TRABAJO PRÁCTICO N°2.1 - GENERADORES PSEUDOALEATORIOS

Gnavi Faustino
Universidad Tecnológica Nacional
Ingeniería en Sistemas
Legajo 49417
faustinognavi@gmail.com

Mercé Alexis
Universidad Tecnológica Nacional
Ingeniería en Sistemas
Legajo 50174
alexis.am.2001@gmail.com

Pieroni Tiago
Universidad Tecnológica Nacional
Ingeniería en Sistemas
Legajo 49487
tepieroni@gmail.com

Roca Felipe
Universidad Tecnológica Nacional
Ingeniería en Sistemas
Legajo 49850
feliroca01@gmail.com

Sebben Mateo
Universidad Tecnológica Nacional
Ingeniería en Sistemas
Legajo 49609
mateosebben114@gmail.com

Tosello Mateo
Universidad Tecnológica Nacional
Ingeniería en Sistemas
Legajo 49475
tosellomateo0@gmail.com

13 de junio de 2024

RESUMEN

Este informe explora la implementación y evaluación generadores pseudoaleatorios. Utilizando el lenguaje de programación Python, se desarrollarán generadores y se generarán muestras de números pseudoaleatorios. La calidad de estos números se evaluará mediante diversas pruebas estadísticas. Los resultados de estas pruebas proporcionan una visión integral sobre la eficiencia y fiabilidad de cada generador, ofreciendo una base sólida para su aplicación en distintas áreas que requieren números aleatorios de alta calidad.

1. Introducción

Si decidiésemos realizar el sorteo de Navidad de Lotería Nacional mediante un ordenador, seguramente la gente no confiaría en la aleatoriedad del ordenador y se quejaría. En su lugar, se prefiere un método físico y sencillo de entender, como extraer bolas de un bombo. Incluso este tipo de métodos requiere tomar ciertas precauciones: todas las bolas debe tener idéntico peso, deben de estar bien mezcladas en el bombo y se deben cambiar regularmente para reducir las posibilidades de que unas aparezcan más que otras. Claramente este procedimiento no es práctico para una simulación computacional que requiere la generación de cientos de miles de números aleatorios. El método más conveniente y más fiable de generar números aleatorios es utilizar algoritmos deterministas que posean alguna base matemática sólida. Estos algoritmos producen una sucesión de números que se asemeja a la de una sucesión de realizaciones de variables aleatorias U(0, 1), aunque realmente no lo sea. Es por ello que este tipo de números se denominan pseudo-aleatorios y el algoritmo que los produce se llama generador de números pseudo-aleatorios.

Los generadores de números pseudoaleatorios (PRNG, por sus siglas en inglés) son herramientas esenciales en diversas disciplinas científicas y tecnológicas. En el campo de la simulación, los números pseudoaleatorios son fundamentales

para modelar la incertidumbre y variabilidad de sistemas complejos, como los tiempos de espera en colas o las fluctuaciones en los mercados financieros.

Este informe se enfoca en la implementación y evaluación de dos métodos clásicos de generación de números pseudoaleatorios: el Generador Congruencial Lineal (GCL) y el Generador de Cuadrados Medios. El GCL es conocido por su simplicidad y eficiencia, utilizando una relación recursiva con operaciones aritméticas modulares. En contraste, el Generador de Cuadrados Medios, uno de los métodos más antiguos, basa su funcionamiento en la extracción de dígitos centrales del cuadrado de la semilla inicial.

Para asegurar la calidad de los números generados, es fundamental someterlos a rigurosas pruebas estadísticas. En este trabajo, aplicamos la prueba de Chi Cuadrado, la prueba de Paridad, la prueba de Póker y la prueba de Kolmogorov-Smirnov. Estas pruebas permiten evaluar si las secuencias producidas exhiben propiedades similares a las de una distribución verdaderamente aleatoria.

El propósito de este informe es presentar la implementación de estos generadores en Python y analizar su rendimiento a través de pruebas estadísticas. De este modo, proporcionamos una evaluación comparativa que destaca las fortalezas y limitaciones de cada generador en el contexto de aplicaciones de simulación.

2. Descripción del Trabajo

El trabajo de investigación consiste en construir programas en lenguaje Python 3.x que generen números pseudoaleatorios y que estos se comporten como se espera. Para esto se debe tener en cuenta lo siguientes temas a investigar:

- Generadores de números aleatorios reales.
- Generadores de números pseudoaleatorios (Método de los cuadrados, GCLs, otros).
- Test para determinar el comportamiento de los generadores.

