Carlos Delgado 181866 18 de septiembre de 2019
EDA

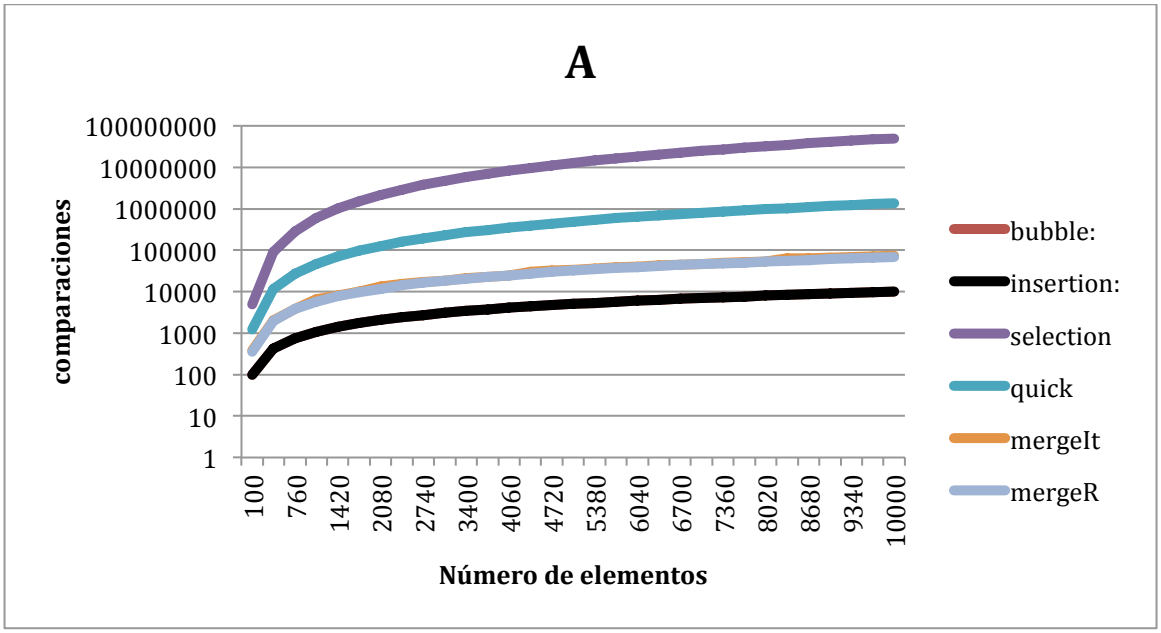
Comparación de algoritmos de ordenamiento

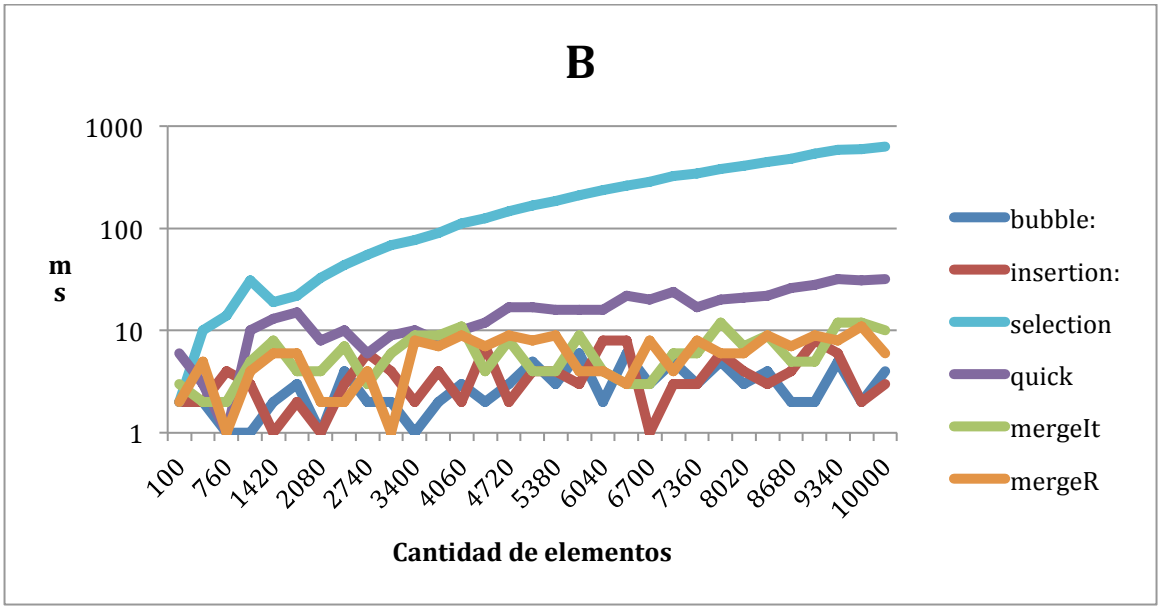
Se analiza empíricamente la eficiencia de los algoritmos de ordenación vistos en clase en función de la cantidad de comparaciones y el número de milisegundos que tarda en ordenar el arreglo.

