(the number of) DOMINO TILINGS

Marko Puza

adapted from the (brilliant) text of Brendan W. Sullivan
revised by Simon Holm

Festival of Creative Learning, 21/02/2017

Introduction

The problem around which the whole of today will revolve is simple to state and hard to solve.

Rather than trying to solve it directly, we encourage you to take it step by step, work in groups and co-operate and simply explore the interesting context of the problem. We do not expect anyone of you to have found the solution at the end of the day, or to have gone through all of the following pages. However, we do want you to produce a group report - a coherent and self-contained document in which you describe your findings, what made or did not make sense, and what you considered curious etc. It can contain known theorems and results, your own observations, proofs, pictures, graphs, computer programs or whatever you find suitable to express yourselves with. The report might become a piece of work that you can be proud of! After the event, we will read your reports and try to give constructive feedback, with a small prize for the best report!

To help you with the exploration, you are given this guidance document - even though it is unlikely that you will finish all of it, you might want to at least skim it through. Essentially, it describes a step-by-step derivation of a simple algorithm that solves the problem. The document consists of a series of suggestions of what you might do, but these are exactly what they say - suggestions. You are free to go into whatever direction you find appropriate. You are allowed to use the internet to find whatever you need, and you are also encouraged to do so.

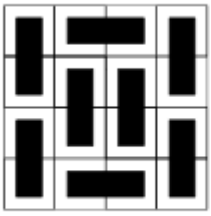
So without further ado, let's get going! And remember, the most important thing is: have fun!

The Problem

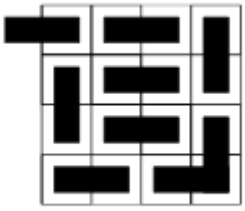
Problem (The Problem)**.** Given an $m \times n$ rectangular grid (board), in how many ways can we tile it by dominoes of size 2×1?[1]

Remark 1. By tiling a grid we mean placing dominoes onto the grid in such a way that they cover all of it, while no domino sticks out of the grid. Examples of a tiling and a non-tiling can be found below.

A domino tiling of a 4 × 4 board



A non-tiling of a 4 × 4 board



Remark 2. From now on, we will denote the number of tilings of a board $m \times n$ by $F(m, n)$.

Right of the bat I will tell you that the problem in general is not simple at all - it does have a closed-form solution, though. The next sections will go into direction of finding it.

[1] These can be, of course, placed horizontally or vertically.

Exploration 1 [Empirical experiments, Big-$\mathcal{O}$]

To start with, let us try to get a feel of how the number of tilings actually behaves. This is the part where you will have the most freedom and you are encouraged to get creative!

Suggestion 1. What are the sufficient and necessary conditions for a tiling to exist?

We have prepared a python script that, given as an input the integers m and n, computes[2] the number of tilings of the board $m \times n$. It can be found here:
`https://github.com/markopuza/Problem-Solving-Maths-Group/blob/master/PSEvent/Count_tilings.py`

If you currently cannot run Python on your laptop, you may still run the script using an online tool (such as `https://repl.it`) But beware - laptops have limited capacity (mine can only handle numbers up to around 18×18)! Hence, be careful to use too large numbers.

Suggestion 2. Try to play around with different sizes of the board. How fast does the number of tilings get very large?

The *Big $\mathcal{O}$ notation*, roughly speaking, describes how fast a function grows. More precisely, we say that a function f is *big-$\mathcal{O}$* of a function g (written $f = \mathcal{O}(g)$) if

$$\text{there are constants } c, x_0 \in \mathbb{R} \text{ such that } |f(x)| \leq c|g(x)| \text{ for all } x \geq x_0$$

This means that g grows fundamentally faster than f (and it somehow provides an upper bound on the growth of f). As an example, we have

$$1 = \mathcal{O}(log(n)),\ log(n) = \mathcal{O}(n),\ n = \mathcal{O}(n \cdot log(n)),\ n \cdot log(n) = \mathcal{O}(n^2),\ n^{100} = \mathcal{O}(2^n)$$

Suggestion 3. Can you estimate how fast F grows? Perhaps producing a graph could give you a nice visualisation.

Suggestion 4. How could you go around designing a program that counts the number of tilings? How efficient would it be?

Suggestion 5. Explore what happens if there are *holes* in your board (i.e. squares that are obstructed and cannot be tiled).

