单位： 紫金山实验室

部门： 前沿交叉

项目： 安全密钥分发

密级：

日期： 2020.03

基于国家标准（GB/T32915-2016）的二元序列随机性检测软件

用户操作手册

编写说明

**类别：项目文档**

**密级：**

**撰稿人：王旭阳**

**修改人：**

**编辑软件：vs2019**

**版本信息：1.0**

|  |  |
| --- | --- |
| **版本** | **修改点说明** |
|  |  |
|  |  |

**目录**

1 引言 1-1

1.1 编写目的 1-1

1.2 项目背景 1-1

1.3 术语定义 1-1

1.4 参考资料 1-1

2 软件概述 2-3

2.1 目标 2-3

2.2 功能 2-3

2.3 性能 2-3

3 运行环境 3-4

3.1 硬件 3-4

3.2 支持软件 3-4

4 使用说明 4-5

4.1 安装和初始化 4-5

4.2 输入 4-5

4.2.1 数据背景 4-5

4.2.2 数据格式 4-5

4.2.3 输入举例 4-6

4.3 输出 4-6

4.3.1 数据背景 4-6

4.3.2 数据格式 4-6

4.3.3 举例 4-6

4.4 出错和恢复 4-6

4.5 求助查询 4-6

5 运行说明 5-7

5.1 运行表 5-7

5.2 运行步骤 5-7

5.2.1 运行控制 5-7

5.2.2 操作信息 5-7

5.2.3 输入／输出文件 5-7

5.2.4 启动或恢复过程 5-8

6 非常规过程 6-9

7 操作命令一览表 6-10

8 程序文件(或命令文件)和数据文件一览表 6-11

9 用户操作举例 6-12

# 引言

随着信息技术的发展和个人信息安全意识的提高，人们对信息安全的需求也越来越高。随机数作为密码设计构成中的一个重要参数，在信息安全中扮演者重要的作用。其中，如何生成足够“随机”的二元序列和如何验证得到的二元序列是否足够“随机”是人们在使用随机数的过程中经常面对的两个问题。

## 编写目的

本手册的编写目的主要是为了阐述二元序列随机性检测软件所具备的功能、软件的使用方式和如何查看二元序列的随机性检测结果。进一步地，本手册还给出了二元序列随机性检测软件的更新和维护方式。

本手册主要面向从事与数据处理相关的事务的科研工作人员。

## 项目背景

项目来源：紫金山实验室

主管部门：前沿交叉科学研究中心

项目课题：安全密钥分发研究部

## 术语和符号定义

本手册中使用的专门术语的定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 专门术语 | 说明/定义 | 英文全称 |
| 1 | 二元序列 | 由“0”和“1”组成的比特串 | Binary sequence |
| 2 | 随机数发生器 | 产生随机二元序列的器件或程序 | Random number generator |
| 3 | 随机性检测 | 用于二元序列检测的函数或过程 | Randomness test |
| 4 | 显著性水平 | 随机性检测中错误地判断某一个二元序列为非随机序列的概率，用表示 | Significance level |
| 5 | 样本 | 用于随机性检测的二元序列 | Sample |
| 6 | 样本长度 | 一个样本的比特长度 | Sample length |
| 7 | 样本数量 | 随机性检测的样本的个数 | Sample size |
| 8 | 检测参数 | 随机性检测需要设定的参数 | Test parameter |
| 9 | 频数 | 样本中数据的比例 | Frequency |
| 10 | 块 | 序列中由“0”和“1”组成的子序列 | Block |
| 11 | 游程 | 序列中由连续的“0”或“1”组成的子序列，并且该序列的前导或后继元素都与其本身的元素不同 | Run |
| 12 | 秩 | 矩阵中线性无关的行或列的值 | Rank |
| 13 | 熵 | 衡量数据体系随机性的值 | Entropy |

本手册中使用的（伪）随机数生成算法如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 随机数生成算法 | 说明/定义 | 英文全称 |
| 1 | 线性同余 | 通过线性同余生成随机数() | Linear congruential |
| 2 | Ⅰ型二次同余 | 通过二次同余生成随机数() | Quadratic congruential I |
| 3 | Ⅱ型二次同余 | 通过二次同余生成随机数() | Quadratic congruential II |
| 4 | Ⅱ型立方同余 | 通过立方同余生成随机数() | Cubic congruential II |
| 5 | 异或 | 通过异或运算生成随机数() | Exclusive OR (XOR) |
| 6 | 指数模 | 通过指数运算后取模生成随机数() | Modular exponentiation |
| 7 | Blum-Blum-Shub | 通过随机选择互质数据生成随机数() | Blum-Blum-Shub |
| 8 | Micali-Schnorr | 通过截尾，指数运算取模生成随机数 | Micali-Schnorr |
| 9 | SHA1 | 通过SHA1算法生成随机数 | Secure Hash (G-SHA1) |

