

# Reporte del segundo parcial

Carlos Tonatihu Barrera Pérez  
Profesor: Genaro Juárez Martínez  
Teoría Computacional  
Grupo: 2CM4

4 de diciembre de 2016

# Índice

<b>1. AFD Palabras que contienen 'web' y/o 'ebay'</b>	<b>2</b>
1.1. Descripción del problema . . . . .	2
1.2. Código . . . . .	3
1.3. Pruebas . . . . .	11
<b>2. Palindromo usando gramática libre de contexto</b>	<b>14</b>
2.1. Descripción del problema . . . . .	14
2.2. Código . . . . .	14
2.3. Pruebas . . . . .	17
<b>3. Gramática libre de contexto no ambigua</b>	<b>19</b>
3.1. Descripción del problema . . . . .	19
3.2. Código . . . . .	19
3.3. Pruebas . . . . .	22
<b>4. Autómata de Pila</b>	<b>24</b>
4.1. Descripción del problema . . . . .	24
4.2. Código . . . . .	25
4.3. Pruebas . . . . .	29
<b>5. Maquina de Turing</b>	<b>36</b>
5.1. Descripción del problema . . . . .	36
5.2. Código . . . . .	37
5.3. Pruebas . . . . .	41

# 1. AFD Palabras que contienen 'web' y/o 'ebay'

## 1.1. Descripción del problema

La elaboración de este programa consistió en convertir un autómata finito no determinista como el de la figura 1 , que reconoce las palabras 'web' y 'ebay', a un autómata finito determinista como el de la figura 2.

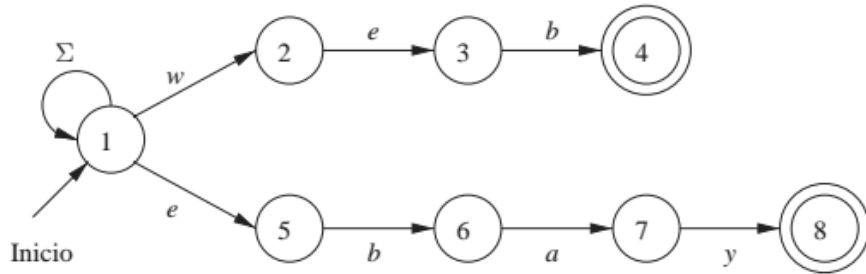


Figura 1: Diagrama de un AFN que busca las palabras 'web' y 'ebay'. [1]

Además, el programa contará con modo manual y automático, en el modo manual el usuario ingresará una cadena y se le mostrará las palabras encontradas, su posición y cuántas fueron encontradas.

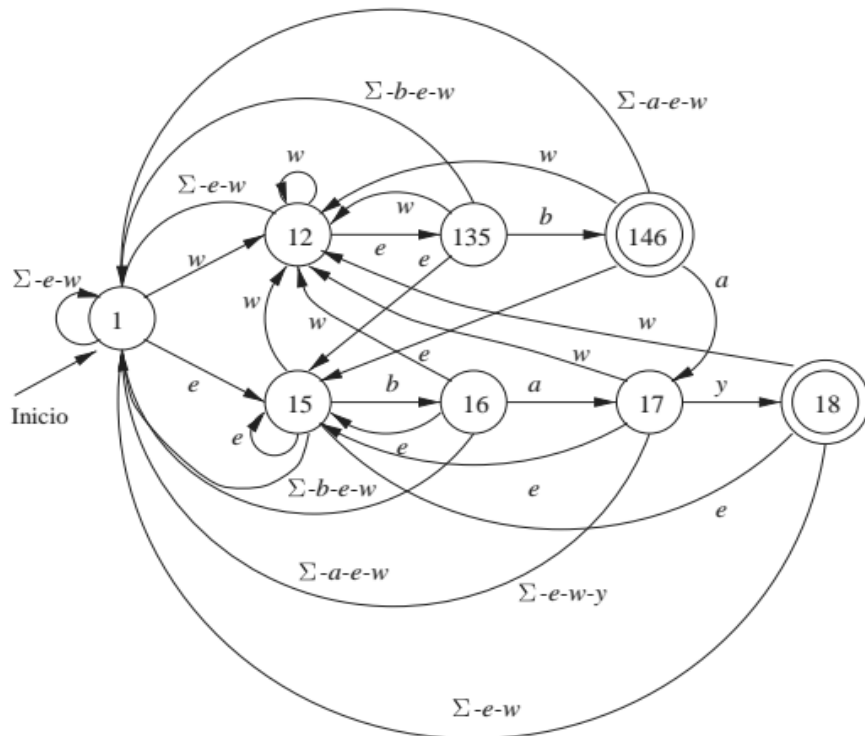


Figura 2: Conversion del AFN a un AFD. [1]

En el modo automático se hará lo mismo pero con el agregado de que el texto se obtendrá de un archivo con extensión '.txt' y se mostrara la linea de texto en la que fue encontrada la palabra. Es importante señalar que todo aquello que no es un símbolo del alfabeto español,  $\Sigma = \{a, b, \dots, z, A, B, \dots, Z\}$ , es tomado como un espacio. Además, debe tener una opción para visualizar el diagrama de la figura 2.

## 1.2. Código

El código fue realizado en Python 3.5.  
Archivo: main\_webay.py

---

```
# main_webay.py
# -*- coding: utf-8 -*-
from __future__ import print_function
from diagrama import Diagrama
from automata import automata

separador = '='*50

def main():
    continuar = True
    while continuar:
        opcion = imprimir_menu()
        if opcion == 1:
            entrada_consola()
        elif opcion == 2:
            entrada_archivo()
        elif opcion == 3:
            ver_diagrama()
        else:
            break
    print('*' * 100)
    opcion = input("Reintentar [s/n]: ")
    if opcion.lower() != 's':
        continuar = False

    print('Saliendo del programa...')

def imprimir_menu():
    print('\n\n%sMenu%s' % (separador, separador))
    print("""
    1.- Entrada en consola
    2.- Ingresar nombre del archivo
    3.- Ver diagrama de estados
    4.- Salir
    """)
    try:
        opcion = int(input("Selecciona una opcion valida: "))
```

```

        return opcion
    except Exception as e:
        print('Error ', e)
    return 0

def entrada_consola():
    texto = input("Escribe el texto: ")
    texto += ' '
    diccionario = {}
    diccionario = automata(texto)
    imprimir_diccionario(diccionario)

def entrada_archivo():
    texto = input('Escribe el nombre del archivo: ')
    i = 1
    try:
        archivo = open(texto, 'r')
    except Exception as e:
        print('Error al abrir archivo: ', e)
        return 0
    diccionario = []
    num_linea = 1
    for linea in archivo:
        diccionario.append(automata(linea))
        num_linea += 1
    while i < num_linea:
        print('\nEn la linea: ', i)
        imprimir_diccionario(diccionario[i-1])
        i += 1
    archivo.close()

def imprimir_diccionario(diccionario):
    print('\nSe encontraron %s web y %s ebays' %(diccionario['num_web'],
        diccionario['num_ebay']))
    print('En las posiciones: ' )
    print('%s para web' %diccionario['web_pos'])
    print('%s para ebay' %diccionario['ebay_pos'])
    print('Las palabras encontradas fueron: %s' %diccionario['palabras'])

def ver_diagrama():
    print('Mostrando diagrama del automata. Cierre la ventana para continuar')
    try:
        diagrama = Diagrama()
        diagrama.master.title('Diagrama del automata webay')
        diagrama.mainloop()
    except Exception as e:
        print("Error", e)

```

main()

---

Archivo: automata\_webay.py

---

```
# automata_webay.py
# -*- coding: utf-8 -*-
from __future__ import print_function

def automata(texto):
    estado = 'B'
    palabra_aux = ''
    num_palabra = 1
    cumple = False
    ebay = 0
    palabra_posicion = 0
    ebay_pos = []
    web_pos = []
    web = 0
    palabras = []
    diccionario = {}
    for simbolo in texto:
        palabra_posicion += 1
        simbolo_aux = simbolo.lower()
        if simbolo == '\n':
            simbolo = '\\n'
        print('-> delta(%s,%s)' % (estado, simbolo), end="\t")
        estado = estados(estado, simbolo_aux)
        if estado == 'G' or estado == 'I':
            cumple = True
        if estado == 'G':
            web += 1
            web_pos.append([palabra_posicion-2, palabra_posicion])
        elif estado == 'I':
            ebay += 1
            ebay_pos.append([palabra_posicion-3, palabra_posicion])

        if ((ord(simbolo_aux) < 123 and ord(simbolo_aux) > 96) or
            ord(simbolo_aux) == 241):
            palabra_aux += simbolo
        else:
            if cumple:
                palabras.append(palabra_aux)
                cumple = False
            palabra_aux = ''
    diccionario = {'num_web': web, 'num_ebay': ebay, 'web_pos': web_pos,
                  'ebay_pos': ebay_pos, 'palabras': palabras}
    return diccionario
```

```

def estados(estado, simbolo):
    if estado == 'B':
        estado = estado_B(simbolo)
    elif estado == 'C':
        estado = estado_C(simbolo)
    elif estado == 'D':
        estado = estado_D(simbolo)
    elif estado == 'E':
        estado = estado_E(simbolo)
    elif estado == 'F':
        estado = estado_F(simbolo)
    elif estado == 'G':
        estado = estado_G(simbolo)
    elif estado == 'H':
        estado = estado_H(simbolo)
    elif estado == 'I':
        estado = estado_I(simbolo)
    else:
        estado = 'A'

    return estado

def estado_B(simbolo):
    if simbolo == 'w':
        return 'C'
    elif simbolo == 'e':
        return 'D'
    return 'B'

def estado_C(simbolo):
    if simbolo == 'w':
        return 'C'
    elif simbolo == 'e':
        return 'E'
    return 'B'

def estado_D(simbolo):
    if simbolo == 'b':
        return 'F'
    elif simbolo == 'e':
        return 'D'
    elif simbolo == 'w':
        return 'C'
    return 'B'

def estado_E(simbolo):
    if simbolo == 'b':
        return 'G'

