

# Análisis y Diseño de Algoritmos

Primer Parcial: Insertion Sort, Bubble Sort, Merge Sort

**Profesor Rodrigo Soulé de Castro**

38192666

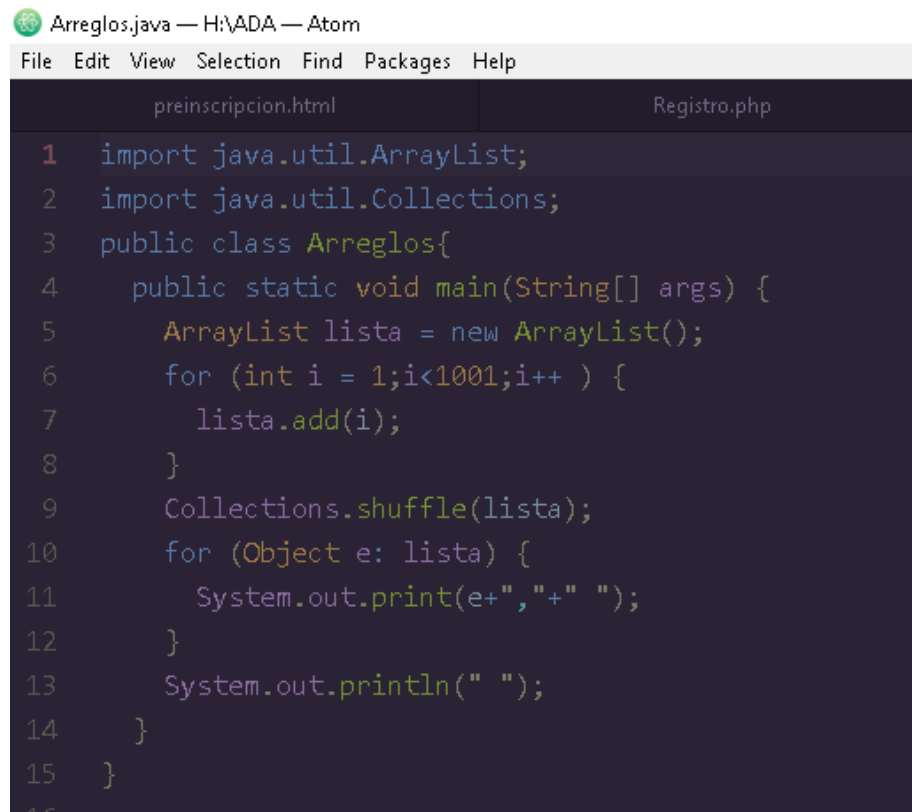
José Emmanuel Sandoval Sánchez

## Contenido

Creación de Array con Java .....	2
Insertion Sort .....	3
Código Fuente .....	3
Ejecución .....	4
Bubble Sort .....	5
Código fuente .....	5
Ejecución .....	6
Merge Sort.....	7
Código Fuente.....	7
Ejecución .....	9
Resultados obtenidos.....	10
Tabla y grafico de Insertion Sort .....	10
Tabla y grafico de Bubble Sort .....	11
Tabla y Grafico de Merge Sort .....	12
Comparación de algoritmos .....	13
Conclusiones.....	14

## Creación de Array con Java

Antes de comenzar a utilizar cualquier algoritmo primero necesite crear arrays, por lo cual cree un pequeño código usando Java

