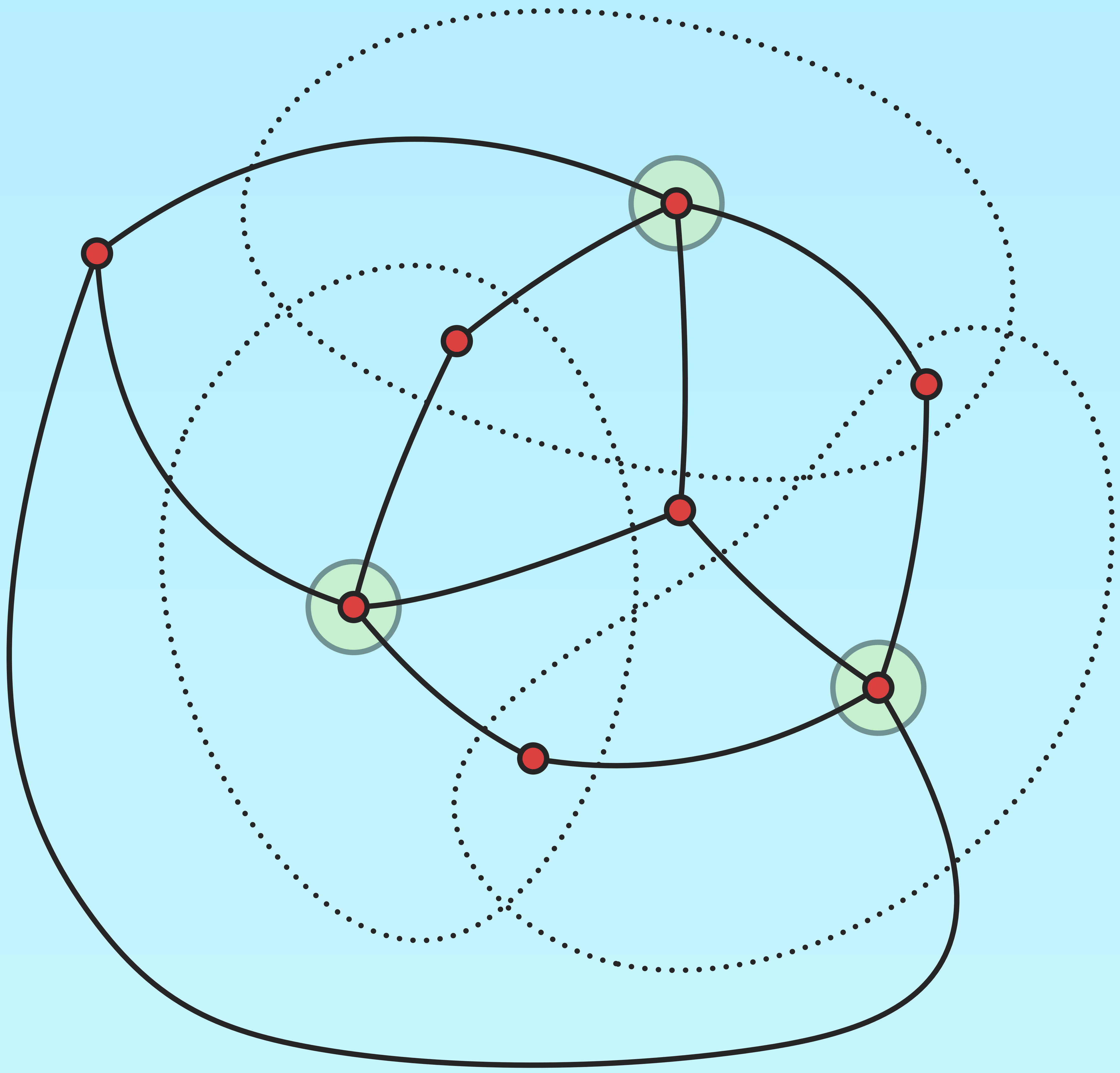
Look at the dual graph!



Task: We must choose one resolving piece from each popular face, and then connect them.

We can only go through each cell of the arrangement at most once.

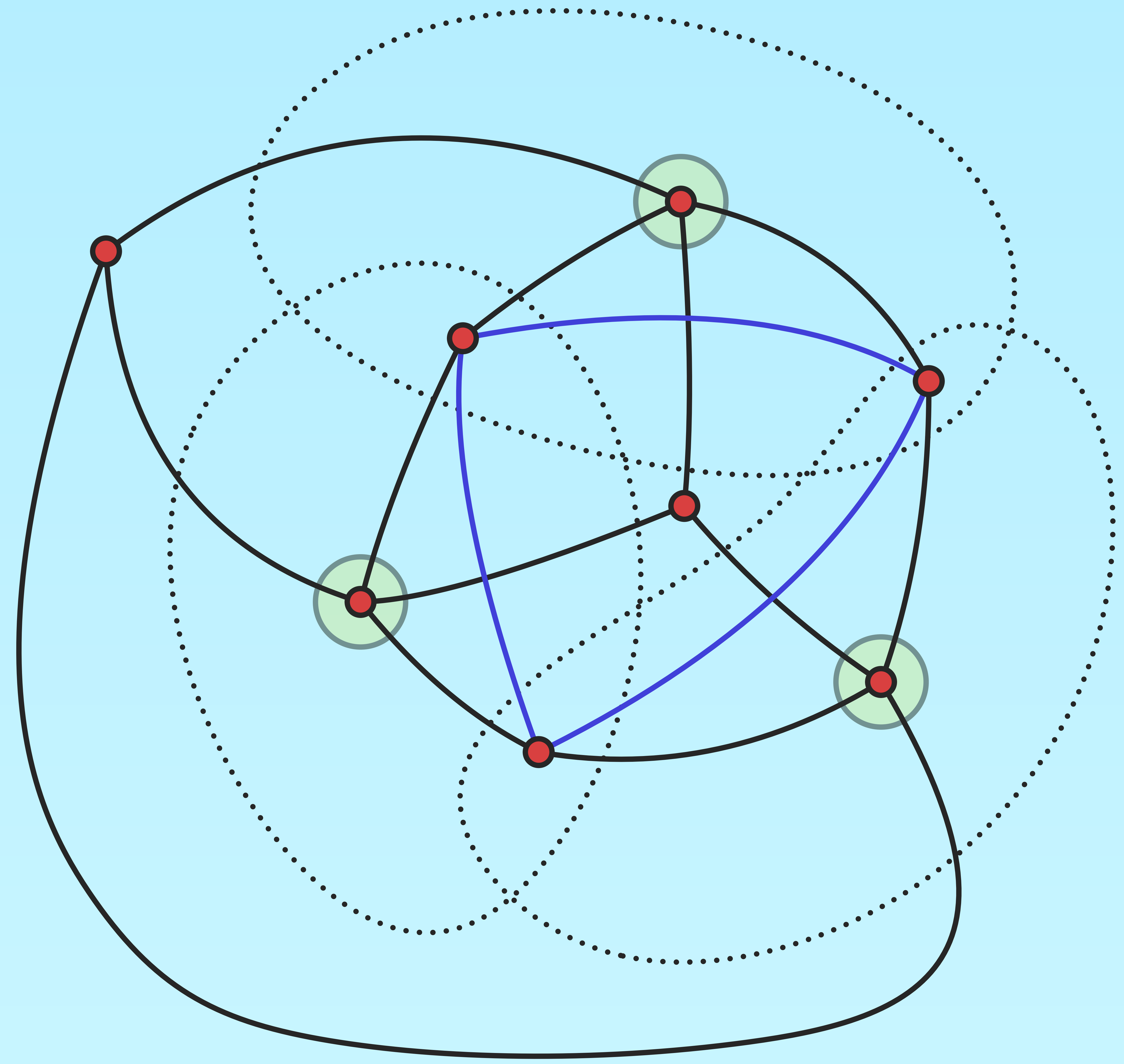
Look at the dual graph!



Task: We must choose one resolving piece from each popular face, and then connect them.

We can only go through each cell of the arrangement at most once.

Look at the dual graph!

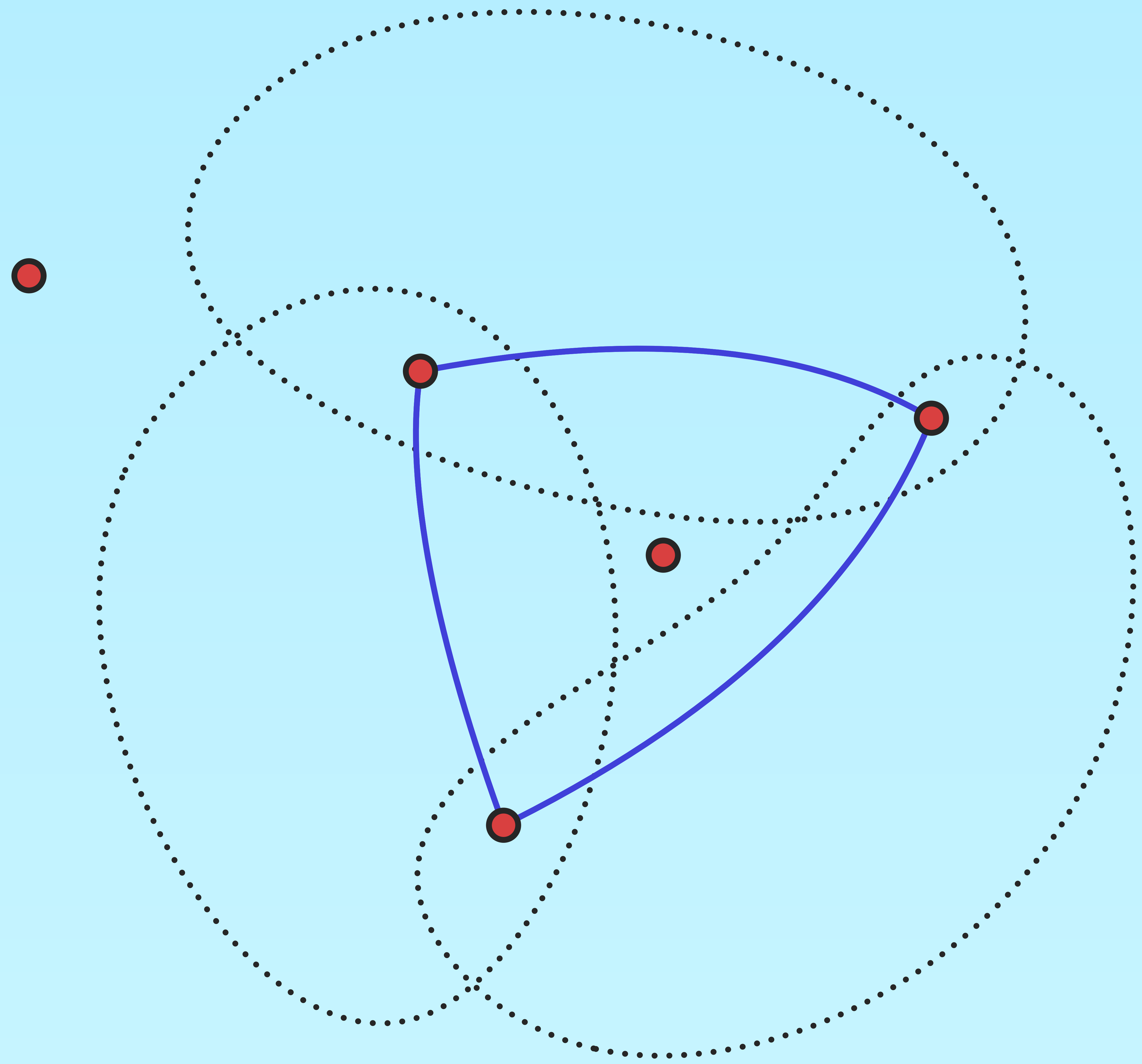


Task: We must choose one resolving piece from each popular face, and then connect them.

We can only go through each cell of the arrangement at most once.

Look at the dual graph!

Resolution pieces are also edges in the dual.



Task: We must choose one resolving piece from each popular face, and then connect them.

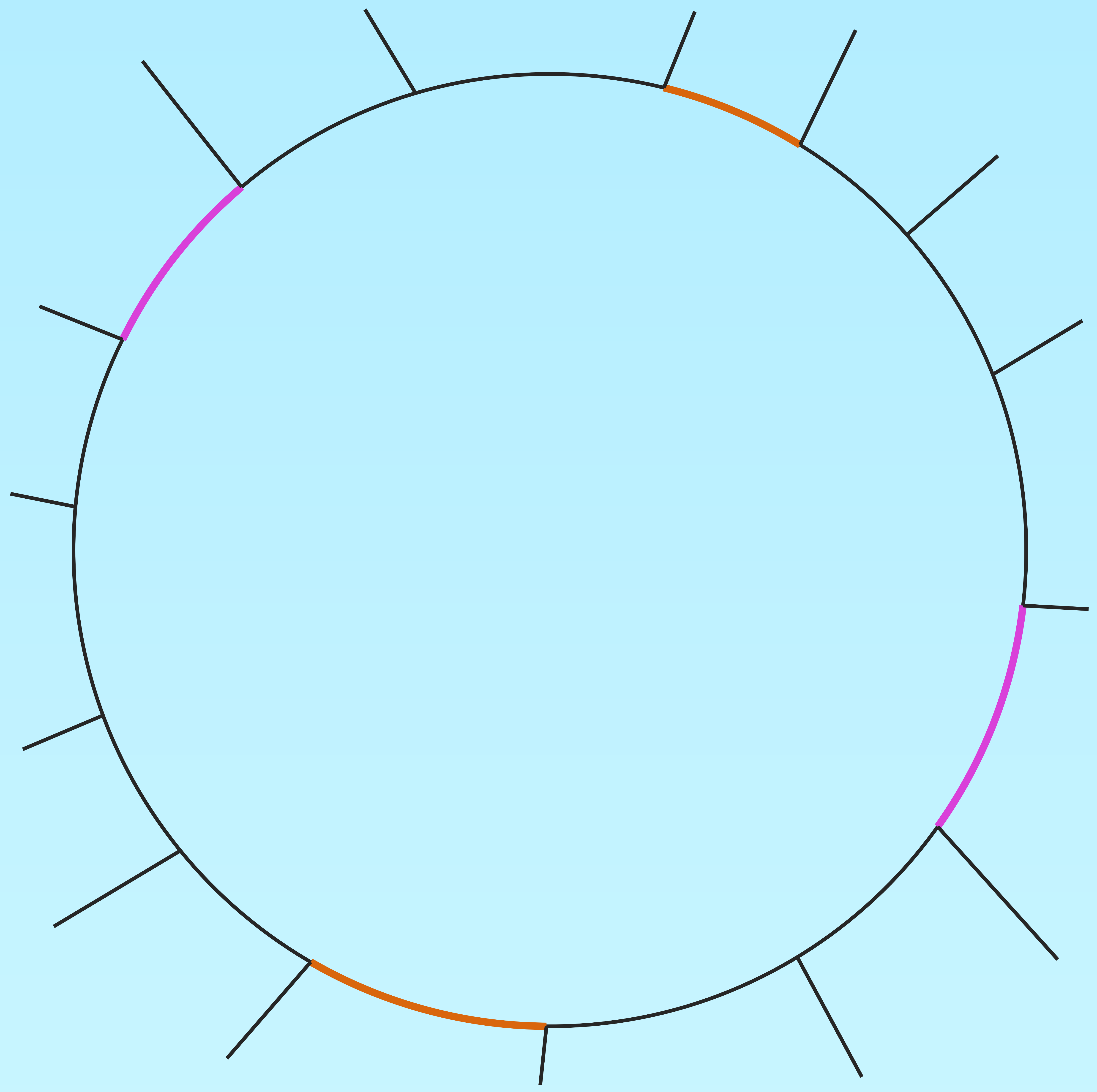
We can only go through each cell of the arrangement at most once.

Look at the dual graph!

Resolution pieces are also edges in the dual.

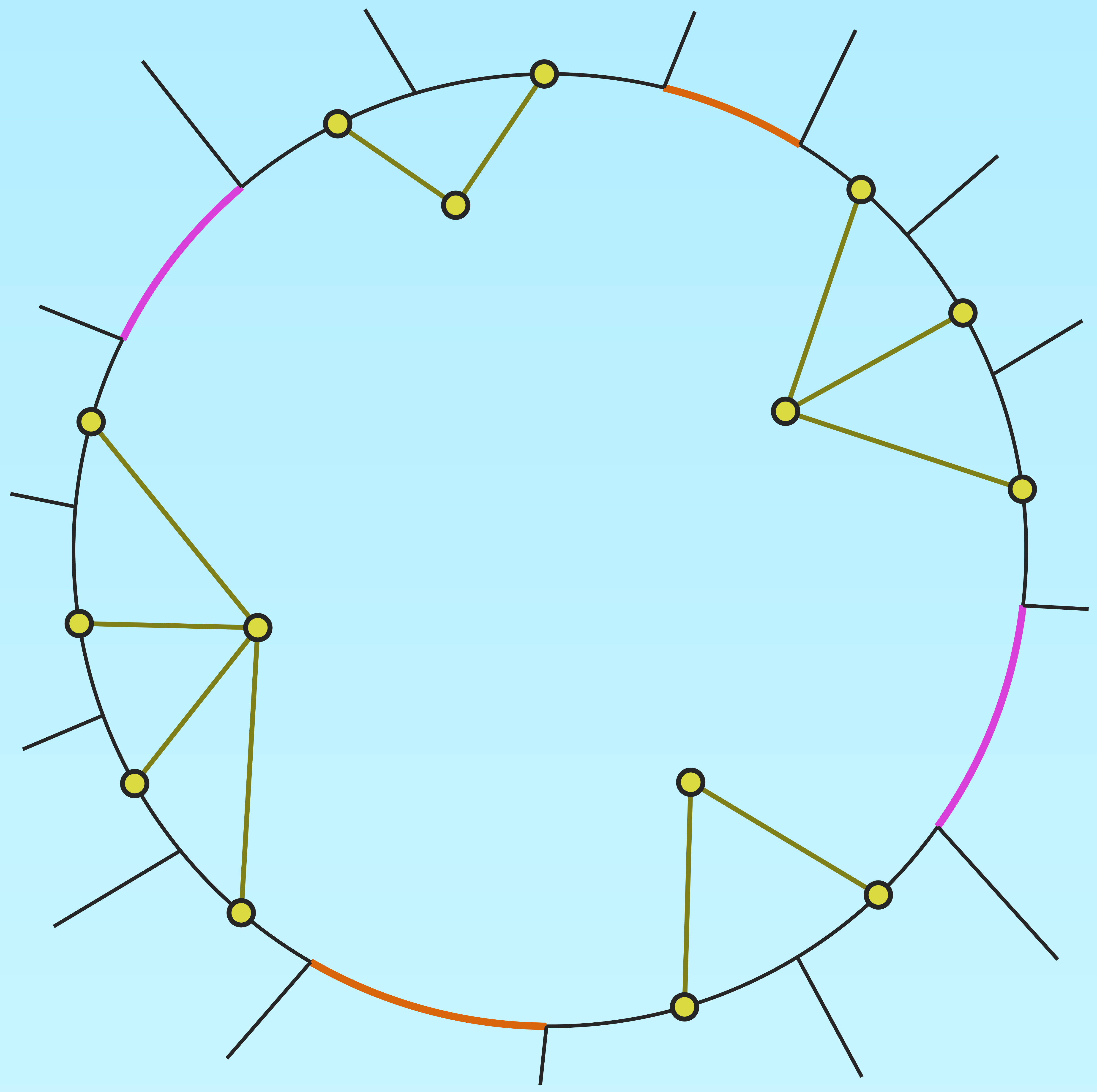
Actual popular faces can be removed.

Issue: This doesn't work for adjacent popular faces.



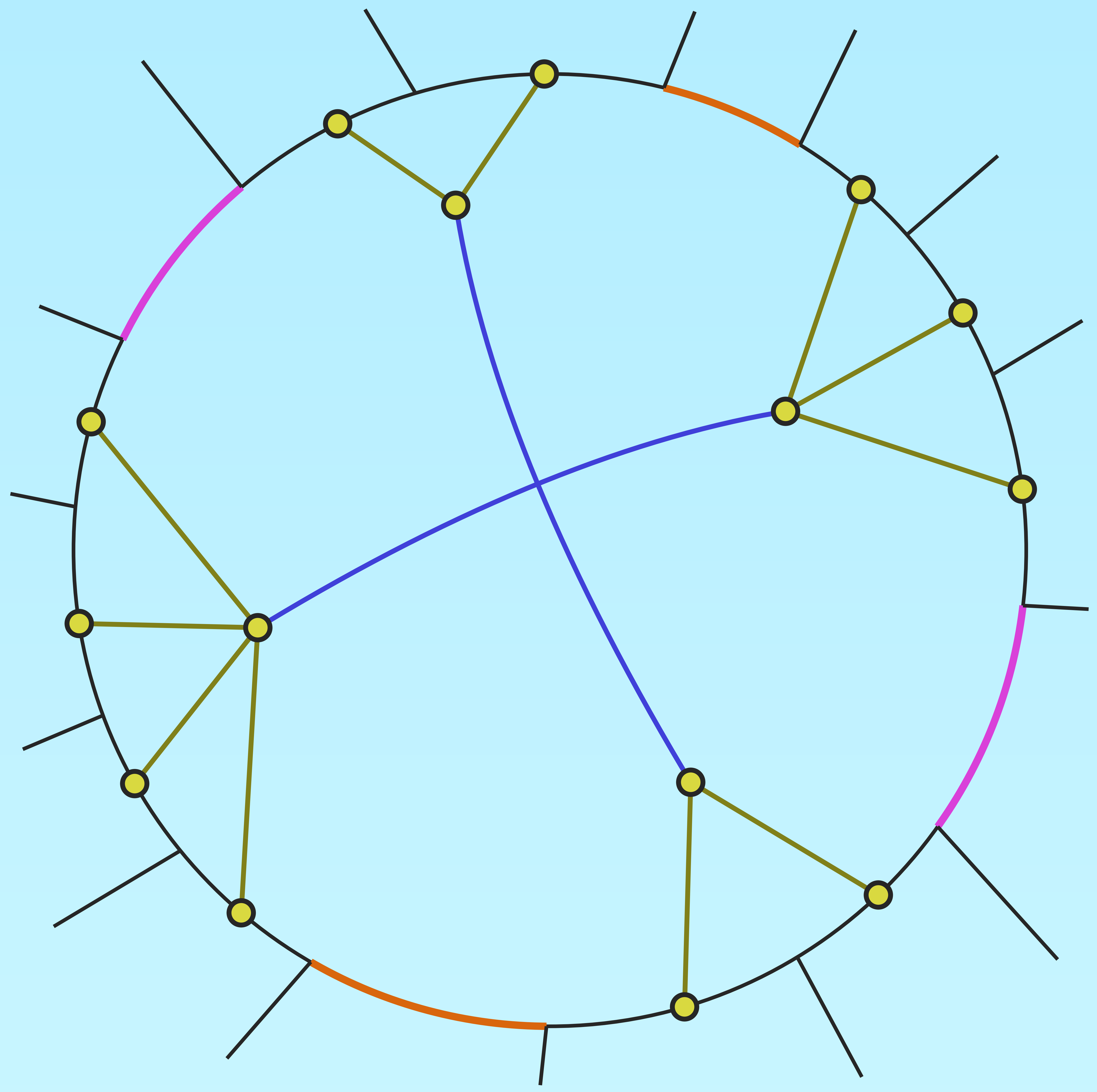
Issue: This doesn't work for adjacent popular faces.

Instead, we build a somewhat more involved structure for each popular face.



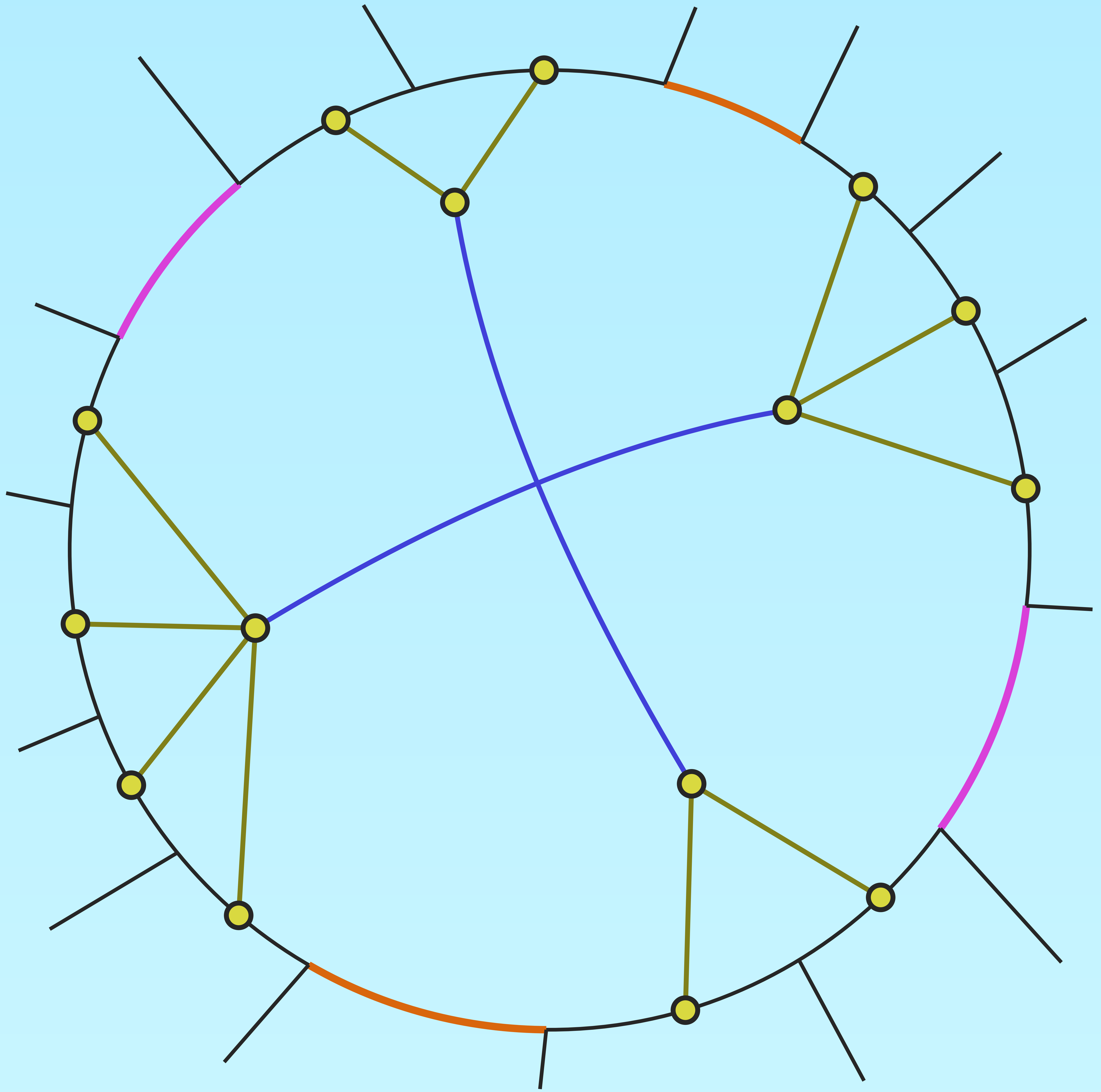
Issue: This doesn't work for adjacent popular faces.

Instead, we build a somewhat more involved structure for each popular face.



Issue: This doesn't work for adjacent popular faces.

Instead, we build a somewhat more involved structure for each popular face.



Issue: This doesn't work for adjacent popular faces.

Instead, we build a somewhat more involved structure for each popular face.

Added benefit: only $O(n)$ resolution edges per popular face!

Our results:

Deciding whether it is possible to insert one more curve to an existing arrangement such that there are no more popular faces, is NP-hard.

The problem is randomized FPT in the number of popular faces in the input arrangement.

Part 3

FPT

Our result is heavily inspired by Björklund et al.

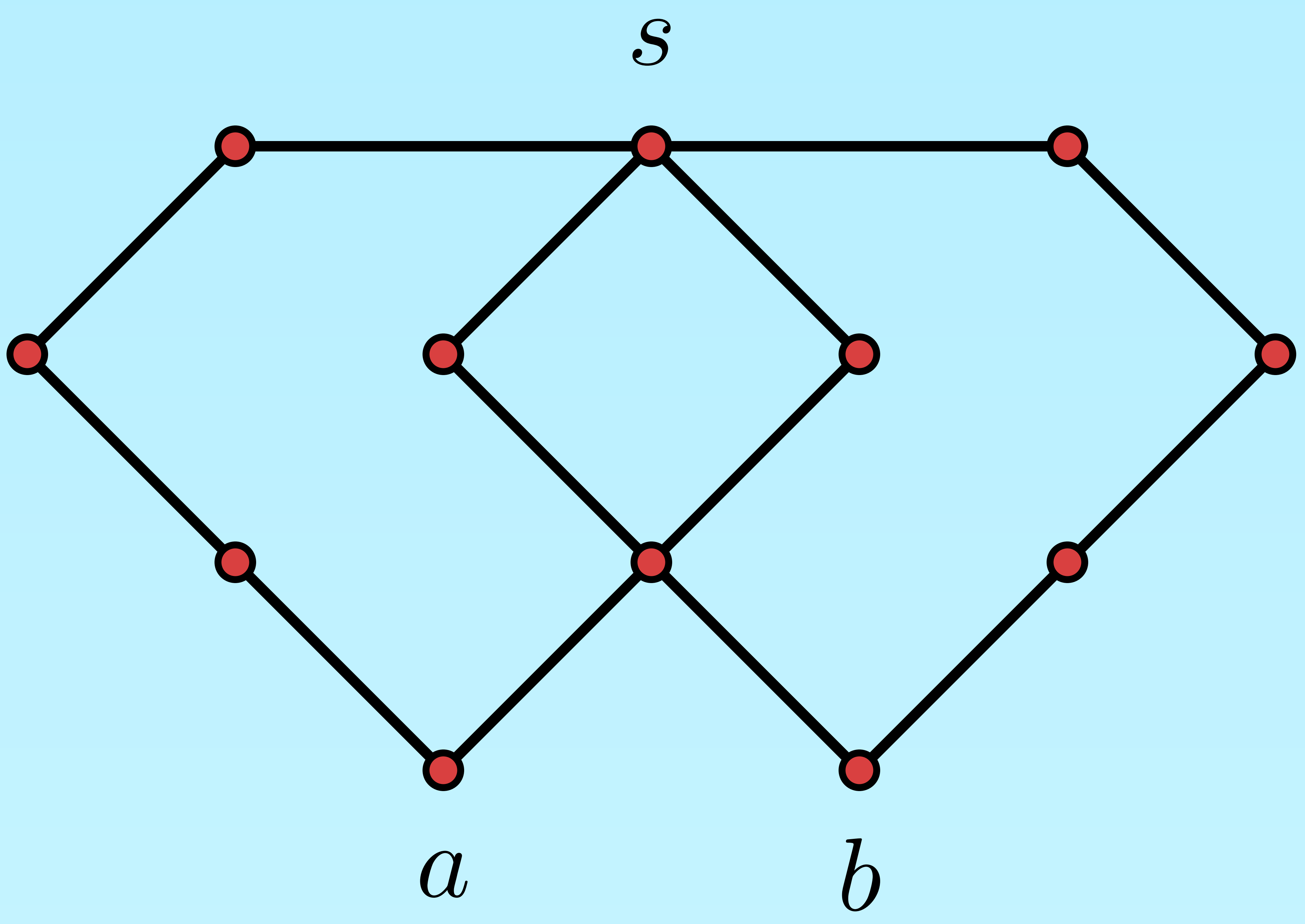
[Björklund, Husfeld & Taslaman, SODA 2012]

Our result is heavily inspired by Björklund et al.

