

Programa 1: Universo

Dueñas Jiménez Cristian Alexis

November 6, 2022

1 Introducción

Como bien hemos visto en clase, un autómata es un modelo computacional que consiste en un conjunto de estados bien definidos, un estado inicial, un alfabeto de entrada y una función de transición. El problema consta de encontrar el Universo(cantidad de combinaciones posibles de cadenas dado un alfabeto) dada una N . Como se nos plantea, el alfabeto es binario, con el uno y el cero como únicos caracteres. La N representa la longitud en bits máxima que se pueden encontrar en el Universo. Para encontrar esto, se hace uso de la suma binaria. Se suma de uno en uno, para encontrar todas las combinaciones posibles. Sin embargo, hay algunas combinaciones que no se reflejan únicamente sumando de uno en uno.

2 Marco Teórico

Un autómata es un modelo matemático para una máquina de estado finito, en el que dada una entrada de símbolos, salta mediante una serie de estados de acuerdo a una función de transición (que puede ser expresada como una tabla). Esta función de transición indica a qué estado cambiar dados el estado actual y el símbolo leído. Su universo está dado por todas las posibilidades que puede encontrar un autómata dado un lenguaje.

3 Desarrollo

Para encontrar todas las combinaciones posibles, implementamos una función llamada *sigma* que tiene como parámetro la N que ingresa el usuario.

Primero, se hace uso de listas para guardar datos que nos serán de utilidad para posteriormente graficar nuestros datos.

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 import os
4 import sys
5
6 def graficar(largo, unos, ejeX, ejeY):
```

```

7
8 plt.plot(largo, unos, marker="")
9 plt.xlabel(ejeX)
10 plt.ylabel(ejeY)
11 plt.title("GRAFICA")
12 plt.show()
13
14 def sigma(n):
15
16     unos = [] #Arreglo para almacenar la cantidad de unos en cada
17     cadena binaria.
18     largo =[] #Arreglo para determinar cuantas cadenas se generan
19     cont = -1 #Contador el cual incrementa con cada iteraci n de
20     los fors.
21     concat = ""
22     countElements = 0
23     newString = ""
24     unos64 = []
25     cadenas64 = 0
26     largo64 = []
27     f1 = open("../PARCIAL 1/PROGRAMAS/Programa1/outputs/universo.
28     txt", "w", encoding="utf-8")
29     f2 = open("../PARCIAL 1/PROGRAMAS/Programa1/outputs/
30     Concatenadas.txt", "w", encoding="utf-8")
31     f3 = open("../PARCIAL 1/PROGRAMAS/Programa1/outputs/Segmentadas
32     .txt", "w", encoding="utf-8")
33     f1.write("{")
34
35     for a in range(n+1): ##Generamos nuestro bucle, este s lo es
36     para imprimir la cantidad de iteraciones que se llevan a cabo.
37     Ponemos n+1, ya que
38         print("n = ",a) #Imprimimos las iteraciones realizadas,
39         conforme se va realizando cada una de ellas.
40         for x in range(2 ** a): #En este segundo for, determinamos
41         la cantidad de cadenas que habr seg n el valor de a, es
42         decir, seg n el n mero de iteraci n. Esto se determina con
43         2^a.
44
45         cont += 1 #Incrementamos el contador que inicializamos
46         anteriormente, esto para llevar el conteo de cadenas.
47         binario = bin(x)[2:]
48
49         if len(binario) < a: #Condici n para agregar ceros.
50             binario = ('0' * (a-len(binario))) + binario
51
52         if a == 0:
53             binario = ' ' #Cuando nuestro valor ingresado es
54             "0", el universo se determina con su respectivo s mbolo.
55
56         if a > 0:
57             concat = concat+binario
58
59         f1.write(binario + ',') #Mandamos a escribir en nuestro
60         archivo cada una de las cadenas generadas.
61         unos.append(binario.count('1')) #Realizamos el conteo
62         de 1's en cada cadena y lo guardamos en el arreglo de unos.
63         largo.append(cont) #Almacenamos la cantidad de cadenas

```

