UNIVERSIDAD DEL BÍO-BÍO

05 Histograma

Tutorial Procesamiento de Imagen con webcam

Estudiantes Practicantes:
Luis Pereira

Profesor:
Luis Vera

Laboratorio CIMUBB

2023-2

Introducción:

A continuación, se detallará el código que crea una interfaz gráfica que permite al usuario interactuar con un archivo de texto, realizar diversas operaciones y visualizar resultados en un área de texto y gráficos.

Importación de librerías:

```
import tkinter as tk
from tkinter import scrolledtext
import os
import random
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
```

Estas líneas importan las bibliotecas necesarias para la interfaz gráfica (Tkinter), manipulación de archivos y directorios (os), generación de números aleatorios (random), y creación de gráficos (matplotlib).

Funciones de Manipulación de Datos:

1) **limpiar_array_float(lista):** Esta función elimina los elementos vacíos ("") de una lista.

```
def limpiar_array_float(lista):
    return [x for x in lista if x != ""]
```

2) **mostrar_recurrencias()**

```
def mostrar_recurrencias():
    # Se abre el archivo
    with open('na.txt', 'r') as fichero:
        # Se lee el archivo
        arr = fichero.read()
        # Se reemplazan los saltos de línea por nada
        data = arr.replace("\n", "")
        # Se separan los datos por espacio
        data_replace = data.split(" ")
        # Se limpia el array de datos
        mostrar = limpiar_array_float(data_replace)
        # Se crea un diccionario para guardar los datos
        frecuencias = {}
        # Ciclo para guardar los datos en el diccionario
        for i in range(len(mostrar)):
            numero = mostrar[i]
            if numero not in frecuencias:
                frecuencias[numero] = 1
            else:
                frecuencias[numero] += 1
        # Ciclo para ordenar e imprimir los datos
        result = ""
        for numero, frecuencia in sorted(frecuencias.items(), key=lambda x: x[1], reverse=True):
            result += f"{numero}\t{frecuencia}\n"

        return result
```

Propósito: Esta función lee un archivo de texto llamado "na.txt" y cuenta la frecuencia de ocurrencia de cada número en el archivo. Luego, devuelve una cadena de texto que muestra los números ordenados por frecuencia de mayor a menor.

Parámetros: No recibe parámetros.

Retorno: Retorna una cadena de texto que contiene los números ordenados por frecuencia.

3) **mostrar_histograma()**

```python
def mostrar_histograma():
    # Se abre el archivo
    with open('na.txt', 'r') as fichero:
        # Se lee el archivo
        arr = fichero.read()
        # Se reemplazan los saltos de línea por nada
        data = arr.replace("\n", "")
        # Se separan los datos por espacio
        data_replace = data.split(" ")
        # Se limpia el array de datos
        mostrar = limpiar_array_float(data_replace)
        # Se crea un diccionario para guardar las frecuencias
        frecuencias = {}
        # Ciclo para contar las frecuencias
        for i in range(len(mostrar)):
            numero = mostrar[i]
            if numero not in frecuencias:
                frecuencias[numero] = 1
            else:
                frecuencias[numero] += 1

        # Obtener listas separadas para el eje x y el eje y
        numeros, frecuencias_valores = zip(*sorted(frecuencias.items(), key=lambda x: int(x[0])))

        # Crear el histograma con barras más anchas (ajustar el valor de width según sea necesario)
        plt.bar(list(map(int, numeros)), frecuencias_valores, width=0.8)
        plt.xlabel('Intervalo [0,255]')
        plt.ylabel('Frecuencia')
        plt.title('Histograma de Frecuencias')

        plt.show()
```

Propósito: Esta función lee el archivo de texto "na.txt", cuenta la frecuencia de ocurrencia de cada número y crea un histograma visual utilizando la biblioteca **matplotlib**.

Parámetros: No recibe parámetros.

Retorno: No tiene un valor de retorno directo, pero muestra el histograma en una nueva ventana utilizando **plt.show()**

4) **mostrar_contenido_txt()**

```
def mostrar_contenido_txt():
    # Se abre el archivo
    with open('na.txt', 'r') as leer:
        # Se lee el contenido
        return leer.read()
```

Propósito: Esta función lee y devuelve el contenido completo del archivo de texto "na.txt".

Parámetros: No recibe parámetros.

Retorno: Retorna una cadena de texto que contiene el contenido completo del archivo.

