Monomino-Domino Tatami Coverings

Alejandro Erickson

Joint work with Frank Ruskey at The University of Victoria, Canada

December 4, 2013
Durham University, ACiD Seminar.

Japanese Tatami mats

Traditional Japanese floor mats made of soft woven straw.

A 17th Century layout rule:

No four mats may meet.

No four dominoes (mats) may meet

Tatami coverings of rectangles were considered by Mitsuyoshi Yoshida, and Don Knuth (about 370 years later).

215. [*21*] Japanese tatami mats are 1×2 rectangles that are traditionally used to cover rectangular floors in such a way that no four mats meet at any corner. For example, Fig. 29(a) shows a 6×5 pattern from the 1641 edition of Mitsuyoshi Yoshida's *Jinkōki*, a book first published in 1627.

Find all domino coverings of a chessboard that are also tatami tilings.

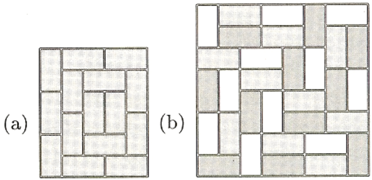


Fig. 29. Two nice examples:
(a) A 17th-century tatami tiling;
(b) a tricolored domino covering.

No four dominoes (mats) may meet

Tatami coverings of rectangles were considered by Mitsuyoshi Yoshida, and Don Knuth (about 370 years later).

215. [*21*] Japanese tatami mats are 1×2 rectangles that are traditionally used to cover rectangular floors in such a way that no four mats meet at any corner. For example, Fig. 29(a) shows a 6×5 pattern from the 1641 edition of Mitsuyoshi Yoshida's *Jinkōki*, a book first published in 1627.

Find all domino coverings of a chessboard that are also tatami tilings.

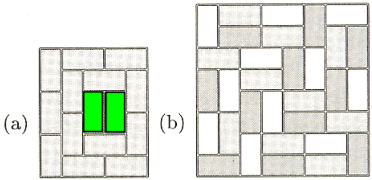


Fig. 29. Two nice examples:
(a) A 17th-century tatami tiling;
(b) a tricolored domino covering.

No four dominoes (mats) may meet

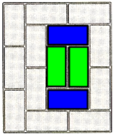
Tatami coverings of rectangles were considered by Mitsuyoshi Yoshida, and Don Knuth (about 370 years later).

215. [*21*] Japanese tatami mats are 1×2 rectangles that are traditionally used to cover rectangular floors in such a way that no four mats meet at any corner. For example, Fig. 29(a) shows a 6×5 pattern from the 1641 edition of Mitsuyoshi Yoshida's *Jinkōki*, a book first published in 1627.

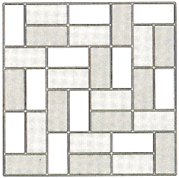
Find all domino coverings of a chessboard that are also tatami tilings.

Fig. 29. Two nice examples:
(a) A 17th-century tatami tiling;
(b) a tricolored domino covering.

(a)



(b)



No four dominoes (mats) may meet

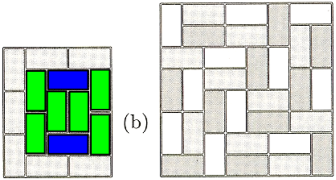
Tatami coverings of rectangles were considered by Mitsuyoshi Yoshida, and Don Knuth (about 370 years later).

215. [*21*] Japanese tatami mats are 1×2 rectangles that are traditionally used to cover rectangular floors in such a way that no four mats meet at any corner. For example, Fig. 29(a) shows a 6×5 pattern from the 1641 edition of Mitsuyoshi Yoshida's *Jinkōki*, a book first published in 1627.

Find all domino coverings of a chessboard that are also tatami tilings.

Fig. 29. Two nice examples:
(a) A 17th-century tatami tiling;
(b) a tricolored domino covering.

(a) (b)



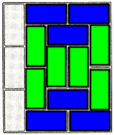
No four dominoes (mats) may meet

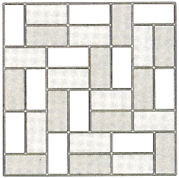
Tatami coverings of rectangles were considered by Mitsuyoshi Yoshida, and Don Knuth (about 370 years later).

215. [*21*] Japanese tatami mats are 1×2 rectangles that are traditionally used to cover rectangular floors in such a way that no four mats meet at any corner. For example, Fig. 29(a) shows a 6×5 pattern from the 1641 edition of Mitsuyoshi Yoshida's *Jinkōki*, a book first published in 1627.

Find all domino coverings of a chessboard that are also tatami tilings.

Fig. 29. Two nice examples:
(a) A 17th-century tatami tiling;
(b) a tricolored domino covering.

(a) 

(b) 

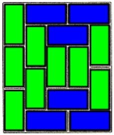
No four dominoes (mats) may meet

Tatami coverings of rectangles were considered by Mitsuyoshi Yoshida, and Don Knuth (about 370 years later).

215. [*21*] Japanese tatami mats are 1×2 rectangles that are traditionally used to cover rectangular floors in such a way that no four mats meet at any corner. For example, Fig. 29(a) shows a 6×5 pattern from the 1641 edition of Mitsuyoshi Yoshida's *Jinkōki*, a book first published in 1627.

Find all domino coverings of a chessboard that are also tatami tilings.

(a)



(b)

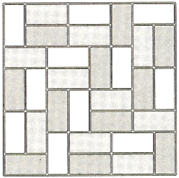


Fig. 29. Two nice examples:
(a) A 17th-century tatami tiling;
(b) a tricolored domino covering.

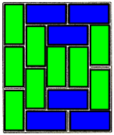
No four dominoes (mats) may meet

Tatami coverings of rectangles were considered by Mitsuyoshi Yoshida, and Don Knuth (about 370 years later).

215. [*21*] Japanese tatami mats are 1×2 rectangles that are traditionally used to cover rectangular floors in such a way that no four mats meet at any corner. For example, Fig. 29(a) shows a 6×5 pattern from the 1641 edition of Mitsuyoshi Yoshida's *Jinkōki*, a book first published in 1627.

Find all domino coverings of a chessboard that are also tatami tilings.

(a)



(b)

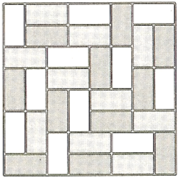
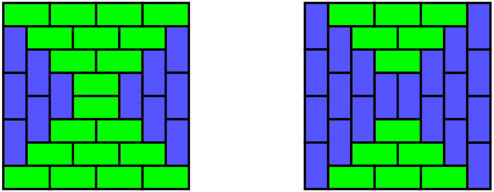


Fig. 29. Two nice examples:
(a) A 17th-century tatami tiling;
(b) a tricolored domino covering.

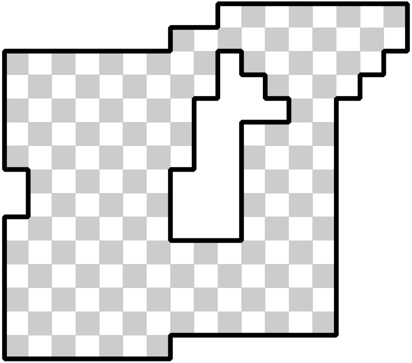
Coverings of the chessboard

There are exactly two

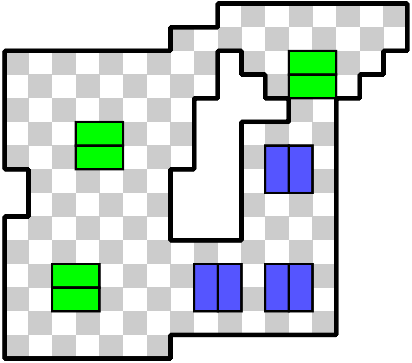
Generalized by Ruskey, Woodcock, 2009, using Hickerson's decomposition.



