

Instituto Politécnico Nacional

ESCUELA SUPERIOR DE COMPUTACIÓN

TEORÍA DE LA COMPUTACIÓN

PRÁCTICA 4

Buscador de palabras mediante un AFD

Profesor: Juarez Martinez Genaro

Alumno: Jimenez Luna Rodrigo Efren

Email: jimenez.luna.efren@gmail.com

Grupo: 4CM2

Contents

1	Introducción	2
2	Marco Teórico	2
2.1	Autómatas	2
2.2	AFN a AFD	2
3	Desarrollo	3
3.1	Programa 4 Buscador de palabras mediante un AFD	3
3.2	Planteamiento y solución	3
3.3	Código	4
3.3.1	Código de la función principal (main.py)	4
3.3.2	Código de los métodos utilizados (functions.py)	5
4	Resultados	16
5	Conclusión	18

1 Introducción

Hoy en día los buscadores son una herramienta muy importante en la computación, los podemos observar en proyectos tan grandes como lo es Google, además diversos editores de texto tales como el bloc de notas de Windows, Nano, Vim o hasta un IDE como Visual Studio Code o Pycharm, implementan un buscador de cadenas que permite al usuario encontrar de forma rápida y sencilla ciertas cadenas en específico. Y el uso de un autómata ajusta muy bien para lo que sea deseado lograr con estos buscadores, ya que los autómatas también pueden analizar cadenas de caracteres y comprobar si estas cumplen ciertas características.

A lo largo de la presente práctica se presenta la implementación de un autómata capaz de encontrar 7 palabras clave: web, website, webpage, webmaster, site, page e ebay. Más adelante se plantearán las características específicas del programa.

2 Marco Teórico

2.1 Autómatas

En informática, se entiende por autómata a un modelo matemático creado para una máquina el cuál tiene finitos estados. Este recibe una serie de símbolos y cambia de estado mediante algún tipo de "función". Los autómatas suelen recibir como entrada un alfabeto con el cuál trabajan [?].

Existen diferentes tipos de autómatas, entre ellos están los autómatas finitos deterministas o por sus siglas AFD, lo que significa que solo existe una única posibilidad para el cambio de estado [?].

Formalmente se puede definir a un AFD como una tupla de 5 valores $(Q, \Sigma, q_0, \delta, F)$ donde Q representa todo el conjunto de posibles estados del autómata; Σ como su notación lo indica es el alfabeto que recibe como entrada el AFD; q_0 representa un estado inicial ($q_0 \in Q$); δ se define como la función de transición, la cual dependiendo el estado actual y un valor de entrada nos llevará al siguiente estado y por último $F \subseteq Q$ es un subconjunto de estados finales también conocidos como estados de aceptación [?].

2.2 AFN a AFD

Por definición, los AFD se pueden considerar casos particulares de los AFN, por lo que es posible pasar un AFN (Q, S, d, q_0, F) a un AFD (Q', S, d', q'_0, F') que acepte exactamente el mismo lenguaje que el AFD del que proviene siguiendo ciertas reglas.

La consideración es que el estado inicial q_0 del AFN será considerado como

estado q_0 del AFD, además si un estado p pertenece a los estados del AFN llegando a el a partir del estado inicial siguiendo los símbolos $a_1a_2...a_m$. El conjunto de estados p del AFN se convertirán en estados únicos del AFD [?]

3 Desarrollo

3.1 Programa 4 Buscador de palabras mediante un AFD

En esta práctica se plante el siguiente problema: Desarrollar un buscador de las palabras ('web', 'website', 'webpage', 'webmaster', 'site', 'page', 'ebay') mediante el uso de un autómata finito determinista. El programa aceptara el nombre de un archivo de texto el cual contendrá el texto a analizar o bien, la URL de una pagina web para analizar su contenido. Como salida, se deberán crear dos archivos de texto, uno donde se plasmará las transiciones del autómata al analizar el texto, similar a un historial, mientras que el segundo, deberá mostrar cuantas palabras se encontraron y en que posición del texto. Por último el programa también deberá permitir graficar el autómata.

