1.

AN EXPLORATION OF PERFECT FARO SHUFFLES

SIDDHARTHA GUPTA, ANIRUDH ARUNPRASAD

1. INTRODUCTION

Card shuffling is a craft that has captivated magicians, casinos, and cardists for ages. The art of deck manipulation has intrigued those familiar with cards to the point where hundreds of different methods of card manipulation have been perpetuated over the last century. Of these methods, the Faro shuffle, or the "perfect shuffle" stands out as one of the more unique types of card manipu\lation. Unlike a traditional "riffle shuffle," the Faro perfectly interlaces the cards from the halves of the deck. Other shuffles like a non-perfect riffle may give 2 or 3 cards in one clump, randomizing the deck. The Faro shuffle has an interesting property: 8 "normal" Faro shuffles performed on a standard deck of 52 cards, will return that deck to its original state. The goal of this paper is to examine why perfect shuffles have this property, and what other properties they may have.

In this paper, we'll explore various ways the position of a card in a given deck can be modeled as perfect shuffles are carried out. We'll also develop formulae to model the cyclical path of cards and card trajectory as the deck is perfectly shuffled. To conclude, we'll model several theorems for decks of any size.

Abstract

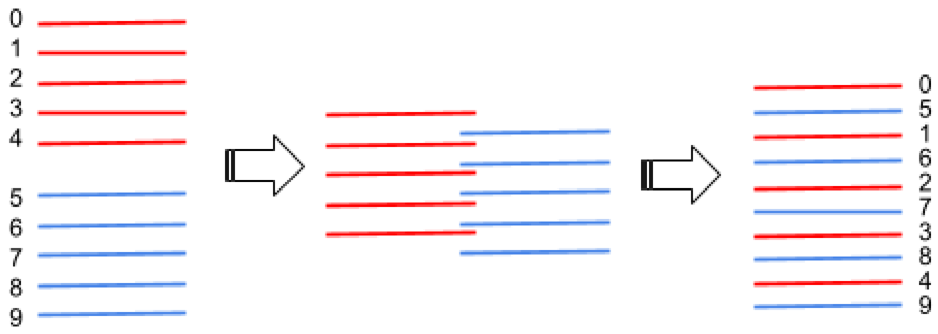
Decks of cards seem like casual objects, but in the early 18th century, a unique type of shuffle called a perfect or Faro shuffle was created. This shuffle returns a previously unshuffled deck towards again being unshuffled after 8 shuffles, and though this seems like a simple coincidence, the Faro shuffle is a special sorting algorithm that holds a similar restorative property for every deck size. Previously on the topic, researchers have recorded the equations for Faro shuffles as a conditional modulus function, which could also be represented in piecewise or floor/ord notation. Researchers have established two different types of faro shuffles: out shuffles, which we call normal shuffles, and in shuffles, which we call alternate shuffles. Shuffle numbers are denoted by binary representation using cycle and orbital context of given shuffle paths. Algorithms have also been found to control card position through binary tree notation, however we present a more rigorous and simple proof on the subject. Our proof of card I benign forces to position index J provides a special case of a reverse BFS sorting algorithm. Our team used a mix of code for bashing and pattern-finding, and number theory to represent the maximum and minimum shuffle numbers using Euler's totient function. Our method gives a rigorous approach towards proving the existence of resolvable shuffles for all, as well as allowing for specific cases where we guarantee a given occurrence of minimum shuffles. Also, as a deviation into the concept of card cycles, we recognize that future work could be done on a more powerful approach to denoting minimal shuffles as the

LCM of list cycle length for decks of n size. Further expansions on the work could also yield better algorithms using a special case of the discrete logarithm problem.

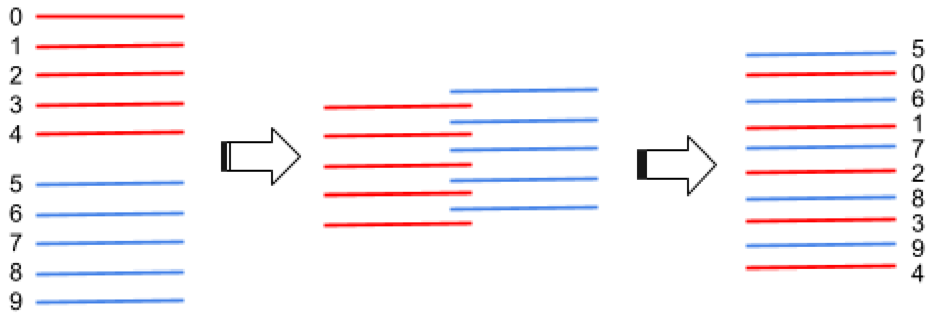
2. FOUNDATIONAL RESULTS AND OBSERVATIONS

To begin, let's distinguish the types of perfect shuffles. We will assume by default that a perfect shuffle will be a Normal Shuffle unless otherwise stated.