Se pide programar al menos dos generadores de números pseudoaleatorios en particular el generador GCL del cual se debe testear con al menos cuatro pruebas para determinar la calidad de generación. También se pide comparar los generadores programados con otros (incluyendo el que posee el lenguaje Python).

3. Desarrollo

Cuando hablamos de números generados aleatorios debemos tener en cuenta sus orígenes y sus usos. Existen dos tipos: Aleatorios (TRNG) y Pseudoaleatorios (PRNG).

Los TRNG extraen la aleatoriedad de los fenómenos físicos y la introducen en una computadora. Por su parte, un número pseudoaleatorio es un número generado en un proceso que parece producir números al azar, pero no lo hace realmente.

Las secuencias de números pseudoaleatorios no muestran ningún patrón o regularidad aparente desde un punto de vista estadístico a pesar de haber sido generadas por un algoritmo completamente determinista. La diferencia básica entre PRNG y TRNG es fácil de entender si se comparan números aleatorios generados por computadora con tiradas de un dado. Debido a que los PRNG generan números aleatorios mediante el uso de fórmulas matemáticas o listas precalculadas, el uso de uno corresponde a alguien que tira un dado muchas veces y anota los resultados. Cada vez que pides una tirada de dados, obtienes el siguiente en la lista. Efectivamente, los números parecen aleatorios, pero están realmente predeterminados. Los TRNG funcionan haciendo que una computadora realmente tire el dado o, más comúnmente, usen algún otro fenómeno físico que sea más fácil de conectar a una computadora que un dado.

Los PRNG son eficientes, lo que significa que pueden producir muchos números en poco tiempo, y deterministas, es decir, que una secuencia de números dada puede reproducirse en una fecha posterior si se conoce el punto de partida de la secuencia.

La eficiencia es una buena característica si tu aplicación necesita muchos números, y el determinismo es útil si necesitas volver a reproducir la misma secuencia de números en una etapa posterior. Los PRNG también suelen ser periódicos, esto significa que la secuencia eventualmente se repetirá. Si bien la periodicidad casi nunca es una característica deseable, los PRNG modernos tienen un período que es tan largo que puede ignorarse para la mayoría de los propósitos prácticos.

Estas características hacen que los PRNG sean adecuados para aplicaciones donde se requieren muchos números y donde es útil que la misma secuencia se pueda reproducir fácilmente. Ejemplos populares de tales aplicaciones son las aplicaciones de simulación y modelado. Los PRNG no son adecuados para aplicaciones en las que es importante que los números sean realmente impredecibles, como el cifrado de datos y los juegos de azar.

3.1. Generador Congruencial Lineal (GLC)

3.1.1. Descripción del método

El Método de Congruencias Lineales (GLC) es uno de los algoritmos más simples y utilizados para la generación de números pseudoaleatorios. Este método se basa en una fórmula recursiva que produce una secuencia de números a partir de un valor inicial, conocido como semilla.

El generador se define a partir de la siguiente relación de recurrencia:

$$X_{n+1} = (aX_n + c) \mod m$$

donde X es la secuencia de valores pseudoaleatorios, y

- **m**: $0 < m$, es el módulo,
- **a**: $0 < a < m$, es el multiplicador,
- **c**: $0 \leq c < m$, es el incremento, y
- X_0: $0 \leq X_0 < m$, es la semilla.

3.1.2. Parámetros del Método GLC

- **Semilla** (X_0): Es el valor inicial a partir del cual se genera la secuencia. Elegir una semilla adecuada es crucial para obtener una secuencia de buena calidad.
- **Multiplicador** (a): Este valor afecta cómo los números de la secuencia son distribuidos. Debe ser elegido cuidadosamente para asegurar una buena dispersión de valores.
- **Incremento** (c): Este valor añade variabilidad adicional a la secuencia. En algunos casos, se puede elegir $c = 0$, en cuyo caso el método se llama Multiplicativo.
- **Módulo** (m): Este valor define el rango de la secuencia de números pseudoaleatorios. Normalmente se elige grande, como una potencia de 2, para cubrir un rango amplio de valores posibles.

3.1.3. Propiedades Deseadas

Para que el generador de números pseudoaleatorios (PRNG) tenga buenas propiedades estadísticas, los valores de a, c, y m deben ser seleccionados cuidadosamente. Algunas propiedades deseadas incluyen:

- **Periodo Máximo**: La secuencia debe tener el periodo más largo posible, m, lo que significa que la secuencia de números no se repetirá hasta que se generen m números.

- **Uniformidad**: Los números generados deben ser distribuidos uniformemente a lo largo del rango permitido.

- **Independencia**: Los números en la secuencia deben ser estadísticamente independientes.