3.2 Planteamiento y solución

Para la solución de este problema se deberá diseñar el autómata que resuelva la búsqueda, pero ya que esto resulta una tarea complicada, se optó por primero desarrollar un AFN para posteriormente obtener el AFD que acepta el mismo lenguaje que este.

A continuación en la figura 1, se muestra el AFN que se desarrollo para el problema. Obsérvese que este propone un estado diferente para comenzar a buscar cada palabra.

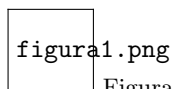


Figura 1. Diagrama del autómata no determinístico que busca las palabras 'web', 'website', 'webpage', 'webmaster', 'site', 'page', 'ebay'. Fuente: Elaboración propia.

Para el siguiente paso se procedió a crear las tablas de transiciones. La primera que se muestra en la Tabla 1, representa las transiciones del AFN, para posteriormente comenzar a formar las uniones y crear una nueva tabla de transiciones para el AFD (Tabla 2), ya que este tendrá los estados que se encuentran en la primera columna de la tabla.

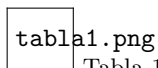


Tabla 1. Tabla de transiciones del AFN. Fuente: Elaboración propia

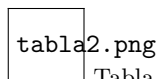


Tabla 2. Tabla de transiciones para los estados del AFD. Fuente: Elaboración propia.

Una vez realizada la tabla de transiciones, solo hace falta implementarlo en una función dentro de nuestro código, para ello solo hace falta tomar los conjuntos de estados que se encuentran en la primer columna de la tabla 2 como los estados del AFD. Ya una vez realizado solo hace falta considerar que cuando se llegue a un estado de aceptación deberá contar la palabra la cuál corresponde. El diagrama del AFD se muestra en la figura 2, este también se puede generar mediante el programa utilizando la opción 2.

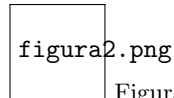


Figura 2. Gráfico del AFD. Fuente: Elaboración propia.

3.3 Código

A partir de la solución planteada en la sección anterior, se muestra el código implementado en el lenguaje de programación python, en este caso se separo en dos archivos diferentes, el primero dedicado para la función principal y el segundo para implementar los métodos:

3.3.1 Código de la función principal (main.py)

```
# IPN – Escuela Superior de Computacion
# Autor: Jimenez Luna Rodrigo Efren
# Grupo: 4CM2

import os
from functions import *

while True:
# Menu
print(" Elija alguna de las siguientes opciones:")
print("1) Buscador de palabras")
print("2) Graficar Automata")
print("3) Salir")

option = int(input())

if option == 1:
word_finder()
elif option == 2:
graph_automata()
elif option == 3:
break
else:
os.system(" cls ")
print("Opcion invalida , por favor vuelva a intentarlo!\n")
```

3.3.2 Código de los métodos utilizados (functions.py)

```
# IPN – Escuela Superior de Computacion
# Autor: Jimenez Luna Rodrigo Efren
# Grupo: 4CM2

import os
import urllib.request
from inscriptis import get_text
import graphviz

# Imprimir resultados
# La siguiente funcion toma el conteo de palabras y posiciones o
# por el AFD y las muestra en un archivo de texto
def print_results(file, words):
    for word in words:
        if 'position' in word and len(words[word]) > 0:
            file.write("La palabra '{}' fue encontrada {} veces en las pos\n".format(word, len(words[word])))
            file.write("\n\n")
        elif 'position' in word:
            file.write("No se encontro la palabra {} en el texto!\n".format(word))
            file.write("\n")

# Buscador de Palabras con un AFD
# La funcion contiene al Automata el cual se encargara de buscar
# 'web', 'webpage', 'website', 'webmaster', 'site', 'page' e 'ebay'
def DFA_word_finder(text):
    record_file = open("record.txt", "w")
    result_file = open("result.txt", "w")
    position = 0

    words = {
        'web': 0,
        'web-positions': [],
        'webpage': 0,
        'webpage-positions': [],
        'website': 0,
        'website-positions': [],
        'webmaster': 0,
        'webmaster-positions': [],
        'page': 0,
        'page-positions': [],
        'site': 0,
        'site-positions': [],
        'ebay': 0,
```

```

        'ebay-positions ': [],
    }

    state = "1"
    prev_state = "1"

    for c in text:
        position += 1

    # Automata Finito Determinista
    if state == "1":
        if c in ("w", "W"):
            state = "1251219"
        elif c in ("e", "E"):
            state = "136"
        elif c in ("s", "S"):
            state = "28"
        elif c in ("p", "P"):
            state = "32"
        else:
            state = "1"
    elif state == "1251219":
        if c in ("w", "W"):
            state = "1251219"
        elif c in ("e", "E"):
            state = "136361320"
        elif c in ("s", "S"):
            state = "28"
        elif c in ("p", "P"):
            state = "32"
        else:
            state = "1"
    elif state == "136":
        if c in ("w", "W"):
            state = "1251219"
        elif c in ("e", "E"):
            state = "136"
        elif c in ("b", "B"):
            state = "137"
        elif c in ("s", "S"):
            state = "28"
        elif c in ("p", "P"):
            state = "32"
        else:
            state = "1"
    elif state == "28":