En todos los casos se utilizaron 30 casos por algoritmo, y el número de elementos está en progresión aritmética desde 100 a 10000. Además, para cada tamaño n, los diferentes algoritmos ordenaron el mismo arreglo para mejorar la comparación, incluso en la parte del arreglo aleatorio. En esta última, para cada número de elementos, se eligieron n elementos al azar, y cada algoritmo ordenó esos mismos n elementos.

Arreglo ordenado

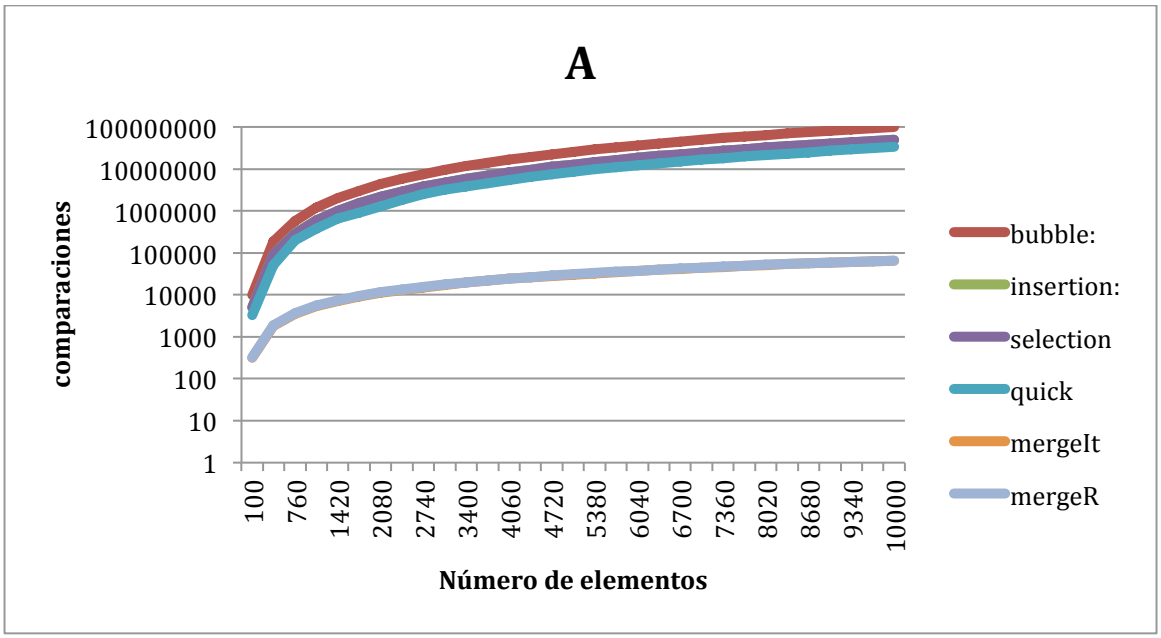
Aquí, aunque no puede apreciarse, la curva generada por bubble sort es la misma que la del insertion sort. En términos de comparaciones, podemos ver que los más eficientes fueron estos últimos, seguidos por las dos implementaciones del merge sort y que el menos eficiente fue el selection sort, pues este siempre compara n(n-1)/2. En términos de tiempo no fue muy diferente, aunque la diferencia entre los merge, bubble e insertion fue menor.

