UNIVERSIDAD PANAMERICANA®

Proyecto 2° parcial

Interfaz gráfico algoritmos de ordenamiento

Nombre de los alumnos:

Díaz Álvarez Sebastián

Herrera Guerrero Aarón Alejandro

Fecha:

22 de abril del 2024

Materia:

Estructuras de Datos y Algoritmos I

Nombre del profesor:

Ricardo Tachiquín Gutiérrez

A lo largo del 2° parcial, en la materia de Estructuras de Datos y Algoritmos I, se estuvieron trabajando y analizando 9 distintos algoritmos de ordenamiento principalmente para ordenar listas aleatorias de menor a mayor. Dichos ordenamientos o *"sorts"* fueron:

1. *Bubble sort*
2. *Insertion sort*
3. *Selection sort*
4. *Shell sort*
5. *Heap sort*
6. *Merge sort*
7. *Counting sort*
8. *Quick sort*
9. *Radix sort.*

De manera que, como proyecto para este 2° parcial, desarrollamos una interfaz gráfica (GUI) en Python, utilizando el módulo de Tkinter, en la que por medio de animaciones pudimos ver el progreso de cada sort al ordenar listas aleatorias, así como sus gráficas Big O y respecto al tiempo que tardan en ejecutarse.

De igual manera, implementamos el uso de un algoritmo de ordenamiento no visto en clase **"*Comb sort*",** siendo este el décimo ordenamiento con el que se va a trabajar en el proyecto.

A continuación, se plantearán las instrucciones para el uso de la interfaz gráfica mencionada anteriormente, así como algunas advertencias durante su uso y el rol de cada integrante en el proyecto.

EXPLICACIÓN INTERFAZ GRÁFICA

- ESCOGER VENTANA

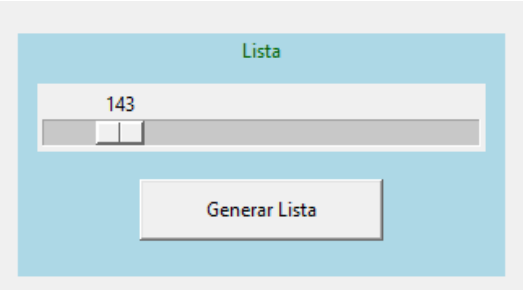
Al principio de la GUI van a aparecer dos botones con los cuales el usuario puede escoger cual ventana quiere ver, ya sea en donde se muestra la gráfica con el progreso del ordenamiento (click en **Gráfica ordenamiento**), o la en donde se muestran las gráficas de complejidad de cada algoritmo (click en **BIG O**).

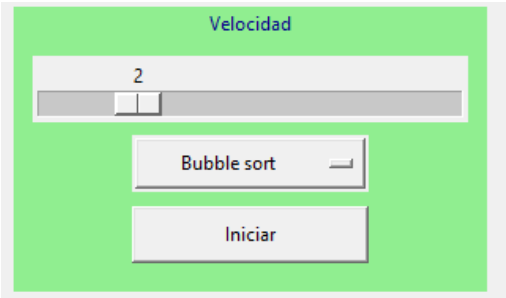
1. VENTANA 1: GRÁFICA ORDENAMIENTO

En esta ventana se desplegarán distintos widgets (herramientas) para poder generar una lista aleatoria (de un rango de 10 a 1000 valores no repetidos) y a partir de dicha lista ejecutar y visualizar el progreso del sort que el usuario decida, teniendo la opción de manipular detalles como la velocidad.

En un principio no van a aparecer muchas opciones en la GUI, solamente se puede apreciar el título “Algoritmos de Ordenamiento”, además de una etiqueta con un saludo de bienvenida, un espacio gris (donde se va a mostrar la gráfica de progreso de cada sort), un botón (**Generar lista**) y una barra deslizadora.

