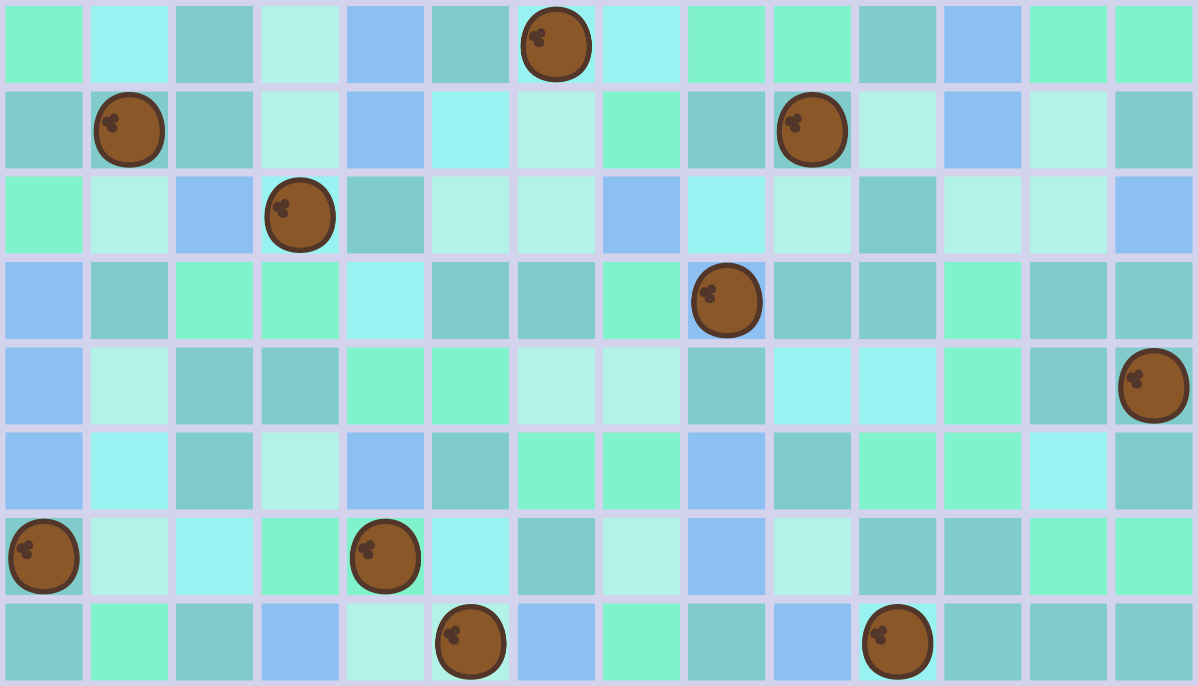
TRASH COMPACTION

Hugo Akitaya ○ Greg Aloupis ○ Maarten Löffler ○ Anika Rounds

Coconut Compaction

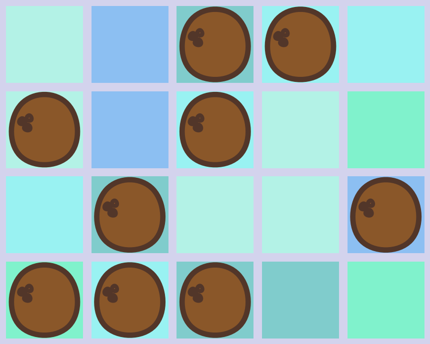


Hugo Akitaya
Greg Aloupis
Maarten Löffler
Anika Rounds

COCONUT BASICS

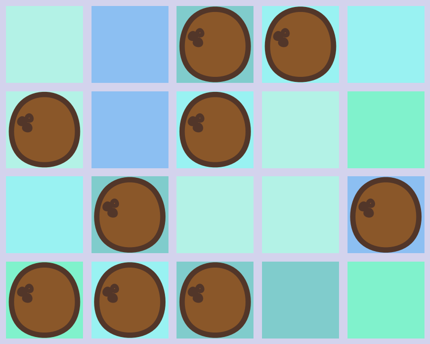
Let P be a set of n coconuts in a swimming pool.

Let P be a set of n coconuts in a swimming pool.



Let P be a set of n coconuts in a swimming pool.

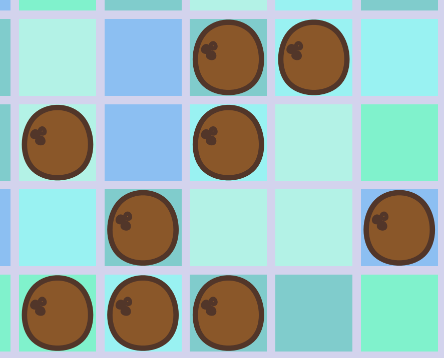
We assume the coconuts are aligned to a grid.



Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

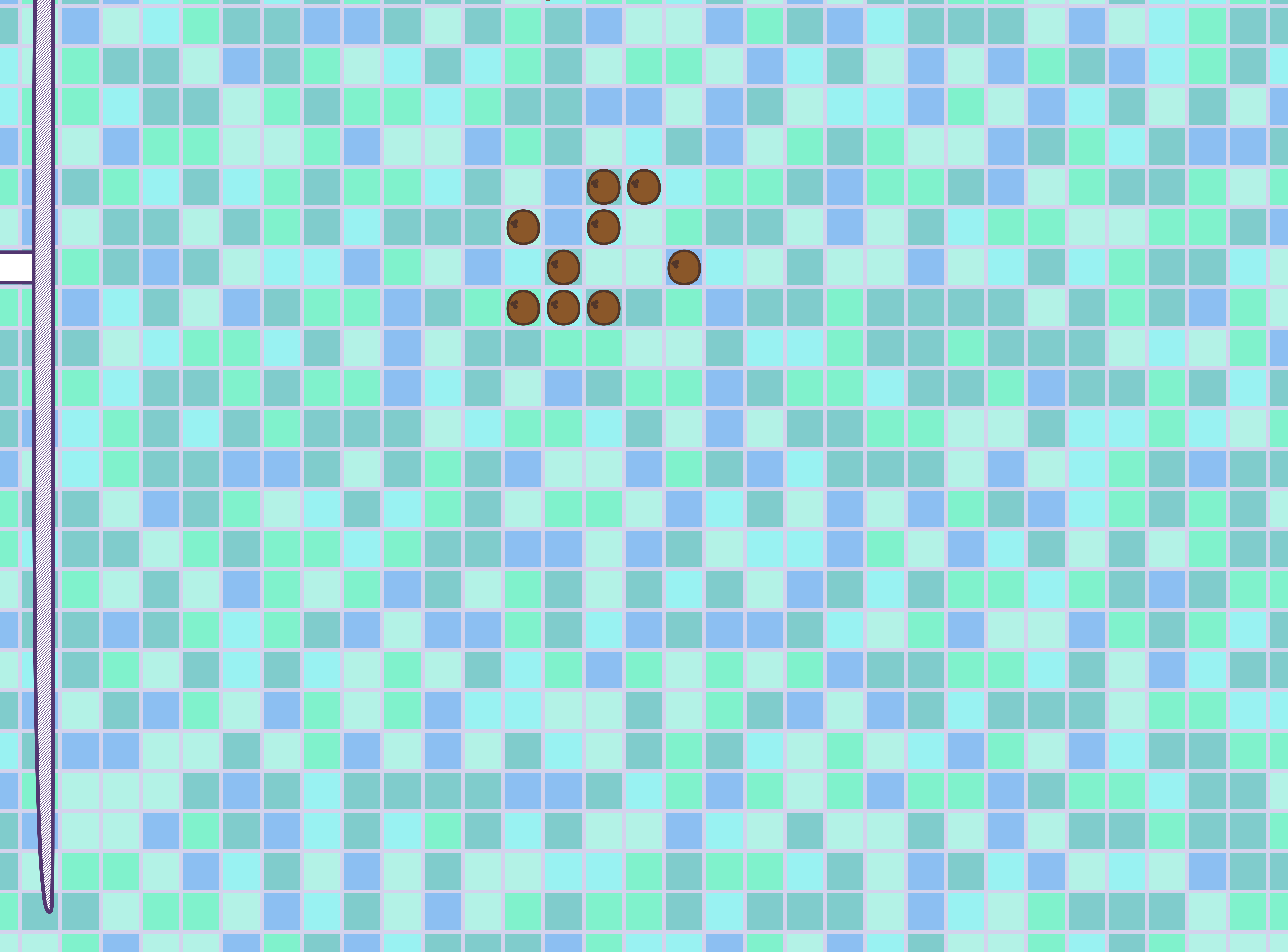
Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.



Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

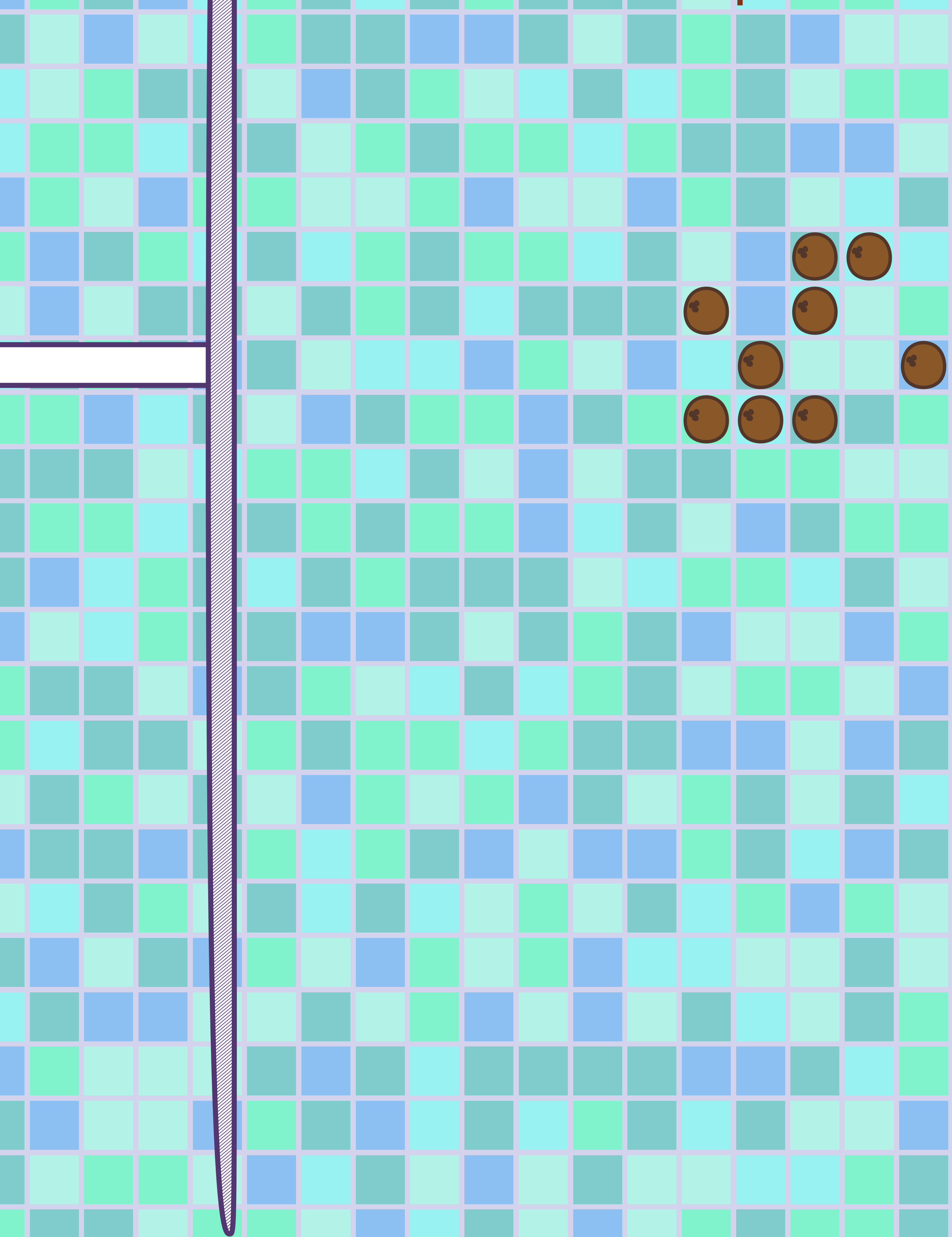
Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.



Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

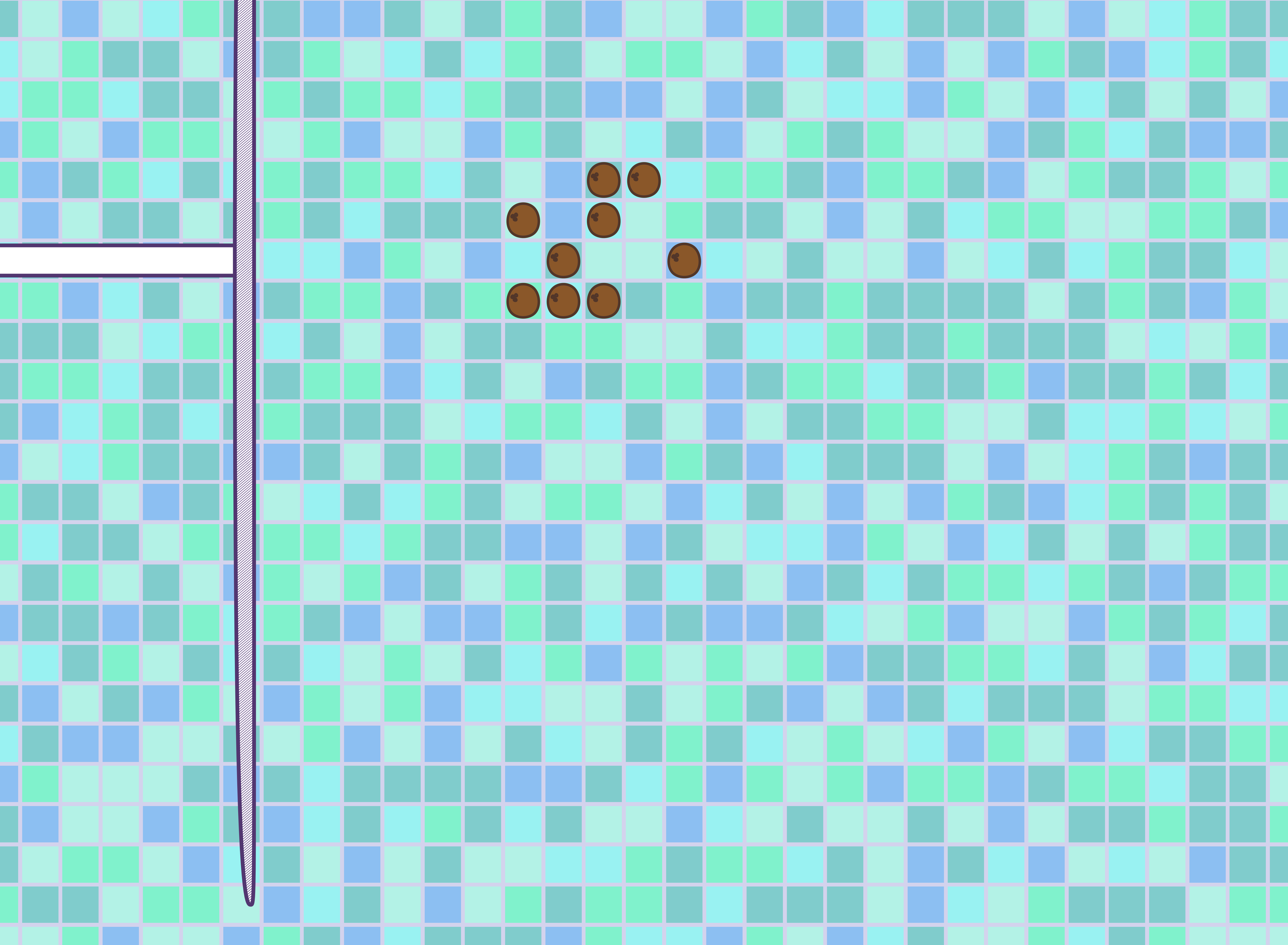
Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.



Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

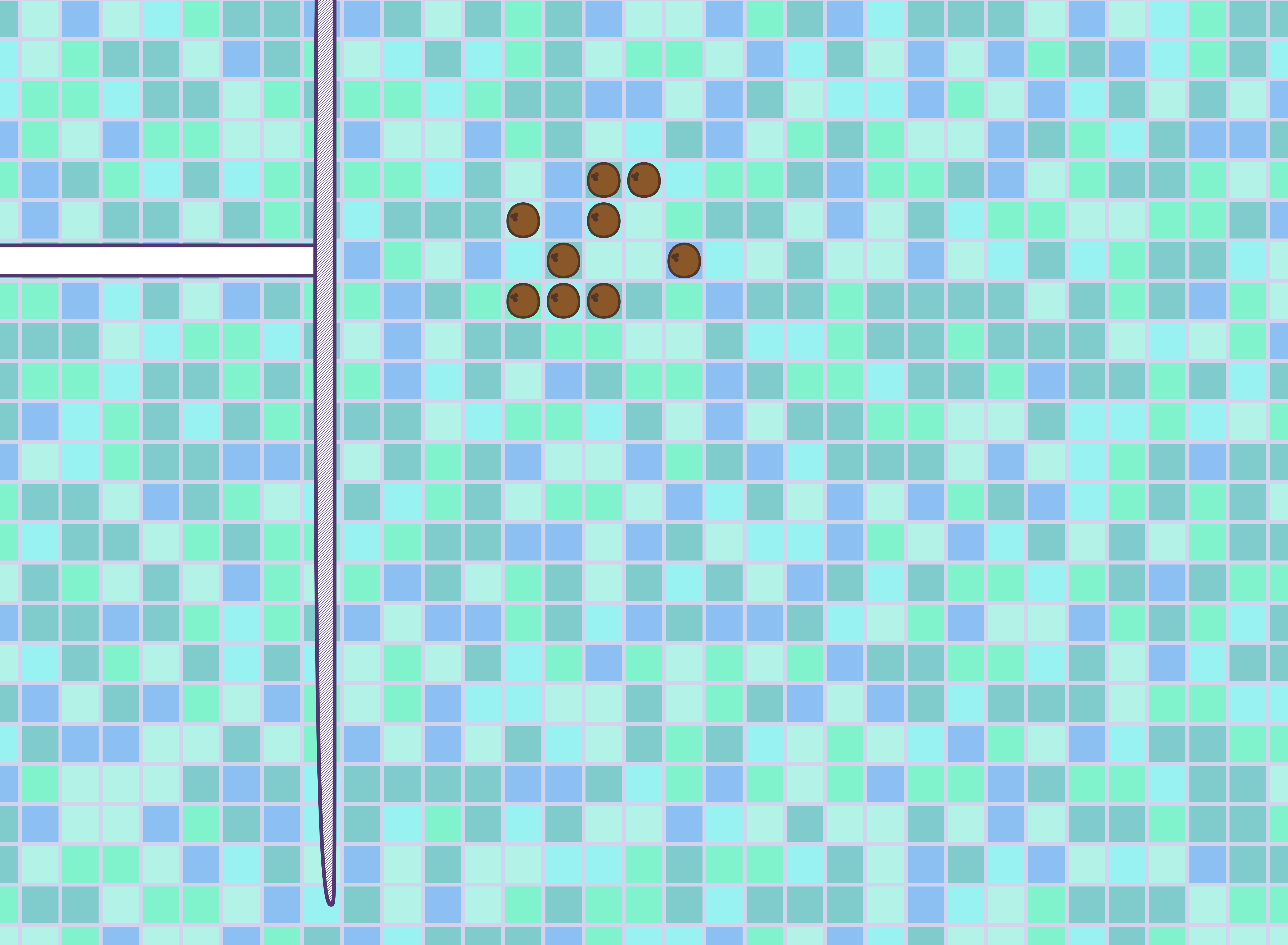
Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.



Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

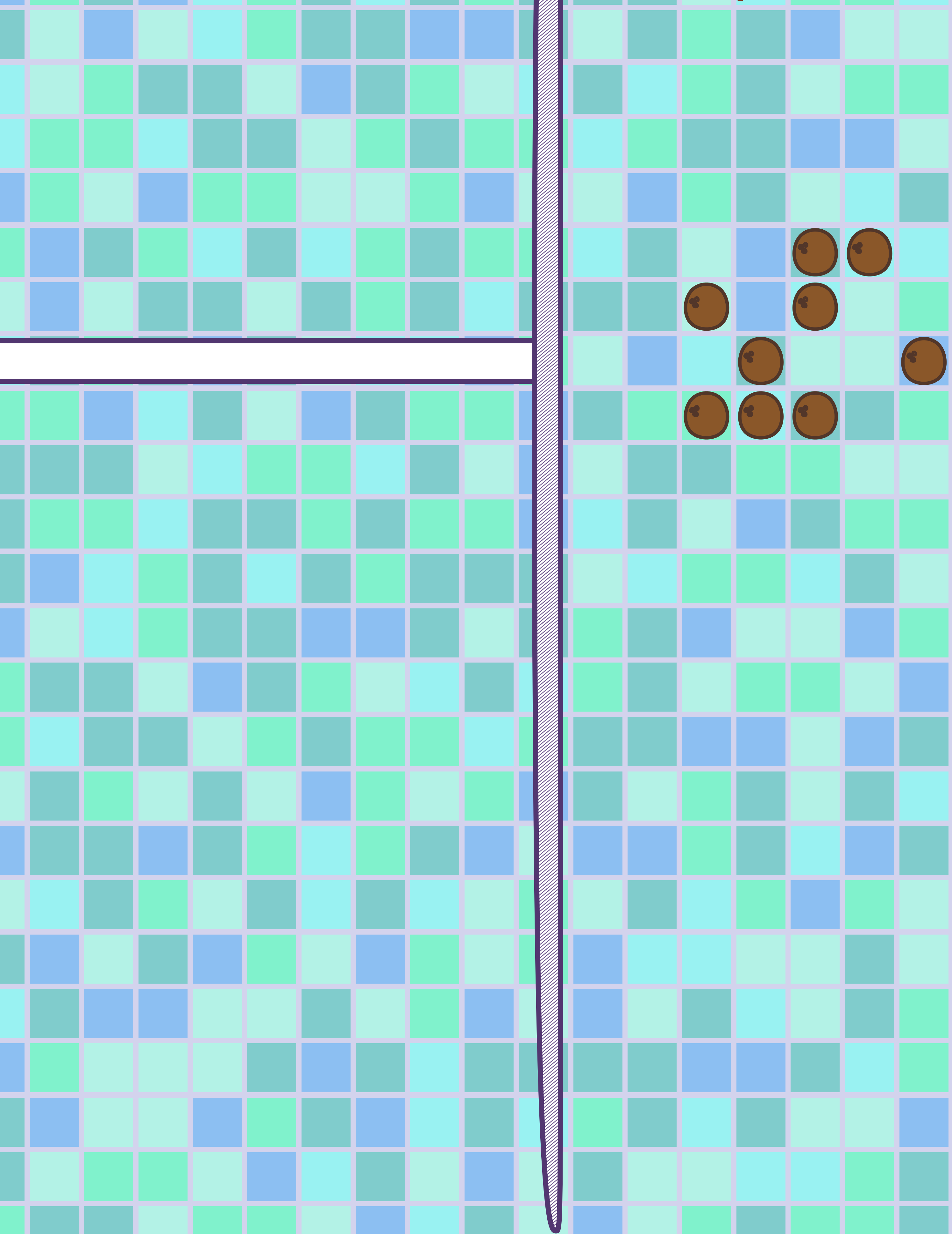
Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.



Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

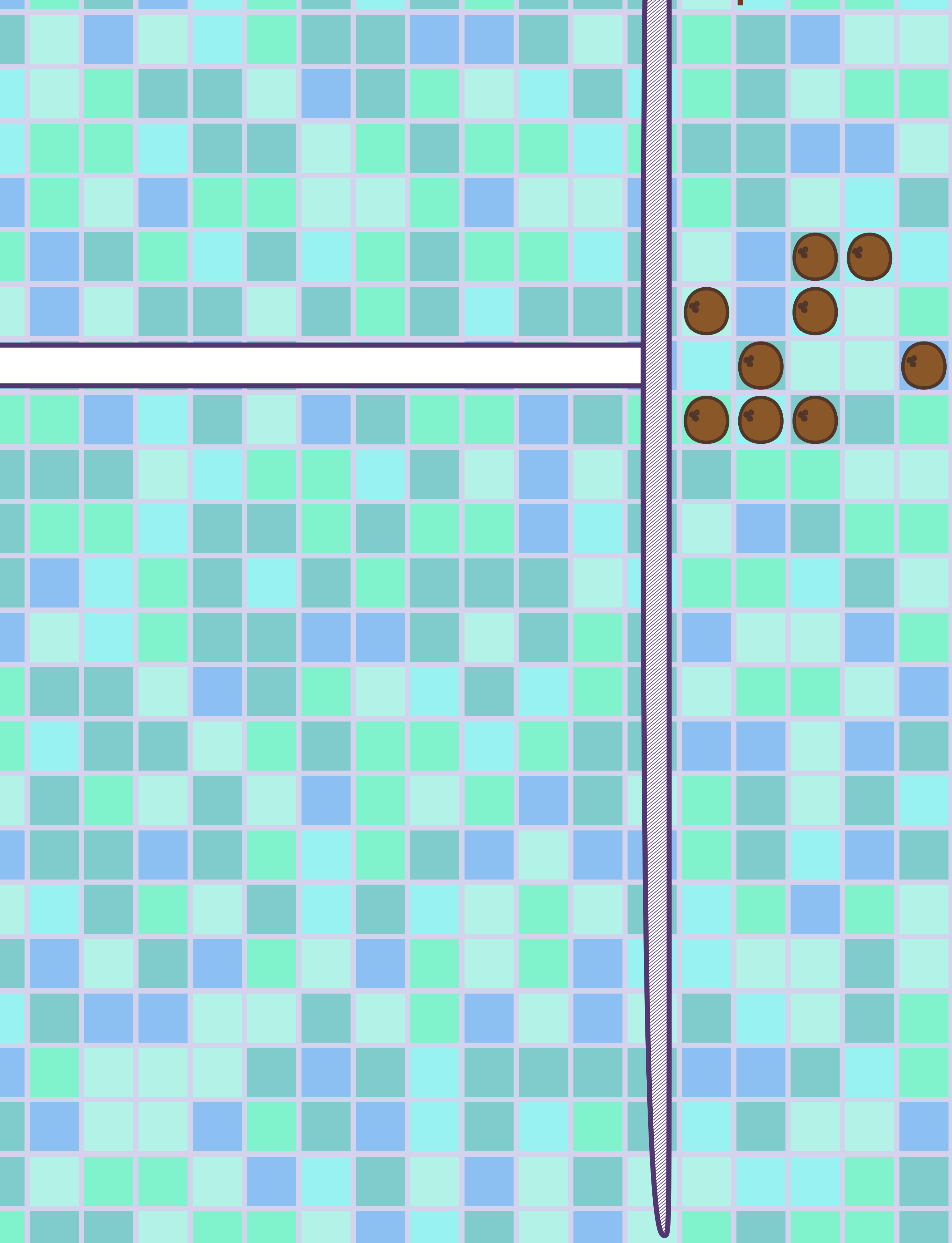
Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.



Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

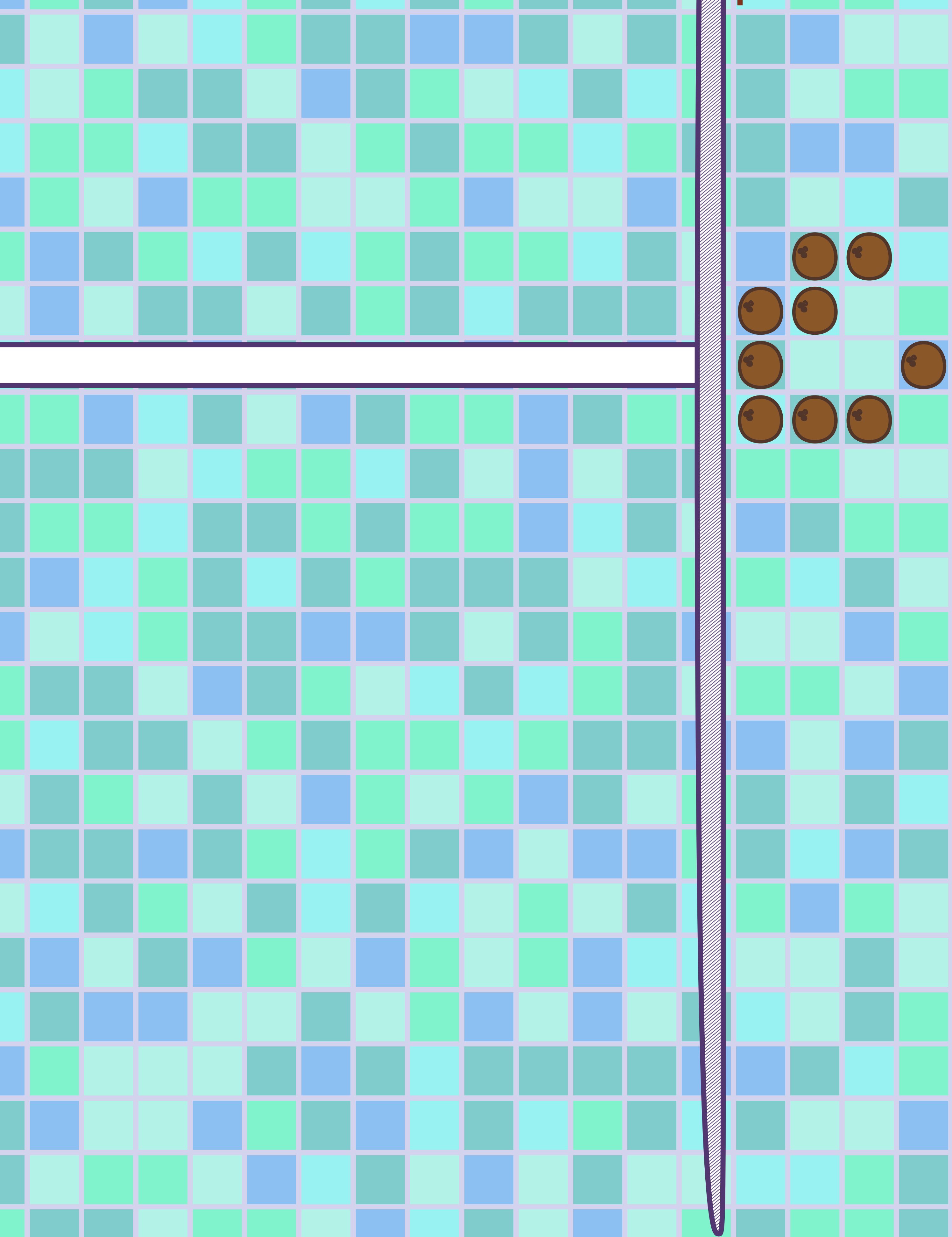
Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.



Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.



Let P be a set of n coconuts in a swimming pool.

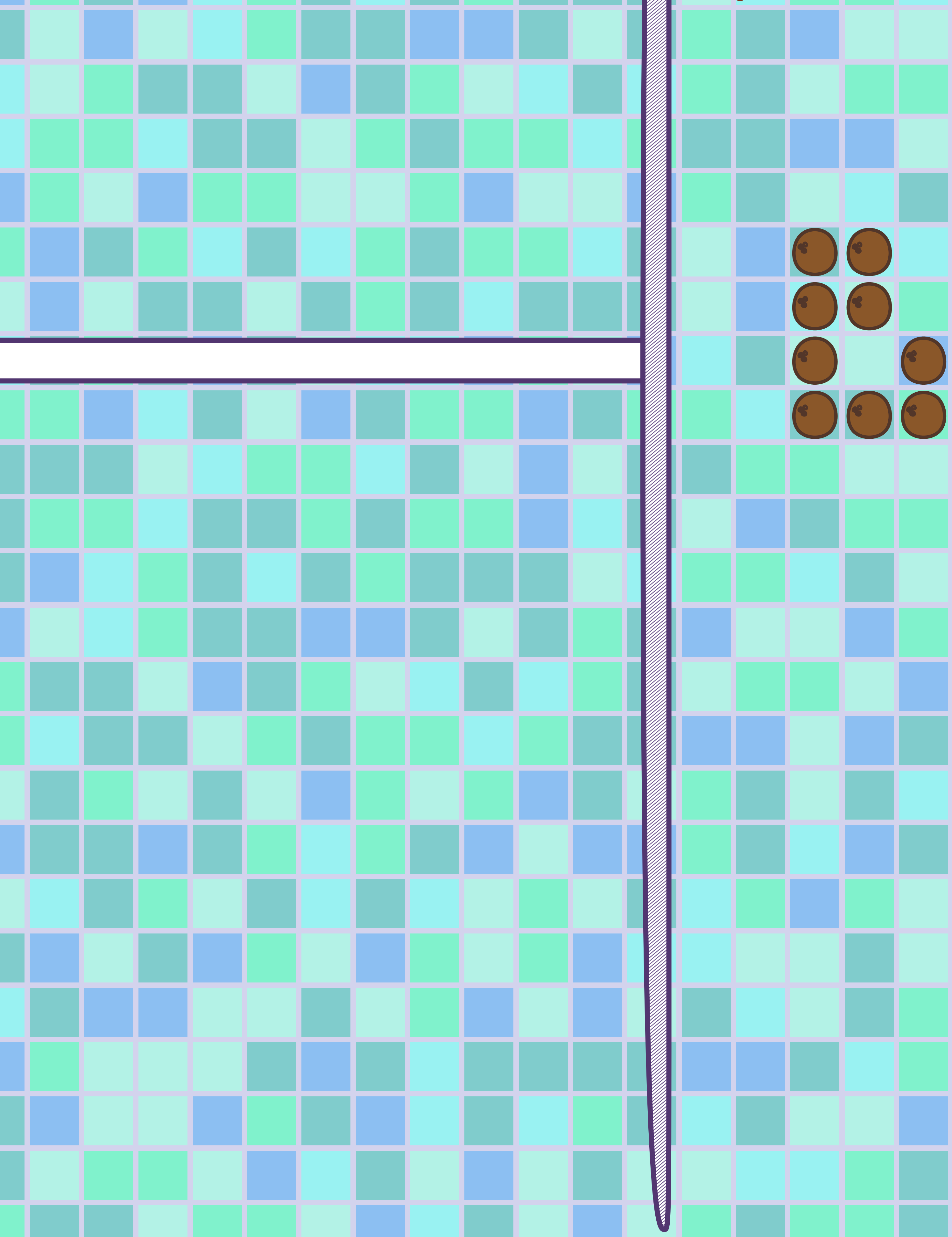
We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

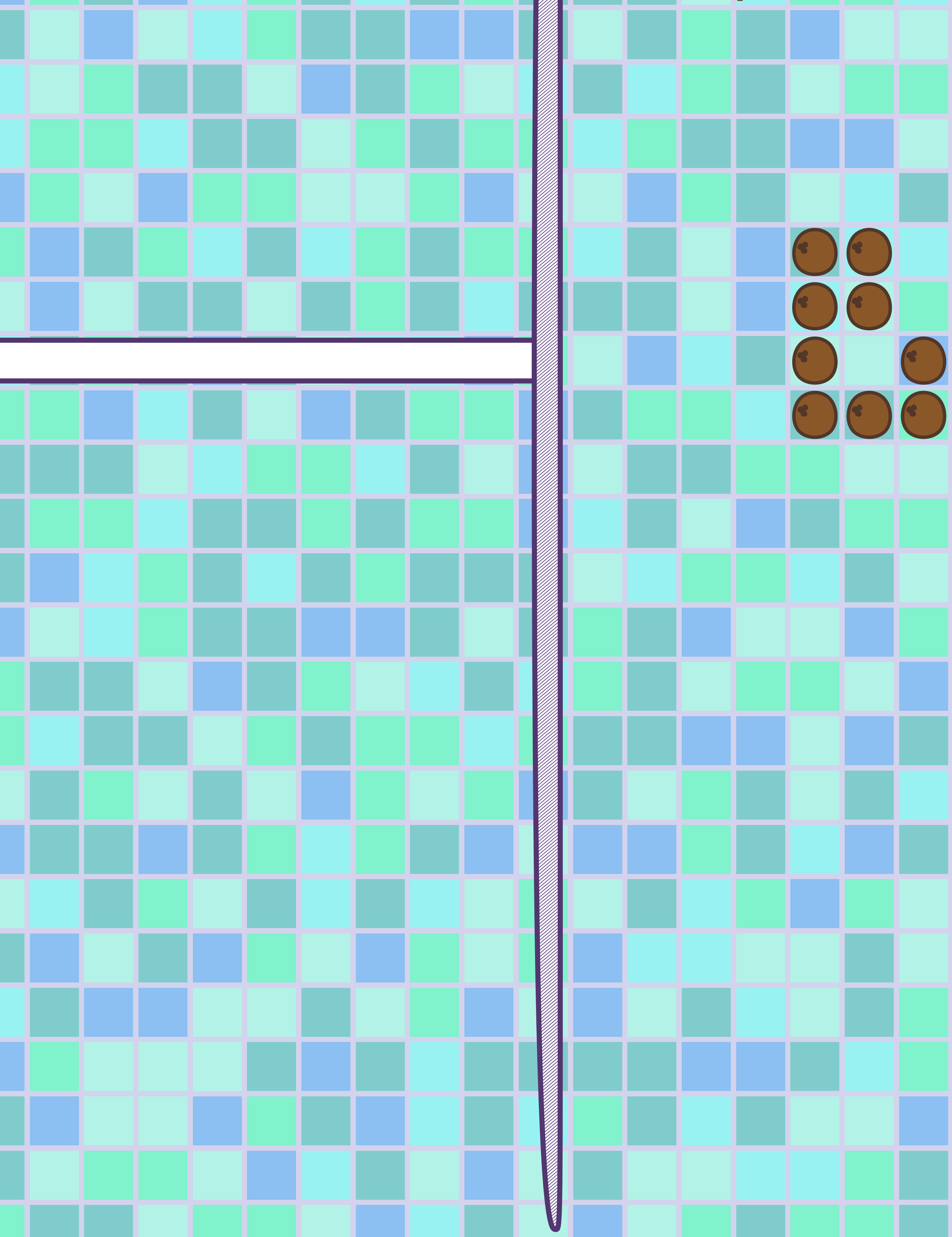
Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.



Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

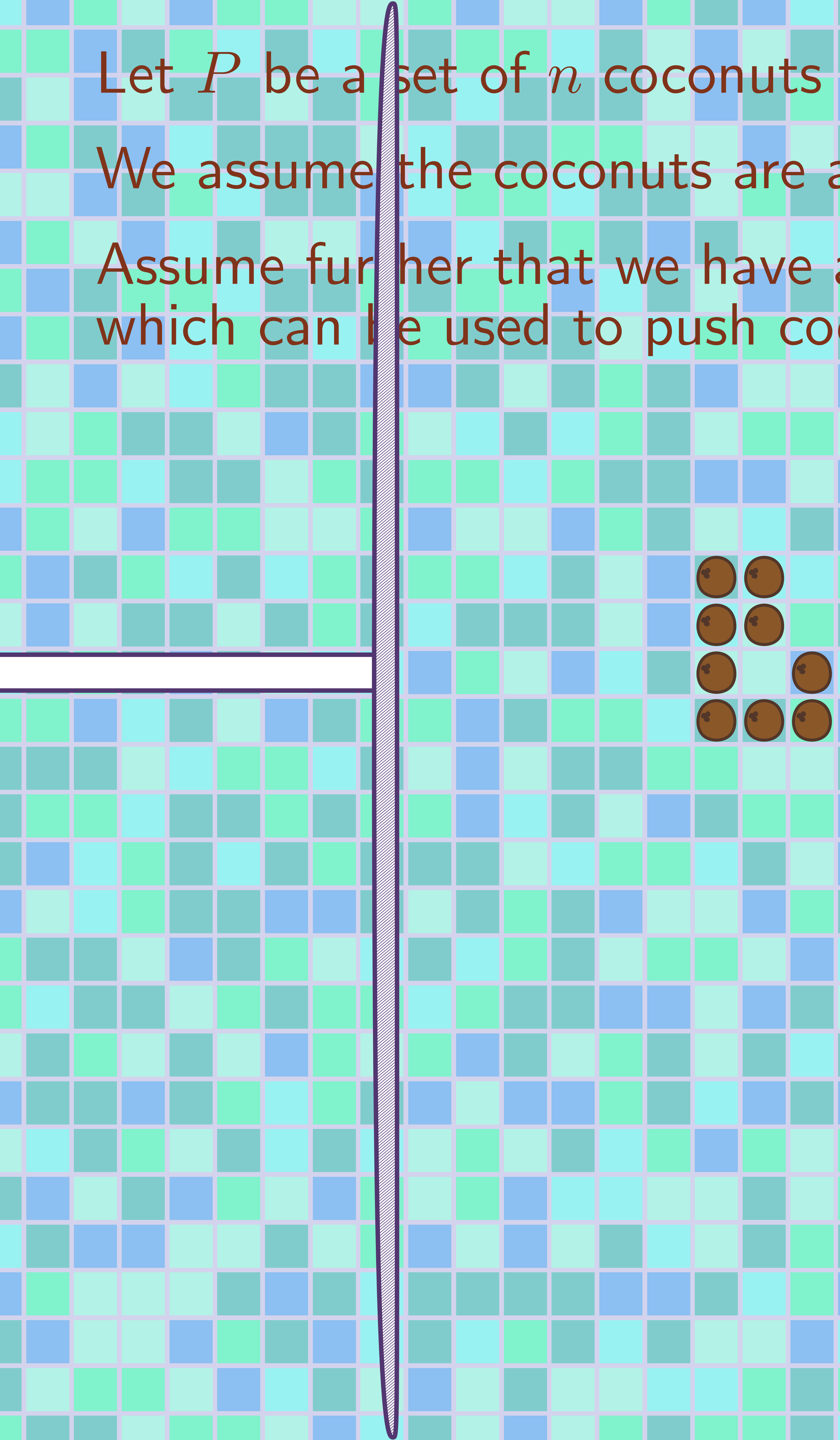
Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.



Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

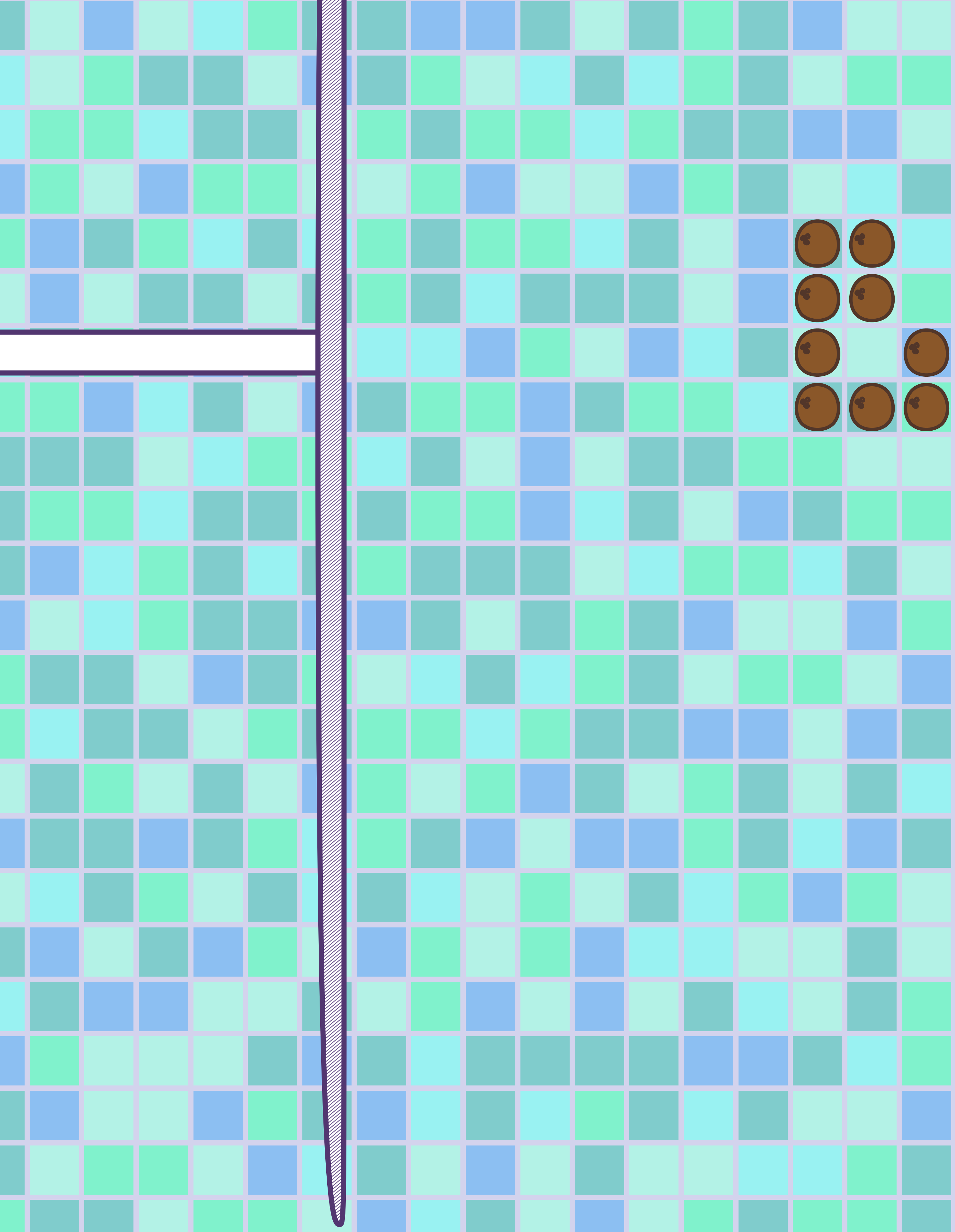
Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.



Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

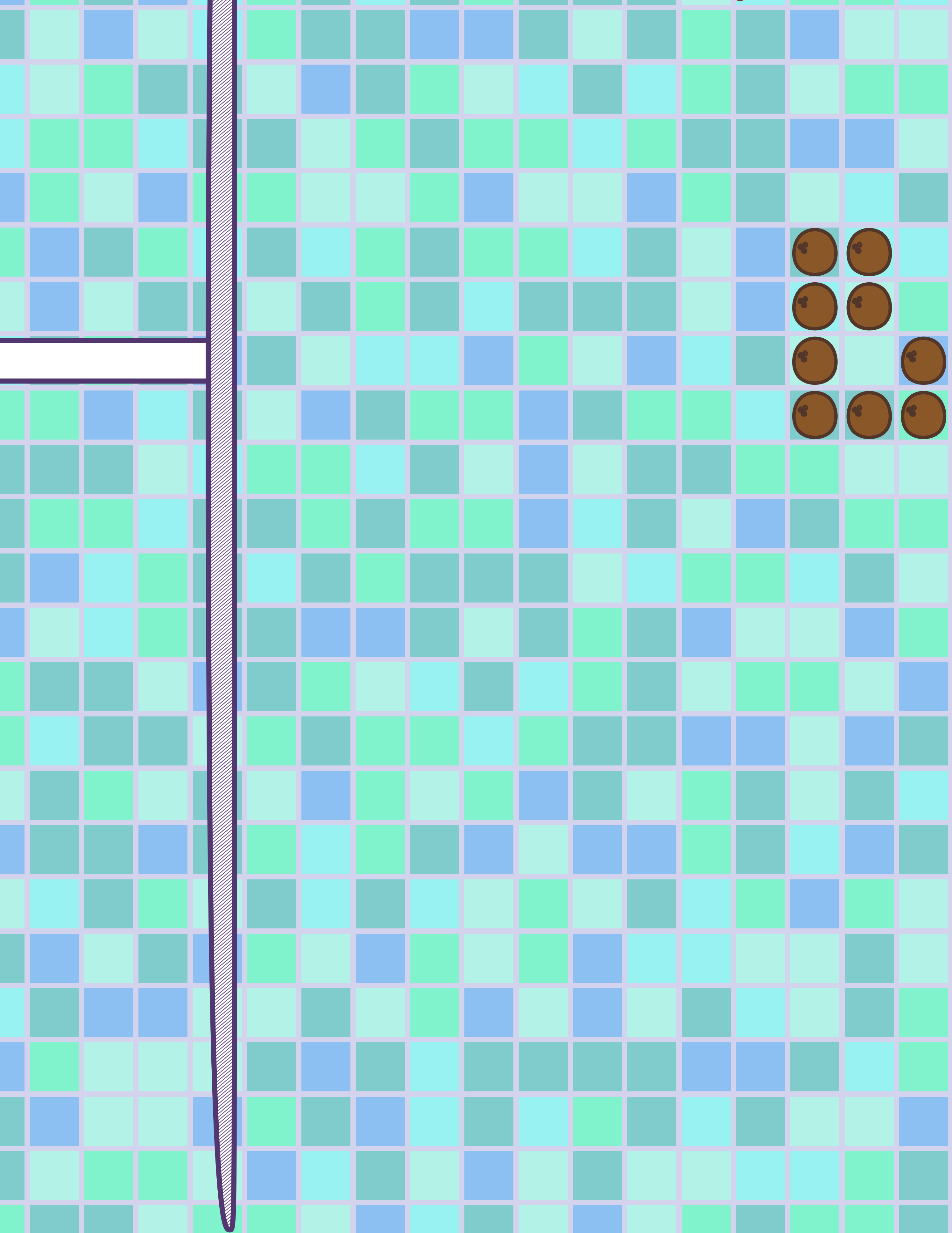
Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.



Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

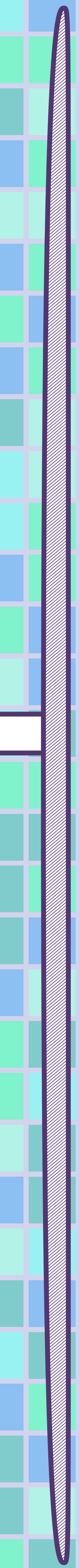
Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.



Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.



Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

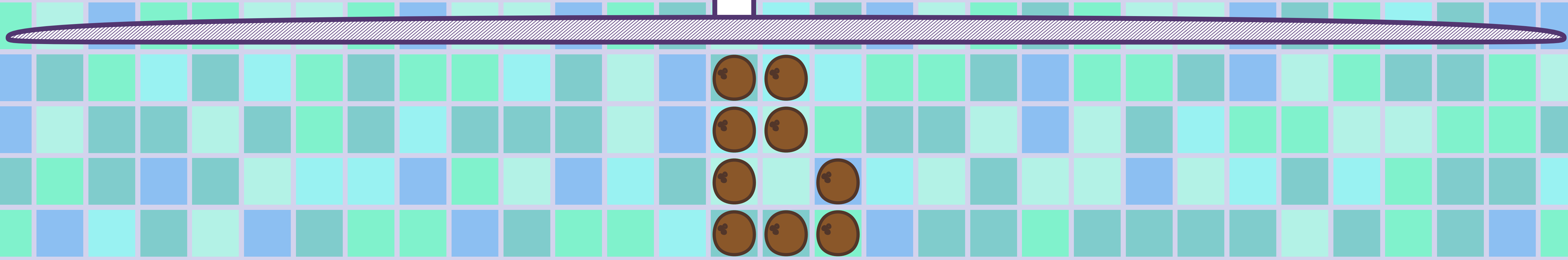
...or more accurately, four of them.

Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.



Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

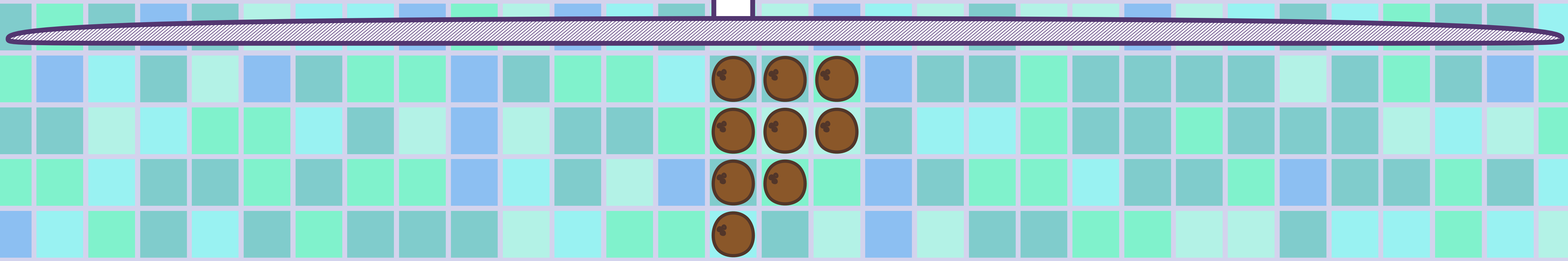
...or more accurately, four of them.

Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

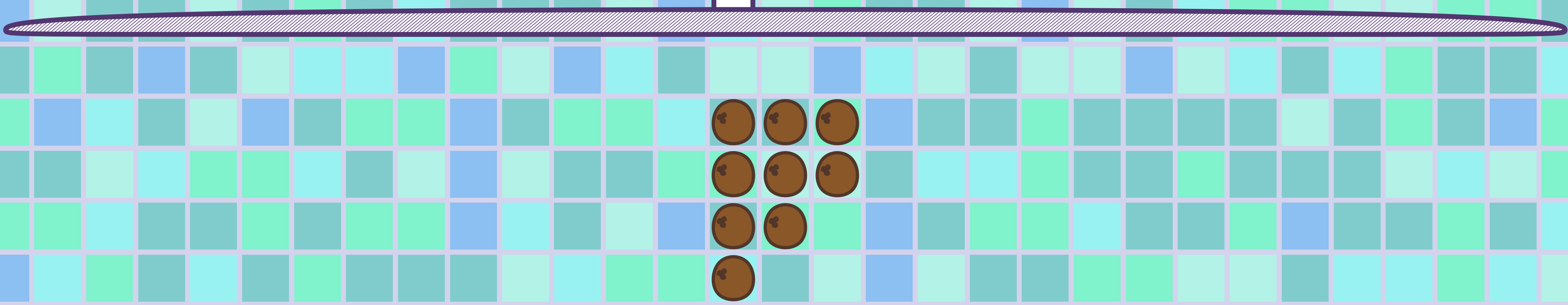


Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

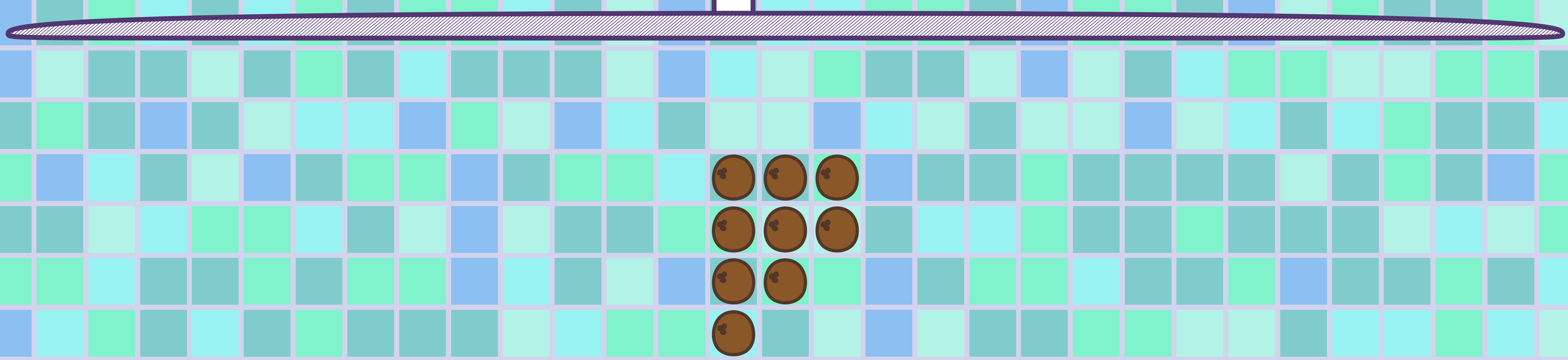


Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.



Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

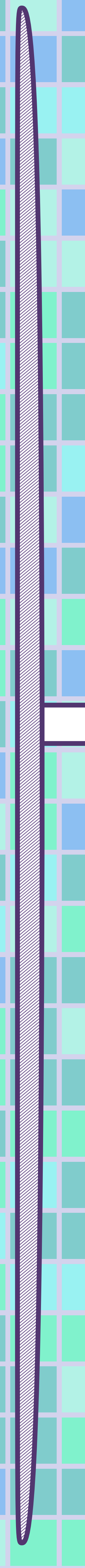
...or more accurately, four of them.

Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

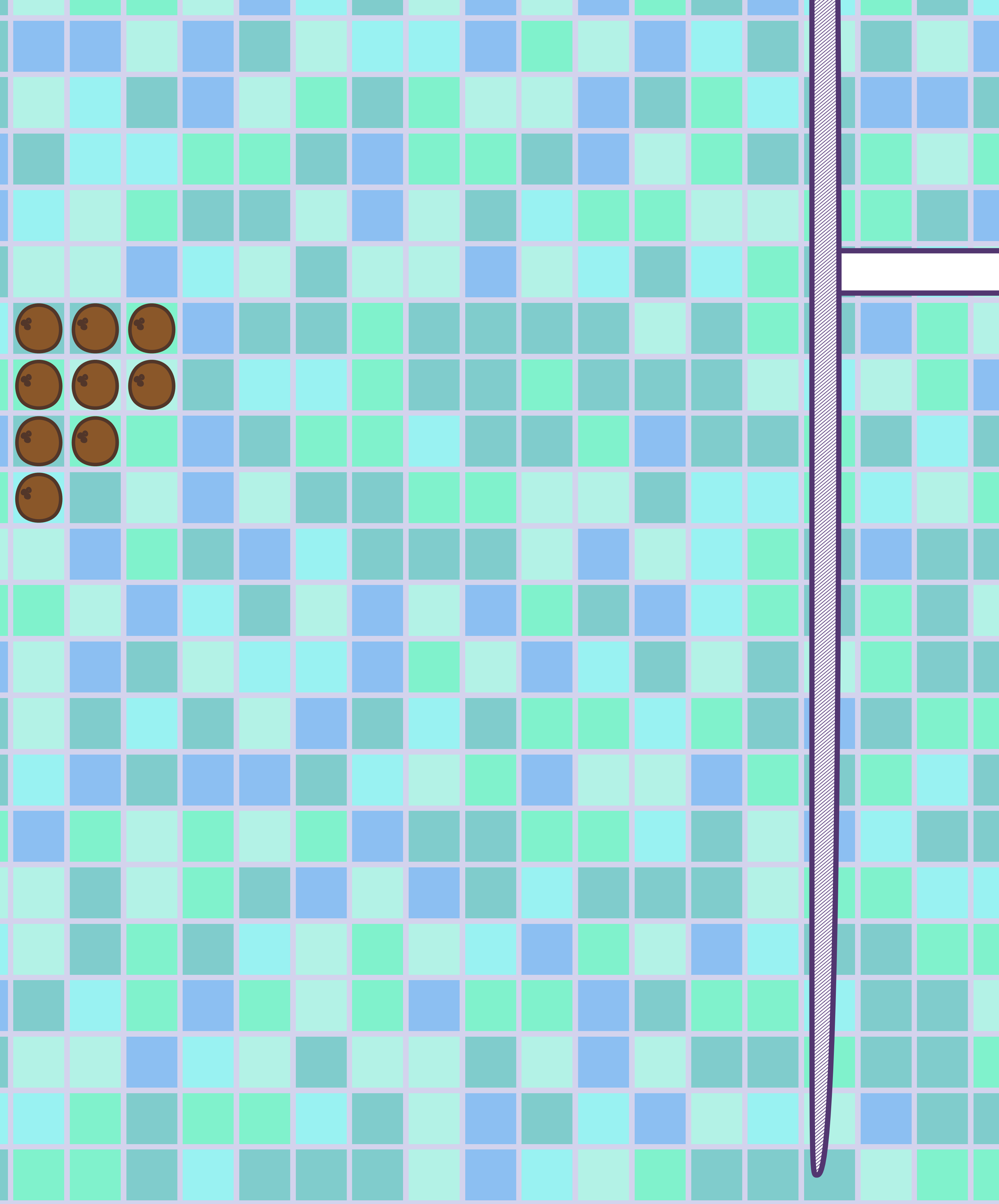


Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

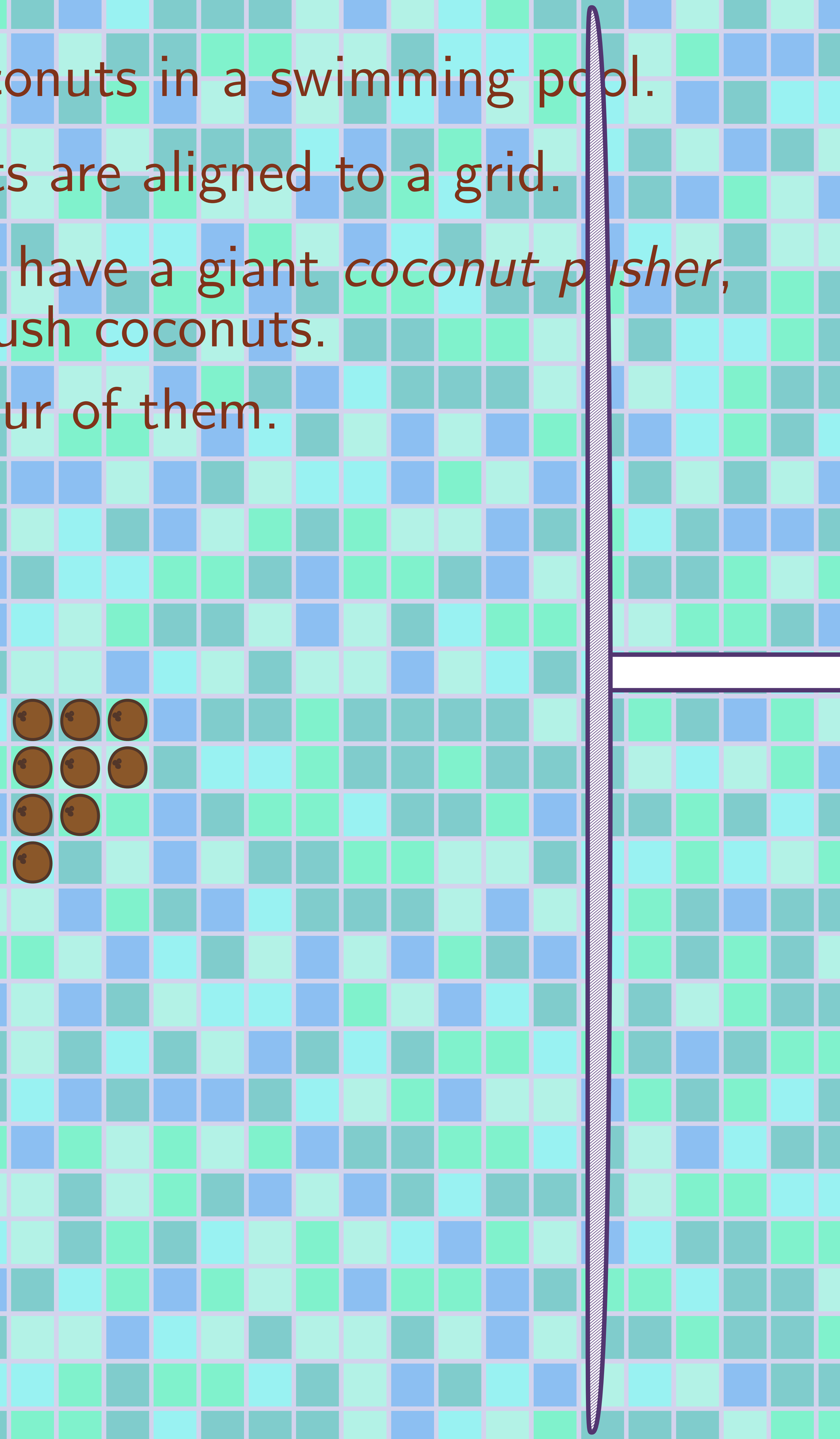


Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

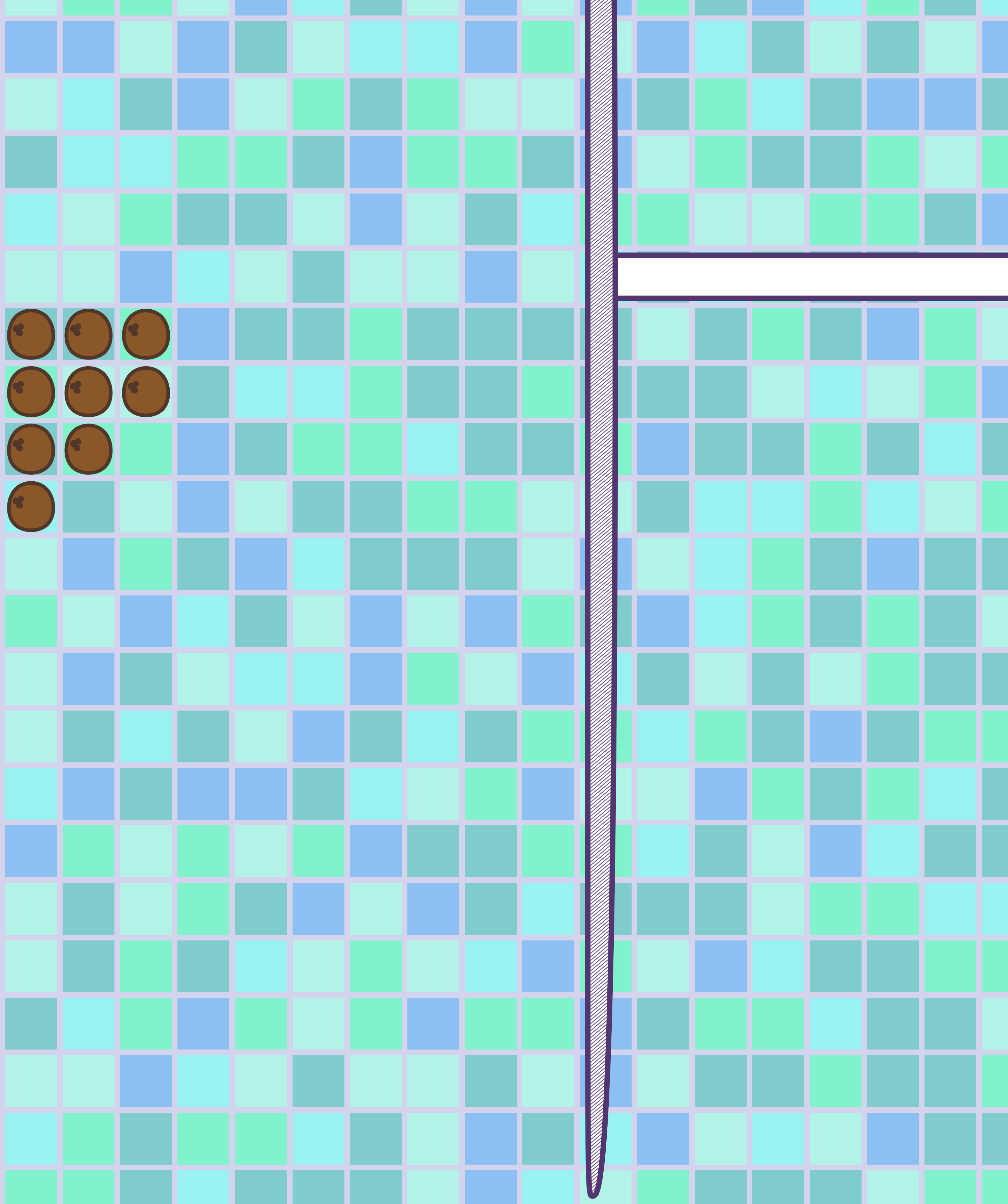


Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

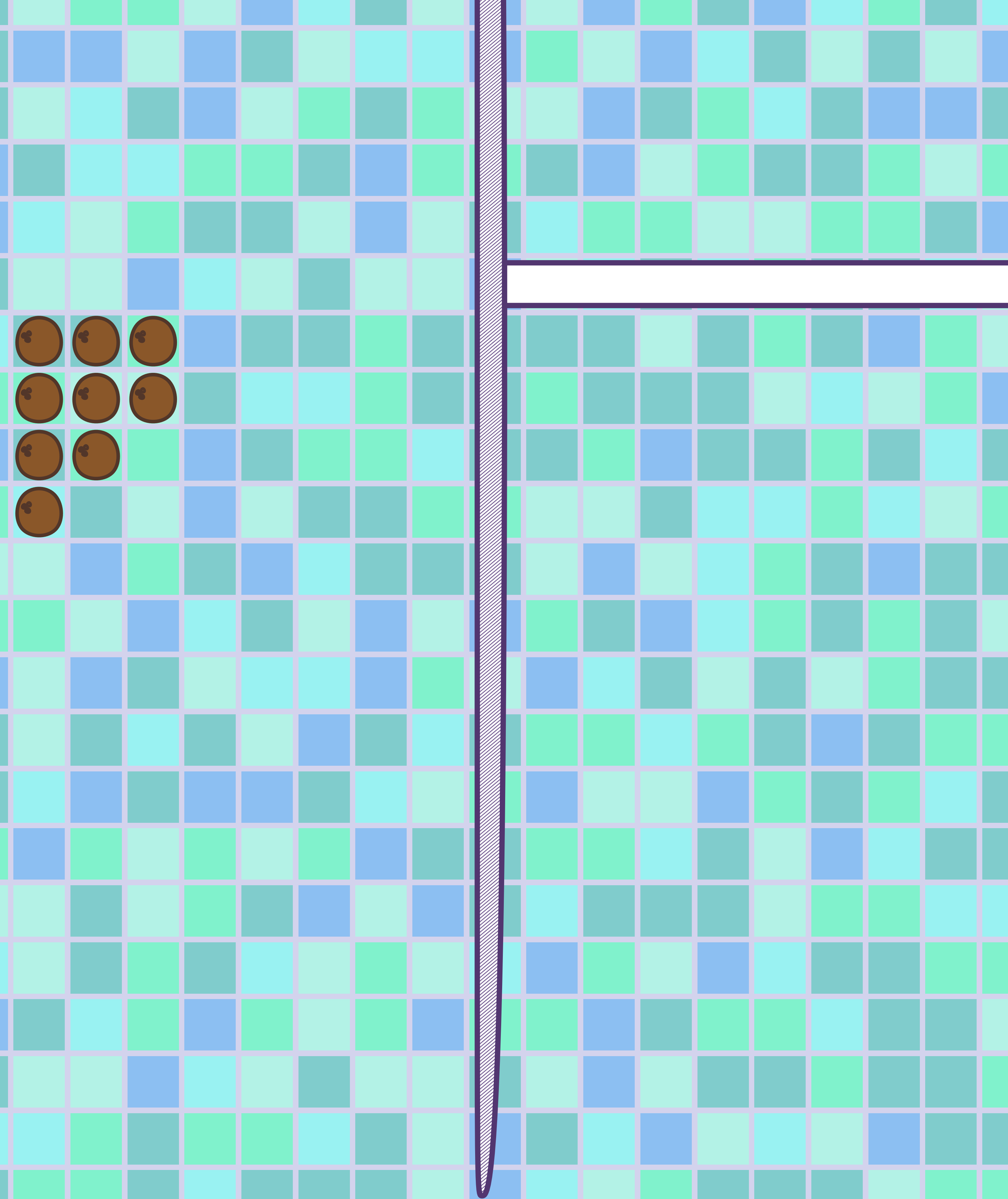


Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

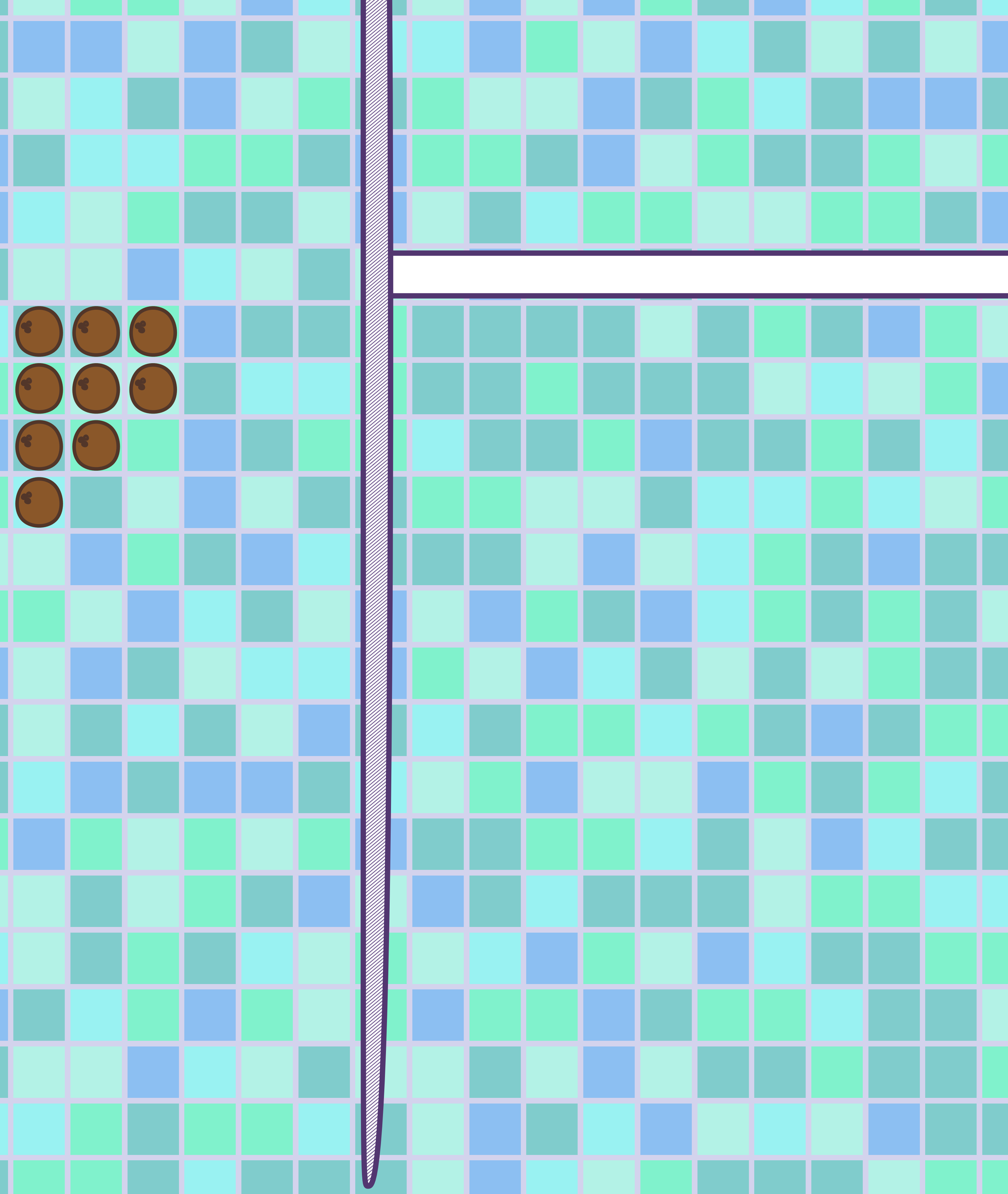


Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts

...or more accurately, four of them.

