Práctica 01

DOCENTE	CARRERA	CURSO
MSc. Vicente Enrique Machaca Arceda	Escuela Profesional de Ingeniería de Software	Teoría de la computación

PRÁCTICA	TEMA	DURACIÓN
01	Implementación de autómatas	6 horas

1. Datos de los estudiantes

- Grupo: Teoría de la computación - 23
- Integrantes:
  - Bekam Eddy Huaracha Cabrera
  - Daniel Antonio Casas Soto

2. Ejercicios

1. Implemente un programa en Python para la representación de Automatas. Este programa deberá tener las siguientes entradas:
   - Archivo CSV con la representación de transiciones.
   - Estado inicial.
   - Estados finales F.

   Luego, debera generar 2000 palabras de forma aleatoria de diferente tamano, segun el alfabeto del lenguaje, e indicar cuales son aceptadas y cuales no, por el automata. Por ejemplo, esta podria ser la salida (en un archivo de texto) para un automata que reconoce palabras que terminan en 1 con "Lenguaje- 0, 1:

   Listing 1: Definición del resultado esperado

   ```
   >> 00101 -> Aceptado
   >> 00110 -> No Aceptado
   >> 10101 -> Aceptado
   >> 00000 -> No Aceptado
   ```

   Finalmente, debera generar un archivo .dot con contenido en lenguaje Graphviz que represente de forma grafica el automata.

3. Solución

.....

1. Trabajamos un ejemplo con el código presentado por el profesor, para poder utilizar la librería **pandas** para leer un .csv y generar de allí una matriz.

Listing 2: Código base para resolución del problema

```
import pandas as pd

# definicin del automata ####################
delta = pd.read_csv("delta.csv", index_col=0)
alfabeto = ["0", "1"]
estado_inicial = "A"
F = ["C"]
#############################################

palabra_test = "00001"

# evaluar si palabra_test pertence al lenguaje

x = delta.loc[ "A", "0" ]
```

2. Con este código en mente trabajamos en poder resolver el problema.

3. Continuaremos generando las entradas para evaluar si pertenecen al lenguaje del autómata. Para el ejemplo utilizaremos esta tabla de transformaciones.

	0	1
A	B	A
B	A	C
C	A	C

4. En base a esto podremos generar este código en Python para poder trabajar y generar los números aleatorios. Se trabajo este código:

Listing 3: Trabajo de código para construir solución

```
import pandas as pd
import random
import sys

original_stdout = sys.stdout

nombre_archivo = "Trabajo.txt"

with open(nombre_archivo, 'w') as archivo:
    sys.stdout = archivo

```

```
    delta = pd.read_csv("delta.csv", index_col=0)
    alfabeto = ["0", "1"]
    estado_inicial = "A"
    F = ["C"]


    for i in range(0, 2000, 1):
        x = estado_inicial
        longitud = random.randint(10, 50)
        combinacion = ''.join([str(random.choice(alfabeto)) for _ in range(longitud)])
        print(">>", combinacion, " --> ", end="")

        for i in range(0, len(combinacion), 1):
            x = delta.loc[x, combinacion[i]]

        if x in F:
            print("Aceptado")
        else:
            print("No Aceptado")

sys.stdout = original_stdout

print("Resultado generado exportado en", nombre_archivo)
```

Explicaremos las líneas más importantes del código que son base para el funcionamiento del problema.

- En las lineas 1-3 incluimos las importaciones de las bibliotecas necesarias: pandas para manipulación de datos tabulares, random para generar números aleatorios y sys para interactuar con la salida estándar (texto).
- En las lineas 5-10 se inicializa el nombre del futuro archivo y la función 'w', que es escribir; para que se imprima en el archivo los resultados.
- Lineas 12-15, definimos el lenguaje y los estados de nuestro autómata.
- Linea 18, se inicia un bucle para iterar las 2000 palabras para el autómata.
- Lineas 19-22, se crea la lógica para las palabras y en cada iteración, se escriben en el documento y se almacena la palabra en 'combinación'.
- Lineas 24-30, se compara la locación actual en la "matriz deltaçon el resultado que debería estar en esa posición, esto se hace iterando por cada dígito de la palabra.

Ahora nos falta poder trabajar con el código en Graphviz, con un código proporcionado por el profesor, tenemos que hacer que nuestro programa en Python devuelva una entrada como esta:

Listing 4: Definición del resultado esperado para Graphviz

```
A -> B [label = 0 ];
A -> A [label = 1 ];
B -> A [label = 0 ];
B -> C [label = 1 ];
C -> A [label = 0 ];
C -> C [label = 1 ];
```

En base a eso, se agrego una función de un 'for' anidado que itera por cada línea de la matriz 'delta' y nos devuelve los valores de las filas y columnas, agregando a esta un valor (la locación adecuada según el autómata).

