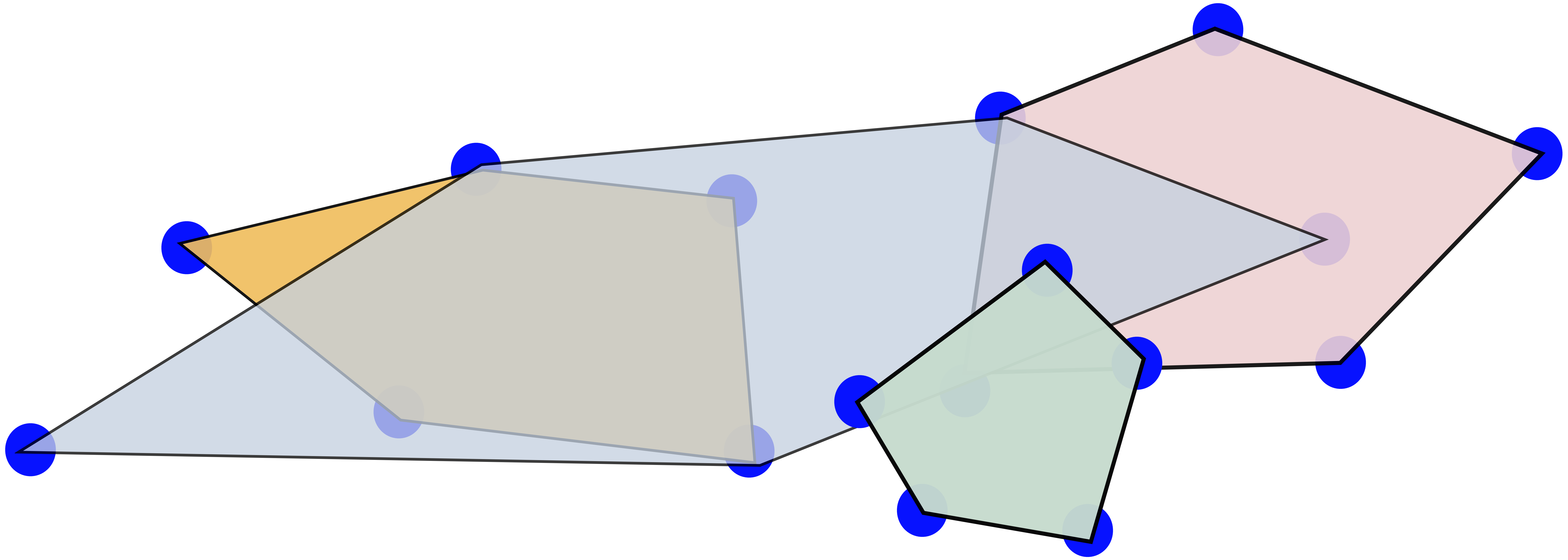
Automated
Mathematical Discovery and Verification

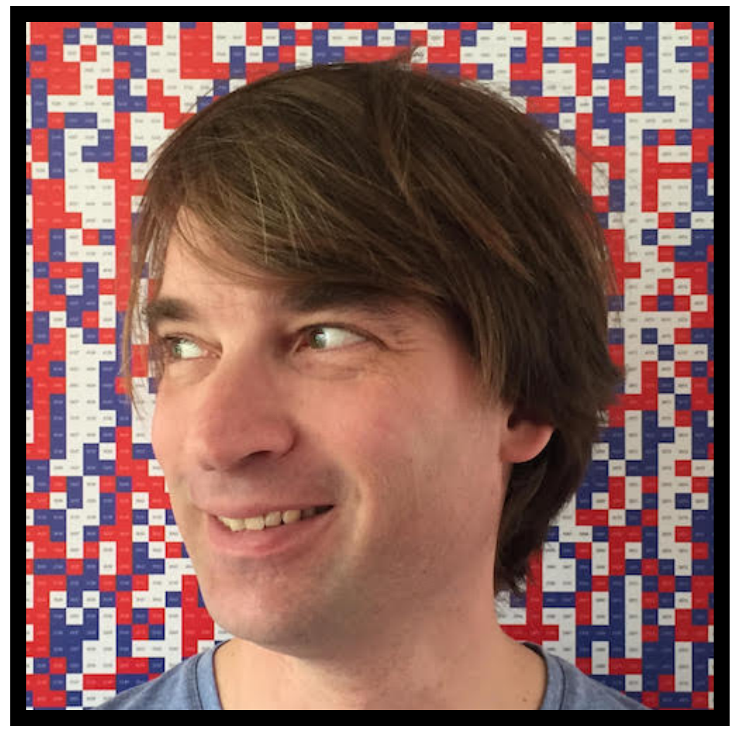


Minimizing Pentagons in the Plane

Bernardo Subercaseaux, John Mackey, Marijn Heule, and Ruben Martins

Carnegie Mellon University

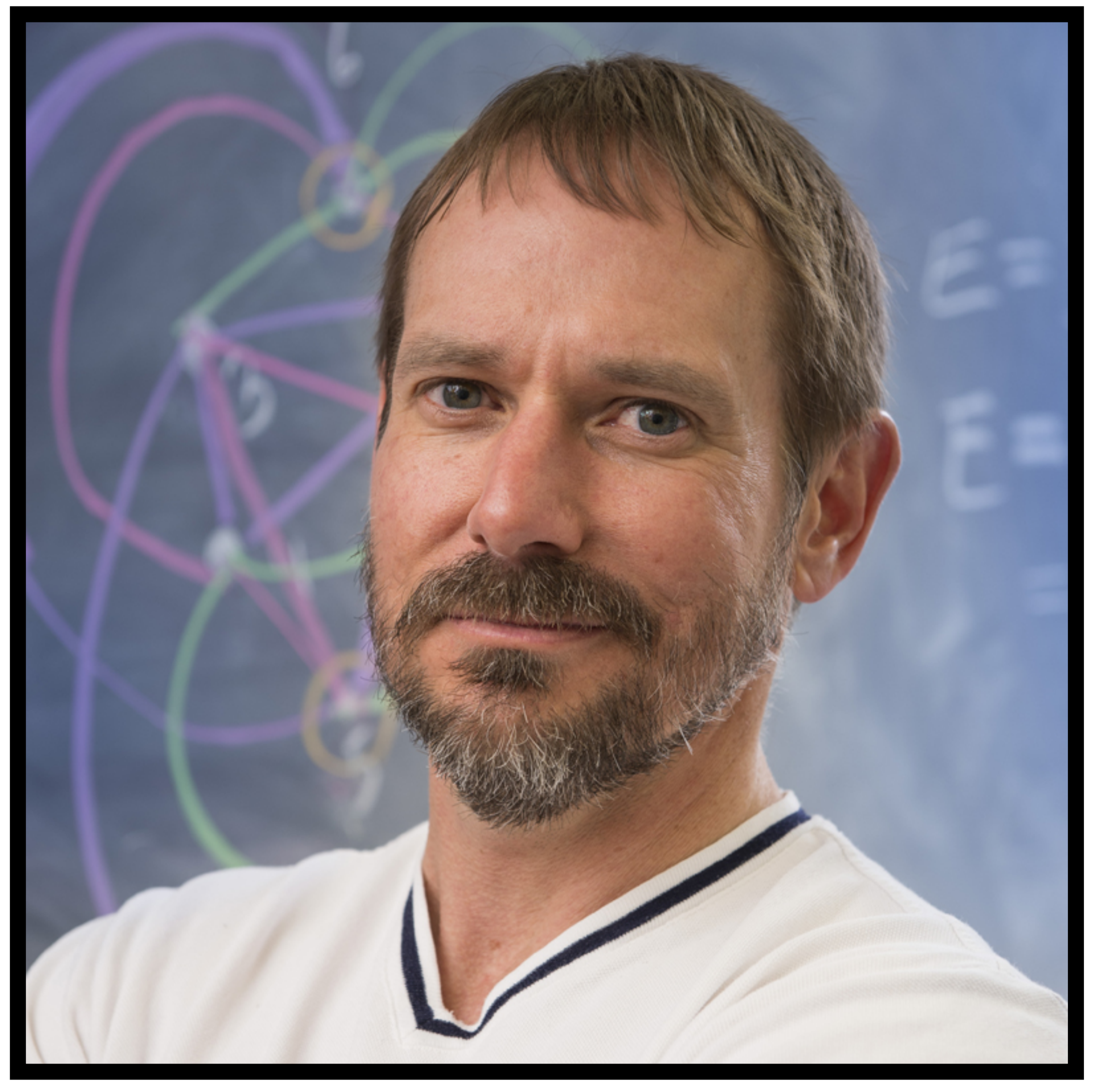
Automated Reasoning (CS) meets Math



Marijn Heule
CMU CS

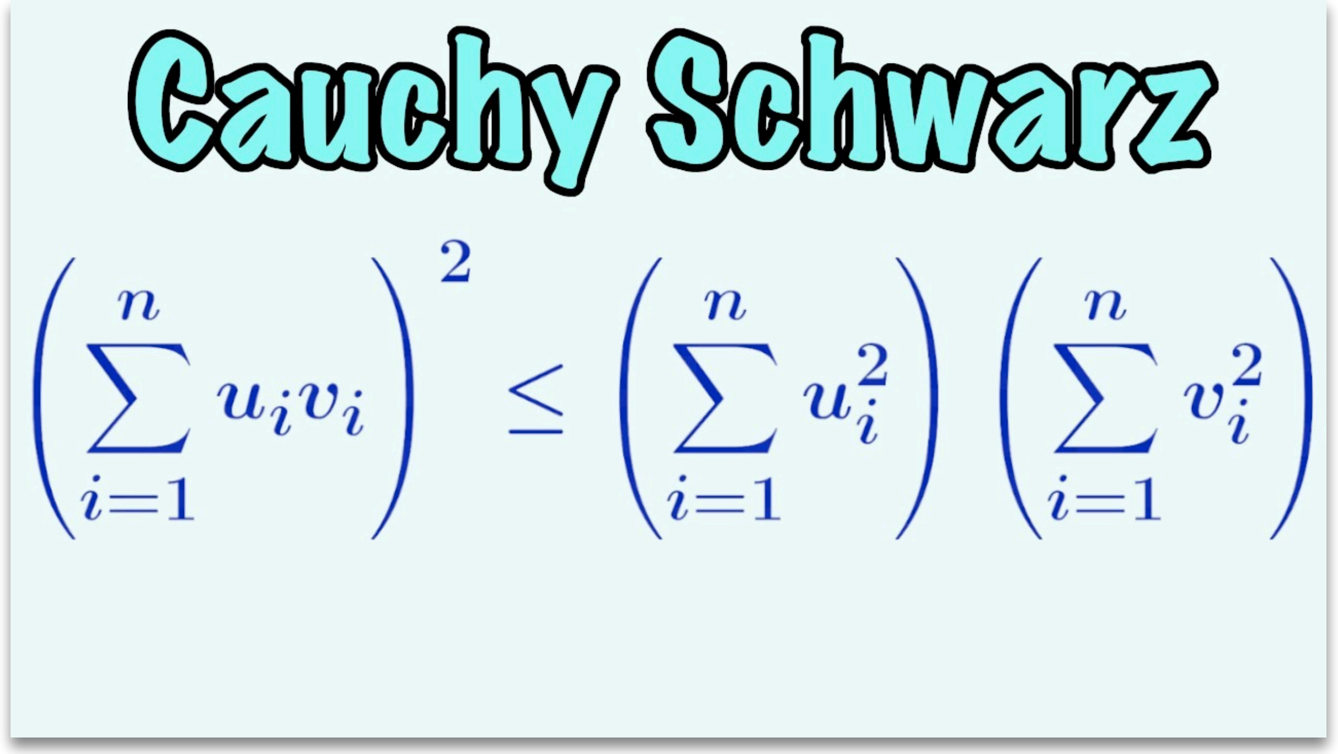
Ruben Martins
CMU CS

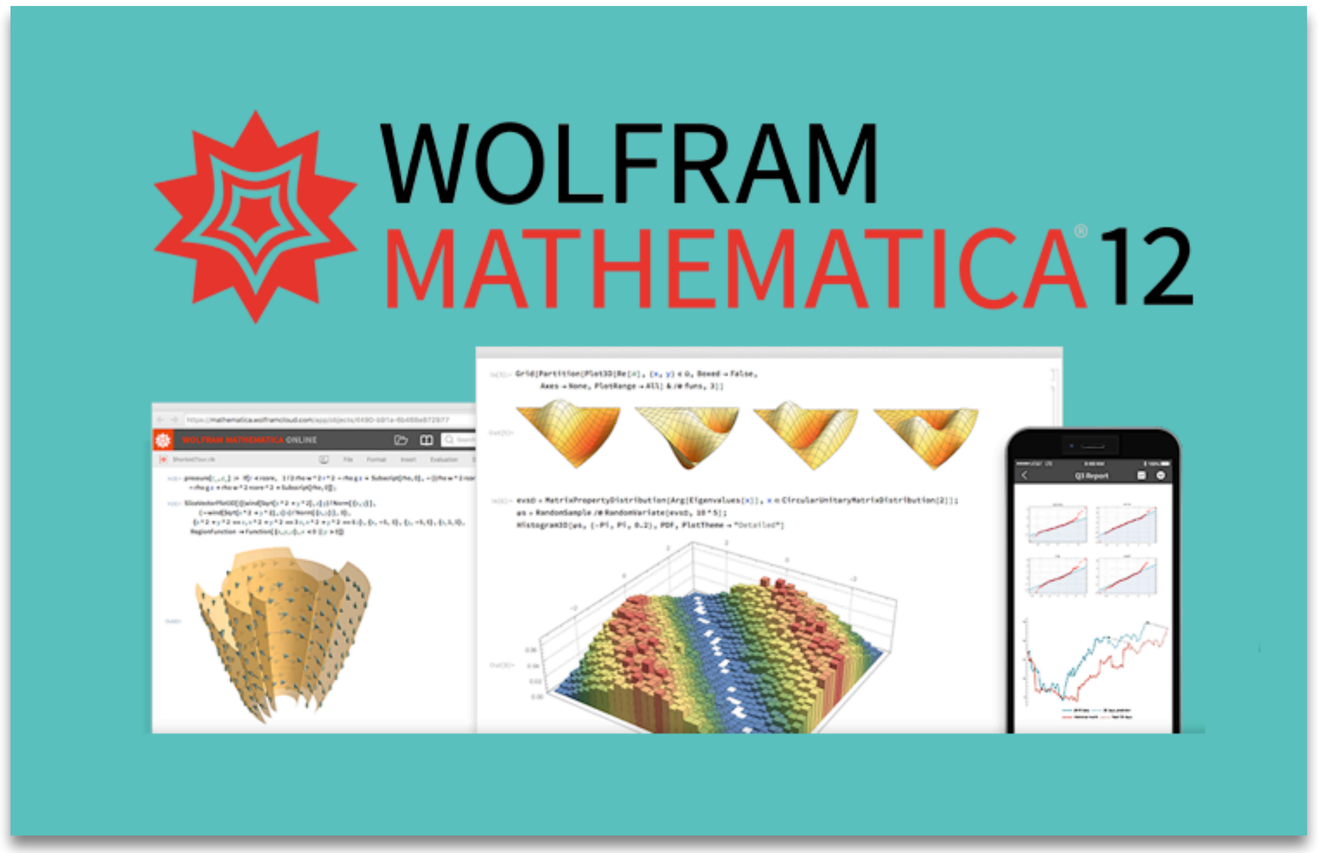
Me
CMU CS

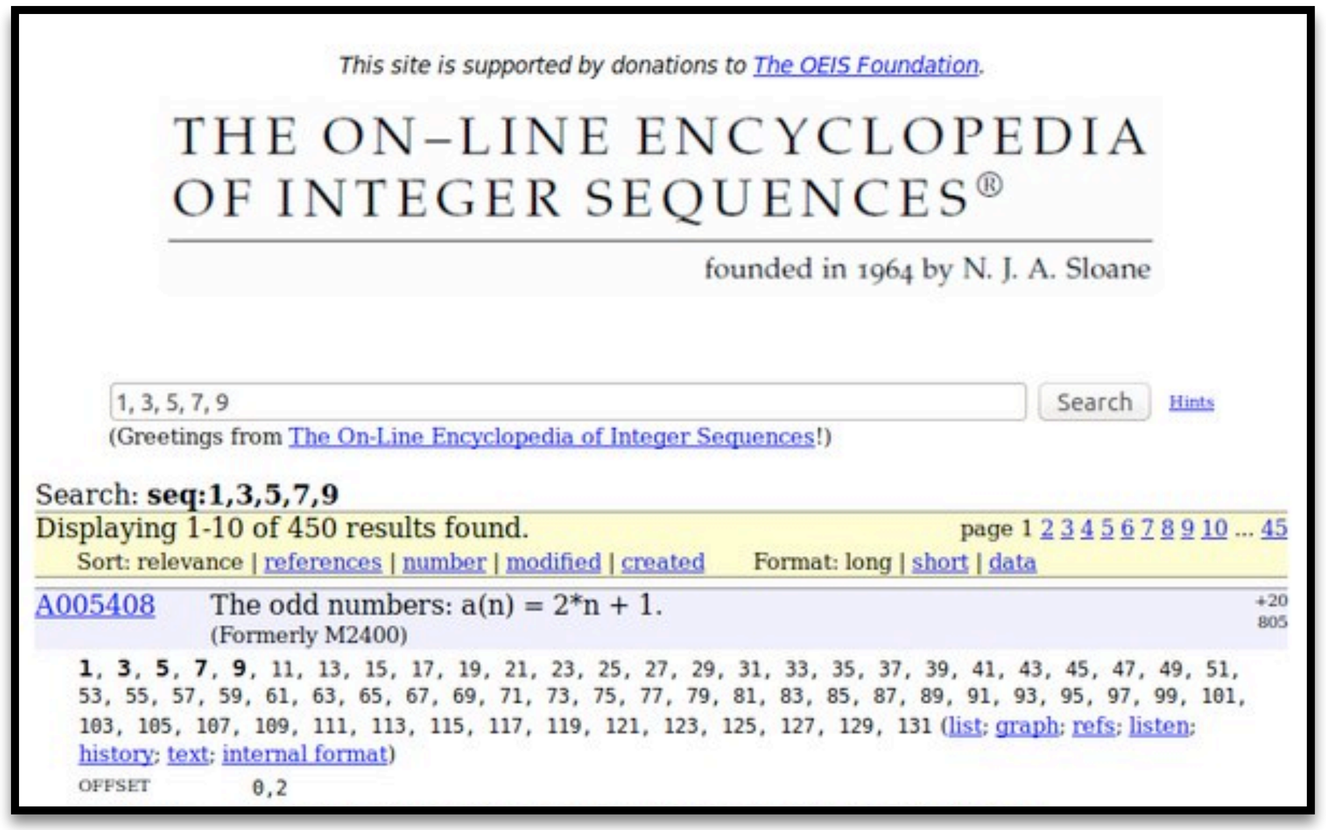


John Mackey
CMU Math

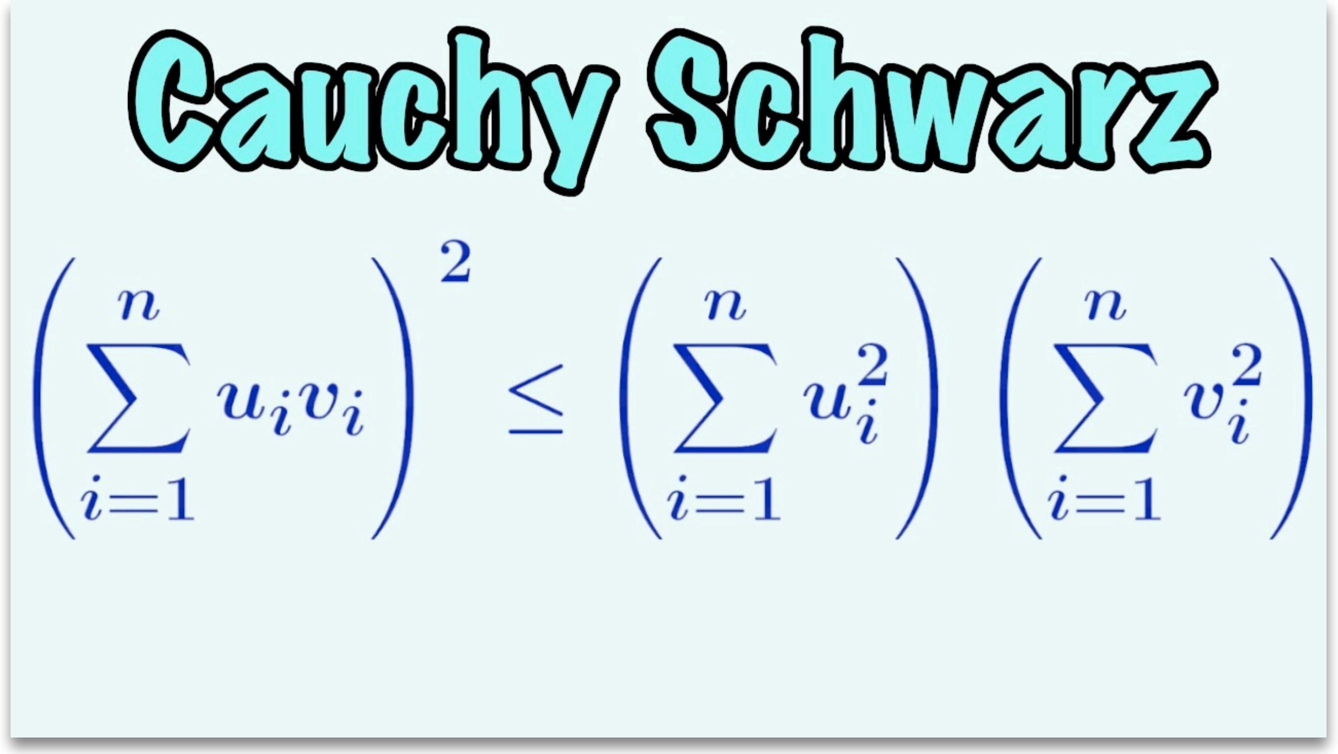
A mathematician's toolbox

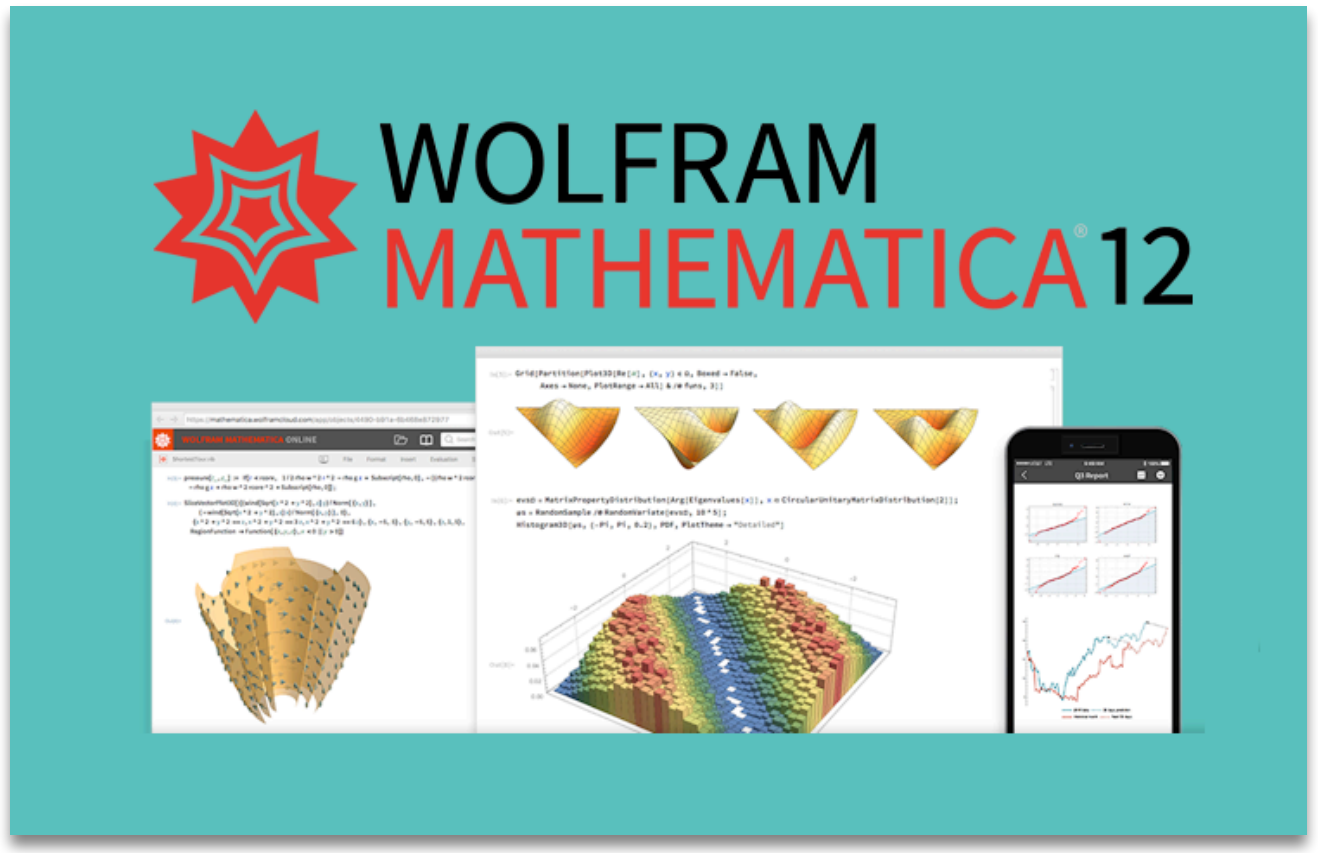


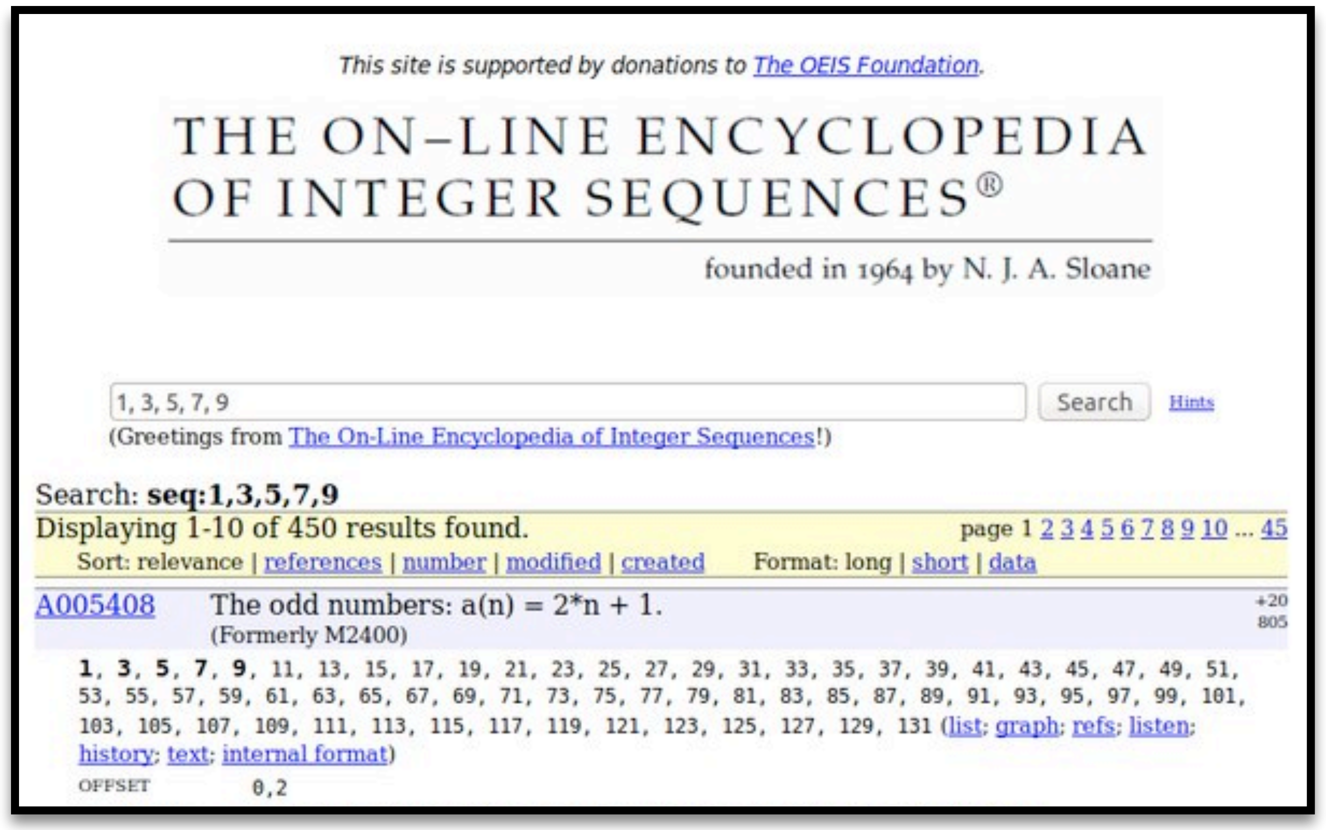




A mathematician’s toolbox






This site is supported by donations to The OEIS Foundation.

