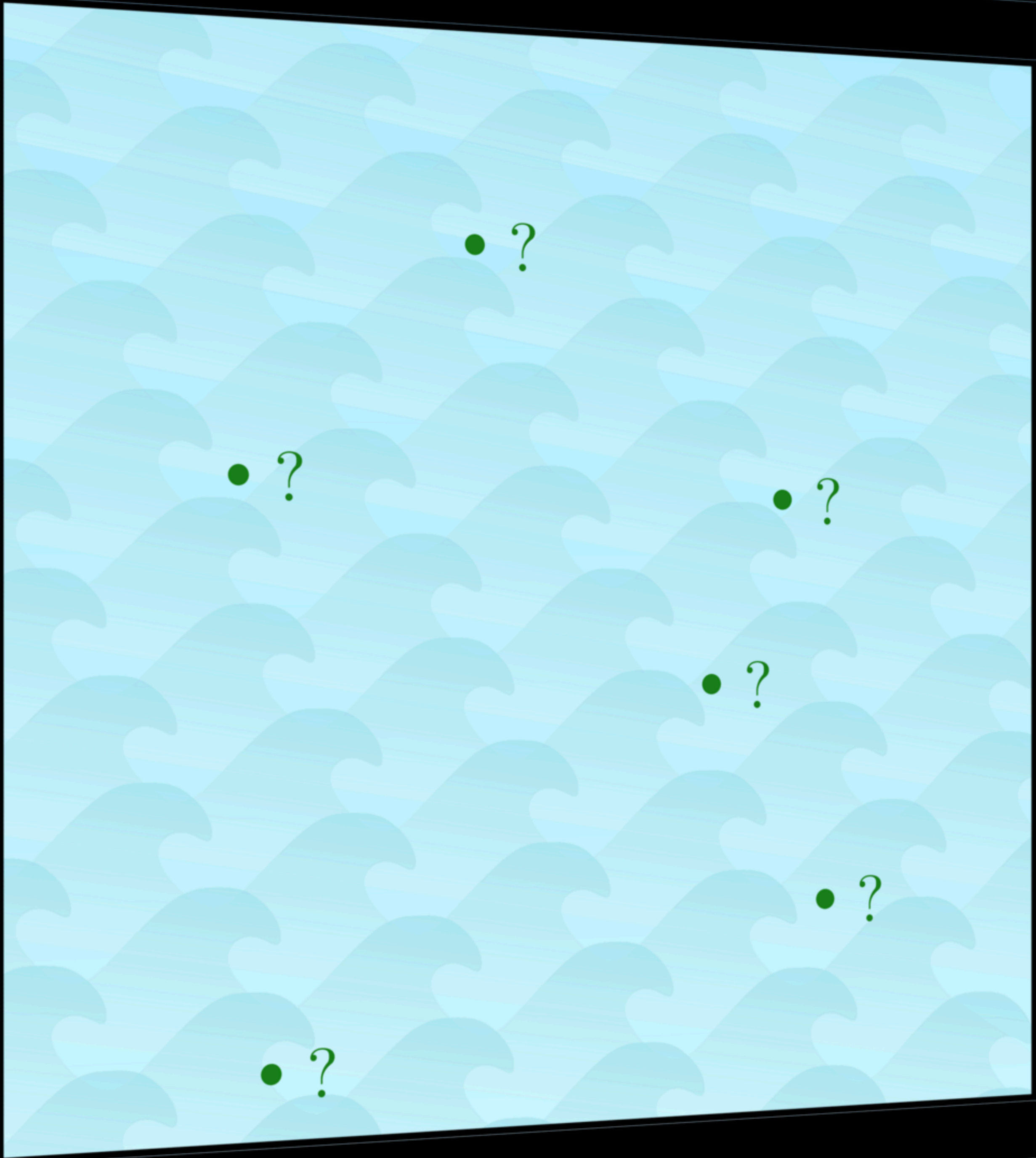
Geometry,
Uncertainty,
&
BATTLESHIPS!!!

Eva Hainzl
Maarten Löffler
Daniel Perz
Josef Tkadlec
Markus Wallinger

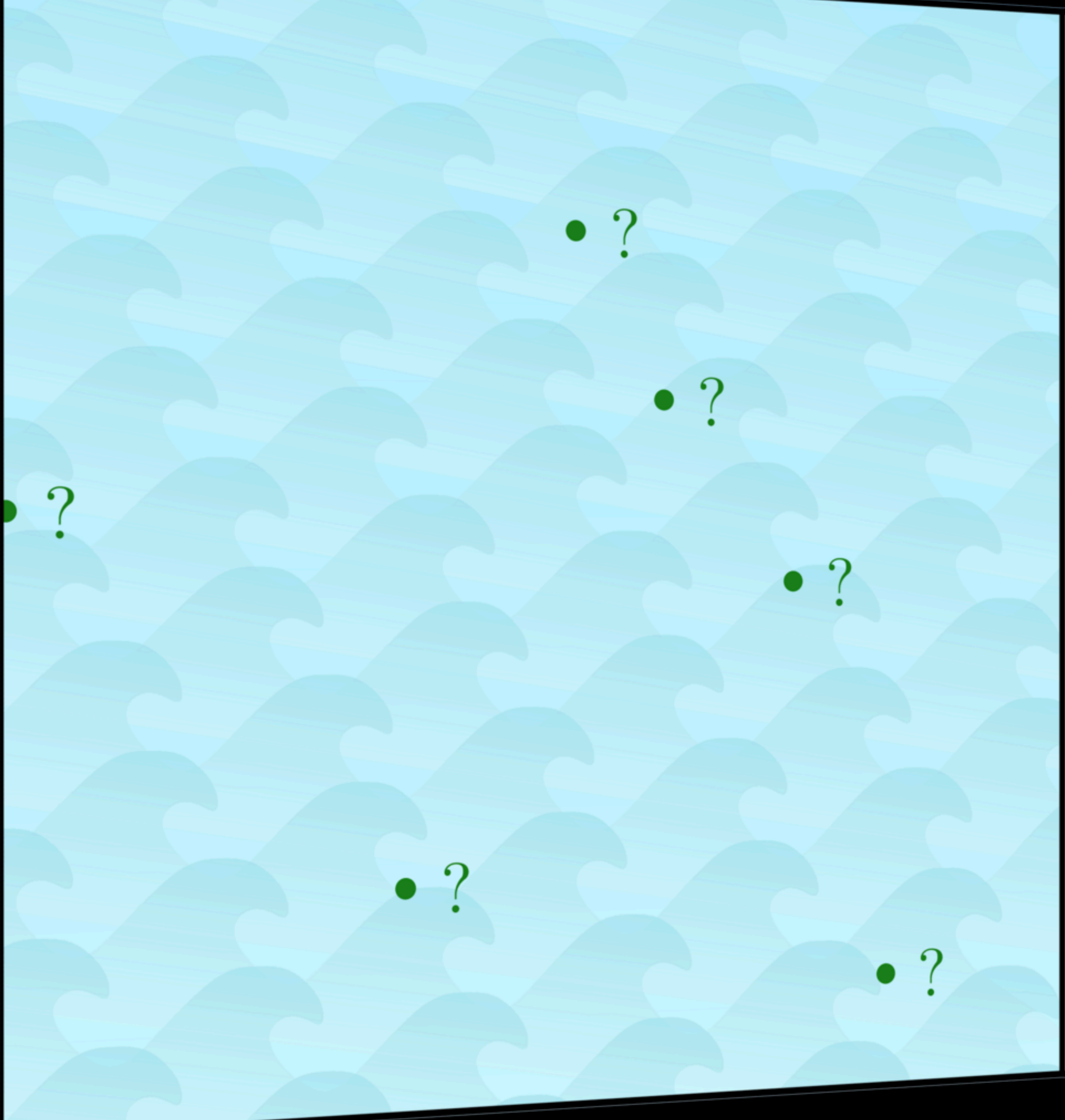
CHAPTER 1

UNCERTAINTY

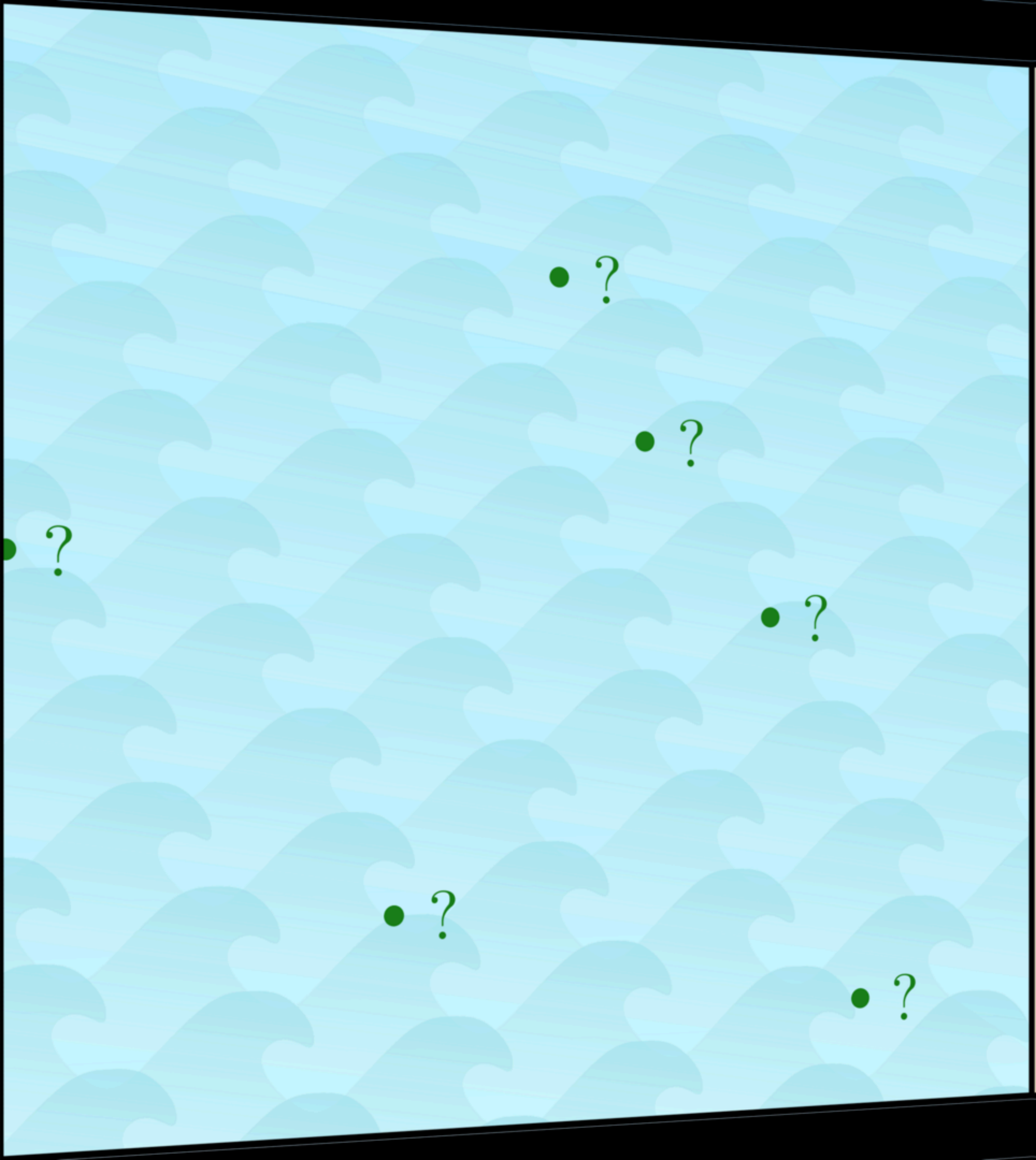
Suppose we have a set of points, but we don’t know where they are.



Suppose we have a set of points, but we don't know where they are.



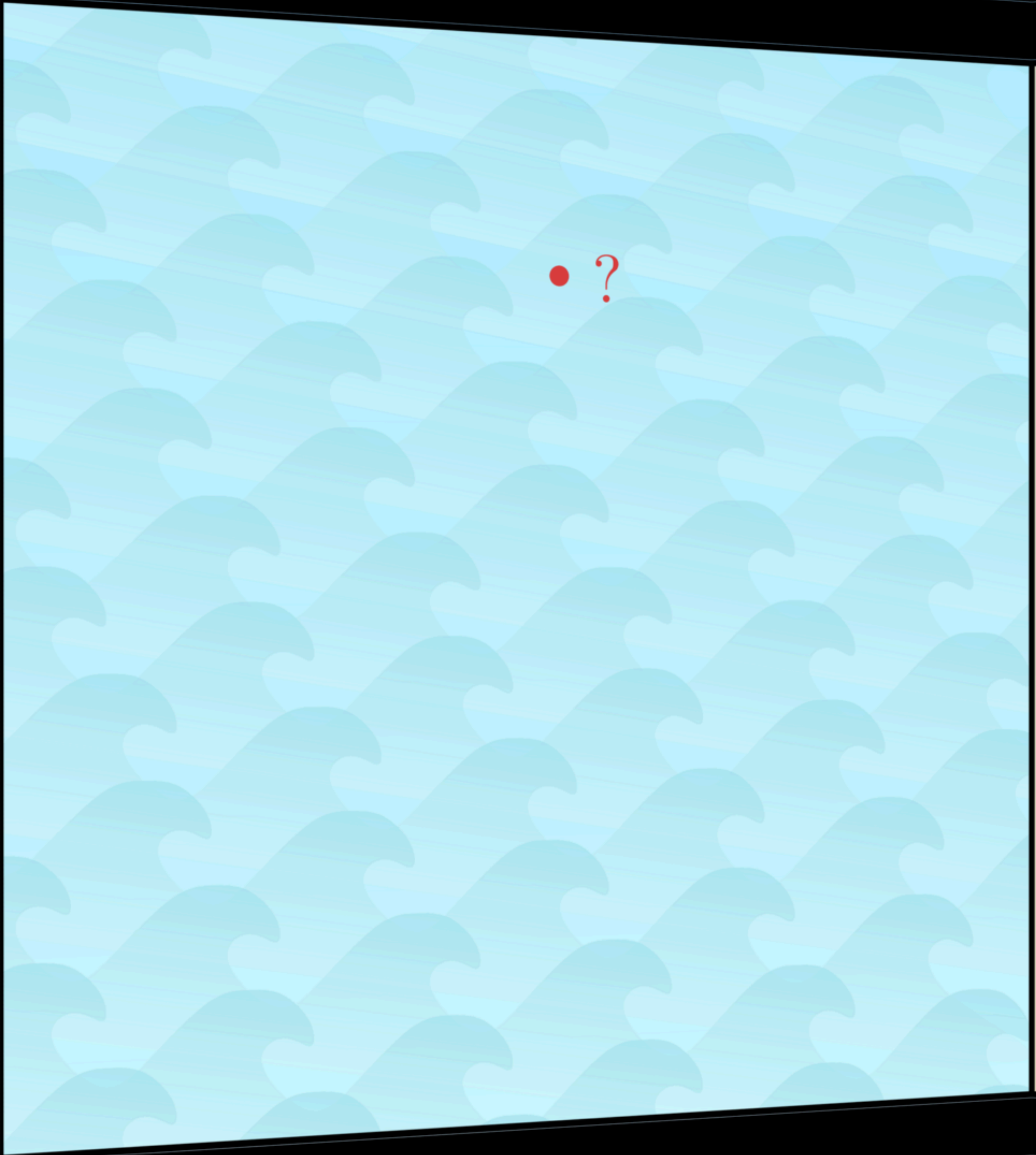
Suppose we have a set of points, but we don't know where they are.



Suppose we have a set of points, but we don't know where they are.

Half an hour ago, I asked:

"*Where* are you"?



Suppose we have a set of points, but we don’t know where they are.

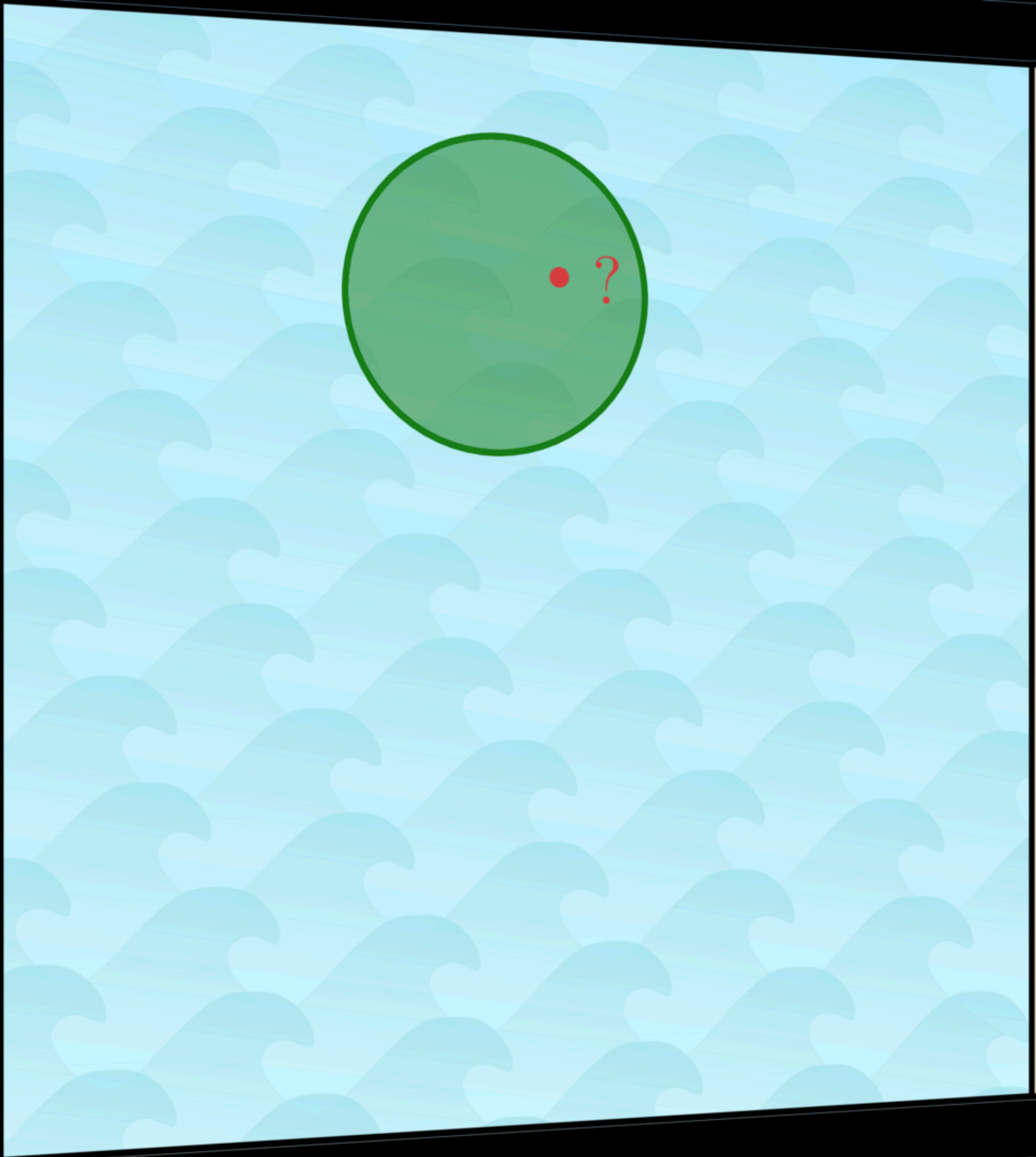
Half an hour ago, I asked:

“*Where* are you”?

Suppose we have a set of points, but we don't know where they are.

Half an hour ago, I asked:

"*Where* are you"?



Suppose we have a set of points, but we don't know where they are.

Half an hour ago, I asked:

"*Where* are you"?

ID → DISK

Suppose we have a set of points, but we don't know where they are.

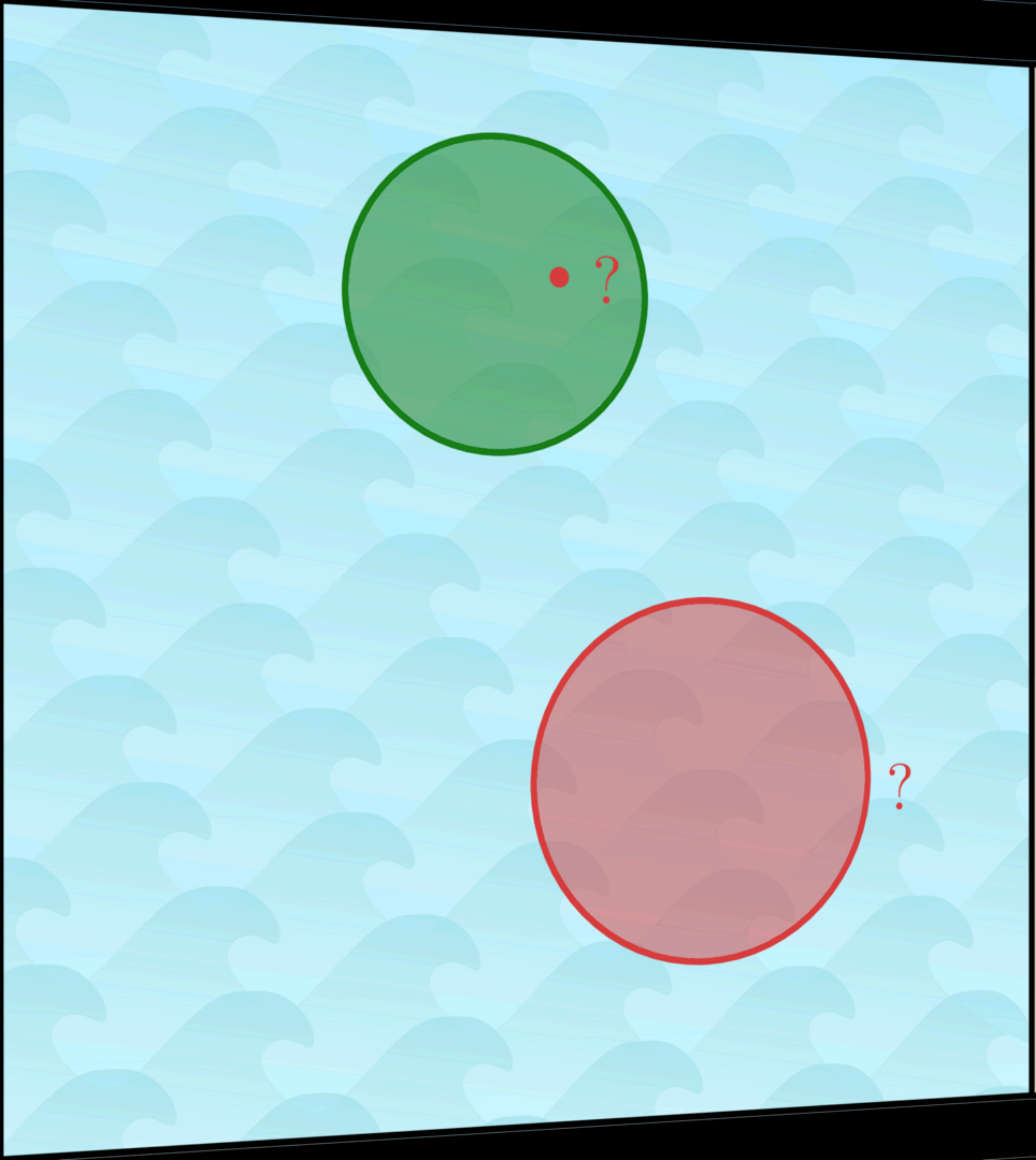
Half an hour ago, I asked:

"*Where* are you"?

ID → DISK

Maybe we can also ask:

"There are *who*"?



Suppose we have a set of points, but we don't know where they are.

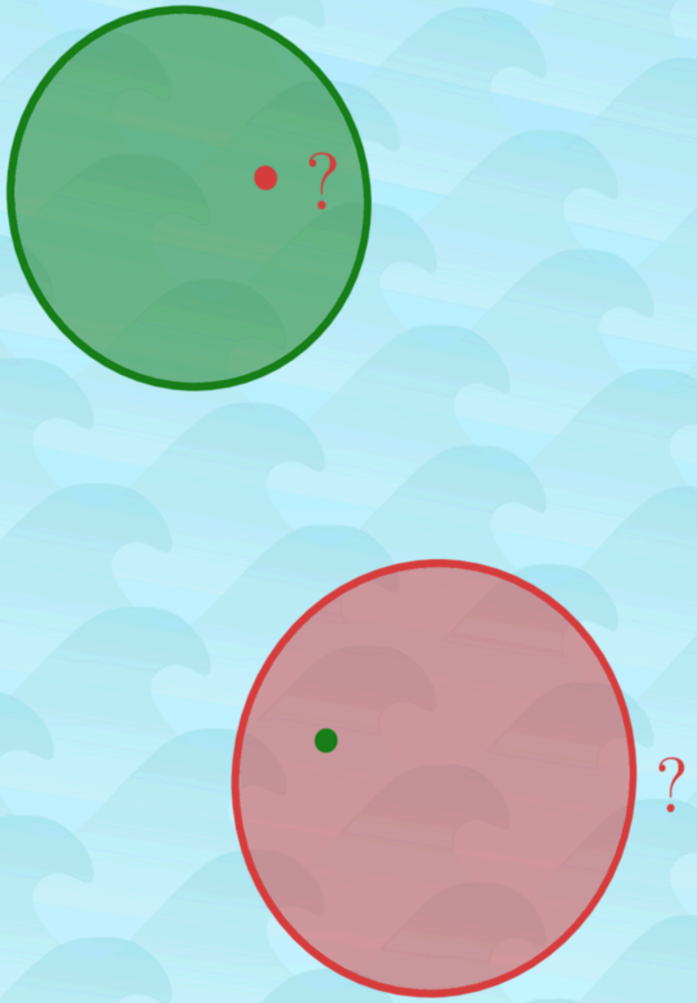
Half an hour ago, I asked:

"*Where* are you"?

ID → DISK

Maybe we can also ask:

"There are *who*"?



Suppose we have a set of points, but we don't know where they are.

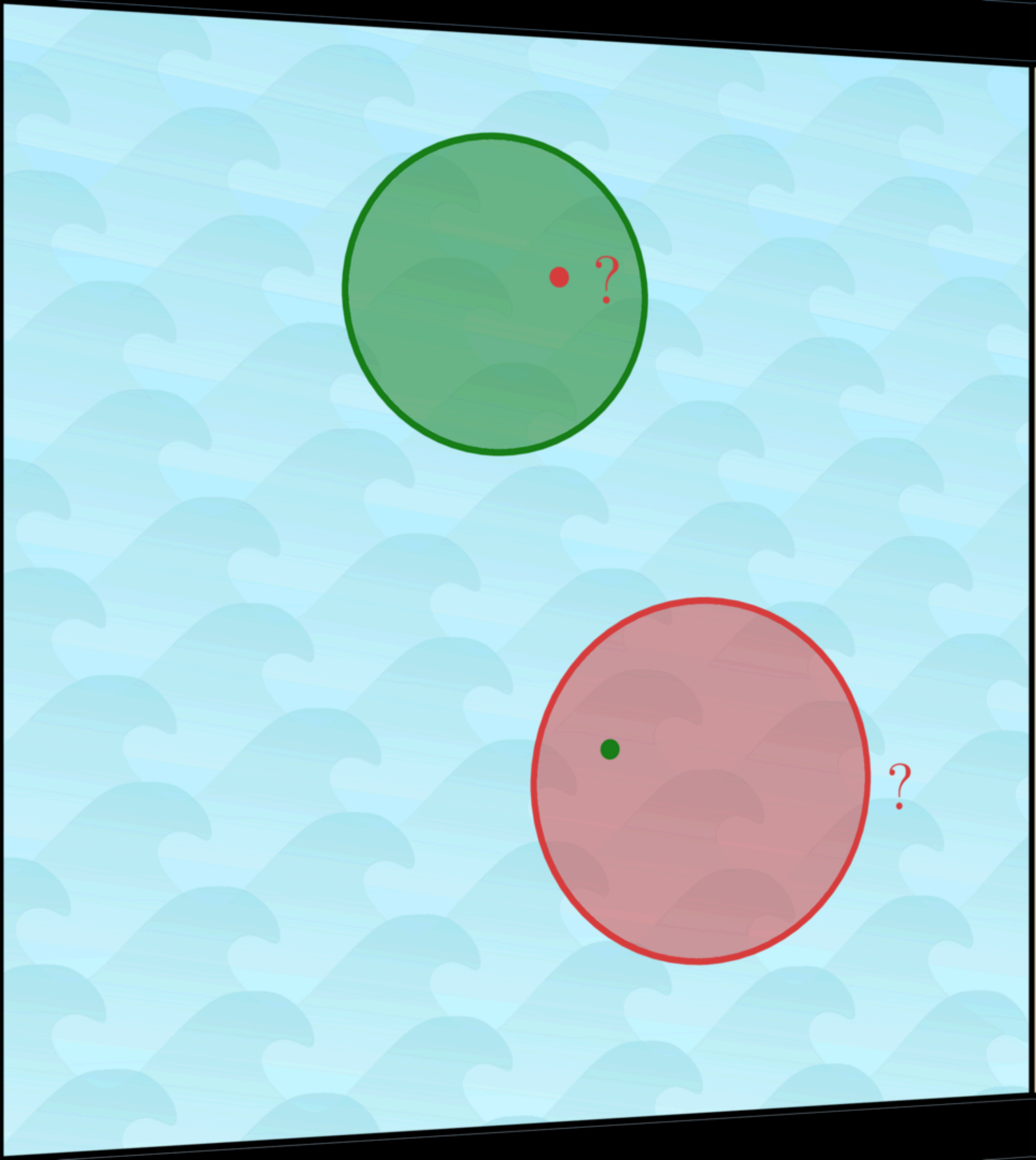
Half an hour ago, I asked:

"*Where* are you"?

ID → DISK

Maybe we can also ask:

"There are *who*"?



Suppose we have a set of points, but we don't know where they are.

Half an hour ago, I asked:

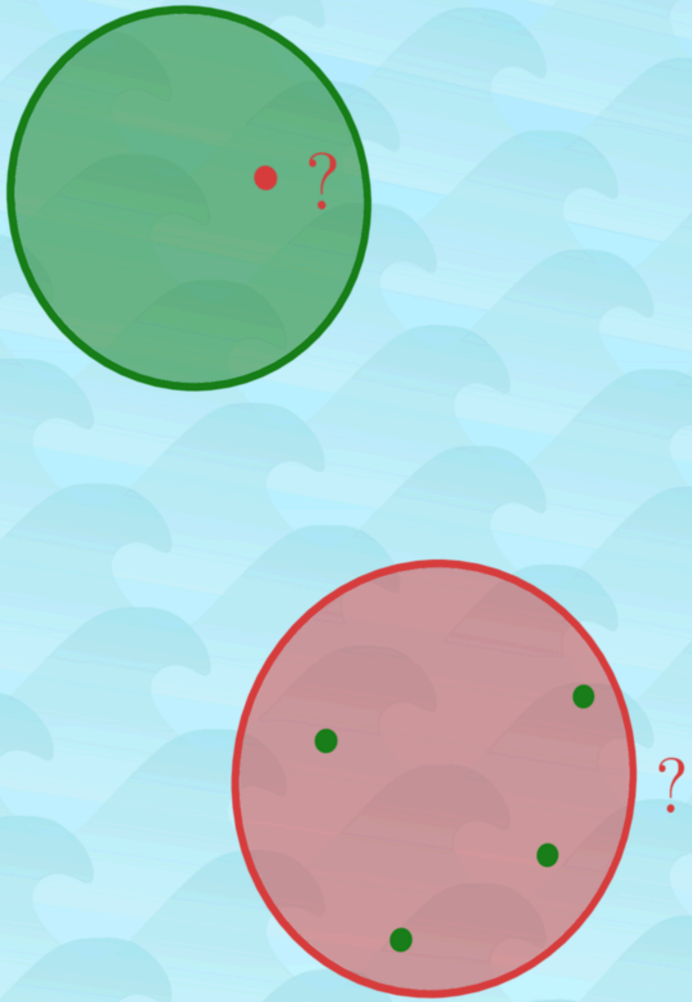
"*Where* are you"?

ID → DISK

Maybe we can also ask:

"There are *who*"?

DISK → [ID]



Suppose we have a set of points, but we don't know where they are.

Half an hour ago, I asked:

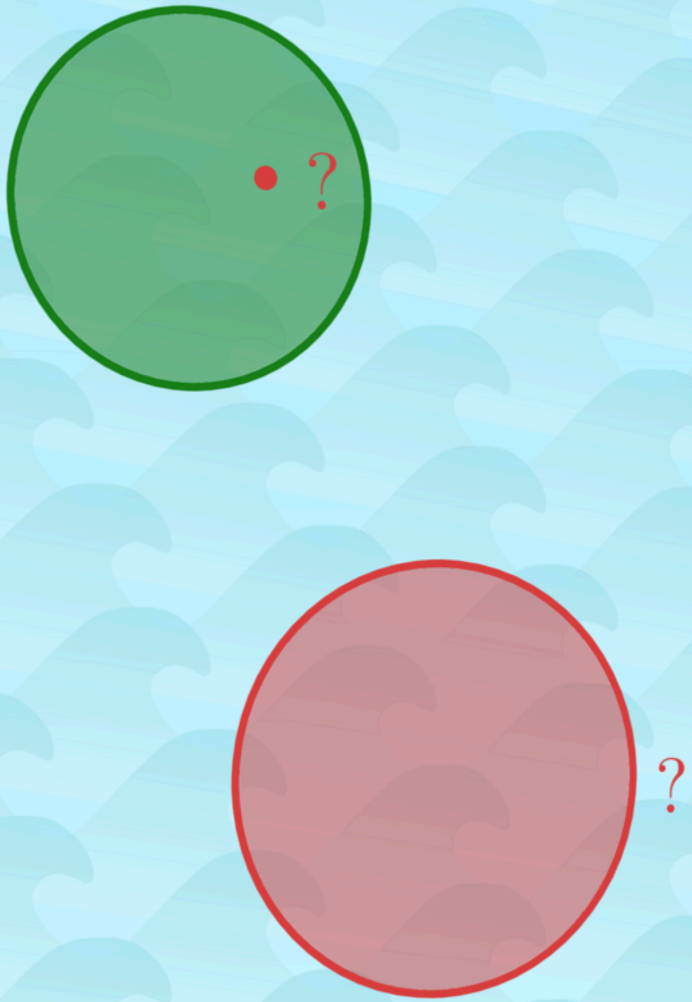
"*Where* are you"?

ID → DISK

Maybe we can also ask:

"There are *who*"?

DISK → [ID]



Suppose we have a set of points, but we don't know where they are.

Half an hour ago, I asked:

"*Where* are you"?

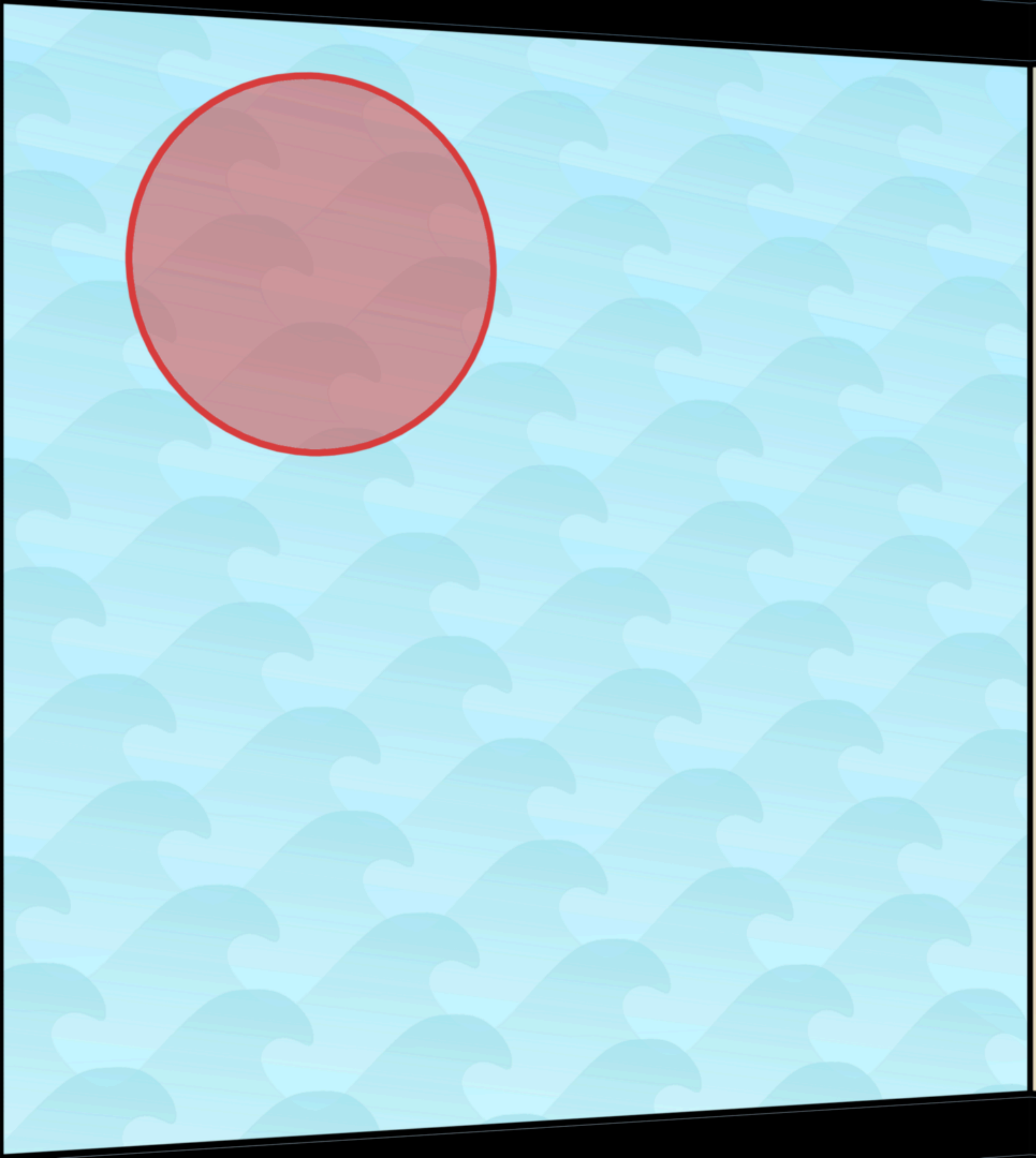
ID → DISK

Maybe we can also ask:

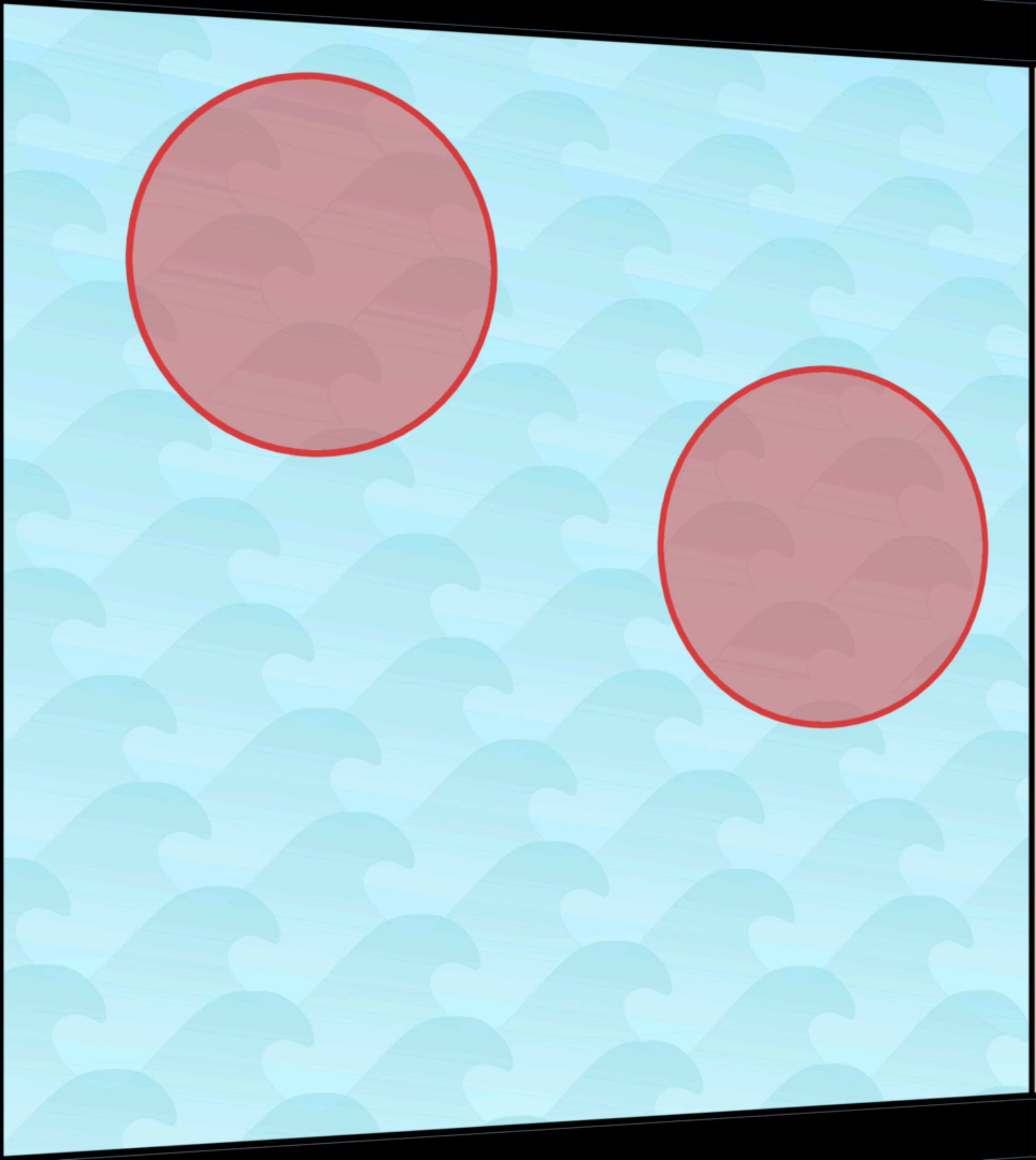
"There are *who*"?

DISK → [ID]

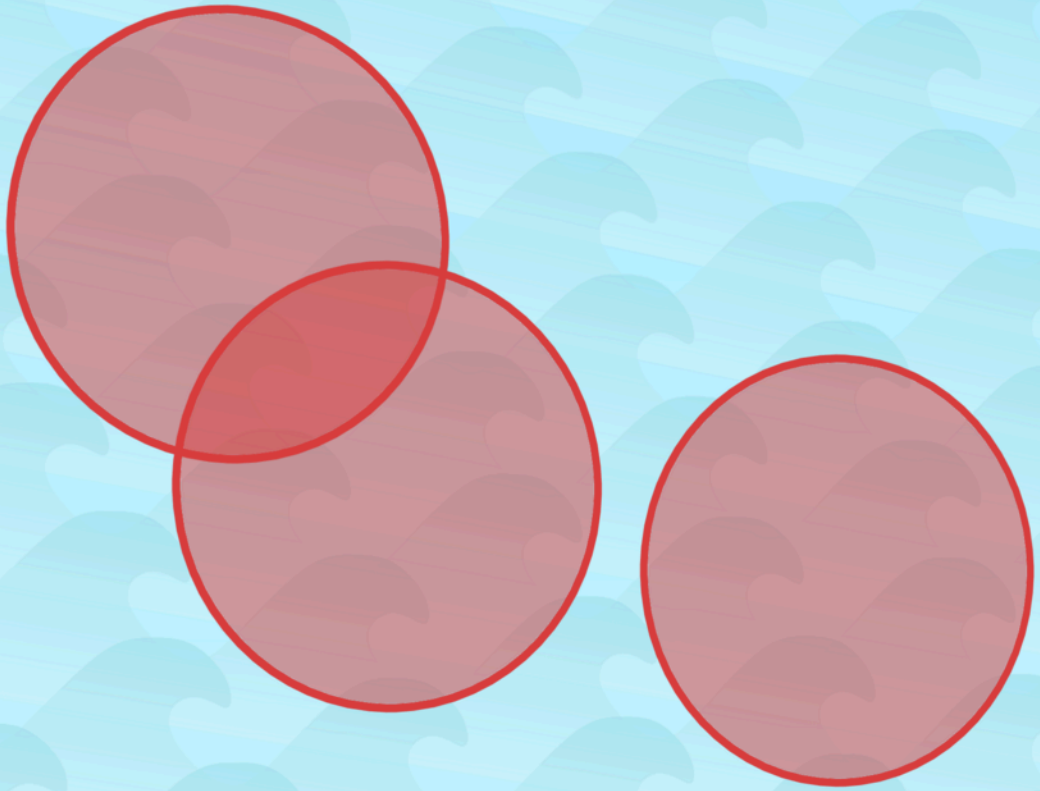
In this model, how should we query to find a specific object?



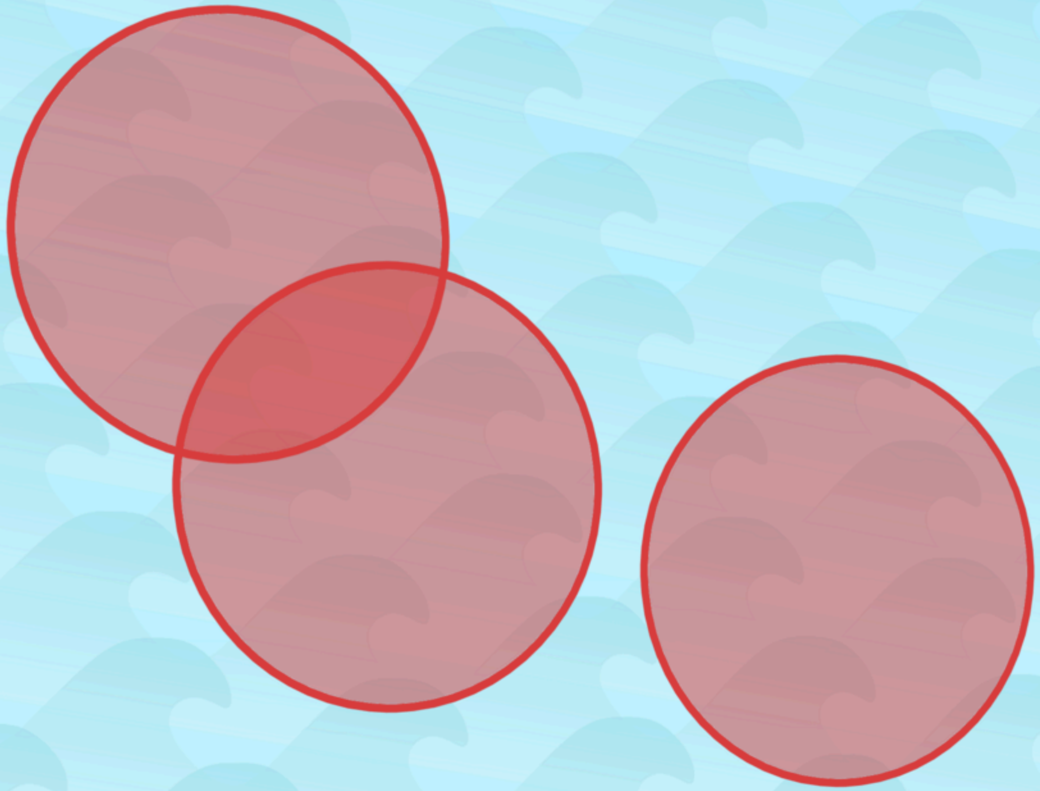
In this model, how should we query to find a specific object?



In this model, how should we query to find a specific object?

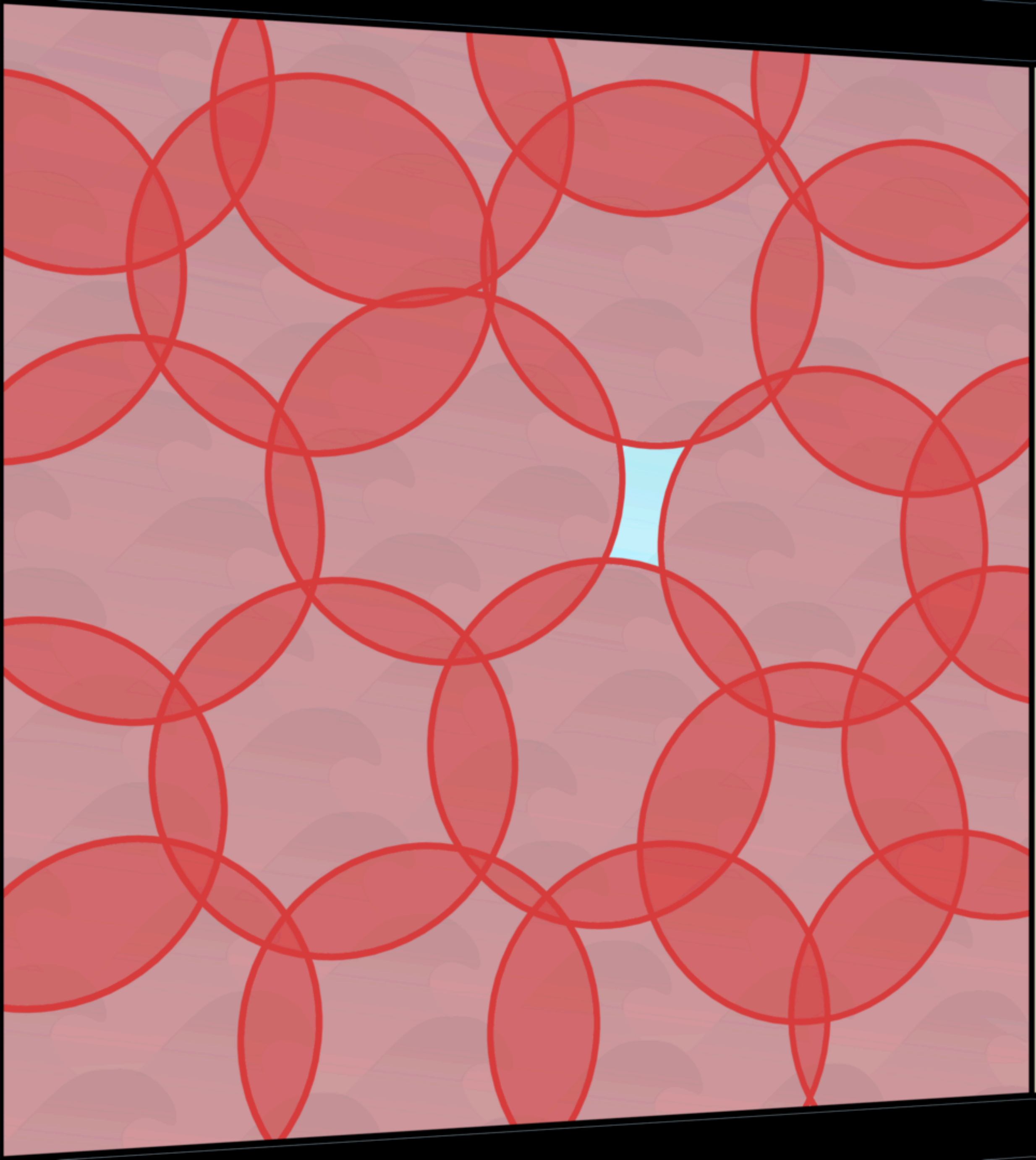


In this model, how should we query to find a specific object?



