Simplicity Generation of Discrete Random Variables with Coin Flipping

Yuxin Feng yf1079

April 2021

Abstract

In this paper, I review the concepts of *Buffon machine* and von Neumann algorithms in the form introduced in [1], as well as their relations to computational problems. A Buffon machine is a device that produces perfect simulations to discrete random variables with a sequence of unbiased coin flips as a discrete source of randomness. Inspired by the famous Buffon's needle experiment, it provides a human-accessible framework that can be implemented within finite computations and produces experiments with probability of success expressible in terms of, for example, square root and exponential numbers.

Contents

1 Background Information

1.1 Monte Carlo Simulation

A Monte Carlo simulation is a technique that relies on repeated random sampling to give a probabilistic interpretation on problems that might be deterministic in principle. The particular pattern of a Monte Carlo simulation is to generate random inputs over a predetermined domain, assign empirical values after a deterministic computation on the inputs and provide an estimate after averaging the results.

Monte Carlo simulations are named after the popular gambling destination in Monaco.The technique was first developed by Stanislaw Ulam, a mathematician who worked on the Manhattan Project that produced the first nuclear weapons during World War II. After the war, while recovering from brain surgery, Ulam entertained himself by playing countless games of solitaire. He became interested in plotting the outcome of each of these games in order to observe their distribution

and determine the probability of winning. After he shared his idea with John Von Neumann, the two collaborated to develop the Monte Carlo simulation[2]. An early variant of the Monte Carlo method was designed to solve the Buffon's needle problem.

1.2 Buffon's Needle experiment

Buffon's Needle is a famous probability problem first posed by Georges-Louis Leclerc, Comte de Buffon in the 18th century in his publications *Histoire naturelle, générale et particuliére*. Imagine a floor marked with an infinite number of parallel, equidistant lines, a width l apart. We now toss a needle of length $L < l$ onto the floor, where it lands at a uniform random position and with a uniform random orientation[3]. Buffon's needle problem is to determine the probability that the needle intersects one of the lines on the floor, and therefore, π can be estimated after conducting a Monte Carlo Simulation. The following is a mathematical deduction on the problem.

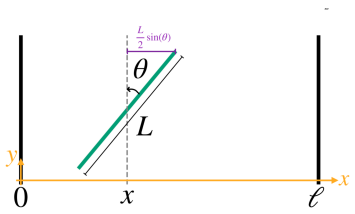

Figure 1: Representing a needle

Practically, instead of considering an infinite floor, we consider two successive lines. First, build a Cartesian coordinate to represent position and orientation of the needle. The center of the needle is at (x, y)*, and its acute angle with the vertical axis* y *is* θ*. Note that since the* y *coordinate does not impact whether the*

needle intersects a line or not, only the x coordinate matters.

The state space R of a needle is $R = \{(x, \theta)|0 \leq x \leq l, 0 \leq \theta \leq \frac{\pi}{2}\}$. The difference between the x-coordinates of the two endpoints and the center is $\frac{L}{2}\sin\theta$. Therefore a needle which is randomly thrown on the floor crosses a line iff

$$x + \frac{L}{2}\sin\theta \geq l \textbf{ or } x - \frac{L}{2}\sin\theta \leq 0$$

The uniform PDF of x is $\begin{cases} \frac{1}{l}, 0 \leq x \leq l \\ 0, otherwise \end{cases}$ *The uniform PDF of θ is* $\begin{cases} \frac{1}{\pi} \leq \theta \leq \pi \\ 0, otherwise \end{cases}$

Since x and θ are independent, the joint PDF is $\begin{cases} \frac{1}{l\pi}, 0 \leq x \leq l, 0 \leq \theta \leq \pi \\ 0, otherwise \end{cases}$

Therefore, the probability P that a needle crosses a line is

$$P = \int_0^{\pi}\int_0^{\frac{L}{2}\sin\theta}\frac{1}{l\pi}dxd\theta + \int_0^{\pi}\int_{l-\frac{L}{2}\sin\theta}^{l}\frac{1}{l\pi}dxd\theta = \frac{2L}{\pi l}$$

The probability P can be estimated with a Monte Carlo simulation. Since l and L are known, by the law of large numbers, one can get the inverse of π by tossing a large number of needles on the floor and then counting the fraction of needles that crosses a line.