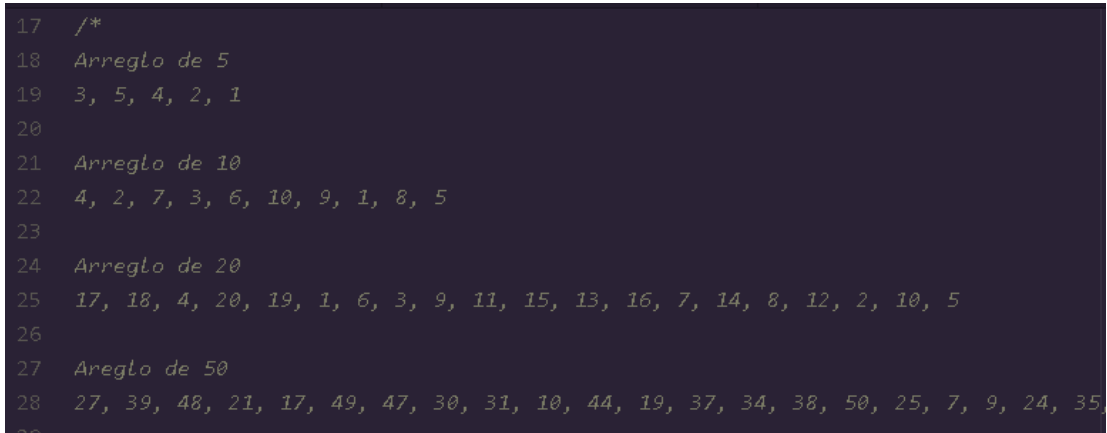


```
Arreglos.java — H:\ADA — Atom
File Edit View Selection Find Packages Help

preinscripcion.html Registro.php

1 import java.util.ArrayList;
2 import java.util.Collections;
3 public class Arreglos{
4     public static void main(String[] args) {
5         ArrayList lista = new ArrayList();
6         for (int i = 1;i<1001;i++ ) {
7             lista.add(i);
8         }
9         Collections.shuffle(lista);
10        for (Object e: lista) {
11            System.out.print(e+","+ " ");
12        }
13        System.out.println(" ");
14    }
15 }
```

Creo el array, después lo mezclo con shuffle para después imprimirlo para copiarlo al algoritmo. Los arrays los guarde en el mismo código de los algoritmos por cuestión de comodidad



```
17  /*
18  Arreglo de 5
19  3, 5, 4, 2, 1
20
21  Arreglo de 10
22  4, 2, 7, 3, 6, 10, 9, 1, 8, 5
23
24  Arreglo de 20
25  17, 18, 4, 20, 19, 1, 6, 3, 9, 11, 15, 13, 16, 7, 14, 8, 12, 2, 10, 5
26
27  Arreglo de 50
28  27, 39, 48, 21, 17, 49, 47, 30, 31, 10, 44, 19, 37, 34, 38, 50, 25, 7, 9, 24, 35,
29
```

# Insertion Sort

## Código Fuente

InsertionSortExample.java — H:\ADA — Atom

File Edit View Selection Find Packages Help

```
preinscripcion.html Registro.php Telemetry Consent Welco

1 public class InsertionSortExample {//Abre Clase
2     static int contador=0;
3     public static void main(String a[]){//Abre Main
4         int[] arr1 = {};
5         System.out.println("Antes de Insertion Sort");
6         for(int i:arr1){
7             System.out.print(i+ " ");
8         }
9         System.out.println();
10
11         insertionSort(arr1);//sorting array using insertion sort
12
13         System.out.println("Despues de Insertion Sort");
14         for(int i:arr1){
15             System.out.print(i+ " ");
16         }
17         System.out.println();
18         System.out.println(contador + " instrucciones");
19     }//Cierra Main
20
21     public static void insertionSort(int array[]) {
22         int n = array.length; //Se saca el tamaño del arreglo dado y se pone en n
23         for (int j = 1; j < n; j++) { //Con este for recorremos el arreglo
24             int key = array[j]; // Aquí guardamos el segundo numero del array,
25                                 //osea el que esta en la segunda posicion
26                                 //puesto que j = 1
27
28             int i = j-1; // i sera siempre un numero menos que j
29             while ( (i > -1) && ( array [i] > key ) ) { //Mientras i sea mayor a -1 y array[i]
30                                                         //(que seria una posicion anterior a la po
31                                                         //sea mayor a key se ejecutaran lo siguien
32                 array [i+1] = array [i]; //Aquí copiamos el numero que estaba en la posicion i a l
33                 contador+=1;
34                 i--; //decrementamos i
35             }
36             array[i+1] = key; //una vez terminado el while se pone key en array[i+1], que se supon
37                             //hay que recordar que i se fue decrementando en while
38             contador+=1;
39         }
40     }//Cierra Clase
41 }
```

