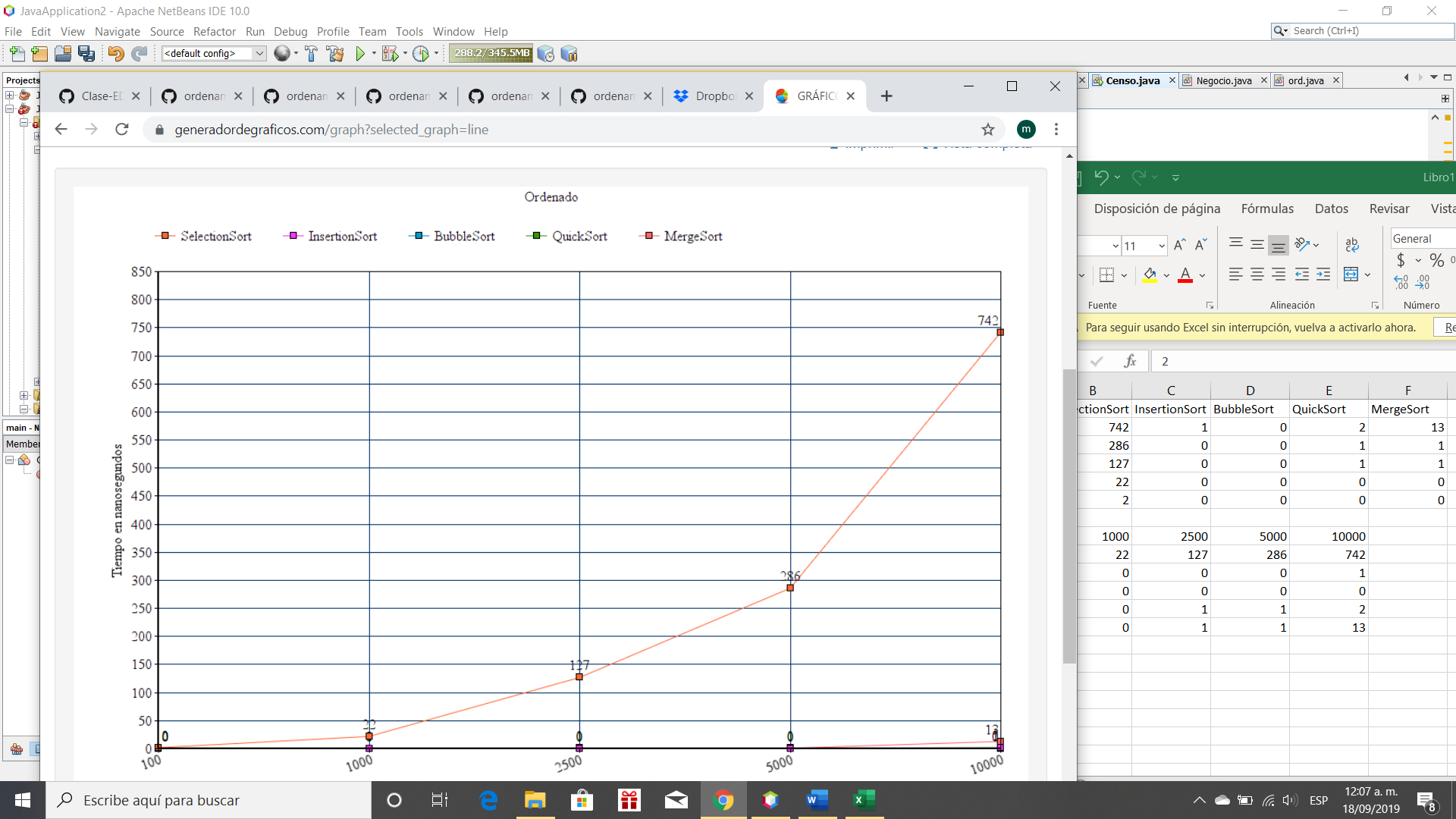
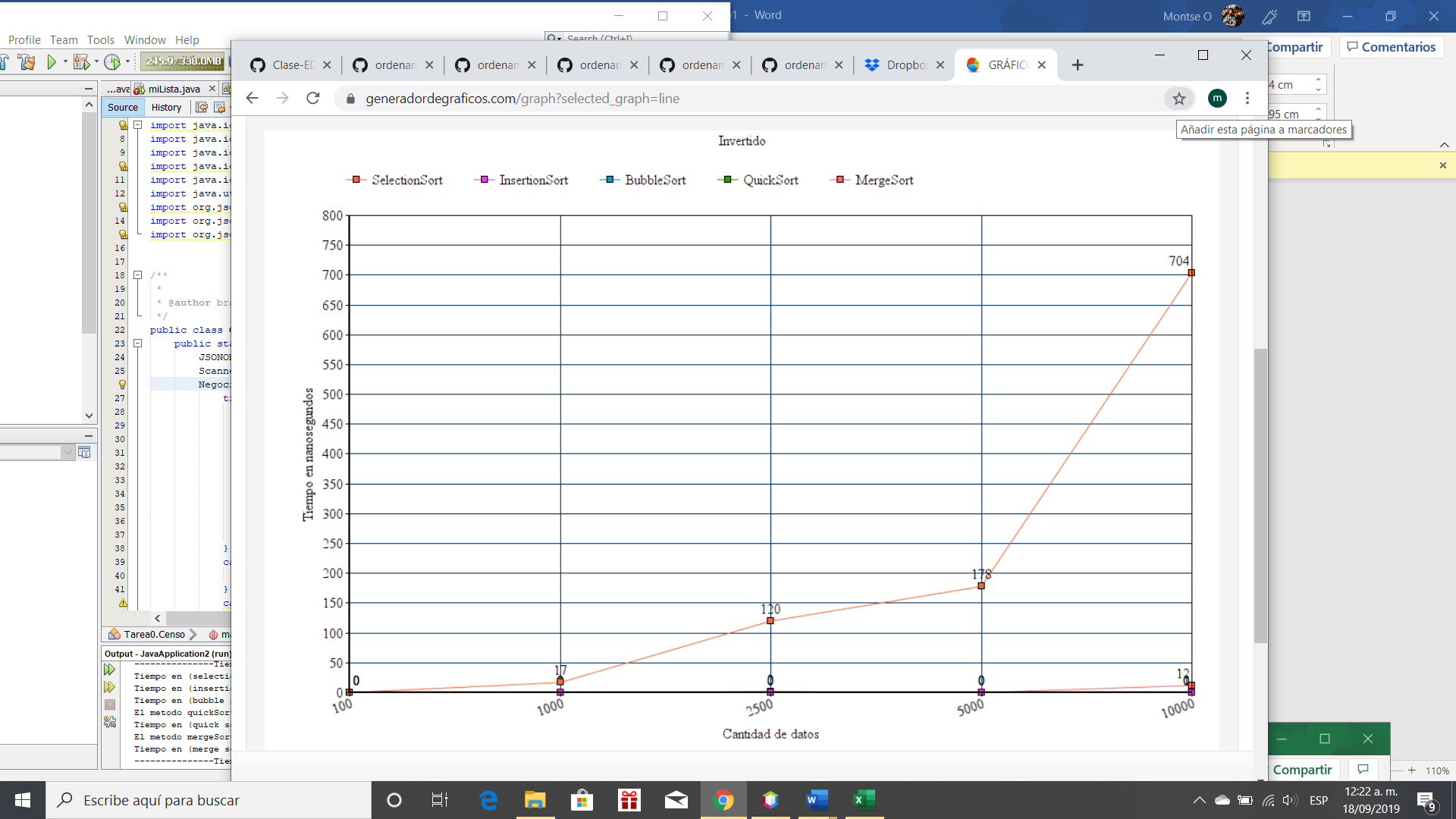
**Comparación entre Algortimos de Ordenamiento**

Montserrat Olivares Ruiz 179905

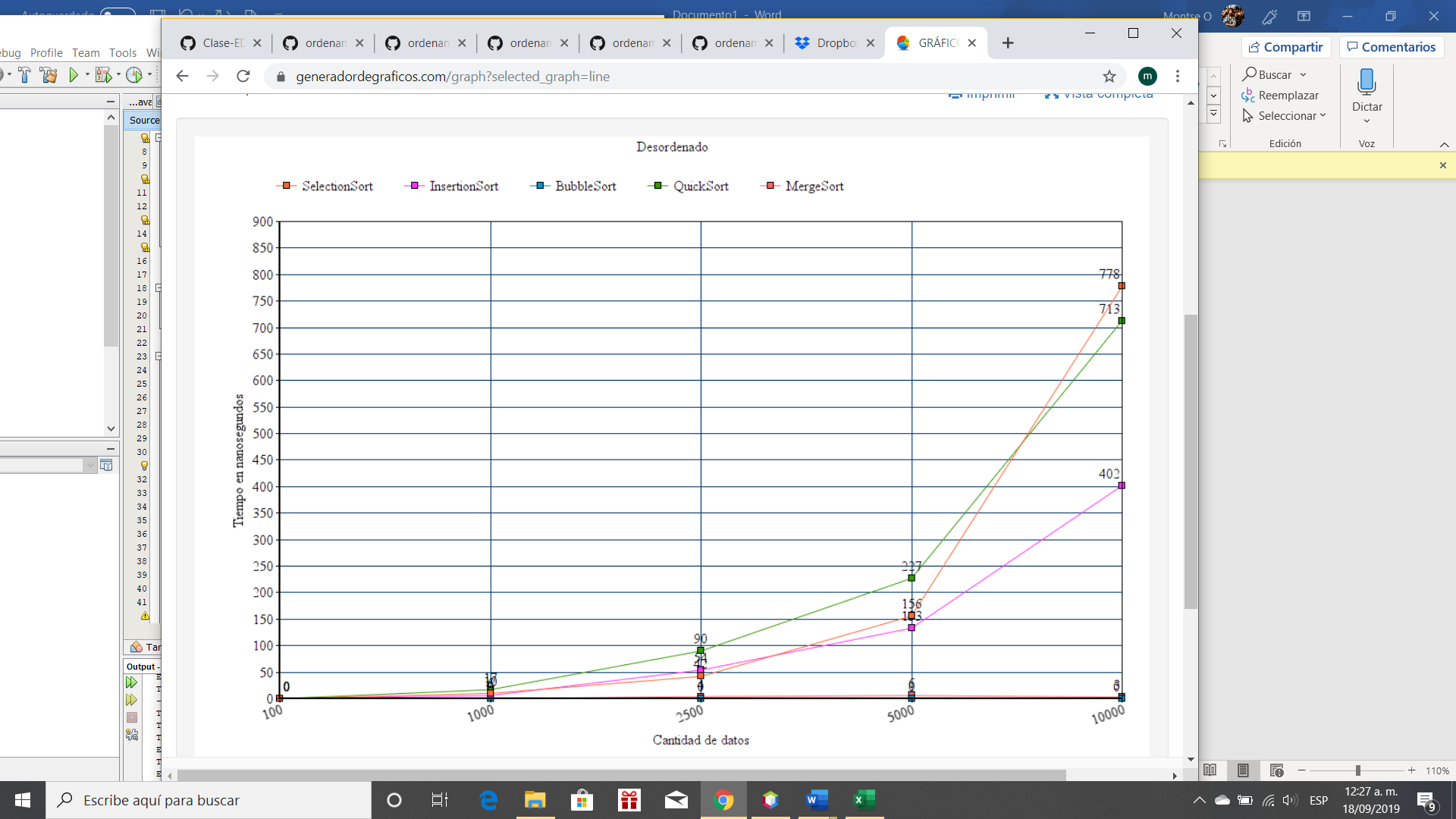
En este trabajo se presentan los resultados al probar los algoritmos de ordenamiento vistos en clase: selection sort, insertion sort, bubble sort, quick sort y merge sort. Con cada uno de los métodos se probaron con 100 1000, 2500, 5000 y 10000 elementos para comparar su rendimiento dependiendo del tamaño de la entrada.



En esta primera gráfica, se comparan los 5 algoritmos cuando tienen un arreglo ordenado. El que toma más tiempo es el selection sort, que incrementa conforme van incrementando la cantidad de elementos. Los que menos tiempo toman son el merge Sort y sobretodo bubble sort, que es el más rápido.

