INSTITUTO TECNOLÓGICO DE COSTA RICA
Escuela de Computación
Bachillerato en Ingeniería en Computación
IC-1803 Taller de Programación
Prof. Mauricio Avilés

Tarea Corta 6 – Algoritmos de Ordenamiento

Para esta tarea debe entregarse un archivo ZIP con:

- PDF del análisis de resultados
- PY de las pruebas realizadas

Debe entregarse el archivo ZIP en el TEC-Digital.

Problema

Esta tarea consiste en la elaboración de pruebas para medir el desempeño de diferentes algoritmos de ordenamiento básicos. Los algoritmos a evaluar serán los siguientes:

1. Bubble Sort
2. Insert Sort
3. Selection Sort
4. Merge Sort
5. Quick Sort

Generación de datos

Por cada algoritmo de ordenamiento, deben realizarse diferentes corridas con el propósito de medir el tiempo exacto de ejecución de cada uno. Las pruebas van a ejecutarse con una cantidad de elementos (n) generados de forma aleatoria. Por cada valor de n deben hacerse al menos 10 pruebas. Debe calcularse el tiempo promedio que tardó el algoritmo en ordenar los n elementos en esas 10 pruebas.

La idea es generar datos del promedio de las ejecuciones lo más detalladamente posible. Se inician las pruebas a partir de listas de 100 elementos, y luego se va aumentando la cantidad de elementos en las listas para realizar las demás pruebas. Entre menor sea el incremento, más detallados serán los gráficos resultantes. Se recomienda un incremento de 500 elementos, pero puede utilizar alguno menor si así lo desea.

```
bubbleSort: 10 pruebas de 100 elementos. Promedio: 0.00039887428283691406.
bubbleSort: 10 pruebas de 600 elementos. Promedio: 0.01954209804534912.
bubbleSort: 10 pruebas de 1100 elementos. Promedio: 0.07290809154510498.
bubbleSort: 10 pruebas de 1600 elementos. Promedio: 0.1640460252761841.
bubbleSort: 10 pruebas de 2100 elementos. Promedio: 0.279989218711853.
bubbleSort: 10 pruebas de 2600 elementos. Promedio: 0.43105568885803225.
...
```

Ejemplo de programa que está realizando pruebas del Bubble Sort. Por cada cantidad de elementos se realizan 10 pruebas, se mide el tiempo y se promedian. Independientemente de las impresiones que el programa de prueba haga en pantalla, deben guardarse los resultados en dos listas, una lista con la cantidad de elementos utilizados en el grupo de 10 pruebas y la otra con los promedios obtenidos.

```
bubbleSortN = [100, 600, 1100, 1600, 2100, 2600]
bubbleSortT = [0.00039887428283691406, 0.01954209804534912, 0.07290809154510498,
0.1640460252761841, 0.279989218711853, 0.43105568885803225]
```

En este ejemplo se muestran los datos obtenidos en las primeras seis pruebas, pero para el trabajo debe tratar de llevar la cantidad de elementos a un número bastante alto. Usted debe decidir hasta qué punto hacer las pruebas con cada algoritmo. Por ejemplo, el algoritmo Bubble Sort con 20 000 elementos tarda mucho tiempo, por lo que las pruebas pueden llevarse hasta un número menor a este. Por otro lado, un algoritmo más eficiente como el Quick Sort es capaz de ordenar hasta 1 000 000 de elementos en pocos segundos, por lo que es válido hacer mayor cantidad de pruebas.

El objetivo de esta primera parte es generar las dos listas por cada algoritmo con la mayor cantidad de datos posible. Automatice las pruebas de la mejor manera para evitar el trabajo manual repetitivo.