## Ejecución

Como primer paso, compilo el programa para después ejecutarlo.

```
em-geek@emgeek-HP-240-G6-Notebook-PC: ~/Documentos
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
em-geek@emgeek-HP-240-G6-Notebook-PC:~/Documentos/Laboratorio/ADA$ javac InsertionSortExample.java
em-geek@emgeek-HP-240-G6-Notebook-PC:~/Documentos/Laboratorio/ADA$ java InsertionSortExample
```

Después muestro la cadena antes de ser ordenada

```
Antes de Insertion Sort
26 201 210 12 266 99 118 154 177 92 223 221 90 5 258
127 138 300 175 182 270 98 48 273 59 264 148 135 102
24 176 107 86 108 253 10 116 249 94 65 288 214 251 15
4 188 93 70 145 153 78 142 279 167 151 282 259 216 16
9 2 128 30 245 169 181 229 256 204 271 208 25 113 274
74 268 225 62 233 53 17 7 286 262 191 241 160 212 27
195 243 295 209 297 44 60 180 40 179 11 172 150 277 8
```

Para al final mostrar la cadena ordenada más el número de instrucciones

```
Despues de Insertion Sort
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
0 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86
2 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132
155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174
197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
39 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258
1 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300
22330 instrucciones
```

Para este ejemplo use un array de 300 numeros

# Bubble Sort

## Código fuente

```
1 public class BubbleSortExample {
2     static int contador = 0;
3     public static void main(String[] args) {};
4         System.out.println("Array Before Bubble Sort");
5         for(int i=0; i < arr.length; i++){
6             System.out.print(arr[i] + " ");
7         }
8         System.out.println();
9         bubbleSort(arr);
10        System.out.println("Array After Bubble Sort");
11        for(int i=0; i < arr.length; i++){
12            System.out.print(arr[i] + " ");
13        }
14        System.out.println("");
15        System.out.println(contador + " instrucciones");
16    }
17    static void bubbleSort(int[] arr) {
18        int n = arr.length; //Se saca el tamaño del arreglo dado y se pone en n
19        int temp = 0; //Se crea la variable temp que sera de utilidad mas adelante
20        for(int i=0; i < n; i++){ //Este for hara que se recorra numero por numero
21            for(int j=1; j < (n-i); j++){ //Este for sirve para poder mover los numeros
22                if(arr[j-1] > arr[j]){
23                    //Aqui se cambian los valores usando una variable auxiliar
24                    temp = arr[j-1];
25                    arr[j-1] = arr[j];
26                    contador+=1;
27                    arr[j] = temp;
28                    contador+=1;
29                }
30            }
31        }
32    }
```

BubbleSortExample.java 22:49

## Ejecución

Como el algoritmo anterior, ya que estoy trabajando con Java, lo primero es compilar para después ejecutar

```
em-geek@emgeek-HP-240-G6-Notebook-PC:~/Documentos/Laboratorio/ADA$ javac BubbleSortExample.java
em-geek@emgeek-HP-240-G6-Notebook-PC:~/Documentos/Laboratorio/ADA$ java BubbleSortExample
```

Primero imprimo el array sin ordenar.


```
Array Before Bubble Sort
26 201 210 12 266 99 118 154 177 92 223 221 90 5 258 194 110 152 220 41 163
127 138 300 175 182 270 98 48 273 59 264 148 135 102 35 37 217 57 15 73 294
24 176 107 86 108 253 10 116 249 94 65 288 214 251 155 260 143 69 281 68 106
4 188 93 70 145 153 78 142 279 167 151 282 259 216 165 47 170 54 64 147 239
9 2 128 30 245 169 181 229 256 204 271 208 25 113 274 215 272 131 46 67 207
74 268 225 62 233 53 17 7 286 262 191 241 160 212 27 129 120 298 63 14 186
195 243 295 209 297 44 60 180 40 179 11 172 150 277 81 234 23 1 296 109 20 8
```

