阅读文献，叙述一类随机数生成原理：硬件随机数生成器

量子随机数发生器

利用量子力学中量子态可叠加和分解的特性，以及转换成经典粒子特性时的测量塌缩的随机性，得到基于量子物理学基本原理的真随机码串的一种装置。这种随机性起源于 量子物理学中关于微观世界的波粒二象性，其随机性无法进行人为控制，完全由物理学定律决定，由此产生的随机数称为量子随机数发生器。在密码学、博弈等需要随机数不可控制的应用场合具有重要价值。量子随机数发生器可以分为三大类：第一类是实用化的量子随机数发生器，这类发生器需要充分信任设备，即只有可靠的设备产生的随机数才能产生量子随机数，这种量子随机数发生器的码产生率可以很高；第二类是可自检测的量子随机数发生器，可以在不需要保证设备完全可信的条件下产生量子随机数；第三类是半自检测量子随机数发生器，它介于前两类之回，只需要部分信任设备就能获得相对较高速率的量子随机数。

随机数检验工具

NIST测试工具

NIST 测试套件是由15个测试组成的统计软件包，这些是为了测试随机（任意长度）由基于硬件或软件的密码随机或伪随机数生成器产生的二进制序列。测试关注于各种不同类型的已存在的非随机序列。有些测试可以分成各种子测试。

15个测试主要是（属于密码算法安全测试方法）：

1.频率（单比特）测试

```
1  频率测试是一种用于测量信号的频率的方法。在单比特频率测试中，我们关注的是单个比特（二进制位）在一定时间内翻转的频率。
```

2.块内频数测试（Frequency Test within a Block）

```
1  一种用于检测生成的随机比特流块内的0和1的频率分布是否符合期望均匀分布的测试方法
```

3.动向（Run）测试

```
检测生成的随机比特流中重复连续相同比特（run）的数量，以验证生成的随机性。
```

4.最大游程检测

```
评估随机数生成器（RNG）产生的比特流中是否包含比期望长的连续重复序列（最大游程）。此测试统计最长的连续重复比特序列，然后与理论期望值进行比较，以确定生成的随机数据是否包含过多的非随机模式
```

5.二进制矩阵秩（Binary Matrix Rand）测试

```
将生成的比特流分成多个块，每个块形成一个二进制矩阵，然后计算这些矩阵的秩。测试的目标是确保这些矩阵的秩分布符合预期的统计特性，以验证生成的随机数据是否满足随机性的要求。
```

6.频谱测试

```
分析生成的数据的频域特性来验证其随机性
```

7.非重叠字匹配测试

```
检查生成的比特流中是否包含特定的非重叠字（patterns）或序列
```

8.重叠字匹配测试

```
检查生成的比特流中是否包含特定的重叠字（patterns）或序列
```

9.Maurer通用统计检测

```
检测生成的比特流是否包含统计特性，如不可压缩性和高度复杂性
```

10、线性复杂度测试

检查生成的比特流中是否存在线性关系，以验证生成的数据是否具有足够的随机性，而不容易被预测或分析。

11、系列（Serial）测试

关注生成的比特流中是否包含特定的连续序列或模式，以验证生成的数据是否满足随机性要求。此测试通常包括检查比特流中相邻序列的相关性，以检测是否存在重复或可预测的序列

12、近似熵测试

测量生成的比特流中包含的信息熵或随机性水平，以确保生成的数据足够随机且具有高度不确定性

13、累积和测试

将生成的比特流中的比特依次相加，形成累积和，并检查这些和的统计特性

14、随机游程（Random Excursions）测试

检测在生成的比特流中是否存在显著的随机游程，即连续的相同或不同比特序列

15、随机游程变量（Random Excursions Variant）测试

随机游程测试的变种，专注于检测在生成的比特流中是否存在显著的随机游程，即连续的相同或不同比特序列，并分析游程的统计特性以验证数据的随机性。

使用举例

1.使用python随机库random中随机函数生成十万个随机数，以二进制形式保存至文件pyRandomNumbers中，再使用nist测试工具测试，查看测试结果

```python
import numpy as np
import random
random_numbers = [random.random() for _ in range(100000)]
# 转化为8位二进制数
bin_random_numbers = [format(int(num*255),"08b") for num in
random_numbers]
# 写入文件
with open("pyRandomNumbers",'wb') as file:
    for bin_num in bin_random_numbers:
        byte = int(bin_num,2).to_bytes(1,'big')
        file.write(byte)
```