1.3 Turing Machine

Turing Machine [4] was introduced by Alan Turing in 1936 to perform calculations which were used to be done on sheets of paper by computers. A Turing machine is an abstract mathematical model of computation that is capable of

simulating any given computer algorithm. It imitates two behaviors of a human 'computer':(i)write or erase a symbol, (ii) shift attention from one place to another.

To implement this idea, a Turing machine consists of (i) an infinite memory *tape* with discrete "cells", and each cell contains a symbol from some finite alphabet, (ii) a *head* which can 'reads' the symbol in each cell and move the tape left or right one cell at a time, (iii) a finite *table* of elementary instructions, such as "in state 1, if the symbol seen is 0, write a 1; if the symbol seen is 1, change to state 2" etc., (iv) a *state register* that stores the current state of the machine.

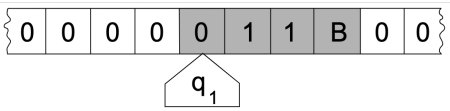

Figure 2: A simple Turing machine with an infinite tape pre-filled with 0. The current internal state(q_1) is shown inside the head.

Given the current machine state (q_i) and the symbol (a_j) in the cell currently under the head, the machine follows the instructions told by the finite table and does the following: (i) either erase or write a symbol(replacing a_j with a_{j1}), (ii) move the head(d_k), with value L for one step left, R for one step right or N for staying in the same place, (iii) assume the same state or go to a new state(q_{i1})

Tape symbol	Current state A			Current state B			Current state C		
	Write symbol	Move tape	Next state	Write symbol	Move tape	Next state	Write symbol	Move tape	Next state
0	1	R	B	1	L	A	1	L	B
1	1	L	C	1	R	B	1	R	HALT

Figure 3: A state table for 3-state, 2-symbol Turing machine

Remark 1. *A system of data-manipulation rules (such as a computer's instruction set) is said to be Turing-complete or computationally universal if it can be used to simulate any Turing machine.*

2 Buffon Machine

The idea of *Buffon machine* was first introduced by Flajolet, Pelletier and Soria in their work *On Buffon Machines and Numbers* [1]. The original motivation is to provide an experimental framework for discrete random variable generation required in the design and implementation of various combinatorial samplers, such as Boltzmann samplers [5].

The Buffon's needle experiment can be regarded as a simple continuous device with real uniform [0,1] input(given by the coordinate of the center and angle of the needle) and a discrete uniform [0,1] output. Generalizing the experiment, the article gives a framework of probabilistic algorithms allowing to simulate many probability distributions using only a source of random bits, i.e. a sequence of i.i.d. unbiased coin flips, within finite computations. A formal definition of Buffon machine is as below.

Definition 2.1. *[1] A Buffon machine is a deterministic device belonging to a computationally universal class, equipped with an external source of independent uniform random bits and an input-output tapes to store integers(*$0, 1$*-bits), which is assumed to halt with probability 1.*

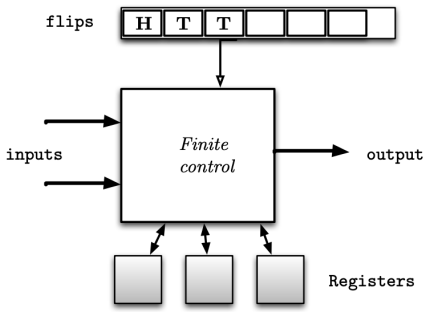

Figure 4: A Buffon machine with two inputs, one output and three states

The simplest Buffon machine M is input-free, and produces a Bernoulli random variable output X with value in $\{0,1\}$. Then M is said to be the *Buffon machine* for number $p := \mathbb{P}(X = 1)$. The Buffon machines also allows the composition of simple probabilistic experiments. That is to say, based solely on iterations of unbiased coin flips, one can get Bernoulli variables with probabilities of success with values consisting of square roots, exponentials or logarithms etc. Inspired by the Buffon's needle problem, theses values can then be estimated through Monte Carlo simulations.

Denote $\Gamma B(\lambda)$ as a Bernoulli generator of parameter λ. A Buffon machine can realize a function ϕ by calling $\Gamma B(\lambda)$ as an input(possibly several times), and then outputting a Bernoulli variable of parameter $\phi(\lambda)$, i.e. the construction $\Gamma B(\lambda) \xrightarrow{\phi} \Gamma B(\phi(\lambda))$. The realizability of ϕ is defined below.

