Práctica 01

DOCENTE	CARRERA	CURSO
Vicente Machaca Arceda	Maestría en Ciencia de la Computación	Estructura de Datos y Algoritmos

PRÁCTICA	TEMA	DURACIÓN
01	–	3 horas

1. Datos de los estudiantes

- Grupo: 3
- Integrantes:
  - Lizarraga Mendoza David Jesus
  - Saenz Mamani Alex Alberto
  - Huaman Hilari Julissa Zaida
  - Chara Condori Julio Cesar
  - Acuña Chavez Melvin

2. Ejercicios

1. Preparación de los datos. Usted debe generar varios conjuntos de datos (archivos .txt), por ejemplo va a generar números aleatorios y los va a almacenar en varios archivos. Cada archivo deberá contener respectivamente 100, 500, 1000, 2000, 3000, ... , 10000, 20000, 30000, ... ,100000 datos.

   Solución:

   Se generó 21 archivos txt utilizando el siguiente algoritmo desarrollado en python, ubicado en el siguiente link: `https://github.com/julissah/EDA-M/blob/development/GenarateFile/generatefile.py`

2. Implemente los siguientes algoritmos en C++ y Python:

   Solución:

   Se implentó los siguientes algoritmos de ordenamiento, los cuales estan ubicados en el siguiente link `https://github.com/julissah/EDA-M/tree/development`:

   Bubble sort
   Counting sort
   Heap sort

Insertion sort
Merge sort
Quick sort
Selection sort

3. Realice comparaciones del tiempo de procesamiento de cada algoritmo por cada lenguaje de programación.

   a) En la Figura 1, se muestra una comparación del Buble sort y Python. El eje x representa diferentes tamaños de vector a ordenar y el eje y, representa el tiempo de procesamiento.

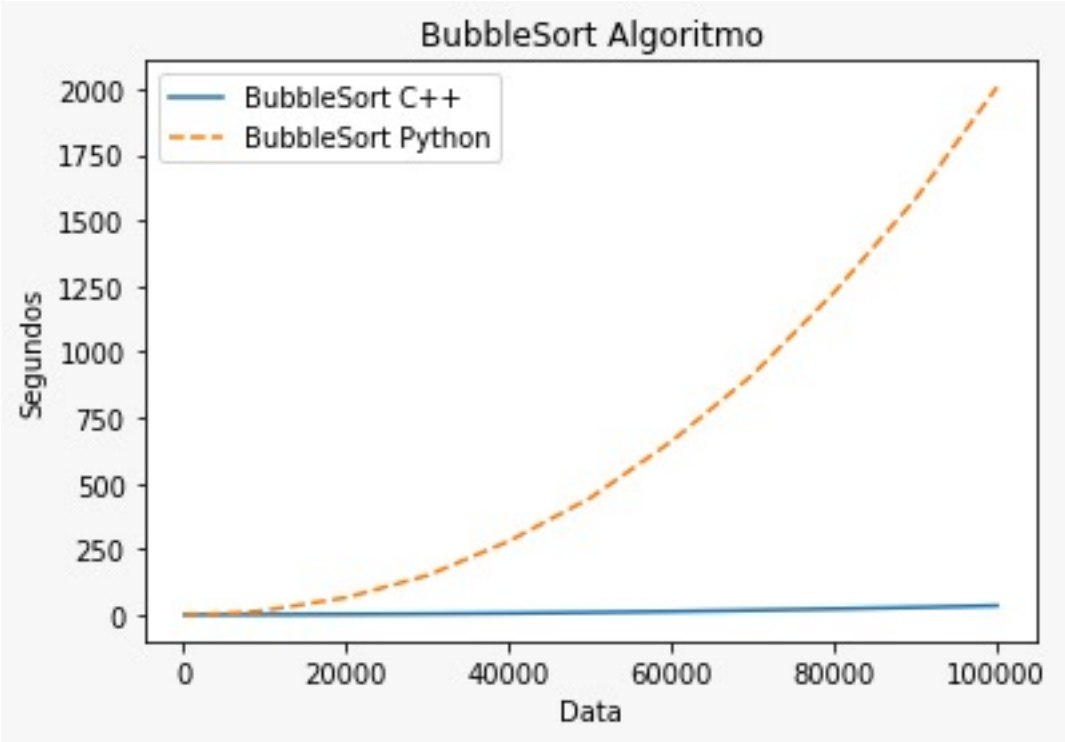

Figura 1: Comparación de Buble Sort en C++ y Python.