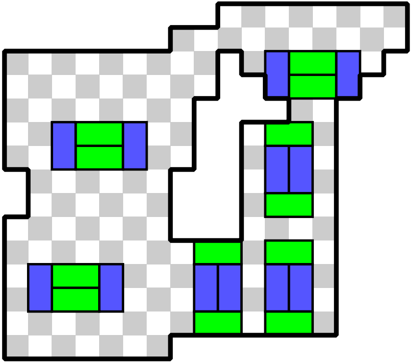
Domino Tatami Covering



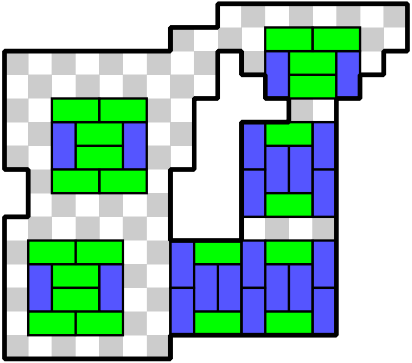
Domino Tatami Covering



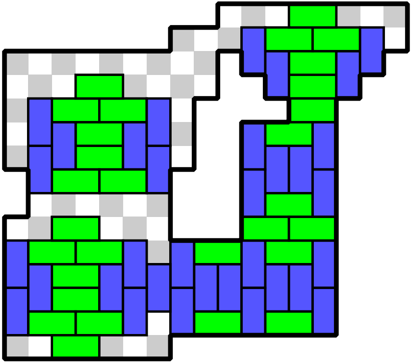
Domino Tatami Covering



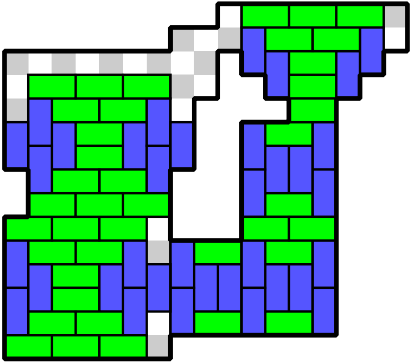
Domino Tatami Covering



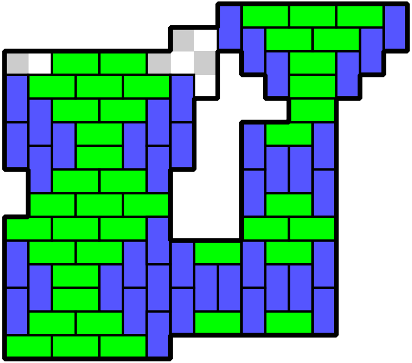
Domino Tatami Covering



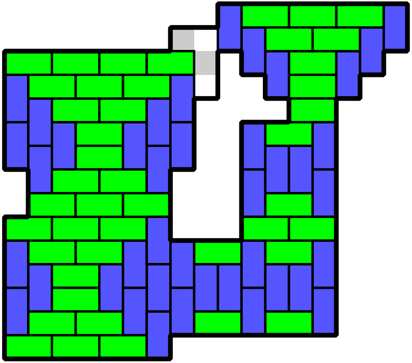
Domino Tatami Covering



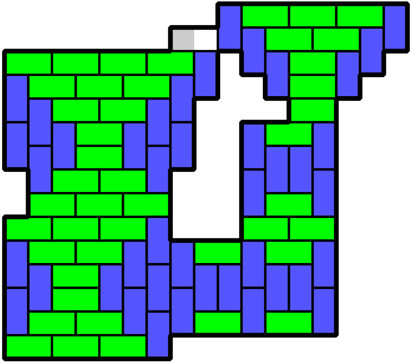
Domino Tatami Covering



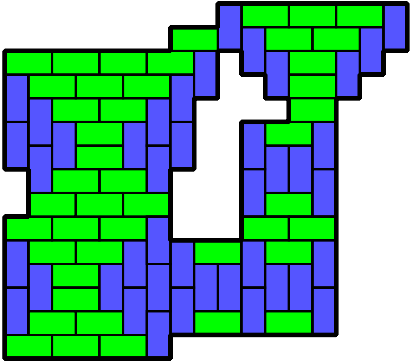
Domino Tatami Covering



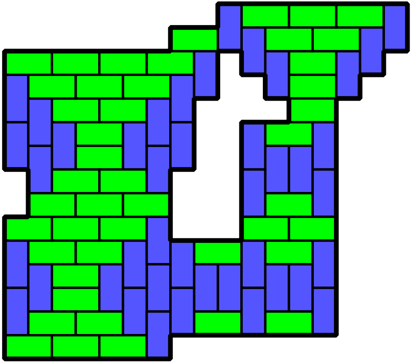
Domino Tatami Covering



Domino Tatami Covering



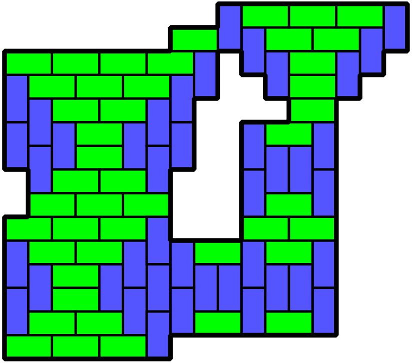
Domino Tatami Covering



(Ruskey, 2009)

INPUT: A region, R, with n grid squares.
QUESTION: Can R be tatami covered with dominoes?

Domino Tatami Covering

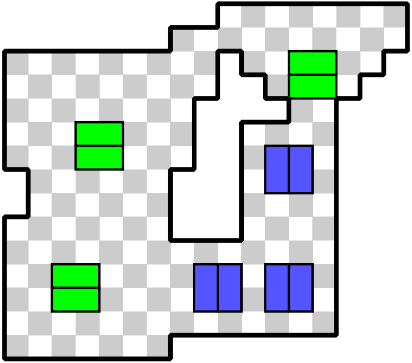


(Ruskey, 2009)

INPUT: A region, R, with n grid squares.
QUESTION: Can R be tatami covered with dominoes?

Is this NP-hard?

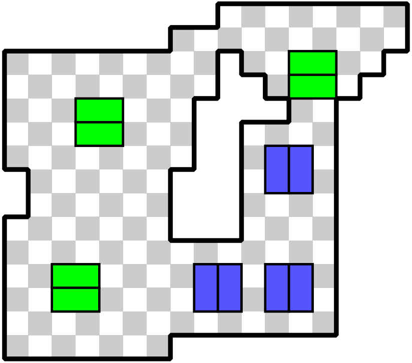
Domino Tatami Covering



(Ruskey, 2009)

INPUT: A region, R, with n grid squares.
QUESTION: Can R be tatami covered with dominoes?

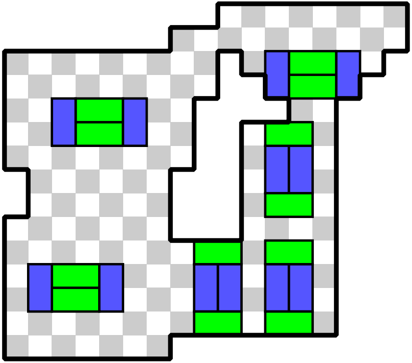
Domino Tatami Covering



(Ruskey, 2009)

INPUT: A region, R, with n grid squares.
QUESTION: Can R be tatami covered with dominoes?

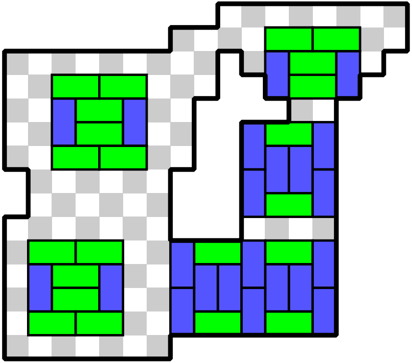
Domino Tatami Covering



(Ruskey, 2009)

INPUT: A region, R, with n grid squares.
QUESTION: Can R be tatami covered with dominoes?

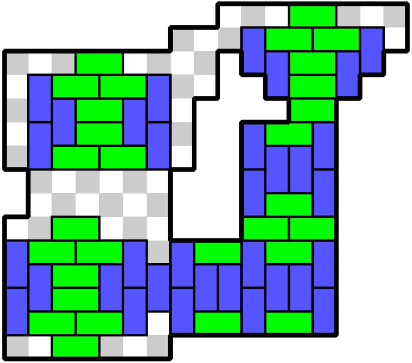
Domino Tatami Covering



(Ruskey, 2009)

INPUT: A region, R, with n grid squares.
QUESTION: Can R be tatami covered with dominoes?

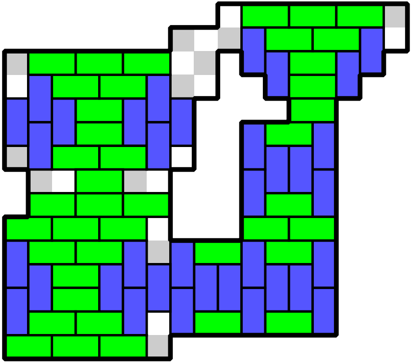
Domino Tatami Covering



(Ruskey, 2009)

INPUT: A region, R, with n grid squares.
QUESTION: Can R be tatami covered with dominoes?

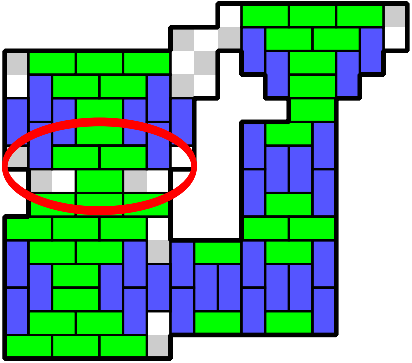
Domino Tatami Covering



(Ruskey, 2009)

INPUT: A region, R, with n grid squares.
QUESTION: Can R be tatami covered with dominoes?

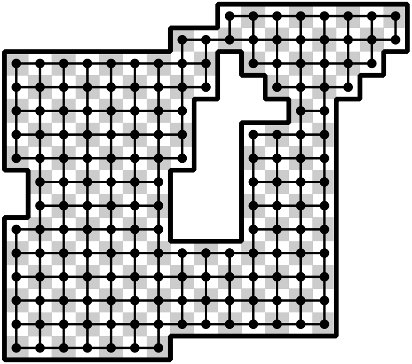
Domino Tatami Covering



(Ruskey, 2009)

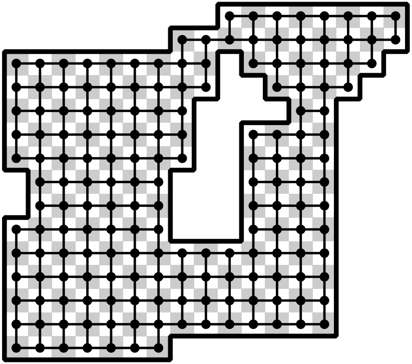
INPUT: A region, R, with n grid squares.
QUESTION: Can R be tatami covered with dominoes?

Domino ~~Tatami~~ Covering is polynomial



A domino covering is a perfect matching in the underlying graph.