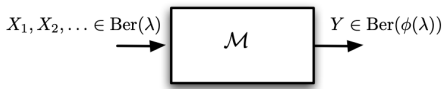

Figure 5: The realizability of a Buffon machine

Definition 2.2. *[1] The function $\lambda \mapsto \phi(\lambda)$, $\lambda \in (0,1)$ and $\phi(\lambda) \in (0,1)$ is* ***weakly realizable*** *if there is a machine M with input a Bernoulli variable of unknown parameter λ perfectly generates a Bernoulli variable of parameter $\phi(\lambda)$ with probability 1. The function ϕ is* ***strongly realizable*** *if the total coin flips C(a random number) needed has finite expectation.*

The question has now become: given a coin with an unknown parameter λ of getting a "head" and a function $\phi : (0,1) \to (0,1)$, for what kinds of functions ϕ is it possible to realize a $\phi(\lambda)$-coin?

The problem was first raised by S.Asmussen and J.Propp. In 1994, Keane and O'Brien [6] determined the functions ϕ in their work *A Bernoulli Factory* when there are no computational restrictions on the simulation scheme. Later, Mossel and Peres [7, 8] proved in their work *New coins from old: computing with unknown bias* that in general, the $\phi(\lambda)$-coin can be simulated within finite flips of the λ-coin iff ϕ is a rational function over rational numbers. Indeed, the following two theorems give two important characterizations of ϕ.

Theorem 2.1. *[1] (i) Any polynomial $\phi(\lambda) : (0,1) \to (0,1)$ with rational coefficients is strongly realizable by a finite graph. (ii) Any rational functions $\phi(\lambda) : (0,1) \to (0,1)$ with rational coefficients is strongly realizable by a finite graph.*

Flajolet, Pelletier and Soria propose an approach to produce a Bernoulli generator of any rational parameter λ by combing a simpler dyadic generator with iterations. For example, to get a $\Gamma B(\frac{1}{3})$, one can flip the unbiased coin twice: if observing 11 $\to$ "success"; if observing 01 or 10 $\to$ "failure"; if observing 00 $\to$ repeat the experiment. This is equivalent to a finite control graph(or Markov chain), and thus any rational number is realizable.

Moreover, one can also construct a geometric generator, $\Gamma G(\lambda)$, from a Bernoulli of the same parameter by repeatedly flipping the coin until getting a failure.

$$\Gamma G(\lambda) := \{K := 0; \textbf{do } \{\textbf{if } \Gamma B(\lambda) = 0 \textbf{ then return } K; \textbf{else return } K := K + 1\}\}$$

3 Von Neumann schema

The fundamental question is how the uniform random bits generate exactly and efficiently discrete distributions, including geometric, Poisson and logarith-

mic, within finite computations. Such random bit model was first suggested by Von Neumann [9]. In 1976, Knuth and Yao [10] provided a rigorous theoretical framework and generic optimal algorithms able to stimulate any distribution. However, the algorithms were generally not practically usable as they proposed an infinite discrete distribution generating (DDG) tree that requires an infinite precision arithmetic to calculate the binary expansion of the probabilities. In Flajolet, Pelletier and Soria's work, they introduced a general schema, *the von Neumann schema*, denoted as ΓVN. The general properties of the von Nuemann schema are as follow.

Theorem 3.1. *[1] (i) Given an arbitrary class P of permutations and a parameter $\lambda \in (0,1)$, the von Neumann schema $\Gamma VN[P](\lambda)$ produces exactly a discrete random variable with probability distribution*

$$\mathbb{P}(N = n) = \frac{1}{P(\lambda)} \frac{P_n \lambda^n}{n!}$$

(ii) The number K of iterations has expectation $\frac{1}{s}$ where $s = (1-\lambda)P(\lambda)$, and its distribution is $1 + Geo(1-s)$

(iii) The number C of flips consumed by the algorithm is a random variable with probability generating function

$$\mathbb{E}(q^C) = \frac{H^+(\lambda, q)}{1 - H^-(\lambda, q)}$$

where

$$H^+(\lambda, q) = (1-z)\sum_{n=0}^{\infty} \frac{P_n}{n!} h_n \langle q \rangle z^n, H^-(\lambda, q) = (1-z)\sum_{n=0}^{\infty} (1 - \frac{P_n}{n!}) h_n \langle q \rangle z^n$$

. and the coefficients $h_n(q)$ statistics the recurrence equation

$$h_n(q) = \frac{1}{1 - q^n 2^{1-n}} \sum_{k=1}^{n-1} \frac{1}{2^n} \binom{n}{k} h_k(q) h_{n-k}(q), \text{ for } n \geq 2$$

The distribution of C has exponential tails.