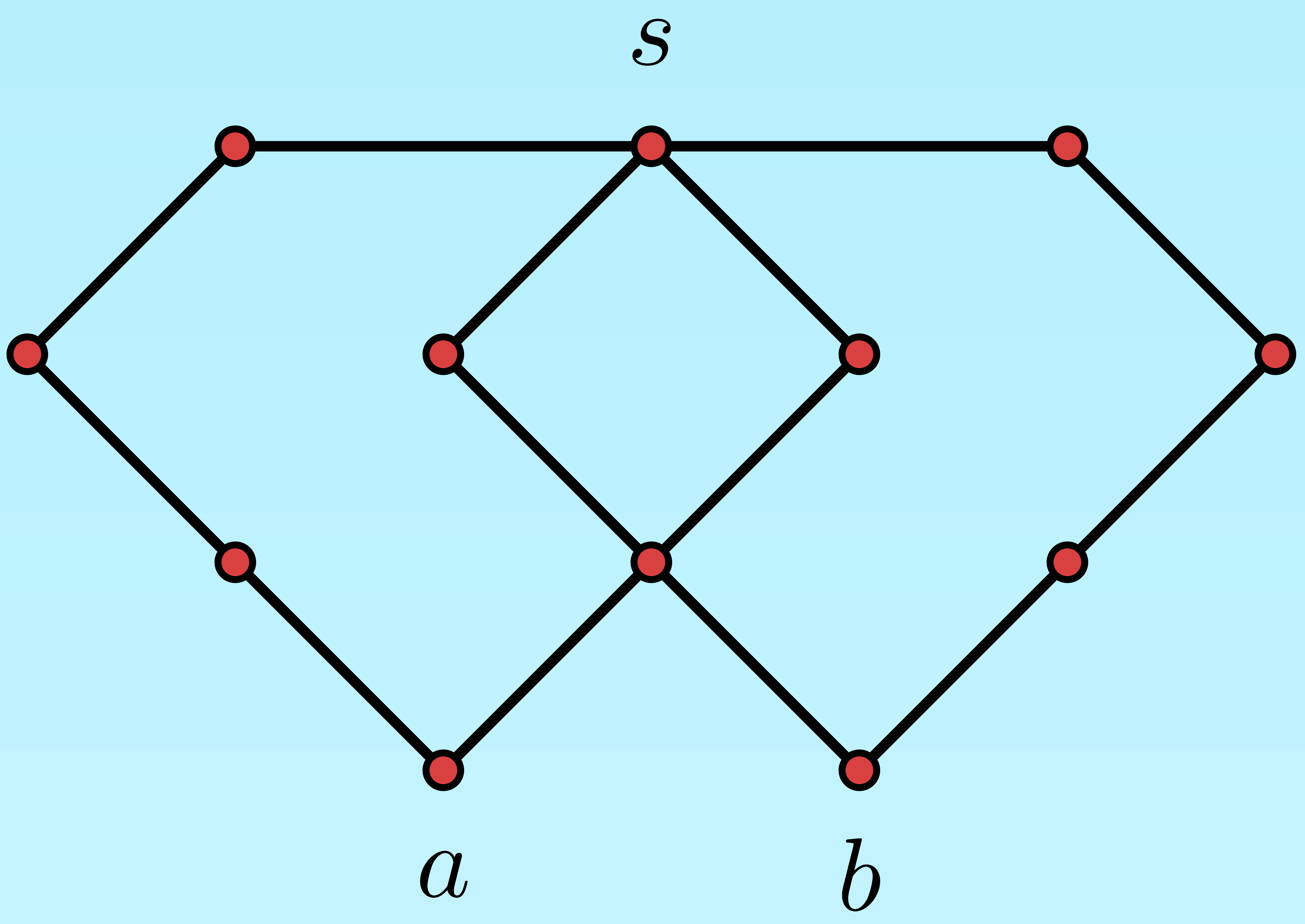
Let's look at a simplified / slightly different problem.

[Björklund, Husfeld & Taslaman, SODA 2012]

Problem. Given a graph, find the shortest simple path from a to b that passes through s.

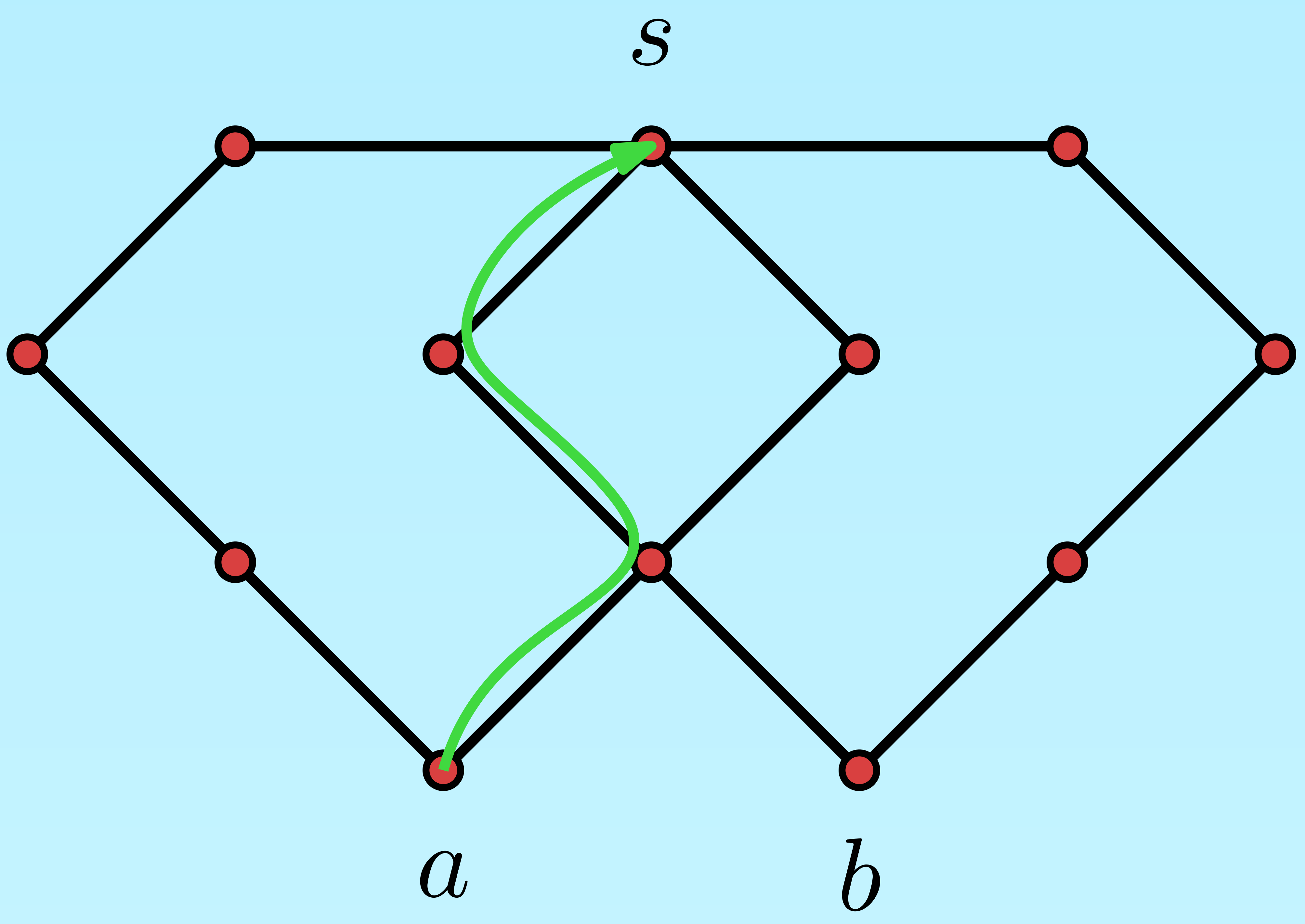


Problem. Given a graph, find the shortest simple path from a to b that passes through s.



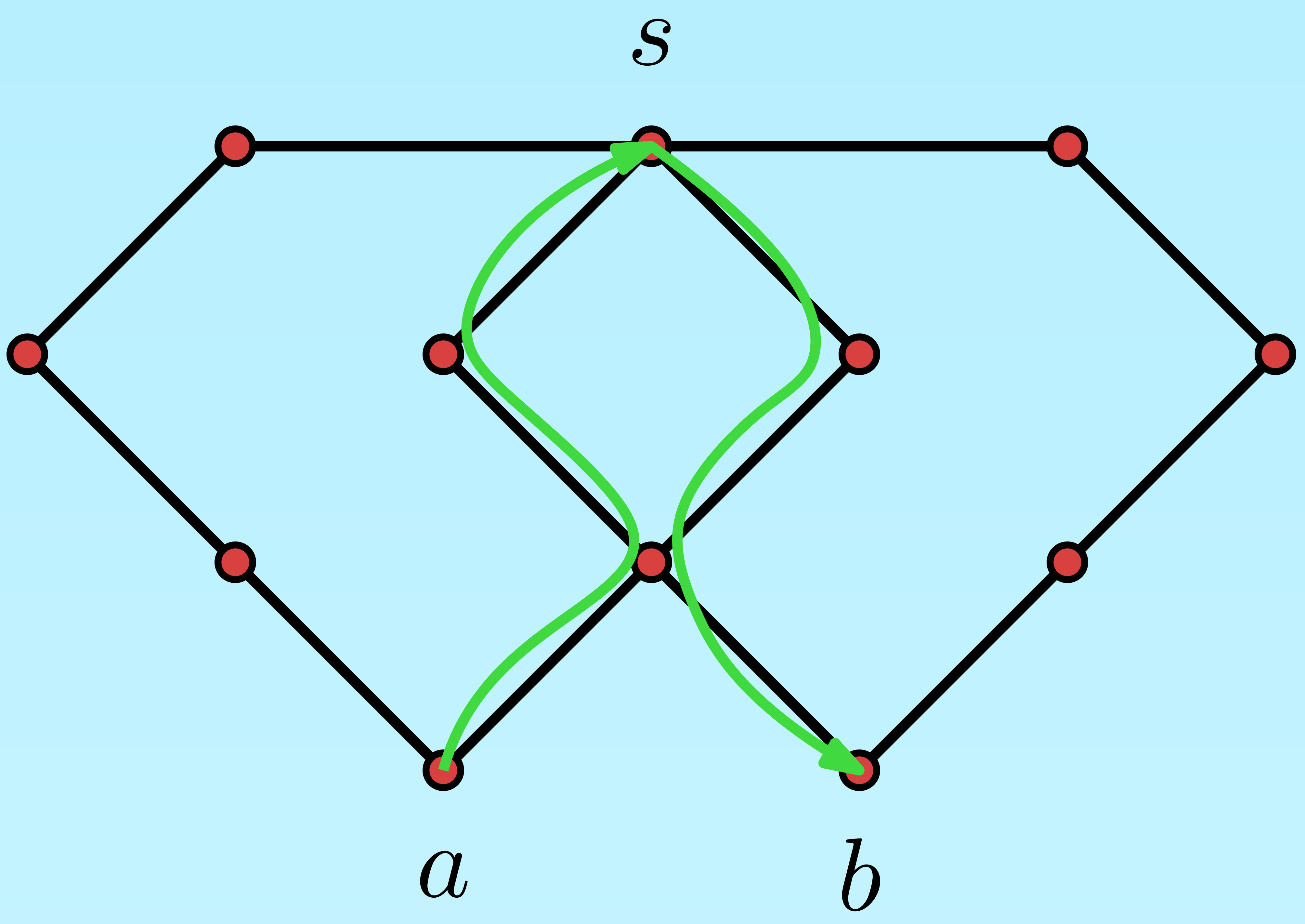
Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Observation. Shortest path from a to s + shortest path from s to b might not be simple.



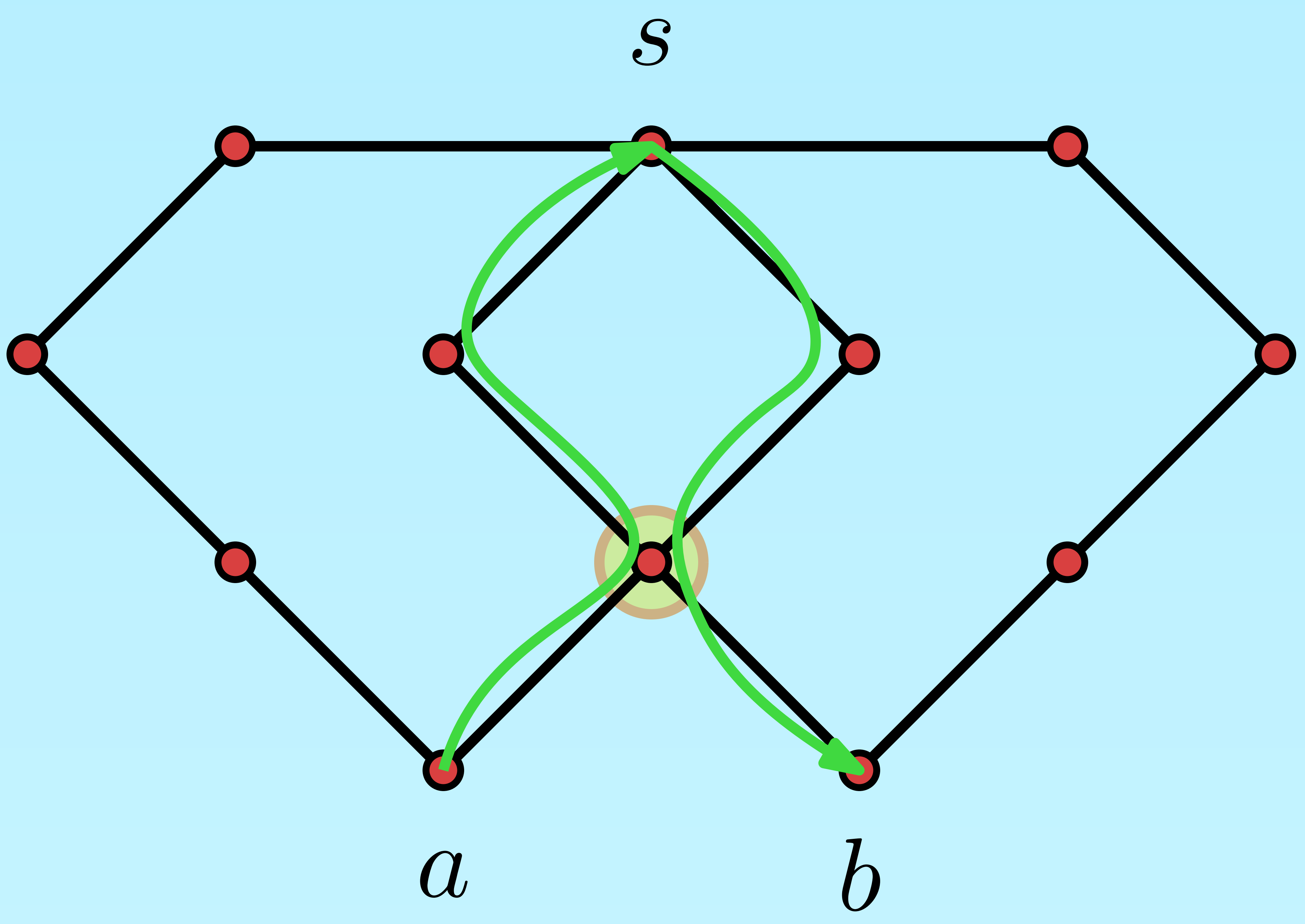
Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Observation. Shortest path from a to s + shortest path from s to b might not be simple.



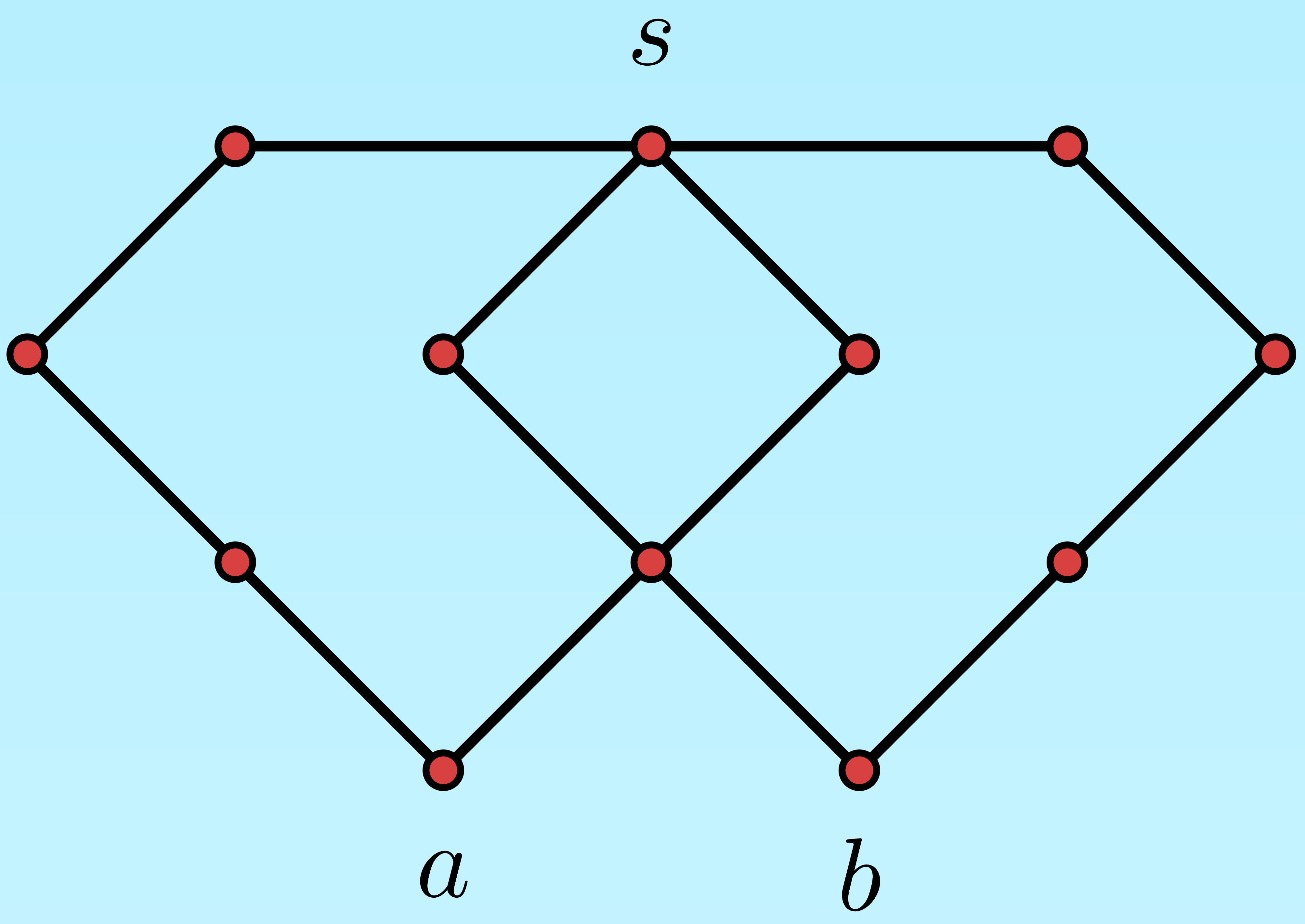
Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Observation. Shortest path from a to s + shortest path from s to b might not be simple.

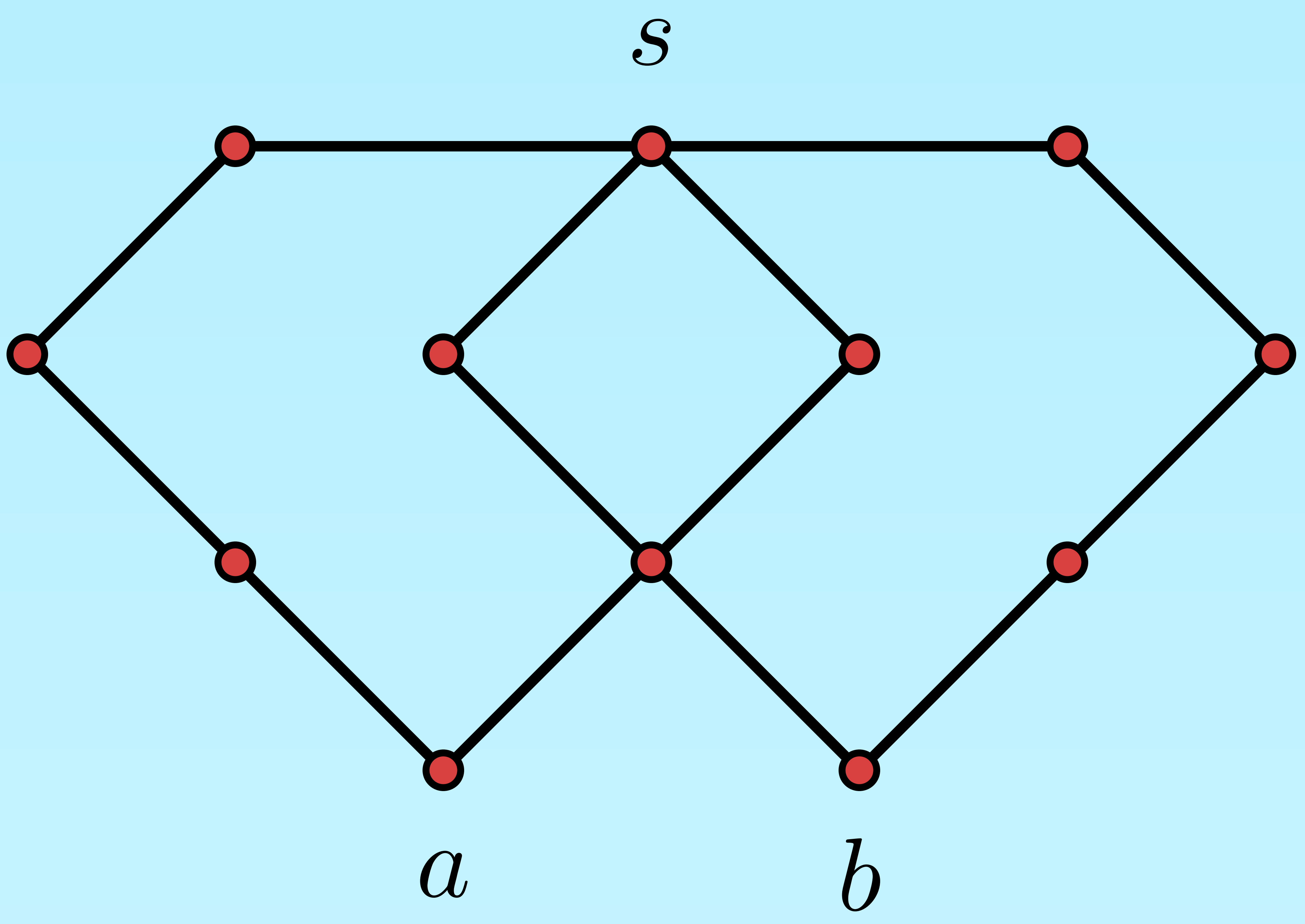


Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Observation. Shortest path from a to s + shortest path from s to b might not be simple.

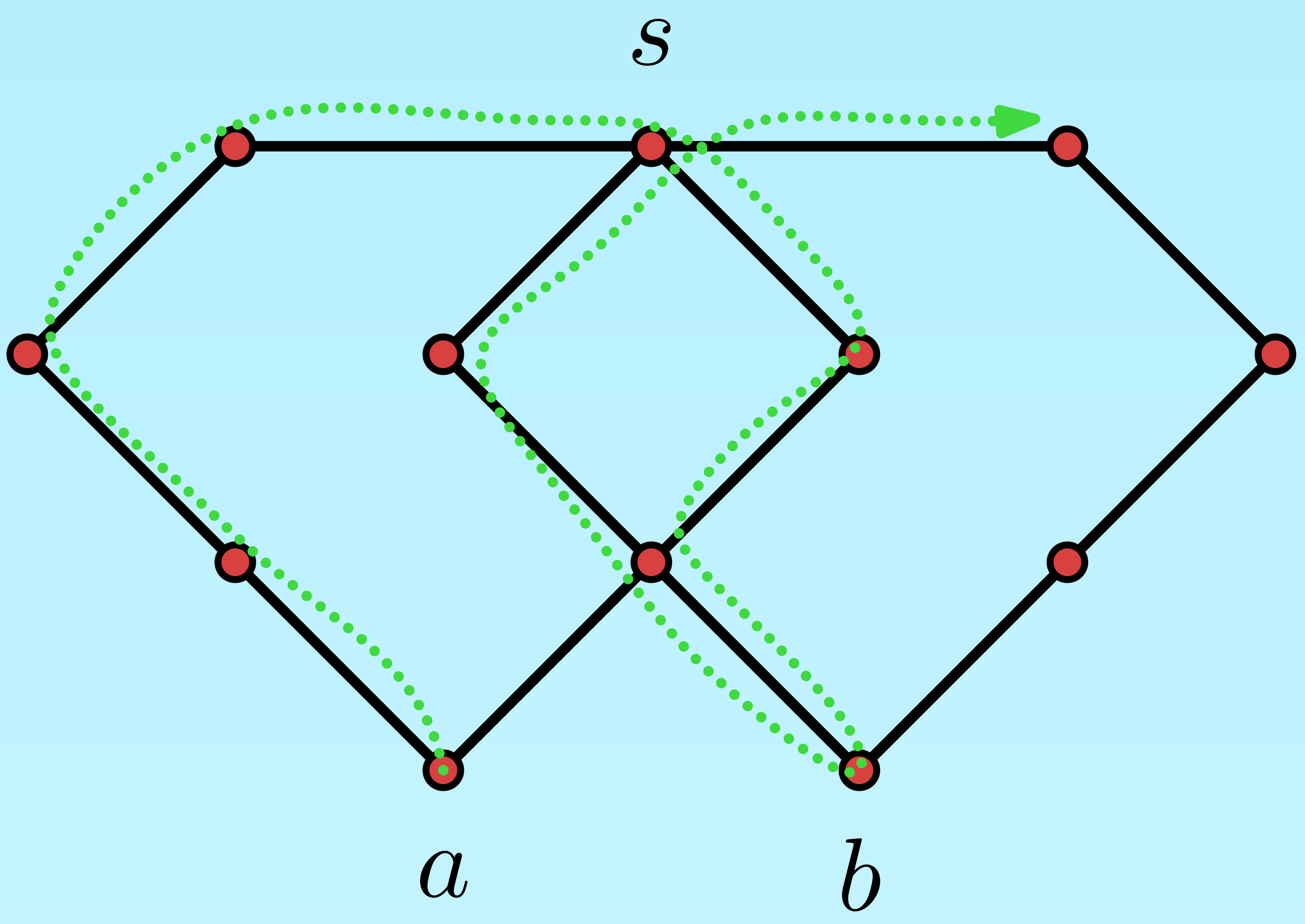


Problem. Given a graph, find the shortest simple path from a to b that passes through s.



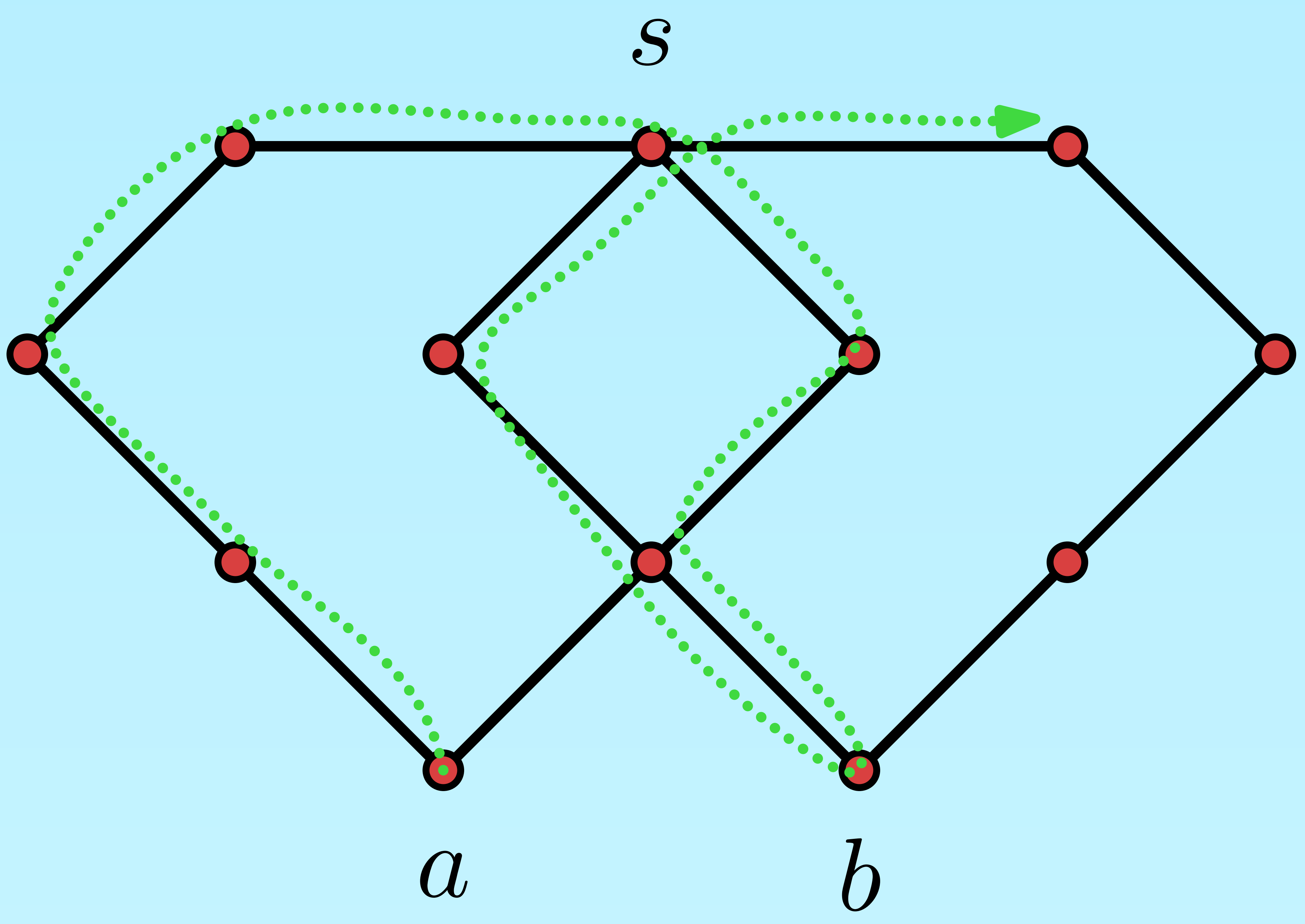
Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Define a walk to be a sequence of adjacent vertices.



Problem. Given a graph, find the shortest simple path from a to b that passes through s.

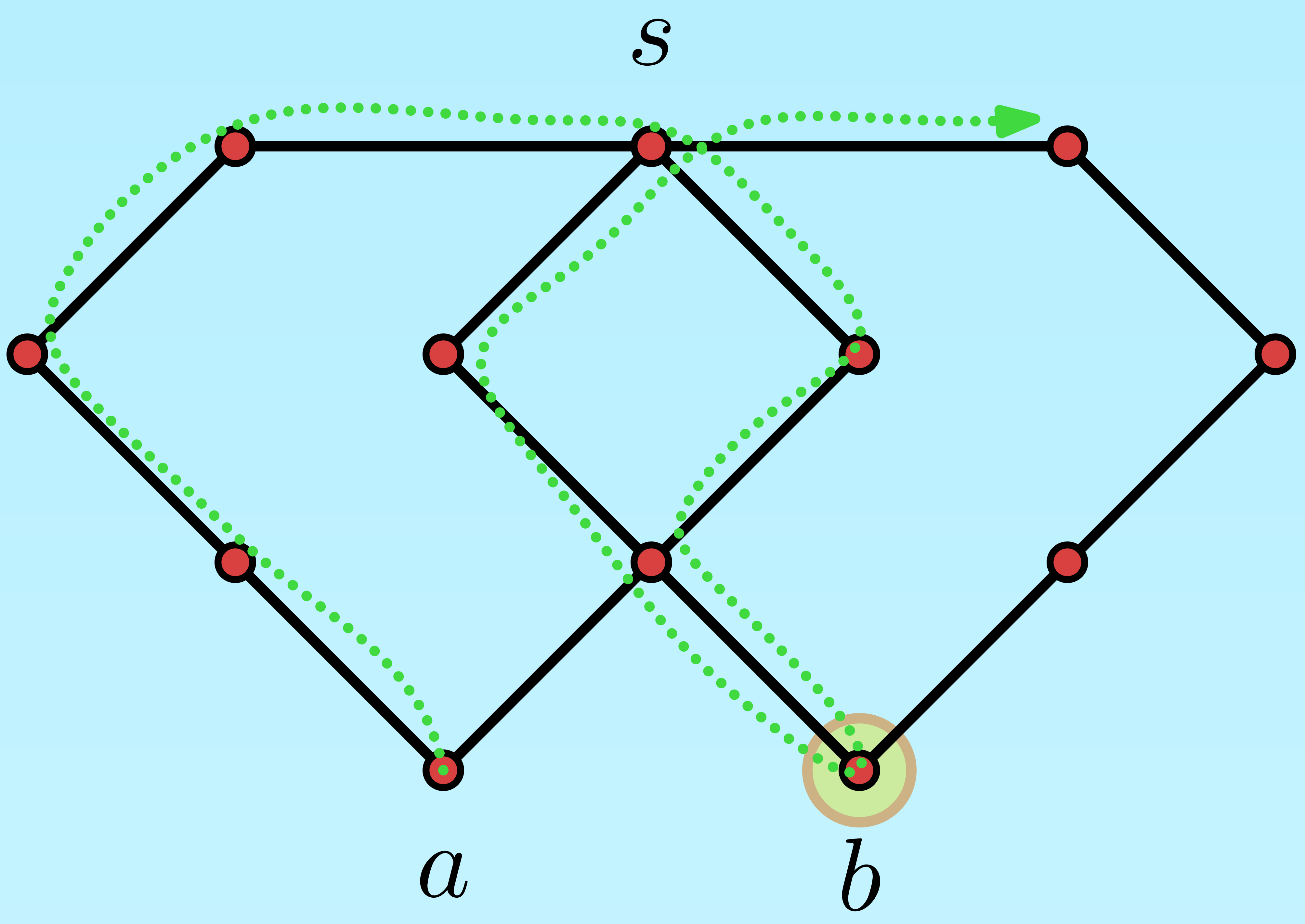
Define a walk to be a sequence of adjacent vertices.



Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Define a walk to be a sequence of adjacent vertices.

