



# Instituto Politécnico Nacional

## Escuela Superior de Cómputo

### Teoría de la Computación

### Programa Universo

*Profesor: Dr. Juarez Martinez Genaro*

*Alumno: Chen Yangfeng*

*chen1436915478@gmail.com*

*GRUPO*  
*5BM2*

13 de octubre de 2024

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Marco Teórico</b>	<b>2</b>
2.1. Cadenas Binarias . . . . .	2
2.2. Generación de Cadenas Binarias . . . . .	2
2.3. Visualización de Datos . . . . .	2
<b>3. Desarrollo</b>	<b>2</b>
3.1. Programa <i>Universo</i> . . . . .	2
3.1.1. Librerías Utilizadas . . . . .	2
3.1.2. Generación de Cadenas Binarias . . . . .	3
3.1.3. Guardar en Archivo . . . . .	3
3.1.4. Solicitar Valor . . . . .	3
3.1.5. Visualización de Datos . . . . .	4
3.1.6. Función Principal . . . . .	5
3.2. Ejemplo de Ejecución . . . . .	5
<b>4. Gráficas</b>	<b>7</b>
<b>5. Conclusión</b>	<b>8</b>
<b>6. Referencias Bibliográficas</b>	<b>8</b>
<b>7. Anexo</b>	<b>9</b>
7.1. Código completo del <i>Universo</i> implementado en Python . . . . .	9

# 1. Introducción

El estudio de cadenas binarias es fundamental en la teoría de la computación y en el análisis de algoritmos. Las cadenas binarias están compuestas por dos símbolos, comúnmente 0 y 1, y son la base de la representación de datos en computadoras y sistemas digitales. Este programa tiene como objetivo generar un conjunto de cadenas binarias de longitud variable, visualizarlas gráficamente y permitir la exploración de las propiedades de estas cadenas, como la cantidad de ceros y unos en cada cadena. La visualización se realiza tanto en un formato estándar como en un formato logarítmico, proporcionando diferentes perspectivas sobre los datos generados.

## 2. Marco Teórico

### 2.1. Cadenas Binarias

Una cadena binaria es una secuencia de bits, donde cada bit puede ser 0 o 1. La longitud de la cadena se refiere al número total de bits en la secuencia. La teoría de lenguajes formales y autómatas estudia cómo estas cadenas pueden ser generadas y aceptadas por diferentes modelos computacionales.

### 2.2. Generación de Cadenas Binarias

El proceso de generación de cadenas binarias se basa en el concepto de combinatoria. Para una longitud  $n$ , se pueden generar  $2^n$  combinaciones distintas de cadenas binarias. Esto se realiza utilizando la función binaria  $bin(i)$  para convertir números enteros en sus representaciones binarias, rellenándolos con ceros a la izquierda para alcanzar la longitud deseada.

### 2.3. Visualización de Datos

La visualización de datos es un componente crucial en el análisis de información. En este programa, se utilizan gráficos de dispersión para representar la cantidad de ceros y unos en las cadenas generadas. Se utilizan diferentes colores para indicar las características de cada cadena: rojos para ceros, azules para unos y púrpuras para cadenas con igual cantidad de ambos.

## 3. Desarrollo

### 3.1. Programa *Universo*

#### 3.1.1. Librerías Utilizadas

- **NumPy** (`import numpy as np`)

La librería *NumPy* se utiliza principalmente para realizar operaciones matemáticas eficientes. En este caso, se aplica la función logarítmica `np.log10` para generar una gráfica logarítmica de las cadenas binarias. *NumPy* es fundamental para manejar grandes conjuntos de datos y realizar operaciones matemáticas avanzadas.

- **Matplotlib** (import matplotlib.pyplot as plt)

La librería *Matplotlib* permite generar gráficos para visualizar los resultados. En este programa, se utiliza para crear dos tipos de gráficos:

- Un gráfico que representa las cadenas binarias y la cantidad de ceros y unos utilizando `plt.scatter`.
- Un gráfico logarítmico aplicando logaritmos en las cantidades de ceros y unos.

- **Random** (import random)

Esta librería se emplea para generar un valor aleatorio para  $n$ , en caso de que el usuario no proporcione un valor manualmente. La función `random.randint(1, 10)` selecciona un número aleatorio entre 1 y 10.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import random
```

### 3.1.2. Generación de Cadenas Binarias

La función `generar_cadenas_binarias(n)` genera cadenas binarias desde la longitud 0 hasta  $n$ , utilizando un bucle que itera sobre la cantidad total de combinaciones posibles.