Remark 2. *Denote P as a class of permutations, P_n as the subset of P of size n and p_n as the cardinally of P_n.*

(i) The exponential generation function(EGF) of p_n is $P(z) := \sum_{n \geq 0} p_n \frac{z^n}{n!}$. Generally, EGFs are more convenient than ordinary generating functions for solving combinatorial enumeration problems that involve labelled objects.

*(ii) Denote Q as **all** permutations ($q_n = n!$), R as **sorted** permutations ($r_n = 1$) and S as **cyclic** permutations ($s_n = (n-1)!$). Therefore, $Q(z) = \frac{1}{1-z}$, $R(z) = e^z$, $S(z) = log\frac{1}{1-z}$*

Remark 3. *Let $U = (U_1, ..., U_n)$ be a vector of real numbers. One can get a permutation $\sigma = (\sigma_1, ..., \sigma_n)$ by replacing each U_j by its rank in U. Denote σ as **type**(U).*

Proof. (proof for part (i) and (ii)) We first construct a von Neumann schema relative to a class P of permutations, denoted as $\Gamma VN[P](\lambda)$ as below. Here,

$\Gamma G(\lambda)$ is a geometric generator with unknown parameter λ.

$$\begin{aligned}
\Gamma VN[P](\lambda) := \{&\mathbf{do}\{\\
&N := \Gamma G(\lambda);\\
&\mathbf{let}\ \mathbf{U} := (U_1, ..., U_N), \text{a vector of } [0,1]\text{-uniform}\\
&\mathbf{let}\ \tau := \text{trie}(\mathbf{U});\\
&\mathbf{let}\ \sigma := \text{type}(\mathbf{U});\\
&\mathbf{if}\ \sigma \in P_N\ \mathbf{then}\ \text{return}\ N\ \mathbf{else}\ \text{repeat}\}\}.
\end{aligned}$$

Here, *trie*(U) is the process of constructing a *digital tree* which is used to gradually sort U_j by generating the minimal numbers of bits needed to distinguish one from another. Thus, the order type σ can be determined afterwards.

At each stage of the iteration, a value N is generated by $\Gamma G(\lambda)$ with probability $(1-\lambda)\lambda^N$. Next, check if type(U) is in P_N. For example, in a sorted permutation with increasing order, R_3 is $1, 2, 3$. Then getting a type$(U) = 2, 1, 3$ will not pass the trial, and thus enter the next iteration.

Therefore, a series of iterations that eventually returns $N = n$ is of probability

$$\begin{aligned}
\mathbb{P}(N = n|\ \text{exit the iteration}) &= \frac{\mathbb{P}(N = n \cap (\text{exit the iteration}))}{\mathbb{P}(\text{exit the iteration with any } n)}\\
&= \frac{\frac{P_n}{n!}((1-\lambda)\lambda^n)}{\sum_{n=0}^{\infty}\frac{P_n}{n!}((1-\lambda)\lambda^n)}\\
&= \frac{1}{P(\lambda)}\frac{P_n\lambda^n}{n!}
\end{aligned}$$

This completes the proof for part (i). For part (ii), it immediately follows that

the probability of unsuccessfully exit the iteration each time is

$$1 - \mathbb{P}(\text{exit the iteration}) = 1 - P(\lambda)(1-\lambda)$$

Since each trial is independent from each other, the time of iterations, K, follows $\text{Geo}(1 - P(\lambda)(1-\lambda)) + 1$ (proof for part (iii)) As measured by the number of coin flips, the cost of each iteration is exactly that of generating a digital tree τ of random size N. The number of coin flips to build τ coincides with the path length of τ, denoted as $\omega(\tau)$, which satisfies the induction equation

$$\omega(\tau) = |\tau| + \omega(\tau_0) + \omega(\tau_1), |\tau| \geq 2$$

where $\tau = \langle \tau_0, \tau_1 \rangle$, and $|\tau|$ is the size of τ.

