# 24h Challenge:
# The solitaire problem

Emanuele Renzoni

October 2025

## 1 Introduction and Motivations

It all began on a Thursday afternoon, after a long day of university lectures, when I noticed one of my classmates playing a simple solitaire game with a real deck of cards. The rules were straightforward: the deck consisted of several sets of identical values, and the goal was to sequentially uncover and match cards according to specific conditions, earning points for every valid move until no further matches were possible.

Observing him play, I started to wonder whether it might be possible to achieve the *theoretical maximum score* in such a game. After a short period of reasoning, I concluded that it was fundamentally impossible — no matter how large the deck was, once only five cards remained, there were no natural numbers or move sequences capable of "winning" all of them simultaneously.

This simple observation sparked a deeper curiosity in me: given a deck of size $N$, what is the maximum score that can theoretically be achieved? That question occupied my thoughts for the rest of the day, pushing me to formalize the game's mechanics, define a scoring model, and design an algorithmic strategy to approximate the optimal configuration.

What began as a casual observation of a card game quickly evolved into a small research challenge — a blend of combinatorics, optimization, and computational experimentation. This project, though modest, represents my passion for mathematics and curiosity-driven reasoning, and how they allow me to approach even seemingly trivial problems with structure, precision, and intellectual rigor.

## 2 Formalization of the Problem

We consider an abstract deck composed of $N$ distinct values (labels $1, 2, \ldots, N$), each appearing exactly four times, for a total of $4N$ cards.
The initial configuration of the deck is represented by a permutation of this multiset.

**Deck and configuration space:**
Fix an integer $N \geq 1$ and let the alphabet of card values be

$$\Sigma = \{1, 2, \ldots, N\}.$$

The deck is the multiset

$$M = \{1^4, 2^4, \ldots, N^4\},$$

containing exactly four copies of each value in $\Sigma$ (hence $4N$ cards in total).
An initial configuration is an ordered sequence

$$A = (a_1, a_2, \ldots, a_{4N}) \in \Sigma^{4N}.$$

The set of all admissible initial configurations is

$$\Omega = \Big\{(a_1, \ldots, a_{4N}) \in \Sigma^{4N} : \ \#\{i : \ a_i = v\} = 4 \ \ \forall v \in \Sigma\Big\}.$$

**Counting elimination rule:**

At the beginning of each round, the current deck is the queue

$$Q = (q_1, q_2, \ldots, q_m),$$

with $q_1$ at the top. We scan positions $j = 1, 2, \ldots, m$ and remove the *first* index $j$ such that

$$q_j = j.$$

When such $j$ exists (a *match*), we eliminate $q_j$ and cyclically move the preceding block $(q_1, \ldots, q_{j-1})$ to the bottom (relative order preserved), obtaining

$$Q' = (q_{j+1}, \ldots, q_m, \ q_1, \ldots, q_{j-1}).$$

If no index $j$ satisfies $q_j = j$, the process terminates. **Round position and score.** If a match occurs at position $j$, we set $k_i := j$ for that round $i$ and assign the round reward

$$r_i = k_i = j \quad \text{(since } q_j = j \text{ under this rule).}$$

Let $T(A)$ be the number of successful eliminations starting from $A$ (so $0 \leq T(A) \leq 4N$). The total score is

$$F(A) = \sum_{i=1}^{T(A)} r_i = \sum_{i=1}^{T(A)} k_i.$$

**Deterministic dynamics:**
Given the initial order of the deck, this rule defines a completely deterministic process: at each round i, the count proceeds deterministically according to the self-counting rule $q_j = j$; the first matching card (if any) is eliminated. Since each value appears exactly four times, at most $4N$ eliminations can occur in total. The entire sequence of eliminations and the total score are thus uniquely determined by the initial permutation.

**Objective:**
Each possible initial order $A$ of the deck produces a unique total score $F(A)$ obtained by summing all target values successfully eliminated during the process. The optimization problem is to choose the initial permutation $A$ that maximizes this total score:

$$\max_{A \in \Omega} F(A),$$

where $\Omega$ is the set of all distinct permutations of the multiset $\{1^4, 2^4, \dots, N^4\}$.