b) En la Figura 2, se muestra una comparación del counting sort y Python. El eje x representa diferentes tamaños de vector a ordenar y el eje y, representa el tiempo de procesamiento.

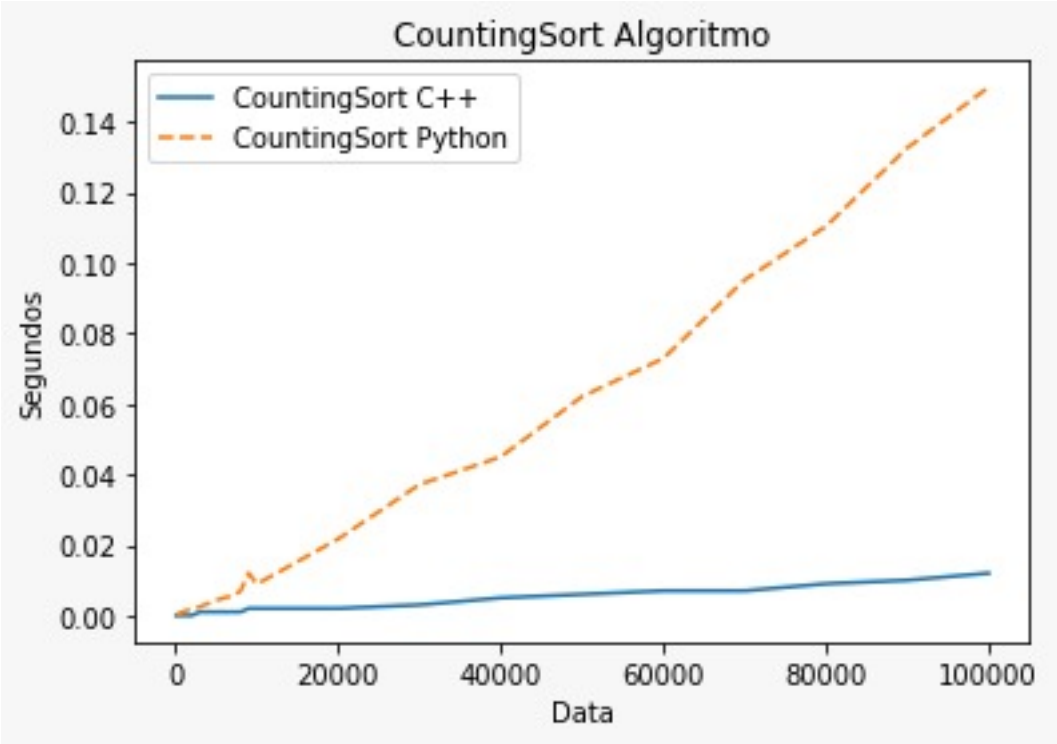

Figura 2: Comparación de Counting Sort en C++ y Python.

c) En la Figura 3, se muestra una comparación del heap sort y Python. El eje x representa diferentes tamaños de vector a ordenar y el eje y, representa el tiempo de procesamiento.

