Escuela de Ingeniería
Departamento de Computación
Estructuras de datos y algoritmos fundamentales
(TC1031)
Profesor: Dr. Leonardo Chang

Tecnológico de Monterrey

Actividad práctica. Algoritmos de ordenamiento

Título	**Análisis de algoritmos de ordenamiento**
Aprendizaje esperado (objetivo)	El alumno demostrará su capacidad para programar diferentes algoritmos de ordenamiento y medir el tiempo de ejecución de los mismos bajo determinadas condiciones, así como analizar e interpretar los resultados obtenidos, comparando los diferentes algoritmos.
Instrucciones	Utilizando programación genérica (templates) y sobrecarga de operadores en C++, programa una clase Sorter que incluya los siguientes métodos y atributos: Métodos: Selection sort Bubble sort Insertion sort Merge sort Quick sort Cada método debe recibir por referencia el vector a ordenar. Genere un arreglo de 100 000 números enteros de manera aleatoria. Realice el ordenamiento de una copia del arreglo inicial (debemos hacer copiar sino se intentará ordenar un arreglo ya ordenado) Mida el tiempo de ejecución de cada caso y complete las tablas que aparecen más adelante en este documento. Genere algunas gráficas (en Google Sheets) comparando los resultados de todos los algoritmos y sus tiempos de ejecución. Analice e interprete los resultados alcanzados. Realice una copia de este documento en Google Docs y complete las secciones indicadas más adelante. Suba a la plataforma Canvas el archivo con sus resultados. Suba a Github todos los códigos programados. No se aceptan trabajos fuera de fecha ni por correo electrónico.
Lugar en que se llevará a cabo	Casa

Forma de trabajo	Individual
Recursos	Foros de información en Internet Wikipedia (http://www.wikipedia.org) Códigos de algoritmos vistos en la materia Computadora
Tiempo estimado	5 horas

Respuestas

Repositorio de GitHub:

https://github.com/louloubadillo/DS_TC1031/tree/master/Class/Class6

Tablas con los resultados de las mediciones:

Algoritmos de ordenamiento												
Tabla de resultados a completar (*tiempo en ms*)												
Alg.	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**Media**	**Stdev**
Selection sort	30053,80	33082,80	33373,10	31667,0	31743,60	31398,70	30554,50	31033,20	31857,10	33508,10	31827,190	1173,980
Bubble sort	32538,90	36878,70	33279,70	33279,40	32773,50	32813,60	32056,20	31241,50	32525,90	32049,40	32943,680	1512,138
Insertion sort	1,380	1,000	0,898	0,934	0,950	0,906	0,923	0,896	0,896	0,867	0,965	0,150
Merge Sort	27,970	29,70	23,490	23,330	32,887	29,842	22,688	23,333	21,750	21,927	25,692	4,014
Quick sort												

Interpretación de los resultados:

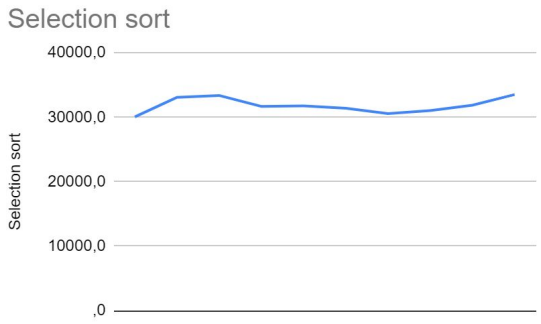
El algoritmo más lento y, por lo tanto, menos eficiente en relación a su complejidad temporal, es Bubble Sort. Esto tiene sentido debido a que es un algoritmo muy intuitivo y sencillo de programar, pero que requiere de muchos recursos.

Selection Sort tuvo tiempos de ejecución muy similares al de Bubble Sort. Por lo que la implementación sí fue más eficiente, pero sigue siendo $O(n^2)$.