**Observation on the "Last 5 cards":**
Under this elimination rule, the only way to remove the very last card is if the target at that moment equals 1 (an ace). Iterating this reasoning backwards, the only possible configuration that allows the complete removal of the last four cards is that they are all aces, since the four final targets must find the remaining four aces still in the deck. However, when only five cards remain, there are at most four aces available in total. Therefore, it is *impossible* to win the game completely from a configuration with five remaining cards.

## 3 Mathematical Definition of the Problem

We consider a deck composed of $N$ distinct values (labels $1, 2, \ldots, N$), each appearing exactly four times. The deck can therefore be represented as the multiset

$$M = \{1^4, 2^4, \ldots, N^4\}.$$

A specific arrangement of the deck is an ordered sequence

$$A = (a_1, a_2, \ldots, a_{4N}),$$

which we call the *initial configuration.*

The set of all possible configurations, that is, all distinct permutations of the multiset $M$, is denoted by

$$\Omega = \Big\{(a_1, \ldots, a_{4N}) \in \Sigma^{4N} \ : \ \#\{i : a_i = v\} = 4 \quad \forall v \in \{1, \ldots, N\}\Big\}.$$

Each element $A \in \Omega$ corresponds to a valid initial deck ordering.

**Game dynamics:**
The game proceeds through discrete rounds indexed by $i = 1, 2, 3, \ldots$.
At the beginning of each round, the player starts *counting from 1* while scanning the deck from the top. The deck at the start of round $i$ is represented by an ordered sequence

$$Q_{i-1} = (q_1, q_2, \ldots, q_m),$$

where $m$ is the number of remaining cards and $q_1$ represents the top of the deck.

During the counting process, each card $q_j$ is associated with the current count number $c = j$. If, while counting, a card satisfies the condition

$$q_j = c,$$

then that card is immediately removed from the deck, and the round ends.
All the cards that have been counted but not eliminated, namely $(q_1, q_2, \ldots, q_{j-1})$, are moved to the end of the deck in the same relative order.
If no such index $j$ exists (that is, no card satisfies $q_j = j$), the game terminates.

Formally, the position of the eliminated card during round $i$ (if any) is

$$k_i = \min\{j \in \{1, \dots, m\} \mid q_j = j\}.$$

The deck transition rule is therefore

$$Q_i = \begin{cases} (q_{k_i+1}, \dots, q_m,\, q_1, \dots, q_{k_i-1}), & \text{if such } k_i \text{ exists,} \\ Q_{i-1}, & \text{otherwise.} \end{cases}$$

In the first case, the eliminated card $q_{k_i}$ awards a score equal to its face value $q_{k_i} = k_i$. In the second case (no match found), the configuration remains unchanged and the game stops.

**Total score:**
Each successful elimination contributes a number of points equal to the position at which the match occurred. The total score obtained from an initial configuration $A$ is

$$F(A) = \sum_{i=1}^{T(A)} r_i,$$

where $r_i$ is the reward of round $i$, given by

$$r_i = \begin{cases} k_i, & \text{if a match occurs at position } k_i, \\ 0, & \text{if no match occurs.} \end{cases}$$

The process terminates after $T(A)$ successful eliminations, or sooner if no card satisfies the matching condition. Since each value appears exactly four times, each card can be removed at most once, and therefore $0 \le T(A) \le 4N$, ensuring that $F(A)$ is finite.

**Optimization problem:**
The objective is to determine the initial ordering of the deck that maximizes the total achievable score:

$$\boxed{\max_{A \in \Omega} F(A)}.$$

A configuration is said to be *perfect* if it allows the elimination of all $4N$ cards, in which case the player achieves the theoretical upper bound

$$B(N) = 4 \sum_{i=1}^{N} i = 2N(N+1)$$

corresponding to a complete resolution of the deck with every round producing a successful match.

## 4 Five-card impossibility under the counting rule

**Lemma 1 (Terminal structure for $k \leq 4$).**
Before proving the main theorem, we first characterize all configurations from which the game can be completely resolved when only a few cards remain. The following result shows that, for up to four remaining cards, completion is possible if and only if all cards are aces.

$$\text{any configuration } Q = (q_1, \ldots, q_k),\ 1 \leq k \leq 4,\ \text{is completable} \iff q_j = 1\ \forall j.$$

*Proof:*

We proceed by induction on $k$.