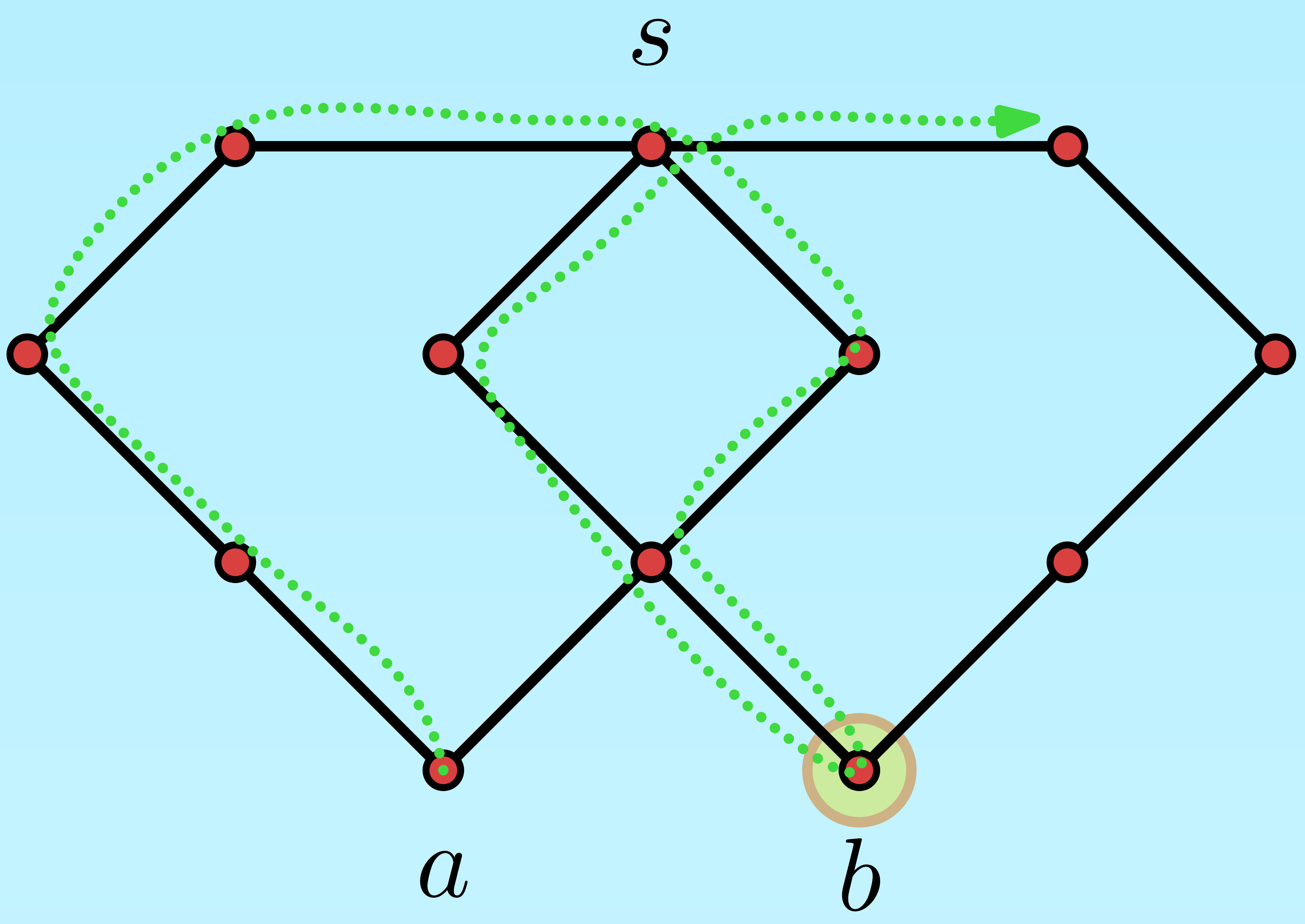
A spur is a part of a walk of the form uvu.



Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Define a walk to be a sequence of adjacent vertices.

A spur is a part of a walk of the form uvu.

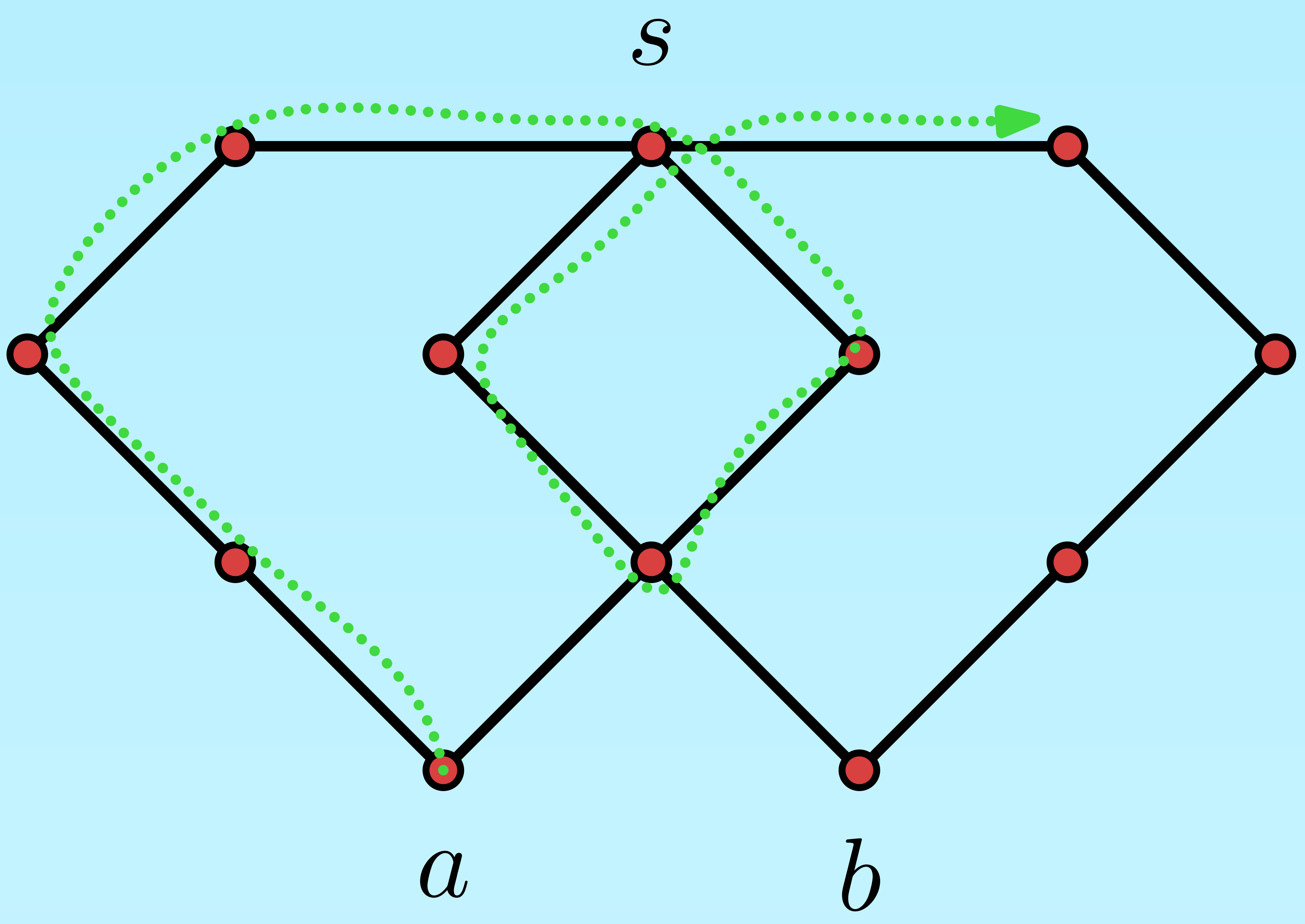


Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Define a walk to be a sequence of adjacent vertices.

A spur is a part of a walk of the form uvu.

A spurless walk is a walk without spurs.

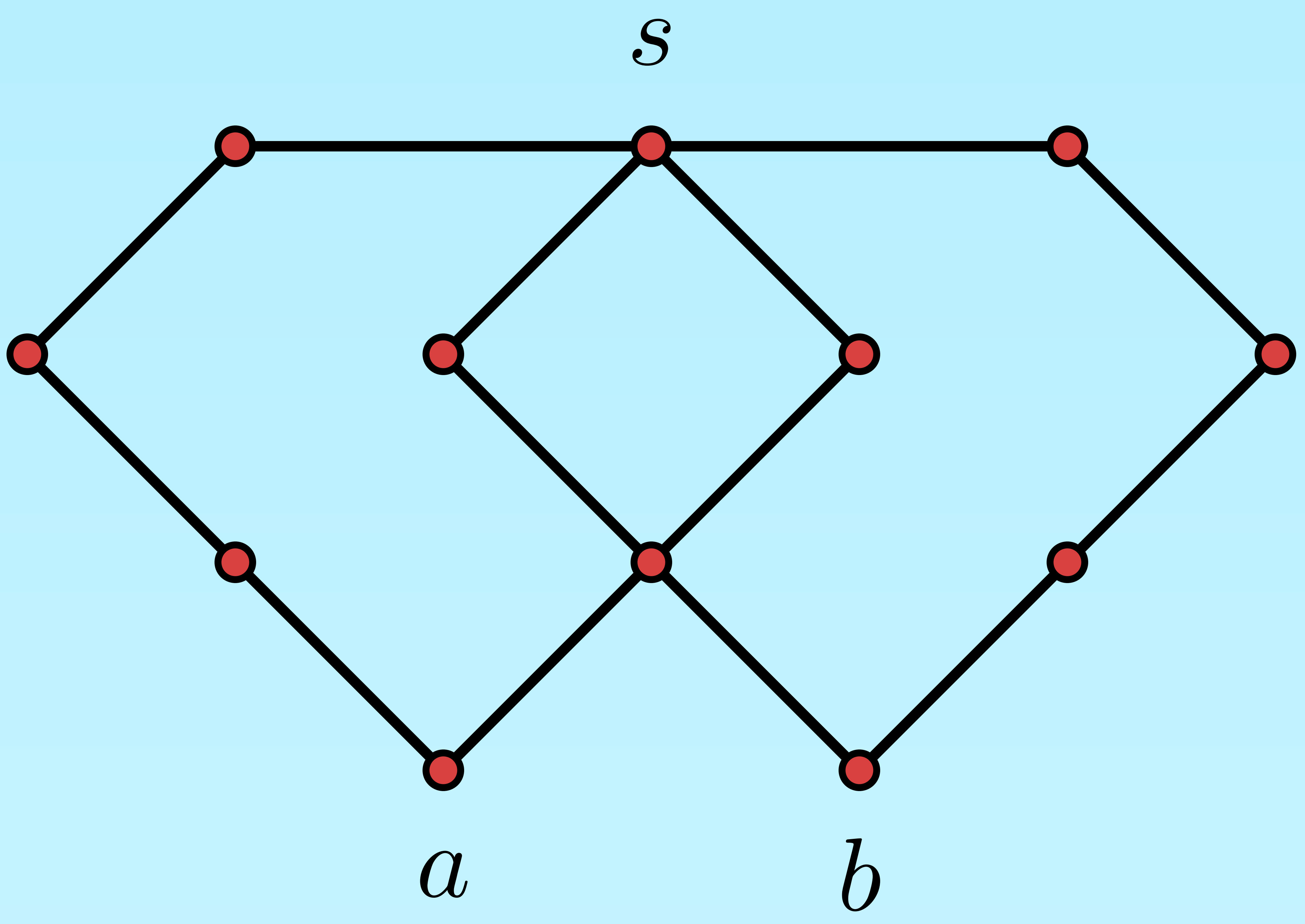


Problem. Given a graph, find the shortest simple path from a to b that passes through s.

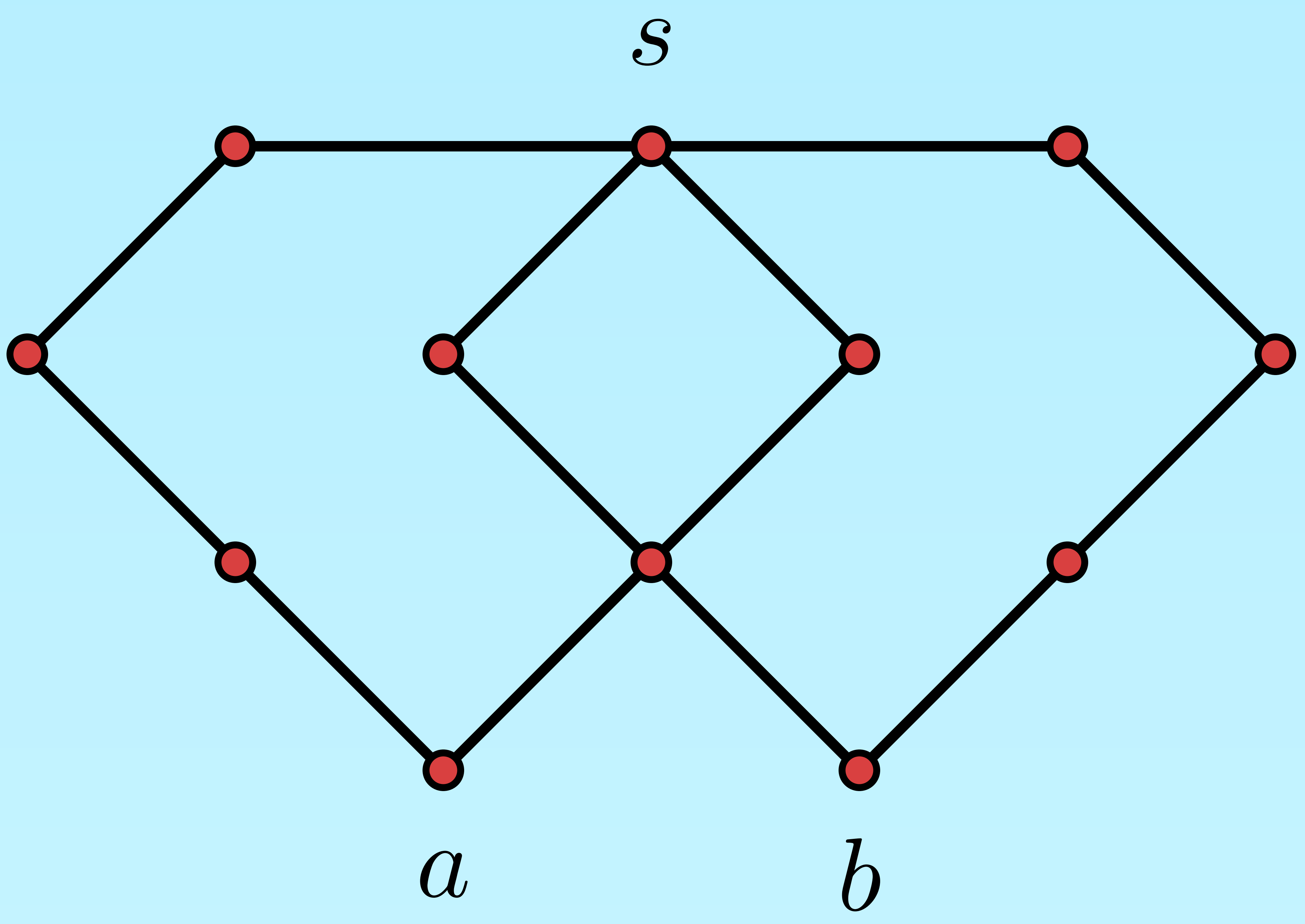
Define a walk to be a sequence of adjacent vertices.

A spur is a part of a walk of the form uvu.

A spurless walk is a walk without spurs.

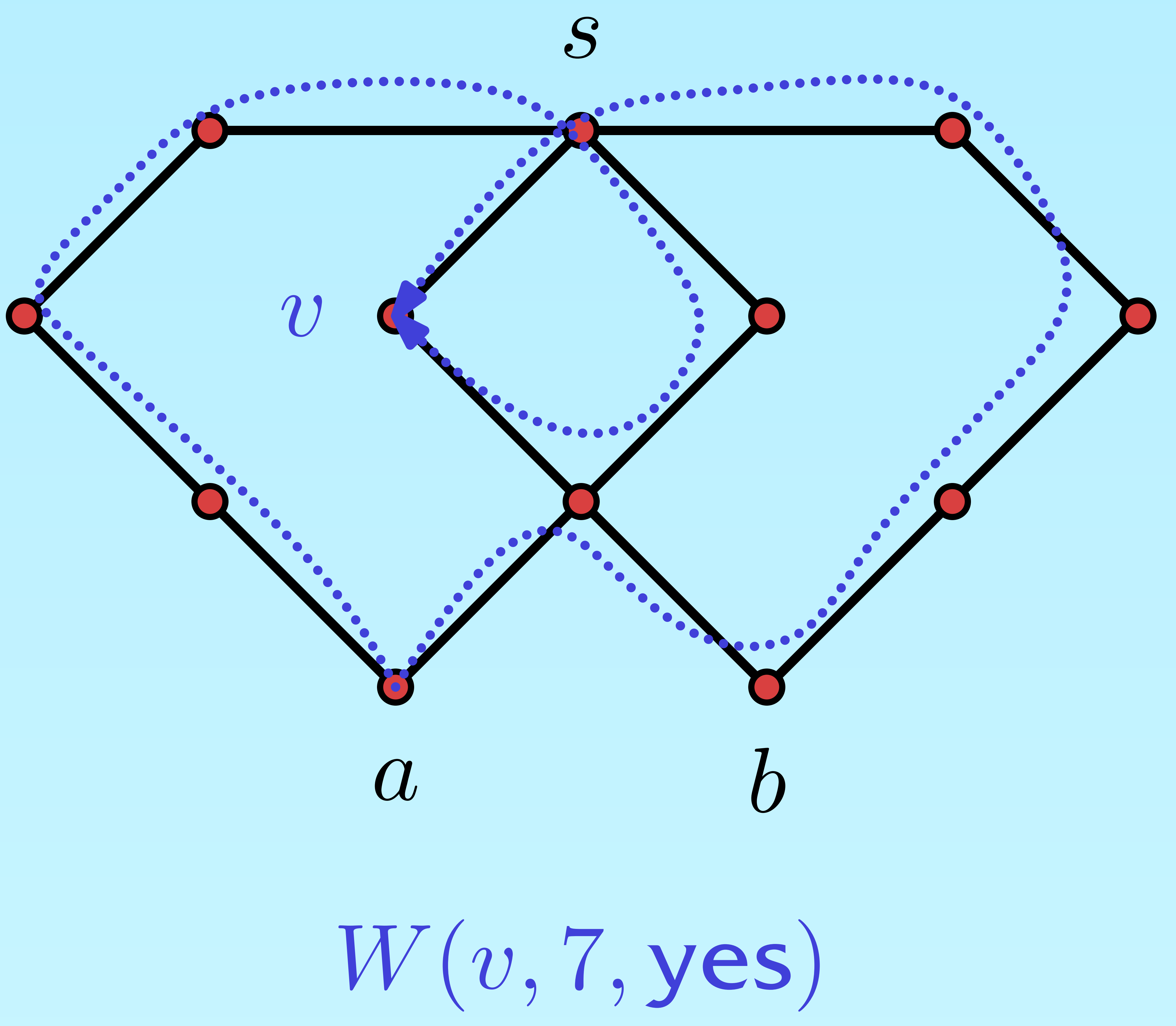


Problem. Given a graph, find the shortest simple path from a to b that passes through s.



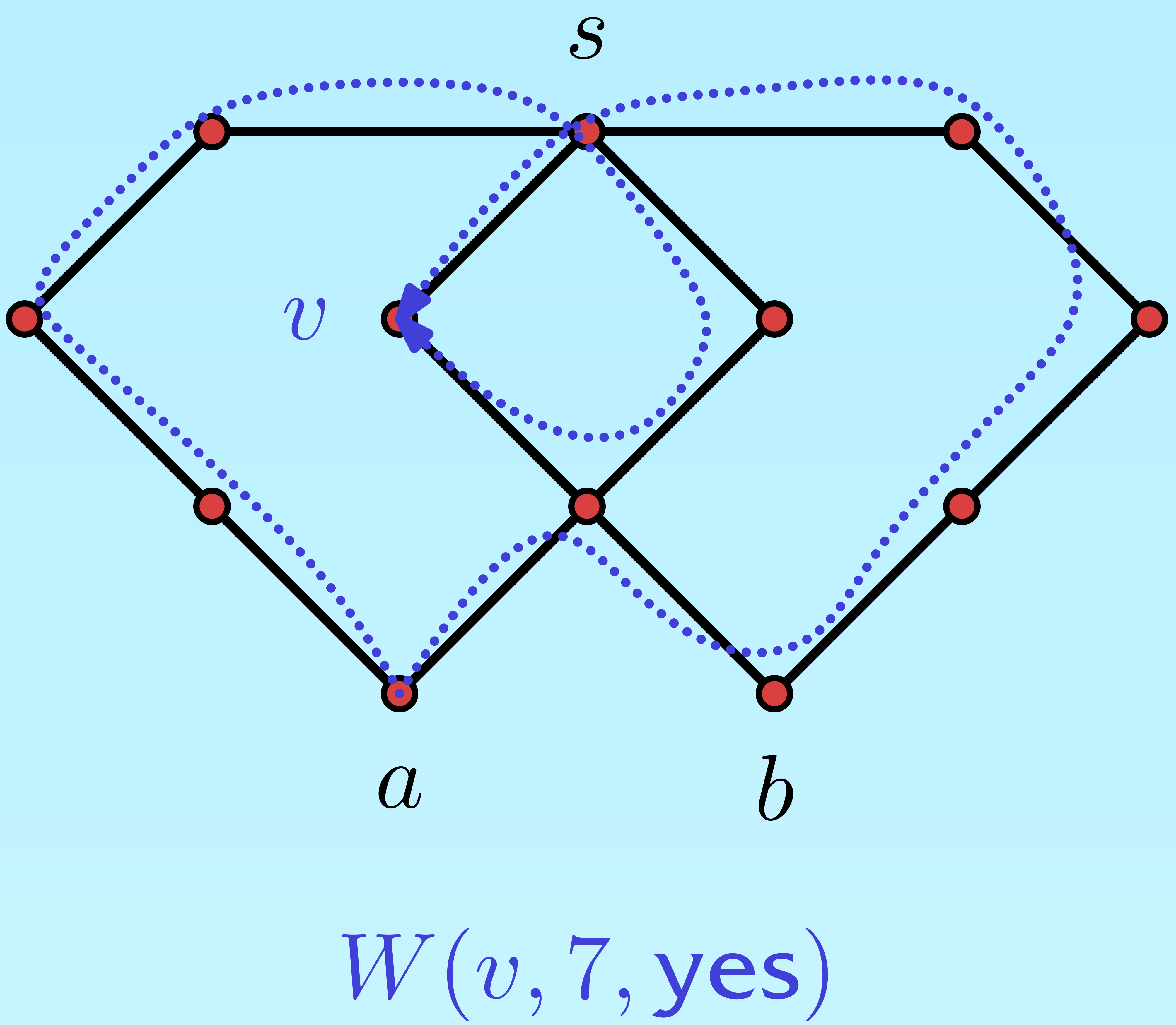
Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Let $W(v, \ell, \text{yes/no})$ be the set of all spurless walks from a to v of length ℓ that do/don't pass through s.



Problem. Given a graph, find the shortest simple path from a to b that passes through s.

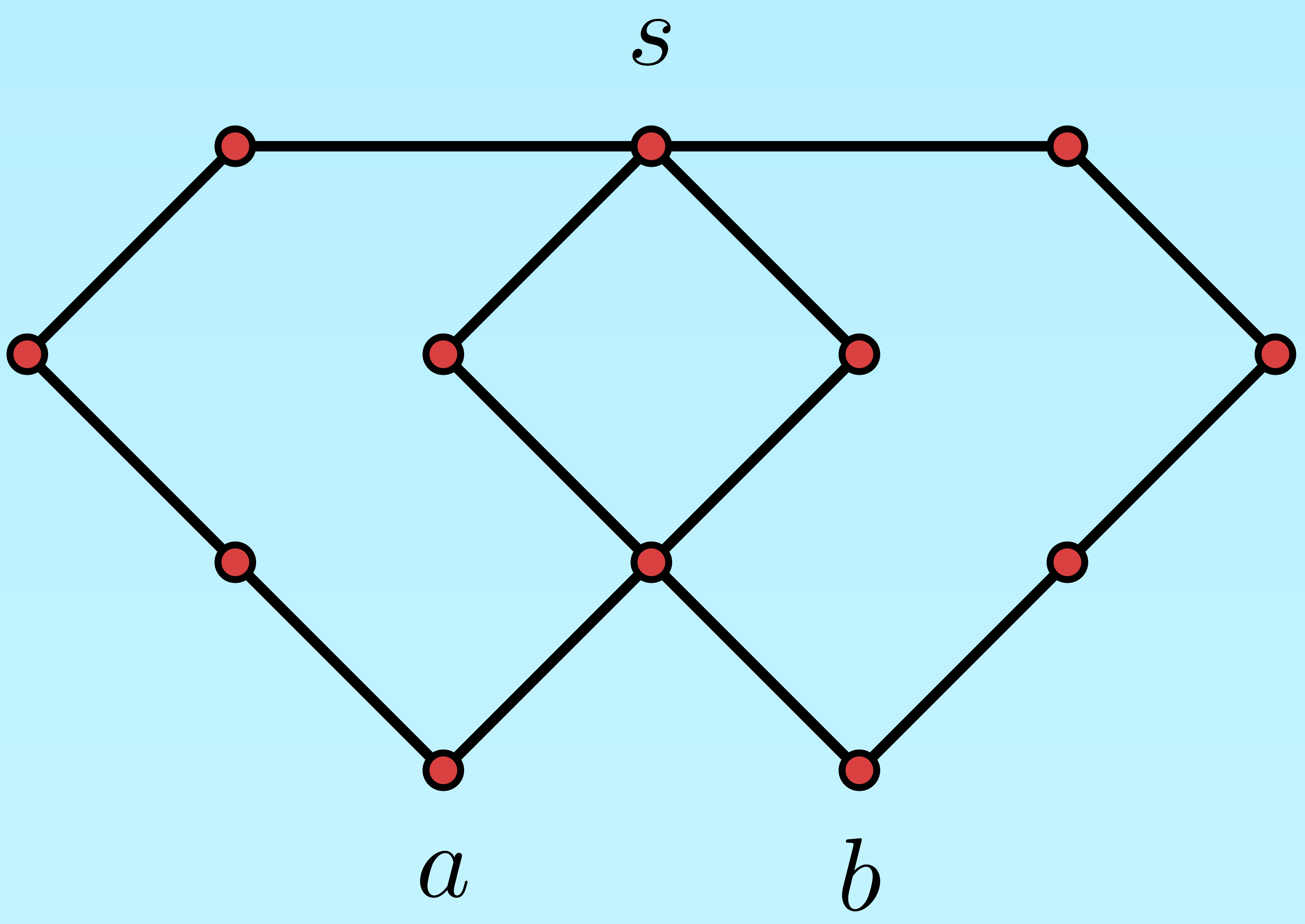
Let $W(v, \ell, \text{yes/no})$ be the set of all spurless walks from a to v of length ℓ that do/don't pass through s.



Problem. Given a graph, find the shortest simple path from a to b that passes through s.

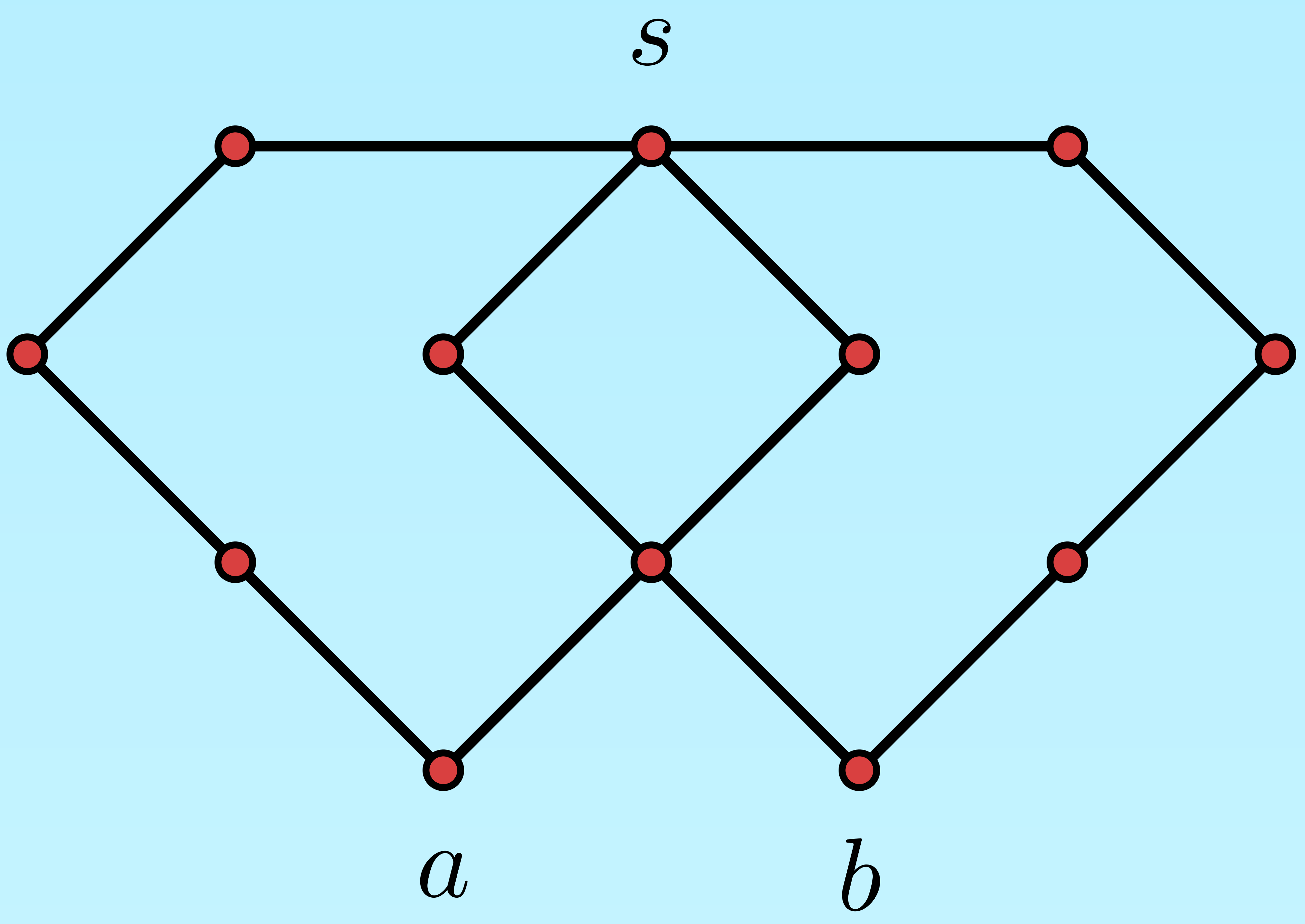
Let $W(v, \ell, \text{yes/no})$ be the set of all spurless walks from a to v of length ℓ that do/don't pass through s.

Then we want to find the smallest ℓ for which $W(b, \ell, \text{yes})$ contains a simple walk.



Problem. Given a graph, find the shortest simple path from a to b that passes through s.

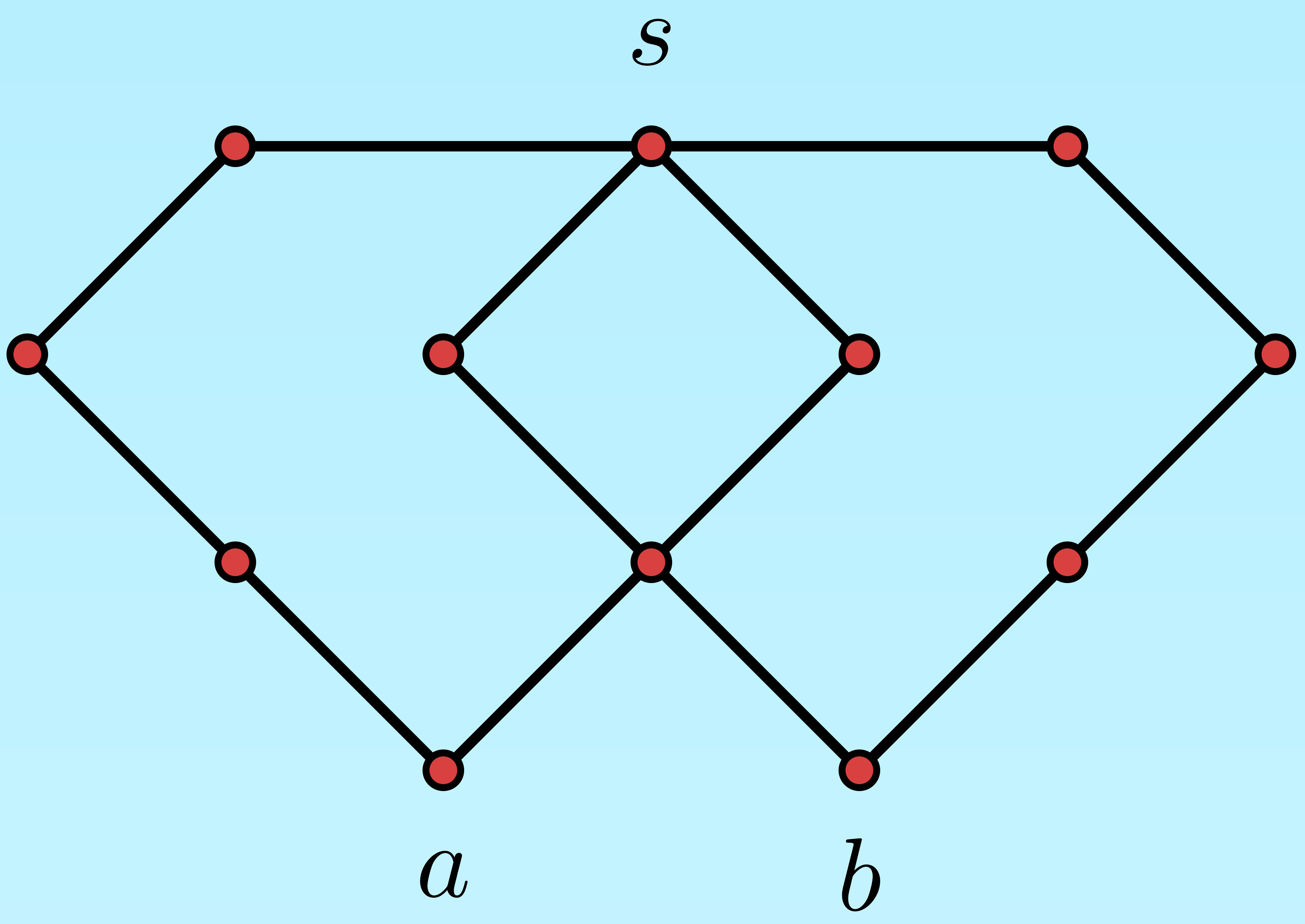
Let $W(v, \ell, \text{yes/no})$ be the set of all spurless walks from a to v of length ℓ that do/don't pass through s.



Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Let $W(v, \ell, \text{yes/no})$ be the set of all spurless walks from a to v of length ℓ that do/don't pass through s.

$W(\cdot, \cdot, \cdot)$ can easily be defined recursively.

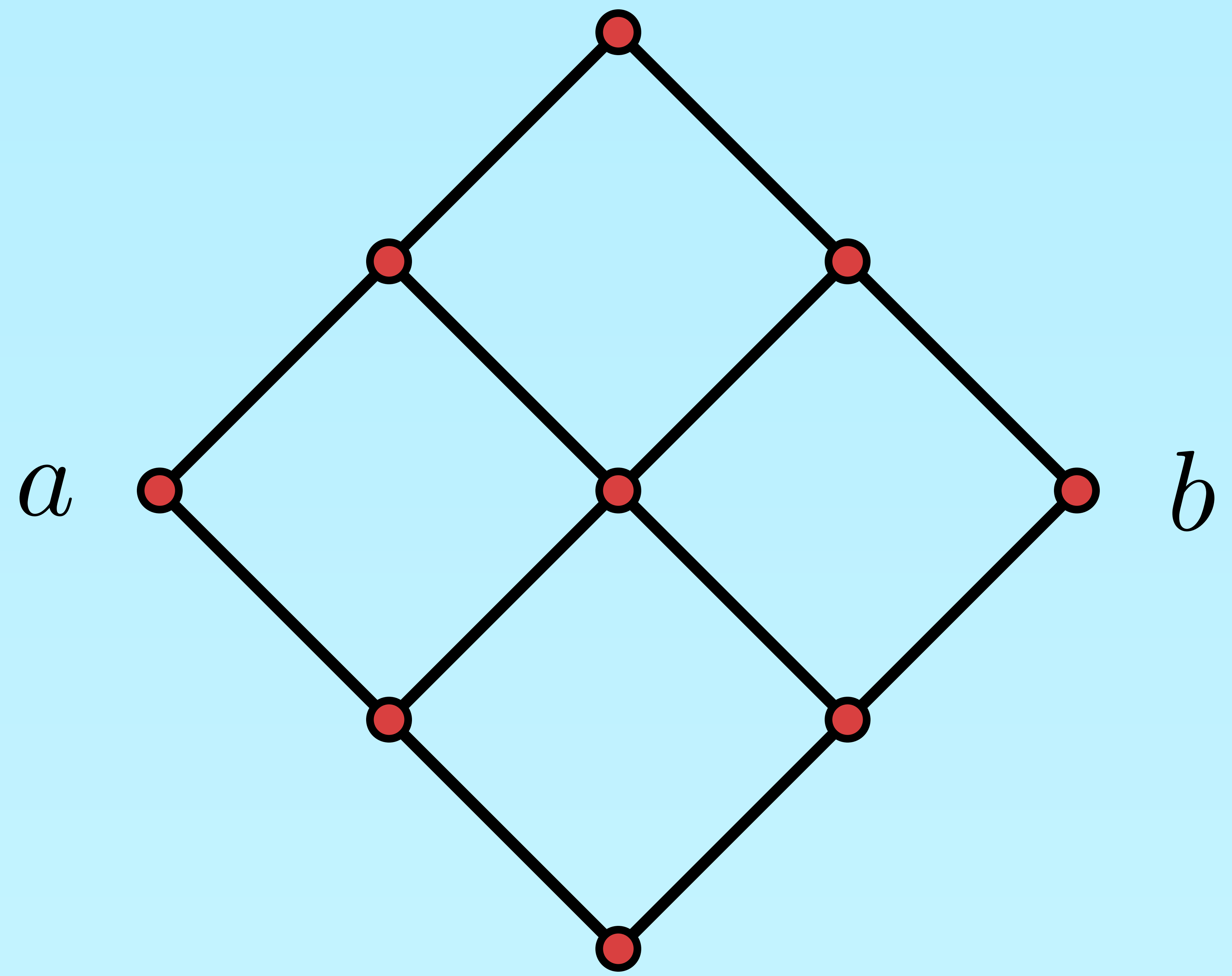


Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Let $W(v, \ell, \text{yes/no})$ be the set of all spurless walks from a to v of length ℓ that do/don't pass through s.

$W(\cdot,\cdot,\cdot)$ can easily be defined recursively.

Issue: $W(\cdot,\cdot,\cdot)$ has exponential size.

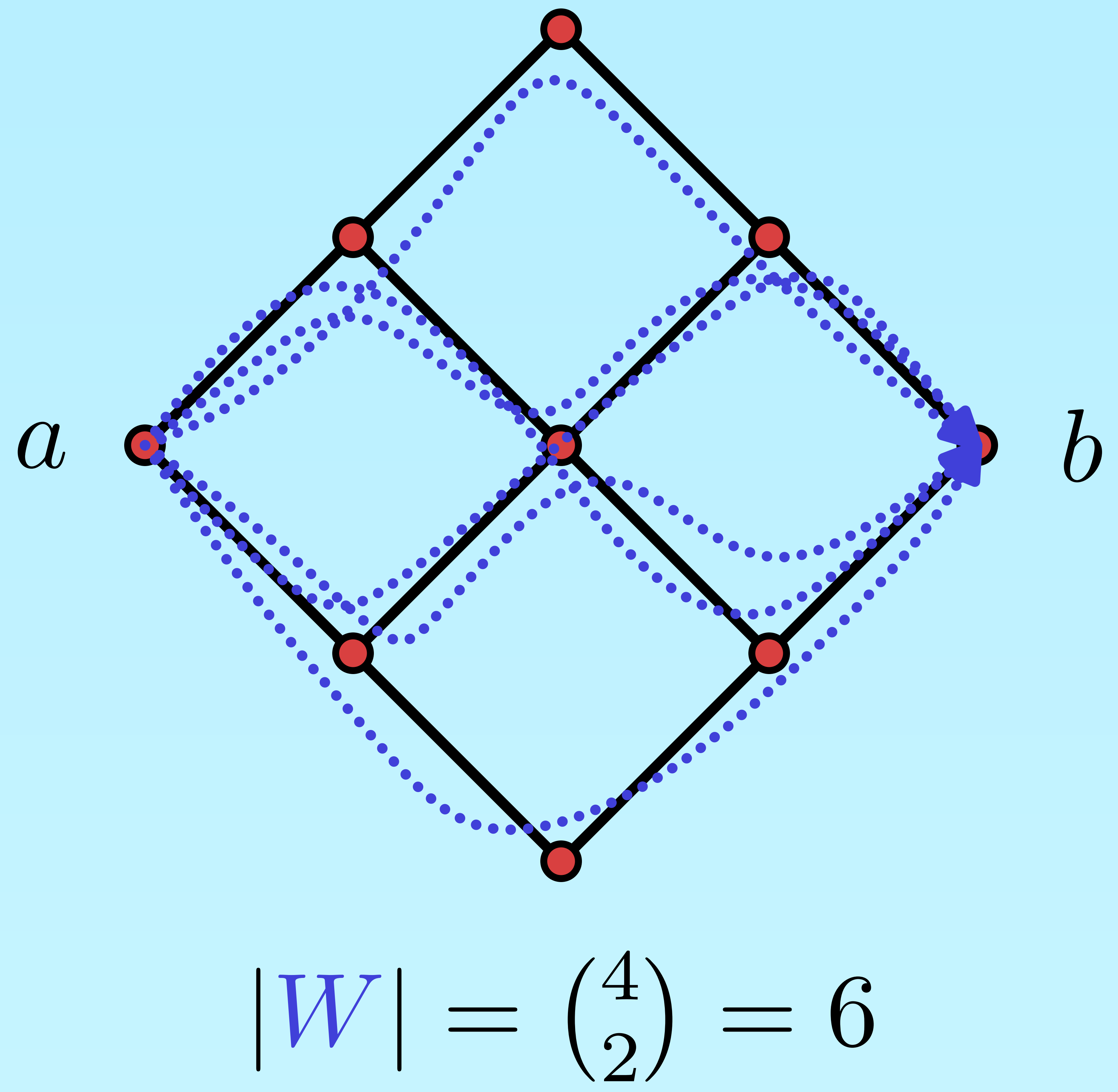


Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Let $W(v, \ell, \text{yes/no})$ be the set of all spurless walks from a to v of length ℓ that do/don't pass through s.

$W(\cdot,\cdot,\cdot)$ can easily be defined recursively.

Issue: $W(\cdot,\cdot,\cdot)$ has exponential size.

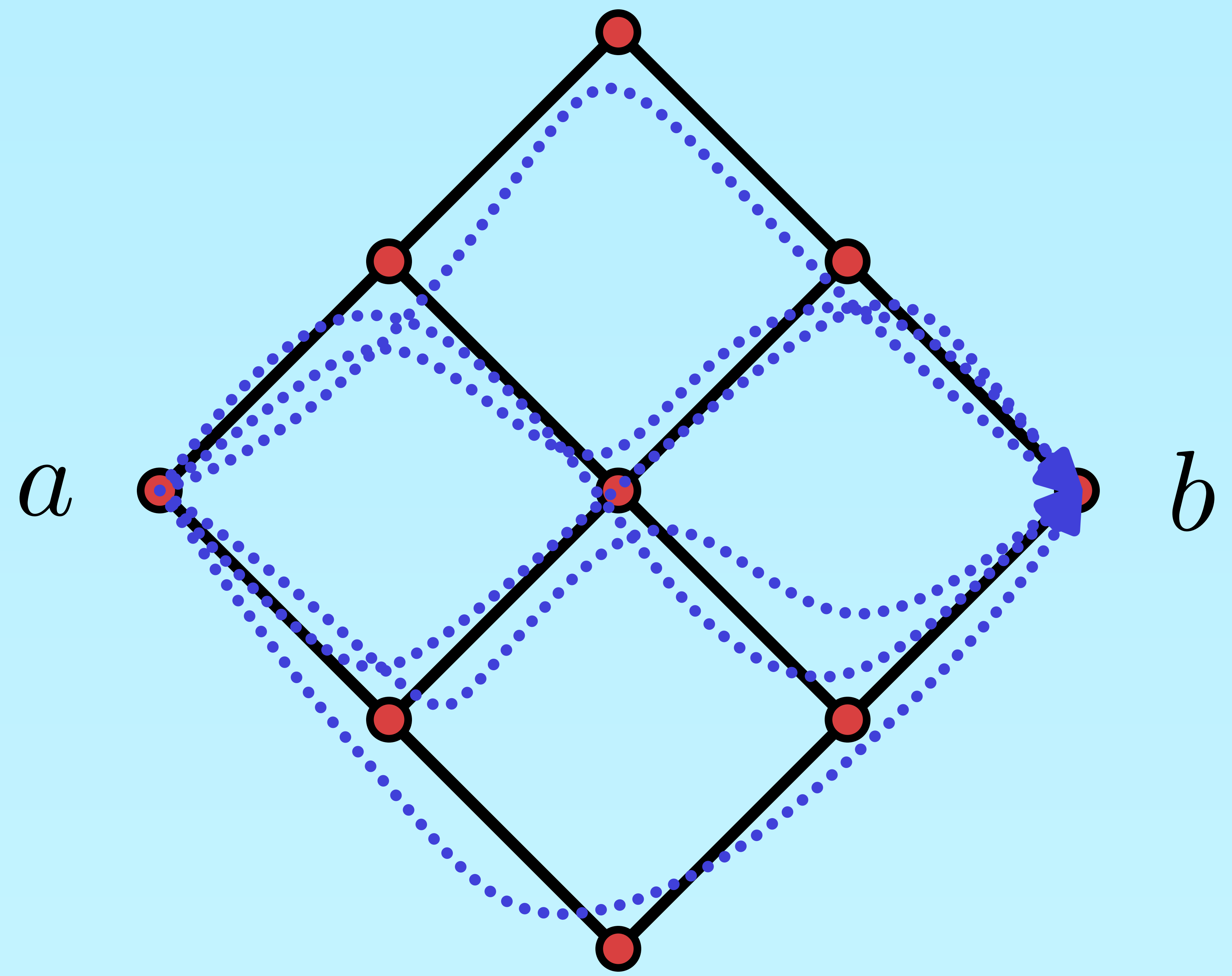


Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Let $W(v, \ell, \text{yes/no})$ be the set of all spurless walks from a to v of length ℓ that do/don't pass through s.

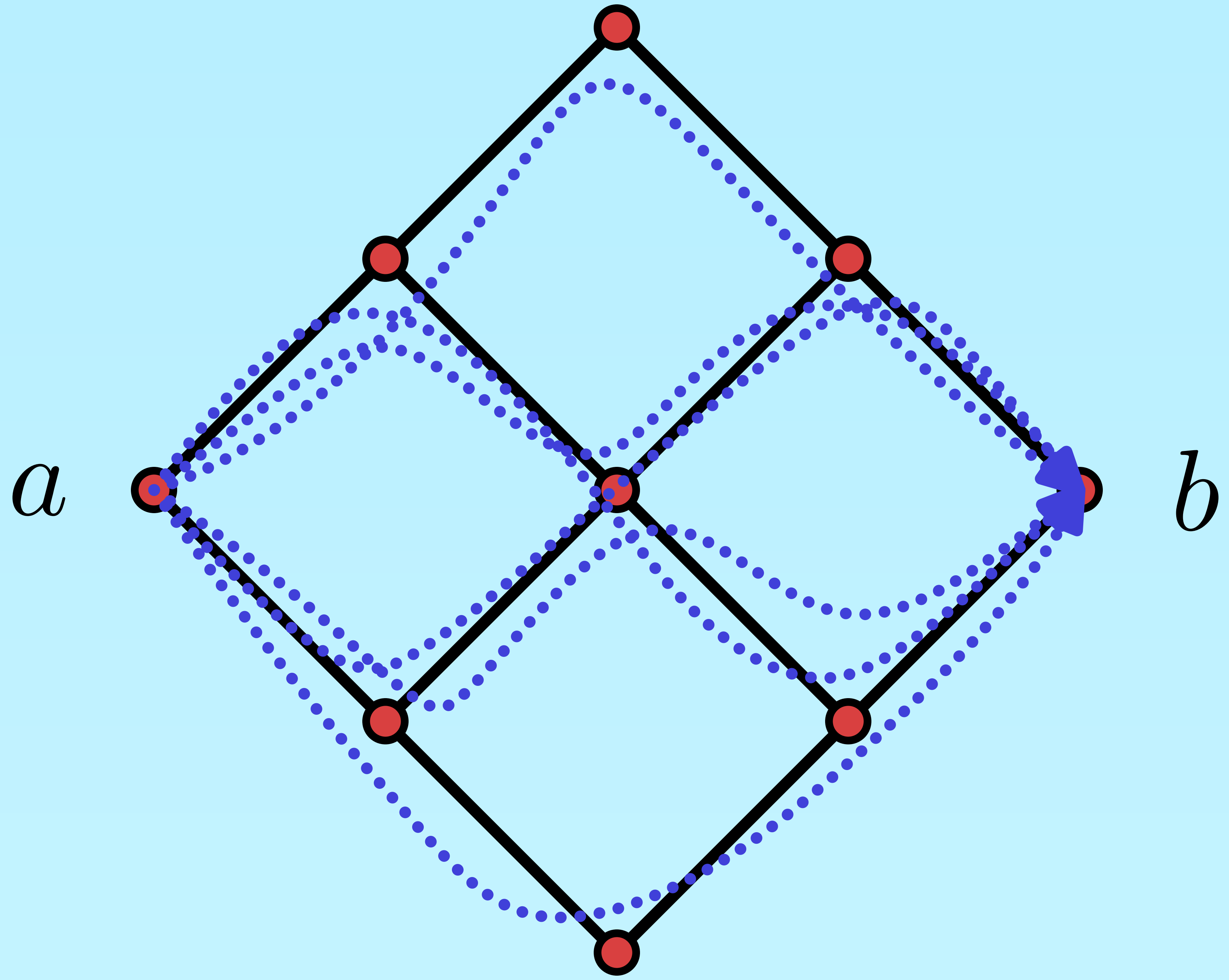
$W(\cdot,\cdot,\cdot)$ can easily be defined recursively.

Issue: $W(\cdot,\cdot,\cdot)$ has exponential size.



$$|W| = \binom{4}{2} = 6$$

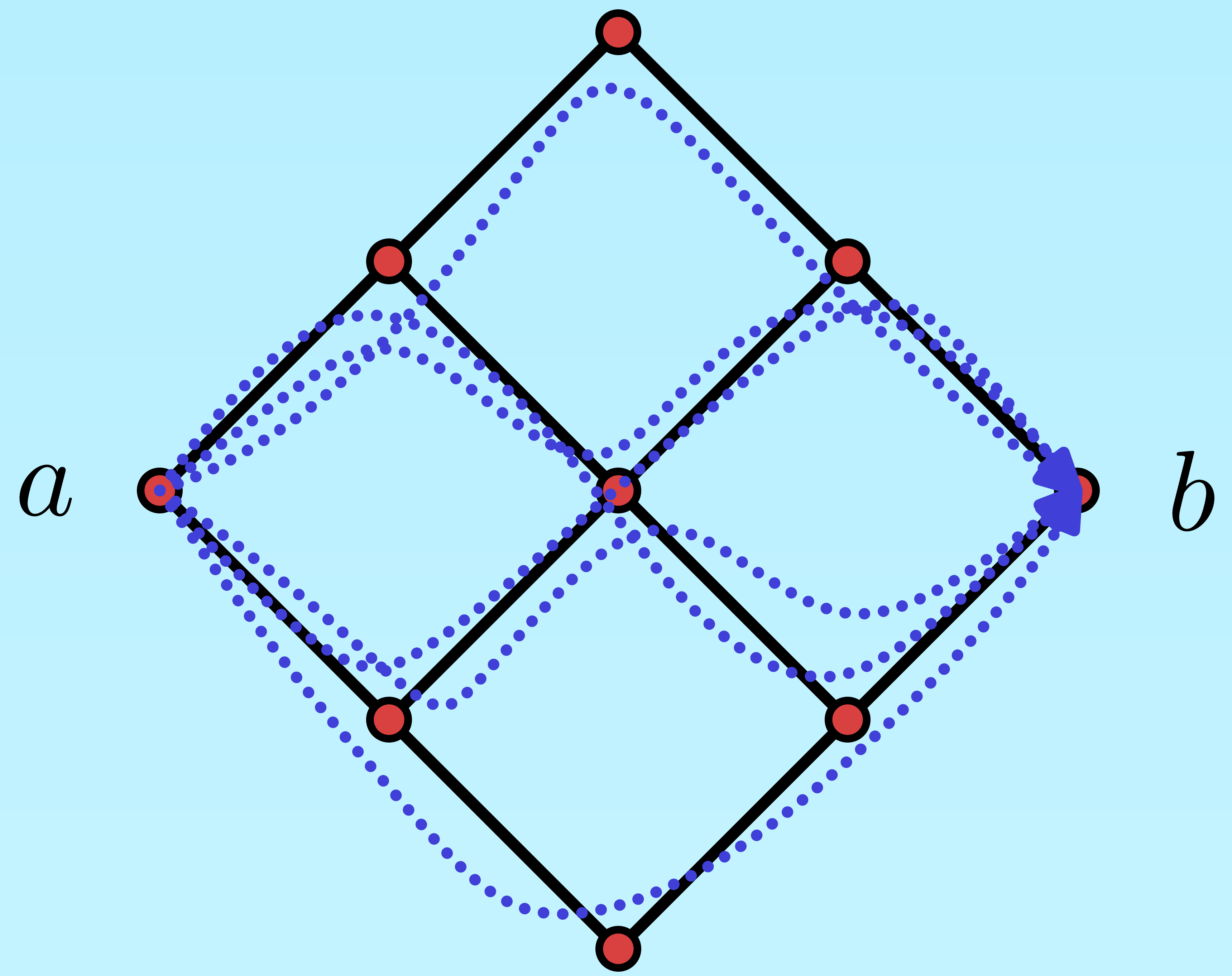
Idea: Maintain a signature of $W(\cdot,\cdot,\cdot)$.



$$|W| = \binom{4}{2} = 6$$

Idea: Maintain a signature of $W(\cdot,\cdot,\cdot)$.

But what kind of signature?



$|W| = \binom{4}{2} = 6$

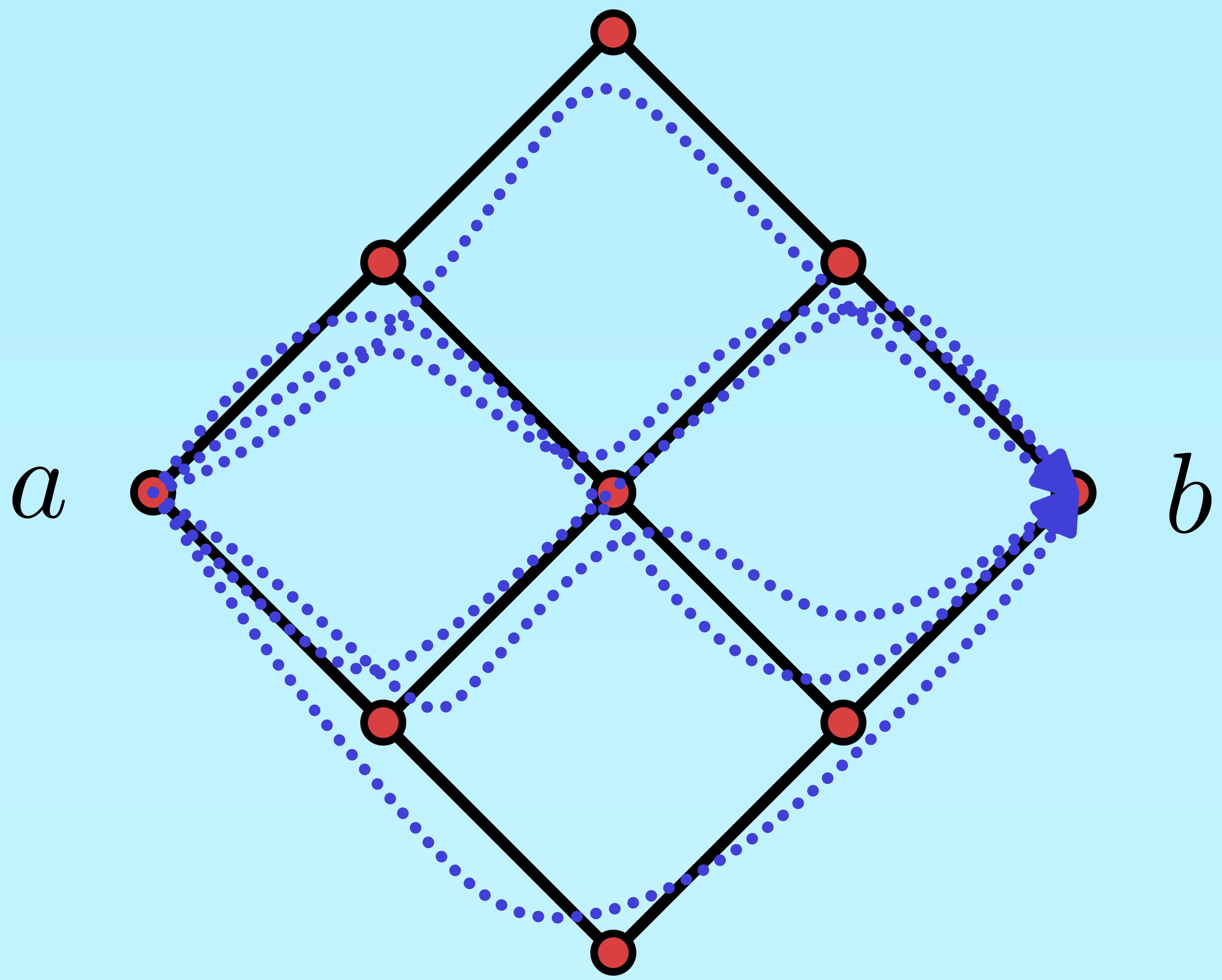
Idea: Maintain a signature of $W(\cdot,\cdot,\cdot)$.

But what kind of signature?

1. It should have small size.

2. It should still allow a recursive definition.

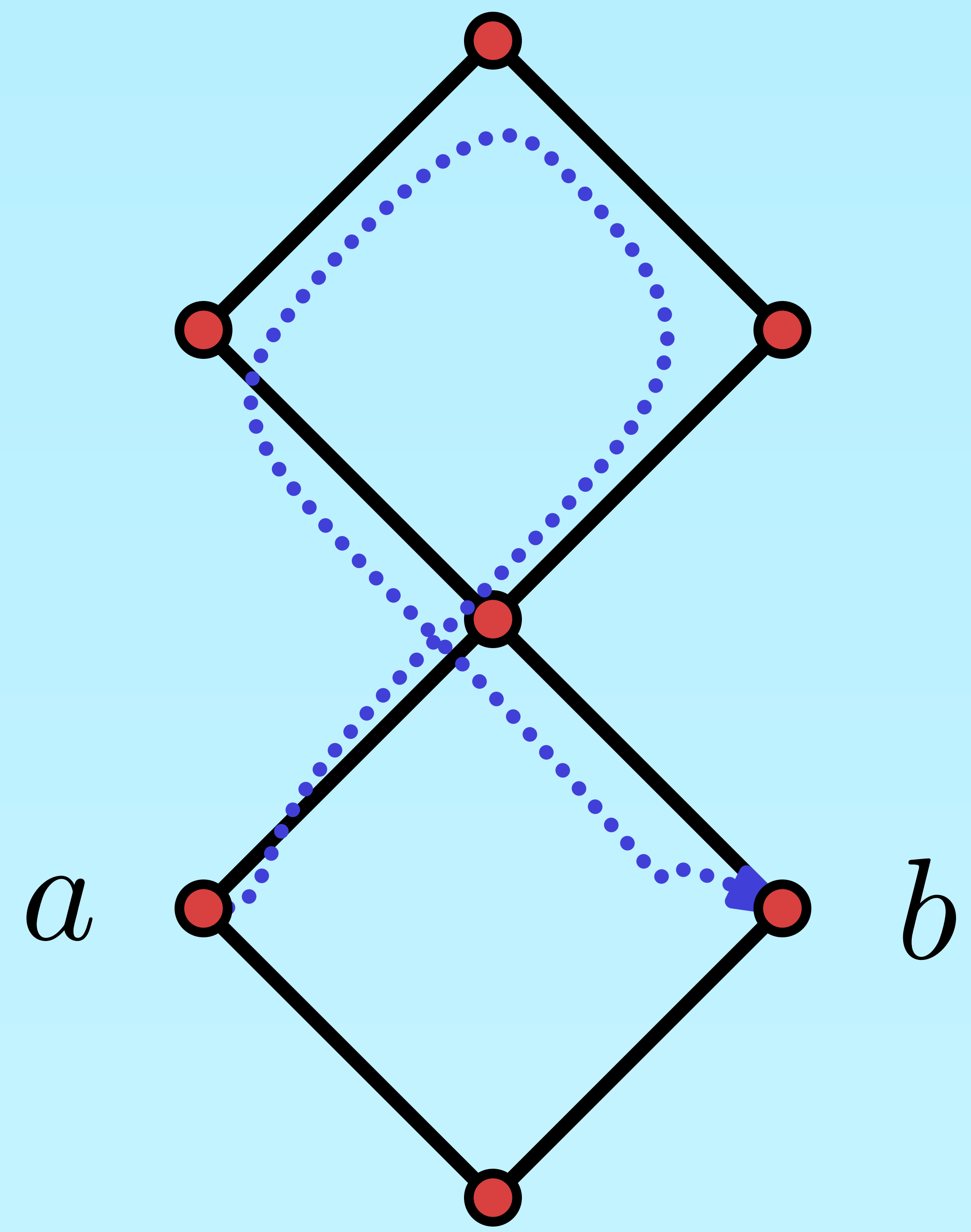
3. It should contain information about whether W contains a simple walk.



$|W| = \binom{4}{2} = 6$

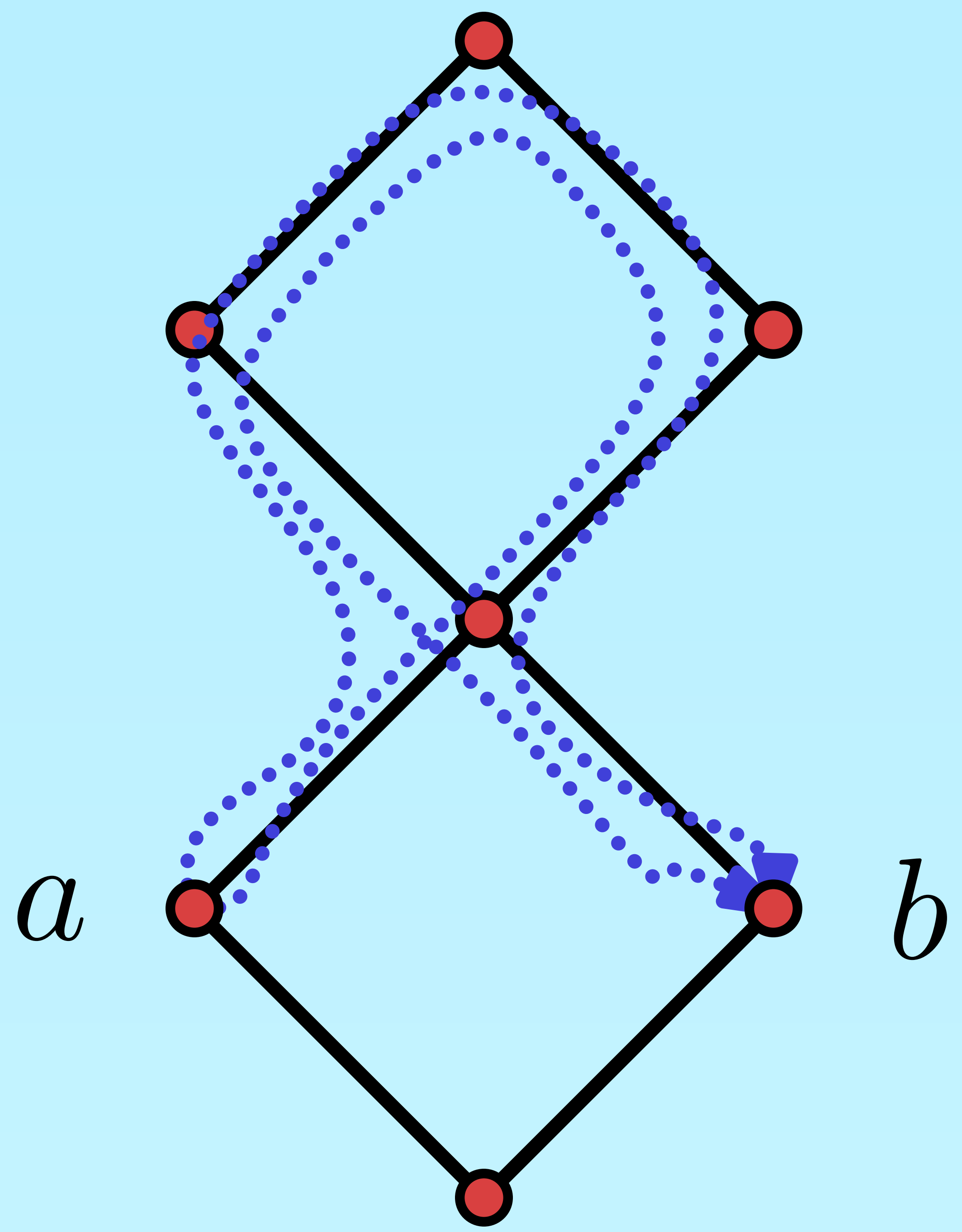
Idea: Maintain a signature of $W(\cdot,\cdot,\cdot)$.

Observation: all non-simple walks in W come in pairs.