3.1.4. Análisis de los resultados obtenidos

Para la ejecución del programa se consideraron los siguientes parámetros:

- **Semilla** (X_0): Se utilizaron cuatro semillas distintas para evaluar el método. Primeramente se utilizó la función ***time*** de Python para poder generar una semilla aleatoria a partir del tiempo actual del sistema. Luego se eligieron aleatoriamente tres números, aumentando la longitud de los mismos.

  Luego se determinaron los parámetros, siendo adecuados los siguientes:

- **Multiplicador** (a): 1664525

- **Incremento** (c): 1013904223

- **Módulo** (m): 2^{32} = 4294967296

Para evaluar la calidad de un generador de números aleatorios mediante pruebas estadísticas, suele ser suficiente generar entre 10^5 y 10^6 números. Hemos elegido una serie de 200.000 números generados.

A partir de la semilla y los parámetros especificados se obtuvieron las siguientes gráficas luego de ejecutar el programa:

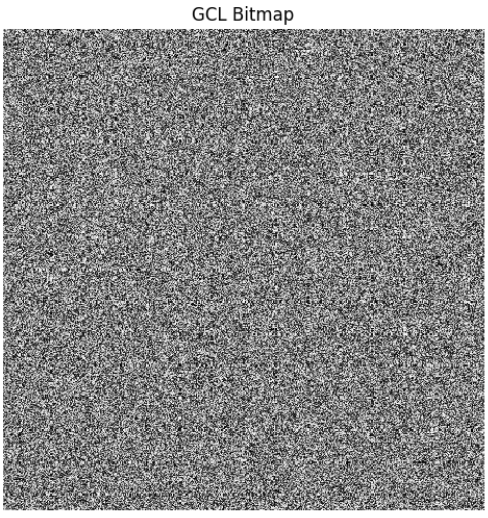

Figura 1: Semilla: 1718104268

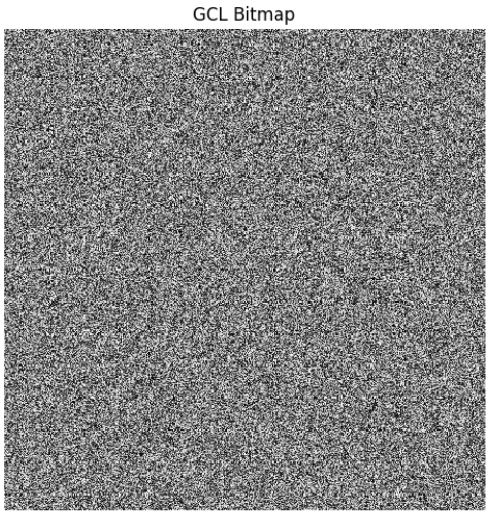

Figura 2: Semilla: 759302

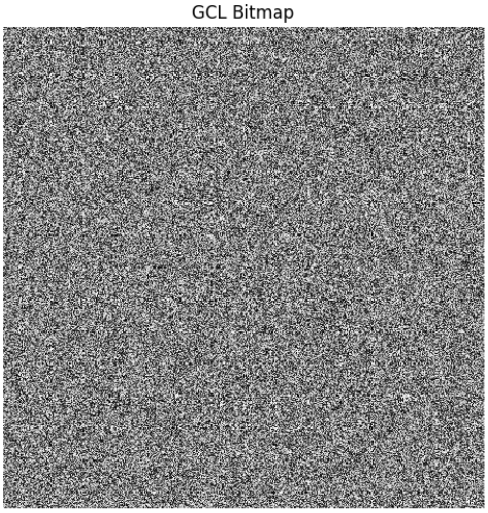

Figura 3: Semilla: 3183856186

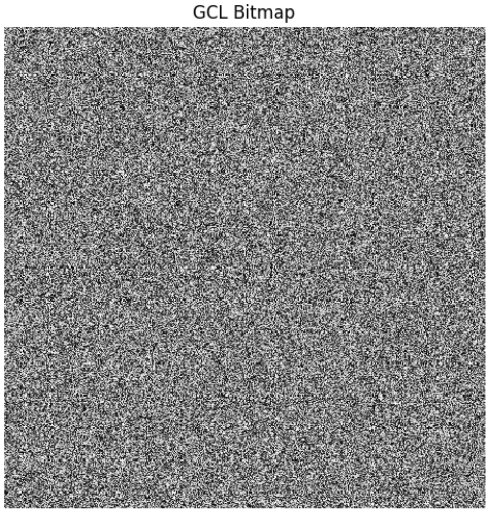

Figura 4: Semilla: 796225369226628988

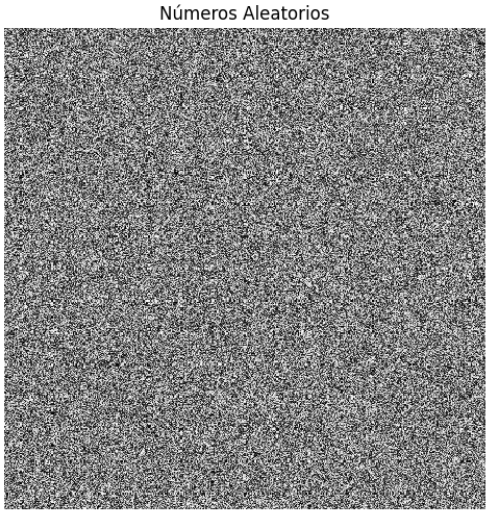

Figura 5: Secuencia de números aleatorios mediante Random (Python)

3.2. MÉTODO DE LOS CUADRADOS MEDIOS

3.2.1. Descripción del método

El método de los cuadrados medios se basa en una operación matemática sencilla: el cuadrado de un número. A partir de un número semilla, el método sigue los siguientes pasos:

- **Elección de la semilla**: Seleccionar un número inicial X_0.
- **Cuadrado del número**: Elevar al cuadrado el número actual para obtener X_n^2.
- **Extracción de dígitos centrales**: Tomar un número fijo de dígitos centrales del resultado para formar el siguiente número en la secuencia X_{n+1}.