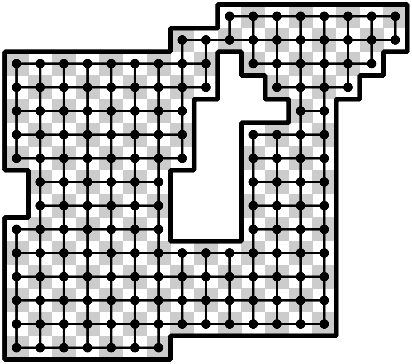
Domino ~~Tatami~~ Covering is polynomial



A domino covering is a perfect matching in the underlying graph.

INPUT: A region, R, with n grid squares.
QUESTION: Can R be ~~tatami~~ covered with dominoes?

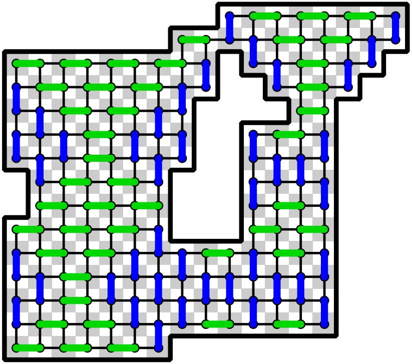
Domino ~~Tatami~~ Covering is polynomial



A domino covering is a perfect matching in the underlying graph.

INPUT: A region, R, with n grid squares.
QUESTION: Can R be ~~tatami~~ covered with dominoes?
This can be answered in $O(n^2)$, since the underlying graph is bipartite.

Tatami coverings as matchings



The tatami restriction is the additional constraint, that every 4-cycle contains a matched edge.

DTC is NP-hard

Domino Tatami Covering

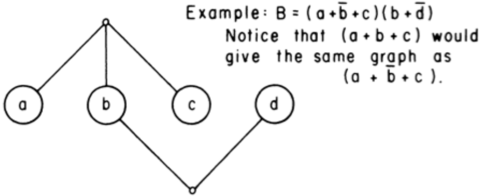
INPUT: A region, R, with n grid squares.
QUESTION: Can R be tatami covered with dominoes?

Theorem (E., Ruskey, 2013)

Domino Tatami Covering is NP-hard.

Planar 3SAT

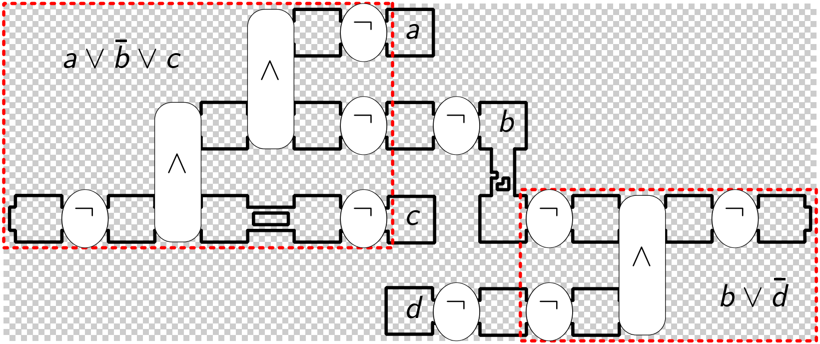
Let ϕ be a 3CNF formula, with variables U, and clauses C. Let $G = (U \cup C, E)$, where $\{u, c\} \in E$ iff one of the literals u or $\bar{u}$ is in the clause c. The formula is *planar* if there exists a planar embedding of G.

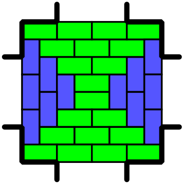


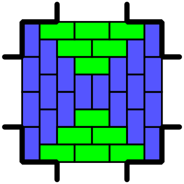
Planar 3SAT is NP-complete (Lichtenstein, 1982).

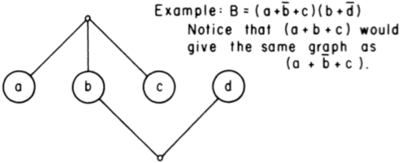
Reduction to Planar 3SAT

Working backwards from the answer...



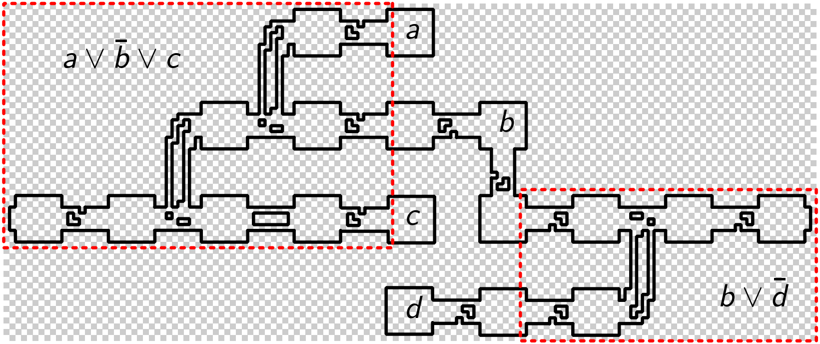


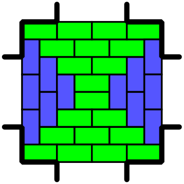


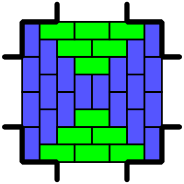


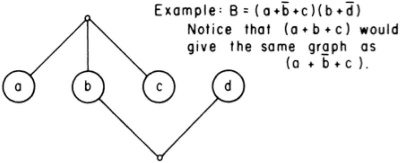
Reduction to Planar 3SAT

Working backwards from the answer...

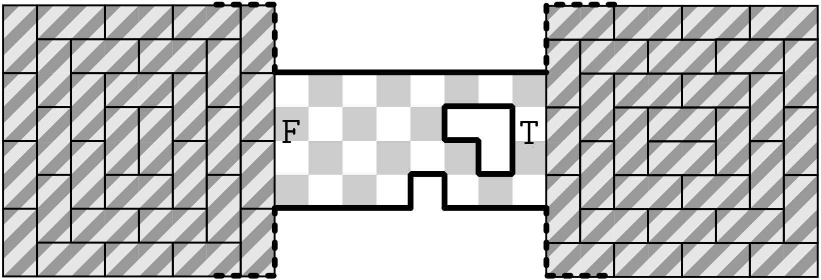








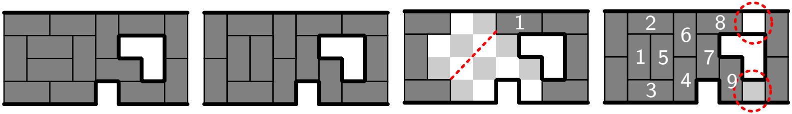
Verify the NOT gate



NOT gate covering can be completed with all "good" signals, but no "bad" signal.

"good"	"bad"
F⟶T	T⟶T
T⟶F	F⟶F

Verify the NOT gate

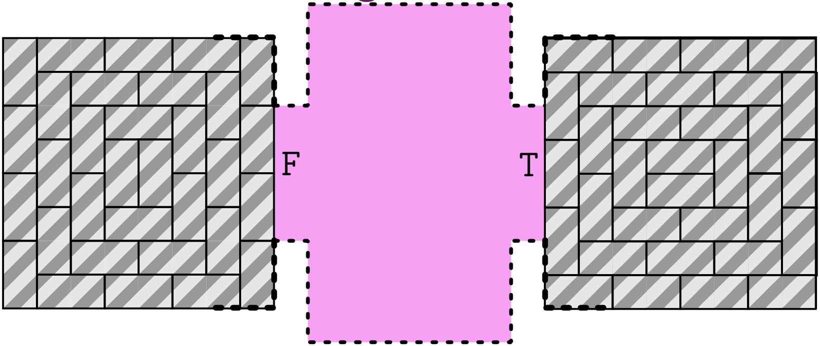


F $\longrightarrow$ T   T $\longrightarrow$ F   F $\longrightarrow$ F   T $\longrightarrow$ T

NOT gate covering can be completed with all "good" signals, but no "bad" signal.

"good"	"bad"
F $\longrightarrow$ T	T $\longrightarrow$ T
T $\longrightarrow$ F	F $\longrightarrow$ F

Search for a NOT gate

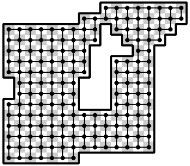


Search for sub-region, R, of the pink area. If R and the chessboards can be covered with all "good" signals, but no "bad" signal, we are done!

"good"	"bad"
$\mathtt{F} \longrightarrow \mathtt{T}$	$\mathtt{T} \longrightarrow \mathtt{T}$
$\mathtt{T} \longrightarrow \mathtt{F}$	$\mathtt{F} \longrightarrow \mathtt{F}$

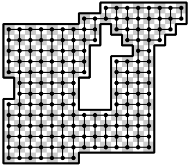
SAT-solvers

- A SAT-solver is software that finds a satisfying assignment to a Boolean formula, or outputs UNSATISFIABLE. We used MiniSAT.



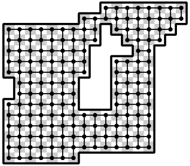
SAT-solvers

- A SAT-solver is software that finds a satisfying assignment to a Boolean formula, or outputs UNSATISFIABLE. We used MiniSAT.
- Given an instance of DTC, the corresponding SAT instance has the edges of the underlying graph G, as variables. A satisfying assignment sets matched edges to TRUE and unmatched edges to FALSE.