```
1 def generar_cadenas_binarias(n):
2     cadenas = ['']
3
4     for longitud in range(1, n + 1):
5         max_num = 2 ** longitud
6         for i in range(max_num):
7             binario = bin(i)[2:] # convertir el numero a binario y quitar el prefijo 0b
8             binario = binario.zfill(longitud) # rellenar ceros a la izquierda
9             cadenas.append(binario)
10
11     return cadenas
```

### 3.1.3. Guardar en Archivo

La función `guardar_en_archivo(cadenas)` permite almacenar las cadenas generadas en un archivo de texto. Esto es útil para la persistencia de datos y su posible análisis posterior.

```
1 def guardar_en_archivo(cadenas, nombre_archivo="Bloque_1\\Universo\\universo.txt"):
2     with open(nombre_archivo, "w") as archivo:
3         for cadena in cadenas:
4             archivo.write(cadena + "\n")
```

### 3.1.4. Solicitar Valor

La función `solicitar_valor()` gestiona la entrada del usuario para determinar la longitud máxima de las cadenas binarias. Permite tanto la entrada manual como la generación aleatoria de un valor si el usuario no proporciona uno.

```
1 def solicitar_valor():
2     while True:
3         try:
4             n = input("Ingrese el valor de n >= 0 o presione Enter para un valor aleatorio: ")
5
```

```

6         if n == "": # si el usuario no ingresa un valor
7             n = random.randint(1, 10)
8             print(f"Se ha generado un valor aleatorio: {n}")
9             return n
10
11         n = int(n)
12         if n >= 0:
13             return n
14         else:
15             print("Ingrese un valor entero positivo.")
16     except ValueError:
17         print("Valor no válido. Por favor, ingrese un número entero >= 0.")

```

### 3.1.5. Visualización de Datos

Las funciones `grafica(cadenas)` y `grafica_log(cadenas)` se encargan de representar gráficamente las cadenas generadas. La primera gráfica la cantidad de ceros y unos en cada cadena, mientras que la segunda aplica una escala logarítmica a estas cantidades, mejorando la visualización de cadenas con diferencias significativas en longitud.

```

1  def grafica(cadenas):
2      plt.figure(figsize=(10, 6))
3
4      posiciones = []
5      cantidades = []
6      colores = []
7
8      for i, cadena in enumerate(cadenas):
9          len_rojo = cadena.count('0')
10         len_azul = cadena.count('1')
11
12         if len_rojo > 0:
13             posiciones.append(i)
14             cantidades.append(len_rojo)
15             colores.append('red') # color para ceros
16
17         if len_azul > 0:
18             posiciones.append(i)
19             cantidades.append(len_azul)
20             colores.append('blue') # color para unos
21
22         if len_rojo == len_azul and len_rojo > 0:
23             posiciones.append(i)
24             cantidades.append(len_rojo)
25             colores.append('purple') # color para el mismo len de 1 y 0
26
27     # graficar todos los puntos
28     plt.scatter(posiciones, cantidades, color=colores)
29     plt.scatter(0, 0, color='black') # graficar cadena vacia
30
31     plt.xlabel('Cadenas')
32     plt.ylabel('Longitud de 0 y 1 (0=rojo, 1=azul)')
33     plt.title('Grafica del universo binario')
34     plt.grid(True)
35     plt.show()
36
37
38  def grafica_log(cadenas):
39      plt.figure(figsize=(10, 6))
40
41      posiciones = []
42      cantidades = []
43      colores = []
44
45      for i, cadena in enumerate(cadenas):
46          len_rojo = cadena.count('0')
47          len_azul = cadena.count('1')
48
49          # Se aplican logaritmos, aadiendo un pequeño valor (1e-5) para evitar log(0)
50          if len_rojo > 0:
51              posiciones.append(i)

```

```

52     cantidades.append(np.log10(len_rojo + 1e-5)) # logaritmo de la cantidad de
53         ceros
54     colores.append('red') # color para ceros
55
56     if len_azul > 0:
57         posiciones.append(i)
58         cantidades.append(np.log10(len_azul + 1e-5)) # logaritmo de la cantidad de
59             unos
60         colores.append('blue') # color para unos
61
62     if len_rojo == len_azul and len_rojo > 0:
63         posiciones.append(i)
64         cantidades.append(np.log10(len_rojo + 1e-5)) # logaritmo para igualdad de 1
65             y 0
66         colores.append('purple') # color para el mismo len de 1 y 0
67
68 # Graficar todos los puntos
69 plt.scatter(posiciones, cantidades, color=colores)
70 plt.scatter(1, np.log10(1e-5), color='black') # graficar cadena vac a
71
72 plt.xlabel('Cadenas')
73 plt.ylabel('log10 de la Longitud de 0 y 1 (0=rojo, 1=azul)')
74 plt.title('Grafica logar tmica del universo binario')
75 plt.grid(True)
76 plt.show()

```

### 3.1.6. Función Principal

La función `main()` orquesta el flujo del programa, gestionando la interacción con el usuario, la generación de cadenas, su almacenamiento y la visualización.