- **Repetición**: Utilizar X_{n+1} como el siguiente número en la secuencia y repetir los pasos.

3.2.2. Propiedades y Limitaciones

Propiedades:

- **Simplicidad**: Es muy fácil de implementar y no requiere de operaciones complejas.
- **Determinismo**: La secuencia de números es completamente determinada por la semilla inicial.

Limitaciones:

- **Ciclo corto**: Las secuencias tienden a entrar en ciclos repetitivos rápidamente, especialmente si los números se vuelven pequeños o contienen muchos ceros.
- **Sensibilidad a la semilla**: La calidad de la secuencia generada depende en gran medida de la elección de la semilla inicial.
- **Poca aleatoriedad**: No proporciona la calidad de aleatoriedad necesaria para aplicaciones modernas.

3.2.3. Aplicaciones

Debido a sus limitaciones, el método de los cuadrados medios se utiliza más como una curiosidad histórica o para fines educativos que en aplicaciones prácticas. Hoy en día, los métodos modernos como los generadores congruenciales lineales y otros algoritmos avanzados ofrecen una mejor calidad de aleatoriedad y ciclos más largos.

3.2.4. Análisis de los resultados obtenidos

Para la ejecución del programa se utilizaron cuatro semillas distintas para evaluar el método. Primeramente se utilizó la función ***time*** de Python para poder generar una semilla aleatoria a partir del tiempo actual del sistema. Luego se eligieron aleatoriamente tres números, aumentando la longitud de los mismos.

Para evaluar la calidad de un generador de números aleatorios mediante pruebas estadísticas, suele ser suficiente generar entre 10^5 y 10^6 números. Hemos elegido una serie de 200.000 números generados.

A partir de la semilla se obtuvieron las siguientes gráficas al ejecutar el programa:

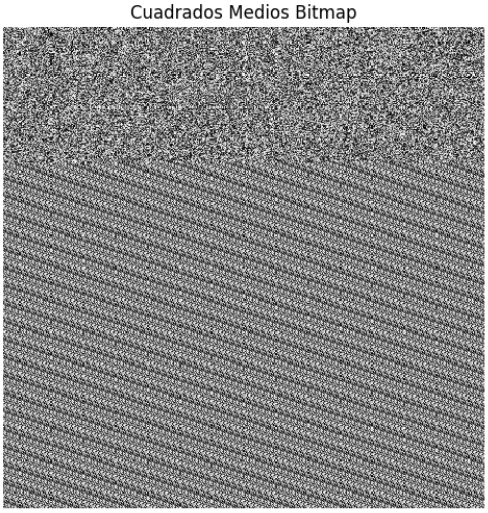

Figura 6: Semilla: 1718104268

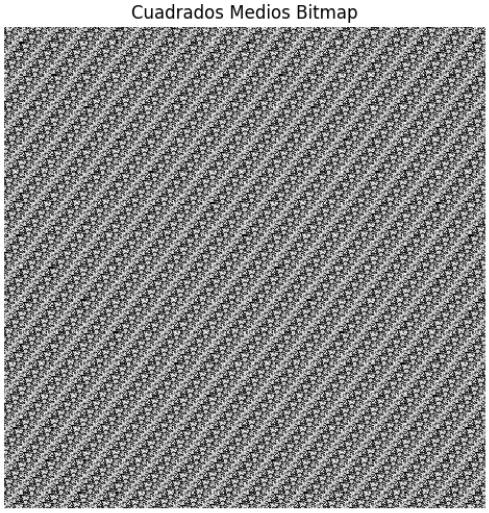

Figura 7: Semilla: 759302

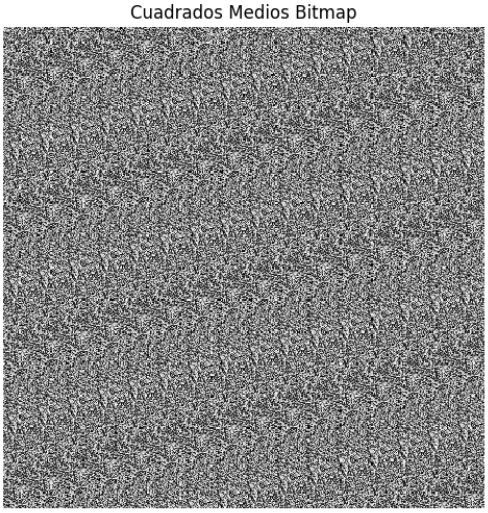

Figura 8: Semilla: 3183856186

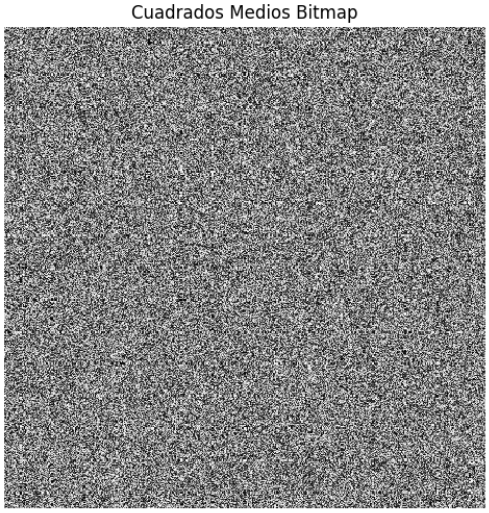

Figura 9: Semilla: 796225369226628988

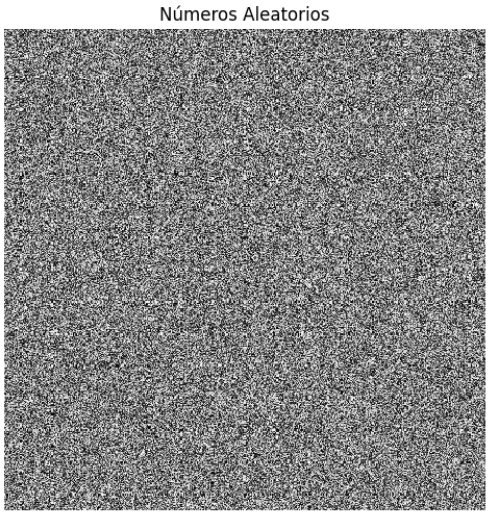

Figura 10: Secuencia de números aleatorios mediante Random (Python)

4. Tests

4.1. Prueba de Chi-Cuadrado

La prueba de Chi-Cuadrado es una técnica estadística que se utiliza para determinar si existe una diferencia significativa entre la distribución observada de una variable categórica y una distribución esperada o teórica. En el contexto de pruebas de aleatoriedad, se emplea para evaluar si una secuencia de números aparentemente aleatorios sigue una distribución uniforme o alguna otra distribución específica.

4.1.1. Procedimiento

1. División en Categorías: Los números generados se agrupan en un número predeterminado de categorías o bins. Por ejemplo, si se generan números aleatorios entre 0 y 1, pueden dividirse en 10 categorías de ancho 0.1 cada una.
2. Conteo de Observaciones: Se cuenta cuántos números caen en cada categoría.
3. Cálculo del Estadístico Chi-Cuadrado: El estadístico de prueba Chi-Cuadrado se calcula utilizando la fórmula:

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

   Donde O_i es el número observado en la i-ésima categoría y E_i es el número esperado en la i-ésima categoría.
4. Determinación del p-valor: Se calcula el p-valor para el estadístico Chi-cuadrado usando la distribución Chi-cuadrado con $k - 1$ grados de libertad, donde k es el número de categorías. Este p-valor indica la probabilidad de obtener un valor del estadístico de prueba al menos tan extremo como el observado, si la hipótesis nula es verdadera. Se compara el p-valor con un nivel de significancia predefinido (α).