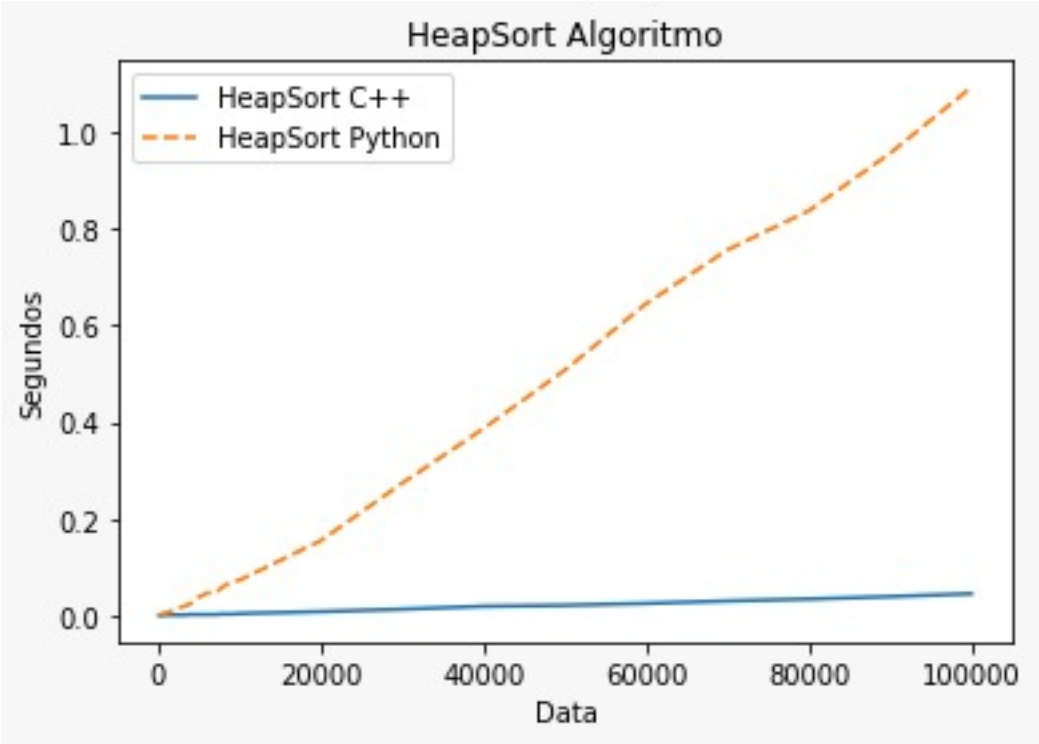

Figura 3: Comparación de Heap sort en C++ y Python.

d) En la Figura 4, se muestra una comparación del insertion sort y Python. El eje x representa diferentes tamaños de vector a ordenar y el eje y, representa el tiempo de procesamiento.