```

```

if c in ("i", "I"):
    state = "29"
else:
    state = "1"
elif state == "32":
    if c in ("a", "A"):
        state = "33"
    else:
        state = "1"
elif state == "136361320":
    if c in ("w", "W"):
        state = "1251219"
    elif c in ("e", "E"):
        state = "136"
    elif c in ("b", "B"):
        words['web'] += 1
        words['web-positions'].append(position - 2)
        state = "137471421"
    elif c in ("s", "S"):
        state = "28"
    elif c in ("p", "P"):
        state = "32"
    else:
        state = "1"
elif state == "29":
    if c in ("t", "T"):
        state = "30"
    else:
        state = "1"
elif state == "33":
    if c in ("g", "G"):
        state = "34"
    else:
        state = "1"
elif state == "137471421":
    if c in ("w", "W"):
        state = "1251219"
    elif c in ("e", "E"):
        state = "136"
    elif c in ("s", "S"):
        state = "2815"
    elif c in ("p", "P"):
        state = "328"
    elif c in ("a", "A"):
        state = "138"
    elif c in ("m", "M"):

```



```

state = "122"
else:
state = "1"
elif state == "2815":
if c in ("i", "I"):
state = "2916"
else:
state = "1"
elif state == "328":
if c in ("a", "A"):
state = "339"
else:
state = "1"
elif state == "137":
if c in ("w", "W"):
state = "1251219"
elif c in ("e", "E"):
state = "136"
elif c in ("s", "S"):
state = "28"
elif c in ("p", "P"):
state = "32"
elif c in ("a", "A"):
state = "138"
else:
state = "1"
elif state == "122":
if c in ("w", "W"):
state = "1251219"
elif c in ("e", "E"):
state = "136"
elif c in ("s", "S"):
state = "28"
elif c in ("p", "P"):
state = "32"
elif c in ("a", "A"):
state = "123"
else:
state = "1"
elif state == "2916":
if c in ("t", "T"):
state = "3017"
else:
state = "1"
elif state == "339":
if c in ("g", "G"):

```

```

state = "3410"
else:
state = "1"
elif state == "138":
if c in ("w", "W"):
state = "1251219"
elif c in ("e", "E"):
state = "136"
elif c in ("s", "S"):
state = "28"
elif c in ("p", "P"):
state = "32"
elif c in ("a", "A"):
state = "23"
elif c in ("y", "Y"):
words['ebay'] += 1
words['ebay-positions'].append(position - 3)
state = "139"
else:
state = "1"
elif state == "123":
if c in ("w", "W"):
state = "1251219"
elif c in ("e", "E"):
state = "136"
elif c in ("s", "S"):
state = "2824"
elif c in ("p", "P"):
state = "32"
else:
state = "1"
elif state == "139":
if c in ("w", "W"):
state = "1251219"
elif c in ("e", "E"):
state = "136"
elif c in ("s", "S"):
state = "28"
elif c in ("p", "P"):
state = "32"
else:
state = "1"
elif state == "2824":
if c in ("i", "I"):
state = "29"
elif c in ("t", "T"):

