2048 Without Merging

Hugo A. Akitaya* Erik D. Demaine† Jason S. Ku‡
Jayson Lynch§ Mike Paterson¶ Csaba D. Tóth‖

Abstract

Imagine $t \leq mn$ unit-square tiles in an $m \times n$ rectangular box that you can tilt to cause all tiles to slide maximally in one of the four orthogonal directions. Given two tiles of interest, is there a tilt sequence that brings them to adjacent squares? We give a linear-time algorithm for this problem, motivated by 2048 endgames. We also bound the number of reachable configurations, and design instances where all t tiles permute according to a cyclic permutation every four tilts.

1 Introduction

2048 is a popular open-source video game by then-19-year-old Gabriele Cirulli [Cir14, Wik20] that took the world by storm in 2014. It was inspired by another game called 2048 by Saming, which in turn was inspired by a similar game called 1024!, which in turn was inspired by the genesis game Threes! by Vollmer, Wohlwend, and Hinson released just two months earlier, all of which inspired many other variants. See [LU18] for more on the history and descriptions of several game-variant rules. Cirulli's 2048, Threes!, Fives, Det2048, and Fibonacci all feature the same kind of movement, also identical to the 2011 physical puzzle game Tilt [BDF+19, BLC+19, BGC+20]: unit-square tiles in a rectangular box with only four global ***tilt*** controls — sliding all tiles maximally in an orthogonal direction among $\{N, E, S, W\}$. Each tile has a label, and certain labeled tiles merge together (into a single newly labeled tile) when slid into each other; the goal is generally to produce a tile with a particular label. Each game also has a (possibly randomized) algorithm for introducing new tile(s) after each move. Most of these games (in their perfect-information form) are NP-hard [LU18, AAD16].

Our results. In this paper, we consider a variant where no tile merging can happen, no additional tiles are introduced (as in [AAD16]), and there are no fixed obstacles (as in most games above, but unlike Tilt and e.g. 1024!). This minimal variant, which we call ***2048 without merging***, intends to capture some core mathematical structure of the many game variants listed above.

In particular, we study the problem of whether two particular tiles can be made adjacent by a sequence of tilt operations, which is motivated by a subproblem arising near the end of a 2048 game, where the board has two 1024-labeled tiles and the player wants to make them adjacent so that another tilt merges them into a 2048-labeled tile. This problem was posed by Mike Paterson in 2018. We solve this problem in Section 3 by giving an $O(t)$-time algorithm to decide, given an initial $m \times n$ board configuration of $t \leq mn$ tiles and two marked tiles t_1 and t_2, whether there exists a tilt sequence that brings t_1 and t_2 to adjacent squares. In the positive case, our algorithm also outputs the minimum tilt sequence. This algorithm generalizes to decide in $O(st)$ time whether any pair among s special tiles (1024s) can be made adjacent. In particular, this running time is $O(t)$ for $s = O(1)$ and always $O(t^2)$.

We also consider the combinatorial structure of the motion of all tiles, which is roughly described by powers of a single permutation. In Section 4, we give a lower bound of $2^{\Omega(\sqrt{t})}$ and an upper bound of $2^{O(\sqrt{t \log t})}$ on the number of different states that can be reached by a tilt sequence from an initial $m \times n$ board configuration with $t = \Theta(mn)$ tiles. Section 5 shows that there exist initial $m \times n$ board configurations with permutation cycles of length $\Omega(mn)$ and, for even m and n, there is a configuration in which every tile is part of the same permutation cycle. In the latter configurations, each tile can reach any possible target square via a tilt sequence of length $O(mn)$.

2 Definitions and Basics

We base our terminology on [BLC+19, BGC+20]. A ***board*** is a rectangular region of the 2D square lattice, whose 1×1 cells we refer to as ***squares***. We represent an $m \times n$ board B by $\{(x, y) \mid x \in \{0, 1, \ldots, m-1\}, y \in \{0, 1, \ldots, n-1\}\}$ where $(0, 0)$ represents the bottom-left square. Let $\mathcal{T}$ be a set of t objects called ***(slidable)***

*School of Computer Science, Carleton University, hugoakitaya@gmail.com
†CSAIL, MIT, edemaine@mit.edu
‡EECS, MIT, jasonku@mit.edu
§CSAIL, MIT, jaysonl@mit.edu
¶Dept of Computer Science, University of Warwick, UK, M.S.Paterson@warwick.ac.uk
‖CSU Northridge and Tufts University, cdtoth@eecs.tufts.edu

tiles. A ***configuration*** is an injective function $C : \mathcal{T} \to B$. We call a square ***full*** if it is in the image of C, and ***empty*** otherwise.

