

Instituto Politécnico Nacional

ESCUELA SUPERIOR DE COMPUTACIÓN

TEORÍA DE LA COMPUTACIÓN

PRÁCTICA 1

Universo de 0's y 1's

Profesor: Juarez Martinez Genaro

Alumno: Jimenez Luna Rodrigo Efren

Email: jimenez.luna.efren@gmail.com

Grupo: 4CM2

Índice

1. Introducción	2
2. Marco Teórico	2
2.1. Lenguaje, alfabeto y cadenas	2
3. Desarrollo	2
3.1. Programa 1 Universo	2
3.2. Planteamiento y solución	3
3.3. Código	3
4. Resultados	6
5. Conclusión	8

1. Introducción

Para el estudio posterior de la teoría de autómatas, es primordial comprender conceptos como los lenguajes, alfabetos, cadenas, etc. En este caso el objetivo será generar mediante algún lenguaje de programación una cantidad de n cadenas obtenidas a partir del lenguaje formado por todas las combinaciones de 0's y 1's.

Una vez implementando este lenguaje, se procederá a realizar pruebas de escritorio al programa evaluando como aumenta la cantidad de símbolos en cada cadena, en este caso los símbolos 1.

2. Marco Teórico

2.1. Lenguaje, alfabeto y cadenas

En el campo de la teoría de la computación podemos definir a un alfabeto como un conjunto no vacío el cual contiene símbolos, se utiliza la letra griega (Σ) para representarlo, además cuando un símbolo (σ) pertenece a este alfabeto se denotará como $\sigma \in \Sigma$ [1].

Dentro de un alfabeto encontraremos cadenas, las cuales se componen por una serie de símbolos que pertenecen a un cierto alfabeto. Estas se denotan regularmente por la letra w [1].

Asimismo, un lenguaje esta formado por las cadenas que genera un determinado alfabeto, este también es un conjunto y se suele representar por la letra L . Añadido a esto, existe la notación de cerradura (Σ^*), esta hace referencia al lenguaje que contiene todas las posibles cadenas que se pueden formar a partir de un alfabeto Σ [1].

3. Desarrollo

3.1. Programa 1 Universo

Para esta práctica, se deberá desarrollar un programa que genere el universo de cadenas binarias (Σ^n). Dado un entero n ya sea que pueda ser introducido por el usuario o que el programa lo genere de forma aleatoria, cumpliendo que los valores de n se encuentren en el intervalo cerrado $[0, 1000]$.

El programa deberá contar con un menú el cual indique tres opciones para generar un valor de n aleatorio, otra opción para indicar el valor manualmente o salir del programa, así como imprimir la salida en un archivo de texto en forma de notación de conjuntos.

3.2. Planteamiento y solución

Con el objetivo de plantear una solución para el problema planteado, primero habrá que entender lo que se solicita, en este caso generar el universo de cadenas binarias. Por lo que podemos afirmar que se deberá generar el lenguaje formado por el alfabeto $\{0, 1\}$, donde cada cadena será las posibles combinaciones de símbolos de longitud n .

$$L = \{\epsilon, 0, 1, 00, 01, 10, 11, \dots\}$$

Si bien, este lenguaje se parece a otro formado por la cuenta en binario desde 0 hasta n , en este lenguaje no encontraremos las cadenas $\{00, 01, 000, 001, \dots\}$. Es por ello que una mejor forma de abordar el problema es formando todas las posibles combinaciones de 0's y 1's para una cadena de longitud n . Nótese que si tomamos todas las cadenas formadas para $n = 1$, podemos formar las siguientes para $n = 2$ con tan solo unir los símbolos de nuestro diccionario y estos.

En matemáticas, se conoce un proceso similar a este, llamado producto cartesiano, donde realizamos operaciones entre todos los elementos de pares ordenados. Para este problema podemos realizar el mismo proceso, pero entre los elementos realizaremos la operación de concatenación.

Así que para generar este lenguaje, basta con tomar los elementos de nuestro diccionario, en este caso $\{0, 1\}$, y realizar el producto cartesiano con los elementos de longitud n anteriores.

Por ejemplo, para generar las cadenas de longitud $n = 3$, tomaremos las cadenas anteriores de longitud $n = 2$ y realizaremos el producto con el par ordenado $(0, 1)$. El proceso se vería algo así:

$$(0, 1) * (00, 01, 10, 11) = (000, 001, 010, 011, 100, 101, 110, 111)$$

Con esto, es posible generar cualquier cadena de n elementos siguiendo las reglas del lenguaje. Traducido a un lenguaje de programación, que en este caso será python3, solo basta con realizar la concatenación con cada elemento de la cadena de n elementos anterior y la cadena $[0, 1]$ utilizando bucles.