```

1  def main():
2      n = solicitar_valor()
3      cadenas = generar_cadenas_binarias(n)
4      guardar_en_archivo(cadenas)
5
6      opc = input("Presiona '1' para la grafica, o Enter para terminar el programa: ")
7      if opc == "1":
8          grafica(cadenas)
9          grafica_log(cadenas)
10         print(f"Terminando el programa...")
11
12
13
14 if __name__ == "__main__":
15     main()

```

## 3.2. Ejemplo de Ejecución

Al ejecutar el programa, el usuario puede ingresar un valor para  $n$  o permitir que el programa genere un valor aleatorio. Luego, se generan las cadenas binarias y se almacenan en un archivo. Finalmente, el usuario tiene la opción de visualizar las gráficas.

En este caso, hemos decidido ingresar manualmente el valor  $n = 29$ , lo que significa que el programa generará todas las cadenas binarias desde la cadena vacía hasta las de longitud 29.

Ingrese el valor de  $n \geq 0$  o presione Enter para un valor aleatorio: 29

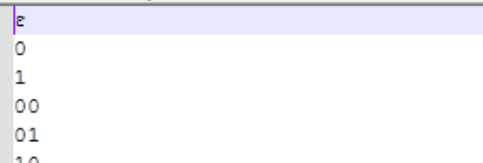
Figura 1: Ingrese 29 en el terminal.

