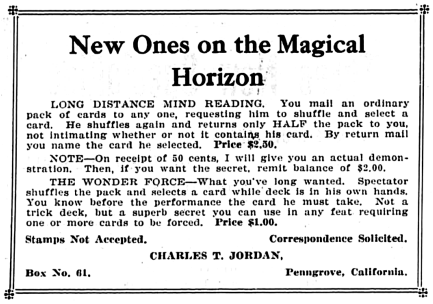
Programming Assignment 3: Shuffling

We learned how to simulate randomness on a computer using the linear congruential method. And we used this method to simulate two of the principal sources of randomness in games of chance—coins and dice. In this programming assignment, you will use this method to simulate a third principal source of randomness in games of chance—**card shuffling.**

The three most common forms of shuffling are:

- **The overhand shuffle.** Hold the deck in one hand, pull out clumps of cards with the other, and push them back in randomly. *Never do this.* It's terrible.
- **The wash shuffle.** Spread out the deck on a table and make washing movements with your hands. This method yields good results, *if you wash for at least 60 seconds.*
- **The riffle shuffle.** Hold half of the deck in each hand, thumbs inward, and flick through so that the cards riffle into each other. *Recommended, if you do it enough.*



New Ones on the Magical Horizon

LONG DISTANCE MIND READING. You mail an ordinary pack of cards to any one, requesting him to shuffle and select a card. He shuffles again and returns only HALF the pack to you, not intimating whether or not it contains his card. By return mail you name the card he selected. **Price $2.50.**

NOTE—On receipt of 50 cents, I will give you an actual demonstration. Then, if you want the secret, remit balance of $2.00.

THE WONDER FORCE—What you've long wanted. Spectator shuffles the pack and selects a card while deck is in his own hands. You know before the performance the card he must take. Not a trick deck, but a superb secret you can use in any feat requiring one or more cards to be forced. **Price $1.00.**

Stamps Not Accepted. **Correspondence Solicited.**

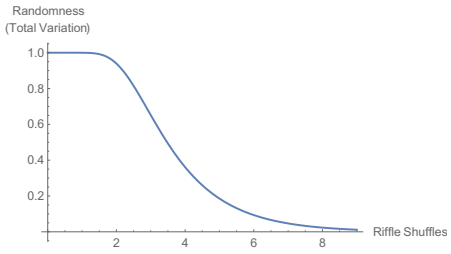
CHARLES T. JORDAN,

Box No. 61. **Penngrove, California.**

How often should you riffle shuffle? More often than you think. In 1916 magician, radio repairman, rebus syndicate operator, and chicken farmer, Charles Thornton Jordan placed an ad in *The Sphinx*, The Official Organ of the Society of American Magicians, for tricks called "Long Distance Mind Reading" and "Psychola". For $2.50 (about $125 today) he would mail you a deck of cards with the instructions: "Give the cards a cut, give them a shuffle, then give them another cut and another shuffle, give them a few more cuts. I'm sure you'll agree that no human can know the name of the top card. Please take this card off, look at it, and remember it. Insert it into the middle of the deck. Give the cards a random cut and another shuffle. Mail the deck back to me. Oh yes, at 6:00 p.m., every evening, please concentrate on your card." Upon receiving the deck, Jordan would mail back the card you chose, and an explanation how he found it.

How is this possible? He sends you the deck in a known order. Imagine that after cutting, the left half of the deck is all one color in ascending order, and the right half is another color and also in ascending order. Riffling them together leaves the cards in two interlocking chains. Although you don't know exactly how the chains are interlocked, if you look at just one of the chains, i.e. the cards of a single color, you know exactly how they are ordered. So the result of a single riffle shuffle is not very random.

