

**Instituto Politécnico Nacional**

ESCUELA SUPERIOR DE COMPUTACIÓN

TEORÍA DE LA COMPUTACIÓN

PRÁCTICA 6

*Palíndromos*

Profesor: Juarez Martinez Genaro

Alumno: Jimenez Luna Rodrigo Efren

Email: jimenez.luna.efren@gmail.com

Grupo: 4CM2

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Marco Teórico</b>	<b>2</b>
2.1. Gramáticas libres de contexto . . . . .	2
<b>3. Desarrollo</b>	<b>2</b>
3.1. Programa 6 Palíndromos . . . . .	2
3.2. Planteamiento y solución . . . . .	2
3.3. Código . . . . .	3
3.4. Código de la función principal (main.py) . . . . .	3
3.5. Código de los métodos utilizados (functions.py) . . . . .	4
<b>4. Resultados</b>	<b>5</b>
<b>5. Conclusión</b>	<b>6</b>

## 1. Introducción

A lo largo de esta practica se implementará la gramática libre de contexto que genera palíndromos compuestos de 0's y 1's de una longitud  $N$ , el programa permitirá al usuario indicar el valor  $N$  o que este sea elegido de forma aleatoria. Más adelante se presentará con más detalle el funcionamiento del programa.

## 2. Marco Teórico

### 2.1. Gramáticas libres de contexto

Las gramáticas libres de contexto consisten en una colección de reglas de sustitución llamadas producciones las cuáles generan un lenguaje libre de contexto. Las producciones estan compuestas por variables y símbolos terminales. Todas las gramáticas comienzan con una variable inicial y termina una vez que ya no quedan variables y solo hay terminales. Las gramáticas libres de contexto utilizan la recursividad para ir sustituyendo las variables conforme se van aplicando las reglas. Para representar las transiciones que va realizando se utiliza la notación  $\alpha A \beta \Rightarrow \alpha \gamma \beta$  [1]

## 3. Desarrollo

### 3.1. Programa 6 Palíndromos

En esta práctica se desarrollará un programa que implemente la gramática libre de contexto que genere palíndromos de tamaño  $N$  ingresado por el usuario o elegido de forma aleatoria y el resultado se imprimirá en pantalla, así como imprimir las derivaciones en un archivo de texto.

Con el uso de un menú el usuario podrá elegir entre 3 opciones: Modo manual, modo aleatorio, salir. Si la cadena es mayor de 20 caracteres esta no se imprimirá en pantalla, solo en el archivo.

### 3.2. Planteamiento y solución

Para implementar la solución a este problema será necesario implementar la gramática libre de contexto con las siguientes producciones:

- $P \rightarrow \epsilon$
- $P \rightarrow 0$
- $P \rightarrow 1$
- $P \rightarrow 0P0$
- $P \rightarrow 1P1$

Además, para que el palíndromo sea diferente en cada ejecución del programa, se implementará de forma que la gramática elija entre las producciones de forma aleatoria hasta llegar a la cantidad de caracteres deseado.

### 3.3. Código

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

### 3.4. Código de la función principal (main.py)

```
import os
import random

from functions import *

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("clear")

    if option in (1, 2, 3):
        if option == 1:
            print(" Ingrese el tamaño de la cadena en un intervalo [1, 100000]:")
            size = int(input())
            # print("{}".format(str))

            if size > 100000 or size < 0:
                print("El intervalo es inválido\n")
                continue
        elif option == 2:
            size = random.randint(1, 100000)
            print("Generando la cadena de tamaño {}".format(size))
        else:
            print("Sesión terminada")
            break
```

```

else:
    print("Opcion Invalida")
    continue;

str = generatePal(size)

if size < 20:
    print("La cadena resultante es: " + str + "\n")
else:
    print("La cadena se muestra en el archivo output.txt\n")

```

### 3.5. Código de los métodos utilizados (functions.py)

```

import random
import time

def generatePal(size):
    output = open("output.txt", "w")
    str = "P"
    current_size = 0

    output.write("Tamaño de la cadena a generar: {}\n\n".format(size))
    output.write("→ P\n")

    random.seed(time.time())

    for i in range(size + 1):
        if current_size <= size - 2:
            if random.choice([True, False]):
                str = str.replace("P", "0P0")
                output.write("→ " + str + "\n")
                current_size += 2
            else:
                str = str.replace("P", "1P1")
                output.write("→ " + str + "\n")
                current_size += 2
        elif current_size == size - 1:
            if random.choice([True, False]):
                str = str.replace("P", "0")
                output.write("→ " + str + "\n")
            else:
                str = str.replace("P", "1")
                output.write("→ " + str + "\n")
            break
    else:

```

```

        str = str.replace("P", "")
        output.write("-> " + str + "\n")
        break

    output.close()

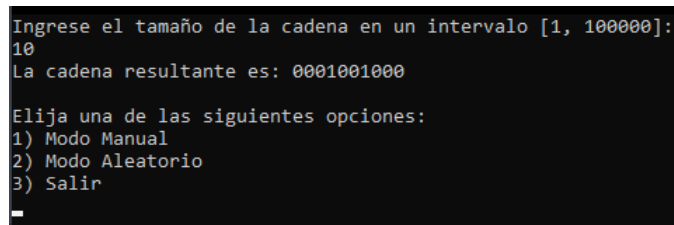
    return str

```

## 4. Resultados

Para probar la implementación del problema planteado, se realizó una prueba de escritorio en donde eligiendo la opción número uno del menú (Modo manual) se introdujeron las siguientes cadenas "000111", "000110" y "000000111111". Para la primer cadena el autómata muestra la animación como se observa en las figuras 1, 2 y 3.

Con el objetivo de corroborar el funcionamiento del programa se realizaron pruebas de escritorio con diferentes longitudes. En la figura 1 y 2 se muestra la salida de la terminal para un  $N = 10$  y el archivo output.txt



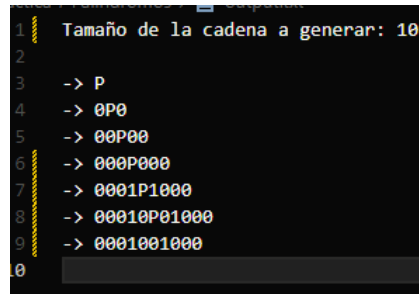
```

Ingrese el tamaño de la cadena en un intervalo [1, 100000]:
10
La cadena resultante es: 0001001000

Elija una de las siguientes opciones:
1) Modo Manual
2) Modo Aleatorio
3) Salir
_

```

Figura 1. Usuario ingresa un valor de  $N = 10$ . Fuente: elaboración propia



```

1  Tamaño de la cadena a generar: 10
2
3  -> P
4  -> 0P0
5  -> 00P00
6  -> 000P000
7  -> 0001P1000
8  -> 00010P01000
9  -> 0001001000
10

```

Figura 2. Contenido del archivo output.txt para un  $N = 10$ . Fuente: Elaboración propia.

En este caso el palíndromo generado fue 0001001000. Posteriormente se eligió el modo aleatorio y los resultados se muestran en la figura 3, donde el programa eligió un valor de  $N = 86168$  y generó un resultado que se muestra en el archivo de texto. En la figura 4 se muestra el tamaño del archivo de salida, ya que para este valor de  $N$  el archivo pesa 1,75GB.

```
Generando la cadena de tamaño 86168...
La cadena se muestra en el archivo output.txt

Elija una de las siguientes opciones:
1) Modo Manual
2) Modo Aleatorio
3) Salir
```

Figura 3. Palíndromo generado para un  $N$  aleatorio. Fuente: Elaboración propia.

```
Generando la cadena de tamaño 86168...
La cadena se muestra en el archivo output.txt

Elija una de las siguientes opciones:
1) Modo Manual
2) Modo Aleatorio
3) Salir
```

Figura 4. Tamaño del archivo de salida. Fuente: Elaboración propia.

## 5. Conclusión

Si bien las gramáticas libres de contexto no se encuentran en el nivel más alto de la clasificación de Chomsky, estas pueden generar incluso parte de un código sintácticamente correcto. Por lo visto en esta práctica y en la anterior, las gramáticas libres de contexto se complementan de buena forma haciendo una herramienta poderosa para la interpretación de código.

## Referencias

- [1] J. Ullman, “Cs154: Introduction to automata and complexity theory,” 2009. [Online]. Available: <http://infolab.stanford.edu/~ullman/ialc/spr10/spr10.html#LECTURE%20NOTES>