本手册中所使用的随机数检测算法如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 随机数检测算法 | 说明/定义 | 英文全称 |
| 1 | 单比特频数检测 | 检测样本序列中“0”和“1”的频数 | Frequency |
| 2 | 块内频数检测 | 将样本分块，检测样本各块序列中“0”和“1”的频数 | Block frequency |
| 3 | 扑克检测 | 将样本分块，检测各子序列的频数 | Poker |
| 4 | 重叠子序列检测 | 按比特计算并比较样本中特定长度子序列的频数 | Serial |
| 5 | 游程总数检测 | 检测样本中游程总数的分布概率 | Runs |
| 6 | 游程分布检测 | 检测样本中特定长度游程的分布概率 | Runs frequency |
| 7 | 块内最大“1”游程检测 | 将样本分块，检测每个块中最大“1”游程的分布概率 | Longest run of ones |
| 8 | 二元推导检测 | 检测样本k次推导后新样本中“0”和“1”的频数，每次推导为对相邻元素进行异或操作 | Binary derivation |
| 9 | 自相关检测 | 检测样本与将其逻辑左移d位后得到的新样本的关联程度 | AutoCorrelation |
| 10 | 矩阵秩检测 | 通过特定尺寸矩阵的秩，检测样本中给定长度子序列数据之间的随机性 | Rank |
| 11 | 累加和检测 | 以0为基准，检测样本累加和偏移的分布情况 | Cumulative sums |
| 12 | 近似熵检测 | 分别计算两个相邻长度子序列在样本中的频数的熵，检测熵差值的分布情况 | Approximate entropy |
| 13 | 线性复杂度检测 | 将样本分块，检测各个子序列的线性复杂度 | Linear complexity |
| 14 | Maurer通用统计检测 | 检测样本能否显著地被无损压缩 | Universal statistical |
| 15 | 离散傅里叶检测 | 通过频谱的方法，检测样本执行傅里叶变换后的尖峰是否超过门限值 | Discrete Fourier transform |

本手册中使用的符号定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 符号 | 全称 | 说明/定义 |
| 1 |  | 显著性水平 | 用来界定置信区间的范围，置信度为1- |
| 2 |  | 样本长度 | 待检测二元序列的数据长度 |
| 3 | *m* | 块的长度 | 随机性检测算法执行分块操作时块子序列的数据长度 |
| 4 | p\_value | 余误差 | 样本随机性检测值，当p\_value时即可认为样本是随机的 |
| 5 | erfc | 双截尾函数 | 左右都截断的高斯函数，某些检测算法中通过它来计算余误差 |
| 6 | igamc | 不完全伽马函数 | 右截尾的卡方函数，某些检测算法中通过它来计算余误差 |

## 参考资料

* 项目计划任务书

提出二元序列随机性测试方法并给出符合国密要求的测试结果；

* 项目开发计划

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 起止时间 | 开发计划 | 预期目标 |
| 1 | 2019.11-2019.12 | 研究分析二元序列随机性检测的国家标准 | 熟悉了解相应的国家标准 |
| 2 | 2019.12-2020.2 | 根据二元序列随机性检测国家标准，编写相应的检测程序。 | 编写完成符合国家标准的检测程序 |
| 3 | 2020.2-2020.3 | 调试完善随机性检测程序 | 对测试程序进行调试，验证其正确性 |

* 项目开发历史

版本：Vision 1.0

开发者：王旭阳

时间：2020.3

* 采用的软件工程标准如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 文档编号 | 文档名称 | 作者/修订者 | 版本号 | 出版日期 |
| GB/T 32915-2016 | 信息安全技术二元序列随机性检测方法 | 李大为，冯登国等 | 第一版 | 2016 |
| NIST Special Publication 800-22 | A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications | Andrew Rukhin，Juan Soto， et.al | Revision 1a | April, 2010 |

* 参考的软件程序如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 软件名称 | 作者/修订者 | 发布日期 | 备注 |
| NIST Statistical Test Suite | Juan Soto | December 1999 | Version 1.0 by J. Soto, October 1999  Revised by J. Soto, November 1999  Revised by Larry Bassham, March 2008 |

# 软件概述

## 目标

根据国家标准GB/T 32915-2016 《信息安全技术 二元序列随机性检测方法》，构造符合上述国家标准的二元序列随机性检测软件。

## 功能

本软件继承了NIST随机性软件测试包的伪随机数生成和部分随机性检测功能，并根据随机性检测国家标准编写加入了部分NIST随机性软件测试包所不具有的随机性检测功能。具体来说，本软件具有以下功能：