**Base case ($k = 1$).**
A single card $(q_1)$ is removed if and only if

$$q_1 = 1$$

Hence the statement holds for $k = 1$.

**Inductive step:**
Assume the statement holds for every configuration of length strictly less than $k$, where $2 \leq k \leq 4$. We show it also holds for length $k$. Let $Q = (q_1, \ldots, q_k)$ be completable. By the rules of the game, there exists $j \in \{1, \ldots, k\}$ such that $q_j = j$ and

$$Q' = (q_{j+1}, \ldots, q_k, q_1, \ldots, q_{j-1})$$

is the residual configuration after the first successful removal. Since $Q$ is completable, so is $Q'$, and $|Q'| = k - 1$. By the inductive hypothesis,

$$q'_r = 1 \quad \forall r \in \{1, \ldots, k-1\},$$

where $(q'_1, \ldots, q'_{k-1})$ denote the elements of $Q'$ in order. However, $Q'$ is a rotation of $Q$ with one element removed. In particular, $Q'$ contains all elements of $Q$ except $q_j$, preserving their relative order. Thus, for $Q'$ to consist entirely of 1's, all surviving elements of $Q$ must equal 1. If $q_j \neq 1$, then $Q$ contains at least one non-unit value and cannot produce an all-1 residual sequence after the first removal, contradicting the inductive hypothesis. Hence $q_j = 1$ as well, and therefore

$$q_1 = q_2 = \cdots = q_k = 1.$$

Conversely, if $Q = (1, 1, \ldots, 1)$, then at each round the first card satisfies $q_1 = 1$ and is removed, leading to the sequence $(1, \ldots, 1)$ of length $k - 1$, which by induction is completable. Thus the statement holds for all $k \leq 4$, in accordance with the deck composition rule that allows at most four aces.

**Theorem (Five-card impossibility):**
It is impossible to complete the game starting from any configuration containing exactly five cards.
*Proof:*
Assume by contradiction that some 5-card sequence

$$Q = (a, b, c, d, e)$$

can be completely resolved. During the first of the final five rounds, a card is removed, leaving a 4-card configuration $Q'$.
By Lemma 1, this residual configuration must be

$$Q' = (1, 1, 1, 1).$$

Hence, before that last removal, the multiset of the five remaining cards must have been

$$\{1, 1, 1, 1, x\},$$

where $x$ is the value of the card removed during that round. If the first card equals 1, the first round removes it and the remaining four are $(1, 1, 1, 1)$; thus the original configuration would have been $\{1, 1, 1, 1, 1\}$, which is impossible because each value appears at most four times in the entire deck.

If the first card is not 1, then the first successful removal occurs at some position $j \in \{2, 3, 4, 5\}$ with $q_j = j$. After eliminating that card, the residual 4-card state is a rotation of the set $\{a, b, c, d\}$ that still contains a non-1 entry.
By Lemma 1 such a configuration cannot be completed, giving a contradiction.

Therefore, no configuration with five remaining cards can be completely resolved.

**Corollary**:
Since the deck contains at most four copies of each value, in particular at most four aces, a configuration with five cards remaining can never satisfy the necessary condition for completion. The *five-card barrier* thus represents an intrinsic structural limit of the counting dynamics:

*If the game is completable, then the final tail length satisfies $k \leq 4$.*
*Conversely, if $k \leq 4$ and all the last $k$ cards equal 1, the game completes.*

## 5 Reflections and the search for a constructive algorithm

The reasoning developed so far originated almost by chance during a dinner conversation with a university colleague. After formalizing the counting rule and recognizing the intrinsic limit imposed by the five-card barrier, we began to wonder about the vast landscape of possible initial configurations and their corresponding scores. How many distinct orderings of an $N$-card deck can lead to a maximal outcome? Does a theoretical maximum score actually exist for each $N$, and, if so, does it depend smoothly on the size of the deck or fluctuate unpredictably? These questions, simple in appearance yet profoundly combinatorial in nature, opened an entirely new line of thought.

