Escuela Superior de Cómputo

Teoría de la Computación

Práctica 1:

Universo

Autor:

Rodrigo Gerardo Trejo Arriaga

Octubre 2023

Práctica 1:

Universo

En la teoría de la computación, un alfabeto se define como un conjunto finito de símbolos o caracteres. Estos símbolos son la base fundamental para construir cadenas o secuencias de símbolos. Por ejemplo, en el contexto de la teoría de la computación, es común utilizar el alfabeto 0, 1 para representar símbolos binarios.

Una cadena, también conocida como palabra, es una secuencia finita de símbolos tomados de un alfabeto dado. Estas cadenas desempeñan un papel esencial en la representación de datos y forman la base de los lenguajes formales en la teoría de la computación.

El conjunto universo, por otro lado, hace referencia a un conjunto que contiene todos los objetos posibles que son relevantes para un problema o lenguaje específico. Por ejemplo, si se trata de un lenguaje de programación, el conjunto universo podría consistir en todos los programas concebibles que se pueden escribir en ese lenguaje.

En el contexto de la teoría de la computación, un lenguaje formal se define como un conjunto de cadenas (palabras) construidas a partir de un alfabeto dado. Estos lenguajes formales se utilizan para describir propiedades de las cadenas y son fundamentales en la definición de gramáticas formales, autómatas y en la resolución de diversos problemas relacionados con la informática teórica.

En esta práctica presentaremos un programa que conjunta todos estos conceptos.

I. Descripción del problema

Este programa calcula el universo de cadenas binarias (Σ^n), donde "n.^{es} un valor determinado por el usuario o calculado automáticamente por el programa. El rango de "n.^{es}tá restringido al intervalo [0, 1000].

1. Ejecute el programa y especifique el valor de "n"que desea calcular.
2. El programa preguntará si desea calcular otro valor de "n.º salir.
3. Los resultados se guardarán en un archivo de texto en notación de conjunto.
4. A continuación, graficaremos el número de unos en cada cadena. El eje "xrepresenta las cadenas y el eje zrepresenta el número de unos en cada cadena.
5. En el informe, se explicará, calculará y graficará el caso específico en el que "n=28".
6. Además, se calculará una segunda gráfica utilizando el logaritmo en base 10 del número de unos.

