UNIVERSIDAD NACIONAL JORGE BASADRE GROHMANN

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA EN INFORMÁTICA Y SISTEMAS

“Comparación entre cycle sort y intercambio directo con señal”

DOCENTE: Ing. Hugo Manuel Barraza Vizcarra

CURSO: Programación Avanzada

CICLO: 2do

FACULTAD: Ingeniería

TURNO: Tarde

ESCUELA: ESIS

INTEGRANTES:

- Franklin Jhonatan Mamani Apaza 2023-119030
- Josue Fernando Mamani Lima 2025-119055
- Josue Santiago Maquera Pilco 2025-119005

TACNA - PERÚ

2025

Comparación entre cycle sort y intercambio directo con señal

INFORMACIÓN DE AUTORES

nombre: josue fernando mamani lima **- universidad:** Universidad Nacional Jorge Basadre Grohmann - **ciudad**: Tacna - **país:** Perú - **correo institucional:** jfmamanil@unjbg.edu.pe

nombre: Franklin Jhonatan Mamani Apaza **- Universidad:** Universidad Nacional Jorge Basadre Grohmann - **Ciudad**: Tacna - **País**: Perú - **Correo institucional**: fjmamania@gmail.com

- **nombre:** Josue Santiago Maquera Pilco **- universidad:** Universidad Nacional Jorge Basadre Grohmann - **ciudad**: Tacna - **país:** Perú - **correo institucional:** jmaquerap@unjbg.edu.pe

RESUMEN AND ABSTRACT

El presente artículo analiza comparativamente los algoritmos Cycle Sort y el método de intercambio directo con señal, dos enfoques distintos para el ordenamiento de datos. Se examinan sus características, complejidad temporal, estabilidad, cantidad de intercambios y eficiencia en diferentes contextos. El Cycle Sort destaca por minimizar las escrituras en memoria, siendo útil en sistemas con recursos limitados o dispositivos de almacenamiento con desgaste. En cambio, el intercambio directo con señal, una mejora del *Bubble Sort*, se caracteriza por su simplicidad y por detectar rápidamente si una lista ya está ordenada. Los resultados de la comparación muestran que, aunque ambos algoritmos tienen complejidad $O(n^2)$, difieren significativamente en su comportamiento práctico según el tipo de datos y la cantidad de elementos a ordenar.

This article presents a comparative analysis between Cycle Sort and the Direct Exchange Method with Signal, two distinct approaches to data sorting. Their characteristics, time complexity, stability, number of swaps, and efficiency in various scenarios are examined. Cycle Sort stands out for minimizing write operations, making it suitable for systems with limited resources or storage devices

prone to wear. On the other hand, the Direct Exchange Method with Signal, an enhanced version of *Bubble Sort*, is noted for its simplicity and its ability to quickly detect when a list is already sorted. The comparison results indicate that, although both algorithms share an $O(n^2)$ complexity, their practical performance varies significantly depending on the data type and dataset size.

palabras claves: algoritmos de ordenamiento, eficiencia computacional, Cycle Sort, intercambio directo con señal, complejidad algorítmica.

INTRODUCCIÓN

El ordenamiento de datos es una operación fundamental en el campo de la informática, ya que influye directamente en la eficiencia de numerosos algoritmos y sistemas. A lo largo del tiempo se han desarrollado diversos métodos de ordenación que buscan optimizar el uso del tiempo y la memoria, dependiendo del contexto de aplicación. Entre ellos destacan Cycle Sort y el método de intercambio directo con señal, dos algoritmos que, aunque comparten la misma finalidad, difieren en su estructura, eficiencia y propósito.

El algoritmo Cycle Sort se caracteriza por su enfoque en reducir al mínimo el número de escrituras, lo que lo hace especialmente útil en entornos donde la durabilidad del almacenamiento es un factor crítico. En contraste, el intercambio directo con señal una variante mejorada del clásico Bubble sort introduce una bandera de control que permite detener la ejecución cuando la lista ya está ordenada, optimizando el tiempo en casos favorables.

Este estudio busca comparar ambos métodos considerando su complejidad computacional, estabilidad, eficiencia y número de operaciones, con el objetivo de analizar sus ventajas y limitaciones en distintos escenarios. De esta manera, se pretende aportar una visión más clara

sobre la elección del algoritmo de ordenamiento más adecuado según el contexto de uso y las restricciones del sistema.

