Estructura de Datos y Algoritmos II
Aproximación Intuitiva a la Complejidad Algorítmica

Nombre: Sara López Marín Clase: 1658 Actividad: Práctica 0

Entregables

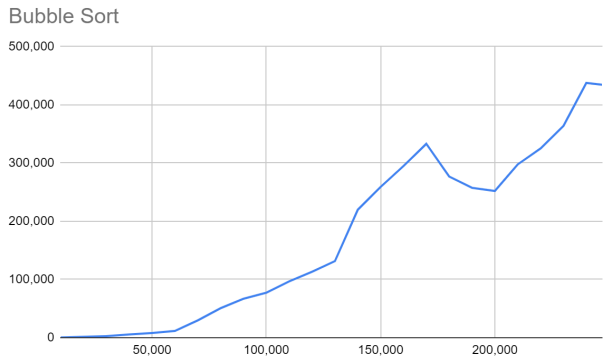
1. Código fuente: Archivos adjuntos en la carpeta. Repositorio con ambos: https://github.com/slopma/EDyA_Practica0_ComplejidadAlgoritmica

2. Tabla: Sirve para visualizar los resultados de la ejecución.

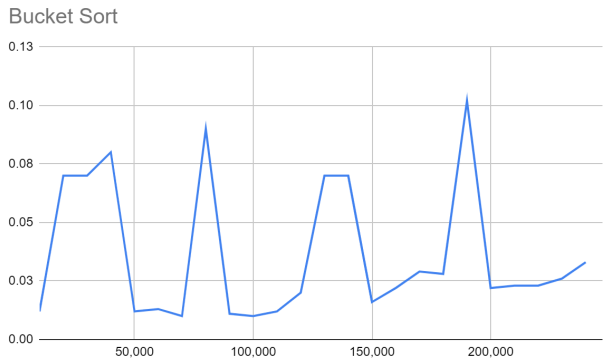
Ciclo	Número de datos	Bubble Sort (s)	Bucket Sort (s)
1	10,000	274	0.01
2	20,000	1,449	0.07
3	30,000	2,657	0.07
4	40,000	5,598	0.08
5	50,000	8,113	0.01
6	60,000	11,616	0.01
7	70,000	29,711	0.01
8	80,000	50,775	0.09
9	90,000	66,856	0.01
10	100,000	77,312	0.01
11	110,000	96,748	0.01
12	120,000	113,329	0.02
13	130,000	131,668	0.07
14	140,000	219,907	0.07
15	150,000	259,017	0.02
16	160,000	295,052	0.02
17	170,000	333,155	0.03
18	180,000	276,972	0.03
19	190,000	257,416	0.10
20	200,000	252,067	0.02
21	210,000	297,732	0.02
22	220,000	325,154	0.02
23	230,000	363,949	0.03
24	240,000	437,776	0.03
25	247,047	434,455	0,04
Total Tiempo:		**4,349,366 s**	**0,919 s**

3. <u>Máquina utilizada:</u> Las especificaciones de la máquina utilizada:
    - Procesador: 12th Gen Intel(R) Core(TM) i5-1235U
    - Velocidad: 1300 MHz (1.3 GHz)
    - Número de Núcleos: 10 núcleos principales.
    - Número de Hilos: 12 hilos lógicos.
    - Memoria Física Total: 7,69 GB
    - Capacidad Disco: 474 GB

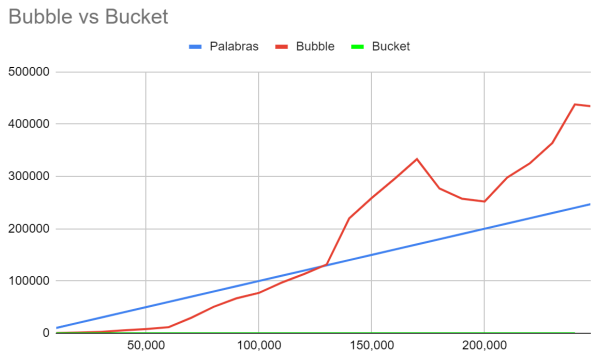
4. <u>Gráficas:</u> Sirve para visualizar de forma gráfica los resultados de la ejecución.
    - Bubble:



    - Bucket:



    - Comparación:



5. Análisis y conclusiones:

   Rendimiento de Bubble Sort vs Bucket Sort:

   - *Bubble Sort* muestra una tendencia a incrementar su tiempo de ejecución de manera no lineal, especialmente a medida que aumenta el número de palabras. Esto es típico de un algoritmo con una complejidad de tiempo de $O(n^2)$, ya que compara pares de elementos y, para listas más grandes, su desempeño se deteriora considerablemente.
   - *Bucket Sort*, es prácticamente insignificante en términos de tiempo de ejecución a lo largo de los diferentes tamaños de entrada. Esto es esperable debido a su eficiencia en casos donde los datos pueden ser distribuidos uniformemente en los buckets, con una complejidad de tiempo cercana a $O(n + k)$ en el mejor de los casos. Este rendimiento es mucho mejor que el de Bubble Sort, incluso con un número creciente de palabras.

     Específicamente para el caso propuesto de la práctica 0 o en tareas en las que se espera un gran volumen de datos, **Bucket Sort** es la elección adecuada, ofreciendo un rendimiento mejor. **Bubble Sort** debería evitarse en la mayoría de los casos donde la eficiencia sea una prioridad.

6. Video funcionamiento: https://youtu.be/jQbZK8YgUWE