Contents

Clase 1: Introducción a la Generación de Números Aleatorios

Objetivos de la clase:

- Comprender la importancia de los números aleatorios en la simulación y el modelado.
- Definir qué es un generador de números aleatorios (GNA) y sus características principales.
- Explorar el concepto de pseudoaleatoriedad y su relevancia práctica.
- Introducir el Generador Congruencial Lineal (GCL) como un método fundamental de generación.

Contenido Teórico Detallado:

1. **Introducción a la Aleatoriedad en la Simulación:**
   - La simulación de eventos y sistemas complejos a menudo requiere la introducción de aleatoriedad para modelar la variabilidad y la incertidumbre.
   - Los números aleatorios son esenciales para representar eventos probabilísticos, como tiempos de llegada, fallas, o resultados de experimentos.
   - Ejemplos de aplicaciones donde los números aleatorios son cruciales:
     - Simulación de colas (teoría de colas): Modelar el tiempo de espera de clientes en un sistema de servicio.
     - Simulaciones de Monte Carlo: Estimar valores numéricos mediante muestreo aleatorio.
     - Juegos de azar y apuestas: Simular el resultado de juegos con elementos de azar.
     - Criptografía: Generar claves seguras y datos aleatorios para cifrado.
2. **¿Qué es un Generador de Números Aleatorios (GNA)?**
   - Un GNA es un algoritmo que produce una secuencia de números que aparentan ser aleatorios.
   - Idealmente, los números generados deberían ser:
     - **Uniformemente distribuidos:** Cada número tiene la misma probabilidad de aparecer en un rango dado.
     - **Independientes:** Cada número es independiente de los números anteriores en la secuencia.
   - En la práctica, los GNA son deterministas; producen secuencias que, aunque parezcan aleatorias, están completamente determinadas por un valor inicial llamado "semilla".
   - Por lo tanto, se les conoce más precisamente como generadores de números *pseudoaleatorios*.
3. **Pseudoaleatoriedad:**
   - La pseudoaleatoriedad implica que la secuencia de números generada es determinista pero pasa las pruebas estadísticas de aleatoriedad.
   - Características importantes de un buen GNA pseudoaleatorio:
     - **Largo período:** La secuencia debe repetirse solo después de un número muy grande de iteraciones.
     - **Eficiencia:** El algoritmo debe ser computacionalmente rápido.
     - **Portabilidad:** El GNA debe producir resultados similares en diferentes plataformas.
     - **Capacidad de pasar pruebas estadísticas:** La secuencia generada debe aprobar pruebas de aleatoriedad (ej. prueba de Chi-cuadrado, prueba de Kolmogorov-Smirnov).
4. **Generador Congruencial Lineal (GCL):**
   - El GCL es uno de los GNA más antiguos y ampliamente utilizados.

- La fórmula general del GCL es:
  - $X_{i+1} = (aX_i + c) \bmod m$
  - Donde:
    - X_{i+1} es el siguiente número en la secuencia.
    - X_i es el número actual.
    - a es el multiplicador.
    - c es el incremento.
    - m es el módulo.
- La semilla inicial es X_0.
- La elección de los parámetros a, c y m es crítica para el rendimiento del GCL. Una mala elección puede resultar en un período corto y/o patrones no aleatorios.
- Para obtener números aleatorios uniformemente distribuidos entre 0 y 1, se divide cada X_i por m:
  - $U_i = X_i / m$

Ejemplos/Casos de Estudio:

1. **Ejemplo de GCL:**
   - Considera un GCL con a = 5, c = 3, m = 16 y $X_0 = 1$.
   - Calcular los primeros 5 números de la secuencia:
     - $X_1 = (5 * 1 + 3) \bmod 16 = 8$
     - $X_2 = (5 * 8 + 3) \bmod 16 = 7$
     - $X_3 = (5 * 7 + 3) \bmod 16 = 6$
     - $X_4 = (5 * 6 + 3) \bmod 16 = 1$
     - $X_5 = (5 * 1 + 3) \bmod 16 = 8$
   - Observar que la secuencia se repite después de 4 iteraciones. Este es un ejemplo de un período corto debido a la mala elección de los parámetros.
   - Los números aleatorios uniformes correspondientes serían: 8/16, 7/16, 6/16, 1/16, 8/16.
2. **Caso de Estudio: Simulación de Lanzamiento de una Moneda:**
   - Utilizar un GCL para simular el lanzamiento de una moneda justa (50% cara, 50% cruz).
   - Generar un número aleatorio U entre 0 y 1 usando el GCL.
   - Si U < 0.5, se considera "cara".
   - Si U >= 0.5, se considera "cruz".
   - Repetir esto muchas veces para simular múltiples lanzamientos y verificar si la proporción de caras y cruces se acerca al 50%.

Problemas Prácticos/Ejercicios con Soluciones:

1. **Ejercicio:** Implementar un GCL en un lenguaje de programación (ej. Python, C++, Java) con los parámetros a = 1664525, c = 1013904223, $m = 2^{32}$ y una semilla inicial $X_0 = 0$. Generar los primeros 1000 números aleatorios y calcular el promedio.
   - **Solución (Python):**

"'python def gcl(a, c, m, x0, n): """Genera una secuencia de n números aleatorios usando un GCL.""" sequence = [] xi = x0 for _ in range(n): xi = (a * xi + c) % m sequence.append(xi / m) # Normalizar entre 0 y 1 return sequence

Parámetros

a = 1664525 c = 1013904223 m = 2**32 x0 = 0 n = 1000

Generar la secuencia

random_numbers = gcl(a, c, m, x0, n)

Calcular el promedio

average = sum(random_numbers) / n print("Promedio de los primeros 1000 números:", average) “‘

1. **Ejercicio:** Dado un GCL con a = 7, c = 0, m = 13, y semilla $X_0 = 5$, encontrar el período de la secuencia generada.

   - **Solución:** Calcular iterativamente los números hasta que la secuencia se repita.
     - $X_1 = (7 * 5 + 0) \bmod 13 = 9$
     - $X_2 = (7 * 9 + 0) \bmod 13 = 11$
     - $X_3 = (7 * 11 + 0) \bmod 13 = 12$
     - $X_4 = (7 * 12 + 0) \bmod 13 = 6$
     - $X_5 = (7 * 6 + 0) \bmod 13 = 3$
     - $X_6 = (7 * 3 + 0) \bmod 13 = 8$
     - $X_7 = (7 * 8 + 0) \bmod 13 = 4$
     - $X_8 = (7 * 4 + 0) \bmod 13 = 2$
     - $X_9 = (7 * 2 + 0) \bmod 13 = 1$
     - $X_{10} = (7 * 1 + 0) \bmod 13 = 7$
     - $X_{11} = (7 * 7 + 0) \bmod 13 = 10$
     - $X_{12} = (7 * 10 + 0) \bmod 13 = 5$ (Se repite la secuencia)
   - El período es 12.

Materiales Complementarios Recomendados:

- Libro: "Simulation Modeling and Analysis" de Averill M. Law.
- Artículo: "Random Number Generators: Good ones are hard to find" de Park y Miller (Communications of the ACM, 1988).
- Documentación de las funciones `random` en Python o similares en otros lenguajes. (Entender que internamente usan GNA's más sofisticados).