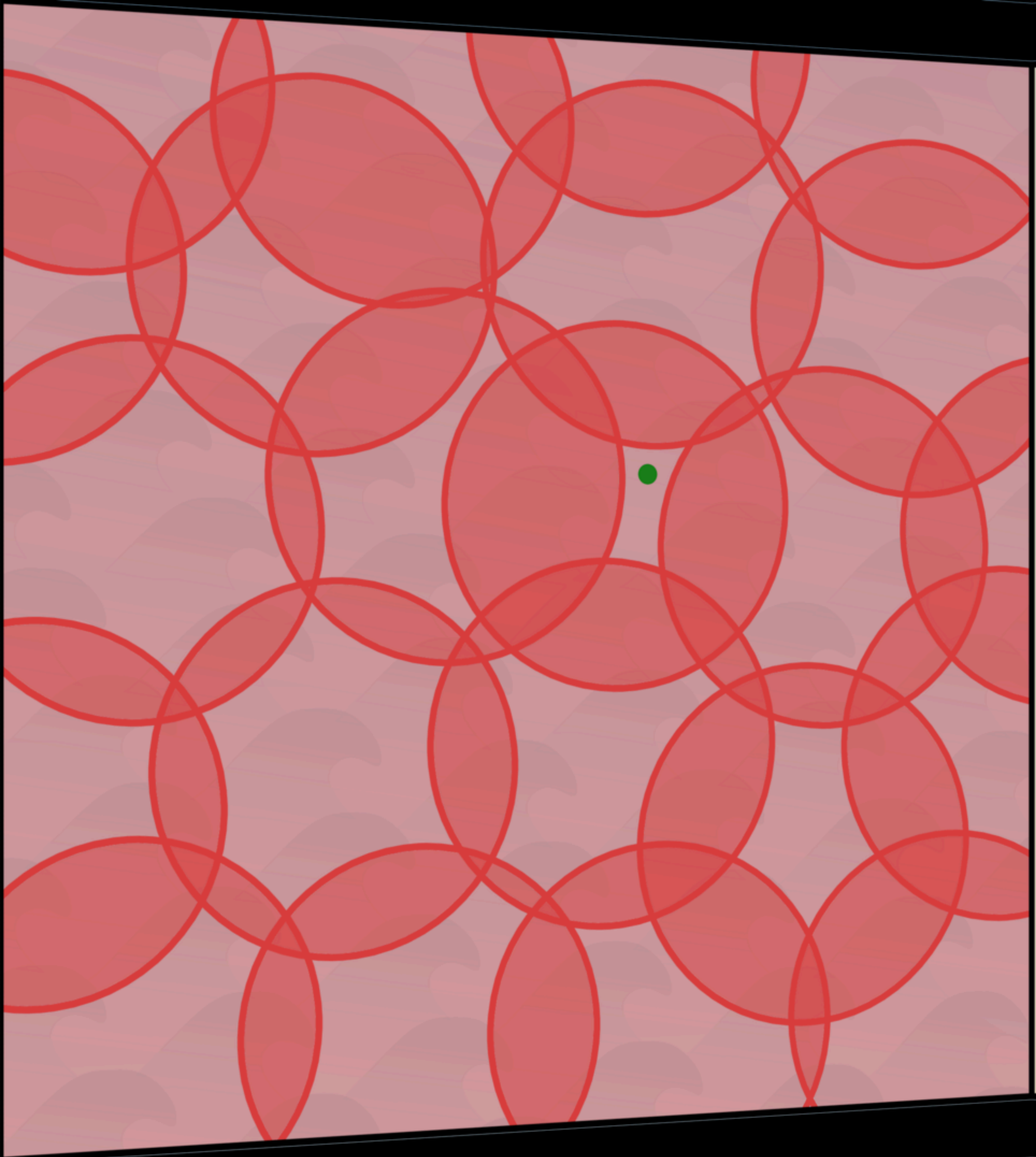
In this model, how should we query to find a specific object?

In the worst case, we find it with the *last* query.



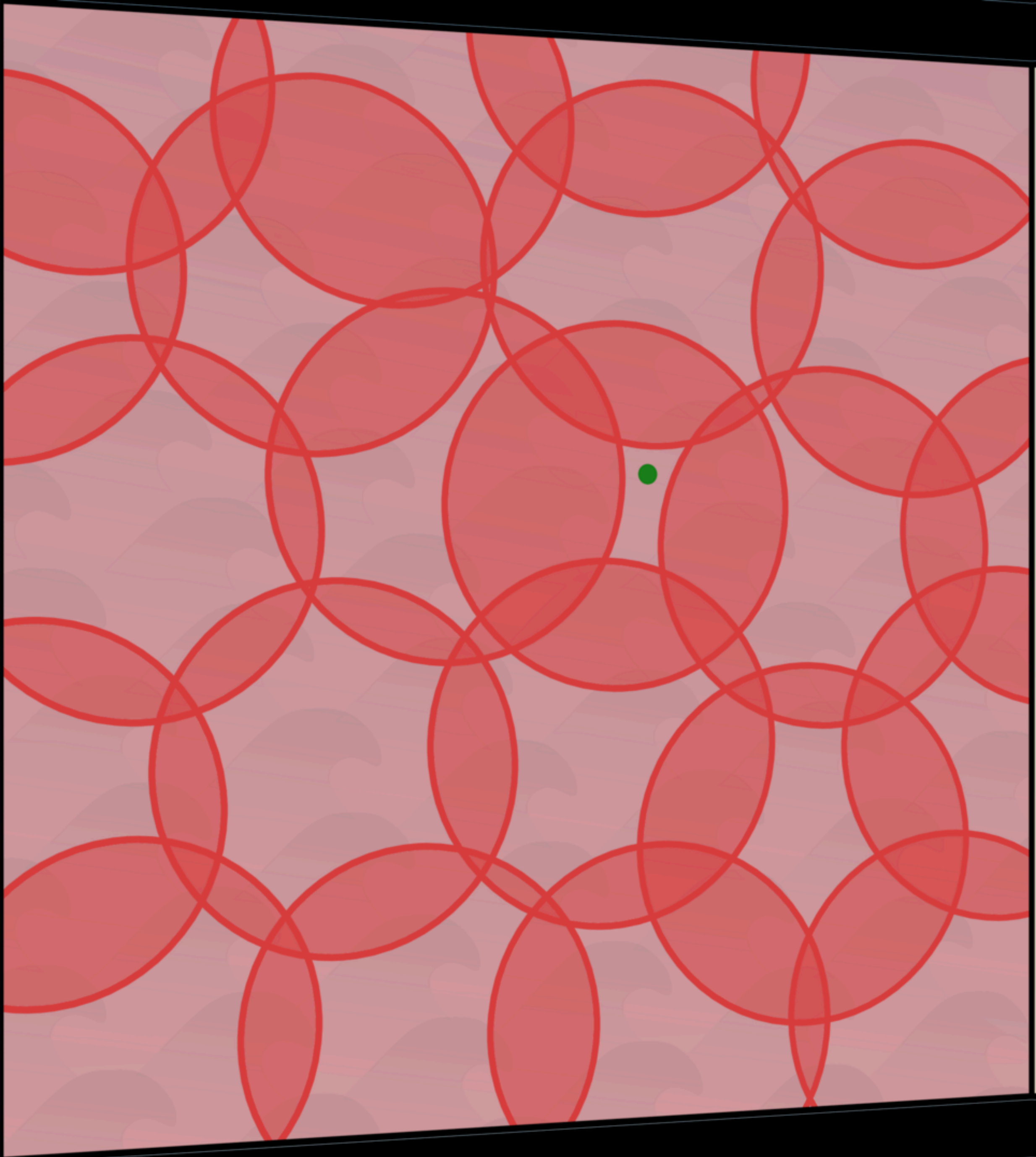
In this model, how should we query to find a specific object?

In the worst case, we find it with the *last* query.



In this model, how should we query to find a specific object?

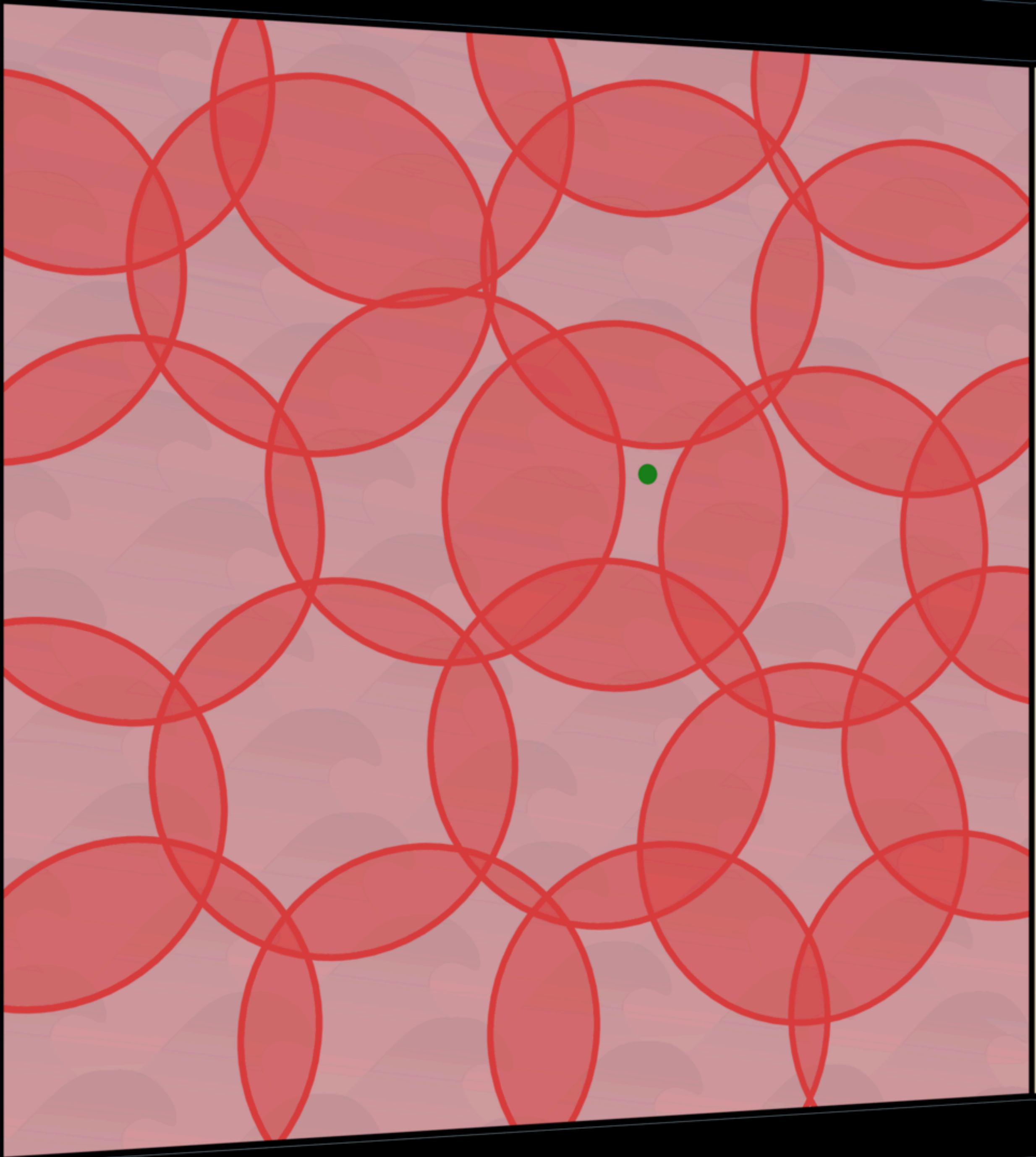
In the worst case, we find it with the *last* query.



In this model, how should we query to find a specific object?

In the worst case, we find it with the *last* query.

How many queries do we need at least?

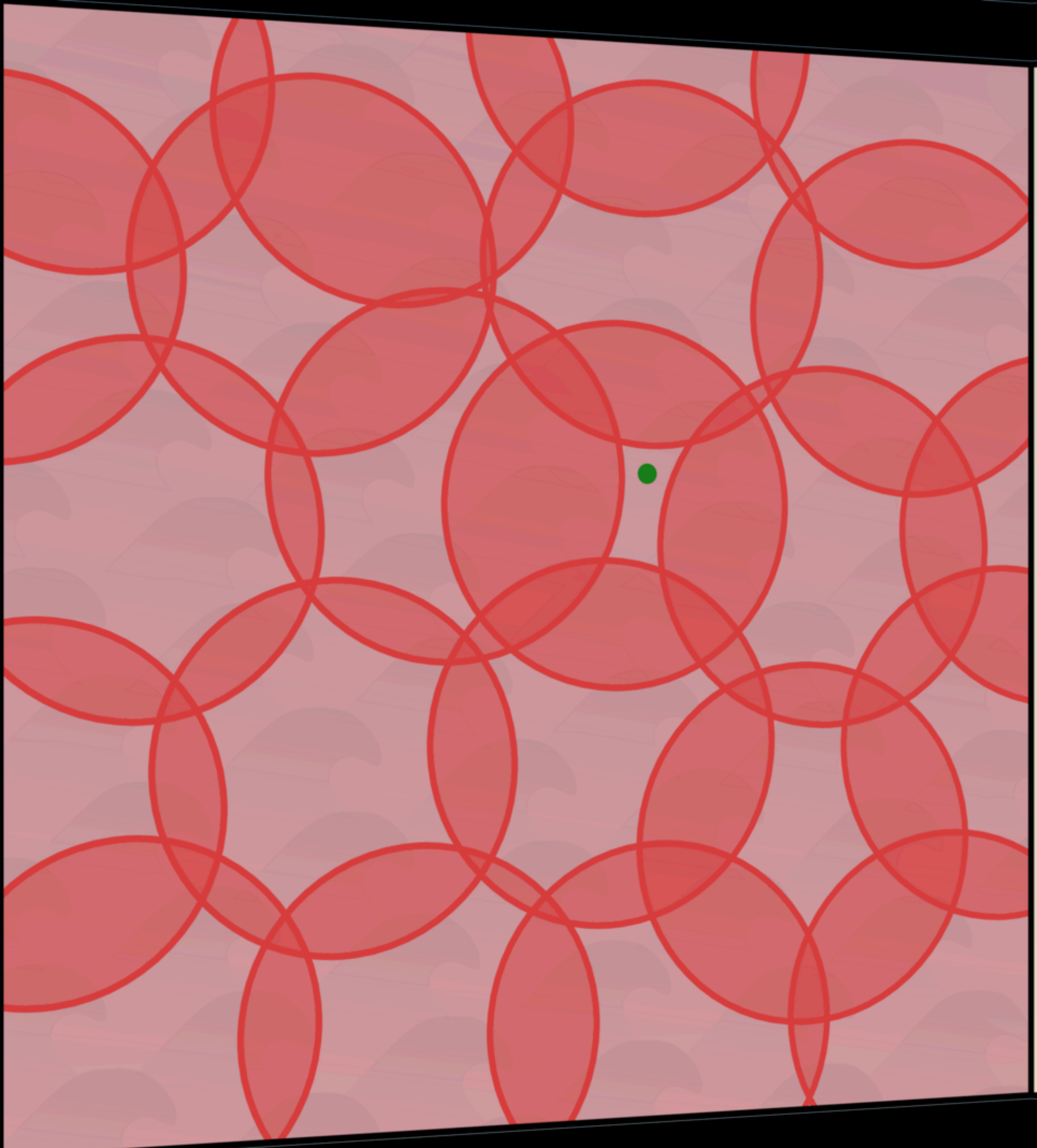


In this model, how should we query to find a specific object?

In the worst case, we find it with the *last* query.

How many queries do we need at least?

How many disks do we need to *cover* a region?



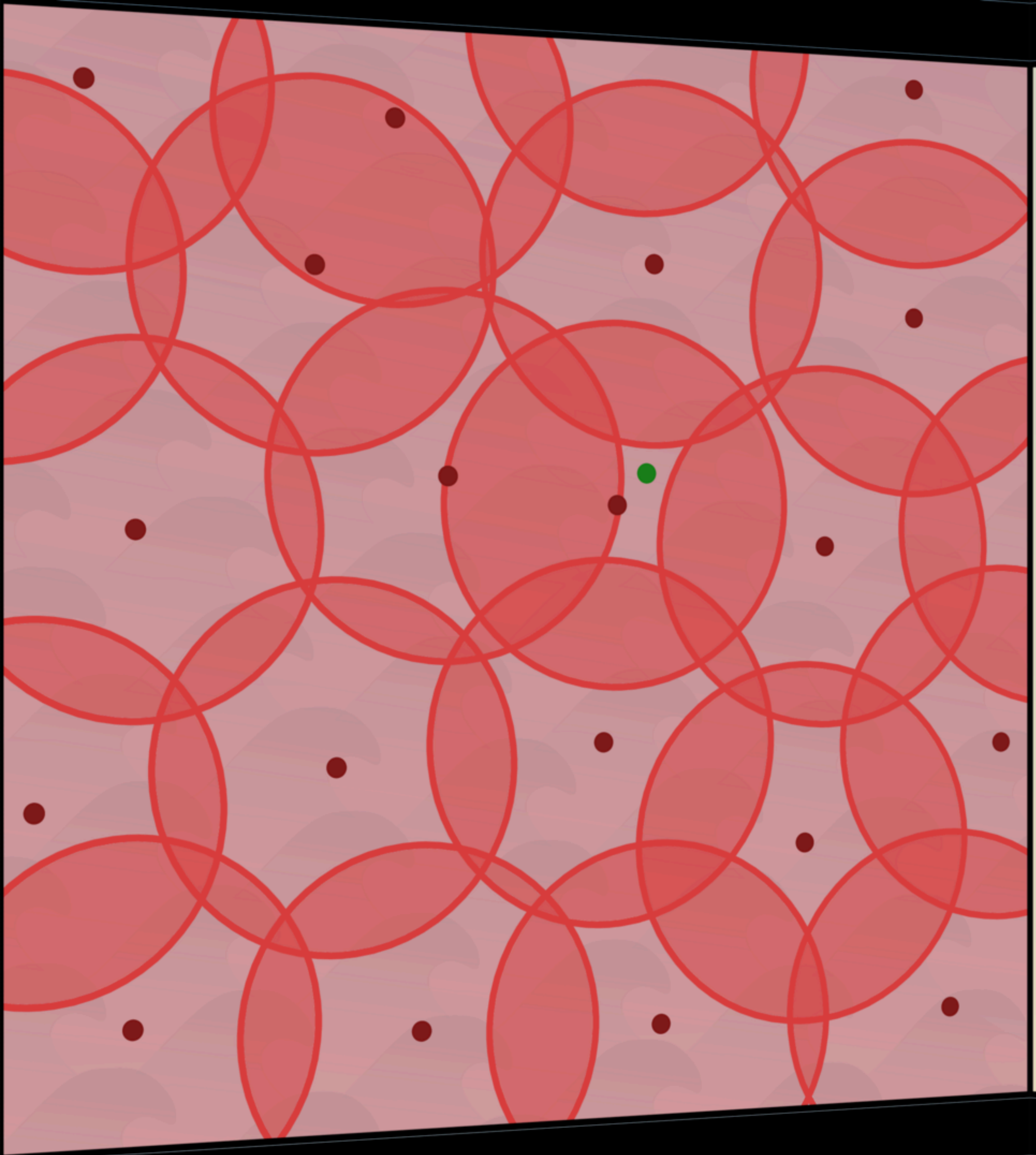
In this model, how should we query to find a specific object?

In the worst case, we find it with the *last* query.

How many queries do we need at least?

How many disks do we need to *cover* a region?

Dually, how many points do we need to *hit* a disk?



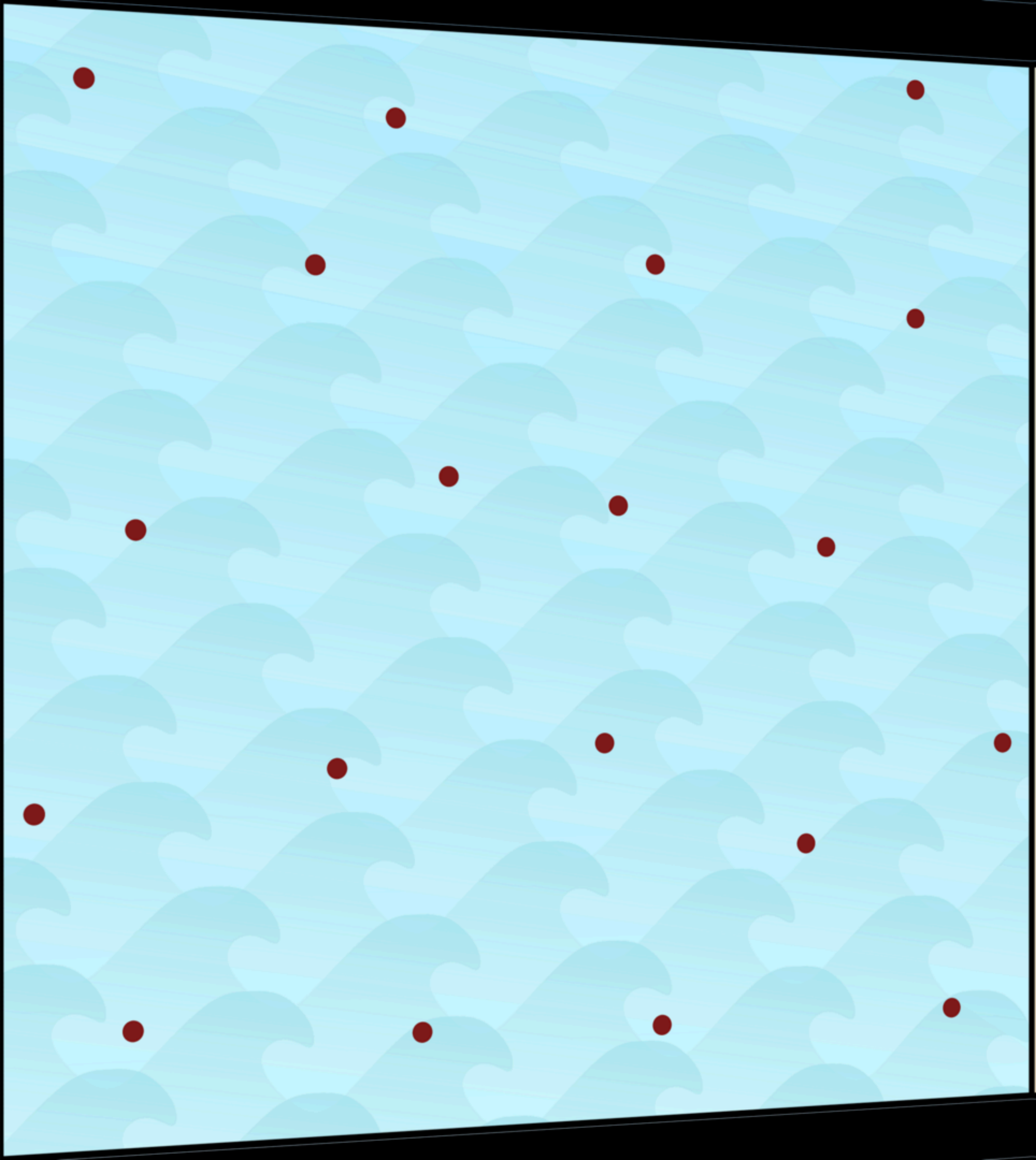
In this model, how should we query to find a specific object?

In the worst case, we find it with the *last* query.

How many queries do we need at least?

How many disks do we need to *cover* a region?

Dually, how many points do we need to *hit* a disk?



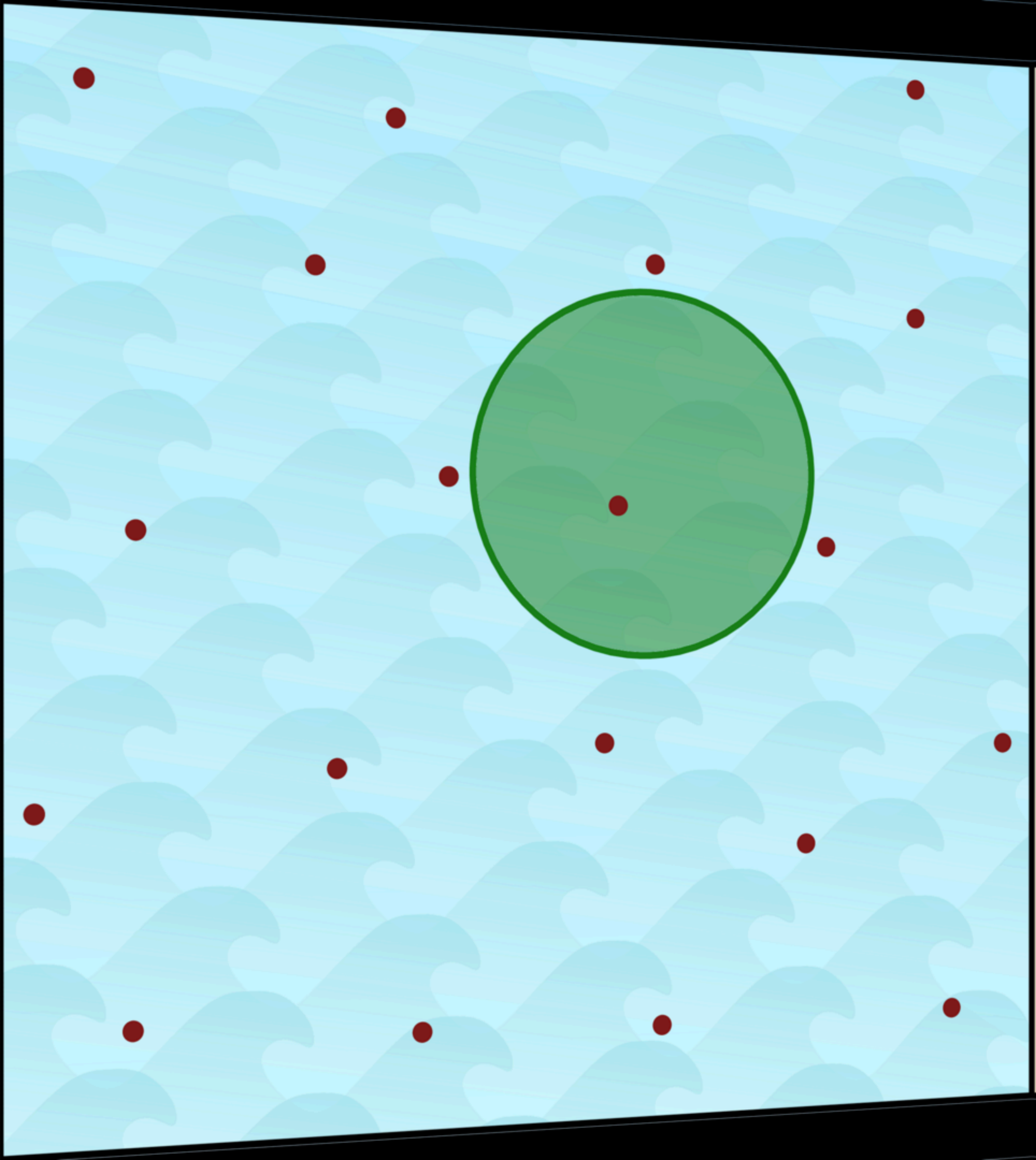
In this model, how should we query to find a specific object?

In the worst case, we find it with the *last* query.

How many queries do we need at least?

How many disks do we need to *cover* a region?

Dually, how many points do we need to *hit* a disk?



In this model, how should we query to find a specific object?

In the worst case, we find it with the *last* query.

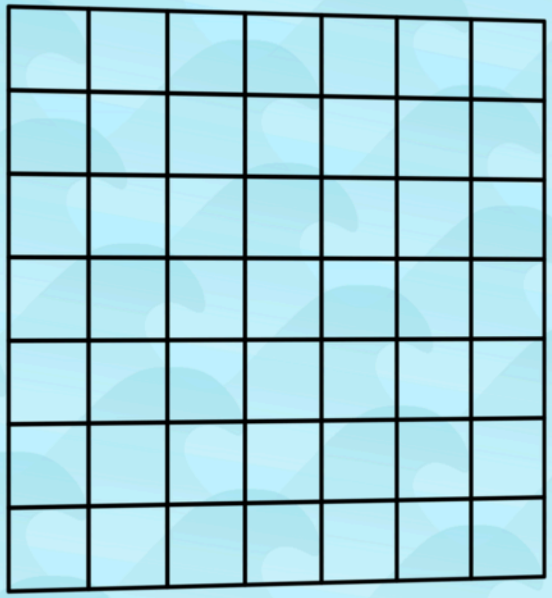
How many queries do we need at least?

How many disks do we need to *cover* a region?

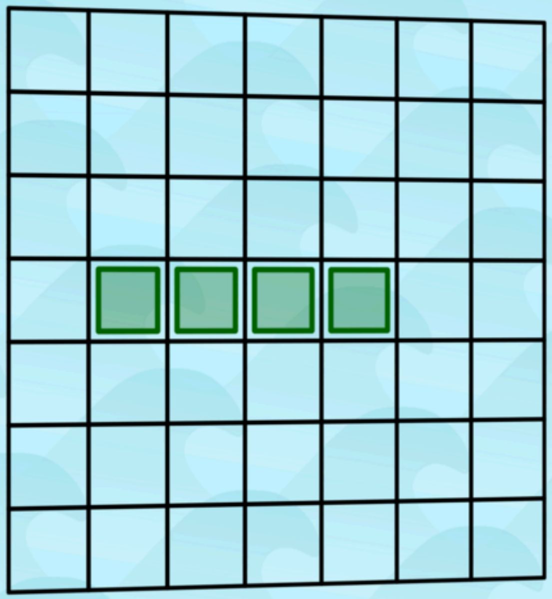
Dually, how many points do we need to *hit* a disk?

CHAPTER 2

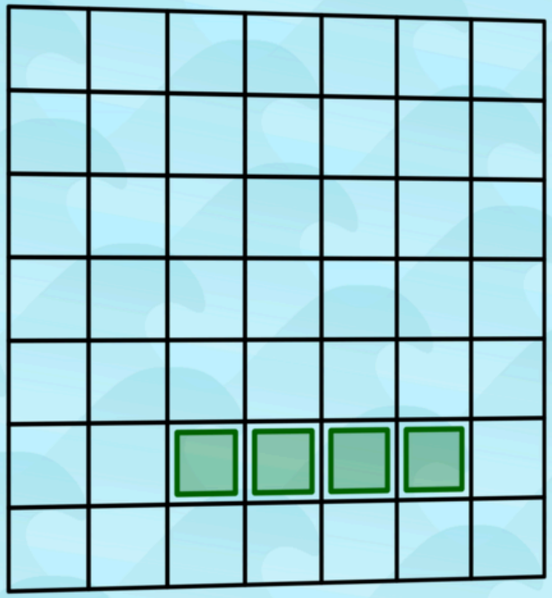
BATTLESHIPS



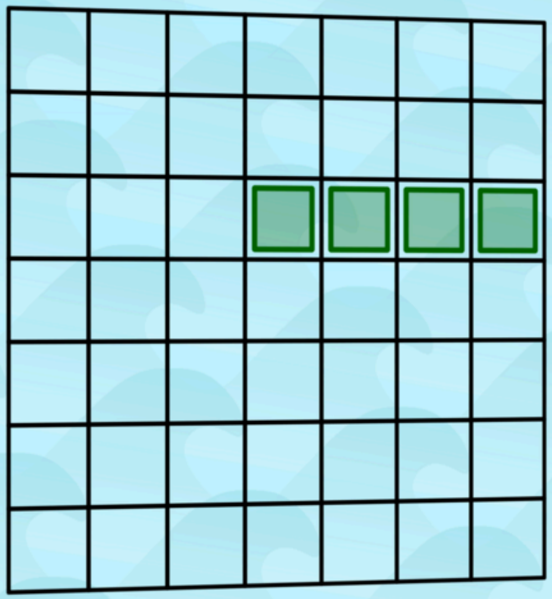
Somewhere on a grid is a battleship of known *shape* but unknown *location*.



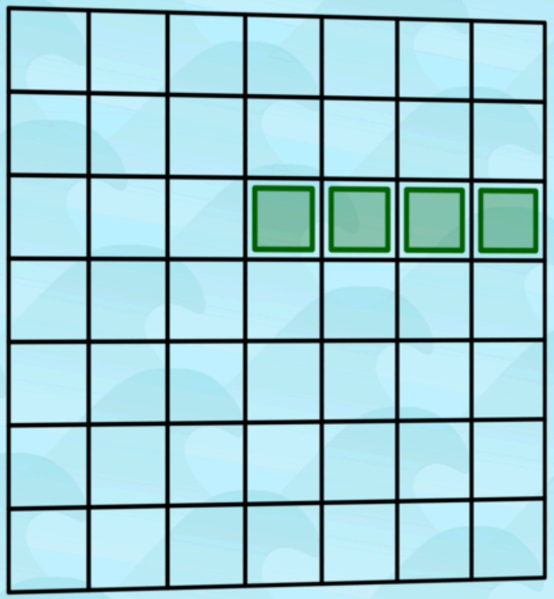
Somewhere on a grid is a battleship of known *shape* but unknown *location*.



Somewhere on a grid is a battleship of known *shape* but unknown *location*.

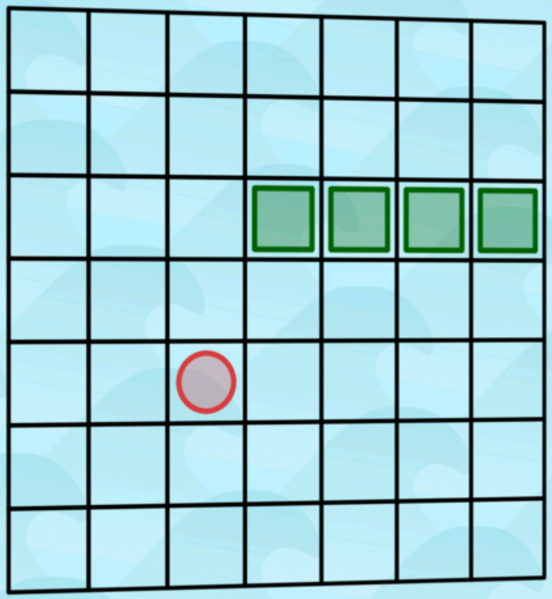


Somewhere on a grid is a battleship of known *shape* but unknown *location*.



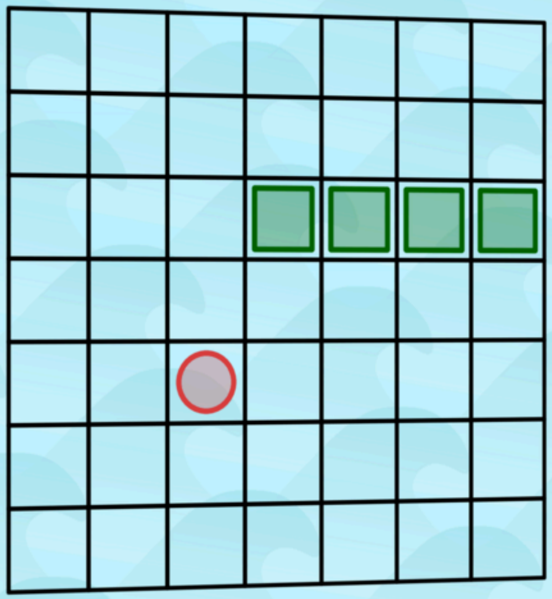
Somewhere on a grid is a battleship of known *shape* but unknown *location*.

We can query a *cell* and ask: is anything there?



Somewhere on a grid is a battleship of known *shape* but unknown *location*.

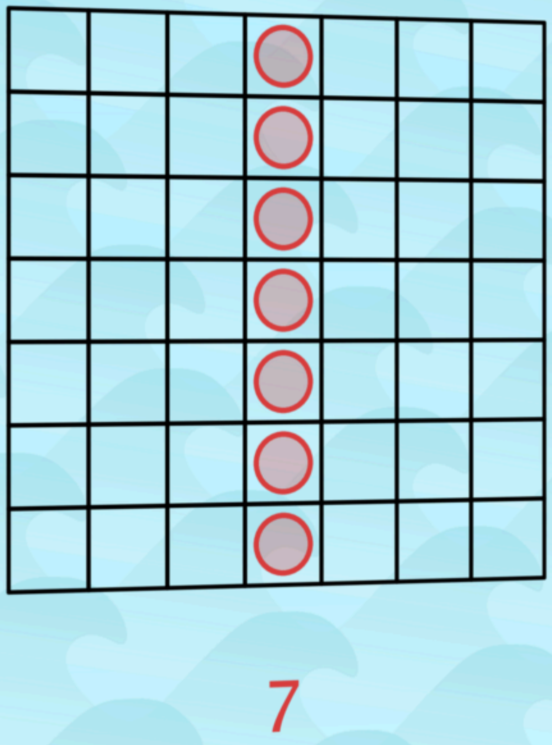
We can query a *cell* and ask: is anything there?



Somewhere on a grid is a battleship of known *shape* but unknown *location*.

We can query a *cell* and ask: is anything there?

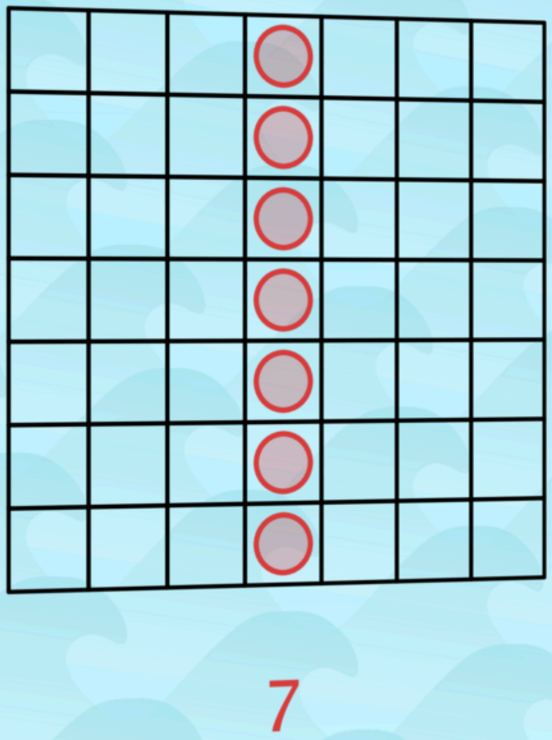
How many cells de we need to query to find the ship?



Somewhere on a grid is a battleship of known *shape* but unknown *location*.

We can query a *cell* and ask: is anything there?

How many cells de we need to query to find the ship?

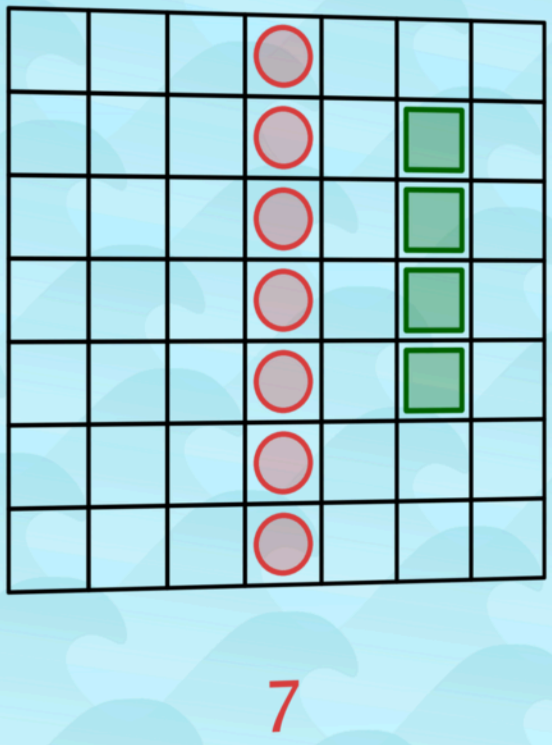


Somewhere on a grid is a battleship of known *shape* but unknown *location*.

We can query a *cell* and ask: is anything there?

How many cells de we need to query to find the ship?

In the game, ships may also be *rotated*.

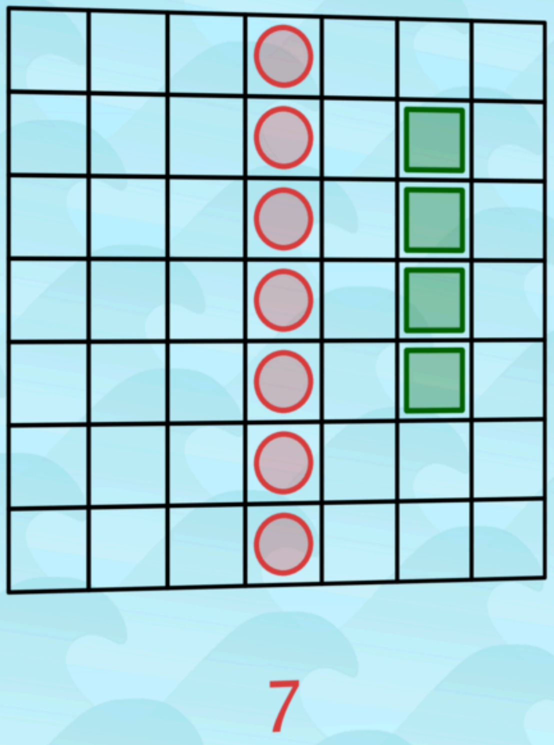


Somewhere on a grid is a battleship of known *shape* but unknown *location*.

We can query a *cell* and ask: is anything there?

How many cells de we need to query to find the ship?

In the game, ships may also be *rotated*.

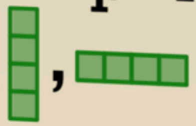


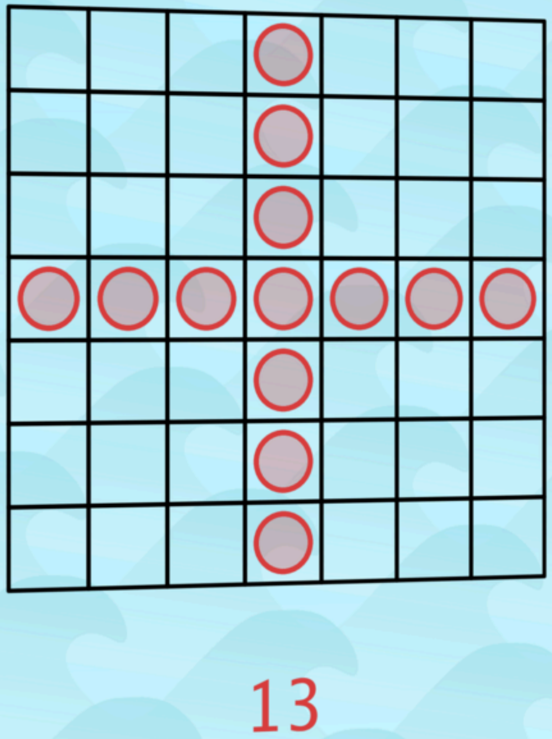
Somewhere on a grid is a battleship of known *shape* but unknown *location*.

We can query a *cell* and ask: is anything there?

How many cells de we need to query to find the ship?

In the game, ships may also be *rotated*.

If so, we say the ship has *unknown shape*: S={}

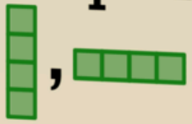


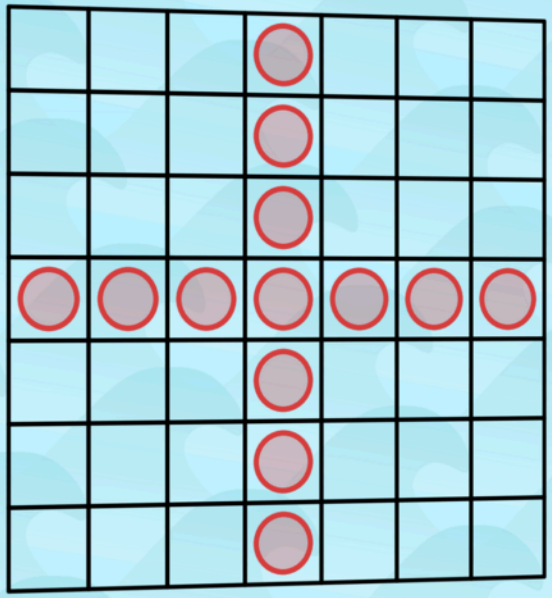
Somewhere on a grid is a battleship of known *shape* but unknown *location*.

We can query a *cell* and ask: is anything there?

How many cells de we need to query to find the ship?

In the game, ships may also be *rotated*.

If so, we say the ship has *unknown shape*: S={  }



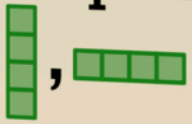
13

Somewhere on a grid is a battleship of known *shape* but unknown *location*.

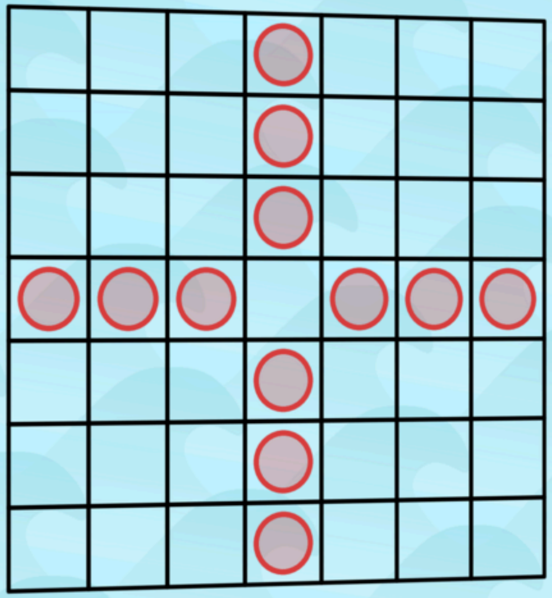
We can query a *cell* and ask: is anything there?

How many cells de we need to query to find the ship?

In the game, ships may also be *rotated*.

If so, we say the ship has *unknown shape*: S={  }

What is the best pattern?



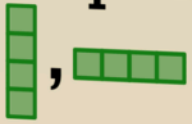
12

Somewhere on a grid is a battleship of known *shape* but unknown *location*.

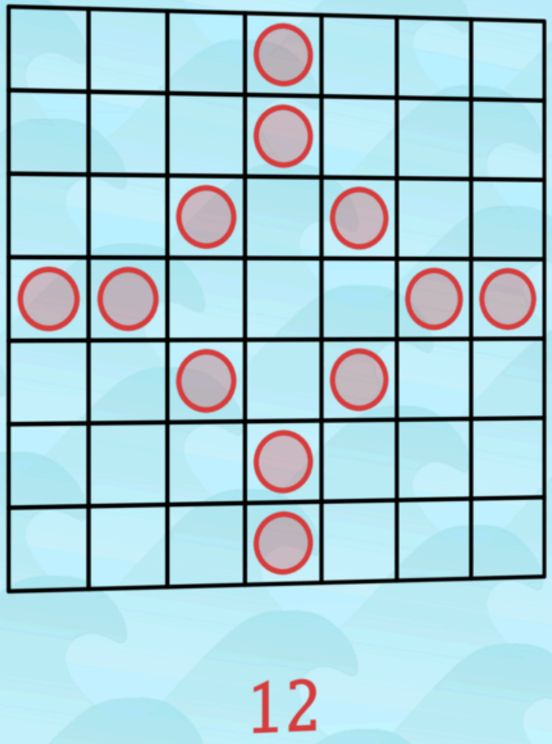
We can query a *cell* and ask: is anything there?

How many cells de we need to query to find the ship?

In the game, ships may also be *rotated*.

If so, we say the ship has *unknown shape*: S={  }

What is the best pattern?

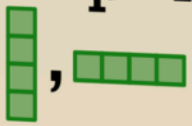


Somewhere on a grid is a battleship of known *shape* but unknown *location*.

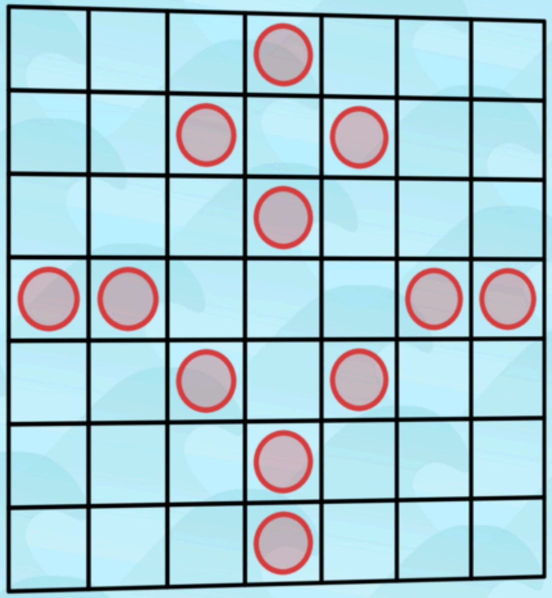
We can query a *cell* and ask: is anything there?

