Estructura de Datos Avanzadas

Comparación de Algoritmos de Ordenamiento

Carla Lorena Caballero Enciso

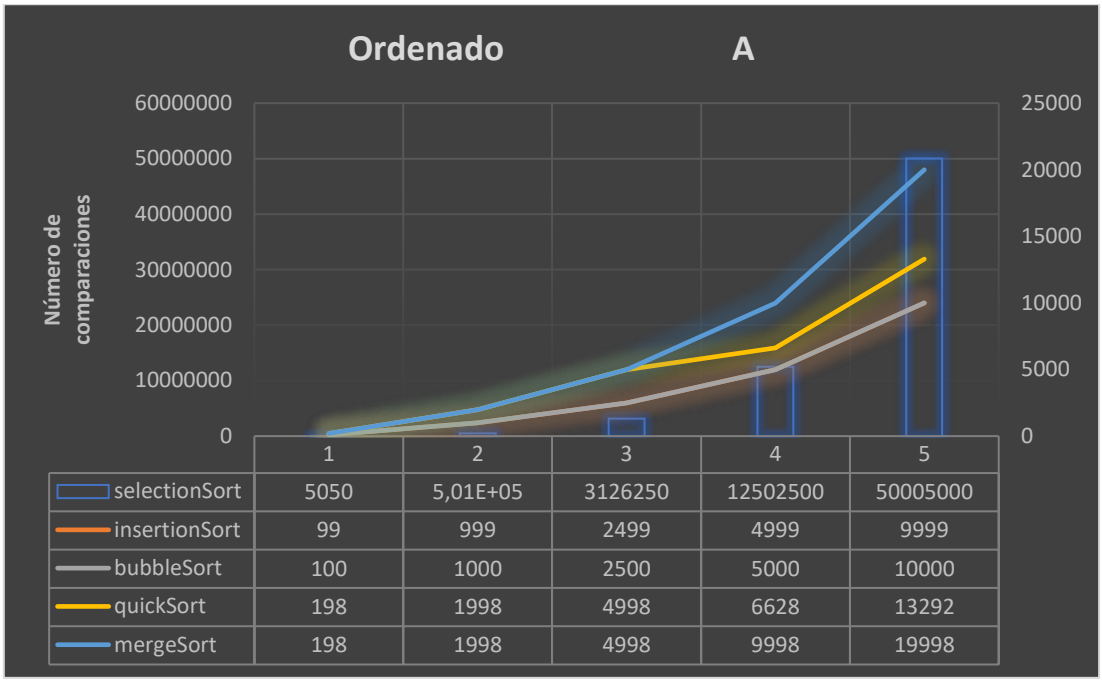
Las siguientes gráficas representan el desempeño de los códigos vistos en clase. Las gráficas A representan el número de comparaciones y las B el tiempo de ejecución en nano segundos.

Al observar las gráficas cuando están ordenadas podemos notar que selectionSort es el que se tarda más y hace más comparaciones. Además, quickSort se tarda más o menos el mismo tiempo, pero hace menos comparaciones. También, cabe recalcar que insertionSort y bubbleSort hacen casi el mismo número de comparaciones, pero bubbleSort se tarda menos.