Lacking computers or formal tools at that moment, we took a restaurant napkin and began sketching sequences by hand for the smallest decks: first for $N = 1$, containing only aces, then for $N = 2$ (aces and twos), and finally for $N = 3$ (aces, twos, and threes). After several manual simulations, an intriguing pattern emerged: excluding the degenerate case of the four aces, the maximum achievable score was consistently equal to the theoretical upper bound minus two points. This observation suggested a structural gap—an inherent inefficiency in the counting dynamics—that prevents even optimal configurations from reaching the absolute theoretical maximum.

Encouraged by this small but concrete discovery, we started to imagine how an algorithm might systematically explore the enormous configuration space, searching for orderings that either attain or approach the best possible score. Such an algorithm would have to balance exhaustive enumeration with heuristic reasoning, trading completeness for efficiency in a problem that grows combinatorially even for modest values of $N$. The question then became not merely one of enumeration, but of optimization: how to construct or approximate an arrangement that pushes the limits of what the counting rule allows, and whether the observed deficit of two points persists for all $N$ or fades asymptotically with deck size.

## 6 First version of the combination test (V1)

The first implementation of the combinatorial test, referred to as *Version 1*, was designed to systematically explore all distinct permutations of a multiset representing the deck up to $N = 4$. The deck is constructed as

$$\{1^4, 2^4, 3^4, 4^4\},$$

and for each permutation the program simulates the game dynamics according to the counting rule described in Section 2. The algorithm removes the first $q_j = j$, accumulating points equal to the index of the successful match. After each removal, the array is cyclically rotated so that the counting always restarts from the card immediately following the removed one. If no match is found for a number of consecutive steps exceeding a threshold (in this version, four misses), the run is stopped and the partial score is recorded.

The program exhaustively evaluates all distinct permutations using the `distinct_permutations` function from the `more_itertools` library, identifying the sequence that yields the highest possible score for each $N$.
Version 1 successfully resolves all cases for $N = 1, 2, 3, 4$, finding for each the best ordering of the multiset and its corresponding score. Beyond $N = 4$, the factorial growth of the search space makes full enumeration impractical, thus motivating the development of stochastic and heuristic approaches in later versions.

The complete implementation of *Test Combinazioni V1* can be found on the project's GitHub repository:

github.com/V1

## 7 Empirical results and the emergence of a two-point gap after the V1 implementation:

An unexpected yet highly consistent result emerged from the exhaustive enumeration performed by *Test Combinazioni V1.* For the cases $N = 3$ and $N = 4$, the algorithm consistently found at least one initial sequence that achieved a final score equal to the theoretical maximum minus two points, that is:

$$F_{\max}(N) = B(N) - 2.$$

This pattern held across all optimal sequences identified, regardless of the specific ordering of cards leading to the solution. In both cases, every configuration that reached the end of the game with a single card remaining yielded the same two-point deficit relative to the theoretical upper bound $B(N)$, defined as the sum of the deck sizes at all successful rounds up to termination.

This empirical regularity led us to hypothesize that the "minus two" gap could be a *structural property* of the counting process itself. In other words, no matter how large $N$ becomes, the self-counting dynamics might inherently prevent a player from achieving the absolute theoretical maximum, forcing a shortfall of exactly two points in every optimal configuration. Such a conjecture bridges empirical observation and formal combinatorial reasoning: the constraint emerges not from randomness or algorithmic limitation, but from the intrinsic structure of the rule that restarts the count at each turn.

Encouraged by this finding, we began exploring whether a general proof could be established, showing that for any $N \geq 2$ there always exists at least one sequence that attains $B(N)-2$, and that no configuration can exceed this bound. This question marks the transition from computational exploration to the search for a formal, purely mathematical explanation of the phenomenon.

## 8 Second version of the combination test (V2)

After the limitations observed in the exhaustive approach of *Version 1*, a new implementation was developed with the goal of scaling the search process beyond $N = 4$. The second version, referred to as *V2*, introduces a stochastic optimization strategy based on *hill-climbing* and multi-start random exploration. Instead of enumerating every distinct permutation, the algorithm generates random configurations of the deck and incrementally improves them through local perturbations, accepting only those changes that do not decrease the current score. This modification allows the search to efficiently explore a much larger configuration space while avoiding the factorial explosion of the previous version.