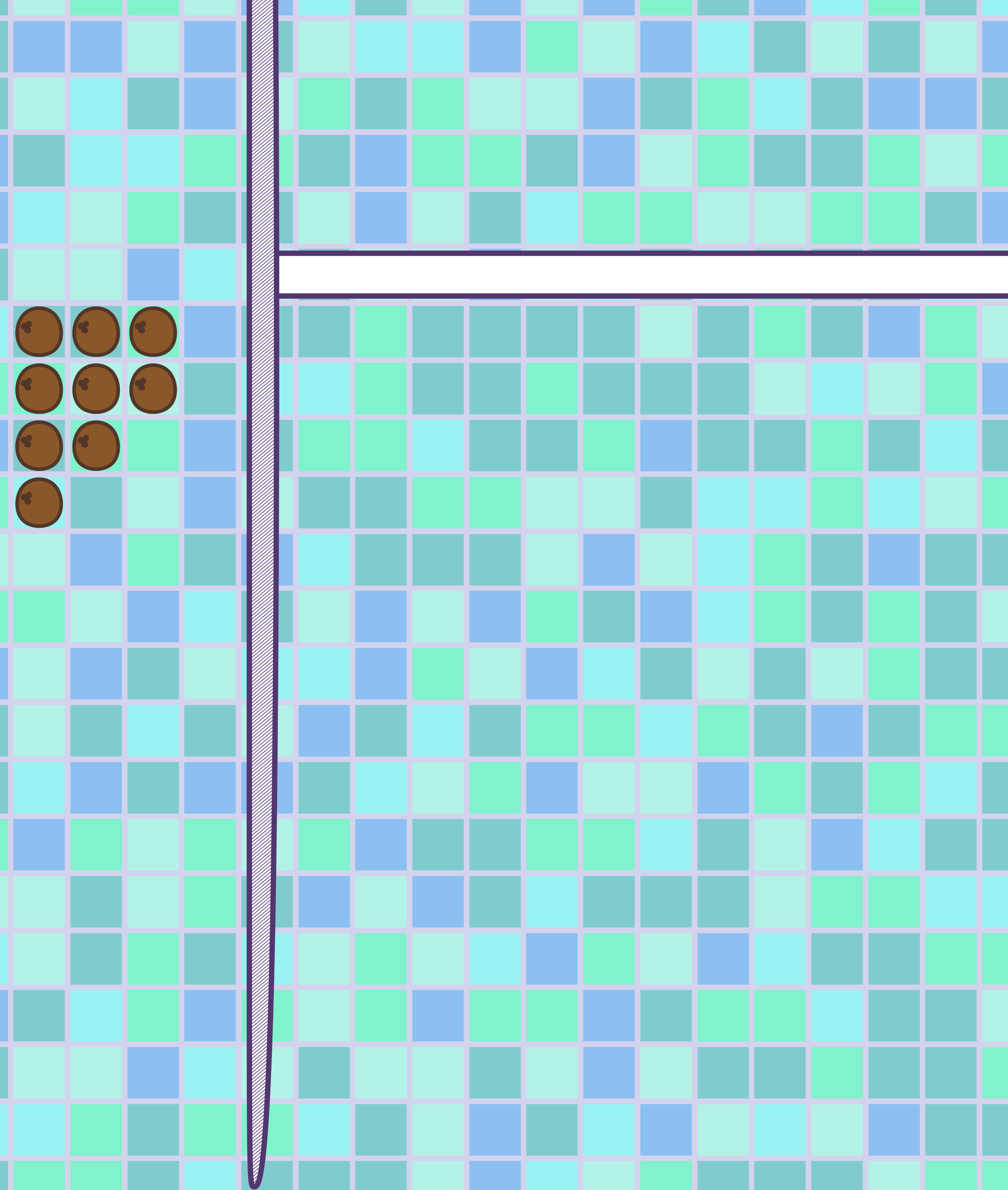


Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

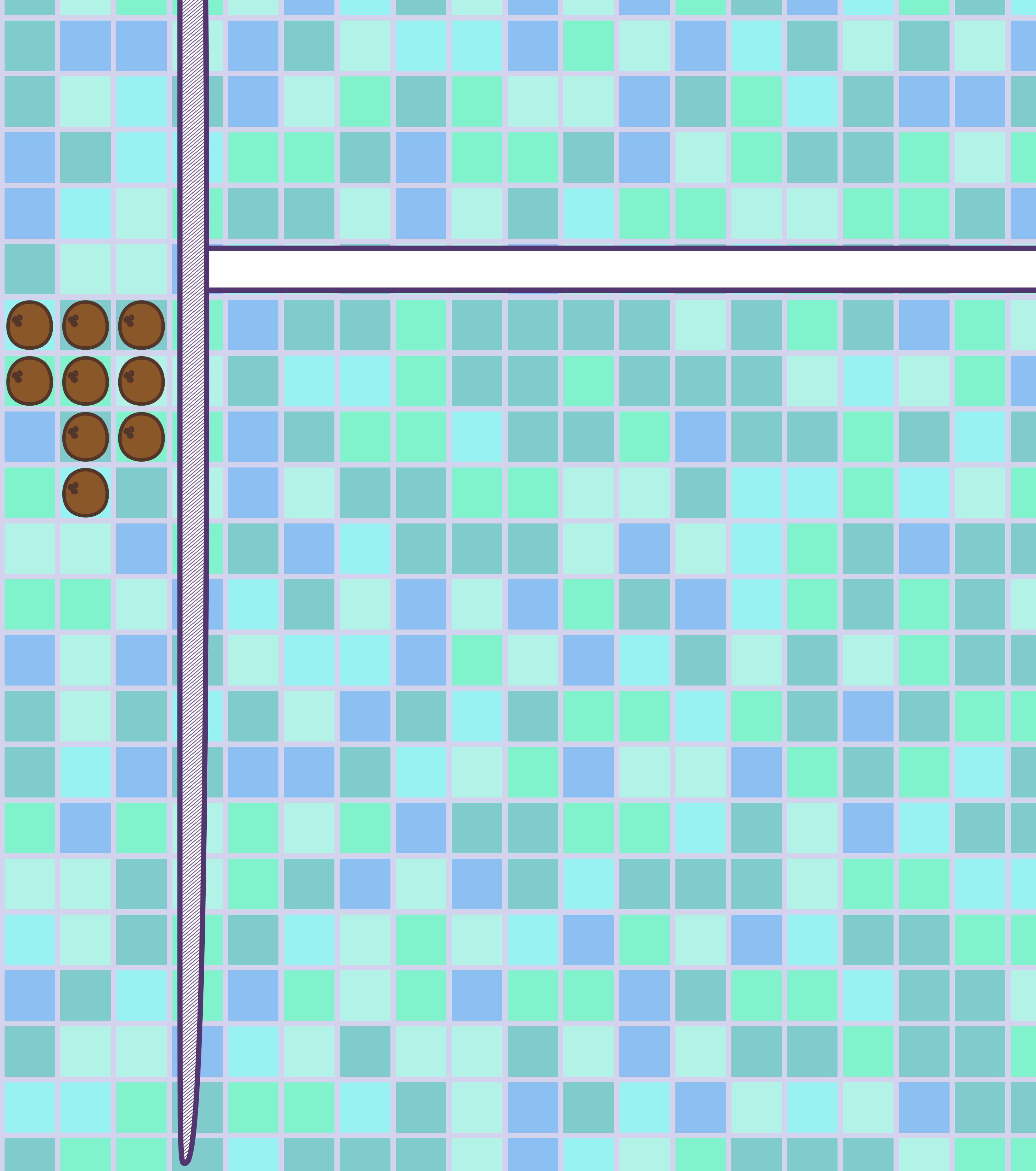


Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

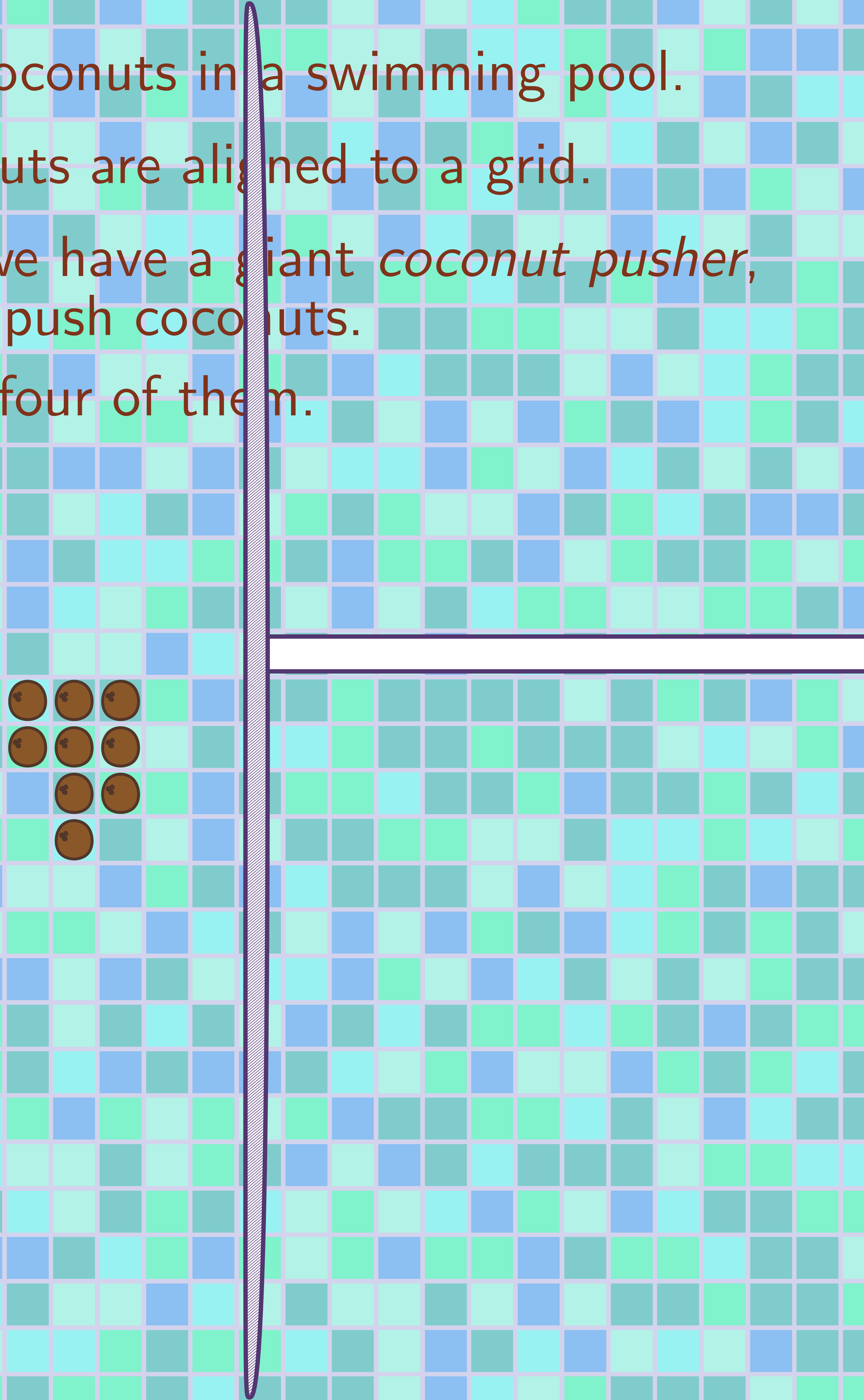


Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

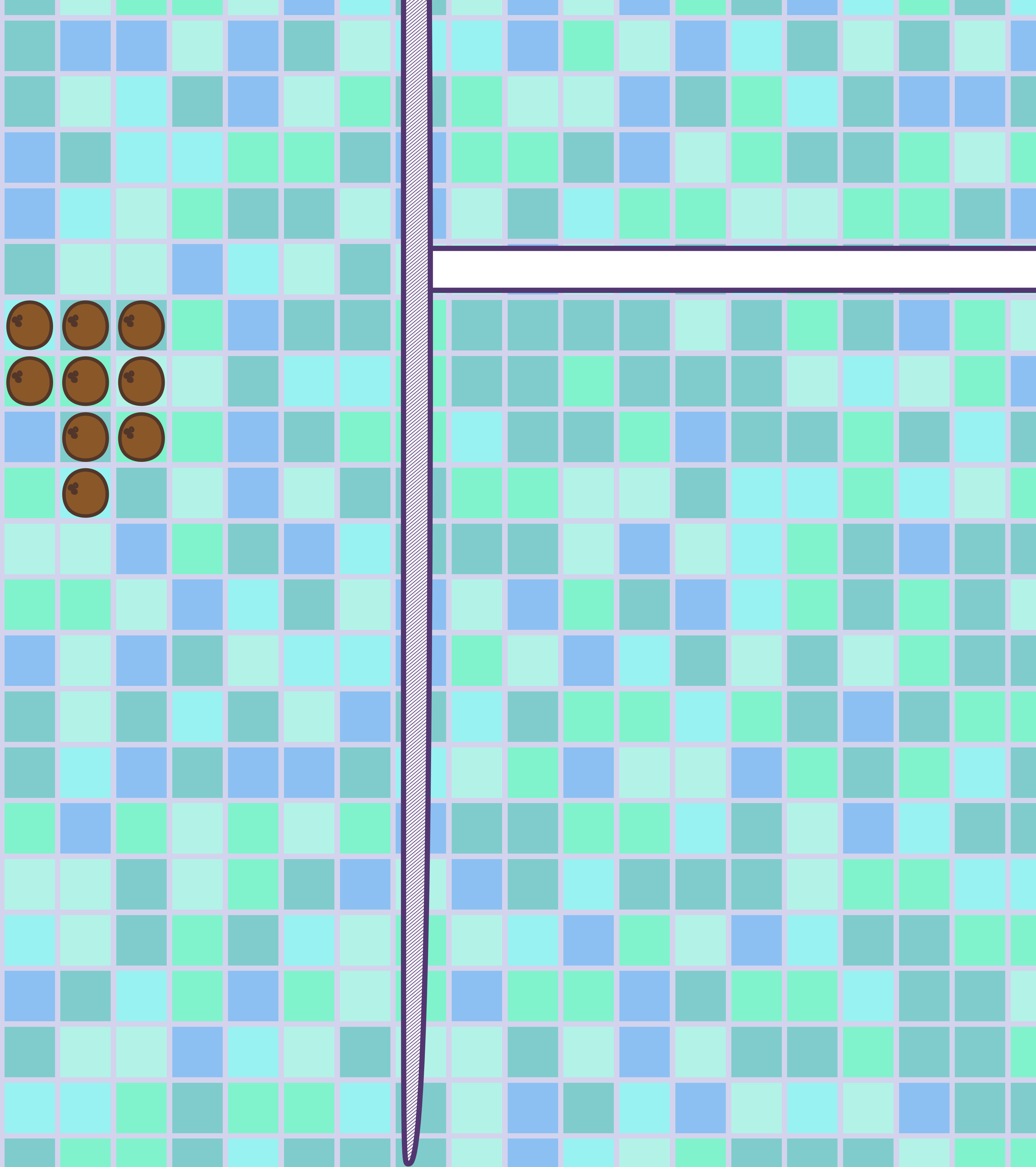


Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts

...or more accurately, four of them.

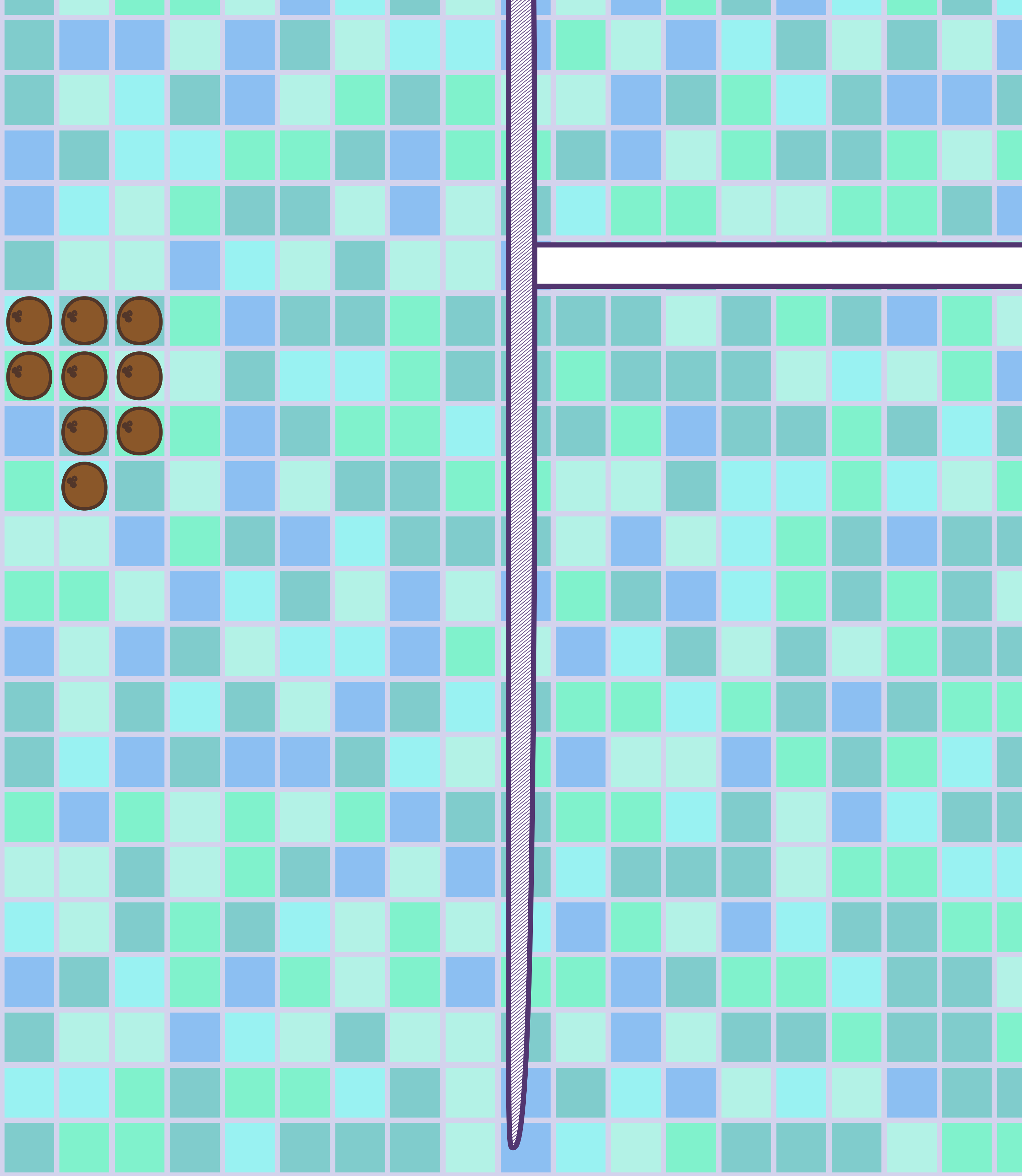


Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

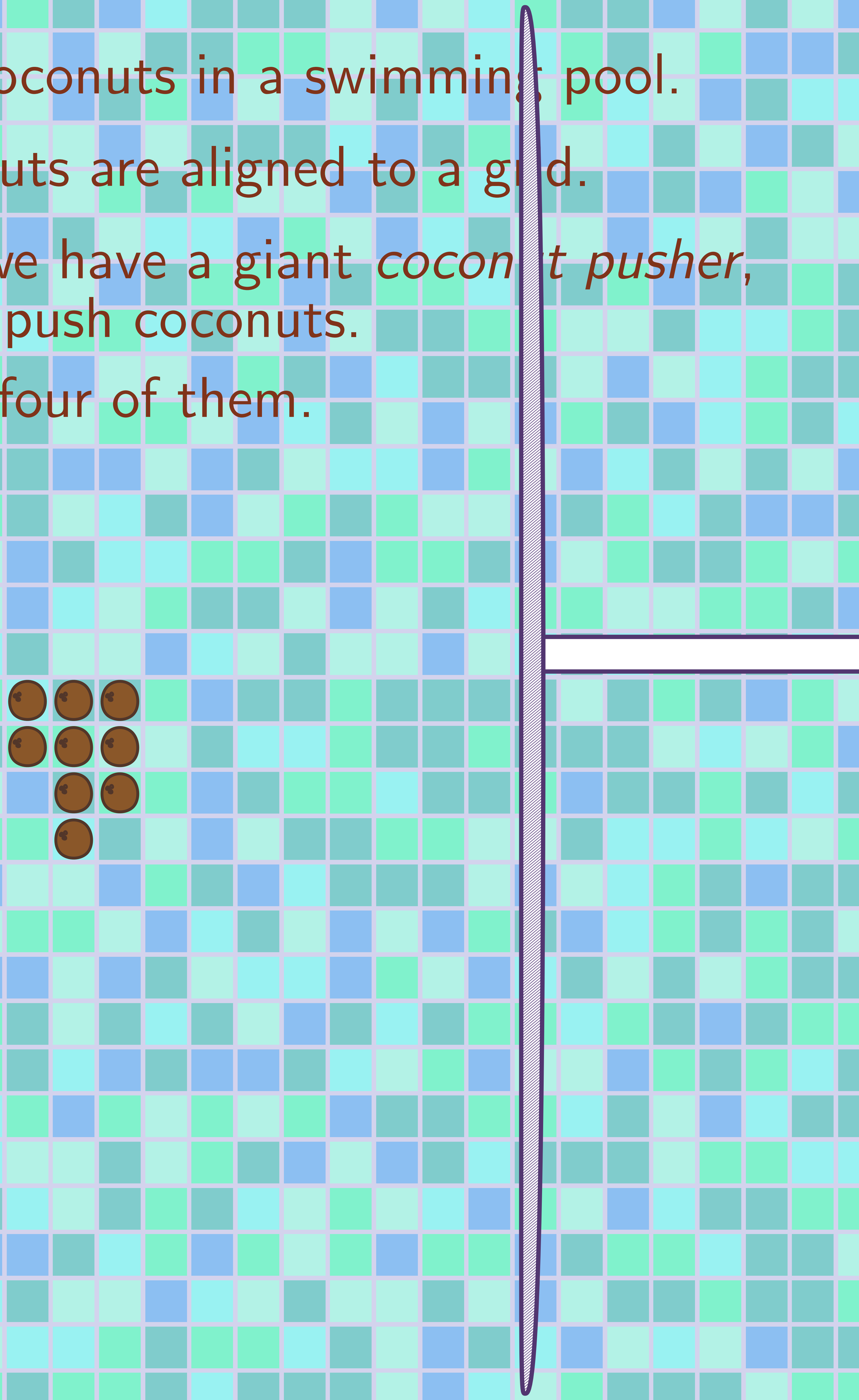


Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

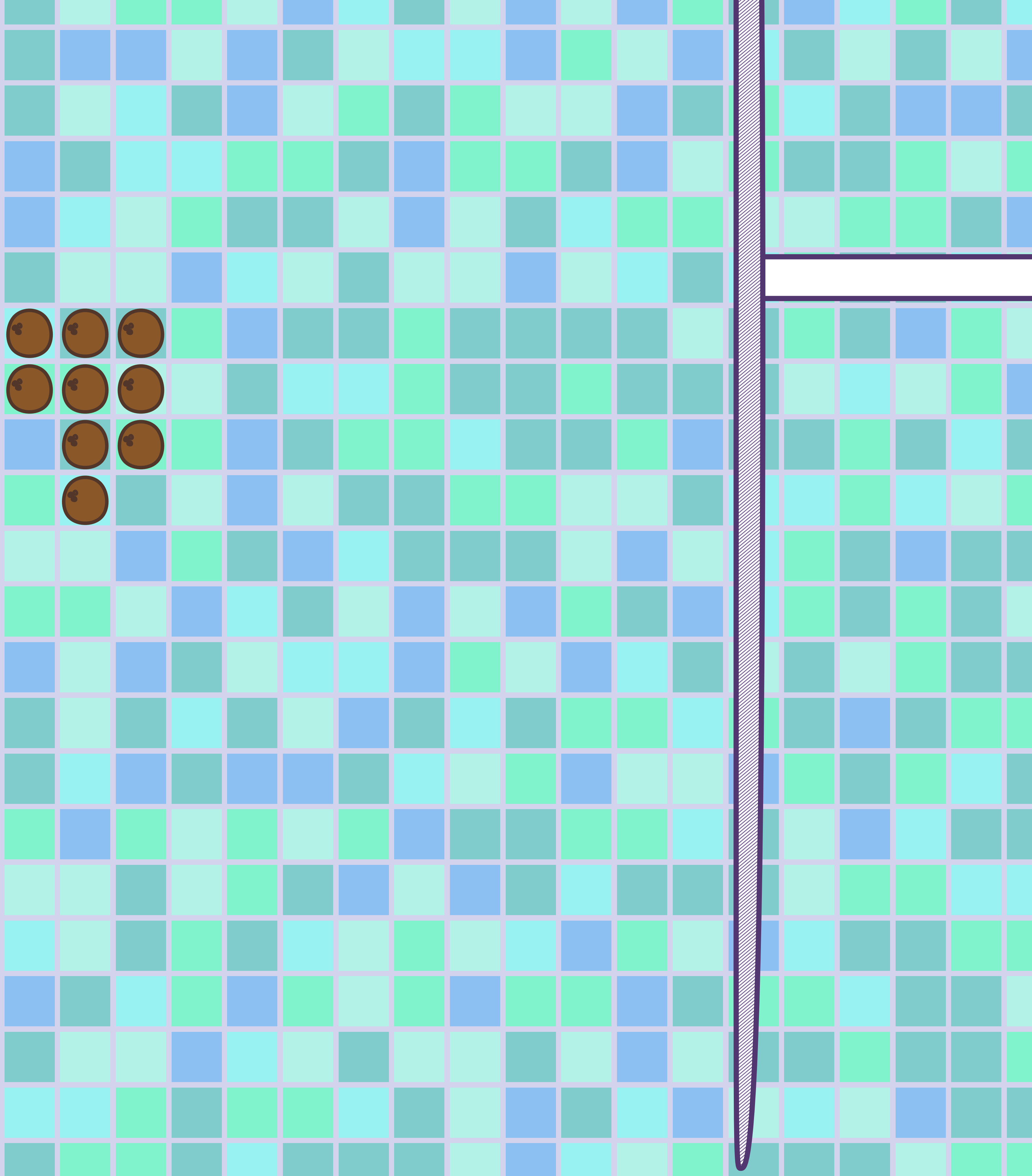


Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

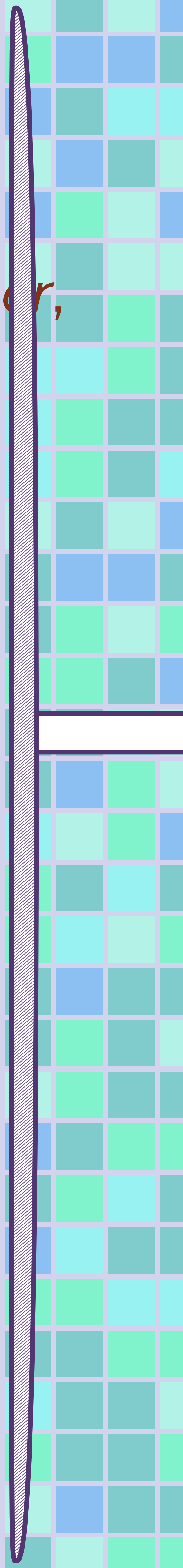


Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

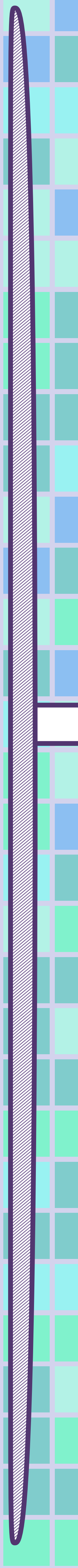


Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.



Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

Let P be a set of n coconuts in a swimming pool.

We assume the coconuts are aligned to a grid.

Assume further that we have a giant *coconut pusher*, which can be used to push coconuts.

...or more accurately, four of them.

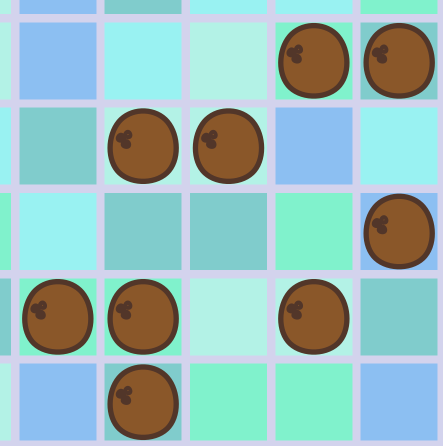
We wish to study the behaviour of sequences of such coconut pushes.

Question

Can we always push our coconuts into tidy rectangles?

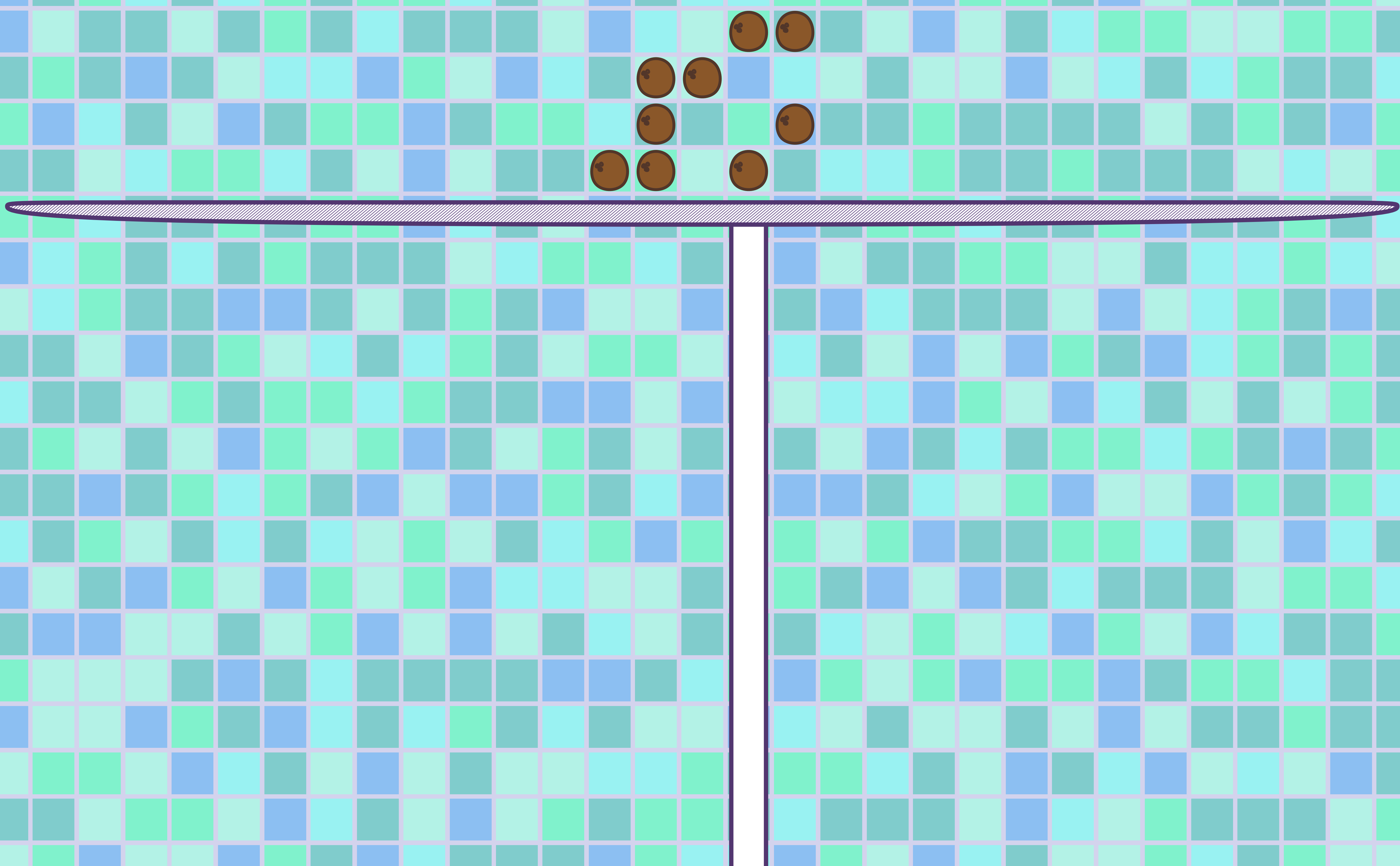
Question

Can we always push our coconuts into tidy rectangles?



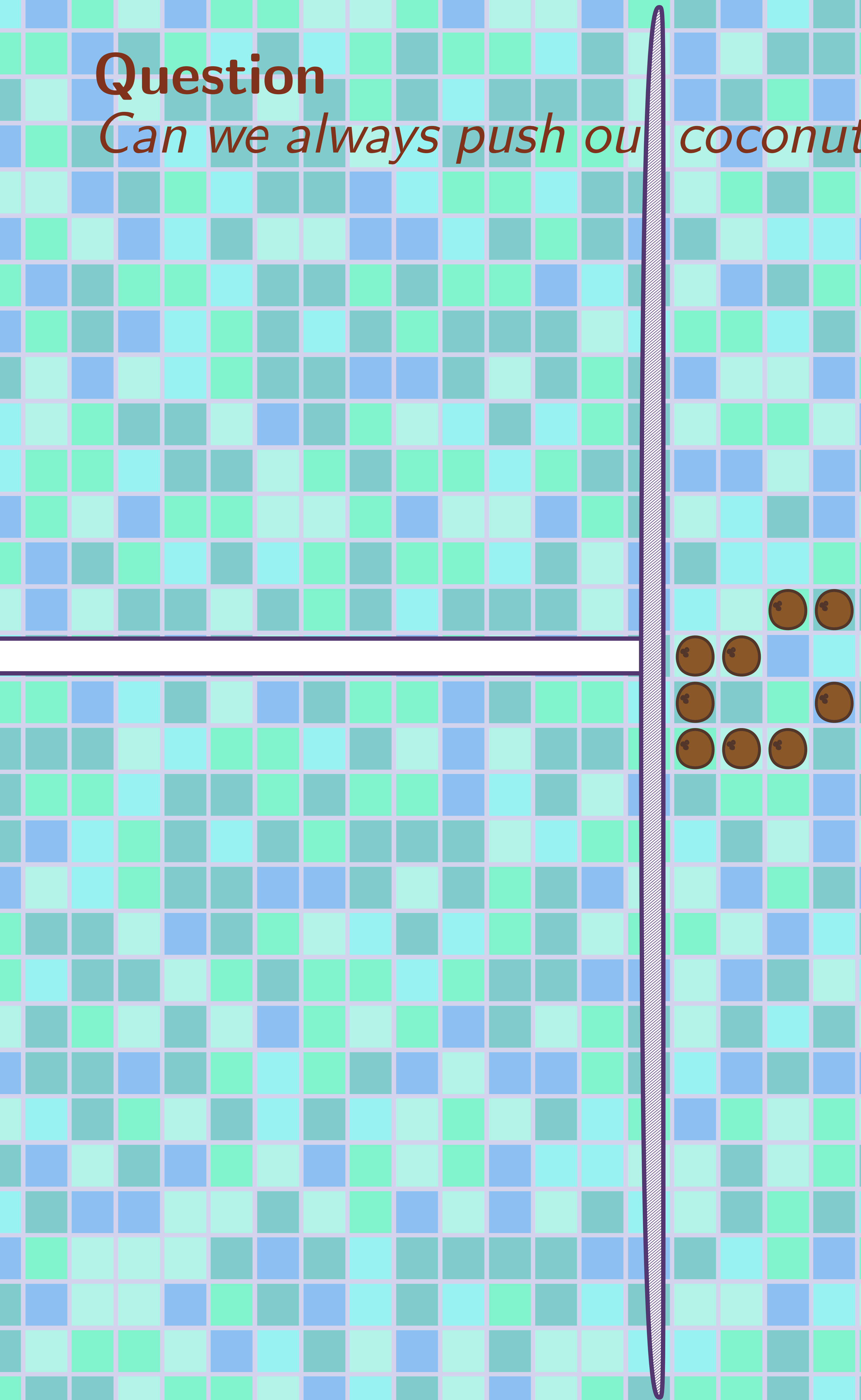
Question

Can we always push our coconuts into tidy rectangles?



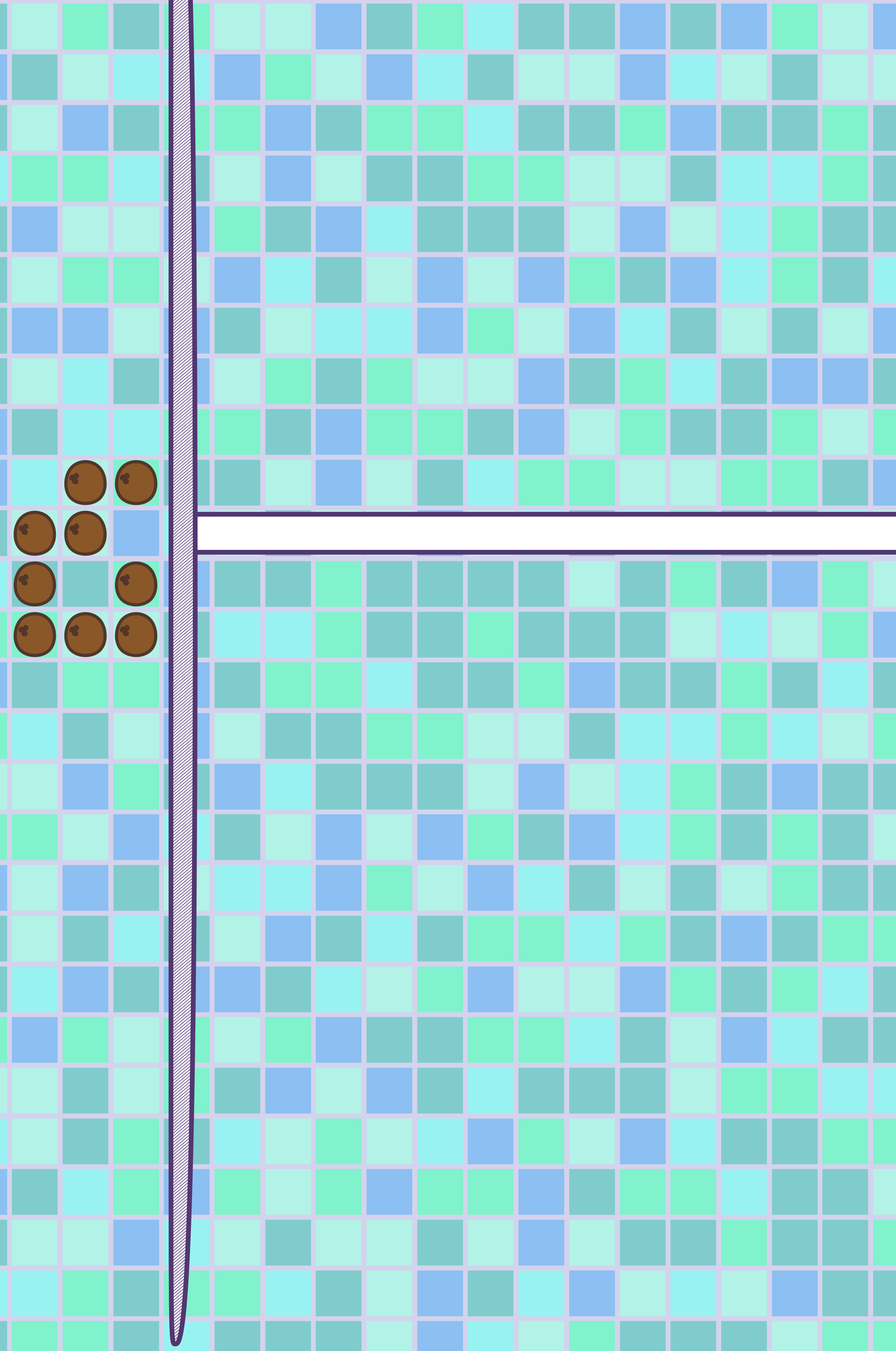
Question

Can we always push our coconuts into tidy rectangles?



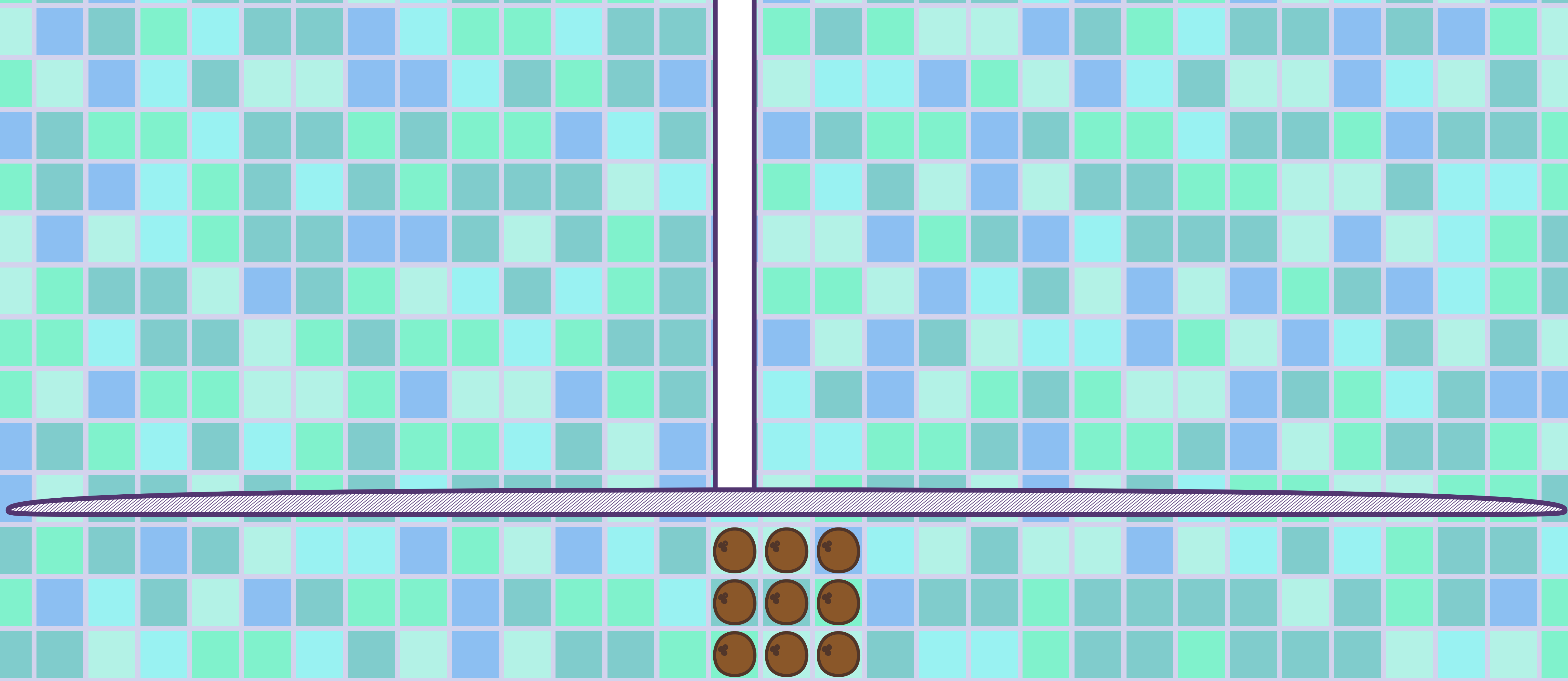
Question

Can we always push our coconuts into tidy rectangles?



Question

Can we always push our coconuts into tidy rectangles?



Question

Can we always push our coconuts into tidy rectangles?

Question

Can we always push our coconuts into tidy rectangles?

Obviously, n coconuts can only be pushed into a $a \times b$ rectangle if $ab = n$!

Question

Can we always push our coconuts into tidy rectangles?

Obviously, n coconuts can only be pushed into a $a \times b$ rectangle if $ab = n$!

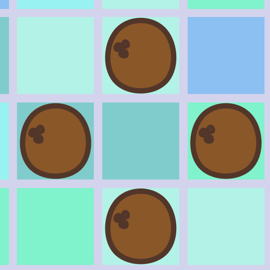
But is it always possible?

Question

Can we always push our coconuts into tidy rectangles?

Obviously, n coconuts can only be pushed into a $a \times b$ rectangle if $ab = n$!

But is it always possible?



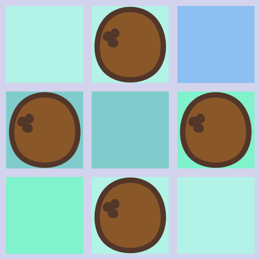
Question

Can we always push our coconuts into tidy rectangles?

Obviously, n coconuts can only be pushed into a $a \times b$ rectangle if $ab = n$!

But is it always possible?

No!



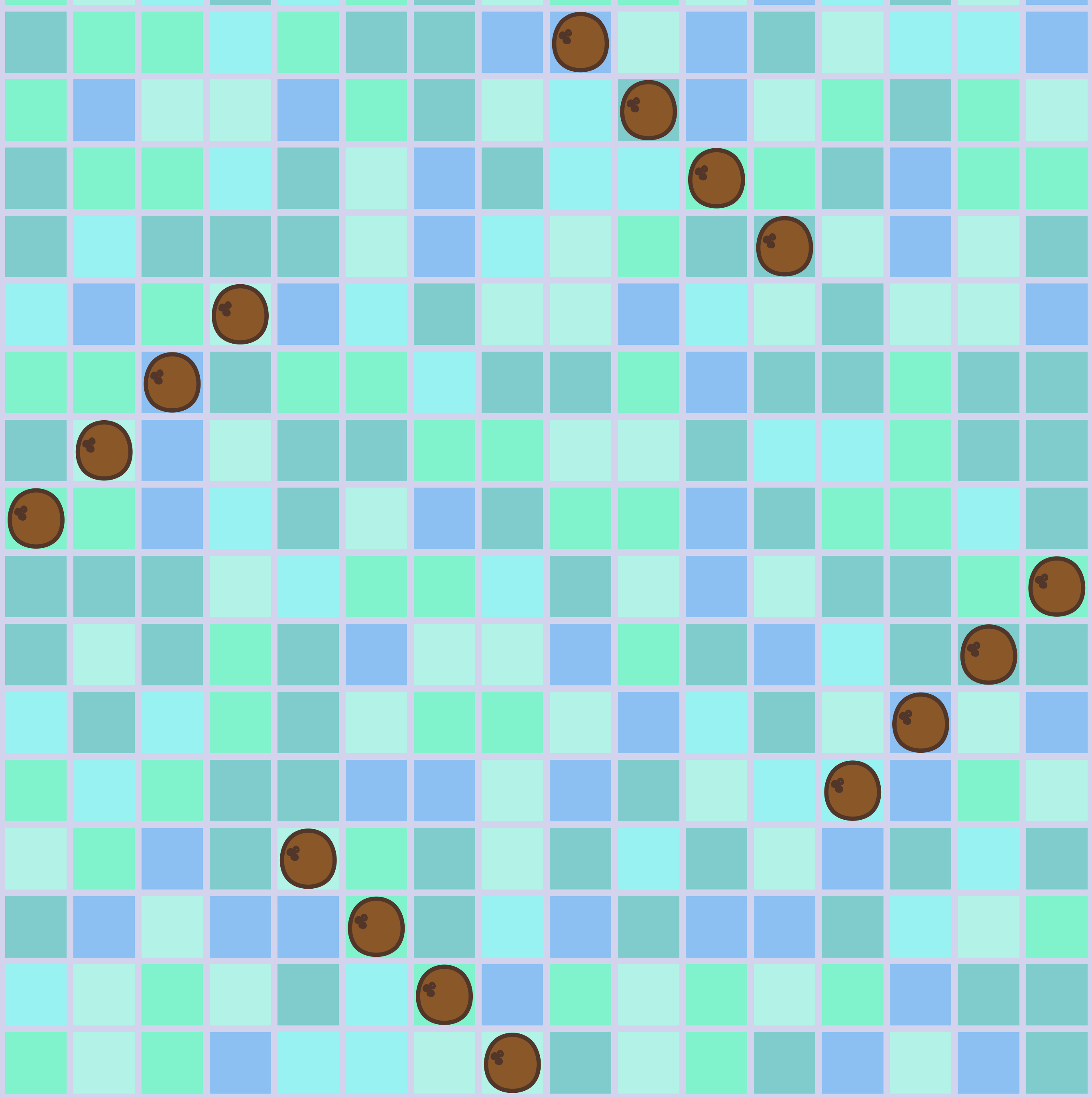
Question

Can we always push our coconuts into tidy rectangles?

Obviously, n coconuts can only be pushed into a $a \times b$ rectangle if $ab = n$!

But is it always possible?

No!



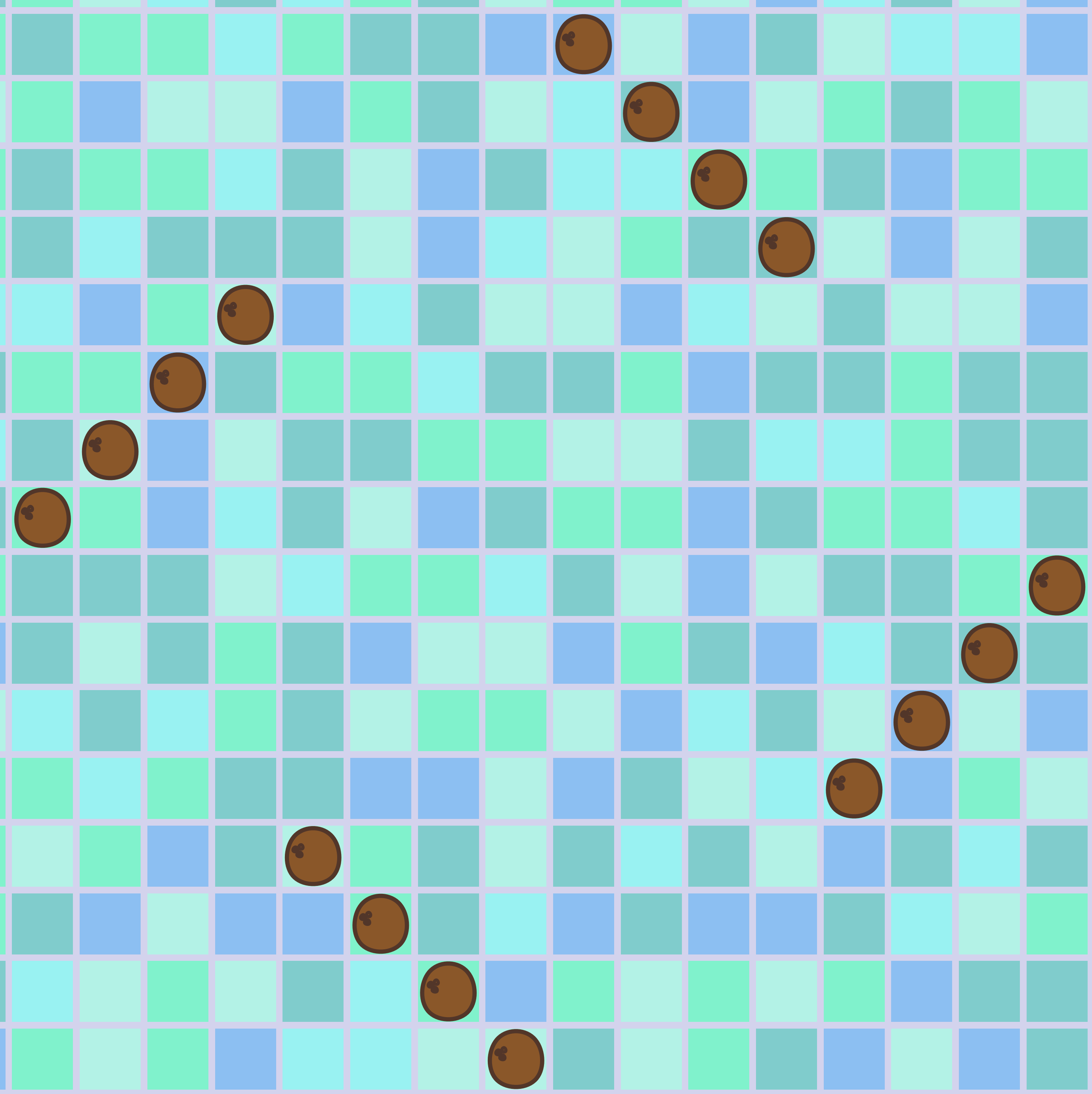
Question

Can we always push our coconuts into tidy rectangles?

Obviously, n coconuts can only be pushed into a $a \times b$ rectangle if $ab = n$!

But is it always possible?

No!



...and *some* aspect ratios are not even possible if we start with at most one coconut per row and column.

Results

Results

Deciding whether n coconuts can be pushed into an $a \times b$ rectangle is NP-complete.

Results

Deciding whether n coconuts can be pushed into an $a \times b$ rectangle is NP-complete.

Deciding whether n coconuts which occupy at most k rows can be pushed into a $2 \times n/2$ rectangle is polynomial.

Results

Deciding whether n coconuts can be pushed into an $a \times b$ rectangle is NP-complete.

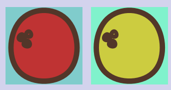
Deciding whether n coconuts which occupy at most k rows can be pushed into a $2 \times n/2$ rectangle is polynomial.

Everything inbetween is still open!

Observation
Coconuts have no identity.

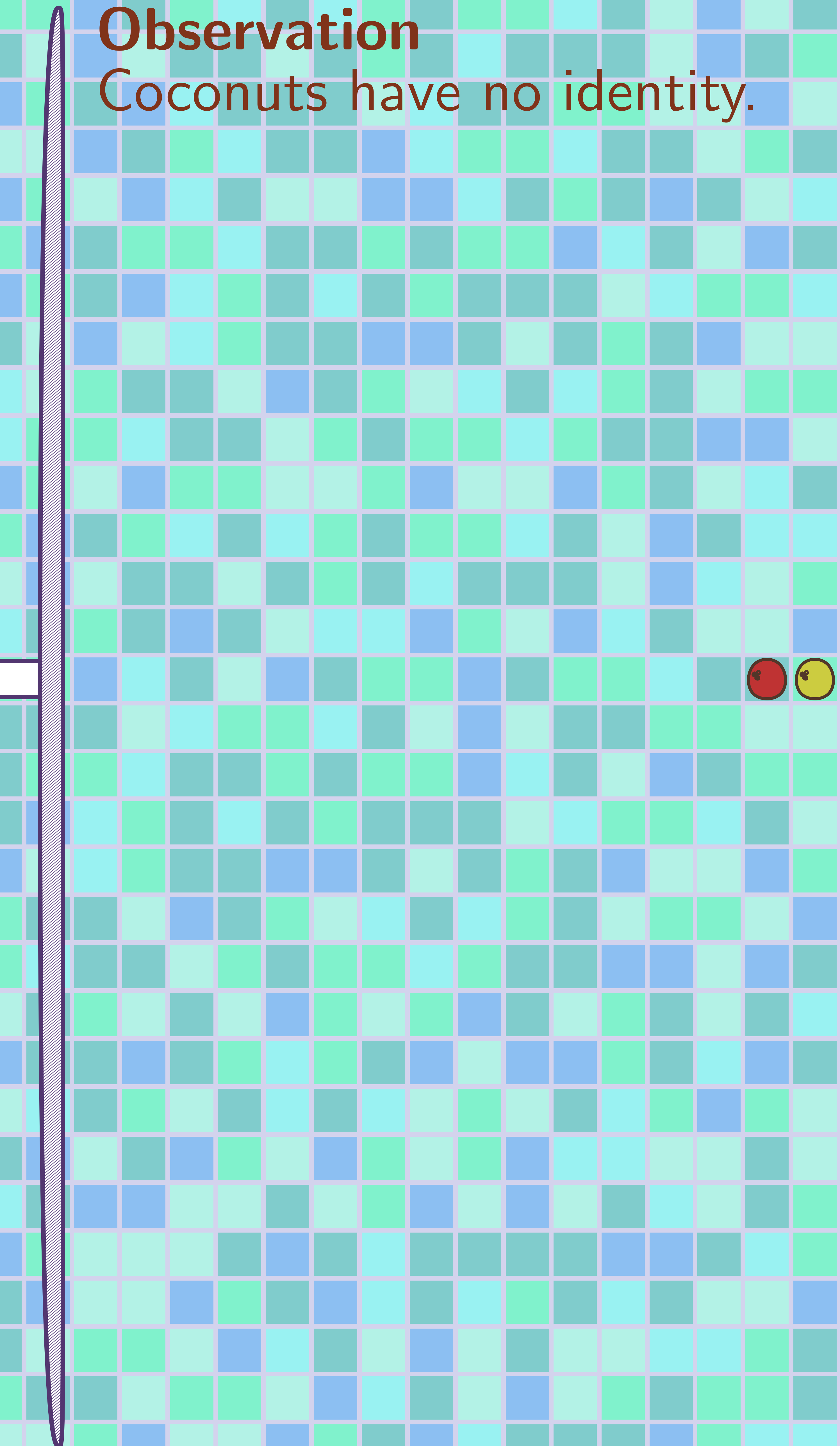
Observation

Coconuts have no identity.



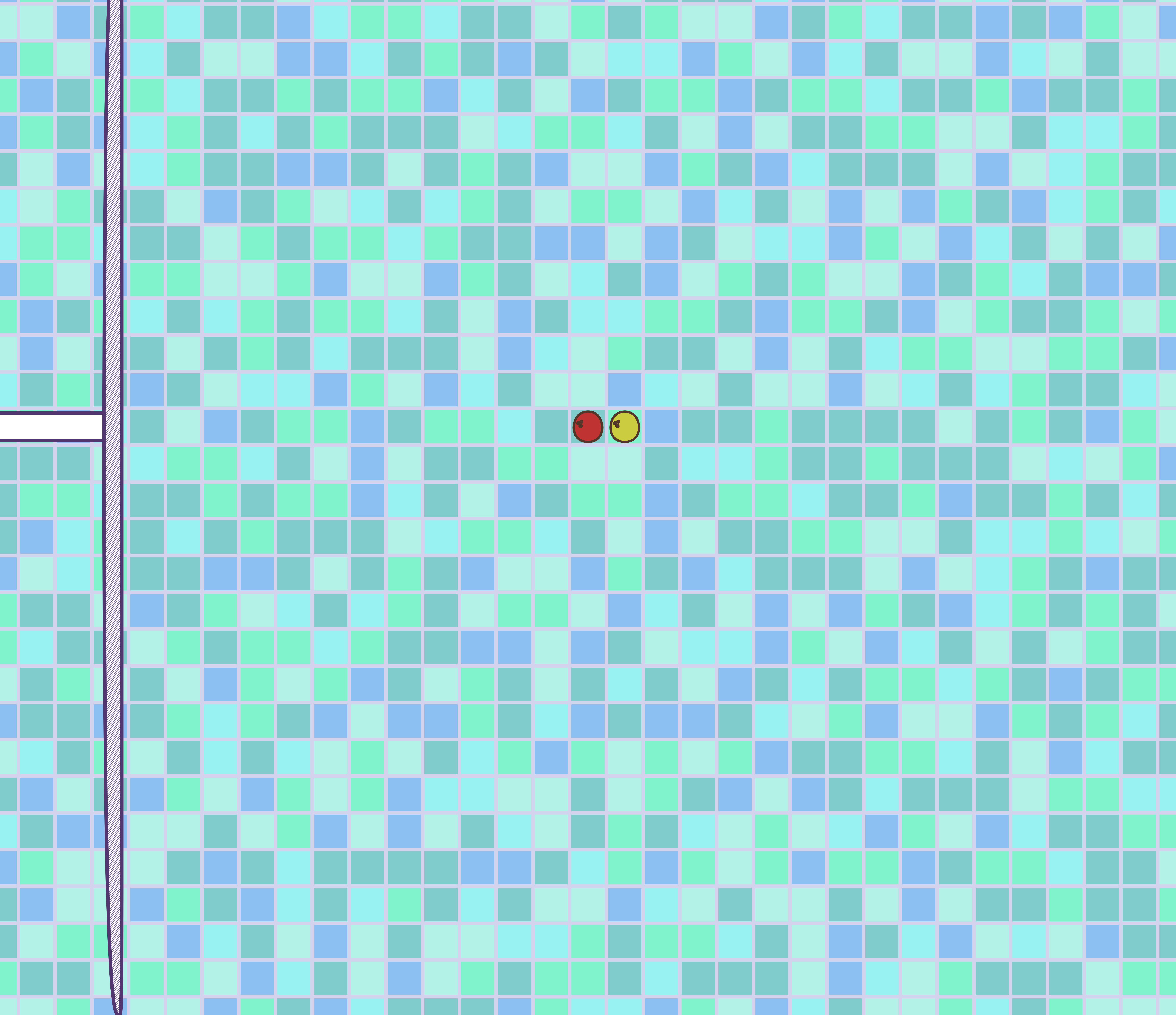
Observation

Coconuts have no identity.



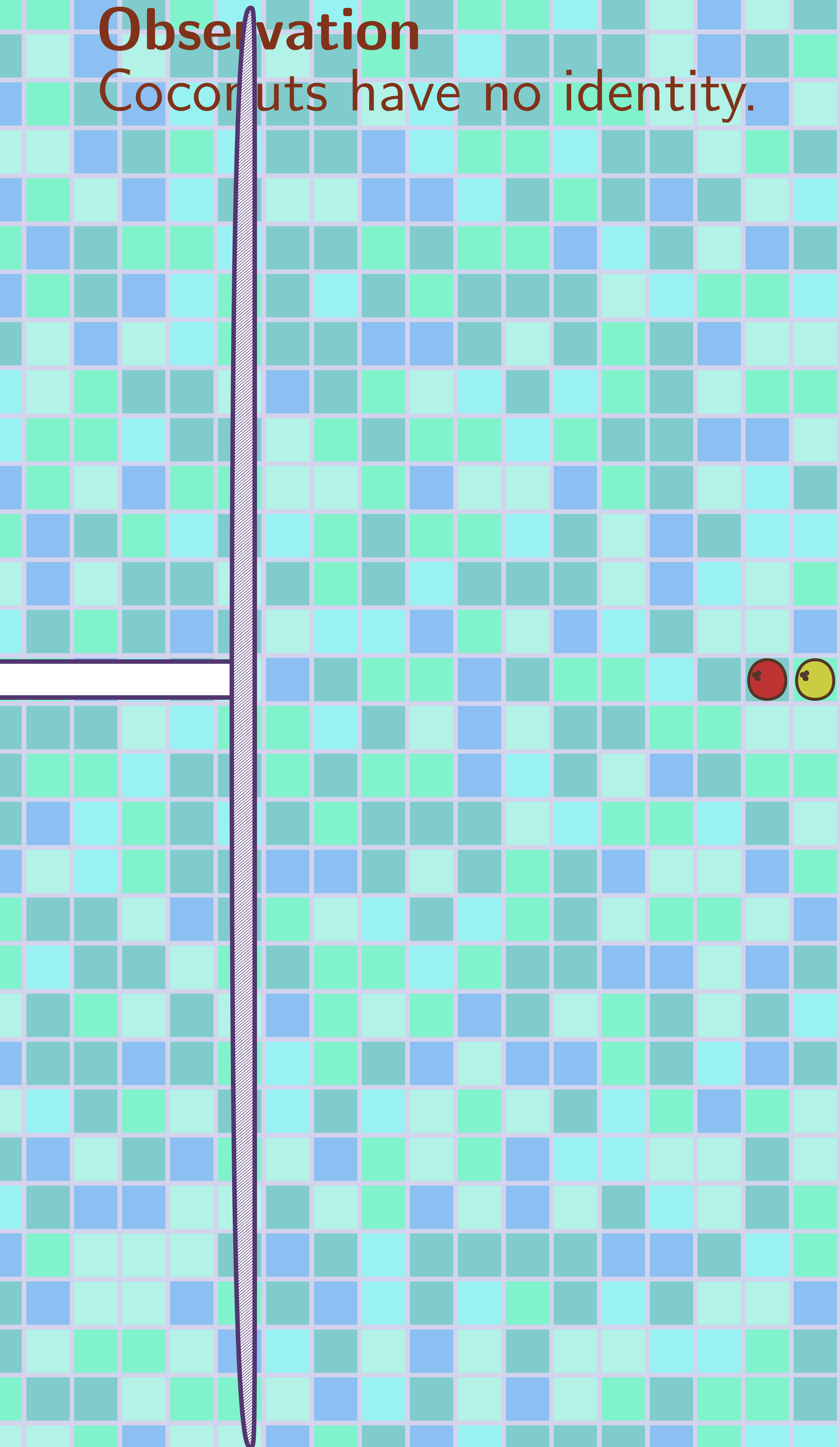
Observation

Coconuts have no identity.



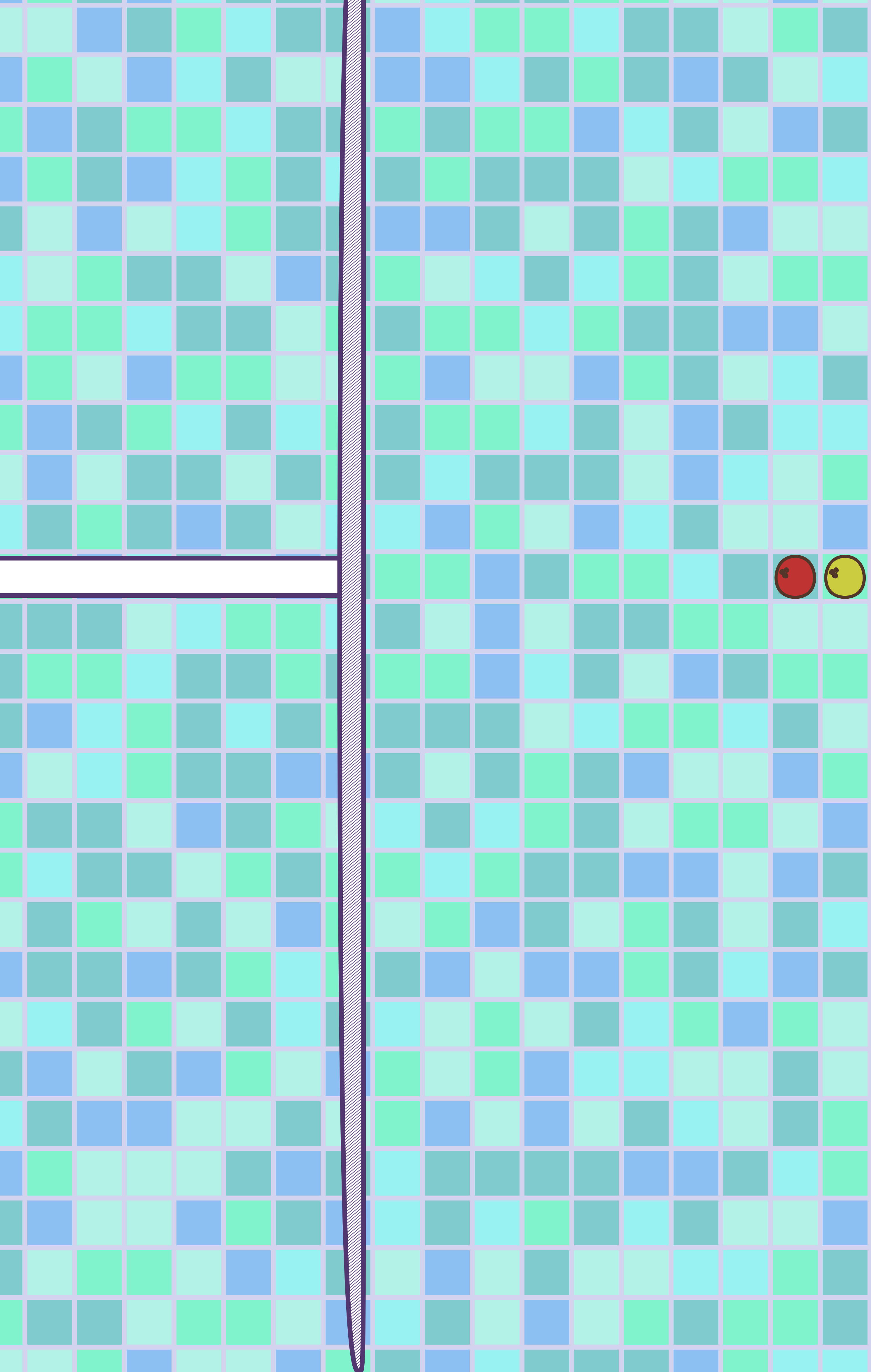
Observation

Coconuts have no identity.



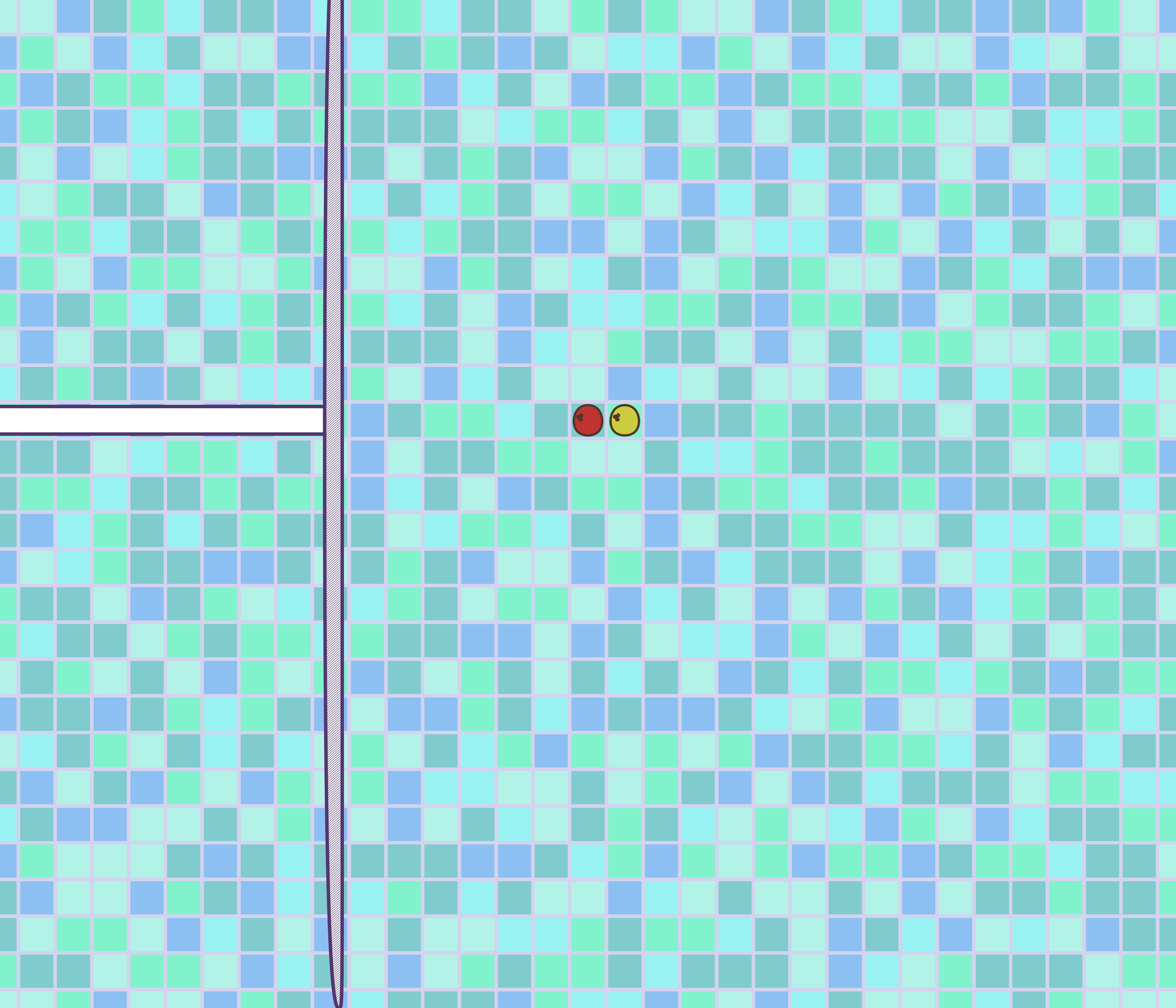
Observation

Coconuts have no identity.



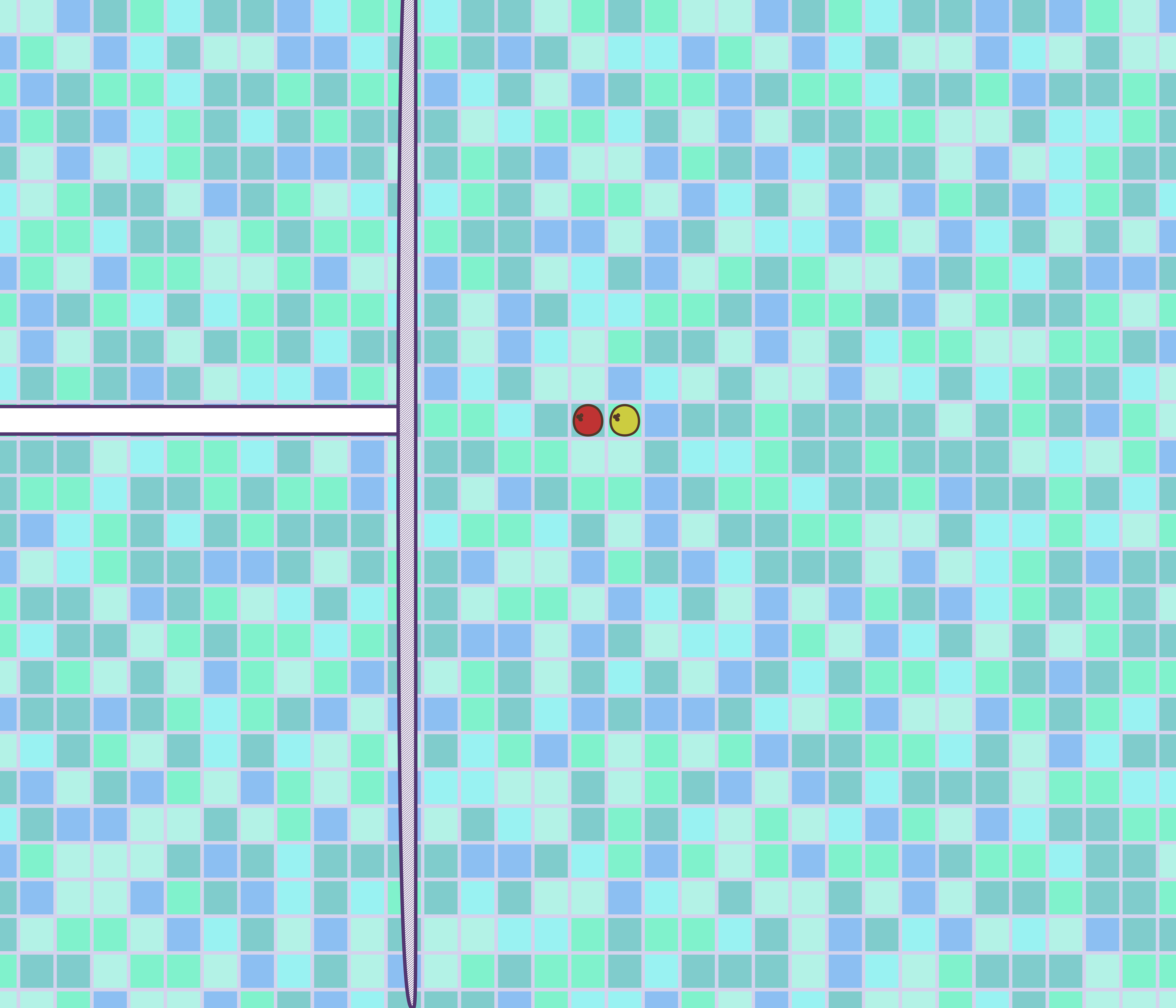
Observation

Coconuts have no identity.



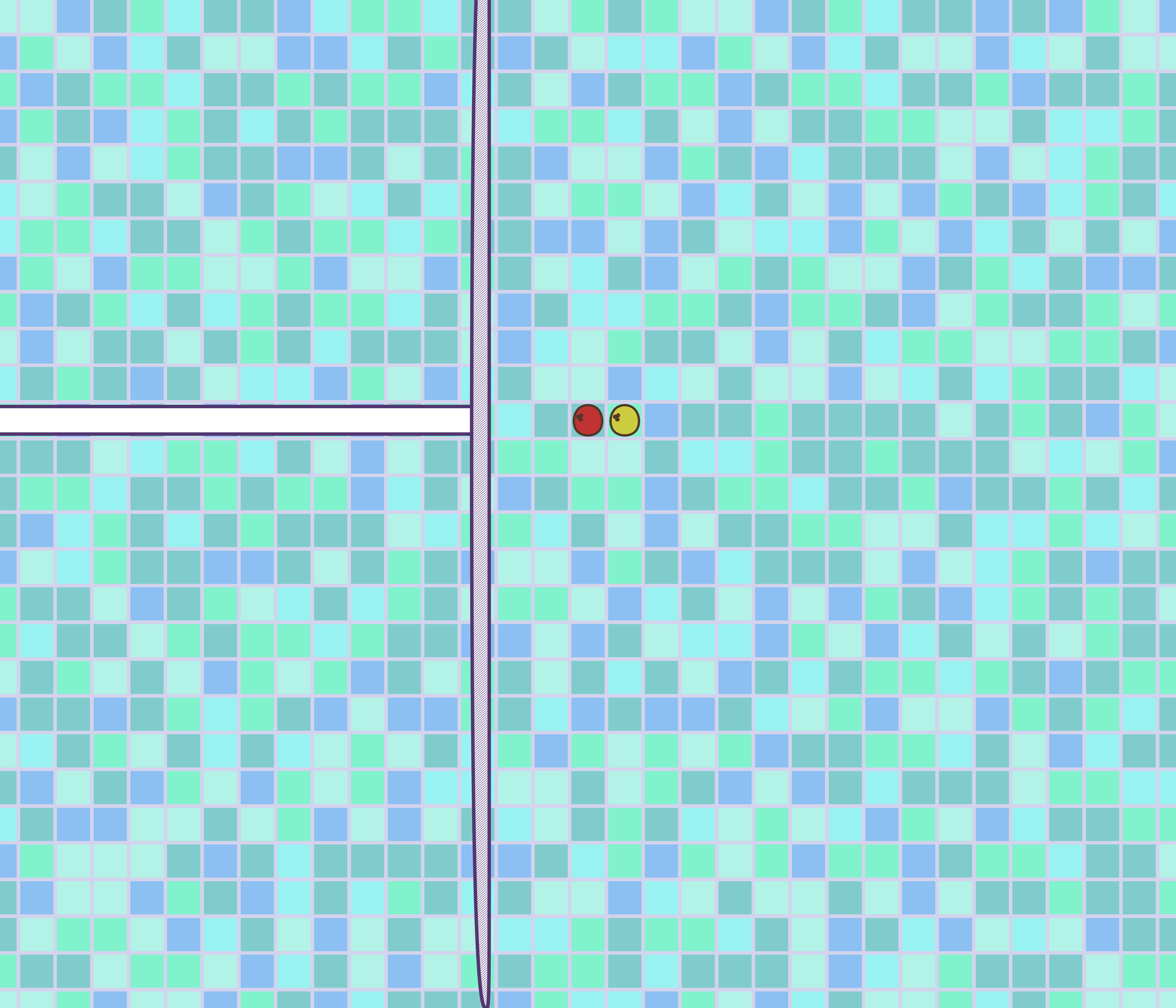
Observation

Coconuts have no identity.



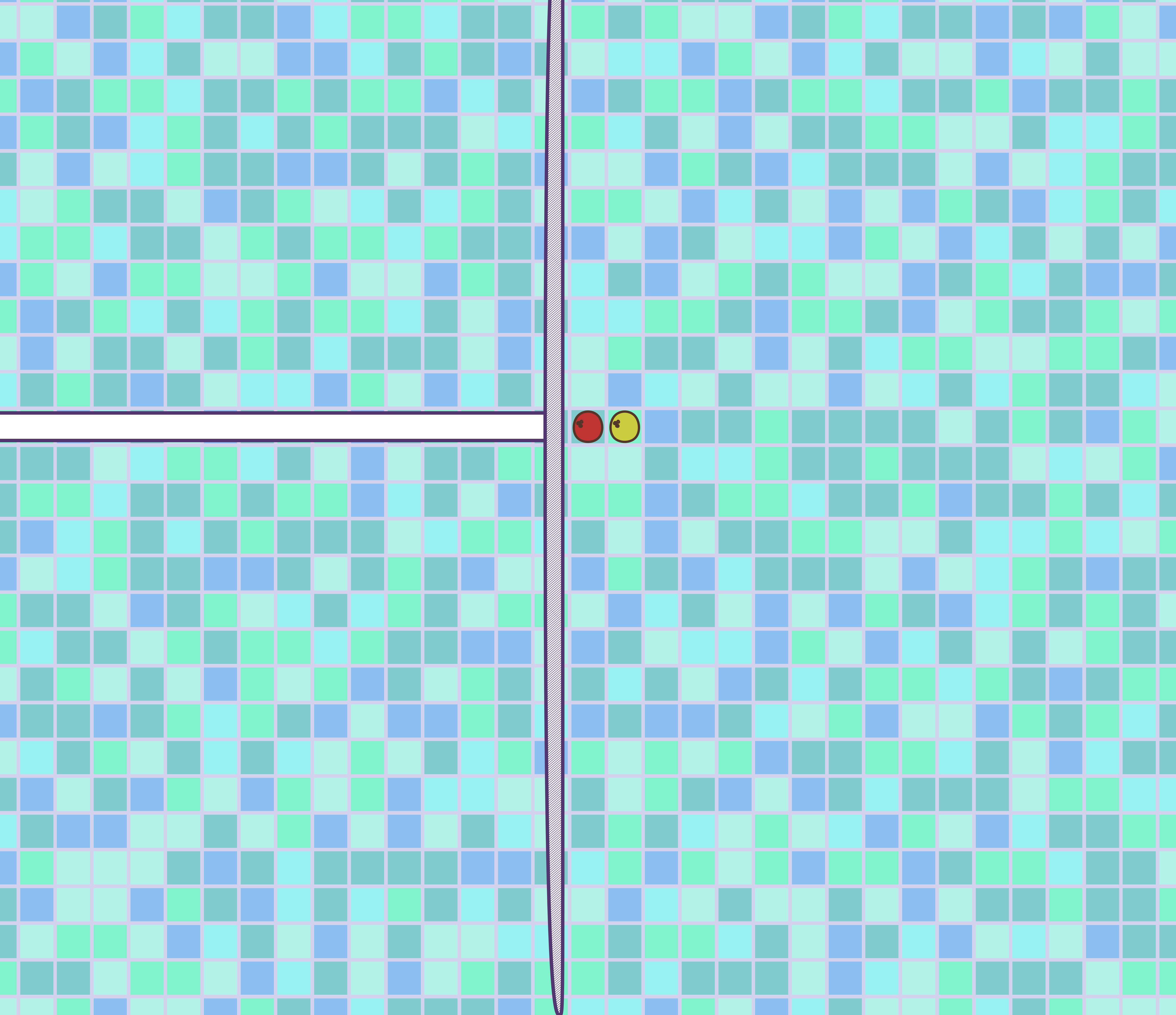
Observation

Coconuts have no identity.



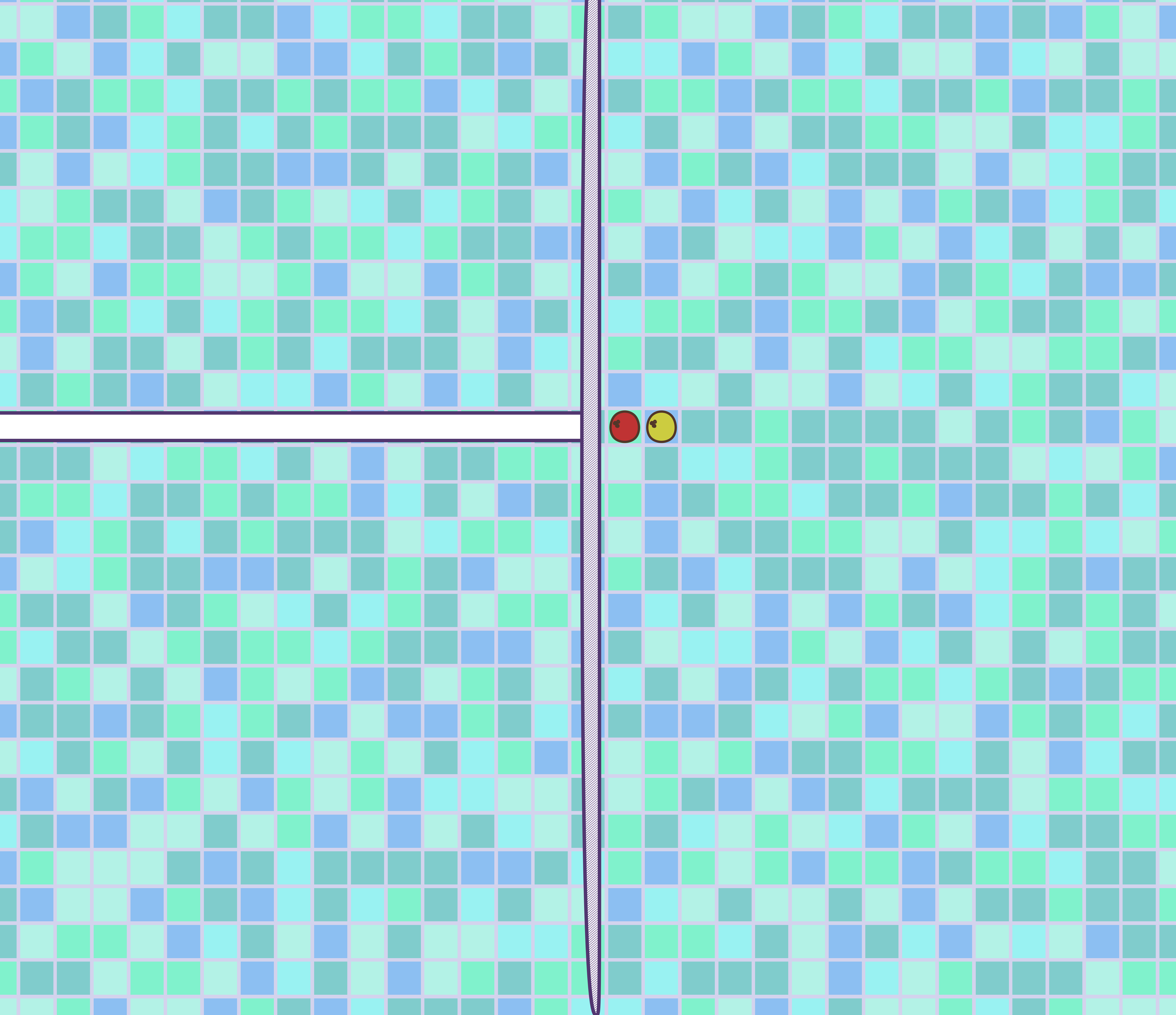
Observation

Coconuts have no identity.



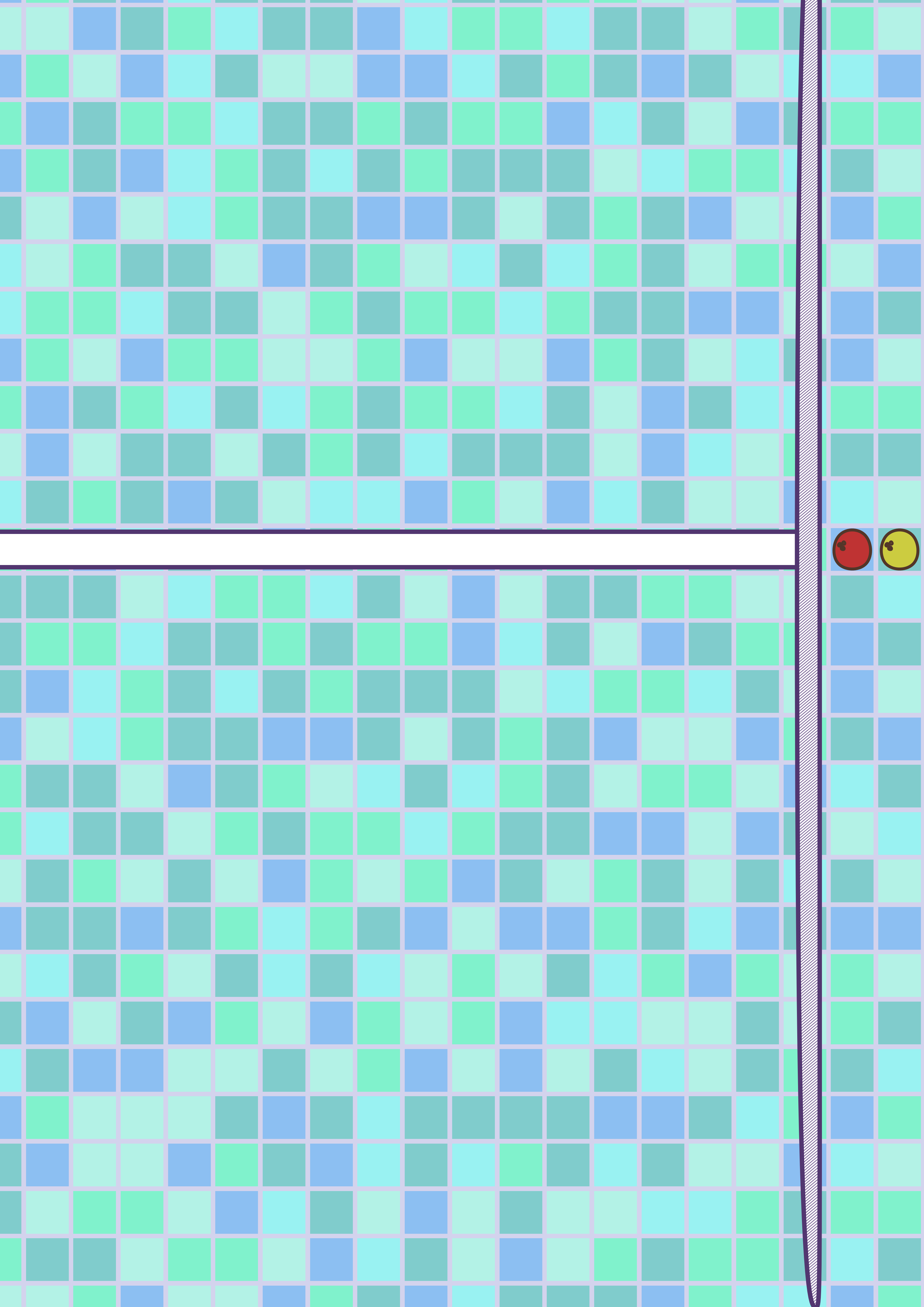
Observation

Coconuts have no identity.



Observation

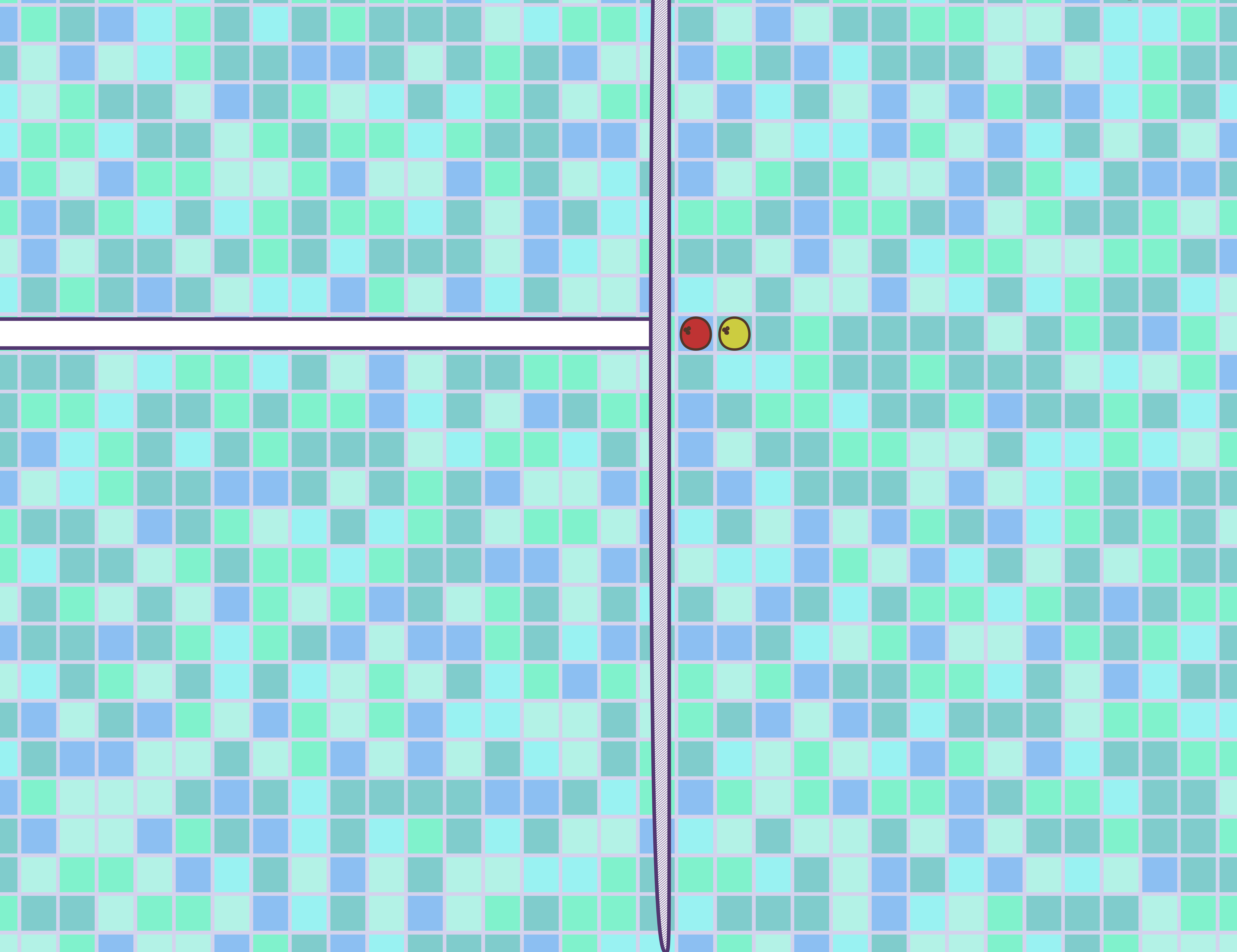
Coconuts have no identity.



Observation

Coconuts have no identity.

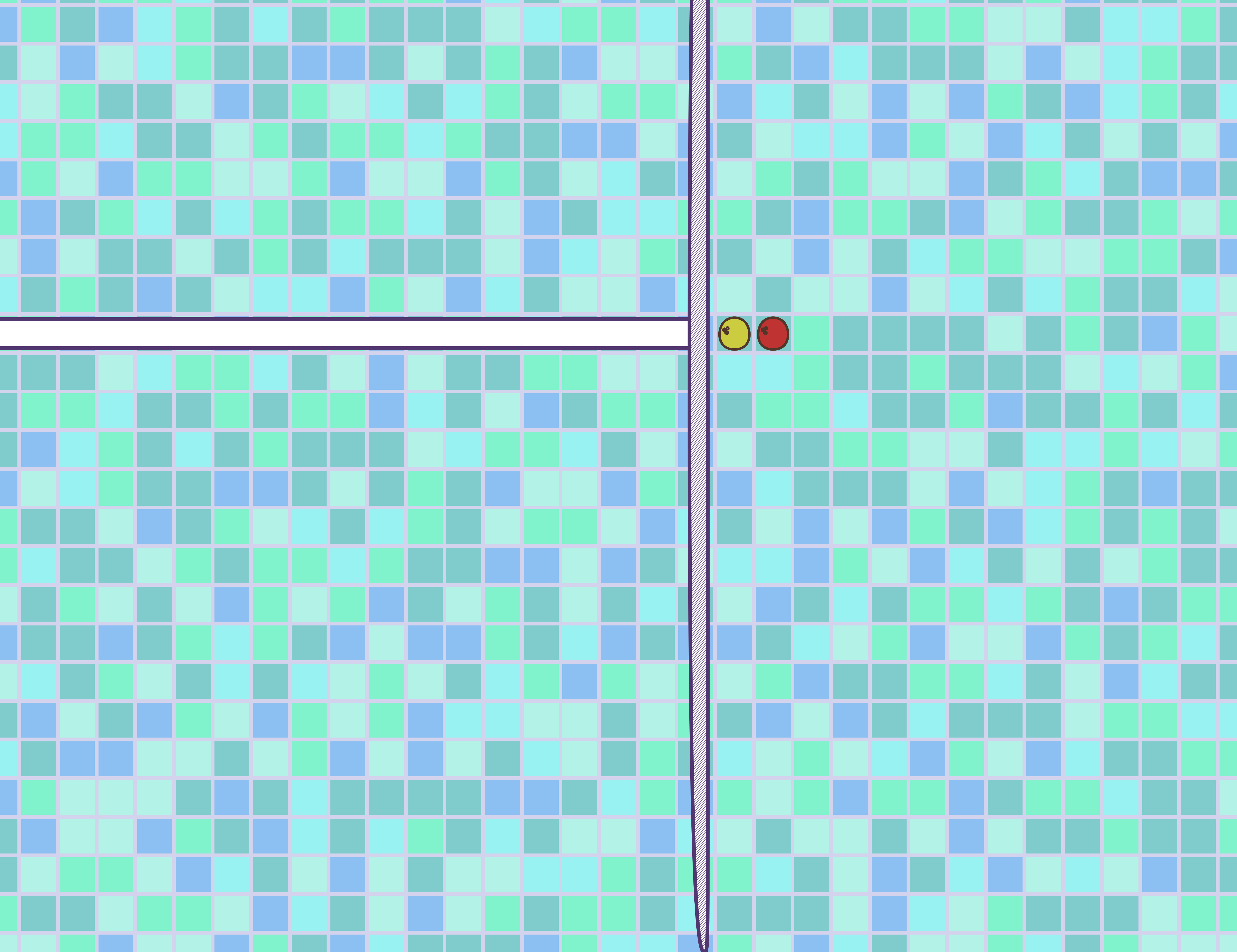
Instead of pushing a whole row of coconuts, we could also move the coconut we hit to the first available spot.



Observation

Coconuts have no identity.

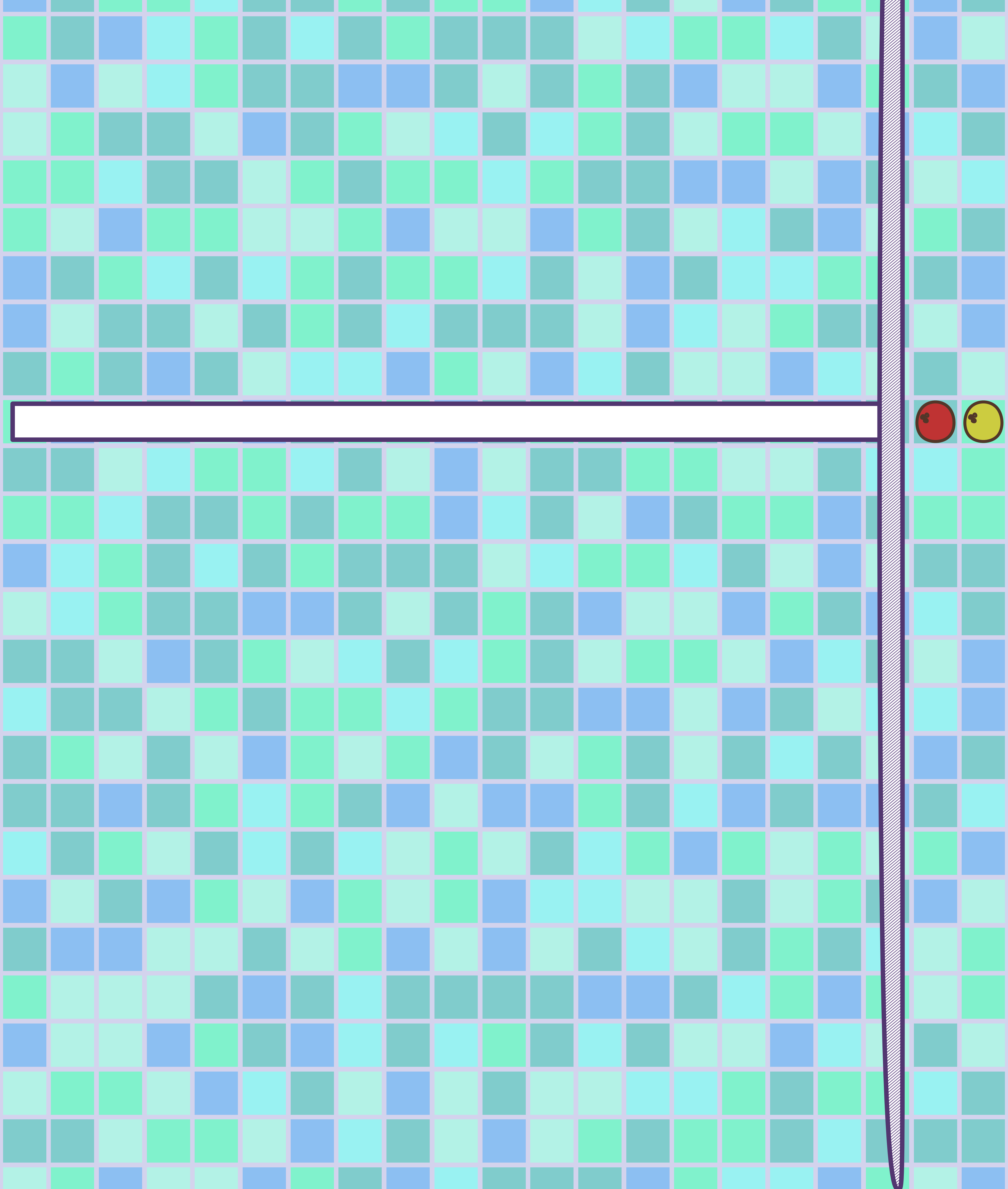
Instead of pushing a whole row of coconuts, we could also move the coconut we hit to the first available spot.



Observation

Coconuts have no identity.

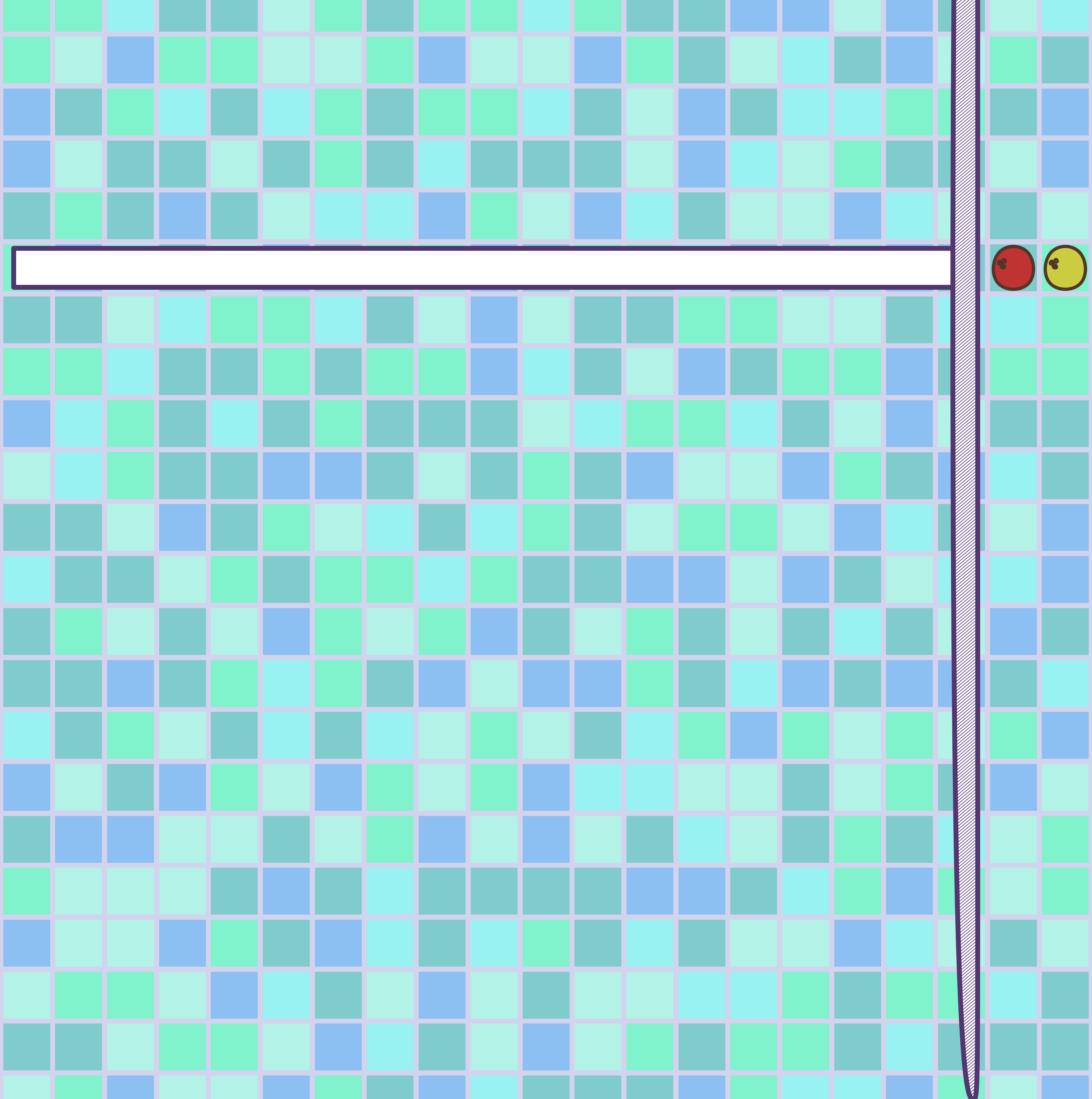
Instead of pushing a whole row of coconuts, we could also move the coconut we hit to the first available spot.



Observation
Coconuts have no identity.

Instead of pushing a whole row of coconuts, we could also move the coconut we hit to the first available spot.

We will use the second rule from now on, to make it easier to follow what happens...

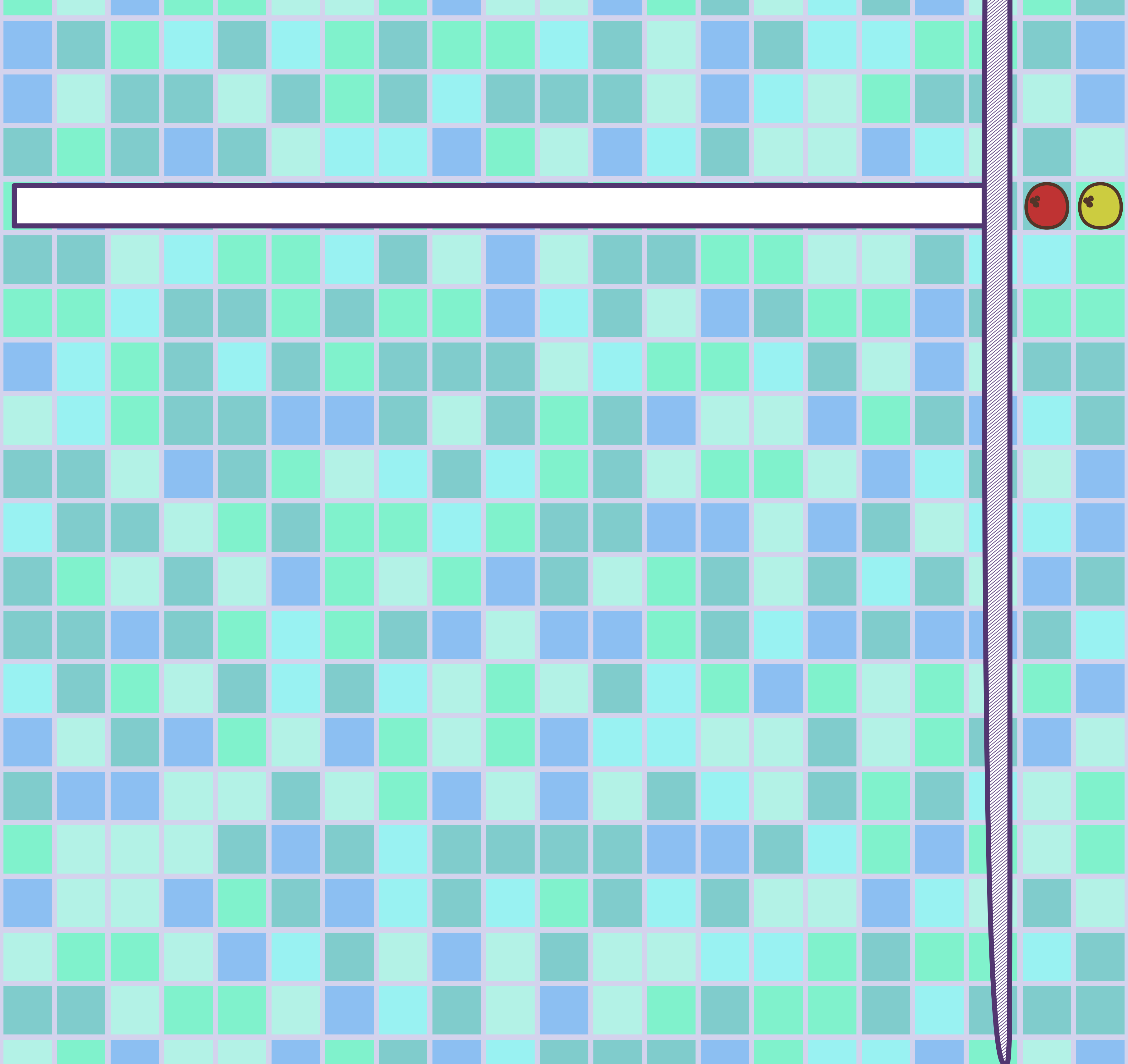


Observation
Coconuts have no identity.

Instead of pushing a whole row of coconuts, we could also move the coconut we hit to the first available spot.

We will use the second rule from now on, to make it easier to follow what happens...

...just remember that all coconuts are equal!



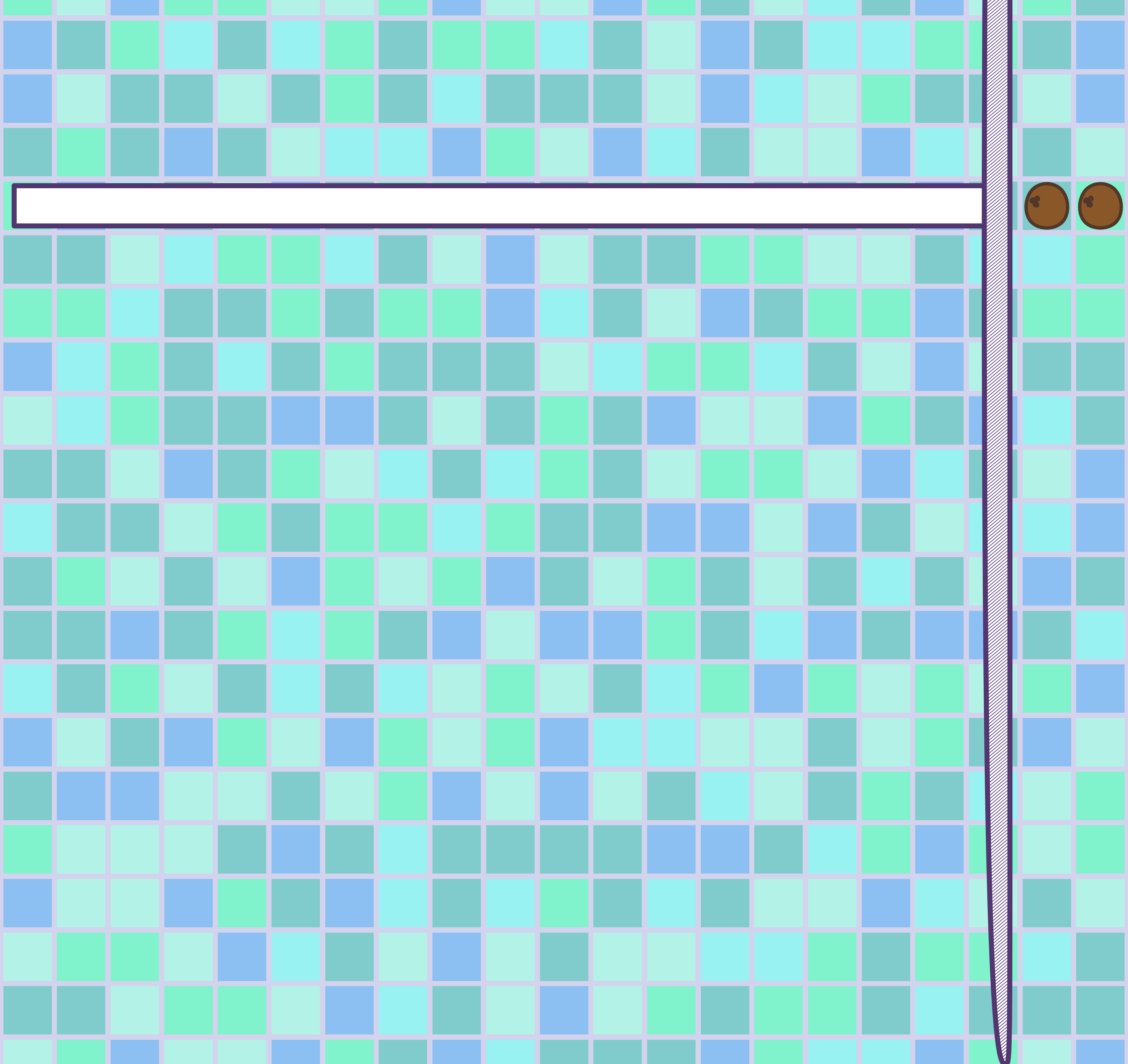
Observation

Coconuts have no identity.

Instead of pushing a whole row of coconuts, we could also move the coconut we hit to the first available spot.

We will use the second rule from now on, to make it easier to follow what happens...

...just remember that all coconuts are equal!



HARD COCONUTS

Theorem

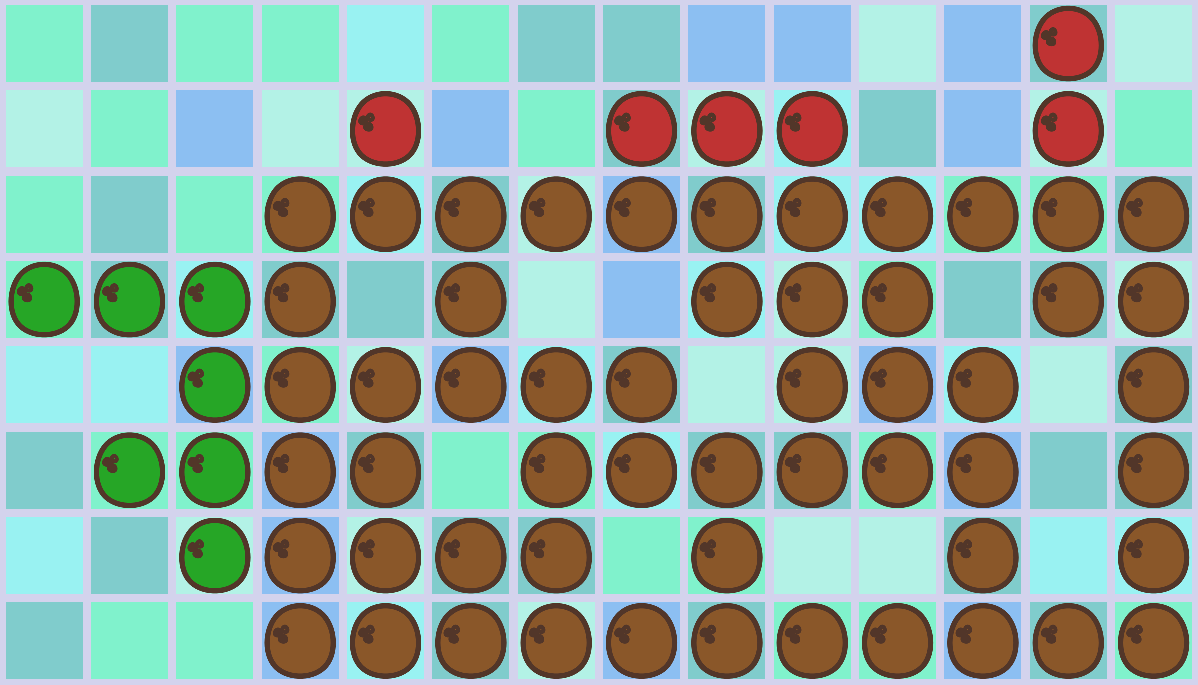
Decising coconut compactability is NP-hard.

Theorem
Decising coconut compactability is NP-hard.

General idea: we will create a big coconut rectangle with holes, and put some coconuts on the side to fill them.

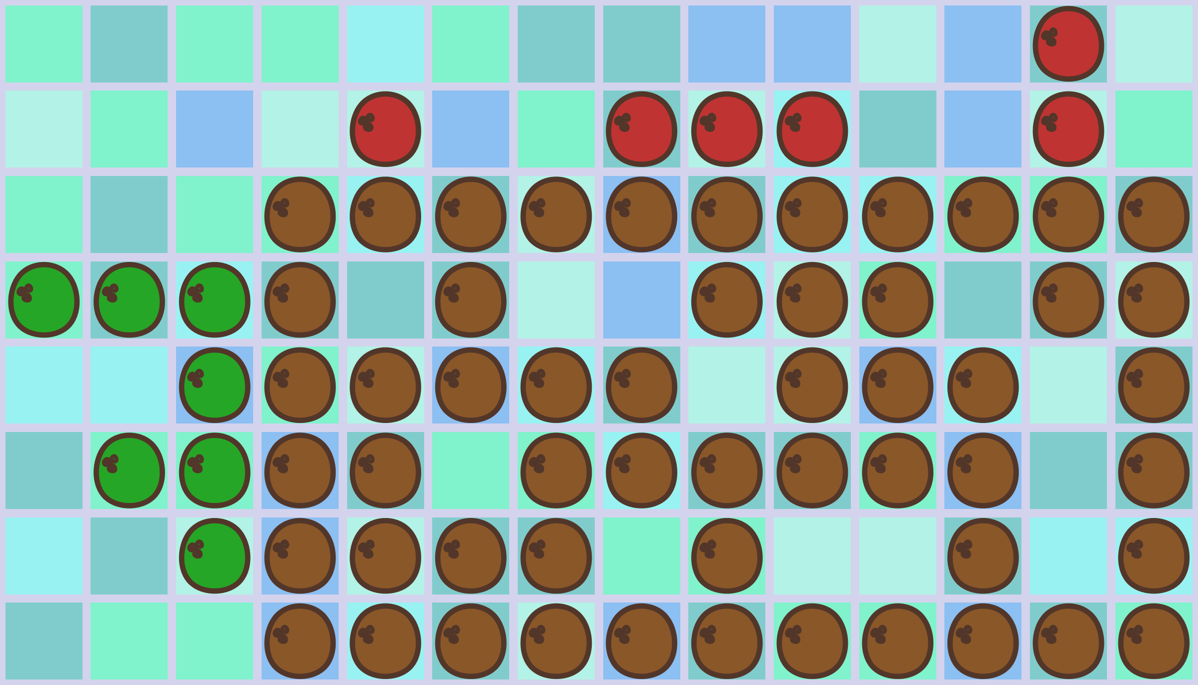
Theorem
Decising coconut compactability is NP-hard.

General idea: we will create a big coconut rectangle with holes, and put some coconuts on the side to fill them.



Theorem
Decising coconut compactability is NP-hard.

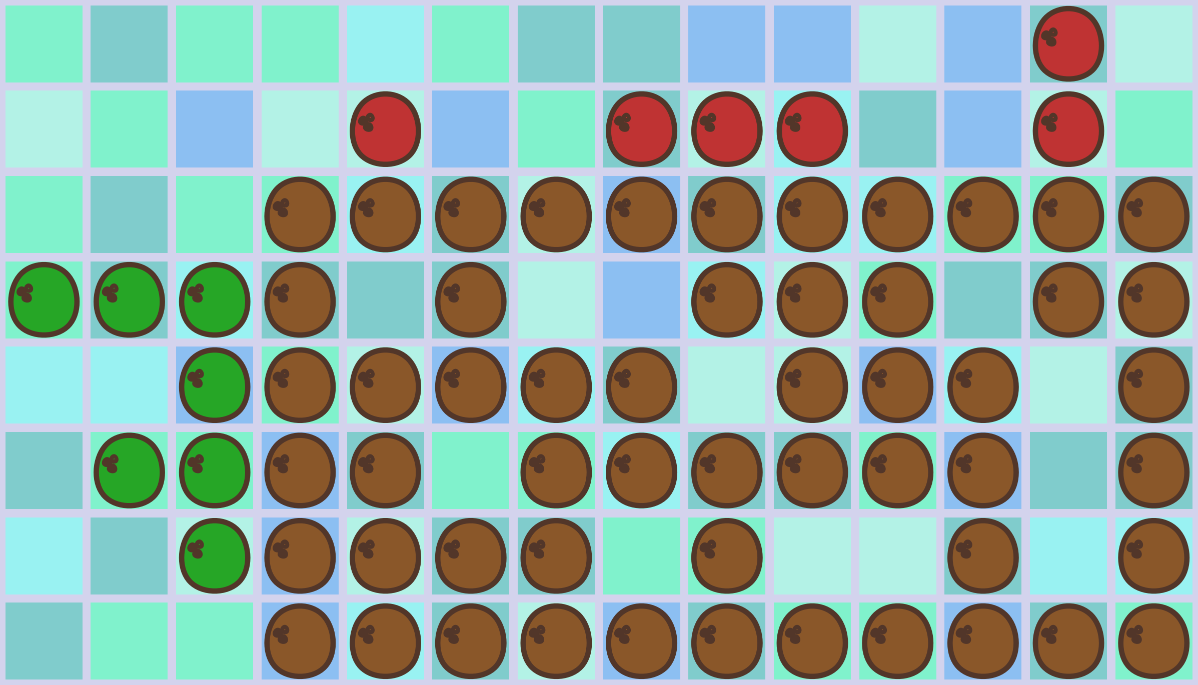
General idea: we will create a big coconut rectangle with holes, and put some coconuts on the side to fill them.



For simplicity, we will assume only two push directions for now.

Theorem
Decising coconut compactability is NP-hard.

General idea: we will create a big coconut rectangle with holes, and put some coconuts on the side to fill them.



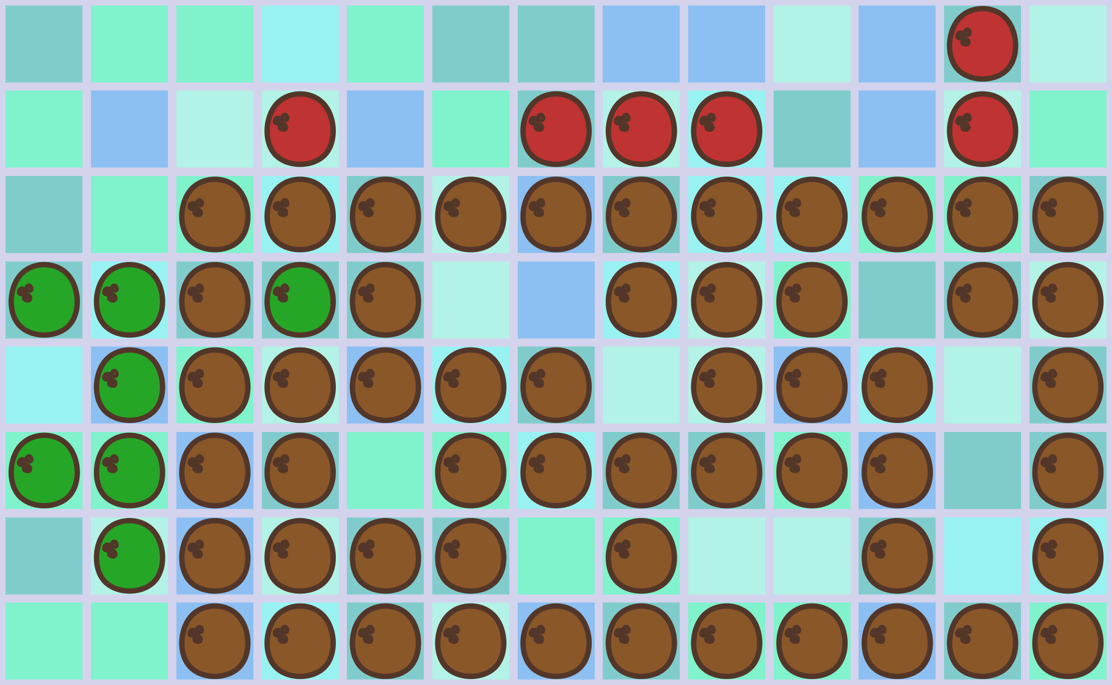
For simplicity, we will assume only two push directions for now.

The question then becomes in which order to push to make everything fit.

Theorem

Decising coconut compactability is NP-hard.

General idea: we will create a big coconut rectangle with holes, and put some coconuts on the side to fill them.

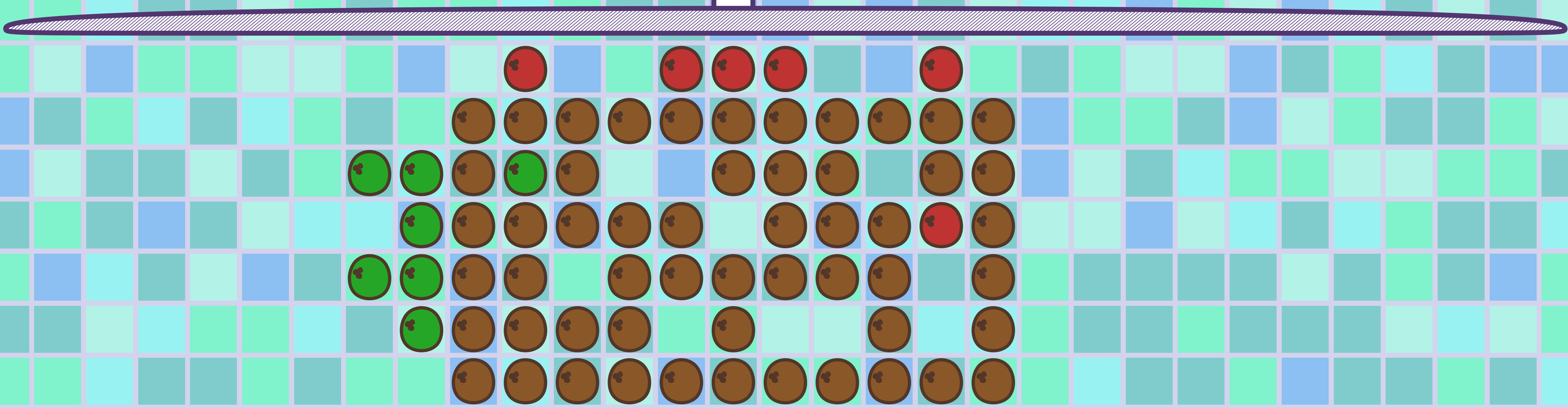


For simplicity, we will assume only two push directions for now.

The question then becomes in which order to push to make everything fit.

Theorem
Decising coconut compactability is NP-hard.

General idea: we will create a big coconut rectangle with holes, and put some coconuts on the side to fill them.

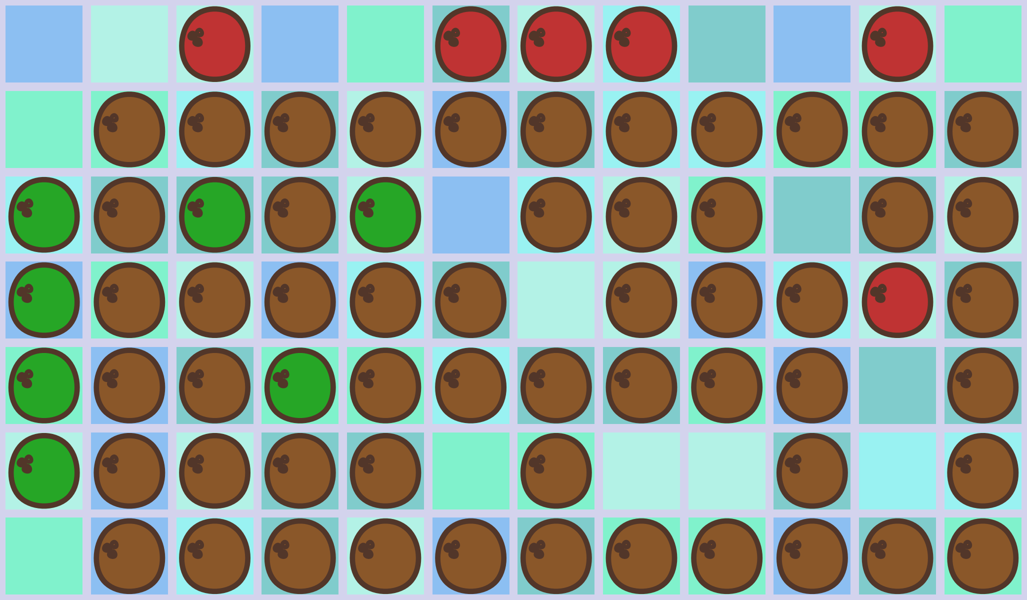


For simplicity, we will assume only two push directions for now.

The question then becomes in which order to push to make everything fit.

Theorem
Decising coconut compactability is NP-hard.

General idea: we will create a big coconut rectangle with holes, and put some coconuts on the side to fill them.



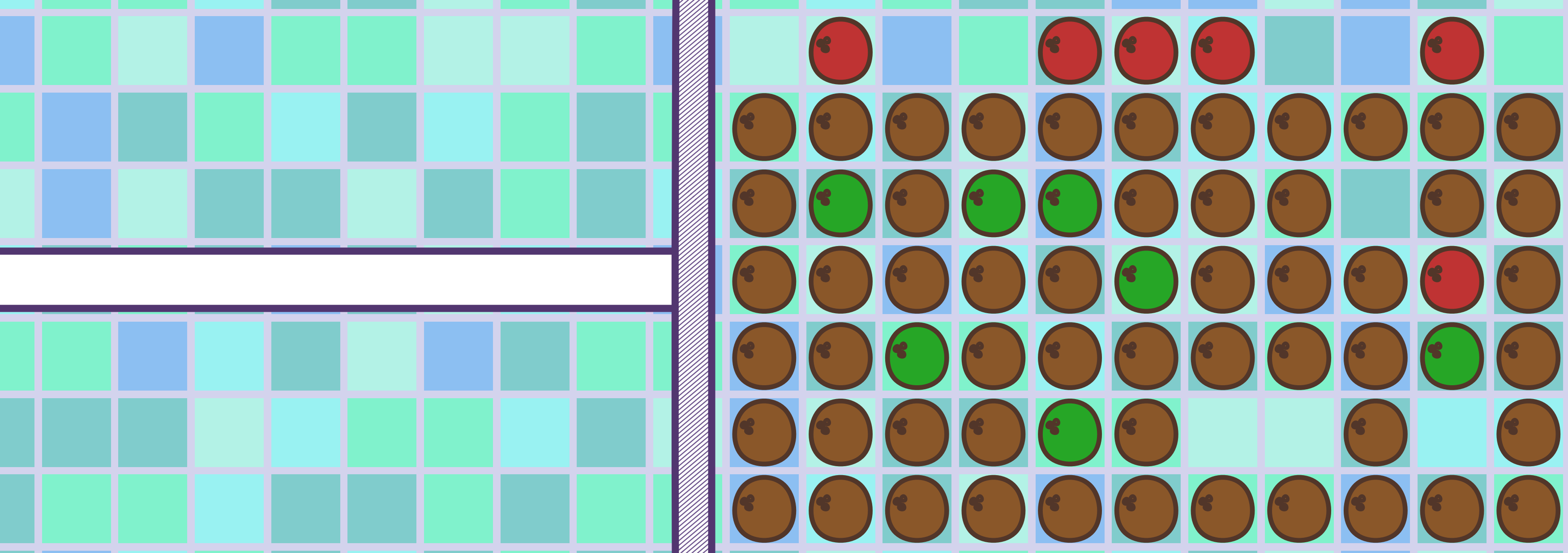
For simplicity, we will assume only two push directions for now.

