Comparación de eficiencia en algoritmos de ordenamiento

Luis Carlos Quesada R. B65580

***Resumen*—Análisis de eficiencia en algoritmos de ordenamientos de datos, para obtener datos y crear un punto comparativo entre estos.**

I. Introducción

Incluso antes del inicio de las ciencias de la computación, el ordenamiento de conjuntos de enteros con tamaño n ha sido una tarea necesaria para el funcionamiento de múltiples algoritmos, dicho problema ha originado múltiples soluciones, unas más complejas que otras, y de manera análoga, más eficientes que otras.

En el presente documento se lleva a cabo un análisis sobre la eficiencia de 3 algoritmos de ordenamiento, como objetivo se propone buscar y crear datos suficientes para poder establecer un punto de comparación entre dichos algoritmos, y de esta forma, considerar beneficios y debilidades, ademas de identificar casos donde estos algoritmos pueden resultar más efectivos.

II. Metodología

Para la obtención de los datos que se busca comparar, era necesario crear una forma de medición del tiempo y eficiencia de cada algoritmo individualmente, para esto, utilizando el lenguaje de programación C++ se ha realizado una implementación de los algoritmos a estudiar, a su vez, se ha escrito un modulo de pruebas utilizando la clase *CTime* y de esta forma, realizar mediciones del tiempo en segundos necesario para completar una tarea con n elementos, el funcionamiento de este modulo se explicará de forma breve a continuación. Inicialmente se selecciona un algoritmo, a continuación se selecciona una cantidad de elementos a ordenar, llamaremos a esta cantidad n, y seguidamente, se realizan 5 pruebas de ordenamiento de tamaño n, al inicio de cada prueba se inicia un contador y se obtienen sus resultados al fin de la prueba, al finalizar las 5 pruebas se calcula un tiempo promedio para cada prueba y se almacena en un archivo de texto, esto se repite hasta hacer 5 pruebas para cada cantidad n de elementos, las cuales se han fijado en 200,000, 400,000, 800,000 y 1,200,000 en cada uno de los 3 algoritmos.

Los datos recolectados en las pruebas de eficiencia son presentados dentro de este documento en la sección de resultados III mediante tablas y gráficos, de esta manera facilitando el análisis y las conclusiones posibles a las preguntas de la introducción. I

III. Resultados

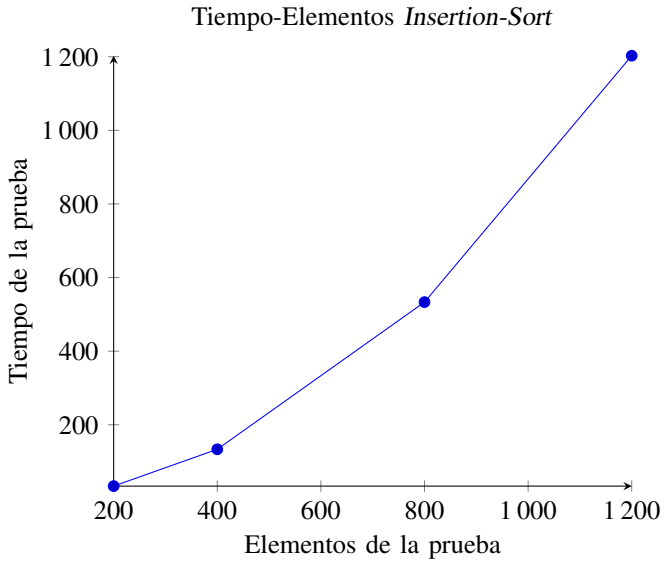
A continuación se expondrán los resultados obtenidos con el modulo de prueba mencionado en la sección II del documento.

Inicialmente se evaluó el ordenamiento por inserción, seguidamente se evaluó el algoritmo de ordenamiento por selección y finalmente se evalúa el algoritmo de ordenamiento por mezcla.

III-A. Insertion-sort

El algoritmo de ordenamiento por selección presenta un crecimiento en tiempo alto según la cantidad de elementos a ordenar, pero en casos de elementos *casi* [1] ordenados tiene tiempos de ejecución bastante buenos, aún así como se muestra en la figura 1, su tiempo de ejecución crece de forma acelerada.

