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 segundos*)												
Alg.	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**Media**	**Stdev**
Selection sort	28.9902	30.4434	31.0545	29.7004	32.3766	34.8206	33.4995	35.5602	39.1854	40.1796	33.58104	3.8618378
Bubble sort	73.2229	74.169	77.0348	78.5387	87.2815	87.8256	88.7731	88.3101	102.885	102.194	86.02347	10.565234 97
Insertion sort	16.7499	15.9337	16.8432	17.0757	18.7217	16.7941	20.7647	20.6872	19.6745	18.8235	18.20682	1.7612771 7
Merge Sort	0.0364287	0.0344385	0.0329725	0.0336716	0.0453541	0.0340377	0.046525	0.0460884	0.0450545	0.0326931	0.0387264 1	0.0061436 23
Quick sort												

Interpretación de los resultados:

El algoritmo más lento y, por lo tanto, menos eficiente en relación a su complejidad temporal, fue Bubble Sort. Esto tiene sentido debido a que es un algoritmo muy intuitivo y sencillo de programar, pero que requiere de muchos recursos, siendo $O(n^2)$.

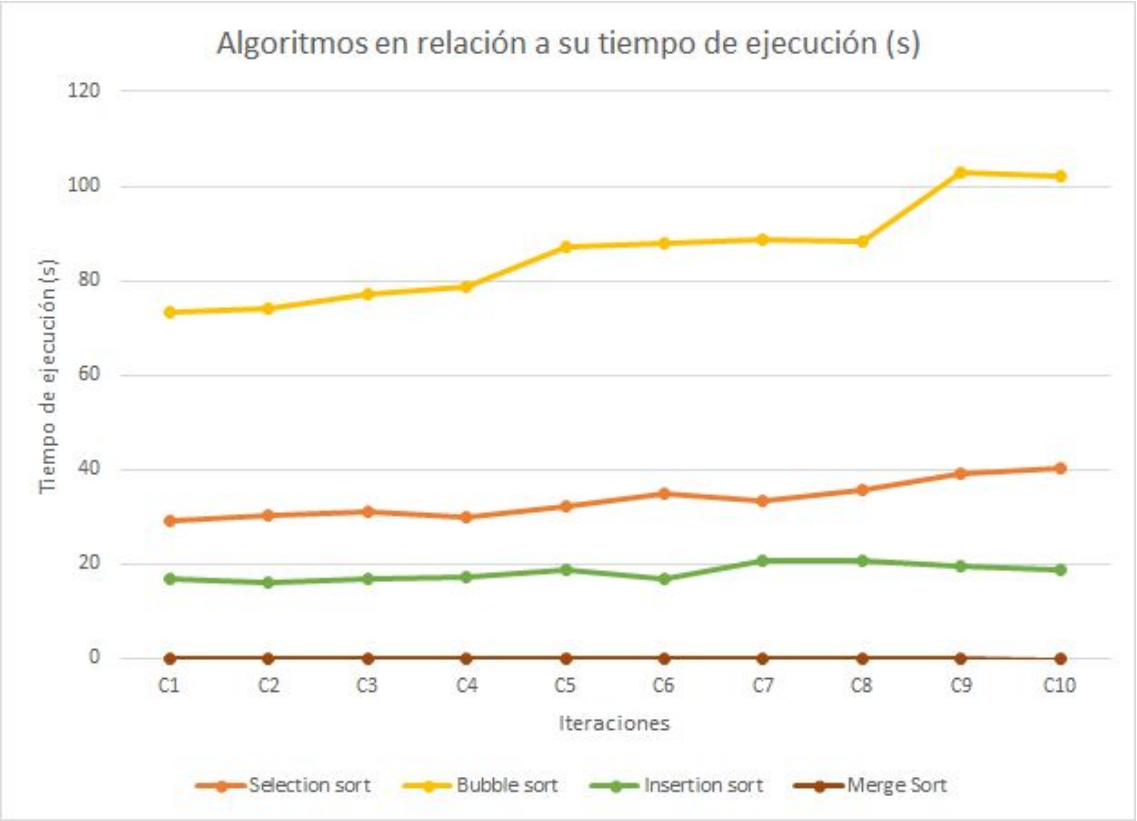
En cuanto a tiempo de ejecución, el siguiente algoritmo es Selection Sort. Este algoritmo definitivamente mostró ser más eficiente que Bubble Sort, incluso teniendo la misma complejidad temporal $O(n^2)$, sin embargo, sigue siendo una implementación que requiere de una gran cantidad de recursos, especialmente al incrementar n.