Each local move is implemented either as a *swap* between two random positions or as a *block move* in which a small contiguous segment of length 2–4 is displaced elsewhere in the sequence. These two complementary operations guarantee both local fine-tuning and global re-shuffling of the deck structure. For every randomly generated starting permutation, the algorithm performs a series of local improvements (up to a fixed number of iterations), restarting from new random configurations multiple times. An *early-stop* condition interrupts the search as soon as a target score is reached, ensuring that computational time is spent only on potentially optimal regions of the search space.

This approach proved capable of solving cases up to $N = 6$ and, despite the stochastic nature of the algorithm, consistently rediscovered the same "minus-two" gap observed in Version 1. The use of controlled randomness and local optimization not only improved scalability but also strengthened the empirical evidence that the $B(N) - 2$ bound is a structural property of the system rather than a numerical coincidence.

*Note:*
To efficiently climb the score landscape and avoid premature convergence, the algorithm's parameters must be balanced between exploration and exploitation. Empirically, smaller *local move radii* (segment lengths of 2–4) combined with a moderate number of iterations per restart (typically $10^3$–$10^4$) yield the best trade-off between convergence speed and solution diversity. A higher number of random restarts increases the chance of escaping local optima, while too many iterations within a single run often lead to redundant exploration around a sub-optimal peak.

The complete implementation of *Test Combinazioni V2* is available on the project's GitHub repository:

github.com/V2

## 9 Third version of the combination test (V3):

The third iteration of the algorithm represents a major improvement in both scalability and computational efficiency. While *Version 2* already introduced stochastic exploration through random restarts and hill-climbing, *Version 3* extends this idea to a fully parallel architecture, exploiting all available CPU cores through Python's `multiprocessing` library. Each process executes an independent sequence of stochastic searches, each consisting of randomized restarts followed by localized hill-climbing refinement, and communicates progress to a shared master process via asynchronous queues. A global synchronization event allows all workers to stop cooperatively as soon as one process reaches the target score, preventing unnecessary computation and enabling near-linear scalability with the number of cores.

From a methodological point of view, *V3* maintains the same search logic as *V2* but distributes the workload across multiple processes, allowing the algorithm to test hundreds of thousands of random initial configurations in parallel. The combined use of asynchronous communication, distributed early-stop, and dynamic load balancing ensures that each process explores different regions of the configuration space, maximizing diversity while minimizing redundant evaluations. This design significantly accelerates convergence, enabling successful optimization up to $N = 7$ — a result unattainable by the previous sequential implementation.

In practice, the transition from a single-core to a multi-core setup increases the effective exploration rate by roughly a factor equal to the number of logical cores available, making it possible to perform large-scale stochastic searches that would otherwise be computationally prohibitive. Thus, *Version 3* not only improves performance quantitatively, but also represents a qualitative leap toward a distributed search paradigm, transforming the original algorithm into a parallel stochastic optimization framework.

The complete implementation of *Test Combinazioni V3* is available on the project's GitHub repository:

github.com/V3

## 10 Performance comparison between algorithmic versions:

The progressive development from *Version 1* to *Version 3* marks a clear evolution in both algorithmic design and computational efficiency. *Version 1*, based on full combinatorial enumeration, was able to completely explore the configuration space only up to $N = 4$. Although exact and deterministic, its factorial complexity

$$O(\frac{(4N)!}{(4!)^N})$$

quickly rendered the approach infeasible for larger decks, as the number of distinct permutations grows factorially with $N$.

*Version 2* replaced exhaustive enumeration with a stochastic hill-climbing strategy augmented by random restarts and local swaps, reducing the practical time complexity from factorial to approximately

$$O(K \cdot L),$$

where $K$ is the number of restarts and $L$ the number of local iterations per run. This shift allowed the algorithm to reach $N = 6$ within reasonable computation time while maintaining a consistent detection of the two-point gap structure. However, since all runs were still executed sequentially, the algorithm's total runtime remained proportional to the number of independent trials.

