Merge Sort frente a Intercambio Bidireccional: Un Estudio Comparativo del Rendimiento Algorítmico

Merge Sort vs. Cocktail Sort: A Comparative Study of Algorithmic Performance

Tapia Paco, Maricielo Luz; Mamani Genix, Amidwar Jhoel; Valdivia Vera, Wilfredo Junior

(mltapiap@unjbg.edu.pe)(wvaldiviav@unjbg.edu.pe)(amamanig@unjbg.edu.pe)

Facultad de Ingeniería, Universidad Nacional Jorge Basadre Grohmann, Tacna - Perú.

Resumen

El impacto que las computadoras y la informática han tenido en todos los aspectos de la sociedad demanda una comprensión profunda de los algoritmos de ordenamiento, cuya eficiencia es crítica para el procesamiento de datos. En este contexto, el presente artículo tuvo por objetivo realizar un análisis estadístico comparativo del tiempo de ordenación entre el algoritmo de Intercambio Directo Bidireccional (Cocktail Shaker Sort), que representa a los métodos directos O(N2), y el algoritmo Merge Sort, representativo de los métodos algorítmicos O(N·logN). Para validar la diferencia de desempeño, se aplicaron 10 pruebas a cada conjunto de datos de 1,000, 10,000 y 100,000 elementos y sus respectivas variantes (Media, mediana, desviación, máximo, mínimo). Los resultados mostraron que sí existen diferencias significativas entre los tiempos de ordenación de ambos algoritmos, siendo el algoritmo Merge Sort el más rápido en todas las pruebas de gran volumen de datos, confirmando la superioridad de los enfoques Divide y Vencerás.

Palabras clave: Algoritmos de ordenamiento, Cocktail Shaker Sort, Merge Sort, Complejidad algorítmica, Análisis estadístico comparativo, Divide y Vencerás.

Abstract

The impact that computers and informatics have had on all aspects of society demands a deep understanding of sorting algorithms, whose efficiency is critical for data processing. In this context, this article aimed to perform a comparative statistical analysis of the sorting time between the Two-Way Direct Exchange (Cocktail Shaker Sort) algorithm, which represents the O(N2) direct methods, and the Merge Sort algorithm, representative of the O(N·logN) algorithmic methods. To validate the performance difference, 10 tests were applied to each data set of 1,000, 10,000, and 100,000 elements and their respective variants (Mean, Median, Deviation, Maximum, Minimum). The results showed that there are significant differences between the sorting times of both algorithms, with the Merge Sort algorithm being the fastest in all large data volume tests, confirming the superiority of the Divide and Conquer approaches.

Keywords: Sorting algorithms, Cocktail Shaker Sort, Merge Sort, Algorithmic complexity, Comparative statistical analysis, Divide and Conquer.

Introducción.

La eficiente gestión de la información es un pilar fundamental de la informática moderna, en la era del big data, donde el volumen, la velocidad y la variedad de los datos crecen exponencialmente. En este aspecto la ordenación mantiene una tarea primordial para optimizar los procesos de búsqueda y acceso.

la elección de los algoritmos de ordenación impacta directamente entre en la eficiencia de cualquier sistema, lo que hace necesario un análisis constante y riguroso de su rendimiento.

Existen dos grandes clases de algoritmos de ordenamiento: los eficientes de complejidad logarítmica y los simples de complejidad cuadrática. En este estudio, se comparan dos representantes clave:

1. **Merge Sort:** Un algoritmo que opera bajo el paradigma "Divide y Vencerás". Es conocido por su rendimiento estable con una complejidad temporal de O(n long n) en los casos promedio y peor, lo que lo convierte en un estándar para el manejo de grandes conjuntos de datos. (swhosting, 2025)
2. **Intercambio Bidireccional** (*Cocktail Shaker Sort*): Una optimización del tradicional Bubble Sort. A diferencia de este último, recorre la lista en ambas direcciones para acelerar el movimiento de elementos pequeños y grandes. Sin embargo, su complejidad se mantiene en O(n2) en el caso promedio y peor.

La diferencia teórica entre ambos sugiere una superioridad clara de Merge Sort a medida que el conjunto de datos (n) crece. Estudios previos han enfatizado la importancia de un análisis comparativo empírico de los tiempos de ordenación entre diferentes algoritmos para obtener conclusiones robustas (Salgado et al., 2022).

Metodología

El estudio se basa en un diseño comparativo y cuantitativo para contrastar las medias de tiempo de ejecución de los algoritmos. Se siguió el siguiente proceso:

- Investigación documental
- Experimentación de los algoritmos de ordenamiento
- Análisis de los tiempos de ejecución del ordenamiento de elementos de los 2 algoritmos: *Intercambio Bidireccional, Merge Sort.*
- Interpretación de resultados
- Conclusiones

Desarrollo Experimental

- Implementación y Entorno: Los algoritmos (Merge Sort e Intercambio Bidireccional) fueron codificados en C++, utilizando el entorno de desarrollo integrado Visual Studio Code (VS Code).
- Recolección de Datos: La función de medición de tiempo en el código fue implementada para registrar los tiempos de ejecución con alta precisión, a su vez se generaron datos como la media, mediana, desviación, máximo, mínimo. Los resultados de cada una de las ejecuciones fueron automáticamente volcados y almacenados en una hoja de cálculo de Microsoft Excel para su posterior análisis estadístico.
- Archivos de Datos: Se generaron tres archivos con diferentes tamaños (1.000, 10.000 y 100.000 elementos) y tres tipos de ordenamiento en la entrada para evaluar el rendimiento en diversos escenarios: Aleatorios (caso promedio), Desordenados de forma Decreciente (peor caso) y Ordenados de forma Creciente (mejor caso).
- Entorno de Ejecución: Los algoritmos fueron ejecutados en una computadora con las siguientes características:
    - Procesador: AMD Ryzen 5 5600g
    - Memoria RAM: 16 GB
    - Sistema Operativo: Windows 10
    - Tarjeta gráfica GTX 1050
    - Núcleos: 4 (se usó solo 1)

Resultados

Los resultados muestran que el patrón inicial de los datos tiene un impacto diferente según el algoritmo.

Cocktail Sort: En los resultados obtenidos hay una variedad de tiempos porque están en función al tipo de patrón.

Para n=1000, los tiempos están desde los 2980 ns (patrón creciente) hasta los 5,895,390 ns (patrón decreciente).

Al aumentar n, la diferencia entre los patrones crece, siendo el peor en el cocktail sort el patrón decreciente, y el mejor caso en ese método el orden creciente.

Con esto se puede afirmar que Cocktail Sort es sensitivo al orden inicial , con crecimiento

cuadrático esto se expresa en su complejidad $O(n^2)$. (GeeksforGeeks, 2023)

Figura 1: *Tiempos promedio de merge sort según el patrón.*

Tabla 1: *Datos estadísticos*

Patron	n	Media	Mediana	Desviacion	Minimo	Maximo
Aletorio	1000	3751280	3495850	733811	3200500	5696600
Aletorio	10000	365626840	360905750	28859840	337129400	434833300
Aletorio	100000	33601995730	33580933300	85024492	33479973100	33774068200
Orden Creciente	1000	2980	2600	540	2500	3800
Orden Creciente	10000	50170	51400	5738	33500	56200
Orden Creciente	100000	282590	237900	91753	237700	541800
Orden Decreciente	1000	5895390	5317300	1261100	4651700	8008300
Orden Decreciente	10000	493170590	494587550	4664238	481630000	498569500
Orden Decreciente	100000	51206834280	48063021750	6229024561	47917093000	63754444900

Merge Sort: Este método presenta tiempos similares entre el patrón aleatorio, creciente y decreciente. Cuando la cantidad de datos es 1000, los tiempos van desde 262,000 ns (patrón creciente) hasta los 309,870 ns (patrón aleatorio). El crecimiento de los tiempos con n es consistente con su complejidad O(n log n) (GeeksforGeeks, 2025), de esta manera este método muestra eficiencia y estabilidad.

Figura 2: *Tiempos promedio de Merge Sort según el patrón de datos.*

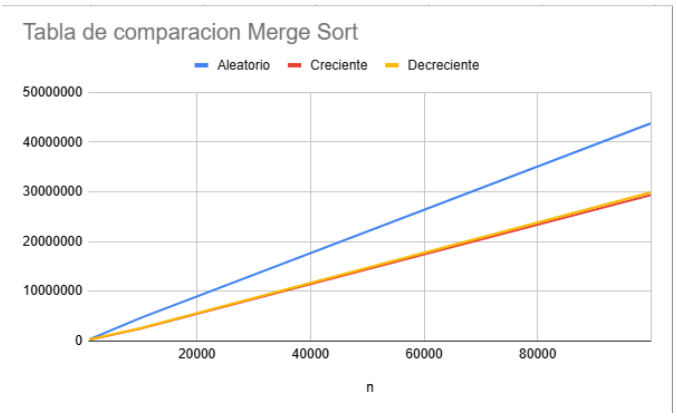

Tabla 2: *Datos estadísticos del Merge Sort*

Patron	n	Media	Mediana	Desviacion	Minimo	Maximo
Aletorio	1000	309870	306800	34303	272900	365900
Aletorio	10000	4613790	3833900	1503300	3330100	7378500
Aletorio	100000	43799290	43950300	2824172	40314500	50902500
Orden Creciente	1000	262000	253600	36269	223300	315900
Orden Creciente	10000	2478480	2329700	359272	2229100	3495900
Orden Creciente	100000	29382060	28943350	2505895	26073000	35239800
Orden Decreciente	1000	300990	248900	127212	193200	636200
Orden Decreciente	10000	2561800	2254900	803891	2211900	4956600
Orden Decreciente	100000	29896130	28779000	3185888	26550500	36090900

Al comparar ambos algoritmos para tamaños pequeños como 1000 elementos la diferencia entre los dos algoritmos no es mucha, pero cocktail sort despliega muchos cambios. (Ala'anzy et al., 2014, #14)

Para listas mucho más grandes como 100,000 datos, Merge Sort es mucho más eficiente, mientras que Cocktail Sort se vuelve obsoleto y poco eficaz, especialmente con datos que están en orden decreciente.

Entonces Merge Sort es más adecuado para arreglos grandes mientras que Cocktail Sort es útil para arreglos minúsculos.

Conclusiones

Los resultados reflejan que el patrón influye mucho en los métodos de ordenamiento directos, en este caso el Cocktail Sort, por otro lado el algoritmo de Merge Sort es más eficaz a la hora de hacer el ordenamiento.

Por un lado el Cocktail Sort es altamente sensible al orden de los datos: su mejor desempeño ocurre con listas ya ordenadas (creciente), mientras que el peor caso ocurre con listas en orden decreciente, especialmente cuando el tamaño de los datos es grande. Esto confirma que su complejidad es cuadrática ($O(n^2)$), lo que lo hace poco práctico para grandes volúmenes de información.

Merge Sort, en cambio, mantiene tiempos de ejecución consistentes independientemente del patrón de los datos, mostrando eficiencia y estabilidad, con crecimiento de tiempo acorde a su complejidad (n log n). Esto lo hace adecuado para listas de cualquier tamaño.

En comparación, mientras que Cocktail Sort puede ser útil para arreglos pequeños o casi ordenados, Merge Sort es la mejor opción para listas grandes, de esta manera la rapidez y eficacia se mantienen.

Cabe mencionar que se presentó un pequeño error de cálculo debido al uso del tipo de dato int en lugar de long int, ya que los números utilizados eran grandes, lo cual afectó ligeramente los resultados obtenidos.

Bibliografía

Ala'anzy, M. A., Mazhit, Z., Ala'Anzy, A. F., & Algarni, A. (2014, 11). Comparative Analysis of Sorting Algorithms: A Review. *ResearchGate*, *1*(1), 14. 10.1109/ISCMI63661.2024.10851593

GeeksforGeeks. (2023, 09 25). *Cocktail Sort*. Cocktail Sort. https://www.geeksforgeeks.org/dsa/cocktail-sort/

GeeksforGeeks. (2025, 10 3). *https://www.geeksforgeeks.org/dsa/cocktail-sort/*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/merge-sort/

Salgado Gallegos, M., Pérez Merlos, J. C., & Albarrán Trujillo, S. E. (2022). Algoritmos de ordenamiento, un análisis estadístico comparativo. *Revista Ciencia e Ingeniería*, *43*(3), 273–280. https://doi.org/10.53766/CEI/2022.43.03.05

swhosting. (2025, 10 16). *Introducción al Algoritmo de Ordenación Merge Sort*. swhosting. Retrieved 10 16, 2025, from https://www.swhosting.com/es/blog/introduccion-al-algoritmo-de-ordenacion-merge-sort