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

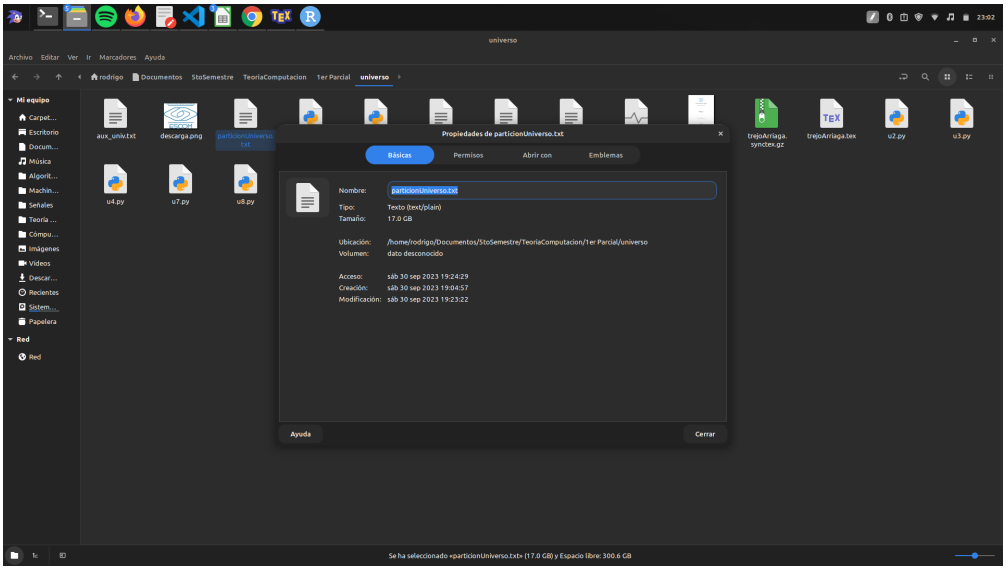

Figura 1: Gráfica de numero vs num de unos

III. Código de Implementación

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

INGENIER A EN INTELIGENCIA ARTIFICIAL

TEOR A DE LA COMPUTACI N
PARTICI N DEL UNIVERSO DE PALABRAS BINARIAS

GRUPO: 5BM1
ALUMNO: TREJO ARRIAGA RODRIGO GERARDO

ESTE PROGRAMA GENERA UNA PARTICI N DEL UNIVERSO DE PALABRAS
   BINARIAS:
i) ESCRIBE LA PARTICI N EN UN ARCHIVO DE TEXTO
ii) GENERA UNA GR FICA DE LAS PALABRAS BINARIAS VS EL
   N MERO DE UNOS QUE TIENEN
iii) GENERA UNA GR FICA DE LAS PALABRAS BINARIAS VS EL
   LOGARITMO DEL N MERO DE UNOS
PARA APRECIAR MEJOR SU CRECIMIENTO

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

#
   ------------------------------------------------------------------
```

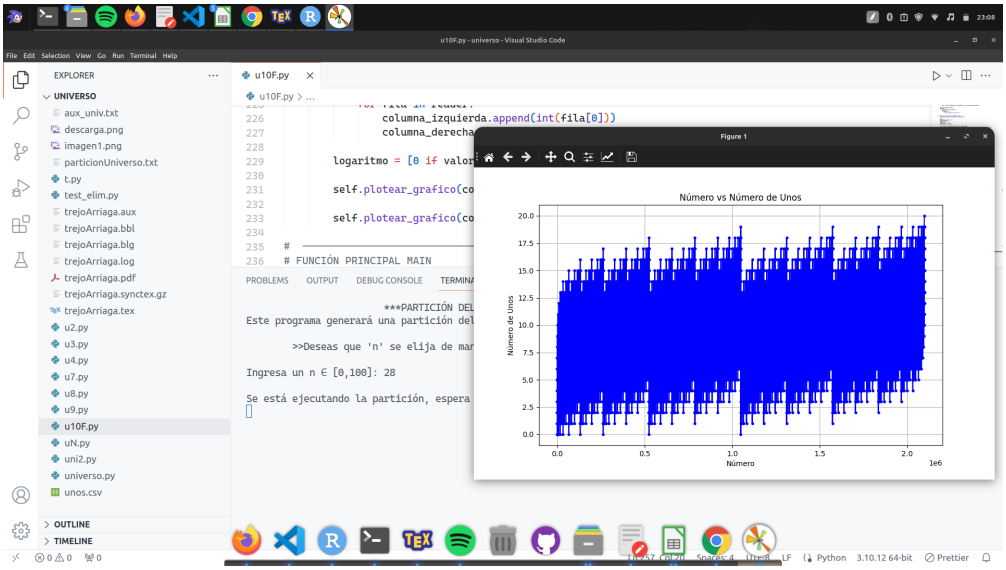

Figura 2:

```
# M DULOS Y LIBRER AS IMPORTADAS


import os
import time
import random
import matplotlib.pyplot as plt
import csv
import math

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

# 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)

#
    ------------------------------------------------------------------
```

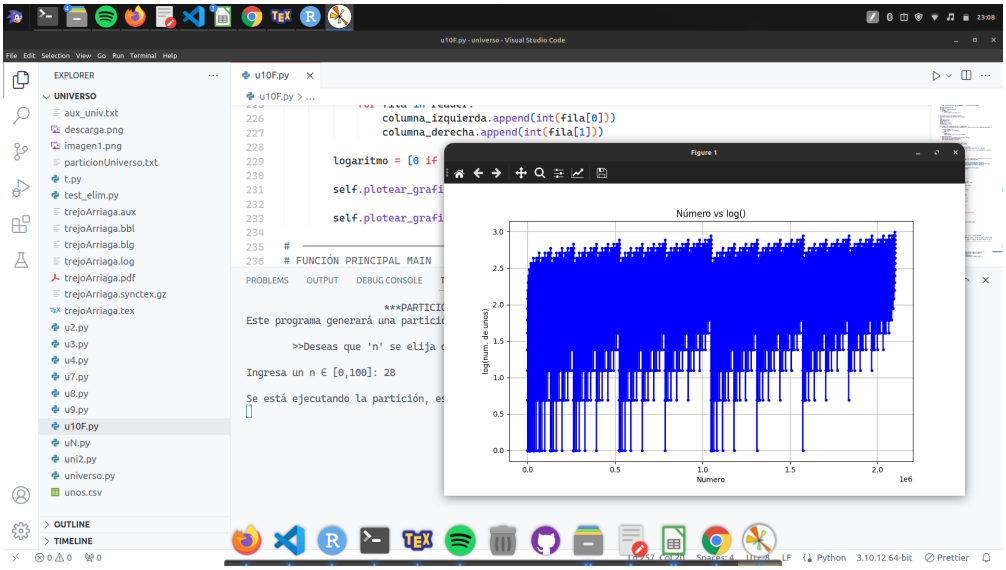

Figura 3: Gráfica Logaritmo

```
# CLASES


class ParticionUniverso:

def __init__(self, tam_particion: int, nombre_arch="
    particionUniverso", nombre_aux="aux_univ.txt") -> None:
"""Constructor de la clase que hace la partici n del
    universo

Args:
tam_particion (int): Tama o de la particion que se desea
    reaizar
nombre_arch (str, optional): Nombre del archivo donde se
    almacena la particion de longitud especificada. Defaults
    to "particionUniverso".
nombre_aux (str, optional): Archivo que sirve como auxiliar
    de las particiones. Defaults to "aux_univ.txt".
"""
self.nombre_arch = nombre_arch
self.tam_particion = tam_particion
self.universo = ["0", "1"]
self.archivo_universo = open(f"{nombre_arch}.txt", "a+",
    encoding="utf-8")
self.archivo_universo.write("   = {  , 0, 1, ")
    self.aux_universo = open(nombre_aux, "a+", encoding="utf
        -8")
    self.contador = 2
```