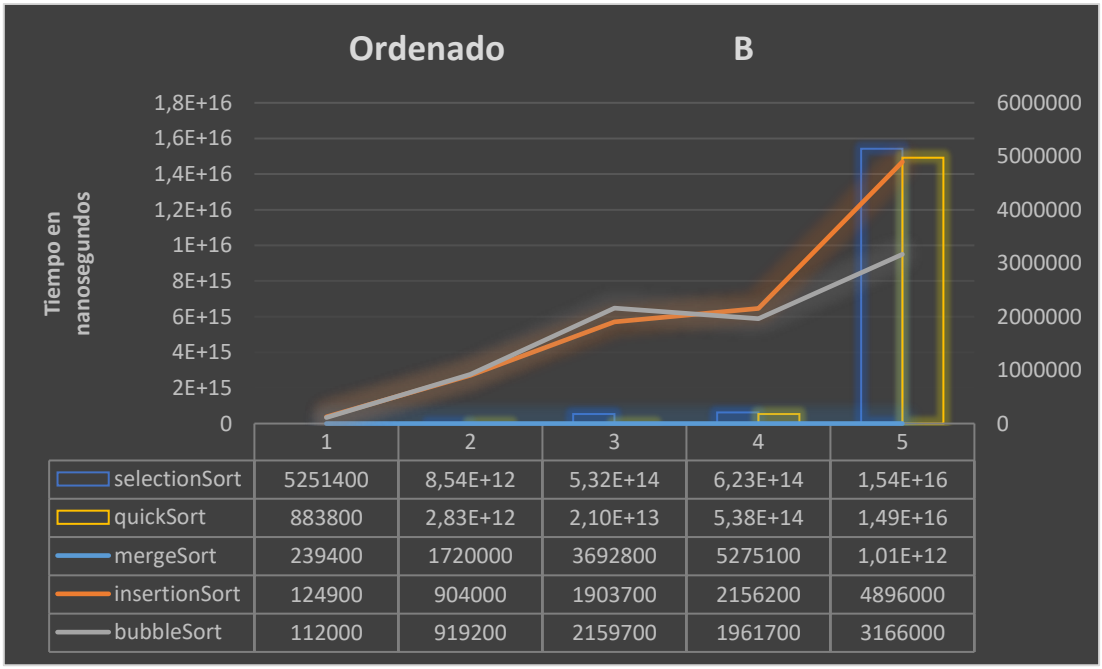
Por otra parte, cuando están los datos en orden inverso tanto selectionSort como bubbleSort hacen casi el mismo número de comparaciones, sólo que bubbleSort se tarda mucho más. Aquí, insertionSort también hace muchas comparaciones, pero su tiempo de ejecución es menor que cualquier otro método. Asimismo, mergeSort y quickSort tienen unas rectas similares, a excepción de que quickSort lo hace más rápido y con menos comparaciones.

Por último, cuando los datos se presentan en un orden aleatorio podemos notar que selectionSort es la peor opción, ya que se tarda mucho y hace muchas comparaciones. Incluso, insertionSort se tarda menos que los otros métodos, no obstante, hace más comparaciones que quickSort y mergeSort. También podemos decir que quickSort es el que hace menos comparaciones, pero es el segundo en tardarse más. Finalmente, mergeSort, a pesar de hacer más comparaciones que el anterior es el que se tarda menos.

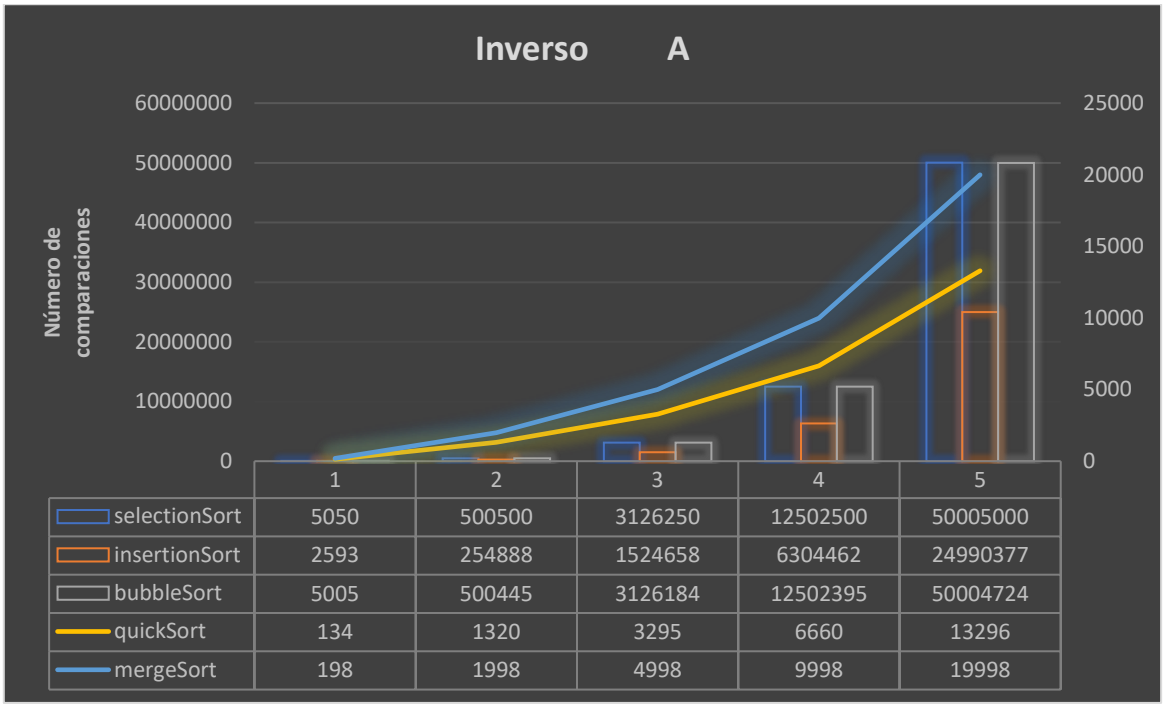
En conclusión, selectionSort debe ser tu última opción si lo que deseas es ordenar algo de forma eficiente. De igual manera, mergeSort es la mejor opción al ordenar, a pesar de que no haya sido el mejor en todos los casos.