THE ON-LINE ENCYCLOPEDIA OF INTEGER SEQUENCES®

founded in 1964 by N. J. A. Sloane

1, 3, 5, 7, 9 Search Hints

(Greetings from The On-Line Encyclopedia of Integer Sequences!)

Search: **seq:1,3,5,7,9**

Displaying 1-10 of 450 results found. page 1 2 3 4 5 6 7 8 9 10 ... 45

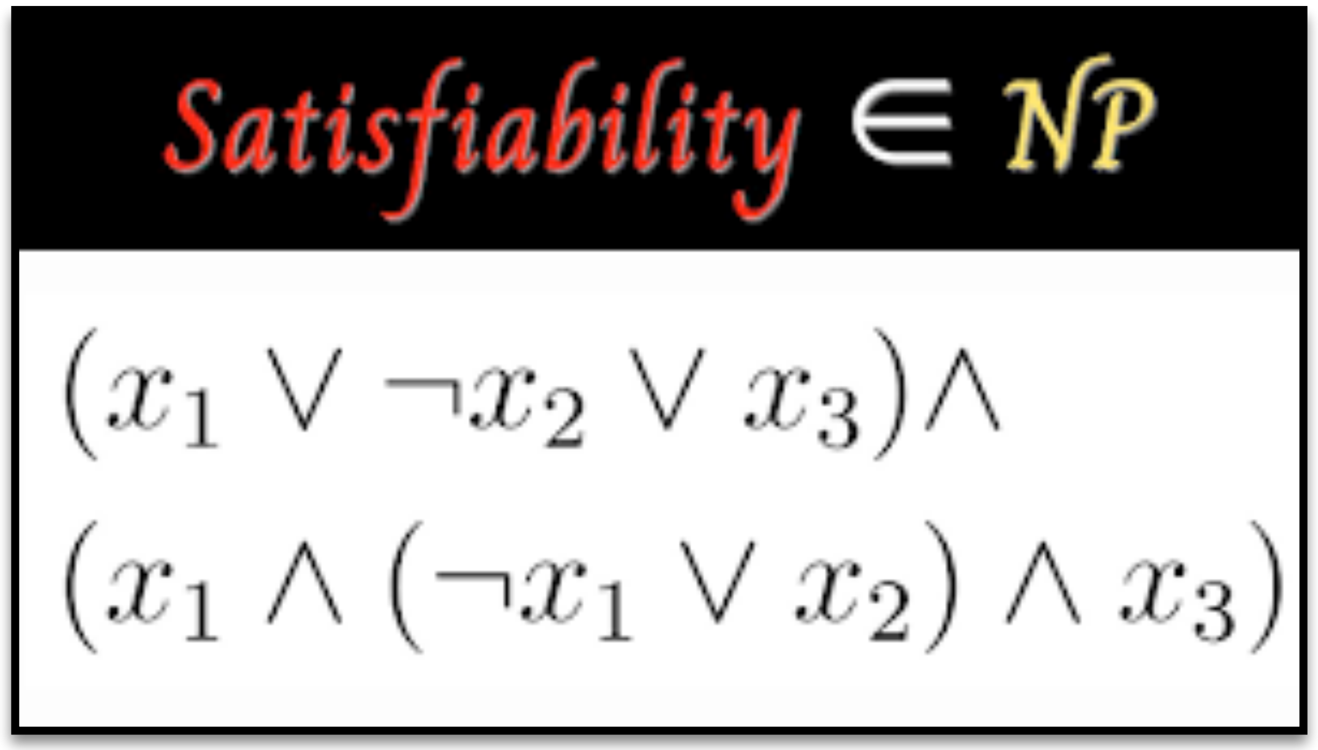
Sort: relevance | references | number | modified | created Format: long | short | data

A005408 The odd numbers: a(n) = 2*n + 1. +20 805

(Formerly M2400)

1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99, 101, 103, 105, 107, 109, 111, 113, 115, 117, 119, 121, 123, 125, 127, 129, 131 (list; graph; refs; listen; history; text; internal format)

OFFSET 0,2



SAT Solvers?

SAT Solvers

- Hyper-optimized programs for solving a single problem: ***Boolean satisfiability***

$$(x_1 \vee x_4 \vee \overline{x_5}) \wedge (x_2 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_1} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3)$$

SAT Solvers

- Hyper-optimized programs for solving a single problem: ***Boolean satisfiability***

$$(x_1 \vee x_4 \vee \overline{x_5}) \wedge (x_2 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_1} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3)$$

$$x_1 = 0, \quad x_2 = 1, \quad x_3 = 1, \quad x_4 = 1, \quad x_5 = 0$$

SAT Solvers

- Hyper-optimized programs for solving a single problem: ***Boolean satisfiability***

$$(x_1 \lor x_4 \lor \overline{x_5}) \land (x_2 \lor \overline{x_3} \lor x_4) \land (\overline{x_1} \lor \overline{x_4}) \land (\overline{x_1} \lor \overline{x_2} \lor x_3)$$

$$x_1 = 0, \quad x_2 = 1, \quad x_3 = 1, \quad x_4 = 1, \quad x_5 = 0$$

SAT Solvers

- Hyper-optimized programs for solving a single problem: ***Boolean satisfiability***

$$(x_1 \lor x_4 \lor \overline{x_5}) \land (x_2 \lor \overline{x_3} \lor x_4) \land (\overline{x_1} \lor \overline{x_4}) \land (\overline{x_1} \lor \overline{x_2} \lor x_3)$$

$$x_1 = 0, \quad x_2 = 1, \quad x_3 = 1, \quad x_4 = 1, \quad x_5 = 0$$

- **Because of NP-completeness, it can encode a variety of combinatorial problems!**

```python
def encode(clues):
    """
    Encode a sudoku puzzle as a SAT problem.

    Parameters
    ----------
    clues : list of lists of ints
        9x9 int matrix, non-zero values represent clues

    Returns
    -------
    model : eznf.Model
        A SAT model representing the sudoku puzzle.
    """
    Z = modeler.Modeler()  # modeler object from my library, always starts like this.

    # create vars
    for i in range(9):
        for j in range(9):
            for n in range(1, 10):
                Z.add_var(f"x_{i, j, n}", description=f"Cell ({i}, {j}) gets number {n}")
```

```python
# exactly one number per cell
for i in range(9):
    for j in range(9):
        Z.exactly_one([f"x_{i, j, n}" for n in range(1, 10)])

# respect clues
for i in range(9):
    for j in range(9):
        if clues[i][j] != 0:
            Z.constraint(f"x_{i, j, clues[i][j]}")

# exactly-one constraints
for n in range(1, 10):
    # rows
    for i in range(9):
        Z.exactly_one([f"x_{i, j, n}" for j in range(9)])

    # cols
    for j in range(9):
        Z.exactly_one([f"x_{i, j, n}" for i in range(9)])

    # sub_grids
    sub_grids = [[[] for sj in range(3)] for si in range(3)]
    for i in range(9):
        for j in range(9):
            sub_grids[i//3][j//3].append((i, j))
    for si in range(3):
        for sj in range(3):
            Z.exactly_one([f"x_{*cell, n}" for cell in sub_grids[si][sj]])
```

SAT Solvers in Math

Some success stories:

- (2014) Boolean Erdős Discrepancy Problem
- (2016) Boolean Pythagorean Triples
- (2018) Schur Number 5
- (2019) Keller's Conjecture
- (2023) Packing Chromatic Number of the Grid
- (2024) An Empty Hexagon in every 30 points

SAT Solvers in Math

Some success stories:

- (2014) Boolean Erdős Discrepancy Problem
- ythagorean Triples
- mber 5
- onjecture
- hromatic Number of the Grid
- (2024) An Empty Hexagon in every 30 points

These all follow a common pattern:

Using SAT solvers to tackle a hard combinatorial problem that brute force computation (i.e., backtracking) would take forever on

SAT Solvers in Math

Some success stories:

- (2014) Boolean Erdős Discrepancy Problem

These all follow a common pattern:

Using SAT solvers to tackle a hard combinatorial problem that brute force computation (i.e., backtracking) would take forever on

Today we will talk about a different use:

SAT solvers can be used at different stages of mathematical research; to build examples, get ideas and ellicit conjectures.

- (2024) An Empty Hexagon in every 30 points

Problem: Minimizing Convex Pentagons

Given an integer N, what is, $\mu_5(N)$ the minimum number of convex pentagons we can get by placing N points in the Euclidean plane without 3 on a line?

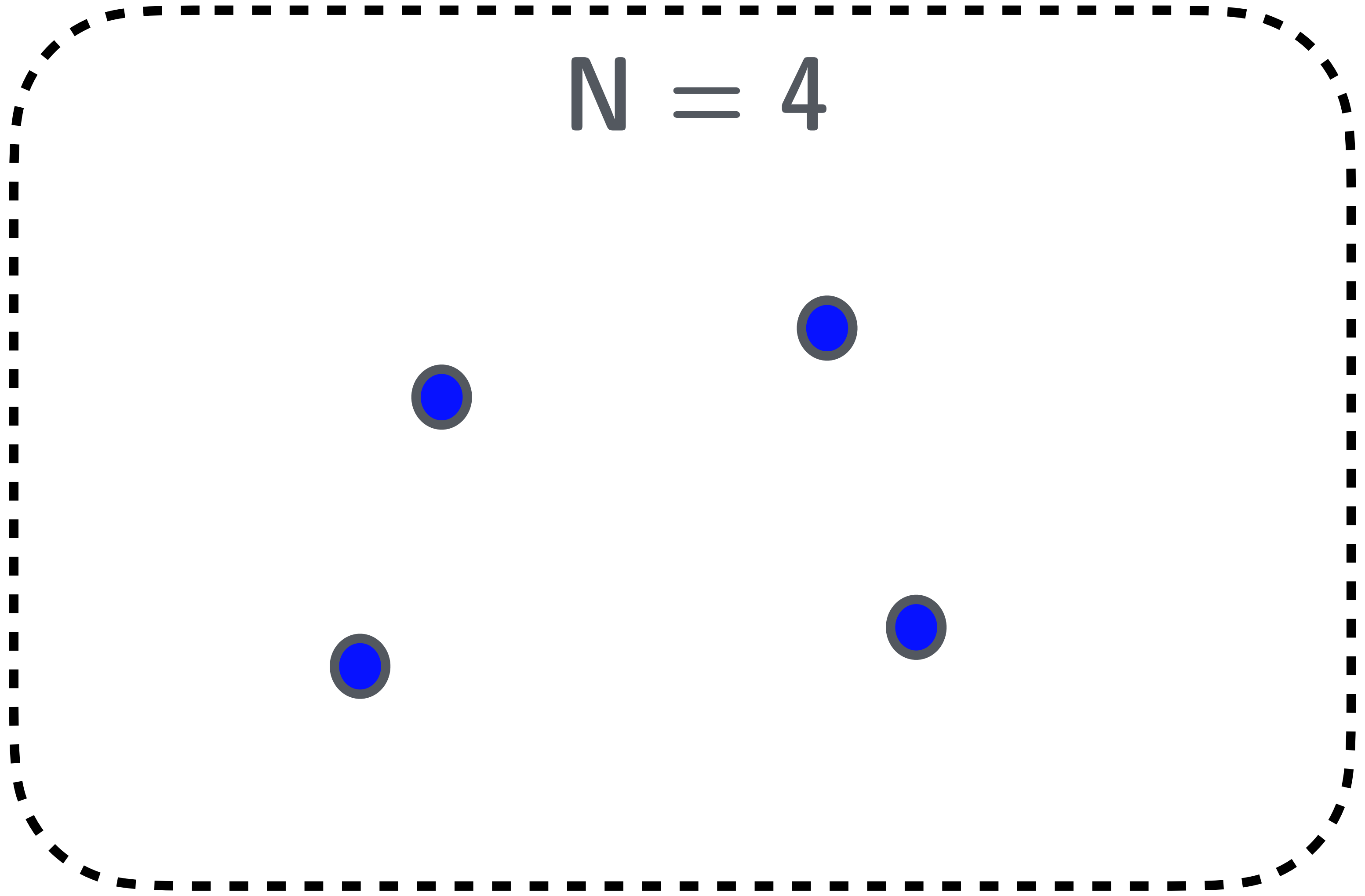
Problem: Minimizing Convex Pentagons

Given an integer N, what is, $\mu_5(N)$ the minimum number of convex pentagons we can get by placing N points in the Euclidean plane without 3 on a line?

N $=$ 4

Problem: Minimizing Convex Pentagons

Given an integer N, what is, $\mu_5(N)$ the minimum number of convex pentagons we can get by placing N points in the Euclidean plane without 3 on a line?

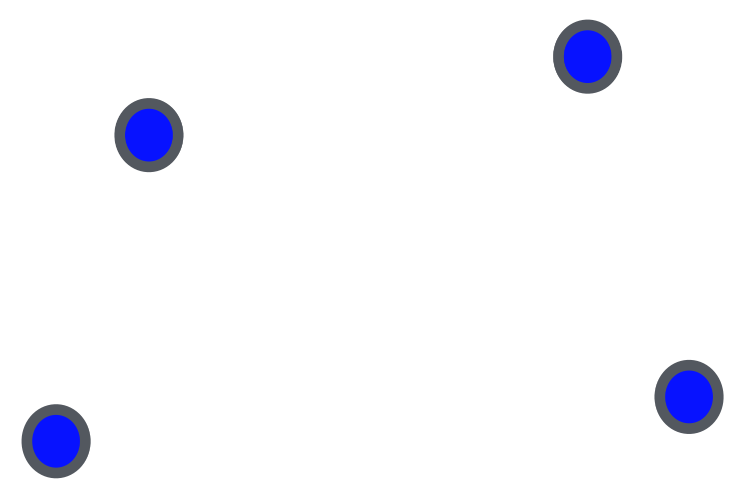


Answer = 0

Problem: Minimizing Convex Pentagons

Given an integer N, what is, $\mu_5(N)$ the minimum number of convex pentagons we can get by placing N points in the Euclidean plane without 3 on a line?

N = 4

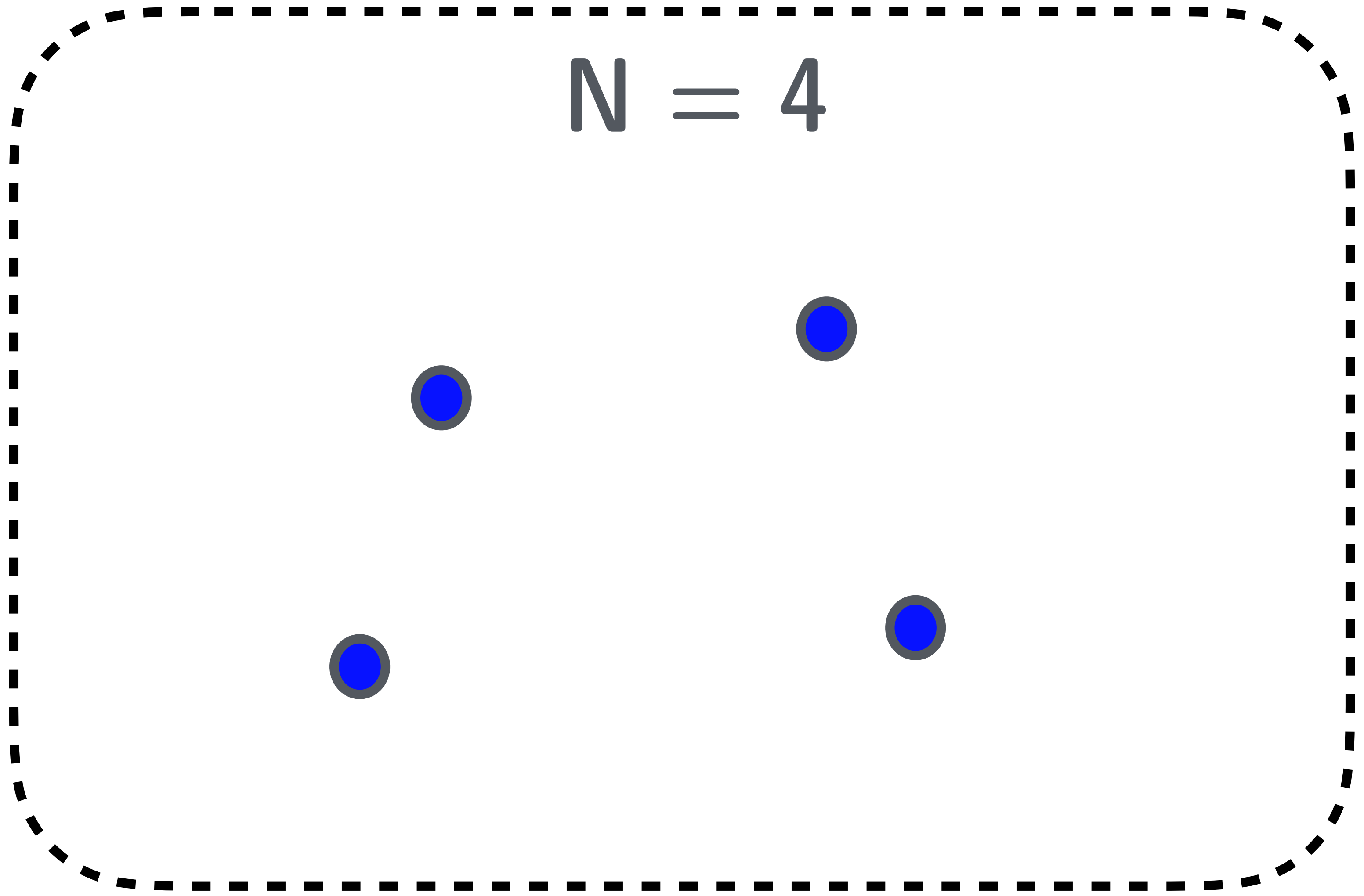


N = 8

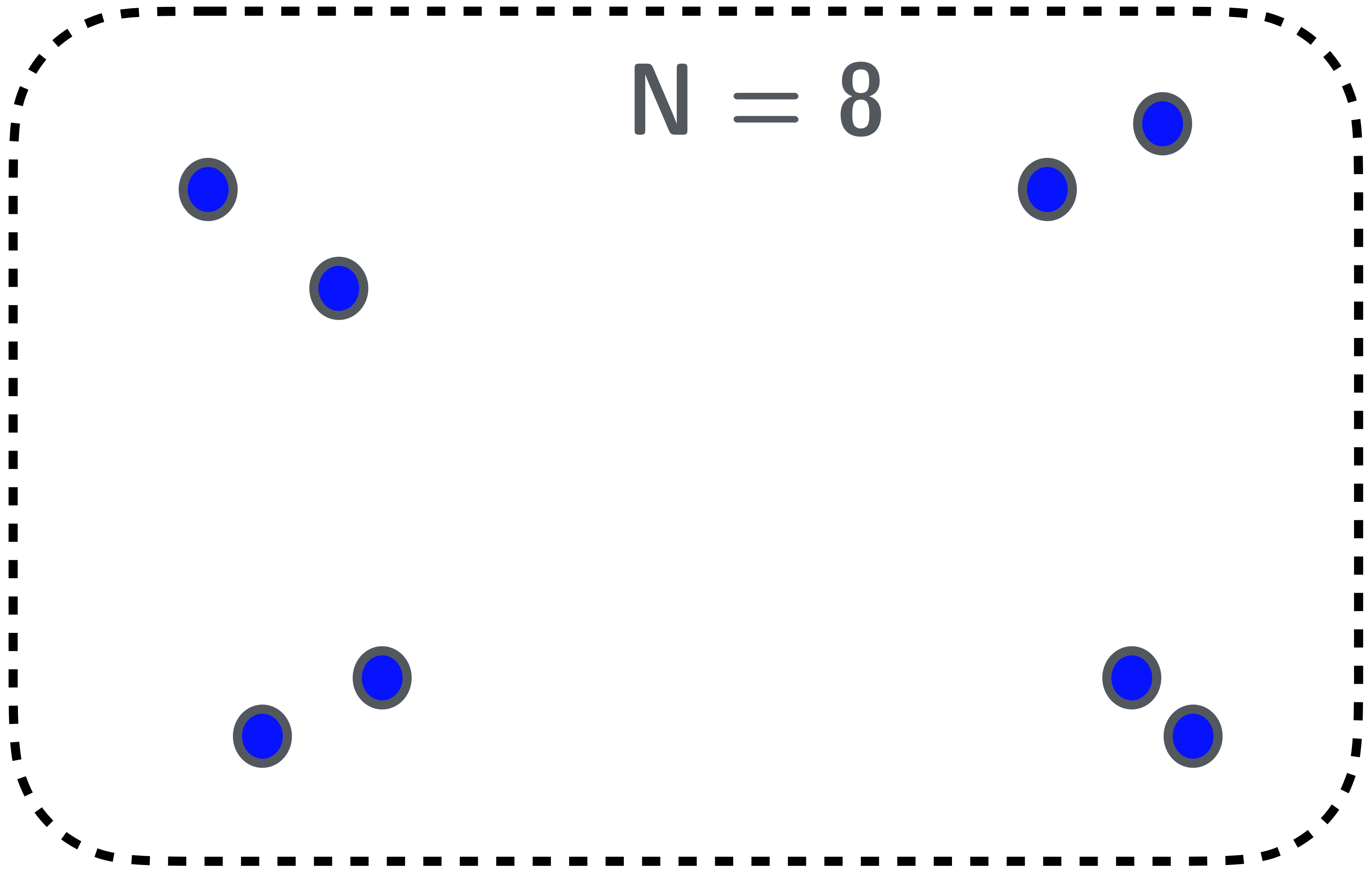
Answer = 0

Problem: Minimizing Convex Pentagons

Given an integer N, what is, $\mu_5(N)$ the minimum number of convex pentagons we can get by placing N points in the Euclidean plane without 3 on a line?



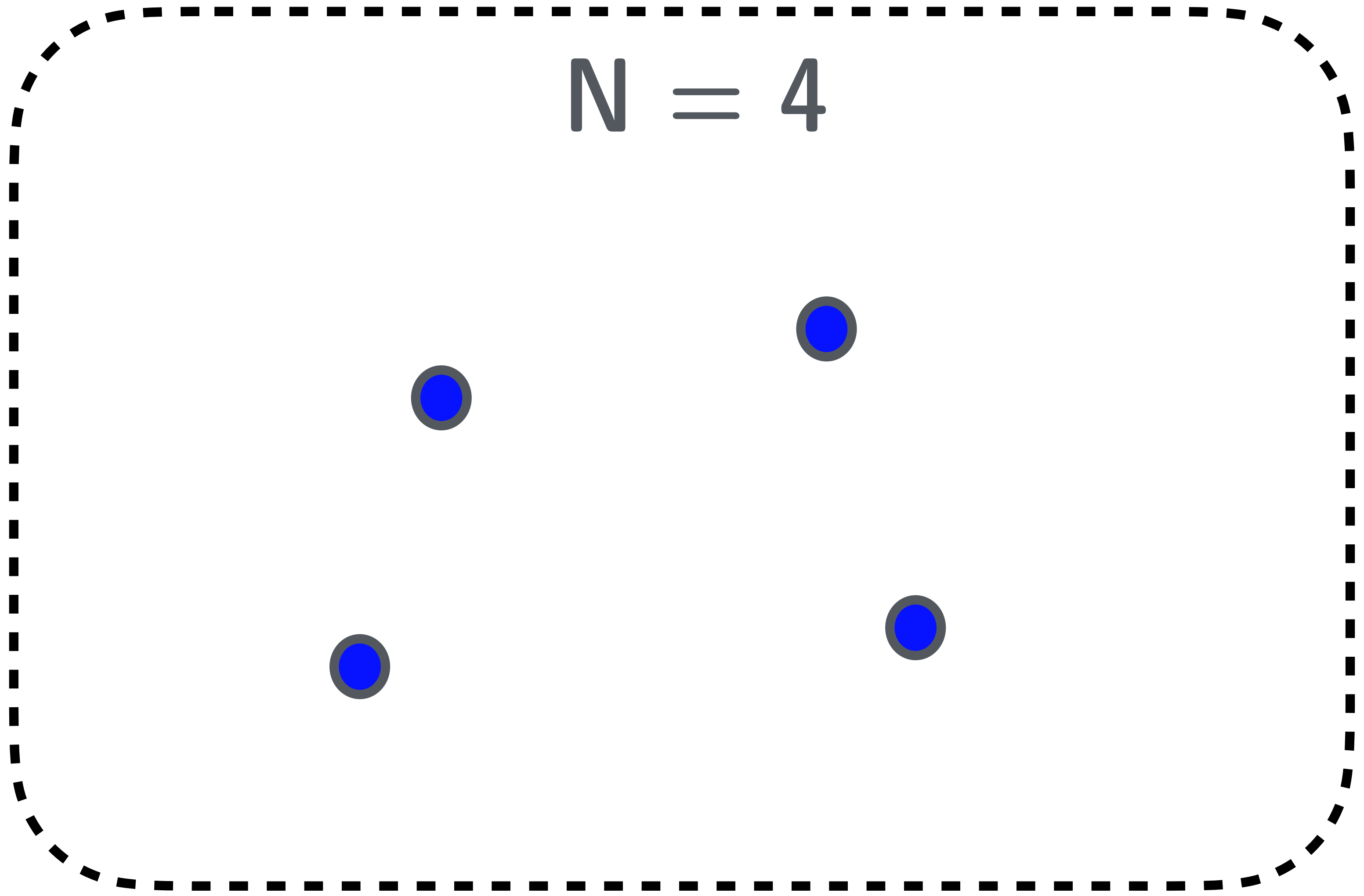
Answer = 0



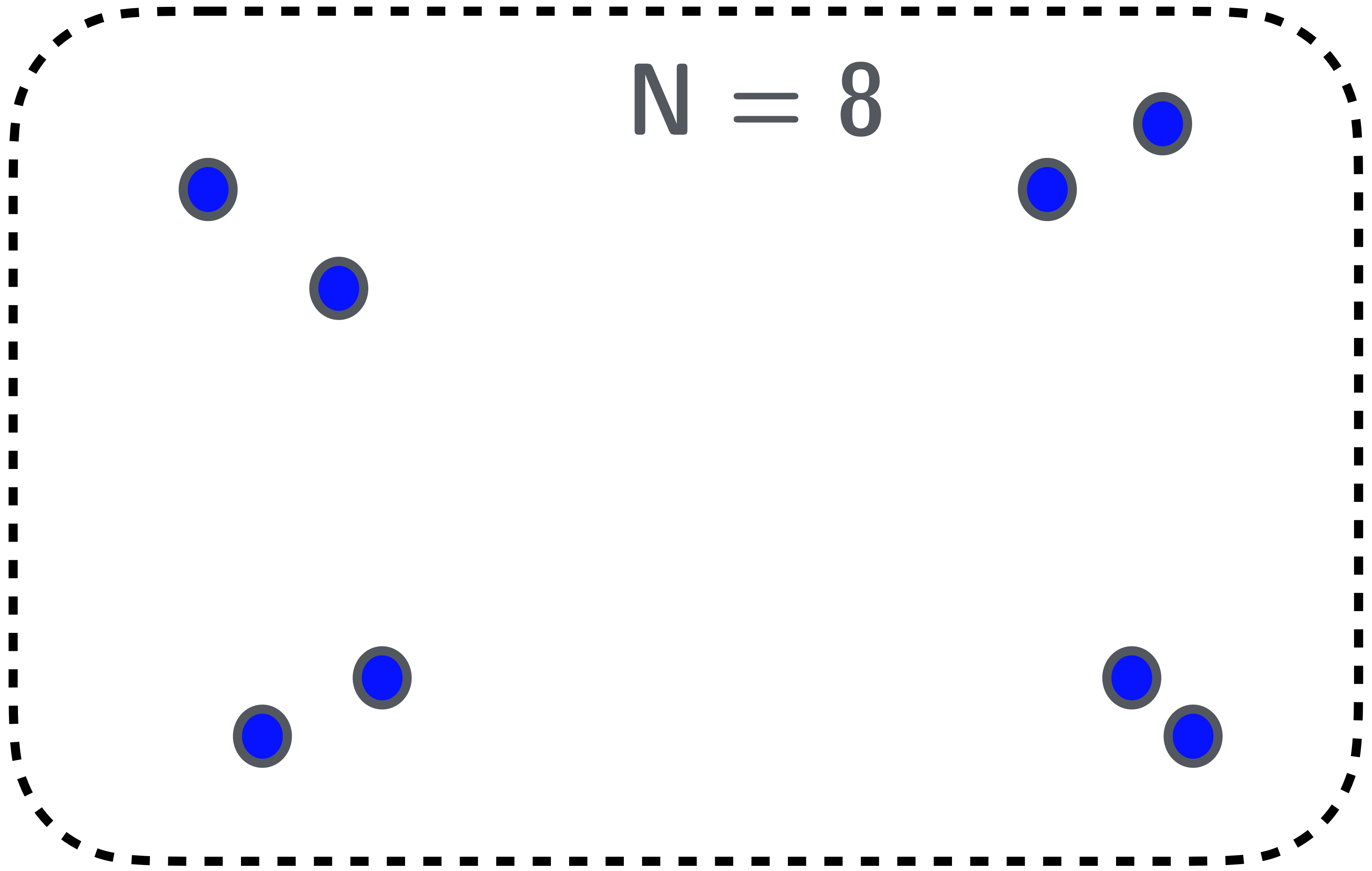
Answer = 0

Problem: Minimizing Convex Pentagons

Given an integer N, what is, $\mu_5(N)$ the minimum number of convex pentagons we can get by placing N points in the Euclidean plane without 3 on a line?



