Universidad del Valle de Guatemala
Cifrado de Información
Sección 10
Docente: Alan Reyes

Jose Pablo Ponce 19092
Gabriel Quiroz 19255
Alexa Bravo 18831
Agosto 20, 2021

REPORTE LABORATORIO 04

INTRODUCCIÓN

Los tests estadísticos se utilizan para validar o rechazar las hipótesis de modelación. Tratan de distinguir entre lo qué brinda un resultado satisfactorio de lo qué brinda un resultado erróneo en el marco del modelo dado. Existen distintos tipos de tests estadísticos, en donde se determinan diferentes requisitos de los modelos creados.

En este laboratorio se implementaron diez tests con el objetivo de evaluar la calidad de los generadores pseudoaleatorios creados en el laboratorio pasado. Luego se muestra una tabla con el resumen de los tests estadísticos y por último una gráfica en donde se pueden observar los tests qué fallaron. Esto se realizó con los tres ejemplos de cadenas qué se implementaron en el Laboratorio 3.

METODOLOGÍA

Batería de diez tests estadísticos

Se implementaron diez tests, los cuales se encuentran en los links de github brindado por el licenciado, a cada uno se le realizaron las modificaciones necesarias para funcionar con los generadores pseudoaleatorios. Los tests implementados son:

1. monobit_test.py
2. frequency_within_block_test.py
3. runs_test.py
4. longest_run_ones_in_a_block_test.py
5. binary_matrix_rank_test.py
6. linear_complexity_test.py
7. random_excursion_variant_test.py
8. random_excursion_test.py
9. approximate_entropy_test.py
10. random_excursion_test.py

Tabla de resumen de los diez tests estadísticos

Se realizó una función que, dada una cadena de bits, elabora una tabla con el resumen de los diez tests estadísticos implementados, esta función se aplicó a los ejemplos del Laboratorio 03 para poder tener una métrica de evaluación de los mismos.

Histograma de frecuencias

Se realizó una función qué con los generadores pseudo-aleatorios implementados en el Laboratorio 03, genera mil cadenas pseudo-aleatorias diversas y se ejecuta la batería de test para estas cadenas, luego se grafica un histograma de frecuencias del número de tests fallados por las cadenas aleatorias.

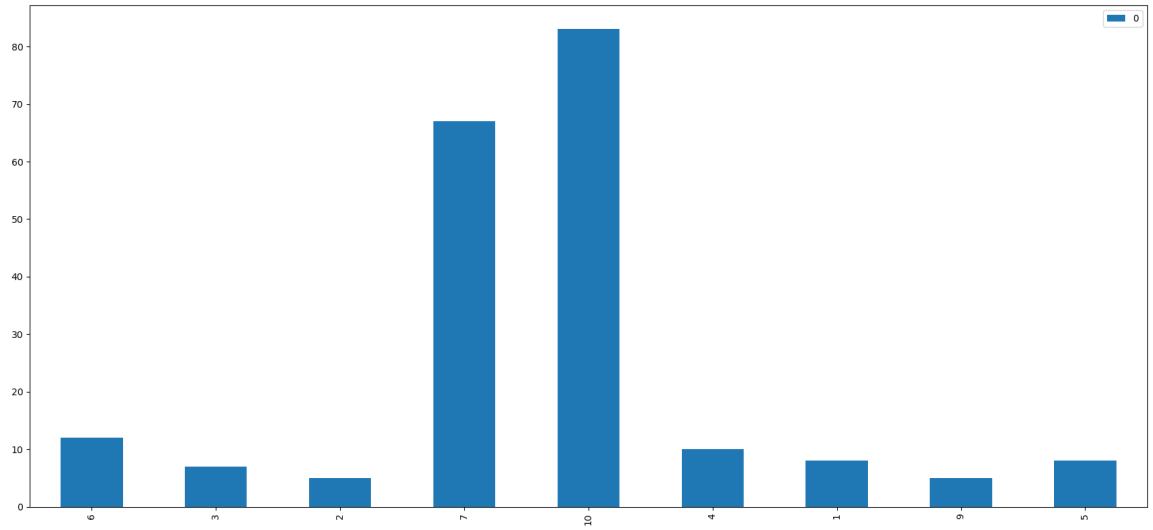
RESULTADOS

LCG

Corrida con 1000 cadenas

```
>>> %Run lab4.py

  [0.02628330560039922, True]
  [0.17349378869243082, True]
  [0.1611932512168122, True]
  [0.682855154252068, True]
  [0.6764125675463646, True]
  [0.958259417089796, True]
  [0.6202533789591361, True]
  [0.9998847052703788, True]
  [0.15869637663197816, True]
  [0.03016703402894871, True]
```