II. Resultados

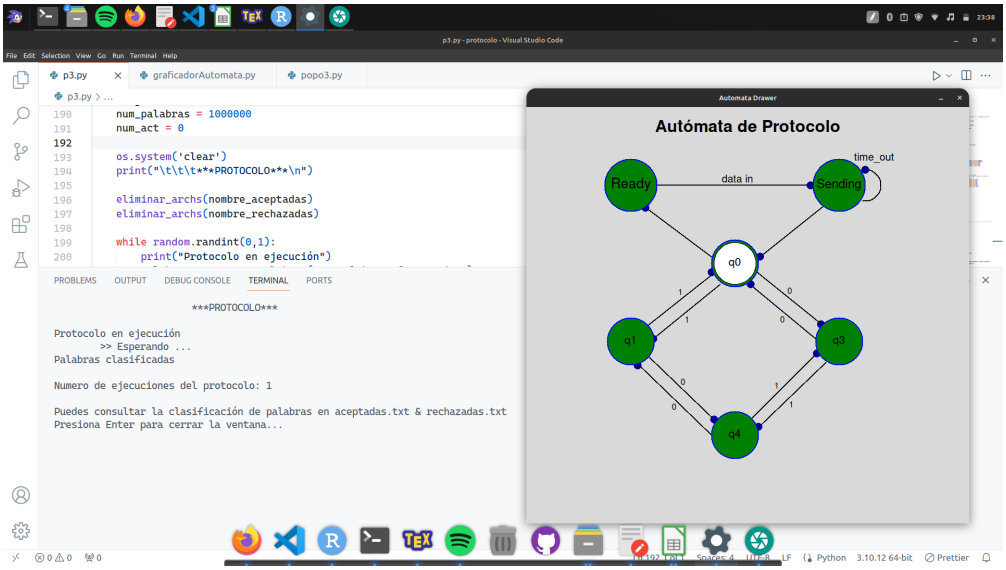

Figura 1: Protocolo con su autómata

III. Código de Implementación

III.1. Programa principal

```
'''
INSTITUTO POLIT CNICO NACIONAL
ESCUELA SUPERIOR DE C MPUTO

INGENIER A EN INTELIGENCIA ARTIFICIAL

TEOR A DE LA COMPUTACI N
PROTOCOLO

GRUPO: 5BM1
ALUMNO: TREJO ARRIAGA RODRIGO GERARDO

ESTE PROGRAMA SIMULA UN PROTOCOLO EN EL QUE CLASIFICA PALABRAS
   BINARIAS SEG N SU PARIDAD:

i) GENERA PALABRAS RANDOM Y LAS CLASIFICA DEPENDIENDO SI LA
   CANTIDAD DE UNOS Y CEROS QUE TIENEN
ES PAR O NO
ii) ESCRIBE LAS PALABRAS CON UNOS Y CEROS PAR EN UN ARCHIVO DE
   TEXTO
iii) GRAFICA EL AUT MATA DEL PROTOCOLO Y EL DE PARIDAD

 LTIMA  MODIFICACI N: 13/10/2023
```

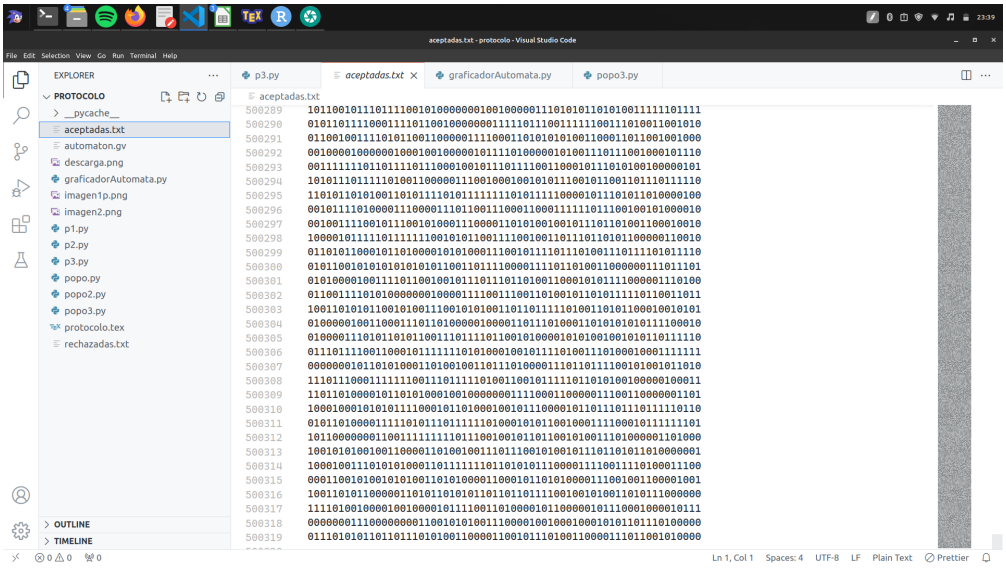

Figura 2: Archivo de aceptadas

```
    '''

    #
        ------------------------------------------------------------------------

    # M DULOS Y LIBRER AS IMPORTADAS


    import random
    import os
    from graficadorAutomata import *

    #
        ------------------------------------------------------------------------

    # FUNCIONES


    def eliminar_archs(nombre_arch):
    """Funci n que elimina un archivo si existe en el directorio

    Args:
    nombre_arch (str): Nombre del archivo que deseas eliminar
    """
    archivo1 = nombre_arch
    if os.path.exists(archivo1):
    os.remove(archivo1)