```

```

state = "25"
else:
state = "1"
elif state == "30":
if c in ("e", "E"):
state = "31"
else:
state = "1"
elif state == "31":
words["site"] += 1
words['site-positions'].append(position - 3)
if c in ("w", "W"):
state = "1251219"
elif c in ("e", "E"):
state = "136"
elif c in ("s", "S"):
state = "28"
elif c in ("p", "P"):
state = "32"
else:
state = "1"
elif state == "34":
if c in ("e", "E"):
state = "35"
else:
state = "1"
elif state == "35":
words['page'] += 1
words['page-positions'].append(position - 3)
if c in ("w", "W"):
state = "1251219"
elif c in ("e", "E"):
state = "136"
elif c in ("s", "S"):
state = "28"
elif c in ("p", "P"):
state = "32"
else:
state = "1"
elif state == "3017":
if c in ("e", "E"):
state = "3118"
else:
state = "1"
elif state == "3118":
words['website'] += 1

```

```

words['site'] += 1
words['website-positions'].append(position - 6)
words['site-positions'].append(position - 3)
if c in ("w", "W"):
    state = "1251219"
elif c in ("e", "E"):
    state = "136"
elif c in ("s", "S"):
    state = "28"
elif c in ("p", "P"):
    state = "32"
else:
    state = "1"
elif state == "3410":
    if c in ("e", "E"):
        state = "3511"
    else:
        state = "1"
elif state == "3511":
    words['webpage'] += 1
    words['page'] += 1
    words['webpage-positions'].append(position - 6)
    words['page-positions'].append(position - 3)
    if c in ("w", "W"):
        state = "1251219"
    elif c in ("e", "E"):
        state = "136"
    elif c in ("s", "S"):
        state = "28"
    elif c in ("p", "P"):
        state = "32"
    else:
        state = "1"
    elif state == "25":
        if c in ("e", "E"):
            state = "26"
        else:
            state = "1"
    elif state == "26":
        if c in ("r", "R"):
            state = "27"
        else:
            state = "1"
    elif state == "27":
        words['webmaster'] += 1
        words['webmaster-positions'].append(position - 8)

```

```

if c in ("w", "W"):
    state = "1251219"
elif c in ("e", "E"):
    state = "136"
elif c in ("s", "S"):
    state = "28"
elif c in ("p", "P"):
    state = "32"
else:
    state = "1"

try:
    record_file.write("f({}, '{}') -> {}\n".format(prev_state, c, st
except:
    continue
finally:
    prev_state = state

print_results(result_file, words)

record_file.close()
result_file.close()

# Buscador de palabras
# La funci n se encarga de obtener los caracteres del archivo d
# que se desea analizar o el contenido de una pagina web, para p
# ingresarla al automata y encontrar las coincidencias
def word_finder():
    os.system("cls")
    file_direction = input("Introduzca el nombre del archivo o la UR

if file_direction[:4] == "http":
    try:
        html = urllib.request.urlopen(file_direction).read().decode('utf
        text = get_text(html)
    except urllib.error.URLError as e:
        os.system("cls")
        print("Se fallo al conectarse con el servidor.")
        print("Razon: ", e.reason, "\n")
        return
    else:
        file = open(file_direction, "r")
        text = file.read()

DFA_word_finder(text)

```

```

os.system("cls")
print("Busqueda terminada!")

# Grafico del grafico del automata
# La funcion permite visualizar de forma grafica al automata
def graph_automata():
    d = graphviz.Digraph('2')
    d.attr(rankdir='LR')

    d.attr('node', shape='circle')
    d.node('1')

    d.attr('node', shape='doublecircle')
    d.node('137471421')

    d.attr('node', shape='plaintext')
    d.edge('', '1', label='Inicio')

    d.attr('node', shape='circle')
    d.edge('1', '1', label=' - w - e - p - s')
    d.edge('1', '1251219', label='w')
    d.edge('1', '136', label='e')
    d.edge('1', '28', label='s')
    d.edge('1', '32', label='p')

    d.edge('1251219', '1', label=' - w - e - p - s')
    d.edge('1251219', '1251219', label='w')
    d.edge('1251219', '136361320', label='e')
    d.edge('1251219', '28', label='s')
    d.edge('1251219', '32', label='p')

    d.edge('136', '1', label=' - w - e - p - s - b')
    d.edge('136', '1251219', label='w')
    d.edge('136', '136', label='e')
    d.edge('136', '137', label='b')
    d.edge('136', '28', label='s')
    d.edge('136', '32', label='p')

    d.edge('28', '1', label=' - i')
    d.edge('28', '29', label='i')

    d.edge('32', '1', label=' - a')
    d.edge('32', '33', label='a')

    d.edge('136361320', '1', label=' - w - e - p - s')

