Practica 4 2024

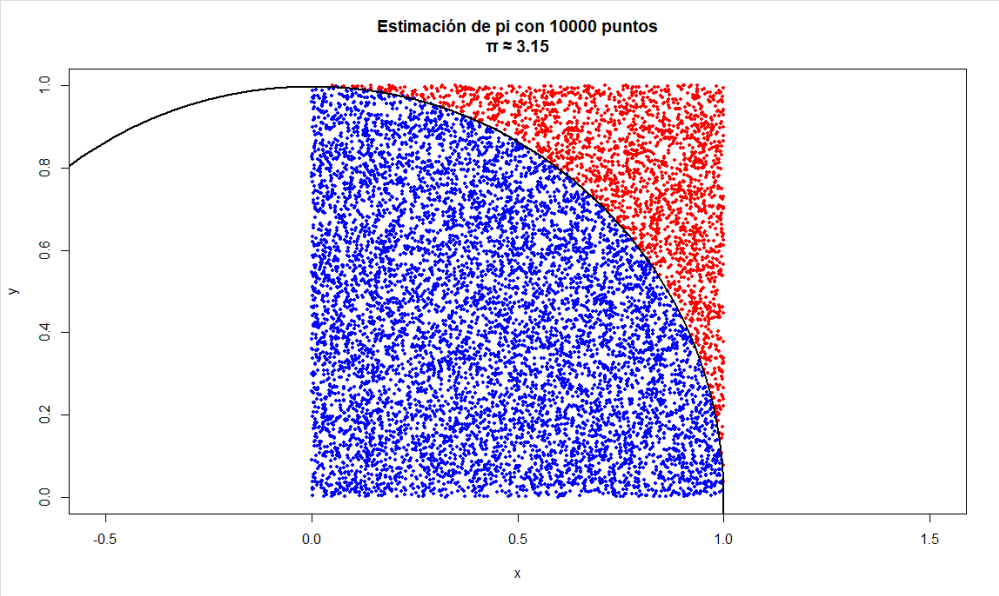
Ignacio Fernández Contreras

ifcau3z@uma.es
Planificación de Proyectos y Análisis de Riesgos . E.T.S Informática.

1 Ejercicio 1

Usa simulación de Montecarlo para estimar el valor de pi. Ten en cuenta que el área de la circunferencia es A=pi*radio**2. Si la circunferencia tiene su centro en el origen y el radio es 1 (circunferencia goniométrica) entonces A=pi y si nos centramos en el primer cuadrante A=pi/4. Genera valores aleatorios, que serán puntos en el primer cuadrante del plano. Dichos puntos pueden caer o no dentro del área de la circunferencia. Caerán dentro de esa área si la distancia del punto aleatorio al origen es menor que 1. La relación entre los puntos bajo la curva con los puntos que están fuera de la misma tendera a pi/4. Realiza un programa en R que realice dicha simulación.

```r
# Número de puntos aleatorios a generar
n_puntos <- 10000

# Generar los puntos aleatorios en el primer cuadrante
set.seed(42)  # Para reproducibilidad
x <- runif(n_puntos, min = 0, max = 1)
y <- runif(n_puntos, min = 0, max = 1)

# Verificar si el punto está dentro de la circunferencia
dentro <- x^2 + y^2 <= 1

# Calcular la estimación de pi
pi_estimado <- (sum(dentro) / n_puntos) * 4

# Mostrar el resultado de la estimación
cat("Estimación de pi usando", n_puntos, "puntos aleatorios:", pi_estimado, "\n")

# Plotear los puntos
plot(x, y, col = ifelse(dentro, "blue", "red"), pch = 16, cex = 0.6,
     main = paste("Estimación de pi con", n_puntos, "puntos\nπ ≈", round(pi_estimado, 5)),
     xlab = "x", ylab = "y", asp = 1)

# Dibujar el cuarto de circunferencia
symbols(0, 0, circles = 1, inches = FALSE, add = TRUE, fg = "black", lwd = 2)
```



2 Ejercicio 2

Realiza un programa en Java, C++ o Mathlab que genere números aleatorios usando el método DE LOS CUADRADOS DEL MEDIO. Este método consiste en generar aleatoriamente un número de cuatro dígitos, denominado la semilla, elevarlo al cuadrado y establecer una forma de tomar los cuatro números centrales del resultado de la exponenciación, ya sea quitando dos o un sólo dígito de cada extremo del resultado.

```
public class MiddleSquareMethod {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Introduce una semilla de 4 d gitos:");
        int seed = scanner.nextInt();

        System.out.print("Introduce el n mero de valores aleatorios a generar:");
        int count = scanner.nextInt();

        System.out.println("N meros generados:");
        for (int i = 0; i < count; i++) {
            seed = middleSquare(seed);
            System.out.println(seed);
        }

        scanner.close();
    }

    public static int middleSquare(int seed) {
        // Elevar al cuadrado la semilla
        int squared = seed * seed;

        // Convertir el resultado en un String para extraer los d gitos centrales
        String squaredStr = String.format("%08d", squared); // Asegurarse de que el
            cuadrado tenga 8 d gitos con ceros a la izquierda
```

```
        // Tomar los cuatro d gitos centrales
        String middleDigits = squaredStr.substring(2, 6);

        // Convertir los d gitos centrales de vuelta a entero
        return Integer.parseInt(middleDigits);
    }
}
```

```
Introduce una semilla de 4 dígitos: 5555
Introduce el número de valores aleatorios a generar: 15
Números generados:
8580
6164
9948
9627
6791
1176
3829
6612
7185
6242
9625
6406
368
1354
8333
```

