**BubbleSort vs QuickSort**

a. ¿Cuál algoritmo es más rápido y por qué?

- Quicksort es más rápido porque opera en O(n log n) usando divide y vencerás, mientras Bubblesort es $O(n^2)$ al comparar elementos adyacentes repetidamente.

b. ¿El tiempo de ejecución será el mismo si la implementación del algoritmo es iterativa o recursiva?

- La complejidad teórica es idéntica sea iterativo o recursivo, pero en práctica la versión iterativa es más eficiente por evitar la sobrecarga de llamadas recursivas.

c. ¿Es posible que exista un algoritmo de ordenamiento que sea muy eficiente en consumo de recursos pero que a la vez sea relativamente Rápido?

- Sí, Heapsort es un buen ejemplo pues logra O(n log n) de velocidad mientras mantiene un uso eficiente de memoria al trabajar in-place.

d. Suponga que se planea ejecutar el algoritmo en un sistema computacional con extremadamente bajos recursos de memoria. ¿Cuál de los dos algoritmos de ordenamiento escogería y por qué?

- En sistemas con memoria muy limitada elegiría Bubblesort porque usa memoria O(1) y es simple de implementar, aunque sea más lento.

**Aplicaciones de los algoritmos**

1. ¿Cuál es la diferencia entre el algoritmo de búsqueda lineal y búsqueda por Interpolación?

- La búsqueda lineal examina cada elemento secuencialmente con complejidad O(n), mientras la búsqueda por interpolación estima la posición del elemento usando una fórmula basada en sus valores, logrando O(log log n) en datos uniformemente distribuidos.

2. Suponga que se tiene que buscar un elemento en una lista desordenada, pero se desea optimizar el tiempo de búsqueda por sobre cualquier otra métrica ¿Cómo se podría hacer eso?

- Para optimizar la búsqueda en una lista desordenada, la mejor estrategia sería ordenarla primero (usando Quicksort por ejemplo) y luego aplicar búsqueda binaria. Aunque el ordenamiento inicial tome tiempo, las búsquedas posteriores serán mucho más rápidas (O(log n)).

3. Busque y explique alguna aplicación de la vida real donde el tiempo de búsqueda en una lista o en un arreglo sea crítico para que la aplicación se pueda dar

- Los motores de búsqueda web como Google son un ejemplo crítico, donde millones de consultas deben procesarse en milisegundos. Sin algoritmos de búsqueda eficientes y estructuras de datos optimizadas (como índices invertidos), sería imposible obtener resultados relevantes tan rápidamente como los usuarios esperan.