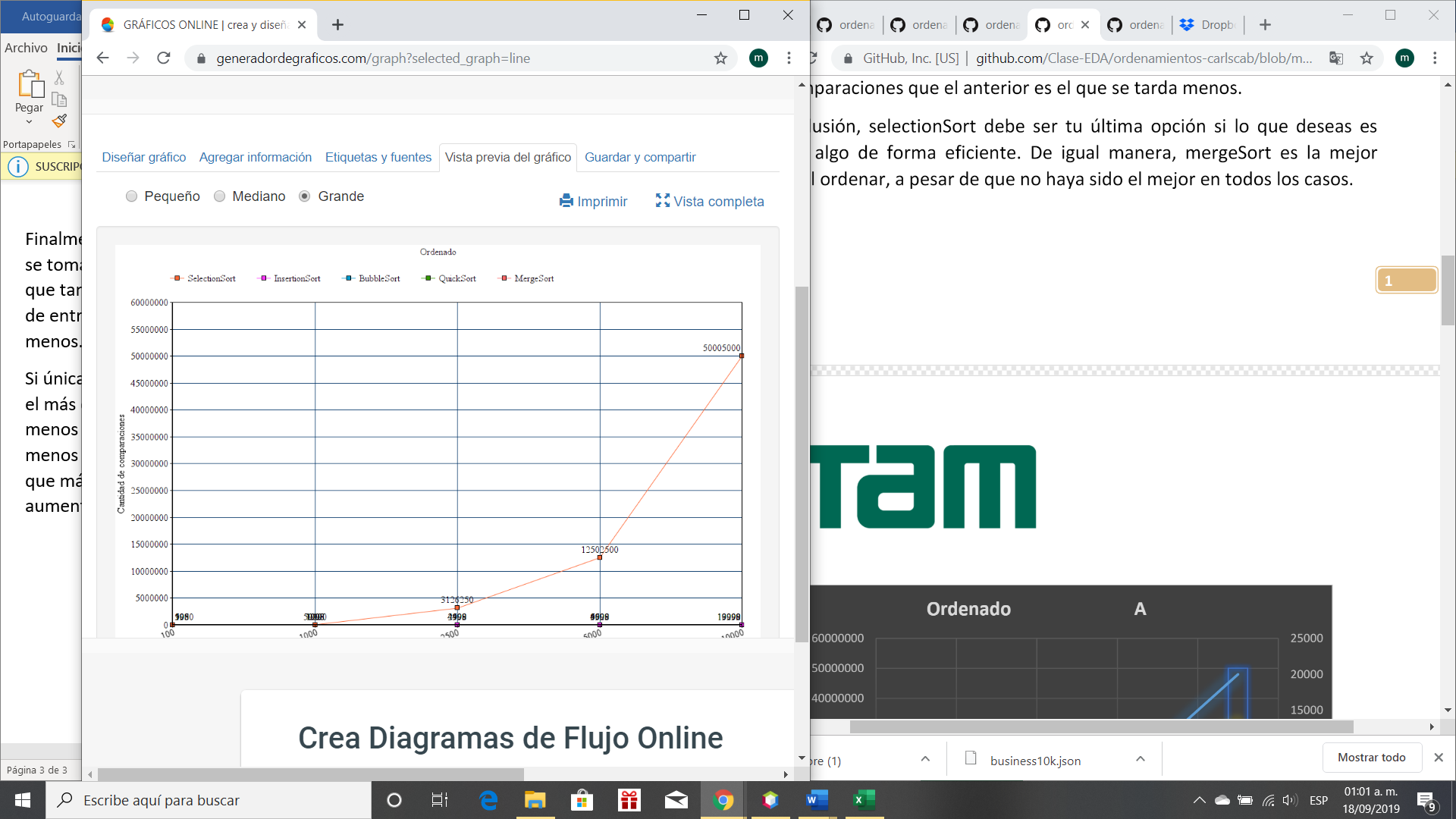


En esta gráfica, los elementos de entrada estuvieron en orden inverso. El algoritmo de merge sort es el que se toma el menor tiempo, aunque el quick sort se toma un tiempo similar. Los algortimos que se toman mayor cantidad de tiempo son selection sort y bubble sort, que también van aumentando conforme incrementan la cantidad de elementos en el arreglo.

