A Project Presentation on

PSEUDO-RANDOM NUMBER GENERATORS

Under the supervision of

Dr. Sukanta Das

By
Avijit Borah, Saugata Debnath , Akash Rao

What is PRNG?

Pseudo Random Number Generator(PRNG) refers to an **algorithm** that uses mathematical formulae to produce sequences of random numbers.PRNGs generate sequence of numbers approximating the properties of random numbers.

A PRNG starts from an arbitrary starting state using a **seed** state. Many numbers are generated in a short time and can also be reproduced later, if the starting point in thesequence is known. Hence, the numbers are **deterministic** and **efficient.**

Why Do We Need PRNG?

- surprisingly as it may seem,it is difficult to get a computer to do something by chance as computer follows the given instructions blindly and is therefore **completely predictable**. It is **not possible to generate truly random numbers** from deterministic thing like computers so PRNG is a technique developed to generate random numbers using a computer.

Characteristics of PRNG:

- **Efficient:**

  Produce many numbers in a short time.

- **Deterministic:**

  A given sequence of numbers can be reproduced at a later date if the starting point in the sequence is known.

- **Periodic**:

  PRNGs are periodic, which means that the sequence will eventually repeat itself. While periodicity is **hardly ever a desirable characteristic.**

- modernPRNGs have a period that is so long that it can be ignored for most practical.

Applications of PRNG

- simulation and modeling applications.
- selecting random samples from larger data sets.
- generating data of encryption keys.
- popular for making games .

Test Suites for testing randomness of numbers generated through PRNGs

- **Diehard Tests**
- **NIST Statistical Tests**

Diehard tests

- Developed by->**Dr.George Marsaglia**(1995)
- Most of the tests in DIEHARD return a **p-value**, which should be **uniform on [0,1)** if the input file contains truly independent random bits.
- **p=1-F(X)**, where **F is the assumed distribution of the sample random variable X**.
- p-values near 0 or 1 for passing the test near aprrox for all the dieheard tests .

Test overview of dihard

- **Birthday spacings:** spaced points should be asymptotically exponentially distributed.
- **Overlapping permutations:** Analyze sequences of five consecutive randomnumbers. The 120 possible orderings should occur with statistically equal probability.
- **Ranks of matrices:**Select some number of bits random numbers to form a matrix over {0,1}, then determine the rank of the matrix. Count the ranks .
- **Monkey tests:**Treat sequences of some number of bits as "words". Count the overlapping words in a stream.
- **Count the 1s:** Count the 1 bits in each of either successive or chosen bytes. Convert the counts to "letters", and count the occurrences of five-letter "words".
- **Parking lot test:**Randomly place unit circles in a 100×100 square. A circle is successfully parked if it does not overlap an existing successfully parked one.

Test overview of Diehard

- **Minimum distance test:**Randomly place 8000 points in a 10000×10000 square,then find the minimum distance between the pairs. The square of this distance should be exponentially distributed.
- **Random spheres test:**Randomly choose 4000 points in a cube of edge 1000. Center a sphere on each point, whose radius is the minimum distance to another point. The smallest sphere's volume should be exponentially distributed with a certain mean.
- **The squeeze test:**Multiply 231 by random floats on (0,1) until you reach 1. Repeat this 100000 times. The number of floats needed to reach 1 should follow a certain distribution.
- **Overlapping sums test:**Generate a long sequence of random floats on (0,1). Add sequences of 100 consecutive floats. The sums should be normally distributed with characteristic mean and variance.
- **The craps test:** Play 200000 games of craps, counting the wins Each count should follow a certain distribution.
- **Runs test:**Count ascending and descending runs. The counts should follow a certain distribution.

NIST Statistical Tests

- Each **NIST STS test** is defined by the test statistic of one of the following three types and examines randomness of the sequence according to:
- **1. bits –** these tests analyse various characteristics of bits like **proportion of bits,frequency of bit change** (runs) and **cumulative sums**.
- **2. m-bit blocks –** these tests analyse distribution of m-bit blocks (**m is typically smaller than 30 bits**) within the sequence or its parts,
- **3. M-bit parts –** these tests analyse complex property of M-bit (**M is typically larger than 1000 bits**) parts of the sequence like **rank of the sequence viewed as a matrix, spectrum of the sequence** or **linear complexity of the bitstream.**

Overview of NIST Tests:

1. The Frequency (Monobit) Test.
2. Frequency Test within a Block.
3. The Runs Test.
4. Tests for the Longest-Run-of-Ones in a Block.
5. The Binary Matrix Rank Test.
6. The Discrete Fourier Transform (Spectral) Test.
7. The Non-overlapping Template Matching Test.
8. The Overlapping Template Matching Test.

Overview of NIST tests

9. Maurer's "Universal Statistical" Test.
10. The Linear Complexity Test.
11. The Serial Test.
12. The Approximate Entropy Test.
13. The Cumulative Sums (Cusums) Test.
14. The Random Excursions Test.
15. The Random Excursions Variant Test.