Figura 1. Relaciones tiempo-tamaño *Insertion-Sort* - Grafico



III-B. Selection-Sort

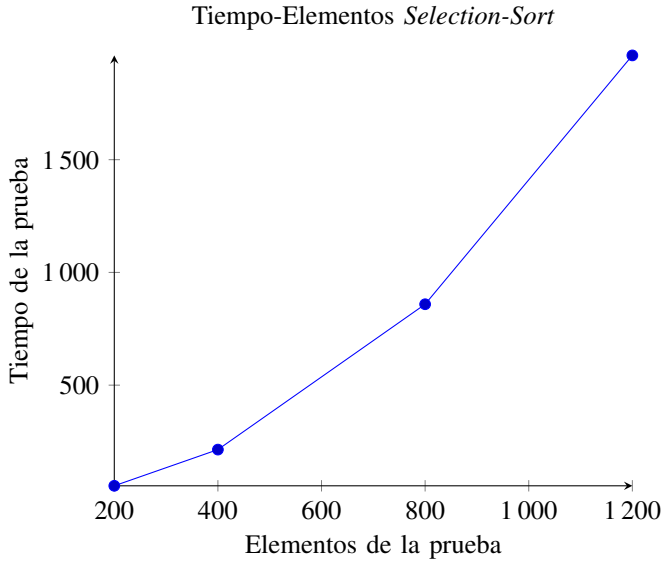
El algoritmo de ordenamiento por selección, sencillo de implementar, posee un tiempo de ejecución que crece de forma exponencial, como se puede observar en la figura 2, los tiempos de ejecución en cada prueba se mantuvieron muy constantes, como se puede apreciar en la figura 6, por lo que pareciera indicar que tanto su mejor como peor caso, tienen el mismo tiempo de ejecución para n elementos.

III-C. Merge-Sort

En el caso de el algoritmo de ordenamiento por mezcla los resultados obtenidos son realmente pequeños, incluso para

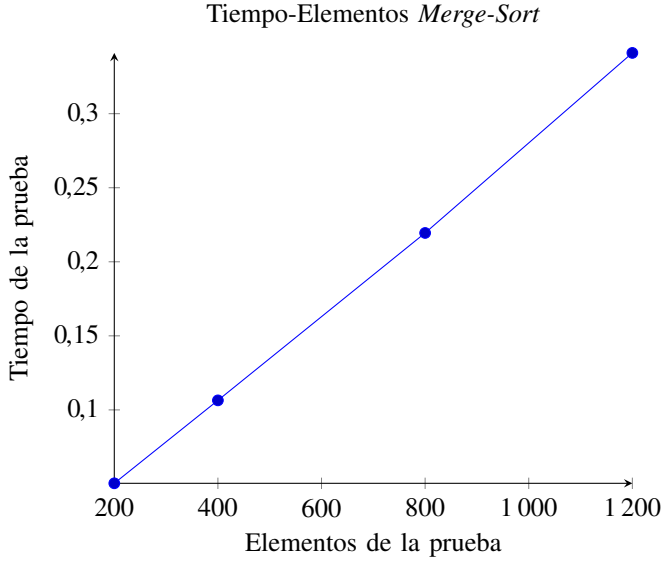
[1] El que tan ordenado está, es relativo al uso que se le dará al arreglo de elementos

Figura 2. Relaciones tiempo-tamaño *Selection-Sort* - Grafico



grandes conjuntos de elementos, los tiempos de ejecución son realmente rápidos, tal como se puede apreciar en la figura 3, este algoritmo ordena cantidades enormes de elementos en un tiempo realmente pequeño, y dependiendo de la forma en que se implementa, puede ser muy económico en espacio de memoria, a pesar de eso, al ser recursívo exige una cantidad de memoria mayor a los algoritmos anteriores.

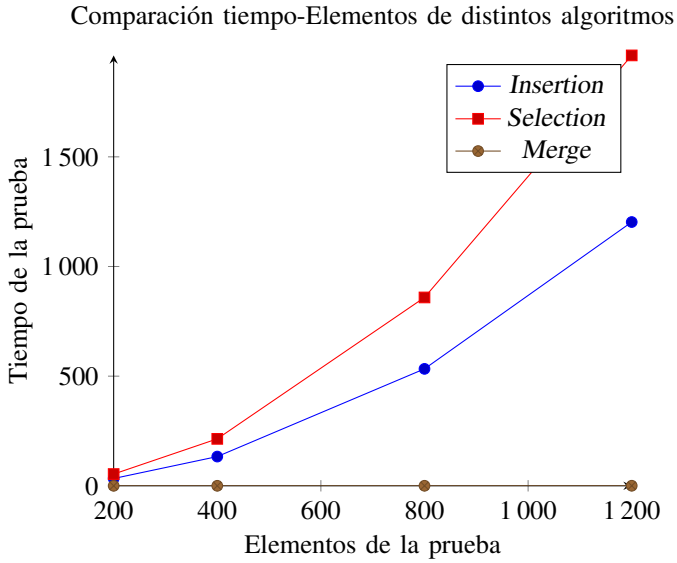
Figura 3. Relaciones tiempo-tamaño *Merge-Sort* - Grafico