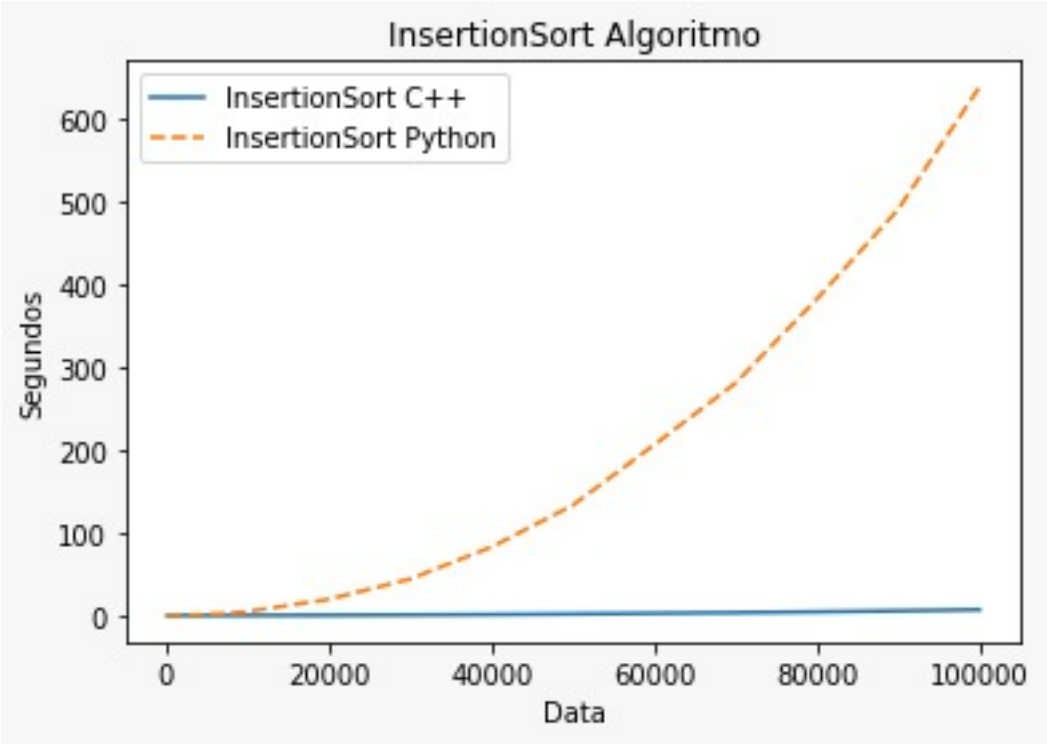

Figura 4: Comparación de Insertion sort en C++ y Python.

e) En la Figura 5, se muestra una comparación del merge sort y Python. El eje x representa diferentes tamaños de vector a ordenar y el eje y, representa el tiempo de procesamiento.

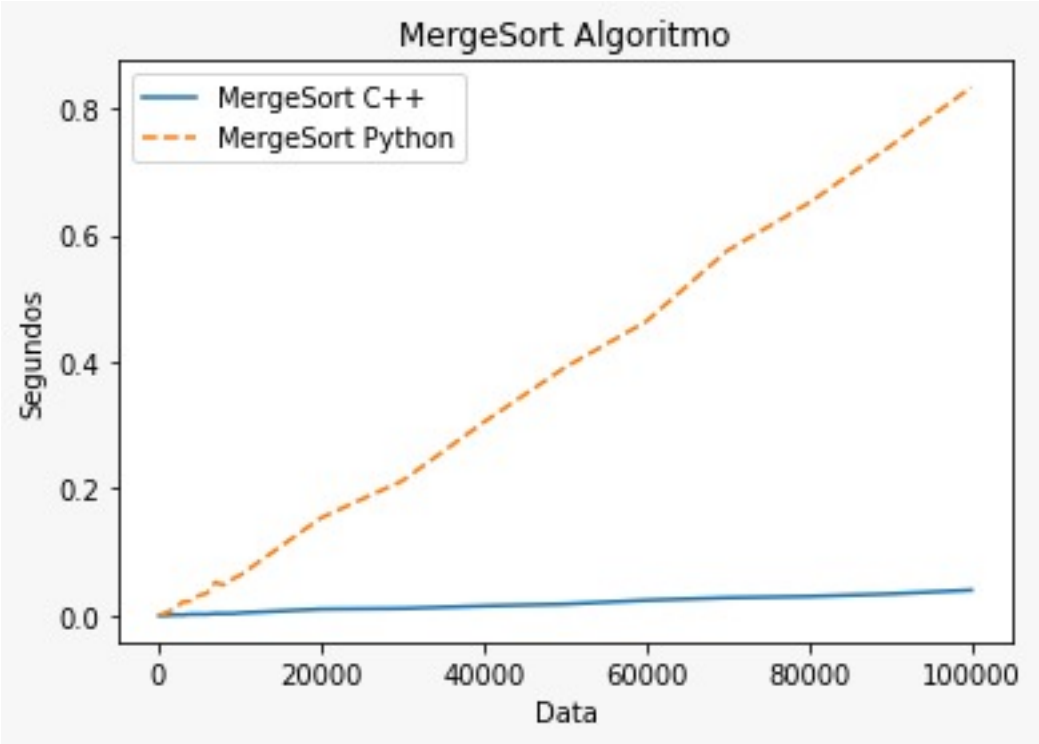

Figura 5: Comparación de Merge sort en C++ y Python.

f) En la Figura 6, se muestra una comparación del quick sort y Python. El eje x representa diferentes tamaños de vector a ordenar y el eje y, representa el tiempo de procesamiento.