Remark 3. If you wish to count the ways that a particular board with holes can be tiled, consult me and I will show you how to do it with the script.

Suggestion 6. Do holes in different places of the board have different influences on the number of tilings?

Exploration 2 [Recurrence]

The Problem somehow looks like it could be coped using a *divide-and-conquer* approach (that is, divide your problem into smaller sub-problems, solve each of these recursively, and combine the results to get your answer). For example, in the special case of $n = 1$, we can clearly see that $F(1, n) = F(1, n - 2)$.

Suggestion 7. What happens in the special case $n = 2$? Find the recurrence relation of the number of tilings.

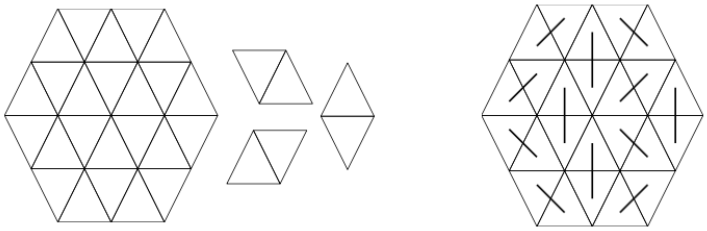
Suggestion 8. What about the special case $n = 3$? How much more difficult is it to derive this recurrence relation?

Suggestion 9. Pick your favourite board size $x \times y$ up to, say, 5×5 and try to write $F(x, y)$ only in terms of F's of smaller boards.

Suggestion 10. Explain why this is a feasible/unfeasible approach.

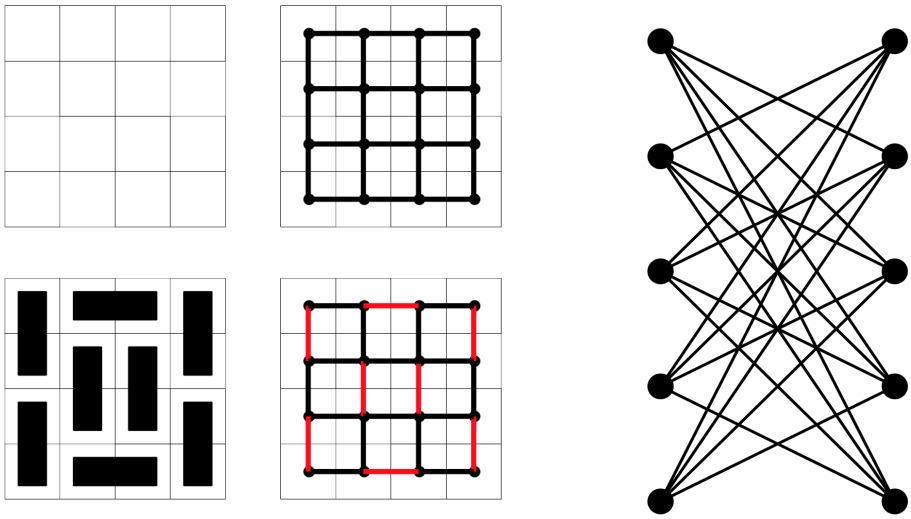
Suggestion 11. Explore recurrences on other types of tilings. An example of other tiling may be a hexagonal tiling, as below:

[2] And it does so directly - we are not using any clever formula here.



Exploration 3 [Graphs]

Let us now change our approach to the problem, quite drastically, using graphs. One may notice that the board can be represented simply as a graph, with each square being a *vertex* and *edges* corresponding to two squares being adjacent.

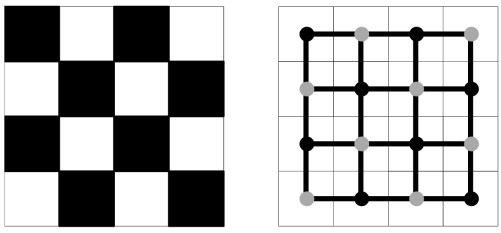


Definition 1. A **perfect matching** of a graph is a selection of edges that covers every vertex exactly once. On the left picture above, this corresponds to the set of red edges.

Definition 2. A **bipartite** graph is one that can have vertices partitioned into two disjoint sets, such that the edges only go between them (no edges within the same set). See the right picture above.