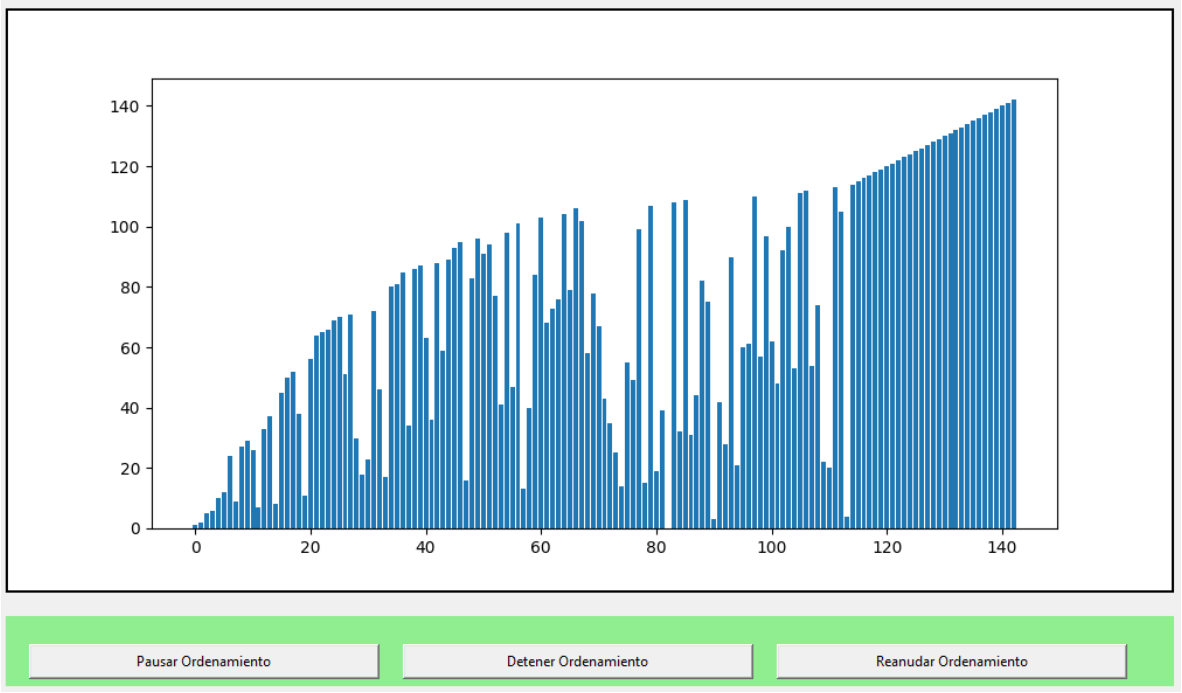
Para iniciar con el código, se escoge el rango de la lista (de 10 a 1000) con la **barra deslizadora** para posteriormente dar click en el botón generar lista, obteniendo así la lista con la que se va a estar trabajando, además de que va a aparecer en forma de gráfica en el espacio gris. Se puede generar una lista nueva cuantas veces quiera el usuario.





Después de dar click en generar lista, aparecerá una lista despegable, otro botón (**Iniciar**) y otra barra deslizadora. Con la **barra deslizadora** se asigna la velocidad con la que se va a estar ejecutando el algoritmo, siendo en este caso más rápido entre menor sea el valor asignado (que es en microsegundos).

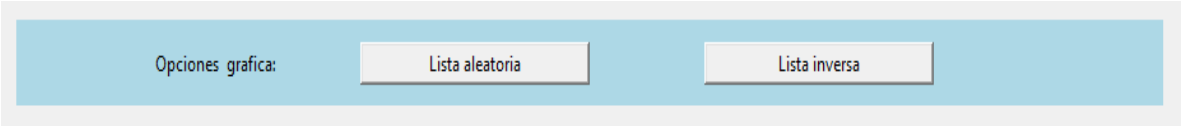
Después se escoge el ordenamiento con la **lista despegable** (entre los 10 sorts que se mencionaron al principio), para así finalmente dar click en Iniciar, de manera que, con el uso de **animaciones**, la gráfica en la que se mostraba la lista se va a ir actualizando hasta quedar ordenada de menor a mayor (formando una rampa), mostrando así el progreso del sort escogido.