1. 支持线性同余、Ⅰ型二次同余、Ⅱ型二次同余、Ⅱ型立方同余、异或、指数模、互质抽样和SHA1一共8种伪随机数生成方法，并对生成的伪随机数进行随机性检测；
2. 支持对用户的个人数据进行随机数检测；
3. 支持国家标准GB/T 32915-2016 《信息安全技术 二元序列随机性检测方法》种提出的所有15种检测方法，同时，用户可以根据需求自行选择所需要的检测方法；
4. 支持用户自行添加随机数生成算法和随机性检测算法。

## 性能

1. 数据

输入：

伪随机数生成算法可以生成任意长度的伪随机数二元序列；

可读取任意长度格式为Ascii或二进制的二元序列；

除Maurer通用统计检测外，其他算法理论上可对任意长度的二元序列进行随机性检测。但为了检测的准确性，建议待检测序列长度不低于10000比特；

Maurer通用统计检测对待检测序列的长度要求不低于40万比特。

输出：

执行的伪随机性生成算法并不直接输出生成的随机数，而是输出相应的随机性检测结果；

对于单个随机性检测，检测程序会在测试方法对应的文件夹输出检测结果。除了输出余误差之外，还输出检测算法在检测过程中的一些关键参数；

除此之外，检测程序还会对随机性检测得到的余误差的随机性进行分析，并给出相应的分析结果。

1. 灵活性

支持多种操作系统，在Windows10和Ubuntu16.04上均成功运行；

操作方便，用户只需按照要求，输入样本长度、来源即可成功运行；

版本更新方便，用户可随时更新相关随机数生成算法和检测算法。

# 运行环境

以下运行环境本软件均已成功运行。

## Ubuntu操作系统

系统版本号：16.04

编译器：gcc编译器

其他必要的支持软件：无

## Windows操作系统

系统版本号：win10-1909

编译器：gcc编译器

其他必要的支持软件：Cygwin

# 使用说明

## 安装和初始化

本软件提供了两种类型的随机性测试软件包，类型一软件包中只有编译通过的源程序，而不包含相应的编译文件；类型二软件包中只有已经成功编译的编译文件。

首先给出类型一的随机性测试软件包的安装和初始化说明。下面分别就Ubuntu和Windows两个操作系统给出相应的安装步骤：

1. Ubuntu操作系统：

存储形式：硬盘存储。直接将文件gbt-1.0复制到系统文件夹./home下。

初始化：

1. 终端控制台输入cd gbt-1.0打开文件；
2. 输入make执行编辑；
3. *输入./assess查看数据输入方式及要求；*
4. *按要求输入待检测数据长度./assess <message length>。*
5. Windows操作系统：

存储形式：硬盘储存。直接将文件gbt-1.0复制到用户文件夹，如D:/gbt-1.0。

安装Cygwin：根据操作系统的情况从网站https://cygwin.com/install.html下载Cygwin安装程序，并选择合适位置安装。

编译器安装：运行Cygwin，选择gcc编译器安装。

初始化：

1. 终端控制台输入文件地址打开文件，如cd d:/gbt-1.0；
2. 输入make执行编辑；如出现未找到命令错误提示，须从网站<http://ftp.gnu.org/gnu/make/> 中下载安装make工具包；
3. *输入./assess或./assess.exe查看数据输入方式及要求；*
4. *按要求输入待检测数据长度./assess <message length>。*

下面给出类型二的随机性测试软件包的安装和初始化说明。下面分别就Ubuntu和Windows两个操作系统给出相应的安装步骤：

1. Ubuntu操作系统：

存储形式：硬盘存储。直接将文件gbt-1.0复制到系统文件夹./home下。

初始化：

1. 终端控制台输入cd gbt-1.0打开文件；
2. *输入./assess查看数据输入方式及要求；*
3. *按要求输入待检测数据长度./assess <message length>。*
4. Windows操作系统：

存储形式：硬盘储存。直接将文件gbt-1.0复制到用户文件夹，如D:/gbt-1.0。

安装Cygwin：根据操作系统的情况从网站https://cygwin.com/install.html下载Cygwin安装程序，并选择合适位置安装。

编译器安装：运行Cygwin，选择gcc编译器安装。

初始化：

1. 终端控制台输入文件地址打开文件，如cd d:/gbt-1.0；
2. *输入**./assess或./assess.exe查看数据输入方式及要求；*
3. *按要求输入待检测数据长度./assess <message length>。*

