

ALGORITMO DE ORDENAMIENTO

BUBBLE SORT

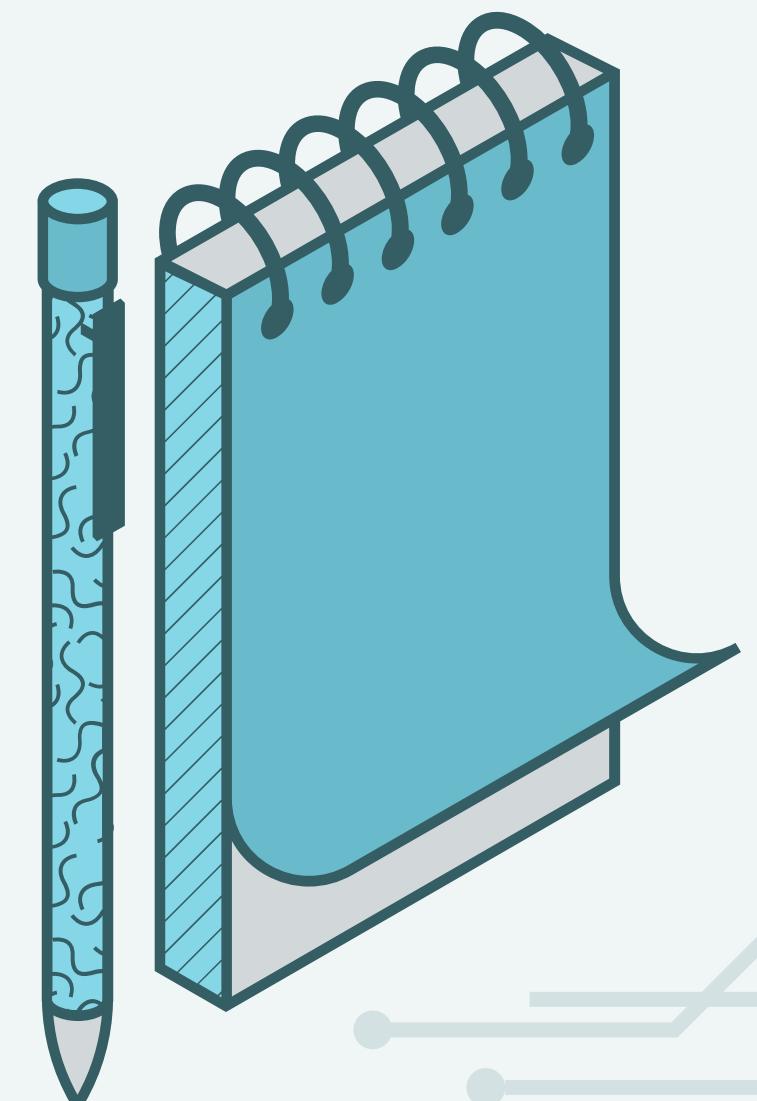
ORDENAMIENTO POR INTERCAMBIO

E01 -----Diego Álvarez -----Valeria Ix

BUBBLE SORT (ORDENAMIENTO DE BURBUJA)