```

```

        elif simbolo == 'e':
            return 'D'
        elif simbolo == 'w':
            return 'C'
        return 'B'

def estado_F(simbolo):
    if simbolo == 'a':
        return 'H'
    elif simbolo == 'e':
        return 'D'
    elif simbolo == 'w':
        return 'C'
    return 'B'

def estado_G(simbolo):
    if simbolo == 'a':
        return 'H'
    elif simbolo == 'e':
        return 'D'
    elif simbolo == 'w':
        return 'C'
    return 'B'

def estado_H(simbolo):
    if simbolo == 'y':
        return 'I'
    elif simbolo == 'e':
        return 'D'
    elif simbolo == 'w':
        return 'C'
    return 'B'

def estado_I(simbolo):
    if simbolo == 'e':
        return 'D'
    elif simbolo == 'w':
        return 'C'
    return 'B'

```

---

Archivo: diagrama\_webay.py

---

```

# diagrama_webay.py
# -*- coding: utf-8 -*-
from __future__ import print_function
import tkinter as tk

class Diagrama(tk.Frame):

```



```

def __init__(self, master=None):
    super().__init__(master, background='white')
    self.pack(fill=tk.BOTH, expand=tk.YES)
    self.canvas = tk.Canvas(self, bg='white')
    self.canvas.pack(fill=tk.BOTH, expand=1)
    self.dibujarDiagrama()
    self.centrarVentana()

def dibujarDiagrama(self):
    coord_circulo = [100, 205, 150, 255]
    self.dibujarLinea([25, 275, 100, 275])
    self.dibujarCirculo([100, 255, 150, 305])
    self.dibujar_bases()
    self.reflexiva()
    self.normal()
    self.reves()
    self.masflechas()
    self.puntos()
    self.letras()
    self.etiquetas()

def etiquetas(self):
    self.canvas.create_text(75, 245, text='S-e-w')
    self.canvas.create_text(150, 180, text='S-e-w')
    self.canvas.create_text(405, 565, text='S-e-w')
    self.canvas.create_text(260, 465, text='S-a-e-w')
    self.canvas.create_text(515, 90, text='S-a-e-w')
    self.canvas.create_text(180, 400, text='S-b-e-w')
    self.canvas.create_text(300, 90, text='S-b-e-w')
    self.canvas.create_text(380, 530, text='S-e-w-y')
    self.canvas.create_text(500, 510, text='e')
    self.canvas.create_text(400, 460, text='e')
    self.canvas.create_text(300, 420, text='e')
    self.canvas.create_text(210, 325, text='e')
    self.canvas.create_text(180, 335, text='e')
    self.canvas.create_text(300, 170, text='e')
    self.canvas.create_text(355, 215, text='e')
    self.canvas.create_text(480, 230, text='e')
    self.canvas.create_text(600, 310, text='w')
    self.canvas.create_text(500, 320, text='w')
    self.canvas.create_text(450, 100, text='w')
    self.canvas.create_text(310, 140, text='w')
    self.canvas.create_text(220, 130, text='w')
    self.canvas.create_text(165, 215, text='w')
    self.canvas.create_text(220, 235, text='w')
    self.canvas.create_text(280, 255, text='w')
    self.canvas.create_text(470, 170, text='b')
    self.canvas.create_text(305, 370, text='b')

```

```

self.canvas.create_text(605, 370, text='y')

self.canvas.create_text(455, 370, text='a')
self.canvas.create_text(555, 250, text='a')

def letras(self):
    self.canvas.create_text(50, 265, text='Inicio')
    self.canvas.create_text(125, 150+130, text='B')
    self.canvas.create_text(225, 100+80, text='C')
    self.canvas.create_text(385, 100+80, text='E')
    self.canvas.create_text(545, 100+80, text='G')
    self.canvas.create_text(225, 250+130, text='D')
    self.canvas.create_text(385, 250+130, text='F')
    self.canvas.create_text(545, 250+130, text='H')
    self.canvas.create_text(705, 250+130, text='I')

def dibujarCirculo(self, coordenadas):
    self.canvas.create_oval(coordenadas)

def dibujarLinea(self, coord):
    self.canvas.create_line(coord)
    self.canvas.create_oval(coord[2]-3, coord[3]-3, coord[2]+3, coord[3]+3,
        fill='black')

def dibujar_bases(self):
    x, y = 100, 100
    for num in range(3):
        self.dibujarLinea([140, 260, 200, 185])
        self.dibujarCirculo([100+x, 255-y, 150+x, 305-y])
        self.dibujarLinea([250, 180, 360, 180])
        self.dibujarLinea([410, 180, 520, 180])
        self.dibujarLinea([140, 300, 200, 380])
        self.dibujarCirculo([100+x, 255+y, 150+x, 305+y])
        self.dibujarLinea([150+x, 380, 260+x, 380])
        x = x + 160
        if num == 2:
            self.dibujarCirculo([100+x, 255+y, 150+x, 305+y])
            self.dibujarCirculo([105+x, 260+y, 145+x, 300+y])
            self.dibujarCirculo([105+x-160, 260-y, 145+x-160, 300-y])

def reflexiva(self):
    extra = {'start': 20, 'extend': 255}
    self.crear_arco([85, 245, 120, 275], extra)
    extra = {'start': -20, 'extend': 275}
    self.crear_arco([195, 135, 225, 165], extra)
    extra = {'start': -20, 'extend': 275}
    self.crear_arco([195, 335, 225, 365], extra)

```

```

def normal(self):
    extra = {'start': 87, 'extend': 105}
    self.crear_arco([125, 175, 270, 310], extra)
    extra = {'start': 45, 'extend': 95}
    self.crear_arco([225, 150, 390, 250], extra)
    extra = {'start': 25, 'extend': 135}
    self.crear_arco([235, 105, 545, 300], extra)
    extra = {'start': 30, 'extend': 165}
    self.crear_arco([120, 100, 385, 350], extra)
    extra = {'start': 20, 'extend': 150}
    self.crear_arco([120, 25, 545, 450], extra)

def reves(self):
    extra = {'start': -87, 'extend': -105}
    self.crear_arco([125, 250, 270, 390], extra)
    extra = {'start': -45, 'extend': -95}
    self.crear_arco([225, 310, 390, 410], extra)
    extra = {'start': -25, 'extend': -135}
    self.crear_arco([235, 260, 545, 455], extra)
    extra = {'start': -30, 'extend': -160}
    self.crear_arco([125, 205, 385, 455], extra)
    extra = {'start': -24, 'extend': -150}
    self.crear_arco([123, 115, 550, 520], extra)
    extra = {'start': -20, 'extend': -150}
    self.crear_arco([123, 50, 730, 580], extra)
    extra = {'start': -20, 'extend': -145}
    self.crear_arco([235, 200, 730, 500], extra)

def masflechas(self):
    self.dibujarLinea([545, 205, 555, 355])

    self.dibujarLinea([545, 355, 235, 200])
    self.dibujarLinea([700, 355, 235, 200])
    self.dibujarLinea([385, 355, 235, 200])
    self.dibujarLinea([545, 205, 240, 360])
    self.dibujarLinea([385, 205, 240, 360])
    self.dibujarLinea([230, 355, 235, 200])

def crear_arco(self, coord, extra=None):
    self.canvas.create_arc(coord, start=extra['start'],
        extent=extra['extend'], style='arc')

def puntos(self):
    self.canvas.create_oval(123, 250, 129, 257, fill='black')
    self.canvas.create_oval(124, 301, 131, 308, fill='black')
    self.canvas.create_oval(204, 361, 211, 368, fill='black')
    self.canvas.create_oval(242, 389, 249, 396, fill='black')
    self.canvas.create_oval(204, 161, 211, 168, fill='black')

```

```

self.canvas.create_oval(242, 163, 249, 170, fill='black')

def centrarVentana(self):
    ancho, altura = 850, 605
    ancho_pantalla = self.winfo_screenwidth()
    altura_pantalla = self.winfo_screenheight()
    posicion_x = (ancho_pantalla - ancho)/2
    posicion_y = (altura_pantalla - altura)/2
    self.master.geometry(' %dx%d+%d+%d' % (ancho, altura, posicion_x,
        posicion_y))