```

49     que van en cada iteraci n
50     f2.write(concat)
51     f2.close()
52     f1.write("}") #Una vez finalizados los ciclos, cerramos nuestro
                    conjunto con una llave.
53     f1.close() #Cerramos el archivo
54     f3.write("Segmentadas" + os.linesep)
55
56     for char in concat:
57         countElements+=1
58         newString = newString+char
59
60         if countElements == 64:
61             cadenas64 += 1
62             unos64.append(newString.count('1'))
63             largo64.append(cadenas64)
64             countElements = 0
65             f3.write(newString + os.linesep)
66             newString = ""
67
68     graficar(largo,unos, "Cantidad de Cadenas", "Cantidad de 1's")
69     #Mandamos llamar la funci n graficar, la cual nos muestra la
    cantidad de 1's en cada cadena, por lo cual le mandamos la
    variable largo y unos.
70     graficar(largo,np.log10(unos), "Cantidad de Cadenas", "Cantidad
    de 1's (Log10)")
71     graficar(largo64,unos64, "Cantidad de Cadenas", "Cantidad de 1'
    s")
72     graficar(largo64,np.log10(unos64), "Cantidad de Cadenas", "
    Cantidad de 1's (Log10)")
73 #Generamos nuestro men de opciones
74 opc = "1"
75 print("MENU DE OPCIONES")
76 while opc == "1":
77
78     opc = input("Seleccione una opci n: \n1.- Ingresar n \n2.-
    Finalizar\n")
79
80     if opc == "1":
81         n = int(input("Ingrese el valor de n: "))
82         sigma(n)
83     else:
84         if opc == "2":
85             print(f"FIN DE LA EJECUCION")
86             sys.exit(0)
87         else:
88             if opc != "1":
89                 print("Seleccione una opcion valida")

```

Algoritmo 1: Implementaci3n del Universo

El primer ciclo, nos ayuda a saber en que n nos encontramos. En otras palabras, nos dice la cantidad de bits que vamos a tener en la cadena generada. El segundo for hace uso de la suma binaria. Aumenta de uno en uno los elementos.

La sentencia $if len(binario) < a$: nos permite agregar ceros a la izquierda para tener una cadena que sea existente dentro de la cantidad de bits. Para entenderlo mejor, supongamos que el usuario ingresa $n = 3$. El primer ciclo for irá de 0 a 3. En la primera iteración, la longitud en bits máxima que puede tener la cadena es 0. En la segunda iteración, la longitud ahora es de uno. Entra en el segundo for, donde despliega todas las combinaciones posibles cuando $n = 1$. Al terminar el ciclo, $n = 2$. Entra de nuevo en el segundo for donde mostrará de nuevo las combinaciones posibles. Sin embargo, al convertir los decimales en binario, no toma en cuenta la condición de la longitud. Para que esto se cumpla, resta la cantidad esperada de bits, menos la longitud de la cadena la arrojada al convertir de decimal a binario. El resultado de eso es la cantidad de ceros a la izquierda que se la agrega a la cadena. De esta forma, se garantiza que el programa no tendrá cadenas repetidas. Una vez obtenida la cadena, la escribe en el archivo. Del mismo modo, guarda en arreglos la cantidad de unos en la cadena, el total de caracteres en la cadena, y la cantidad de cadenas encontradas al momento. Además, cada que se genera una cadena, se va concatenando para que al final forme una sola, establecemos una condición para que no se agregue el símbolo de Epsilon en la cadena. Una vez que tenemos la cadena unida, la recorremos para formar cadenas de 64 bits.

Para graficar, hacemos uso de la librería matplotlib. El programa tiene una función la cual se reciben los arreglos llenados en la función sigma, que usa para representarlos en la gráfica. Dado que el archivo generado para $n=27$ es muy grande y ocupa mucho espacio, se tomó una prueba con $n=20$, donde el archivo sí es manejable.

3.1 Implementación

[illegible]

Archivo Generado