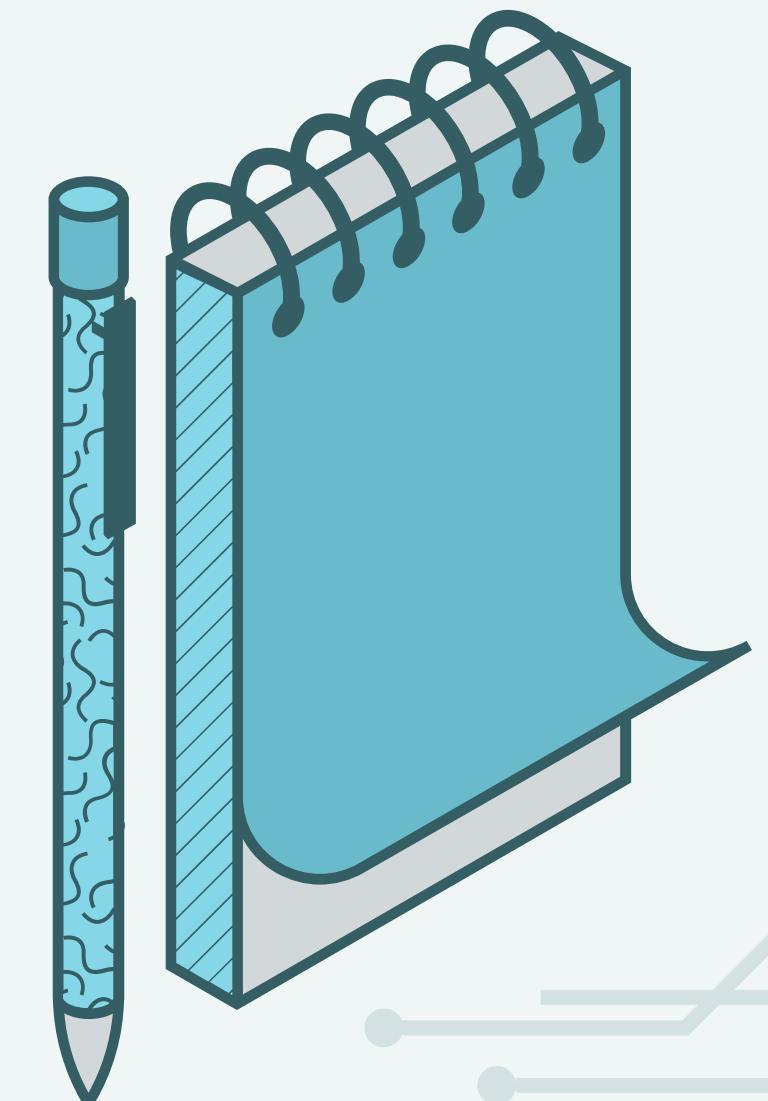
CATEGORÍA: Ordenamiento por Intercambio

- Fue documentado por primera vez en un artículo de 1956.
- Es uno de los algoritmos más populares en la enseñanza de las ciencias de la computación debido a su gran simplicidad.



BUBBLE SORT (ORDENAMIENTO DE BURBUJA)

- Compara elementos adyacentes
- Intercambia si están en orden incorrecto
- Repite hasta que no haya intercambios
- El valor no ordenado más grande siempre alcanzará su posición final después de cada iteración
- El algoritmo debe iterar $n-1$ veces para listas de longitud n .



FUNCIONAMIENTO:

Lista ejemplo: [5, 2, 8, 1, 14, 17, 4, 3, 19]

En cada pasada, el algoritmo recorre la lista comparando pares.

- Si el elemento de la izquierda es mayor que el de la derecha, se intercambian
- El proceso continúa hasta que no hay más intercambios.

FUNCIONAMIENTO:

Lista ejemplo: [5, 2, 8, 1, 14, 17, 4, 3, 19]

Primera pasada:

- Compara 5 y 2 → intercambia → [2, 5, 8, 1, 14, 17, 4, 3, 19]
- Compara 5 y 8 → no intercambia
- Compara 8 y 1 → intercambia → [2, 5, 1, 8, 14, 17, 4, 3, 19]
- Compara 8 y 14 → no intercambia
- Compara 14 y 17 → no intercambia
- Compara 17 y 4 → intercambia → [2, 5, 1, 8, 14, 4, 17, 3, 19]
- Compara 17 y 3 → intercambia → [2, 5, 1, 8, 14, 4, 3, 17, 19]
- Compara 17 y 19 → no intercambia

Resultado después de la primera pasada:

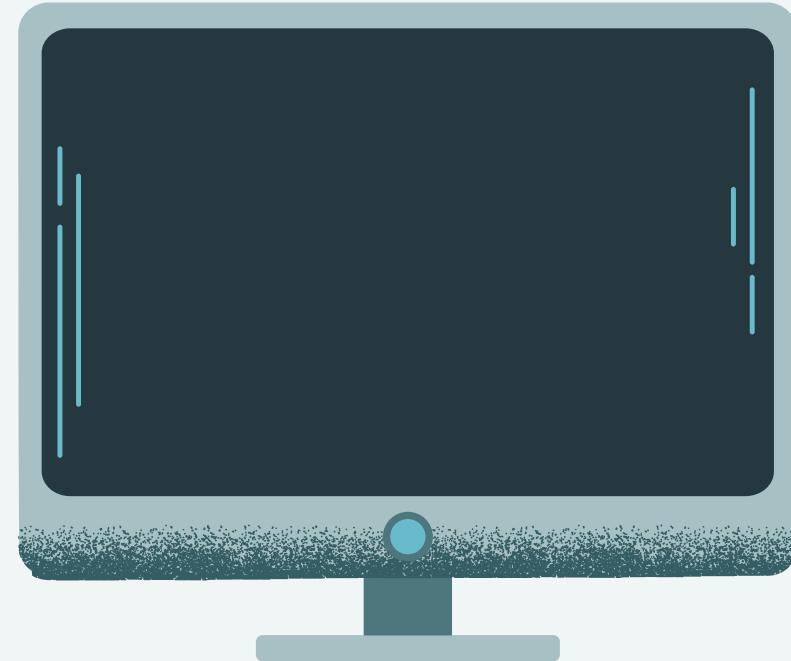
[2, 5, 1, 8, 14, 4, 3, 17, 19]

VENTAJAS Y DESVENTAJAS:



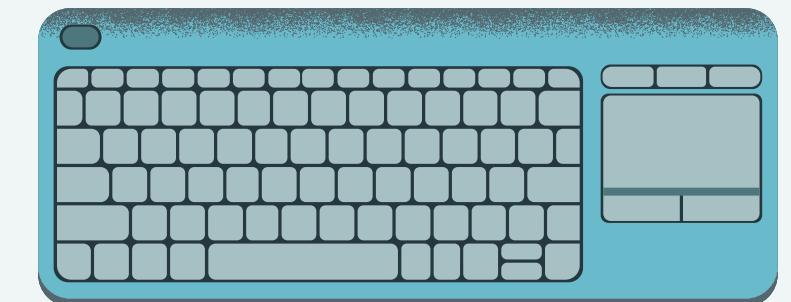
VENTAJAS

- Simple de entender e implementar.
- Estable y in-place.
- Eficiente para listas pequeñas o casi ordenadas.