How many cells de we need to query to find the ship?

In the game, ships may also be *rotated*.

If so, we say the ship has *unknown shape*: S={  }

What is the best pattern?



12

Somewhere on a grid is a battleship of known *shape* but unknown *location*.

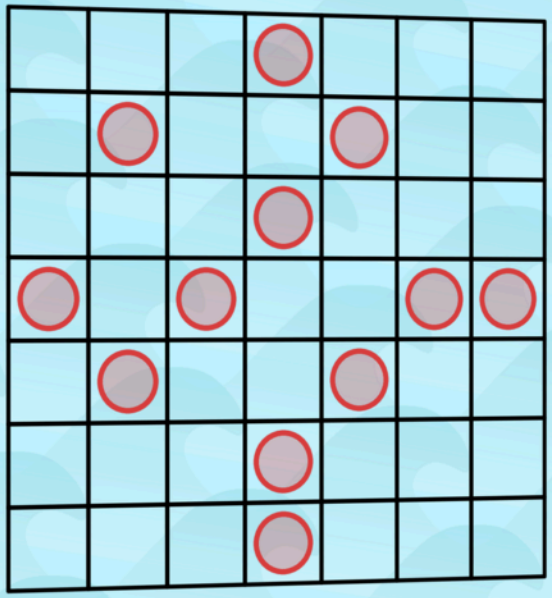
We can query a *cell* and ask: is anything there?

How many cells de we need to query to find the ship?

In the game, ships may also be *rotated*.

If so, we say the ship has *unknown shape*: S={,}

What is the best pattern?



12

Somewhere on a grid is a battleship of known *shape* but unknown *location*.

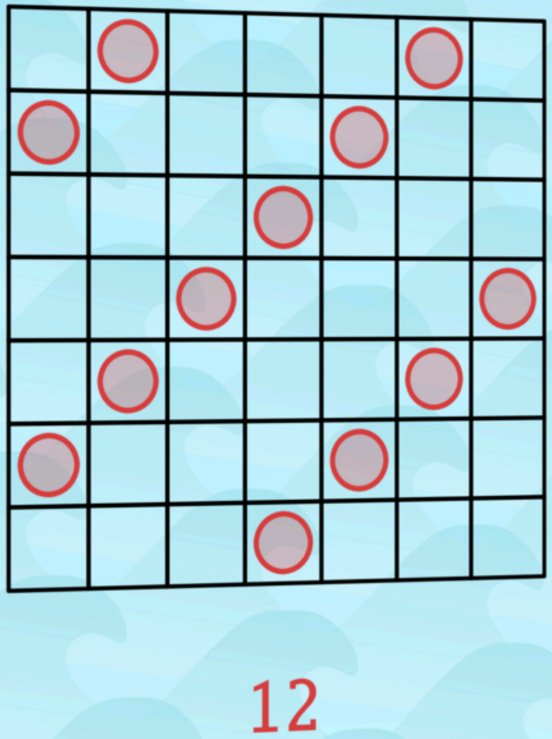
We can query a *cell* and ask: is anything there?

How many cells de we need to query to find the ship?

In the game, ships may also be *rotated*.

If so, we say the ship has *unknown shape*: S={, }

What is the best pattern?



Somewhere on a grid is a battleship of known *shape* but unknown *location*.

We can query a *cell* and ask: is anything there?

How many cells de we need to query to find the ship?

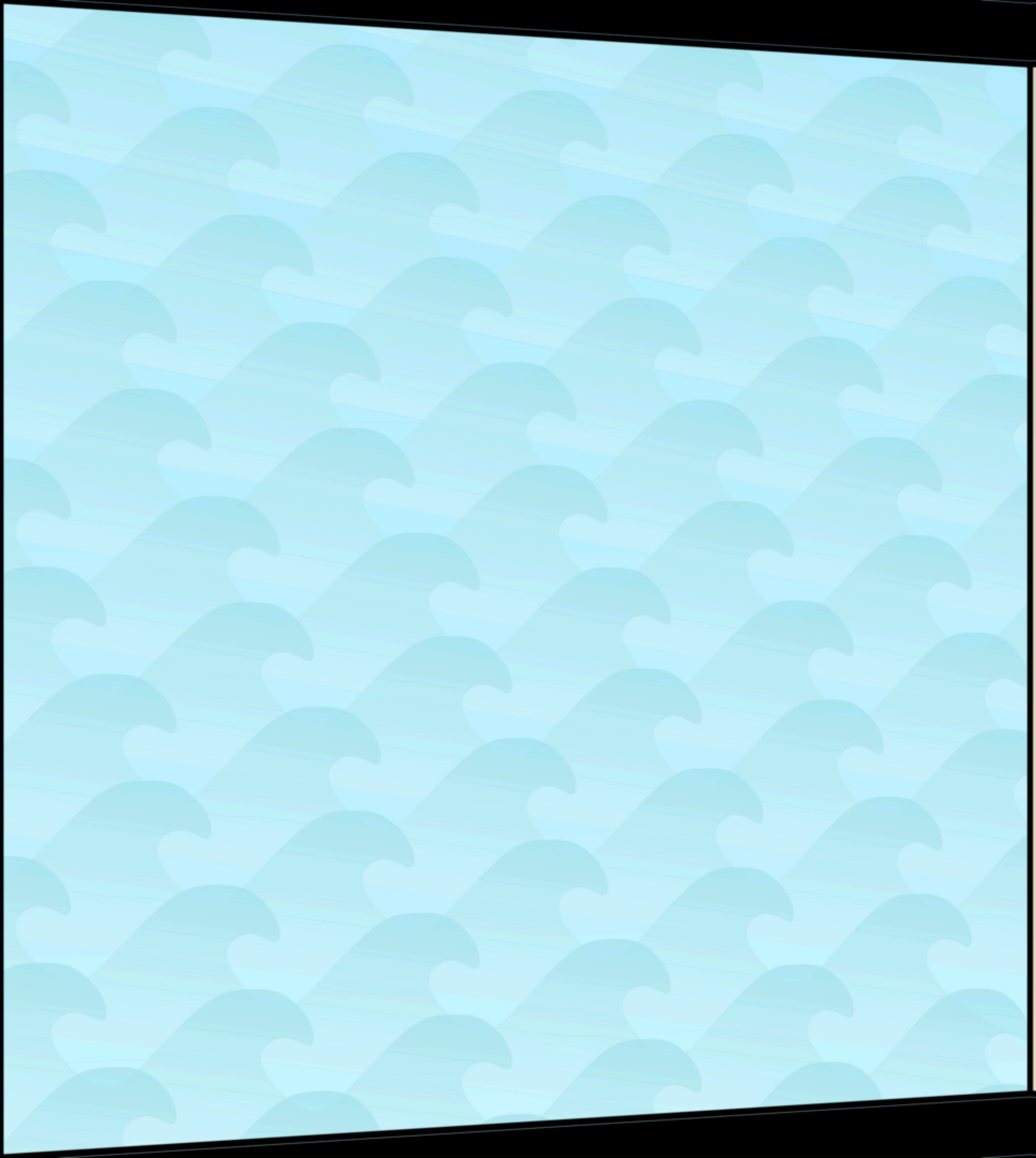
In the game, ships may also be *rotated*.

If so, we say the ship has *unknown shape*: S={ , }

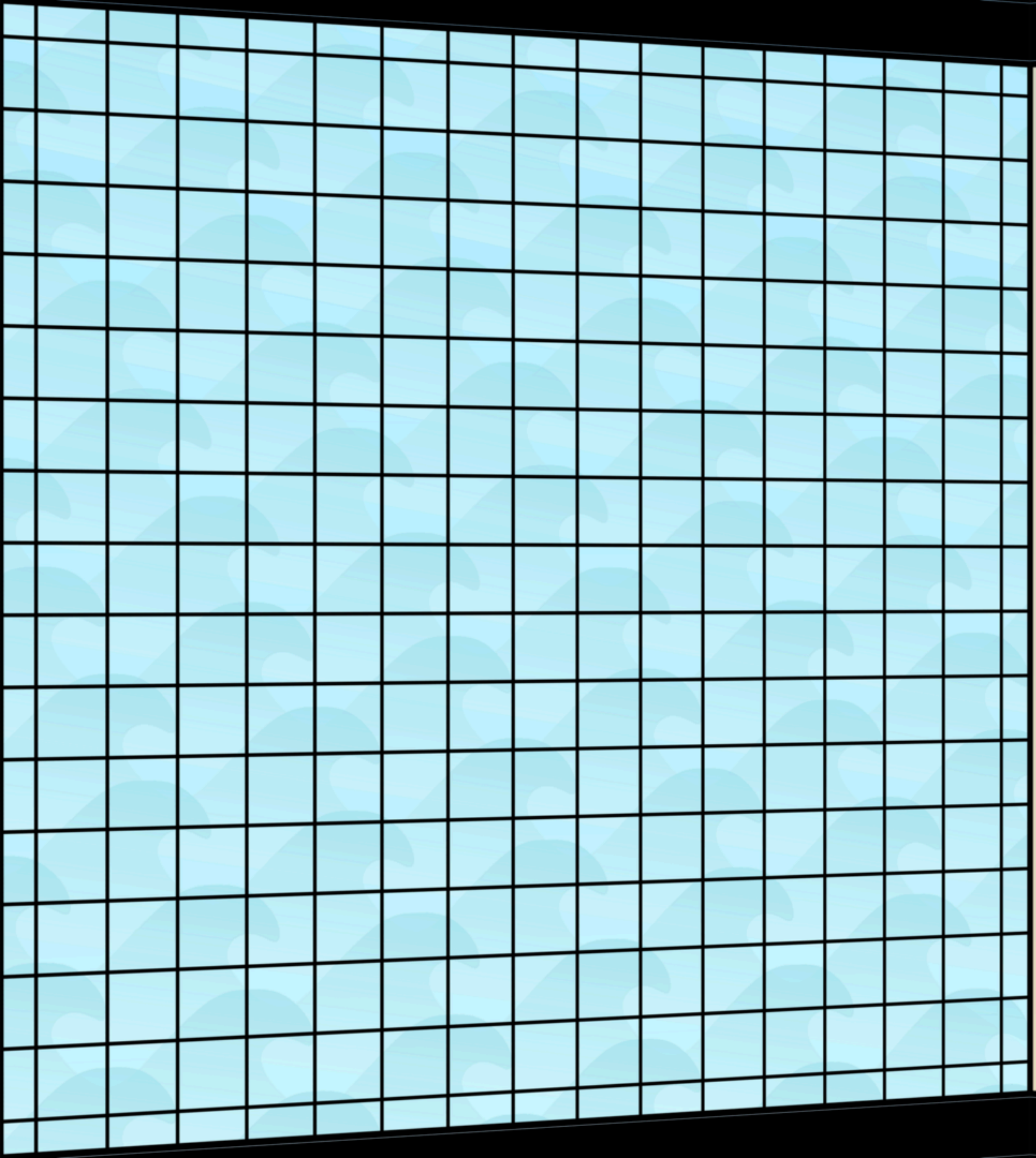
What is the best pattern?

CHAPTER 3

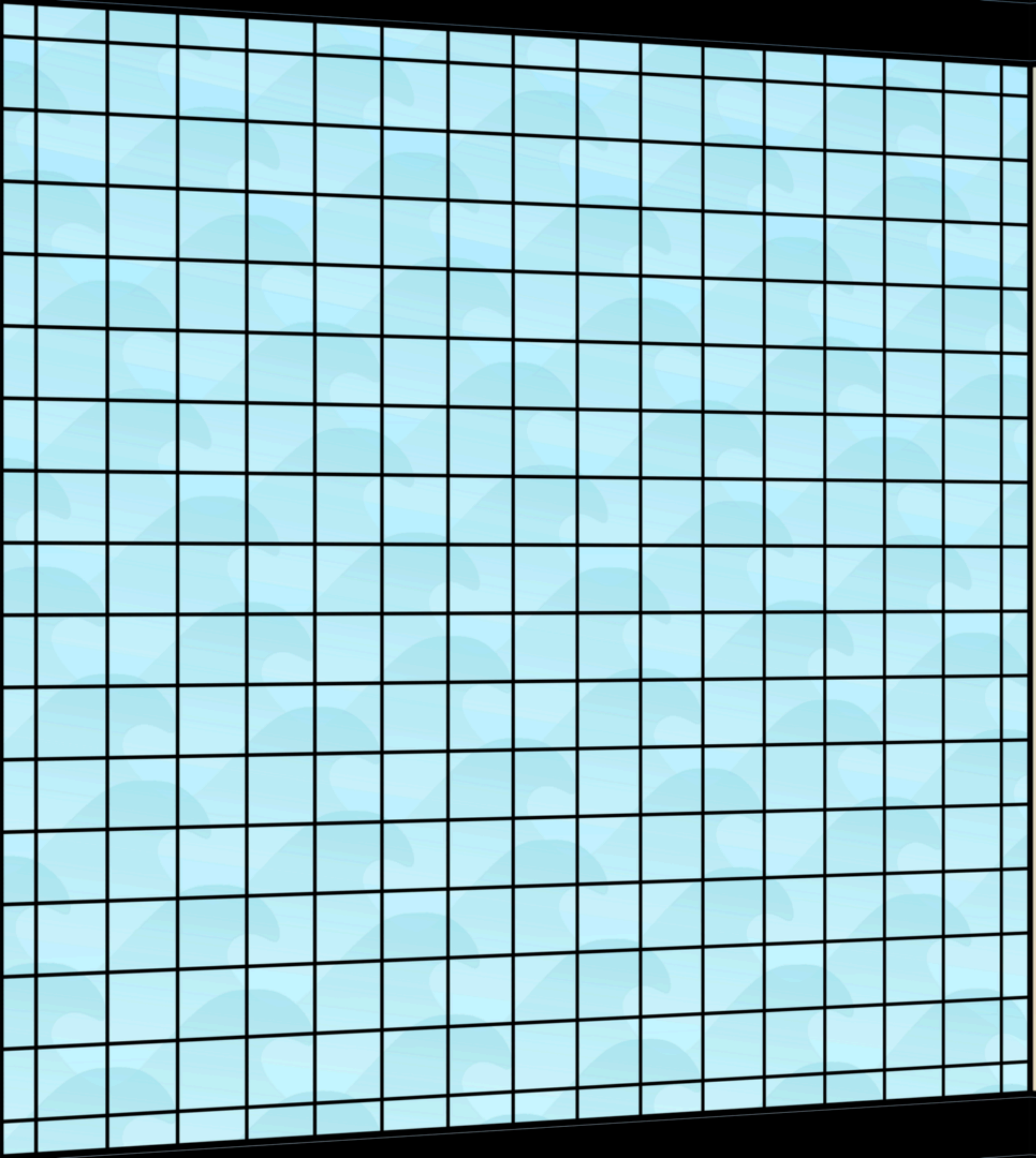
DENSITY



Let’s consider battleships on an *infinite* board.

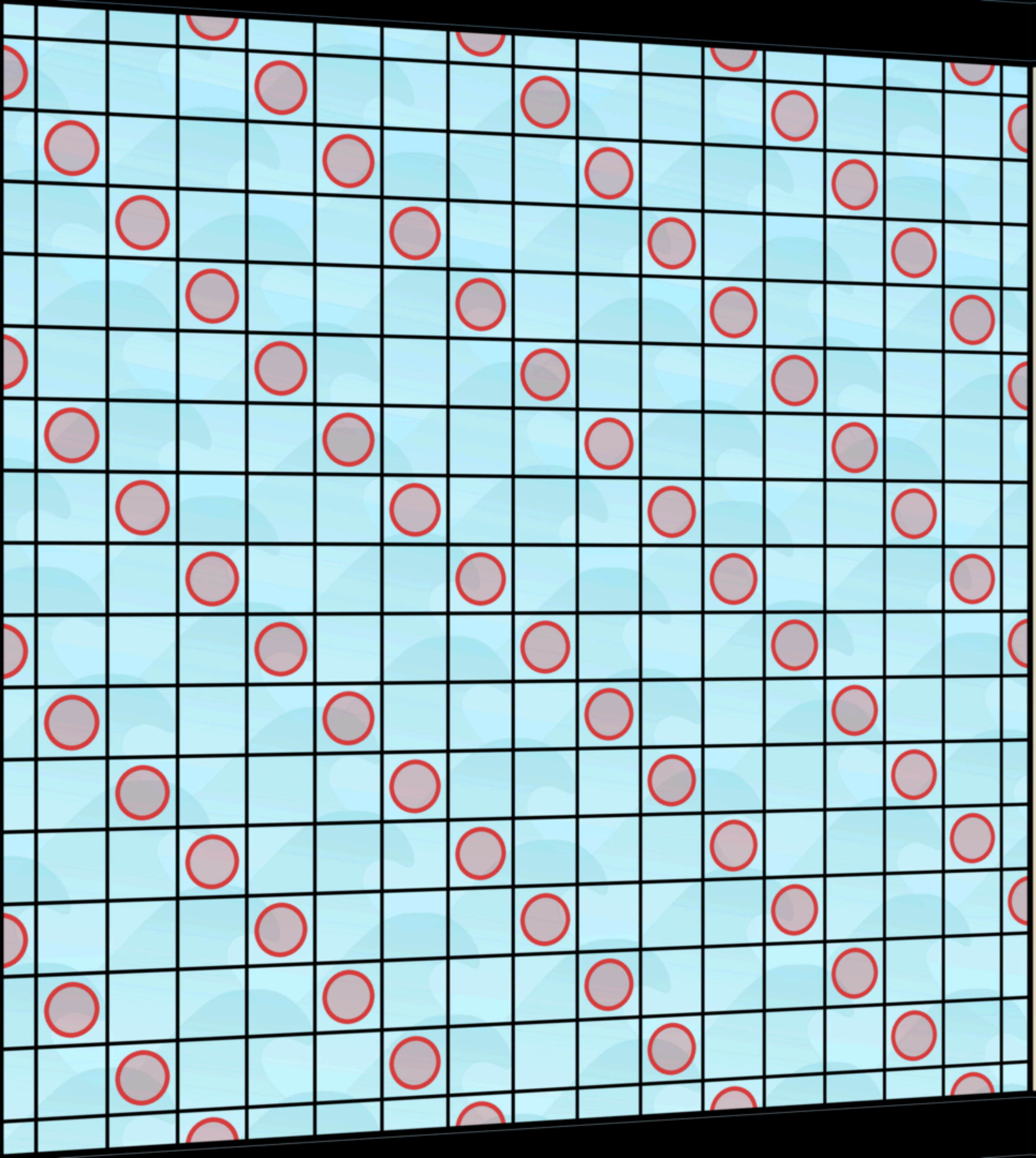


Let’s consider battleships on an *infinite* board.



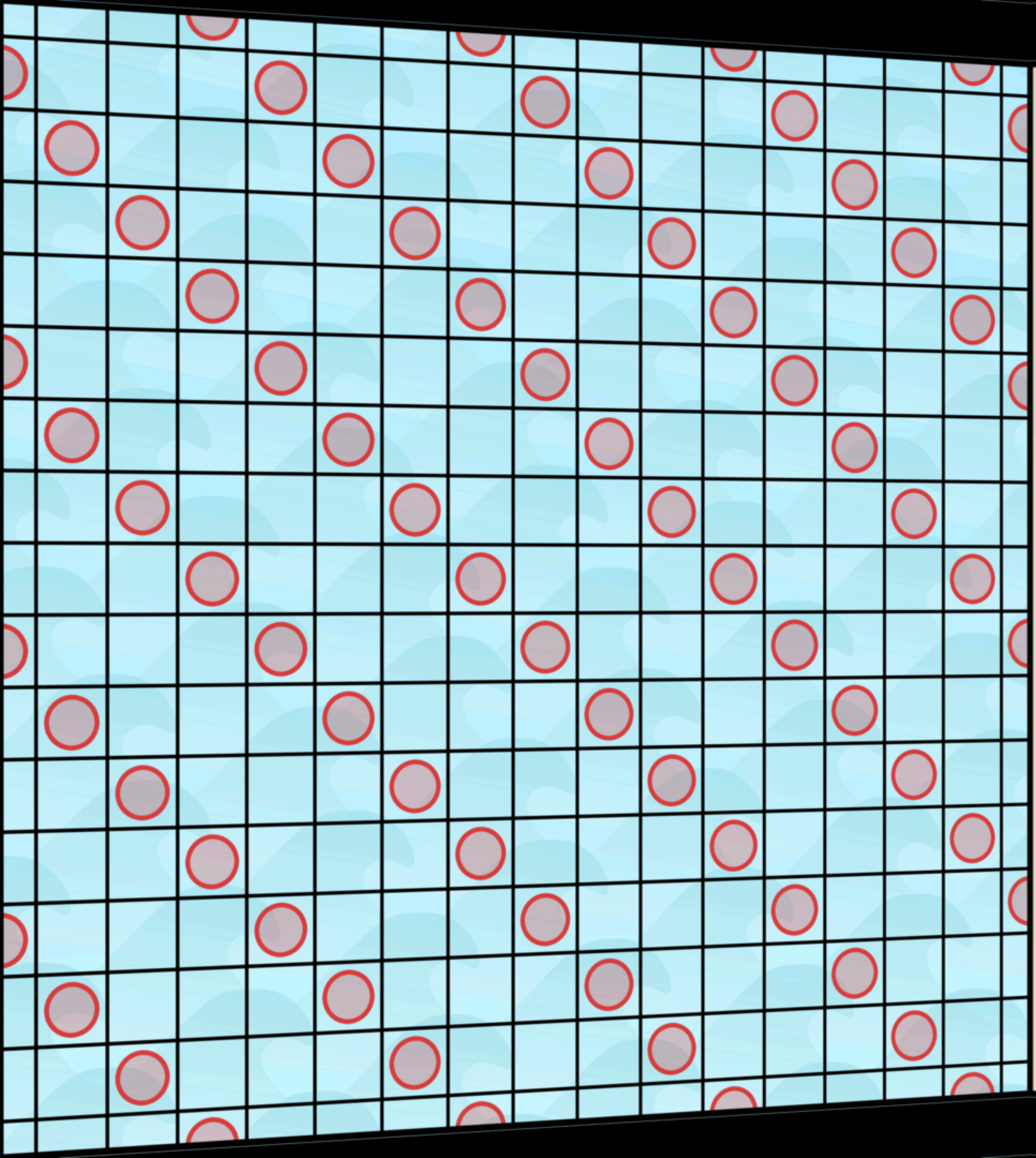
Let's consider battleships on an *infinite* board.

What is the best pattern to find S={,}?



Let's consider battleships on an *infinite* board.

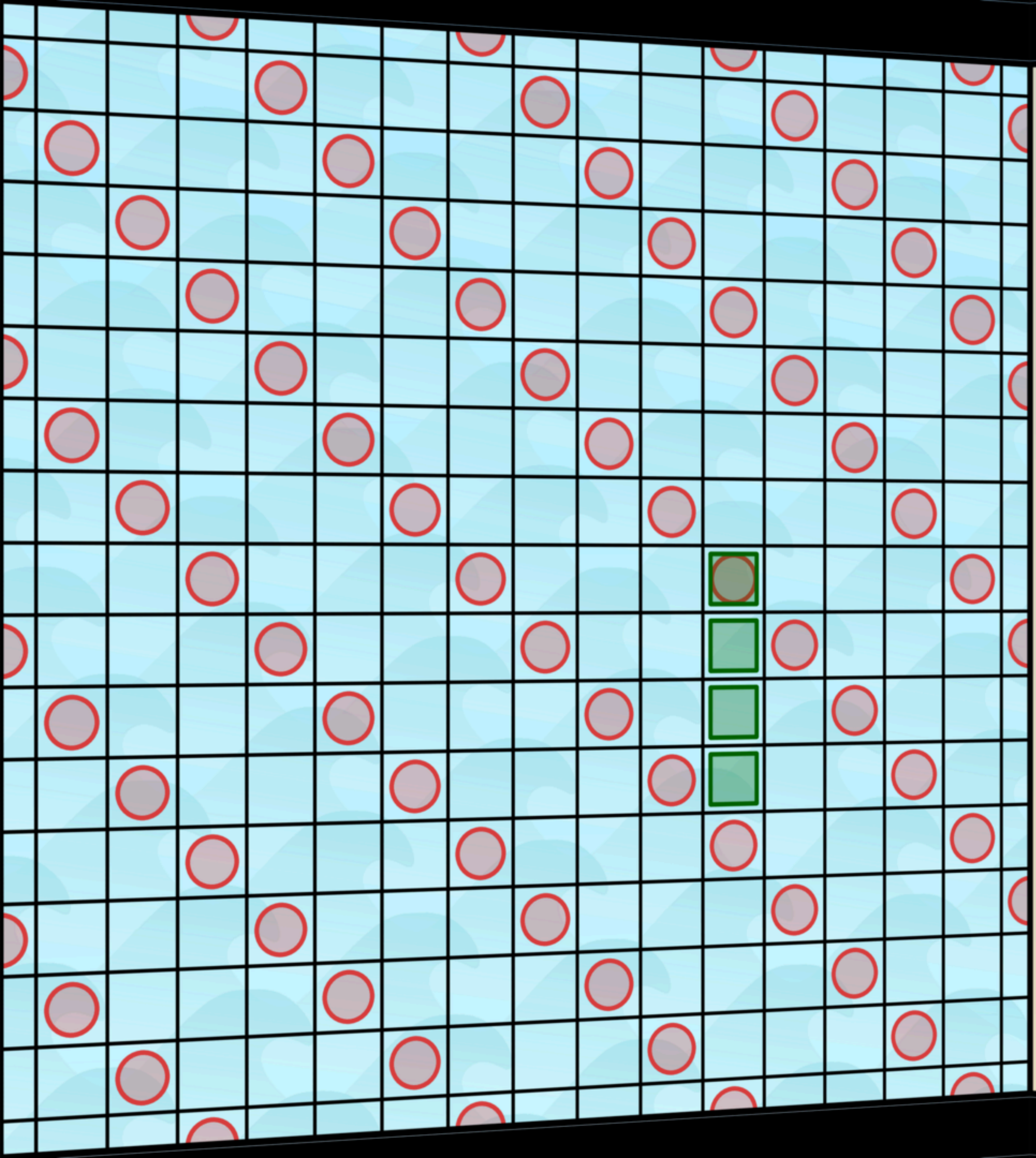
What is the best pattern to find S={,}?



Let's consider battleships on an *infinite* board.

What is the best pattern to find S={,}?

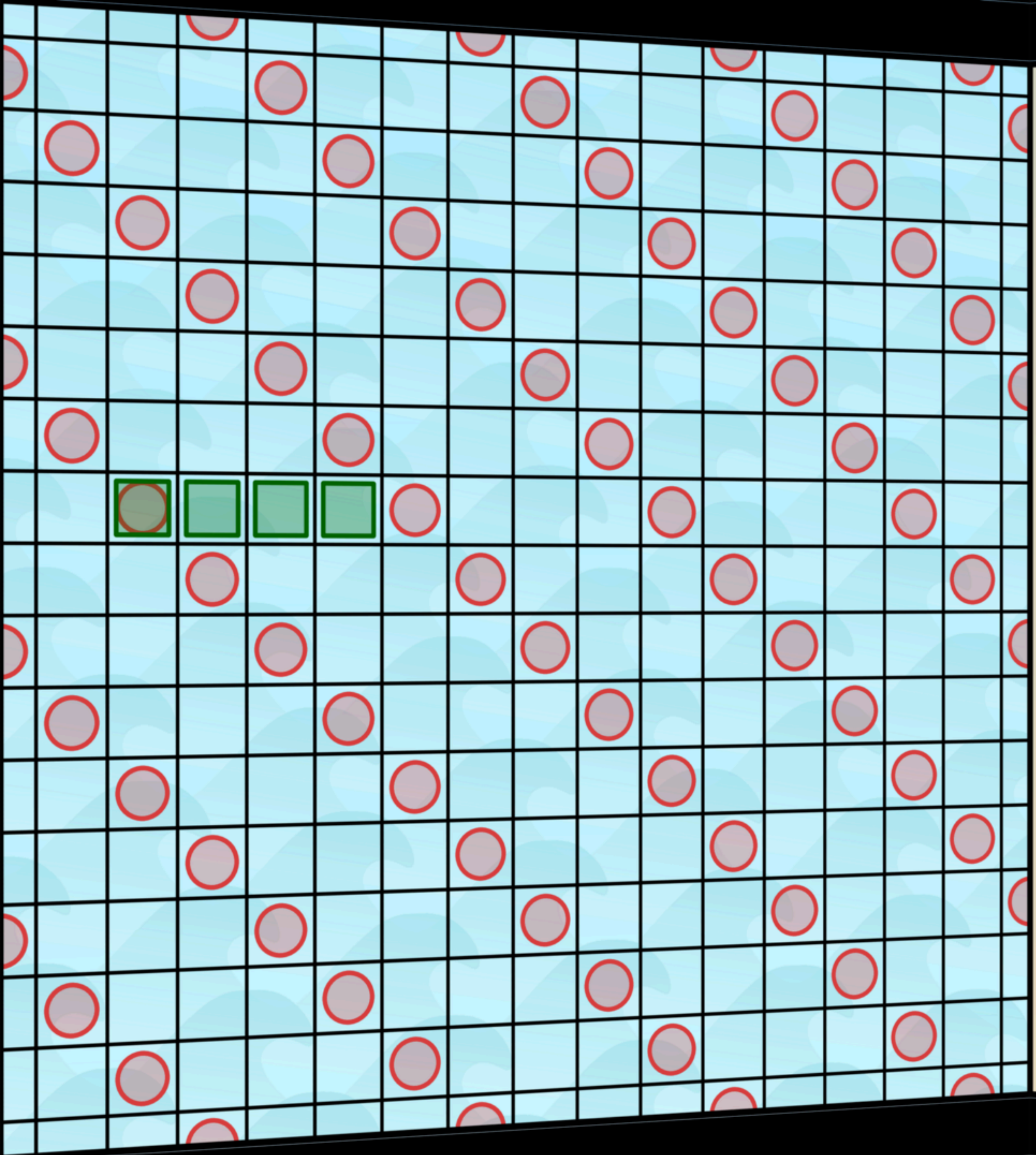
Now, it is easy to see that this pattern is optimal.



Let's consider battleships on an *infinite* board.

What is the best pattern to find S={,}?

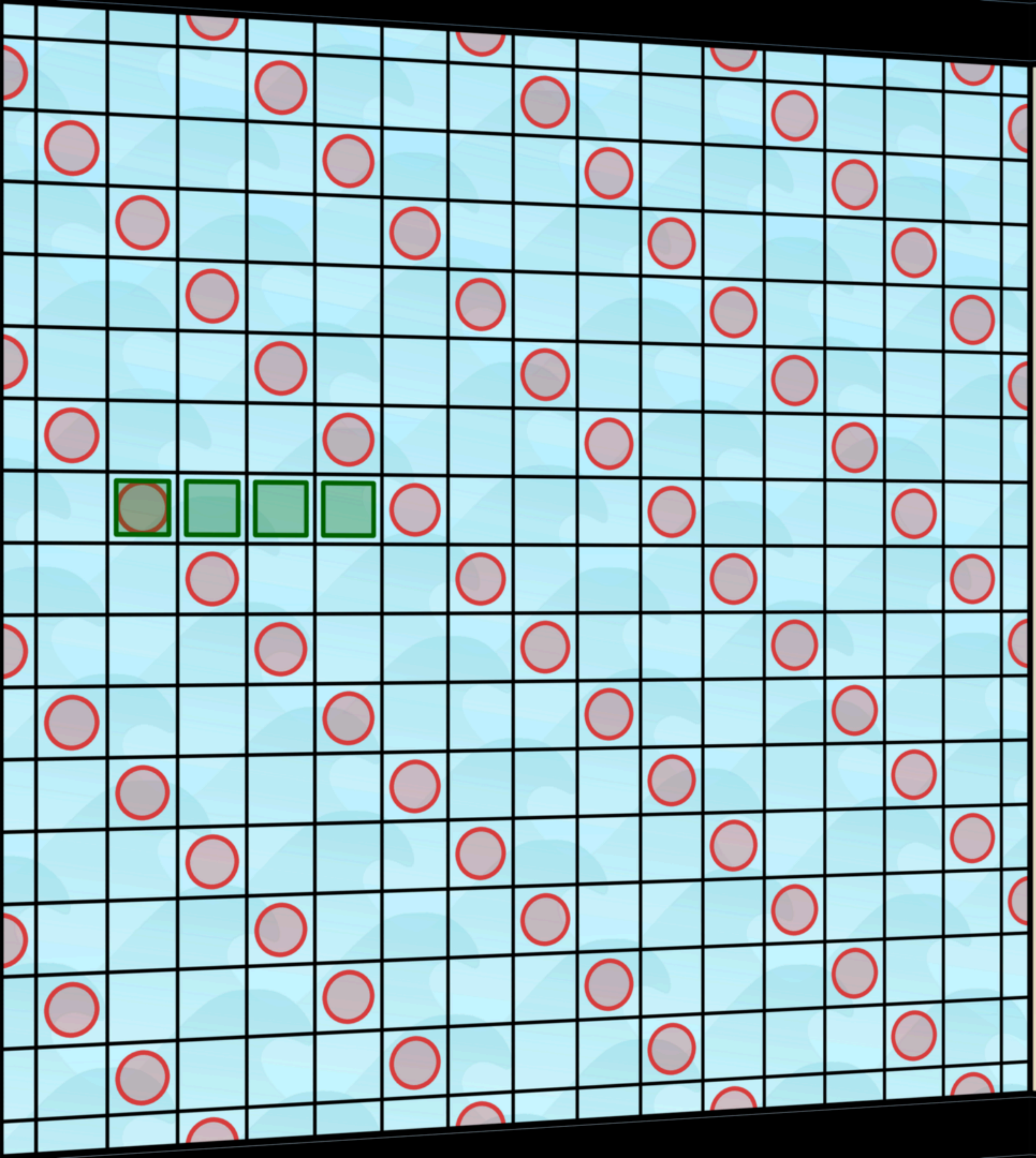
Now, it is easy to see that this pattern is optimal.



Let's consider battleships on an *infinite* board.

What is the best pattern to find S={,}?

Now, it is easy to see that this pattern is optimal.

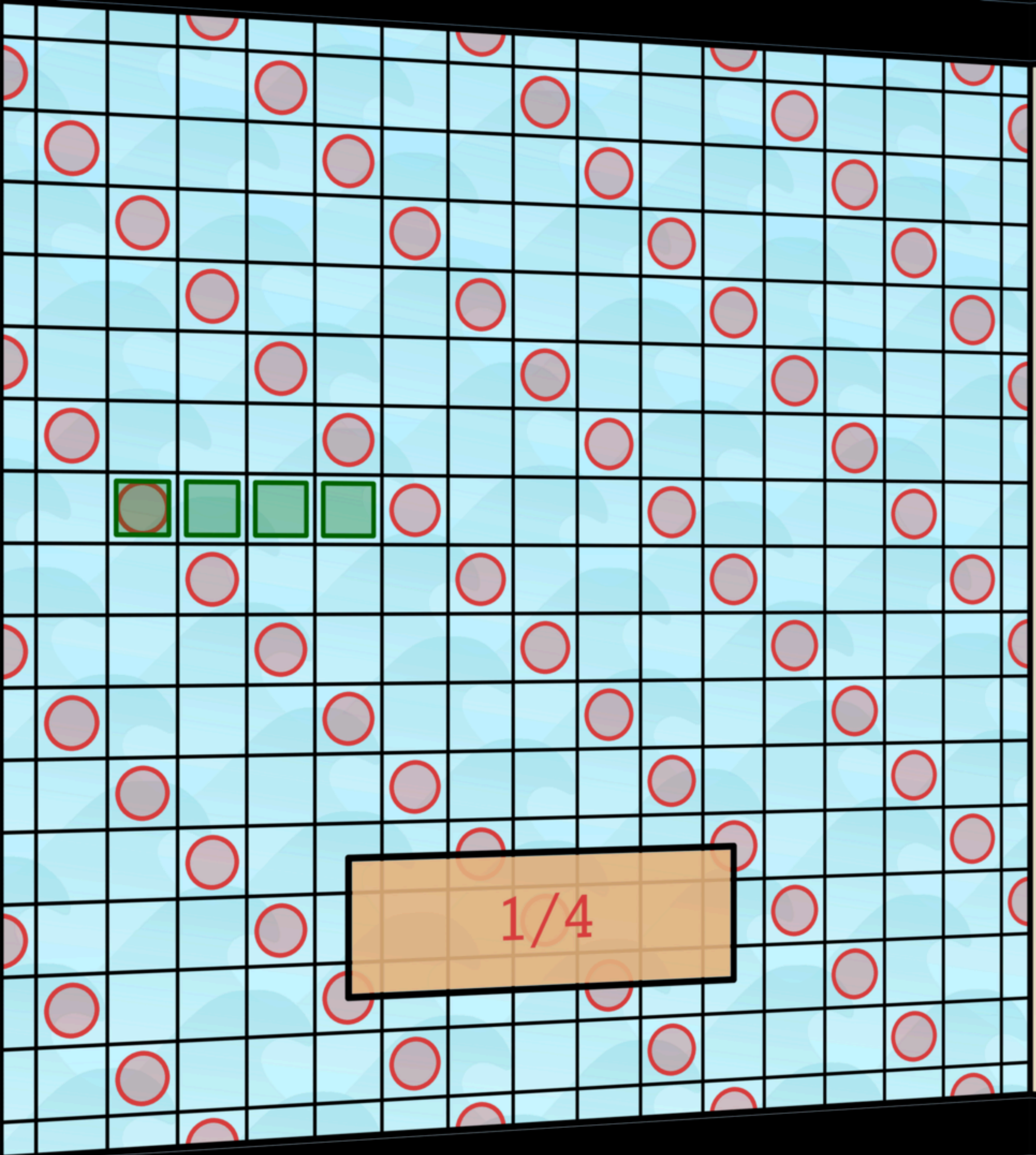


Let's consider battleships on an *infinite* board.

What is the best pattern to find S={,}?

Now, it is easy to see that this pattern is optimal.

The *density* of a solution is the fraction of filled cells.

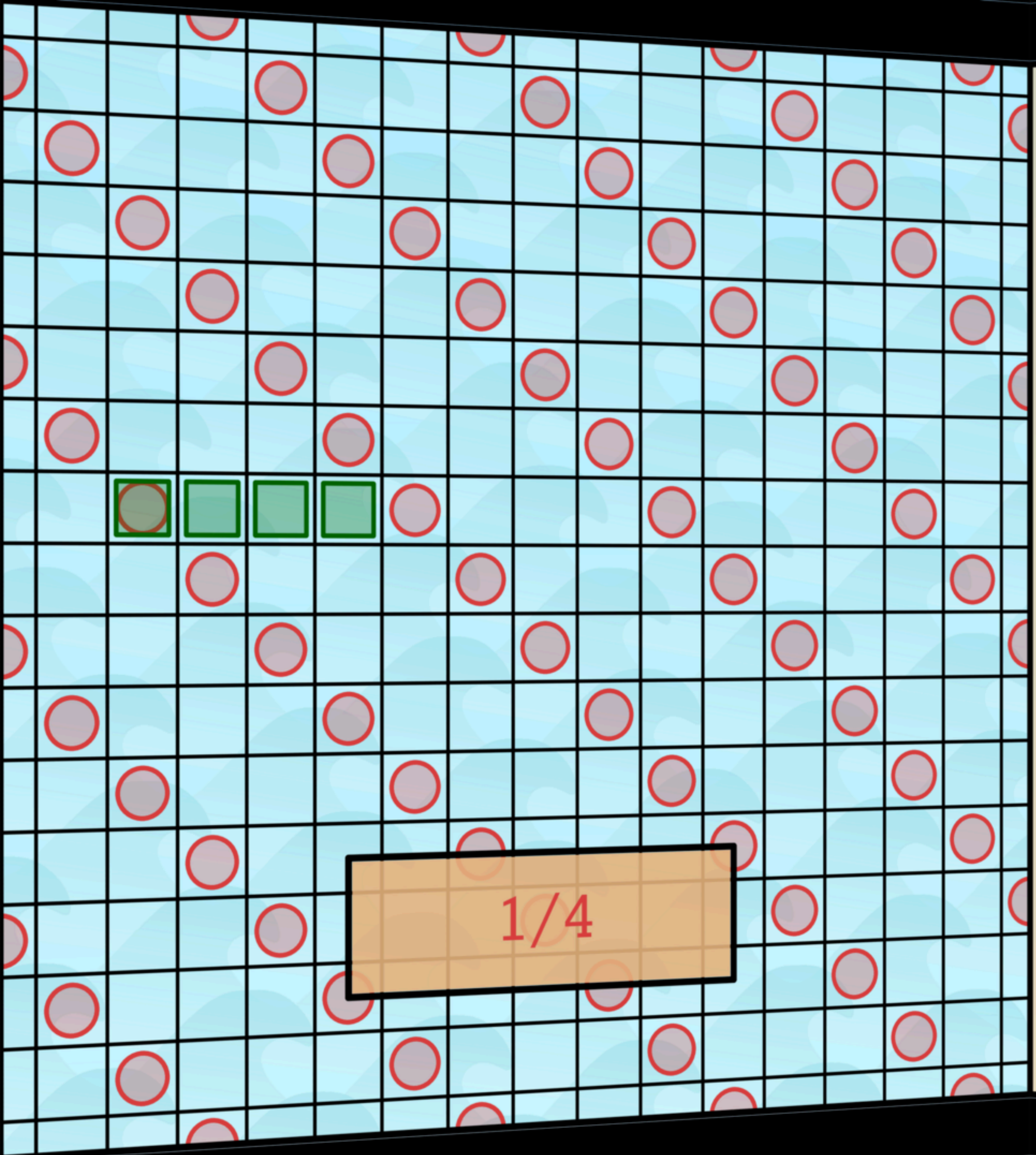


Let's consider battleships on an *infinite* board.

What is the best pattern to find S={,}?

Now, it is easy to see that this pattern is optimal.

The *density* of a solution is the fraction of filled cells.



Let's consider battleships on an *infinite* board.

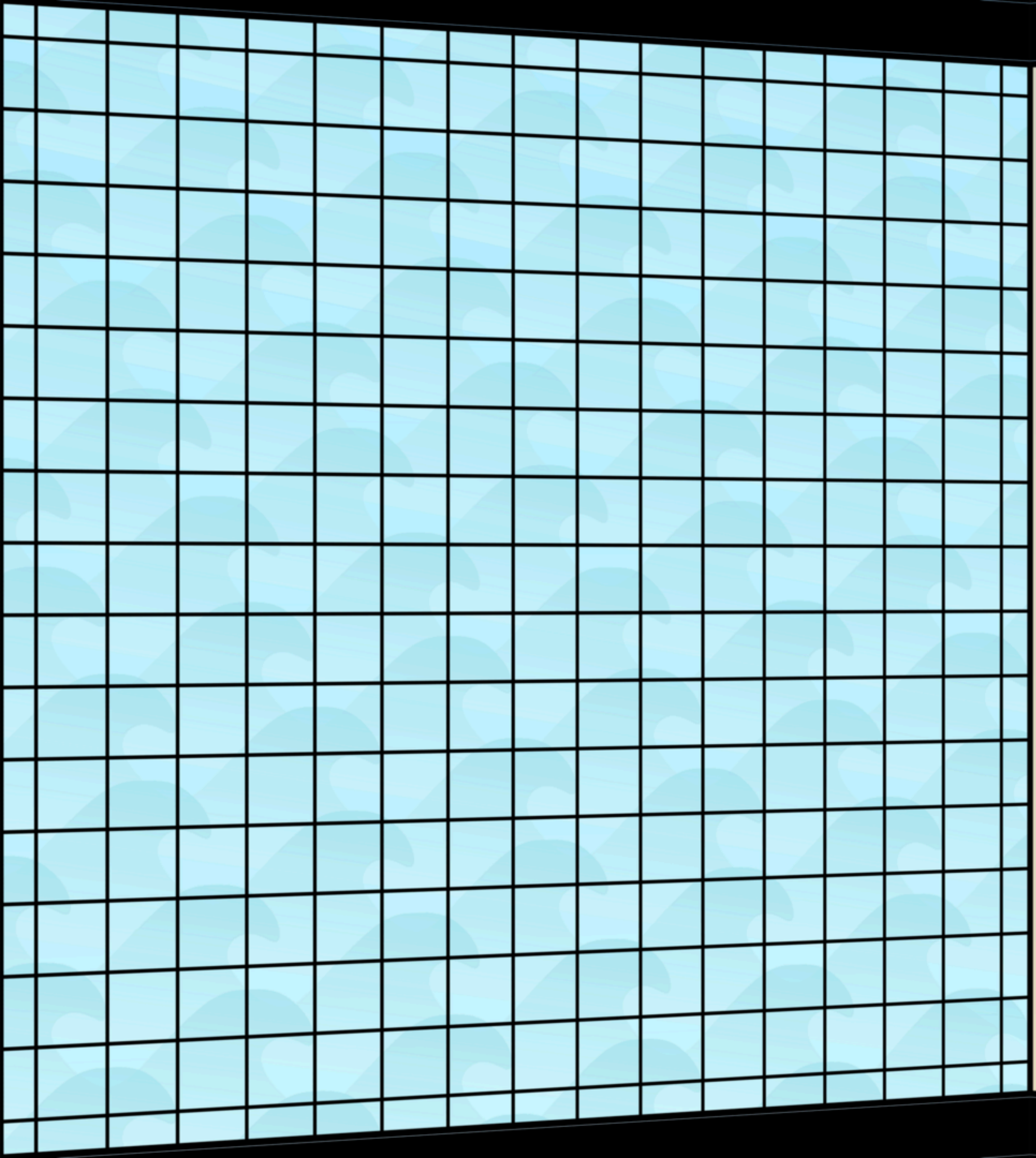
What is the best pattern to find S={,}?

Now, it is easy to see that this pattern is optimal.

The *density* of a solution is the fraction of filled cells.

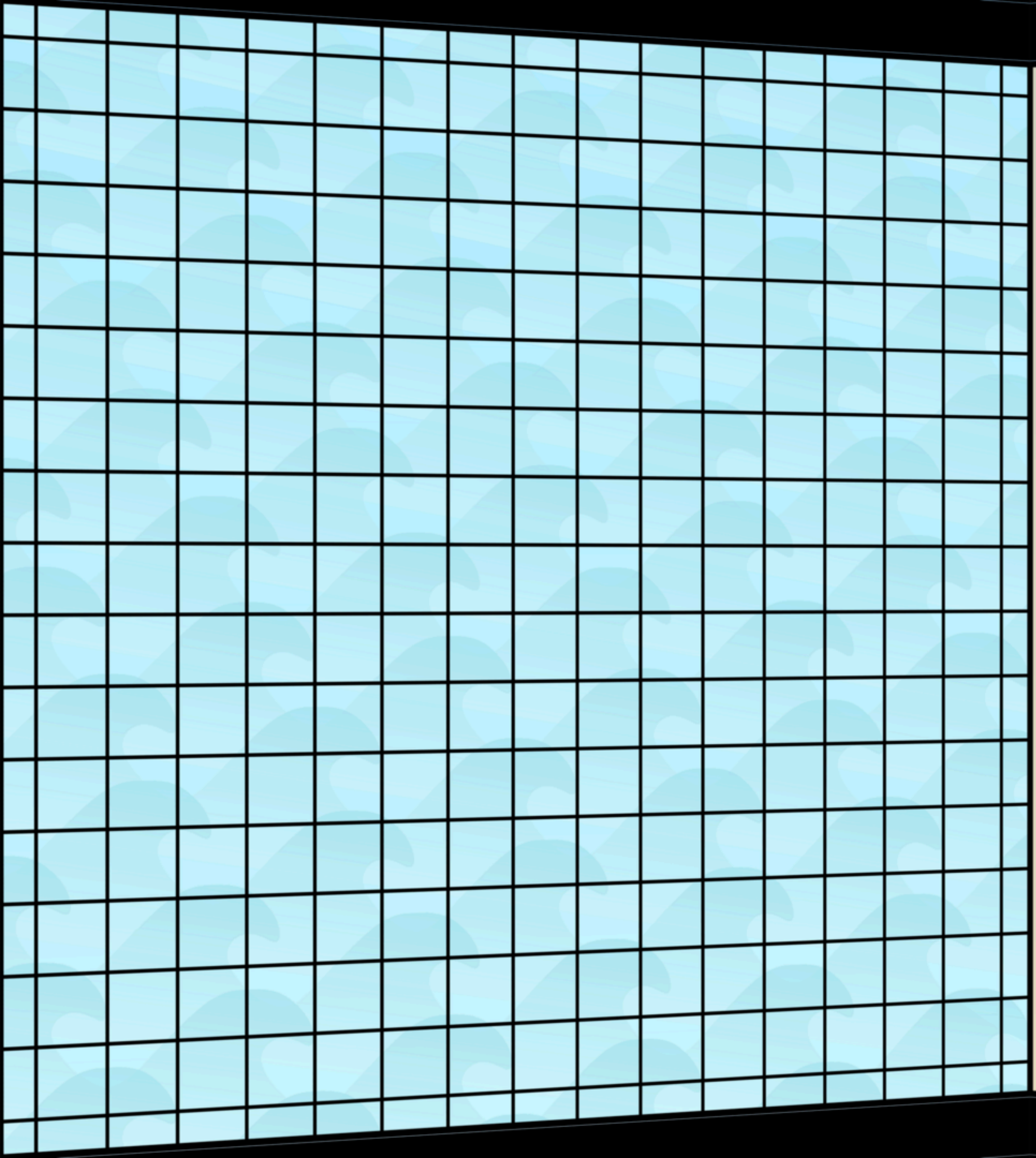
Note that the *same* pattern works for S={}!

Does rotation matter?



Does rotation matter?