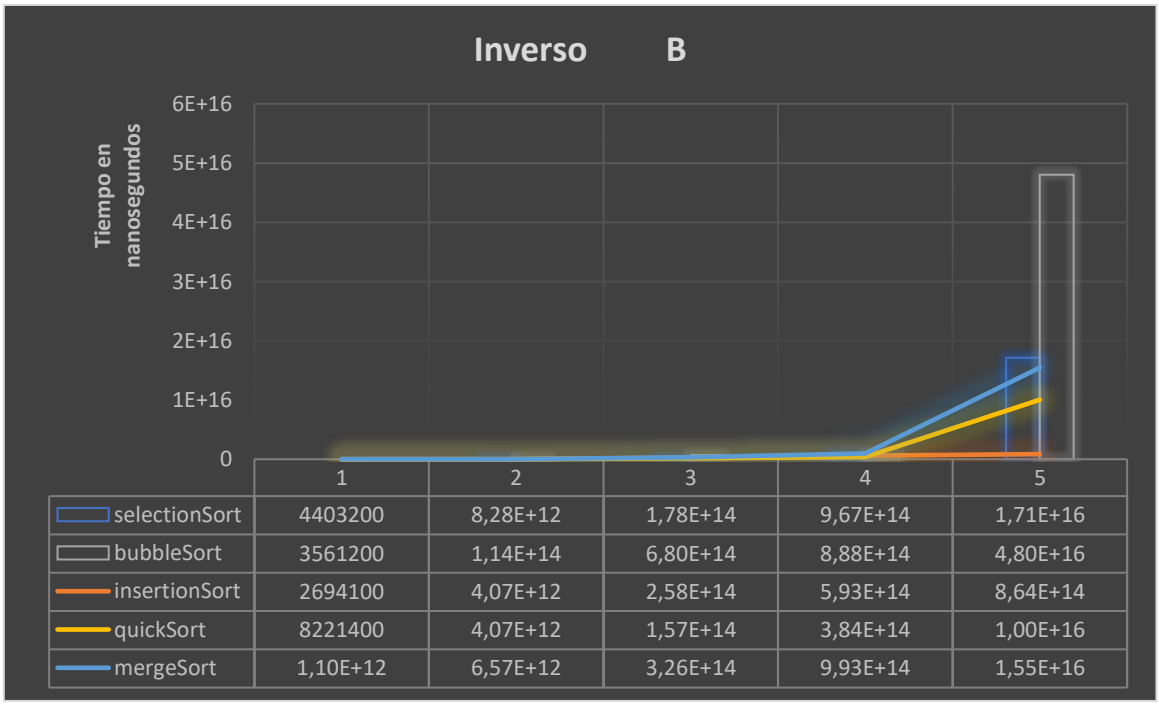
	1	2	3	4	5
selectionSort	5050	5,01E+05	3126250	12502500	50005000
insertionSort	99	999	2499	4999	9999
bubbleSort	100	1000	2500	5000	10000
quickSort	198	1998	4998	6628	13292
mergeSort	198	1998	4998	9998	19998