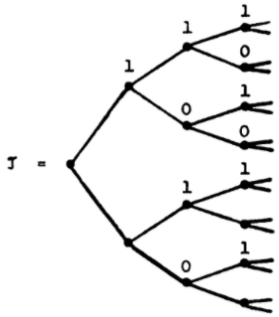

Figure 6: An example of a digital tree τ of a binary expansion

According to the work of Knuth in the book *The Art of Computer Programming* [11], the expectation of the path length of n uniform binary sequences is finite,

$$\mathbb{E}_n[\omega] = n \sum_{k=0}^{\infty} \left[1 - \left(1 - \frac{1}{2^k} \right)^{n-1} \right]$$

This implies that $\mathbb{E}(C)$ is finite, and the generator $\Gamma VN[P](\lambda)$ is realizable.

According to the work of Mahmoud *et al.* [12], the path length has asymptotically Gaussian distribution as $n \to \infty$. The bivariate EGF(generating functions with two variables) that satisfies the nonlinear functional equation $H(z,q) = H(\frac{zq}{w},q)^2 + z(1-q)$ with $H(0,q) = 1$ is

$$H(z,q) := \mathbb{E}_n[q^\omega]\frac{z^n}{n!}$$

which ensures an exponential tail for C, and thus the strong simulation property.

■

For P one of the three classes Q, R, S describe above gives three distributions. Therefore, one can produce the Geometric, Poisson and logarithmic distributions by means of permutations obeying simple restrictions.

Theorem 3.2. *The Poisson and logarithmic distributions of parameter $\lambda \in (0,1)$ have a strong simulation by a Buffon machine, $\Gamma VN[R](\lambda)$ and $\Gamma VN[S](\lambda)$, respectively by only using a single string register.*

all ($\mathcal{Q}$)	sorted ($\mathcal{R}$)	cyclic ($\mathcal{S}$)
$(1-\lambda)\lambda^n$	$e^{-\lambda}\frac{\lambda^n}{n!}$	$\frac{1}{L}\frac{\lambda^n}{n}$
geometric	Poisson	logarithmic.

Figure 7: The three distributions produced by classes Q, R, S respectively

4 Using a Buffon machine to compute square roots

Here is an example of constructing a Buffon machine to compute square-root numbers. Consider tossing a fair coin $2n$ times, the probability of getting as many heads as tails is $\varpi_n = \frac{1}{2^{2n}}\binom{2n}{n}$. One can build a square-root computer using coin flips as follow.

$$\begin{aligned}\Gamma B(\sqrt{1-\lambda}) :=& \{\textbf{let } N := \Gamma G(\lambda);\\ & \textbf{draw } X_1, ...X_{2N} \text{ with } \mathbb{P}(X_j = 1) = \mathbb{P}(X_j = -1) = \frac{1}{2};\\ & \textbf{set } \Delta := \sum_{j=0}^{2N} X_j;\\ & \textbf{if } \Delta = 0 \textbf{ then return } 1 \textbf{ else return } 0\}.\end{aligned}$$

Theorem 4.1. *[1] The square-root construction yields a Bernoulli generator of parameter $\sqrt{1-\lambda}$, given a $\Gamma B(\lambda)$. The mean number of coin flips required(not counting the ones involved in the calls to $\Gamma B(\lambda)$), is $\frac{2\lambda}{1-\lambda}$. The function $\sqrt{1-\lambda}$ is strongly realizable.*

Proof. Let N be a random variable with Geometric distribution of parameter λ, Then, the probability of success, i.e. getting as many heads as tails with any $N = n$ flips is

$$S(\lambda) := \sum_{n=0}^{\infty} \mathbb{P}(\text{generate a } N = n)\, \mathbb{P}(\text{get a 'success' with } N = n) = \sum_{n=0}^{\infty} (1-\lambda)\lambda^n \varpi_n$$

Since the binomial expansion of $\frac{1}{\sqrt{1-\lambda}} = \sum_{n=0}^{\infty} \frac{\prod_{i=1}^{n}(2i-1)}{2^n n!}\lambda^n$,

and $\binom{2n}{n} = 2^n \frac{1\times3\times5\ldots(2n-1)}{n!} = 2^n \frac{\prod_{i=1}^{n}(2i-1)}{n!}$, $S(\lambda) = (1-\lambda)\frac{1}{\sqrt{1-\lambda}} = \sqrt{1-\lambda}$.

