



CUCEI

CENTRO UNIVERSITARIO DE
CIENCIAS EXACTAS E INGENIERÍAS

UNIVERSIDAD DE GUADALAJARA
CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Análisis de algoritmos

Mtro. Jorge Ernesto Lopez Arce Delgado

Act. Visualizador de métodos de Ordenamiento

Realizado por:

Jennifer Patricia Valencia Ignacio

Juan Pablo Solis Regin

Introducción

Los algoritmos de ordenamiento son muy importantes en la programación, ya que se puede organizar datos para facilitar búsquedas y datos. Por medio de esta práctica, buscamos conocer las funciones que realizan los distintos métodos de ordenamiento como el Selection sort, Bubble sort, Mergesort y Quicksort. En temas de practicidad y mejor apartado visual, se busca también implementar una GUI (usando python y tkinter) donde se integren todos los métodos de ordenamiento ya mencionados

Objetivo:

Por medio de esta actividad, lo que buscamos es analizar cómo se comportan los distintos métodos de ordenamiento ingresando el número de barras con el que queremos trabajar, para después agregar el método de ordenamiento y la velocidad con la que se ejecuta, además de contar con otros botones para mezclar y limpiar nuestro ordenamiento de búsqueda seleccionado.

Desarrollo

Creamos un canvas donde cada barra representa un dato de la lista. Lo principal fue un entry para poner el número de barras, un combobox para elegir el algoritmo, un scale para cambiar la velocidad de la animación y cuatro botones que generan, ordenan, mezclan y limpian las barras.

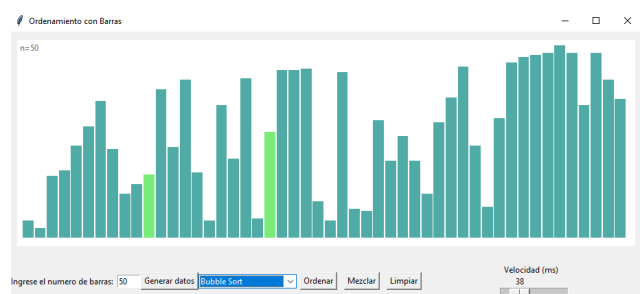
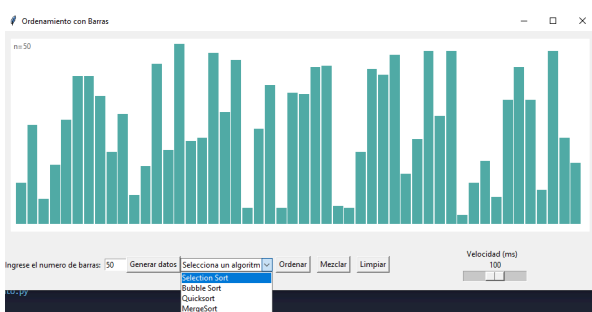
La función de dibujar_barras() dibuja todas las barras y pinta las barras que se están comparando/intercambiando.

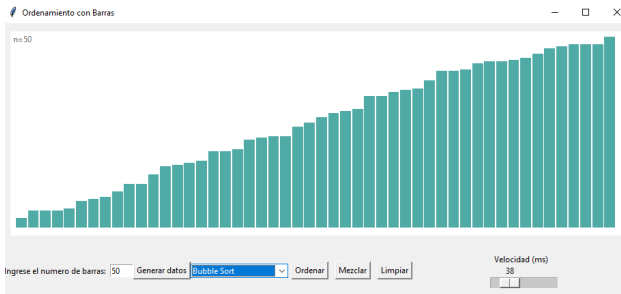
Cada algoritmo (selection_sort_steps, bubble_sort_steps, quicksort_steps, merge_sort_steps) usa *yield*, lo que pausa el algoritmo y así se puede ver la animación paso a paso.

- En selection sort busca el valor más pequeño en cada pasada y se coloca en su lugar.
- En bubble sort se compara y cambia elementos vacíos, este es el más lento.
- Quicksort divide la lista con un pivote y ordena las partes.
- Mergesort se divide en mitades y luego va juntando sublistas ordenadas.

Además, agregamos un diccionario de colores para que sea fácil reconocer cual algoritmo se está usando.

Capturas de código:





Cantidad de datos (barras)	Selection	Bubble	QuickSort	Mergesort
10	6 seg	6 seg	4 seg	4 seg
20	20 seg	30 seg	12 seg	11 seg
30	50 seg	1 min y 10 seg	22 seg	18 seg

Conclusiones:

Con esta actividad entendimos cómo trabajan los algoritmos y cual es el más rápido y cuál es el más lento. El uso de yield ayuda mucho porque se puede ver paso a paso cómo se ordenan los datos.

Vimos que el selection sort y el bubble sort funcionan más en listas chiquitas, mientras que el quicksort y el mergesort son mejores para listas más grandes.

Fuentes de consulta:

- *Algoritmo Quicksort y como implementarlo en Python* | Andros Fenollosa. (2023). Andros.dev.
<https://andros.dev/blog/eea72544/algoritmo-quicksort-y-como-implementarlo-en-python/>
- Team, S. (2024, May). *Introducción al Algoritmo de Ordenación Merge Sort*. Swhosting.com; SW Hosting.
<https://www.swhosting.com/es/blog/introduccion-al-algoritmo-de-ordenacion-merge-sort>
- *Algoritmo MergeSort: cómo implementarlo en Python* | Alura Cursos Online. (2022, June 9). Alura.
<https://www.aluracursos.com/blog/algoritmo-mergesort-como-implementarlo-en-python>