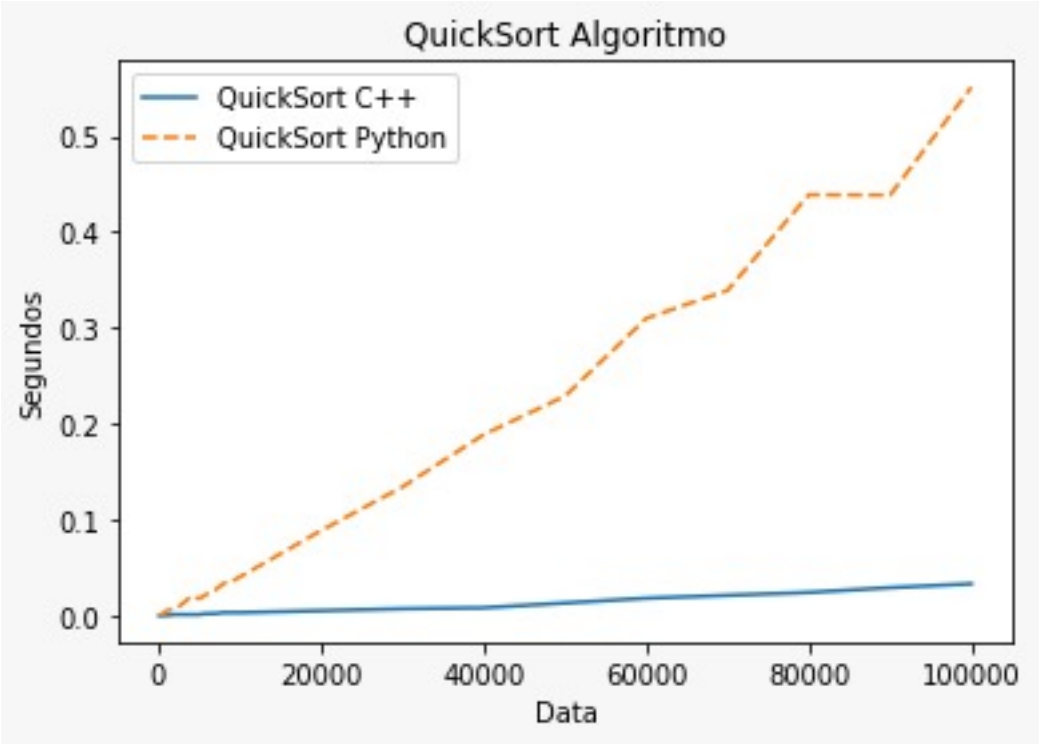

Figura 6: Comparación de Quick sort en C++ y Python.

g) Comparación del tiempo de procesamiento de todos los algoritmos implementados en C++. En la Figura 7, se muestra la comparación en C++.