## 输入

根据流程，本软件需要用户输入样本长度、待检测样本数据来源和检测算法的检测参数。

根据操作系统的不同，用户可以在编译器中输入./assess或./assess.exe查看样本长度的输入方式及要求。

检测程序运行后，用户须判断待检测样本数据的获取途径，如果此数据为用户自己采集的，用户须按提示输入数据的存储位置。

输入检测数据后，系统会提示用户选择检测算法的检测参数。用户可选择采用系统的默认参数，如果用户想自行选择检测参数，可按提示输入相应的检测参数。

### 数据背景

样本长度：用来确定待检测样本的数据长度的参数，对应二元序列随机性检测国家标准中的样本长度n。用户每次使用随机性检测软件包时都需要确定待检测样本的数据长度。由于随机性测量算法统计近似于特定统计分布的，因此样本长度越长，检测结果就越准确。但是对于一些随机数生成器来说，多时间内生成足够长度的待检测样本是无法实现的事。因此，本检测软件建议样本长度在不低于10 000比特的情况下，用户根据其需求自行决定。需要注意的是，Maurer通用检测要求样本长度不低于400 000比特。

样本数据：本软件支持用户对自己采集的数据进行随机数验证，同时也提供了9种伪随机数生成算法可以生成相应的伪随机数。当用户执行本软件时，用户需选择待检测数据的来源，如果是用户自己采集的数据，用户需要根据提示输入数据的地址。需要注意的是，对于用户自己采集的数据，本软件要求数据为ASCII或二进制格式的文件。

检测参数：本软件提供了随机性检测国家标准给出的所有检测算法，其中，有些算法在实现过程种需要用户确定相应的检测参数。本软件根据国家标准给出了相应的预设参数。同时，考虑到有些用户的特殊需求，本软件支持用户自己设定特殊的检测参数。

### 数据格式

样本长度：用阿拉伯数字表示样本长度，具体输入格式为./assess <数字>。

样本数据：遵循文件的常用表示方法，默认读取位置为测试软件所在的文件夹。

检测参数：根据算法的不同，值为比样本长度小不同数量级的正整数。

### 输入举例

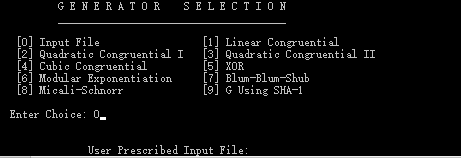
样本长度：

样本长度为1000 000比特，输入./assess 1000 000



样本数据：

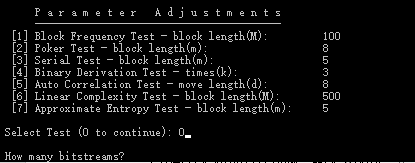
样本数据来源为用户自己采集，输入0；再输入样本数据地址。



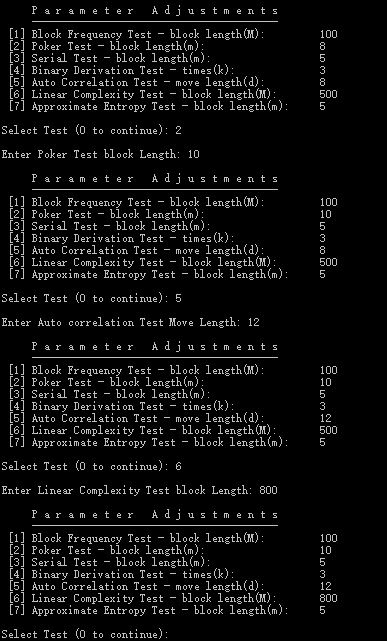
样本数据来源为自带伪随机算法生成，选择相应的伪随机算法，输入对应的编号即可。

检测参数：

采用软件提供的默认参数，输入0即可。



用户自行选择检测参数，依次输入对应的编号，然后输入期望中的参数。



## 输出

本软件的主要功能是检测二元序列的随机性，并给出相应的检测结果。因此，本软件的输出主要是随机性检测的相关结果，默认存储位置为软件所在文件夹内的./experiments子文件夹内。软件的输出按照随机数来源的不同，分成了10个输出文件夹。在每个输出文件夹中，根据检测结果存储位置的不同，软件输出分为两个部分：一部分位于输出文件夹的主文件夹，为整体输出结果，如给出了同一来源数据通过随机性性检测后各个样本的余误差的分布情况；一部分位于输出文件夹的子文件夹（以随机性检测算法名字命名），输出随机性检测所采用的具体算法所采用的检测参数和检测中间值等关键参数。

### 数据背景

本软件的输出数据默认存储于软件所在文件夹内的./experiments子文件夹内，experiments文件夹共有10个子文件夹，分别对应了检测软件所支持的10种二元序列输入方式，具体子文件夹如下：