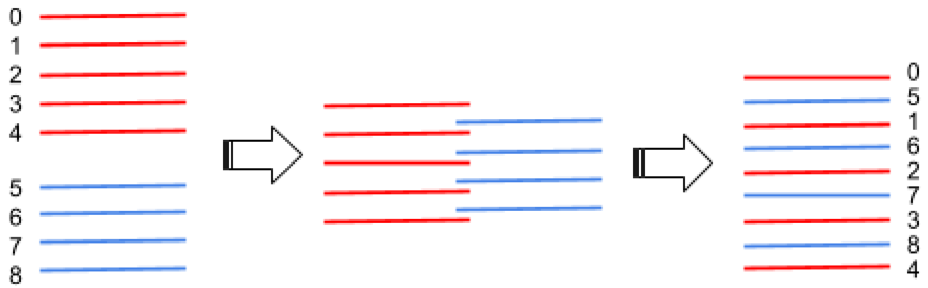
Definition: A ***Normal shuffle*** cuts and interlaces the two halves of the deck perfectly, with the top card of the top half of the original deck occupying the top position of the newly shuffled deck.



Definition: An ***Alternate Shuffle*** cuts and interlaces the two halves of the deck perfectly, with the top card of the bottom half of the original deck occupying the top position of the newly shuffled deck.



Definition: An ***Odd Shuffle*** is a shuffle where the deck has an odd number of cards. In a normal odd shuffle, the middle card belongs to the top half of the deck, and the bottom half is interlaced between each space within the top half of the deck. The top and bottom cards on the deck consequently belong to the top half. In an alternate odd shuffle, the middle card belongs to the bottom half of the deck, and the top half is interlaced between each space within the bottom half of the deck. The top and bottom cards on the deck consequently belong to the bottom half. Below is a normal odd shuffle.



We quickly made a conjecture which we later proved to be true: Any deck of n cards has a number of perfect shuffles s, to return to its original state

We will go through the proof after covering other essential theorems and concepts.

The first is a simple Lemma regarding fixed cards.

Lemma 1: The top and bottom cards in an even deck, will never change in position regardless of the number of times the deck is shuffled while doing a normal shuffle

Proof:

Because the top card of a given deck will always go above the top card of the second half of the deck when performing a perfect shuffle, no amount of normal perfect shuffles will be enough to displace the top card.

Likewise, because the bottom card of a given deck must always be below the bottom card of the first half of the deck by definition of a normal shuffle, no card will ever be able to be below the bottom card.

Therefore, both the top and bottom cards of a given deck must remain in their position when doing a normal shuffle.

This fixed card lemma is important because it proposes a different question. It makes us ask what the other cards within the deck do during this shuffle. Below is a simple transcription of a piecewise equation to model shuffling within a card deck.

Piecewise Shuffle Theorem: The new indices of each card can be modeled by the piecewise functions $2i$ for the first half of the deck (including the middle card if the deck has an odd number of cards), and the second half can be found with 2i - n, assuming the indices start with 0.

Proof:

The first step of the perfect shuffle is to *dilate* the deck by multiplying the indices by 2, leaving gaps to interlace the cards. Then, the “stretched out” bottom half of the cards is interlaced into the “stretched out” top half by shifting the indices of the bottom half back n. Therefore, the expression for the new index of the top half is $2i$, and for the bottom half, it is $2i - n$.

This proof should provide us with a simple theorem with which to view the mechanics of the shuffle itself. As we

3. MECHANICS OF SHUFFLING

To truly appreciate the mechanics of a shuffle, we must first understand the correlations between different types of shuffles. Decks of odd numbers are quite intriguing, because, unlike even decks, their bottom card is unconserved. However, this poses an opportunity to visualize odd shuffles more easily:

Steven-Todd Theorem: For a deck of size n, where n is an odd number, the number of shuffles needed to resolve the deck of size *n* will be the same as the number of shuffles needed to resolve a deck of size $(n + 1)$

Proof:

Consider a deck with an even number of cards E. The normal shuffle of such a deck would proceed as follows: The deck would be divided in two, and interlaced as usual. The top and bottom cards would remain consistent regardless of the number of shuffles, or deck size.