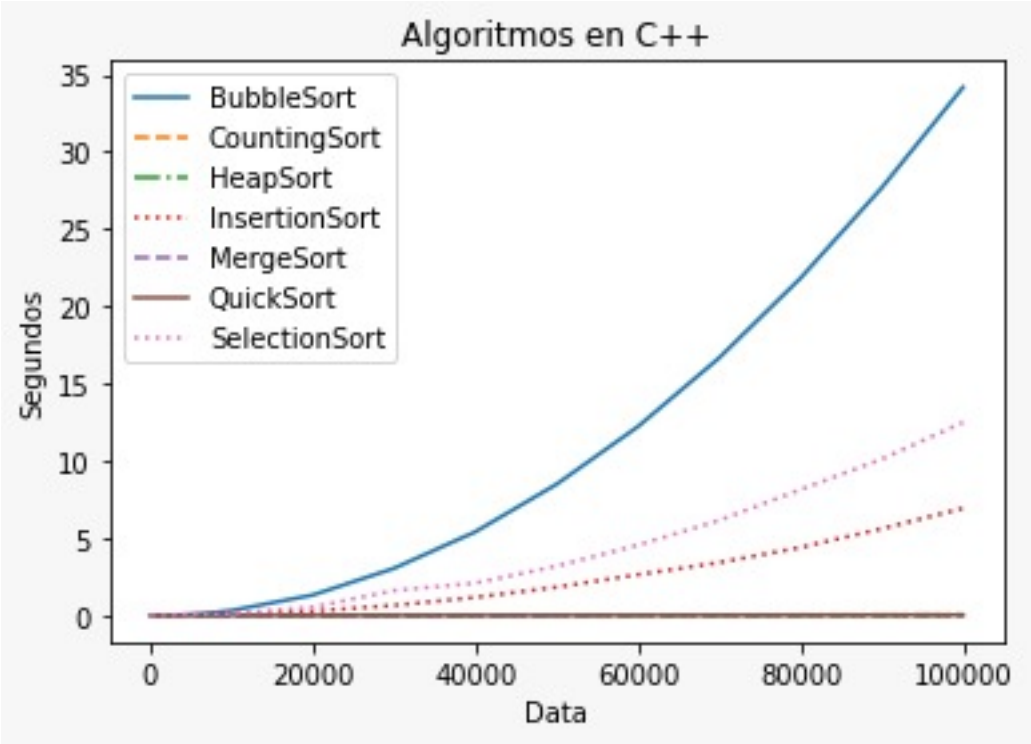

Figura 7: Comparación de los algoritmos de ordenamiento en C++.

h) Comparación del tiempo de procesamiento de todos los algoritmos implementados en Python. En la Figura 8, se muestra la comparación en Python.

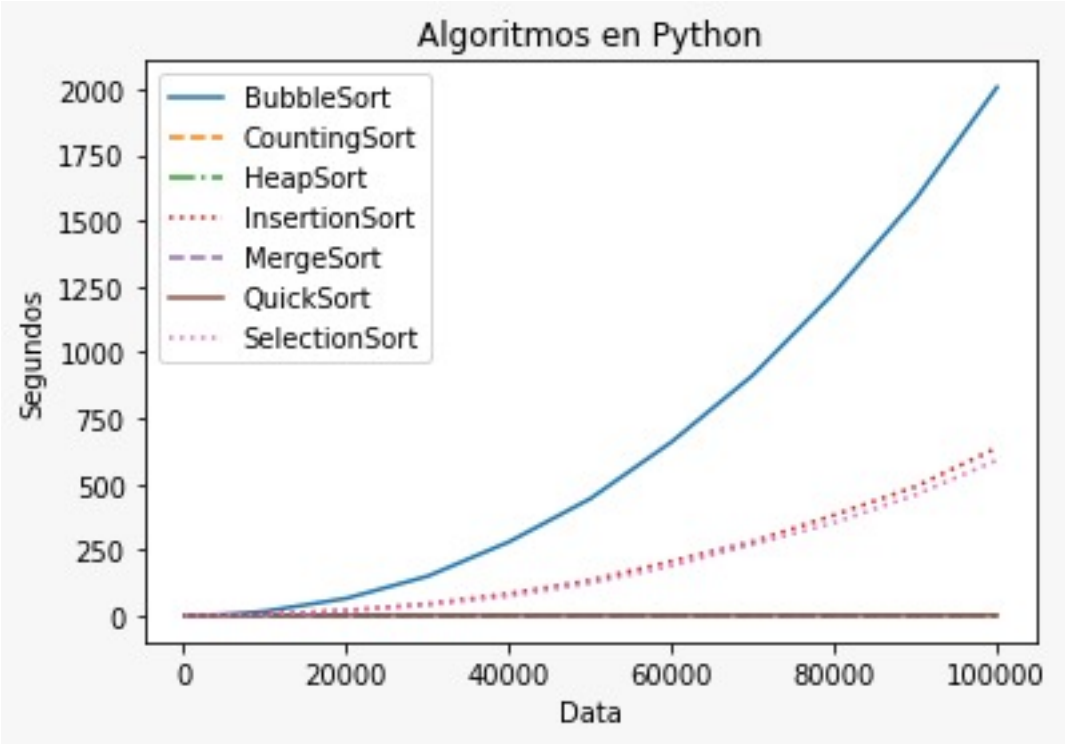

Figura 8: Comparación de los algoritmos de ordenamiento en Python.