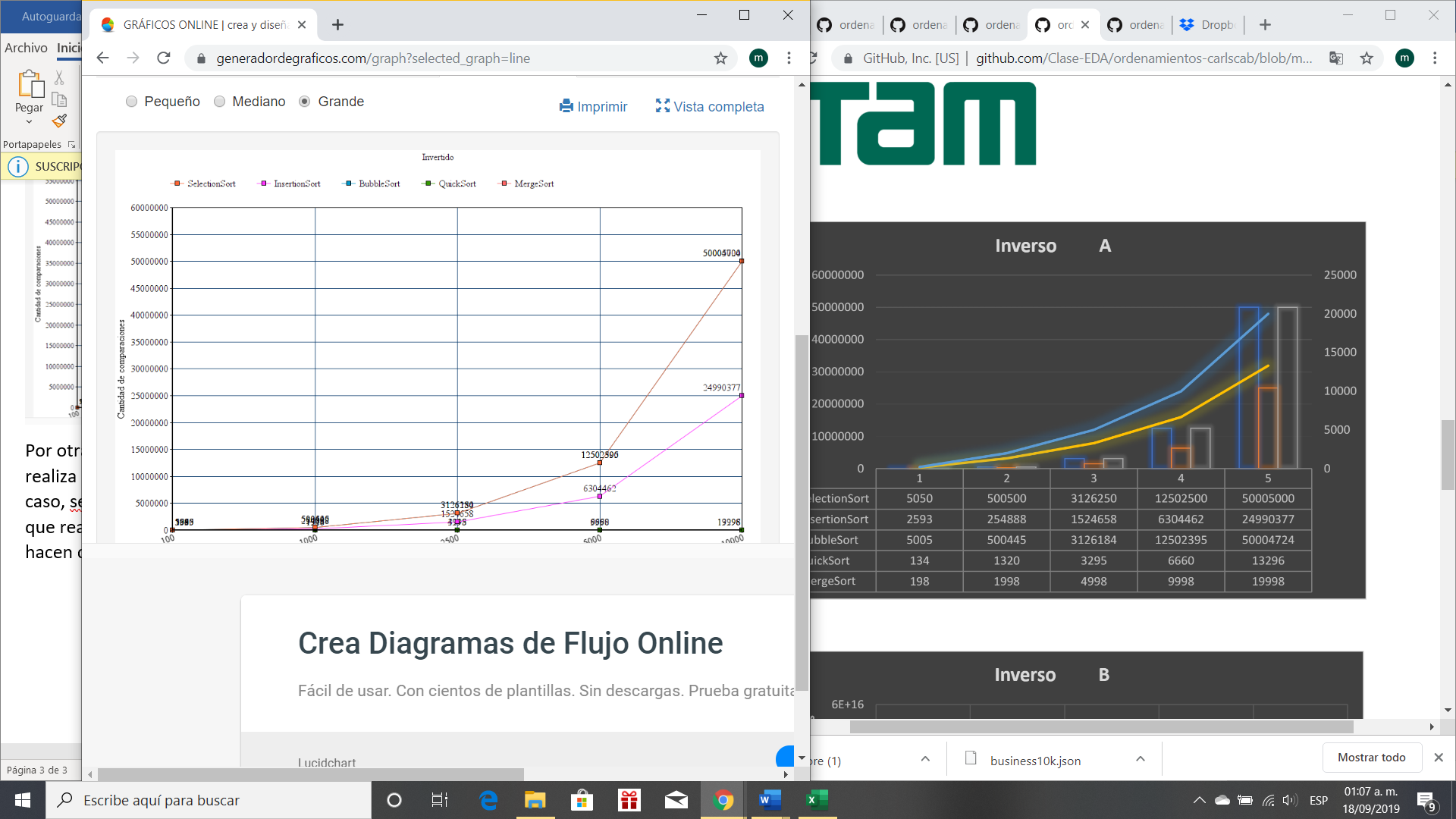


Finalmente, cuando los elementos están en desorden, selection sort es el que se toma la mayor cantidad de tiempo. Le siguen insertion sort y quick sort, que también tardan bastante tiempo conforme incrementan los elementos de entrada. Bubble sort y especialmente merge sort, son los que tardan menos.