Definition 3. We will say that a bipartite graph is **complete** if it contains all of the possible edges between its two vertex sets.

Suggestion 12. What does the perfect matching have to do with The Problem?



Suggestion 13. What happens if we colour the board as a chessboard? What does a bipartite graph have to do with this?

Suggestion 14. Are there any other relevant colourings that provide a nice insight into The Problem?

Suggestion 15. Could the graph-theoretical insight provide a clearer way to write a program which computes F?

Let us now colour the board as a chessboard for real. From now on, we will denote the set of vertices corresponding to white squares W and the set of vertices corresponding to black squares B.

Suggestion 16. What is the cardinality of these sets (denote it as N)? Is the underlying bipartite graph complete?

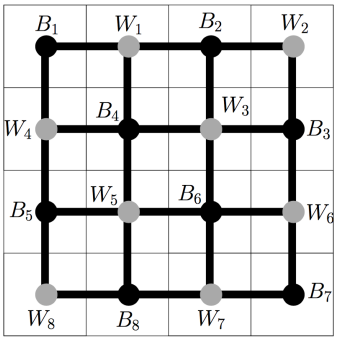
We will also number the vertices of B and W, so we can reference them easily. Say $B = \{b_i : 1 \leq i \leq N\}$ and $W = \{w_j : 1 \leq j \leq N\}$. Now we are ready to dive into the connection that The Problem has with linear algebra!

Exploration 4 [Adjacency matrices]

An **adjacency matrix** of a bipartite graph G with N vertices in each part is the $N \times N$ square matrix given by the entries:

$$a_{ij} = \begin{cases} 1 & \text{if } (b_i, w_j) \text{ is an edge in } G \\ 0 & \text{otherwise} \end{cases}$$

This deserves an example:



$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Suggestion 17. Given a bipartite graph, can you find its adjacency matrix? Given an adjacency matrix, can you find its bipartite graph?

Definition 4. A **permutation** π of K elements is a bijective map $\pi : \{1, 2, \ldots, K\} \to \{1, 2, \ldots, K\}$. There are $K!$ distinct permutations of K elements.

Suggestion 18. B and W both have N labeled vertices. Can we encode a perfect matching by a permutation of $\{1, 2, \ldots, N\}$?

Suggestion 19. Step back a little and try to explain in simple terms what the connection is between a given tiling of a board and a permutation of $\{1, 2, \ldots, N\}$.

Suggestion 20. Given a permutation π, does this always correspond to a perfect matching? What are the necessary conditions that the adjacency matrix A must satisfy for π to represent a perfect matching?

Whether you managed to get an insight to these conditions or not, I will have to include the answer now - these results are really nice.

A permutation π corresponds to a perfect matching if all the edges $(i, \pi(i))$ are present in G. Or, in other words, that all the entries $a_{i,\pi(i)}$ in the adjacency matrix A are 1!

Suggestion 21. Convince yourself that this is the same as requiring

$$\Pi_{i=1}^{N} a_{i,\pi(i)} = a_{1,\pi(1)} \cdot a_{2,\pi(2)} \cdots a_{N,\pi(N)} = 1$$

Suggestion 22. Convince yourself that $F(m, n)$ is equal to the number of permutations π with $a_{i,\pi(i)} = a_{1,\pi(1)} \cdot a_{2,\pi(2)} \cdots a_{N,\pi(N)} = 1$.

Definition 5. The **set of all permutations** of N elements is usually referenced to as S_n (this is the *symmetric group* on N elements).

Suggestion 23. Try to put your findings into a compact math notation! That is, verify that $F(m, n) = \sum_{\pi \in S_N} \prod_{i=1}^{N} a_{i,\pi(i)}$.

Remark 4. Note here that when summing over all permutations π, all those terms that do not correspond to a perfect matching will be 0.

Exploration 5 [Permanents, Determinants, Computational Complexity]

Definition 6. The **permanent** of a $N \times N$ square matrix A is

$$\text{per}(A) = \sum_{\pi \in S_N} \prod_{i=1}^{N} a_{i,\pi(i)}.$$

Does this look familiar?

Definition 7. The **parity** of a permutation π is either *even* or *odd*, depending on the number of *transpositions* (exchange of 2 elements) needed to turn π into identity. This corresponds to $sgn(\pi) = \pm 1$.