Tilt is an operation that takes a configuration C and a direction $d \in \{N, E, S, W\}$ and returns a configuration C' as follows. A tilt is ***horizontal*** if $d \in \{E, W\}$, or ***vertical*** if $d \in \{N, S\}$. We describe a tilt for $d = N$; the other cases are symmetric. For all rows j from top to bottom, and for all columns i, if (i, j) is full, then move the tile at (i, j) (marking (i, j) empty) to the topmost square (i, j') in the ith column marked as empty, where $j' \geq j$ (marking (i, j') as full).

A ***tilt sequence*** is a sequence of tilts applied to a configuration represented by a sequence of directions $D = (d_1, d_2, \ldots, d_k)$. Two tilt sequences are equivalent if they produce the same configuration. A row (column) is called ***monotone non-increasing*** if every full square is to the left of (below) every empty square in the row (column). ***Monotone non-decreasing*** is defined analogously. We call a configuration ***SW-canonical*** if every row and every column is monotone non-increasing. Alternatively, C is SW-canonical if a tilt with direction S or W would return C. Symmetrically, we can define NE-, NW-, and SE-canonical configurations.

Lemma 1 *After one horizontal and one vertical tilt, not necessarily in this order, we get a canonical configuration.*

Proof. Without loss of generality, we apply the tilt sequence (S, W). After the first tilt, every column is monotone non-increasing. Consequently, the number of full squares in a row is monotone (non-increasing) from bottom to top. Similarly, the second tilt makes every row monotone non-increasing. By definition, a horizontal tilt does not change the number of full squares in each row. The columns in the resulting configurations are also monotone non-increasing. The result is a SW-canonical configuration. □

The following lemma will allow us to focus only on a clockwise or counterclockwise tilt sequence, i.e., a substring of $(N, E, S, W)^*$ or $(N, W, S, E)^*$ (where the Kleene Star notation A^* denotes sequence A repeated zero or more times).

Lemma 2 *Every shortest tilt sequence between two configurations is either a clockwise or counterclockwise tilt sequence.*

Proof. It is clear by definition that the tilt sequence (S, S) is equivalent to (S). Similarly, (N, S) is equivalent to S. Then the shortest tilt sequence between configurations either has length less than 2 or one of its two first tilts is horizontal and the other one is vertical. Without loss of generality, let (S, W) be the prefix of such sequence. By Lemma 1, after these two initial tilts we get a SW-canonical configuration and hence the third tilt cannot be in the S or W direction. It cannot be E, since (S, E) is equivalent and shorter than (S, W, E). Hence the sequence starts with (S, W, N). We can then induct on the length of the sequence to show that all subsequent tilts must follow a clockwise order. □

The following lemma allows us to describe the movement of the tiles using permutations.

Lemma 3 *After every four tilts in a shortest sequence, the union of the filled squares form the same shape, but with permuted tile positions.*

Proof. Suppose C is a SW-canonical configuration where the length of row i is a_i and the length of column j is b_j. Since every row and column in C is monotone non-increasing, $a_0 \geq \ldots \geq a_{m-1}$, and we find that $b_j = |\{i \mid a_i \geq j\}|$. After a horizontal tilt in direction E to reach configuration C', all the row lengths remain as before and are monotone non-increasing. Since C' is a SE-canonical configuration the column lengths are non-decreasing and $b_{n-1-j} = |\{i \mid a_i \geq j\}|$, as before but counting from the right. So the sequence of column lengths is now reversed, i.e., $b_{m-1}, \ldots, b_0$. From C' a vertical tilt in direction N would reverse the row length order, and so on, and a complete cycle of four tilts will return row and column lengths to their original sequences. □

Let g be the permutation referred by Lemma 3. Our techniques will be based on the cyclic subgroup generated by g and its cycle decomposition. For example, if the initial configuration is given by a lower-triangular matrix in a square board where exactly the squares on and below the main diagonal are full, the permutation g will induce cycles of length 3.

If $\mathcal{C}$ is a cycle in the permutation generated by g, and $s_1, s_2 \in \mathcal{C}$, then the ***cycle index*** $\text{ind}(s_1, s_2)$ is the smallest nonnegative integer i such that $g^i(s_1) = s_2$, that is, i successive application of g carries s_1 to s_2. We say that two cycles are ***adjacent*** if there exists a pair of adjacent squares with one square in each cycle.

3 Algorithm

In this section we give an algorithm that decides the following problem. Given an initial configuration C_0 on an $m \times n$ board, and two tiles $t_1, t_2 \in \mathcal{T}$, can a tilt sequence produce a configuration C_k in which t_1 and t_2 are in adjacent squares? If yes, output a shortest such sequence. We denote by C_i the configuration obtained after the ith tilt.