Recall that the bottom card of a perfectly shuffled even-sized deck stays consistent. The case for a normally shuffled odd deck is the same as the even deck, except that the bottom card is missing. This is good for our purposes, however. Because the bottom card is always constant, we essentially can ignore it. In a normal shuffle with an even deck of cards, the middle n-2 cards of the deck will be the only cards that are shuffled. The top and bottom simply remain at their positions.

If one card is removed from the deck, then the deck will be odd and the bottom half will no longer have the bottom card. This means that the shuffle will simply be the same as the normal shuffle above it in terms of the number of shuffles. The only difference between the decks is whether the bottom card is present or not.

The odd-even pattern shows us the correlation between similar types of shuffles, however in order to calculate the number of shuffles at all, one requires a useful method. The previous piecewise approximation, though surely important, is not nestable, and therefore it is recursive instead of explicit. In order to fix this, we first have to picture the true arrangement of the cards. The piecewise function acts as a clean signifier of the ability to roll value over after the maximum index of n-1 has been reached, however, there are other functions that are suitable for the purpose. The most useful for our purposes would be a function using the modulo operation.

Card-Stacking Theorem: The Explicit function giving the index i_s after s normal shuffles (assuming normal perfect shuffles), where n (deck size) is an even number, and i_0 is the initial index (assuming indices go from 0 to n-1), can be modeled by:

$$i_s = i_0 * 2^s \, mod \, (n - 1)$$

with the exception of the bottom card, which remains in place instead of going to index 0.

If n is odd, it is:

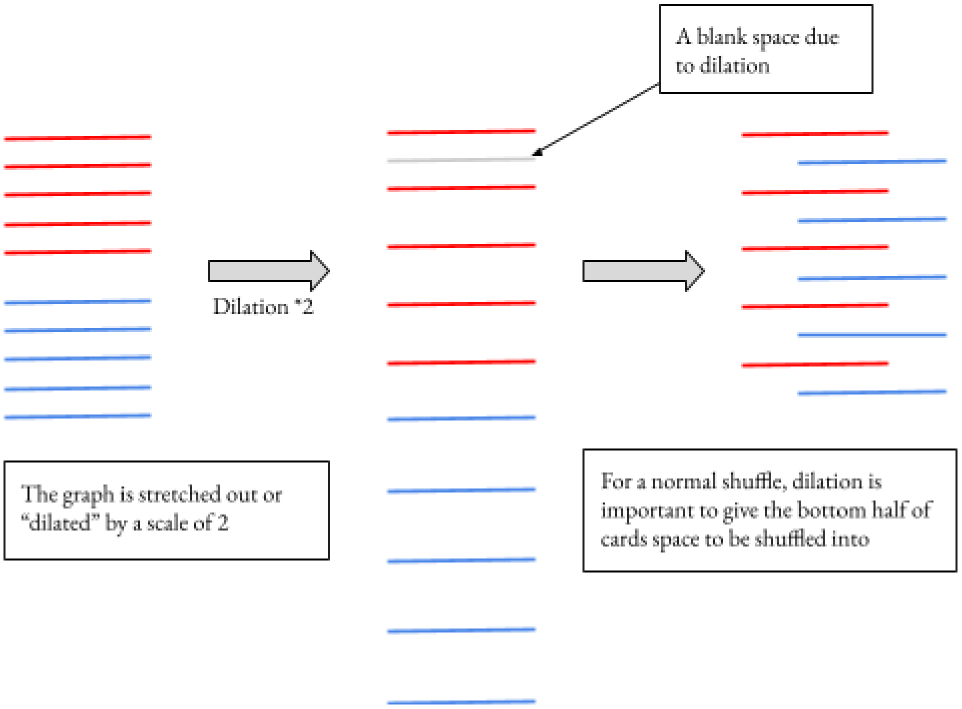
$$i_s = i_0 * 2^s \, mod \, (n)$$

Proof:

Before finding the explicit formula, we can start with a recursive formula. Following the logic of *dilating* the deck, the index is multiplied by 2.

If twice the previous index is out of the range of the deck, then it can be shifted back into the deck either by using piecewise logic like in the Piecewise Shuffle Theorem, or by utilizing modular arithmetic to expand it to a greater number of recursions.

For the even case, this would be modulo $(n - 1)$, and for the odd case, it would be modulo (n). This results in the recursive formula being $i_{s+1} = 2i_s$ mod (n-1) for even deck size, or $i_{s+1} = 2i_s$ mod (n) for odd deck size. Since there is a factor of 2 for each recursion, this can be expressed as exponentiation in the explicit formula.