Definition 8. The **determinant** of A is

$$\det(A) = \sum_{\pi \in S_n} sgn(\pi) \prod_{i=1}^{N} a_{i,\pi(i)}.$$

At this point it is clear that finding F is equivalent to finding $\text{per}(A)$! The question is how difficult this is. To get an idea, let us briefly discuss computational complexity.

Definition 9. A **decision problem** is a set D and a query $Q \subseteq D$. The problem is: given $d \in D$, do we have $d \in Q$?

Remark 5. Indeed, a decision problem is simply the problem of deciding whether an element of a set is contained in some other subset.

Let us now take a look at some complexity classes that a decision problem can be in.

Definition 10. A decision problem is in **P** if it can be computed (using a deterministic Turing machine) in polynomial ($\mathcal{O}(n^k)$ for some k) time.

Definition 11. A decision problem is in **NP** if it can be computed (using a *non*deterministic[3] Turing machine) in polynomial time.

Note here that P is considered to be *easy* and NP is considered to be *hard* (by which I mean really hard) to compute. To this day, one of the most important open problems in mathematics is whether or not $P = NP$. A great majority of the internet security is built upon the fact that NP is hard.

An example of a NP-complete[4] problem is: "Given a set of integers, is there a subset that sums up to 0?"

Suggestion 24. Explore the topic a bit - find some famous problems which are in NP.

Definition 12. A **counting problem** is a counting version of a decision problem (e.g. "Given a set of integers, *how many* subsets sum up to 0?").

Definition 13. **#P** is the class of counting problems associated with decision problems in NP. These are believed to be *even harder* than NP problems.

Suggestion 25. Find out how difficult it is to find a perfect matching of a graph.

Suggestion 26. Find out how difficult it is to compute a permanent of our adjacency matrix.

Suggestion 27. Find out how difficult it is to compute a determinant.

Suggestion 28. If we are able to find a matrix $\widehat{A}$ such that $|\det(\widehat{A})| = \text{per}(A)$, how will it help us?

Exploration 6 [Signing]

Let us assign a *weight* to each edge of our graph. We will, however, be modest in the choices of possible weights and we will only choose weights ± 1.

Definition 14. A **signing** of a graph G is an assignment of ± 1 weights to the edges of a graph. It is simply a map $\sigma :$ edges of G $\to \{\pm 1\}$.

[3] Such a machine can take several computational paths at once.

[4] Differs slightly, conceptually, from NP.

Definition 15. The **signed adjacency matrix** A^σ will be given by:

$$a_{ij}^\sigma = \begin{cases} \sigma((b_i, w_j)) & \text{if } (b_i, w_j) \text{ is an edge in } G \\ 0 & \text{otherwise} \end{cases}$$

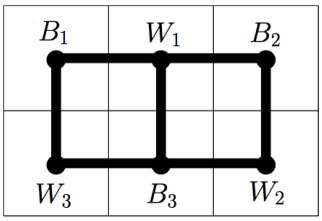
Remark 6. We simply assigned $+$ or $-$ to every 1 present in the adjacency matrix A.

Definition 16. If the signing σ satisfies the property that we are looking for, that is $\text{per}(A) = |\det(A^\sigma)|$, we will say that it is a **Kasteleyn signing**.

Suggestion 29. Convince yourself that we have $\text{per}(A) = |\det(A^\sigma)|$ precisely when all of the terms in $\det(A^\sigma) = \sum_{\pi \in S_n} sgn(\pi) \prod_{i=1}^N a_{i,\pi(i)}^\sigma$ have the *same sign.*

Suggestion 30. At this point, there is a good chance of being lost in the whole argument. Try to go through the main points once more to (re=)realize why finding a Kasteleyn signing efficiently will allow us to find $F(m, n)$ efficiently.

Suggestion 31. Go through a concrete example. Consider a board 2×3. Find its corresponding bipartite graph and adjacency matrix. Find the permanent of this matrix. Find its Kasteleyn signing.[5]



Suggestion 32. Explore other small boards. Can you always find a Kasteleyn signing?

Let us now start our journey to find a Kasteleyn signing of an arbitrary $m \times n$ board graph.

Suggestion 33. Find out what a **planar** graph is.

Definition 17. A graph is **2-connected** if upon removing of any edge, the graph still stays connected.