Answer = 0

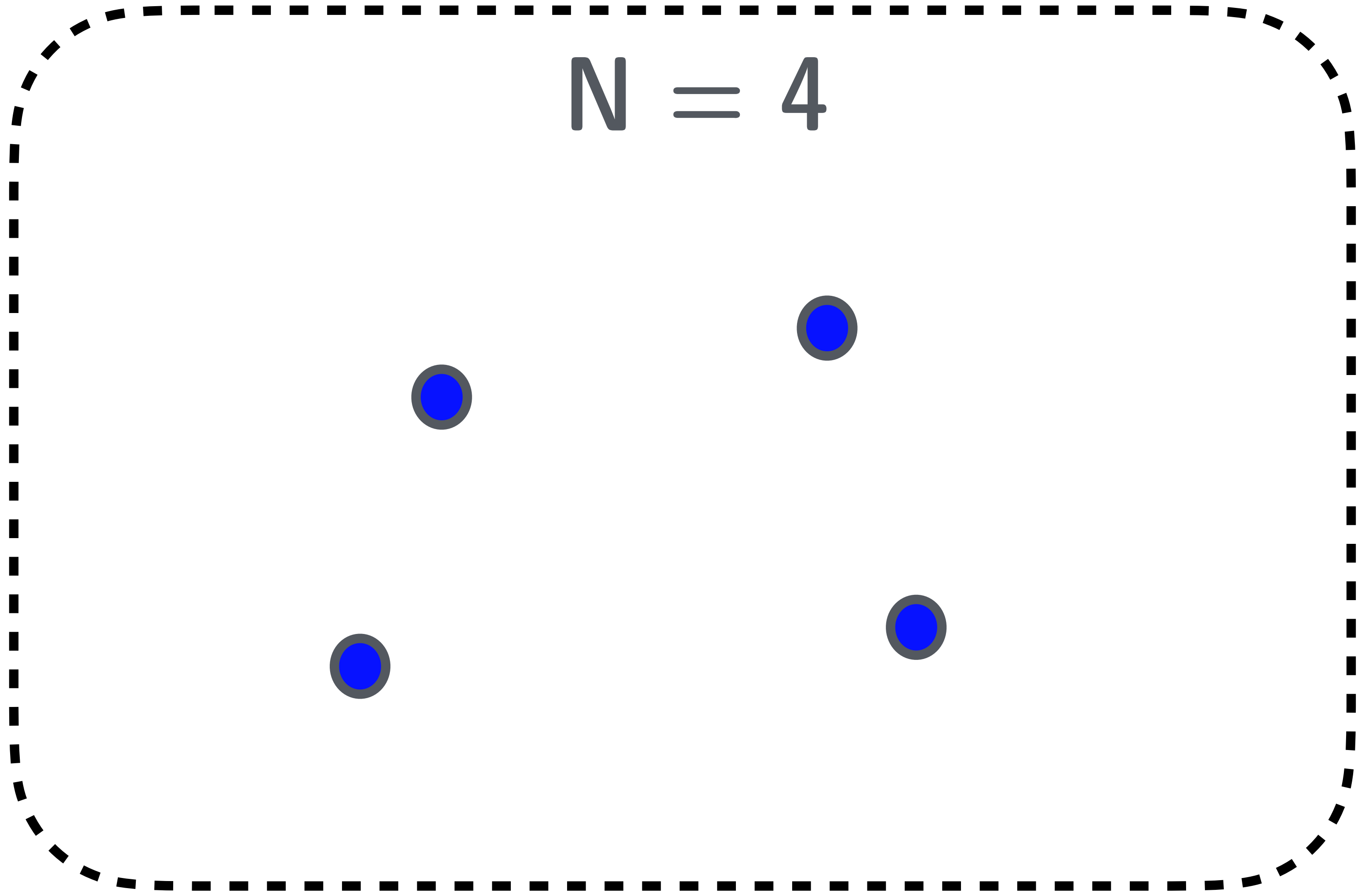


Answer = 0

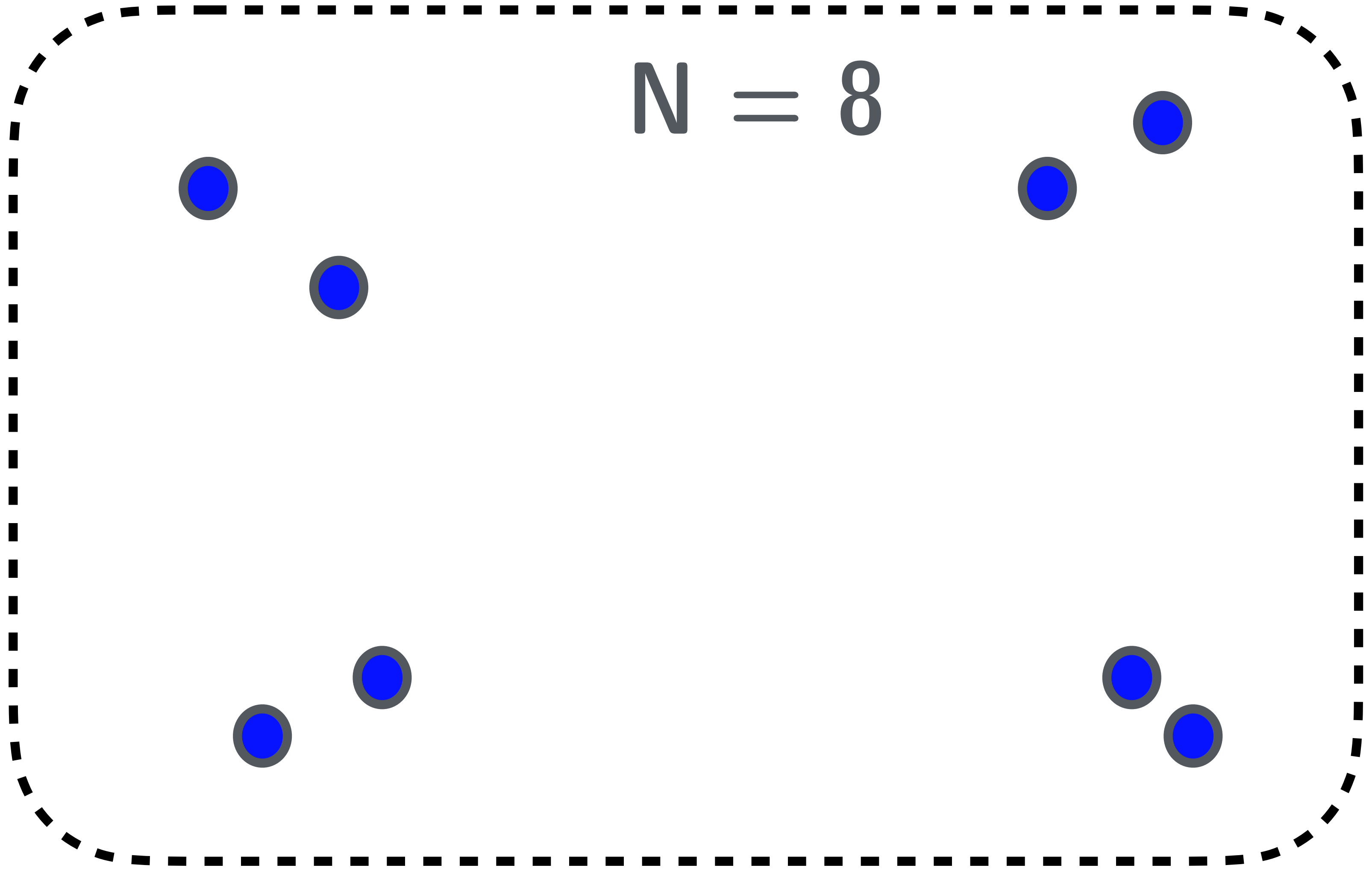
N	$\mu_5(N)$
4	0
8	0

Problem: Minimizing Convex Pentagons

Given an integer N, what is, $\mu_5(N)$ the minimum number of convex pentagons we can get by placing N points in the Euclidean plane without 3 on a line?



Answer = 0

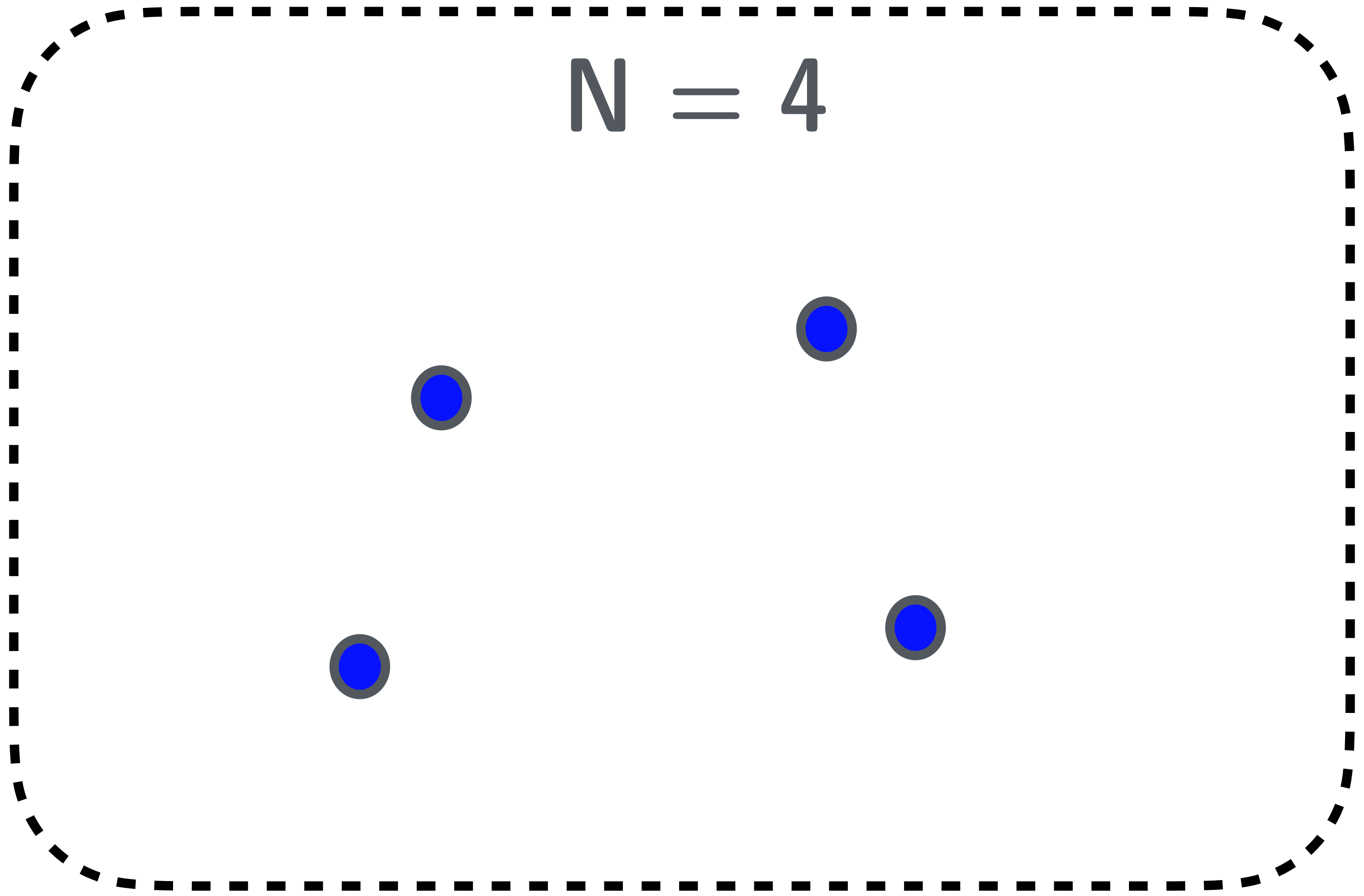


Answer = 0

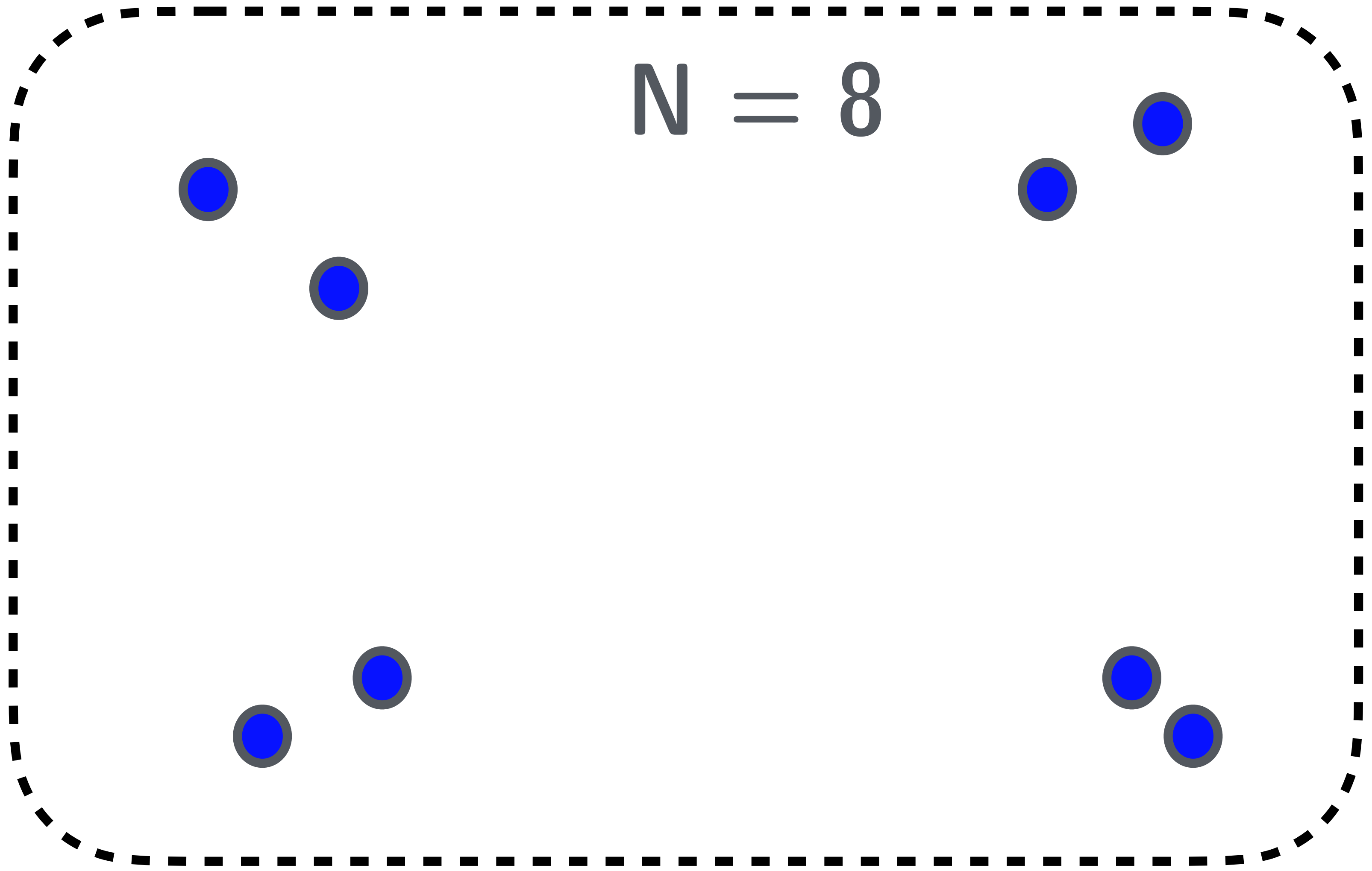
N	$\mu_5(N)$
4	0
8	0
9	

Problem: Minimizing Convex Pentagons

Given an integer N, what is, $\mu_5(N)$ the minimum number of convex pentagons we can get by placing N points in the Euclidean plane without 3 on a line?



Answer = 0

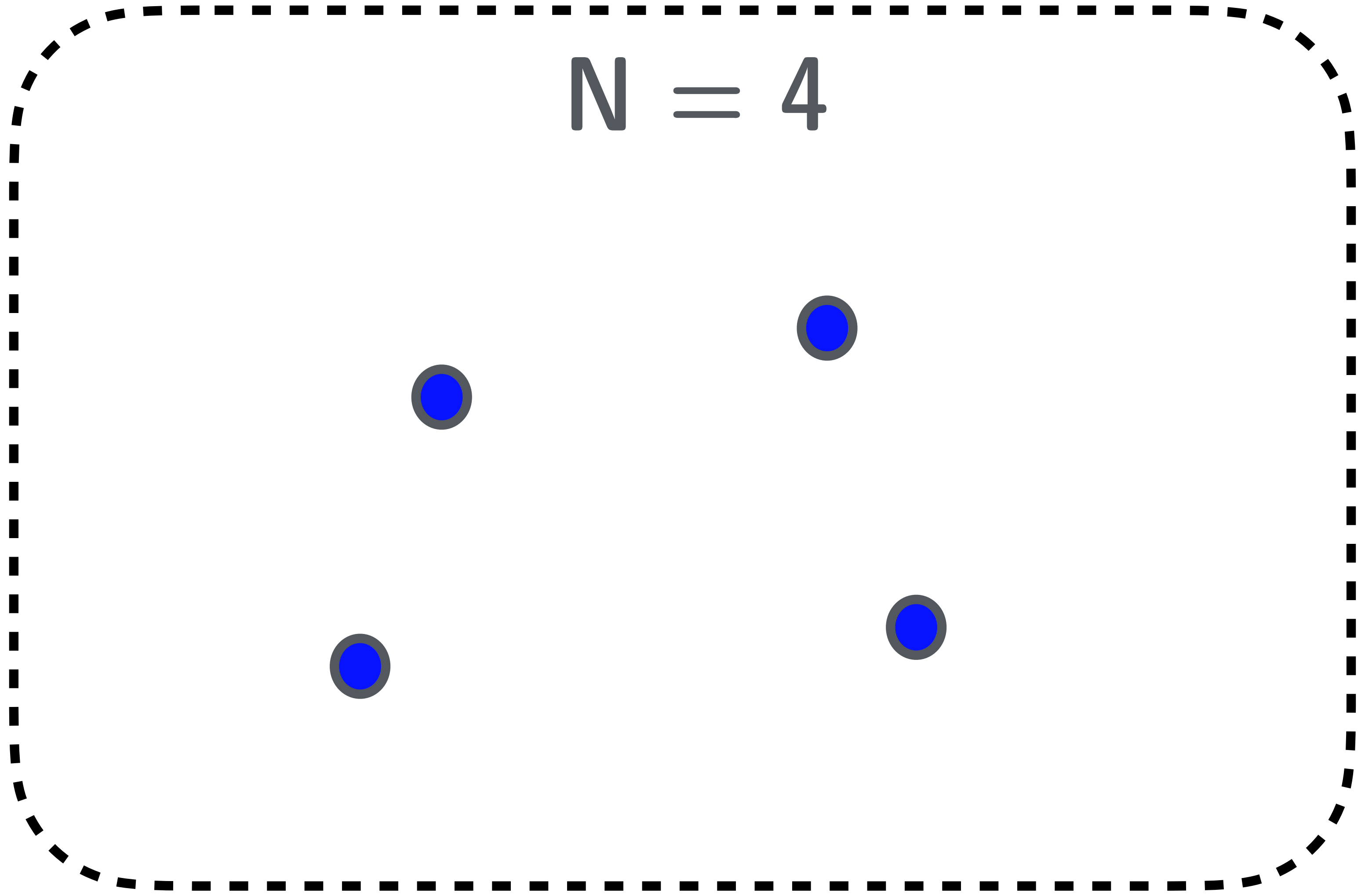


Answer = 0

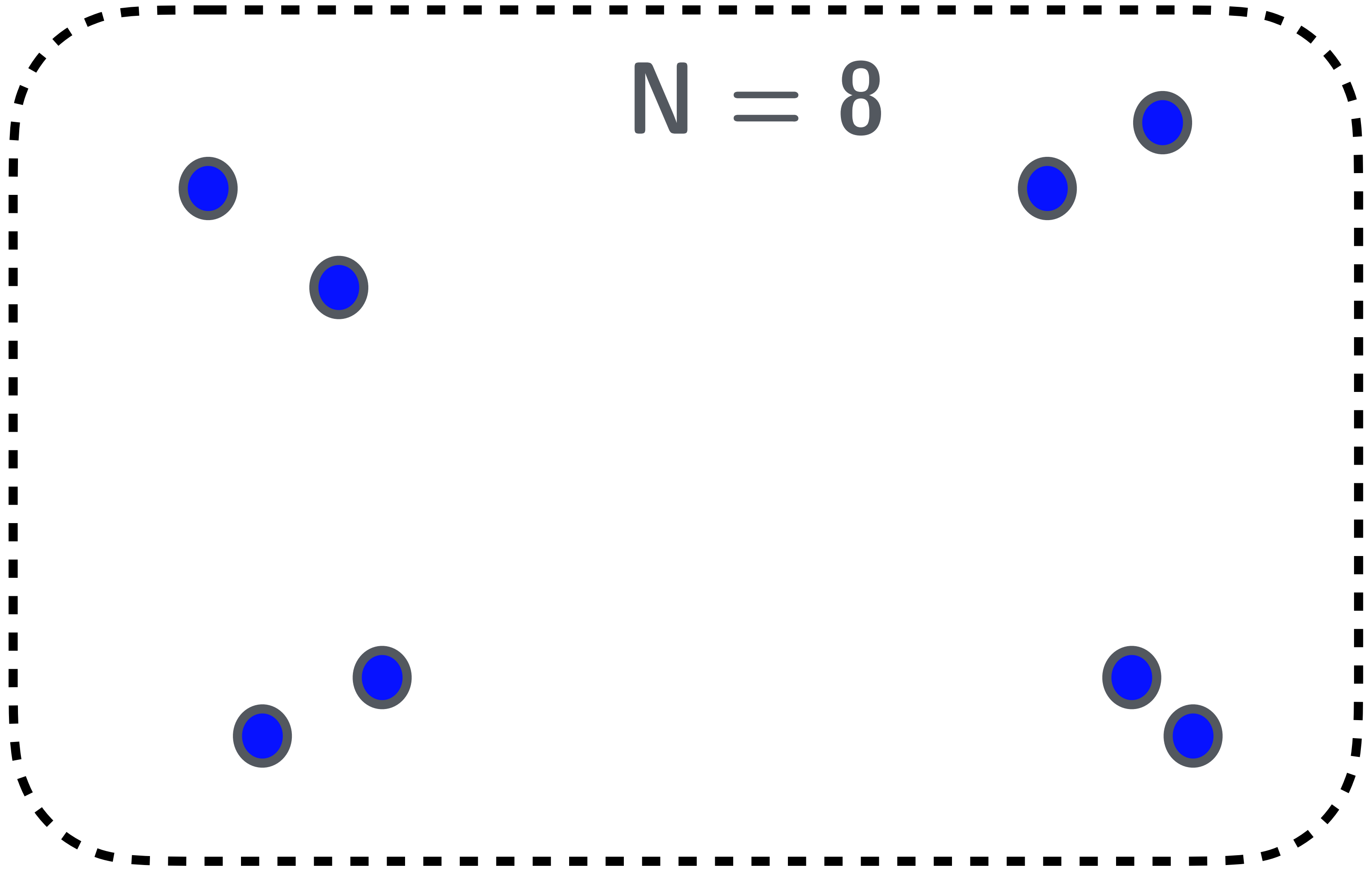
N	$\mu_5(N)$
4	0
8	0
9	1

Problem: Minimizing Convex Pentagons

Given an integer N, what is, $\mu_5(N)$ the minimum number of convex pentagons we can get by placing N points in the Euclidean plane without 3 on a line?



