Estructuras de Datos Avanzadas

Análisis de algoritmos de ordenamiento

Rodrigo González Torres

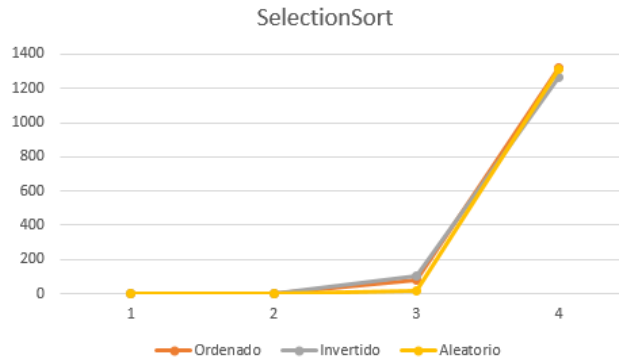
000183873

Fecha de entrega:18/09/19

En este ejercicio se hicieron pruebas de los algoritmos de ordenamiento vistos en clase, con entradas de 10, 100, 1000, 10000 datos, los mismos se miden en mili segundos.

1. SelectionSort

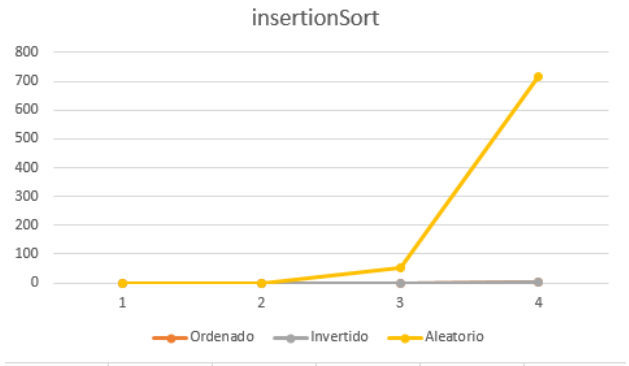
SelectSort				
tamaño	10	100	1000	10000
Ordenado	0	2	82	1320
Invertido	0	0	107	1269
Aleatorio	0	1	15	1312



Como se puede observar en la tabla y en la gráfica, en entradas pequeñas (podríamos incluir aquí también al 100) no hay una gran diferencia entre cómo están acomodados los mismos datos, conforme vamos agregando más datos, se dispersan un poco, pero la diferencia sigue siendo mínima. Para 10000 datos, hizo un total de 50015000 comparaciones. Mientras que para 1000 hizo 501500 comparaciones. Y con una entrada de 10 datos hace 65 comparaciones.

2. insertionSort

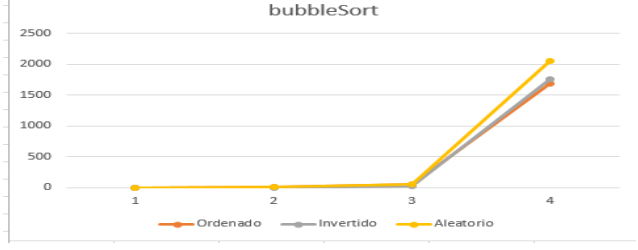
insertiomSort				
tamaño	10	100	1000	10000
Ordenado	0	0	0	2
Invertido	0	0	0	2
Aleatorio	0	0	54	715



Como se puede observar, solo nos muestra una variación cuando esta de forma aleatorio, suponiendo que la mayoría de los casos estará acomodado de una forma no secuencial, este resulta muy poco práctico. Con una entrada de 10000 datos, si el mismo se encuentra ordenado de alguna forma hace un total de 9999 comparaciones por cada uno, mientras que de forma aleatorio con la misma entrada nos hace 24256413 comparaciones.

3. bubbleSort

bubbleSort				
tamaño	10	100	1000	10000
Ordenado	0	3	53	1686
Invertido	0	0	16	1752
Aleatorio	0	1	53	2050



El bubbleSort nos muestra un incremento en el tiempo de casi 20% en el aleatorio con respecto al ordenado con una entrada de 10000, con la misma nos hace 50004999 comparaciones por cada uno.

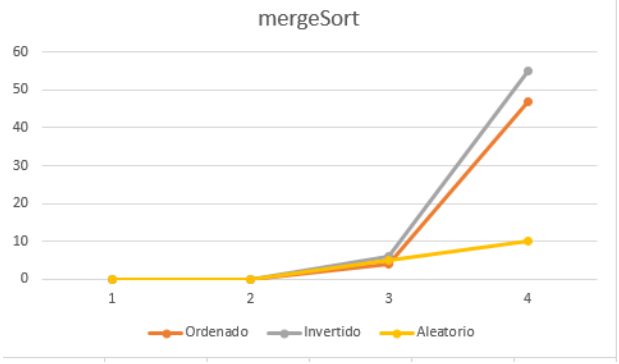
4. quickSort

quickSort				
tamaño	10	100	1000	10000
Ordenado	0	0	1	3
Invertido	0	0	0	0
Aleatorio	0	1	115	1474

Algo raro de este algoritmo es que en el mejor de los casos, nos muestra un tiempo muy pequeño, y en el peor de los casos, nos muestra un tiempo muy reducido, lo que si varia y bastante es el número de comparaciones que hace para 10000 entradas, las cuales son: 13699, 22442, 48627670, las cuales aumentan considerablemente dependiendo de qué tan desordenado este, lo cual es lógico, por el tipo de ordenamiento.

5. mergeSort

mergeSort				
tamaño	10	100	1000	10000
Ordenado	0	0	4	47
Invertido	0	0	6	55
Aleatorio	0	0	5	10



Aquí podemos observar que no hay una gran variación en los tiempos, y que son los tiempos más bajos, también, no afecta el que tanto este desordenado u ordenado, hace la misma cantidad de comparaciones en los tres casos, que en una entrada de 10000, es 250848.

Conclusión

Como conclusión podemos decir, que el mergeSort es el mejor algoritmo de ordenamiento, ya que no solo nos muestra los mejores tiempos, sino nos muestra tiempos constantes, y que no se disparan dependiendo de qué tan desordenado este.