Finally, the mean number of coin flips used is simply obtained by the differentiation of generating functions. Therefore, $\sqrt{1-\lambda}$ is strongly realizable. ■

5 Conclusion

This paper has introduced a few simple Buffon machines that generate geometric, Possion and logarithmic distributions using a finite number of coin flips. With some variations in the schema of the previous constructions, one can also get parameters which are exponential, logarithmic and trigonometric numbers with strong simulations, such as $e^{-\lambda}$, $\lambda e^{1-\lambda}$, $\frac{\lambda}{\log(1-\lambda)^{-1}}$, $\cos\lambda$, $\frac{\lambda}{\tan x}$ etc. A summary of the relevant works can be found in Keane and O'Brien's *A Bernoulli factory* [6].

While this paper only considers the generation of discrete random variables, there are many studies focused on the generation of continuous random variables X, specified by a distribution function $F(x) = \mathbb{P}(X \leq x)$. For example, Knuth and Yao [10] has proposed a framework that includes the generations of various types of nonuniform random variables with unrestricted devices, i.e. sequences of approximation functions of increasing complexity.

The full potentiality of Buffon machines is still being developed. The current area of interests is in the implementations to achieve the optimal average coin flipping consumption in different scenarios (see [13]).

References

[1] Flajolet, Philippe; Pelletier, Maryse; Soria, Michèle. On Buffon Machines and Numbers. *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 2011, pages 172–183. SIAM, 2011.

[2] Kenton, Will.(2020 December 27). *Monte Carlo Simulation*. Received from https://www.investopedia.com/terms/m/montecarlosimulation.asp.

[3] Buffon, Comte De. *Histoire naturelle, générale et particuliére. Servant de suite à l'Histoire Naturelle de l'Homme*. Imprimerie Royale, Paris, 1749– 1789, pp. 95-105.

[4] Turing, A.M. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society 2*, 1937, 42: 230-265.

[5] Duchon, Philippe; Flajolet, Philippe; Louchard, Guy; Schaeffer, Gilles. Boltzmann Samplers for the Random Generation of Combinatorial Structures. *Combinatorics, Probability and Computing*, July 2004, 13 (4–5): 577–625.

[6] Keane, M. S.; O'Brien, G. L. A Bernoulli factory. *ACM Transactions on Modeling and Computer Simulation* (TOMACS) 4, 2 (1994), 213–219.

[7] Mossel, E.; Peres, Y. New coins from old: Computing with unknown bias. *Combinatorica* 25, 6 (2005), 707–724.

[8] Nacu, S.; Peres, Y. Fast simulation of new coins from old. *The Annals of Applied Probability 15*, 1A (2005), 93–115.

[9] Von Neumann, J. Various techniques used in connection with random digit. *National Bureau of Standards Applied Math.* Series 12(1951), pp. 36-38

[10] Knuth, D. E.; Yao, A. C. The complexity of nonuniform random number generation. *Algorithms and complexity (Proc. Sympos., Carnegie- Mellon Univ., Pittsburgh, Pa., 1976)*, Academic Press, New York, 1976, pp. 357–428.

[11] Knuth, D. E. *The Art of Computer Programming*, 3rd ed., vol. 2: Seminumerical Algorithms. Addison- Wesley, 1998.

[12] Mahmoud, H. M.; Flajolet, P.; Jacquet, P.; Régnier, M. Analytic variations on bucket selection and sorting. *Acta Informatica*, 36, 9-10 (2000), 735– 760.

[13] Lumbroso, J. Optimal Discrete Uniform Generation from Coin Flips, and Applications. arXiv:1304.1916.

Faculty Mentor Statement of Completion for Senior Thesis in Mathematics

Senior Thesis in Mathematics Committee:

As the faculty mentor of the senior student **Yuxin Feng**, who is completing her senior thesis for partial fulfillment of the major in Mathematics, I hereby certify that the manuscript "**Simplicity Generation of Discrete Random Variables with Coin Flipping**" of the student has met all the requirements and is now acceptable as a senior thesis. It shall now be submitted to the Senior Thesis Program Committee for evaluation.

Name of the student: Yuxin Feng

Name of the Faculty Mentor: Alejandro Ramírez

Date: May 7, 2021

Signature of the Faculty Mentor: