Universidad Central “Marta Abreu” de las Villas

Facultad de Ingeniería Eléctrica

Carrera de Ingeniería en Telecomunicaciones y Electrónica

Asignatura Programación II

Seminario

Comparing algorithms execution time.

Autores:

Carlos Javier Águila, Abraham Fernández, Pedro Addiel Docina, Leandro Emanuel Castellón, Marian Llopiz, Karel Pérez, Javier Alejandro Amador, Bárbara Liset Roque, Victor Manuel Pupo, Lioguel Concepción, Regla Sofía Mursulí, Vladimir Paz, Moisés Obregón, José Antonio Hernández, Ernesto Alejandro Águila

2023

Introducción

La notación Big O es un concepto fundamental en el análisis de algoritmos y la eficiencia de los mismos. Se utiliza para describir el tiempo de ejecución o la complejidad de un algoritmo en términos de la cantidad de datos de entrada, lo cual nos permite comparar algoritmos y determinar cuál es más eficiente en términos de tiempo de ejecución.

En este seminario, nos enfocaremos en dos métodos de ordenamiento: el método de ordenamiento burbuja y el método de ordenamiento por combinación. Estos dos métodos son ampliamente utilizados en la programación y son buenos ejemplos para analizar la eficacia de los algoritmos y la aplicación de la notación Big O.

El método de ordenamiento burbuja es uno de los algoritmos más simples de implementar, pero también uno de los menos eficientes en términos de tiempo de ejecución. Consiste en comparar pares de elementos adyacentes e intercambiarlos si están en el orden incorrecto. Este proceso se repite hasta que la lista esté completamente ordenada. El tiempo de ejecución del método de ordenamiento burbuja está en el orden de O(n^2), lo que significa que su eficiencia disminuye rápidamente a medida que aumenta el tamaño de los datos de entrada.

Por otro lado, el método de ordenamiento Merge sort es mucho más eficiente en términos de tiempo de ejecución. Utiliza un enfoque de divide y vencerás para ordenar la lista. Divide la lista en sub-listas más pequeñas, las ordena y luego las fusiona para obtener la lista ordenada final. El tiempo de ejecución del método de ordenamiento Merge sort está en el orden de O(n log n), lo que lo hace mucho más eficiente que el método de ordenamiento burbuja, especialmente para conjuntos de datos grandes.

En este trabajo, analizaremos la eficacia de estos dos métodos de ordenamiento en términos de su tiempo de ejecución y compararemos sus complejidades utilizando la notación Big O. También exploraremos cómo estas diferencias en eficiencia se traducen en la práctica, mediante la implementación y comparación de ambos algoritmos en un entorno de programación.

Además, discutiremos las aplicaciones y limitaciones de cada método de ordenamiento, así como las situaciones en las que uno puede ser más apropiado que el otro. También consideraremos cómo la elección del método de ordenamiento puede impactar el rendimiento y la eficiencia de un programa o sistema en general.

En resumen, este trabajo se centrará en el análisis comparativo de la notación Big O y la eficacia de los métodos de ordenamiento burbuja y Merge sort. Exploraremos cómo la notación Big O nos permite entender y comparar la eficiencia de los algoritmos, y cómo esta comprensión puede influir en la toma de decisiones sobre qué método de ordenamiento utilizar en diferentes situaciones.

Desarrollo

Notación Big O:

En la notación Big O, se utiliza la letra "O" seguida de una función matemática que representa la complejidad del algoritmo. Por ejemplo, O(n) significa que el tiempo de ejecución del algoritmo aumenta linealmente con el tamaño de los datos de entrada, O(n^2) significa que el tiempo de ejecución aumenta cuadráticamente, y O(log n) significa que el tiempo de ejecución aumenta logarítmicamente.

La notación Big O nos permite identificar rápidamente cómo crece el tiempo de ejecución de un algoritmo a medida que aumenta el tamaño de los datos de entrada.

Para determinar la complejidad de un algoritmo con la notación Big O, se deben seguir una serie de propiedades “aritméticas”, las cuales se exponen a continuación:

1-O(g(n)) * c = O(c * g(n)) = O(g(n))

2-O(g0(n)) * O(g1(n)) = O(g0(n) * g1(n))

3-O(g0(n)) + O(g1(n)) = O(g0(n) + g1(n)) = O(max(g0(n), g1(n))

Además de esta serie de propiedades también es necesario conocer cómo aplicarlas al calcular la complejidad de un algoritmo. Estas reglas de cálculo se describen a continuación:

1-Operaciones básicas (declaración, inicialización, aritmética) - O(1)

2-Bifurcaciones. - O(gif(n) + gelse if 1(n) + ... + gelse if q(n) +gelse(n))

3-Ciclos. - O(gq(n) * gcontent(n)) donde gq(n) es la cantidad de iteraciones en función de n

Conociendo lo expuesto anteriormente se puede comenzar a calcular la complejidad de tus propios algoritmos en búsqueda de la mayor eficacia.

Ordenamiento por burbuja:

El método de ordenamiento burbuja es un algoritmo simple que recorre una lista de elementos varias veces. En cada pasada, compara los elementos adyacentes y los intercambia si están en el orden incorrecto. Este proceso se repite hasta que la lista esté completamente ordenada.

El nombre "burbuja" proviene del hecho de que, durante cada pasada, los elementos más grandes "burbujean" hacia arriba a sus posiciones correctas, de manera similar a cómo las burbujas suben a la superficie del agua.

El método de ordenamiento burbuja es fácil de entender e implementar, pero no es muy eficiente en términos de tiempo de ejecución, especialmente para conjuntos de

datos grandes. Su complejidad es O(n^2), lo que significa que su tiempo de ejecución aumenta cuadráticamente con el tamaño de los datos de entrada.

A pesar de su baja eficiencia, el método de ordenamiento burbuja todavía se utiliza en situaciones donde la simplicidad y la facilidad de implementación son más importantes que la eficiencia. También puede ser útil en conjuntos de datos pequeños o casi ordenados, donde su rendimiento puede ser aceptable.

Ordenamiento Merge sort:

El ordenamiento por combinación es un algoritmo de ordenamiento que sigue el enfoque de "divide y conquista". Funciona dividiendo la lista de elementos en sub-listas más pequeñas, ordenando esas sub-listas y luego fusionándolas para obtener una lista ordenada.

El proceso de Merge sort comienza dividiendo la lista original en dos mitades iguales. Luego, cada mitad se divide nuevamente en mitades más pequeñas, y así sucesivamente, hasta que cada sub-lista contenga solo un elemento. Después, las sub-listas se van fusionando en pares ordenados, y luego esos pares se fusionan en sub-listas ordenadas más grandes. Este proceso se repite hasta que se obtiene una única lista ordenada que contiene todos los elementos.

El Merge sort es eficiente en términos de tiempo de ejecución, ya que tiene una complejidad de O(n log n), lo que significa que su tiempo de ejecución aumenta de forma logarítmica con el tamaño de los datos de entrada. Esto lo hace adecuado para conjuntos de datos grandes.

Aunque el merge sort es más complejo de implementar que el método de ordenamiento burbuja, su eficiencia lo hace una opción popular para ordenar conjuntos de datos grandes. Además, es estable, lo que significa que conserva el orden relativo de elementos con el mismo valor, y puede ser implementado de manera recursiva o iterativa.

Comparación de los tiempos de ejecución:

Para la comparación de los tiempos de ejecución de cada algoritmo, se generaron 12 juegos de datos aleatoriamente de maneras distintas: distribución uniforme, normal, poisson y chi cuadrado.

Cada distribución contiene 3 arreglos de tamaños de 40, 800 y 4000 elementos respectivamente, sobre los cuales se usaron ambos métodos de ordenamiento y se midió el tiempo de ejecución.

Para poder arribar a acertadas conclusiones se decidió realizar las mediciones 5 veces sobre cada juego de datos con ambos algoritmos, para luego hallar un

promedio de tiempo de medición de cada algoritmo por arreglo. Las mediciones realizadas se muestran en las siguientes tablas:

Mediciones de tiempo método burbuja (Primeros 6 arreglos):

Número medición	Arreglo 1	Arreglo 2	Arreglo 3	Arreglo 4	Arreglo 5	Arreglo 6
1	24384	6367077	169083461	14846	6551385	165971077
2	19923	6089462	180984846	16000	6427616	179987385
3	19462	6131308	192082692	15616	6458769	163412230
4	38462	6746385	180800692	15769	6510769	163324308
5	20538	6605231	157907923	15692	6508692	163419077

Mediciones de tiempo método burbuja (Últimos 6 arreglos):

Número medición	Arreglo 7	Arreglo 8	Arreglo 9	Arreglo 10	Arreglo 11	Arreglo 12
1	27692	6049462	151758846	14846	5997461	151490770
2	15384	5938769	149735384	15000	6050769	153178846
3	16000	6065846	156559077	14769	6017923	150635692
4	15077	5987923	149774770	14231	6044000	151551462
5	15461	6074230	149860000	16616	6021616	158424231

Mediciones de tiempo Merge sort (Primeros 6 arreglos):

Número medición	Arreglo 1	Arreglo 2	Arreglo 3	Arreglo 4	Arreglo 5	Arreglo 6
1	9154	215693	1239923	13154	200846	1193923
2	8615	211616	1235615	11308	200846	1233230
3	9307	216231	1244000	6462	200538	1259077
4	9846	203692	1201846	6692	201231	1213846
5	8923	207923	1210154	6538	205462	1220385

Mediciones de tiempo Merge sort (Últimos 6 arreglos):

Número medición	Arreglo 7	Arreglo 8	Arreglo 9	Arreglo 10	Arreglo 11	Arreglo 12
1	6461	200923	1192000	6230	201231	1203692
2	6385	200923	1191616	6461	200615	1197770
3	6461	205846	1196692	6462	200000	1191616
4	6461	200846	1216923	6231	205539	1237000
5	6461	203077	1195923	6154	200539	1192385

También se midió el tiempo total que demoró cada método en ordenar los 12 arreglos:

Mediciones totales:

Número medición	Burbuja	Merge sort
1	663361539	5686385
2	688468539	5707384
3	687437616	5746153
4	670835769	5714077
5	654898539	5667385

Una vez realizadas todas las mediciones, podemos empezar a analizar los datos obtenidos. Empecemos por el método más ineficiente de ambos: el método burbuja. A través, de las siguientes mediciones, se puede observar que mientras más crece la cantidad de elementos del arreglo, el tiempo de ejecución aumenta drásticamente. De antemano ya se conocía la complejidad O(n^2) del algoritmo y era de esperarse que el tiempo de ejecución del algoritmo aumentara bastante. Esto es fácilmente visible a través de la siguiente figura:

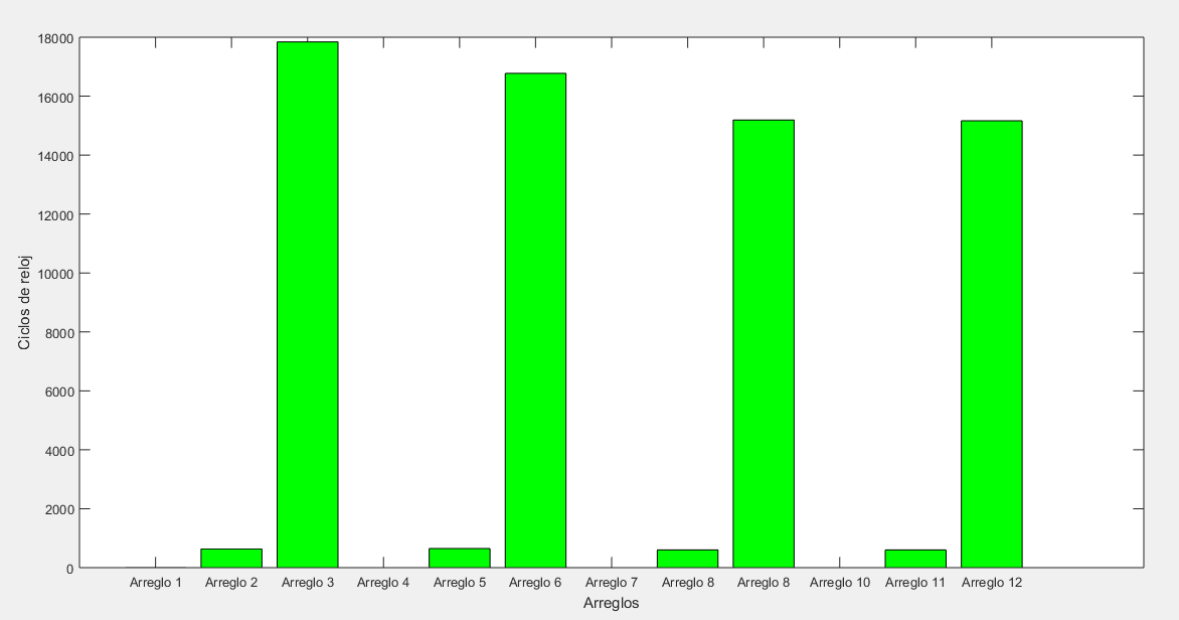

Figura 1.0

Si nos fijamos en la figura 1.0 podemos darnos cuenta mirando el eje y como aumentan, por mucho, los ciclos de reloj, a medida que aumenta la cantidad de elementos. Nótese que para los arreglos de 40 elementos no se aprecia la barra correspondiente, esto es debido a la gran escala del eje y.

Si analizamos, entonces, los datos obtenidos por las mediciones con Merge sort, podemos darnos cuenta que en comparación son bastante más pequeños, teniendo

hasta 2 cifras menos en ciclos de reloj. Nuevamente hagamos esto visible a través de la siguiente figura:

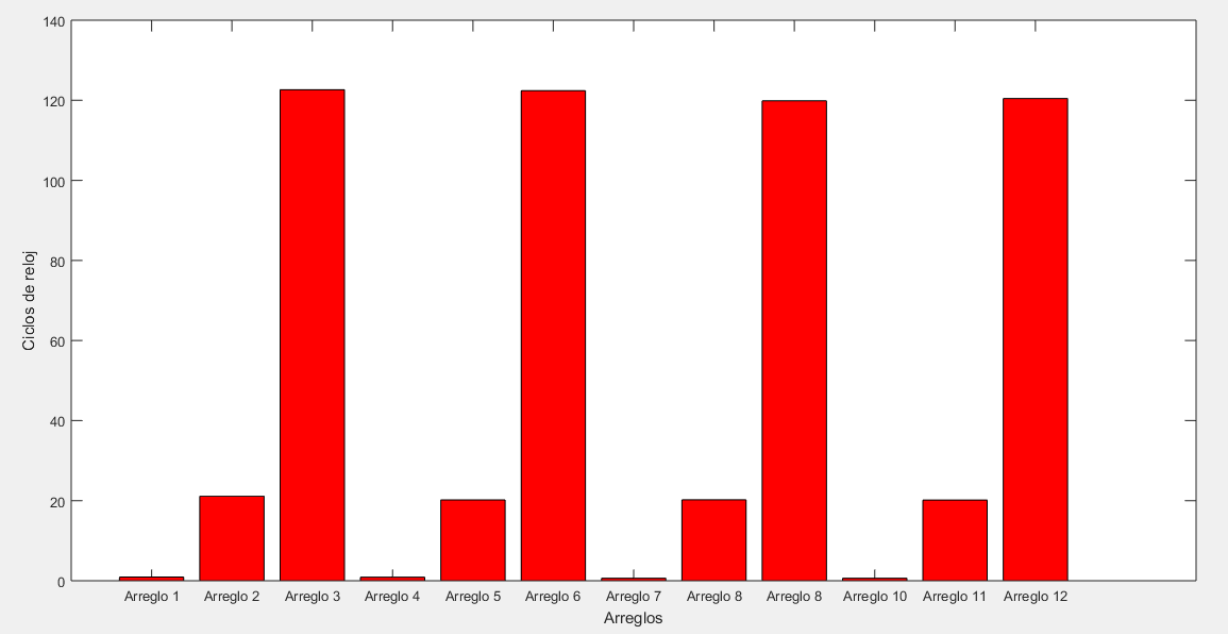

Figura 1.1

En la figura 1.1 se aprecia igualmente que a medida que aumenta la cantidad de elementos también aumenta bastante el tiempo de ejecución, pero, si analizamos la escala del eje y nuevamente podemos notar que los tiempos de ejecución son hasta 100 veces menores cuando se trata de una cantidad considerable de elementos. Nótese que los tiempos de ejecución fueron divididos entre 1000 para trabajar con números más pequeños, ya que al efecto de graficar no afecta.

En la figura 1.2 se puede ver claramente la gran diferencia que existe entre la eficacia de ambos algoritmos:

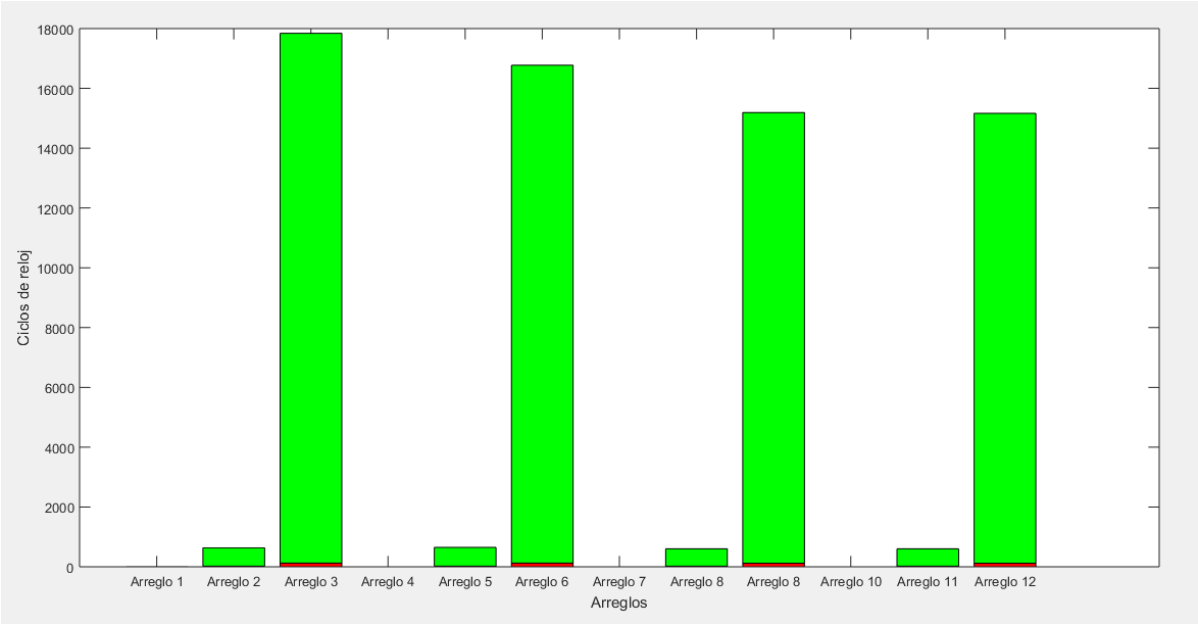

Figura 1.2 Rojo: Merge sort, Verde: Bubble sort

Es tan abismal la diferencia entre los tiempos de ejecución, que las barras que representan los ciclos de reloj del ordenamiento por combinación son casi imperceptibles, debido a que los ciclos del método burbuja son demasiado grandes.

De la figura 1.2 también se puede ver que, para arreglos pequeños, a pesar de existir una diferencia de una cifra en los tiempos, podría ser conveniente por la facilidad de implementación del método burbuja utilizarlo en estos casos sobre el ordenamiento por combinación.

Ahora comparemos uno por uno, los tiempos en porcentaje, de los arreglos de 40, 800 y 4000 elementos. En las figuras 1.3, 1.4 y 1.5 podemos ver los gráficos de los tamaños 40, 800 y 4000 respectivamente:

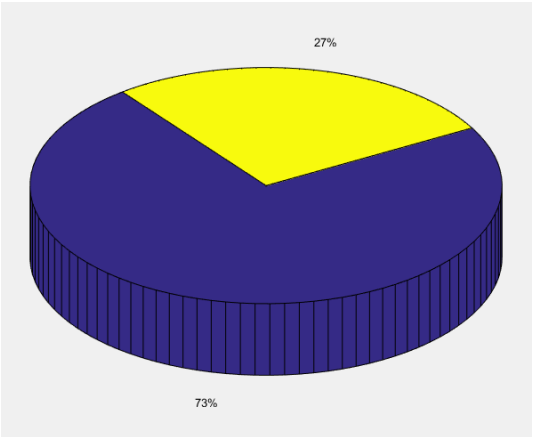

Figura 1.3

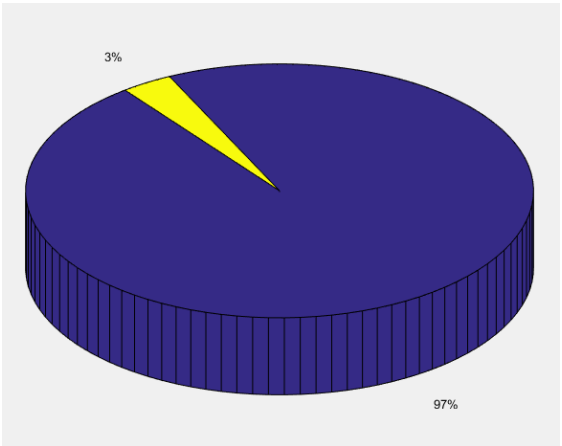

Figura 1.4

Amarillo: Merge sort, Azul: Bubble sort

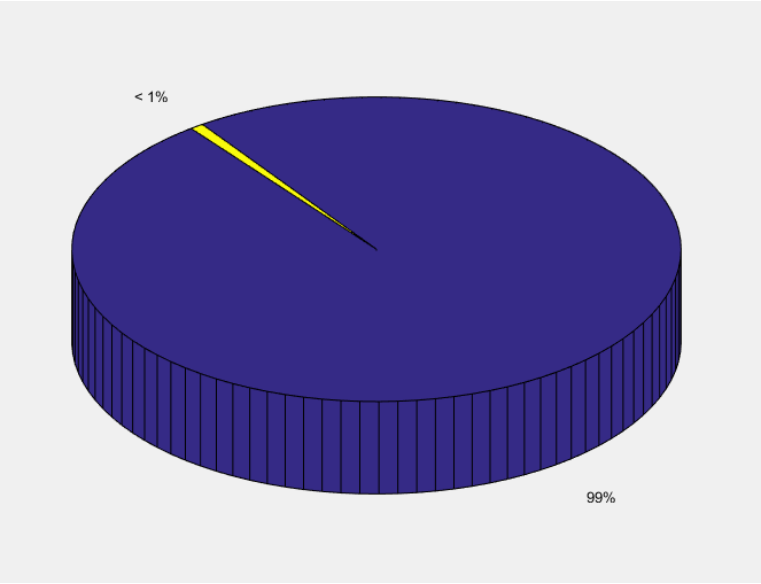

Figura 1.5

En las figuras 1.3, 1.4 y 1.5, se puede ver que solo en arreglos pequeños, a pesar de no existir una diferencia tan marcada, aún es una diferencia abultada, pero nada excesivo, como sí es en los siguientes 2 casos.

