Hard and Easy Instances of L-Tromino Tilings [1]

Javier T. Akagi [1], Carlos F. Gaona [1], Fabricio Mendoza [1],
Manjil P. Saikia [2], Marcos Villagra [1]

[1]Universidad Nacional de Asunción
NIDTEC, Campus Universitario, San Lorenzo C.P. 2619, Paraguay

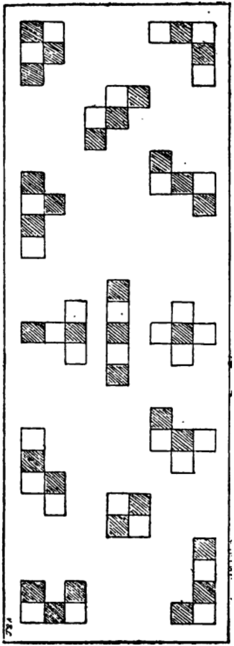
[2]Fakultät für Mathematik, Universität Wien
Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria

October 13, 2020

[1]Theoretical Computer Science, Volume 815, May 2, 2020, Pages 197-212.
In proceedings of 13th WALCOM, February 27 - March 02, 2019.
arXiv:1710.04640.

Outline

1 Introduction



Outline

Outline

Polyominoes

Polyominoes

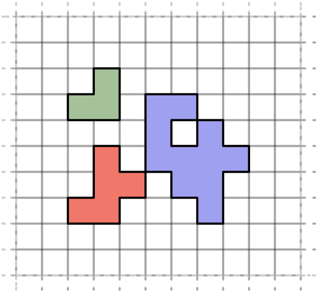
Definition

A **polyomino** is a planar figure made from one or more equal-sized squares, each joined together along an edge [S. Golomb (1953)].

Polyominoes

Definition

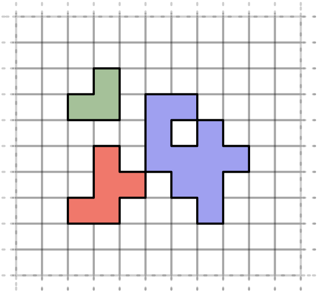
A **polyomino** is a planar figure made from one or more equal-sized squares, each joined together along an edge [S. Golomb (1953)].



Polyominoes

Definition

A **polyomino** is a planar figure made from one or more equal-sized squares, each joined together along an edge [S. Golomb (1953)].

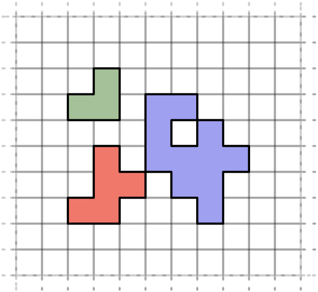


- Every cell (square) is fixed in a square lattice.

Polyominoes

Definition

A **polyomino** is a planar figure made from one or more equal-sized squares, each joined together along an edge [S. Golomb (1953)].



- Every cell (square) is fixed in a square lattice.
- Two cell are adjacent if the Manhattan distance is 1.

Outline

L-Tromino Tiling Problem

L-Tromino Tiling Problem

Definition

L-Tromino Tiling Problem

Definition

Given:

- A set of L-trominoes Σ called a **tile set**, $\Sigma = \{$ , , ,  $\}$

L-Tromino Tiling Problem

Definition

Given:

- A set of L-trominoes Σ called a **tile set**, $\Sigma = \{$ , , ,  $\}$
- and a polyomino R called **region**.

L-Tromino Tiling Problem

Definition

Given:

- A set of L-trominoes Σ called a **tile set**, $\Sigma = \{$ , , ,  $\}$
- and a polyomino R called **region**.

Goal: Place tiles from Σ to fill the region R covering every cell without overflowing the perimeter of R and without overlapping between the tiles.

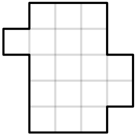
L-Tromino Tiling Problem

Definition

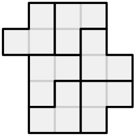
Given:

- A set of L-trominoes Σ called a **tile set**, $\Sigma = \{$ , , ,  $\}$
- and a polyomino R called **region**.

Goal: Place tiles from Σ to fill the region R covering every cell without overflowing the perimeter of R and without overlapping between the tiles.



(a) A region R



(b) A tiling of region R

L-Tromino Tiling Problem (cont'd)

L-TROMINO tiling problem

L-Tromino Tiling Problem (cont'd)

L-Tromino tiling problem

INPUT: A region R.

L-Tromino Tiling Problem (cont'd)

L-TROMINO tiling problem

INPUT: A region R.
OUTPUT: "Yes" if R has a cover and "no" otherwise.

L-Tromino Tiling Problem (cont'd)

L-Tromino tiling problem

INPUT:	A region R.
OUTPUT:	"Yes" if R has a cover and "no" otherwise.

C. Moore and J. M. Robson (2000) proved that deciding the existence of a L-Tromino tiling in a given region is **NP-complete** with a reduction from Cubic Planar Monotone 1-in-3 SAT.

L-Tromino Tiling Problem (cont'd)

L-Tromino tiling problem

INPUT: A region R.
OUTPUT: "Yes" if R has a cover and "no" otherwise.

C. Moore and J. M. Robson (2000) proved that deciding the existence of a L-Tromino tiling in a given region is **NP-complete** with a reduction from Cubic Planar Monotone 1-in-3 SAT.

T. Horiyama, T. Ito, K. Nakatsuka, A. Suzuki and R. Uehara (2012) constructed a one-one reduction from 1-in-3 SAT.

Outline

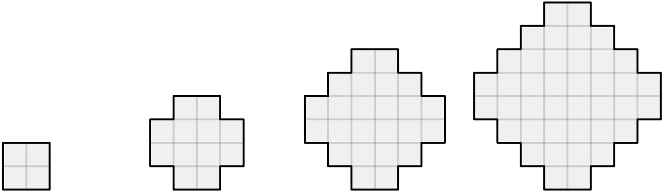
Outline

Aztec Rectangle

Aztec Rectangle

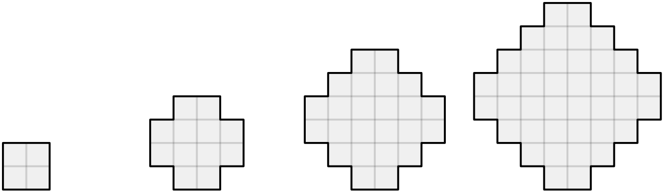
The **Aztec Diamond** $AD(n)$ is the union of all cell inside the contour $|x| + |y| = n + 1$.

The **Aztec Diamond** $AD(n)$ is the union of all cell inside the contour $|x| + |y| = n + 1$.



(a) $\mathcal{AD}_1$ (b) $\mathcal{AD}_2$ (c) $\mathcal{AD}_3$ (d) $\mathcal{AD}_4$

The **Aztec Diamond** $AD(n)$ is the union of all cell inside the contour $|x| + |y| = n + 1$.

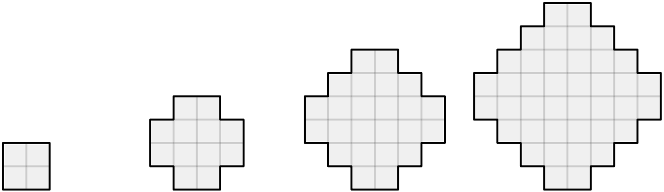


(a) $\mathcal{AD}_1$ (b) $\mathcal{AD}_2$ (c) $\mathcal{AD}_3$ (d) $\mathcal{AD}_4$

The **Aztec Rectangle** $\mathcal{AR}_{a,b}$ is a generalization of an Aztec Diamond.

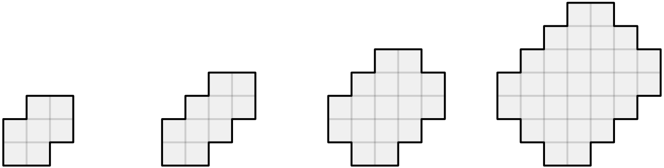
Aztec Rectangle

The **Aztec Diamond** $AD(n)$ is the union of all cell inside the contour $|x| + |y| = n + 1$.



(a) $\mathcal{AD}_1$ (b) $\mathcal{AD}_2$ (c) $\mathcal{AD}_3$ (d) $\mathcal{AD}_4$

The **Aztec Rectangle** $\mathcal{AR}_{a,b}$ is a generalization of an Aztec Diamond.

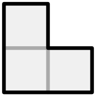


(a) $\mathcal{AR}_{1,2}$ (b) $\mathcal{AR}_{1,3}$ (c) $\mathcal{AR}_{2,3}$ (d) $\mathcal{AR}_{3,4}$

Tiling Aztec Rectangle (cont'd)

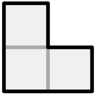
Tiling Aztec Rectangle (cont'd)

Each piece of L-tromino **covers 3 cells**.



Tiling Aztec Rectangle (cont'd)

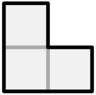
Each piece of L-tromino **covers 3 cells**.



In any L-tromino tiling, the number of covered cells is always **multiple of 3**.

Tiling Aztec Rectangle (cont'd)

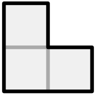
Each piece of L-tromino **covers 3 cells**.



In any L-tromino tiling, the number of covered cells is always **multiple of 3**.

The **number of cells** in an $\mathcal{AR}_{a,b}$ is given by

Each piece of L-tromino **covers 3 cells**.



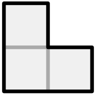
In any L-tromino tiling, the number of covered cells is always **multiple of 3**.

The **number of cells** in an $\mathcal{AR}_{a,b}$ is given by

$$|\mathcal{AR}_{a,b}| = a(b+1) + b(a+1).$$

Tiling Aztec Rectangle (cont'd)

Each piece of L-tromino **covers 3 cells**.



In any L-tromino tiling, the number of covered cells is always **multiple of 3**.

The **number of cells** in an $\mathcal{AR}_{a,b}$ is given by

$$|\mathcal{AR}_{a,b}| = a(b+1) + b(a+1).$$

Theorem

An Aztec rectangle $\mathcal{AR}_{a,b}$ has a tiling with L-trominoes
$\iff |\mathcal{AR}_{a,b}| \equiv 0 \pmod 3$
$\iff (a,b)$ *is equal to* $(3k, 3k')$ *or* $(3k-1, 3k'-1)$ *for some* $k, k' \in \mathbb{N}$.

Tiling Aztec Rectangle (cont'd)

Tiling Aztec Rectangle (cont'd)

The problem of tiling an Aztec Rectangle can be solved **recursively**.

Tiling Aztec Rectangle (cont'd)

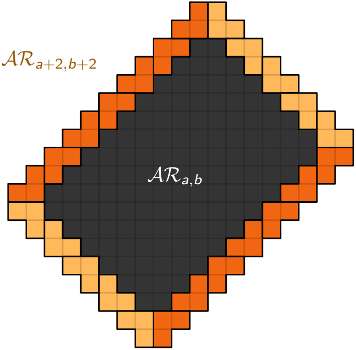
The problem of tiling an Aztec Rectangle can be solved **recursively**.

- If (a, b) equals $(3k, 3k')$, use pattern 1.
- If (a, b) equals $(3k - 1, 3k' - 1)$, use pattern 2.

Tiling Aztec Rectangle (cont'd)

The problem of tiling an Aztec Rectangle can be solved **recursively**.

- If (a, b) equals $(3k, 3k')$, use pattern 1.
- If (a, b) equals $(3k-1, 3k'-1)$, use pattern 2.

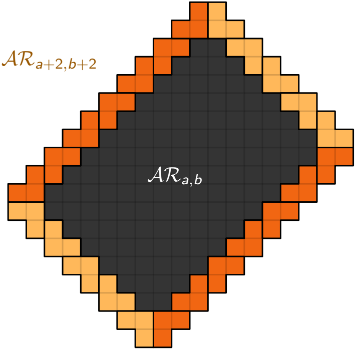


(a) Pattern 1

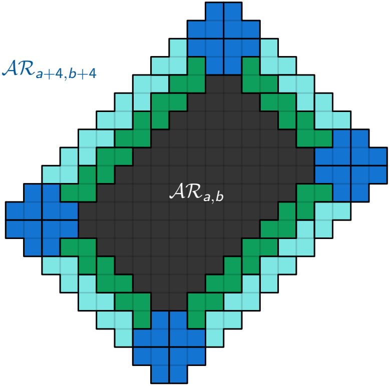
Tiling Aztec Rectangle (cont'd)

The problem of tiling an Aztec Rectangle can be solved **recursively**.

- If (a, b) equals $(3k, 3k')$, use pattern 1.
- If (a, b) equals $(3k-1, 3k'-1)$, use pattern 2.



(a) Pattern 1



(b) Pattern 2

Tiling Aztec Rectangle (cont'd)

The problem of tiling an Aztec Rectangle can be solved **recursively**.

Tiling Aztec Rectangle (cont'd)

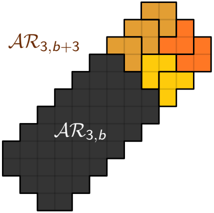
The problem of tiling an Aztec Rectangle can be solved **recursively**.

- If (a, b) equals $(3, 3k')$, use pattern 3.
- If (a, b) equals $(2, 3k' - 1)$, use pattern 4.