```

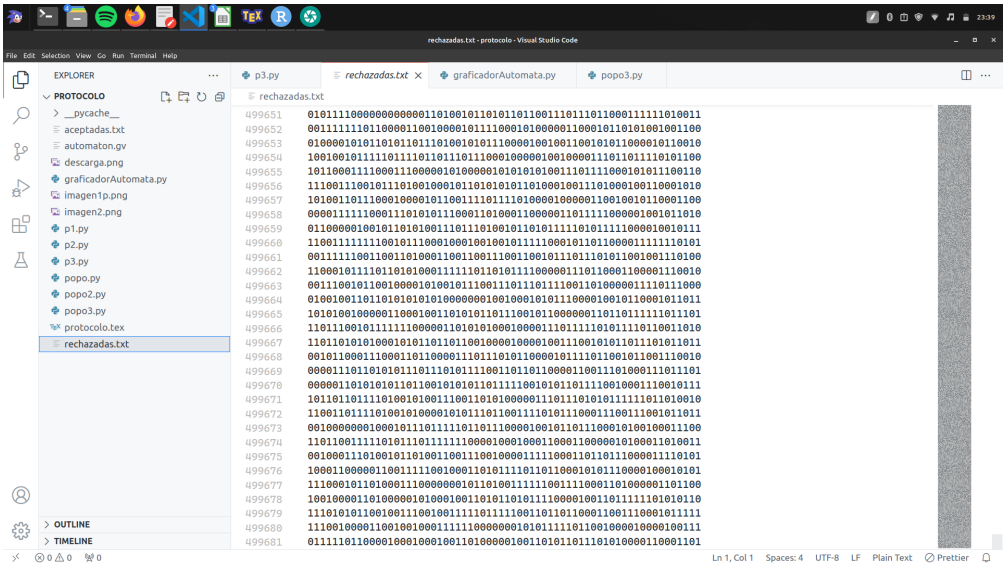

Figura 3: Archivo de rechazadas

```
47  def automata_paridad(transiciones:dict, palabra: str) -> int:
48  """Funci n que simula las trancisiones del aut mata de paridad
        para evaluar una palabra
49
50  Args:
51  transiciones (dict): Diciionario de transiciones del aut mata
52  palabra (str): Palabra que se eval a en el aut mata
53
54  Returns:
55  int: Estado en el que se qued  la palabra
56  """
57  estado = 0
58  for caracter in palabra:
59  estado = transiciones[(estado, caracter)]
60  return estado
61
62
63  def generar_palabras(num_palabras: int, long_cadena:int) -> list:
64  """Funci n que genera la lista de palabras binarias random
65
66  Args:
67  num_palabras (int): N mero de palabras que tendr  la lista
68  long_cadena (int): Longitud de cada palabra binaria
69
70  Returns:
71  list: Lista de palabras binarias
72  """
73  return [''.join([str(random.randint(0, 1)) for _ in range(
        long_cadena)]) for _ in range(num_palabras)]
```

```
74
75
76      def filtrar_palabras(palabras:list, transiciones) -> tuple:
77      """Funci n que filtra palabras binarias seg n su paridad de
           unos y ceros
78
79      Args:
80      palabras (list): Lista de palabras que se evaluar n
81      transiciones (_type_): Diccionario con las transiciones del
           aut mata
82
83      Returns:
84      tuple: Lista de palabras aceptadas y rechazadas por el aut mata
85      """
86      aceptadas = []
87      rechazadas = []
88      for palabra in palabras:
89      if automata_paridad(transiciones, palabra) == 0:
90      aceptadas.append(palabra)
91      else:
92      rechazadas.append(palabra)
93      return aceptadas, rechazadas
94
95
96      def escribir_en_archivo(nombre_arch: str, datos:list) -> None:
97      """Funci n que escribe una lista de palabras en un archiivo
98
99      Args:
100     nombre_arch (str): Nombre del archivo en el que se escribir n
           los datos
101     datos (list): Lista de datos que se escribir n
102     """
103     if datos:
104     with open(nombre_arch, "a+") as archivo:
105     archivo.write("\n ".join(datos))
106     archivo.write("\n ")
107
108
109     def graficar_estados(graficador)-> None:
110     """Funci n que dibuja los estados
111
112     Args:
113     drawer (graphic): Objeto que dibuja un aut mata
114     """
115     graficador.dibujar_estado(200, 150, 2.5, radio=50, flag_inicial=
           False, etiqueta="Ready")
116     graficador.dibujar_estado(600, 150, 2.7, radio=50, flag_inicial=
           False, etiqueta="Sending")
```

```
117     graficador.dibujar_estado(400, 300, 2.7, radio=45, flag_inicial=
            True, etiqueta="q0")
118     graficador.dibujar_estado(200, 450, 2.7, radio=45, flag_inicial=
            False, etiqueta="q1")
119     graficador.dibujar_estado(600, 450, 2.7, radio=45, flag_inicial=
            False, etiqueta="q3")
120     graficador.dibujar_estado(400, 630, 2.7, radio=45, flag_inicial=
            False, etiqueta="q4")
121
122
123     def graficar_transiciones(graficador)->None:
124     """Funci n quen dibuja las transiciones
125
126     Args:
127     drawer (graphic): Objeto que dibuja un aut mata
128     """
129     graficador.dibujar_transicion(250, 150, 550, 150, "data in", 15,
            2, 10, -20)
130     graficador.dibujar_ciclo(650, 150, radio=30, etiqueta="time_out",
             cte_x=17, cte_y=-37, cte_text=0.1)
131     graficador.dibujar_transicion(225, 190, 358, 290, ancho_linea=2)
