UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Seminario de Algoritmia

Reporte de práctica

Nombre del alumno:	Jorge Alejandro Amaro Granero
Profesor:	Erasmo Gabriel Martínez Soltero
Título de la práctica:	"Tarea 2.- Graficar métodos de ordenamientos"
Fecha:	07 Febrero 2022

Introducción

Con el uso de las librerías aprendidas en clase, se nos permite ingresar a nuevas funciones, en este caso, nos apoyan a crear una serie de imágenes las cuales convertimos a MP4. El objetivo por completar consiste en generar un video en base a las iteraciones generadas en las gráficas, y subir el resultado en formato mp4 a YouTube, para comprobar el correcto funcionamiento del código desarrollado.

Metodología

Para la realizacion de este programa, se inicio con el importe de las librerias aprendidas en clase para llevar a cabo la actividad, con estas logramos hacer las graficas, y agregar las animaciones en cada una de estas mientras los valores cambiaban. Se utilizo como arreglo principal el proporcionado en clase, y asi dar un valor igual al de los compañeros y asegurarnos de nuestro correcto desarrollo.

Resultados

```
import random
import numpy as np
import matplotlib.pyplot as plt
import time
import matplotlib.animation as manimation


random.seed(10)
np.random.seed(10)

def bubblesort(lista):
    iterations = 0
    for i in range(len(lista)):
        iterations+=1
        for j in range(0, len(lista) - i - 1):
            iterations+=1
            if lista[j] > lista[j + 1]:
                temp = lista[j]
                lista[j] = lista[j+1]
                lista[j+1] = temp
    return lista, iterations

def mergesort(lista):
    iterations = 0
    if len(lista) > 1:
        mid = len(lista)//2
```

```
        L = lista[:mid]
        R = lista[mid:]
        iterations+=mergesort(L)
        iterations+=mergesort(R)
        i = j = k = 0
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                lista[k] = L[i]
                i += 1
            else:
                lista[k] = R[j]
                j += 1
            k += 1
            iterations+=1

        while i < len(L):
            lista[k] = L[i]
            i += 1
            k += 1
            iterations+=1

        while j < len(R):
            lista[k] = R[j]
            j += 1
            k += 1
            iterations+=1

    return iterations

elementNumArray=[100,200,400,800,1600,3200,6400,12800,25600,51200,102400,204800,20480

elementNumArray=[100,200,400,800,1600,3200,6400,12800]
#elementNumArray=np.arange(100,204800,100)
tiemposBi=[]
iteracionesBi=[]
tiemposLi=[]
iteracionesLi=[]

FFMpegWriter = manimation.writers['ffmpeg']

metadata = dict(title='BubbleSort vs MergeSort', artist='Jorge Amaro',comment='Deja t
writer = FFMpegWriter(fps=24, metadata=metadata)

fig, (ax1, ax2) = plt.subplots(1, 2,figsize=(18,9))
ax1.title.set_text('BubbleSort')
ax2.title.set_text('MergeSort')
```

```
ax1.set_ylabel("Consultas")
ax1.set_xlabel("# Elementos")
ax2.set_xlabel("# Elementos")

with writer.saving(fig, "BubbleSortvsMergeSort.mp4", 100):

    plt.ion()
    for idx,elementNum in enumerate(elementNumArray):
        lista=np.sort(np.linspace(0,100000,elementNum))

        start=time.time()
        # Almacena el arreglo de datos ordenados
        # res = metodo(lista de python), para convertir numpy arr a lista de python
        # usa lista.tolist()
        r_bubblesort,it=bubblesort(lista.tolist())
        finish=time.time()
        iteracionesBi.append(it)
        tiemposBi.append(finish-start)

        start=time.time()
        it=mergesort(lista.tolist())
        finish=time.time()
        iteracionesLi.append(it)
        tiemposLi.append(finish-start)

        ax1.plot(elementNumArray[:idx+1], iteracionesLi, 'r-',label='merge')
        ax2.plot(elementNumArray[:idx+1], iteracionesBi, 'b--',label='Bubble')

        fig.canvas.draw()
        fig.canvas.flush_events()
        plt.show(block=False)

        time.sleep(.1)
        for i in range(24):
         writer.grab_frame()
```

Link del resultado

https://youtu.be/rN9WA4IHalY

Conclusiones

El desarrollo lógico que vamos generando en Python me ha sido de mucho agrado, debido a que logramos por nuestra cuenta aprender de nuestros errores y obtener un resultado grafico por el momento, para así confirmar nuestra correcta ejecución del código. Se me complico el desarrollo de la gráfica debido a que esta continuamente daba lineal, algo raro debido a los cambios constantes en cada iteración, lo que supongo es que no logre dominar el tema de las iteraciones, y solo estoy imprimiendo una tras otra igual. Me gustó mucho la práctica debido a que se retomaron temas de estructura de datos 1, los cuales ya se me habían olvidado un poco.

Referencias

1. Overleaf. (s. f.). Lists. Overleaf, Editor de LaTeX online. Recuperado 05 de febrero de 2022, de https://es.overleaf.com/learn/latex/ListsExam ples$_o f_b asic_l ists$

2. Overleaf. (s. f.-a). Including Graphics on Overleaf. Overleaf, Editor de La-TeX online. Recuperado 05 de febrero de 2022, de https://es.overleaf.com/learn/how-to/Including$_i mages_o n_O verleaf$

3. Alameda, J. (2022, 16 enero). Operadores en Python: Aritméticos, comparación, lógicos, asignación, . . . J2LOGO. Recuperado 05 de febrero de 2022, de https://j2logo.com/python/tutorial/operadores-en-python/