

Entrega Final de Programas

Aburto Pérez Luis Mario

3 de diciembre de 2016

Índice

1. WEB/EBAY	3
1.1. Descripción del Problema	3
1.2. Código fuente	3
1.3. Pruebas	12
2. Planetas	17
2.1. Descripción del Problema	17
2.2. Código fuente	17
2.3. Pruebas	20
3. Palindromo	24
3.1. Descripción del Problema	24
3.2. Código fuente	24
3.3. Pruebas	26
4. Parentesis	29
4.1. Descripción del Problema	29
4.2. Código fuente	29
4.3. Pruebas	32
5. Pila	35
5.1. Descripción del Problema	35
5.2. Código fuente	35
5.3. Pruebas	40
6. Máquina de Turing	44
6.1. Descripción del Problema	44
6.2. Código fuente	44
6.3. Pruebas	47

1. WEB/EBAY

1.1. Descripción del Problema

Se desea realizar un programa el cual funcione como un buscador de texto, en este caso buscara sub-cadenas dentro de otras. Las cadenas que buscara serán web y ebay. Las cadenas que sean validadas por el autómata las guardara en un archivo además de indicar cuál es su posición con fila y carácter de la fila.

1.2. Código fuente

Este programa fue elaborado en el lenguaje Python, su código fuente se muestra a continuación:

Archivo: webay.py

```
from tkinter import *
from random import *
from grafico import *

def validacion(num):
    if (num>=97 and num<=122):
        return 1
    else:
        return 0

def automata(cadena):
    f=open("estados.txt","a")
    estado=0
    control=0
    if(cadena==''):
        return 0
    else:
        for c in cadena:
            if(estado==0):
                f.write("Estado_"+str(estado)+"_"+c+"_")
                if(c=='w'):
                    estado=1
                elif(c=='e'):
                    estado=4
                else:
                    estado=0
            elif(estado==1):
                f.write("Estado_"+str(estado)+"_"+c+"_")
                if(c=='e'):
                    estado=2
                elif(c=='w'):
                    estado=1
                else:
                    estado=0
```

```

elif(estado==2):
    f.write("Estado_"+str(estado)+"_"+ c+" ")
    if(c=='b'):
        estado=3
        control+=1
    elif(c=='e'):
        estado=4
    elif(c=='w'):
        estado=1
    else:
        estado=0
elif(estado==3):
    f.write("Estado_"+str(estado)+"_"+ c+" ")
    if(c=='a'):
        estado=6
    elif(c=='e'):
        estado=4
    elif(c=='w'):
        estado=1
    else:
        estado=0
elif(estado==4):
    f.write("Estado_"+str(estado)+"_"+ c+" ")
    if(c=='b'):
        estado=5
    elif(c=='e'):
        estado=4
    elif(c=='w'):
        estado=1
elif(estado==5):
    f.write("Estado_"+str(estado)+"_"+ c+" ")
    if(c=='a'):
        estado=6
    elif(c=='e'):
        estado=4
    elif(c=='w'):
        estado=1
    else:
        estado=0
elif(estado==6):
    f.write("Estado_"+str(estado)+"_"+ c+" ")
    if(c=='y'):
        estado=7
        control+=1
    elif(c=='e'):
        estado=4
    elif(c=='w'):
        estado=1
    else:
        estado=0
elif(estado==7):
    f.write("Estado_"+str(estado)+"_"+ c+" ")

```

```

        if(c=='w'):
            estado=1
        elif(c=='e'):
            estado=4

    if(control>0):
        return 1
    else:
        return 0

def auto():
    f = open("archivo.txt","r")
    linea=""
    noLinea=1
    control=0
    temporal=''
    linea=f.readline()
    while (linea != ''):
        posTemporal=0
        pos=0
        palabra=linea.lower()
        for c in palabra:
            if(validacion(ord(c))==0):
                control=1
                if(automata(temporal)==1):
                    guardarPalabra(pos,linea,noLinea)
                    temporal=''
                    posTemporal+=1
                    pos=posTemporal
            else:
                temporal+=c
                posTemporal+=1
                control=0
        linea=f.readline()
        noLinea+=1
    if(control==0):
        if(automata(temporal)==1):
            guardarPalabra(pos,linea,noLinea)
    f.close()
    f=open("estados.txt","a")
    f.write("\n\n")
    f.close()
    f=open("cadenas.txt","a")
    f.write("\n\n")
    f.close()
    repetir(2)

def manual():
    control=0
    pos=0
    posTemporal=0
    temporal=''
    cadena=input("Ingresa_la_cadena_")

```

```

palabra=cadena.lower()
for c in palabra:
    if (validacion(ord(c))==0):
        control=1
        if (automata(temporal)==1):
            print("Cadena_Valida")
            guardarPalabra(pos,cadena,1)
        else:
            print("Cadena_Invalida")
            temporal=''
            posTemporal+=1
            pos=posTemporal
    else:
        temporal+=c
        posTemporal+=1
        control=0
if (control==0):
    if (automata(temporal)==1):
        guardarPalabra(pos,cadena,1)
        print("Cadena_Valida")
    else:
        print("Cadena_Invalida")
f=open("estados.txt","a")
f.write("\n\n")
f.close()
f=open("cadenas.txt","a")
f.write("\n\n")
f.close()
repetir(1)

def guardarPalabra(caracter,cadena, linea):
    f=open("cadenas.txt","a")
    temporal=''
    x=caracter
    while (x<len(cadena) and (validacion(ord(cadena[x].lower()))!=0)):
        temporal+=cadena[x]
        x+=1
    f.write("Fila_"+str(linea)+"_Caracter_"+str(caracter+1)+"_"+temporal+"
    ")
    f.close()

def crearArchivos():
    cadenas=open("cadenas.txt","w")
    cadenas.close()
    estados=open("estados.txt","w")
    estados.close()

def menu():
    crearArchivos()
    eleccion=0
    while (eleccion!=4):
        eleccion=input("Selecciona una opcion\n1. Manual\n2. Automatico\

```

```

        n3.Mostrar_Grafico\n4. Salir\n====>_")
    if (eleccion=='1'):
        print("Opcion_Manual_seleccionada")
        manual()
    elif (eleccion=='2'):
        print("Opcion_Automatica_seleccionada")
        auto()
    elif (eleccion=='3'):
        print("Visualizar_Grafico")
        grafico()
    elif (eleccion=='4'):
        print("Adios")
        return 0
    else:
        print("Opcion_Invalida ,_Intentalo_de_nuevo")

def repetir(modos):
    rep=''
    if (modos==1):
        while (rep!='1' and rep!='2'):
            rep=input("Deseas_Repetir_en_este_modos\n1._Si_\n2._No_\n==>_")
            if (rep=='1'):
                print("Seleccionaste_repetir")
                manual()
            elif (rep=='2'):
                print("Seleccionaste_NO_repetir")
            else:
                print("Opcion_Invalida ,_Intentalo_de_nuevo")
        elif (modos==2):
            rep=choice(['1', '2'])
            if (rep=='1'):
                print("Seleccionaste_repetir")
                auto()
            elif (rep=='2'):
                print("Seleccionaste_NO_repetir")

menu()

```

Archivo: grafico.py

```

from tkinter import *
def grafico():
    root = Tk()
    root.title('Automata_Terminacion_"web/ebay"_Grafico') # Nombre de la
        ventana
    canvas=Canvas(root,width=1000,height=700)
    canvas.pack()

    canvas.create_oval(100, 300, 200, 400) #Circulo 0
    canvas.create_arc(180,340,30,360, start=90, extent=180, style='arc') #

```

```

    arco 0-0
canvas.create_oval(90,355,100,365,fill="red")
#canvas.create_line(0,350,100,350)#Linea incial
canvas.create_line(200,340,300,250)#Linea 0-1
canvas.create_oval(290,250,300,260,fill="red")
canvas.create_line(200,360,300,450)#Linea 0-4
canvas.create_oval(290,440,300,450,fill="red")
canvas.create_arc(150,700,950,0, start=188, extent=149, style='arc') #
    arco 0-7
canvas.create_arc(150,600,800,125, start=188, extent=138, style='arc')
    #arco 0-6
canvas.create_arc(150,550,600,180, start=190, extent=125, style='arc')
    #arco 0-5
canvas.create_arc(150,495,450,265, start=190, extent=92, style='arc')
    #arco 0-4
canvas.create_oval(150,400,160,410,fill="red")

canvas.create_oval(300, 200, 400, 300) #Circulo 1
canvas.create_line(400,250,500,250)#Linea 1-2
canvas.create_oval(490,245,500,255,fill="red")
canvas.create_arc(150,350,340,200, start=45, extent=150, style='arc')
    #arco 1-0
canvas.create_oval(145,290,155,300,fill="red")
canvas.create_arc(370,205,330,160, start=300, extent=300, style='arc')
    #arco 1-1
canvas.create_oval(330,195,340,205,fill="red")
canvas.create_arc(370,205,330,160, start=300, extent=300, style='arc')
    #arco 1-1

canvas.create_oval(500, 200, 600, 300) #Circulo 2
canvas.create_line(600,250,700,250)#Linea 2-3
canvas.create_oval(690,245,700,255,fill="red")
canvas.create_arc(525,200,375,240, start=20, extent=140, style='arc')
    #arco 2-1
canvas.create_oval(380,205,390,215,fill="red")
canvas.create_arc(550,150,150,360, start=28, extent=158, style='arc')
    #arco 2-0
canvas.create_line(550,300,370,405)#Linea 2-4

canvas.create_oval(700, 200, 800, 300, width=3) #Circulo 3
canvas.create_arc(720,70,150,380, start=5, extent=180, style='arc') #
    arco 3-0
canvas.create_arc(725,190,380,240, start=10, extent=150, style='arc')
    #arco 3-1
canvas.create_line(750,300,370,405)#Linea 3-4
canvas.create_oval(370,405,380,395,fill="red")

canvas.create_oval(300, 400, 400, 500) #Circulo 4
canvas.create_line(400,450,500,450)#Linea 4-5
canvas.create_oval(490,445,500,455,fill="red")
canvas.create_arc(290,460,310,480, start=95, extent=255, style='arc')
    #arco 4-4

