

**Instituto Politécnico Nacional**

ESCUELA SUPERIOR DE COMPUTACIÓN

TEORÍA DE LA COMPUTACIÓN

PRÁCTICA 5

*Automata de Pila*

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. Lenguajes libres de contexto . . . . .	2
2.2. Autómatas de pila . . . . .	2
<b>3. Desarrollo</b>	<b>2</b>
3.1. Programa 5 Autómata de Pila . . . . .	2
3.2. Planteamiento y solución . . . . .	3
3.3. Código . . . . .	3
3.4. Código de la función principal (main.py) . . . . .	3
3.5. Código de los métodos utilizados (functions.py) . . . . .	5
<b>4. Resultados</b>	<b>8</b>
<b>5. Conclusión</b>	<b>10</b>

## 1. Introducción

A lo largo de esta practica se implementará un autómata de pila en el lenguaje de programación python. Autómata que recibirá como entrada cualquier cadena y este comprobará si la cadena pertenece al lenguaje  $0^n 1^n | n \geq 1$ . Para ello es necesario comprender que son los lenguajes libres de contexto y comprender los autómatas de pila.

## 2. Marco Teórico

### 2.1. Lenguajes libres de contexto

En la teoría de la computación, los lenguajes libres de contexto se encuentran en el nivel 2 dentro de la jerarquía de Chomsky. Estos lenguajes resultan particularmente útiles debido a que con ellos podemos representar la sintaxis de los lenguajes de programación por lo que los autómatas de pila (Autómatas utilizados para manejar estos lenguajes) son implementados en intérpretes y compiladores. Es necesario aclarar que estos lenguajes solo son útiles para la sintaxis de los lenguajes y no para su ejecución. Se dice que un lenguaje es libre de contexto si y solo si puede ser reconocido por una gramática libre de contexto [1].

### 2.2. Autómatas de pila

Los autómatas de pila tal como su nombre lo dice, a diferencia de los autómatas finitos no deterministas estos cuentan con una pila, lo que hace posible que puedan recibir lenguajes no regulares. Estos autómatas son complementarios a las gramáticas libres de contexto, estas pueden generar las cadenas y los autómatas de pila recibirlos. La pila con la que cuentan estos autómatas es similar a un contador [2].

## 3. Desarrollo

### 3.1. Programa 5 Autómata de Pila

En esta práctica se desarrollará un programa que implemente un autómata de pila que reciba el lenguaje  $0^n 1^n | n \geq 1$ . Dada una cadena ya sea introducida por el usuario o generada de forma aleatoria que únicamente contengan 0's y 1's, el autómata deberá indicar si esta cadena es parte del lenguaje.

Mediante un menú el usuario indicará de que forma introducir la cadena ya sea manual o aleatoria. En caso de que la cadena contenga 10 caracteres o menos el programa animará el proceso del autómata de pila para analizar la cadena.

### 3.2. Planteamiento y solución

Para implementar la solución a este problema, es necesario obtener la función de transición para el autómata. En este caso son las siguientes:

- $\delta(q, 0, Z_0) = (q, X Z_0)$
- $\delta(q, 0, X) = (q, X X)$
- $\delta(q, 1, X) = (p, \epsilon)$
- $\delta(p, 1, X) = (p, \epsilon)$
- $\delta(p, \epsilon, Z_0) = (f, Z_0)$

Además el programa deberá imprimir las descripciones instantáneas en un archivo de texto junto con la cadena generada o introducida por el usuario. Las descripciones instantáneas deberán tener la siguiente estructura:

$$ID = (q, w, \alpha)$$

Donde:

- $q$  es el estado en el que se encuentra.
- $w$  es la cadena restante.
- $\alpha$  es el contenido de la pila.

Por último para la animación del autómata se empleará la librería de Tkinter de python la cual permite crear interfaces de usuario y animaciones.

### 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)

```
# IPN – Escuela Superior de Computaci n
# Autor: Jimenez Luna Rodrigo Efren
# Grupo: 4CM2
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 una cadena de 0's y 1's menor de 100,000 caracteres:")
            str = input()

            if validateStr(str):
                print("La cadena contiene caracteres diferentes de 0 y 1!\n")

            if len(str) > 100000:
                print("La cadena contiene m s caracteres de los permitidos\n")
                continue
            elif option == 2:
                str = createStr(random.randint(1, 100000))
                print("Evaluando la cadena de tama o {} en el automata...".format(len(str)))
            else:
                print("Sesion terminada")
                break
        else:
            print("Opcion Invalida")
            continue

    if len(str) <= 10:
        if animatePDA(str):
            print("Presione <Enter> para continuar\n")
            input()
            os.system("clear")
        else:
            print("Presione <Enter> para continuar\n")
            input()
            os.system("clear")
    else:
        if pushDownAutomata(str):
            print("La cadena es v lida!\n")
            print("Presione <Enter> para continuar\n")