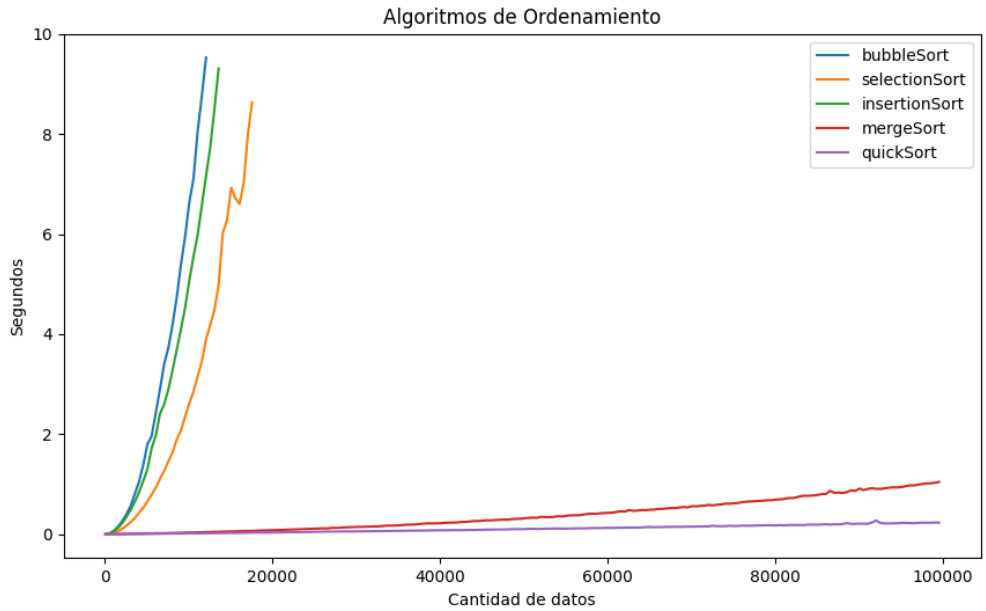
Gráfica de los datos

A partir de los datos generados con las pruebas vamos a crear un gráfico de línea que muestre el comportamiento de los diferentes algoritmos según la cantidad de datos a ordenar. Para esto utilizaremos la biblioteca de Python Matplotlib.

Para instalar la biblioteca debe ejecutarse el siguiente comando en la consola del sistema operativo.

```
pip install matplotlib
```

Esto instalará la biblioteca y todas las dependencias necesarias para que funcione. El objetivo es generar una gráfica similar a la mostrada a continuación:



La gráfica de ejemplo muestra una serie de datos por cada algoritmo probado. Se utiliza la subrutina plot(X, Y, label = nombre) por cada serie de datos, donde X es la lista con los valores del eje X, Y es la lista con los valores del eje Y, y nombre es el nombre del algoritmo utilizado. Las listas X y Y corresponderían a las mostradas en el ejemplo de generación de datos bubbleSortN y bubbleSortT. Para asignar etiquetas al X y Y se utilizan las subrutinas xlabel y ylabel. Para asignar un título al gráfico se usa la subrutina title. Para mostrar la leyenda con los colores de las series se utiliza la subrutina legend. Finalmente para mostrar el gráfico se utiliza la subrutina show. Puede consultar el siguiente enlace para ver ejemplos de utilización:
https://www.geeksforgeeks.org/graph-plotting-in-python-set-1/

Documento

Además del código que genera los datos y la gráfica debe escribirse un documento con el análisis de los resultados obtenidos. Este documento debe incluir un apartado por cada algoritmo utilizado. Primero incluya una breve descripción sobre cómo trabaja cada algoritmo, es decir, la estrategia que utiliza para ordenar elementos. Refiérase a las características de las implementaciones utilizadas (estable o no, in place o no, basado en comparaciones, basado en intercambios, iterativo o recursivo). Incluya una tabla donde se muestren los promedios de las pruebas realizadas con el algoritmo. A continuación, se muestra una tabla de ejemplo con los resultados obtenidos con Bubble Sort.

Cantidad de elementos	Tiempo promedio de 10 pruebas
100	0.00039887428283691406
600	0.01954209804534912
1100	0.07290809154510498
1600	0.1640460252761841
2100	0.279989218711853
2600	0.43105568885803225
…	…

Seguidamente, presente el gráfico de línea que ilustra el rendimiento de los diferentes algoritmos según su tiempo de ejecución y cantidad de elementos. Escriba un análisis donde se comparen los resultados obtenidos entre los diferentes algoritmos. Establezca diferentes hipótesis sobre porqué se dieron los resultados de los experimentos.