3D Matrix approach

- We took a **3 dimensional array (10*10*10 matrix)**
- And each time a random number is being generated, firstly all of the **1000 cells are filled with random numbers generated by rand() function.**
- Then one of the cells is chosen using rand() by getting **its co-ordinate using rand() function** and value of that coordinate is returned as one random to the user.

ALGO:3D Matrix approach

- next ← 1
- **RANDR_R**(*seed)

  *seed ← *seed * 1103515245 + 12345;

  return (*seed % ((unsigned int)RAND_MAX + 1))
- **rand**(void)

  return (**RAND_R**(&next))
- **srand**(seed)

  next ← seed

Result:3D Matrix approach

- **Results:**

  1. Diehard tests passed: 4/15

  2. NIST Statistical suite: 3/15

  Outcome: UNSATISFACTORY

- **Reasons for outcome:**

  Not properly following the propertoies of PRNGs

- Not time Efficient.
- Not purely deDeterministic.

Circular array approach

- We took a **closed circular array of n** (n is **prime**)numbers and inserted **seed** value as **n-length circular array** (initial values) of one digit each to array blocks of our own choice.
- Everytime a random number is needed to be generated, the array is changed by using some simple linear equation like **self_new=(left*self) +right**, and the n-length subarray is the the n-length pseudo random number generated.

ALGO:Circular array approach

- **FUNC**(A, l , s, r)

```
return (((A[s]*A[r])+A[l])+2)%10 ;
```

- **RANDOM_N(A)**

```
MAX← length(A)
for s 0← to MAX
    if s = 0
        then r← s+1
             l← MAX-1
             ARRAY[s]← FUNC(A,l,s,r)
    else if s= MAX-1
        then l← s-1
             r← 0
             ARRAY[s]← FUNC(A,l,s,r)
```

ALGO:Circular array approach

```
        else
            l← s-1
            r← s+1
            ARRAY[s]← FUNC(arr,l,s,r)
    for s 0← to MAX
        do A[s]← ARRAY[s]
    x← 0
    for j 0← to 9
        do x← x+ A[j]*pow(10,9-1-j)
    return x
```

Time Complexity: O(n)

Result: Circular Array Approach

1. Diehard tests passed**: 11/15**

2. NIST Statistical tests passed**: 12/15**

- **Outcome:** VERY GOOD generator and better than most.

- **Reasons for outcome:**

Efficient.

Deterministic.

Low -Periodicity.

Results of the circular array approach: Diehard

- root@LAPTOP-HJK1OTMN:/home/project/diehard# ./proggie filetest.txt
- Name of input file = /home/project/diehard/ran14.bin
- 0.482846 | -nan | 0.246915 | 0.792844 | 0.942610 | 16/20 | 19/23 | 28/28 | 30/31 |0.082209 | 25/25 | 0.000000 | 0.382962 | 0.608611 | 0.147101 | 0.176678 | 0.270971| 0.124389 | 0.626044 | 0.963108 |

- **Diehard tests passed = 11 out of 15**

Results of the circular array approach: Diehard

- **Test passed**

1.Birthday Spacings

2.Ranks of 31x31 and 32x32 matrices

3. Ranks of 6x8 Matrices

4.Count the 1`s in a Stream of Bytes

5.Count the 1`s in Specific Bytes

6.Minimum Distance Test

7.Random Spheres Test

8.The Sqeeze Test

9. Overlapping Sums Test

10.Runs Test

11. The Craps Test

- **Test failed**

1.Overlapping Permutations

2.Monkey Tests on 20-bit Words

3.Monkey Tests OPSO,OQSO,DNA

4.Parking Lot Test

NIST tests passed = 12 out of 15

- **Tests passed**
  1. Frequency
  2. BlockFrequency
  3. CumulativeSums
  4. Runs
  5. LongestRun
  6. Rank
  7. FFT
  8. OverlappingTemplate
  9. Universal
  10. RandomExcursions
  11. RandomExcursionsVariant
  12. LinearComplexity

- **Tests failed**
  1. NonOverlappingTemplate
  2. ApproximateEntropy
  3. Serial

Results of other well-known PRNGs

Name of PRNG	Diehard tests	NIST tests
1. Borland LCG	1	4-5
2. rand()	1	2-3
3. random()	1	1
4. PCG-32	9-10	13
5. MT-19937-32	8-10	12-14
6. XORShift-32	2-4	7-10
7. SFMT-32	9-10	15
8. 3-state CA	2-3	0-6
9. Hybrid CA with rules 30 & 45	0-3	0-3
10. WELL 512a	7-10	14
11. Tauss88	0-1	0
12. LFSR113	5-11	0-12
13. LFSR258	0-11	0-13
14. MRG31k3p	0-1	1-2
15. lrand48()	1	2

CONCLUSIONS & FUTURE SCOPE

- **Finally, we can conclude that the pseudo random number generator made by us using the circular array approach is an efficient and a very good PRNG.**
- **The only competitor to our algorithm are some Mersenne Twister PRNGs of the Linear Feedback Shift register(LFSR) class.**
- **This algorithm has competition with only the best PRNGs available, now in its earlystage itself, thus it makes it the potentially best PRNG.**