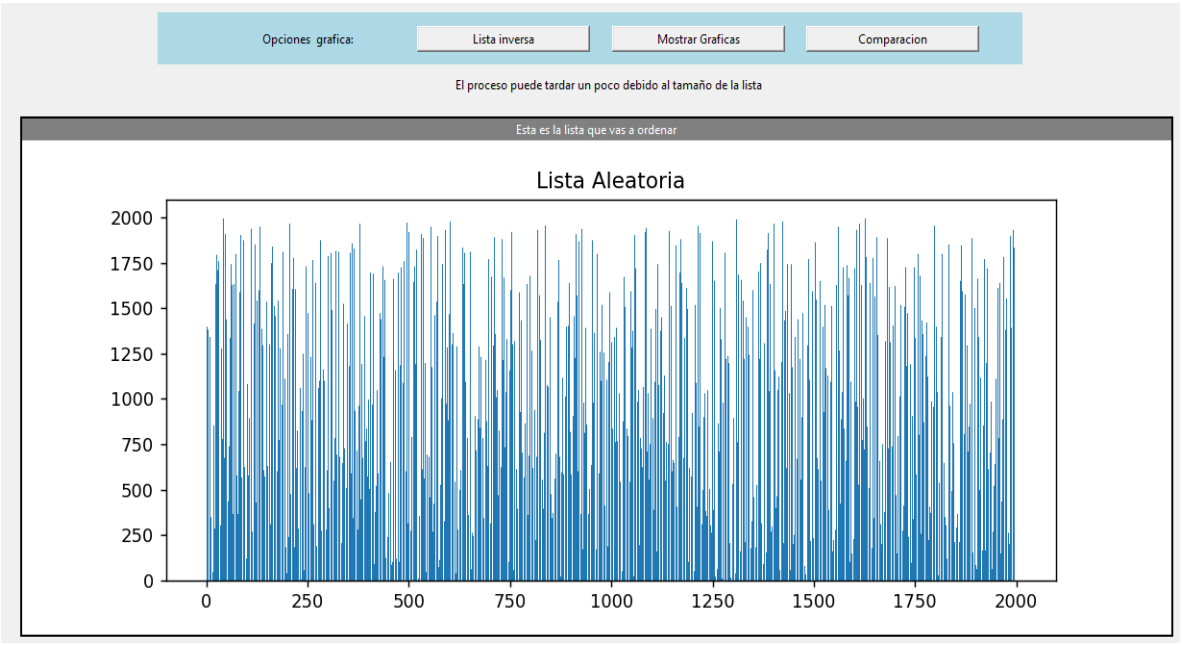
Además de mostrar el progreso de la gráfica, van a aparecer tres botones más con los que se puede **pausar**, **reanudar** o **detener** (dejar de ejecutar) el ordenamiento.

2. VENTANA 2: BIG O

En la segunda ventana, en un principio van a aparecer dos opciones, primero un botón para generar una lista aleatoria de 2000 valores (**Lista aleatoria**) y otro para generar también una lista con 2000 valores solo con el detalle de que va a estar ordenada de mayor a menor (**Lista inversa**).

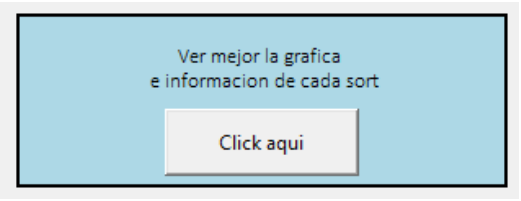


Una vez que se haya escogido una de las dos opciones se mostrará la lista en forma de gráfica debajo de los botones, además de desplegar otros dos botones. En base a dichas listas, se generará la gráfica de la complejidad y tiempo de cada algoritmo de ordenamiento expuesto al principio.