When you shuffle the second time, you end up with four interlocking chains. When you shuffle a third time, you end up with eight interlocking chains. To truly shuffle the deck, you need to break all these chains. Now, the average length of the eight chains is one eighth of the deck. More generally, after n shuffles, you have 2^n interlocking chains, of average length $\frac{52}{2^n}$. So a natural guess is that you want to take n big enough to make this average length 1, i.e. $n = \log_2(52) \approx 5.7$. This isn't quite enough, but it's close.



In 1990, Dave Bayer and Persi Diaconis got into *The New York Times* for showing that the right number is about $\frac{3}{2}\log_2(52) \approx 8.6$. More precisely, they computed the graph at the right. In particular, they found that *riffle shuffling 5 times isn't nearly enough*, but that 7 is nearly as good as 8 or 9.

How to Simulate a Riffle Shuffle. To derive their result, Bayer-Diaconis relied on the Gilbert-Shannon-Reeds model of shuffling. Diaconis had experimentally shown that this model is a good description of the way real people shuffle real cards.

(Diaconis is a cardshark. He dropped out of high school to become a professional magician but got so curious about probability theory that he later earned a PhD in mathematics. It is said that he was admitted on the strength of his perfect riffle shuffle. This is when you manage to precisely alternate between cards in your left and right hands. There are sleights of hand for doing this easily, but it is very difficult to do honestly. If you could perform eight perfect riffle shuffles in a row, you would return the deck to its original ordering.)

There are several equivalent ways to describe the Gilbert-Shannon-Reeds model:

(1) Cut the deck such that the probability of grabbing k of the n cards with your left hand is $\binom{n}{k}/2^n$. Then, move one card at a time, repeatedly, from the bottom of one hand to the top of the shuffled deck, such that if l cards remain on your left and r cards remain on your right, then the probability of choosing a card from your left is $l/(l+r)$ and the probability of choosing a card from your right is $r/(l+r)$. In other words, at any moment during the riffle, the probability that the card against your left thumb is the next to fall equals the proportion of the not-yet-fallen cards which are in your left hand.

(2) It can be shown that the above model generates a permutation of the initial deck in which each card is equally likely to have passed through your left or right hand. So work backwards: To generate a random permutation according to this model, begin by flipping a fair coin n times, to determine for each position of the shuffled deck whether it came from the left or right. Then cut into two packets whose sizes are the number of tails and the number of heads flipped, and use the same coin flip sequence to determine from which packet to pull each card of the shuffled deck.

You may use either description to complete the programming assignment.

To complete this assignment, **upload your code** to Gradescope, where it will be automatically tested, and checked for code similarity against your classmates'.

What You Need To Do

Write code to simulate the *riffle shuffle* (described above) and the *top-to-random* shuffle.

Top-to-random is a simplistic shuffling method where you take the top card from the deck and put it in a random position in the deck, each position being equally likely. (The random position can include leaving it at the top of the deck.).

For example, suppose we have a deck containing cards numbered 0 through 9. Imagine that initially the position of the cards is (from top to bottom):

0 1 2 3 4 5 6 7 8 9

Then after one top-to-random shuffle, the order might be:

1 2 3 0 4 5 6 7 8 9

so that 1 is now at the top. At first, the deck will not be very random. But with many repetitions, it will become uniformly random. *Why?* Eventually, some card will be placed after the 9. And after that, eventually, a second card will be placed after the 9. The second will be equally likely to be placed before or after the first. In this way, although the cards before the 9 may not be uniformly random, the two cards after the 9 will be. Slowly, the stack of uniformly random cards after the 9 grows, until the 9 reaches the top of the deck, and the 9 is inserted uniformly randomly, and at that point, the deck becomes uniformly random. (The top-to-random shuffle is a riffle shuffle in which you cut one card from the top or bottom of the stack.)

Analysis and Task. We can represent the process of repeated shuffling with a Markov chain, where the state is the current order of the cards in the deck, and each transition is a single round of our shuffle technique. Throughout, we will assume cards are just numbered, and do not have suits or anything like that.

