**ADA 2 MÉTODOS DE ORDENAMIENTO**

**ShellSort**

Descripción del ejercicio: Se ordenará una lista de números enteros utilizando el método de ordenamiento ShellSort.

Justificación del método de ordenamiento: Se utiliza ShellSort debido a su eficiencia para ordenar listas de tamaño mediano y grande.

Posible mejora con otro método de ordenamiento: Para listas pequeñas, se podría considerar el uso de un método de ordenamiento más simple como el de inserción.

Conclusiones: ShellSort es útil cuando se necesitan resultados rápidos en listas de tamaño mediano o grande.

**Quicksort**

Descripción del ejercicio: Se ordenará una lista de palabras alfabéticamente utilizando el método de ordenamiento Quicksort.

Justificación del método de ordenamiento: Se utiliza Quicksort debido a su rapidez y eficiencia para ordenar listas de cualquier tamaño.

Posible mejora con otro método de ordenamiento: Quicksort es adecuado para este caso y no necesita mejoras adicionales.

Conclusiones: Quicksort es una excelente opción para ordenar listas de palabras o cadenas de caracteres.

**Heapsort**

Descripción del ejercicio: Se ordenará una lista de números enteros en orden descendente utilizando el método de ordenamiento Heapsort.

Justificación del método de ordenamiento: Se utiliza Heapsort debido a su garantía de un rendimiento O(n log n) en cualquier caso.

Posible mejora con otro método de ordenamiento: Para listas pequeñas de productos, se podría considerar el uso de un método de ordenamiento más simple como el de selección.

Conclusiones: Heapsort es una excelente opción cuando se requiere un rendimiento garantizado en listas de cualquier tamaño.

**Radix**

Descripción del ejercicio: Se ordenará una lista de números enteros de forma ascendente utilizando el método de ordenamiento Radix sort.

Justificación del método de ordenamiento: Se utiliza Radix sort porque es eficiente para ordenar números con un número fijo de dígitos, como la edad de los estudiantes.

Posible mejora con otro método de ordenamiento: Para listas de registros de estudiantes de tamaño pequeño, se podría considerar el uso de un método de ordenamiento más simple como el de inserción.

Conclusiones: Radix sort es útil cuando se necesita ordenar números con un número fijo de dígitos de manera eficiente, como en el caso de las edades de los estudiantes.