ACTIVIDAD 8: DESARROLLO BACKEND DEL ALGORITMO DE HOFFMAN

Flores Granero Mario Zahit

UNIVERSIDAD DE GUADALAJARA

sección: D04 2024-A

En el reporte del día de hoy será un “complemento” de la actividad anterior donde solo entregamos y vimos el fronted, en este reporte ahora se vera el backend en donde se implementará el algoritmo de Huffman a el fronted que teníamos antes, y a continuación se verán unos ejemplos de cómo es su funcionamiento.

Objetivos

- Unir el backend con el fronted
- Implementar el algoritmo de Huffman

```
class Nodo:
    def __init__(self, caracter, frecuencia):
        self.caracter = caracter
        self.frecuencia = frecuencia
        self.izquierda = None
        self.derecha = None
```

Esta como su nombre dice es la clase para el nodo, en donde lo mas importante seria que izquierda y derecha le “asignamos” un valor “0” o “nulo”

```
def Arbol(simbolos, frecuencias):
    cola_prioridad = [Nodo(caracter, frecuencia) for caracter, frecuencia in fre
    while len(cola_prioridad) > 1:
        cola_prioridad.sort(key=lambda nodo: nodo.frecuencia)
        izquierda = cola_prioridad.pop(0)
        derecha = cola_prioridad.pop(0)
        nuevo_nodo = Nodo(None, izquierda.frecuencia + derecha.frecuencia)
        nuevo_nodo.izquierda = izquierda
        nuevo_nodo.derecha = derecha
        cola_prioridad.append(nuevo_nodo)
    return cola_prioridad[0]
```

Aquí es el desglose del árbol donde se crea una lista y que esta tendrá objetos en este caso de tipo “nodo”, usamos while para que mientas “cola_prioridad” sea menor a 1 este siga ya que queremos que siga hasta quede un solo “nodo”, también se ordena la lista de menor a mayor, extraemos los nodos con frecuencias mas bajas y creamos un nuevo nodo y cuando termina como solo queda un nodo este nodo se devuelve como resultado de la función.

```
def Codigos(arbol, codigo_actual="", codigos={}):
    if arbol.caracter:
        codigos[arbol.caracter] = codigo_actual
    else:
        Codigos(arbol.izquierda, codigo_actual + "0", codigos)
        Codigos(arbol.derecha, codigo_actual + "1", codigos)
    return codigos
```

Creamos los códigos binarios de las letras o los caracteres del árbol de huffman , este recorre desde la raíz al nodo para poder asignar el código.

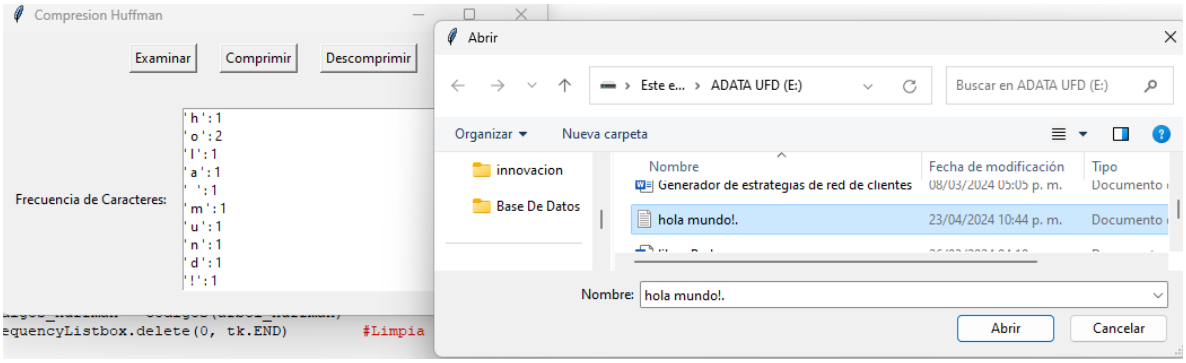
```
def Comprimir_texto(texto, codigos_huffman):
    texto_comprimido = ""
    for caracter in texto:
        texto_comprimido += codigos_huffman[caracter]
    return texto_comprimido

def Descomprimir_texto(texto_comprimido, arbol_huffman):
    texto_original = ""
    nodo_actual = arbol_huffman
    for bit in texto_comprimido:
        if bit == "0":
            nodo_actual = nodo_actual.izquierda
        else:
            nodo_actual = nodo_actual.derecha
        if nodo_actual.caracter:
            texto_original += nodo_actual.caracter
            nodo_actual = arbol_huffman
    return texto_original
```