5) **crear_archivo_aleatorio():**

```
def crear_archivo_aleatorio():
    with open('na.txt', 'w') as archivo:
        for _ in range(int(numeroUmbra.get())):
            lista = [str(random.randint(0, 255)) for _ in range(int(numeroUmbra.get()))]
            archivo.write(" ".join(lista) + ' \n')  # Agregar un espacio al final de cada fila

    # Habilitar los botones después de crear el archivo
    button_histograma.config(state=tk.NORMAL)
    button_contenido.config(state=tk.NORMAL)
```

Propósito: Esta función crea un archivo de texto llamado "na.txt" con una cantidad de líneas especificada por el usuario (obtenida a través del Spinbox). Cada línea contiene una serie de números aleatorios separados por espacios.

Parámetros: No recibe parámetros.

Retorno: No tiene un valor de retorno directo, pero crea y guarda el archivo "na.txt" en el sistema de archivos.

Manejo de Eventos y Configuración de la Interfaz:

```python
def on_button_click(option):
    try:
        if option == 1:
            result = mostrar_recurrencias()
        elif option == 2:
            result = mostrar_contenido_txt()
        elif option == 3:
            root.destroy()  # Cerrar la ventana principal para salir del programa
        elif option == 4:
            crear_archivo_aleatorio()
            result = "Archivo 'na.txt' creado con números aleatorios."
        elif option == 5:
            mostrar_histograma()
            result = "Histograma mostrado en una nueva ventana."
        else:
            result = ""

        text_area.config(state=tk.NORMAL)  # Hacer el área de texto editable
        text_area.delete("1.0", tk.END)
        text_area.insert(tk.END, result)
        text_area.config(state=tk.DISABLED)  # Deshabilitar el área de texto nuevamente
    except tk.TclError:
        pass  # Capturamos la excepción si la ventana ya ha sido destruida
```

Aquí se configura la ventana principal de la interfaz, se establece el título y las dimensiones. La función **on_button_click()** maneja los eventos asociados a los botones, realizando acciones según la opción seleccionada al hacer clic en algún botón del programa.

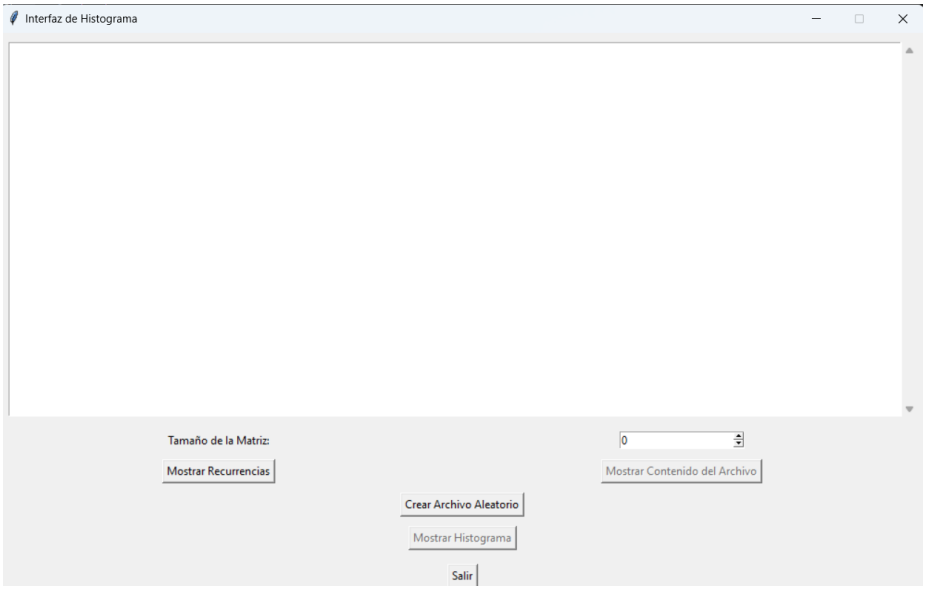
Configuración de Componentes de la Interfaz:

```python
# Configuración del área de texto
text_area = scrolledtext.ScrolledText(root, width=120, height=25, wrap=tk.WORD, state=tk.DISABLED)
text_area.grid(row=0, column=0, columnspan=2, padx=10, pady=10)

# Configuración del Label
label_spinbox = tk.Label(root, text="Tamaño de la Matriz:")
label_spinbox.grid(row=1, column=0, pady=5, padx=10)

# Configuración del SpinBox
numeroUmbra = tk.Spinbox(root, from_=0, to=255)
numeroUmbra.grid(row=1, column=1, pady=5, padx=10)

# Configuración de los botones
button_histograma = tk.Button(root, text="Mostrar Recurrencias", command=lambda: on_button_click(1))
button_histograma.grid(row=2, column=0, padx=10, pady=5)

button_contenido = tk.Button(root, text="Mostrar Contenido del Archivo", command=lambda: on_button_click(2), state=tk.DISABLED)
button_contenido.grid(row=2, column=1, padx=10, pady=5)

button_crear_archivo = tk.Button(root, text="Crear Archivo Aleatorio", command=lambda: on_button_click(4))
button_crear_archivo.grid(row=3, column=0, columnspan=2, pady=5)

button_histograma = tk.Button(root, text="Mostrar Histograma", command=lambda: on_button_click(5), state=tk.DISABLED)
button_histograma.grid(row=4, column=0, columnspan=2, pady=5)

button_salir = tk.Button(root, text="Salir", command=lambda: on_button_click(3))
button_salir.grid(row=5, column=0, columnspan=2, pady=10)
```