Answer = 0

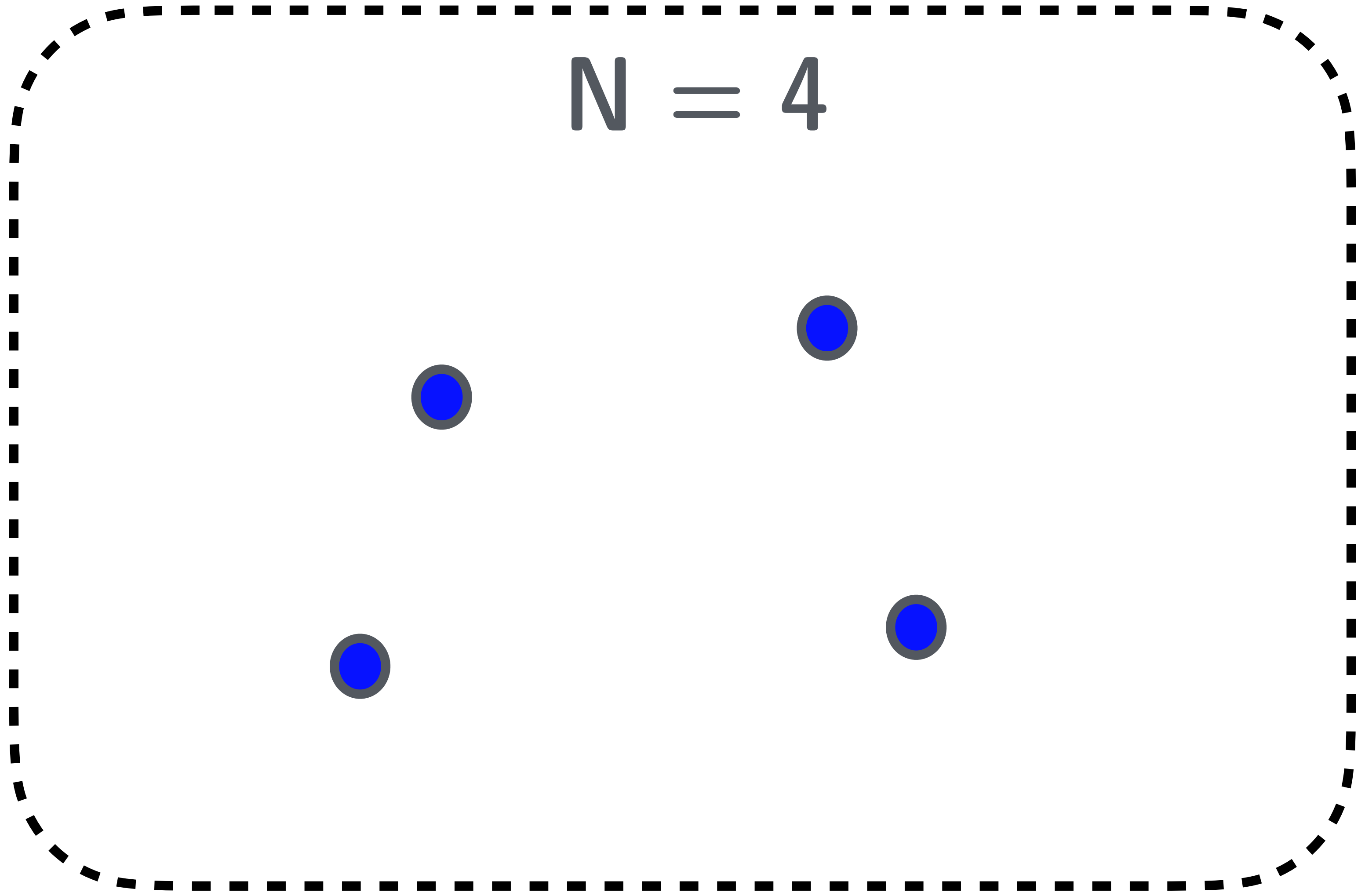


Answer = 0

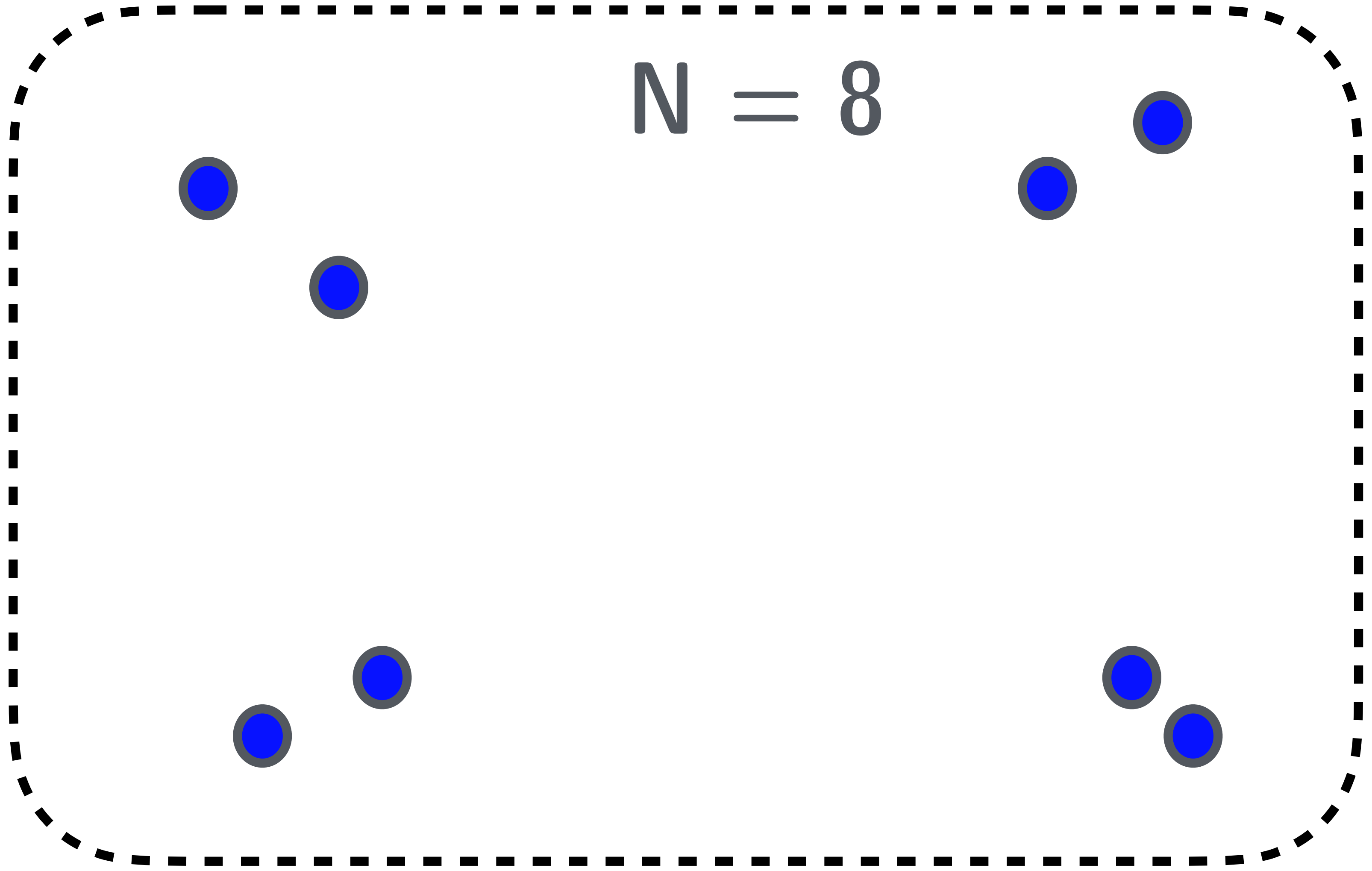
N	$\mu_5(N)$
4	0
8	0
9	1
15	

Problem: Minimizing Convex Pentagons

Given an integer N, what is, $\mu_5(N)$ the minimum number of convex pentagons we can get by placing N points in the Euclidean plane without 3 on a line?



Answer = 0

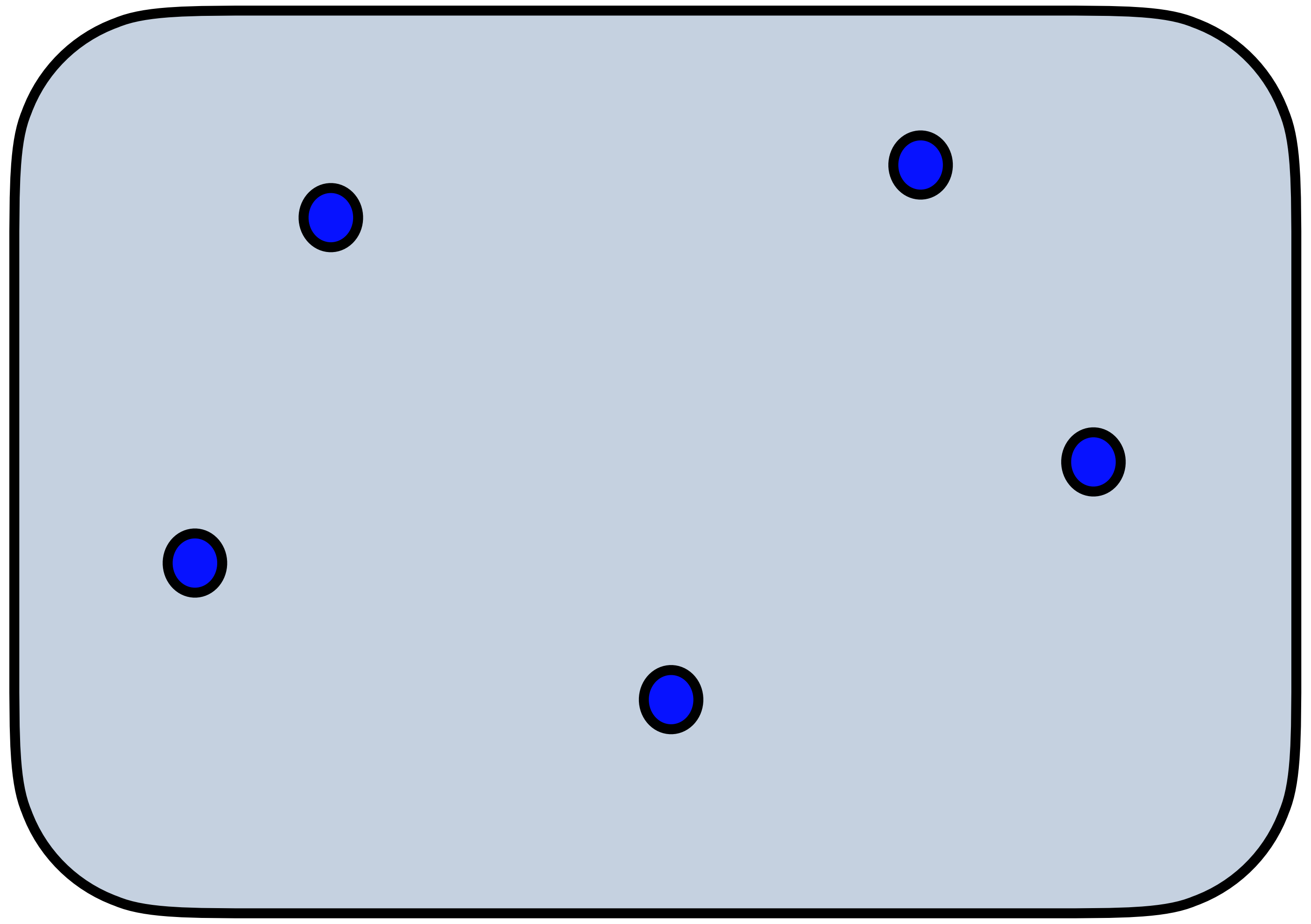


Answer = 0

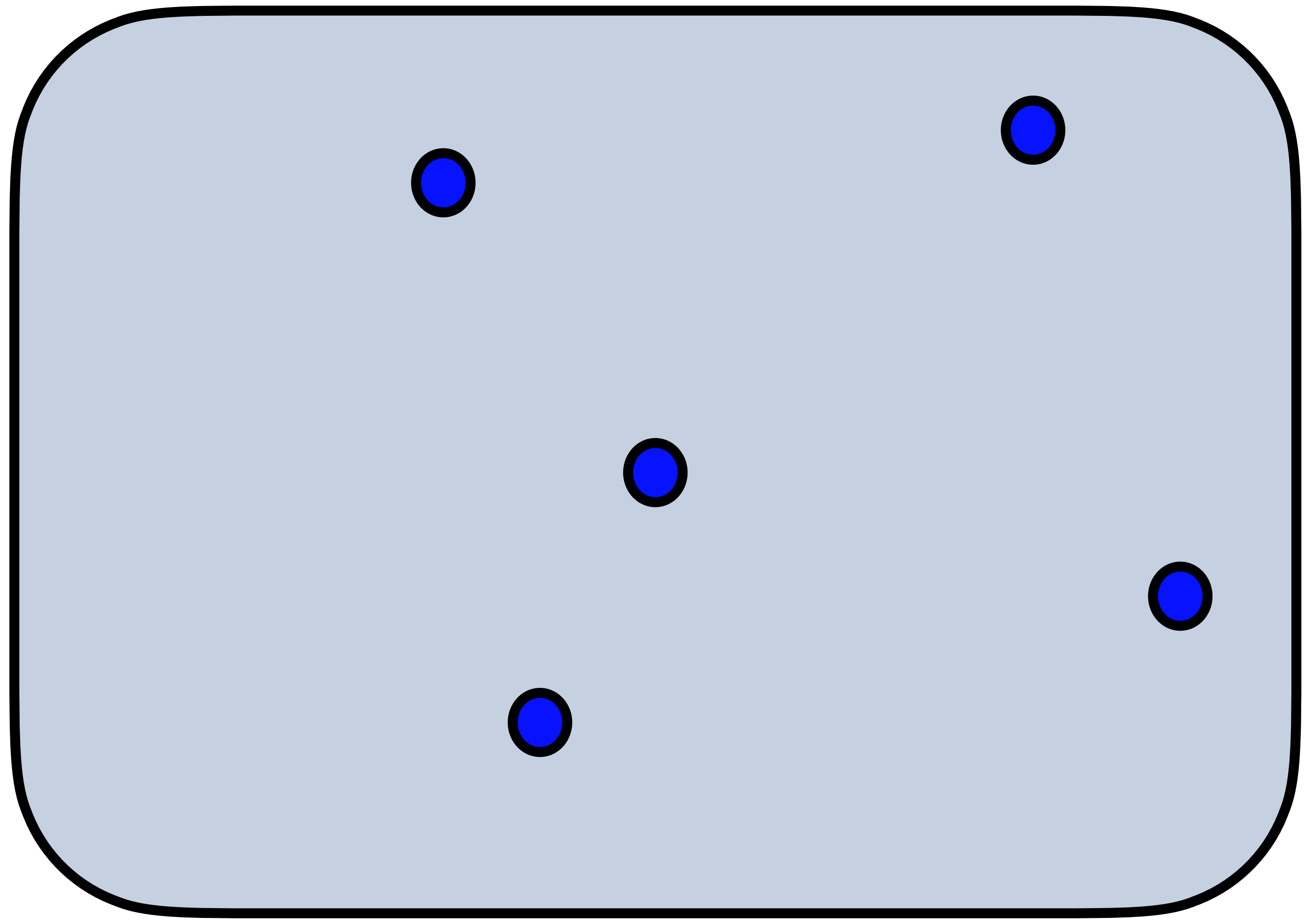
N	$\mu_5(N)$
4	0
8	0
9	1
15	**77**

Let's do something easier first...

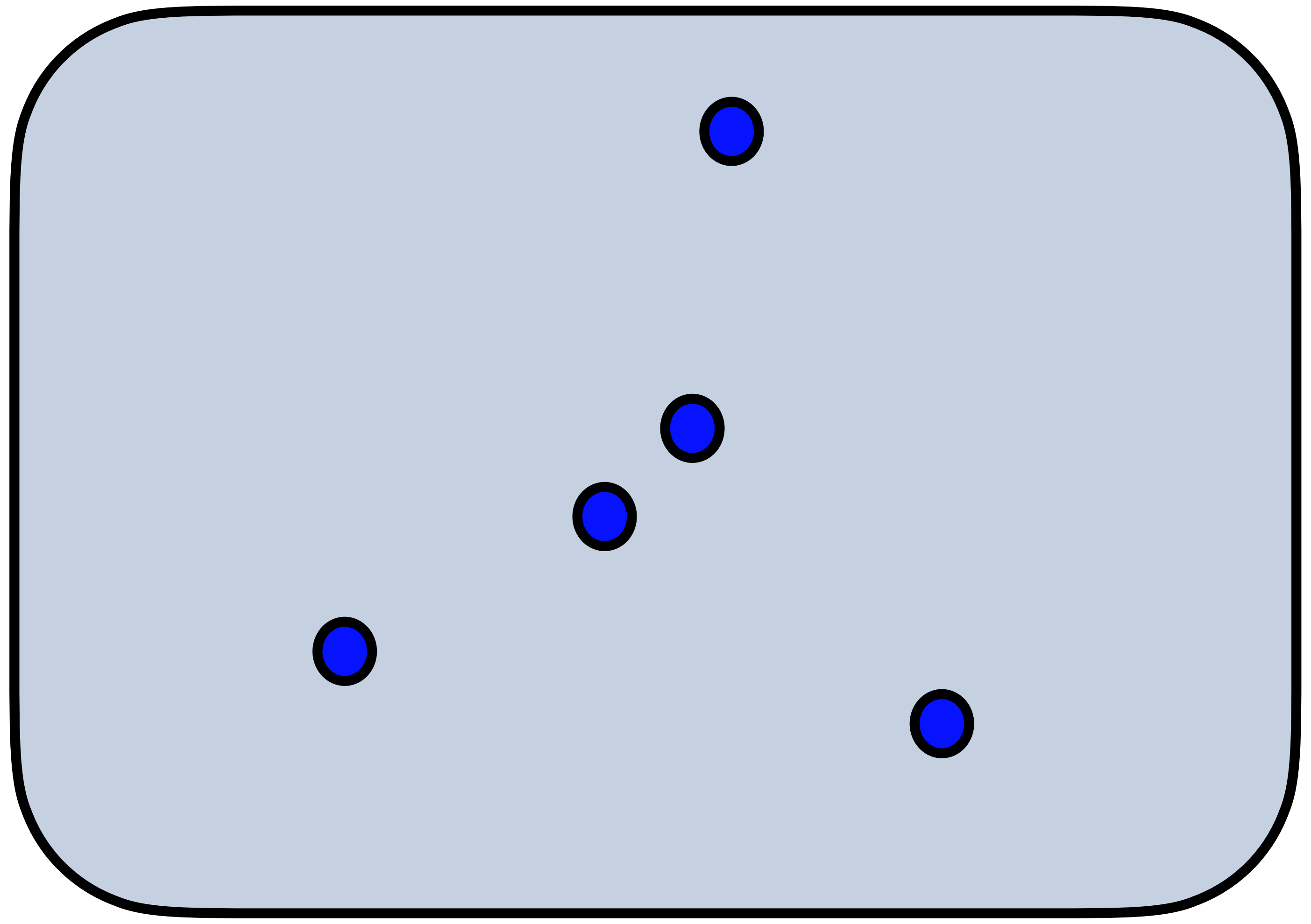
How many points without 3 on a line guarantee a **convex quadrilateral**?



Case 1: 5 points in Convex Hull



Case 2: 4 points in Convex Hull

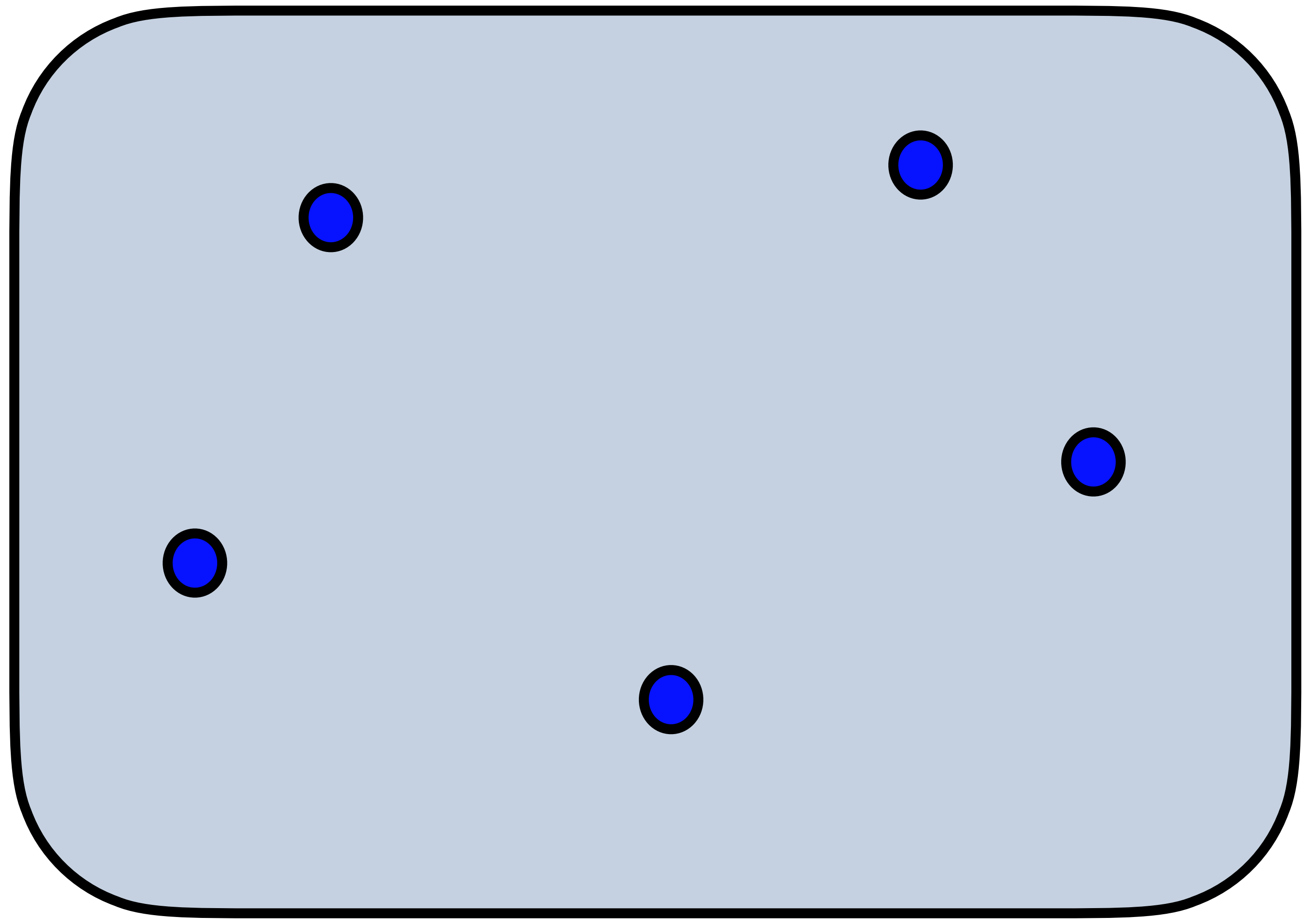


Case 3: 3 points in Convex Hull

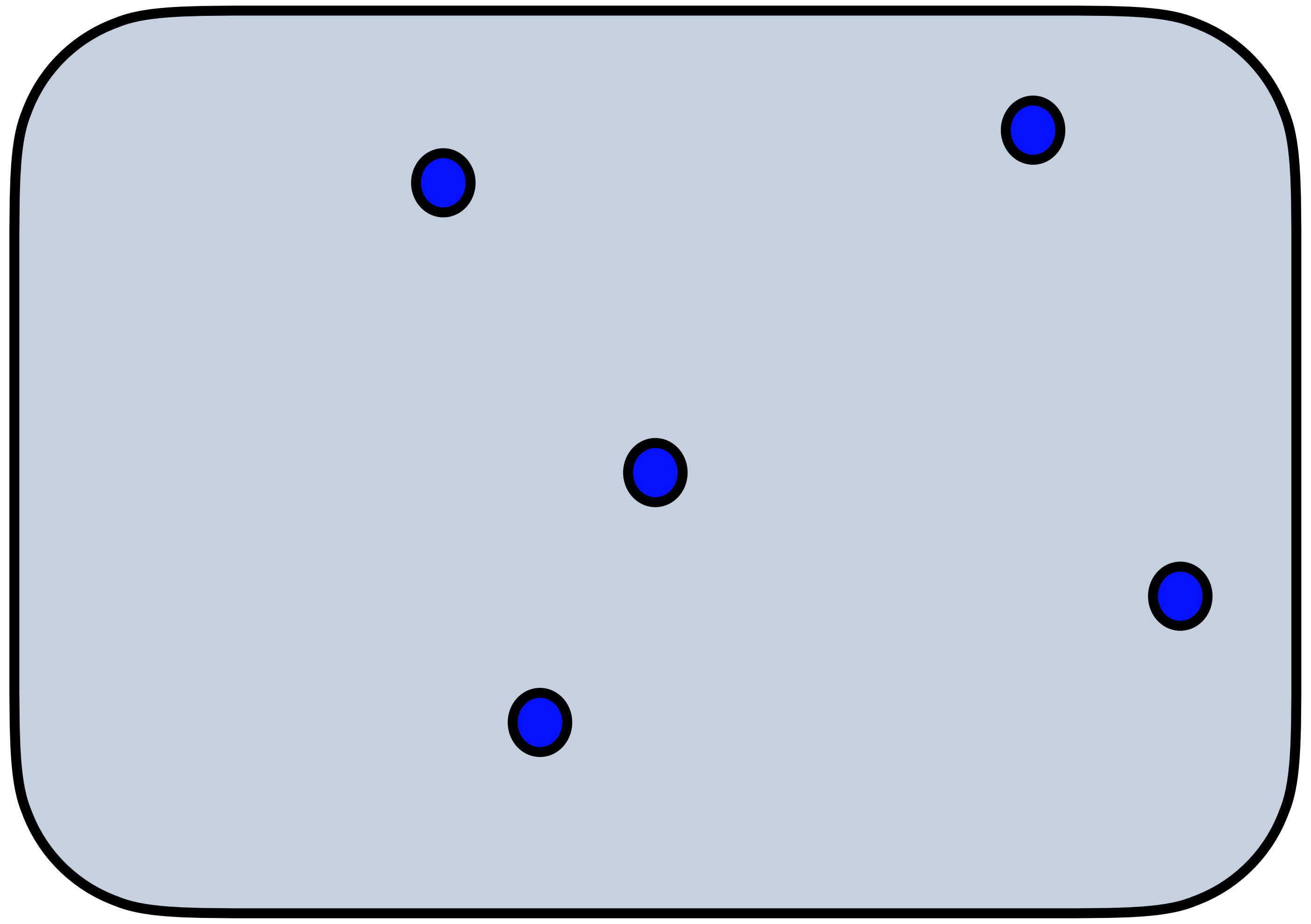
Let's do something easier first...

How many points without 3 on a line guarantee a **convex quadrilateral**?

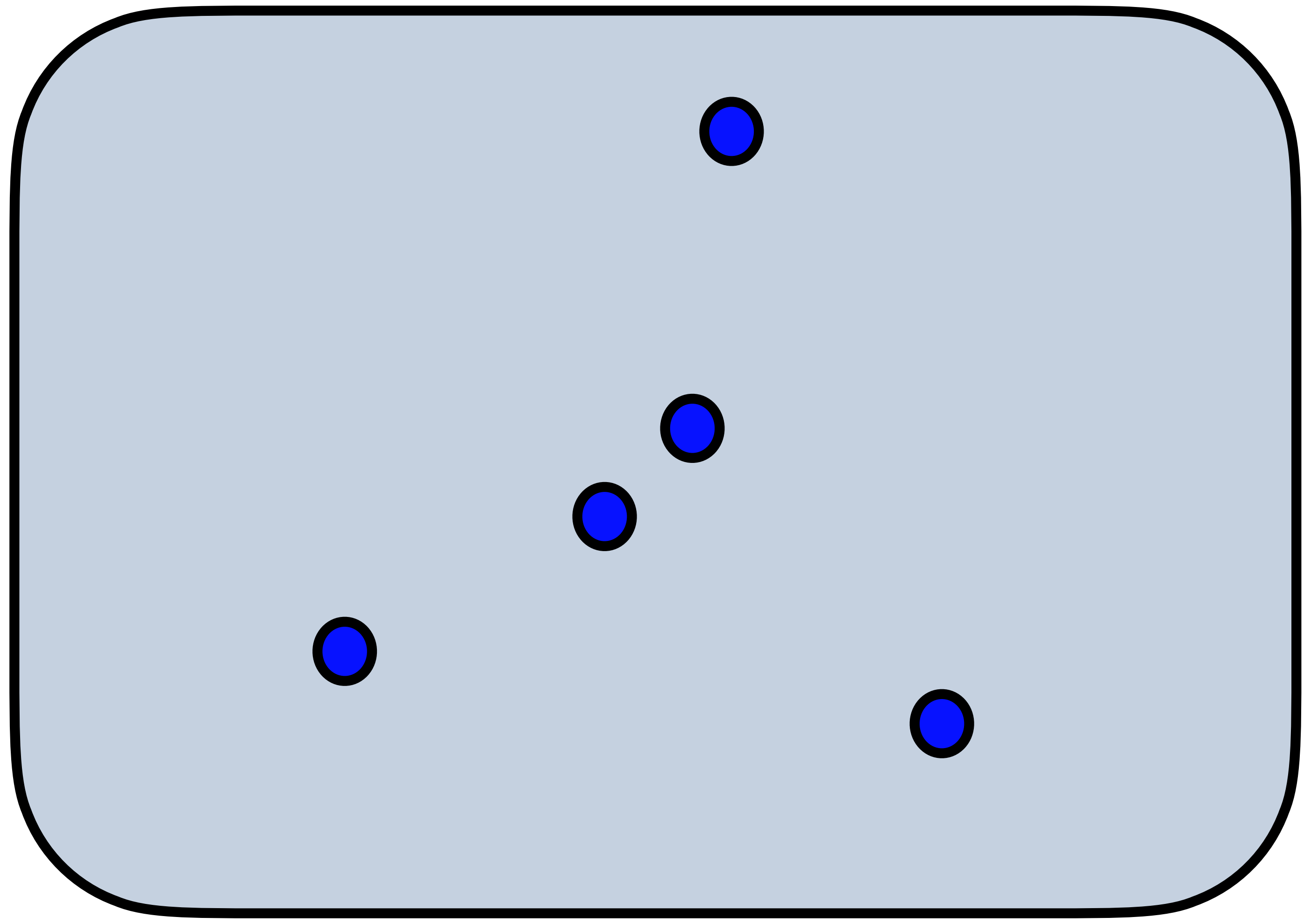
Answer: 5, Klein, *"Happy Ending Theorem"*