```

---

### 1.3. Pruebas

Pruebas de las opciones del menú.  
Modo de manual.

```

MINGW32/c:/Users/USER/Documents/tona/git/teoria-computacional/segundo-parcial/webay
=====Menu=====
1.- Entrada en consola
2.- Ingresar nombre del archivo
3.- Ver diagrama de estados
4.- Salir
Selecciona una opcion valida: 1
Escribe el texto: la web es buena porque esta ebay con mucha webwebwebweb la webay es mejor
-> delta(B,l) -> delta(B,a) -> delta(B, ) -> delta(B,w) -> delta(C,e) -> delta(E,b) -> delta
(G, ) -> delta(B,e) -> delta(D,s) -> delta(B, ) -> delta(B,b) -> delta(B,u) -> delta(B,e) -
> delta(D,n) -> delta(B,a) -> delta(B, ) -> delta(B,p) -> delta(B,o) -> delta(B,r) -> delta
(B,q) -> delta(B,u) -> delta(B,e) -> delta(D, ) -> delta(B,e) -> delta(D,s) -> delta(B,t) -
> delta(B,a) -> delta(B, ) -> delta(B,e) -> delta(D,b) -> delta(F,a) -> delta(H,y) -> delta
(I, ) -> delta(B,c) -> delta(B,o) -> delta(B,n) -> delta(B, ) -> delta(B,m) -> delta(B,u) -
> delta(B,c) -> delta(B,h) -> delta(B,a) -> delta(B, ) -> delta(B,w) -> delta(C,e) -> delta
(E,b) -> delta(G,w) -> delta(C,e) -> delta(E,b) -> delta(G,w) -> delta(C,e) -> delta(E,b) -
> delta(G,w) -> delta(C,e) -> delta(E,b) -> delta(G, ) -> delta(B,l) -> delta(B,a) -> delta
(B, ) -> delta(B,w) -> delta(C,e) -> delta(E,b) -> delta(G,a) -> delta(H,y) -> delta(I, ) -
> delta(B,e) -> delta(D,s) -> delta(B, ) -> delta(B,m) -> delta(B,e) -> delta(D,j) -> delta
(B,o) -> delta(B,r) -> delta(B, )
Se encontraron 6 web y 2 ebays
En las posiciones:
[[4, 6], [44, 46], [47, 49], [50, 52], [53, 55], [60, 62]] para web
[[29, 32], [61, 64]] para ebay
Las palabras encontradas fueron: ['web', 'ebay', 'webwebwebweb', 'webay']
*****
Reintentar [s/n]:

```

Figura 3: Historia del autómata y las palabras con 'web' y/o 'ebay'.

## Modo automático

```

=====Menu=====
1.- Entrada en consola
2.- Ingresar nombre del archivo
3.- Ver diagrama de estados
4.- Salir
Selecciona una opcion valida: 2
Escribe el nombre del archivo: texto.txt
-> delta(B,H) -> delta(B,o) -> delta(B,w) -> delta(C,) -> delta(B,t) -> delta(B,o) -> delta
(B,) -> delta(B,C) -> delta(B,r) -> delta(B,e) -> delta(D,a) -> delta(B,t) -> delta(B,e) -
> delta(D,) -> delta(B,a) -> delta(B,) -> delta(B,w) -> delta(C,e) -> delta(E,b) -> delta
(G,s) -> delta(B,i) -> delta(B,t) -> delta(B,e) -> delta(D,) -> delta(B,t) -> delta(B,o) -
> delta(B,) -> delta(B,S) -> delta(B,e) -> delta(D,l) -> delta(B,l) -> delta(B,) -> delta
(B,I) -> delta(B,t) -> delta(B,e) -> delta(D,m) -> delta(B,s) -> delta(B,) -> delta(B,o) -
> delta(B,n) -> delta(B,) -> delta(B,e) -> delta(D,B) -> delta(F,a) -> delta(H,y) -> delta
(I,) -> delta(B,\n) -> delta(B,E) -> delta(D,b) -> delta(F,a) -> delta(H,y) -> delta(I,) -
> delta(B,) -> delta(B,a) -> delta(B,n) -> delta(B,) -> delta(B,o) -> delta(B,n) -> delta
(B,l) -> delta(B,i) -> delta(B,n) -> delta(B,e) -> delta(D,) -> delta(B,a) -> delta(B,u) -
> delta(B,c) -> delta(B,t) -> delta(B,i) -> delta(B,o) -> delta(B,n) -> delta(B,) -> delta
(B,w) -> delta(C,e) -> delta(E,b) -> delta(G,s) -> delta(B,i) -> delta(B,t) -> delta(B,e) -
> delta(D,) -> delta(B,) -> delta(B,l) -> delta(B,e) -> delta(D,t) -> delta(B,s) -> delta
(B,) -> delta(B,b) -> delta(B,u) -> delta(B,y) -> delta(B,e) -> delta(D,r) -> delta(B,s) -
> delta(B,) -> delta(B,a) -> delta(B,n) -> delta(B,d) -> delta(B,) -> delta(B,s) -> delta
(B,e) -> delta(D,l) -> delta(B,l) -> delta(B,e) -> delta(D,r) -> delta(B,s) -> delta(B,) -
> delta(B,e) -> delta(D,x) -> delta(B,c) -> delta(B,h) -> delta(B,a) -> delta(B,n) -> delta
(B,g) -> delta(B,e) -> delta(D,) -> delta(B,g) -> delta(B,o) -> delta(B,o) -> delta(B,d) -
> delta(B,s) -> delta(B,) -> delta(B,t) -> delta(B,h) -> delta(B,r) -> delta(B,o) -> delta
(B,u) -> delta(B,g) -> delta(B,h) -> delta(B,) -> delta(B,t) -> delta(B,h) -> delta(B,e) -
> delta(D,) -> delta(B,I) -> delta(B,n) -> delta(B,t) -> delta(B,e) -> delta(D,r) -> delta

```

Figura 4: Parte de la historia del autómata y las palabras con 'web' y/o 'ebay'.

```

(D,) -> delta(B,\n)
En la línea: 1
Se encontraron 1 web y 1 ebays
En las posiciones:
[[17, 19]] para web
[[42, 45]] para ebay
Las palabras encontradas fueron: ['Website', 'eBay']
En la línea: 2
Se encontraron 1 web y 1 ebays
En las posiciones:
[[25, 27]] para web
[[1, 4]] para ebay
Las palabras encontradas fueron: ['Ebay', 'website']
En la línea: 3
Se encontraron 1 web y 1 ebays
En las posiciones:
[[55, 57]] para web
[[80, 83]] para ebay
Las palabras encontradas fueron: ['website', 'eBay']
En la línea: 4
Se encontraron 1 web y 1 ebays
En las posiciones:
[[13, 15]] para web

```

Figura 5: Parte de la historia del autómata y las palabras con 'web' y/o 'ebay'.

Diagrama.

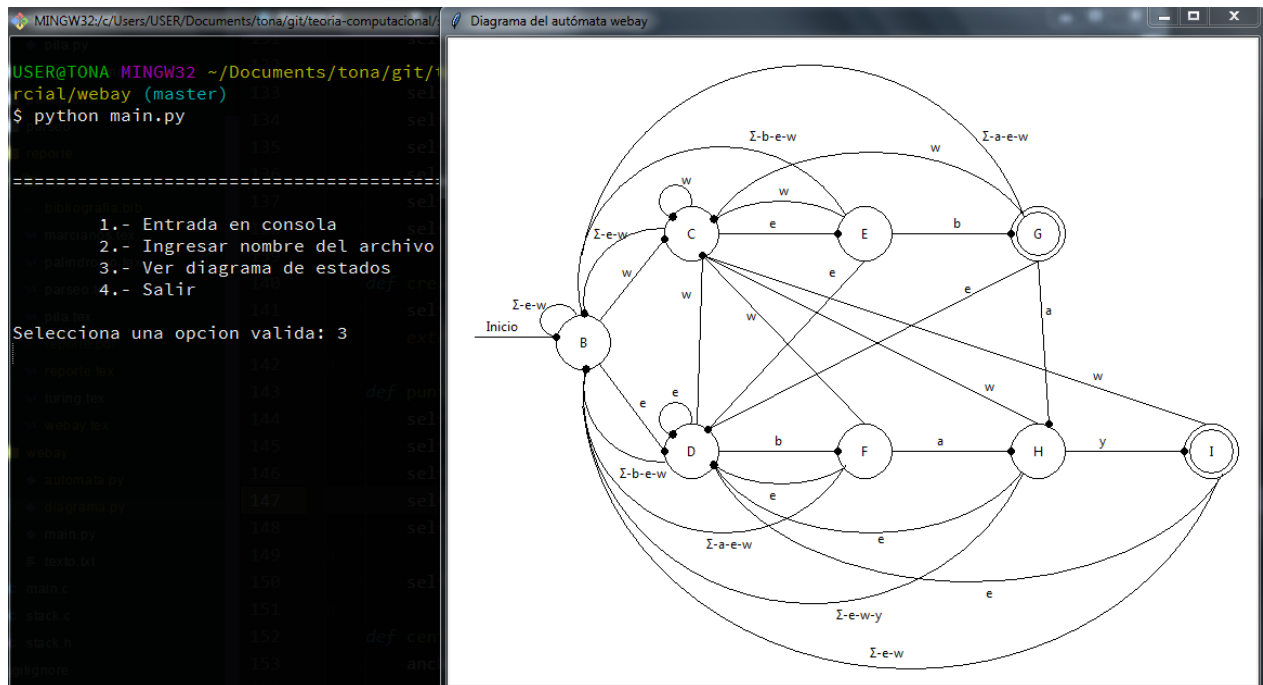


Figura 6: Diagrama de transiciones del autómata 'webay'.