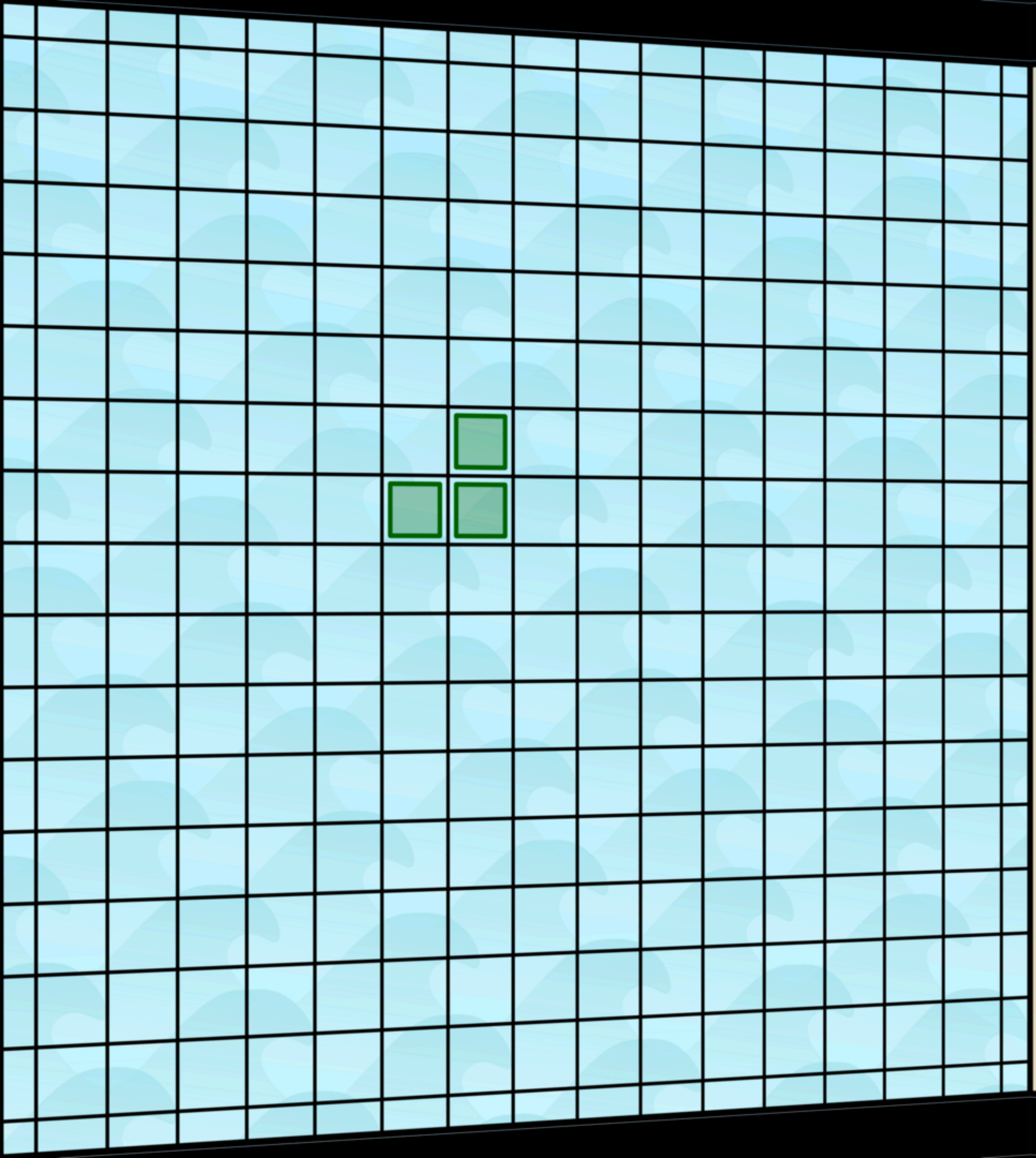
For traditional ships ($1 \times k$ rectangles) not.



Does rotation matter?

For traditional ships (1×k rectangles) not.

What about other shapes?

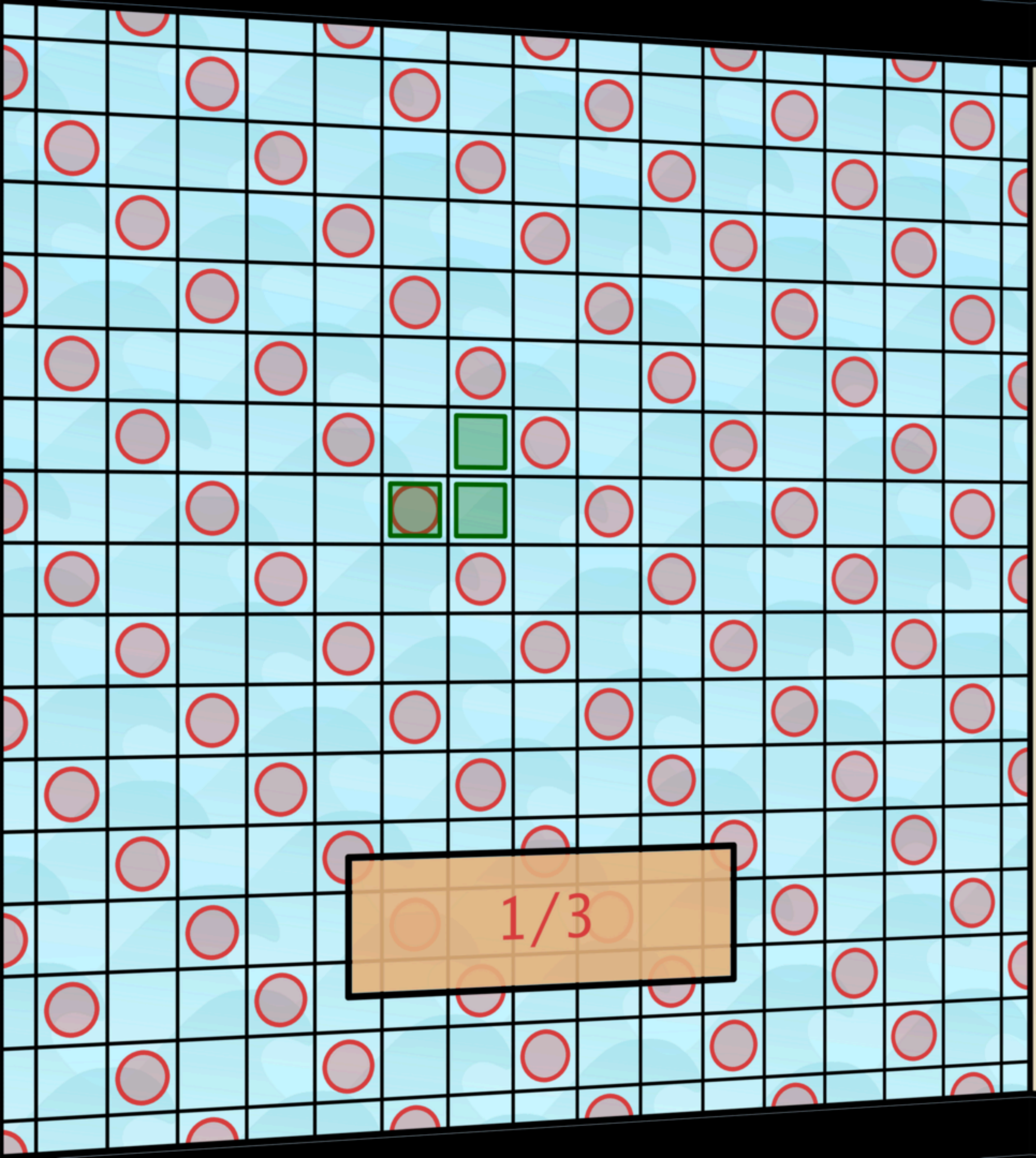


Does rotation matter?

For traditional ships (1×k rectangles) not.

What about other shapes?

Say, the L-triomino S={}.

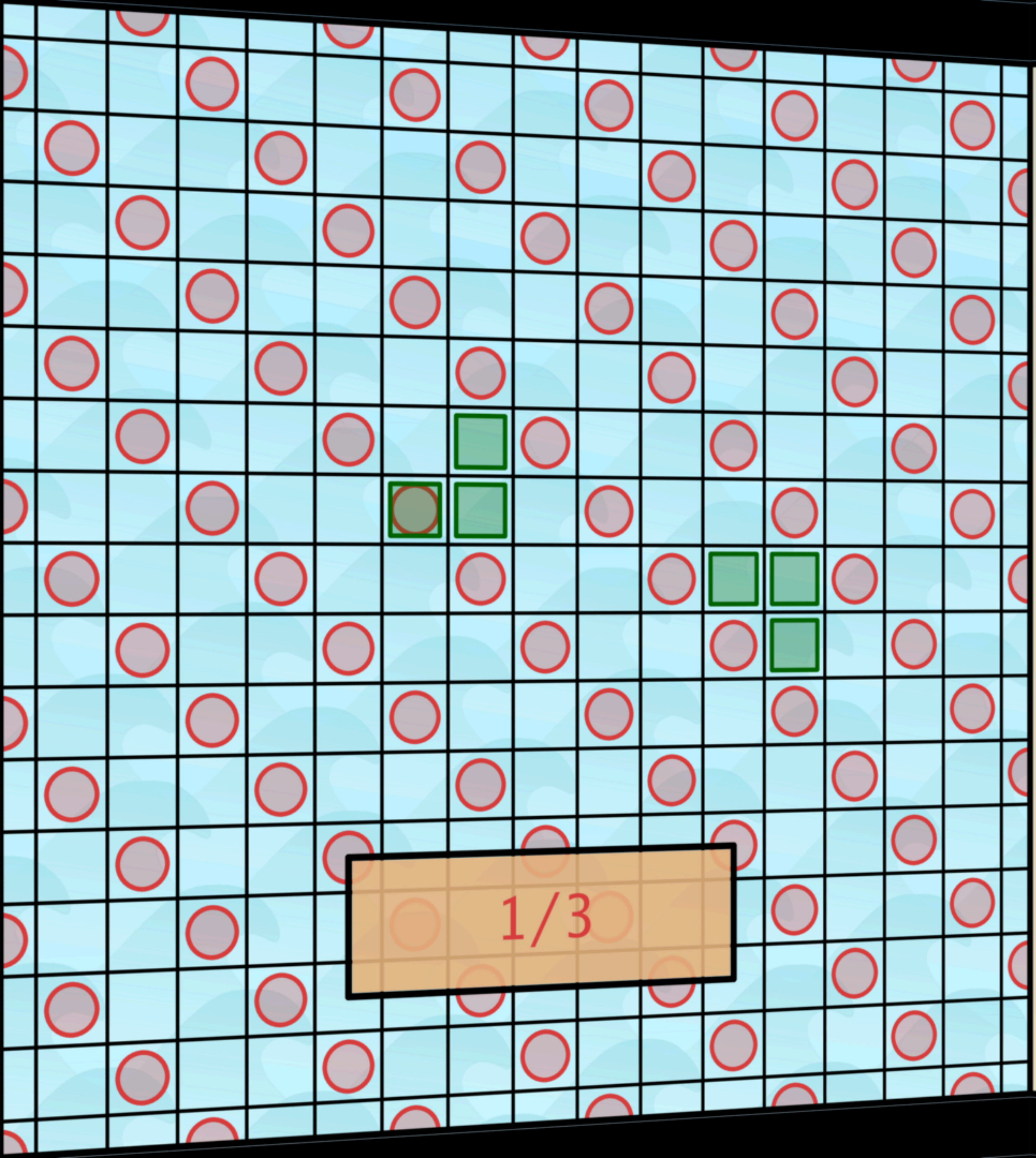


Does rotation matter?

For traditional ships (1×k rectangles) not.

What about other shapes?

Say, the L-triomino S={}.

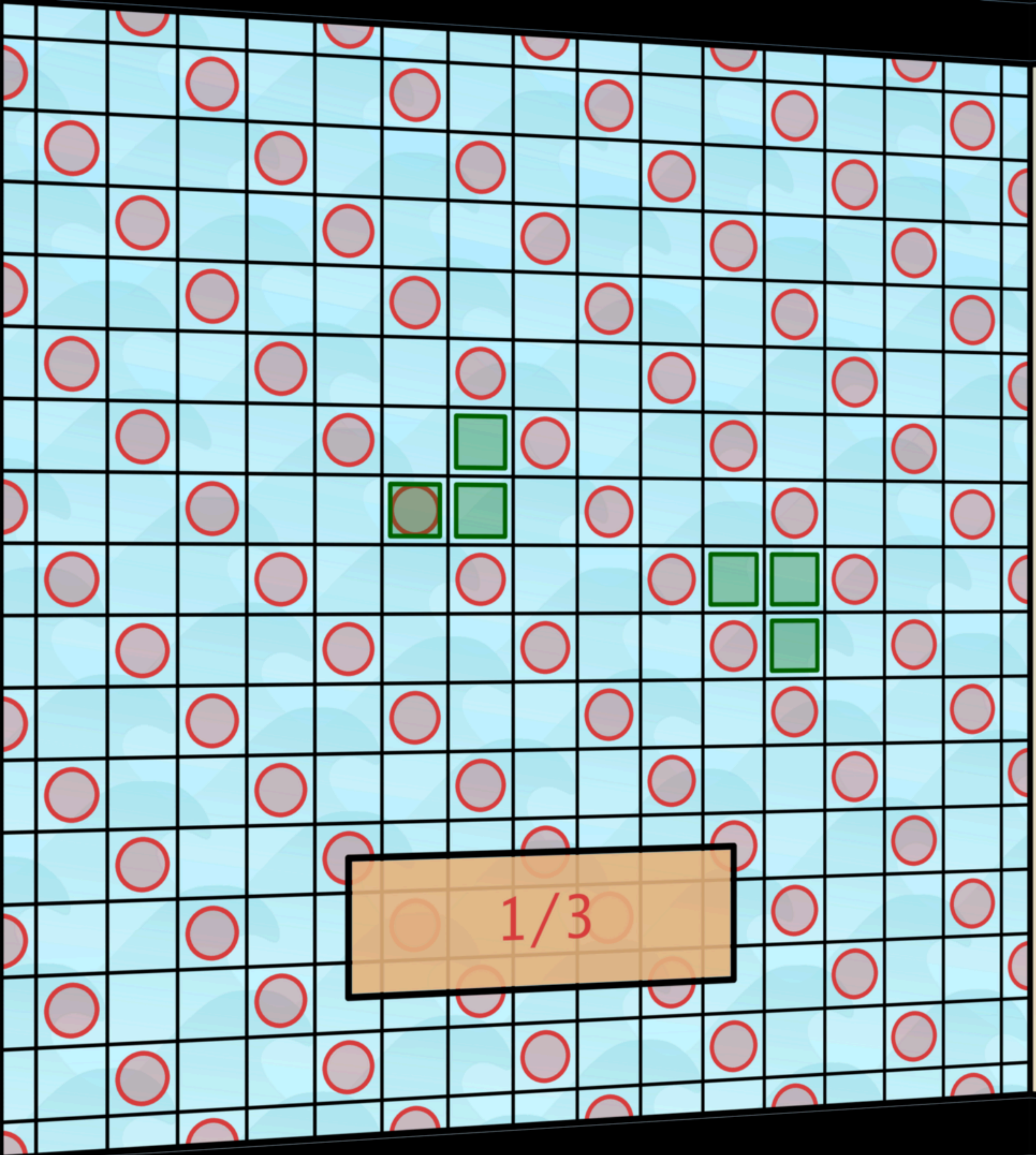


Does rotation matter?

For traditional ships ($1\times k$ rectangles) not.

What about other shapes?

Say, the L-triomino S={}.



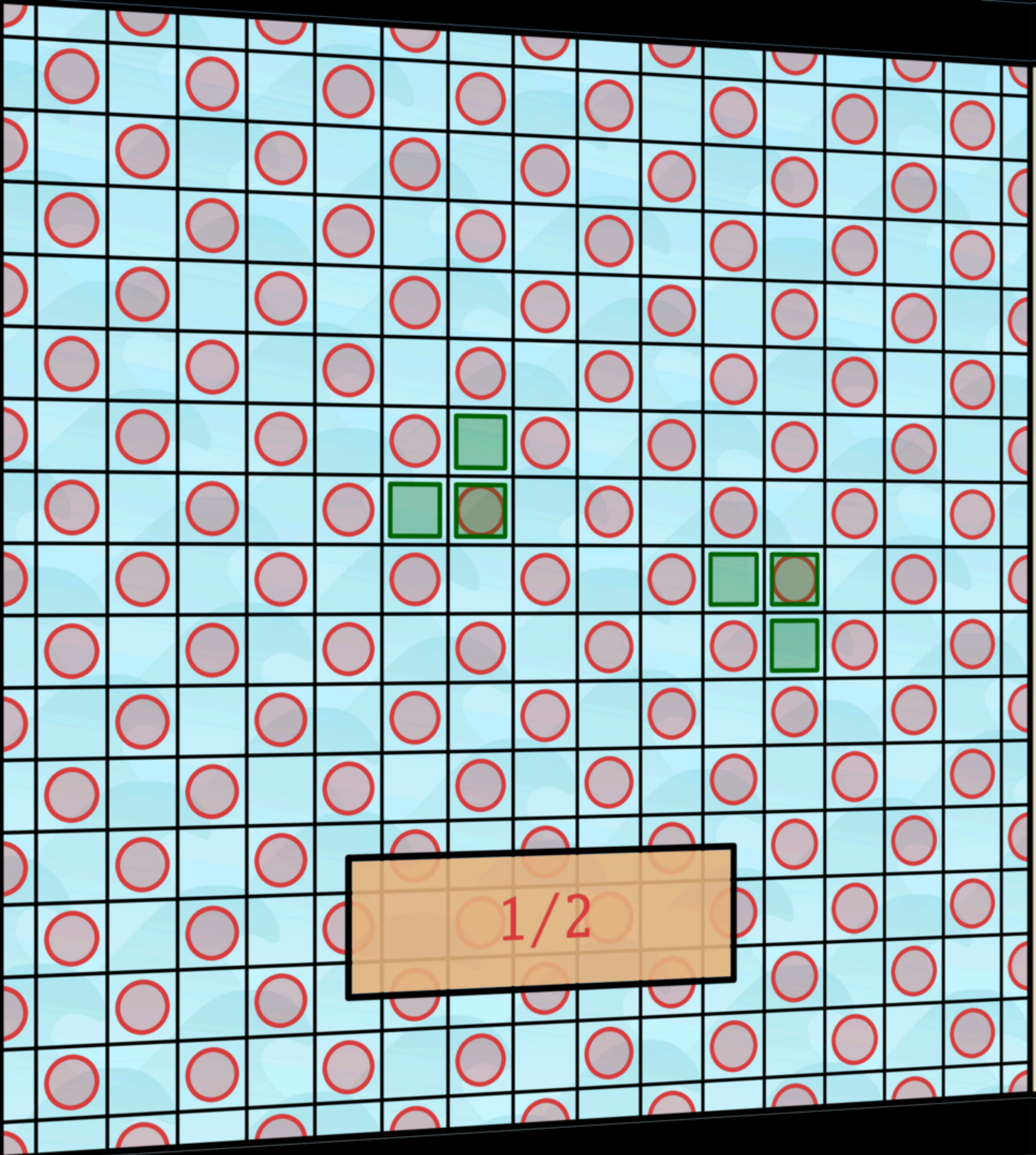
Does rotation matter?

For traditional ships ($1 \times k$ rectangles) not.

What about other shapes?

Say, the L-triomino S={}.

Not the same as S={,}!



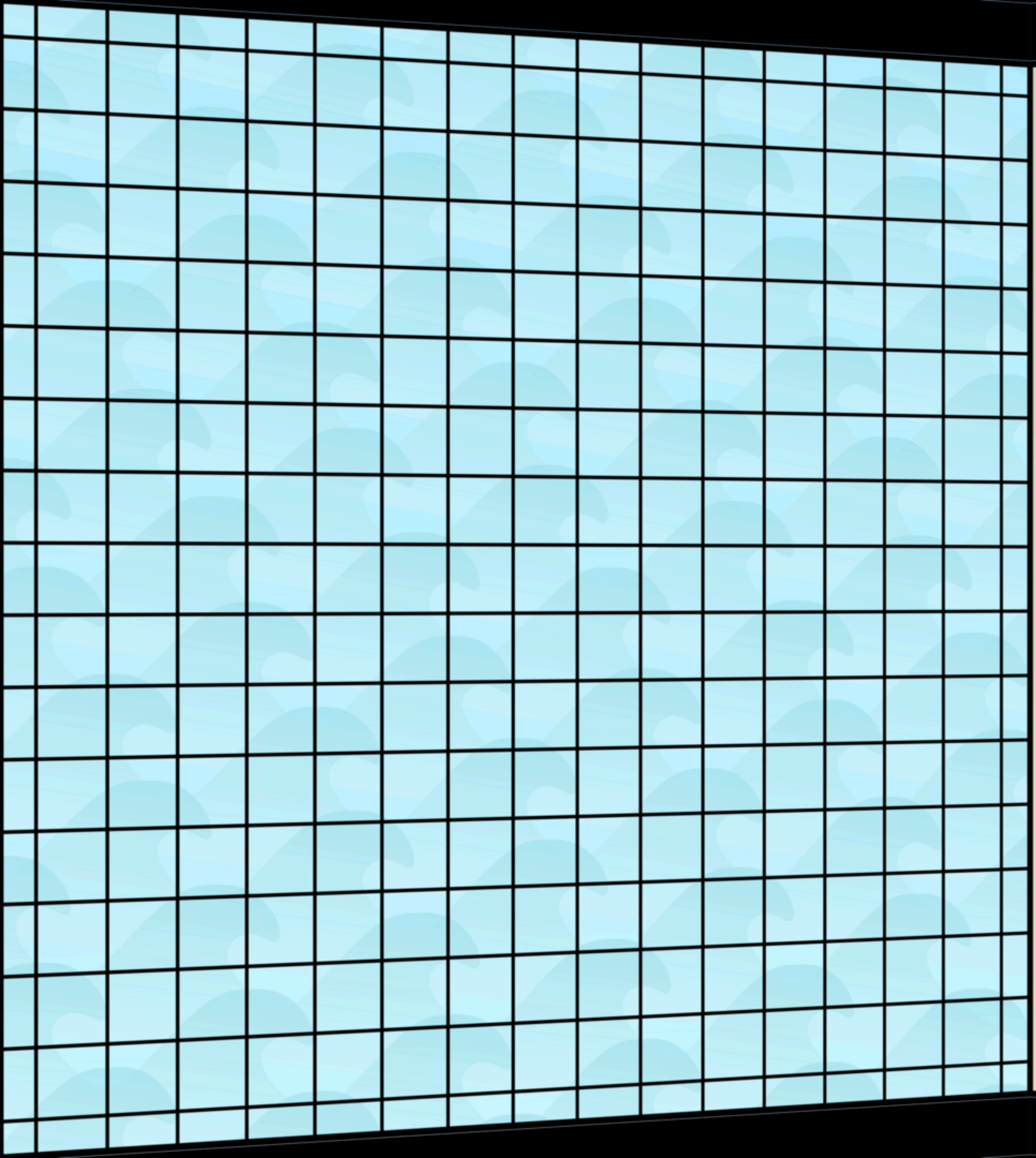
Does rotation matter?

For traditional ships ($1 \times k$ rectangles) not.

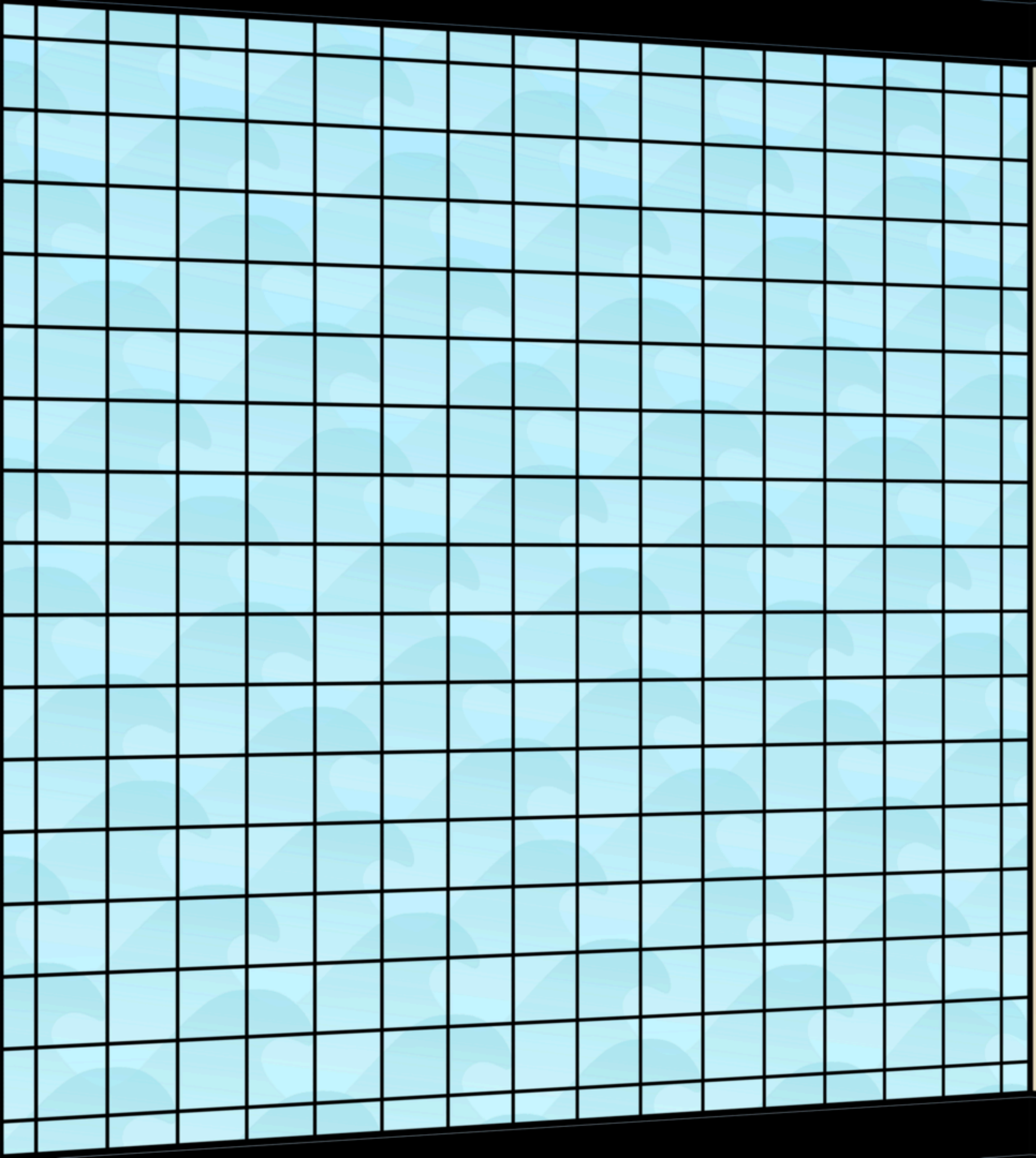
What about other shapes?

Say, the L-triomino S={}.

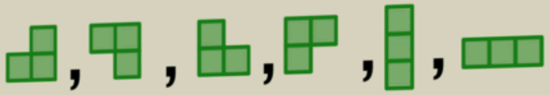
Not the same as S={,}!

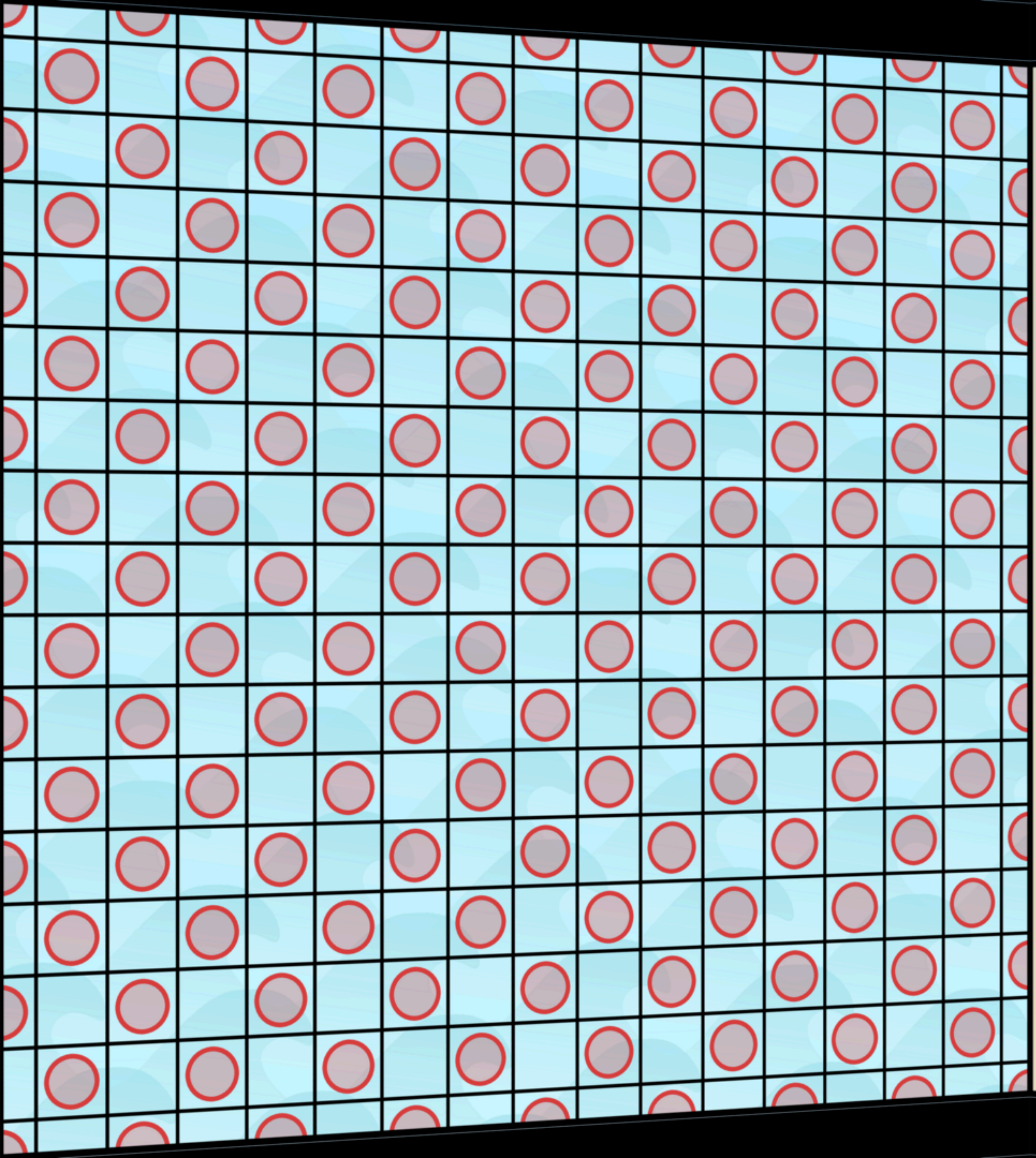


Suppose that all we know is the *size* of the ship.

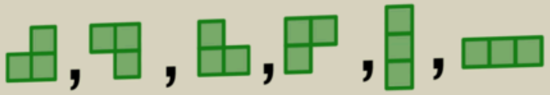


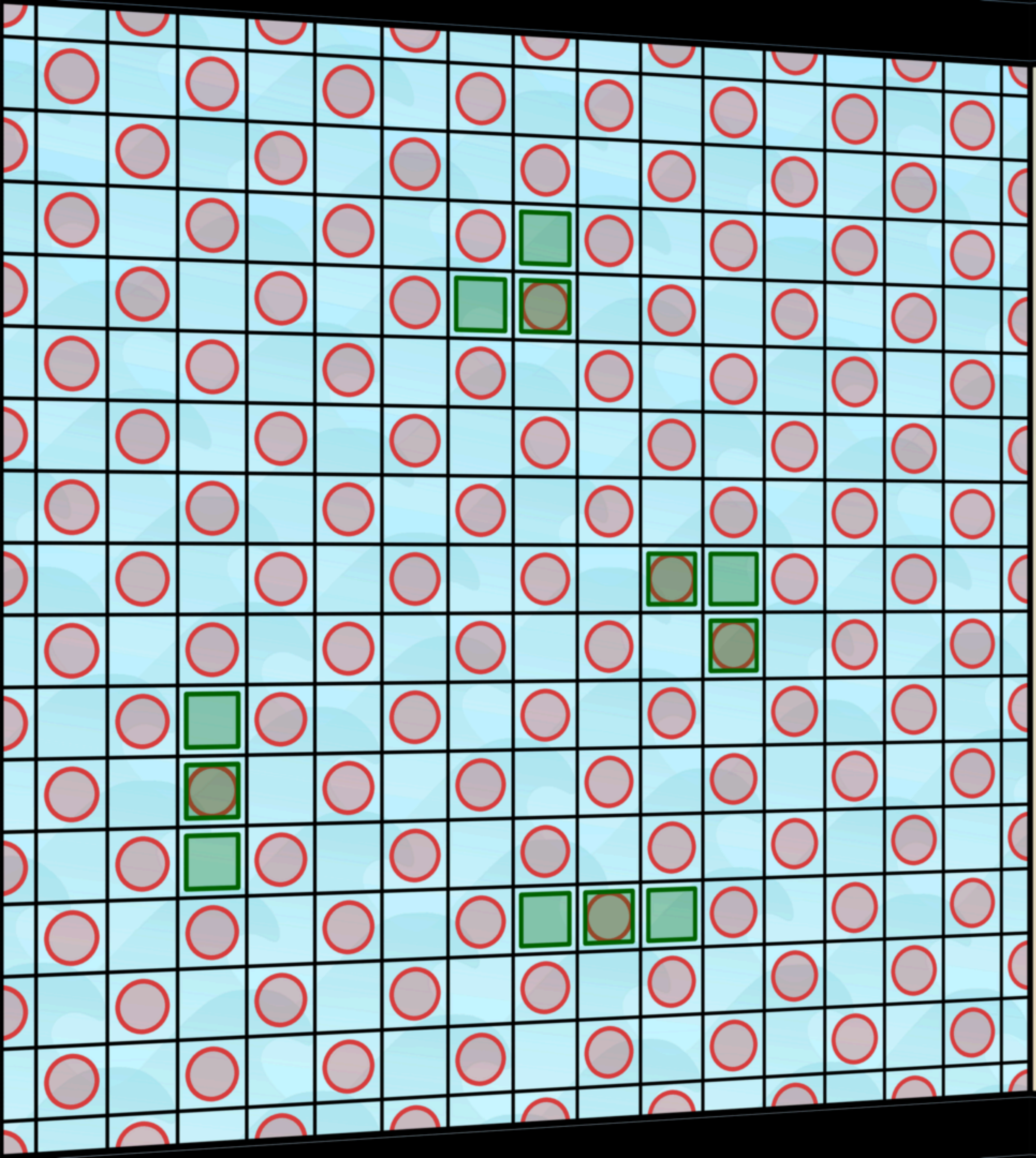
Suppose that all we know is the *size* of the ship.

S={  }.

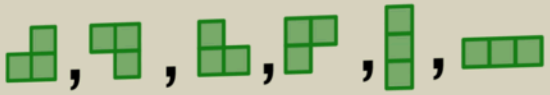


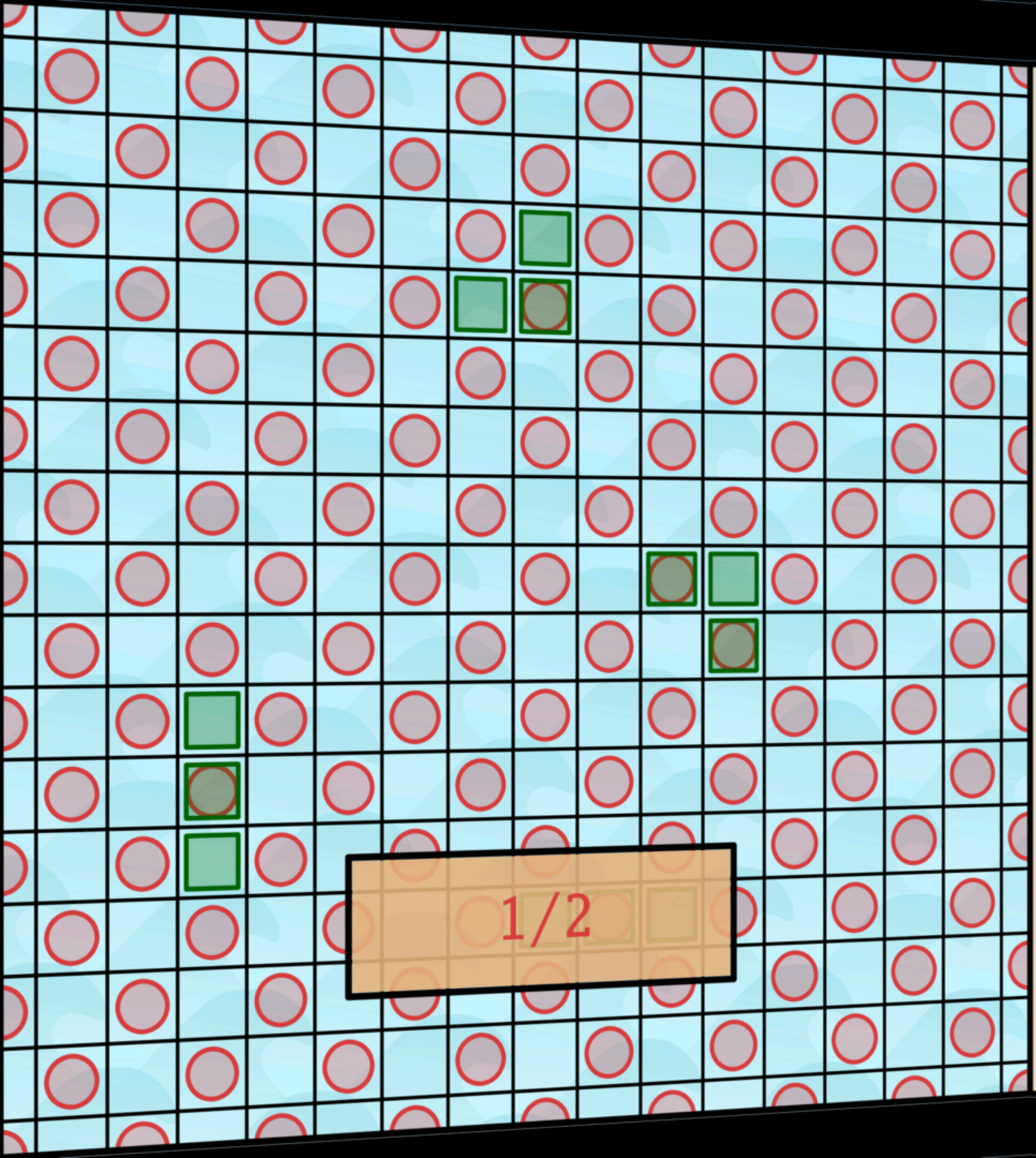
Suppose that all we know is the *size* of the ship.

S={  }.

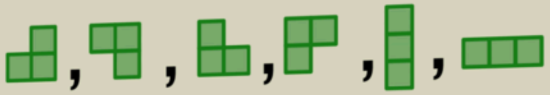


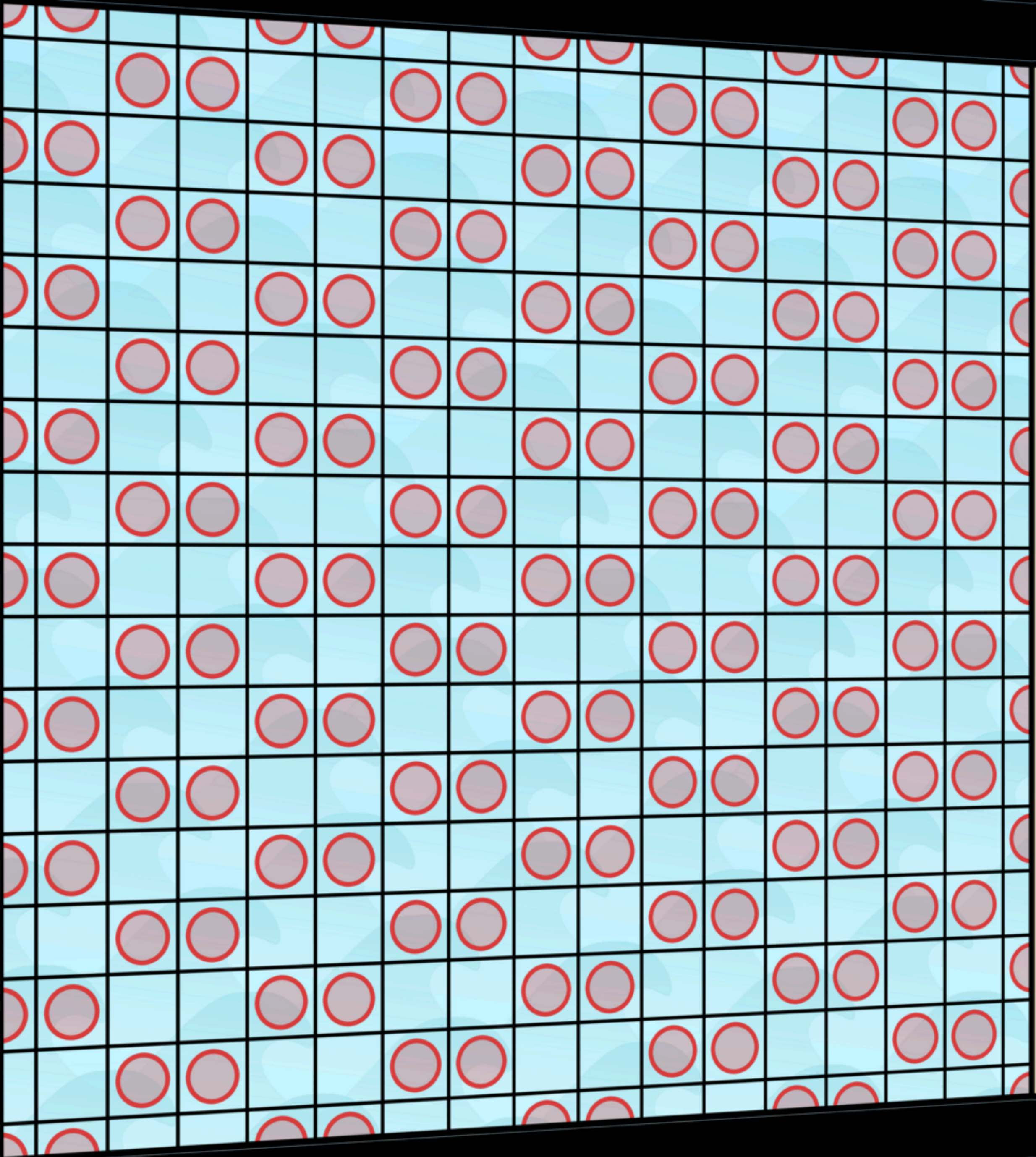
Suppose that all we know is the *size* of the ship.

S={  }.



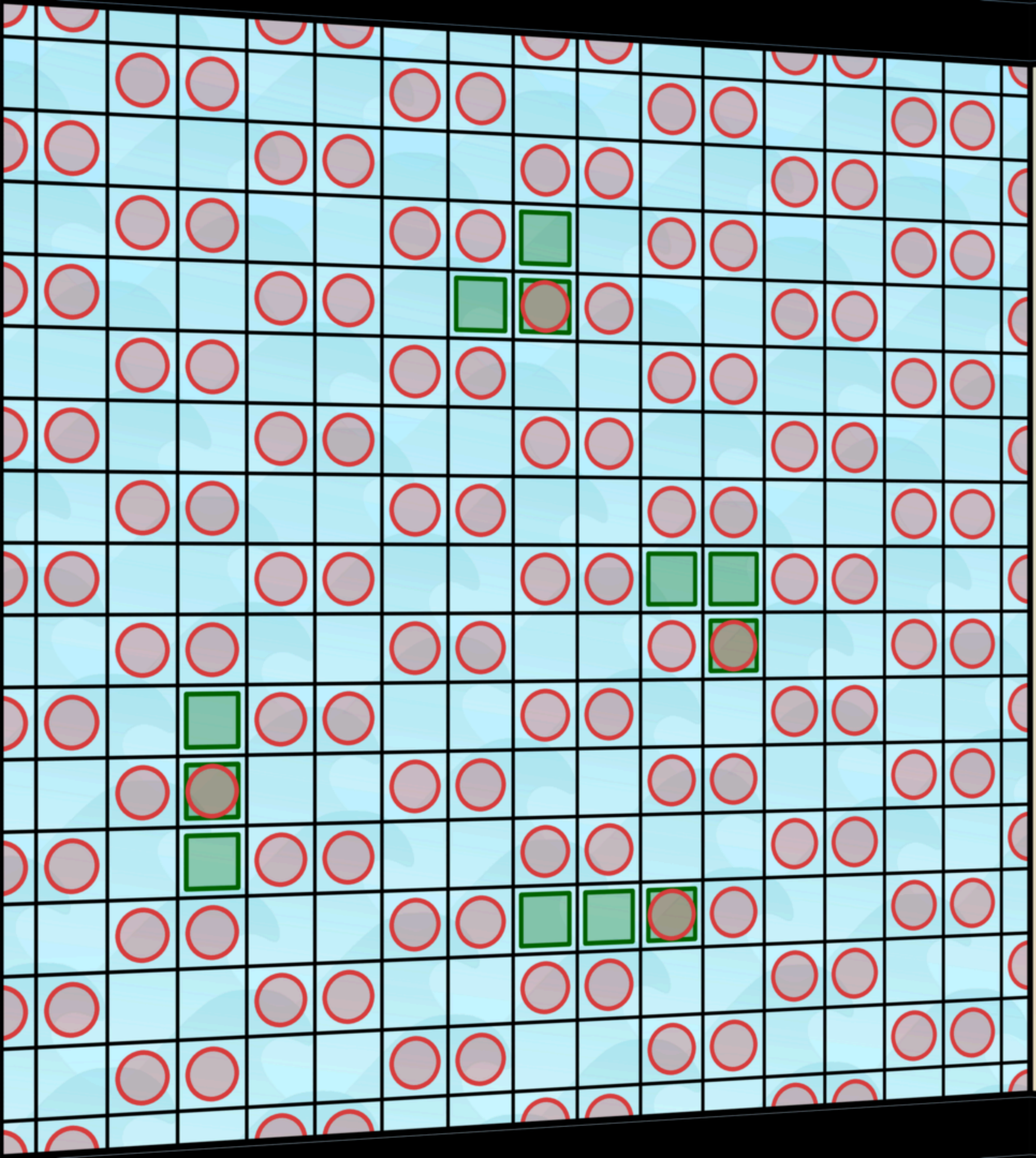
Suppose that all we know is the *size* of the ship.

S={  }.



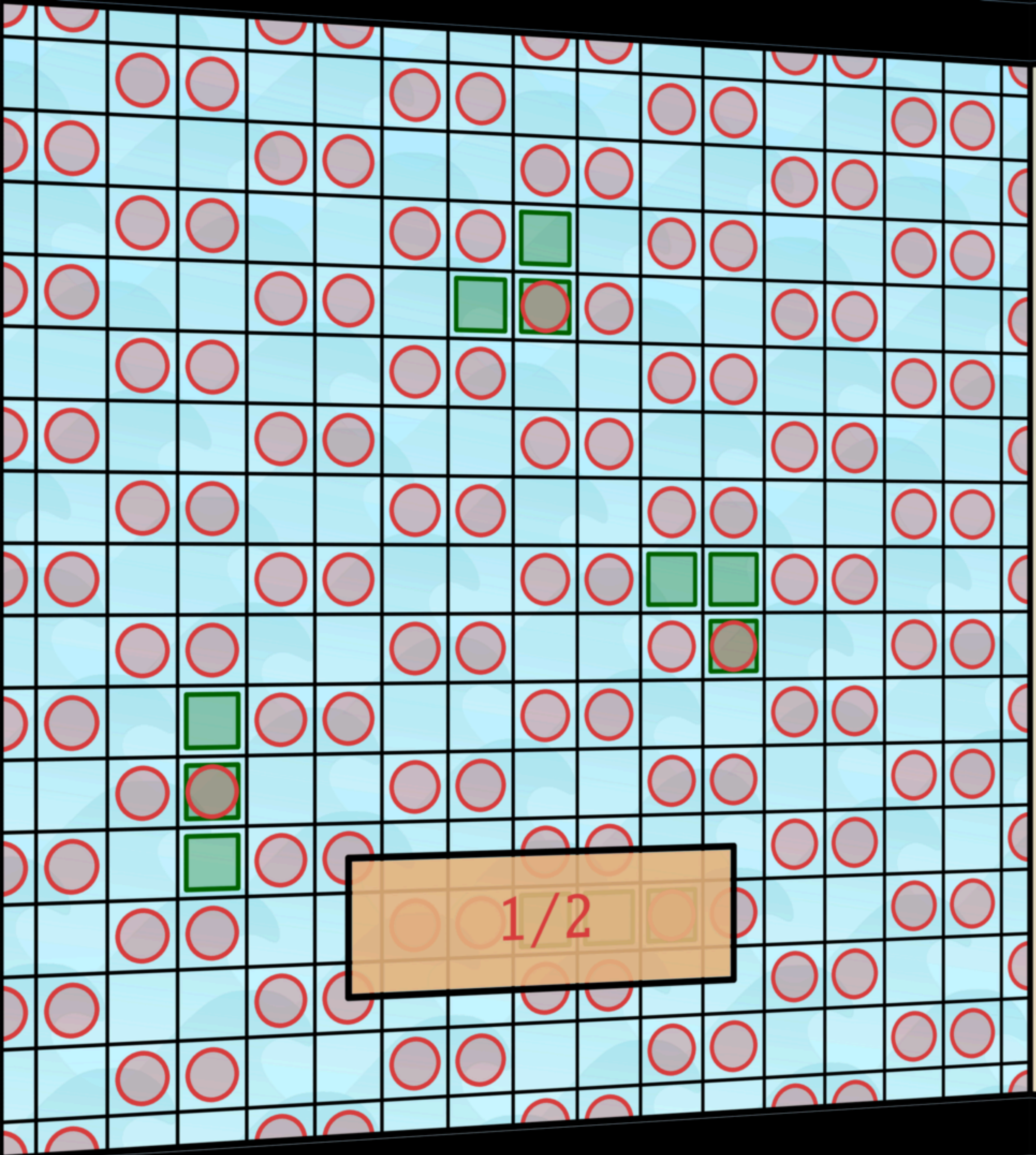
Suppose that all we know is the *size* of the ship.