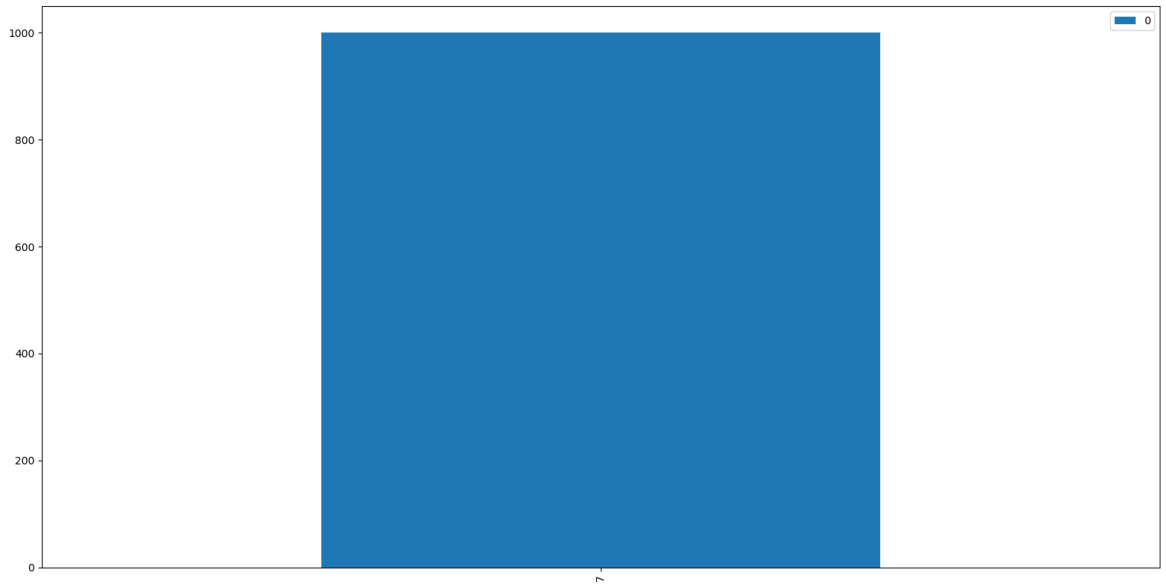


Se obtuvo la tabla de resumen aplicada a una cadena aleatoria utilizando el generador LCG y como se puede observar paso los 10 tests por lo que si se puede considerar como cadena aleatoria. En cuanto al histograma se observa que solo la prueba 8 no dio ningún error y todas las demás si, por lo que tomando en cuenta esto podríamos decir que no es muy confiable este generador. Sin embargo, observando en el histograma las frecuencias de error de cada prueba podemos notar que son bajas, llegando a ser las más altas la prueba 7 con 67 y la prueba 10 con 83 respectivamente, por lo que podría llegar a considerarse como un generador confiable.