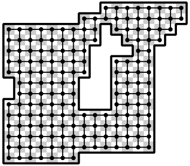
SAT-solvers

- A SAT-solver is software that finds a satisfying assignment to a Boolean formula, or outputs UNSATISFIABLE. We used MiniSAT.
- Given an instance of DTC, the corresponding SAT instance has the edges of the underlying graph G, as variables. A satisfying assignment sets matched edges to TRUE and unmatched edges to FALSE.
- Three conditions must be enforced:



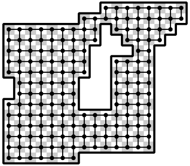
SAT-solvers

- A SAT-solver is software that finds a satisfying assignment to a Boolean formula, or outputs UNSATISFIABLE. We used MiniSAT.
- Given an instance of DTC, the corresponding SAT instance has the edges of the underlying graph G, as variables. A satisfying assignment sets matched edges to TRUE and unmatched edges to FALSE.
- Three conditions must be enforced:
    1. TRUE edges are not incident.



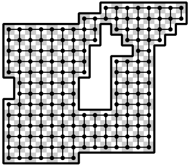
SAT-solvers

- A SAT-solver is software that finds a satisfying assignment to a Boolean formula, or outputs UNSATISFIABLE. We used MiniSAT.
- Given an instance of DTC, the corresponding SAT instance has the edges of the underlying graph G, as variables. A satisfying assignment sets matched edges to TRUE and unmatched edges to FALSE.
- Three conditions must be enforced:
    1. TRUE edges are not incident.
    2. An edge at each vertex is TRUE.



SAT-solvers

- A SAT-solver is software that finds a satisfying assignment to a Boolean formula, or outputs UNSATISFIABLE. We used MiniSAT.
- Given an instance of DTC, the corresponding SAT instance has the edges of the underlying graph G, as variables. A satisfying assignment sets matched edges to TRUE and unmatched edges to FALSE.
- Three conditions must be enforced:
  1. TRUE edges are not incident.
  2. An edge at each vertex is TRUE.
  3. An edge of each 4-cycle is TRUE.



SAT-solvers

We can generate, test cover, and forbid regions with SAT-solvers.

```
4
12
CC#......#CC
CC#......#CC
CC#......#CC
CC#......#CC
2
.A........<>
.V........A.
.A........V.
.V........<>

<>........A.
.A........V.
.V........A.
<>........V.
```

```
2
<>........<>
.A........A.
.V........V.
<>........<>

.A........A.
.V........V.
.A........A.
.V........V.
```

Combine python scripts with the SAT-solver MiniSAT (fast, lightweight, pre-compiled for my system.)

Gadget Search

- request candidate region, R, from MiniSAT, satisfying "good" signals.
- MiniSAT to test each "bad" signal in R.
- if every test UNSATISFIABLE R is the answer!
- Else, "forbid" R in next iteration.

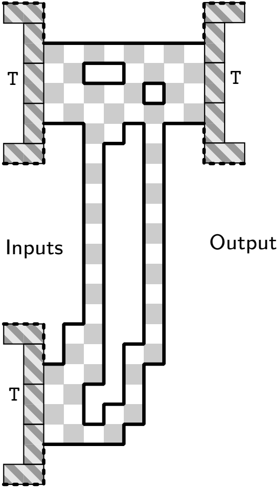
```
numRegions = 0 #count the number of regions we have tried
prevR = []
while(True):
    numRegions += 1
    sp = subprocess.Popen(['./minisat',satinFilename,satoutFilename],stdout=subpr
ocess.PIPE)
    sp.wait()
    if(numRegions%100 == 0):
        print "number of regions checked", numRegions
    if(sp.returncode==10): #satisfied
        g = getSATAssignment(satoutFilename)
        R = g[:r*c] #the region output from last minisat of f
        if(prevR == R):
            quitError('error: two regions the same')
        if(numRegions%100 == 0):
#            print R
            displayRegion(R)
            print "good configurations"
            for k in range(C):
                displayTiling(g,k)
        prevR = R
        rClauses = '' #make clauses to enforce that region
        for _clause in R:
            rClauses = rClauses + str(_clause) + ' 0\n'
        badFlag = False
        for k in range(badC):
            #for each bad configuration, check if it can be completed
            #in the region R.
            badConfig = open(badsatinFilename,'w')
            badConfig.write(badCNFstring[k] + rClauses)
            badConfig.close()
            sp = subprocess.Popen(['./minisat',badsatinFilename,badsatoutFilename
],stdout=subprocess.PIPE)
            sp.wait()
            if(sp.returncode==10):
                badFlag = True
                if(numRegions%100==0):
                    print 'bad configuration'
                    displayTiling(getSATAssignment(badsatoutFilename),0)
                break
            elif(sp.returncode != 20):
                quitError('bad minisat returned bad code: ' + str(sp.returncode))
        if(badFlag == False):
            #we have found a good region!
            print "HORRAY", R
            sys.stdout.flush()
            sys.exit(0)
        #we are going to append a forbidden region to satinFilename
        f = open(satinFilename,'r+')
        #change the first line with the number of clauses
        f.seek(0,0)
        f.write('p cnf ' + str(nGoodVars) + ' ' + str(len(goodClauses)) + '\n')
        #make a clause from the forbidden region
        clause(map(neg,R))
        CNFstring = ''
        for lit in goodClauses[-1]:
            CNFstring = CNFstring + ' ' + str(lit)
        CNFstring = CNFstring + ' 0\n'
        #append this to the end of the file
        f.seek(0,2)
        f.write(CNFstring)
        f.close()
    elif(sp.returncode != 20):
        quitError('good minisat returned bad code: ' + str(sp.returncode))
    else:
        sys.stdout.write('There is no region that satisfies the input.')
        sys.stdout.flush()
        sys.exit(0)
```

Huge search space

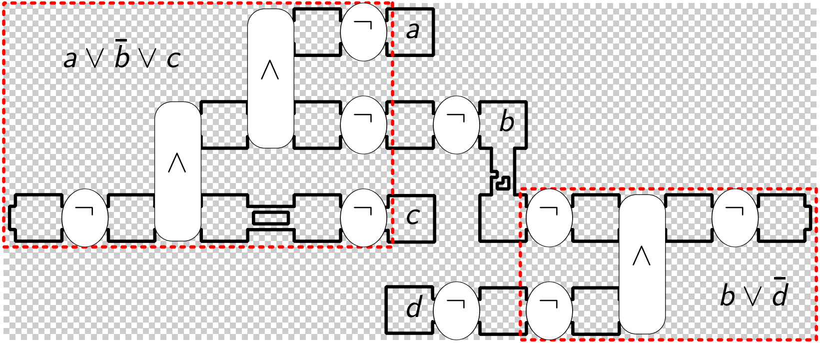
```
CC#....#CC
CC#....#CC
CC#....#CC
CC#..#.#CC
XXX.#..XXX
XXX..#.XXX
CC#.#..XXX
CC#....XXX
CC#....XXX
CC#....XXX
```

Require and forbid some grid squares (`#`, `X`) to be in R to reduce number of disconnected regions. Search a smaller area.

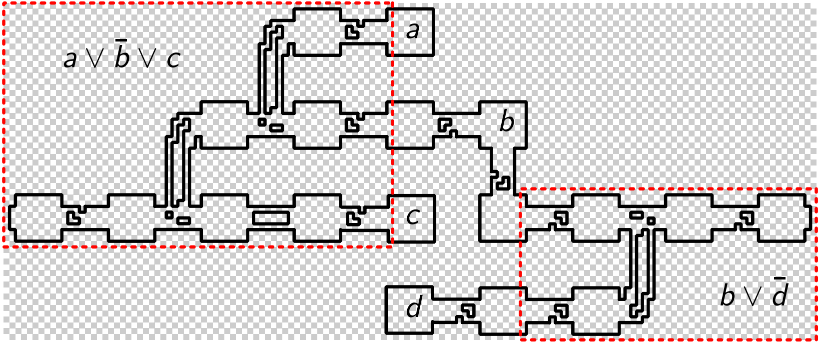
It worked!