Para después imprimir el array ya ordenado junto con su número de instrucciones.

```
Array After Bubble Sort
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
0 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78
2 113 114 115 116 117 118 119 120 121 122 123 124 125 126
155 156 157 158 159 160 161 162 163 164 165 166 167 168
197 198 199 200 201 202 203 204 205 206 207 208 209 210
39 240 241 242 243 244 245 246 247 248 249 250 251 252 253
1 282 283 284 285 286 287 288 289 290 291 292 293 294 295
44062 instrucciones
```

# Merge Sort

## Código Fuente

 MergeSortExample.java — H:\ADA — Atom

```
File Edit View Selection Find Packages Help

preinscripcion.html Registro.php Telemetry Cons...

1  /* Java program for Merge Sort */
2  class MergeSort {
3      static int contador = 0;
4      public static void main(String args[]) {
5          int arr[] = {};
6          System.out.println("Array dada");
7          printArray(arr);
8          MergeSort ob = new MergeSort();
9          ob.sort(arr, 0, arr.length-1);
10         System.out.println("\nArray Ordenado");
11         printArray(arr);
12         System.out.println(contador + " instrucciones");
13     }
14     void merge(int arr[], int l, int m, int r) {
15         int n1 = m - l + 1;
16         int n2 = r - m;
17         int L[] = new int [n1];
18         int R[] = new int [n2];
19         contador+=2;
20         for (int i=0; i<n1; ++i)
21             L[i] = arr[l + i];
22         for (int j=0; j<n2; ++j)
23             R[j] = arr[m + 1+ j];
24         int i = 0, j = 0;
25         int k = l;
26         while (i < n1 && j < n2) {
27             if (L[i] <= R[j]) {
28                 arr[k] = L[i];
29                 i++;
30             }
31             else {
32                 arr[k] = R[j];
```



```

33         j++;
34     }
35     k++;
36 }
37 while (i < n1) {
38     arr[k] = L[i];
39     i++;
40     k++;
41 }
42 while (j < n2) {
43     arr[k] = R[j];
44     j++;
45     k++;
46 }
47 }
48 void sort(int arr[], int l, int r)
49 {
50     if (l < r) {
51         int m = (l+r)/2;
52         sort(arr, l, m);
53         sort(arr, m+1, r);
54
55         merge(arr, l, m, r);
56         contador+=2;
57     }
58 }
59 static void printArray(int arr[])
60 {
61     int n = arr.length;
62     for (int i=0; i<n; ++i)
63         System.out.print(arr[i] + " ");
64     System.out.println();
65 }

```

MergeSortExample.java\* 47:4

## Ejecución

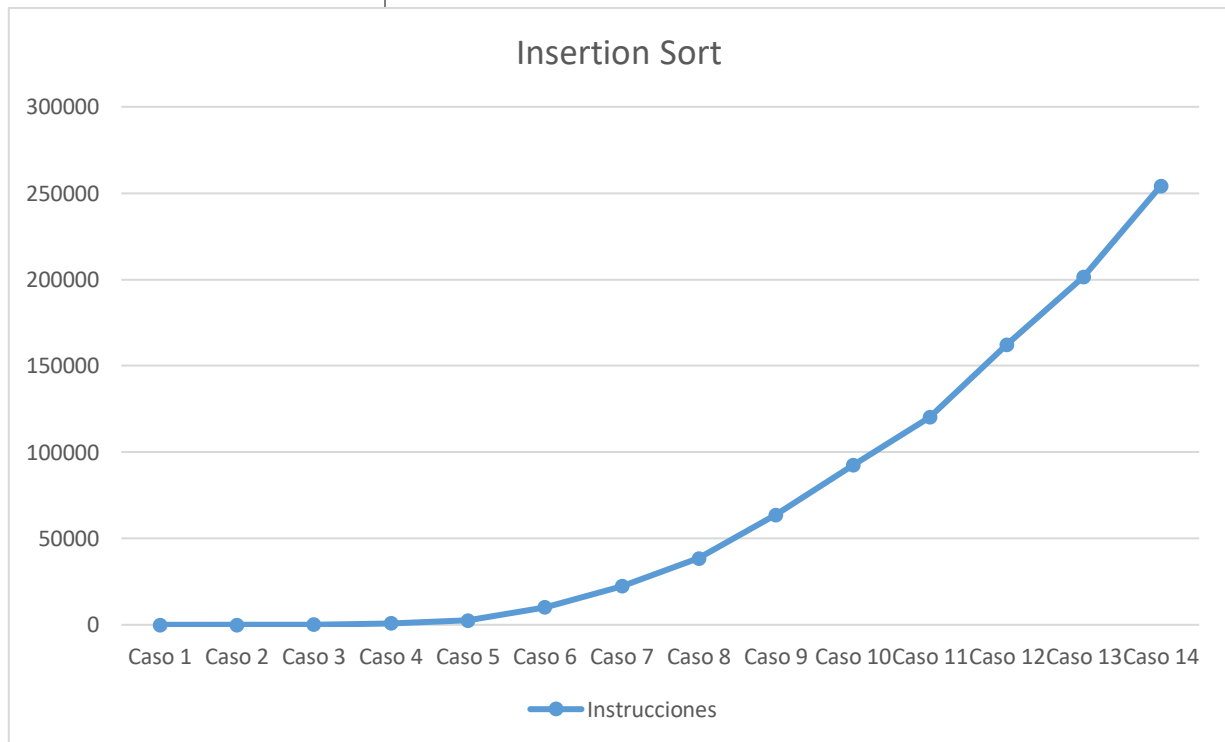
```
em-geek@emgeek-HP-240-G6-Notebook-PC:~/Documentos/Laboratorio/ADA$ javac MergeSortExample.java
em-geek@emgeek-HP-240-G6-Notebook-PC:~/Documentos/Laboratorio/ADA$ java MergeSort
Array dada
4 2 7 3 6 10 9 1 8 5
36 instrucciones
Array Ordenado
1 2 3 4 5 6 7 8 9 10
36 instrucciones
em-geek@emgeek-HP-240-G6-Notebook-PC:~/Documentos/Laboratorio/ADA$ javac MergeSortExample.java
em-geek@emgeek-HP-240-G6-Notebook-PC:~/Documentos/Laboratorio/ADA$ java MergeSort
Array dada
17 18 4 20 19 1 6 3 9 11 15 13 16 7 14 8 12 2 10 5
76 instrucciones
Array Ordenado
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
76 instrucciones
em-geek@emgeek-HP-240-G6-Notebook-PC:~/Documentos/Laboratorio/ADA$ javac MergeSortExample.java
em-geek@emgeek-HP-240-G6-Notebook-PC:~/Documentos/Laboratorio/ADA$ java MergeSort
Array dada
27 39 48 21 17 49 47 30 31 10 44 19 37 34 38 50 25 7 9 24 35 45 18 42 20 12 23 36 11 16 13 29 40
196 instrucciones
Array Ordenado
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
196 instrucciones
em-geek@emgeek-HP-240-G6-Notebook-PC:~/Documentos/Laboratorio/ADA$ javac MergeSortExample.java
em-geek@emgeek-HP-240-G6-Notebook-PC:~/Documentos/Laboratorio/ADA$ java MergeSort
Array dada
53 82 79 78 57 10 8 65 51 73 41 88 24 18 89 35 59 76 6 58 29 61 100 26 62 54 71 3 48 33 95 5 21 42
396 instrucciones
Array Ordenado
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
396 instrucciones
em-geek@emgeek-HP-240-G6-Notebook-PC:~/Documentos/Laboratorio/ADA$ javac MergeSortExample.java
em-geek@emgeek-HP-240-G6-Notebook-PC:~/Documentos/Laboratorio/ADA$ java MergeSort
Array dada
77 124 119 135 40 116 9 63 65 178 104 60 22 137 36 118 182 83 72 192 35 141 115 46 52 120 13 41 11
1 70 21 109 1 129 143 149 183 130 151 78 131 31 140 39 181 29 156 174 127 80 162 105 128 48 191 2
```

## Resultados obtenidos

### Tabla y grafico de Insertion Sort

Grado de complejidad:  $T(n) = n^2$

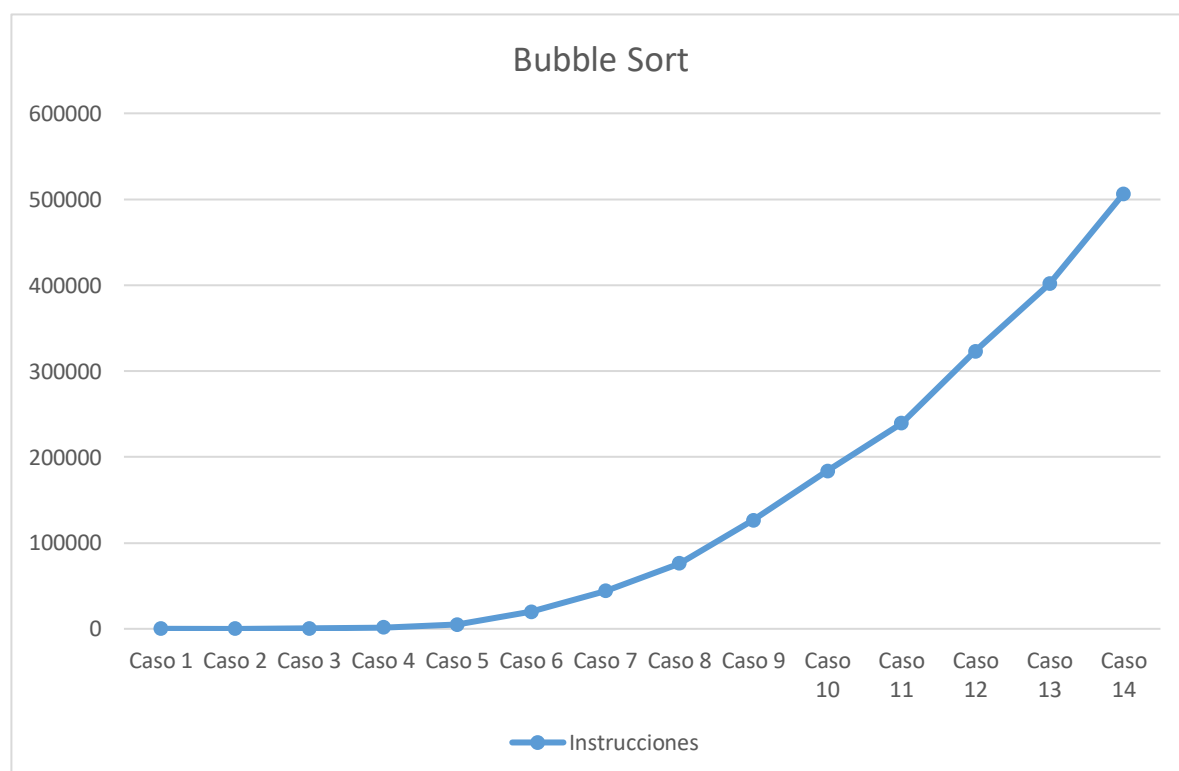
CASO	NUMERO DE ELEMENTOS	NUMERO DE INSTRUCCIONES
1	5	12
2	10	28
3	20	132
4	50	814
5	100	2564
6	200	10059
7	300	22330
8	400	38428
9	500	63620
10	600	92426
11	700	120289
12	800	162242
13	900	201668
14	1000	254182