4.1.2. Interpretación

- **Hipótesis Nula (H0)**: La hipótesis nula asume que los números generados siguen la distribución especificada (por ejemplo, una distribución uniforme).

- **Decisión**:
  - Si el p-valor es mayor que el nivel de significancia ($p - valor > \alpha$), no se rechaza la hipótesis nula. Esto sugiere que no hay evidencia suficiente para afirmar que los números no siguen la distribución especificada.
  - Si el p-valor es menor que el nivel de significancia ($p - valor < \alpha$), se rechaza la hipótesis nula. Esto sugiere que hay suficiente evidencia para afirmar que los números no siguen la distribución especificada.

4.2. Prueba de Paridad (Bit Frequency Test)

La prueba de paridad, también conocida como Bit Frequency Test, es una técnica utilizada para evaluar la aleatoriedad de una secuencia de bits en números generados aleatoriamente. Este test examina si la frecuencia de unos y ceros en la secuencia es aproximadamente la misma, lo que se espera en una secuencia verdaderamente aleatoria.

4.2.1. Procedimiento

1. Conversión a Bits: Los números aleatorios se convierten en una secuencia de bits.
2. Conteo de Unos y Ceros: Se cuentan los bits de valor 1 y 0.
3. Cálculo de la Diferencia de Paridad:
$$p = \frac{|N_{\text{unos}} - N_{\text{ceros}}|}{N_{\text{total}}}$$
Donde N_{unos} y N_{ceros} son el número de bits de 1 y 0 respectivamente, y N_{total} es el número total de bits.

4.2.2. Interpretación

- $p > \alpha$: No se rechaza la hipótesis nula (la frecuencia de unos y ceros es aproximadamente igual).
- $p < \alpha$: Se rechaza la hipótesis nula (la frecuencia de unos y ceros no es igual).

4.3. Prueba de Póker (Poker Test)

La prueba de Póker, también conocida como Poker Test, es una técnica utilizada para evaluar la aleatoriedad de una secuencia de números generados aleatoriamente mediante el análisis de patrones específicos en la secuencia.

4.3.1. Procedimiento

1. Agrupación de Números: Los números se agrupan en grupos de m dígitos.
2. Clasificación de Patrones: Se clasifican los grupos de dígitos en patrones, como un par, dos pares, trío, etc.
3. Conteo de Frecuencias: Se cuenta la frecuencia de cada patrón.
4. Cálculo del Estadístico Chi-Cuadrado: Similar a la prueba de Chi-cuadrado, se compara la frecuencia observada de cada patrón con la frecuencia esperada.

4.3.2. Interpretación

- $p - valor > \alpha$: No se rechaza la hipótesis nula (los patrones se distribuyen como se esperaría por azar).
- $p - valor < \alpha$: Se rechaza la hipótesis nula (los patrones no se distribuyen como se esperaría por azar).

4.4. Prueba de Kolmogorov-Smirnov (K-S Test)

La prueba de Kolmogorov-Smirnov compara la distribución empírica de los números generados con una distribución teórica, generalmente la distribución uniforme.

4.4.1. Procedimiento

1. **Ordenamiento de los Datos**: Se ordenan los números generados aleatoriamente en orden ascendente.
2. **Cálculo de la Distribución Acumulada Empírica**: Se calcula la distribución acumulada empírica de los datos ordenados. Para un conjunto de n datos $X_1, X_2, \ldots, X_n$, la distribución acumulada empírica $F_n(x)$ se define como:
$$F_n(x) = \frac{\text{número de datos } \leq x}{n}$$
3. **Comparación con la Distribución Teórica**: Se compara la distribución acumulada empírica con la distribución acumulada teórica (por ejemplo, la distribución uniforme o normal) utilizando la distancia de Kolmogorov-Smirnov.
4. **Cálculo del Estadístico de Prueba**: Se calcula el estadístico de prueba de Kolmogorov-Smirnov, que es la máxima diferencia absoluta entre las distribuciones acumuladas empírica y teórica:
$$D_n = \text{máx}(|F_n(x) - F(x)|)$$
donde $F(x)$ es la función de distribución acumulada teórica.

4.4.2. Interpretación

- Si el valor del estadístico de prueba es menor que el valor crítico asociado a un nivel de significancia predefinido, no se rechaza la hipótesis nula. Esto sugiere que no hay evidencia suficiente para afirmar que los datos no provienen de la distribución especificada.
- Si el valor del estadístico de prueba es mayor que el valor crítico asociado a un nivel de significancia predefinido, se rechaza la hipótesis nula. Esto indica que hay suficiente evidencia para afirmar que los datos no provienen de la distribución especificada.