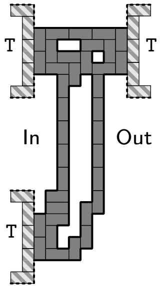
Recall the context



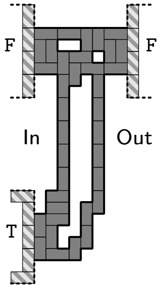
Recall the context



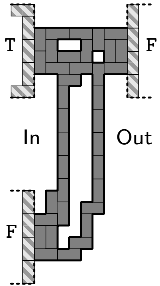
Verifiable by hand



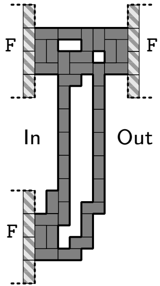
TT$\longrightarrow$T



TF$\longrightarrow$F

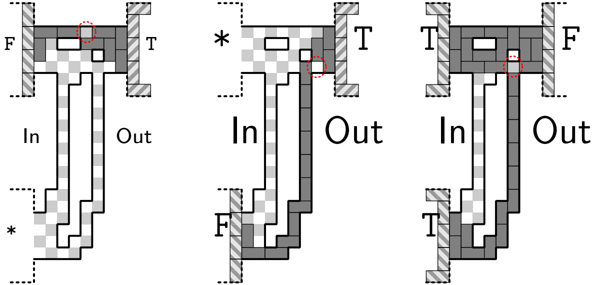


FT$\longrightarrow$F



FF$\longrightarrow$F

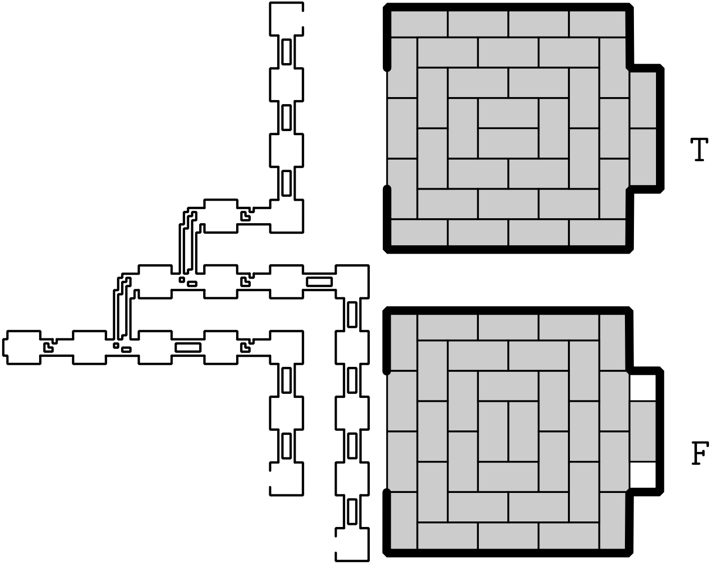
Verifiable by hand



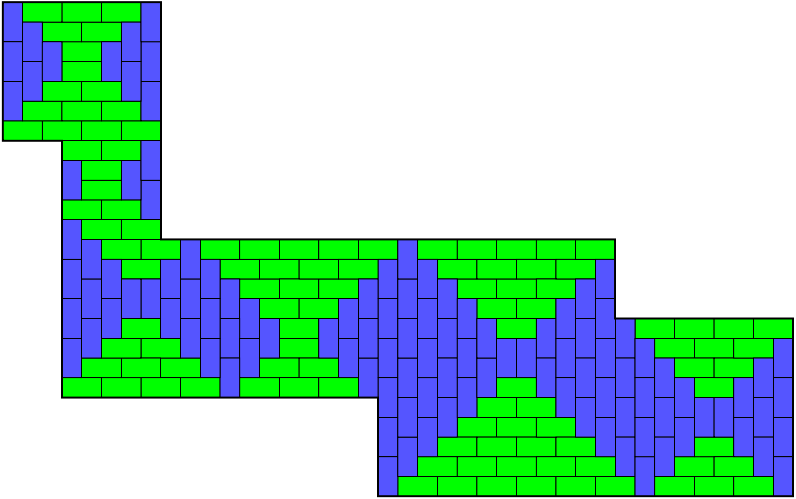
$*\mathtt{F} \longrightarrow \mathtt{T}$ $\mathtt{F}* \longrightarrow \mathtt{T}$ $\mathtt{TT} \longrightarrow \mathtt{F}$

Impossible AND gate coverings, where $*$ denotes F or T.

Testing a clause

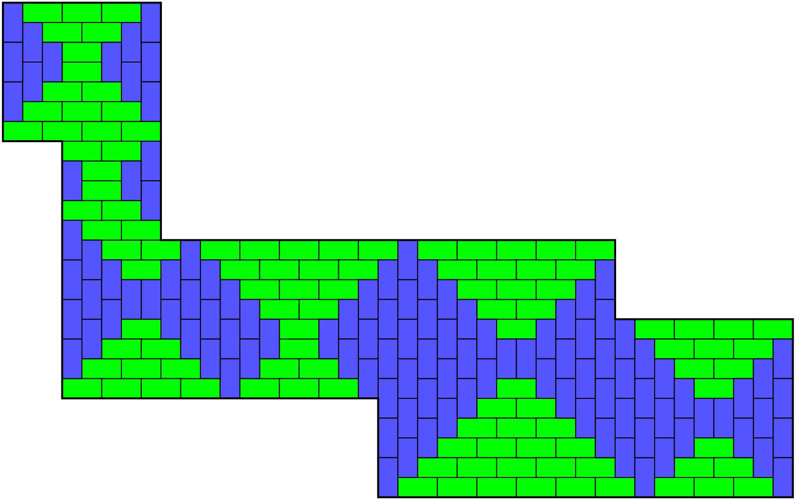


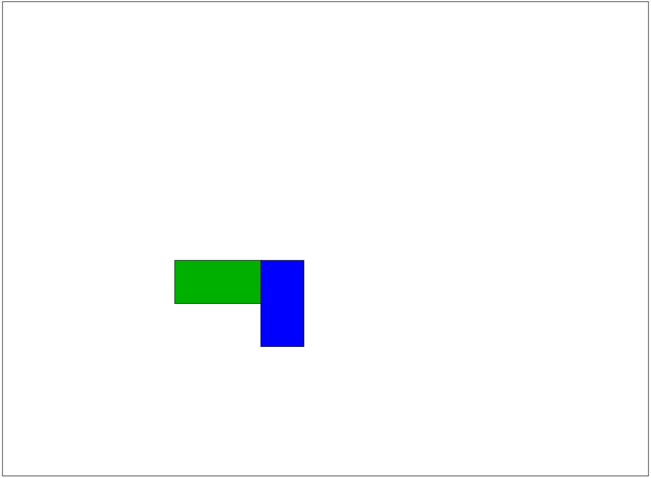
Simply Connected DTC



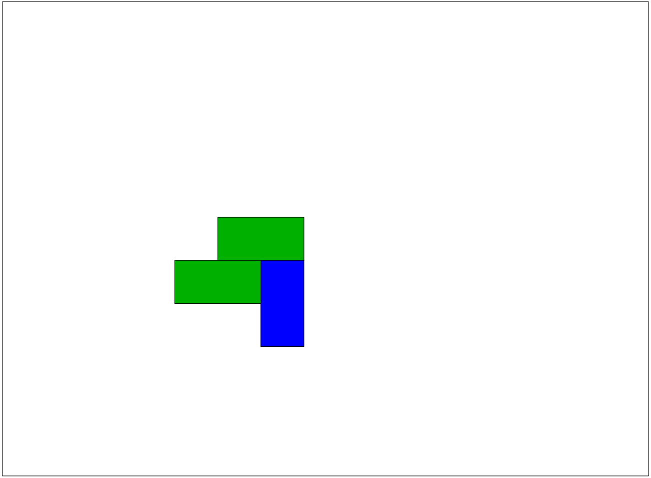
Is DTC NP-hard even if the region is simply connected?

The Structure of Tatami Coverings

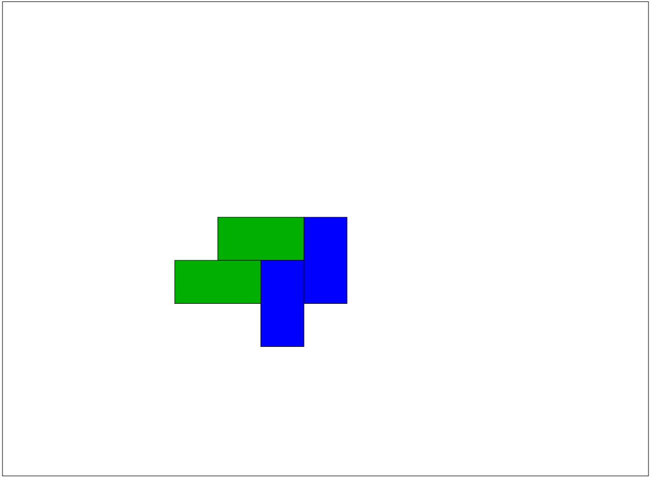




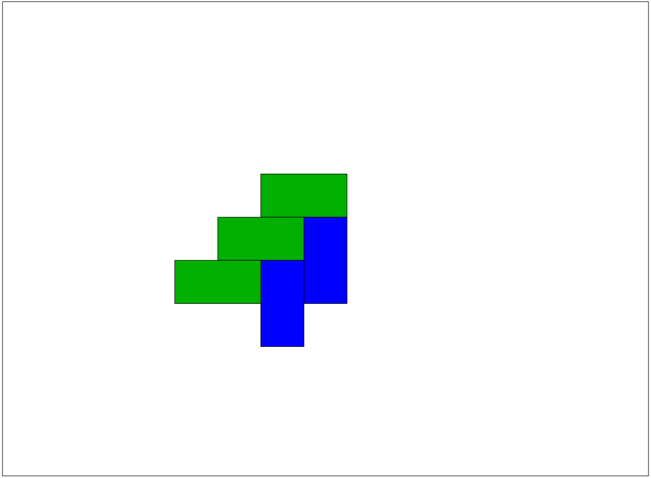
What are the consequences of this arrangement?



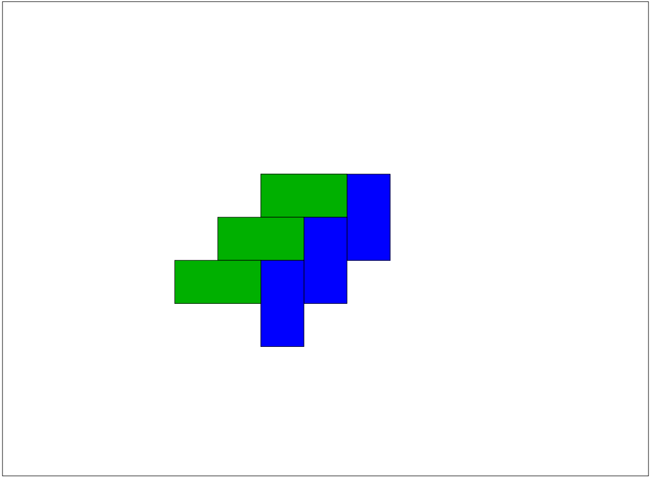
This placement is forced.