./experiments /AlgorithmTesting：对应待检测随机数来源是[0]用户自行采集的情况。当样本数据是用户自行采集的时，完成随机性检测后，软件会将随机性检测结果存储到此文件夹中；

./experiments /LCG：对应待检测随机数来源是通过[1]线性同余算法生成的情况。当样本数据是线性同余算法生成的时，完成随机性检测后，软件会将随机性检测结果存储到此文件夹中；

./experiments /QCR1：对应待检测随机数来源是通过[2]Ⅰ型二次同余算法生成的情况。当样本数据是Ⅰ型二次同余算法生成的时，完成随机性检测后，软件会将随机性检测结果存储到此文件夹中；

./experiments /QCR2：对应待检测随机数来源是通过[3]Ⅱ型二次同余算法生成的情况。当样本数据是Ⅱ型二次同余算法生成的时，完成随机性检测后，软件会将随机性检测结果存储到此文件夹中；

./experiments /CCG：对应待检测随机数来源是通过[4]Ⅱ型立方同余算法生成的情况。当样本数据是Ⅱ型立方同余算法生成的时，完成随机性检测后，软件会将随机性检测结果存储到此文件夹中；

./experiments /XOR：对应待检测随机数来源是通过[5]异或算法生成的情况。当样本数据是异或算法生成的时，完成随机性检测后，软件会将随机性检测结果存储到此文件夹中；

./experiments /MODEXP：对应待检测随机数来源是通过[6]指数模算法生成的情况。当样本数据是指数模算法生成的时，完成随机性检测后，软件会将随机性检测结果存储到此文件夹中；

./experiments /BBS：对应待检测随机数来源是通过[7] Blum-Blum-Shub算法生成的情况。当样本数据是Blum-Blum-Shub算法生成的时，完成随机性检测后，软件会将随机性检测结果存储到此文件夹中；

./experiments /MS：对应待检测随机数来源是通过[8] Micali-Schnorr算法生成的情况。当样本数据是Micali-Schnorr算法生成的时，完成随机性检测后，软件会将随机性检测结果存储到此文件夹中；

./experiments /G-SHA1：对应待检测随机数来源是通过[9] SHA1算法生成的情况。当样本数据是SHA1算法生成的时，完成随机性检测后，软件会将随机性检测结果存储到此文件夹中。

前文中说过，随机性检测的检测结果分为两个部分。在检测完成后，每个子文件夹都会给出最终的随机性检测结果finalAnalysisReport.txt和相应的比特分布情况freq.txt。

进一步地，为了方便用户了解样本在通过具体随机性检测算法时的情况，每个子文件夹还额外包含15个与随机性检测算法对应的子文件夹。当用户选择了某个检测算法后，在检测完成后，此检测算法子文件夹会给出相应的关键检测数据stats.txt和最终结果results.txt。

### 数据格式

finalAnalysisReport.txt：由两个部分组成：检测样本数据来源和相应的检测结果。进一步地，检测结果包括检测所采用的检测算法、p\_value值和样本通过检测的比例。

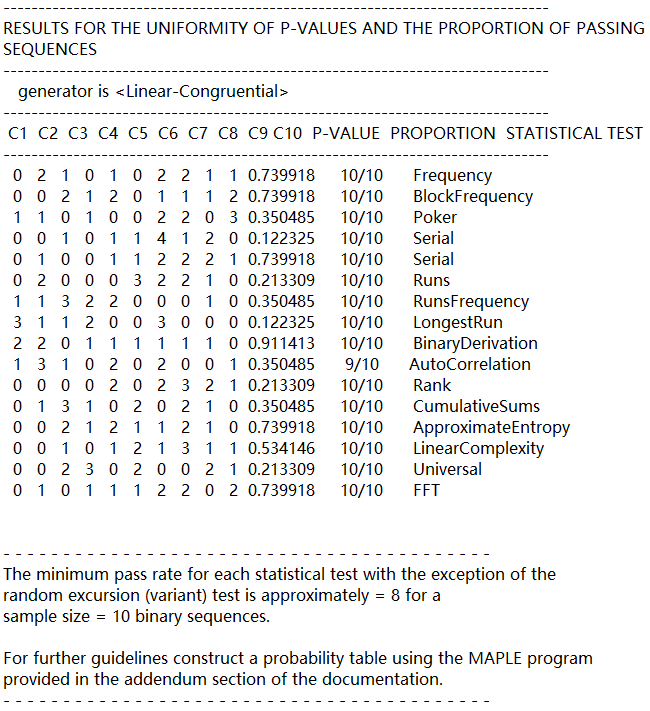
freq.txt：由两个部分组成：检测样本数据来源和频数分布。频数分布为样本中0和1的元素个数统计情况。