The Markov chain we obtain this way turns out to be regular, for both top-to-random and GSR shuffling. Thus, in principle we could represent the Markov chain as a matrix and compute large powers of this matrix to understand what happens when we repeatedly shuffle. The main issue is that the number of states in the chain is very large! Since each permutation of the cards is a state in the chain, then for a 10 card deck we would have $10! = 3628800$ states. For 52 cards, the size of the matrix would be astronomical.

In fact, it is possible using more advanced techniques in probability theory to analyze this Markov chain despite its size. If you go on to study more probability theory, you may learn about them. However, for now we will instead use Monte Carlo simulations to analyze some behavior of the shuffling process.

In particular, the question we'll examine is what happens to the relative ordering of pairs of cards in the deck as we shuffle. Say we start with a deck of n cards, numbered 0 through $(n-1)$ from top to bottom, and we shuffle k times. We might then ask, **what is the probability that the card numbered i is above the card numbered j at the end?** If the deck is "well-shuffled" then this probability should be close to $1/2$, otherwise, there is some kind of bias in the positioning of the cards. (Of course, that's not the only test of how well shuffled the deck is, and you might find it interesting to explore other tests.)

Your task is to write Monte Carlo simulation code to find this probability for different values of n, k, i, and j, and for the two types of shuffling procedures discussed in the previous section.

Implement this in Python, and structure your code (for autograding) as follows:

- Write functions `gsr(l)` and `top_to_random(l)` which take a list `l` as input and return the result of applying one round of the shuffle procedures to `l`.
  IMPORTANT: we suggest you avoid modifying the input list `l` in your shuffle code, as this is a common source of errors (see the appendix for more details).
- Write a function `test_order(i, j, l)` which tests whether the card numbered `i` occurs before `j` in the list `l`.
- Write a function `num_trials(err, prob)` which computes (using Chebyshev's inequality) how many trials to run in order to guarantee that the probability that your simulated probability (for either `gsr` or `top_to_random`) is off by more than `err` is less than `prob`. *[You may need to review what we learned in the lecture on the Markov & Chebyshev inequalities to do this.]*

- Write functions `gsr_order_prob(n, k, i, j, err, prob)` and `top_to_random_order_prob(n, k, i, j, err, prob)` which do repeated Monte Carlo trials (the number of trials given by your `test_order` function) to estimate the probabilities, using your `test_order` function to check the output after shuffling.

The top-to-random-shuffle is simpler to implement, so we would suggest you start there and get that working first.

Appendix A. Common Bugs and Errors

A common source of bugs in the past on similar problem sets have been related to the way lists work in Python. The issue is that when we call a function and pass it a list as an argument, changes in the body of the function modify the original list. For example, consider the code below:

```
def test(l):
    l[1] = 6


l = [1,2,3]
print(l)
test(l)
print(l)
```

The output is

```
[1, 2, 3]
[1, 6, 3]
```

rather than printing `[1, 2, 3]` twice. Therefore, a common issue is that students modify the original list representing a deck of cards in their Monte Carlo simulation. Then they inadvertently re-use that modified list in subsequent iterations of their Monte Carlo loop. The effect of this is that it would be as if subsequent trials started with an already pre-shuffled deck.

References.

- C. T. Jordan, Long distance mind reading. *The Sphinx* **15** 57, 1916. http://cs.bc.edu/ mctaguec/long-distance-mind-reading.pdf
- Persi Diaconis, *Magical Mathematics*, Princeton Univ. Press, 2015. https://archive.org/details/magicalmathemati0000diac.
- In Shuffling Cards, 7 Is Winning Number, by Gina Kolata for the *The New York Times*, Jan. 9, 1990. https://www.nytimes.com/1990/01/09/science/in-shuffling-cards-7-is-winning-number.html
- Dave Bayer and Persi Diaconis, Trailing the Dovetail Shuffle to its Lair, Ann. Appl. Probab. 2 (1992), no. 2, 294–313, doi:10.1214/aoap/1177005705