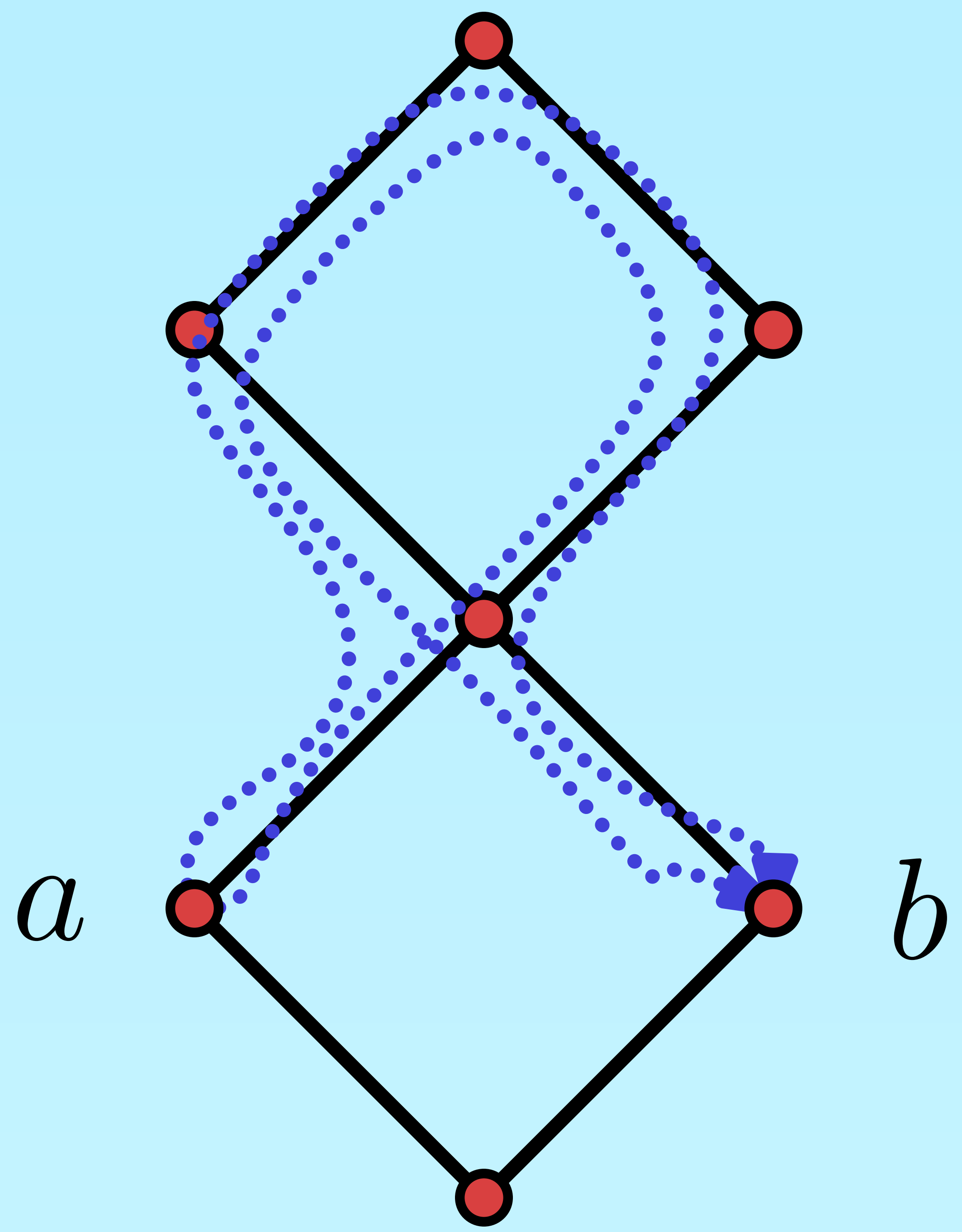
Idea: Maintain a signature of $W(\cdot,\cdot,\cdot)$.

Observation: all non-simple walks in W come in pairs.



Idea: Maintain a signature of $W(\cdot, \cdot, \cdot)$.

Observation: all non-simple walks in W come in pairs.

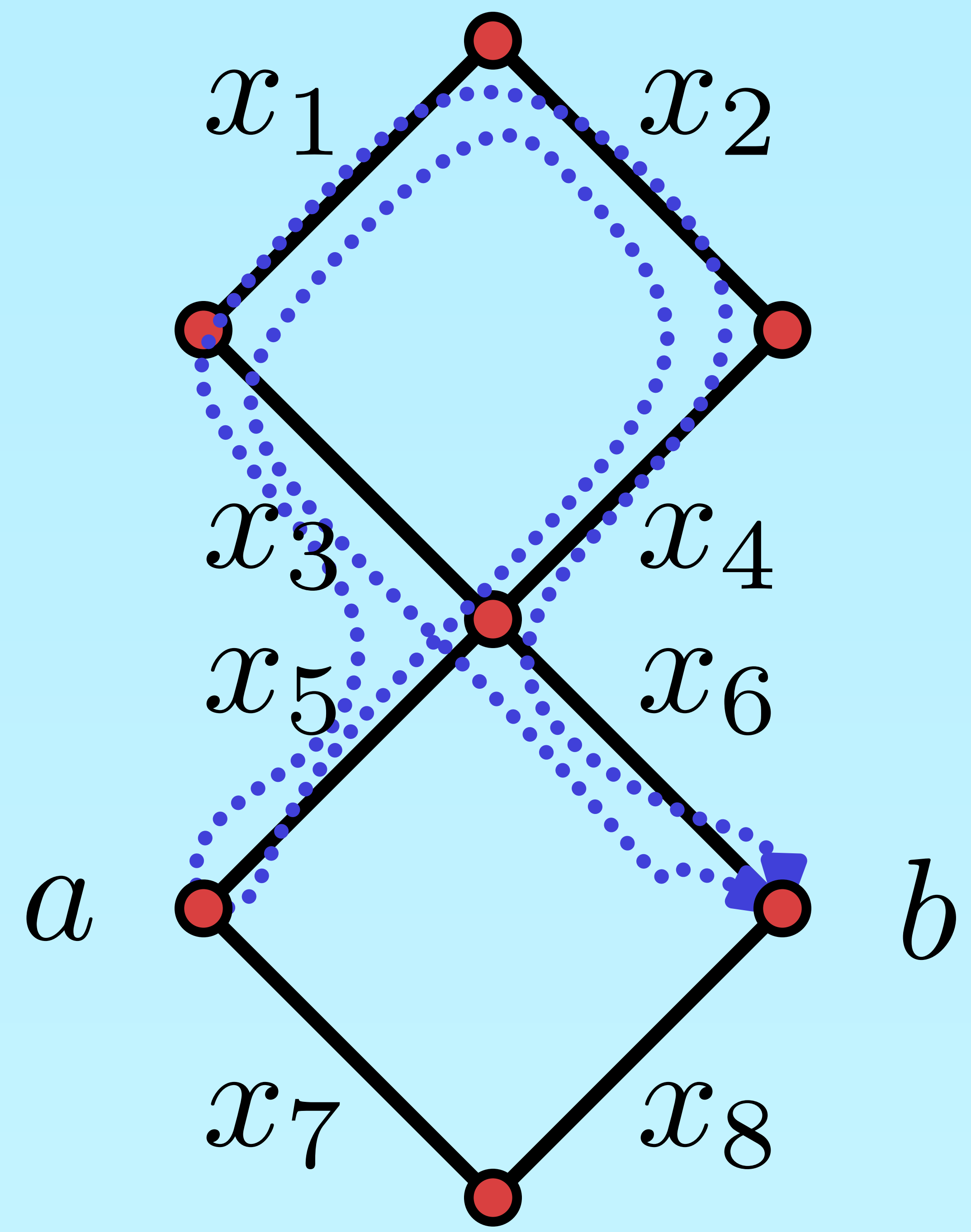


Idea: Maintain a signature of $W(\cdot,\cdot,\cdot)$.

Observation: all non-simple walks in W come in pairs.

Assign a variable to each edge.

For a walk w, let $\sigma(w)$ be the product of its edges.



$$\sigma(w_1) = x_5 x_4 x_2 x_1 x_3 x_6$$

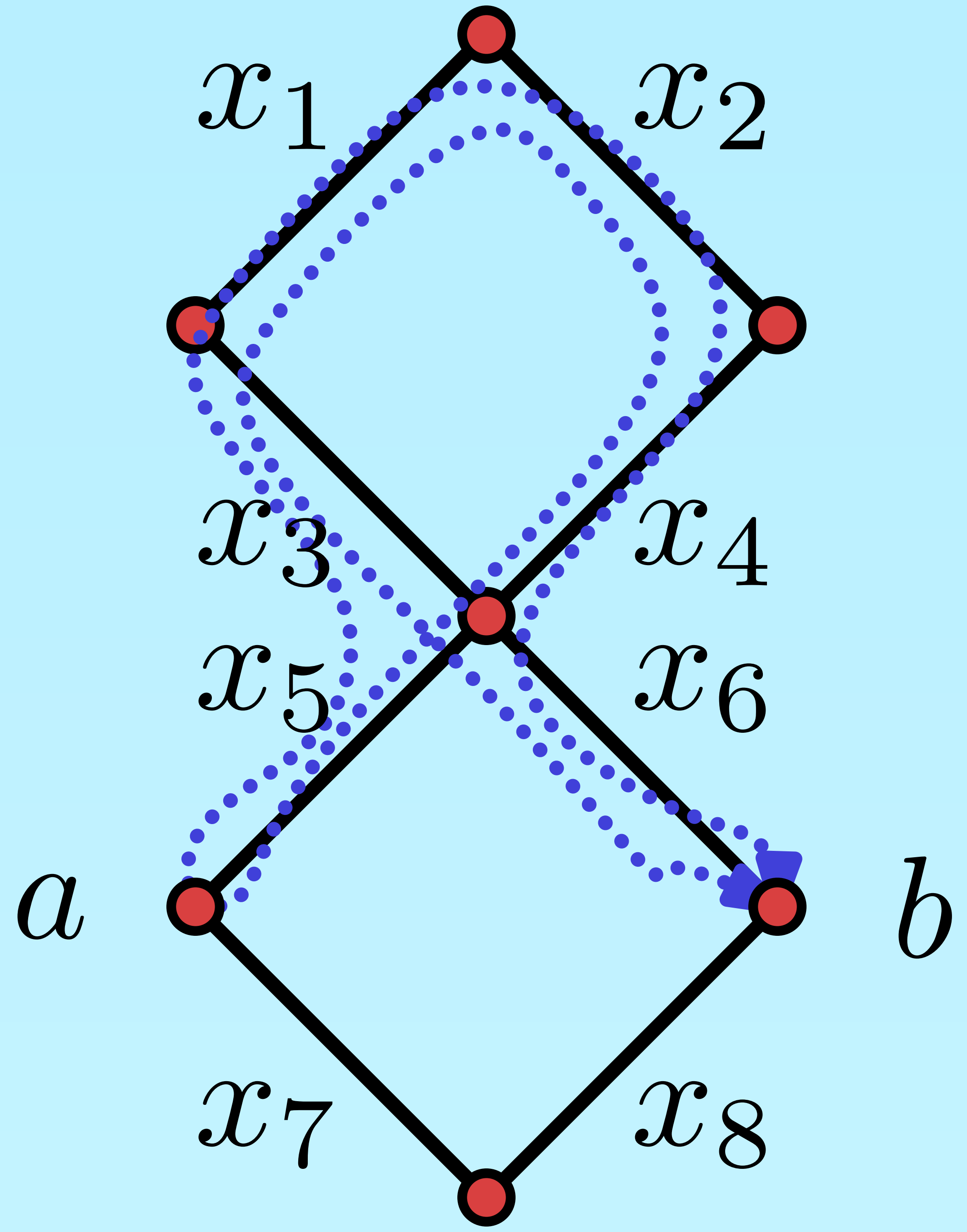
$$\sigma(w_2) = x_5 x_3 x_1 x_2 x_4 x_6$$

Idea: Maintain a signature of $W(\cdot,\cdot,\cdot)$.

Observation: all non-simple walks in W come in pairs.

Assign a variable to each edge.

For a walk w, let $\sigma(w)$ be the product of its edges.



$\sigma(w_1) = x_5 x_4 x_2 x_1 x_3 x_6$

$\sigma(w_2) = x_5 x_3 x_1 x_2 x_4 x_6$

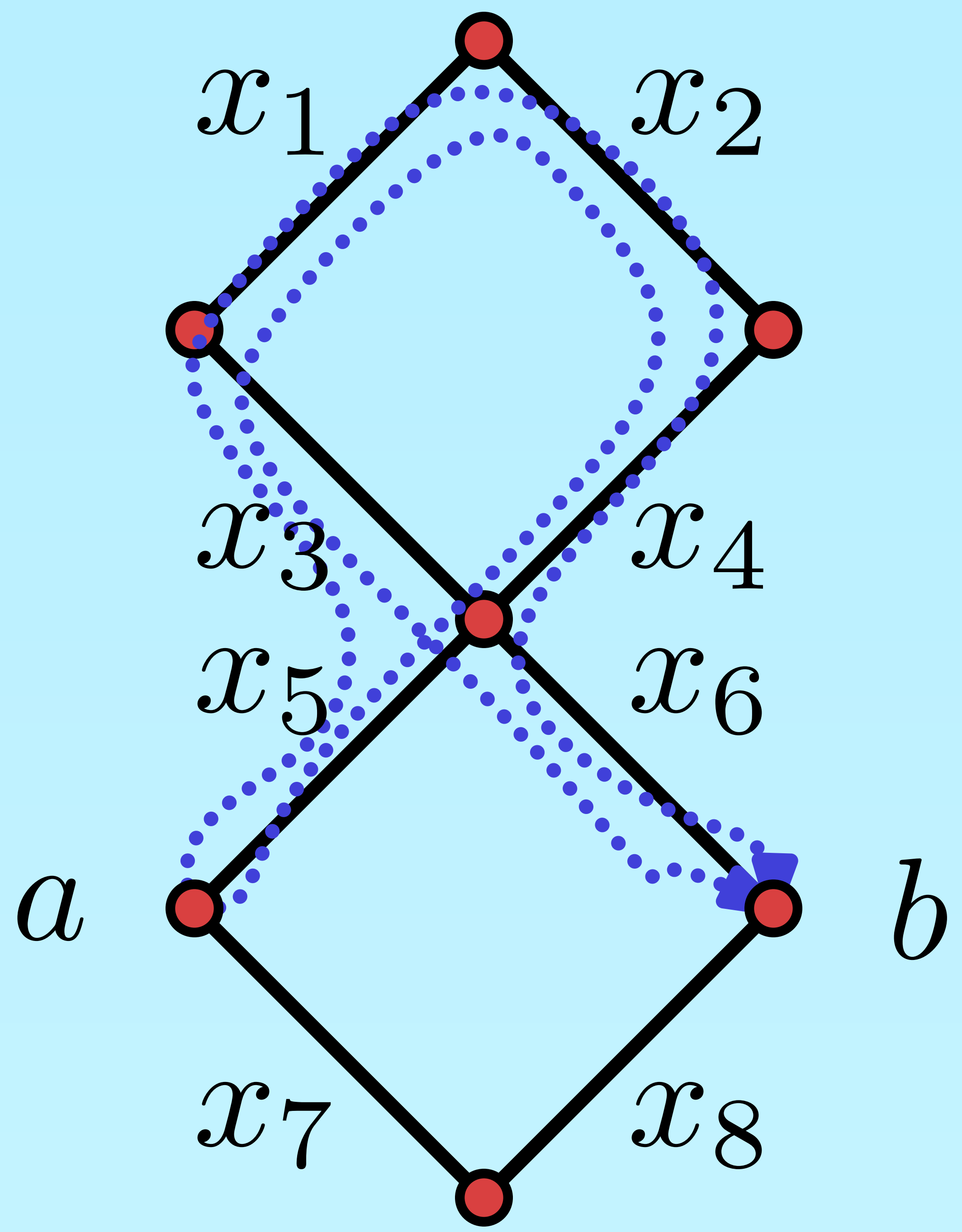
Idea: Maintain a signature of $W(\cdot, \cdot, \cdot)$.

Observation: all non-simple walks in W come in pairs.

Assign a variable to each edge.

For a walk w, let $\sigma(w)$ be the product of its edges.

Now, let $\sigma(W) = \sum_{w \in W} \sigma(w)$



$\sigma(w_1) = x_5x_4x_2x_1x_3x_6$

$\sigma(w_2) = x_5x_3x_1x_2x_4x_6$

Idea: Maintain a signature of $W(\cdot,\cdot,\cdot)$.

Observation: all non-simple walks in W come in pairs.

Assign a variable to each edge.

For a walk w, let $\sigma(w)$ be the product of its edges.

Now, let $\sigma(W) = \sum_{w \in W} \sigma(w)$

Does $\sigma(W)$ have an odd term?

Trick: Replace each x_i be an element of the finite field $\mathbb{F}_{2^s}$

Trick: Replace each x_i be an element of the finite field $\mathbb{F}_{2^s}$

You can think of these elements as bit vectors.

$$\mathbb{F}_{2^8} = \{ 01001010, 11010110, 00011011, 01111000, \ldots$$

Trick: Replace each x_i be an element of the finite field $\mathbb{F}_{2^s}$

You can think of these elements as bit vectors.

$$\mathbb{F}_{2^8} = \{\ 01001010,\ 11010110,\ 00011011,\ 01111000,\ \ldots$$

Trick: Replace each x_i be an element of the finite field $\mathbb{F}_{2^s}$

You can think of these elements as bit vectors.

Addition is bitwise XOR.

$$\mathbb{F}_{2^8} = \{ 01001010, 11010110, 00011011, 01111000, \dots$$

$$01001010 + 11010110 = 10011100$$

Trick: Replace each x_i be an element of the finite field $\mathbb{F}_{2^s}$

You can think of these elements as bit vectors.

Addition is bitwise XOR.

$$\mathbb{F}_{2^8} =$$

$$\{ 01001010, 11010110, 00011011, 01111000, \dots$$

$$01001010 + 11010110 = 10011100$$

Trick: Replace each x_i be an element of the finite field $\mathbb{F}_{2^s}$

You can think of these elements as bit vectors.

Addition is bitwise XOR.

Multiplication is, well, you consider the bits to be the coefficients of a polynomial, and then you multiply the polynomials, then divide the resulting polynomial by a certain irreducible polynomial of degree s, and then the new element is essentially the remainder of that division.

$$\mathbb{F}_{2^8} =$$

$$\{ \quad 01001010, \quad 11010110, \quad 00011011, \quad 01111000, \quad \ldots$$

$$01001010 + 11010110 = 10011100$$

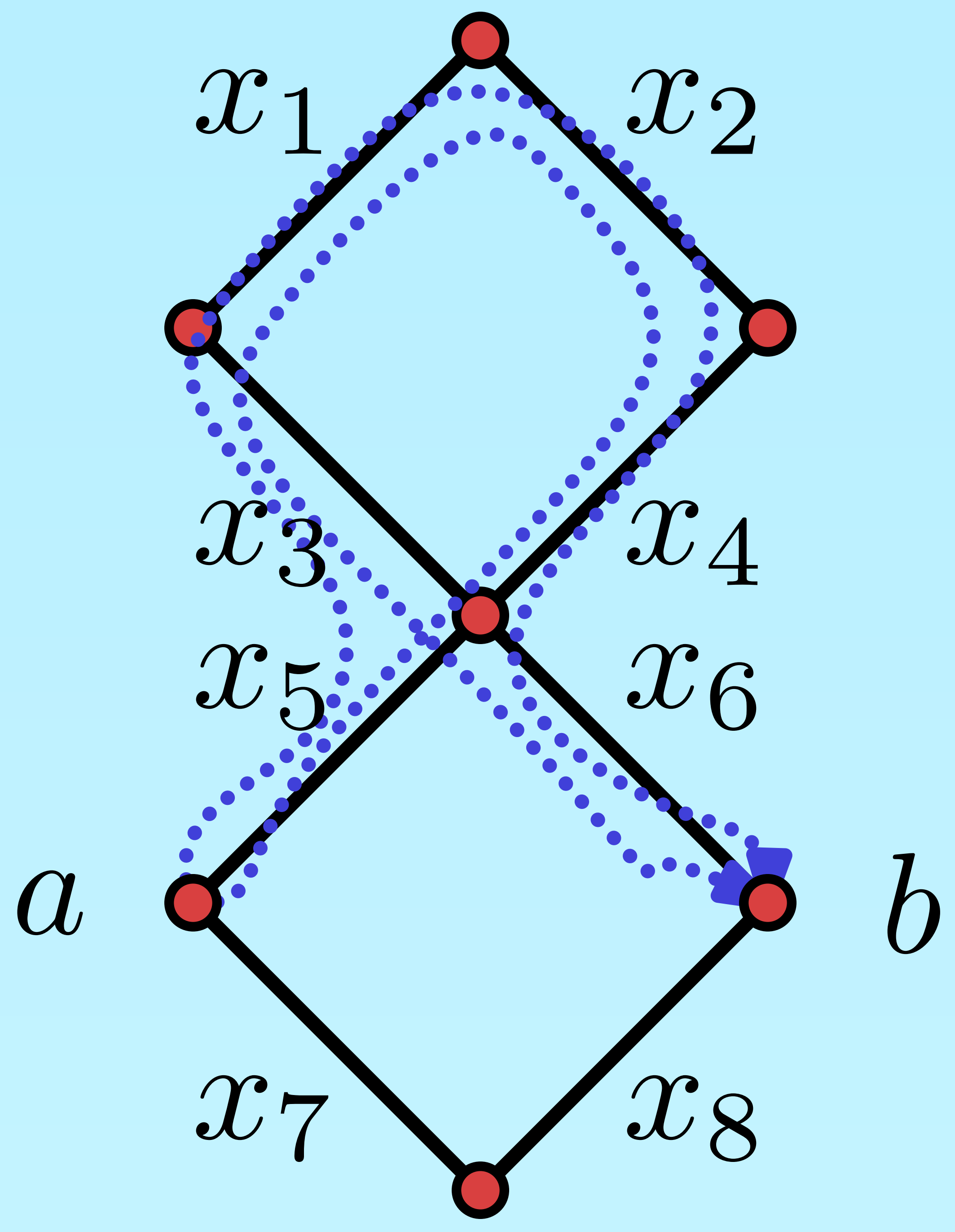
$$01001010 \times 11010110 = 01011101$$

Trick: Replace each x_i be an element of the finite field $\mathbb{F}_{2^s}$

You can think of these elements as bit vectors.

Addition is bitwise XOR.

Multiplication is, well, you consider the bits to be the coefficients of a polynomial, and then you multiply the polynomials, then divide the resulting polynomial by a certain irreducible polynomial of degree s, and then the new element is essentially the remainder of that division.



$$\sigma(w_1) = x_5 x_4 x_2 x_1 x_3 x_6$$

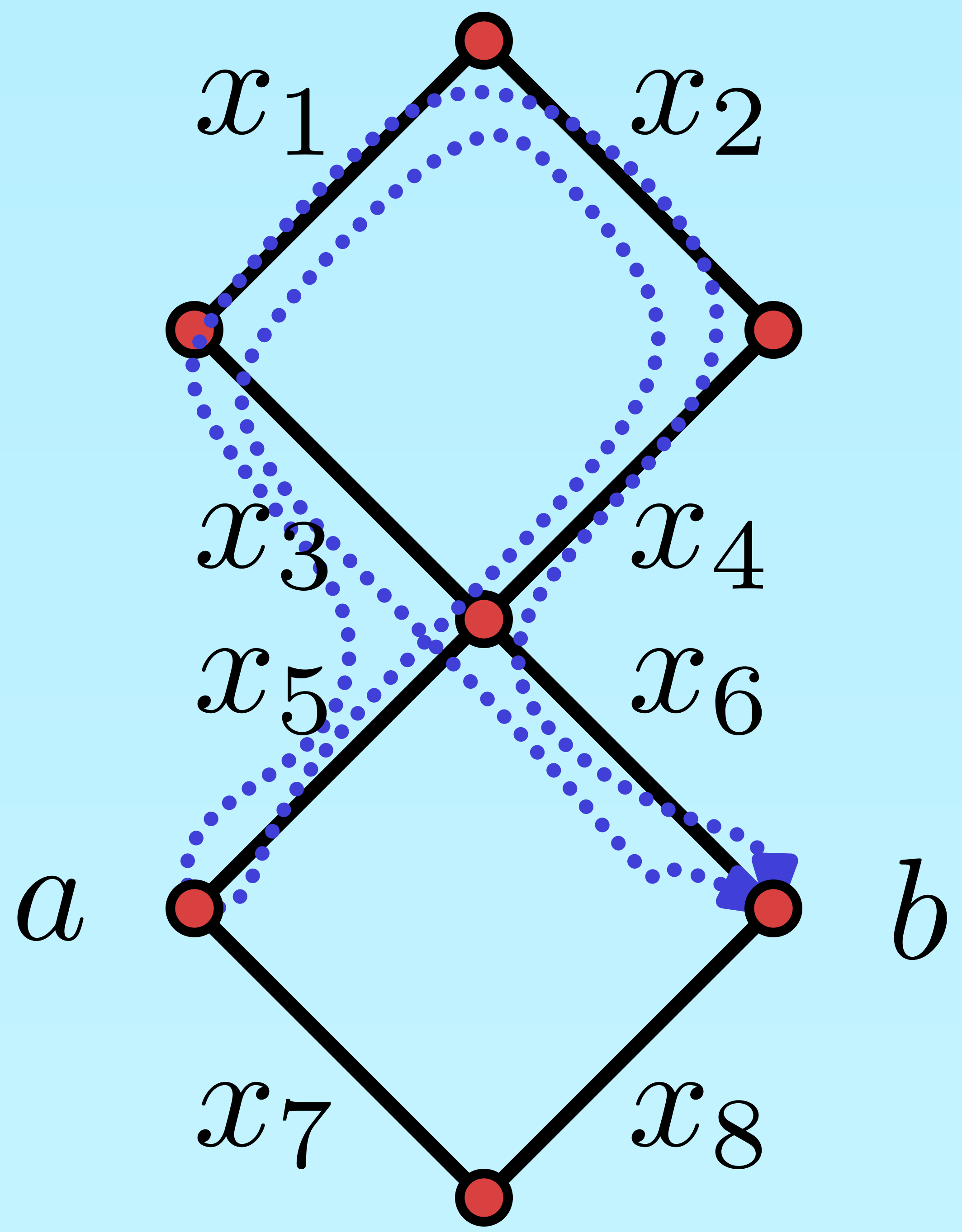
$$\sigma(w_2) = x_5 x_3 x_1 x_2 x_4 x_6$$

Trick: Replace each x_i be an element of the finite field $\mathbb{F}_{2^s}$

You can think of these elements as bit vectors.

Addition is bitwise XOR.

Multiplication is, well, you consider the bits to be the coefficients of a polynomial, and then you multiply the polynomials, then divide the resulting polynomial by a certain irreducible polynomial of degree s, and then the new element is essentially the remainder of that division.



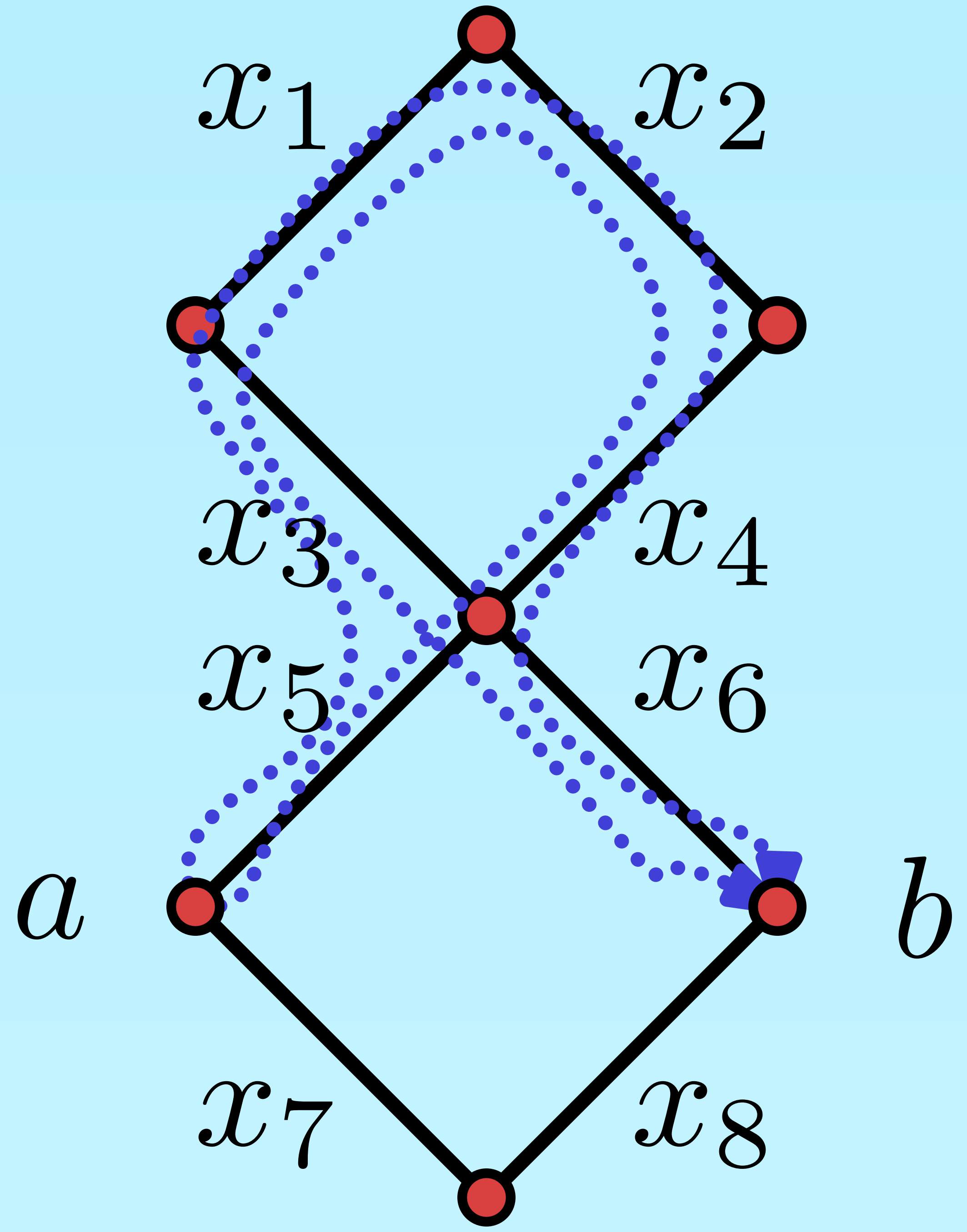
$$\sigma(w_1) + \sigma(w_2) = 0$$

Trick: Replace each x_i be an element of the finite field $\mathbb{F}_{2^s}$

You can think of these elements as bit vectors.

Addition is bitwise XOR.

