

# Práctica 7. “Palíndromos”

Carmona Serrano Ian Carlo  
Ingeniería en Inteligencia Artificial 5BM1  
Teoría de la Computación , ESCOM- IPN

6 de Enero 2024

## 1 Introduction

En esta práctica, se desarrollará un programa para la generación de palíndromos en un lenguaje binario. El programa permitirá construir palíndromos de longitud específica de manera aleatoria, siguiendo reglas gramaticales predefinidas. Se ofrecerán dos opciones: permitir al usuario definir la longitud del palíndromo o generarla automáticamente.

El objetivo es aplicar reglas de producción definidas en una gramática libre de contexto, permitiendo la generación de estas secuencias simétricas mediante reglas recursivas. La práctica explorará el concepto de palíndromos y su construcción a través de reglas gramaticales en un entorno de programación.

## 2 Marco Teórico

### 2.1 Gramáticas Libres de Contexto

Las gramáticas libres de contexto se utilizan en la teoría de la computación para describir lenguajes formales. Estas gramáticas consisten en un conjunto finito de reglas de producción que generan cadenas mediante la sustitución de símbolos no terminales por secuencias de símbolos terminales y no terminales. Son fundamentales en la modelización de estructuras de datos y en el análisis sintáctico de lenguajes de programación.

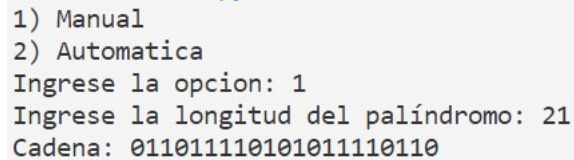
### 2.2 Palíndromos en Cadenas Binarias

Los palíndromos en cadenas binarias son secuencias simétricas que se leen igual en ambas direcciones. Estas estructuras son de interés en diversos campos, como la teoría de la información y la genética, debido a sus propiedades simétricas. En el contexto de las cadenas de ceros y unos, un palíndromo es una secuencia que permanece inalterada al leerla de izquierda a derecha y de derecha a izquierda.

### 3 Desarrollo

Para la implementación del programa que genera palíndromos de un lenguaje binario, se utilizó el lenguaje de programación Python. El programa se estructuró para ofrecer dos opciones al usuario: la generación de palíndromos de manera automática o permitir que el usuario defina la longitud del palíndromo deseado. La generación de los palíndromos se basa en las reglas de producción de una gramática libre de contexto, especificadas como:

$$P \rightarrow eP \rightarrow 0P \rightarrow 1P \rightarrow 0P0P \rightarrow 1P1$$



```
1) Manual
2) Automatica
Ingrese la opcion: 1
Ingrese la longitud del palíndromo: 21
Cadena: 011011110101011110110
```

Figure 1: Consola

Estas reglas permiten construir palíndromos de manera recursiva, comenzando con producciones simples como  $P \rightarrow 0$  o  $P \rightarrow 1$ , y luego agregando elementos simétricos en los extremos mediante las producciones  $P \rightarrow 0P0$  o  $P \rightarrow 1P1$ .

El programa se encarga de generar la cadena correspondiente siguiendo estas reglas y verifica que cumpla con la longitud deseada. La salida del programa se redirige a un archivo de texto donde se muestra el proceso de construcción del palíndromo, indicando la regla seleccionada en cada etapa y la cadena resultante hasta llegar al palíndromo final.

```
Regla aplicada: P -> 1P1
Regla aplicada: P -> 0P0
Regla aplicada: P -> 1P1
Regla aplicada: P -> 1P1
Regla aplicada: P -> 1P1
Regla aplicada: P -> 1P1
Regla aplicada: P -> 0P0
Regla aplicada: P -> 1P1
Regla aplicada: P -> 1P1
Regla aplicada: P -> 0P0
Regla aplicada: P -> 0

Cadena: 011011110101011110110
```

Figure 2: txt

## Código

```
1 import random
2
3 def construir_palindromo(longitud):
4
5     if longitud <= 0:
6         f.write("Regla aplicada: P -> e\n")
7         return ""
8
9     elif longitud == 1:
10        c = random.choice(["0", "1"])
11        f.write("Regla aplicada: P -> ")
12        if c == "0" :
13            f.write("0\n")
14        else:
15            f.write("1\n")
16        return c
17
18    elif longitud == 2:
19        c1 = random.choice(["0", "1"])
20        f.write("Regla aplicada: P -> ")
21        if c1 == "0" :
22            f.write("0\n")
23        else:
```

```

24         f.write("1\n")
25         return c1 + c1
26
27     else:
28         palindromo = ""
29         for _ in range(longitud // 2):
30             caracter = random.choice(["0", "1"])
31             f.write("Regla aplicada: P -> ")
32             if caracter == "0":
33                 f.write("0P0\n")
34             else:
35                 f.write("1P1\n")
36             palindromo = caracter + palindromo + caracter
37
38         if longitud % 2 == 1:
39             c = random.choice(["0", "1"])
40             f.write("Regla aplicada: P -> ")
41             if c == "0":
42                 f.write("0\n")
43             else:
44                 f.write("1\n")
45             palindromo = palindromo[:len(palindromo) // 2] +
c + palindromo[len(palindromo) // 2:]
46
47         return palindromo
48
49
50 print('1) Manual')
51 print('2) Automatica')
52 opcion = input("Ingrese la opcion: ")
53
54 if opcion == "1":
55     longitud = int(input("Ingrese la longitud del
pal ndromo: "))
56 elif opcion == "2":
57     longitud = random.randint(1, 100000)
58
59 f = open("palindromo.txt", "w")
60
61 palindromo = construir_palindromo(longitud)
62
63 f.write("\nCadena: " + palindromo + "\n")
64 print("Cadena: " + palindromo + "\n")
65
66 f.close()

```

Listing 1: Practica 7

## 4 Conclusión

La implementación exitosa del programa para generar palíndromos de un lenguaje binario demuestra la versatilidad de las gramáticas libres de contexto en la construcción de estructuras simétricas. El uso de reglas de producción recursivas permitió generar palíndromos de manera automática, ofreciendo al usuario la posibilidad de definir la longitud deseada. Esta práctica subraya cómo la aplicación de reglas gramaticales puede llevar a la creación sistemática de cadenas con propiedades específicas, como en este caso, la simetría palindrómica en lenguajes formales.

## 5 Bibliografía

Ullman, J.D. (2009-10). "CS154: Introduction to Automata and Complexity Theory". Sitio web: <http://infolab.stanford.edu/~ullman/ialc/spr10/spr10.html>LECTURE