Ejercicio01 (1pto):

Justifica con tus propias palabras por que la complejidad de Mergesort es de 0(nlog n) para el mejor caso, caso promedio y peor caso.

La complejidad de Mergesort es O(n log n) en todos los casos (mejor, promedio y peor). Debido a que Mergesort divide repetidamente la lista en mitades hasta que tenga sub-listas de un solo elemento y luego combina esas sub-listas en orden. La división en mitades es una operación logarítmica, ya que se dividen las listas en dos en cada iteración. Y la combinación de las sub-listas requiere tiempo lineal, ya que se deben comparar y fusionar los elementos. En conjunto, esto resulta en una complejidad general de O(n log n) en todos los casos.

Ejercicio02 (2pts):

Investigar cual es el algoritmo de ordenamiento que usa el método sort de Python. Explicar cuál es su complejidad computacional.

El método sort de Python utiliza el algoritmo de ordenamiento Timsort. Timsort es un algoritmo de ordenamiento híbrido que combina Merge Sort y Insertion Sort. Su complejidad computacional varía según la situación, pero en la mayoría de los casos, tiene un rendimiento de O(n log n).

Mejor caso: En situaciones en las que la lista ya está parcialmente ordenada, Timsort puede aprovechar esto y realizar la ordenación más eficientemente con una complejidad cercana a O(n).

Peor caso: El peor caso de Timsort, al igual que en otros algoritmos de ordenamiento comparativos como Mergesort y Quicksort, ocurre cuando la lista está desordenada de manera aleatoria y no presenta ninguna estructura o patrón que el algoritmo pueda aprovechar para acelerar el proceso de ordenación. En esta situación, Timsort no puede realizar optimizaciones y debe realizar una cantidad significativa de comparaciones y fusiones para ordenar la lista, lo que conduce a una complejidad de O(n log n) en el peor caso.

Timsort es altamente eficiente en la práctica debido a su capacidad para lidiar con una variedad de escenarios de datos, y es el algoritmo de elección en Python para la función sort. Su capacidad para adaptarse a diferentes situaciones lo hace una excelente opción en muchas aplicaciones de ordenamiento.