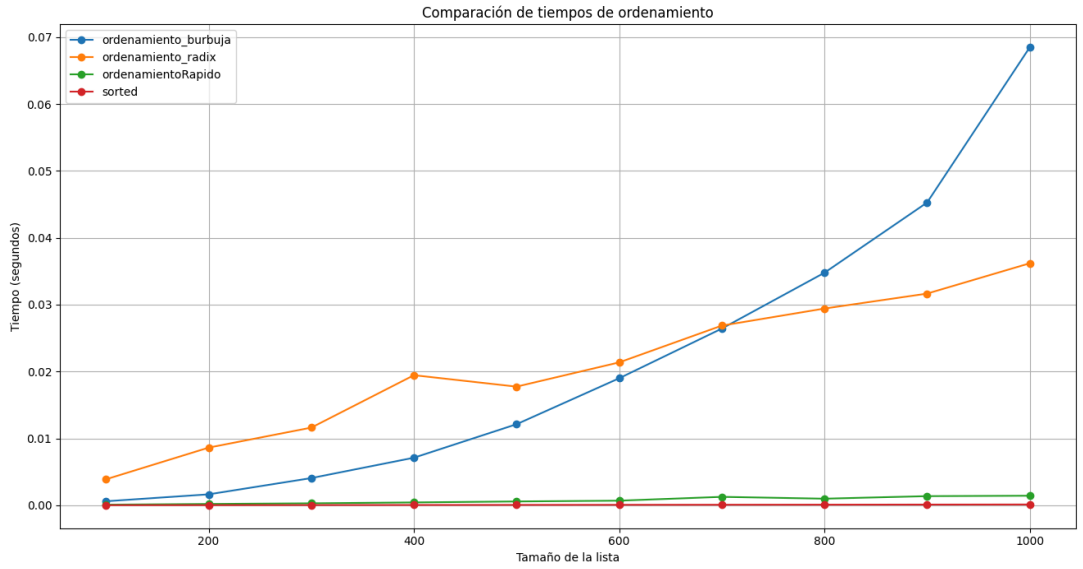
Problema 3:



En la siguiente gráfica observamos el desempeño de los siguientes algoritmos cuyo eje X tenemos la cantidad de elementos en una lista y en el eje Y el tiempo(costo temporal) de su implementación:

a)Ordenamiento burbuja(azul)

b)Ordenamiento radix (naranja)

c)Ordenamiento de residuo(verde)

d)Ordenamiento por sorted(rojo)

- Burbuja:Podemos notar que a mayor tamaño de elementos en una lista, el algoritmo de burbuja tiende asemejarse a una función cuadrática por lo que podemos deducir que presenta complejidad de O(n^2) donde vendría a ser el peor caso posible para este algoritmo,esto se debe a qué este algoritmo hace múltiples pasadas dónde en cada una coloca el elemento en su posición correcta haciendo los intercambios necesarios.

- Quicksort:En base a la gráfica fue el algoritmo de mejor desempeño, asemejándose más a una complejidad O(1) que O(n), deducimos esto porque a mayor cantidad de elementos no notamos un cambio notable en su tiempo de ejecución,dicho algoritmo se basa en el emblema “dividir para vencer”.

- Radix:Presenta una gráfica lineal demostrando una mejor respuesta en su implementación en cuanto a costo temporal que el de burbuja, su análisis de complejidad en este caso podría tratarse de O(n) porque a medida que crecen los elementos de la lista también lo hace de forma proporcional el costo

temporal.

Correcciones:

Cabe mencionar que las gráficas de los algoritmos de ordenamiento Radix y de Residuo las vemos de forma lineal debido a que la implementación del ordenamiento burbuja al ser el primero en ejecutarse consume mayor tiempo que los demás algoritmos , pero lo quedebriamos esperar es que su orden de complejidad debería ser similar al de Burbuja, es decir, una función cuadrática

A su vez comparamos dichos algoritmos con el método sorted que viene incluido en la biblioteca de python el cual consiste en recibir como parámetro un objeto iterable como puede ser una lista,una tupla, entre otros y devuelve un objeto nuevo(mismo tipo de dato) ordenado.Este algoritmo de ordenamiento se basa en el algoritmo Timsort el cual combina las ventajas de los algoritmos de ordenamiento de mezcla y por inserción, el mismo tiene un orden de complejidad o(n) para los mejores casos y en promedio o peor de los casos tiene orden de complejidad o(n log n).

En base a estos datos los que podemos esperar es un rendimiento más efectivo en comparación con los demás algoritmos(gráfica casi lineal) pero que a medida que se aumenta la cantidad de elementos la misma tiene a tener un ligero “curvamiento” pero no llega a ser tan exuberante como la gráfica de una función cuadrada