Tipos de datos	N	Bubble Sort (s)	Merge Sort (s)	Counting Sort (s)	Shell Sort (s)	Mejor algoritmo	Explicación
Aleatorios	1.000	0.0287	0.0021	0.000000	0.002532	Counting	Counting es el más rápido por su complejidad lineal.
Aleatorios	10.000	3.0579	0.0371	0.000000	0.076361	Counting	Counting mantiene mejor rendimiento con más datos.
Aleatorios	100.000	460.37	0.1925	0.053457	1.187629	Counting	Counting escala bien con grandes volúmenes
Ordenados	1.000	0.0000	0.0010	0.000000	0.001104	Counting	Counting sigue siendo eficiente por su bajo costo
Ordenados	10.000	0.0003	0.0102	0.007633	0.018176	Bubble	Bubble destaca porque detecta el orden y se detiene pronto.
Ordenados	100.000	50.04	0.1193	0.061550	0.255829	Counting	Counting escala bien con grandes volúmenes

Inversos	1.000	0.0310	0.0009	0.000000	0.001956	Counting	Counting no depende del orden inicial, por eso rinde mejor.
Inversos	10.000	3.9212	0.0105	0.006525	0.051780	Counting	Counting supera a los otros pese al orden inverso.
Inversos	100.000	820.57	0.1306	0.061055	0.448632	Counting	Counting mantiene tiempos estables incluso en peores casos

1.

Algoritmo	Cuándo es más rápido	Por qué
Counting Sort	Cuando los valores están en un rango pequeño y conocido (como tus datos).	Tiene complejidad lineal O(n+k), no depende del orden inicial.
Merge Sort	Listas grandes de datos con valores muy variados.	Su complejidad es garantizada O(n log n) y no empeora por el orden.
Shell Sort	Tamaños medianos; mejora sobre inserción pero no es óptimo.	Reduce gradualmente las inversiones grandes.
Bubble Sort	Solo cuando la lista ya está ordenada o casi ordenada.	Detecta rápidamente que no tiene trabajo y termina.

2.

Algoritmo	Efecto en lista ordenada
Bubble Sort	Se vuelve muy rápido (O(n)), porque no realiza intercambios.
Merge Sort	Tiempo no cambia, sigue siendo O(n log n), no aprovecha el orden.

Counting Sort	Tiempo igual, no depende del orden, solo del rango.
Shell Sort	Se acelera un poco, pero no tanto como Bubble, sigue O(n log n) aprox.

3.

- Está basado en divide y vencerás, divide la lista en mitades y ordena de manera recursiva.
- Mantiene una eficiencia estable O(n log n) en todos los casos (mejor, peor y promedio).
- Puede adaptarse fácilmente a versiones paralelas.
- Es estable (no altera el orden de elementos iguales).

Por eso funciona mejor que Shell o Bubble en grandes volúmenes.

4.

Algoritmo	Optimización posible
Bubble Sort	Detectar si ya no hubo intercambios → terminar antes. (Muchos ya lo tienen).
Shell Sort	Usar una mejor secuencia de gaps, como Tokuda o Ciura, mejora rendimiento.
Merge Sort	Hacer una versión híbrida: cuando las sublistas son pequeñas usar Insertion Sort.
Counting Sort	Ajustar el rango de valores para evitar gastar memoria innecesaria.

5.

1. Counting Sort es el más rápido siempre que los valores tengan un rango pequeño.
2. Bubble Sort solo gana en listas ordenadas, pero es el peor con listas grandes o inversas.
3. Merge Sort mantiene tiempos estables, independientemente del orden de los datos.
4. Shell Sort mejora con respecto a Bubble, pero no alcanza a MergeSort en eficiencia general.

5. El orden inicial influye mucho en Bubble y Shell, pero casi nada en Merge y Counting.

Conclusión: nuestro equipo llega a la conclusión que Counting Sort es el algoritmo más eficiente y estable en casi todos los escenarios evaluados, demostrando un excelente comportamiento en tiempo y escalabilidad, mientras que Bubble Sort solo resulta competitivo con listas previamente ordenadas.