Multiplication is, well, you consider the bits to be the coefficients of a polynomial, and then you multiply the polynomials, then divide the resulting polynomial by a certain irreducible polynomial of degree s, and then the new element is essentially the remainder of that division.



$$\sigma(w_1) + \sigma(w_2) = 0$$

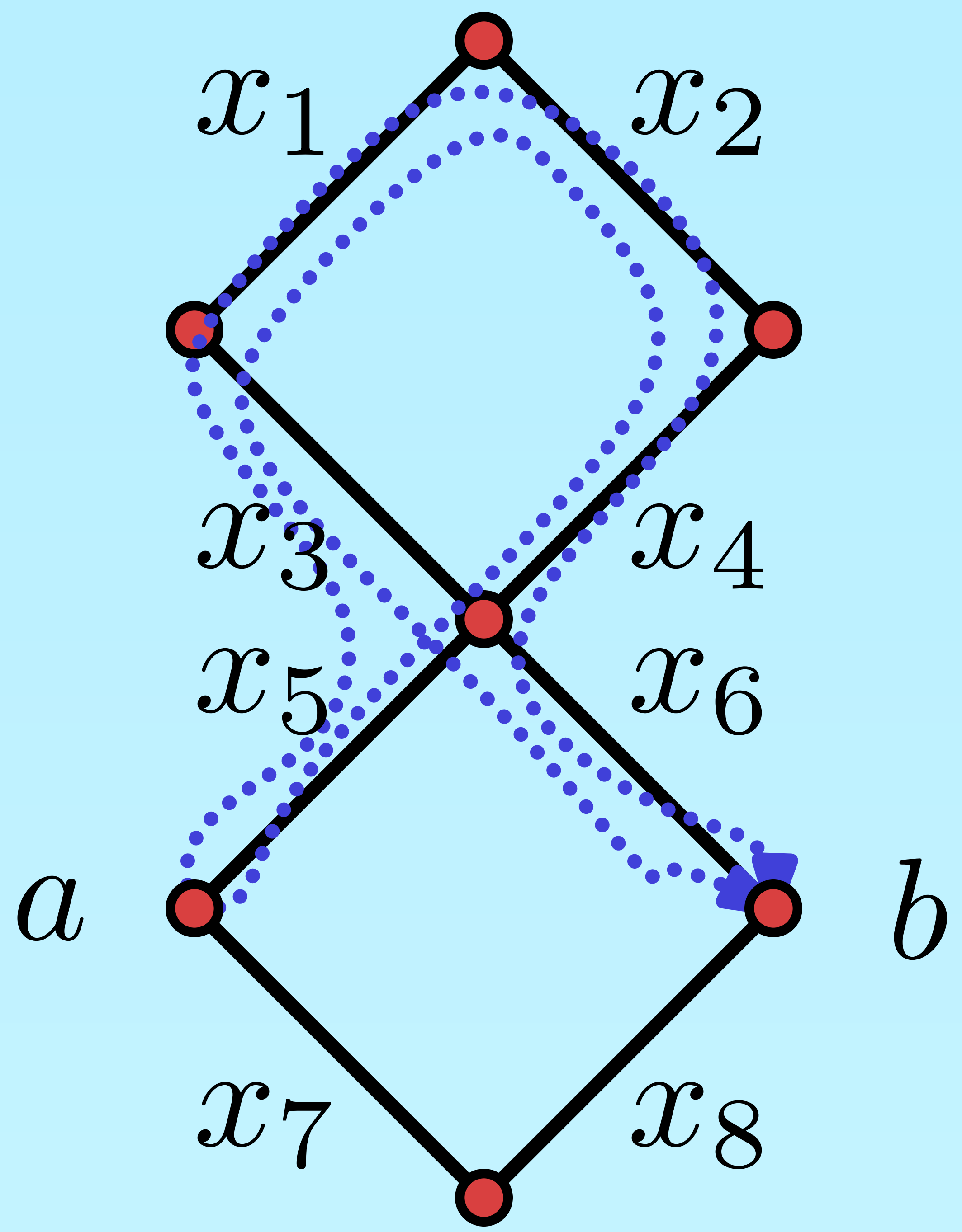
$$\sigma(\{w_1, w_2\}) = 0$$

Trick: Replace each x_i be an element of the finite field $\mathbb{F}_{2^s}$

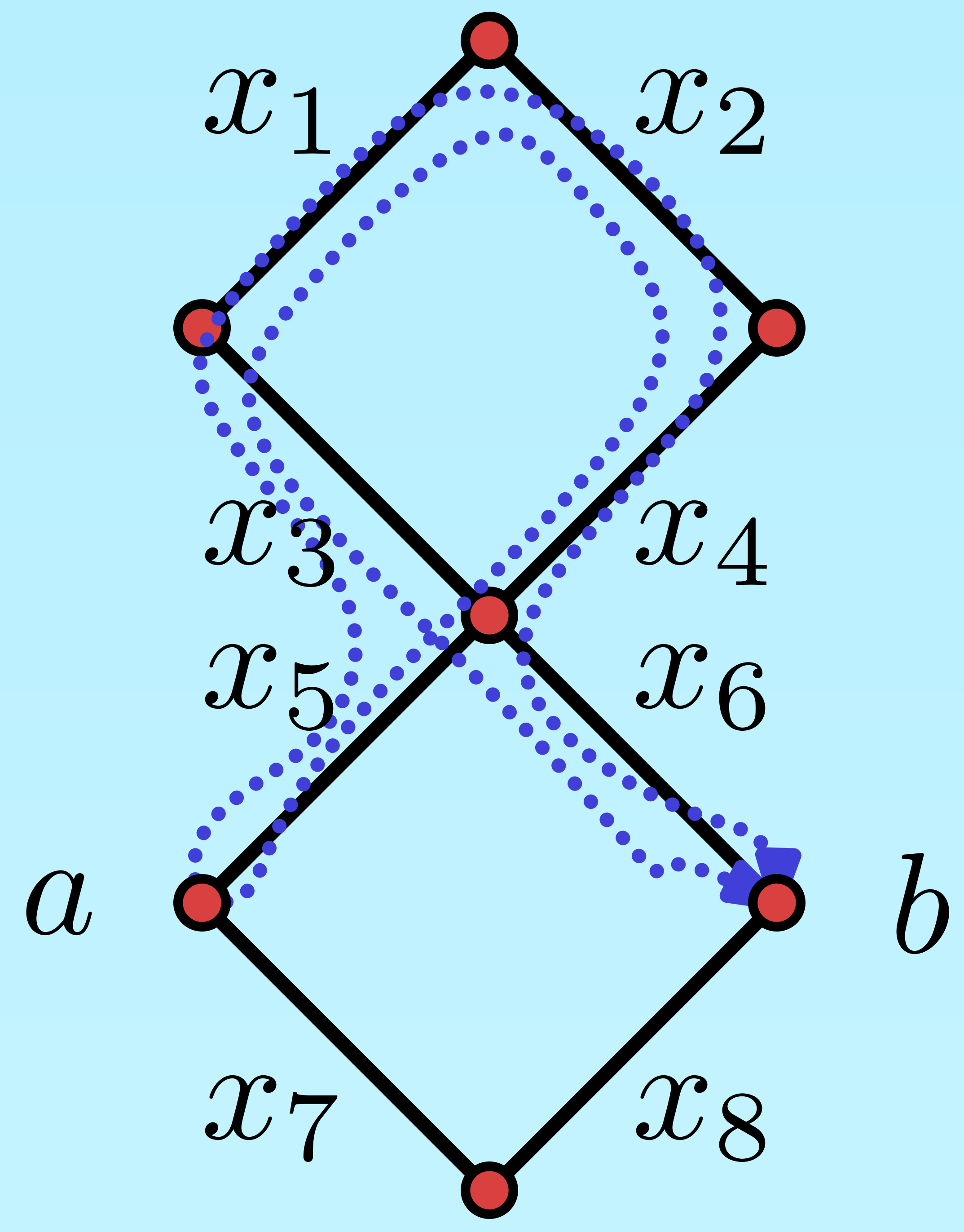
You can think of these elements as bit vectors.

Addition is bitwise XOR.

Multiplication is, well, you consider the bits to be the coefficients of a polynomial, and then you multiply the polynomials, then divide the resulting polynomial by a certain irreducible polynomial of degree s, and then the new element is essentially the remainder of that division.

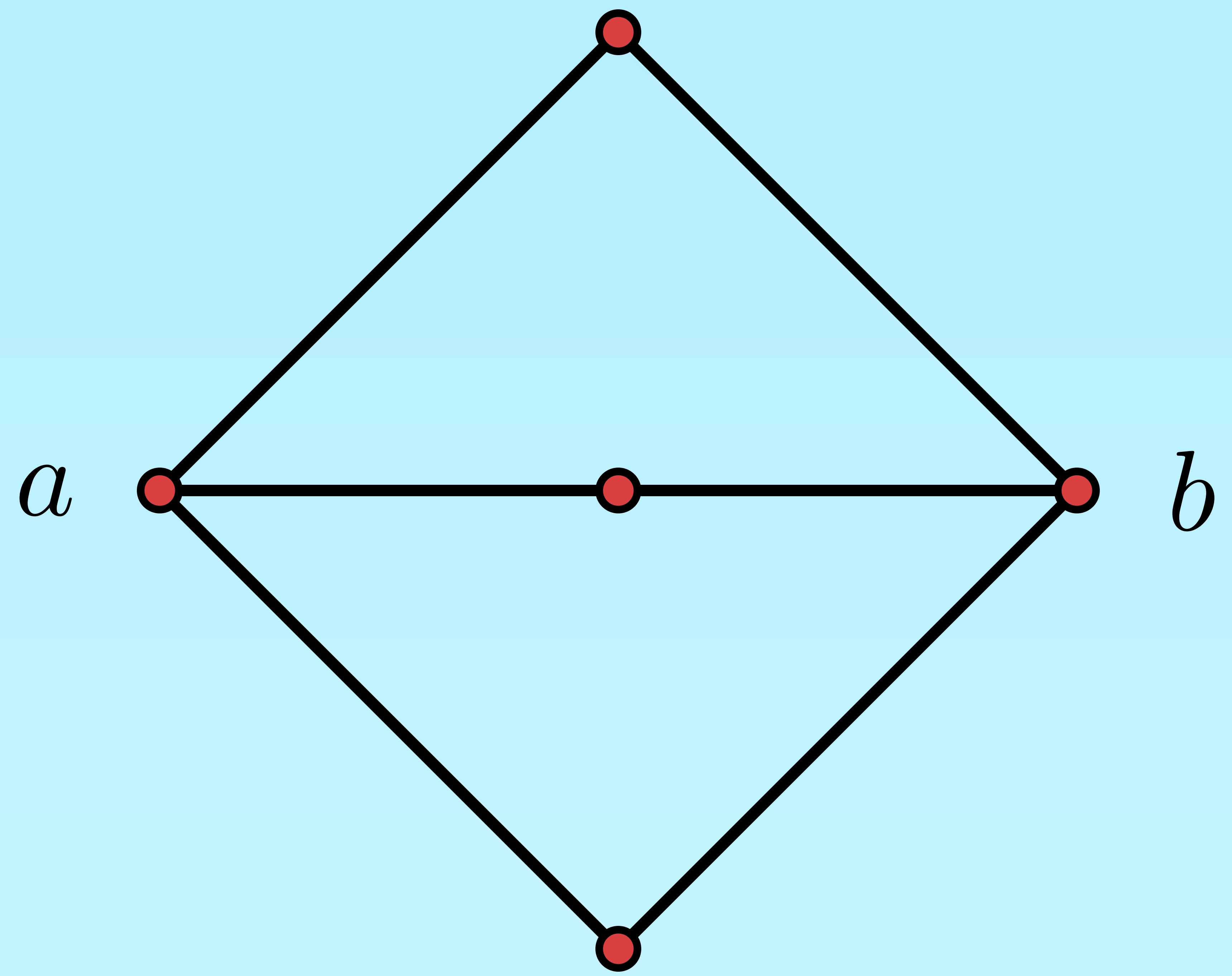


If W contains no simple walk, then $\sigma(W) = 0$.



If W contains no simple walk, then $\sigma(W) = 0$.

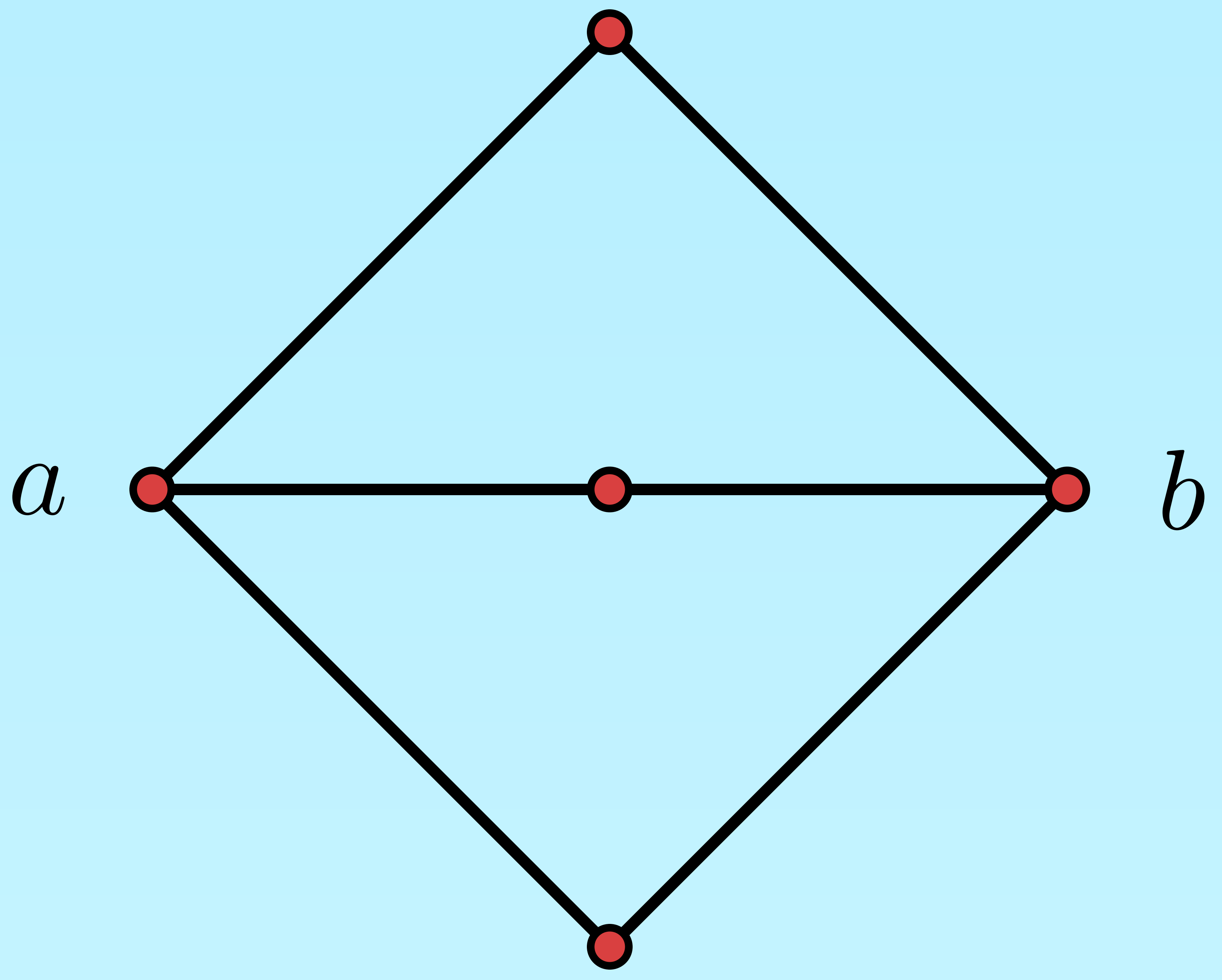
If W does contain a simple walk, then can $\sigma(W)$ be 0?



If W contains no simple walk, then $\sigma(W) = 0$.

If W does contain a simple walk, then can $\sigma(W)$ be 0?

Yes! A sum of three different elements might be 0.

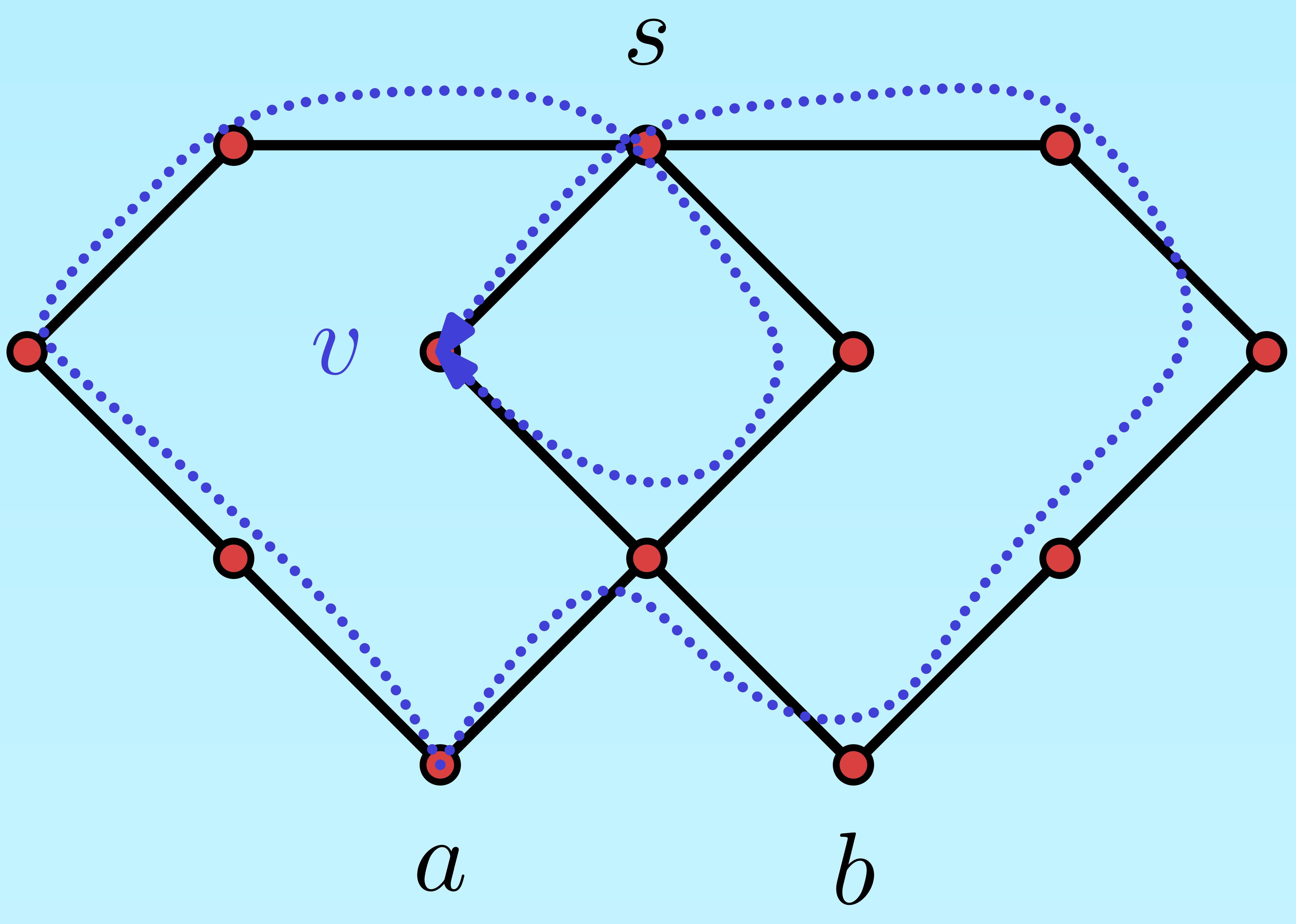


If W contains no simple walk, then $\sigma(W) = 0$.

If W does contain a simple walk, then can $\sigma(W)$ be 0?

Yes! A sum of three different elements might be 0.

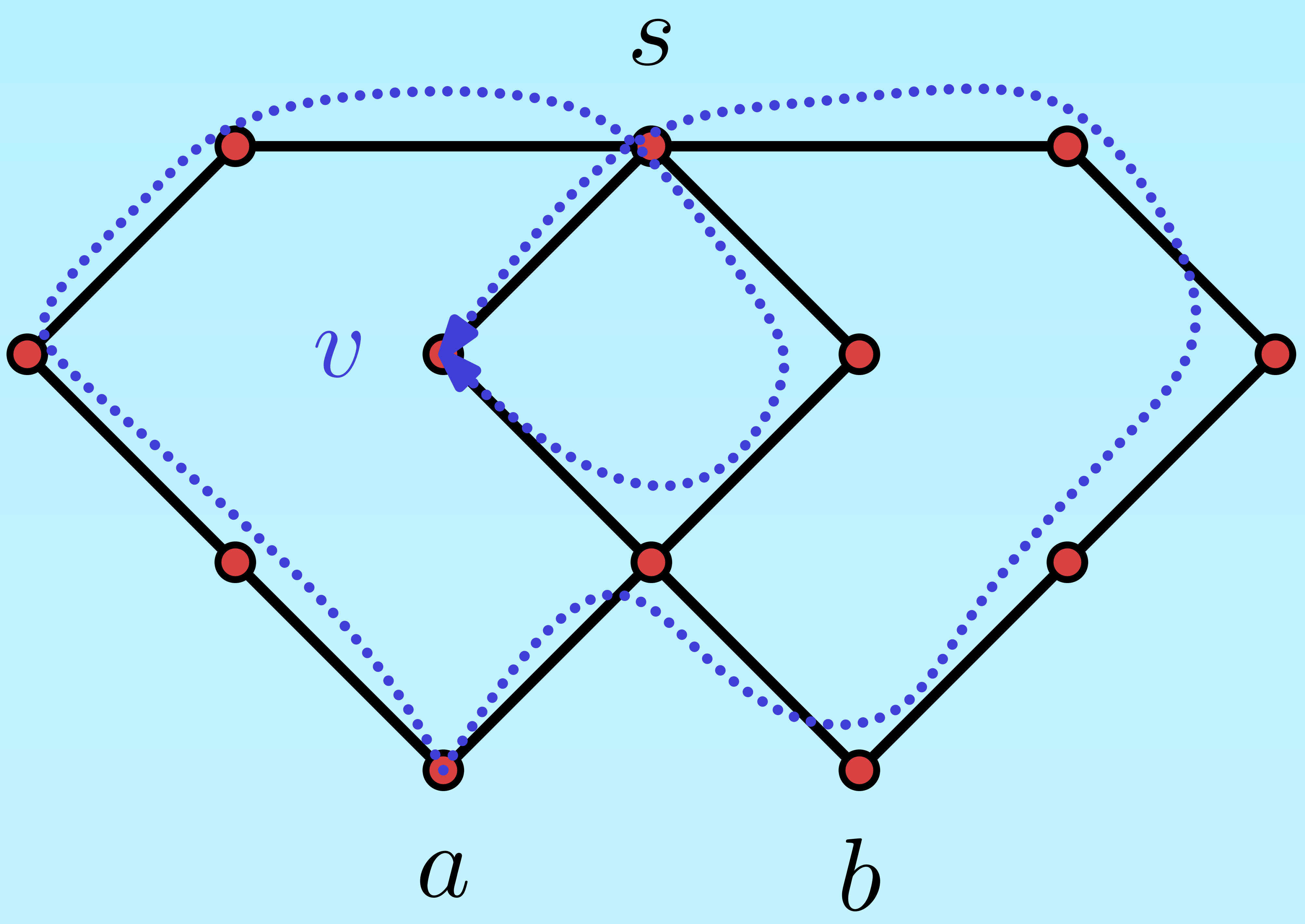
Claim: If we pick random weights, the probability that this happens is small.



Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Let $W(v, \ell, \text{yes/no})$ be the set of all spurless walks from a to v of length ℓ that do/don't pass through s.

Then we want to find the smallest ℓ for which $W(b, \ell, \text{yes})$ contains a simple walk.

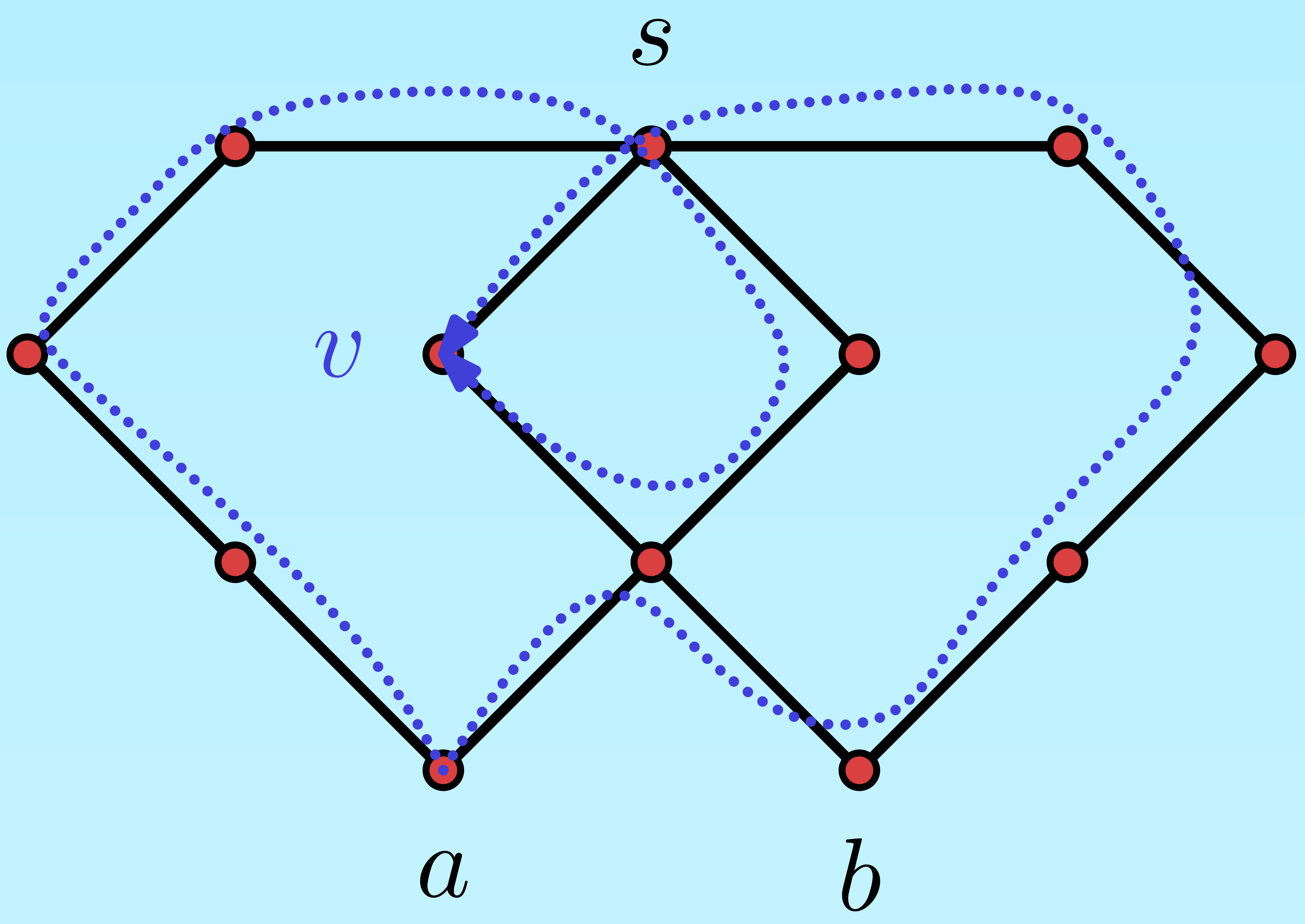


$\sigma(\ W(v, 7, \text{yes})\ ) \neq 0$

Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Maintain $\sigma(W(v, \ell, \text{yes/no}))$, the signature of W.

Then we want to find the smallest ℓ for which $W(b, \ell, \text{yes})$ contains a simple walk.

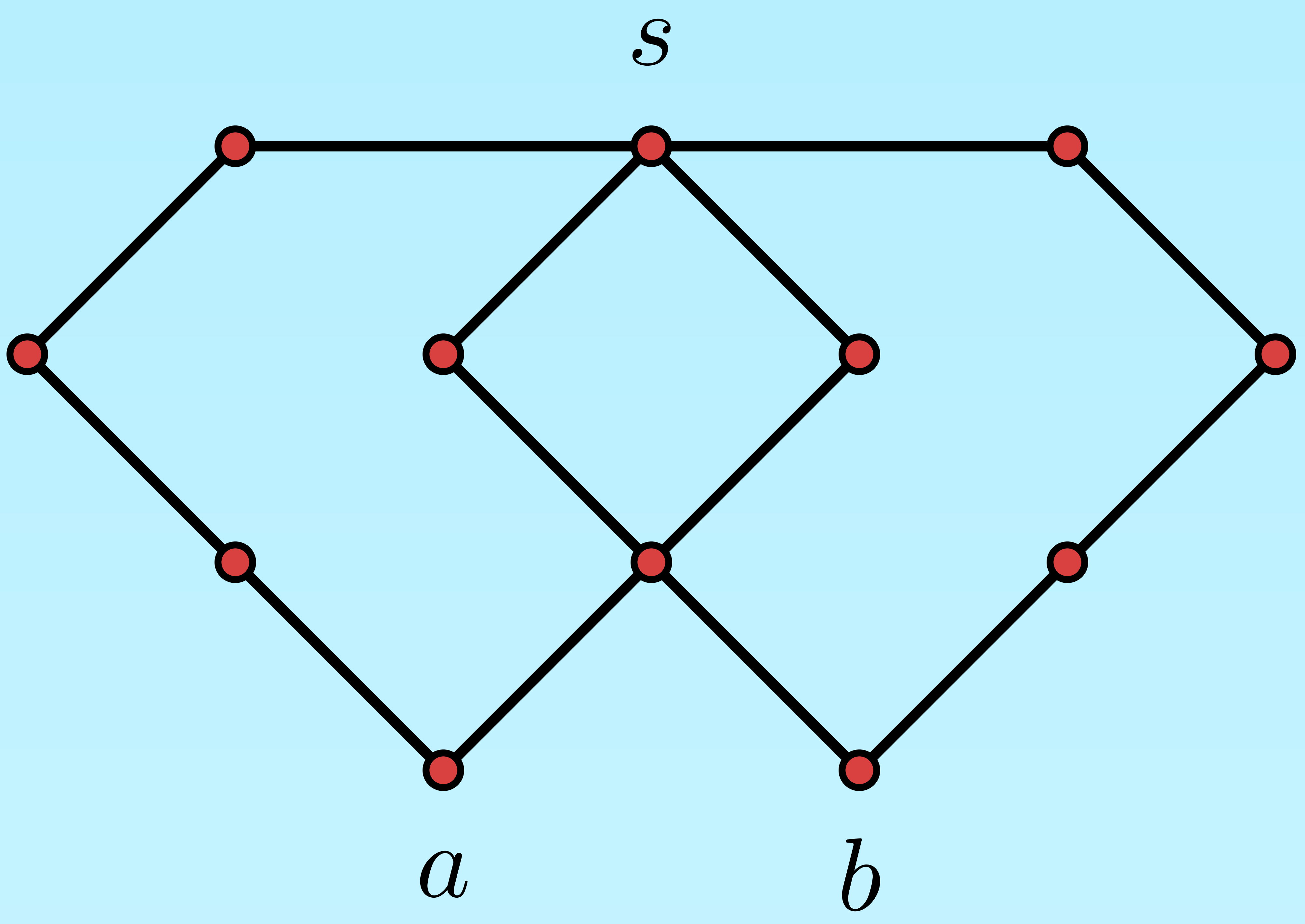


$\sigma(\ W(v, 7, \text{yes})\ ) \neq 0$

Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Maintain $\sigma(W(v, \ell, \text{yes/no}))$, the signature of W.

Then we want to find the smallest ℓ for which $\sigma(W(b, \ell, \text{yes})) \neq 0$.

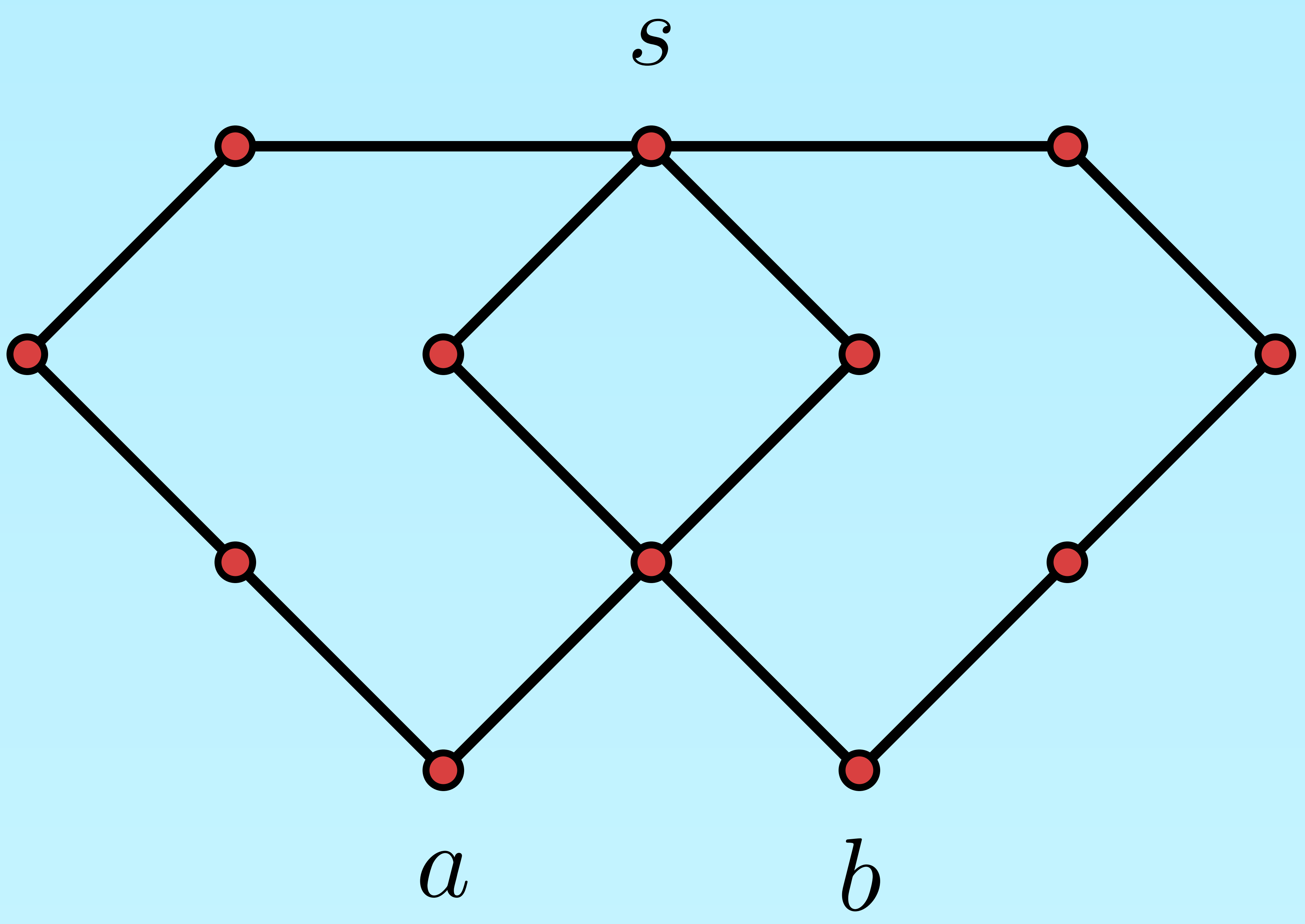


$$\sigma(W(b, 1, \text{yes})) = 0$$

Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Maintain $\sigma(W(v, \ell, \text{yes/no}))$, the signature of W.

Then we want to find the smallest ℓ for which $\sigma(W(b, \ell, \text{yes})) \neq 0$.