3 Ejercicio 3

Realiza un programa en Java, C++ o Mathlab que genere números aleatorios usando el método de congruencia mixto:

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

donde:

a = es la constante multiplicativa
c = es la constante adivita
m = es la magnitud del módulo
X0 = es la semilla

1. Muestra la secuencia generada para a=5, b=7, m=8 y X0=4 (Semilla)
2. Modifica el programa para que detecte los ciclos. Como salida debe mostrar la secuencia y el numero de números generados antes del ciclo.
3. Los parámetros que se utilizaron en diferentes máquinas son

UNIVAC:	$a = 515$		**IBM:**	$a = 314$
	$c = 1$			$c = 453$
	$m = 2^{35}$			$m = 2^{31}$
VAX:	$a = 269$		**DEC:**	$a = 159$
	$c = 245$			$c = 806$
	$m = 2^{31}$			$m = 2^{31}$

Comprueba que no tienen ciclos y que la secuencia de números generados están unifórmenle distribuidos en el intervalo [0,1] y que su media es próxima a 1/2.

4. Realiza la siguiente mejora: Se genera una secuencia jn de números aleatorios con el método de congruencia mixto (GCL) y un numero aleatorio y. Se determina el índice k=[y*N/m]. Se reasigna el valor y con jk. El valor de jk se renueva con el GCL. Se vuelve a repetir el proceso desde la determinación del índice k.

3.1 Apartado 1

```
import java.util.Scanner;

public class GeneradorCongruencialMixto {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Leer las constantes y la semilla
        System.out.print("Constante multiplicativa (a): ");
        int a = sc.nextInt();

        System.out.print("Constante aditiva (c): ");
        int c = sc.nextInt();

        System.out.print("M dulo (m): ");
        int m = sc.nextInt();

        System.out.print("Semilla inicial (x0): ");
        int semilla = sc.nextInt();

        sc.close();

        // Llamar a la funci n para generar la secuencia y mostrar el per odo
        generarSecuencia(semilla, a, c, m);
    }

    /**
     * M todo que genera n meros pseudoaleatorios usando el m todo de congruencia
         mixta
     * y determina el per odo de la secuencia.
     */
    private static void generarSecuencia(int semilla, int a, int c, int m) {
        int primerValor = semilla; // Guardar el primer valor de la secuencia
        int periodo = 0;           // Contador del per odo

        System.out.println("\nSecuencia generada:");

        do {
            // Aplicar la f rmula de congruencia mixta
            semilla = (a * semilla + c) % m;
            System.out.println("N mero " + (periodo + 1) + ": " + semilla);
            periodo++;
        } while (semilla != primerValor && periodo < m);

        // Verificar si el per odo es completo o incompleto
        if (periodo == m) {
            System.out.println("\nPer odo completo: " + periodo);
        } else {
            System.out.println("\nPer odo incompleto: " + periodo);
```

```
        }
    }
}
```

```
Constante multiplicativa (a): 5
Constante aditiva (c): 7
Módulo (m): 8
Semilla inicial (x0): 4

Secuencia generada:
Número 1: 3
Número 2: 6
Número 3: 5
Número 4: 0
Número 5: 7
Número 6: 2
Número 7: 1
Número 8: 4

Período completo: 8
```

3.2 Apartado 2

```
import java.util.HashSet;
import java.util.Scanner;
import java.util.Set;

public class GeneradorCongruencialMixto {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Leer las constantes y la semilla
        System.out.print("Constante multiplicativa (a): ");
        int a = sc.nextInt();

        System.out.print("Constante aditiva (c): ");
        int c = sc.nextInt();

        System.out.print("M dulo (m): ");
        int m = sc.nextInt();

        System.out.print("Semilla inicial (x0): ");
        int semilla = sc.nextInt();

        sc.close();

        // Llamar a la funci n para generar la secuencia y detectar ciclos
        detectarCiclo(semilla, a, c, m);
    }

    /**
     * M todo que genera n meros pseudoaleatorios usando el m todo de congruencia
         mixta
```

```java
     * y detecta ciclos en la secuencia.
     */
    private static void detectarCiclo(int semilla, int a, int c, int m) {
        Set<Integer> valoresGenerados = new HashSet<>(); // Conjunto para detectar
            ciclos
        int periodo = 0;  // Contador para el n mero de valores generados antes del
             ciclo
        boolean cicloDetectado = false;

        System.out.println("\nSecuencia generada:");

        // Generar n meros hasta que se detecte un ciclo
        while (!cicloDetectado) {
            System.out.println("N mero " + (periodo + 1) + ": " + semilla);

            // Verificar si el n mero ya fue generado antes (indica un ciclo)
            if (valoresGenerados.contains(semilla)) {
                cicloDetectado = true;
                System.out.println("\nCiclo detectado a partir del n mero: " +
                    semilla);
                System.out.println("N meros generados antes del ciclo: " + periodo)
                    ;
            } else {
                // A adir el n mero al conjunto y continuar
                valoresGenerados.add(semilla);
                semilla = (a * semilla + c) % m;
                periodo++;
            }
        }
    }
}
```

```
Constante multiplicativa (a): 5
Constante aditiva (c): 7
Módulo (m): 8
Semilla inicial (x0): 4

Secuencia generada:
Número 1: 4
Número 2: 3
Número 3: 6
Número 4: 5
Número 5: 0
Número 6: 7
Número 7: 2
Número 8: 1
Número 9: 4

Ciclo detectado a partir del número: 4
Números generados antes del ciclo: 8
```