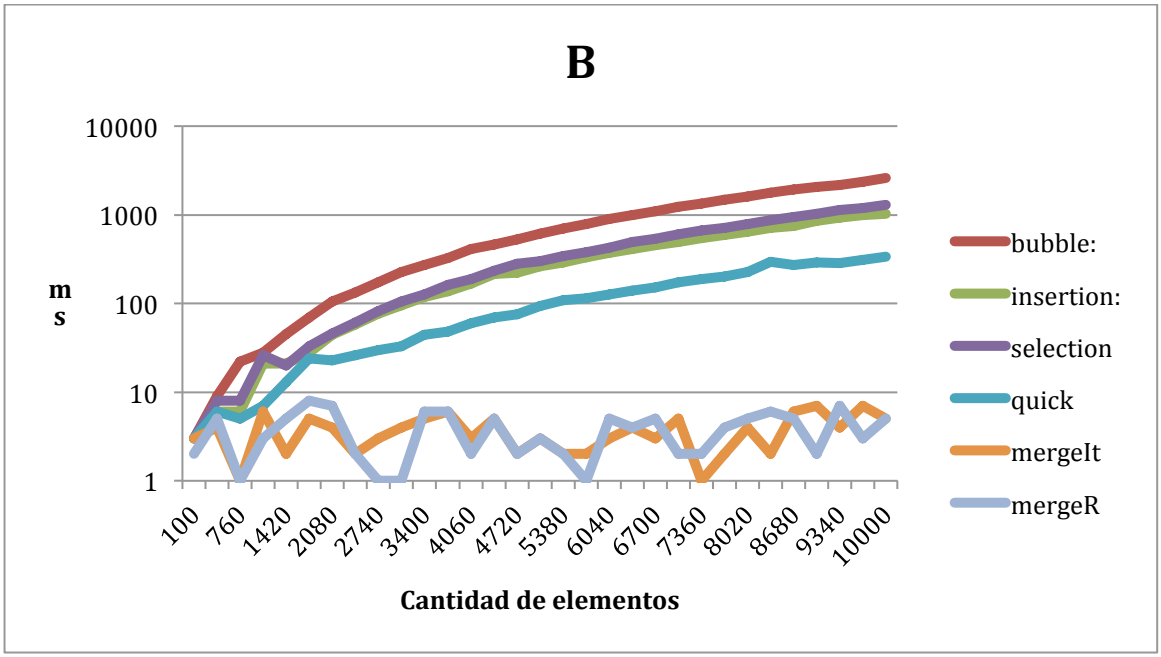




Orden inverso

Aquí no puede observarse la de insertion sort, pues coincide con la de selection sort. En términos de comparaciones, bubble, insertion y quick sort perdieron eficiencia y se fueron a la par que selection sort, mientras que selection y merge sort se mantuvieron casi igual al caso anterior. Aún así, quick sort se mantuvo relativamente coherente. En términos de tiempo los resultados son muy similares, con una gran eficiencia de merge sort.

