**The Different Methods of Random Number Generation**

Alexander Bodycoat

4/6/18

**Introduction:** Random number generation is an integral piece of technology that forms the backbone of many modern electronic systems. As the name implies, random number generation is the process in which supposedly random digits are produced one after the other, forming a series of numbers in which one number has no distinct relation to any numbers that have come before or will come after it. Random number generation is split into two main groupings, pseudorandom, and true random. Pseudorandom as the name implies are not truly random generations and are series of digits produced by a mathematical algorithm, and a provided seed number. If the same seed is used twice, the same numbers will always be generated. Furthermore, some random number generators will need to be re-seeded to provide an additional source of randomness. [4] Ways of combining pseudorandom generators also exist, using pseudorandom generation to influence more pseudorandom generation. [2] True random, as the name implies are digits generated by methods we cannot reasonably predict by any means known. Examples of this is generating random digits through observing phenomena in real life, such as wind patterns or the shapes that a lava lamp makes as it bobs up and down. The three parts of a true random number generator normal comprise of a source of noise or entropy, based upon an unpredictable phenomena, an extractor for said randomness, a testing suite for proving the results are truly random.[6] The significance of random number generation is that a series of random numbers can also be considered a random number of conditions, which in the field of simulation and modeling is integral to almost any tests given to said simulation or model. Other examples of using random number generation include network protocols design, algorithmic research, various unique identifiers, and security protocols.[5] This study is designed to analyze and study the various types of random number generation, report on the statistics that each method has, namely the measure of its randomness, as well as analyzing the various different methods that true random numbers can be generated. Background will be provided on how random numbers are integral to modern computing, as well as various terms. In this paper the results that will be presented will include common methods of random number generation, examples of true random number generation through observable phenomena, and various statistics related to the measure of randomness each method produces.

**Methods and Materials:** Many random number generation methods already exist, almost every modern programming language has a simple random generation function baked in, these functions are the first step in determining the differing methods. Research will be done into determining the algorithms and functions used to drive these random functions and determine whether they are similar to each other. Two groups will be analyzed, pseudorandom and true random because of the differences in the way they are generated. The main analysis of the random number generation will be done through a program known as RaBiGeTe. RaBiGeTe performs a multitude of mathematical tests on a series of random numbers that are provided to it, and generates a report based on the level of randomness that it can determine from the series that is given. This report will be used to give a much better understanding into how two methods of random number generation can differ from one another, furthermore it will provide numerical data in the form of graphs that provide much easier to process information as well as comparisons of the data provided over the entire testing procedures.

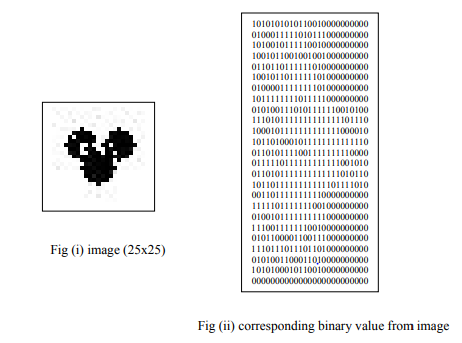
**Results:** The results of this research are an effort to determine what methods of random number generation are the most effective, as well as analyze and show said methods. The first table shown is a diagram of various types of random number generation that have been researched and chosen. These three methods are the main number generators I will be focusing on, due to the different qualities of each, the middle square method is known to be somewhat shoddy and will begin to repeat generated numbers after a relatively short time, the Mersenne Twister is the exact opposite, and is one of the most complex random number generators developed. The multiply with carry method lies in the middle, being neither excessively complex nor not complex enough. These three methods show the full range of random generation capabilities as well as a view on how one can be more effective than another.

*Examples of Pseudorandom Number Generation*

|  |  |  |
| --- | --- | --- |
| **Method Name:** | **Date of Creation:** | **Description:** |
| Middle Square MethodF:\Downloads\250px-Middle-square_method.svg.png | 1949 – By John von Neumann | The Middle square method works by taking a 4-digit number, squaring it to produce an 8-digit number. The middle four digits are then used to create the next 8-digit number, and so on. |
| Mersenne Twister | 1997 – By Makoto Matsumoto and Takuji Nishimura | The Mersenne Twister is based on some of the same principles of the middle square method, namely that a number or ”state” is changed through another mathematical equation, however the Mersenne Twister is extremely complicated and nearly impossible to explain without significant mathematical background. |
| Multiply-With-Carry | 1991 – By George Marsaglia | This random number generator relies on a multitude of random seeds in order to generate random numbers, it is a very effective algorithm in that it only involves simple computer arithmetic. |