## Tabla y grafico de Bubble Sort

Grado de complejidad:  $T(n) = n^2$

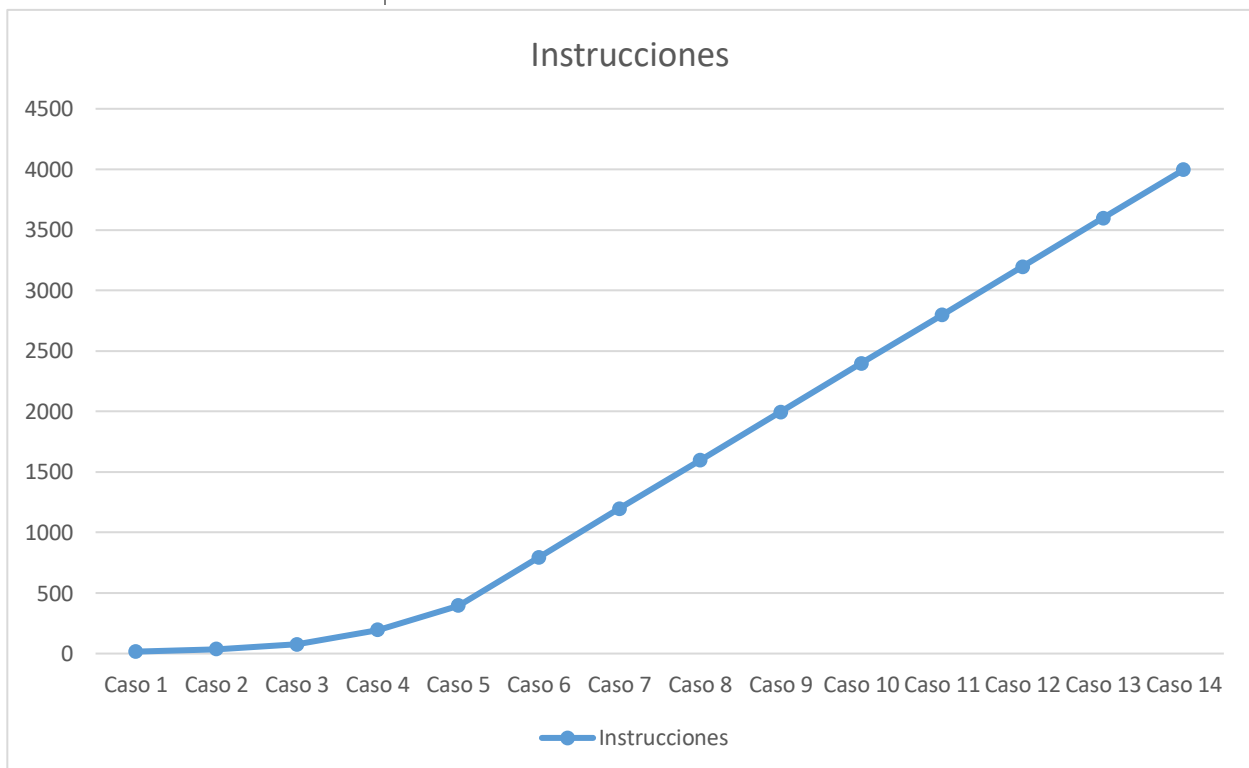
CASO	NUMERO DE ELEMENTOS	NUMERO DE INSTRUCCIONES
1	5	16
2	10	38
3	20	226
4	50	1530
5	100	4930
6	200	19720
7	300	44062
8	400	76058
9	500	126242
10	600	183654
11	700	239180
12	800	322886
13	900	401538
14	1000	506366



## Tabla y Grafico de Merge Sort

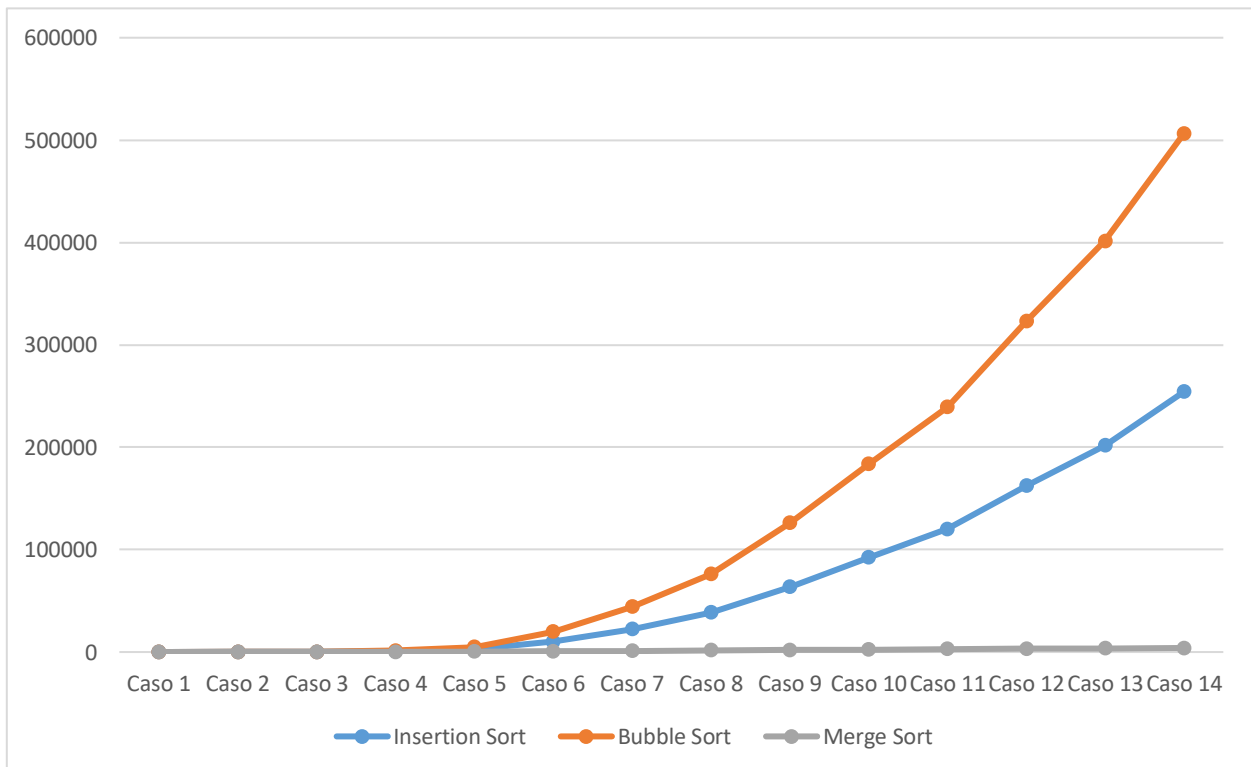
Grado de complejidad:  $T(n) = n \log n$

CASO	NUMERO DE ELEMENTOS	NUMERO DE INSTRUCCIONES
1	5	16
2	10	36
3	20	76
4	50	196
5	100	396
6	200	796
7	300	1196
8	400	1596
9	500	1996
10	600	2396
11	700	2796
12	800	3196
13	900	3596
14	1000	3996



## Comparación de algoritmos

CASO	NUMERO DE ELEMENTOS	INSTRUCCIONES EN INSERTION SORT	INSTRUCCIONES EN BUBBLE SORT	INTRUCCIONES EN MERGE SORT
1	5	12	16	16
2	10	28	38	36
3	20	132	226	76
4	50	814	1530	196
5	100	2564	4930	396
6	200	10059	19720	796
7	300	22330	44062	1196
8	400	38428	76058	1596
9	500	63620	126242	1996
10	600	92426	183654	2396
11	700	120289	239180	2796
12	800	162242	322886	3196
13	900	201668	401538	3596
14	1000	254182	506366	3996



## Conclusiones

La grafica comparativa y la tabla de datos muestran algo bastante interesante, y es que Bubble Sort, a pesar de tener complejidad de  $N^2$  realiza mas acciones a comparación de Insertion Sort y Merge Sort, pero es el que tiene el código mas sencillo y entendible.

Insertion Sort permanece en la media, aunque aun asi a comparación de Merge Sort realiza muchísimo mas instrucciones, aunque no tantas como Bubble Sort, su código no es tan complicado como el de Merge, ya que su código depende muchísimo de un par de for y while, por lo que, no es tan fácil de leer como el de Bubble pero tampoco es imposible

Ahora, Merge Sort parece ser el indicado para ordenar arrays puesto que con una array de 1000 elementos no realiza mas de 4000 instrucciones, y lo mas curioso es que esto sucede con cualquier array desordenada, no se eleva tanto, por lo que es el mejor algoritmo entre estos tres, lo malo es que su código es un poco extenso a comparación de los otros algoritmos, a mi me cuesta bastante trabajo comprenderlo como tal, se como funciona, pero no se exactamente que hace cada elemento de su código, aun asi es el mejor para ordenar. Para poder ver con claridad las instrucciones se usaron los mismos arrays para todos los algoritmos