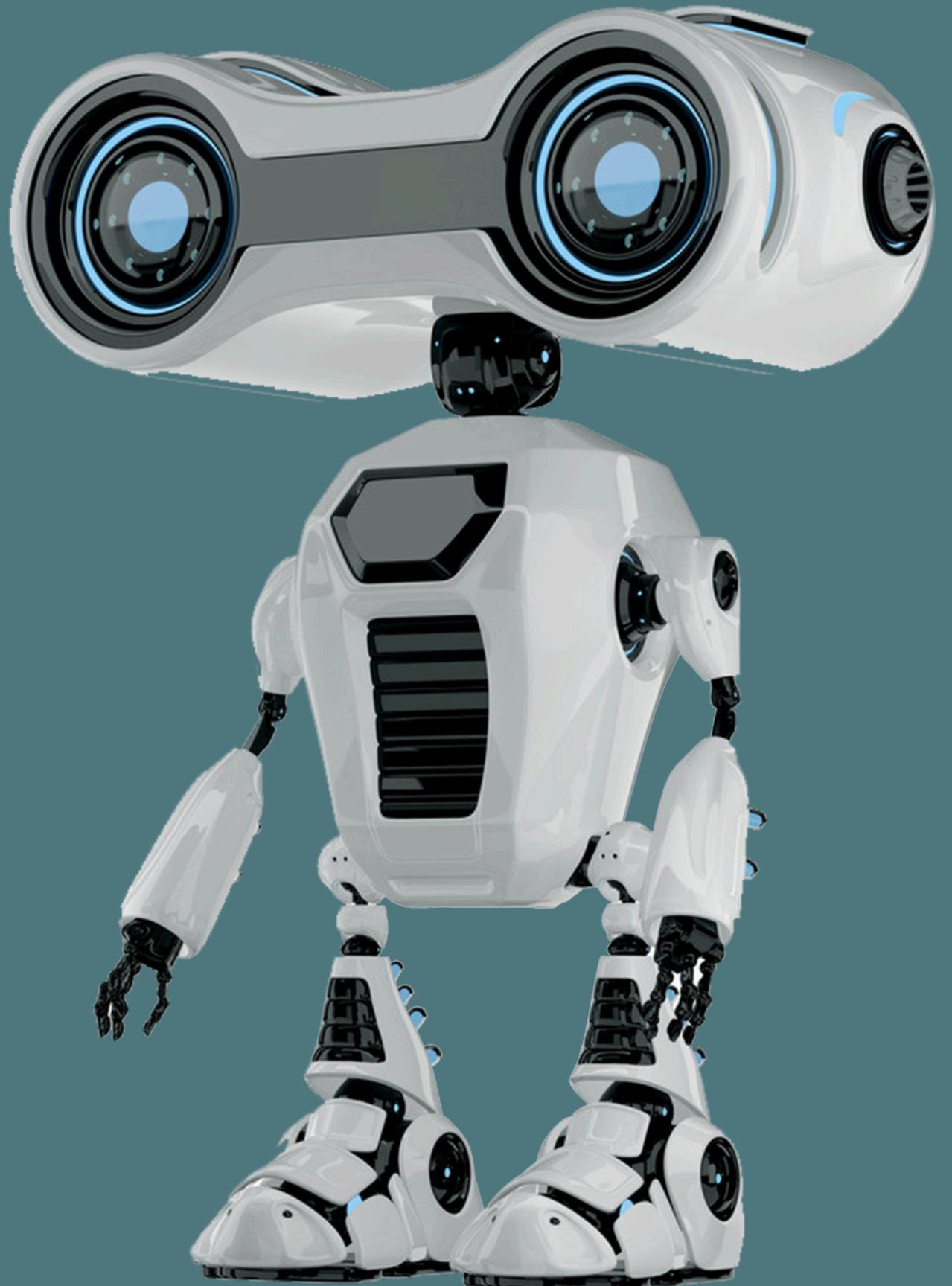
DESVENTAJAS

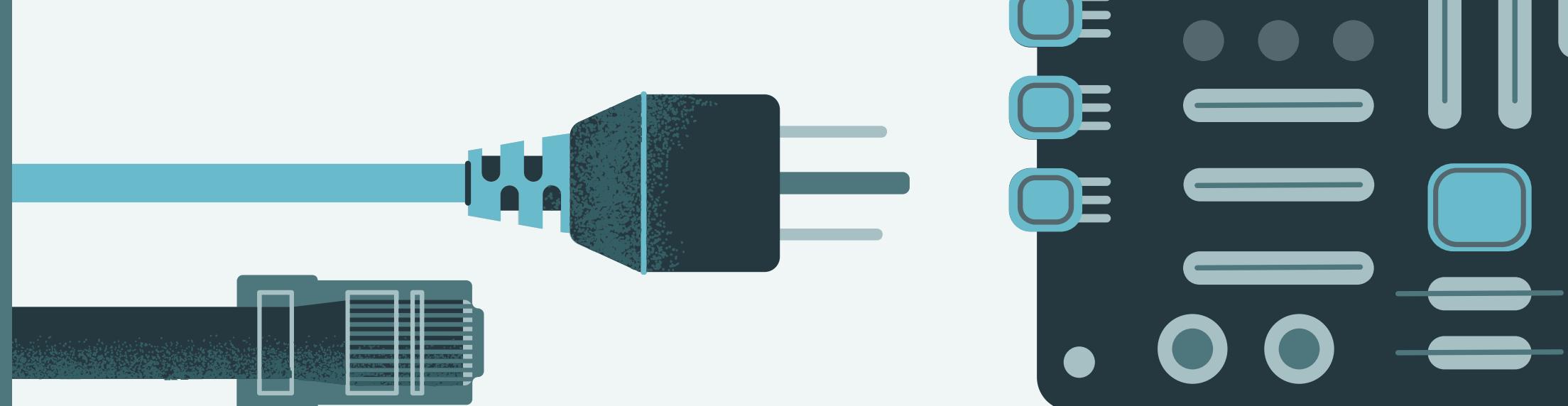
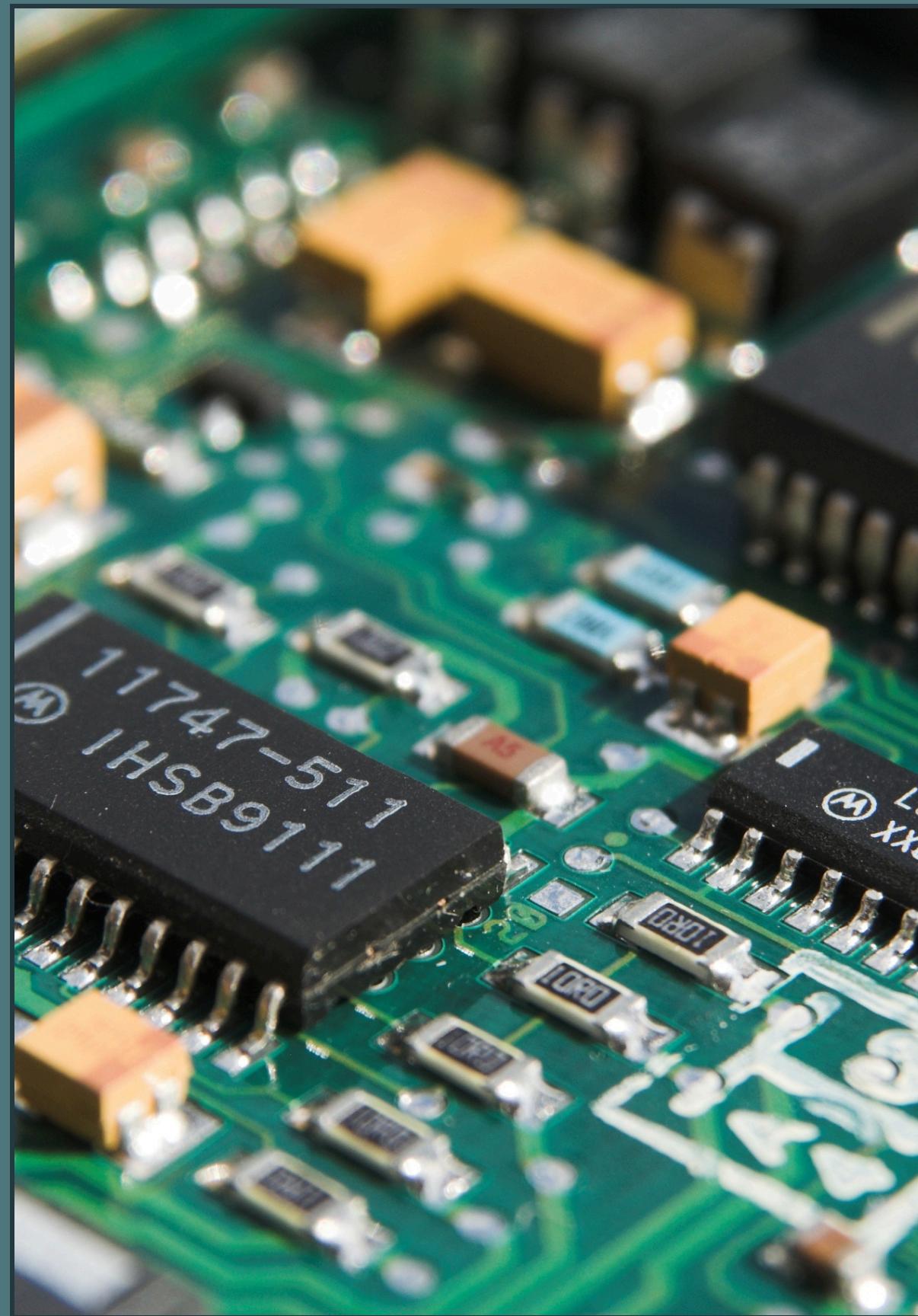
- Muy ineficiente para listas grandes ($O(n^2)$).
- No es adecuado para aplicaciones que requieren alto rendimiento.