132     graficador.dibujar_transicion(445, 290, 570, 190, ancho_linea=2)
133     graficador.dibujar_transicion(232, 420, 358, 320, "1",
            ancho_linea=2, cte_y=-26)
134     graficador.dibujar_transicion(243, 445, 372, 340, "1",
            ancho_linea=2, cte_y=35)
135     graficador.dibujar_transicion(570, 420, 440, 315, "0",
            ancho_linea=2, cte_y=-26)
136     graficador.dibujar_transicion(555, 445, 427, 339, "0",
            ancho_linea=2, cte_y=35)
137     graficador.dibujar_transicion(440, 620, 575, 490, "1",
            ancho_linea=2, cte_y=35)
138     graficador.dibujar_transicion(400, 630, 560, 470, "1",
            ancho_linea=2, cte_y=-26)
139     graficador.dibujar_transicion(355, 630, 210, 490, "0",
            ancho_linea=2, cte_y=35)
140     graficador.dibujar_transicion(365, 605, 235, 480, "0",
            ancho_linea=2, cte_y=-26)
141
142
143     def graficar_uniones(graficador) ->None:
144     """Funci n que dinuja las uniones
145
146     Args:
147     drawer (graphic): Objeto que dibuja un aut mata
148     """
149     graficador.dibujar_circulo(545, 150, 7)
150     graficador.dibujar_circulo(650, 120, 7)
```

```
151     graficador.dibujar_circulo(228, 193, 7)
152     graficador.dibujar_circulo(448, 287, 7)
153     graficador.dibujar_circulo(355, 317, 7)
154     graficador.dibujar_circulo(243, 445, 7)
155     graficador.dibujar_circulo(565, 417, 7)
156     graficador.dibujar_circulo(430, 340, 7)
157     graficador.dibujar_circulo(445, 614, 7)
158     graficador.dibujar_circulo(557, 473, 7)
159     graficador.dibujar_circulo(213, 496, 7)
160     graficador.dibujar_circulo(360, 600, 7)
161
162
163     def graficar_protocolo():
164     """Funci n que genera el aut mata
165     """
166     graficador = AutomataDrawer(850, 800)
167     graficador.colocar_titulo("Aut mata de Protocolo")
168     graficar_transiciones(graficador)
169     graficar_uniones(graficador)
170     graficar_estados(graficador)
171     input("Presiona Enter para cerrar la ventana...")
172     graficador.close()
173
174     #
            ------------------------------------------------------------------------
175     # FUNCI N PRINCIPAL MAIN
176
177
178     if __name__ == "__main__":
179
180     transiciones = {
181         (0, '0'): 1, (0, '1'): 3,
182         (1, '0'): 0, (1, '1'): 2,
183         (2, '0'): 3, (2, '1'): 1,
184         (3, '0'): 2, (3, '1'): 0
185     }
186
187     nombre_aceptadas = "aceptadas.txt"
188     nombre_rechazadas = "rechazadas.txt"
189
190     long_cadena = 64
191     num_palabras = 1000000
192     num_act = 0
193
194     os.system('clear')
195     print("\t\t\t***PROTOCOLO***\n")
196
```

```
    eliminar_archs(nombre_aceptadas)
    eliminar_archs(nombre_rechazadas)

    while random.randint(0,1):
    print("Protocolo en ejecuci n")
    palabras = generar_palabras(num_palabras, long_cadena)
    print("\t>> Esperando ...")
    aceptadas, rechazadas = filtrar_palabras(palabras, transiciones)
    escribir_en_archivo(nombre_aceptadas, aceptadas)
    escribir_en_archivo(nombre_rechazadas, rechazadas)
    print("Palabras clasificadas\n")
    num_act += 1

    if num_act != 0:
    print(f"Numero de ejecuciones del protocolo: {num_act}\n")
    print(f"Puedes consultar la clasificaci n de palabras en {
       nombre_aceptadas} & {nombre_rechazadas}")
    else:
    print("El protocolo no se ha podido ejecutar, int ntalo
       nuevamente.")

    graficar_protocolo()
```