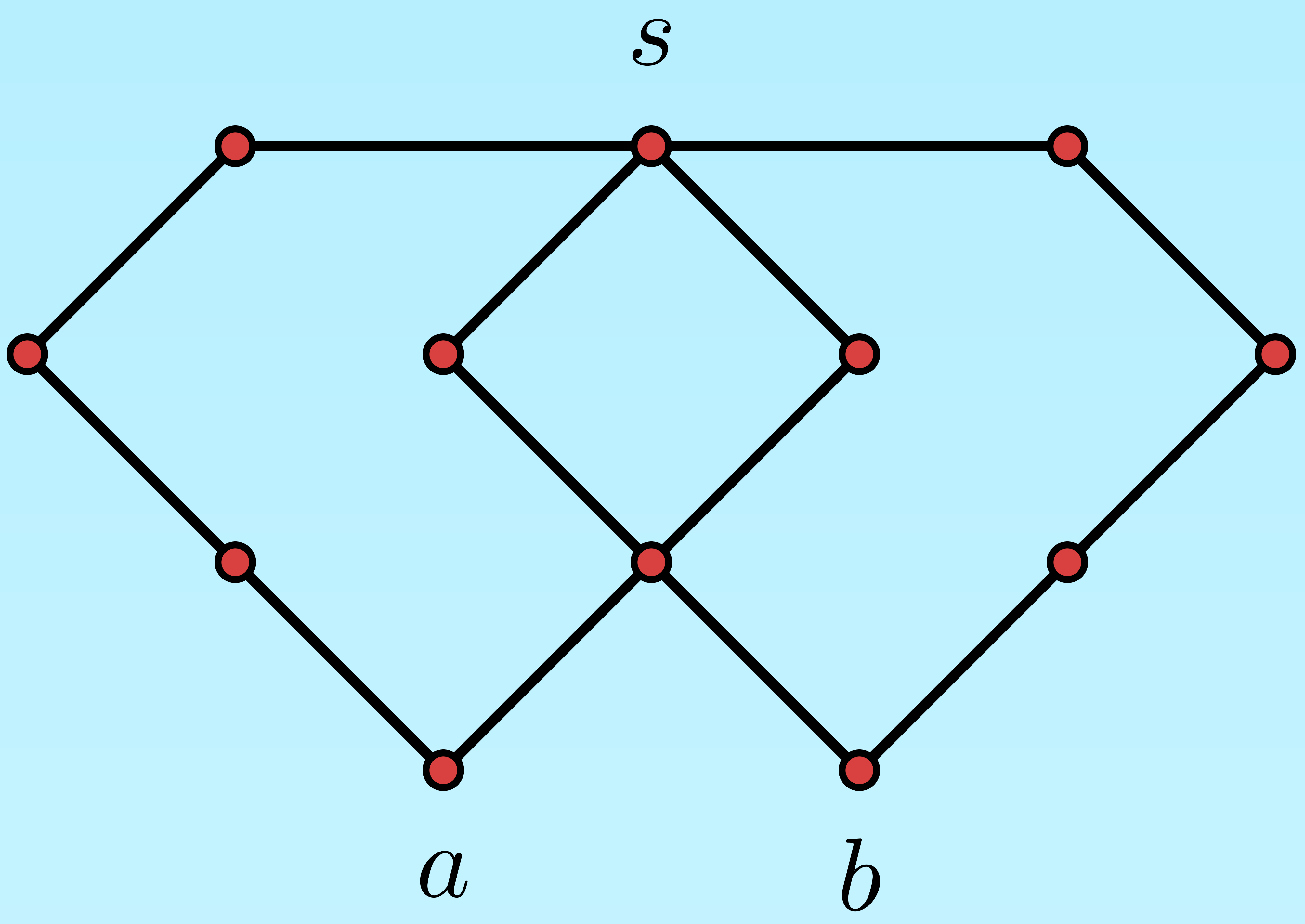


$\sigma(W(b, 2, \text{yes})) = 0$

Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Maintain $\sigma(W(v, \ell, \text{yes/no}))$, the signature of W.

Then we want to find the smallest ℓ for which $\sigma(W(b, \ell, \text{yes})) \neq 0$.

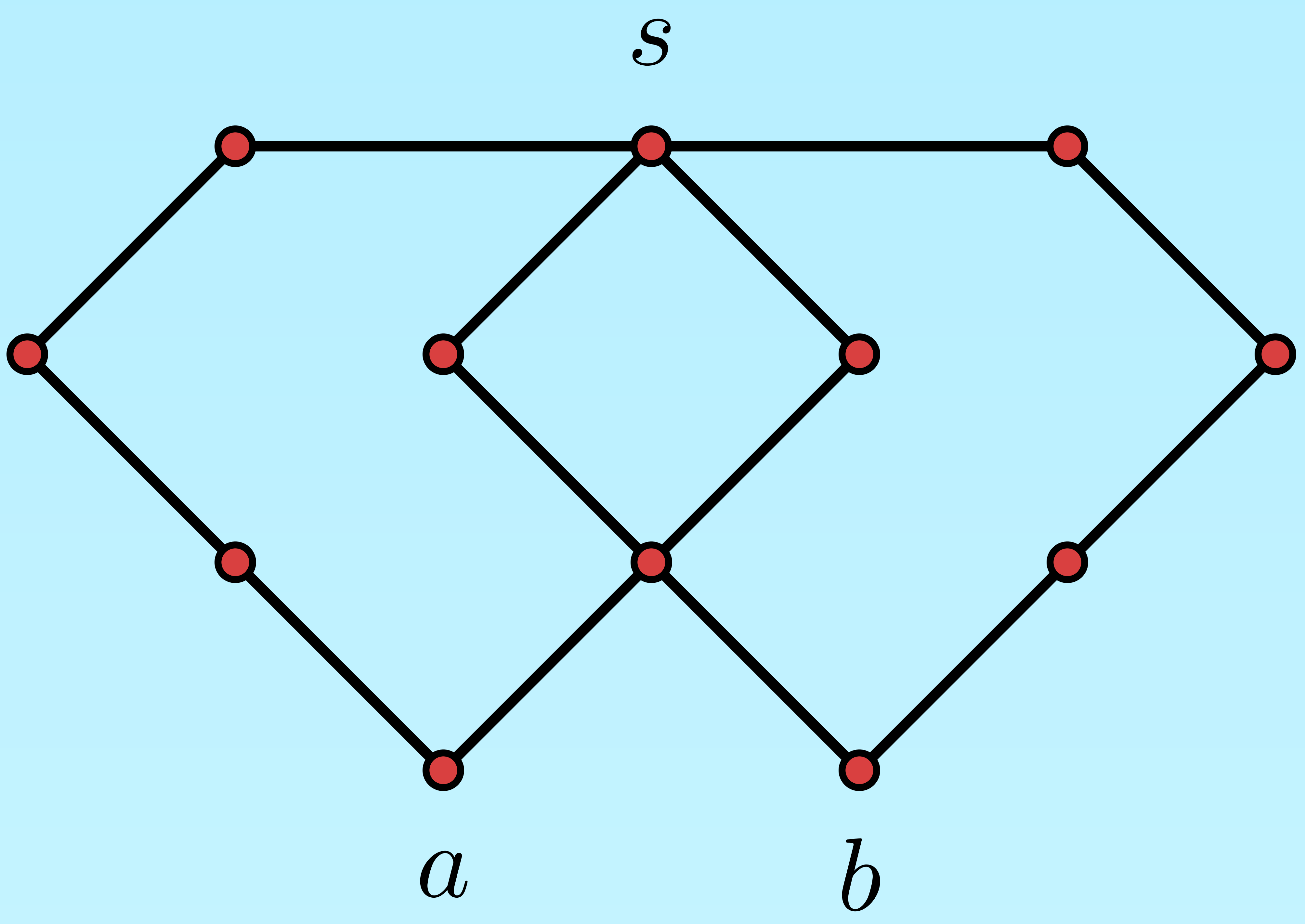


$\sigma(W(b, 3, \text{yes})) = 0$

Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Maintain $\sigma(W(v, \ell, \text{yes/no}))$, the signature of W.

Then we want to find the smallest ℓ for which $\sigma(W(b, \ell, \text{yes})) \neq 0$.

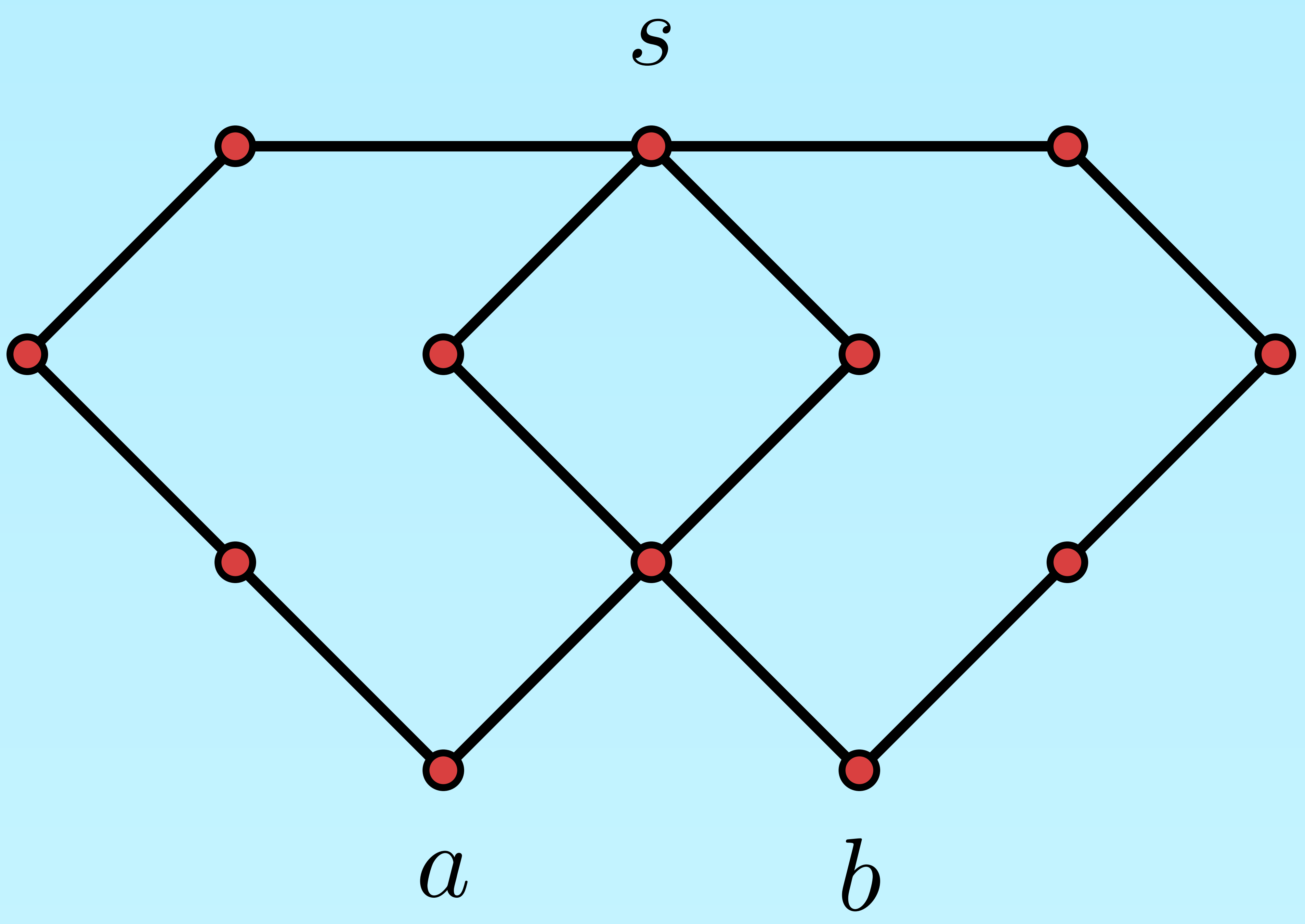


$\sigma(W(b, 4, \text{yes})) = 0$

Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Maintain $\sigma(W(v, \ell, \text{yes/no}))$, the signature of W.

Then we want to find the smallest ℓ for which $\sigma(W(b, \ell, \text{yes})) \neq 0$.

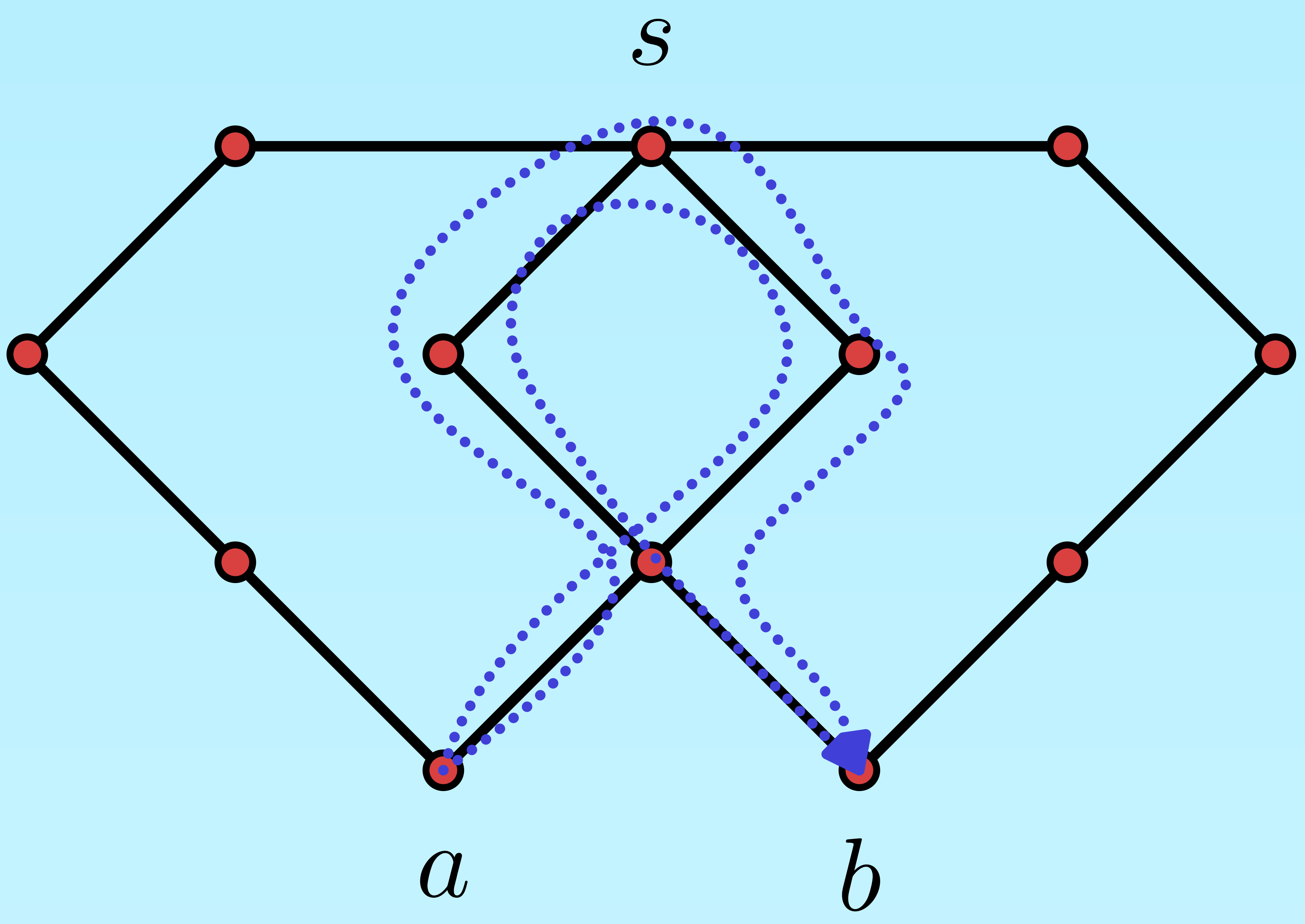


$$\sigma(W(b, 5, \text{yes})) = 0$$

Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Maintain $\sigma(W(v, \ell, \text{yes/no}))$, the signature of W.

Then we want to find the smallest ℓ for which $\sigma(W(b, \ell, \text{yes})) \neq 0$.

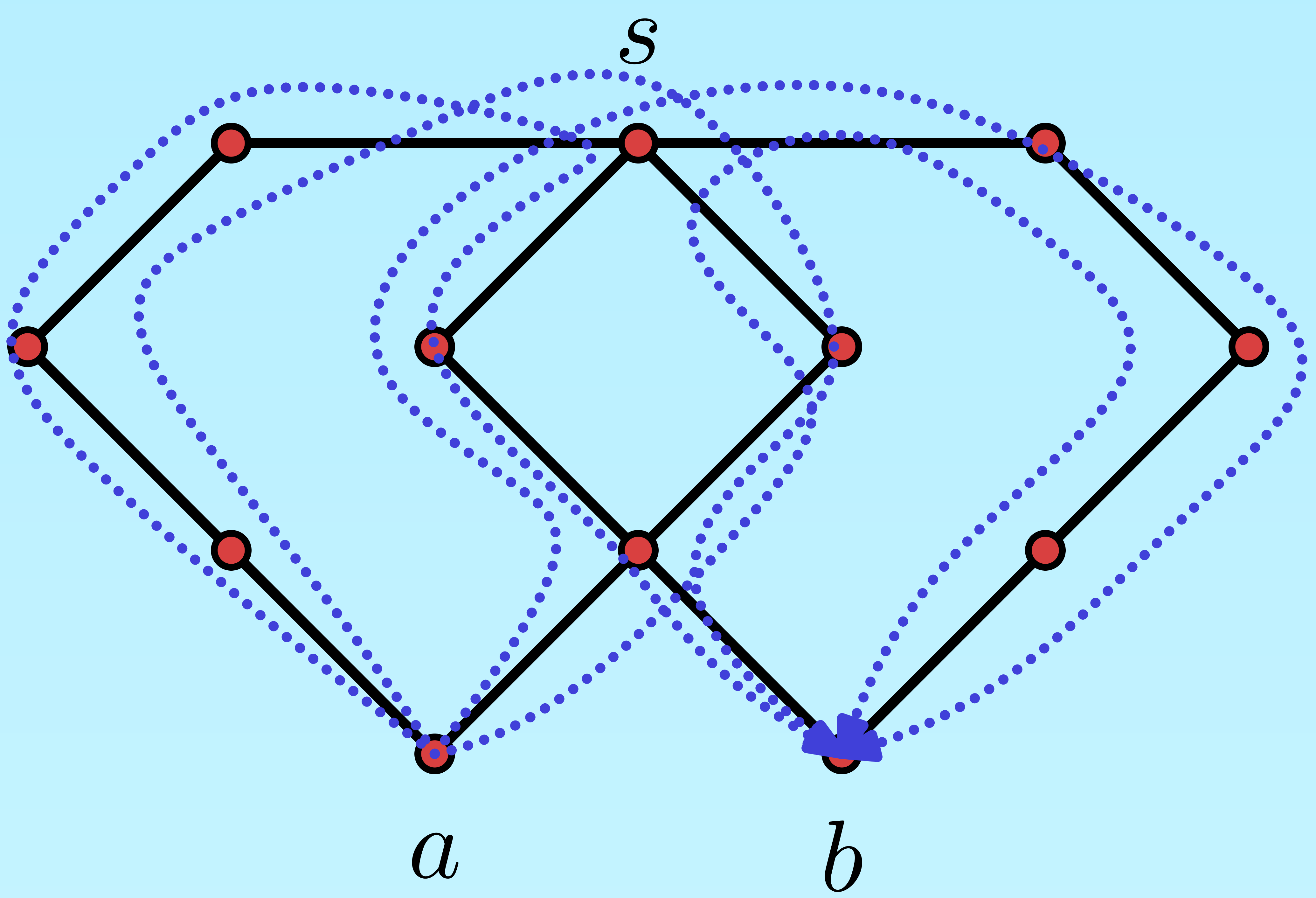


$\sigma(W(b, 6, \text{yes})) = 0$

Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Maintain $\sigma(W(v, \ell, \text{yes/no}))$, the signature of W.

Then we want to find the smallest ℓ for which $\sigma(W(b, \ell, \text{yes})) \neq 0$.

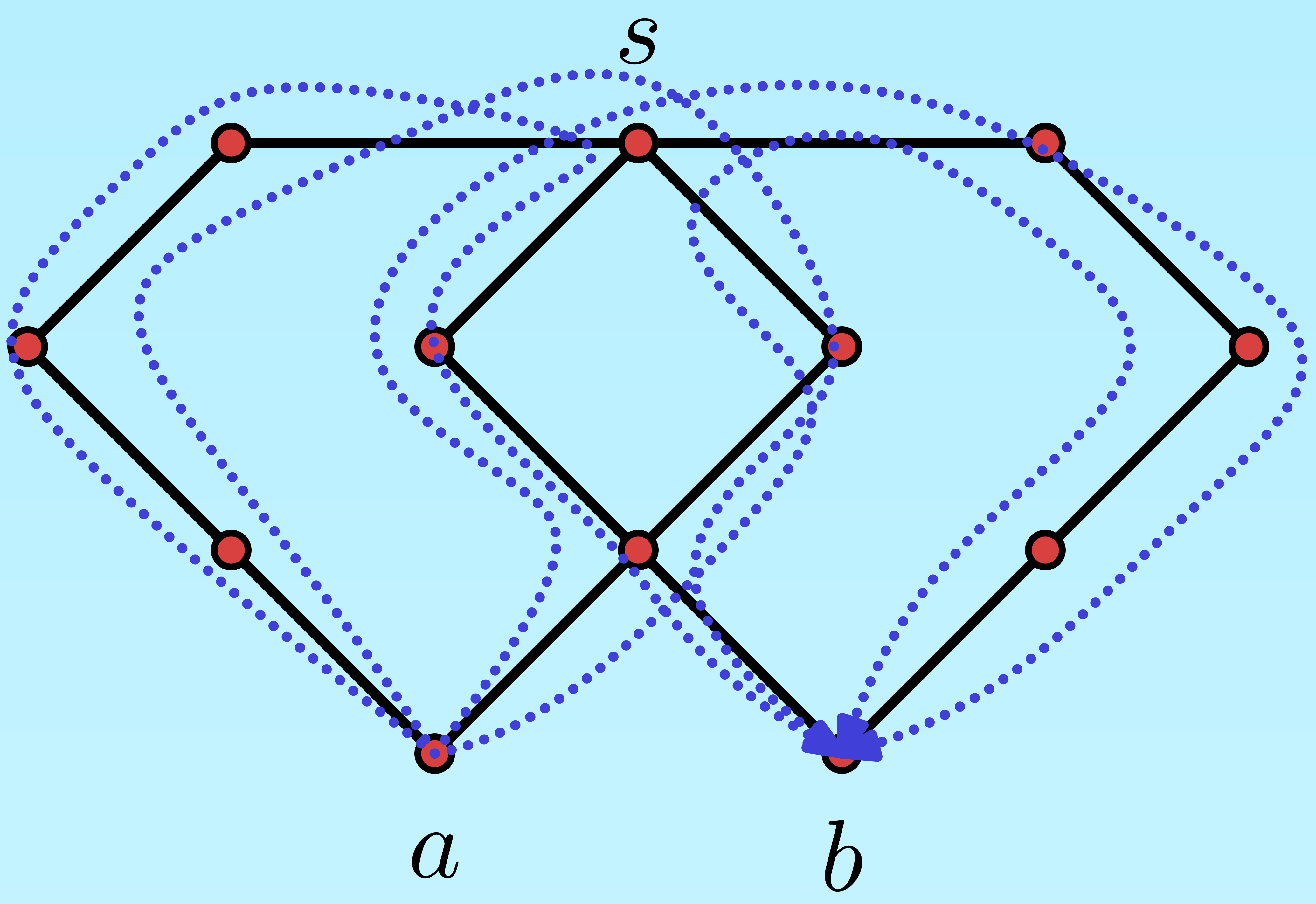


$\sigma(W(b, 7, \text{yes})) \neq 0$

Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Maintain $\sigma(W(v, \ell, \text{yes/no}))$, the signature of W.

Then we want to find the smallest ℓ for which $\sigma(W(b, \ell, \text{yes})) \neq 0$.



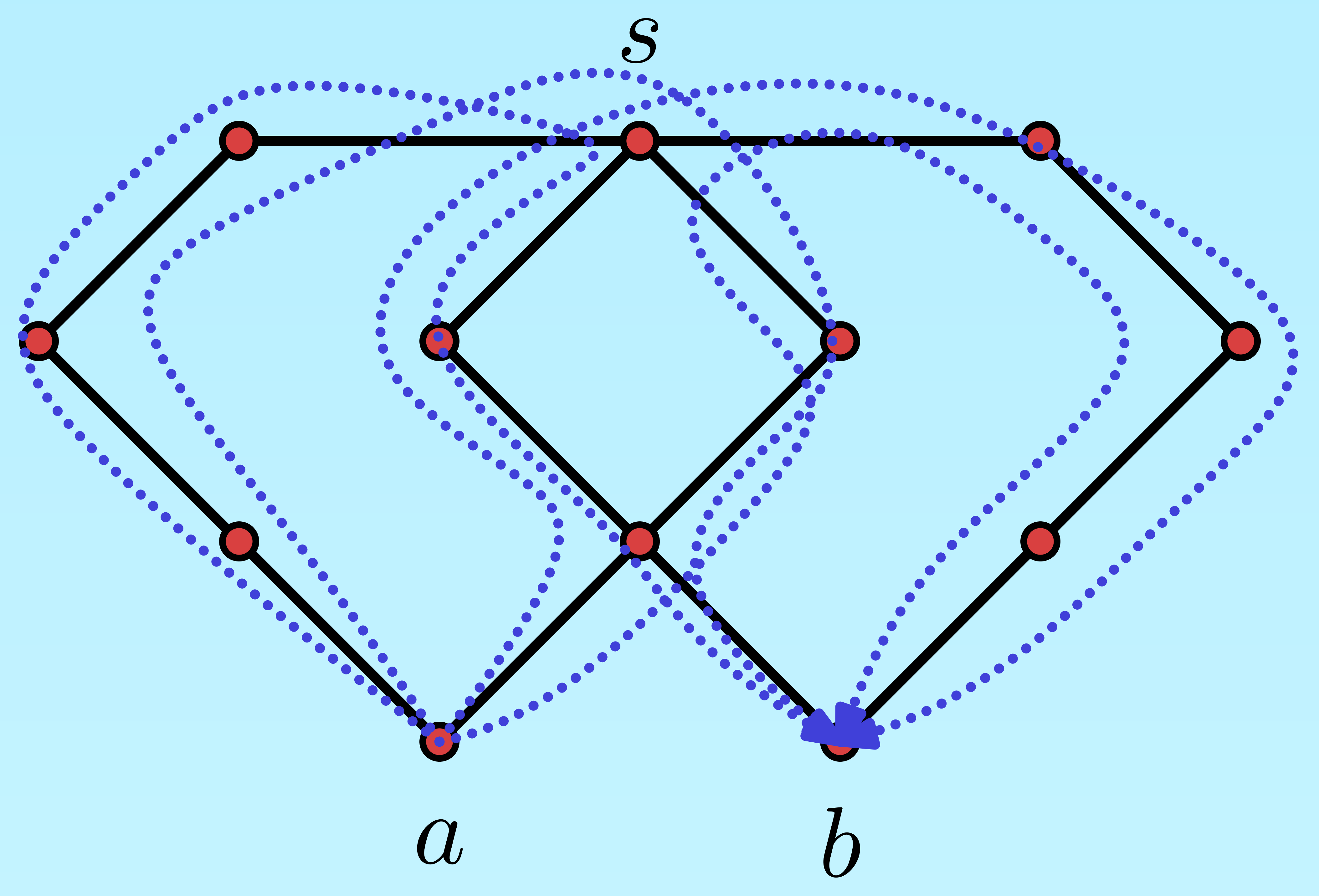
$\sigma(W(b, 7, \text{yes})) \neq 0$

Problem. Given a graph, find the shortest simple path from a to b that passes through s.

Maintain $\sigma(W(v, \ell, \text{yes/no}))$, the signature of W.

Then we want to find the smallest ℓ for which $\sigma(W(b, \ell, \text{yes})) \neq 0$.

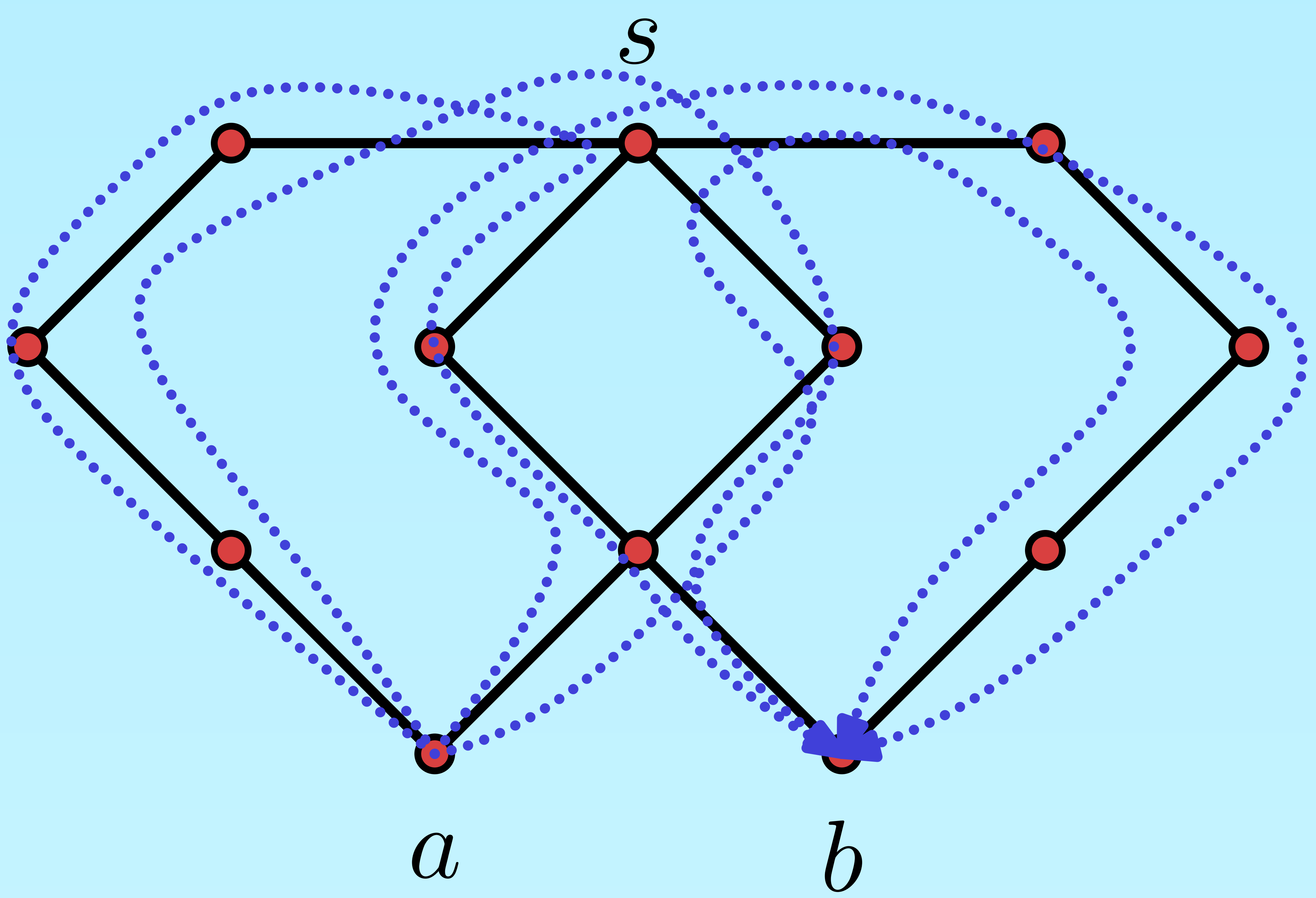
We can compute the values of $\sigma(W(\cdot, \cdot, \cdot))$ using dynamic programming!



$$\sigma(W(b, 7, \text{yes})) \neq 0$$

Problem. Given a graph, find the shortest simple path from a to b that passes through s.

This algorithm is not constructive, but can be used as a subroutine in a constructive algorithm.