3.3. Código

A partir de la solución planteada en la sección anterior, se muestra el código implementado en el lenguaje de programación python:

```
# IPN — Escuela Superior de Computaci n
# Autor: Jimenez Luna Rodrigo Efren
# Grupo: 4CM2

import itertools
import os
```

```

import random
import matplotlib.pyplot as plt
import math

from tqdm import tqdm

option = 1

while option:
    # Menu
    #####
    print("Elija una de las siguientes opciones:")
    print("1) Modo Manual")
    print("2) Modo Aleatorio")
    print("3) Salir")

    option = int(input())

    os.system("cls")

    if option in (1, 2, 3):
        if option == 1:
            print("Indique el valor de n en un intervalo de [0, 1000]: ")
            size = int(input())
            if size > 1000 or size < 0:
                print("Valor de n fuera de reango, intente de nuevo!\n")
                continue
        elif option == 2:
            size = random.randint(0, 1001)
            print("El valor generado para n es {}".format(size))
        else:
            print("Sesion terminada")
            break;
    else:
        print("Opcion Invalida")
        continue;
    #####

    #Genera Universo
    #####
    file = open("output.txt", "w")
    file.write("L = {e}")
    y = [0]

    # Solo se calculan las combinaciones para un valor de n diferente de 0
    if size > 0:

```

```

ones_count = 0
aux = ['0', '1']

file.write(", 0, 1")

y.append(1)

with tqdm(total = 2** (size + 1) - 4) as pgbars:
    for i in range(size - 1):
        memory = aux
        aux = []
        # Concatenaci3n del resultado anterior con el par ordenado {0,
        for j in ['0', '1']:
            for k in memory:
                file.write(", " + j + k)
                aux.append(j + k)
                ones_count += (j + k).count('1')
                pgbars.update(1)

        y.append(ones_count)
        ones_count = 0

file.write("}")
file.close()
#####

#Graficaci3n
#####
plt.plot(y)
plt.xlabel('String')
plt.ylabel('Number of 1s')
plt.title('String and 1s relation')
plt.show()

log10_values = []

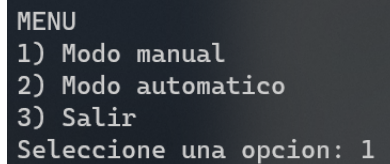
for value in y:
    if value != 0:
        log10_values.append(math.log10(value))

plt.plot(log10_values)
plt.xlabel('String')
plt.ylabel('Log10 of 1s')
plt.title('Log10 ones and string relation')
plt.show()
#####

```

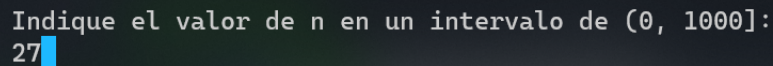
4. Resultados

Para probar la implementación del problema planteado, se realizó una prueba de escritorio tal como se muestra en la figura 1 y 2, donde eligiendo la opción número uno del menú (Modo manual) y se indicó un valor de $n = 27$.



```
MENU
1) Modo manual
2) Modo automatico
3) Salir
Seleccione una opcion: 1
```

Figura 1. Selección de modo. Fuente: Elaboración propia



```
Indique el valor de n en un intervalo de [0, 1000]:
27
```

Figura 2. Se introduce un valor de $n = 27$. Fuente: Elaboración propia.

Posteriormente, el programa comenzó a ejecutarse, demorando 13 minutos y 45 segundos en generar el archivo de texto. Archivo el cuál contiene el universo en notación de conjuntos.

Tal y como se muestra en la figura 3, al graficar la relación de 1's que contiene cada cadena notamos que esta tiene un comportamiento exponencial, lo cual es sencillo mostrar, ya que si estos mismo valores los graficamos aplicando la función logarítmica como se observa en la figura 4, la gráfica resultante es una línea recta.