```
68              self.unos = ["1, 0", "2, 1"]
69              self.num_palabra = [1, 2]
70              self.arch_num_unos = open("unos.csv", "a+", encoding="utf
                    -8")
71
72
73              def escribir_en_archivo(self, datos:list, escribir_coma:
                    bool)-> None:
74              """M todo que escribe datos en el archivo de particion
                    de universo
75
76              Args:
77              datos (list): Datos pertenecientes a la partici n
78              escribir_coma (bool): Bandera para escribir coma al final
                     del uktimo caracter
79              """
80              self.archivo_universo.write(", ".join(datos))
81              if escribir_coma:
82              self.archivo_universo.write(", ")
83
84
85              def escribir_en_archivo_unos(self, escribir_salto: bool)
                    -> None:
86              """M todo que escribe datos en el archivo de particion
                    de universo
87
88              Args:
89              datos (list): Datos pertenecientes a la partici n
90              escribir_coma (bool): Bandera para escribir coma al final
                     del uktimo caracter
91              """
92              self.arch_num_unos.write("\n".join(self.unos))
93              if escribir_salto:
94              self.arch_num_unos.write("\n")
95
96
97              def leer_linea_n(self, n:int) -> list:
98              """M todo que lee la linea n- sima  de un archivo
99
100             Args:
101             n (int): N mero de linea que se desea leer
102
103             Returns:
104             list: Lista de palabras que corresponden a la n- sima
                    linea del archuvo
105             """
106             self.aux_universo.seek(0)
107             contador = 0
```

```
108             for linea in self.aux_universo:
109             contador += 1
110             if contador == n:
111             return linea.strip().split(",")
112 
113 
114             def generar_particion_universo(self) -> None:
115             """M todo que genera una aprtici n del universo
116             """
117             long = 1
118             datos_para_escribir = []
119             conta_llenado = 0
120             leidas = 0
121             bandera_kill = False
122             lim_escritura = 1677721
123 
124             while long < self.tam_particion and not bandera_kill:
125             aux_universo, datos_para_escribir, bandera_kill = self.
                    procesar_palabras(datos_para_escribir, lim_escritura,
                    bandera_kill)
126 
127             if conta_llenado == 0:
128             self.universo = aux_universo
129             long += 1
130             else:
131             leidas += 1
132             self.universo = self.leer_linea_n(leidas)
133 
134             if datos_para_escribir:
135             self.escribir_en_archivo(datos_para_escribir, False)
136             self.escribir_en_archivo_unos(False)
137             self.archivo_universo.write("}")
138 
139 
140         def contar_unos(self, palabra:str, cte_escritura:int):
141         """M todo que cuenta los unos en una palabra binaria
142 
143         Args:
144         palabra (str): Palabra de la que se contar n los unos
145         cte_escritura (int): l mite de cadenas a almacenar antes de
                escribir en el archivo
146         """
147         self.unos.extend([f"{self.contador+1}, {(palabra+'0').count
                ('1')}", f"{self.contador+2}, {(palabra+'1').count('1')}"
                ])
148         self.contador += 2
149 
150         if len(self.unos) > cte_escritura:
```

```
151         self.escribir_en_archivo_unos(True)
152         self.unos = []
153 
154 
155         def procesar_palabras(self, datos_para_escribir:list ,
               lim_escritura:list , bandera_kill: bool) -> tuple:
156         """M todo que genera nuevas palabras y las escribe en el
               archivo de texto
157 
158         Args:
159         datos_para_escribir (list): Datos almacenados en una lista
               antes de su escritura
160         lim_escritura (list): N mero m ximo de elementos en la
               lista antes de escribir en el archivo
161         bandera_kill (bool): Bandera que termina el bucle cuando se
               termina la partici n
162 
163         Returns:
164         tuple: Tupla con la lista auxiliar de la corrida anterior del
                universo,
165         la lista de datos que esperan ser escritos
166         y la bandera de t rmino de ciclo
167         """
168         aux_universo = []
169         conta_llenado = 0
170         for palabra in self.universo:
171 
172         if self.tam_particion == 28:
173         if (self.contador+2) % 1000:
174         self.contar_unos(palabra, lim_escritura)
175         else:
176         self.contar_unos(palabra, lim_escritura)
177         aux_universo.extend([palabra + "0", palabra + "1"])
178         datos_para_escribir.extend([palabra + "0", palabra + "1"])
179 
180         if len(datos_para_escribir) >= lim_escritura:
181         self.escribir_en_archivo(datos_para_escribir, True)
182         datos_para_escribir = []
183 
184         if len(aux_universo) == lim_escritura:
185         self.aux_universo.write(",".join(aux_universo))
186         self.aux_universo.write("\n")
187         conta_llenado += 1
188         aux_universo = []
189 
190         if palabra.count("1") + 1 == self.tam_particion:
191         bandera_kill = True
192 
```

```
193         return aux_universo, datos_para_escribir, bandera_kill
194
195
196         def plotear_grafico(self, x, y, label_x, label_y, titulo):
197         """M todo que plotea una gr fica seg n dos listas
198
199         Args:
200         x (list): Lista de datos que ir n en el eje x
201         y (list): Lista de datos que ir n en el eje y
202         titulo (str): T tulo de la gr fica
203         label_x (str): Etiqueta del eje x
204         label_y (str): Etiqueta del eje y
205         """
206         plt.figure(figsize=(10, 6))
207         plt.plot(x, y, marker='o', linestyle='-', color='b',
               markersize=3)
208         plt.title(titulo)
209         plt.xlabel(label_x)
210         plt.ylabel(label_y)
211         plt.grid(True)
212         plt.show()
213
214
215         def graficar_num_unos(self):
216         """Funci n que hace la gr fica del n mero de unos por
               n mero primo
217         """
218         columna_izquierda = []
219         columna_derecha = []
220
221         with open("unos.csv", 'r+') as archivo_csv:
222         reader = csv.reader(archivo_csv)
223
224
225         for fila in reader:
226         columna_izquierda.append(int(fila[0]))
227         columna_derecha.append(int(fila[1]))
228
229         logaritmo = [0 if valor == 0 else math.log(valor) for valor
               in columna_derecha]
230
231         self.plotear_grafico(columna_izquierda, columna_derecha, '
               N mero', 'N mero de Unos', 'N mero vs N mero de Unos')
232
233         self.plotear_grafico(columna_izquierda, logaritmo, "Numero",
               "log(num. de unos)", 'N mero vs log()')
234
```

```
235 #
    ------------------------------------------------------------------------

236 # FUNCI N PRINCIPAL MAIN
237
238
239 if __name__ == "__main__":
240 os.system('clear')
241 eliminar_archs("particionUniverso.txt")
242 eliminar_archs("aux_univ.txt")
243 eliminar_archs("unos.csv")
244 print("\t\t\t***PARTICI N DEL UNIVERSO***")
245 print("Este programa generar  una partici n del alfabeto
       binario, seg n una longitud 'n'")
246 modo = input("\n\t>>Deseas que 'n' se elija de manera
       autom tica o manual? [a/m]: ")
247 while not modo.lower() != "a" and not modo.lower() != "m":
248 print("Modo inv lido, ingres selo nuevamente para continuar
       ")
249 modo = input("\t>>Deseas que 'n' se elija de manera
       autom tica o manual? [a/m]: ")
250
251 match modo:
252 case "a":
253 n = random.randint(0, 28)
254 print(f"\nSe ha elegido un n={n}")
255 case _:
256 n = int(input("\nIngresa un n     [0,100]: "))
257
258 print("\nSe est  ejecutando la partici n, espera un momento
       .")
259 inicio = time.perf_counter()
260 test = ParticionUniverso(n)
261 test.generar_particion_universo()
262 fin = time.perf_counter()
263 tiempo_transcurrido = fin - inicio
264 test.arch_num_unos.close()
265 test.graficar_num_unos()
266 eliminar_archs("aux_univ.txt")
267 eliminar_archs("unos.csv")
268 print("\nListo, puedes consultar la partici n en el archivo:
        'particionUniverso.txt'.")
269 print(f"Tiempo de generacion de la particion: {
       tiempo_transcurrido:.5f} segundos")
```