*Version 3* addressed this limitation through full parallelization using Python's `multiprocessing` package. Each CPU core runs an independent stochastic process, performing randomized exploration and communicating results asynchronously to the main thread. The effective runtime therefore scales approximately as

$$T_{\mathrm{V3}} \approx \frac{T_{\mathrm{V2}}}{P},$$

where $P$ is the number of logical CPU cores available. Empirical testing confirmed near-linear speed-up for $P \leq 8$, and substantial improvements even beyond that when combined with cooperative early-stop synchronization.

## 11 Motivation for a general proof and higher-$N$ cases:

At this stage, the computational evidence leaves little doubt that the maximum attainable score for the counting solitaire is always equal to the theoretical upper bound minus two, that is, $F_{\max}(N) = B(N) - 2$. Nevertheless, what we still lack is a rigorous mathematical proof of this property. The goal of my ongoing research is therefore twofold: on one hand, to formalize and generalize the combinatorial structure of the problem, and on the other, to construct an explicit example of an optimal sequence for the classical 40-card Italian deck, corresponding to $N = 10$. Such a configuration would not only provide a convincing demonstration of the theory, but also serve as a tangible solution to share with the colleague with whom this entire investigation first began.

Due to the computational constraints of my hardware and the factorial growth of the search space, the stochastic algorithms developed so far have successfully found optimal sequences only up to $N = 7$. Beyond that threshold, particularly for $N = 8$ and higher, the search becomes prohibitively time-consuming even with parallel processing. Proving the existence of an optimal sequence for $N = 10$ (the analogue of the full 40-card deck) would not only complete the challenge but also provide crucial insight into the structural reason behind the two-point deficit. Such a proof could, in turn, suggest a more analytical method to generate near-optimal or even optimal sequences directly, without relying on large-scale random exploration.

Intuitively, the persistent "minus two" gap can be understood as a natural consequence of the counting rule: while aces (1's) can always be removed immediately when they appear at the top of the deck, the smallest remaining value that inevitably escapes elimination at the end of the process is the 2. In the next section, I will attempt to provide a formal demonstration of the existence of a sequence that achieves this outcome for any deck size $N$, showing that the process necessarily converges to a terminal configuration with a single card of value 2.

*Computational evidence:*
Our experiments up to $N = 7$ consistently find initial configurations whose evolution ends in the single-card state (2), attaining the benchmark $B(N) - 2$ in all tested cases. This *strongly suggests* that the phenomenon persists for all $N$, but a general proof remains open (see the Conjecture in the next section)

## 12 Existence conjecture:

In this section we make precise what is *proved* and what is still a *conjecture*. We work under the counting rule "remove the first $q_j$ with $q_j = j$", moving the preceding block to the bottom.

**Goal:**
For any integer $N \geq 2$ we aim to exhibit an ordering $A = (a_1, a_2, \dots, a_{4N})$ of the multiset $\{1^4, 2^4, \dots, N^4\}$ such that applying the counting rule to $A$ results in the terminal configuration $Q_{\text{final}} = (2)$.

**Lemma 2 (Backward-safe inverse step):**
Let $S = (s_1, \dots, s_{m-1})$ be a valid post-elimination configuration of length $m-1$. For any $j \in \{1, \dots, \min\{N, m\}\}$, write

$$S = (\, S' \,\|\, T \,), \quad |T| = j - 1,\ |S'| = m - j,$$

where $T$ is the suffix of $S$ of length $j - 1$. If

$$T[p] \neq p \quad \forall\, p \in \{1, \dots, j-1\},$$

then the sequence $Q_{\text{prev}} := (\, T,\, j,\, S' \,)$ is a valid configuration such that one application of the counting rule removes the card $j$ at position $j$ and produces exactly $S$ as the next state. When the counting rule is applied to $Q_{\text{prev}}$, the first $j-1$ counts scan the prefix $T$. By the non-coincidence assumption on $T$, none of these positions satisfies $q_i = i$, hence no removal occurs before position $j$. At position $j$ the card value equals the count ($q_j = j$), so the card is removed; the block $T$ is cyclically moved to the bottom preserving its order, producing $(S' \,\|\, T) = S$.