## 2. Palindromo usando gramática libre de contexto

### 2.1. Descripción del problema

Este problema fue parte de la introducción a las gramáticas libres de contexto o Context Free Grammars con el objetivo de generar cadenas que fueran parte del lenguaje de los palíndromos formados por ceros y unos como por ejemplo las cadenas 0011, 1111, 11011 entre otras. Dicho de otra forma, una cadena  $w$  forma parte del  $L_{pal}$  si y sólo si  $w = w^R$ . Para generar el palíndromo se utilizó la siguiente GIC.[1]

$$G_{pal} = (P, 0, 1, A, P)$$

Donde A representa el conjunto de las siguientes 5 producciones.

1.  $P \rightarrow \epsilon$
2.  $P \rightarrow 0$
3.  $P \rightarrow 1$
4.  $P \rightarrow 0P0$
5.  $P \rightarrow 1P1$

El programa cuenta con modo manual y automático en los cuales se introduce la longitud  $0 \leq n \leq 1000$  de la cadena que se desea obtener. Y se muestra el proceso de producción en pantalla y un archivo de texto.

### 2.2. Código

El código fue realizado en Python 3.5.  
Archivo: main\_palindromo.py

---

```
# main_palindromo.py#
# -*- coding: utf-8 -*-
from __future__ import print_function
from palindromo import palindromo
import random

separador = '='*50

def main():
    continuar = True
    while continuar:
        opcion = imprimir_menu()
        if opcion == 1:
            entrada_consola()
        elif opcion == 2:
            entrada_automatico()
        else:
            break
```

```

        print('=' * 100)
        opcion = input("Reintentar [s/n]: ")
        if opcion.lower() != 's':
            continuar = False

    print('Saliendo del programa...')

def imprimir_menu():
    print("""\n\nGramatica libre de contexto Gpal = ({P},{0,1},A,P)
    Donde A es:
    1. P -> e
    2. P -> 0
    3. P -> 1
    4. P -> OP0
    5. P -> 1P1
    """)
    print('\n%sMenu%s' % (separador, separador))
    print("""
    1.- Modo manual
    2.- Modo automatico
    4.- Salir
    """)
    try:
        opcion = int(input("Selecciona una opcion valida: "))
        return opcion
    except Exception as e:
        print('Error ', e)
        return 0

def entrada_automatico():
    longitud = random.randint(0, 1000)
    print("Generando palindromo con longitud = %s" % longitud)
    palindromo(longitud)

def entrada_consola():
    longitud = int(input('Introduce un numero entre 0 y 1000: '))
    if longitud > 1000 or longitud < 0:
        print('Algo salio mal =(')
        return 0
    print("Generando palindromo con longitud = %s" % longitud)
    palindromo(longitud)

main()

```

---

Archivo: palindromo.py

---

```

# palindromo.py
# -*- coding: utf-8 -*-
from __future__ import print_function

```



```

import random
def palindromo(repeticiones):
    archivo = open('palindromo-historia.txt', 'w')
    cadena = 'P'
    base_random = ''
    archivo.write('Longitud = %s\n' %repeticiones)
    archivo.write(cadena + '\n')
    print(cadena)
    if repeticiones % 2 == 1:
        cadena = generar_cadena(cadena, (repeticiones-1)/2, archivo)
        base_random = random.choice(['0', '1'])
    else:
        cadena = generar_cadena(cadena, repeticiones/2, archivo)
    cadena = cadena.replace('P', base_random)
    if base_random == '':
        base_random = 'e'
    archivo.write('Cadena final con base P=%s -> %s\n' %(base_random, cadena))
    print('Cadena final con base P=%s -> %s' %(base_random, cadena))
    archivo.close()

def generar_cadena(cadena, repeticiones, archivo):
    if repeticiones > 0:
        regla = random.choice(['0', '1'])
        cadena = cadena.replace('P', regla+'P'+regla)
        print('%s Regla usada: %sP%s ' %(cadena, regla, regla))
        archivo.write('%s Regla usada: %sP%s \n' %(cadena, regla, regla))
        repeticiones = repeticiones - 1
        cadena = generar_cadena(cadena, repeticiones, archivo)
    return cadena

```

---

## 2.3. Pruebas

Pruebas de las opciones del menú.  
Modo de manual

```
$ python main.py

Gramatica libre de contexto Gpal = ({P},{0,1},A,P)
Donde A es:
1. P -> e
2. P -> 0
3. P -> 1
4. P -> 0P0
5. P -> 1P1

=====
1.- Modo manual
2.- Modo automatico
4.- Salir

Selecciona una opcion valida: 1
Introduce un numero entre 0 y 1000: 2
Generando palindromo con longitud = 2
P
0P0 Regla usada: 0P0
Cadena final con base P=e -> 00
*****
Reintentar [s/n]: s
```

Figura 7: Historia de la generación del palíndromo en consola.

```
1 Longitud = 2
2 P
3 0P0 Regla usada: 0P0
4 Cadena final con base P=e -> 00
5
```

Figura 8: Historia de la generación del palíndromo en archivo.

## Modo de automático

```
Gramatica libre de contexto Gpal = ({P},{0,1},A,P)
Donde A es:
1. P -> e
2. P -> 0
3. P -> 1
4. P -> 0P0
5. P -> 1P1
palindromo py
10
=====Menu=====
1.- Modo manual
2.- Modo automatico
4.- Salir
Selecciona una opcion valida: 2
Generando palindromo con longitud = 12
P
0P0 Regla usada: 0P0
00P00 Regla usada: 0P0
000P000 Regla usada: 0P0
0001P1000 Regla usada: 1P1
00011P11000 Regla usada: 1P1
000110P011000 Regla usada: 0P0
Cadena final con base P=e -> 000110011000
*****
```

Figura 9: Historia de la generación del palíndromo en consola.

```
1 Longitud = 12
2 P
3 0P0 Regla usada: 0P0
4 00P00 Regla usada: 0P0
5 000P000 Regla usada: 0P0
6 0001P1000 Regla usada: 1P1
7 00011P11000 Regla usada: 1P1
8 000110P011000 Regla usada: 0P0
9 Cadena final con base P=e -> 000110011000
```

Figura 10: Historia de la generación del palíndromo en archivo.

## 3. Gramática libre de contexto no ambigua

### 3.1. Descripción del problema

En este problema se debe de construir una única derivación por la izquierda para una cadena de paréntesis balanceada. Usando las siguiente GIC.

$$\begin{aligned} B &\rightarrow (RB|\epsilon \\ R &\rightarrow )|(RR \end{aligned}$$

En donde  $B$  es el símbolo inicial y  $R$  genera cadenas que tienen un paréntesis derecho más que uno izquierdo.

Si necesitamos expandir  $B$  entonces usamos  $B \rightarrow (RB$  si el siguiente símbolo es  $)$  y  $\epsilon$  al final.

Si necesitamos expandir  $R$ , usamos  $R \rightarrow )$  si el siguiente símbolo es  $)$  y  $(RR$  si es  $($  [2] El debe contener un modo manual y automático, en el modo manual el usuario ingresara una cadena de paréntesis y sera procesada hasta llegar a una cadena final que nos indicara si la cadena esta balanceada o no.

Por otro lado, en el modo automático se genera una cadena aleatoria de paréntesis con longitud  $0 \leq n \leq 1000$  que recibirá el mismo tratamiento que en el modo manual.

### 3.2. Código

El código fue realizado en Python 3.5.

Archivo: main\_webay.py

---

```
# main_parseo.py
# -*- coding: utf-8 -*-
from __future__ import print_function
from parseo import proceso
import random

separador = '='*50
def iniciar():
    continuar = True
    while continuar:
        opcion = imprimir_menu()
        if opcion == 1:
            entrada_consola()
        elif opcion == 2:
            ejecutar_random()
        else:
            break
    print('=' * 100)
    opcion = input("Reintentar [s/n]: ")
    if opcion.lower() != 's':
```

```

        continuar = False

    print('Saliendo del programa...')

def imprimir_menu():
    print('\n\n%sMenu%s' % (separador, separador))
    print("""
        1.- Entrada en consola (Manual)
        2.- Aleatorio (Automatico)
        3.- Salir
    """)
    try:
        opcion = int(input("Selecciona una opcion valida: "))
        return opcion
    except Exception as e:
        print('Error ', e)
        return 0

def entrada_consola():
    texto = input("Escribe la cadena de parentesis: ")
    proceso(texto)

def ejecutar_random():
    i = 0
    longitud_random = random.randint(1, 20)
    cadena = ''
    while i < longitud_random:
        cadena += random.choice(['(', ')'])
        i += 1

    print("La cadena es: ", cadena)
    proceso(cadena)

iniciar()

```

---

Archivo: parseo.py

---

```

# parseo.py
# -*- coding: utf-8 -*-
from __future__ import print_function

def proceso(cadena):
    derivacion = 'B'
    archivo = open('historia-parentesis.txt', 'w')
    cadena += ' '
    print('Cadena: ', cadena)
    archivo.write('Cadena: %s'%cadena)
    continuar = True

```

```

for simbolo in cadena:
    if not continuar:
        break
    print(derivacion, end = '\t')
    archivo.write('\n%s ' %derivacion)
    for paso in derivacion:
        if paso == 'B':
            if simbolo == '(':
                derivacion = derivacion.replace('B', '(RB', 1)
                print('Regla usada: B->(RB')
                archivo.write('Regla usada: B->(RB')
                break
            elif simbolo == ' ':
                derivacion = derivacion.replace('B', '', 1)
                print('Regla usada: B->e')
                archivo.write('Regla usada: B->e')
                break
            elif simbolo == ')':
                continuar = False
                break
        elif paso == 'R':
            if simbolo == ')':
                derivacion = derivacion.replace('R', ')', 1)
                print('Regla usada: R->e')
                archivo.write('Regla usada: R->e')
                break
            elif simbolo == '(':
                derivacion = derivacion.replace('R', '(RR', 1)
                print('Regla usada: R->(RR')
                archivo.write('Regla usada: R->(RR')
                break
            elif simbolo == ' ':
                continuar = False
                break
    archivo.write('\nFinal: %s' %derivacion)
    print('\nFinal: ', derivacion)
    if paso == 'B' and simbolo == ' ':
        print('\nCadena balanceada')
        archivo.write('\nCadena balanceada')
    else:
        print('\nCadena no balanceada')
        archivo.write('\nCadena no balanceada')
archivo.close()

```

---

Pruebas de las opciones del menú.  
Modo de consola.

```
=====Menu=====
recibí 1.- Entrada en consola (Manual) ((( )))
parcial 2.- Aleatorio (Automatico)ada: B->(RB
abierta 3.- Salir (RB Regla usada: R->)
Selecciona una opcion valida: 1
Escribe la cadena de parentesis: ()()()()()()()()
Cadena: ()()()()()()()() Regla usada: B->(RB
B Regla usada: B->(RB RB Regla usada: R->)
(RB Regla usada: R->) ()B Regla usada: B->(RB
()B Regla usada: B->(RB ()(RB Regla usada: R->(RR
()B Regla usada: B->(RB()((RRB Regla usada: R->)
()()RB Regla usada: R->) ()()RB Regla usada: R->)
()()B Regla usada: B->(RB ()()B Regla usada: B->(RB
()()RB Regla usada: R->(RR ()()B Regla usada: R->(R
()()()((RRB Regla usada: R->) ((RB Regla usada: R->
()()()()RB Regla usada: R->) (RRB Regla usada: R->
()()()()B Regla usada: B->(RB (RRB Regla usada: R
()()()()RB Regla usada: R->(RR (RRB Regla usada: R
()()()()()((RRB Regla usada: R->) B Regla usada: R
()()()()()((RRB Regla usada: R->) Regla usada:
()()()()()()RB Regla usada: R->(RR Regla usada:
()()()()()()()RB Regla usada: R->) Regla usada:
()()()()()()()B Regla usada: B->e()())
Cadena balanceada
Final: ()()()()()()()()
Cadena balanceada
```

Figura 11: Historia de las derivaciones en consola.

```

1 Cadena: ()()()()()()()() -H~
2 B Regla usada: B->(RBH~
3 (RB Regla usada: R->)H~
4 ()B Regla usada: B->(RBH~
5 ()(RB Regla usada: R->)H~
6 ()()B Regla usada: B->(RBH~
7 ()()()RB Regla usada: R->)H~
8 ()()()()B Regla usada: B->(RBH~
9 ()()()()RB Regla usada: R->(RRH~
10 ()()()()()RRB Regla usada: R->)H~
11 ()()()()()()RB Regla usada: R->)H~
12 ()()()()()()()B Regla usada: B->(RBH~
13 ()()()()()()()RB Regla usada: R->(RRH~
14 ()()()()()()()()RRB Regla usada: R->(RRH~
15 ()()()()()()()()()RRRB Regla usada: R->)H~
16 ()()()()()()()()()()RRB Regla usada: R->)H~
17 ()()()()()()()()()()()RB Regla usada: R->(RRH~
18 ()()()()()()()()()()()()RRB Regla usada: R->)H~
19 ()()()()()()()()()()()()()RB Regla usada: R->)H~
20 ()()()()()()()()()()()()()()B Regla usada: B->e
21 Final: ()()()()()()()()()()H~
22 Cadena balanceada

```

Figura 12: Parte de la historia de las derivaciones en archivo.

## Modo automático

```
$ python main.py

=====Menu=====

    1.- Entrada en consola (Manual)
    2.- Aleatorio (Automatico)
    3.- Salir

Selecciona una opcion valida: 2
La cadena es:  ())))(())(())
Cadena:  ())))(())(())(
B          Regla usada: B->(RB
(RB        Regla usada: R->)
()B
Final:  (())B
Cadena no balanceada
```

Figura 13: Modo automático desde consola.

```
1  Cadena: ())))(())(())(
2  B      Regla usada: B->(RB
3  (RB    Regla usada: R->)
4  ()B
5  Final: (())B
6  Cadena no balanceada
```

Figura 14: Historia del modo automático en archivo.



## 4. Autómata de Pila

### 4.1. Descripción del problema

En este programa se desarrollo un autómata de pila (vease figura 15) que aceptara el lenguaje de  $\{0^n 1^n \mid n \geq 1\}$ . El programa tiene un modo manual y automático en ambos modos se evalúa una cadena de ceros y unos de longitud  $1 \leq n \leq 1000$  y se muestra si la cadena es valida o no junto con los pasos para llegar a este resultado que se mostraran en archivo y consola. Además, se puede observar la animación de este proceso como el de la figura ?? si así se desea, esta acción remplaza a la historia del autómata en consola pero no en archivo.

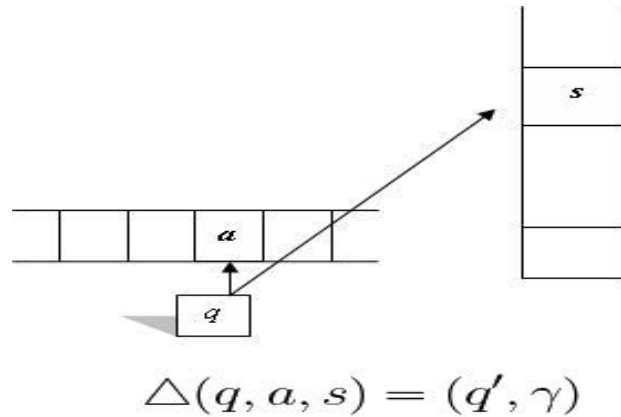


Figura 15: Representación de un autómata de Pila.

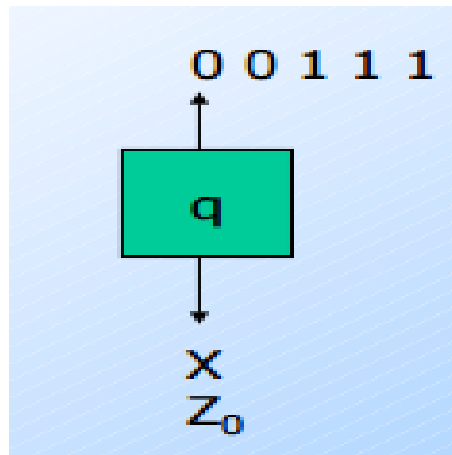


Figura 16: Representación de un autómata de Pila.

## 4.2. Código

El código fue realizado en Python 3.5.  
Archivo: main\_pila.py

---

```
#main_pila.py
# -*- coding: utf-8 -*-
from __future__ import print_function
from pila import automata
import random

separador = '='*50
def iniciar():
    continuar = True
    while continuar:
        opcion = imprimir_menu()
        if opcion == 1:
            entrada_consola()
        elif opcion == 2:
            ejecutar_random()
        else:
            break # Sal del programa
    print('=' * 100)
    opcion = input("Reintentar [s/n]: ")
    if opcion.lower() != 's':
        continuar = False

    print('Saliendo del programa...')

def imprimir_menu():
    print('\n\n%sMenu%s' % (separador, separador))
    print("""
        1.- Entrada en consola (Manual)
        2.- Aleatorio (Automatico)
        3.- Salir
    """)
    try:
        opcion = int(input("Selecciona una opcion valida: "))
        return opcion
    except Exception as e:
        print('Error ', e)
        return 0

def entrada_consola():
    texto = input("Escribe el numero binario: ")
    animacion = ver_animacion()
    automata(texto, animacion)
```

```

def ver_animacion():
    opcion = input("Ver animacion [s/n]: ")
    if opcion == 's':
        return True
    else:
        return False

def ejecutar_random():
    i = 0
    longitud_random = random.randint(1, 10)
    numero_binario = ''
    while i < longitud_random:
        numero_binario += random.choice(['0', '1'])
        i += 1

    print("El numero aleatorio es: ", numero_binario)
    animacion = ver_animacion()
    automata(numero_binario, animacion)

iniciar()

```

---

Archivo: automata\_pila.py

---

```

#automata_pila.py
# -*- coding: utf-8 -*-
from __future__ import print_function
import time

class Pila(object):
    def __init__(self):
        self.altura = -1
        self.elementos = []

    def vacio(self):
        if self.altura == -1:
            return True
        else:
            return False

    def sacar(self):
        if self.vacio():
            return 'e'
        else:
            valor = self.elementos[self.altura]
            self.altura -= 1
            return valor

    def meter(self, elemento):

```

```

        self.altura += 1
        self.elementos[self.altura:] = [elemento]

def mostrar(self):
    i = self.altura
    cadena = ''
    while(i>-1):
        cadena += self.elementos[i]
        i -= 1
    return cadena

def automata(cadena, ver_animacion):
    pila = Pila()
    archivo = open('historia-pila.txt', 'w')
    pila.meter('Zo')
    estado = 'q'
    cadena_aux = cadena
    cadena = cadena + ' '
    archivo.write('La cadena es: ' + cadena_aux + '\n')
    for simbolo in cadena:
        if cadena_aux == '':
            cadena_aux = 'e'
        if ver_animacion:
            time.sleep(1)
            pintar(estado, cadena_aux, pila)
        else:
            print('(%s, %s, %s)' %(estado, cadena_aux, pila.mostrar()), end='
|- ')
        archivo.write('(%s, %s, %s) |- ' %(estado, cadena_aux,
            pila.mostrar()))
        if estado == 'q':
            if simbolo == '0':
                pila.meter('X')
            elif simbolo == '1':
                if pila.sacar() == 'Zo':
                    pila.meter('Zo')
                    break
            estado = 'p'
        else:
            estado = 'f'
            break
    elif estado == 'p':
        if simbolo == '1':
            if pila.sacar() == 'Zo':
                estado = 'f'
                pila.meter('Zo')
                break
        elif simbolo == '0':

```

```

        pila.meter('X')
        cadena_aux = cadena_aux[1:]
        break
    elif simbolo == ' ':
        estado = 'f'
        if pila.mostrar() == 'Zo':
            break
        cadena_aux = cadena_aux[1:]

if cadena_aux == '':
    cadena_aux = 'e'
if ver_animacion:
    time.sleep(1)
    pintar(estado, cadena_aux, pila)
else:
    print('(%s, %s, %s)' %(estado, cadena_aux, pila.mostrar()))
    print('\n')
archivo.write('(%s, %s, %s)' %(estado, cadena_aux, pila.mostrar()))
if (pila.mostrar() == 'Zo') and cadena_aux == 'e':
    print('\nCadena valida')
    archivo.write('\nCadena valida')
else:
    print('\nCadena invalida')
    archivo.write('\nCadena invalida')
archivo.close()

def pintar(estado, cadena_aux, stack):
    pila = 'Zo'
    if stack.mostrar() != '':
        pila = stack.mostrar()

    print('\n')
    print('\n')
    print('\n')
    print('\n')
    print('\n')
    print('    %s' %cadena_aux)
    print('    ^')
    print('    |')
    print('    |')
    print('    |')
    print('-----')
    print('|          |')
    print('|    %s    |' %estado)
    print('|          |')
    print('-----')
    print('    |')
    print('    |')

```

```
print('    |')
print('    v')
print('    %s' %pila)
print('\n')
print('\n')
print('\n')
print('\n')
print('\n')
```

---

### 4.3. Pruebas

Pruebas de las opciones del menú.  
Modo de manual.

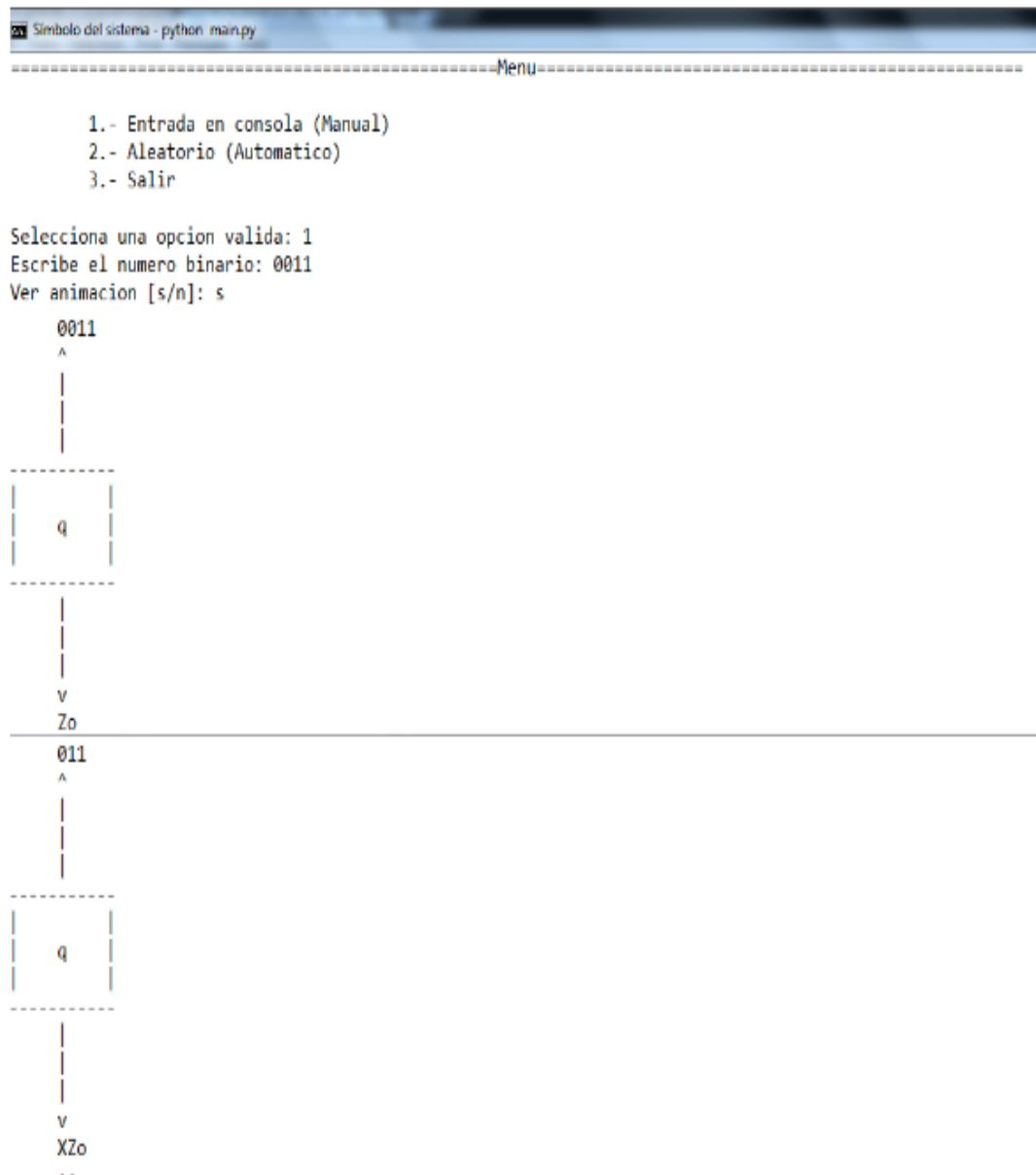


Figura 17: Historia del Autómata de Pila en animación 1

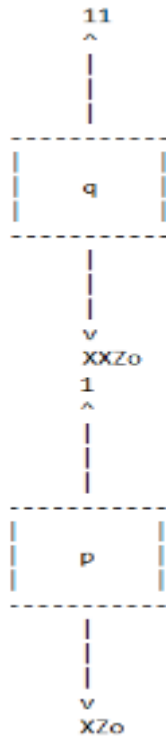


Figura 18: Historia del Autómata de Pila en animación 2



Figura 19: Historia del Autómata de Pila en animación 3



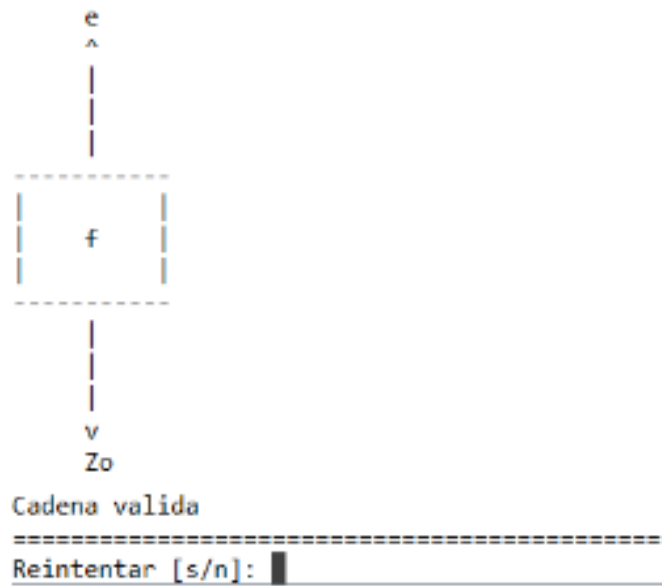


Figura 20: Historia del Autómata de Pila en animación 4

```

1  La cadena es: 0011|_
2  (q, 0011, Zo) |- (q, 011, XZo) |- (q, 11, XXZo) |- (p, 1, XZo) |- (p, e, Zo) |- (f, e, Zo)
3  Cadena valida

```

Figura 21: Historia del Autómata de Pila.

Modo automático

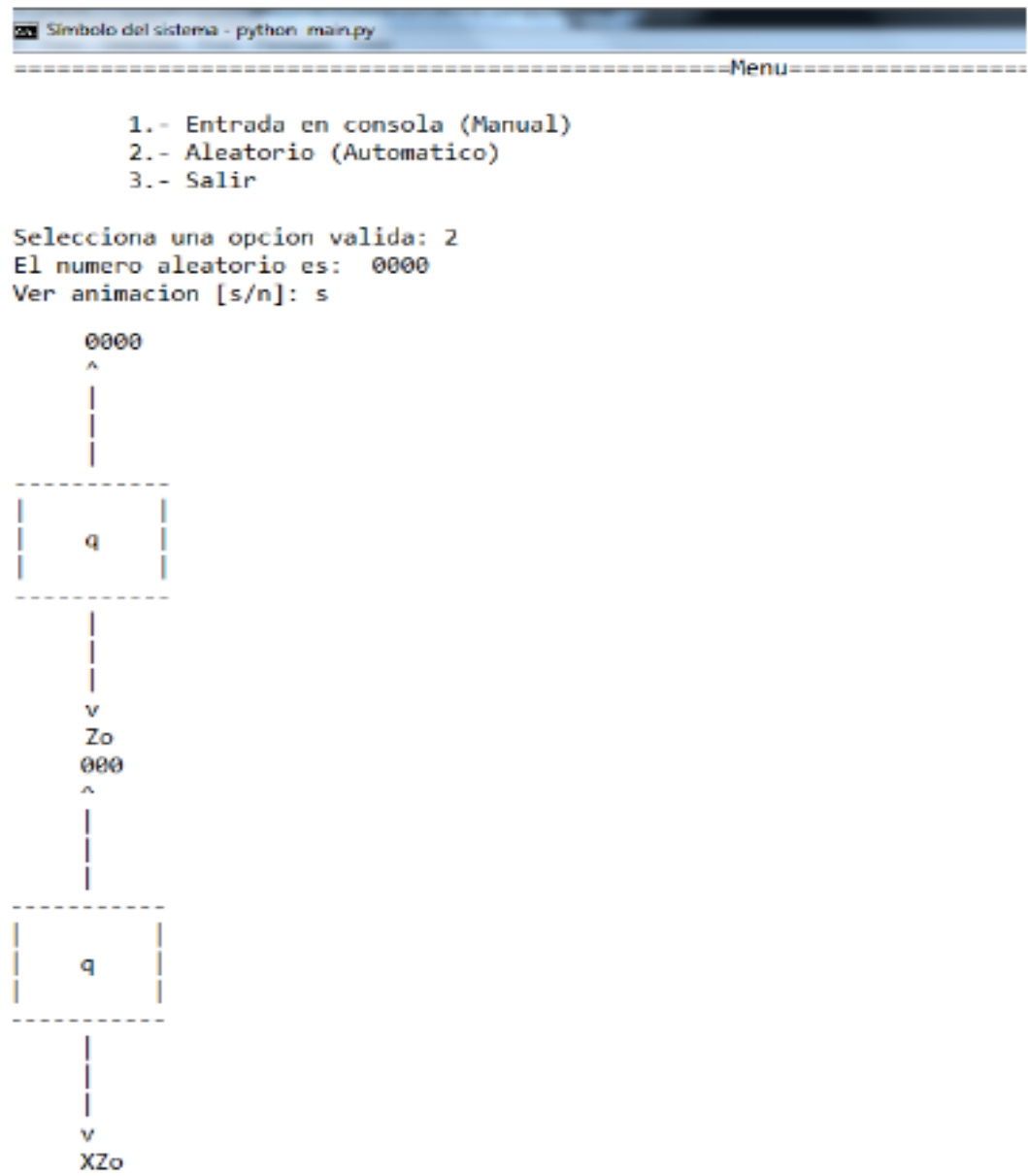


Figura 22: Historia del Autómata de Pila en animación 1

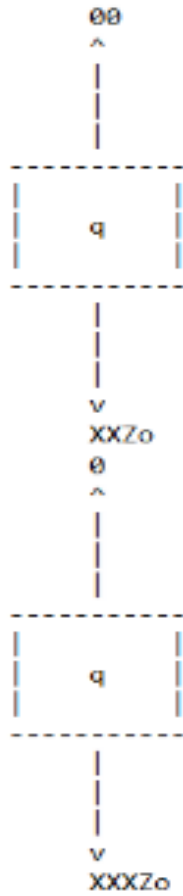


Figura 23: Historia del Autómata de Pila en animación 2

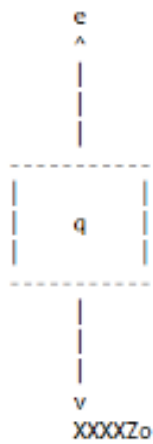


Figura 24: Historia del Autómata de Pila en animación 3

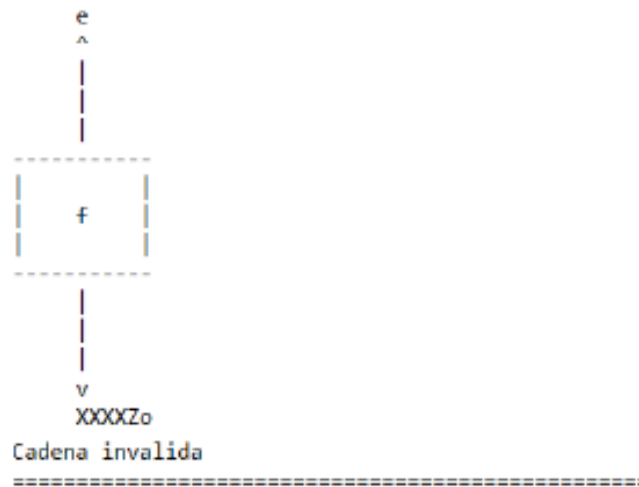


Figura 25: Historia del Autómata de Pila en animación 4

```

1  La cadena es: 0000 |
2  (q, 0000, Zo) |- (q, 000, XZo) |- (q, 00, XXZo) |- (q, 0, XXXZo) |- (q, e, XXXXZo) |- (f, e, XXXXZo)
3  Cadena invalida

```

Figura 26: Historia del Autómata de Pila en archivo.

## 5. Maquina de Turing

### 5.1. Descripción del problema

Para este programa se desarrollo una maquina de Turing que acepte el lenguaje  $\{0^n 1^n \mid n \geq 1\}$ . Que de manera formal se define como lo siguiente:

$$M = (q_0, q_1, q_2, q_3, q_4, 0, 1, 0, 1, X, Y, B, \delta, q_0, B, q_4)$$

donde  $\delta$  se especifica como en la siguiente tabla:

Símbolo					
Estado	0	1	X	Y	B
$q_0$	$(q_0, X, R)$	-	-	$(q_0, X, R)$	-
$q_1$	$(q_0, X, R)$	$(q_0, X, R)$	-	$(q_0, X, R)$	-
$q_2$	$(q_0, X, R)$	-	$(q_0, X, R)$	$(q_0, X, R)$	-
$q_3$	-	-	-	$(q_0, X, R)$	$(q_0, X, R)$
$q_4$	-	-	-	-	-

El funcionamiento de esta maquina se puede entender mejor con el diagrama de la figura 27.

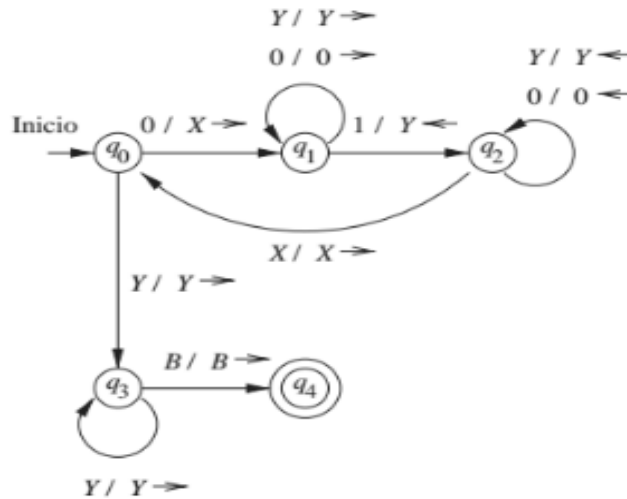


Figura 27: Diagrama de transiciones de una Maquina de Turing que acepta las cadenas  $0^n 1^n$ . [1]

El programa debe de contar con un modo manual y automático, en ambos modos se ingresara una cadena de ceros y unos que sera trabajada por la maquina de Turing a través de una unidad de control y una banda (véase la figura 28) la cual sera la cadena que se ingrese y se mostrara si es una cadena valida junto al proceso que se realizo para llegar a dicho resultado.

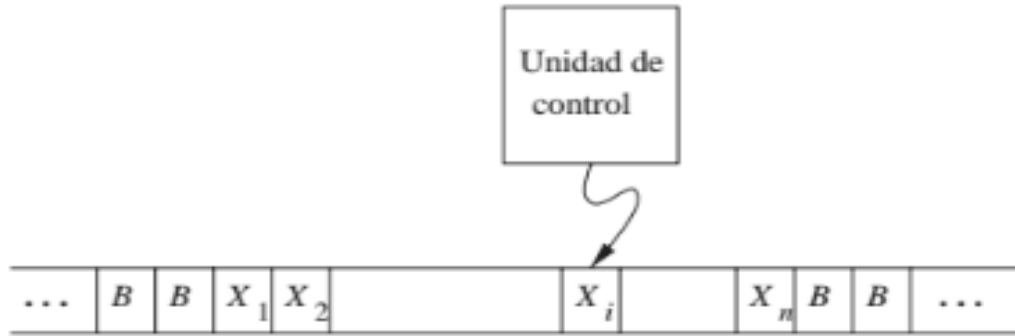


Figura 28: Representación de una Máquina de Turing. [1]

## 5.2. Código

El código fue realizado en Python 3.5.

Archivo: main\_turing.py

---

```
# main_turing.py
# -*- coding: utf-8 -*-
from __future__ import print_function
from metodos import maquina
import random

separador = '='*50
def iniciar():
    continuar = True
    while continuar:
        opcion = imprimir_menu()
        if opcion == 1:
            entrada_consola()
        elif opcion == 2:
            ejecutar_random()
        else:
            break # Sal del programa
    print('=' * 100)
    opcion = input("Reintentar [s/n]: ")
    if opcion.lower() != 's':
        continuar = False

    print('Saliendo del programa...')

def imprimir_menu():
    print('\n\n%sMenu%s' % (separador, separador))
    print("""
    1.- Entrada en consola (Manual)
```

```

        2.- Aleatorio (Automatico)
        3.- Salir
    """)
    try:
        opcion = int(input("Selecciona una opcion valida: "))
        return opcion
    except Exception as e:
        print('Error ', e)
        return 0

def entrada_consola():
    texto = input("Escribe un numero binario: ")
    print('\n Historia de la maquina de Turing')
    maquina(texto)

def ejecutar_random():
    i = 0
    longitud_random = random.randint(1, 1000)
    binario = ''
    while i < longitud_random:
        binario += random.choice(['0', '1'])
        i += 1

    print("La cadena es: ", binario)
    print('\n Historia de la maquina de Turing')
    maquina(binario)

iniciar()

```

---

Archivo: maquina\_turing.py

---

```

# maquina de turing.py
# -*- coding: utf-8 -*-
from __future__ import print_function

def maquina(cadena):
    continuar = True
    archivo = open('turing-historia.txt', 'w')
    i = 0
    estado = 0
    cadena_aux = list(cadena)
    cadena_final = ''
    archivo.write('La cadena es: %s \n' %cadena)
    while continuar:
        try:
            simbolo = cadena_aux[i]
        except Exception as e:
            cadena_aux.append('B')

```

```

        simbolo = cadena_aux[i]
        cadena_final = imprimir_secuencia(estado, cadena_aux, i)
        print(cadena_final, end = '')
        archivo.write(cadena_final)
        resultado = funcion_transicion(estado, simbolo)
        if len(resultado) == 0:
            break
        estado = resultado[0]
        cadena_aux[i] = resultado[1]
        if resultado[2] == 'R':
            i += 1
        elif resultado[2] == 'L':
            i -= 1
        print(' |- ', end='')
        archivo.write(' |- ')
    print('\n')
    if estado == 4:
        print('Cadena valida')
        archivo.write('\n\nCadena valida')
    else:
        print('Cadena invalida')
        archivo.write('\n\nCadena invalida')
    archivo.close()

def imprimir_secuencia(estado, cadena, indice):
    cadena_aux = ''
    i = 0
    while i < len(cadena):
        if i == indice:
            cadena_aux += '(q' + str(estado) + ' '
            cadena_aux += cadena[i]
            i += 1
    return cadena_aux

def funcion_transicion(estado, simbolo):
    if estado == 0:
        return estado_cero(simbolo)
    elif estado == 1:
        return estado_uno(simbolo)
    elif estado == 2:
        return estado_dos(simbolo)
    elif estado == 3:
        return estado_tres(simbolo)
    elif estado == 4:
        return estado_cuatro(simbolo)

def estado_cero(simbolo):
    if simbolo == '0':

```



```

        return [1, 'X', 'R']
    elif simbolo == 'Y':
        return [3, 'Y', 'R']
    return []

def estado_uno(simbolo):
    if simbolo == '0':
        return [1, '0', 'R']
    elif simbolo == '1':
        return [2, 'Y', 'L']
    elif simbolo == 'Y':
        return [1, 'Y', 'R']
    return []

def estado_dos(simbolo):
    if simbolo == '0':
        return [2, '0', 'L']
    elif simbolo == 'X':
        return [0, 'X', 'R']
    elif simbolo == 'Y':
        return [2, 'Y', 'L']
    return []


def estado_tres(simbolo):
    if simbolo == 'Y':
        return [3, 'Y', 'R']
    elif simbolo == 'B':
        return [4, 'B', 'R']
    return []

def estado_cuatro(simbolo):
    return []

```

---

Pruebas de las opciones del menú.  
Modo de manual.

```
=====Menu=====
Mas de 70 destinos
La manera más
comoda de viajar,
conocélos
a través de

Compre Aquí
sus boletos.

Selecciona una opcion valida: 1
Escribe un numero binario: 00001111

Historia de la maquina de Turing
(q0)00001111 |- X(q1)0001111 |- X0(q1)001111 |- X00(q1)01111 |- X000(q1)1111 |- X00(q2)0Y111 |- X0(q2)00Y111 |- X(q2)0
00Y111 |- (q2)X000Y111 |- X(q0)000Y111 |- XX(q1)00Y111 |- XX0(q1)0Y111 |- XX00(q1)Y111 |- XX00Y(q1)111 |- XX00(q2)YY11
|- XX0(q2)0YY11 |- XX(q2)00Y11 |- X(q2)X00Y11 |- XX(q0)00Y11 |- XXX(q1)0Y111 |- XXX0(q1)Y111 |- XXX0Y(q1)Y11 |- XX
X0YY(q1)11 |- XXX0Y(q2)YY1 |- XXX0(q2)YYY1 |- XXX(q2)0Y111 |- XX(q2)X0Y11 |- XXX(q0)0Y111 |- XXXX(q1)Y111 |- XXXXY(q1
)YY1 |- XXXXY(q1)Y1 |- XXXXY(q1)Y1 |- XXXXY(q1)Y1 |- XXXXY(q2)YY |- XXXXY(q2)YY1 |- XXXX(q0)Y111 |- XXXX(q2)XY111 |- XXXX(q0)Y111
|- XXXXY(q3)YY1 |- XXXXY(q3)YY |- XXXXY(q3)Y |- XXXXY(q3)B |- XXXXY(q3)B
(q0)001100101111 |- X(q1)01100101111 |- X0(q1)1100101111 |- X(q2)0Y100101111 |- (q2)X0Y100101111 |-
Cadena valida X(q1)Y100101111 |- XX(q1)Y100101111 |- XXY(q1)100101111 |- XX(q2)YY00101111 |- X(q2)XY00101111 |-
```

Figura 29: Parte de la historia de la Maquina de Turing en consola.

```

1 La cadena es: 00001111
2 (q0)00001111 | - X(q1)0001111 | - X0(q1)001111 | - X00(q1)01111 | - X000(q1)1111 | - X00(q2)0Y111 | -
• X0(q2)00Y111 | - X(q2)000Y111 | - (q2)X000Y111 | - X(q0)000Y111 | - XX(q1)00Y111 | - XX0(q1)0Y111 | -
• XX00(q1)Y111 | - XX00Y(q1)111 | - XX00(q2)YY11 | - XX0(q2)0YY11 | - XX(q2)00YY11 | - X(q2)X00YY11 | -
• XX(q0)00YY11 | - XXX(q1)0YY11 | - XXX0(q1)YY11 | - XXX0Y(q1)Y11 | - XXX0YY(q1)11 | - XXX0Y(q2)YY1 | -
• XXX0(q2)YYY1 | - XXX(q2)0YYY1 | - XX(q2)X0YYY1 | - XXX(q0)0YYY1 | - XXXX(q1)YYY1 | - XXXXY(q1)YY1 | -
• XXXXY(q1)Y1 | - XXXXY(q1)1 | - XXXXY(q2)YY | - XXXXY(q2)YYY | - XXXX(q2)YYYY | - XXX(q2)XYYYY | -
• XXXX(q0)YYYY | - XXXXY(q3)YYY | - XXXXY(q3)YY | - XXXXY(q3)Y | - XXXXY(q3)B | - XXXXY(q3)B(q4)B
3
4 Cadena valida

```

Figura 30: Historia de la Maquina de Turing en archivo.

## Modo automático

```
=====Menu=====
1.- Entrada en consola (Manual)
2.- Aleatorio (Automatico)
3.- Salir

Selecciona una opcion valida: 2
La cadena es: 001100101111

Historia de la maquina de Turing
(q0)001100101111 |- X(q1)01100101111 |- X0(q1)1100101111 |- X(q2)0Y100101111 |- (q2)X0Y100101111 |- X(q0)0Y100101111 |-
- XX(q1)Y100101111 |- XX(q1)Y100101111 |- XX(q2)YY00101111 |- X(q2)XY00101111 |- XX(q0)YY00101111 |- XX(q3)Y00101111
|- XXYY(q3)00101111

Cadena invalida

Reintentar [s/n]:
```

Figura 31: Parte de la historia de la Maquina de Turing en consola.

```
1 La cadena es: 001100101111
2 (q0)001100101111 |- X(q1)01100101111 |- X0(q1)1100101111 |- X(q2)0Y100101111 |- (q2)X0Y100101111 |-
• X(q0)0Y100101111 |- XX(q1)Y100101111 |- XX(q1)Y100101111 |- XX(q2)YY00101111 |- X(q2)XY00101111 |-
• XX(q0)YY00101111 |- XX(q3)Y00101111 |- XXYY(q3)00101111
3 Cadena invalida
```

Figura 32: Historia de la Maquina de Turing en un archivo.

## Referencias

- [1] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introducción a La Teoría De Autómatas, Lenguajes Y Computación*. Addison-Wesley, 2007.
- [2] J. D. Ullman, “Finite Automata.” <http://infolab.stanford.edu/~ullman/ialc/spr10/slides/fa1.pdf>, 2010. [Consultado: 2016-12-3].