```

```

d.edge('136361320', '137471421', label='b')
d.edge('136361320', '1251219', label='w')
d.edge('136361320', '136', label='e')
d.edge('136361320', '28', label='s')
d.edge('136361320', '32', label='p')

d.edge('29', '1', label=' - t')
d.edge('29', '30', label='t')

d.edge('33', '1', label=' - g')
d.edge('33', '34', label='g')

d.edge('137471421', '1', label=' - w - e - p - s - a - m')
d.edge('137471421', '1251219', label='w')
d.edge('137471421', '136', label='e')
d.edge('137471421', '2815', label='s')
d.edge('137471421', '328', label='p')
d.edge('137471421', '138', label='a')
d.edge('137471421', '122', label='m')

d.edge('2815', '1', label=' - i')
d.edge('2815', '2916', label='i')

d.edge('328', '1', label=' - a')
d.edge('328', '339', label='a')

d.edge('137', '1', label=' - w - e - p - s - a')
d.edge('137', '1251219', label='w')
d.edge('137', '136', label='e')
d.edge('137', '28', label='s')
d.edge('137', '32', label='p')
d.edge('137', '138', label='a')

d.edge('122', '1', label=' - w - e - p - s - a')
d.edge('122', '1251219', label='w')
d.edge('122', '136', label='e')
d.edge('122', '28', label='s')
d.edge('122', '32', label='p')
d.edge('122', '123', label='a')

d.edge('2916', '1', label=' - t')
d.edge('2916', '3017', label='t')

d.edge('339', '1', label=' - g')
d.edge('339', '3410', label='g')

```

```

d.edge('138', '1', label='    - w - e - p - s - y')
d.edge('138', '1251219', label='w')
d.edge('138', '136', label='e')
d.edge('138', '28', label='s')
d.edge('138', '32', label='p')

d.edge('123', '1', label='    - w - e - p - s')
d.edge('123', '1251219', label='w')
d.edge('123', '136', label='e')
d.edge('123', '2824', label='s')
d.edge('123', '32', label='p')

d.edge('139', '1', label='    - w - e - p - s')
d.edge('139', '1251219', label='w')
d.edge('139', '136', label='e')
d.edge('139', '28', label='s')
d.edge('139', '32', label='p')

d.edge('2824', '1', label='    - i - t')
d.edge('2824', '29', label='i')
d.edge('2824', '25', label='t')

d.edge('25', '1', label='    - e')
d.edge('25', '26', label='e')

d.edge('26', '1', label='    - r')

d.attr('node', shape='doublecircle')

d.edge('26', '27', label='r')

d.edge('27', '1', label='    - w - e - s - p')
d.edge('27', '1251219', label='w')
d.edge('27', '136', label='e')
d.edge('27', '28', label='s')
d.edge('27', '32', label='p')

d.edge('138', '139', label='y')

d.edge('30', '1', label='    - e')
d.edge('30', '31', label='e')
d.edge('31', '1', label='    - w - e - p - s')
d.edge('31', '1251219', label='w')
d.edge('31', '136', label='e')
d.edge('31', '28', label='s')
d.edge('31', '32', label='p')

```



```

d.edge('34', '1', label=' - e')
d.edge('34', '35', label='e')
d.edge('35', '1', label=' - w - e - p - s')
d.edge('35', '1251219', label='w')
d.edge('35', '136', label='e')
d.edge('35', '28', label='s')
d.edge('35', '32', label='p')

d.edge('3017', '1', label=' - e')
d.edge('3017', '3118', label='e')
d.edge('3118', '1', label=' - w - e - p - s')
d.edge('3118', '1251219', label='w')
d.edge('3118', '136', label='e')
d.edge('3118', '28', label='s')
d.edge('3118', '32', label='p')

d.edge('3410', '1', label=' - e')
d.edge('3410', '3511', label='e')

d.edge('3511', '1', label=' - w - e - p - s')
d.edge('3511', '1251219', label='w')
d.edge('3511', '136', label='e')
d.edge('3511', '28', label='s')
d.edge('3511', '32', label='p')

d.render('DFA', view=True)

os.system("cls")