1. Guess the first two tilt directions from $\{(S, W)$, (S, E), (N, W), (N, E), (E, N), (E, S), (W, N), $(W, S)\}$, checking whether t_1 and t_2 are adjacent

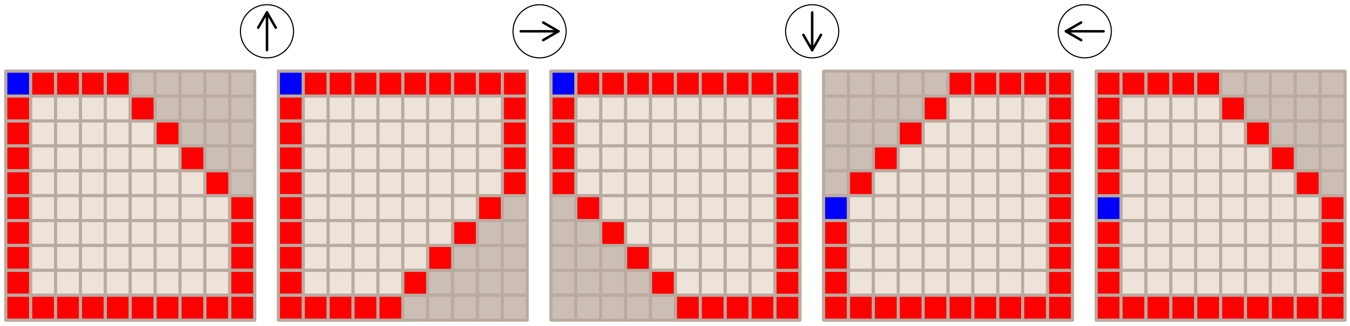

Figure 1: Application of tile sequence (N, E, S, W) on a single ring of C_n resulting in permutation $g(C_n)$. Each tile moves cyclically counterclockwise by k positions, where $2k$ is the width of the ring, e.g., the tile marked in blue.

in C_0 and C_1. The obtained configuration C_2 is in canonical form.

2. Guess a type of canonical configuration from $\{SE, SW, NE, NW\}$, computing the first configuration in the sequence, i.e., a configuration C' in $\{C_2, C_3, C_4, C_5\}$.

3. Compute the permutation g equivalent to four tilts in clockwise or counterclockwise direction depending on the guessed first two tilts.

4. Compute the cycle decomposition of g.

5. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be the cycles containing t_1 and t_2 respectively. By restricting our attention to the elements of $\mathcal{C}_1$ and $\mathcal{C}_2$, successively applying g results in $\mathrm{lcm}(|\mathcal{C}_1|, |\mathcal{C}_2|)$ possible different permutations of these elements. For example, if $\mathcal{C}_1 = \mathcal{C}_2$, then the number of such states is $|\mathcal{C}_1|$. Check in each of these states whether t_1 is adjacent to t_2.

Theorem 4 *Given an initial configuration C_0 on an $m \times n$ board, and two tiles $t_1, t_2 \in \mathcal{T}$, a shortest tilt sequence required to make t_1 and t_2 adjacent can be computed in $O(|\mathcal{T}|)$ time.*

Proof. The correctness of our algorithm follows directly from Lemmas 1–3. After the first two configurations, a clockwise or counterclockwise tilt sequence produces only canonical configurations. In Step 1, the algorithm tries all possible starting tilts and both rotation directions. Step 2 considers all possible types of canonical configurations. Steps 3–5 considers all possible relative positions of t_1 and t_2 given a rotation direction of the tilt sequence and a type of canonical configuration. Hence, the algorithm is correct.

Most of the algorithm runs in $O(|\mathcal{T}|)$ time. Steps 1–2 add a $8 \cdot 4 = 32$ multiplicative factor to the runtime. Steps 3–4 can be executed in $O(|\mathcal{T}|)$ time by simulating four tilts to obtain a directed graph representing g, and obtaining the cycles of g containing t_1 and t_2 via a DFS from such vertices. Here we use that a tilt operation can be simulated in $O(|\mathcal{T}|)$ time using counting sort on the coordinates; for instance, a horizontal tilt determines the x order of tiles in each row, then moves the tiles to one extreme of the board without changing such order.

In Step 5, we need more care. A naïve implementation iterates through all $\mathrm{lcm}(|\mathcal{C}_1|, |\mathcal{C}_2|) = O(|\mathcal{T}|^2)$ different possible positions of t_1 and t_2 since both $|\mathcal{C}_1|$ and $|\mathcal{C}_2|$ are $O(|\mathcal{T}|)$. The resulting running time is $O(|\mathcal{T}|^2)$.

We can reduce the runtime of Step 5 to $O(|\mathcal{T}|)$ as follows. For all $|\mathcal{C}_1| \leq |\mathcal{T}|$ possible positions of t_1, we can try all possible adjacent squares (up to four), checking whether a tilt sequence can bring tiles t_1 and t_2 simultaneously to those squares, as follows. Suppose the desired positions are indeed in $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively, say the pth and qth positions of $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively, counting from the initial positions of t_1 and t_2 with zero indexing. By Bézout's Identity [JJ98], the set of all integer linear combinations $i|\mathcal{C}_1| + j|\mathcal{C}_2|$ is exactly the set of integer multiples of $d = \gcd(|\mathcal{C}_1|, |\mathcal{C}_2|)$. Thus the desired tiles can meet at the specified position exactly if $p \equiv q \pmod{d}$. We can then find the actual number of tilts by using the Chinese Remainder Theorem. Since we only need to compute d once, spending $O(|\mathcal{T}|)$ time, and we can test each of the $4|\mathcal{C}_1|$ meeting positions in constant time, the total runtime is $O(|\mathcal{T}|)$. □

Generalization to $|\mathcal{S}|$ special tiles. We can generalize Theorem 4 to the case of a subset $\mathcal{S} \subseteq \mathcal{T}$ of special tiles, where $2 \leq |\mathcal{S}| \leq |\mathcal{T}|$. Specifically, we consider the following problem. Given an initial configuration C_0 on an $m \times n$ board, and a subset $\mathcal{S} \subseteq \mathcal{T}$ of $|\mathcal{S}|$ special tiles, can a tilt sequence produce a configuration C_k in which two special tiles are in adjacent squares? If yes, output a shortest such sequence. We modify our previous algorithm, designed for the case $|\mathcal{S}| = 2$, by replacing Step 5 with the following:

5*. For each pair $\{s_1, s_2\}$ of adjacent squares that belong respectively to ***special cycles*** $\mathcal{C}_i$ and $\mathcal{C}_j$ of g (that is, cycles containing at least one special tile in $\mathcal{S}$), check whether any of the special tiles in $\mathcal{C}_i$

and $\mathcal{C}_j$ can simultaneously move to s_1 and s_2 by successively applying g, as follows:

- Let $d = \gcd(|\mathcal{C}_i|, |\mathcal{C}_j|)$.
- For each $t \in \mathcal{C}_i \cap \mathcal{S}$, compute the cycle index $\operatorname{ind}(t, s_1)$ (as defined in Section 2), and let $I_1 = \{\operatorname{ind}(t, s_1) \bmod d \mid t \in \mathcal{C}_i \cap \mathcal{S}\}$.
- Similarly, compute $I_2 = \{\operatorname{ind}(t, s_2) \bmod d \mid t \in \mathcal{C}_j \cap \mathcal{S}\}$.
- Check whether $I_1 \cap I_2 \neq \emptyset$.

Theorem 5 *Given an initial configuration of $|\mathcal{T}|$ tiles on an $m \times n$ board and a set $\mathcal{S} \subset \mathcal{T}$ of special tiles, a shortest tilt sequence required to make two special tiles adjacent can be computed in $O(|\mathcal{S}| \cdot |\mathcal{T}|)$ time.*

Proof. For the correctness of the algorithm, assume that there exist two special tiles $t_1 \in \mathcal{C}_i \cap \mathcal{S}$, $t_2 \in \mathcal{C}_j \cap \mathcal{S}$ and an integer $h \geq 0$ such that $g^h(t_1) = s_1$ and $g^h(t_2) = s_2$, where s_1 and s_2 are adjacent squares in a canonical configuration C'. Then $h \equiv \operatorname{ind}(t_1, s_1) \pmod{|\mathcal{C}_i|}$ and $h \equiv \operatorname{ind}(t_2, s_2) \pmod{|\mathcal{C}_j|}$. By Bézout's Identity, $\operatorname{ind}(t_1, s_1) \equiv \operatorname{ind}(t_2, s_2) \pmod{d}$, where $d = \gcd(|\mathcal{C}_i|, |\mathcal{C}_j|)$. Conversely, if $\operatorname{ind}(t_1, s_1) \equiv \operatorname{ind}(t_2, s_2) \pmod{d}$, then there exits an integer $h \geq 0$ such that $h \equiv \operatorname{ind}(t_1, s_1) \pmod{|\mathcal{C}_i|}$ and $h \equiv \operatorname{ind}(t_2, s_2) \pmod{|\mathcal{C}_j|}$.