Listing 5: Iterador de matriz implementado

```
import pandas as pd
import random
import sys

original_stdout = sys.stdout

nombre_archivo = "Trabajo.txt"

with open(nombre_archivo, 'w') as archivo:
    sys.stdout = archivo

    delta = pd.read_csv("delta.csv", index_col=0)
    alfabeto = ["0", "1"]
    estado_inicial = "A"
    F = ["C"]

    for index, row in delta.iterrows():
        estado_actual = index
        for column, value in row.items():
            simbolo_entrada = column
            estado_siguiente = value
               ## estado_inicial "->" x "[label ="valor"];"
            print(estado_actual, "->", estado_siguiente, "[label =",
                simbolo_entrada,"];")


    print("\n")
    print("-----------------------------------------------------------")
    print("\n")



    for i in range(0, 5, 1):
        x = estado_inicial
        longitud = random.randint(10, 50)
        combinacion = ''.join([str(random.choice(alfabeto)) for _ in range(longitud)])
        print(">>", combinacion, " --> ", end="")

        for i in range(0, len(combinacion), 1):
            x = delta.loc[x, combinacion[i]]

        if x in F:
            print("Aceptado")
        else:
            print("No Aceptado")

sys.stdout = original_stdout

print("Resultado generado exportado en", nombre_archivo)
```

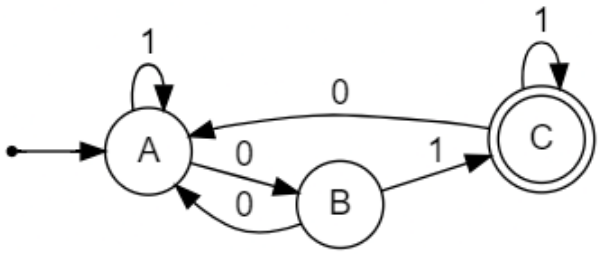
Esta función la veremos ejemplificada en las lineas 17-23, donde recolectara el valor primero en cada fila (row) y luego en cada columna (column), para ejercerlo en el formato que hemos construido para que lo imprima.

5. Con esto ya resuelto, podremos ver que nos da un resultado como este:

Listing 6: Definición del resultado esperado para Graphviz

```
A -> B [label = 0 ];
A -> A [label = 1 ];
B -> A [label = 0 ];
B -> C [label = 1 ];
C -> A [label = 0 ];
C -> C [label = 1 ];


------------------------------------------------------------


>> 10001000111111101100101101101 0011 --> No Aceptado
>> 11011001100011010100000101100100010101001000000001 --> No Aceptado
>> 101100111100 --> No Aceptado
>> 11100110001101010110111101110100011010 --> No Aceptado
>> 01111110011111111101010000100011010001000 11 --> Aceptado...
```

6. Ahora procediendo a pegar la salida que nos interesa en el código .DOT para el graphviz. Tendríamos este autómata determinista.



Este sería el código:

```dot
digraph finite_state_machine {
    fontname="Helvetica,Arial,sans-serif"
    node [fontname="Helvetica,Arial,sans-serif"]
    edge [fontname="Helvetica,Arial,sans-serif"]
    rankdir=LR;
    r [shape = point];
    node [shape = doublecircle]; C;
    node [shape = circle];
    r -> A

    ///Devolución
            A -> B [label = 0 ];
            A -> A [label = 1 ];
            B -> A [label = 0 ];
            B -> C [label = 1 ];
            C -> A [label = 0 ];
            C -> C [label = 1 ];
}
```

7. Esto concluiría con el problema propuesto, sin embargo ahora se hace necesario poder trabajar con el desafío dado en clase:

   - Probar los resultados con un autómata de 5 estados como mínimo y con este alfabeto = 0,1,2.

   Para trabajar con este, solamente necesitariamos determinar una matriz con estos estados y generar el código que se pidio, solamente modificando las entradas para tener el resultado esperado.

Esta sería la matriz pensada:

	0	1	2
A	B	A	B
B	C	F	A
C	F	B	D
D	D	A	F
E	E	G	E
F	G	A	C
G	A	B	A

8. Modificando el código con el nuevo .ªlfabeto", el estado inicial nuevo .ᴬz los estados Finales Çz "G". Ahora podemos general el texto para el trabajo.