A blank space due to dilation

The graph is stretched out or "dilated" by a scale of 2

For a normal shuffle, dilation is important to give the bottom half of cards space to be shuffled into

Let q the initial index be i_0, and the number of shuffles, or in other words, the number of recursions applied, is s. The explicit formula is $i_s = i_0 * 2^s \, mod\ (n - 1)$ if n is even, with the exception of the bottom card, which remains in place instead of going to index 0, and $i_s = i_0 * 2^s \, mod\ (n)$ if n is odd.

These ideas are further extended as follows.

Extended Card-Stacking Theorem: The Explicit function giving the index i_s after s alternate shuffles (assuming normal perfect shuffles), where n (deck size) is an even number, and i_0 is the initial index (assuming indices go from 1 to n), can be modeled by:

$i_s = i_0 * 2^s \, mod\ (n + 1)$

with the exception of the bottom card, which remains in place instead of going to index 0.

If n is odd, it is:

$i_s = i_0 * 2^s \, mod\ (n)$

Now, from our original piecewise dynamic, we have a concrete method of shuffle tracking. These functions are incredibly useful, especially because of their ability to be used somewhat explicitly.

4. MINIMUM SHUFFLES

With a concrete understanding of the problems at hand, we now have a

Foolproof Magic Theorem: Any deck of n cards has a number of normal perfect shuffles s, to return to its original state.

Proof:

Let s_m be the minimum number of perfect shuffles for the deck to return to its original state, and i be any given index. If n is even, then by the Card-Stacking Theorem, this would require there to be a value s such that $i = i * 2^s \, mod\ (n - 1)$, which means $2^{\phi(n)} = 1\ mod\ (n)$. Since $n - 1$ is odd, this means 2 and $n - 1$ are coprime.

Therefore, Euler's Theorem can be applied to determine that $2^{\varphi(n-1)} = 1\ mod\ (n - 1)$, where $\varphi(n)$ is the Euler totient function. This shows that $\varphi(n - 1)$ is the maximum possible value for s, and we can conclude that s is a divisor of $\varphi(n - 1)$, or $s \mid \varphi(n - 1)$. If n is odd, then a similar logic can be applied, since 2 and n would be coprime, to determine that $s \mid \phi(n)$.

Another way of expressing the solution is using the number theory function ord$_n$a, which is defined as the smallest positive integer $m \geq 1$ such that $a^m = 1\ mod\ (n)$. For the even case, the solution is s = ord$_{n-1}$2, and for the odd case, it is s = ord$_n$2.

Extended Foolproof Magic Theorem:

Any deck of n cards has a number of alternate perfect shuffles s, to return to its original state.

Proof:

Let s_m be the minimum number of perfect shuffles for the deck to return to its original state, and i be any given index $1 \leq i \leq n$. If n is even, then by the Extended Card-Stacking Theorem, this would require there to be a value s such that $i = i * 2^s\ mod\ (n + 1)$, which means $2^s = 1\ mod\ (n + 1)$. Since $n + 1$ is odd, this means 2 and $n + 1$ are coprime.

Therefore, Euler's Theorem can be applied to determine that $2^{\varphi(n+1)} = 1\ mod\ (n + 1)$, where $\phi(n)$ is the Euler totient function. This shows that $\varphi(n + 1)$ is the maximum possible value for s, and we can conclude that s is a divisor of $\varphi(n + 1)$, or $s \mid \varphi(n + 1)$. If n is odd, then a similar logic can be applied, since 2 and n would be coprime, to determine that $s \mid \varphi(n)$.

Another way of expressing the solution is using the number theory function ord$_n$a, which is defined as the smallest positive integer $m \geq 1$ such that $a^m = 1\ mod\ (n)$. For the even case, the solution is s = ord$_{n+1}$2, and for the odd case, it is s = ord$_n$2.

Despite there not being a generalized solution for s$_m$ (which is interestingly a special case of the unsolved discrete logarithm problem in cryptography), there are special cases of n for which there exist formulas for s$_m$. Here are some which we found:

Odd Power Theorem: If n is an odd power of 2, or $n = 2^p$ where p is odd, then the deck will not go through a reversal state, and the deck will take p shuffles to resolve.