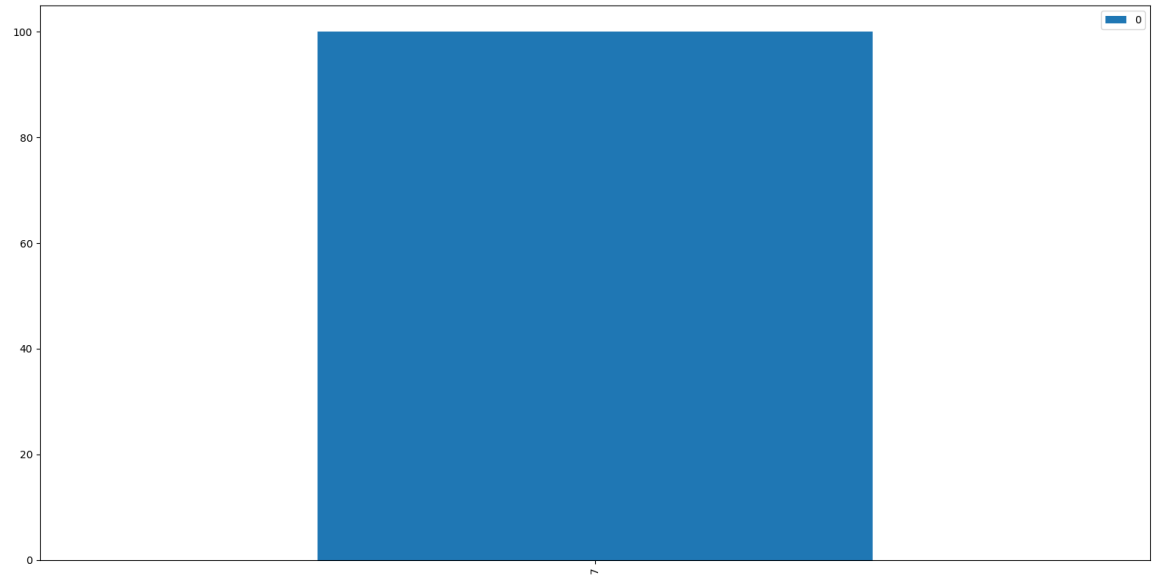
Wichman Hill

```
----------Wichman Hill----------
1001010011000101101101100000100101001111110000101110110101111010101100 …
[0.20839175624837297, True]
[0.36196268857315217, True]
[0.3263047752049874, True]
[0.9404069286829482, True]
[0.0366617836017767, True]
[0.3213326313166891, True]
[0.2639541060342644, True]
[0.9998070441954143, True]
[0.8814437887947022, True]
[0.723564352097616, True]
```

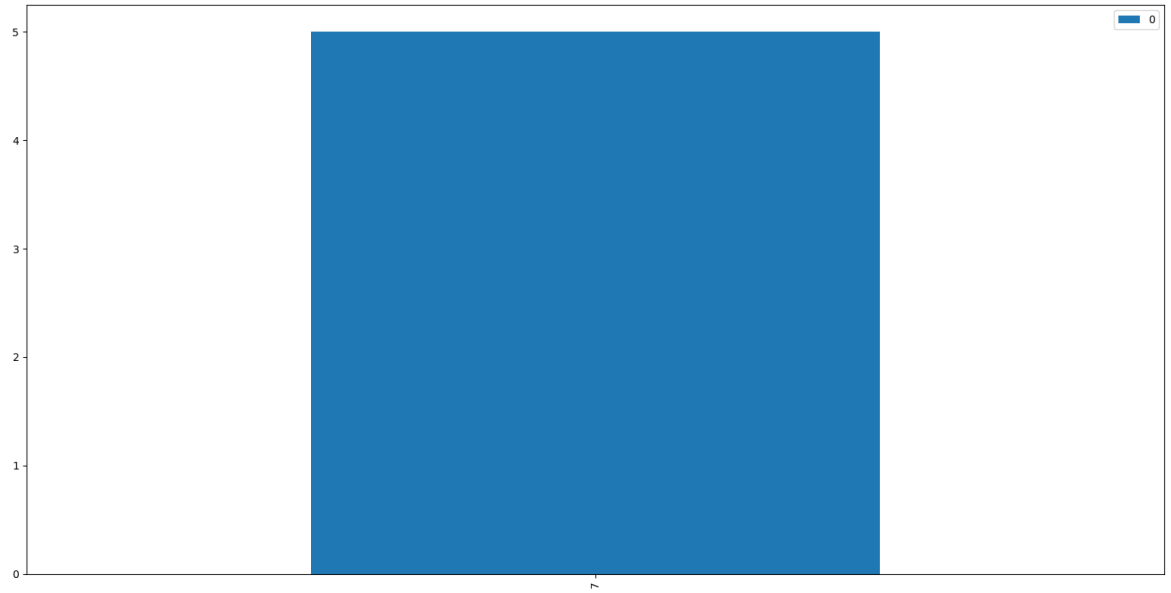
Corrida con 1000 cadenas



Corrida con 100 cadenas



Corrida con 5 cadenas



Se obtuvo la tabla de resumen aplicada a una cadena aleatoria utilizando el generador Wichman-Hill y como se puede observar paso los 10 tests por lo que sí se puede considerar como cadena aleatoria. En cuanto al histograma se observa que tiene un patrón muy extraño, ya que de los 10 tests que se aplicaron sólo el número 7 da error, pero da un error del 100% . Al notar este comportamiento se probó con menos cadenas (de 100 y 5) y el resultado fue el mismo, 100% de error en la prueba 7. Sin embargo al no existir ningún error en las otras pruebas es posible que este fallo se deba a alguna mala práctica en la implementación, por lo que se considera como confiable este generador.