III.2. Graficador de Autómata

```
    from graphics import *
    import math

    class AutomataDrawer:
    """Clase que dibuja los elementos de un aut mata de manera
       gr fica
    """


    def __init__(self, ancho:int , alto: int):
    """Constructor de la clase graficadora de aut matas

    Args:
    ancho (int): Ancho de la ventana donde se dibujar  el aut mata
    alto (int): Alto de la ventana donde se dibujar  el aut mata
    """
    self.win = GraphWin("Automata Drawer", ancho, alto)
    self.states = []


```

```
20      def colocar_titulo(self, titulo: str, pos: tuple =(425, 40),
            tam_letra: int =24) -> None:
21      """M todo que a ade el t tulo del aut mata
22
23      Args:
24      titulo (str): t tulo que se colocar  en la gr fica
25      pos (tuple, optional): Coordenadas donde ir  el t tulo.
            Defaults to (425, 40).
26      tam_letra (int, optional): tama o del t tulo. Defaults to 24.
27      """
28      titulo = Text(Point(pos[0], pos[1]), titulo)
29      titulo.setSize(tam_letra)
30      titulo.setStyle("bold")
31      titulo.draw(self.win)
32
33
34      def dibujar_estado(self, x:int, y:int, tam_text:int, radio:int
            =20, flag_inicial:bool=False, etiqueta:str=None, ancho_linea:
            int=2) -> None:
35      """M todo que dibuja un estado del aut mata
36
37      Args:
38      x (int): Coordenada x del centro del c rculo
39      y (int): Coordenada y del centro del c rculo
40      tam_text (int): Tama o del estado del aut mata
41      radio (int, optional): Radio del estado. Defaults to 20.
42      flag_inicial (bool, optional): Bandera que indica si es estado
            inicial. Defaults to False.
43      etiqueta (str, optional): Nombre del estado. Defaults to None.
44      ancho_linea (int, optional): Ancho de la l nea del c rculo.
            Defaults to 2.
45      """
46      estado = Circle(Point(x, y), radio)
47      estado.setWidth(ancho_linea)
48      estado.setFill("green")
49      estado.setOutline("blue")
50      estado.draw(self.win)
51
52      if flag_inicial:
53      circulo_ini = Circle(Point(x, y), radio - 5)
54      circulo_ini.setFill("white")
55      circulo_ini.draw(self.win)
56
57      if etiqueta:
58      text = Text(Point(x, y), etiqueta)
59      text.setSize(int(radio / tam_text))
60      text.draw(self.win)
61
```

```
62      self.states.append(estado)
63
64
65      def dibujar_transicion(self, x1:int, y1:int, x2:int, y2:int,
            etiqueta:str=None, tam_text:int=12, ancho_linea:int=2, cte_x:
            int=0, cte_y:int=0) -> None:
66      """M todo que dinuja la linea de transici n entre estados
67
68      Args:
69      x1 (int): Posicion inicial en x de la l nea
70      y1 (int): Posicion inicial en y de la l nea
71      x2 (int): Posicion final en x de la l nea
72      y2 (int): Posicion final en y de la l nea
73      etiqueta (str, optional): Etiqueta de la transici n. Defaults to
            None.
74      tam_text (int, optional): Tama o de la etiqueta. Defaults to 12.
75      ancho_linea (int, optional): Ancho de la etiqueta. Defaults to 2.
76      cte_x (int, optional): Desplazamiento en x de la etiqueta con
            respecto al centro de la linea. Defaults to 0.
77      cte_y (int, optional): Desplazamiento en y de la etiqueta con
            respecto al centro de la linea. Defaults to 0.
78      """
79      linea = Line(Point(x1, y1), Point(x2, y2))
80      linea.setOutline("black")