5. Pruebas realizadas

En cada ejecución del programa con los distintos métodos generadores de números pseudoaleatorios y las distintas semillas empleadas, se realizaron los 4 tests detallados anteriormente.

El valor de α se utiliza en pruebas de hipótesis estadísticas como el nivel de significancia. Representa la probabilidad de rechazar la hipótesis nula cuando en realidad es verdadera (error de tipo I). En otras palabras, α es el umbral para decidir si un resultado es estadísticamente significativo.

El valor de α debe ser elegido antes de realizar cualquier prueba estadística. Para este trabajo, se utilizó α = 0.05 ya que este es el nivel de significancia más común, que indica un 5 % de probabilidad de cometer un error de tipo I. Este valor es ampliamente aceptado en muchas disciplinas.

Si el p-valor obtenido de la prueba es menor que 0.05, se rechaza la hipótesis nula. Esto significa que existe menos del 5 % de probabilidad de que la diferencia observada sea debida al azar.

En el contexto de pruebas de aleatoriedad (como la prueba de Kolmogorov-Smirnov, la prueba de Chi-Cuadrado, etc.), el valor de α se utiliza para determinar si la secuencia de números generados pasa o no la prueba.

Por ejemplo:

- Prueba de Chi-Cuadrado: Se compara el estadístico de la prueba con el valor crítico de la distribución χ^2 para un nivel de significancia α. Si el estadístico es mayor que el valor crítico, se rechaza la hipótesis nula de que los números siguen la distribución esperada.

- Prueba de Kolmogorov-Smirnov: Se compara el estadístico D con el valor crítico de la distribución de Kolmogorov-Smirnov para α. Si D es mayor que el valor crítico, se rechaza la hipótesis nula de que los números siguen la distribución acumulada teórica.

Se obtuvieron los siguientes resultados al aplicar los distintos tipos de tests, a los métodos generadores y para las distintas semillas empleadas.

Cuadro 1: Resultados obtenidos al aplicar tests

Generador	Semilla	Test χ^2	Test Paridad	Test Poker	Test Kolmogorov-Smirnov
GCL	1718104268 (time)	Aprueba	Rechaza	Rechaza	Aprueba
GCL	759302	Aprueba	Rechaza	Rechaza	Aprueba
GCL	3183856186	Aprueba	Rechaza	Rechaza	Aprueba
GCL	796225369226628988	Aprueba	Rechaza	Rechaza	Aprueba
Cuadrados Medios	1718104268 (time)	Rechaza	Rechaza	Rechaza	Rechaza
Cuadrados Medios	759302	Rechaza	Rechaza	Rechaza	Rechaza
Cuadrados Medios	3183856186	Rechaza	Rechaza	Rechaza	Rechaza
Cuadrados Medios	796225369226628988	Aprueba	Rechaza	Rechaza	Aprueba

6. Conclusiones

A partir de las gráficas obtenidas y los resultados de los test realizados, podemos concluir los siguientes puntos:

6.1. Generador Congruencial Lineal (GLC)

- A simple vista, no se detectan patrones evidentes en las secuencias de números generadas utilizando el método GCL con ninguna de las cuatro semillas. La aparente aleatoriedad de las secuencias puede indicar una buena distribución y dispersión de los números generados, lo que dificulta la identificación de patrones visuales o estructuras repetitivas.
- A pesar de la falta de patrones visuales discernibles, los resultados de los tests de aleatoriedad proporcionan una evaluación objetiva de la aleatoriedad de las secuencias generadas, mostrando una variabilidad en la capacidad de dichas secuencias para pasar los criterios de aleatoriedad establecidos por los tests. Se observa una combinación de resultados de tests que van desde la aprobación hasta el rechazo para diferentes semillas, lo que nos indica que la aleatoriedad de las secuencias puede variar dependiendo de la semilla utilizada.
- La elección de una buena semilla y la configuración cuidadosa de los parámetros del GCL son aspectos críticos en la generación de secuencias aleatorias de alta calidad. Estos elementos deben ser considerados con atención para asegurar la fiabilidad y la aleatoriedad de los resultados obtenidos.

En resumen, aunque el método GCL puede generar secuencias que parecen aleatorias a simple vista, es necesario realizar pruebas estadísticas para confirmar su verdadera aleatoriedad y su adecuación para su uso en aplicaciones donde se requiere aleatoriedad confiable.