Listing 7: Código para autómata con 7 estados

```
import pandas as pd
import random
import sys

original_stdout = sys.stdout

nombre_archivo = "Trabajo.txt"

with open(nombre_archivo, 'w') as archivo:
    sys.stdout = archivo

    delta = pd.read_csv("Ejemplo_7Est.csv", index_col=0)
    alfabeto = ["0", "1", "2"]
    estado_inicial = "A"
    F = ["C", "G"]

    for index, row in delta.iterrows():
        estado_actual = index
        for column, value in row.items():
            simbolo_entrada = column
            estado_siguiente = value
               ## estado_inicial "->" x "[label ="valor"];"
            print(estado_actual, "->", estado_siguiente, "[label =",
                simbolo_entrada,"];")


    print("\n")
    print("----------------------------------------------------------------")
    print("\n")



    for i in range(0, 5, 1):
        x = estado_inicial
        longitud = random.randint(10, 50)
        combinacion = ''.join([str(random.choice(alfabeto)) for _ in range(longitud)])
        print(">>", combinacion, " --> ", end="")

        for i in range(0, len(combinacion), 1):
            x = delta.loc[x, combinacion[i]]
```

```
40
41          if x in F:
42              print("Aceptado")
43          else:
44              print("No Aceptado")
45
46  sys.stdout = original_stdout
47
48  print("Resultado generado exportado en", nombre_archivo)
```

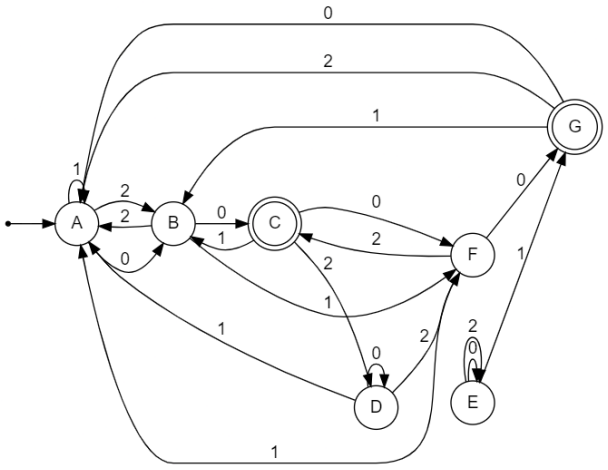
Este código nos genera un archivo con los datos exactos para poder general el autómata en Graphviz, esto demostraría cual sería ese archivo.

Listing 8: Definición del resultado esperado para Graphviz v.2 (7 Estados)

```
    A -> B [label = 0 ];
    A -> A [label = 1 ];
    A -> B [label = 2 ];
    B -> C [label = 0 ];
    B -> F [label = 1 ];
    B -> A [label = 2 ];
    C -> F [label = 0 ];
    C -> B [label = 1 ];
    C -> D [label = 2 ];
    D -> D [label = 0 ];
    D -> A [label = 1 ];
    D -> F [label = 2 ];
    E -> E [label = 0 ];
    E -> G [label = 1 ];
    E -> E [label = 2 ];
    F -> G [label = 0 ];
    F -> A [label = 1 ];
    F -> C [label = 2 ];
    G -> A [label = 0 ];
    G -> B [label = 1 ];
    G -> A [label = 2 ];


    ------------------------------------------------------------


    >> 101122022102 --> No Aceptado
    >> 020212220001201022010211211111211000120201220 --> No Aceptado
    >> 122102212000102022100100002000000021121112110020001 --> No Aceptado
    >> 22022021022121020120022 --> No Aceptado
    >> 012100011002202022212221201020020202000 --> Aceptado
```

Mostraremos cuál sería el resultado tanto en código DOT como el .png del autómata creado por la matriz.

```
digraph finite_state_machine {
    fontname="Helvetica,Arial,sans-serif"
    node [fontname="Helvetica,Arial,sans-serif"]
    edge [fontname="Helvetica,Arial,sans-serif"]
    rankdir=LR;
    r [shape = point];
    node [shape = doublecircle]; C, G;
    node [shape = circle];
    r -> A

    ///Devolución
        A -> B [label = 0 ];
        A -> A [label = 1 ];
        A -> B [label = 2 ];
        B -> C [label = 0 ];
        B -> F [label = 1 ];
        B -> A [label = 2 ];
        C -> F [label = 0 ];
        C -> B [label = 1 ];
        C -> D [label = 2 ];
        D -> D [label = 0 ];
        D -> A [label = 1 ];
        D -> F [label = 2 ];
        E -> E [label = 0 ];
        E -> G [label = 1 ];
        E -> E [label = 2 ];
        F -> G [label = 0 ];
        F -> A [label = 1 ];
        F -> C [label = 2 ];
        G -> A [label = 0 ];
        G -> B [label = 1 ];
        G -> A [label = 2 ];
}
```



9. Se subieron los archivos y los códigos al GitHub para poder trabajar en el repositorio: CT-23 `https://github.com/DACS-SLL/CT-23.git`