Proof:

Since n is even, by the Card-Stacking Theorem, $i_s = i_0 * 2^s\ mod\ (n - 1)$. Let it cycle in s shuffles ($i_s = i_0$), and since $n = 2^p$, this means that $2^s = 1\ mod\ (2^p - 1)$, which can be expressed as $2^s = 1 + (2^p - 1)k$. Iterate through values of k to find a minimum value for s.

If $k = 0$, then $2^s = 1 \rightarrow s = 0$, which is not a valid solution since that just means not shuffling the deck at all. Next, checking k = 1, $2^s = 1 + (2^p - 1) = 2^p \rightarrow s = p$.

Primitive Deck Theorem: The minimum shuffles for a deck to cycle is $s = \varphi(n - 1)$ for even n or $s = \varphi(n)$ for odd n, if and only if 2 is a primitive root of n-1 is n is even or n if n is odd.

Proof:
Definition 1 Let a, n be integers, n ≥ 1 such that gcd(a,n) = 1. If $\mathrm{ord}_{\mathrm{modulo}}2$ = modulo , we say a is primitive root mod n.

In the Card-Stacking Theorem, we determined that the minimum shuffles s is of the form $\mathrm{ord}_{n-1}2$ for even n and ord_n2 for odd n. If 2 is a primitive root of n-1 for even n or n for odd n, then $\mathrm{ord}_{n-1}2 = \varphi(n-1)$ if n is even or $\mathrm{ord}_n2 = \varphi(n)$ if n is odd.

No simple general formula to compute primitive roots modulo n is known, but one can test whether 2 is a primitive root of n-1 if n is even or n if n is odd by checking the powers of 2 modulo n/n-1 up to $\varphi(n)/\varphi(n-1)$. If it is, then s = φ(n-1) for even n and s = φ(n) for odd n.

5. CARD FORCING

Alternative Primitive Deck Theorem: Using alternate shuffles, any card i can reach any index $1 \le i \le n$ if and only if 2 is a primitive root of n+1 if n is even or n if n is odd.

Proof:
Definition 2 Suppose a and n ≥ 1 are integers such that gcd(a, n) = 1 and a is a primitive root mod n. Then $1, a^1, a^2, \dots, a^{\varphi(n)-1}$ forms a complete reduced residue system mod n.

Definition 3 A complete reduced residue system is a group of integers that contains every integer possible mod n, as in $0, 1, 2, \dots, n - 2, n - 1 \; mod \; n$.

Consider the proof for the even case. Assuming indices are $1 \le i \le n$, $i_s = i_0 * 2^s \; mod \; (n + 1)$, where n is an even number of cards in the deck, i_0 is the initial index, and i_s is the index where it will be after s shuffles. If 2 is a primitive root of n+1, then the minimum shuffles s for the deck to cycle is φ(n+1) since $2^{\varphi(n+1)} = 1 \; mod \; (n + 1)$ by the Extended Card-Stacking Theorem. By Definition 2, the powers of 2 would also form a complete reduced residue system mod n+1. What this means is that for a certain number of shuffles $0 \le s \le \varphi(n+1) - 1$, any index i_0 can reach any index i_s in the deck. $s_m = \phi(modulo)$

However, we may seek a more appropriate and universal approach to more powerfully solve this problem.

Lemma: Cycles

Recall the ability for a normal shuffle to shuffle and then unshuffle any deck of n cards with m minimum shuffles. This ability can be viewed in another way, through the lens of card cycles.

To illustrate, we take a deck of 52 cards and apply a normal shuffle.
The card at i =0, will remain where it is. The card at i = 1, will move to i = 2, then i = 4, then i = 8, then i = 16, then i = 32, and so on until the shuffles return the card to its original position at i = 1 within the deck.

If we transcribe this motion for every card using lists, we see this:

[0] - 1 cycle
[1, 2, 4, 8, 16, 32, 13, 26, 1] - 8 cycle
[3, 6, 12, 24, 48, 45, 39, 27, 3] - 8 cycle
[5, 10, 20, 40, 29, 7, 14, 28, 51] - 8 cycle
[9,18, 36, 21, 42, 33, 15, 30, 9] - 8 cycle
[11, 22, 44, 37, 23, 46, 41, 31, 11] - 8 cycle
[17, 34, 17] - 2 cycle
[19, 38, 25, 50, 49, 47, 43, 35, 19] - 8 cycle
[51] - 1 cycle

()
These are known as cycles.

