

## Universidad Nacional Autónoma de México



## Facultad de Estudios Superiores Aragón

## Ingeniería en Computación

Estructura de Datos

Jesús Hernández Cabrera

**Axel Yahir Moreno Rodríguez** 



Turno Vespertino Grupo 1360

```
⊳ ∨ ф …
                                                J MergeSort.java ●
Estructura de Datos > Tareas > Tarea 12 > tarea12 > src > main > java > unam > mx > merge > 🔳 MergeSort.java > 🔀 MergeSort > 🕤 mezclar(int[], int[], int[])
                Run | Debug
public static void main(String[] args) {
    System.out.println(x:".-.-. MERGE --.-.");
    int[] datos = {38, 27, 43, 3, 9, 82, 10, 19, 50, 61};
                     ordenarPorMezcla(datos);
                     System.out.println("Arreglo ordenado: " + Arrays.toString(datos));
                public static void ordenarPorMezcla(int[] datos) {
                          int[] izquierda = Arrays.copyOfRange(datos, from:0, mitad);
                         int[] derecha = Arrays.copyOfRange(datos, mitad, datos.length);
                         System.out.println(Arrays.toString(izquierda) + " --- " + Arrays.toString(derecha));
                         ordenarPorMezcla(izquierda);
                         ordenarPorMezcla(derecha);
                         mezclar(datos, izquierda, derecha);
                     System.out.println("Regresando de recursión: " + Arrays.toString(datos));
                          J PilaMedia.iava

J MergeSort.iava ●

                                                                                                                                           D ~ CD ...
PotenciaRecursiva.iava
Estructura de Datos > Tareas > Tarea 12 > tarea12 > src > main > java > unam > mx > merge > 🤳 MergeSort,java > 🔀 MergeSort > 😙 mezclar(int[], int[], int[])
 4 public class MergeSort {
               public static void ordenarPorMezcla(int[] datos) {
                    System.out.println("Regresando de recursión: " + Arrays.toString(datos));
                private static void mezclar(int[] datos, int[] izquierda, int[] derecha) {
                    while (i < izquierda.length && d < derecha.length) {
                             datos[k++] = izquierda[i++];
                         } else {
                         datos[k++] = izquierda[i++];
                        datos[k++] = derecha[d++];
```

```
PROBLEMS 27 OUTPUT DEBUG CONSOLE TERMINAL PORTS
                                                                                                                                                                                        及 Run: MergeSort + ∨ 口 歯 ··· ∨ ×
PS C:\Users\axelx\Documents\FES\3er semestre\Estructura de Datos>
PS C:\Users\axelx\Documents\FES\3er semestre\Estructura de Datos' ; & c:; cd 'c:\Users\axelx\Documents\FES\3er semestre\Estructura de Datos'; &
                                                                                                                                                    '-cp' 'C:\Users\axelx\Documents\FES\3er semestre\Estr
.-.-.-. MERGE --.-.-
[38, 27, 43, 3, 9] --- [82, 10, 19, 50, 61]
[38, 27] --- [43, 3, 9]
 [38] --- [27]
Regresando de recursión: [38]
Regresando de recursión: [27]
 Regresando de recursión: [27, 38]
[43] --- [3, 9]
Regresando de recursión: [43]
 Regresando de recursión: [3]
Regresando de recursión: [9]
Regresando de recursión: [3, 9]
Regresando de recursión: [3, 9, 43]
Regresando de recursión: [3, 9, 27, 38, 43]
[82, 10] --- [19, 50, 61]
[82] --- [10]
Regresando de recursión: [82]
Regresando de recursión: [10]
Regresando de recursión: [10, 82]
 [19] --- [50, 61]
 Regresando de recursión: [19]
 [50] --- [61]
 Regresando de recursión: [50]
Regresando de recursión: [50]
Regresando de recursión: [61]
Regresando de recursión: [50, 61]
Regresando de recursión: [19, 50, 61]
Regresando de recursión: [10, 19, 50, 61, 82]
Regresando de recursión: [3, 9, 10, 19, 27, 38, 43, 50, 61, 82]
Arreglo ordenado: [3, 9, 10, 19, 27, 38, 43, 50, 61, 82]
PS C:\Users\axelx\Documents\FES\3er semestre\Estructura de Datos>
```

```
package unam.mx.merge;
import java.util.Arrays;
public class MergeSort {
        public static void main(String[] args) {
            System.out.println(".-.-. MERGE --.-.");
            int[] datos = {38, 27, 43, 3, 9, 82, 10, 19, 50, 61};
            ordenarPorMezcla(datos);
            System.out.println("Arreglo ordenado: " +
Arrays.toString(datos));
        public static void ordenarPorMezcla(int[] datos) {
            if (datos.length > 1) {
                int mitad = datos.length / 2;
                int[] izquierda = Arrays.copyOfRange(datos, 0, mitad);
                int[] derecha = Arrays.copyOfRange(datos, mitad,
datos.length);
                System.out.println(Arrays.toString(izquierda) + " --- " +
Arrays.toString(derecha));
                ordenarPorMezcla(izquierda);
                ordenarPorMezcla(derecha);
               mezclar(datos, izquierda, derecha);
```

```
System.out.println("Regresando de recursión: " +
Arrays.toString(datos));
        private static void mezclar(int[] datos, int[] izquierda, int[]
derecha) {
            int i = 0, d = 0, k = 0;
            while (i < izquierda.length && d < derecha.length) {</pre>
                 if (izquierda[i] < derecha[d]) {</pre>
                     datos[k++] = izquierda[i++];
                 } else {
                     datos[k++] = derecha[d++];
            while (i < izquierda.length) {</pre>
                 datos[k++] = izquierda[i++];
            while (d < derecha.length) {</pre>
                 datos[k++] = derecha[d++];
```