The question then becomes in which order to push to make everything fit.

Theorem

Decising coconut compactability is NP-hard.

General idea: we will create a big coconut rectangle with holes, and put some coconuts on the side to fill them.



For simplicity, we will assume only two push directions for now.

The question then becomes in which order to push to make everything fit.

Theorem
Decising coconut compatability is NP-hard.

General idea: we will create a big coconut rectangle with holes, and put some coconuts on the side to fill them.

For simplicity, we will assume only two push directions for now.

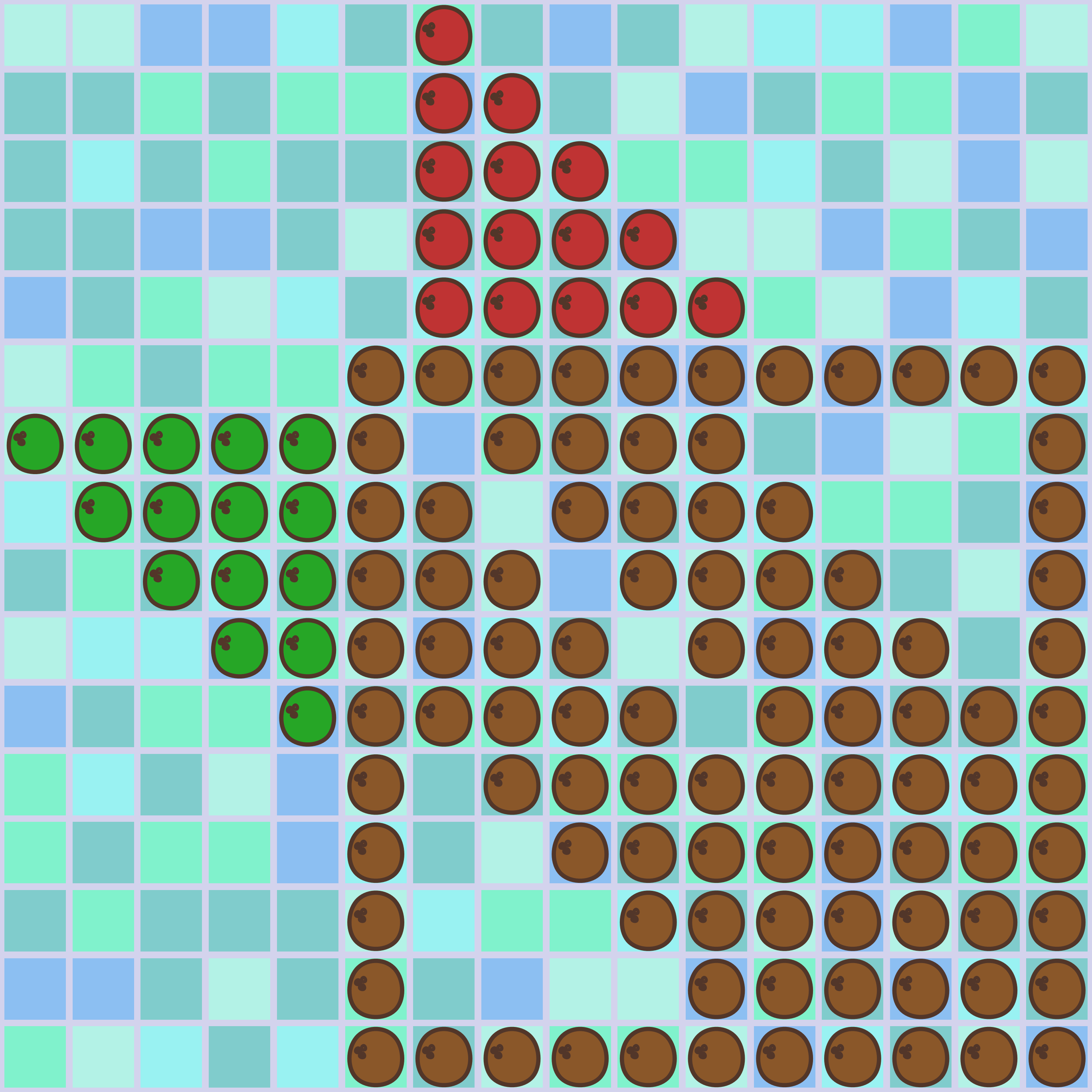
The question then becomes in which order to push to make everything fit.

Idea

Let’s encode a sequence of choices.

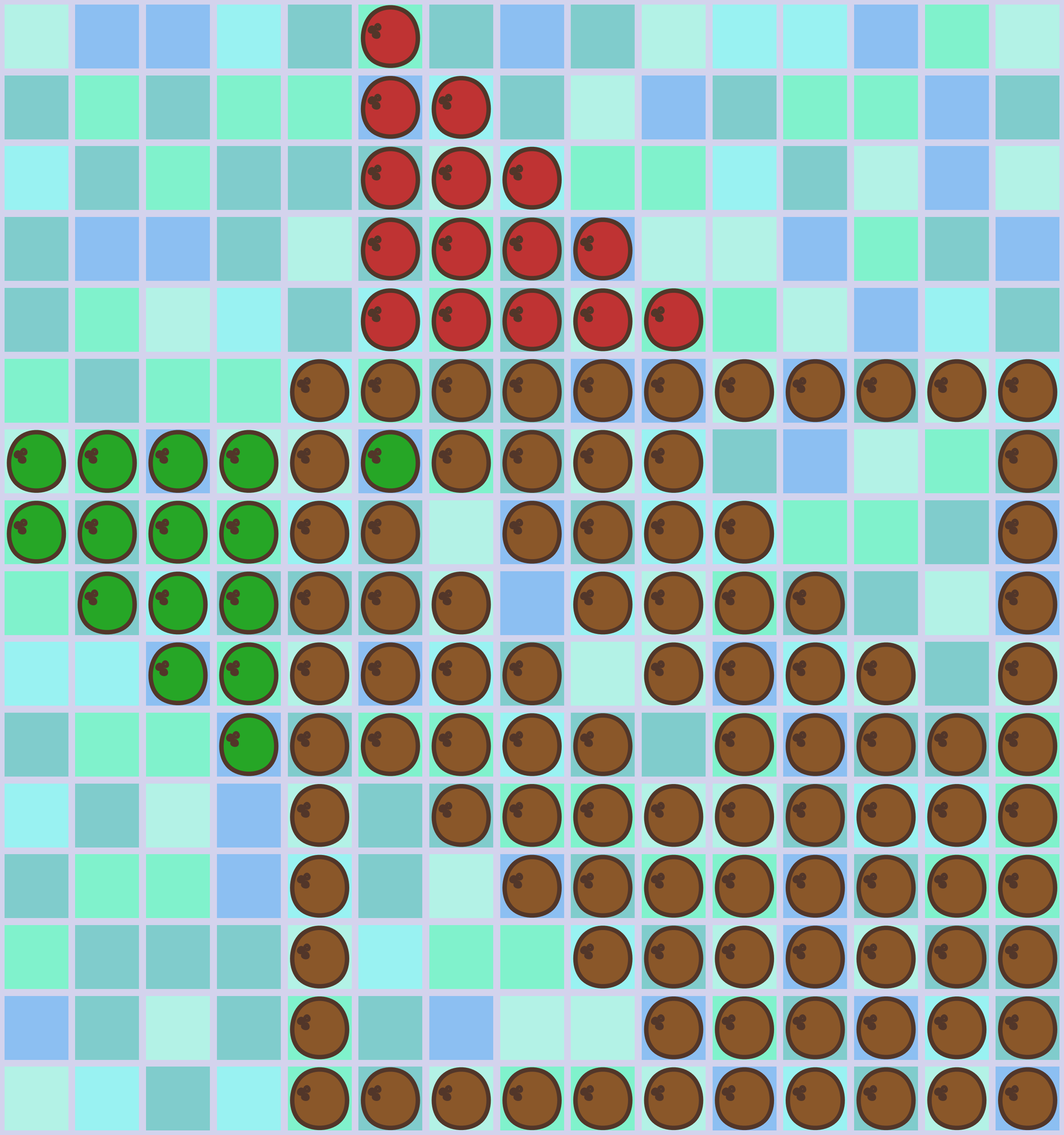
Idea

Let's encode a sequence of choices.



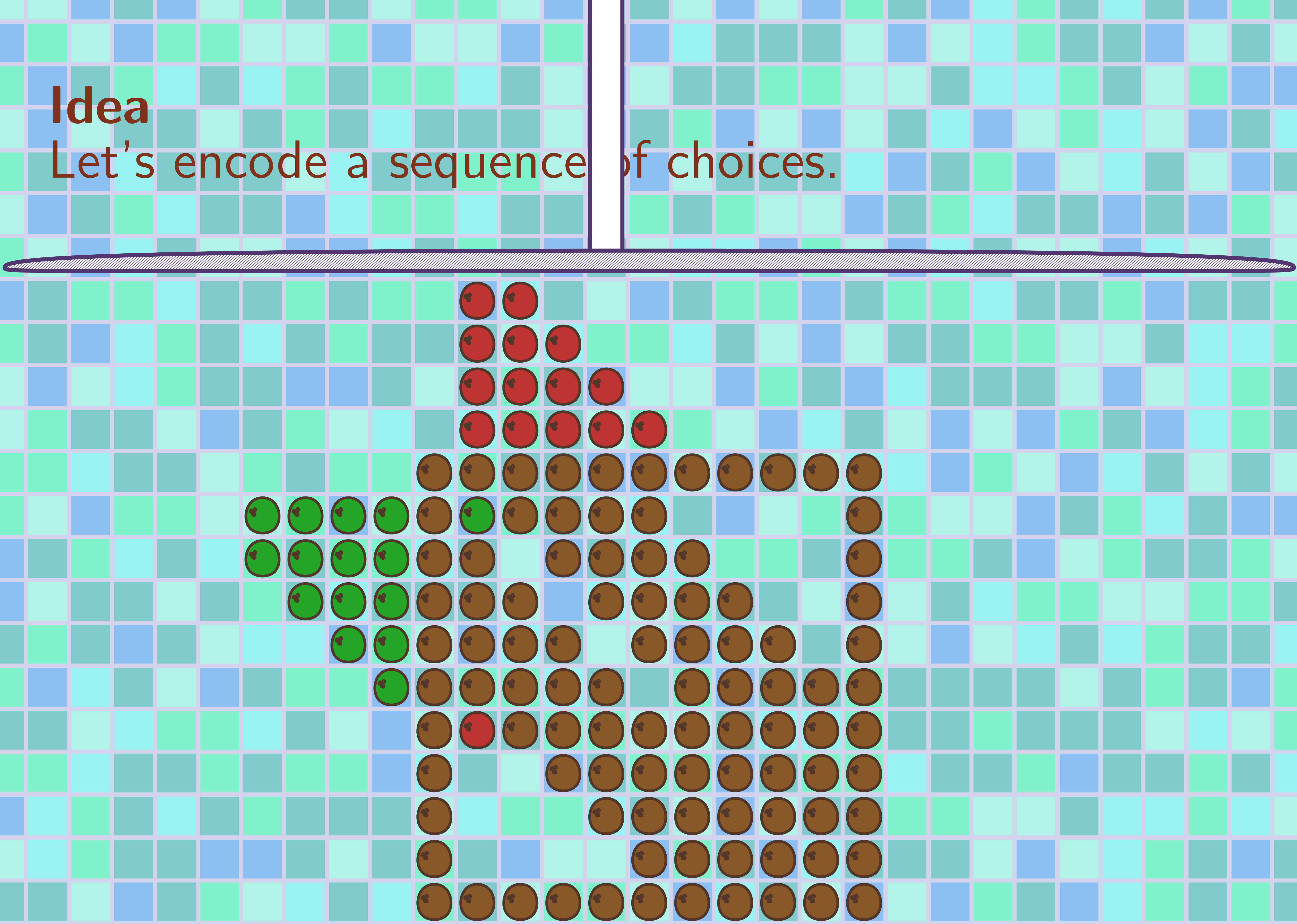
Idea

Let's encode a sequence of choices.



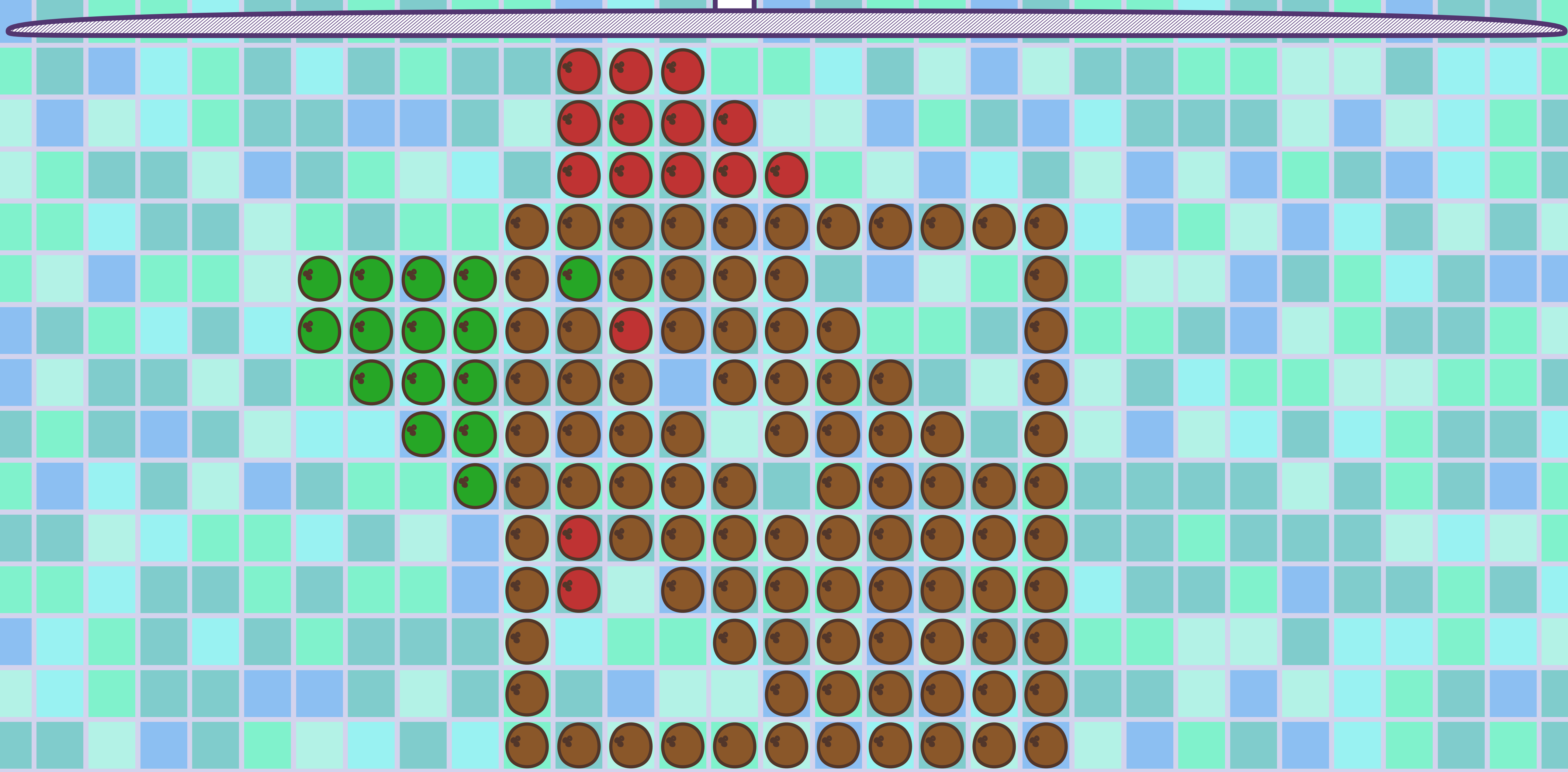
Idea

Let's encode a sequence of choices.



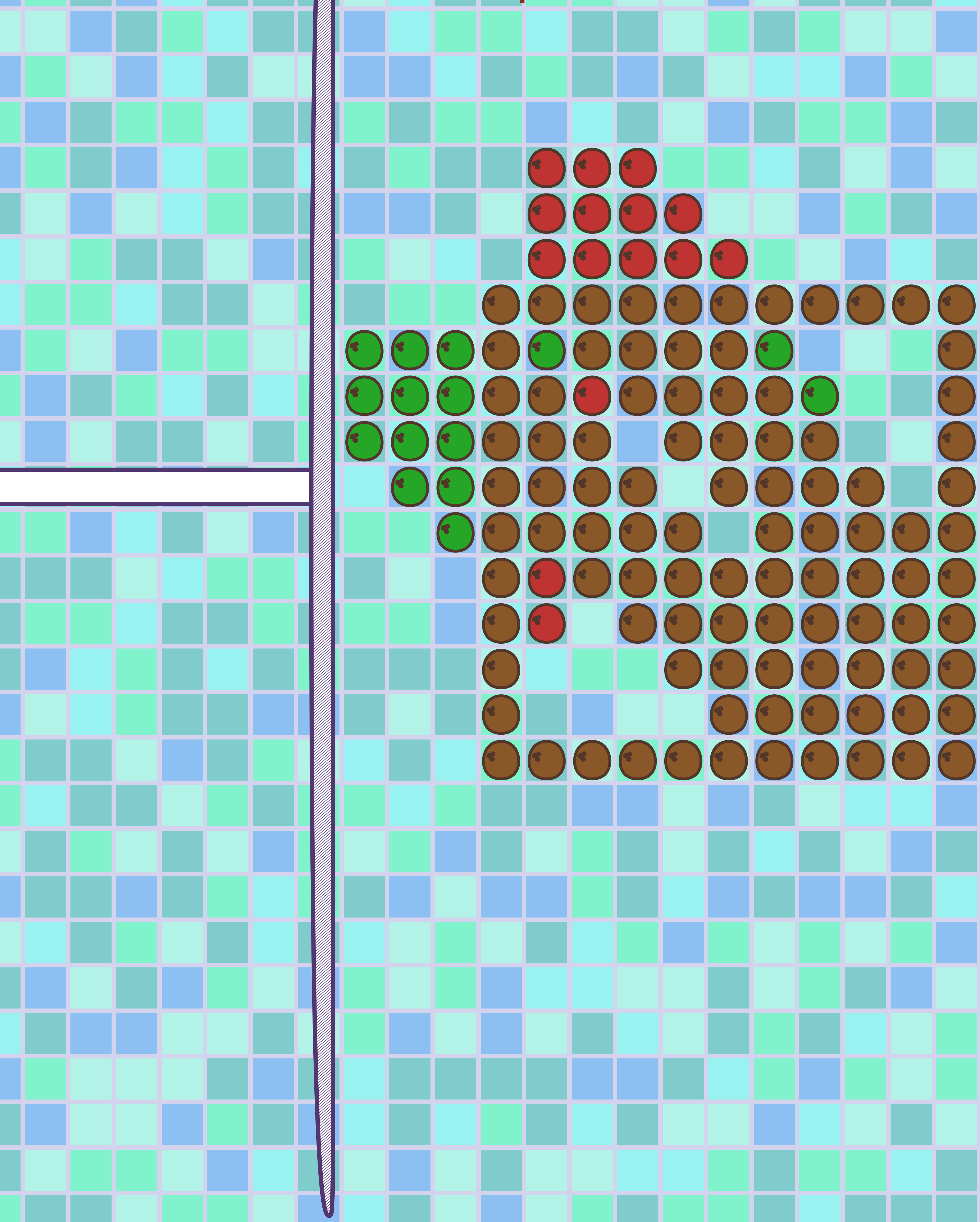
Idea

Let's encode a sequence of choices.



Idea

Let's encode a sequence of choices.

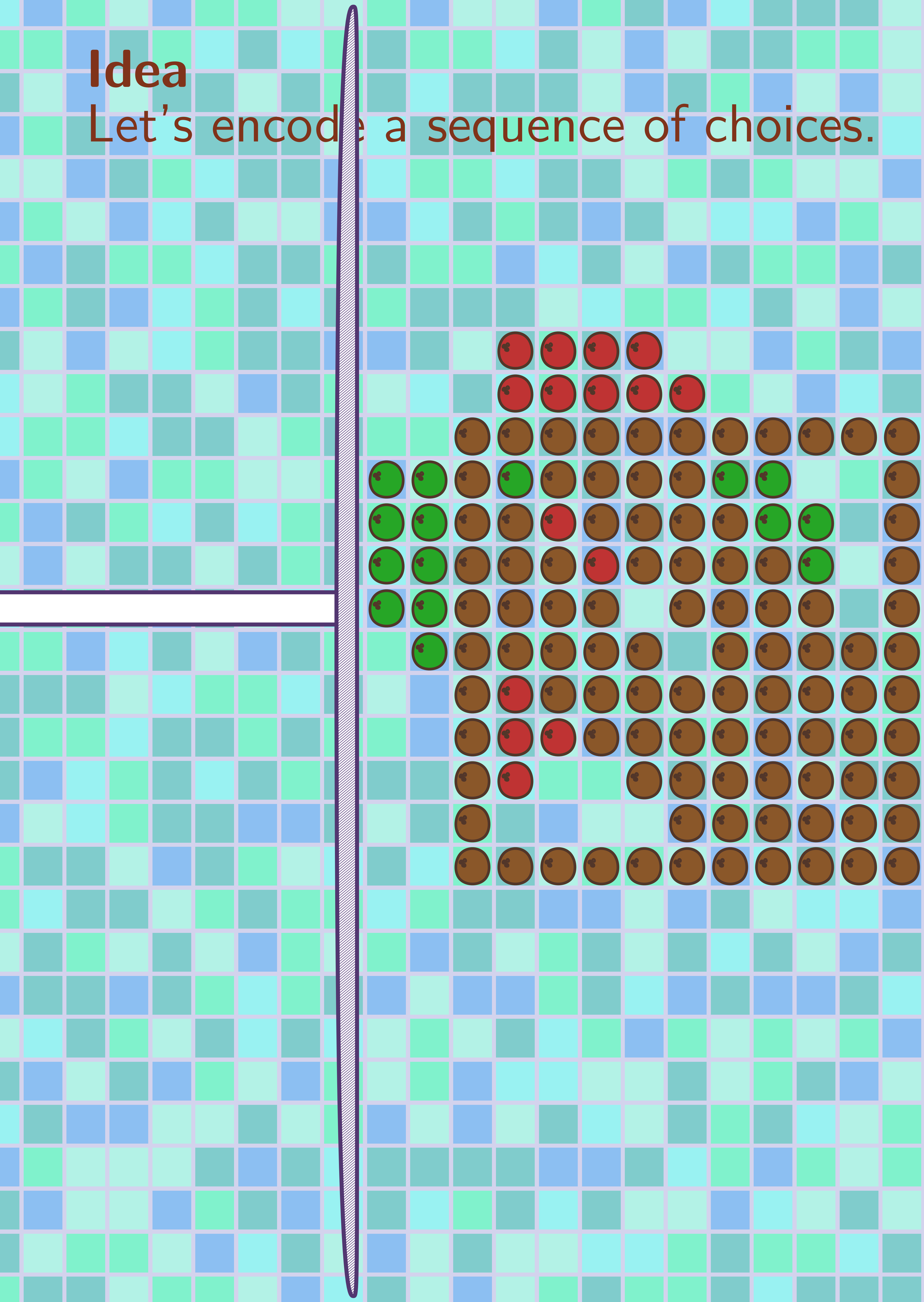


Idea

Let's encode a sequence of choices.

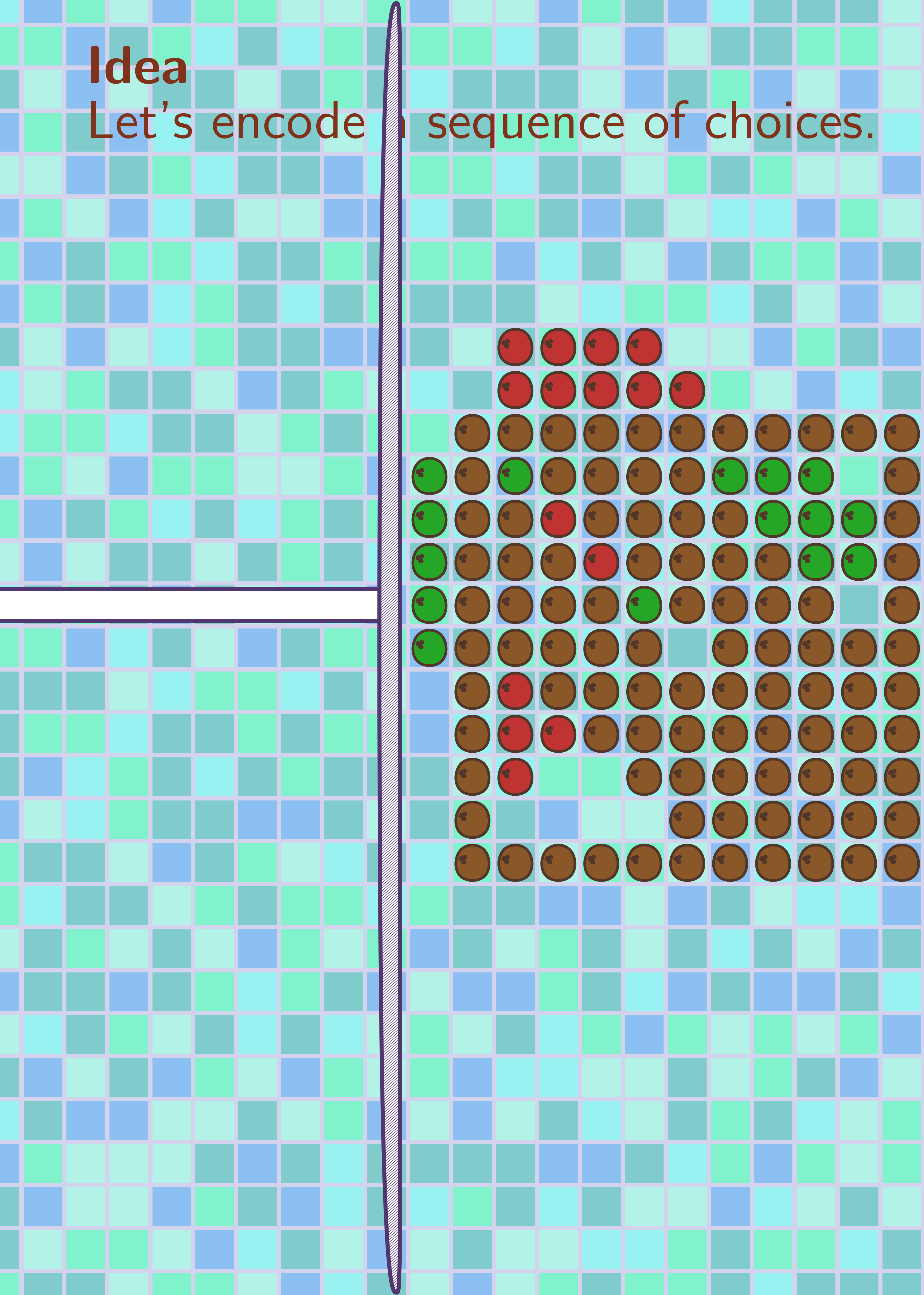
Idea

Let's encode a sequence of choices.



Idea

Let's encode a sequence of choices.



Idea

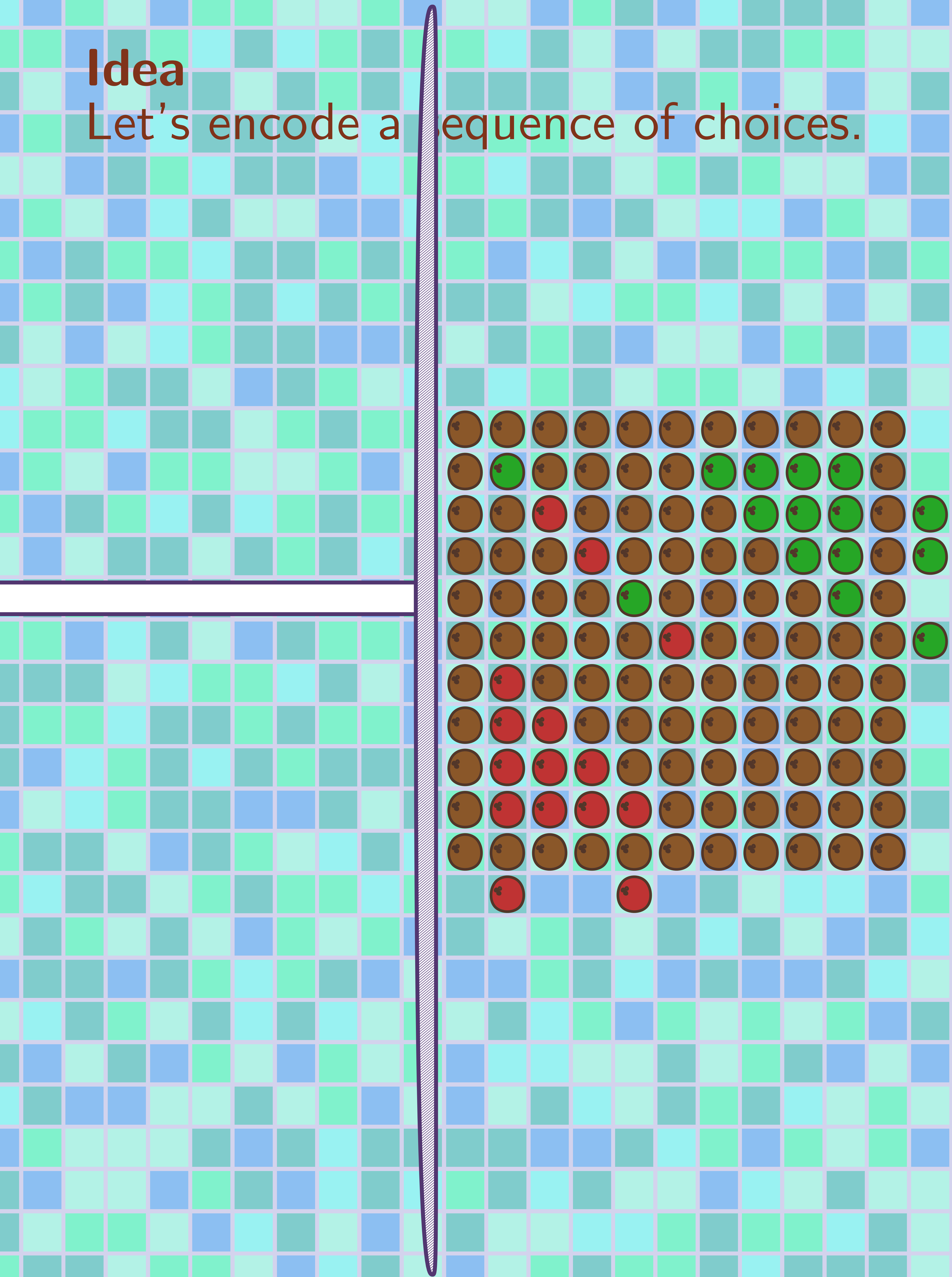
Let's encode a sequence of choices.

Idea

Let's encode a sequence of choices.

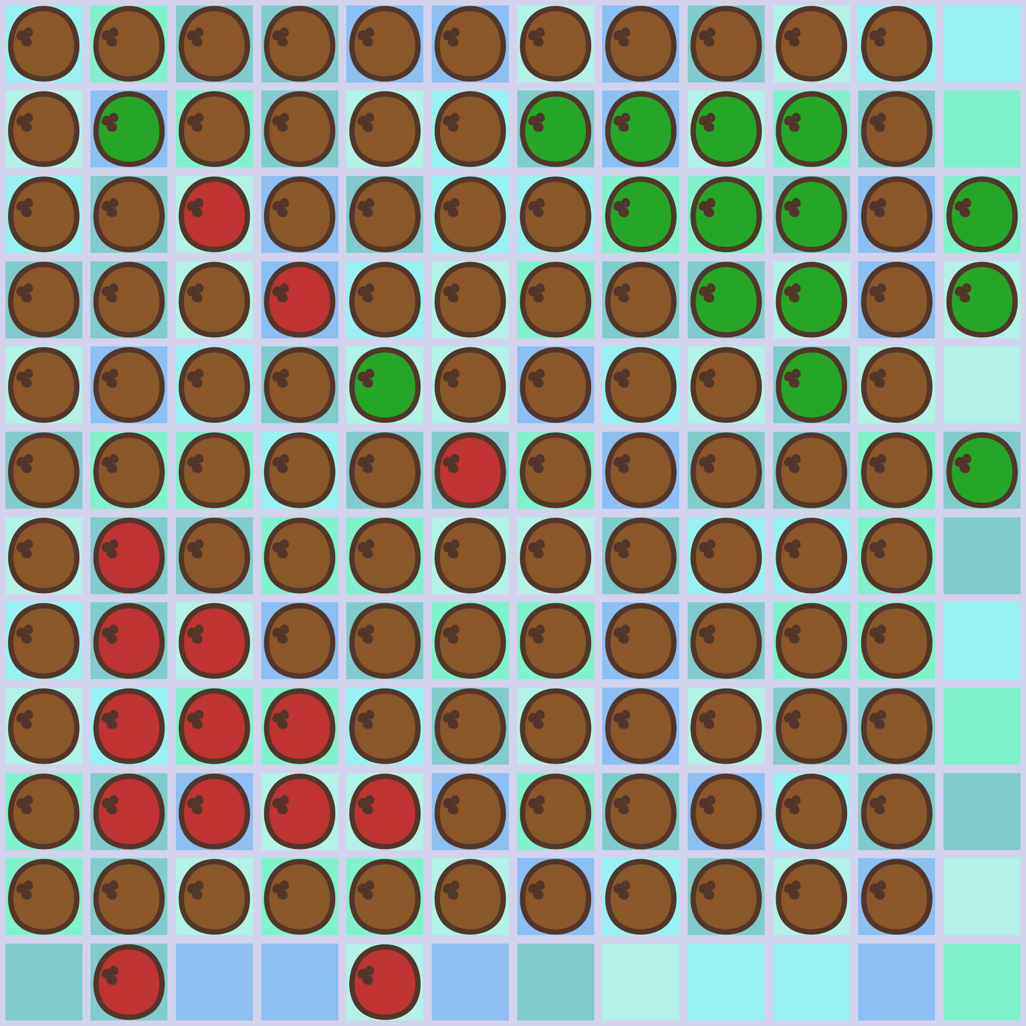
Idea

Let's encode a sequence of choices.



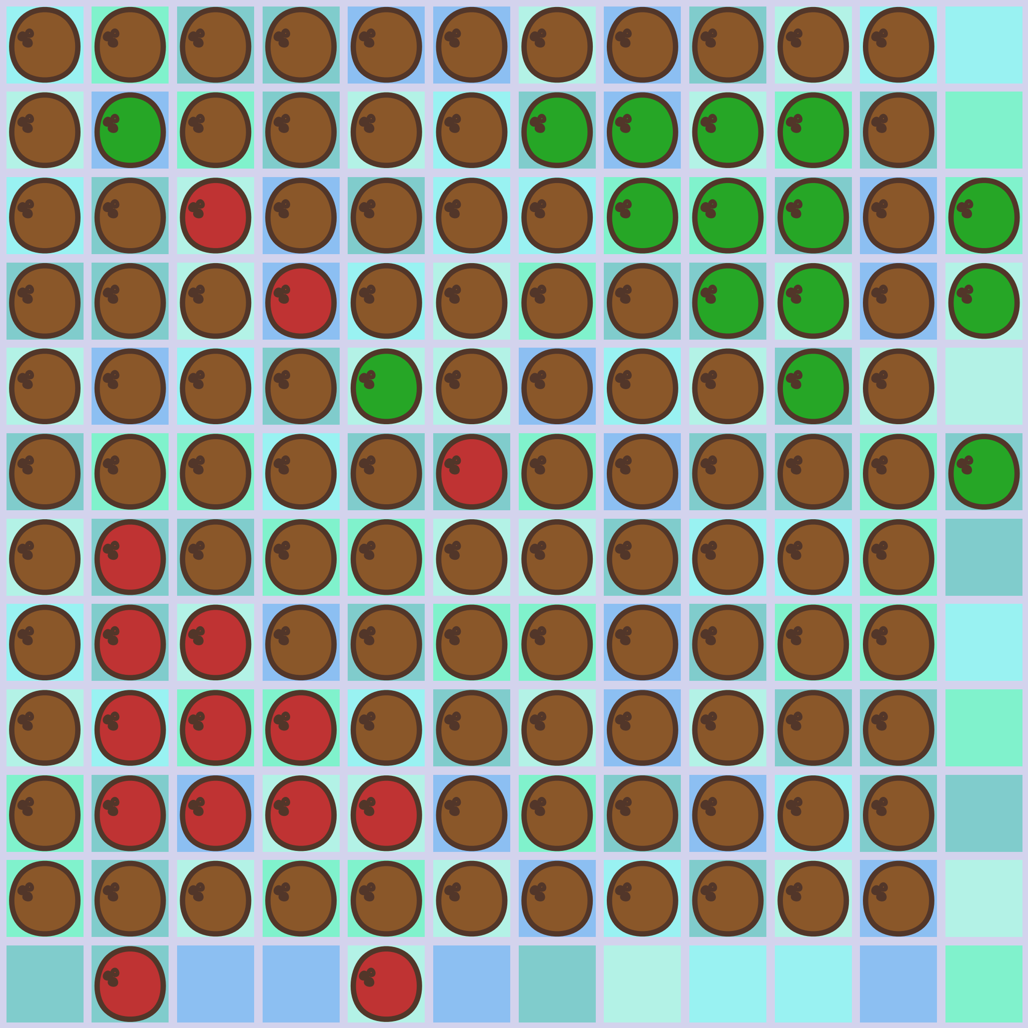
Idea
Let's encode a sequence of choices.

By choosing the order of pushes, we can make any subset spill over to the right (and bottom).



Idea
Let's encode a sequence of choices.

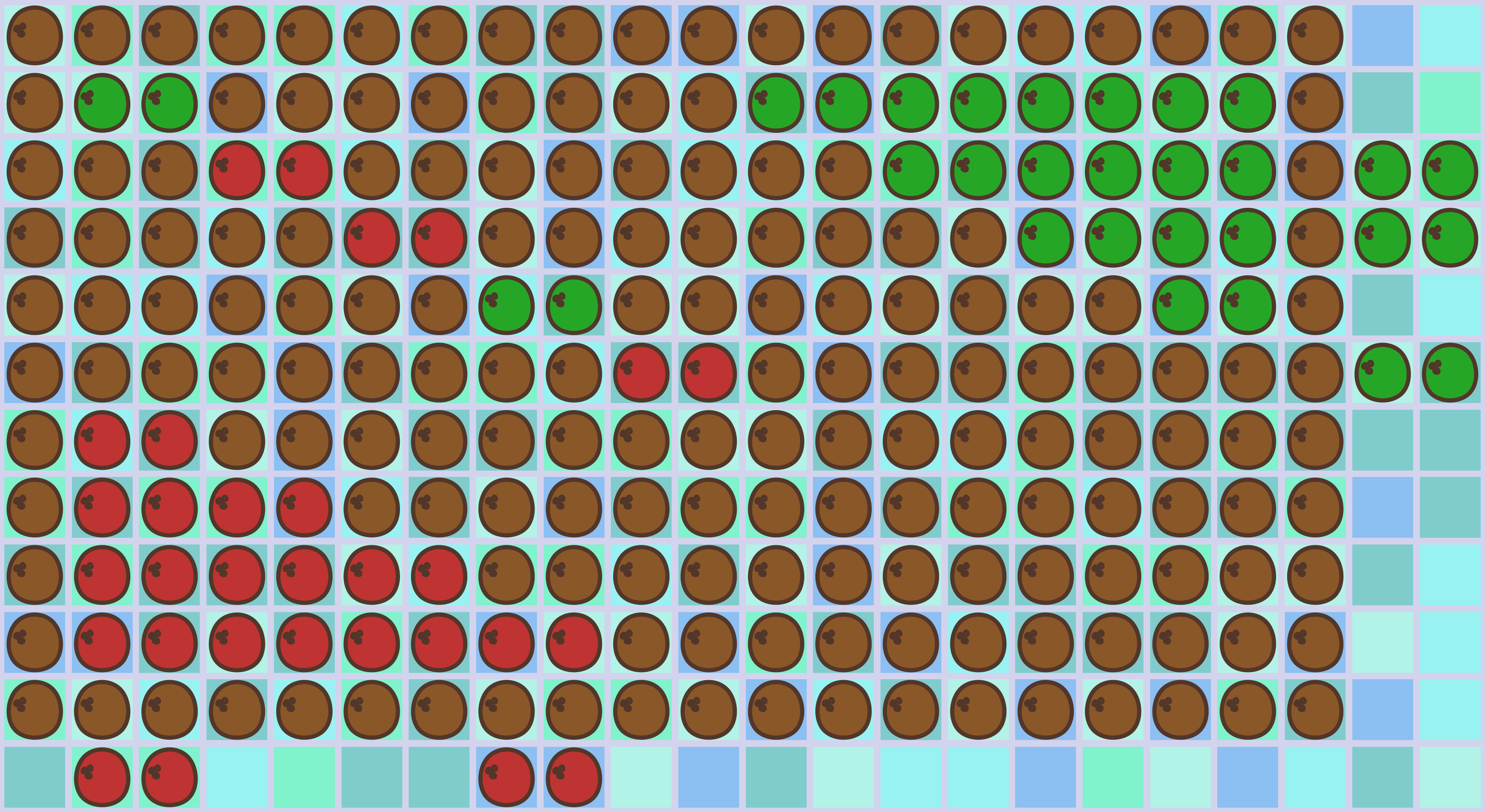
By choosing the order of pushes, we can make any subset spill over to the right (and bottom).