```



```

canvas.create_oval(300,480,310,470,fill="red")
canvas.create_arc(370,450,520,500, start=210, extent=125, style='arc')
    #arco 4-5
canvas.create_arc(370,440,720,550, start=180, extent=185, style='arc')
    #arco 4-6
canvas.create_arc(370,400,905,600, start=180, extent=173, style='arc')
    #arco 4-7
canvas.create_oval(365,495,375,505,fill="red")
canvas.create_oval(380,485,390,495,fill="red")
canvas.create_line(350,400,350,300)#Linea 4-1
canvas.create_oval(345,300,355,310,fill="red")

canvas.create_oval(500, 400, 600, 500) #Circulo 5
canvas.create_line(600,450,700,450)#Linea 5-6
canvas.create_oval(690,445,700,455,fill="red")
canvas.create_line(550,400,355,305)#Linea 5-1

canvas.create_oval(700, 400, 800, 500) #Circulo 6
canvas.create_line(800,450,900,450)#Linea 6-7
canvas.create_oval(890,445,900,455,fill="red")
canvas.create_line(750,400,355,305)#Linea 6-1
canvas.create_line(750,300,770,408)#Linea 3-6
canvas.create_oval(765,408,775,398,fill="red")

canvas.create_oval(900,400,1000,500, width=3)#Circulo 7
canvas.create_line(950,400,355,305)#Linea 7-1

#Estados
letra=Label(root ,text="Q0",font=("Helvetica",15))
letra.place(x=125,y=330)
letra=Label(root ,text="Q1",font=("Helvetica",15))
letra.place(x=330,y=225)
letra=Label(root ,text="Q2",font=("Helvetica",15))
letra.place(x=530,y=225)
letra=Label(root ,text="Q3",font=("Helvetica",15))
letra.place(x=730,y=225)
letra=Label(root ,text="Q4",font=("Helvetica",15))
letra.place(x=330,y=425)
letra=Label(root ,text="Q5",font=("Helvetica",15))
letra.place(x=530,y=425)
letra=Label(root ,text="Q6",font=("Helvetica",15))
letra.place(x=730,y=425)
letra=Label(root ,text="Q7",font=("Helvetica",15))
letra.place(x=930,y=425)
#Letras e
letra=Label(root ,text="e")
letra.place(x=250,y=400)
letra=Label(root ,text="e")
letra.place(x=390,y=375)
letra=Label(root ,text="e")
letra.place(x=690,y=295)
letra=Label(root ,text="e")

```

```

letra . place (x=280,y=450)
letra=Label (root , text="e" )
letra . place (x=430,y=490)
letra=Label (root , text="e" )
letra . place (x=550,y=540)
letra=Label (root , text="e" )
letra . place (x=690,y=585)
letra=Label (root , text="e" )
letra . place (x=450,y=230)
#Letras b
letra=Label (root , text="b" )
letra . place (x=650,y=230)
letra=Label (root , text="b" )
letra . place (x=450,y=430)
#Letras a
letra=Label (root , text="a" )
letra . place (x=650,y=430)
letra=Label (root , text="a" )
letra . place (x=750,y=330)
#Letras y
letra=Label (root , text="y" )
letra . place (x=850,y=430)
#Letras compuestas
letra=Label (root , text="E-e-w" )
letra . place (x=20,y=330)
letra=Label (root , text="E-e-w" )
letra . place (x=220,y=180)
letra=Label (root , text="E-b-e-w" )
letra . place (x=270,y=150)
letra=Label (root , text="E-a-e-w" )
letra . place (x=430,y=80)
letra=Label (root , text="E-b-e-w" )
letra . place (x=220,y=475)
letra=Label (root , text="E-a-e-w" )
letra . place (x=320,y=530)
letra=Label (root , text="E-y-e-w" )
letra . place (x=340,y=570)
letra=Label (root , text="E-e-w" )
letra . place (x=450,y=650)
# Letras w
letra=Label (root , text="w" )
letra . place (x=650,y=180)
letra=Label (root , text="w" )
letra . place (x=340,y=160)
letra=Label (root , text="w" )
letra . place (x=450,y=200)
letra=Label (root , text="w" )
letra . place (x=420,y=330)
letra=Label (root , text="w" )
letra . place (x=600,y=360)
letra=Label (root , text="w" )
letra . place (x=850,y=380)

```

```
letra=Label(root , text="w")  
letra . place (x=340,y=350)  
letra=Label(root , text="w")  
letra . pack ()  
letra . place (x=250,y=300)  
root . mainloop ()
```

1.3. Pruebas

Las pruebas que se realizaron sobre este programa fueron 3 una para modo manual, una en modo automático y para mostrar su gráfico se muestran a continuación:

```
C:\Users\Luis\Desktop>python webay.py
Selecciona una opcion
1.Manual
2.Automatico
3.Mostrar Grafico
4.Salir
==> 2
Opcion Automatica seleccionada
Seleccionaste NO repetir
Selecciona una opcion
1.Manual
2.Automatico
3.Mostrar Grafico
4.Salir
==> 4
..
```

Figura 1: Modo Automático

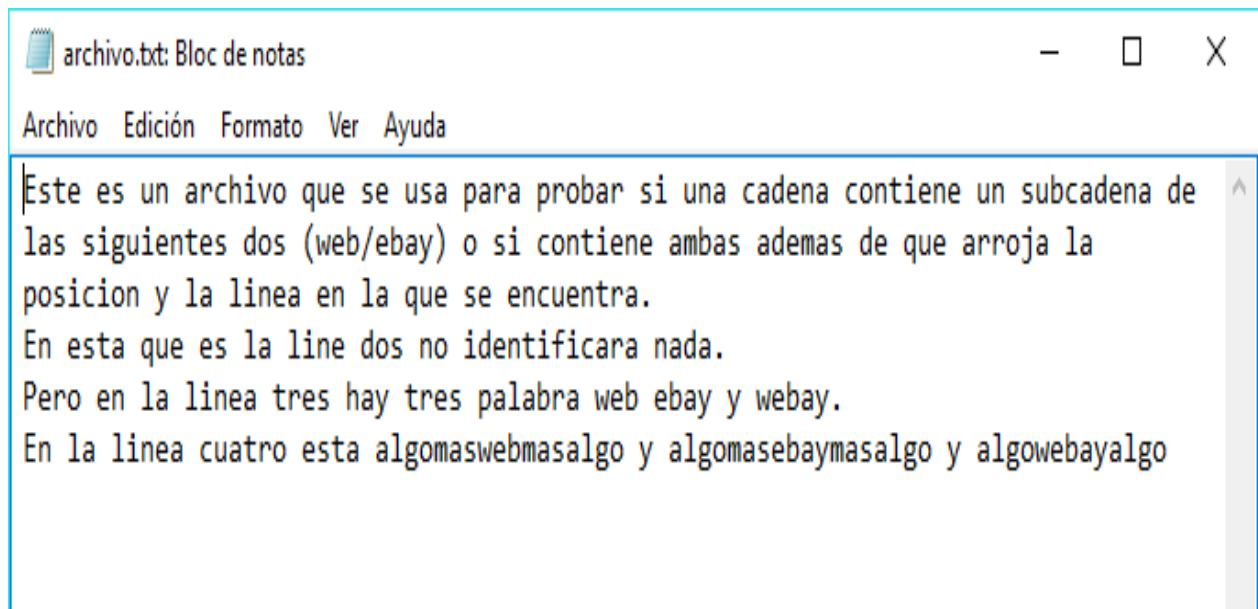


Figura 2: Para el modo automático lee un archivo en este caso archivo.txt

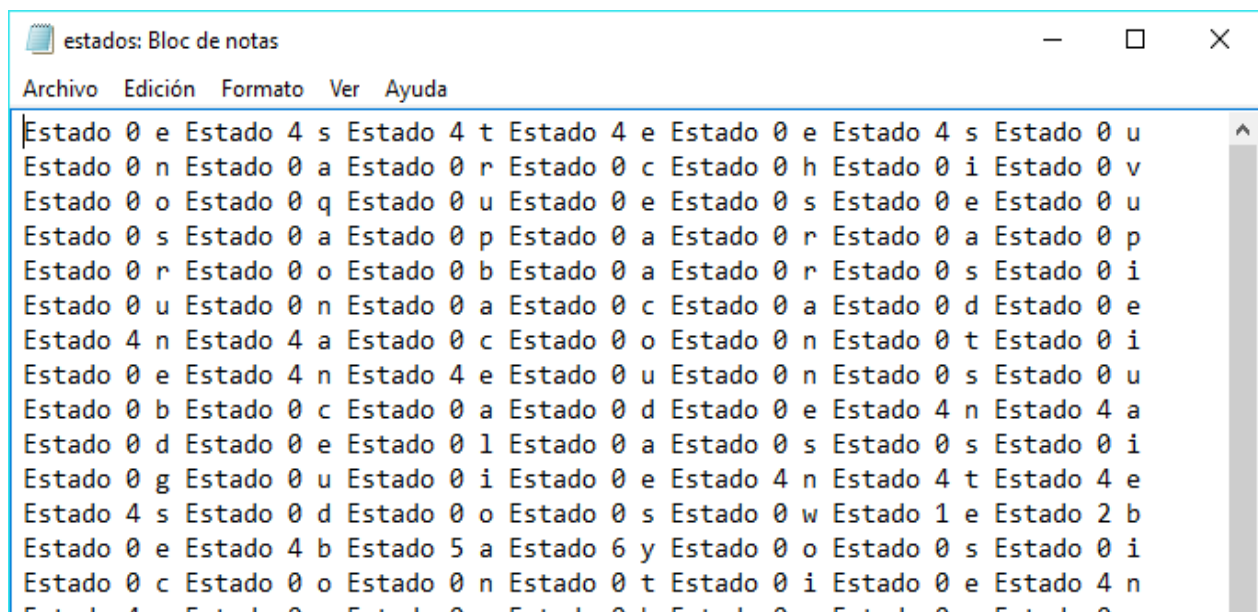


Figura 3: Estados evaluados por el autómata

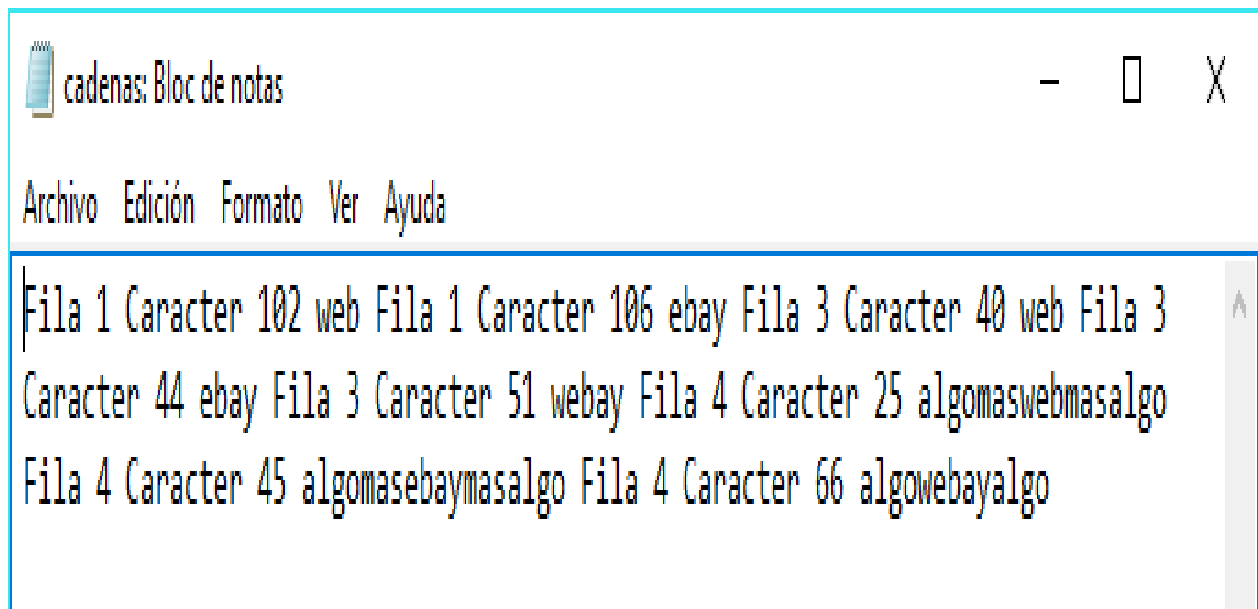


Figura 4: Cadenas validas guardadas en el archivo cadenas.txt

```
C:\Users\Luis\Desktop>python webay.py
Selecciona una opcion
1.Manual
2.Automatico
3.Mostrar Grafico
4.Salir
==> 1
Opcion Manual seleccionada
Ingresa la cadena El programa lee cualquier cadena que contenga la palabra web o eb
ay tambien si encuanta una palabra con ambas como webay o algomaswebbayoalgomas
```