<u>Definition:</u> **Cycles**

- Cycles are the full-length path of a given set of cards as they are shuffled through a deck.
- For example, the card in indicie 1 will go to 2, 4, 8, 16, etc., and eventually reposition at i = 1 after 8 shuffles.
- Likewise, every other number within the list will follow that cycle until they also reposition themselves at their initial index. For example, the card at i = 2, will be shifted to i = 4, and then to i = 8 and eventually it will reposition itself at i = 2, similarly to each number within the cycle set.
- Cycles are represented by the format: [] x cycle where x is the number of shuffles needed to resolve a cycle. The card values are included in the list

Because each list is essentially a catalog of cards, and because all cards must be included in a list, we know that the total number of cards n in a deck must be because of the total additive of the cycle numbers. In this case, 1+8+8+8+8+8+2+8+1 = 52. Also, we know that a cycle must be resolved. By definition, a cycle must not intersect another cycle, and no two cycles share an element.

There are two possibilities for cycle intersection, and we can easily disprove both.

1. The first one is where a cycle simply fuses with another one. In this case, it would generate a cycle with the additive length of both cycles. However, this would create a bigger cycle, and at that point, it would be easier to label this as a single cycle from the origin instead of 2 which had been fused.
2. The second case is, where a cycle transposes to a different cycle and does not resolve. This is impossible by cycle definition, and because of the earlier proof that for all decks, some number m of minimum perfect shuffles resolve a deck.

Recall that it takes exactly 8 normal perfect shuffles for a standard deck of 52 cards to return to its original state.

A majority of the cycles also end in 8 shuffles, simply because that is the S_m for the deck size of 52. This is because, if the cycle was longer than 8, then by the 8th shuffle each card in the cycle would not return to its initial position. However, the cycle can be shorter as demonstrated by the i = 0, 7, and 51 cycles. However, for these cycles to resolve themselves within 8 shuffles, they must be factors of m which in this case is 8.

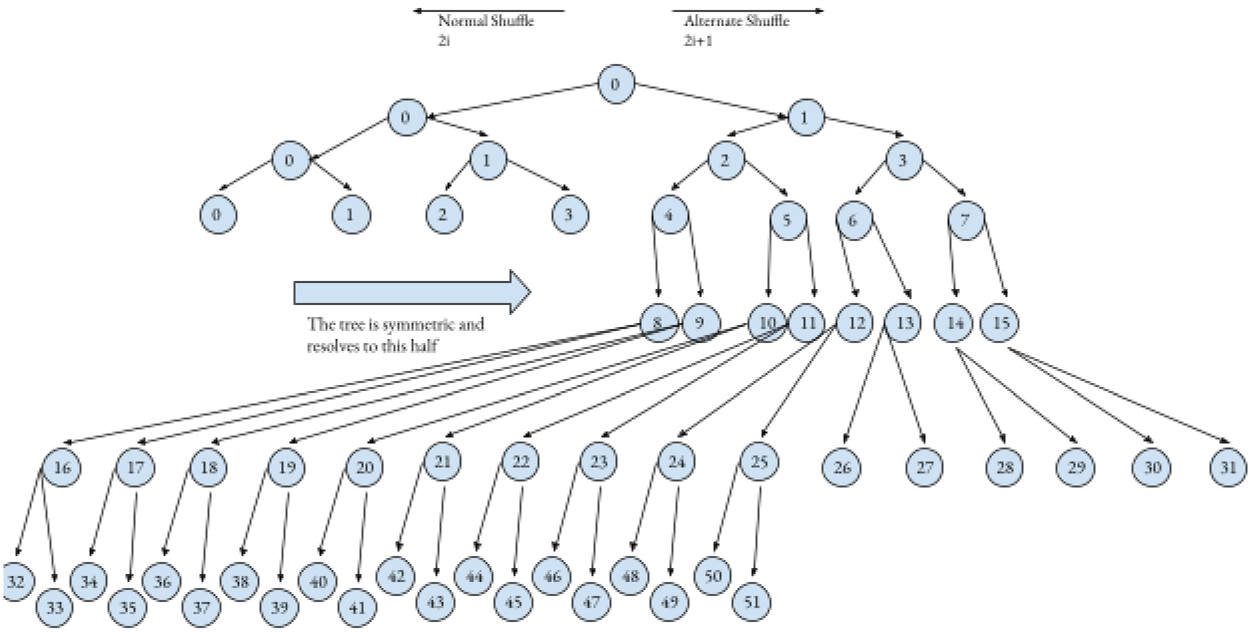
This presents an interesting argument. If cycles must be factors of the minimum number of shuffles m, then the value of m can simply be found by taking the LCM of all cycle numbers.