cd到assess文件所在目录下。使用./assess启动程序
参数100000表示数据块的长度，这里有80万个byte，取8组测试 (How many bitstreams? 8)，每组10万byte。(./assess 100000)

```
luojunxun@ljx sts-2.1.2 % ./assess 100000
           G E N E R A T O R    S E L E C T I O N
           ______________________________________

     [0] Input File                 [1] Linear Congruential
     [2] Quadratic Congruential I   [3] Quadratic Congruential II
     [4] Cubic Congruential         [5] XOR
     [6] Modular Exponentiation     [7] Blum-Blum-Shub
     [8] Micali-Schnorr             [9] G Using SHA-1

    Enter Choice: 0


     User Prescribed Input File: ../../pyRandomNumbers

                S T A T I S T I C A L   T E S T S
                _________________________________

     [01] Frequency                      [02] Block Frequency
```

```
     [03] Cumulative Sums                 [04] Runs
     [05] Longest Run of Ones             [06] Rank
     [07] Discrete Fourier Transform      [08] Nonperiodic Template
Matchings
     [09] Overlapping Template Matchings  [10] Universal Statistical
     [11] Approximate Entropy             [12] Random Excursions
     [13] Random Excursions Variant       [14] Serial
     [15] Linear Complexity

          INSTRUCTIONS
             Enter 0 if you DO NOT want to apply all of the
             statistical tests to each sequence and 1 if you DO.

    Enter Choice: 1

         P a r a m e t e r   A d j u s t m e n t s
         ----------------------------------------
     [1] Block Frequency Test - block length(M):         128
     [2] NonOverlapping Template Test - block length(m): 9
     [3] Overlapping Template Test - block length(m):    9
     [4] Approximate Entropy Test - block length(m):     10
     [5] Serial Test - block length(m):                  16
     [6] Linear Complexity Test - block length(M):       500

    Select Test (0 to continue): 0

    How many bitstreams? 8

    Input File Format:
     [0] ASCII - A sequence of ASCII 0's and 1's
     [1] Binary - Each byte in data file contains 8 bits of data

    Select input mode:  1

      Statistical Testing In Progress.........

```

```
55      Statistical Testing Complete!!!!!!!!!!!
```

在目录experiments/AlgorithmTesting下，有15个测试类型的文件夹和两个汇总txt文件，每个文件夹下有相应的result.txt和stats.txt，分别存储了测试的数据结果和分析
freq.txt和finalAnalysisReport.txt是测试的总体结果

```
1  luojunxun@ljx AlgorithmTesting % cat freq.txt
2  ------------------------------------------------------------------------
   ---------
3
4      FILE = ../../pyRandomNumbers    ALPHA = 0.0100
5  ------------------------------------------------------------------------
   ---------
6
7      BITSREAD = 100000 0s = 50080 1s = 49920
8      BITSREAD = 100000 0s = 50176 1s = 49824
9      BITSREAD = 100000 0s = 50353 1s = 49647
10     BITSREAD = 100000 0s = 50472 1s = 49528
11     BITSREAD = 100000 0s = 50028 1s = 49972
12     BITSREAD = 100000 0s = 50366 1s = 49634
13     BITSREAD = 100000 0s = 50272 1s = 49728
14     BITSREAD = 100000 0s = 50166 1s = 49834
15 luojunxun@ljx AlgorithmTesting % cat finalAnalysisReport.txt
16 ------------------------------------------------------------------------
   -------
17 RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING
   SEQUENCES
18 ------------------------------------------------------------------------
   -------
19    generator is <../../pyRandomNumbers>
20 ------------------------------------------------------------------------
   -------
21  C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION
   STATISTICAL TEST
```

```
------------------------------------------------------------------------------
  4   0   2   0   0   0   1   0   1   0     ----       7/8
Frequency
  0   0   0   1   1   1   0   1   3   1     ----       8/8
BlockFrequency
  4   1   0   0   1   2   0   0   0   0     ----       7/8
CumulativeSums
  4   1   0   1   1   1   0   0   0   0     ----       6/8    *
CumulativeSums
  4   1   0   1   1   1   0   0   0   0     ----       8/8       Runs
  8   0   0   0   0   0   0   0   0   0     ----       3/8    *
LongestRun
  0   0   0   2   2   2   1   1   0   0     ----       8/8       Rank
  2   0   1   0   1   1   0   0   1   2     ----       8/8       FFT
  0   0   0   1   0   1   1   2   1   2     ----       8/8
NonOverlappingTemplate
  ...
  ...
  0   0   0   0   0   0   0   0   0   0     ----     ------
RandomExcursionsVariant
  1   0   0   0   1   3   0   1   1   1     ----       8/8       Serial
  1   0   2   2   0   1   0   1   0   1     ----       8/8       Serial
  1   2   1   2   1   0   1   0   0   0     ----       8/8
LinearComplexity


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
The minimum pass rate for each statistical test with the exception of
the
random excursion (variant) test is approximately = 7 for a
sample size = 8 binary sequences.

The minimum pass rate for the random excursion (variant) test is
undefined.
```

```
46
47  For further guidelines construct a probability table using the MAPLE
    program
48  provided in the addendum section of the documentation.
49  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    - - - - -
50
```