Figura 5: Modo Manual

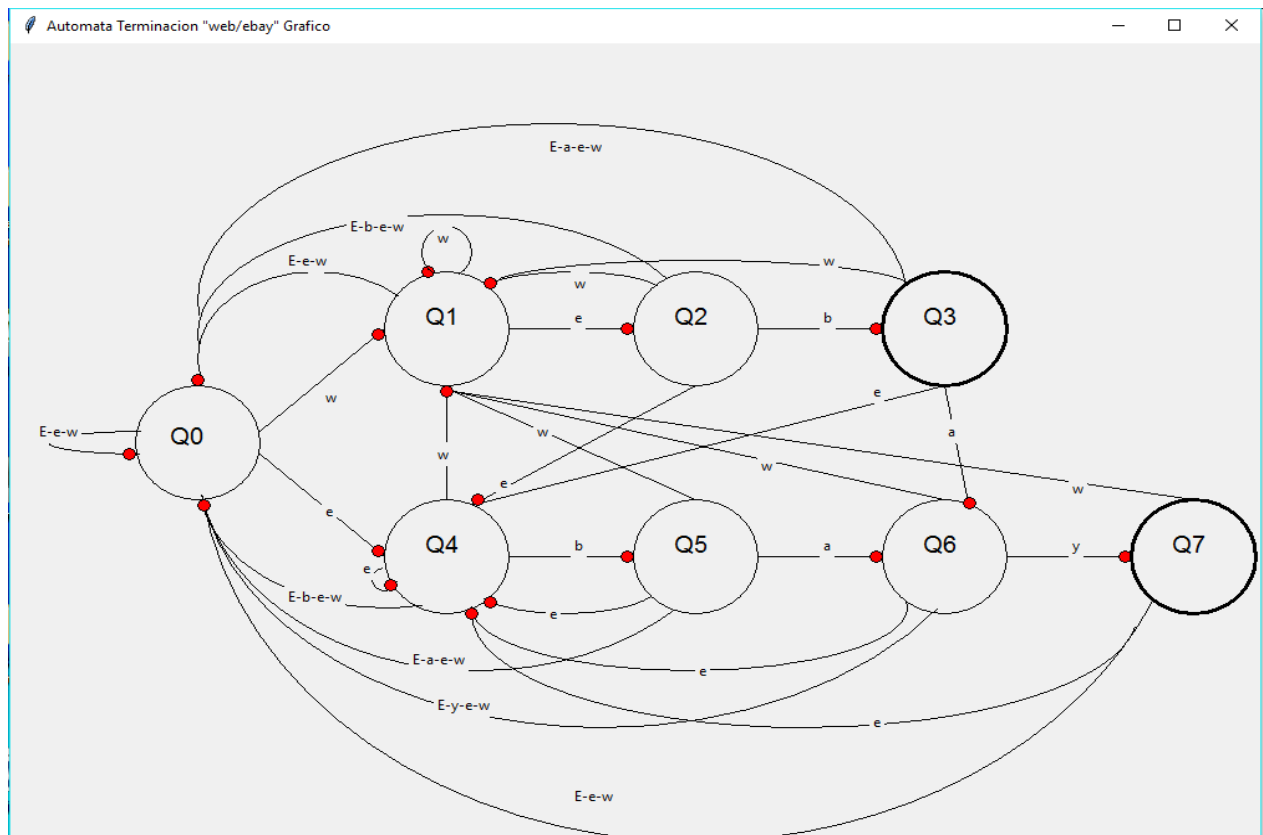


Figura 8: Gráfico del autòmata

2. Planetas

2.1. Descripción del Problema

Se desea realizar un programa el cual de acuerdo a un número n de habitantes genere las combinaciones que son posibles para tres especies dentro del planeta, una vez generadas las combinaciones evaluara la combinación y realizara un proceso para verificar cuales combinaciones fallan y cuáles no, la combinación falla cuando una especie es dominante, es decir, las otras dos especies tiene cero habitantes. El proceso que se aplicara será matar a uno dos especies para añadir dos a la otra especie. El programa solo realizara proceso con combinaciones que no fallen desde el inicio y dará todos los caminos posibles para cada combinación.

2.2. Código fuente

Este programa fue elaborado en el lenguaje Python, su código fuente se muestra a continuación:

Archivo: planetas.py

```
from random import *

def eliminarFinales(listas):
    x=0
    for lista in listas:
        if(lista[0]==0 and lista[1]==0):
            del(listas[x])
        elif(lista[0]==0 and lista[2]==0):
            del(listas[x])
        elif(lista[2]==0 and lista[1]==0):
            del(listas[x])
        x+=1

def generarCombinaciones(listas,n):
    for x in range(0,n+1):
        for y in range(0,n+1):
            for z in range(0,n+1):
                if((x+y+z)==n):
                    listas.append([x,y,z])

    eliminarFinales(listas)

def falla(lista):
    if(lista[0]==0 and lista[1]==0):
        return 1
    elif(lista[0]==0 and lista[2]==0):
        return 1
    elif(lista[2]==0 and lista[1]==0):
        return 1
    else:
```

```

        return 0

def enRuta(lista, ruta):
    if(ruta==[]):
        return 0
    for casilla in ruta:
        if(casilla==lista):
            return 1
    return 0

def proceso(especie, lista, ruta):
    nueva=lista.copy()
    if(enRuta(nueva, ruta)==1):
        f=open("NoTerminan.txt", "a")
        ruta.append(especie)
        ruta.append(lista.copy())
        del(ruta[0])
        f.writelines(str(ruta[0])+"_No_Termina_"+str(ruta)+"\n")
        f.close()
    elif(falla(nueva)==1):
        f=open("Terminan.txt", "a")
        ruta.append(especie)
        ruta.append(lista.copy())
        del(ruta[0])
        f.writelines(str(ruta[0])+"_TERMINA_"+str(ruta)+"\n")
        f.close()
    else:
        ruta.append(especie)
        ruta.append(nueva.copy())
        if(nueva[0]==0):
            nueva[0]+=2
            nueva[1]-=1
            nueva[2]-=1
            proceso("A", nueva.copy(), ruta.copy())
        elif(nueva[1]==0):
            nueva[1]+=2
            nueva[0]-=1
            nueva[2]-=1
            proceso("B", nueva.copy(), ruta.copy())
        elif(nueva[2]==0):
            nueva[2]+=2
            nueva[0]-=1
            nueva[1]-=1
            proceso("C", nueva.copy(), ruta.copy())
        else:
            nueva1=nueva.copy()
            nueva2=nueva.copy()
            nueva3=nueva.copy()
            nueva1[0]+=2
            nueva1[1]-=1
            nueva1[2]-=1
            proceso("A", nueva1.copy(), ruta.copy())

```

```

        nueva2[1]+=2
        nueva2[0]-=1
        nueva2[2]-=1
        proceso("B",nueva2.copy(),ruta.copy())
        nueva3[2]+=2
        nueva3[0]-=1
        nueva3[1]-=1
        proceso("C",nueva3.copy(),ruta.copy())

def crearArchivo():
    f=open("Terminan.txt","w")
    f.close()
    f=open("NoTerminan.txt","w")
    f.close()

def manual():
    listas=[]
    habitantes=0
    habitantes=int(input("Ingresa el numero de habitantes en el planeta_
(2-15)_==>_"))
    while(habitantes>15 or habitantes<2):
        print("Opcion Invalida")
        habitantes=int(input("Ingresa el numero de habitantes en el_
planeta_(2-15)_==>_"))
    f=open("Terminan.txt","a")
    g=open("NoTerminan.txt","a")
    generarCombinaciones(listas,habitantes)
    for lista in listas:
        ruta=[]
        proceso("A",lista,ruta)
    f.write("Termina_Ejecucion\n")
    g.write("Termina_Ejecucion\n")
    f.close()
    g.close()
    repetir(1)

def auto():
    listas=[]
    habitantes=randint(2,15)
    f=open("Terminan.txt","a")
    g=open("NoTerminan.txt","a")
    print("El_numero_seleccionado_es_" +str(habitantes))
    generarCombinaciones(listas,habitantes)
    f.write("Termina_Ejecucion\n")
    g.write("Termina_Ejecucion\n")
    for lista in listas:
        ruta=[]
        proceso("A",lista,ruta)
    f.close()
    g.close()
    repetir(2)

```

```

def menu() :
    eleccion=0
    crearArchivo()
    while (eleccion!=4):
        eleccion=input("Selecciona una opcion\n1. Manual\n2. Automatico\n3. Salir\n==>")
        if (eleccion=='1'):
            print("Opcion_Manual_seleccionada")
            manual()
        elif (eleccion=='2'):
            print("Opcion_Automatica_seleccionada")
            auto()
        elif (eleccion=='3'):
            print("Adios")
            return 0
        else:
            print("Opcion_Invalida , Intentalo_de_nuevo")

def repetir(modos):
    rep=''
    if (modos==1):
        while (rep!='1' and rep!='2'):
            rep=input("Deseas Repetir en este modo\n1. Si\n2. No\n==>")
            if (rep=='1'):
                print("Seleccionaste repetir")
                manual()
            elif (rep=='2'):
                print("Seleccionaste NO repetir")
            else:
                print("Opcion_Invalida , Intentalo_de_nuevo")
        elif (modos==2):
            rep=choice(['1', '2'])
            if (rep=='1'):
                print("Seleccionaste repetir")
                auto()
            elif (rep=='2'):
                print("Seleccionaste NO repetir")

menu()