Tiling Aztec Rectangle (cont'd)

The problem of tiling an Aztec Rectangle can be solved **recursively**.

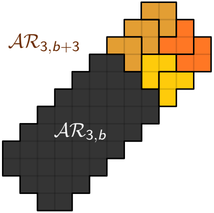
- If (a, b) equals $(3, 3k')$, use pattern 3.
- If (a, b) equals $(2, 3k' - 1)$, use pattern 4.



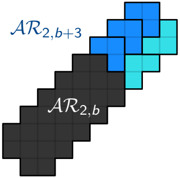
(a) Pattern 3

The problem of tiling an Aztec Rectangle can be solved **recursively**.

- If (a, b) equals $(3, 3k')$, use pattern 3.
- If (a, b) equals $(2, 3k' - 1)$, use pattern 4.



(a) Pattern 3

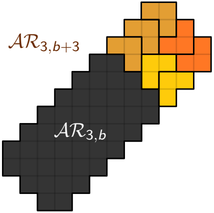


(b) Pattern 4

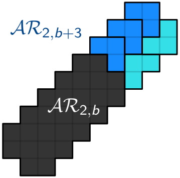
Tiling Aztec Rectangle (cont'd)

The problem of tiling an Aztec Rectangle can be solved **recursively**.

- If (a, b) equals $(3, 3k')$, use pattern 3.
- If (a, b) equals $(2, 3k' - 1)$, use pattern 4.

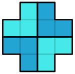


(a) Pattern 3



(b) Pattern 4

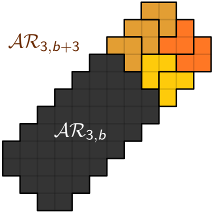
Base case: $\mathcal{AR}_{2,2}$ and $\mathcal{AR}_{3,3}$.



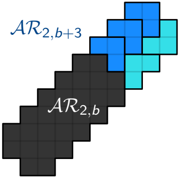
Tiling Aztec Rectangle (cont'd)

The problem of tiling an Aztec Rectangle can be solved **recursively**.

- If (a, b) equals $(3, 3k')$, use pattern 3.
- If (a, b) equals $(2, 3k' - 1)$, use pattern 4.

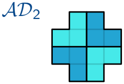


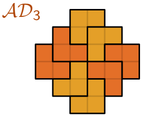
(a) Pattern 3



(b) Pattern 4

Base case: $\mathcal{AR}_{2,2}$ and $\mathcal{AR}_{3,3}$.





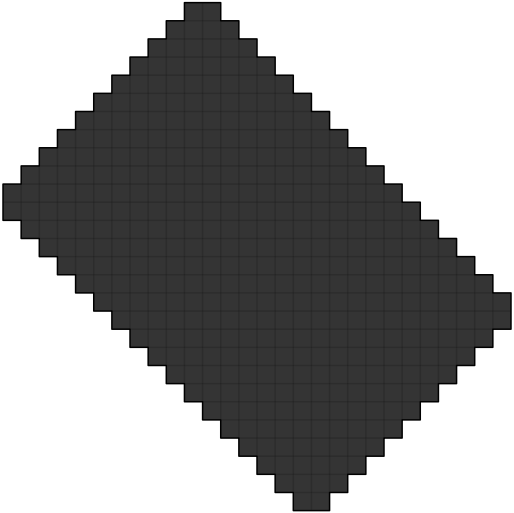
Tiling Aztec Rectangle (cont'd)

Tiling Aztec Rectangle (cont'd)

The problem of tiling an Aztec Rectangle can be solved **recursively**.

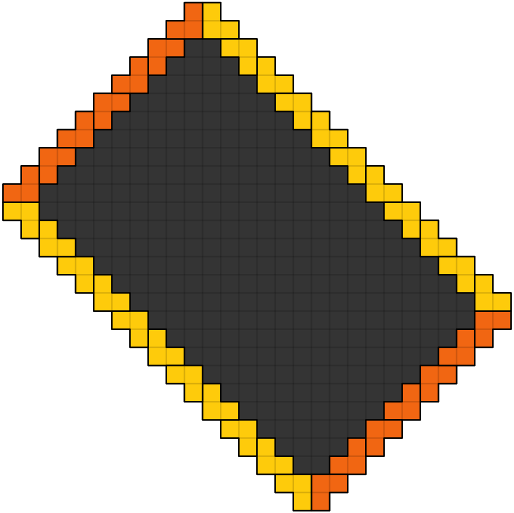
Tiling Aztec Rectangle (cont'd)

The problem of tiling an Aztec Rectangle can be solved **recursively**.



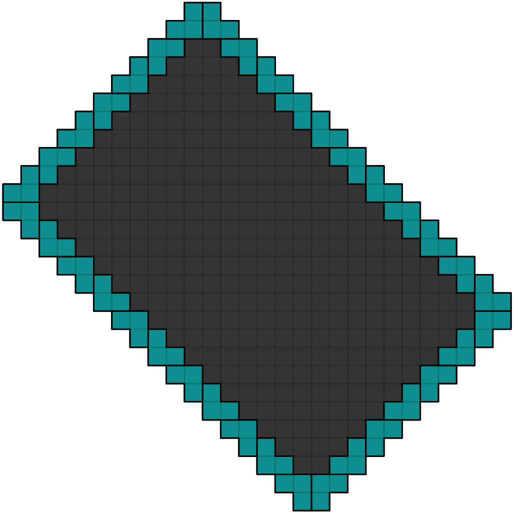
Tiling Aztec Rectangle (cont'd)

The problem of tiling an Aztec Rectangle can be solved **recursively**.



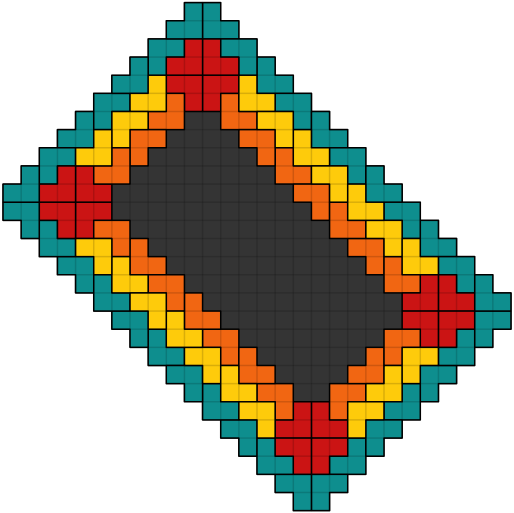
Tiling Aztec Rectangle (cont'd)

The problem of tiling an Aztec Rectangle can be solved **recursively**.



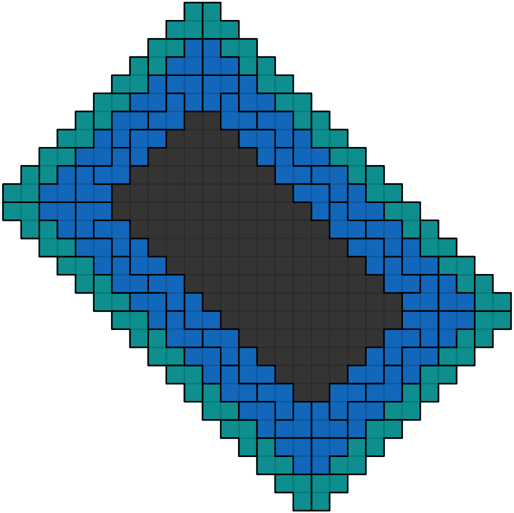
Tiling Aztec Rectangle (cont'd)

The problem of tiling an Aztec Rectangle can be solved **recursively**.



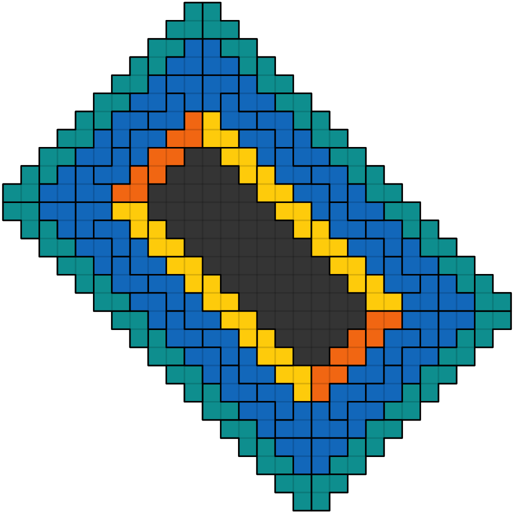
Tiling Aztec Rectangle (cont'd)

The problem of tiling an Aztec Rectangle can be solved **recursively**.



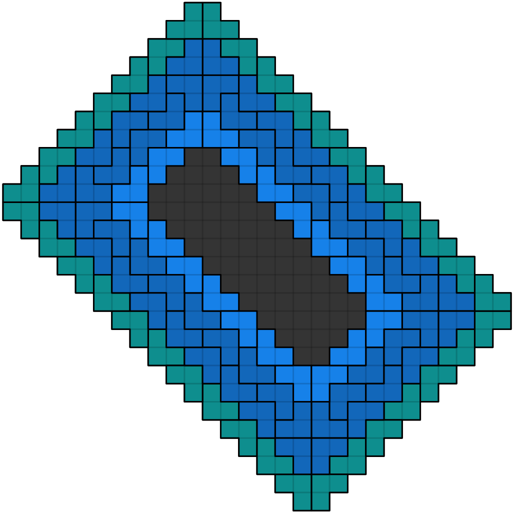
Tiling Aztec Rectangle (cont'd)

The problem of tiling an Aztec Rectangle can be solved **recursively**.

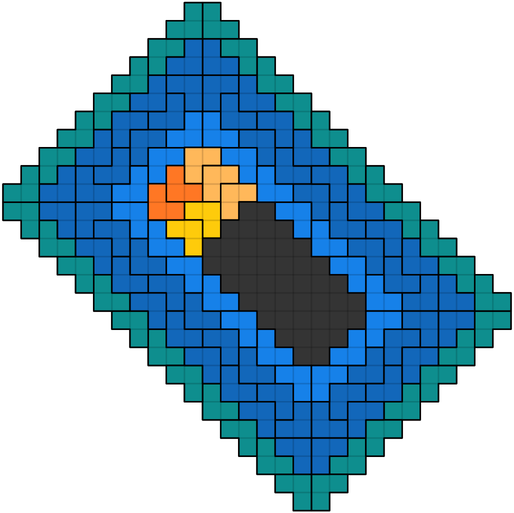


Tiling Aztec Rectangle (cont'd)

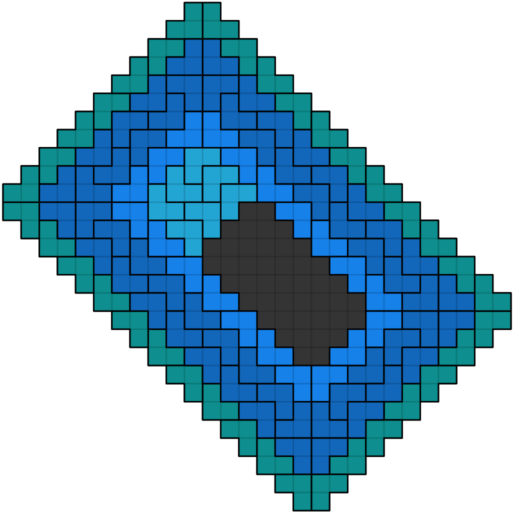
The problem of tiling an Aztec Rectangle can be solved **recursively**.



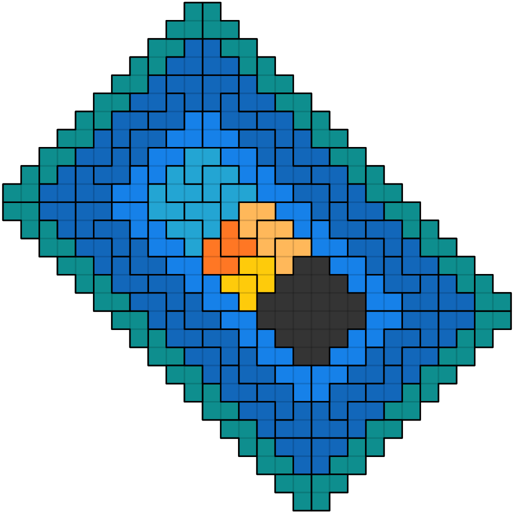
The problem of tiling an Aztec Rectangle can be solved **recursively**.



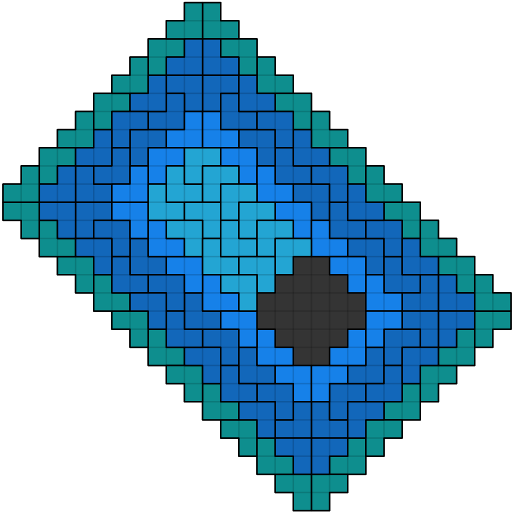
The problem of tiling an Aztec Rectangle can be solved **recursively**.