***Figure 1***

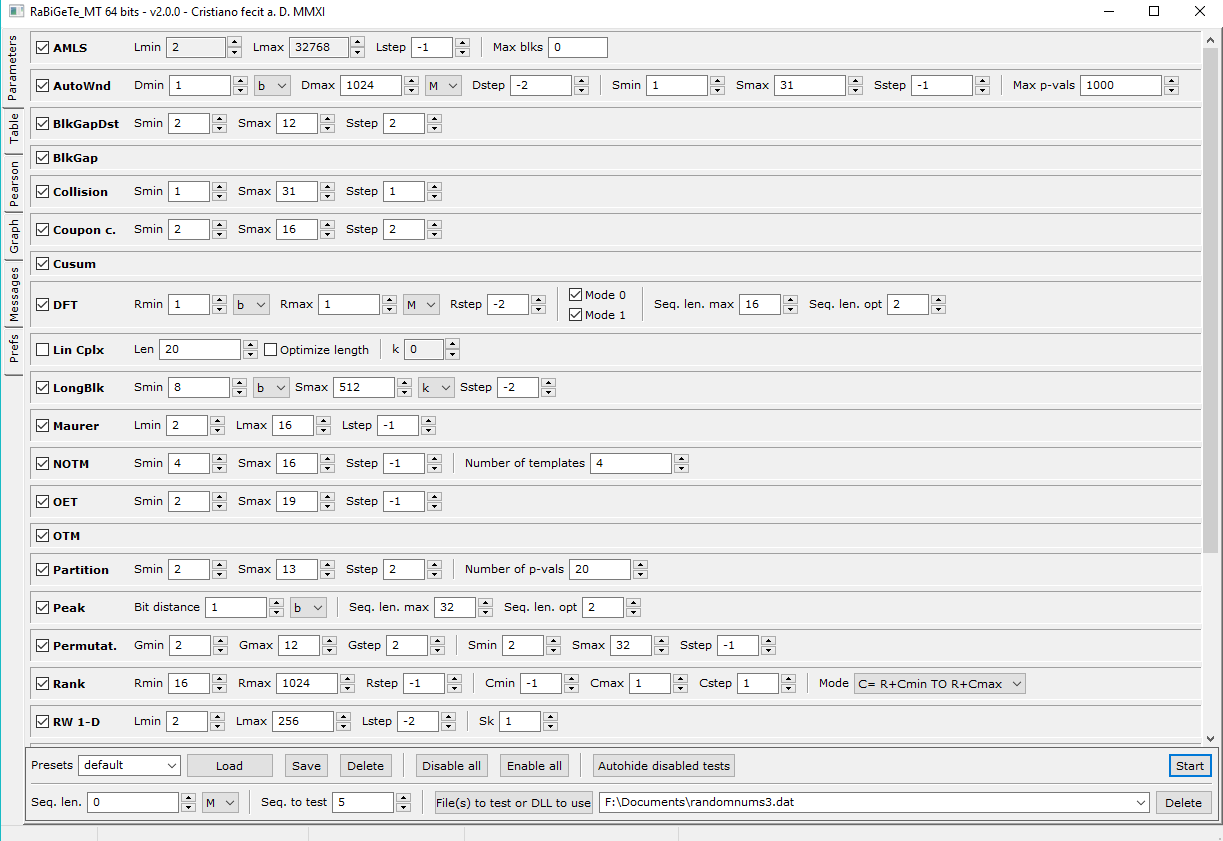
Furthermore, we must also recognize true random number generation as a counterpart to pseudorandom generation. A real-life example of this type of true random number generation is generation via translating an image into binary strings, an example of which is below.

*An Example of a Picture Being Translated to Binary.*

***Fig. 2***

This random number generation works by taking the average RGB value of a given area, namely every pixel in most cases concatenating the numbers as necessary to give proper results. This is the principle technology behind what is known as lavarand, a random number generator developed by the company CloudFlare, which uses pictures taken of over 100 lava lamps in order to produce random numbers to be used for encryption.[1] However, DIEhard is primarily a Linux based program, and as such an alternative program to determine randomness from a series of integers or other digits. The program I have chosen to provide this measurement of randomness is known as RaBiGeTe. Because it is an application and not a command prompt-based system, RaBiGeTe is much more user friendly and can be customized to a large degree when taking into account the number of tests is performs. More so, each test can be customized individually which allows for an even greater degree of customizability. The following is an example of the RaBiGeTe interface and the various test that it can perform.

*Example Screen of RaBiGeTe*

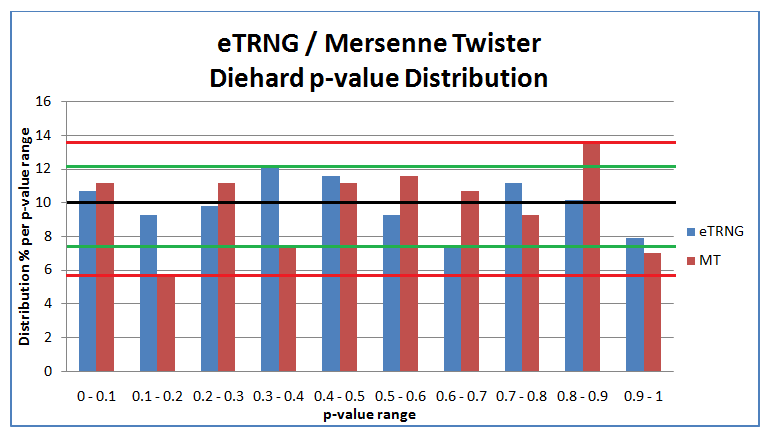


***Fig 3***

For my testing I provided the program with a list of 5000 integers that I created using the website random.org, which is a true random number generator-based service.