```

2.3. Pruebas

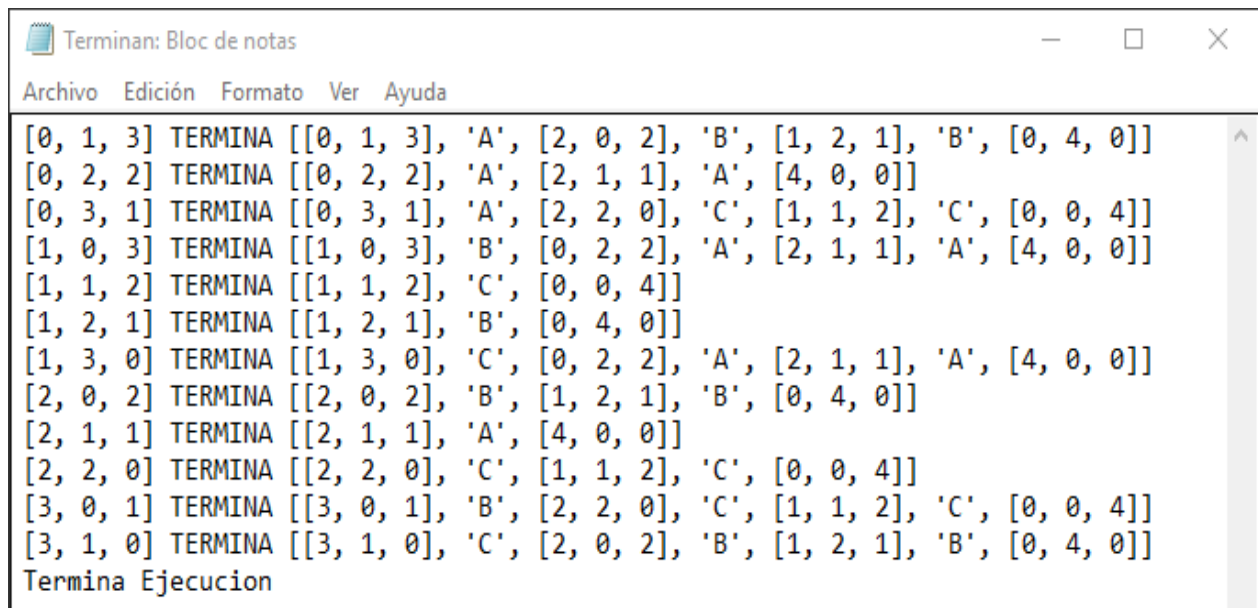
Para este programa se realizaron dos pruebas, en su modo automático donde el número de habitantes es generado por un random y otro en el modo manual que es ingresado por el usuario. También se mostrara todos los caminos posibles que son tomados por la combinaciones generadas.

```

C:\Users\Luis\Desktop>python planetas.py
Selecciona una opcion
1.Manual
2.Automatico
3.Salir
==> 2
Opcion Automatica seleccionada
El numero seleccionado es 4
Seleccionaste NO repetir
Selecciona una opcion
1.Manual
2.Automatico
3.Salir
==> 3
Adios

```

Figura 9: Modo Automático



```

Terminan: Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda

[0, 1, 3] TERMINA [[0, 1, 3], 'A', [2, 0, 2], 'B', [1, 2, 1], 'B', [0, 4, 0]]
[0, 2, 2] TERMINA [[0, 2, 2], 'A', [2, 1, 1], 'A', [4, 0, 0]]
[0, 3, 1] TERMINA [[0, 3, 1], 'A', [2, 2, 0], 'C', [1, 1, 2], 'C', [0, 0, 4]]
[1, 0, 3] TERMINA [[1, 0, 3], 'B', [0, 2, 2], 'A', [2, 1, 1], 'A', [4, 0, 0]]
[1, 1, 2] TERMINA [[1, 1, 2], 'C', [0, 0, 4]]
[1, 2, 1] TERMINA [[1, 2, 1], 'B', [0, 4, 0]]
[1, 3, 0] TERMINA [[1, 3, 0], 'C', [0, 2, 2], 'A', [2, 1, 1], 'A', [4, 0, 0]]
[2, 0, 2] TERMINA [[2, 0, 2], 'B', [1, 2, 1], 'B', [0, 4, 0]]
[2, 1, 1] TERMINA [[2, 1, 1], 'A', [4, 0, 0]]
[2, 2, 0] TERMINA [[2, 2, 0], 'C', [1, 1, 2], 'C', [0, 0, 4]]
[3, 0, 1] TERMINA [[3, 0, 1], 'B', [2, 2, 0], 'C', [1, 1, 2], 'C', [0, 0, 4]]
[3, 1, 0] TERMINA [[3, 1, 0], 'C', [2, 0, 2], 'B', [1, 2, 1], 'B', [0, 4, 0]]
Termina Ejecucion

```

Figura 10: Combinaciones que fallan con su proceso.

Figura 11: Combinaciones que no fallan con su proceso.

```
C:\Users\Luis\Desktop>python planetas.py
Selecciona una opcion
1.Manual
2.Automatico
3.Salir
==> 1
Opcion Manual seleccionada
Ingresa el numero de habitantes en el planeta (2-15) ==> 5
Deseas Repetir en este modo
1. Si
2. No
==> 2
.
.
```

Figura 12: Modo Manual

```

Terminan: Bloc de notas
Archivo Edición Formato Ver Ayuda

[[0, 1, 4] TERMINA [[0, 1, 4], 'A', [2, 0, 3], 'B', [1, 2, 2], 'A', [3, 1, 1],
'A', [5, 0, 0]]
[0, 2, 3] TERMINA [[0, 2, 3], 'A', [2, 1, 2], 'B', [1, 3, 1], 'B', [0, 5, 0]]
[0, 3, 2] TERMINA [[0, 3, 2], 'A', [2, 2, 1], 'C', [1, 1, 3], 'C', [0, 0, 5]]
[0, 4, 1] TERMINA [[0, 4, 1], 'A', [2, 3, 0], 'C', [1, 2, 2], 'A', [3, 1, 1],
'A', [5, 0, 0]]
[1, 0, 4] TERMINA [[1, 0, 4], 'B', [0, 2, 3], 'A', [2, 1, 2], 'B', [1, 3, 1],
'B', [0, 5, 0]]
[1, 1, 3] TERMINA [[1, 1, 3], 'C', [0, 0, 5]]
[1, 2, 2] TERMINA [[1, 2, 2], 'A', [3, 1, 1], 'A', [5, 0, 0]]
[1, 3, 1] TERMINA [[1, 3, 1], 'B', [0, 5, 0]]

```

Figura 13: Combinaciones que fallan con su proceso.

```

NoTerminan: Bloc de notas
Archivo Edición Formato Ver Ayuda

[[0, 1, 4] No Termina [[0, 1, 4], 'A', [2, 0, 3], 'B', [1, 2, 2], 'A', [3, 1, 1],
'B', [2, 3, 0], 'C', [1, 2, 2]]
[0, 1, 4] No Termina [[0, 1, 4], 'A', [2, 0, 3], 'B', [1, 2, 2], 'A', [3, 1, 1],
'C', [2, 0, 3]]
[0, 1, 4] No Termina [[0, 1, 4], 'A', [2, 0, 3], 'B', [1, 2, 2], 'B', [0, 4, 1],
'A', [2, 3, 0], 'C', [1, 2, 2]]
[0, 1, 4] No Termina [[0, 1, 4], 'A', [2, 0, 3], 'B', [1, 2, 2], 'C', [0, 1, 4]]
[0, 2, 3] No Termina [[0, 2, 3], 'A', [2, 1, 2], 'A', [4, 0, 1], 'B', [3, 2, 0],
'C', [2, 1, 2]]
[0, 2, 3] No Termina [[0, 2, 3], 'A', [2, 1, 2], 'B', [1, 3, 1], 'A', [3, 2, 0],
'C', [2, 1, 2]]
[0, 2, 3] No Termina [[0, 2, 3], 'A', [2, 1, 2], 'B', [1, 3, 1], 'C', [0, 2, 3]]
[0, 2, 3] No Termina [[0, 2, 3], 'A', [2, 1, 2], 'C', [1, 0, 4], 'B', [0, 2, 3]]
[0, 3, 2] No Termina [[0, 3, 2], 'A', [2, 2, 1], 'A', [4, 1, 0], 'C', [3, 0, 2],
'B', [2, 2, 1]]
[0, 3, 2] No Termina [[0, 3, 2], 'A', [2, 2, 1], 'B', [1, 4, 0], 'C', [0, 3, 2]]
[0, 3, 2] No Termina [[0, 3, 2], 'A', [2, 2, 1], 'C', [1, 1, 3], 'A', [3, 0, 2],
'B', [2, 2, 1]]

```

Figura 14: Combinaciones que no fallan con su proceso.

3. Palindromo

3.1. Descripción del Problema

Se desea hacer un programa que genere un palíndromo (cadena que es igual al leerla de izquierda a derecha como leerla de derecha a izquierda) binario, es decir, de 0's y 1's. La cadena que se generara será dependiendo de un numero el cual es la longitud de la cadena además de indicar el número de veces que entrara en el proceso la cadena. La cadena será generada por reglas de producción que serán tomadas de forma aleatoria.

3.2. Código fuente

Este programa fue elaborado en el lenguaje Python, su código fuente se muestra a continuación:

Archivo: palindromo.py

```
from random import *
def generarCadena(n):
    f=open("proceso.txt","a")
    cadena=""
    if (n%2==0):
        rep=0
        while (rep<(n/2)):
            numero=randint(1,2)
            if (numero==1):
                cadena+="0"
                cadenaDos=invertir(cadena)
                #print(" 0P0 | "+cadena+"P"+cadenaDos)
                f.write("_0P0_|_"+cadena+"P"+cadenaDos+"\n")
            if (numero==2):
                cadena+="1"
                cadenaDos=invertir(cadena)
                #print(" 1P1 | "+cadena+"P"+cadenaDos)
                f.write("_1P1_|_"+cadena+"P"+cadenaDos+"\n")
            rep+=1
        cadenaDos=invertir(cadena)
        #print("   e | "+cadena+"e"+cadenaDos)
        f.write("_e_|_"+cadena+"e"+cadenaDos+"\n")
        cadena=cadena+cadenaDos
    else:
        rep=0
        while (rep<(n/2)-0.5):
            numero=randint(1,2)
            if (numero==1):
                cadena+="0"
                cadenaDos=invertir(cadena)
                #print(" 0P0 | "+cadena+"P"+cadenaDos)
```



```

        f.write("_OP0_|_|"+cadena+"P"+cadenaDos+"\n")
    if(numero==2):
        cadena+="1"
        cadenaDos=invertir(cadena)
        #print(" 1P1 | "+cadena+"P"+cadenaDos)
        f.write("_1P1_|_|"+cadena+"P"+cadenaDos+"\n")
    rep+=1
    cadenaDos=invertir(cadena)
    numero=randint(1,2)
    if(numero==1):
        cadena+="0"
        cadena=cadena+cadenaDos
        #print("  0 | "+cadena)
        f.write("_0_|_|"+cadena+"\n")
    elif(numero==2):
        cadena+="1"
        cadena=cadena+cadenaDos
        #print("  1 | "+cadena)
        f.write("_1_|_|"+cadena+"\n")
    print("El_palindromo_binario_generado_es_"+cadena)
    f.write("El_palindromo_binario_generado_es_"+cadena+"\n\n")
    f.close()

def invertir(cadena):
    return cadena[::-1]

def crearArchivo():
    f=open("proceso.txt","w")
    f.close()

def menu():
    crearArchivo()
    eleccion=0
    while (eleccion!=4):
        eleccion=input("Selecciona una opcion\n1. Manual\n2. Automatico\n3. Salir\n====>")
        if(eleccion=='1'):
            print("Opcion_Manual_seleccionada")
            manual()
        elif(eleccion=='2'):
            print("Opcion_Automatica_seleccionada")
            auto()
        elif(eleccion=='3'):
            print("Adios")
            return 0
        else:
            print("Opcion_Invalida , Intentalo de nuevo")

def manual():
    numero=int(input("Ingresa el numero "))
    generarCadena(numero)
    repetir(1)