```

```

        input()
        os.system("clear")
    else:
        print("La cadena no se encuentra en el lenguaje  $\{0^n 1^n \mid n \geq 1\}!$ ")
        print("Presione <Enter> para continuar\n")
        input()
        os.system("clear")

```

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

```

import random
import tkinter as tk
import time

def validateStr(str):
    for c in str:
        if not c in ("1", "0"):
            return 1

    return 0

def createStr(size):
    aproved = random.getrandbits(1)
    string = ""

    if aproved:
        for i in range(size // 2):
            string += "0"
        for i in range(size - size // 2):
            string += "1"
    else:
        for i in range(size):
            string += str(random.getrandbits(1))

    return string

def pushDownAutomata(str):
    output = open("output.txt", "w")
    state = "Q"
    symbol = ""
    count = 0

    output.write("Cadena: " + str + "\n\n")

    if str[0] == "1":

```

```

        output.write("Stack underflow!\n")
        output.close()
        print("Stack underflow!\n")
        return 0

for i, c in enumerate(str):
    if state == "Q":
        if c == "0" and count == 0:
            count += 1
            symbol = "X"
        elif c == "0" and count > 0:
            count += 1
            symbol = "X"
        elif c == "1" and count > 0:
            count -= 1
            symbol = " "
            state = "P"
    elif state == "P":
        if c == "1":
            symbol = " "
            count -= 1
        elif c == "0":
            count = -1
            break

    output.write("(" + state + ", " + c + ", " + symbol + ") \n")

output.close()

if count == 0:
    return 1
else:
    return 0

def animatePDA(str):
    output = open("output.txt", "w")
    state = "Q"
    symbol = ""
    count = 0
    screen_str = str
    stack = "Z"

    output.write("Cadena: " + str + "\n\n")

    root = tk.Tk()

```

```

root.title("Push Down Automata Animation")
root.geometry("600x400")
canvas = tk.Canvas(root, width=600, height=400)
canvas.pack()

canvas.create_rectangle(250, 150, 350, 250, fill="lightgreen")

canvas.create_text(299, 199, text=state, font=("Arial", 14))
canvas.create_text(298, 80, text=screen_str, font=("Arial", 14), anchor=tk.N)
canvas.create_text(298, 300, text=stack, font=("Arial", 14), anchor=tk.NW)

canvas.create_line(300, 150, 300, 100, arrow=tk.LAST, width=2)
canvas.create_line(300, 250, 300, 300, arrow=tk.LAST, width=2)

for c in str:
    root.update()
    time.sleep(1)
    canvas.delete(tk.ALL)

    screen_str = screen_str[1:]

    if state == "Q":
        if c == "0" and count == 0:
            count += 1
            symbol = "X"
        elif c == "0" and count > 0:
            count += 1
            symbol = "X"
        elif c == "1" and count > 0:
            count -= 1
            symbol = " "
            state = "P"
    elif state == "P":
        if c == "1":
            symbol = " "
            count -= 1
        elif c == "0":
            count = -1
            break

    if str[0] == "1":
        canvas.create_text(300, 200, text="Stack Underflow!", font=("Arial", 14),
            output.write("Stack underflow!\n"))
        count = -1
        break

```



```

if state == "Q":
    stack = symbol + stack
elif state == "P":
    stack = stack[1:]
output.write("(" + state + ", " + c + ", " + symbol + ")    \n")

canvas.create_rectangle(250, 150, 350, 250, fill="lightgreen")

canvas.create_text(299, 199, text=state, font=("Arial", 14))
canvas.create_text(298, 80, text=screen_str, font=("Arial", 14), anchor=
canvas.create_text(298, 300, text=stack, font=("Arial", 14), anchor=tk.N

canvas.create_line(300, 150, 300, 100, arrow=tk.LAST, width=2)
canvas.create_line(300, 250, 300, 300, arrow=tk.LAST, width=2)

if count == 0:
    canvas.create_text(300, 30, text="Cadena V lida!", font=("Arial", 14))
else:
    canvas.create_text(300, 30, text="La cadena no pertenece al lenguaje!",

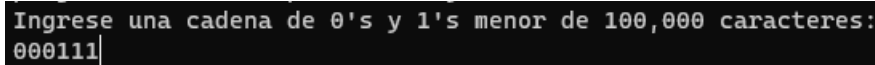
root.mainloop()
output.close()

if count == 0:
    return 1
else:
    return 0

```

## 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.



```

Ingrese una cadena de 0's y 1's menor de 100,000 caracteres:
000111|

```

Figura 1. Cadena ingresada por el usuario. Fuente: elaboración propia

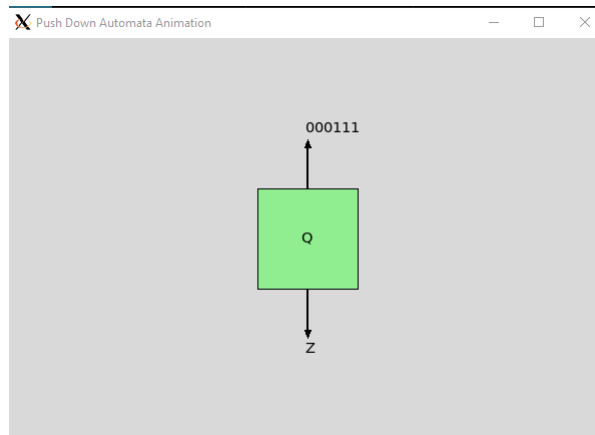


Figura 2. Animación del autómata en Tkinter. Fuente: Elaboración propia.

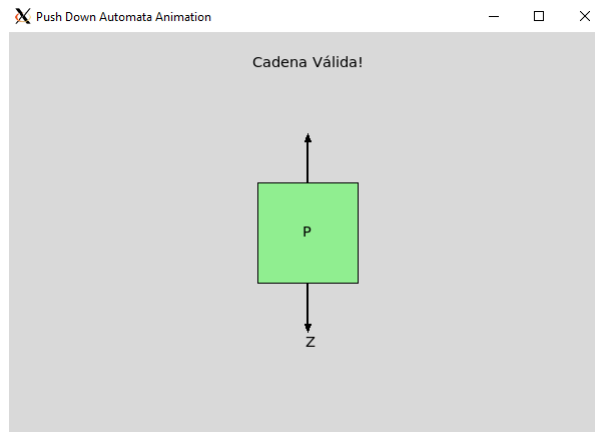


Figura 3. Final de la animación. Fuente: Elaboración propia.

Obsérvese que la cadena es válida y al ser de un tamaño menor a 10 caracteres se muestra la animación. Para el caso de las siguientes dos cadenas, una se muestra su resultado en la figura 4 y la otra se muestra en la figura 5. Para la tercer cadena el programa ya no muestra la animación y simplemente retorna el resultado. Para todos los casos el programa imprime las descripciones instantáneas en un archivo de texto identificado como `.output.txt`, en la figura 6 se muestra el resultado para la tercer cadena.



Figura 4. Resultado al ingresar la cadena 000110. Fuente: elaboración propia.

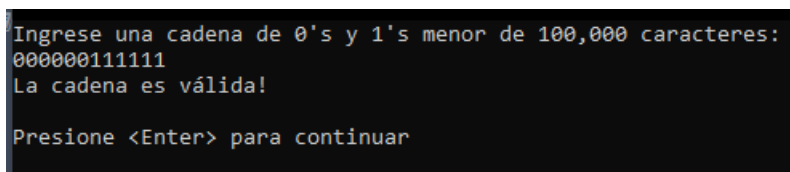


Figura 5. Resultado al ingresar la cadena 000000111111. Fuente: elaboración propia.

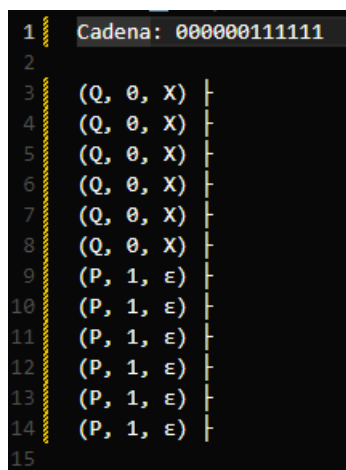


Figura 6. Archivo output.txt al ingresar la cadena 000000111111. Fuente: elaboración propia.

## 5. Conclusión

Para esta práctica se abarco un nuevo tema lo cual son los lenguajes libres de contexto y los autómatas de pila, temas que me parecieron interesantes debido a sus aplicaciones tal como analizar la sintaxis de los lenguajes de programación.

Además durante el desarrollo de esta práctica se implementó el uso de interfaces gráficas las cuáles son útiles para presentar de una forma más amigable un programa es por ello que los programas hoy en día utilizan este método para llegar a una gran cantidad de usuarios.

Además en este caso se mostró una vez más que los autómatas tienen la ventaja de ser algoritmos con complejidad lineal de  $O(n)$  independientemente del tipo de cadenas que aceptan.

## Referencias

- [1] “Lenguaje libre de contexto,” 2023. [Online]. Available: [https://www.ecured.cu/Lenguaje\\_libre\\_de\\_contexto](https://www.ecured.cu/Lenguaje_libre_de_contexto)
- [2] J. J. Fuentes, “Autómata de pila,” 2023. [Online]. Available: <https://www.utm.mx/~jjf/tc/3.1%20AUTOMATA.DE.PILA.pdf>