**Discussion:** The three random number generation methods chosen are examples of three different efficiencies of random number generation, the middle square method is an example of an algorithm with very low efficiency, in that its generation period, which is the number of digits that can be produced by a random generation algorithm before its begins to loop and repeat, is very low, with a best-case scenario of only 750000 digits. This may seem like a large number but it is in fact a very low amount for other algorithms. The multiply with carry method is an example of an algorithm with mid-range efficiency and complexity with an extremely long period ranging from 2^60 to 2^20000000 digits before repetition. The Mersenne Twister is an example of an algorithm that is very high in complexity and period length. With the period of the generation always being what is known as a Mersenne Prime. The period for this generation stands at 2^19937 – 1. These 3 methods show a wide diversity in the field of random generation. To determine randomness with RaBiTeGe each provided value that is being tested is assigned a p-value from 0 to 1, in the best-case scenario, each p-value distribution should have a roughly 10% distribution overall. The provided graph is an example of an electronic true random number generator vs the p value results of a Mersenne Twister random number algorithm, the black line represents the 10% p-value that is considered the best-case scenario. The Red and Green lines are the upper and lower bounds of the values provided by the random numbers being tested.

*Example P-Value Distrobution*



***Fig 4***

The significance of these results is that we can see with mathematical certainty which pseudorandom generation methods can be considered better than another. These results can be compared against random generation from random.org, which is a true random number generator based on generating binary digits from atmospheric white noise. This shows that compared to random.org pseudorandom number generators are less random than their true random number generator counterparts. This is because by design pseudorandom generators True random generators are of very high interest in today’s technological world, and many different methods have been developed to produce them. Images are not the only real-life item that randomness can be derived from, examples of random number generation can arise from many different types of phenomena. [3] There are some limitations to these results however, because of the limited number of digits that is able to be provided to the Diehard tests within a reasonable amount of time the test results can only be considered an exploratory view into the measure of randomness these methods provide. Some examples of tests that could be undertaken would include taking and analyzing sources from other pseudo and true random number generators in order to form a more complete picture of the scope of randomness provided by both types of generators. Another way to analyze random number generation is through other methods of testing. The NIST statistical Test suit is another source of statistic information related to the randomness of an algorithm and can be used in a method similar to the program Diehard and RaBiGeTe. [8] Furthermore we can examine random number generation through neither computer nor random event, rather we can analyze random number generation through human creation. [7] Researching into better methods to generate numbers from real-life events could provide a way to create an even better algorithm to create random digits.

Github Link - https://github.com/alexlee215/randomNumGen

**Biography**

[1] Charlie Osborne, How Cloudflare uses lava lamps to encrypt the Internet, ZDNe, ZDNet, <http://www.zdnet.com/article/how-lava-lamps-are-used-to-encrypt-the-internet/>

[2]P. L'Ecuyer , Efficient and portable combined random number generators, Published in:Communications of the ACM CACM Homepage archive, Volume 31 Issue 6, June 1988, Pages 742-751, <https://dl.acm.org/citation.cfm?id=62969>

[3] Frequenz, Volume 39, Issue 7-8, Pages 187–190, ISSN (Online) 2191-6349, ISSN (Print) 0016-1136, DOI: https://doi.org/10.1515/FREQ.1985.39.7-8.187.

[4] Practical Random Number Generation in Software, John Viega, Virginia Tech, [viega@list.org](mailto:viega@list.org), https://www.acsac.org/2003/papers/79.pdf

[5] Pseudo Random Number Generators in Programming Languages, Aviv Sinai, The Interdisciplinary Center, Herzlia Efi Arazi School of Computer Science, <http://portal.idc.ac.il/en/schools/cs/research/documents/sinai_2011.pdf>

[6] Andrei Marghescu, “Politehnica” University of Bucharest, CETTI, Romania, Paul Svasta “Politehnica” University of Bucharest, CETTI, Romania, Design and Technology in Electronic Packaging (SIITME), 2015 IEEE 21st International Symposium, <https://ieeexplore.ieee.org/document/7342346/>

[7] Schulz M-A, Schmalbach B, Brugger P, Witt K (2012) Analysing Humanly Generated Random Number Sequences: A Pattern-Based Approach. PLoS ONE 7(7): e41531. <https://doi.org/10.1371/journal.pone.0041531>

[8] Statistical Testing of Random Number Generators, Juan Soto, National Institute of Standards & Technology, <https://csrc.nist.gov/csrc/media/publications/conference-paper/1999/10/21/proceedings-of-the-22nd-nissc-1999/documents/papers/p24.pdf>

[9] <http://cristianopi.altervista.org/RaBiGeTe_MT/#updates>