We can also do the same for multiple coconuts per row.

Idea
Let's encode a sequence of choices.

By choosing the order of pushes, we can make any subset spill over to the right (and bottom).



We can also do the same for multiple coconuts per row.

Now, we can reduce from Exact Hitting Set.

Now, we can reduce from Exact Hitting Set.

Given: n elements and m subsets of these elements.

Now, we can reduce from Exact Hitting Set.

Given: n elements and m subsets of these elements.

Goal: choose k elements, exactly 1 from each subset.

Now, we can reduce from Exact Hitting Set.

Given: n elements and m subsets of these elements.

Goal: choose k elements, exactly 1 from each subset.

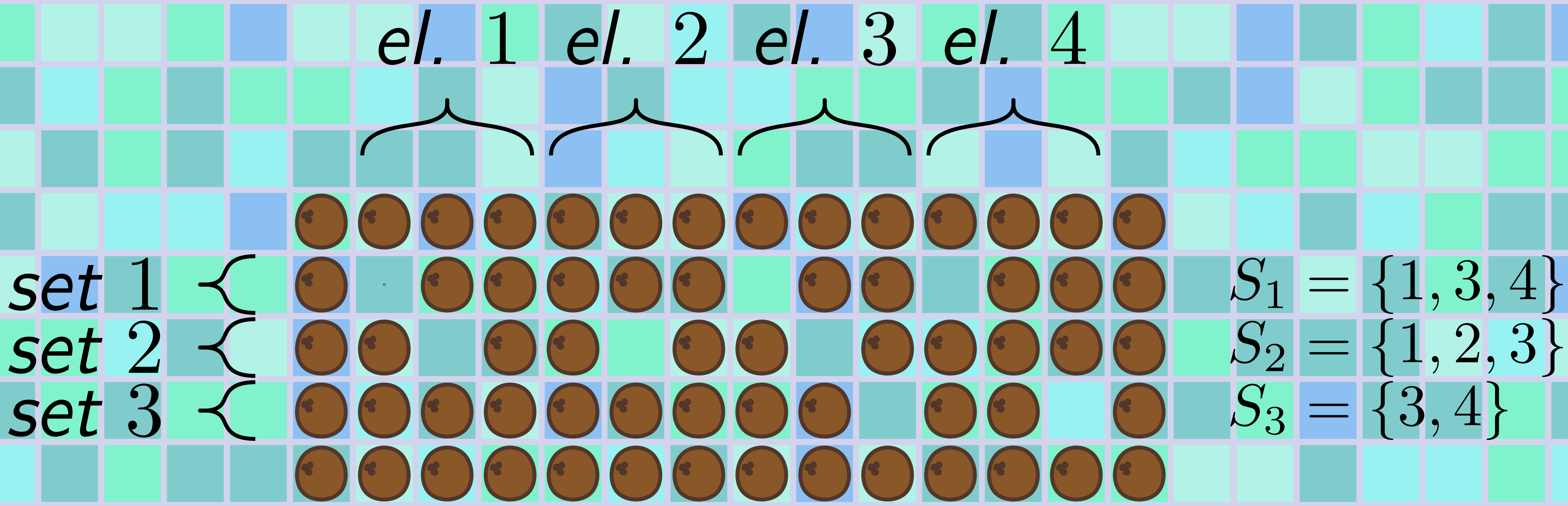
We encode the instance by making holes in a $mn \times m$ rectangle of coconuts.

Now, we can reduce from EXACT HITTING SET.

Given: n elements and m subsets of these elements.

Goal: choose k elements, exactly 1 from each subset.

We encode the instance by making holes in a $mn \times m$ rectangle of coconuts.

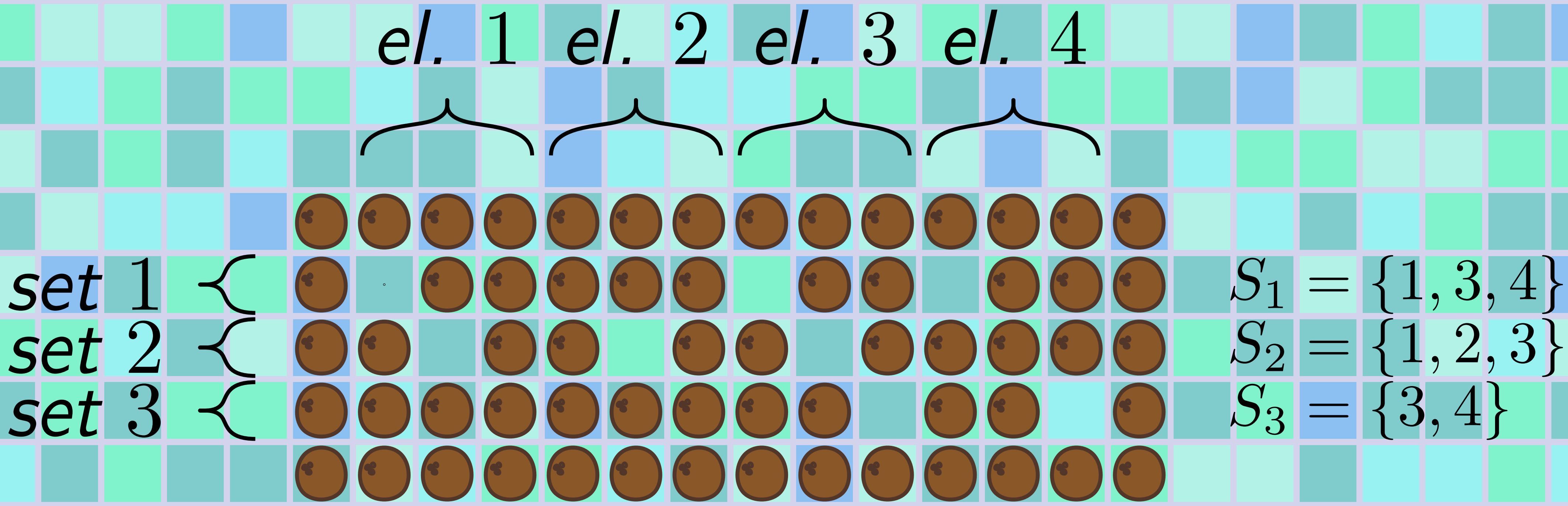


Now, we can reduce from EXACT HITTING SET.

Given: n elements and m subsets of these elements.

Goal: choose k elements, exactly 1 from each subset.

We encode the instance by making holes in a $mn \times m$ rectangle of coconuts.



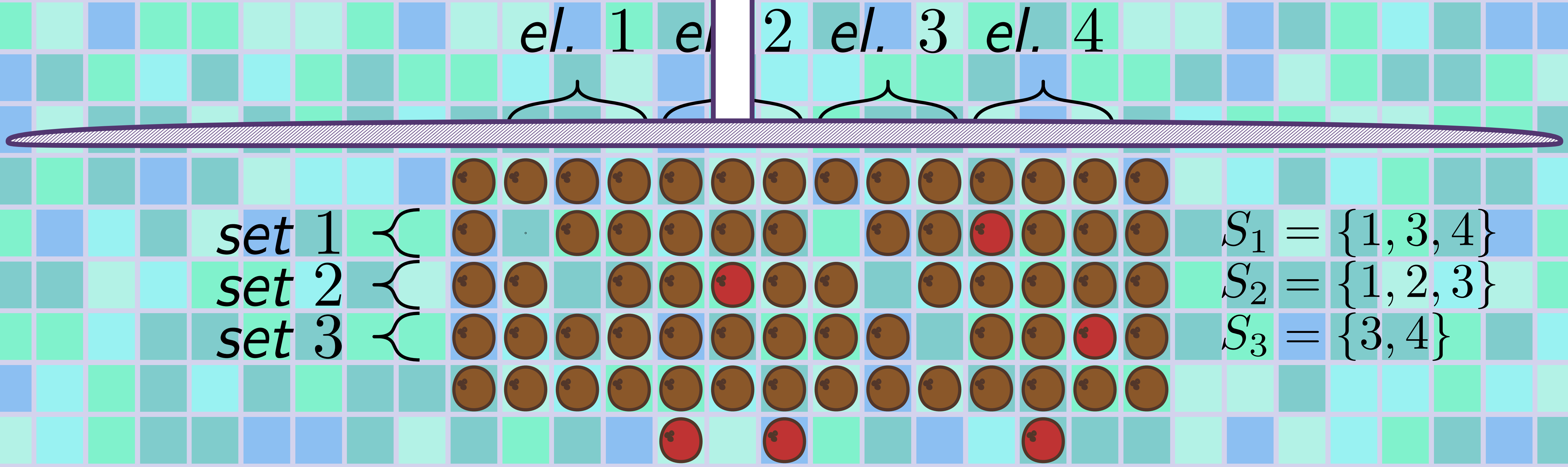
Now suppose we have coconuts coming from above in the columns of k of those elements.

Now, we can reduce from Exact Hitting Set.

Given: n elements and m subsets of these elements.

Goal: choose k elements, exactly 1 from each subset.

We encode the instance by making holes in a $mn \times m$ rectangle of coconuts.



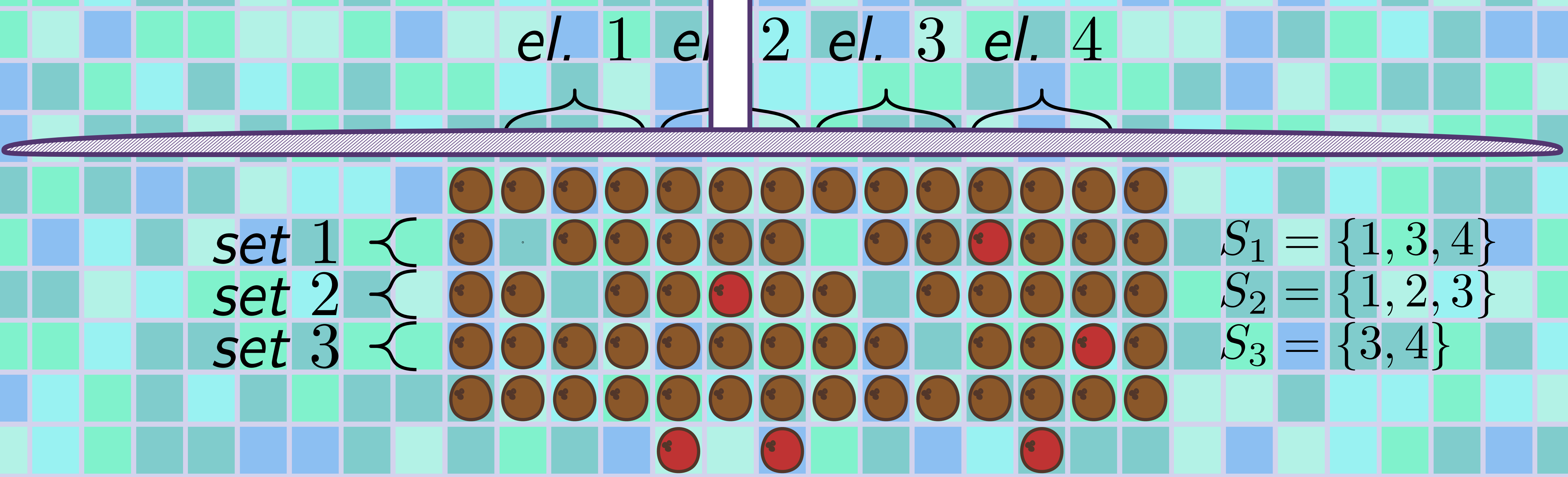
Now suppose we have coconuts coming from above in the columns of k of those elements.

Now, we can reduce from Exact Hitting Set.

Given: n elements and m subsets of these elements.

Goal: choose k elements, exactly 1 from each subset.

We encode the instance by making holes in a $mn \times m$ rectangle of coconuts.



Now suppose we have coconuts coming from above in the columns of k of those elements.

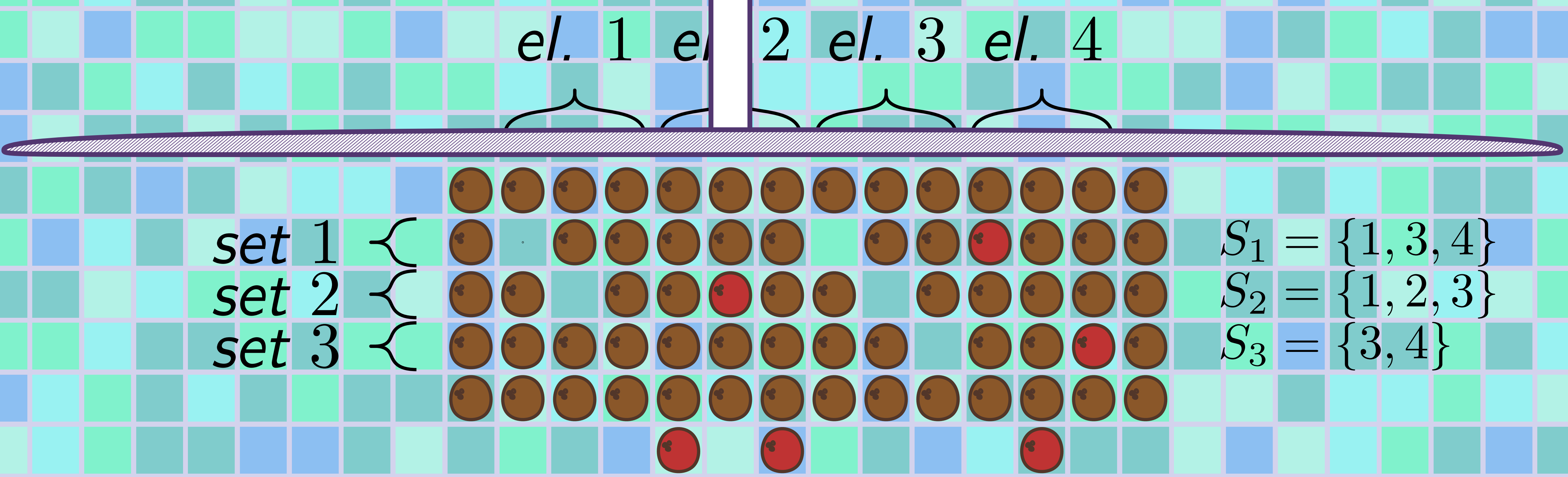
If the elements we chose were an exact hitting set, we now have exactly one red coconut per row.

Now, we can reduce from EXACT HITTING SET.

Given: n elements and m subsets of these elements.

Goal: choose k elements, exactly 1 from each subset.

We encode the instance by making holes in a $mn \times m$ rectangle of coconuts.



Now suppose we have coconuts coming from above in the columns of k of those elements.

If the elements we chose were an exact hitting set, we now have exactly one red coconut per row.

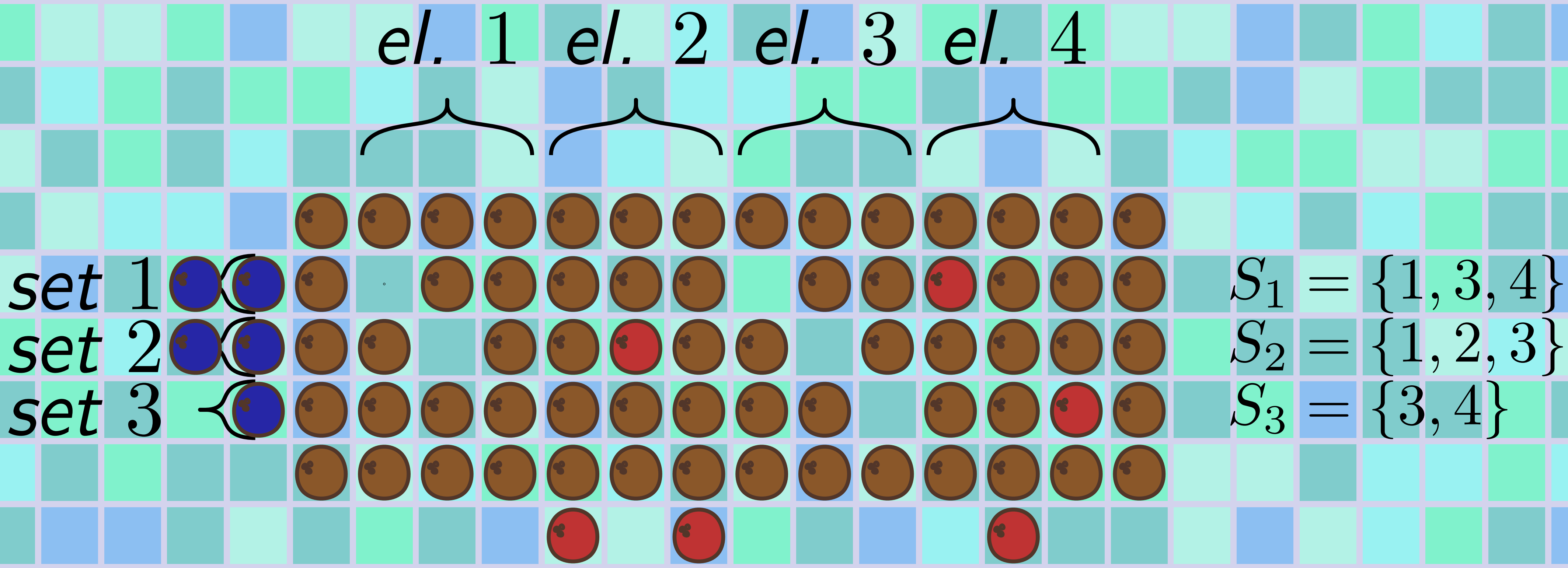
We verify this by pushing $|S_i| - 1$ coconuts into each row from the left.

Now, we can reduce from EXACT HITTING SET.

Given: n elements and m subsets of these elements.

Goal: choose k elements, exactly 1 from each subset.

We encode the instance by making holes in a $mn \times m$ rectangle of coconuts.



Now suppose we have coconuts coming from above in the columns of k of those elements.

If the elements we chose were an exact hitting set, we now have exactly one red coconut per row.

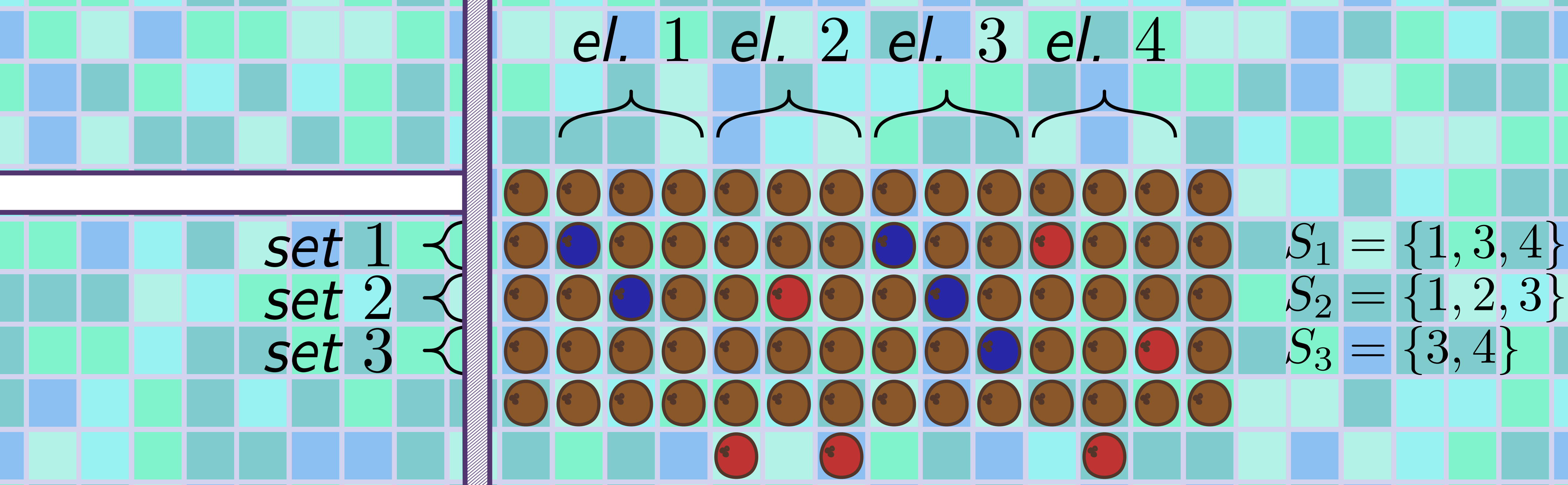
We verify this by pushing $|S_i| - 1$ coconuts into each row from the left.

Now, we can reduce from Exact Hitting Set.

Given: n elements and m subsets of these elements.

Goal: choose k elements, exactly 1 from each subset.

We encode the instance by making holes in a $mn \times m$ rectangle of coconuts.

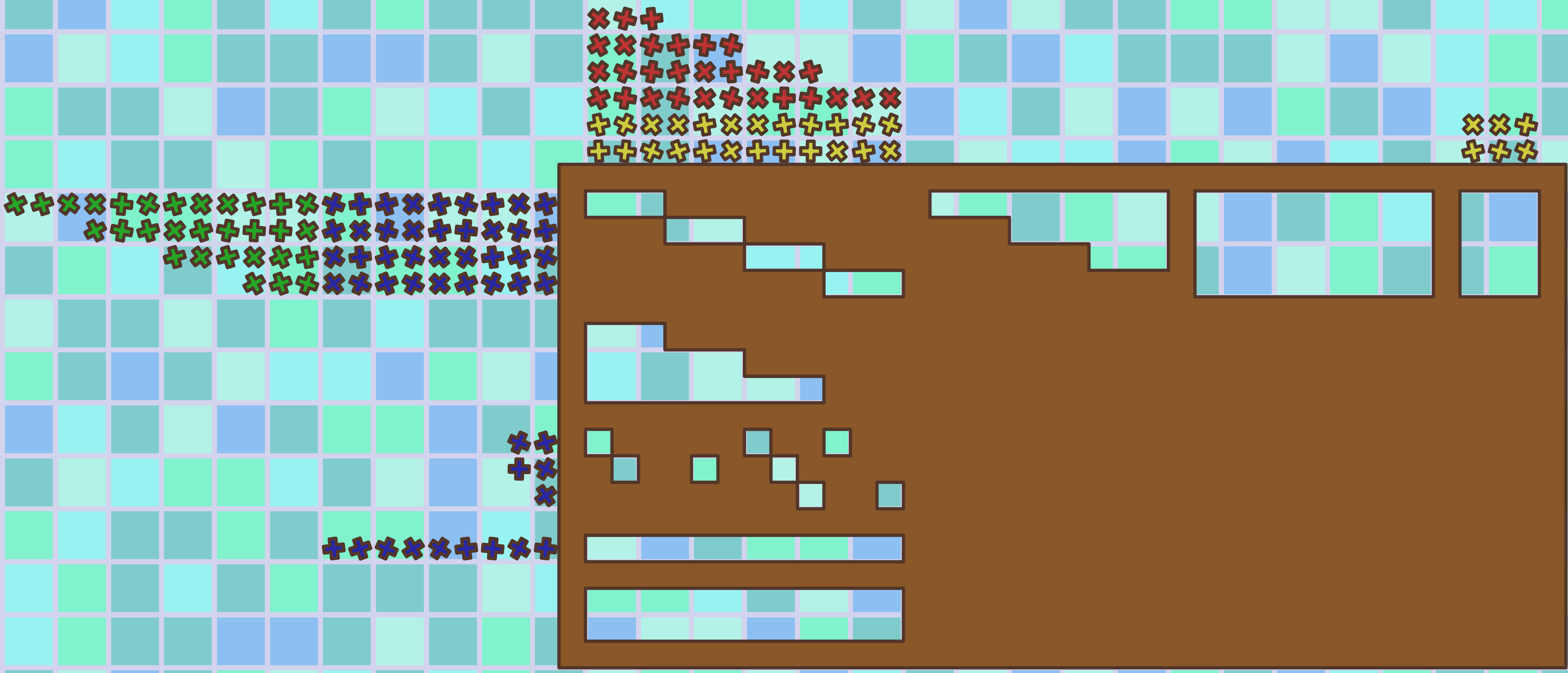


Now suppose we have coconuts coming from above in the columns of k of those elements.

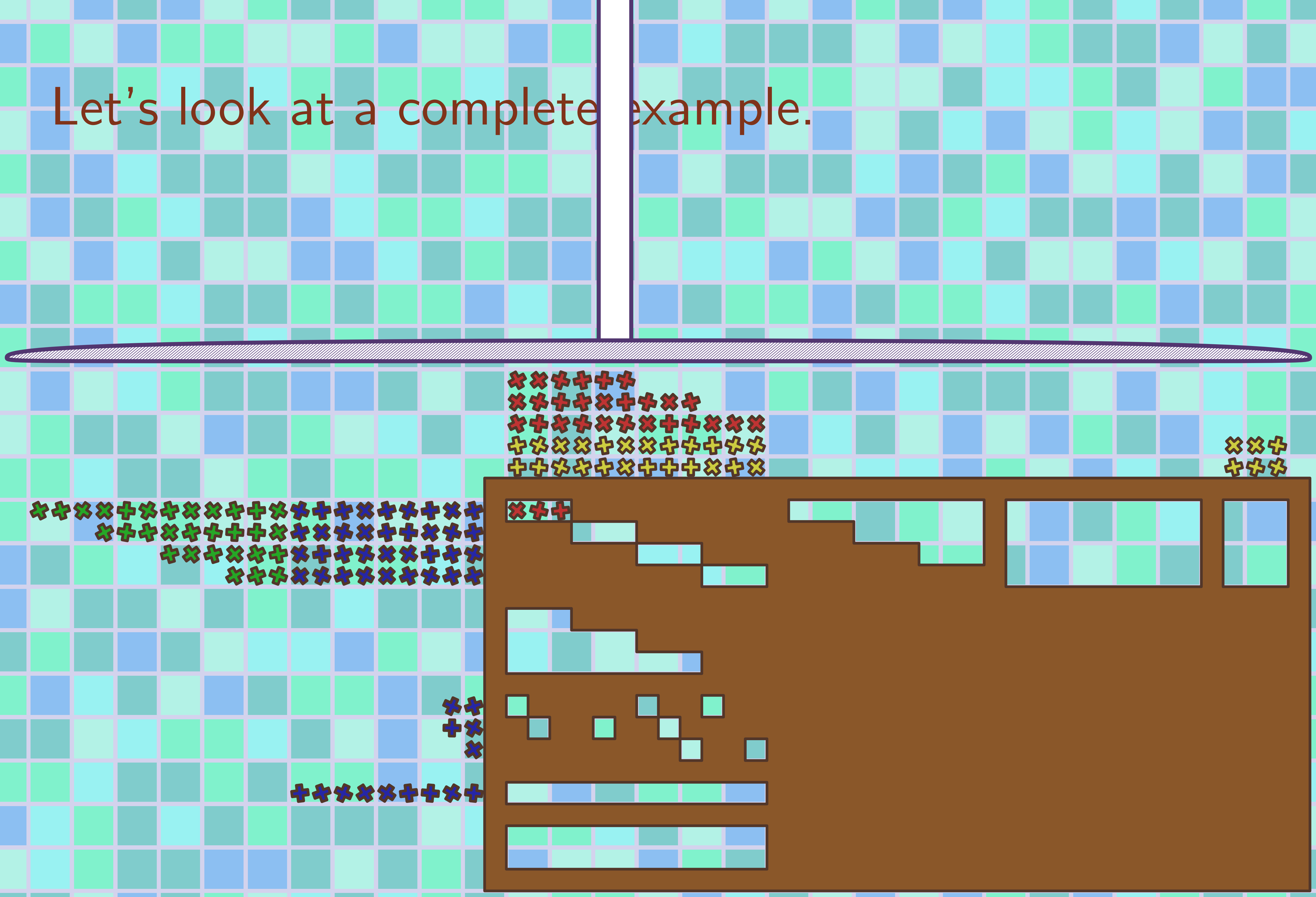
If the elements we chose were an exact hitting set, we now have exactly one red coconut per row.

We verify this by pushing $|S_i| - 1$ coconuts into each row from the left.

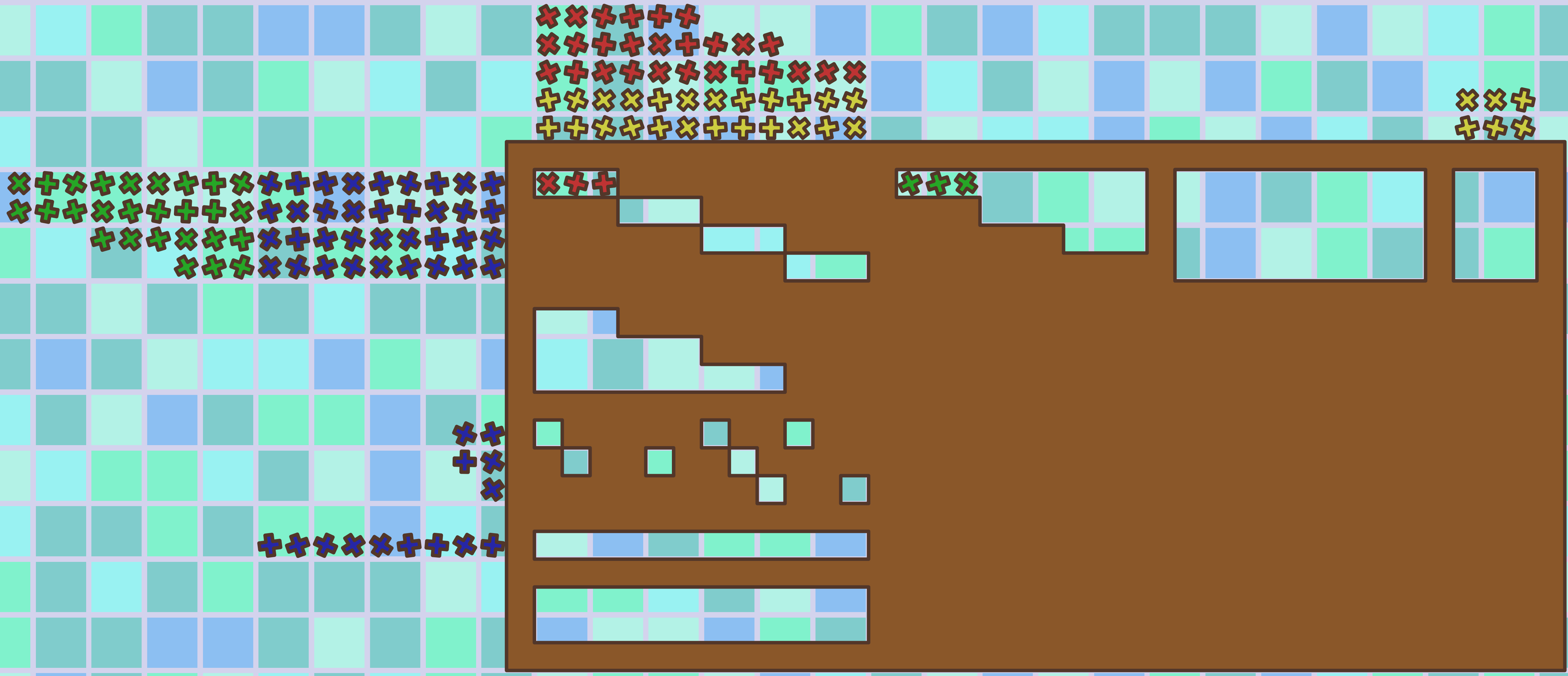
Let's look at a complete example.



Let's look at a complete example.

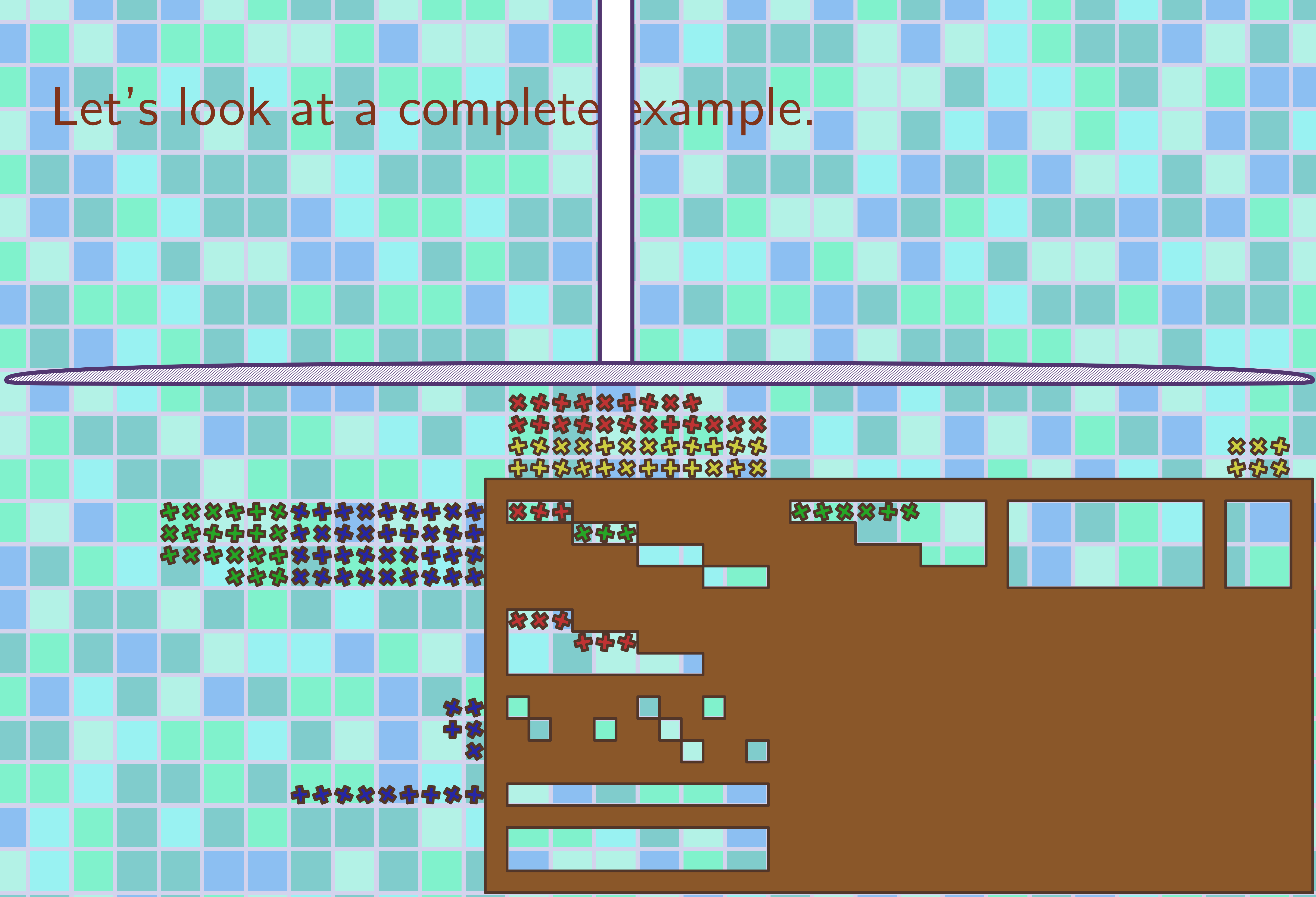


Let's look at a complete example.

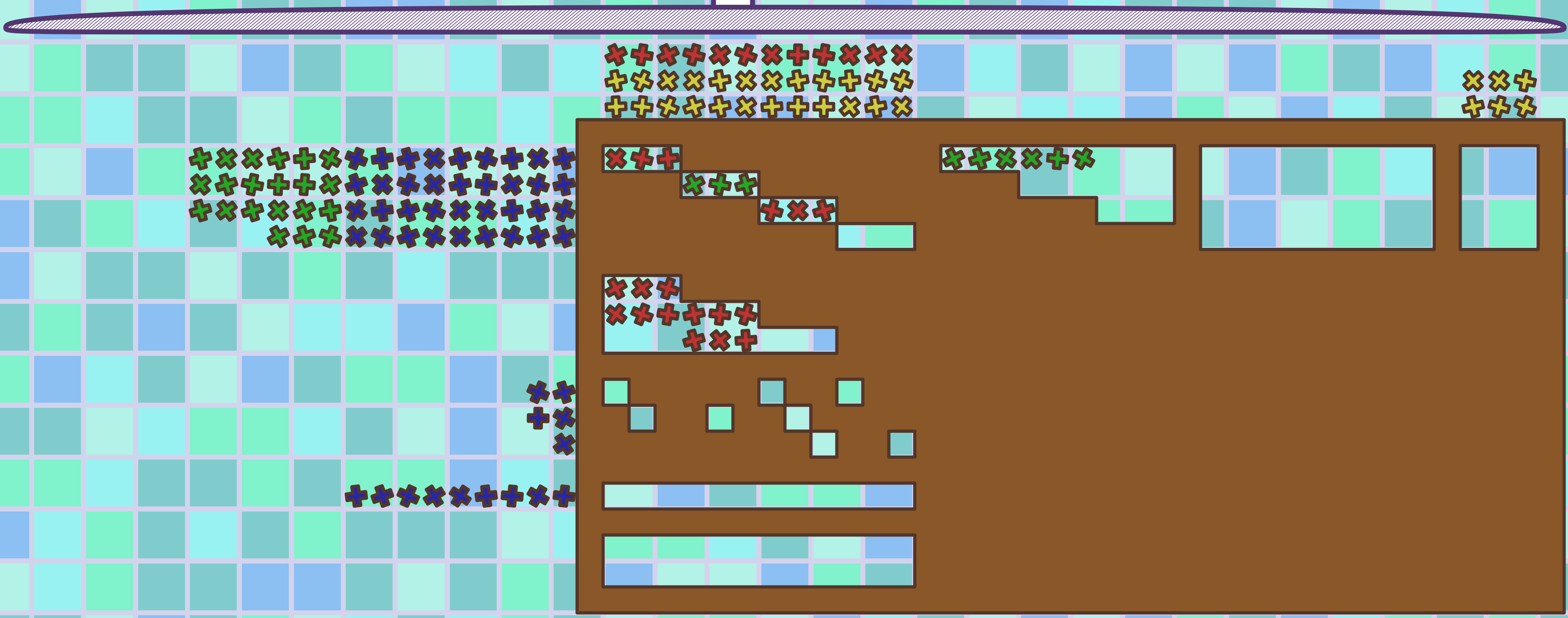


Let's look at a complete example.

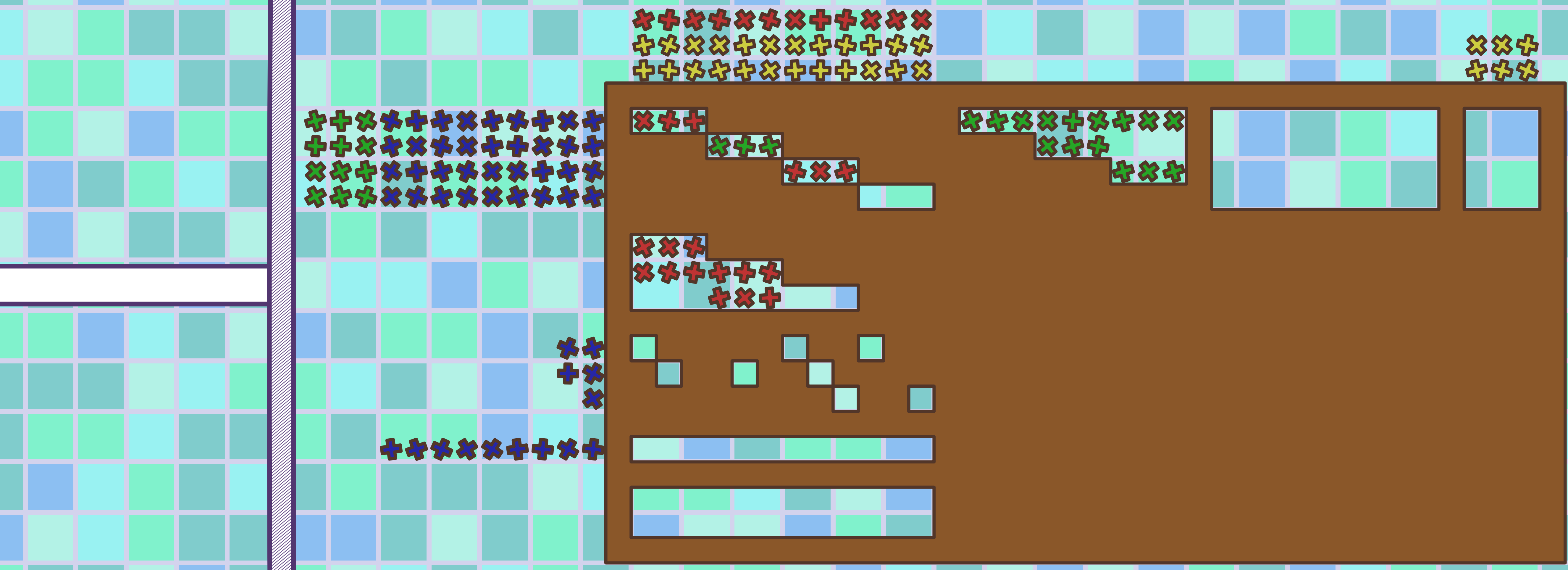
Let's look at a complete example.



