Universidad de Sonora

Simulación de Sistemas para Ingeniería de software

Maestro Alejandro Gomez Cardenas

Lopez Chaidez Luis Enrique

Laboratorio 4

Hermosillo, Sonora, México 1 de dic. de 2021

Resumen **3**

Introducción **3**

Desarrollo **3**

Diseño e implementación **4**

Evaluación y análisis **6**

Conclusiones finales **7**

Referencias **7**

Números Aleatorios (Laboratorio 4)

Resumen

Librería de generación de números aleatorios.

Introducción

En el lenguaje de programación de tu preferencia y en pareja con un compañero o compañera, se tiene que elaborar una librería que permita generar números aleatorios o pseudoaleatorios. La librería debe incorporar algún mecanismo que permita mantener un nivel aceptable de entropía en el tiempo.

Además, deberás desarrollar una aplicación sencilla que permita hacer uso de la librería. La aplicación deberá tomar como entrada el número de números a generar y opcionalmente, el rango donde se ubicaran los valores generados. En caso de no establecer un rango, los valores generados deberán estar entre 0 y 1. En este mismo sentido, la aplicación producirá como salida un archivo de texto plano con n líneas, donde n es el número de números que se generaron y cada línea tendrá uno de esos números.

Los entregables de este laboratorio son:

- Documentación en formato PDF.
- Código fuente de la librería desarrollada.
- Librería desarrollada y aplicación compiladas (puede ser una webapp).
- Un archivo de texto plano de los primeros 100,000 números aleatorios/pseudoaleatorios generados.

El documento deberá incluir una portada con los nombres de las personas que trabajaron, una introducción, el algoritmo en pseudocódigo explicando el funcionamiento y lógica utilizada en la librería, una sección explicando la estrategia elegida para preservar la entropía de la solución propuesta, una sección en donde se analicen esos primeros 100,000 números generados con algún software especializado en la búsqueda de patrones y los resultados obtenidos de dichos números, y, finalmente, una conclusión sobre el trabajo realizado.

Desarrollo

En cuanto al desarrollo del proyecto, primero empezamos a buscar información y a realizar la algoritmo, al principio nos decidimos por usar un metodo de internet, esté implementaba el método de cuadrados medios para la realización o la obtención de números aleatorios, este método al final no funcionó como se esperaba y solo se dejó al final dentro del archivo .py para documentar el proceso. El método que al final utilizamos utiliza una técnica de generador lineal congruencial o También conocido por sus siglas GLC, este genera un número pseudoaleatorio utilizando una semilla,

un multiplicador, un número o constante aditiva y un módulo, estos son con el que se hacen las operaciones al cambiar los tres valores indicados anteriormente se puede cambiar los resultados obtenidos por la función además se agregó bastantes opciones para que cualquiera que utilice nuestra librería pueda adaptarla a su gusto.

Diseño e implementación

Primero mostraremos el método principal para generar los números aleatorios, se llama randNumWithLCG y este cuenta con 1 atributo obligatorio y 7 atributos opcionales

```
def randNumWithLCG(n, start=0, end=1, write_file=False, roundOutput=False, a=1507471482614205224 5,c=1442695040888963407,m=2**64, *args):
```

A continuación se listan los atributos de dicho método:

- **n** (*int*): Cantidad de números a generar
- **start** (*int*, opcional): Inicio del rango en el cual se desea que se encuentren los números, si no se indica ninguno, este inicia en 0.
- **end** (*int*, opcional): Inicio del rango en el cual se desea que se encuentren los números, si no se indica ninguno, este inicia en 1 .
- **write_file** (*bool*, opcional): Opción para crear un archivo de texto con los números generados. Por defecto este es False.
- **roundOutput** (*bool*, opcional): Opción para redondear los resultados de los números generados. Por defecto este es False.
- **a** (*int*, opcional): Número con el cual se multiplica la semilla en cada iteración para crear un número nuevo. Por defecto este es 15074714826142052245.
- **c** (*int*, opcional): Constante la cual se suma a la semilla en cada iteración para crear un número nuevo. Por defecto este es 1442695040888963407.
- **m** (*int*, opcional): módulo que se aplica a la operación de crear un número nuevo. La operación módulo obtiene el resto de la división de un número entre otro (a veces llamado residuo). Por defecto este es 2**64, en algunos casos se puede cambiar por 2**64.