81      linea.setWidth(ancho_linea)
82      linea.draw(self.win)
83
84      if etiqueta:
85      x = (x1 + x2 +cte_x) / 2
86      y = (y1 + y2 +cte_y) / 2
87      text = Text(Point(x, y), etiqueta)
88      text.setSize(tam_text)
89      text.draw(self.win)
90
91
92      def dibujar_circulo(self, x:int, y:int, radius:int, color:str="
            darkblue")->None:
93      """Dibuja un c rculo para simular 'la llegada' de una
            transici n
94
95      Args:
96      x (int): Coordenada en x del centro del c culo
97      y (int): Coordenada en y del centro del c culo
98      radius (int): Radio del c rculo
99      color (str, optional): Color del c rculo. Defaults to "darkblue
            ".
100     """
101     circulo = Circle(Point(x, y), radius)
```

```
102     circulo.setFill(color)
103     circulo.setOutline(color)
104     circulo.draw(self.win)
105
106
107     def dibujar_ciclo(self, x:int, y:int, radio:int=20, etiqueta:str=
            None, flag_hor:bool=False, cte_x:int=0, cte_y:int=0, cte_text:
            int=0, grosor_linea:int=2)-> None:
108     """M todo que simula la transici n al mismo estado mediante un
            semic rculo
109
110     Args:
111     x (int): Coordenada en x del centro del semic rculo
112     y (int): Coordenada en x del centro del semic rculo
113     radio (int, optional): Radio del seminc rculo. Defaults to 20.
114     etiqueta (str, optional): Etiqueta de la transici n. Defaults to
            None.
115     flag_hor (bool, optional): Bandera de si el semic rculo ser
            horizontal. Defaults to False.
116     cte_x (int, optional): Desplazamiento en x de la etiqueta con
            respecto al centro del semicirculo. Defaults to 0.
117     cte_y (int, optional): Desplazamiento en y de la etiqueta con
            respecto al centro del semicirculo. Defaults to 0.
118     cte_text (int, optional): Ajusta el tama o del texto. Defaults
            to 0.
119     grosor_linea (int, optional): Grosor de la l nea de transici n.
            Defaults to 2.
120     """
121     num_segments = 20
122     if flag_hor:
123     angulo_ini = 0
124     end_angle = 180
125     else:
126     angulo_ini = -90
127     end_angle = 90
128     incremento = (end_angle - angulo_ini) / num_segments
129
130     for i in range(num_segments + 1):
131     angulo = angulo_ini + i * incremento
132     x1 = x + radio * math.cos(math.radians(angulo))
133     y1 = y + radio * math.sin(math.radians(angulo))
134     angulo += incremento
135     x2 = x + radio * math.cos(math.radians(angulo))
136     y2 = y + radio * math.sin(math.radians(angulo))
137     line = Line(Point(x1, y1), Point(x2, y2))
138     line.setWidth(grosor_linea)
139     line.setOutline("black")
140     line.draw(self.win)
```

```
141
142     if etiqueta:
143     if flag_hor:
144     text = Text(Point(x+cte_x, y+cte_y), etiqueta)
145     text.setSize(int(radio / 2+cte_text))
146     text.draw(self.win)
147     else:
148     text = Text(Point(x+cte_x, y+cte_y - radio * 0.5), etiqueta)
149     text.setSize(int(radio / 2+cte_text))
150     text.draw(self.win)
151
152
153     def clear(self)-> None:
154     """M todo que borra todos los elementos dibujados en la ventana
155     """
156     for state in self.states:
157     state.undraw()
158     self.states = []
159     self.win.update()
160
161
162     def close(self):
163     """M todo que cierra la ventana de la gr fica
164     """
165     self.win.close()
```