

# UNIVERSIDAD DE GUADALAJARA



## CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

INGENIERO EN COMPUTACIÓN

### ANÁLISIS DE ALGORITMOS

**MACIEL VARGAS OSWALDO DANIEL**

**GARCÍA SALDIVAR HUGO GABRIEL**

Actividad en equipos

*Visualizador de métodos de Ordenamiento*

## I. Introducción

Para comprender adecuadamente el funcionamiento de un algoritmo de ordenamiento y la importancia del tiempo en su ejecución, necesitamos visualizar como ocurre este proceso en el que se intercambian los elementos ya sea por una comparación o por el agrupamiento de los datos de menor a mayor. Para progresar en la implementación de una GUI, añadiremos funciones importantes para hacer un programa más completo y funcional.

## II. Objetivos

El objetivo de nuestro programa es diseñar una GUI interactiva donde principalmente se pueda visualizar como nuestros elementos representados por barras, se van ordenando en una gráfica dependiendo de que algoritmo de ordenamiento se esté usando. En esta gráfica el eje x corresponde al elemento y el eje y al valor de este mismo.

Tendremos diferentes botones en una nuestra GUI para interactuar. Primero vamos a ingresar un número de datos a generar, y seleccionaremos el botón correspondiente. Una vez generados los datos, podremos seleccionar un algoritmo y ejecutar el programa, a partir de aquí se empezarán a ordenar las barras. Para una mejor comprensión y visualización del proceso, añadiremos un scale de 0 a 200 ms, el cual nos permitirá modificar la velocidad de la visualización del ordenamiento.

## III. Desarrollo

Partiendo de la base que ya teníamos, es decir el código que ya ordenaba visualmente, pero con una cantidad de datos fijas y solo con selection sort, empezamos a modificar los solicitado para esta actividad. Lo primero que integramos, fueron otros 3 algoritmos, Bubble Sort, Merge Sort y Quick Sort, esto lo hicimos a través de una función que selecciona el algoritmo, dependiendo del que se elija en un dropdown.

Una vez implementados los algoritmos, continuamos con los botones solicitados. El primer botón que integramos fue el de mezclar (shuffle), este fue sencillo, ya que es un botón que llama a una función que dentro de ella modifica a la lista de datos con otra función predefinida de Python llamada shuffle(). Una vez realizado esto, continuamos con el botón para generar una cantidad distinta de datos. Esto se logró gracias a un entry, en el que después de asignar un valor, este se

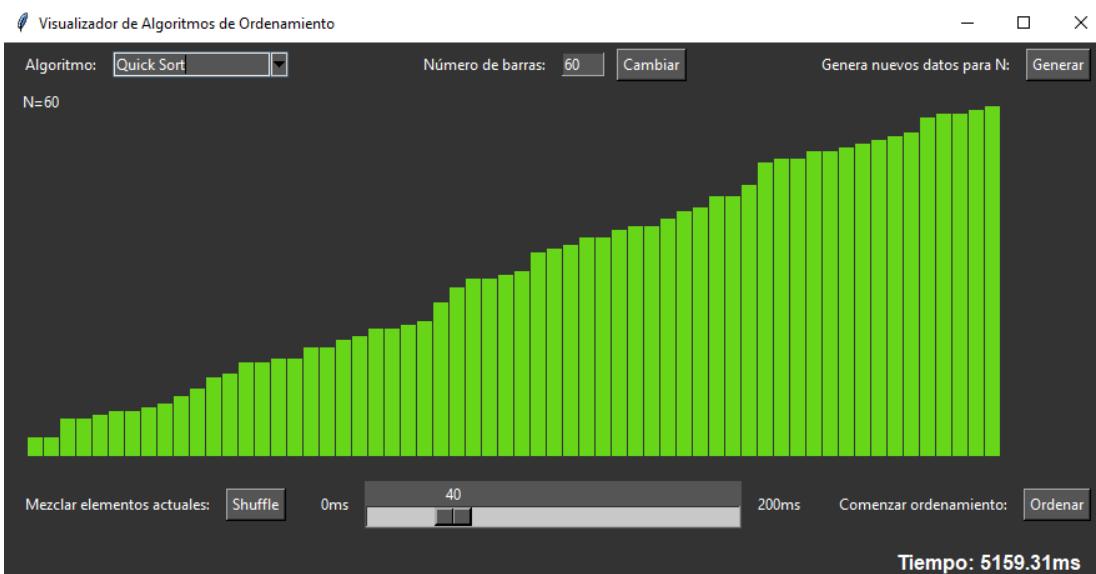
toma y se pasa a una función llamada por un botó, donde dentro de ella se validan ciertos valores y el valor del entry se asigna en nuestra constante que define el número de barras, después de esto ahí mismo se llamada a la función generar() la cual ya estaba, esta utiliza el valor de N\_BARRAS, para crear nuevos datos y después llama ahí mismo a la función dibujar\_barras() para mostrarlas.