Case 1: 5 points in Convex Hull



Case 2: 4 points in Convex Hull

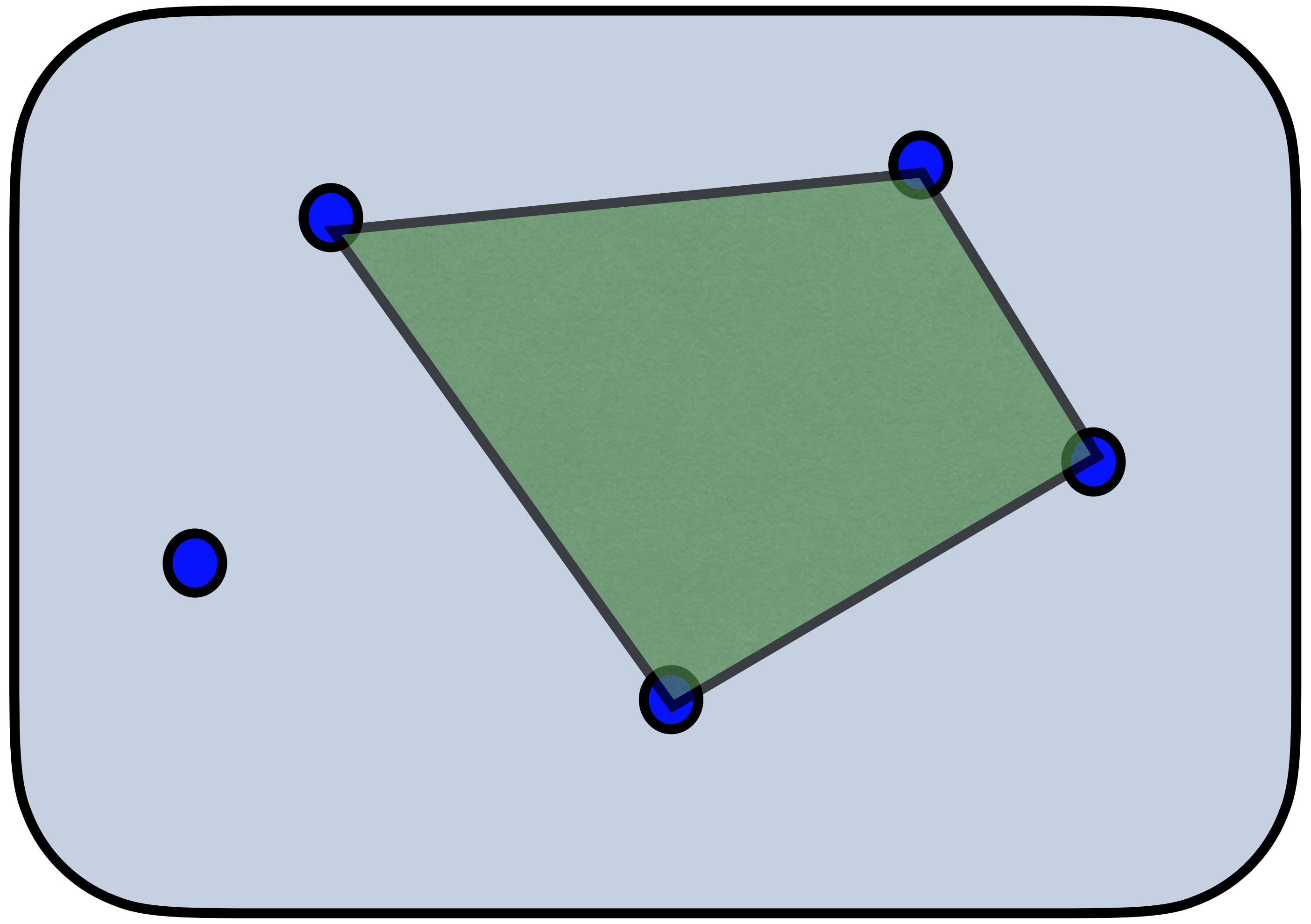


Case 3: 3 points in Convex Hull

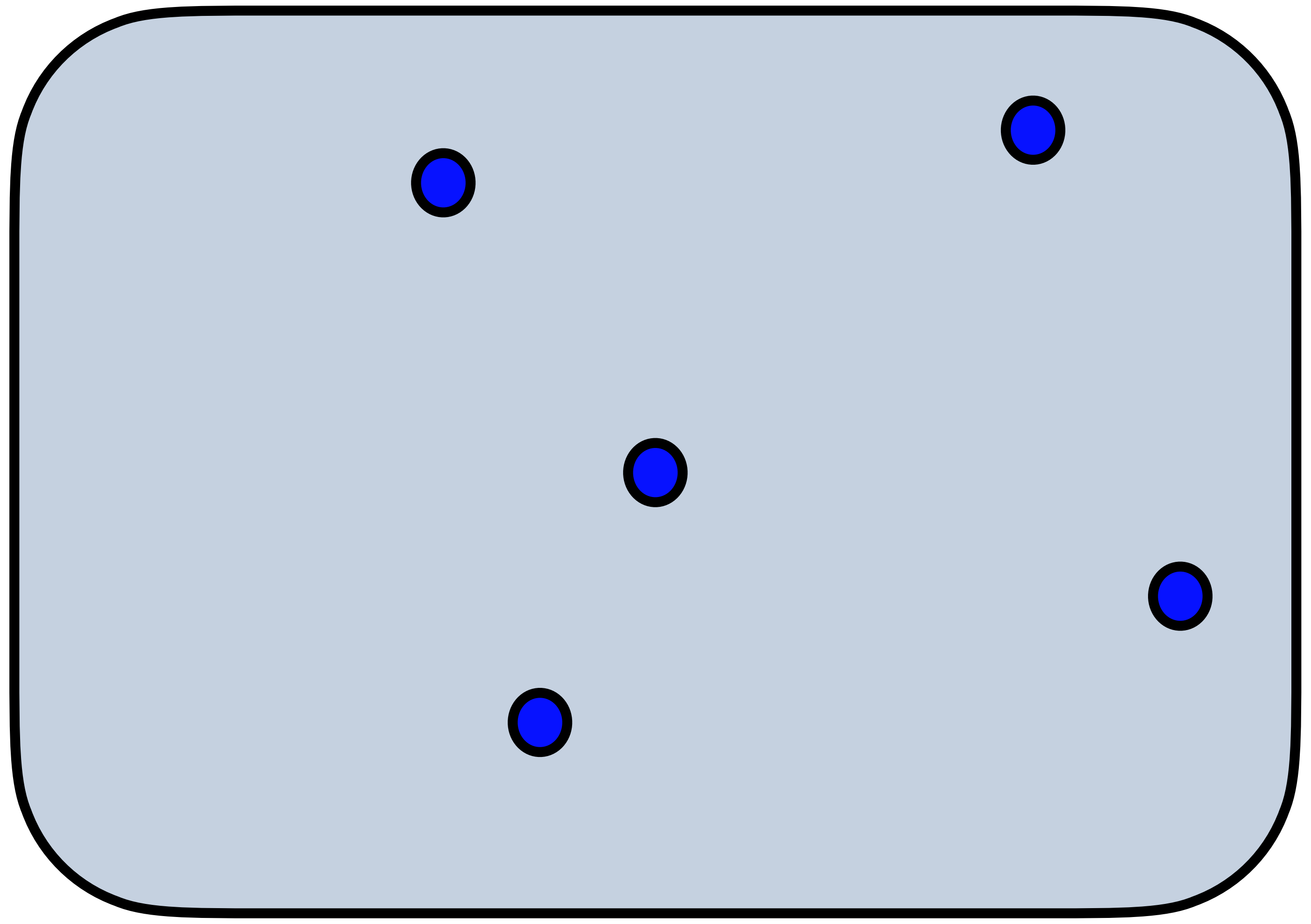
Let's do something easier first...

How many points without 3 on a line guarantee a **convex quadrilateral**?

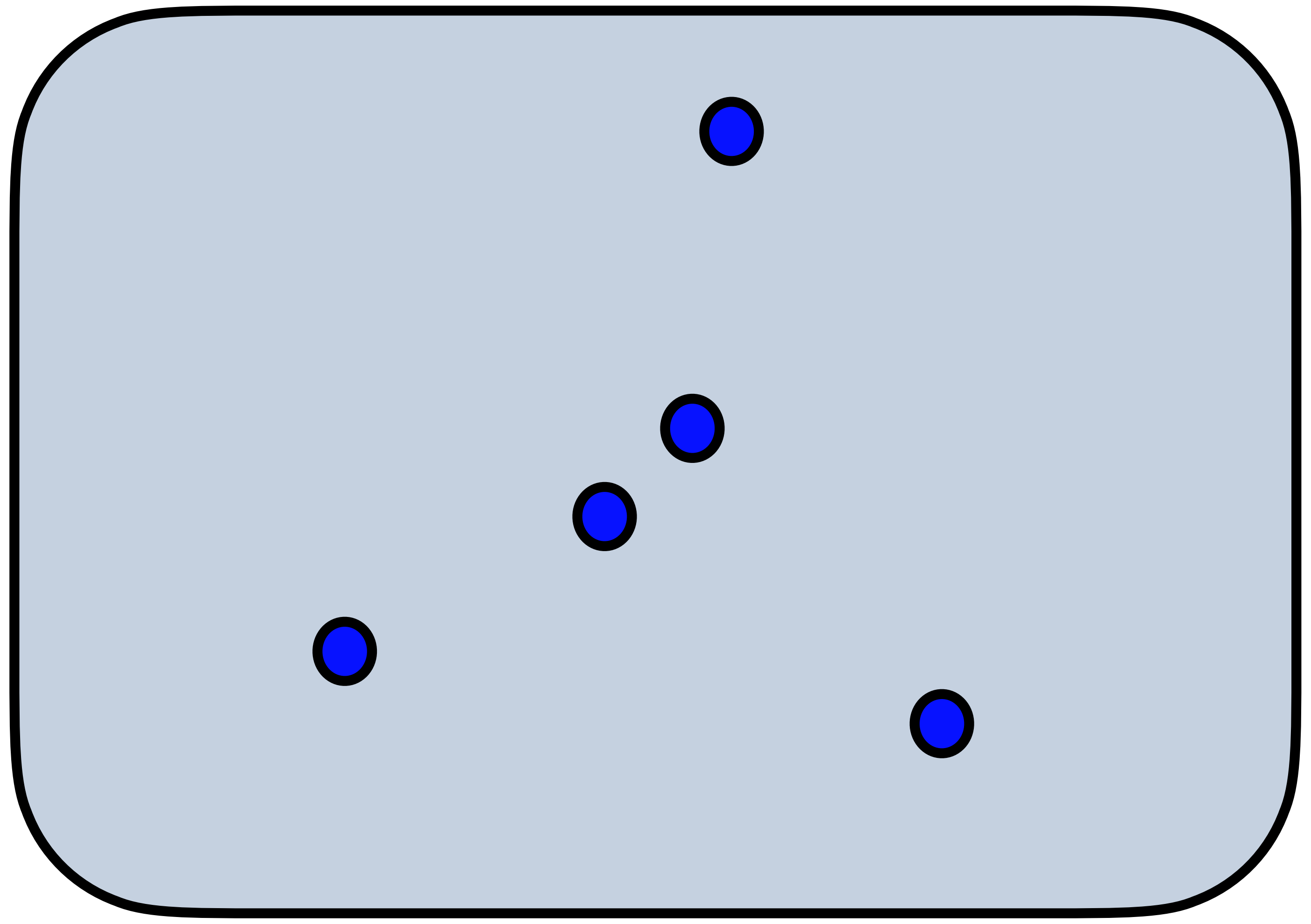
Answer: 5, Klein, *"Happy Ending Theorem"*



Case 1: 5 points in Convex Hull



Case 2: 4 points in Convex Hull

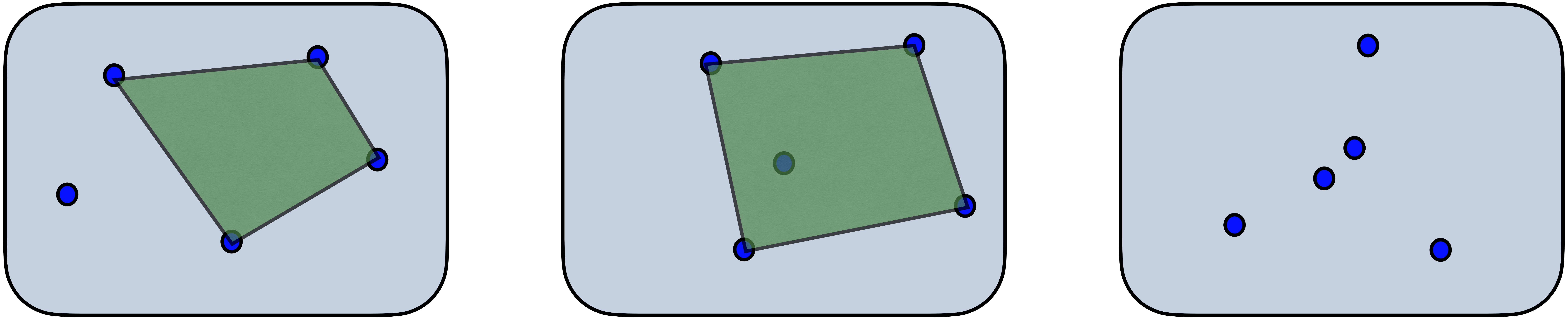


Case 3: 3 points in Convex Hull

Let's do something easier first...

How many points without 3 on a line guarantee a **convex quadrilateral**?

Answer: 5, Klein, *"Happy Ending Theorem"*



Case 1: 5 points in Convex Hull

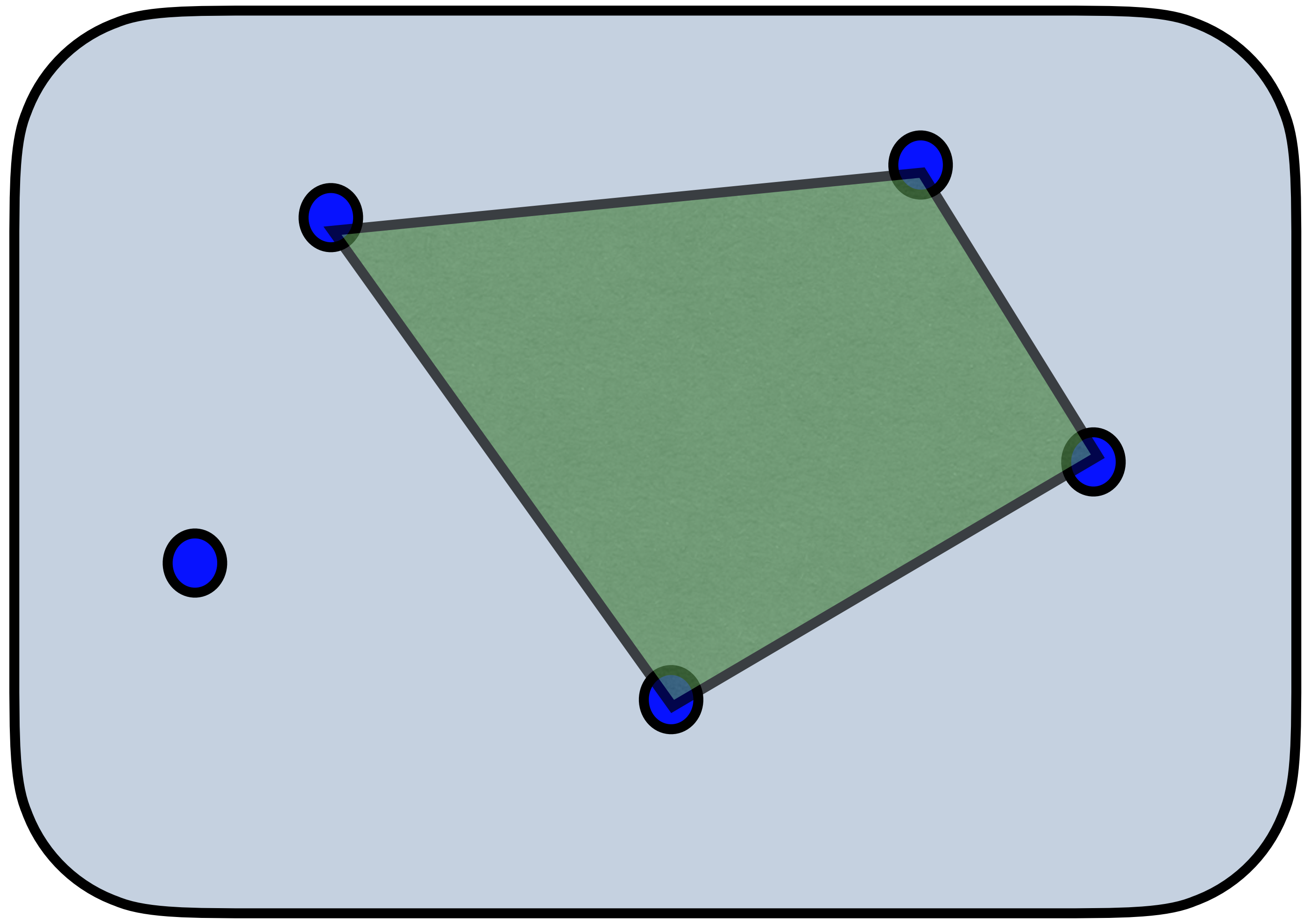
Case 2: 4 points in Convex Hull

Case 3: 3 points in Convex Hull

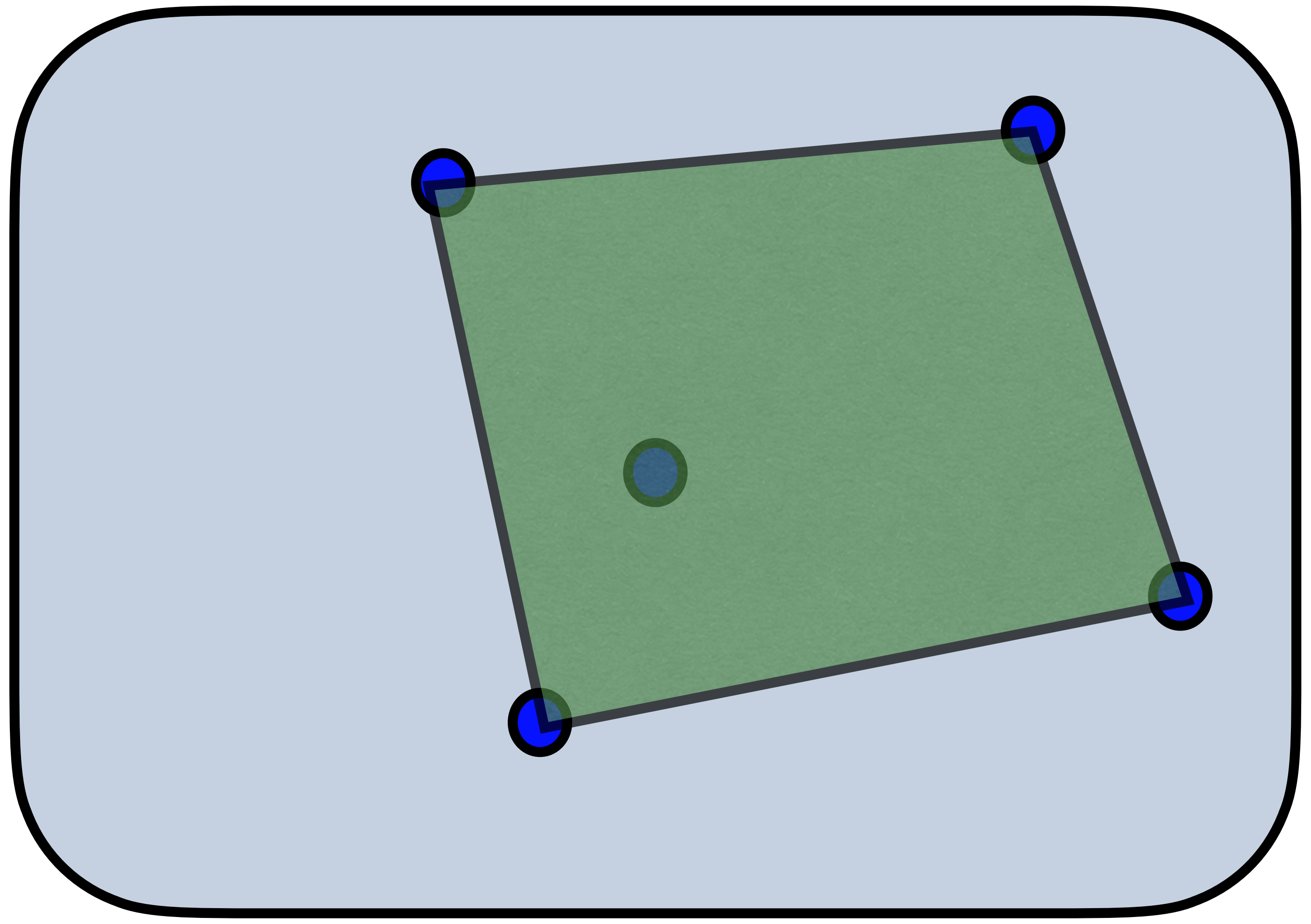
Let's do something easier first...

How many points without 3 on a line guarantee a **convex quadrilateral**?

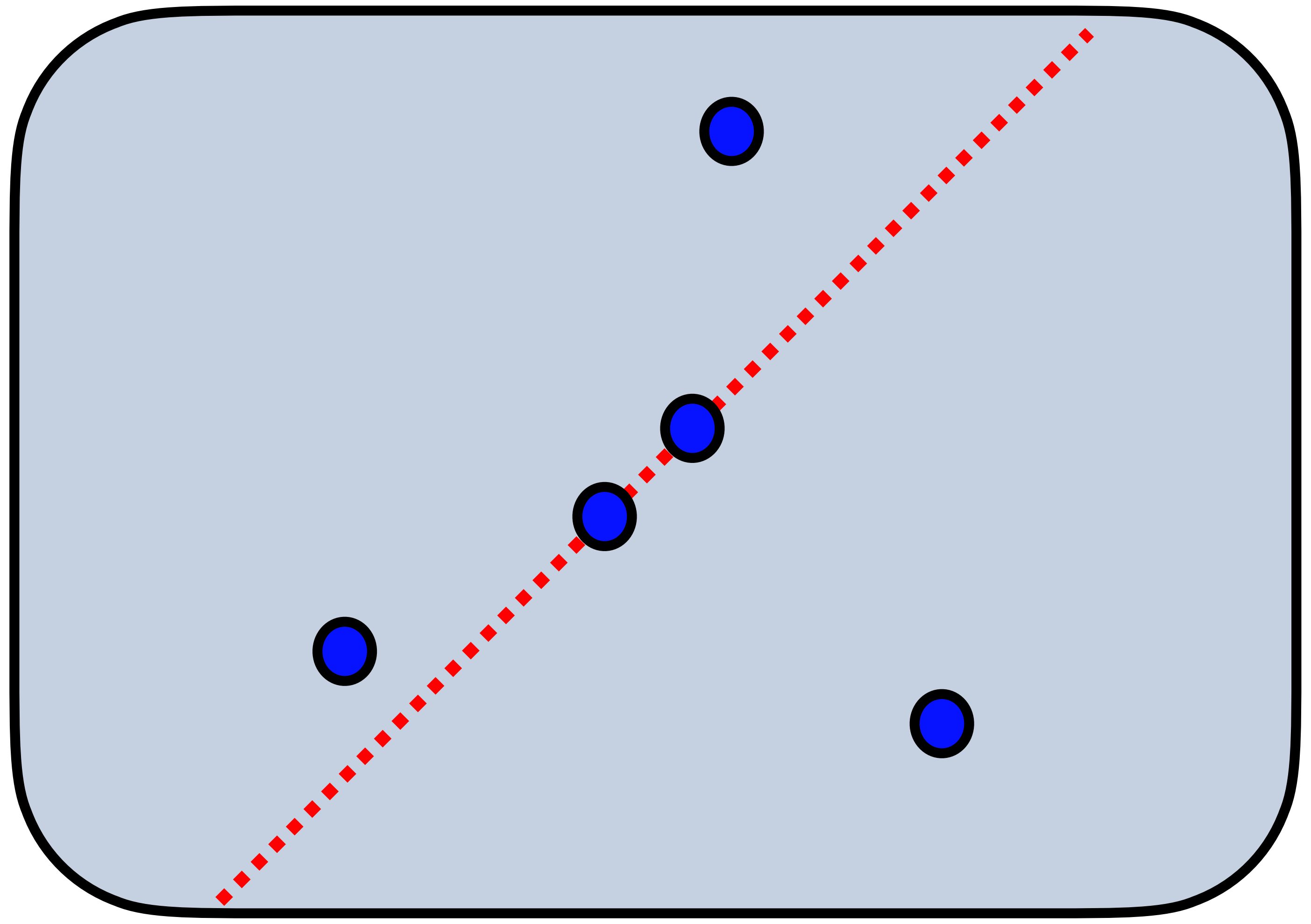
Answer: 5, Klein, *"Happy Ending Theorem"*



Case 1: 5 points in Convex Hull



Case 2: 4 points in Convex Hull

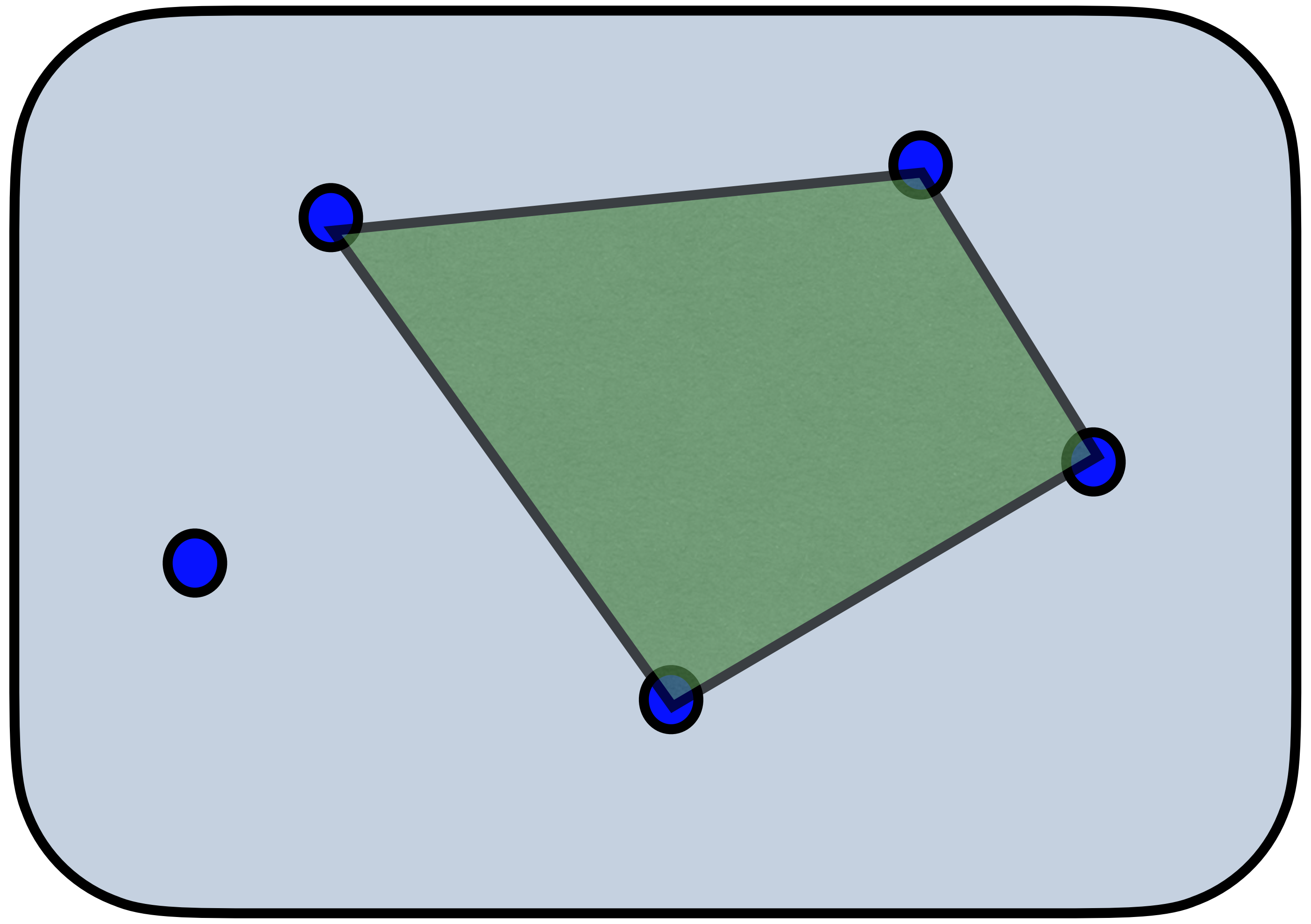


Case 3: 3 points in Convex Hull

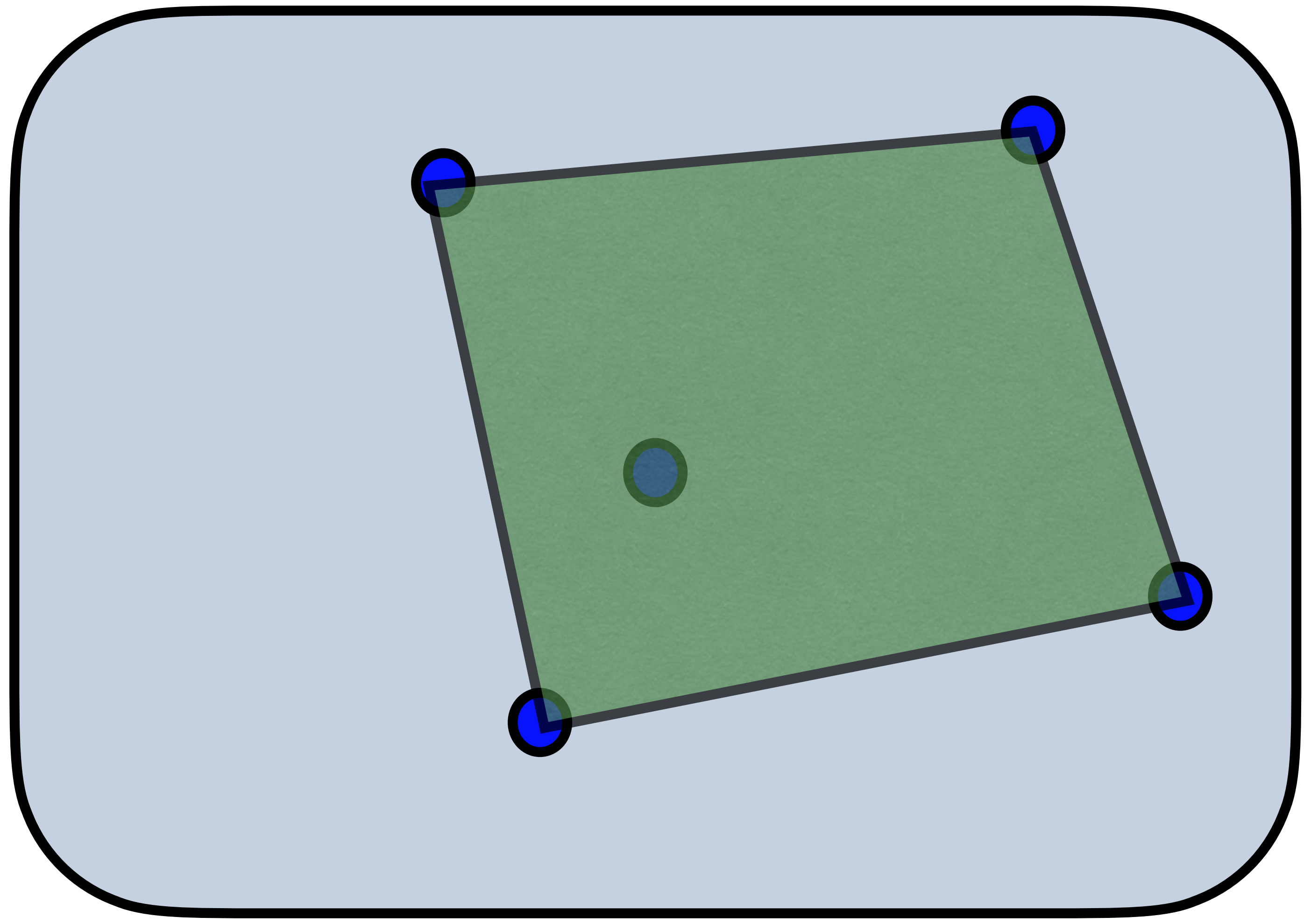
Let's do something easier first...

How many points without 3 on a line guarantee a **convex quadrilateral**?

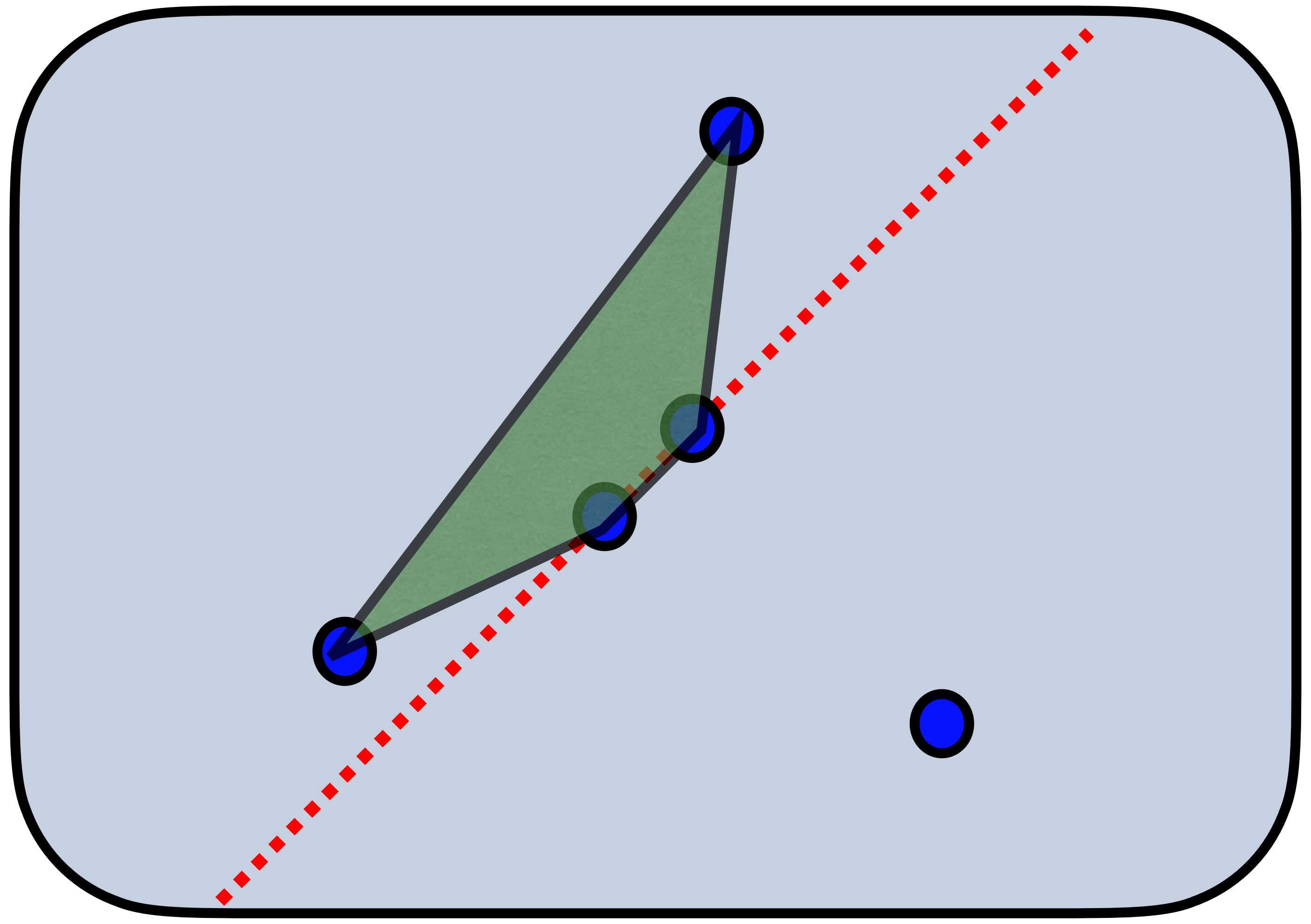
Answer: 5, Klein, *"Happy Ending Theorem"*



Case 1: 5 points in Convex Hull



Case 2: 4 points in Convex Hull



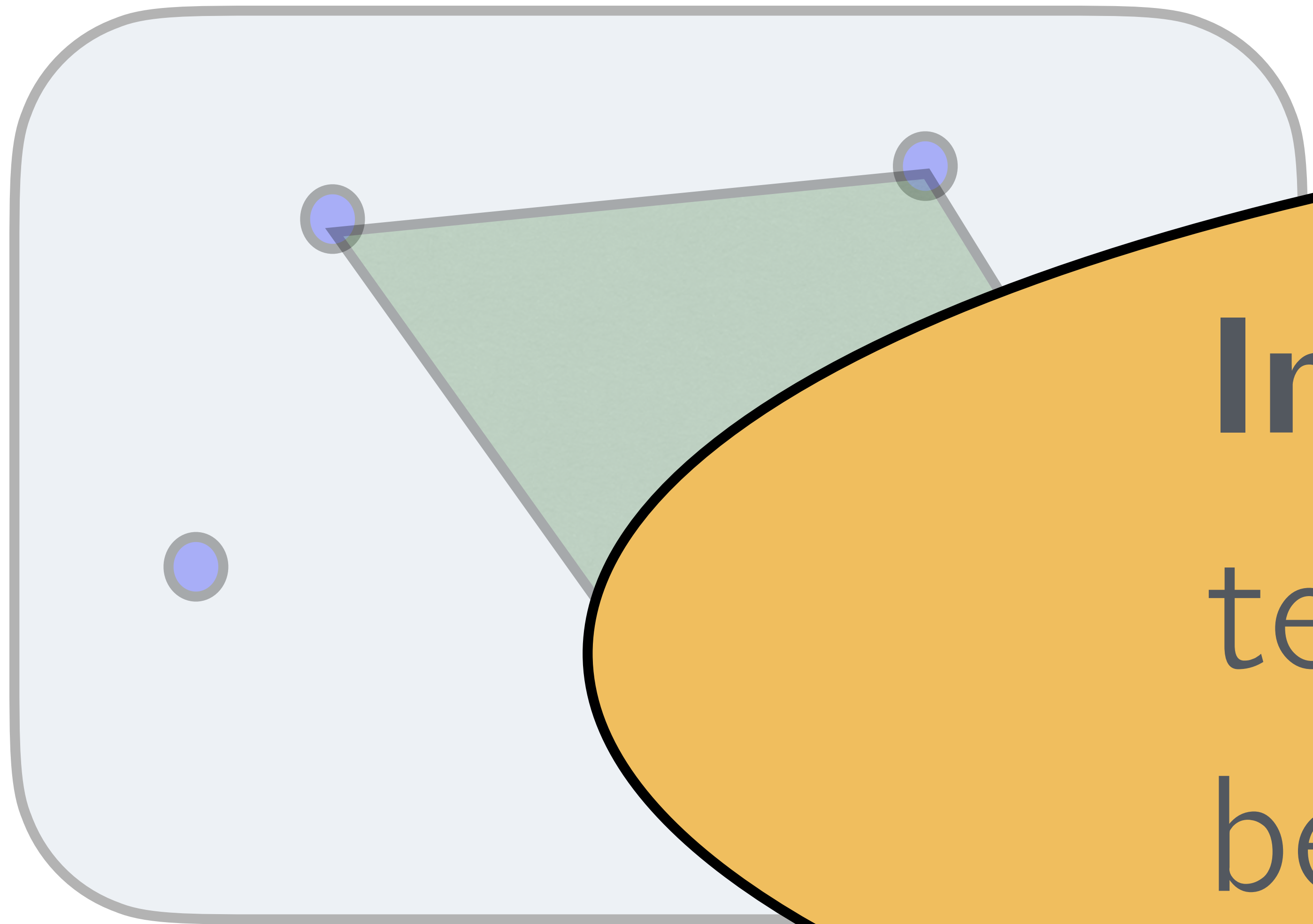
Case 3: 3 points in Convex Hull

Let's do something easier first...

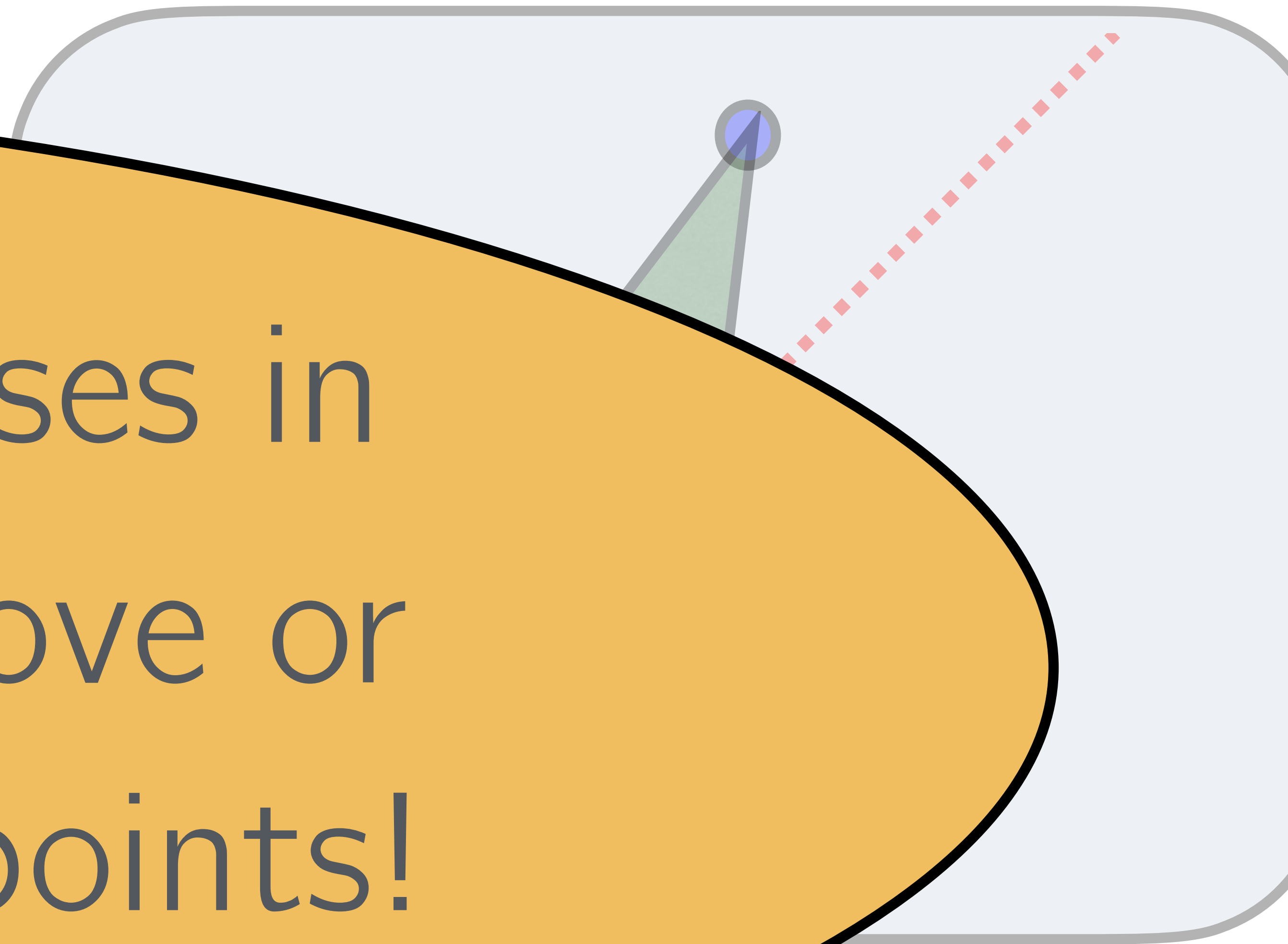
How many points without 3 on a line guarantee a **convex quadrilateral**?

Answer: 5, Klein, *"Happy Ending Theorem"*

Insight: we can reason by cases in terms of which points are above or below lines formed by other points!

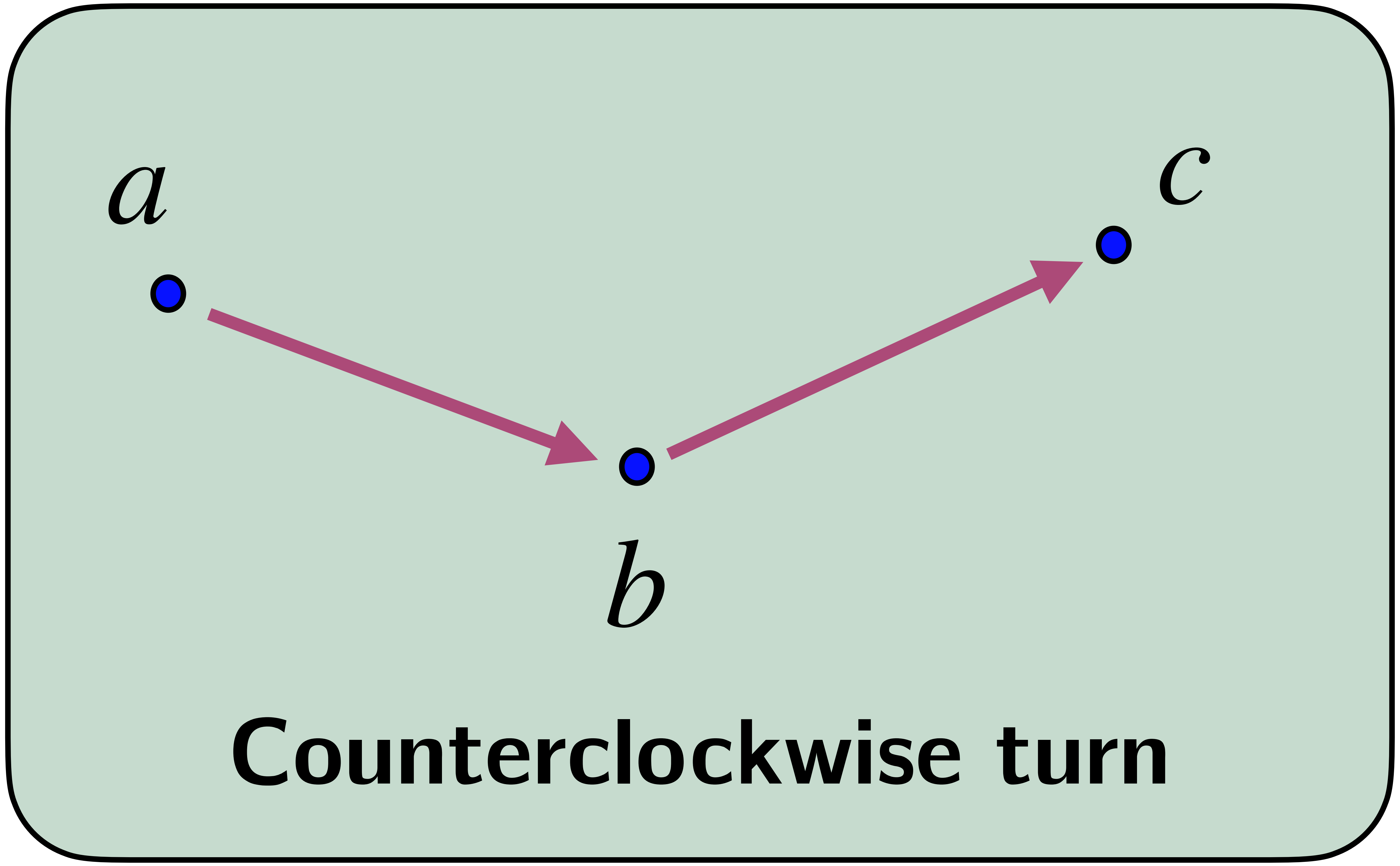


Case 1: 5 points in Convex

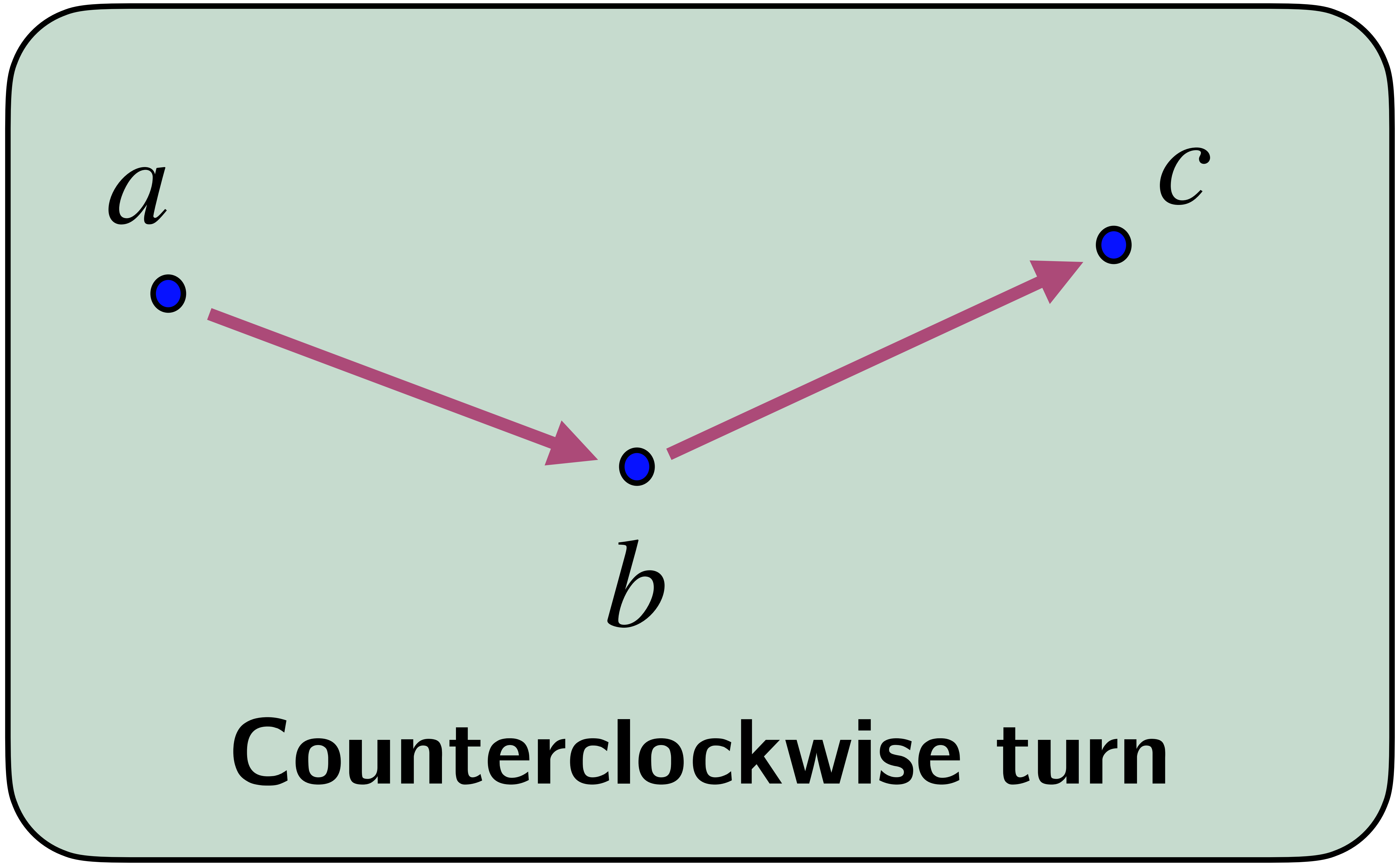


points in Convex Hull

Combinatorial Structure: Triple Orientations



Combinatorial Structure: Triple Orientations

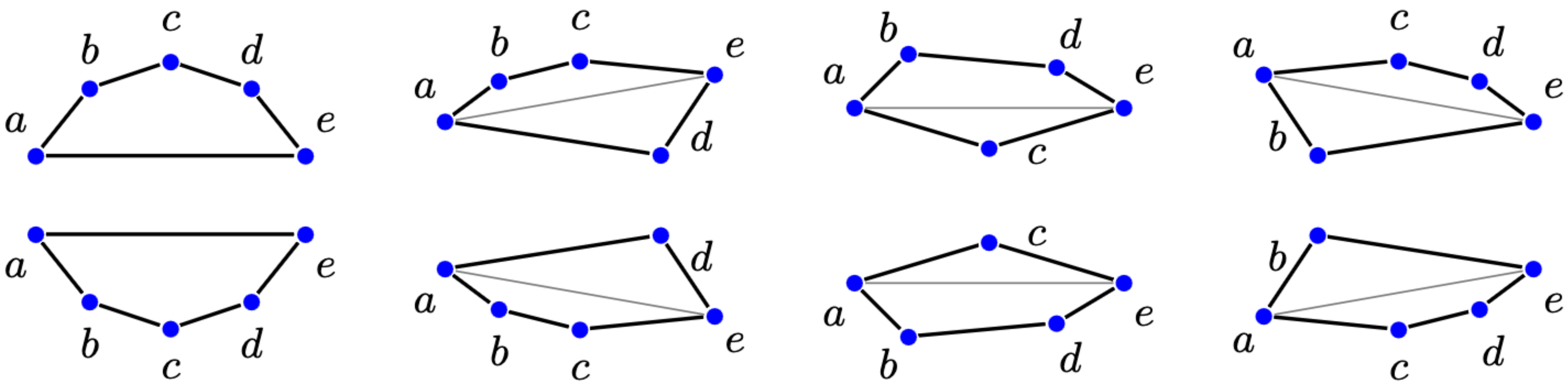


$$\sigma(a, b, c) = \text{true}$$

$$\Longleftrightarrow$$

$$\det \begin{bmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ 1 & 1 & 1 \end{bmatrix} > 0$$

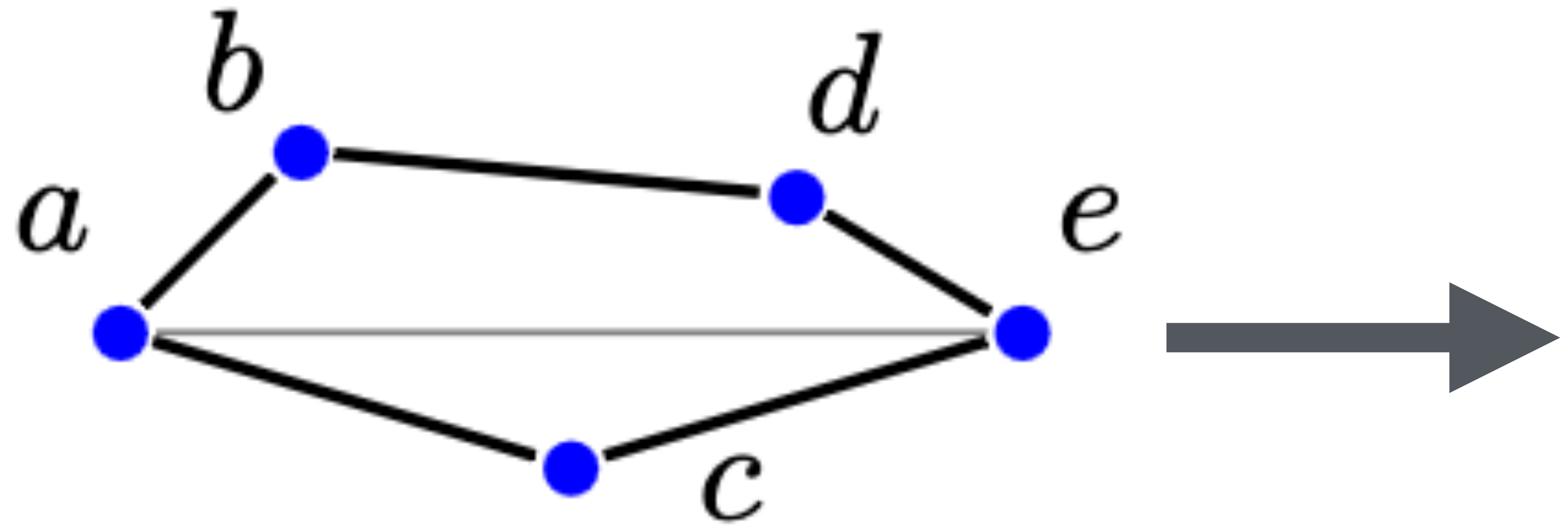
Szekeres and Peters: 8 cases for convex pentagons