```

4 Resultados

Para demostrar el funcionamiento del programa, se realizaron 2 pruebas con 2 diferentes entradas. Para la primera de ellas, se le dio a analizar al programa un archivo de texto *prueba.txt*, a continuación se muestra un fragmento de este:

”La web ha revolucionado la forma en que interactuamos con el mundo. Desde la creación del primer sitio web en 1991 por Tim Berners-Lee, la web ha crecido a un ritmo asombroso, convirtiéndose en una parte integral de nuestras vidas. Hoy en día, la web es una red global de información y servicios accesibles a través de sitios web, páginas web y tiendas en línea como eBay.

Un sitio web es una colección de páginas web interconectadas que se alojan en un servidor y están disponibles en la web. Un sitio web puede tener diferentes propósitos, como proporcionar información, ofrecer servicios, vender productos o promover una marca. Un sitio web puede ser creado y administrado por un webmaster, que es el encargado de asegurarse de que el sitio web esté funcionando correctamente y se actualice regularmente.

Una página web es un documento digital que se muestra en un navegador web y se compone de texto, imágenes, videos y otros elementos multimedia. Una página web puede ser parte de un sitio web más grande o puede ser una página independiente. Las páginas web se organizan en jerarquías y se enlazan entre sí a través de hipervínculos, lo que permite la navegación de una página a otra.”

Al ingresar el archivo anterior para que el programa lo analizará se obtuvieron los resultados que se muestran en la figura 3 dentro del archivo *results.txt*. Como se puede observar estos resultados concuerdan con los resultados que obtenemos de la herramienta de buscar integrada en el IDE Visual Studio Code. En la figura 4 se puede observar un de estos ejemplos donde la cantidad de palabras 'web' en el archivo coincide con el resultado plasmado por el programa.




figura3.png

Figura 3. Resultados del programa al analizar el archivo de texto prueba.txt. Fuente: Elaboración propia




figura4.jpg

Figura 4. Resultados de la herramienta para buscar dentro de VS Code. Fuente: Elaboración propia

Para la segunda prueba, se optó por elegir una página web que pudiese contener las palabras que puede buscar el AFD, el vínculo del sitio web se puede encontrar aquí [?]. La figura 5 muestra la salida del programa dentro del archivo *results.txt*, del cuál podemos corroborar que son correctos tal como lo muestra la herramienta de buscar incluida dentro del navegador (figura 6).




figura5.png

Figura 5. Resultados del programa tras analizar la pagina web. Fuente: Elaboración propia


 figura6.png

Figura 6. Resultados de la herramienta para buscar dentro del navegador.
Fuente: Elaboración propia

De igual forma podemos observar un ejemplo de algunas secciones del archivo *record.txt* el cuál contiene el historial de transiciones que realizó el autómata, la figura 7 y 8 muestran una parte del historial para el archivo de texto y la pagina web respectivamente.


 figura7.png

Figura 7. Historial de transiciones del autómata tras analizar el archivo de texto.
Fuente: Elaboración propia


 figura8.png

Figura 8. Historial de transiciones del autómata tras analizar el archivo de texto.
Fuente: Elaboración propia

5 Conclusión

Esta práctica resulto una de las más interesantes, comenzando por el hecho de que este programa tiene bastantes aplicaciones en la vida diaria, ya que como se comento en la introducción, hoy en día los buscadores se utilizan para múltiples aplicaciones. Añadido a esto, resulto un gran reto obtener el AFD a partir del AFN, a pesar de ya conocer el proceso para esto, al contener muchos estados de transición, las tablas resultaron bastante grandes por lo que se cometían varios errores al implementar estas tablas en código, no solo para que este funcionara y encontrara las palabras, también para que el programa fuese capaz de mostrar el diagrama del AFD. Para terminar, el programa resultó ser bastante eficiente, ya que es capaz de analizar páginas web enteras en cuestión de segundos, gracias al autómata.