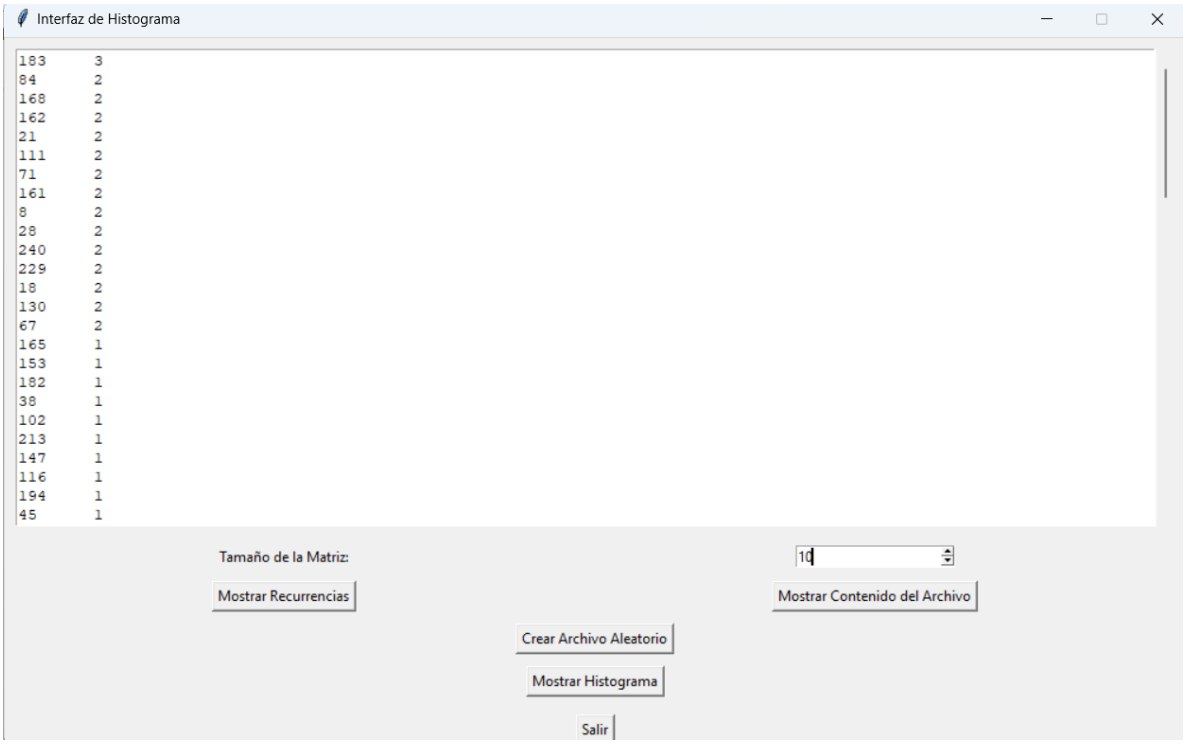
Estas líneas configuran los elementos de la interfaz, como el área de texto, etiquetas, SpinBox y botones.

Resultado Final:

En funcionamiento al establecer el tamaño y hacer clic en mostrar el contenido del archivo:



Al presionar en mostrar recurrencias:



Finalmente, el histograma generado en otra pestaña: