Tarea 1
Algoritmos de ordenamiento

Datos ordenados

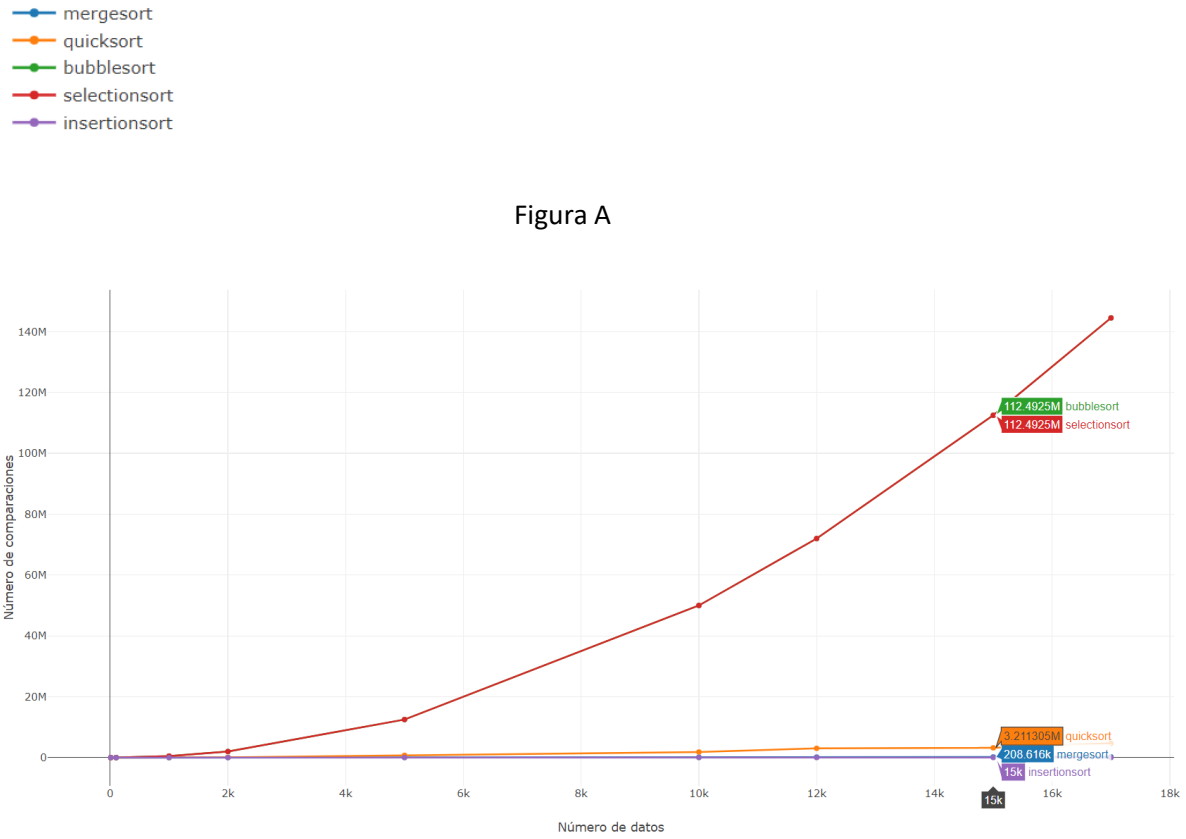


Figura A

Figura B

Juan Carlos Sigler Priego
Estructuras de Datos Avanzadas