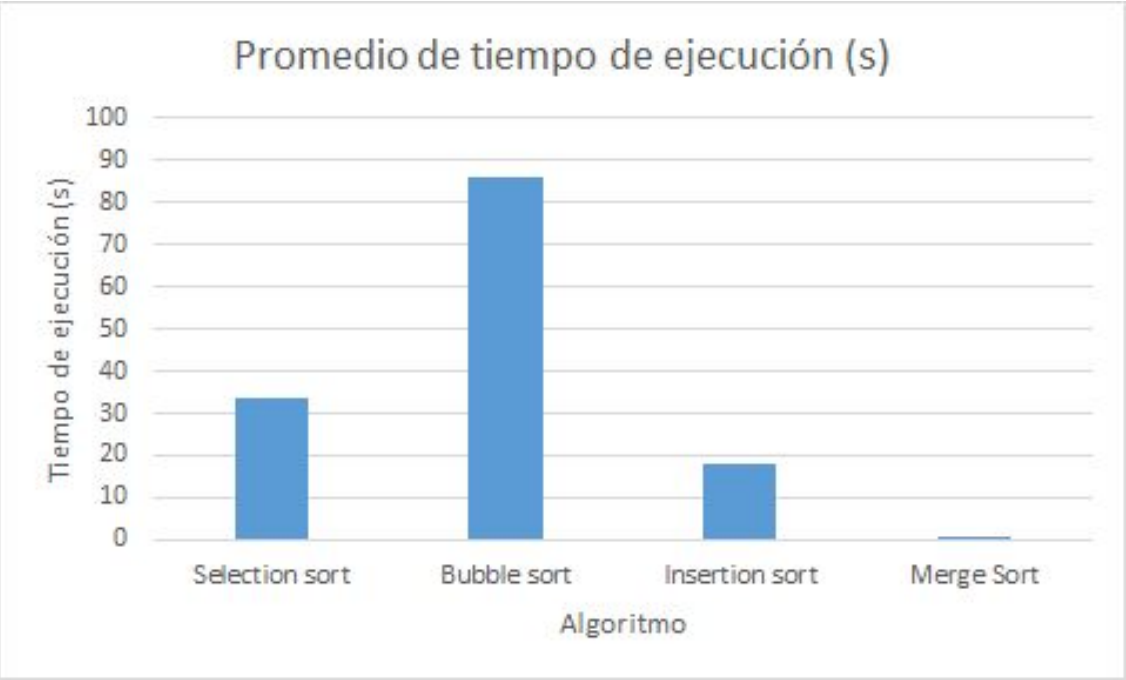
El siguiente algoritmo es Insertion Sort. Este también es $O(n^2)$, pero a diferencia de los dos anteriores, es $\Omega(n)$.

Finalmente, el algoritmo que mostró ser más eficiente fue Merge Sort. Esto hace mucho sentido debido a que es $O(n \log n)$. La diferencia entre el máximo de este algoritmo y el de cualquier otro de los que implementamos anteriormente es impresionante. Esto no sucede con n menores, por lo que podemos concluir que conviene la implementación

de este algoritmo de ordenamiento cuando se va a trabajar con varios elementos dentro de un arreglo.

Gráficas comparativas




Promedio de tiempo de ejecución (s)
Tiempo de ejecución (s)
100
90
80
70
60
50
40
30
20
10
0
Selection sort
Bubble sort
Insertion sort
Merge Sort
Algoritmo