stats.txt：由两个部分组成：检测关键中间值情况和p\_value值情况。

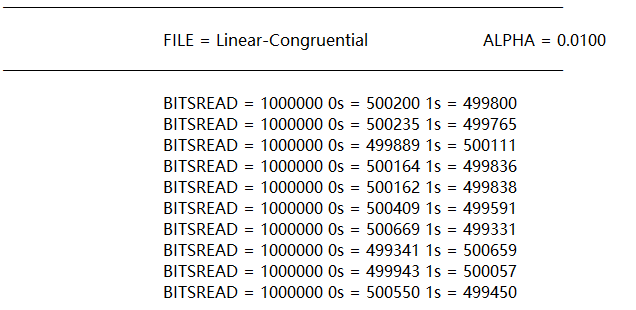
results.txt：列出了一次检测中所有样本的p\_value检测结果。

### 举例

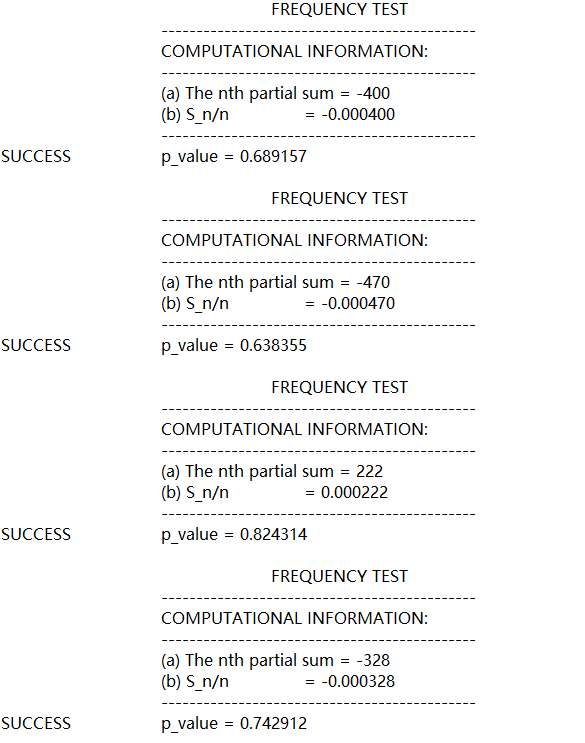
finalAnalysisReport.txt：检测由线性同余算法生成的伪随机数，输出结果位于./gbt/experiments/LCG



freq.txt：检测由线性同余算法生成的伪随机数，输出结果位于./gbt/experiments/LCG



stats.txt：用单比特频数检测算法检测由线性同余算法生成的伪随机数，输出结果位于./gbt/experiments/LCG/Frequency



results.txt：用单比特频数检测算法检测由线性同余算法生成的伪随机数，输出结果位于./gbt/experiments/LCG/Frequency



## 出错和恢复

输入错误类型：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 出错信息 | 含意 | 修改措施 |
| 1 | File Error: file <\*\*> could not be opened. | 文件地址输入错误导致待测样本数据文件无法打开 | 输入正确的文件地址 |
| 2 | ERROR IN FUNCTION fileBasedBitStreams: file <\*\*> could not be opened.\n | 测试样本数据格式错误导致测试无法进行 | 修改样本数据格式为ASCII或8位二级制 |
| 3 | ERROR: Insufficient data in file <\*\*> bits were read. | 测试样本没有足够的数据导致测试无法进行 | 1. 增加样本数据量 2. 减少检测样本长度 |
| 4 | BITSTREAM DEFINITION: Insufficient memory available.  Statistical Testing Aborted! | 检测程序没有申请到足够内存，无法完成动态数组申请 | 1，减少检测样本长度，从而减少数组长度  2，加大计算机内存 |
| 5 | igamc: UNDERFLOW | igamc函数计算时下溢出 | 检测样本p\_value太小导致igamc算法无法准确计算p\_value值 |

输出错误类型：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 出错信息 | 含意 | 修改措施 |
| 1 | ERROR: L IS OUT OF RANGE.  -OR-: Q IS LESS THAN 320.000000.  -OR-: Unable to allocate T.  UNIVERSAL STATISTICAL TEST | 检测样本长度小于40W或无法申请到足够内存导致通用测试无法完成 | 1. 样本长度小于40W时增加样本长度或不采用通用检测 2. 样布长度符合要求时加大内存 |
| 2 | p\_value值为0 | 余误差函数值为0 | 测试样本随机性不足导致 |

# 运行说明

## 运行表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 运行情况 | 运行目的 | 指令 |
| 1 | 软件编译 | 对随机性检测软件进行编译，为进一步运行做准备 | make |
| 2 | 输入帮助查看 | 查看检测软件对于待测数据的样本长度输入格式要求 | ./assess.exe |
| 3 | 样本长度输入 | 输入待测数据的样本长度 | ./assess <length> |
| 4 | 样本来源选择 | 选择待测数据的数据来源 | 0-9 |
| 5 | 样本数据读取 | 当样本为用户提供时，软件读取相应的用户数据 |  |
| 6 | 检测算法选取 | 用户根据需求，自行从15个算法中选择合适的算法 | 0或1 |
| 7 | 检测参数选取 | 用户根据需求，选择合适的算法参数 |  |
| 8 | 样本格式选择 | 用户根据样本的数据格式，选择相应的数据读取方式 | 0或1 |
| 9 | 检测运行中 | 根据用户的选择，软件执行样本数据的随机性检测 |  |
| 10 | 输出检测结果 | 检测结束，软件输出相应的检测结果 |  |

## 运行步骤

本软件的运行包括软件启动、运行操作与控制、输入和输出等步骤，具体的运行步骤如下：

### 运行控制

|  |  |  |
| --- | --- | --- |
| 序号 | 阶段 | 功能 |
| 1 | 编译 | 对随机性检测软件进行编译 |
| 2 | 启动 | 启动随机性检测软件 |
| 3 | 运行 | 执行随机性测试软件，对样本进行检测 |
| 4 | 输入 | 包括输入样本，操作和控制数据等 |
| 5 | 操作与控制 | 对随机性检测过程中，用户根据需求进行相应操作 |
| 6 | 输出 | 当样本为用户提供时，软件读取相应的用户数据 |

编译

未成功启动

启动

成功启动

运行

输入

操作和控制

输出

### 操作信息

1. 软件编译

运行目的：对随机性检测软件进行编译，为进一步运行做准备

预计运行时间：即时

操作命令格式：make

命令说明：无

1. 输入帮助查看

运行目的：查看检测软件对于待测数据的样本长度输入格式要求

预计运行时间：即时

操作命令格式：./assess.exe或./assess

命令说明：./assess.exe为标准输入格式，./assess不标准但是不妨碍输出相应的结果

1. 样本长度输入

运行目的：输入待测数据的样本长度

预计运行时间：即时

操作指令格式：./assess <length>

命令说明：./assess和长度之间是有空格的，否则软件会报错：No such file or directory。长度为标准的阿拉伯数字表达方式，如样本长度为一万比特即为./assess 10000，中间不需要加空格，否则软件会按操b)执行。

1. 样本来源选择

运行目的：选择待测数据的数据来源

预计运行时间：即时

操作指令格式：0-9

命令说明：序号0-9分别表示不同的样本数据来源，其中0表示数据为用户自己采集提供，1-9为伪随机数生成器生成。

1. 样本数据读取

运行目的：当样本为用户提供时，软件读取相应的用户数据

预计运行时间：即时

操作指令格式：绝对地址\数据名称或相对地址\数据名称

命令说明：无。

1. 检测算法选取

运行目的：用户根据需求，自行从15个算法中选择合适的算法

预计运行时间：即时

操作指令格式：是否运行全部算法（0或1），不运行全部算法时选择指定算法（0或1）

命令说明：软件会首先提醒用户是否选择执行所有检测算法，如果是，输入1；如果不执行，输入0。当用户选择不执行所有检测算法时，系统会输出123456789111111，分别代表所有15个检测算法。用户在需要执行的算法位置输入1即可调动相应的算法，如打算执行第2个和第10个算法，只需输入010000000100000即可。

1. 检测参数选取

运行目的：用户根据需求，选择合适的算法参数

预计运行时间：即时

操作指令格式：是否按默认参数执行（0或1），不按默认参数执行时输入新参数

命令说明：软件会首先提醒用户是否按默认参数执行检测算法，如果是，输入0；如果不是，用户先输入不按默认参数执行的算法对应的序号（与算法对应序号不同，此时最多有7个算法支持用户输入参数，因此序号最多为1-7），然后输入期望修改后的参数值。

1. 样本格式选择

运行目的：用户根据样本的数据格式，选择相应的数据读取方式

预计运行时间：即时

操作指令格式：0（ASCII编码）或1（8位二进制编码）

命令说明：用户需要按照样本数据格式输入正确的选项，否则检测算法会输出错误的检测结果。

### 输入／输出文件

详见本使用手册4.2，4.3节。

# 操作命令一览表

|  |  |  |
| --- | --- | --- |
| 序号 | 操作指令 | 功能 |
| 1 | make | 对随机性检测软件进行编译，获取可执行文件 |
| 2 | ./assess | 启动随机性检测软件 |
| 3 | ./assess.exe | 启动随机性测试软件 |
| 4 | ./assess <length> | 输入样本长度 |
| 5 | make clean | 清理编译得到的可执行文件 |

# 程序文件(或命令文件)和数据文件一览表

头文件

|  |  |  |
| --- | --- | --- |
| 序号 | 文件名 | 说明 |
| 1 | cephes.h | 统计函数相关 |
| 2 | config.h | 全局变量相关 |
| 3 | decls.h | 全局常量相关，包括随机数生成算法和检测算法 |
| 4 | defs.h | 随机性检测算法定义相关 |
| 5 | extern.h | 输出相关 |
| 6 | generators.h | 随机数生成算法相关 |
| 7 | genutils.h | 数学运算相关 |
| 8 | matrix.h | 矩阵操作相关 |
| 9 | stat\_fncs.h | 随机性检测算法函数定义 |
| 10 | utilities.h | 其他函数定义 |

源文件：

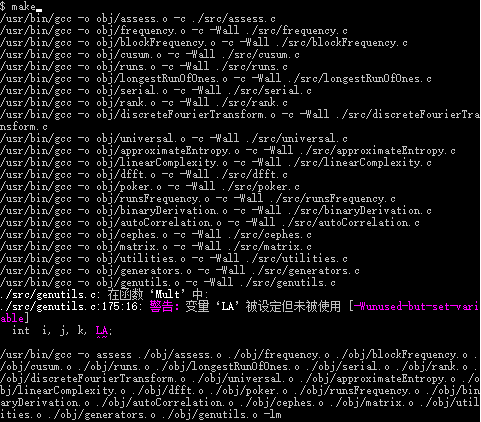
|  |  |  |
| --- | --- | --- |
| 序号 | 文件名 | 说明 |
| 1 | approximateEntropy.c | 近似熵检测算法 |
| 2 | assess.c | 主程序 |
| 3 | autoCorrelation.c | 自相关检测算法 |
| 4 | binaryDerivation.c | 二元推导检测算法 |
| 5 | blockFrequency.c | 块内频数检测算法 |
| 6 | cephes.c | 统计函数 |
| 7 | cusum.c | 累加和检测算法 |
| 8 | dfft.c | 快速傅里叶变换 |
| 9 | discreteFourierTransform.c | 离散傅里叶检测算法 |
| 10 | frequency.cn | 单比特频数检测算法 |
| 11 | generators.c | 随机数生成 |
| 12 | genutils.c | 数学运算 |
| 13 | linearComplexity.c | 线性复杂度检测算法 |
| 14 | longestRunOfOnes.c | 块内最大“1”游程检测算法 |
| 15 | matrix.c | 矩阵操作 |
| 16 | poker.c | 扑克检测算法 |
| 17 | rank.c | 矩阵秩检测算法 |
| 18 | runs.c | 游程总数检测算法 |
| 19 | runsFrequency.c | 游程分布检测算法 |
| 20 | serial.c | 重叠子序列检测算法 |
| 21 | universal.c | Maurer通用统计检测算法 |
| 22 | utilities.c | 其他函数 |

# 用户操作举例

1. 清除可执行文件



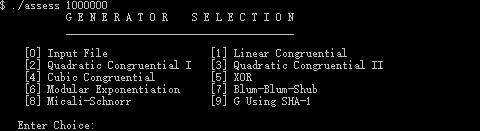
1. 编译



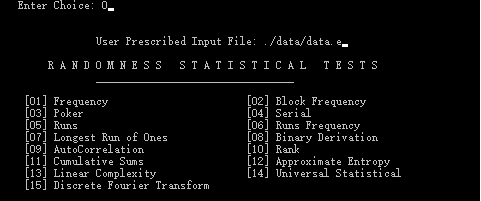
1. 启动



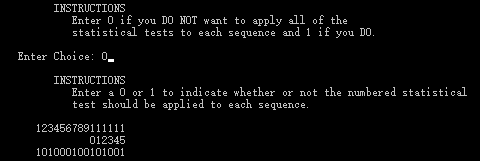
1. 输入样本长度



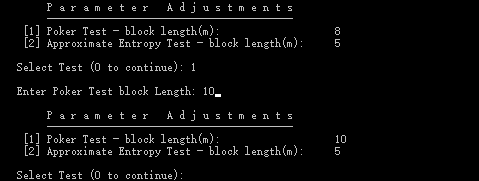
1. 选择样本来源并输入样本



1. 选择随机性检测算法



1. 确定算法参数



1. 输入样本数量



1. 选择样本数据结构类型



1. 输出



结果查看：