LFSR

Tablas de resumen

Cadena Pequeña

```
[0.8509806870320558, True]
[6.593780318779072e-17, False]
[2.532848240735858e-07, False]
[0.3052349628786285, True]
  Number of blocks must be greater than 37
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Error. Need at least 10^6 input
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0.6240645324651697, False]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0.0004971363870032016, False]
[0.026874059769912282, True]
  Number of blocks must be greater than 37
Error. Need at least 10^6 input
```

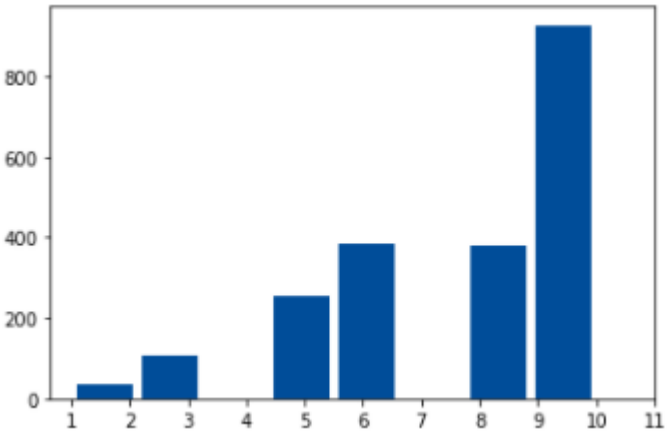
Cadena Mediana

```
[0.1037416782365416, True]
[1.690527951098661e-08, False]
[0.0, False]
[0.3052349628786285, True]
[0.0001469416388451505, False]
Error. Need at least 10^6 input
[0, 0, 0, 0, 0, 0, 0, 0, 0]
[0.8949687316194994, True]
[0, 0, 0, 0, 0, 0, 0, 0]
[0.0, False]
[0.3658646460136505, True]
Error. Need at least 10^6 input
```

Cadena Larga

```
[0.059020195330246134, True]
[1.0, True]
[0.0, False]
[0.3052349628786285, True]
[6.917189050140267e-07, False]
Error. Need at least 10^6 input
[0, 0, 0, 0, 0, 0, 0, 0, 0]
[0.8949687316194994, True]
[0, 0, 0, 0, 0, 0, 0, 0]
[0.0, False]
[0.3658646460136505, True]
Error. Need at least 10^6 input
```

Histograma de Frecuencias



Con las mil cadenas aleatorias se obtuvieron resultados qué nos indica qué el generador pseudoaleatorio no es muy eficiente con todos los tests ya qué de los diez qué se implementaron seis dieron error siendo el décimo con una mayor frecuencia. Sin embargo se realizaron pruebas de los test en donde se utilizaron cadenas de utilizaron 2000000 bits y todos dieron resultados positivos, por lo cual no se tiene un histograma de frecuencias pero si la tabla de resumen.

```
[0.7162666018385737, True]
[0.6852145037037066, True]
[0.6376217718716537, True]
[0.043884241248943814, True]
[0.30745847269066523, True]
[0.9600343318212248, True]
[0.4053059033546922, True]
[0.9998425124232174, True]
[0.692450822212496, True]
[0.7161130247629203, True]
```

CONCLUSIONES

1. El generador pseudoaleatorio LCG presenta errores en casi todos los tests pero en cada test son muy pocos los errores por lo que es posible decir que es un generador confiable.
2. El generador pseudoaleatorio Wichman Hill presenta un 100% de error en la prueba 7 y un 100% de acierto en las otras 9 pruebas por lo que es posible decir que es un generador confiable.
3. El generador pseudoaleatorio LFSR implementado es más eficiente cuando se trabaja con cadenas muy largas de bits y al igual qué los resultados obtenidos cifrando las imágenes el en Laboratorio 03, podemos observar qué mientras más larga sea la cadena, mejor son los resultados de los tests.

LINK DEL REPOSITORIO

https://github.com/JosePabloPonce/lab4