Bueno como su nombre lo dice es para comprimir el texto o la cadena, se podría decir que estos dos trabajan de la mano para hacer lo que se supone que deben hacer que es comprimir y descomprimir

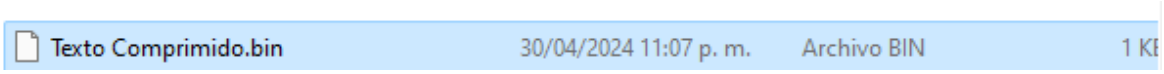
```
def Compresion_Archivo(codigos_huffman):
    filePath = filedialog.askopenfilename()
    if filePath:
        file = open(filePath, 'r')
        content = file.read()
        file.close()
        texto_comprimido = Comprimir_texto(content, codigos_huffman)
        Compresion_Archivo = open("Texto Comprimido.bin", "w")
        Compresion_Archivo.write(texto_comprimido)
        Compresion_Archivo.close()

def Descomprimir_Archivo(arbol_huffman):
    filePath = filedialog.askopenfilename()
    if filePath:
        file = open(filePath, 'r')
        texto_comprimido = file.read()
        file.close()
        texto_original = Descomprimir_texto(texto_comprimido, arbol_huffman)
        Descomprimir_Archivo = open("Texto Descomprimido.txt", "w")
        Descomprimir_Archivo.write(texto_original)
        Descomprimir_Archivo.close()
```

y estas de aquí son para comprimir los archivos y descomprimir, están abren el explorador de archivos, para que lo puedas seleccionar el archivo que vas a comprimir o descomprimir, no se si sea la manera mas optima de hacerlo, pero no

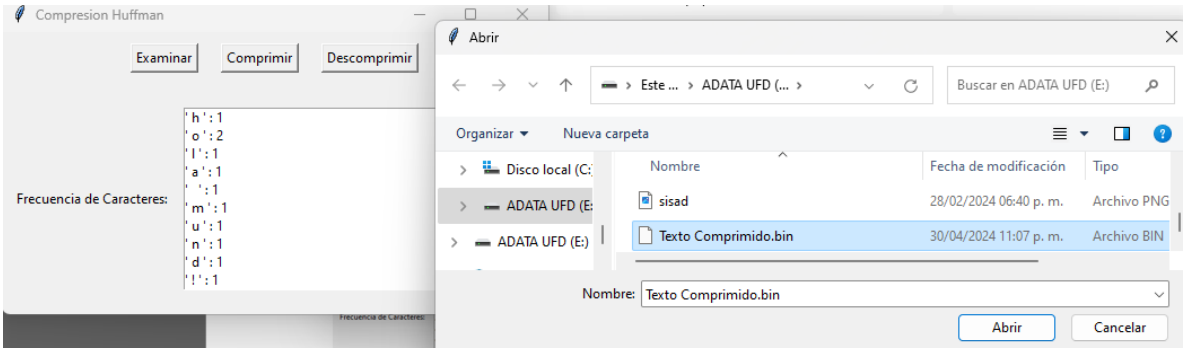
te deja “comprimir” o “descomprimir” otro archivo que no hayas seleccionado, te marcar error entonces eso es algo bueno.



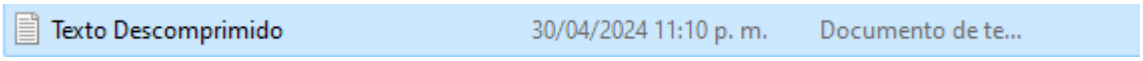
Seleccionamos el archivo que vamos a comprimir en este caso el hola mundo una vez que lo seleccionamos se nos cerrara el explorador de archivos y automáticamente se nos guardara un archivo.bin como se ve a continuación:



Ese es el archivo que se genera.



Después si se presiona el botón de descomprimir abre otra vez en explorador para que selecciones el archivo y lo seleccionamos y como en el anterior nos genera un archivo automáticamente. Como en este ejemplo:



Así es como se muestra

Para concluir quiero decir que la verdad esto si fue bastante complicado y mas la parte del árbol ya que por lo menos para mi fue lo mas difícil de todo, ya que juntarlo no fue difícil el trabajo del compañero fue muy bueno. También quiero mencionar que el único defecto que vi es que a la hora de comprimir los archivos

como se guardan con un el mismo nombre estos van cambiando conforme examinas y comprimes otro, ósea que no puedes quedártelos a menos que antes tu le cambies el nombre al archivo.