```

```

def auto():
    numero=randint(0,1000)
    print("El_numero_seleccionado_es_"+str(numero))
    generarCadena(numero)
    repetir(2)

def repetir(modos):
    rep=''
    if(modos==1):
        while(rep!='1' and rep!='2'):
            rep=input("Deseas_Repetir_en_este_modos\n1._Si_\n2._\nNo_\n==>_")
            if(rep=='1'):
                print("Seleccionaste_repetir")
                manual()
            elif(rep=='2'):
                print("Seleccionaste_NO_repetir")
            else:
                print("Opcion_Invalida ,_Intentalo_de_nuevo")
        elif(modos==2):
            rep=choice(['1', '2'])
            if(rep=='1'):
                print("Seleccionaste_repetir")
                auto()
            elif(rep=='2'):
                print("Seleccionaste_NO_repetir")

menu()

```

3.3. Pruebas

Para este programa se realizaron dos pruebas, en su modo automático que generara la longitud de la cadena a través de un random y otro en el modo manual donde el número es ingresado por el usuario. Además de mostrar el proceso por el cual es generado el palíndromo.

```
C:\Users\Luis\Desktop>python palindromo.py
Selecciona una opcion
1.Manual
2.Automatico
3.Salir
==> 2
Opcion Automatica seleccionada
El numero seleccionado es 2
El palindromo binario generado es 11
Seleccionaste NO repetir
Selecciona una opcion
1.Manual
2.Automatico
3.Salir
==> 3
```

Figura 15: Modo Automático

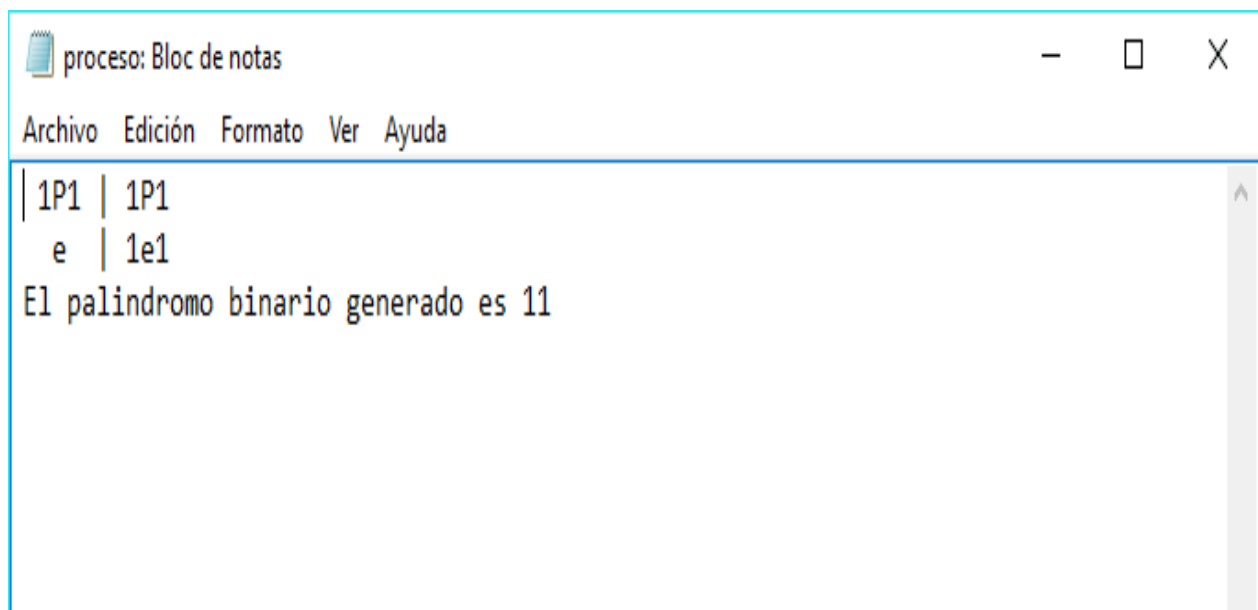


Figura 16: Proceso por el cual es formado el palíndromo.

```

C:\Users\Luis\Desktop>python palindromo.py
Selecciona una opcion
1.Manual
2.Automatico
3.Salir
==> 1
Opcion Manual seleccionada
Ingresa el numero 5
El palindromo binario generado es 01110
Deseas Repetir en este modo
1. Si
2. No
==> 2
Seleccionaste NO repetir
Selecciona una opcion
1.Manual
2.Automatico
3.Salir
==> 3
Adios

```

Figura 17: Modo Manual

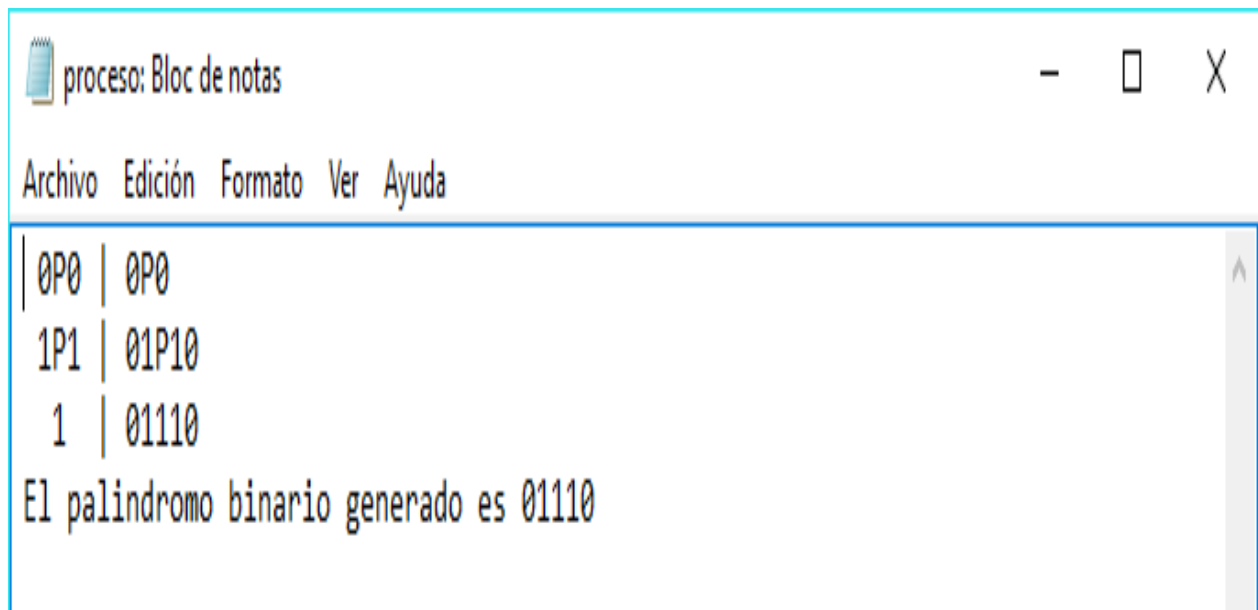


Figura 18: Proceso por el cual es formado el palíndromo.

4. Parentesis

4.1. Descripción del Problema

Se desea realizar un programa el cual evalúe si una cadena esta balanceada en cuanto al número de paréntesis que abren y cierran. La evaluación se hará dependiendo de las reglas de producción que se tienen, dependiendo del carácter que se lea será la regla que se utilizara para la evaluación que será ir sustituyendo el carácter que llega por la regla de producción y al final encontrar si la cadena es válida o no, es decir, si esta balanceada en los paréntesis.

4.2. Código fuente

Este programa fue elaborado en el lenguaje Python, su código fuente se muestra a continuación: Archivo: parentesis.py

```
from random import*

def imprimir( lista ,c, regla):
    f=open( "proceso.txt", "a")
    cad=""
    for x in lista:
        cad+=x
    while(len(c)!=20):
        c+="_"
    if(regla==0):
        f.write( "Inicio_|_|"+c + " |_|"+cad+"\n")
    elif(regla==1):
        f.write( "B->(R_|_|"+c + " |_|"+cad+"\n")
    elif(regla==2):
        f.write( "B->_|_|_|"+c + " |_|"+cad+"\n")
    elif(regla==3):
        f.write( "R->(R_|_|"+c + " |_|"+cad+"\n")
    elif(regla==4):
        f.write( "R->_|_|_|"+c + " |_|"+cad+"\n")
    f.close()

def acomodarCadena(cadena):
    cadenaAux=""
    if(cadena==""):
        return "Resultado"
    for x in range(1,len(cadena)):
        cadenaAux+=cadena[x]
    return cadenaAux

def automata(cadena):
    cadenaAux=cadena
    regla=0
    pasos=[ 'B' ]
```

```

imprimir(pasos,cadena,regla)
for c in cadena:
    cadenaAux=acomodarCadena(cadenaAux)
    x=0
    while (pasos[x]== '(' or pasos[x]== ')') :
        x+=1
    estado=pasos[x]
    if (estado=='B') :
        if (c=='(') :
            pasos.insert(x, '(')
            pasos.remove('B')
            pasos.append('R')
            pasos.append('B')
            regla=1
        elif (c==')') :
            return 0
    elif (estado=='R') :
        if (c=='(') :
            pasos.insert(x-1, '(')
            pasos.insert(x+1, 'R')
            regla=3
        if (c==')') :
            pasos.insert(x, ')')
            pasos.remove('R')
            regla=4
    imprimir(pasos,cadenaAux,regla)
if (pasos[-2]== ') ' and pasos[-1]=='B') :
    pasos.remove('B')
    imprimir(pasos,acomodarCadena(cadenaAux),2)
    return 1
else :
    return 0

def crearArchivo() :
    f=open("proceso.txt","w")
    f.close()

def menu() :
    crearArchivo()
    eleccion=0
    while (eleccion!=4):
        eleccion=input("Selecciona una opcion\n1. Manual\n2. Automatico\n3. Salir\n==>")
        if (eleccion=='1') :
            print("Opcion_Manual_seleccionada")
            manual()
        elif (eleccion=='2') :
            print("Opcion_Automatica_seleccionada")
            auto()
        elif (eleccion=='3') :
            print("Adios")
            return 0

```

```

        else:
            print("Opcion_Invalida ,_Intentalo_de_nuevo")

def manual():
    f=open("proceso.txt","a")
    cadena=input("Ingresa_la_cadena_")
    if(automata(cadena)==1):
        print("La_cadena_"+cadena+"_es_Valida")
        f.write("La_cadena_"+cadena+"_es_Valida\n\n")
    else:
        print("La_cadena_"+cadena+"_NO_esta_balanceada")
        f.write("La_cadena_"+cadena+"_NO_esta_balanceada\n\n")
    f.close()
    repetir(1)

def auto():
    cadena=generarCadena()
    f=open("proceso.txt","a")
    if(automata(cadena)==1):
        print("La_cadena_"+cadena+"_es_Valida")
        f.write("La_cadena_"+cadena+"_es_Valida\n\n")
    else:
        print("La_cadena_"+cadena+"_NO_esta_balanceada")
        f.write("La_cadena_"+cadena+"_NO_esta_balanceada\n\n")
    f.close()
    repetir(2)

def generarCadena():
    longitud=randint(1,20)
    cadena=''
    i=0
    while(i<longitud):
        cadena+=choice(['(', ')'])
        i += 1
    print("La_cadena_generado_es:_"+cadena)
    return cadena

def repetir(modos):
    rep=''
    if(modos==1):
        while(rep!='1' and rep!='2'):
            rep=input("Deseas_Repetir_en_este_modos\n1._Si_\n2._No\n_==>_")
            if(rep=='1'):
                print("Seleccionaste_repetir")
                manual()
            elif(rep=='2'):
                print("Seleccionaste_NO_repetir")
            else:
                print("Opcion_Invalida ,_Intentalo_de_nuevo")
        elif(modos==2):

```

```

rep=choice(['1', '2'])
if(rep=='1'):
    print("Seleccionaste_repetir")
    auto()
elif(rep=='2'):
    print("Seleccionaste_NO_repetir")

menu()

```

4.3. Pruebas

Para este programa se realizaron dos pruebas, en su modo automático que generara una cadena de paréntesis la cual luego verificara si está o no balanceada y la otra será en modo manual en la cual el usuario ingresara la cadena para validar.

```

C:\Users\Luis\Desktop>python parentesis.py
Selecciona una opcion
1.Manual
2.Automatico
3.Salir
==> 2
Opcion Automatica seleccionada
La cadena generado es: (((((
La cadena ((((( NO esta balanceada
Seleccionaste repetir
La cadena generado es: )))))(
La cadena )))))( NO esta balanceada
Seleccionaste repetir
La cadena generado es: ))((((
La cadena ))(((( NO esta balanceada
Seleccionaste repetir
La cadena generado es: ))))((
La cadena ))))(( NO esta balanceada
Seleccionaste NO repetir
Selecciona una opcion
1.Manual
2.Automatico
3.Salir
==> 3
Adios

```

Figura 19: Modo Automático


```

proceso: Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda

Inicio | (((()( | B
B->(RB | ((((| (RB
R->(RR | (((| ((RRB
R->(RR | ()(| (((RRRB
R->(RR | )(| (((RRRRB
R->) | ( | (((()RRRB
R->(RR | | (((()RRRRB
La cadena (((() NO esta balanceada

Inicio | ))()))() | B
La cadena ))()))() NO esta balanceada

Inicio | ))(((())) | B
La cadena ))(((())) NO esta balanceada

Inicio | ()))() | B
B->(RB | ()))() | (RB
R->) | ()))() | ()B
La cadena ()))() NO esta balanceada

```

Figura 20: Proceso de evaluación.

```

C:\Users\Luis\Desktop>python parentesis.py
Selecciona una opcion
1.Manual
2.Automatico
3.Salir
==> 1
Opcion Manual seleccionada
Ingresa la cadena (()())
La cadena (()()) es Valida
Deseas Repetir en este modo
1. Si
2. No
==> 2
Seleccionaste NO repetir
Selecciona una opcion
1.Manual
2.Automatico
3.Salir
==> 3
Adios

```

Figura 21: Modo Manual

```
proceso: Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda

Inicio | (())() | B
B->(RB | (())() | (RB
R->(RR | ))() | ((RRB
R-> ) | )() | (()RB
R-> ) | () | (()B
B->(RB | ) | (()RB
R-> ) | | (()B
B-> e | Resultado | (()B
La cadena (())() es Valida
```

Figura 22: Proceso de evaluación.

5. Pila

5.1. Descripción del Problema

Se desea realizar un programa que emplee el autómata de pila. Para este programa se realizara una clase llamada Pila que tendrá las funciones típicas de este tipo de dato abstracto como el visto en el curso de Estructura de Datos en el semestre anterior. La pila evaluara una secuencia de ceros y unos manejando estados de la pila, caracteres que entran y salen a la pila, en este caso una 'X'. Cada vez que vea un cero se realizara un push() y cada uno un pop() al final se evaluara si la pila esta vacía. Lo que indicara que la cadena es válida.

5.2. Código fuente

Este programa fue elaborado en el lenguaje Python, su código fuente se muestra a continuación: Archivo: pila.py

```
#Pila
class Pila:
    lista=[]
    tope=0
    estado=""

    def inicializarPila(self):
        self.lista.append('Zo')
        self.lista.append(None)
        self.tope=1
        self.estado="q"

    def push(self):
        self.lista[self.tope]='X'
        self.lista.append(None)
        self.tope+=1

    def pop(self):
        del(self.lista[self.tope])
        self.tope-=1
        self.lista[self.tope]=None

    def mostrar(self):
        cad=""
        x=len(self.lista)-2
        while(x>=0):
            cad+=str(self.lista[x])
            x-=1
        return cad

    def esVacia(self):
```

```

        if (self.tope==1):
            return 1
        else:
            return 0

```

Archivo: aPila.py

```

from time import *
from Pila import *
from random import *
from tkinter import *

def automata(pila,cadena):
    f=open("historial.txt","a")
    root = Tk()
    root.title('Grafico') # Nombre de la ventana
    canvas=Canvas(root,width=510,height=500)
    canvas.pack()
    canvas.create_line(175,150,175,50)
    canvas.create_line(175,300,175,400)
    estado=Label(root,text="____",bg="black",font=("Helvetica",100))
    estado.place(x = 100, y = 150)
    pila.estado="q"
    cadenaAux=cadena
    f.write((" "+pila.estado+" "+cadena+" "+pila.mostrar()+" | -")
    estado=Label(root,text="_____",font=(
        Helvetica",15))
    estado.place(x = 170, y = 35)
    estado=Label(root,text="_____",font=(
        Helvetica",15))
    estado.place(x = 170, y = 400)
    estado=Label(root,text=cadenaAux,font=("Helvetica",15))
    estado.place(x = 170, y = 35)
    estado=Label(root,text=pila.mostrar(),font=("Helvetica",15))
    estado.place(x = 170, y = 400)
    estado=Label(root,text=pila.estado+"_",bg="black",font=("Helvetica",
        25),fg="white")
    estado.place(x = 175, y = 200)
    root.update()
    sleep(1)
    for x in cadena:
        if(x=='0'):
            pila.estado='q'
            pila.push()
            cadenaAux=eliminarCaracter(cadenaAux)
            f.write((" "+pila.estado+" "+cadenaAux+" "+pila.mostrar()
                +") | -")
            estado=Label(root,text="_____",font=("Helvetica",15))
            estado.place(x = 170, y = 35)
            estado=Label(root,text="_____",font=("Helvetica",15))

```

```

estado.place(x = 170, y = 400)
estado=Label(root , text=cadenaAux, font=("Helvetica" ,15)
)
estado.place(x = 170, y = 35)
estado=Label(root , text=pila.mostrar() , font=("Helvetica" ,15))
estado.place(x = 170, y = 400)
estado=Label(root , text=pila.estado+"_", bg="black" ,
font=("Helvetica" ,25) , fg="white")
estado.place(x = 175, y = 200)
root.update()
sleep(1)
elif(x=='1'):
    if(pila.tope>1):
        pila.pop()
        pila.estado='p'
        cadenaAux=eliminarCaracter(cadenaAux)
        f.write("(" +pila.estado+" , "+cadenaAux+" , "+pila
.mostrar()+") | -")
        estado=Label(root , text=" _____
        _____" , font=("Helvetica" ,15))
        estado.place(x = 170, y = 35)
        estado=Label(root , text=" _____
        _____" , font=("Helvetica" ,15))
        estado.place(x = 170, y = 400)
        estado=Label(root , text=cadenaAux, font=(
"Helvetica" ,15))
        estado.place(x = 170, y = 35)
        estado=Label(root , text=pila.mostrar() , font=(
"Helvetica" ,15))
        estado.place(x = 170, y = 400)
        estado=Label(root , text=pila.estado+"_" , bg="
black" , font=("Helvetica" ,25) , fg="white")
        estado.place(x = 175, y = 200)
        root.update()
        sleep(1)
    else:
        root.mainloop()
        f.write("\n")
        f.close()
    return 0
if(pila.esVacia()==1):
    pila.estado='f'
    f.write("(" +pila.estado+" , "+cadenaAux+" , "+pila.mostrar()+") \n\
n")
    estado=Label(root , text=" _____" , font=(
"Helvetica" ,15))
    estado.place(x = 170, y = 35)
    estado=Label(root , text=" _____" , font=(
"Helvetica" ,15))
    estado.place(x = 170, y = 400)
    estado=Label(root , text=cadenaAux, font=("Helvetica" ,15))

```

```

estado.place(x = 170, y = 35)
estado=Label(root ,text=pila.mostrar() ,font=("Helvetica",15))
estado.place(x = 170, y = 400)
estado=Label(root ,text=pila.estado+"_",bg="black",font=(
    Helvetica",25),fg="white")
estado.place(x = 175, y = 200)
root.update()
sleep(1)
root.mainloop()
f.write("\n")
f.close()
return 1

def eliminarCaracter(cadena):
    x=1
    cadenaAux=""
    while(x<len(cadena)):
        cadenaAux+=cadena[x]
        x+=1
    if(cadenaAux==""):
        return "epsilon"
    else:
        return cadenaAux

def manual(pila):
    f=open("cadenas.txt","a")
    cadena=input("Ingresa_la_cadena_")
    if(automata(pila,cadena)==1):
        print("Cadena_Valida")
        f.write(cadena+"_Cadena_Valida\n\n")
    else:
        print("Cadena_Invalida")
        f.write(cadena+"_Cadena_Invalida\n\n")
    f.close()
    repetir(1,pila)

def auto(pila):
    f=open("cadenas.txt","a")
    cadena=generarCadena()
    if(automata(pila,cadena)==1):
        print("Cadena_Valida")
        f.write(cadena+"_Cadena_Valida\n\n")
    else:
        print("Cadena_Invalida")
        f.write(cadena+"_Cadena_Invalida\n\n")
    f.close()
    repetir(2,pila)

def generarCadena():
    longitud=randint(1,1000)
    numero=' '
    i=0