S={, , , , , }.

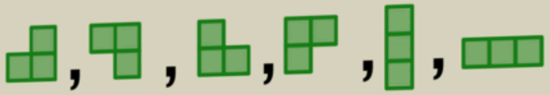


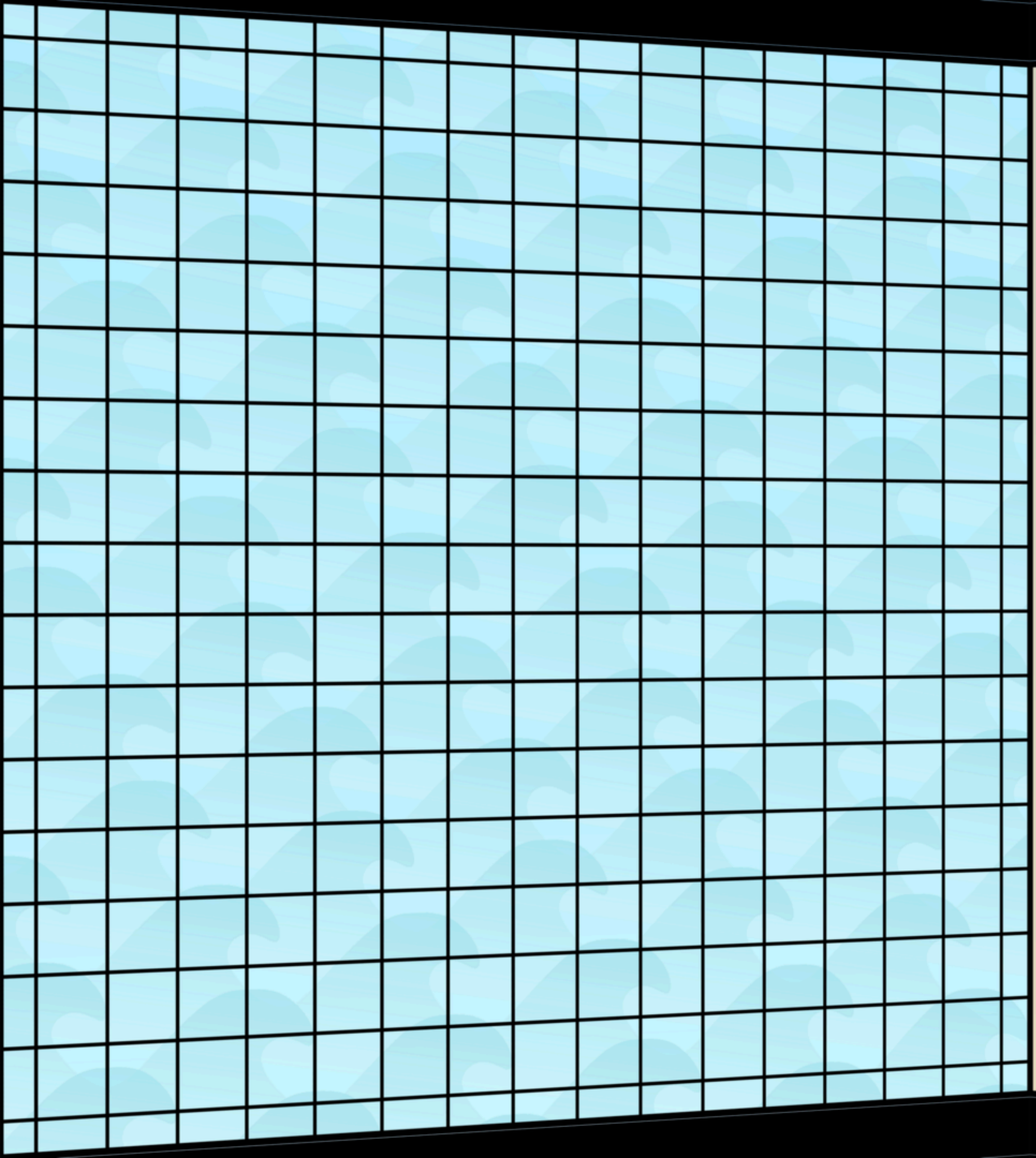
Suppose that all we know is the *size* of the ship.

S={, , , , , }.

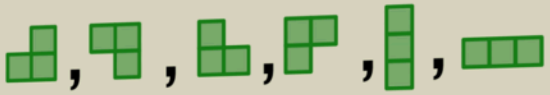


Suppose that all we know is the *size* of the ship.

S={}.

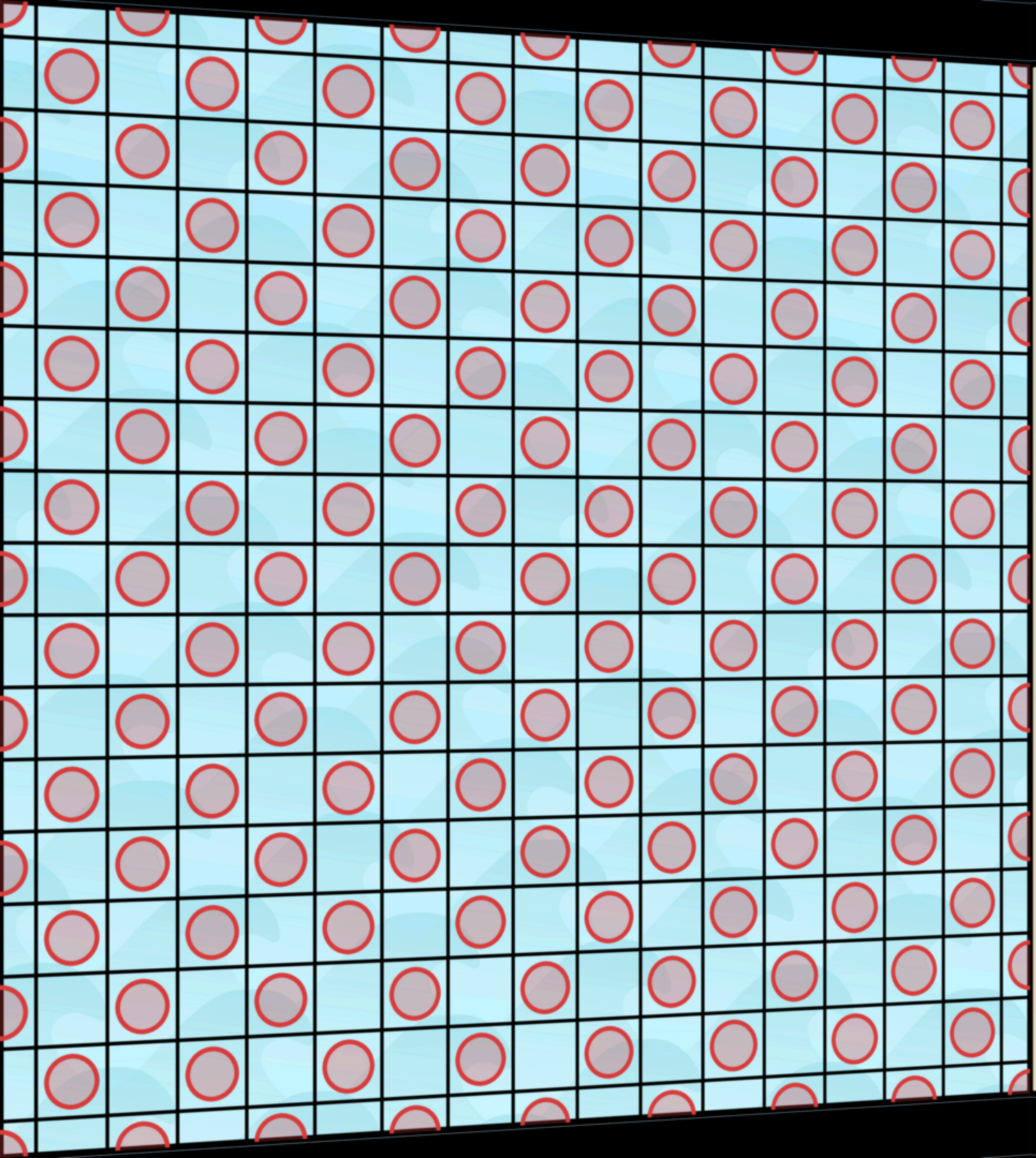


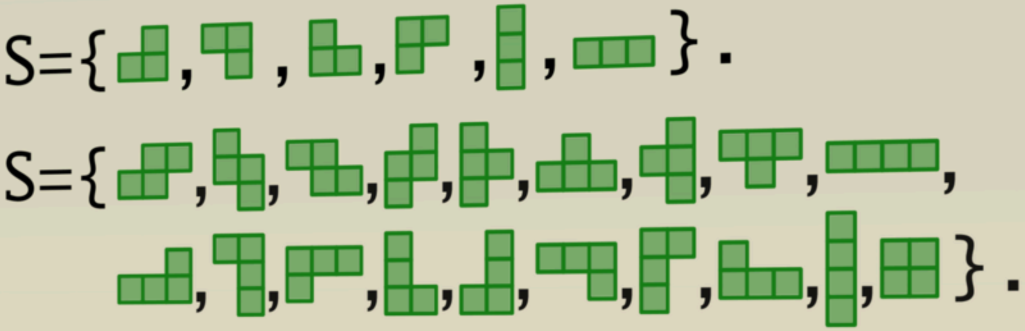
Suppose that all we know is the *size* of the ship.

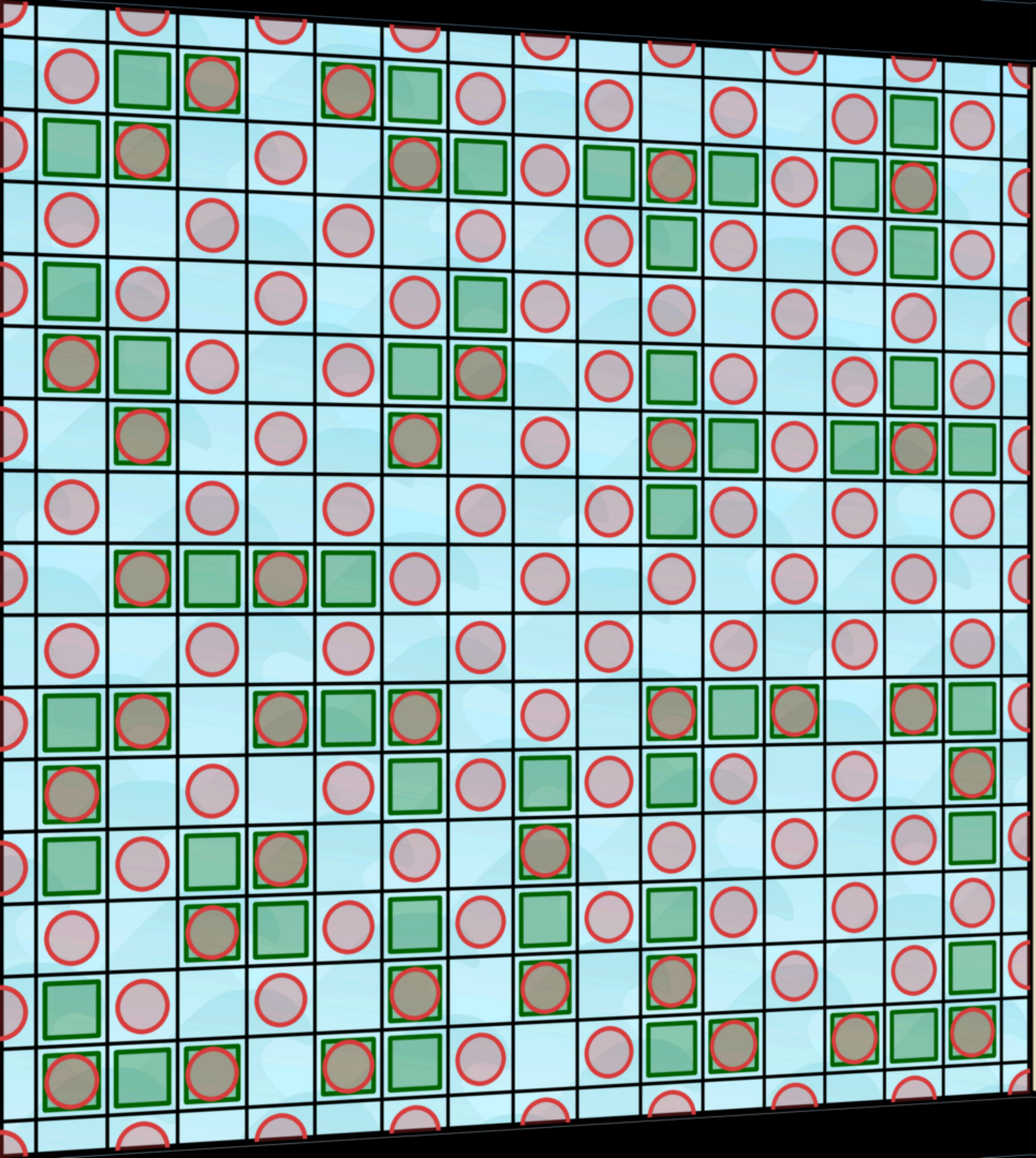
S={}.

S={}.

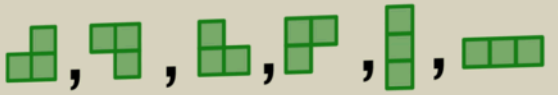
Suppose that all we know is the *size* of the ship.

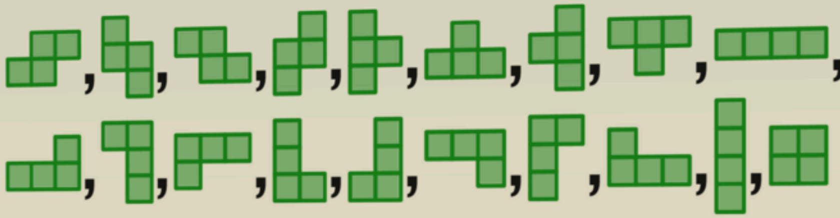


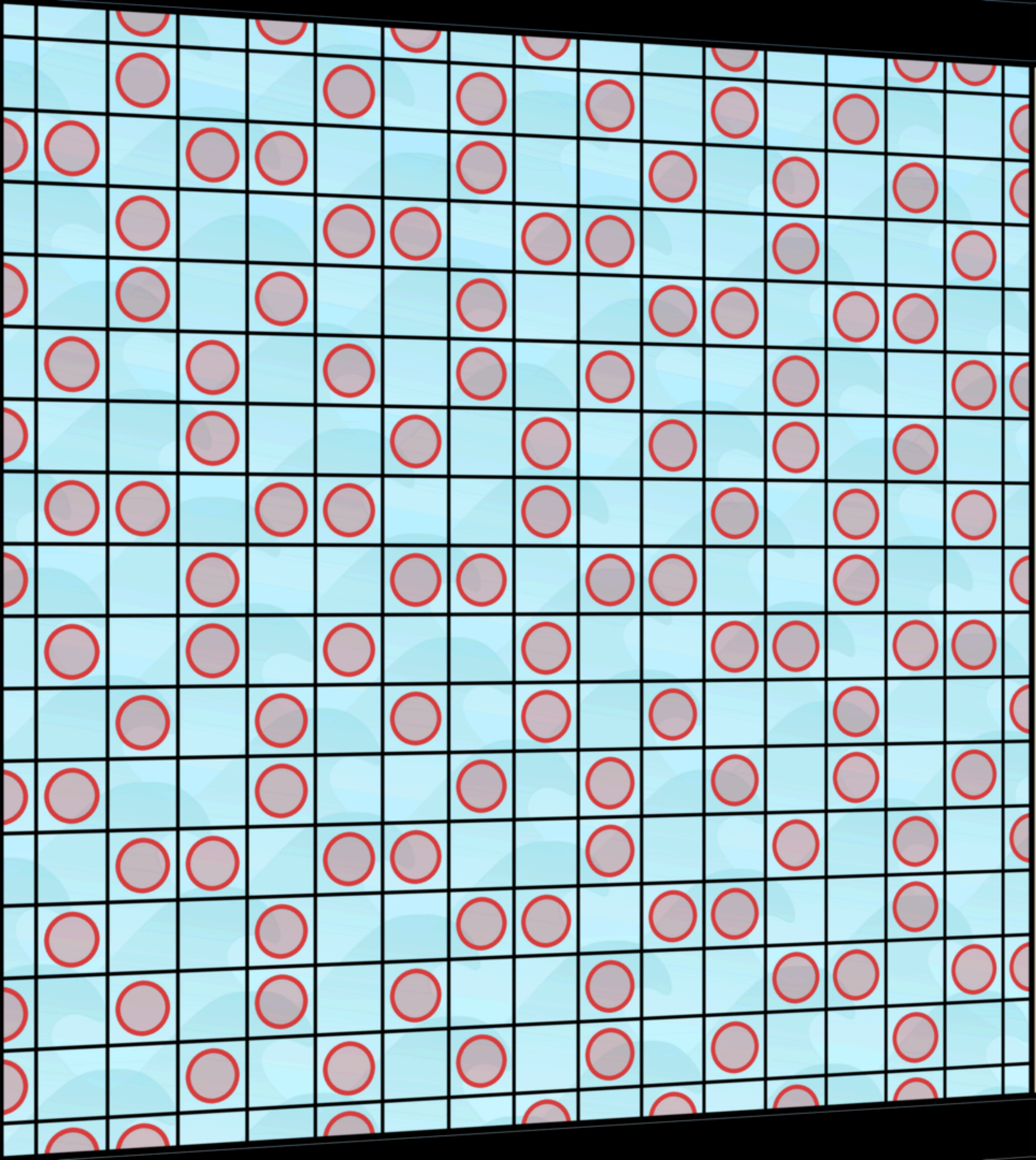




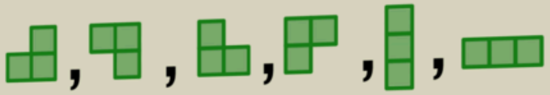
Suppose that all we know is the *size* of the ship.

S={}.

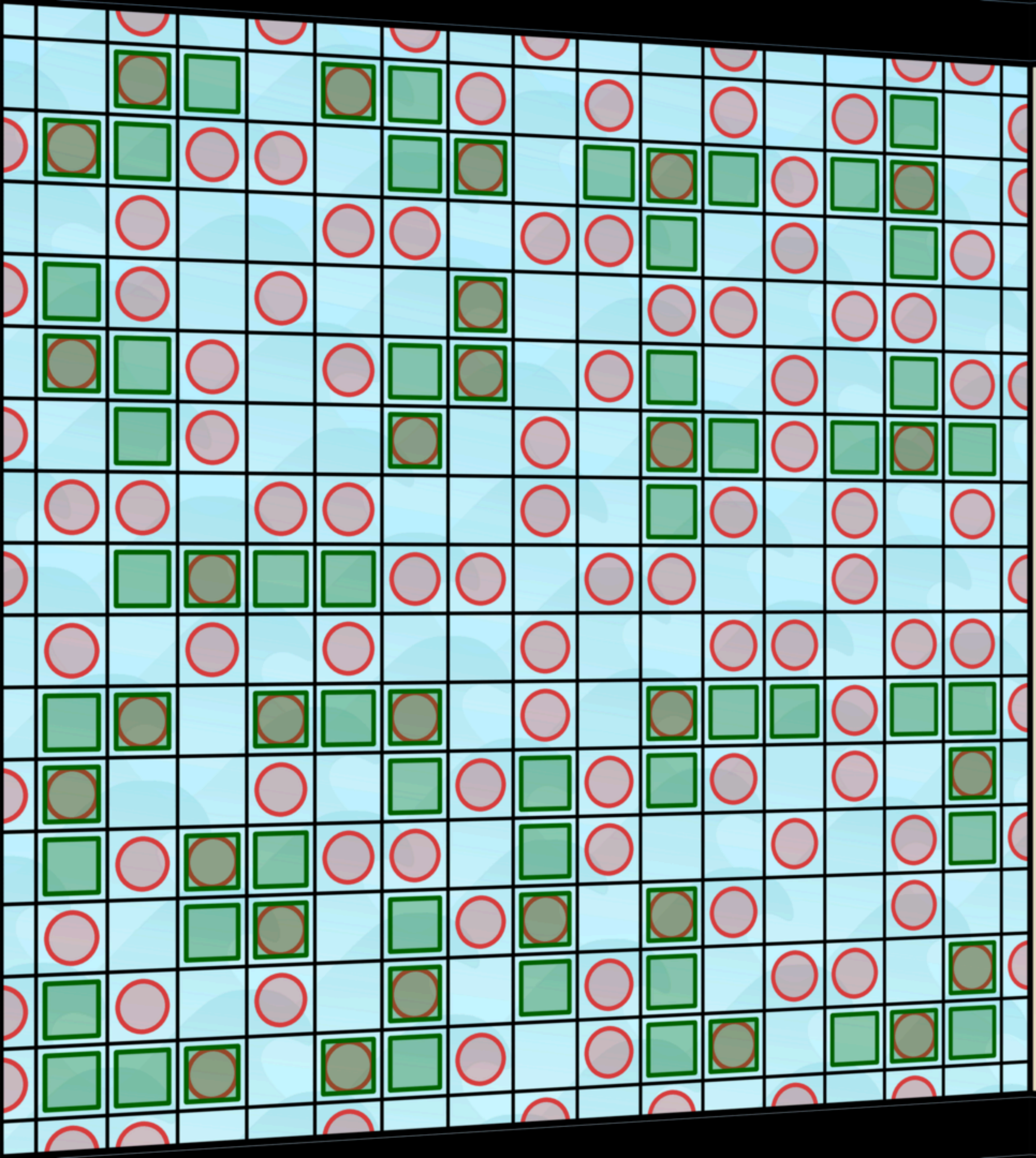
S={}.



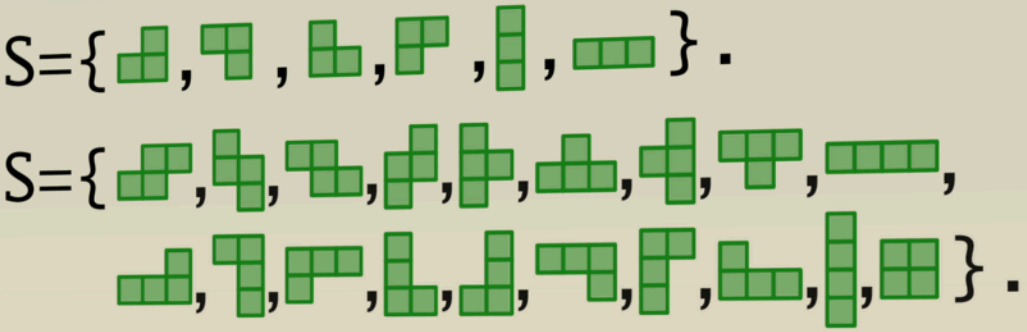
Suppose that all we know is the *size* of the ship.

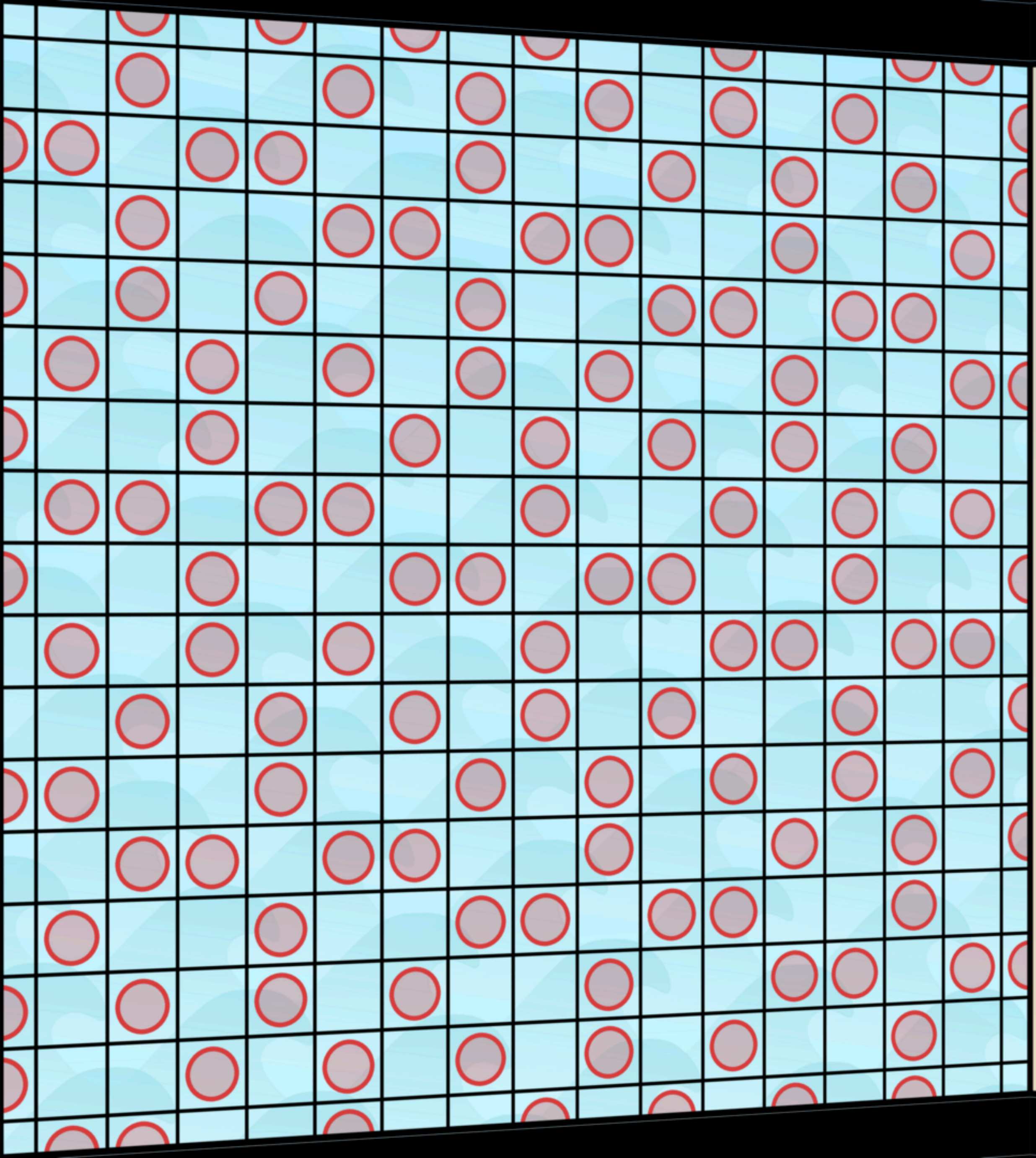
S={  }.

S={  }.

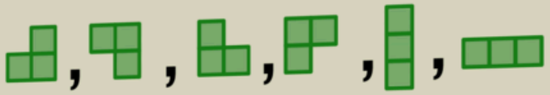


Suppose that all we know is the *size* of the ship.

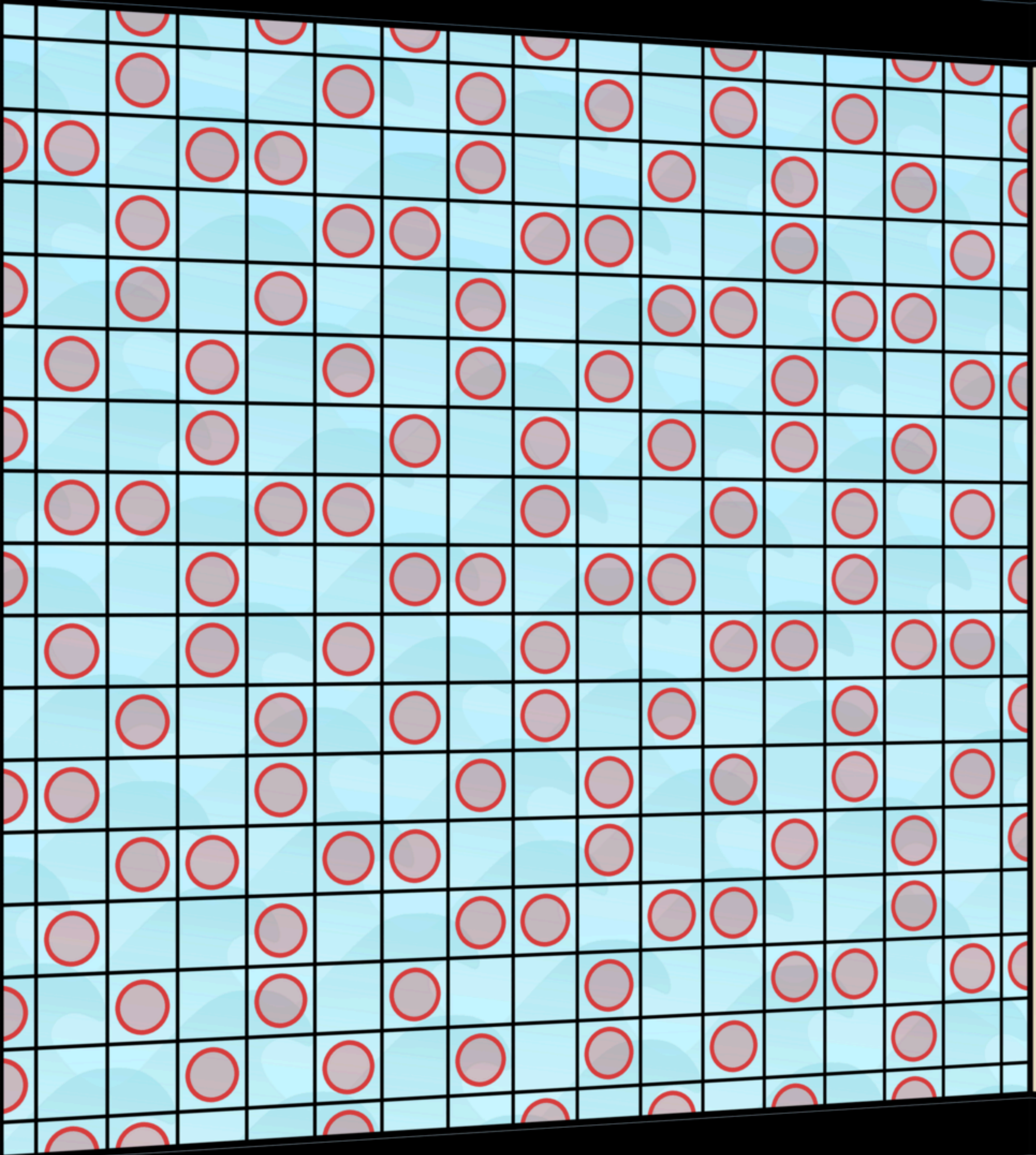




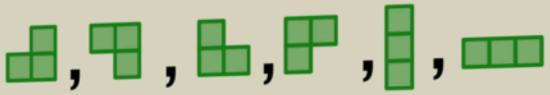
Suppose that all we know is the *size* of the ship.

S={  }.

S={  }.

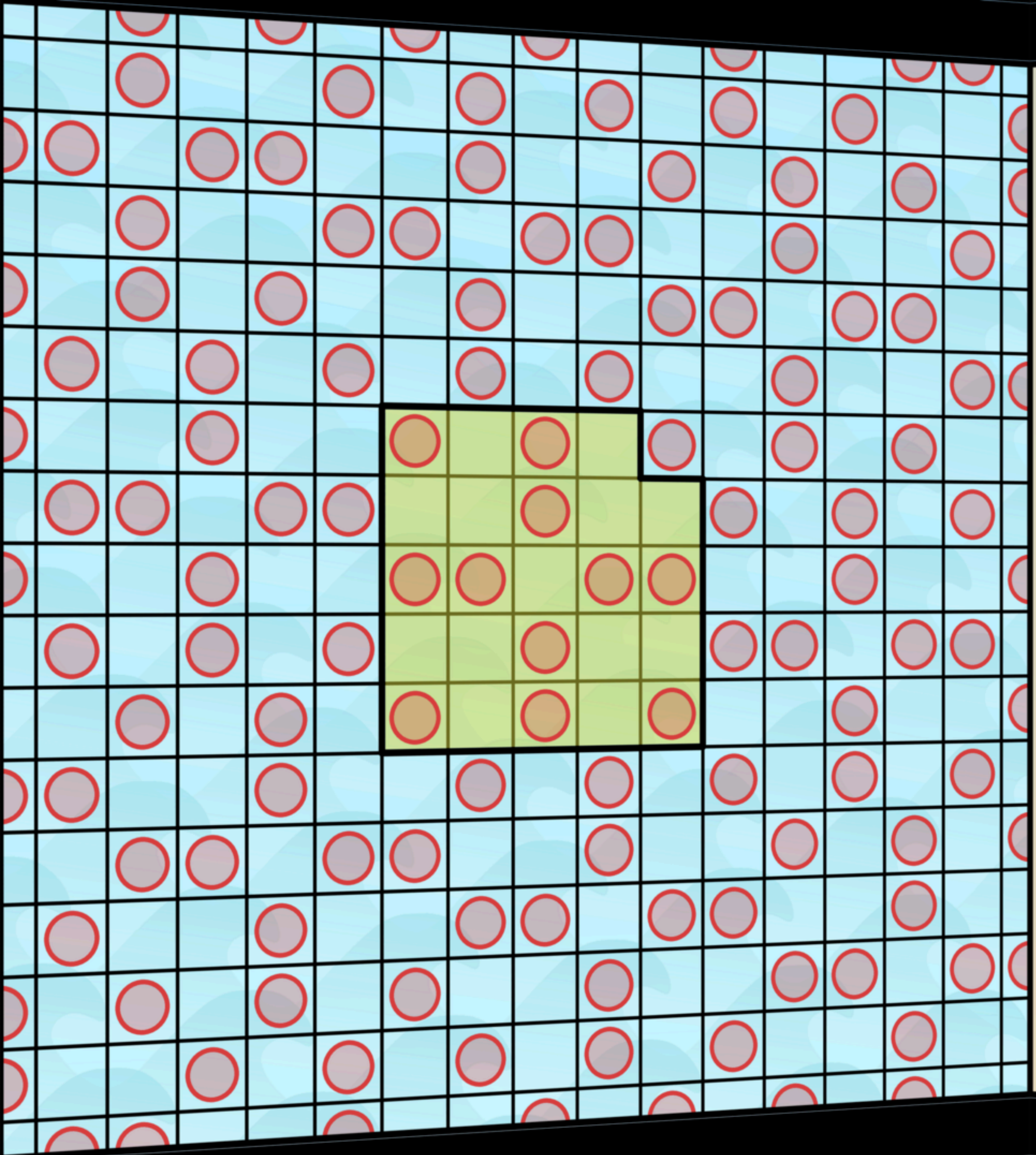


Suppose that all we know is the *size* of the ship.

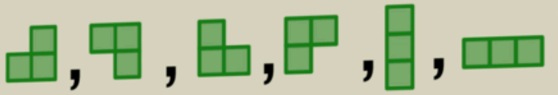
S={}.

S={}.

What is the density of this pattern?

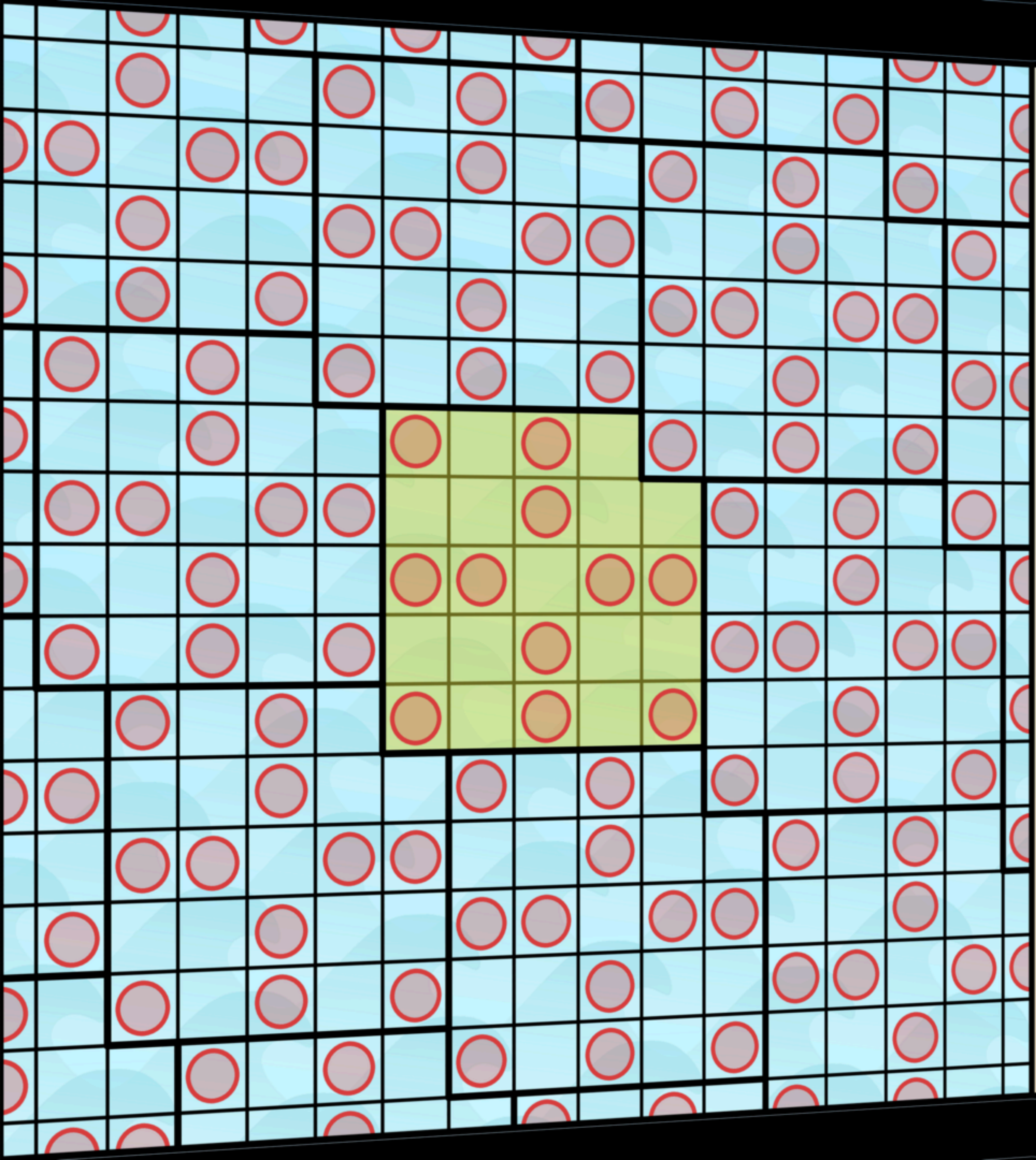


Suppose that all we know is the *size* of the ship.

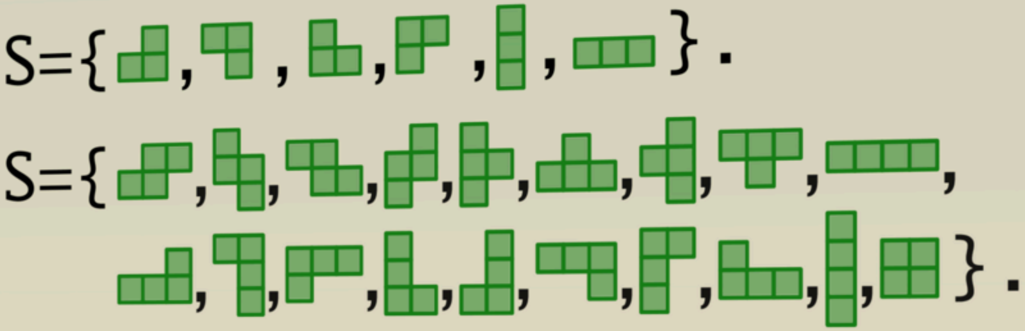
S={}.

S={}.

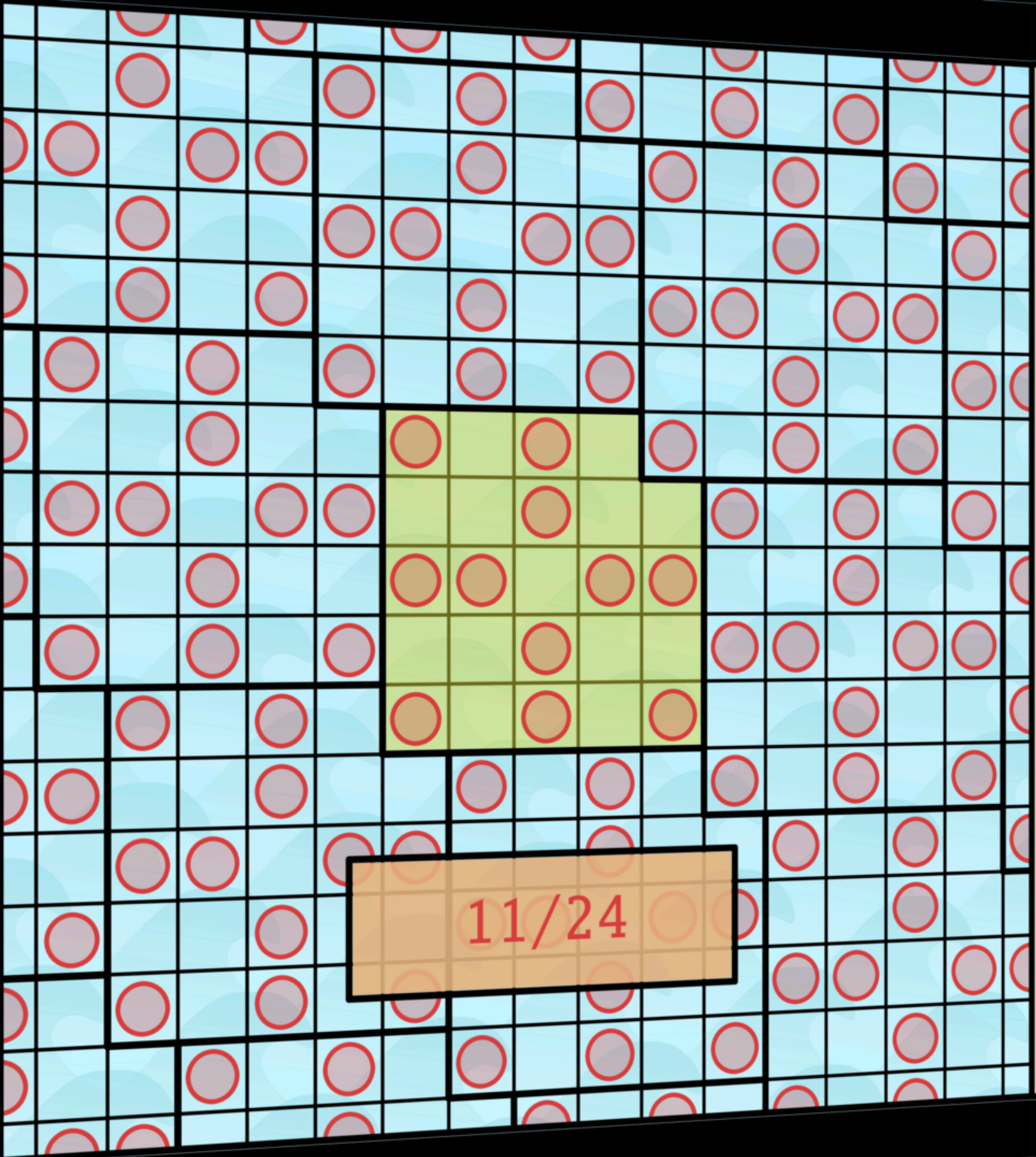
What is the density of this pattern?



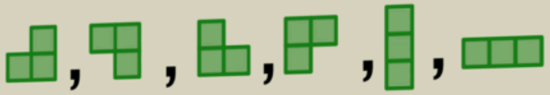
Suppose that all we know is the *size* of the ship.

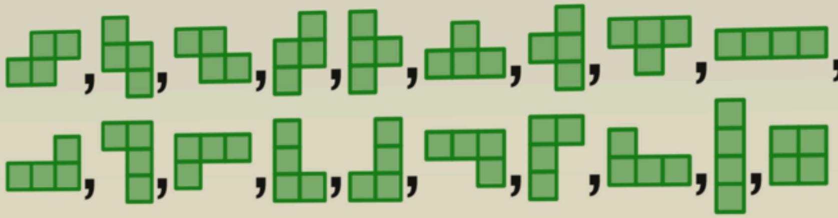


What is the density of this pattern?

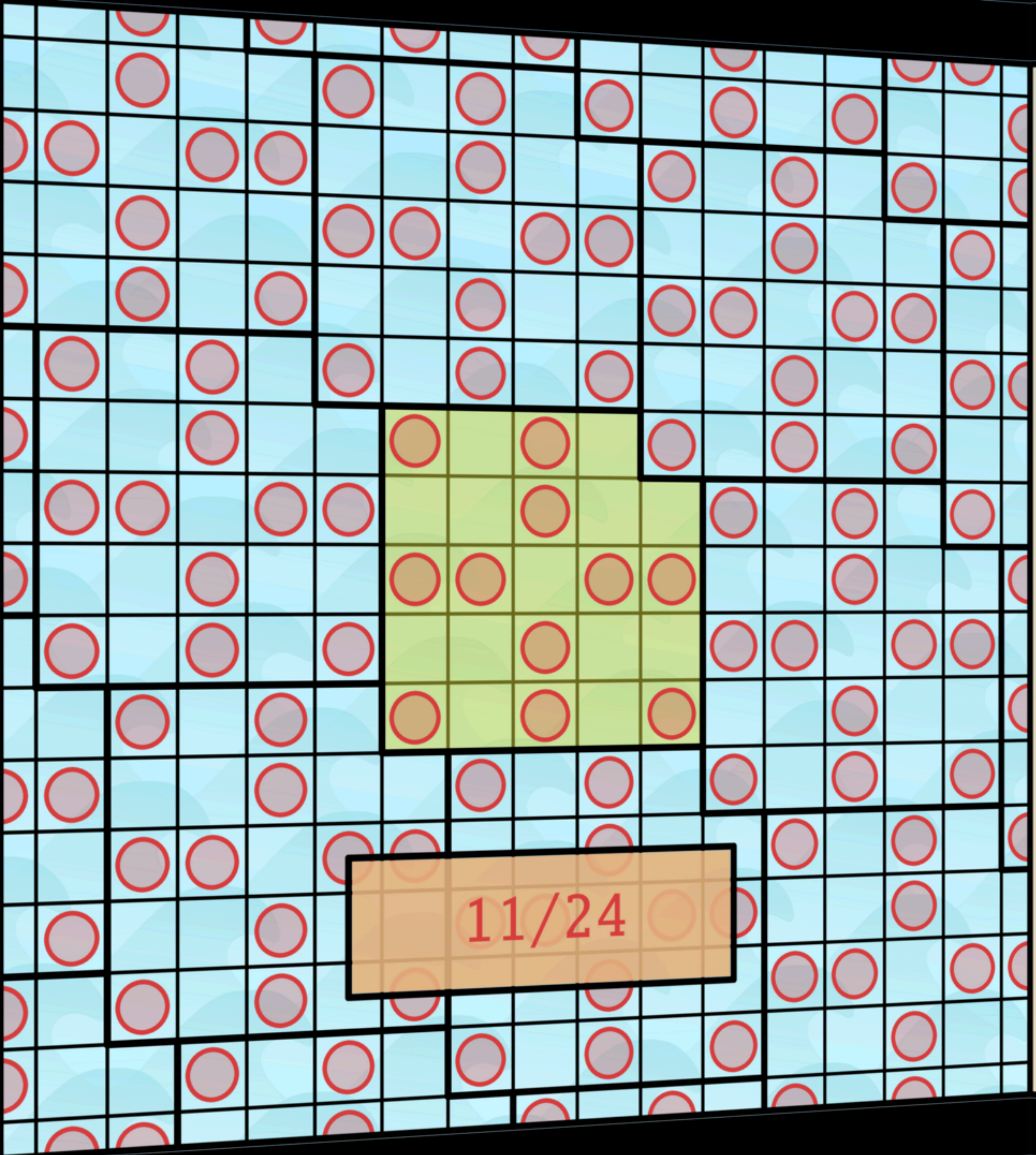


Suppose that all we know is the *size* of the ship.

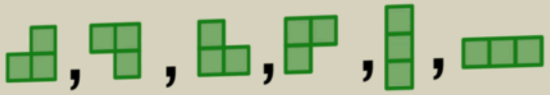
S={}.

S={}.

What is the density of this pattern?