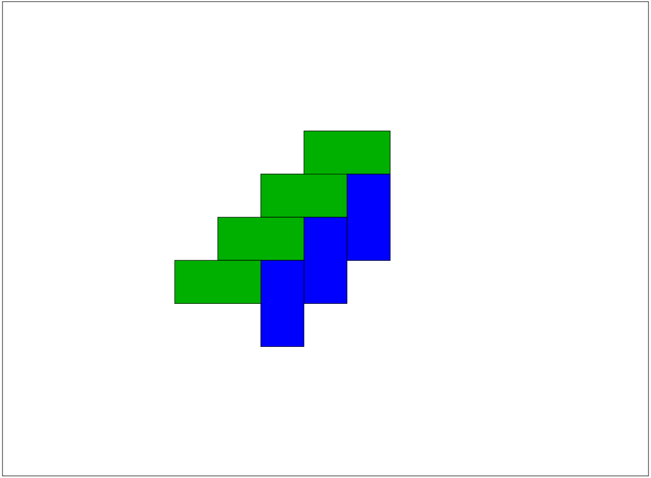
And this placement is also forced.



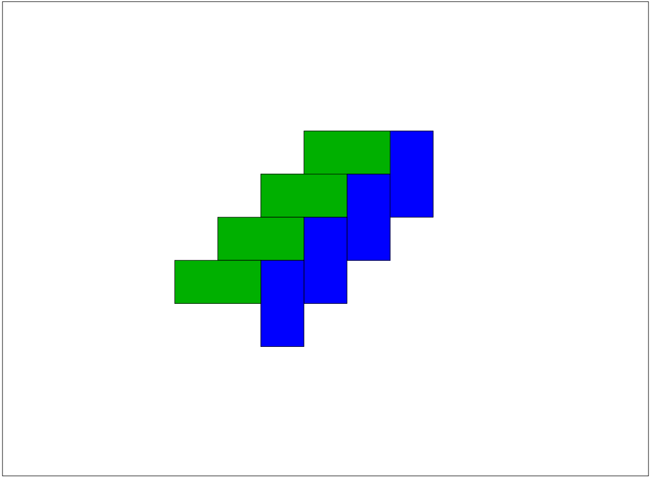
As is this.



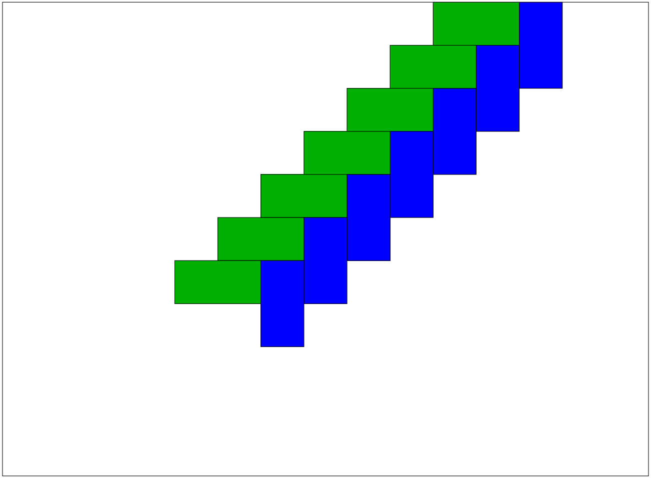
And this.



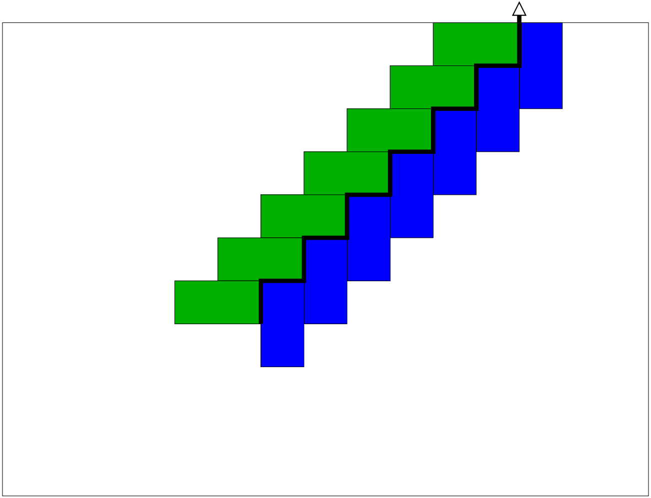
Ditto.



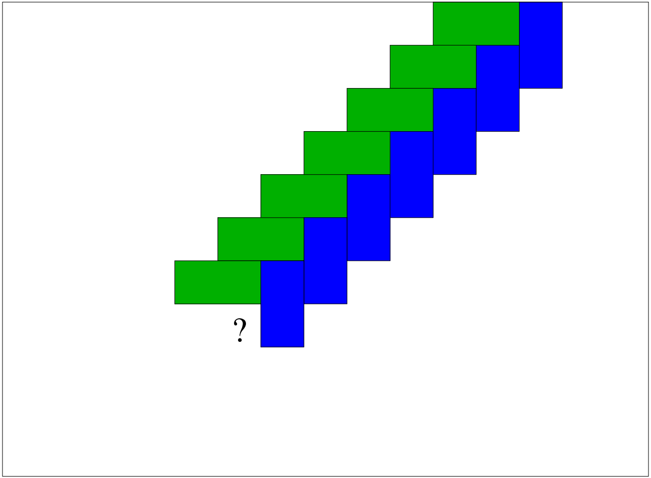
Etc.



Until we reach the boundary.

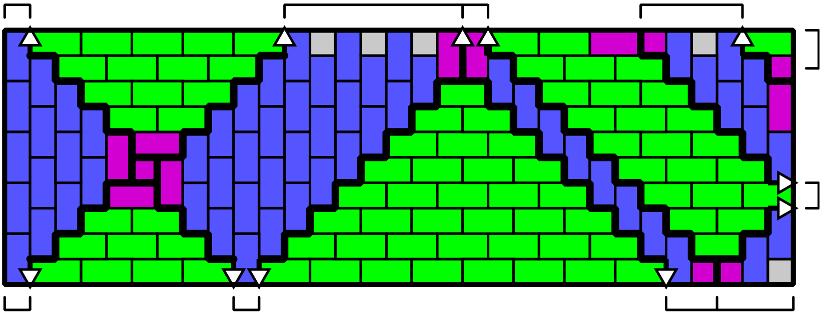


This a **ray**. They can go NE, NW, SE, SW.

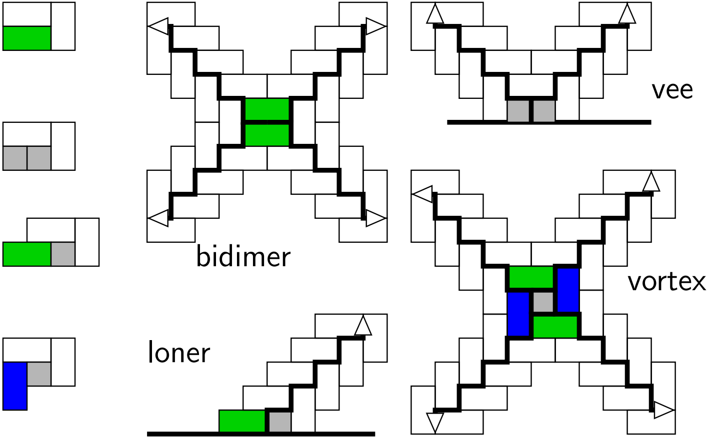


How do rays start? (The question mark.) Not a vertical domino.

Monomino-Domino Tatami Coverings



Monomino-Domino Tatami Coverings



We can enumerate and generate them!

For example, the number of coverings of the $n \times n$ square with n monominoes is $n2^{n-1}$.

We can enumerate and generate them!

For example, the number of coverings of the $n \times n$ square with n monominoes is $n2^{n-1}$.

Generating functions

Let X_k be a set of a_k combinatorial objects for each $k \geq 0$. The *generating function* for $\{X_k\}_{k \geq 0}$ is

$$f(z) = a_0 z^0 + a_1 z^1 + a_2 z^2 + a_3 z^3 + a_4 z^4 + \ldots.$$

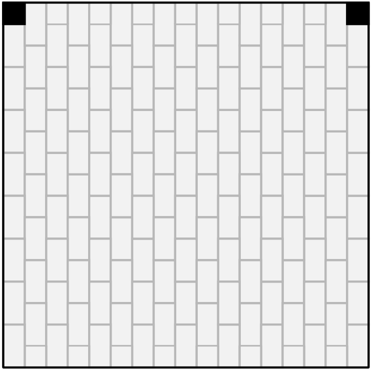
If $a_k = 0$ for some $k \geq K$, then $f(z)$ is a polynomial.

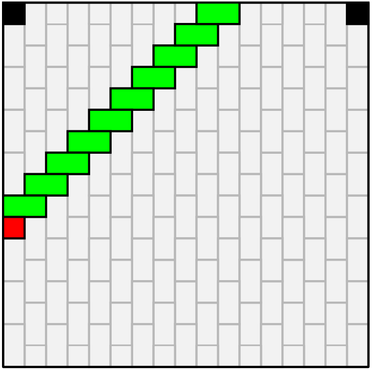
Email from Knuth to Ruskey:

... I looked also at generating functions for the case $m = n$, with respect to horizontal versus vertical dominoes. ... for example when $n = 11$, the generating function for tatami tilings with exactly 11 monominoes and 55 dominoes turns out to be $2(1+z)^5(1+z^2)^2(1-z+z^2)(1+z^4)(1-z+z^2-z^3+z^4)p(z)$, when subdivided by the number of say horizontal dominoes, where $p(z)$ is a fairly random-looking irreducible polynomial of degree 36. One naturally wonders if there's a good reason for so many cyclotomic polynomials in this factorization. ...