También podemos analizar los resultados de las mediciones, graficando los datos obtenidos de la forma cantidad de elementos por tiempo de ejecución. Esto daría como resultado una representación que a inicialmente debería dejar una función cuadrática para el método burbuja y una especie de función logarítmica para el método por combinación.

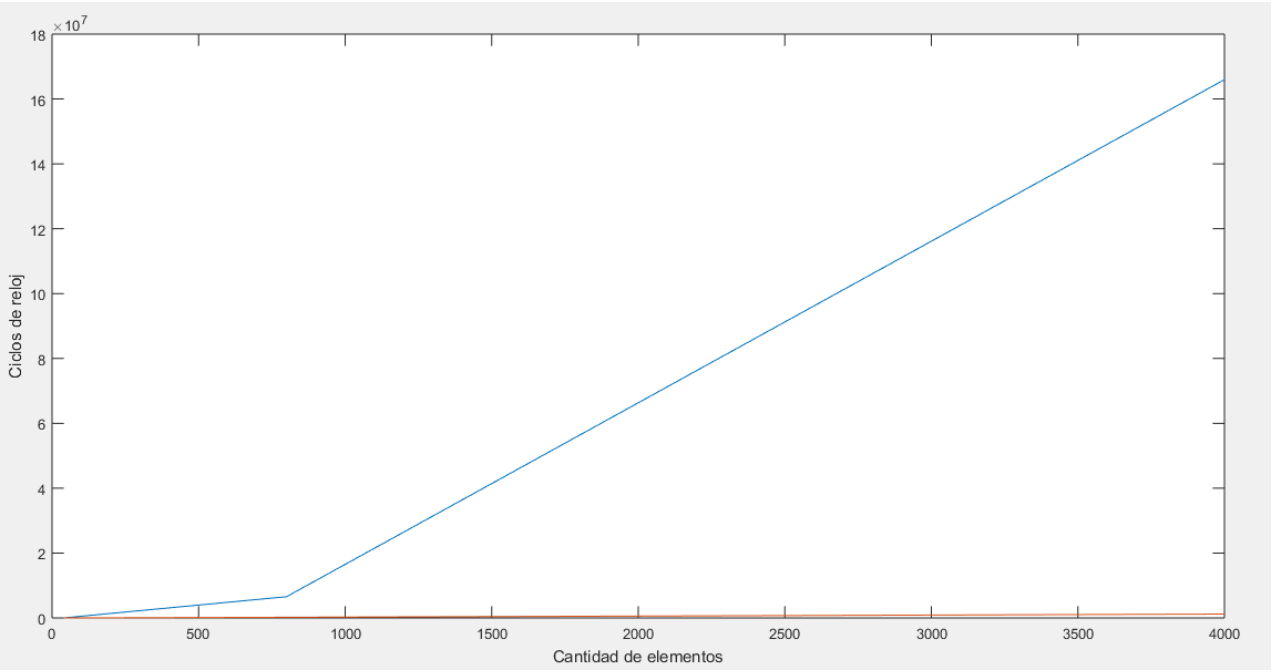

Figura 1.6 Azul: Bubble sort, Naranja: Merge sort

En la figura 1.6 están graficados 3 puntos, para ambas funciones, pero de esa manera no se puede apreciar realmente la forma de la curva.

Si se realizan los ajustes pertinentes a ambas curvas se puede apreciar mejor la forma que tienen estas curvas.

En la figura 1.7 se puede ver como la curva que representa al método burbuja obtiene la forma de una parábola, (la curva del método por combinación no se aprecia correctamente, debido a la inmensa escala del eje y dada por los valores que toman los ciclos de reloj del método burbuja.