**Construction (backward synthesis, safe version):**
Fix $N \geq 2$. We build a candidate configuration $A^{(N)}$ of length $4N$ by iteratively applying Lemma 2 backward in time, starting from the target state (2).

1. Initialize $S_1 = (2)$ and set the multiplicities
$$m_v = \begin{cases} 3, & v = 2, \\ 4, & v \in \{1, 3, \ldots, N\}, \end{cases}$$
since one 2 must remain at the end.
2. For $t = 1, 2, \ldots, 4N - 1$:
    (a) Let $\ell = |S_t|$ and let $r_t \geq 0$ be the length of the maximal *alignment run* in the tail of $S_t$, i.e. the largest $r$ such that the last $r$ entries of $S_t$ equal $(1, 2, \ldots, r)$.
    (b) Choose $j \in \{1, \ldots, \min\{N, \ell + 1\}\}$ with $m_j > 0$ so that the suffix of $S_t$ of length $j - 1$ satisfies the non-coincidence condition of Lemma 2. *Safe choice:* if possible, pick any $j \geq r_t + 2$ with $m_j > 0$ (see Lemma B below), which guarantees the non-coincidence.
    (c) Define $S_{t+1} = (T, j, S')$ as in Lemma 2 and decrement $m_j \leftarrow m_j - 1$.
3. After $4N - 1$ iterations, output $A^{(N)} := S_{4N}$.

By construction, running the counting rule forward on $A^{(N)}$ produces the sequence of states $S_{4N}, S_{4N-1}, \ldots, S_1 = (2)$.

**Lemma B (Sufficient availability condition):**
Let $S$ have length $\ell$ and let $r \geq 0$ be the maximal alignment run in its tail (the final block $(1, 2, \ldots, r)$, possibly with $r = 0$). For any $j \geq r + 2$ with $m_j > 0$, the suffix $T$ of $S$ of length $j - 1$ is *non-coincident*, hence the inverse step of Lemma 2 is admissible.

**What remains open (global existence):**
The previous version incorrectly argued that an admissible $j$ *always* exists by a counting heuristic. In reality, existence at each step depends both on the tail structure of $S_t$ and on the residual multiplicities $m_j$. Proving that an admissible $j$ exists *at every step for every* $N$ requires an *unconditional availability lemma*, which is currently missing.

**Conjecture (Global existence):**
For every integer $N \geq 2$ there exists at least one initial configuration $A^{(N)}$ such that, under the counting rule, the process terminates with $Q_{\text{final}} = (2)$.

**Remark:**
The text above fully proves the *conditional* statement (correctness of the backward synthesis whenever an admissible choice exists at each step, with a practical sufficient criterion via Lemma B). The *unconditional* existence for all $N$ is stated as a conjecture pending a complete availability lemma.

## 13 Motivation and potential applications.

Beyond the playful origin, this study touches several areas where "count–eliminate–rotate" dynamics arise naturally. First, it offers a compact testbed for *backward synthesis* and *invariant design* in discrete processes, skills that transfer to scheduling on cyclic buffers, round–robin queues, cache/eviction patterns, and pipeline stall analysis. Second, the counting rule connects to themes in combinatorics and dynamical systems—derangement–like avoidance in suffixes, rotor–router/chip–firing–style transports, and structural properties of permutations with local constraints—suggesting possible cross–fertilization with probabilistic coupling and potential–function methods. Finally, the constructive viewpoint ("safe" inverse steps under availability) is algorithmic: it hints at heuristic generators, certificates of correctness, and search–pruning strategies useful in broader sequence–design and verification tasks.

**Epilogue.**

It is, in a sense, the most rewarding outcome: a problem posed in jest became a serious object of inquiry. We decomposed it, isolated the truly hard point (the unconditional availability at each backward step), and pushed a clean line between what is *proved* and what is *conjectured*. For now, we have *explicit* sequences up to $N = 7$ that evolve to the single–card state (2), and a logically sound conditional framework (backward synthesis + local sufficiency via the tail–run criterion). We leave the full resolution to an interested reader: closing the availability gap would elevate a playful puzzle to a tidy theorem—and, perhaps more importantly, it showcases how curiosity and careful structure can turn a game into mathematics worth studying.