The problem of tiling an Aztec Rectangle can be solved **recursively**.

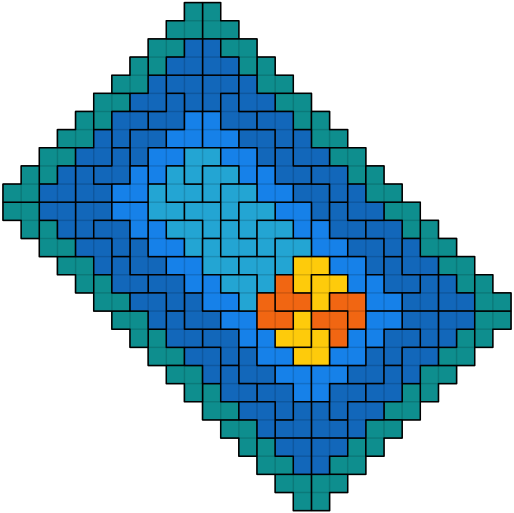


The problem of tiling an Aztec Rectangle can be solved **recursively**.



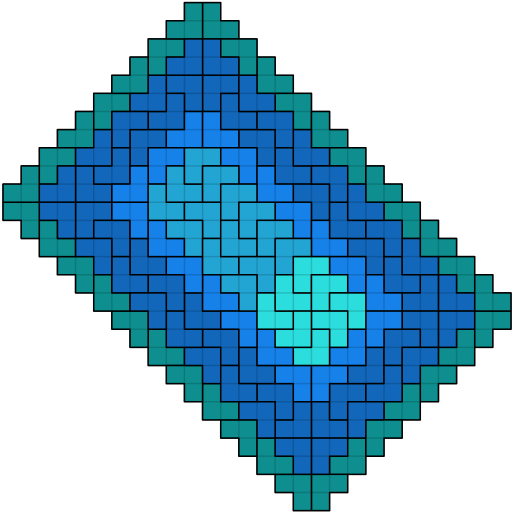
Tiling Aztec Rectangle (cont'd)

The problem of tiling an Aztec Rectangle can be solved **recursively**.



Tiling Aztec Rectangle (cont'd)

The problem of tiling an Aztec Rectangle can be solved **recursively**.

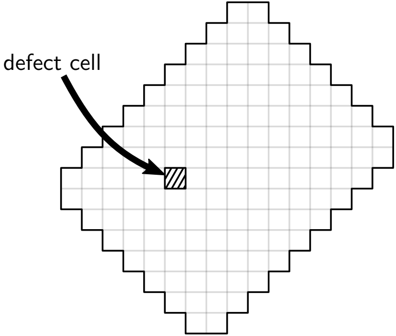


Outline

Tiling Aztec Rectangle with a single defect

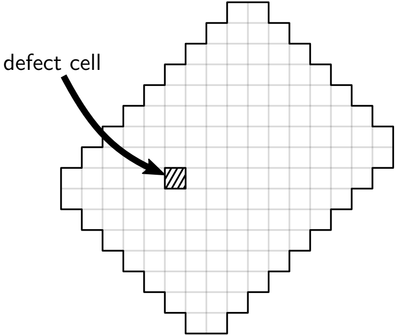
A **defect cell** is a cell in which no tromino can be placed on top.

A **defect cell** is a cell in which no tromino can be placed on top.



Tiling Aztec Rectangle with a single defect

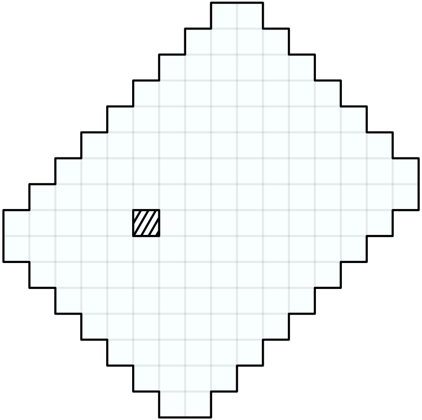
A **defect cell** is a cell in which no tromino can be placed on top.



Theorem

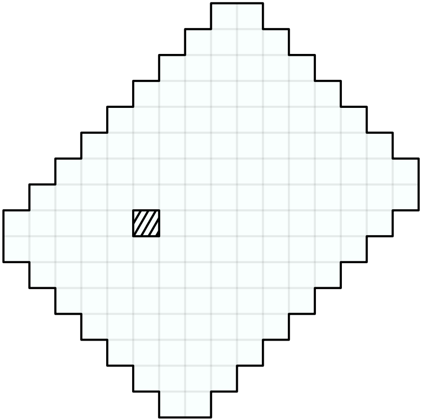
An Aztec rectangle $\mathcal{AR}_{a,b}$ *with one defect has a tiling with L-trominoes*
$\iff$ $|\mathcal{AR}_{a,b}| \equiv 1 \pmod 3$
$\iff$ *a or b is equal to* $3k - 2$ *for some* $k \in \mathbb{N}$.

Tiling Aztec Rectangle with a single defect (cont'd)



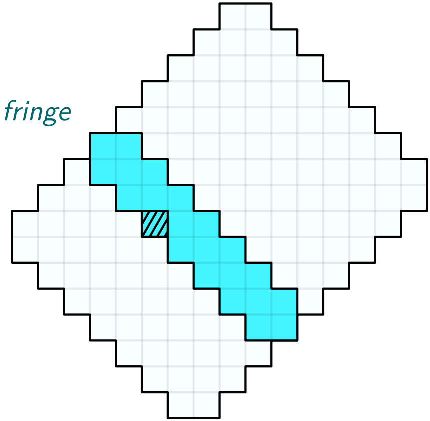
Tiling Aztec Rectangle with a single defect (cont'd)

- Place a *fringe* where it covers the defect.



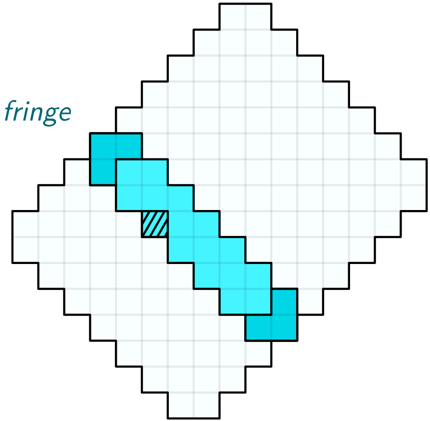
Tiling Aztec Rectangle with a single defect (cont'd)

- Place a *fringe* where it covers the defect.



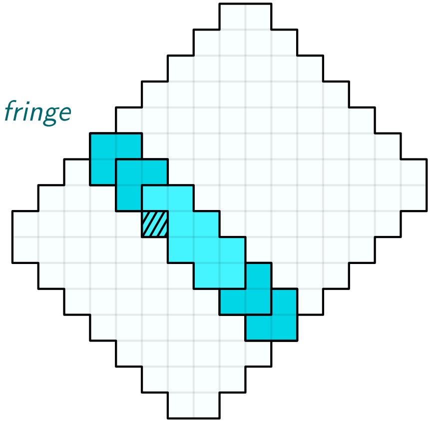
Tiling Aztec Rectangle with a single defect (cont'd)

- Place a *fringe* where it covers the defect.



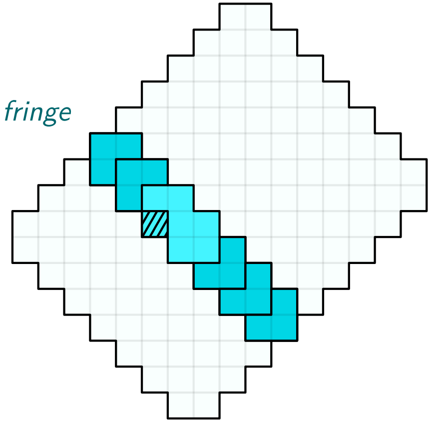
Tiling Aztec Rectangle with a single defect (cont'd)

- Place a *fringe* where it covers the defect.

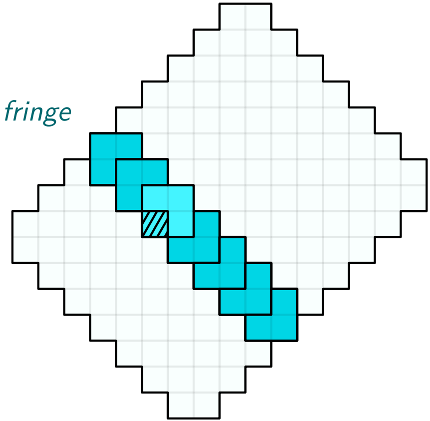


Tiling Aztec Rectangle with a single defect (cont'd)

- Place a *fringe* where it covers the defect.

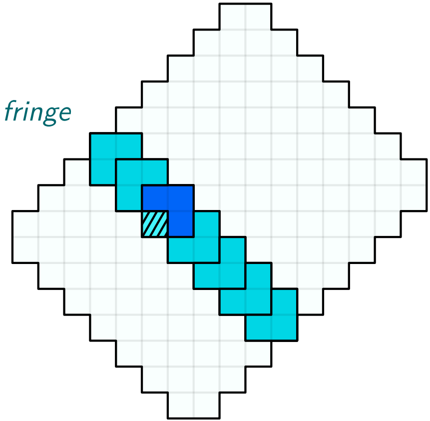


- Place a *fringe* where it covers the defect.



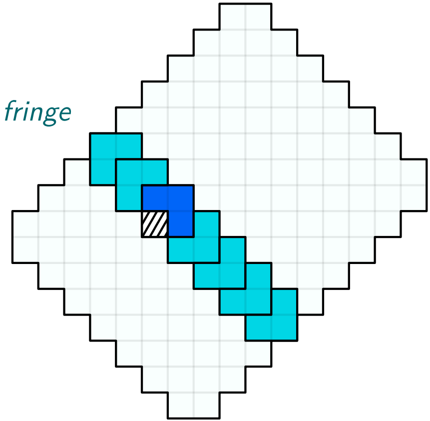
Tiling Aztec Rectangle with a single defect (cont'd)

- Place a *fringe* where it covers the defect.



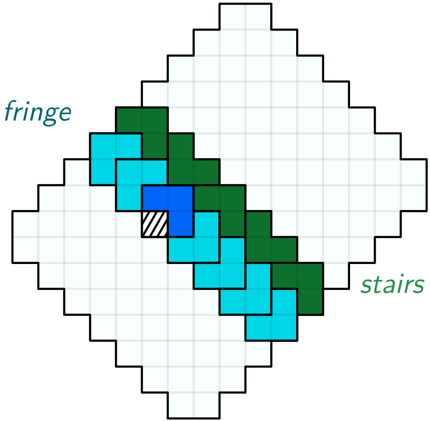
Tiling Aztec Rectangle with a single defect (cont'd)

- Place a *fringe* where it covers the defect.
- Place *stairs* to cover other cells.



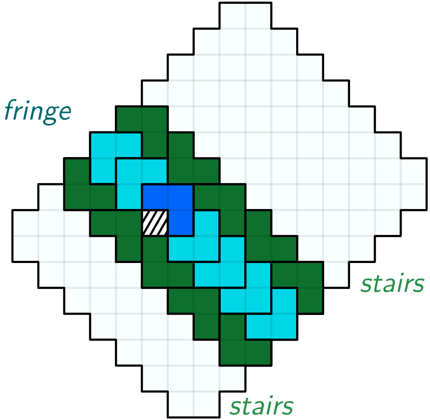
Tiling Aztec Rectangle with a single defect (cont'd)

- Place a *fringe* where it covers the defect.
- Place *stairs* to cover other cells.



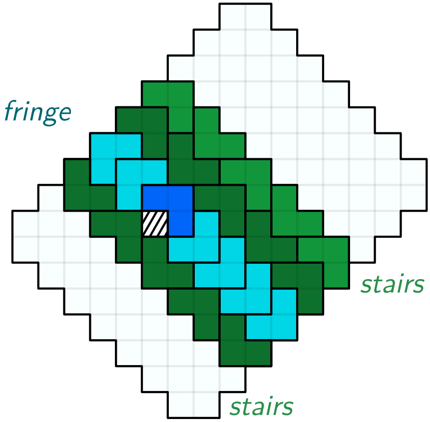
Tiling Aztec Rectangle with a single defect (cont'd)

- Place a *fringe* where it covers the defect.
- Place *stairs* to cover other cells.



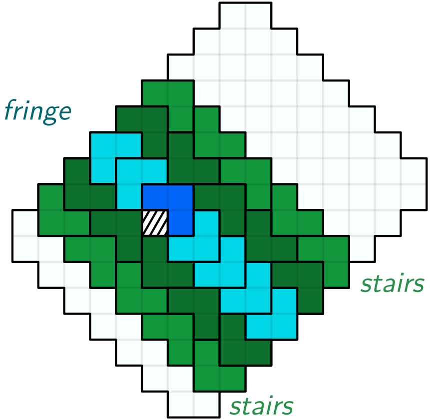
Tiling Aztec Rectangle with a single defect (cont'd)

- Place a *fringe* where it covers the defect.
- Place *stairs* to cover other cells.