```

```

while(i<longitud):
    numero+=choice([ '0', '1' ])
    i += 1
print ("El_numero_generado_es:_"+numero)
return numero

def repetir(modos, pila):
    rep=''
    if (modos==1):
        while (rep!='1' and rep!='2'):
            rep=input ("Deseas_Repetir_en_este_modos\n1._Si_\n2._\n\n==>_")
            if (rep=='1'):
                print ("Seleccionaste_repetir")
                manual(pila)
            elif (rep=='2'):
                print ("Seleccionaste_NO_repetir")
            else:
                print ("Opcion_Invalida ,_Intentalo_de_nuevo")
        elif (modos==2):
            rep=choice([ '1', '2' ])
            if (rep=='1'):
                print ("Seleccionaste_repetir")
                auto(pila)
            elif (rep=='2'):
                print ("Seleccionaste_NO_repetir")

def crearArchivo():
    f=open("historial.txt","w")
    f.close()
    f=open("cadenas.txt","w")
    f.close()

def menu(pila):
    crearArchivo()
    eleccion=0
    while (eleccion!=4):
        eleccion=input ("Selecciona_una_opcion\n1. Manual\n2. Automatico\n\n3. Salir\n\n==>_")
        if (eleccion=='1'):
            print ("Opcion_Manual_seleccionada")
            manual(pila)
        elif (eleccion=='2'):
            print ("Opcion_Automatica_seleccionada")
            auto(pila)
        elif (eleccion=='3'):
            print ("Adios")
            return 0
        else:
            print ("Opcion_Invalida ,_Intentalo_de_nuevo")

pila=Pila()

```

```
pila.inicializarPila()  
menu(pila)
```

5.3. Pruebas

Para este programa se realizaron dos pruebas, en su modo automático que generara una cadena binaria la cual luego verificara si tiene o no el mismo número de 1's y 0's y la otra será en modo manual en la cual el usuario ingresara la cadena para validar. Además se mostrara el proceso en modo gráfico.

```
C:\Users\Luis\Desktop>python aPila.py  
Selecciona una opcion  
1.Manual  
2.Automatico  
3.Salir  
==> 2  
Opcion Automatica seleccionada  
El numero generado es: 010101  
Cadena Valida  
Seleccionaste repetir  
El numero generado es: 01  
Cadena Valida  
Seleccionaste repetir  
El numero generado es: 01010  
Cadena Invalida  
Seleccionaste NO repetir
```

Figura 23: Modo Automático

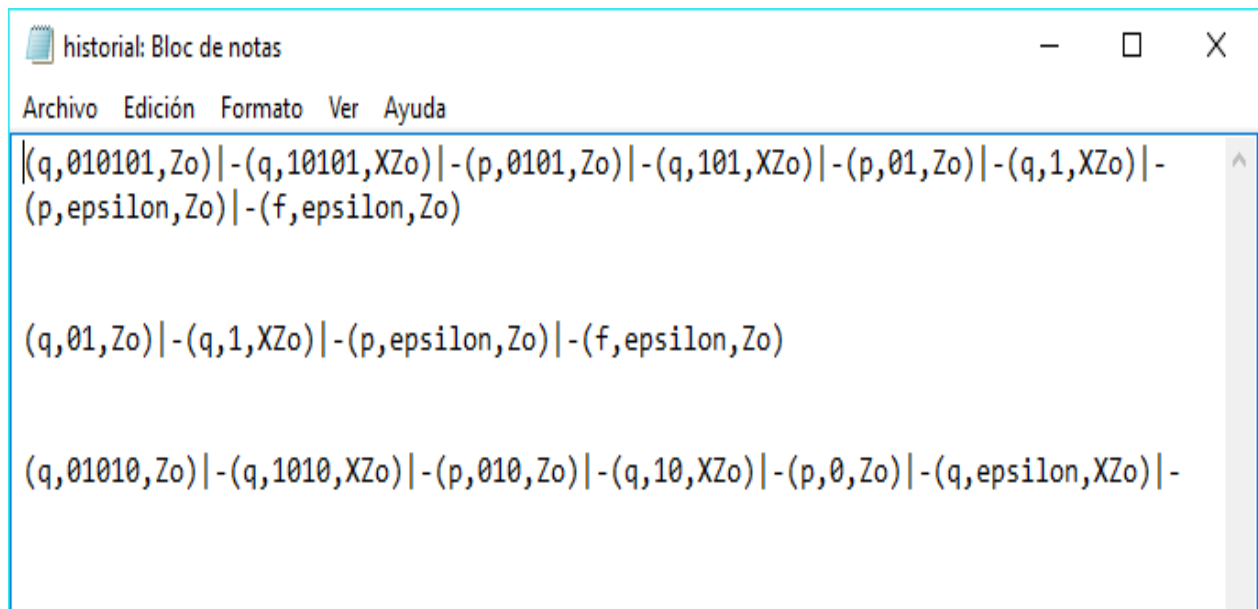


Figura 24: Historial de estados y caracteres por los cuales pasa la pila.

```

C:\Users\Luis\Desktop>python aPila.py
Selecciona una opcion
1.Manual
2.Automatico
3.Salir
==> 1
Opcion Manual seleccionada
Ingresa la cadena 000111
Cadena Valida
Deseas Repetir en este modo
1. Si
2. No
==> 2
Seleccionaste NO repetir

```

Figura 25: Modo Manual

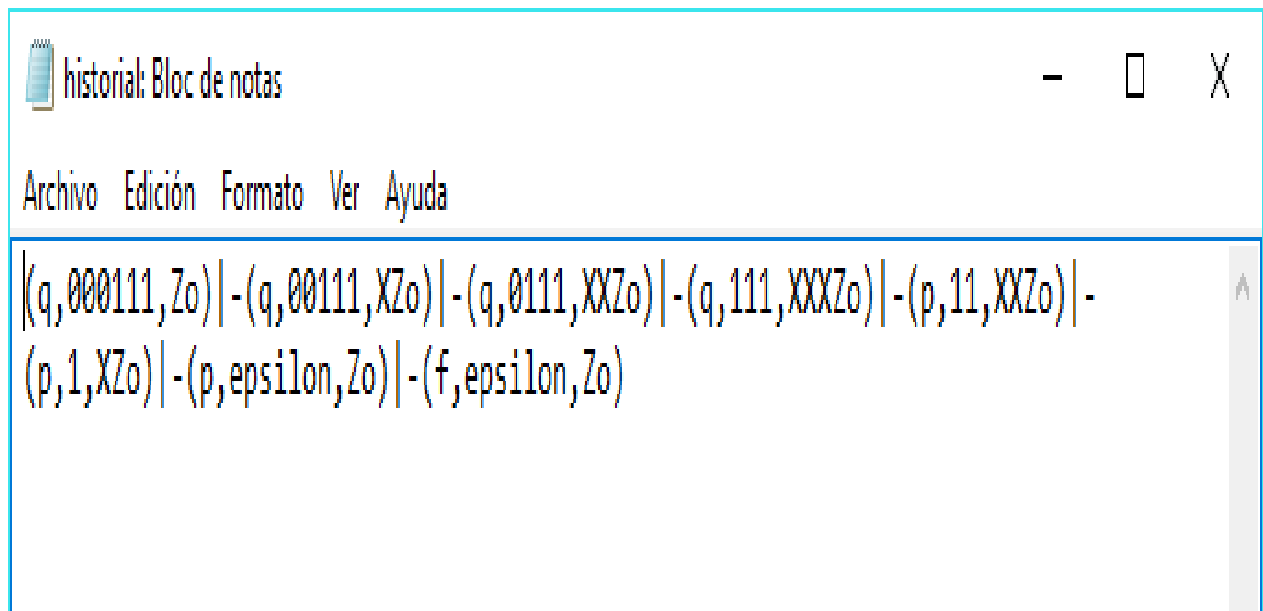


Figura 26: Historial de estados y caracteres por los cuales pasa la pila.

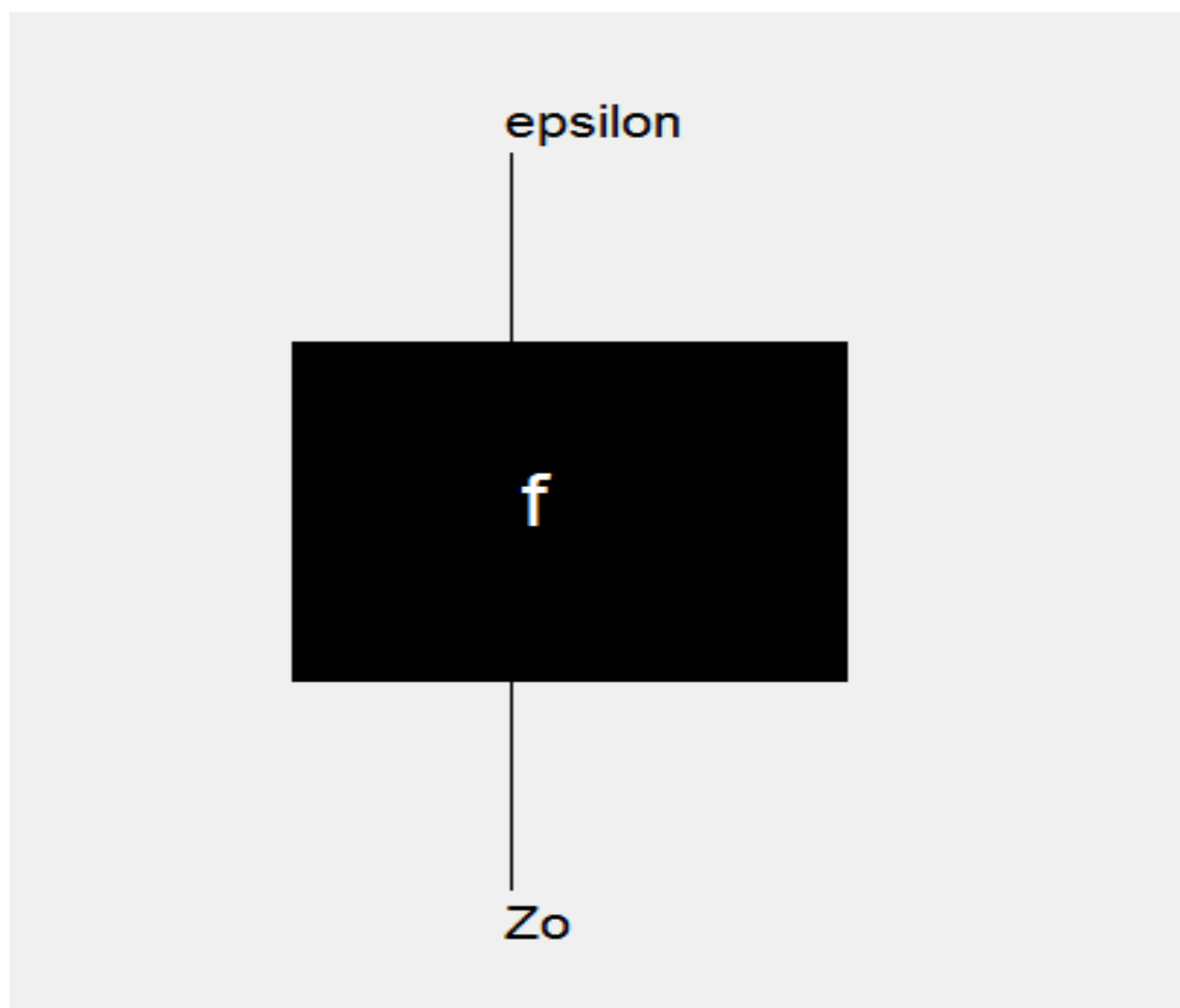


Figura 27: Grafico Pila.