Suggestion 34. See that we are dealing with a bipartite, planar, 2-connected graph.

As it turns out, there is a polynomial time algorithm to compute the Kasteleyn signing for such a graph.

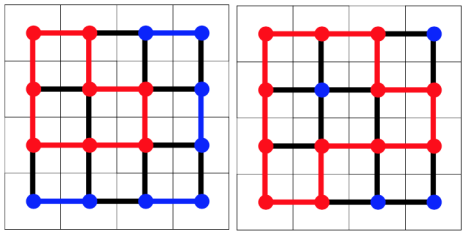
Exploration 7 [Cycles and signs, Proof of Part I]

Definition 18. A **cycle** in a graph G is some sequence of vertices and edges that starts and terminates in the same vertex.

Suggestion 35 (not so relevant)**.** How many cycles are there in a graph corresponding to $m \times n$ board?

Definition 19. We will say that a cycle C is **evenly-placed** if we can remove all of the vertices and edges contained in C and the rest of the remaining graph will still have a perfect matching.

An example and a non-example of such a cycle can be found below:

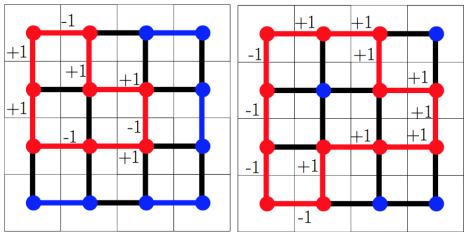


[5] Try assigning -1 to (b_1, w_3) and (b_2, w_2).

Suggestion 36. Convince yourself that any possible cycle in a bipartite graph has an even length.

Definition 20. If we are given a signing σ and a cycle C of even length, we will say that C is **properly-signed** if the *half of the length* of the cycle and the number of edges weighted by -1 that it contains are *of opposite parity*.

An example and a non-example of such a cycle can be found below:



Now let us venture into proving one of the important results that we need in order to find our polynomial solution.

Part I

Theorem (Part I)**.** Given a signing σ, if every evenly-placed cycle in G is properly-signed, then σ is a Kasteleyn signing.

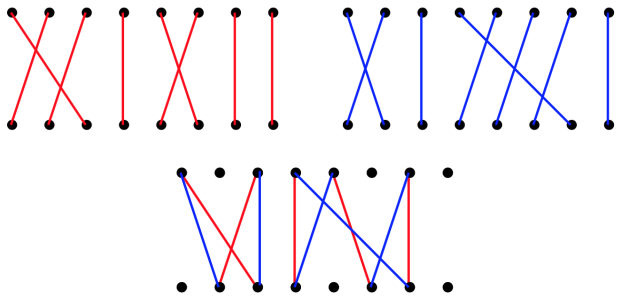
Definition 21. Let us *define* the **sign of a perfect matching** M to be

$$sgn(M) = sgn(\pi)a^{\sigma}_{1,\pi(1)} \cdot a^{\sigma}_{2,\pi(2)} \cdots a^{\sigma}_{N,\pi(N)} = sgn(\pi) \prod_{edge \in M} \sigma(\text{edge}).$$

Remark 7. This is just one term in the sum of $\det(A^{\sigma})$. Hence, to prove the Part I theorem, we will need to show that if σ is such that every evenly-placed cycle is properly-signed, then $sgn(M)$ for all perfect matchings M will have the same sign (and then we will have $\text{per}(A) = |\det(A^{\sigma})|$).

Suggestion 37. Convince yourself that all $sgn(M)$ having the same sign is equivalent to $sgn(M)sgn(M') = 1$ for arbitrary perfect matchings M, M'.

Given two perfect matchings M and M', we will now define the set of edges $M \Delta M'$ to be the edges that are either in M or in M', but not in both[6]. As an example, below are two perfect matchings M and M' followed by $M \Delta M'$:



Now, since any edge that is common to both perfect matchings M and M' will contribute to the product

$$sgn(M)sgn(M') = sgn(\pi)sgn(\pi') \left(\prod_{\text{edge} \in M} \sigma(\text{edge}) \right) \left(\prod_{\text{edge} \in M'} \sigma(\text{edge}) \right)$$

exactly twice (σ^2 of such an edge will be 1), we only need to consider those in $M \Delta M'$ - hence we have

$$sgn(M)sgn(M') = sgn(\pi)sgn(\pi') \left(\prod_{edge \in M \Delta M'} \sigma(edge) \right).$$

[6] this is analogous to the XOR operation.

Suggestion 38 (important)**.** Show that for any two perfect matchings M and M', $M\Delta M'$ is a disjoint union of cycles.

Remark 8. These cycles will also have even length, since we are dealing with a bipartite graph.

Suggestion 39. Show now, by a simple argument, that every cycle in $M\Delta M'$ is evenly-placed (that is, that if we remove this cycle from the original graph, there still exists a perfect matching on the rest of it).

Suppose now that $M\Delta M'$ contains k disjoint, evenly-placed cycles C_i (they are of even length). By the assumption in the Part I theorem, we have that every evenly-placed cycle is also *properly-signed*. If we denote the lengths of these k cycles by $2l_1, 2l_2, \ldots, 2l_k$, we have

$$\begin{aligned} sgn(M)sgn(M') &= sgn(\pi)sgn(\pi')\left(\prod_{edge\in M\Delta M'}\sigma(edge)\right), \\ &= sgn(\pi)sgn(\pi')\prod_{i=1}^{k}\left(\prod_{edge\in C_i}\sigma(edge)\right), \\ &= sgn(\pi)sgn(\pi')\prod_{i=1}^{k}(-1)^{l_i-1}, \\ &= sgn(\pi)sgn(\pi')(-1)^{(l_1-1)+(l_2-1)+\cdots+(l_k-1)}. \end{aligned}$$

Remark 9. The properly-signed property of the cycles gives us the second-to-last equality.

Let us for simplicity denote $(l_1-1)+(l_2-1)+\cdots+(l_k-1)$ as L. Then we have simply:

$$sgn(M)sgn(M') = sgn(\pi)sgn(\pi')(-1)^L.$$

We have now reduced the matter of proving the Part I theorem into a simple-looking formula

$$sgn(\pi)sgn(\pi')(-1)^L = 1$$

.

Suggestion 40. Convince yourself that if we could prove that $sgn(\pi) = sgn(\pi')(-1)^L$, this would prove the above formula and thus the Part I theorem. Notice also that by the definition of the parity of a permutation, it would be enough to show that π and π' differ by L transpositions!

Suggestion 41 (harder)**.** Try to prove that π and π' differ by L transpositions, by considering each of the disjoint cycles in $M\Delta M'$ separately. Note that this also produces an algorithm.

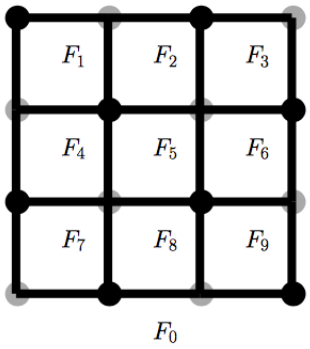
This would conclude the proof of the Part I. Let me now summarize everything that we have concluded in this section:

1. π and π' differ by L transpositions;
2. Hence $sgn(\pi) = sgn(\pi')(-1)^L$;
3. Using this identity, we may prove that $sgn(\pi)sgn(\pi')(-1)^L = 1$ by plugging it into the above formula;
4. Using the assumption that every evenly-placed cycle in G is properly-signed, this will prove that all the terms in $det(A^\sigma)$ have the same sign;
5. Hence $|\det(A^\sigma)| = \mathrm{per}(A)$ and σ is Kasteleyn

Exploration 8 [Making use of planarity, Proof of Part II]

Definition 22. The **face** of a planar graph is a region completely enclosed by other edges. There is also one **outer face**.

An example may be found below:



Euler's formula says that for any planar graph we have

$$V - E + F = 2,$$

where V is the number of vertices, E number of edges and F number of faces of the graph.

Suggestion 42. Look up or think about a proof of Euler's formula related to solids.

For reference, I am leaving here the statement of Part I again. You may find it useful in the following section.

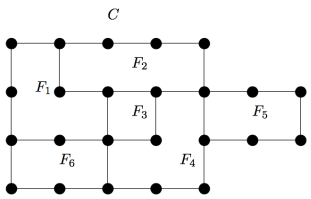
Theorem (Part I)**.** Given a signing σ, if every evenly-placed cycle in G is properly-signed, then σ is a Kasteleyn signing.