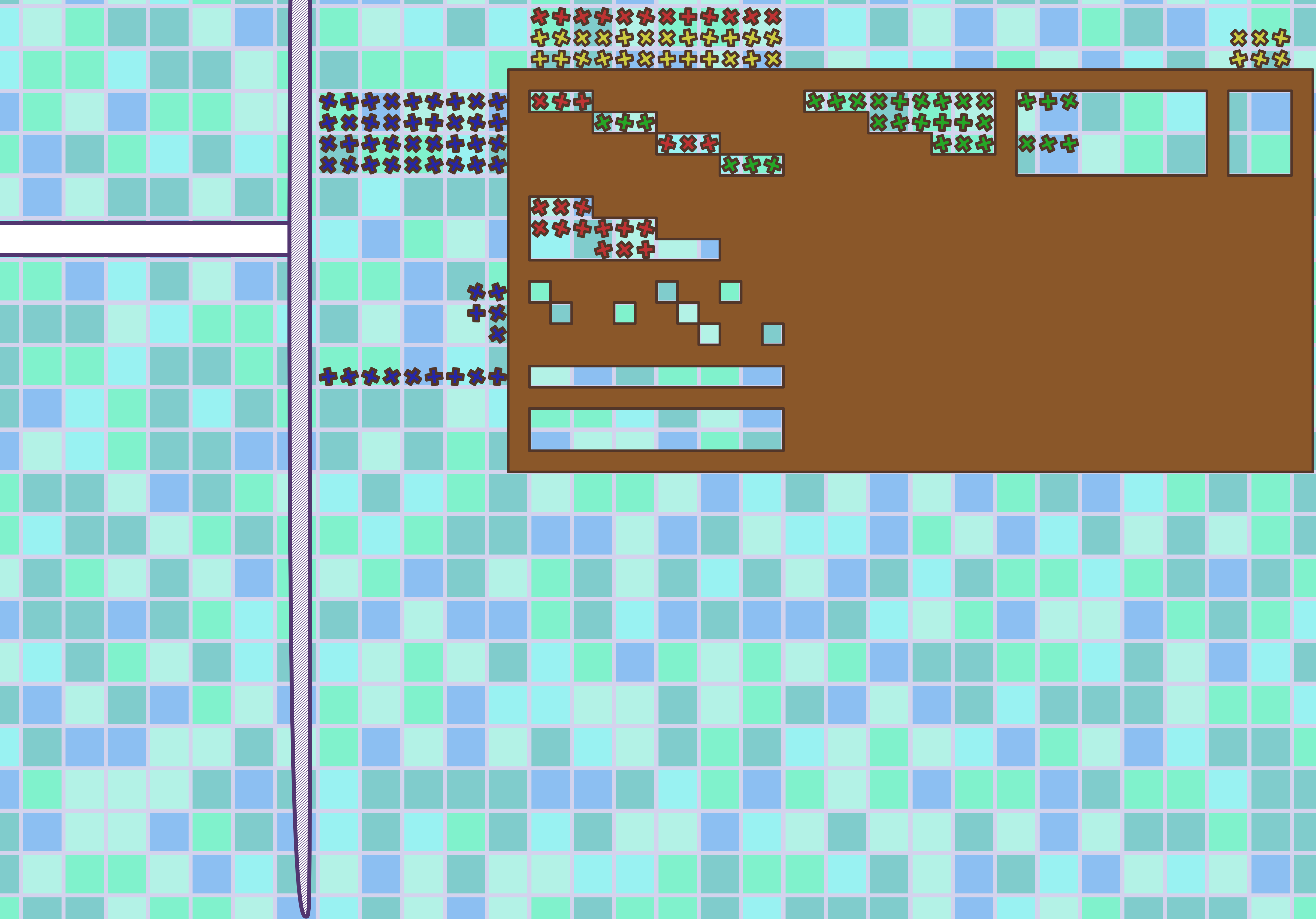
Let's look at a complete example.



Let’s look at a complete example.

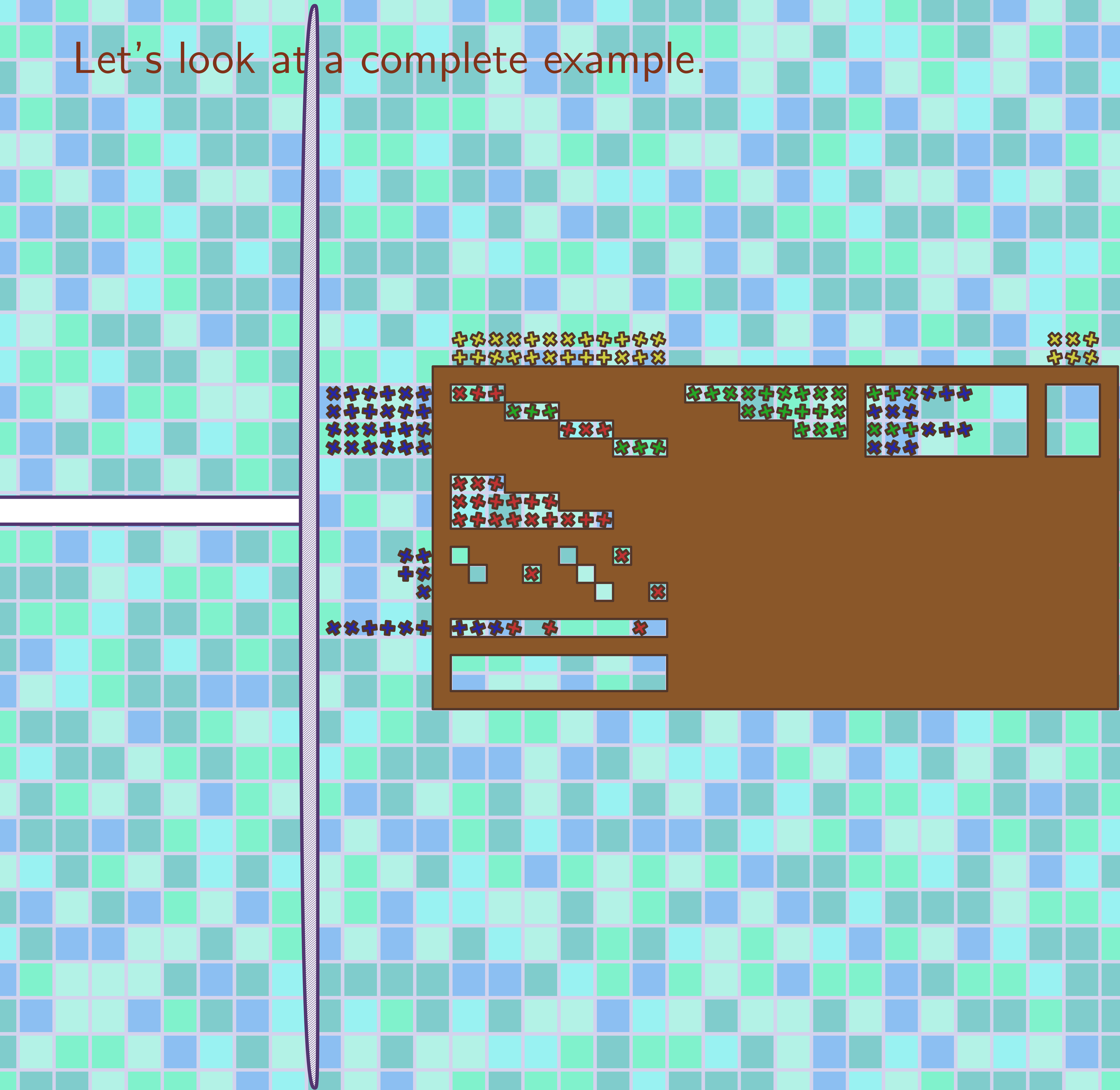


Let's look at a complete example.

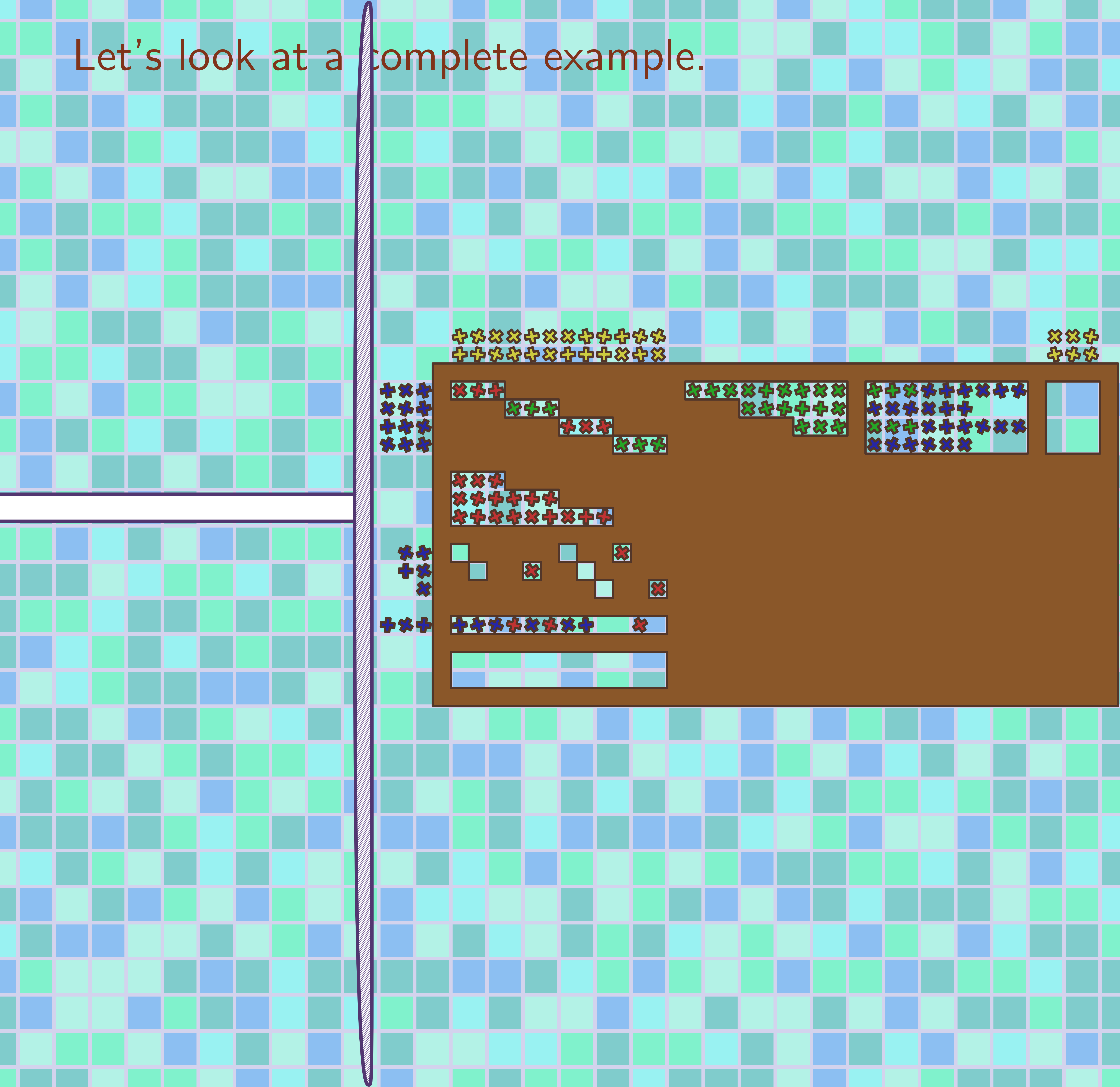


Let's look at a complete example.

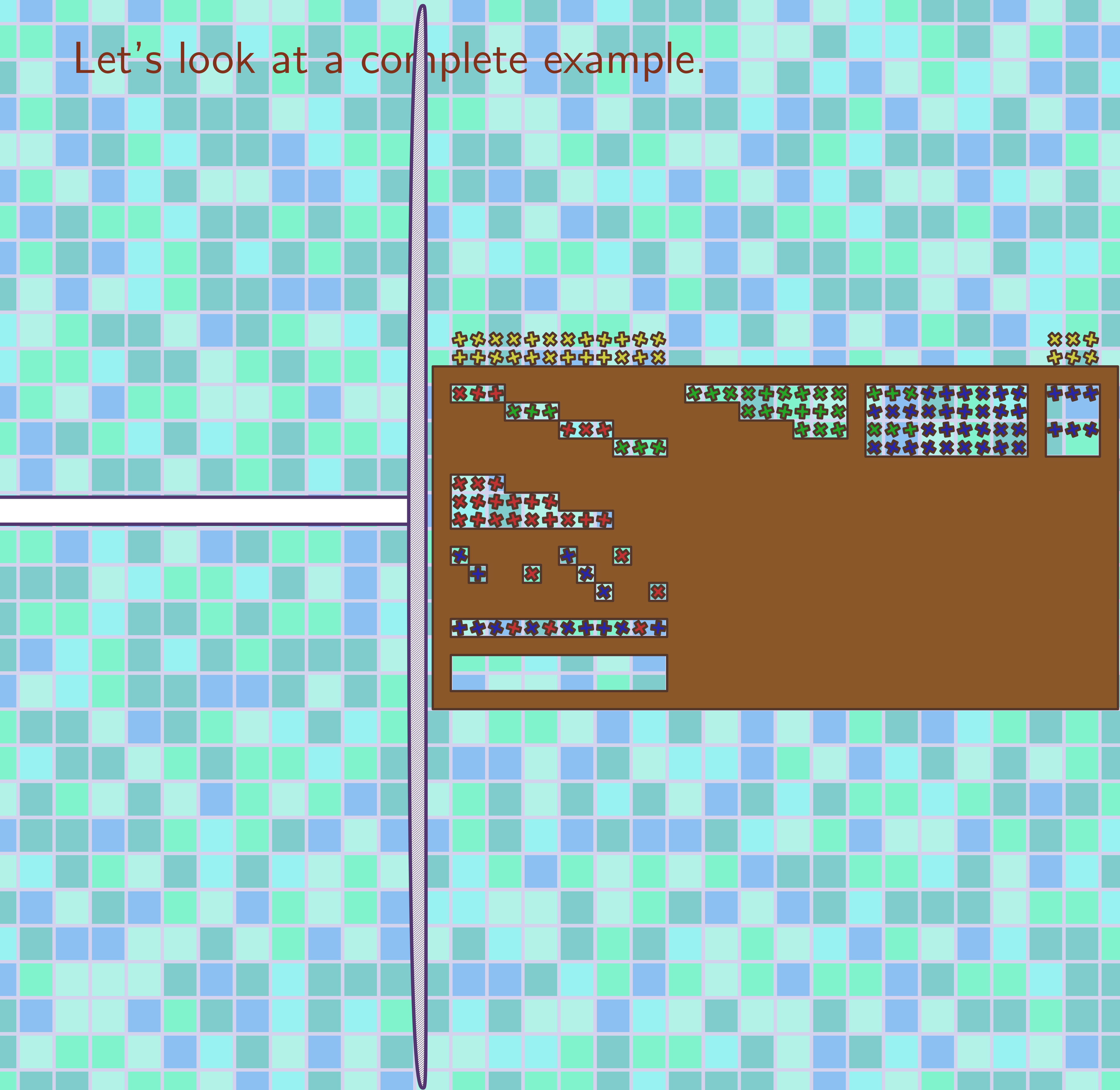
Let's look at a complete example.



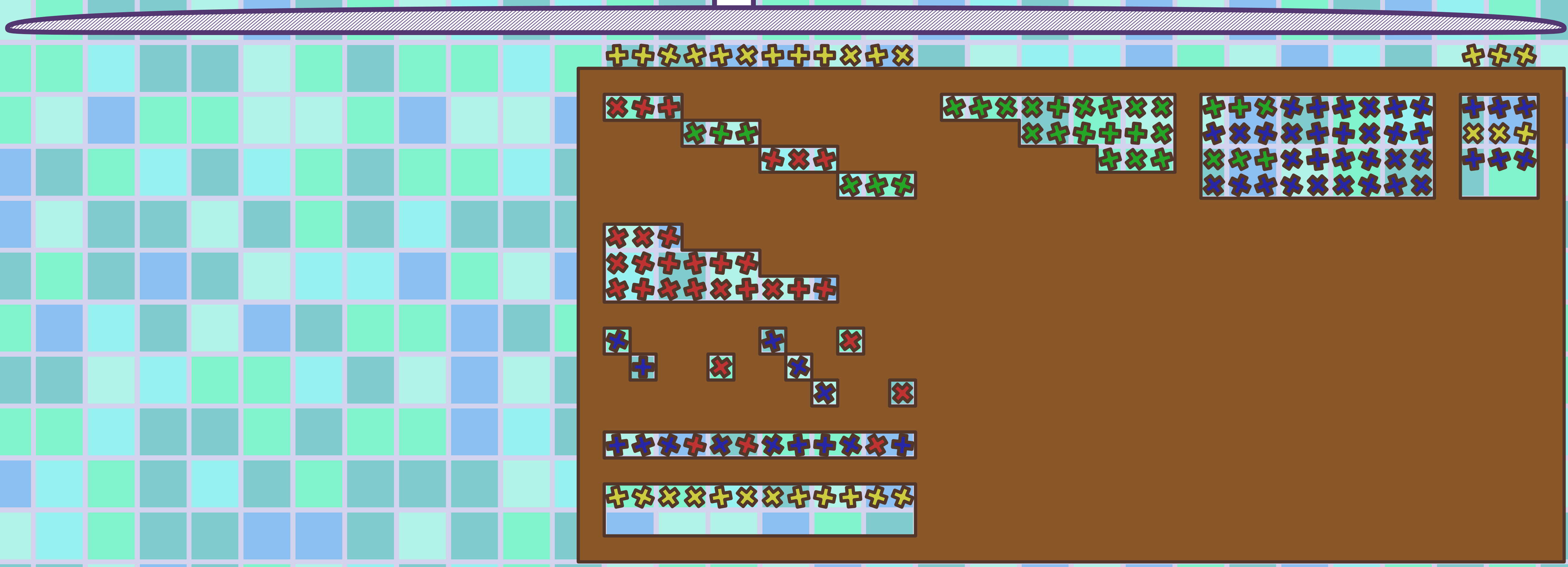
Let’s look at a complete example.



Let's look at a complete example.

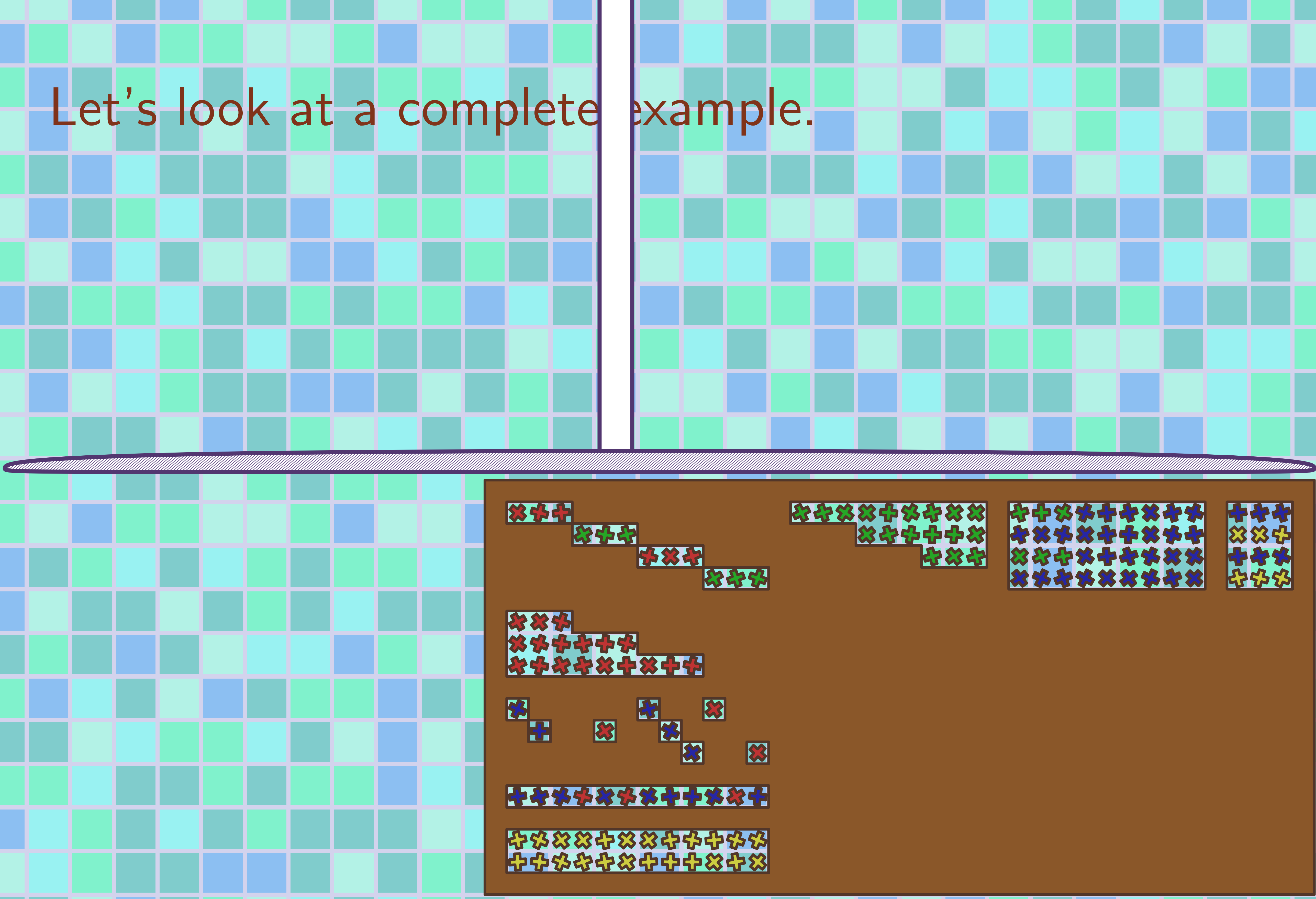


Let’s look at a complete example.



Let's look at a complete example.

Let's look at a complete example.



Quod erat demonstrandum.

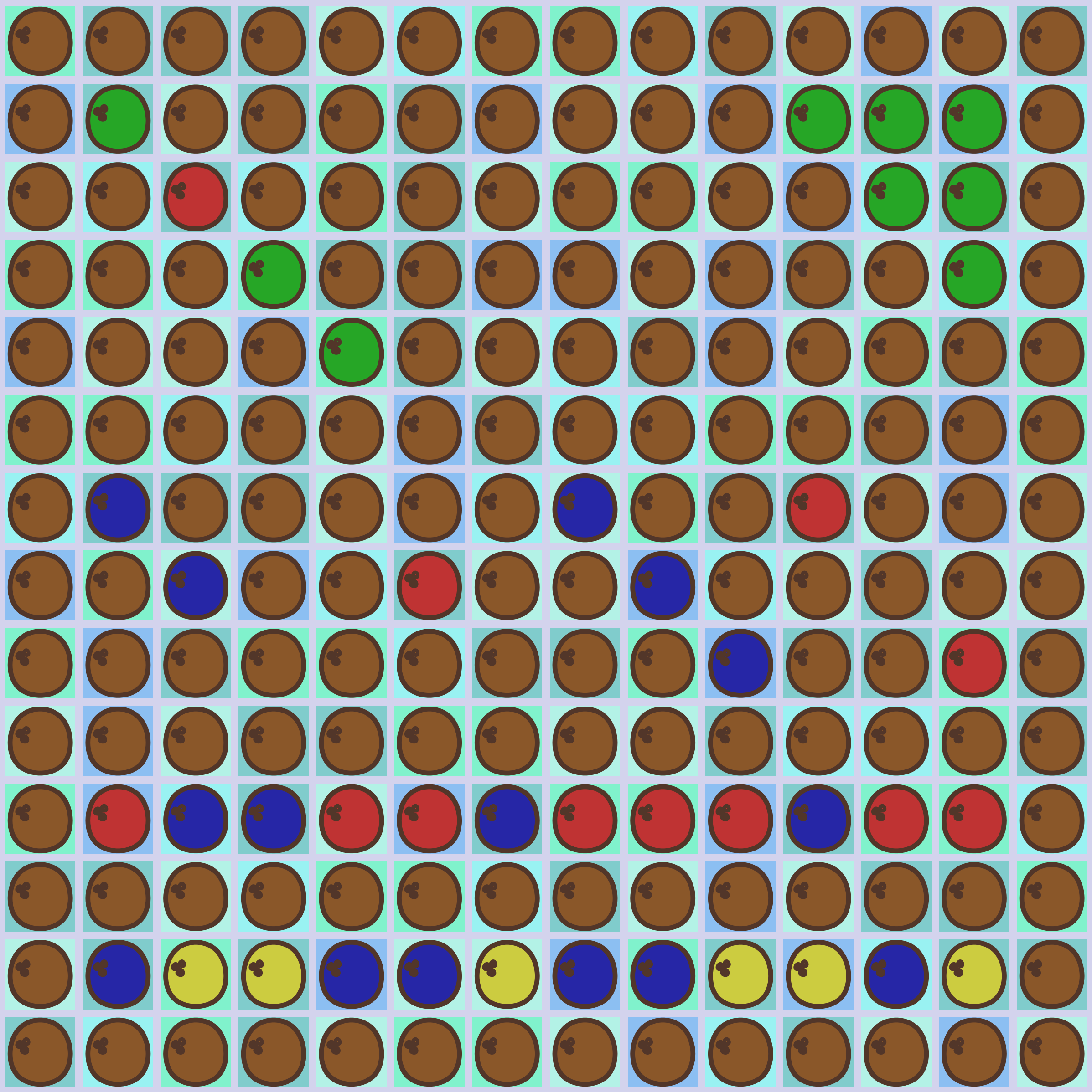
How about four-sided coconut compaction?

How about four-sided coconut compaction?

We can use the same reduction, and add one extra gadget.

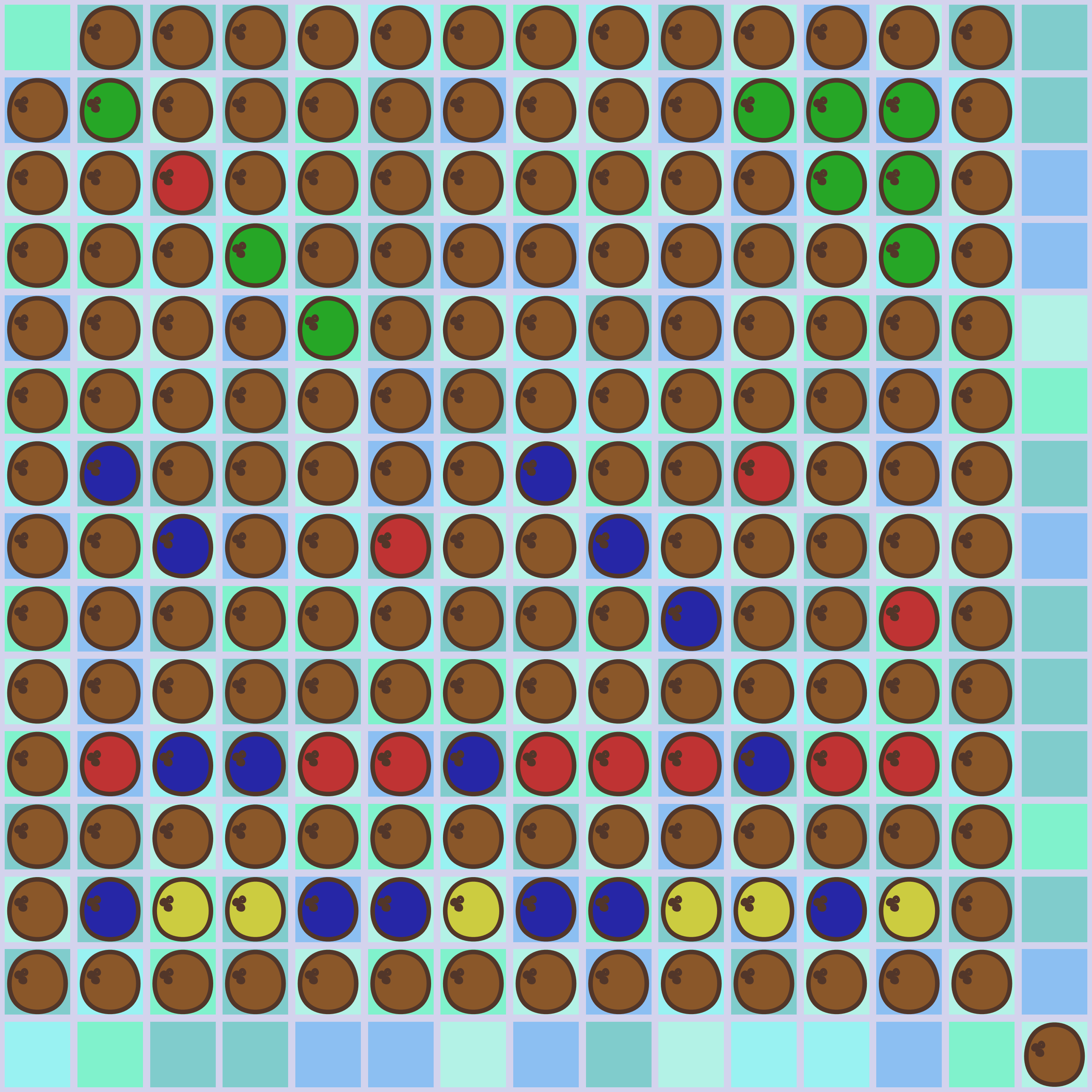
How about four-sided coconut compaction?

We can use the same reduction, and add one extra gadget.



How about four-sided coconut compaction?

We can use the same reduction, and add one extra gadget.



How about four-sided coconut compaction?

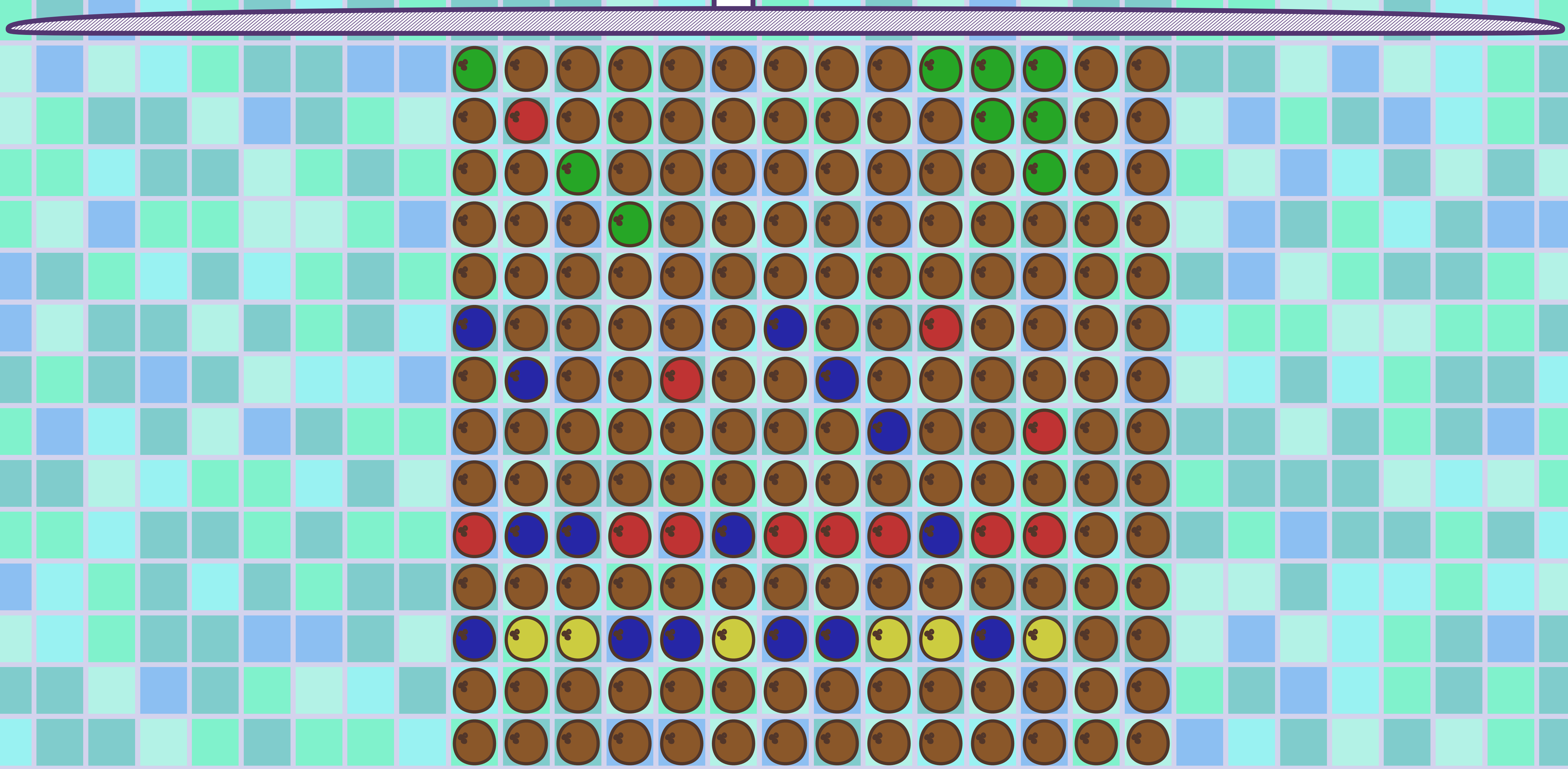
We can use the same reduction, and add one extra gadget.

How about four-sided coconut compaction?

We can use the same reduction, and add one extra gadget.

How about four-sided coconut compaction?

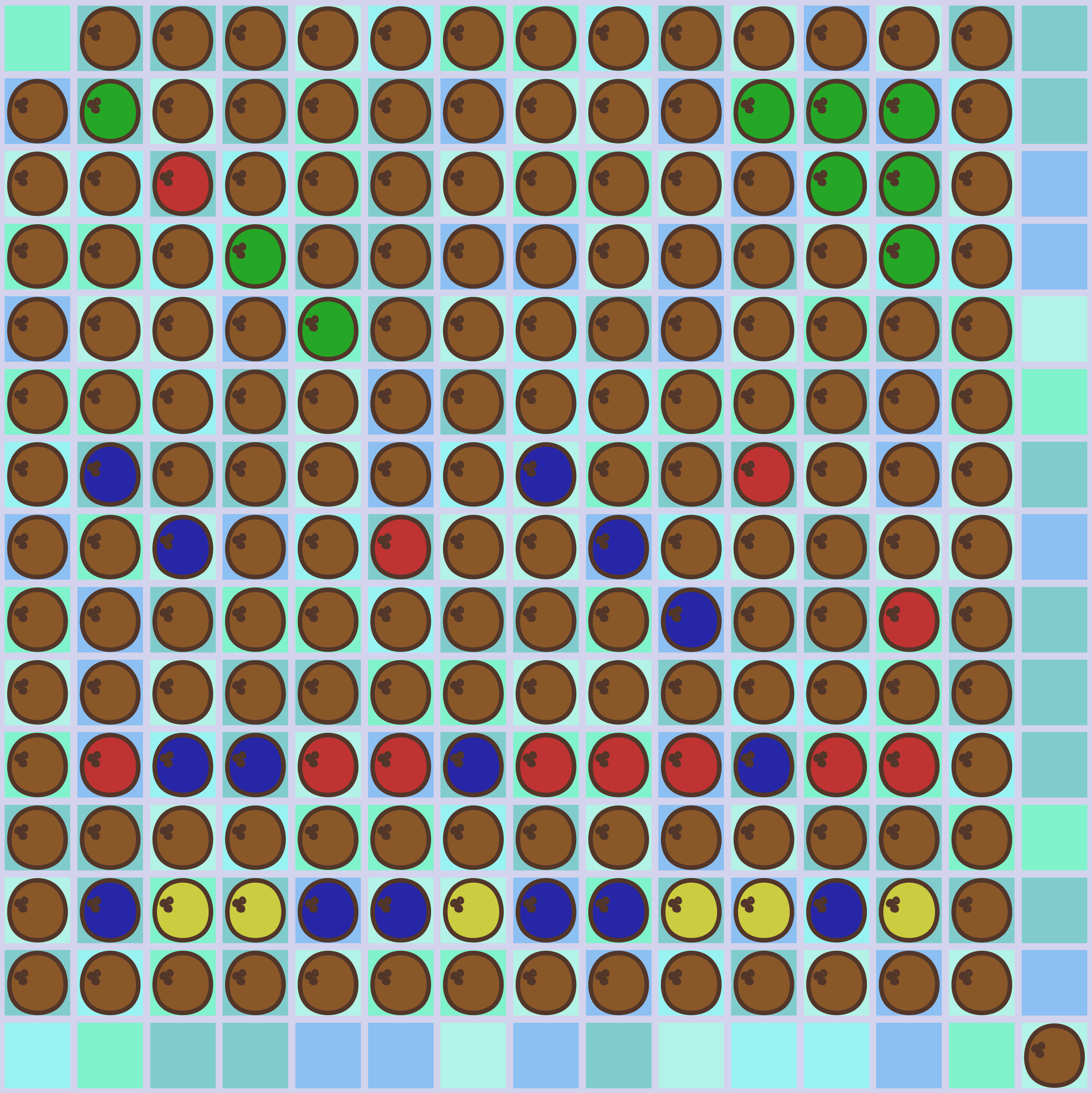
We can use the same reduction, and add one extra gadget.



We just move one brown coconut to the bottom left.

How about four-sided coconut compaction?

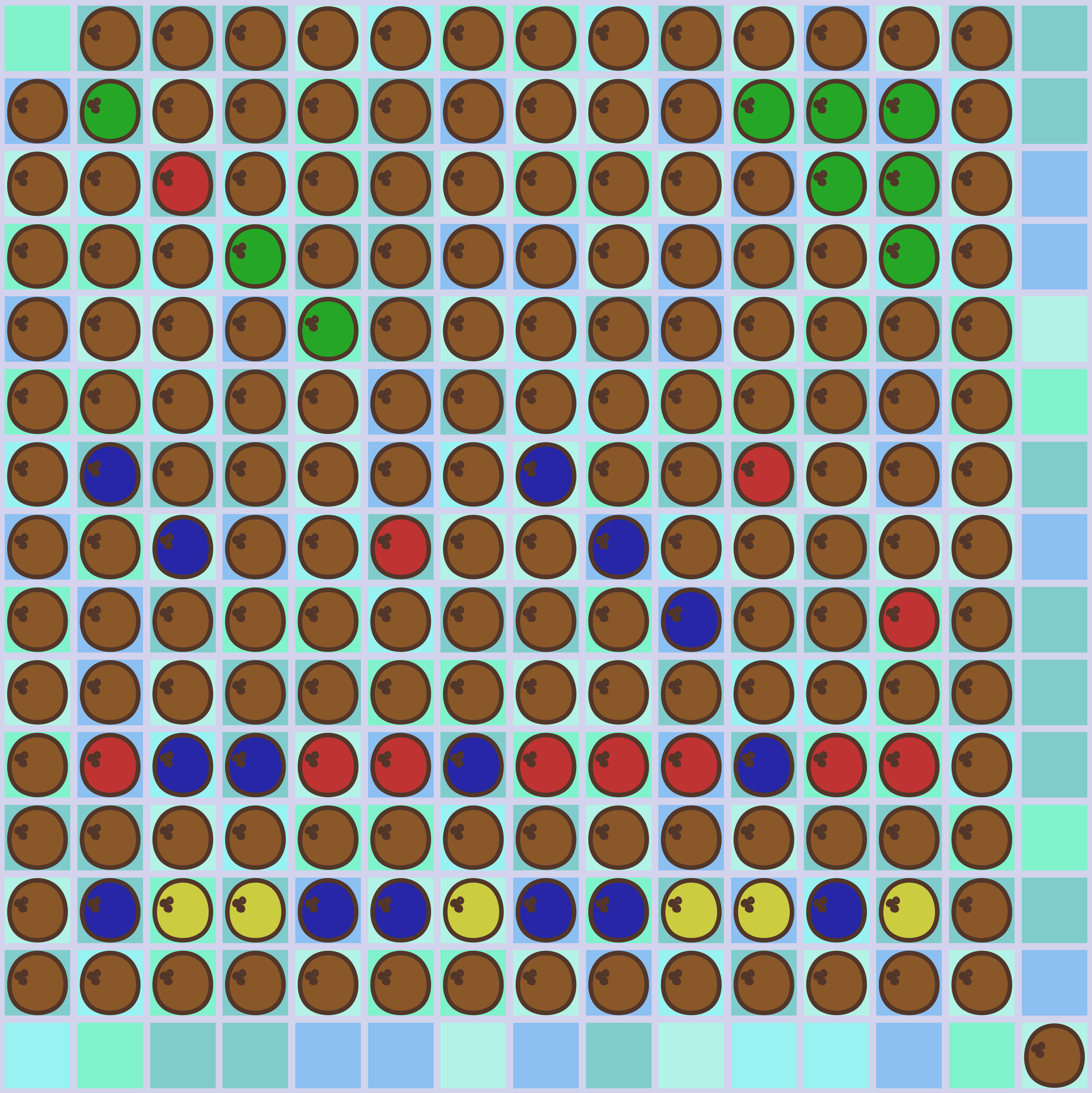
We can use the same reduction, and add one extra gadget.



We just move one brown coconut to the bottom left.

How about four-sided coconut compaction?

We can use the same reduction, and add one extra gadget.



We just move one brown coconut to the bottom left.

Observation
Pushing from the bottom or right at any time creates a row or column with too many coconuts.

COCONUT QOUESTIONS?