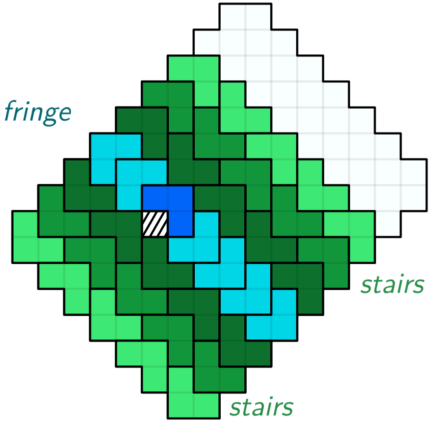
Tiling Aztec Rectangle with a single defect (cont'd)

- Place a *fringe* where it covers the defect.
- Place *stairs* to cover other cells.



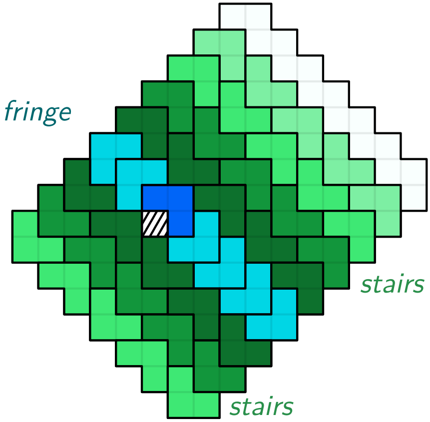
Tiling Aztec Rectangle with a single defect (cont'd)

- Place a *fringe* where it covers the defect.
- Place *stairs* to cover other cells.



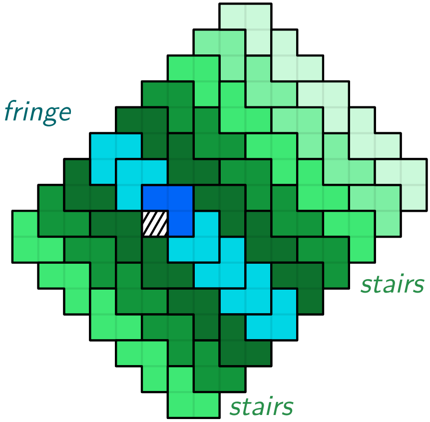
Tiling Aztec Rectangle with a single defect (cont'd)

- Place a *fringe* where it covers the defect.
- Place *stairs* to cover other cells.



Tiling Aztec Rectangle with a single defect (cont'd)

- Place a *fringe* where it covers the defect.
- Place *stairs* to cover other cells.



Outline

Tiling Aztec Rectangle with an unbounded number of defects

Tiling Aztec Rectangle with an unbounded number of defects

Theorem

The problem of tiling Aztec Rectangle $\mathcal{AR}_{a,b}$ with an unbounded number of defects is ***NP-complete****.*

Tiling Aztec Rectangle with an unbounded number of defects

Theorem

The problem of tiling Aztec Rectangle $\mathcal{AR}_{a,b}$ with an unbounded number of defects is ***NP-complete****.*

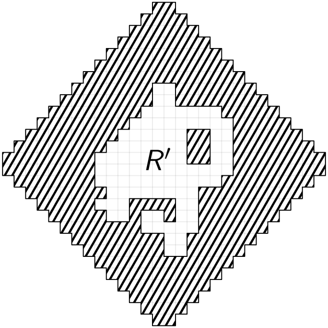
Given a region R', we can embed R' inside a sufficiently large Aztec Rectangle $\mathcal{AR}_{a,b}$.

Tiling Aztec Rectangle with an unbounded number of defects

Theorem

*The problem of tiling Aztec Rectangle $\mathcal{AR}_{a,b}$ with an unbounded number of defects is **NP-complete**.*

Given a region R', we can embed R' inside a sufficiently large Aztec Rectangle $\mathcal{AR}_{a,b}$.



3 180-Tromino Tiling

Outline

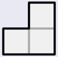
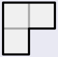
Outline

Definition

The 180-tromino tiling problem only allows 180° rotations of L-trominoes, i.e., the tile set can be

Definition

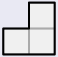
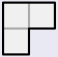
The 180-tromino tiling problem only allows 180° rotations of L-trominoes, i.e., the tile set can be

$\Sigma = \{$ **right-oriented** 180-trominoes $\} = \{$ ,  $\}$

180°L-Tromino Tiling

Definition

The 180-tromino tiling problem only allows 180° rotations of L-trominoes, i.e., the tile set can be

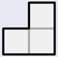
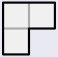
$\Sigma = \{$ **right-oriented** 180-trominoes $\} = \{$ ,  $\}$

or

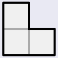
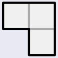
180°L-Tromino Tiling

Definition

The 180-tromino tiling problem only allows 180° rotations of L-trominoes, i.e., the tile set can be

$\Sigma = \{$ **right-oriented** 180-trominoes $\} = \{$ ,  $\}$

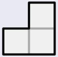
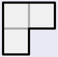
or

$\Sigma = \{$ **left-oriented** 180-trominoes $\} = \{$ ,  $\}$.

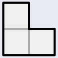
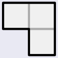
180°L-Tromino Tiling

Definition

The 180-tromino tiling problem only allows 180° rotations of L-trominoes, i.e., the tile set can be

$\Sigma = \{$ **right-oriented** 180-trominoes $\} = \{$ ,  $\}$

or

$\Sigma = \{$ **left-oriented** 180-trominoes $\} = \{$ ,  $\}$.

With no loss of generality, we will only consider **right-oriented 180-trominoes**.

180°L-Tromino Tiling (cont'd)

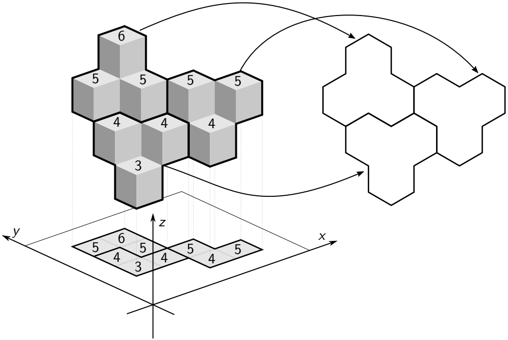
180°L-Tromino Tiling (cont'd)

Theorem (Conway y Lagarias, 1990)

There is a one-one correspondence between 180-tromino tiling and the triangular trihex tiling.

Theorem (Conway y Lagarias, 1990)

There is a one-one correspondence between 180-tromino tiling and the triangular trihex tiling.



Transformation from **triangular trihex** to **180-tromino**

Definition

A **cell tetrasection** is a division of a cell into 4 equal size cells.

180°L-Tromino Tiling (cont'd)

Definition

A **cell tetrasection** is a division of a cell into 4 equal size cells.

Definition

A **tetrasected polyomino** $P^{\boxplus}$ is obtained by tetrasecting each cell of a poylomino P.

180°L-Tromino Tiling (cont'd)

Definition

A **cell tetrasection** is a division of a cell into 4 equal size cells.

Definition

A **tetrasected polyomino** $P^{\boxplus}$ is obtained by tetrasecting each cell of a poylomino P.

If there is a I-tromino tiling for some R, then there is also a 180-tromino tiling for $R^{\boxplus}$.

180°L-Tromino Tiling (cont'd)

Definition

A **cell tetrasection** is a division of a cell into 4 equal size cells.

Definition

A **tetrasected polyomino** $P^{\boxplus}$ is obtained by tetrasecting each cell of a poylomino P.

If there is a I-tromino tiling for some R, then there is also a 180-tromino tiling for $R^{\boxplus}$.

180°L-Tromino Tiling (cont'd)

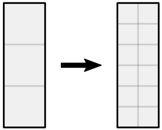
Definition

A **cell tetrasection** is a division of a cell into 4 equal size cells.

Definition

A **tetrasected polyomino** $P^{\boxplus}$ is obtained by tetrasecting each cell of a poylomino P.

If there is a I-tromino tiling for some R, then there is also a 180-tromino tiling for $R^{\boxplus}$.



180°L-Tromino Tiling (cont'd)

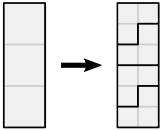
Definition

A **cell tetrasection** is a division of a cell into 4 equal size cells.

Definition

A **tetrasected polyomino** $P^{\boxplus}$ is obtained by tetrasecting each cell of a poylomino P.

If there is a I-tromino tiling for some R, then there is also a 180-tromino tiling for $R^{\boxplus}$.



180°L-Tromino Tiling (cont'd)

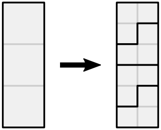
Definition

A **cell tetrasection** is a division of a cell into 4 equal size cells.

Definition

A **tetrasected polyomino** $P^{\boxplus}$ is obtained by tetrasecting each cell of a poylomino P.

If there is a I-tromino tiling for some R, then there is also a 180-tromino tiling for $R^{\boxplus}$.



However, it is not known if the converse statement is true or false.

180°L-Tromino Tiling (cont'd)

180°L-Tromino Tiling (cont'd)

Horiyama et al. proved that the l-tromino tiling problem is **NP-Complete**.

180°L-Tromino Tiling (cont'd)

Horiyama et al. proved that the I-tromino tiling problem is **NP-Complete**.

Theorem (Horiyama *et al.*, 2012)

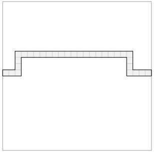
1-IN-3 SAT $\leq_p$ I-TROMINO

180°L-Tromino Tiling (cont'd)

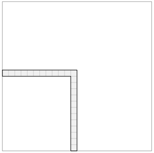
Horiyama et al. proved that the I-tromino tiling problem is **NP-Complete**.

Theorem (Horiyama *et al.*, 2012)

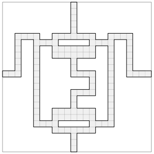
1-IN-3 SAT $\leq_p$ I-TROMINO



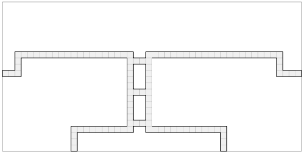
(a) Line gadget.



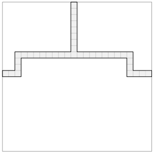
(b) Corner gadget.



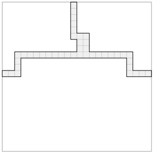
(c) Cross gadget.



(d) Duplicator gadget.



(e) Clause gadget.

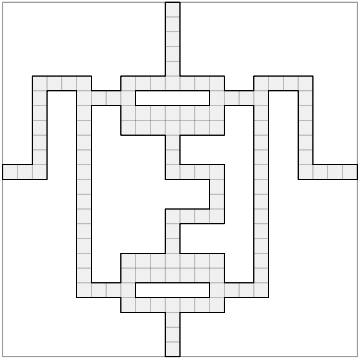


(f) Negated clause gadget.

180°L-Tromino Tiling (cont'd)

In each gadget G, I-tromino tiling for G can be simulated with 180-tromino tiling for $G^{\boxplus}$.

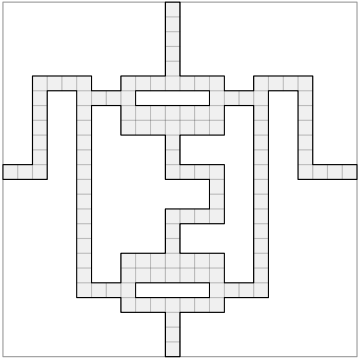
In each gadget G, I-tromino tiling for G can be simulated with 180-tromino tiling for $G^{\boxplus}$.



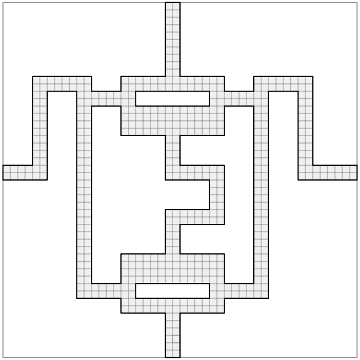
(a) Original gadget G.

180°L-Tromino Tiling (cont'd)

In each gadget G, I-tromino tiling for G can be simulated with 180-tromino tiling for $G^{\boxplus}$.



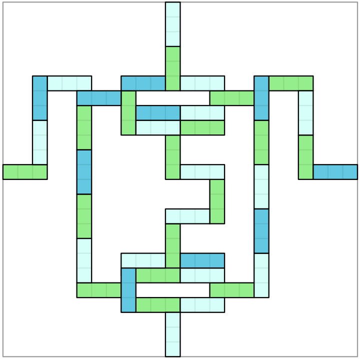
(a) Original gadget G.



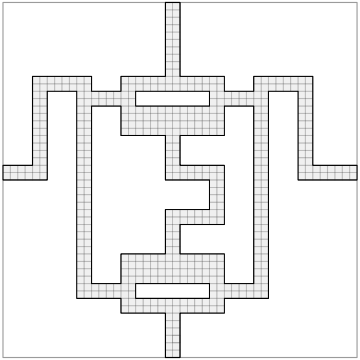
(b) Tetrasected gadget $G^{\boxplus}$.

180°L-Tromino Tiling (cont'd)

In each gadget G, I-tromino tiling for G can be simulated with 180-tromino tiling for $G^{\boxplus}$.