I II III IV

Szekeres and Peters: 8 cases for a convex pentagon

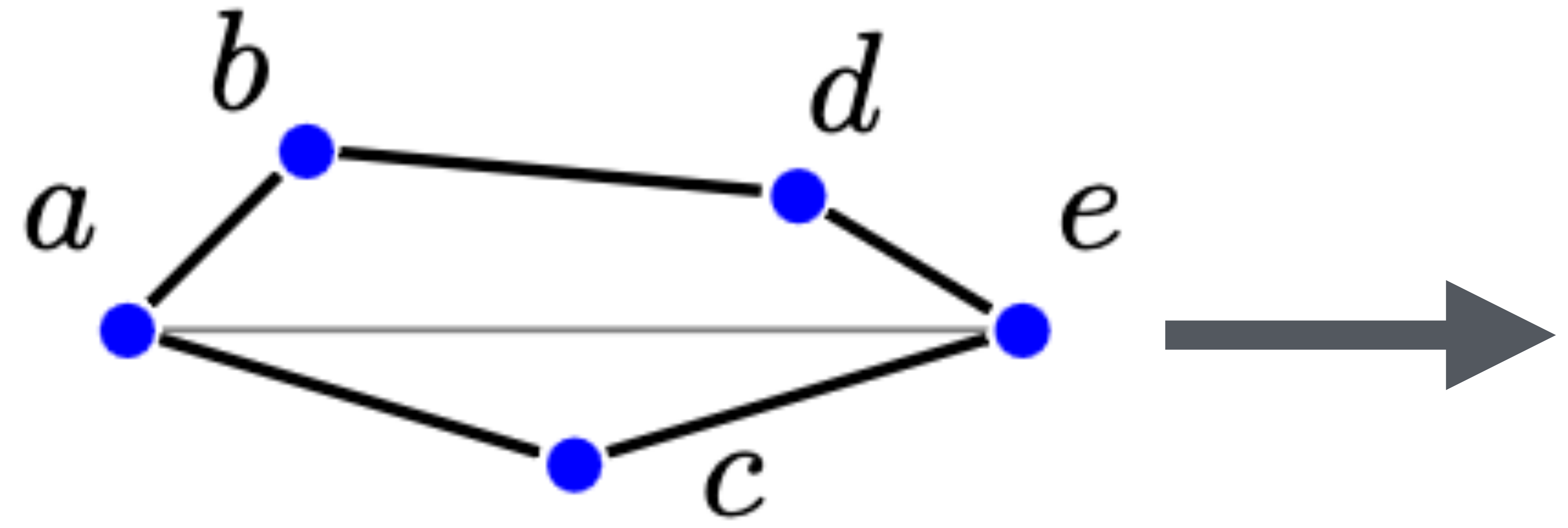
Encoding Example:



$$\neg\sigma(a,b,d) \wedge \neg\sigma(b,d,e) \wedge \sigma(a,c,e)$$

Szekeres and Peters: 8 cases for a convex pentagon

Encoding Example:

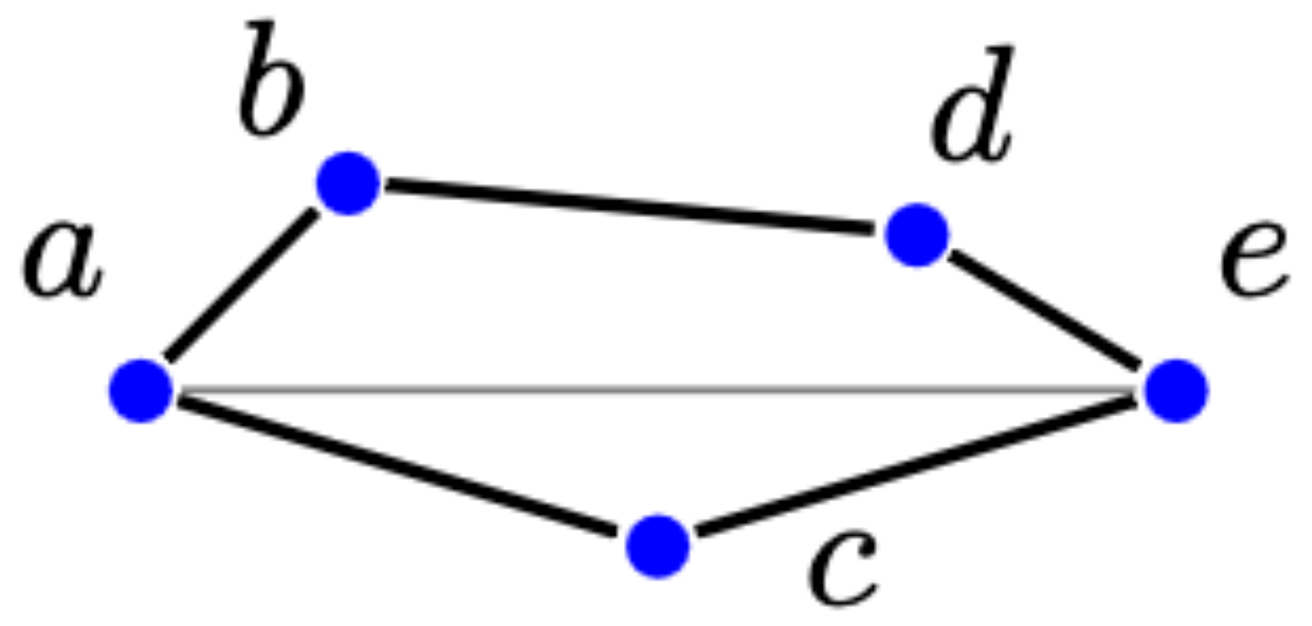


$$\neg\sigma(a, b, d) \wedge \neg\sigma(b, d, e) \wedge \sigma(a, c, e)$$

So to forbid that pentagon in one clause we do

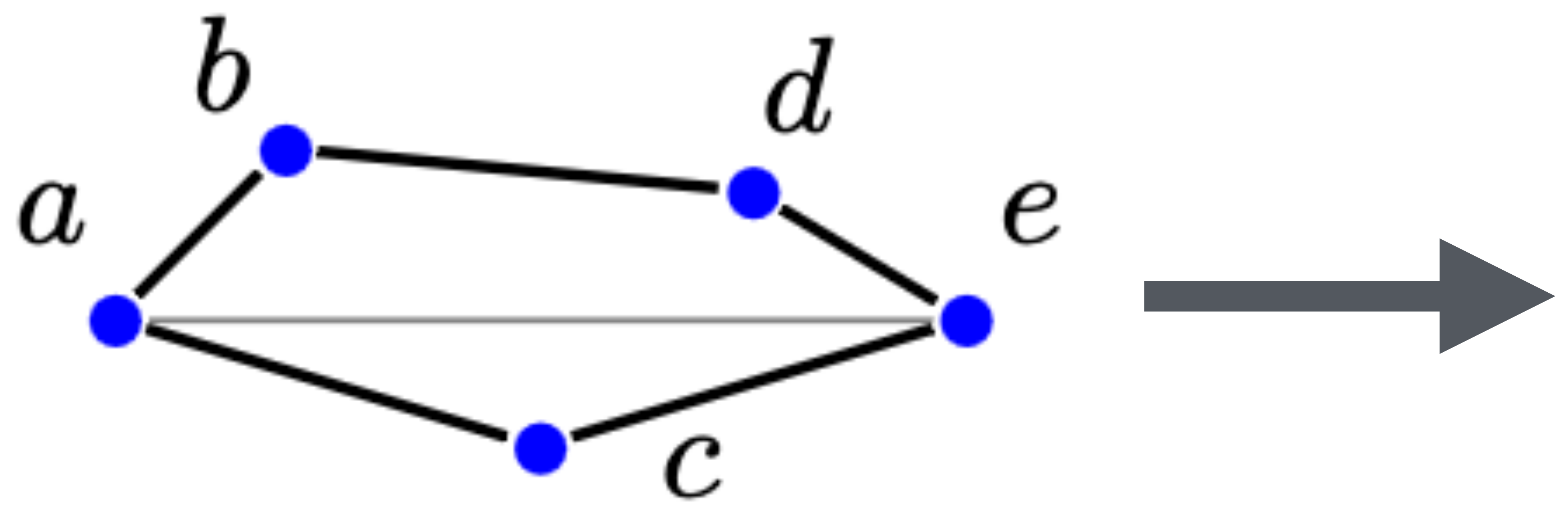
$$\sigma(a, b, d) \vee \sigma(b, d, e) \vee \neg\sigma(a, c, e)$$

We just said:



$$\neg\sigma(a,b,d) \wedge \neg\sigma(b,d,e) \wedge \sigma(a,c,e)$$

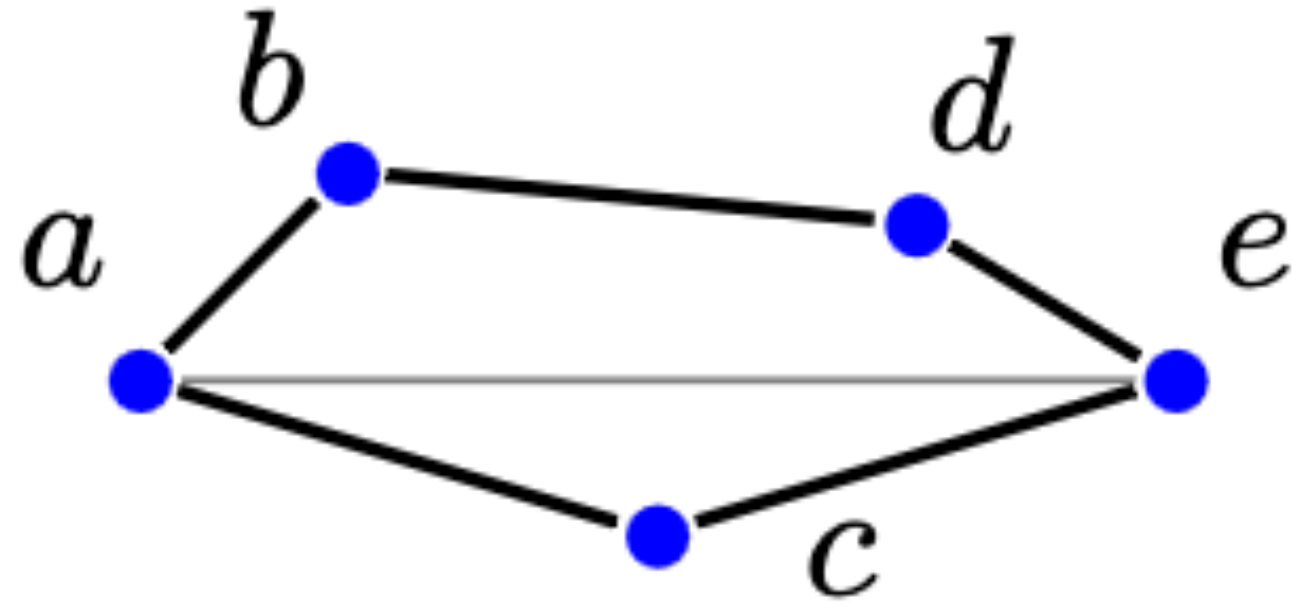
We just said:



$$\neg\sigma(a,b,d) \land \neg\sigma(b,d,e) \land \sigma(a,c,e)$$

Question: is the converse ⟵ also true?

We just said:



$\Longrightarrow$ $\neg\sigma(a,b,d) \wedge \neg\sigma(b,d,e) \wedge \sigma(a,c,e)$

Question: is the converse $\Longleftarrow$ also true?

Not necessarily! We don't know if a $\boldsymbol{\sigma}$-assignment corresponds to an actual pointset !

Stochastic Local Search

SAT algorithm that minimize falsified clauses by flipping variables.

No guarantees!

Stochastic Local Search

SAT algorithm that minimize falsified clauses by flipping variables.

No guarantees!

We add, for each ordered five tuple (a, b, c, d, e),
the 8 clauses that forbid the (mutually exclusive)
different ways those points could form a convex pentagon

Stochastic Local Search

SAT algorithm that minimize falsified clauses by flipping variables.

No guarantees!

We add, for each ordered five tuple (a, b, c, d, e),
the 8 clauses that forbid the (mutually exclusive)
different ways those points could form a convex pentagon

Number of falsified clauses ≈ Number of convex pentagons*

Stochastic Local Search Results

N	Best	Time
9	**1**	0.00 s
10	**2**	0.00 s
11	**7**	0.00 s
12	12	0.00 s
13	27	0.01 s
14	42	0.01 s
15	77	0.01 s
16	112	0.02 s

...

N	Best	Time
23	1254	12 s
24	1584	472 s
25	2079	64 s
26	2574	5269 s
27	3289	1556 s
28	4004	1792 s
29	5005	467 s
30	6007	18244 s

Note*: these are not necessarily optimal*

Notice something odd?

We would like to know:

$$c_5 = \lim_{N\to\infty} \frac{\mu_5(N)}{\binom{N}{5}}$$

Notice something <u>odd</u>?

We would like to know:

$$c_5 = \lim_{N \to \infty} \frac{\mu_5(N)}{\binom{N}{5}}$$

$$\hat{c}_5(N) = \frac{\mu_5(N)}{\binom{N}{5}}$$

Notice something odd?

We would like to know:

$$c_5 = \lim_{N\to\infty} \frac{\mu_5(N)}{\binom{N}{5}}$$

$$\hat{c}_5(N) = \frac{\mu_5(N)}{\binom{N}{5}}$$

N	Best	$\hat{c}_5(N)$
9	1	0.007936
10	2	0.007936
11	7	0.015151
12	12	0.015151
13	27	0.020979
14	42	0.020979
15	77	0.025641
16	112	0.025641
17	182	0.029411
18	252	0.029411

Notice something odd?

We would like to know:

$$c_5 = \lim_{N \to \infty} \frac{\mu_5(N)}{\binom{N}{5}}$$

$$\hat{c}_5(N) = \frac{\mu_5(N)}{\binom{N}{5}}$$

N	Best	$\hat{c}_5(N)$
9	1	0.007936
10	2	0.007936
11	7	0.015151
12	12	0.015151
13	27	0.020979
14	42	0.020979
15	77	0.025641
16	112	0.025641
17	182	0.029411
18	252	0.029411

SAT Inspired Math Idea!

Plug SAT results into Mathematica!

```
In[67]:= (*Define the points*)
points = {{10, 2}, {12, 12}, {14, 42}, {16, 112}, {18, 252}, {20, 504}, {22, 924},
   {24, 1584}};

(*Compute the interpolating polynomial of degree 5*)
poly = InterpolatingPolynomial[points, n];

(*Simplify the polynomial to make it easier to read*)
simplifiedPoly = Simplify[poly];

(*Display the simplified polynomial*)
simplifiedPoly
```

Out[70]= $\frac{n\left(384-400 n+140 n^{2}-20 n^{3}+n^{4}\right)}{1920}$

In[71]:= FullSimplify$\left[\frac{n\left(384-400 n+140 n^{2}-20 n^{3}+n^{4}\right)}{1920}\right]$

Out[71]= $\frac{(-8+n) \times(-6+n) \times(-4+n) \times(-2+n)\, n}{1920}$

Plug SAT results into Mathematica!

Out[71]= $\frac{(-8+n)\times(-6+n)\times(-4+n)\times(-2+n)\,n}{1920}$

In[72]:= `Binomial[n, 5]`

Out[72]= $\frac{1}{120}\times(-4+n)\times(-3+n)\times(-2+n)\times(-1+n)\,n$

Plug SAT results into Mathematica!

Out[71]= $\frac{(-8+n)\times(-6+n)\times(-4+n)\times(-2+n)\,n}{1920}$

In[72]:= `Binomial[n, 5]`

Out[72]= $\frac{1}{120}\times(-4+n)\times(-3+n)\times(-2+n)\times(-1+n)\,n$

$$\binom{n}{5} = \frac{n\cdot(n-1)\cdot(n-2)\cdot(n-3)\cdot(n-4)}{120}$$
$$= \frac{2n\cdot(2n-2)\cdot(2n-4)\cdot(2n-6)\cdot(2n-8)}{120\cdot 32}$$
$$= \frac{2n\cdot(2n-2)\cdot(2n-4)\cdot(2n-6)\cdot(2n-8)}{2\cdot 1920} = \frac{\widehat{\mu(2n)}}{2}$$

Plug SAT results into Mathematica!

Out[71]= $\frac{(-8+n)\times(-6+n)\times(-4+n)\times(-2+n)\,n}{1920}$

In[72]:= `Binomial[n, 5]`

Out[72]= $\frac{1}{120}\times(-4+n)\times(-3+n)\times(-2+n)\times(-1+$ …

SAT Inspired Math Idea!

$$\binom{n}{5} = \frac{n\cdot(n-1)\cdot(n-2)\cdot(n-3)\cdot(n-4)}{120}$$
$$= \frac{2n\cdot(2n-2)\cdot(2n-4)\cdot(2n-6)\cdot(2n-8)}{120\cdot 32}$$
$$= \frac{2n\cdot(2n-2)\cdot(2n-4)\cdot(2n-6)\cdot(2n-8)}{2\cdot 1920} = \frac{\widehat{\mu(2n)}}{2}$$

How do these solutions look like?

We need to solve the ***Realizability*** problem ($\exists\mathbb{R}$ complete):
Given a set of orientations, can they all be *realized* by a configuration of points in the plane

(Only available at the in-person talk, sorry)

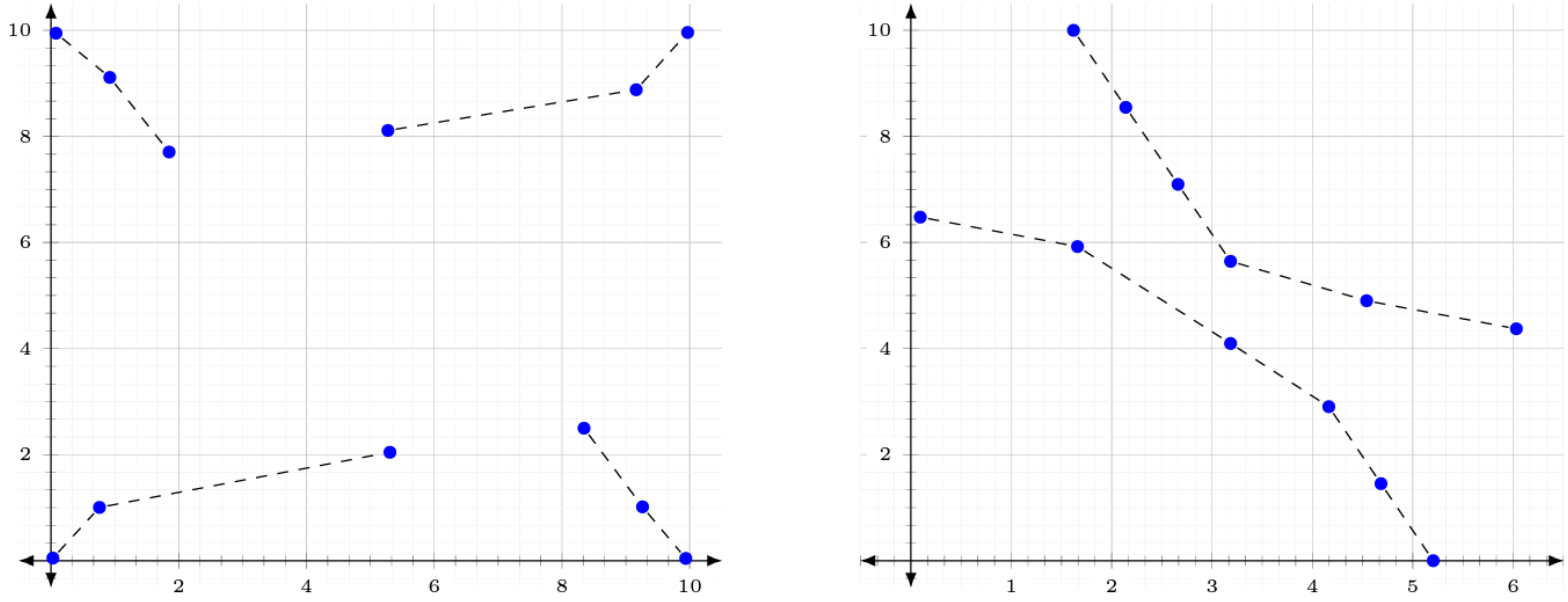
How do these solutions look like?

We need to solve the ***Realizability*** problem ($\exists\mathbb{R}$ complete):
Given a set of orientations, can they all be *realized* by a configuration of points in the plane

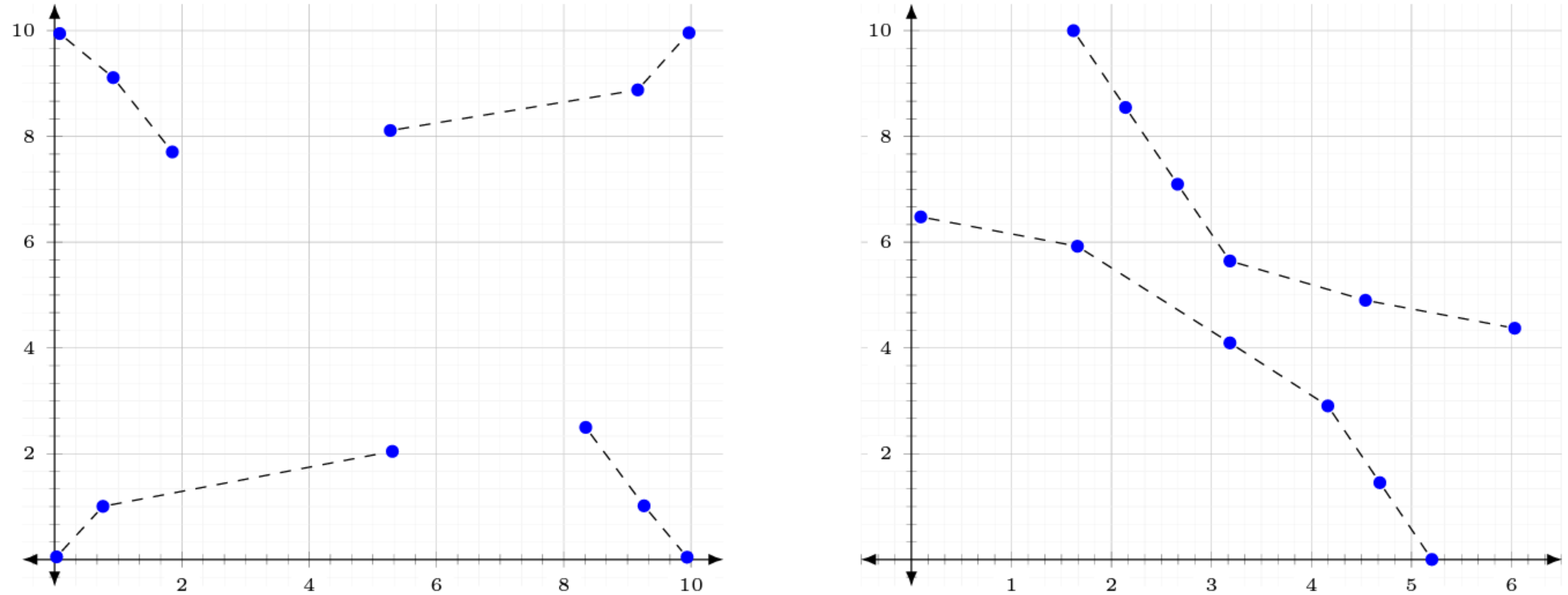
(Only available at the in-person talk, sorry)

Demo

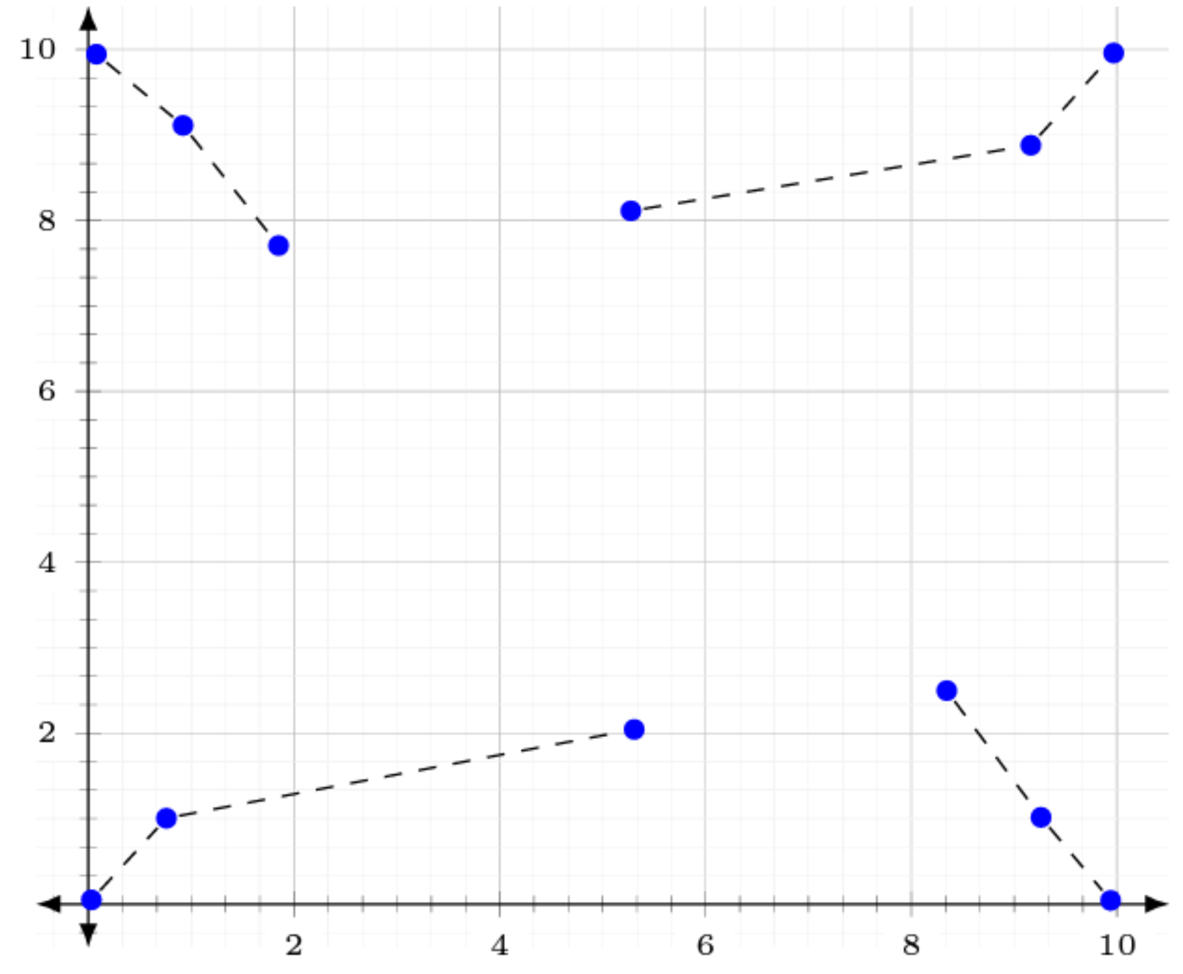
How do these solutions look like?

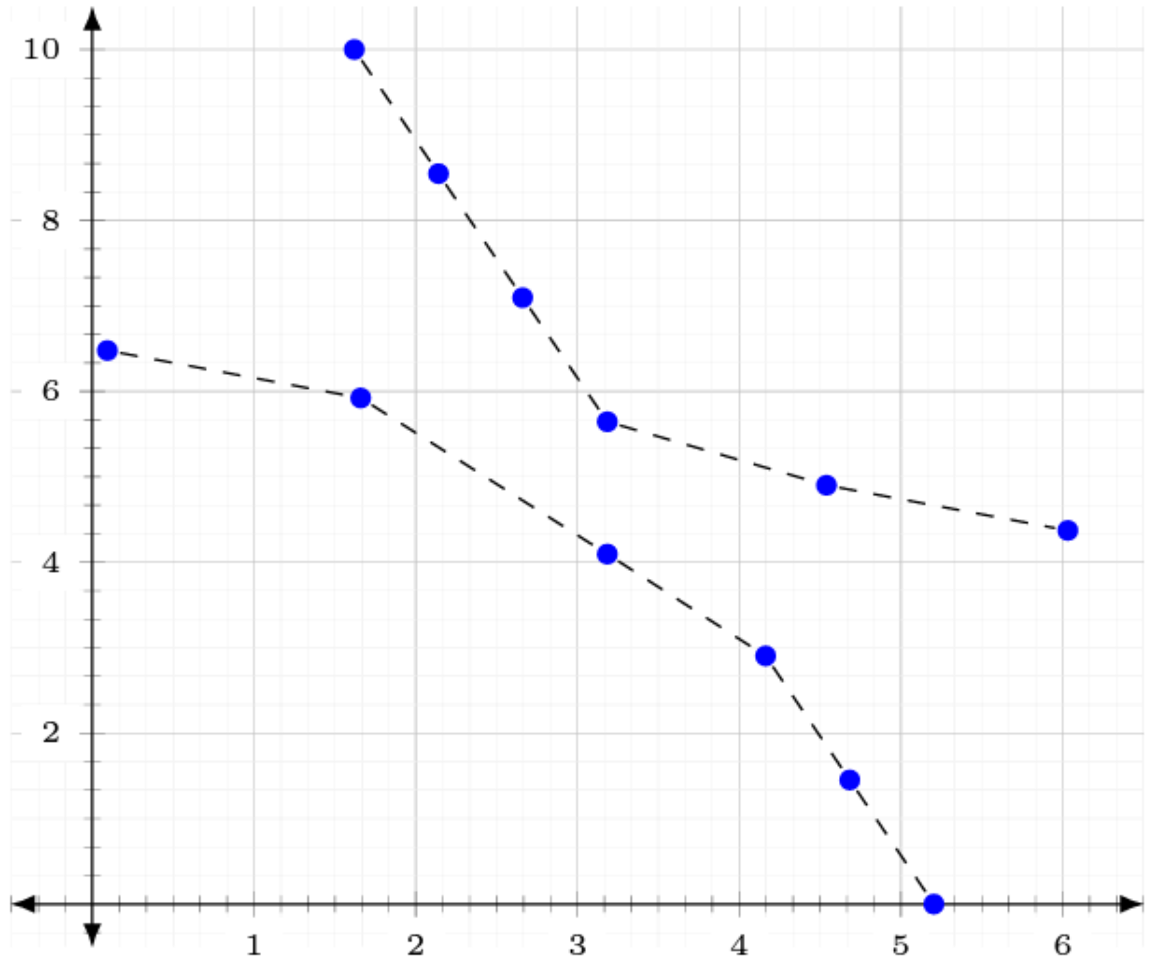


Generalize them as constructions!