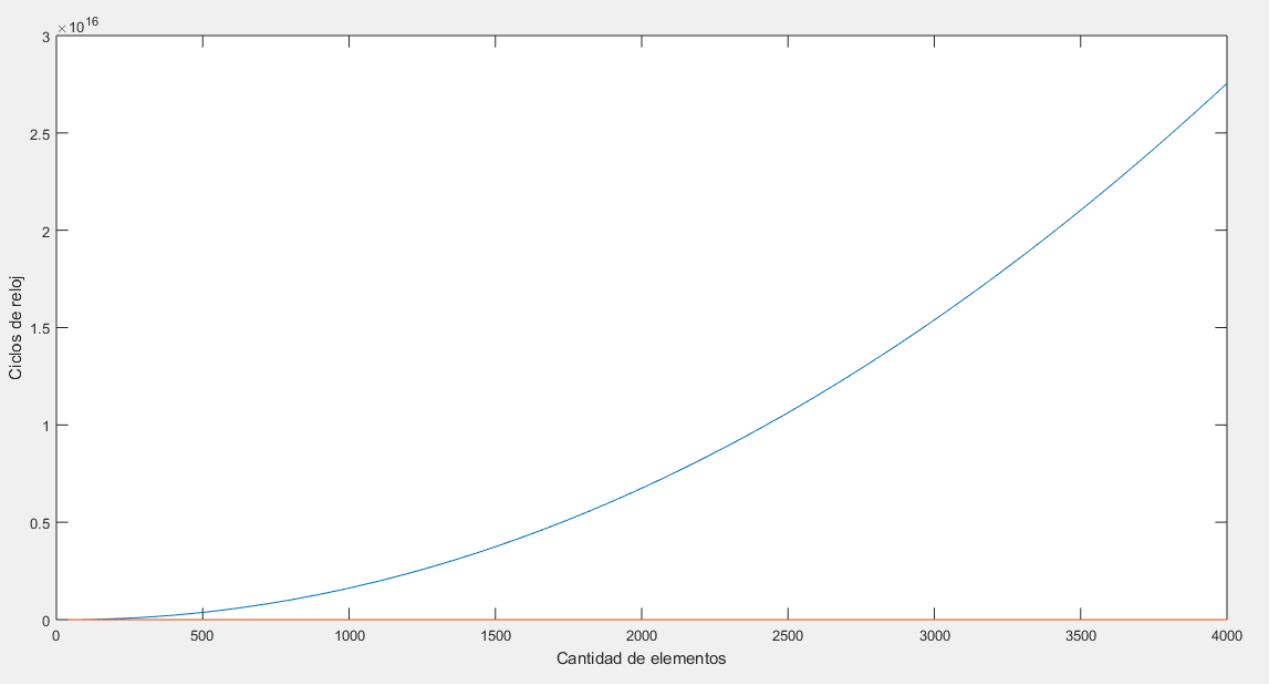

Figura 1.7 Azul: Bubble sort, Naranja: Merge sort

Conclusiones

En conclusión, luego de haber realizado todas las mediciones sobre los diferentes juegos de datos, y haber analizado los resultados podemos recalcar inicialmente 2 puntos importantes:

1-En primer lugar, se observó que el método burbuja mostró un desempeño considerablemente inferior al de merge sort en todos los escenarios analizados. En arreglos pequeños, la diferencia en los tiempos de ejecución fue notable, con merge sort superando consistentemente al método burbuja. Sin embargo, esta disparidad se hizo aún más evidente en arreglos grandes, donde el tiempo de ejecución del método burbuja aumentó exponencialmente, mientras que merge sort mantuvo un desempeño estable y eficiente.

2-Se observó que la eficiencia de cada algoritmo se mantuvo constante independientemente del tamaño del arreglo.

A través de estas observaciones se puede llegar fácilmente a las siguientes conclusiones:

- Siempre el método por combinación será más eficaz, sin embargo, para arreglos de pocos elementos no existe una diferencia muy abultada en los tiempos de ejecución que justifique utilizar el método por combinación antes que el método burbuja. Por esta razón, se puede optar por una implementación más sencilla sin sufrir las consecuencias.

-Para arreglos de muchos elementos, debido a la inmensa diferencia entre ambos métodos, se debe utilizar un método de poca complejidad para evitar tiempos de ejecución que entorpezcan el correcto funcionamiento del programa.

Por último, es necesario siempre conocer la complejidad de un algoritmo, para evitar programas ineficientes, y optar por soluciones a las problemáticas más elegantes.