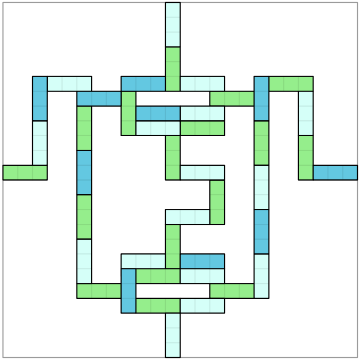
(a) Original gadget G.



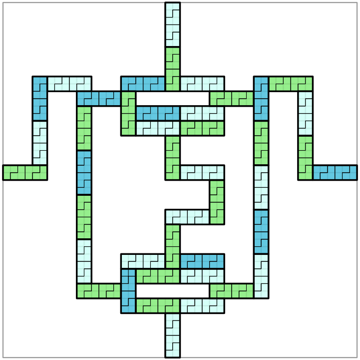
(b) Tetrasected gadget $G^{\boxplus}$.

180°L-Tromino Tiling (cont'd)

In each gadget G, I-tromino tiling for G can be simulated with 180-tromino tiling for $G^{\boxplus}$.

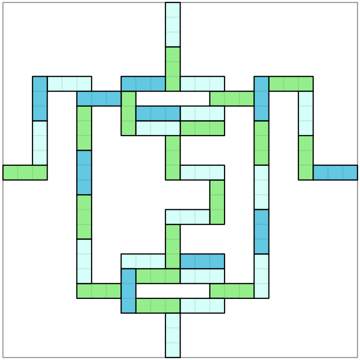


(a) Original gadget G.

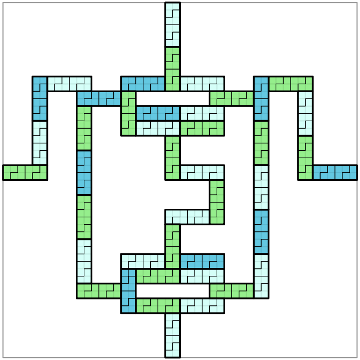


(b) Tetrasected gadget $G^{\boxplus}$.

In each gadget G, I-tromino tiling for G can be simulated with 180-tromino tiling for $G^{\boxplus}$.



(a) Original gadget G.



(b) Tetrasected gadget $G^{\boxplus}$.

Theorem

180-tromino tiling is ***NP-complete****.*

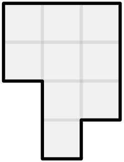
Outline

1. Introduction
   - Polyominoes
   - L-Tromino Tiling Problem
2. Tiling Aztec Rectangles
   - Aztec Rectangle
   - Aztec Rectangle with a single defect
   - Tiling Aztec Rectangle with unbounded number of defects
3. 180-Tromino Tiling
   - A rotation constraint
   - Forbidden Polyominoes
4. Tiling Tetrasected Region
   - Special case: the Dual Graph is a Tree
   - General case

The 180-tromino tiling can also be reduced to the **Maximum Independent Set** problem.

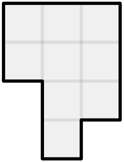
The 180-tromino tiling can also be reduced to the **Maximum Independent Set** problem.

R



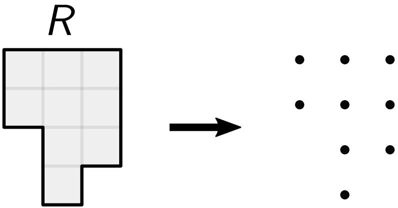
The 180-tromino tiling can also be reduced to the **Maximum Independent Set** problem.

R



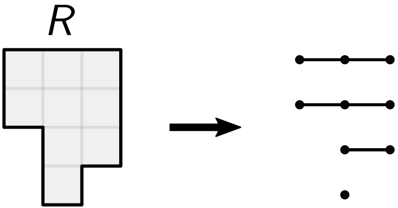
- Transformation from R to G_R:
  - Transform every cell of R to vertices of G_R.
  - Add horizontal, vertical and northeast-diagonal edges.

The 180-tromino tiling can also be reduced to the **Maximum Independent Set** problem.



- Transformation from R to G_R:
    - Transform every cell of R to vertices of G_R.
    - Add horizontal, vertical and northeast-diagonal edges.

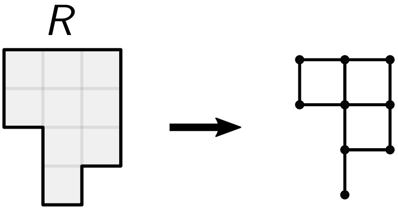
The 180-tromino tiling can also be reduced to the **Maximum Independent Set** problem.



- Transformation from R to G_R:
  - Transform every cell of R to vertices of G_R.
  - Add horizontal, vertical and northeast-diagonal edges.

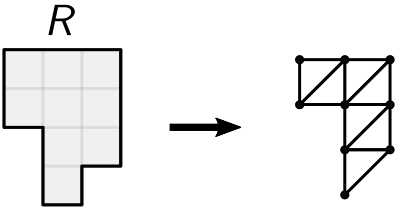
Forbidden Polyominoes

The 180-tromino tiling can also be reduced to the **Maximum Independent Set** problem.



- Transformation from R to G_R:
  - Transform every cell of R to vertices of G_R.
  - Add horizontal, vertical and northeast-diagonal edges.

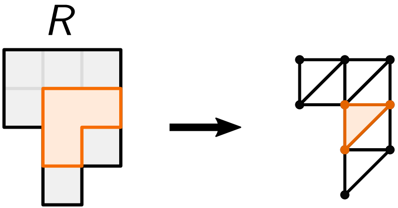
The 180-tromino tiling can also be reduced to the **Maximum Independent Set** problem.



- Transformation from R to G_R:
  - Transform every cell of R to vertices of G_R.
  - Add horizontal, vertical and northeast-diagonal edges.

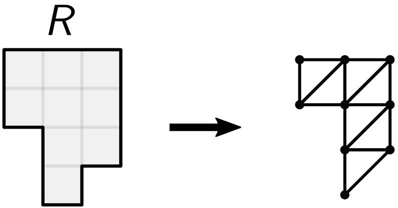
Forbidden Polyominoes

The 180-tromino tiling can also be reduced to the **Maximum Independent Set** problem.



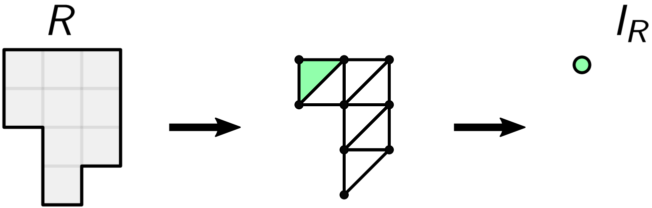
- Transformation from R to G_R:
  - Transform every cell of R to vertices of G_R.
  - Add horizontal, vertical and northeast-diagonal edges.

The 180-tromino tiling can also be reduced to the **Maximum Independent Set** problem.



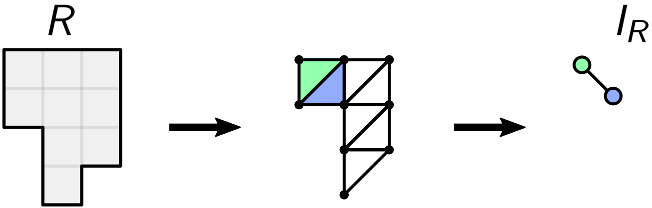
- Transformation from R to G_R:
  - Transform every cell of R to vertices of G_R.
  - Add horizontal, vertical and northeast-diagonal edges.

The 180-tromino tiling can also be reduced to the **Maximum Independent Set** problem.



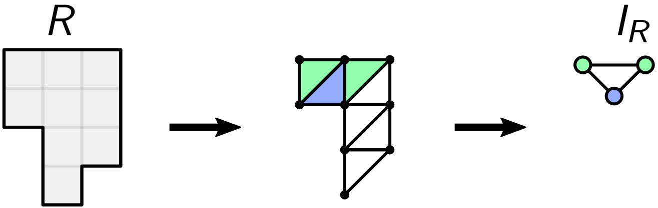
- Transformation from R to G_R:
  - Transform every cell of R to vertices of G_R.
  - Add horizontal, vertical and northeast-diagonal edges.
- Transformation from G_R to I_R:
  - Transform every 3-cycle of G_R to vertices of I_R.
  - Add an edge where 3-cycles intersects.

The 180-tromino tiling can also be reduced to the **Maximum Independent Set** problem.



- Transformation from R to G_R:
  - Transform every cell of R to vertices of G_R.
  - Add horizontal, vertical and northeast-diagonal edges.
- Transformation from G_R to I_R:
  - Transform every 3-cycle of G_R to vertices of I_R.
  - Add an edge where 3-cycles intersects.

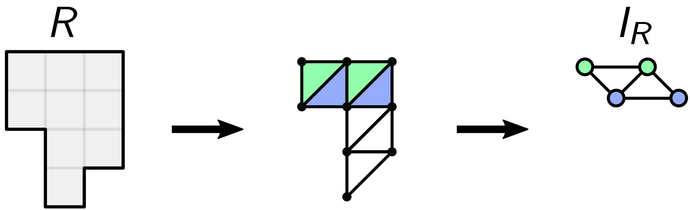
The 180-tromino tiling can also be reduced to the **Maximum Independent Set** problem.



- Transformation from R to G_R:
  - Transform every cell of R to vertices of G_R.
  - Add horizontal, vertical and northeast-diagonal edges.
- Transformation from G_R to I_R:
  - Transform every 3-cycle of G_R to vertices of I_R.
  - Add an edge where 3-cycles intersects.

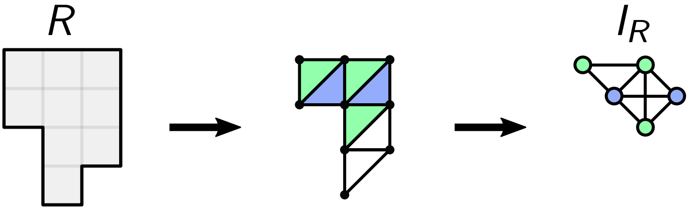
Forbidden Polyominoes

The 180-tromino tiling can also be reduced to the **Maximum Independent Set** problem.



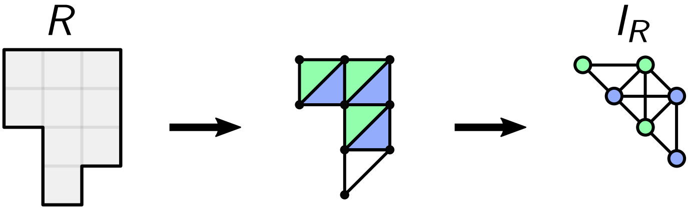
- Transformation from R to G_R:
  - Transform every cell of R to vertices of G_R.
  - Add horizontal, vertical and northeast-diagonal edges.
- Transformation from G_R to I_R:
  - Transform every 3-cycle of G_R to vertices of I_R.
  - Add an edge where 3-cycles intersects.

The 180-tromino tiling can also be reduced to the **Maximum Independent Set** problem.



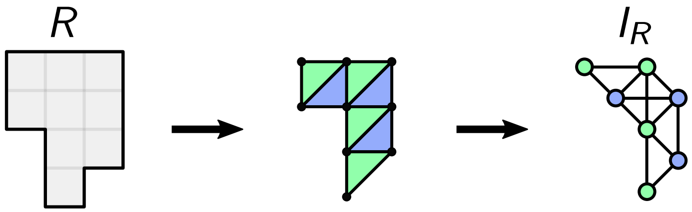
- Transformation from R to G_R:
  - Transform every cell of R to vertices of G_R.
  - Add horizontal, vertical and northeast-diagonal edges.
- Transformation from G_R to I_R:
  - Transform every 3-cycle of G_R to vertices of I_R.
  - Add an edge where 3-cycles intersects.

The 180-tromino tiling can also be reduced to the **Maximum Independent Set** problem.



- Transformation from R to G_R:
  - Transform every cell of R to vertices of G_R.
  - Add horizontal, vertical and northeast-diagonal edges.
- Transformation from G_R to I_R:
  - Transform every 3-cycle of G_R to vertices of I_R.
  - Add an edge where 3-cycles intersects.

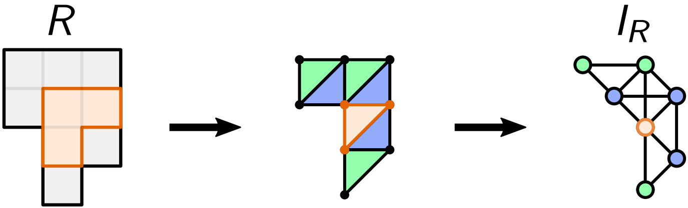
The 180-tromino tiling can also be reduced to the **Maximum Independent Set** problem.



- Transformation from R to G_R:
  - Transform every cell of R to vertices of G_R.
  - Add horizontal, vertical and northeast-diagonal edges.
- Transformation from G_R to I_R:
  - Transform every 3-cycle of G_R to vertices of I_R.
  - Add an edge where 3-cycles intersects.