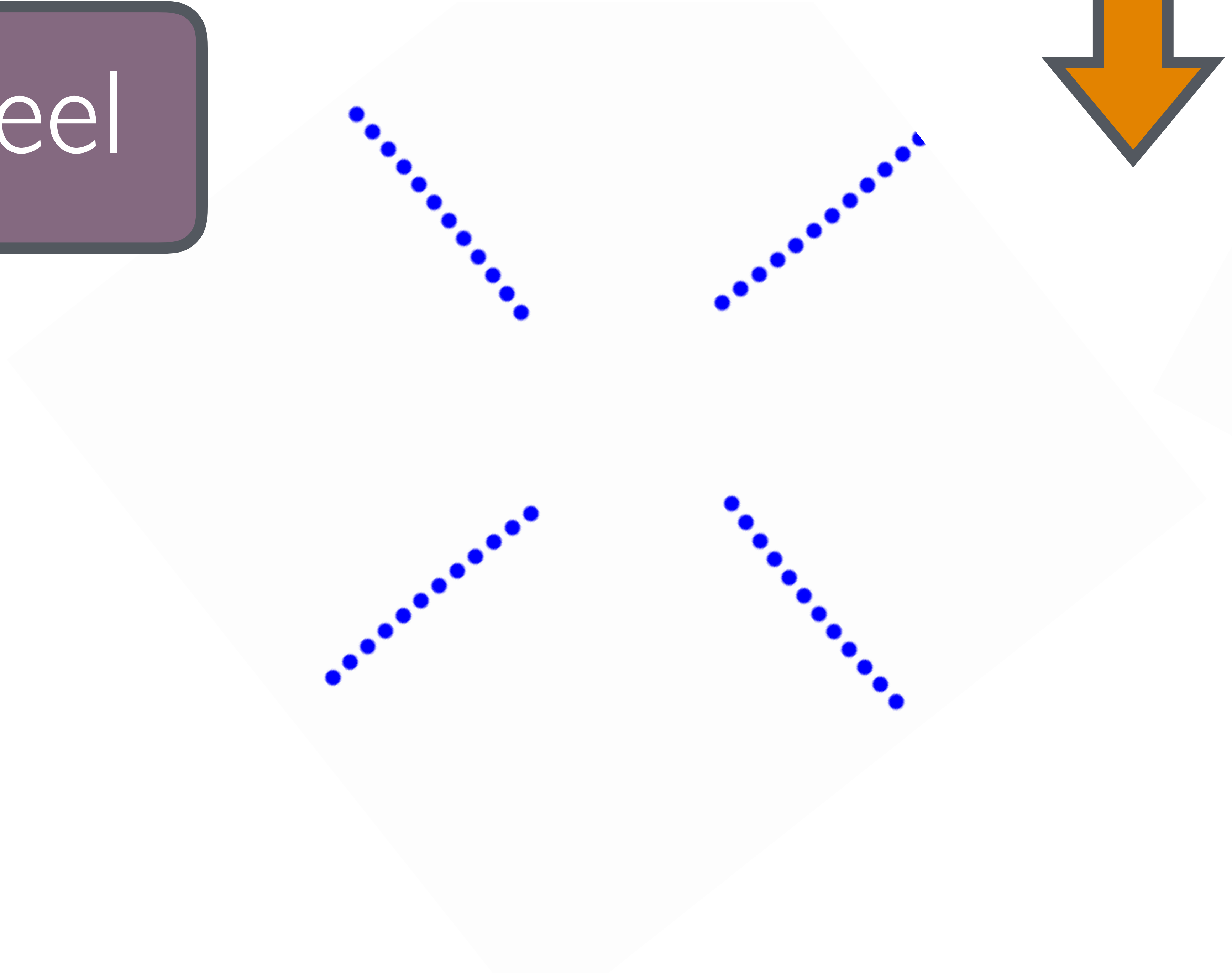


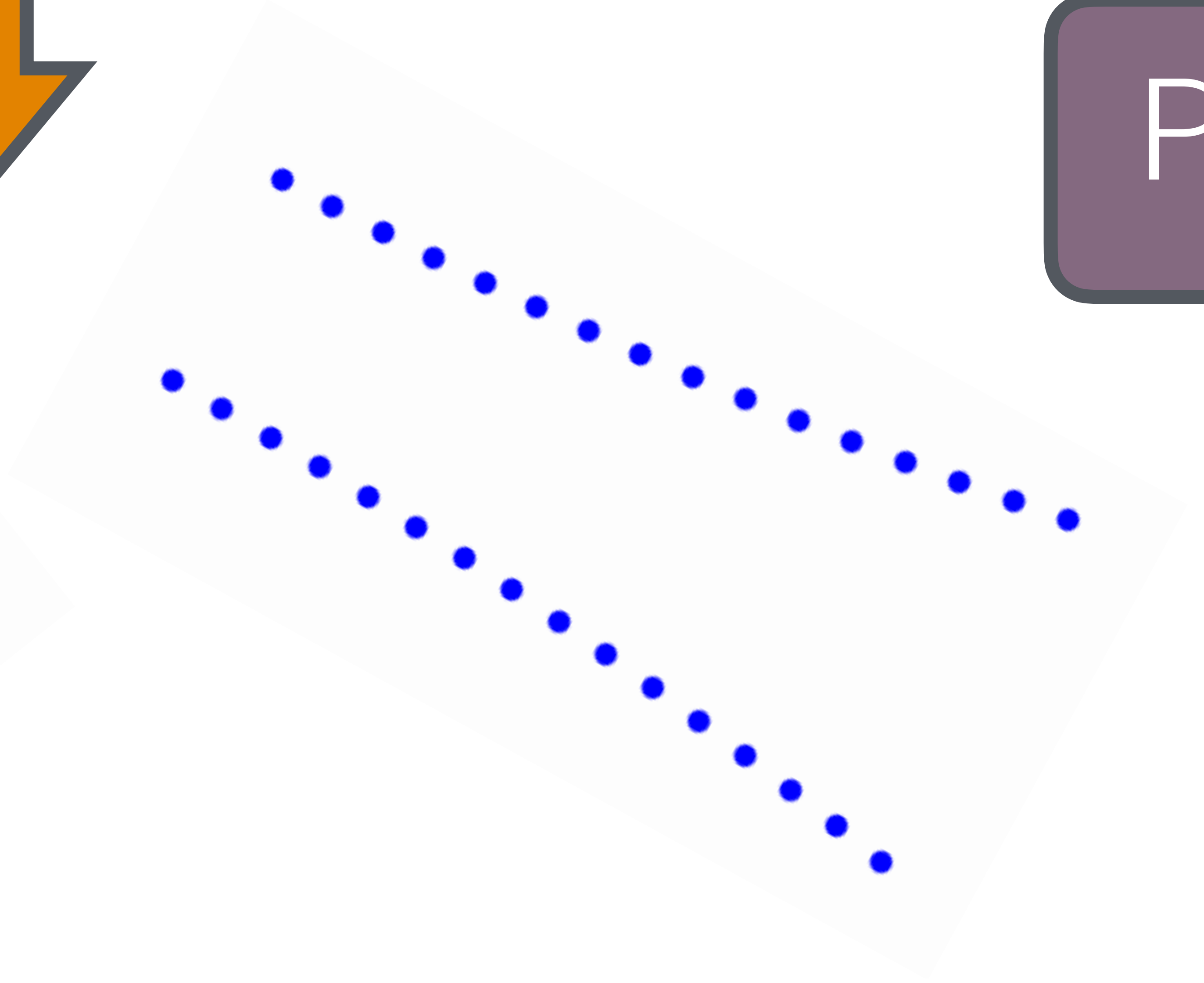
Generalize them as constructions!





Pinwheel





Parabolic

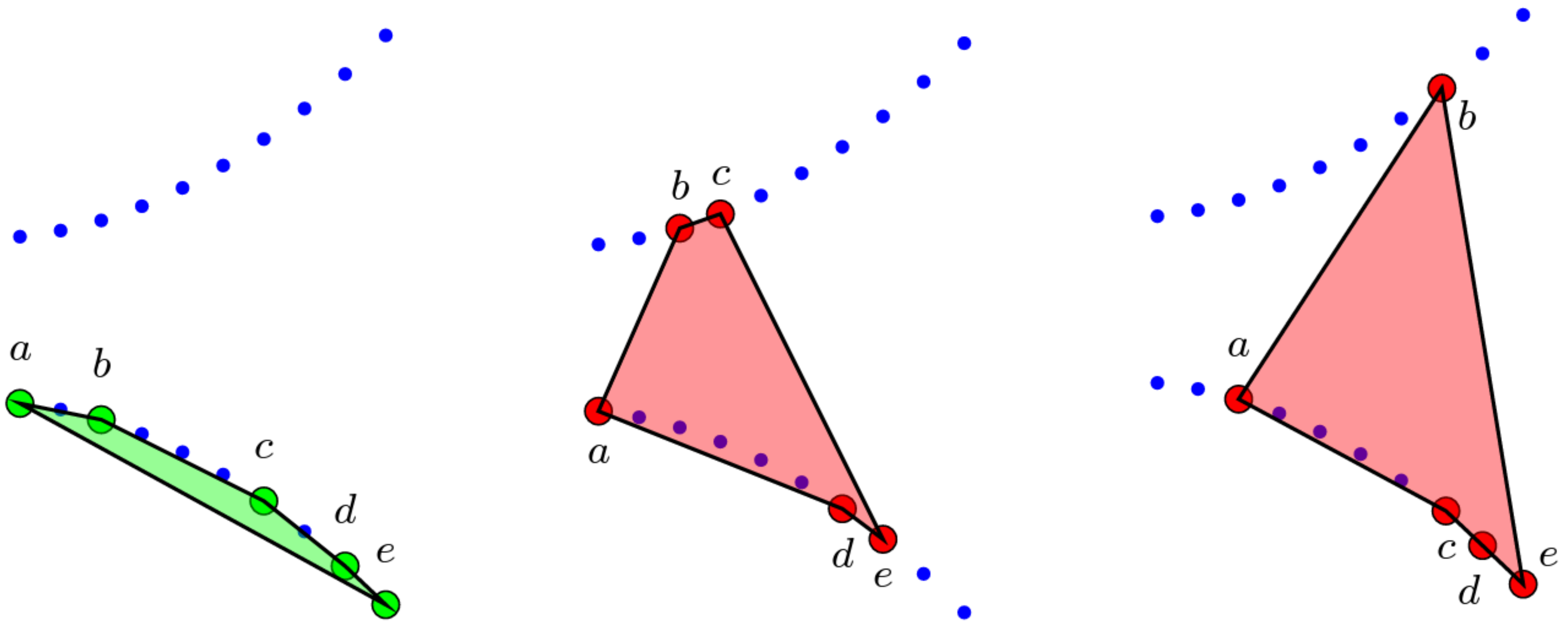
General Upper Bound

Theorem $\mu_5(N) \leq \binom{\lfloor n/2 \rfloor}{5} + \binom{\lceil n/2 \rceil}{5}$

General Upper Bound

Theorem

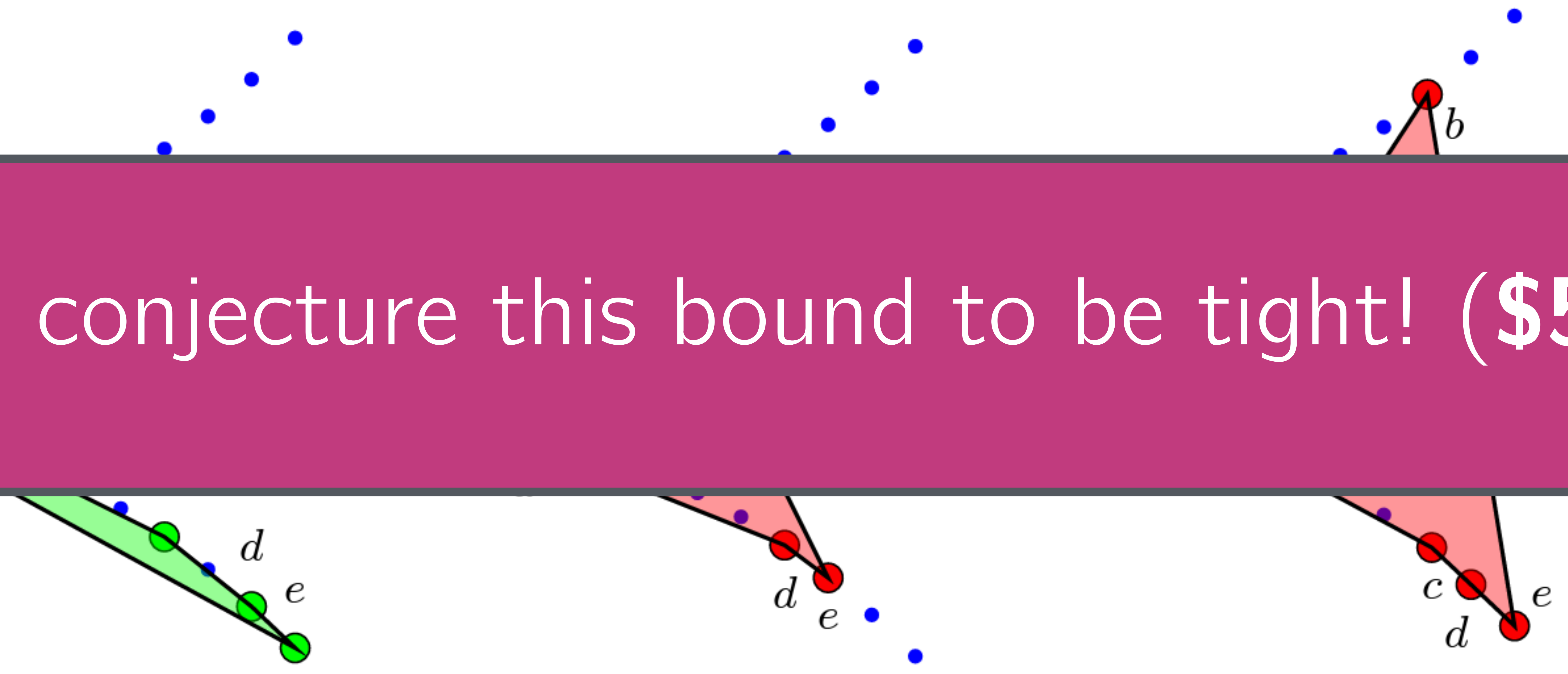
$$\mu_5(N) \leq \binom{\lfloor n/2 \rfloor}{5} + \binom{\lceil n/2 \rceil}{5}$$



General Upper Bound

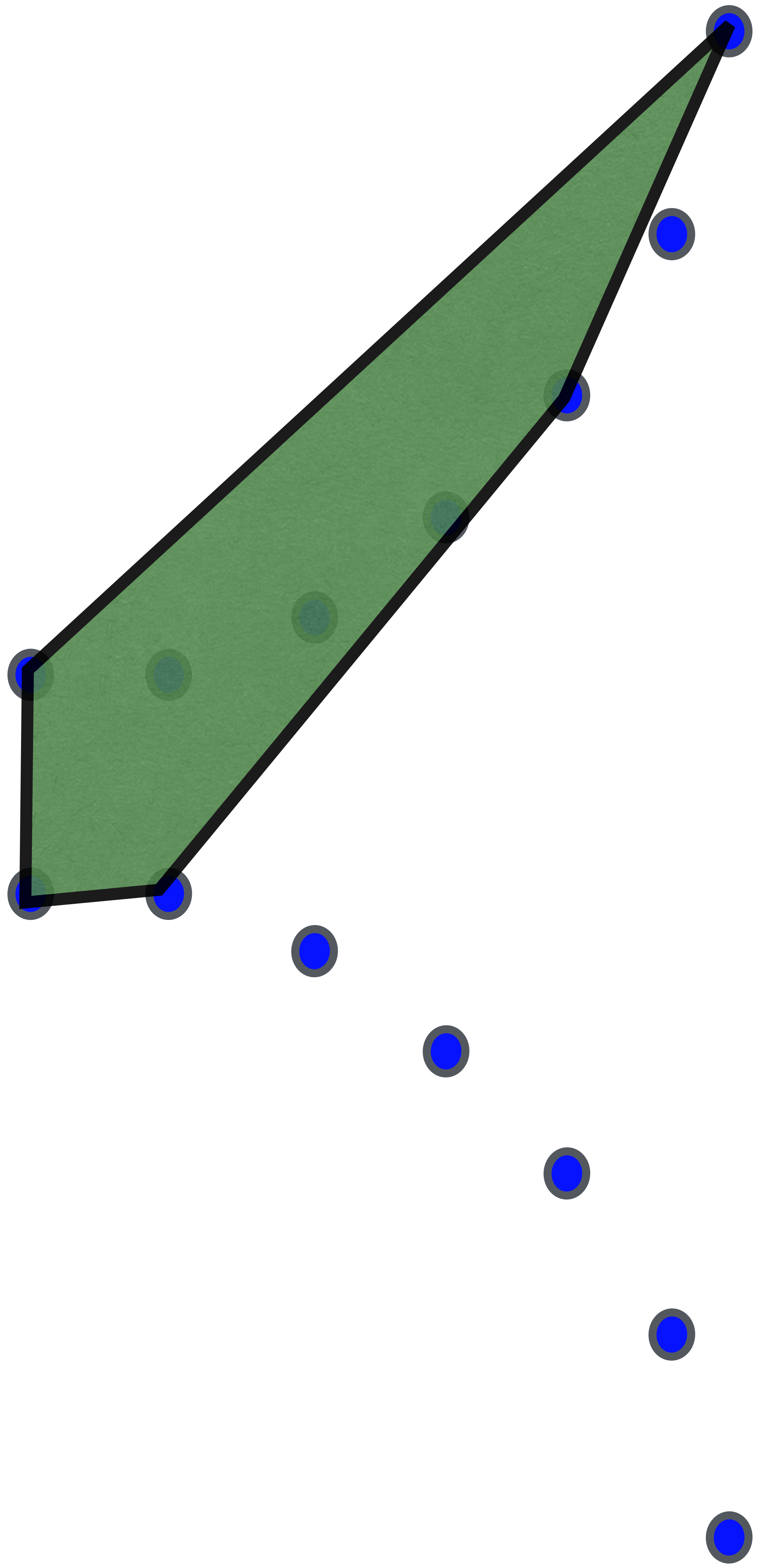
Theorem

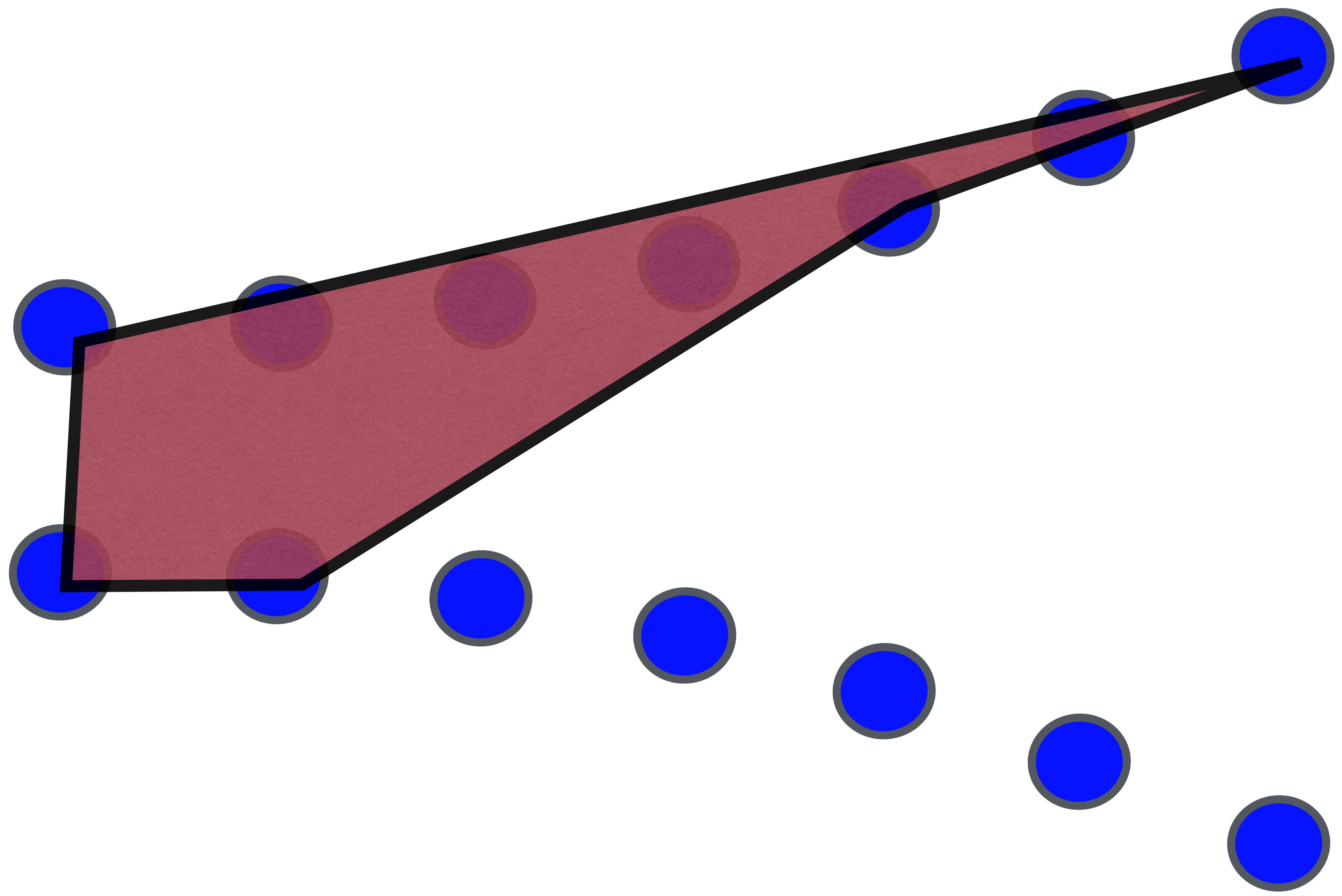
$$\mu_5(N) \leq \binom{\lfloor n/2 \rfloor}{5} + \binom{\lceil n/2 \rceil}{5}$$



We conjecture this bound to be tight! (**$500**)

We need to be careful though!





$$p_i^\top = \left(i, 2+\frac{i^2}{n^2}\right), \forall i \in \left[\left[\left\lfloor \frac{n}{2} \right\rfloor\right]\right] \quad \text{and} \quad p_i^\perp = \left(i, -2-\frac{i^2}{n^2}\right), \forall i \in \left[\left[\left\lceil \frac{n}{2} \right\rceil\right]\right]$$

Remember something odd?

N	Best	
9	1	0.007936
10	2	0.007936
11	7	0.015151
12	12	0.015151
13	27	0.020979
14	42	0.020979
15	77	0.025641
16	112	0.025641
17	182	0.029411
18	252	0.029411

Remember something odd?

N	Best	
9	1	0.007936
10	2	0.007936
11	7	0.015151
12	12	0.015151
13	27	0.020979
14	42	0.020979
15	77	0.025641
16	112	0.025641
17	182	0.029411
18	252	0.029411

SAT Inspired Math Idea!

Remember something odd?

N	Best	
9	1	0.007936
10	2	0.007936
11	7	0.015151
12	12	0.015151
13	27	0.020979
14	42	0.020979
15	77	0.025641
16	112	0.025641
17	182	0.029411
18	252	0.029411

SAT Inspired Math Idea!

Theorem: Odd-Even

If the conjecture holds for $2N + 1$,

Then it must hold for $2N + 2$

MaxSAT Verification

A **major benefit of SAT** (or its optimization variant MaxSAT) is the community emphasis on **verification**

MaxSAT Verification

A **major benefit of SAT** (or its optimization variant MaxSAT) is the community emphasis on **verification**

We generate independently-checkable proofs for the optimal bounds up to **N = 15**

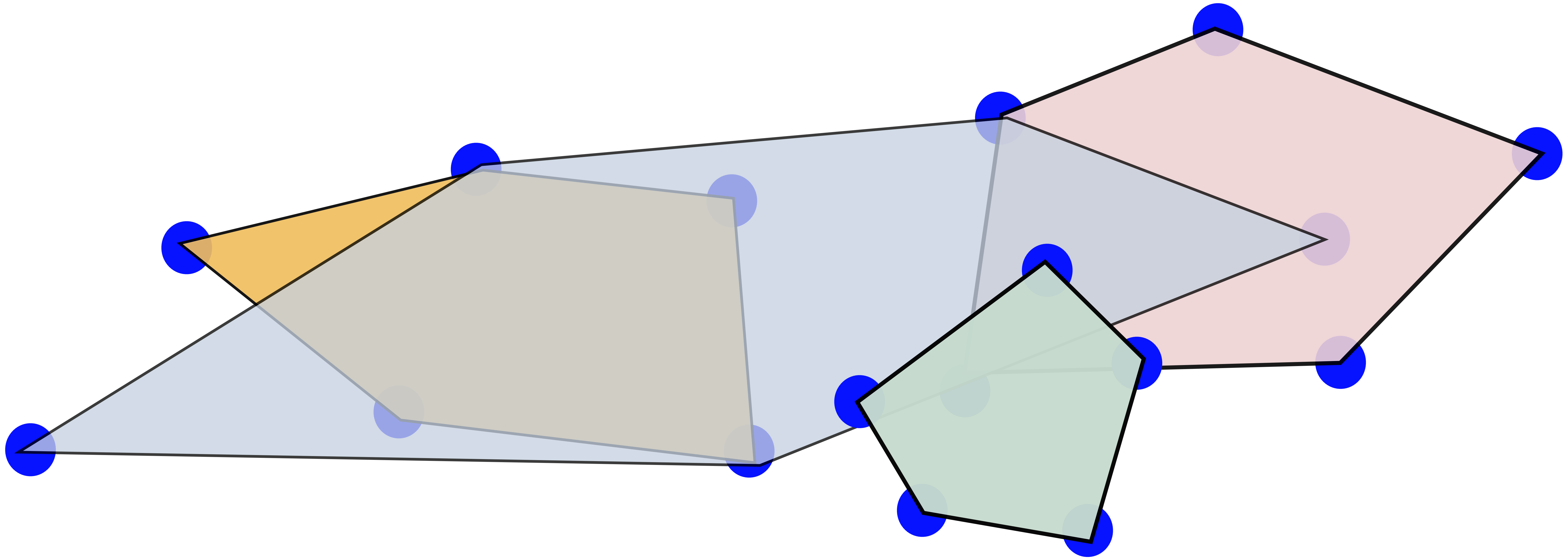
MaxSAT Verification

A **major benefit of SAT** (or its optimization variant MaxSAT) is the community emphasis on **verification**

We generate independently-checkable proofs for the optimal bounds up to **N = 15**

Therefore we verify the conjecture up to **N = 16** :)

Automated
Mathematical Discovery and Verification



Minimizing Pentagons in the Plane

Bernardo Subercaseaux, John Mackey, Marijn Heule, and Ruben Martins

Carnegie Mellon University