2.线性同余生成器

```
1   class myLCG():
2       def __init__(self,seed=10086):
3           # KISS84
4           self.a = 69069
5           self.c = 12345
6           self.m = 2**32
7           self.state = seed
8
9       def LCG(self):
10          result = (self.a * self.state + self.c) % self.m
11          self.state = result
12          return result
13
14  op = myLCG()
15  my_random_numbers = [op.LCG()/op.m for _ in range(100000)]
16  my_bin_random_numbers = [format(int(num*255),'08b') for num in
    my_random_numbers]
17  with open("myRandomNumbers",'wb') as file:
18      for bin_num in my_bin_random_numbers:
19          byte = int(bin_num,2).to_bytes(1,'big')
20          file.write(byte)
```

```
1   luojunxun@ljx AlgorithmTesting % cat freq.txt
2   _____________________________________________________________________
    _________
```

```
3
4       FILE = ../../myRandomNumbers    ALPHA = 0.0100
5   ------------------------------------------------------------------------
    ---------
6
7       BITSREAD = 100000 0s = 50313 1s = 49687
8       BITSREAD = 100000 0s = 49938 1s = 50062
9       BITSREAD = 100000 0s = 50134 1s = 49866
10      BITSREAD = 100000 0s = 50055 1s = 49945
11      BITSREAD = 100000 0s = 50126 1s = 49874
12      BITSREAD = 100000 0s = 50142 1s = 49858
13      BITSREAD = 100000 0s = 50216 1s = 49784
14      BITSREAD = 100000 0s = 50086 1s = 49914
15  luojunxun@ljx AlgorithmTesting % cat finalAnalysisReport.txt
16  ------------------------------------------------------------------------
    -------
17  RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING
    SEQUENCES
18  ------------------------------------------------------------------------
    -------
19     generator is <../../myRandomNumbers>
20  ------------------------------------------------------------------------
    -------
21   C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION
    STATISTICAL TEST
22  ------------------------------------------------------------------------
    -------
23    1   1   0   2   1   1   1   1   0   0     ----       8/8
    Frequency
24    0   0   0   0   3   0   0   1   2   2     ----       8/8
    BlockFrequency
25    1   2   1   0   0   0   2   1   0   1     ----       8/8
    CumulativeSums
26    ...
27    ...
28    0   0   0   1   1   2   2   1   1   0     ----       8/8       Serial
```

```
  0   2   1   2   0   1   0   0   1   1     ----       8/8
LinearComplexity


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
The minimum pass rate for each statistical test with the exception of
the
random excursion (variant) test is approximately = 7 for a
sample size = 8 binary sequences.

The minimum pass rate for the random excursion (variant) test
is approximately = 0 for a sample size = 1 binary sequences.

For further guidelines construct a probability table using the MAPLE
program
provided in the addendum section of the documentation.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

```

可见迭代格式$X_i = (69069X_{i-1}12345) \mod 2^{32}$的随机数生成效果也很好