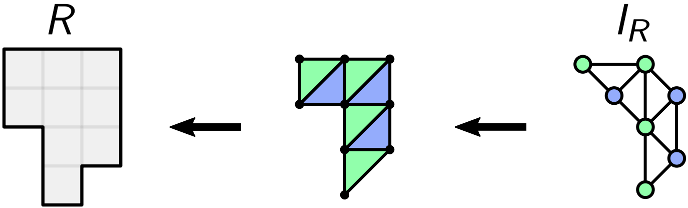
Forbidden Polyominoes

The 180-tromino tiling can also be reduced to the **Maximum Independent Set** problem.

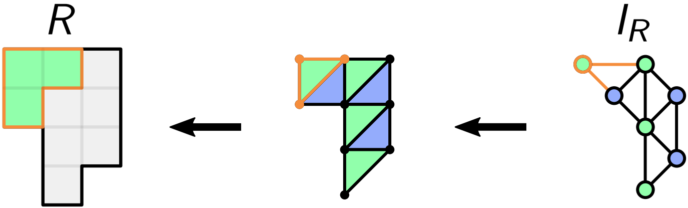


- Transformation from R to G_R:
  - Transform every cell of R to vertices of G_R.
  - Add horizontal, vertical and northeast-diagonal edges.
- Transformation from G_R to I_R:
  - Transform every 3-cycle of G_R to vertices of I_R.
  - Add an edge where 3-cycles intersects.

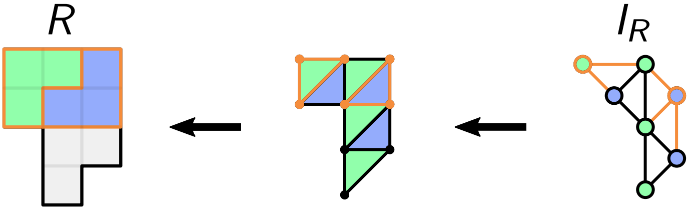
Forbidden Polyominoes (cont'd)



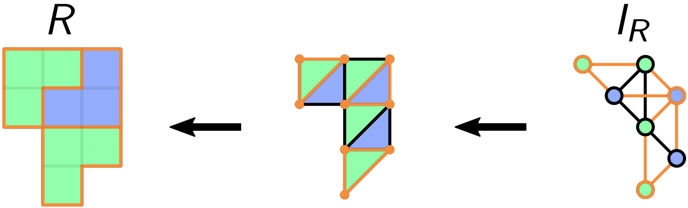
Forbidden Polyominoes (cont'd)

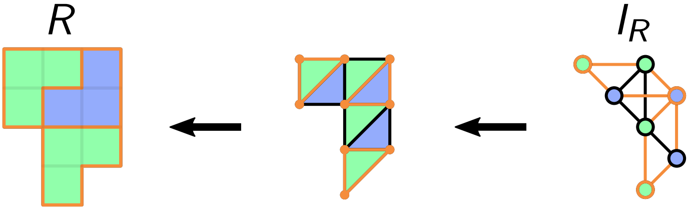


Forbidden Polyominoes (cont'd)



Forbidden Polyominoes (cont'd)





Theorem

Maximum Independent Set of I_R is equal to $\frac{|R|}{3}$
$\iff$ *R has a 180-tromino tiling .*

where $|R|$ the number of cells in a region R.

Forbidden Polyominoes (cont'd)

Forbidden Polyominoes (cont'd)

If I_G is **claw-free**, i.e., does not contain a claw as induced graph, then computing Maximum Independent Set can be computed in polynomial time.

Forbidden Polyominoes (cont'd)

If I_G is **claw-free**, i.e., does not contain a claw as induced graph, then computing Maximum Independent Set can be computed in polynomial time.

Forbidden Polyominoes (cont'd)

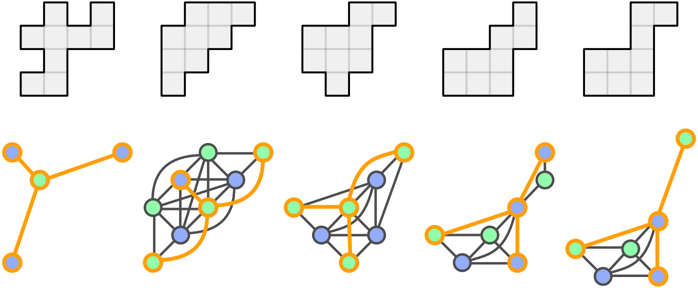
If I_G is **claw-free**, i.e., does not contain a claw as induced graph, then computing Maximum Independent Set can be computed in polynomial time.

The following five polyominoes generates a distinct I_G with a **claw** in it.

Forbidden Polyominoes (cont'd)

If I_G is **claw-free**, i.e., does not contain a claw as induced graph, then computing Maximum Independent Set can be computed in polynomial time.

The following five polyominoes generates a distinct I_G with a **claw** in it.

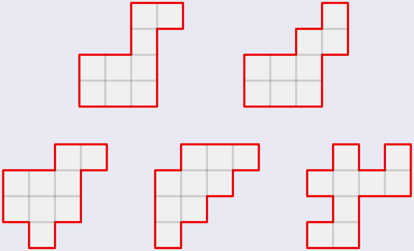


Forbidden Polyominoes (cont'd)

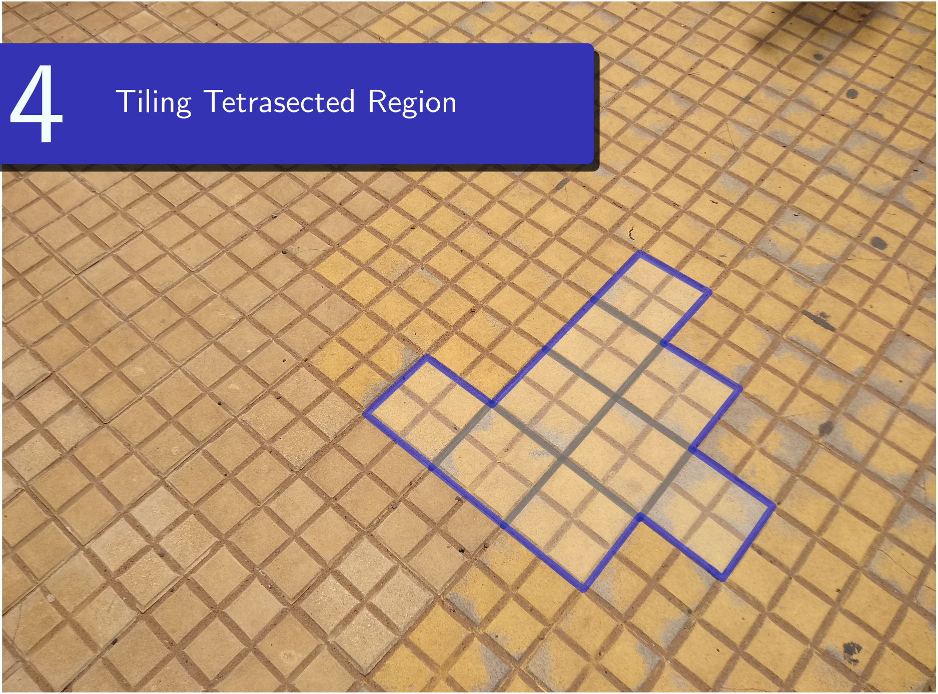
Forbidden Polyominoes (cont'd)

Theorem

*If a region R **doesn't** contains a rotated, reflected or sheared **forbidden polyomino**, then 180-tromino tiling can be computed in polynomial time.*



4 Tiling Tetrasected Region



Outline

Outline

Tiling a Tetrasected Region

Tiling a Tetrasected Region

Theorem

A tetrasected region $R^{\boxplus}$ has a L-tromino tiling if and only if $|R^{\boxplus}| \equiv 0 \pmod 3$. Furthermore, there exist a $O(n \log n)$ algorithm that computes the L-tromino tiling of $R^{\boxplus}$, where $n = |R|$.

Theorem

A tetrasected region $R^{\boxplus}$ has a L-tromino tiling if and only if $|R^{\boxplus}| \equiv 0 \pmod 3$. Furthermore, there exist a $O(n \log n)$ algorithm that computes the L-tromino tiling of $R^{\boxplus}$, where $n = |R|$.

We divide the algorithm in two cases:

Theorem

A tetrasected region $R^{\boxplus}$ has a L-tromino tiling if and only if $|R^{\boxplus}| \equiv 0 \pmod 3$. Furthermore, there exist a $O(n \log n)$ algorithm that computes the L-tromino tiling of $R^{\boxplus}$, where $n = |R|$.

We divide the algorithm in two cases:

- **Special case** is when the dual graph of R is a tree.

Theorem

A tetrasected region $R^{\boxplus}$ has a L-tromino tiling if and only if $|R^{\boxplus}| \equiv 0 \pmod 3$. Furthermore, there exist a $O(n \log n)$ algorithm that computes the L-tromino tiling of $R^{\boxplus}$, where $n = |R|$.

We divide the algorithm in two cases:

- **Special case** is when the dual graph of R is a tree.
- **General case**.

Special case: the Dual Graph is a Tree

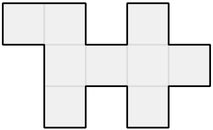
Special case: the Dual Graph is a Tree

Definition

A **Dual Graph** G_R of a polyomino R is a graph obtained by replacing the cells by vertices and connecting vertices for each pair of adjacent cells.

Definition

A **Dual Graph** G_R of a polyomino R is a graph obtained by replacing the cells by vertices and connecting vertices for each pair of adjacent cells.

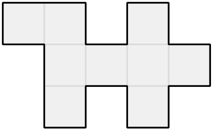


(a) A region R.

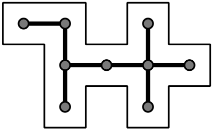
Special case: the Dual Graph is a Tree

Definition

A **Dual Graph** G_R of a polyomino R is a graph obtained by replacing the cells by vertices and connecting vertices for each pair of adjacent cells.



(a) A region R.



(b) Dual graph G_R.

Special case: the Dual Graph is a Tree (cont'd.)

Special case: the Dual Graph is a Tree (cont'd.)

Let R be a region where the dual graph is a tree and $|R| \equiv 0 \pmod 3$.

Special case: the Dual Graph is a Tree (cont'd.)

Let R be a region where the dual graph is a tree and $|R| \equiv 0 \pmod 3$.

Definition

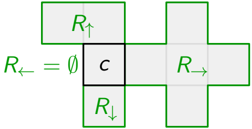
The **tags** of a cell c of R are the number of cells modulo 3 of the spanning subregion in each direction.

Special case: the Dual Graph is a Tree (cont'd.)

Let R be a region where the dual graph is a tree and $|R| \equiv 0 \pmod 3$.

Definition

The **tags** of a cell c of R are the number of cells modulo 3 of the spanning subregion in each direction.



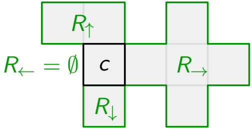
(a) Spanning subregion from c.

Special case: the Dual Graph is a Tree (cont'd.)

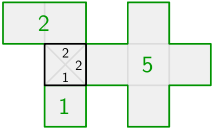
Let R be a region where the dual graph is a tree and $|R| \equiv 0 \pmod 3$.

Definition

The **tags** of a cell c of R are the number of cells modulo 3 of the spanning subregion in each direction.



(a) Spanning subregion from c.



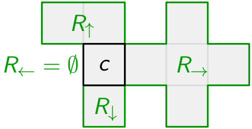
(b) Tags of c.

Special case: the Dual Graph is a Tree (cont'd.)

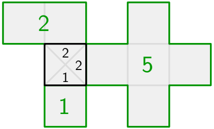
Let R be a region where the dual graph is a tree and $|R| \equiv 0 \pmod 3$.

Definition

The **tags** of a cell c of R are the number of cells modulo 3 of the spanning subregion in each direction.



(a) Spanning subregion from c.



(b) Tags of c.

Lemma

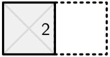
The sum of the tags of any cell is always equal to $2 \pmod 3$.

Special case: the Dual Graph is a Tree (cont'd.)

There are exactly four kinds of **tagged cells**.

Special case: the Dual Graph is a Tree (cont'd.)

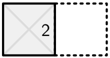
There are exactly four kinds of **tagged cells**.



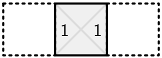
(a) Leaf

Special case: the Dual Graph is a Tree (cont'd.)

There are exactly four kinds of **tagged cells**.

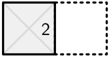


(a) Leaf

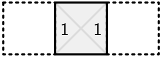


(b) Straight trunk

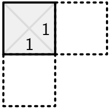
There are exactly four kinds of **tagged cells**.



(a) Leaf



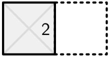
(b) Straight trunk



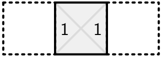
(c) Bent trunk

Special case: the Dual Graph is a Tree (cont'd.)

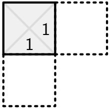
There are exactly four kinds of **tagged cells**.



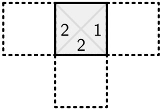
(a) Leaf



(b) Straight trunk

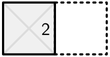


(c) Bent trunk

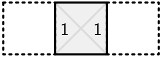


(d) 2-2-1 Fork

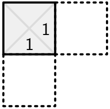
There are exactly four kinds of **tagged cells**.



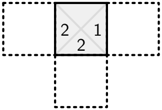
(a) Leaf



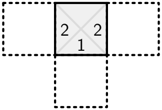
(b) Straight trunk



(c) Bent trunk

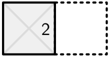


(d) 2-2-1 Fork

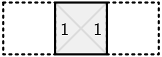


(e) 2-1-2 Fork

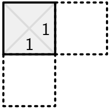
There are exactly four kinds of **tagged cells**.



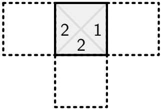
(a) Leaf



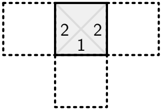
(b) Straight trunk



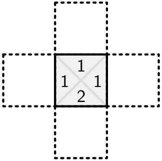
(c) Bent trunk



(d) 2-2-1 Fork

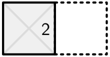


(e) 2-1-2 Fork

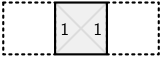


(f) 2^1 Cross

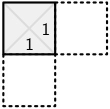
There are exactly four kinds of **tagged cells**.



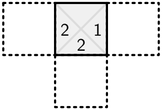
(a) Leaf



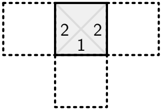
(b) Straight trunk



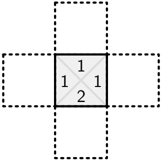
(c) Bent trunk



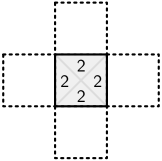
(d) 2-2-1 Fork



(e) 2-1-2 Fork



(f) 2^1 Cross



(g) 2^4 Cross

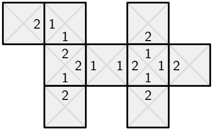
Caso especial cuando el Grafo Dual es un Árbol (cont.)

Caso especial cuando el Grafo Dual es un Árbol (cont.)

With a single DFS traversal of the dual graph of R, we can tag every cell of R.

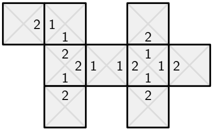
Caso especial cuando el Grafo Dual es un Árbol (cont.)

With a single DFS traversal of the dual graph of R, we can tag every cell of R.



(a) Region with tagged cells.

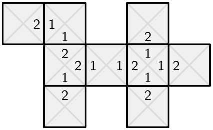
With a single DFS traversal of the dual graph of R, we can tag every cell of R.



(a) Region with tagged cells.

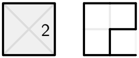
If we replace each tagged cell with the following **tiling patterns**,

With a single DFS traversal of the dual graph of R, we can tag every cell of R.

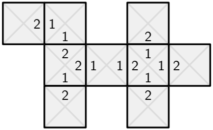


(a) Region with tagged cells.

If we replace each tagged cell with the following tiling patterns,

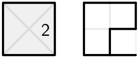


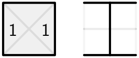
With a single DFS traversal of the dual graph of R, we can tag every cell of R.



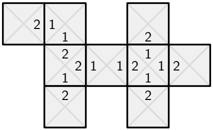
(a) Region with tagged cells.

If we replace each tagged cell with the following tiling patterns,



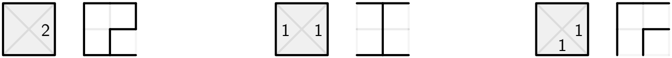


With a single DFS traversal of the dual graph of R, we can tag every cell of R.

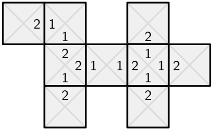


(a) Region with tagged cells.

If we replace each tagged cell with the following **tiling patterns**,

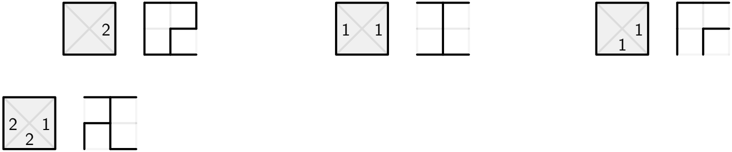


With a single DFS traversal of the dual graph of R, we can tag every cell of R.



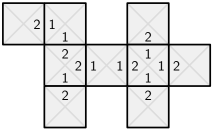
(a) Region with tagged cells.

If we replace each tagged cell with the following **tiling patterns**,



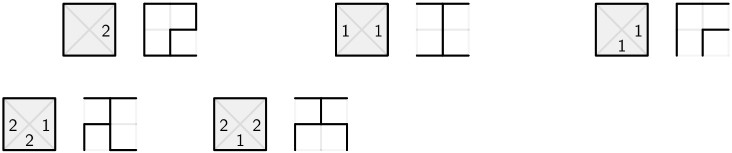
Caso especial cuando el Grafo Dual es un Árbol (cont.)

With a single DFS traversal of the dual graph of R, we can tag every cell of R.



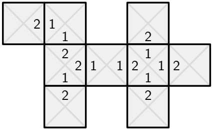
(a) Region with tagged cells.

If we replace each tagged cell with the following **tiling patterns**,



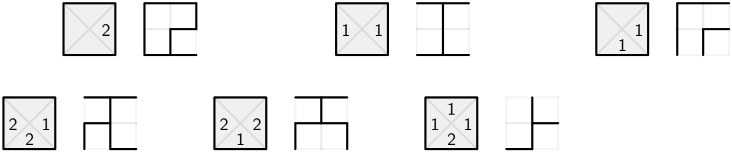
Caso especial cuando el Grafo Dual es un Árbol (cont.)

With a single DFS traversal of the dual graph of R, we can tag every cell of R.

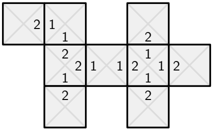


(a) Region with tagged cells.

If we replace each tagged cell with the following **tiling patterns**,

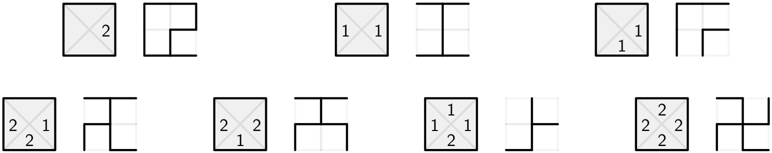


With a single DFS traversal of the dual graph of R, we can tag every cell of R.

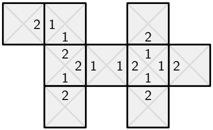


(a) Region with tagged cells.

If we replace each tagged cell with the following **tiling patterns**,

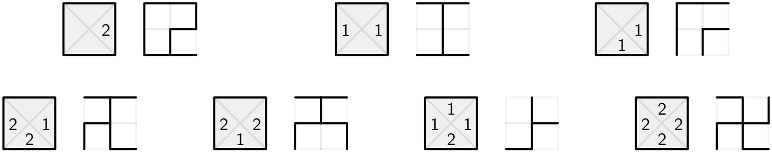


With a single DFS traversal of the dual graph of R, we can tag every cell of R.

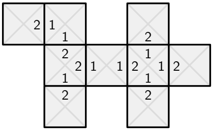


(a) Region with tagged cells.

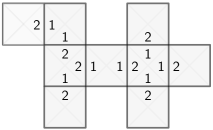
If we replace each tagged cell with the following **tiling patterns**,



With a single DFS traversal of the dual graph of R, we can tag every cell of R.

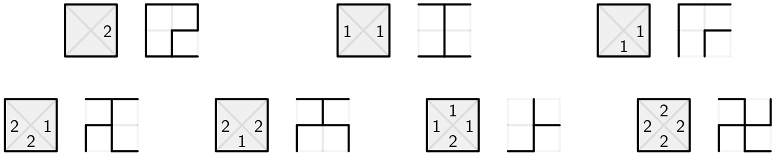


(a) Region with tagged cells.

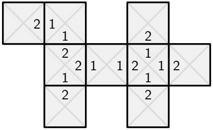


(b) After replacing with tiling patterns.

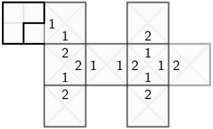
If we replace each tagged cell with the following **tiling patterns**,



With a single DFS traversal of the dual graph of R, we can tag every cell of R.

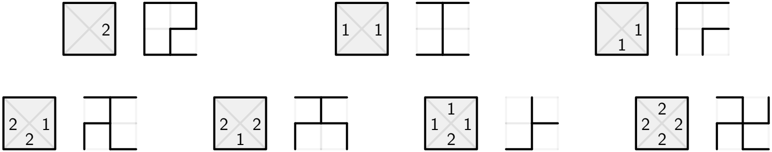


(a) Region with tagged cells.



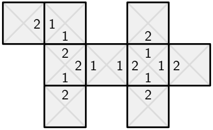
(b) After replacing with tiling patterns.

If we replace each tagged cell with the following **tiling patterns**,

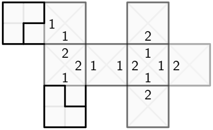


Caso especial cuando el Grafo Dual es un Árbol (cont.)

With a single DFS traversal of the dual graph of R, we can tag every cell of R.

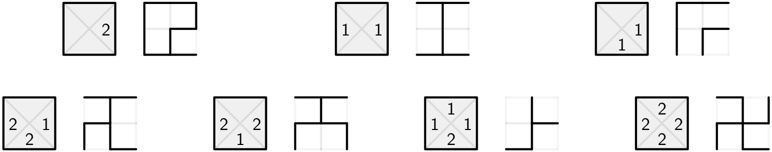


(a) Region with tagged cells.

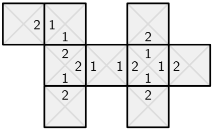


(b) After replacing with tiling patterns.

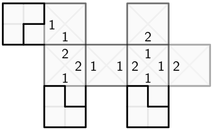
If we replace each tagged cell with the following **tiling patterns**,



With a single DFS traversal of the dual graph of R, we can tag every cell of R.

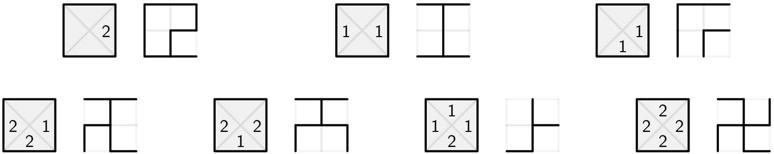


(a) Region with tagged cells.



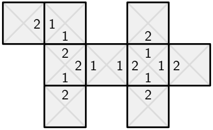
(b) After replacing with tiling patterns.

If we replace each tagged cell with the following **tiling patterns**,

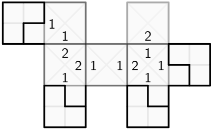


Caso especial cuando el Grafo Dual es un Árbol (cont.)

With a single DFS traversal of the dual graph of R, we can tag every cell of R.

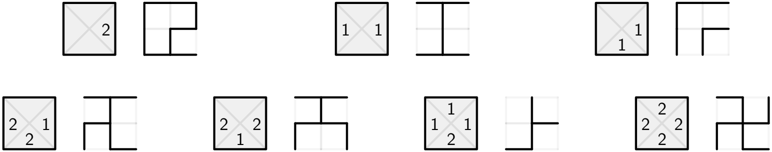


(a) Region with tagged cells.

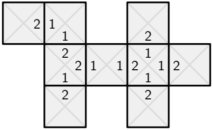


(b) After replacing with tiling patterns.

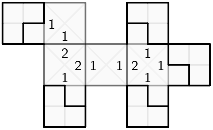
If we replace each tagged cell with the following **tiling patterns**,



With a single DFS traversal of the dual graph of R, we can tag every cell of R.

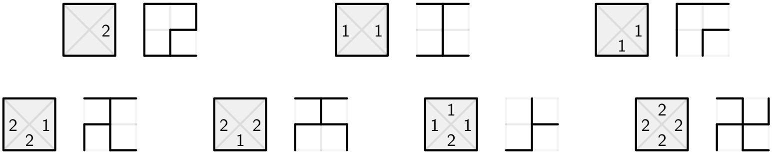


(a) Region with tagged cells.



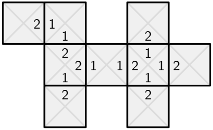
(b) After replacing with tiling patterns.

If we replace each tagged cell with the following **tiling patterns**,

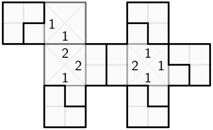


Caso especial cuando el Grafo Dual es un Árbol (cont.)

With a single DFS traversal of the dual graph of R, we can tag every cell of R.

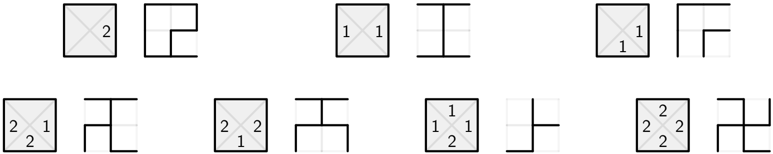


(a) Region with tagged cells.

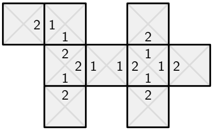


(b) After replacing with tiling patterns.

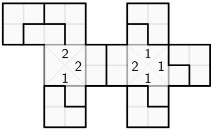
If we replace each tagged cell with the following **tiling patterns**,



With a single DFS traversal of the dual graph of R, we can tag every cell of R.