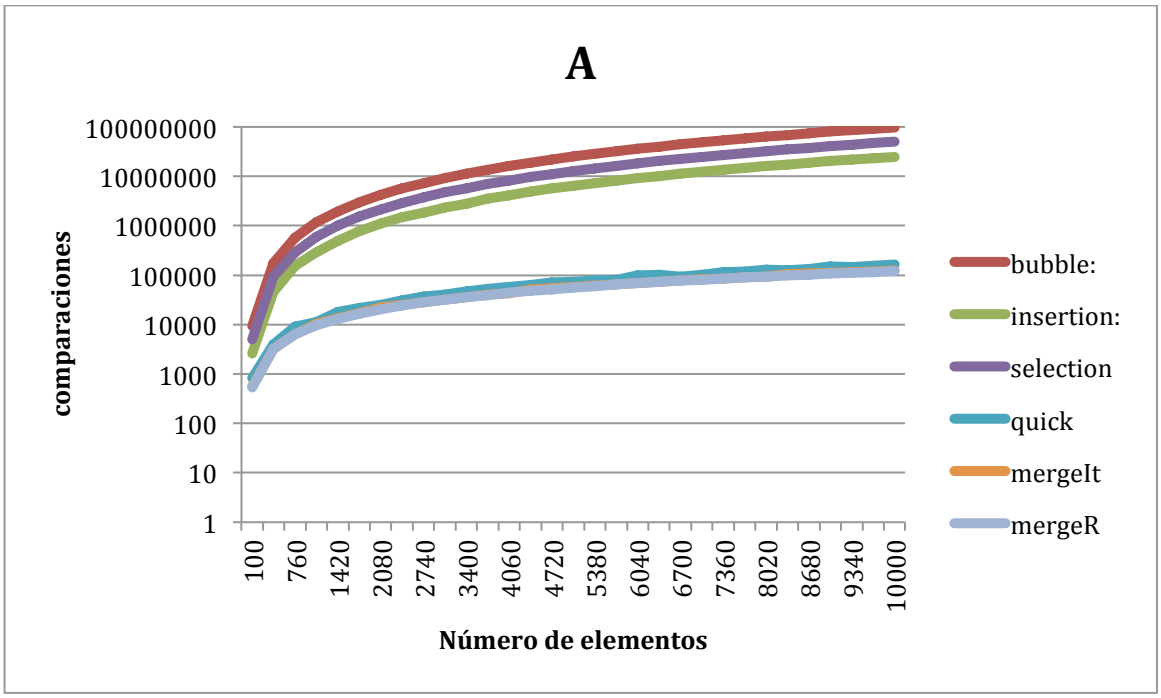




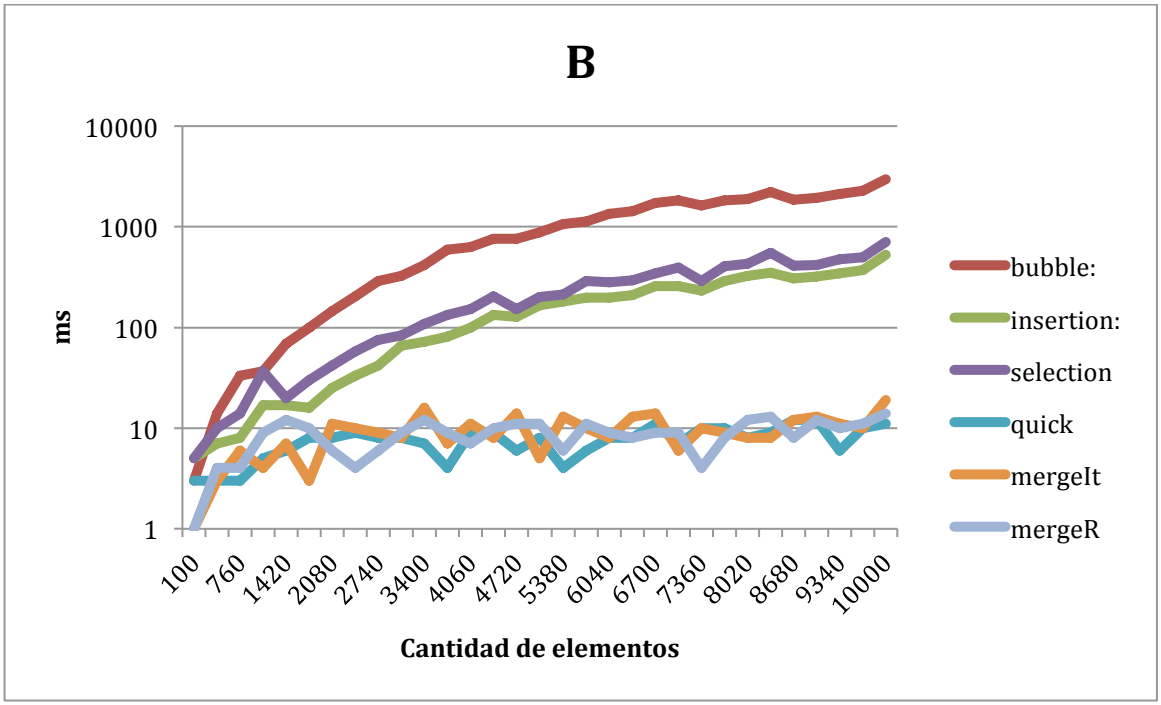
Orden aleatorio

En este caso, en términos de comparaciones, podemos ver que bubble, insertion y selection sort permanecieron menos eficientes, mientras que quick sort ganó eficiencia y quedó ligeramente arriba, pero prácticamente a la par con merge sort, que permaneció eficiente y relativamente igual que en las ocasiones pasadas.

Muy similar fue en tiempo, donde bubble permanece como el más ineficiente y la relación se mantiene entre los otros.



Carlos Delgado 181866
18 de septiembre de 2019
EDA



Conclusiones:
Podemos observar que merge sort fue el algoritmo que combino la eficiencia y consistencia durante los ejemplos. También debemos destacar el caso de orden aleatorio, pues es el más común en términos prácticos; los primeros son raros. Así que quick sort también es un buen algoritmo, con la ventaja de no tener que

Por otro lado, bubble, insertion y selection sort permanecieron en la mayoría con la menor eficiencia. Bubble e insertion fueron los más eficientes, pero sólo en el caso más raro, que tengas que ordenar algo que ya estaba ordenado. Y selection sort fue consistente pero ineficiente.