César Darío Sotelo Aportela

Algoritmos de Ordenamiento

20/02/2023

Profesor: Fernando Esponda

Introducción:

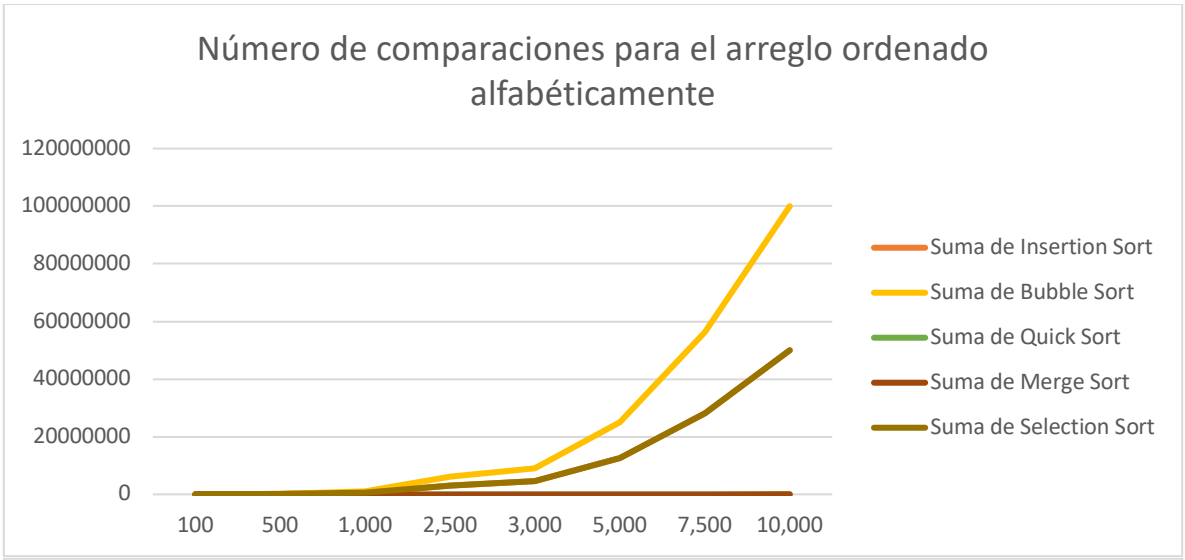
El objetivo de este proyecto es comparar los algoritmos de ordenamiento, para eso registramos el tiempo tardado por cada uno de los algoritmos junto con el número de comparaciones que hicieron cada uno de los algoritmos. Para tener un análisis más acertado, tomamos un arreglo de películas ordenadas alfabéticamente y este arreglo fue ordenado por su ID de película.

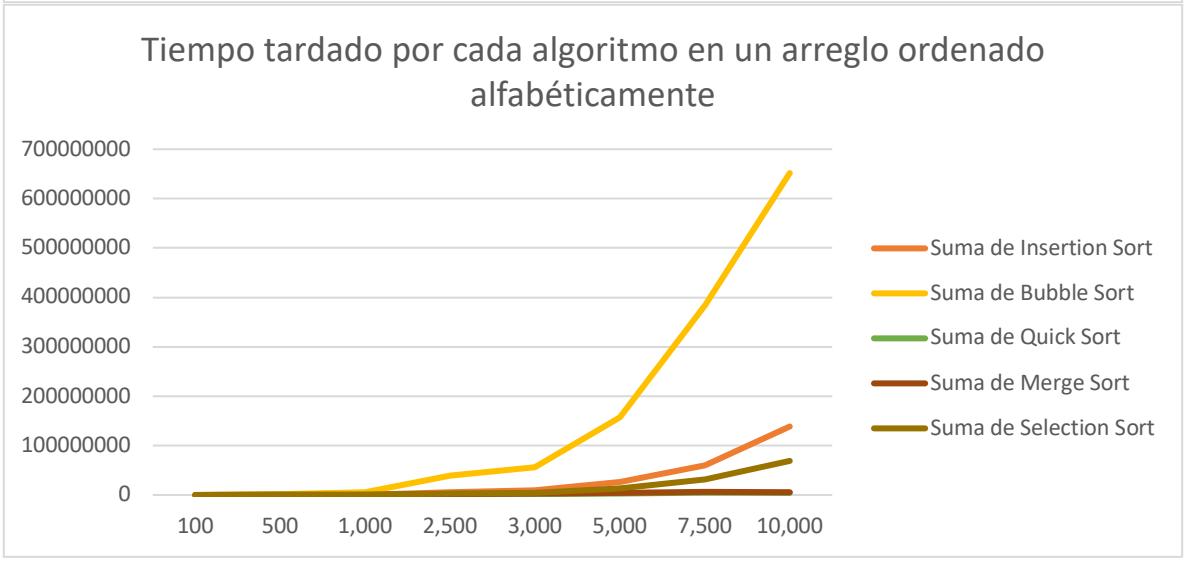
Cada película tiene tres campos: título, ID de la película y año en la que salió. A mí me pareció mejor ordenar las películas por su ID, porque por título sería redundante y por el año habría varias repeticiones, es decir, no ordenaría las películas que salieron el mismo año y el objetivo es comparar exhaustivamente el rendimiento de cada uno de los algoritmos.

Es importante notar que todos los métodos de ordenamiento fueron hechos de forma recursivamente.

A continuación presento los resultados.