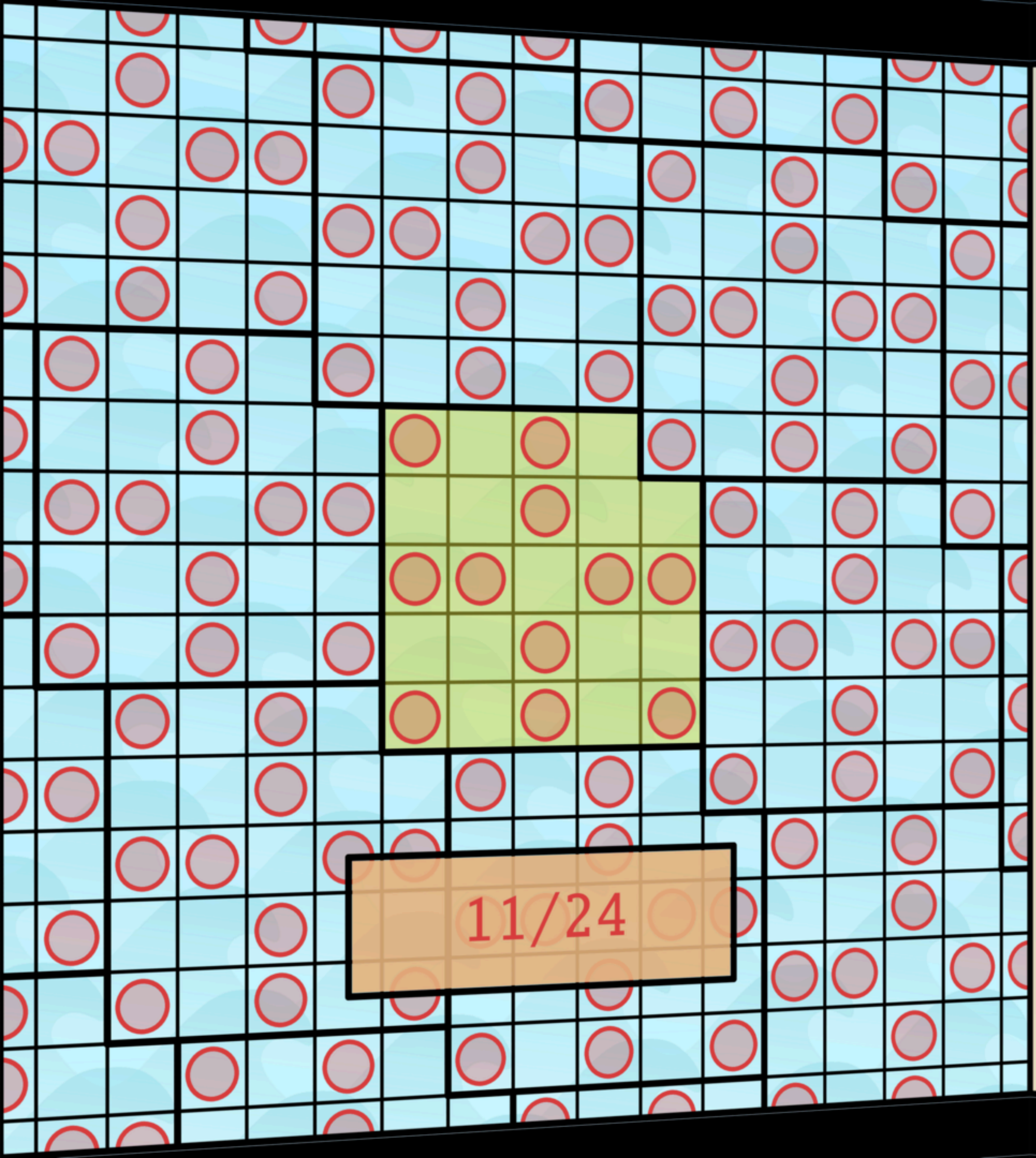
Suppose that all we know is the *size* of the ship.

S={}.

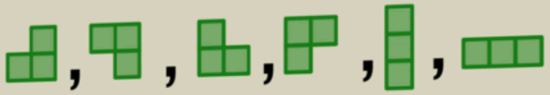
S={}.

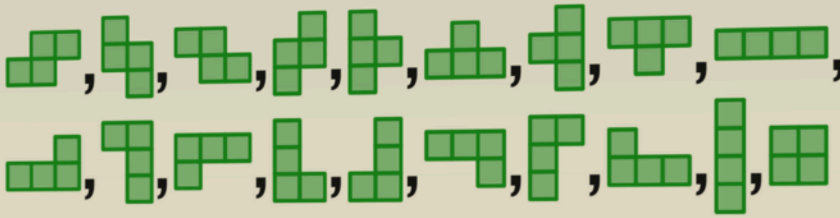
What is the density of this pattern?

Is this optimal?



Suppose that all we know is the *size* of the ship.

S={}.

S={}.

What is the density of this pattern?

Is this optimal?

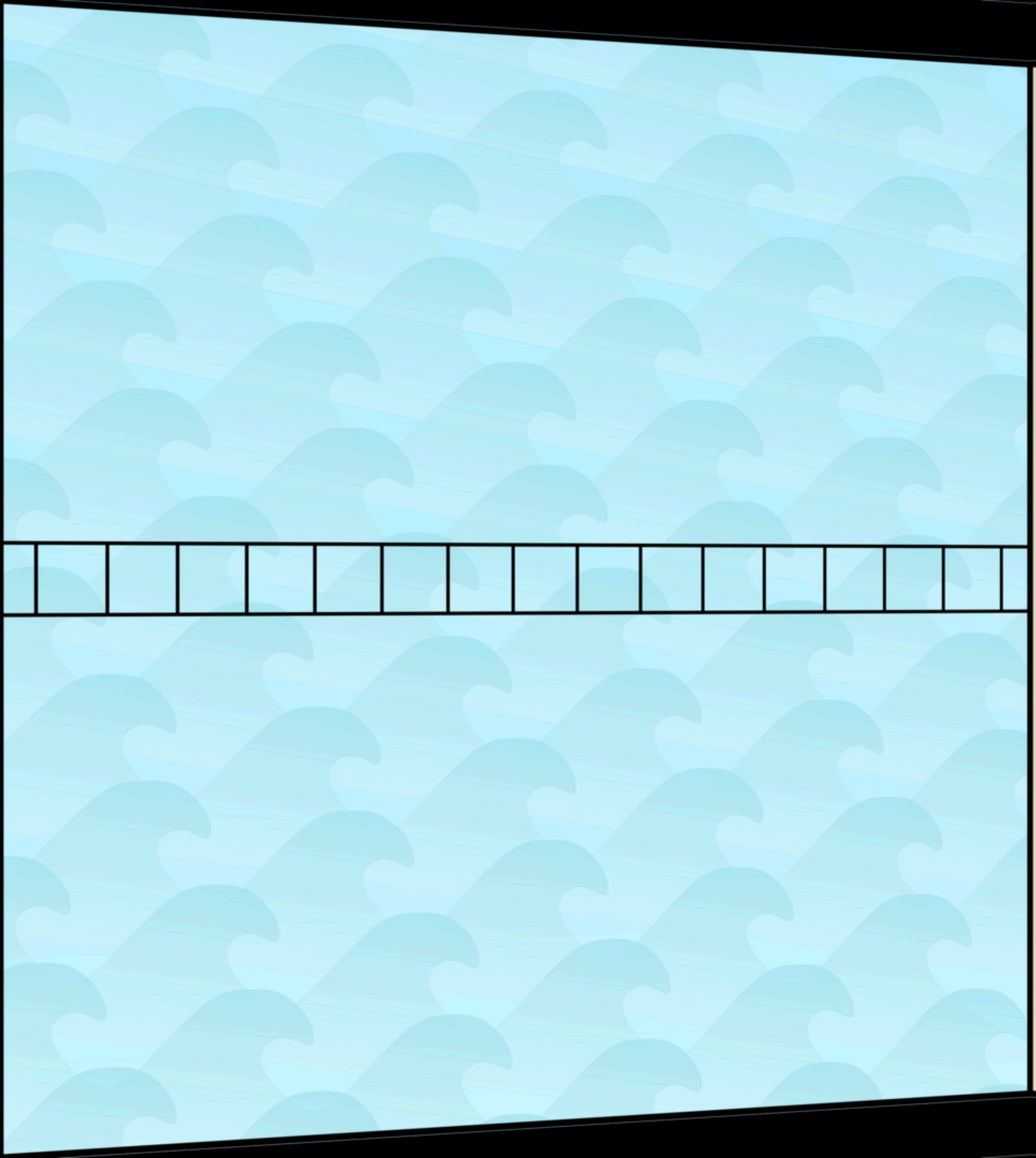
How does this continue?
How do we *find* such patterns?

4

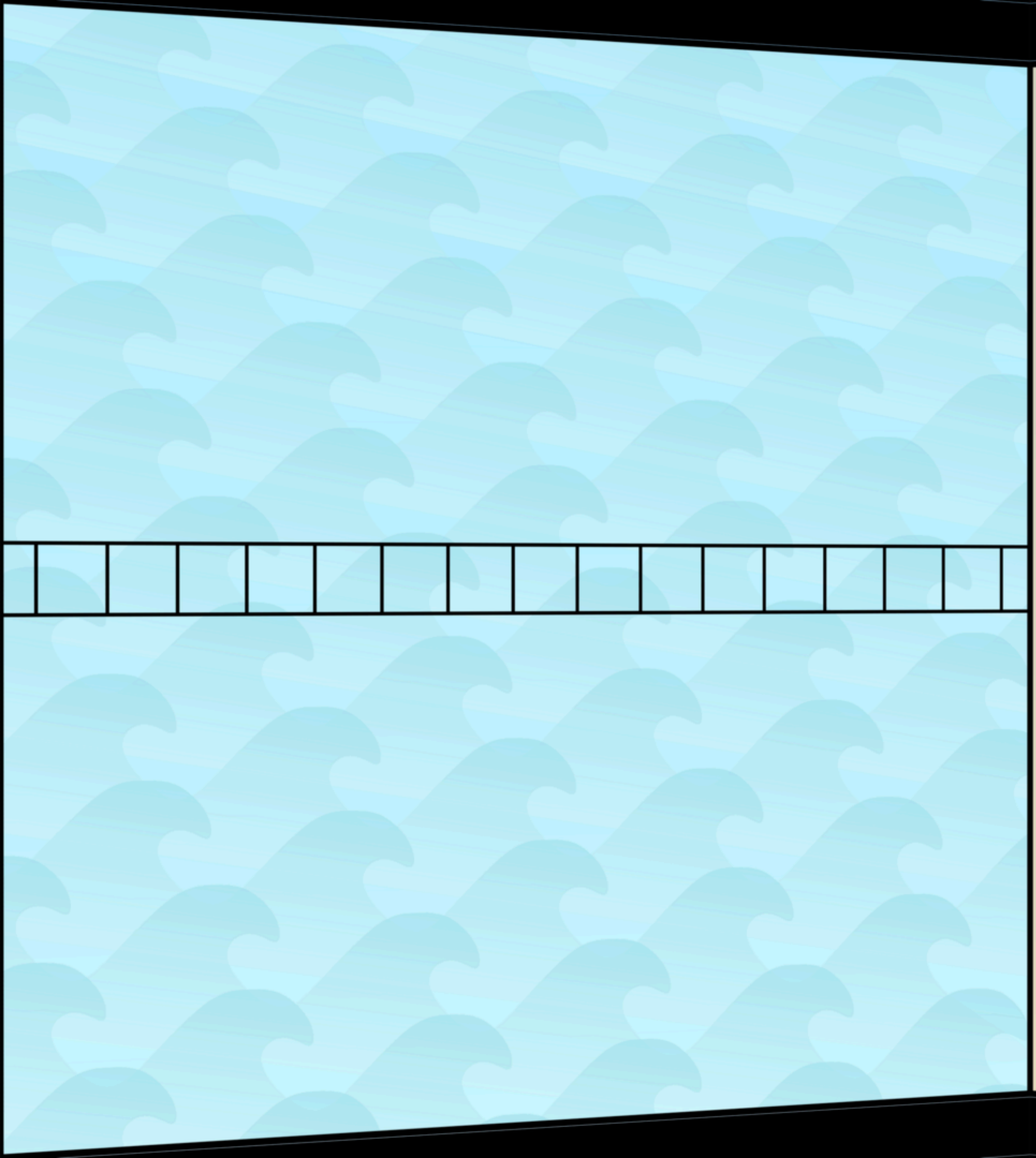
CHAPTER

1-DIMENSIONAL SHIPS

Consider an infinite
1-dimensional grid.

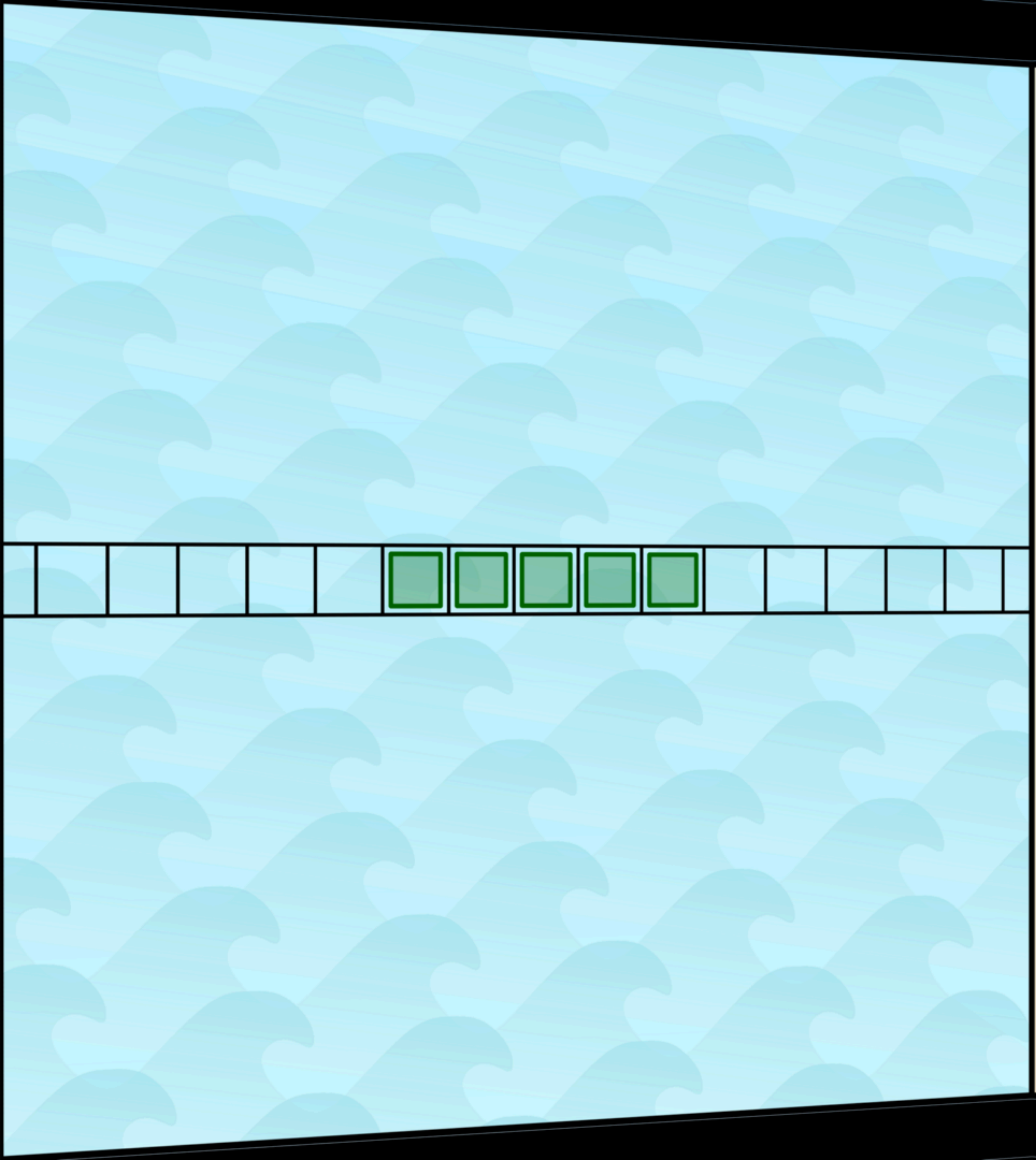


Consider an infinite
1-dimensional grid.



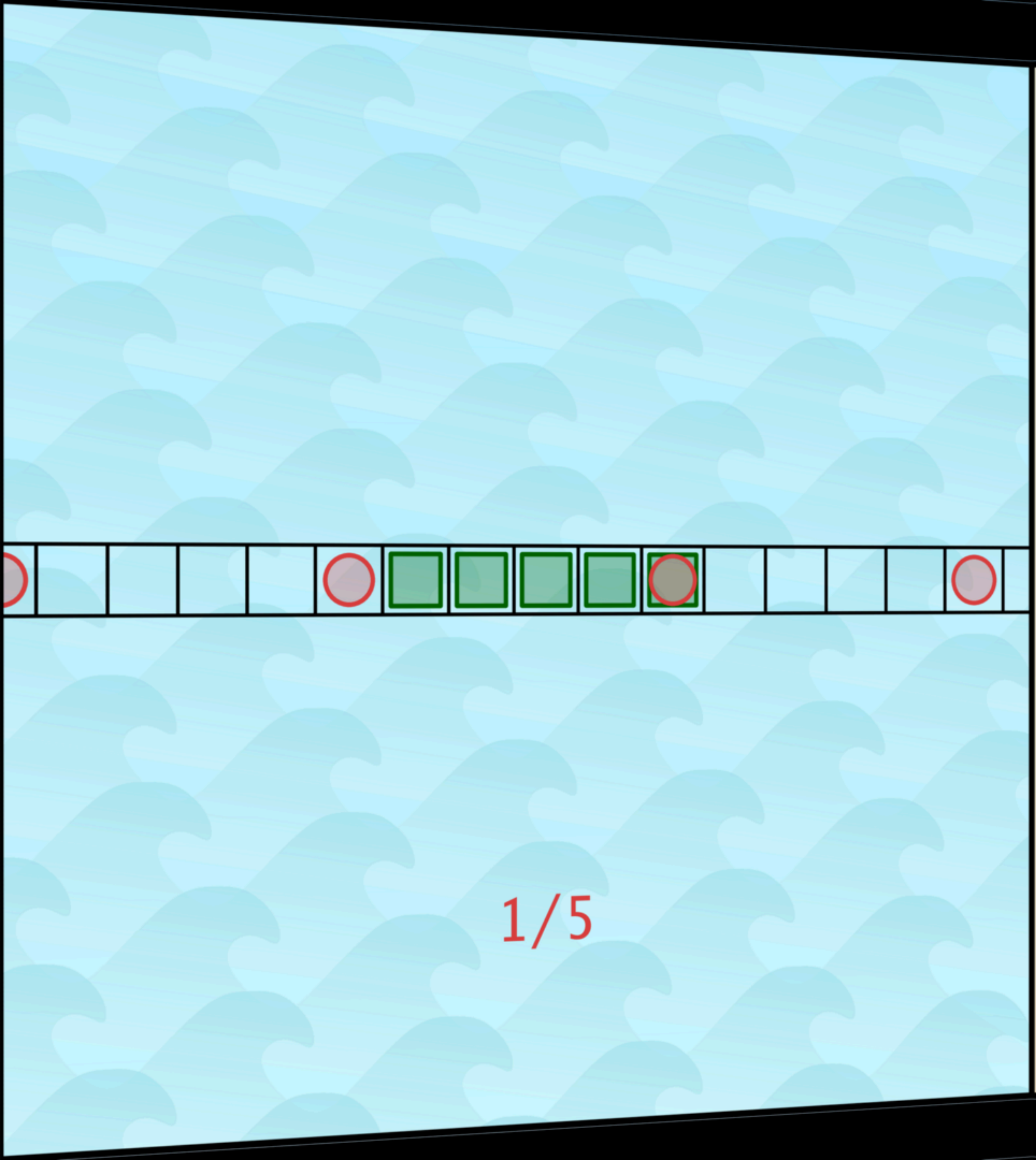
Consider an infinite 1-dimensional grid.

Ships of length k can be hit with density 1/k...



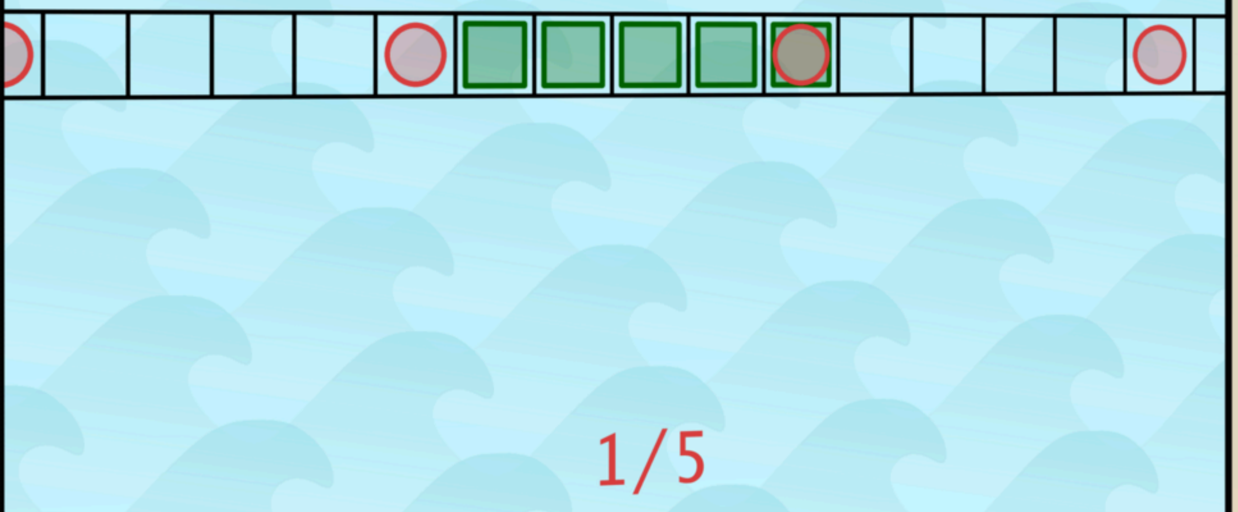
Consider an infinite 1-dimensional grid.

Ships of length k can be hit with density 1/k...



Consider an infinite 1-dimensional grid.

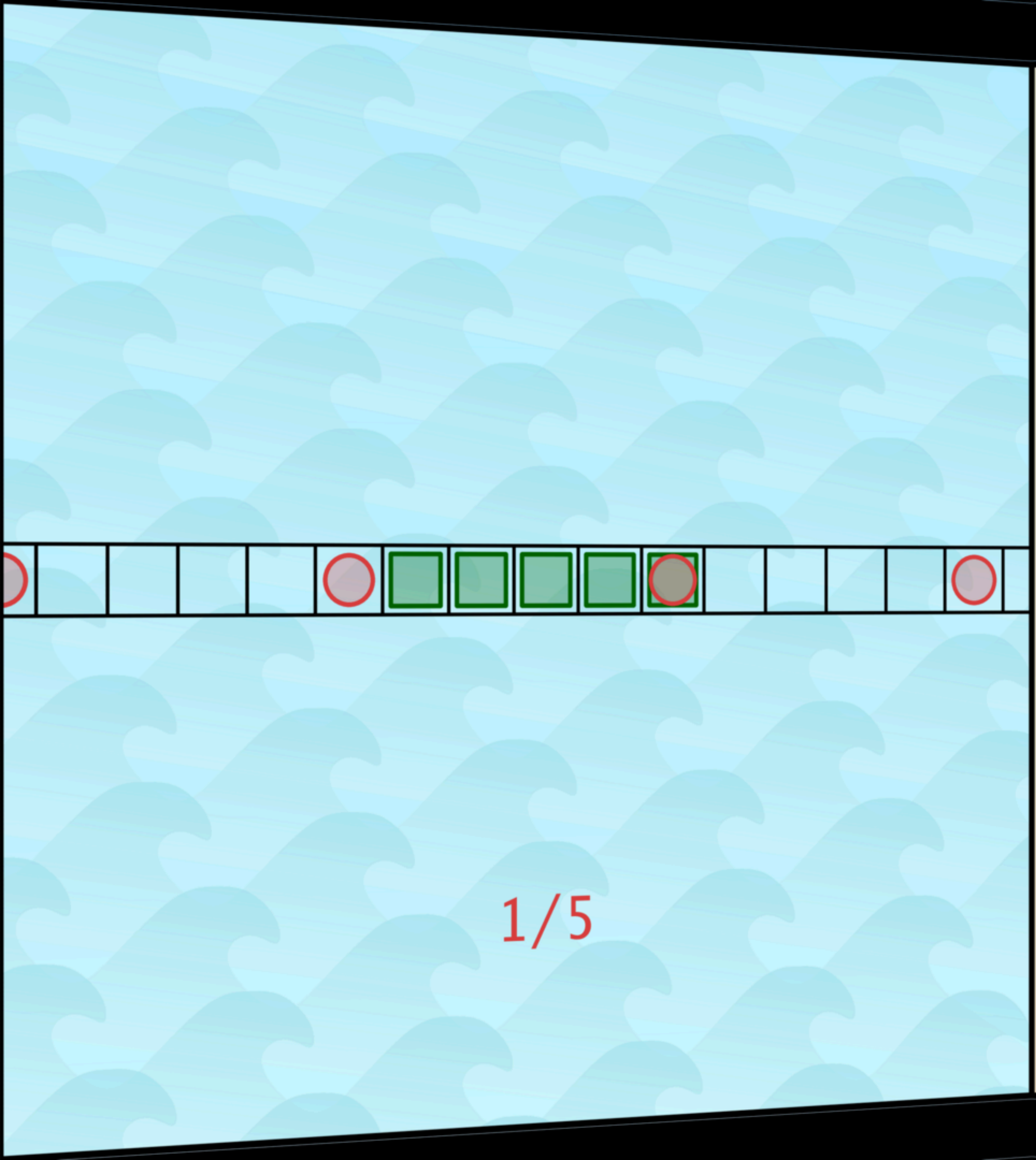
Ships of length k can be hit with density 1/k...



Consider an infinite 1-dimensional grid.

Ships of length k can be hit with density 1/k...

...if they are *connected*.

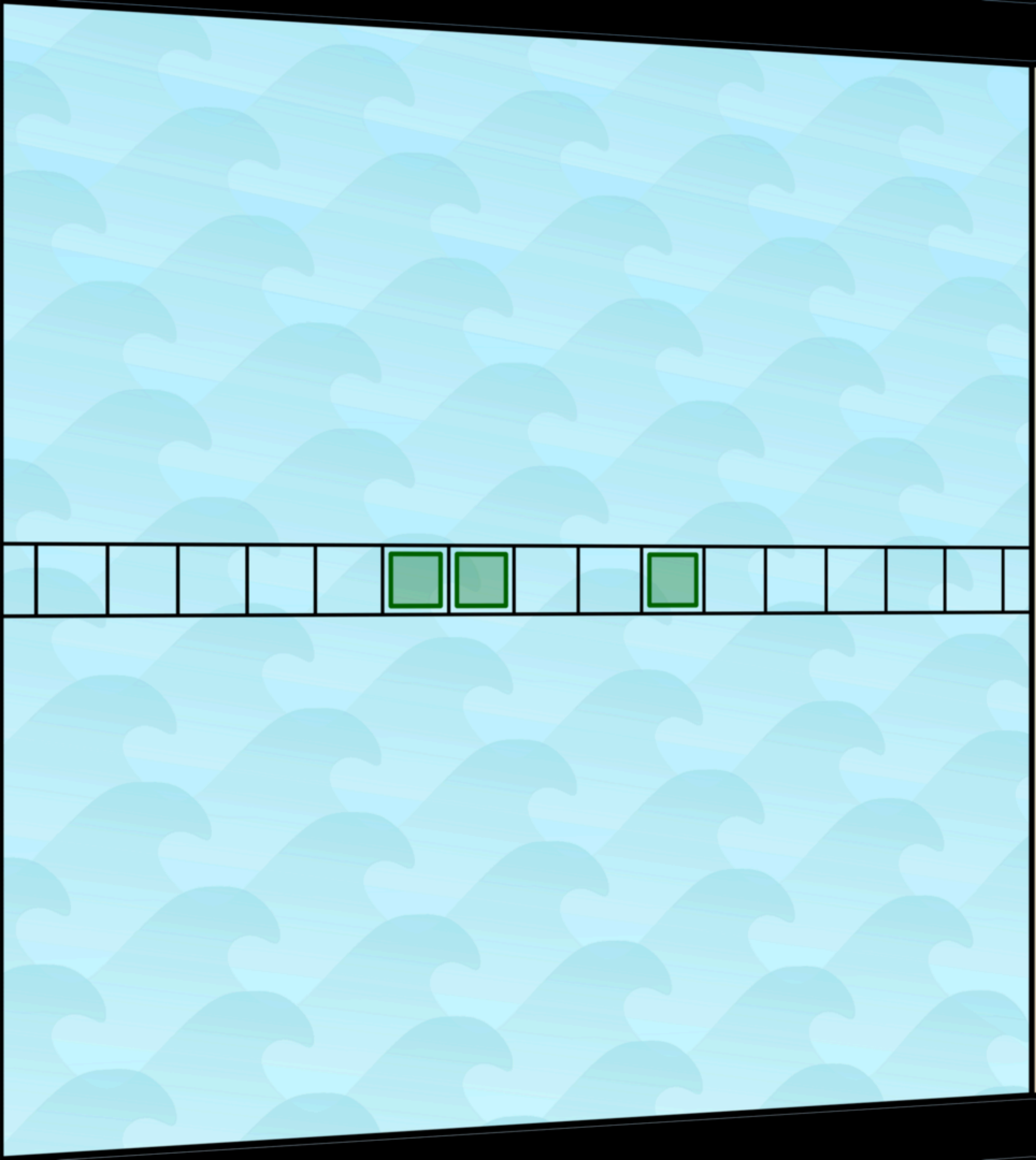


Consider an infinite 1-dimensional grid.

Ships of length k can be hit with density 1/k...

...if they are *connected*.

Disconnected ships behave in less predictable ways.

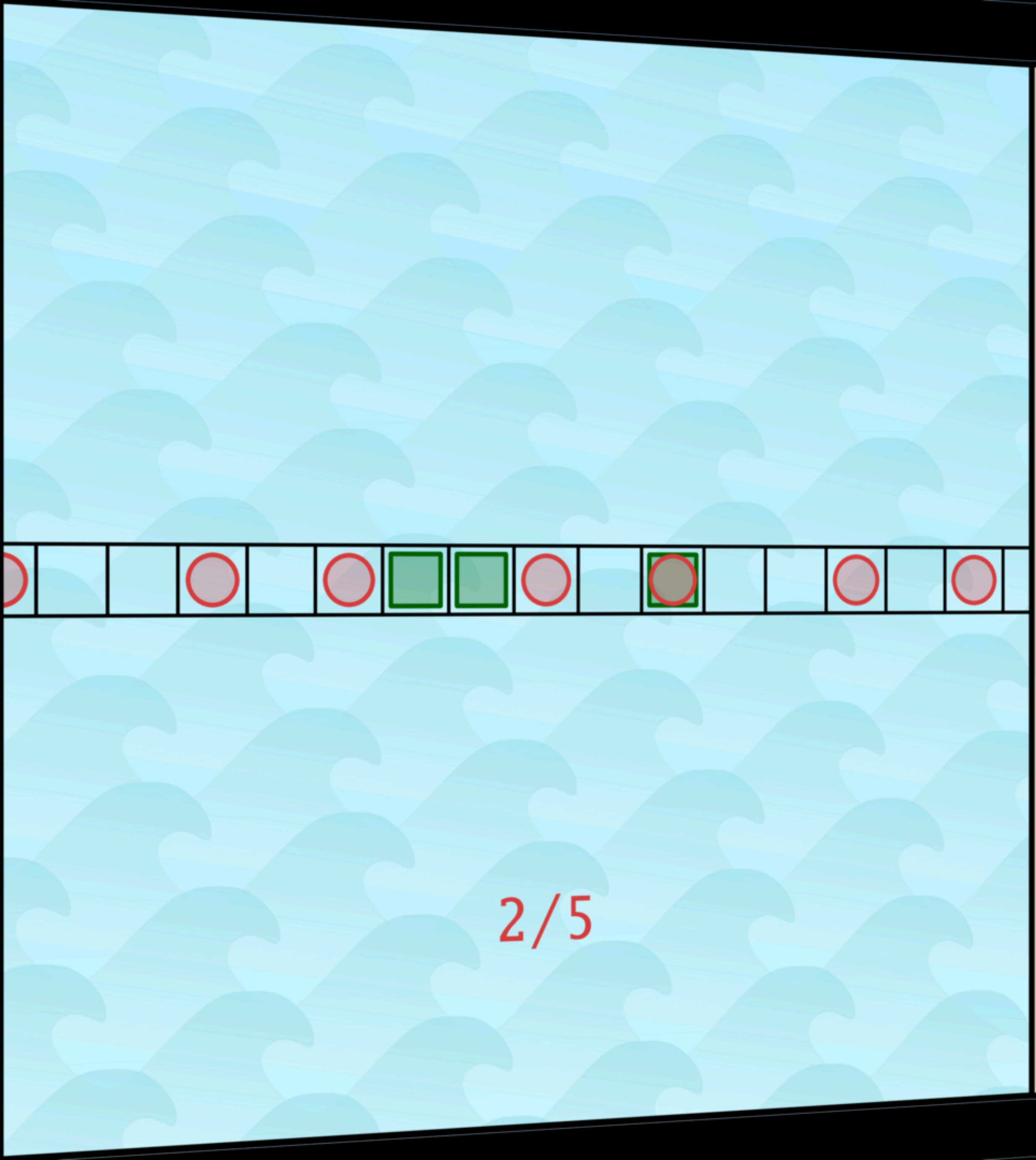


Consider an infinite 1-dimensional grid.

Ships of length k can be hit with density 1/k...

...if they are *connected*.

Disconnected ships behave in less predictable ways.

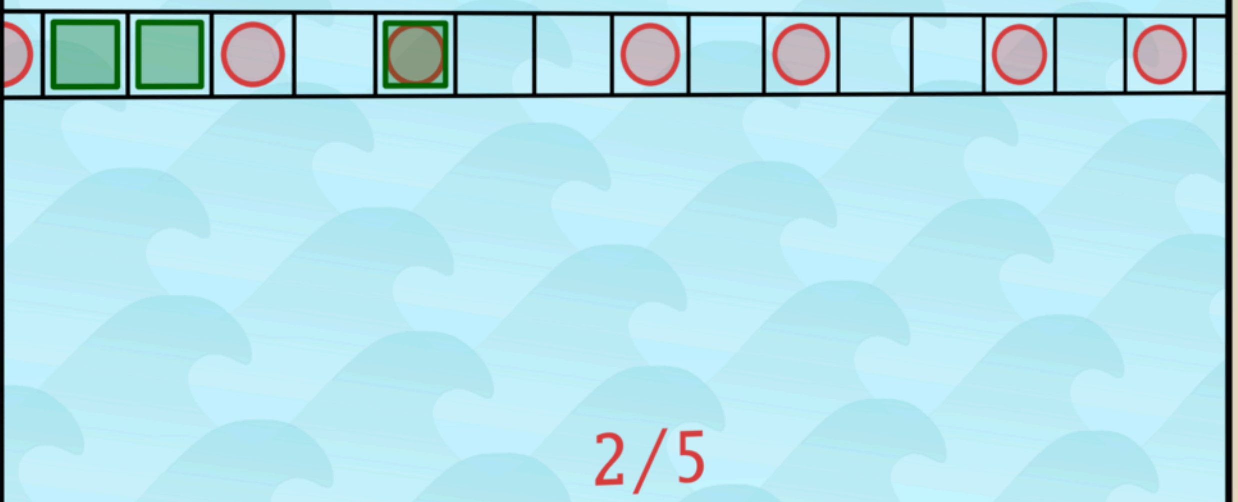


Consider an infinite 1-dimensional grid.

Ships of length k can be hit with density 1/k...

...if they are *connected*.

Disconnected ships behave in less predictable ways.

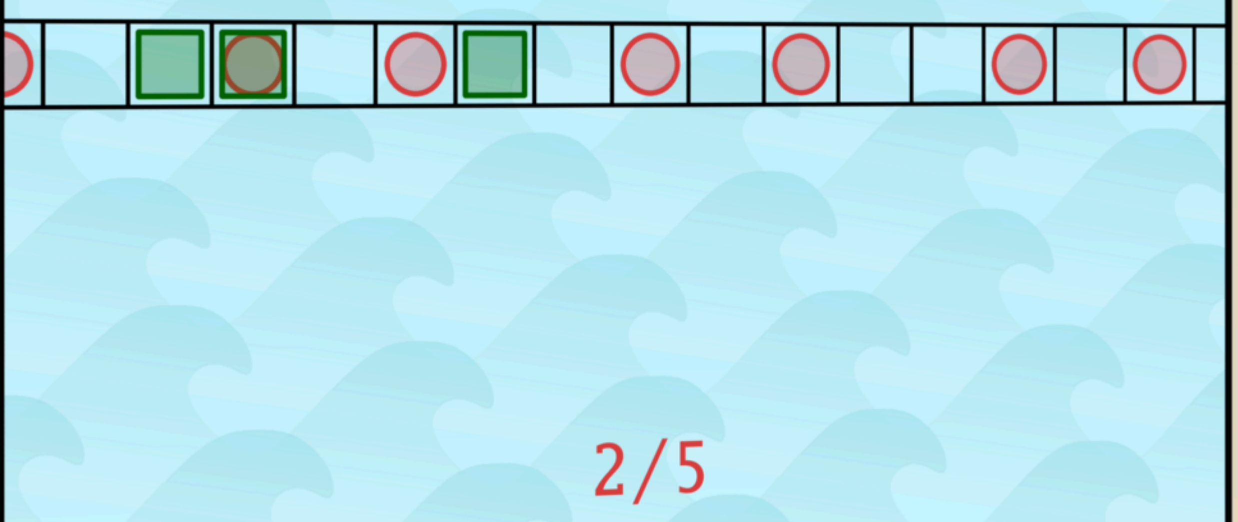


Consider an infinite 1-dimensional grid.

Ships of length k can be hit with density 1/k...

...if they are *connected*.

Disconnected ships behave in less predictable ways.

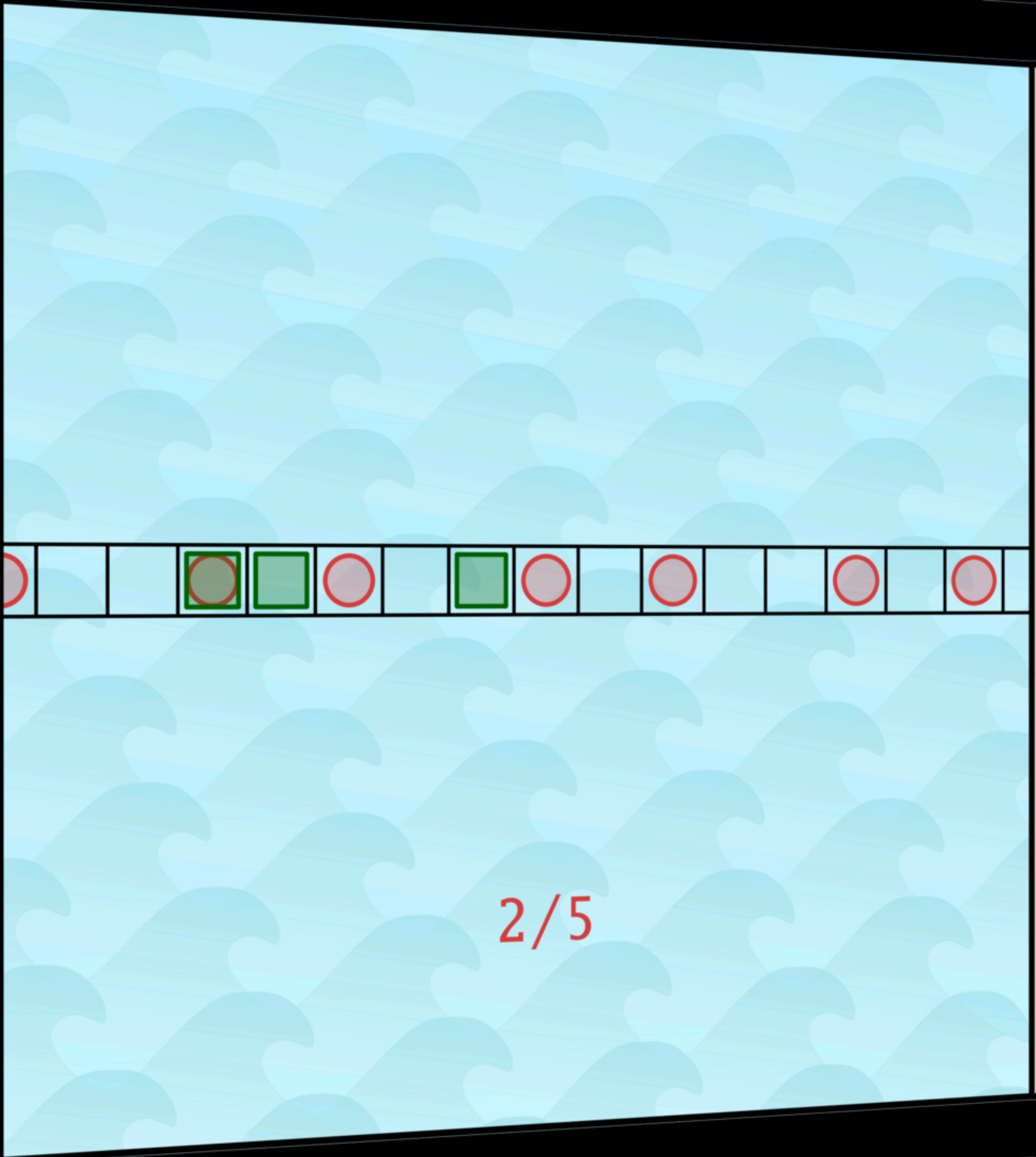


Consider an infinite 1-dimensional grid.

Ships of length k can be hit with density 1/k...

...if they are *connected*.

Disconnected ships behave in less predictable ways.

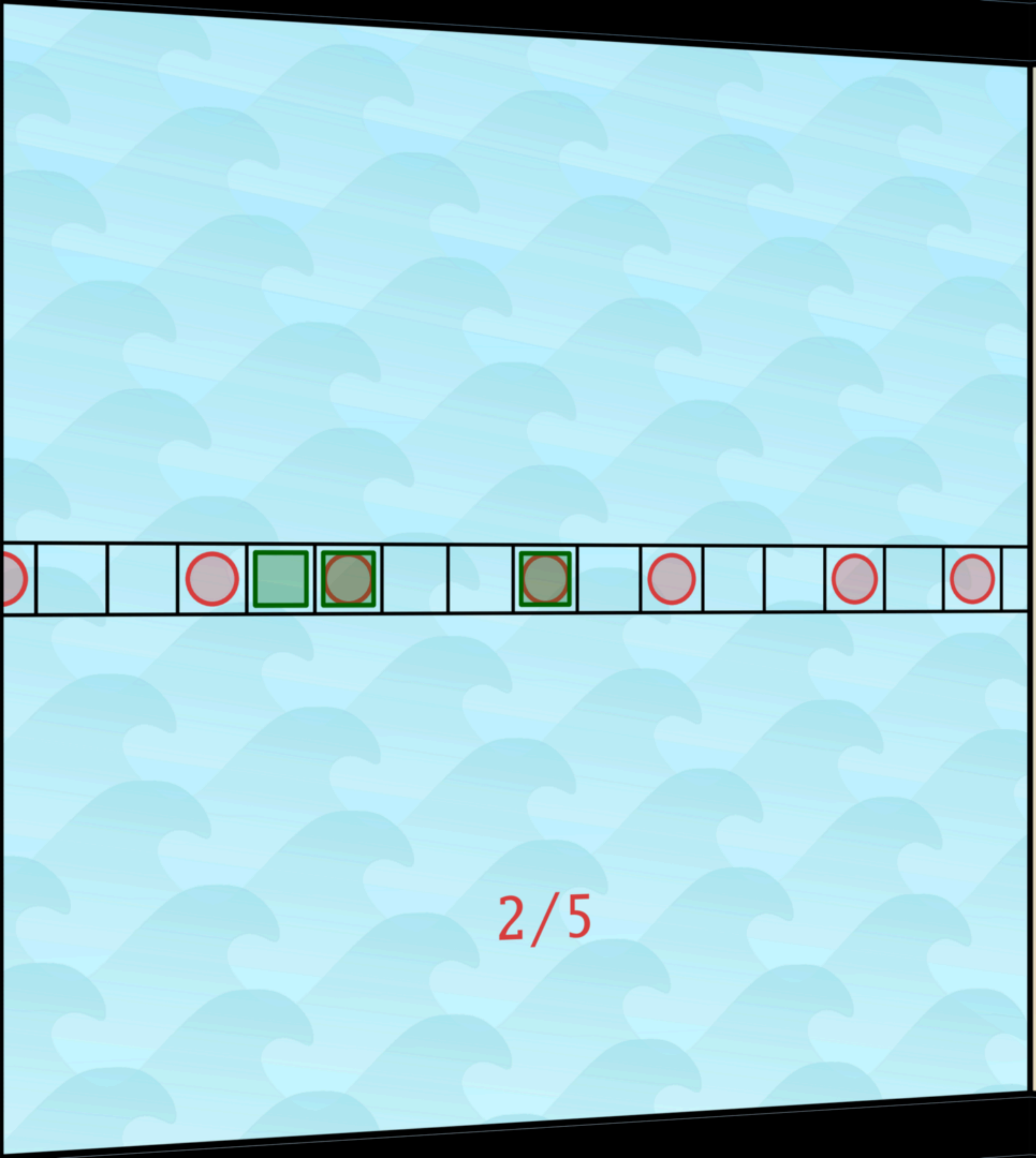


Consider an infinite 1-dimensional grid.

Ships of length k can be hit with density 1/k...

...if they are *connected*.

Disconnected ships behave in less predictable ways.



Consider an infinite 1-dimensional grid.

Ships of length k can be hit with density 1/k...

...if they are *connected*.

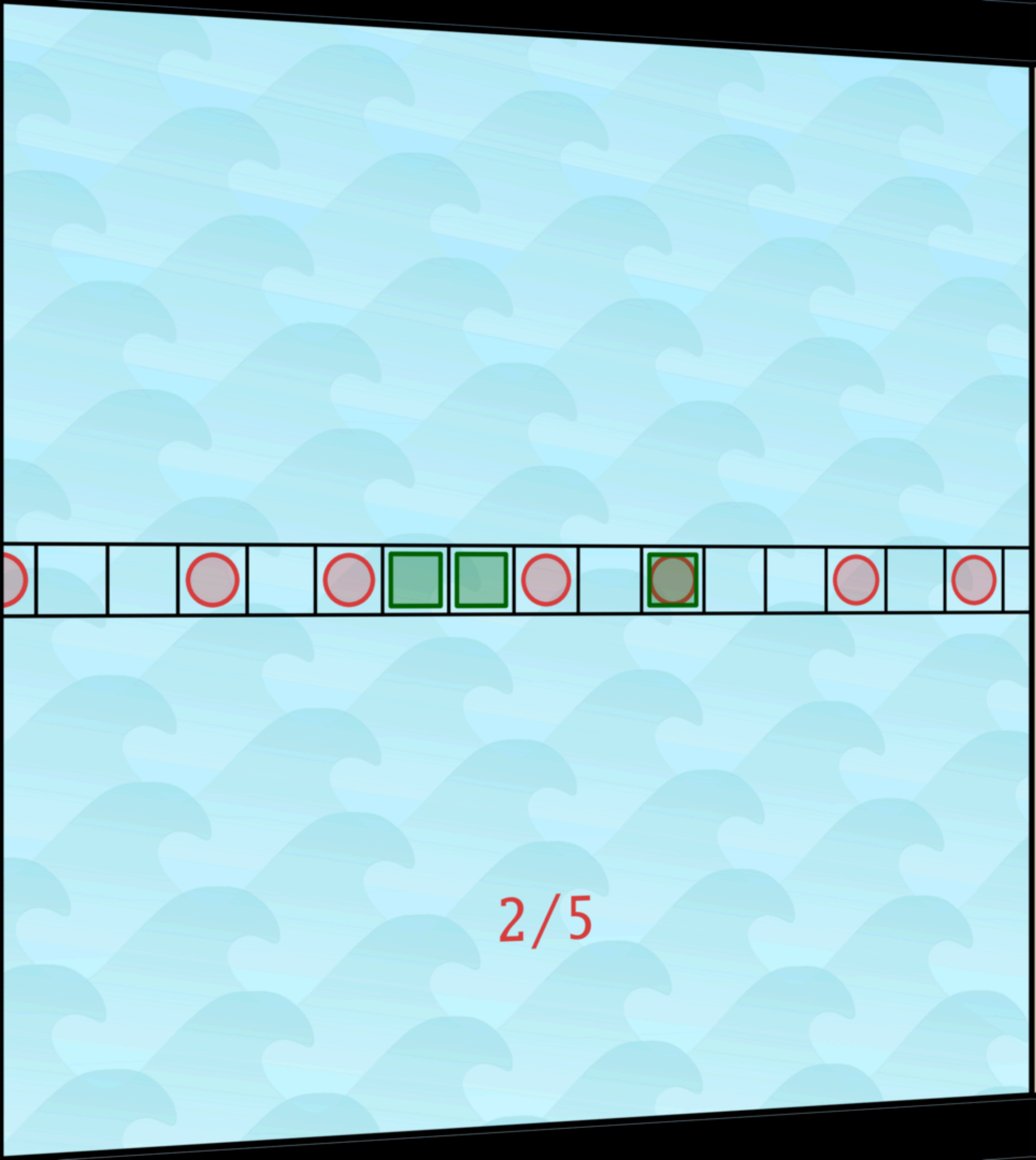
Disconnected ships behave in less predictable ways.

Consider an infinite 1-dimensional grid.

Ships of length k can be hit with density 1/k...

...if they are *connected*.

Disconnected ships behave in less predictable ways.

