

# Universidad José Cecilio del Valle

Estructuras de Datos.

Informe Comparativo Bubble sort y Merge sort

**Catedrático:**

Ing. Kenneth Vittetoe

**Alumno:**

Jefry Manuel Cantarero Tercero.

**Número de Cuenta:**

#2016110290

**Fecha:16/10/2019.**

## **Índice.**

<b>1. Introducción.....</b>	<b>pag.1</b>
<b>2. Bubble Sort.....</b>	<b>pag.2</b>
<b>2.1 Aplicación y 2.2 Algoritmo de Bubble Sort.....</b>	<b>pag.3</b>
<b>3. Merge Sort y 3.1 Pasos.....</b>	<b>pag.4</b>
<b>3.2 Aplicación y 3.3 Algoritmo de Merge Sort.....</b>	<b>pag.5</b>
<b>4. Cual algoritmo tiene mejor Desempeño.....</b>	<b>pag.6 y pag.7</b>
<b>5. Conclusiones.....</b>	<b>pag.8</b>
<b>6. Bibliografía.....</b>	<b>pag.9</b>

## **1. Introducción.**

El ordenamiento de datos es algo fundamental para tener la facilidad al momento de buscar o controlar los registros, por ejemplo: la agenda de contacto que tenemos en el celular y también en el control de las llamadas telefónicas que se registran mediante hora y fecha de ejecución, y como último ejemplo el caso de las universidades ordena sus datos por apellido alfabéticamente o mediante el número de cuenta. Existen dos algoritmos muy conocidos que pueden realizar estas funciones los cuales son el ordenamiento burbuja (Bubble Sort) y el ordenamiento por mezcla (Merge Sort).

El siguiente informe detallará cada algoritmo con su respectiva definición, aplicación y el código de este en el lenguaje Java. De igual manera se muestra una comparación entre ambos algoritmos con arreglos de diferentes tamaños para cotejar los tiempos de ejecución de los algoritmos mencionados anteriormente

## 2. Bubble Sort “Algoritmo Burbuja.”

El **ordenamiento burbuja** hace múltiples pasadas a lo largo de una lista. Compara los ítems adyacentes e intercambia los que no están en orden. Cada pasada a lo largo de la lista ubica el siguiente valor más grande en su lugar apropiado. En esencia, cada ítem “burbujea” hasta el lugar al que pertenece.

El siguiente ejemplo muestra el proceso de forma gráfica:

''' Variables '''				''' Vector '''								
				pos	0	1	2	3	4	5	6	7
i	j	a[j]	a[j+1]	inicio	44	55	12	42	94	18	6	67
0	1	55	12	cambio	44	12	55	42	94	18	6	67
0	2	55	42	cambio	44	12	42	55	94	18	6	67
0	4	94	18	cambio	44	12	42	55	18	94	6	67
0	5	94	6	cambio	44	12	42	55	18	6	94	67
0	6	94	67	cambio	44	12	42	55	18	6	67	94
1	0	44	12	cambio	12	44	42	55	18	6	67	94
1	1	44	42	cambio	12	42	44	55	18	6	67	94
1	3	55	18	cambio	2	42	44	18	55	6	67	94
1	4	55	6	cambio	12	42	44	18	6	55	67	94
2	2	44	18	cambio	12	42	18	44	6	55	67	94

Una de las deficiencias del algoritmo es que ya cuando ha ordenado parte del vector vuelve a compararlo cuando esto ya no es necesario. Dado un vector a1, a2, a3, ... an-1) Comparar a1 con a2 e intercambiarlos si a1>a2 2) Seguir hasta que todo se haya comparado an-1 con an3) Repetir el proceso anterior n-1 veces

- El ordenamiento de burbuja tiene una complejidad  $\Omega(n^2)$

Vector original

45	17	23	67	21
----	----	----	----	----

Iteración 1:

45	17	23	67	21	Se genera cambio
17	45	23	67	21	Se genera cambio
17	23	45	67	21	No hay cambio
17	23	45	67	21	Se genera cambio
17	23	45	21	67	Fin primera iteración

d

## 2.1 Aplicación.

A pesar de que el ordenamiento de burbuja es uno de los algoritmos más sencillos de implementar, su orden  $O(n^2)$  lo hace muy ineficiente para usar en listas que tengan más que un número reducido de elementos. Incluso entre los algoritmos de ordenamiento de orden  $O(n^2)$ , otros procedimientos como el ordenamiento por inserción son considerados más eficientes. Dada su simplicidad, el ordenamiento de burbuja es utilizado para introducir el concepto de algoritmo de ordenamiento para estudiantes de ciencias de la computación. A pesar de esto, algunos investigadores como Owen Astrachan han criticado su popularidad en la enseñanza de ciencias de la computación, llegando a recomendar su eliminación de los planes de estudio. Sumado a esto, Jargon File, un libro ampliamente citado en la cultura hacker, lo denomina "el mal algoritmo genérico", y Donald Knuth, uno de los mayores expertos en ciencias de la computación, afirma que el ordenamiento de burbuja "no parece tener nada para recomendar su uso, a excepción de un nombre pegajoso y el hecho de que conlleva a problemas teóricos interesantes". El ordenamiento de burbuja es asintóticamente equivalente, en tiempos de ejecución, con el ordenamiento por inserción en el peor de los casos, pero ambos algoritmos difieren principalmente en la cantidad de intercambios que son necesarios. Resultados experimentales como los descubiertos por Astrachan han demostrado que el ordenamiento por inserción funciona considerablemente mejor incluso con listas aleatorias. Por esta razón, muchos libros de algoritmos modernos evitan usar el ordenamiento de burbuja, reemplazándolo por el ordenamiento por inserción. El ordenamiento de burbuja interactúa vagamente con el hardware de las CPU modernas. Requiere al menos el doble de escrituras que el ordenamiento por inserción, el doble de pérdidas de cache, y asintóticamente más predicción de saltos. Varios experimentos de ordenamiento de cadenas en Java hechos por Astrachan muestran que el ordenamiento de burbuja es 5 veces más lento que el ordenamiento por inserción, y 40% más lento que el ordenamiento por selección.

## 2.2 Algoritmo del Ordenamiento Burbuja en Java.

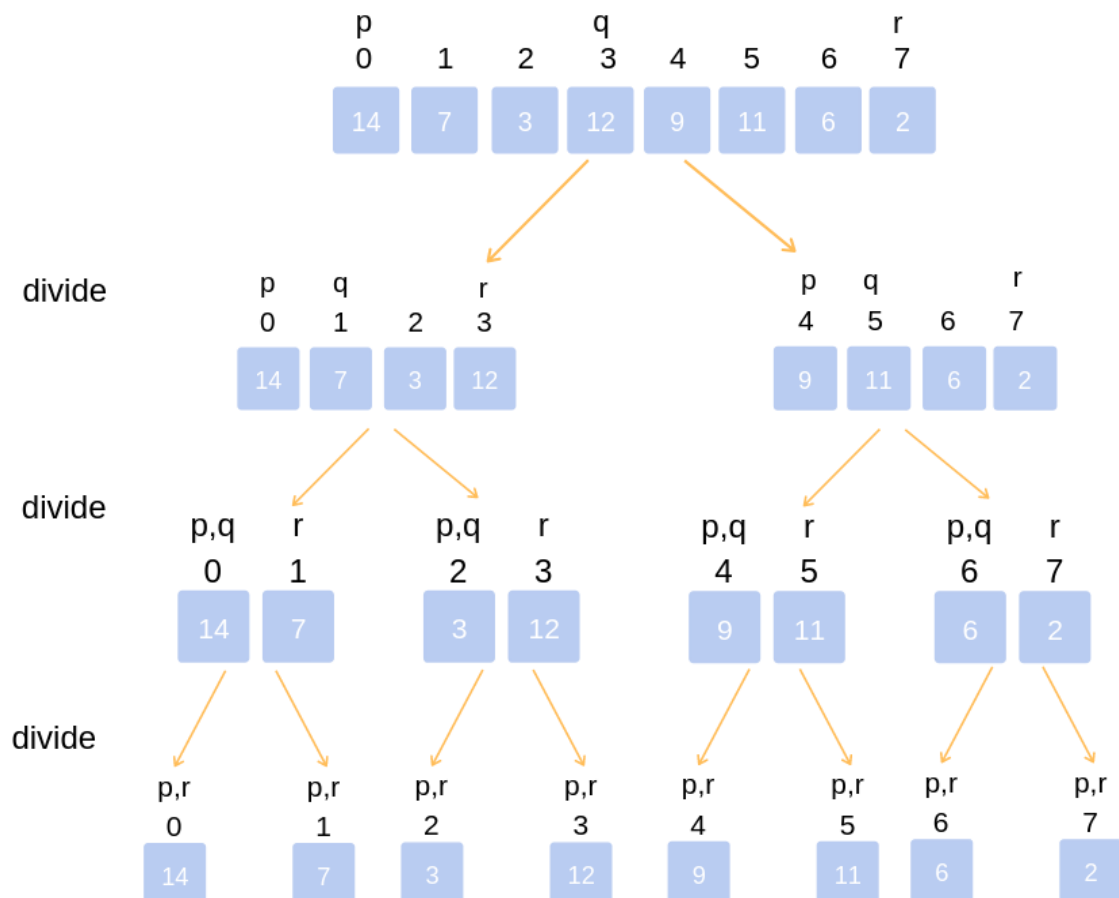
```
//Ordenamiento por Burbuja
Public int[] OrdenarBurbuja(int[] n){
    int temp;
    int t = n.length;
    for (int i = 1; i < t; i++) {
        for (int k = t- 1; k >= i; k--) {
            if(n[k] < n[k-1]){
                temp = n[k];
                n[k] = n[k-1];
                n[k-1]= temp;
            }//fin if
        }// fin 2 for
    }//fin 1 for
    return n;
}//fin
```

### 3. Merge Sort “Ordenamiento por Mezcla.”

**Merge sort.** Algoritmo basado en la técnica de diseño de algoritmos Divide y Vencerás, Consiste en dividir el problema a resolver en subproblemas del mismo tipo que a su vez se dividirán, mientras no sean suficientemente pequeños o triviales.

#### 3.1 Pasos.

1. Ordenar una secuencia S de elementos:
  1. Si S tiene uno o ningún elemento, está ordenada
  2. Si S tiene al menos dos elementos se divide en dos secuencias S1 y S2, S1 conteniendo los primeros  $n/2$ , y S2 los restantes.
  3. Ordenar S1 y S2, aplicando recursivamente este procedimiento.
  4. Mezclar S1 y S2 ordenadamente en S
2. Mezclar dos secuencias ordenadas S1 y S2 en S:
  1. Se tienen referencias al principio de cada una de las secuencias a mezclar (S1 y S2).
  2. Mientras en alguna secuencia queden elementos, se inserta en la secuencia resultante (S) el menor de los elementos referenciados y se avanza esa referencia una posición



### 3.2 Aplicación

Sirven para correr cintas magnéticas como dispositivos I/O ya que requiere, muy poca memoria.

Los algoritmos de ordenamiento de mezcla permitieron a juegos de datos grandes para ser clasificados para los ordenadores de la antigüedad que tenían poca memoria de acceso arbitrarias por normas modernas. Los registros fueron almacenados sobre la cinta magnética y procesados sobre los bancos de unidades de cinta magnética magnéticas, como esta IBM 729s.

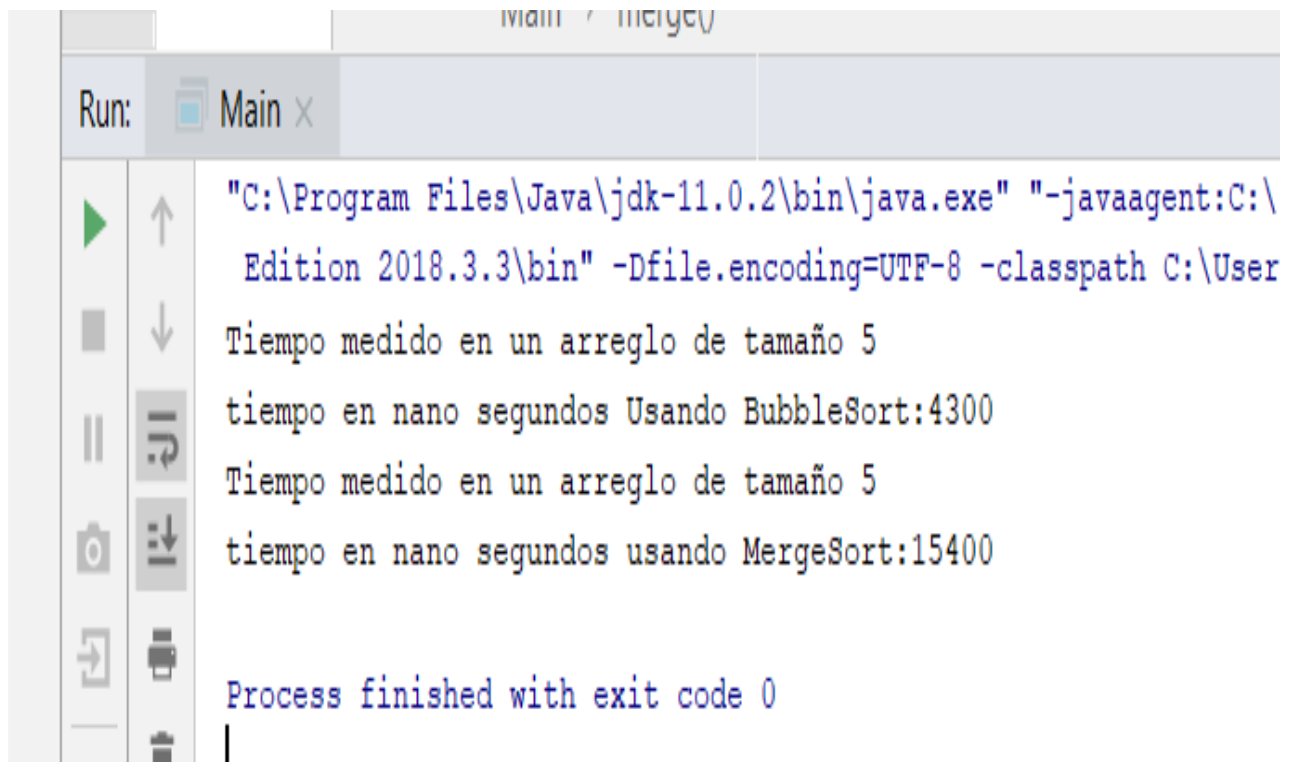
### 3.3 Algoritmo de Merge Sort.

## Merge Sort

```
Inicio combina( A : vector ; B : vector )
Crear Vector C con longitudes de A y B
I ← 1
J ← 1
K ← 1
Mientras (I < longitud A) y (J < longitud B)
  hacer
    Si A[ I ] < B[ J ] entonces
      C[ K ] ← A[ I ]
      K ← K + 1
      I ← I + 1
    de lo contrario
      C[ K ] ← B[ J ]
      K ← K + 1
      J ← J + 1
  fin hacer
```

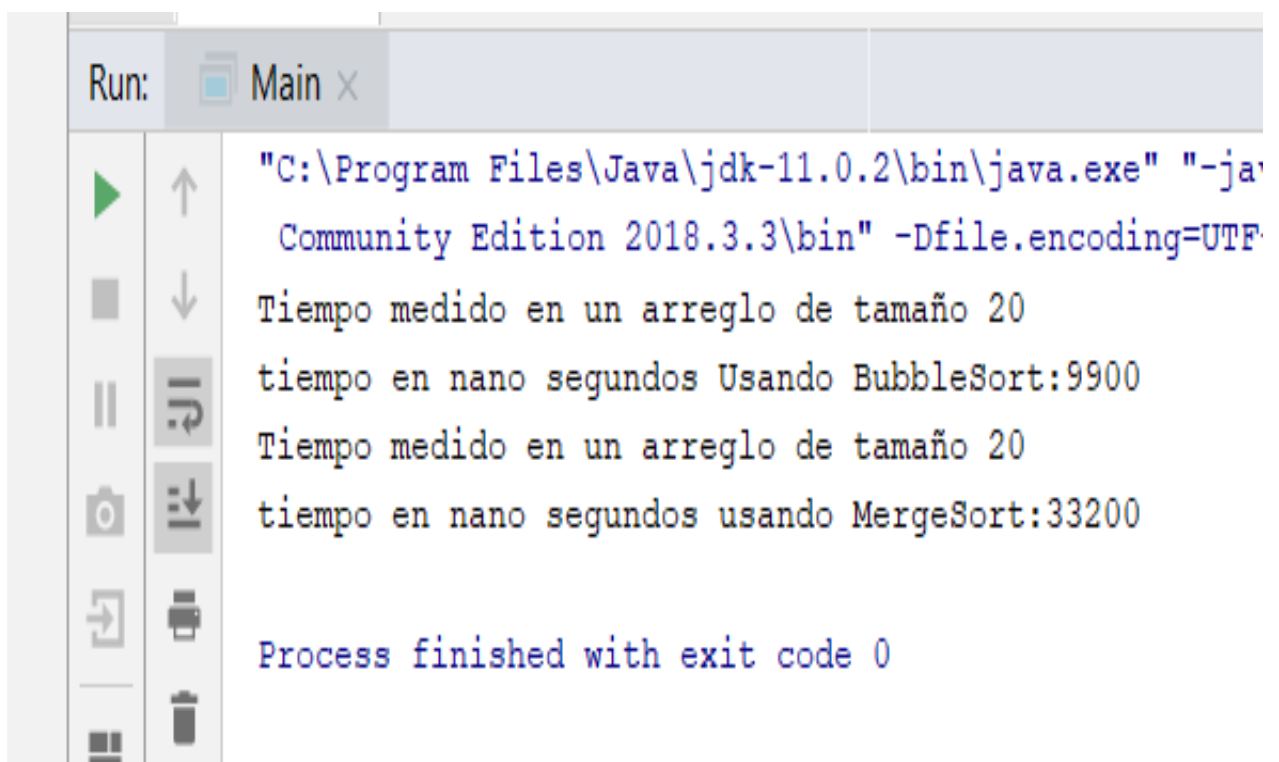
```
Mientras (I < longitud A) hacer
  C[ K ] ← A[ I ]
  K ← K + 1
  I ← I + 1
Mientras (J < longitud B) hacer
  C[ K ] ← B[ J ]
  K ← K + 1
  J ← J + 1
Fin combina
```