As noted above, Steps 1–4 of the algorithm run in $O(|\mathcal{T}|)$ time. In Step 5*, we can memoize the gcd of the lengths of all pairs of adjacent special cycles. For each such pair $\{\mathcal{C}_i, \mathcal{C}_j\}$, the Euclidean algorithm computes $\gcd(|\mathcal{C}_i|, |\mathcal{C}_j|)$ in $O(\log(|\mathcal{C}_i| + |\mathcal{C}_j|)) = O(\log|\mathcal{T}|)$ time. There are at most $|\mathcal{S}|$ special cycles, and hence $O(|\mathcal{S}|^2)$ pairs of special cycles. Furthermore, every pair of adjacent cycles contain a pair of adjacent squares, and since each square has at most four neighbors, there are $O(|\mathcal{T}|)$ adjacent pairs of squares. Therefore, the gcds can be computed in $O(\min\{|\mathcal{S}|^2, |\mathcal{T}|\} \log|\mathcal{T}|)$ time. This time bound is always $O(|\mathcal{S}| \cdot |\mathcal{T}|)$: if $|\mathcal{S}|^2 \leq |\mathcal{T}|$, then we have $O(|\mathcal{S}|^2 \log|\mathcal{T}|) = O(|\mathcal{S}| \cdot |\mathcal{S}| \log|\mathcal{T}|) = O(|\mathcal{S}|\sqrt{|\mathcal{T}|} \log|\mathcal{T}|) = O(|\mathcal{S}| \cdot |\mathcal{T}|)$; and if $|\mathcal{T}| \leq |\mathcal{S}|^2$, then we have $O(|\mathcal{T}| \log|\mathcal{T}|) = O(|\mathcal{T}|\sqrt{|\mathcal{T}|}) = O(|\mathcal{T}| \cdot |\mathcal{S}|)$.

Step 5* iterates through all $O(|\mathcal{T}|)$ pairs (s_1, s_2) of adjacent squares. Suppose that $s_1 \in \mathcal{C}_i$ and $s_2 \in \mathcal{C}_j$ where $\mathcal{C}_i$ and $\mathcal{C}_j$ are special cycles. Given the precomputed indices of g, the index sets I_1 and I_2 can be computed in $O(|\mathcal{C}_i \cap \mathcal{S}| + |\mathcal{C}_j \cap \mathcal{S}|) = O(|\mathcal{S}|)$ time. Checking whether $I_1 \cap I_2 = \emptyset$ via hashing takes $O(|I_1| + |I_2|) = O(|\mathcal{S}|)$ time. For each index in $I_1 \cap I_2$, we can then find the actual number of tilts using the Chinese Remainder Theorem, in overall $|I_1 \cap I_2| = O(|\mathcal{S}|)$ time. After memoizing the gcd of adjacent special cycles, Step 5* of the algorithm thus runs in $O(|\mathcal{S}| \cdot |\mathcal{T}|)$ time. $\square$

4 Bounds on Reachable Configurations

Lower bound. For even n, consider the configuration

$$C_n = \begin{bmatrix} F & L \\ F & F \end{bmatrix}$$

on an $n \times n$ board where F is a full $n/2 \times n/2$ square, and L is a $n/2 \times n/2$ matrix where exactly the squares below the main diagonal are full (see Figure 1). Let the outer ***ring*** be the set of extremal tiles in N, E, S, or W direction, and define inner rings recursively. The outer ring contains $7(n/2 - 1) + 3$ tiles, while each successive inner ring contains 7 fewer tiles. So C_n comprises $n/2$ rings, with $7k+3$ tiles in ring k for $k \in \{0, \ldots, n/2-1\}$, where the innermost ring 0 is an L-shaped tromino.

Lemma 6 *Each ring in C_n is self-contained (does not mix with adjacent rings), and the cyclic order of elements around the ring does not change after applying g.*

Proof. By symmetry, it suffices to show that each ring in C'_n, the NW-canonical configuration obtained after a N tilt from C_n, is exactly a ring in C_n, and that the cyclic order of tiles does not change. The outermost ring is composed of the first and last columns, and the extremal tiles in each remaining column. After the tilt, all such tiles remain extremal and their order along the convex hull remains the same. No other tiles become extremal because the number of tiles in adjacent columns differ by at most one. Hence, the outermost ring remains the same. If we look at the remaining tiles after removing the outermost ring, they form a configuration C_{n-2} in the $(n-2) \times (n-2)$ board obtained after removing the extremal rows and columns. The configuration C'_{n-2} obtained after a N tilt from C_{n-2} can be obtained in the same way from C'_n. Hence, the second outermost ring also remains the same in C_n and C'_n. By induction, all rings remain the same. $\square$

Theorem 7 *The number of different configurations reachable from C_n is $e^{\Theta(n)} = e^{\Theta(\sqrt{t})}$ where t is the number of tiles in C_n.*

Proof. We first show that, after applying g, the elements in ring k (with width $2k$) shift by k positions counterclockwise along its ring. By Lemma 6 it suffices to show that the top-left tile x moves down by k. As shown in Figure 1, x does not move after the N and E tilts, moves down by k after the S tilt, and again does not move after the W tilt.

We prove the claimed bound by focusing on a subset of rings and bounding the number of different states of the tiles in the selected rings that are obtained by successively applying g. We can restrict to rings with prime lengths of the form $7k+3$. In such cases, the ring induces a single cycle in g because the length of the ring and the

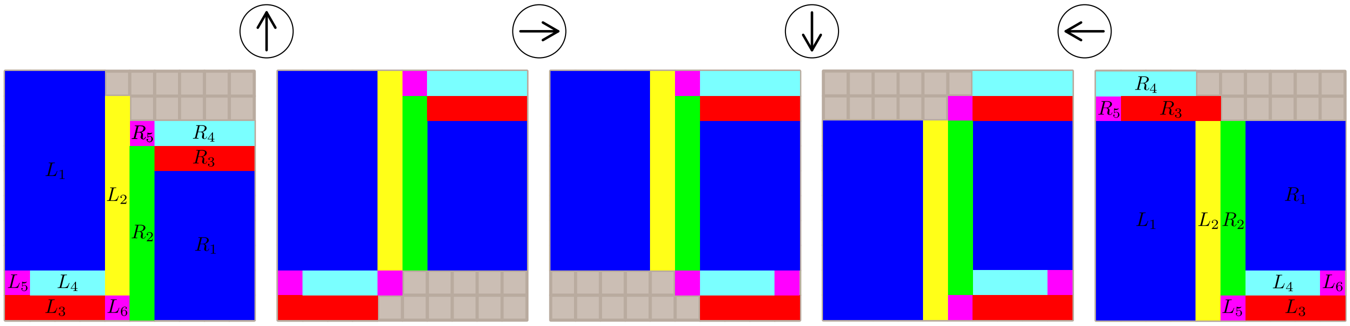

Figure 2: Application of the tilt sequence (N, E, S, W) to C_{mn} resulting in the permutation $g(C_{mn})$, depicting the movement of tile subsets $\{L_1, L_2, L_3, L_4, L_5, L_6, R_1, R_2, R_3, R_4, R_5\}$. Each of these subset moves as a rigid block.

amount that each element shifts around the ring after g are coprimes and each element will eventually visit every position in the ring. Using a variant of the prime number theorem for arithmetic progressions [Dab89, Tao09],

$$\sum_{\substack{p \le x \\ p \equiv a \bmod q}} \log p \equiv \frac{x}{\varphi(q)}(1 + o(1))$$

if $(a, q) = 1$, where φ is Euler's totient function. In particular, the product of primes
$p \le n/2$, $p \equiv 3 \pmod 7$, is $e^{\Theta(n)} = e^{\Theta(\sqrt{t})}$. □

Upper bound. Starting from a SW-canonical configuration containing t tiles in an $m \times n$ board, the number of reachable configurations equals the least common multiple of the cycle lengths. Every cycle has length at most t (the number of tiles), hence the lcm of the cycle lengths is bounded above by $\mathrm{lcm}(1, 2, \ldots, t) = e^{t(1+o(1))}$.

We can improve upon this bound by realizing that the sum of the cycle lengths must be t. The maximal least common multiple of a partition of t into positive integers is known as Landau's function [Nic13], and it is asymptotically $e^{\Theta(\sqrt{t \log t})}$.

5 Long Cycles

In this section we provide a construction for an initial $m \times n$ board configuration C_{mn} with $\Theta(mn)$ tiles with a single permutation cycle. We can assume that m and n are even, or else we construct the instance of size $(2\lfloor m/2 \rfloor) \times (2\lfloor n/2 \rfloor)$ and add an empty row and/or column. Refer to Figure 2. We leave empty the last $n/2+1$ $(n/2)$ squares in the top (second to top) row of C_{mn}. The remaining squares are filled with tiles.

We now show that the permutation g induced by the tilt sequence (N, E, S, W) has a single cycle. We first describe g by giving a successor function based on the coordinates of a given square, then we describe an algorithm that outputs in order all elements in a cycle in g and show that the number of outputs equals the number of tiles. Let $s(x, y)$ be such a successor function where $s(x, y)$ is the coordinates of the square after the square (x, y) in g. Let $\Delta(x, y) = s(x, y) - (x, y)$ be the vector from (x, y) to its successor. Refer to Figure 2. We partition the occupied squares in the board into 11 regions as follows, where $p(x, y)$ denotes the region containing square (x, y):

$$p(x,y) = \begin{cases} L_1 & \text{if } 0 \le x < m/2 - 1 \text{ and } 2 \le y < n \\ L_2 & \text{if } x = m/2 - 1 \text{ and } 1 \le y < n - 1 \\ L_3 & \text{if } 0 \le x < m/2 - 1 \text{ and } y = 0 \\ L_4 & \text{if } 1 \le x < m/2 - 1 \text{ and } y = 1 \\ L_5 & \text{if } x = 0 \text{ and } y = 1 \\ L_6 & \text{if } x = m/2 - 1 \text{ and } y = 0 \\ R_1 & \text{if } m/2 + 1 \le x < m \text{ and } 0 \le y < n - 4 \\ R_2 & \text{if } x = m/2 \text{ and } 0 \le y < n - 3 \\ R_3 & \text{if } m/2 + 1 \le x < m \text{ and } y = n - 4 \\ R_4 & \text{if } m/2 + 1 \le x < m \text{ and } y = n - 3 \\ R_5 & \text{if } x = m/2 \text{ and } y = n - 3 \end{cases}$$