Después tenemos otra función dentro de la función que nos permite crear una semilla a utilizar

```
def getSeed():
    """Metodo que genera una semilla para los metodos siguientes, utiliza la hora del usuario

    Returns:
        [int]: Numero basado en el tiempo establecido en el reloj del usuario actual
    """
    d = datetime.utcnow()
    seed = (d.year + d.month + d.day + d.hour +
            d.minute + d.second + d.microsecond)
    return seed ** 2
```

En cuanto a las clases secundarias tenemos una clase que nos da un numero flotante para que este sea multiplicado por la semilla y tengamos un numero con decimales

```python
def multiplyLenNormalize(seed):
    """Funcion que regresa un numero flotante que nos permite normalizar los numeros ingresados al poder multiplicarlo despues por el mismo

    Args:
        seed (int): Numero que deseamos saber su numero "normalizador"

    Returns:
        [float]: numero flotante que se puede multiplicar despues por seed para "normalizarlo"
    """
    # Determinamos la longitud de la semilla introducida por el usuario
    tam1 = len(str(seed))
    s = "0."
    for x in range(tam1 - 1):  # Creamos una cadena con 0's dependiendo de la longitud de la semilla
        s = s + "0"
    s = s + "1"
    f = float(s)#Convertimos la cadena a valor flotante
    return f
```

El programa main.py al correrse en la consola, le muestra al usuario 3 opciones, Si desea generar números presione 1, si desea cambiar ciertas opciones presiona 2, y si desea salir del programa presiona 3

```
Que desea Hacer?
1. Generar Numeros.
2. Opciones.
3. Salir.

#
```

Si vamos a la sección opciones vemos que podemos elegir cambiar el inicio del rango de números, que por defecto esta en 0, y el final del rango de números por igual pero este se encuentra en 1, además podemos elegir si al generar los números aleatorios creamos un nuevo archivo que contenga dichos datos, además tenemos que podemos elegir si los resultados de números nos los muestre redondeados al número más cercano, ya por último tenemos que podemos salir del programa.

```
Opciones
1. Inicio de rango de numeros: 0
2. Final de rango de numeros: 1
3.Escribir archivo: No
4.Redondear resultado: No
5. salir

#
```

Evaluación y análisis

La librería quedó bien implementada como se esperaba, ya que haciendo pruebas de resultados nos damos cuenta que este cuenta con resultados realmente aleatorios al ser bastante parejos en cuanto a la cantidad de resultados dados.

El código con el cual se hizo la comprobación es el siguiente:

```
[1]: # Las siguientes líneas importan las librerías numpy y datascience
import numpy as np
from datascience import *
from datetime import datetime
import random_numbers as rand

# Estas líneas configuran lo necesario para poder trabajar con graficas
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')
```

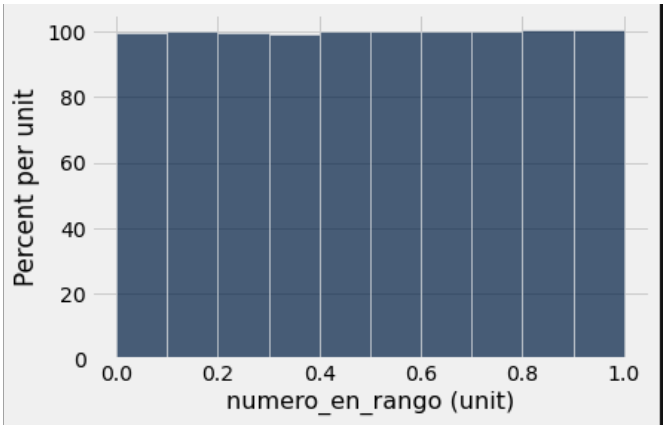
```
[2]: def histogramas(t):
    numero = t.column('numero_en_rango')
    numero_bin = ((max(numero) - min(numero)) / 1).astype(np.int64)


    t.hist('numero_en_rango',unit='unit')

    return numero_bin
```

```
[5]: numeros_aleatorios = Table().read_table("numeros_aleatorios.csv")
numeros_aleatorios.show(3)
```

id	numero_normal	numero_en_rango
0	0.152104	0.824559
1	0.151368	0.820568
2	0.647916	0.351231

... (999997 rows omitted)

Y se muestra la tabla de resultados en cuanto a las proporciones



Esta se encuentra bastante apegada a lo que se esperaría de un generador real de números aleatorios, si bien parte del código no es perfecto, se prevé que pueda llegar a haber algún problema con el generador de semillas en un futuro pero a la fecha no hay problema.

Conclusiones finales

Esta librería sin dudarse, puede ser usada y mejorada para proyectos futuros y ahora se como se implementan ciertas tecnologías para generar mayor aleatoriedad, el trabajo fue más complicado de lo esperado pero esto es por la naturaleza de las computadoras ya que estas no están diseñadas para ser aleatorias, solo pueden seguir un orden.

Referencias

Steele, GL, Vigna, S. Computationally easy, spectrally good multipliers for congruential pseudorandom number generators. *Softw Pract Exper*. 2021; 1- 16. doi:10.1002/spe.3030

Colaboradores de Wikipedia. *Generador lineal congruencial* [en línea]. Wikipedia, La enciclopedia libre, 2021 [fecha de consulta: 1 de diciembre del 2021]. Disponible en <https://es.wikipedia.org/w/index.php?title=Generador_lineal_congruencial&oldid=132913133>.