Casi para terminar integramos el scale, una barra que va de 0 ms a 200 ms, donde 200 ms representa el tiempo entre cada movimiento en la visualización del algoritmo, esto se logró gracias a que el valor del scale se pasa a la función paso y en tiempo de ejecución se puede ir modificando la velocidad.

En la siguiente primera captura se observa la interfaz general del código, aquí se puede seleccionar entre los 4 algoritmos de ordenamiento, definir el numero de barras (3 a 1000), generar nuevos valores para las barras sin mezclarlas, un botón en la parte inferior para mezclar las barras sin cambiar los datos, el scale para definir el retraso antes o durante la ejecución y el botón para comenzar a ordenar los elementos:



En la segunda captura se muestra un arreglo de 60 elementos ya ordenados mediante el algoritmo quick sort:



En esta tabla podemos ver el tiempo que toma cada algoritmo en ordenar una N cantidad de elementos

N	Selection Sort	Bubble Sort	Quick Sort	Merge Sort
10	21.27 ms	20.69 ms	17.82 ms	28.64 ms
25	191.34 ms	242.09 ms	105.57 ms	199.64 ms
50	1157.00 ms	1357.59 ms	408.92 ms	683.02 ms
75	4176.09 ms	4044.55 ms	1362.02 ms	1435.00 ms
100	8496.43 ms	8045.71 ms	2003.80 ms	2524.65 ms
500	X	X	73168.85 ms	75287.79 ms

Gracias a esta tabla podemos darnos cuenta de que cosas muy curiosas: los algoritmos selection y bubble son bastante similares entre sí, sin embargo, al tratar de ordenar un arreglo de muchos elementos comienzan a ser significativamente ineficientes, por ejemplo; en la tabla al querer acomodar 500 elementos con un algoritmo selección o bubble el tiempo requerido era tan grande que preferimos cerrar el programa.

Similar a esto, los algoritmos quick y merge son bastante parecidos en sus tiempos de ordenamiento, siendo quick el más rápido significativamente entre estos 4 algoritmos. En nuestros casos de prueba el quick fue el más rápido respecto a N en comparación con merge que con pocos elementos tenía tiempos similares a selection y bubble, por lo que lo mejor sería usar merge con

una gran cantidad de elementos. Algo interesante es que quick y merge ordenan 100 elementos en una cuarta parte del tiempo en lo que lo hace selection y bubble por lo que podemos concluir que son algoritmos muy rápido para una gran cantidad de elementos, sobre todo quick (asi como para pocos) mientras que merge es bueno solo con una gran cantidad de elementos (con pocos no es malo, pero hay implementaciones más sencillas).

## IV. Conclusión

Oswaldo Maciel:

Después de realizar esta práctica, pude comprender el funcionamiento de manera visual en un algoritmo, y la importancia que tiene el tiempo en estos. Sinceramente no me pareció una actividad difícil de implementar, ya que la primera actividad que hicimos nos dio un empujón respecto al uso de Python. Cada vez me parece más sencillo implementar una GUI, espero poder realizar actividades más complejas y seguir mejorando mis conocimientos en algoritmos, Python y diseño de GUI.

Hugo Garcia:

Esta practica fue de mi agrado porque previo a este curso habia visto algunos videos de esta forma para visualizar los métodos de ordenamiento (usando barras) y creo que es una manera muy acertada para que las personas entiendan y comprendan como funcionan y hacen los cambios cada método de ordenamiento, asi como comprender en que contexto es mejor el utilizar un método u otro, y finalmente me siguen sorprendiendo los usos y funciones que podemos utilizar mientras construimos un código con python.

## V. Referencias

- Moore, A. D. (2018). Python GUI Programming with Tkinter: Develop Responsive and Powerful GUI Applications with Tkinter. Packt Publishing
- Domínguez Minguez, T. (2025). Tkinter. Desarrollo de interfaces gráficas con Python (2.<sup>a</sup> ed.). [Editorial desconocida]
- Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2013). Data Structures and Algorithms in Python. Hoboken, NJ: Wiley.