

# Verificación y validación

## Práctica inicial

```
1  ► public class MergeSort {
2      public static void sort(int arr[], int l, int r) {
3          if (l < r) {
4              int m = (l + r) / 2;
5
6              sort(arr, l, m);
7              sort(arr, m + 1, r);
8
9              merge(arr, l, m, r);
10         }
11     }

```

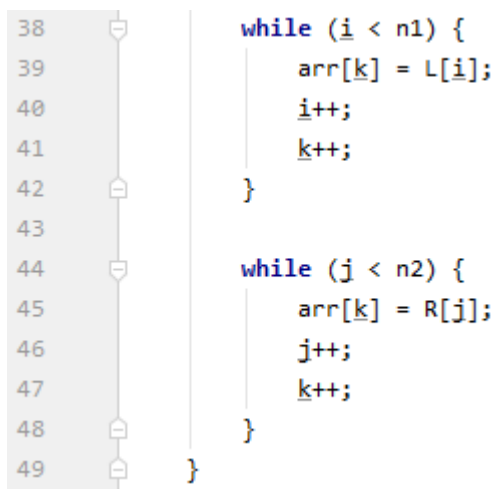
```
13  public static void merge(int arr[], int l, int m, int r) {
14      int n1 = m - l + 1;
15      int n2 = r - m;
16
17      int L[] = new int[n1];
18      int R[] = new int[n2];
19
20      for (int i = 0; i < n1; ++i)
21          L[i] = arr[l + i];
22      for (int j = 0; j < n2; ++j)
23          R[j] = arr[m + 1 + j];
24      int i = 0, j = 0;
25
26      int k = l;
27      while (i < n1 && j < n2) {
28          if (L[i] <= R[j]) {
29              arr[k] = L[i];
30              i++;
31          } else {
32              arr[k] = R[j];
33              j++;
34          }
35          k++;
36      }
37  }
```

Al ejecutar el programa con el debugger nos dimos cuenta de que el fallo estaba en la función “merge”. Lo que ocurría era que al comparar los subarrays L[] y R[] se introducía en el array ordenado (arr[]) el número menor, pero cuando el puntero llegaba al final de uno de los dos subarrays, no se añadía el resto de los números de la otra cadena.

Debido a esto, todos los números de la cadena se perdían, excepto el 0 (pues es el menor) y el 2 (ya que, al ser una cadena impar, cuando el algoritmo hace la última comparación, el subarray L[] tiene una longitud 3 y el R[] de 2; el número 2 está en la última posición de L[] y, por lo tanto, no se compara con los demás ceros de R[]), resultando la cadena final “00200”.

Lo que hemos hecho para solventar el error ha sido añadir el código necesario para que, cuando uno de los subarrays haya terminado de recorrerse (y por ello provocar la salida del bucle while de la línea 27), se termine de recorrer el otro subarray, manteniendo así todos los números originales.

El código, que hemos añadido dentro de la función “merge” después del bucle while, es:



```
38 while (i < n1) {  
39     arr[k] = L[i];  
40     i++;  
41     k++;  
42 }  
43  
44 while (j < n2) {  
45     arr[k] = R[j];  
46     j++;  
47     k++;  
48 }  
49 }
```

De este modo, se recorrerán los dos subarrays sin perder información.