

Proyecto Realizado por Pau Ripoll

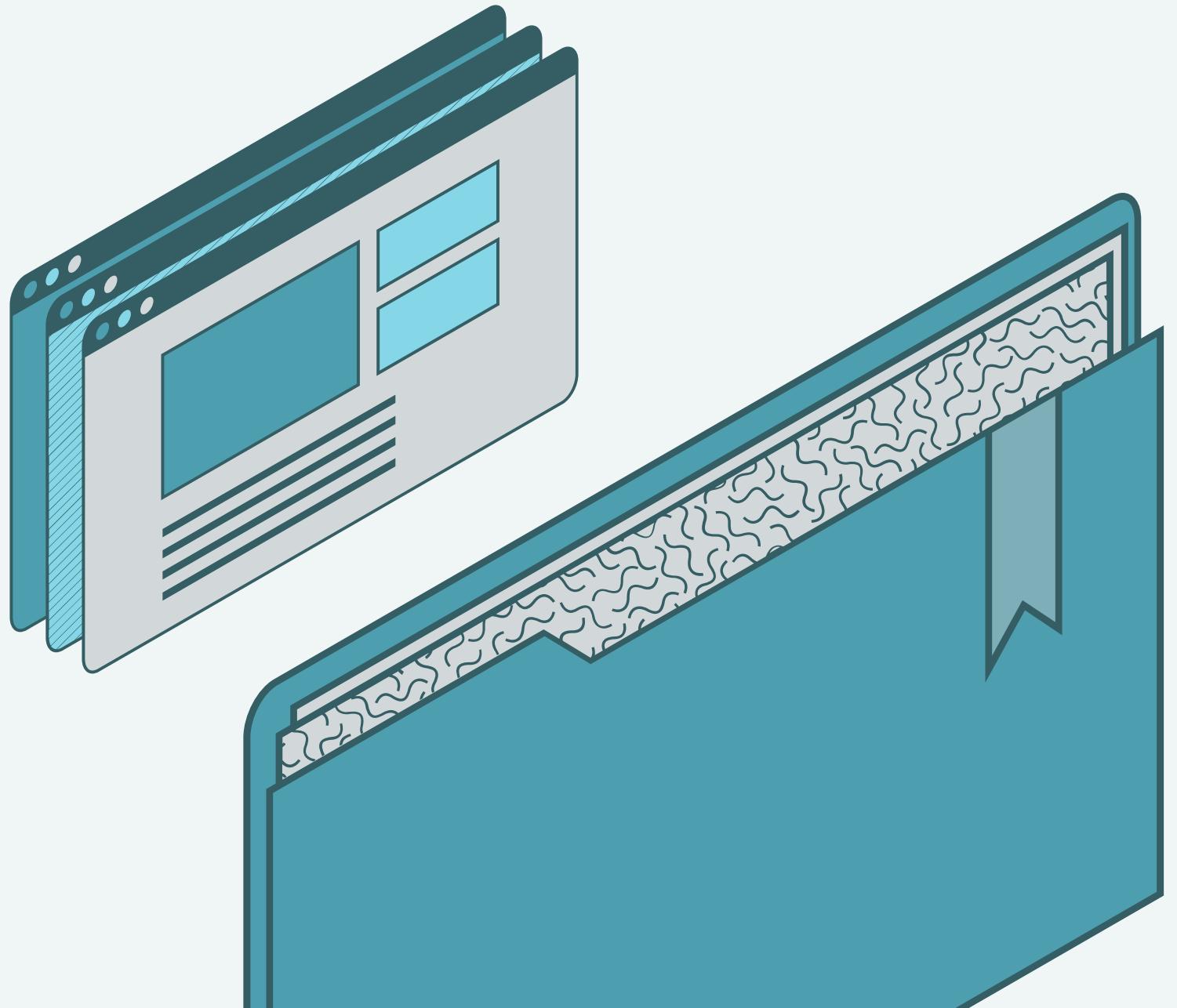
# ANÁLISIS COMPARATIVO ENTRE BURBUJA SEÑAL - RADIXSORT

Integrantes:

Jhosep Rodrigo Arocutipa Mamani 2024-119027

Alexder Bernabé Flores Cutipa 2024- 119020

# íNDIC



---

**01. Introducción**

---

**02. Algoritmos**

---

**03. comparaciones**

---

**04. Resultados**

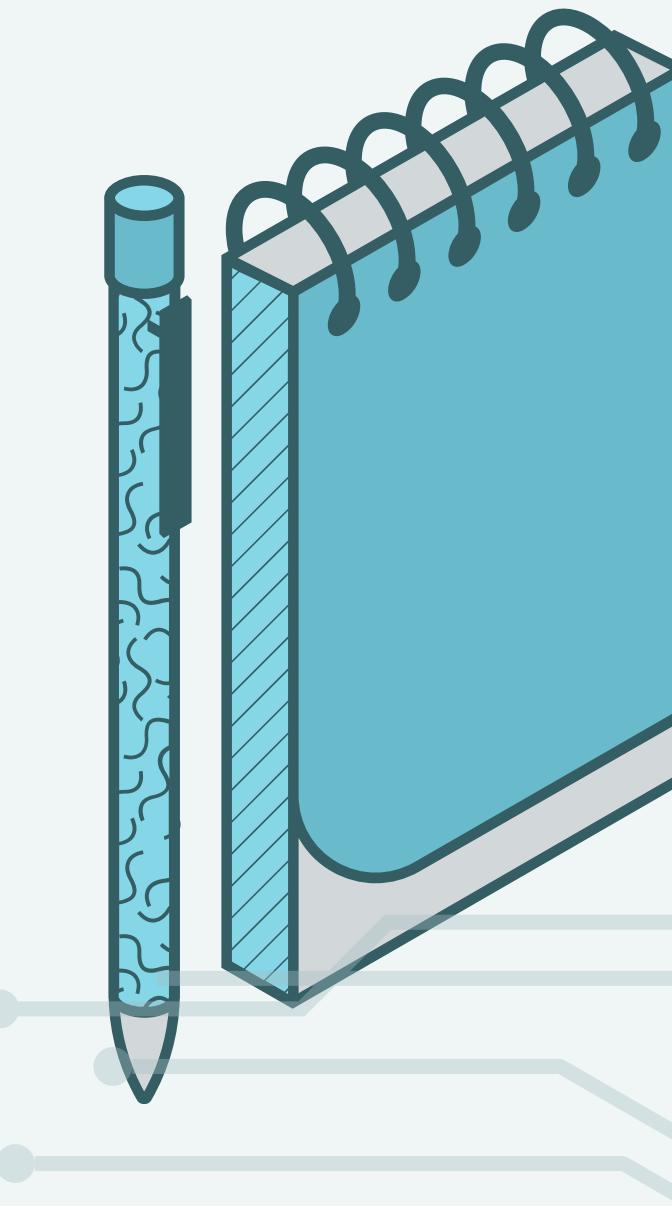
---

**05. Conclusiones**

---

# INTRODUCCIÓN

En el campo del análisis de algoritmos, la comparación de métodos de ordenamiento constituye una parte fundamental para evaluar la eficiencia computacional de los procesos que manipulan grandes volúmenes de datos. Cada algoritmo presenta un comportamiento distinto frente al aumento del tamaño de entrada ( $n$ ), lo que se refleja en su complejidad temporal, espacial y costo computacional real.



# MÉTODO DE ORDENAMIENTO DIRECTO POR SEÑAL O BURBUJA SEÑAL

Intercambio directo con señal es una variante del método de ordenamiento por intercambio directo que emplea una señal (o bandera) para marcar el último punto en el que se efectuó un intercambio. Esta estrategia permite al algoritmo detenerse anticipadamente cuando, en una pasada completa, no se produce ningún intercambio, indicando que el conjunto ya está ordenado

PASADA 0

50 | 20 | 40 | 80 | 30

Intercambio 50 y 20

20 | 50 | 40 | 80 | 30

Intercambio 50 y 40

20 | 40 | 50 | 80 | 30

50 y 80 Ordenados

20 | 40 | 50 | 80 | 30

Intercambio 80 y 30

20 | 40 | 50 | 30 | 80

Elemento mayor es 80

Indicador = TRUE

## Pasada 1



20 Y 40 Ordenados



40 y 50 Ordenados



Se Intercambian 50 y 30



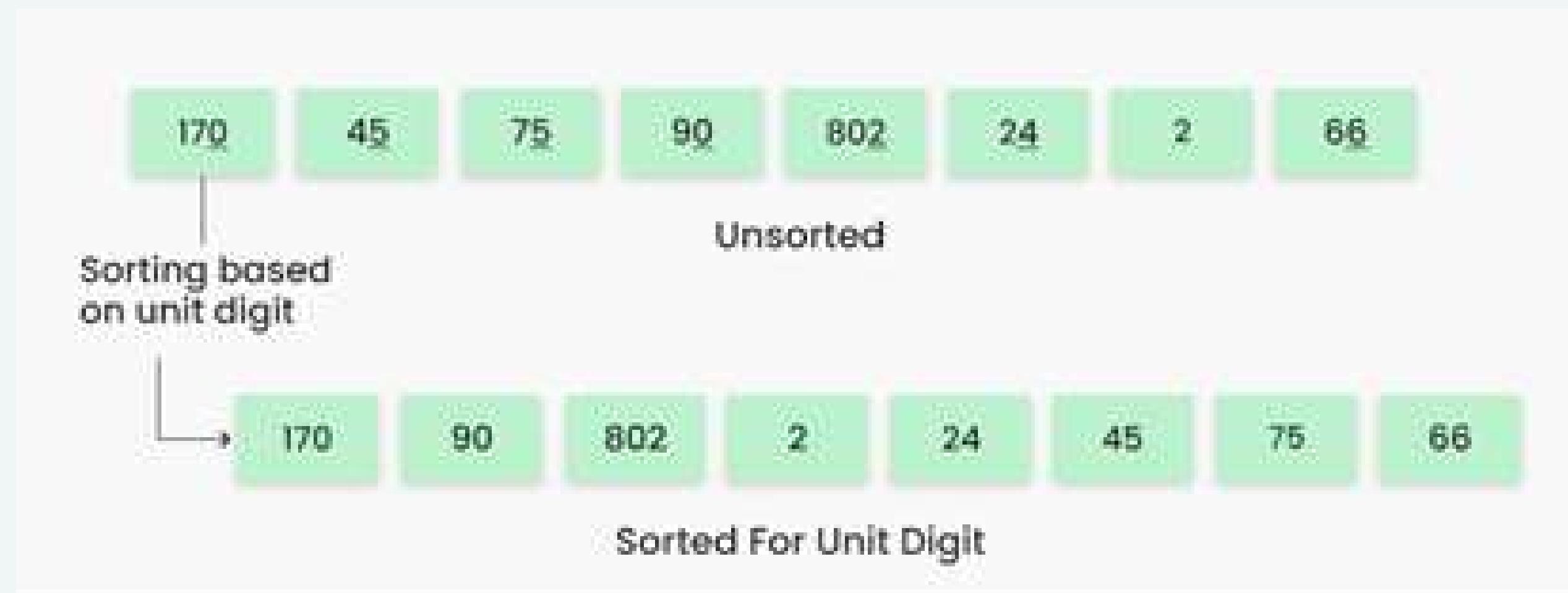
50 y 80 Elementos  
mayores y ordenados

Indicador = TRUE

## MÉTODO DE ORDENAMIENTO RADIXSORT

un método de ordenamiento en el que se ordena los elementos de una lista procesando sus dígitos de forma individual. Funciona clasificando los elementos según los valores de los dígitos de menor a mayor, primero en función del dígito menos significativo, luego en función del siguiente dígito más significativo, y así sucesivamente, hasta que todos los dígitos hayan sido considerados.

**Paso 1:** Encuentra el elemento más grande, que es 802.  
Tiene tres dígitos, por lo que lo iteraremos tres veces.  
**Paso 2:** Ordena los elementos según los dígitos de la unidad  
(X=0).



Sorting based  
on 10's digit

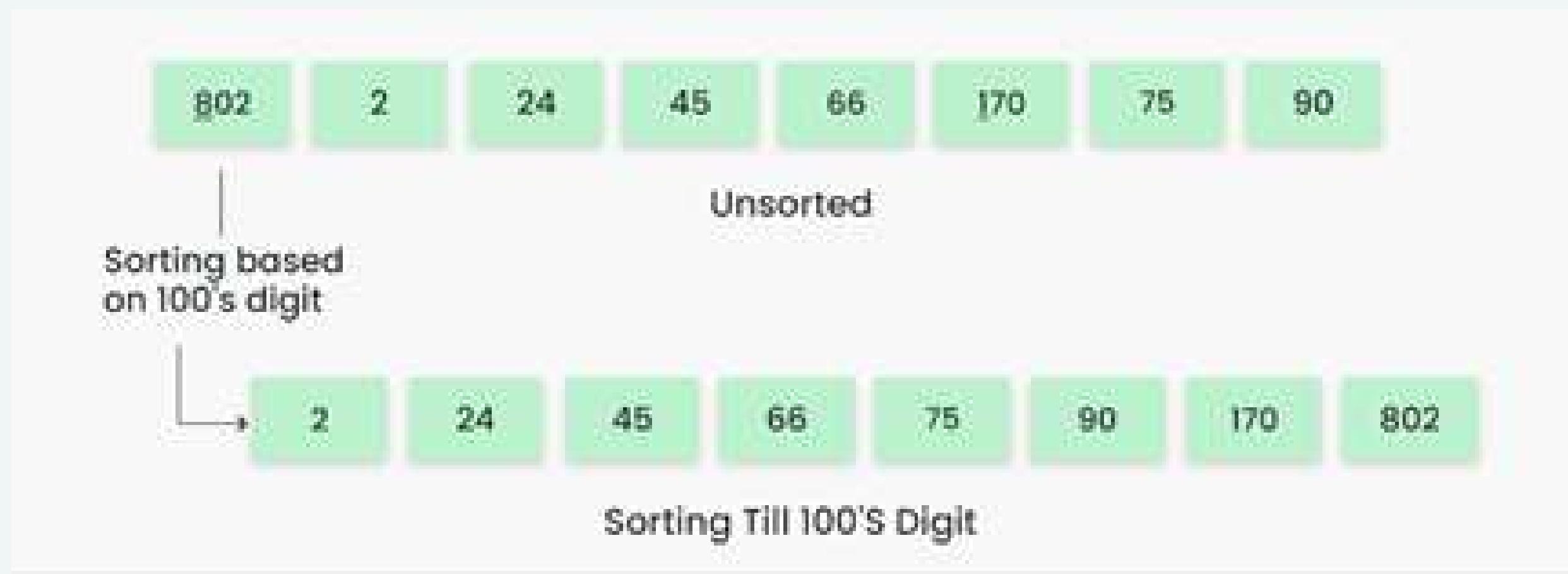
170    90    802    2    24    45    75    66

Unsorted

802    2    24    45    66    170    75    90

Sorted Till 10's Digit

**PASO 3: ORDENA LOS ELEMENTOS SEGÚN LOS DÍGITOS DE LAS DECENAS.**



**PASO 4: ORDENA LOS ELEMENTOS SEGÚN LOS DÍGITOS DE LAS CENTENAS.**

Array after performing **Radix Sort** for all digits

2

24

45

66

75

90

170

802

**PASO 5: LA MATRIZ AHORA ESTÁ ORDENADA EN ORDEN ASCENDENTE**

# COMPLEJIDAD COMPUTACIONAL

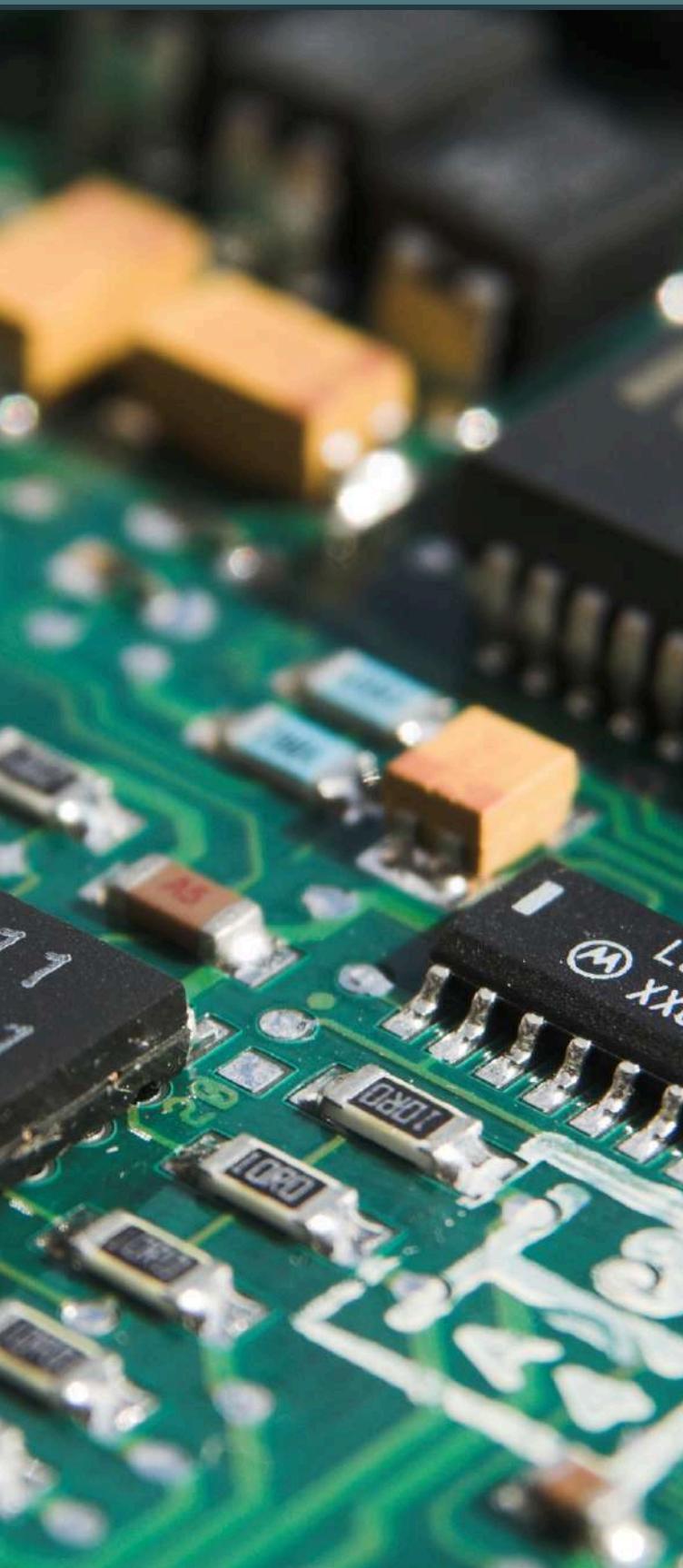
- La complejidad computacional es un área fundamental dentro de la informática que estudia los recursos necesarios que se requieren para resolver problemas computacionales. Específicamente, se centra en dos aspectos clave: el tiempo y el espacio. Por tiempo, nos referimos a la cantidad de pasos necesarios para llegar a una solución, mientras que el espacio se refiere a la cantidad de memoria que un algoritmo necesita durante su ejecución.



**Eficiencia:** En un mundo cada vez más orientado a los datos, elegir algoritmos eficientes es crucial para procesar información rápidamente.

**Limitaciones:** Al comprender la complejidad de un problema, los investigadores pueden saber hasta dónde llegan sus soluciones y si son viables.

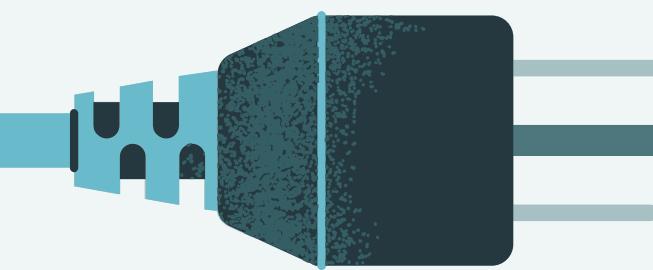
**Desarrollo de nuevas tecnologías:** La complejidad computacional impulsa la innovación, guiando la creación de algoritmos de búsqueda en motores como Google o sistemas de inteligencia artificial.

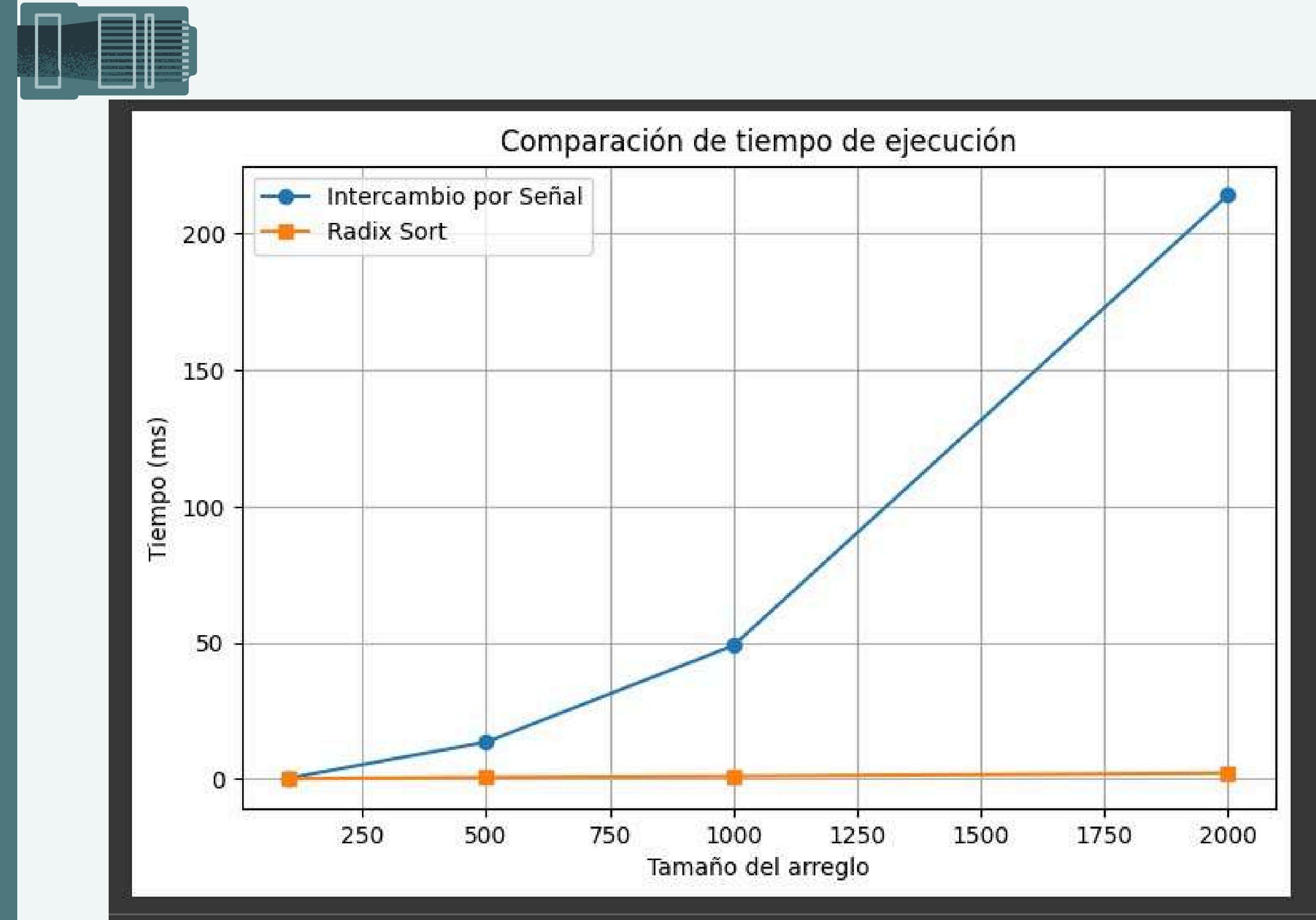
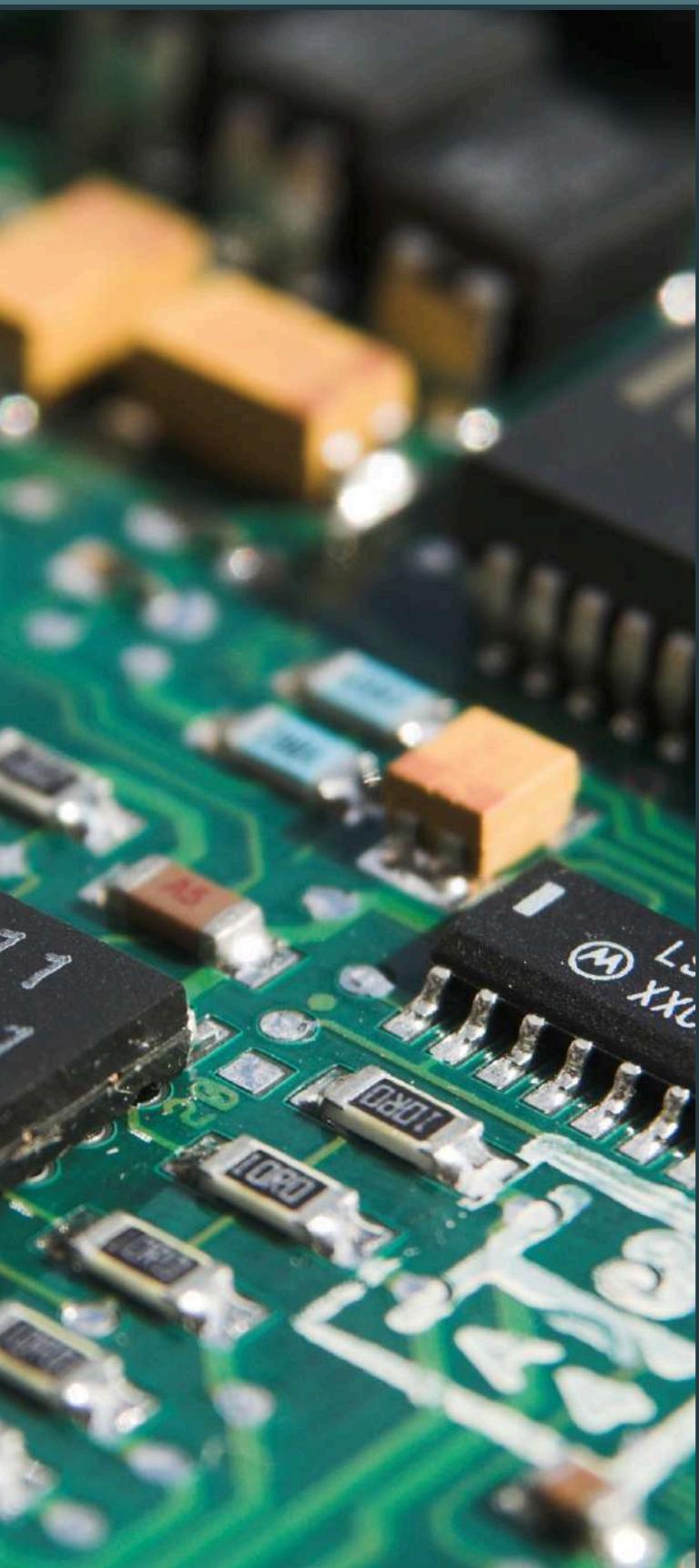


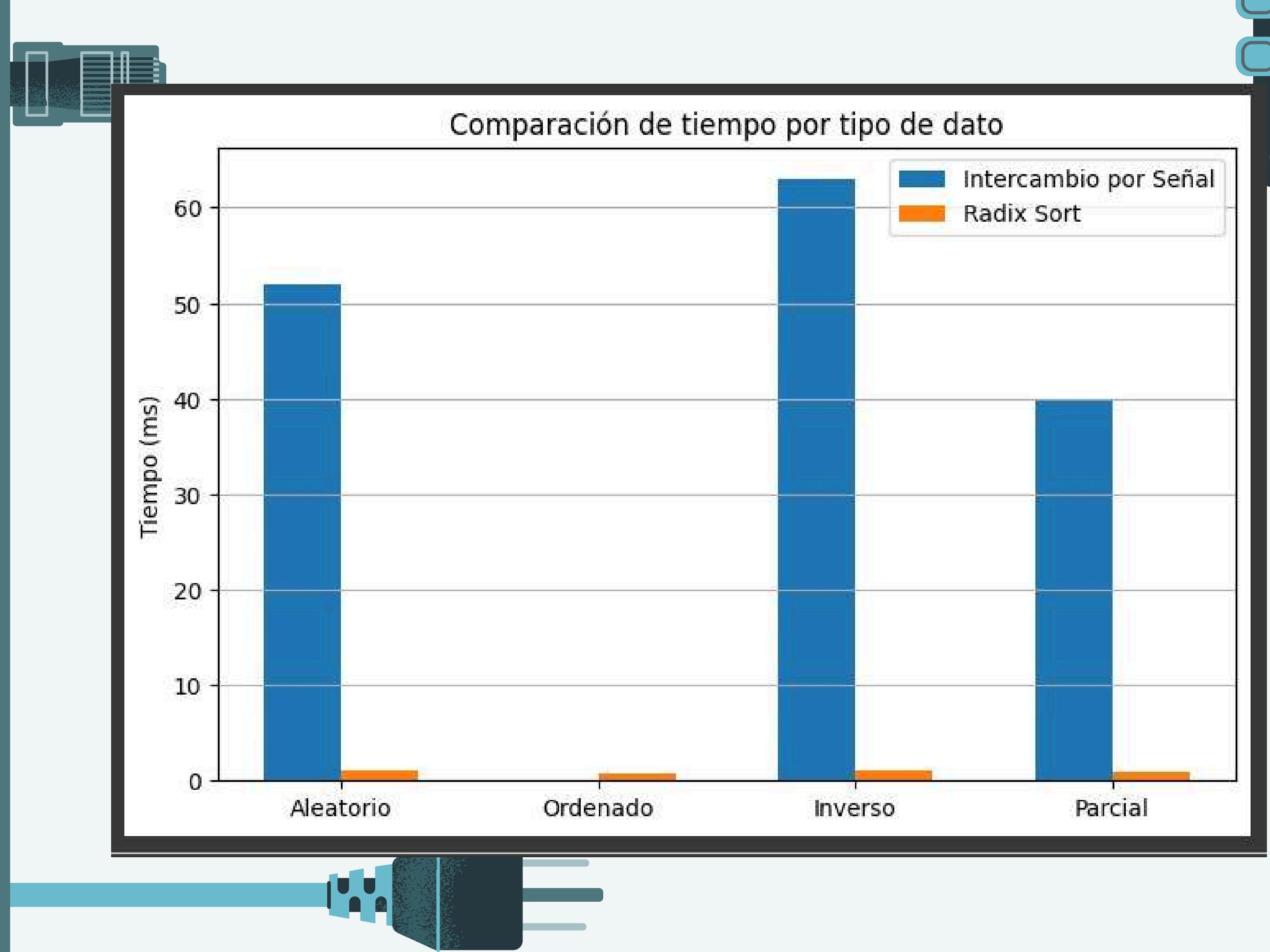
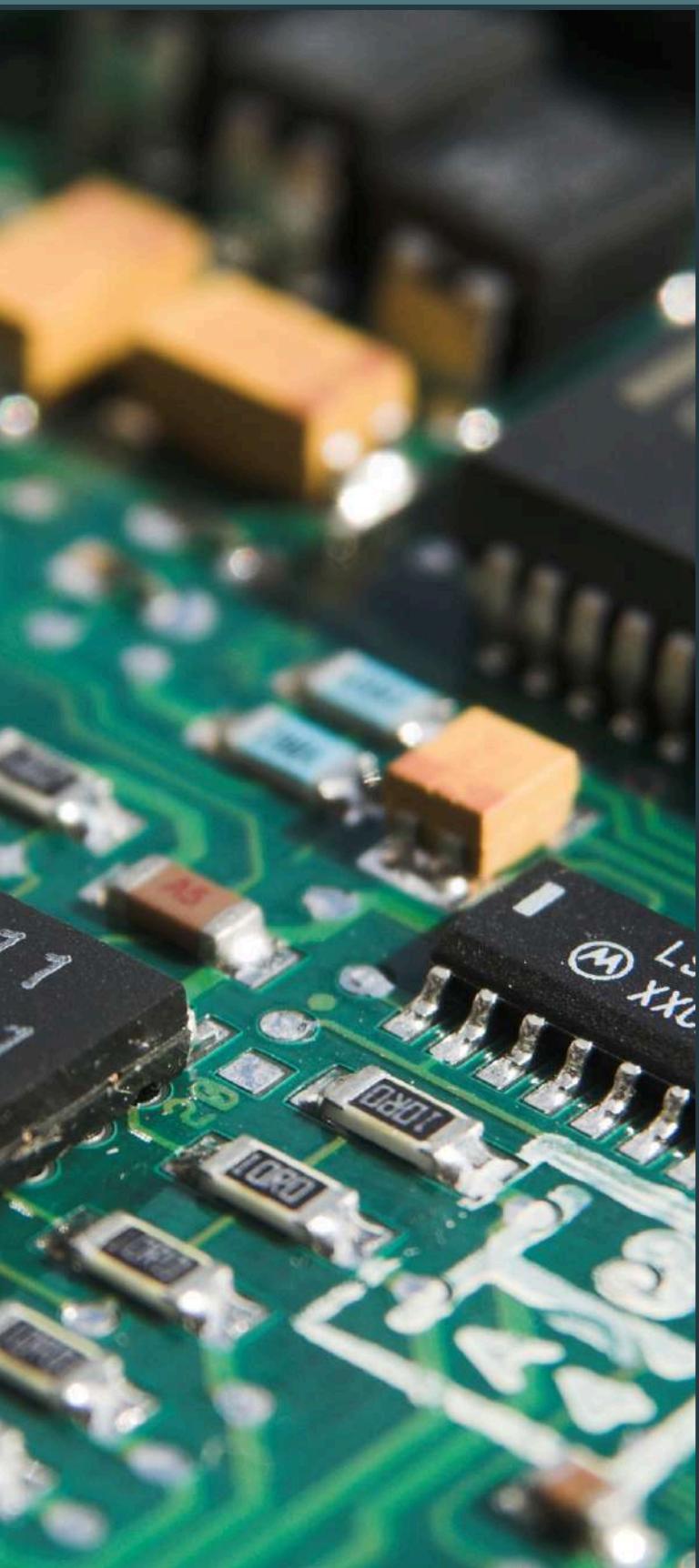
# IMPORTANCIA

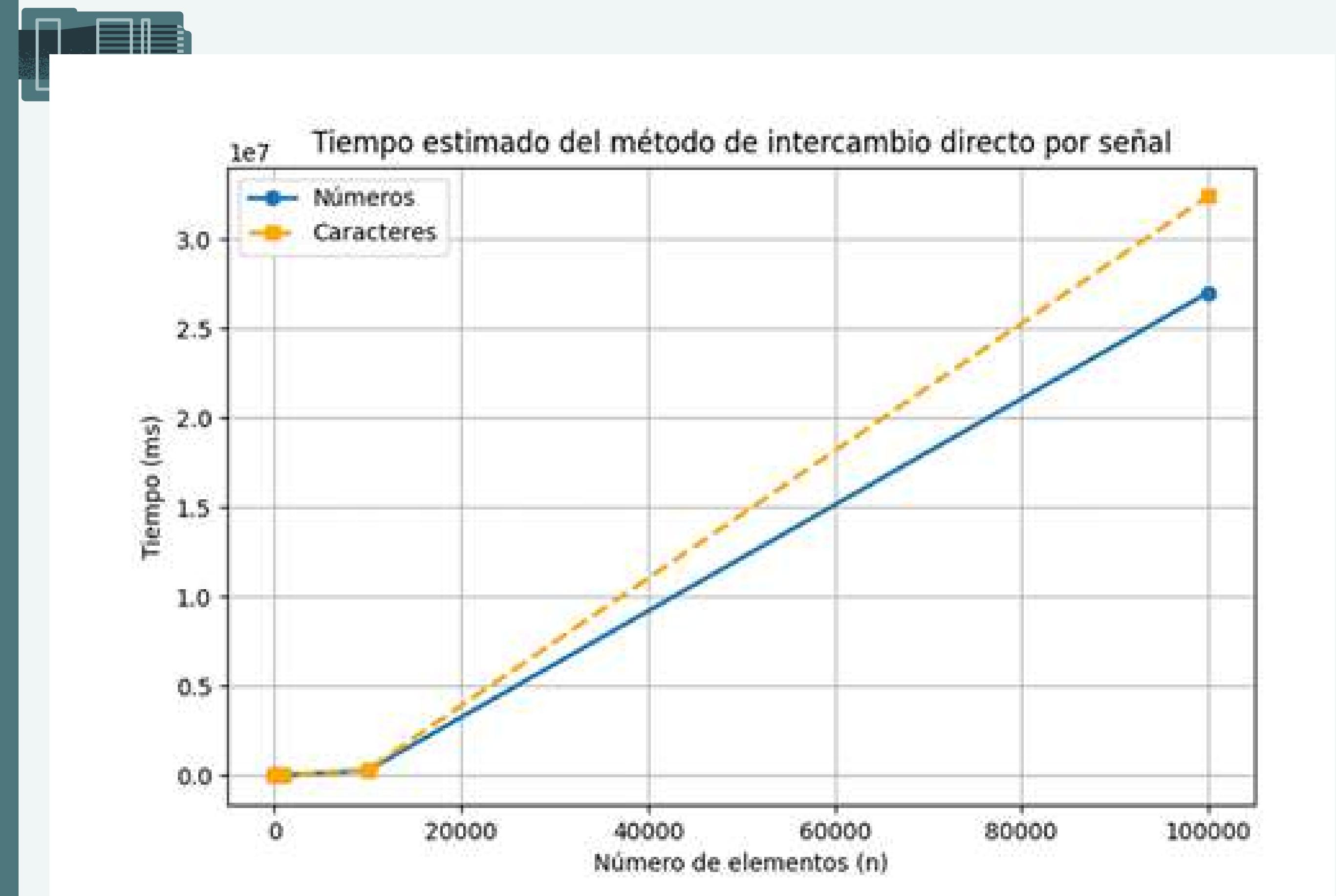
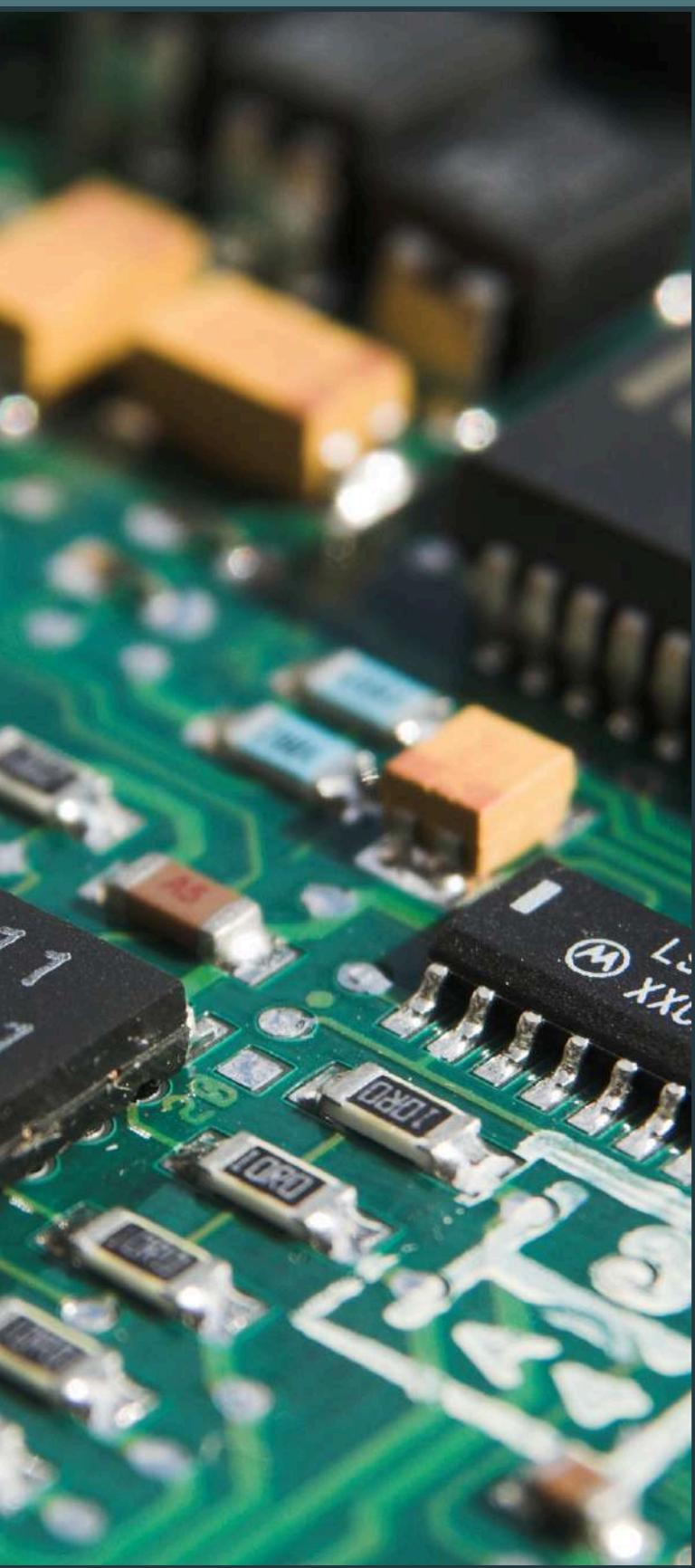
**La importancia de la complejidad computacional radica en su aplicación práctica y en su influencia en diversas áreas de la tecnología y la ciencia.**

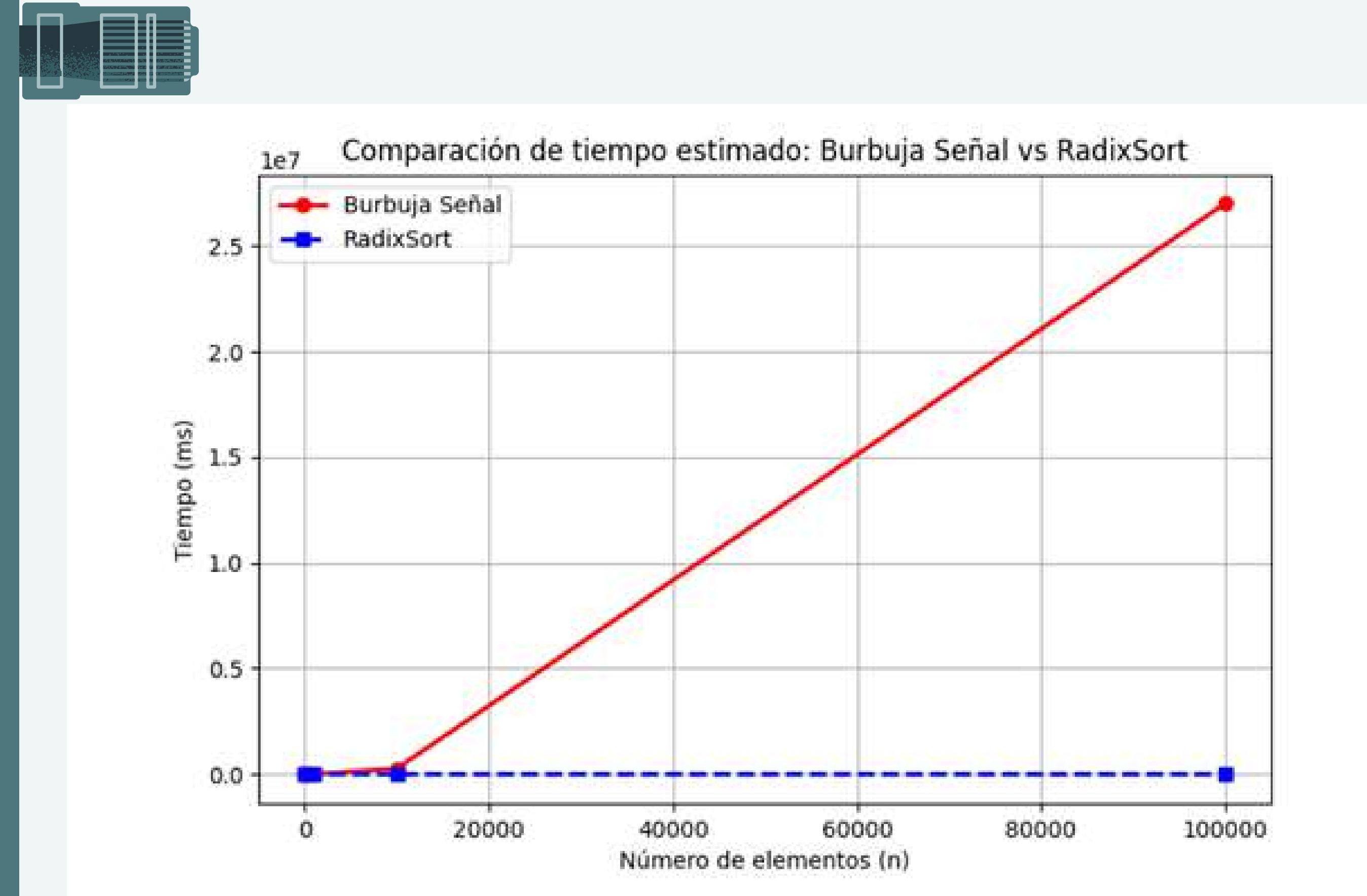
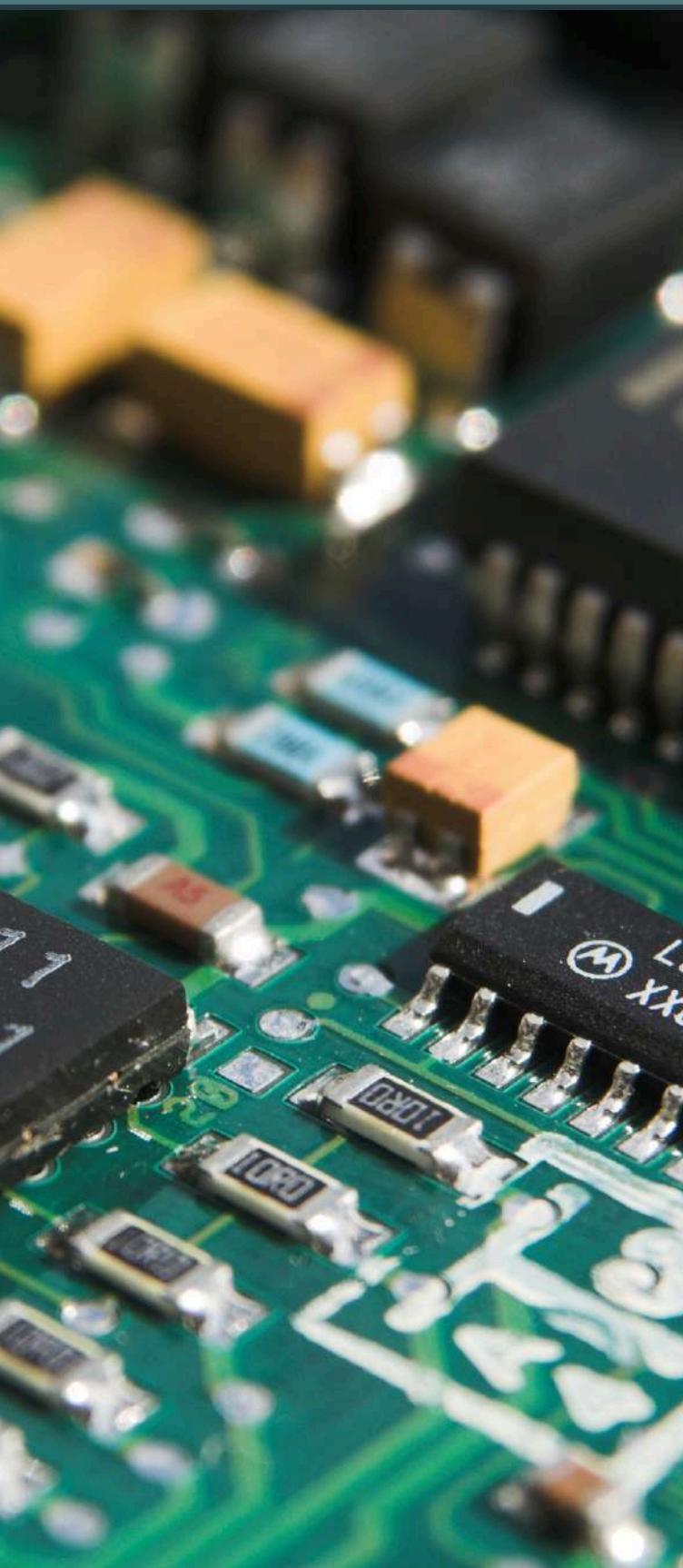
**Algunos puntos destacados**

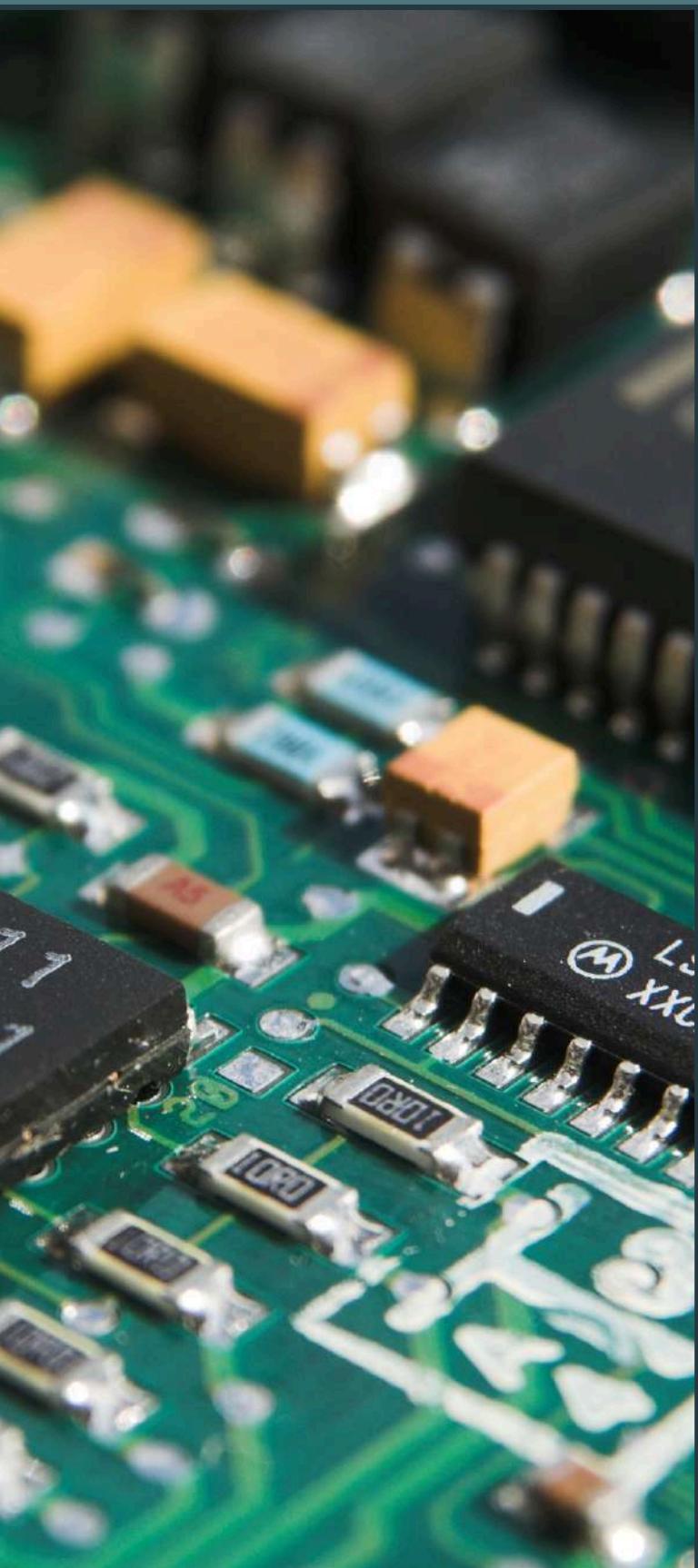




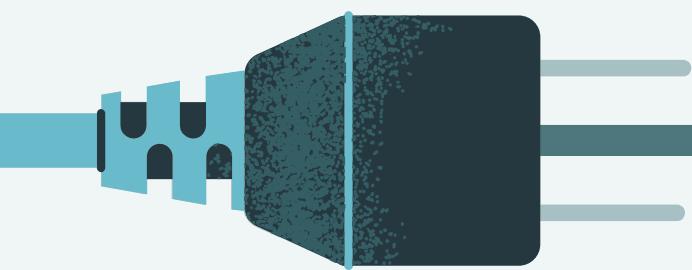
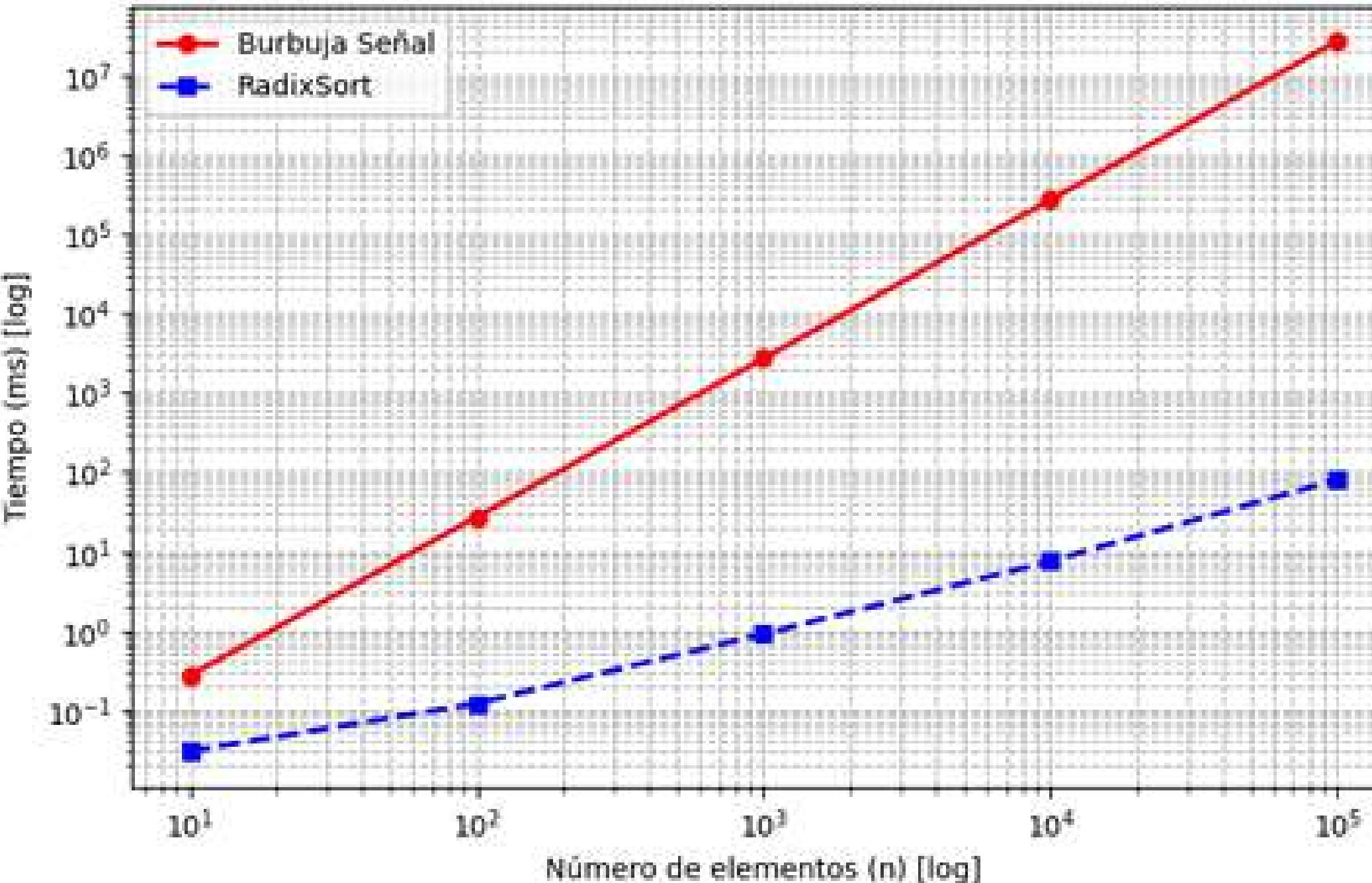


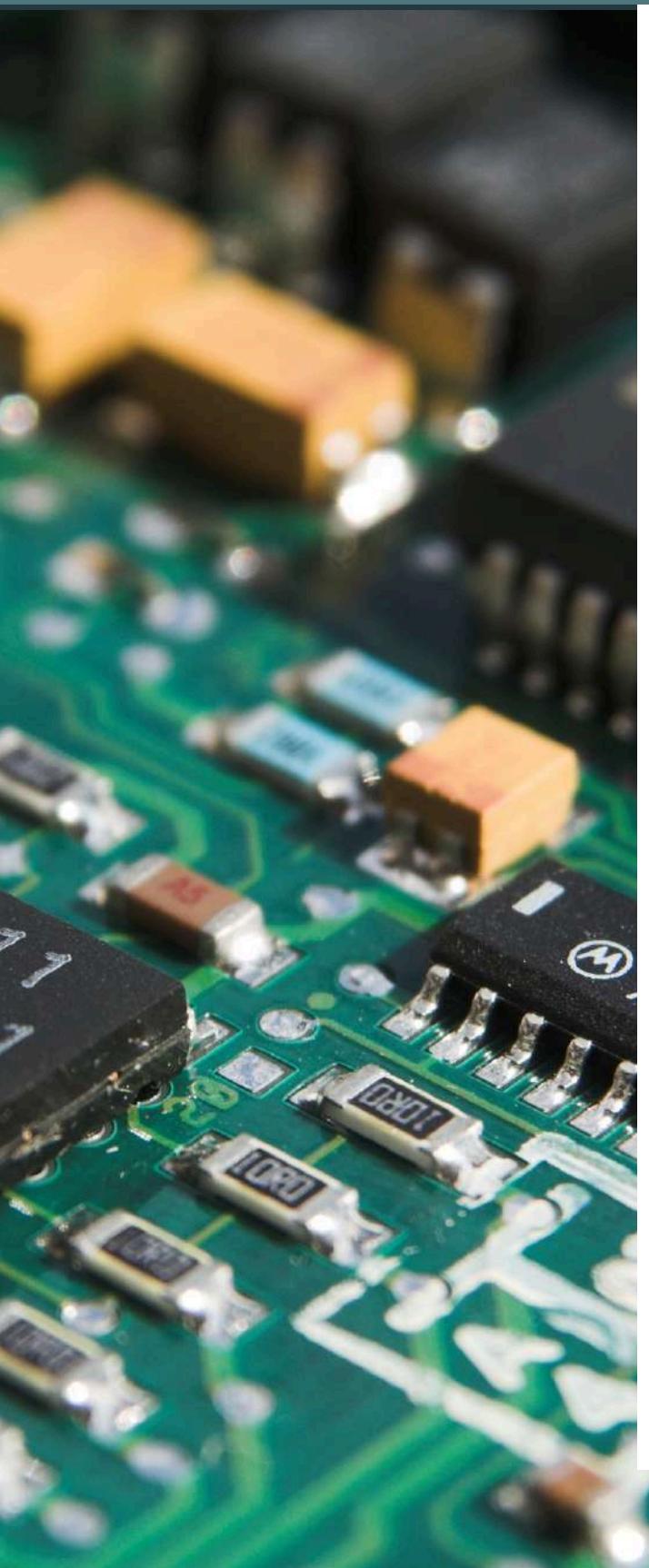




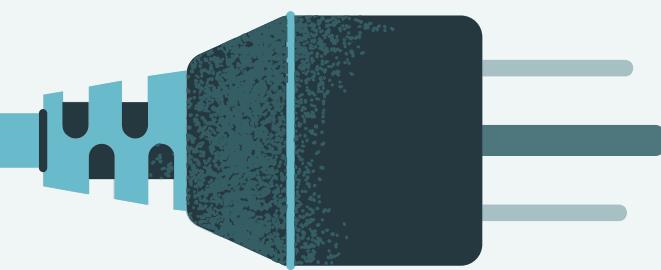
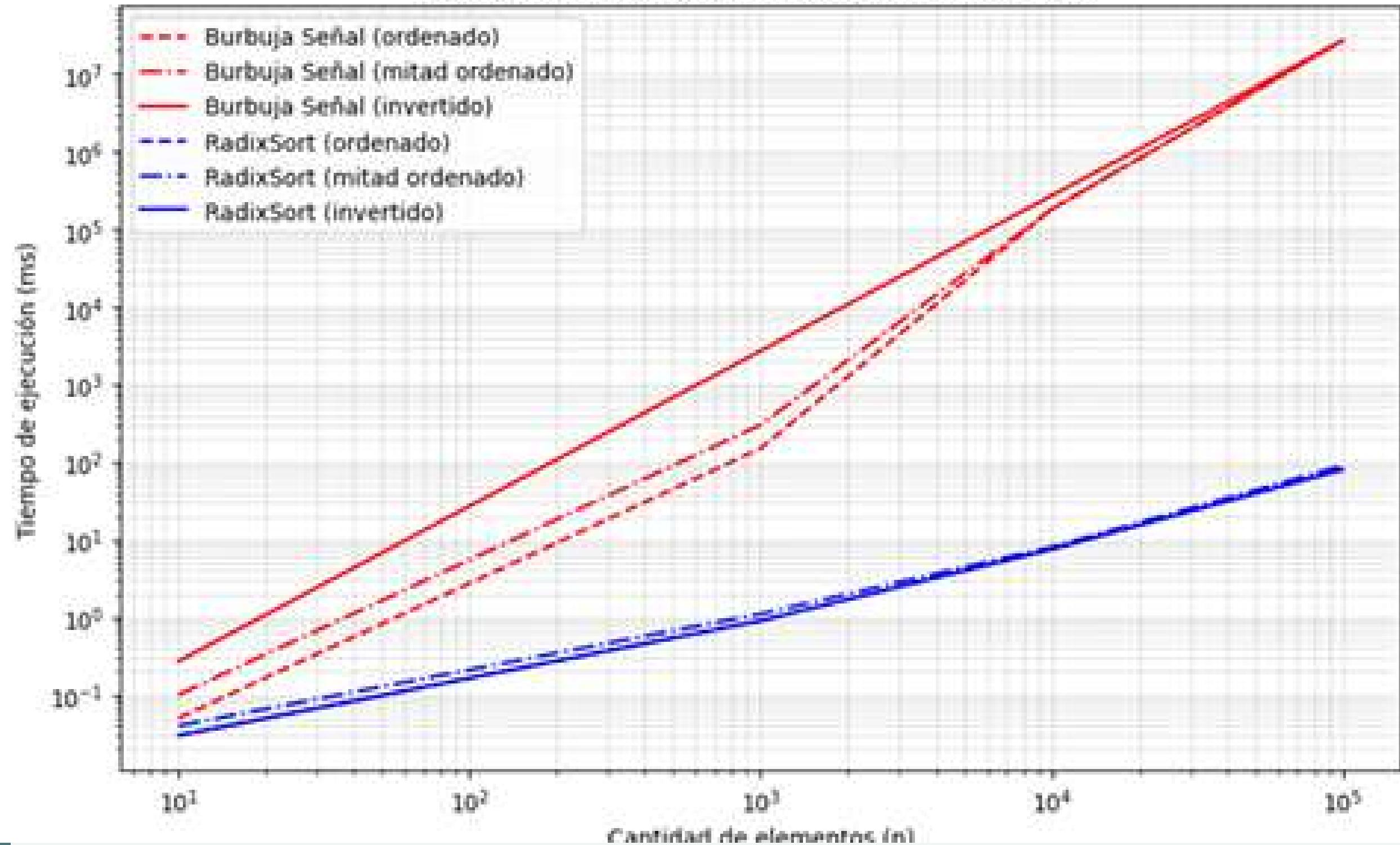


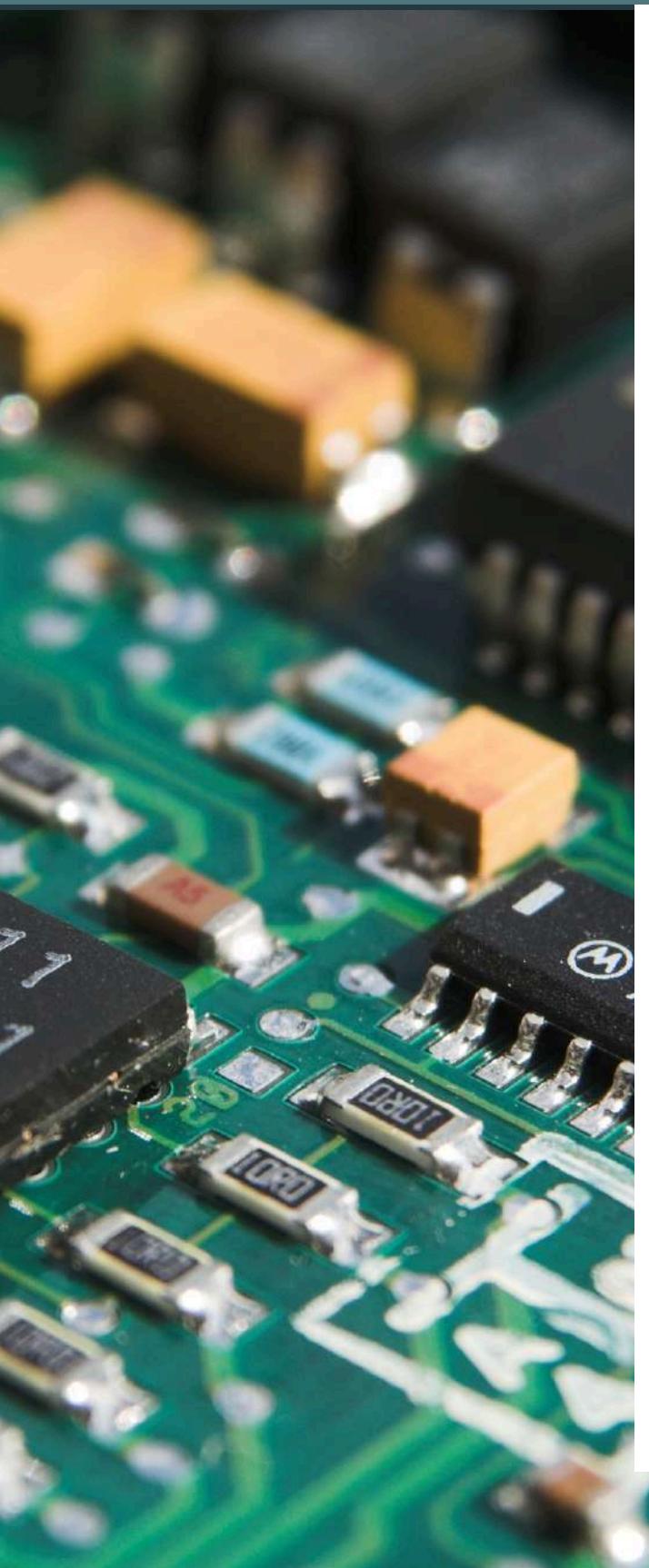
## Escala logarítmica - Crecimiento cuadrático vs casi lineal



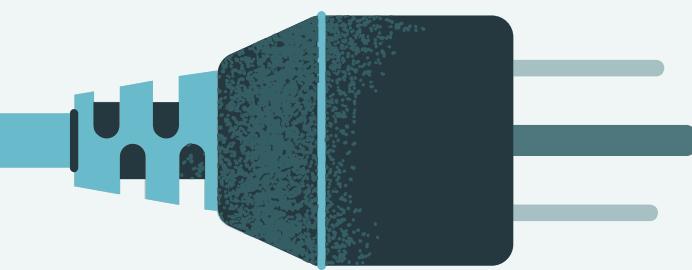
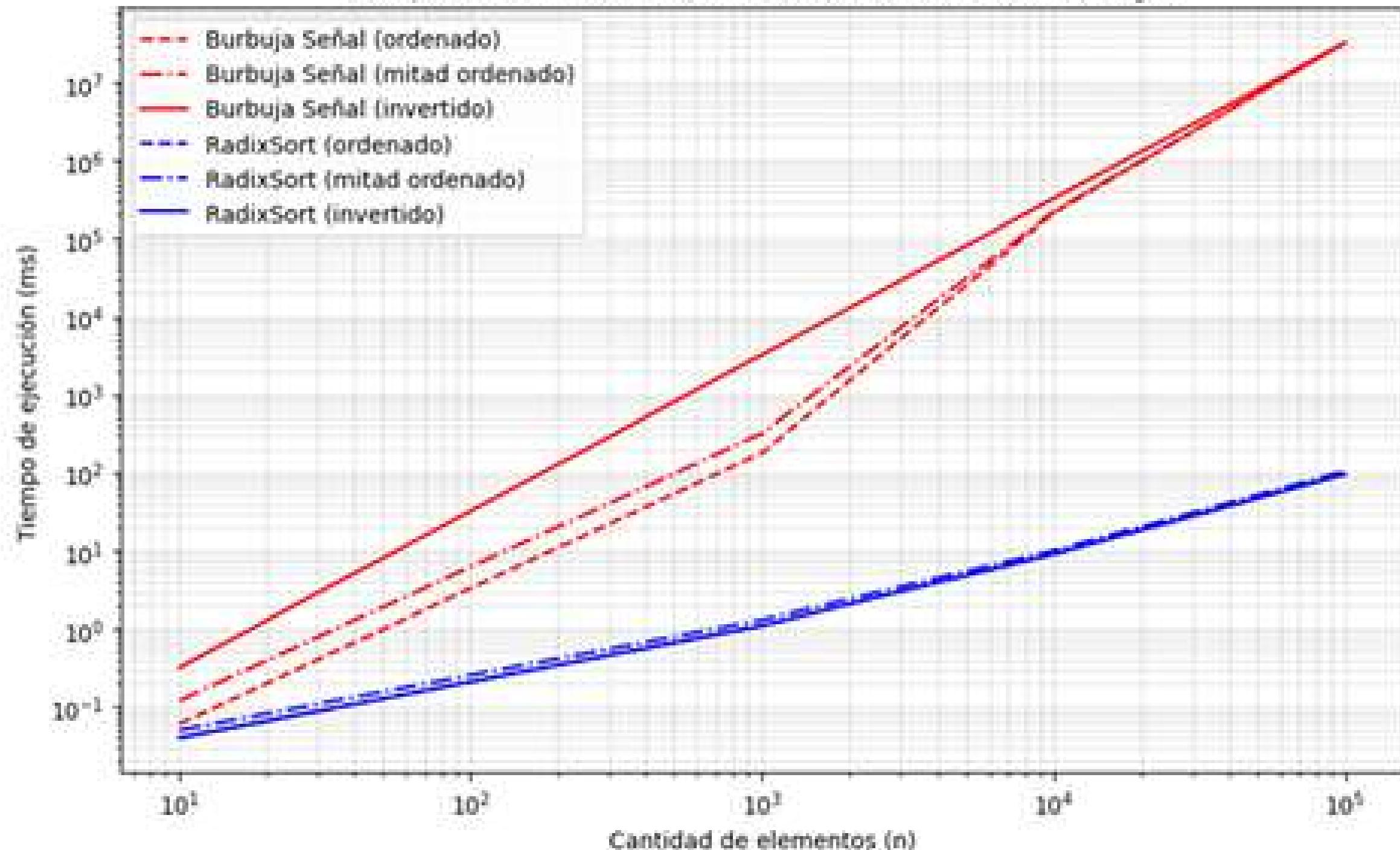


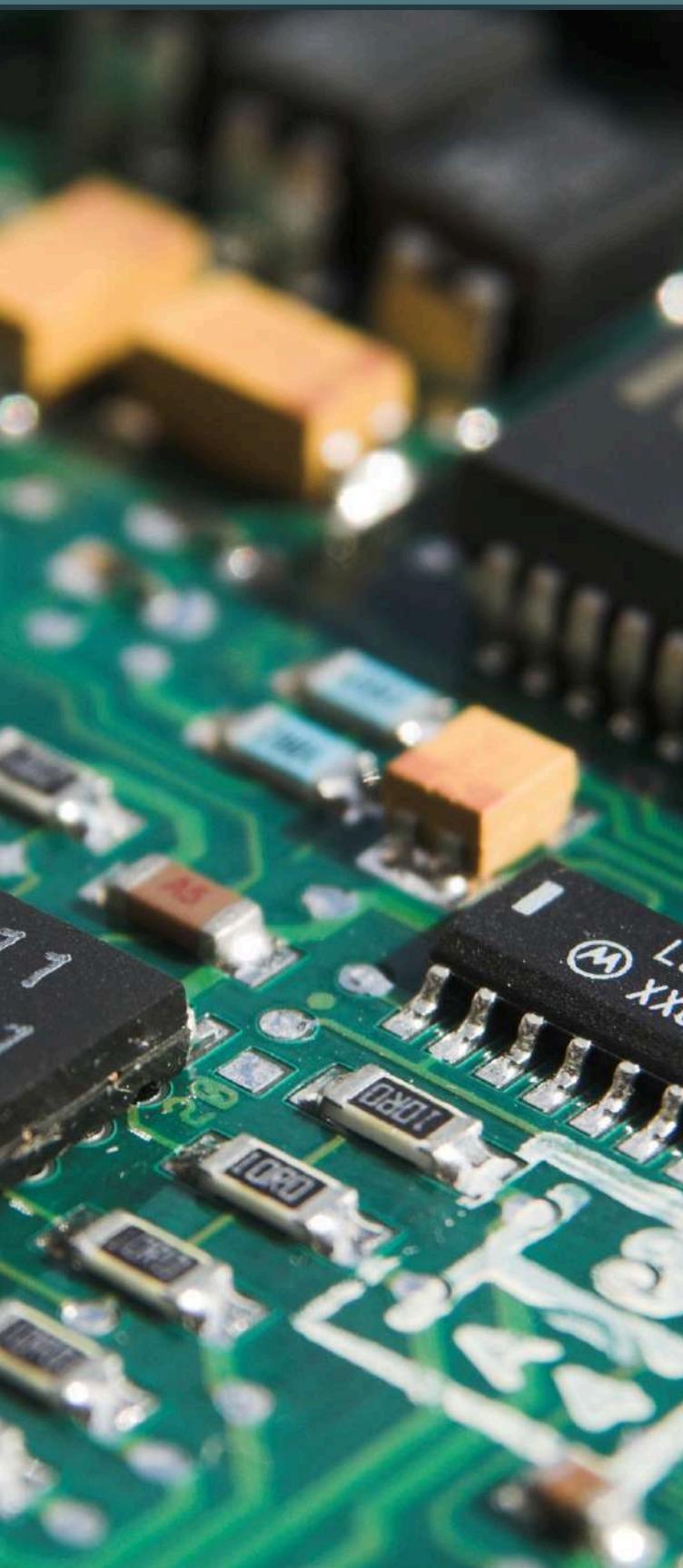
## Comparación de rendimiento - Datos numéricos





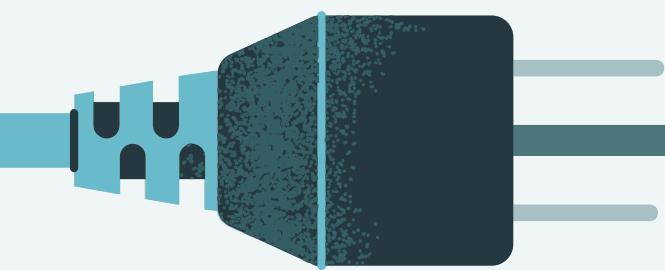
## Comparación de rendimiento - Datos de caracteres (strings)

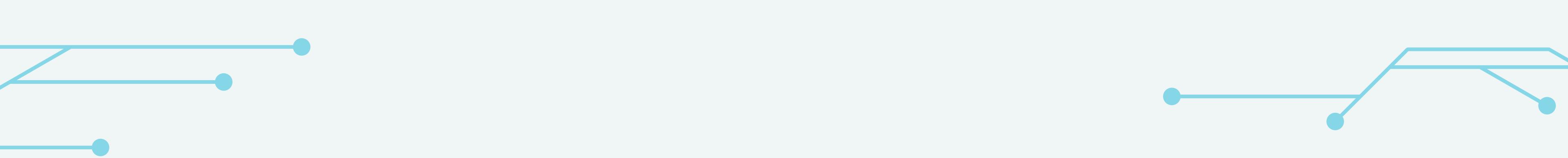




## CONCLUSIONES:

- Radix Sort mantiene mejor rendimiento en grandes volúmenes de datos.
- El método Burbuja con señal mejora el clásico, pero sigue siendo cuadrático.
- Radix Sort es más escalable y eficiente en tiempo.
- Burbuja señal es útil solo en listas pequeñas o educativas.
- La eficiencia de Radix Sort aumenta con el tamaño del conjunto.





[www.unsitiogenial.es](http://www.unsitiogenial.es)

# MUCHAS GRACIAS