```
Presiona '1' para la grafica, o enter para terminar el programa: 1
```

Figura 2: Ingresar el numero '1' para visualizar las gráficas.

Terminando el programa...

Figura 3: Terminando el programa del Universo.



universo\_n29.txt

1	1
2	0
3	1
4	00
5	01
6	10
7	11
8	000
9	001
10	010
11	011
12	100
13	101
14	110
15	111

Figura 4: Los primeros términos del archivo txt con  $n=29$ .

[illegible]

Figura 5: Los últimos términos del archivo txt con  $n=29$ .

## 4. Gráficas

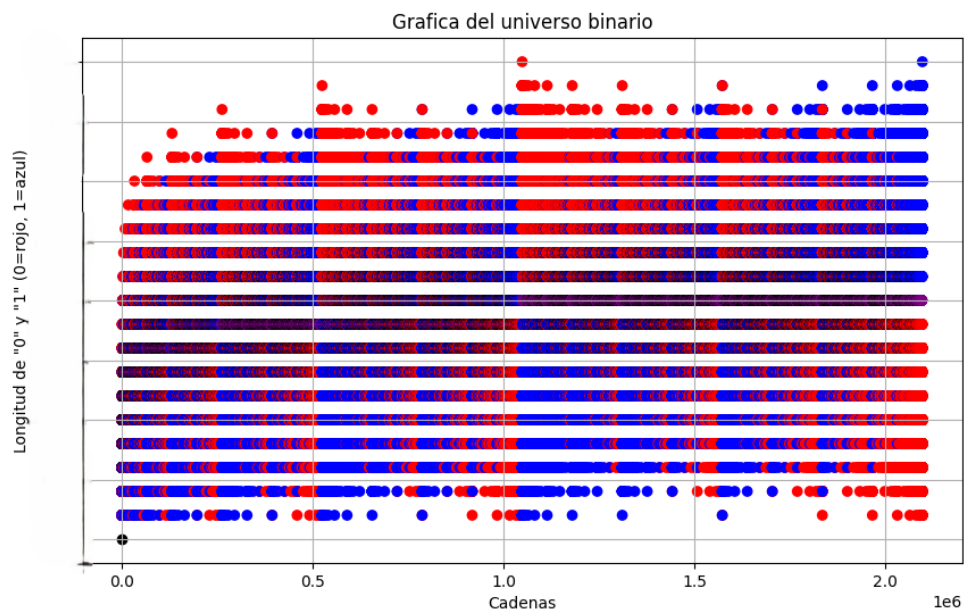


Figura 6: Gráfica del universo.

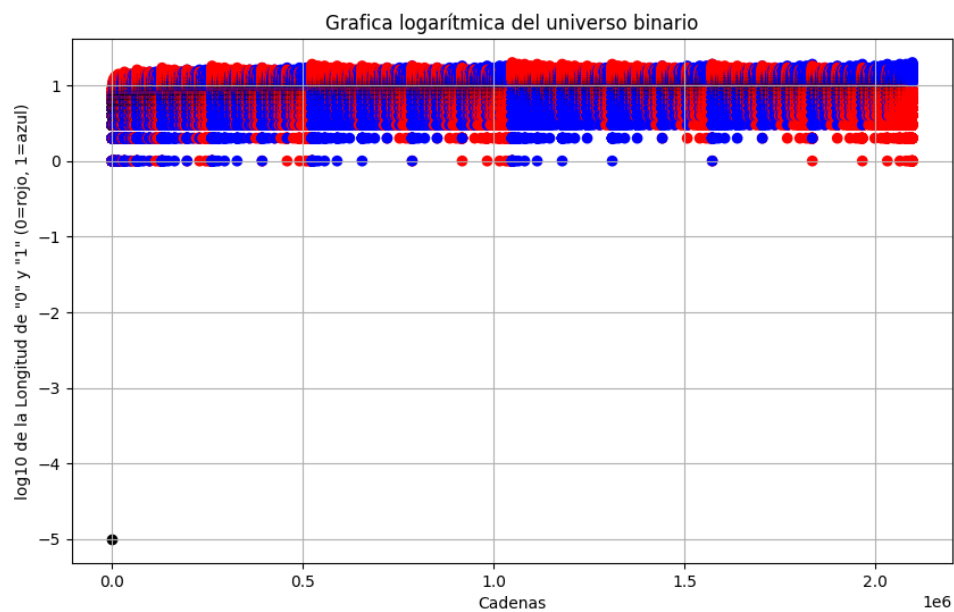


Figura 7: Gráfica con logaritmo.



## 5. Conclusión

El programa *Universo de Binarios* proporciona una herramienta sencilla y efectiva para explorar y visualizar cadenas binarias. A través de la generación combinatoria y la representación gráfica, se pueden analizar las propiedades de las cadenas generadas de una manera clara y accesible. La posibilidad de almacenar los resultados en un archivo también permite un análisis posterior. Este tipo de programas es fundamental en el aprendizaje de la teoría de la computación y en la comprensión de cómo se manipulan y visualizan datos binarios en un entorno digital.

Además de su simplicidad, el programa destaca por su versatilidad. Al permitir la personalización del valor de  $n$ , ofrece a los usuarios la capacidad de explorar diferentes universos binarios, desde los más pequeños hasta los más complejos. Esto lo convierte en una herramienta útil tanto para principiantes que desean comprender los conceptos básicos de las cadenas binarias como para usuarios más avanzados interesados en estudiar el comportamiento de las combinaciones binarias a mayor escala.

La visualización gráfica es otro aspecto clave que facilita la interpretación de los resultados. Las gráficas no solo muestran la cantidad de ceros y unos presentes en cada cadena, sino que también ofrecen una perspectiva más profunda mediante el uso de gráficos logarítmicos, lo que permite observar patrones que podrían no ser evidentes a simple vista. Este enfoque visual ayuda a identificar simetrías, tendencias y diferencias entre las cadenas binarias de diversas longitudes.

Finalmente, el programa resalta la importancia de la representación y análisis de datos en la informática. Las cadenas binarias son la base del procesamiento digital, y comprender su estructura y patrones es crucial para áreas como el diseño de algoritmos, la teoría de la información, y la criptografía. Este tipo de ejercicios fomenta el pensamiento algorítmico y refuerza los conceptos matemáticos subyacentes, contribuyendo al desarrollo de habilidades críticas en el ámbito de la computación y la ciencia de datos.

## 6. Referencias Bibliográficas

### Referencias

- [1] Hopcroft, J., Motwani, R., Ullman, J. (2001). *Introduction to Automata Theory, Languages, and Computation* <https://www-2.dc.uba.ar/staff/becher/Hopcroft-Motwani-Ullman-2001.pdf>

## 7. Anexo

### 7.1. Código completo del *Universo* implementado en Python