Count $n \times n$ coverings with n monominoes, h horizontal dominoes

A *diagonal flip* is an operation on coverings which preserves the tatami restriction, and changes the orientation of some dominoes.



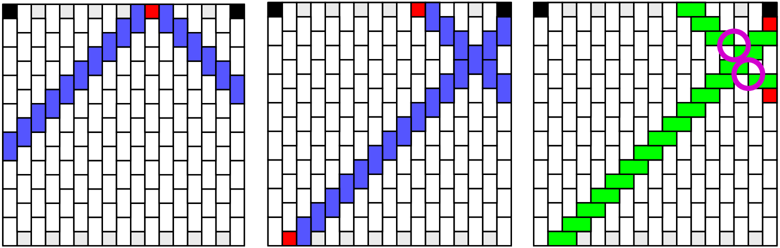


Count $n \times n$ coverings with n monominoes, h horizontal dominoes

Good: Every covering is obtainable via a sequence of diagonal flips.

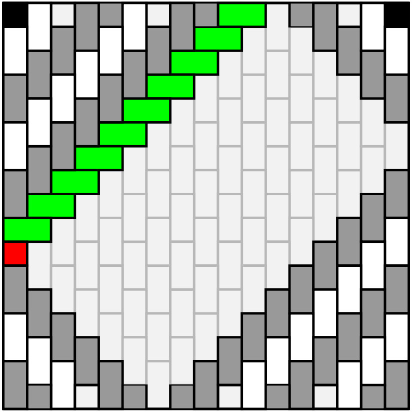
Bad: Conflicting flips complicate enumeration.

Solution: Equivalence classes with independent flips.



Classes of coverings with independent diagonal flips

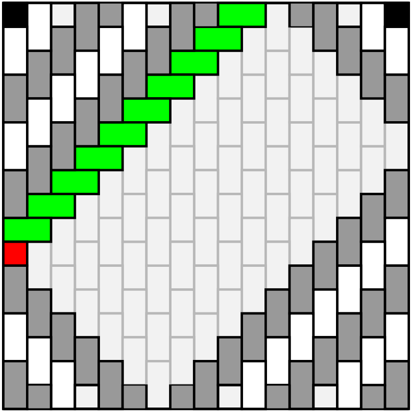
Longest "flipped" diagonal in green.



- Flippable diagonals are independent of one another.

Classes of coverings with independent diagonal flips

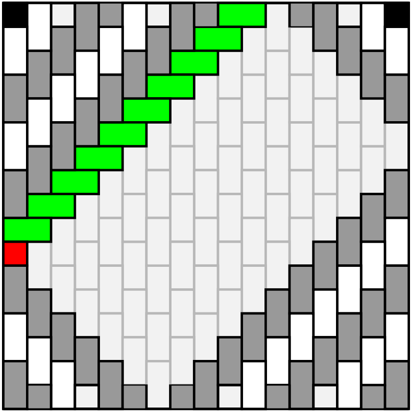
Longest "flipped" diagonal in green.



- Flippable diagonals are independent of one another.
- Can change s to s in a predictable way.

Classes of coverings with independent diagonal flips

Longest “flipped” diagonal in green.



- Flippable diagonals are independent of one another.
- Can change s to s in a predictable way.
- Consider k-sum subsets of multiset $\{{\color{purple}1, 2, \ldots, 9}, {\color{blue}1, 2, \ldots, 5}\}$

Generating polynomial (E.,Ruskey)

$S_n(z) = \prod_{k=1}^{n}(1+z^k)$ "generates" k-sum subsets of $\{1, 2, ..., n\}$. (algebra omitted...) Let

$$VH_n(z) = P_n(z) \prod_{j \geq 1} S_{\lfloor \frac{n-2}{2j} \rfloor}(z)$$

where $P_n(z)$ is a polynomial. For odd n, the kth coefficient of $VH_n(z)$ is the number of $n \times n$ coverings with n monominoes and h horizontal dominoes.

Generating polynomial (E.,Ruskey)

$S_n(z) = \prod_{k=1}^{n}(1+z^k)$ "generates" k-sum subsets of $\{1, 2, ..., n\}$. (algebra omitted...) Let

$$VH_n(z) = P_n(z) \prod_{j \geq 1} S_{\lfloor \frac{n-2}{2j} \rfloor}(z)$$

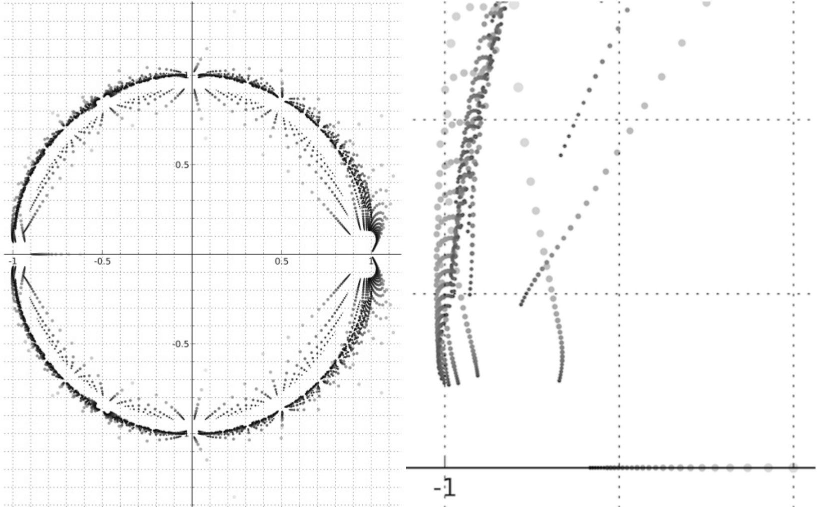
where $P_n(z)$ is a polynomial. For odd n, the kth coefficient of $VH_n(z)$ is the number of $n \times n$ coverings with n monominoes and h horizontal dominoes.

Cyclotomic polynomial factors:

$$S_n(z) = \prod_{j=1}^{n} \phi_{2j}^{\lfloor \frac{n+j}{2j} \rfloor}(z)$$

"... where $p(z)$ is a fairly random-looking irreducible polynomial..."

Not as random as it looks. e.g. complex zeros of $P_n(z)$. n is odd. Large n plotted with darker, smaller dots.



"... where $p(z)$ is a fairly random-looking irreducible polynomial..."

Not as random as it looks.

- $\deg(P_n(z)) = \sum_{k=1}^{n-2} Od(k)$, where $Od(k)$ is the largest odd divisor of k.

"... where $p(z)$ is a fairly random-looking irreducible polynomial..."

Not as random as it looks.

- $\deg(P_n(z)) = \sum_{k=1}^{n-2} Od(k)$, where $Od(k)$ is the largest odd divisor of k.
- $P_n(1) = n2^{\nu(n-2)-1}$, where $\nu(k)$ is the binary weight of k.

"... where $p(z)$ is a fairly random-looking irreducible polynomial..."

Not as random as it looks.

- $\deg(P_n(z)) = \sum_{k=1}^{n-2} Od(k)$, where $Od(k)$ is the largest odd divisor of k.
- $P_n(1) = n2^{\nu(n-2)-1}$, where $\nu(k)$ is the binary weight of k.
- ... etc. But, is $P_n(z)$ irreducible?

"... where $p(z)$ is a fairly random-looking irreducible polynomial..."

Not as random as it looks.

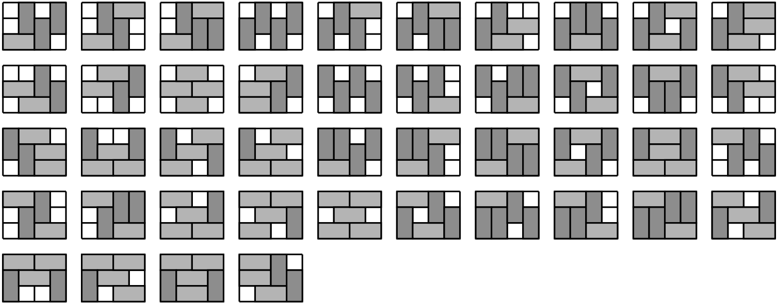
- $\deg(P_n(z)) = \sum_{k=1}^{n-2} Od(k)$, where $Od(k)$ is the largest odd divisor of k.
- $P_n(1) = n2^{\nu(n-2)-1}$, where $\nu(k)$ is the binary weight of k.
- ... etc. But, is $P_n(z)$ irreducible?
- Can it be computed without dividing

$$VH_n(z) = P_n(z) \prod_{j \geq 1} S_{\lfloor \frac{n-2}{2^j} \rfloor}(z).$$

Open problems

- Efficient combinatorial generation

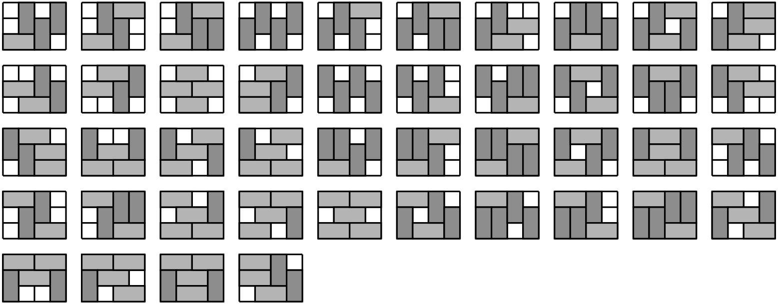
All tatami coverings of the 3×4 grid.



