Problema 3

Objetivos

- Implementar los algoritmos de ordenamiento quicksort, burbuja y radix sort
- Corroborar que funcionen correctamente con listas de números aleatorios de cinco dígitos generados aleatoriamente (mínimamente de 500 números en adelante).
- Medir los tiempos de ejecución de tales métodos con listas de tamaño entre 1 y 1000.
- Graficar en una misma figura los tiempos obtenidos.
- Determinar el orden de complejidad O de cada algoritmo y justificar con un análisis a priori

Desarrollo

En este programa se utilizan los siguientes algoritmos de ordenamiento:

Ordenamiento Burbuja

Este es el más simple pero también el más ineficiente. Su proceso consta de hacer múltiples pasadas a una lista comprando los ítems adyacentes e intercambiándolos si no están en orden hasta que la lista esté ordenada.

- Complejidad:
  Mejor de los casos: O(n)
  Peor de los casos: $O(n^2)$
  Caso promedio: $O(n^2)$

Ordenamiento Quicksort

Selecciona un pivote, un elemento de la lista, y particiona el resto de la lista en sublistas de elementos mayores al pivote y menores al pivote.

- Complejidad:
  Mejor de los casos: O(n log(n))
  Peor de los casos: $O(n^2)$
  Caso promedio: O(n log(n))

Ordenamiento Radix Sort

Ordena los elementos analizando cada dígito y los procesa del menos significativo al más significativo. Utiliza el ordenamiento por cuentas como algoritmo auxiliar para ordenar los números por posición de dígito.

- Complejidad:
  Mejor de los casos: O(n log(n))
  Peor de los casos: $O(n^2)$
  Caso promedio: O(n log(n))

Ordenamiento Sort de las listas implementadas en Python

La función sort de python usan un algoritmo llamado timesort que combina el algoritmo de Mezcla (divide la lista por la mitad recursivamente) y de inserción (Mantiene una sublista ordenada en las posiciones inferiores de la lista y cada ítem nuevo se reinserta en la sublista previa).
Utiliza el ordenamiento por inserción para ordenar pequeñas secuencias, que es muy rápido para listas cortas o casi ordenadas, y luego las fusiona mediante mezcla.

- Complejidad:
  Mejor de los casos: O(n)
  Peor de los casos: O(n log(n))
  Caso promedio: O(n log(n))