6. Máquina de Turing

6.1. Descripción del Problema

Se desea emplear una máquina de Turing que evalúe si una cadena está compuesta de la siguiente forma

$$0^n 1^n$$

El proceso que seguirá la máquina de Turing es cambiar los ceros por X's y los unos por Y's al final evaluará si la cadena cumple con esa regla. La forma en la que se procesara la cadena será a través de estados los cuales cambiarán los caracteres y se moverán a la derecha o izquierda de la cadena para evaluar la cadena de forma correcta.

6.2. Código fuente

Este programa fue elaborado en el lenguaje Python, su código fuente se muestra a continuación:

Archivo: turing.py

```
from random import *

def automata(c):
    c.append('B')
    pos=0
    f=open("proceso.txt","a")
    estado=0
    mov='_'
    while(pos<len(c)):
        f.write("(q"+str(estado)+"," +c[pos]+"," +mov+")\n")
        f.write(pasarCadena(c.copy(),pos,estado)+"\n")
        if(estado==0):
            if(c[pos]=='0'):
                estado=1
                c[pos]='X'
                pos+=1
                mov='R'
            elif(c[pos]=='Y'):
                estado=3
                c[pos]='Y'
                pos+=1
                mov='R'
            else:
                return 0
        elif(estado==1):
            if(c[pos]=='0'):
                estado=1
                c[pos]='0'
                pos+=1
```

```

        mov= 'R'
    elif(c[pos]== '1') :
        estado=2
        c[pos]= 'Y'
        pos-=1
        mov= 'L'
    elif(c[pos]== 'Y') :
        estado=1
        c[pos]= 'Y'
        pos+=1
        mov= 'R'
    else :
        return 0
elif(estado==2):
    if(c[pos]== '0') :
        estado=2
        c[pos]= '0'
        pos-=1
        mov= 'L'
    elif(c[pos]== 'X') :
        estado=0
        c[pos]= 'X'
        pos+=1
        mov= 'R'
    elif(c[pos]== 'Y') :
        estado=2
        c[pos]= 'Y'
        pos-=1
        mov= 'L'
    else :
        return 0
elif(estado==3):
    if(c[pos]== 'Y') :
        estado=3
        c[pos]= 'Y'
        pos+=1
        mov= 'R'
    elif(c[pos]== 'B') :
        estado=4
        c[pos]= 'B'
        return 1
    else :
        return 0
f.write("Termina_Proceso\n\n")
f.close()

def pasarLista(cadena):
    lista=[]
    for c in cadena:
        lista.append(c)
    return lista

```

```

def pasarCadena(lista ,pos ,estado):
    q="_q"+str(estado)+"_"
    lista.insert(pos,q)
    cad=""
    for elemento in lista:
        cad+=elemento
    return cad

def crearArchivo():
    f=open("proceso.txt","w")
    f.close()

def menu():
    crearArchivo()
    eleccion=0
    while (eleccion!=4):
        eleccion=input("Selecciona una opcion\n1. Manual\n2. Automatico\n3. Salir\nn==>_")
        if(eleccion=='1'):
            print("Opcion_Manual_seleccionada")
            manual()
        elif(eleccion=='2'):
            print("Opcion_Automatica_seleccionada")
            auto()
        elif(eleccion=='3'):
            print("Adios")
            return 0
        else:
            print("Opcion_Invalida ,_Intentalo_de_nuevo")

def manual():
    cadena=input("Ingresa_la_cadena_")
    lista=pasarLista(cadena)
    if(automata(lista)==1):
        print("La_cadena_"+cadena +"_es_Valida")
    else:
        print("La_cadena" +cadena +"_es_Invalida")
    repetir(1)

def auto():
    n=0
    cad=""
    numero=randint(1,1000)
    while(n<numero):
        cad+=choice(['0','1'])
        n+=1
    print("La_cadena_generada_es_"+cad)
    lista=pasarLista(cad)
    if(automata(lista)==1):
        print("La_cadena_"+str(numero) +"_es_Valida")
    else:
        print("La_cadena" +str(numero) +"_es_Invalida")

```

```

    repetir(2)

def repetir(modos):
    rep = ''
    if(modos==1):
        while(rep != '1' and rep != '2'):
            rep = input("Deseas Repetir en este modo\n1. Si\n2. No\n==> ")
            if(rep == '1'):
                print("Seleccionaste repetir")
                manual()
            elif(rep == '2'):
                print("Seleccionaste NO repetir")
            else:
                print("Opcion Invalida , Intentalo de nuevo")
    elif(modos==2):
        rep = choice(['1', '2'])
        if(rep == '1'):
            print("Seleccionaste repetir")
            auto()
        elif(rep == '2'):
            print("Seleccionaste NO repetir")

menu()

```

6.3. Pruebas

Para este programa se realizaron dos pruebas, en su modo automático que generara una cadena binaria la cual luego verificara si cumple con

$$0^n 1^n$$

La otra prueba será en modo manual en la cual el usuario ingresara la cadena para validar.

```
C:\Users\Luis\Desktop>python turing.py
Selecciona una opcion
1.Manual
2.Automatico
3.Salir
==> 2
Opcion Automatica seleccionada
La cadena generada es 10
La cadena2 es Invalida
Seleccionaste NO repetir
```

Figura 28: Modo Automático

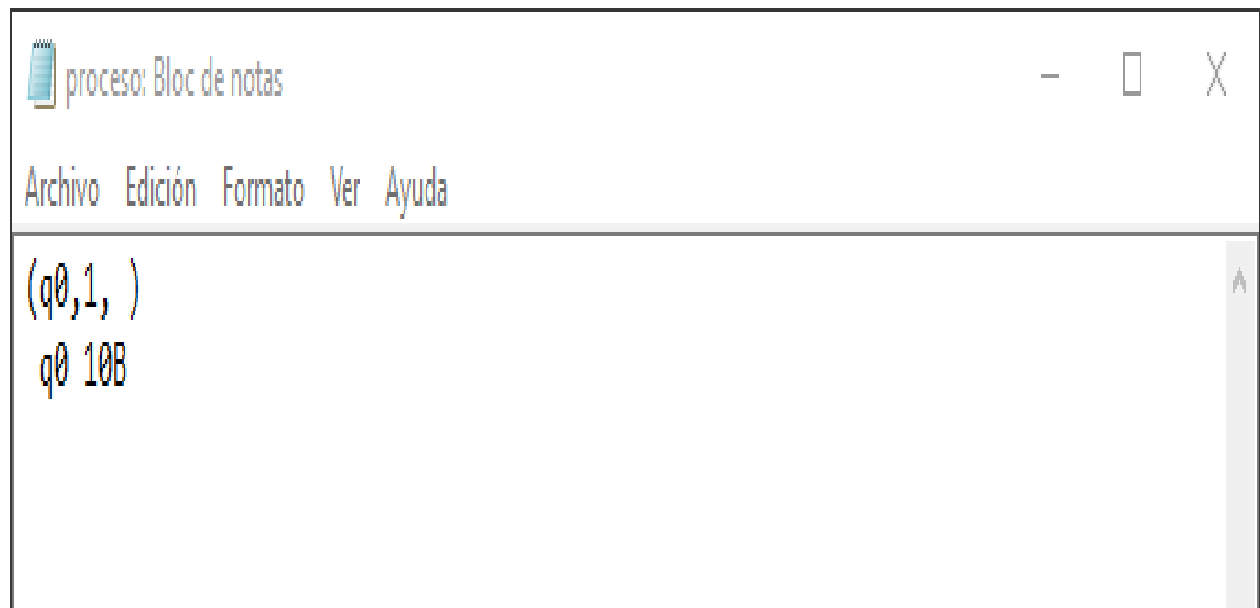
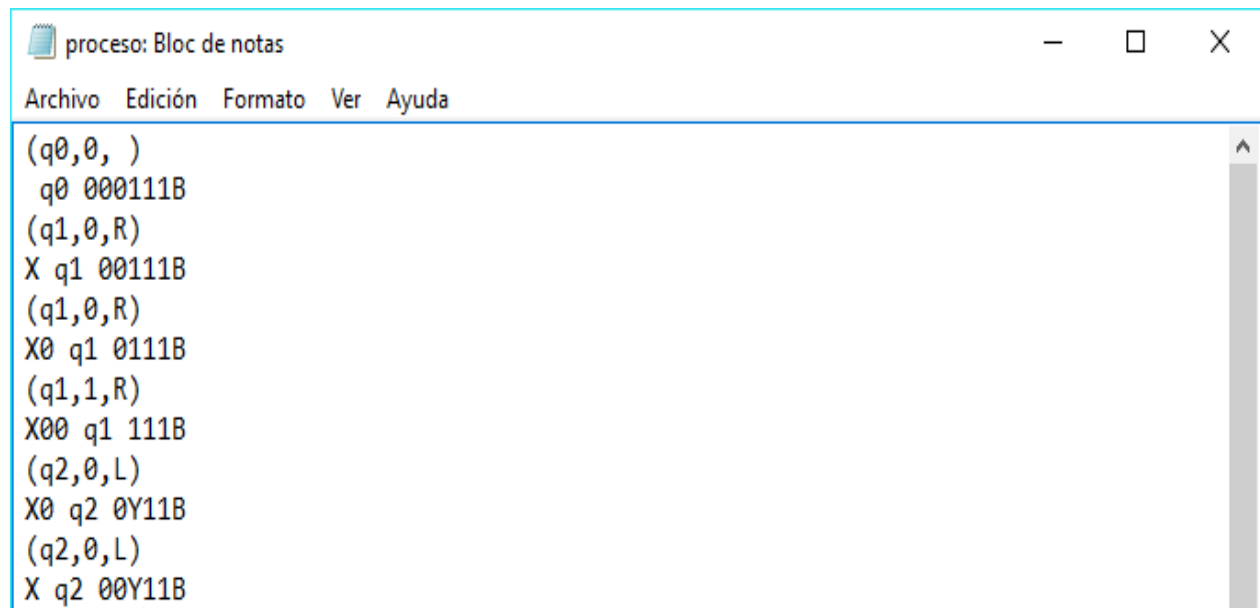


Figura 29: Proceso por el cual pasa la máquina.


```
C:\Users\Luis\Desktop>python turing.py
Selecciona una opcion
1.Manual
2.Automatico
3.Salir
==> 1
Opcion Manual seleccionada
Ingresa la cadena 000111
La cadena 000111 es Valida
Deseas Repetir en este modo
1. Si
2. No
==> 2
Seleccionaste NO repetir
```

Figura 30: Modo Manual



proceso: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
(q0,0, )
q0 000111B
(q1,0,R)
X q1 00111B
(q1,0,R)
X0 q1 0111B
(q1,1,R)
X00 q1 111B
(q2,0,L)
X0 q2 0Y11B
(q2,0,L)
X q2 00Y11B
```

Figura 31: Proceso por el cual pasa la máquina.

Referencias

- [1] John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman. (2007). Introducción a la teoría de autómatas, lenguajes y computación.. Madrid, España: PEARSON EDUCACIÓN.