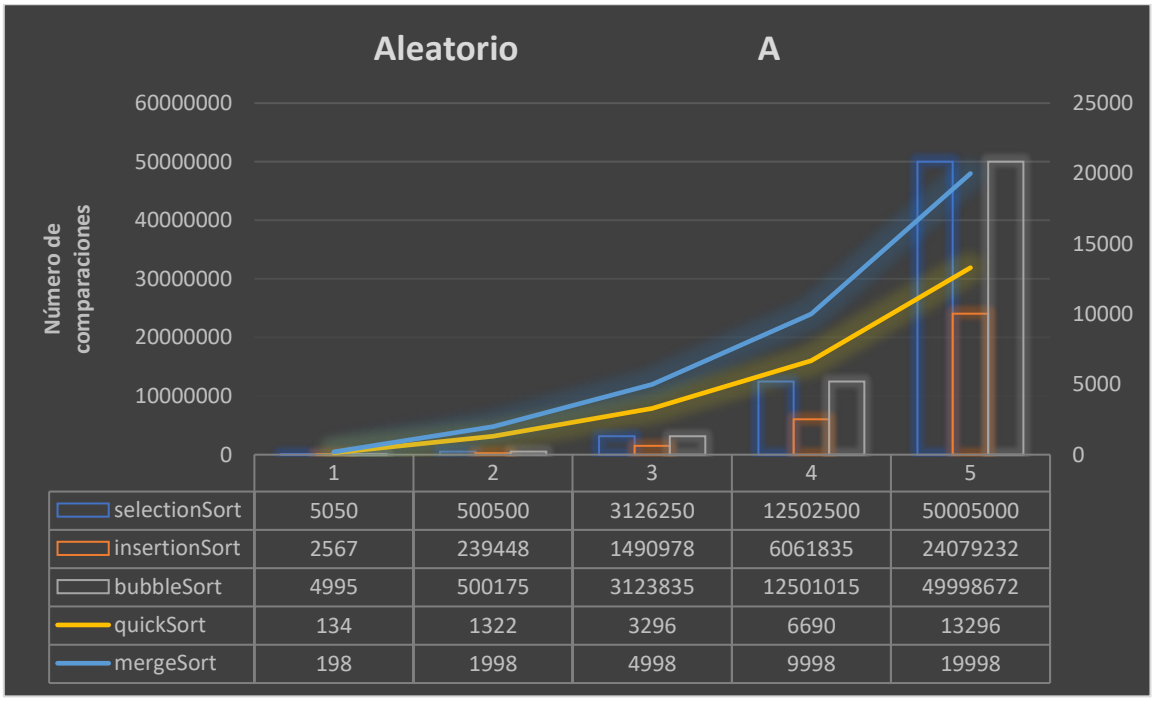
	1	2	3	4	5
selectionSort	5251400	8,54E+12	5,32E+14	6,23E+14	1,54E+16
quickSort	883800	2,83E+12	2,10E+13	5,38E+14	1,49E+16
mergeSort	239400	1720000	3692800	5275100	1,01E+12
insertionSort	124900	904000	1903700	2156200	4896000
bubbleSort	112000	919200	2159700	1961700	3166000