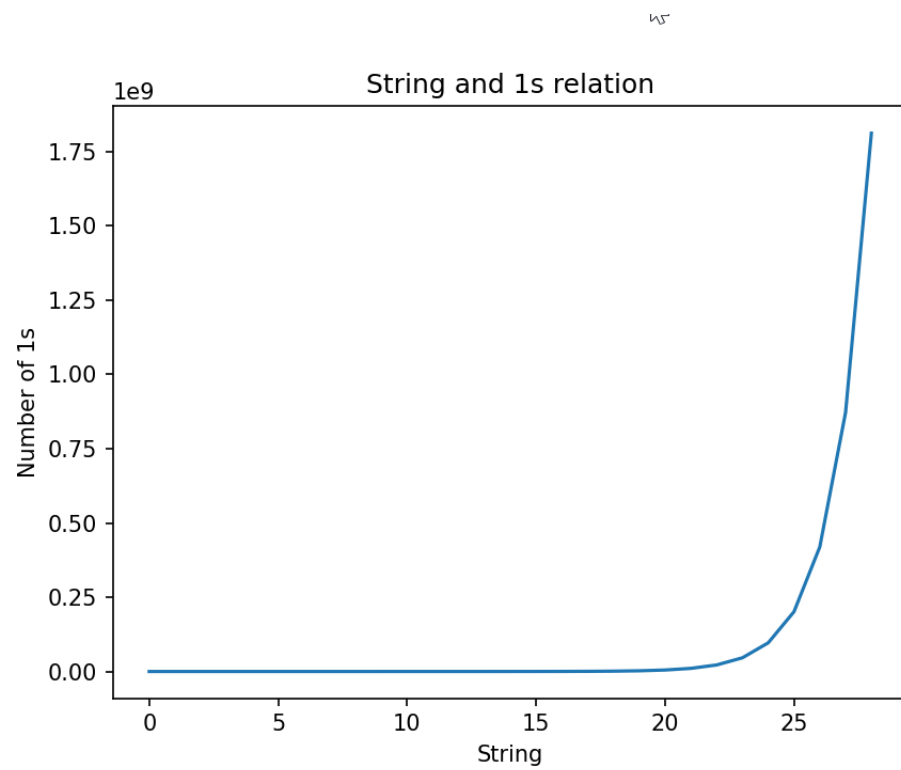


Figura 3. Gráfica de número de 1's por cadena.

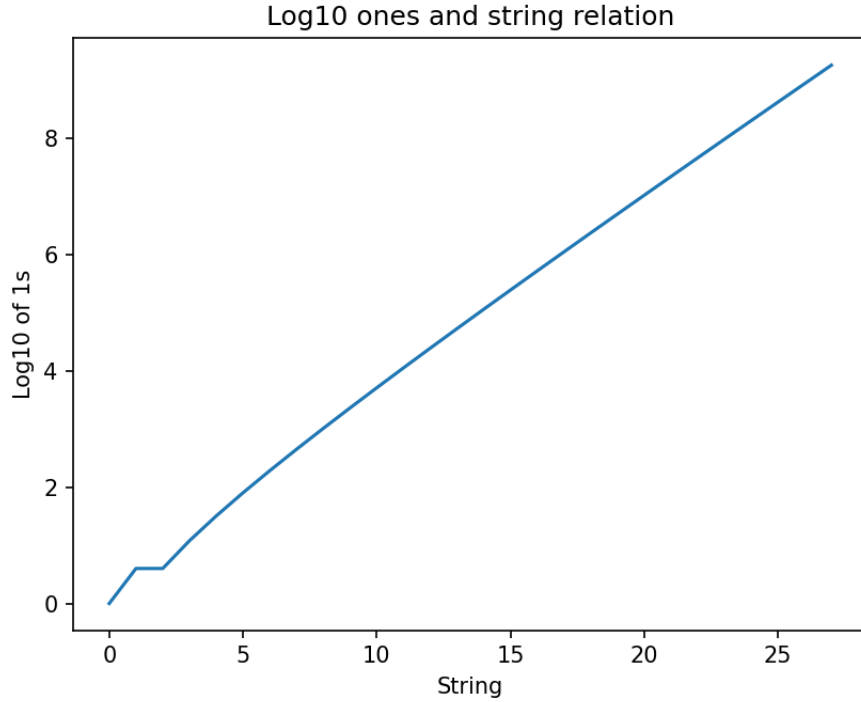


Figura 4. Gráfica \log_{10} del número de 1's por cadena.

5. Conclusión

A lo largo de esta practica se abarco el tema de los lenguajes, alfabetos y cadenas, por lo que desarrollar un programa que trabaje con estos tres conceptos ayuda a mejorar el entendimiento de estos. En este caso el generar un lenguaje tiene un gran impacto ya que posteriormente este tipo de programas pueden ayudar a generar los archivos de entrada para otros algoritmos donde se hace uso de un lenguaje, en el caso de la materia, este algoritmo podría ser un autómata.

Además fue posible trabajar en la optimización de nuestros programas para que estos puedan trabajar con grandes cantidades de datos y no solo cuenten con la lógica para resolver el problema, además de lograr mostrar estos datos de forma entendible para su interpretación. Durante esta práctica se mostró en forma de gráfica los resultados del lenguaje generado, ayudando así en comprender el comportamiento del lenguaje que se generó, el cuál esta vez se compone de todas las combinaciones de 1's y 0's, y como se vio en la primera gráfica, la cantidad de 1's se comportó de manera exponencial, explicando así por que para un valor más grande de n , el coste computacional aumentaba.

Referencias

- [1] G. Padilla, “Alfabetos, cadenas y lenguajes,” 2005. [Online]. Available: http://delta.cs.cinvestav.mx/~mcintosh/comun/summer2006/algebraPablo_html/node4.html