ANALISIS DEL ORDEN DE COMPLEJIDAD DEL ALGORITMO DE ORDENAMIENTO IMPLEMENTADO

Integrantes:
- González Gabriela
- Labanca Alvaro
- Pérez Albarracín Maximiliano

Para el método de ordenamiento realizado, se utilizó un comportamiento de intercambio directo, recorriendo la lista n veces para asignar la cabecera de comparación y a su vez cada cabecera será comparada con el elemento siguiente de la lista, mediante otra operación while que se repetirá (n-1) veces, como se puede ver en el código siguiente:

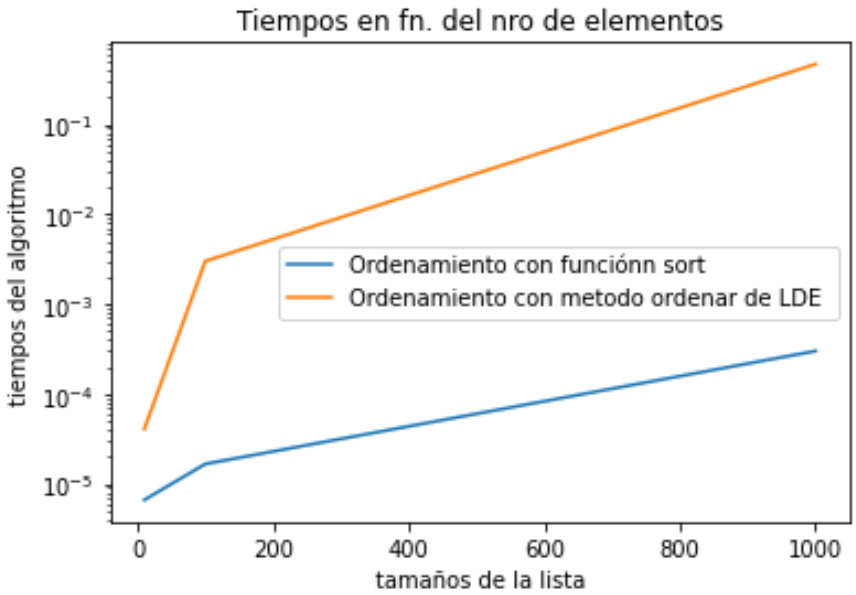
```
def ordenar(self):
    end = None
    while end != self.cabeza:
        p = self.cabeza
        while p.siguiente != end:
            q = p.siguiente
            if p.dato > q.dato:
                p.dato, q.dato = q.dato, p.dato
            p = p.siguiente
        end = p
```

Por lo cual este comportamiento se aproxima a una función $T(n^2)$ o en término de *orden de magnitud* $O(n^2)$.

Como se puede observar en el script adjunto “análisis_orden_complejidad.py”.Para realizar la gráfica del orden de complejidad, se tuvo en cuenta el tiempo de ejecución necesario para llevar a cabo lo operación de ordenamiento mediante el método creado “ordenar()” en función del aumento de tamaño de la lista. Para lo cual se creó la lista doblemente enlazada, a la cual se le agrego elementos al azar, y se fue aumentando el tamaño de la misma.

A su vez para poder determinar la eficiencia del método con respecto a métodos avanzados para ordenar listas en Python, se repitió el mismo procedimiento en iguales condiciones, pero esta vez creando una lista y posterior aplicación de la función sort(), el cual es un método con orden de complejidad O(n.logn).

A continuación se muestra las gráficas en iguales condiciones:



En la gráfica se puede observar que a medida que aumenta el tamaño de la lista, al método ordenar() le lleva más tiempo de ejecución poder ordenar la lista.

Por otro lado al comparar con el tiempo que le incurre al método avanzado sort, ordenar una lista de iguales tamaños, es mucho menor que con el implementado, en aproximadamente dos órdenes de magnitud menor. Por lo cual se concluye que el método ordenar(), no sería tan eficiente como un método avanzado.