```
111111101111001111,1111111110111010000,1111111101111010001,1111111101111010010,1111111101111010011,1111111101
11,11111111100000000000,11111111110000000001,11111111110000000010,11111111110000000011,11111111110000000100,111111
0110000,111111111110000110001,11111111110000110010,11111111110000110011,11111111110000110100,11111111110000110101,1
10001100001,11111111110001100010,11111111110001100011,11111111110001100100,11111111110001100101,11111111110001100110,
1111111100010010010,11111111110010010011,11111111110010010101,11111111110010010101,11111111110010010110,111111111100
0,11111111110011000011,11111111110011000100,11111111110011000101,11111111110011000110,11111111110011000111,11111111
10011,11111111110011101100,11111111110011110101,11111111110011110110,11111111110011110111,11111111110011111000,11
10100100100,11111111110100100101,11111111110100100110,11111111110100100111,11111111110100101000,111111111101001010
111110101010101,11111111110101010110,11111111110101010111,11111111110101010100,11111111110101010101,11111111110101
,11111111110110000110,11111111110110000111,11111111110110001000,11111111110110001001,11111111110110001010,11111111
10110,11111111110110110111,11111111110110111000,11111111110110111001,11111111110110111010,11111111110110111011,11
0111100111,11111111110111101000,11111111110111101001,11111111110111101010,11111111110111101011,1111111111011110110
11111000011000,11111111111000011001,11111111111000011010,11111111111000011011,11111111111000011100,11111111111000
11111111111001001001,11111111111001001010,11111111111001001011,11111111111001001100,11111111111001001101,1111111111
1001,11111111111001111010,11111111111001111011,11111111111001111100,11111111111001111101,11111111111001111110,111
010101010,11111111111010101011,11111111111010101100,11111111111010101101,11111111111010101110,11111111111010101111
11111011011011,1111111111101011100,1111111111101011101,1111111111101011110,1111111111101011111,1111111111101111
1111111111100001100,11111111111100001101,1111111111100001110,1111111111100001111,1111111111100010000,11111111111
100,11111111111100111101,11111111111100111110,11111111111100111111,11111111111101000000,11111111111101000001,11111
0101101,11111111111101101110,11111111111101101111,11111111111101110000,11111111111101110001,11111111111101110010,
11111110011110,11111111111110011111,11111111111110100000,1111111111110100001,1111111111110100010,1111111111111010
111111111111001111,1111111111111010000,1111111111111010001,1111111111110100010,1111111111111010011,1111111111111010011
11,}
```

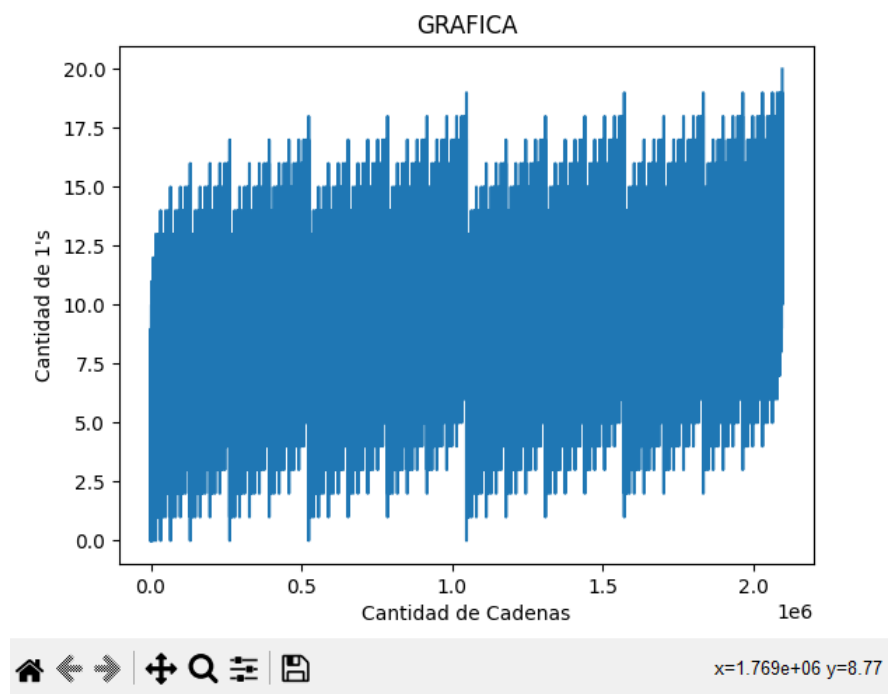
```

1.- Ingresar n
2.- Finalizar
1
Ingrese el valor de n: 20
n = 0
n = 1
n = 2
n = 3
n = 4
n = 5
n = 6
n = 7
n = 8
n = 9
n = 10
n = 11
n = 12
n = 13
n = 14
n = 15
n = 16
n = 17
n = 18
n = 19
n = 20

```

Consola

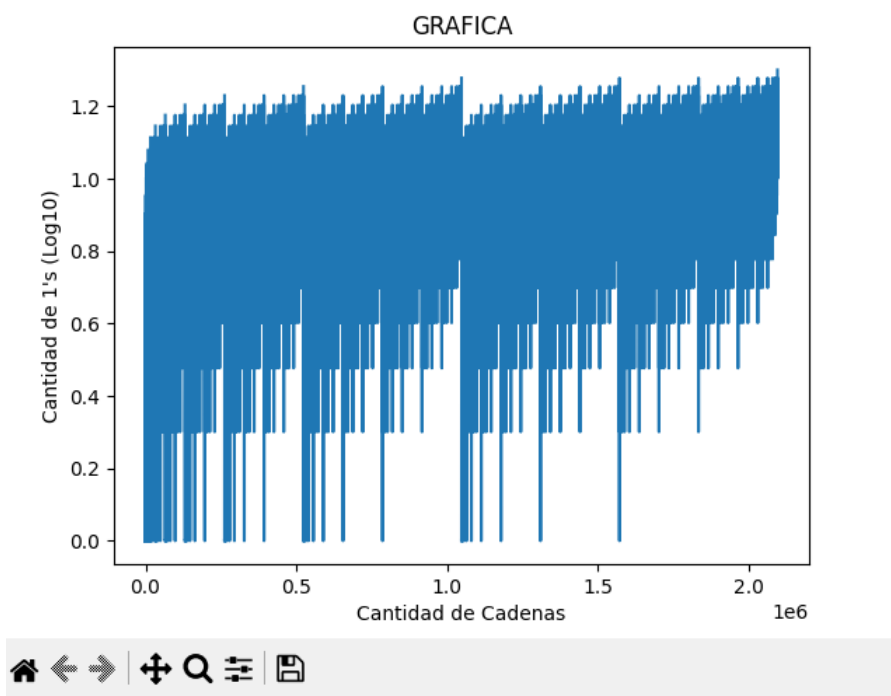
Figure 1



Grafica 1)

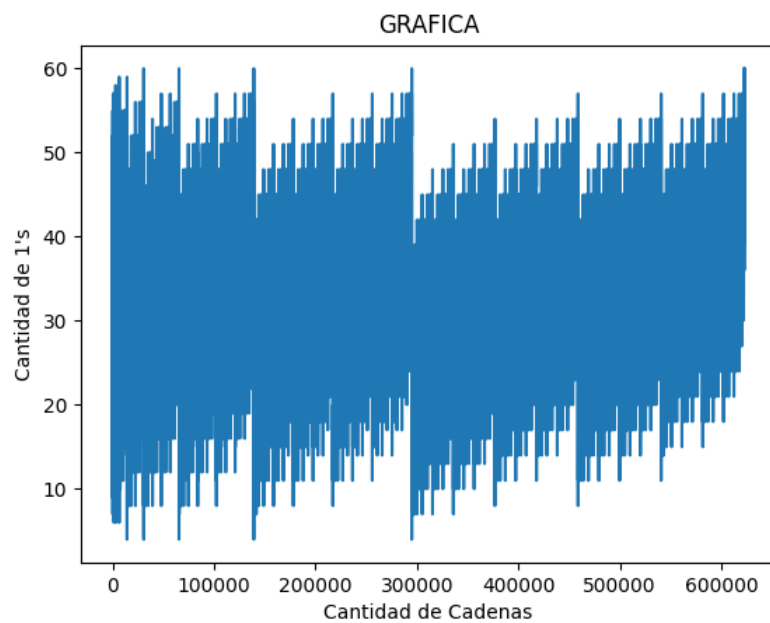
Figure 1

— □ ×



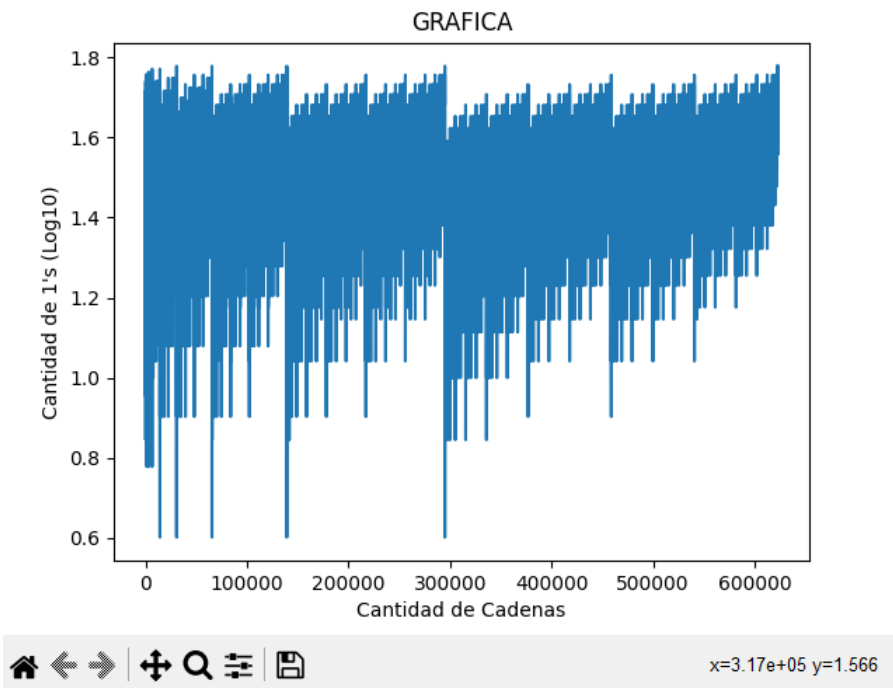
Grafica 2

Figure 1



Grafica 3 (Cadenas de 64 bits)

Figure 1



Grafica 4 (Cadenas de 64 bits)

Segmentadas