Merge Sort sería el siguiente algoritmo en cuanto a su complejidad temporal. Se supone que debió haber sido el más rápido de todos, sin embargo mi implementación del algoritmo no es la más eficiente que existe, lo cual afectó su tiempo de ejecución e hizo que requiriera más recursos que Insertion Sort. A pesar de esto, sí se notó una mayor eficiencia, comparado con los dos algoritmos anteriormente mencionados.

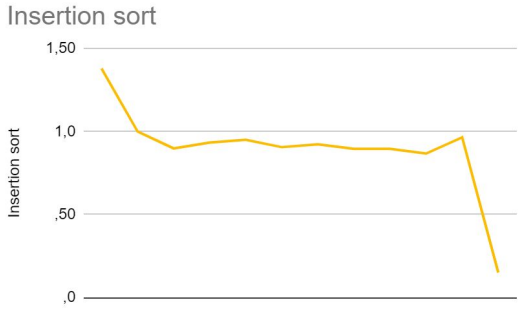
Finalmente, el algoritmo que mostró ser más eficiente fue Insertion Sort. Este algoritmo es $O(n^2)$, pero a diferencia de los dos primeros, es $\Omega(n)$.

Gráficas comparativas

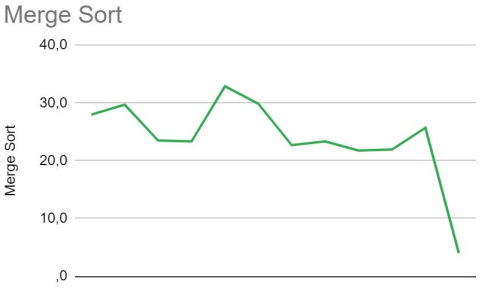


Gráfica 1. Tiempo de ejecución de selection sort

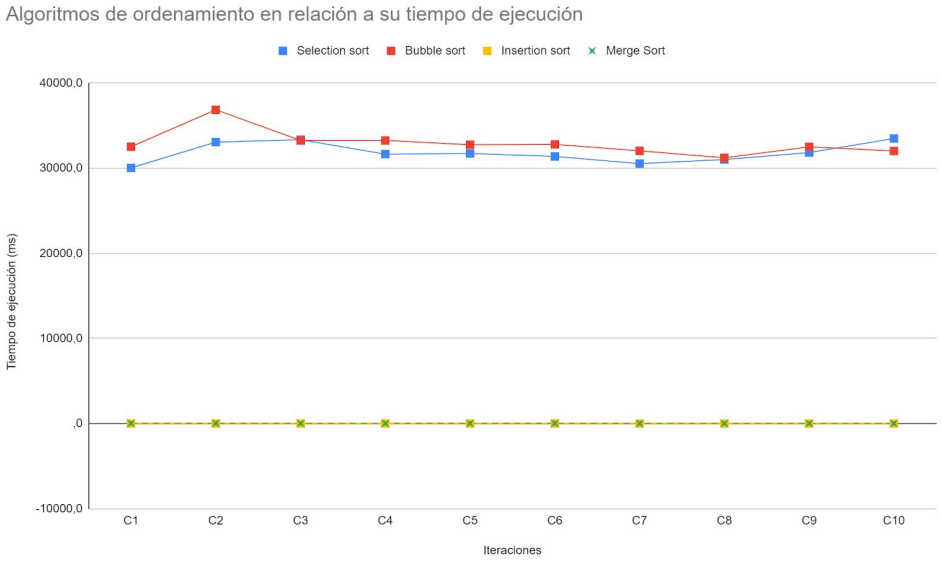
Gráfica 2. Tiempo de ejecución de bubble sort



Gráfica 3. Tiempo de ejecución de insertion sort



Gráfica 4. Tiempo de ejecución de merge sort



Gráfica 5. Tiempo de ejecución de los cuatro algoritmos de ordenamiento