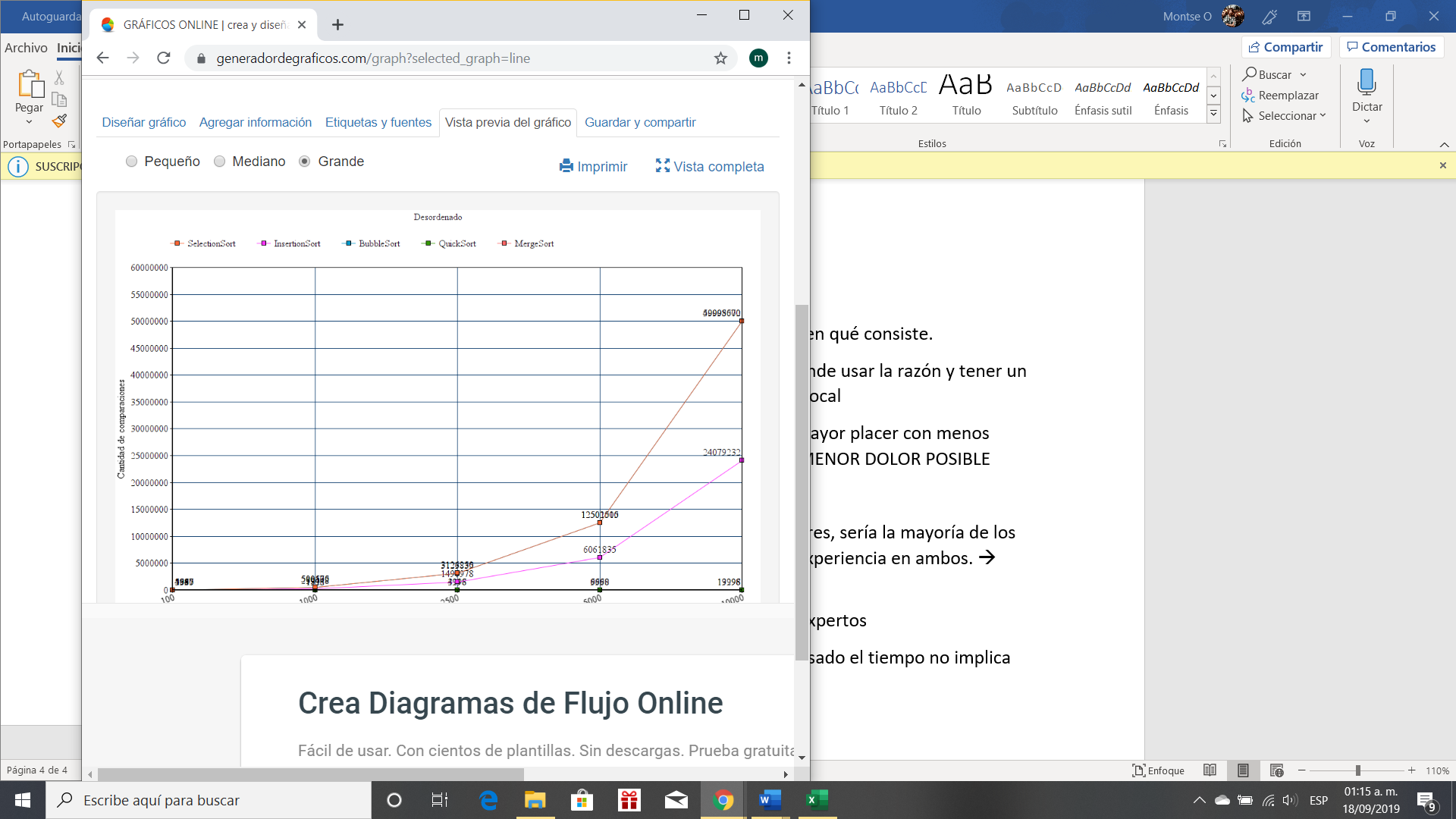
Si únicamente tomáramos en cuenta el tiempo que le toma a cada algoritmo, el más conveniente sería merge sort, porque en general, fue el que se tomó menos tiempo en operar, aún cuando los elementos incrementaban. El menos conveniente sería el selection Sort, ya que fue en todos los casos, el que más tiempo se tardaba, especialmente si los elementos del arreglo aumentaban.



Por otra parte, en esta gráfica se comparan el número de comparaciones que realiza cada algoritmo en relación al número de entradas que recibe. En este caso, selection sort es el que hace más comparaciones y el quick sort es el que realiza el menor número de comparaciones. Insertion sort y bubble sort hacen casi la misma cantidad de comparaciones



Cuando los elementos están invertidos, el selection Sort y el bubble Sortrealizan prácticamente el mismo número de comparaciones, Bubble sort haciendo un poco más, pero ambas incrementan conforme los valores aumentan. Merge Sort y quick sort hacen casi el mismo número de comparaciones, aunque quick sort realiza menos.



Cuando los datos se encuentran en desorden, el que realiza más comparaciones es selection sort, en relación al aumento de elementos. Quick sort es el que realiza menos comparaciones, seguido por merge sort.

Si únicamente tomamos en cuenta el número de comparaciones, selection sort nuevamente es el que hace más comparaciones, por lo que sigue siendo el menos conveniente. Quick sort fue el que realizó el menos número de comparaciones, por lo que en este parámetro, sería el más eficiente.