Now we define $\Delta(x, y)$ based on the above partition, which can be easily verified by following the tiles initially in each region after four clockwise tilts as shown in Figure 2.

$$\Delta(x,y) = \begin{cases} (0,-2) & \text{if } p(x,y) = L_1 \\ (0,-1) & \text{if } p(x,y) = L_2 \\ ((m/2+1),0) & \text{if } p(x,y) = L_3 \\ (m/2,0) & \text{if } p(x,y) = L_4 \\ (m/2,-1) & \text{if } p(x,y) = L_5 \\ (m/2,1) & \text{if } p(x,y) = L_6 \\ (0,2) & \text{if } p(x,y) = R_1 \\ (0,1) & \text{if } p(x,y) = R_2 \\ (-m/2,2) & \text{if } p(x,y) = R_3 \\ (-(m/2+1),2) & \text{if } p(x,y) = R_4 \\ (-m/2,1) & \text{if } p(x,y) = R_5 \end{cases}$$

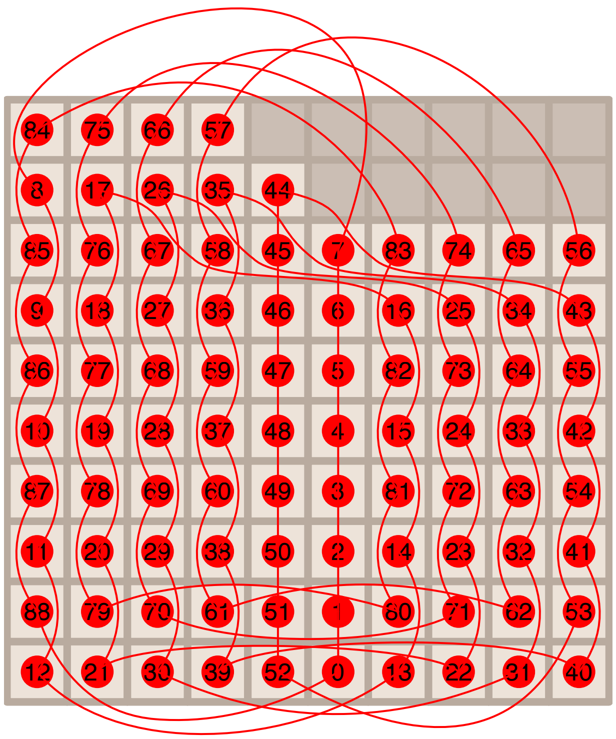

Figure 3: An example output cycle of g applied to C_{mn} for $m = n = 10$. Indexing starts with 0 at $(\frac{m}{2}, 0)$ increasing to 88 at $(0, 1)$, where increasing indices are outlined with increasing opacity.

We now describe an algorithm that outputs one cycle of g starting with square $(m/2, 0)$. Refer to Figure 3 for an example of output of the algorithm.

1. For $0 \le j < n-2$: output $(\frac{m}{2}, j)$
2. For $0 \le i < \frac{m}{2} - 1$:
   (a) For $0 \le j < \frac{n}{2}$: output $(i, n-2-2j)$
   (b) For $0 \le j < \frac{n}{2} - 1$: output $(\frac{m}{2} + 1 + i, 2j)$
3. For $0 \le j < n-1$: output $(\frac{m}{2} - 1, n-2-j)$
4. For $0 \le i < \frac{m}{2} - 1$:
   (a) For $0 \le j < \frac{n}{2} - 1$: output $(m-1-i, 1+2j)$
   (b) For $0 \le j < \frac{n}{2}$: output $(\frac{m}{2} - 2 - i, n-1-2j)$

Theorem 8 *The permutation g of the tilt sequence (N, E, S, W) on configuration C_{mn} has a single cycle of length $mn - m - 1$.*