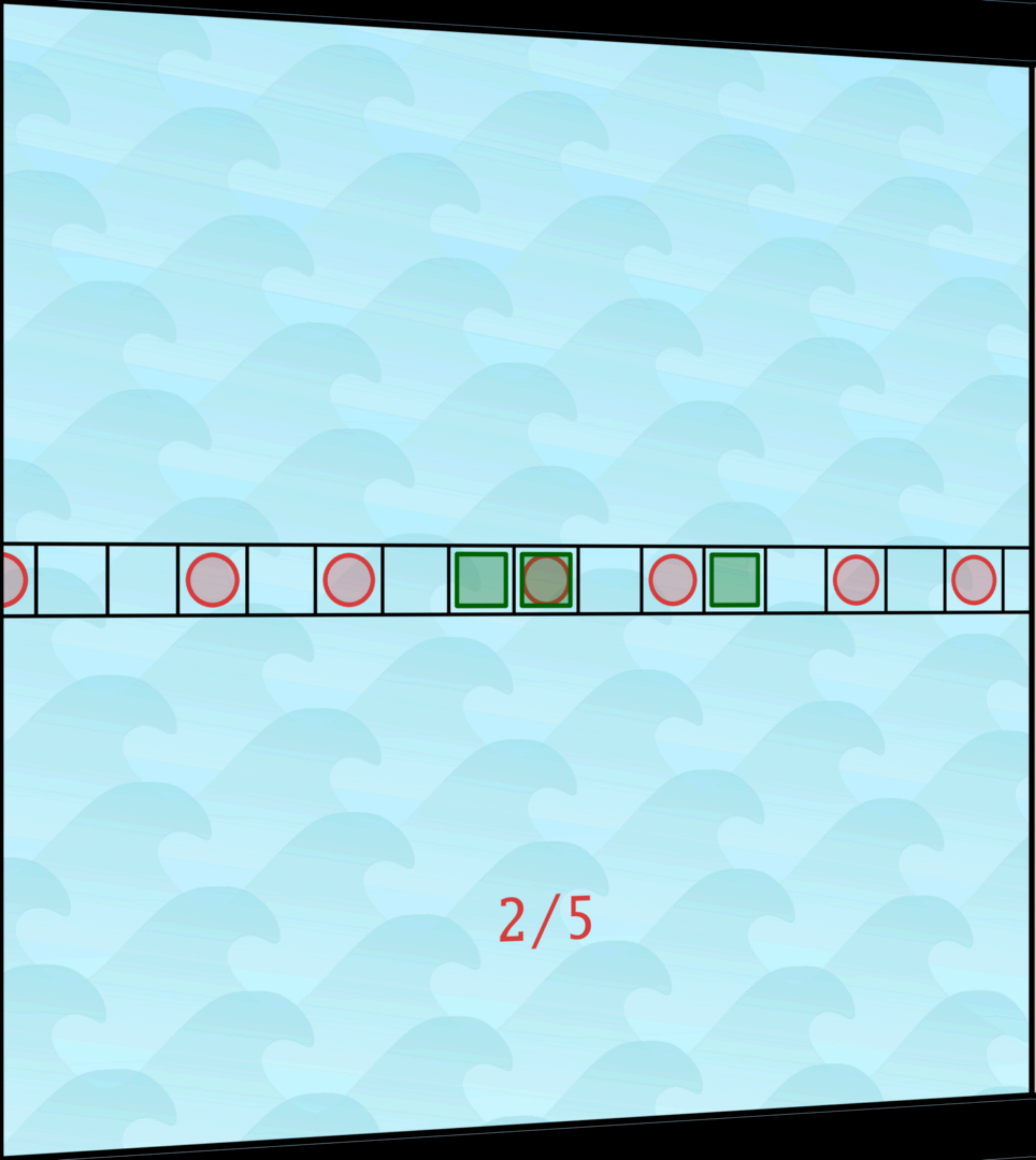


Consider an infinite 1-dimensional grid.

Ships of length k can be hit with density 1/k...

...if they are *connected*.

Disconnected ships behave in less predictable ways.

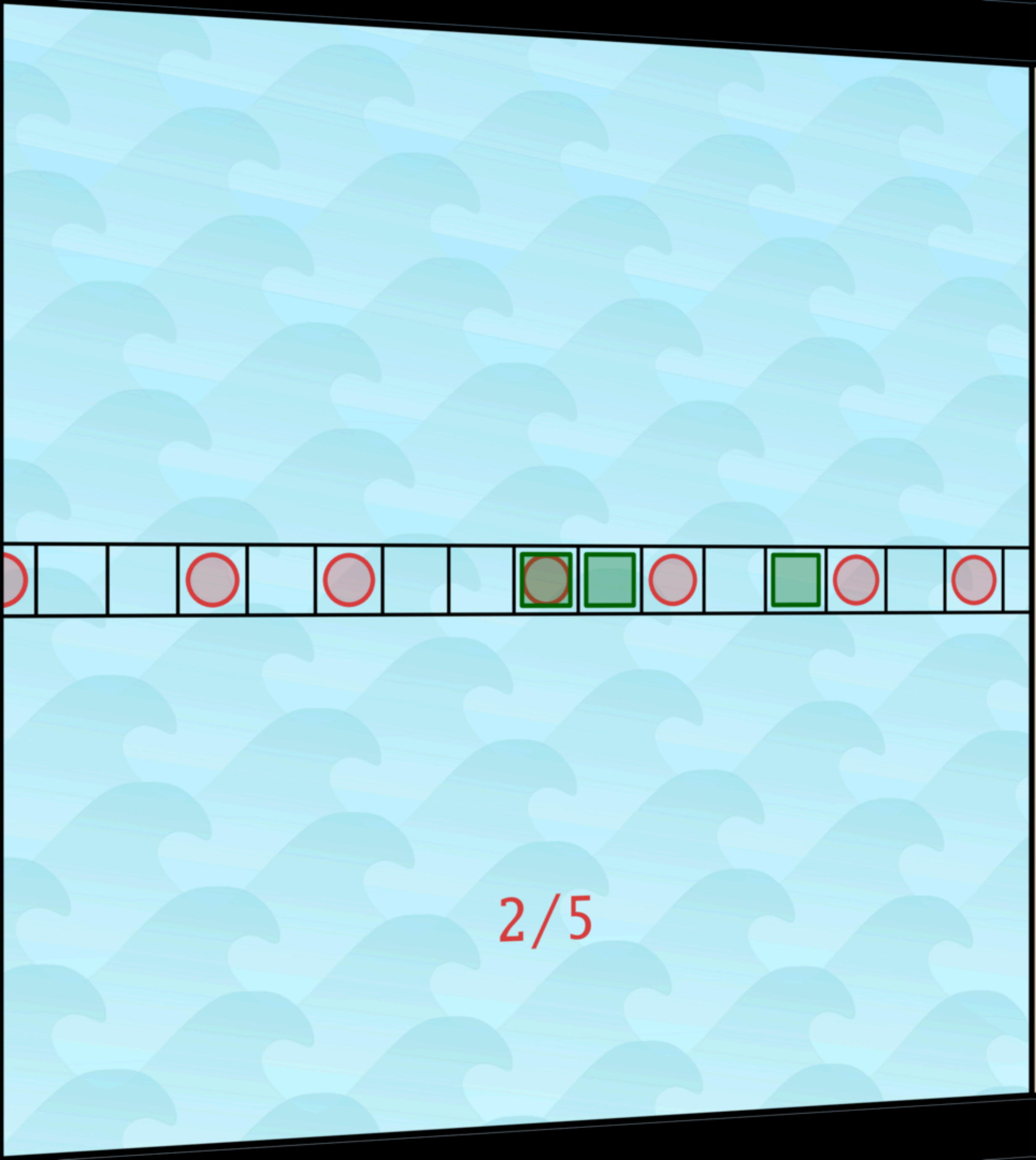


Consider an infinite 1-dimensional grid.

Ships of length k can be hit with density 1/k...

...if they are *connected*.

Disconnected ships behave in less predictable ways.

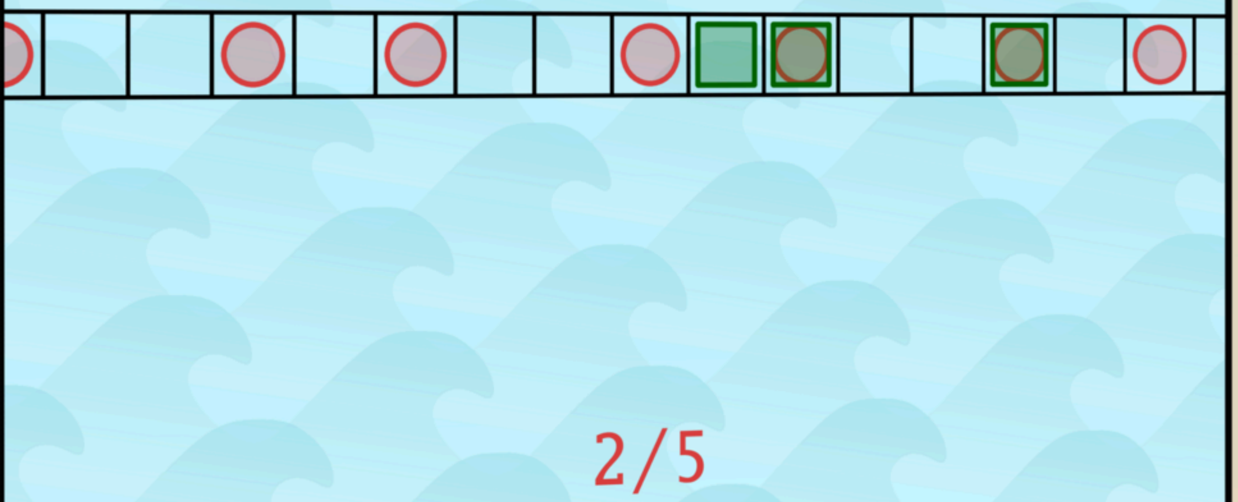


Consider an infinite 1-dimensional grid.

Ships of length k can be hit with density 1/k...

...if they are *connected*.

Disconnected ships behave in less predictable ways.

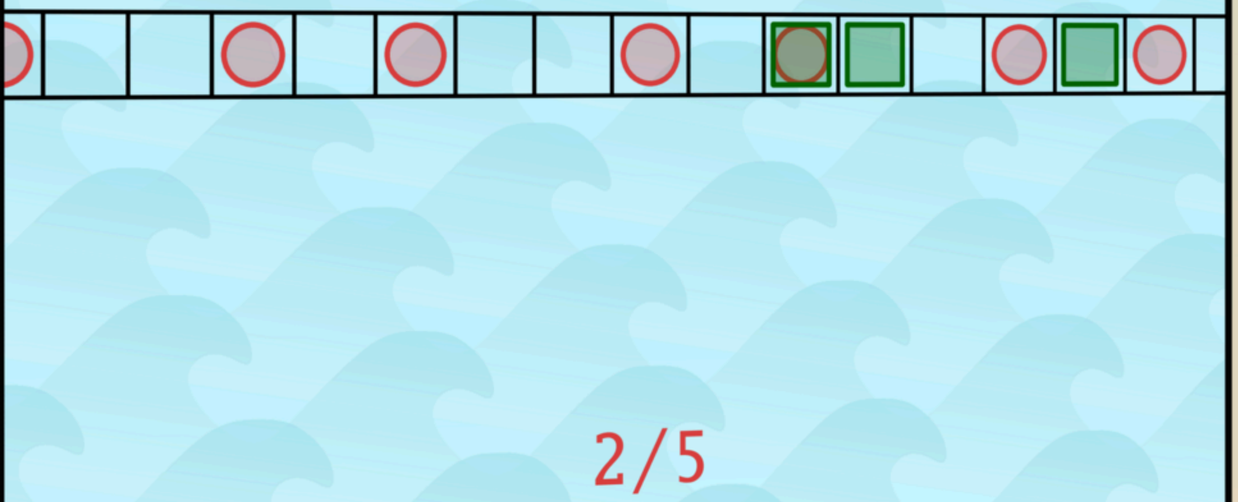


Consider an infinite 1-dimensional grid.

Ships of length k can be hit with density 1/k...

...if they are *connected*.

Disconnected ships behave in less predictable ways.



Consider an infinite 1-dimensional grid.

Ships of length k can be hit with density 1/k...

...if they are *connected*.

Disconnected ships behave in less predictable ways.

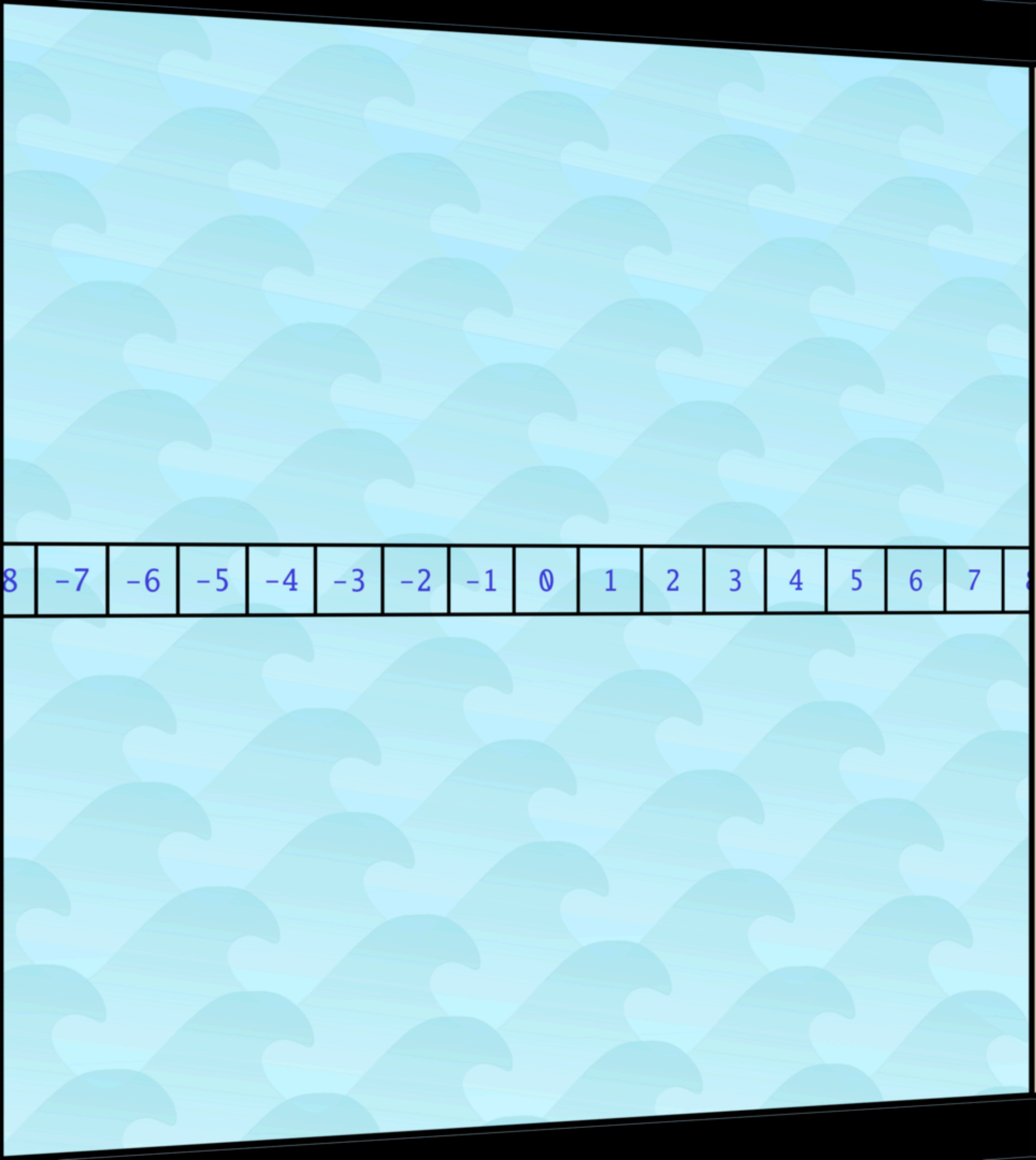
2/5

Consider an infinite
1-dimensional grid.

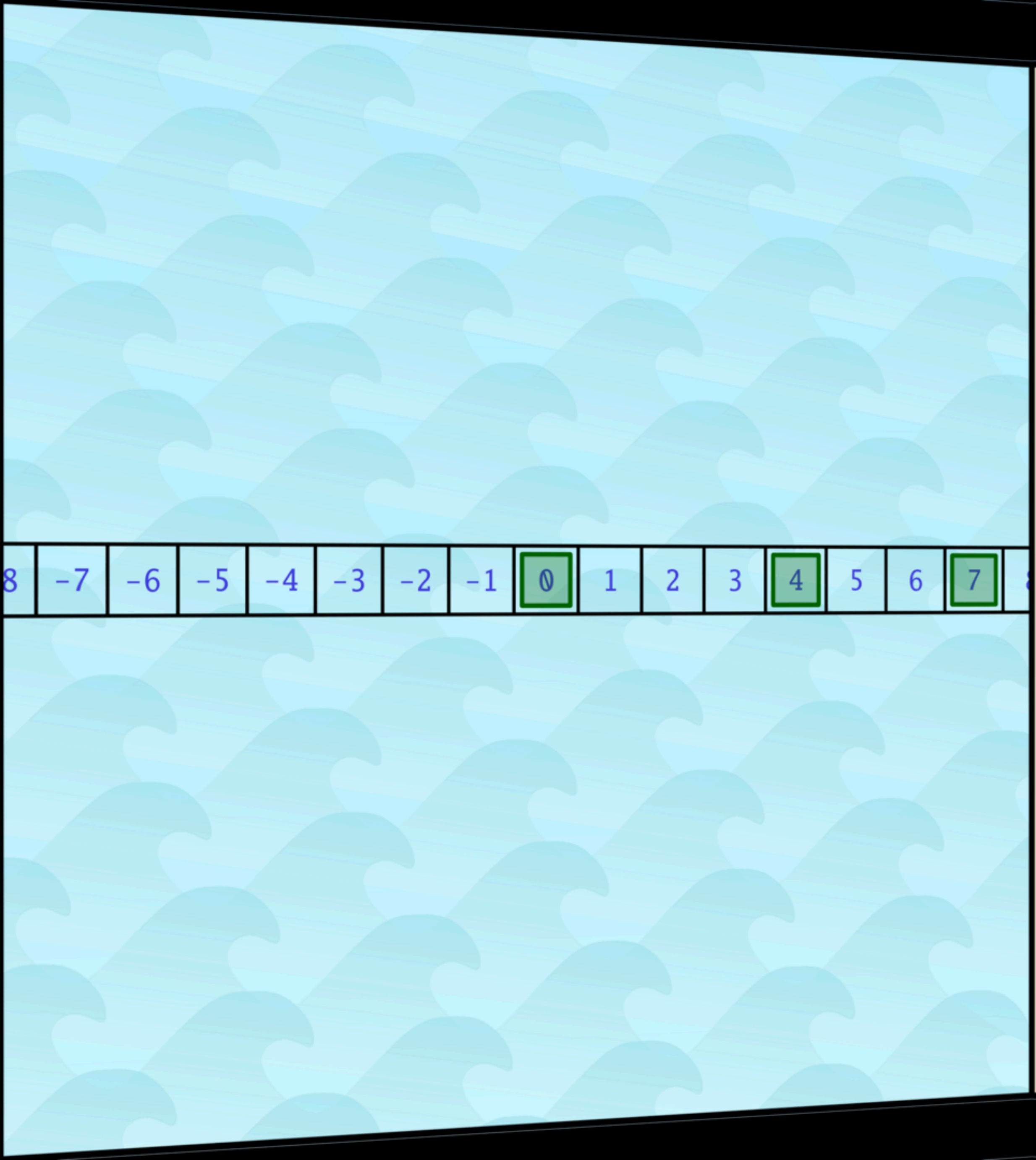
Ships of length k can be
hit with density 1/k...

...if they are *connected*.

Disconnected ships behave
in less predictable ways.



Ships $[0, a, b]$ with a and b coprime.



Ships $[0, a, b]$ with a and b coprime.

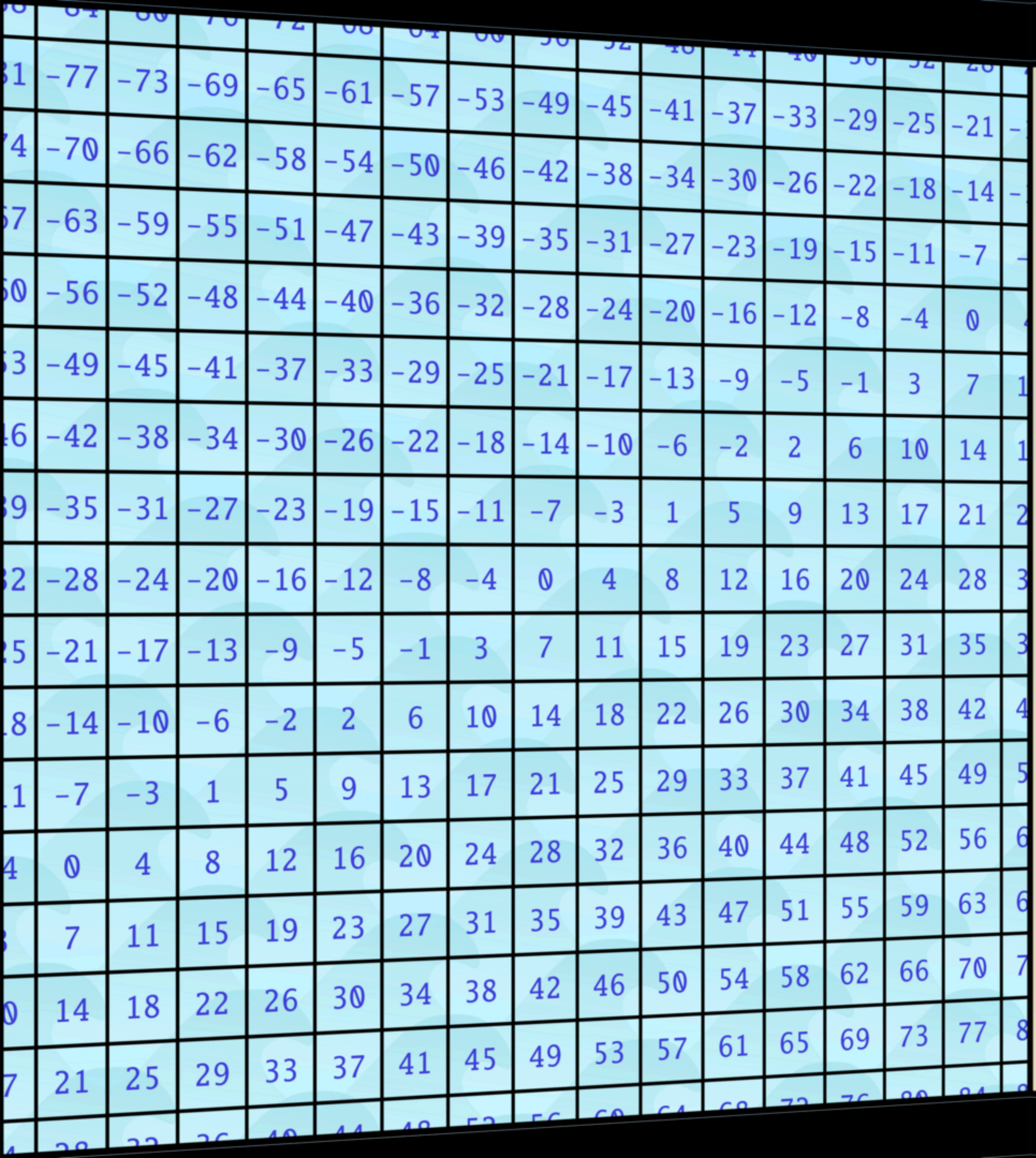
Ships $[0, a, b]$ with a and b coprime.

We can reuse our understanding about 2-dimensional ships!

Ships $[0, a, b]$ with a and b coprime.

We can reuse our understanding about 2-dimensional ships!

Consider the map $(x, y) \mapsto ay + bx$.



Ships $[0, a, b]$ with a and b coprime.

We can reuse our understanding about 2-dimensional ships!

Consider the map $(x, y) \mapsto ay + bx$.

Ships $[0, a, b]$ with a and b coprime.

We can reuse our understanding about 2-dimensional ships!

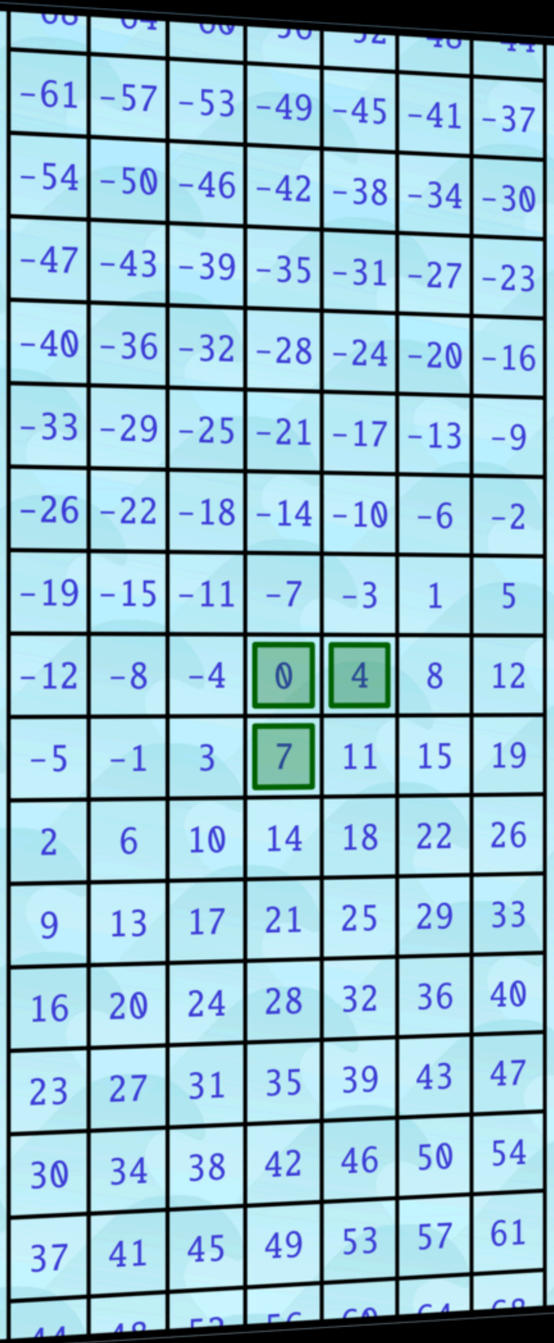
Consider the map $(x, y) \mapsto ay + bx$.

Since a and b are coprime, every strip of width b contains all integers exactly once!

-68	-64	-60	-56	-52	-48	-44
-61	-57	-53	-49	-45	-41	-37
-54	-50	-46	-42	-38	-34	-30
-47	-43	-39	-35	-31	-27	-23
-40	-36	-32	-28	-24	-20	-16
-33	-29	-25	-21	-17	-13	-9
-26	-22	-18	-14	-10	-6	-2
-19	-15	-11	-7	-3	1	5
-12	-8	-4	0	4	8	12
-5	-1	3	7	11	15	19
2	6	10	14	18	22	26
9	13	17	21	25	29	33
16	20	24	28	32	36	40
23	27	31	35	39	43	47
30	34	38	42	46	50	54
37	41	45	49	53	57	61
44	48	52	56	60	64	68

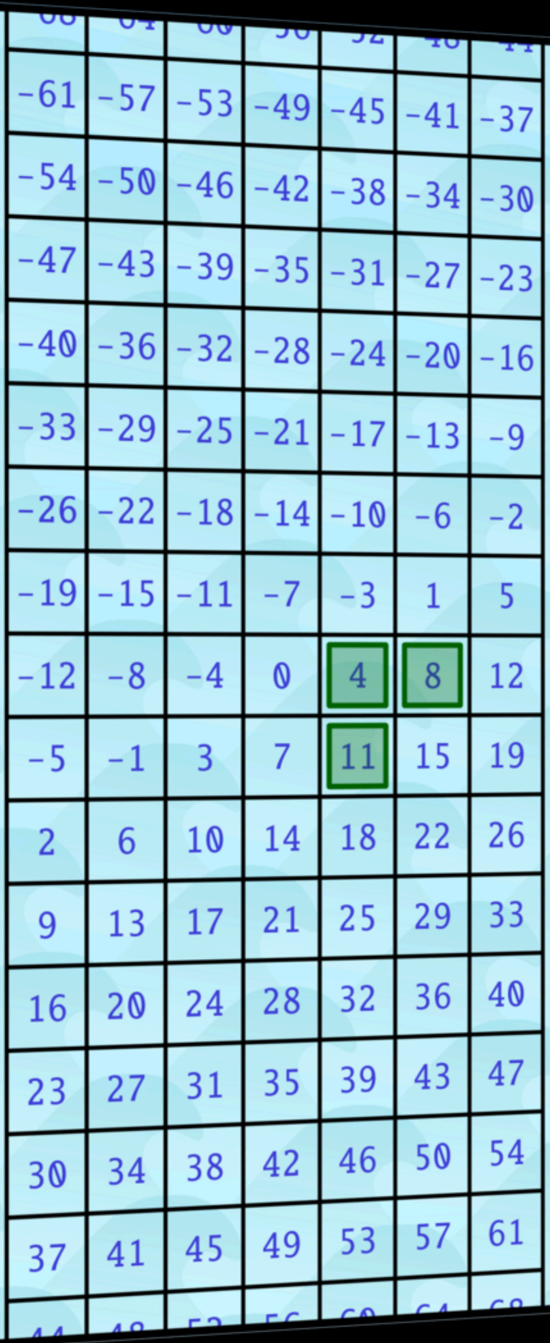
Ships $[0, a, b]$ with a and b coprime.

A disconnected 1-dimensional ship becomes a connected 2-dimensional ship...



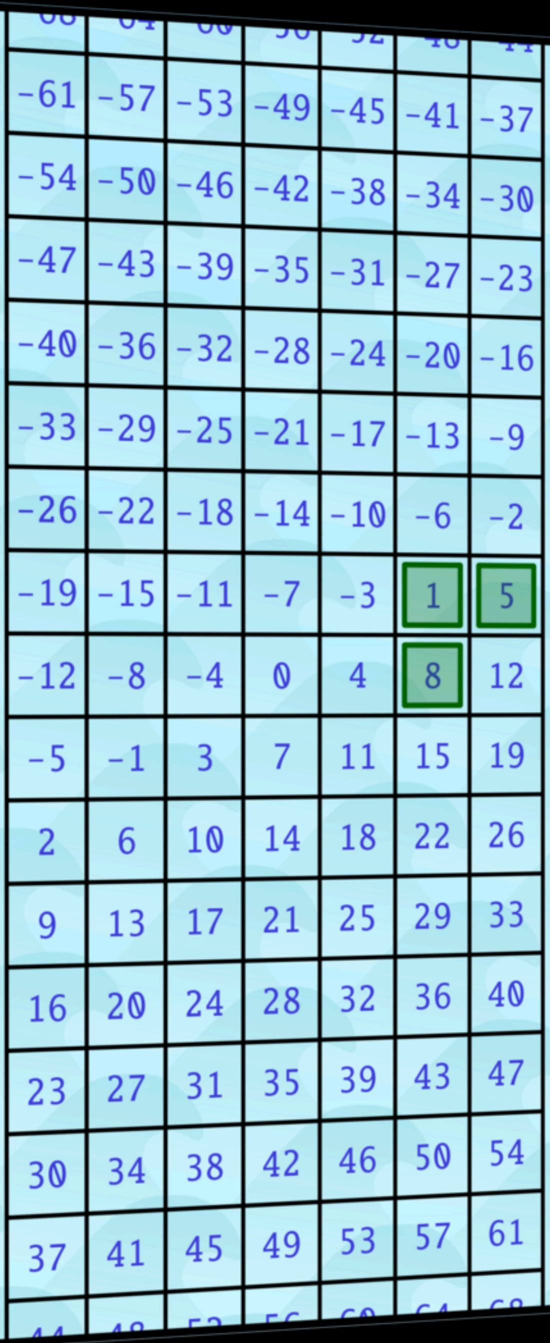
Ships $[0, a, b]$ with a and b coprime.

A disconnected 1-dimensional ship becomes a connected 2-dimensional ship...



Ships $[0, a, b]$ with a and b coprime.

A disconnected 1-dimensional ship becomes a connected 2-dimensional ship...



Ships $[0, a, b]$ with a and b coprime.

A disconnected 1-dimensional ship becomes a connected 2-dimensional ship...

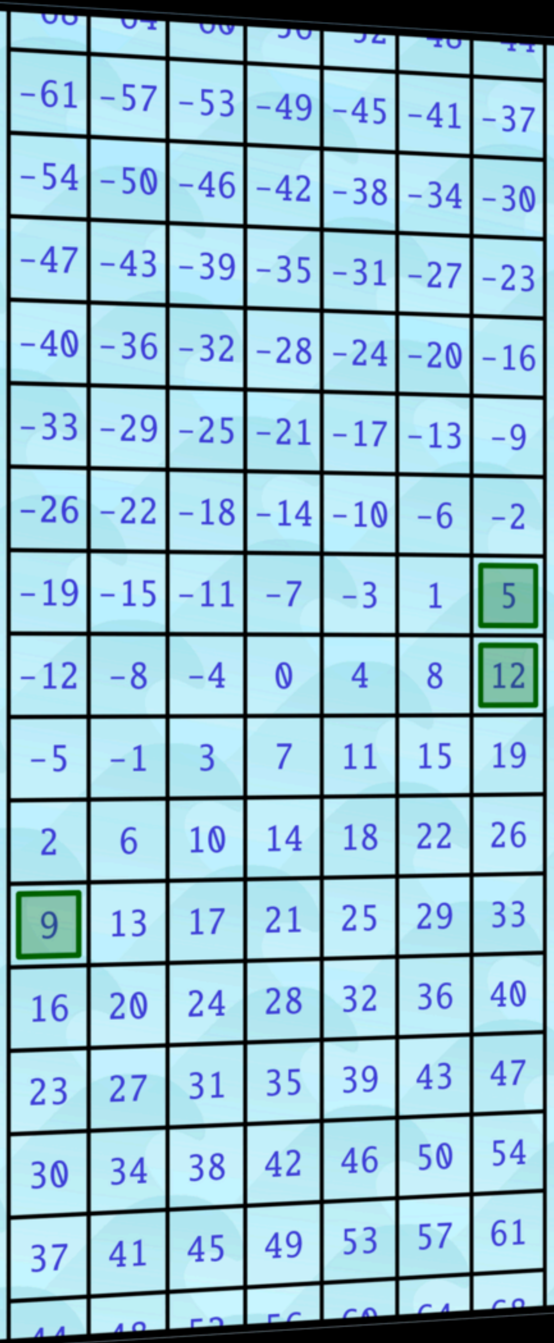
Ships $[0, a, b]$ with a and b coprime.

A disconnected 1-dimensional ship becomes a connected 2-dimensional ship...

Ships $[0, a, b]$ with a and b coprime.

A disconnected 1-dimensional ship becomes a connected 2-dimensional ship...

...except on the border.



Ships $[0, a, b]$ with a and b coprime.

A disconnected 1-dimensional ship becomes a connected 2-dimensional ship...

...except on the border.

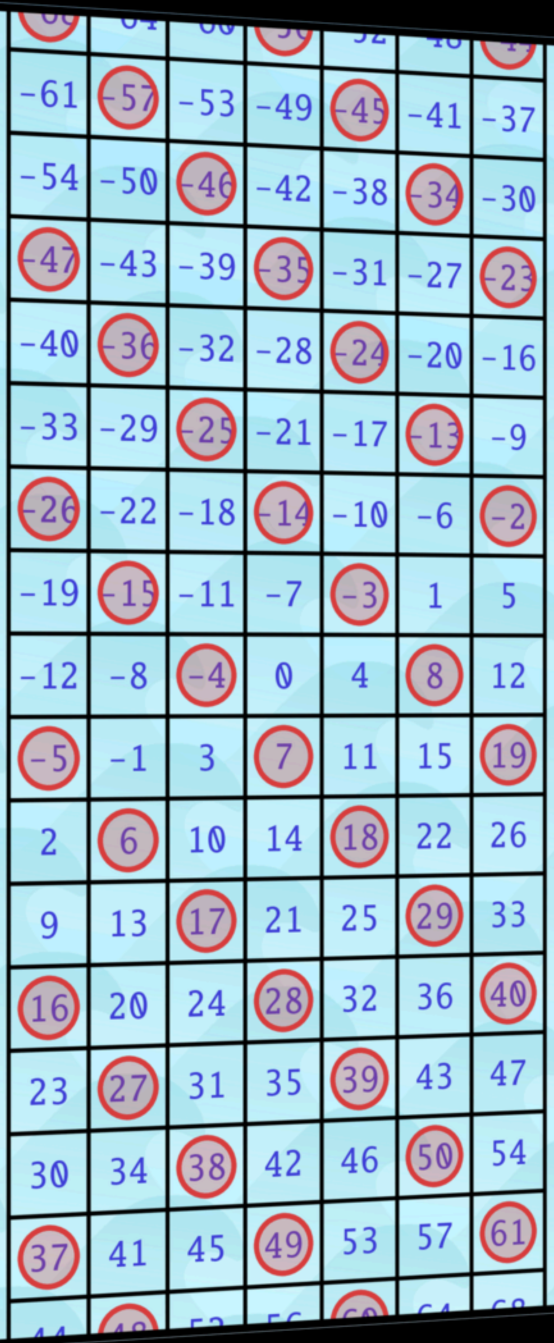
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
-61	-57	-53	-49	-45	-41	-37
-54	-50	-46	-42	-38	-34	-30
-47	-43	-39	-35	-31	-27	-23
-40	-36	-32	-28	-24	-20	-16
-33	-29	-25	-21	-17	-13	-9
-26	-22	-18	-14	-10	-6	-2
-19	-15	-11	-7	-3	1	5
-12	-8	-4	0	4	8	12
-5	-1	3	7	11	15	19
2	6	10	14	18	22	26
9	13	17	21	25	29	33
16	20	24	28	32	36	40
23	27	31	35	39	43	47
30	34	38	42	46	50	54
37	41	45	49	53	57	61
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Ships $[0, a, b]$ with a and b coprime.

A disconnected 1-dimensional ship becomes a connected 2-dimensional ship...

...except on the border.

We know how to optimally hit this ship!

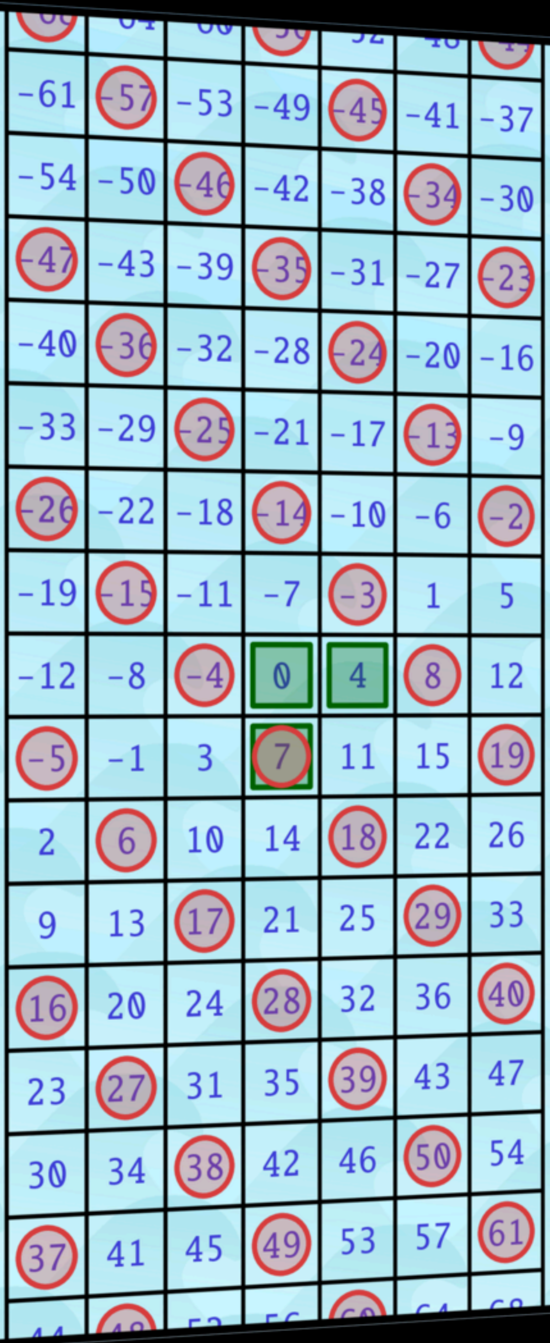


Ships $[0, a, b]$ with a and b coprime.

A disconnected 1-dimensional ship becomes a connected 2-dimensional ship...

...except on the border.

We know how to optimally hit this ship!

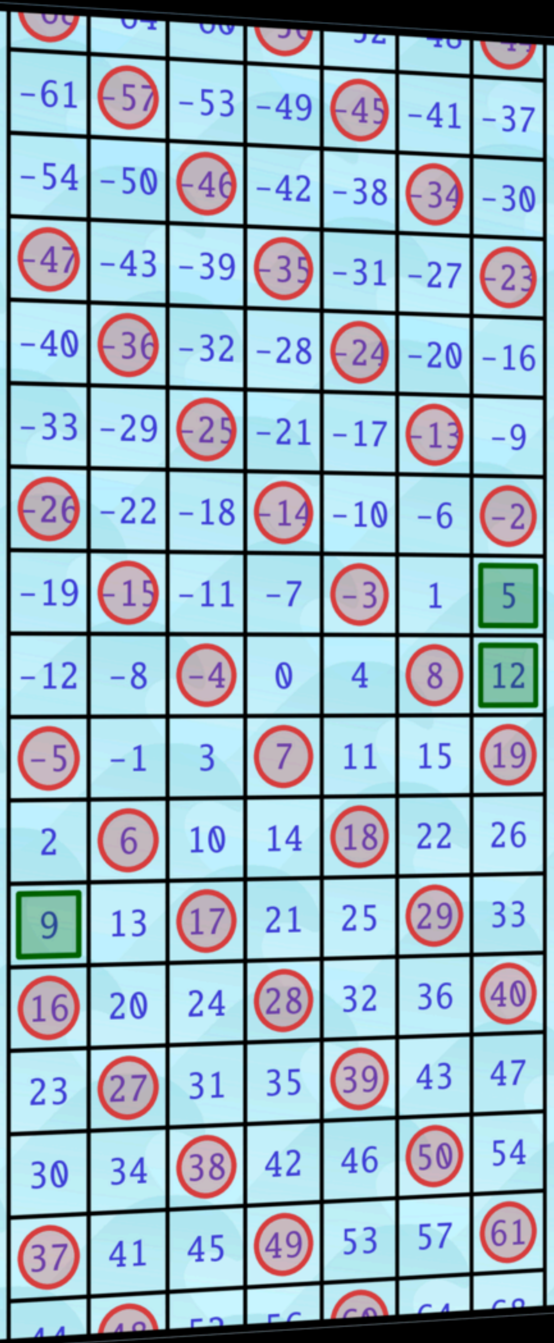


Ships $[0, a, b]$ with a and b coprime.

A disconnected 1-dimensional ship becomes a connected 2-dimensional ship...

...except on the border.

We know how to optimally hit this ship!

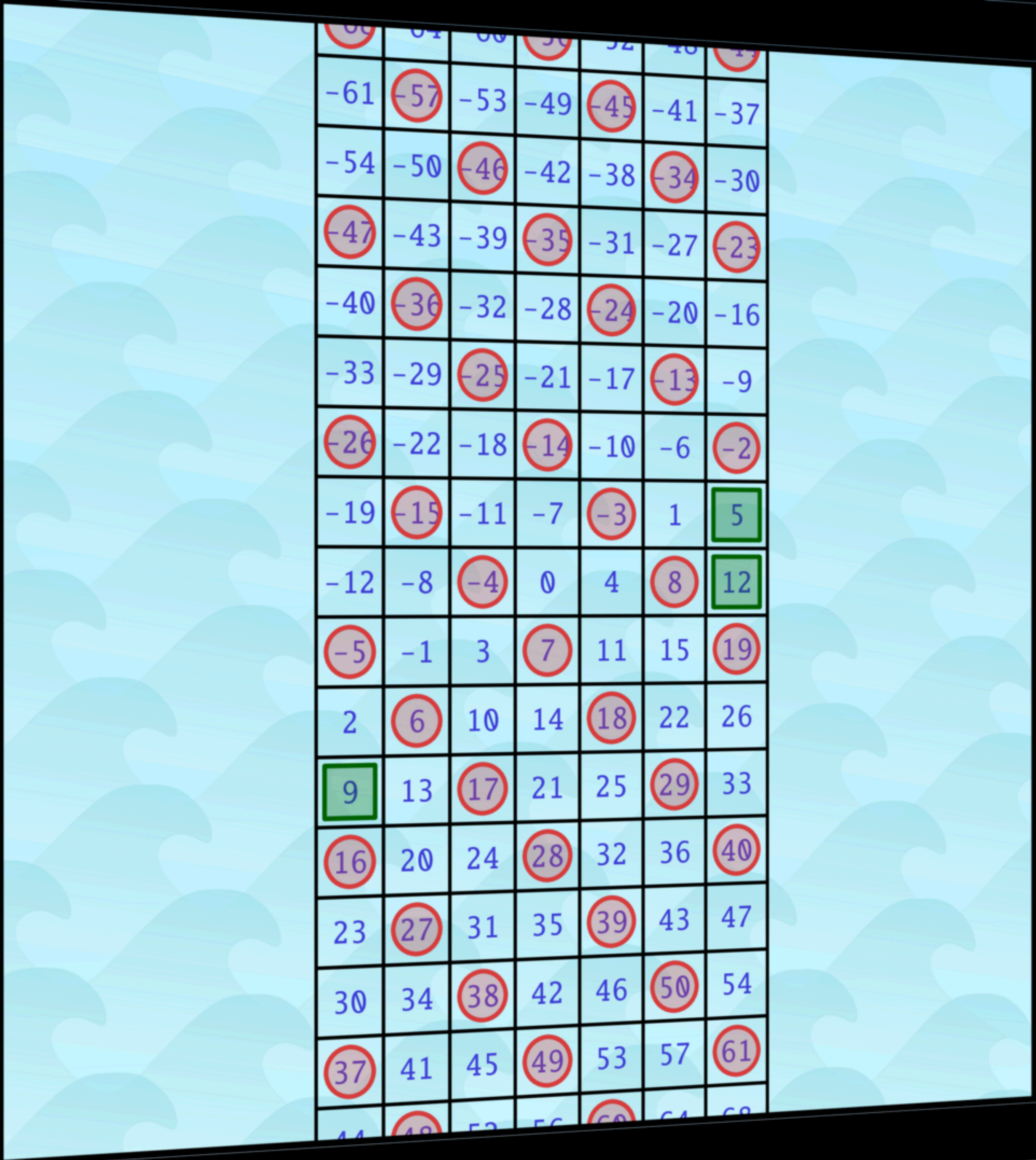


Ships $[0, a, b]$ with a and b coprime.

A disconnected 1-dimensional ship becomes a connected 2-dimensional ship...

...except on the border.

We know how to optimally hit this ship!



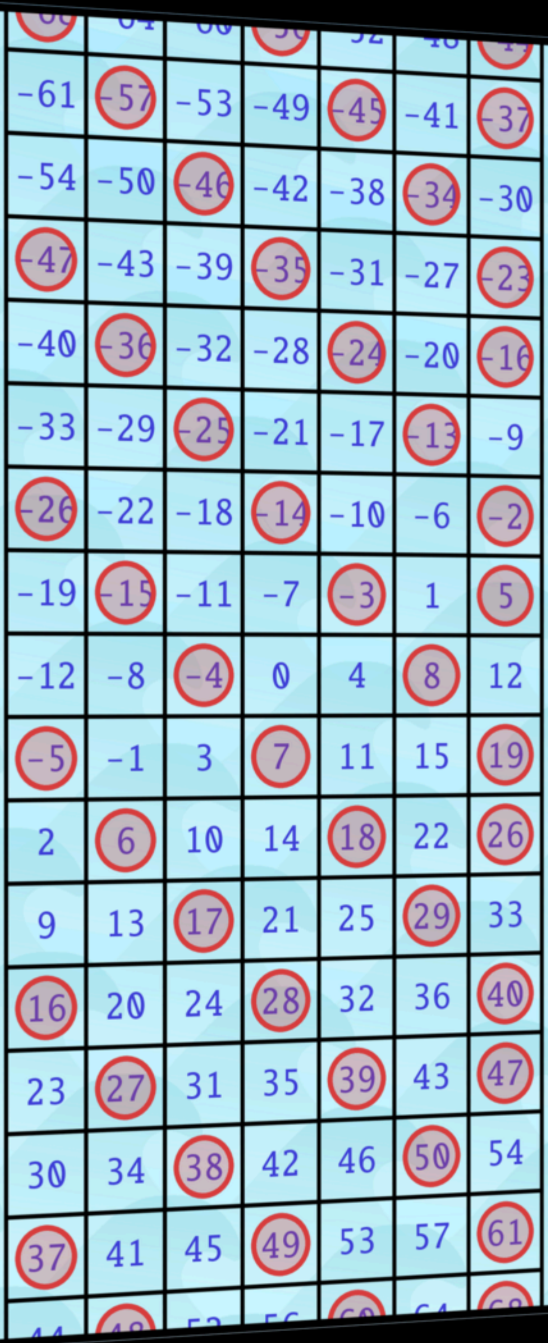
Ships $[0, a, b]$ with a and b coprime.

A disconnected 1-dimensional ship becomes a connected 2-dimensional ship...

...except on the border.

We know how to optimally hit this ship!

We just need a few extra.



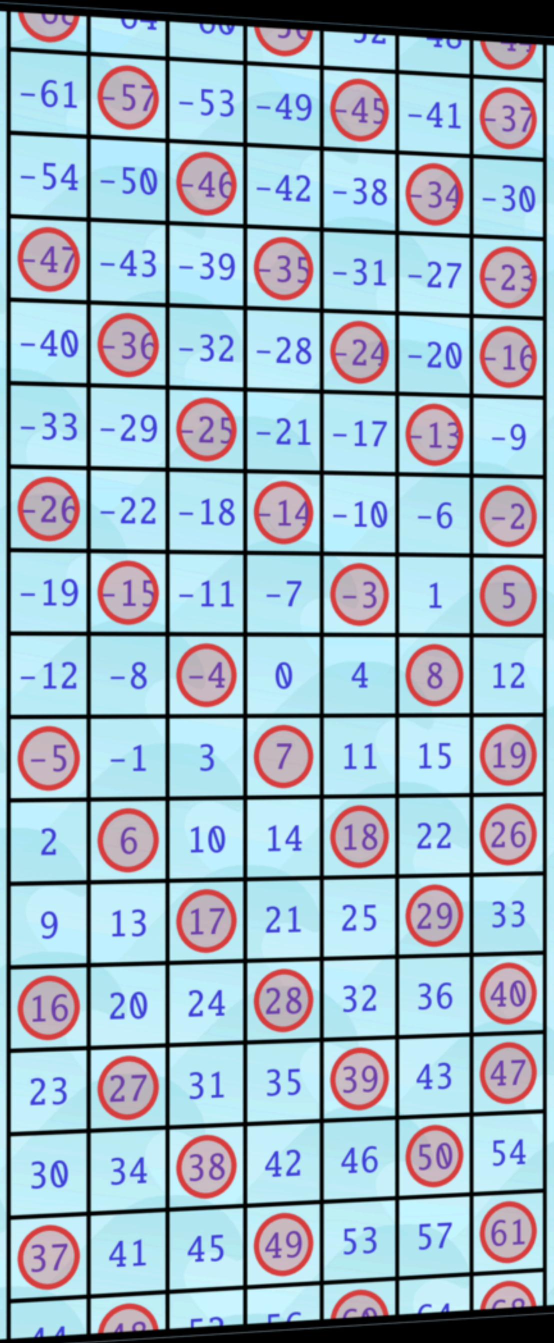
Ships $[0, a, b]$ with a and b coprime.