The most valuable and surprisingly intuitive part of this lemma, for now, will be this resultant statement: Because each cycle is a closed loop, and because each deck must have cycles if a shuffler has an amount of shuffles $\leq$ m, they can force a card to access every indicie within the cycle it is presently occupying. Furthermore, any deck that resolves itself with either normal or alternate shuffling must have at least 1 cycle of cards simply because each card must—after a number of shuffles m—return to its original indices.

Lemma: Binary Tree

Consider the two types of shuffles performable upon a standard deck of cards. We can perform a Normal perfect shuffle or an Alternate perfect shuffle. We can therefore represent this shuffling in the format of a binary tree.

A graphic for 52 cards of represented as follows:



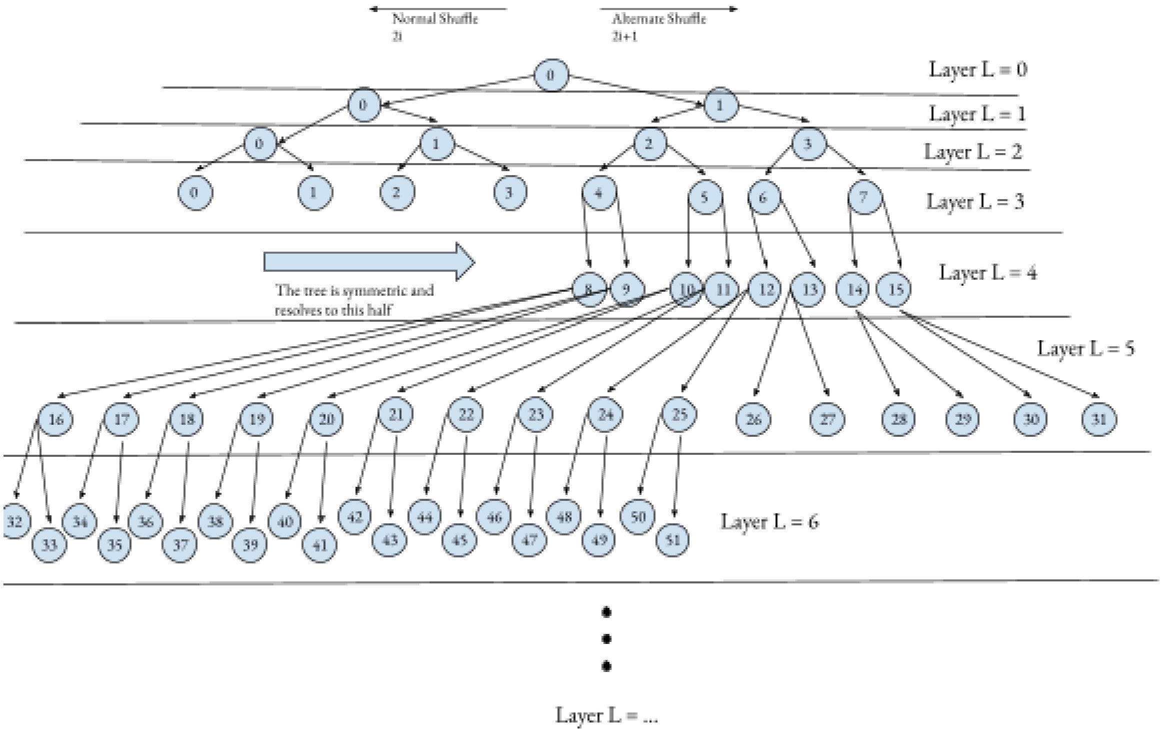
The top number of this deck is at i = 0, and because of this, it has the option of choice. As you can see, all numbers are depicted as being a subchoice of option 0 through a combination of either a normal or alternate shuffle. This works in principle for any deck of size n, because all even numbers and odd numbers can be depicted as 2x and 2x+1 respectively where x is some whole number. By adding 0, you gain 1 and 2, and by inputting those you gain 3, 4, 5, and 6. Recursively, you could find every Natural number for $n \leq \infty$.

This means that to guarantee that a card may be forced to any slot within the deck, one simply has to get their card to i = 0.

Lemma: zero, zero, it's our hero

Recall that our ability to force any card to any location in the deck is easiest when the card starts at index i = 0. This means, that if there is a way to consistently return the card to 0 using a combination of normal and alternate shuffles, then we can completely force the index of a given card.

Let us reframe the problem to make it easier to solve. Instead of the standard binary tree that we had presented previously, let us label each "layer" of the tree.