Tarea 1
Algoritmos de ordenamiento

Datos en orden inverso

Figura A

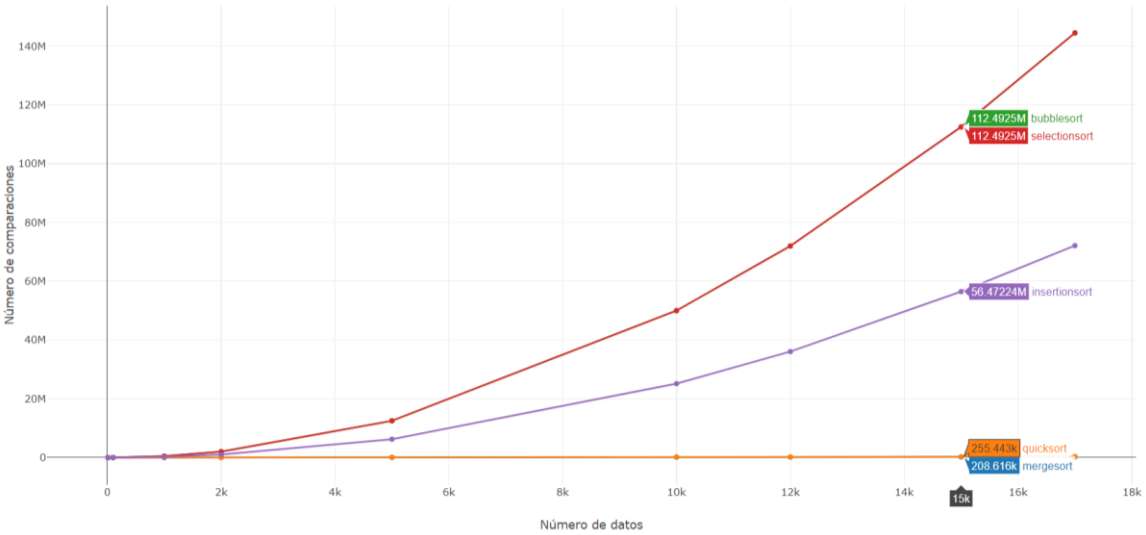
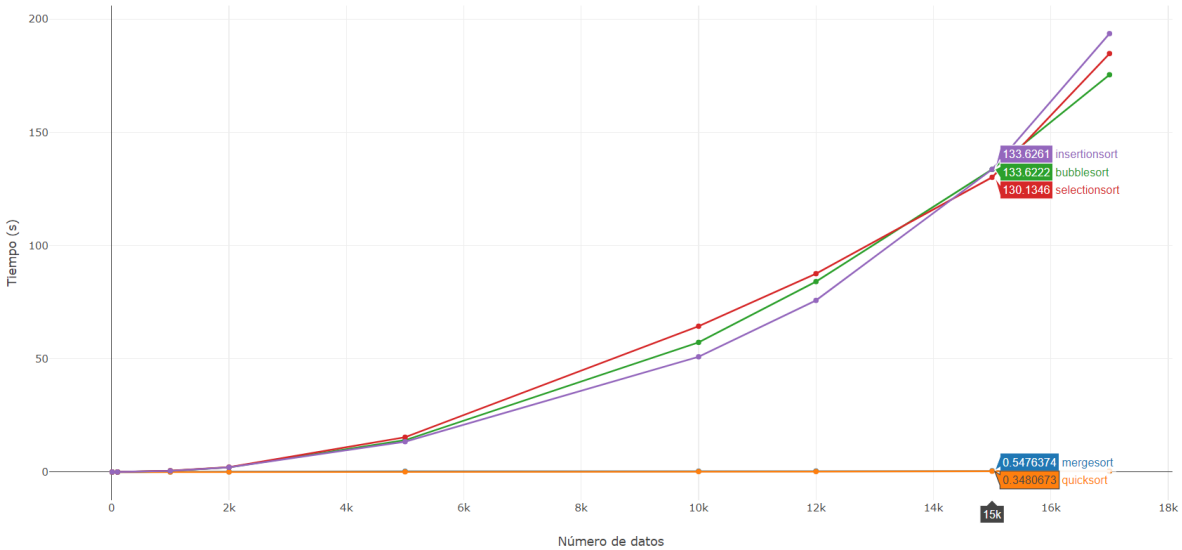