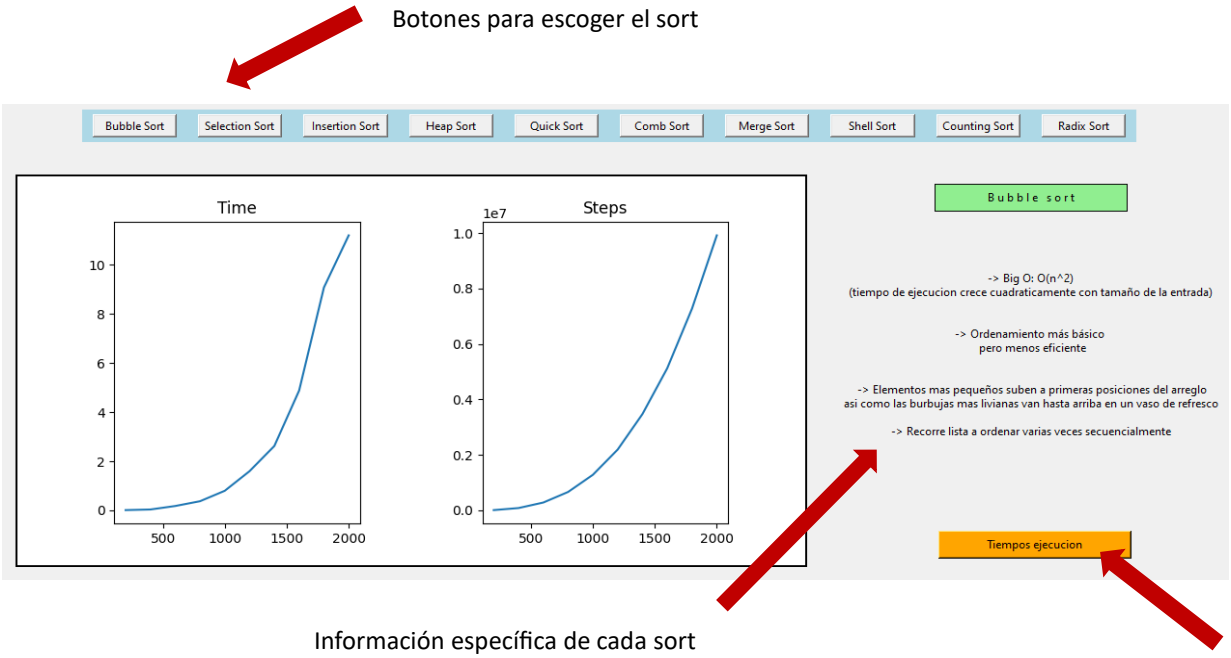


1) Con primero (**Mostrar gráficas**), al darle click, se mostrarán en la ventana las gráficas Big O de cada sort y el tiempo que tardan en ejecutarse por separado, pudiendo así analizar y comparar el comportamiento de cada algoritmo.

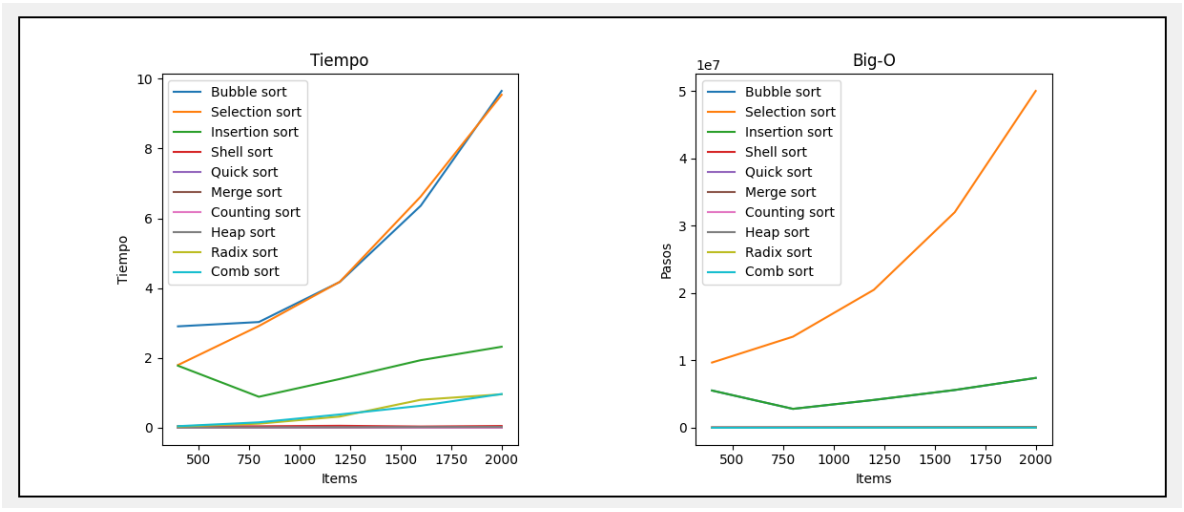
   Asimismo, mostrará un botón (**Click aquí**), el cual al presionarlo reiniciara la ventana dando la posibilidad al usuario de poder ver cada gráfica (de cada ordenamiento) individualmente y con un mejor tamaño.



