Universidad del Istmo
Curso: Análisis de datos
Catedrático: Juan Andrés García Porres

Investigación e implementaciones

Tarea 2

Juan Pablo Estrada Lucero
Ingeniería en sistemas y ciencias de la computación
0000012782
Fecha de entrega: 31 / 08 / 2025

Primera parte

- **Generadores de congruencia lineal (LCG)**
  Algoritmo que permite obtener secuencia de números pseudoaleatorios calculados con una función lineal definida a trozos discontinua.

  - Fundamento matemático:
    La secuencia de números enteros X_1, X_2, X_3, … está definida por la relación de recurrencia:

    $$X_i = (aX_{i-1} + c) \bmod m$$

    con $0 \leq X_i \leq m - 1$ donde m (el módulo), a (el multiplicador), c (el incremento) y X_0 (la semilla o valor inicial) son números enteros no negativos.

    Además a los números m , a , c y X_0 se les impone las condiciones $m > 0$, $0 < a < m$, $0 \leq c < m$ y $0 \leq X_0 < m$.

    Su período es la longitud máxima de la secuencia antes de repetirse en m.

    Tendrá período completo m para cualquier valor de la semilla X_0 en { 0, 1, … , m − 1 } si y sólo si:
    1. c & m son coprimos, mcd(c,m) = 1
    2. a - 1 es múltiplo de todos los factores primos de m, es decir, a = 1 mod z para todo z que sea factor primo de m.
    3. Si m es múltiplo de 4, entonces a - 1 también lo es.

  - Pseudocódigo:

```
Función LCG(semilla, a, c, m, cantidad)
  // semilla: número inicial
  // a: multiplicador
  // c: incremento
  // m: módulo
  // cantidad: cuántos números aleatorios generar

  X = semilla  // Valor inicial

  Repetir cantidad veces:
    X = (a * X + c) mod m  // Fórmula principal
    Print(X)  // Imprimir el número pseudoaleatorio generado
  Fin Repetir
Fin Función
```

- **Método de cuadros medios o middle-square method**
  Método para generar números pseudoaleatorios.

  - Fundamento matemático:
    Cada número sucesivo se genera tomando, los “n” dígitos centrales del cuadrado del número anterior de n dígitos.

- Pseudocódigo:

```
Función CuadrosMedios(semilla, cantidad, n_digitos)
   // semilla: número inicial
  // cantidad: cantidad de números a generar
  // n_digitos: cantidad de dígitos que tendrá cada número generado

   X = semilla  // Valor inicial

   Repetir cantidad veces:
      cuadrado = X * X  // Elevar al cuadrado
       texto = convertir a cadena con ceros a la izquierda hasta tener el
doble de dígitos

      // Extraer los dígitos del centro
      inicio = posición desde donde extraer los dígitos del centro
      centro = extraer n_digitos desde la posición inicio

      X = convertir centro a número
      Print(X) // Imprimir número pseudoaleatorio generado
   Fin Repetir
Fin Función
```

- **Mersenne Twister**
  - Fundamento matemático:
  - Pseudocódigo:

- **Blum Blum Shub**

  Generador pseudoaleatorio de números.
  - Fundamento matemático:

    Este se compone de la siguiente manera:

    $$x_{n+1} = (x_n)^2 \bmod M$$

    - donde:

      M = p*q (donde p y q son dos números primos muy grandes)

    En cada paso del algoritmo se obtiene un resultado para x_n los dos números primos, p y q, deben ser ambos congruentes a 3 (mod 4)

  - Pseudocódigo:

```
Función BlumBlumShub(semilla, M, cantidad)
   // semilla: número inicial (debe ser coprimo con M)
      // M: número compuesto obtenido de multiplicar dos primos
grandes
   // cantidad: cuántos números pseudoaleatorios se quieren generar
```

```
    X = semilla

    Repetir cantidad veces:
      X = (X * X) mod M //Elevación al cuadrado

      Print(X)  // Imprimir número pseudoaleatorio generado
    Fin Repetir
  Fin Función
```

- **RANDU**
  - Fundamento matemático:
    Es un LCG, el cual se define a través de la siguiente recurrencia:

    $$V_{j+1} = 65539 * V_j \bmod 2^{31}$$

    Teniendo en consideración que V_0 es un número impar.

    La normalización del valor se obtiene mediante:

    $$X_j = V_j / 2^{31}$$

  - Pseudocódigo:

```
    Función RANDU(semilla, cantidad)
      // semilla: número inicial
      // cantidad: cuántos números pseudoaleatorios a generar
      a = 65539  // Multiplicador específico del RANDU
      m = 2^31

      X = semilla

      Repetir cantidad veces:
        X = (a * X) mod m  // Fórmula del generador
        Print(X) // Imprimir el número pseudoaleatorio generado
      Fin Repetir
    Fin Función
```

Tabla comparativa entre cada algoritmo:

Algoritmo	Velocidad	Periodo	Seguridad / Criptografía	Casos de uso típicos
LCG (Linear Congruential Generator)	Muy rápida	Moderado, depende de los parámetros (hasta ~2^{31}–1 para 32 bits)	Baja, predecible, no seguro criptográficamente	Simulaciones simples, juegos, prototipos, generación de datos no críticos
Método de cuadrados medios	Lenta, especialmente para números	Muy corto, suele degenerar rápidamente	Muy baja, altamente predecible y con ciclos cortos	Educación, experimentos históricos,

(Middle-square)	grandes			demostración de conceptos
Mersenne Twister	Muy rápida	Muy largo (~2^{19937}–1)	Media, no es seguro para criptografía	Simulaciones científicas, videojuegos, generación de números aleatorios de propósito general
Blum Blum Shub	Lenta, requiere operaciones de factorización	Muy largo (depende de los números primos grandes usados)	Muy alta, es seguro criptográficamente	Aplicaciones criptográficas, generación de claves, tokens de seguridad
RANDU	Rápida	Moderado (ciclos de ~$2^{31}/4$)	Muy baja, tiene correlaciones graves	Históricamente usado en mainframes IBM, hoy en día obsoleto

Bibliografía:

https://es.wikipedia.org/wiki/Generador_lineal_congruencial
https://www.youtube.com/watch?v=mXBGXU0zJnw
https://es.wikipedia.org/wiki/M%C3%A9todo_del_medio_del_cuadrado
https://simulacion2017.wordpress.com/2017/02/23/2-2-1-algoritmo-de-cuadrados-medios/
https://es.wikipedia.org/wiki/Blum_Blum_Shub
https://en.wikipedia.org/wiki/RANDU