Figura B



Juan Carlos Sigler Priego
Estructuras de Datos Avanzadas

Tarea 1
Algoritmos de ordenamiento

Datos en orden aleatorio

Figura A

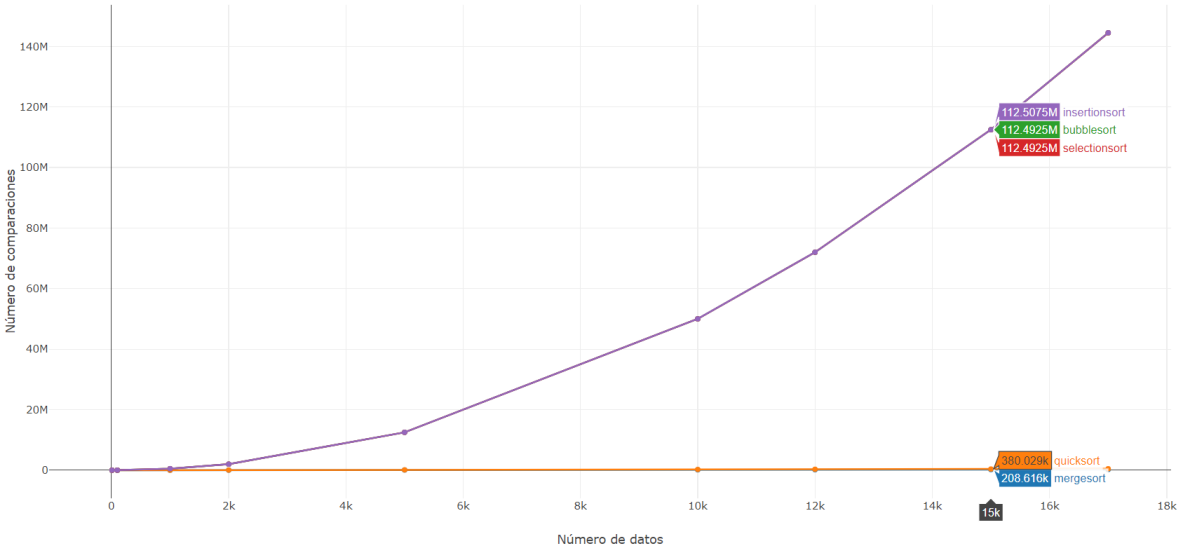
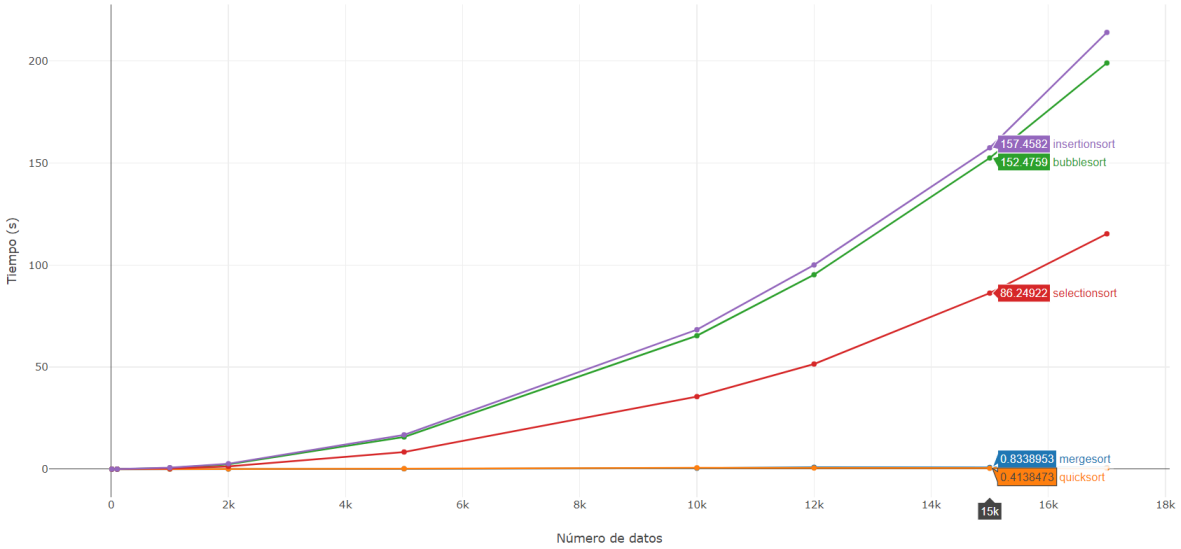


Figura B



Juan Carlos Sigler Priego
Estructuras de Datos Avanzadas

Tarea 1

Algoritmos de ordenamiento

Conclusiones

Analizando los resultados arrojados por el Bubble Sort y Selection Sort se llega a las siguientes aseveraciones. Primero, el número de comparaciones y el tiempo de ejecución no depende del orden inicial de los datos; además, se puede ver que el número de comparaciones es exactamente igual en los tres incisos. Otra observación es que en el caso particular de selection sort, es ligeramente más rápido que el bubble sort por el número de veces que intercambia elementos. Por último, se puede inferir que la complejidad de ambos algoritmos es $O(n^2)$.

Para el Insertion Sort se puede hacer las siguientes dos afirmaciones. La primera es que es el algoritmo más rápido y por lo tanto el que hace menos comparaciones si los datos están inicialmente ordenados; para este caso el algoritmo tiene complejidad lineal. La segunda es que, para los otros dos incisos, su gráfica se parece mucho al bubble sort. En conclusión, la complejidad de este algoritmo es $O(n^2)$.

En el Merge Sort se puede inferir por la gráfica que su comportamiento es muy semejante para los tres incisos, por lo que se puede asumir que es el más seguro y de los más eficientes ordenando datos, sin importar el orden inicial de estos. La complejidad de este algoritmo es O(nlgn).

Por último, el algoritmo Quick Sort resultó competir también por el primer puesto en complejidad. Obtuvo un mejor resultado que el merge sort en los dos últimos incisos. Una buena explicación del por qué pasa esto es que el merge sort consume más tiempo y memoria en su implementación, contrario al quick sort. Es importante decir que el algoritmo implementado es el quick sort que ocupa un pivote aleatorio. Por esta razón, no importa realmente la configuración inicial de los datos. Su complejidad, atestiguado por las gráficas, es O(nlgn).

Notas

- Python es excesivamente lento comparado con lenguajes de menor nivel, como C++.
- La complejidad de los algoritmos de ordenamiento depende en su mayoría por el número de comparaciones que este realiza.
- Los algoritmos de la misma complejidad solo se diferencian por una constante.

Juan Carlos Sigler Priego

Estructuras de Datos Avanzadas