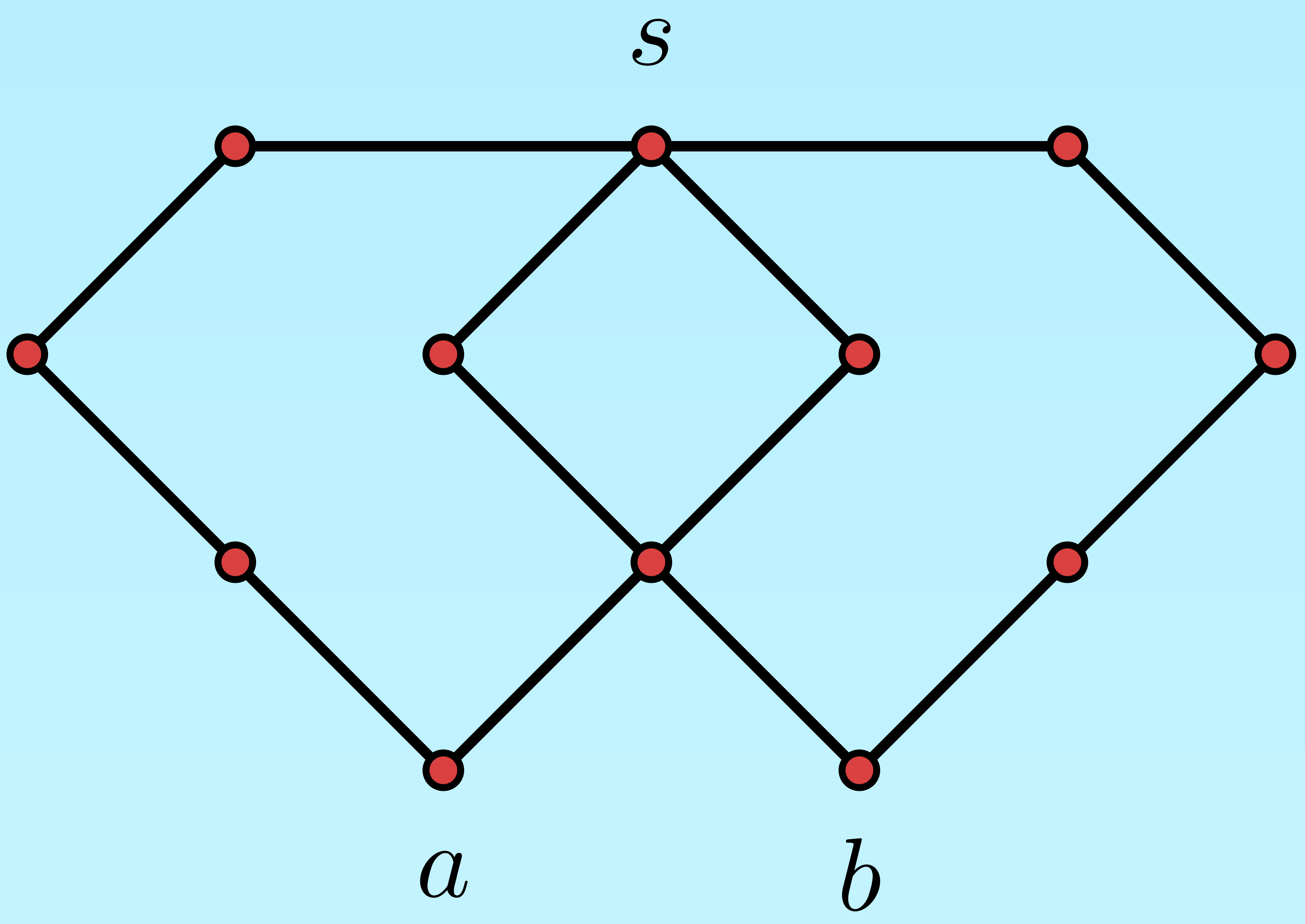


$$\sigma(W(b,7,\textbf{yes})) \neq 0$$

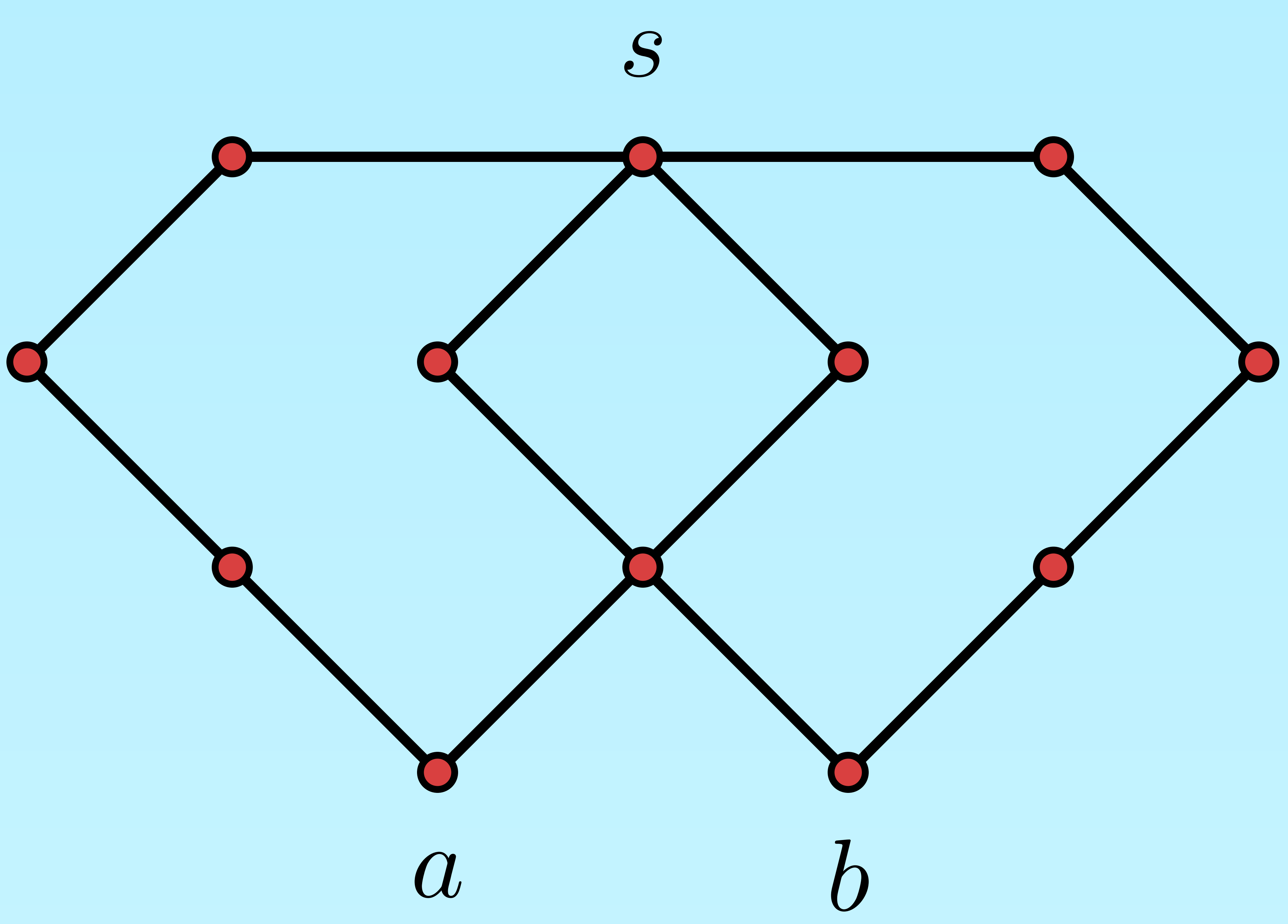
Problem. Given a graph, find the shortest simple path from a to b that passes through s.

This algorithm is not constructive, but can be used as a subroutine in a constructive algorithm.

Theorem. This problem can be solved in polynomial time with high probability.

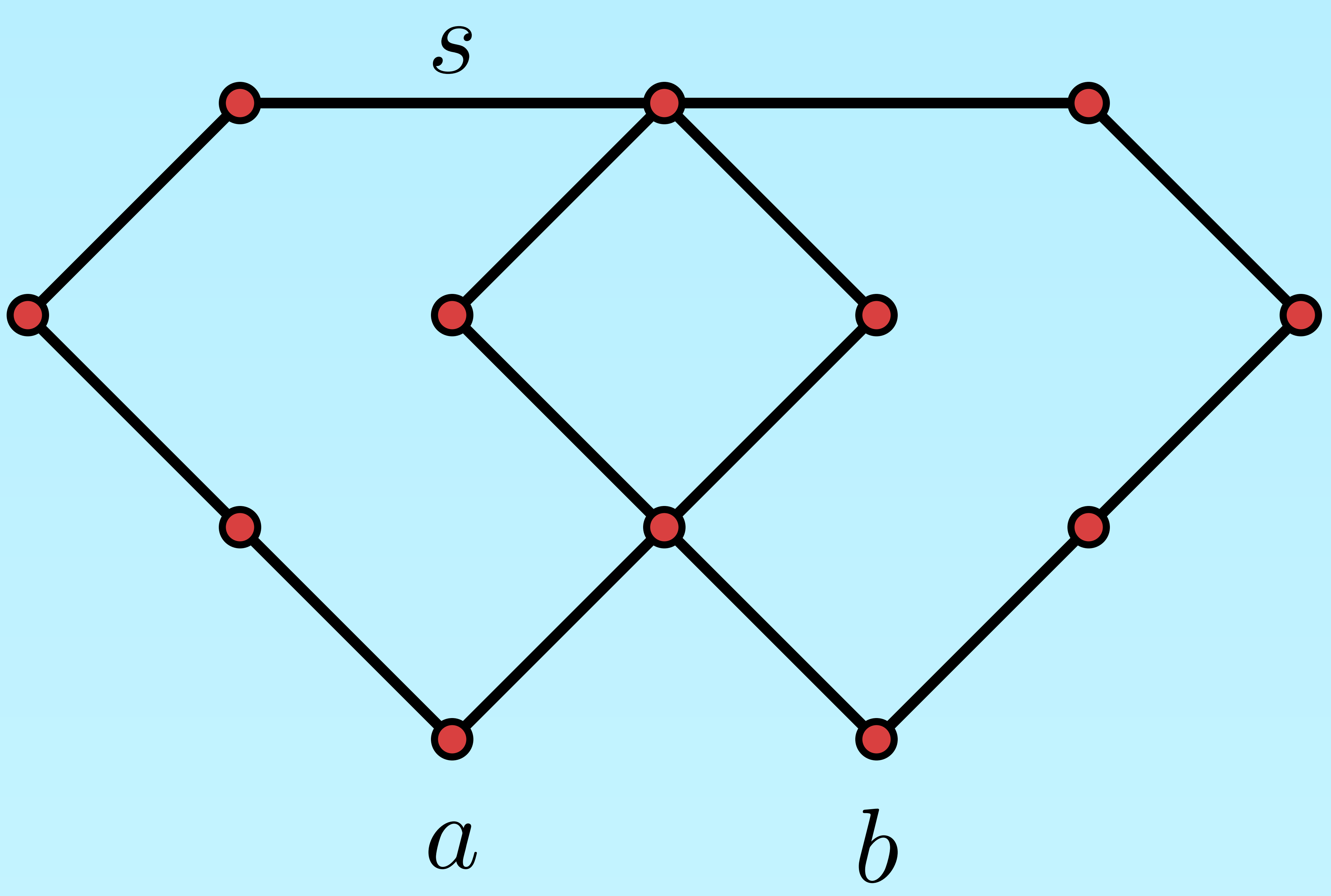


What’s different for us?



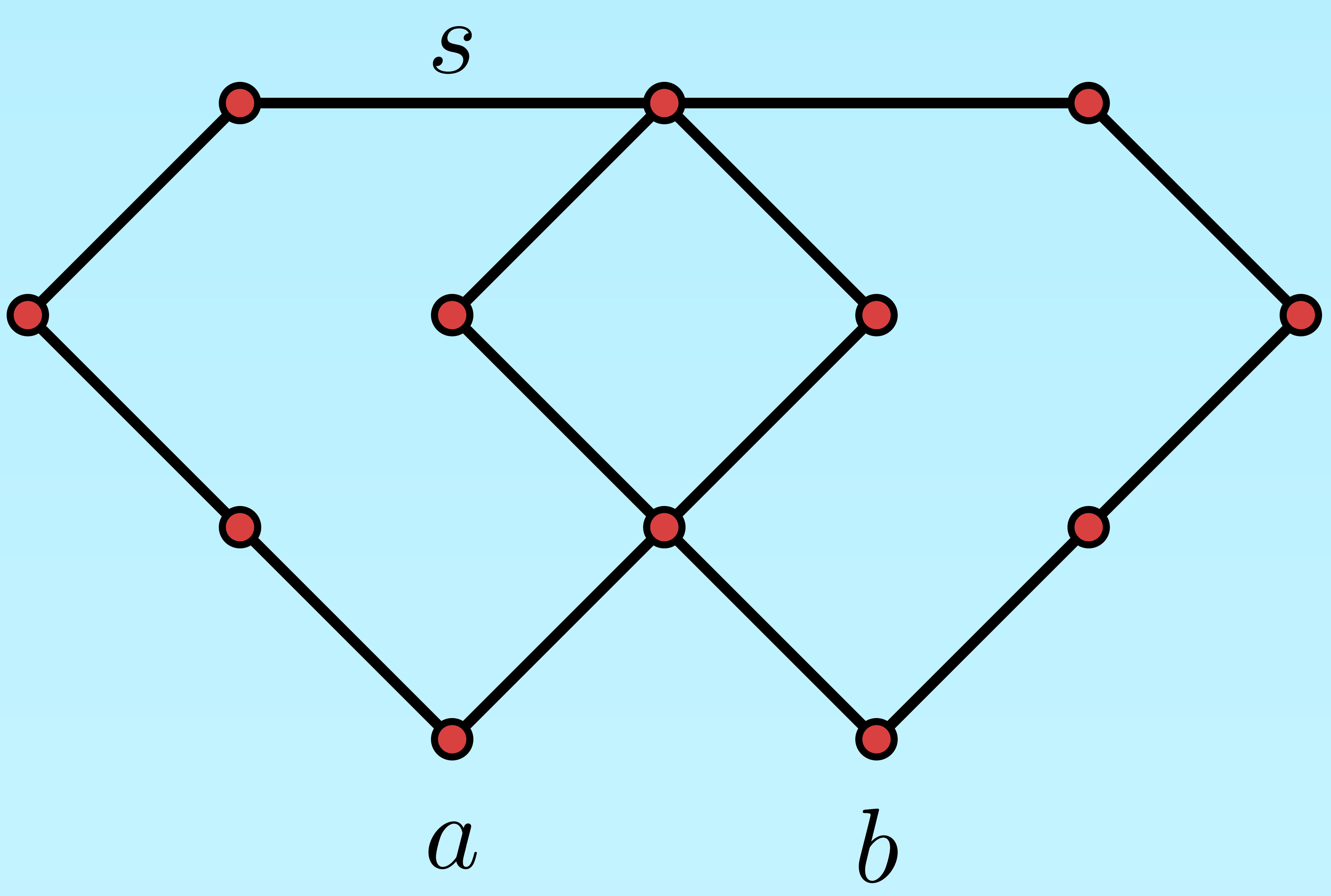
What's different for us?

We want to visit edges, not vertices.



What's different for us?

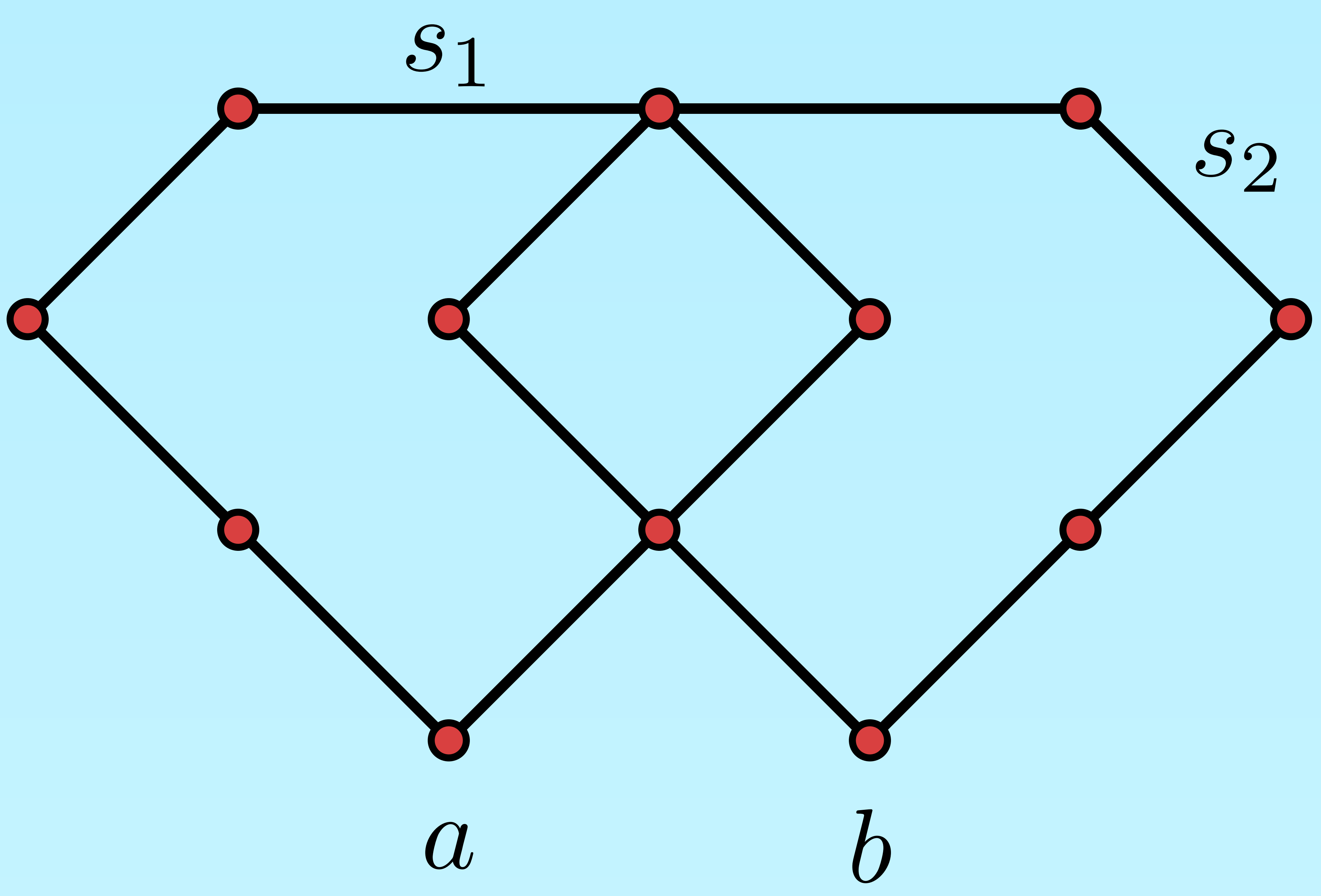
We want to visit edges, not vertices.



What's different for us?

We want to visit edges, not vertices.

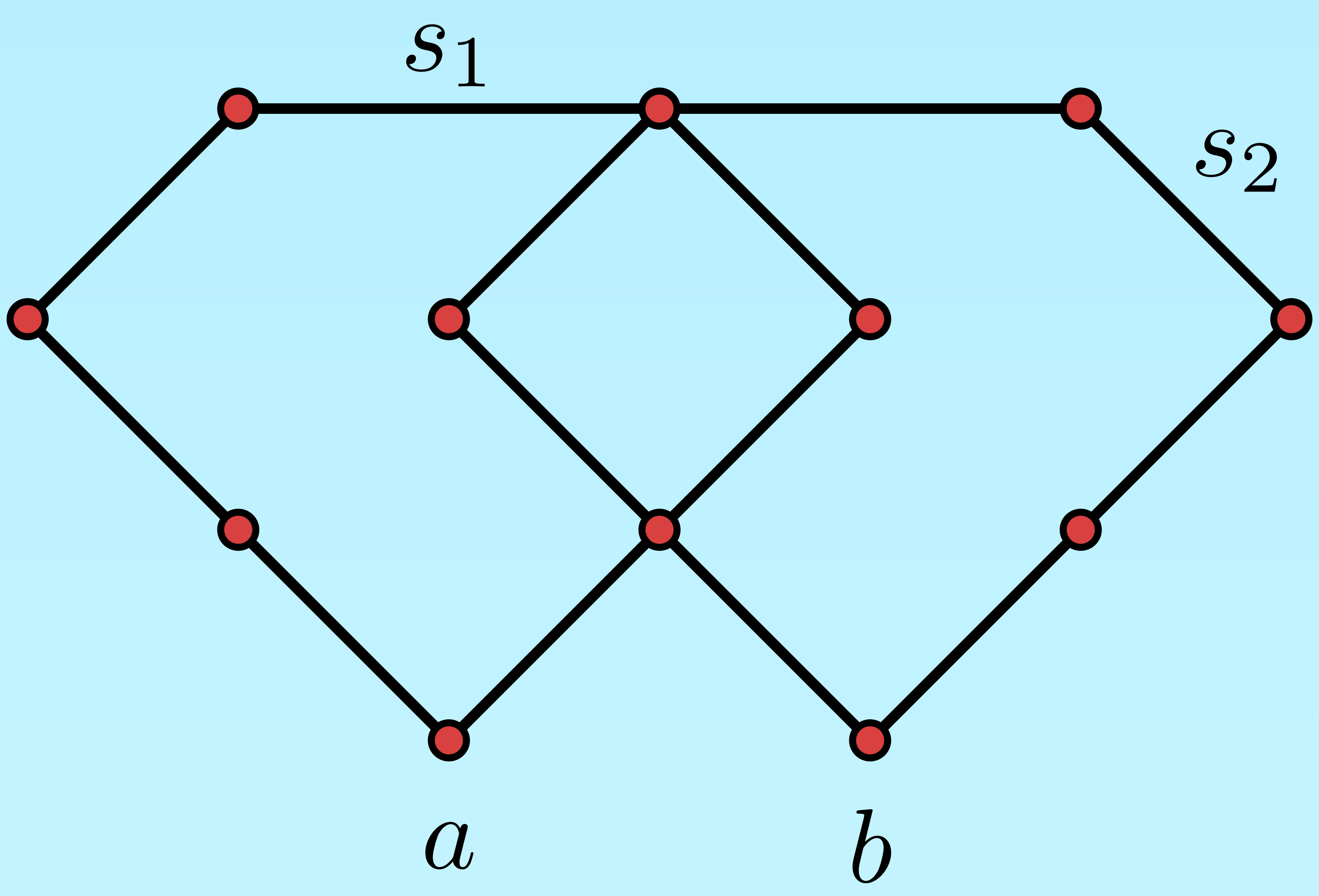
In fact, we want to visit multiple edges.



What's different for us?

We want to visit edges, not vertices.

In fact, we want to visit multiple edges.

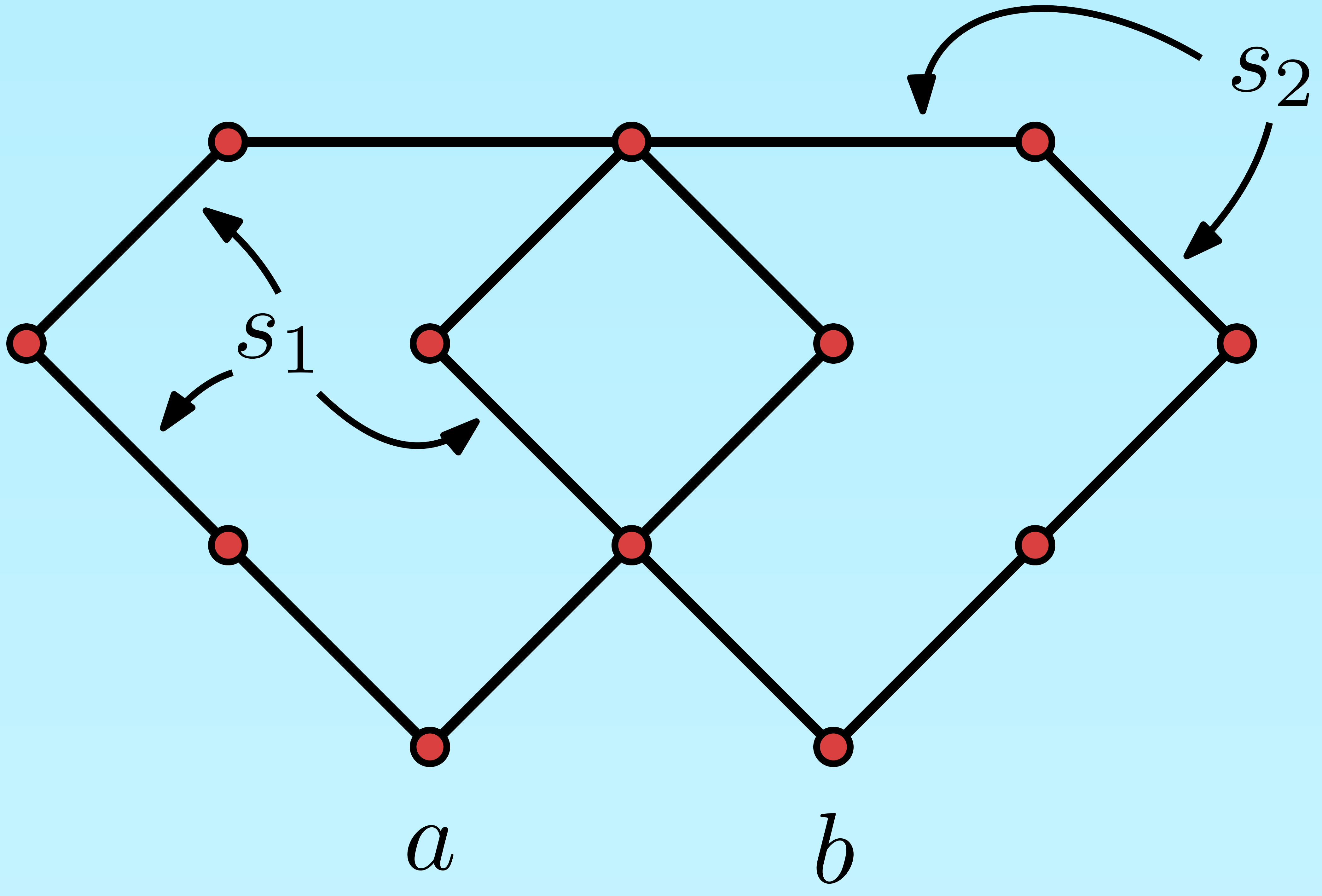


What's different for us?

We want to visit edges, not vertices.

In fact, we want to visit multiple edges.

Actually, we want to visit exactly one edge out of multiple sets of edges.

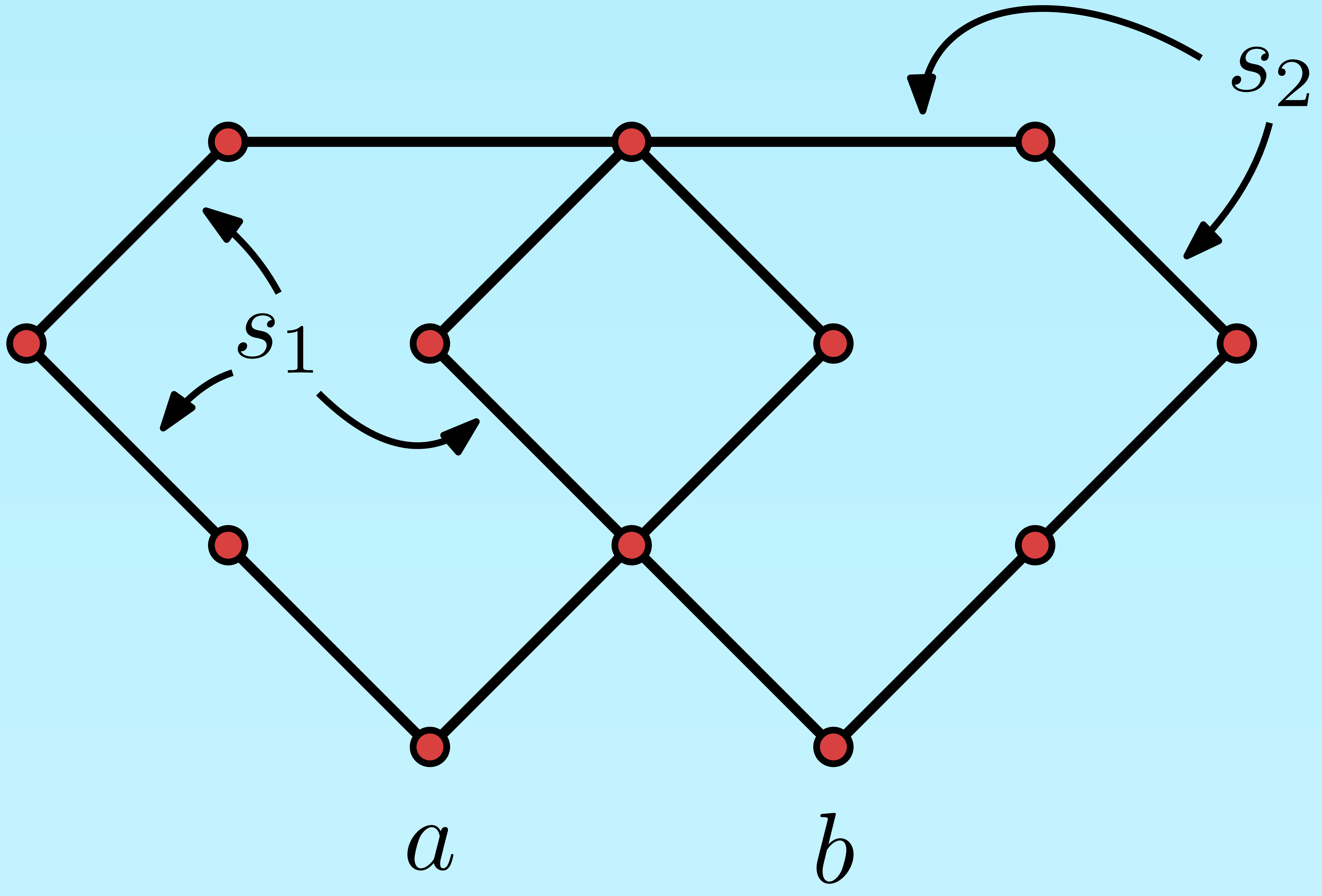


What's different for us?

We want to visit edges, not vertices.

In fact, we want to visit multiple edges.

Actually, we want to visit exactly one edge out of multiple sets of edges.



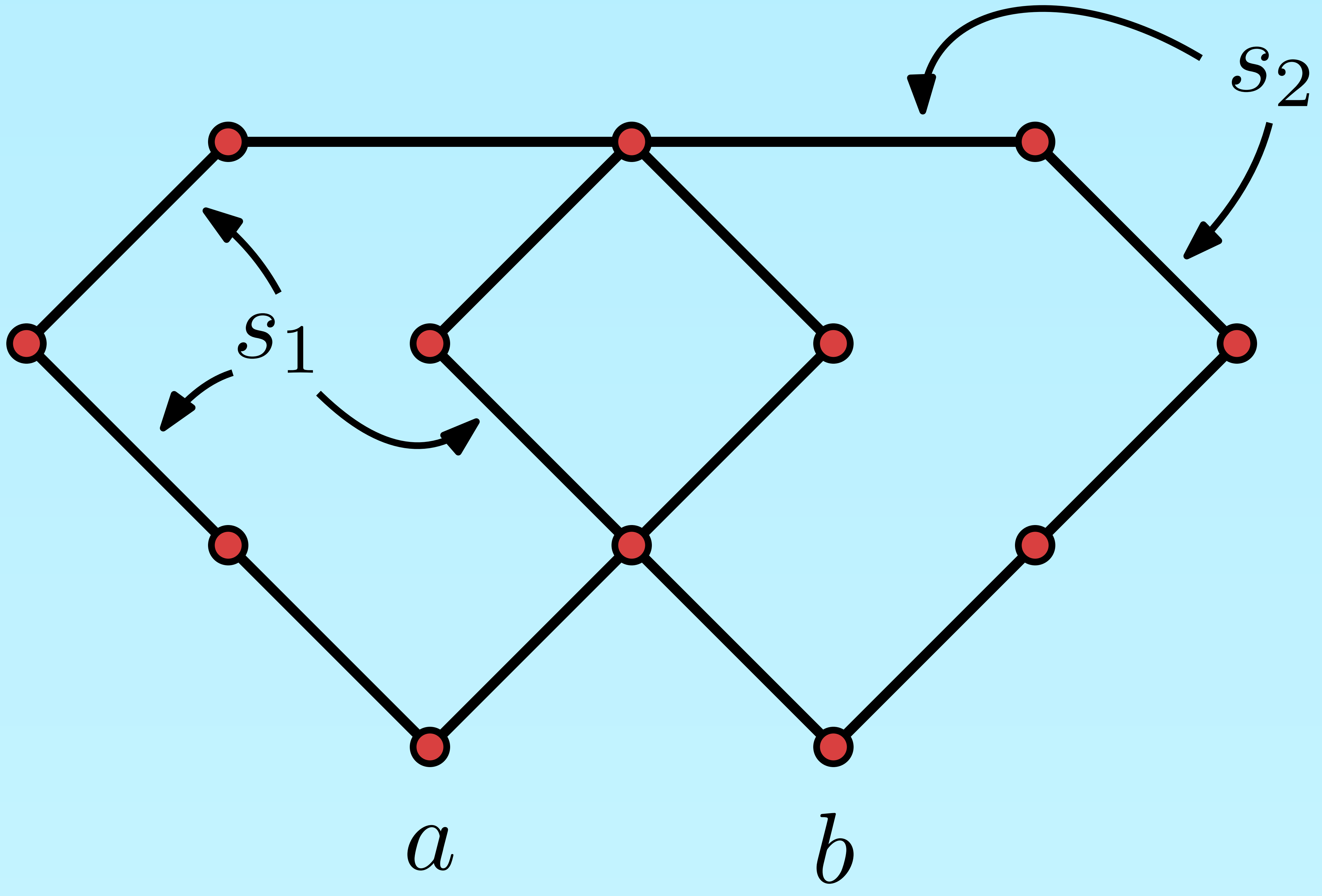
What's different for us?

We want to visit edges, not vertices.

In fact, we want to visit multiple edges.

Actually, we want to visit exactly one edge out of multiple sets of edges.

We don't have a specified start and end point...



What's different for us?

We want to visit edges, not vertices.

In fact, we want to visit multiple edges.

Actually, we want to visit exactly one edge out of multiple sets of edges.

We don't have a specified start and end point...
...and we completely ignored the bounding box.

Thank You!

Eliminating Popular Faces
(in curve arrangements)

Maarten Löffler
Utrecht University / Tulane University

Joint work with

Phoebe de Nooijer
Alexandra Weinberger
Soeren Terziadis
Zuzana Masárová
Tamara Mchedlidze
Günter Rote