Proof. It suffices to show that the algorithm above correctly outputs a cycle in order, and that it outputs all $mn - m - 1$ full squares of C_{mn}. We now focus on the former. Step 1 outputs all squares in R_2 and R_5 from bottom to top. This matches the successor function for R_2. The successor of the square in R_5 is $(0, n-2)$ which is the first position output by Step 2. Step 2(a) outputs squares in L_1 and a square in L_3 at the end of the loop, according to the successor function in L_1, i.e., two units below the previous one. The first square output by each execution of Step 2(b) is the successor of the square in L_3 output by Step 2(a), i.e., $m/2+1$ units to the right. Step 2(b) outputs squares in R_1 and a square in R_3 at the end of the loop, according to the successor function in R_1, i.e., two units above the previous one. The next output square is by either another execution of Step 2(a), or by Step 3, both obey the successor function of R_3, $\Delta(x, y) = (-m/2, 2)$. Step 3 outputs squares in L_2 from top to bottom and then outputs the square in L_6, obeying the successor function in L_2. The next square is $(m-1, 1)$ in R_1 satisfying the successor function in L_6. Step 4(a) outputs squares in R_1 and a square in R_4 at the end of the loop, according to the successor function in R_1, i.e., two units above the previous one. The first square output by each execution of Step 4(b) is the successor of the square in R_4 output by Step 4(a), i.e., $\Delta(x, y) = (-m/2-1, 2)$. Step 4(b) outputs squares in L_1 and a square in L_4 at the end of the loop or, in the last execution of the loop, it outputs the square in L_5. The order is the same as specified in L_1.

The number of outputs of the algorithm is:

$$\begin{matrix} (n-2) + (m/2-1)(n/2+n/2-1) + \\ (n-1) + (m/2-1)(n/2-1+n/2) \end{matrix} = mn - m - 1,$$

which is equal to t, as desired. □

6 Open Problems

A few interesting problems in this space remain open:

1. Close the gap between $2^{\Omega(\sqrt{t})}$ and $2^{O(\sqrt{t \log t})}$ for the number of reachable configurations in 2048 without merging.

2. Are there examples where all t tiles permute in a single cycle, for even t? (Our construction works only for odd t.)

3. Can Theorem 5 be improved, that is, is it possible to decide in $o(st)$ time whether any pair of s special tiles among t total tiles in a given configuration can be made adjacent?

4. What happens in higher dimensions, such as 3D, where Lemma 2 no longer holds? (This question was posed by Martin Demaine in 2018.)

Acknowledgments

We thank Fae Charlton, Martin Demaine, and Leonie Ryvkin for helpful discussions on this topic.

References

[AAD16] Ahmed Abdelkader, Aditya Acharya, and Philip Dasler. 2048 without new tiles is still hard. In Erik D. Demaine and Fabrizio Grandoni, editors, *Proceedings of the 8th International Conference on Fun with Algorithms*, volume 49 of *LIPIcs*, pages 1:1–1:14, 2016.

[BDF+19] Aaron T. Becker, Erik D. Demaine, Sándor P. Fekete, Jarrett Lonsford, and Rose Morris-Wright. Particle computation: complexity, algorithms, and logic. *Natural Computing*, 18(1):181–201, 2019.

[BGC+20] Jose Balanza-Martinez, Timothy Gomez, David Caballero, Austin Luchsinger, Angel A. Cantu, Rene Reyes, Mauricio Flores, Robert T. Schweller, and Tim Wylie. Hierarchical shape construction and complexity for slidable polyominoes under uniform external forces. In *Proceedings of the 31st ACM-SIAM Symposium on Discrete Algorithms*, pages 2625–2641, 2020.

[BLC+19] Jose Balanza-Martinez, Austin Luchsinger, David Caballero, Rene Reyes, Angel A. Cantu, Robert T. Schweller, Luis Angel Garcia, and Tim Wylie. Full tilt: Universal constructors for general shapes with uniform external forces. In *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms*, pages 2689–2708, 2019.

[Cir14] Gabriele Cirulli. 2048. Github repository, 2014. https://github.com/gabrielecirulli/2048. Playable version at https://play2048.co/.

[Dab89] Hédi Daboussi. On the prime number theorem for arithmetic progressions. *Journal of Number Theory*, 31(3):243–254, 1989.

[JJ98] Gareth A. Jones and J. Mary Jones. Divisibility. In *Elementary Number Theory*, pages 1–17. Springer, London, 1998.

[LU18] Stefan Langerman and Yushi Uno. Threes!, fives, 1024!, and 2048 are hard. *Theoretical Computer Science*, 748:17–27, 2018.

[Nic13] Jean-Louis Nicolas. On Landau's function $g(n)$. In Ronald L. Graham, Jaroslav Nešetřil, and Steve Butler, editors, *The Mathematics of Paul Erdős, I*, Algorithms and Combinatorics, pages 207–220. Springer, 2013.

[Tao09] Terence Tao. The prime number theorem in arithmetic progressions, and dueling conspiracies. Blog post, 2009. https://terrytao.wordpress.com/2009/09/24/the-prime-number-theorem-in-arithmetic-progressions-and-dueling-conspiracies/.

[Wik20] Wikipedia. 2048 (video game). https://en.wikipedia.org/wiki/2048_(video_game), 2020.