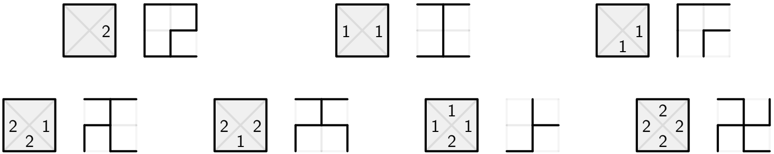


(a) Region with tagged cells.

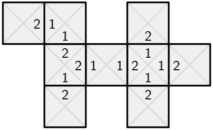


(b) After replacing with tiling patterns.

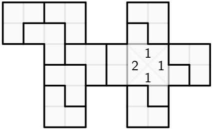
If we replace each tagged cell with the following **tiling patterns**,



With a single DFS traversal of the dual graph of R, we can tag every cell of R.

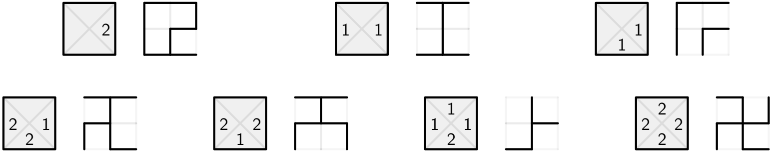


(a) Region with tagged cells.



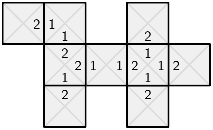
(b) After replacing with tiling patterns.

If we replace each tagged cell with the following **tiling patterns**,

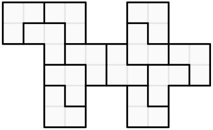


Caso especial cuando el Grafo Dual es un Árbol (cont.)

With a single DFS traversal of the dual graph of R, we can tag every cell of R.

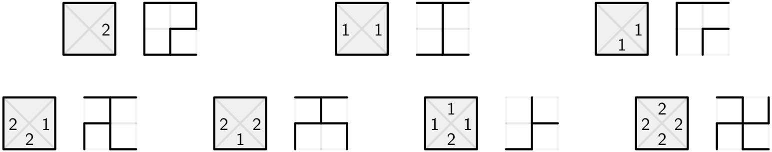


(a) Region with tagged cells.



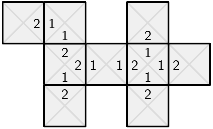
(b) After replacing with tiling patterns.

If we replace each tagged cell with the following **tiling patterns**,

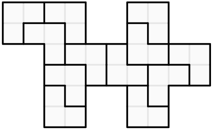


Caso especial cuando el Grafo Dual es un Árbol (cont.)

With a single DFS traversal of the dual graph of R, we can tag every cell of R.

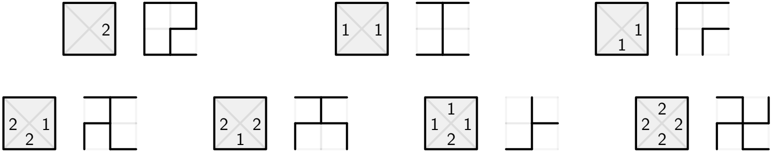


(a) Region with tagged cells.



(b) After replacing with tiling patterns.

If we replace each tagged cell with the following **tiling patterns**,



it will emerge a **L-tromino tiling** of the region $R^{\boxplus}$.

Outline

General case

General case

In the **general case**, we consider a dual graph with **cycles**.

General case

In the **general case**, we consider a dual graph with **cycles**.

Given a region R such that $|R| \equiv 0 \pmod 3$, the following procedure will produce a L-tromino tiling of $R^{\boxplus}$:

General case

In the **general case**, we consider a dual graph with **cycles**.

Given a region R such that $|R| \equiv 0 \pmod 3$, the following procedure will produce a L-tromino tiling of $R^{\boxplus}$:

- Partition the region R into two or more subregions such that:

General case

In the **general case**, we consider a dual graph with **cycles**.

Given a region R such that $|R| \equiv 0 \pmod 3$, the following procedure will produce a L-tromino tiling of $R^{\boxplus}$:

- Partition the region R into two or more subregions such that:
    - the number of cells in each subregions is a multiple of three, and

General case

In the **general case**, we consider a dual graph with **cycles**.

Given a region R such that $|R| \equiv 0 \pmod 3$, the following procedure will produce a L-tromino tiling of $R^{\boxplus}$:

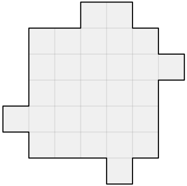
- Partition the region R into two or more subregions such that:
  - the number of cells in each subregions is a multiple of three, and
  - the dual graph of each subregion is a tree.

General case

In the **general case**, we consider a dual graph with **cycles**.

Given a region R such that $|R| \equiv 0 \pmod 3$, the following procedure will produce a L-tromino tiling of $R^{\boxplus}$:

- Partition the region R into two or more subregions such that:
  - the number of cells in each subregions is a multiple of three, and
  - the dual graph of each subregion is a tree.

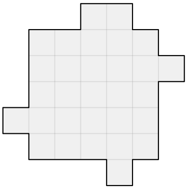


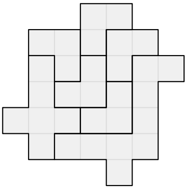
General case

In the **general case**, we consider a dual graph with **cycles**.

Given a region R such that $|R| \equiv 0 \pmod 3$, the following procedure will produce a L-tromino tiling of $R^{\boxplus}$:

- Partition the region R into two or more subregions such that:
  - the number of cells in each subregions is a multiple of three, and
  - the dual graph of each subregion is a tree.



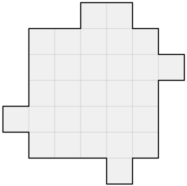


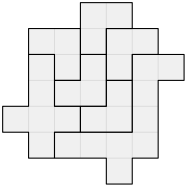
General case

In the **general case**, we consider a dual graph with **cycles**.

Given a region R such that $|R| \equiv 0 \pmod 3$, the following procedure will produce a L-tromino tiling of $R^{\boxplus}$:

- Partition the region R into two or more subregions such that:
  - the number of cells in each subregions is a multiple of three, and
  - the dual graph of each subregion is a tree.





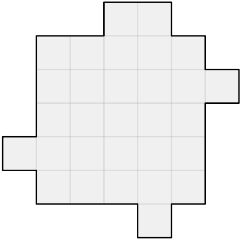
- Then, partition each subregion using the tiling pattern.

Caso general (cont.)

The algorithm to partition a region R is:

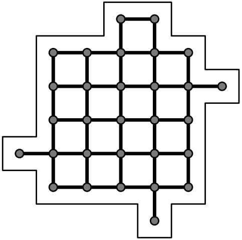
Caso general (cont.)

The algorithm to partition a region R is:



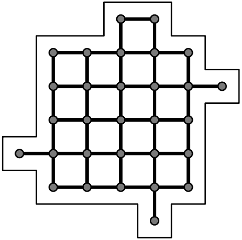
Caso general (cont.)

The algorithm to partition a region R is:



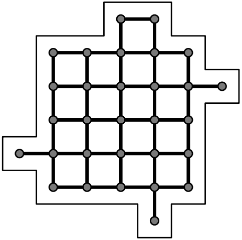
Caso general (cont.)

The algorithm to partition a region R is:



Caso general (cont.)

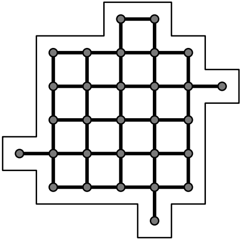
The algorithm to partition a region R is:

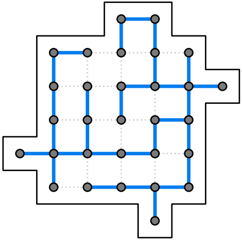


1 Compute any spanning tree of the dual graph G_R.

Caso general (cont.)

The algorithm to partition a region R is:

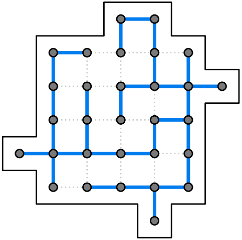




1. Compute any spanning tree of the dual graph G_R.

Caso general (cont.)

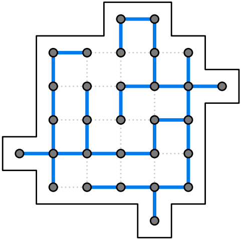
The algorithm to partition a region R is:



1 Compute any spanning tree of the dual graph G_R.

Caso general (cont.)

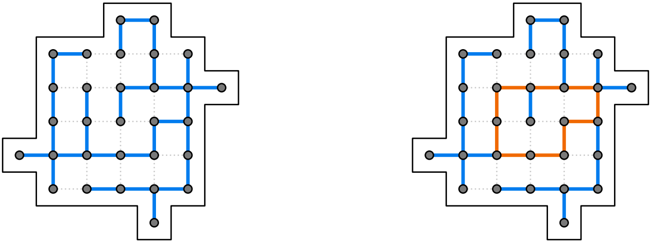
The algorithm to partition a region R is:



1. Compute any spanning tree of the dual graph G_R.
2. Take an edge from G_R and insert it into the spanning tree closing a cycle.

Caso general (cont.)

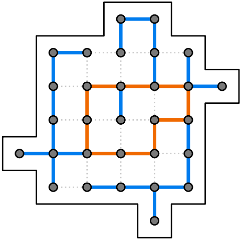
The algorithm to partition a region R is:



1. Compute any spanning tree of the dual graph G_R.
2. Take an edge from G_R and insert it into the spanning tree closing a cycle.

Caso general (cont.)

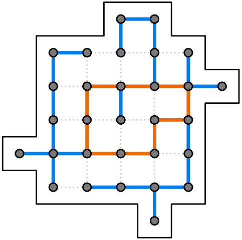
The algorithm to partition a region R is:



1. Compute any spanning tree of the dual graph G_R.
2. Take an edge from G_R and insert it into the spanning tree closing a cycle.

Caso general (cont.)

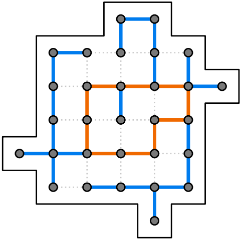
The algorithm to partition a region R is:



1. Compute any spanning tree of the dual graph G_R.
2. Take an edge from G_R and insert it into the spanning tree closing a cycle.
3. Select four different edges arbitrarily from the cycle and find a cut such that

Caso general (cont.)

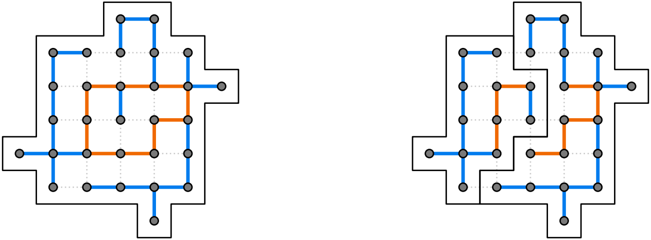
The algorithm to partition a region R is:



1. Compute any spanning tree of the dual graph G_R.
2. Take an edge from G_R and insert it into the spanning tree closing a cycle.
3. Select four different edges arbitrarily from the cycle and find a cut such that
   3.1 Each subregion contains a multiple of three number of cells.

Caso general (cont.)

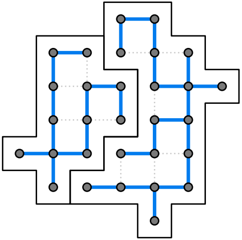
The algorithm to partition a region R is:



1. Compute any spanning tree of the dual graph G_R.
2. Take an edge from G_R and insert it into the spanning tree closing a cycle.
3. Select four different edges arbitrarily from the cycle and find a cut such that
   3.1 Each subregion contains a multiple of three number of cells.

Caso general (cont.)

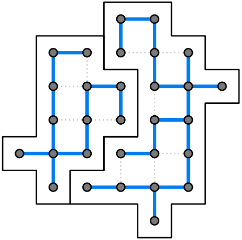
The algorithm to partition a region R is:



1. Compute any spanning tree of the dual graph G_R.
2. Take an edge from G_R and insert it into the spanning tree closing a cycle.
3. Select four different edges arbitrarily from the cycle and find a cut such that
   3.1 Each subregion contains a multiple of three number of cells.

Caso general (cont.)

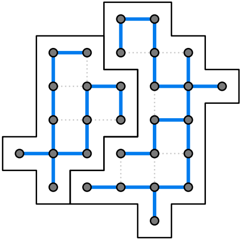
The algorithm to partition a region R is:

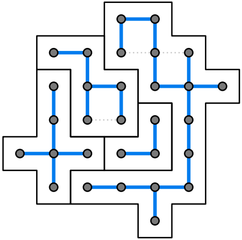


1. Compute any spanning tree of the dual graph G_R.
2. Take an edge from G_R and insert it into the spanning tree closing a cycle.
3. Select four different edges arbitrarily from the cycle and find a cut such that
   3.1 Each subregion contains a multiple of three number of cells.
4. Apply this algorithm for each subregion.

Caso general (cont.)

The algorithm to partition a region R is:





1. Compute any spanning tree of the dual graph G_R.
2. Take an edge from G_R and insert it into the spanning tree closing a cycle.
3. Select four different edges arbitrarily from the cycle and find a cut such that
   3.1 Each subregion contains a multiple of three number of cells.
4. Apply this algorithm for each subregion.


THANK
YOU!