6.2. Método de los Cuadrados Medios

- Para la primer semilla (generada aleatoriamente a partir del tiempo actual del sistema) se observa un comportamiento particular en la secuencia generada, caracterizado por un rango inicial de aleatoriedad seguido por la aparición de un patrón específico. A pesar del inicio aleatorio, la presencia del patrón detectado lleva a la

no aleatoriedad de la secuencia. Todos los tests de aleatoriedad rechazan la hipótesis de aleatoriedad, lo que confirma la presencia de dicho patrón y la falta de uniformidad en la distribución de los números generados.

- Para las semillas 759302 y 3183856186 se identifican patrones visibles en las secuencias producidas desde el inicio de la generación. Estos patrones son evidentes a simple vista y corroborados por los resultados de los tests de aleatoriedad, que rechazan la hipótesis de aleatoriedad en todas las pruebas. La presencia de estos patrones indica una falta de uniformidad en la distribución de los números generados, lo que afecta la calidad y fiabilidad del método de los cuadrados medios para estas semillas específicas.
- En cuanto a la última semilla (la de mayor longitud), a diferencia de las anteriores, esta semilla no muestra patrones evidentes a simple vista en la secuencia generada. Sin embargo, los resultados de los tests de aleatoriedad son mixtos, con algunas pruebas que sugieren aleatoriedad y otras que rechazan esta hipótesis, lo que indica posibles limitaciones en la uniformidad de la distribución de los números generados.

Estas conclusiones subrayan la importancia de elegir cuidadosamente las semillas iniciales en el método de los cuadrados medios y resaltan la necesidad de realizar pruebas rigurosas de aleatoriedad para evaluar la calidad y fiabilidad de las secuencias generadas. Los resultados obtenidos indican que la presencia de patrones visibles o la falta de aleatoriedad pueden comprometer la utilidad del método de los cuadrados medios en la generación de números aleatorios para aplicaciones críticas.

6.3. Conclusión general

Mientras que el Generador Congruencial Lineal tiende a producir secuencias más aleatorias y uniformes, el método de los Cuadrados Medios muestra limitaciones en cuanto a la generación de secuencias verdaderamente aleatorias y uniformes, especialmente cuando se usan semillas inadecuadas. Esto destaca la importancia de evaluar cuidadosamente los métodos de generación de números aleatorios y seleccionar el más apropiado según los requisitos específicos de la aplicación.

La biblioteca random de Python produce secuencias que parecen más aleatorias y no muestran patrones visibles en las gráficas de mapa de bits. Aunque los resultados de los tests de aleatoriedad pueden variar, en general, la biblioteca random produce secuencias que son más uniformes y aleatorias en comparación con el GCL y los cuadrados medios.

La comparación entre el GCL, los cuadrados medios y la biblioteca random de Python, resalta las limitaciones de los métodos de generación de números pseudoaleatorios y destaca la importancia de utilizar bibliotecas probadas y bien establecidas para obtener secuencias verdaderamente aleatorias.

Además de comparar los métodos de generación de números pseudoaleatorios, es importante tener en cuenta que, a diferencia de los métodos pseudoaleatorios, los métodos verdaderamente aleatorios se basan en fuentes de aleatoriedad física o natural, como el ruido atmosférico, los fenómenos cuánticos o el movimiento caótico. Estos métodos proporcionan una fuente de aleatoriedad más confiable y pueden ser preferibles en aplicaciones donde la seguridad y la imprevisibilidad son críticas, como la criptografía, los juegos de azar y la generación de claves criptográficas.

Aunque los métodos verdaderamente aleatorios pueden ser más costosos y difíciles de implementar en comparación con los métodos pseudoaleatorios, ofrecen una garantía más sólida de aleatoriedad y son menos susceptibles a patrones o sesgos. Por lo tanto, en aplicaciones donde la calidad y la seguridad son primordiales, los métodos verdaderamente aleatorios pueden ser la mejor opción.

En resumen, mientras que los métodos pseudoaleatorios son útiles para una amplia gama de aplicaciones donde se requiere una aleatoriedad aceptable pero no perfecta, los métodos verdaderamente aleatorios son preferibles en situaciones donde la fiabilidad y la seguridad son fundamentales. Es crucial seleccionar el método de generación de números adecuado según los requisitos específicos de cada aplicación.

7. Bibliografía

- https://www.random.org/analysis/Analysis2005.pdf
- https://tereom.github.io/est-computacional-2018/numeros-pseudoaleatorios.html
- https://www.random.org/analysis/
- https://www.random.org/randomness/
- https://es.wikipedia.org/wiki/Generador_congruencial_lineal
- https://es.wikipedia.org/wiki/Generador_de_n%C3%BAmeros_pseudoaleatorios