A disconnected 1-dimensional ship becomes a connected 2-dimensional ship...

...except on the border.

We know how to optimally hit this ship!

We just need a few extra.



Ships $[0, a, b]$ with a and b coprime.

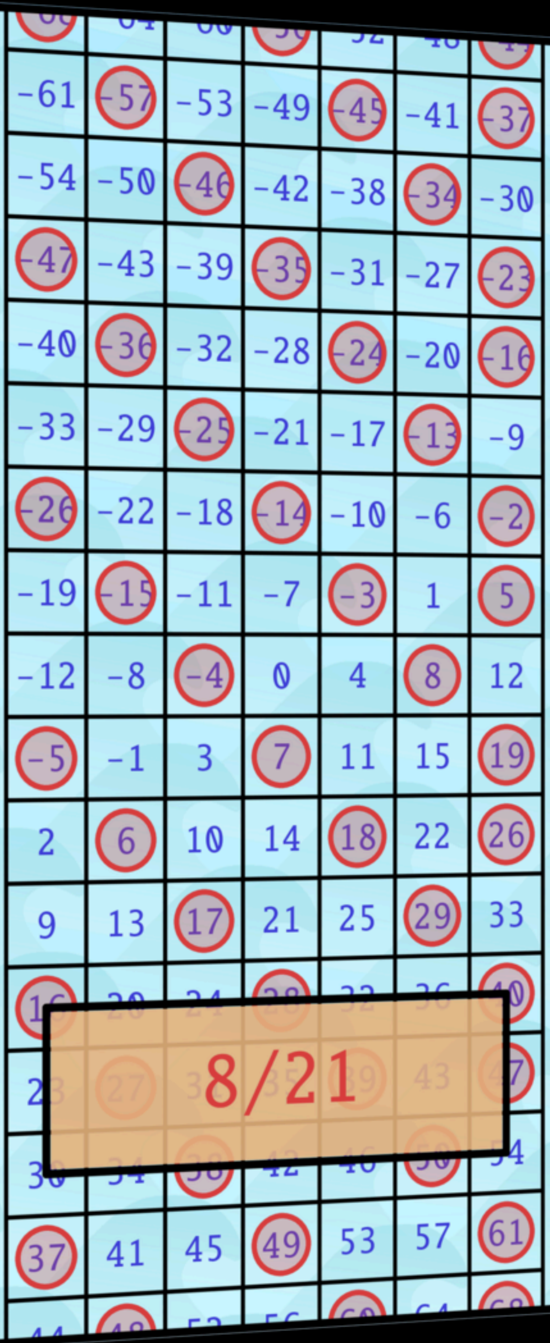
A disconnected 1-dimensional ship becomes a connected 2-dimensional ship...

...except on the border.

We know how to optimally hit this ship!

We just need a few extra.

This pattern works. What is the density?



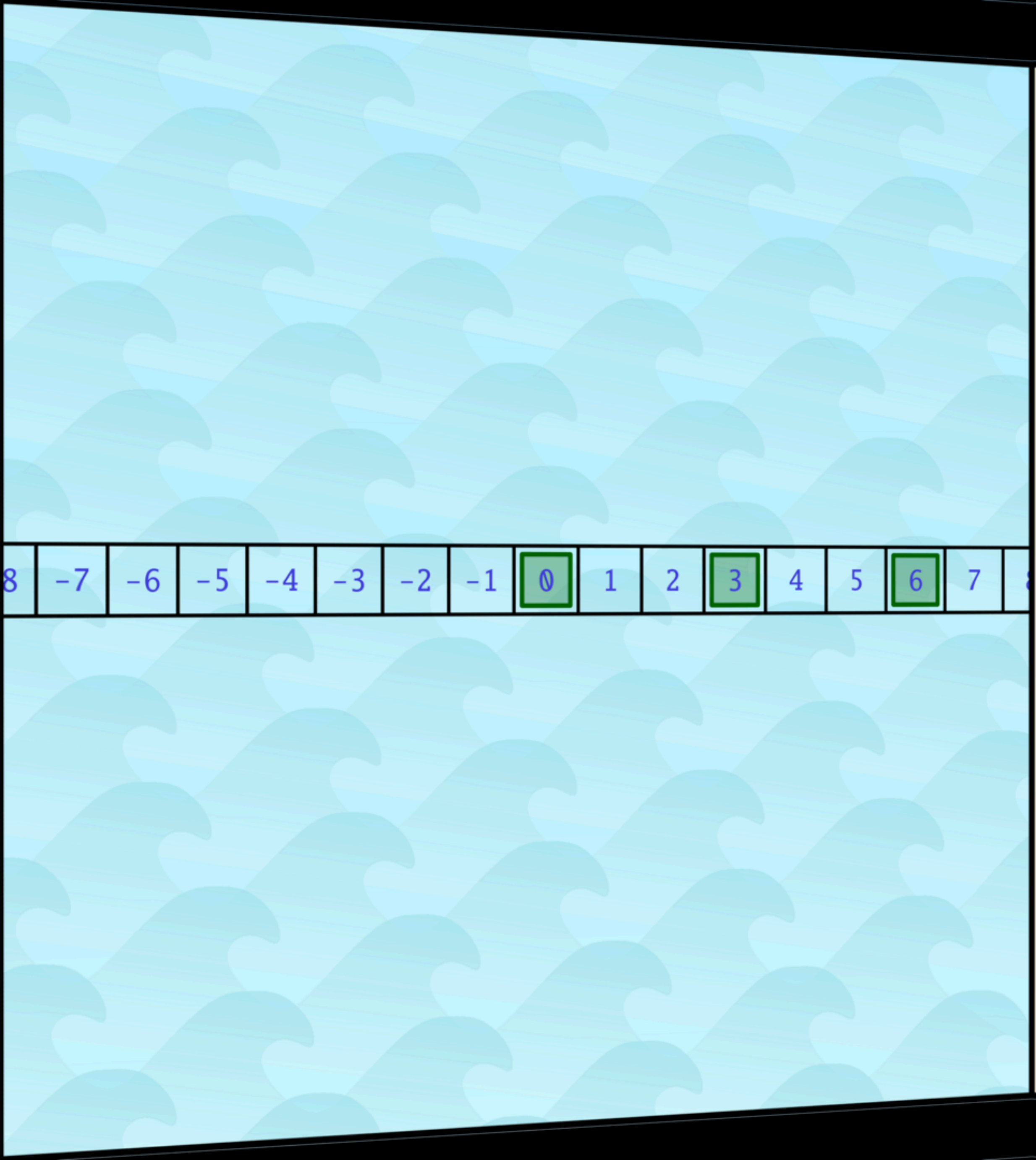
Ships $[0, a, b]$ with a and b coprime.

A disconnected 1-dimensional ship becomes a connected 2-dimensional ship...

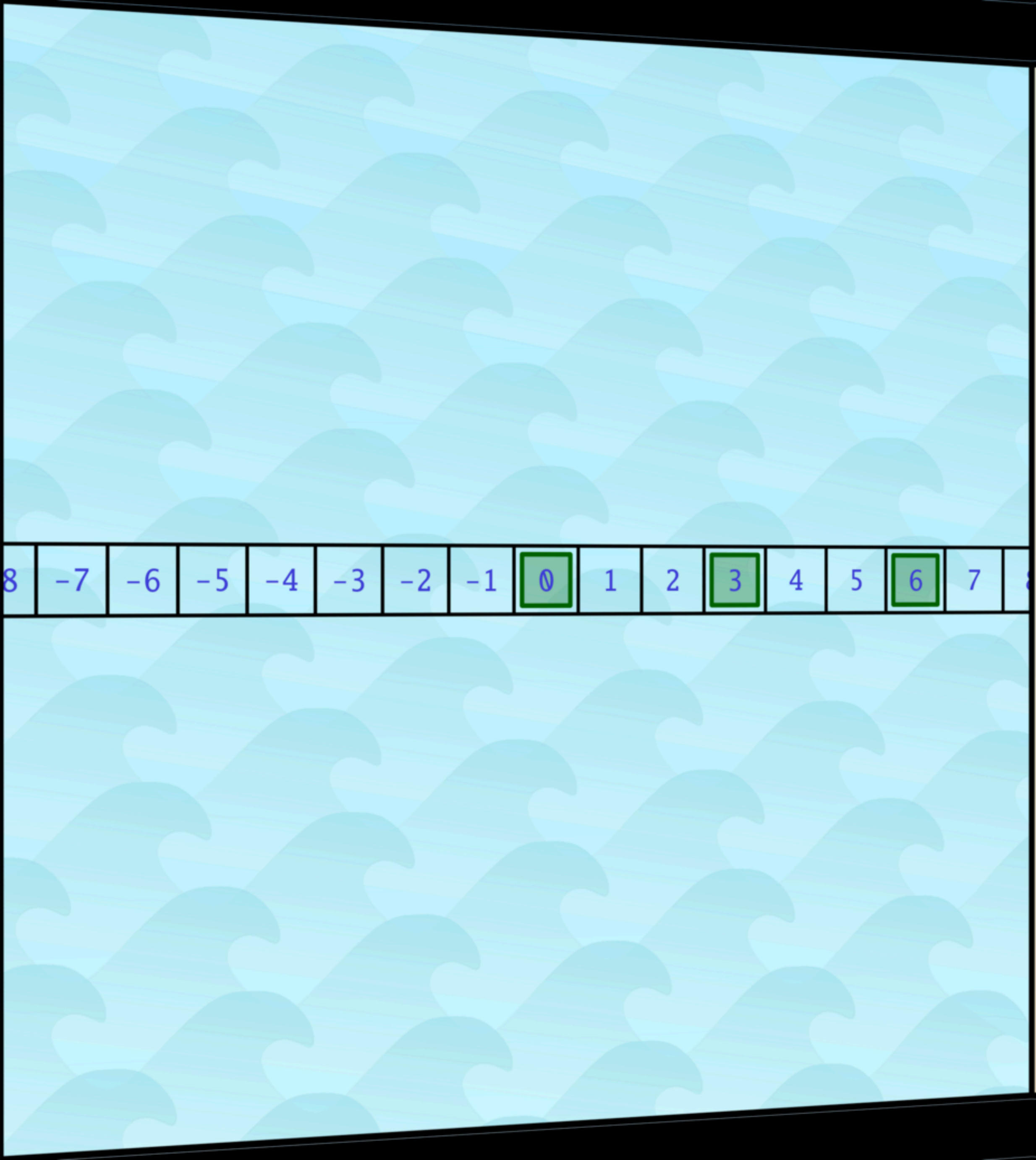
density = $\frac{b-1}{b} \times \frac{1}{3} + \frac{1}{b} \times \frac{2}{3}$

density $\to 1/3$ as $b \to \infty$

Small disconnected ships are harder to hit!

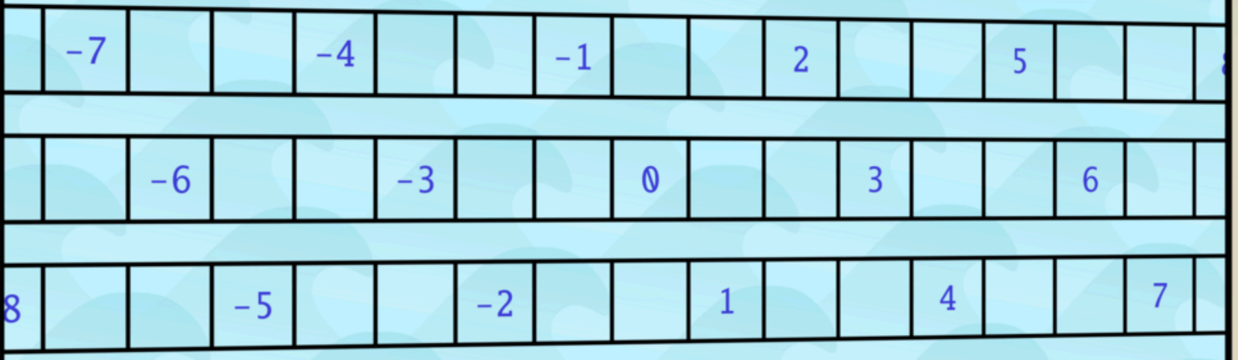


Ships $[0, a, b]$ with a and b *not* coprime



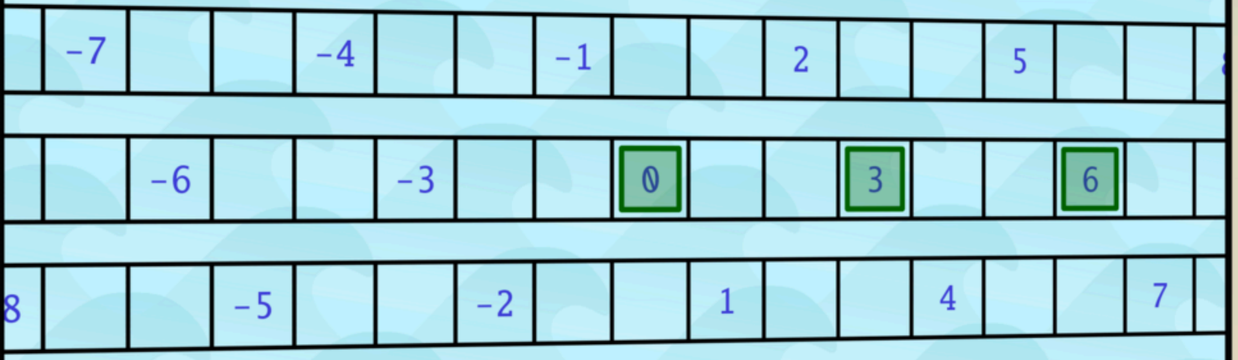
Ships $[0, a, b]$ with a and b *not* coprime

$$a = k \cdot a' \quad b = k \cdot b'$$



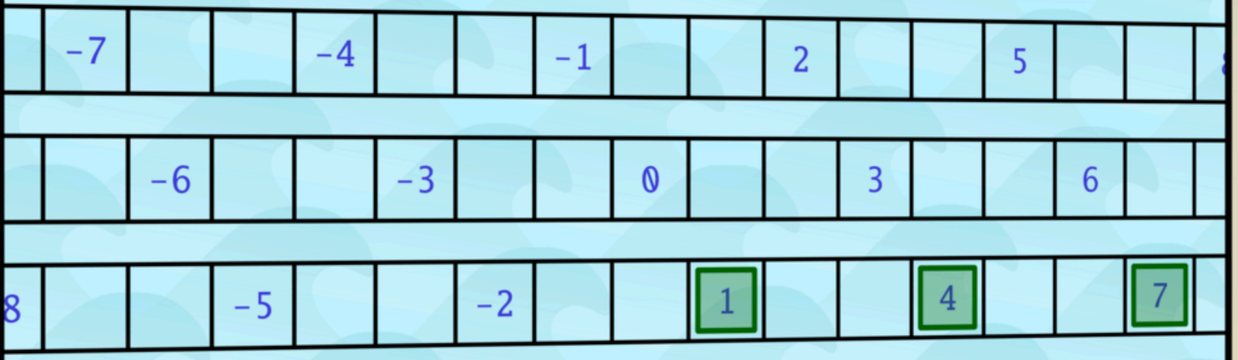
Ships $[0, a, b]$ with a and b *not* coprime

$$a = k \cdot a' \quad b = k \cdot b'$$



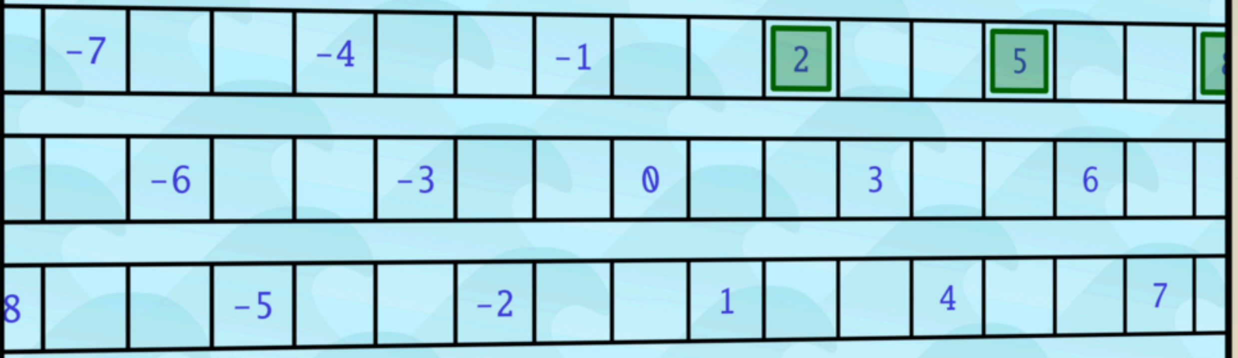
Ships $[0, a, b]$ with a and b *not* coprime

$a = k \cdot a'$ $b = k \cdot b'$



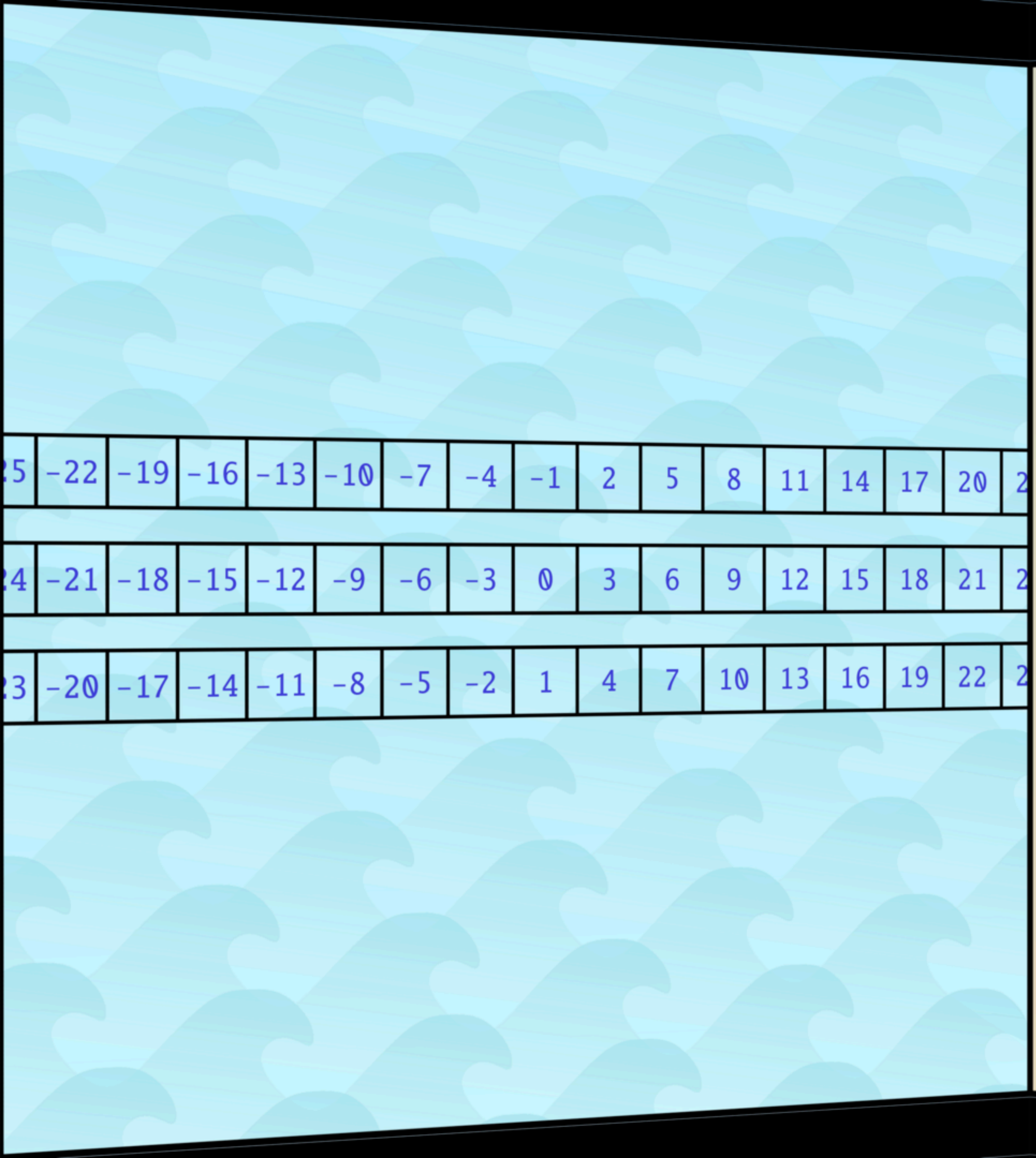
Ships $[0, a, b]$ with a and b *not* coprime

$$a = k \cdot a' \quad b = k \cdot b'$$



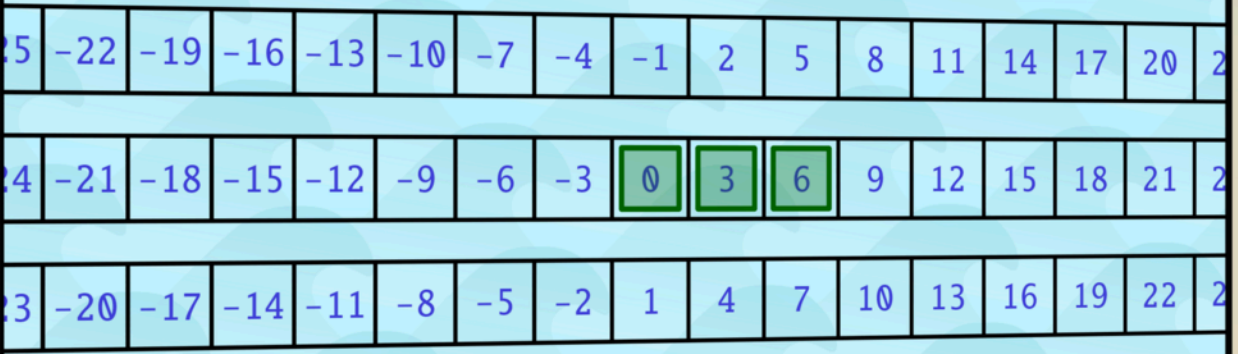
Ships $[0, a, b]$ with a and b *not* coprime

$a = k \cdot a'$ $b = k \cdot b'$



Ships $[0, a, b]$ with a and b *not* coprime

$$a = k \cdot a' \quad b = k \cdot b'$$



Ships $[0, a, b]$ with a and b *not* coprime

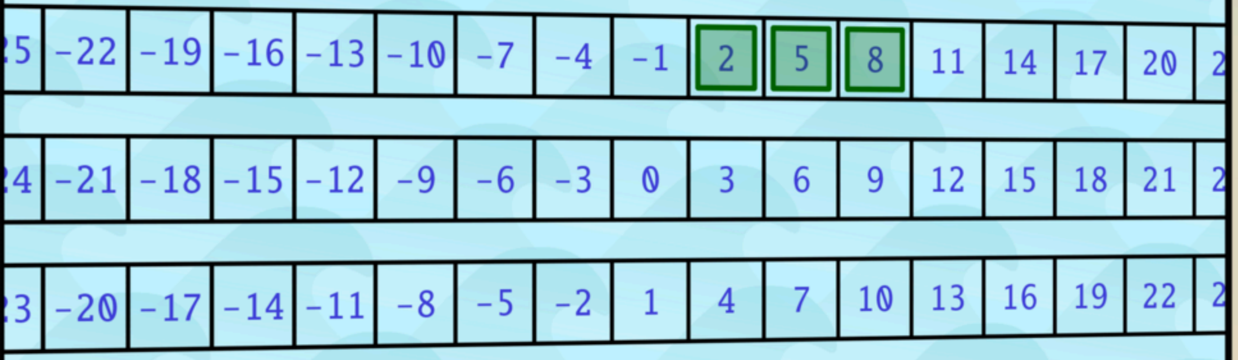
$$a = k \cdot a' \quad b = k \cdot b'$$

Ships $[0, a, b]$ with a and b *not* coprime

$$a = k \cdot a' \quad b = k \cdot b'$$

Ships $[0, a, b]$ with a and b *not* coprime

$a = k \cdot a' \;\; b = k \cdot b'$



Ships $[0, a, b]$ with a and b *not* coprime

$a = k \cdot a' \;\; b = k \cdot b'$

Same density as $[0, a', b']$!

CHAPTER 5

CONTRIBUTION

Theorem 1 *Given a ship S with span s, the density of S can be computed in time polynomial in 2^s.*

Theorem 2 $S = \{[0, da], [0, db]\}$, a, b *coprime. Density is*

$$\begin{cases} 1/2 & \text{if } a \text{ and } b \text{ odd,} \\ \frac{a+b+1}{2(a+b)} & \text{otherwise.} \end{cases}$$

Theorem 3 *The toughest set of n shapes of size 2 has density $n/(n+1)$.*

Theorem 4 *Let S be a 3-shape and its reflection. Then the density is $\leq \frac{2}{5}$.*

Theorem 5 *The simplest set of n shapes of size k has density $\frac{1}{k}$.*

Theorem 6 *The toughest set of n shapes of size k has density d where*

$$1 - \frac{k-1}{\sqrt[k]{nk!}} \leq d \leq \min\left\{\frac{n}{n+1}, \frac{1+\log(kn)}{k}\right\}$$

Generalizations to 2D

CHAPTER 6

OPEN PROBLEMS

Problem 1.

Is a 1-dimensional optimal solution always periodic?

Problem 1.
Is a 1-dimensional optimal solution always periodic?

Problem 1.
Is a 1-dimensional optimal solution always periodic?

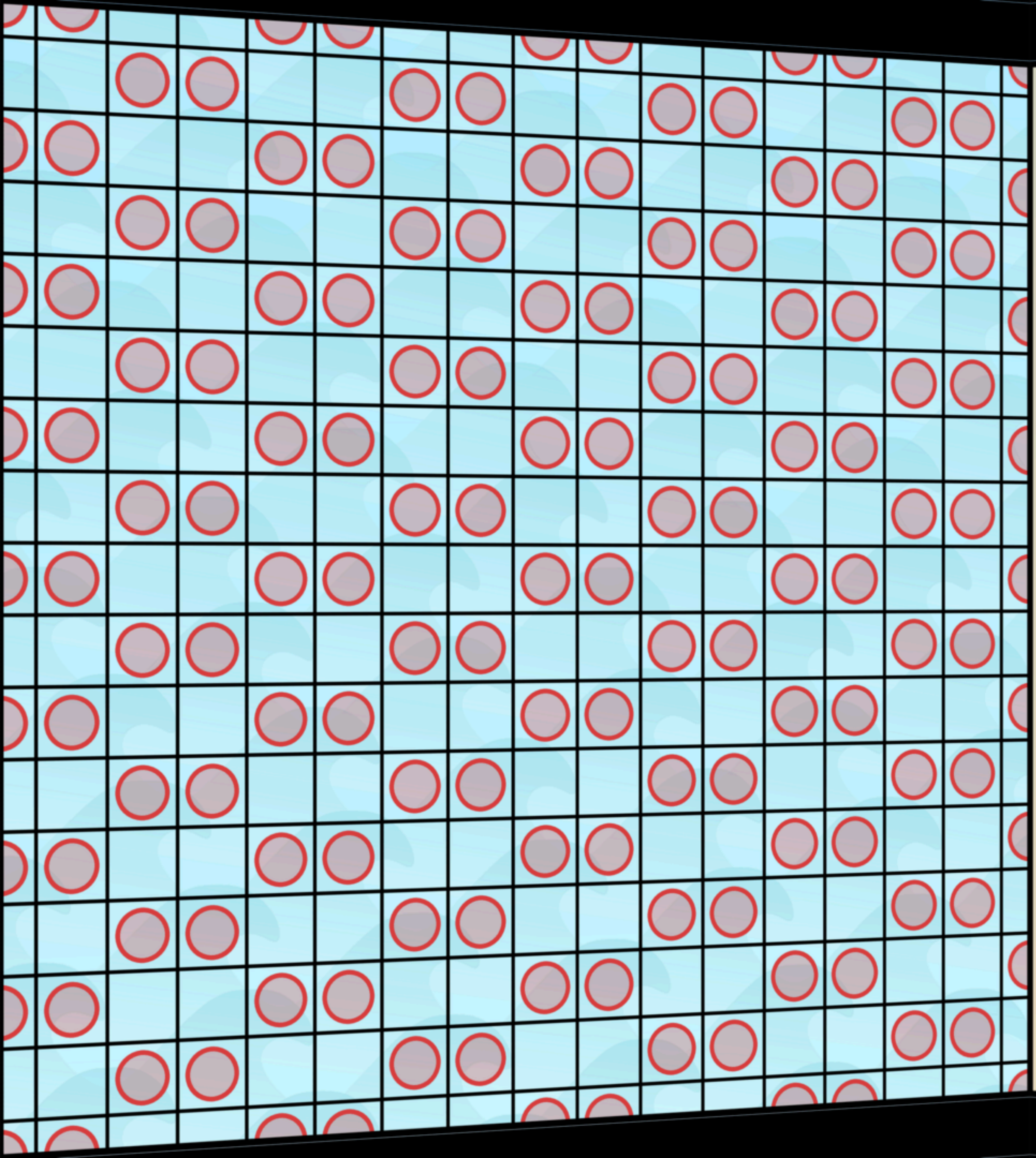
Problem 1.
Is a 1-dimensional optimal solution always periodic?

Short period would imply possible algorithms.

Problem 1.
Is a 1-dimensional optimal solution always periodic?

Short period would imply possible algorithms.

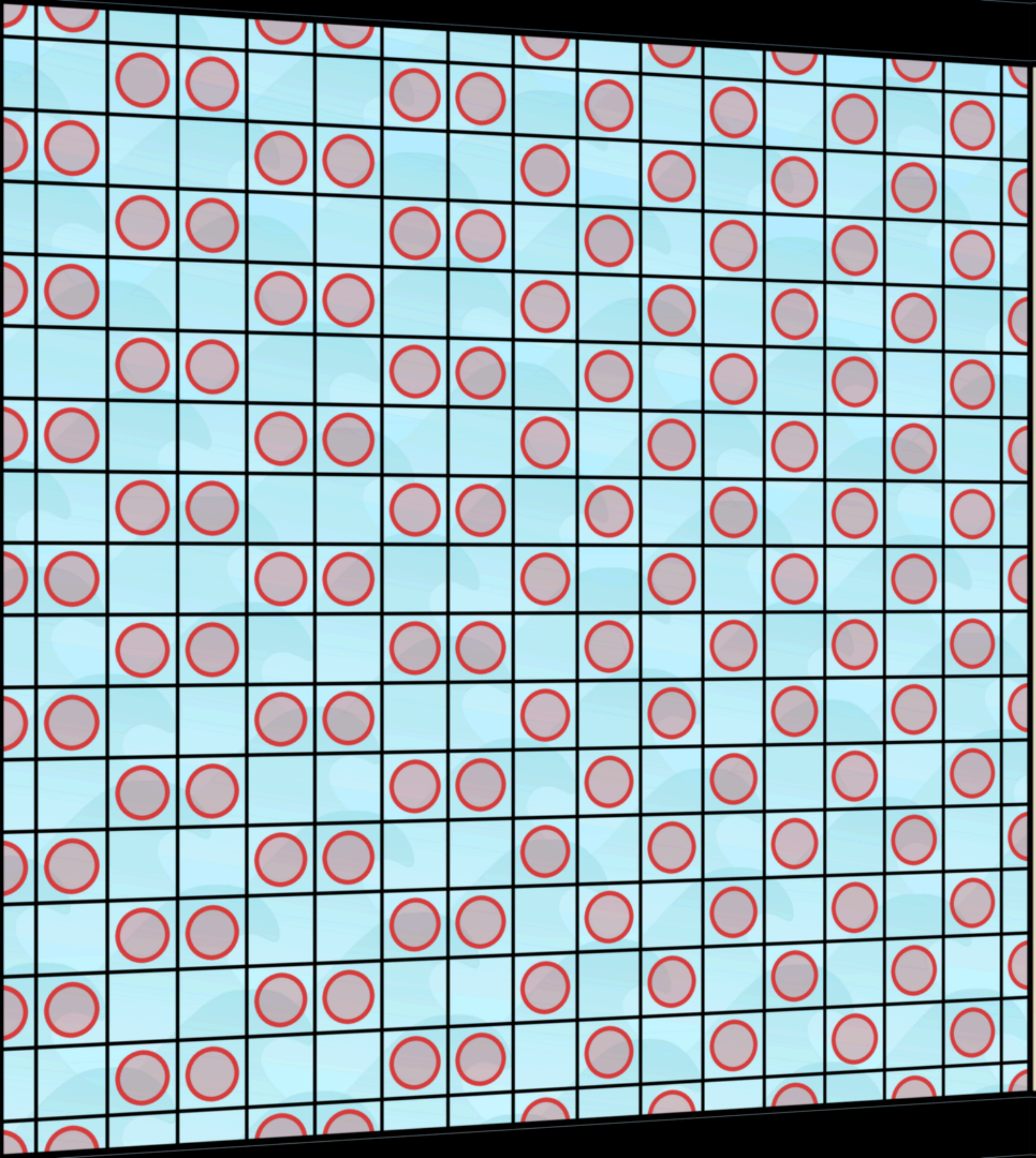
What about 2-dimensional optimal solutions?



Problem 1.
Is a 1-dimensional optimal solution always periodic?

Short period would imply possible algorithms.

What about 2-dimensional optimal solutions?



Problem 1.
Is a 1-dimensional optimal solution always periodic?

Short period would imply possible algorithms.

What about 2-dimensional optimal solutions?

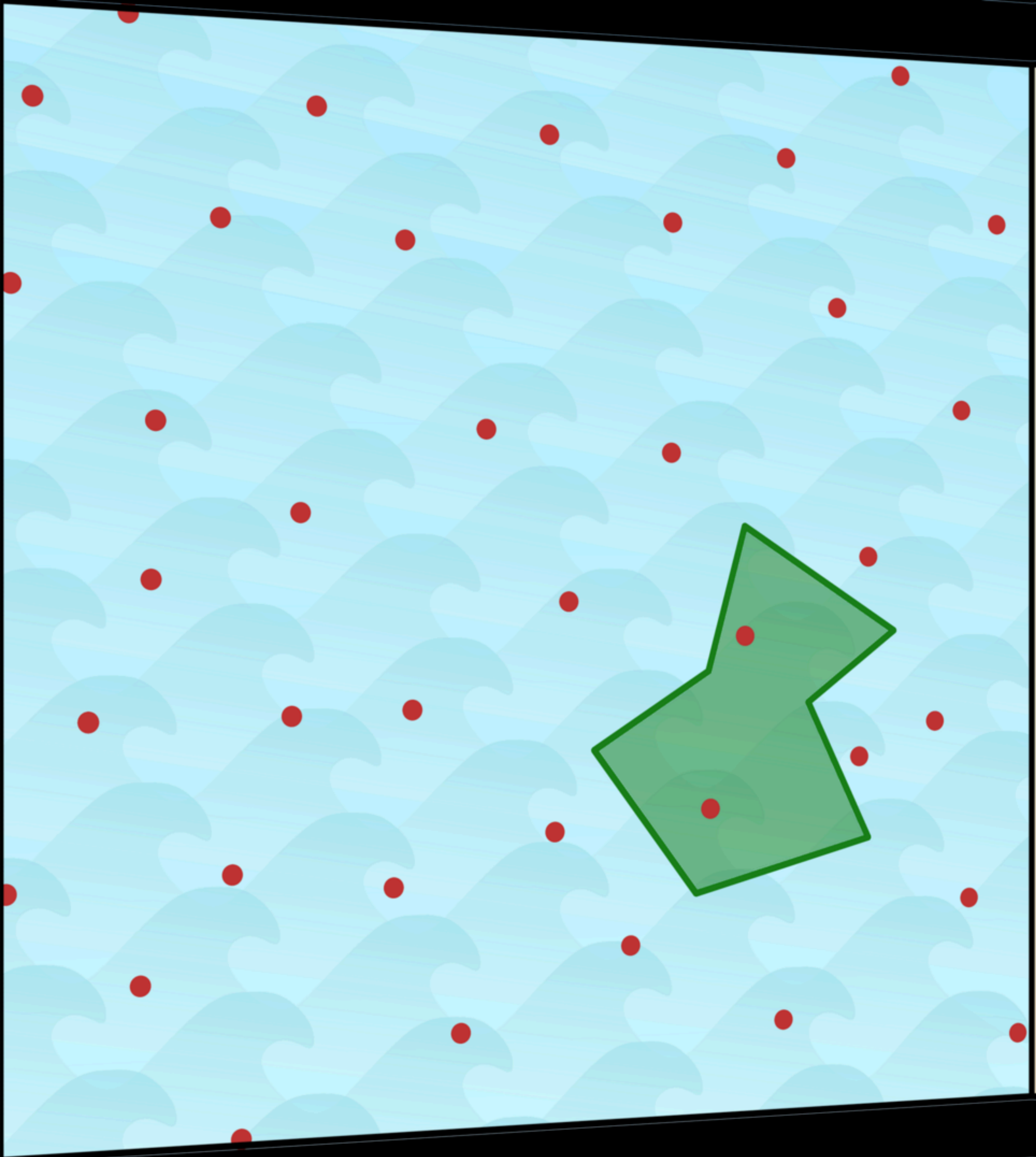
Problem 2.
How does any of this insight help us with the original (continuous) problem?

Problem 2.
How does any of this insight help us with the original (continuous) problem?

E.g.: given a polygon, what is the best point set to hit it no matter where it is hiding.

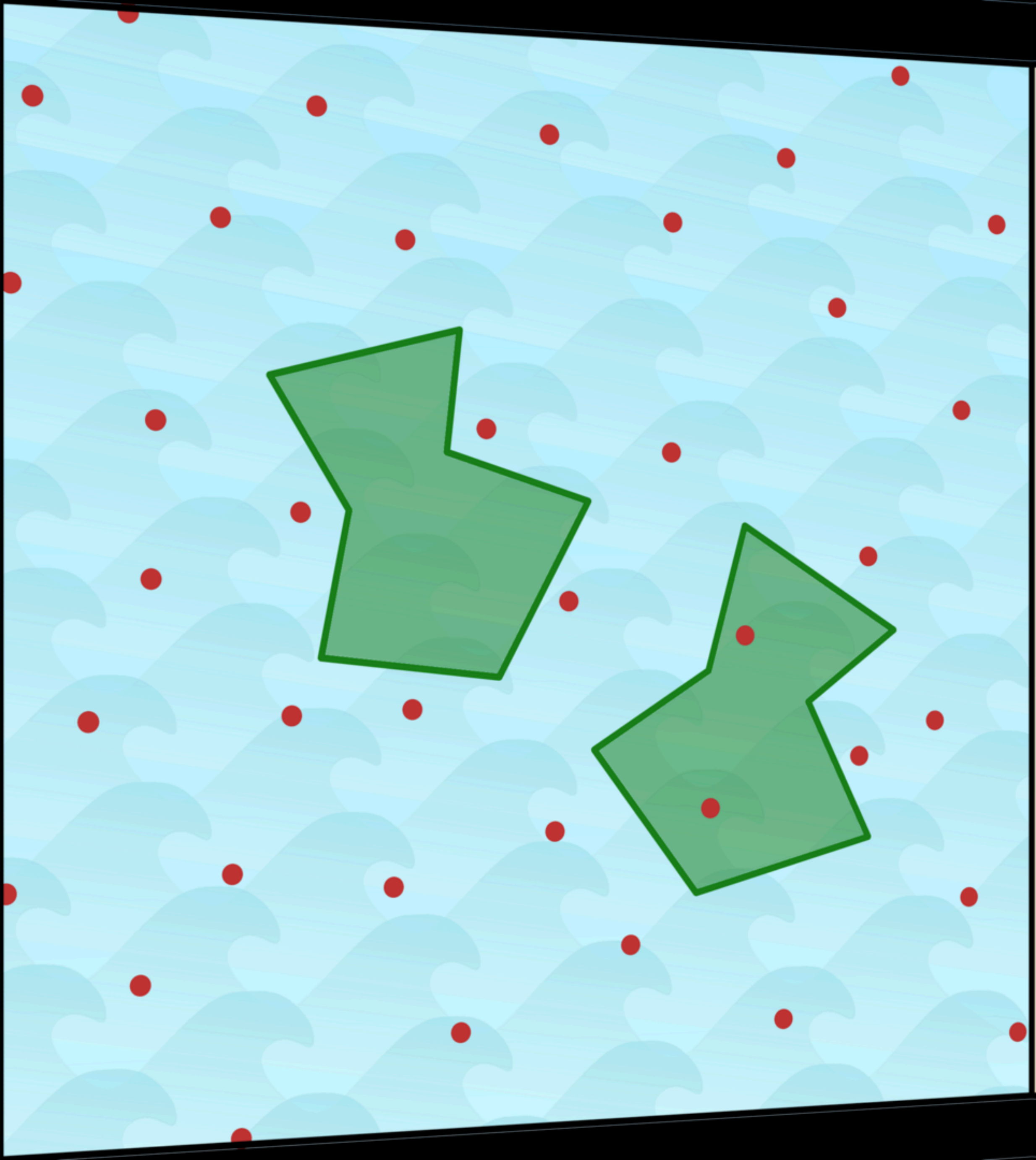
Problem 2.
How does any of this insight help us with the original (continuous) problem?

E.g.: given a polygon, what is the best point set to hit it no matter where it is hiding.



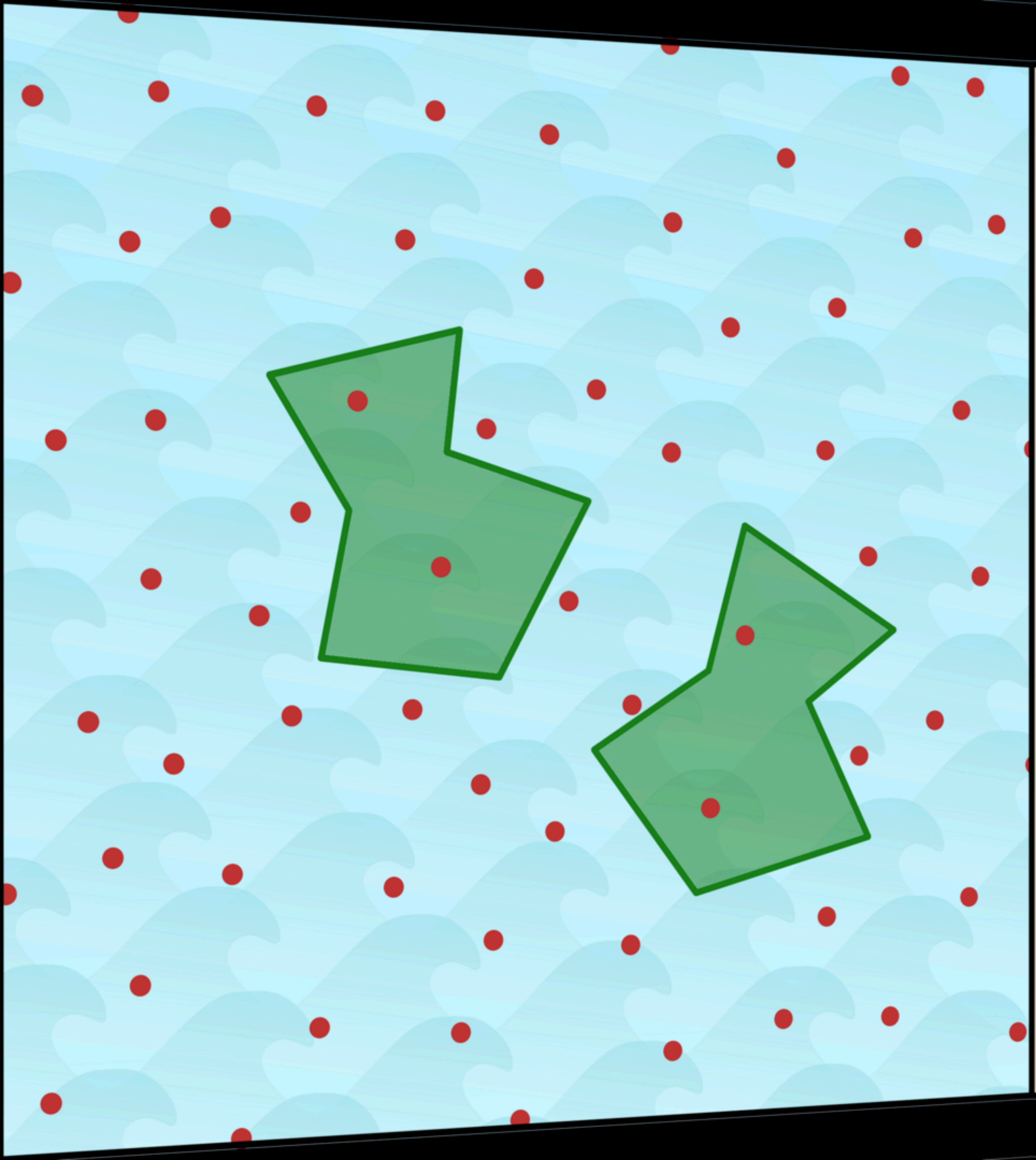
Problem 2.
How does any of this insight help us with the original (continuous) problem?

E.g.: given a polygon, what is the best point set to hit it no matter where it is hiding.



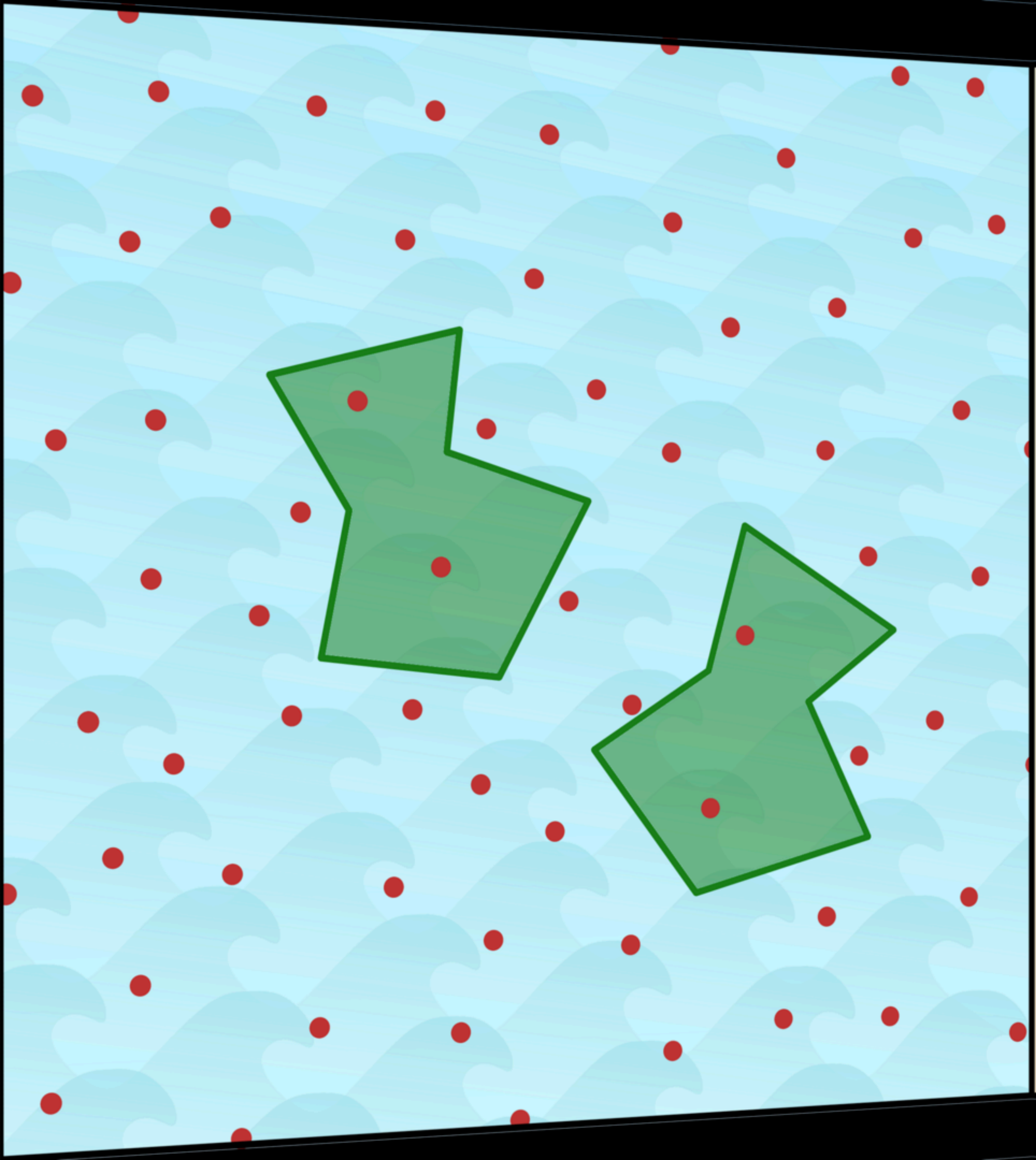
Problem 2.
How does any of this insight help us with the original (continuous) problem?

E.g.: given a polygon, what is the best point set to hit it no matter where it is hiding.



Problem 2.
How does any of this insight help us with the original (continuous) problem?

E.g.: given a polygon, what is the best point set to hit it no matter where it is hiding.



Problem 2.
How does any of this insight help us with the original (continuous) problem?

E.g.: given a polygon, what is the best point set to hit it no matter where it is hiding.

Not much is known!

Geometry,
Uncertainty,
&
BATTLESHIPS!!!

Eva Hainzl
Maarten Löffler
Daniel Perz
Josef Tkadlec