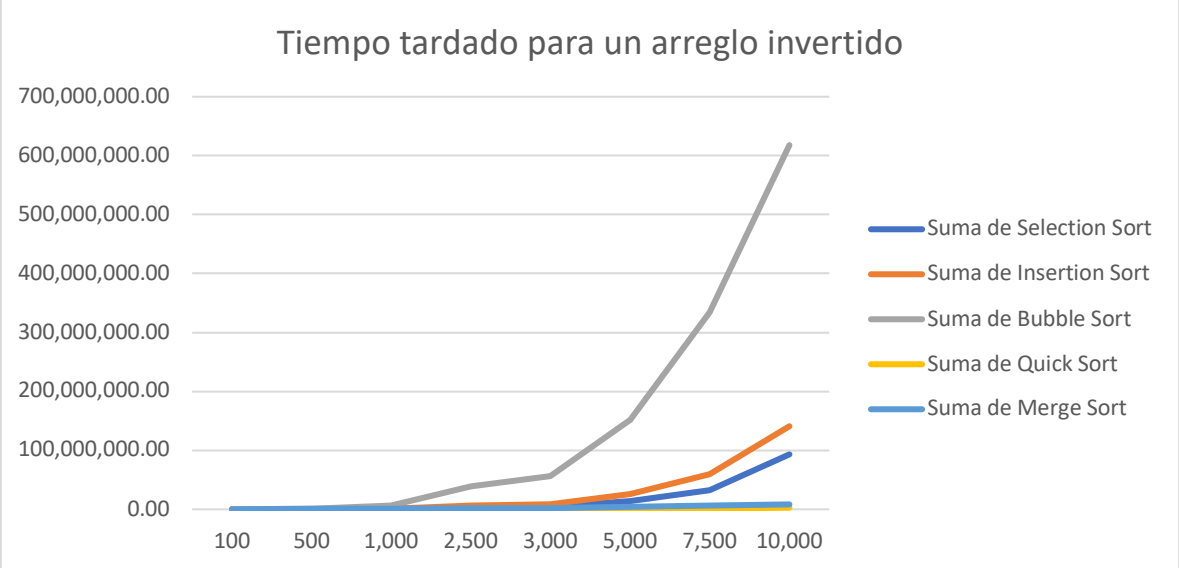
Análisis para el arreglo ordenado de forma alfabéticamente:





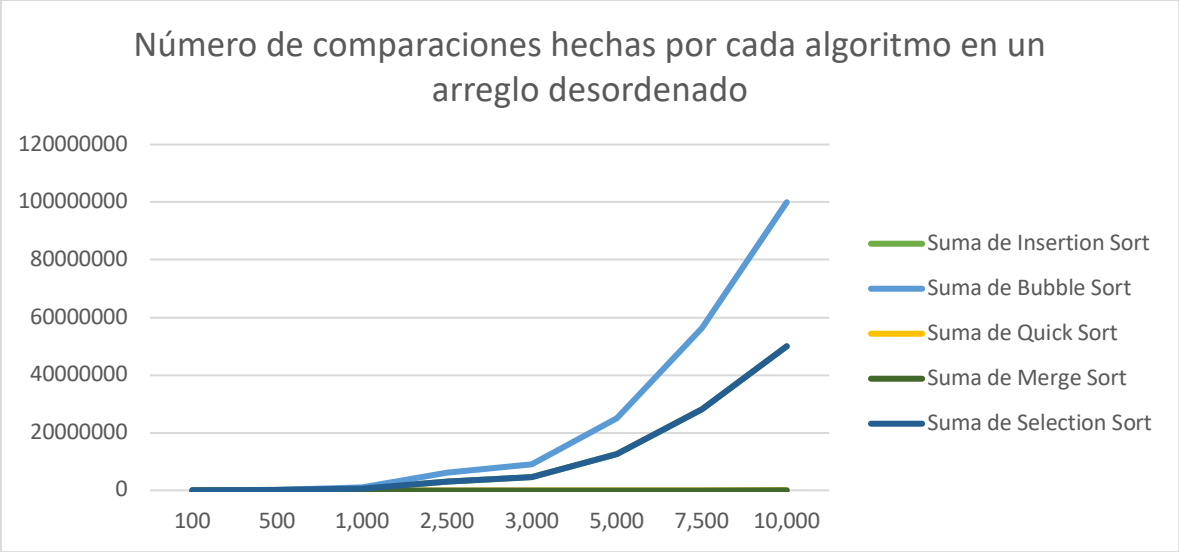
Claramente podemos ver que el Bubble Sort es el algoritmo más tardado, seguido por el insertion sort, selection sort y al final el quick sort y el merge sort pudieron alcanzar una velocidad mucho más grande que los otros algoritmos. No es de sorprender entonces que el bubble sort sea el que más comparaciones hace, siendo este el algoritmo más ineficiente de todos.

Análisis para el arreglo ordenado de forma alfabéticamente en reversa del anterior:



Otra vez podemos ver que el bubble sort es el algoritmo más ineficiente. El selection sort a pesar de que hace muchas comparaciones, no se lleva el segundo lugar en ser el algoritmo más tardado. Los algoritmos de quick sort y merge sort se llevan los premios en ser los más eficientes otra vez.

Análisis para el arreglo desordenado:



Para este análisis se hizo un promedio de varias respuestas de el ordenamiento de los algoritmos y obtuvimos estas gráficas. Para poder analizar este arreglo, se leyó el archivo y después se organizó de forma aleatoria con la biblioteca de Collections en java. En este resultado podemos ver otra vez que el bubble sort es el algoritmo más ineficiente y el insertion sort sigue su lugar.

Conclusión:

Algo que se vio bastante en las respuestas es que el bubble sort es el algoritmo más ineficiente tanto en comparaciones como en tiempo tardado para ordenar el arreglo, pero es de los más fáciles de entender. Otra conclusión de este proyecto es que los algoritmos más eficientes fueron el quick sort y el merge sort, los cuales sí reflejan el resultado teórico aprendido en clase.