```
1  """ Universo de binarios """
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import random
5
6  def generar_cadenas_binarias(n):
7      cadenas = [' ', '0', '1']
8
9      for longitud in range(1, n + 1):
10         max_num = 2 ** longitud
11         for i in range(max_num):
12             binario = bin(i)[2:] # convertir el numero a binario y quitar el
13                                   # prefijo 0b
14             binario = binario.zfill(longitud) # rellenar ceros a la izquierda
15             cadenas.append(binario)
16
17     return cadenas
18
19 def guardar_en_archivo(cadenas, nombre_archivo="Bloque_1\\Universo\\universo.txt"):
20     with open(nombre_archivo, "w") as archivo:
21         for cadena in cadenas:
22             archivo.write(cadena + "\n")
23
24
25 def solicitar_valor():
26     while True:
27         try:
28             n = input("Ingrese el valor de n >= 0 o presione Enter para un valor
29                       aleatorio: ")
30
31             if n == "": # si el usuario no ingresa un valor
32                 n = random.randint(1, 10)
33                 print(f"Se ha generado un valor aleatorio: {n}")
34                 return n
35
36             n = int(n)
37             if n >= 0:
38                 return n
39             else:
40                 print("Ingrese un valor entero positivo.")
41         except ValueError:
42             print("Valor no válido. Por favor, ingrese un número entero >= 0.")
43
44
45 def grafica(cadenas):
46     plt.figure(figsize=(10, 6))
47
48     posiciones = []
49     cantidades = []
50     colores = []
51
52     for i, cadena in enumerate(cadenas):
53         len_rojo = cadena.count('0')
54         len_azul = cadena.count('1')
55
56         if len_rojo > 0:
57             posiciones.append(i)
58             cantidades.append(len_rojo)
59             colores.append('red') # color para ceros
60
61         if len_azul > 0:
62             posiciones.append(i)
63             cantidades.append(len_azul)
64             colores.append('blue') # color para unos
65
66         if len_rojo == len_azul and len_rojo > 0:
67             posiciones.append(i)
68             cantidades.append(len_rojo)
69             colores.append('purple') # color para el mismo len de 1 y 0
```

```

70
71 # graficar todos los puntos
72 plt.scatter(posiciones, cantidades, color=colores)
73 plt.scatter(0, 0, color='black') # graficar cadena vacia
74
75 plt.xlabel('Cadenas')
76 plt.ylabel('Longitud de "0" y "1" (0=rojo, 1=azul)')
77 plt.title('Grafica del universo binario')
78 plt.grid(True)
79 plt.show()
80
81
82 def grafica_log(cadenas):
83     plt.figure(figsize=(10, 6))
84
85     posiciones = []
86     cantidades = []
87     colores = []
88
89     for i, cadena in enumerate(cadenas):
90         len_rojo = cadena.count('0')
91         len_azul = cadena.count('1')
92
93         # Se aplican logaritmos, aadiendo un peque o valor (1e-5) para evitar
94         # log(0)
95         if len_rojo > 0:
96             posiciones.append(i)
97             cantidades.append(np.log10(len_rojo + 1e-5)) # logaritmo de la cantidad
98             # de ceros
99             colores.append('red') # color para ceros
100
101         if len_azul > 0:
102             posiciones.append(i)
103             cantidades.append(np.log10(len_azul + 1e-5)) # logaritmo de la cantidad
104             # de unos
105             colores.append('blue') # color para unos
106
107         if len_rojo == len_azul and len_rojo > 0:
108             posiciones.append(i)
109             cantidades.append(np.log10(len_rojo + 1e-5)) # logaritmo para igualdad
110             # de 1 y 0
111             colores.append('purple') # color para el mismo len de 1 y 0
112
113     # Graficar todos los puntos
114     plt.scatter(posiciones, cantidades, color=colores)
115     plt.scatter(1, np.log10(1e-5), color='black') # graficar cadena vac a
116
117     plt.xlabel('Cadenas')
118     plt.ylabel('log10 de la Longitud de "0" y "1" (0=rojo, 1=azul)')
119     plt.title('Grafica logar tmica del universo binario')
120     plt.grid(True)
121     plt.show()
122
123 def main():
124     n = solicitar_valor()
125     cadenas = generar_cadenas_binarias(n)
126     guardar_en_archivo(cadenas)
127
128     opc = input("Presiona '1' para la grafica, o enter para terminar el programa: ")
129     if opc == "1":
130         grafica(cadenas)
131         grafica_log(cadenas)
132     print(f"Terminando el programa...")
133
134 if __name__ == "__main__":
135     main()

```