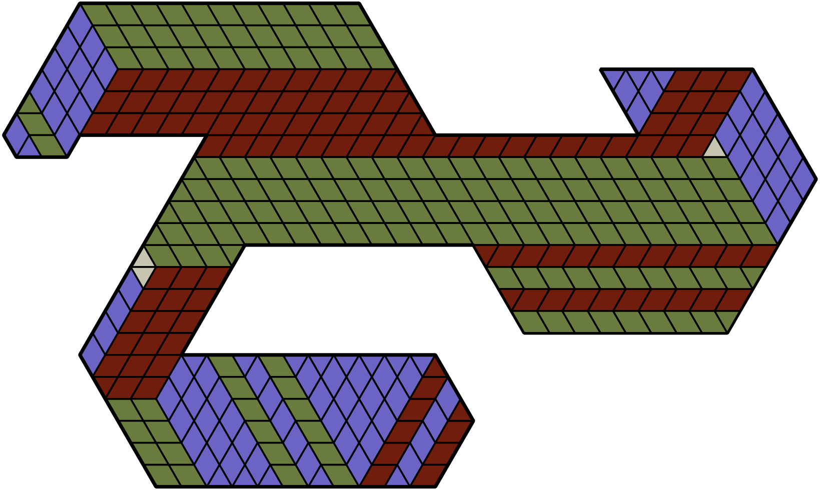
Open problems

- Efficient combinatorial generation
- Generalizations to other tiles

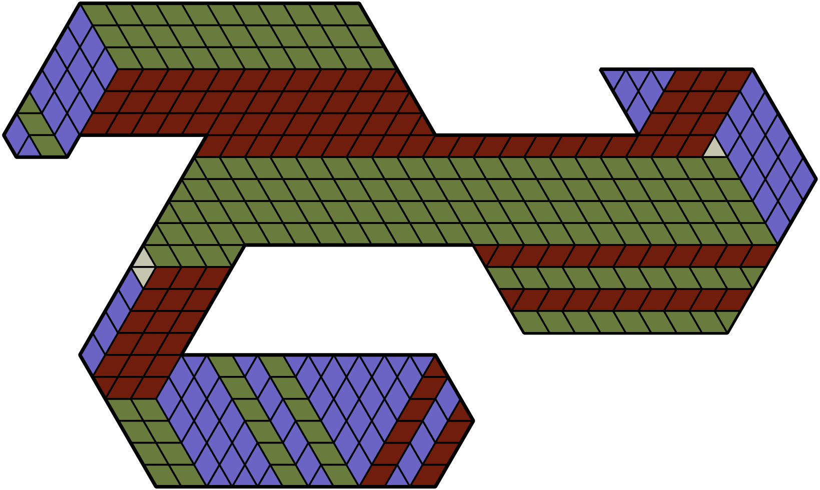
All tatami coverings of the 3×4 grid.



Lozenge 5-Tatami Covering

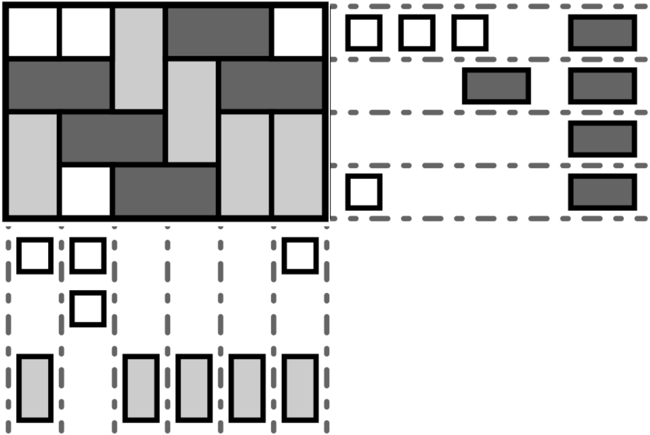


Lozenge 5-Tatami Covering



Is Lozenge 5-Tatami Covering NP-hard?

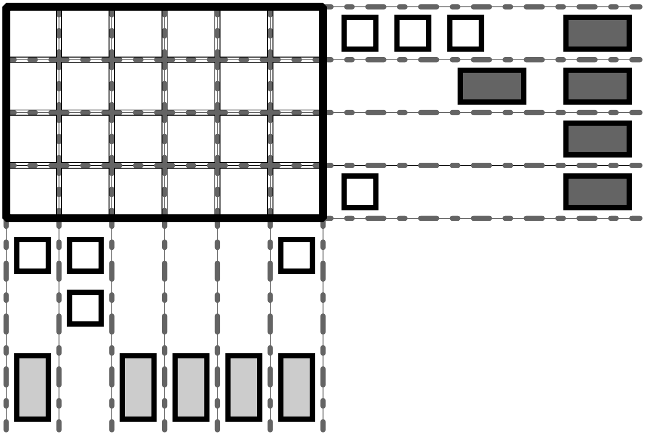
Tomoku!



INSTANCE: A $r \times c$ grid and tiles completely contained in each row and column.

QUESTION: Is there a tatami covering of this grid with these row and column projections?

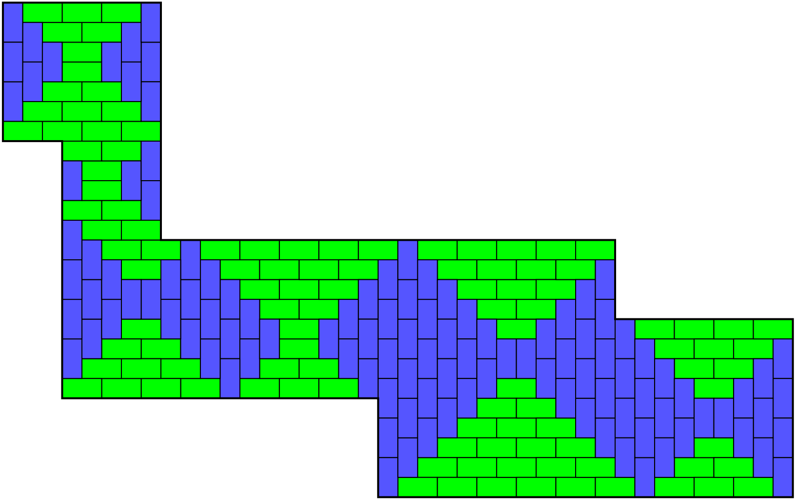
Tomoku!



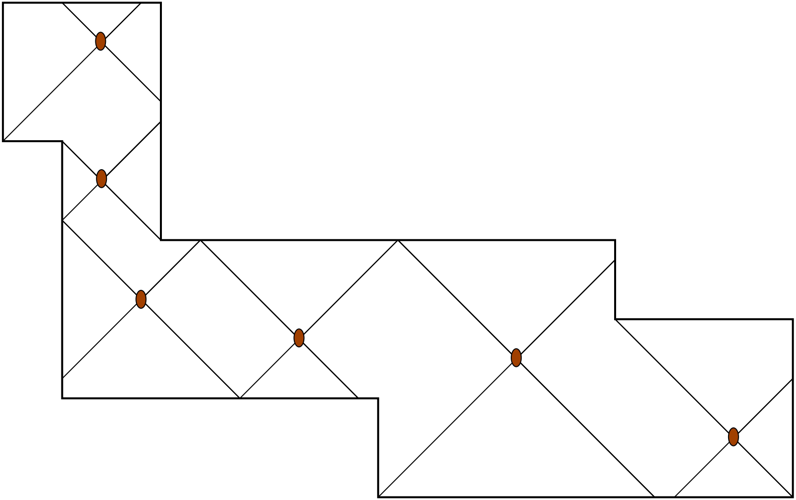
INSTANCE: A $r \times c$ grid and tiles completely contained in each row and column.

QUESTION: Is there a tatami covering of this grid with these row and column projections?

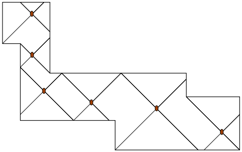
Water Strider Problem



Water Strider Problem



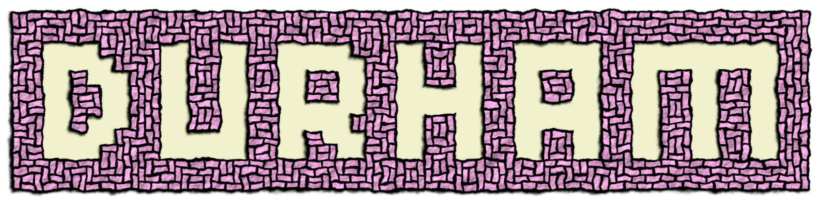
Water Strider Problem



INSTANCE: A rectilinear region, R, with n segments, and vertices in $\mathbb{R}^2$.

QUESTION: Is there a configuration of at most k water striders, such that no two water striders intersect, and no more water striders can be added?

Thank you



Thanks also to Bruce Kapron and Don Knuth. Part of this research was conducted at the 9th McGill-INRIA Workshop on Computational Geometry.

Slides at `alejandroerickson.com`