To move any card index to 0, we will need to iteratively decrease the card's layer. To do this, we will prove that for layer L where L ≥ 1, a card will always be able to travel to layer L-1 using normal or alternate shuffles.

First, take a random node on the tree. The node will have a corresponding layer, but more importantly, it will have a way to reach it by either an alternate or normal shuffle. For example, a card represented by node 25 could be reached by alternate shuffling it from node 12.

Here, the principle of cycles is incredibly important. Recall that by **Lemma: Cycles**, each deck resolvable by a function(normal or alternate shuffling) must have at least 1 cycle resolvable by that function, and that all cards within the deck must be contained within a cycle. Therefore, if a binary tree was constructed for a deck of size n, every node would be part of a cycle.

Because each node on the tree must have at least one way to reach it through alternate or normal shuffling, and because each index on the tree must belong to a cycle, we can say that if two nodes are joined by an operation, (either a normal or alternate shuffle), they must belong to the same cycle of that operation.

Therefore, for a node on rung L, there will be a connection to a node on L-1. The nodes will either be connected by the Normal Shuffle operation, or by the Alternate shuffle operation. If the same operation used to connect both nodes is continuously used, the cycle will eventually resolve, and place the card in the node on the rung L-1. By induction, it must be true that from any rung, you will be able to move higher up on the tree, and therefore you must be able to reach 0.

Therefore, because it is always possible to guarantee a card to the 0th index of the deck, by **Lemma: Binary Tree** it is also always possible for a card at any position within a deck of size n to be controlled to any other position with a number of normal and alternate shuffles.

$$s_m = \phi(modulus)$$

[CONCLUSIONS]

Ultimately, though there are still several questions regarding perfect shuffles, this paper has examined many things about them. We have delved into equations to transcribe card movement, possibilities for solving the minimum shuffles question, and even proved the ability to force a card to any position for any n-sized deck. Though this problem still has many open variations, especially about the minimum shuffles question and the Discrete logarithm Problem, this paper has provided a deeper insight into the inner workings of card shuffling, and the mechanics of sorting algorithms.

Lemmas:

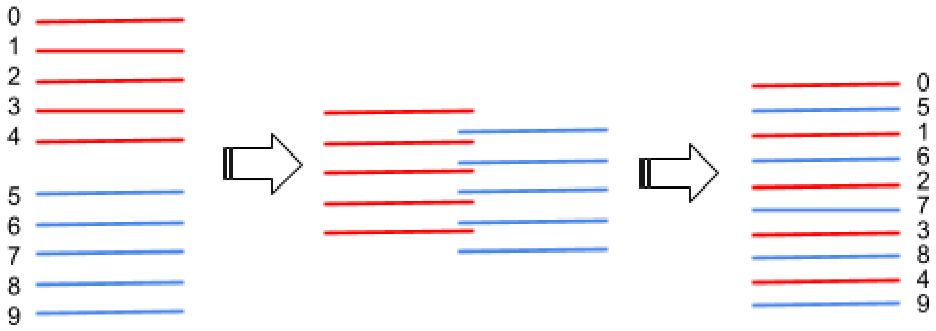
Lemma: 0 and N-1

Take a deck with n cards. If n must be even, then n can be represented as $2h$ where h is one-half the size of the deck. Therefore ½ of the deck will be equal to some value h.

A normal shuffle is a shuffle where the bottom half of the deck is interlaced between each space within the top half of the deck, with the top half taking the first card. Both halves of the deck, as stated earlier must be equal.

This graphic depicts such a shuffle:

Because the top half of the deck will always take the first card, and, because the first card in the deck will always be in the top half, for a normal shuffle on an even deck, the first card will remain in the same position no matter how many shuffles are done.



Likewise, the bottom card in the shuffled deck will always remain at the bottom because the last card in the bottom half of the deck will always have to remain under every other card. No cards can be added after that card and no card can take its position, meaning that it must remain at the last slot no matter how many shuffles are done.

Appendix/References:

Variables:

n - The number of cards in a given deck

s - The number of shuffles performed on a given deck

s_m - The minimum number of shuffles needed to be performed on a given deck to return it to its original state

i - The index of a given card

i_s - The index of a given card after s amount of shuffles

Intro to Modular Arithmetic (from Brilliant)

Euler Totient Function (from Art of Problem Solving)

Euler's Theorem (from Art of Problem Solving)

Order, Complete Reduced Residue Systems, Primitive Roots (from AwesomeMath)

Code (Google Colab)