   Además de mostrar a un lado información relacionada con su Big O y un último botón (**Tiempos de ejecución**) que al darle click permite ver cuánto tiempo tarda el algoritmo en trabajar con “x” cantidad de elementos.



2) Al presionar el segundo botón (**Comparación**), muestra en la ventana una gráfica en la cual es posible mostrar al mismo tiempo todas las Big O de cada sort, y el tiempo que tarda en ejecutarse cada uno, siendo así más fácil lograr hacer una comparación entre cada algoritmo de ordenamiento.



ESPECIFICACIONES

- Se recomienda ejecutar el proyecto en el programa (IDE) Visual Studio Code, es posible que en otros IDE como Spyder no funcione, mas que nada por problemas con la función de matplotlib de FuncAnimation.
- Una vez que al principio se haya escogido una de las ventanas, ya no se podrá cambiar a la otra; en todo caso tendría que cerrar la GUI (el código) y volver a ejecutarla ahora escogiendo la otra ventana.
- La animación de algunos algoritmos de ordenamiento (como Merge sort) puede que tarden en ejecutarse, pero esto es más por la naturaleza de los algoritmos, debido a que ejecutan varias acciones antes de empezar a acomodar los elementos de la lista.

- Al crear listas de 600 a 800 elementos (en la ventana 1) aparecen espacios en blanco en la gráfica a pesar de que la lista no los tenga.
- En la ventana 1 no dar click al botón Iniciar si aún no se ha escogido el ordenamiento que se va a utilizar.
- En la ventana 1, si el tamaño de la lista es muy pequeño y la velocidad muy rápida puede que no se tenga el suficiente tiempo de usar los botones para reanudar, pausar o detener el proceso.
- Para el funcionamiento de las animaciones, se reciben varias gráficas en las que cada una representa un cambio en la lista a ordenar, de manera que al final con un ciclo for se recorren todas continuamente logrando así mostrar el progreso (paso a paso) de cada sort.
- En las gráficas de Big O, puede que en ocasiones marque que el código llega al limite de recursividada (por lo que no se ejecuta como debería), con solo volver a correr el código debería solucionarse el problema. Pero en caso de que no lo haga, tal vez se deba a que la memoria de su laptop no sea lo suficientemente grande para correr el código.
- En la gráfica donde se comparan todas las BIG O al mismo tiempo, algunas pueden parecer lineales. La razón de esto son la gran diferencia entre los *steps* de cada algoritmo de ordenamiento (podría decirse que son escalas), por ejemplo, Bubble realiza unos 100000000 mientras que otros algoritmos solo unos 7000, por ello no se ven tan grandes.

LO QUE HIZO CADA INTEGRANTE:

- Animación del progreso de cada sort: **Sebastián Díaz**
- Generalizar y adaptar los algoritmos de ordenamiento: **Sebastián Díaz y Aarón Herrera**
- Gráficas Big O lista aleatoria: **Sebastián Díaz**
- Gráficas Big O lista inversa: **Aarón Herrera**
- Diseño de la GUI (orden de widgets, colores, cursor, etiquetas): **Aarón Herrera**
- Botones para pausar, reanudar y detener el ordenamiento: **Sebastián Díaz**
- Barras deslizadoras para rango de lista y velocidad: **Sebastián Díaz**
- Botones para escoger la ventana: **Sebastián Díaz**
- Gráfica para comparar las Big O al mismo tiempo: **Aarón Herrera**
- Visualización individual de cada gráfica Big O: **Sebastián Díaz**
- Información de la Big O de cada sort: **Aarón Herrera**
- Messagebox con el tiempo de cada sort: **Aarón Herrera**
- Botones para avanzar en la ventana de las Big O: **Sebastián Díaz**
- Investigación nuevo ordenamiento: **Aarón Herrera**