Part II

Theorem (Part II)**.** Consider a given bipartite, planar and 2-connected graph G with signing σ. If the boundary cycle of every inner face is properly-signed, then σ is Kasteleyn.

Notice now that if we manage to prove Part II and find an algorithm that finds such a signing σ, we will have won and found an algorithm for computing $F(m, n)$. Let us therefore dive in to proving Part II.

Consider a bipartite, planar and 2-connected graph G and an *evenly-placed* cycle C in G. Such a cycle will enclose some inner faces $F_1, F_2, \ldots F_k$ with corresponding boundary cycles C_i of lengths $2l_i$. For illustration purposes, it may look something like this:



Let r denote the number of vertices *inside* C.

Suggestion 43. Use Euler's formula to prove that

$$r + 2l + k + 1 = l + l_1 + \cdots + l_k + 2.$$

Suggestion 44. Deduce that r is even and hence

$$l - 1 \equiv l_1 + \cdots + l_k - k \pmod 2.$$

Suggestion 45. Show that if n_C denotes the number of edges signed with -1 in C, then $n_C \equiv n_{C_1} + \cdots + n_{C_k} \pmod 2$.

Suggestion 46. Deduce finally that if the boundary cycle of every inner face is properly-signed, then we have:

$$n_C \equiv n_{C_1} + \cdots + n_{C_k} \equiv (l_1 - 1) + \cdots + (l_k - 1) \equiv l - 1 \pmod 2.$$

This, however, means that every evenly-placed cycle in G is properly-signed and so Part I finishes of our proof of Part II. What's left now? If we are able to construct a signing σ that satisfies Part II, we can use this signing to find $\mathrm{per}(A)$!

Exploration 9 [Construction of σ satisfying Part II, Summary]

Suggestion 47. Constructing a signing that satisfies Part II is not that hard at all - try to come up with a way to find it![7]

Hence we have now arrived to a complete algorithm that computes $F(m, n)$. Let me recapitulate the steps:

1. Construct a bipartite graph for a given board.
2. Sign the edges with a signing σ such that the condition of Part II is satisfied.
3. Find the adjacency matrix A^σ.
4. Compute $F(m, n) = |\det(A^\sigma)|$.

Suggestion 48. Find the computational complexity of this algorithm. Which step takes the most time?

Suggestion 49. Simulate the algorithm step-by-step on a moderately sized board (say 3×5) and hence find the number of ways to tile it with dominoes.

Remark 10. You may be in a need of a tool that computes determinants - I found this one useful:
`https://matrix.reshish.com/determinant.php`

Final Exploration

If you are reading this, congratulations - you either got very far in this text or you are just checking out what's in the end. In any case, I would like to finish the explorations with a few ideas that I got during writing this text and haven't invested much thoughts into them (yet).

Suggestion 50. What happens if we allow the dominoes to be oriented? (You can imagine they have an arrow drawn on them)

Suggestion 51. Try to apply the whole approach to a grid that is not rectangular - in what cases does everything still work out?

Suggestion 52. Try to find the nicest/most complicated example of a grid such that the approach is working on it.

Suggestion 53. What happens to the whole argument if we use other types of tiles?

Suggestion 54. Using the eigenvalues (these are used to compute determinants) of a Cartesian product of two graphs $1 \times n$ and $m \times 1$, one can even arrive to a closed form formula:

$$F(m, n) = \prod_{k=1}^{m} \prod_{l=1}^{n} \left(4 \cos^2 \frac{k\pi}{m+1} + 4 \cos^2 \frac{l\pi}{n+1} \right)^{\frac{1}{2}}$$

If you are brave enough, try to find out more about its derivation. What does the adjacency matrix of such Cartesian product look like?

References

The above text was heavily influenced by the text of Brendan W. Sullivan, Carnegie Mellon University:
`http://www.math.cmu.edu/~bwsulliv/domino-tilings.pdf`

Very handy came in the lecture notes of the Introduction to Theoretical Computer science course by Julian Bradfield, University of Edinburgh:
`http://www.inf.ed.ac.uk/teaching/courses/itcs/itcs-slides.pdf`

[7] Hint: While possible, try removing edges from a graph such that it always "destroys" one inner face. Assign 1's to what is left and...