#### 4. Cual Algoritmo tiene Mejor Desempeño.



```
Run: Main x
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" "-javaagent:C:\
Edition 2018.3.3\bin" -Dfile.encoding=UTF-8 -classpath C:\User
Tiempo medido en un arreglo de tamaño 5
tiempo en nano segundos Usando BubbleSort:4300
Tiempo medido en un arreglo de tamaño 5
tiempo en nano segundos usando MergeSort:15400

Process finished with exit code 0
```



```
Run: Main x
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" "-ja
Community Edition 2018.3.3\bin" -Dfile.encoding=UTF
Tiempo medido en un arreglo de tamaño 20
tiempo en nano segundos Usando BubbleSort:9900
Tiempo medido en un arreglo de tamaño 20
tiempo en nano segundos usando MergeSort:33200

Process finished with exit code 0
```

Podemos ver que en un arreglo de 5 y 20 el Bubble Sort se Ejecuta en menor tiempo que el Merge Sort.



```
Run: Main x
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" "-javaag
Community Edition 2018.3.3\bin" -Dfile.encoding=UTF-8 -
Tiempo medido en un arreglo de tamaño 200
tiempo en nano segundos Usando BubbleSort:781400
Tiempo medido en un arreglo de tamaño 200
tiempo en nano segundos usando MergeSort:162300

Process finished with exit code 0
```

```
Run: Main x
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" "-javaag
Community Edition 2018.3.3\bin" -Dfile.encoding=UTF-8 -
Tiempo medido en un arreglo de tamaño 400
tiempo en nano segundos Usando BubbleSort:1677700
Tiempo medido en un arreglo de tamaño 400
tiempo en nano segundos usando MergeSort:344400

Process finished with exit code 0
```

**En comparación de un arreglo de 200 y 400 podemos observar que el Merge Sort se ejecuta en menor tiempo que el Bubble Sort.**

## 5. Conclusiones.

El ordenamiento por mezcla o también conocido como Merge Sort es el mejor algoritmo de ordenamiento entre estos dos, ya que el Merge Sort entre Mas Grande es el arreglo menor tiempo tarda en comparación del Bubble Sort que absorbe mayor tiempo para ordenar.

A pesar de que el Bubble Sort es el primer algoritmo de ordenamiento que enseñan en las clases de las ciencias de computación no significa que es el mejor. El Bubble Sort no debería ser enseñado ya que al final en la programación casi no es muy usado porque en la vida real los arreglos contienen miles de datos por lo tanto un algoritmo que su impacto crece  $O(n^2)$  no se recomienda en lo absoluto.

## 6. Bibliografía (Ecured, 2019)

academy, B. S. (13 de 10 de 2019). *runestone.academy*. Obtenido de <https://runestone.academy/runestone/static/pythoned/SortSearch/ElOrdenamientoBurbuja.html>

*Bubble Sort ecured*. (s.f.). Obtenido de [https://www.ecured.cu/Ordenamiento\\_de\\_burbuja](https://www.ecured.cu/Ordenamiento_de_burbuja)

Ecured, M. s. (13 de 10 de 2019). *Ecured*. Obtenido de [https://www.ecured.cu/MergeSort#Complejidad\\_del\\_algoritmo](https://www.ecured.cu/MergeSort#Complejidad_del_algoritmo)

<https://github.com/tercero97/Informe-Comparativo-Bubble-sort-Merge-sort>