CARACTÉRÍSTICAS TÉCNICAS

- Estabilidad: Sí. Mantiene el orden relativo de elementos iguales.
- In-place: Sí. Utiliza espacio adicional constante ($O(1)$).
- Adaptabilidad: Sí. Se beneficia de listas parcialmente ordenadas o ya ordenadas





EFICIENCIA DEL ALGORITMO

Complejidad Temporal:

Mejor Caso: $O(n)$ (Cuando la lista ya está ordenada, debido a la adaptabilidad).

Caso Promedio: $O(n^2)$.

Peor Caso: $O(n^2)$ (Cuando la lista está en orden inverso).

Complejidad Espacial:

Memoria adicional utilizada: $O(1)$ (Espacio constante).

COMPARACIÓN CON OTROS ALGORITMOS

Algoritmo	Mejor Caso	Caso Promedio	Peor Caso	Estable
Bubble Sort	$O(n)$	$O(n^2)$	$O(n^2)$	✓
Quick Sort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	✗
Cocktail Sort	$O(n)$	$O(n^2)$	$O(n^2)$	✓

CASOS DE USO Y APLICACIONES PRÁCTICAS

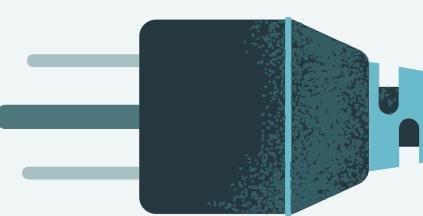
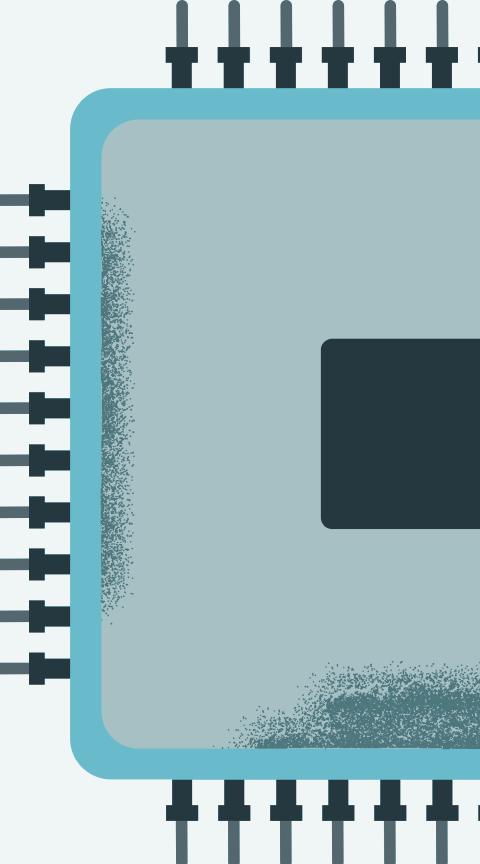
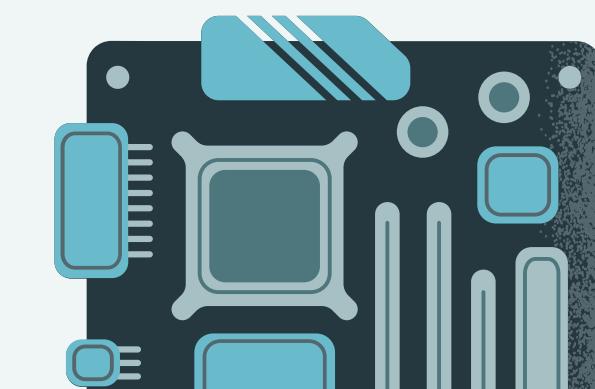
Escenarios de Mayor Eficiencia:

- Cuando la lista es muy pequeña.
- Cuando se espera que la lista esté casi ordenada (debido a su mejor caso $O(n)$).
- Prototipado rápido: Cuando la simplicidad es clave
- Aplicaciones con recursos limitados: Donde el código simple es más importante que la velocidad

Ejemplos Reales:

Se utiliza principalmente con fines educativos.

Raramente se utiliza en sistemas de alto rendimiento como bases de datos o renderización de gráficos debido a su lentitud



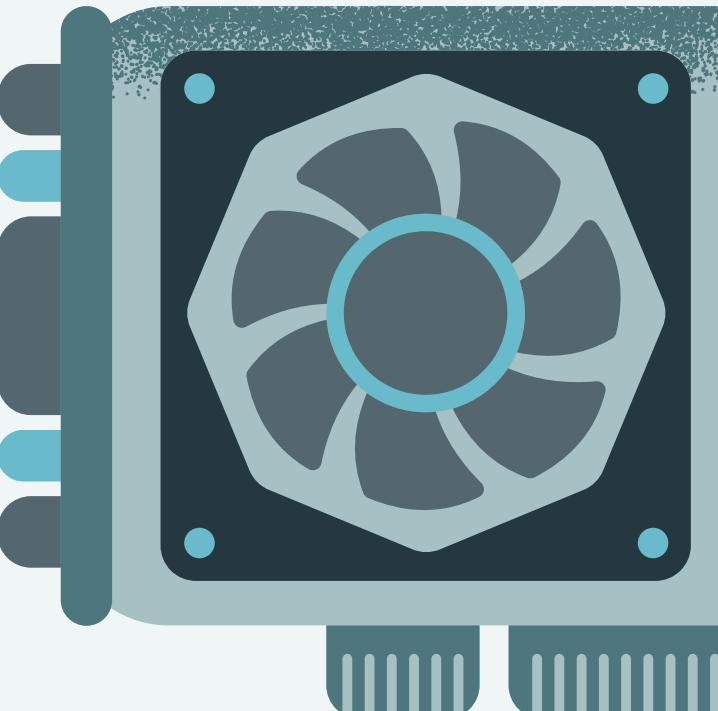
CONCLUSIONES Y RECOMENDACIONES



Conclusiones:

El Bubble Sort es un algoritmo estable y de bajo uso de memoria ($O(1)$).

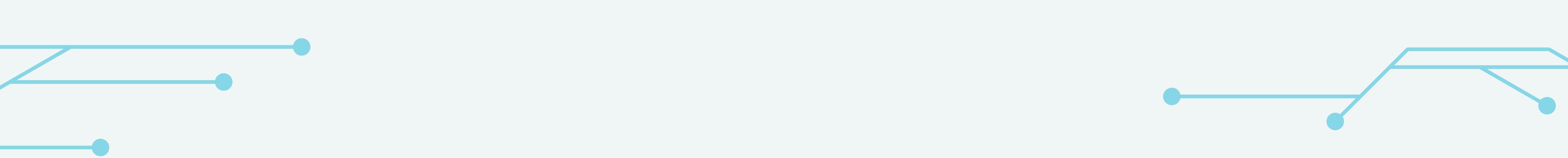
Sin embargo, su complejidad $O(n^2)$ lo hace ineficiente para el ordenamiento de grandes volúmenes de datos.



Recomendaciones de Uso:

Usar solo cuando la simplicidad de implementación es más importante que la velocidad.

Evitar su uso en escenarios donde el rendimiento sea crítico.



www.unsitiogenial.es

MUCHAS GRACIAS