```
0100011011000001010011100101110111000000010010001101000101011001
111000100110101011110011011110111100000000100010000110010000101
0011000111010000100101010010110110001101011100111110000100011001
0100111010010101101101011111000110011101011011111001110111110111
110000000000100001000001100010000010100011000011100100000100100
10100010110011000011010011110001111010000010001010010010011010100
0101010101100101110110000110010110100110110111000111010111100111
1110000010000110001010001110010010010110011010011110100010100110
1010101011101100101101101110101111110000110001110010110011110100
1101011101101101111110001110011110101110111111001111011111101111
110000000000001000001000000110000100000010100001100000111000100
0000100100010100001011000110000011010001110000111100100000010001
0010010001001100101000010101001011000101110011000001100100110100
0110110011100001110100111100011111010000001000010100010010001101
0010001001010100110010011101010000101001010101010101101101100010
1101010111001011110110000011000101100100110011011010001101010110
1100110111011100001110010111010011101101111000111101011111001111
1110000001000001100001010000111000100100010110001101000111100100
0100100110010101001011100110010011011001110100111110100001010001
```

Cadenas de 64 bits

References

- [1] Sánchez, E. H. (2019). Lic. en Informática. Unidad de Apoyo para el Aprendizaje. <https://programas.cuaed.unam.mx/repositorio/moodle/pluginfile.php/1163/modresource/content/1/contenido/index.html>
- [2] GeeksforGeeks. (2019, 16 octubre). N Queen Problem — Backtracking-3. <https://www.geeksforgeeks.org/n-queen-problem-backtracking-3/?ref=rp>
- [3] Ocho reinas reina atacarr.JPG. (2006, 20 mayo). [Imagen]. Wikipedia. https://upload.wikimedia.org/wikipedia/commons/d/d6/Ocho_reinas_reina_atacarr.JPG
- [4] Ocho reinas reina atacar fila. (2006, 10 mayo). [Imagen]. Wikipedia. https://upload.wikimedia.org/wikipedia/commons/a/ad/Ocho_reinas_reina_atacar_fila.JPG
- [5] GeeksforGeeks. (2021, 9 febrero). Backtracking — Introduction. <https://www.geeksforgeeks.org/backtracking-introduction/>
- [6] Backtracking. (s. f.). [Imagen]. InterviewBit. <https://ibpublicimages.s3-us-west-2.amazonaws.com/tutorial/backtracking1.png> Figura 2
- [7] [Árbol de búsqueda del problema de las 4 reinas]. (2015, 5 febrero). Ivan Voroshilin's Blog. <https://ivoroshilin.files.wordpress.com/2015/02/backtrack.png>