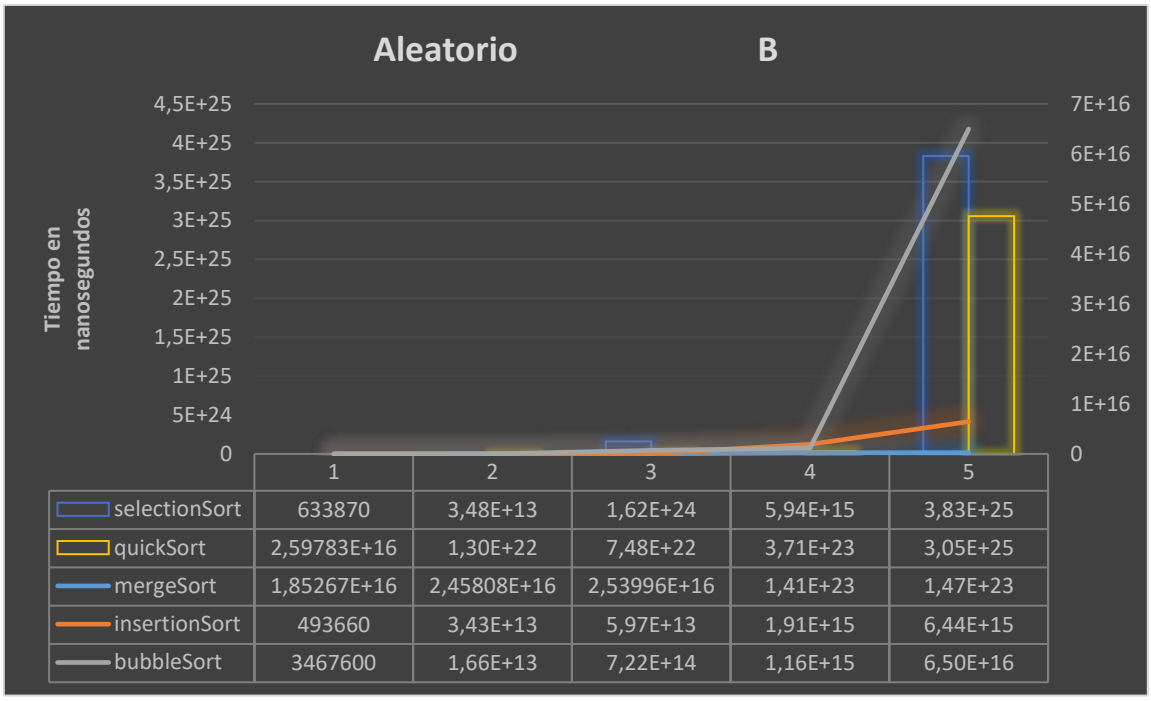
	1	2	3	4	5
selectionSort	5050	500500	3126250	12502500	50005000
insertionSort	2593	254888	1524658	6304462	24990377
bubbleSort	5005	500445	3126184	12502395	50004724
quickSort	134	1320	3295	6660	13296
mergeSort	198	1998	4998	9998	19998



	1	2	3	4	5
selectionSort	4403200	8,28E+12	1,78E+14	9,67E+14	1,71E+16
bubbleSort	3561200	1,14E+14	6,80E+14	8,88E+14	4,80E+16
insertionSort	2694100	4,07E+12	2,58E+14	5,93E+14	8,64E+14
quickSort	8221400	4,07E+12	1,57E+14	3,84E+14	1,00E+16
mergeSort	1,10E+12	6,57E+12	3,26E+14	9,93E+14	1,55E+16



	1	2	3	4	5
selectionSort	5050	500500	3126250	12502500	50005000
insertionSort	2567	239448	1490978	6061835	24079232
bubbleSort	4995	500175	3123835	12501015	49998672
quickSort	134	1322	3296	6690	13296
mergeSort	198	1998	4998	9998	19998



	1	2	3	4	5
selectionSort	633870	3,48E+13	1,62E+24	5,94E+15	3,83E+25
quickSort	2,59783E+16	1,30E+22	7,48E+22	3,71E+23	3,05E+25
mergeSort	1,85267E+16	2,45808E+16	2,53996E+16	1,41E+23	1,47E+23
insertionSort	493660	3,43E+13	5,97E+13	1,91E+15	6,44E+15
bubbleSort	3467600	1,66E+13	7,22E+14	1,16E+15	6,50E+16