Métodos

Cycle Sort

El Cycle Sort es un algoritmo que destaca por su capacidad de minimizar el número de escrituras en el arreglo. Según Haddon (1990), es posible descomponer una permutación del arreglo en ciclos, y rotar cada ciclo poniendo cada elemento directamente en su posición correcta, con un único "write" por elemento que necesita moverse ("either written zero times, if it's already in its correct position, or written one time to its correct position").

En su versión general, para cada posición de inicio cycle_start, el algoritmo cuenta cuántos elementos son menores que el valor actual para determinar su posición destino. Si el elemento no está ya en su lugar, se intercambia directamente al lugar correcto, y así sucesivamente se continúa hasta cerrar el ciclo. Esto asegura que cada elemento se mueve lo mínimo necesario.

Sin embargo, la complejidad temporal del algoritmo sigue siendo $O(n^2)$ en casos promedio y peores, debido al conteo para determinar la posición correcta de cada elemento. Este algoritmo es "in-place" (requiere espacio adicional O(1)) y no estable (puede cambiar el orden entre elementos iguales).

Se ha observado también que en casos muy específicos, bajo ciertas condiciones de los datos (por ejemplo, cuando las claves pertenecen a un conjunto pequeño y se conoce la frecuencia de las claves), puede conseguirse un rendimiento lineal mediante variaciones del algoritmo general (denominadas general_cycle_sort y special_cycle_sort). No obstante, estas versiones especializadas no se aplican de forma general en datos arbitrarios.

Adicionalmente, se han propuesto enfoques de paralelización para Cycle Sort, utilizando paradigmas como MPI (Message Passing Interface) y CUDA para GPUs con el fin de mitigar

la penalización del tiempo en grandes volúmenes de datos (asumiendo particionamiento del problema en subciclos). En esos estudios se confirma que el algoritmo base conserva su propiedad de minimización de escrituras mientras se gana en tiempo gracias al paralelismo.

Intercambio Directo con Señal

El método de intercambio directo con señal es una mejora sobre el clásico *Bubble Sort*, mediante la introducción de una bandera (flag) que detecta si en una pasada no se realizaron intercambios, lo que permite detener el algoritmo tempranamente si el arreglo ya está ordenado (o está casi ordenado).

La versión básica de Bubble Sort consiste en iterar repetidamente sobre el arreglo, comparando pares adyacentes y permutándolos si están en desorden, "bubbleando" el elemento máximo hacia el final en cada pasada

Una posible mejora consistiria en reducir el límite del recorrido en cada iteración porque al final de cada pasada el elemento más grande "burbujeado" queda en su lugar definitivo, por ello no es necesario revisarlo en pasadas siguientes.

Con esta optimización, el algoritmo alcanza complejidad O(n) en el mejor caso (cuando el arreglo ya está ordenado y no se realiza ningún intercambio en la primera pasada). En el peor caso y caso promedio, sigue siendo $O(n^2)$.

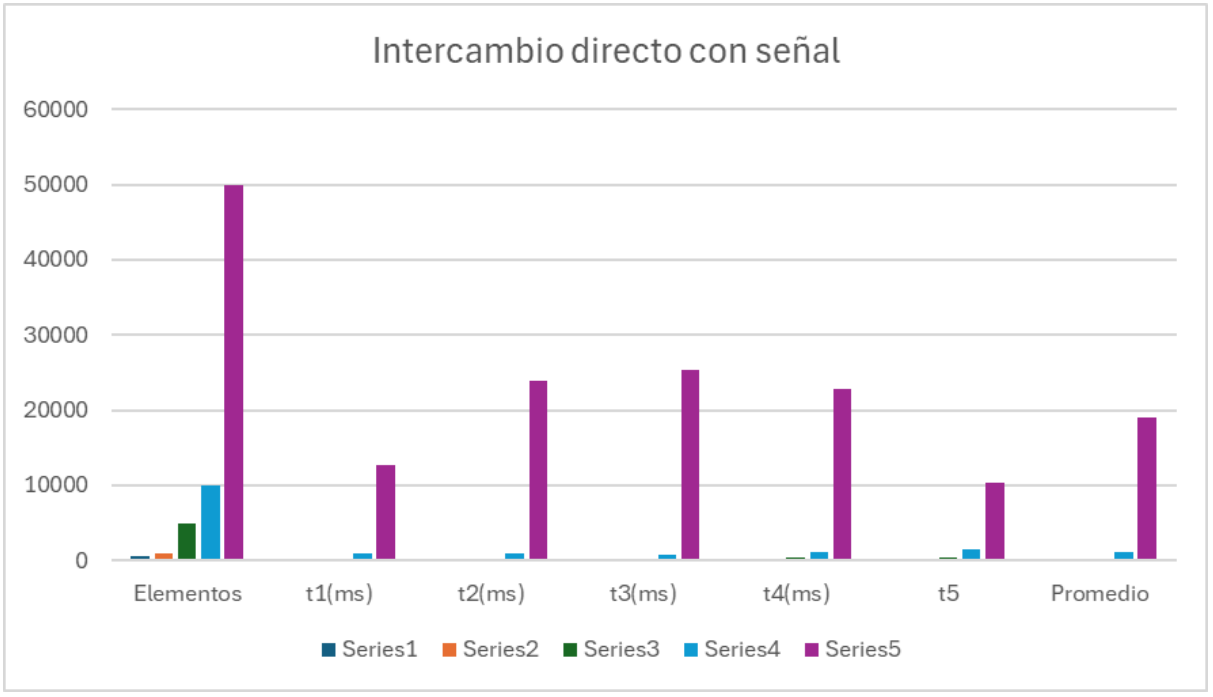
RESULTADOS

Intercambio directo con señal

Elementos	**t_1 (ms)**	**t_2 (ms)**	**t_3 (ms)**	**t_4 (ms)**	**t_5 (ms)**	**Promedio**

500	**0**	**2**	**1**	**2**	**2**	**1.4**
1000	**10**	**11**	**9**	**14**	**10**	**10.8**
5000	**300**	**231**	**190**	**379**	**376**	**295.2**
10000	**947**	**974**	**746**	**1128**	**1515**	**1062**
50000	**12668**	**23882**	**25288**	**22773**	**10373**	**18996.8**

En el siguiente cuadro se observa las medidas tomadas en diferentes cantidades de datos (500, 1000, 5000, 10000, 50000), y los tiempos tomados, que en este caso fueron 5.

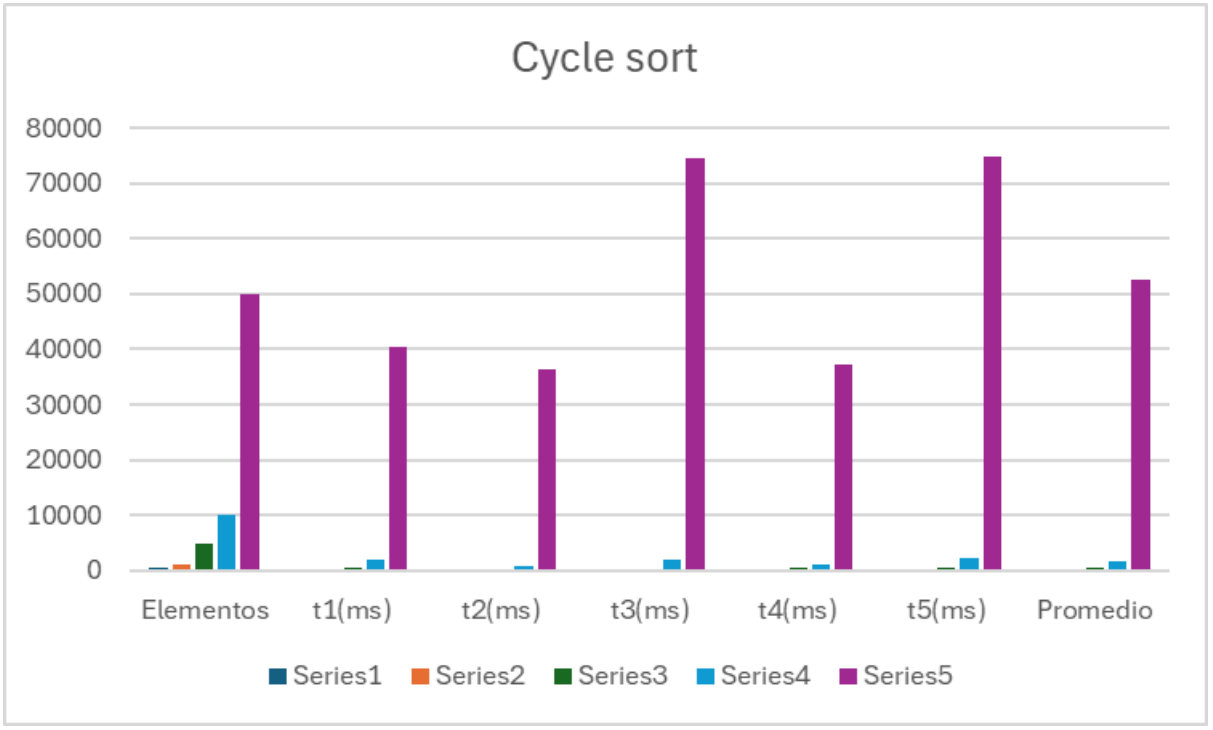


Se realizó el gráfico el cual se observa a continuación, referente a los datos obtenidos en la tabla anterior. En el gráfico la primera secuencia de barras representan la cantidad de datos, por ejemplo el morado está en 50000, que es la última cantidad de datos dado, y después en las siguientes secuencia de barras se muestra el tiempo en que demora ordenar los datos el algoritmo, observaremos que llega a tener un crecimiento acelerado mientras más grande es el arreglo

Cycle sort

Elementos	t_1 (ms)	t_2 (ms)	t_3 (ms)	t_4 (ms)	t_5 (ms)	Promedio
500	9	10	4	2	6	6.2
1000	19	22	24	22	13	20
5000	504	285	341	516	654	460
10000	2000	925	2140	1256	2197	1703.6
50000	40289	36325	74614	37209	74804	52648.2

El siguiente gráfico sigue la secuencia del anterior mostrado, la diferencia sería que los datos obtenidos son del método de ordenación cycle sort



A continuación se mostrará un gráfico con los datos implementados y el tiempo que demora cada tamaño de datos, el siguiente gráfico es lo mismo que el anterior mostrado solo que del

método cycle sort, en la cual apreciamos un mayor incremento del tiempo comparado con el anterior método.

DISCUSIONES

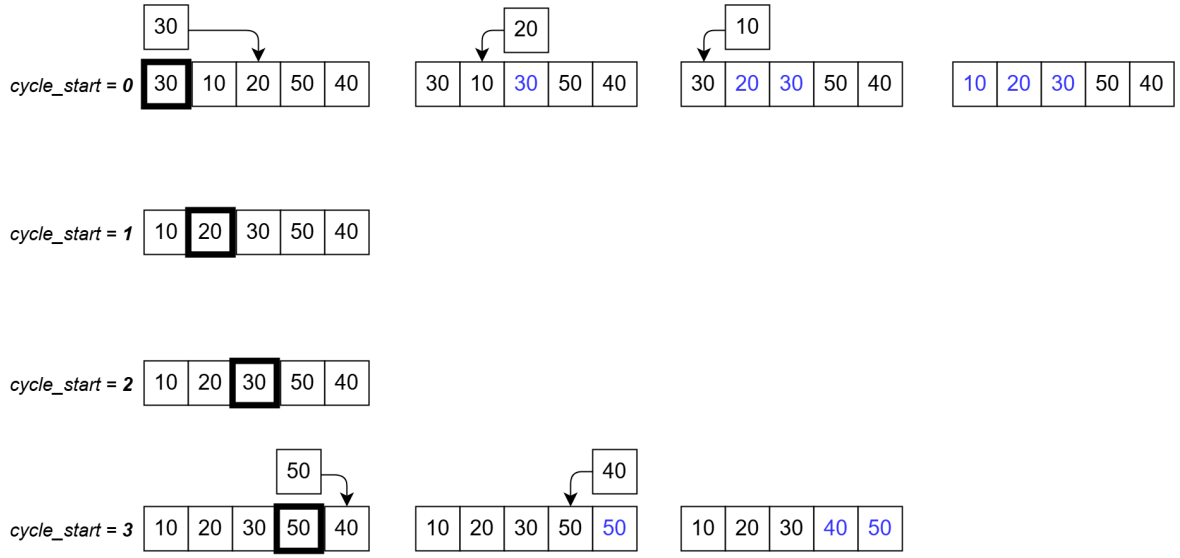
Como se habrá podido observar, podemos ver claramente en los cuadros implementados anteriormente como el método de intercambio directo con señal tiende a tener un recorrido de forma exponencial, ya que mientras más grande sea el arreglo, mucho más tiempo demora en ordenarlo, osea que su complejidad es cuadrática($O(n^2)$), esto sería tanto para el peor caso como caso promedio, pero para el mejor de los casos su complejidad sería lineal ($O(n)$). por que al estar los elementos ya ordenados o casi ordenados solo realizaría unos cuantos recorridos y al no encontrar más elementos daría por concluida la función. . En cambio en método de cycle sort es muy diferente, como se podrá ver en la segunda gráfica , este también tiende a tener un recorrido exponencial, por lo que se podría decir que su complejidad es cuadrática($O(n^2)$.) tanto para el peor, promedio y mejor de los casos, debido a que el código fuente del algoritmo si o si va a realizar recorridos para ver si cada elemento se encuentra en su posición correcta. Gracias a ello podemos darnos cuenta cual de los dos métodos es más eficiente, en este caso para ordenamiento convendría utilizar el método de intercambio directo por que si el arreglo esta medio ordenado para a realizar menos iteraciones.

AGRADECIMIENTOS

En primer lugar agradecemos al ingeniero Hugo Manuel Barraza y a la ingeniera Calisaya por leer y escuchar nuestro informe de investigación, también por responder a nuestras dudas con respecto a los algoritmos de ordenación, ayudarnos a completar nuestro proyecto de la mejor forma y compartir material que resultó ser un punto clave a la hora de realizar las comparaciones entre algoritmos, también agradecemos a nuestros compañeros y por supuesto a nuestra delegada de curso.

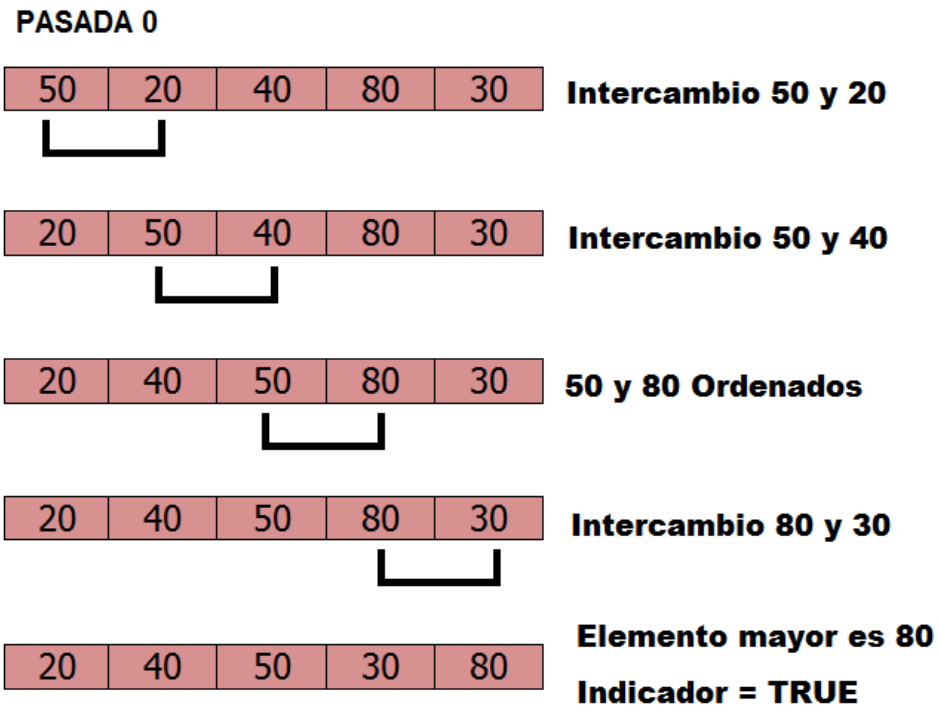
TABLAS Y FIGURAS

Figura 1



fuente: https://www.baeldung.com/cs/cycle-sort-algorithm

Figura 2



fuente: https://ordenacionsenal.blogspot.com/2013/04/descripcion.html

BIBLIOGRAFÍAS

Cycle Sort – una forma de minimizar escrituras *ResearchGate: "Cycle-Sort: A Linear Sorting Method"*

https://www.researchgate.net/publication/220459798_Cycle-Sort_A_Linear_Sorting_Method

Cycle Sort – paralelización (MPI / CUDA) *"Parallelization of Cycle Sort Algorithm" (PDF)*

https://thegrenze.com/pages/servej.php?association=GRENZE&fn=733.pdf&id=3465&issue=2&journal=GIJET&name=Parallelization+of+Cycle+Sort+Algorithm&volume=10&year=2024

Bubble Sort – versión con bandera / optimización temprana *Artículo "Bubble Sort: An Archaeological Algorithmic Analysis"*

https://users.cs.duke.edu/~ola/papers/bubble.pdf

Optimización del Bubble Sort – artículo "Unveiling the Optimized Bubble Sort Algorithm"

https://www.athensjournals.gr/reviews/2023-5588-AJTE.pdf

Análisis de optimización del Bubble Sort *"Analysis on Bubble Sort Algorithm Optimization"*

https://www.researchgate.net/publication/251971353_Analysis_on_Bubble_Sort_Algorithm_Optimization