III-D. Resultados finales

Los resultados de la sección III han sido suficientes para formar un gráfico comparativo de los 3 algoritmos, donde se muestra la eficiencia Tiempo-Trabajo de los algoritmos, y de esta forma, mas facilmente formar algunas conclusiones.

Figura 4. Resultados Finales



IV. Conclusión

Para concluir, se ha encontrado que el algoritmo de ordenamiento por inserción es similar al algoritmo de ordenamiento por selección, pero difieren en aspectos clave, el algoritmo de ordenamiento por inserción es mucho mas efectivo en conjuntos de elementos parcialmente ordenados, a diferencia del algoritmo de ordenamiento por selección, demostrando una de sus debilidades, otra de las debilidades de ambos algoritmos es la forma en que crece su tiempo de ejecución, pues a mayor cantidad de elementos, el tiempo de ejecución crece de forma exponencial, a pesar de eso, el tiempo de ejecución en el algoritmo por inserción es ligeramente mejor, en cambio, los resultados obtenidos por el algoritmo de ordenamiento por mezcla son fascinantes, procesando la misma cantidad de elementos que los algoritmos anteriores en fracciones de segundos, este algoritmo fue problemático de medir, pues al dar resultados tan pequeños, a la hora de compararle con resultados de los demás algoritmos parece casi una linea recta en la figura 4, a pesar de ser un algoritmo eficiente, tiene un problema que es muy dependiente de su implementación, siendo este la cantidad de memoria extra necesaria por cada arreglo extra de elementos, aunque en la implementación utilizada para este experimento en cuestión, soluciona este problema haciendo uso de indices extra en la división del arreglo inicial.

Finalmente, se puede concluir que la implementación de estos algoritmos han demostrado como existen algoritmos más potentes que otros para realizar la misma tarea, en algunos casos haciendo sacrificios de memoria, como ocurre con el algoritmo de ordenamiento por mezcla, o simplemente mejorando la implementación de otro algoritmo, como ocurre con el ordenamiento por inserción, y es importante conocer los beneficios y debilidades de cada algoritmo para saber cual de estos nos puede ser útil en una situación determinada.

V. ANEXOS

Figura 5. Relaciones tiempo-tamaño *Insertion-Sort* - Tabla

Relación tiempo-tamaño *Insertion-Sort*						
Elementos	Caso 1	Caso 2	Caso 3	Caso 4	Caso 5	Promedio
200,000	33.3775	33.2502	33.2417	33.2961	33.2953	33,29216
400,000	133.262	133.651	133.241	133.512	133.032	133,3396
800,000	533.791	532.604	532.674	533.497	533.648	533,2428
1,200,000	1202.06	1202.59	1203.33	1201.92	1203.84	1202,74

Figura 6. Relaciones tiempo-tamaño *Selection-Sort* - Tabla

Relación tiempo-tamaño *Selection-Sort*						
Elementos	Caso 1	Caso 2	Caso 3	Caso 4	Caso 5	Promedio
200,000	53.6833	53.6526	53.464	53.4723	53.6212	53,57868
400,000	214.125	214.547	214.276	214.089	213.997	214,2068
800,000	858.241	858.343	860.513	858.499	858.792	858,8776
1,200,000	1984.89	1956.05	1955.1	1959.71	1960.8	1962,51

Figura 7. Relaciones tiempo-tamaño *Merge-Sort* - Tabla

Relación tiempo-tamaño *Merge-Sort*						
Elementos	Caso 1	Caso 2	Caso 3	Caso 4	Caso 5	Promedio
200,000	0.050589	0.05039	0.050293	0.050622	0.050291	0,050437
400,000	0.109111	0.105656	0.105492	0.105874	0.106049	0,1064364
800,000	0.219725	0.219248	0.219074	0.21972	0.219551	0,2194636
1,200,000	0.337155	0.358163	0.337052	0.336481	0.336274	0,341025